System Design

R.L. Brown

REACTIVE COMPUTER SYSTEMS

DESIGN SPECIFICATION

Page _____
Date _____

Subject _____

TABLE OF CONTENTS

I.

RPG II

DESIGN OBJECTIVES

I.A.1 PURPOSE

The RPG II compiler is beinq developed to allow the Memorex B and C machines to compete with the IBM SYSTEM/3 and IBM 360/20. In order to accomplish this objective, the RPG II product will duolicate SYSTEM/3 RPG II as closely as reasonably possible. This will provide compatability with IBM SYSTEM/3 RPG II, as well as with a larqe number of IBM 360/20 RPG programs. RPG II will operate under OPSYS/1, and will be upward compatible with OPSYS/2 RPG II.

I.A.2 CONFIGURATION

The RPG II compiler and object proqram it qenerates will operate under OPSYS/1. Minimum confiquration will consist of operator's console, one disk, CPU, and 8K-byte memory for the compiler or object proqram (exclusive of OPSYS/1 support routines). In addition, it is necessary for the minimum confiquratior to be capable of supportinq OPSYS/1, including its Data Management, Linkaqe Editor, Loader, and Job Control Lansuaqe features. Thus 16K bytes of memory are reauired. The confiquration will also employ the I/O units required by the object programs to be used on the machine.

I.A.3 INTERFACE CONSIDERATIONS

RPG II will interface with the following areas of the operating system:

- o Linkage Editor and Overlay Loader
- o Data Management
- o Job Control
- o Sort
- o Operator Communications

Since RPG II will be developed in parallel with the operating system, RPG 11 will require a development system including an assembler and simulation and/or emulation of the computer and the above system facilities.

I.A.4 PERFORMANCE GOALS

The RPG II compiler will operate in an 8K-byte partition. Minimal RPG II object programs will be able to execute in an 8K-byte partition.

Compilation on a minimum configuration machine will be almost I/O bound. Performance on larger configurations will benefit from additional memory (up to some finite limit) and improved I/O facilities.

I.A.5 STANDARDS

IBM SYSTEM/3 RPG II will be used as the standard for validating the MEMOREX RPG II implementation.

I.A.6 TECHNICAL SUPPORT REQUIREMENTS

6.1 PUBLISHED DOCUMENTS

- RPG II Reference Manual (SRL)
- RPG II Programmer's Guide (SRL)
- RPG II Systems Manual, (PLY)
- RPG 11 Reference Card

6.2 RPG 11 UPDATES TO OTHER PUBLICATIONS

- System Description Manual (or equivalent)
- Messages and Completion Codes
- Terminal Services Manual (or equivalent)
- Job Control Language Reference Manual
- Linkage Editor Manual (or equivalent)

6.2 RPG 11 UPDATES TO OTHER PUBLICATIONS (Cont'd)

- o Job Control Lanquape Reference Card
- o Sales & Systems Planninq Guide for ^PSYS/1 (or eauivalent)

6.3 MEMOREX INTERNAL EDUCATION

In addition to standard announcement type clases, a one- to two-week RPG II Internals calss will be necessary in order for the MEMOREX Education and Product Test qrouos to prepare for their product support requirements.

6.4 INTERNAL DEVELOPMENT PROGRAMMER TRAINING

No known reauirements currently exist for internal development programmer trainiNq.

I.A.7 PRODUCT TEST REQUIREMENTS

A comprehensive set of well debuqqed, self-checkinq, and self-documenting test cases for which predefined results exist will be required. This set of test cases must insure regression testina for future product reports.

REPORT PROGRAM GENERATOR

I.B GENERAL DESIGN

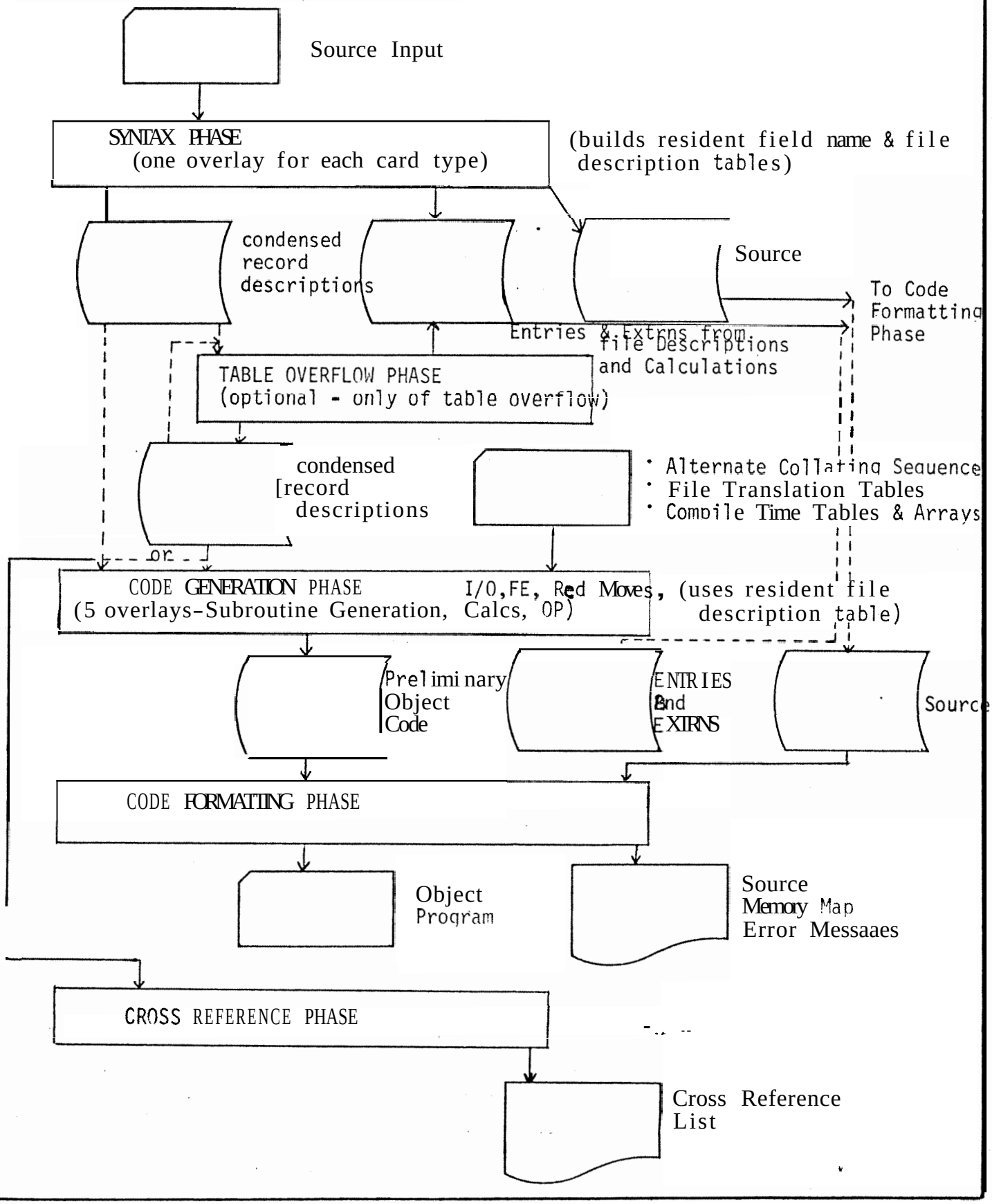
The followinq qeneral desian is intended to show the information flow throuqh the compiler and the general compiler orqanization. Since the compiler should always be Input/Output bound, one can get a good idea of the speed of the compiler.

There is also a brief description of each separate module with a rough size estimate.

I.B **GENERAL DESIGM** (Cont'd)

The total compiler should be about 15,000 instructions, plus 2,000 instructions of object time subroutines, plus error messages, plus OPSYS/1 Data Management, plus table and buffer sizes.

Subject GENERAL COMPILER FLOW



I.B.1 The Compiler

One program will be resident with all phases (sections of it may be overlaid). This is the:

EXECUTIVE PPOGRAM (1000 bytes)

- o This will contain the COMMON variable used by all programs.
  - • Fixed Table Addresses & limit
  - • Control Section Address Counters
  - • Control Card Information
  - • Program Data used in all phases
- o It will also contain all I/O interfaces
- o Table lookup routine
- o Phase Overlay Control
- o Blocking/Deblocking

— The rest of the compiler will be broken into four main phases each of which may consist of several overlays.

I.B.1.1 Syntax Checking Phases

a. Header Card Scan (2000 bytes)
'Moves information from control card to common area
'Gets System information (date, Time)
'Opens & assigns RPG System files
'Any other initialization/housekeeping

b. File Description Scan (3000 bytes)
'Error checks file description card
'Builds resident file & file disc table
'Write out encoded file description record

c. File Extension Scan (2000 bytes)
'Builds RAF table entries in file description table
'Writes table lookup/array coded records (in final format)
'Error checkin
'Calculates field name table size 6 puts in PAGE, PAGE 1, PAGE 2

d. Line Counter Scan (1000 bytes)
'Writes line counter table on disc
'Error checking

e. Input Specifications Scan (4000 bytes)
'Initiates buildinq field name & description table
'Writes record, identification & field description records
'Error checking

f. Calculation Specifications Scan (4000 bytes)
'Write calculation coded record on disk
'Error checkinq
'Adds to field name & description tables

g. Output Specifications Scan (4000 bytes)
'Writes record ID & field description records
'Transforms edit picture to edit control characters
•Error checking

I.B.l.2 Table Overflow Phase (Optional) (1000 bytes)

If the resident field table overflows durina the snecification phase processinq continues but no more entries are added to the table, with all references to entries in the table resolved. A special pass then takes place, which takes the output of the first pass and builds table entries for those fields not entered in the first pass - this continues until all entries have been entered in the table once.

I.B.1.3 Code Generation Phase

a. Object Control Proqram Generation* (2000 bytes)

'Decides which parts of object control program are necessary and include them in object proqram

'Generates part of object time common region

* The actual object time subroutines include (7000 bytes)

1. Basic control program
2. Look ahead routine
3. Record identification
4. Field moving input
5. Field moving output
6. Level break test
7. Matchinq record
8. Record selection
9. Table lookup
10. Some calculations (Divide, multiply, subroutine linkaqe)
11. Output record selection

b. Input/Output aeneration (6000 bytes)

'Generates interface between loqical I/O and compiler

'Decides which loqical I/O routines are to be called

'Processes compile time tables & puts their address in control table

'Processes file extensions & line counter

c. File Extension - Alt Collatina Seq.-Tables-Line Counter

'Processes file extension records

'Reads and qenerates alternate collating seauence tables

'Reads and senerates tables/arrays

'Processes line counter

d. Record Handling Generator (5000 bytes)

'Generates record identification table

'Generates code to move data from input to work area includinq conversion from binary or unpacked decimal.

e. Calculation Generator (6000 bytes)
'Eliminate duplicate indicator testing
'Generates indicator testinq & settina
'Generates claculation code
'Selects needed subroutines

f. Output Generation (5000 bytes)
'Output record selection table
'Generate field moving/editing code

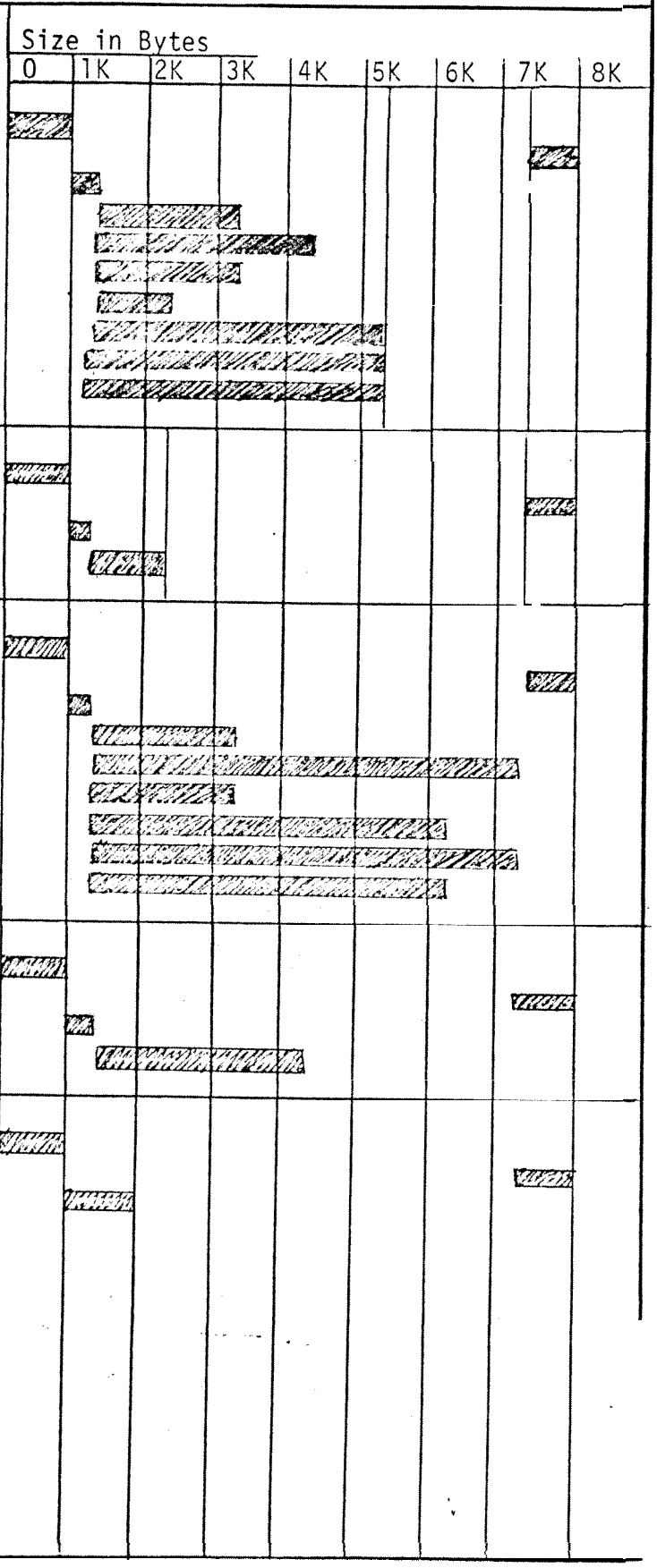
I.B.1.4 Code Formatter Phase (3000 bytes & messages)

'Produce object disk
'List error messages
'Memory & indicator map
'Debugging code dump
'Load and make part of object program compile time tables/arrays

I.B. 1.5 Cross Reference Phase (1000 bytes)
'Cross Reference list

Subject COMPILER CORE UTILIZATION

Phase	
A. Syntax Phase	
1. Compiler Executive	1,000
2. Buffers	730
3. Fixed Table Snace	256
4. a. Header Scan	2,000
b. File Description	3,000
c. File Extension	2,000
d. Line Counter	1,000
e. Input Specifications	4,000
f. Calculation Specifications	4,000
g. Output Specifications	4,000
B. Table Overlay Phase	
1. Compiler Executive	1,000
2. ,Buffers	730
3. 'Fixed Table Space	256
4. Table Overlay Proqram	1,000
C. Code Generation Phase	
1. Compiler Executive	1,000
2. Buffers	560
3. Fixed Table Space	256
4. a. Control Proqram Generator	2,000
b. Input/Output Generator	6,000
c. File Extension-Line Counter	2,000
d. Input Record Handling	5,000
e. Calculations	6,000
f. Output Record Handling	5,000
D. Code Formatter	
1. Compiler Executive	1,000
2. Buffers	994
3. Fixed Table Space	256
4. Code Formatter	3,000
E. Cross Reference	
1. Compiler Executive	1,000
2. Buffers	470
3. Cross Reference Program	1,000



I.C Memory Overflow Contingencies

1. In a Syntax Checking Phase

   These will all be fairly small programs with little likelihood of exceeding the size estimates given. If one or two are larger than expected, then the possibility of moving some error checking to the generator phase will be looked at (the generator phase must have room available), or the available table space will be reduced for field names.

2. Field Name Table Overflow

   The Field Name and Description table occupies all the space between the largest syntax scan overlay and the bottom of the file description table. If all the available entries are filled and there are more left to put in, the table overflow phase is called after all the syntax overlays are processed.

   The table overflow phase reads the condensed description records outputted by the Syntax checking phase and checks field, tag and subroutine names not processed by the syntax phase. Undefined and duplicate names are diagnosed, and a new condensed description file is written.

   The table overflow phase reprocesses the new condensed description file as many times as necessary.

3. Code Generation Programs

   The Calculation Generator will have the most difficult fit in core. To cover the possibility that it may not fit in core, it will be coded in isolated subroutines corresponding to Calculation operator types. If it will not fit as a whole in one overlay, the least used options can be made into an overlay to be ping-ponged with the more popular options.

II.

		COMPILER MODULES	RUN-TIME MODULES
F$	File Control Table, FCT		x
S$	Target Source Table		x
T$	Table / Array Control Table		x
W$	Run Time Common Area	x	x
X$	Condition Codes		
Y$	File Name Table		
Z$	Field Name Table		

II. RPG Coding Conventions

A. All entry points and program names begin with $RG (this does not apply to regular labels in a program).

B. The compiler communication region will be in common (DCOM), and all fields in it will begin with X$. .

All programs will use the same common definitions which will be stored in the system macro library and referenced by the macro name.

X$COM

C. Linkage within compiler will use registers as follows:

	Register	Use
	0	Work register, not saved
C	1	Work register, not saved
	2	Must be saved if used
	3	Must be saved if used
	4	Must be saved if used
	5	Must be saved if used
RP	6	Parameter list pointer
RR	7	Return address

Note: Registers 0 and 7 will become available when the final machine does (1972). At that time we can switch to using register 7 as the return address and 0-1 will be work registers

D. Linkage registers 7 and 6 will be referred to as RR and RP.

E. All linkage to Compiler Executive Subroutines (e.g. $RGLUP, $RGIND, $RGPUT) will be with the LINK macro.

operator	operand1	operand 2 (optional)
LINK	$RGLUP /	PARLST

operand 1 is the subroutine name, operand 2 is the'pointer to the parametor list

F. All Input/Output records will be addressed symbolically (Instead of using a displacement to a field it will have a name). All names begin with the character at the beginning of the record followed by a $.

For example the file number in the Output Record Identification record could be called:

O$FILN

III.

III. A RPG Indicator Processing

Indicators can be both defined and referenced. It will be one of the error checking functions of the compiler to diagnosed when a referenced indicator has not been defined (has never had possibility of being set on by being predefined, a result or record indicator). For this purpose the $RGIND is provided and must be used to process indicators, for it keeps two bit maps of referenced and defined indicators.

Only those indicators used (with the possible exception of H0-H9, L0-L9, LR, MR, 0A-0V and 1P) will be assigned locations in the object program. Those locations will not be assigned until the code formatter phase. Internally the indicators will be assigned as follows:

Hex	Indicator
00	Not used
01-63	01-99
64-6D	H0-H9
6E-77	L0-L9
78	LR
79	MR
7A-81	0A-0V
82	1P
83-8A	U1-U8

III. B Subroutine name: $RGLUP

Functions: 1) To look up the Field Definition Table and verify if a given name occurs there;

2) To stash an entry in the table if requested and if no duplicate is found.

Inputs: RG points to the parameter list that contains the following information:

Byte

0-1 request coding one of the following values:

0 - only look-up

2 - look-up and stash

2-3 - pointer to the entry

4-7 - used to return information

Outputs: The following information is placed in the parameter list starting at byte 4:

4-5 return code; one of the following valves:

0 - no duplicate entry/name not found

2 - duplicate entry/name found

4 - table overflow (only used for stash request)

6-7 - points to either the old entry located in the table or to the location where the new entry has been stored.

Notes: 1) The name field is contained in bytes 0-4 of the entry.
2) The entry is assumed to be 12 bytes long. The subroutine does not store array appendages with array entries.
3) If a name already occurs in the table, code 2 is returned together with the address of the old entry. In this case it does not matter if the request is look-up or stash.
4) If entry is stashed, bit 1 of Z$SWT2 is set indicating entry has been made in table.

Name of subroutine: $RGCKS

Function: to scan variable length string(s) and check for invalid EBCDIC characters.

Input: RG points to the parameter list containing the following information:

Byte
0-1 pointer to beginning of field to scan
2-3 length of field in bytes
4 - used to return information

Outputs: The following information will be placed in the parameter list starting at byte 4:

4-5 return code; one of the following values:

0 - filed blank
2 - alphameric string starting with an alphabetic character
4 - field starts with a quote, +, -, or numeric
6 - leading blank or imbedded blank found; no illegal characters
8 - illegal character found

6-7 number of synificant characters in the string.

8-11 the same information as in bytes 4-7 for the second string if there is one. (Note: if there are two strings, they must be separated by a comma).

Description: The string is scanned and each character is inspected. If an' illegal EBCDIC character is found (other than A-Z, 0-9, #, $ or @) the scan is terminated and the return code is set to -2. If a blank is found, the rest of the field is checked. If there is a character other than blank following an initial blank, the return code is set to -1. If there are two strings on the field, separated by a comma, both strings are checked. If a string starts with a quote, no further check is made on the field.

Number of synificant characters will be returned only with return code 1 or 2.

Auxiliary Subroutine: $RGCKA

Function: to check 1 character for Alphabetic A-Z,#,$,@

Input: R1 = character to be checked

Output: R0 = o valid character
R0 = 2 invalid

Subroutine name: $RGIND

Functions: 1) To check an EBCDIC indicator and convert it to the appropriate binary code.

2) To set the pertinent list for the indicator in the definition map or the reference map.

Inputs: RG points to the parameter list containing the following information.

Byte

0-1 indicator with one of the following values:

0 - reference
1 - definition

2-3 pointer to two-character field containing indicator in EBCDIC

4-7 used to return information

Outputs: the following information will be placed in the parameter list starting at byte 4:

Byte

4-5 return code; one of the following values:

0 - blank
2 - valid indicator
4 - invalid indicator

6-7 binary code for the indicator

Description: $RGIND checks a 2-character field to determine if it contains a valid indicator. If it does the pertinent binary code is obtained and used as an index to set on a bit in either the indicator reference map or the indicator definition map.

Note: if the indicator is expressed is a 1-digit number, it must be right adjusted'in the field.

Subroutine name - $RGCKN

Function: to verify a variable length EBCDIC numeric string and to convert it to binary.

Inputs: RG points to the parameter list containing the following information:

Byte	
0-1	pointer to beginning of string
2-3	size of field in bytes
4-7	used to return information

Outputs: the following information is placed in the parameter list starting at byte 4:

Byte	
4-5	return code; one of the following values: 0 - field is blank 2 - valid number 4 - not a valid number
6-7	numeric value of the string converted to binary.

Notes: 1) sign or decimal point are not allowed.
2) the string must be right adjusted in the field.

Subroutine name - $RGLFN

Function: To look up the File Description Table for a given file name.

Inputs: RG points to the parameter list that contains the following information

Byte

0-1 pointer to 1-8 character file name (left adjusted, padded with EBCDIC blanks)

2-3 Unused

4-7 used to return information

Outputs: The following information is placed in the parameter list starting at byte 4:

4-5 return code; one of the following values:

0 - name not found

2 - name found

6-7 pointer to entry if found.

Subroutine name - $RGCND

Function: To condense a string of EBCDIC characters by stripping off the high order 2 bits and packing the reduced six-bit characters next to each other.

Inputs: RG points to the parameter list that contains the following information:

Byte	
0-1	pointer to the EBCDIC string
2-3	number of characters in the string
4-5	pointer to work area where the condensed characters are to be placed (calling routine must initialize this area to zero)

Outputs: Condensed string in the specified work area.

Note: The input characters are not checked for validity.

SUBROUTINE NAME: $RGETC

FUNCTION: To read a card from card reader.

INPUT: RG points to a parameter list in which the following information will be placed.

OUTPUT: Byte

0-1 Return code; one of the following
0 - successful
1 - end of file

2-3 Buffer address

DESCRIPTION: A card is read from the card reader into a physical buffer whose address is placed in bytes 2-3 of the parameter list.

SUBROUTINE NAME: $RGDMP

FUNCTION: To make readable dump of resident tables

INPUT: RG contains one of the following:

zero - Dump all of file table
Dump all of field table

non-zero - pointer to field name table entry to dump

OUTPUT: Output goes to printer - dump of tables

DESCRIPTION: The file description table will be printed as follows:

cols	1-2	hex file number
	4-11	file name
	14-17	switches (hex)
	19-20	file type (hex)
	22-23	linked file (hex)
	25-28	pointer (hex)
	30-31	seg limit on overflow in (hex)
	33-34	high match red level (hex)
	36-39	record length (hex)

The field description table will be printed as follows:

cols	1-4	table address (hex)
	6-11	field name (uncondensed)
	13-18	field type and switches in hex
	20-23	blank or zero ind and field size in hex
	25-28	storage address in hex
	30-33	hex FF and table entry in hex
	35-38	number of table entries (hex)
	40-43	storage address (hex)

SUBROUTINE NAME: $RGET

FUNCTION: TO READ A RECORD FROM DISC.

INPUT: RP points to parameter list which contains following information:

	Byte	
	0-1	FILE IDENTIFICATION NO. 1 - Condensed Record Descriptions 2 - Source Records 3 - Entries and Extrns 4 - Condensed Record Descriptions (pingponged with #1) 5 - Preliminary Object Code
	2-3	RECORD ADDRESS
OUTPUT:	4-5	RECORD LENGTH IN BYTES
	6-7	RETURN CODE 0 - Successful 1 - EOF 2 - Unsuccessful

SUBROUTINE NAME: $RGPUT

FUNCTION: WRITE A RECORD TO SPECIFIED DISC FILE

INPUT: RP points to parameter list that contains the following information:

Byte	
0-1	FILE IDENTIFICATION NO..
	1 - Condensed Record Descriptions
	2 - Source Records
	3 - Entries and Extrns
	4 - Condensed Record Descriptions (pingponged with #1)
	5 - Preliminary Object Code
2-3	RECORD LENGTH IN BYTES
4-5	RECORD ADDRESS

SUBROUTINE NAME: $RGING

FUNCTIONS:
1. Return relative address of indicator in Object code group 21 (see page VI(7)).
2. Return status showing whether indicator is undefined or unreferenced.

INPUT: RG points to the parameter list that contains the following information

Byte

0-1 Indicator (as defined on page III(0)).

OUTPUT: The following information is placed in the parameter list starting at byte 2:

2-3 Return code; one of the following values

0 - Indicator defined and referenced
2 - Indicator not defined
4 - Indicator not referenced

4-5 Relative address of indicator in code group 21.

DESCRIPTION: $RGING checks the defined and referenced indicator bit maps (X$INDF and X$INRF) to determine if the indicator had been defined and referenced. It then OR's the two indicator maps together and computes the number of bits on before it gets to the one currently being converted and that number is the relative address in the indicator group (21) for the current indicator.

IV.

IV. A. Resident File Name and Description Table

	Column	Field	Length
Y$NAME	0	File Name	8
Y$SWT1&2	8	Switches	2
Y$TYPE	10	File Type	1
Y$LINK	11	Linked File (RAF or ADDROUT)	1
Y$RID	12	Record ID or Line Counter Chain	2
Y$SEQL	14	Sequence Limit or Overflow Indicator	1
Y$MTCH	15	High Matching Record Level	1
Y$RCDL	16	Record Length	2
			18 bytes

This table is built down from the top of core. It is created by the File Description Scan, added to by the File Extension and Input Specification Scans and used by the Calculation Scan, Output Scan and Input/Output Generator. Some fields contain different information depending on whether it is an input or an output file.

Field Descriptions

	Columns	Description
Y$NAME	0-7	File Name - same as it appears on File Description Form
Y$SWT1	8-9	Switches
		Column 8 Bit 0 = 1 Primary File
		1 = 1 Secondary File
		2 = 1 Chained
		3 = 1 RAF/ADDROUT
		4 = 1 Table/Array
		5 = 1 Demand
		6 = 1 Print File
		7 = 1 Variable Length File
Y$SWT2		Column 9 Bit 0 = 1 Ascending Seauence
		1 = 1 Descending Seauence
		2 = 1 File Extensions necessary
		3 = 1 Line Counter Neeessary
		4 = 1 New record will be added to file
		5 = 1 Alphanumeric keys
		6 = 1 Packed decimal keys
		7 = 1 Card file

Y$TYPE	10	File type . 0 = Input 2 = Output 4 = Update 6 = Display 8 = (Combined) not implemented
Y$LINK	11	Linked File (for RAF or ADDROUT files) File Table Entry Number (by order of entry into table) that this file is linked to by the File Extension Spec.
Y$RID	12-13	Record Identification pointer (for Input files) or line counter pointer (for output files) - a minus one indicates not used - initialize to minus one This entry is filled in by the Input Specifications Scan or the Line Counter Scan, with the relative address of the record identification table/line counter table for this file This is used by the Input/Output generator to help build the File Control Table
Y$SEQL	14	For DSPLY files is set to zero when referenced, key field length for indexed files or overflow indicator. The overflow indicator and key field length both come from the file description scan.
Y$MTCH	15	High Matching Record level - from Input specifications.
Y$RCDL	16-17	Record Length - for checking maximum record positions in input and output specifications

Y$

FILE NAME DESCRIPTION TABLE

	FILE NAME (8 BYTES)	
+8	SWT 1	SWT
+10	TYPE	LINK
+12	RID	
+14	SEQL	MTCH
+16	RCDL	

IV.B. Resident Field Name and Description Table

	Column	Field	Lenqth
Z$NAME	0	Field Name (compressed)	5
Z$TYPE	5	Field Type	1
Z$SWT1	6	Switches	1
Z$SWT2	7	Switches	1
Z$BZI	8	Blank or Zero Indicator	1
Z$SIZE	9	Field Size	1
Z$ADDR	10	Storage Address	2
			12 bytes

The Field Name and Description Table is a randomly addressable table occupying all the area available below the File Table. It is built and referenced by the Input Specification, File Extension and Calculation Scans, and referenced by the Output Scan. If this table fills up before all new field names are processed then a special field Table overflow routine is used.

Field Descriptions

	Columns	Description
Z$NAME	0-4	Field Name (condensed) The field name is packed 6 bits to a character (total of 36 bits or 4 1/2 bytes) by simply removina the hiqh order 2 bits of each character (e.q. A, which in binary is 11000001 becomes simply 000001)
Z$TYPE	5	Field Type

Bit 0=1 Numeric
1=1 Alpha
2=1 Taq name
3=1 Subroutine Name
4=1 Code formatter is not to create adcon

Z$SWT1 6 Switches

Bit 0 = 1 Table Name
1 = 1 Array Name
2 = 1 Table in Ascending Order
3 = 1 Table in Descending Order
4 = 1 Extrn
5 = 1 Field used as array index (must have zero decimal positions)
6 = 1 Input Data type - packed decimal
7 = 1 Input Data type - binary

Z$SWT2 7 Switches

Bit 0 = 1 Redefinable field (PAGE, PAGE1, PAGE2)
These 3 entries are made by FE scan in field name table-but may be redefined by input specs or calc specs.
1 = 1 Entry has been made in table (set when a new field name is put in field name table)
2 = 1 Name is defined
3 = 1 Name is referenced
4 = 1 RLABL
5 = 1 RLABL indicator (INXX) Indicator is in storage address (right justified)
6 = 1 Non Redefinable and Non Alterable field (UDATE, UMONTH, UDAY, UYEAR or Lookahead)
7 = 1 Page Redefined.

Z$BZI 8 Blank or Zero Indicator

For fields from input specifications or calculation specifications that have a blank or zero indicator associated with them, that indicator number is put here. (Code is generated to cause this indicator to he reskt for a blank after of this field name on output)

Z$SIZE 9 Field Size

Alphameric - Lenath of field (1-255)
Numeric - Bits 0-3 Decimal Positions (0-9)
Bits 4-7 Number of diaits in field (1-15)
(for packed decimal or binary fields this is not the same as field size)

To compute the number of bytes the field will occupy in the work area from this field size use the following:
for Alphameric - field size
for Numeric - field size/2+1

Z$ADDR 10-11 Storage Address
Relative address of field in work area (where numeric fields are in packed format) This address is assigned by the scan phase overlays
If RLABL indicator byte 11 holds indicator number
If EXIT orULABL name contains EXTRN ordinal*or Label processing or SPECIAL processing
or Table Chain for Table and/or Array fields
If Table/Array, bytes 10-11 point to a 6 byte table entry with the following format.

	Column	Description
Z$EXT	0	binary 255 (FF) identifies this as a table/array extension
Z$TABN	1	Table file entry number (1-60) consecutively assigned in order of definition
Z$ENTS	2-3	Number of table entries (maximum index value)
Z$THLD	4-5	Storage Address For tables this points to last found table element hold area - The hold area if followed by table proper For Arrays this points to beginning of array. For RLABL, EXIT, ULABL or file processing EXTRN see Z$ADDR description

*1 EXTRN Ordinal - for EXIT or ULABL to get indirect address of field use ordinal *2 as displacement in code group (CSECT) 02.

TAG entry:

	Byte	
Z$CTL	6	Control level on which TAG is defined

6E - 77	L0-L9
78	LR
FF	SR

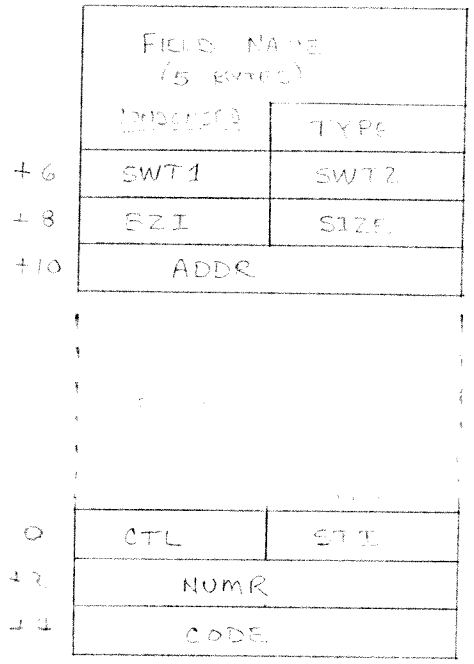
	Byte	
Z$STI	7	Status indicator Bit 1 = 1 entry made in the table Bit 2 = 1 label defined Bit 3 = 1 label referenced
Z$NUMR	8-9	Number of references to this TAG
Z$CODE	10-11	TAG ordinal

Subroutine entry:

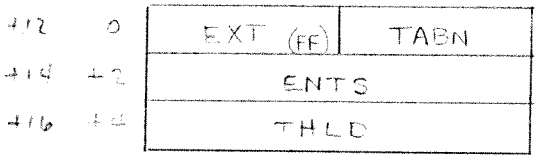
	Byte	
Z$CTL	6	SR
Z$STI	7	Status indicator Bit 1 = 1 entry made in the table Bit 2 = 1 label defined Bit 3 = 1 label referenced
Z$NUMR	8-9	Number of references to this name.
Z$CODE	10-11	Subroutine identification code.

FIELD NAME DESCRIPTION TABLE (1)

Z$



TAG/SUBROUTINE EXTENTION



ARRAY/TABLE EXTENTION

V.

V. DISK RECORDS

The records kept on Disk are encoded forms of the input records, written by the scans, and read by the table overflow phase, code generating phases and cross reference phase.

Each record beqins with information identifying the record type and the associated input line number as follows:

Byte 0 - Record Type

- X = Error Record
- F = File Description
- T = Table and Array Description
- L = Line Counter
- I = Input Record Identification
- N = Input Field Description
- C = Calculations
- O = Output Record Identification
- U = Output Field Descriptions
- G = Generated Code
- E = Entries and Extrns

Byte 1 Blank not used
Bytes 2-3 = Input Line Number

V.A. Field Descriptions in Intermediate Records

Applies to Record types T, N, C and U

The condensed field descriptions in intermediate records have exactly the same 12 byte format as the Resident Field Table except for the case of Tables/Arrays, which is:

TABLE/ARRAY Intermediate record condensed field description - 16 bytes

Bytes	Description
0-9	Same as Resident Field Table
10	FF
11	Table/Array entry number (1-60)
12-13	Number of table entries
14-15	Storage address

For tables this points to last found table element hold area. The hold area is followed by the table proper. For arrays this points to beginning of array.

X - Error Record

Error records are written for all errors, whether they have been printed out when found or not.

Columns	Description
0	X - Error record
2-3	Record number
4-6	EBCDIC PHASE ID
7	0=warning, 1=serious, 2=disastrous (Binary 0, 1 or 2)
8-9	Phase Number in binary (see Schedule for phase number)
10-11	Error number
12-n	variable field data - (defined as needed)

CLG 13

F - File Description

The file descriptions are created by the File Description Scan and used by the 1/0 Generator.

Columns	Field		
0	F - File descriptions		
2-3	Record number		
4	File type	1=0 Input file 0=2 Output file U=4 Update file D=6 Display file C=8 Combined file	
5	File designation	P=0 Primary S=1 Secondary C=2 Chained R=3 Record address T=4 Table or array D=5 Demand Ø=6 No designation	
6	End of file	0= No end 1= E specified	
7	Sequence	0= No sequence 1= Ascending sequence 2= Descending sequence	
8-9	Block length in binary	1 - 32,767 Bytes)	maximum depends
10-11	Record length in binary	1 - 32,767 Bytes	on device
12	File format	0= Fixed length records 1= variable length records	
13	Mode of processing	L=0 Sequential within limits R=1 Random Ø=2 Sequential	
14-15	Langth of Key or record 'address field in binary (O-?)		
16	Record address type	A=0 Record keys are used I=1 ADDROUT processing Ø=2 other K=3 360/20 option (Assume A)	
17	File organiration	1=0 Indexed file T=11 ADDROUT file 1-9 Additional I/O areas Ø=10 Use one 1/0 area D=12 360/20 option (Assume 1)	

18 Overflow indicator
- Blank=0
- OA = 1
- OB = 2
- :
- OG = 7
- OV = 8

19 Extension code
- Ø=0 No extensions
- E=1 File extensions
- L=2 Line counter used

20-21 Key field starting location in binary

22 Device
- 0 - PRINTER
- 1 - READ
- 2 - TAPE
- 3 - DISC
- 4 - CONSOLE
- 5 - PUNCH
- 6 - SPECIAL

23 Labels
- Ø=0 No labels
- S=1 Standard labels
- E=2 Standard labels followed by user labels
- N=3 Non standard labels

24-25 ORDINAL of label exit from file description card or user's written subroutine name (if device = special)

26 File addition/unordered
- Ø=0
- A=1 New records will be added to file
- U=2 Load in unordered sequence

27 Tape rewind
- Ø=0 Rewind only
- N=1 No rewind
- U=2 Rewind and unload

28-29 Core index - number of bytes (in binary) reserved for core index

30 File condition indicator or zero
- blank = 0
- U1 =1
- U2 =2
- U3 =3
- •
- U8 =8

T - Table and Array Descriptions

These records are built by the file extension scan and used by the Input-Output generator to generate table lookup and control tables.

Columns	Field
0	T - table and array descriptions
1	Blank (Bit 7=1; do not generate code)
2-3	Record number
4	From file name table entry (1-20) (0=compile or execution time)
5	To file name table entry (1-20) (0=no run time output)
6-7	Entries per record in binary
8-9	Entry length in binary
10-25	Condensed field description entry
26	Switches 0 = execution time array 1 = pre-execution time table/array 2 = compile time table/array
27	Not used
28-29	Alternating table entry length in binary A zero entry here indicates end of record (0=no condensed field description follows)
30-45	Condensed Field description entry

L - Line Counter Descriptions

Built by Line Counter Scan and used by the Input/Output generator

0	L = Line Counter Description
2-3	Record number
4-5	File name table entry (relative pointer)
6-7	Line Number
8-9	Channel Number (1=top of page, 12=overflow line) 14=lines per page

Repeat 6-9 as often as necessary. A zero entry signified end of record.

I - Input Record Identification

These records are created by the Input Specification Scan from the record identification portion of the Input Specifications. They are used by the Record Handling Generator.

0	I = Input record identification (Bit 7=1 DO NOT generate code)
1	Blank
2-3	Record number
4	AND-0, OR=2, End of Alphas-set for first numeric equiv. to AND, End of Table=6, Trailer=8
5	Option - 0=2, blank=0
6	Stacker select
7	Indicator associated with record
8	Number 0=Not applicable 2=Only one record permissable 4=More than one record OK
9	Not 0=blank 2=N
10	Portion of character to test 0=character 2=numeric 4=zone 6=no character to test
11	Character or portion of to be tested
12-13	Displacement within record of character to be tested
14-15	File number

N - Input Field Descriptions

These records are built by the Input Specifications Scan from the field description portion of the records, and are used by the Record Handling Generator.

0	N-Input field descriptions
1	blank
2-3	Record number
4-5	Displacement of field within record
6-7	Input field length
8	Level (0-9)
9	Matching record number (0-9)
10	Field record relation-indicator associated with field
11	Plus indicator
12	Mnus indicator
13	Blank or zero indicator
14	Switches
	BIT 0=1 Packed Field
	1=1 Binary field
15	Field type
	2=Regular field (12 byte entry follows)
	4=Table or Array (16 byte entry follows)
16-n	Condensed field description
	n+1 zero
n+2	Field type
	0=end of record (no entry follows)
	2=Index is regular field (12 byte entry follows)
	4=Index is table name (16 byte entry follows)
	6=Index is constant (2 byte entry follows)
n+3	Condensed field description or 2 byte binary constant

C - Calculation Description Record

Built during calculation specification scan and used by the following routines:

1) Table Overflow phase
2) Code Generator

0	C for record type
1	blank not used
2-3	statement number
4	control level indicator 0=detail calculation 6E=L0 6F=LI . . . . . . 77=L9 78=LR FF=SR
5	Switch 0 = 1 operation record 1 = 1 indicator record 2 = 1 do not generate code 3 = 1 half adjust

Indicator records: only one record is written for a set if conditioning indicators even if AN/OR statements are used.

6-n	8 byte fields for groups of 1-3 indicators as follows:
0	Not switch for indicator 1 (bit 0=1 for NOT)
1	Indicator 1
2	Mot switch for indicator 2
3	Indicator 2
4	Not switch far indicator 3
5	Indicator 3
6-7	AN/OR/end of record 0 - end of record X'FD' - AND X'FE' - OR

Calculation records: one to one correspondence between statements and records.

6 Operation code.

1-ADD
2-BEGSR
3-BITOF
4-BITON
5-CHAIN
6-COMP
7-DEBUG
8-DIV
9-DSPLY
10-ENDSR
11-EXCPT
12-EXIT
13-EXSR
14-FORCE
15-GOTO
16-LOKUP
17-MHHZO
18-MHLZO
19-MLHZO
20-MLLZO
21-MOVE
22-MOVEL
23-MULT
24-MVR
25-READ
26-RLABL
27-SETOF
28-SETON
29-SQRT
30-SUB

31-TAG
32-TESTB
33-TESTZ
34-ULABL
35-XFOOT
36-Z-ADD
37-Z-SUB

7-9 Result indicators - high, low, equal
10-1315 Program control (switches for moving error checking to generate or)
16 14=n Variable length subfields as follows:
0 Indicator

bit 0=1 literal
1=1 file name
2=1 other
3=1 factor 1
4=1 factor 2
5=1 result
6=1 index for one of the above
7=1 error in name

If all bits are zero, this is the end of the record

1 unused

Literal

2 indicator

bit 0=1 alphameric
1=1 packed decimal
2=1 bit map (1 byte) for BITON/BITOF

3 length of literal

4 number of digits

5 number of decimal positions

6-13 literal (left-adjusted for alpha & 8-byte packed decimal for numeric.

File name

2-3 relative location of pertinent entry in the resident file description table.

Field, table, array, tag or subroutine name

2-13 (2-19 for an array) - dunlicate of the related entry in the resident field description table.

0 - Output Record Identification

This is built by the output description scan and used by the Output Handling Generator

Columns Field

0	0-Output Record Identification
1	7 Do not generate code
2-3	Record number
4	AND/OR relationship with previous record 0=1 OR 1=1 ADD 2=print file 3=card file 4=variable length record 5=file name entered on output spec.
5	Stacker select/Fetched Overflow for stacker select is hopper number for fetched overflow = 15
6	Space before
7	Space after
8	Skip before
9	Skip after
10-11	Record length
12-13	File number
14	Group 1=First page 2=Headers and details not conditioned by overflow & and chained 3=Totals not conditioned by overflow & not chained 4=Totals conditioned by overflow or chained 5=Headers and details conditionea by overflow or chained 6=Exception records
15	Number of Indicator sets
16	Output indicator set bits 0-1 indicator 1 0=not used 1=must be on 2=must be off 2-3 indicator 2, usage as in bits 0-1 4-5 indicator 3, usage as in bits 0-1
17	Indicator 1
18	Indicator 2
19	Indicator 3
20+	Same as 16-19 for each set of 'AND' indicators.

U - Output Field Descriptions

This is built by the Output Description Scan and used by the Output Generator.

0	U=Output Field Descriptions
- 1	bit 7 do not generate code for this record
2-3	record number
4-7	Field Conditioning Indicators - See Output Record Identification bytes 16-19
8-9	End position in record
10	Switches 0=1 *PLACE specified 1=1 Blank after 2=1 P-Packed format 3=1 G-Binary format 4=1 2 byte binary format 5=1 Card print option 6=1 *PRINT specified 7=1 4 byte binary format
11	Edit code from column 38 Output Description Specification bits 0-5 bits 2-7 of edit code bit 6 '$' bit 7 '*'
12	Descriptor 0=end of record 2=condensed field name description 4=index condensed field name description 6=literal index 8=literal 10=edit word
13	Length of following
14n	Condensed field description, literal, edit word, or binary index
n+1	Repeat 14n as needed

E - Entries and Extrns Record

Entries and Extrns records are written for all fields defined as external to the RPG or are RPG fields used by external routines. This applies to exits specified on file description specifications and EXITs, RLABLs, and ULABLs on Calculation Specifications.

Columns	Description
0	E - Entries and Extrns Record
1	Not Used
2-3	Record N.
~~14-15 or 19~~ 4-15 4-19	Condensed Field Description Entry for field

E - Entries and Extrns Record

Entries and Extrns records are written,for all fields defined as external to the RPG or are RPG fields used by external routines. This applies to exits specified on file description specifications and EXITs, RLABLs, and ULABLs on Calculation Specifications.

Columns	Description
0	E - Entries and Extrns Record
1	Not Used
2-3	Record
14-15 or 19	Condensed Field Description Entry for field

G - Generated Code Record

Generated code records are written by the generators and contain a preliminary version of the generated program. Control section addresses are not yet defined. These will be resolved in the code formatter phase. Note: error records are passed on as x records by the generator.

Columns	Field	
0	G - Generated Code Record	
1	Blank	
2-3	Record Number	
4	Group (CSECT)	
5	Length (Group Size - accumulated text length	
6-7	Address (Relative)	
8	Text Relocation Group	0 = end of record 1-253 = relocation group 254 = The following byte is not to be relocated and is to be propagated as indicated by the repeat count 255 = The following bytes are absolute
9	Repeat Count	either the number of following text words/bytes (words for 1-253, byte for 255 text relocation group) which share the relocation attribute specified by the TEXT relocation group, or the number of bytes to be propagated Special case - if repeat count = 255 then relocation group refers to an external ordinal and the implied length is two.
10-n	Text	
n+1 -	Same as 8-n	

REACTIVE COMPUTER SYSTEMS

DESIGN SPECIFICATION

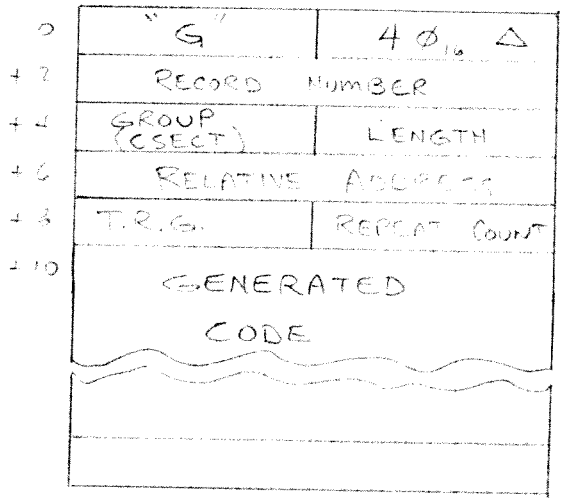
G$

Page _____

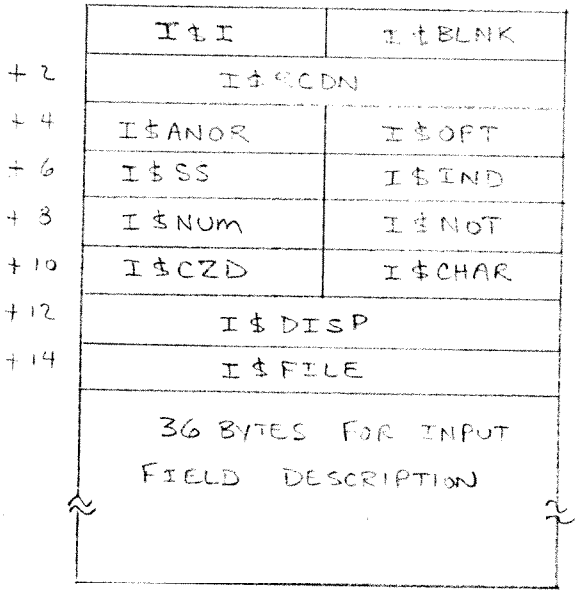
Date _____

Subject DISC RECORD FORMAT

GENERATED CODE REC



INPUT CONDENSED RECORD DESCRIPTION FORMAT



I$I = "I"

I$BLNK - BIT 7 ON = ERR

I$RCDN - RECORD NUMBER

I$ANOR - Ø = AND
2 = OR
4 = ALPHA
6 = END TABLE

I$OPT - Ø = BLANK
2 = ZERO

I$SS - STACKER SELECT

I$IND - INDICATOR ASSOCIATED WITH RECORD

I$NUM - NUMBER, Ø = NOT APPL
2 = ONE ONLY
4 = N

I$NOT - Ø = BLANK
2 = NOT CONDITION

I$CHAR - CHARACTER TO BE TESTED FOR (from I27, 34, 41)

I$DISP - CHARACTER DISPLACEMENT WITHIN RECORD

I$FILE - FILE NUMBER (RELATIVE TO BEGIN OF TABLE)

REACTIVE COMPUTER SYSTEMS

DESIGN SPECIFICATION

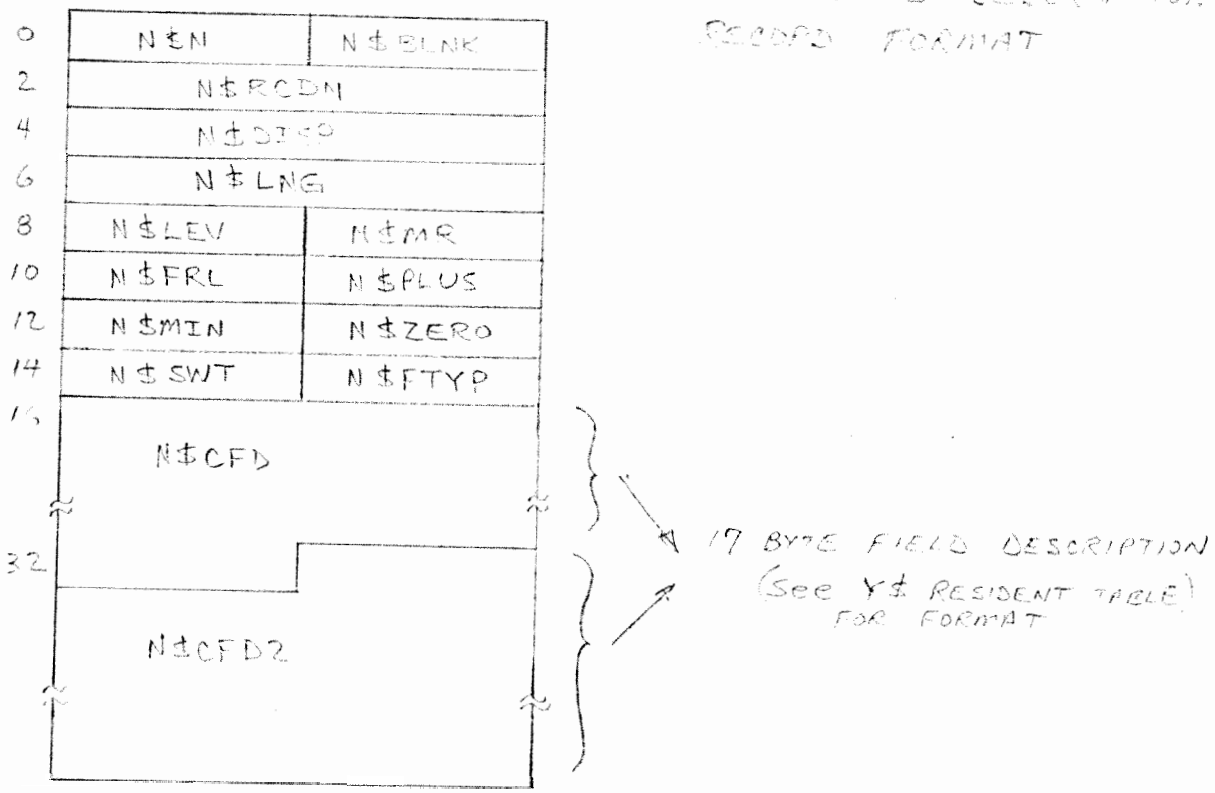
N$

Page

Date

Subject MODULE DISK [illegible]

INPUT FIELD DESCRIPTION RECORD FORMAT



N$N = "I"

N$BLNK - Bit 7 on = Error

N$RCDN - RECORD NUMBER

N$DISP - FIELD DISPLACEMENT IN RECORD

N$LNG - FIELD LENGTH

N$LEV - LEVEL FOR FIELD (L1-L9 OR ZERO)

N$MR - MATCHING REC LEV (M1-M9)

N$FRL - FIELD REC RELATION INDICATOR (01-99)

N$PLUS - PLUS INDICATOR

N$MIN - MINUS INDICATOR

N$ZRO - ZERO/BLANK INDICATOR

N$SWT - Bit 0 = PACKED
1 = BINARY

N$FTYP - 2 = REGULAR (12 BYTE ENTRY FOLLOWS)
4 = ARRAY (16 BYTE ENTRY FOLLOWS)

N$CFD - CONDENSED FIELD DESCRIPTION

VI.

VI. RPG GENERATED OBJECT PROGRAM

The RPG qenerated code consists of a main control routine, several optional subroutines, interpreted table and in line code.

A. Main Control Routine

The RPG built in loqic is in this routine, It determines when records are read, when to do detail and total calculations and output and calls various subroutines.

It operates chiefly on the File Control Table. From the information in this table it controls the RPG program's files.

There are pointers in the File Control Table to other qenerated tables (e.g. Record ID, Field moves etc). The control program passes the addresses of these tables to the appropriate subroutine.

B. Subroutines

1. **1/0** routines - Open, Close, Get, Put, Position
2. Table building - will read a table/array file and usinq the table description(s) in the File Control table will build the table.
3. Table Output - will write out an array/table accordinq to File Control table entries.
4. Record Identification - Processes the record identification table to identify input records-when a match is found it stores the record identification table address in the file control table for future reference.
5. Field moving (Input) - Processes the input field move table to move fields from input record to work areas.
   a. Test field record relation to see if field should be moved.

b. Pack a decimal field, move a packed decimal or alpha field, convert binary fields to packed decimal (for matching field moves-sign in forced to prositive) (for matching field alpha moves with alternate collating sequence-translate field after moving it).

c. Set field indicators as needed

6. Field moving (output) - Process the output field move table to create an output record.
   a. Test output field indicators.
   b. Perform move (edit).
   c. Do blank.

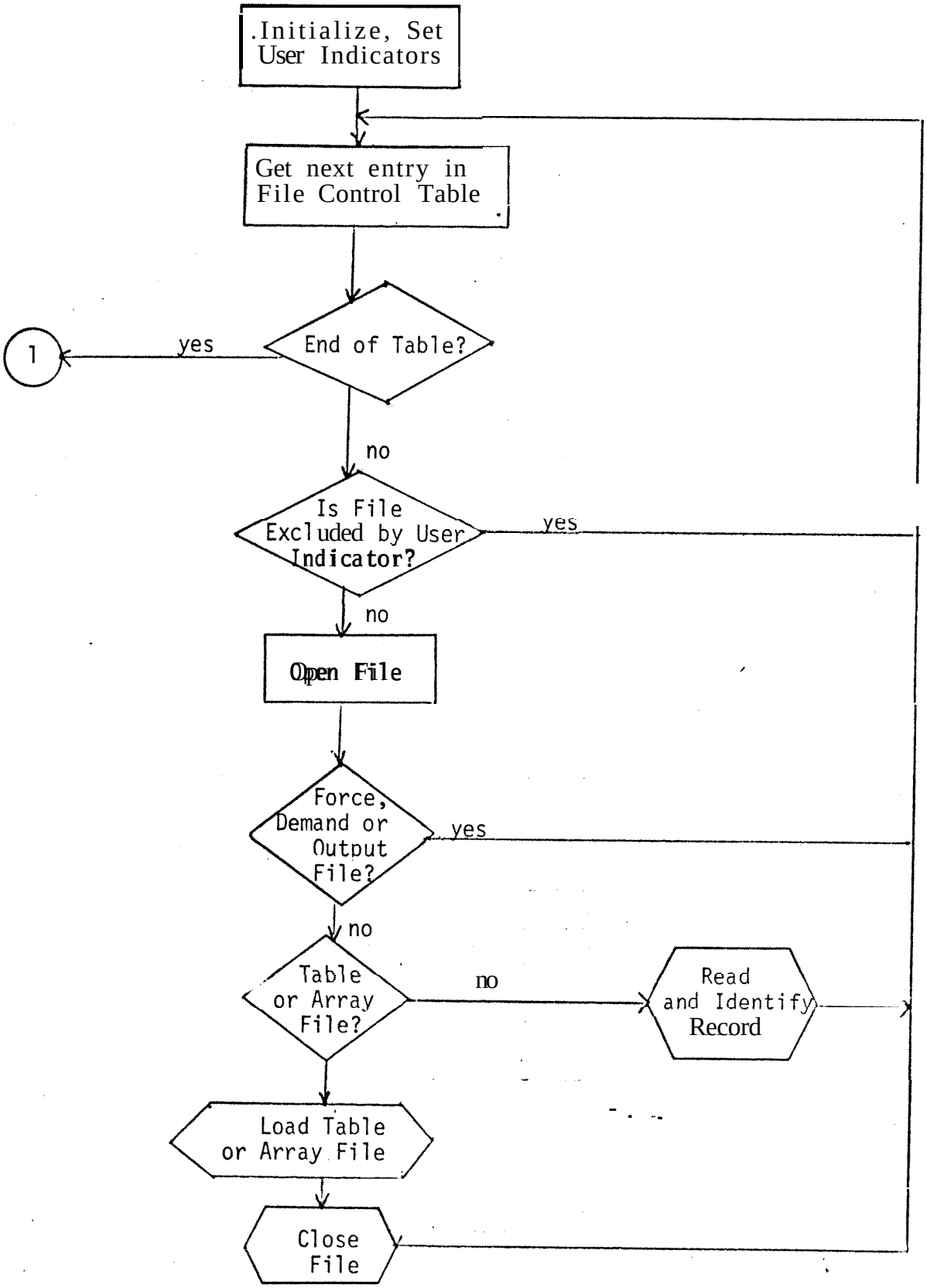
7. Line counter table processing.

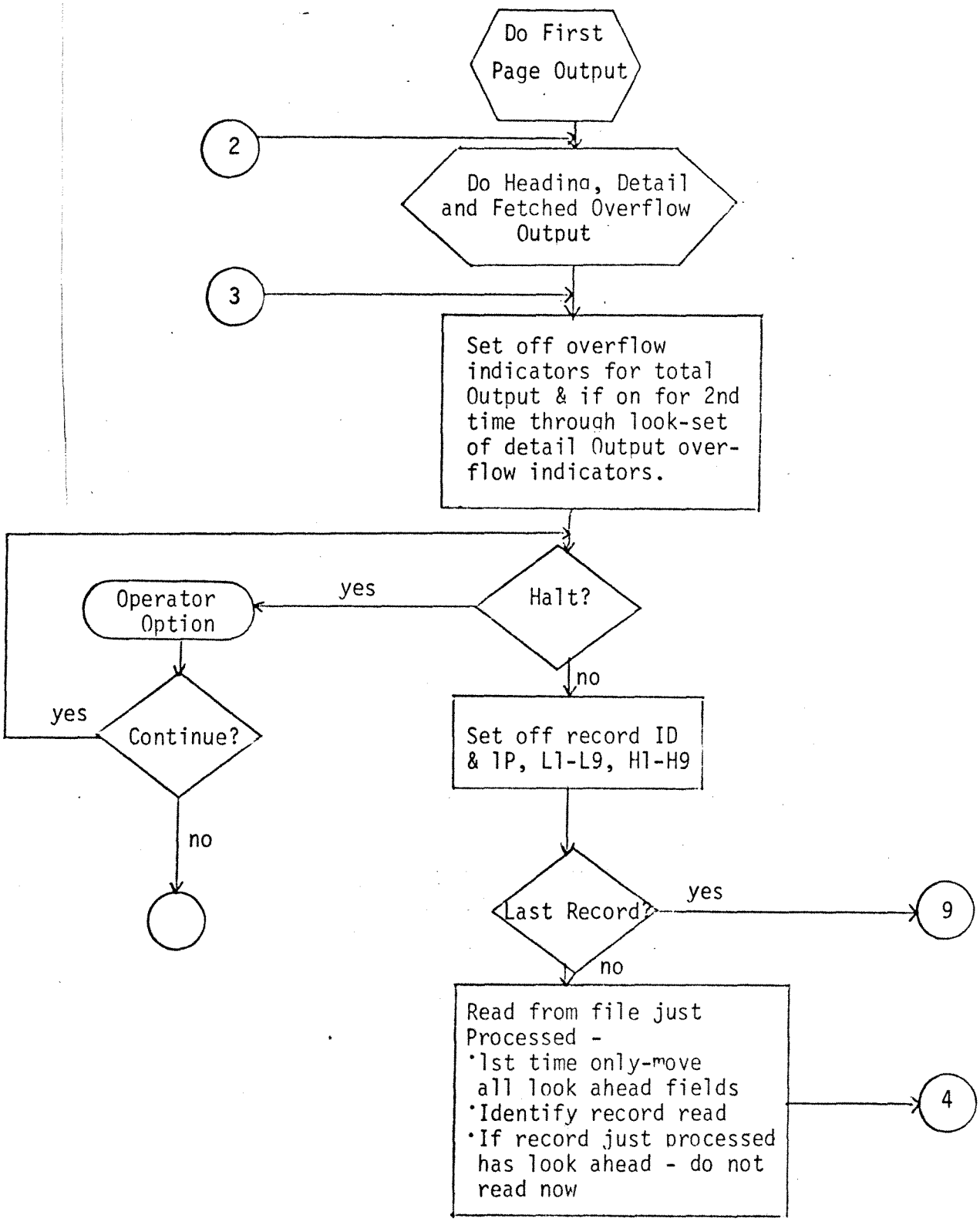
8. Level break test.

9. Matching Record test.

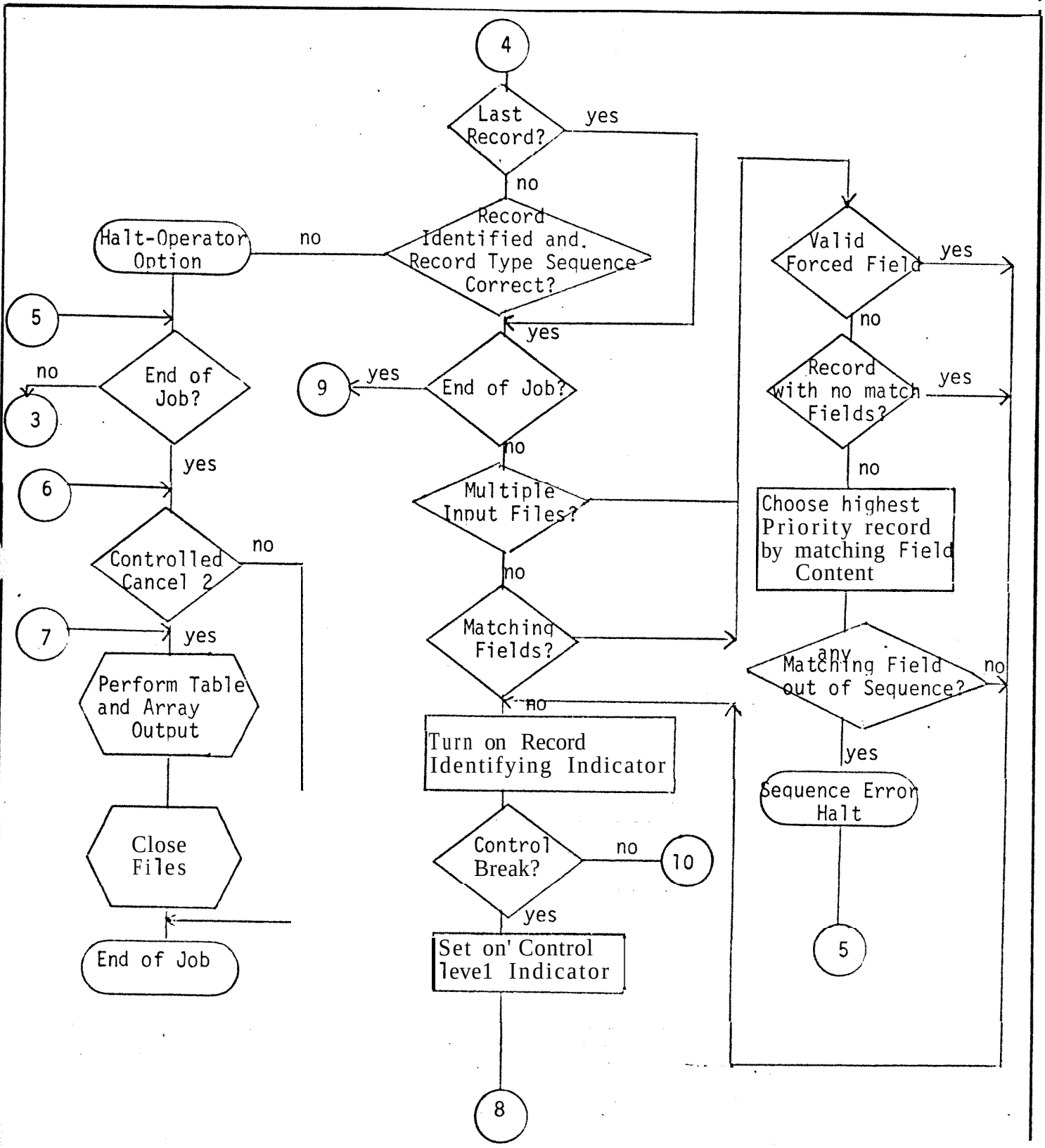
10. Calculation subroutines (divide, multiply, square root, etc. see Calc code section).

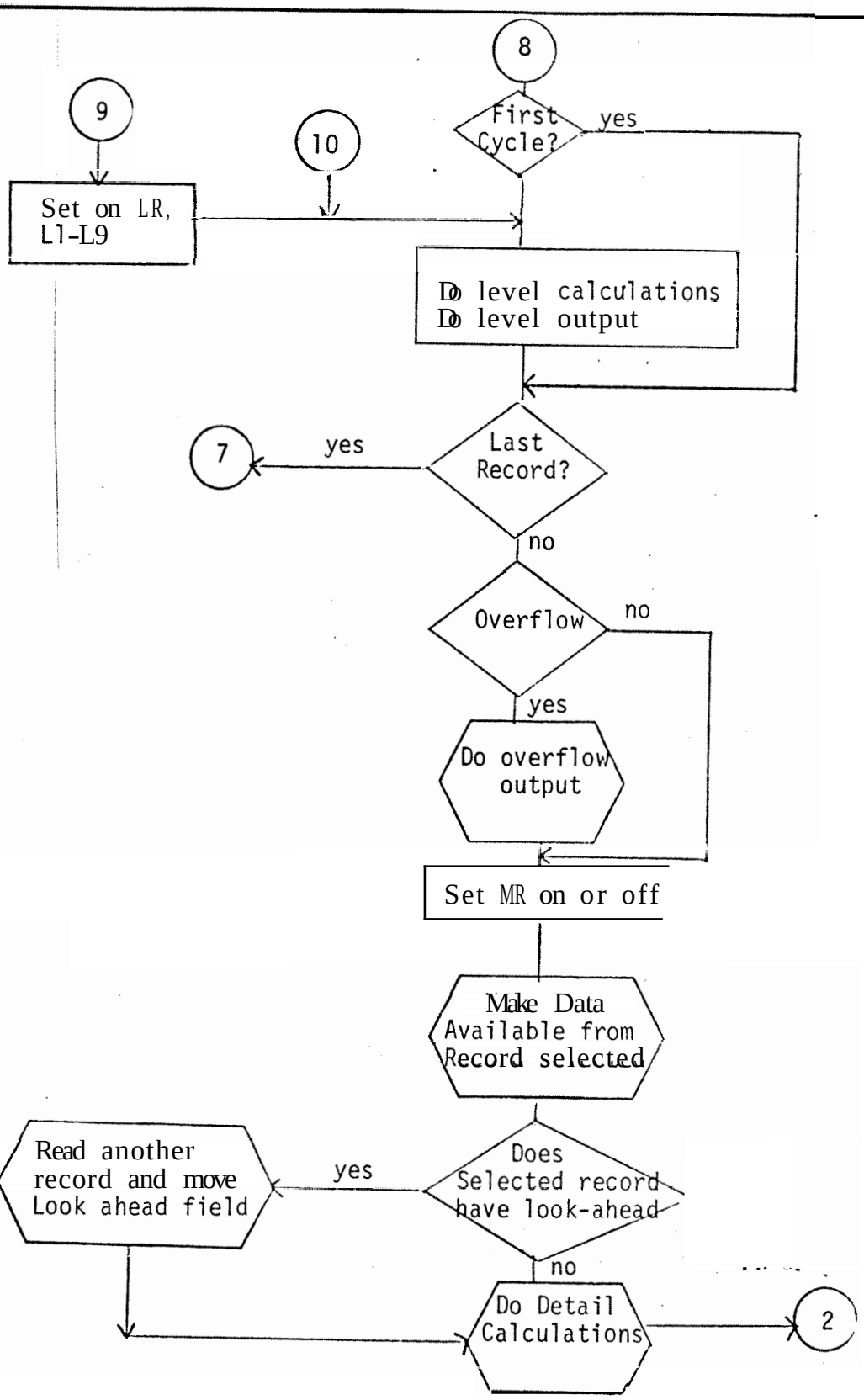
Subject OBJECT CONTROL PROGRAM




Do First Page Output
2
Do Heading, Detail and Fetched Overflow Output
3
Set off overflow indicators for total Output & if on for 2nd time through look-set of detail Output overflow indicators.
Halt?
yes
Operator Option
Continue?
yes
no
no
Set off record ID & 1P, L1-L9, H1-H9
Last Record?
yes
9
no
Read from file just Processed -
•1st time only-move all look ahead fields
•Identify record read
•If record just processed has look ahead - do not read now
4

Subject RPG GENERATED OBJECT PROGRAM




8
9
10
First Cycle?
yes
Set on LR, L1-L9
Do level calculations
Do level output
7
yes
Last Record?
no
Overflow
no
yes
Do overflow output
Set MR on or off
Make Data Available from Record selected
Read another record and move Look ahead field
yes
Does Selected record have look-ahead
no
Do Detail Calculations
2

Generated Code Groups

Each of the generation overlays generates code in one or more of the compile time Groups. The size of each Group is kent track of and the code formatter assigns the correct address to generated code b, resolving the startinq location of each Group.

Below is a list of each Group by the number that identifies it in the generated code. Included with each Group is a list of all the generators that produce code in that Group. The generators are identified as follows:

1. Control Program Generator
2. Input/Output Generator
3. File Extension/Line Counter Generator
4. Input Handling Generator
5. Calculations Generator
6. Output Handling Generator

The CSECTS are as follows:

No.	Description	Generated by					
		1	2	3	4	5	6
1	Object Time Communications Region	x	x	x	x	x	x
2	Not Used						
3	I/O/BDT		x				
4	File Control Table		x	x	x		x
5	Table Lookup/Line Counter Table			x			
6	Input Field Moves-Level/Matching Field				x		
7	Input Field Moves-Other				x		
8	Input Record ID Table				x		
9	Calculation					x	
10	Output Moves						x
11	Not Used						
12	Constants, Literals					x	x
13	Output Identification Table - Group 1						x
14	Output Identification Table - Group 2						x
15	Output Identification Table - Group 3						x
16	Output Identification Table - Group 4						x
17	Output Identification Table - Group 5						x
18	Output Identification Table - Group 6						x
19	Data Hold Area			x	x	x	
20	Forward References					x	
21	Indicators	x	x		x	x	x
22	Buffers and record area		x				

1. FILE CONTROL TABLE

One per file - fixed length 40 bytes each.

The File Control Table has a different format for Input, Output, Update, Record Address and Tables/Arrays.

Following is the portion of the table that applies to all file types.

Bytes	Description	Filled in TO v=variable BY phase number	TO (see pg. VI(7))	BY
F$TYPE 0	File Type Switches - More than one bit may be on			
	Bit 0=1 Input file		v	2
	1=1 Output file		v	2
	2=1 Record Address File (Limits)		v	2
	3=1 Record Address File (Record Numbers or keys)		v	2
	4=1 RAFed file - file RAF links to		v	2
	5=1 Chained File		v	2
	6=1 Combined/Update file		v	2
	7=1 MFCU File		v	2
F$TYP2 1	File Type Switches (continued)			
	Bit 0=1 Table/Array File		v	2
	1=1 Console File		v	2
	2=1 Demand File		v	2
	3=1 Random by keys		v	2
	4=1 Random by Record Number		v	2
	5=1 Unused		0	2
	6=1 E Specified (when all E's reach end of file-LR)		v	2
	7=1 End of file reached		0	2
F$TYP3 2	File Type Switches			
	Bit 0=1 Card File		v	2
	1=1 Print File		v	2
	2=1 Special File		v	2
	3=1 Matching Records Specified		v	2
	4=1 Matching Fld in current RCD		0	2
	5=1 Matching Records found		0	2
	6=1 Trailer in current RCD		0	2
	7=1 Lev 1 Flds in current RCD		0	2

Bytes	Description	Filled In	TO	BY
F$TYP4 3	Level 2-9 Present Switches Bit 0=1 Lev 2 FLDS in current RCD 1=1 Lev 3 FLDS in current RCD 2=1 Lev 4 FLDS in current RCD 3=1 Lev 5 FLDS in current RCD 4=1 Lev 6 FLDS in current RCD 5=1 Lev 7 FLDS in current RCD 6=1 Lev 8 FLDS in current RCD 7=1 Lev 9 FLDS in current RCD		0	2
F$BDT 4-5	Buffer Description Table (BDT) address or address of SPECIAL device support routine		v	2
F$LRAD 6-7	Logical Record Address		v	2
F$RLEN 8-9	Record Length (Maximum for Variable Length Records) At object time this is changed to actual record length		v	2
F$TRAN 10-11	File Translation Table Pointer		v	3
F$USER 12-13	User Indicator Address		v	2
F$LINK 14-15	Pointer to next input or prnt FCT		v	2
F$UN1 16-17	Unused		0	2
F$UN2 18	Unused		0	2
F$UN3 19	Unused		0	2

Below the portions referring to Input and Output files combine for update files.

Following is the portion of the table that pertains to Input files. (except RAF and Table/Array files.)

Bytes	Description	Filled in	To	By
F$PRI 20-21	File Priority (0=Primary, 1-19=Secondary)		v	2
F$RID 22-23	Record Identification Table Address		v	2
F$LOOK 24-25	Look ahead field Move Pointer		v	4
F$MACH 26-27	Matching Record Hold Area Pointer		v	4
F$LEV 23-29	Level Control Fields hold area Pointer		v	4

Byte	Description	Filled in	TO	BY
F$CRID	30-31	Current Record Identification Table Entry Pointer	0	4
F$RAFP	38-39	RAF File Pointer (from RAFed File)		

Following is the portion of the table that pertains to output files.

F$LINE	20-21	Line Counter Table Pointer	V	3
F$HAD	22-23	Output Record Identification Table chaining pointer - Header and Detail Records	V	6
F$T	24-25	Output Record Identification Table chaining pointer - Total Records	V	6
F$UP1	26-31	Left for possible update file		
F$MFCU	32-33	MFCU interpret area address	V	2
F$WORK	34-35	Work area - Last Line Number (Print files only) or relative key location in record (indexed files only)	0	6
F$SKIP	36	Switches Bit 0=1 Last Line had a skip after or 36-37 Pointer to Low Trailer Displacement	0	6
F$STAT	37	Overflow Status 0=Overflow work not yet done 1=Fetched overflow done Output will change this to 2 instead of doing normal overflow 2=All overflow work done - exec should turn off overflow indicator and this switch set by output routines after overflow processing.	0	6
F$OVER	38-39	Pointer to Overflow Indicator or Trailer Displacement	V	2

Following is the portion of the table that pertains to Record Address Files (RAF)

F$ELEN	20-21	RAF Element Length	V	2
F$CLOC	22-23	RAF current location in record	0	2
F$FPTR	24-40	RAF File FCT Pointer	V	2

Following is the portion of the table that pertains to Table/Array files.

Byte	Description	Filled in	TO	BY
26-27	Pointer to Table Control (alternate table pointed to by TACT chain)		v	3
FSTCON 26-27	~~Pointer to TACT (table/array control table)~~		~~v~~	~~3~~

REACTIVE COMPUTER SYSTEMS

DESIGN SPECIFICATION

Page
Date
RLB

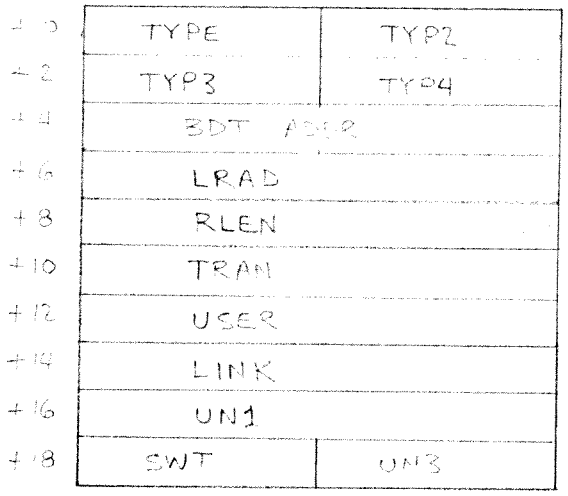
Subject RMS GENERATED OBJECT CODE

F$

FILE CONTROL TABLE (FCT)

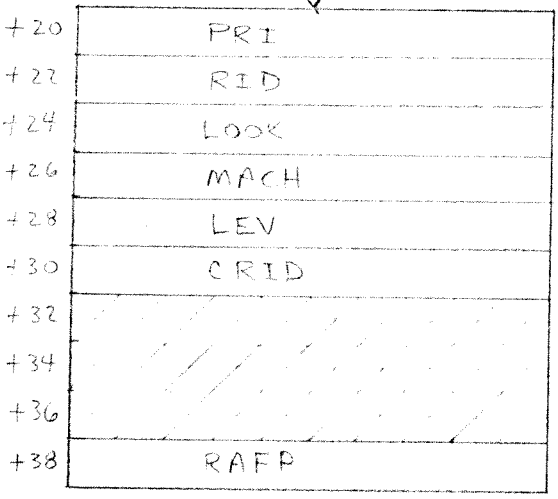
see PAGE VI (8) in D.S.

FCT - fixed length, 40 bytes

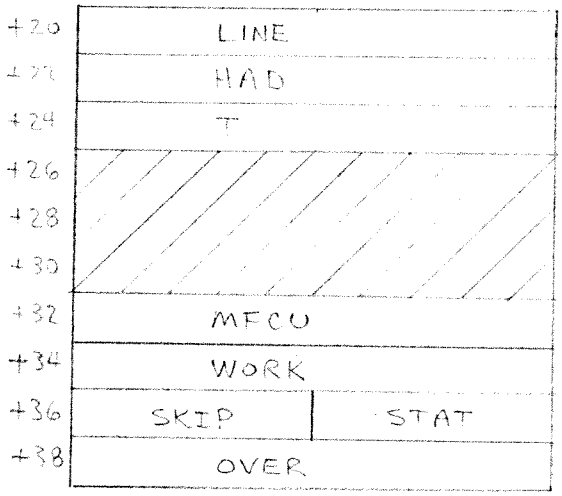


BASE FORMAT, FOLLOWED BY 1 OF 4 LAYOUTS SHOWN BELOW

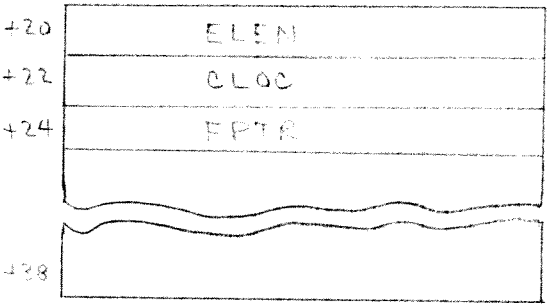
* INPUT



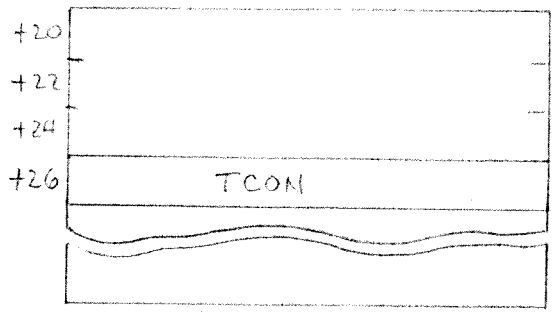
* OUTPUT



* RECORD ADDRESS



* TABLE/ARRAY



2. Input Record Identification

The input record identification routine must:
A, Identify Record Type
B. Check for proper sequence of input records

To accomplish this the Input Record Identification routine interpretively processes the Input Record Identification Table.

The beginning of the table is pointed to by a field in the file control table. Also there is a work area in the file control table for saving the location of the last identified record identification entry, and a switch to indicate whether alpha or numeric sequence is being processed.

Processing of the table always resumes where it had last left off. If the alpha sequences are being processed, all of them must be checked in a circular chain before going to the numeric sequences. The numeric sequences directly follow the alphas - when numeric sequences are reached, the position is marked and the routine returns to the beginning of the table, when all alphas are processed, the routine tnen goes to the numeric sequences. If a non-optional numeric sequence is passed in the table an error switch is turned on but the search continues till all table entries are checked.

INPUT RECORD IDENTIFICATION TABLE

Bytes Description

R$ TYPE 0 Type of record
0 = AND
2 = OR
4 = First Numeric Entry (an assumed OR)
6 = End of Table
8 = Trailer Description

R$ SWT 1-2 Switches
Bit 0=1 Option = 0
1=1 Numeric Entry
2=1 Only One Record Permissable
3=1 Not = N
4=1 Beginning of record id

R$ SS 3 Stacker Select

R$ IND 4-5 Pointer to Indicator associated with record

R$ CNZ 6 Portion of Character to test
0 = Character
2 = Numeric
4 = Zone
6 = No Character to test - Record Identified

R$

R$CHAR	7	Character or Portion to be tested.
R$DISP	8-9	Displacement within record of character to be tested
R$MOVE	10-11	Pointer to field moves table
R$LMV	12-13	Pointer to level/MR moves table

R$

INPUT RECORD IDENTIFICATION TABLE

See PAGE VI(9) in D.S.

+0	TYPE	SWT
+2	(SWT+1)	SS
+4	IND	
+6	CNT	CHAR.
+8	DISP	
+10	MOVE	
+12	LMV	

3. Output Record Identification Table

	1	And/Or relationship with previous table entry Bit 0=1, And, Bit 1=1, OR, Bit 2=1 ADD, Bit 3=1 End of Table
	1	Stacked Select/Fetched overflow, 15=Fetched overflow
	1	Space before
	1	Space after
	1	Skip before
	1	Skip after
	2	Field description pointer for this record type
	2	File control table pointer
3*	1	1=presence is required, 2=must be off (NOT)
3*	1	Indicator to be tested

4, Input Field Move

This table is in three sections:
a. Level/Matching fields move (10 bytes entries)
b. Regular field moves (10-13 byte entries
c. Look Ahead field moves (10-13 byte entries)

	1	Switches Bit 0 = 1 Level Move 1 = 1 Matching Field Move 2 = 1 Array Move - constant index 3 = 1 Array Move - variable Index 4 = 1 Packed Decimal 5 = 1 Bytes 2-3 of this entry point to TR (trailer) moves? 6 = 1 Field indicator trailers 7 = 1 End of table (no entries follow)
Instruction image for all but binary moves	1	Fie1d Record relation indicator (or zero)
	1	Move operation code (PACK, MOVX, binary)
	1	50_{16} (M=5, R=0) Index 5 will contain the address of the input record
	2	From displacement in record
	2	To address
	1	From length
	1	To length
	2	Array, index (present only for array move)
group of 3 is optional	1	Plus field indicator or zero
	1	Minus field indicator or zero
	1	Zero/blank field indicator orzero

5. Output Field Moves

1 Type of data in table entry

value = 0 Field conditioning indicators
2 Edit (edit word must be moved to OP then edit operation)
4 Binary OP field
6 Regular move
8 Unpack
10 Blank after
12 Array element
14 Full array loop control
16 * PLACE
18 End of moves

The above precedes each of the following formats

---- Field conditioning indicators

3 Bit 0 = 1 Not specified (N)
1-7 = Indicator
The above is repeated 3 times zero means no indicator to test.

---- Edit

1 Switches
Bit 0 = 1 floating dollar sign

2 From location
2 to location
2 Edit word pointer
1 From length
1 Edit word length

---- Binary field

1 From length
2 From location
2 To location

---- Regular move or Unpack

1	Not used
1	Operation Code
1	06 16 - Register 6 is record base
2	From location
2	To displacement in record
1	From length
1	To length

Blank after

1	Blank after indicator

---- Array Element

1	Switches Bit 7=1 Immediate Binary value is index Bit 6=1 Pointer to binary value is index
2	Address from - (Address of array table)
2	Indes (Immediate value or pointer to binary or decimal index) (Note - result of this operation is the From Address for the next move entry in table - the result of this array operation will be put in a work area - which will be pointed to by the next operation).

---- Full Array Loop Control

1	Not Used
2	Array index increment in binary
2	Pointer to array index (in previous table entry)
2	Maximum index value
2	Transfer address

---- * PLACE

1	Switch Bit 6=1 * PLACE currently in process (set at execution time - if off then register 6 must be incrernented, if on register 6 must be decremented).
2	Increment/Decrement to register 6 in binary
2	Transfer address

Table and Array Processing

Tables and arrays require a run time control table in Code Group 5. The location in Group 5 of the control table for a table/array can be found by taking the table file entry number (from the field description table) minus one times 24. The result is the relative location in Code Group 5.

The format of the table/array control table is:

	Bytes	Description
T$TYP	0	Bit 0=0 Table Name 0=1 Array Name 1=1 Ascending 2-1 Descending 3-1 Input Data - packed decimal 4-1 Input Data - binary 5-1 Numeric 6-1 Alpha 7-1 Alternating table
T$ILEN	1	~~Not used~~ Pointer , input length (A)
T$STRT	2-3	Pointer to beginning of table/array
T$NEXT	4-5	Pointer to byte following table/array
T$LEN T$LEN1	6-7	Entry length
T$ENT T$ENT1	8-9	Number of entries
T$LELM	10-11	Pointer to last element found area
T$NUM	12-13	Number of entries per record
T$IN	14-15	Chaining input TACT address
T$OUT	16-17	Chaining output TACT address
T$LAST	18-19	Last entry address of upper search table
T$UPER	20	Power 2 of upper table length +1
T$LOWR	21	Power 2 of lower table length +1
T$FRST	22-23	First entry address of lower search table

REACTIVE COMPUTER SYSTEMS

DESIGN SPECIFICATION

Page _____
Date _____

Subject _____

See page VI (13) in D.S.

T$

	TYP	ILEN
+2	STRT	
+4	NEXT	
+6	LEN	LEN1
+8	ENT	ENT1
+10	LELM	
+12	NUM	
+14	IN	
+16	OUT	
+18	LAST	
+20	UPER	LOWR
+22	FRST	

Object time line counter table.

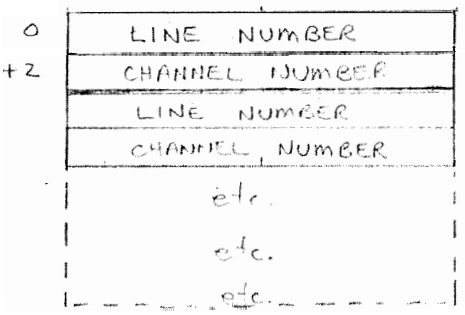
The Object time Line Counter Table is pointed to by F$LINE in the File Control Table. The Line Counter table has the following format.

Byte Description

0-1 Line Number

2-3 Channel number (1 = top of page, 12 = overflow line, 14 = line number contains lines per page.)

0-3 are repeated as often as necessary (a maximum of 11 times) the terminating entry is a word of zeros.



SPECIAL FILE LINKAGE

Special Device types are read in a program external to the RPG. This external program must provide all the interfaces with data management and the record buffer(s) and GET logical record area.

Linkage to the external routine from the RPG uses the standard linkage.

- the parameter list address will be in register six (6)
- the save area address will be set in register seven (7)
- the return address will be set in the first word of the save area.

The format of the parameter list is:

Word	Contents
1	Length of List (always 11)
2	Bits 0 - 7 - function code = 0 GET 4 = OPEN = 1 PUT 5 = CLOSE Bit 12 = 1 variable length records are specified, and word 7 must point to the record size. Bit 13 = 1 end of file return address specified in word 9.
3	Error return code - set by the external routine (if non-zero the job will be stopped and the error return code printed out) ,
4-6	Not used.
7	Record size address - Points to a one word (2 byte) location which will contain the record size. (Bit 12 of word 2 must be = 1) For GET, the record size address and record size are set by the external program. For PUT they are set by the RPG.
8	not used
9	End of file return address (always specified - bit 13 of word 2 will always be = 1)
10-11	not used.

Word	Contents
12	Address of record • For Get this address is supplied by the external routine. • For Put this addres's is supplied by the RPG compiler.

The Save Area (pointed to by register seven) format is:

Word	Contents
1	Return address
2-9	Used by called program to save registers.

The Linkage to the run time error routine is:

```
LODD    error number          R6
LOD     variable field (or ptr) R1
JSR     W$ERR                 R7
```

If the operator specifies the continue option, control will be returned to the instruction following the JSR.

If there is no optional field, R1 need not be initialized.

A tentative list of run time error messages follows. Please supply me with any changes or additions.

Err No.	Message	Variable Field	Continue	By Pass	Cntld Cancel	Immed. Cancel
1	Indicator____ is on	H0-H9	X		X	X
2	Negative Square Root at line number ____	line number (in binary)	X		X	X
3	Arithmetic Overflow at line number ____	line number (in binary)	X		X	X
4	Divide by zero at line number____	line number (in binary)	X		X	X
5	Variable index is not within array bounds (zero,minus or too large) at line number__	line number (in binary)	X		X	X
6	Table/Array out of sequence, from file _____, T/A number within file_____, record number within T/A _____	ptr to 6 byte field with file ID ptr and T/A number in files & record #	X		X	X

Err No.	Message	Variable Field	Continue	By Pass	Cutld Cancel	Immed. Cancel
7	Table/Array not found from file _____, T/A number within file _____	ptr to 4 byte field with file ID ptr & TA number in file.	X		X	X
8	Too many entries for a table/array	(same as for 7)	X		X	X
9	Available partition size exceeded	none				X
10	IP Forms Alignment	none	X	X		
11	Record out of Sequence for file _____	ptr to file name		X	X	X
12	File Out of Matching Record sequence for file _____	ptr to file name		X	X	X
13	Unidentified Record from file _____	ptr to file name		X	X	X
14	Invalid Numerical Data at line _____	line number (in binary)	X		X	X
15	Channel not defined on Line Counter for file_____	ptr to filename	X			X
16	Binary conversion overflow at line number_____	line number (in binary)	X			X
17-n	10 Errors (not defined)	ptr to filename				

```
                +*
                +*                  OBJECT TIME COMMUNICATIONS AREA
                +*
                +*
                +*                  FOLLOWING ARE FILLED IN BY GENERATORS OR FORMATTER
                +*
0000 +W$FA      @EQU  0             JUMP $RGEXEC - 1ST EXECUTABLE RPG INS   0000006
0004 +W$SWT     @EQU  4             SWITCHES                                0000006
                +*                   BIT 0=1 MATCHING FIELDS IN CUR KCD
                +*                       1=1 SIGN FORCING ON INPUT TBLS
                +*                       2=1 HALT ON OVERFLOW
                +*                       3=1 FIRST CYCLE
                +*                       4=1 LOW SEQ MTCH FLDS PRI FILE
                +*                       5=1 ONE OF MTCH PLUS SEC FILE
                +*                       6=1 MTCH FILES IN DESC SEQ
                +*                       7=1 NOT DOMESTIC FORMAT
0005 +W$SWT2    @EQU  5             SWITCHES                                0000006
0006 +W$FCT     @EQU  6             BEGINNING FCT POINTER                   0000006
0008 +W$FCTE    @EQU  8             NUMBER OF FCT ENTRIES                   0000006
000A +W$HALT    @EQU  10            POINTER TO H0-H9,L0-L9,LR,MR,OA-OV      0000006
000C +W$OG1     @EQU  12            POINTER TO FIRST PAGE GROUP             0000006
000E +W$OG2     @EQU  14            POINTER TO HDRS AND DETAIL              0000006
0010 +W$OG3     @EQU  16            POINTER TO TOTALS                       0000006
0012 +W$OG4     @EQU  18            POINTER TO TOTALS - OVERFLOW            0000006
0014 +W$OG5     @EQU  20            POINTER TO HDRS & DETAIL - OVERFLOW     0000006
0016 +W$OG6     @EQU  22            POINTER TO EXCEPTION RECORDS            0000006
0018 +W$LEVH    @EQU  24            POINTER TO MASTER LEVEL HOLD AREA       0000006
001A +W$MTCL    @EQU  26            LENGTH OF MASTER MATCHING FIELD AREA    0000006
001C +W$MTCH    @EQU  28            POINTER TO MASTER MTCH FIELD AREA       0000006
001E +W$IND     @EQU  30            POINTER TO BEGINNING OF INDICATORS      0000006
0020 +W$DATE    @EQU  32            PTR TO DATE - ODDSOMMSOYYSOODMMYYS      0000006
0022 +W$TACT    @EQU  34            POINTER TO BEGINNING OF TACT TABLE      0000006
0024 +W$CALC    @EQU  36            POINTER TO DETAIL CALC ROUTINE          0000006
0026 +W$L0      @EQU  38            POINTER TO LEVEL 0 CALCULATIONS         0000006
0028 +W$L1      @EQU  40            POINTER TO LEVEL 1 CALCULATIONS         0000006
002A +W$L2      @EQU  42            POINTER TO LEVEL 2 CALCULATIONS         0000006
002C +W$L3      @EQU  44            POINTER TO LEVEL 3 CALCULATIONS         0000006
002E +W$L4      @EQU  46            POINTER TO LEVEL 4 CALCULATIONS         0000006
0030 +W$L5      @EQU  48            POINTER TO LEVEL 5 CALCULATIONS         0000006
0032 +W$L6      @EQU  50            POINTER TO LEVEL 6 CALCULATIONS         0000006
0034 +W$L7      @EQU  52            POINTER TO LEVEL 7 CALCULATIONS         0000006
0036 +W$L8      @EQU  54            POINTER TO LEVEL 8 CALCULATIONS         0000006
0038 +W$L9      @EQU  56            POINTER TO LEVEL 9 CALCULATIONS         0000006
003A +W$LR      @EQU  58            POINTER TO LAST RECORD CALCULATIONS     0000006
003C +W$NPUT    @EQU  60            POINTER TO FIRST INPUT FILE             0000006
003E +W$PRNT    @EQU  62            POINTER TO FIRST PRINT FILE FCT         0000006
0040 +W$FORC    @EQU  64            POINTER TO FORCE FCT ENTRY              0000006
0042 +W$ALTS    @EQU  66            POINTER TO ALT COLLATING SEQ TABLE      0000006
```

```
      +*
      +*                    FOLLOWING ADDRESSES ARE GENERATED THROUGH EXTRNS
      +*
0044  +W$MRL     @EQU  68    POINTER TO $RGMRL OR ZERO                0000006
0046  +W$LEV     @EQU  70    POINTER TO $RGLEV OR ZERO                0000006
0048  +W$BINP    @EQU  72    POINTER TO BINARY TO PACKED CONVERSN     0000006
004A  +W$BINA    @EQU  74    POINTER TO INDEXED BIN TO PACKED CNV     0000006
004A  +W$BINM    @EQU  74    POINTER TO MTCH FLD BIN TO PCKD CNV      0000006
004C  +W$BINL    @EQU  76    POINTER TO LEVEL FLD BIN TO PCKD CNV     0000006
004E  +W$FIP     @EQU  78    POINTER TO NUMERIC FLD TESTING RTN       0000006
0050  +W$FIPA    @EQU  80    POINTER TO INDEXED NUM FLD TEST RTN      0000006
0052  +W$FIA     @EQU  82    POINTER TO ALPHA FLD TESTING RTN         0000006
0054  +W$FIAA    @EQU  84    POINTER TO INDEXED ALPHA FLD TST RTN     0000006
0056  +W$CVI     @EQU  86    PTR TO FIND ARRAY ELEMENT OFFSET         0000006
0058  +W$FINU    @EQU  88    POINTER TO FINISH PACK ROUTINE           0000006
0058  +W$CAN     @EQU  88    POINTER TO FINISH PACK ROUTINE           0000006
005A  +W$OFP     @EQU  90    POINTER TO FIELD INDICATOR PROC RTN      0000006
005C  +W$OFB     @EQU  92    POINTER TO CONVERT TO BINARY ROUTINE     0000006
005E  +W$OAC     @EQU  94    POINTER TO ARRAY OUTPUT CONTROL RTN      0000006
0060  +W$OPL     @EQU  96    POINTER TO *PLACE PROCESSING ROUTINE     0000006
0062  +W$AR1     @EQU  98    PTR TO ADD/SUB WITH HALF ADJUST          0000006
0064  +W$AR2     @EQU  100   PTR TO Z-ADD/Z-SUB WITH HALF-ADJUST      0000006
0066  +W$CSO     @EQU  102   PTR TO SIGN FORCING ON UNPACKED DEC      0000006
0068  +W$HA      @EQU  104   PTR TO RUNTIME HALF-ADJUST ROUTINE       0000006
006A  +W$TSB     @EQU  106   PTR TO RUNTIME TEST BIT ROUTINE          0000006
006C  +W$TSZ     @EQU  108   PTR TO RUNTIME TEST ZONE ROUTINE         0000006
006E  +W$CHN     @EQU  110   PTR TO RUN TIME CHAIN ROUTINE            0000006
0070  +W$DBG     @EQU  112   PTR TO RUN TIME DEBUG ROUTIN             0000006
0072  +W$DIV     @EQU  114   PTR TO RUN TIME DIVIDE ROUTINE           0000006
0074  +W$MUL     @EQU  116   PTR TO RUNTIME MULTIPLY ROUTINE          0000006
0076  +W$DSP     @EQU  118   PTR TO RUN TIME DISPLAY ROUTINE          0000006
0078  +W$REA     @EQU  120   PTR TO RUNTIME READ ROUTINE              0000006
007A  +W$LPB     @EQU  122   PTR TO RUN TIME BINARY LOOKUP RTN        0000006
007C  +W$LPS     @EQU  124   PTR TO RUNTIME SEQUENTIAL LOOKUP         0000006
007E  +W$SQR     @EQU  126   PTR TO RUNTIME SQUARE ROOT ROUTINE       0000006
```

```
007E +W$SQR     @EQU  126                PTR TO RUNTIME SQUARE ROOT ROUTINE   0000006
       +*
       +*                  FOLLOWING ADDRESS 2 SUPPLIED BY RUN TIME EXEC
       +*
0080 +W$ERR     @EQU  128                JUN $RGERR                           0000006
0084 +W$GET     @EQU  132                POINTER TO $RGGET                    0000006
0086 +W$PUT     @EQU  134                POINTER TO $RGPUT                    0000006
0088 +W$ORS     @EQU  136                PTR TO RUNTIME EXCPT ROUTINE         0000006
008A +W$SGNC    @EQU  138                POINTER TO SIGN FORCING ROUTINE      0000006
008A +W$CSI     @EQU  138                PTR TO SIGN FORCING ON PACKED DEC    0000006
008C +W$SGNA    @EQU  140                POINTER TO INDEXED SIGN FORCING RTN  0000006
008E +W$ZERL    @EQU  142                POINTER TO LEVEL SIGN ELIMINATION    0000006
0090 +W$ZERM    @EQU  144                POINTER TO MATCH SIGN ELIMINATION    0000006
0092 +W$OSP     @EQU  146                POINTER TO FORCE SIGN PKD FLD RTN    0000006
0094 +W$OSU     @EQU  148                POINTER TO FORCE SIGN UNPKD FLD RTN  0000006
0096 +W$CIN     @EQU  150                PTR TO TEST FOR CONDITIONING INDS    0000006
0098 +W$RIN     @EQU  152                PTR TO SET ON/OFF RESULT INDICATORS  0000006
009A +W$OVF     @EQU  154                PTR TO RUNTIME OVERFLOW PROCESS      0000006
009C +W$OFM     @EQU  156                END OF OUTPUT MOVES RETURN POINT     0000006
009E +W$XL      @EQU  158                LEVEL CALC RETRUN ADDRESS            0000006
00A0 +W$XD      @EQU  160                DETAIL CALC RETURN ADDRESS           0000006
00A2 +W$LMRM    @EQU  162                LEVEL/MR MOVES RETRUN ADDRESS        0000006
00A4 +W$FLDM    @EQU  164                REGULAR FIELD MOVES RETRUN ADDRESS   0000006
       +*
       +*                  CONSTANTS AND WORK AREAS
       +*
00A6 +W$MSK1    @EQU  166                X'0FFF' MASK                         0000006
00A8 +W$MSK2    @EQU  168                X'FF0F' MASK                         0000006
00AA +W$LEVL    @EQU  170                LENGTH OF EACH LEVEL FIELD (0-9)     0000006
00B4 +W$LEVT    @EQU  180                TOTAL LEVEL LENGTH                   0000006
00B6 +W$SAV     @EQU  182                REGISTER SAVE AREA POINTER(16BYTES)  0000006
00C6 +W$WA      @EQU  198                PTR TO ARITH WORK AREA (16 BYTES)    0000006
```

VII.

VII. G. Calculation Specification Scan

Internal tables:

Parameter Definition Table (PDT)

Number of entries = 35
Entry size = 12 bytes

Byte	Bit	Description
0-4		Command name
5		Index into SRBT (0 = no special processing)
6		Bit map. Meaning of each bit: bit=0, field must be blank; bit=1, field does not have to be blank.
	0	conditioning ind.
	1	field length
	2	decimal position
	3	half-adjust
	4	result indicator
	5	factor 1
	6	factor 2
	7	result field
7		Bit map. Meaning of each bit: bit=0, field may be blank; bit=1, field must be present, .
	0-7	For bit assignment see byte 6.
8		Indicators for factor 1. Meaning of each bit: bit=0, type not allowed; bit=1, type allowed.
	0	literal
	1	field or element
	2	entire table or array
	3	special name
	4	TAG or subroutine name
	5	file name
	6	alphameric
	7	numeric
9		Indicators for factor 2. For specifics, see byte 8.

10		Indicators for result field. For specifics, see byte 8.
11		Miscellaneous indicators
	0=1	control level must be SR.
	1=1	only AN/OR is acceptable

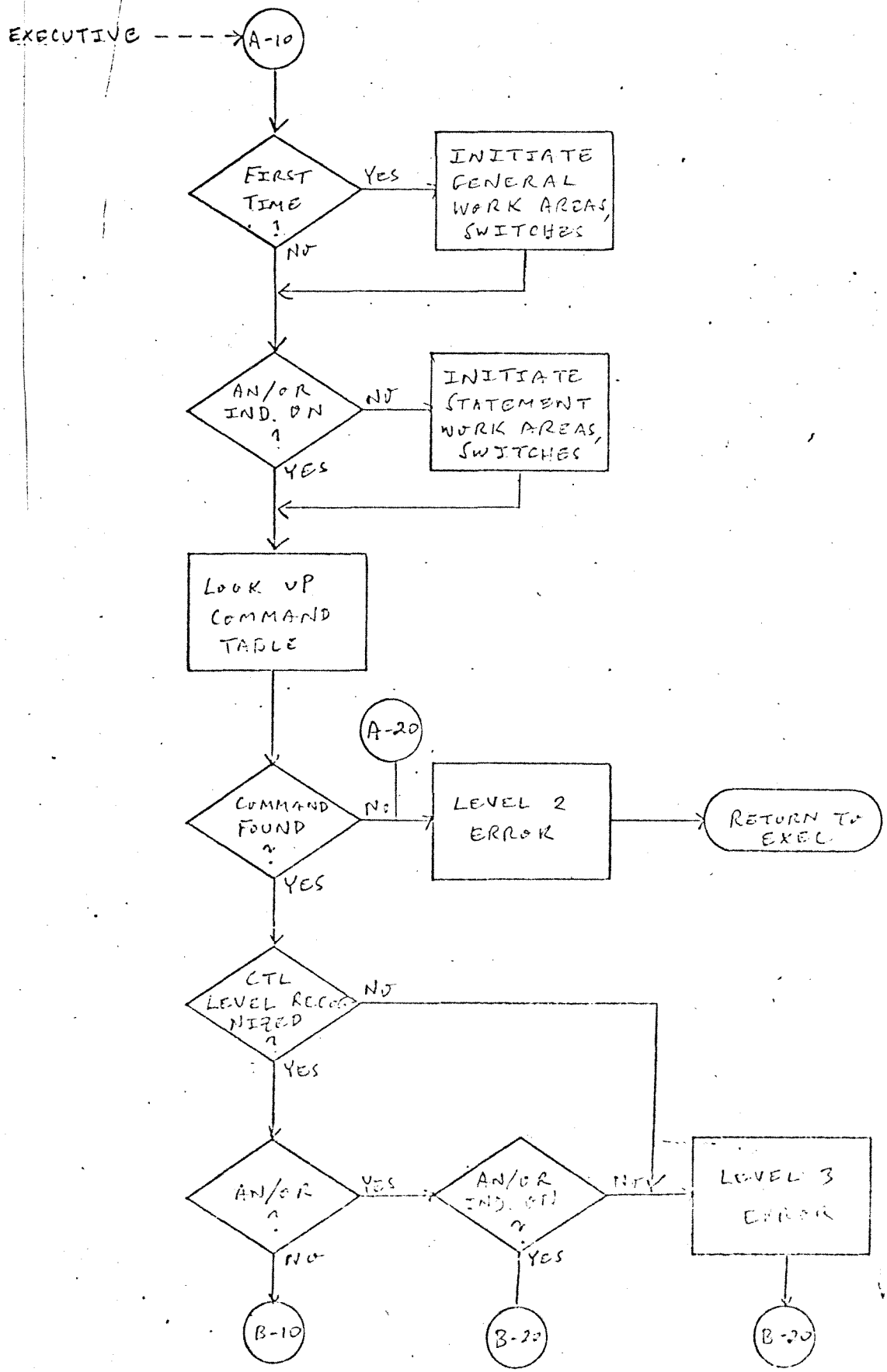
Special Requirement Branch Table (SRBT)

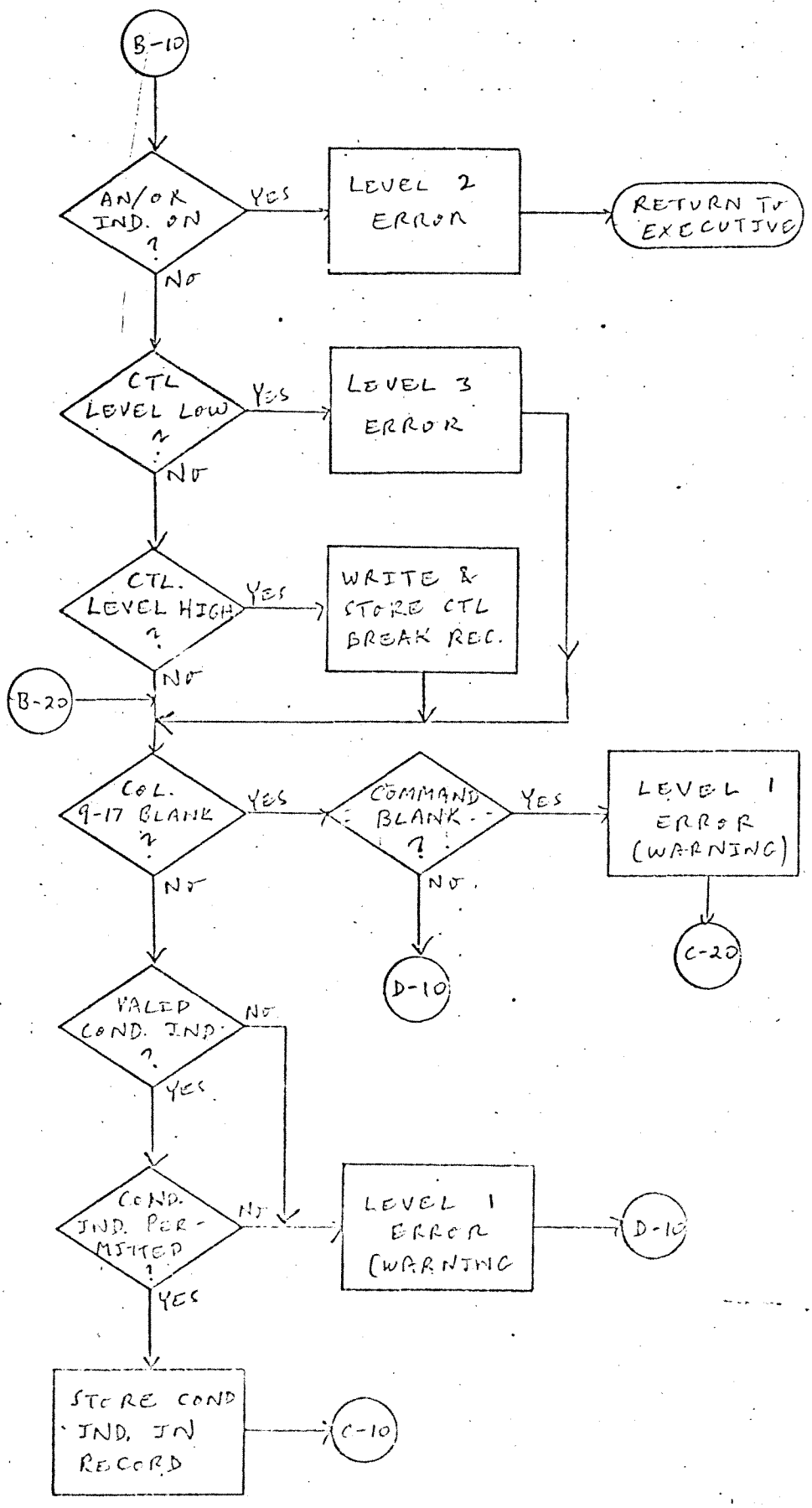
Number of entries: 15
Entry size: 2 bytes

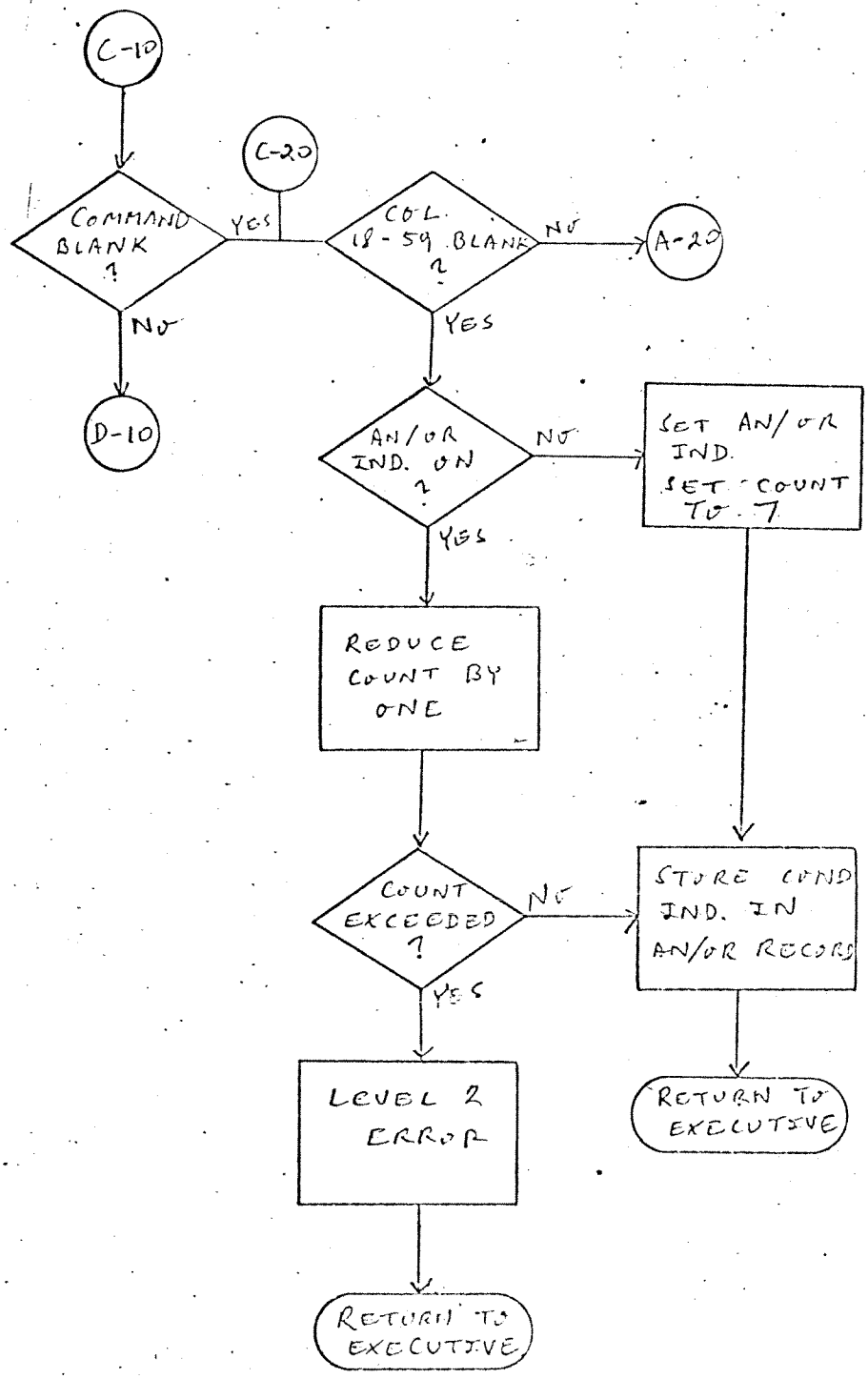
Branch addresses to routines that perform specific command-related processing. For details, see flow chart pp. 10-16.

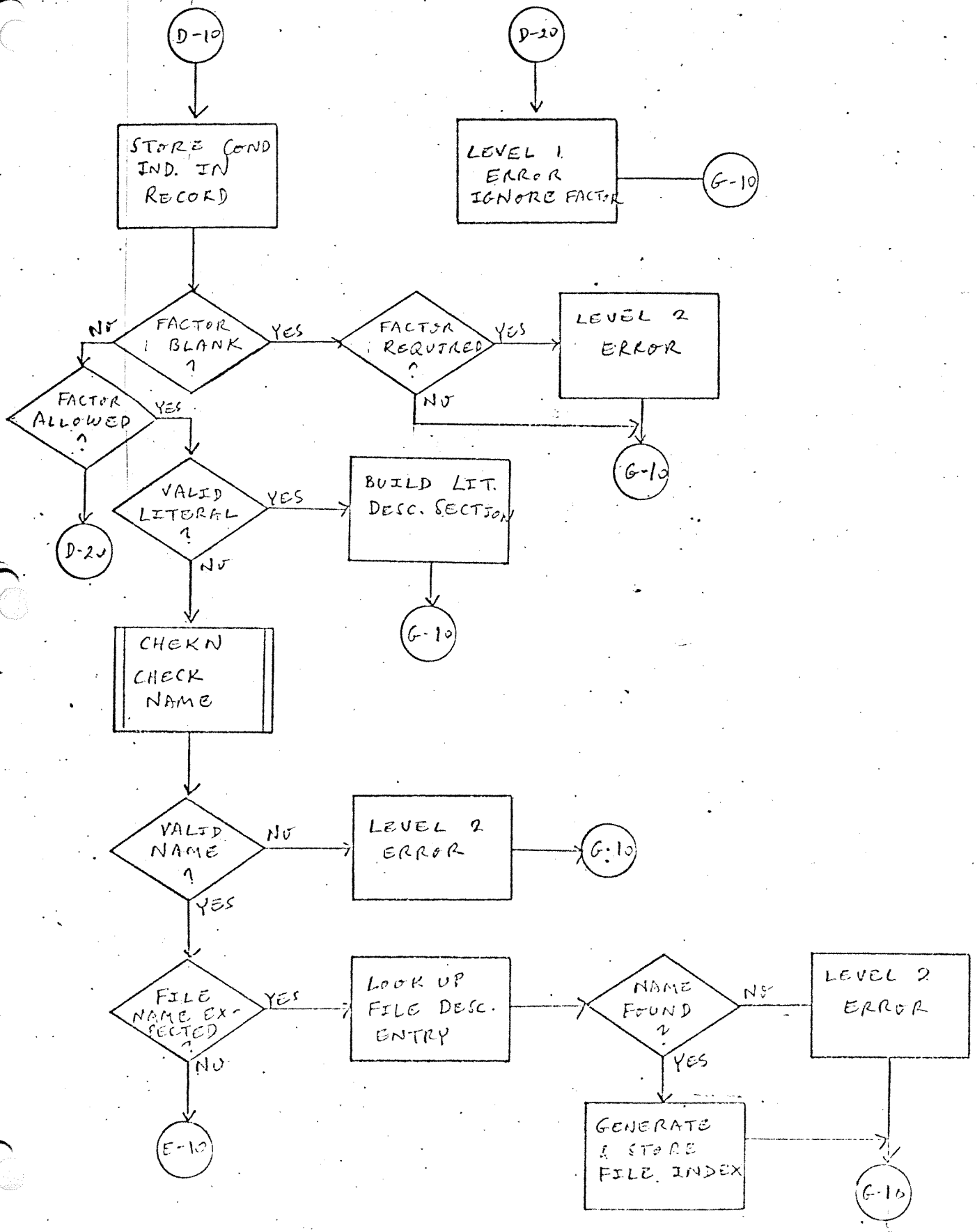
Calculation Specification Scan

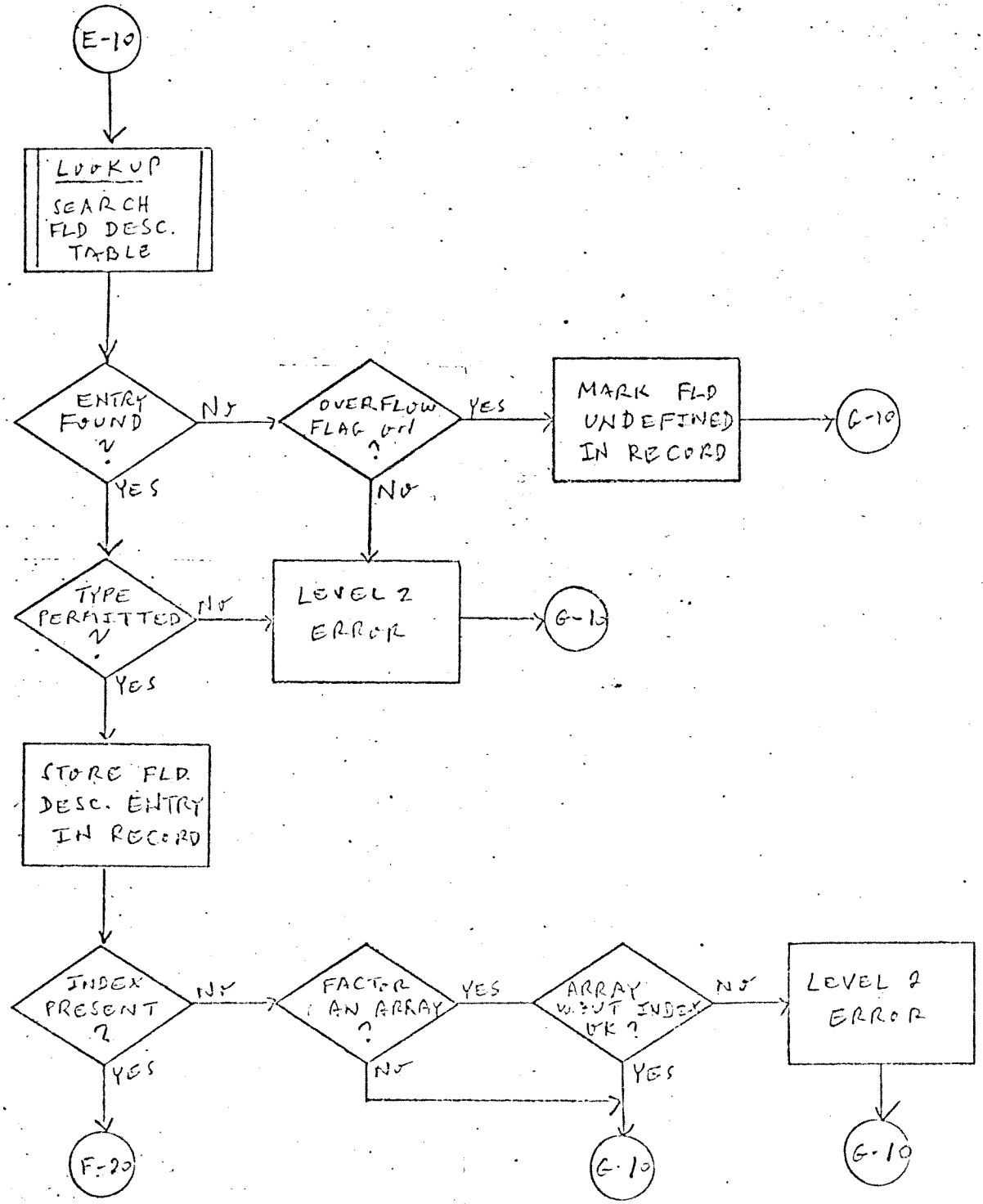
Flow Chart

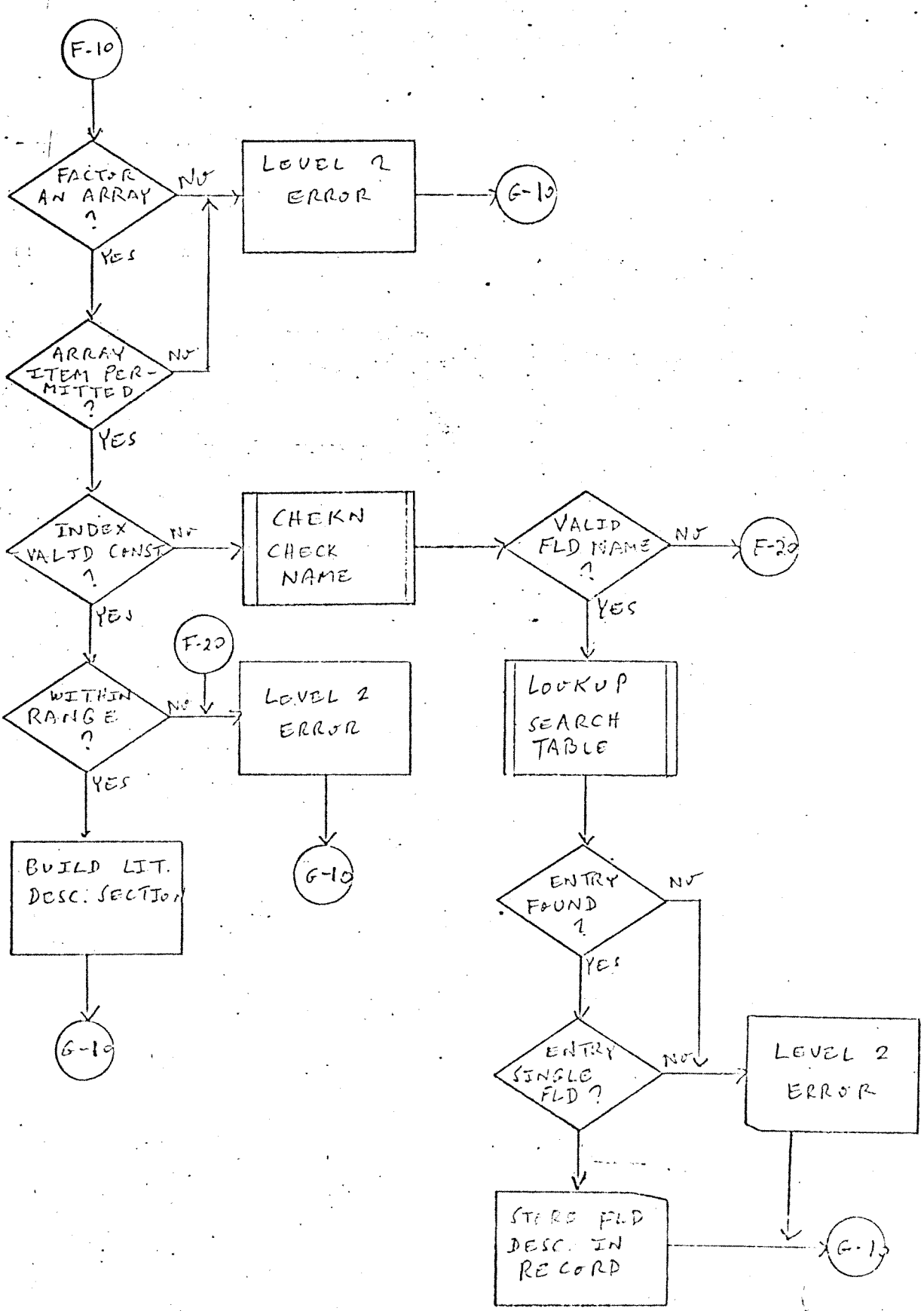



B-10
AN/OR IND. ON ?
YES
LEVEL 2 ERROR
RETURN TO EXECUTIVE
NO
CTL LEVEL LOW ?
YES
LEVEL 3 ERROR
NO
CTL. LEVEL HIGH ?
YES
WRITE & STORE CTL BREAK REC.
NO
B-20
COL. 9-17 BLANK ?
YES
COMMAND BLANK ?
YES
LEVEL 1 ERROR (WARNING)
C-20
NO
D-10
NO
VALID COND. IND. ?
NO
YES
COND. IND. PERMITED ?
NO
LEVEL 1 ERROR (WARNING
D-10
YES
STORE COND IND. IN RECORD
C-10

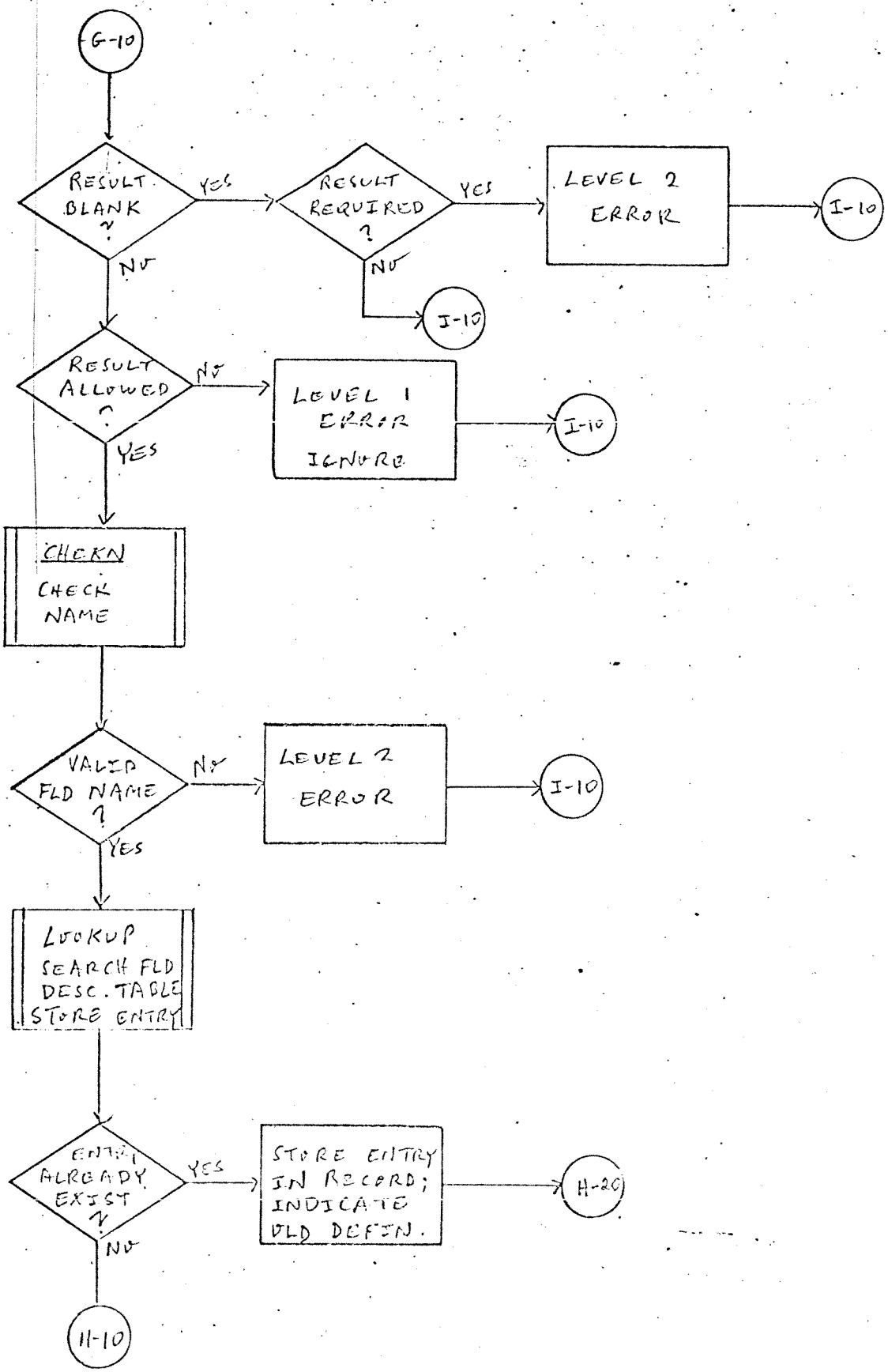

C-10
C-20
COMMAND BLANK ?
YES
COL. 18-59 BLANK ?
NO
A-20
NO
YES
D-10
AN/OR IND. ON ?
NO
SET AN/OR IND. SET COUNT TO 7
YES
REDUCE COUNT BY ONE
COUNT EXCEEDED ?
NO
STORE COND IND. IN AN/OR RECORD
YES
LEVEL 2 ERROR
RETURN TO EXECUTIVE
RETURN TO EXECUTIVE

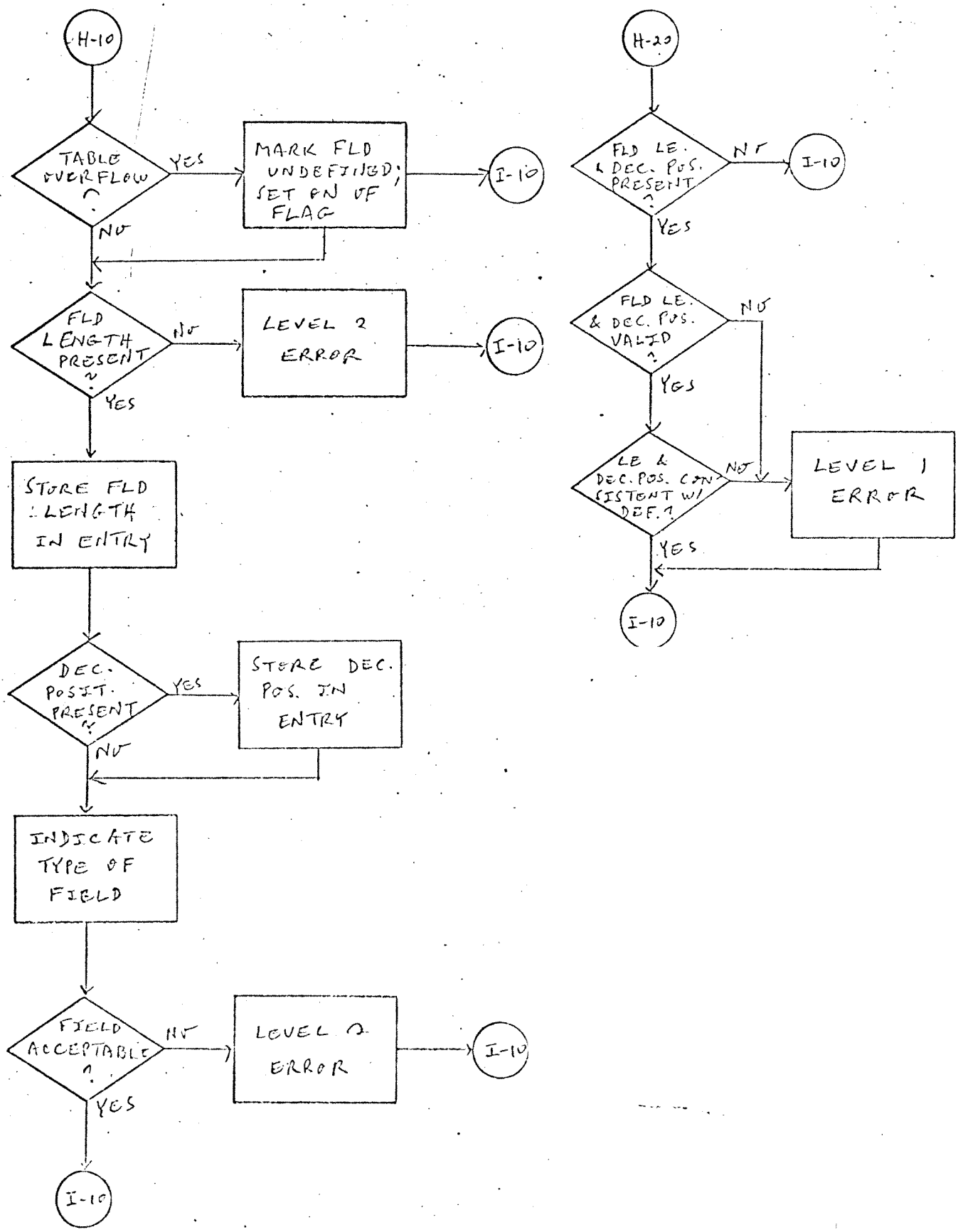

D-10
D-20
STORE COND IND. IN RECORD
LEVEL 1 ERROR IGNORE FACTOR
G-10
FACTOR 1 BLANK ?
NO
YES
FACTOR REQUIRED ?
YES
LEVEL 2 ERROR
NO
G-10
FACTOR ALLOWED ?
YES
D-20
VALID LITERAL ?
YES
BUILD LIT. DESC. SECTION
G-10
NO
CHEKN CHECK NAME
VALID NAME ?
NO
LEVEL 2 ERROR
G-10
YES
FILE NAME EXPECTED ?
YES
LOOK UP FILE DESC. ENTRY
NAME FOUND ?
NO
LEVEL 2 ERROR
YES
NO
E-10
GENERATE & STORE FILE INDEX
G-10

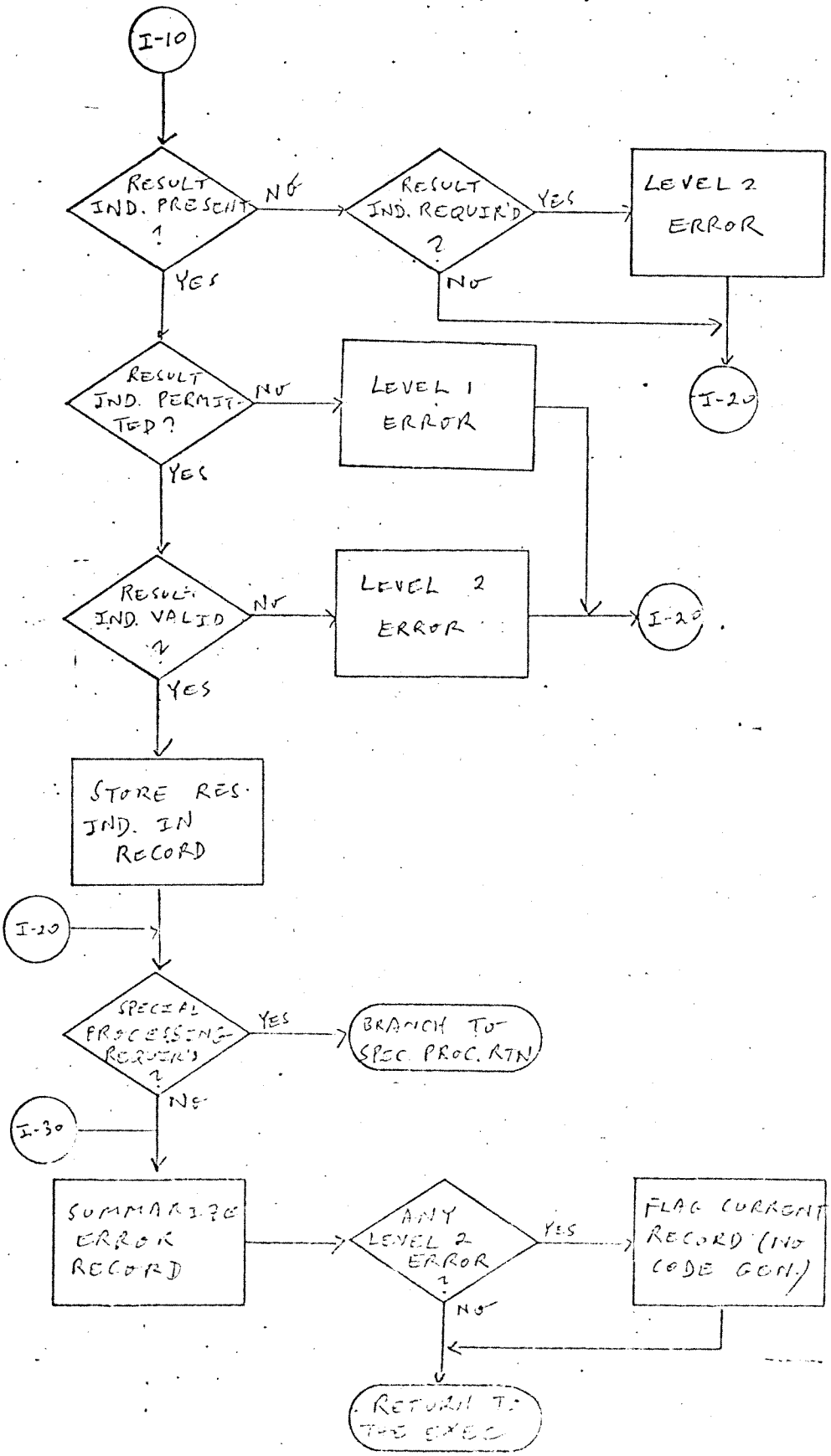

E-10
LOOKUP
SEARCH FLD DESC. TABLE
ENTRY FOUND ?
NO
YES
OVERFLOW FLAG on ?
YES
NO
MARK FLD UNDEFINED IN RECORD
G-10
TYPE PERMITTED ?
NO
YES
LEVEL 2 ERROR
G-10
STORE FLD DESC. ENTRY IN RECORD
INDEX PRESENT ?
NO
YES
FACTOR 1 AN ARRAY ?
YES
NO
ARRAY W/BUT INDEX OK ?
NO
YES
LEVEL 2 ERROR
F-20
G-10
G-10


F-10
FACTOR AN ARRAY ?
NO
LEVEL 2 ERROR
G-10
YES
ARRAY ITEM PERMITTED ?
NO
YES
INDEX VALID CONST ?
NO
CHEKN CHECK NAME
VALID FLD NAME ?
NO
F-20
YES
F-20
WITHIN RANGE ?
NO
LEVEL 2 ERROR
YES
LOOKUP SEARCH TABLE
BUILD LIT. DESC. SECTION
G-10
G-10
ENTRY FOUND ?
NO
YES
ENTRY SINGLE FLD ?
NO
LEVEL 2 ERROR
STORE FLD DESC. IN RECORD
G-10

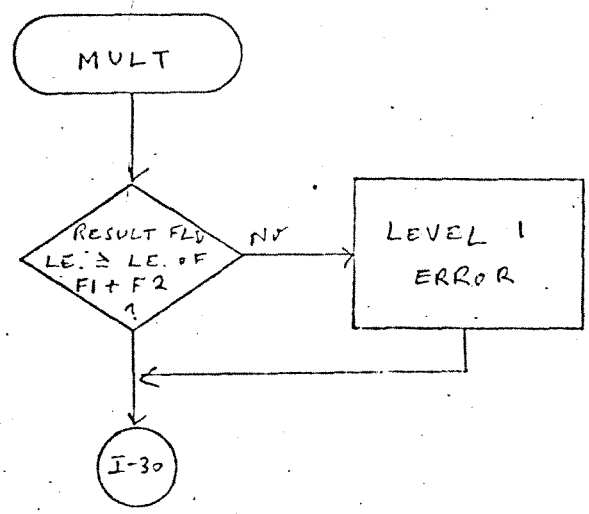
Calculation Spec's Scan (Continues)

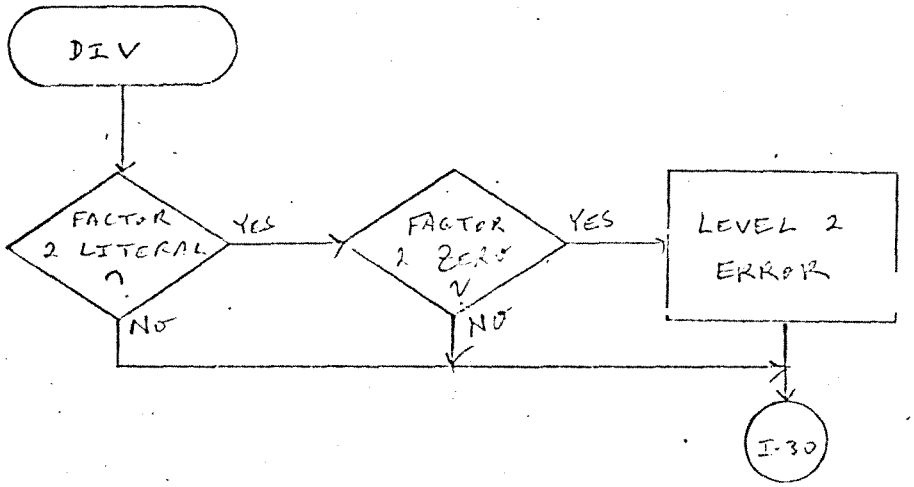


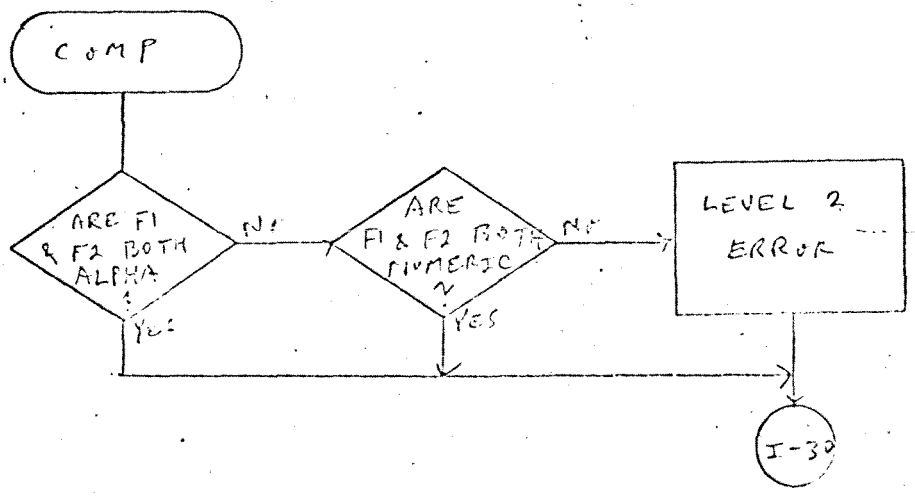

H-10
TABLE OVERFLOW ?
YES
MARK FLD UNDEFINED; SET ON OF FLAG
I-10
NO
FLD LENGTH PRESENT ?
NO
LEVEL 2 ERROR
I-10
YES
STORE FLD LENGTH IN ENTRY
DEC. POSIT. PRESENT ?
YES
STORE DEC. POS. IN ENTRY
NO
INDICATE TYPE OF FIELD
FIELD ACCEPTABLE ?
NO
LEVEL 2 ERROR
I-10
YES
I-10
H-20
FLD LE. & DEC. POS. PRESENT ?
NO
I-10
YES
FLD LE. & DEC. POS. VALID ?
NO
YES
LE & DEC. POS. CONSISTENT W/ DEF. ?
NO
LEVEL 1 ERROR
YES
I-10

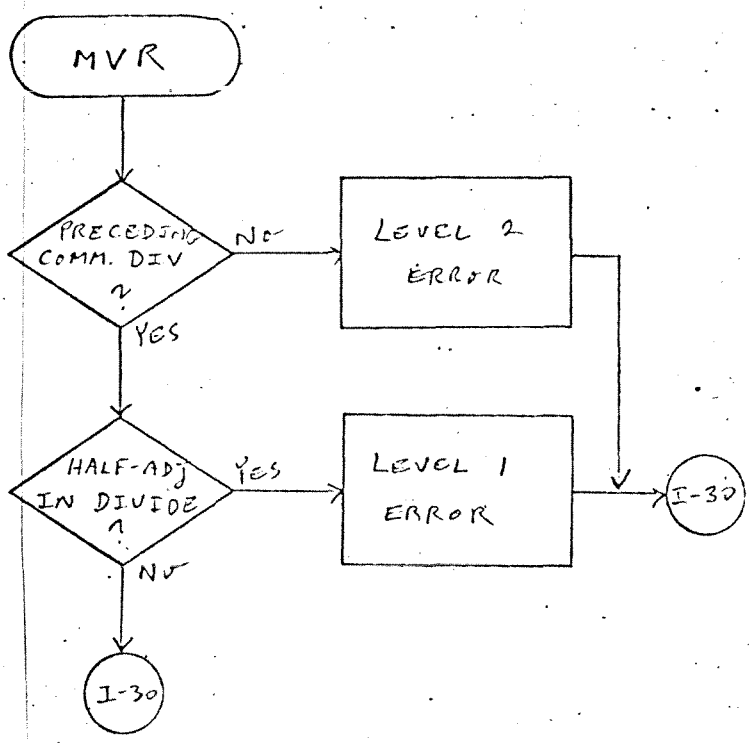

I-10
RESULT IND. PRESENT 1
NO
RESULT IND. REQUIR'D 2
YES
LEVEL 2 ERROR
NO
YES
I-20
RESULT IND. PERMITTED?
NO
LEVEL 1 ERROR
YES
RESULT IND. VALID 2
NO
LEVEL 2 ERROR
I-20
YES
STORE RES. IND. IN RECORD
I-20
SPECIAL PROCESSING REQUEST'D 2
YES
BRANCH TO SPEC PROC. RTN
NO
I-30
SUMMARIZE ERROR RECORD
ANY LEVEL 2 ERROR 2
YES
FLAG CURRENT RECORD (NO CODE GEN.)
NO
RETURN TO THE EXEC

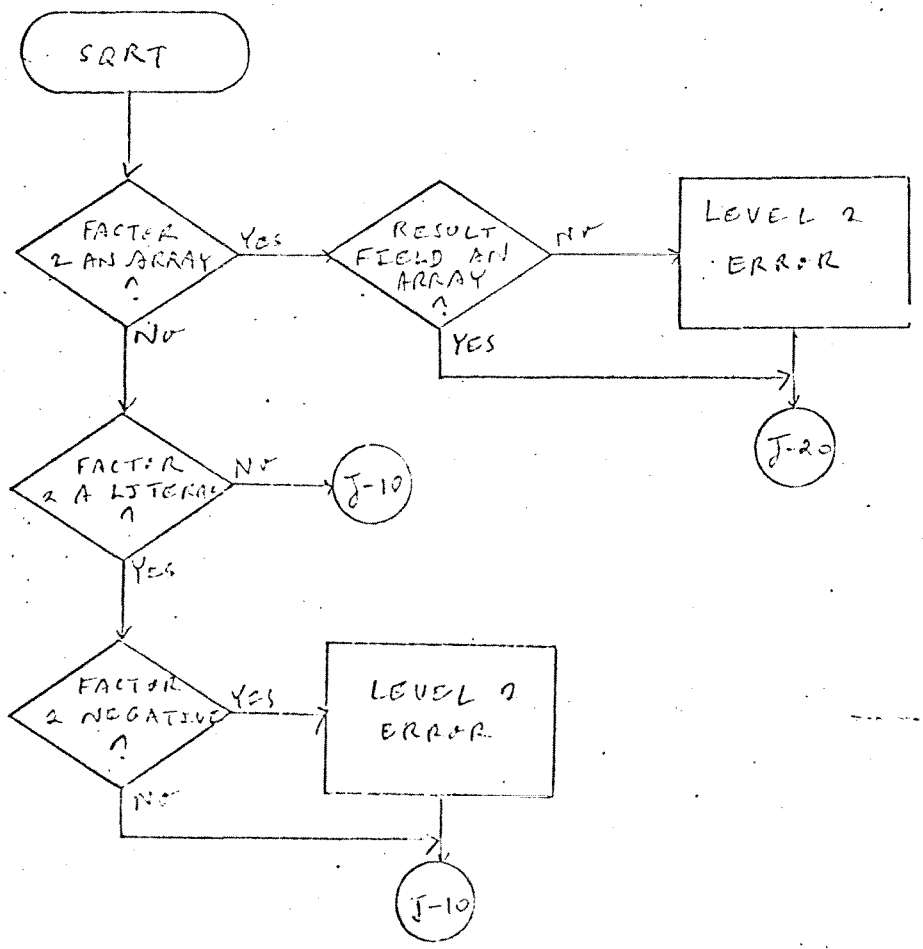
Special Processing Routines

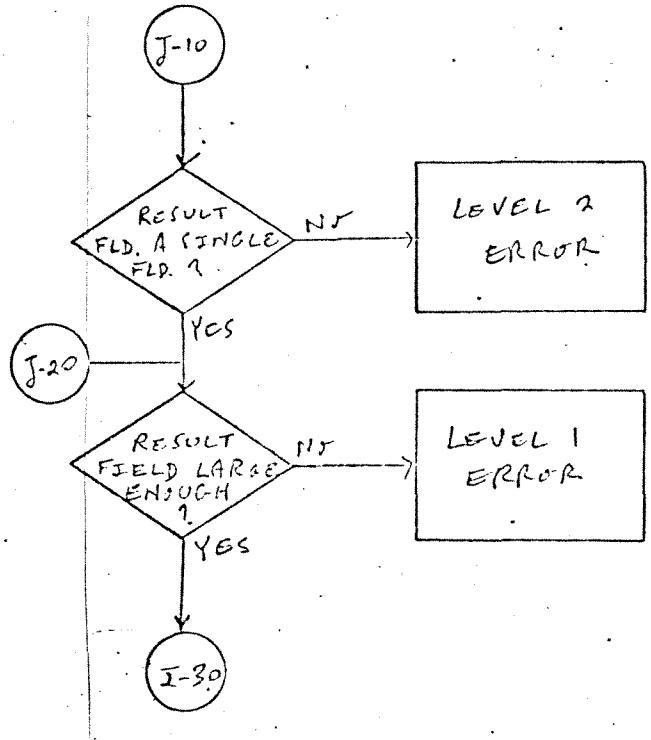


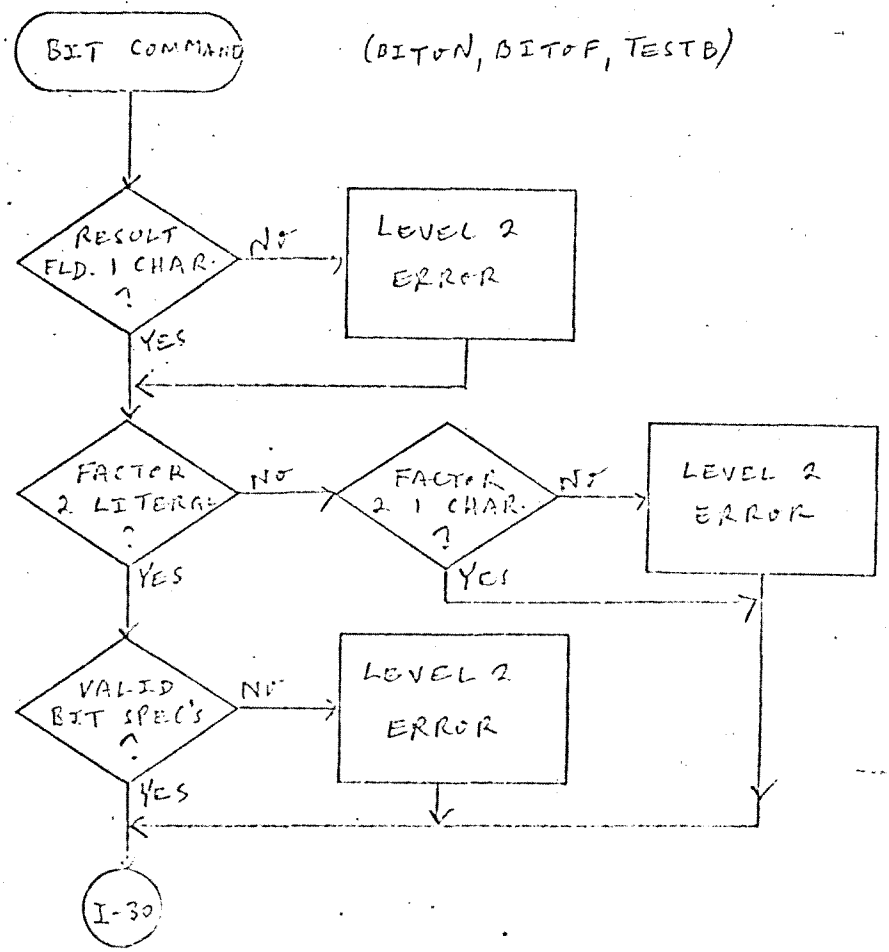


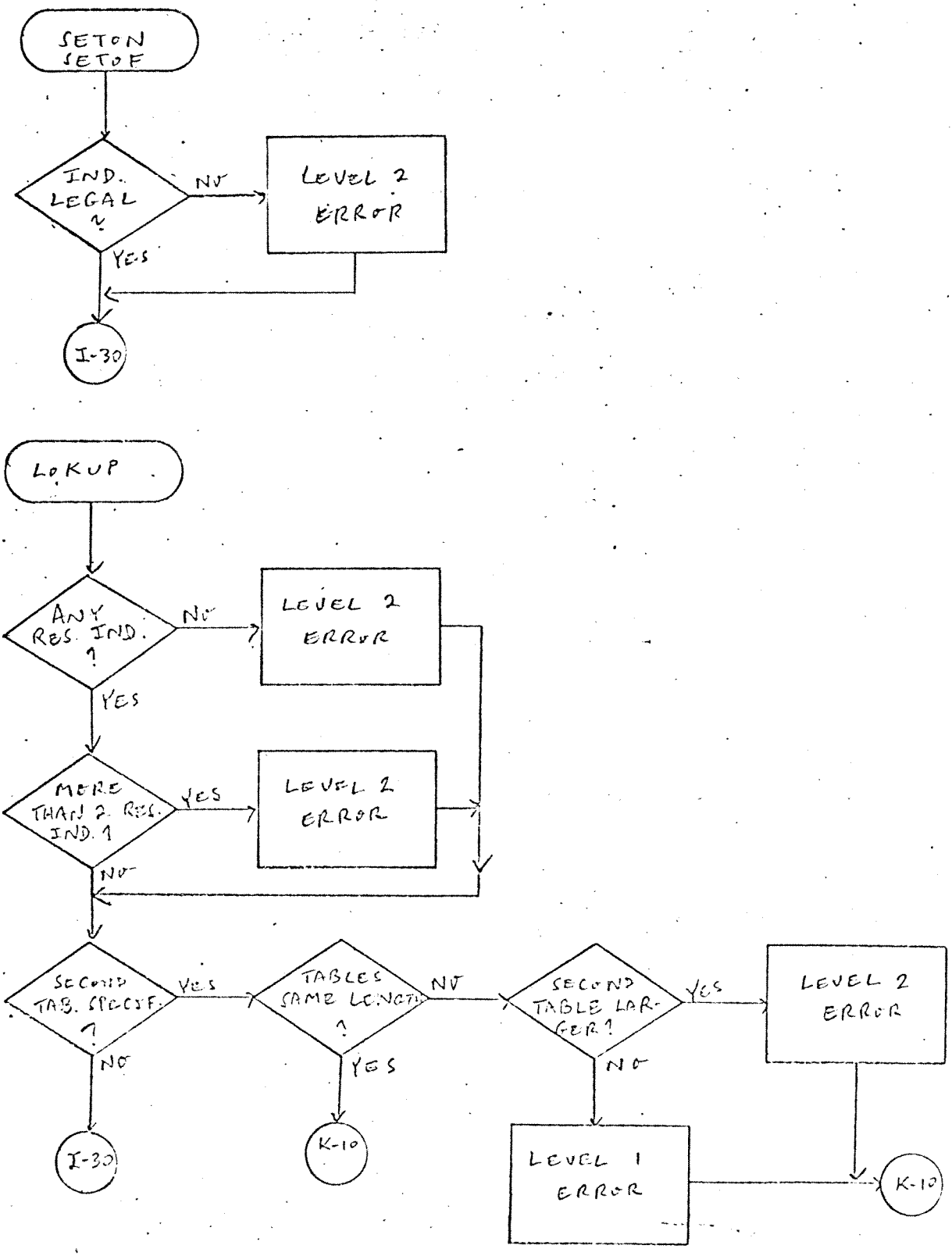


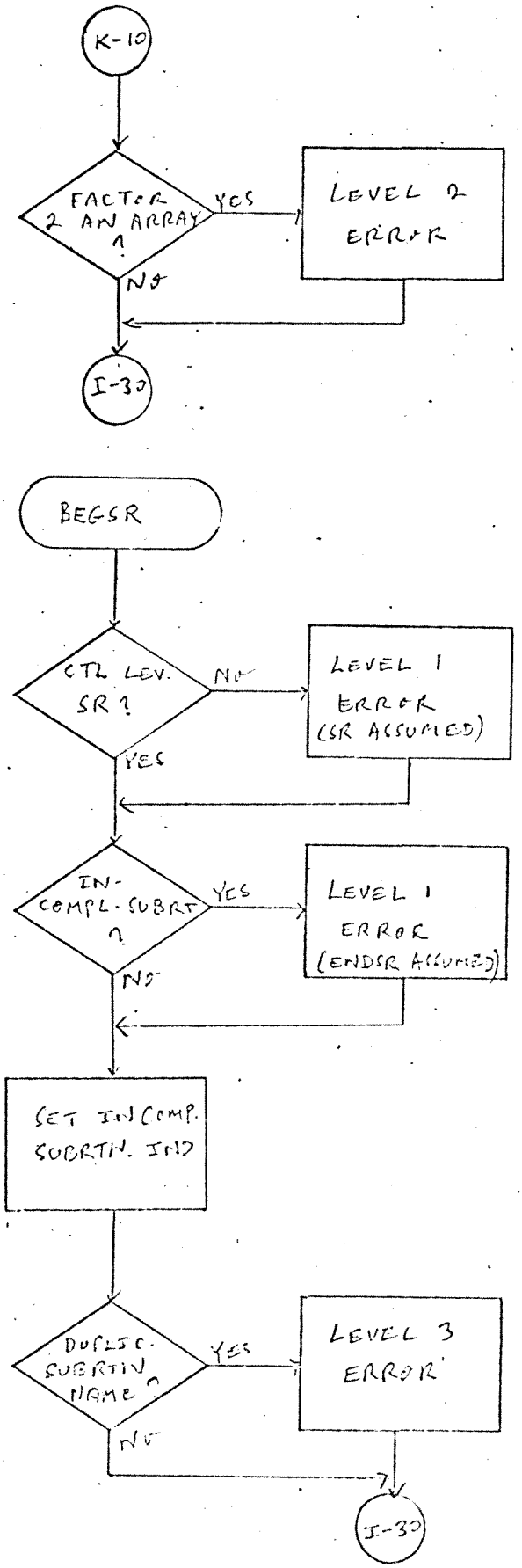

MVR
PRECEDING COMM. DIV ?
NO
LEVEL 2 ERROR
YES
HALF-ADJ IN DIVIDE ?
YES
LEVEL 1 ERROR
I-30
NO
I-30

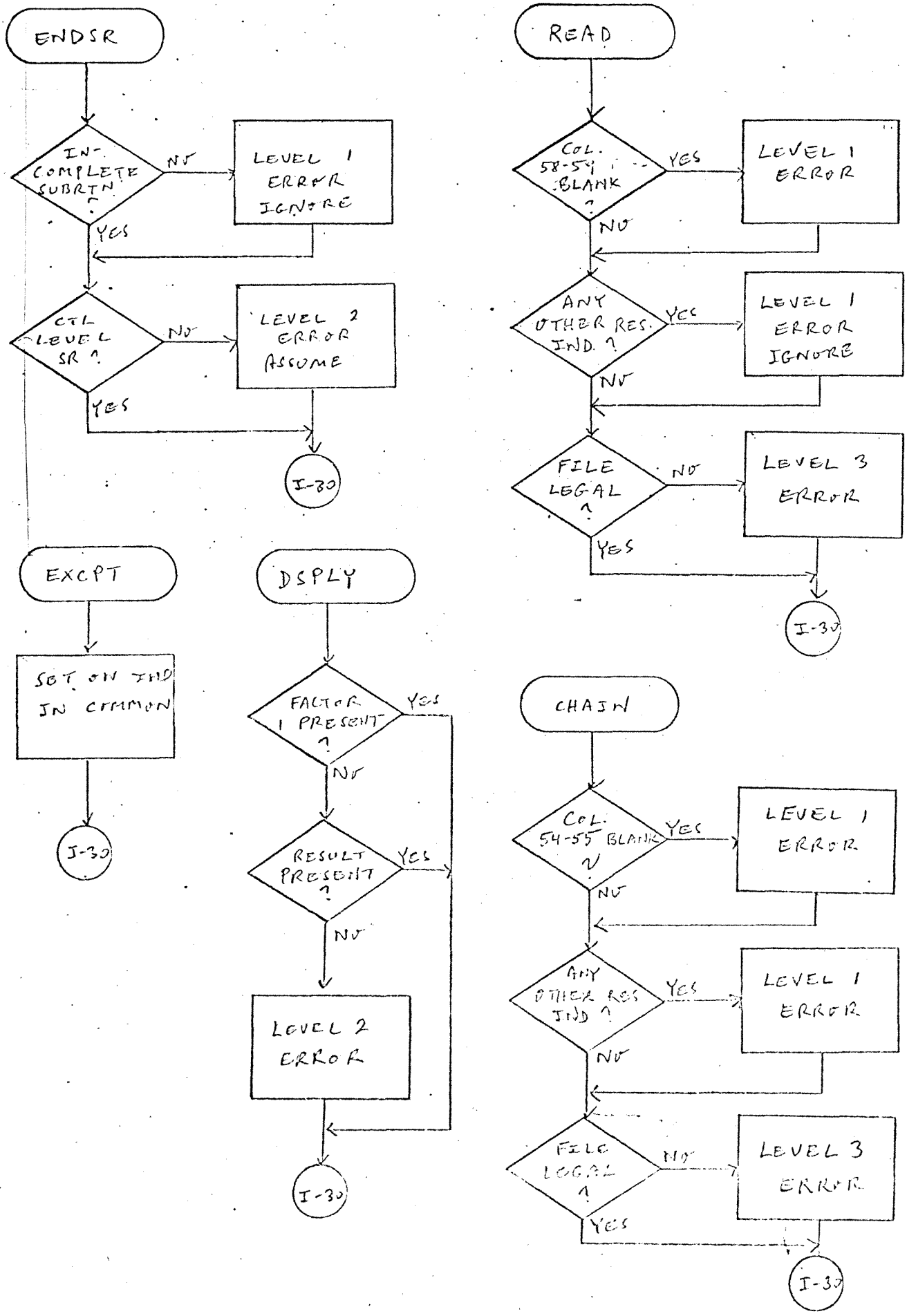

SQRT
FACTOR 2 AN ARRAY ?
YES
RESULT FIELD AN ARRAY ?
NO
LEVEL 2 ERROR
YES
J-20
NO
FACTOR 2 A LITERAL ?
NO
J-10
YES
FACTOR 2 NEGATIVE ?
YES
LEVEL 1 ERROR
NO
J-10


J-10
RESULT FLD. A SINGLE FLD. ?
NO
LEVEL 2 ERROR
YES
J-20
RESULT FIELD LARGE ENOUGH ?
NO
LEVEL 1 ERROR
YES
I-30


BIT COMMAND
(BITON, BITOF, TESTB)
RESULT FLD. 1 CHAR. ?
NO
LEVEL 2 ERROR
YES
FACTOR 2 LITERAL ?
NO
FACTOR 2 1 CHAR. ?
NO
LEVEL 2 ERROR
YES
YES
VALID BIT SPEC'S ?
NO
LEVEL 2 ERROR
YES
I-30


SETON
SETOF
IND. LEGAL ?
NO
LEVEL 2 ERROR
YES
I-30
LOKUP
ANY RES. IND. ?
NO
LEVEL 2 ERROR
YES
MORE THAN 2 RES. IND. ?
YES
LEVEL 2 ERROR
NO
SECOND TAB. SPECIF. ?
YES
TABLES SAME LENGTH ?
NO
SECOND TABLE LARGER ?
YES
LEVEL 2 ERROR
NO
YES
NO
I-30
K-10
LEVEL 1 ERROR
K-10


K-10
FACTOR 2 AN ARRAY ?
YES
LEVEL 2 ERROR
NO
I-30
BEGSR
CTL LEV. SR ?
NO
LEVEL 1 ERROR (SR ASSUMED)
YES
IN-COMPL-SUBRT ?
YES
LEVEL 1 ERROR (ENDSR ASSUMED)
NO
SET INCOMP. SUBRTN. IND
DUPLIC-SUBRTN NAME ?
YES
LEVEL 3 ERROR
NO
I-30


ENDSR
IN-COMPLETE SUBRTN ?
NO
LEVEL 1 ERROR IGNORE
YES
CTL LEVEL SR ?
NO
LEVEL 2 ERROR ASSUME
YES
I-30
READ
COL. 58-59 BLANK ?
YES
LEVEL 1 ERROR
NO
ANY OTHER RES. IND. ?
YES
LEVEL 1 ERROR IGNORE
NO
FILE LEGAL ?
NO
LEVEL 3 ERROR
YES
I-30
EXCPT
SET ON IND IN COMMON
I-30
DSPLY
FACTOR 1 PRESENT ?
YES
NO
RESULT PRESENT ?
YES
NO
LEVEL 2 ERROR
I-30
CHAIN
COL. 54-55 BLANK ?
YES
LEVEL 1 ERROR
NO
ANY OTHER RES IND ?
YES
LEVEL 1 ERROR
NO
FILE LEGAL ?
NO
LEVEL 3 ERROR
YES
I-30

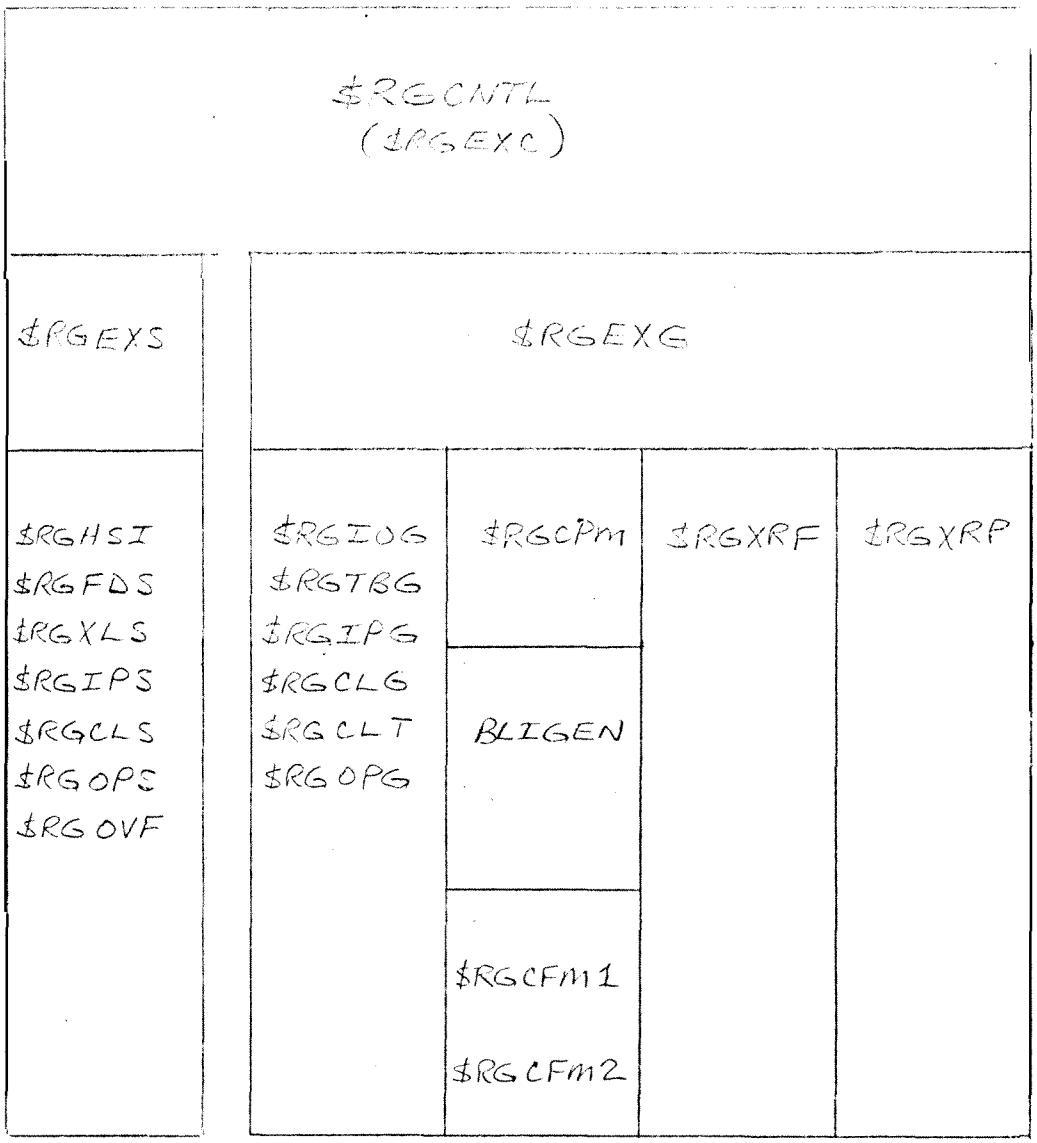
The RPG compiler has two primary phases or functions:

1. Syntatical evaluation of program specification cards and collection of attributes

2. Generation of object code (instructions and tables)

In addition, there are modules to handle table overflow conditions and to produce cross reference and error messages

The basic memory lay-out for the compiler modules is:



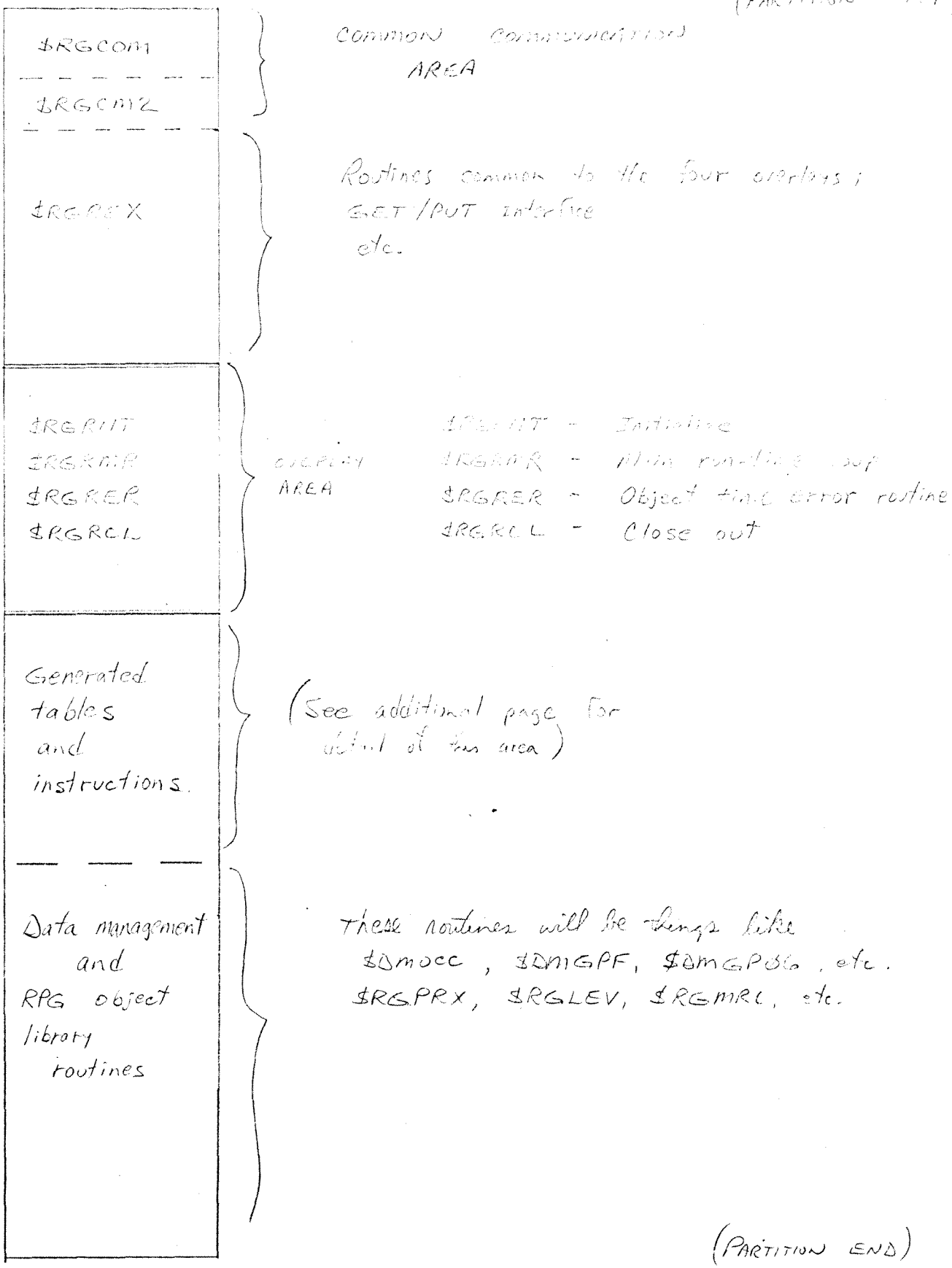
FFFF

7-17-73 RB

The memory layout for RPG object execution is:

0000

(PARTITION ...)



(PARTITION END)

FFFF

7-17-73 ...

BREAK DOWN OF [illegible] T[illegible]S AND [illegible] :

BCH $RGREX	
BDT's	One Buffer Description Table [illegible] for each file. (See Data management [illegible] for detail description.)
FCT's	File Control Tables : one for each file Contains attributes and status of files. (See F$xxx for table description.)
TABLE/ARRAY & LINE CONTROL TABLES	See T$ table description (and L$ for line counter table)
L/M [illegible] code	[illegible]
Regular Field moves code	Code generated to move <u>each</u> data field, of the selected record, to a data field work area.
RIT's Input Record Identification Tables	Tables containing information required to correctly identify each input record. (See R$ [illegible])
Calculations Code	Generated instructions to perform Detail and Total calculations.
Output Fields move code	Instructions generated to move the selected data fields to the record output buffer.
Output record Identification Table	Table information to determine if a particular record meets requirements for output
DATA HOLD AREA	Working storage for each field described.
INDICATORS	one byte for each indicator : 'FØ' = ON , 'ØØ' = OFF (Ø1-99 generated only if used ; others always reserved)
Buffer and Record areas	Data Management does Block I/O from/to <u>Buffers</u> and moves records to/from <u>Record</u> <u>AREAS</u>

REACTIVE COMPUTER SYSTEMS

DESIGN SPECIFICATION

Page

Date

RLB

Subject , . NF, MOD OBJECT CODE

FILE CONTROL TABLE (FCT)

see PAGE VI(8) in D.S.

FCT - fixed length, 40 bytes

+0	TYPE	TYP2
+2	TYP3	TYP4
+4	BDT ADDR	
+6	LRAD	
+8	RLEN	
+10	TRAN	
+12	USER	
+14	LINK	
+16	UN1	
+18	SWT	UN3

BASE FORMAT, FOLLOWED BY 1 OF 4 LAYOUTS SHOWN BELOWN

* INPUT

+20	PRI
+22	RID
+24	LOOK
+26	MACH
+28	LEV
+30	CRID
+32	
+34	
+36	
+38	RAFP

* OUTPUT

+20	LINE	
+22	HAD	
+24	T	
+26		
+28		
+30		
+32	MFCU	
+34	WORK	
+36	SKIP	STAT
+38	OVER	

* RECORD ADDRESS

+20	ELEN
+22	CLOC
+24	FPTR
+38	

* TABLE/ARRAY

+20	
+22	
+24	
+26	TCON

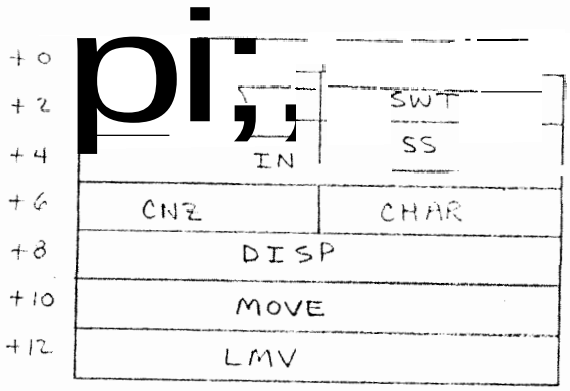
See page VI (13) in D.S.

T$

	TYP	ILEN
+2	STRT	
+4	NEXT	
+6	LEN	LEN1
+8	ENT	CNT1
+10	LELM	
+12	NUM	
+14	IN	
+16	OUT	
+18	LAST	
+20	UPER	LOWR
+22	FRST	

INPUT RECORD IDENTIFICATION TABLE

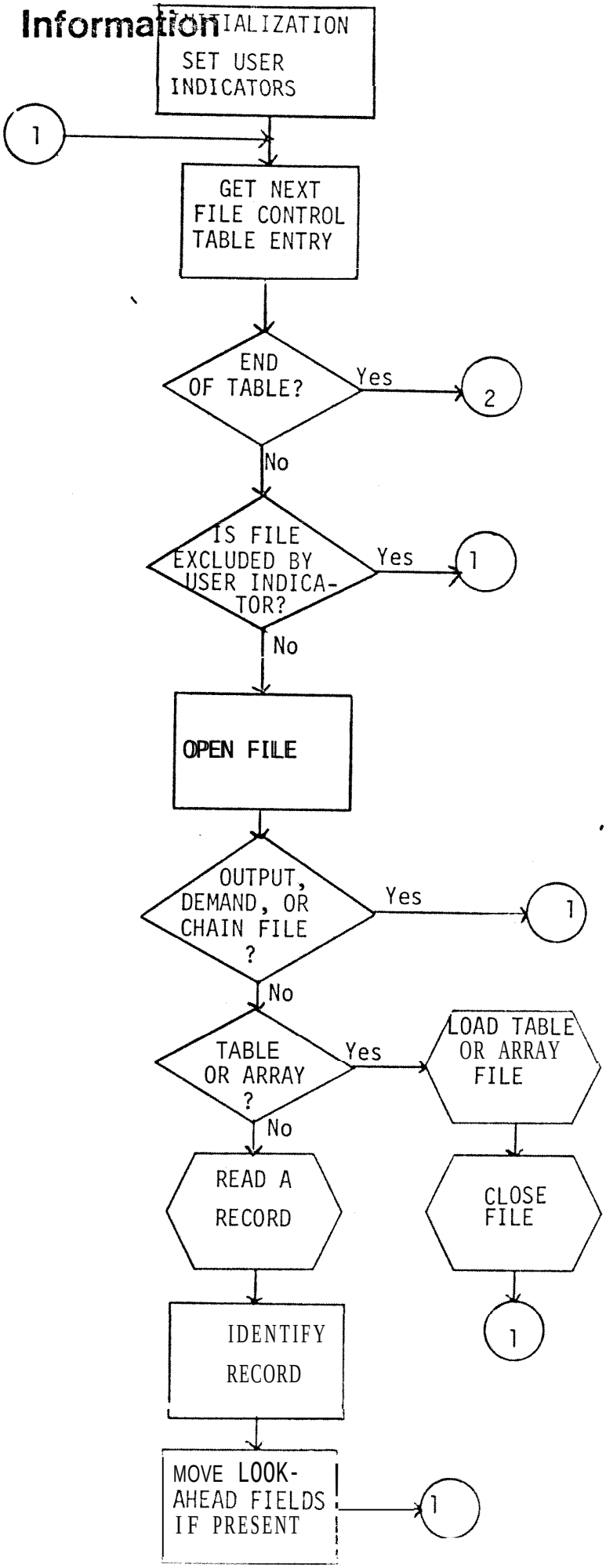
See PAGE VI(9) in D.S.

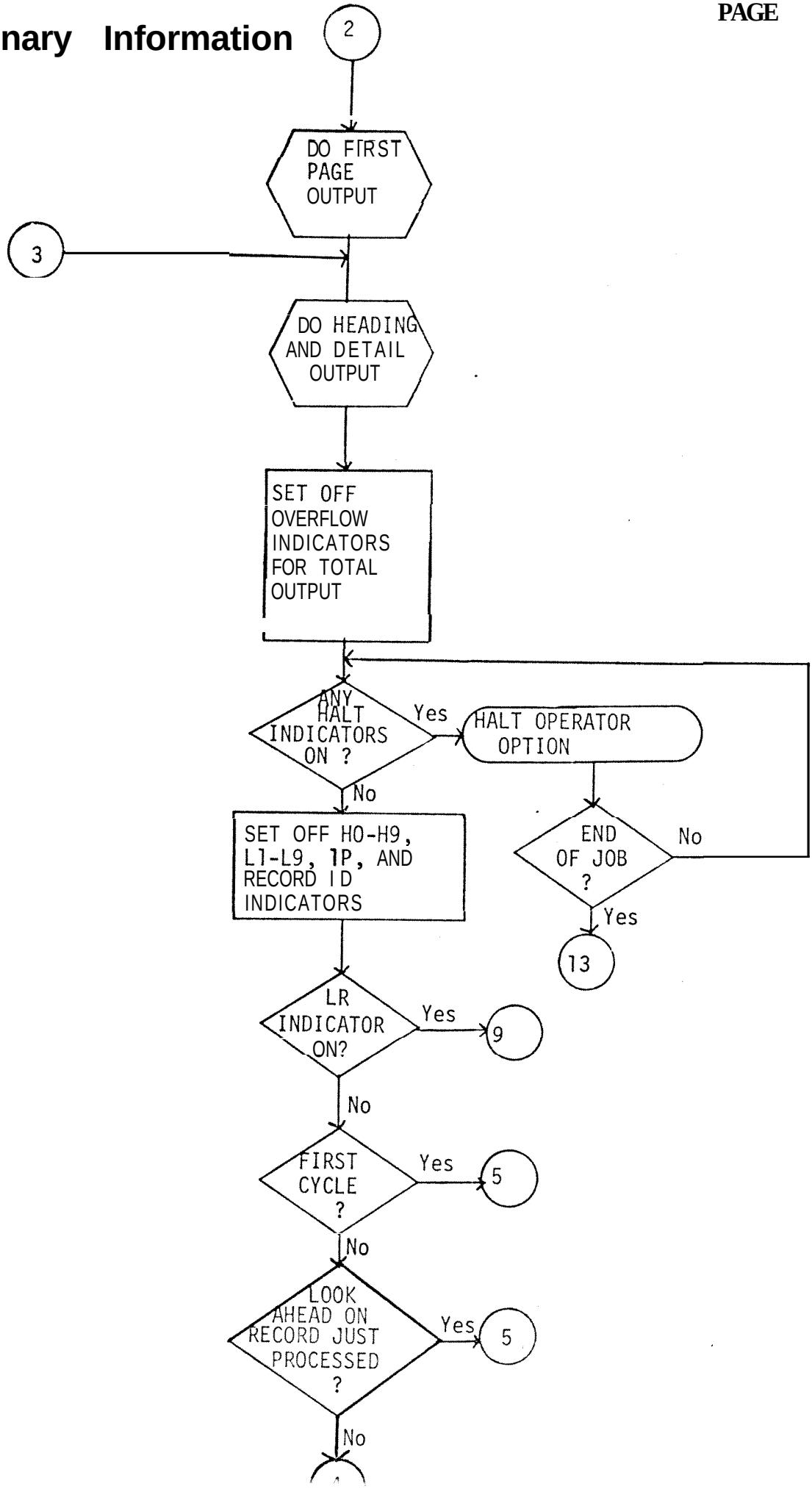


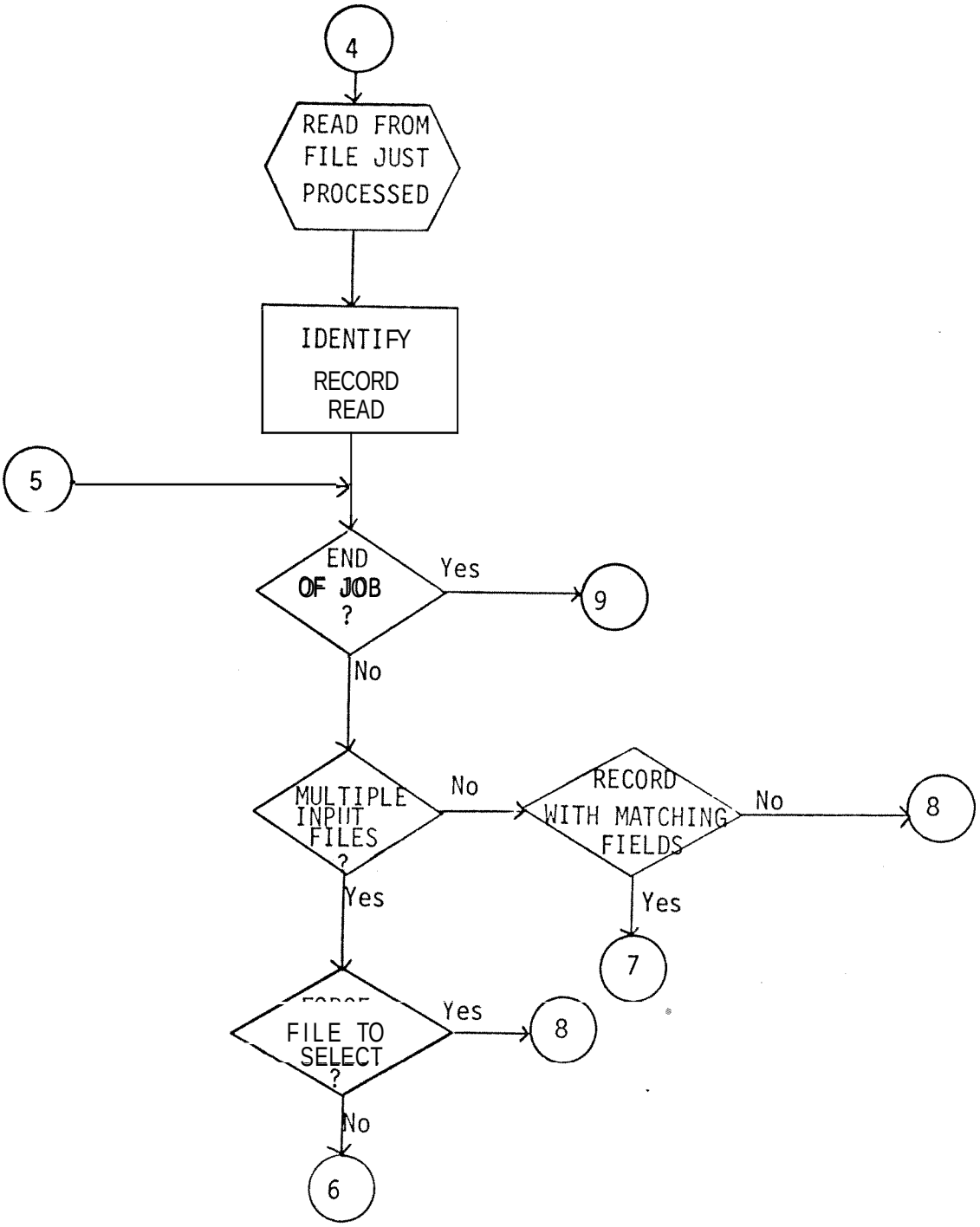
RPG MODULE DESCRIPTIONS

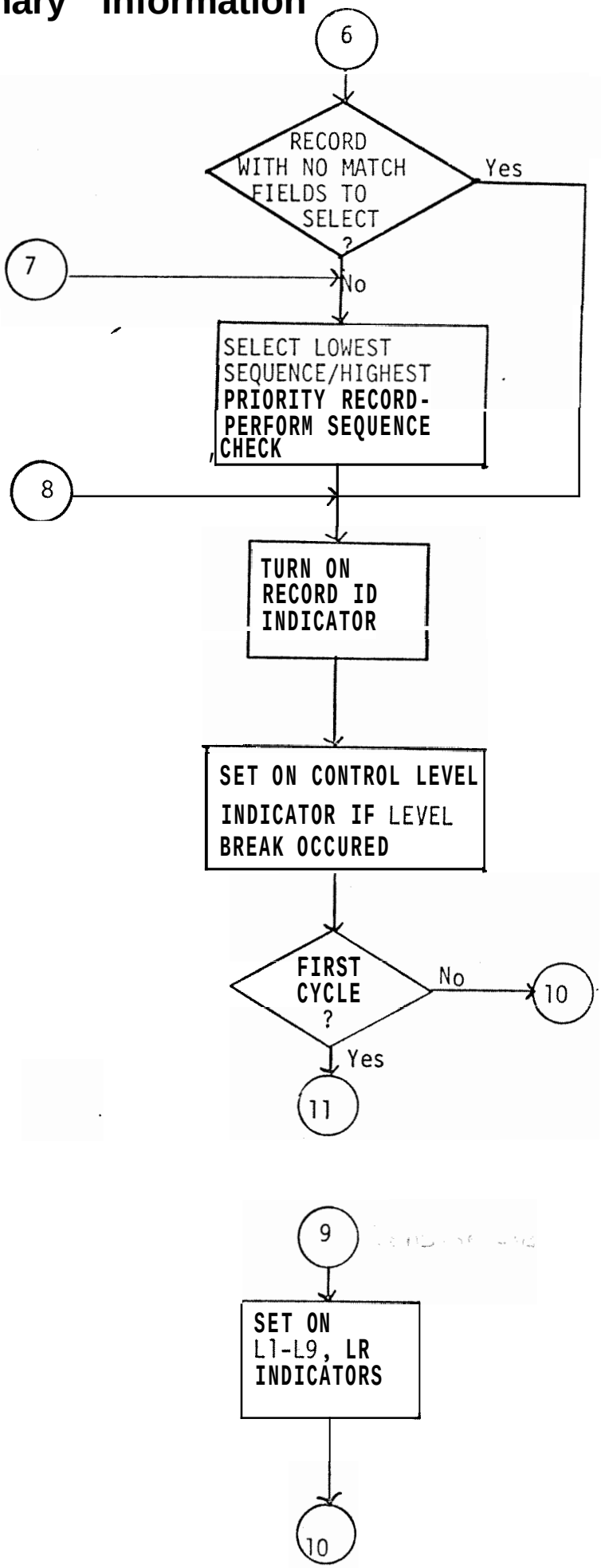
Routine Modules - $NUCLIB

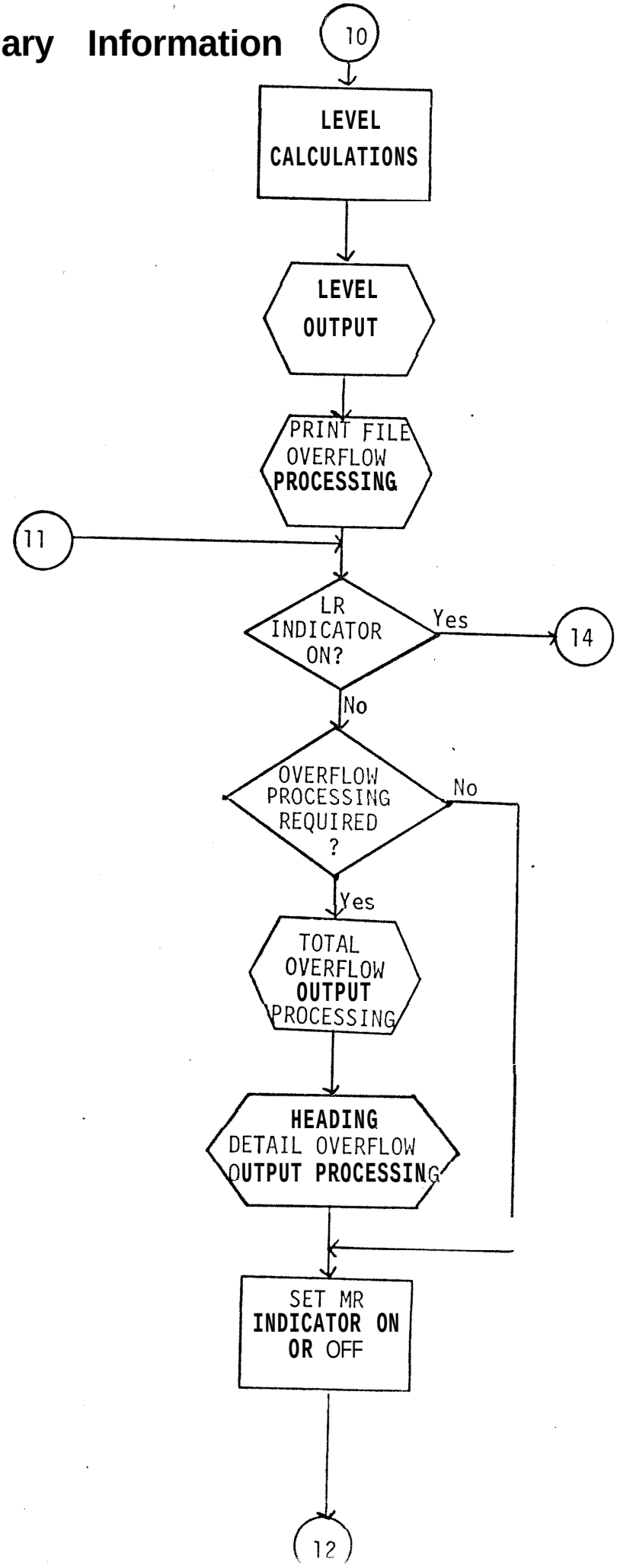
Object Library Member Name	Source Name	Description
$RGEXC	-	Run Time Executive Root
$RGEXI	-	Run Time Executive - Initialization Place
$RGEXE	-	Run Time Executive - Run Phase
$RGEXL	-	Run Time Executive - End of Job Phase
$RGMRL	-	Matching Record Processing
$RGLEV	-	Level Processing
$RGBP	-	Binary to Packed Conversion
$RGFIP	-	Numberic Field Test Routine
$RGFIA	-	Alpha Field Test Routine
$RGCVI	-	Array Processing
$RGCAN	-	Finish Pack Routine
$RGOFP	-	Fie1d Indicator Processing
$RGOFB	-	Convert to Binary Routine
$RGOAC	-	Array Output Routine
$RGOPL	-	*PLACE Processing
$RGAR1	-	ADD/SUB with half adjust
$RGAR2	-	Z-ADD/Z-SUB with half adjust
$RGCSO	-	Sign forcing for unpacked decimal
$RGHA	-	Half adjust routine
$RGTSB	-	Test bit routine
$RGTSZ	-	Test zone routine
$RGCHN	-	Chaining routine
$RGDBG	-	Debug processing
$RGDIV	-	Divide routine
$RGMUL	-	Multiply routine
$RGDSP	-	Display Routine
$RGREA	-	Read routine (from calculations)
$RGLPB	-	Binary Lookup of table/array
$RGLPS	-	Sequential lookup of tab1e/array
$RGSQR	-	Square Root Routine

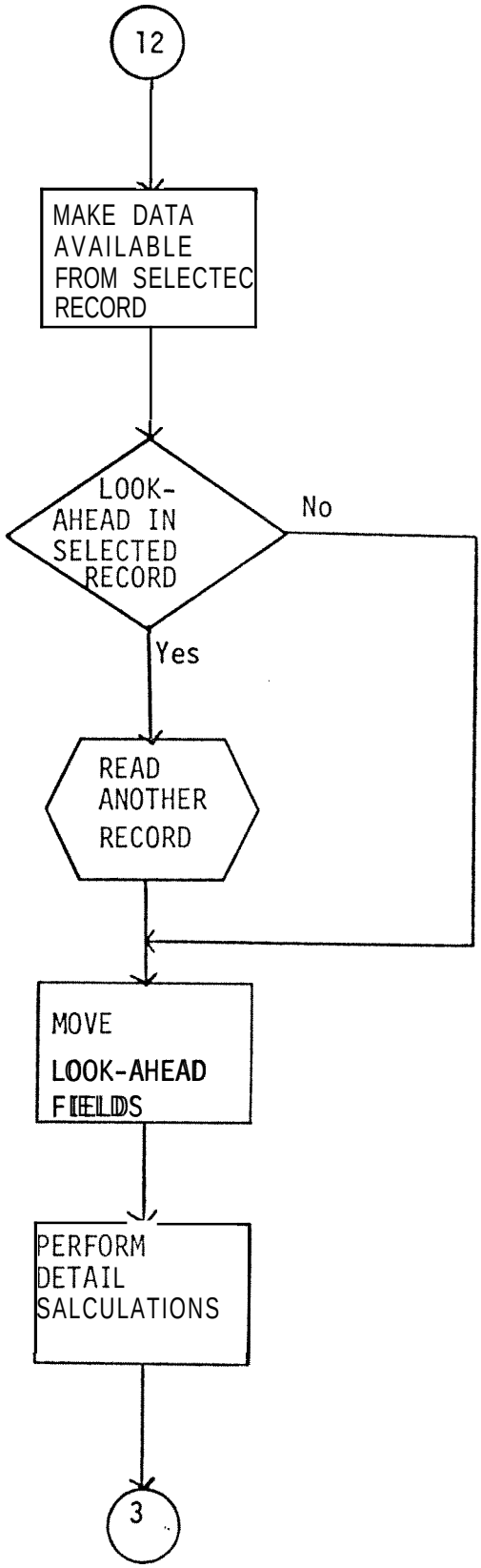


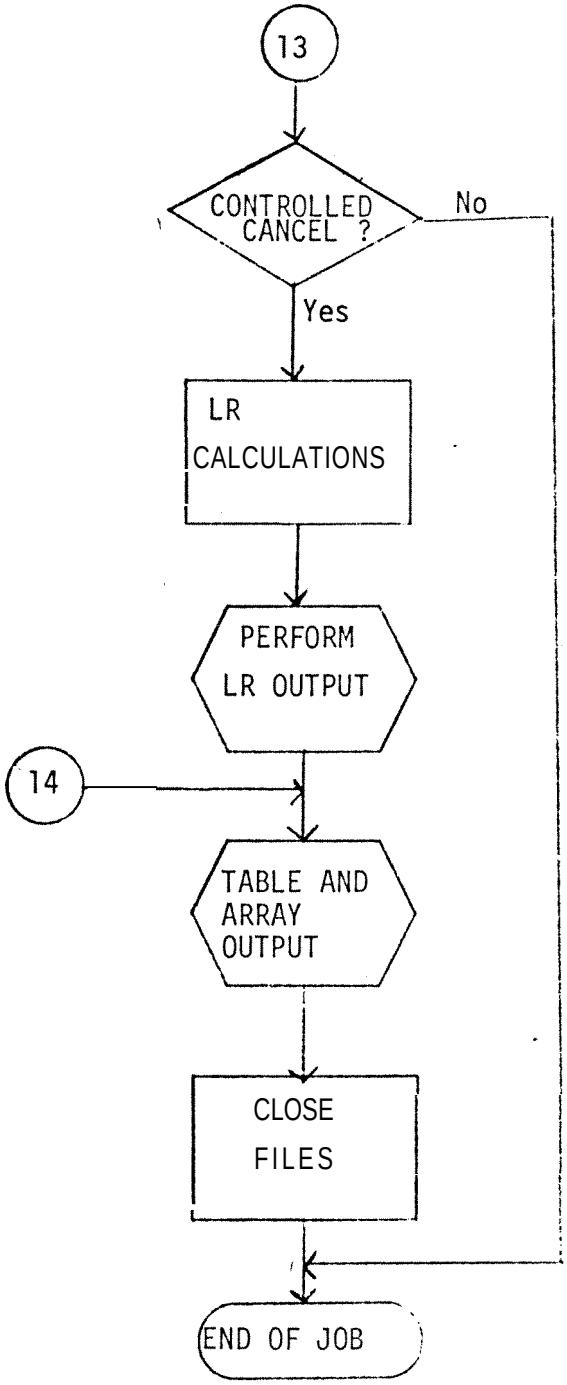

2
DO FIRST PAGE OUTPUT
3
DO HEADING AND DETAIL OUTPUT
SET OFF OVERFLOW INDICATORS FOR TOTAL OUTPUT
ANY HALT INDICATORS ON ?
Yes
HALT OPERATOR OPTION
No
END OF JOB ?
No
Yes
13
SET OFF H0-H9, L1-L9, 1P, AND RECORD ID INDICATORS
LR INDICATOR ON?
Yes
9
No
FIRST CYCLE ?
Yes
5
No
LOOK AHEAD ON RECORD JUST PROCESSED ?
Yes
5
No


4
READ FROM FILE JUST PROCESSED
IDENTIFY RECORD READ
5
END OF JOB ?
Yes
9
No
MULTIPLE INPUT FILES ?
No
RECORD WITH MATCHING FIELDS
No
8
Yes
Yes
7
FILE TO SELECT ?
Yes
8
No
6


6
RECORD WITH NO MATCH FIELDS TO SELECT ?
Yes
7
No
SELECT LOWEST SEQUENCE/HIGHEST PRIORITY RECORD-PERFORM SEQUENCE CHECK
8
TURN ON RECORD ID INDICATOR
SET ON CONTROL LEVEL INDICATOR IF LEVEL BREAK OCCURED
FIRST CYCLE ?
No
10
Yes
11
9
SET ON L1-L9, LR INDICATORS
10


10
LEVEL CALCULATIONS
LEVEL OUTPUT
PRINT FILE OVERFLOW PROCESSING
11
LR INDICATOR ON?
Yes
14
No
OVERFLOW PROCESSING REQUIRED ?
No
Yes
TOTAL OVERFLOW OUTPUT PROCESSING
HEADING DETAIL OVERFLOW OUTPUT PROCESSING
SET MR INDICATOR ON OR OFF
12


12
MAKE DATA AVAILABLE FROM SELECTEC RECORD
LOOK-AHEAD IN SELECTED RECORD
No
Yes
READ ANOTHER RECORD
MOVE LOOK-AHEAD FIELDS
PERFORM DETAIL SALCULATIONS
3


13
CONTROLLED CANCEL ?
No
Yes
LR CALCULATIONS
PERFORM LR OUTPUT
14
TABLE AND ARRAY OUTPUT
CLOSE FILES
END OF JOB

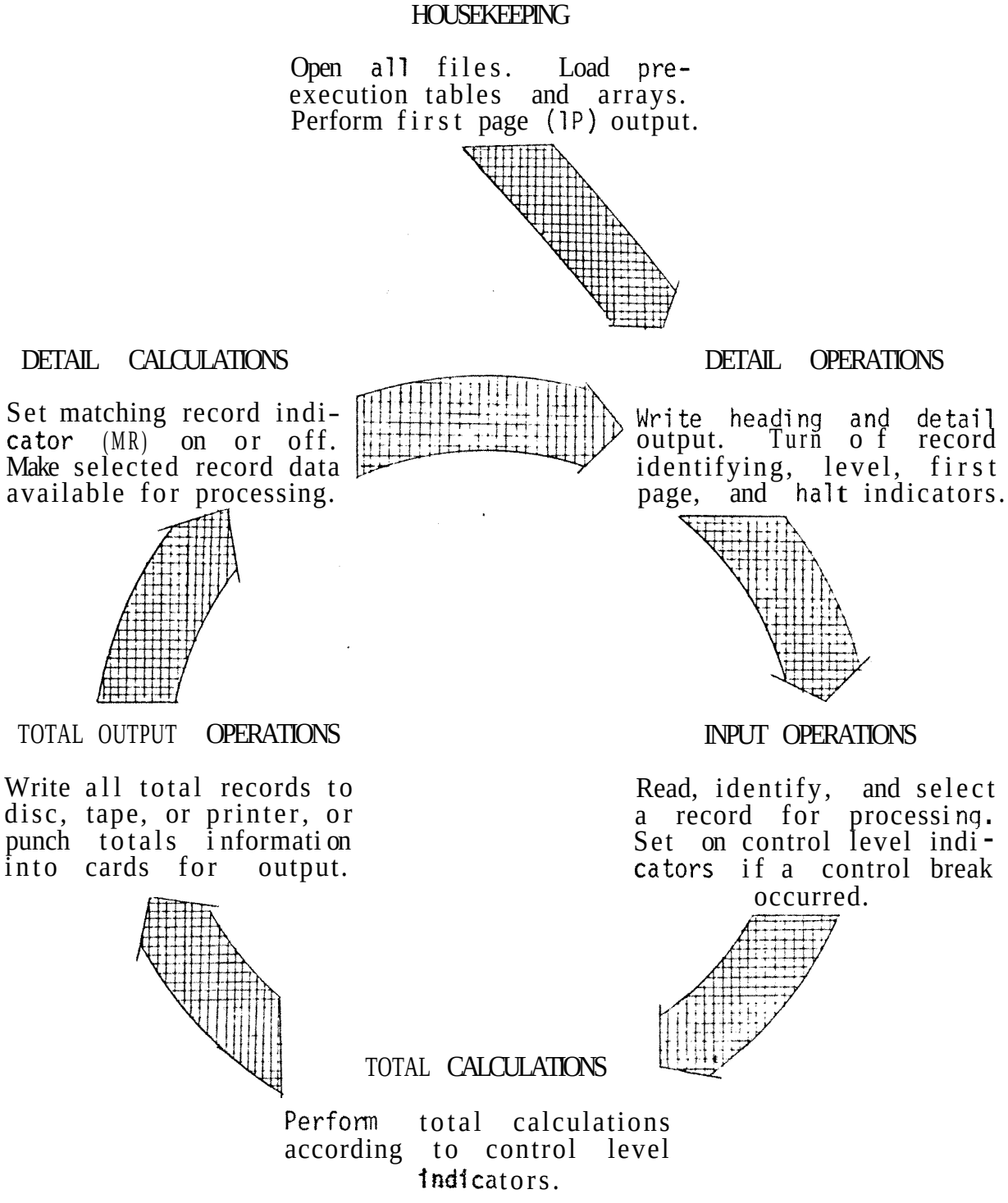
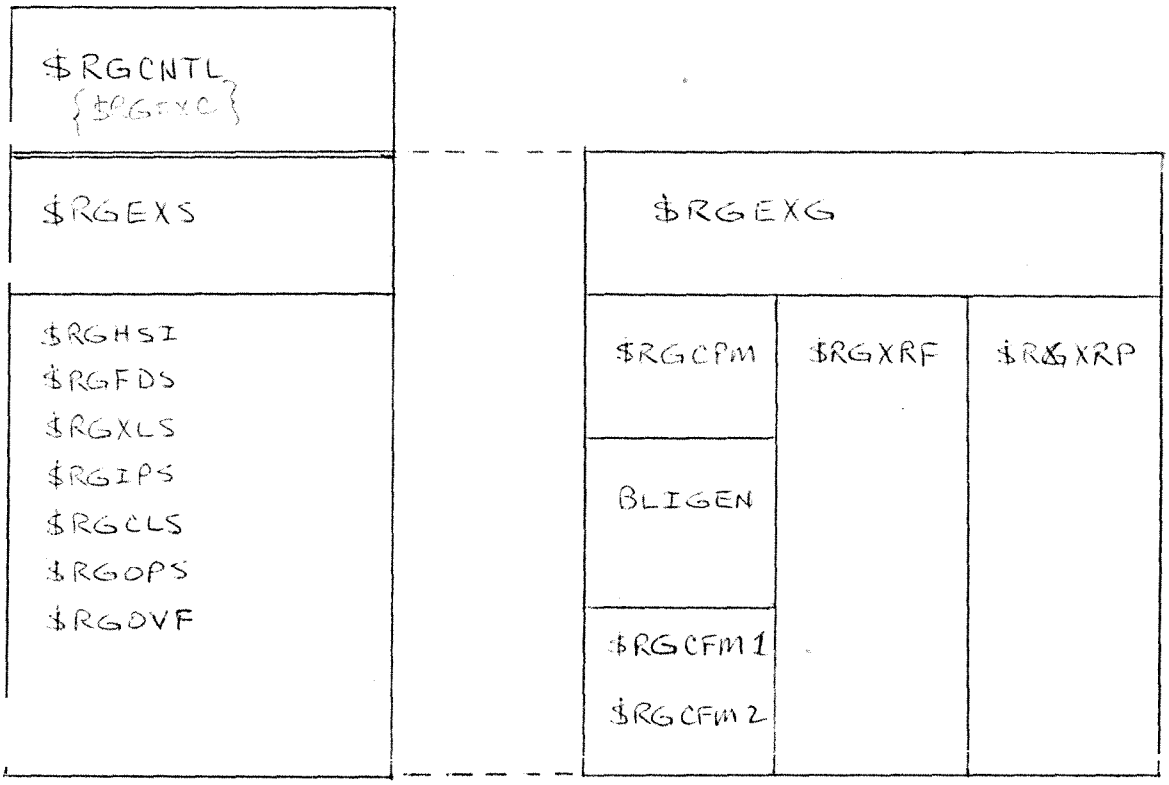


Figure 1-4. Object Program Operating Cycle.

RPG MODULE DESCRIPTIONS

Compiler Modules - $SYSLODLIB

Load Library Member	Object Library Member	Source - indicates same	Description
RPG	$RGCNTL	-	Compiler Executive
" -	$RGHSN	-	Common Initialization
-	$RGHSI	-	Header Card Scan, File Allocations
" -	$RGSCS EXS	-	Scan Common Subroutines
$RGFDS	-	-	File Description Scan
$RGXLS	-	-	File Extension /Line Counter Scan
$RGIPS	-	-	Input Description Scan
$RGCLS	-	-	Calculation Description Scan
$RGOPS	-	-	Output Description Scan
$RGOVF	-	-	Table Overflow Phase
$RGIOG	$RGGCS EXG	-	Generation Common Subroutines
" -	$RGIOG	-	Input/Output Generator
$RGTBG	-	-	Table/Array Generator
$RGIPG	-	-	Input Processing Generator
$RGCLG	-	-	Calculation Generator - Pass 1
$RGCLT	-	-	Calculation Generator - Pass 2
$RGOPG	-	-	Output Generator
$RGCFM	$RGCFRM	-	Code Formatter - Root
" -	$RGCFM1	-	Code Formatter - Listing
-	$RGCFE M2	-	Code Formatter - End of job processing
-	$RGBLG BLIGCN	-	Buffered Library Generator
$RGXRF	-	-	Cross Reference - Pass 1
$RGXRP	-	-	Cross Reference - Pass 2


$RGCNTL
{$RGEXC}
$RGEXS
$RGHSI
$RGFDS
$RGXLS
$RGIPS
$RGCLS
$RGOPS
$RGOVF
$RGEXG
$RGCPM
$RGXRF
$RGXRP
BLIGEN
$RGCFM1
$RGCFM2

MEMOREX

RPG II Specifications

Punching Instructions

Graphic

Punch

Date_____ Page____ of____

Programmer_____

Program_____

Control Card

1 PAGE NUMBER	3 LINE NUMBER	6 TYPE OF FORM	12 STORAGE NEEDED TO EXECUTE	15 DEBUG CODE	21 D/H/J INVERTED PRINT	26 S/T COLLATING SEQUENCE	34 B TABLE LOOKUP	40 N/S/I/O/3 SIGN PROCESS	42 1 FORMS POSITIONING	43 F FILE TRANSLATION	47 S SUPPRESS SKIP - CHAN. 1	52 X CROSS REFERENCE LIST	53 L CARRIAGE CONTR. TYPE	54 SQUEEZE-SEQ. CHECK	55 7 IGNORE ARITH. O'FLOW	PROGRAM IDENTIFICATION
	01	H														

File Description

1 PAGE NUMBER	3 LINE NUMBER	6 TYPE OF FORM	FILENAME	15 I/O/U/D TYPE OF FILE	16 P/S/C/R/T/D DESIGNATION	17 E END-OF-FILE CODE	18 A/D SEQUENCE	19 F/V FILE FORMAT	LENGTH OF BLOCK	LENGTH OF RECORD	29 L/R PROCESSING MODE	LENGTH OF KEY FIELD OR RECORD ADDR. FIELD	31 A/I RECORD ADDR. TYPE	32 I/T/1/9 FILE ORG.	33 OA,OG,OV O'FLOW IND	STARTING LOCATION OF KEY FIELD	39 E/L EXTENSION CODE	DEVICE	HIGH-LEVEL DIRECTORY SIZE	TAPE LABELS	INDEX BUFFER SIZE	66 A FILE ADDITION	70 N/U TAPE REWIND	71 U1 - U8 FILE CONDITION	PROGRAM IDENTIFICATION
	02	F																DISC							
	03	F																PUNCH							
	04	F																TAPE							
	05	F																READ							
	06	F																CONSOLE							
	07	F	ADDROUT															PRINTER							

MEMOREX

RPG II Specifications

Punching Instructions

Graphic

Punch

Date _____ Page _____ of _____

Programmer _____

Program _____

Input

PAGE NUMBER	LINE NUMBER	TYPE OF FORM	FILENAME	OR / AND	SEQUENCE	N NUMBER	O OPTION	RECORD INDICATOR	RECORD CODE CHARACTERS: FIRST POSITION	FIRST N NOT	FIRST C/Z/D PORTION	FIRST CHARACTER	RECORD CODE CHARACTERS: SECOND POSITION	SECOND N NOT	SECOND C/Z/D PORTION	SECOND CHARACTER	RECORD CODE CHARACTERS: THIRD POSITION	THIRD N NOT	THIRD C/Z/D PORTION	THIRD CHARACTER	STACKER SELECT	P/B PACKED OR BINARY FIELD	FIELD LOCATION: BEGINS	FIELD LOCATION: ENDS	DECIMAL POSITIONS	FIELD NAME	L1-L9 CONTROL LEVEL	M1-M9 MATCH FIELDS	FIELD RECORD RELATION INDICATOR	FIELD INDICATORS: PLUS	FIELD INDICATORS: MINUS	FIELD INDICATORS: ZERO OR BLANK		PROGRAM IDENTIFICATION
1 2	3 4 5	6	7 8 9 10 11 12 13	14	15 16	17	18	19 20	21 22 23 24	25	26	27	28 29 30 31	32	33	34	35 36 37 38	39	40	41	42	43	44 45 46 47	48 49 50 51	52	53 54 55 56 57 58	59 60	61 62	63 64	65 66	67 68	69 70	71 72 73 74	75 76 77 78 79 80
	01	I																																
	02	I																																
	03	I																																
	04	I																																
	05	I																																
	06	I																																
	07	I																																
	08	I																																
	09	I																																
	10	I																																
	11	I																																
	12	I																																
	13	I																																
	14	I																																
	15	I																																

227L003 PRINTED IN U.S.A.

MEMOREX

RPG II Specifications

Punching Instructions

Graphic

Punch

Date_____ age_____ of_____

Programmer_____

Program_____

Calculation

PAGE NUMBER	LINE NUMBER	TYPE OF FORM	CONTROL LEVEL L0-L9/LR/SR/AN, OR	OPERATION INDICATORS: FIRST N NOT ON	FIRST INDICATOR	SECOND N NOT ON	SECOND INDICATOR	THIRD N NOT ON	THIRD INDICATOR	FACTOR 1: FIELD NAME, TABLE, ARRAY, ARRAY ITEM, OR DATA LITERAL	OPERATION	FACTOR 2: FIELD NAME, TABLE, ARRAY, ARRAY ITEM, OR DATA LITERAL	RESULT FIELD: FIELD NAME, TABLE, ARRAY, OR ARRAY ITEM	RESULT FIELD LENGTH	DECIMAL POSITIONS	H HALF ADJUST	RESULTING INDICATORS: ARITHMETIC PLUS	ARITHMETIC MINUS	ARITHMETIC ZERO	COMPARE HIGH	COMPARE LOW	COMPARE EQUAL	LOOK-UP TABLE (FACTOR 2) IS HIGH	LOOK-UP TABLE (FACTOR 2) IS LOW	LOOK-UP TABLE (FACTOR 2) IS EQUAL	COMMENTS	PROGRAM IDENTIFICATION
1 2	3 4 5	6	7 8	9	10 11	12	13 14	15	16 17	18 19 20 21 22 23 24 25 26 27	28 29 30 31 32	33 34 35 36 37 38 39 40 41 42	43 44 45 46 47 48	49 50 51	52	53	54	55	56	57	58	59	60	61	62	63 64 65 66 67 68 69 70 71 72 73 74	75 76 77 78 79 80
	0 1	C																									
	0 2	C																									
	0 3	C																									
	0 4	C																									
	0 5	C																									
	0 6	C																									
	0 7	C																									
	0 8	C																									
	0 9	C																									
	1 0	C																									
	1 1	C																									
	1 2	C																									
	1 3	C																									
	1 4	C																									
	1 5	C																									
1 2	3 4 5	6	7 8	9	10 11	12	13 14	15	16 17	18 19 20 21 22 23 24 25 26 27	28 29 30 31 32	33 34 35 36 37 38 39 40 41 42	43 44 45 46 47 48	49 50 51	52	53	54	55	56	57	58	59	60	61	62	63 64 65 66 67 68 69 70 71 72 73 74	75 76 77 78 79 80

22/1204 PRINTED IN U.S.A.

MEMOREX

RPG II Specifications

Punching Instructions

Graphic

Punch

Date_____ Page_____ of_____

Programmer_____

Program_____

Output-Format

PAGE NUMBER	LINE NUMBER	TYPE OF FORM	FILENAME	OR / AND	ADD	STACKER/OFLOW	BEFORE SPACE	AFTER SPACE	BEFORE SKIP	AFTER SKIP	OUTPUT INDICATORS FIRST N NOT ON	FIRST INDICATOR	SECOND N NOT ON	SECOND INDICATOR	THIRD N NOT ON	THIRD INDICATOR	FIELD NAME	EDIT CODE	BLANK AFTER	ENDING POSITION	P/B/Z/4 PACKED/BINARY	EDIT WORD OR CONSTANT	PROGRAM IDENTIFICATION
1 2	3 4 5	6	7 8 9 10 11 12 13	14	15	16	17	18	19 20	21 22	23	24 25	26	27 28	29	30 31	32 33 34 35 36 37	38	39	40 41 42 43	44	45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70	71 72 73 74 75 76 77 78 79 80
	0 1	O																					
	0 2	O																					
	0 3	O																					
	0 4	O																					
	0 5	O																					
	0 6	O																					
	0 7	O																					
	0 8	O																					
	0 9	O																					
	1 0	O																					
	1 1	O																					
	1 2	O																					
	1 3	O																					
	1 4	O																					
	1 5	O																					

COMMA	ZERO BALANCES TO PRINT	NO SIGN	CR	–
YES	YES	1	A	J
YES	NO	2	B	K
NO	YES	3	C	L
NO	NO	4	D	M

X = REMOVE PLUS SIGN
Y = DATE FIELD EDIT
Z = ZERO SUPPRESS

MEMOREX

RPG II Specifications

Punching Instructions

Graphic

Punch

Date_____ Page_____ of_____

Programmer_____

Program_____

Extension

PAGE NUMBER	LINE NUMBER	TYPE OF FORM	"FROM" FILENAME	"TO" FILENAME	TABLE OR ARRAY NAME	NUMBER OF ITEMS ON EACH RECORD	NUMBER OF ITEMS IN TABLE OR ARRAY	LENGTH OF ITEM	P/B PACKED/BINARY	DECIMAL POSITIONS	A/D SEQUENCE	ALTERNATING TABLE OR ARRAY: TABLE OR ARRAY NAME (ALTERNATING FORMAT)	ALTERNATING TABLE OR ARRAY: LENGTH OF ITEM	ALTERNATING TABLE OR ARRAY: PACKED/BINARY	ALTERNATING TABLE OR ARRAY: DECIMAL POSITIONS	ALTERNATING TABLE OR ARRAY: A/D SEQUENCE	COMMENTS	PROGRAM IDENTIFICATION
1 2	3 4 5	6	7 8 9 10	11 12 13 14 15 16 17 18	19 20 21 22 23 24 25 26	27 28 29 30 31 32	33 34 35	36 37 38 39	40 41 42	43	44	45	46 47 48 49 50 51	52 53 54	55	56	57	58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74	75 76 77 78 79 80
	0 1	E																
	0 2	E																
	0 3	E																
	0 4	E																
	0 5	E																
	0 6	E																
	0 7	E																

Line Counter

PAGE NUMBER	LINE NUMBER	TYPE OF FORM	FILENAME	FIRST: LINES AVAILABLE OR LINE NO.	FIRST: FORM LENGTH (FL) OR CHANNEL NO.	SECOND: OVERFLOW LINE NO. OR LINE NO.	SECOND: OVERFLOW LINE (OL) OR CHAN. NO.	THIRD: LINE NO.	THIRD: CHANNEL NO.	FOURTH: LINE NO.	FOURTH: CHANNEL NO.	FIFTH: LINE NO.	FIFTH: CHANNEL NO.	SIXTH: LINE NO.	SIXTH: CHANNEL NO.	SEVENTH: LINE NO.	SEVENTH: CHANNEL NO.	EIGHTH: LINE NO.	EIGHTH: CHANNEL NO.	NINTH: LINE NO.	NINTH: CHANNEL NO.	TENTH: LINE NO.	TENTH: CHANNEL NO.	ELEVENTH: LINE NO.	ELEVENTH: CHANNEL NO.	TWELFTH: LINE NO.	TWELFTH: CHANNEL NO.	PROGRAM IDENTIFICATION
1 2	3 4 5	6	7 8 9 10 11 12 13 14	15 16 17	18 19	20 21 22	23 24	25 26 27	28 29	30 31 32	33 34	35 36 37	38 39	40 41 42	43 44	45 46 47	48 49	50 51 52	53 54	55 56 57	58 59	60 61 62	63 64	65 66 67	68 69	70 71 72	73 74	75 76 77 78 79 80
	0 8	L																										
	0 9	L																										
	1 0	L																										

MEMOREX

RPG II Specifications

Punching Instructions

Graphic

Punch

Date_____ Page_____ of_____

Programmer_____

Program_____

Alternate Collating Sequence and Translation

CODE	MRX GRAPHIC	ENTRY	REPLACES/ REPLACED BY
00000000		00	
00000001		01	
00000010		02	
00000011		03	
00000100		04	
00000101		05	
00000110		06	
00000111		07	
00001000		08	
00001001		09	
00001010		0A	
00001011		0B	
00001100		0C	
00001101		0D	
00001110		0E	
00001111		0F	
00010000		10	
00010001		11	
00010010		12	
00010011		13	
00010100		14	
00010101		15	
00010110		16	
00010111		17	
00011000		18	
00011001		19	
00011010		1A	
00011011		1B	
00011100		1C	
00011101		1D	
00011110		1E	
00011111		1F	
00100000		20	
00100001		21	
00100010		22	
00100011		23	
00100100		24	
00100101		25	
00100110		26	
00100111		27	
00101000		28	
00101001		29	
00101010		2A	
00101011		2B	
00101100		2C	
00101101		2D	
00101110		2E	
00101111		2F	
00110000		30	
00110001		31	
00110010		32	
00110011		33	

CODE	MRX GRAPHIC	ENTRY	REPLACES/ REPLACED BY
00110100		34	
00110101		35	
00110110		36	
00110111		37	
00111000		38	
00111001		39	
00111010		3A	
00111011		3B	
00111100		3C	
00111101		3D	
00111110		3E	
00111111		3F	
01000000	Blank	40	
01000001		41	
01000010		42	
01000011		43	
01000100		44	
01000101		45	
01000110		46	
01000111		47	
01001000		48	
01001001		49	
01001010	¢	4A	
01001011	.	4B	
01001100	<	4C	
01001101	(	4D	
01001110	+	4E	
01001111	|	4F	
01010000	&	50	
01010001		51	
01010010		52	
01010011		53	
01010100		54	
01010101		55	
01010110		56	
01010111		57	
01011000		58	
01011001		59	
01011010	!	5A	
01011011	$	5B	
01011100	*	5C	
01011101	)	5D	
01011110	;	5E	
01011111	¬	5F	
01100000	-	60	
01100001	/	61	
01100010		62	
01100011		63	
01100100		64	
01100101		65	
01100110		66	
01100111		67	

CODE	MRX GRAPHIC	ENTRY	REPLACES/ REPLACED BY
01101000		68	
01101001		69	
01101010		6A	
01101011	,	6B	
01101100	%	6C	
01101101	-	6D	
01101110	>	6E	
01101111	?	6F	
01110000		70	
01110001		71	
01110010		72	
01110011		73	
01110100		74	
01110101		75	
01110110		76	
01110111		77	
01111000		78	
01111001		79	
01111010	:	7A	
01111011	#	7B	
01111100	@	7C	
01111101	'	7D	
01111110	=	7E	
01111111	"	7F	
10000000		80	
10000001		81	
10000010		82	
10000011		83	
10000100		84	
10000101		85	
10000110		86	
10000111		87	
10001000		88	
10001001		89	
10001010		8A	
10001011		8B	
10001100		8C	
10001101		8D	
10001110		8E	
10001111		8F	
10010000		90	
10010001		91	
10010010		92	
10010011		93	
10010100		94	
10010101		95	
10010110		96	
10010111		97	
10011000		98	
10011001		99	
10011010		9A	
10011011		9B	

CODE	MRX GRAPHIC	ENTRY	REPLACES/ REPLACED BY
10011100		9C	
10011101		9D	
10011110		9E	
10011111		9F	
10100000		A0	
10100001		A1	
10100010		A2	
10100011		A3	
10100100		A4	
10100101		A5	
10100110		A6	
10100111		A7	
10101000		A8	
10101001		A9	
10101010		AA	
10101011		AB	
10101100		AC	
10101101		AD	
10101110		AE	
10101111		AF	
10110000		B0	
10110001		B1	
10110010		B2	
10110011		B3	
10110100		B4	
10110101		B5	
10110110		B6	
10110111		B7	
10111000		B8	
10111001		B9	
10111010		BA	
10111011		BB	
10111100		BC	
10111101		BD	
10111110		BE	
10111111		BF	
11000000		C0	
11000001	A	C1	
11000010	B	C2	
11000011	C	C3	
11000100	D	C4	
11000101	E	C5	
11000110	F	C6	
11000111	G	C7	
11001000	H	C8	
11001001	I	C9	
11001010		CA	
11001011		CB	
11001100		CC	
11001101		CD	
11001110		CE	
11001111		CF	

CODE	MRX GRAPHIC	ENTRY	REPLACES/ REPLACED BY
11010000		D0	
11010001	J	D1	
11010010	K	D2	
11010011	L	D3	
11010100	M	D4	
11010101	N	D5	
11010110	O	D6	
11010111	P	D7	
11011000	Q	D8	
11011001	R	D9	
11011010		DA	
11011011		DB	
11011100		DC	
11011101		DD	
11011110		DE	
11011111		DF	
11100000	\	E0	
11100001		E1	
11100010	S	E2	
11100011	T	E3	
11100100	U	E4	
11100101	V	E5	
11100110	W	E6	
11100111	X	E7	
11101000	Y	E8	
11101001	Z	E9	
11101010		EA	
11101011		EB	
11101100		EC	
11101101		ED	
11101110		EE	
11101111		EF	
11110000	0	F0	
11110001	1	F1	
11110010	2	F2	
11110011	3	F3	
11110100	4	F4	
11110101	5	F5	
11110110	6	F6	
11110111	7	F7	
11111000	8	F8	
11111001	9	F9	
11111010		FA	
11111011		FB	
11111100		FC	
11111101		FD	
11111110		FE	
11111111		FF	

2271,000 PRINTED IN U.S.A.

Harry

OPSYS/1 DEVELOPMENT WORKBOOK

TABLE OF CONTENTS

6.5 Phase 3 - Code Generation

6.5.1 Control Program Generator
6.5.2 I/O Generator
6.5.3 File Extension/Alternate Collating Sequence/Table/Line Counter Generator
6.5.4 Input Record Handling Generator
6.5.5 Calculations Generator
6.5.6 Output Record Handling Generator

6.6 Phase 4 - Code Formatter

6.7 Phase 5 - Cross-Reference

7. ENHANCEMENTS

8. APPENDICES

8.1 Diagnostics
8.2 Module/Subroutine Cross-Reference
8.3 Subroutine/Module Cross-Reference
8.4 Glossary

1. FILE CONTROL TABLE

One per file - fixed length 40 bytes each.

The File Control Table has a different format for Input, Output, Update, Record Address and Tables/Arrays.

Following is the portion of the table that applies to all file types.

Bytes		Description	TO	BY	Filled in v=variable phase number (see pg. VI(7)	TO	BY
F$TYP1	0	File Type Switches - More than one bit may be on					
		Bit 0=1 Input file				v	2
		1=1 Output file				v	2
		2=1 Record Address File (Limits)				v	2
		3=1 Record Address File (Record Numbers)				v	2
		4=1 RAFed file - file RAF links to				v	2
		5=1 Chained File				v	2
		6=1 Combined/Update file				v	2
		7=1 MFCU File				v	2
F$TYP2	1	File Type Switches (continued)					
		Bit 0=1 Table/Array File				v	2
		1=1 Console File				v	2
		2=1 Demand File				v	2
		3=1 Random by keys				v	2
		4=1 Random by Record Number				v	2
		5=1 Unused				0	2
		6=1 E Specified (when all E's reach end of file-LR)				v	2
		7=1 End of file reached				0	2
F$TYP3	2	File Type Switches					
		Bit 0=1 Card File				v	2
		1=1 Print File				v	2
		2=1 Special File				v	2
		3=1 Matching Records Specified				v	2
		4=1 Matching Fld in current RCD				0	2
		5=1 Matching Records found				0	2
		6=1 Trailer in current RCD				0	2
		7=1 Lev 1 Flds in current RCD				0	2

Bytes	Description	Filled In	TO	BY
F$TYP4 3	Level 2-9 Present Switches Bit 0=1 Lev 2 FLDS in current RCD 1=1 Lev 3 FLDS in current RCD 2=1 Lev 4 FLDS in current RCD 3=1 Lev 5 FLDS in current RCD 4=1 Lev 6 FLDS in current RCD 5=1 Lev 7 FLDS in current RCD 6=1 Lev 8 FLDS in current RCD 7=1 Lev 9 FLDS in current RCD		0	2
F$BDT 4-5	Buffer Description Table (BDT) address or address of SPECIAL device support routine		v	2
F$LRAD 6-7	Logical Record Address		v	2
F$RLEN 8-9	Record Length (Maximum for Variable Length Records) At object time this is changed to actual record length		v	2
F$TRAN 10-11	File Translation Table Pointer		v	3
F$USER 12-13	User Indicator Address		v	2
F$LINK 14-15	Pointer to next input or prnt FCT		v	2
F$UN1 16-17	Unused		0	2
F$UN2 18	Unused		0	2
F$UN3 19	Unused		0	2

Below the portions referring to Input and Output files combine for update files.

Following is the portion of the table that pertains to Input files. (except RAF and Table/Array files.)

Bytes	Description	Filled in		
F$PRI 20-21	File Priority (0=Primary, 1-19=Secondary)		v	2
F$RID 22-23	Record Identification Table Address		v	2
F$LOOK 24-25	Look ahead field Move Pointer		v	4
F$MACH 26-27	Matching Record Hold Area Pointer		v	4
F$LIV 28-29	Level Control Fields hold area Pointer		v	4

Byte	Description	Filled in	TO	BY
F$CRID	30-31 Current Record Identification Table Entry Pointer		0	4
F$RAFP	38-39 RAF File Pointer (from RAFed File)			
	Following is the portion of the table that pertains to output files.			
F$LINE	20-21 Line Counter Table Pointer		v	3
F$HAD	22-23 Output Record Identification Table chaining pointer - Header and Detail Records		v	6
FST	24-25 Output Record Identification Table chaining pointer - Total Records		v	6
F$UP1	26-31 Left for possible update file			
F$MFCU	32-33 MFCU interpret area address		v	2
F$WORK	34-35 Work area - Last Line Number (Print files only)		0	6
F$SKIP	36 Switches Bit 0=1 Last Line had a skip after		0	6
F$STAT	37 Overflow Status 0=Overflow work not yet done 1=Fetched overflow done Output will change this to 2 instead of doing normal overflow 2=All overflow work done - exec should turn off overflow indicator and this switch set by output routines after overflow processing.		0	6
F$OVER	38-39 Pointer to Overflow Indicator		v	2
	Following is the portion of the table that pertains to Record Address Files (RAF)			
F$ELEN	20-21 RAF Element Length		v	2
F$CLOC	22-23 RAF current location in record		0	2
F$FPTR	24-40 RAF File FCT Pointer		v	2

Following is the portion of the table that pertains to Table/Array files.

	Byte	Description	Filled in	TO	BY
	(26-27	Pointer to Table Control (alternate table pointed to by TACT chain)		v	3
FSTCON	~~26-27 Pointer to TACT (table/array control tabte)~~			~~v~~	~~3~~

Generated Code Summary

There are two major categories of generated code; In line code and interpreted tables. In line code consists of both subroutine calls-and straight line code. Interpreted tables require run time subroutines to process them.

Following is a short summary of code generated by RPG Compiler:

Description	Type	Generated By
1. File Control Table	T	1/0 Generator
2. Buffer Description Table	T	1/0 Generator
3. Line Counter Table	T	FE/Line Counter Generator
4. Table/Array Control Table	T	FE/Line Counter Generator
5. Compile Time Tables/Arrays	T	FE/Line Counter Generator
6. Input Record Identification Table	T	Input Handling Generator
7. Input Field Moves	IL	Input Handling Generator
8. Level/Matching Field Moves	IL	Input Handling Generator
9. Calculations	IL	Calculation Generator
10. Output Moves	IL	Output Generator
11. Constants	T	Calc. & Output Generators
12. Output Record Identification Table	T	Output Generator

Generated Code Groups

Each of the generation overlays generates code in one or more of the compile time Groups. The size of each Group is kent track of and the code formatter assigns the correct address to generated code b, resolving the starting location of each Group.

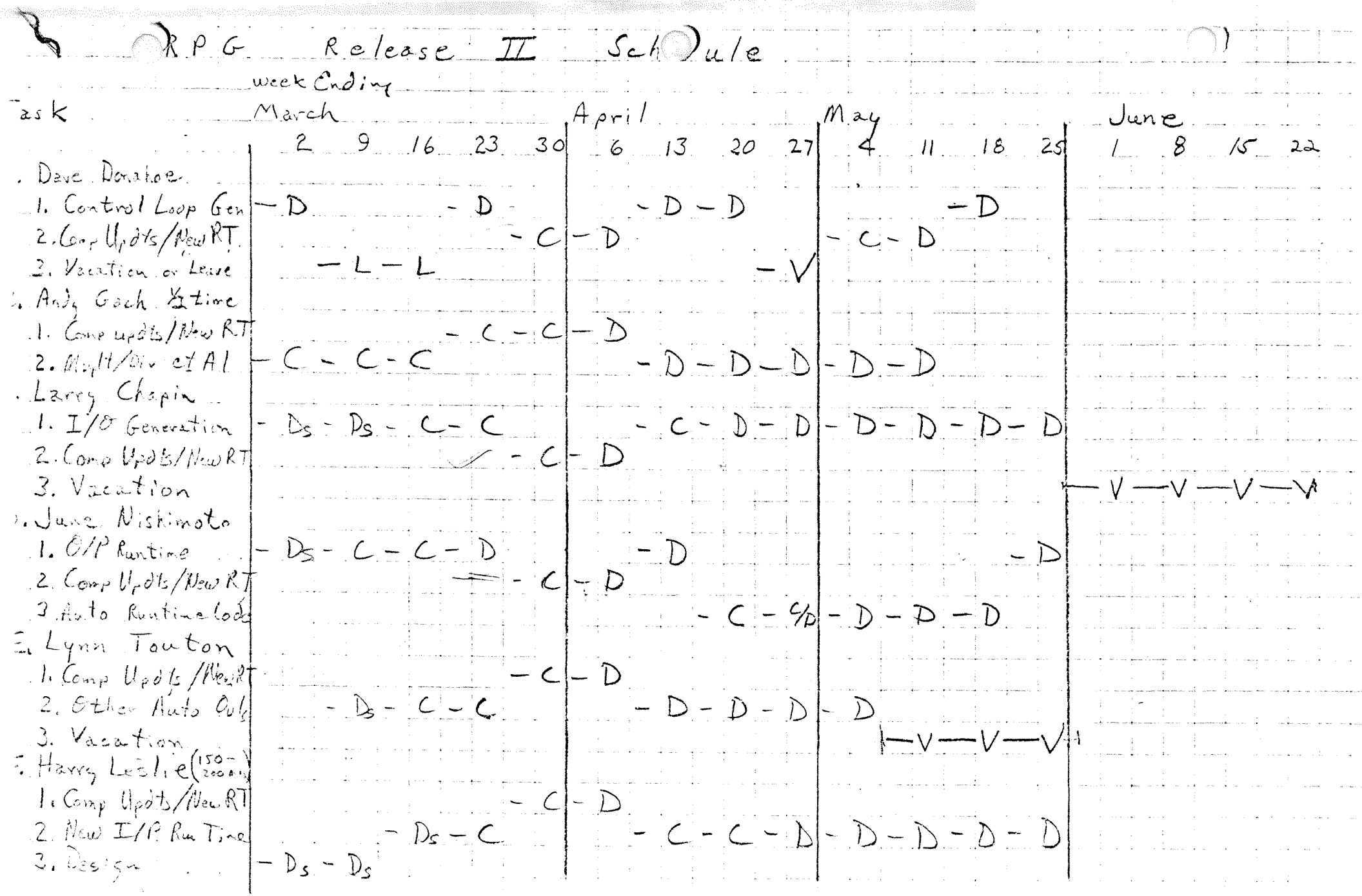
Below is a lis't of each Group by the number that identifies it in the generated code. Included with each Group is a list of all the generators that produce code in that Group. The aenerators are identified as follows:

1. Control Program Generator
2. Input/Output Generator
3. File Extension/Line Counter Generator
4. Input Handling Generator
5. Calculations Generator
6. Output Handling Generator

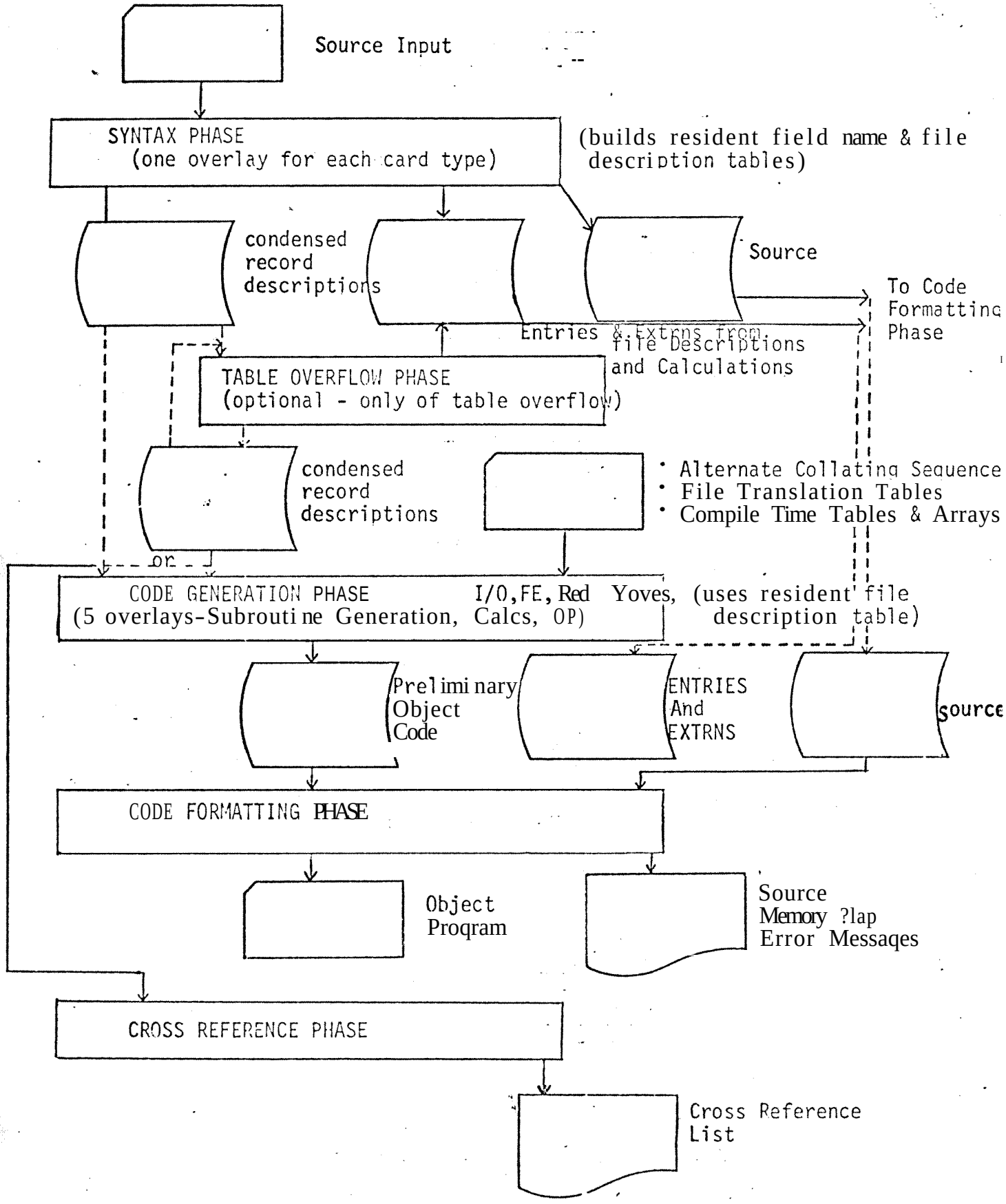
The CSECTS are as follows:

No.	Descrintion	Generated by					
		1	2	3	4	5	6
1	Object Time Communications Region	x	x	x	x	x	x
2	Not Used						
3	I/O/BDT		x				
4	File Control Table		x	x	x		x
5	Table Lookup/Line Counter Table			x			
6	Input Field Moves-Level/Matching Field				x		
7	Input Field Moves-Other				x		
8	Input Record ID Table				x		
9	Calculation					x	
10	Output Moves						x
11	Not Used						
12	Constants, Literals					x	x
13	Output Identification Table - Group 1						x
14	Output Identification Table - Group 2						x
15	Output Identification Table - Group 3						x
16	Output Identification Table - Group 4						x
17	Output Identification Table - Group 5						x
18	Output Identification Table - Group 6						x
19	Data Hold Area			x	x	x	
20	Not Used						
21	Indicators	x	x		x	x	x
22	Buffers and record area		x				

RPG Release II Schedule



BLOCK DIAGRAM OF R. P. G. COMPILERS



Program Overlay	#INPUT Cards in Source MAX	Normal(%)	Size Bytes	Read (Size)	Write (Size)
Phase 1 - Scan					
Compiler Executive			2,000	Open 0,1,2,3	
Header Card Scan #INS=1050/line	1	1	750	0(80)	2(84)
File Description Scan #INS-1230/line	20	2 or 3	4,400	0(80)	1(40,2(84),3(20) very few=0
File Extension Scan #INS-1000/line	60	0 to 2 } one overlay	3,900	0(80)	1(46),2(84)
Line Counter Scan #INS=700/line	8	1 }		0(80)	1(20),2(84)
Input Spec. Scan #INS=1000/line	∞	(33)	4,200	0(80)	1(27),2(84)
Calculation Spec. Scan #INS=1200/line	∞	(33)	4,164	0(80)	1(50),2(84),3(20) very few=0
Output Spec. Scan #INS=1000/line	∞	(33)	3,770	0(80)	1(28),2(84)
		(*Note: Close 3, Reading and writing 1&2 is a 1—1 relationship)			
Phase 2					
Overflow Phase (Optional)			2,000	1(as above)	4(Same as input length)
Phase 3 - Code Generation					
I/O Generator #INS=1500/line	3	3	4,000	1(40 open 5	5(300)in 8 records
Table/Array/Line counter Gen #INS=600/line	3	3	2,000	*0(80),1(46),1(20)	5(50)
Input Handling Generator #INS=800/line	∞	(33)	4.168	Close 2,0 1(27)	5(40),2(84)
CALC.GEN,SUB. PHASE I #INS=700/line	∞	(33)	3,876	open 6 1(50)	6(50)
Calc.Gen.Sub.Phase I #INS=600/line	∞	(33)	5,356	6(50)	5(50)
Output Handling Generator #INS=700/line	∞	(33)	5,630	1(28)	5(60) in 2 records
				close1,6; OPEN 2,3, OBJECT & PRINTER	
Phase 4					
Code Formatter #INS=600/line			4,500	5(50),2(84),3(20) , OP(50),List(132) close 2,3,5	
Phase 5					
Cross Reference (optional) Not used					

FILES: 1,2,3,4,5,6 = work files; 0 = source file; OP = object program; list = source listing

* (Reads in any compile time tables or arrays)

IV. MEMORY ALLOCATION

A. Minimum buffer sizes

File:				
0	(source input)*	- 84	Header Scan	T/A,LCGen.
1	[intermediate output)**	-140	Header Scan	Output Gen
2	(source intermediate)	- 84	Header Scan	T/A,LC Gen.
3	(extrns, entries)	- 24	Header Scan	Output Scan
4	(alternate intermediate)	-140	Overflow Phase	**
5	(generated code)	-140	1/0 Gen	End of Job
6	(inter. gen. code)	-140	Calc Gen I	Output Gen.

* 1 or 4 is closed at end of Overflow Phase cross reference must reopen 1 or 4.

** The size buffer needed for file 0 (source input) is gotten from data management.

Input Source	Buffer Size	Init.
Card Reader	84	Init first 4 bytes as CSD header
Teletype	124	Init first 4 bytes as CSD header
Spooled	Dynamic	None
Library	Dynamic	None

B. Memory Allocation - Partition Size greater than 8K.

1. Available free storage for tables and buffers =

   High core address - 7000 bytes

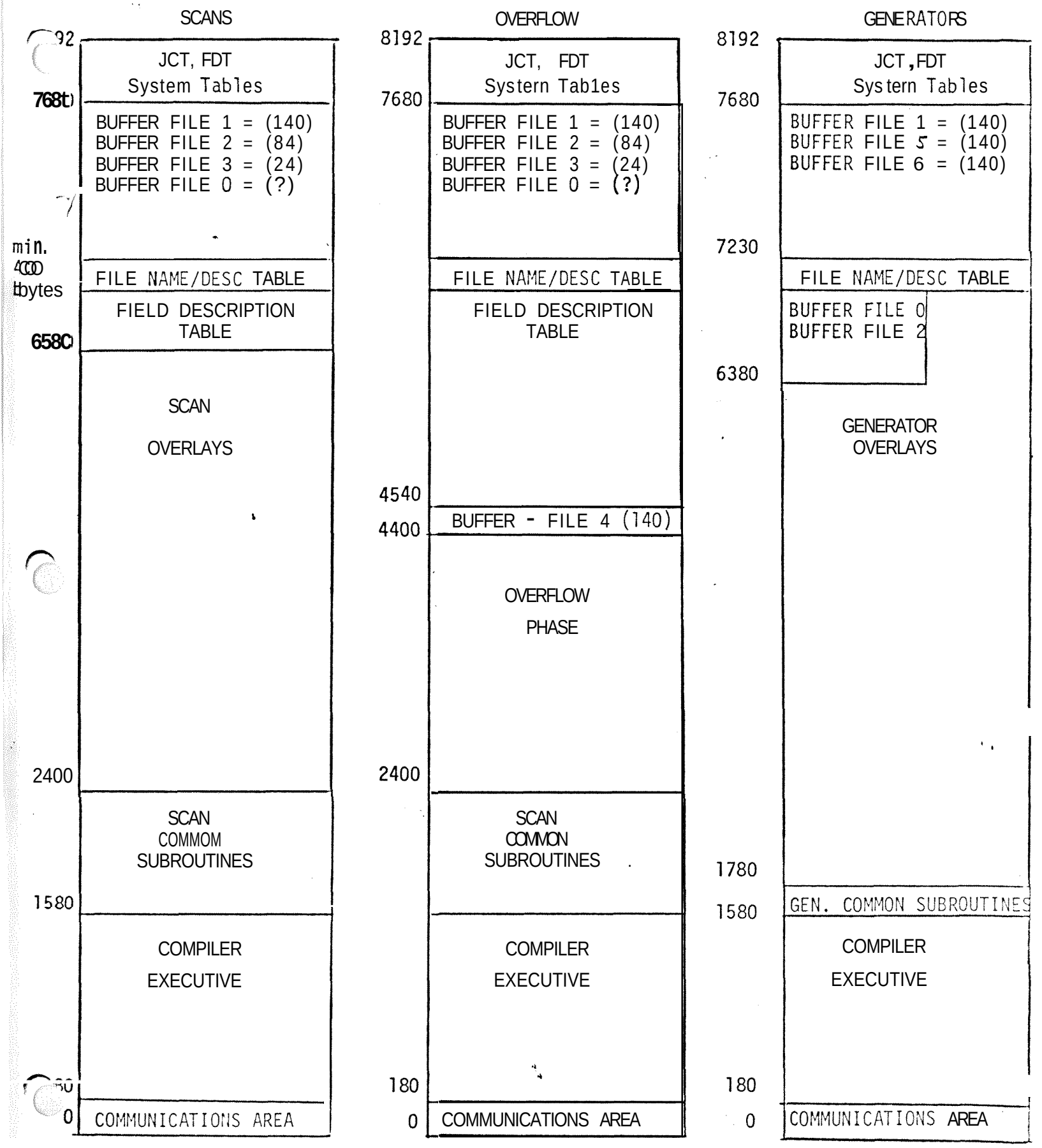
2. Allocate File 0 (source input buffer) same as 8K system

3. Allocate other buffer sizes as:

   File:
   1 = 16% of available free storage
   2 = 16% of available free storage
   3 = 48 bytes ava
   4 = 16% of available free storage
   5 = 16% of available free storage
   6 = 16% of available free storage

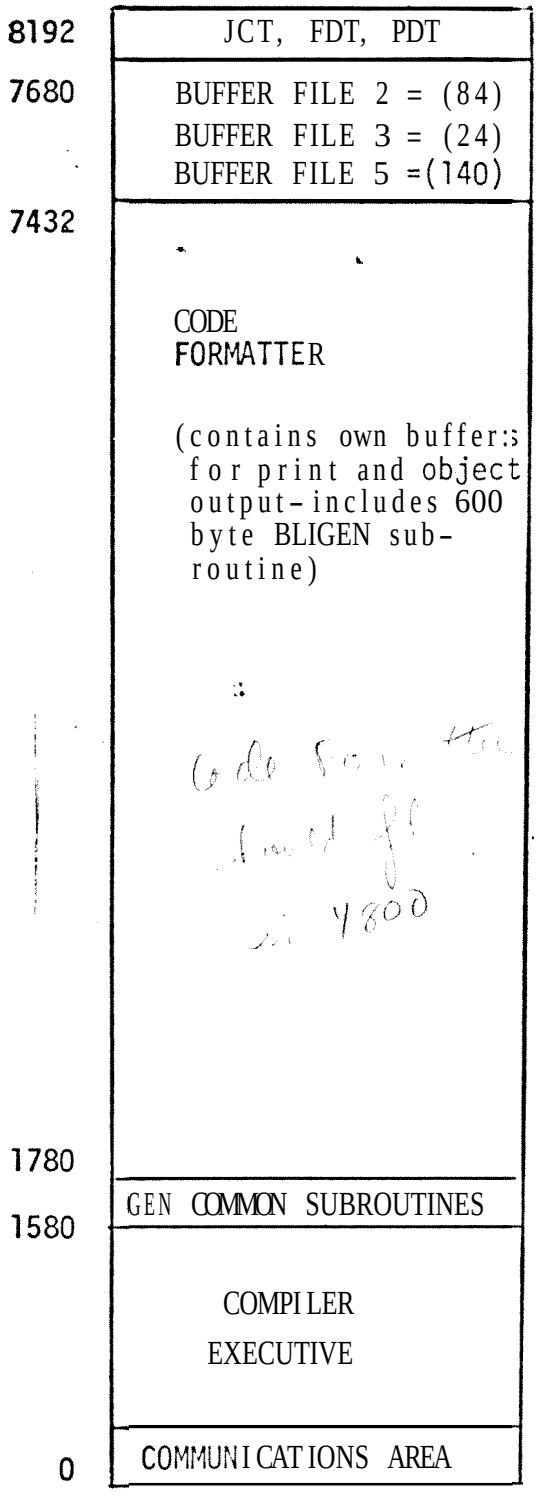
4. Allocate remainder of available space to tables:

C Memory Allocation - Partition Size = 8K

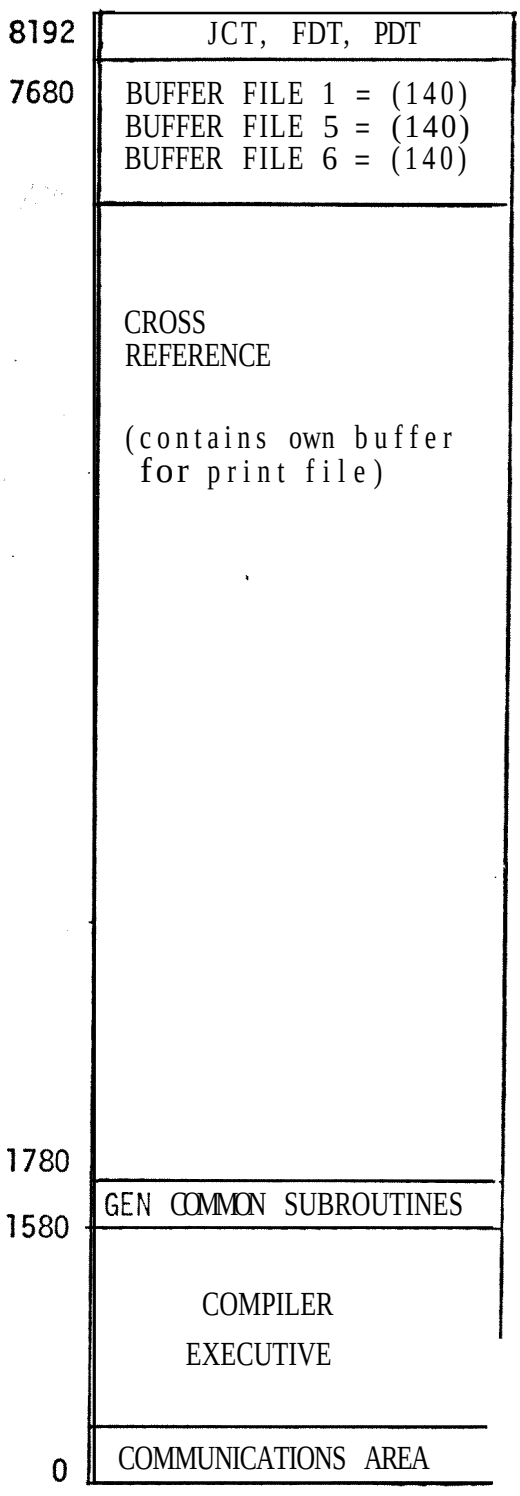


C. (Continued)

CODE FORMATTER

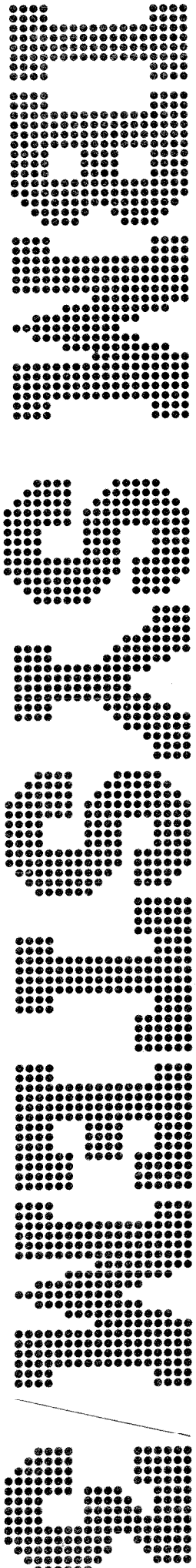


CROSS REFERENCE



IDOS Cobol

Pre-Release



Disk System
RPG II
Reference Manual

Preface

intent of the Manual

This publication is intended as a reference for programmers writing RPG II specifications for the IBM System/3, Disk System. Before using this manual, the reader must understand the concepts and terms described in the following publications:

1. IBM System/3 Disk System Introduction
2. IBM System/3 Card and Disk System RPG II Fundamentals Programmer's Guide

Division of Chapters

This publication has ten chapters. Chapter 1 is the introduction. Chapter 2 contains information common to all RPG coding sheets. Chapters 3-9 describe the seven types of RPG specifications in the order required by the RPG II compiler. Chapter 10 contains supplementary information on subjects referenced in other chapters. The subjects presented in this last chapter are in alphabetical order.

Column Descriptions

Specifications for each coding sheet are described column by column as a programmer would write them. Information in every column description is presented in this order:

1. List of possible entries.
2. General discussion of use of column and considerations for all possible entries.
3. Specific discussion of each entry
4. Charts and examples.

Page and Figure Numbers

Figure numbers and page numbers consist of two numbers separated by a hyphen. The first number identifies the chapter, and the second number identifies the figure or page within the chapter. For example, Figure 4-10 refers to the tenth figure in Chapter 4.

First Edition

Some illustrations in this manual have a code number in the lower comer. This is a publishing control number and is not related to the subject matter.

Changes are continually made to the specifications herein; any such change will be reported in subsequent revisions or Technical Newsletters.

Requests for copies of IBM publications should be made to your IBM representative or to the IBM branch office serving your locality.

A form for reader's comments is provided at the back of this publication. If the form has been removed, comments may be addressed to IBM Corporation, Programming Publications, Department 425, Rochester, Minnesota 55901.

FUNCTION OF RPG II

The RPG II language consists of a symbolic programming language and a compiler program. The RPG II symbolic language is designed as a highly flexible, problem-solving language. It allows programming solutions to a wide variety of data processing problems. The compiler program translates the symbolic language program (source program) into a machine language program (object program). The object program is used by System/3 to process information according to the programmer's specifications.

Basically, then the program you have written undergoes two basic processes:

1. Compilation (source program translated into object program).

2. Execution (object program used to process data).

In the first case, program specifications defined by the programmer are used to produce machine-language instructions. Storage areas are automatically assigned, constants or other reference factors are included, and program routines for checking, for input/output operations, and for other functions are produced.

In the second case, the machine-language instructions are combined with the input data files and both are processed through the system to produce the desired reports and output files.

USING RPG II

The preparation of a report by means of RPG I1 consists of the general operations illustrated in Figure 1-1 and described as follows. (The circled numbers in Figure 1-1 refer to the numbers in the following text.)

1. The programmer evaluates the report requirements to determine the format of the input files and the layout of the finished report. For example, he determines what fields in the input records are to be used, what calculations are to take place, where the data is to be located in the output records and how many and what kind of totals must be accumulated.

2. After the programmer has evaluated the requirements of the report, he provides the RPG II program with information about these requirements.

   a. He describes all files used by the object program (input files, output files, table files, etc.) by making entries on the File Description Specifications sheet.
   b. If the programmer uses record address files, tables, or arrays in his object program, he furnishes information about them through entries on the Extension Specifications sheet.
   c. He describes his input (record layout fields used, etc). This is done by making entries on the Input Specifications sheet.
   d. He states what processing is to be done (add, subtract, multiply, divide, etc.) by means of entries on a Calculation Specifications sheet.
   e. He defines the layout of the desired report (print positions, carriage control, etc). This is accomplished by making entries on the Output-Format Specifications sheet.

3. After the specifications have been written on the appropriate forms, the data on the forms is recorded in punched cards. Each line on the form is punched into one card.

4. These punched cards (called a source deck) are preceded by the RPG II control card. The source deck and the control card are placed into a card-reading device and processed by the RPG II compiler under control of the *Disk System.* At the end of this processing run (referred to as the compilation run), the object program is produced and stored in the working storage area of the disk. This program contains all the machine instructions required to prepare the desired report.

5. The programmer may now have the object program punched into cards for storage or he may proceed directly to processing of the object program.

6. The input files are then read into the system and production of the report begins. This is known as the object run.

MACHINE REQUIREMENTS

The minimum System/3, Disk System machine requirements for use of the RPG II language are:

- 12K bytes of core storage
- 5424 Multi-Function Card Unit
- 5203 Printer
- 5444 Disk Storage Drive
- 5410 Processing Unit

The optional machine devices allowed are:

- 16K, 24K, or 32K bytes of core storage
- Two 5444 Disk Storage Drives
- 5471 Printer Keyboard

RPG Specification Sheets

The RPC specification sheets are used when coding an RPG II program. The format and column headings on each of these sheets guide you in making the appropriate entries. The sheets are designed so that one card is keypunched from each specification line. There are five specification sheets:

1. *Control Card and File Description Sheet.* This sheet contains two types of specifications:

   a. Control card specifications provide information to the RPG II compiler.
   b. File description specifications provide information about all files used in the program.

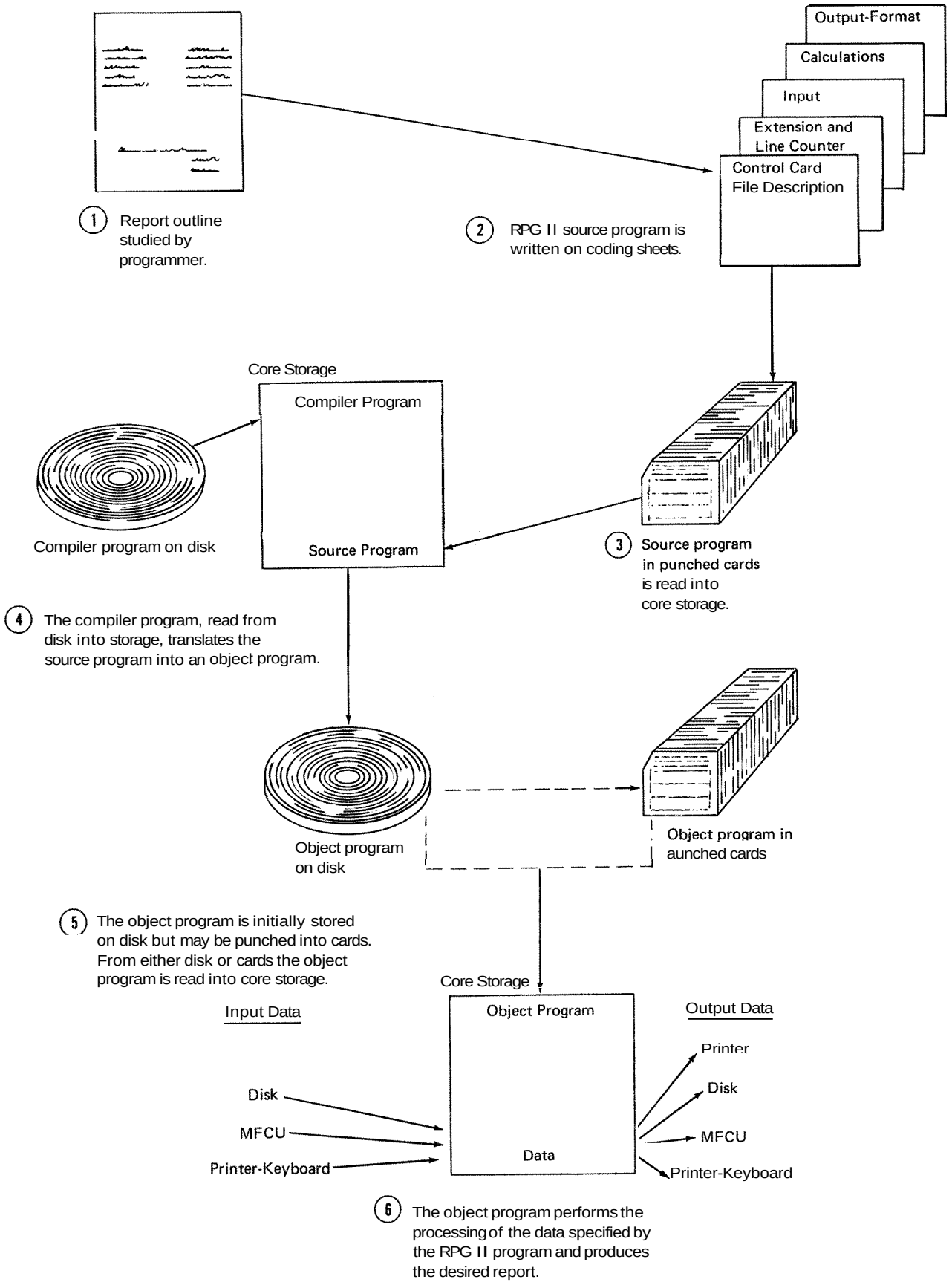


Figure 1-1. Reparation of a Report Using RPG II

2. ***Extension and Line Counter Sheet.*** This sheet contains two types of specifications:

   a. Extension specifications provide information about tables, arrays, and record address files.
   b. Line counter specifications provide information about the number of lines to be printed on the forms that are used.

3. ***Input Sheet.*** This sheet is used to describe the records in an input file.

4. ***Calculation Sheet.*** This sheet is used to describe all operations that are to be performed on the data.

5. ***Output-Format Sheet.*** This sheet is used to specify the arrangement and type of data that will be written or punched on printed reports or cards, or stored on disk.

The information on the specification sheets is recorded in punched cards to form your source program. The arrangement of the cards in a source program deck is shown in Figure 1-2.

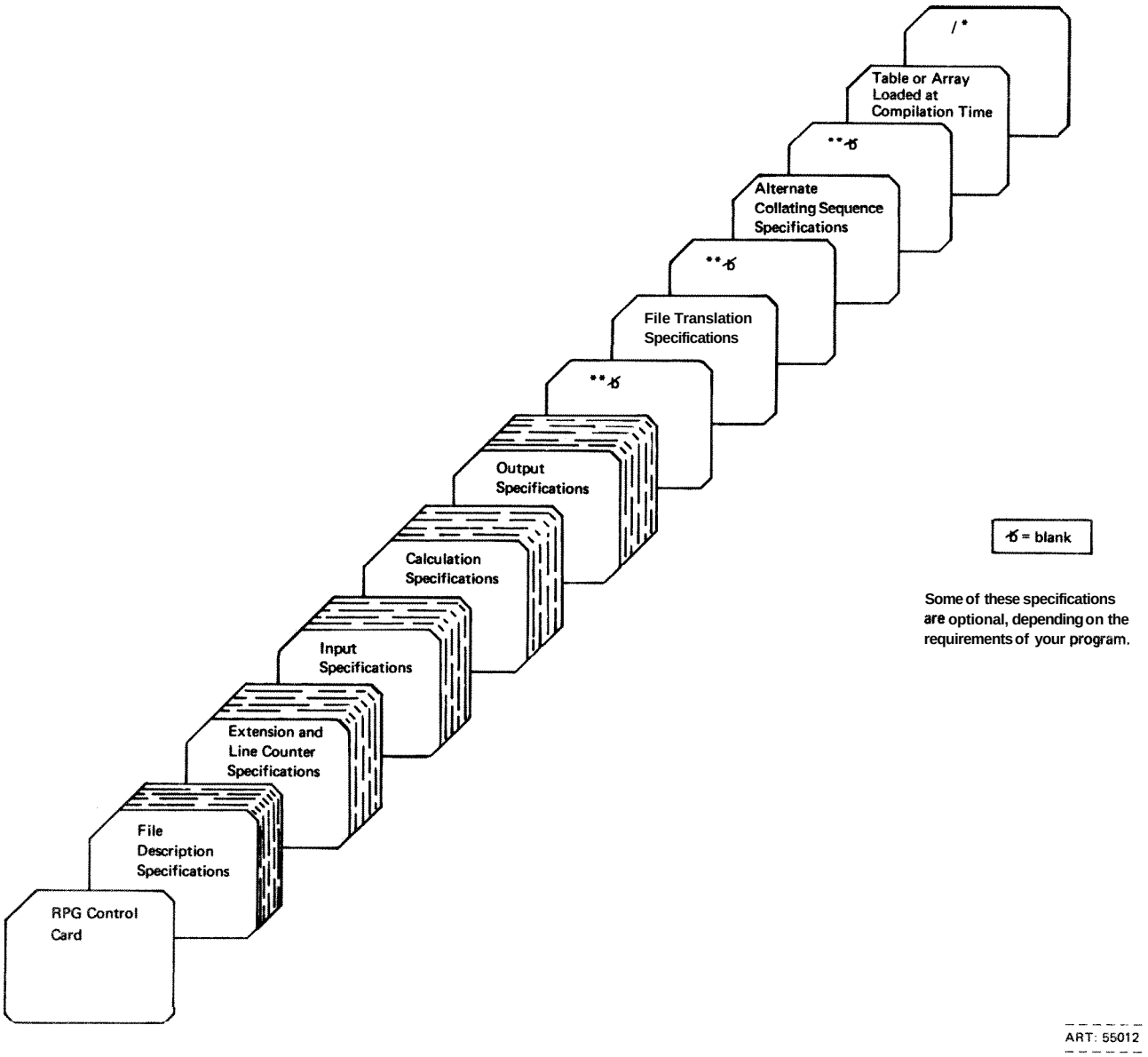


Figure 1-2. **Card Arrangement in the RPG II Source Deck**

This chapter defines the entries which are common to all RPG coding sheets. Each coding sheet contains the following entries:

1. Columns 1-2 (PAGE)
2. Columns 3-5 (LINE)
3. Column 6 (FORM TYPE)
4. Column 7 (COMMENTS)
5. Columns 75-80 (PROGRAM IDENTIFICATION)

COLUMNS 1-2 (PAGE)

Entry	Explanation
01-99	Page number

Columns 1-2 in the upper right corner of each sheet are used to number the specifications sheets for your job. You may use more than one of each type of sheet if you need to, but keep all sheets of the same type together. When all the specifications sheets are filled out, arrange them in the following order:

1. Control Card and File Description
2. Extension and Line Counter
3. Input
4. Calculation
5. Output-Format

Number the sheets in ascending order.

COLUMNS 3-5 (LINE)

Entry	Explanation
Any numbers	Line numbers

Columns 3-5 are used to number the lines on each page. Columns 3-4 are preprinted on each sheet, so in most cases line numbering is already done for you. For instance, the Control Card and File Descriptions sheet contains line numbers for lines 01-07. If you need more than 7 lines on one sheet, enter 08 in columns 3-4 below line 07. Then 09 can be entered if it is required. The blank areas below the preprinted numbers can be used to insert a line between two lines you have completed (see Example).

The control card specification line is always line 01. Any other lines on the sheets can be skipped. The line numbers you use need not be consecutive, but should be in ascending order.

Example

Figure 2-1 shows the insertion of a line between two lines that are already written. To show that a line belongs between line 02 and line 03, a 5 is placed in column 5 (any number 1-9 can be used). Line 025 should be inserted between 02 and 03. All lines inserted between existing lines should be written after the last line with a printed line number.

Note: After the source cards have been punched, cards from insert lines must be placed in proper sequence.

Entry	*Explanation*
E	Extension specifications
L	Line counter specifications
I	Input specifications
C	Calculation specifications
O	Output-Format specifications

COLUMN 6 (FORM TYPE)

Entry	*Explanation*
H	Header card (control card specification)
F	File Description specifications

Column 6 contains a pre-printed letter on all sheets. The letter identifies the type of specifications for each line of coding. The *H* entry in column 6 of the Control Card stands for header card. The header card must always be the first card in the RPG II source program deck (Figure 1-2).

IBM — International Business Machines Corporation — Form X21-9092 Printed in U.S.A.

RPG CONTROL CARD AND FILE DESCRIPTION SPECIFICATIONS

Date _____ Program _____ Programmer _____

Punching Instruction: Graphic / Punch — Page — Program Identification

Control Card Specifications

Line	Form Type	Core Size to Compile	Object Output	Listing Options	Core Size to Execute	Debug	MFCM Stacking Sequence	Sterling: Input-Shillings	Sterling: Input-Pence	Sterling: Output-Shillings	Sterling: Output-Pence	Inverted Print	360/20 2501 Buffer	Number Of Print Positions	Alternate Collating Sequence	Refer to the specific System Reference Library manual for actual entries.
01	H															

File Description Specifications

Line	Form Type	Filename	File Type	File Designation	End of File	Sequence	File Format	Block Length	Record Length	Mode of Processing	Length of Key Field or of Record Address Field	Record Address Type	Type of File Organization or Additional Area	Overflow Indicator	Key Field Starting Location	Extension Code E/L	Device	Symbolic Device	Labels (S, N, or E)	Name of Label Exit	Extent Exit for Dam	Cylinder Index in Core	File Addition/Unordered	Number of Tracks for Cylinder Overflow	Number of Extents	Tape Rewind	File Condition U1-U8
			I/O/U/C/D	P/S/C/R/T/D	E	A/D	F/V			L/R		A/K/I	I/D/T or 1-9										A/U	N/U			
02	F	MASTER	I	P	E	A			96								MFCU1											
03	F	PRINT	O														PRINTER											
04	F																										
05	F																										
06	F																										
07	F																										
025	F	NEWSALES	I	S	A				96								MFCU2										
	F																										

Figure 2-1. Insertion of Lines

COLUMN 7 (COMMENTS)

Entry	*Explanation*
*	Comment line

You often want to write comments that will help you understand or remember what you are doing in a certain section of coding. RPG II allows you to use an entire line for these comments. The comment line is identified by placing an asterisk in column 7. Any characters in the character set may be used in a comment line. A card is punched from this line and the comments appear in the source program listing.

Comments are *not* instructions to the RPG II program. They serve only as a means of documenting your program. A comment line cannot be written in the Control Card specifications line.

Note: To be compatible with other RPG systems, the specification sheets show only 80 card columns that are used for RPG II coding.

COLUMNS 7580 (PROGRAM IDENTIFICATION)

Entry	*Explanation*
Any characters	Program identification.
Blank	RPGOBJ is assumed.

Control Cards

Columns 75-80 of the control card are used to name your object program. This name is used in a program directory which contains the location of your program on disk. You may use any combination of characters in these columns. The compiler places the first four characters (columns 75-78) into positions 89-92 of each record in your object program. If columns 75-80 are left blank, the compiler assumes the entry is RPGOBJ. (The compiler uses columns 93-96 of each object program record for consecutive numbering of the records.)

All Other Source Cards

Columns 75-80 on all source program cards, except the control card, may contain any characters. These columns may use the program name in the control card, or the column may contain any other characters to identify a certain portion of the program. These entries are ignored by the compiler, but will appear in the source program listing.

One control card is required for every program. It provides special information about your program and describes your system to the RPG II compiler. Without this information your source program cannot be translated into an RPG II object program. To help you code the control card, one specification line is provided on the Control Card and File Description sheet (Figure 3-1).

COLUMNS 1-2 (PAGE)

See Chapter 2.

COLUMNS 3-5 (LINE)

See Chapter 2.

COLUMN 6 (FORM TYPE)

An *H* must appear in column *6*. A control card with an *H* in column *6* must be entered for every program even if all the other control card columns are left blank.

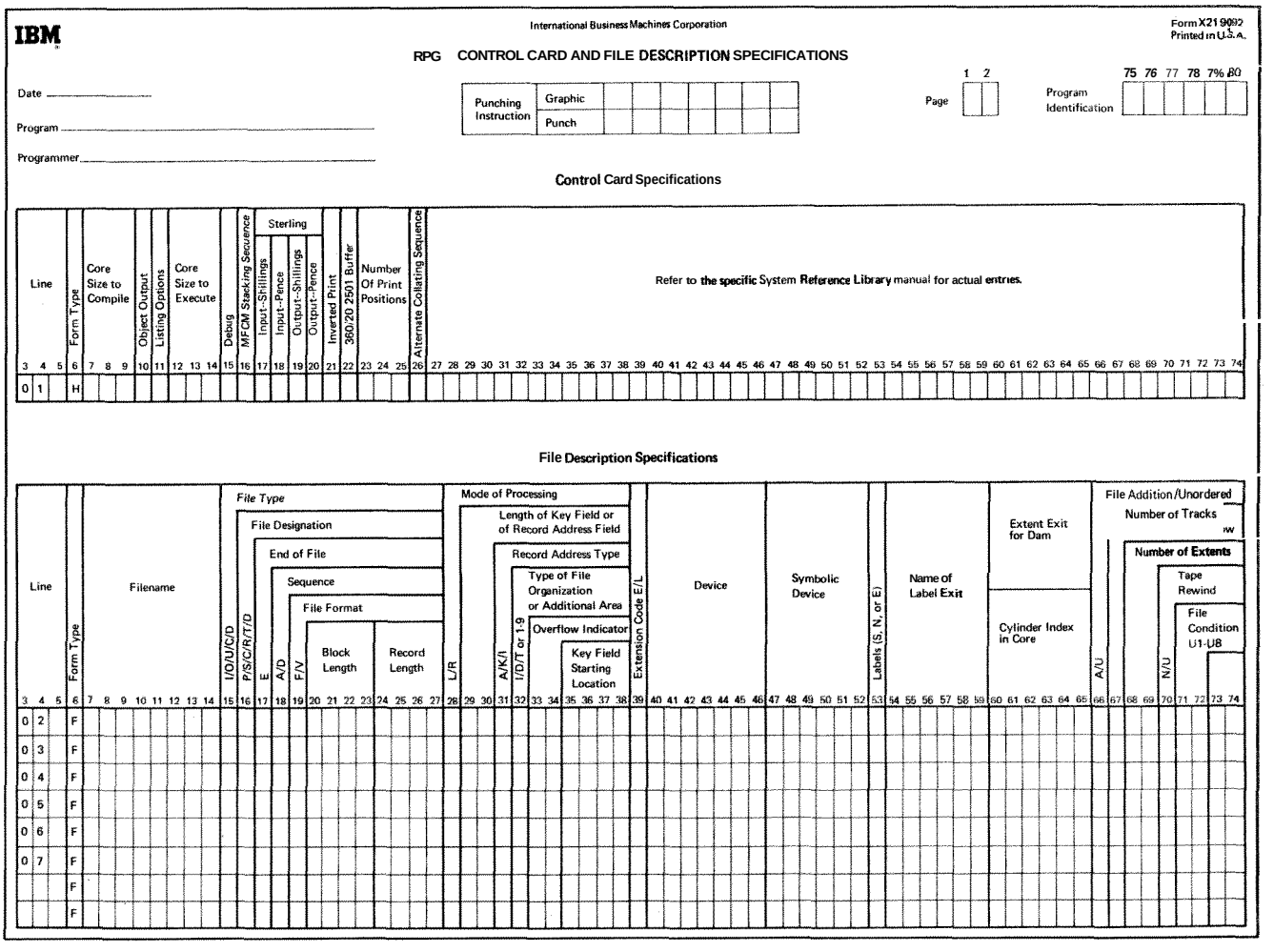


Figure 3-1. Control Card and File Description Sheet

COLUMNS 7-9 (CORE SIZE TO COMPILE)

Columns 7-9 are not used. Leave them blank. Any entry in these columns is ignored by the compiler. The program is compiled in the available core storage.

COLUMN 10 (OBJECT OUTPUT)

Entry	*Explanation*
Blank	Object program is written temporarily in the object library.
C	Object program is written permanently in the object library.
P	Object program is punched into cards.

Column 10 is used to indicate the output you want as a result of compiling the source program. The object program is always written in the same object library in which the compiler resides.

You will usually want the object program written temporarily in the object library until you have corrected any severe errors in your program. When a program is written temporarily in the object library, it will be overlaid by the next program written in the object library.

COLUMN 11 (LISTING OPTIONS)

Entry	*Explanation*
Blank	1. The object program is produced (if no severe errors are found). 2. A program listing is printed. 3. A core map is printed.
D	1. The object program is not produced. (The D entry overrides any entry in column 10 of the Control Card Specifications.) 2. The program listing is printed. 3. A core map is printed.

Column 11 provides for listing options at the time your source program is compiled. Compilation will cease after the program listing is complete if any severe errors are found.

The blank entry is the usual case, producing an object program (if no severe errors are found), an object program listing, and a core map. The program listing consists of the source program and error messages. The core map lists relative addresses of fields, constants, I/O areas, etc.

The D entry means that no object program will be produced; however, a program listing with error messages and a core map are printed. This entry can be used if you think the program has many initial errors. In such a case, it is not advisable to produce an object program until these errors are reduced in number. The D entry overrides any entry made in column 10 of the Control Card Specifications line.

COLUMNS 12-14 (CORE SIZE TO EXECUTE)

Entry	*Explanation*
Blank	The core storage available for object program execution is the same as that used to compile the program.
001-029	The core storage available for program execution (if different from core storage available for object program generation).

Columns 12-14 define the core storage available for program execution. The entry must end in column 14. The entry is some multiple of 1K bytes of storage (K=1,024).

This entry may be different from the core storage available for object program generation for two reasons: I) Your program may be executed on a different system from the one that compiled your program, or 2) You are using the Dual Program Feature.

If the system used for program execution is different from that used for compilation, subtract the amount of core storage occupied by the supervisor from the total core storage of the system used for execution.

If you are using the Dual Program Feature, subtract the amount of core storage occupied by the second object program and the supervisor from the total core storage of the system used for program execution.

COLUMN 15 (DEBUG)

Entry	*Explanation*
Blank	DEBUG operation is not performed.
1	DEBUG operation is performed.

Use column 15 to indicate whether or not the DEBUG operation is performed. In order to perform a DEBUG operation:

1. A *1* must appear in column 15 when the source program is compiled,
2. The DEBUG operation code must appear in calculation specifications.

See *Operation Codes, Debug Operation* in Chapter 10 for more information.

COLUMN 16

Column 16 is not used. Leave it blank.

COLUMNS 17-20 (STERLING)

Columns 17-20 are used to describe the format of the sterling fields used in sterling currency. If you are not using sterling, these columns *must* be left blank. See *Sterling* in Chapter 10 for more information.

COLUMN 17 (INPUT–SHILLINGS)

Entry	*Explanation*
Blank	Sterling currency is not being used.
1	Input shilling field is in IBM format.
2	Input shilling field is in BSI (British Standards institute) format.

COLUMN 18 (INPUT–PENCE)

Entry	*Explanation*
Blank	Sterling currency is not being used.
1	Input pence field is in IBM format.
2	Input pence field is in BSI format.

COLUMN 19 (OUTPUT–SHILLINGS)

Entry	*Explanation*
Blank	Sterling currency is not being used.
0	Output shilling field is to be printed only.
1	Output shilling field is to be punched in IBM format.
2	Output shilling field is to be punched in BSI format.

COLUMN 20 (OUTPUT–PENCE)

Entry	*Explanation*
Blank	Sterling currency is not being used.
0	Output pence field is to be printed only.
1	Output pence field is to be punched in IBM format.
2	Output pence field is to be punched in BSI format.

The same fields may be both punched and printed. Although they are always punched in the selected format (IBM or BSI), the printed output is not affected by the selected format. Printed fields always have two positions in both the pence and shilling fields. See *Sterling* in Chapter *10* for more information.

COLUMN 21 (INVERTED PRINT)

Entry	*Explanation*
Blank	Domestic format.
I	World Trade format.
J	World Trade format (leading zero remains for zero balances).
D	United Kingdom format.

Use column 21 to describe the format and punctuation used in numeric and UDATE fields. The blank entry specifies the domestic format of month/day/year for UDATE fields (10/15/69), and a decimal point for numeric fields (183.55).

The I entry specifies the World Trade format of day.month.year for UDATE fields (15.10.69), and a decimal comma for numeric fields (183,55).

The J entry specifies the same World Trade format as the I entry with one exception. When the J entry is used, zero balances are always written or punched with one zero to the left of the decimal comma (such as 0,00). Also this leading zero appears for the J entry when there is some value in the field, but there is no value to the left of the decimal comma (such as 0,04 or 0,10). The J entry overrides any edit codes used for zero suppression; that is, the leading zero that appears for the J entry cannot be removed by an edit code.

The D entry specifies the United Kingdom format of day/month/year for UDATE fields (15/10/69), and a decimal point for numeric fields (183.55).

COLUMNS 22-25

Columns 22-25 are not used. Leave them blank.

COLUMN 26 (ALTERNATE COLLATING SEQUENCE)

Entry	***Explanation***
Blank	Normal collating sequence is used.
S	Alternate collating sequence is used.

Use column 26 only if you are altering the normal collating sequence for this job. See ***Alternate Collating Sequence*** in Chapter 10 for more information.

COLUMNS 27-36

Columns 27-36 are not used. Leave them blank.

COLUMN 37 (RPG INQUIRY SUPPORT)

Entry	*Explanation*
Blank	This program cannot be interrupted (does not recognize an inquiry request).
B	This program can be interrupted (does recognize an inquiry request).
I	This program is an inquiry program that can only be executed when an inquiry request is made.

System/3 Disk System allows certain programs to be interrupted while they are being processed. A request for interruption is called an inquiry request (made by depression of the inquiry key on the printer-keyboard). Programs are usually interrupted to permit another program to run and then control is given back to the first program.

A blank entry in column 37 indicates that the program cannot be interrupted (does not recognize an inquiry request).

A "B" entry indicates that the program can be interrupted (will recognize an inquiry request).

An I entry indicates that the program is an inquiry program that is executed only when the inquiry request is made. Usually this type of program is read in only when a B type program is interrupted. In this case the I type program will not recognize an inquiry request. However, if an I type program is loaded in the normal manner (not because of a program interrupt) it can only be executed when an inquiry request is made. While this program is running, it will not recognize an inquiry request.

The RPG inquiry request is outlined in these steps:

1. Only a B type program will recognize an inquiry request.
2. When the program recognizes on inquiry request, a "Roll-Out" routine moves the interrupted program from main storage to disk.
3. The program for which the interrupt was requested is processed. The interrupting program may be any type (blank, B, or I). This interrupting program cannot be interrupted no matter what type it may be.
4. After the interrupting program is executed, the interrupted program moves back into main storage using a "Roll-In" routine. The interrupted program begins execution at the point of interruption and terminates in a normal manner.

Note: In the dual program mode the same specifications apply except that only level I programs can be interrupted and "Rolled-Out".

COLUMNS 38-40

Columns 38-40 are not used. Leave them blank.

COLUMN 41 (FORMS POSITIONING)

Entry	*Explanation*
Blank	First 1P line is printed only once.
I	First 1P line can be printed repeatedly.

When forms are first inserted in the printer, they may not always be in perfect alignment. Sometimes it requires the printing of several lines to determine the correct positioning of the form. Since you do not want to print several lines of your report before you get the forms positioned correctly, you have the option of repeatedly printing the first line conditioned by the first page (1P) indicator. Each time the 1P line is printed, the program halts so you may reposition the forms if needed.

COLUMN 42

Column 42 is not used. Leave it blank.

COLUMN 43 (FILE TRANSLATION TABLES)

Entry	*Explanation*
Blank	No file translation is needed.
F	Input, output, update, or combined files are to be translated.

Use column 43 only when information contained in an input, output, combined, or update file is in a form which is not usable by your program. When file translation is specified for an update or combined file, both the input and output portion of the file is translated.

An F in column 43 indicates either or both of the following: The character code used in the input data must be translated into a form that can be used by your program, or the output data must be in a character code different from that used by your program.

The specifications for forming a file translation table are discussed under ***File Translation*** in Chapter 10.

COLUMN 44 (LEADING ZERO SUPPRESSION)

Entry	*Explanation*
Blank	Leading zeros are removed.
1	Leading zeros are used.

This column applies only to output on the MFCU. If the column is left blank, all numeric output fields on the MFCU will have leading zeros removed. Enter a *1* in column 44 when you wish to have leading zeros on fields punched or printed by the MFCU.

If an edit word or edit code is defined for records to be printed or punched on the MFCU, the edit word or code will override column 44. See ***Editing*** in Chapter 10 for the other ways in which to suppress leading zeros.

COLUMN 45 (UNPRINTABLE CWARACTER OPTION)

Entry	*Explanation*
Blank	Program halts if an unprintable character is to be printed.
1	No program halt for such unprintable characters.

Column 45 is used to bypass machine halts for unprintable characters. This column applies to the printer and the printer keyboard, All characters are known to the system by a numeric code. If a numeric code is formed which is not known to your system (not in your character set) and that character is to be printed. the machine will halt without attempting to print.

If you wish to bypass this halt, enter 1 in column 45. An unprintable character will be printed as a blank and no halt will occur. Note, however, that this option could make some types of output data meaningless.

COLUMNS 46-74

Columns 46-74 are not used. Leave them blank.

COLUMNS 75-80 (PROGRAM IDENTIFICATION)

See Chapter 2.

File description specifications are required for every file used by a program. Write these specifications on the Control Card and File Description sheet (Figure 4-1). Only one line is needed to describe a file.

At the end of this chapter is a series of charts showing all possible files that can be defined on the File Description sheet (see Figures 4-14 through 4-20). The charts are arranged by device, showing the basic entries for all possible disk, card, console, and printer files.

COLUMNS 1-2 (PAGE)

See Chapter 2.

COLUMNS 3-5 (LINE)

See Chapter 2.

IBM

International Business Machines Corporation

Form X21 9092
Printed in U S A

RPG CONTROL CARD AND FILE DESCRIPTION SPECIFICATIONS

Date _____

Program _____

Programmer _____

Punching Instruction: Graphic / Punch

Page 1 2

Program Identification 75 76 77 78 79 80

Control Card Specifications

Line 3 4 5	Form Type 6	Core Size to Compile 7 8 9	Object Output 10	Listing Options 11	Core Size to Execute 12 13 14	Debug 15	MFCM Stacking Sequence 16	Sterling: Input-Shillings 17	Sterling: Input-Pence 18	Sterling: Output-Shillings 19	Sterling: Output-Pence 20	Inverted Print 21	360/20 2501 Buffer 22	Number Of Print Positions 23 24 25	Alternate Collating Sequence 26	Refer to the specific System Reference Library manual for actual entries. 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74
0 1	H															

File Description Specifications

Line 3 4 5	Form Type 6	Filename 7 8 9 10 11 12 13 14	I/O/U/C/D 15	P/S/C/R/T/D 16	E 17	A/D 18	F/V 19	Block Length 20 21 22 23	Record Length 24 25 26 27	L/R 28	A/K/I 29 30 31	I/D/T or 1-9 32	Type of File Organization or Additional Area 33 34	Overflow Indicator 35 36 37 38	Key Field Starting Location	Extension Code E/L 39	Device 40 41 42 43 44 45 46	Symbolic Device 47 48 49 50 51 52	Labels (S, N, or E) 53	Name of Label Exit 54 55 56 57 58 59	Extent Exit for Dam 60 61 62 63 64 65	Cylinder Index in Cote	A/U 66	Number of Tracks for Cylinder Overflow 67 68 69	Number of Extents 70	N/U 71	Tape Rewind 72	File Condition U1-U8 73 74
0 2	F																											
0 3	F																											
0 4	F																											
0 5	F																											
0 6	F																											
0 7	F																											
	F																											
	F																											

Figure 4-1. Control Card and File Description Sheet

COLUMN 6 (FORM TYPE)

An *F* must appear in column 6.

COLUMNS 7-14 (FILENAME)

Use columns 7-14 to assign a unique name to every file used in your program. Every file must be named, with two exceptions:

1. Compile time tables or arrays are not described, and must not be named on the File Description sheet. Compile time tables or arrays are described on the Extension sheet.

2. Execution time tables and arrays are described on the File Description sheet. However, if more than one execution time table or array is assigned to the MFCU, only one of the tables or arrays need be named and described on the File Description sheet. Note, however, that all execution time tables and arrays must have a unique name and must be further described on the Extension sheet.

The filename can be from 1-8 characters long, and must begin in column 7. The first character must be an alphabetic character. The remaining characters can be any combination of alphabetic and numeric characters (special characters are not allowed). Blanks may not appear between characters in the filename.

COLUMN 15 (FILE TYPE)

Entry	*Explanation*
I	Input file
O	Output file
U	Update file
C	Combined file
D	Display file

Use column 15 to identify the way in which your program uses the file.

Input files are records that a program uses as a source of data. When input files are described in a program it indicates that records are to be read from the file. All Input files must be further described on the Input sheet with two exceptions: Table files and Record Address files must be further described in the Extension sheet.

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

Update files are disk files from which a program reads a record, updates fields in the record, and writes the record back in the location from which it was read. Update files must be further described on both the Input and Output-Format sheets,

A *Combined file* is a file which is both an input file and an output file. A combined file must be assigned to the MFCU or the console. A program reads records from a combined file and includes output data on the records in the file. The result is one file that contains both input and output data. Combined files must be further described on both the Input and Output-Format sheets.

A ***Display** file* is a collection of information from fields used by a program. The DSPLY operation code must be used on the Calculation sheet in order to print a field or record directly from storage and/or key data into a field or record in storage. Display files need only be described on the File Description sheet. The device associated with a display file must be a console. See *Operation Codes, Display,* in Chapter *10* for more information.

COLUMN 16 (FILE DESIGNATION)

Entry	*Explanation*
P	Primary file
S	Secondary file
C	Chained file
R	Record address file
T	Table fife (execution time tables or arrays)
D	Demand file

Use column 16 to further identify the use of input, update, and combined files. Leave the column blank for display files, and all output files except chained output files (direct load).

Primary File

A primary file is the main file from which a program reads records. In multifile processing the primary file is used to control the order in which records are selected for processing. It can be an input, update, or combined file. In programs that read records from only one file, that file is the primary file. Every program must have one, and only one, primary file. If more than one primary file is specified, all but the first primary file named are considered secondary files.

If no P entry is made in the file description specifications, the first Secondary file defined will be taken as the Primary file. When no Secondary files are present in the program, the first Demand file will be used as the Primary file. If none of the above are present, a warning is issued and the program cannot be executed.

Secondary Files

Secondary files apply to programs that do multifile processing. All of the files involved in multifile processing except the primary file are *secondary files.* A secondary file can be an input, update, or combined file. Secondary files are processed in the order in which they are written in the file description specifications.

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

See *Multifile Processing* in Chapter *10* for more information on primary and secondary files.

Chained Files

A chained file is a disk file that uses the CHAIN operation code to:

1. read records randomly
2. load a direct file

A chained file can be an Input, Output, or Update file. See *Column 28 (Mode of Processing), Random,* in this chapter, and *Operation Codes, CHAIN,* in Chapter 10.

Record Address Files

A record address file is an input file that indicates (1) which records are to be read from a disk file and (2) the order in which the records are to be read from the disk file. You cannot use more than one record address file in a program. All record address files must be further defined in extension specifications.

Record address files contain record-key limits or disk addresses. Record address files that contain record-key limits can be disk files, card files, or can be entered by the printer-keyboard.

Record address files that contain disk addresses can only be disk files. Those files that contain limits are used with indexed files only. See *Column 28 (Mode of Processing), Sequential Within Limits* in this Chapter for more information.

Record address files on disk that contain disk addresses are called *ADDROUT (address output) files.* They are produced by the Disk Sort program and can be used with any type of disk file. See *Column 28 (Mode of Processing), By ADDROUT File* in this chapter for more information.

Table Files

A *table file* is an input file that contains table entries. The entries can be read into the program during the compilation or execution of the program. Only execution time tables are described on the File Description sheet. However, both execution and compile time tables must be described in extension specifications.

Entries read during compilation become a permanent part of the program. Both compile and execution time tables may be changed at execution time. Compile time tables, however, can be permanently altered only by recompiling the program. Execution time tables can be permanently altered each time the program is executed.

Table files are not involved in record selection and processing. They are only a means of supplying entries for tables used by the program. When table files are read during the execution of the program, the program reads all of the entries from the table files before it begins record processing. All table files must be further defined in extension specifications.

Demand Files

Demand files can be Input, Update, or Combined files. The FORCE operation code must be used in calculation specifications in order to read from a Demand file. See *Operation* Codes, *FORCE,* in Chapter 10.

COLUMN 17 (END OF FILE)

Entry	*Explanation*
E	All records from the file must be processed before the program can end.
Blank	1. The program can end whether or not all of the records from the file have been processed. 2. If column 17 is blank for all of the files, all records from every file must be processed before the program can end.

Column 17 applies to programs that perform multifile processing. Use it to indicate whether or not the program can end before all of the records from the file are processed. It applies only to input, update, and combined files that are used as primary and secondary files.

A program that performs multifile processing could reach the end of one file before reaching the end of the others. It therefore needs some indication of whether it is to continue reading records from the other files or end the program. An entry in column 17 in the descriptions of the files provides that indication.

If the records from all of the files must be processed, column 17 must be blank for all files, or contain *E*'s for all files.

COLUMN 18 (SEQUENCE)

Entry	*Explanation*
A	Sequence checking is to be done. Records in the file are in ascending order.
D	Sequence checking is to be done. Records in the file are in descending order.
Blank	No sequence checking Is to be done.

Column 18 applies to update files, combined files, and all input files except table files. Leave column 18 blank for output, display, record address, or table files, and for all random processing (R in column 28). Use it to indicate whether or not the program is to check the sequence of the records. Use columns 61-62 on the Input sheet to identify the record fields containing the sequence information.

Sequence checking is *required* when match fields are used in the records from the file. When a record from a matching input file is found to be out of sequence, the program halts, and the operator will have three options:

1. bypass the record out of sequence and read the next record from the same file, or
2. bypass the record out of sequence, turn on the LR indicator and perform all end-of-job and final total procedures, or
3. cancel the entire program.

If column 18 is blank and matching fields are used, the compiler assumes that the entry is *A* and prints a warning message. For more information, see *Columns* 61-62 *(Matching Fields)* in Chapter 7.

COLUMN *19 (FILE* FORMAT)

Entry	*Explanation*
F	Fixed-length records

Column 19 must contain an F entry. This entry indicates that all of the records in the file are of the same length. **A** blank entry is assumed as *F.*

COLUMNS 20-23 (BLOCK LENGTH)

Entry	***Explanation***
A number that is a multiple of record length	Records are blocked. The number is some multiple of the record length.
A number that equals record length	Records are unblocked. The block length equals the record length.

Use columns 20-23 to indicate the block length. An entry must be made in these columns for every disk file. For the MFCU, printer, or console enter a value equal to record length. The block length is a multiple of record length, indicating the number of records you want the program to read or write on disk at one time. If the record length is 120, you might have a block length of 480, indicating that you want 4 records read or written on disk at one time. The program runs more quickly when records are blocked. Blocking does not affect the order of records stored on disk. The entry must end in column 23, and leading zeros may be omitted.

The block length must be a multiple of the record length. If the record length is 120, the block length must be 240 or 360 or 480 and so on. However, the system must read and write data from the disk in multiples of 256 characters. If your block length is 80, the system must actually read in 256 characters. If your block length is 260, the system must read in 512 characters (2 x 256).

Therefore, the most efficient blocking you can do is to have your block length as close as possible to a multiple of 256. For a record length of 120,240 is a more efficient block length than 120, since 240 is closer to 256. Also, for the same record length, 480 is a more efficient block length than 360 because 480 is closer to a multiple of 256 (512). If you specified 360 as a block length, 512 characters must be read in by the system, so you might as well include another record, making your block length 480.

COLUMNS 24-27 (RECORD LENGTH)

Entry	*Explanation*
1-4096	The number of characters in each record, limited by the device used.
Blank	The maximum record length of the device is assumed.

Use columns 24-27 to indicate the length of the records in the file. Any entry must be made in these columns for a disk file.

All of the records in one file must be the same length. (For update files, the length of a record after the record is updated must be the same as it was before the record was updated.) The maximum length allowed depends upon the device assigned to the file.

Device	*Maximum Length*
MFCU	96
Printer	96,120, or 132 (depending on the size of the print positions)
Disk	4096
Console	125

The record length may be shorter than the maximum length for the device, but not longer.

The entry you place in these columns must end in column 27. Leading zeros can be omitted. If columns 24-27 are left blank for a disk file, a warning is issued and the program will not be executed. If these columns are left blank for any other devices, the maximum record length of the device is assumed.

COLUMN 28 (MODE OF PROCESSING)

Entry	*Explanation*
L	Sequential Within Limits
R	1. Random By Relative Record Number
	2. Random By Key
	3. By ADDROUT File
	4. Direct file load (random load)
Blank	1. Sequential By Key
	2. Consecutive

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

For disk files specified as primary, secondary, or chained files, the possible methods depend upon the organizations of the files (Figure 4-2). For the other types of files, consecutive processing is the only possible method.

Column 31 is used to further identify the method for the program. See *Column 31 (Record Address Type)* in this chapter.

To indicate a direct file load (random load) enter an *R* in column 28.

Consecutive

The consecutive method applies only to sequential and direct files. During consecutive processing records are read in the order in which they physically appear in the file. The contents of spaces left for missing records in direct files are read as though the records were there. (When a direct file is loaded, such spaces are filled with blanks.)

PRIMARY AND SECONDARY FILES

Organization	Possible Methods
Sequential	1. Consecutively 2. By ADDROUT file
Direct	1. Consecutively 2. By ADDROUT fife
Indexed	1. By ADDROUT file 2. Sequentially by key 3. Sequentially within limits.

CHAINED FILES

Organization	Possible Methods
Sequential	Randomly by relative record number
Direct	Randomly by relative record number
Indexed	Randomly by key

Figure 4-2. Possible Record-Retrieval Methods for Disk Files

The program reads records from the file until either the end of that file is reached or the program ends due to the end of file condition of another file. See *Column 17, End of File* in this chapter for more information about the second condition.

By ADDROUT File

An ADDROUT (address output) file is a record address file on disk produced by the Disk Sort program. It contains addresses of records in a disk file.

When an RPG II program uses an ADDROUT file, it reads a disk address from the ADDROUT file. The program then locates and reads records located at that address in the original disk file. Records are read in this manner until either the end of the ADDROUT file is reached or the program ends due to the end of file condition of another file. See *Column 17, End of File* in this chapter for more information about the second condition.

Sequential By Key

The sequential-by-key method of processing applies only to indexed disk files that are used as primary or secondary files.

Records are read in ascending key sequence (the order in which the record keys are arranged in the index portion of the file). The program reads records until all records in the file are processed or the program ends due to the end of file condition of another file. See *Column 17, End of File* for more information about the second condition.

Sequential Within Limits

The sequential-within-limits method applies only to indexed disk files that are used as primary and secondary files. Records that are identified in certain segments of the index are read. The segments are identified by sets of limits in records from a record address file. A record address file using limits can be located on disk, punched in cards, or entered by the printer-keyboard. A set of limits identifying a segment consists of the lowest record key and the highest record key from the segment.

The program uses one set of limits at a time. Records are read in the order in which the record keys are arranged in the segment of the index identified by the limits. When the records identified in one segment have been read, the program reads another set of limits from the record address file. The program continues reading records in this manner until either the end of the record address file is reached or the program ends due to the end of file condition of another file. See *Column 17, End of File* in this chapter for more infomation about the second condition.

The format of the records in a record address file containing limits must conform to these rules:

1. Only one set of limits is allowed per record in the record address file. The length of the RAF record, therefore, must be twice the length of the record key.

2. The low record key must begin in position one of the record. The high record key must immediately follow the low record key. A record key can be from 1-29 characters in length. No blanks are allowed between the two keys.

3. The low record key and the high record key must be equal in length and each key must be equal in length to the key field length specified in columns 29-30. Therefore, leading zeros may be necessary in specifying numeric record keys.

4. An alphameric record key may contain blanks.

The same set of limits can appear in more than one record address record, Data records, therefore, can be processed as many times as you wish.

The two record keys in a set of limits can be equal. The segment of the index in this case contains only one record key.

Random

The two methods, random by relative record number and random by key, apply to chained files only. They require the use of the CHAIN operation code. The records of a file to be read or written must be processed by the CHAIN operation code. The records are read or written only when the CHAIN statements that identify them are executed.

For sequential and direct files, relative record numbers must be used to identify the records. Relative record numbers identify the positions of the records relative to the beginning of the file. For example, the relative record numbers of the first, fifth, and seventh records in a file are 1, 5, and 7 respectively.

For indexed filer;, record keys must be used to identify the records. A record key is the information from the key field of a record. The information is used in the index portion of the file to identify the record.

Records are read during the calculation phase of the program. Therefore, they can be executed during detail or total calculation. Note then, that fields of records read from chained update files can be read and altered during total calculations and the records can be updated (written back on the file with alterations) during total output; the same also applies to detail calculations and detail output.

COLUMNS 29-30 (LENGTH OF KEY FIELD OR RECORD ADDRESS FIELD)

Entry	*Explanation*
Number	Length of record key or disk address

Columns 29-30 apply only to indexed disk files and record address files. Use it to indicate:

1. The length of the record keys in indexed files and record address files that contain limits.

2. The length of the disk addresses in ADDROUT files.

All of the key fields in the records in an indexed file must be the same length. The maximum is 29 characters. All of the disk addresses contained in an ADDROUT file are three characters long.

COLUMN 31 (RECORD ADDRESS TYPE)

Entry	*Explanation*
A	Record keys are used in processing and loading indexed files.
I	The file is being processed by using disk addresses from the ADDROUT file or the file is an ADDROUT file consisting of disk addresses.
Blank	3. Relative record numbers are used in processing sequential and direct files. 2. **A** sequential or direct file is being loaded. 3. Records are read consecutively.

Column 31 applies to disk files specified as input, update, or chained output files. It indicates the way in which records in the file are identified (see Figure 4-3). Together, columns 28 and 31 indicate:

1. the method by which records are read from the file

2. a direct file load.

For ADDROUT files, column 31 must contain an **I**. It indicates that disk addresses are used in processing.

PRIMARY AND SECONDARY FILES		
Method	Column 28	Column 31
Consecutive	Blank	Blank
By ADDROUT	R	I
Sequential By Key	Blank	A
Sequential Within Limits	L	A

CHAINED FILES		
Method	Column 28	Column 31
Random By Relative Record Number	R	Blank
Random By Key	R	A
Direct File Load (Random Load)	R	Blank *

* A direct file load requires an *O* in column 15 and a *C* in column 16.

Figure 4-3. SpecificationsIdentifying Methods for Retrieving Records

COLUMN 32 (FILE ORGANIZATION OR ADDITIONAL I/O AREA)

Entry	*Explanation*
I	Indexed file
T	ADDROUT file
1-9	Sequential file or direct file. Use two input/output areas for the file.
Blank	Sequential file or direct file. Use one input/output area for the file.

Use column 32 to (1) identify the organization of all files except ADDROUT files (2) identify ADDROUT files, and **(3)** indicate whether one or two input/output areas are to be used for sequential files or direct files.

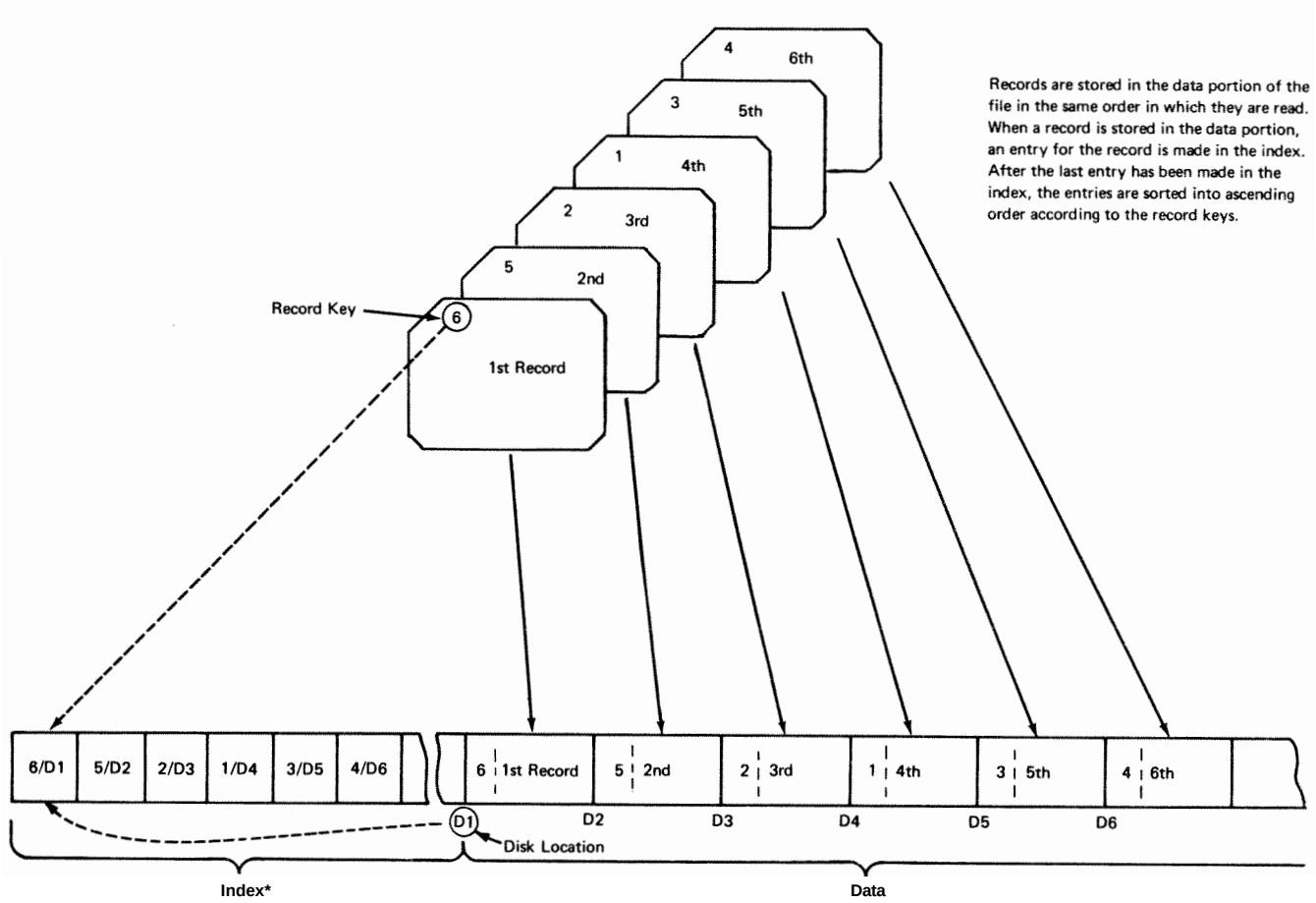
File Organization

File organization is the arrangement of records in a file. The three types are indexed, direct, and sequential. Files organized in these ways are called indexed files, direct files, and sequential files, respectively.

Indexed Files

An indexed file is a disk file in which the location of records is recorded in a separate portion of the file called an index. The index and its associated file occupy adjacent positions on disk. The index contains the record key and disk address of every record (Figure 4-4).

A record key is the information from the key field of a record. The record key can be used to identify the records of an indexed file. Record keys are always required in an indexed file. Indexed files may be loaded with the keys in ascending sequence or keys in non-ascending sequence. After a file is loaded in non-ascending key sequence, the keys in the index are placed in ascending sequence. See Column 66 of the File Description sheet for a definition of the unordered load function.



'Entries are of the form record-key/disk-location (D1=1st disk location, D2=2nd disk location, and so on)

ART: 55013

The order of the records in the data portion remains unchanged when the entries in the index are sorted.

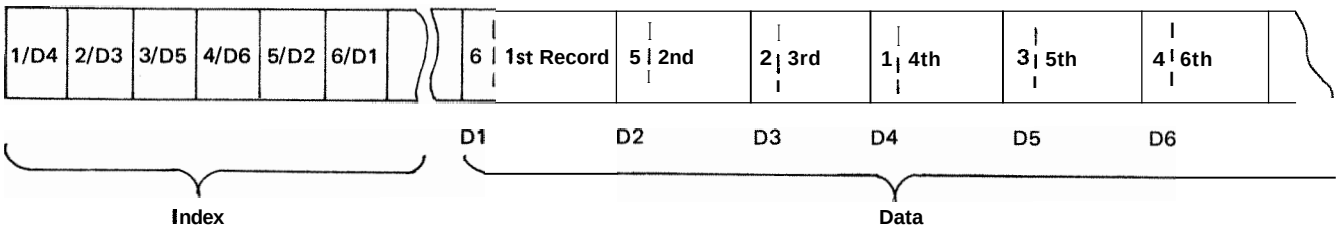


Figure 4-4. Indexed File Organization

Direct Files

Direct files are disk files in which records are assigned specific record positions. Regardless of the order in which the records are put in the file, they always occupy a specific position (a specific disk addres~).Relative record numbers identify the relative position of a record within the file.

Before a direct file is loaded the entire disk area for the direct file is cleared to blanks. Spaces are reserved in a direct file for records not available at the time the file is loaded. (See Figure 4-5.)

Sequential Files

Sequential files are files in which the order of the records is determined by the order in which the records are put in the file. For example, the tenrh record put in the file occupies the tenth record position.

Files other than disk files are always sequential files. Disk files can be sequential, direct, or indexed files, except when the files are used as demand files. Demand files ***must*** be sequential files.

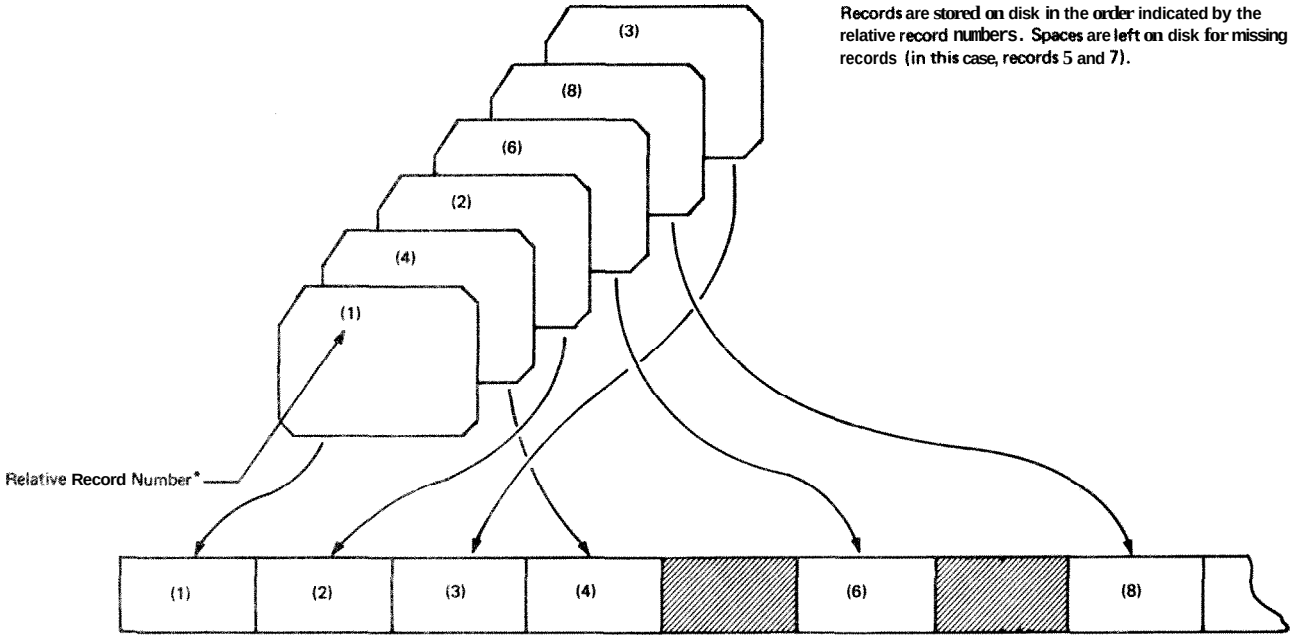
Additional Input/Output Area

Normally the program uses one input/output area for each file. A second area, however, can be used for sequential and direct disk fifes and non-disk files, specified as input or output files in column 15. Additional input/output areas cannot be used for table or demand files. The devices associated with these files can be the disk and MFCU for input or output files, and the printer for output files only. If you want two areas to be used for a card file, do not specify stacker selection for the records in the file. Stacker selection is described under ***Column*** 42, ***Stacker Select*** in Chapter 7.

The use of two I/O areas increases the efficiency of the program. However, it also increases the size of the program. Therefore, before you indicate that two areas are to be used for a file, be sure that the increase in size will not make your program exceed the capacity of your system.

ADDROUT Files

When describing an ADDROUT file, you must place a T in column 32. The ADDROUT file must be a disk file.



* The programmer usually derives relative record numbers from information in the records.

Figure 4-5. Direct File Organization

COLUMNS 33-34 (OVERFLOW INDICATORS)

Entry	*Explanation*
OA-OG, OV	An overflow indicator is used to condition records in the file. The indicator specified is the one used.
Blank	No overflow indicator is used.

Columns 33-34 apply to output files assigned to the printer. Use these columns to indicate that you are using an overflow indicator to condition records being printed in the file. Any overflow indicators used in a program must be unique for each output file assigned to the printer. The use of overflow indicators is described under ***Overflow Indicators*** in Chapter ***10.*** Note that only one overflow indicator can be assigned to a file. Do not assign overflow indicators to a console file.

COLUMNS 35-38 (KEY FIELD STARTING LOCATION)

Entry	*Explanation*
1-4096	Record position in which the key field begins.

Columns 35-38 apply to indexed disk files only. An entry ***must*** be made in these columns for an indexed disk file. Use them to identify the record position in which the key Geld begins. The key field of a record is the field that contains the information that identifies the record. The information is used in the index portion of the file. The key Geld must be in the same location in all of the records in the file.

The number you place in these columns must end in column 38. Leading zeros can be omitted.

COLUMN 39 (EXTENSION CODE)

Entry	*Explanation*
E	Extension specifications further describe the file.
L	Line counter specifications further describe the file.

Column **39** applies only to (1) table and array files that are to be read during program execution and (2) record address files and (3) output files that are assigned to the printer. Use it to indicate whether the file is further described on the Extension sheet or the Line Counter sheet. Output files that are assigned to the printer must be described on the Line Counter sheet. Table, array, and record address files must be described on the Extension sheet.

COLUMNS 40-46 (DEVICE)

Entry	*Explanation*
MFCU1	Multi-Function Card Unit. The cards are in the primary hopper.
MFCU2	Multi-Function Card Unit. The cards are in the secondary hopper.
PRINTER	Printer (whole carriage). If the dual carriage feature is used, this entry refers to the left carriage.
PRINTR2	Right carriage of the printer (dual carriage feature only).
CONSOLE	Printer-keyboard,
DISK	Disk unit.

Use columns 40-46 to identify the input/output device to be used for the file. All entries must begin in column 40. The devices that can be used depend upon the form of the records (Figure 4-6).

Console

Records entered from a console file will be treated as any other records. Every character to be entered must be keyed in. Key the information into the fields as you would into a card. Fields must be properly right-justified and left-justified by you. The system does not do this for you. You must space where blanks appear in a record.

If the operator hits the "cancel" key, those characters of the record already accepted will be "erased", the keying element will return to column I, and the operator may begin to key the record in again.

If the operator keys in more characters than are specified possible for a record, the record is automatically "cancelled" and the operator is notified to key it in again.

Printer Files

The dual carriage feature allows you to produce two separate printer output files in one program. The two output files assigned to the printer must be named PRINTER and PRINTR2. The forms used for the two files are special forms that are both narrower than the regular form for your printer (such as checks or invoices). One form is controlled by the left carriage of the printer (device name PRINTER) and the other form is controlled by the right carriage (device name PRINTR2). The two printer files are considered as separate output files and must be described as such. There are no programming restrictions for these files that are different from a normal printer file (spacing and skipping are independent for each carriage). Note, however, that care must be taken when describing the location (end position) of output fields, to avoid printing in positions where there is no form.

Figure 4-7 shows the columns that cannot be used for the devices named. The shaded columns must be blank for the device named in the specification line. (MFCU is MFCU1 or MFCU2, and PRINTER is PRINTER or PRINTR2.)

COLUMNS 47-59

Columns 47-59 are not used. Leave them blank.

FILE	FORM	POSSIBLE DEVICES
Primary or Secondary Input Files	Cards	MFCU1 or MFCU2
	Disk	DISK
	Keyed in by operator*	CONSOLE
Record Address Files (Containing Record-Key Limits	Cards	MFCU1 or MFCU2
	Disk	DISK
	Keyed in by operator*	CONSOLE
Record Address Files (Containing Disk Addresses (ADDROUT File)	Disk	DISK
Demand Files	Cards	MFCU1 or MFCU2
	Disk	DISK
	Keyed in by operator*	CONSOLE
Table Files	Cards	MFCU1 or MFCU2
	Disk	DISK
	Keyed in by operator*	CONSOLE
Chained Input Files	Disk	DISK
Update Files (Primary, Secondary, or Chained)	Disk	DISK
Combined Files (Primary or Secondary)	Cards	MFCU1 or MFCU2
	Keyed in by operator**	CONSOLE
Output Files	Cards	MFCU1 or MFCU2
	Disk	DISK
	Printed pages	PRINTER, PRINTR2, or CONSOLE
Display File	Printed pages	CONSOLE

* Records are not typed when they are keyed into the program.

** Records are typed when they are keyed into the program.

Figure 4-6. **Device Assignment**

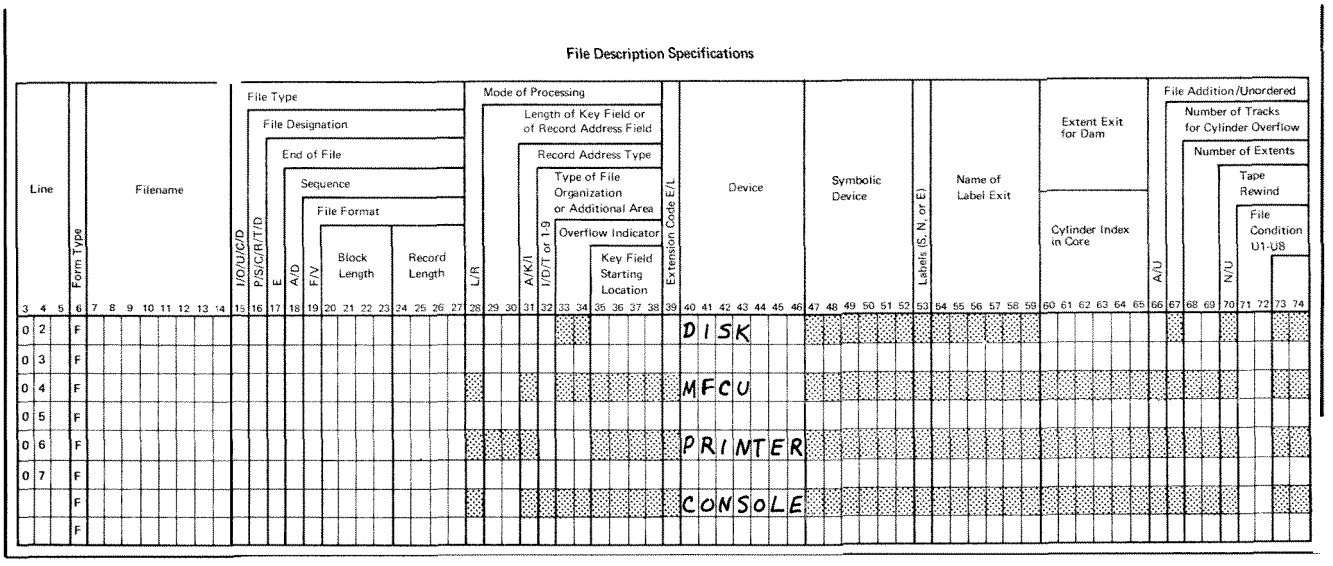


Figure 4-7. Columns That Do Not Apply to Device Named

COLUMNS 60-65 (CYLINDER INDEX IN CORE)

Entry	*Explanation*
1-9999	Number of bytes reserved for the cylinder index

Columns 60-65 apply only to indexed files processed randomly. Entries must be right-justified. Leading zeros are not required. An entry should be made in columns 60-65 if you are processing an indexed file randomly. You can specify up to 9999 bytes for the cylinder index in core.

The cylinder index is a table containing entries for tracks in the index portion of a data file. Each entry contains a track number and the highest key field associated with that track. Figures 4-8 through 4-10 show the layout of an indexed file, INDEXT, and its associated cylinder index. Figure 4-8 shows how the index portion of the file, INDEXT, might be laid out on disk. The location of specific record keys by cylinder and track is shown in Figure 4-9.

Cylinder	Track	Record Keys
10	20	1-30
	21	31-60
11	22	61-90
	23	91-120
12	24	121-150
	25	151-180

Figure 4-9. Location of Records for INDEXT by Cylinder and Track

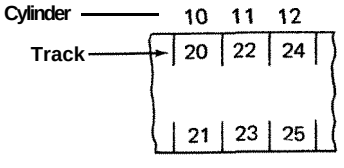


Figure 4-8. Disk Layout Form for INDEXT

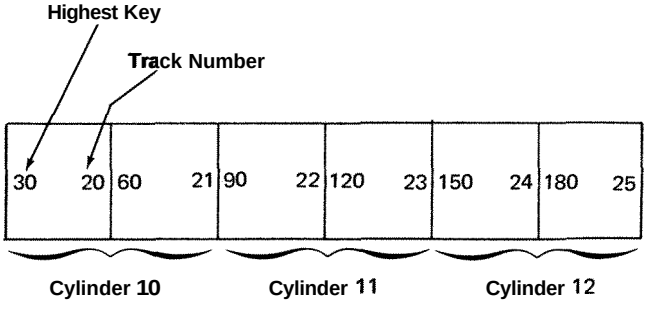


Figure 4-10. Cylinder Index in Core for INDEXT

The use of the cylinder index significantly reduces the amount of time needed to process an indexed file because it enables the system to go more directly to the specific record you want. With the cylinder index, the system can find a specific record by searching only a small portion of the file index. Without the cylinder index, however, all index entries which precede the record you want must be searched. Using the cylinder index shown in Figure 4-10 the record with key field 125 can be found in this manner:

- Search the cylinder index until the first key field higher than 125 is located. In this instance that key is 150; it has track 24 associated with it.
- Search track 24 in the file index until key 125 is located.
- Chain directly to the associated data record.

In columns 60-65 you specify the number of storage positions (bytes) you wish reserved for the cylinder index. Using the amount of core storage you specify, the system builds the most efficient cylinder index it can for you. The cylinder index is built immediately before your RPG II program is executed.

For efficient processing the cylinder index should be large enough to contain one entry (key and track number) for each track of index in the data file. Each entry is equal to key field length plus 2 multiplied by the number of tracks in the file index. Therefore, for an indexed file having a key length of 4 and 10 tracks of file index, the most efficient cylinder index requires 60 bytes of storage (4 plus 2 times 10).

If the storage space you specify in columns 60-65 is not large enough to contain one entry for each track of file index, the system will construct a table containing one entry for every cylinder of file index. Or the cylinder index might only contain one entry for every other cylinder. As the number of entries in the cylinder index become fewer, the amount of processing time increases.

COLUMN *66* (FILE ADDITION)

Entry	*Explanation*
A	New records will be added to the file.
U	Records are to be loaded for an indexed file in unordered sequence (non-ascending sequence),

Column 66 applies to sequential disk and indexed disk files only. This column indicates:

1. you want the program to add new records to the file, or
2. records are loaded in unordered (non-ascending) sequence.

Records added to a sequential file are added at the end of the file. Records added to an indexed file are added at the end of the file and entries for the new records are made in the index. The index is then reorganized so that the record keys (including the new ones) are in ascending order.

File addition in column 66 cannot be specified for (1) direct files, or (2) indexed files from which records are read using the sequential-within-limits method. (New records may be inserted in a direct file by specifying the file as an update file processed consecutively or by the CHAIN operation code.)

After a file has been loaded on disk, it may become necessary to add records to the file. Records can be added at detail, total, or exception time during the program cycle. The records to be added may:

1. Contain keys that are above the highest key presently in the file (in this case, the records constitute an extension of the file), or

2. Contain keys that are either lower than the lowest key presently in the file, or fall between keys already in the file.

To add a record, the program searches the index of the file to determine if the record is on the file. If the record is on the file a halt occurs; otherwise, the record is added. The following options will be given if a halt occurs:

1. Bypass the duplicate record, or

2. Bypass the duplicate record and turn on the LR indicator and perform all end-of-job and final total procedures, or

3. Cancel the entire job.

In Figure 4-11, combinations of entries in File type (column 15) and File Addition (column 66) show the functions that can be performed for indexed files (I in column 32).

Column 15	Column 66	Function
O	Blank	Load records in ascending key sequence to an indexed file.
O	U	Load records in unordered key sequence to an indexed file.
O	A*	Add records to an existing indexed file.
I	Blank	Read records of an indexed file without adding new records or updating records.
I	A*	Read records of an indexed file and add new records to the file that are not presently there. No updating is performed.
U	Blank	Update records of an indexed file without adding new records.
U	A"	Update records of an indexed file and add new records to the file.

* An A in column 66 requires an ADD entry in columns 16-18 of the Output-Format sheet.

Figure 4-11. Various Functions Performed on Indexed Files

Example

Figure 4-12 shows how records can be added to an indexed disk file. The new records are contained in a card file, CARDIN. The file INDEXED is the existing disk file to which new records will be added. A printer file, PRINT, will provide a report showing all the records in CARDIN, with an indication of which records were added to INDEXED and which records were not added.

On the File Description sheet, an ***A*** must appear in column 66 for the file INDEXED, and on the Output sheet ADD must appear in columns 16-18 for the new record to be added.

As defined on the Input sheet, all the cards in CARDIN should have an ***A*** in position 80. The code identifies a record to be added to the disk file, and this record type is assigned indicator 01. On the Output sheet, notice that when 01 is on, the data from the card is written on the disk file INDEXED and is also printed on the file PRINT to keep a visual report of new records.

There may be records in CARDIN that do not belong in that file, or some records may have a keypunch error. These records are identified on the Input sheet as not having the character ***A*** in position 80. These records will turn on indicator 02, and are not to be added to the disk file INDEXED. However, these records are printed on the file PRINT for a visual report, but they must be identified in the printed report as records that were not added to the disk file INDEXED. On the Output sheet, the constant 'RECORD NOT ADDED' is printed only on indicator 02, indicating a record that was not added to the disk file. In this manner, there will be a printed report of all records in CARDIN, and the records not added to INDEXED are identified by the constant 'RECORD NOT ADDED'.

COLUMN *67*

Column 67 is not used. Leave it blank.

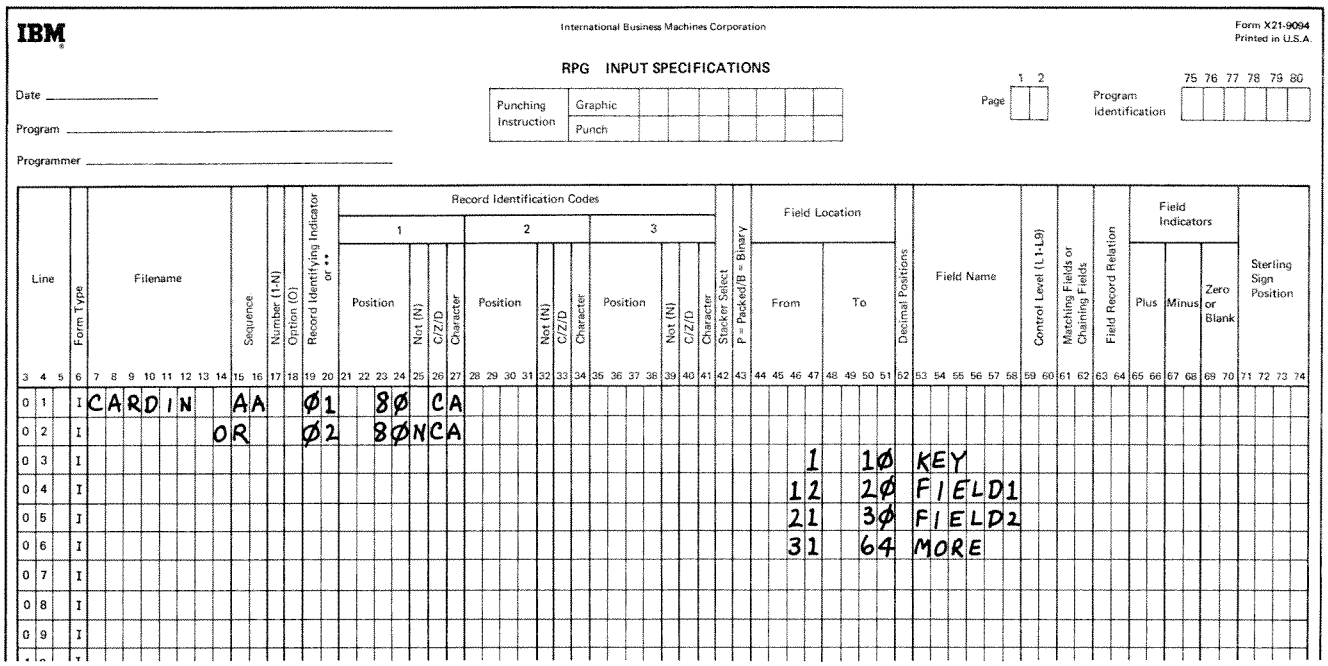
File Description Specifications

Line	Form Type	Filename	I/O/U/C/D	P/S/C/R/T/D	E	A/D	F/V	Block Length	Record Length	L/R	A/K/I	I/D/T or 1-9	Overflow Indicator	Key Field Starting Location	Extension Code E/L	Device	Symbolic Device	Labels (S, N, or E)	Name of Label Exit	Extent Exit for Dam	Cylinder Index in Core	A/U	Number of Tracks for Cylinder Overflow	Number of Extents	Tape Rewind	File Condition U1-U8
02	F	CARDIN	I	P	E	F		96	96							MFCU1										
03	F	INDEXED	O			F		256	64		10A	I		1		DISK						A	01			
04	F	PRINT	O			F		132	132				OA			LPRINTER										
05	F																									
06	F																									
07	F																									
	F																									
	F																									

Line Counter Specifications

Line	Form Type	Filename	1 Line Number	1 FL or Channel Number	2 Line Number	2 OL or Channel Number	3 Line Number	3 Channel Number	4 Line Number	4 Channel Number	5 Line Number	5 Channel Number	6 Line Number	6 Channel Number	7 Line Number	7 Channel Number	8 Line Number	8 Channel Number	9 Line Number	9 Channel Number	10 Line Number	10 Channel Number	11 Line Number	11 Channel Number	12 Line Number	12 Channel Number
11	L	PRINT	66	FL	60	OL																				
12	L																									
13	L																									

Figure *4-12.* **File *Addition (Part 1 of 2)***



IBM

International Business Machines Corporation

Form X21-9094
Printed in U.S.A.

RPG INPUT SPECIFICATIONS

Date _____

Program _____

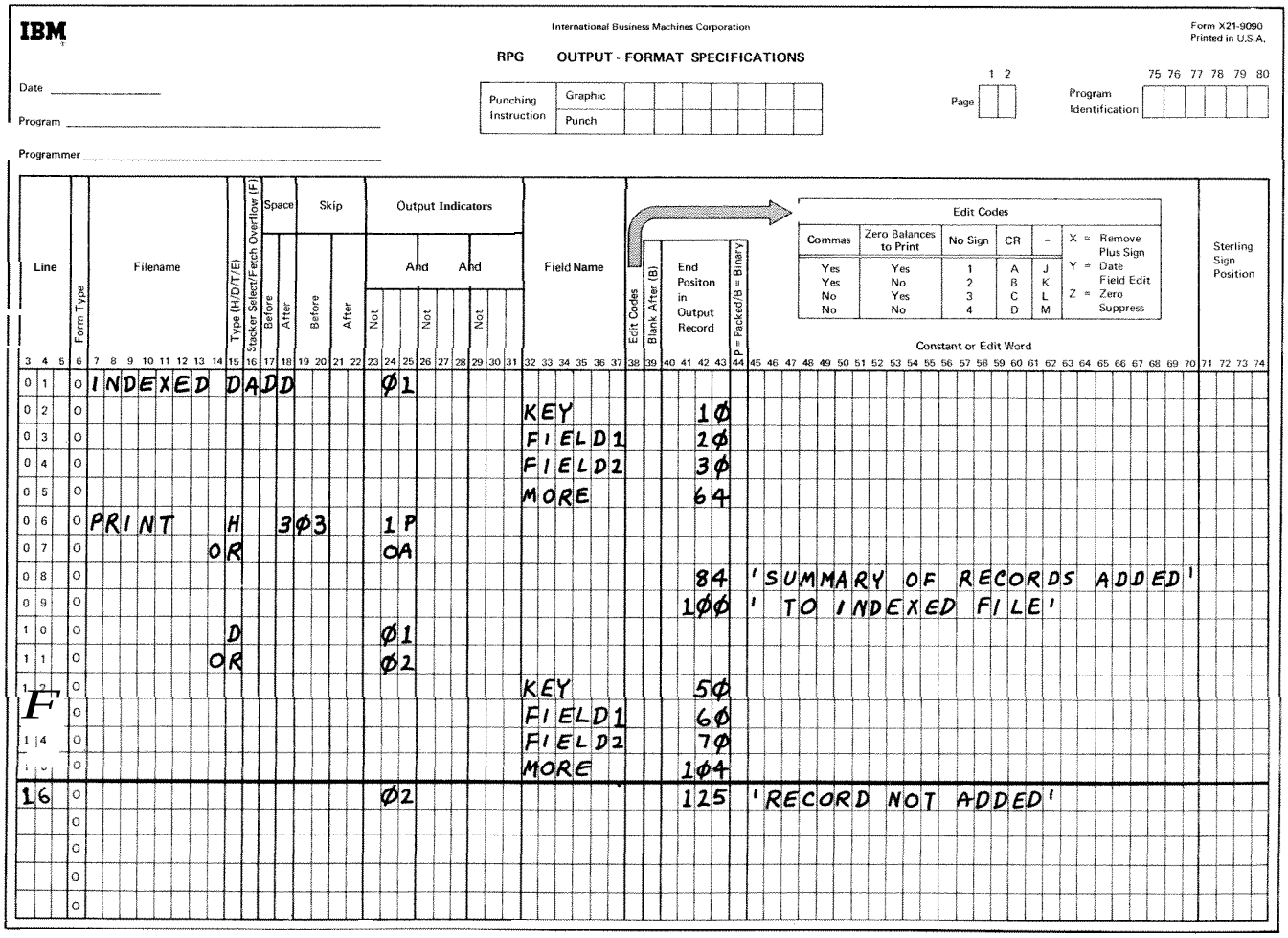
Programmer _____

Punching Instruction	Graphic							
	Punch							

Page 1 2

Program Identification 75 76 77 78 79 80

Line	Form Type	Filename	Sequence	Number (1-N)	Option (O)	Record Identifying Indicator or **	Record Identification Codes 1: Position	1: Not (N)	1: C/Z/D	1: Character	2: Position	2: Not (N)	2: C/Z/D	2: Character	3: Position	3: Not (N)	3: C/Z/D	3: Character	Stacker Select	P = Packed/B = Binary	Field Location: From	Field Location: To	Decimal Positions	Field Name	Control Level (L1-L9)	Matching Fields or Chaining Fields	Field Record Relation	Field Indicators: Plus	Field Indicators: Minus	Field Indicators: Zero or Blank	Sterling Sign Position
01	I	CARDIN	AA			01	80		C	A																					
02	I		OR			02	80	N	C	A																					
03	I																				1	10		KEY							
04	I																				12	20		FIELD1							
05	I																				21	30		FIELD2							
06	I																				31	64		MORE							
07	I																														
08	I																														
09	I																														



IBM

International Business Machines Corporation

Form X21-9090
Printed in U.S.A.

RPG OUTPUT - FORMAT SPECIFICATIONS

Date _____

Program _____

Programmer _____

Punching Instruction	Graphic							
	Punch							

Page 1 2

Program Identification 75 76 77 78 79 80

Edit Codes

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

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

Constant or Edit Word

Line	Form Type	Filename	Type (H/D/T/E)	Stacker Select/Fetch Overflow (F)	Space: Before	Space: After	Skip: Before	Skip: After	Output Indicators: And	And	And	Field Name	Edit Codes	Blank After (B)	End Position in Output Record	P = Packed/B = Binary	Constant or Edit Word	Sterling Sign Position
01	O	INDEXED	D	A	D	D			01									
02	O											KEY			10			
03	O											FIELD1			20			
04	O											FIELD2			30			
05	O											MORE			64			
06	O	PRINT	H				3	03	1P									
07	O		OR						OA									
08	O														84		'SUMMARY OF RECORDS ADDED'	
09	O														100		' TO INDEXED FILE'	
10	O		D						01									
11	O		OR						02									
12	O											KEY			50			
F	O											FIELD1			60			
14	O											FIELD2			70			
	O											MORE			104			
16	O								02						125		'RECORD NOT ADDED'	
	O																	
	O																	
	O																	
	O																	

Figure 4-12. **File Addition (Part 2** of 2)

COLUMNS 68-69 (NUMBER OF EXTENTS)

Entry	*Explanation*
01-80	Number of volumes (disks) that contain the disk file.

Columns 68-69 *must* contain an entry for each disk file. The entry must end in column 69. These columns define the number of volumes (disks) on which the disk file is located. A disk file must occupy consecutive cylinders on each volume. For instance, a disk file could not occupy cylinders 20-30 and 41-50 on one volume. The file could occupy cylinders 20-40 on that volume, or the data in cylinders 41-50 could be placed on another volume.

The number of volumes you can use depends on the mode of processing and number of drives used. For single volume files the entry in columns 68-69 is always 01. For multi-volume files, determine the entry as follows:

1. *Consecutive processing. All* disk files processed consecutively must be located on removable disks. If a multi-volume file is to be processed consecutively, the entry in columns 68-69 can be from 2-80. (If 1 drive is used for multi-volume files, only 1 volume can be on-line at any given time; and if 2 drives are used, only 2 volumes can be on-line at any given time.)

2. *Sequential or Random Processing. A* disk file to be processed sequentially or randomly can be located on a fixed disk, a removable disk, or both. To process a multi-volume disk file sequentially or randomly, the entire file must be available to the system at any given time. Therefore, the entire file must be on-line. (This is unlike consecutive processing of multi-volume files in which portions of the file can be off-line.) If 1 drive is used for multi-volume files, the entry in columns 68-69 is 2. If 2 drives are used for multi-volume files the entry in columns 68-69 can be 3 or 4. Figure 4-13 shows the maximum number of volumes allowed for each processing method and number of drives available

	ONE DRIVE		TWO DRIVES	
	Maximum number of volumes allowed	Maximum number of volumes on-line	Maximum number of volumes allowed	Maximum number of volumes on-line
Consecutive processing (removable disks only)	80	1	80	2
Sequential or Random Processing (removable or fixed disks)	2	2	4	4

Figure **4-13.** Number of Volumes Allowed for Multi-Volume Files

COLUMNS 71-72 (FILE CONDITION)

Entry	*Explanation*
U1-U8	The file is conditioned by the specified external indicator.
Blank	The file is not conditioned by an external indicator.

Columns 71-72 apply to input (excluding table input files), update, output, and combined files. These columns indicate whether or not the file is conditioned by an external indicator. A file conditioned by an external indicator is used only when the indicator is on. When the indicator is off, the file is treated as though the end of the file had been reached. (No records can be read from or written in the file.) See *Indicators, External Indicators,* in Chapter 10 for more information.

COLUMNS 73-74

Columns 73-74 are not used. Leave them blank.

COLUMNS 75-80 (PROGRAM IDENTIFICATION)

See Chapter 2.

COLUMN 70

Column 70 is not used. Leave it blank.

FILE DESCRIPTION CHARTS

The File Description charts in the following pages are for:

1. *Disk* files, presented by disk file organization and processing method.

2. *MFCU, Console,* and *Printer* files.

   - The *entries* in the chart must be made for the processing method and type of file described on that line.

   - The *shaded columns* must be blank for the file described on that line.

   - The *other columns* may be required or optional, but cannot be indicated on the chart because the entries represent information that changes from program to program.

Example

If you are updating an indexed disk file using the CHAIN operation code, look at the chart for: *indexed* disk files, random processing by CHAIN operation code. Then choose the chained update file with or without record addition.

The entries on the chart must be made for the file you are describing. The shaded columns must be blank for that file.

The remaining columns represent information that changes from program to program. For instance, in this example these columns are required but may change from one program to another: Line, Filename, Block Length, Record Length, Length of Key Field, Key Field Starting Location, and Number of Extents. Optional entries are: End of File, Sequence, and File Condition.

Figure 4-14. Processing Methods for ***Indexed*** Disk Files

DISK FILES

File Description Specifications

Type of Processing		Line	Form Type	Filename	I/O/U/C/D	P/S/C/R/T/D	E	A/D	F/V	Block Length	Record Length	L/R	A/K/I	I/O/T or 1-9	Overflow Indicator	Key Field Starting Location	Extension Code E/L	Device	Symbolic Device	Labels (S, N, or E)	Name of Label Exit	Extent Exit for Dam	Cylinder Index in Core	File Addition/Unordered A/U	Number of Tracks for Cylinder Overflow	Number of Extents N/U	Tape Rewind	File Condition U1-U8
		3 4 5	6	7 8 9 10 11 12 13 14	15	16	17	18	19	20 21 22 23	24 25 26 27	28	29 30	31	32 33 34	35 36 37 38	39	40 41 42 43 44 45 46	47 48 49 50 51 52	53	54 55 56 57 58 59	60 61 62 63 64 65	66	67	68 69	70	71 72	73 74
Seq or RAF*	by Key, no ADD		F		I	P			F					A	I			D I S K										
	by Key, no ADD		F		I	S			F					A	I			D I S K										
	by Key, with ADD		F		I	P			F					A	I			D I S K					A					
	by Key, with ADD		F		I	S			F					A	I			D I S K					A					
	by Key, no ADD		F		U	P			F					A	I			D I S K										
	by Key, no ADD		F		U	S			F					A	I			D I S K										
	by Key, with ADD		F		U	P			F					A	I			D I S K					A					
	by Key, with ADD		F		U	S			F					A	I			D I S K					A					
	by Limits (RAF)		F		I	P			F			L		A	I			D I S K										
	by Limits (RAF)		F		I	S			F			L		A	I			D I S K										
	by Limits (RAF)		F		U	P			F			L		A	I			D I S K										
	by Limits (RAF)		F		U	S			F			L		A	I			D I S K										
			F																									
Random	by CHAIN, no ADD		F		I	C			F			R		A	I			D I S K										
	by CHAIN, with ADD		F		I	C			F			R		A	I			D I S K					A					
	by CHAIN, no ADD		F		U	C			F			R		A	I			D I S K										
	by CHAIN, with ADD		F		U	C			F			R		A	I			D I S K					A					
			F																									
	by ADDROUT		F		I	P			F			R		I	I			D I S K										
	by ADDROUT		F		I	S			F			R		I	I			D I S K										
	by ADDROUT		F		U	P			F			R		I	I			D I S K										
	by ADDROUT		F		U	S			F			R		I	I			D I S K										
			F																									
Load	Unordered		F		O				F					A	I			D I S K					U					
	Ordered		F		O				F					A	I			D I S K										
			F																									
Add records only	ADD only		F		O				F					A	I			D I S K					A					

* Sequential processing by key or limits must use the file index, which is always arranged in ascending sequence. When an indexed file is processed record by record from beginning to end, the file is processed through the index using the sequential by key method.

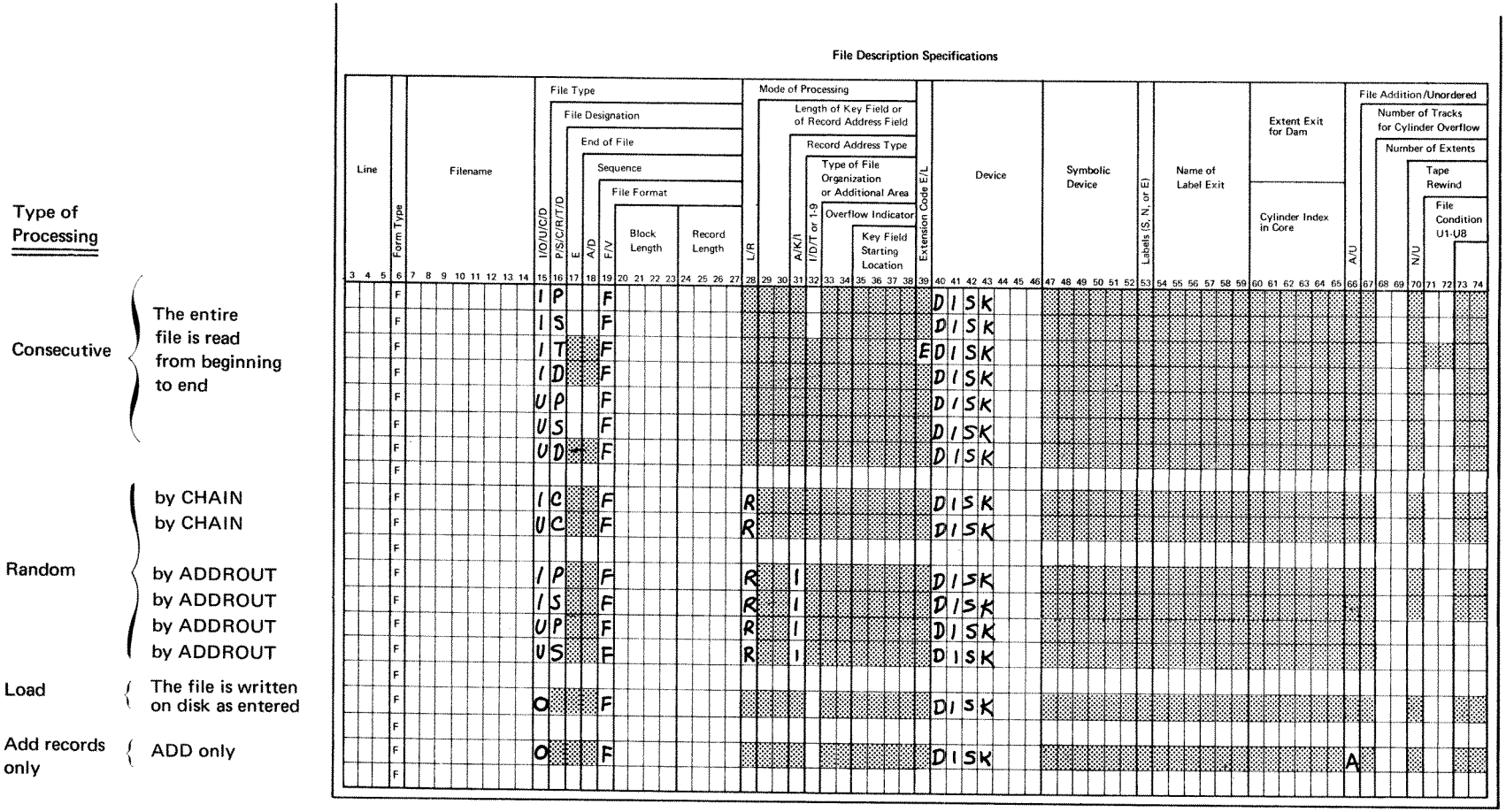


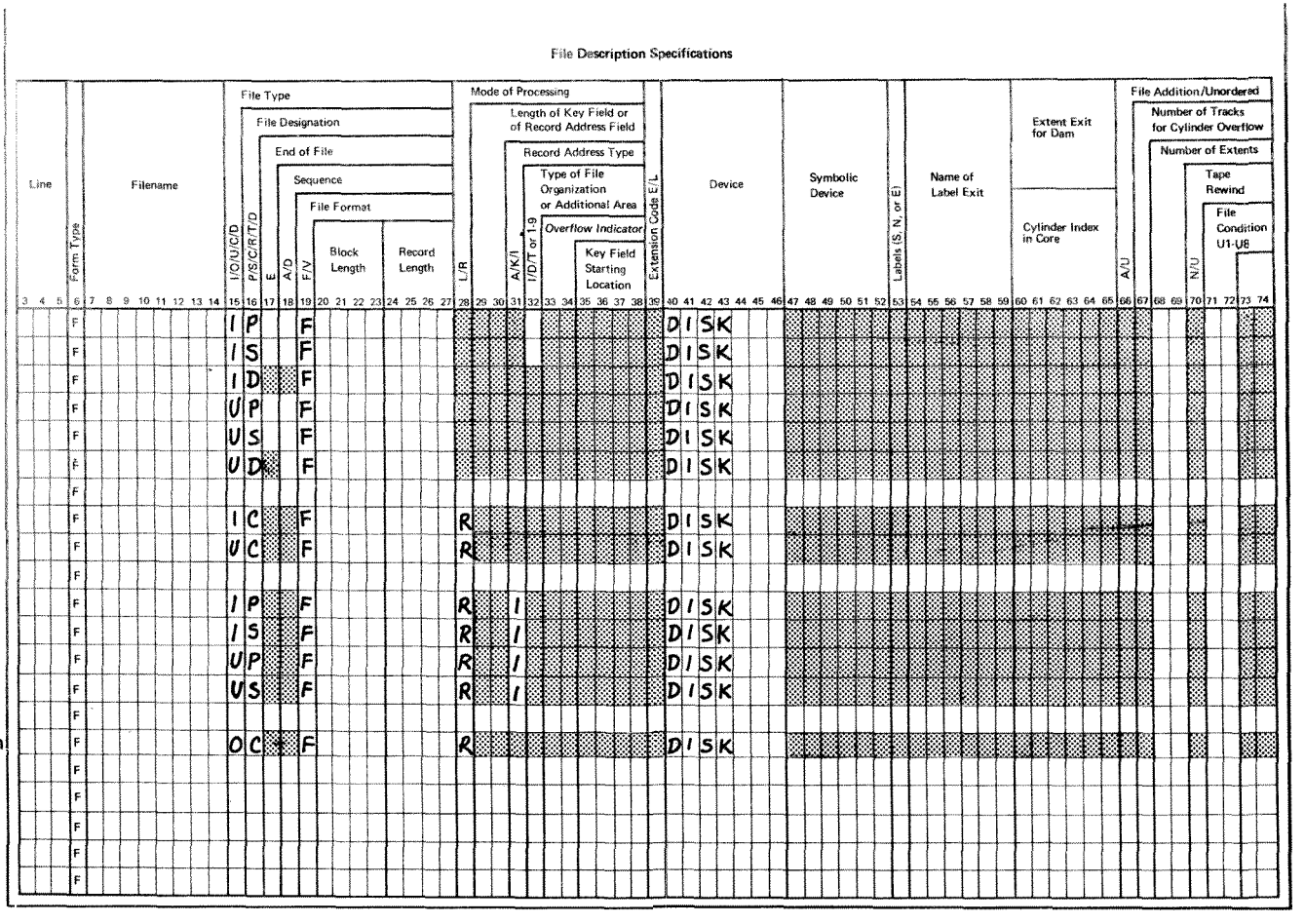
Figure 4-15. Processing Methods for *Sequential* Disk Files

Figure 4-16. Processing Methods for *Direct* Disk Files

DISK FILES

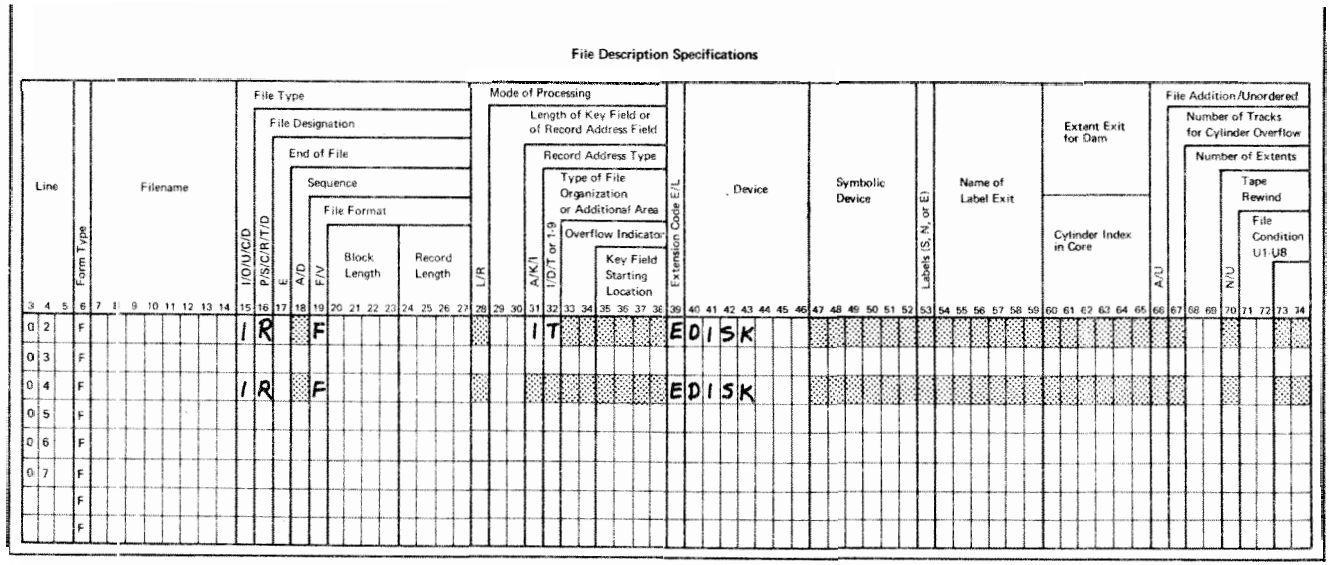
Type of Processing *

- Consecutive — The entire file is read from beginning to end.
- Random — by CHAIN; by CHAIN; by ADDROUT; by ADDROUT; by ADDROUT; by ADDROUT
- Load — Disk addresses are developed for each record entered.



* Records are inserted or changed in a direct file by defining the file as an update file processed consecutively, or an update file processed randomly by the CHAIN operation code.

Record Address Files *
Containing: 1. Disk Addresses (ADDROUT file)
2. Record Key Limits



* Record address files containing disk addresses may be associated with indexed, sequential, or direct disk files.

* Record address files containing record key limits may only be associated with indexed disk files, but may be a disk, MFCU, or console file (see charts for MFCU and console files).

Figure 4-17. Record **Address** Files Located on Disk

MFCU FILES

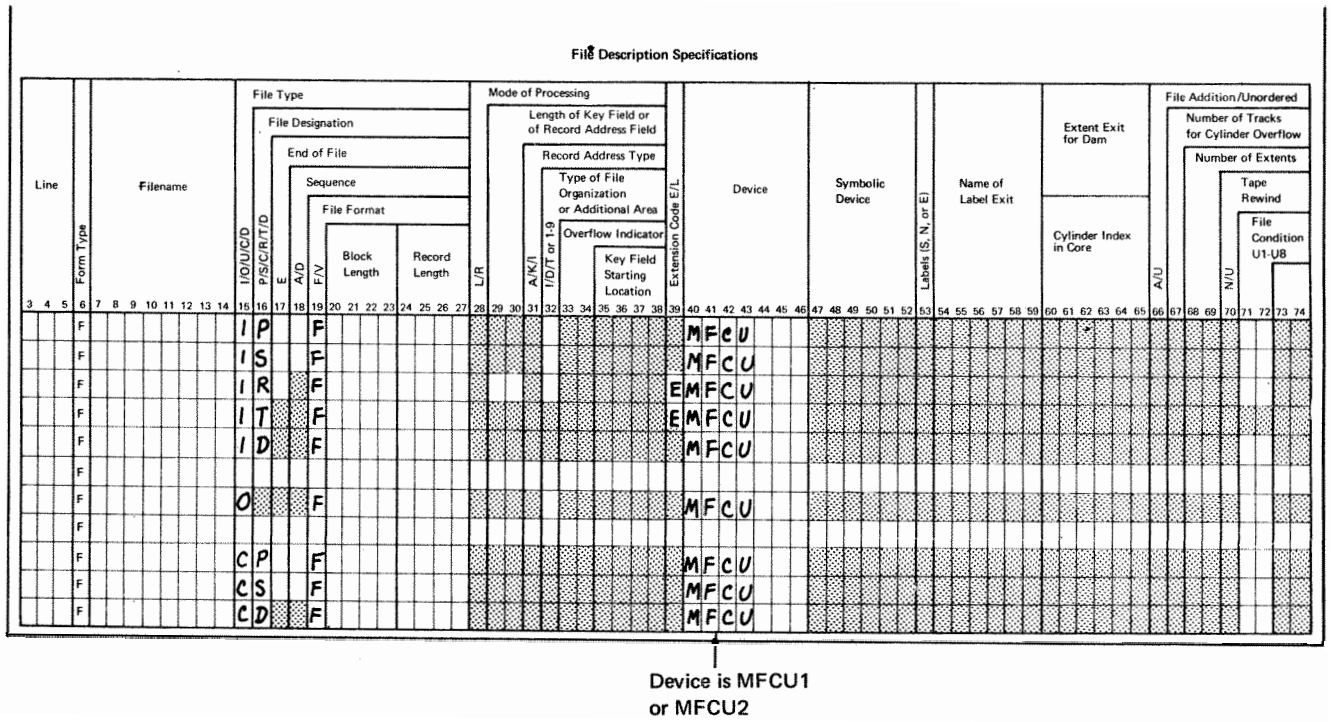


Figure 4-18. MFGU Files

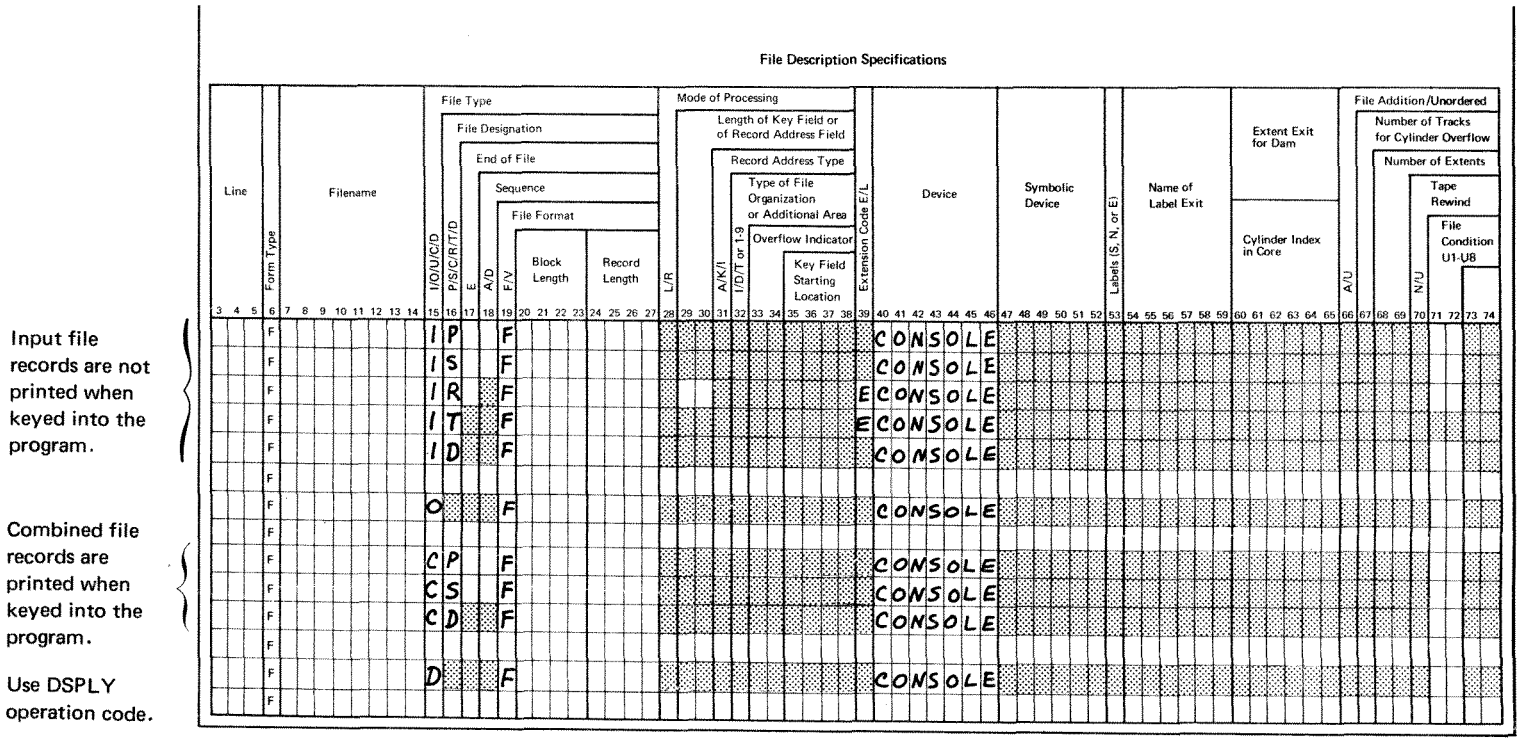


Figure 4-19. Console Files (Printer-Keyboard)

CONSOLE FILES

PRINTER FILES

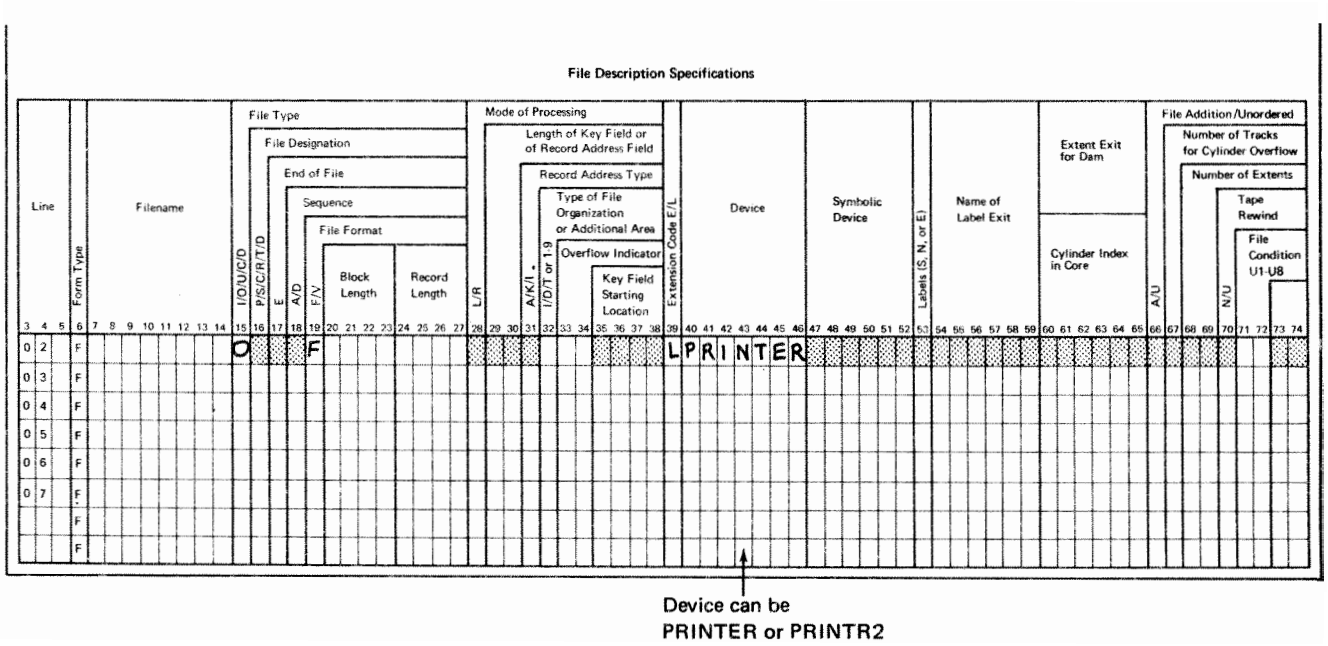


Figure 4-20. Printer Files

Extension specifications are needed to describe the record address files, tables, and arrays you may use in your job. Enter these specifications on the Extension and Line Counter sheet (Figure 5-1). Record address files require eatries on the Extension sheet in columns 11-26.

Execution time tables and arrays are described in columns 11-45. Compile time tables and arrays are described in columns 19-45. If an alternating table or array is to be specified with another table or array, it is described in columns 45-57 of the same line as the first.

A chart showing possible Extension sheet entries is at the end of this chapter (see Figure 5-6).

See *Arrays* in Chapter 10 for more information about arrays.

COLUMNS 1-2 (PAGE)

See Chapter 2.

COLUMNS 3 5 (LINE)

See Chapter 2.

COLUMN 6 (FORM TYPE)

An E must appear in column 6.

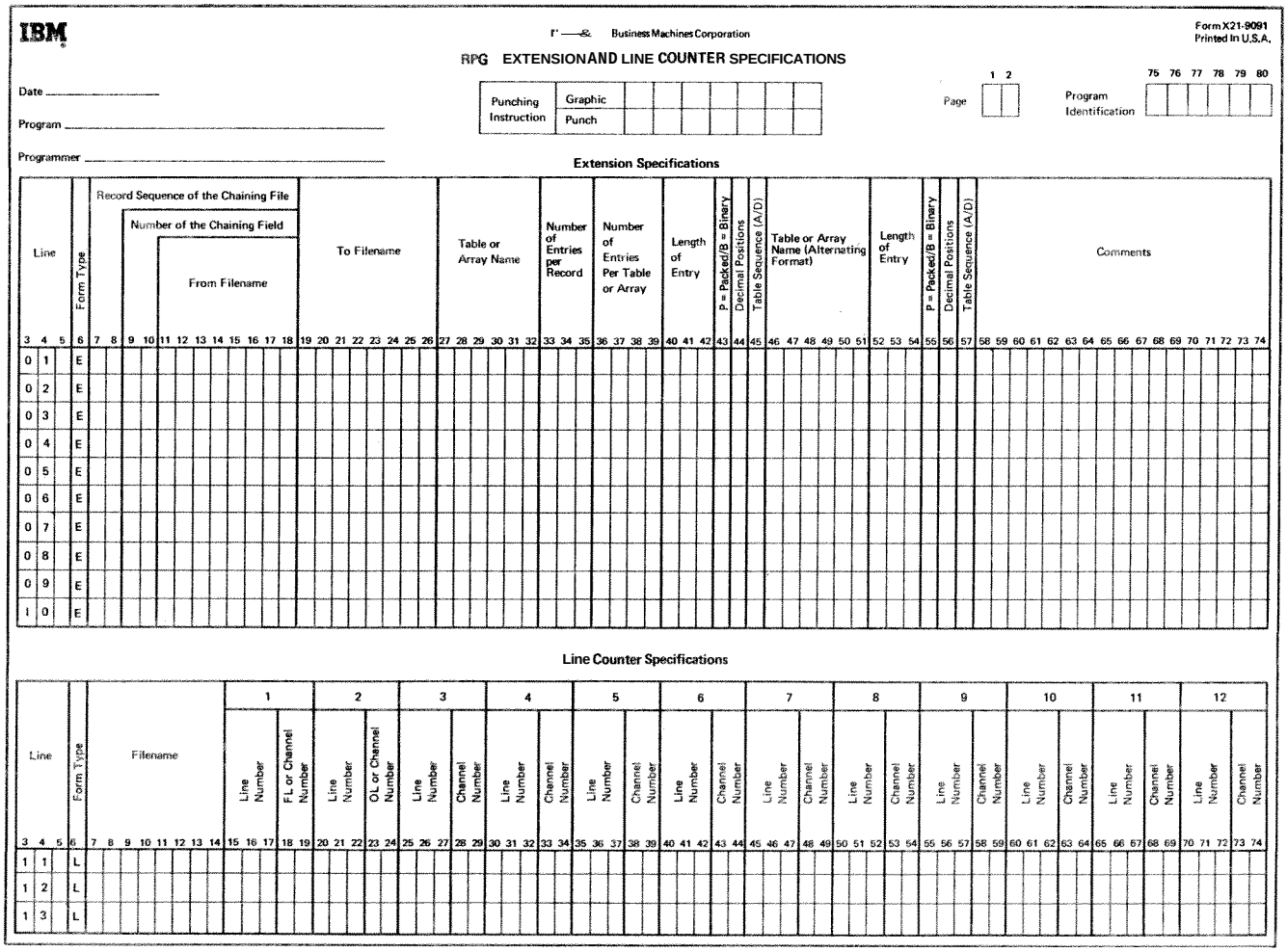


Figure 5-1. Extension and Line Counter Sheer

COLUMNS 7-10

Columns 7-10 are not used. Leave them blank.

COLUMNS 11-18 (FROM FILENAME)

Entry	*Explanation*
Record Address filename	The name of the Record Address fde.
Table Array filename	1. Table loaded at execution time.
	2. Array loaded at execution time if there is an entry in Number of Entries per Record (columns 33-35).
Blank	1. Table loaded at compilation time.
	2. Array loaded at compilation time if there is an entry in Number of Entries per Record (columns 33-35).
	3. Array loaded via input or calculations specifications if there is no entry in Number of entries per Record (columns 33-35).

Columns 11-18 are used to name a table file, array file, or record address file. Filenames must begin in column 11.

The record address filename must always be entered in these columns and in the file description specifications.

Leave columns 11-18 blank for compile time tables or arrays or for arrays loaded via input or calculations specifications.

These columns must contain the table or array filename of every execution time table or array used in your program. When the table or array is loaded at compilation time, it is compiled along with the source program and included in the object program. Thus, a table deck is not needed in addition to the object deck every time the program is run. Only those tables and arrays which do not change often should be compiled with the program.

When tables or arrays are being compiled with the program, table fde records must always follow the RPG II source program. A record with /* in columns 1 and 2 must follow the table file input records. A record with **b in columns 1-3 is also needed to separate the table or array records from the RPG II source program. Tables or arrays must be separated from each other by records with **b in columns 1-3 (Figure 1-2).

Each table or array loaded at execution time must be followed by a record with /* in columns 1-2. Short tables (tables which are not full) may be compiled with the program but a warning is issued. See *Columns 36-39* in this chapter for more information.

COLUMNS 1926 (TO FILENAME)

Entry	***Explanation***
Name of an input or update file	The file processed via the Record Address file named under From Filename.
Name of an output file	The output file to which a table or array is to be written or punched.

Columns 19-26 define the relationship between a file named in these columns and a file named in columns 11-18. Filenames must begin in column 19.

If a record address file is named under From Filename, columns 11-18, the following entry should be made under To Filename, columns 19-26; the name of the input or update file that contains the data records to be processed. Do not enter the record address filename in these columns.

If you wish a table or array to be written or punched, use columns 19-26 to enter the filename of the output file you will use to do this. This output file must have been previously named in the file description specifications. A table or array can be written on only one output device. Leave columns 19-26 blank if you do not want the table or array written or punched.

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

Since the table or array will be written or punched in the same format in which is was entered, you may want to rearrange the output table or array through output-format specifications. You may format table or array output by using exception lines to write out one item at a time (see Operation Codes, Exception in Chapter 10). Tables or arrays should be written or punched only after all records have been processed (Last Record indicator is on).

COLUMNS 27-32 (TABLE OR ARRAY NAME)

Entry	Explanation
Table or Array name	Name of each table or array used in the program.

Use columns 27-32 to name your table or array. No two tables or arrays may have the same name. The name can be from 1-6 characters long, and must begin in column 27. The first character must be alphabetic. The remaining characters can be any combination of alphabetic and numeric characters (no special characters are allowed). Blanks may not appear between characters in the name.

Table Name

Every table used in your program must be given a name beginning with the letters TAB. Any name in these columns which does not begin with TAB is considered an array name. This table name is used throughout the program. However, different results can be obtained depending upon how the table name is used. When the table name is used in Factor 2 or Result Field (on the Calculation sheet) with LOKUP operation, it refers to the entire table. When the table name is used with any other operation code, it refers to the table item last selected from the table by a LOKUP operation.

See Operation Codes, Lookup in Chapter 10 for more information.

Table files are processed in the same order as they are specified on the Extension sheet. Therefore, if you have more than one table file, remember the files are to be loaded in the same order as they appear on the sheet. When you have only one short table, you should specify it after all other tables.

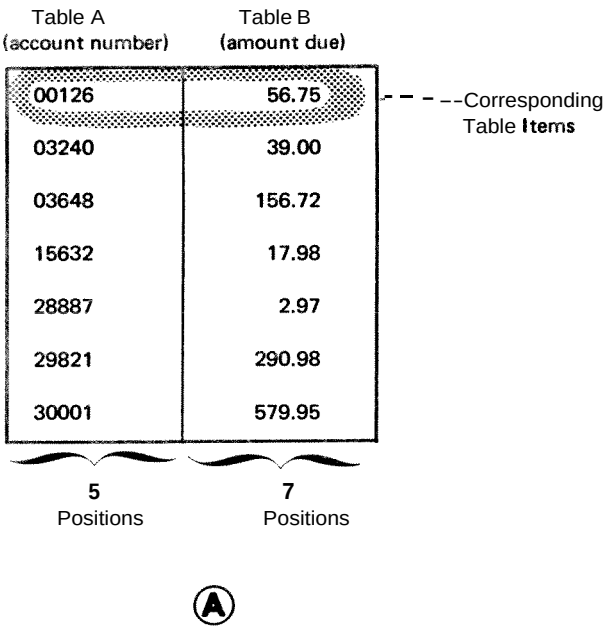
If two related tables are in alternating form in one table file, the table whose item appears first must be named in columns 27-32. The second table is named in columns 46-51 (see Example).

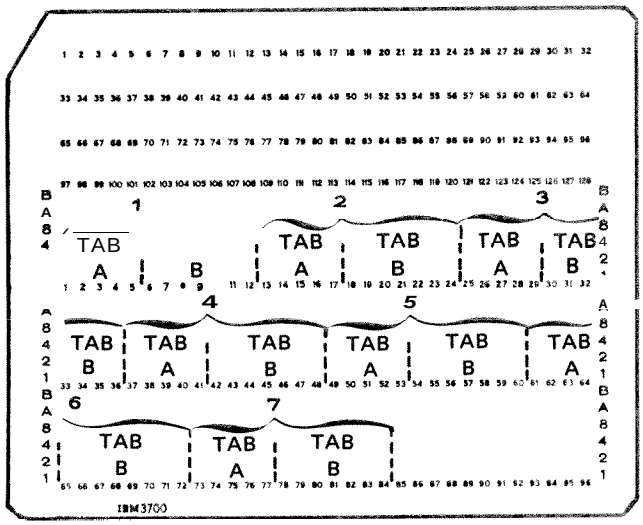
Array Name

Every array used in your program must be given a name. An array name cannot begin with the letters TAB. This array name is used throughout the program. See ***Arrays*** in Chapter 10 for more information.

Example

Figure 5-2, insert A, shows two related tables (TABA and TABB) described in alternating form on a table input card. An item for TABA appears first. Thus, in insert B, TABA is named in columns 27-32 of the Extension sheet: TABB is named in columns 46-51.





The corresponding items from the related tables are punched in alternating format on the table input card. The corresponding items from the two related tables are considered as one entry.

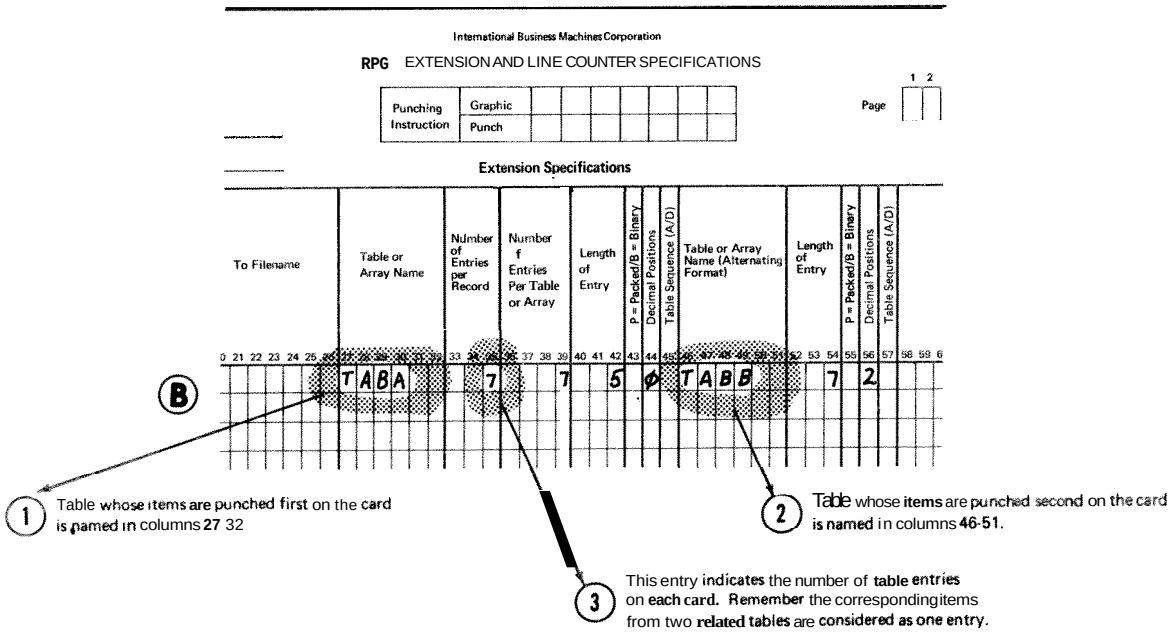


Figure 5-2. Related Tables

COLUMNS 33-35 (NUMBER OF ENTRIES PER RECORD)

Entry	***Explanation***
1-999	Number of table or array entries found in each table or array input record.

Indicate in columns ***33-35*** the exact number of table entries in each table or array input record. Every table or array input record except the last must contain the same number of entries as indicated in columns ***33-35.*** The last record may contain fewer entries than indicated, but never more.

When two related tables are described in one file, each table input record must contain the corresponding items from each table written in alternating form. These table items are considered as one entry (see ***Example).*** The number entered must end in column ***35.*** Corresponding items from related tables must be on the same record. If there is room, comments may be entered on table input record in columns following table entries.

When loading an array the following must be considered:

1. To load an array at execution time, a filename must be entered in columns ***11-18*** and an entry must be made in Number of Entries per record (columns ***33-35).***

2. To load an array at compile time, the filename entry (columns ***11-18)*** must be blank, but an entry must be made in Number of Entries per Record (columns ***33-35).***

3. To load an array via the input and/or calculations specifications, the filename (columns ***11-18)*** entry must be blank and the Number of Entries per Record (columns ***33-35)*** must be blank.

Example

Figure 5-2, insert B, shows table entries for the two related tables, A and B. A1 and B1, the corresponding items in tables A and B, are considered one entry. Even though there are 14 table items on the card, there are only *7* table entries.

COLUMNS 36-39 (NUMBER OF ENTRIES PER TABLE OR ARRAY)

Entry	*Explanation*
1-9999	Maximum number of table or array entries.

Use columns ***36-39*** to indicate the maximum number of table items which can be contained in the table named in columns ***27-32,*** or the maximum number of array items which can be contained in the array named in columns ***27-32.*** This number may apply to one table or to two related tables. Any number entered in these columns must end in column ***39.***

If your table or array is full, this entry gives the exact number of items in it. However, if the table or array is not full, the entry gives the number of items that can be put into it (Figure 5-3). A table that is not full is known as a short table.

TABPRT (Part Number) TABAMT (Price)

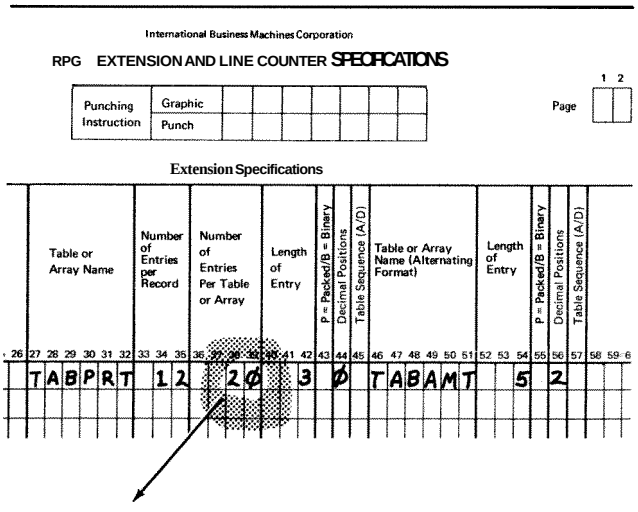
001	127.62
002	198.32
003	.27
004	.01
005	1.98
009	3.79
010	5.67
014	2.33
026	14.67
045	29.33
096	29.34
097	.05
098	.09
099	1.19
100	2.22
101	126.73
110	596.74
115	393.75
126	697.75
137	1.92

If this data is punched on table input cards and entered into the machine, TABPRT and TABAMT will be full (*20* entries fill the table).

TABPRT (Part Number) TABAMT (Price)

001	127.62
002	198.32
003	.27
004	.01
005	1.98

If this data is punched on table input cards and entered into the machine, TABPRT and TABAMT will not be full.



This entry indicates that TABPRT and TABAMT may both have a maximum of *20* entries.

Figure 5-3. Table Entries (Number per Table)

Since the number of table items for two related tables must be the same, the entry in these columns also gives the number of table items in a second table (columns 46-51).

If a table is to be compiled, it should be a full table. However, if it is not a full table (a short table), the table is compiled with the program and a warning is issued. The short table is completed in storage by filling in with blanks or zeros (for alphameric or numeric table, respectively). An execution time table need not be a full table. All execution time tables and arrays must be followed by a record with /* in columns 1 and 2 when they are loaded.

COLUMNS 40-42 (LENGTH OF ENTRY)

Entry	*Explanation*
1-999	Length of entry.

Use columns 40-42 to give the length of each entry in the table or array named in columns 27-32. The number entered must end in column 42. For numeric tables or arrays in packed decimal format, enter the unpacked decimal length in columns 40-42. For numeric tables or arrays in binary format, enter the number of bytes required in storage for the binary field. For a 2 character binary field, the entry in columns 40-42 is 4; for a 4 character binary field the entry is 9.

ALL table items must have the same number of characters. It is almost impossible, however, for every item to be the same length. Therefore, add zeros or blanks to the front of numeric items to make them the same length and add blanks to alphameric items. For alphameric items, blanks may be added either before or after the item (see *Examples, Example* 1).

If two related tables are described in one table file, the specification in columns 40-42 applies to the table whose item appears first on the record (see *Examples, Example* 2).

The maximum length of a numeric table item is 15 characters. The maximum length of an alphameric table item is 256 characters. However, the use of table or array input cards limits you to a length of 96 characters per table item. See *Arrays* in Chapter 10 for more information.

Examples

Example 1: Figure 5-4 shows a table, called TABMO, which lists the months of the year. The name SEPTEMBER, having nine characters, is the longest entry. Because the lengths of the entries must be the same, blanks are added to the remaining names to make each of them nine characters long.

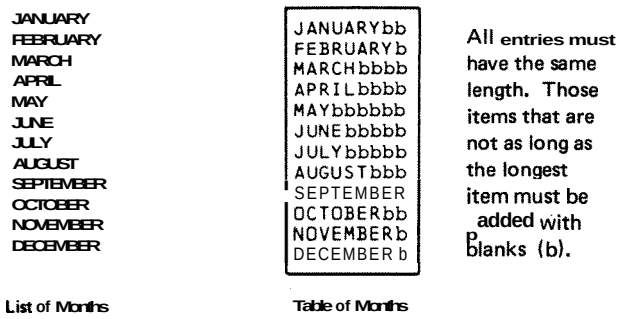
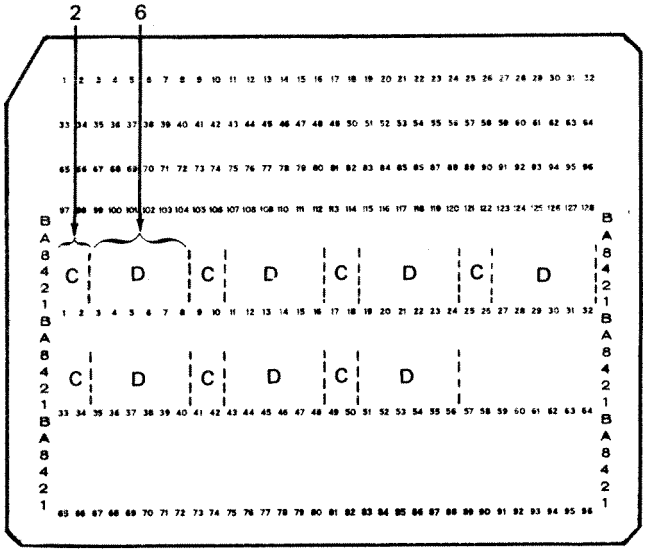


Figure 5-4. Length of Table Entries

Example 2: Figure 5-5 shows entries in a table input card for related tables, C and D. Each item in table C is two characters long; each item in table D is six characters long. Since table C appears first on the card, its length, **2,** is specified in columns 40-42. The length of items in table D is indicated in columns **52-54.**



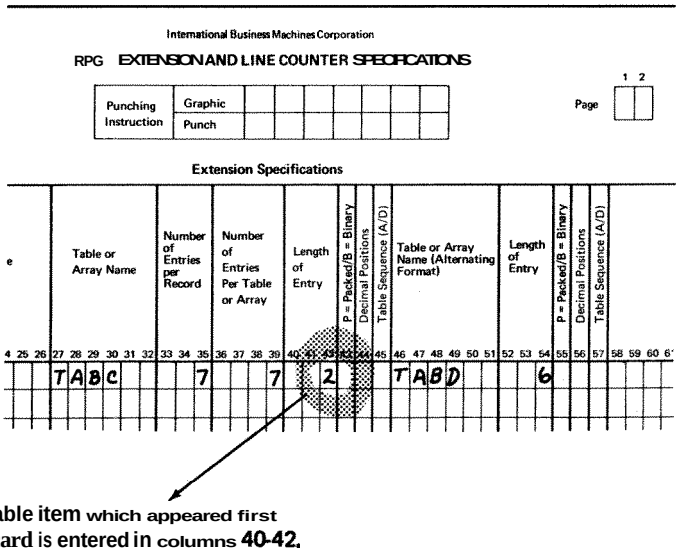


Figure 5-5. Length of Corresponding Table Items

COLUMN 43 (PACKED OR BINARY FIELD)

Entry	*Explanation*
Blank	Data for table or array is in unpacked decimal format or is alphameric.
P	Data for table or array is in packed decimal format.
B	Data for table or array is in binary format.

COLUMN 44 (DECIMAL POSITIONS)

Entry	*Explanation*
Blank	Alphameric table or array.
0-9	Number of positions to the right of the decimal in numeric table or array items.

Column 44 must always have an entry for a numeric table or array. If the items in a table or array have no decimal positions, enter a *0.*

If two related tables are described in one table file, the specification in this column applies to the table containing the item which appears first on the record.

COLUMN 45 (SEQUENCE)

Entry	*Explanation*
Blank	No particular order.
A	Ascending order.
D	Descending order.

Use column 45 to describe the sequence (either ascending or descending) of the data in a table or array file.

When an entry is made in column 45, the table or array is checked for the specified sequence. If a compile time table or array is out of sequence, a severe error occurs. The program will halt after compilation. If an execution time table or array is out of sequence, a severe error occurs and the program halts immediately,

Ascending order means that the table or array items are entered starting with the lowest data item (according to the collating sequence) and proceeding to the highest. Descending order means that the table or array items are entered starting with the highest data item and proceeding to the lowest.

If two related tables or two related arrays are described in one file, the entry in column 45 applies to the table or array containing the item which appears first on the record.

When you are searching a table or array for an item (LOKUP) and wish to know if the item is high or low compared with the search word, your table or array must be in either ascending or descending order. See *Operation* Codes, *Lookup* in Chapter *10* for more information. When a specific sequence has been specified, RPG II checks the data in the table or array to see if it really is in that sequence.

COLUMNS 46-57

Use columns **45-57** only when describing a second table or array which relates to and corresponds with the table or array named in columns **27-32.** All fields in this section have the same significance and require the same entries as the fields with corresponding titles in columns **27-45.** See the previous discussion on those columns for information about correct specifications.

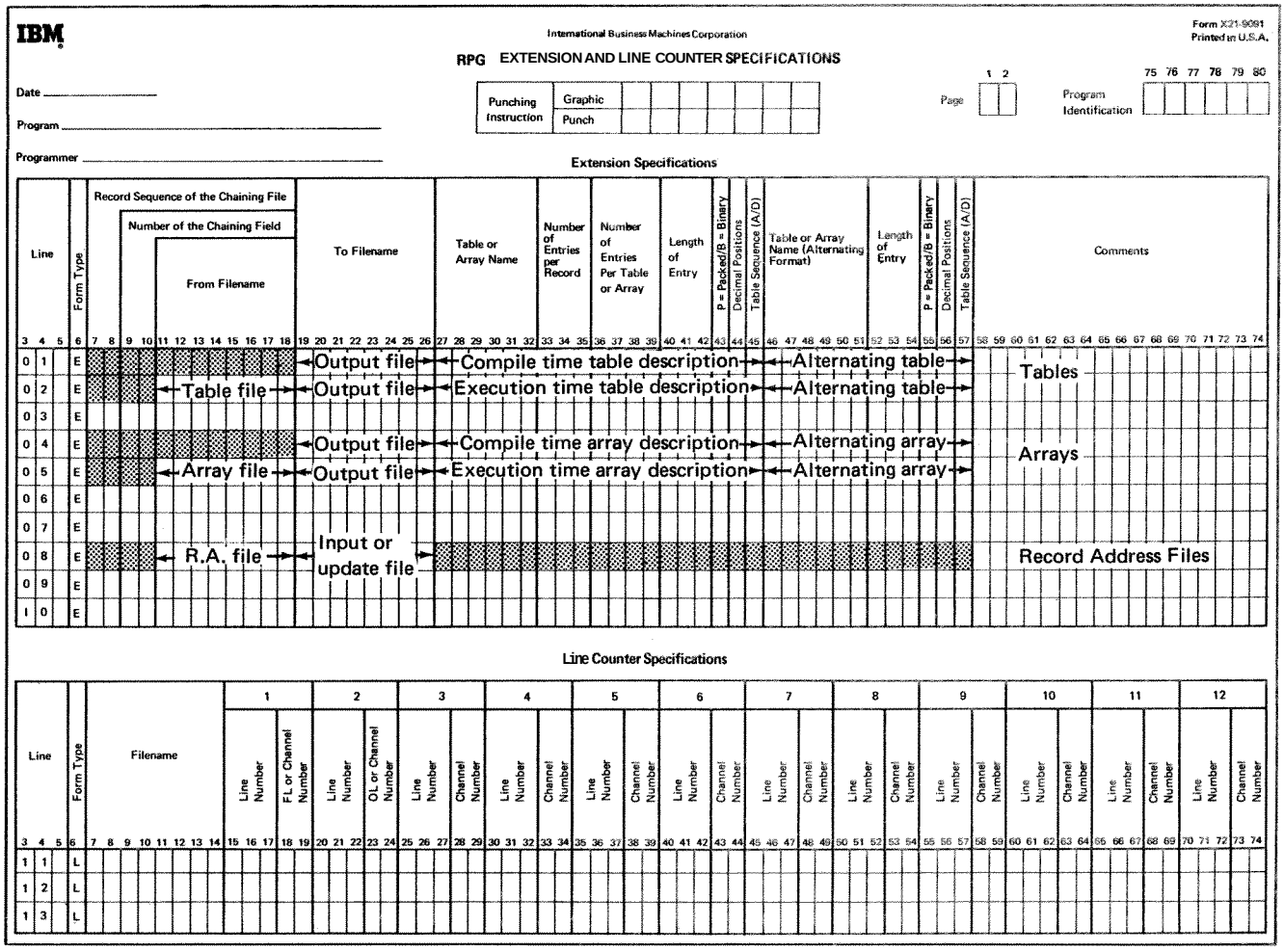
Leave these columns blank for a single table or array.

COLUMNS 58-74 (COMMENTS)

Enter any information you wish in columns 58-74. The comments you use should help you understand or remember what you are doing in each specification line. Comments are not instructions to the RPG II program; they serve only as a means of documenting your program.

COLUMNS 75-80 (PROGRAM IDENTIFICATION)

See Chapter **2.**



- **The shaded columns must be blank for the file named.**
- **For tables and arrays, columns 1426 and columns 46-57 are always optional.**
- **For record address files, columns 11-26 must have entries.**

Figure 5-6. Possible File Entries for Extension Specifications

Line counter specifications must be used for each printer file in your program. If the dual carriage feature is used, two specification lines must be completed. Line counter specifications indicate at what line printing is to begin and end on forms used in a printer. Write these specifications on the Extension and Line Counter sheet (Figure 6-1).

COLUMNS 1-2 (PAGE)

See Chapter 2.

COLUMNS 3 5 (LINE)

See Chapter 2.

COLUMN 6 (FORM TYPE)

An **L** must appear in column 6.

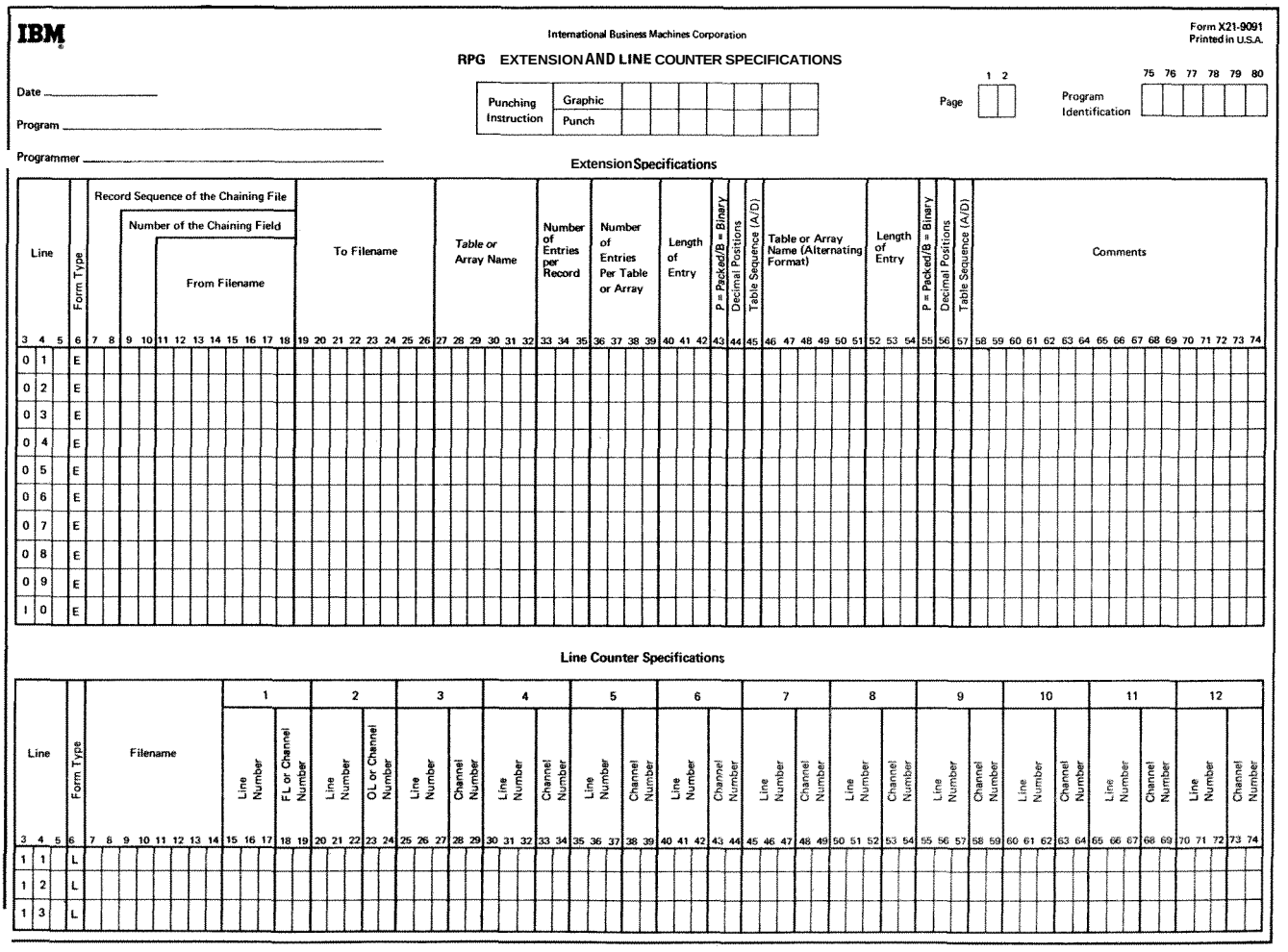


Figure 6-1. Extension and Line Counter Sheet

COLUMNS 7-14 (FILENAME)

Use columns 7-14 to identify the output file to be written on the printer. The filename must begin in column 7.

Any filename entered in these columns must be a filename previously defined on the File Description sheet. The output device assigned to the file on the File Description sheet must be a printer.

COLUMNS 15-17 (LINE NUMBER--NUMBER OF LINES PER PAGE)

Entry	***Explanation***
Blank	Number of printing lines available is *66.*
1-112	Number of printing lines available is from 1-112.

Columns 15-17 specify the exact number of lines available on the form or page to be used. The entry must end in column 17. Leading zeros are not necessary.

COLUMNS 18-19 (FORM LENGTH)

Entry	*Explanation*
FL	Form length

Columns 18-19 must contain the entry *FL.* This entry indicates that the preceding entry (columns 15-17) is the form length.

CQLUMNS 20-22 (LINE NUMBER)

Entry	***Explanation***
Blank	Line 60 is the overflow line.
1-112	**A** line number from 1-112 is the overflow line.

Columns 2822 specify the line number that is the overflow line. The entry must end in column 22. Leading zeros may be omitted.

When the line which you have specified as the overflow line is printed, the overflow indicator turns on to indicate that the end of the page is near. When the overflow indicator is on, the following occur before forms advance to the next page:

1. Detail lines are printed (if this part of the program cycle has not already been completed).
2. Total lines are printed.
3. Total lines conditioned by the overflow indicator are printed.

Because all these lines are printed on the page after the overflow line, you have to specify the overflow line high enough on the page to allow all these lines to print. You know the data you will be printing out after the overflow line is reached. Thus, you can judge what line should be the overflow line on this basis. See ***Overflow Indicators*** in Chapter 10 for more information.

COLUMNS 2324 (OVERFLOW LINE)

Entry	***Explanation***
OL	Overflow line

Columns 23-24 must contain the entry OL. This entry indicates that the preceding entry (columns 20-22) is **the** overflow line.

COLUMNS 2574

Columns 25-74 are not used. Leave them blank.

COLUMNS 7580 (PROGRAM IDENTIFICATION)

See Chapter 2.

Input specifications describe the data files, records, and fields of the records to be used by your program. These specifications may be divided into two categories:

1. File and record type identification (columns 7-42) describes the input record and its relationship to other records in the file,

2. Field description entries (columns 43-74) describe the fields in the records.

The specifications are written on the Input sheet (Figure 7-1). The field description entries must start one line lower than file and record type identification entries.

COLUMNS 1-2 (PAGE)

See Chapter 2.

COLUMNS 3-5 (LINE)

See Chapter 2,

COLUMN 6 (FORM TYPE)

An *I* must appear in column 6.

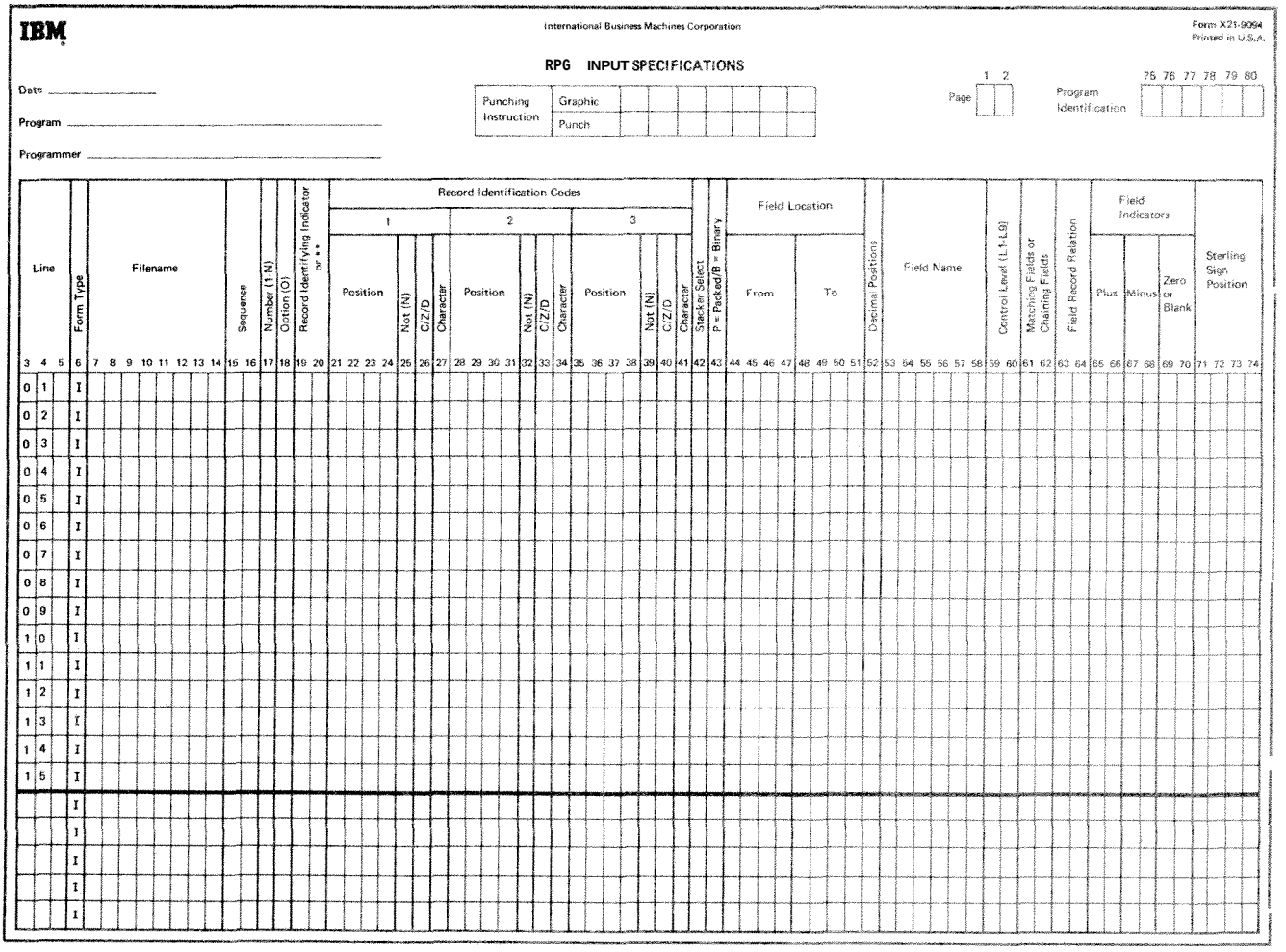


Figure 7-1. Input Sheet

COLUMNS 7-14 (FILENAME)

Columns 7-14 identify the input file you are describing. The input filename must begin in column 7. Use the same filename given in the file description specifications. The name of every input file described in the file description specifications must be entered at least once on this sheet. The filename must appear on the first line that contains information concerning the cards in that file. If the filesame is omitted. the last filename entered is assumed to be the file being described.

COLUMNS 15-16 (SEQUENCE)

Entry	*Explanation*
Any two alphabetic characters	Do not check for special sequence.
Any two-digit number	Check for special sequence.

Columns 15-26 may contain a numeric entry which assigns a special sequence to different record types in a file.

If different types of records do not need to be in any special order, use two alphabetic characters (see *Examples, Example 1*). Alphabetic characters must be used for chained files. Do not use alphabetic entries ND and R6 (Rblank) because the computer may mistake them for the ND or R in an AND or OR line. Within one file, all record types having alphabetic entries in columns 15-16 must be specified before those types with numeric entries.

Use columns 15-16 to assign sequence numbers to different types of records within a file. Your job may require that one record type (identified by a record identification code) must appear before another record type within a sequenced group. For instance, you may want a name record before an address record. You must provide a record identification code for each type of record and then number the record types in the order that they should appear. The program will check this order as the records are read, The first record type must have the lowest sequence number (01), the next record type should be given a higher number, etc. (See *Examples, Example* 2.)

Numeric sequence numbers only ensure that all records of record type 01 precede all records of record type 02, etc,, in any sequenced group. The sequence numbers do not ensure that records within a record type are in any certain order. Numeric sequence numbers have no relationship with control levels, nor do they provide for sequence checking of data in fields of a record (see *Examples, Example 3).*

Caps in sequence numbers are allowed, but the numbers used must be kept in ascending order, The first sequence number *must* be 01.

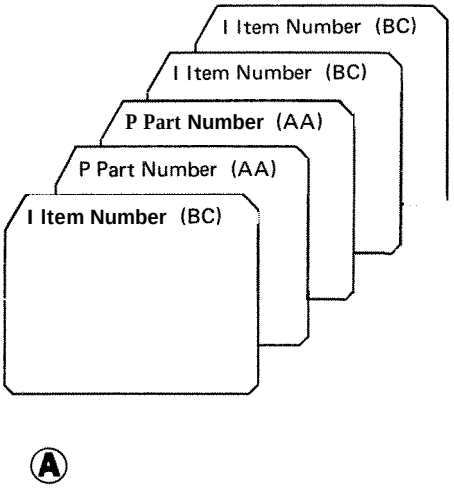
A record type out of sequence causes the program to stop. The program may be restarted by pressing the start key on the Processing Unit. The record that causes the halt is bypassed and the next record is read from the same file.

Records in an OR line cannot have a sequence entry in these columns. The entry in these columns from the previous line also applies to the card in the OR line. See *Columns 53-58* in this chapter for information on OR relationships.

Examples

Example 1: Figure 7-2, insert **A,** shows a file having two types of records (part number and item number) which may appear in any order. Since they are not to be checked for sequencing, they are assigned two alphabetic characters **(AA** and BC, respectively) instead of numbers. See *Figure* 7-2, *insert B,* for the coding of this example.

Example 2: Figure 7-3, insert **A** shows the order of four different types of records within a file. The records are arranged in groups according to some control field. The name record is first in each group and is assigned sequence number 01. Street record is next and is assigned 02. City/state record is 03. Item number is last and is assigned 07. (Remember gaps are allowed.) See Figure 7-3, insert B for the coding of this example.



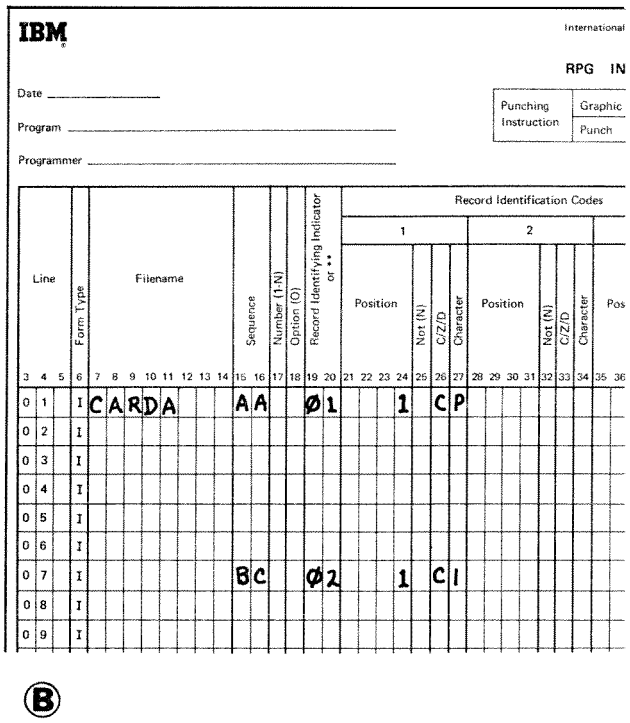
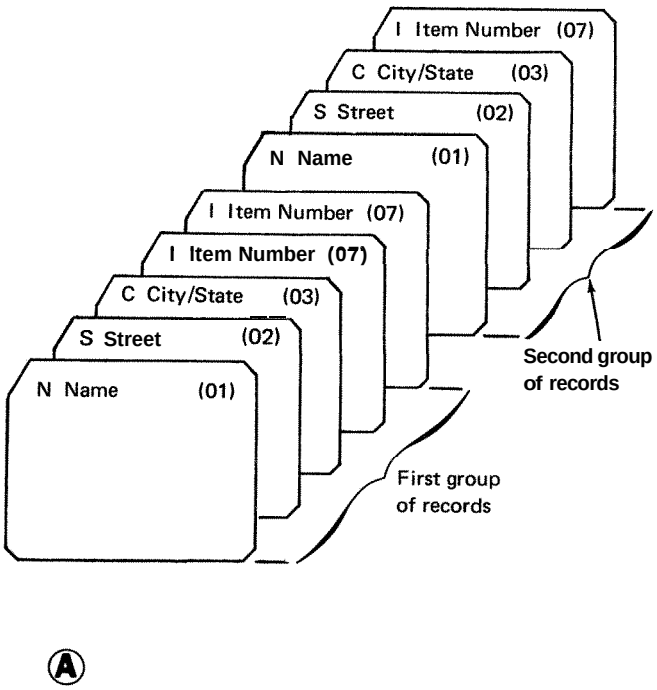


Figure 7-2. Unsequenced Card Types in a File



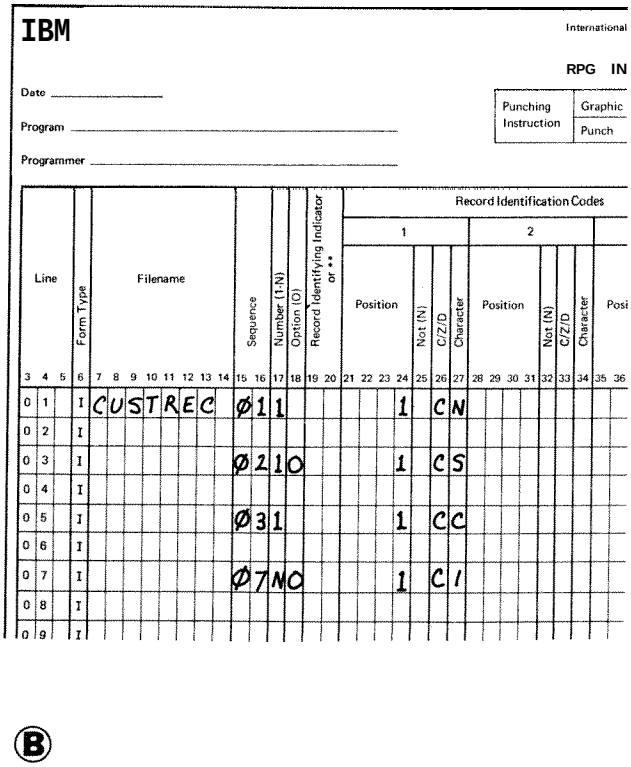


Figure 7-3. Sequence Checking of Record Types

Example 3: Figure *7-4* shows three groups of four different record types. Each group is in proper sequence according to the assigned sequence numbers (01, 02, 03 and 07). Notice, however, that the city/state record for group B is in group C and vice versa, The sequence entry which you specify in columns 15-16 will not catch this mistake since the sequence entry does not cause the data on the record to be checked.

COLUMN 17 (NUMBER)

Entry	*Explanation*
Blank	Record types are not being sequence checked (columns 15-16 have alphabetic entries).
1	Only one record of this type is present in the sequenced group.
N	One or more records of this type may be present in the sequenced group.

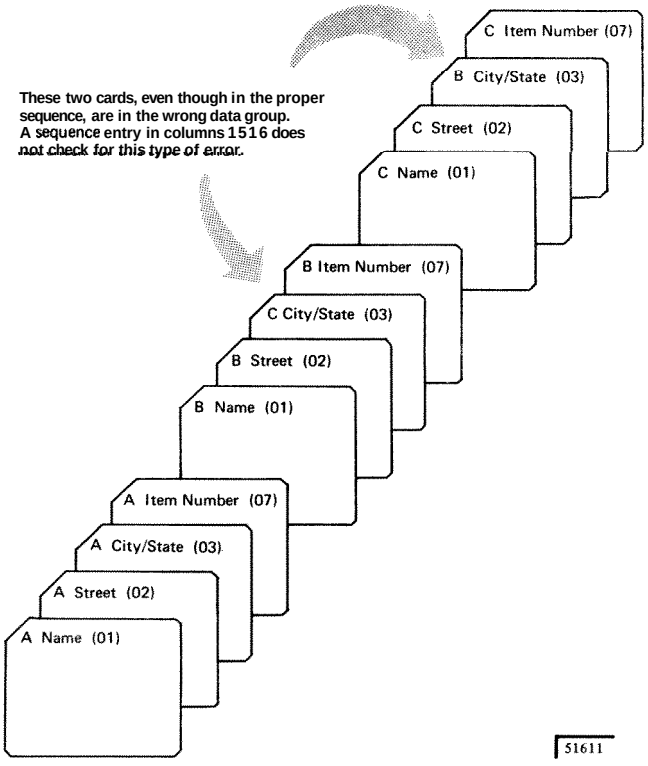


Figure 7-4. Correct Card Sequence (Incorrect Data in Each Group)

Use column 17 only if sequence checking is to be done (columns 15-16 contain numbers). Often, when sequence checking, you may have more than one record of a particular type within the sequenced group (see ***Example).*** Thus you must indicate by an entry in column 17 that a certain number of records of one type may be found in the sequence group.

OR lines (columns 14-15 have the letters OR) should not have an entry in this column. It is assumed that the number of records of this type to be found in the sequenced group is the same as the number entered in column 17 of the previous line. See *Columns 53-58* in this chapter for more information on OR lines.

Example

Figure 7-5 shows a sequencedrecord file in which there is more than one record per type in a group. The record type called item number appears three times.

There is no reason for a name, street, or city/state record to appear more than once in one group. A *1* is entered in column 17 to indicate that these record types appear only once in each group. However, since one person may have purchased more than one item, there may be two or more item number records per group; an *N* is entered in column 17 for this field. See ***Figure*** 7-3, ***insert B,*** for the coding of this example.

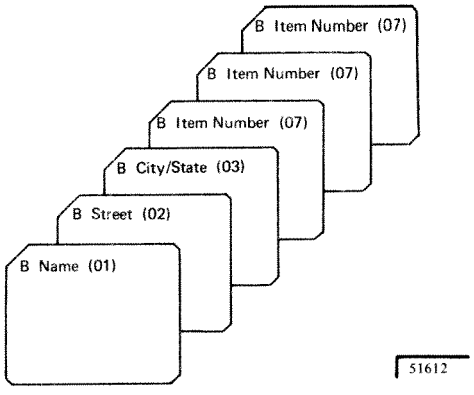


Figure 7-5. **Sequenced** Card **File (More than One Record per Type in a Group)**

COLUMN 18 (OPTION)

Entry	*Explanation*
Blank	Record type must be present (if sequence checking is specified).
O	Option. Record type may or may not be present.

Column 18 is used when record types are being sequence checked. A blank entry specifies that a record of this record type must be present in each sequenced group.

The O entry specifies that a record of this record type may or may not be present in each sequenced group (see *Example*). If all record types are optional, no sequence errors will be found.

OR lines should not have an entry in this column. The entry in this column on the previous line also applies to this record in the OR relationship. See *Columns 53-58* in this chapter for more information on OR lines.

Example

Figure 7-6 shows a sequenced card file in which a card type may be optional. For instance, the street or item number records may not be included. Since it is not always necessary to have a street address, this record is optional. Suppose this job required a list of all items purchased during one month by the individual named in the name record. It is possible that a person might not buy anything during the month. In this case, there would be no item record; therefore, the item record would also be optional. (See *Figure 7-3, insert B* for a coding example.)

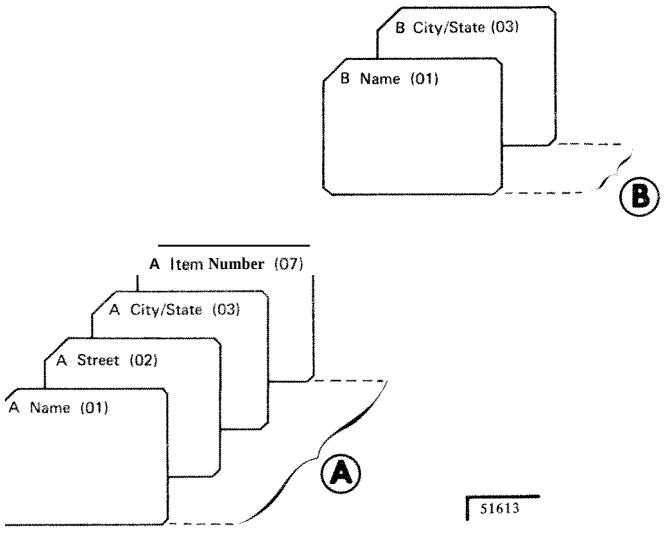


Figure 7-6. Sequenced Card File (Optional Record Types)

COLUMNS 19-20 (RECORD IDENTIFYING INDICATOR, **)

Entry	*Explanation*
01-99	Record identifying indicator.
L1-L9	Control level indicator, used for a record identifying indicator when a record type rather than a control field signals the start of a new control group.
LR	Last record indicator.
H1-H9	Halt indicator, used for a record identifying indicator when checking for a record type that causes an error condition.
**	Look ahead field,

Columns 19-20 may be used for two purposes:

1. to specify record identifying indicators, or
2. to indicate a look ahead field.

Record Identifying Indicators

Use columns 19-20 to assign an indicator to each record type. When you have different types of records within a file, you often want to do different operations for each record type. Therefore, you must have some way of knowing which type of record has just been read. To do this, you assign different record identifying indicators to each record type. Whenever a record type is selected to be processed next, Its corresponding identifying indicator is turned on (all other record identifying indicators are off at this time). This indicator signals throughout the rest of the program cycle which record type has just been selected.

Because the record identifying indicator is on for the rest of the program cycle, you may use it to condition calculation operations (see *Columns 9-17* in Chapter 8) and output operations (see *Columns 23-31* in Chapter *9).*

Record identifying indicators do not have to be assigned in any order.

You may assign the same indicator to two or more different record types provided you want the same operations performed on these types. Do this by using the OR relationship (see *Columns 21-41* in this chapter)

No record identifying indicator may be specified in the AND line of an AND relationship, Resulting indicators for OR lines may be specified for every record type in the OR relationship that requires special processing. See *Columns 21-41* in this chapter for information on AND lines, See *Columns 53-58* in this chapter for information on OR lines.

Look Ahead Fields

Use asterisks in columns 39-70 to indicate that fields named in columns 53-55 in following specifications are look ahead fields. *A* look ahead field allows you to look at information in a field on the next record that is available for processing in any input, update, or combined file. Because of this capability you are able to use the information from the look ahead field to determine what operations should be done next.

Through the use of a look ahead field, you are able to:

1. Determine when you are processing the last card of a control group.

2. Do jobs which the RPG II marching record capability cannot do.

See Look *Ahead* in Chapter 10 for information on when and how to use the look ahead fields.

COLUMNS 21-41 (RECORD IDENTIFICATION CODES)

Use columns 21-41 to describe the information that identifies a record type.

When you have many record types in one file, you often want to perform different operations for each type. Therefore, you must identify each type by giving each a special code consisting of a combination of characters in certain positions in the record. This code must be described in columns 21-41 so that when a record is read the record type can be determined by these specifications.

When more than one record type is used in a file, only one record type will be selected for processing in each cycle. The record identifying indicator for that record type will be turned on at the time of selection. When all records are to be processed alike regardless of their type, or if there is only one type, leave columns 21-41 blank.

AND Relationship

A maximum of three identifying characters may be described in one specification line. Thus, if the identification code consists of more than three characters, an AND line must be used. This means that the first three identifying characters are described in the first line. The additional identifying characters are described in as many following lines as are needed. Write the word AND in columns 14-16 to indicate an AND line (see *Examples, Example 1*).

You may specify as many AND lines as you need in order to describe the record identifying code. The record must contain all the characters indicated as its record identification code before the record identifying indicator will turn on.

OR Relationship

A particular record type may be identified by two different codes. If this is the case, OR lines must be used to indicate that either one of the codes may be present to identify the record. A maximum of twenty OR lines may appear for each record sequence. Write the word OR in columns 14-15 to indicate an OR line (see *Examples, Example* 2).

Seven columns are set aside for the description of one character in the record identification code. Each specification line contains three sets of seven columns: columns 21-27, 28-34, and 35-41. Each set consists of 4 fields: Position, Not, C/Z/D, and Character. Coding is the same for all three sets.

Position

Entry	*Explanation*
Blank	No record identification code is needed.
1-4096	Record position of one character in the record identification code.

Use columns 21-24, 28-31, and 35-38 to give the location in the record of every character in the identification code. Entries in these columns must end in columns 24, 31, and 38 respectively.

Not (N)

Entry	*Explanation*
Blank	Character is present in the specified column.
N	Character is not present in the specified column.

Use columns 25, 32, and 39 to indicate that a certain character should not be present in the specified positicin.

C/Z/D

Entry	*Explanation*
C	Entire character.
Z	Zone portion of character.
D	Digit portion of character.

Use columns 26, 33, and 40 to indicate what portion of a character is used as part of the record identifying code. Only the zone portion, only the digit portion, or both portions (the whole character) may be used (see *Examples, Example 3, and Example 4).* When establishing record identifying codes, remember that many characters have either ths same zone or the same digit portion. For a list of characters that have identical zone or digit portions see *Character Structure* in Chapter 10.

Character

Use any alphabetic character, special character, or digit in columns 27, 34, and 41 to identify the character that was used in the record to serve as the code or part of the code.

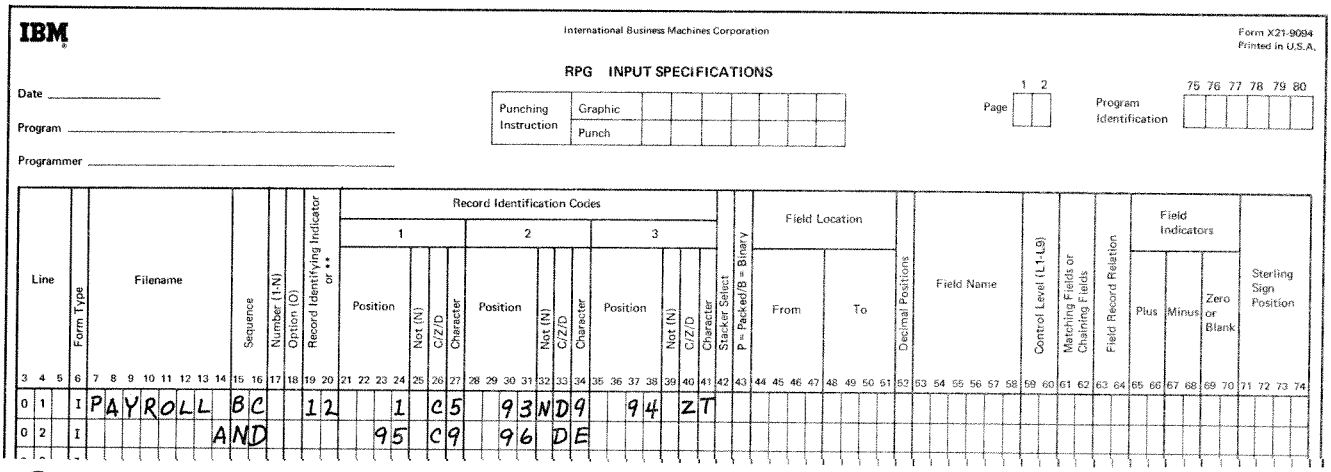
Note: If none of the identifying codes you have specified is found on a record, processing stops. You may continue, however, by pressing the start key on the Processing Unit. The record that caused the halt is not processed, and the next record in that file is read.

Examples

Example 1: Figure 7-7, insert A shows a record identification code consisting of five characters. The first character is located in position 1, the other four in positions 93, 94, 95 and 96. Since only three identifying characters may be described on one line, the word AND must be used on the next line to indicate that the last two characters of the code are part of the preceding record identification entries.

Example 2: Figure 7-7, insert B, shows the use of an OR line to describe record type identification codes. The record assigned resulting indicator 12 can be identified by two different codes. The record can be identified by a code consisting of a 5 in position 1 and a 6 in position 2 or a code consisting of a 6 in position 1.

Example 3: In Figure 7-7, insert A, the entry in column 32 indicates that the digit 9 must not be present in position 93 for record type 12.



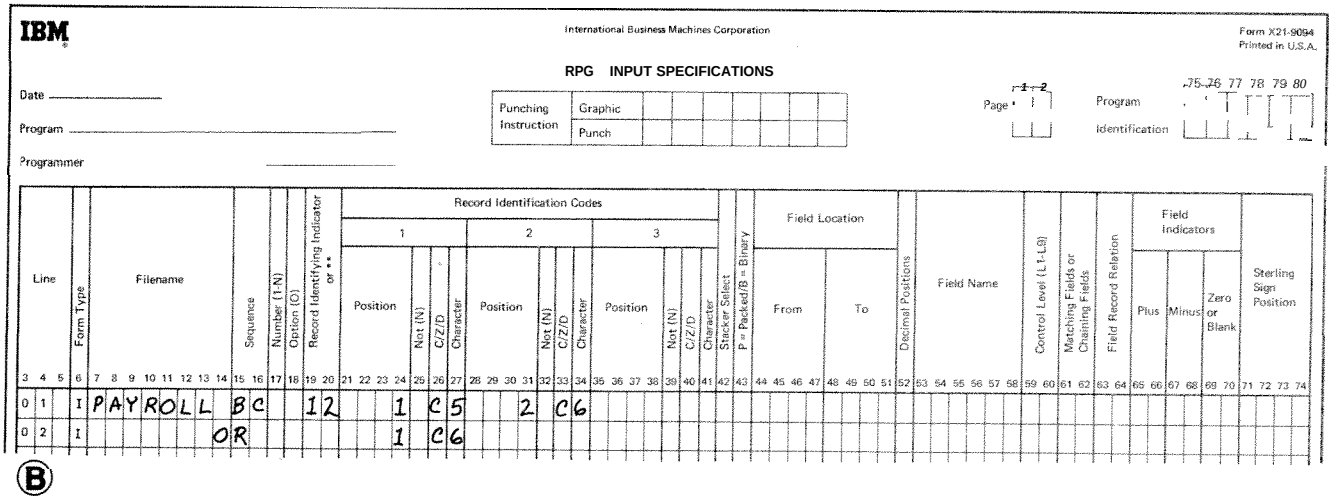


Figure 7-7. Record Identification Codes

Example 4: Figure 7-7, insert **A,** shows that only the zone portion of the character *T* located in position 94 is part of the identifying code. In position 96 only the digit portion of the character E is part of the code.

COLUMN 42 (STACKER SELECT)

Entry	*Explanation*
Blank	Cards automatically fall into a predetermined stacker.
1-4	Stacker into which the card type is stacked.

Column 42 is used to indicate that certain types of input cards must be stacked in a specific stacker. If you make no entry, all cards will go into a predetermined stacker (primary hopper--stacker I, secondary hopper--stacker 4). Only input file and combined file cards may be stacker selected in the input specifications.

You may stacker select cards from the input file in input specifications only. However, cards from a combined file may be stacker selected in either input specifications or output-format specifications (see *Column* 16 in Chapter 9).

Any card type that is stacker selected on the input specifications should not have an output operation specified for it. If an output operation is specified, however, the input stacker selection specification is overridden (see *Column 16* in Chapter 9) if the output is performed.

When the same stacker is indicated for both input and output files, a card from the output file is put in the stacker before a card from the input file. This procedure is reversed (input card before output card) if Look Ahead Fields or dual I/O areas are specified for the input file.

The card type in an OR line may be selected for a special stacker by an entry in column 42. If the card type in an OR line has no entry in column 42, the card goes into the predetermined stacker. (See *Column 53-58* in this chapter for more information on OR lines.) AND lines may not have an entry in stacker select.

COLUMN 43 (PACKED OR BINARY FIELD)

Entry	*Explanation*
Blank	Field is in unpacked decimal format or is alphameric.
P	Field is in packed decimal format.
B	Field is in binary format.

Column 43 is used to indicate that an input field is in packed decimal or binary format. Packed decimal and binary fields are converted into unpacked decimal format for use by the system. The conversion ignores decimal points.

Column 43 must contain a P if the input field named in columns 53-58 is in packed decimal format.

Column 43 must contain a *B* if the input field named in columns 53-58 is in binary format. Binary fields can only be read in from disk. The binary input field can only be 2 or 4 characters in length. The highest decimal number that can be expressed with 2 binary characters is 32,767; thus, 4 bytes of storage are set aside for a two-character binary field. The highest decimal number that can be expressed with 4 binary characters is 2,197,483,608; thus, 9 bytes of storage are set aside for a four-character binary field.

Note: Column 43 begins the field description entries (columns 43-74) which must begin one line below the file and record identification entries (columns 7-42) for each file.

COLUMNS 44-51 (FIELD LOCATION)

Entry	*Explanation*
Two 1-4 digit numbers	Beginning of a field (From) and end of a field (To).

Use columns 44-51 (From and To) to describe the location on the record of each field containing input data named in columns 53-58 (Field Name). Enter the number of the record position in which the field begins in columns 44-47. Enter the number of the record position in which the field ends in columns 48-51.

A single position field is defined by putting the same number in both From (columns 44-47) and To (columns 48-51). If a field of more than one position is defined, the number entered in From (columns 44-47) must be smaller than the number entered in To (columns 48-51).

The maximum field length for a numeric field is 15 digits. The maximum field length for an alphameric field is 256 characters.

Entries in these columns must end in columns 47 and 51. Leading zeros may be omitted.

COLUMN 52 (DECIMAL POSITION)

Entry	*Explanation*
Blank	Alphameric field.
0-9	Number of decimal positions in numeric field.

Use column 52 to indicate the number of positions to the right of the decimal in any numeric field named in columns 53-58. Column 52 must always have an entry when the field named in columns 53-58 is numeric. If you wish to define a field as numeric with no decimal position, enter a *0*. If a field is to be used in arithmetic operations or is to be edited, it must be numeric. If the number of decimal positions specified for a field exceeds the length of that field, the number of decimal positions is assumed equal to the length of the field.

COLUMNS 53-58 (FIELD NAME)

Entry	*Explanation*
1-6 alphameric characters	Field name, array name, or array element.
PAGE	
PAGE1	Special words
PAGE2	

Use columns 53-58 to name a field, array, or array element found on your input records. If you are referencing an array, additional entries may be needed in these columns (see *Arrays* in Chapter 10). Use this name throughout the program whenever you refer to this field. You must indicate the names of the fields for all types of records. However, you should name only the fields that you will use.

A field name can be from 1-6 characters long, and must begin in column 53. The first character must be an alphabetic character. The remaining characters can be any combination of alphabetic and numeric characters (special characters are not allowed). Blanks may not appear between characters in the name.

All fields in one type of record should have different names. If two or more fields on the same record type have the same name, only the field described last is used. However, fields from different record types may have the same name if the fields are the same length and contain the same type of data. This applies even if the fields are found in different locations in each record type. Duplicate field names should not be used if matching fields are specified in your program.

Numeric fields may have a maximum length of 15 characters. Alphameric fields may have a maximum length of 256 characters. However, fields which are read in from a card are limited to the length of one punched card.

Fields that are used in arithmetic operations (see *Operation Codes* in Chapter 10) or fields that are edited or zero suppressed (see *Editing* in Chapter 10) must be defined as numeric. This means that column 52 must have a decimal position entry.

A separate line is used for each field description.

OR Relationship: Even though two or more record types contain identical fields, you must describe each field. This may require duplicate coding. To eliminate duplicate coding of identical fields from different record types, you may use the OR relationship. A maximum of twenty OR lines may be used for each record sequence group.

An OR relationship means that the fields named may be found in either one of the record types. You may use OR lines when:

1. Two or more record types have the same fields in the same positions (see *Example).*
2. Two or more record types have some fields which are identical and some fields which differ in location, length, or type of data. See *Columns* 63-64 in this chapter for sample coding of such record types.

Write the word OR in columns 14 and 15 to indicate an OR line (see *Example).* If there are several AND or OR lines, field description lines start after the last record identification line.

PAGE: If your printed report has several pages, you may want to number the pages. The special word PAGE allows you to indicate that page numbering is to be done. When you use a PAGE entry on the Output-Format sheet, page numbering automatically starts with 1 (see *Columns 32-37* in Chapter 9).

If you want to start at a page number other than 1, you can enter that page number in a field of an input record and name that field PAGE in columns 53-58. The number you enter in the PAGE field of the input record should be one number less than the starting page number. If your numbering should start with 24, enter a 23 in the PAGE field. The PAGE field can be of any length, but cannot have decimal positions (Figure 7-8). Any entry you make in the PAGE field should be right justified, such as 0023.

Page numbering can be restarted during a program run by entering a number in a PAGE field of any input record. The PAGE field can be defined and used in calculations like any other field.

The three possible PAGE entries: PAGE, PAGE1, and PAGE2, are provided for naming different output files. Do not use the same entry for two different output files.

Example

Figure 7-9 shows how the use of OR lines can save duplicate coding. The two different record types (one identified by a 5 in column 1, the other by a 6 in column 1) both have identical fields which must be described. Figure 7-9, insert A, shows one way of doing this. Figure 7-9, insert B, shows the use of OR lines to do the same thing with less coding. The coding in Figure 7-9, insert B, says that all four fields can be found on either the record type identified by the 5 in column 1 or the record type with a 6 in column 1.

COLUMNS 59-60 (CONTROL LEVEL)

Entry	*Explanation*
L1-L9	Any control level indicator.

Use columns 59-60 to assign control level indicators to input fields. (Control level indicators may not be associated with a chained file.) Control level indicators are used to specify the point at which specified operations are to be done. You may assign a control level indicator to any field; this field is then known as a control field. It is checked for a change in information. When information in the control field changes, a control break occurs. All records having the same information in the control field are known as a control group.

Whenever a record containing a control field is read, the data in the control field is compared with data in the same control field from the previous record. When a control break occurs, the control level indicator turns on. Operations conditioned by the control level indicator are then done (see *Columns* 9-17 in Chapter 8 or *Columns 23-31* in Chapter 9.)

There are nine different control levels (L1-L9). When a certain control level indicator turns on, all control level indicators lower than it also turn on. For example, if control level indicator **3** turns on, control level indicators 1 and 2 also turn on.

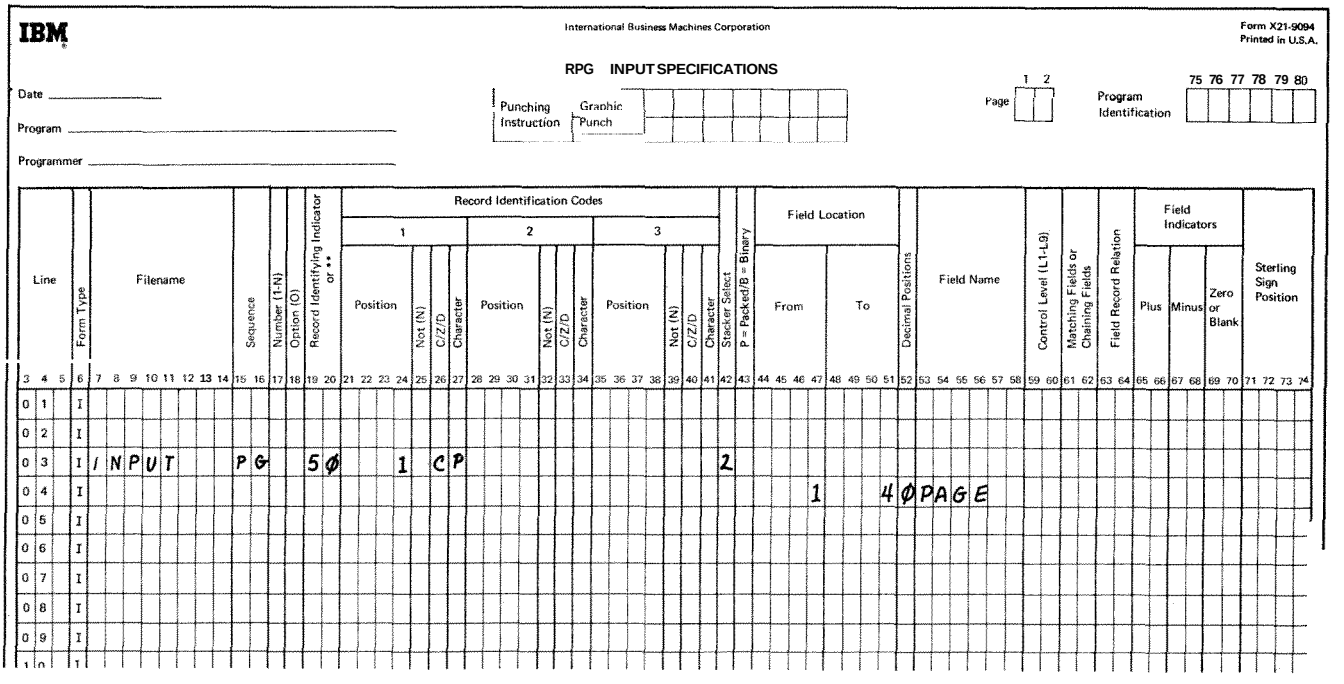


Figure 7-8. **Page** Record Description

IBM

International Business Machines Corporation

Form X21-9094
Printed in U.S.A.

RPG INPUT SPECIFICATIONS

Date _____

Program _____

Programmer _____

Punching Instruction: Graphic / Punch

Page 1 2

Program Identification 75 76 77 78 79 80

Line	Form Type	Filename	Sequence	Number (1-N)	Option (O)	Record Identifying Indicator or **	Record Identification Codes 1: Position	1: Not (N)	1: C/Z/D	1: Character	2: Position	2: Not (N)	2: C/Z/D	2: Character	3: Position	3: Not (N)	3: C/Z/D	3: Character	Stacker Select	P = Packed/B = Binary	Field Location: From	Field Location: To	Decimal Positions	Field Name	Control Level (L1-L9)	Matching Fields or Chaining Fields	Field Record Relation	Field Indicators: Plus	Field Indicators: Minus	Field Indicators: Zero or Blank	Sterling Sign Position
01	I	SALES	AA			14	1		C5																						
02	I																				5	8		DEPT							
03	I																				9	14		EMPNO							
04	I																				46	50		ITEM							
05	I																				66	70		COST							
06	I		BB			15	1		C6																						
07	I																				5	8		DEPT							
08	I																				9	14		EMPNO							
09	I																				46	50		ITEM							
10	I																				66	70		COST							
11	I																														
12	I																														

(A)

IBM

International Business Machines Corporation

Form X21-9094
Printed in U.S.A.

RPG INPUT SPECIFICATIONS

Date _____

Program _____

Programmer _____

Punching Instruction: Graphic / Punch

Page 1 2

Program Identification 75 76 77 78 79 80

Line	Form Type	Filename	Sequence	Number (1-N)	Option (O)	Record Identifying Indicator or **	Record Identification Codes 1: Position	1: Not (N)	1: C/Z/D	1: Character	2: Position	2: Not (N)	2: C/Z/D	2: Character	3: Position	3: Not (N)	3: C/Z/D	3: Character	Stacker Select	P = Packed/B = Binary	Field Location: From	Field Location: To	Decimal Positions	Field Name	Control Level (L1-L9)	Matching Fields or Chaining Fields	Field Record Relation	Field Indicators: Plus	Field Indicators: Minus	Field Indicators: Zero or Blank	Sterling Sign Position
01	I	SALES	AA			14	1		C5																						
02	I	OR				15	1		C6																						
03	I																				5	8		DEPT							
04	I																				9	14		EMPNO							
05	I																				46	50		ITEM							
06	I																				66	70		COST							
07	I																														
08	I																														
09	I																														
10	I																														
11	I																														
12	I																														

(B)

Figure 7-9. Record Types with Identical Fields

The indicators are ranked in order of importance. The larger numbers rank higher than lower numbers. L4 has a higher rank than L1. The importance of a control field in relation to others should determine how you assign indicators. For example, the type of data which demands a subtotal has a lower control level indicator than data which needs a grand total. A field containing department numbers is given a higher control level indicator than a field containing employee numbers. (See *Examples, Example 1*).

Control level indicator L0, since it is always on, cannot be assigned to a control field. Nevertheless, you may use it to condition operations (see *Columns 7-8* in Chapter 8).

Normally control level indicators are used to:

1. Condition certain calculations to be performed when the information in the control field changes.

2. Condition certain punching (summary punching) or printing (total printing) to be done after totals have been found for one control group.

3. Condition certain operations to be done on the record that causes a change in a control field (first record of a new control group).

When assigning control level indicators, remember:

1. If the same control level indicator is used in different record types or in different files, the control fields associated with that control level indicator must be the same length and same type (alphabetic or numeric). See *Examples, Example 2.*

2. In the same record type, record columns in control fields assigned different control level indicators may overlap (Figure 7-10). However, the total number of columns assigned as control fields (counting each control level only once) must not be greater than 144. In Figure 7-10 for example, a total of 15 columns is assigned to control levels.

3. Field names are ignored in control level operations. Therefore, fields from different record types which have been assigned the same control level indicator may have the same name.

4. Control levels need not be written in any sequence. L2 entry can appear before L1. Also, there may be gaps in the control levels assigned.

5. When numeric control fields with decimal positions are compared to see if a control break has occurred, they are always treated as if they have no decimal positions.

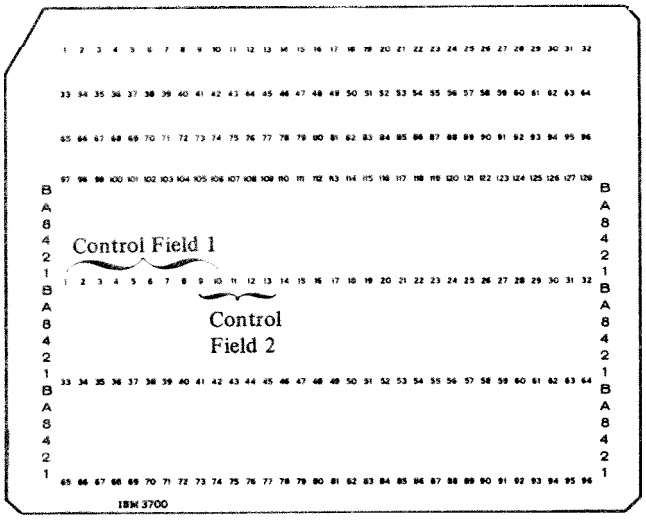


Figure 7-10. Overlapping Control Fields

6 If a field is specified as numeric, only the digit portion is used to determine if a control break has occurred. This means that a field is always considered to be positive. **A** -5 is considered equal to a +5.

7. All control fields given the same control level indicator are considered numeric if any one of those control fields is described as numeric (column *52* has an entry). This means that when numeric control fields are compared to see if the information has changed, only the digit portion of each character is compared.

8. Control fields are initialized to hexadecimal zeros.

9. **A** control break is highly probable after the first record containing a control field is read. The control fields in this record are compared to an area in storage which is void of any type of data. Since fields from two different records are not being compared, totai calculations and total output operations are bypassed for this cycle. **A** control break does occur, then, but it is not considered to be a true control break.

   Total calculations and total output operations are bypassed until the first cycle following a cycle involving a record with control fields specified.

Split Control Fields

If a control field is made up of more than one field of a record, it is then known as a split control field. A split control field is created when the same indicator is assigned to two or more connected or unconnected fields on the same record type.

All fields in one record that have the same control level indicators are combined by the program in the order specified in the input specifications and treated as one control field (see *Examples, Example 3).* Some special rules for split control fields are:

1. For one control level indicator, a field may be split in some record types and not in others if the field names are different. However, the length of the field, whether split or not, must be the same in all record types.

2. The length of the portions of a split control field may vary for different record types if the field names are different. However, the total length of the portions must always be the same.

3. No other specification lines may come between lines which describe split control fields.

4. If one section of a split control field is numeric, the whole field is considered numeric.

5. A numeric split control field may have more than 15 characters if any one portion of the split field does not exceed 15 characters and the sum of all control fields (counting each control level only once) is not greater than 144 characters.

6. A split control field cannot be made up of a packed decimal field and an unpacked decimal field. Both portions of the control field must be packed, or both unpacked.

Note: Additional rules applying to control level indicators when used with indicators in the Field Record Relation columns are discussed in *Columns* 63-64 in this chapter.

Examples

Example 1: Figure 7-11 shows the assignment of three indicators. The names of the control fields (DIVSON, DEPT, EMPLNO) give an indication of their relative importance. The division (DIVSON) is the most important group. It is given the highest control level indicator used (L3). The department (DEPT) ranks below the corporation; L2 is assigned to it. The employee field has the lowest control level indicator (L1) assigned.

IBM

International Business Machines Corporation

Form X21-9094
Printed in U.S.A.

RPG INPUT SPECIFICATIONS

Date _____
Program _____
Programmer _____

Punching Instruction: Graphic / Punch

Page 1 2

Program Identification 75 76 77 78 79 80

Line	Form Type	Filename	Sequence	Number (1-N)	Option (O)	Record Identifying Indicator or **	Record Identification Codes 1: Position	1: Not (N)	1: C/Z/D	1: Character	2: Position	2: Not (N)	2: C/Z/D	2: Character	3: Position	3: Not (N)	3: C/Z/D	3: Character	Stacker Select	P = Packed/B = Binary	Field Location: From	Field Location: To	Decimal Positions	Field Name	Control Level (L1-L9)	Matching Fields or Chaining Fields	Field Record Relation	Field Indicators: Plus	Field Indicators: Minus	Field Indicators: Zero or Blank	Sterling Sign Position
01	I	EMPLREP	AB			10	1		CA										2												
02	I																				5	10		ØEMPLNO	L1						
03	I																				11	30		NAME							
04	I																				80	80		DIVSON	L3						
05	I																				33	33		SHIFT							
06	I																				2	5		DEPT	L2						
07	I		CD			20	1		NCA										3												
08	I																				5	10		ØEMPLNO	L1						
09	I																				11	14		DEPT	L2						
10	I																				60	70		HRSWKD							
11	I																				80	80		DIVSON	L3						
12	I																														

Figure 7-11. Control Level Indicators (Two Record Types)

Example 2: Figure 7-11 shows that the same control level indicators may be used for different record types. Notice, however, that the control fields having the same indicators are the same length. EMPLNO, in both cases, is 6 columns in length, DEPT is 4, and DIVSON is one.

Example 3: Figure 7-12 shows a split control field made up of three portions. The control level indicator (L4) which is used for all three portions indicates that they are all to be treated as one control field. The field can be pictured as follows:

CUSNO	ACCTNO	REGNO
1 6 7 9	8 6 5 3 9 7	1 1 1

The first field assigned the same control level indicator begins the control field; the last ends it.

COLUMNS 61-62 (MATCHING FIELDS)

Entry	*Explanation*
M1-M9	Any matching level

Use columns 61-62 to specify matching fields and sequence checking. Matching fields and sequence checking cannot be specified for chained files.

An entry in columns 61-62 indicates:

1. Matching fields and sequence checking when you have two or more input or combined files with match fields.
2. Sequence checking only when you have just one input or combined file.

Matching Fields

Make an entry in columns 61-62 when you wish to compare records from two or more input or combined files in order to determine when records match. Records can be matched by matching one field, many fields, or entire records. You can indicate as many as nine matching fields (M1-M9). Whenever the contents of the match fields from records of the primary file are the same as the contents of the match fields from a secondary file, the matching record (MR) indicator turns on. M1-M9 are used only to identify fields by which records are matched. The values M1-M9 are not indicators, but do cause MR to turn on when a match occurs. You can then use the MR indicator to condition those operations which are to be done only when records match (see *Columns 9-17* in Chapter 8 and *Columns 23-31* in Chapter *9).*

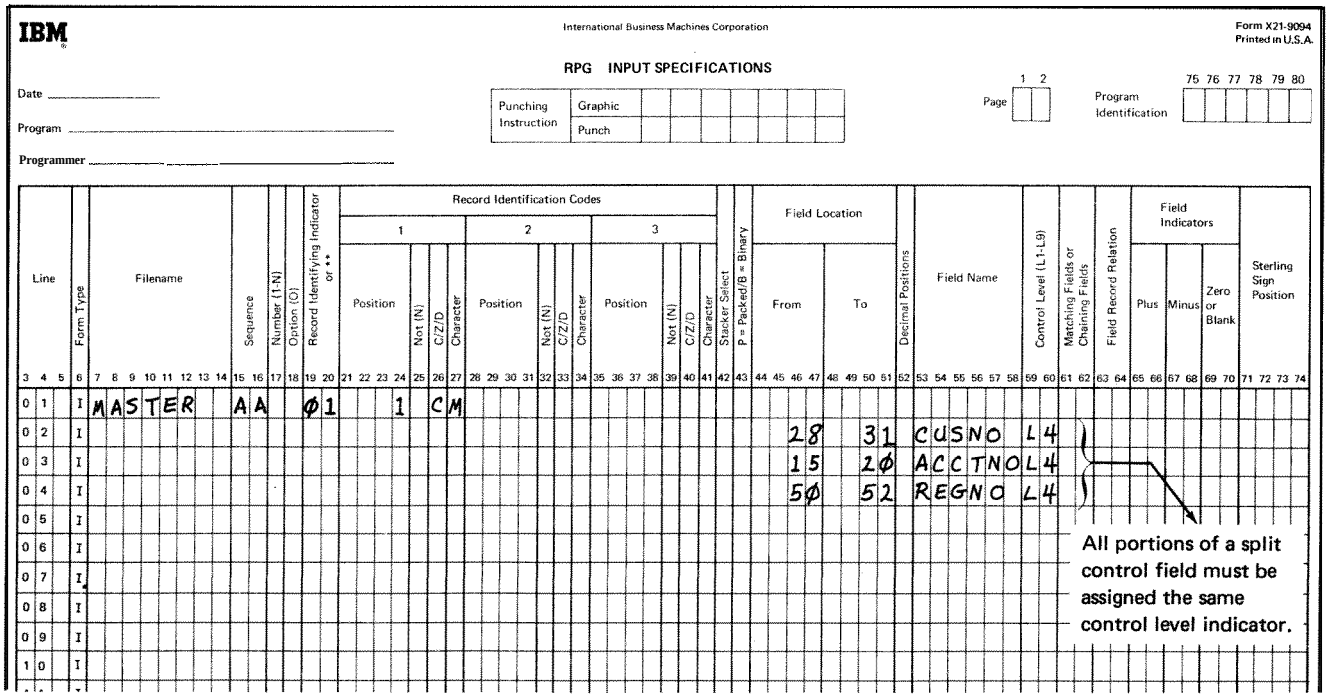


Figure 7-12. **Split** Control **Fields**

When assigning matching field values, remember:

1. Sequence checking is automatically done for all record types with matching field specifications. The contents of the fields to which M1-M9 have been assigned are checked for correct sequence. An error in sequence stops the program. The record which caused the halt is not processed. When the machine is restarted, the next record from the same file is read, Thus, all matching fields must be in the same order, either all ascending or all descending (see *Column* 18 in Chapter 4).

2. Not all files used in the job must have matching fields. Not all record types within one file must have matching fields either. But at least one record type from two files must have matching fields if files are ever to be matched.

3. The same number of matching fields must be specified for all record types which are used in matching. The same matching record values must also be used for all types (see *Examples, Example* 1).

4. All match fields given the same matching record value (Ml-M9) must be the same length and type (alphameric or numeric).

5. Record columns of different matching fields may overlap, but the total length of all fields must not exceed 144 characters.

6. If more than one matching field is specified for a record type, all the fields are combined and treated as one continuous matching field (see *Examples, Example* 2). They are combined according to ascending sequence of matching record values.

7. Matching fields may not be split. This means that the same matching field value cannot be used twice for one type of card.

8. Matching fields may be either alphameric or numeric. However, all matching fields given the same matching record value (M1-M9) are considered numeric if any one of those matching fields is described as numeric.

9. When numeric fields having decimal positions are matched, they are treated as if they had no decimal position.

10. Only the digit portions of numeric match fields are compared. Even though a field is negative it is considered to be positive since the sign of the numeric field is ignored. Thus, a -5 will match with a +5.

11. Whenever more than one matching record value is used, all match fields must match before the MR indicator turns on. For example, if matching fields M1, M2, M3 are specified, all three fields from one record must match all three fields from the other record. A match on only the M1 and M2 fields will not turn on the MR indicator (see *Examples, Example 1*).

12. Field names are ignored in matching record operations. Therefore, fields from different record types which have been assigned the same match level may have the same name.

13. If you have defined an alternate collating sequence for your program, alphameric fields are matched according to the sequence you have specified.

Note: Additional rules applying to matching records when used with entries in the Field Record Relation columns are discussed in *Columns 63-64* in this chapter.

Matching records for two or more files are processed in the following manner:

1. Whenever a record from the primary file matches a record from the secondary file the primary file record is processed first. Then the matching secondary file record is processed. (Remember, the record identifying indicator which identifies the record type just selected is on at the time the record is processed. This indicator is often used to control the type of processing that takes place.)

2. Whenever records from ascending files do not match, the record having the lowest match field content is processed first (Figure 7-13). Whenever records from descending files do not match, the record having the highest match field content is processed first.

3. A record type which has no matching field specification is processed immediately after the record it follows. The MR indicator is off. If this record type is first in the file, it is processed first even if it is not in the primary file (see Figure 7-13).

4. The matching of files makes it possible to enter data from primary records into their matching secondary records since the primary record is processed before the matching secondary record. However, the transfer of data from secondary records into matching primary records can only be done through look ahead fields (see *Look Ahead* in Chapter 10).

For additional information on matching records from more than two files see *Operation Codes, Force,* in Chapter 10.

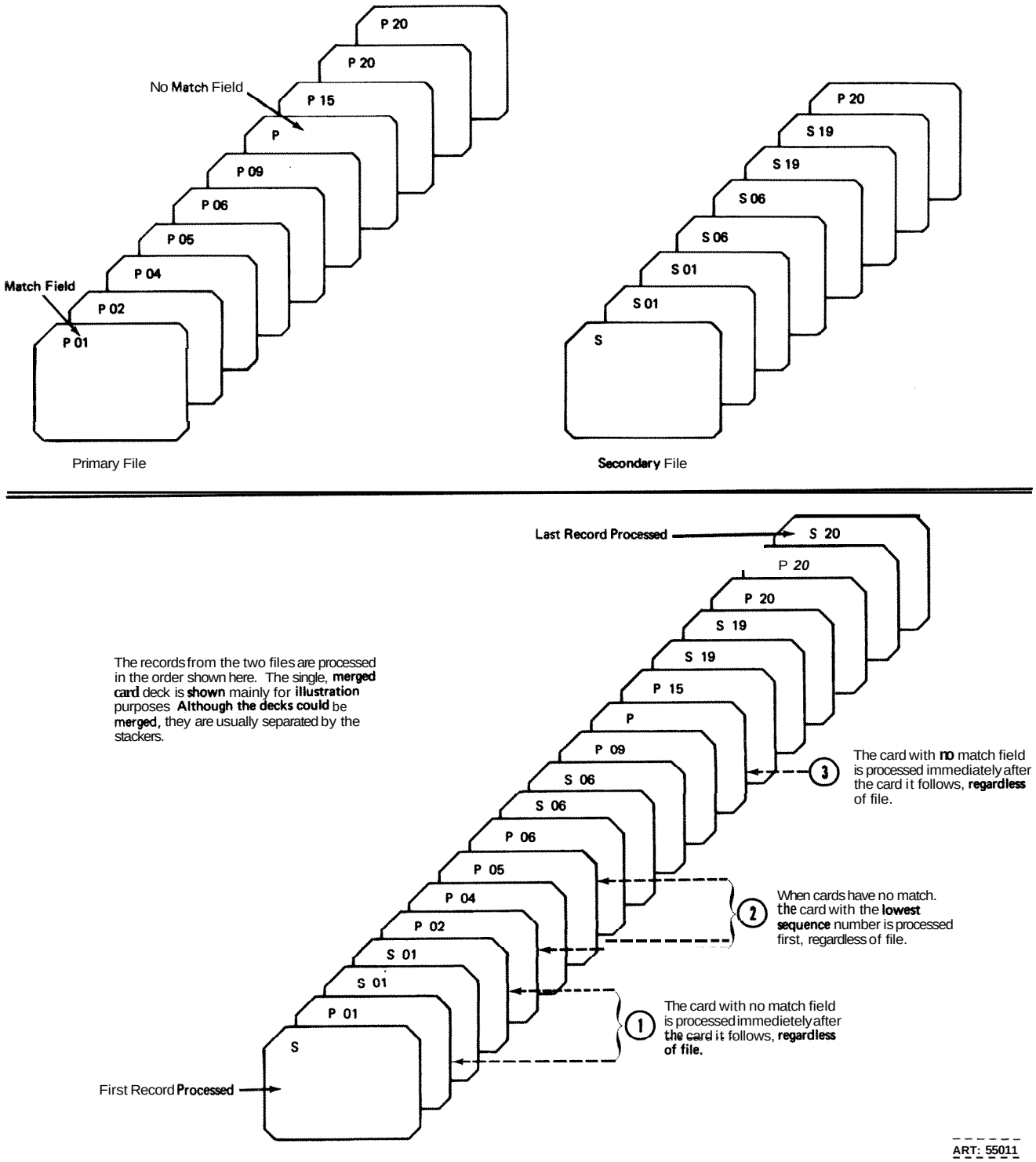


Figure 7-13. Processing Order According to Matching Records

Sequence Checking

Make an entry in columns 61-62 when you want to sequence check records within one input or combined file. This entry causes sequence checking of the data in the fields to which M1-M9 have been assigned (see *Columns* 15-16 in this chapter for sequence checking of record types). You may use as many as nine fields (M1-M9) to sequence check, The sequence (ascending or descending) of your record file must be specified in the file description specifications (see *Column* 18 in Chapter 4). An entry in columns 61-62 indicates that the records are to be checked to see if they really are in the sequence specified (see *Examples, Example 3).*

Examples

Example 1: Figure 7-14 shows three record types that are used in matching records. **All** record types have three matching fields specified and all use the same values (M1, M2, M3) to indicate which fields must match. The MR indicator turns on only if all three match fields in either of the record types from the MASTER file are the same as ail three fields from the record in the WEEKLY file.

Example 2: Figure *7-14* indicates three matching fields on one record. These three are combined and treated as one matching field organized as follows:

DIVSON	DEPT	EMPLNO
M3	M2	M1

The order in which the fields are specified on the input specifications does not affect the organization of the match fields in the computer.

IBM

International Business Machines Corporation

Form X21-9094
Printed in U.S.A.

RPG INPUT SPECIFICATIONS

Date _____

Program _____

Programmer _____

Punching Instruction: Graphic / Punch

Page: 1 2

Program Identification: 75 76 77 78 79 80

Line	Form Type	Filename	Sequence	Number (1-N) Option (O)	Record Identifying Indicator or **	Record Identification Codes 1: Position	1: Not (N)	1: C/Z/D	1: Character	2: Position	2: Not (N)	2: C/Z/D	2: Character	3: Position	3: Not (N)	3: C/Z/D	3: Character	Stacker Select	P = Packed/B = Binary	Field Location: From	Field Location: To	Decimal Positions	Field Name	Control Level (L1-L9)	Matching Fields or Chaining Fields	Field Record Relation	Field Indicators: Plus	Field Indicators: Minus	Field Indicators: Zero or Blank	Sterling Sign Position
01	I	MASTER	AA		01	95		CS																						
02	I																			1	5		EMPLNO		M1					
03	I																			10	25		PAYRAT							
04	I																			36	40		DIVSON		M3					
05	I																			31	35		DEPT		M2					
06	I		BB		02	95		CT																						
07	I																			1	5		EMPLNO		M1					
08	I																			10	15		HRSWKD							
09	I																			20	24		DEPT		M2					
10	I																			25	29		DIVSON		M3					
11	I	WEEKLY	DL		03																									
12	I																			1	5		EMPLNO		M1					
13	I																			6	10		DIVSON		M3					
14	I																			11	15		DEPT		M2					
15	I																			20	25		HRSWKD							
	I																													
	I																													
	I																													
	I																													
	I																													

Figure **7-14. Match Fields**

Example 3: An input file called MASTER is to be sequence checked using three fields (Figure 7-15). Data from two cards is shown below:

Data from First Card		*Data from Second Card*	
DEPT	008	DEPT	003
REGION	051	REGION	025
DIVSON	003	DIVSON	005

In sequence checking, all fields are treated as one continuous field. Thus, the matching fields look like:

	M3	M2	M1
card 1	003	051	008
card 2	005	025	003

The matching field from card 1 is compared with the matching field from card 2. If the file is specified to be in ascending sequence, the cards are in order since 005025003 is higher than 003051008. However, if the file is specified as having a descending sequence, card 2 is out of order.

COLUMNS 63-64 (FIELD RECORD RELATION)

Entry	*Explanation*
01-99	Record identifying indicator assigned to a record type.
L1-L9	Control level indicator previously used.
MR	Matching record indicator.
U1-U8	External indicator previously set.
HI-H9	Halt indicator previously used.

Columns 63-64 have several uses which are discussed after these general rules:

1. All fields, including matching or control fields, that have no field record relation specification should come before those that do.

2. All fields related to one record type (that is, having the same Field Record Relation entry) should be entered as a group in specification lines following one another for more efficient use of core storage. These fields could, however, be entered in any order.

3. All portions of a split control field *must* be assigned the same field record relation indicator and must be entered as a group in specification lines following one another (see *Examples, Example 1*). For more information on split control fields, see *Columns* 59-60 in this chapter.

IBM

International Business Machines Corporation

Form X21-9094
Printed in U.S.A.

RPG INPUT SPECIFICATIONS

Date _____
Program _____
Programmer _____

Punching Instruction: Graphic / Punch

Page

Program Identification 75 76 77 78 79 80

Line	Form Type	Filename	Sequence	Number (1-N)	Option (O)	Record Identifying Indicator or **	Record Identification Codes 1: Position	1: Not (N)	1: C/Z/D	1: Character	2: Position	2: Not (N)	2: C/Z/D	2: Character	3: Position	3: Not (N)	3: C/Z/D	3: Character	Stacker Select	P = Packed/B = Binary	Field Location: From	Field Location: To	Decimal Positions	Field Name	Control Level (L1-L9)	Matching Fields or Chaining Fields	Field Record Relation	Field Indicators: Plus	Field Indicators: Minus	Field Indicators: Zero or Blank	Sterling Sign Position
3 4 5	6	7 8 9 10 11 12 13 14	15 16	17	18	19 20	21 22 23 24	25	26	27	28 29 30 31	32	33	34	35 36 37 38	39	40	41	42	43	44 45 46 47	48 49 50 51	52	53 54 55 56 57 58	59 60	61 62	63 64	65 66	67 68	69 70	71 72 73 74
0 1	I	MASTER																													
0 2	I																				1	10		NAME							
0 3	I																				11	15		NUM							
0 4	I																				17	19		DEPT		M1					
0 5	I																				20	22		REGION		M2					
0 6	I																				23	25		DIVSON		M3					
0 7	I																														

Figure 7-15. Match Fields (Sequence Checking Within a File)

Record Identifying indicators (01-99)

Columns 63-64 are commonly used when several record types have been specified in an OR relationship. Fields which have no field record relation indicator are associated with all the record types in the OR relationship. This is fine when all record types have the same fields. But if the record types in the OR relationship have some fields that are the same and some that are not the same, you do not want to associate every field with all records. Therefore, you must have some way of relating a field to a certain record. To do this, place in columns 63-64 the record identifying indicator found in columns 19-20 of the record type on which the field is found (see ***Examples, Example*** 2).

Control fields (indicated by entries in columns 59-60) and matching fields (indicated by entries in columns 61-62) may also be related to a particular record type in an OR relationship by a field record relation entry. Control fields or matching fields that are not related to any particular record type in the OR relationship by the field record relation indicator are used with all record types in the OR relationship.

When two control fields have the same control level indicator or two matching fields have the same matching level entry, it is possible to assign a field record relation indicator to just one of the control fields or to just one of the matching fields. In this case, only the specification having the field record relation indicator is used when that indicator is on. If none of the field record relation indicators are on for that control field or matching field, the specification without a field record relation indicator is used. Control fields and matching fields cannot have an L1-L9 or MR entry in columns 63-64.

Control Level (L1-L9) and Matching Record (MR) Indicators

Another situation for which you may use these columns is when you wish to accept and use data from a particular field only when a certain condition (such as matching records or a control break) occurs. You indicate the conditions under which you accept data from a field by indicator L1-L9 or MR. Data from the field named in columns 53-58 is accepted only when the indicator is on (see ***Examples, Example 3)***.

External Indicators (U1-U8)

You may also use these columns to condition a specification by an external indicator (U1-U8). The external indicator, which you set prior to processing, controls whether or not the specification is done. When the indicator is on, the specification is done; when it is off, the specification is not done.

External indicators are primarily used when file conditioning is done by an entry in columns 71-72 in the file description specifications. However, they may also be used to condition when a specification should or should not be done even though file conditioning is not specified. See ***Indicators, External Indicators,*** in Chapter 10.

Halt Indicators (H1-H9)

A halt indicator is used to relate a field to a record that is in an OR relationship and also has a halt indicator specified in columns 19-20.

Examples

Example 1: Split control fields on one record type must have the same record relation entry. Figure 7-16, insert **A,** shows several record types with split control fields in each. The record identified by a **1** in column **95** has two split control fields:

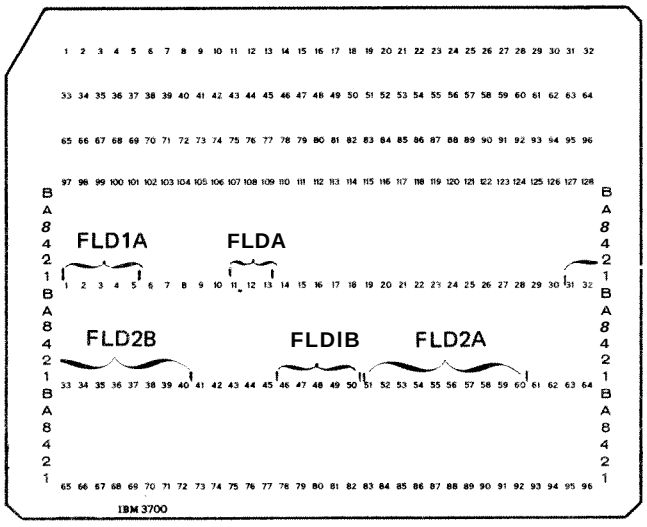
FLDlA and FLD1B
FDL2A and FLD2B

The record with a 2 in column 95 has three split control fields.

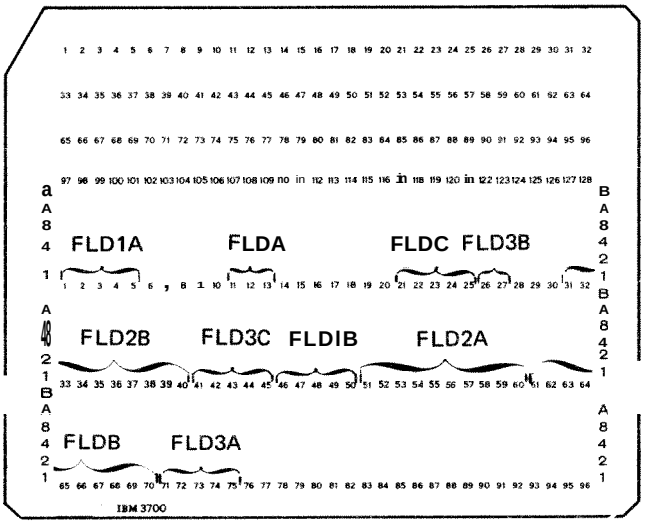
FLD1A and **FLDlB**
FLD2A and FLD2B
FLD3A, FLD3B, and FLD3C

The third record type, identified by the 3 in column **95,** also has three split control fields:

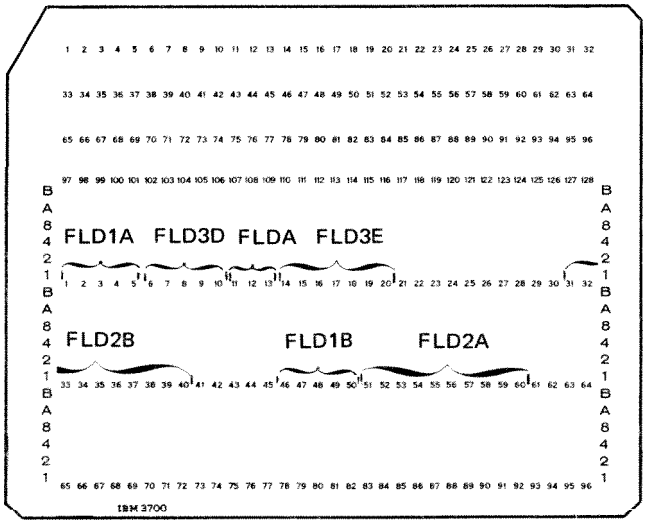
FLDlA and **FLDlB**
FLD2A and FLD2B
FLD3D and FLD3E



Record identification code = 1



Record identification code = 2

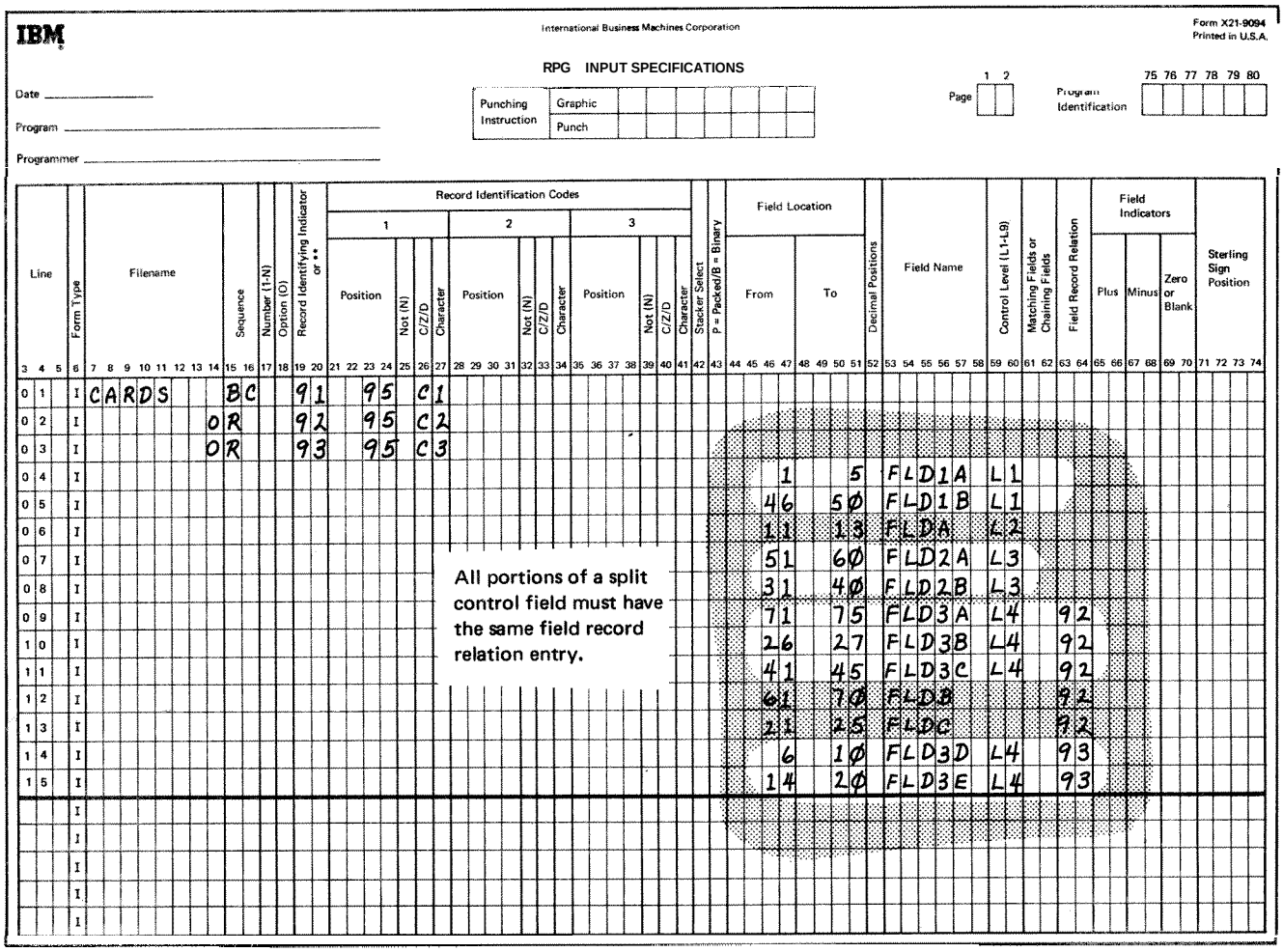


Record identification code = 3

(A)

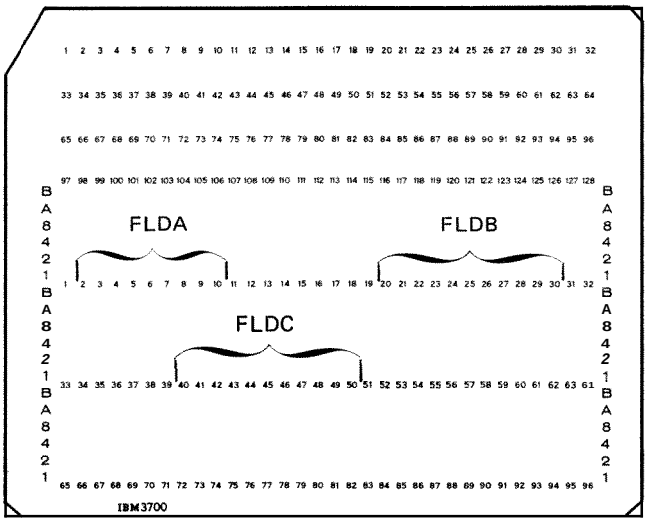
Figure 7-16. Field **Record** Relation **(Split** Control **Fields)** (part 1 of *2)*

All portions of the split control field must be assigned the same control level indicator and all must have the same field record relation entry. Figure 7-16, insert B, shows the field record relation required for the three record types.

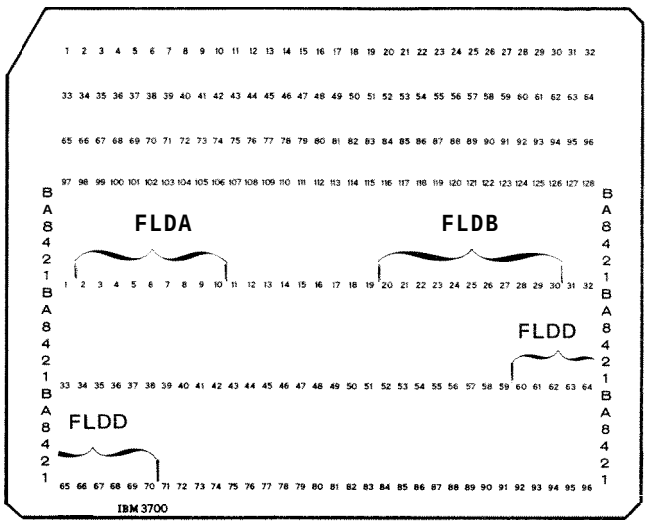


Ⓑ

Figure 7-16, Field Record Relation (Split Control Fields) (part 2 of 2)



Record identification code = 5



Record identification code = 6

IBM

International Business Machines Corporation

Form X21-9094
Printed in U.S.A.

RPG INPUT SPECIFICATIONS

Date _____

Program _____

Programmer _____

Punching Instruction	Graphic							
	Punch							

Page 1 2

Program Identification 75 76 77 78 79 80

Line	Form Type	Filename	Sequence	Number (1-N)	Option (O)	Record Identifying Indicator or **	Record Identification Codes 1: Position	1: Not (N)	1: C/Z/D	1: Character	2: Position	2: Not (N)	2: C/Z/D	2: Character	3: Position	3: Not (N)	3: C/Z/D	3: Character	Stacker Select	P = Packed/B = Binary	Field Location: From	Field Location: To	Decimal Positions	Field Name	Control Level (L1-L9)	Matching Fields or Chaining Fields	Field Record Relation	Field Indicators: Plus	Field Indicators: Minus	Field Indicators: Zero or Blank	Sterling Sign Position
3 4 5	6	7 8 9 10 11 12 13 14	15 16	17	18	19 20	21 22 23 24	25	26	27	28 29 30 31	32	33	34	35 36 37 38	39	40	41	42	43	44 45 46 47	48 49 50 51	52	53 54 55 56 57 58	59 60	61 62	63 64	65 66	67 68	69 70	71 72 73 74
0 1	I	REPORT	AA			14	1		C	5																					
0 2	I		OR			16	1		C	6																					
0 3	I																				2	10		FLDA							
0 4	I																				20	30		FLDB							
0 5	I																				40	50		FLDC							
0 6	I																				60	70		FLDD							
0 7	I																														
0 8	I																														
0 9	I																														
1 0	I																														

Figure 7-17. Field Record Relation

Example 2: Figure **7-17** shows how record identifying indicators are used to relate a field to a record. The file contains two different types of records, one identified by a 5 in column 1 and the other by a 6 in column **1.** FLDC is related by record identifying indicator 14 to the record type which is identified by a 5 in column **I.** FLDD is related to the record type having a *6* in column 1 by record identifying indicator **16.** This means that FLDC is found on only one type of record (that identified by *5* in column 1) and FLDD is found only on the other type. FLDA and FLDB are found on both types since they are not related to any one type by a record identifying indicator.

Example 3: Suppose you were printing a monthly report showing all items sold in each department in your company. You also want the report to list the name of the manager of each department. Each input record then has the department number (DEPT), the manager's name (MANAGR), and one item (ITEM) that was sold by that department. Fields are described as shown in Figure **7-18.** The records are arranged in order by department.

In the report it is not necessary to print the manager's name for every item that was sold in his department. Instead, it should be printed only when the first record containing an item sold in a different department is read. The field called DEPT is established as a control field.

Remember that the manager's name is printed only when information in the control field changes. Thus the information from the field called MANAGR is not used often. It would be wasted time to accept that information every time a record is read. The L1 entry in columns 63-64 indicates that the data from the field called MANAGR is to be accepted and used only when a control break occurs.

COLUMNS 65-70 (FIELD INDICATORS)

Entry	*Explanation*
01-99	Field indicator.
H1-H9	Halt indicator (when checking for an error condition in the data).

Use field indicators 01-99 when you wish to test a field for a condition of either plus, minus, zero, or blank. The indicator specified turns on if the condition is true; it remains off or turns off if the condition is not true. You usually use these same indicators to control certain calculation or output operations (see *Columns* 9-17 in Chapter 8 or *Columns 23-31* in Chapter *9).*

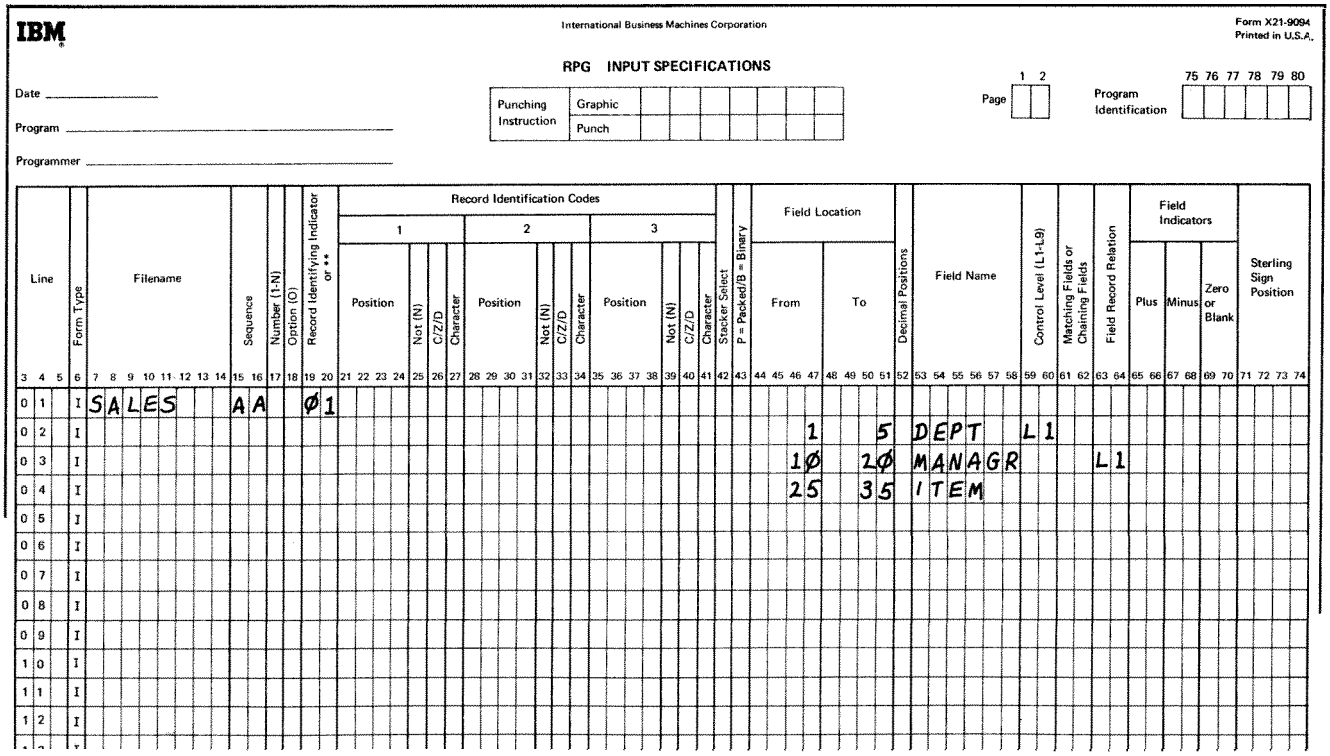


Figure 7-18. Field Record Relation: Accepting Data From a **Field**

The three conditions you may check for are:

1. Plus (columns 65-66). Any valid indicator entered here is turned on if the numeric field named in columns 53-58 is greater than zero.

2. Minus (columns 67-68). Any valid indicator entered here is turned on if the numeric field in columns 53-58 is less than zero.

3. Zero or blank (columns 69-70). Any valid indicator entered here is turned on if a numeric field named in columns 53-58 is all zeros or if an alphameric field is all blanks.

   A numeric field which is all blanks will turn on an indicator specified for all zeros. However, if an alphameric field is all zeros, the field will not turn on an indicator specified for all blanks.

In the input specifications, you specify the indicators that will be used to condition operations. In the calculation specifications and output-format specifications, you actually use these indicators. When conditioning operations, you must know when the indicators will be off and when they will be on. When assigning and using field indicators in columns 65-70 remember:

1. Indicators for plus or minus are off at the beginning of the program. They are not turned on until the condition (plus or minus) is satisfied by the field being tested on the card just read.

2. An indicator assigned to zero or blank is off at the beginning of the program. It remains off until the field being tested is zero or blank.

3. One input field may be assigned two or three field indicators. However, only the one which signals the result of the test turns on; the others are turned off.

4. If the same field indicator is assigned to fields in different record types, its status is always based on the last record type selected.

5. When different field indicators are assigned to fields in different record types, a field indicator turned on will remain on until another record of that type is read. Similarly, a field indicator assigned to more than one field within a single record type will always reflect the status of the last field defined.

6. Field indicators assigned in these columns may be SETON or SETOF in calculation specifications.

Halt Indicators

Specify any halt indicator (H1-H9) in columns 65-70 when you wish to check for an error condition in your data. For example, if a field should not be zero, you can specify a halt indicator to check for that zero condition. If a zero field is found, the halt indicator turns on and the job stops after the record with the zero field has been processed.

Indicators H1-H9 cause the program to halt after the record which caused the indicator to turn on is completely processed.

COLUMNS 71-74 (STERLING SIGN POSITION)

Entry	*Explanation*
Blank	Sterling input is not being used.
S	Sign is in normal position.
1-4096	Number of the column which contains the sign if the sign is not in normal position.

Use columns 71-74 only when processing sterling currency amounts. The position of the sign (+ or -) for the field named in columns 53-58 must be indicated in these columns. The normal position of the sign in a field having decimal positions is in the rightmost decimal position of the pence field. If the field has no decimal positions, the normal sign position is in the last column (units position) of the pounds field. See ***Sterling*** in Chapter 10 for more information.

COLUMNS 75-80 (PROGRAM IDENTIFICATION)

See Chapter 2.

Calculation specifications describe the calculations you want performed on your data and the order in which you want them performed. Each calculation specification can be divided into three parts:

1. When the operation is to be performed (columns 7-17). The indicators entered in these columns determine under what conditions the operation specified is to be done.

2. What kind of operation is to be performed (columns 18-53). Entries in these fields describe the kind of operation to be done. They also specify the data upon which the operation is to be performed.

3. What tests are to be made on the results of the operation (columns 54-59). The indicators entered here signal the result of the operation and may serve to condition other operations.

Write these specifications on the Calculation sheet (Figure 8-1).

COLUMNS 1-2 (PAGE)

See Chapter 2.

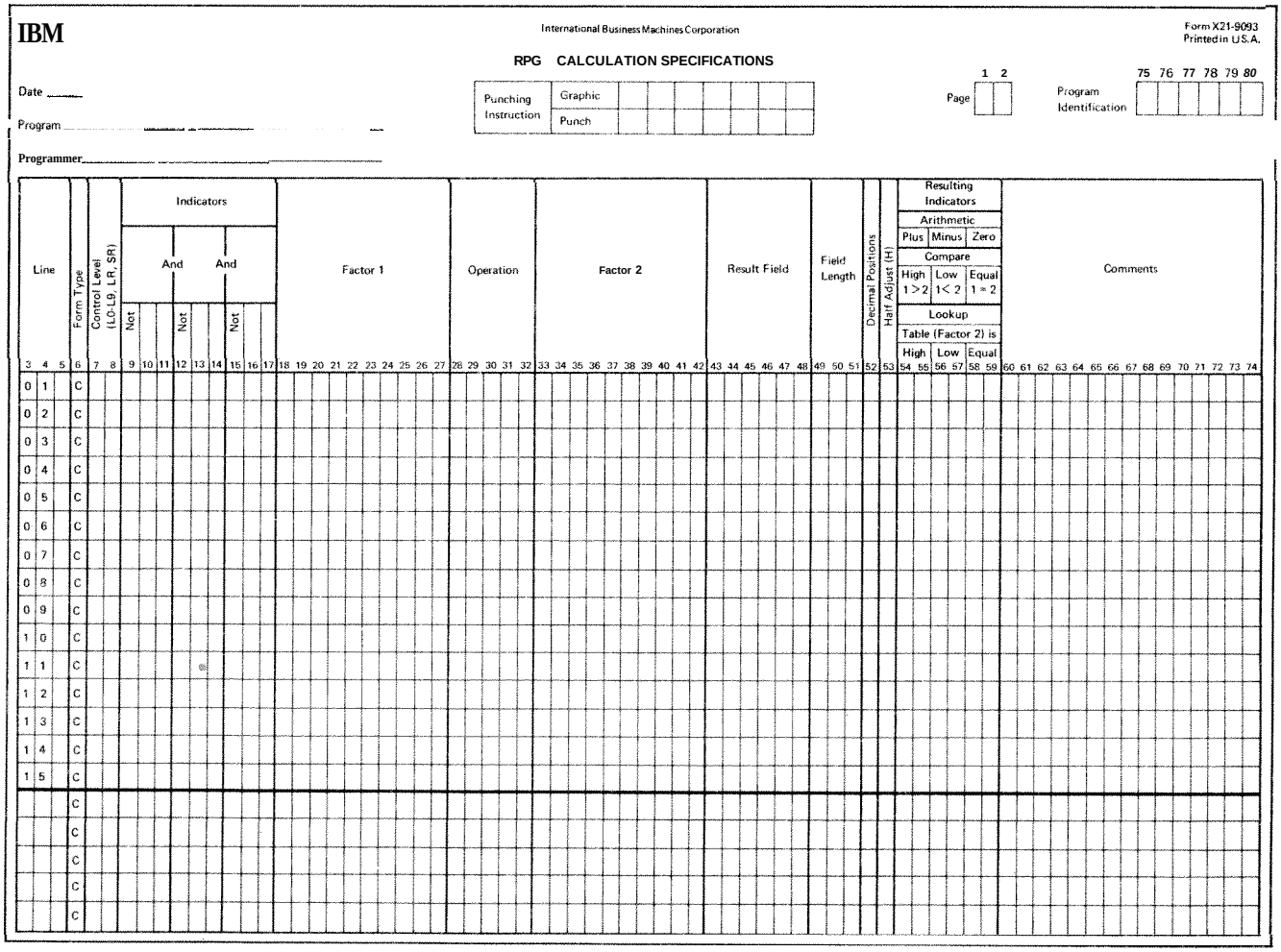


Figure 8-1. Calculation **Sheet**

COLUMNS 3-5 (LINE)

See Chapter 2.

COLUMN 6 (FORM TYPE)

A C must appear in column 6.

COLUMNS 7-8 (CONTROL LEVEL)

Entry	***Explanation***
Blank	Calculation operation is not part of a subroutine and may only be performed for detail calculations.
L0, L1-L9	Calculation operation is done when the appropriate control break occurs:
LR	Calculation operation is done after the last record has been processed.
SR	Calculation operation is part of a subroutine.
AN, OR	Establishes AND and OR relationships between lines of indicators.

If you leave columns 7-8 blank, the operation specified on the same line is done every time a record is read, provided indicators in columns 9-17 allow it (see ***Columns*** 9-17 in this chapter).

Control Level Indicators (LO, L1-L9)

The LO indicator is on during the entire program. You need never assign this indicator, but you may use it. The indicator is often used when no control fields have been assigned. Remember that when a control break occurs, all operations conditioned by control level indicators are done before those that are not conditioned. If you have no control field but want total calculations to be done and total output records to be written or punched, you may use the LO indicator to condition those operations (see ***Examples, Example 1***). Lines conditioned by LO must appear before lines conditioned by L1-L9 or LR.

Use control level indicators L1-L9 to signal when certain operations are to occur. If you specify a control level indicator (L1-L9) in columns 7-8, the operation described on the same specifications line is done only when that indicator is on. Remember that a control level indicator turns on when information in a control field changes (see ***Columns*** 59-60 in Chapter 7).

A control break for a certain level causes all lower control level indicators to turn on. Thus, if you used indicators L3, L2, and L1 in your program, and L3 turns on, L1 and L2 will also turn on. All operations conditioned by L3, L2, and L1 will be done.

An exception is as follows: when a control level indicator used as a record identifying indicator turns on to reflect the type of record read or when a control level indicator turns on by the SETON instruction, only that one control level indicator turns on. All lower level indicators remain off.

Last Record Indicator (LR)

The last record (LR) indicator automatically turns on when the last record is read and processed. You may have certain operations which are to be done only when the last record has been read. Condition these operations with the LR indicator. The last record causes the LR indicator and all other control level indicators specified (L1-L9) to turn on.

Subroutine Lines (SR)

Use columns 7-8 to indicate that a line is part of a subroutine (see ***Subroutines*** in Chapter 10). Subroutine lines must be specified last.

AND/OR Lines (AN,OR)

Columns 7-8 can be used to specify that lines of indicators are in an AND/OR relationship. By using the AND/OR relationship, many lines of indicators may be grouped together to condition an operation. A maximum of twenty OR lines may be used to condition an operation. There is no limit to the number of AND lines that can be used.

The first line of such a group contains blanks in columns 7-8, or an L0-L9, LR, or SR entry if the group of lines is conditioned by a control level indicator or is part of a subroutine. All lines after the first line in the group must have an AN or OR entry in columns 7-8. The last line of the group contains the operation and the necessary operands. All lines in the group prior to the last line must contain blanks in the columns for Factor 1, Factor 2, Operation, Result Field, and Resulting Indicator (see ***Examples, Examples 2 and 3***).

Examples

Example 1: Figure 8-2 shows the format of the report printed by the job described in Figure 8-3. The job shows how total operations can be performed even though there is no control field (no L1-L9 indicators). The job requires:

1. A list of items sold in each district.
2. A total of all sales for each district.
3. A grand total of all sales in all districts.

```
J102      4.50
J202      3.75
K450      2.98
B231      9.08

          20.31 *

G10H     92.79
G10K     98.89
A126      4.29

         195.97 *


         216.28 **
```

Figure 8-2. Format of a Printed Report

The input records have ITEM and COST fields and a one column record identification field. The records are grouped in ascending sequence by district. The record identification code is used to tell which district a record is from. For example, records from district one are identified either by a *1* or an *M* in column 1. Records from district two are identified by either a 2 or an *N* in column 1 (Figure 8-3, insert A).

No field on the records can serve as a control field. Certainly, ITEM and COST cannot. The record identifying field cannot either since one district can be identified by two different codes. This means that the contents of this one column identifying field can change even though the district number cannot. Therefore, in order to get total operations without the use of a control field, LO must be used (see line 05 of Figure 8-3, insert B). Assume that the five records shown in Figure 8-4 are read. Refer to Figure 8-3 as you read the description of operations performed for each record read.

IBM

International Business Machines Corporation

Form X21-9094
Printed in U.S.A.

RPG INPUT SPECIFICATIONS

Date

Program

Programmer

Punching Instruction: Graphic / Punch

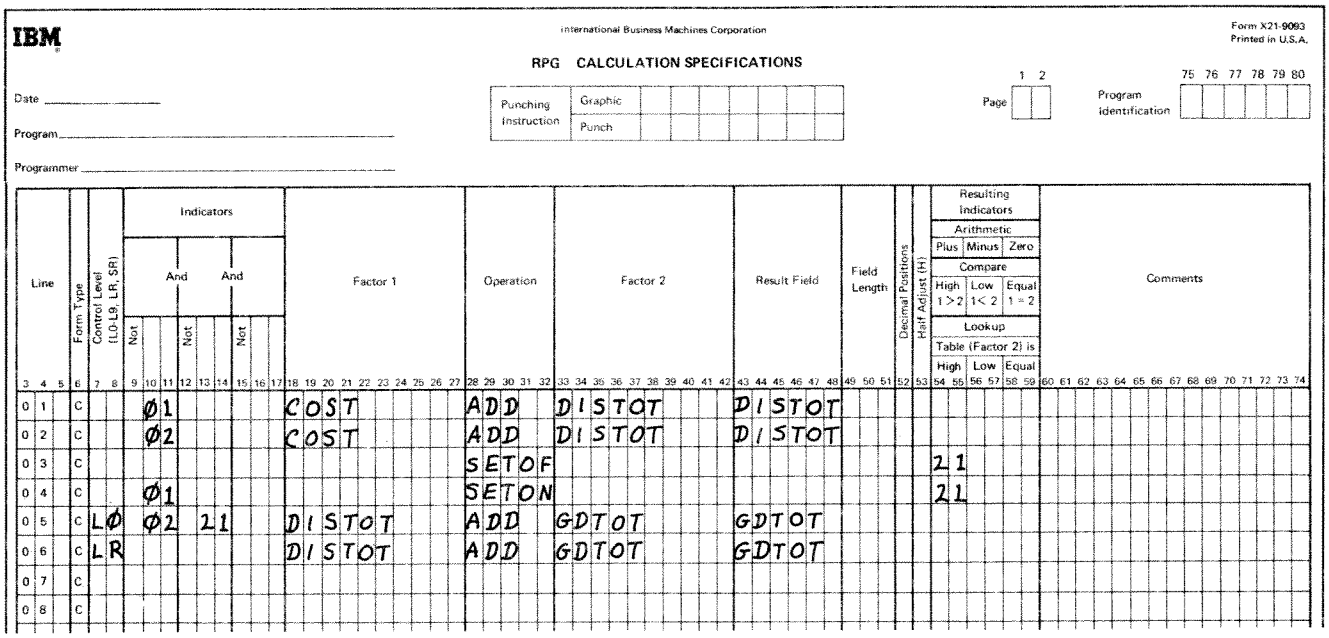
Page 1 2

Program Identification 75 76 77 78 79 80

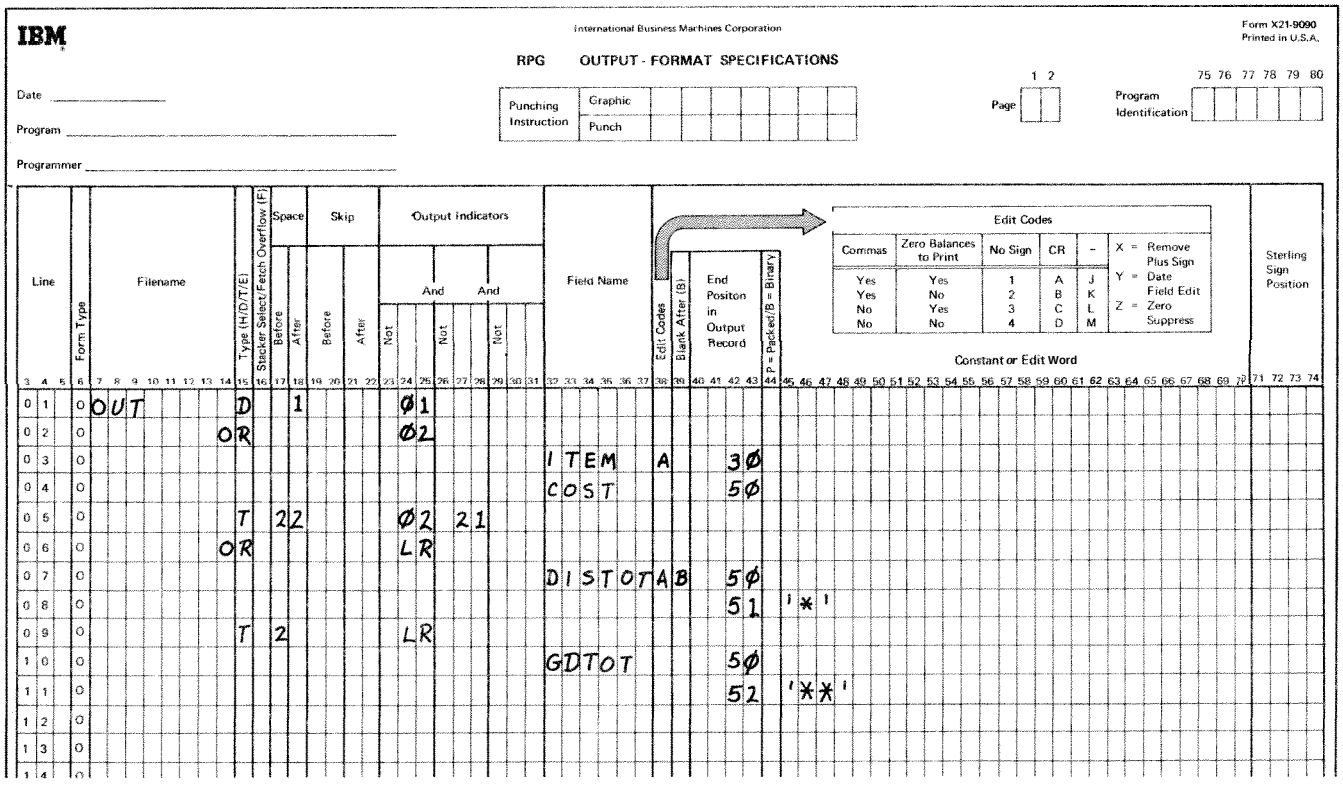
Line	Form Type	Filename	Sequence	Number (1-N)	Option (O)	Record Identifying Indicator or **	Record Identification Codes 1: Position	1: Not (N)	1: C/Z/D	1: Character	2: Position	2: Not (N)	2: C/Z/D	2: Character	3: Position	3: Not (N)	3: C/Z/D	3: Character	Stacker Select	P = Packed/B = Binary	Field Location From	Field Location To	Decimal Positions	Field Name	Control Level (L1-L9)	Matching Fields or Chaining Fields	Field Record Relation	Field Indicators Plus	Field Indicators Minus	Field Indicators Zero or Blank	Sterling Sign Position
01	I	INPUT	AA			01	1		C	1																					
02	I		OR				1		C	M																					
03	I																				10	15		COST							
04	I																				20	30		ITEM							
05	I		BB			02	1		C	2																					
06	I		OR				1		C	N																					
07	I																				30	35		COST							
08	I																				40	50		ITEM							
09	I																														
10	I																														
11	I																														

(A)

Figure 8-3. Use of the LO Indicator (Part 1 of 2)



(B)



(C)

Figure 8-3. Use of the LO Indicator (Part 2 of 2)

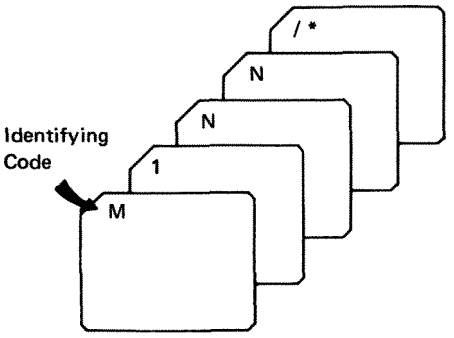
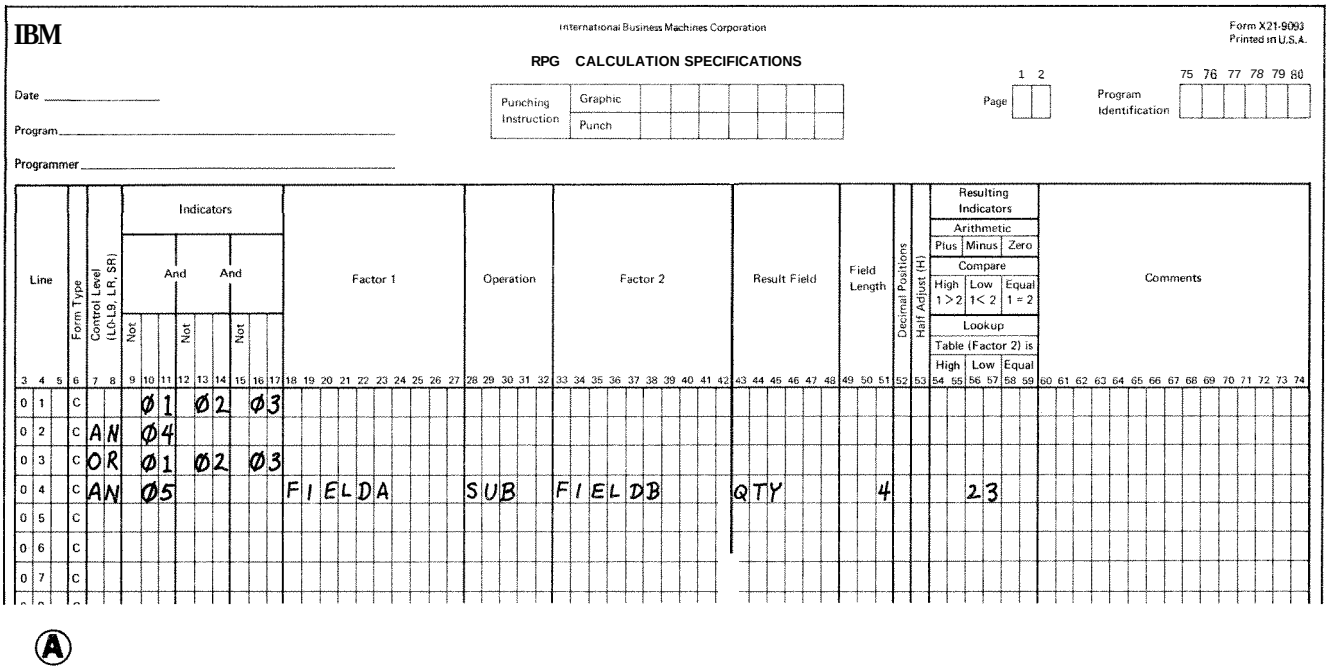


Figure 8-4. Data Cards with No Control Fields

Record	*Indicators On*	*Operations Performed*
(1)	LO	01 turns on. No total operations are performed because conditions in lines 5 and *6* (Calculation sheet) are not met. (Remember that operations conditioned by control level indicators in columns 7-8 are performed first.) **COST** is added to **DISTOT.** 21 is set on. **ITEM** and **COST** are printed out. 01 is turned off. 21 remains on.
(2)	LO, 21	01 is turned on. No total operations are performed. **COST** is added to **DISTOT.** **ITEM** and **COST** are printed out. 01 is turned off. 21 remains on.
(3)	LO, 21	02 turns on. **DISTOT** is added to **GDTOT.** (Conditions for the total operation in line 5 have been met.) **DISTOT** is printed out. **COST** is added to **DISTOT.** 21 is set off. **ITEM** and **COST** are printed out. 02 is turned off.
(4)	LO	02 is turned on. No total operations are performed. **COST** added to **DISTOT.** **ITEM** and **COST** are printed out. 02 is turned off.
(5)	LR	**DISTOT** added to **GDTOT** (LR indicator is on). **DISTOT** and **GDTOT** printed out.

Example 2: Figure 8-5, insert A shows the use of AN and OR entries to group lines of indicators. When indicators 01, 02, 03 and 04 are on, or when indicators 01, 02, 03 and 05 are on, the calculation will be performed.

Example 3: Figure 8-5, insert B illustrates a case in which three conditions will cause the L4 total calculations to be performed: 01 and 02 are on, but not 03; or 01 and 03 are on, but not 02; or 02 and 03 are on but not 01.



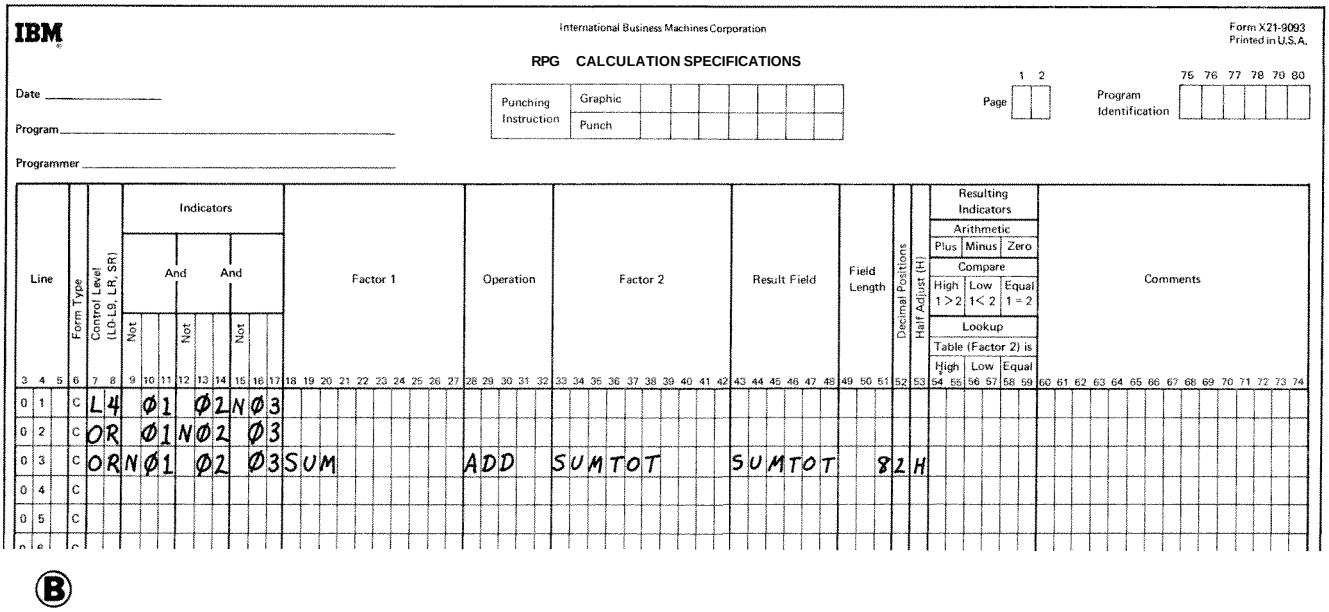


Figure 8-5. Use of AND/OR Lines for Indicators

COLUMNS 9-17 (INDICATORS)

	Explanation
Blank	Operation is performed for every card read.
01-99	Resulting indicators used elsewhere in the program.
L1-L9	Control level indicators previously assigned.
LR	Last record indicator.
MR	Matching record indicator.
H1-H9	Halt indicators assigned elsewhere.
U1-U8	External indicators previously set.
OA-OG, OV	Overflow indicator previously assigned.

Use columns 9-17 to assign indicators that control when an operation is or is not to be done. You may use from one to three indicators on a line. By using AN or OR entries in columns 7-8, many indicators can be used to condition one operation. A maximum of twenty OR lines may be used to condition an operation. There is no limit to the number of AND lines that can be used.

There are three separate fields (9-11, 12-14, and 15-17) on each line, one for each indicator. If the indicator must not be on in order to condition the operation, place an *N* before the appropriate indicator (eolumns 9, 12, 15).

All three indicators on one line are in an AND relationship with each other. The indicators on one line, or indicators in grouped lines, plus the control level indicator (if used in columns 7-8) must all be exactly as specified before the operation is done. See *Examples, Example 1.*

Use any record identifying indicators previously specified in columns 19-20 on the Input sheet to condition an operation that is to be done only for a certain type of record (see *Examples, Example* 2).

Use any field indicators previously specified in columns 65-70 on the Input sheet to condition an operation that is to be done only after the status of a field has been checked and has met certain conditions (see *Examples, Example 3).*

Use any resulting indicators specified in columns 54-59 on the Calculation sheet to condition operations according to the results of previous calculation operations (see the example in *Columns* 54-59 in this chapter).

Use any halt indicators previously used in columns 65-70 on the Input sheet or in columns 54–59 on the Calculation sheet to prevent the operation from being done when a specified error condition has been found in the input data (see *Columns* 19-20 in Chapter 7) or on previous calculations. This is necessary because the record that causes the halt condition will be completely processed before your program stops. Thus, if the operation is performed even on an error condition, the results are in error. It is also possible to use a halt indicator to condition an operation that is to be done only when an error occurs.

Use the matching record (MR) indicator to condition an operation that is to be done only when matching records have been found.

Use any external indicator, including any previously specified in columns 71-72 on the File Description sheet, to condition which operations should be done and which files should be used for a specific job.

Use the last record (LR) indicator to condition all operations that are to be done at the end of the job.

Use any control level indicators specified in columns 59-60 on the Input sheet, or in columns 54-59 on the Calculation sheet. If control level indicators are used in these columns instead of in columns 7-8, the operation is performed on only the first record of a new control group.

Use any overflow indicators previously specified in columns 33-34 on the File Description sheet to condition operations that are to be done when the last line to be printed on a page has been reached. See *Indicators* in Chapter 10 for more information.

The relationship between columns 7-8 and columns 9-17 is explained in the following discussion.

When a control level indicator (L1-L9) is specified in columns 7-8 and MR is specified in columns 9-17, MR indicates the matching condition of the previous record and not the one just read that caused the control break. After all operations conditioned by control level indicators (specified in columns 7-8 of the Calculation sheet) are done, MR then indicates the matching condition of the record just read.

When a control level indicator is used in columns 9-17 and columns 7-8 are not used, the operation conditioned by the indicator is done only on the record that causes that control break or any higher level control break.

In one program cycle all operations conditioned by control level indicators in columns 7-8 are done before operations that are conditioned by control level indicators in columns 9-17 (see *Examples, Examples 4).*

Examples

Example 4: Figure 8-6 shows the use of control level indicators to condition calculation operations. The operation in line **03** may be done when the **L2** indicator is on provided the other conditions are met. Indicator 10 must be on. The **L3** indicator must not be on.

The operation conditioned by both **L2** and NL3 is done only when a control level 2 break occurs. These two indicators are used together because this operation is not to be done when a control level **3** break occurs, even though L2 is also on.

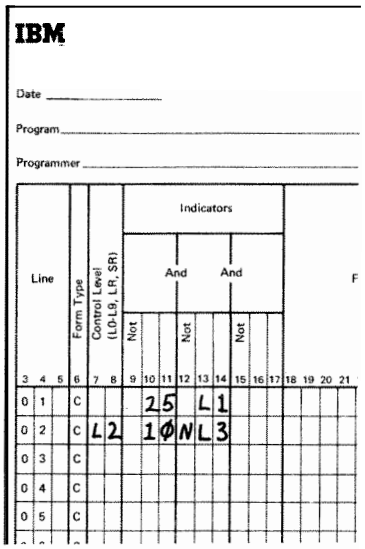
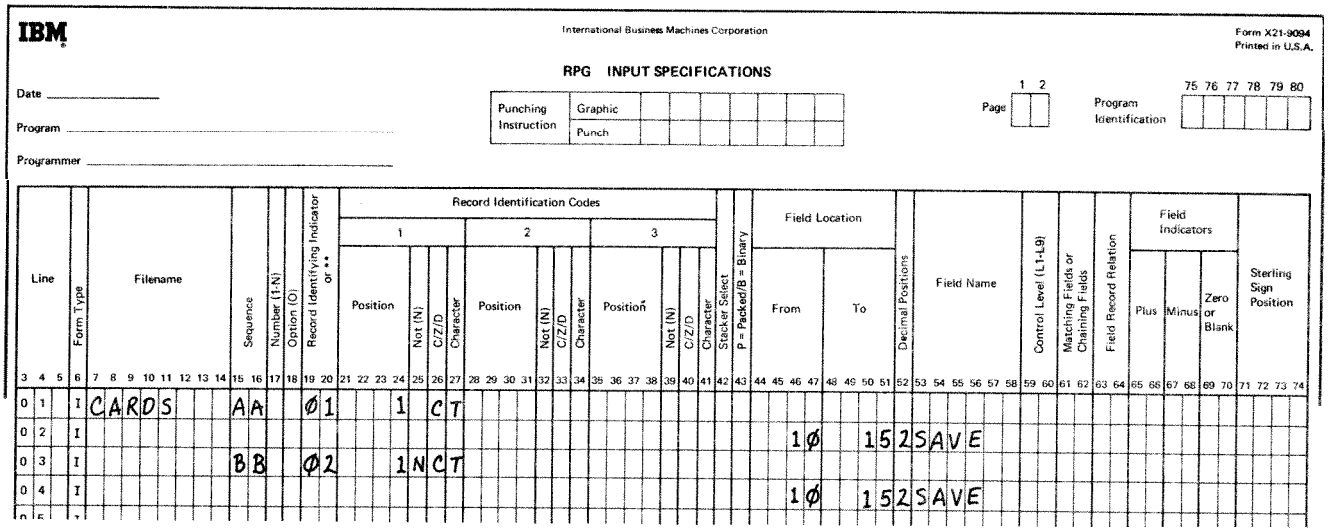


Figure 8-6. Conditioning Operations (Control Level Indicators)

Example 2: Figure 8-7 shows how a record identifying indicator is used to condition an operation. When a record is read that has a T in column 1, the **01** indicator turns on. If this indicator is on, the field named SAVE is added to SUM. When a record having no T in column 1 is read, the **02** indicator is on. The subtract operation, since it is conditioned by **02,** is then done instead of the add operation.



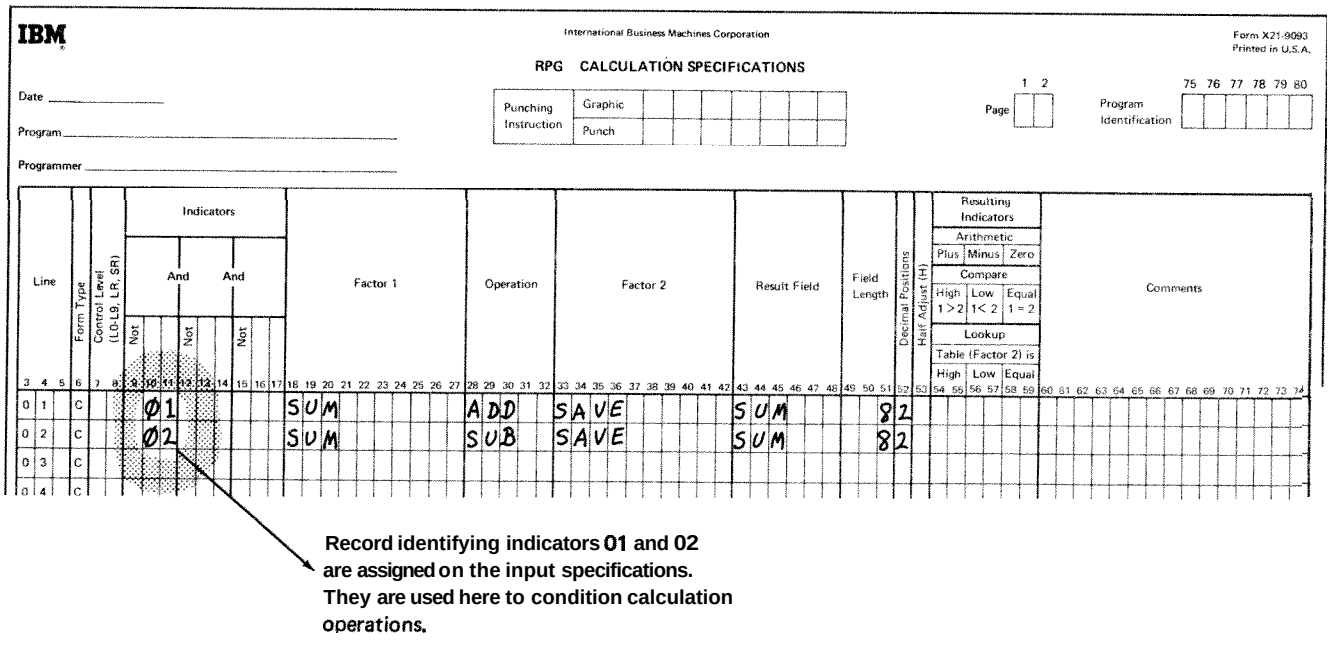
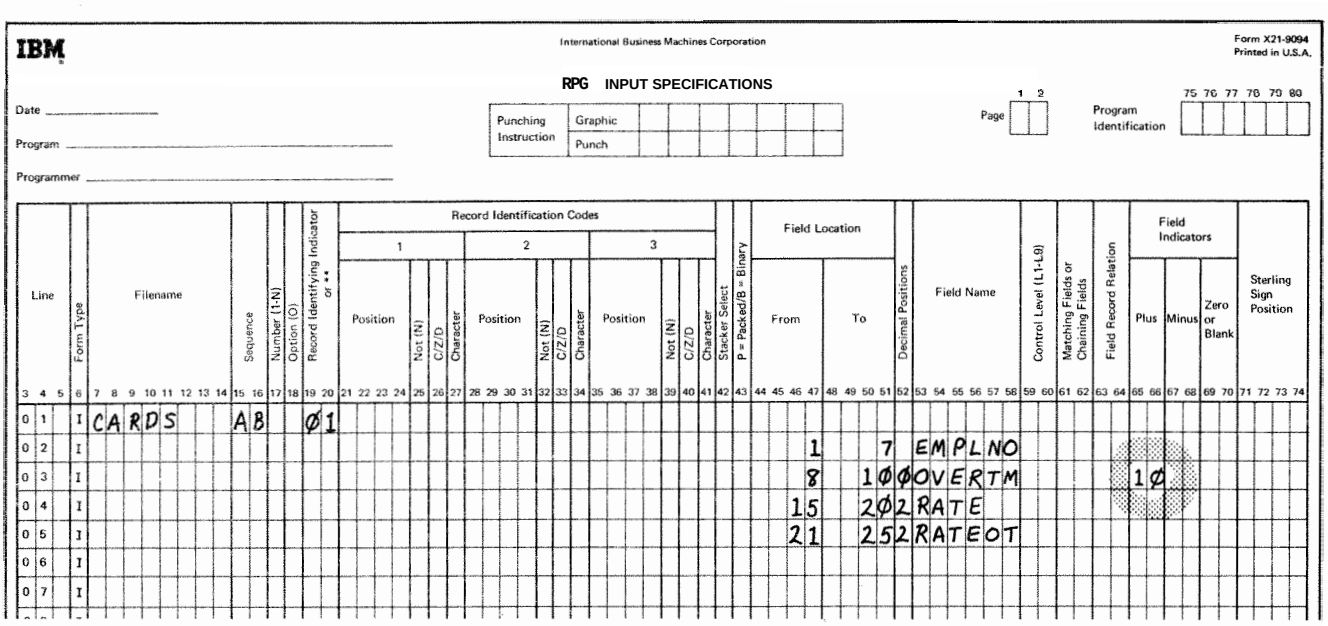


Figure 8-7. Conditioning **Operations (Resulting Indicators)**

Example 3: Figure 8-8 shows the use of field indicators to condition operations. Assume the job is to find weekly earnings including overtime. The overtime field is checked to see if any overtime has been put in. If the employee has worked overtime, the field is positive and indicator 10 turns on. In all cases the weekly regular wage is calculated. However, overtime pay is calculated only if indicator 10 is on (lines 03 and **04).**

Example **4:** Line **02** of Figure **8-8** shows the use of a control level indicator in columns 9-17. Assume that indicator **25** represents a record type and that a control level **2** break occurred when record type 25 was read. L1 and **L2** are both on. **All** operations conditioned by the control level indicators in columns 7-8 are performed before operations conditioned by control level indicators in columns 9-17. Thus, the operation in line **03** occurs before the operation in line **02.** The operation in line 02 is done on the first record of the new control group indicated by 25, whereas the operation in line **03** is a total operation done for all records of the previous control group.



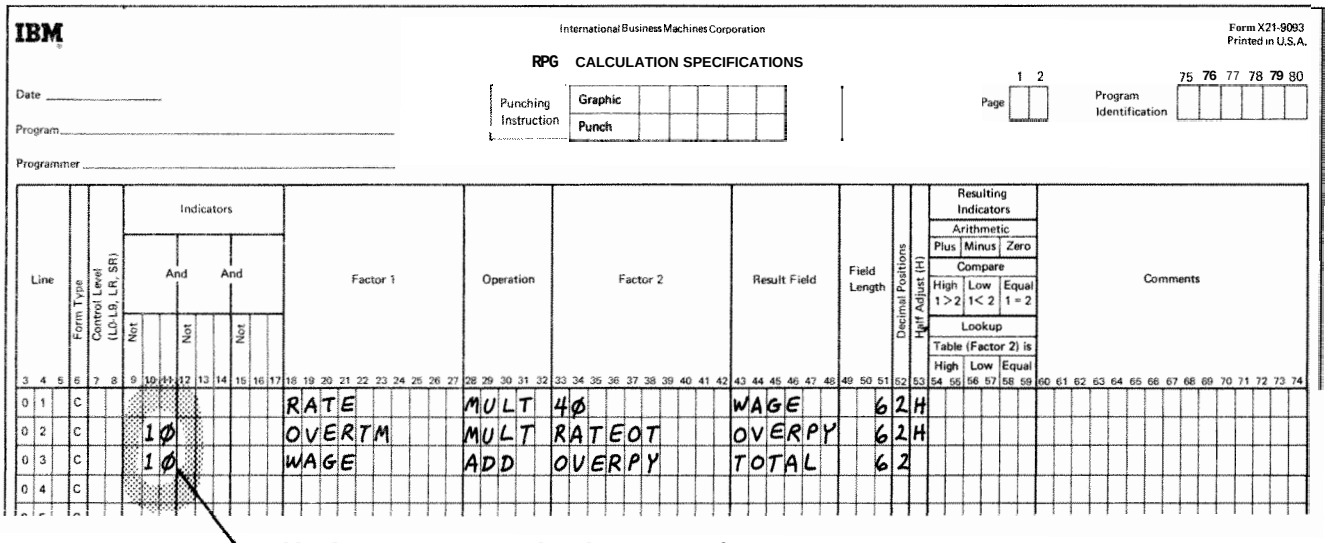


Figure 8-8. Conditioning Operation (Field Indicators)

COLUMNS 18-27 (FACTOR 1) AND COLUMNS 33-42 (FACTOR 2)

Use columns 18-27 and 33-42 to name the fields or to give the actual data (literals) on which an operation is to be performed. The entries you can use are:

1. The name of any field that has been defined.
2. Any alphameric or numeric literal.
3. Any subroutine, table, or array name.
4. Any date field names (UDATE, UMONTH, UDAY, UYEAR).
5. The special names PAGE, PAGE1, or PAGE2.
6. A label for a TAG or ENDSR operation (Factor 1 only). A label for a GOTO operation (Factor 2 only).
7. A filename for a CHAIN, DEBUG, DSPLY, or FORCE operation (Factor 2 only).

An entry in Factor 1 must begin in column 18; an entry in Factor 2 must begin in column 33.

The entries you use depends upon the operation you are describing. Some operations need entries in both sets of columns, some need entries in only one, and some need no entries at all. See *Columns* 28-32 in this chapter for more information on operation codes. If you are naming a subroutine, see *Subroutines* in Chapter 10.

Literals

A literal is the actual data used in an operation rather than the field name representing that data. A literal may be either alphameric or numeric.

Consider the following rules when using an alphameric literal (Figure 8-9, insert A):

1. Any combination of characters may be used in an alphameric literal. Blanks are also valid.
2. The maximum length of an alphameric literal is 8 characters.
3. Alphameric literals must be enclosed by apostrophes (').
4. An apostrophe required as part of a literal is represented by two apostrophes. For example, the literal O'CLOCK would be written as O''CLOCK.
5. Alphameric literals may not be used for arithmetic operations.

Consider the following *rules* when using a numeric literal (see Figure 8-9, insert B):

1. A numeric literal consists of any combination of the digits 0-9. A decimal point or sign may also be included.

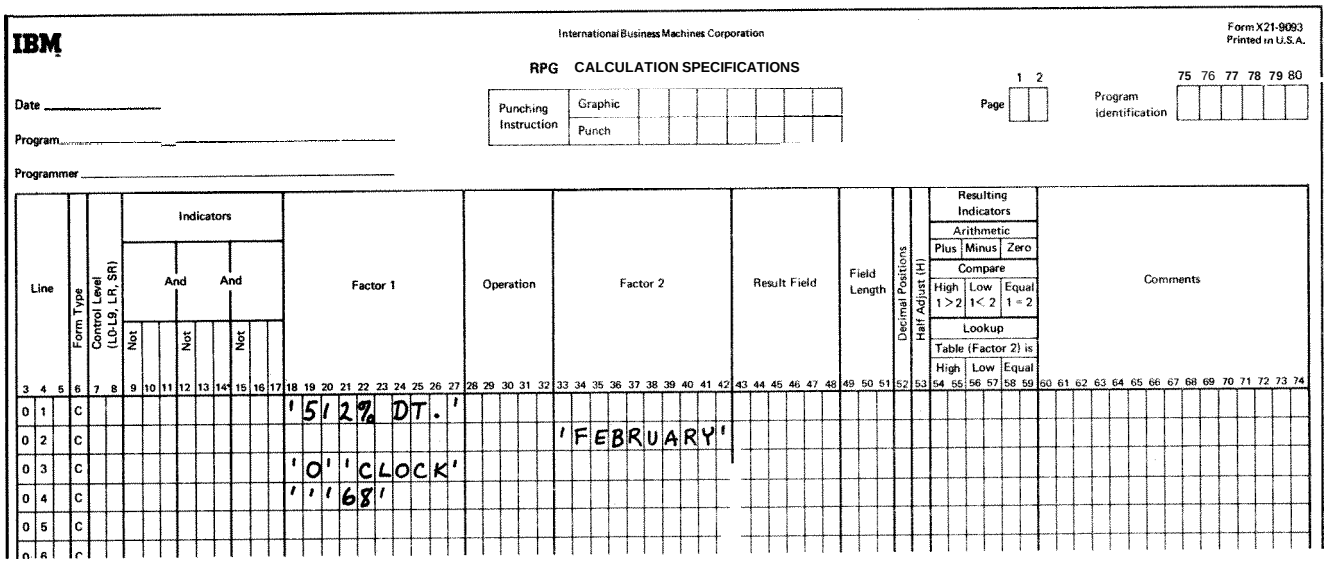
2. The maximum total length of a literal is 10 characters including **signs and** decimal points.

3. Blanks may not appear in the literal.

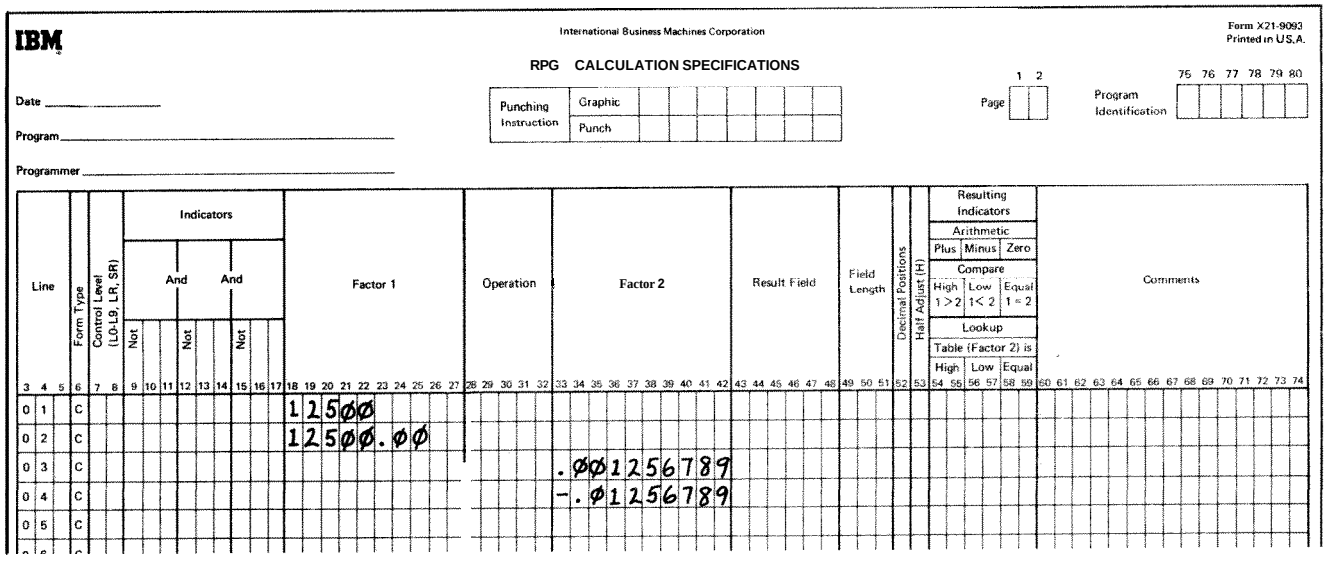
4. The sign, if present, must be the leftmost character. An unsigned literal is treated as a positive number.

5. Numeric literals must not be enclosed by apostrophes (').

6. Numeric literals are used in the same way as a numeric field.



Ⓐ



Ⓑ

Figure 8-9. Alphameric and Numeric Literals

COLUMNS 2832 (OPERATION!

Use columns 28-32 to specify the kind of operation to be performed using Factor 1, Factor 2, and/or the Result Field. The operation code must begin in column 28. A special set of operation codes have been defined which you must use to indicate the type of operation desired. Every operation code used requires certain entries on the same specification line. See Figure 8-10 for a summary of all possible codes and the additional entries required for each code. For further information on the operations that can be performed, see ***Operations Codes*** in Chapter 10.

The operations are performed in the order specified on the Calculation sheet.

All operations conditioned by control level indicators in columns 7-8 (except those which are part of a subroutine) must follow those that are not conditioned by control level indicators.

COLUMNS 4548 (RESULT FIELD)

Entry	*Explanation*
Result field	Field, table, array, or array element.

Use columns 43-48 to name the field, table, array, or array element that will hold the result of the operation specified in columns 28-32. You may use the name of a field, table, array, or array element that has already been defined either in the input specifications or elsewhere in the calculation specifications. (See *Arrays* in Chapter 10 for more information on arrays.)

Otherwise you may define a new field by entering a field name that has not already been used. Any field you define here will be created at the time the program is compiled. The field you name may be either numeric or alphameric. A field used in arithmetic operations (see *Columns 28-32* in this chapter) or numeric compare, or a field edited or zero suppressed in output-format specifications must be numeric.

The result field name must begin with an alphabetic character in column 43 and contain no blanks or special characters.

If you are entering the name of a field that has not been defined elsewhere, columns 49-52 should also contain entries.

If you are entering the name of a field that has been defined, entries in columns 49-52 are not necessary but if specified must agree with the previous definition of that field.

COLUMNS 49-51 (FIELD LENGTH)

Entry	*Explanation*
1-256	Result Field length.

Use columns 49-51 to give the length of a result field that has not been defined previously. If you are naming a new field (one that has not been used before), you must consider the form your data will be in and the length it will have after the operation has been performed.

Whenever the field length is specified for a result field, you should be careful to make the result field long enough to hold the largest possible result. If the result field is too small, significant digits may be lost. For example, you may wish to add field A (eight characters long, four decimal places) to field B (ten characters long, six decimal positions). Fields A and B have four characters to the left of the decimal, but the result field, field C, must allow for more characters to the left of the decimal.

9999.0000	Field A
0001.111111	Field B
10000.111111	Field C (result field)

In this case, Field C was defined as 11 characters long with six decimal positions. Some of the numbers to the right of the decimal could be lost without changing the meaning of the result greatly. However, if field C were defined as 10 characters long with six decimal positions, a significant digit to the left of the decimal would be lost. Field C in this case would be 0000.111111 and the meaning of the result has greatly changed.

Numeric fields have a maximum length of 15 characters. Alphameric fields may be up to 256 characters long. You may indicate the length of a field that has been previously described either in the input specifications or in calculation specifications. However, if you do so, you must specify the same field length and number of decimal positions as was previously given to the field.

If the result field contains the name of a table or array, an entry in these columns is optional. If used, it must agree with the length described in the extension specifications.

COLUMN 52 (DECIMAL POSITIONS)

Entry	*Explanation*
Blank	Alphameric or numeric field described elsewhere.
0-9	Number of decimal places in a numeric result field.

Type of Operation		Operation Code (columns 28-32)	Control Level	Indicators	Factor 1	Factor 2	Result Field	Field Length	Decimal Position	Half Adjust	Resulting Indicators
Arithmetic Operations	Add Factor 2 to Factor 1.	ADD	O	O	R	R	R	O	O	O	O
	Clear Result Field and add Factor 2.	Z-ADD	O	O	B	R	R	O	O	O	O
	Subtract Factor 2 from Factor 1.	SUB	O	O	R	R	R	O	O	O	O
	Clear Result Field and subtract Factor 2.	Z-SUB	O	O	B	R	R	O	O	O	O
	Multiply Factor 1 by Factor 2.	MULT	O	O	R	R	R	O	O	O	O
	Divide Factor 1 by Factor 2.	DIV	O	O	R	R	R	O	O	O	O
	Move remainder of preceding division to a Result Field.	MVR	O	O	B	B	R	O	O	R	O
	Sum elements of an array and put sum in Result Field.	XFOOT	O	O	B	R	R	O	O	O	O
	Derive the square root of Factor 2.	SQRT	O	O	B	R	R	O	O	O	B
Operation	Move Factor 2 into Result Field, right justified.	MOVE	O	O	B	R	R	O	O	B	B
	Move Factor 2 into Result Field, left justified.	MOVEL	O	O	B	R	R	O	O	B	B
	Move zone from low-order position of Factor 2 to low-order position of Result Field.	MLLZO	O	O	B	R	R	O	O	B	B
	Move zone from high-order position of alphameric Factor 2 to high-order of alphameric Result Field.	MHHZO	O	O	B	R	R	O	B	B	B
	Move zone from low-order position of Factor 2 to high-order position of alphameric Result Field.	MLHZO	O	O	B	R	R	O	B	B	B
	Move zone from high-order position of alphameric Factor 2 to low-order position of Result Field.	MHLZO	O	O	B	R	R	O	O	B	B
Compare and Zone Testing Operations	Compare Factor 1 to Factor 2.	COMP	O	O	R	R	B	B	B	B	R
	Identify the zone in the leftmost position of an alphameric Result Field.	TESTZ	O	O	B	B	R	O	B	B	R
Binary Field Operations	Set on specified bits.	BITON	O	O	B	R	R	O	B	B	B
	Set off specified bits.	BITOF	O	O	B	R	R	O	B	B	B
	Test specified bits.	TESTB	O	O	B	R	R	O	B	B	R
Setting Indicators	Set one, two, or three specific indicators on.	SETON	O	O	B	B	B	B	B	B	R
	Set one, two, or three specific indicators off.	SETOF	O	O	B	B	B	B	B	B	R
Branching Within RPG II	Branch to another RPG II calculation specification line.	GOTO	O	O	B	R	B	B	B	B	B
	Identify the name in Factor 1 as a destination label to which GOTO may branch.	TAG	O	B	R	B	B	B	B	B	B
Lookup Operations	Table Lookup.	LOKUP	O	O	R	R	O	O	O	B	R
	Array Lookup.	LOKUP	O	O	R	R	B	B	B	B	R
Subroutine	Beginning of the subroutine.	BEGSR	*	B	R	B	B	B	B	B	B
	End of the subroutine.	ENDSR	*	B	O	B	B	B	B	B	B
	Call to execute the subroutine.	EXSR	O	O	B	R	B	B	B	B	B
Program Control	Forcing record to be read next.	FORCE	B	O	B	R	B	B	B	B	B
	Forcing output printing.	EXCPT	O	O	B	B	B	B	B	B	B
	A field is printed on the printer-keyboard and/or data is entered via the printer-keyboard into a field.	DSPLY	O	O	O	R	O	B	B	B	B
	A record is read from a disk file.	CHAIN	O	O	R	R	B	B	B	B	**
Debug Function	Aid in finding programming errors.	DEBUG	O	O	O	R	O	B	B	B	B

O - Optional
R - Required
B - Blank
* Columns 7-8 must have an SR entry for all subroutine lines.
** See columns 54-59 in this chapter for more information.

Figure **8-10.** Operation Codes

Use column *52* to indicate the number of positions to the right of the decimal in a numeric result field. If the numeric result field contains no decimal positions, enter a *0* (zero).

This column must be left blank if the result field is alphameric. This column may be left blank if the result field is numeric but has been previously described in the input or calculations specifications.

The number of decimal positions must never be greater than the length of the field. The number may, however, be larger or smaller than the number of decimal positions that actually result from an operation. If the number of decimal positions specified is greater than the number of decimal places that actually result from an operation, zeros are filled in to the right. If the number specified is smaller that the number that results from the operation, the rightmost digits are dropped.

Figure 8-11 shows how the contents of a result field after a multiplication operation may change according to the Decimal Positions (column *52)* and Field Length (columns *49-51)* specifications.

COLUMN 53 (HALF ADJUST)

Entry	*Explanation*
Blank	Do not half adjust.
H	Half adjust.

Use column *53* to indicate that the contents of the result field are to be half adjusted (rounded). Half adjusting is done by adding a 5 (-5 if the field is negative) to the number at the right of the last decimal position specified for this field. All decimal positions to the right of the position specified for that field are then dropped (see *Example).*

The half adjust entry is allowed only with arithmetic operations (see *Columns* 28-32 in this chapter).

Example

This example shows a result field being half adjusted to two decimal positions (*2* in column 52 and *H* in column 53).

Multiplication: **98.76** x **1.234** = **121.86984**

Decimal Positions (column 52)	Field Length (columns 49-51)									
	10	9	8	7	6	5	4	3	2	1
9	1.869840000	.869840000								
8	21.86984000	1.86984000	.86984000							
7	121.8698400	21.8698400	1.8698400	.8698400						
6	0121.869840	121.869840	21.869840	1.869840	.869840					
5	00121.86984	0121.86984	121.86984	21.86984	1.86984	.86984				
4	000121.8698	00121.8698	0121.8698	121.8698	21.8698	1.8698	.8698			
3	0000121.869	000121.869	00121.869	0121.869	121.869	21.869	1.869	.869		
2	00000121.86	0000121.86	000121.86	00121.86	0121.86	121.86	21.86	1.86	.86	
1	000000121.8	00000121.8	0000121.8	000121.8	00121.8	0121.8	121.8	21.8	1.8	.8
0	0000000121	000000121	00000121	0000121	000121	00121	0121	121	21	1

Not permitted

Permitted but inaccurate

Recommended

Figure 8-11. Result Field Contents Based on Various Field Length and Decimal Position Specifications

2nd Position
↓

35.7968	Result of an add operation.
5	Add 5 to the number at the right of the last decimal position specified.
35.80XX	Drop all decimal positions to the right at the position specified.
35.80	Result after half adjusting.

COLUMNS 54-59 (RESULTING INDICATORS)

Entry	*Explanation*
01-99	Any two digit number.
H1-H9	Any halt indicator.
L1-L9	Any control level indicator.
LR	Last record indicator.
OA-OG,OV	Any overflow indicator.

Columns 54-59 are used for three different purposes: (1) to test the value of the result field after an arithmetic operation (2) to check the outcome of a CHAIN, LOKUP, COMP, TESTB, or TESTZ operation (see Operation Codes in Chapter 10) and (3) to specify which indicators to set on or off.

Test Results

By entering an indicator in columns 54-59, you specify that the result field is to be tested after the operation specified in columns 28-32 has been performed. (Normally, only indicators 01-99 and H1-H9 are used for testing.) The indicator specified is turned on only if the result field satisfies the condition being tested for. This indicator may then be used to condition following calculations or output operations (see Example). If the same indicator is used to test the result of more than one operation, the operation last performed determines the setting of the indicator.

Notice that three fields (columns 54-55, 56-57, and 58-59) can be used for this purpose. Each field is used to test for different conditions: columns 54-55, plus or high; columns 56-57, minus or low; columns 58-59, zero or equal, You may test for any or all conditions at the same time.

Columns 54-55 (Plus or High): Place an indicator in these columns when testing to find:

1. If the result field in an arithmetic operation is positive.
2. If factor 1 is higher than Factor 2 in a compare operation.
3. If factor 2 is higher than Factor 1 in a table or array lookup operation.
4. The results of a CHAIN, TESTB, or TESTZ operation.

Columns 56-57 (Minus or Low): Place an indicator in these columns when testing the result field to find:

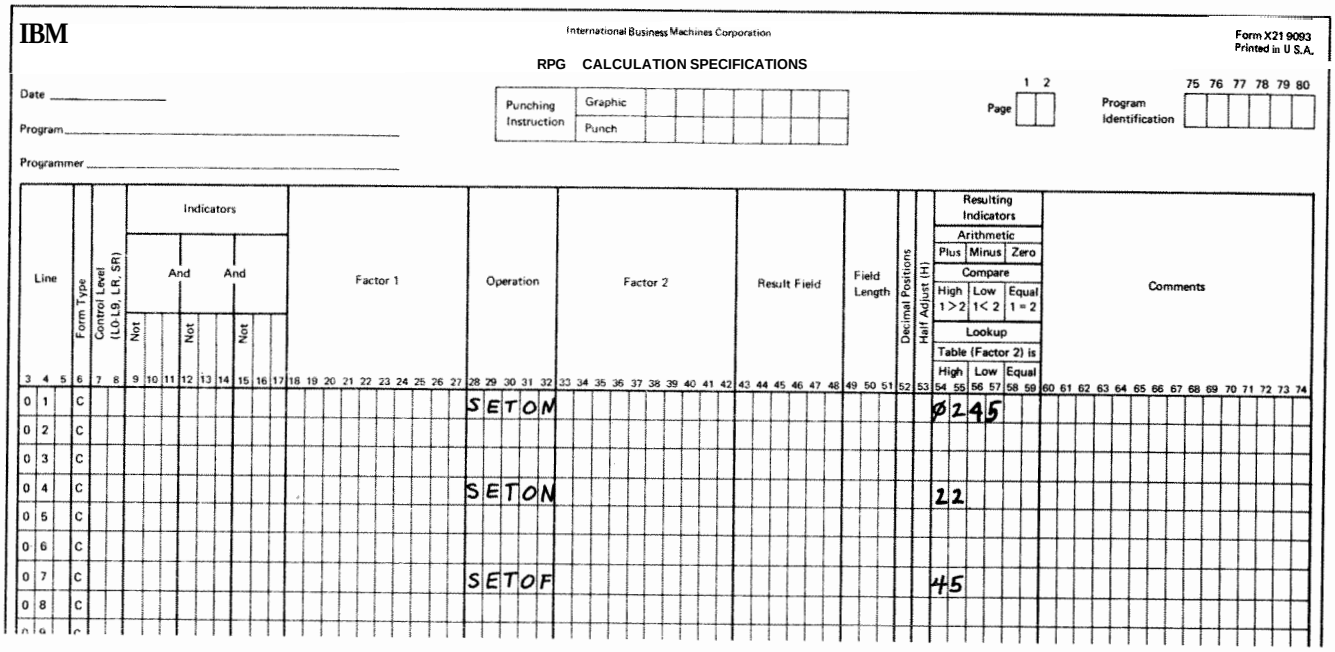
1. If the result field in an arithmetic operation is negative.
2. If factor 1 is lower than Factor 2 in a compare operation.
3. If factor 2 is lower than Factor 1 in a table or array lookup operation.
4. The results of a CHAIN, TESTB, or TESTZ operation.

Columns 58-59 (Zero or Equal): Place an indicator in these columns when testing the result field to find:

1. If the result field in an arithmetic operation is zero.
2. If factor 1 is equal to Factor 2 in a compare operation.
3. If factor 2 is equal to Factor I in a table or array lookup operation.
4. The results of a CHAIN, TESTB, or TESTZ operation.

Setting Indicators

You may enter the indicators that you want to turn on or off by the operations SETON or SETOF. See *Operation* Codes, Setting Indicators in Chapter 10 for more information on these operations. Any indicators to be turned on or off by the SETON or SETOF operations are specified from left to right in the three resulting indicators fields (Figure 8-12). Column headings in columns 54-59 have no meaning for SETON, or SETOF operations.



IBM

International Business Machines Corporation

Form X21 9093
Printed in U S.A.

RPG CALCULATION SPECIFICATIONS

Date _____

Program _____

Programmer _____

Punching Instruction: Graphic / Punch

Page 1 2

Program Identification 75 76 77 78 79 80

Line 3 4 5	Form Type 6	Control Level (L0-L9, LR, SR) 7 8	Indicators: And Not 9 10 11	Indicators: And Not 12 13 14	Indicators: And Not 15 16 17	Factor 1 18-27	Operation 28 29 30 31 32	Factor 2 33-42	Result Field 43-48	Field Length 49 50 51	Decimal Positions 52	Half Adjust (H) 53	Resulting Indicators: Arithmetic Plus 54 55	Resulting Indicators: Arithmetic Minus 56 57	Resulting Indicators: Arithmetic Zero 58 59	Resulting Indicators: Compare High 1>2 54 55	Resulting Indicators: Compare Low 1<2 56 57	Resulting Indicators: Compare Equal 1=2 58 59	Resulting Indicators: Lookup Table (Factor 2) is High 54 55	Resulting Indicators: Lookup Table (Factor 2) is Low 56 57	Resulting Indicators: Lookup Table (Factor 2) is Equal 58 59	Comments 60-74
0 1	C						SETON															
0 2	C																					
0 3	C																					
0 4	C						SETON															
0 5	C																					
0 6	C																					
0 7	C						SETOF															
0 8	C																					

Ø245

22

45

Figure 8-12. Setting Indicators

Example

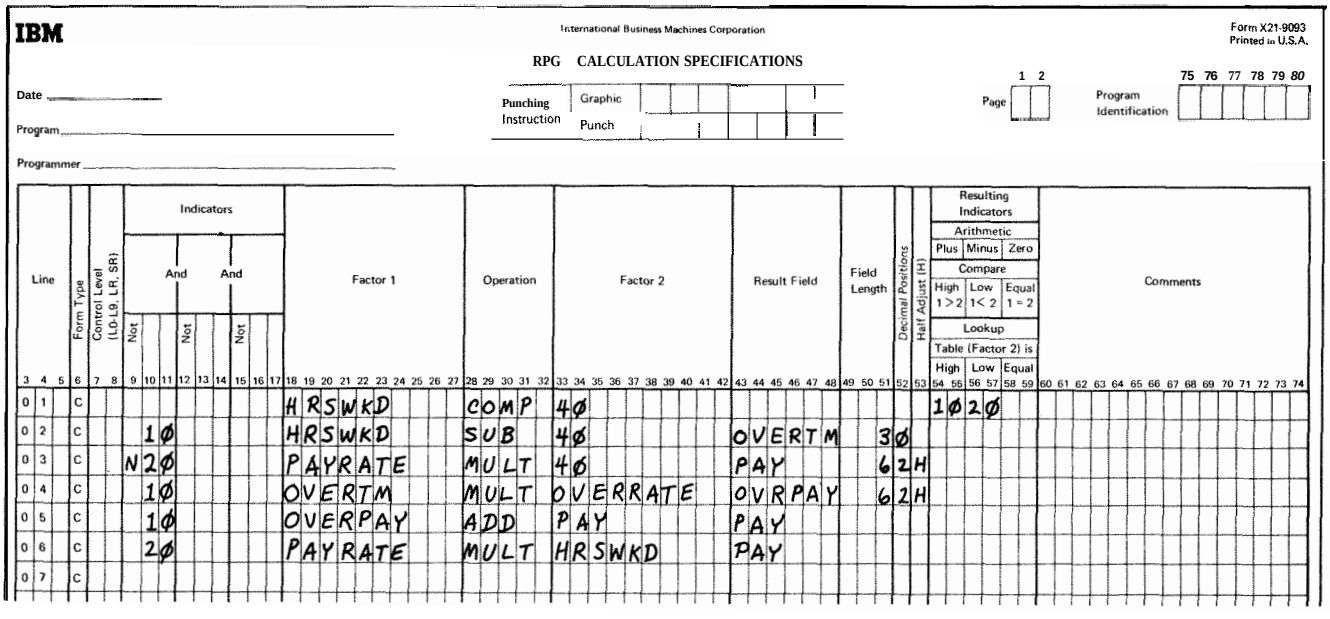
Figure 8-13 shows the entry of two indicators that are used to test for the different conditions in a compare operation. These indicators are used to condition the calculations which must be performed for a payroll job. Indicator 10 is turned on if the hours worked (HRSWKD) are greater than 40 and is then used to condition all operations necessary to find overtime pay. Indicator 20 is turned on if HRSWKD is less than 40. It is also used to condition other operations. In line 03 if 20 is not on (the employee worked 40 or more hours), regular pay based on a 40 hour week is calculated. In line 06 if 20 is on (employee worked less than 40 hours), pay based on less than a 40 hour week is calculated.

COLUMNS 60-74 (COMMENTS)

Enter in columns 60-74 any meaningful information you wish. The comments you use should help you understand or remember what you are doing on each specification line. Comments are not instructions to the RPG II program They serve only as a means of documenting your program.

COLUMNS 7580 (PROGRAM IDENTIFICATION)

See Chapter 2



IBM — International Business Machines Corporation — Form X21-9093 Printed in U.S.A.

RPG CALCULATION SPECIFICATIONS

Date _____ Program _____ Programmer _____

Punching Instruction: Graphic / Punch

Page: 1 2 — Program Identification: 75 76 77 78 79 80

Line	Form Type	Control Level (L0-L9, LR, SR)	Indicators: And / Not	And / Not	Factor 1	Operation	Factor 2	Result Field	Field Length	Decimal Positions	Half Adjust (H)	Resulting Indicators: Arithmetic Plus	Minus	Zero	Compare High 1>2	Low 1<2	Equal 1=2	Lookup Table (Factor 2) is High	Low	Equal	Comments
01	C				HRSWKD	COMP	40											10	20		
02	C		10		HRSWKD	SUB	40	OVERTM		30											
03	C		N20		PAYRATE	MULT	40	PAY		62	H										
04	C		10		OVERTM	MULT	OVERRATE	OVRPAY		62	H										
05	C		10		OVERPAY	ADD	PAY	PAY													
06	C		20		PAYRATE	MULT	HRSWKD	PAY													
07	C																				

Figure 8-13. Conditioning Operations (Resulting Indicators)

Output-Format specifications describe your output records. These specifications may be divided into two general categories:

1. Record description entries (columns 7-31) which describe the output file records to be written or punched.

2. Field description entries (columns 23-74) which indicate the position and the format of data on the output record.

Write the specifications on the Output-Format sheet (Figure 9-1). The field description entries start one line lower than record description entires.

COLUMNS 1-2 (PAGE)

See Chapter 2.

COLUMNS 3-5 (LINE)

See Chapter 2.

COLUMN 6 (FORM TYPE)

An *O* must appear in column 6.

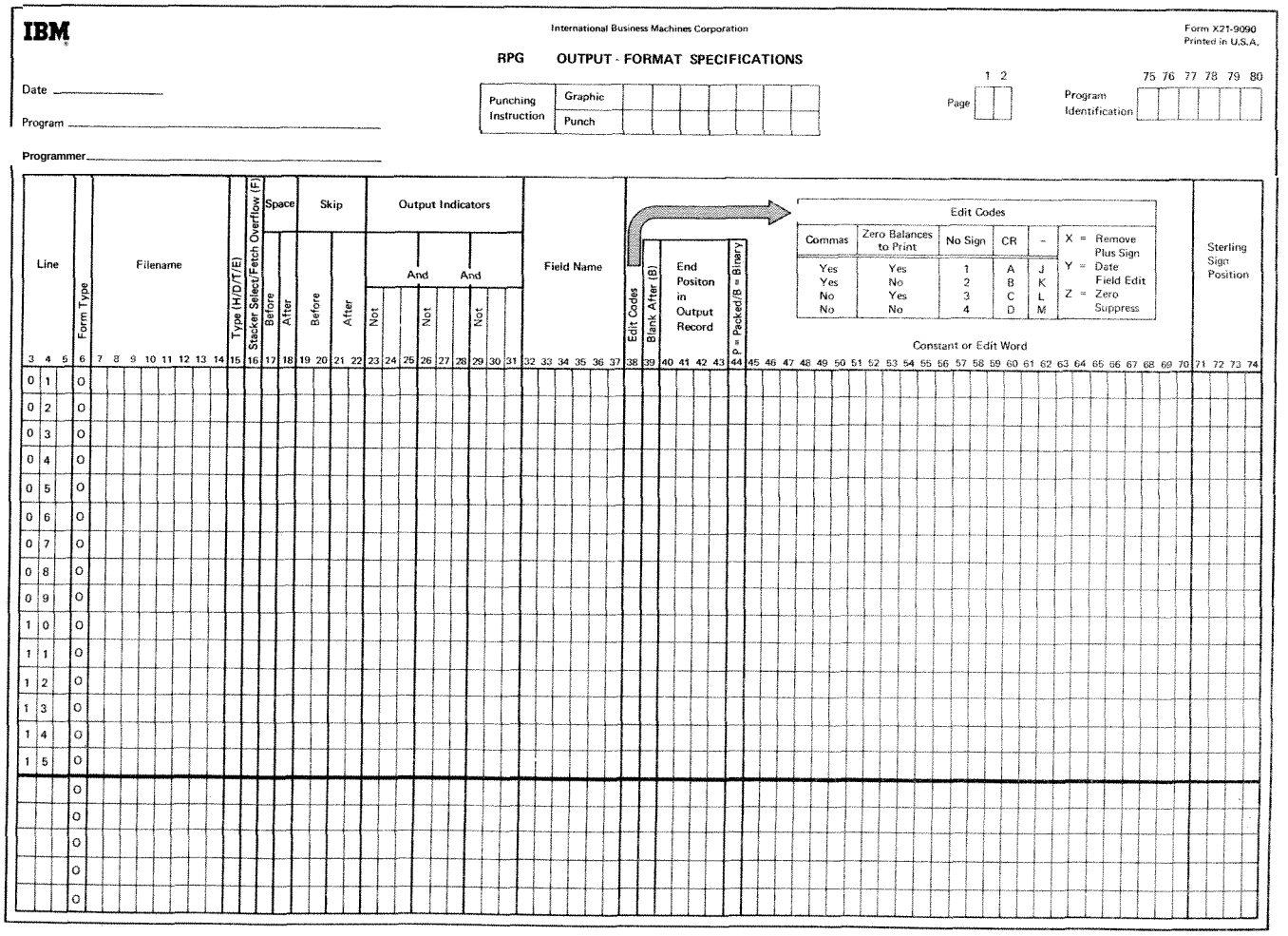


Figure 9-1. Output-Format Sheet

COLUMN 7-14 (FILENAME)

Use columns 7-14 to identify the output file you will be using. The filename must begin in column 7. Use the same filename given in the file description specifications. You need to specify the output filename only once. That name, however, must be on the first line that identifies the file.

COLUMN 15 (TYPE)

Entry	*Explanation*
H	Heading records.
D	Detail records.
T	Total records.
E	Lines to be written during calculation time.

Use column 15 to indicate the type of record that is to be written, This record may be printed, written on disk, or punched or printed on a card. Perhaps the clearest method of describing output files is to enter the records for each file in this order: heading, detail, total, and exception (see Figure 9-2, insert **A).**

Another method is to enter all heading records for all output files, then, all detail records for all output files, etc. The program is compiled faster when records are listed in this manner (see Figure 9-2, insert B).

Heading records usually contain unchanging identifying information such as column headings, as well as page and date.

Detail records are closely connected with input data. Most data in a detail record comes directly from the input record or is the result of calculations performed on data from the input record.

Total records usually contain data that is the end result of specific calculations on several detail records. Total output may not be specified for update files, which are not processed randomly.

Exception records are written or punched during calculation time. This is an unusual case and can be indicated only when the operation code EXCPT is used. *E* may not be specified for a combined file. See *Operation Codes* in Chapter 10 for further information on the EXCPT operation.

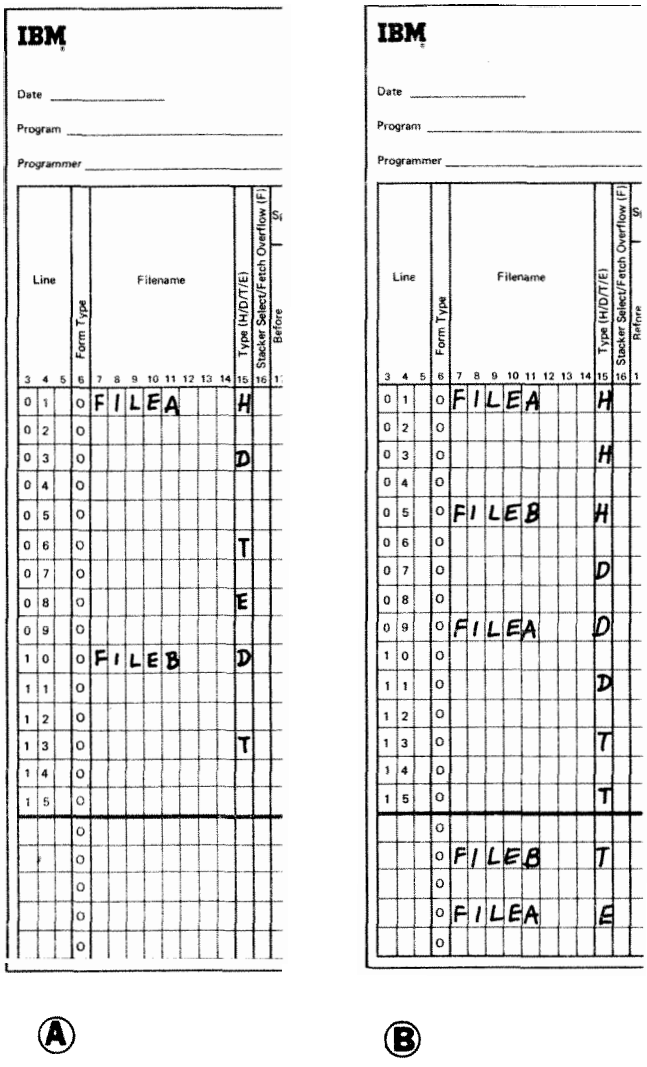


Figure 9-2. Order of Output Record Types

COLUMNS 16-98 (ADD A RECORD)

Entry	*Explanation*
ADD	Add a record.

Columns 16-18 may be used to specify that a record is to be added to an Input, Output, or Update file. The output device for these files must be a disk.

COLUMN 16 (STACKER SELECT/FETCH OVERFLOW)

Entry	*Explanation*
Blank	Cards automatically fall into certain stackers (primary hopper—stacker 1, secondary hopper—stacker 4).
1-4	Indicate stacker you wish.
F	Fetch overflow.

Column 16 may be used for two different purposes:

1. To select a special stacker into which certain cards are to go.
2. To indicate that the overflow routine can be used at this point for a printer file.

Stacker Select

Use column 16 to indicate that certain cards are to be stacked in a specific stacker. If you make no entry, cards go into a predetermined stacker (primary hopper—stacker 1; secondary hopper—stacker 4).

Only combined or output files may be stacker selected in the output-format specifications. If any output operations are to be performed on cards from a combined file that are also to be stacker selected, stacker selection should be done by the output-format specifications not by the input specifications. Stacker selection in output specifications overrides stacker selection in input specifications.

If stacker selection is done on the basis of matching records, it should only be done for detail output (D in column 15). It is only at this time that the MR indicator signals the matching status of the card that should be stacker selected.

OR lines may have different entries in column 16; AND lines may not. An OR line containing a blank in column 16 causes cards to fall into the normal stacker associated with the hopper used. The stacker select entry on the previous line is not assumed.

Fetch Overflow

When the fetch overflow routine is not used, the following usually occurs when the overflow line is sensed:

1. All remaining detail lines in that program cycle are printed (if a printer operation spaced or skipped to the overflow area).
2. All remaining total lines in that program cycle are printed.
3. All lines conditioned by an overflow indicator are printed.
4. Forms advance to a new page if a skip to a new page has been specified.

If you do not want all of the remaining detail and total lines printed on the page before overflow lines are printed and forms advance to the new page, you may cause overflow lines to be printed ahead of the usual time. This is known as fetching the overflow routine and is indicated by the entry in column 16. Overflow is fetched only if all conditions specified by the indicators in columns 23-31 are met and an overflow has occurred. See *Overflow Indicators* in Chapter 10 for detailed information and examples of a fetched overflow routine.

The fetched overflow routine does not automatically cause forms to advance. A skip to line 01 (new page) must also be specified on a line conditioned by the overflow indicator.

F must be used in an OR line if you want that line to condition a record with the overflow indicator.

COLUMNS 17-22 (SPACE/SKIP)

Columns 17-22 are used to specify spacing and line skipping for a printer file. Spacing may be specified for a console file, but not line skipping. If these columns are blank, single spacing occurs automatically after each line is printed.

Line spacing and skipping may be specified both before and after printing of a line. There may be as many as six spaces (three before, three after) between two lines of printing.

If both spacing and skipping are specified on the same line, they are done in this order:

1. Skip before.
2. Space before.
3. Skip after,
4. Space after.

Spacing to or past the overflow line causes the overflow indicator to turn on. Skipping past the overflow line to a line on the next page, however, does not cause the overflow indicator to turn on. If you want to turn on the overflow indicator to condition overflow operations when you skip to a lower line number (higher position) on the next page from a line above the overflow line, you may either use a SETON operation or specify two skips (a skip to the overflow line, then to the first printing line on the next page). This is necessary because the overflow indicator will not be turned on if the skip to a new page occurs on a non-overflow line.

You may save time by specifying that spacing or skipping should be done after printing. This means that the output file does not have to wait for paper movement before it can print.

You may specify different spacing and skipping on OR lines. If there are no spacing or skipping entries in the OR line, spacing and skipping is done according to the entries in the line preceding the OR line.

A zero indicates no movement of the paper. If a zero is indicated for all output lines, the lines will print on top of each other. No spacing may be useful in some cases, however. For example, when you desire two or more output items on the same line but in different positions, you do not specify spacing for one item.

COLUMNS 17-18 (SPACE)

Entry	*Explanation*
0	No spacing.
1	Single spacing.
2	Double spacing.
3	Triple spacing.

Spacing is used in reference to the lines on one page. You may indicate that spacing should be done before (column 17) or after (column 18) a line is printed.

COLUMNS 19-22 (SKIP)

Entry	*Explanation*
0-99	Lines 0-99.
AO-A9	Lines 100-109.
BO-B2	Lines 110-112.

Entries in columns 19-22 must correspond to those entries for the same file on the Line Counter sheet. The skip entry must not be greater than the line number of the overflow line indicated on the Line Counter sheet. Skipping refers to jumping from one printing line to another without stopping at lines in between. This is usually done when a new page is needed. A skip to a lower line number means advance to a new page. Skipping may also be used, however, when a great deal of space is needed between lines. The entry must be the two-digit number which indicates the number of the next line to be printed. You may indicate that skipping should be done before (column 19-20) or after (columns 21-22) a line is printed. If you specify a skip to the same line number as the forms are positioned on, no movement of the paper occurs.

COLUMNS 23-31 (OUTPUT INDICATORS)

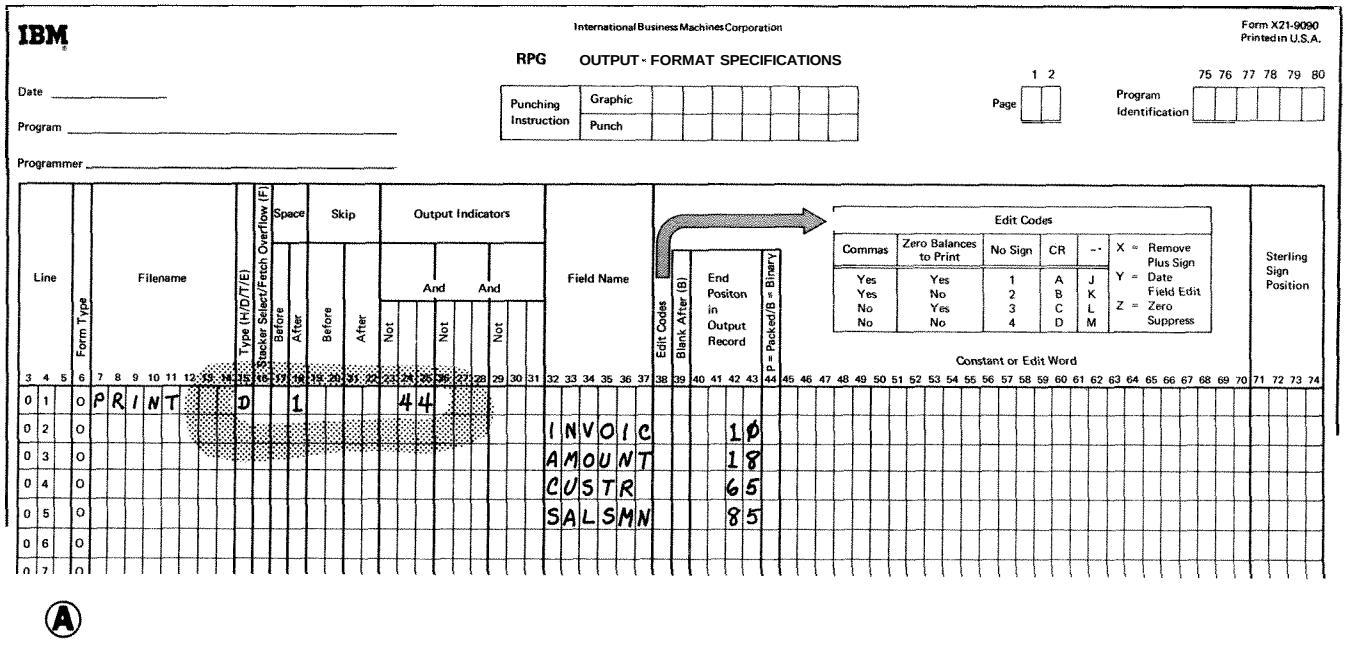
Entry	*Explanation*
01-99	Any resulting indicator, field indicator, or record identifying indicator previously specified.
M-L9	Any control level indicators previously specified.
H1-H9	Any halt indicators previously specified.
U1-U8	Any external indicator set prior to program execution.
OA-OG, OV	Any overflow indicator previously assigned to this file.
MR	Matching record indicator.
LR	Last record indicator.
1P	First page indicator.

Use output indicators to give the conditions under which output operations are to be done. More specifically, use them to tell:

1. When you want to output a line (see *Examples, Example* 1).
2. When you want to output a field (see *Examples, Example* 2).

When you use an indicator to condition an entire line of print, place it on the line which specified the type of record (see Figure 9-3, insert **A).** Place an indicator which conditions when a field is to be printed on the same line as the field name (see Figure 9-3, insert B).

There are three separate output indicator fields (columns 23-25, 26-28, and 29-31). One indicator may be entered in each field. If these indicators are on, the output operation will be done. **An** *N* in the column (23, 26, or 29) preceding each indicator means that the output operation will be done only if the indicator is not on. No output line may be conditioned by all negative indicators (at least one of the indicators used must be positive).



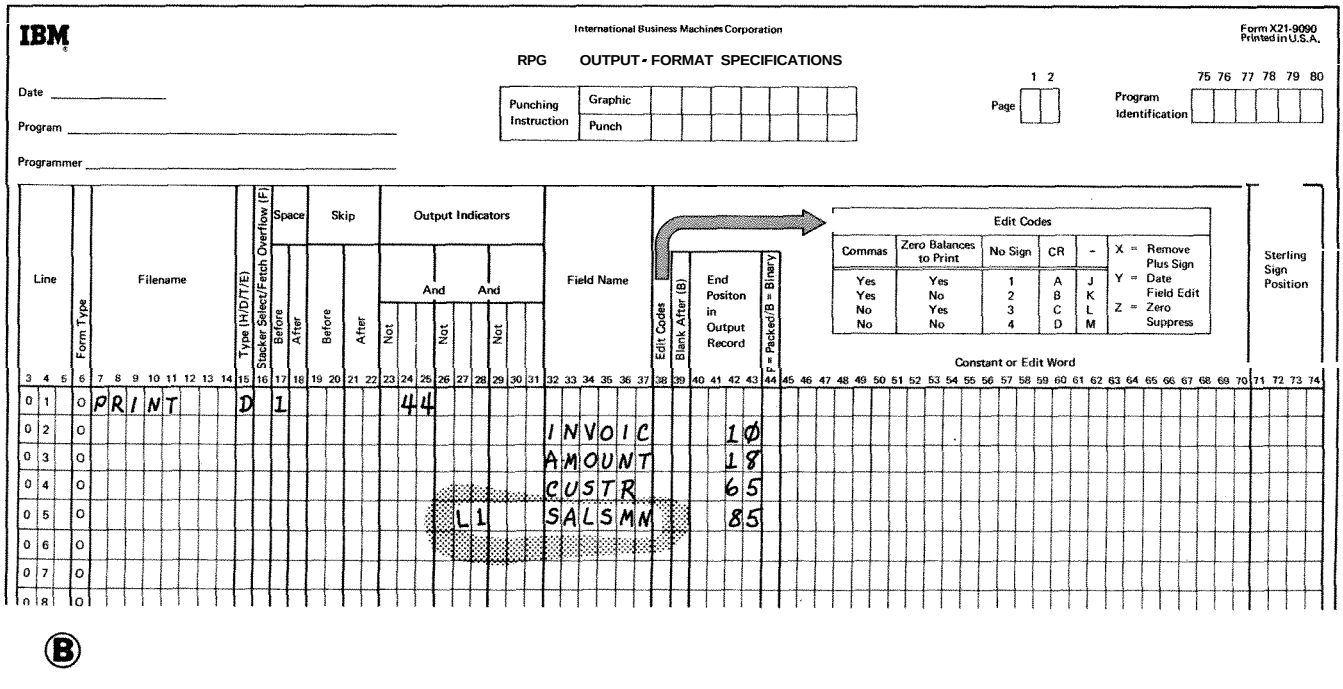


Figure 9-3. Output Indicator

AND and OR tines

If you need to use more than three indicators to condition an output operation, you may use an AND line. Enter the word *AND* in columns 14-16 and as many indicators as needed. The condition for all indicators in an AND relationship must be satisfied before the output operation is done. There is no limit to the number of AND lines that can be used for an output operation.

Output indicators may also be in an OR relationship. If one or the other condition is met, the output operation will be done. OR lines are indicated by the word OR in columns 14-15. A maximum of twenty OR lines may be used for an output operation. Both AND or OR lines may be used together to condition an entire output line. They may not be used, however, to condition a field (see ***Examples, Example 3).***

External Indicators

A file named in the output-format specifications may be conditioned by an external indicator in the file description specifications. In this cage, every output record for that file must be conditioned by the same external indicator used in the file description specifications.

Overflow Indicators

Overflow indicators are used to condition output operations on the printer. The operations conditioned by the overflow indicator are done only after the overflow line (end of page) has been reached.

If you have not assigned an overflow indicator to the printer file in the file description specifications, you may not use an overflow indicator in the output-format specifications. In this case, advancing the forms to a new page is handled automatically, even though no overflo'w indicator has been assigned. If any specification line not conditioned by an overflow indicator specifies a skip to a line on a new page, overflow indicators turn off before forms advance to a new page.

An overflow indicator may appear on either AND or OR lines. However, only one overflow indicator may be associated with one group of output indicators. That overflow indicator must also be the same indicator associated with the file on the File Description sheet.

When the overflow indicator is used in an AND relationship with a record identifying indicator, unusual results are often obtained. This is because the record type might not be the one read when overflow has occurred. Thus, the record type indicator is not on and all lines conditioned by both overflow and record type indicators do not print.

If at all possible, use overflow indicators and record type indicators in an OR relationship when conditioning output lines.

An overflow indicator cannot condition an exception line (E in column 15), but may condition fields within the exception record.

First Page Indicator

The first page (1P) indicator is usually used to allow printing on the first page. It may also be used in connection with the overflow indicator to allow printing on every page (see ***Examples, Example*** 4). The information printed out on the line conditioned by the 1P indicator is usually constant information used as headings. The constant information is specified on the Output-Format sheet.

The 1P indicator is used only with heading or detail output lines. It cannot be used to condition total or exception output lines. Use this indicator only when other indicators (control level or resulting indicators) cannot be used to control printing on every page.

The 1P indicator cannot be used in an AND or OR relationship with control level indicators.

Error Conditions

On certain error conditions, you may not want output performed. Indicators can be used to prevent the data that caused the error from being used (see ***Examples, Example*** 5).

Examples

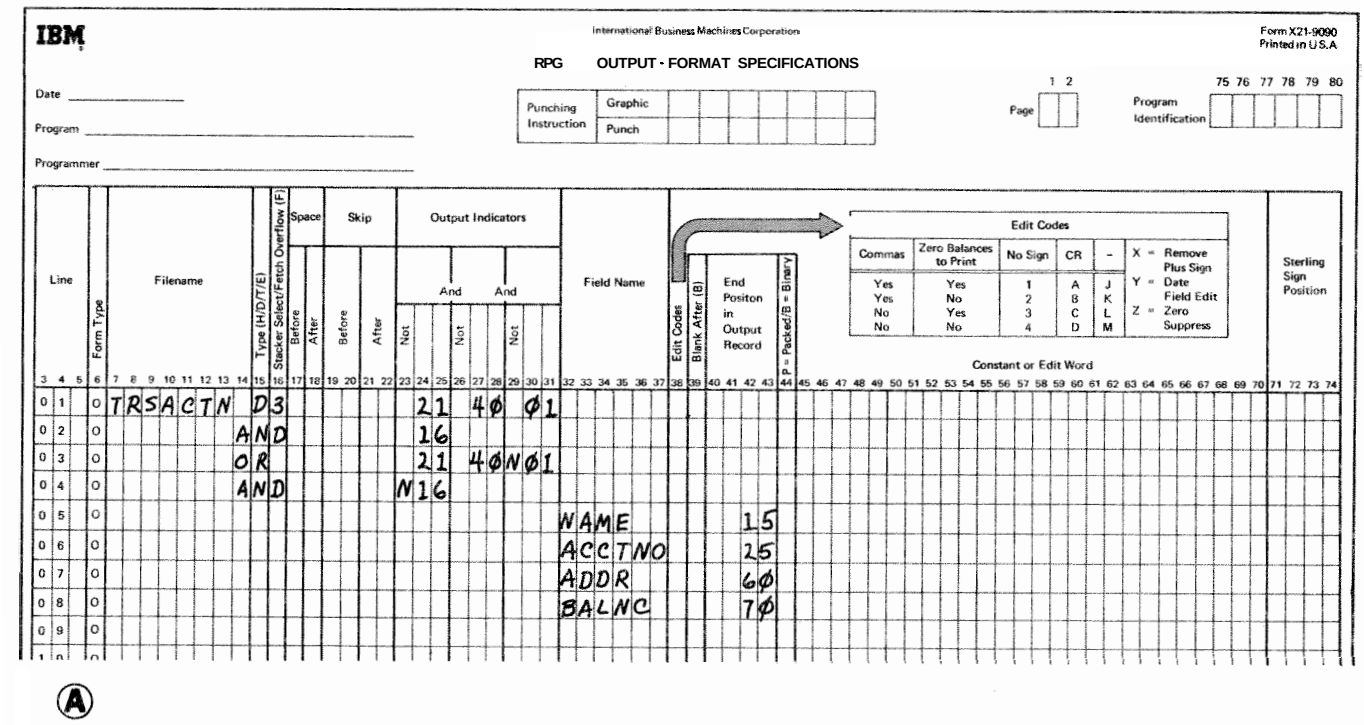
Example 1: Figure 9-3, insert A, shows the use of one indicator to condition an entire line of printing. When 44 is on, the fields named INVOIC, AMOUNT, CUSTR, and SALSMN are all printed.

Example 2: Figure 9-3, insert B, shows the use of a control level indicator to condition when one field should be printed. When indicator 44 is on, fields INVOIC, AMOUNT, and CUSTR are always printed. However, SALSMN is printed only if 44 and L1 are on.

Example 3: The use of indicators in both AND and OR lines to condition an output line is shown by Figure 9-4, insert A. The specifications in lines 01-04 say that the detail line is printed if either one of two sets of conditions is met. If 21, 40, 01, and 16 are all on, the line is printed, or if 21 and 40 are on and 01 and 16 are off, the line is also printed.

A maximum of three indicators may be used on the Output-Format sheet to condition a field since AND and OR lines may not be used to condition an output field (see Figure 9-4, insert B).

However, you can condition an output field with more than three indicators by using the SETON operation in calculations. For instance, indicators 10, 12, 14, 16 and 18 are to condition an output field named PAY. In calculation specifications, you can SETON indicator 20 if indicators 10, 12, and 14 are on. Then condition the output field PAY on indicators 20, 16, and 18 on the Output-Format sheet.



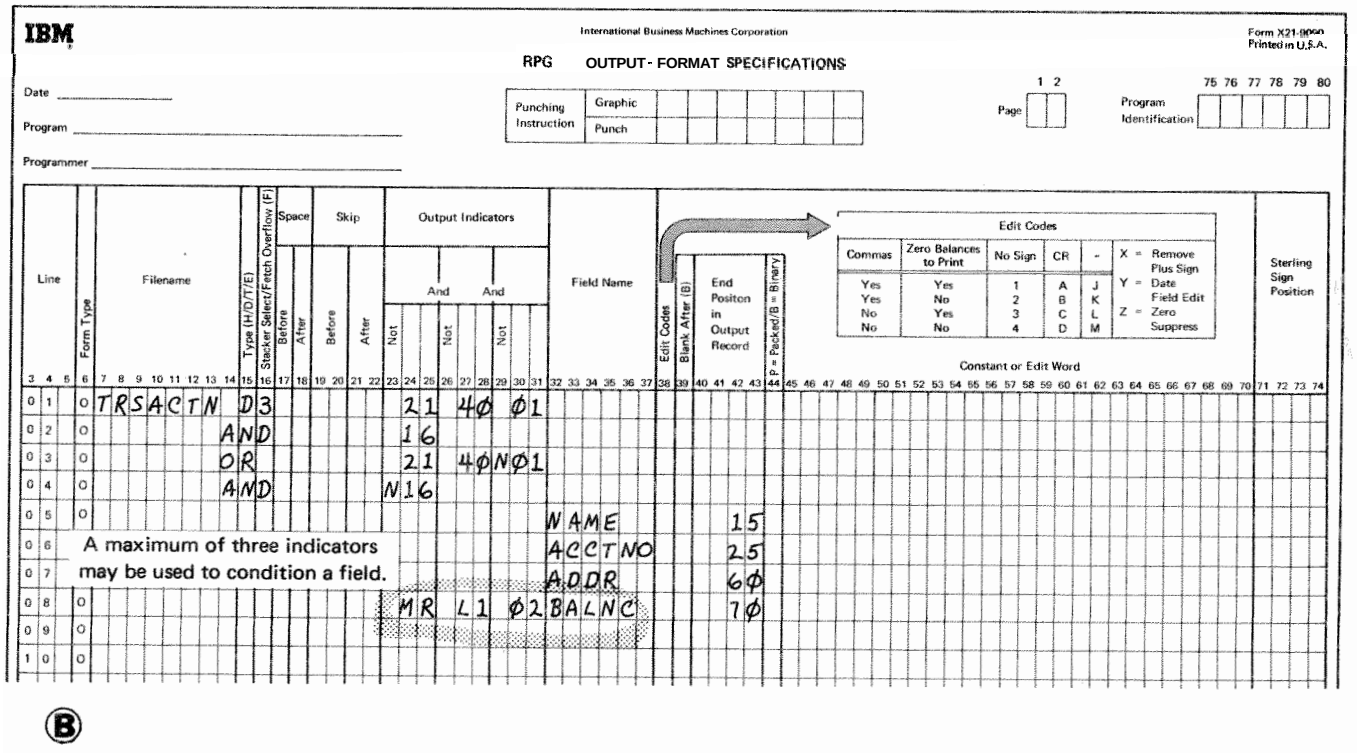
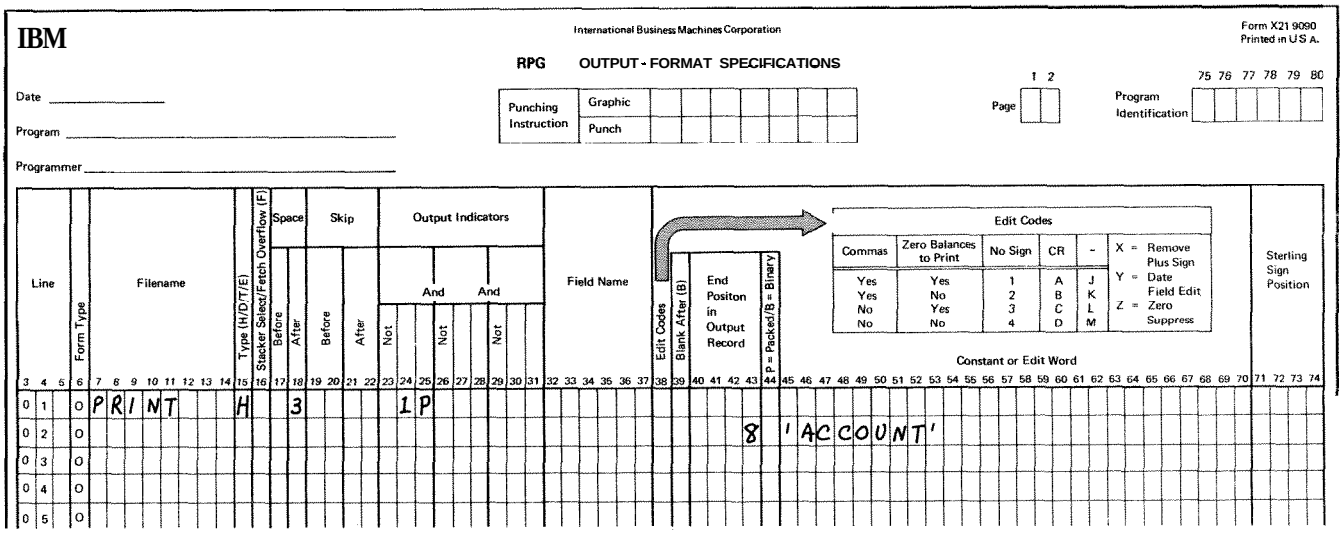


Figure 9-4. **Output Indicators**

Example 4: Figure 9-5, insert **A,** shows how the 1P indicator is used when headings are to be printed on the first page only. Figure 9-5, insert B, shows the use of the 1P indicator and overflow indicator to print headings on every page.

Example 5: Figure 9-6 shows coding necessary to check for an error condition and to stop processing on and printing from the card in error. If FIELDB contains all zeros, halt indicator H1 turns on (see line 03 of Figure 9-6, insert A). In the calculation specifications, if H1 is on, resulting indicator 01 turns off (see line 01 of Figure 9-6, insert B). On the output-format specifications, FIELDA and FIELDB are printed only if 01 is on (see lines 04 and 05 of Figure 9-6, insert C). Therefore, if indicator 01 is off, fields **A** and B are not printed. Use this general format when you do not want information that is in error to be printed.



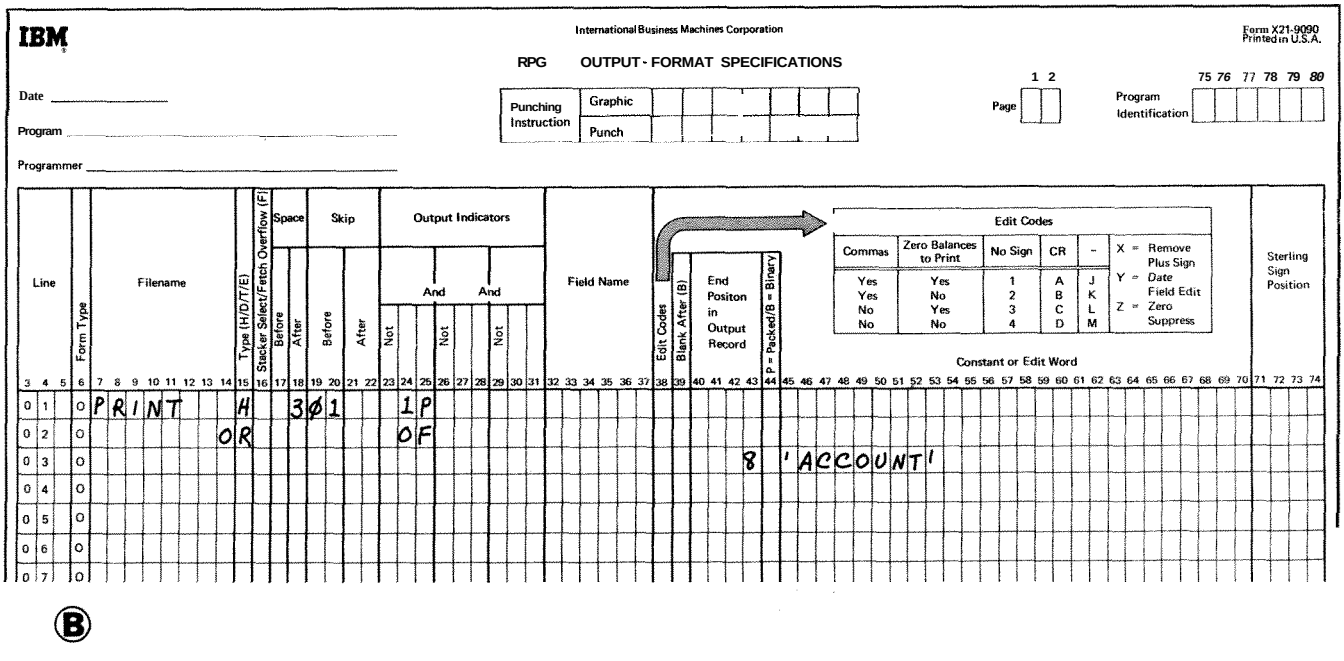
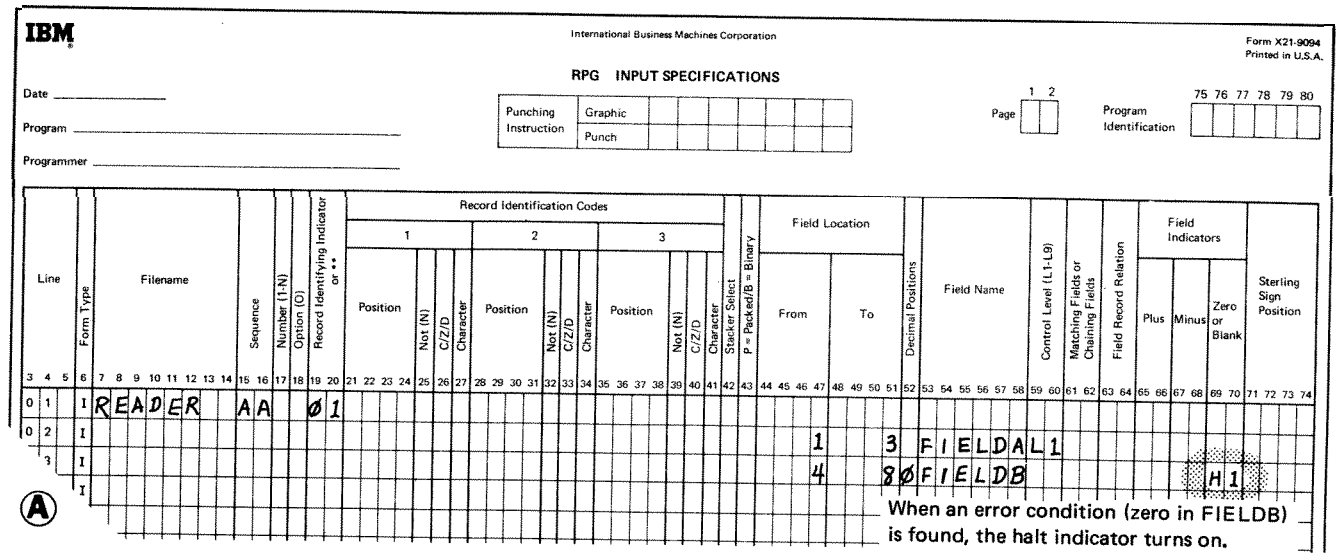
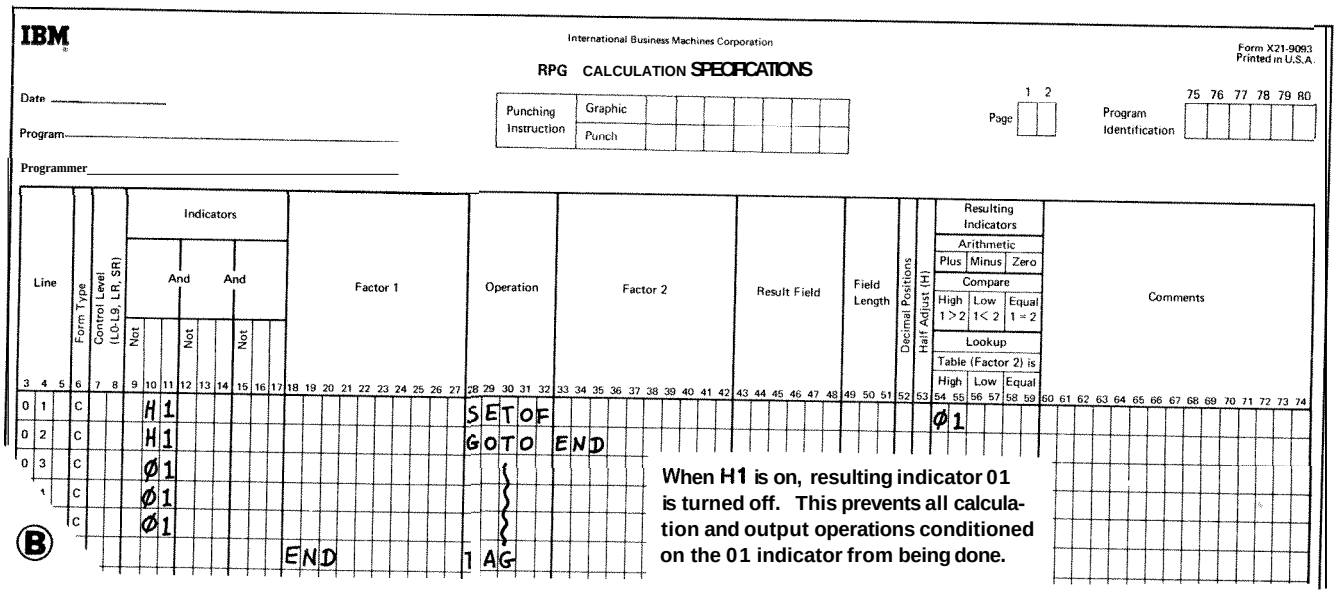


Figure 9-5. 1P Indicator





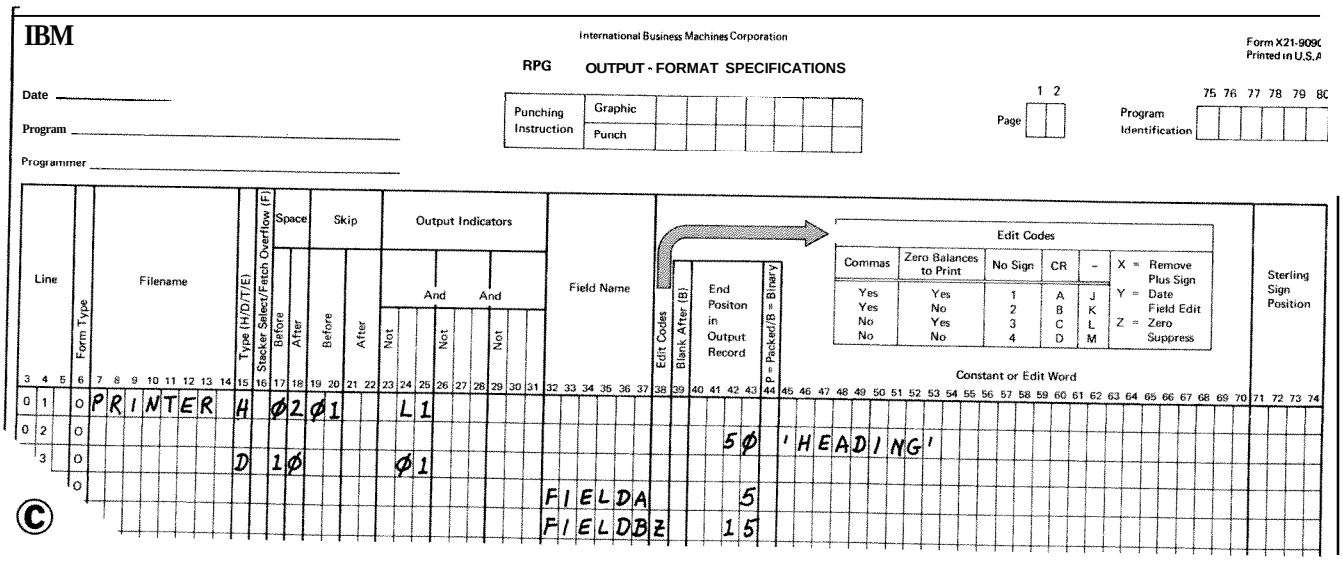


Figure 9-6. **Preventing Fields From Printing**

COLUMNS 32-37 (FIELD NAME)

In columns 32-37, use one of the following to name every field that is to be written out.

1. Any field name previously used in this program.

2. The special words PAGE, PAGE1, PAGE2, *PLACE, *PRINT, UDATE, UDAY, UMONTH, and UYEAR.

3. A table name, array name, or array element.

The field names used are the same as the field names on the Input sheet (columns 53-58) or the Calculation sheet (columns 43-48). Do not use these columns if a constant is used (see *Columns 45-70* in this chapter). If a field name is entered in columns 32-37, columns 7-22 must be blank.

Fields may be listed on the sheet in any order since the sequence in which they appear on the printed form is determined by the entry in columns 40-43. However, they are usually listed sequentially. If fields overlap, only the last field specified is printed.

The sign (+ or -) of a numeric field is in the units position (rightmost digit). Either sign (+ or -) in the units position prints as a letter unless the field is edited (see *Editing* in Chapter 10 or *Column 38* in this chapter).

PAGE: PAGE is a special word, which, when used, causes automatic numbering of your pages. Enter the word PAGE, PAGE1, or PAGE2 in these columns if you wish pages to be numbered. When a PACE field is named in these columns without being defined elsewhere, it is assumed to be a four column, numeric field with no decimal position. Leading zeros are suppressed and the sign is printed in the rightmost position unless an edit word or edit code is specified. The page number starts with 1 unless otherwise specified, and 1 is automatically added for each new page. See *Columns 53-58* in Chapter 7 for information concerning page numbering starting at a number other than 1.

It is possible at any point in your job to restart the page numbering sequence. To do this set the PACE field to zero before it is printed. One method of setting the PAGE field to zero is to use Blank After (see *Column* 39 in this chapter). Another way is to use an output indicator. If the status of the indicator is as specified, the PACE field is reset to zero. Remember that 1 is added to the PAGE field before it is printed (see *Examples, Example 1*).

The three possible PAGE entries: PACE, PAGE1, and PAGE2 may be needed for naming different output files. Do not use the same name for two different output files.

Note: A PAGE field named only in output specifications must be four characters long, and need not be defined elsewhere. However, a PAGE field can be defined in input or calculation specifications and may be of any length. Despite the difference in length, these PAGE fields are treated exactly as if they were named in output specifications only.

*PLACE: "PLACE is a special RPG II word which makes it possible to write or punch the same field in several locations on one record without having to name the field and give its end position each time the field is to be written or punched. The fields are written or punched in the same relative positions ending in the column specified by *PLACE. For example, if you wish FIELDS A, B, and C to appear twice on one line, you can specify this in two ways:

1. Define each field and its corresponding end position each time it is to be printed (Figure 9-7, insert A).

2. Use the special word "PLACE (see Figure 9-7, insert B).

Both coding methods produce a line which looks like this:

```
(Print Positons) 10      20      30      40      50      60
(Fields)  FIELDA FIELDB FIELDC FIELDA FIELDB FIELDC
```

However, it is easy to see that using the special word *PLACE saves extra coding.

When using *PLACE, all Gelds named for each record type (H/D/T/E) are written or punched as usual in the locations specified. The entry *PLACE then causes all of these same fields to be written or punched ending at the position specified in the *PLACE statements.

When using "PLACE, remember:

1. *PLACE must be specified after the field names which are to be placed in different positions in one line (see *Examples, Example* 2).

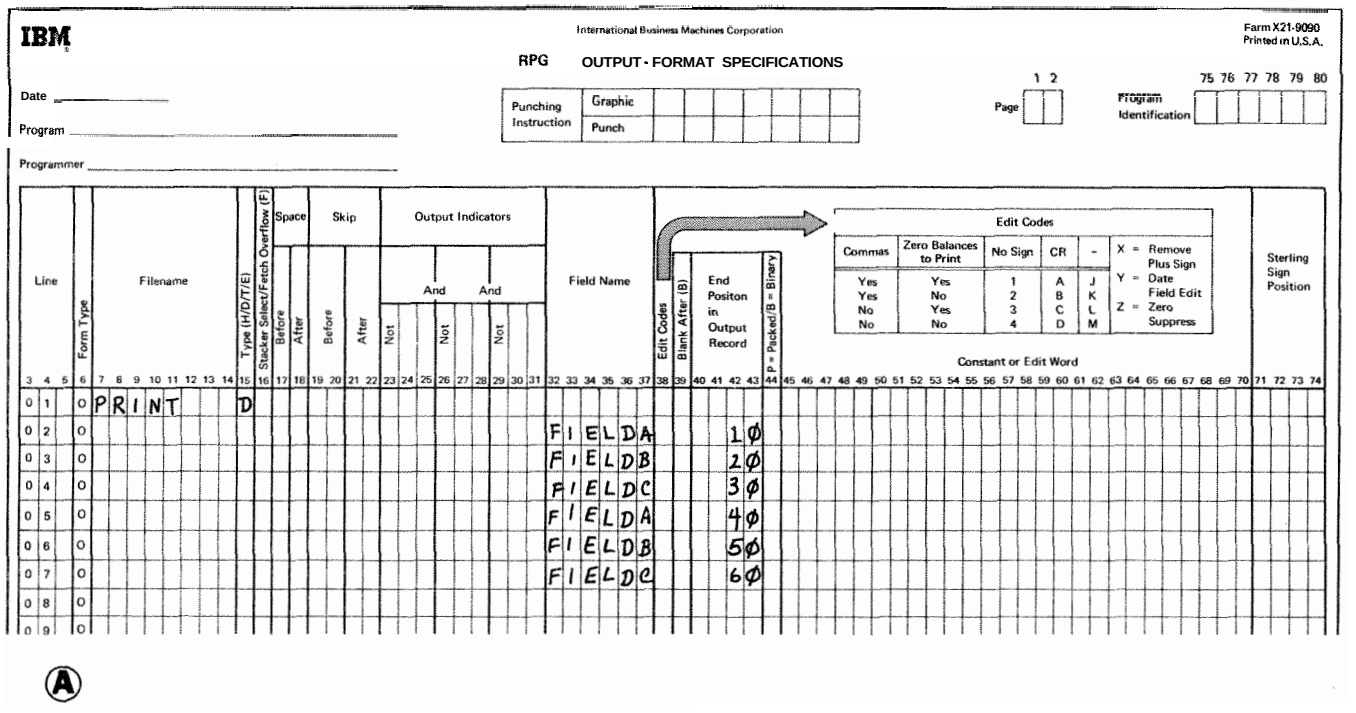
2. *PLACE causes *all* fields (in *a* record type) above the *PLACE entry to be written or punched, not just the field named on the line above "PLACE.

3. *PLACE must appear on a separate specification line for every additional time you want the field or group of fields written or punched.

4. An end position must be specified for every *PLACE line. Be sure to allow enough space for all fields prior to the ***PLACE** to be printed again (see ***Examples,*** *Example* 2). Otherwise overlapping occurs.

5. The leftmost position of the fields to be moved by the ***PLACE** specification is always assumed to be position 1.

6. When ***PLACE** is specified for card output, the fields named above **"PLACE** 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.**



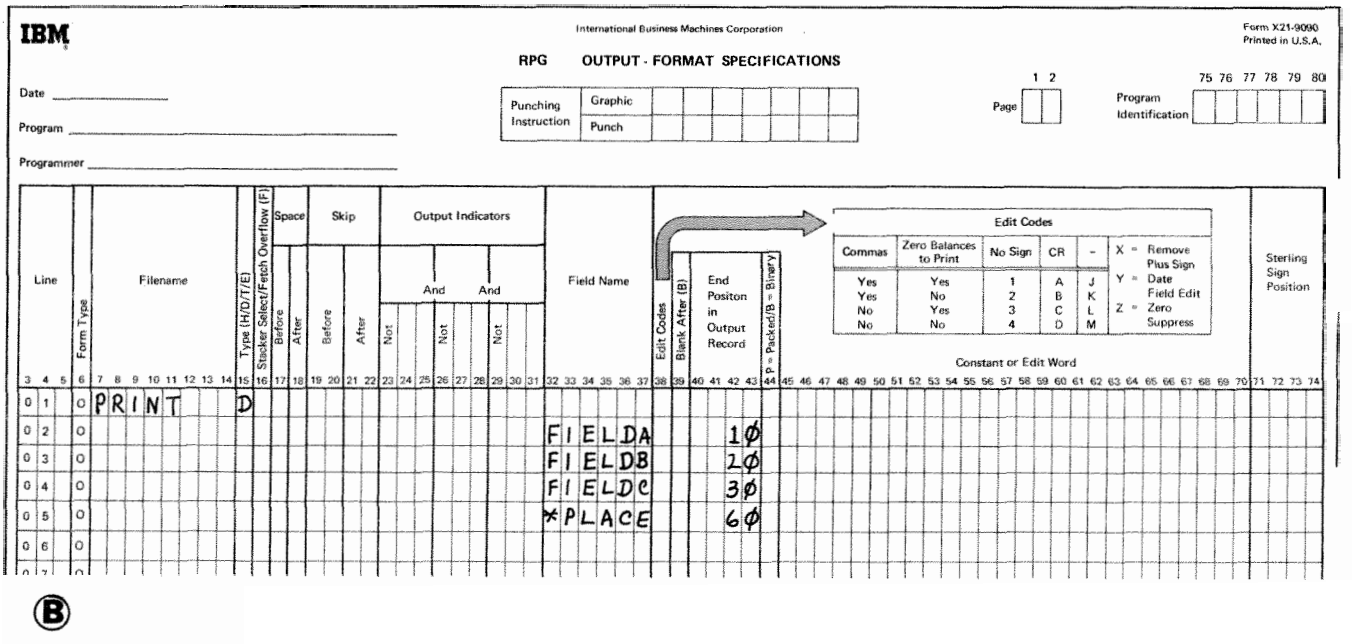


Figure 9-7. Printing Fields Twice **on the Same** Line

**PRINT:* *PRINT is a special RPG II word which causes fields that were punched in the card to be printed on the card. This enables you to more easily determine what information is found on the card. *PRINT prints the field in the positions which correspond one-for-one to the columns in which the field is punched (see *Examples, Example 3).*

When using "PRINT, remember:

1. *PRINT may be used only once for each record.
2. "PRINT must be specified after all punch fields which are to be printed on the card are named.
3. The *PRINT specification may be conditioned by indicators in columns 23-31. Columns 7-22 and 38-74 may not be used.
4. *PRINT may be used on a card file only.

If you want to print the fields in positions other than those which correspond to the punch positions of the fields, you must use the card printing option (see *Columns 40-43* in this chapter).

Date Field: Often you want the date to appear on your printed report, punched card, or program listing. Use special words UDATE, UMONTH, UDAY, and UYEAR to get the date field you desire. The following rules apply to date fields:

1. UDATE gives a six-character numeric date field in one of three formats:

   a. Domestic (rnrnddyy).
   b. United Kingdom (ddmmyy).
   c. World Trade (ddmmyy).

   The format is specified by an entry in *Column* 21 of the control card.

2. UDAY may be used for days only, UMONTH for months only, and WEAR for years only.
3. These fields may not be changed by any operations specified in the program. Thus, these fields are usually used only in compare and test operations.

Examples

Example 1: Figure 9-8 shows how an output indicator can be used to reset a PAGE field to zero. When indicator 15 is on, the PAGE field is reset to zero and a 1 added before the field is printed. When 15 is off, a 1 is added to the contents of the PAGE field before it is printed.

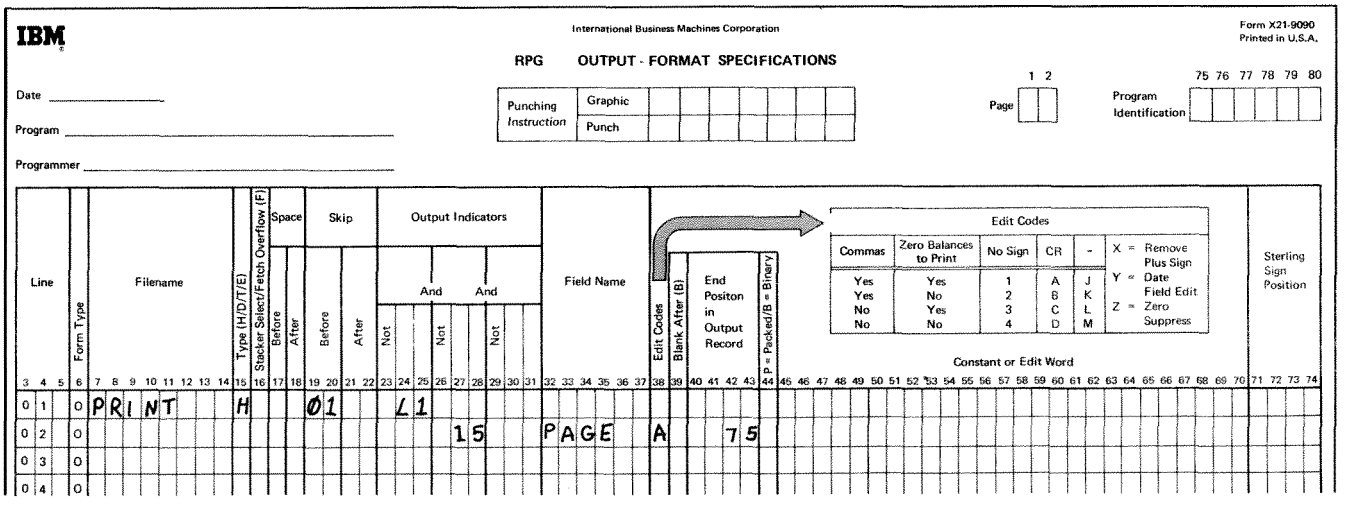
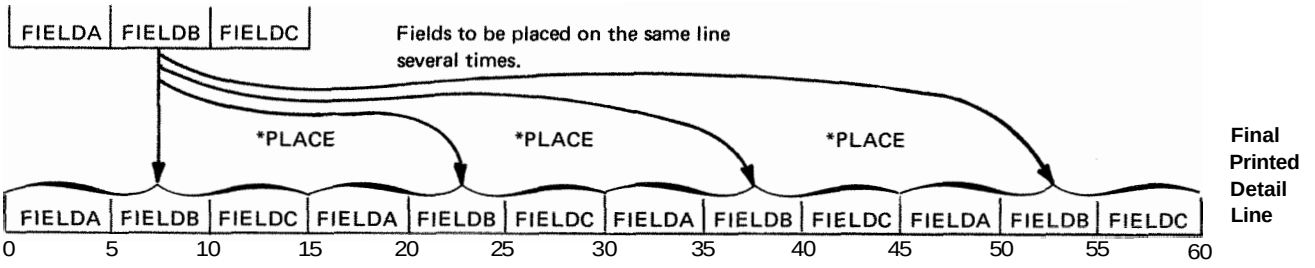


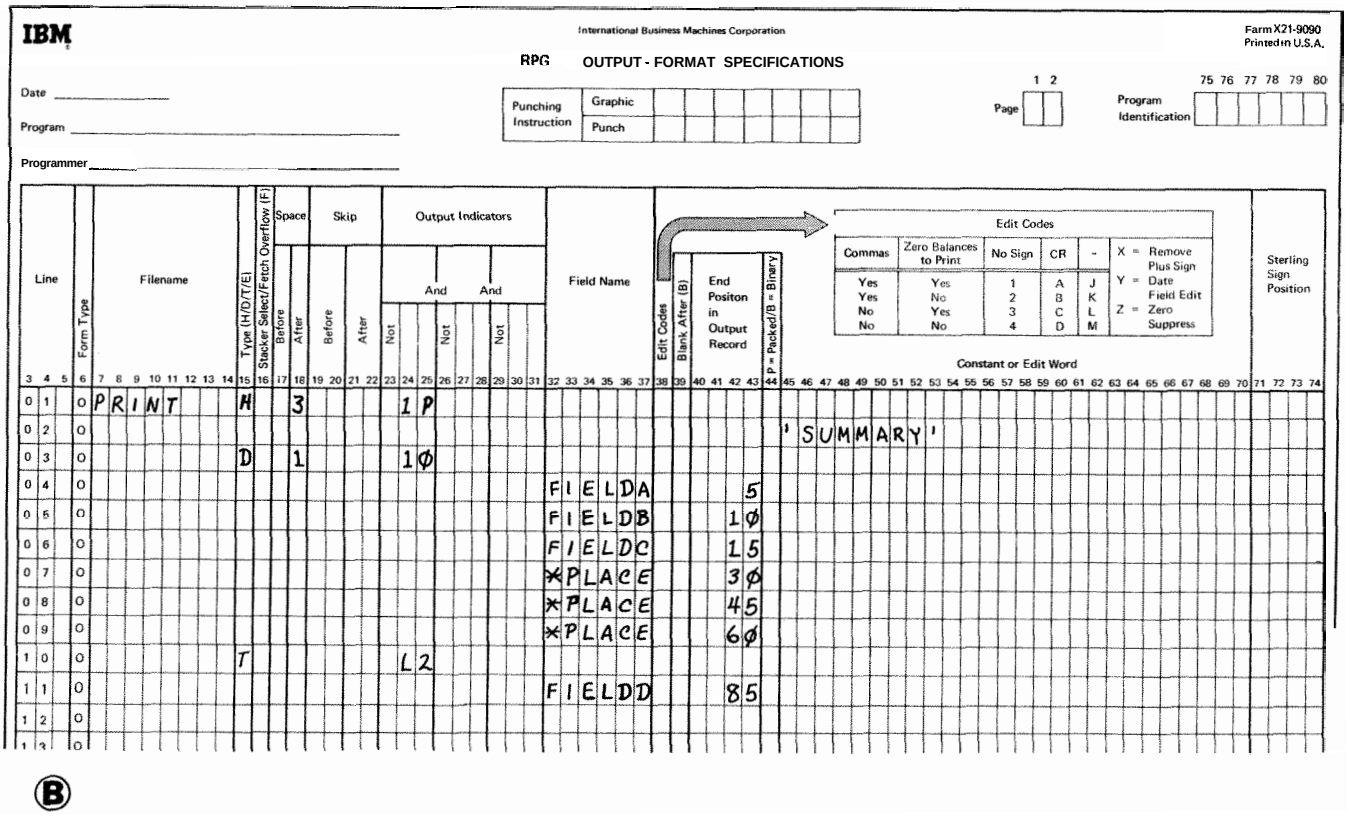
Figure 9-8. Resetting the PAGE Field to Zero

Example 2: Figure 9-9 shows the use of the special word ***PLACE** to print the same fields several times on the same line. Fields **A,** B, and **C** are to be printed four times on one line (see Figure 9-9, insert **A).** They are printed once when they are named and once for every **"PLACE** entry. In Figure 9-9, insert B, *PLACE is specified after the fields which are to be printed several times on the same line. All fields to which **"PLACE** applies appear on the same record. Field D, which appears on the total record, is not affected by ***PLACE.**

Notice also that an end position is given for every ***PLACE.** Fields **A, B,** and **C** have a total length of 15 characters. Thus the end positions given for the **"PLACE** entries all allow room for the printing of 15 characters. This eliminates anv overlapping.



(A)

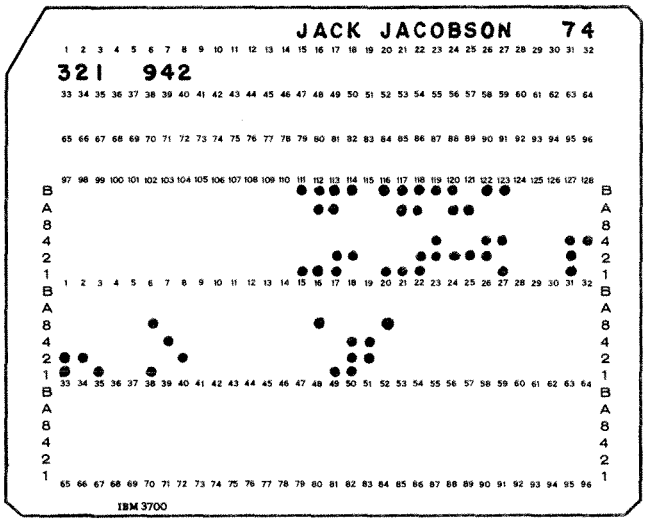


(B)

Figure 9-9. *PLACE

Example 3: Figure 9-10 shows how the special word *PRINT may be used to cause printing of the output fields on the punched cards. The fields EMPLYE, SERNUM, and PAYRT are to be punched on the card (specification lines 05-07). The *PRINT entry in line 08 causes the three fields written above the *PRINT entry (EMPLYE, SERNUM, and PAYRT) to print on the card in positions corresponding one-for-one to the punch positions (see Figure 9-10). The UDATE field (line 09) is punched but not printed because it is written after the *PRINT entry.

Notice in Figure 9-10 that *PRINT is specified after the fields which are to be printed. All fields to which *PRINT applies appear on the same record. Therefore, the *PRINT entry applies only to fields specified in lines 05-07, not to fields specified in lines 02 and 03.



IBM

International Business Machines Corporation

Form X21-9090
Printed in U.S.A.

RPG OUTPUT - FORMAT SPECIFICATIONS

Date _____

Program _____

Programmer _____

Punching Instruction	Graphic							
	Punch							

Page 1 2

Program Identification 75 76 77 78 79 80

Line	Form Type	Filename	Type (H/D/T/E)	Stacker Select/Fetch Overflow (F)	Space Before	Space After	Skip Before	Skip After	Output Indicators And Not	Output Indicators And Not	Output Indicators And Not	Field Name	Edit Codes	Blank After (B)	End Positon in Output Record	P = Packed/B = Binary	Sterling Sign Position
01	O	CARDS	D	1					Ø1								
02	O											EMPLYE			3Ø		
03	O											DEPT			4Ø		
04	O		D	2					Ø2								
05	O											EMPLYE			3Ø		
06	O											SERNUM			35		
07	O											PAYRT			4Ø		
08	O											*PRINT					
09	O											UDATE			52		
10	O																

Edit Codes

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

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

Constant or Edit Word

Figure 9-10. *PRINT

COLUMN 38 (EDIT CODES)

Use column 38 when you want to:

1. suppress leading zeros
2. omit a sign from the low order position of a numeric field
3. punctuate a numeric field without setting up your own edit word.

A table summarizing the edit codes that can be used is printed above columns 45-70 on the Output-Format sheet (see Figure 9-10). Each edit code punctuates differently. If you use an edit code in column 38, columns 45-70 must be blank except for the following condition. If asterisk fill or a floating dollar sign is required, enter '*' or '$' in column 45-70. When an edit code is used to punctuate an array, two spaces are left between fields of the array to the left of each element. Only numeric fields can be edited. For more information on edit codes, see *Editing* in Chapter 10.

COLUMN 39 (BLANK AFTER)

Entry	*Explanation*
Blank	Field is not to be reset.
B	Field specified in columns 32-37 is to be reset after the output operation is complete.

Use column 39 to reset a field to zeros or blanks. Numeric fields are set to zero and alphameric fields are set to blanks. This column must be blank for Look-Ahead and Udate fields.

Resetting fields to zeros is useful when you are accumulating and printing totals for each control group. After finding the total for one group and printing it, you want to start accumulating totals for the next group. Before you do this, however, you want your total field to start with zeros, not with the total it had for the previous group. Blank After will reset the total field to zero after it is printed.

If the field is lo be used for output more than once (i.e., punching and printing), be sure the *B* is entered on the last output line for that field, Otherwise, the field is blanked out before all required output is finished.

COLUMNS 40-43 (END POSITION IN OUTPUT RECORD)

Disk, Punched Cards and Printed Reports

Use columns 40-43 to indicate the location on the output record of the field or constant that is to be written. You enter only the number of the punching or printing position of the rightmost character in the field or constant.

The largest number to be used to indicate end position for disk output is 4,096. The largest number for punched card output is 96. The largest number for printer output depends upon the number of print positions on the printer you have.

When *PLACE is specified for the printer (see *Columns 33-37* in this chapter), end position indicates the end position of the last field of the group that is to be printed. Thus you must be sure you have indicated an end position that allows enough room for all specified fields to be printed.

Be sure to allow enough space (as indicated by end position entries) on your output record to hold edited fields.

Printing on Cards

The MFCU prints and punches fields in the same positions on a card by using *PRINT in columns 32-37. If you want to print fields in positions *other* than those which correspond to the punch positions of the fields, you must:

1. name the field in columns 32-37.
2. place an * in column 40.
3. specify an end position for that field in columns 41-43. The maximum entry for an end position is 128.

The field will be printed in the upper portion of the card in the position you have specified.

All lines with an * in column 40 should follow all lines specifying punching only and all *PRINT lines for that record (see ***Example).*** All the punching for a card is done before the printing.

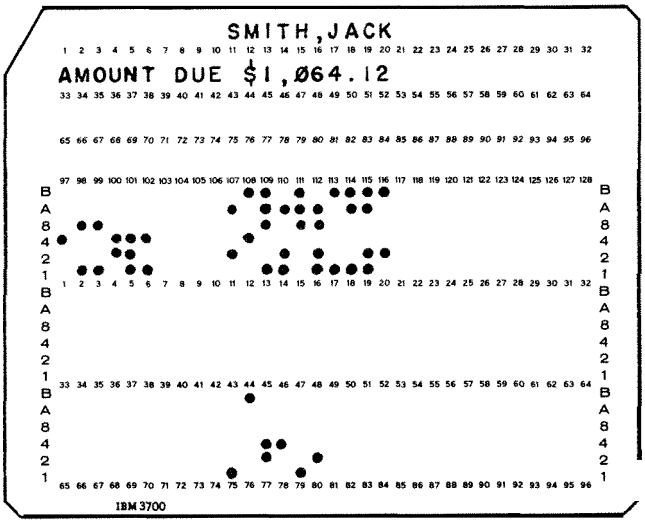
Note: If Blank After (column 39) is specified for a field to be punched and printed, the B entry must be entered on the last line specifying printing for that field. All the printing is done for a card after all the punching, so be careful not to blank out a punch field and then try to print it later. If

*PRINT is the last line specifying printing for a field, the B entry is made in the last punching specification line for that field. If an * is used in column 40 to print a field after it is punched, the B entry is made in the last print specification line for that field. **A** Blank After entry is correctly entered for a punch and print field in Figure **9-1**1.

Example

Figure **9-1**1 shows several examples of printing on a card. The coding shows that the name field will be punched and printed in the same card columns. The account number field is punched only. The amount due field is punched in columns 75-80, but for ease of reading it is printed with an edit word in columns 44-52. For the same reason, a constant is printed to identify the amount due field.

In Line 06, the field AMTDUE is blanked out after it is printed by a B entry in column **39.** If the B entry appeared in column **39** of line 05, the field would be blanked out after punching and would not be available for printing.



IBM

International Business Machines Corporation

Form X21-9090
Printed in U.S.A.

RPG OUTPUT - FORMAT SPECIFICATIONS

Date _____

Program _____

Programmer _____

Punching Instruction: Graphic / Punch

Page 1 2

Program Identification 15 76 71 78 79 80

Line	Form Type	Filename	Type (H/D/T/E)	Stacker Select/Fetch Overflow (F)	Space Before	Space After	Skip Before	Skip After	Output Indicators And Not	Output Indicators And Not	Output Indicators And Not	Field Name	Edit Codes	Blank After (B)	End Positon in Output Record	P = Packed/B = Binary	Constant or Edit Word	Sterling Sign Position
3 4 5	6	7 8 9 10 11 12 13 14	15	16	17	18	19 20	21 22	23 24	25 26	27 28	29 30 31	32 33 34 35 36 37	38	39	40 41 42 43	44	45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70	71 72 73 74
0 1	O	CARDOUT	D						Ø1	Ø3								
0 2	O											NAME			2Ø			
0 3	O											*PRINT						
0 4	O											ACCTNO			6			
0 5	O											AMTDUE			8Ø			
0 6	O											AMTDUE		B	* 52		' , $Ø. '	
0 7	O													*	42		'AMOUNT DUE'	
0 8	O																	
0 9	O																	

Edit Codes

Commas	Zero Balances to Print	No Sign	CR	-	
Yes	Yes	1	A	J	X = Remove Plus Sign
Yes	No	2	B	K	Y = Date Field Edit
No	Yes	3	C	L	Z = Zero Suppress
No	No	4	D	M	

Figure 9-11. **Printing on the** MFCU

COLUMN 44 (PACKED OR BINARY FIELD)

Entry	*Explanation*
Blank	Field is unpacked numeric data, alphameric data, or is to be printed.
P	Field is to be written on disk or punched in the packed decimal format.
B	Field is to be written on disk in the binary format.

Column 44 must have an entry if a numeric field is to be written on disk or punched in cards in the packed decimal format, or written on disk in the binary format. Packed decimal fields cannot be printed, and binary fields cannot be printed or punched.

Fields of 4 or less bytes are converted to 2 bytes of binary data for output, and fields of 5-9 bytes are converted to 4 bytes of binary data for output. The output device for binary fields can only be disk.

Column 44 must be blank if an asterisk (*) appears in column 40 of the same field specification. Column 44 must also be blank for fields in a record that precede *PRINT with an MFCU file or *PLACE with a printer file.

COLUMNS 45-70 (CONSTANT OR EDIT WORD)

Use columns 45-70 to specify a constant or an edit word.

Constant

A constant is any unchanging information that is entered by a specification. Constants are usually words used for report headings, column headings or card identification. To print a constant on a card, an * must be entered in column 40 (see ***Columns 40-43*** in this chapter for printing on cards).

The following rules apply to constants. (Refer to Figure 9-12, insert A for examples.)

1. Field name (columns 32-37) must be blank.

2. A constant must be enclosed in apostrophes. Enter the leading apostrophe in column 45.

3. An apostrophe in a constant must be represented by two apostrophes. For example, if the word you're appears in a constant it must be coded as YOU''RE.

4. Numeric data may be used as a constant.

5. Up to 24 characters of constant information can be placed in one line. Additional lines may be used, but each line must be treated as a separate line of constants. The end position of each line must appear in columns 40-43.

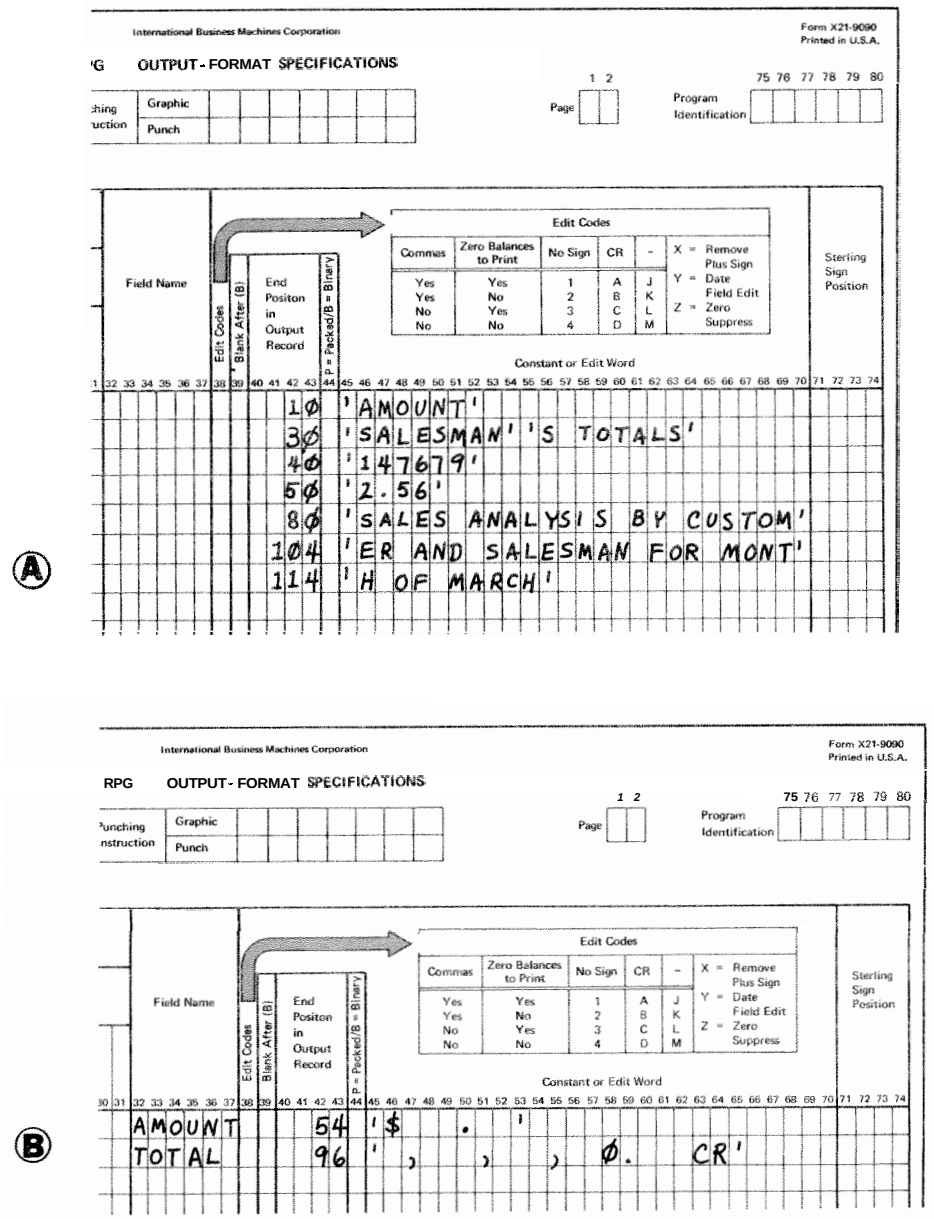


Figure 9-12. Constants and Edit Words

Edit Word

Use an edit word in place of an edit code when you want to punctuate the field named in column **32-37** according to your own format. Edit words can be used to suppress leading zeros, punctuate with decimal points, commas, dollar signs and asterisks, insert spaces and identify negative totals. Figure **9-12,** insert B, shows examples of edit words. For further information on edit words see ***Editing*** in Chapter ***10.***

The following rules apply to edit words:

1. Field name (columns **32-37)** must contain an entry.
2. An edit word must be enclosed in apostrophes. Enter leading apostrophe in column **45.** The edit word itself must begin in column **46.**
3. **Any** printable character is valid, but certain characters in certain positions have special uses (see ***Editing*** in Chapter **10).**
4. An edit word cannot be longer than **24** characters.
5. The numer of replaceable characters in the edit word must be equal to the length of the field to be edited. See ***Editing, Edit Words,*** in Chapter **10** for a discussion of replaceable characters.

COLUMNS 71-74 (STERLING SIGN POSITION)

Use columns **71-74** only when processing sterling currency amounts. For detailed information see ***Sterling*** in Chapter **10.**

Printer

Entry	*Explanation*
All blanks	Field is printed in pence only.
S in column **74**	Field is printed in pounds, shillings, and pence.

Output Devices Other Than the Printer

Entry	*Explanation*
Blank	Sterling output is not used.
Position in record	Number of the column which contains the sign if the sign is not in the normal position.
S in column **74**	Sign is in the normal position.

For output devices other than the printer, these columns are used to indicate the position of the sign of the field. The normal position of the sign in a field having decimal positions is in the rightmost decimal position of the pence field. If the fields have no decimal position, the normal sign position is in the last column (unit position) of the pounds fields.

COLUMNS 75-80 (PROGRAM IDENTIFICATION)

See Chapter **2.**

Chapter 10. Supplementary Information

This chapter further explains topics which were introduced, but not fully explained, in the preceding chapters. Because the discussion of each topic is complete, the sections are arranged alphabetically by section title.

ALTERNATE COLLATING SEQUENCE

Every alphabetic, numeric, or special character holds a special position in relation to all other characters. This special order is known as the collating sequence. System/3 uses a collating sequence based on the way characters are represented in the machine (Figure 10-1).

You may change this collating sequence if you wish. If you want characters to appear in a sequence other than the one used by System/3 or if you want two or more characters to have the same position in the sequence (this means they are considered equal), you must describe an alternate collating sequence.

Note: An alternate collating sequence applies to:

1. matching fields and sequence checking.
2. alphameric compare operations (COMP).

Defining an Alternate Collating Sequence

To define an alternate collating sequence you must first indicate that a sequence other than the normal one is to be used. Do this by entering an S in column 26 of the **RPG II** control card specifications.

A table also must be entered which lists the changes you wish to make in the normal collating sequence. The following entries are needed for each table card entered:

Columns 1-6: Enter ALTSEQ to indicate that you are altering the normal sequence.

Columns 7-8: Leave these columns blank.

Columns 9-10: Enter the hexadecimal number of the character being taken out of sequence. The table in Figure 10-1 lists characters and their hexadecimal equivalents.

Columns 11-12: Enter the hexadecimal number of the character that is replacing the character taken out of sequence.

*Columns 13-16, 17-20, 21-24, **etc:*** These columns are used the way columns 9-12 are used. The first two columns give the character to be replaced by the character specified in the next two columns. There may be as many four-column entries as necessary. Additional cards may be used with the above format. The first blank column terminates the card. A ** or /* ends the table.

The alternate sequence table deck must be preceded by a card with **& in columns 1-3. The remaining columns of the card may be used for comments. This deck must follow the **RPG II** specification deck and file translation cards, if used. Figure 1-2 shows the arrangement of cards in an RPG II source deck.

Translation Table and Alternate Collating Sequence Coding Sheet

The Translation Table and Alternate Collating Sequence Sheet (Figure 10-2) can be used for coding an alternate collating sequence. It helps you more easily determine the entries needed for the alternate collating sequence table input cards.

Collating Sequence	Character	Hexadecimal Equivalent
1	Blank	40
2	Ç	4A
3	.	4B
4	<	4C
5	(	4D
6	+	4E
7	|	4F
8	&	50
9	!	5A
10	$	5B
11	*	5C
12	)	5D
13	;	5E
14	¬	5F
15	- (minus)	60
16	/	61
17	,	6B
18	%	6C
19	_ (underscore)	6D
20	>	6E
21	?	6F
22	:	7A
23	#	7B
24	@	7C
25	'	7D
26	=	7E
27	"	7F
28	A	C1
29	B	C2
30	C	C3
31	D	C4
32	E	C5

Collating Sequence	Character	Hexadecimal Equivalent
33	F	C6
34	G	C7
35	H	C8
36	I	C9
37	}	D0
38	J	D1
39	K	D2
40	L	D3
41	M	D4
42	N	D5
43	O	D6
44	P	D7
45	Q	D8
46	R	D9
47	S	E2
48	T	E3
49	U	E4
50	V	E5
51	W	E6
52	X	E7
53	Y	E8
54	Z	E9
55	0	F0
56	1	F1
57	2	F2
58	3	F3
59	4	F4
60	5	F5
61	6	F6
62	7	F7
63	8	F8
64	9	F9

Figure 10-1. Normal Collating Sequence and Hexadecimal Equivalents of Characters

International Business Machines Corporation

Form X21-9096
Printed in U.S.A.

TRANSLATION TABLE AND ALTERNATE COLLATING SEQUENCE CODING SHEET

Code	System/3 Graphic	Entry	Replaced By
00000000		00	
00000001		01	
00000010		02	
00000011		03	
00000100		04	
00000101		05	
00000110		06	
00000111		07	
00001000		08	
00001001		09	
00001010		0A	
00001011		0B	
00001100		0C	
00001101		0D	
00001110		0E	
00001111		0F	
00010000		10	
00010001		11	
00010010		12	
00010011		13	
00010100		14	
00010101		15	
00010110		16	
00010111		17	
00011000		18	
00011001		19	
00011010		1A	
00011011		1B	
00011100		1C	
00011101		1D	
00011110		1E	
00011111		1F	
00100000		20	
00100001		21	
00100010		22	
00100011		23	
00100100		24	
00100101		25	
00100110		26	
00100111		27	
00101000		28	
00101001		29	
00101010		2A	
00101011		2B	
00101100		2C	
00101101		2D	
00101110		2E	
00101111		2F	
00110000		30	
00110001		31	
00110010		32	

Code	System/3 Graphic	Entry	Replaced By
00110011		33	
00110100		34	
00110101		35	
00110110		36	
00110111		37	
00111000		38	
00111001		39	
00111010		3A	
00111011		3B	
00111100		3C	
00111101		3D	
00111110		3E	
00111111		3F	
01000000	Blank	40	
01000001		41	
01000010		42	
01000011		43	
01000100		44	
01000101		45	
01000110		46	
01000111		47	
01001000		48	
01001001		49	
01001010	¢	4A	
01001011	.	4B	
01001100	<	4C	
01001101	(	4D	
01001110	+	4E	
01001111	|	4F	
01010000	&	50	
01010001		51	
01010010		52	
01010011		53	
01010100		54	
01010101		55	
01010110		56	
01010111		57	
01011000		58	
01011001		59	
01011010	!	5A	
01011011	$	5B	
01011100	*	5C	
01011101	)	5D	
01011110	;	5E	
01011111	¬	5F	
01100000	-	60	
01100001	/	61	
01100010		62	
01100011		63	
01100100		64	
01100101		65	

Code	System/3 Graphic	Entry	Replaced By
01100110		66	
01100111		67	
01101000		68	
01101001		69	
01101010		6A	
01101011	,	6B	
01101100	%	6C	
01101101	_	6D	
01101110	>	6E	
01101111	?	6F	
01110000		70	
01110001		71	
01110010		72	
01110011		73	
01110100		74	
01110101		75	
01110110		76	
01110111		77	
01111000		78	
01111001		79	
01111010	:	7A	
01111011	#	7B	
01111100	@	7C	
01111101	'	7D	
01111110	=	7E	
01111111	"	7F	
10000000		80	
10000001		81	
10000010		82	
10000011		83	
10000100		84	
10000101		85	
10000110		86	
10000111		87	
10001000		88	
10001001		89	
10001010		8A	
10001011		8B	
10001100		8C	
10001101		8D	
10001110		8E	
10001111		8F	
10010000		90	
10010001		91	
10010010		92	
10010011		93	
10010100		94	
10010101		95	
10010110		96	
10010111		97	
10011000		98	

Code	System/3 Graphic	Entry	Replaced By
10011001		99	
10011010		9A	
10011011		9B	
10011100		9C	
10011101		9D	
10011110		9E	
10011111		9F	
10100000		A0	
10100001		A1	
10100010		A2	
10100011		A3	
10100100		A4	
10100101		A5	
10100110		A6	
10100111		A7	
10101000		A8	
10101001		A9	
10101010		AA	
10101011		AB	
10101100		AC	
10101101		AD	
10101110		AE	
10101111		AF	
10110000		B0	
10110001		B1	
10110010		B2	
10110011		B3	
10110100		B4	
10110101		B5	
10110110		B6	
10110111		B7	
10111000		B8	
10111001		B9	
10111010		BA	
10111011		BB	
10111100		BC	
10111101		BD	
10111110		BE	
10111111		BF	
11000000		C0	
11000001	A	C1	
11000010	B	C2	
11000011	C	C3	
11000100	D	C4	
11000101	E	C5	
11000110	F	C6	
11000111	G	C7	
11001000	H	C8	
11001001	I	C9	
11001010		CA	
11001011		CB	

Code	System/3 Graphic	Entry	Replaced By
11001100		CC	
11001101		CD	
11001110		CE	
11001111		CF	
11010000	}	D0	
11010001	J	D1	
11010010	K	D2	
11010011	L	D3	
11010100	M	D4	
11010101	N	D5	
11010110	O	D6	
11010111	P	D7	
11011000	Q	D8	
11011001	R	D9	
11011010		DA	
11011011		DB	
11011100		DC	
11011101		DD	
11011110		DE	
11011111		DF	
11100000		E0	
11100001		E1	
11100010	S	E2	
11100011	T	E3	
11100100	U	E4	
11100101	V	E5	
11100110	W	E6	
11100111	X	E7	
11101000	Y	E8	
11101001	Z	E9	
11101010		EA	
11101011		EB	
11101100		EC	
11101101		ED	
11101110		EE	
11101111		EF	
11110000	0	F0	
11110001	1	F1	
11110010	2	F2	
11110011	3	F3	
11110100	4	F4	
11110101	5	F5	
11110110	6	F6	
11110111	7	F7	
11111000	8	F8	
11111001	9	F9	
11111010		FA	
11111011		FB	
11111100		FC	
11111101		FD	
11111110		FE	
11111111		FF	

ART: 51682

Figure 10-2. Translation Tabb and Alternate Collating Sequence Coding Sheet

Causing Characters to Be Considered Equal

If you want one character to be considered the same as another character, the characters must hold the same position in the collating sequence. For example, you may wish a blank to be considered as a zero. Therefore, you need to define an alternate collating sequence in which the blank is the same as the zero because it holds the same position in the sequence. The alternate collating sequence input card looks like this:

Column	*Entry*
1-6	ALTSEQ
7-8	Blanks
9-12	40F0 (blank takes the zero's position)

Now whenever a blank is read and used in a compare it is considered as a zero. Thus, if you were comparing numbers to 0036 to find an equal condition, *0036* and bb36 (where b=blank) both compare equal to *0036.*

Altering the Normal Collating Sequence

You may after the normal collating sequence in a number of ways. For example, you may insert a character between two existing characters, you may take a character out of the sequence, or you may change characters (put A where Z is and Z where A is). Regardless of how you alter the sequence, you must specify every character that is to be changed by the alteration. For example, if you want the dollar sign ($) to be positioned in the collating sequence between A and B, the normal sequence is changed as follows:

Normal Sequence	*Altered Sequence*
A	A
B	$
C	B
D	C
E	D
F	E
G	F
H	G
	H
	I

Notice on the Translation Table and Alternate Collating Sequence Coding Sheet that there are many characters between I and }, R and S, Z and *0*. These characters can be represented in the computer and on records by a certain code. However, they have no printable graphic symbol. Due to this particular arrangement of graphics, nongraphics, graphics, etc. in the collating sequence, a character, when inserted between A and B, changes only the position of graphics B-I. All other graphics are not affected. B-I ail move down one position causing the I to take the place of the nongraphic represented by hexadecimal CA. This does not matter, however, since the original character CA cannot be printed anyway. See Figure 10-3 for the entries on the Translation Table and Alternate Collating Sequence Coding Sheet.

The alternate sequence input card is punched as follows:

Column	*Entry*
1-6	ALTSEQ
7-8	(blanks)
9-12	5BC2 ($ takes B's position)
13-16	C2C3 (B takes C's position)
17-20	C3C4 (C takes D's position)
21-24	C4C5 (D takes E's position)
25-28	C5C6 (E takes F's position)
29-32	C6C7 (F takes G's position)
33-36	C7C8 (G takes H's position)
37-40	C8C9 (H takes I's position)
41-44	C9CA (I is given a new position held by no other printable character.)

ARRAYS

An array is a continuous series of data fields having like characteristics, that is, same field length and same number of decimal positions.

There are three kinds of arrays: compile time arrays, execution time arrays, and those arrays loaded or created by input and/or calculations specifications.

A compile time array is compiled with the source program and becomes a permanent part of the object program. A compile time array can, then, be permanently changed only by recompiling the source program with the revised array.

An execution time array is loaded with the object program before actual execution of your RPG II program begins (that is, before any input files are read, calculations performed, or output function performed).

An array loaded or created by input or calculation specifications, which might be called a dynamic array, is loaded into the computer after actual execution of your RPG II program has begun (it is read as input data) or is created during the calculation phase of your RPG 11 program. Such a dynamic array must nonetheless be described on the Extension sheet.

International Business Machines Corporation

TRANSLATION TABLE AND ALTERNATE COLLATING SEQUENCE CODING SHEET

Code	System/3 Graphic	Entry	Replaced By
00110011		33	
00110100		34	
00110101		35	
00110110		36	
00110111		37	
00111000		38	
00111001		39	
00111010		3A	
00111011		3B	
00111100		3C	
00111101		3D	
00111110		3E	
00111111		3F	
01000000	Blank	40	
01000001		41	
01000010		42	
01000011		43	
01000100		44	
01000101		45	
01000110		46	
01000111		47	
01001000		48	
01001001		49	
01001010	¢	4A	
01001011	.	4B	
01001100	<	4C	
01001101	(	4D	
01001110	+	4E	
01001111	|	4F	
01010000	&	50	
01010001		51	
01010010		52	
01010011		53	
01010100		54	
01010101		55	
01010110		56	
01010111		57	
01011000		58	
01011001		59	
01011010	!	5A	
01011011	$	5B	C2 (B)
01011100	*	5C	
01011101	)	5D	
01011110	;	5E	
01011111	¬	5F	
01100000	-	60	
01100001	/	61	
01100010		62	
01100011		63	
01100100		64	
01100101		65	

Code	System/3 Graphic	Entry	Replaced By
01100110		66	
01100111		67	
01101000		68	
01101001		69	
01101010	¦	6A	
01101011	,	6B	
01101100	%	6C	
01101101	_	6D	
01101110	>	6E	
01101111	?	6F	
01110000		70	
01110001		71	
01110010		72	
01110011		73	
01110100		74	
01110101		75	
01110110		76	
01110111		77	
01111000		78	
01111001		79	
01111010	:	7A	
01111011	#	7B	
01111100	@	7C	
01111101	'	7D	
01111110	=	7E	
01111111	"	7F	
10000000		80	
10000001		81	
10000010		82	
10000011		83	
10000100		84	
10000101		85	
10000110		86	
10000111		87	
10001000		88	
10001001		89	
10001010		8A	
10001011		8B	
10001100		8C	
		8D	
		8E	
		8F	
10010000		90	
10010001		91	
10010010		92	
10010011		93	
10010100		94	
10010101		95	
10010110		96	
10010111		97	
10011000		98	

$ takes B's position.

Code	System/3 Graphic	Entry	Replaced By
10011001		99	
10011010		9A	
10011011		9B	
10011100		9C	
10011101		9D	
10011110		9E	
10011111		9F	
10100000		A0	
10100001		A1	
10100010		A2	
10100011		A3	
10100100		A4	
10100101		A5	
10100110		A6	
10100111		A7	
10101000		A8	
10101001		A9	
10101010		AA	
10101011		AB	
10101100		AC	
10101101		AD	
10101110		AE	
10101111		AF	
10110000		B0	
10110001		B1	
10110010		B2	
10110011		B3	
10110100		B4	
10110101		B5	
10110110		B6	
10110111		B7	
10111000		B8	
10111001		B9	
10111010		BA	
10111011		BB	
10111100		BC	
10111101		BD	
10111110		BE	
10111111		BF	
11000000		C0	
11000001	A	C1	
11000010	B	C2	C3 (C)
11000011	C	C3	C4 (D)
11000100	D	C4	C5 (E)
11000101	E	C5	C6 (F)
11000110	F	C6	C7 (G)
11000111	G	C7	C8 (H)
11001000	H	C8	C9 (I)
11001001	I	C9	CA
11001010		CA	
11001011		CB	

B takes C's position.

C takes D's position.

(no printable character)

Code	System/3 Graphic
11001100	
11001101	
11001110	
11001111	
11010000	}
11010001	J
11010010	K
11010011	L
11010100	M
11010101	N
11010110	O
11010111	P
11011000	Q
11011001	R
11011010	
11011011	
11011100	
11011101	
11011110	
11011111	
11100000	
11100001	
11100010	S
11100011	T
11100100	U
11100101	V
11100110	W
11100111	X
11101000	Y
11101001	Z
11101010	
11101011	
11101100	
11101101	
11101110	
11101111	
11110000	0
11110001	1
11110010	2
11110011	3
11110100	4
11110101	5
11110110	6
11110111	7
11111000	8
11111001	9
11111010	
11111011	
11111100	
11111101	
11111110	
11111111	

ART: 51683

Figure 10-3. Altering the Collating Sequence

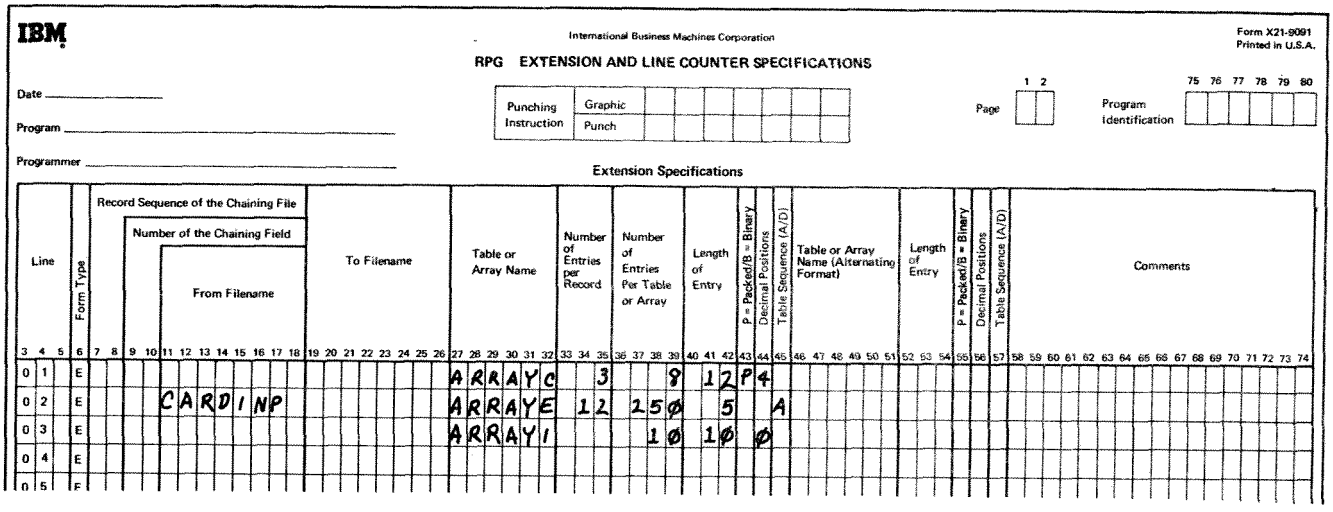
Defining Arrays—Extension Specificatons

Every array must be defmed on extension specifications. The entries are as follows:

Column	*Entry*
6	E
7-10	blank
11-18	Filename of execution time array. Blank for compile time array or for array created by input and/or calculation specifications.
19-26	Filename of output file on which array is written at end of job. Blank—array is not written out at end of job.
27-32	Name of the array. The array name cannot begin with the letters TAB. (See ***Array Name and Index***.)
33-35	Number of array elements found in each input record for execution or compile time arrays. Leave these columns blank for an array defmed in input and calculation specifications.
36-39	Number of elements in the array. This entry must end in column 39. Leading zeros are not required.
4042	Decimal field length of array element. All elements in the array must be the same length. If the array is in packed, binary, or BSI shilling format, this entry must be the converted decimal length.
43	P Array elements are in packed format. B Array elements are in binary format. Blank Array elements are in alphameric or decimal format, Leave this column blank for an array defined in input and calculation specifications.
44	For numeric fields enter the number of digits to the right of the decimal point. This number can be *0*. For alphameric fields leave column 44 blank.
45	A Array is to be checked for ascending sequence. D Array is to be checked for descending sequence. Blank No sequence checking is done. This column must be blank for any array created in input and calculations specifications.
46-57	Use these columns to describe an alternating array.

Figure 1 0 4 shows the necessary extension specifications for each type of array. Line 1 specifies a compile time array, ARRAYC. This array has a total of eight elements (three elements per record). Each element has an unpacked length of 12 positions, including 4 decimal places. Line 2 specifies an execution time array ARRAYE, to be read from file CARDINP. ARRAYE has 250 alphameric elements (10 elements per record); each element is 5 positions long and higher in collating sequence than the previous element. Line 3 specifies an array, ARRAYI, to be read from input records. ARRAYI has 10 numeric elements each 10 positions long.

Any of these specifications may include entries in columns 19-26 (to define a filename of a file to which the array would be output at end of job) and in columns 46-57 (to define an alternating array).



IBM

International Business Machines Corporation

Form X21-9091
Printed in U.S.A.

RPG EXTENSION AND LINE COUNTER SPECIFICATIONS

Date _____

Program _____

Programmer _____

Punching Instruction	Graphic
	Punch

Page 1 2

Program Identification 75 76 77 78 79 80

Extension Specifications

Line 3 4 5	Form Type 6	Record Sequence of the Chaining File 7 8	Number of the Chaining Field 9 10	From Filename 11 12 13 14 15 16 17 18	To Filename 19 20 21 22 23 24 25 26	Table or Array Name 27 28 29 30 31 32	Number of Entries per Record 33 34 35	Number of Entries Per Table or Array 36 37 38 39	Length of Entry 40 41 42	P = Packed/B = Binary 43	Decimal Positions 44	Table Sequence (A/D) 45	Table or Array Name (Alternating Format) 46 47 48 49 50 51	Length of Entry 52 53 54	P = Packed/B = Binary 55	Decimal Positions 56	Table Sequence (A/D) 57	Comments 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74
01	E					ARRAYC	3	9	12	P	4							
02	E			CARDINP		ARRAYE	12	250	5		A							
03	E					ARRAY1		10	10	Ø								
04	E																	
05	E																	

Figure 10-4. Varieties of Arrays

Input Specifications

If you are reading array information from input records; that is, if columns 11-18, and columns 33-35 of the Extension sheet are blank, you must describe that information in your input specifications as well as in the extension specifications. How the entries are made depends on whether the array information is contained in one or more records.

Note: An array name with a variable index cannot be defined as a look-ahead field.

Array Information in One Record

If all of the array information is in one record, it can occupy consecutive positions in the record or be scattered throughout the record.

If the array elements are consecutive on the input record, they may be loaded with a single input specification. Figure 10-5 shows an array, INPARR, of six elements (twelve positions each) being loaded from a single record from the file ARRFILE.

If the array elements are scattered throughout the record, they may be defined and loaded one at a time, one to a specification line. In Figure 10-6, an array, ARRX, of six elements with 12 positions each, is loaded from a single record from file ARRFILE; a blank column appears between each two elements.

Following are the input specifications required for loading an array from a single input record:

Column	*Entry*
6	I
7-42	Blank
43	P (packed), B (binary) or blank.
44-47 and 48-51	Field location of either an entire array (consecutive elements) or individual field locations of single elements of the array.
52	This column can be left blank. If a decimal position entry is made, it must be the same as that specified on the Extension sheet.

IBM

International Business Machines Corporation

Form X21-9091 Printed in U.S.A.

RPG EXTENSION AND LINE COUNTER SPECIFICATIONS

Date _____ Program _____ Programmer _____

Punching Instruction: Graphic / Punch

Page 1 2

Program Identification 75 76 77 78 79 80

Extension Specifications

Line	Form Type	Record Sequence of the Chaining File	Number of the Chaining Field	From Filename	To Filename	Table or Array Name	Number of Entries per Record	Number of Entries Per Table or Array	Length of Entry	P = Packed/B = Binary	Decimal Positions	Table Sequence (A/D)	Table or Array Name (Alternating Format)	Length of Entry	P = Packed/B = Binary	Decimal Positions	Table Sequence (A/D)	Comments
3 4 5	6	7 8	9 10	11 12 13 14 15 16 17 18	19 20 21 22 23 24 25 26	27 28 29 30 31 32	33 34 35	36 37 38 39	40 41 42	43	44	45	46 47 48 49 50 51	52 53 54	55	56	57	58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74
0 1	E					INPARR			12									
0 2	E																	
0 3	E																	

IBM

International Business Machines Corporation

Form X21-9094 Printed in U.S.A.

RPG INPUT SPECIFICATIONS

Date _____ Program _____ Programmer _____

Punching Instruction: Graphic / Punch

Page 1 2

Program Identification 75 76 77 78 79 80

Line	Form Type	Filename	Sequence	Number (1-N)	Option (O)	Record Identifying Indicator or **	Record Identification Codes 1: Position	1: Not (N)	1: C/Z/D	1: Character	2: Position	2: Not (N)	2: C/Z/D	2: Character	3: Position	3: Not (N)	3: C/Z/D	3: Character	Stacker Select	P = Packed/B = Binary	Field Location From	Field Location To	Decimal Positions	Field Name	Control Level (L1-L9)	Matching Fields or Chaining Fields	Field Record Relation	Field Indicators Plus	Field Indicators Minus	Field Indicators Zero or Blank	Sterling Sign Position
3 4 5	6	7 8 9 10 11 12 13 14	15 16	17	18	19 20	21 22 23 24	25	26	27	28 29 30 31	32	33	34	35 36 37 38	39	40	41	42	43	44 45 46 47	48 49 50 51	52	53 54 55 56 57 58	59	60 61	63 64	65 66	67 68	69 70	71 72 73 74
0 1	I	ARRFILE AA				Ø1															1	72		INPARR							
0 2	I																														
0 3	I																														
0 4	I																														

Figure 10-5. **Defining** an **Input** Time **Array with Consecutive Elements**

53-58	The name of the array or the name of a single element (array name with index). This array name must be the same name as that used on the Extension sheet.
59-62	Blank
63-64	Field Record Relation indicator. See ***Columns 63-64*** in Chapter 7 for information on this entry.
65-70	Blank
71-74	Sterling field. See ***Sterling*** in this chapter for information on this entry.

Array Information in More Than One Record

If the array information is in two or more records, there are many methods that may be used to introduce the array to the system. The method you use is primarily based on the size of the array and whether the array information is all together in the input records.

Examples 1-4 show four methods of loading and sorting array information.

Keep in mind that the RPG II program processes one record at a time. You cannot process the entire array until dl of the records containing the array information have been read and the information moved into the array fields. It may, therefore, be necessary to suppress calculation and output operations until the entire array has been read into the system.

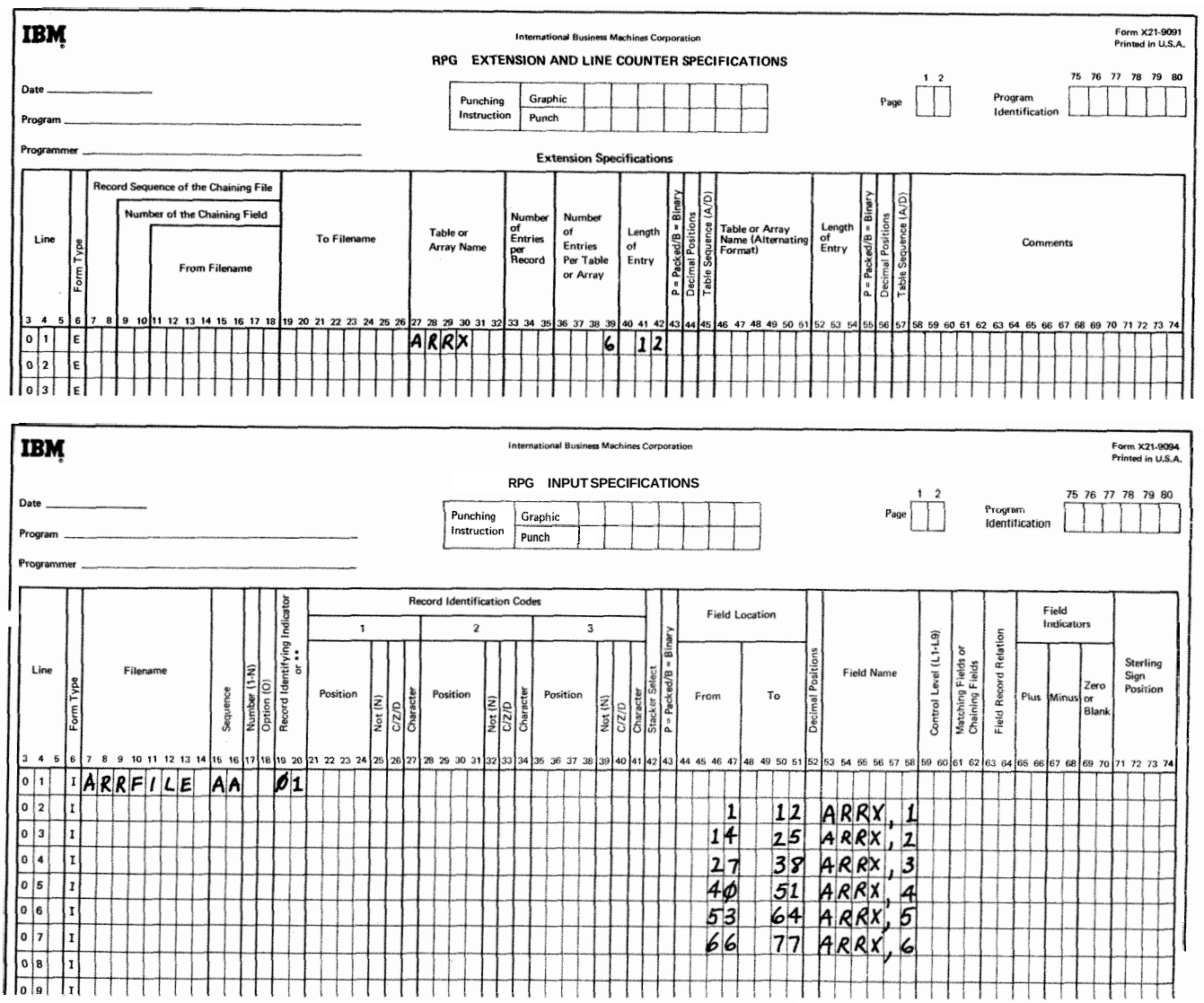
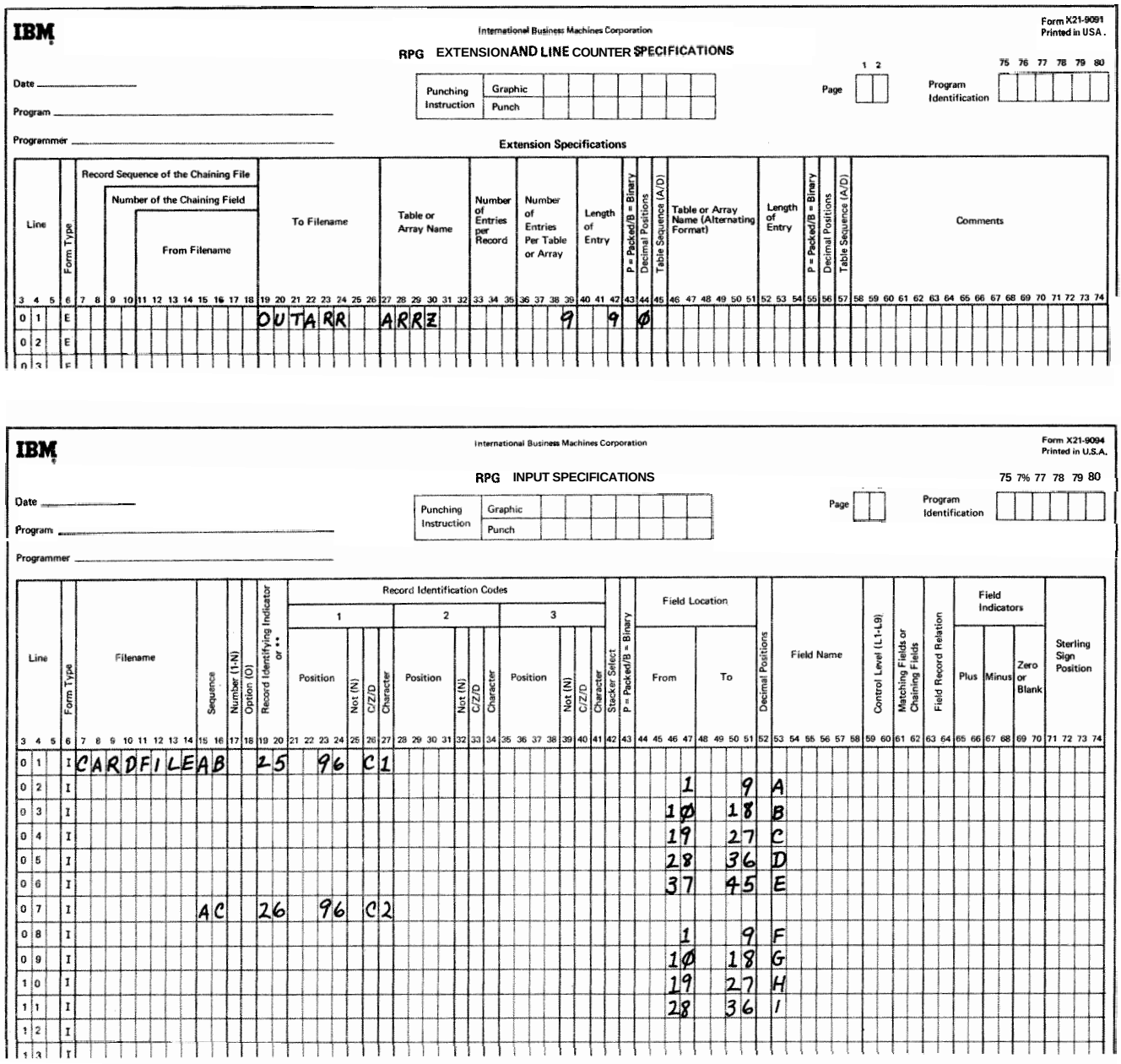


Figure 10-6. Defining an Input Time Array with Scattered Elements

Figure 10-7 shows the creation of an array, **ARRZ,** using fields from input records. Fields are extracted from records from file CARDFILE, and the square root of each of these fields is moved into an element of the array. The output file for this job is named OUTARR. Indicator **25** is on when the record containing data for the first five elements of **ARRZ** has been read. Similarly, indicator **26** is on when data for the remaining four elements has been read. Indicator **27** is turned on only after all elements have been calculated. Because not all the elements of ARRZ can be established in one RPG cycle, you will want to suppress (that is, skip) any calculations or output functions which use **ARRZ** or any of its elements until all of the elements are calculated. To suppress these calculations and output functions you would use indicator **27** as a conditioning indicator in any calculation or output specification which uses **ARRZ** or one of its elements. Line 14 of the Calculation sheet of Figure **10-7** shows indicator **27** used to condition an operation which should not be performed until all elements of **ARRZ** have been calculated.



IBM

International Business Machines Corporation

Form X21-9091 Printed in USA.

RPG EXTENSION AND LINE COUNTER SPECIFICATIONS

Date _____ Program _____ Programmer _____

Punching Instruction: Graphic / Punch

Page 1 2 Program Identification 75 76 77 78 79 80

Extension Specifications

Line	Form Type	Record Sequence of the Chaining File	Number of the Chaining Field	From Filename	To Filename	Table or Array Name	Number of Entries per Record	Number of Entries Per Table or Array	Length of Entry	P = Packed/B = Binary	Decimal Positions	Table Sequence (A/D)	Table or Array Name (Alternating Format)	Length of Entry	P = Packed/B = Binary	Decimal Positions	Table Sequence (A/D)	Comments
01	E				OUTARR	ARRZ			9	9	Ø							
02	E																	
03	E																	

IBM

International Business Machines Corporation

Form X21-9094 Printed in U.S.A.

RPG INPUT SPECIFICATIONS

Date _____ Program _____ Programmer _____

Punching Instruction: Graphic / Punch

Page Program Identification 75 7% 77 78 79 80

Line	Form Type	Filename	Sequence	Number (1-N)	Option (O)	Record Identifying Indicator or **	Record Identification Codes 1: Position	1: Not (N)	1: C/Z/D	1: Character	2: Position	2: Not (N)	2: C/Z/D	2: Character	3: Position	3: Not (N)	3: C/Z/D	3: Character	Stacker Select	P = Packed/B = Binary	Field Location: From	Field Location: To	Decimal Positions	Field Name	Control Level (L1-L9)	Matching Fields or Chaining Fields	Field Record Relation	Field Indicators: Plus	Field Indicators: Minus	Field Indicators: Zero or Blank	Sterling Sign Position
01	I	CARDFILEAB				25	96		C1																						
02	I																				1	9	A								
03	I																				10	18	B								
04	I																				19	27	C								
05	I																				28	36	D								
06	I																				37	45	E								
07	I		AC			26	96		C2																						
08	I																				1	9	F								
09	I																				10	18	G								
10	I																				19	27	H								
11	I																				28	36	I								
12	I																														
13	I																														

Figure 10-7. Array to be Built During Calculations (Part 1 of 2)

Using Arrays

Arrays can be used in input, output, or calculation specifications (see *Examples*). The elements in an array can be referenced individually, or the array can be referenced as a whole. Individual elements are referenced by an array name plus an index. The array name alone references the entire array.

Array Name and Index

The array name must begin in column 27 of the Extension sheet. The array name cannot begin with the letters TAB. However, the name can contain any other combination of letters and numbers but must not begin with the letters TAB. The first character must be alphabetic. The name cannot contain blanks or special characters. The name must not be the same as any other name used in the program.

The length of the array name depends on how the array is being used. The array name can be from 1-6 characters long. The array name by itself is used only when referencing the entire array.

If individual elements of the array are to be referenced, the array name will require an index. The array name and index must be separated by a comma. The array name plus comma plus index will never occupy fewer than three character positions. Total length of an array name plus comma plus index entry is limited to six positions (input, output specifications, or Result Field of calculation specifications) or ten positions (Factor 1 or Factor 2 of calculation specifications).

IBM

International Business Machines Corporation

Form X21-9093
Printed in U.S.A

RPG CALCULATION SPECIFICATIONS

Date _____
Program _____
Programmer _____

Punching Instruction: Graphic / Punch

Page: 1 2

Program Identification: 75 76 77 78 79 80

Line	Form Type	Control Level (L0-L9, LR, SR)	Indicators: And / Not	Factor 1	Operation	Factor 2	Result Field	Field Length	Decimal Positions	Half Adjust (H)	Resulting Indicators: Arithmetic (Plus / Minus / Zero); Compare (High 1>2 / Low 1<2 / Equal 1=2); Lookup (Table (Factor 2) is High / Low / Equal)	Comments
01	C		N25 N26		GOTO	SKIP						
02	C		25		SQRT	A	ARRZ,1					
03	C		25		SQRT	B	ARRZ,2					
04	C		25		SQRT	C	ARRZ,3					
05	C		25		SQRT	D	ARRZ,4					
06	C		25		SQRT	E	ARRZ,5					
07	C	*	ARRAY ARRZ IS PARTIALLY LOADED									
08	C		26		SQRT	F	ARRZ,6					
09	C		26		SQRT	G	ARRZ,7					
10	C		26		SQRT	H	ARRZ,8					
11	C		26		SQRT	I	ARRZ,9					
12	C		26		SETON						27	
13	C			SKIP	TAG							
14	C		27		XFOOT	ARRZ	SUM	110				
15	C											
	C				More Calculations							
	C											
	C											
	C											

Figure 10-7. Array to be Built During Calculations (Part 2 of 2)

Some examples of array names with and without indexes are as follows:

Valid

ARAYO1 B	
AR,I	(the first element of array AR)
X,YY2	(where YY2 is a field name)

Invalid

BALANCE	(array name has more than six characters)
6TOTAL	(first character not alphabetic)
TOTAL-	(name contains special character)
CR TOT	(name contains blank)
A1, A1	(array is used as index)
BAL,XX1	(name including comma has more than six characters. This name is valid for Factor 1 and Factor 2 of the calculation specifications only.)

Calculation Specifications

You can reference an entire array or individual elements in an array using calculation specifications. Process individual elements like normal fields. Remember, if an array field is to be used as a result field, the array name plus comma plus index cannot exceed six characters.

To reference an entire array use only the array name. You may use it in Factor 1, Factor 2 or the Result Field. The operations you may use are: ADD, Z-ADD, SUB, Z-SUB, MULT, DIV, SQRT, MOVE, MOVEL, MLLZO, MLHZO, MHLZO, MHHZO, DSPLY (array element only), DEBUG, BITON, BITOF, XFOOT and LOKW.

The following rules apply when using arrays in calculations:

1. When the factors and the result field are all arrays with the same number of elements, the operation is performed using the first element from every array, then the second element from every array, etc., until all elements in the arrays are processed. If the arrays do not have the same number of entries the operation ends when the last element of the array with the fewest elements has completed processing.

2. When one of the factors is a field or constant and the other is an array, and the result field is an array, the operation is performed once for every element in the shorter array. The same field or constant is used in all of the operations. If the result field is not an array, the operation is performed once, using only the first element of the specified array.

3. Resulting indicators cannot be used due to multiple operations being performed. Exceptions are XFOOT and LOKUP which allow resulting indicators.

4. You can use indicators (columns 7-17) to condition the operation.

5. The arrays you use in arithmetic operations must be numeric. You may indicate Half Adjust (column 53) if you wish.

Two operations are unique in their handling of arrays. They are XFOOT and LOKW.

XFOOT

The XFOOT operation code totals the contents of all elements in the array named in factor 2 and places the total in a field named in the result field. This operation may be conditioned by indicators in columns 7-17. You can half-adjust the total in the result field and use resulting indicators if you wish.

LOKUP

Since arrays are similar to tables, the LOKW operation code can be used to determine whether the contents of an element in an array matches a search word. No special storage areas are required. The specifications for arrays are the same as for tables except that for LOKW, the result field cannot be used if Factor 2 is an array (see *Operation Codes, Lookup* in this chapter).

If you use just the array name in referencing the array, the search begins at the first element in the array. You must use indicators to determine if a match was found.

If you use the array name and an index (which may be a field name or a literal), the search begins at the element identified by the index. If a match is found, the number of the array element containing the match is placed in the field used as an index. If no match is found, the index field is set to 1.

If a literal was used as an index, indicators must be used to determine if a match was found. The content of the element referenced by the literal is not changed.

Output-Format Specifications

You can reference an entire array or individual element in an array in output specifications.

Entire Array

If an array is to be punched or printed in an output record, describe the array along with any normal fields for the record. The columns you use to describe the array and their contents on the output-format sheet are:

Columns	Entry
6	0
23-31	Output indicators. If used, they pertain to the entire array. See *Columns 23-31* in Chapter 9 for more information.
32-37	Array name. This must be the same name as that used on the Extension sheet.
38-39	Edit Code and Blank After. These columns may be used with arrays. See *Column 38* and *Column 39* in Chapter 9 for more information.
40-43	Enter the record position where the last field of the array is to end. Be aware that you must allow room for any editing you perform.
44	P = pack each element. B = convert each element to binary.
45-70	Edit word. If you use an edit word, it pertains to all fields in the array. Do not use an edit word if an edit code is used.
71-74	Blank.

individual Fields

If an output record is to contain certain fields from an array but not the entire array, describe the fields the same way you do normal fields. Use the array name and index (separated by a comma) as the field name. The index can be either a field or a number. Note that the length of the array name and index, including the comma, must not exceed six characters (the maximum length of a field name).

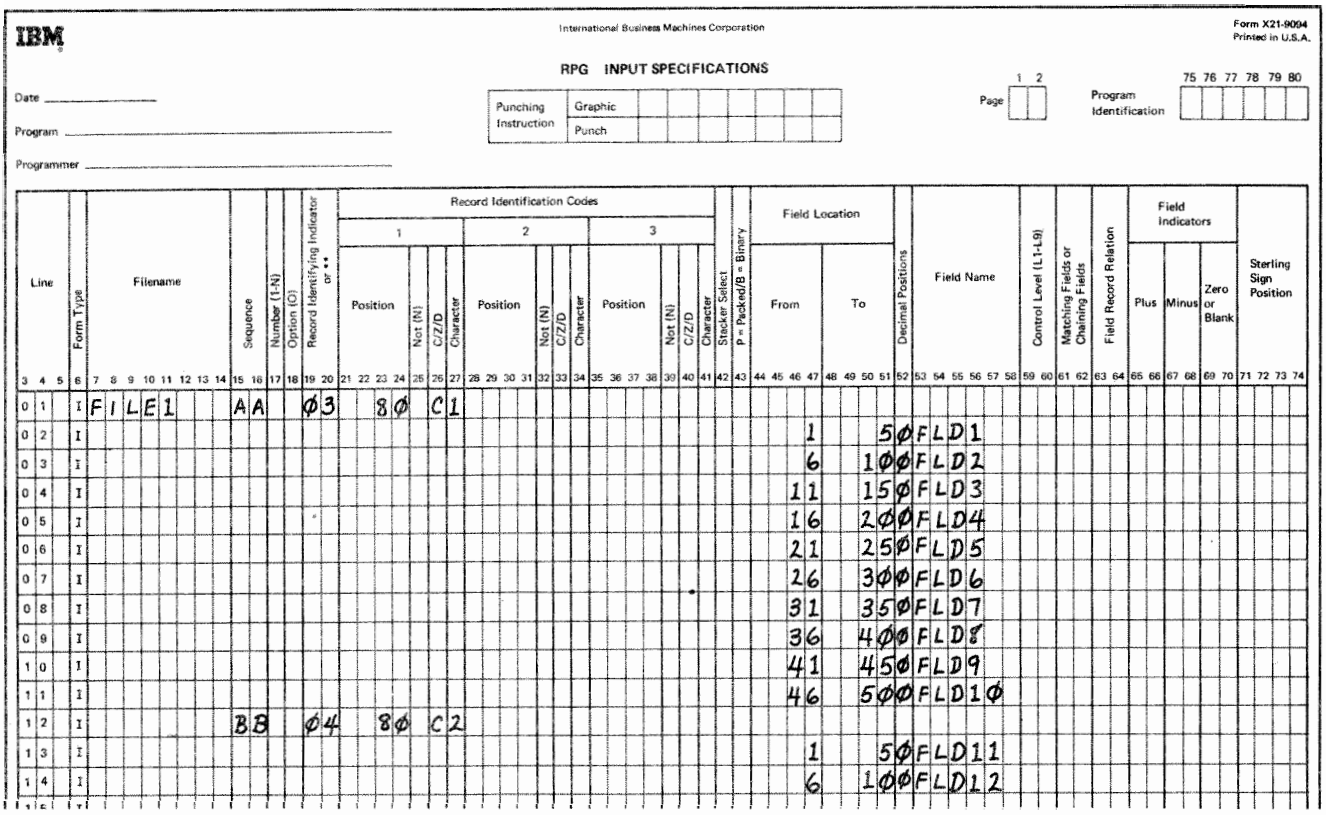
Editing Arrays

In editing arrays, remember that when you reference the entire array any editing you specify applies equally to all fields in the array. If you require different editing for various fields, you have to reference the fields individually.

When you specify an edit code for an entire array (column 38), note that two blanks are automatically inserted lo the left of every field in the array. When you specify an edit word instead, the blanks are not inserted. The edit word must specify all the blanks you want inserted.

Examples

Example 1: Figure 10-8 shows the straight substitution method of loading an array into the system. Array AR1 is to contain 12 elements, each 5 positions long. On the Input sheets each array element is given a separate field name. On the calculation sheet each input field is moved into an array element.



IBM

International Business Machines Corporation

Form X21-9094
Printed in U.S.A.

RPG INPUT SPECIFICATIONS

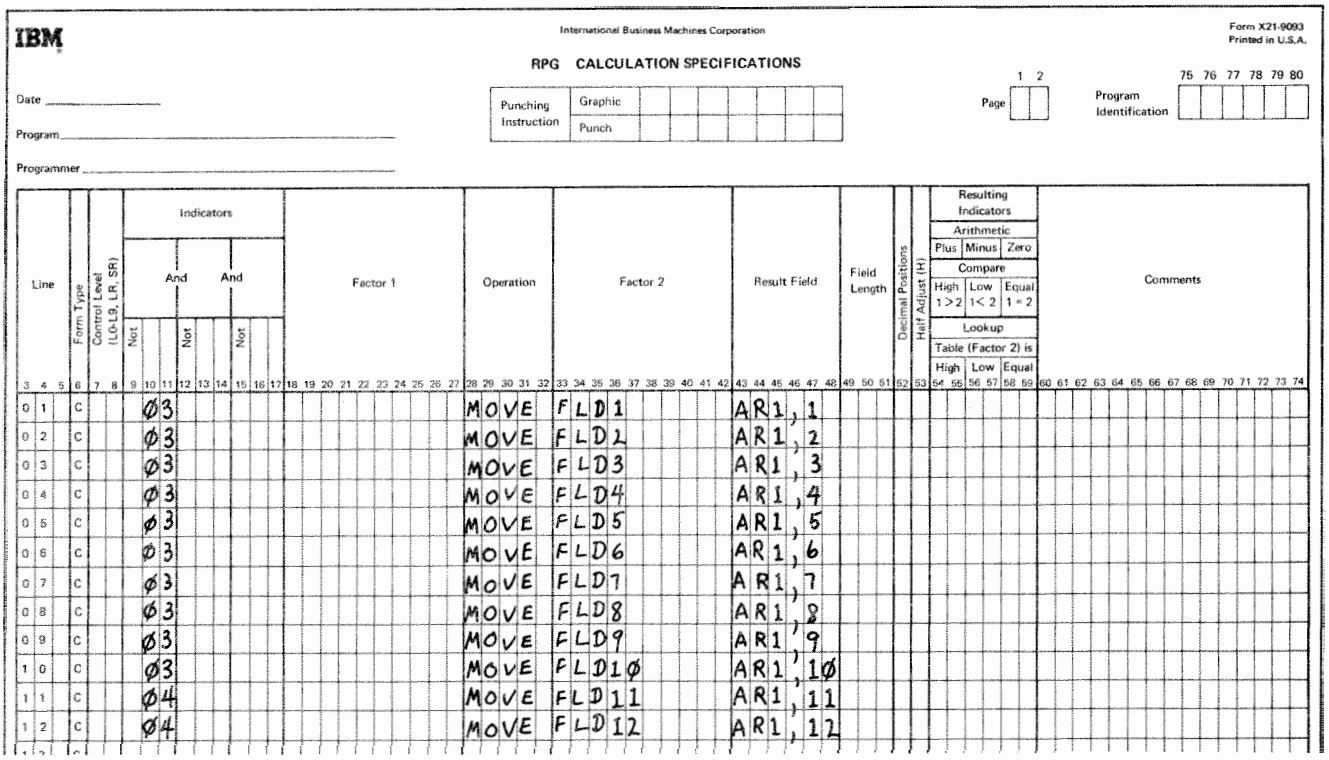
Date _____
Program _____
Programmer _____

Punching Instruction	Graphic								
	Punch								

Page 1 2

Program Identification 75 76 77 78 79 80

Line	Form Type	Filename	Sequence	Number (1-N)	Option (O)	Record Identifying Indicator or **	Record Identification Codes 1: Position	1: Not (N)	1: C/Z/D	1: Character	2: Position	2: Not (N)	2: C/Z/D	2: Character	3: Position	3: Not (N)	3: C/Z/D	3: Character	Stacker Select	P = Packed/B = Binary	Field Location: From	Field Location: To	Decimal Positions	Field Name	Control Level (L1-L9)	Matching Fields or Chaining Fields	Field Record Relation	Field Indicators: Plus	Field Indicators: Minus	Field Indicators: Zero or Blank	Sterling Sign Position
01	I	FILE1	AA			03	80		C1																						
02	I																				1	5	0	FLD1							
03	I																				6	10	0	FLD2							
04	I																				11	15	0	FLD3							
05	I																				16	20	0	FLD4							
06	I																				21	25	0	FLD5							
07	I																				26	30	0	FLD6							
08	I																				31	35	0	FLD7							
09	I																				36	40	0	FLD8							
10	I																				41	45	0	FLD9							
11	I																				46	50	0	FLD10							
12	I		BB			04	80		C2																						
13	I																				1	5	0	FLD11							
14	I																				6	10	0	FLD12							



IBM

International Business Machines Corporation

Form X21-9093
Printed in U.S.A.

RPG CALCULATION SPECIFICATIONS

Date _____
Program _____
Programmer _____

Punching Instruction	Graphic								
	Punch								

Page 1 2

Program Identification 75 76 77 78 79 80

Line	Form Type	Control Level (L0-L9, LR, SR)	Indicators: And Not	Indicators: And Not	Indicators: And Not	Factor 1	Operation	Factor 2	Result Field	Field Length	Decimal Positions	Half Adjust (H)	Resulting Indicators: Arithmetic Plus	Resulting Indicators: Arithmetic Minus	Resulting Indicators: Arithmetic Zero	Resulting Indicators: Compare High 1>2	Resulting Indicators: Compare Low 1<2	Resulting Indicators: Compare Equal 1=2	Resulting Indicators: Lookup Table (Factor 2) is High	Resulting Indicators: Lookup Table (Factor 2) is Low	Resulting Indicators: Lookup Table (Factor 2) is Equal	Comments
01	C		03				MOVE	FLD1	AR1,1													
02	C		03				MOVE	FLD2	AR1,2													
03	C		03				MOVE	FLD3	AR1,3													
04	C		03				MOVE	FLD4	AR1,4													
05	C		03				MOVE	FLD5	AR1,5													
06	C		03				MOVE	FLD6	AR1,6													
07	C		03				MOVE	FLD7	AR1,7													
08	C		03				MOVE	FLD8	AR1,8													
09	C		03				MOVE	FLD9	AR1,9													
10	C		03				MOVE	FLD10	AR1,10													
11	C		04				MOVE	FLD11	AR1,11													
12	C		04				MOVE	FLD12	AR1,12													

Figure 10-8. Building an Array Using Straight Substitution

Blanks and other fields can appear on the input records since the From and To entries identify the location of the field. The array fields need not be in order. They can be ordered in the calculation specifications.

The disadvantages of this method are the amount of coding required for entering a large array and the amount of core required to set up the array.

Example 2: Figure 10-9 illustrates a method of loading an array using indexed entries on the input records. Once again the example shows a 12-element array with element length of 5; however, because of the record identifying indicator the array could be made larger without using additional coding. This is accomplished by assigning a different value to I1, I2, etc. on each input record. Each record that turned on 03 would then load another 10 elements into array AR. The short record (end of the array) turns on indicator 04 in the example.

Blanks and other fields can appear on the input records since the array elements and their index are identified by From and To entries.

While this method requires a minimum of coding and no calculations to set up the array, a considerable amount of work is required to set up the indexing scheme for the input records.

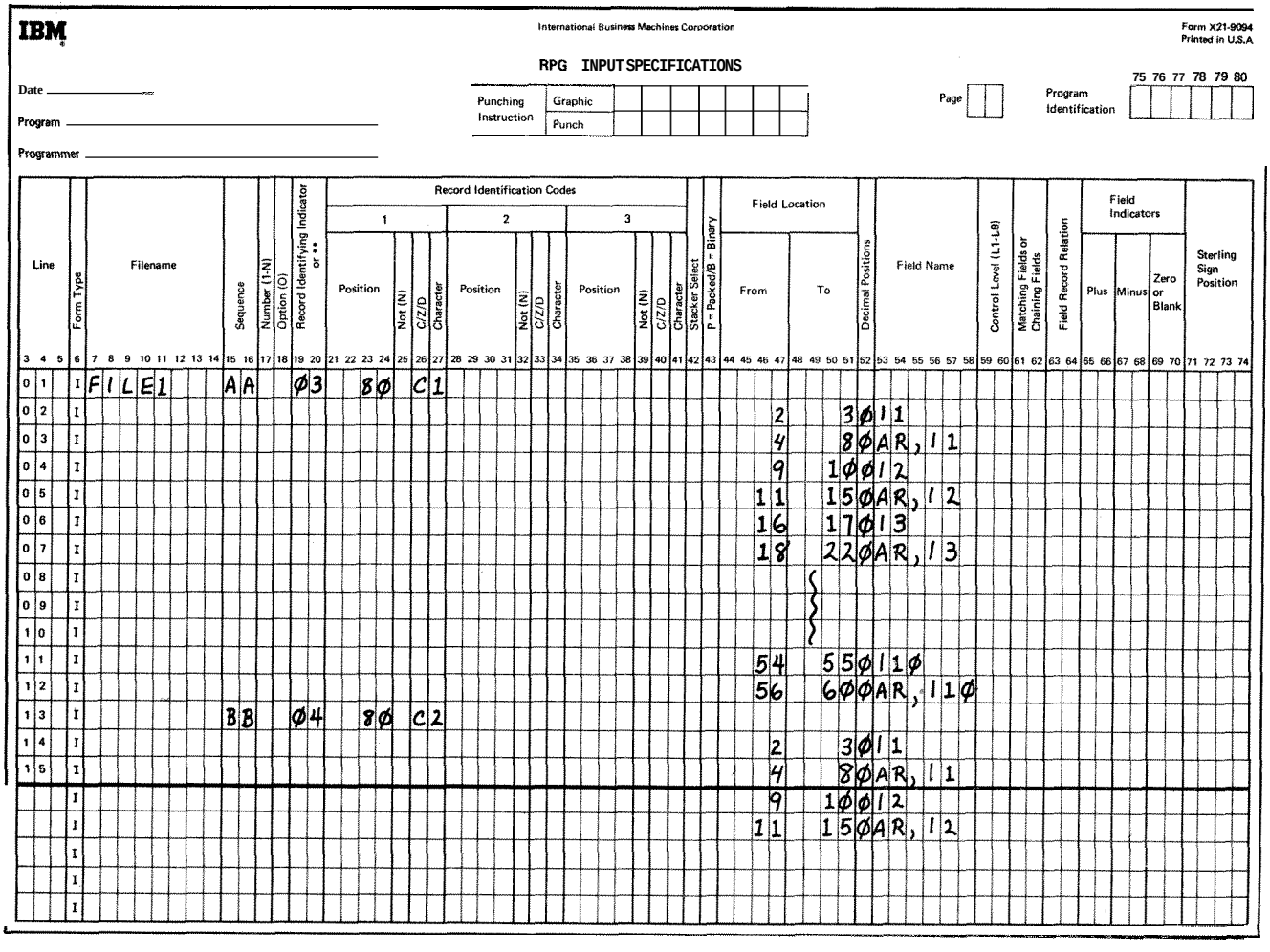


Figure 10-9. Building an Array Using Input Fields as Indexes

Example 3: *In* Figure 10-10 we see a method whereby eighteen 5-position elements of array AR1 are loaded with only 2 specification lines. No blanks or non-array information can appear on the input record. On succeeding lines of the Input sheet other elements of AR1 are loaded one after another until the array is full. Each additional element is coded on a separate line. Each new record requires a separate means of identification. For example, if another 03 record followed the first, the fields on the second record would overlay the fields read in from the first record.

The method illustrated in Example 3 works well for small arrays.

Example 4: The best method of loading large arrays requires the building of two arrays, one to contain the fields read in from a single record, the other to hold the entire array. In Figure 10-11 we are loading a 22-element array, each element being five positions long. Ten elements are read in from each record. Blanks or other fields cannot be mixed with the array information.

On the Extension sheet define two arrays:

AR1 - Ten elements, five positions long, zero decimal positions.

AR2 - Twenty-two elements, five positions long, zero decimal positions.

The first record read must be identified by a I in position 80. This turns on resulting indicator 03 and initializes result field IN which will be used to contain the count of all records read. As shown in the Calculation sheet in Figure 10-11, the following actions then occur:

1. IN is set to 1 and resulting indicator 04 is set on.
2. IN1 is set to 1.
3. The contents of element 1 of AR1 is moved into element 1 of AR2.
4. 1 is added to IN. IN now contains 2.
5. IN is compared to 22. Since it is less than 22, continue.
6. 1 is added to IN1. IN1 now contains 2.
7. IN1 is compared to 10. Since it is less than 10, continue.
8. The conditions of line 10 (04 on, 55 not on) are met. Go to T1.
9. The second element or AR1 (IN1=2) is moved into the second element of AR2 (IN=2).

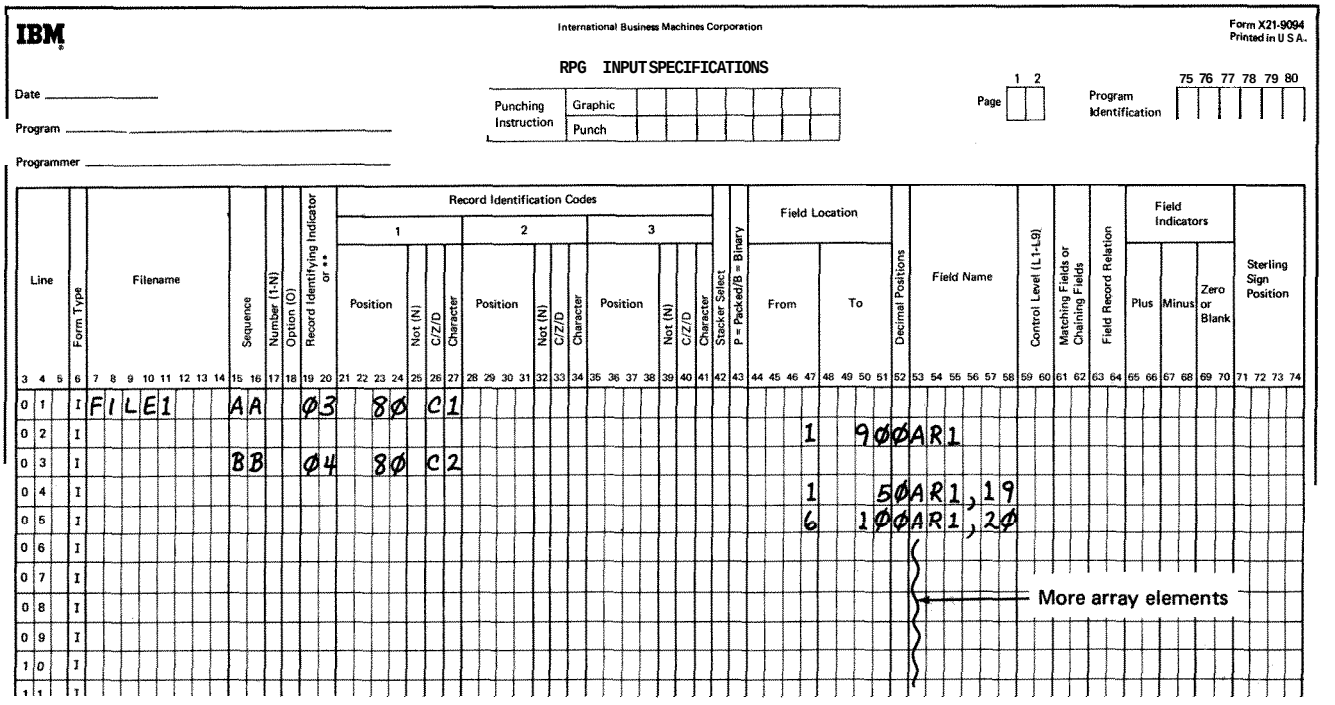


Figure 10-10. Building an Array Using Fixed Indexes

The program continues through the same calculations moving all the fields read into AR1 from the first record into AR2. When the tenth field is moved, IN and IN1 contain 11 (they were originally initialized to 1). On this pass, the compare on line 9 causes indicator 55 to turn on. Ten array elements have been read and moved into processing position.

A new record with a 2 in column 80 must now be read. This turns on indicator 04 and starts the calculations. Note that result field IN is not initialized for this pass. It contained 11 when the second record was read and will be stepped to 21 when the 10 fields from the second record are loaded.

IBM

International Business Machines Corporation

Form X21-9094
Printed in U.S.A.

RPG INPUT SPECIFICATIONS

Date _____
Program _____
Programmer _____

Punching Instruction	Graphic								
	Punch								

Page 1 2

Program Identification 75 76 77 78 79 80

Line	Form Type	Filename	Sequence	Number (1-N)	Option (O)	Record Identifying Indicator or **	Record Identification Codes 1: Position	1: Not (N)	1: C/Z/D	1: Character	2: Position	2: Not (N)	2: C/Z/D	2: Character	3: Position	3: Not (N)	3: C/Z/D	3: Character	Stacker Select	P = Packed/B = Binary	Field Location: From	Field Location: To	Decimal Positions	Field Name	Control Level (L1-L9)	Matching Fields or Chaining Fields	Field Record Relation	Field Indicators: Plus	Field Indicators: Minus	Field Indicators: Zero or Blank	Sterling Sign Position
3 4 5	6	7 8 9 10 11 12 13 14	15 16	17	18	19 20	21 22 23 24	25	26	27	28 29 30 31	32	33	34	35 36 37 38	39	40	41	42	43	44 45 46 47	48 49 50 51	52	53 54 55 56 57 58	59 60	61 62	63 64	65 66	67 68	69 70	71 72 73 74
0 1	I	ARRFILE	AA			Ø3	8Ø		C	1																				
0 2	I		OR			Ø4	8Ø		C	2																				
0 3	I																				1	5ØØAR1								
0 4	I																													
0 5	I																													

IBM

International Business Machines Corporation

Form X21-9093
Printed in U.S.A.

RPG CALCULATION SPECIFICATIONS

Date _____
Program _____
Programmer _____

Punching Instruction	Graphic								
	Punch								

Page 1 2

Program Identification 75 76 77 78 79 80

Line	Form Type	Control Level (L0-L9, LR, SR)	Indicators: And	Indicators: Not	Indicators: And	Indicators: Not	Indicators: And	Indicators: Not	Factor 1	Operation	Factor 2	Result Field	Field Length	Decimal Positions	Half Adjust (H)	Resulting Indicators: Arithmetic Plus	Arithmetic Minus	Arithmetic Zero	Compare High 1>2	Compare Low 1<2	Compare Equal 1=2	Lookup Table (Factor 2) is High	Lookup Low	Lookup Equal	Comments
3 4 5	6	7 8	9 10 11	12 13 14	15 16 17	18 19 20 21 22 23 24 25 26 27	28 29 30 31 32	33 34 35 36 37 38 39 40 41 42	43 44 45 46 47 48	49 50 51	52	53	54 55	56 57	58 59	60 61 62 63 64 65 66 67 68 69 70 71 72 73 74
0 1	C		Ø3				Z-ADD	1	IN	2Ø			Ø4												
0 2	C		Ø4				Z-ADD	1	IN1	2Ø															
0 3	C					T1	TAG																		
0 4	C		Ø4				MOVE	AR1,IN1	AR2,IN																
0 5	C		Ø4			IN	ADD	1	IN																
0 6	C		Ø4			IN	COMP	22																	
0 7	C		Ø4	12			GOTO	T2																	
0 8	C		Ø4			IN1	ADD	1	IN1																
0 9	C		Ø4			IN1	COMP	1Ø																	
1 0	C		Ø4N55				GOTO	T1																	
1 1	C					T2	TAG																		
1 2	C																								

Figure 10-11. Building a Functional Array From an Input Array

In this example we are building a 22-field array; therefore, another record containing the last two elements must be read in. This record must also contain a 2 in column 80. After the second field of this record has been transferred into AR2, IN will contain 23. The compare operation on line 6 will turn on resulting indicator 12. The conditions on line 7 are now met and we will go to T2. The array (AR2) is completely built.

The calculations shown for this example can be used to build any size array. The only changes required are:

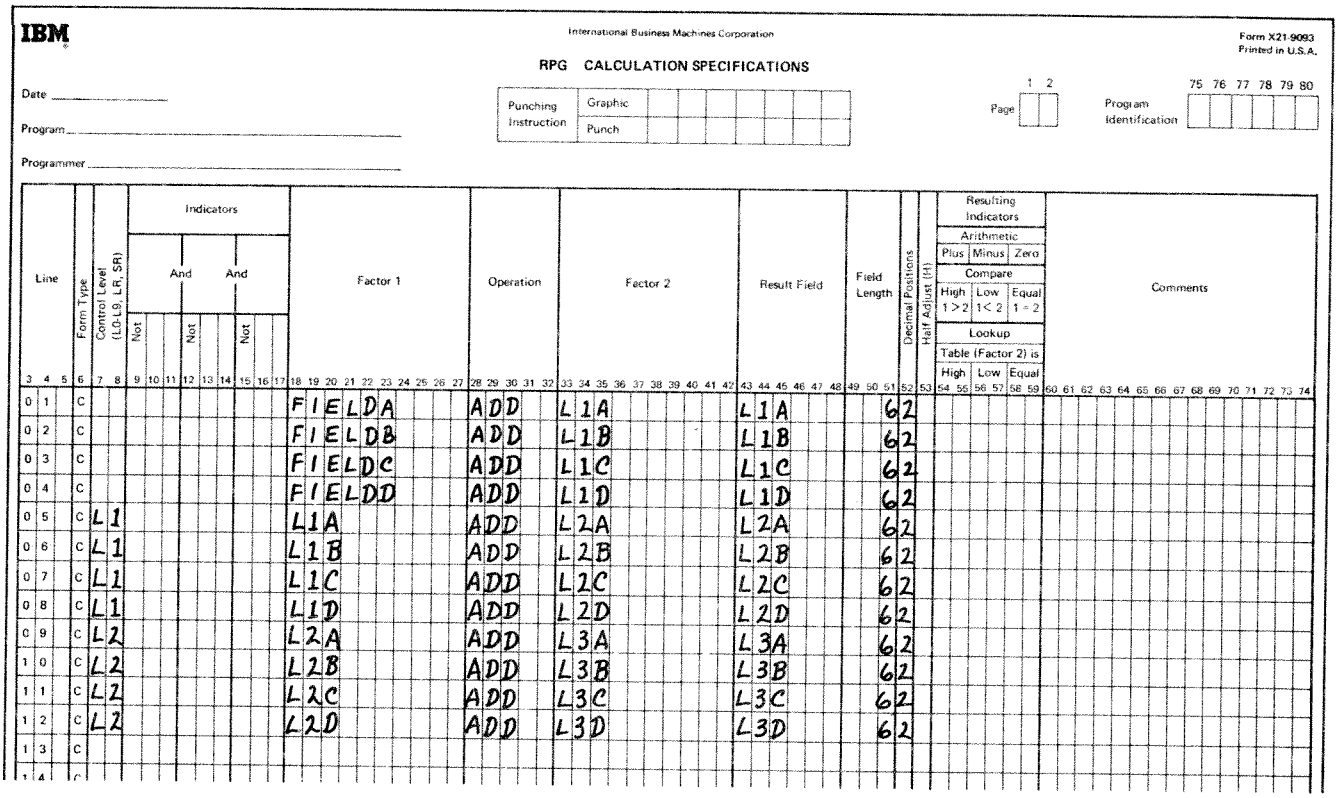
1, Line 06. Change Factor 2 to the total number of elements in the array you are building.

2. Line 09. Change Factor 2 to the total number of fields you are reading from each record. (If the last record contains less than this amount, the total array size in line 06 will handle the short record.)

Use any allowable record identification codes and resulting indicators.

Example 5: The specifications in Figure 10-12 perform the function of tabulating three levels of totals. The fields FIELDA, FIELDB, FIELDC, and FIELDD are added, as they are read from input records, to the first level totals L1A, L1B, L1C, and L1D. These first level totals are added at the time of an L1 control break to totals L2A, L2B, L2C, and L2D. Similarly, at an L2 control break the second level totals are added to third level totals L3A, L3B, L3C, and L3D. In addition, as control breaks occur, L1, L2, and L3 total output is performed; total fields are zeros after they are written on the output device.

Now, Figure 10-13 shows the same functions being performed using arrays. Note the reduction in coding required to specify the function. For example, line 5 of the Calculation sheet performs the same function as lines 5 through 8 of the Calculation sheet of Figure 10-12. Similarly, the output specifications are reduced from 15 lines to 6.



IBM

International Business Machines Corporation

Form X21-9093
Printed in U.S.A.

RPG CALCULATION SPECIFICATIONS

Date _____

Program _____

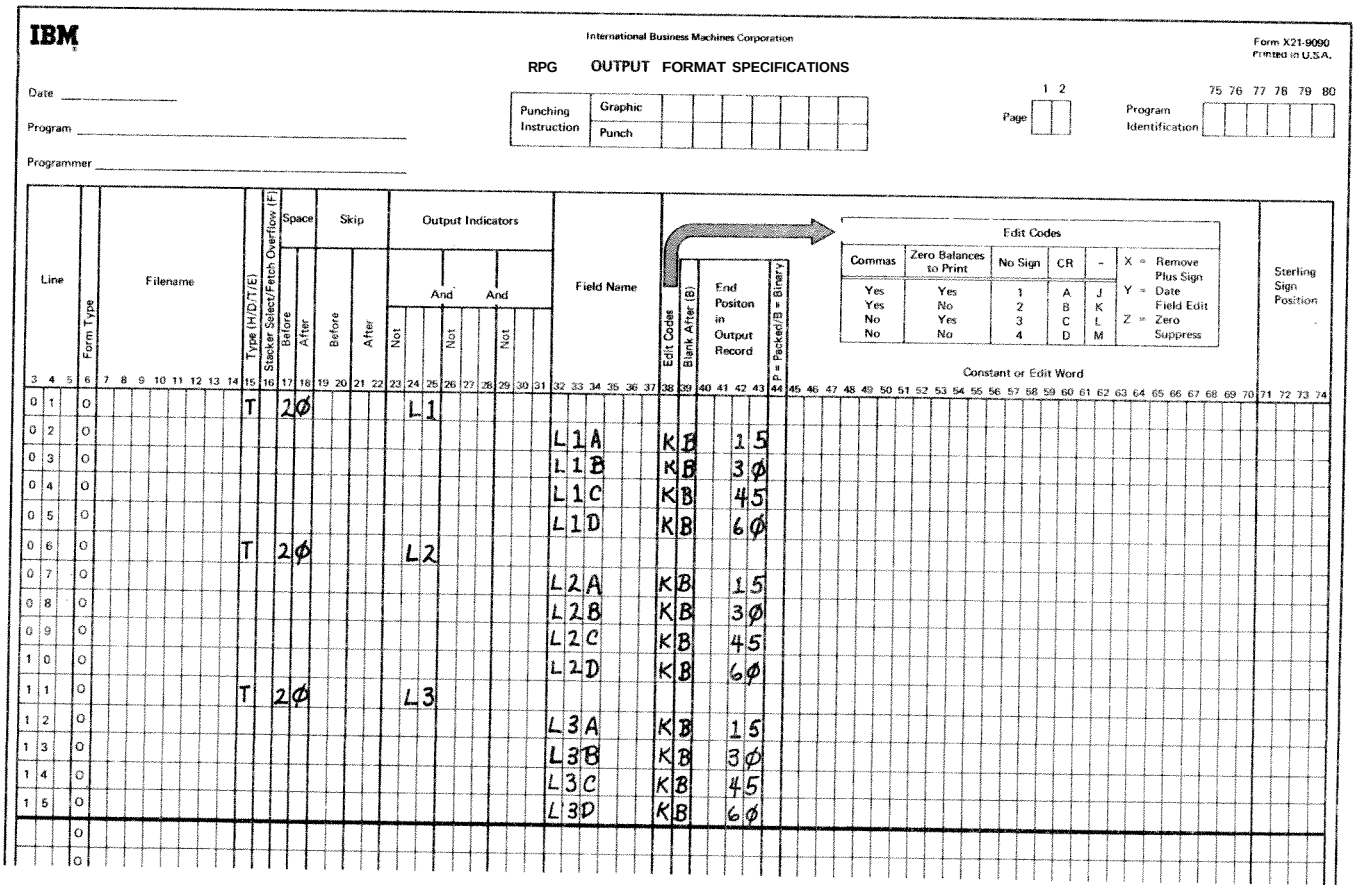
Programmer _____

Punching Instruction: Graphic / Punch

Page: 1 2

Program Identification: 75 76 77 78 79 80

Line	Form Type	Control Level (L0-L9, LR, SR)	Indicators: And / Not	Indicators: And / Not	Indicators: And / Not	Factor 1	Operation	Factor 2	Result Field	Field Length	Decimal Positions	Half Adjust (H)	Resulting Indicators: Arithmetic: Plus	Minus	Zero	Compare: High 1>2	Low 1<2	Equal 1=2	Lookup: Table (Factor 2) is: High	Low	Equal	Comments
01	C					FIELDA	ADD	L1A	L1A	62												
02	C					FIELDB	ADD	L1B	L1B	62												
03	C					FIELDC	ADD	L1C	L1C	62												
04	C					FIELDD	ADD	L1D	L1D	62												
05	C	L1				L1A	ADD	L2A	L2A	62												
06	C	L1				L1B	ADD	L2B	L2B	62												
07	C	L1				L1C	ADD	L2C	L2C	62												
08	C	L1				L1D	ADD	L2D	L2D	62												
09	C	L2				L2A	ADD	L3A	L3A	62												
10	C	L2				L2B	ADD	L3B	L3B	62												
11	C	L2				L2C	ADD	L3C	L3C	62												
12	C	L2				L2D	ADD	L3D	L3D	62												
13	C																					
14	C																					



IBM

International Business Machines Corporation

Form X21-9090
Printed in U.S.A.

RPG OUTPUT FORMAT SPECIFICATIONS

Date _____

Program _____

Programmer _____

Punching Instruction: Graphic / Punch

Page: 1 2

Program Identification: 75 76 77 78 79 80

Line	Form Type	Filename	Type (H/D/T/E)	Stacker Select/Fetch Overflow (F)	Space: Before	Space: After	Skip: Before	Skip: After	Output Indicators: And: Not	Output Indicators: And: Not	Output Indicators: And: Not	Field Name	Edit Codes	Blank After (B)	End Position in Output Record	P = Packed/B = Binary
01	O		T		2	Ø			L1							
02	O											L1A	K	B	15	
03	O											L1B	K	B	3Ø	
04	O											L1C	K	B	45	
05	O											L1D	K	B	6Ø	
06	O		T		2	Ø			L2							
07	O											L2A	K	B	15	
08	O											L2B	K	B	3Ø	
09	O											L2C	K	B	45	
10	O											L2D	K	B	6Ø	
11	O		T		2	Ø			L3							
12	O											L3A	K	B	15	
13	O											L3B	K	B	3Ø	
14	O											L3C	K	B	45	
15	O											L3D	K	B	6Ø	
	O															
	O															

Edit Codes

Commas	Zero Balances to Print	No Sign	CR	-	
Yes	Yes	1	A	J	X = Remove Plus Sign
Yes	No	2	B	K	Y = Date Field Edit
No	Yes	3	C	L	Z = Zero Suppress
No	No	4	D	M	

Constant or Edit Word

Sterling Sign Position

Figure 10-12. Calculating Totals Without Arrays

IBM

International Business Machines Corporation

Form X21-9091
Printed in U.S.A.

RPG EXTENSION AND LINE COUNTER SPECIFICATIONS

Date _____

Program _____

Programmer _____

Punching Instruction	Graphic								
	Punch								

Page 1 2

Program Identification 75 76 77 78 79 80

Extension Specifications

Line	Form Type	Record Sequence of the Chaining File / Number of the Chaining Field / From Filename	To Filename	Table or Array Name	Number of Entries per Record	Number of Entries Per Table or Array	Length of Entry	P = Packed/B = Binary	Decimal Positions	Table Sequence (A/D)	Table or Array Name (Alternating Format)	Length of Entry	P = Packed/B = Binary	Decimal Positions	Table Sequence (A/D)	Comments
0 1	E			SL1			4	6	2							
0 2	E			SL2			4	6	2							
0 3	E			SL3			4	6	2							
0 4	E															

IBM

International Business Machines Corporation

Form X21-9093
Printed in U.S.A.

RPG CALCULATION SPECIFICATIONS

Date _____

Program _____

Programmer _____

Punching Instruction	Graphic								
	Punch								

Page 1 2

Program Identification 75 76 77 78 79 80

Line	Form Type	Control Level (LO-L9, LR, SR)	Indicators And Not	Indicators And Not	Indicators And Not	Factor 1	Operation	Factor 2	Result Field	Field Length	Decimal Positions	Half Adjust (H)	Resulting Indicators: Arithmetic Plus	Arithmetic Minus	Arithmetic Zero	Compare High 1>2	Compare Low 1< 2	Compare Equal 1 = 2	Lookup Table (Factor 2) is High	Lookup Table (Factor 2) is Low	Lookup Table (Factor 2) is Equal	Comments
0 1	C					FIELDA	ADD	SL1,1	SL1,1													
0 2	C					FIELDB	ADD	SL1,2	SL1,2													
0 3	C					FIELDC	ADD	SL1,3	SL1,3													
0 4	C					FIELDD	ADD	SL1,4	SL1,4													
0 5	C	L1				SL1	ADD	SL2	SL2													
0 6	C	L2				SL2	ADD	SL3	SL3													
0 7	C																					

IBM

International Business Machines Corporation

Form X21-9090
Printed in U.S.A.

RPG OUTPUT - FORMAT SPECIFICATIONS

Date _____

Program _____

Programmer _____

Punching Instruction	Graphic								
	Punch								

Page 1 9

Program Identification 75 76 77 78 79 80

Line	Form Type	Filename	Type (H/D/T/E)	Stacker Select/Fetch Overflow (F)	Space Before	Space After	Skip Before	Skip After	Output Indicators And Not	Output Indicators And Not	Output Indicators And Not	Field Name	Edit Codes	Blank After (B)	End Position in Output Record	P = Packed/B = Binary	Constant or Edit Word	Sterling Sign Position
0 1	O		T		2	Ø			L1									
0 2	O											SL1	K	B	6Ø			
0 3	O		T		2	Ø			L2									
0 4	O											SL2	K	B	6Ø			
0 5	O		T		2	Ø			L3									
0 6	O											SL3	K	B	6Ø			
0 7	O																	

Edit Codes

Commas	Zero Balances to Print	No Sign	CR	-	
Yes	Yes	1	A	J	X = Remove Plus Sign
Yes	No	2	B	K	Y = Date Field Edit
No	Yes	3	C	L	Z = Zero Suppress
No	No	4	D	M	

Figure 10-13. Calculating Totals With Arrays

Example 6: This example illustrates the use of three arrays defined as follows. Refer to Figure 10-14.

Array Name	*Number of Fields*	*Field Length*
ARA	4	5
ARB	5	10
ARC	6	4

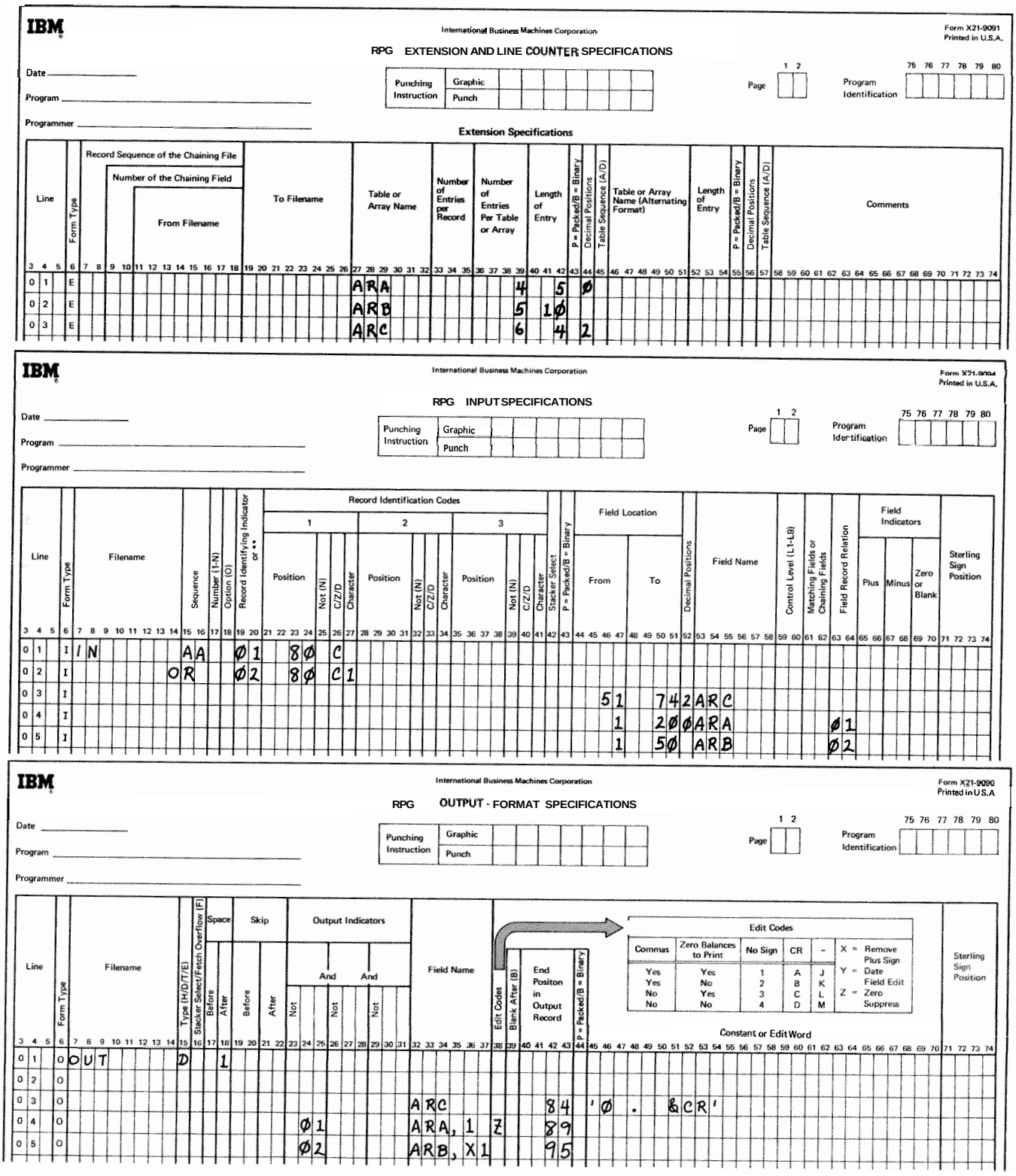


Figure 10-14. Using Arrays to Format Field Output

Array ARA is contained in the input records corresponding to indicator 01, ARB in the records corresponding to 02, and ARC in both types of records. Array ARC and the first field of array ARA are to be included together in an output record as are arrays ARC and a field (identified by field X1) of array ARB. Every field in array ARC is edited according to the edit word OB.bb&CR. (where b represents a blank).

Assume that the contents of the arrays in the first two input records are:

Record	*Array*	*Array Contents*
1	ARA	12345678901234567890
	ARC	012345678901234567898 76N (note that N equals minus 5)
2	ARB	JOHNbDOEbbJOEbSMITHbLEEb MARXbbJIMbKNOTSbTIMbTYLERb
	ARC	(The same as in record 1)

In the first output record, the location and contents of the arrays are (b represents a blank):

Array	*Location*	*Contents*
ARA (fist field)	85-89	12345
ARC	37-84	b1.23bbb45.67bbb 89.01.bbb23.45bbb 67.89bbb87.65bCR

For the second output record assume that the contents of field X1 is 4. The locations and contents of the arrays are:

Array	*Location*	*Contents*
ARB (fourth field)	86-95	JIMbKNOTSb
ARC	37-84	The same as in the first record.

Example 7: Figure 10-15 shows a method of writing short arrays on the output device. The contents of one element of a 22-element array, AR2, is written to the output file ARFILE each time the specification in line 3 of the Calculation sheet is performed.

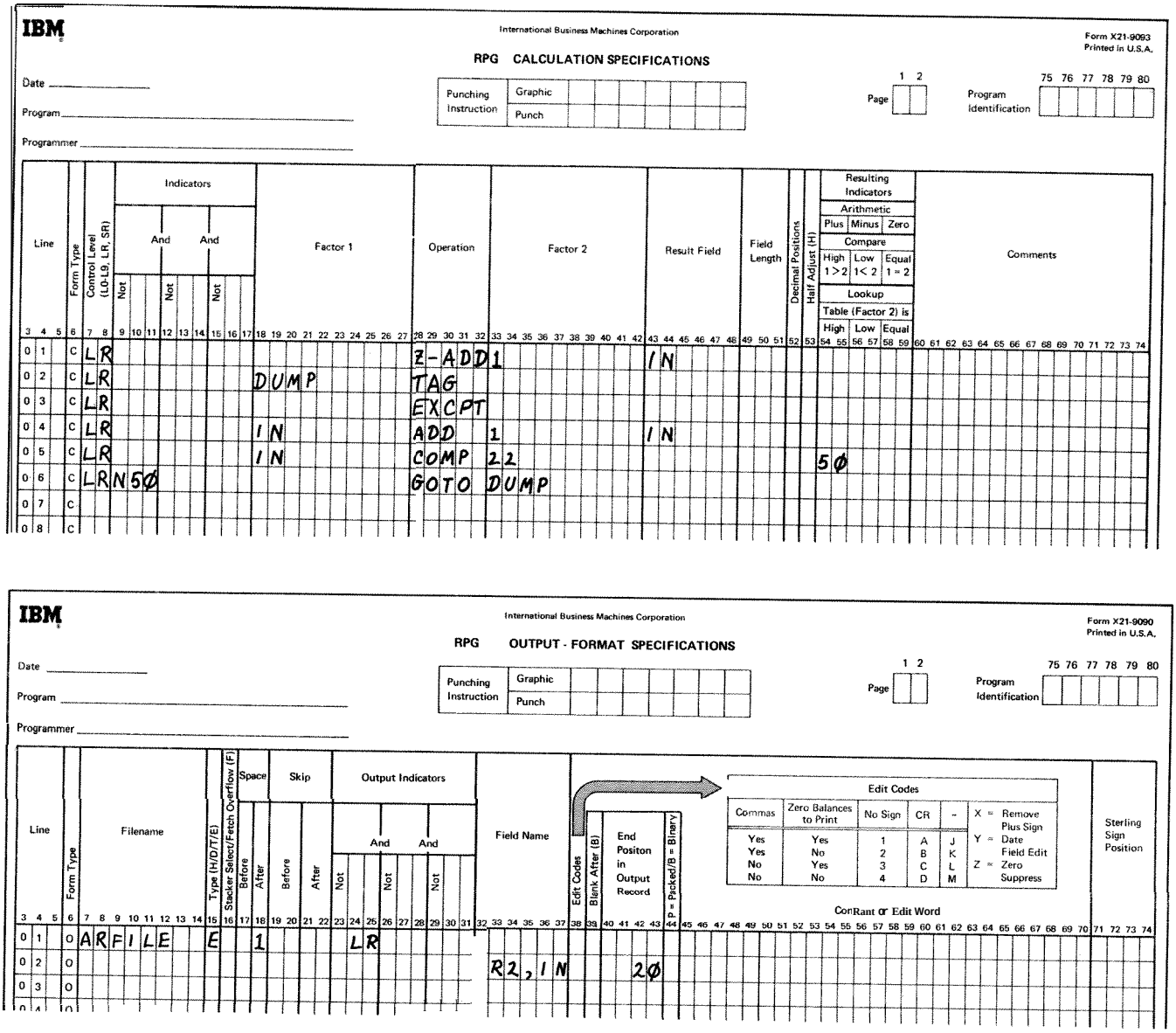
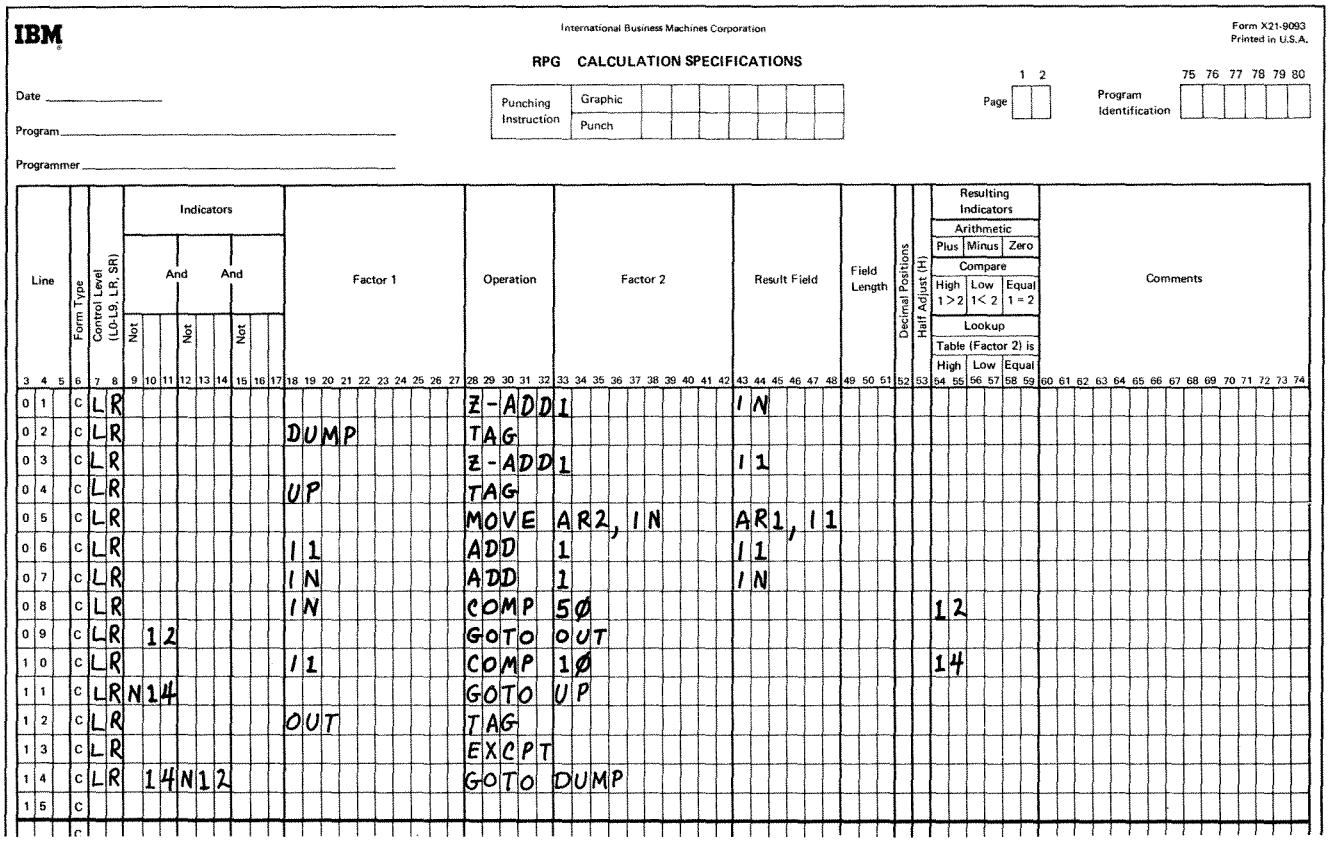


Figure 10-15. Printing One Array Element Per Line

Example 8: Figure 10-16 shows a method of writing a large array on the output device. The number of fields printed on a line depends on the value assigned to the compare on line 10 of the Calculation sheet. If an edit code is used, each array field will be separated by two spaces. These spaces must be considered when computing the end position in the output specifications.



IBM

International Business Machines Corporation

Form X21-9093
Printed in U.S.A.

RPG CALCULATION SPECIFICATIONS

Date _____

Program _____

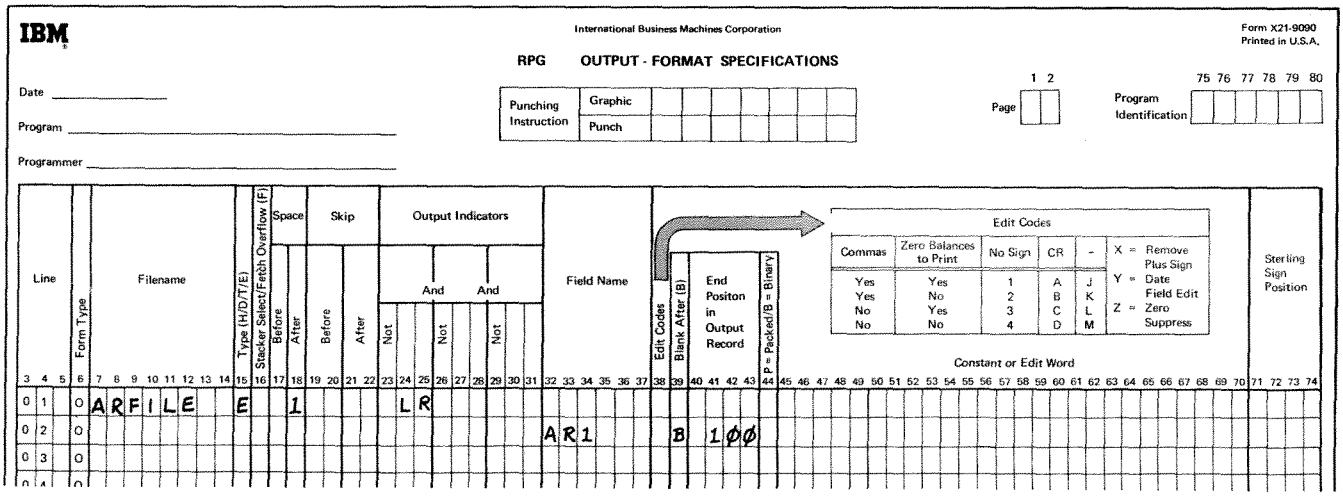
Programmer _____

Punching Instruction: Graphic / Punch

Page 1 2

Program Identification 75 76 77 78 79 80

Line	Form Type	Control Level (L0-L9, LR, SR)	Indicators: And / And — Not	Factor 1	Operation	Factor 2	Result Field	Field Length	Decimal Positions	Half Adjust (H)	Resulting Indicators: Arithmetic (Plus / Minus / Zero); Compare (High 1>2 / Low 1<2 / Equal 1=2); Lookup (Table (Factor 2) is High / Low / Equal)	Comments
01	C	LR			Z-ADD 1		IN					
02	C	LR		DUMP	TAG							
03	C	LR			Z-ADD 1		I1					
04	C	LR		UP	TAG							
05	C	LR			MOVE	AR2,IN	AR1,I1					
06	C	LR		I1	ADD	1	I1					
07	C	LR		IN	ADD	1	IN					
08	C	LR		IN	COMP	50					12	
09	C	LR	12		GOTO	OUT						
10	C	LR		I1	COMP	10					14	
11	C	LR N14			GOTO	UP						
12	C	LR		OUT	TAG							
13	C	LR			EXCPT							
14	C	LR	14N12		GOTO	DUMP						
15	C											
	C											



IBM

International Business Machines Corporation

Form X21-9090
Printed in U.S.A.

RPG OUTPUT - FORMAT SPECIFICATIONS

Date _____

Program _____

Programmer _____

Punching Instruction: Graphic / Punch

Page 1 2

Program Identification 75 76 77 78 79 80

Line	Form Type	Filename	Type (H/D/T/E)	Stacker Select/Fetch Overflow (F)	Space: Before / After	Skip: Before / After	Output Indicators: And / And — Not	Field Name	Edit Codes	Blank After (B)	End Position in Output Record	P = Packed/B = Binary	Constant or Edit Word	Sterling Sign Position
01	O	ARFILE	E		1		LR							
02	O							AR1		B	100			
03	O													
	O													

Edit Codes

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

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

Figure 10-16. Printing More Than One Array Element Per Line

CHARACTER STRUCTURE

Character Grouping by Zone or Digit

When selecting characters for record identification purposes on a digit or zone only basis, it must be understood that all characters having the same zone or digit will be selected by the system as meeting record ID requirements. The reason is that when a character is read into the system it is converted into an 8-bit code. It is the 8-bit code that is tested to see if the character meets the requirements of the record identifying character on the input specifications. Figure 10-17 lists the character grouping for zone or digit only entries in the Character/Zone/Digit columns (26, 33 or 40) and character columns (27, 34, or 41) of the input specifications.

As an example, a digit only entry in C/Z/D and an ***A*** in character would cause all records having a / (slash), ***A,*** J, or **1** in the specified column to be selected.

Using the same letter ***A*** but now selecting records on a zone only basis, $ and ***A-I*** meet the requirements and are selected.

CHARACTER GROUPING BY ZONE (Z)		CHARACTER GROUPING BY DIGIT (D)	
blank ¢ . < (+ |	& A B C D E F G H I	H Q Y 8	blank & -] } 0
! $ *) ; ¬	- \ J K L M N O P Q R	I R Z 9	/ A J 1
/ , % _ > ?	S T U V W X Y Z	¢ I :	B K S 2
: # @ ' = "	0 1 2 3 4 5 6 7 8 9	. $, #	C L T 3
		< * % @	D M U 4
		( ) _ '	E N V 5
		+ ; > =	F O W 6
		| ¬ ? "	G P X 7

Figure 10-17. Characters Interpreted as Having the Same Zone or Digit

Negative Number

Negative numbers have a different character structure than positive numbers because negative numbers are formed by combining a minus sign with the number. Numbers 0-9 have only digit portions. A minus sign is B zone entry. Thus when the zone (minus sign) and the digit (0-9) are put together, a letter is formed. Therefore, negative numbers are represented in the computer by the characters J-R.

EDITING

To edit a field means to punctuate it by adding commas, decimal points, negative value signs, dollar signs or constant information. Data should be edited before it is printed if it is to be understandable. For example, take the following unedited data:

00367964

Is the amount 367,964? Possibly, but many figures are dollars and cents and are edited with two decimal positions. A decimal point is automatic with most edit codes. The number of decimal positions is defined on the input or calculation sheet. For this example, let's assume two decimal positions. If the unedited data is edited with an edit code it can look like this:

3,679.64

If it is edited by an edit word, you can make it look like this:

$**3,679.64

The decimal point prints only on an edited field. You can edit a field by using an edit code or an edit word.

Note: If the inverted print option is specified on the RPG II control card specifications for the job, the edited data is converted according to the option selected (see ***Column*** 21, ***Inverted Print*** in Chapter ***3)***.

Edit Codes

The use of codes is the easiest and most commonly used method of editing. You simply enter the code that you want in column **38** of the Output-Format sheet. The available codes are printed in a table above columns 45-70 of the Output-Format sheet. Figure 10-18 further illustrates these codes and the options they provide. Figure 10-19 illustrates how data looks when it is edited by edit codes. Each code punctuates the field a little differently. **All** codes suppress leading zeros, with the following exception. One of the World Trade formats for output is a J entry in column 21 in the control card specifications. For this J entry, all zero balances and balances with zero values to the left of the decimal comma are always written or punched with one leading zero (such as 0,00 or 0,04). The J entry overrides any edit codes that might suppress the leading zero; that is, the leading zero for the J entry cannot be suppressed by any edit codes in the cases mentioned.

Edit Code	Commas	Decimal Point	Sign For Negative Balance			Print Out On Zero Balance *		Zero Suppress
			No Sign	CR	– (Minus)	Domestic United Kingdom World Trade I	World Trade J	
1	Yes	Yes	No Sign			.00 or 0	0.00 or 0	Yes
2	Yes	Yes	No Sign			Blanks	Blanks	Yes
3		Yes	No Sign			.00 or 0	0,00 or 0	Yes
4		Yes	No Sign			Blanks	Blanks	Yes
A	Yes	Yes		CR		.00 or 0	0,00 or 0	Yes
B	Yes	Yes		CR		Blanks	Blanks	Yes
C		Yes		CR		.00 or 0	0,00 or 0	Yes
D		Yes		CR		Blanks	Blanks	Yes
J	Yes	Yes			–	.00 or 0	0,00 or 0	Yes
K	Yes	Yes			–	Blanks	Blanks	Yes
L		Yes			–	.00 or 0	0.00 or 0	Yes
M		Yes			–	Blanks	Blanks	Yes
X **								
Y ***								Yes
Z								Yes

* Zero balances for the World Trade format are printed or punched in two ways, depending on the entry made in column 21 of the control card specifications.

** The X code performs no editing.

*** The Y code suppresses the leftmost zero only. The Y code edits a three to six digit field according to the following pattern:

nnfn
nn/nn
nn/nn/n
nnlnnlnn

Figure 10-18. **Edit Codes**

Normally, when you use an edit code in column 38 you cannot define an edit word in columns 45-70; however, there are two exceptions:

1. If you want leading zeros replaced by asterisks, enter '*' in columns 45-47 of the line containing the edit code.

2. If you want a dollar sign to appear before the first digit in the field (floating dollar sign), enter '$'in columns 45-47 of the line containing the edit code.

It is also possible to have a dollar sign appear before the asterisk fill (fixed dollar sign). This is accomplished in the following manner:

1. Place '*'in column 45-47 of the line containing the edit code.

2. Place '$' in columns 45-47 of the line following the edit code line. The end position of the field is required in both lines.

Edit Codes	Positive Number - Two Decimal Positions	Positive Number - No Decimal Positions	Negative Number - * Three Decimal Positions	Negative Number - * No Decimal Positions	Zero Balance - ** Two Decimal Positions		Zero Balance - No Decimal Positions
					Domestic United Kingdom World Trade I	World Trade J	
Unedited	1234567	1234567	00012 }	00012 }	000000	000000	000000
1	12,345.67	1,234,567	.120	120	.00	0,00	0
2	12,345.67	1,234,567	.120	120			
3	12345.67	1234567	.120	120	.00	0,00	0
4	12345.67	1234567	.120	120			
A	12,345.67	1,234,567	.120CR	120CR	.00	0,00	0
B	12,345.67	1,234,567	.120CR	120CR			
C	12345.67	1234567	.120CR	120CR	.00	0,00	0
D	12345.67	1234567	.120CR	120CR			
J	12,345.67	1,234,567	.120–	120–	.00	0,00	0
K	12,345.67	1,234,567	.120–	120–			
L	12345.67	1234567	.120–	120–	.00	0,00	0
M	12345.67	1234567	.120–	120–			
X	1234567	1234567	00012 }	00012 }	000000	000000	000000
Y				0/01/20			0/00/00
Z	1234567	1234567	120	120			

* The character } is a negative zero.
** Zero balances for the World Trade format are printed or punched in *two* ways, depending on the entry made in column 21 of the control card specifications.

Figure 10-19. **Examples** of **Edit Code Usage**

Edit Words

An edit word gives you more freedom in punctuating a numeric field than an edit code. You specify directly whether commas, decimal points and zero suppression are needed, if the negative sign should print and, if the output is dollars and cents, whether you want a dollar sign and leading asterisks. Figure 10-20 shows examples of edit words.

Edit words are entered in columns 45-70 of the Output-Format sheet. Each line that contains an edit word must have entries in columns 32-37 (Field Name) and columns 40-43 (End Position in Output Record). Column 38 (Edit Code) must *not* be used.

Editing Considerations

One important thing to keep in mind is that you must leave exactly enough room on the printed form for the edited word. If the field to be edited is seven characters long on the input card, consider whether seven positions allows enough space for it to print on the report. By the time the field is edited, it may contain many more characters than seven.

When computing the length of an edited output field determine how many of the editing characters are replaceable. The number of replaceable characters in the edit word must be equal to the length of the field to be edited. **A** replaceable character is a part of the edit word that does not require a position in the output file. The replaceable characters are:

- 0 (zero suppress)
- * (asterisk fill)
- b (blank)
- $ (when it appears immediately to the left of zero suppress: floating dollar sign)

Note: **An** extra space must be left in the edit word for the floating dollar sign. This ensures a print position for the dollar sign if the output field is full.

A fixed dollar sign, decimal points, floating dollar sign, commas, ampersands (representing blanks), negative signs (- or CR) and constant information all require space in the output field.

Certain combinations of edit characters are not allowed. For example, 'bb0bb0' is invalid. Only one zero suppress can be used: 'bb*bb0' is invalid but 'bb0bb*' is valid. The asterisk in the last example is considered a constant. The position of the zero suppress 0 is important. Consider the following:

'Obbbb'	If the field is five characters, the first position can be zero suppressed.
'Obbbbb'	If the field is five characters, no zero suppression will occur, because a six position edit word is represented.

Formatting Edit Words

The Printer Spacing Chart is helpful when forming edit words. Figure 10-21 shows how an output line can be formatted using this chart. Note that *X*'s and zeros are used

Unedited Data	Edit Words	Edited Data	Comments
	Constant or Edit Word 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		
00074326J	' , , , . '	000743261	Blank edit word removes negative sign from units position.
000000421	' , , , Ø '	421	Ø suppresses leading zeros.
24739000741	' , , , . , '	247,390,007.41	Two decimal positions.
000476222	' , , , . * '	***476,222	Asterisk fill.
02442	' $ Ø . '	$24.42	Floating dollar sign.
00002	' $. '	$ .02	Fixed dollar sign.
00022385	' $, * . '	$***223.85	Fixed dollar sign with asterisk fill.
72432N	' , $ Ø . b - '	$7,243.25 -	Dollar amount followed by space and minus sign.
72432N	' , $ Ø . C R '	$7,243.25CR	Dollar amount followed by CR minus symbol.
000615	' L B S . , Ø b O Z S . '	6 LBS.,15 OZS.	Editing text and figures.
2332	' $. b O L D b B A L '	$23.32 OLD BAL	Editing with constant information.
000015	' L B S . , Ø b O Z S . '	15 OZS.	Characters LBS., are constants in the edit body and are zero suppressed.

ART: 55015

Figure 10-20. Examples of Edit Words

Unedited Data

Item number - 000241
Item cost - 02000
Selling price - 02200
% profit or loss - 25

Printer Spacing Chart

	0									1									2									3									4								
	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																				P	R	I	C	E		L	I	S	T																			
3																																																
4																																																
5		I	T	E	M						I	T	E	M						S	E	L	L	I	N	G					P	E	R	C	E	N	T									
6		N	U	M	B	E	R				C	O	S	T						P	R	I	C	E							P	R	O	F	I	T		O	R		L	O	S	S			
7																																																
8		X	X	X	X	Y	Ø				$	X	X	Ø	.	X	X			$	X	X	Ø	.	X	X					X	X	C	R													
9																																																
10																																																
11																																																
12																																																

Output - Format Sheet

cators And	Not	Field Name	Edit Codes	Blank After (B)	End Positon in Output Record	P = Packed/B = Binery	Constant or Edit Word
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
		INUMB			7		' Ø'
		ICOST			18		'$ Ø. '
		$PRICE			27		'$ Ø. '
		PERCPL			37		' CR'

Edit Codes

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

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

Edited Output

241 $ 20.00 $ 22.00 25

Figure 10-21. Using the Printer Spacing Chart to Format Data

to show field positions. A zero indicates that all zeros to the left of this position (leading zeros) are to be suppressed. An *X* indicates any number can appear in the position. Use blanks in place of the *X*'s when writing the edit words. Two additional *X*'s are provided for Percent Profit or Loss since a negative value must be recognizable.

If it is necessary to show a negative number, a sign must be included in the edit word. You may use either the minus sign (-) or the letters *CR*. These print only for a negative number; however, the character positions they require must be taken into consideration when entering the end position of the field on the Output-Format sheet. Figure 10-21 shows that for the field PERCPL, CR is to be printed for a negative balance. Assume the field PERCPL contains the negative data 2N (-25%). The printed output looks like this:

25CR

If PERCPL was positive, CR would not print and the same field would appear as:

25

You may also use a minus sign to indicate a negative balance. If you want to leave a space between the number and the negative sign, place an ampersand (&) in the edit ward before the minus sign. PERCPL would then print as:

25 -

If you wish to have a dollar sign printed, you also indicate this in your edit word. To print a dollar sign at the left of the field called SPRICE, put the dollar sign ($) next to the first quote mark and then put in the necessary blanks and punctuation. A dollar sign in this position is called a fixed dollar sign The SPRICE field in Figure 10-22, Line A, can look like any of the following (N stands for any number):

$NNN.NN
$ NN.NN
$ N.NN
$ .NN

Suppose, however, you do not want a lot of empty space between the dollar sign and the first digit when zero suppression occurs. (This is commonly the case when writing checks.) You may fill in this empty space with asterisks (*). Instead of using *0* to indicate zero suppresion, you use the asterisk (*) to indicate that all extra spaces should be filled with asterisks. The SPRICE field in Figure 10-22, Line B, look like any of the following (N stands for any number):

$NNN.NN
$*NN.NN
$**N.NN
$***.NN

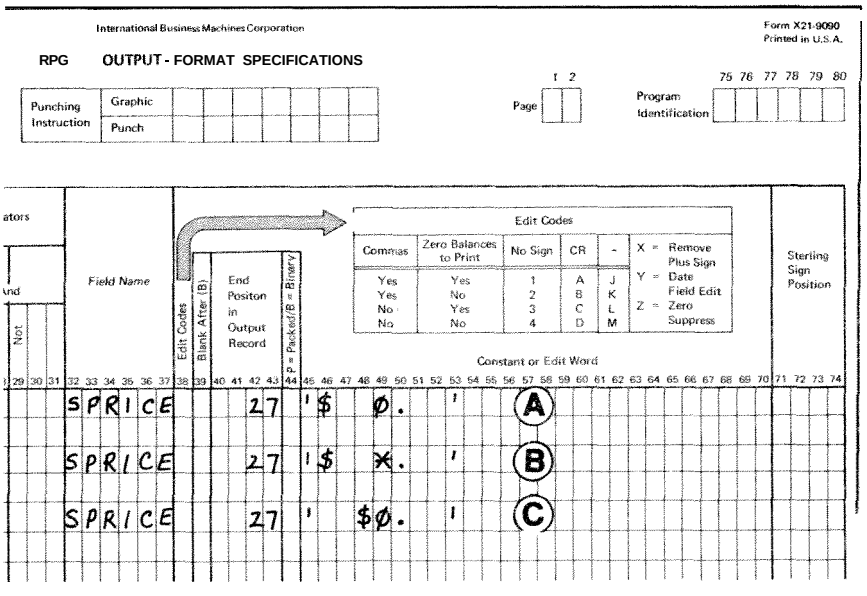


Figure 10-22. Different Edit Words Used **on** the Same Field

You may want the dollar sign to always be next to the left-most digit instead of filling in the space with asterisks or leaving extra blanks. This is indicated in the edit word by placing the $ next to the zero suppress 0. A dollar sign which changes positions depending upon the number of positions zero suppressed is known as a floating dollar sign. When printed, the SPRICE field in Figure 10-22, Line C, can look like any of the following:

```
$NNN.NN
 $NN.NN
  $N.NN
   $.NN
```

Note that an extra space must be left in the edit word for the floating dollar sign. This ensures a print position for the dollar sign if the output field is full.

FILE TRANSLATION

RPG II allows you to translate any character code into another character code. This capability is file translation.

Characters can be translated in input, output, update, and combined files. When update or combined fdes are translated, both the input and output portions of these files are translated.

A different character code used as input can be translated into the code used by System/3, and the code used by System/3 can be translated into a different code for output..

Specifications for File Translation

You must first indicate that there are files to be translated. Do this by entering an F in column 43 of the RPG II control card specifications. Table input cards must also be used to specify how the translation is to be done. The following entries are needed for each fde translation table input card used:

Columns 1-6: Enter *FILES to indicate that all input, output, update, and combined files are to undergo translation (both the input and output portions of update and combined files will be translated). Then use the specifications listed below, beginning with columns 9-10. *All* files will be translated according to the table specified beginning in column 9.

If only certain fdes are to be translated, they must be named individually in column 1-8 as follows:

Columns 1-8: Enter the fdename of the input, output, update, or combined file to be translated (both the input and output portions of update and combined files will be translated). Then use the specifications listed below, beginning with columns 9-10.

Columns 9-15: Enter the hexadecimal number corresponding to the character which is to be translated or replaced by another character.

Columns 11-12: Enter the hexadecimal number of the character taking the place of the character being translated (replaced).

Columns *13-1*6, 17-20, *and* 21-24, etc: These groups of columns are used the same way as columns 9-12 are used. The first two columns of a group give the character which is to be translated to the character named in the last two columns of a group.

All tables for one file must be kept together. The file translation table input deck must be preceded by a card with *** in columns 1-3. The remaining columns of this card may be used for comments. The file translation deck must directly follow the RPG specifications in the source program (see Figure 1-2).

Example

Assume that while working for a department store, you must process cards serving as sales slips for all items sold. Each card contains a punched and printed record of the actual, or wholesale, cost of its associated item along with a retail price.

Obviously, wholesale cost must remain confidential, and so the store uses individual letters of a code-name in place of numbers comprising wholesale costs.

A typical code-name generally consists of a combination of letters that can be easily remembered by the store's personnel. The only restriction, however, is that the code-name must contain ten different letters, one for each of the numbers zero through nine.

Using the code-name **BUCKINGHM** to represent numbers one through nine and zero, the letter B represents the number 1; letter U represents number 2, etc. Letter M represents zero. Individual letters are combined to represent each item's wholesale cost. Thus a wholesale cost of BBU.CC translates as 112.33; that is, one hundred twelve dollars and thirty-three cents.

In the following chart, hexadecimal equivalents of each letter in the word BUCKINGHAM are listed along with the hexadecimal equivalents of numbers one through nine and zero.

Letter in Code-name	*Hexadecimal Equivalent*	*Number*	*Hexadecimal Equivalent*
B	C2	*1*	F1
U	E4	*2*	F2
C	c 3	**3**	F3
K	D2	4	F4
I	C9	5	F5
N	D5	6	F6
G	c 7	7	F7
H	C8	8	F8
A	C1	9	F9
M	D4	0	FO

Hexadecimal equivalents are merely a different way of representing the 8-bit code that the computer examines to recognize individual characters in your language.

See Figure 10-23. Note that if letters BBU were read and never translated, hexadecimal equivalents C2, C2, and E4 would be used by System/3. As a result, it would be impossible to perform an arithmetic operation involving the wholesale cost, BBU. Therefore, with the aid of file translation, the computer replaces the letters BBU with numbers.

A file translation table input card specifications for letters in the word BUCKINGHAM is as follows:

Column	*Entry*
1-6	"FILES
7-8	Blank
9-12	C2F1
13-16	E4F2
17-20	C3F3
21-24	D2F4
25-28	C9F5
29-32	D5F6
33-36	C7F7
37-40	C8F8
41-44	C1F9
45-48	D4F0

Only the letters of the previous example will be specified for translation. All other characters will be handled in the normal manner.

International Business Machines Corporation

TRANSLATION TABLE AND ALTERNATE COLLATING SEQUENCE CODING SHEET

Entry	Replaced By
33	
34	
35	
36	
37	
38	
39	
3A	
3B	
3C	
3D	
3E	
3F	
40	
41	
42	
43	
44	
45	
46	
47	
48	
49	
4A	
4B	
4C	
4D	
4E	
4F	
50	
51	
52	
53	
54	
55	
56	
57	
58	
59	
5A	
5B	
5C	
5D	
5E	
5F	
60	
61	
62	
63	
64	
65	

Code	System/3 Graphic	Entry	Replaced By
01100110		66	
01100111		67	
01101000		68	
01101001		69	
01101010		6A	
01101011	,	6B	
01101100	%	6C	
01101101	—	6D	
01101110	>	6E	
01101111	?	6F	
01110000		70	
01110001		71	
01110010		72	
01110011		73	
01110100		74	
01110101		75	
01110110		76	
01110111		77	
01111000		78	
01111001		79	
01111010	:	7A	
01111011	#	7B	
01111100	@	7C	
01111101	'	7D	
01111110	=	7E	
01111111	"	7F	
10000000		80	
10000001		81	
10000010		82	
10000011		83	
10000100		84	
10000101		85	
10000110		86	
10000111		87	
10001000		88	
10001001		89	
10001010		8A	
10001011		8B	
10001100		8C	
10001101		8D	
10001110		8E	
10001111		8F	
10010000		90	
10010001		91	
10010010		92	
10010011		93	
10010100		94	
10010101		95	
10010110		96	
10010111		97	
10011000		98	

Code	System/3 Graphic	Entry	Replaced By
10011001		99	
10011010		9A	
10011011		9B	
10011100		9C	
10011101		9D	
10011110		9E	
10011111		9F	
10100000		A0	
10100001		A1	
10100010		A2	
10100011		A3	
10100100		A4	
10100101		A5	
10100110		A6	
10100111		A7	
10101000		A8	
10101001		A9	
10101010		AA	
10101011		AB	
10101100		AC	
10101101		AD	
10101110		AE	
10101111		AF	
10110000		B0	
10110001		B1	
10110010		B2	
10110011		B3	
10110100		B4	
10110101		B5	
10110110		B6	
10110111		B7	
10111000		B8	
10111001		B9	
10111010		BA	
10111011		BB	
10111100		BC	
10111101		BD	
10111110		BE	
10111111		BF	
11000000		C0	
11000001	A	C1	
11000010	B	C2	
11000011	C	C3	
11000100	D	C4	
11000101	E	C5	
11000110	F	C6	
11000111	G	C7	
11001000	H	C8	
11001001	I	C9	
11001010		CA	
11001011		CB	

Code	System/3 Graphic	Entry
11001100		CC
11001101		CD
11001110		CE
11001111		CF
11010000	}	D0
11010001	J	D1
11010010	K	D2
11010011	L	D3
11010100	M	D4
11010101	N	D5
11010110	O	D6
11010111	P	D7
11011000	Q	D8
11011001	R	D9
11011010		DA
11011011		DB
11011100		DC
11011101		DD
11011110		DE
11011111		DF
11100000		E0
11100001		E1
11100010	S	E2
11100011	T	E3
11100100	U	E4
11100101	V	E5
11100110	W	E6
11100111	X	E7
11101000	Y	E8
11101001	Z	E9
11101010		EA
11101011		EB
11101100		EC
11101101		ED
11101110		EE
11101111		EF
11110000	0	F0
11110001	1	F1
11110010	2	F2
11110011	3	F3
11110100	4	F4
11110101	5	F5
11110110	6	F6
11110111	7	F7
11111000	8	F8
11111001	9	F9
11111010		FA
11111011		FB
11111100		FC
11111101		FD
11111110		FE
11111111		FF

E4, which if translated would represent the number 2, is the letter U in the code used by the System/3.

C2, which if translated would represent the number 1, is the letter B in the code used by the System/3.

ART: 51757

Figure 10-23. Differences in Character Codes

Transletion Table and Alternate Collating Sequence Coding Sheet

You will find this coding sheet helpful for determing the correct entries you wish to make in the file translation table input card. Figure 10-24 shows the entries made on the sheet for the previous example.

INDICATORS

Indicators are used to signal when certain conditions occur or do not occur. After you have assigned an indicator (on one of the specification sheets) to signal a certain condition, the indicator assigned is associated with that one condition throughout the entire program.

Many times you want operations to be performed only when certain conditions occur. Because the indicator associated with the condition tells whether or not the condition has occurred, you may use the indicator to signal whether or not the operation should be done. In this way, indicators condition operation.

The status (on or off) of an indicator assigned on a specification line is determined by the results of processing the instruction on that specification line. If the condition has been satisfied, the indicator turns on; if it has not, the indicator turns off.

International Business Machines Corporation — Form X21-9090 Printed in U.S.A.

TRANSLATION TABLE AND ALTERNATE COLLATING SEWENCE CODING SHEET

Code	System/3 Graphic	Entry	Replaced By
00110011		33	
00110100		34	
00110101		35	
00110110		36	
00110111		37	
00111000		38	
00111001		39	
00111010		3A	
00111011		3B	
00111100		3C	
00111101		3D	
00111110		3E	
00111111		3F	
01000000	Blank	40	
01000001		41	
01000010		42	
01000011		43	
01000100		44	
01000101		45	
01000110		46	
01000111		47	
01001000		48	
01001001		49	
01001010	¢	4A	
01001011	.	4B	
01001100	<	4C	
01001101	(	4D	
01001110	+	4E	
01001111	|	4F	
01010000	&	50	
01010001		51	
01010010		52	
01010011		53	
01010100		54	
01010101		55	
01010110		56	
01010111		57	
01011000		58	
01011001		59	
01011010	!	5A	
01011011	$	5B	
01011100	*	5C	
01011101	)	5D	
01011110	;	5E	
01011111	¬	5F	
01100000	-	60	
01100001	/	61	
01100010		62	
01100011		63	
01100100		64	
01100101		65	

Code	System/3 Graphic	Entry	Replaced By
01100110		66	
01100111		67	
01101000		68	
01101001		69	
01101010		6A	
01101011	,	6B	
01101100	%	6C	
01101101	_	6D	
01101110	>	6E	
01101111	?	6F	
01110000		70	
01110001		71	
01110010		72	
01110011		73	
01110100		74	
01110101		75	
01110110		76	
01110111		77	
01111000		78	
01111001		79	
01111010	:	7A	
01111011	#	7B	
01111100	@	7C	
01111101	'	7D	
01111110	=	7E	
01111111	"	7F	
10000000		80	
10000001		81	
10000010		82	
10000011		83	
10000100		84	
10000101		85	
10000110		86	
10000111		87	
10001000		88	
10001001		89	
10001010		8A	
10001011		8B	
10001100		8C	
10001101		8D	
10001110		8E	
10001111		8F	
10010000		90	
10010001		91	
10010010		92	
10010011		93	
10010100		94	
10010101		95	
10010110		96	
10010111		97	
10011000		98	

Code	System/3 Graphic	Entry	Replaced By
10011001		99	
10011010		9A	
10011011		9B	
10011100		9C	
10011101		9D	
10011110		9E	
10011111		9F	
10100000		A0	
10100001		A1	
10100010		A2	
10100011		A3	
10100100		A4	
10100101		A5	
10100110		A6	
10100111		A7	
10101000		A8	
10101001		A9	
10101010		AA	
10101011		AB	
10101100		AC	
10101101		AD	
10101110		AE	
10101111		AF	
10110000		B0	
10110001		B1	
10110010		B2	
10110011		B3	
10110100		B4	
10110101		B5	
10110110		B6	
10110111		B7	
10111000		B8	
10111001		B9	
10111010		BA	
10111011		BB	
10111100		BC	
10111101		BD	
10111110		BE	
10111111		BF	
11000000		C0	
11000001	A	C1	F9
11000010	B	C2	F1
11000011	C	C3	F3
11000100	D	C4	
11000101	E	C5	
11000110	F	C6	
11000111	G	C7	F7
11001000	H	C8	F8
11001001	I	C9	F5
11001010		CA	
11001011		CB	

Code	System/3 Graphic	Entry	Replaced By
11001100		CC	
11001101		CD	
11001110		CE	
11001111		CF	
11010000	}	D0	
11010001	J	D1	
11010010	K	D2	F4
11010011	L	D3	
11010100	M	D4	F0
11010101	N	D5	F6
11010110	O	D6	
11010111	P	D7	
11011000	Q	D8	
11011001	R	D9	
11011010		DA	
11011011		DB	
11011100		DC	
11011101		DD	
11011110		DE	
11011111		DF	
11100000		E0	
11100001		E1	
11100010	S	E2	
11100011	T	E3	
11100100	U	E4	F2
11100101	V	E5	
11100110	W	E6	
11100111	X	E7	
11101000	Y	E8	
11101001	Z	E9	
11101010		EA	
11101011		EB	
11101100		EC	
11101101		ED	
11101110		EE	
11101111		EF	
11110000	0	F0	
11110001	1	F1	
11110010	2	F2	
11110011	3	F3	
11110100	4	F4	
11110101	5	F5	
11110110	6	F6	
11110111	7	F7	
11111000	8	F8	
11111001	9	F9	
11111010		FA	
11111011		FB	
11111100		FC	
11111101		FD	
11111110		FE	
11111111		FF	

This is the hexadecimal equivalent of the character to be translated.

This is the hexadecimal equivalent of the System/3 character that will be substituted for the character that is to be translated.

ART: 51758

Figure 10-24. Specifications for File Translation Input **Cards**

Usually indicators are set on or off by the conditions in the program itself. However, you may also set certain indicators by the SETON and SETOF operation. At the start of each program all indicators are off except the 1P indicator, L0 indicator, and any external indicators which have been set on. All indicators which you may use are shown in Figure 10-25.

01-99 (Field Indicators, Record Identifying Indicators, Resulting Indicators, and Conditioning Indicators)

You may assign any of the numbers 01-99 to indicate such things as:

1. The type of record read (see *Columns* 19-20 in Chapter 7).
2. The status (plus, minus, zero/blank) of an input field (see *Columns* 65-76) in Chapter 7).
3. The results of a calculation operation (see *Columns 54-59* in Chapter *8). See Examples, Example 1* and *Example 2.*

Any of these indicators which you have assigned may then also be used to:

1. Condition calculation operations (see *Columns 9-17* in Chapter 8).
2. Condition output operations (see *Columns 23-31* in Chapter 9).
3. Establish field record relations (see *Columns* 63-64 in Chapter 7).

Indicators	File Description Specifications	Input Specifications			Calculation Specifications			Output-Format Specifications
	File Conditioning (71-72)	*Record Identifying Indicator (19-20)	*Field Record Relation (63-64)	Field Indicator (65-70)	Control Level Indicator (7-8)	Conditioning Indicator (9-17)	Resulting Indicator (54-59)	Conditioning Indicator (23-31)
01-99		X	X	X		X	X	X
H1-H9		X	X	X		X	X	X
1P								X ***
MR			X **			X		X
OA-OG,OV						X	X	X ****
L0					X			X
L1-L9		X	X **		X	X	X	X
LR		X			X	X	X	X
U1-U8	X *****		X			X		X

Note: X denotes the indicators that may be used.

* Not valid on look-ahead fields.

** When field named is not a match field or a control field.

*** Only for detail or heading lines.

**** Cannot condition an exception line, but may condition fields within the exception record.

***** Not valid for table input files.

ART: 51762

Figure 10-25. Valid Indicators

Indicators reflect only one condition at a time. When one indicator is used to reflect two or more conditions, it is always set to reflect the condition in the last operation performed. Therefore, it is not usual practice to assign the same number as a field indicator and/or resulting indicator more than once in a program. When you use such an indicator to condition other operations, you may get wrong results since the indicator may not always reflect the condition you think it does (see *Examples, Example* 3).

If any indicator 01-99 is set on or off by the operation codes SETON or SETOF, it remains on or off until an instruction in a specification line containing that same indicator is performed. The indicator is then set to reflect a condition from the operation performed.

Examples

Example 1: Figure 10-26, insert **A,** shows that resulting indicator 10 has been assigned to signal when a minus condition occurs. Indicator 10 turns on if the result after the subtraction operation has been performed is negative. It then remains on (or off depending upon the result) until the same operation is performed again. It is always set to reflect the result of the subtraction operation each time it is done.

Example 2: Figure 10-26, insert B, shows the same operation as insert **A.** However, this operation is conditioned by indicator 01. The operation is done only when indicator 01 is on. Resulting indicator 10 is set on only when the result of the operation is negative.

Example 3: Figure 10-26, insert C, shows the use of the same indicator (10) in two lines. The status of this indicator reflects the result of each operation. For instance, indicator 10 turns on after the operation in line 05 has been done if the result of the opeation is negative. However if the result of the operation in line 07 is positive or zero, indicator 10 turns off. It is then reset only when the operation in line 05 is done again.

H1-H9 (Halt Indicators)

You may use any halt indicator to:

1. Cause the program to stop after finding an unacceptable condition.

2 Condition calculation or output operations that are not to be performed when such an unacceptable condition has occurred. This is necessary because all calculation and detail output operations are still performed for the record that caused the error before processing stops.

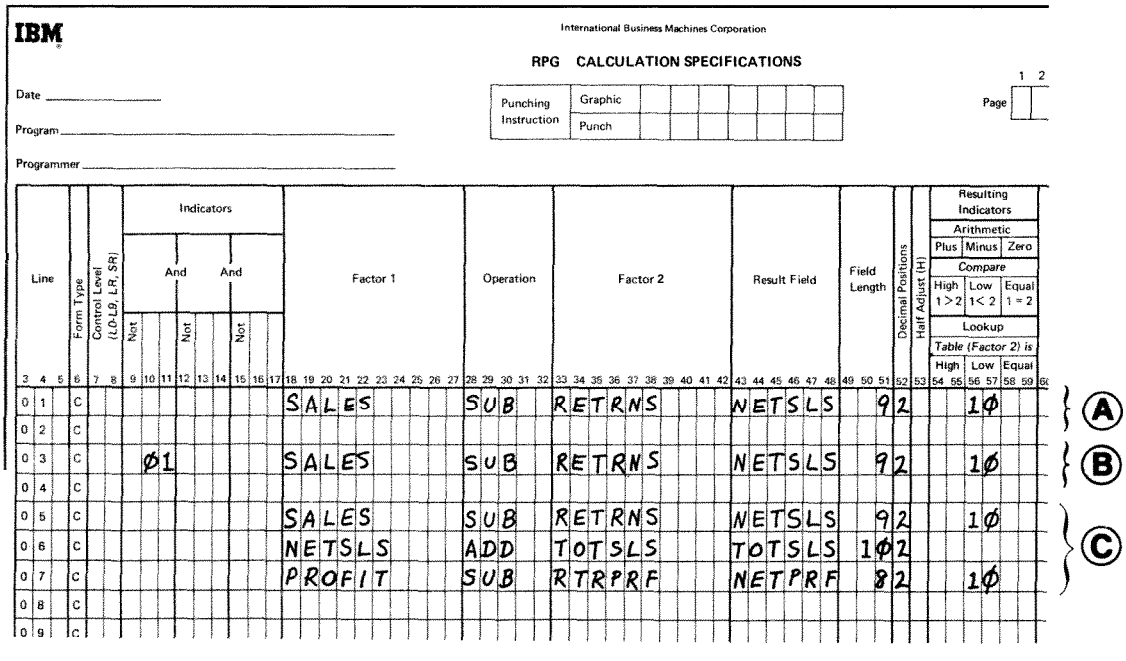
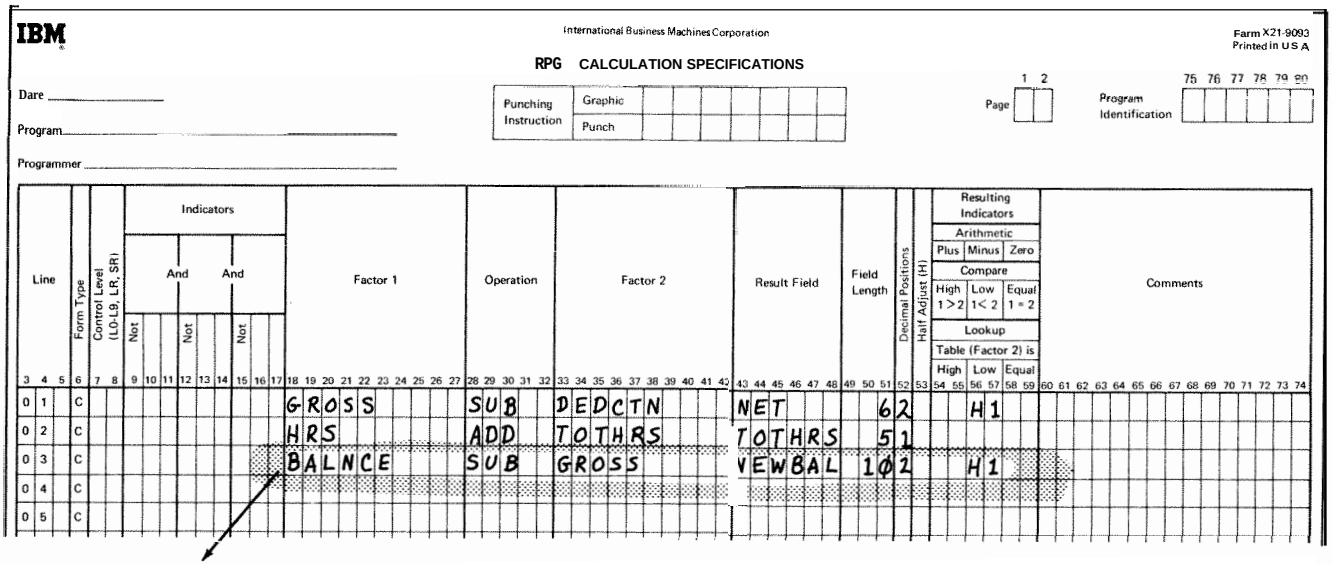


Figure 10-26. **Indicators 01-99**

3. Establish field record relations (see Columns 63-64 in Chapter **7).**

Using the same indicator to test for two or more error conditions is not usually good practice. For example, Figure **10-27,** insert **A,** shows the use of H1 in two different specification lines. If the result of the calculation operation in line **01** is negative, H1 turns on. This is an error condition. Processing continues, however, until this program cycle is completed. Thus, the operation in line **03** is done. If the result of this subtraction operation is positive, H1 turns off. The program does not stop because H1 is not on, even though an error condition has been found in line **01.**

The use of two different halt indicators as shown in Figure **10-27,** insert B, does not allow a situation like the one just described to occur.



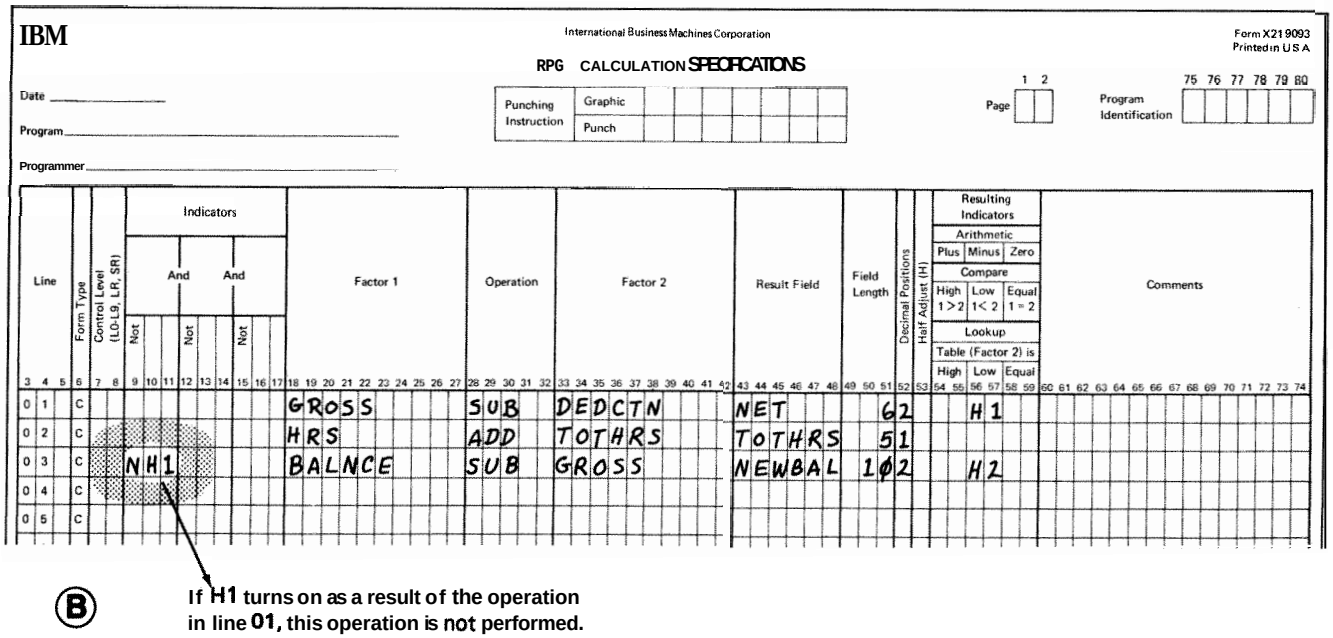


Figure 10-27. One Haft Indicator Testing for Two Error Conditions

Any halt indicator assigned to test for zero or blank is off at the beginning of the program.

Note: If a halt indicator stops processing, it is turned off when the system is restarted. If more than one halt indicator turns on during a program cycle, each halt indicator must be considered separately. Every time the program is restarted, only one halt indicator is bypassed.

1P (First Page Indicator)

Use the first page indicator to condition those lines which are to be printed on only the first page. These lines are usually heading lines. Data is provided for lines conditioned by the 1P indicator by constants entered in columns 45-70 of the Output-Format sheet.

All lines conditioned by the 1P indicator are printed out even before the first record from input file is processed. Therefore, do not condition output fields which are based upon data from input records by the 1P indicator. You get meaningless output if you do.

Calculation operations cannot be conditioned by the 1P indicator either. This indicator is on at the beginning of the program and turns off after the detail output has been performed on the first data record.

MR (Matching Record Indicator)

Use the MR indicator to condition calculation and output operations which are to be done only when records match.

The MR indicator turns on when a primary file record matches any secondary file record on the basis of the matching fields indicated by M1-M9. The matching record indicator is always set on or off (according to a match or nonmatch field) before any calculation operations which are not conditioned by control level indicators (columns 7-8 of the calculation specifications) are performed. It retains this setting for one complete cycle. If all primary file records match all secondary file records, the MR indicator is always on.

If record types for which no matching fields have been specified are read, they are processed as if they belonged to the same match group as the record previously processed. MR is always off for these types.

OA-OG,OV (Overflow Indicators)

Overflow indicators are used only on the printer file. Use them primarily to condition the printing of heading lines. If you intend to use an overflow indicator to condition output lines on the printer, you must assign an overflow indicator to the printer file on the File Description sheet (columns 33-34). This same indicator must then be used to condition all lines that are to be written only when overflow occurs.

If the destination of a space/skip or print operation falls within the form overflow area, the overflow indicator is turned on and remains on until all overflow lines are printed. However, if a skip is specified that advances the form past the overflow line to the first line or past the first line on a new page, the overflow indicator does not turn on. Certainly, you do not want the overflow indicator on to signal a need for a new page when you just skipped to a new page.

If an overflow indicator is used as a conditioning indicator, it indicates that output is to be performed at overflow time. This applies regardless of whether or not the line conditioned by the indicator is in an AND or OR relationship with other indicators.

When an overflow indicator is used, a form skip specification should be made on the last line conditioned by an overflow indicator. Otherwise, forms do not advance. Remember, they advance automatically if you do not use overflow indicators.

The overflow indicator may be set by the SETON or SETOF operation code, After all total records have been written, however, the indicator is set as it normally is in accord with the overflow line. See Overflow *Indicators* in this chapter for further information.

L1-L9 (Control Level Indicators)

Control level indicators are used to signal when a change in a control field has occurred. Because they turn on when the information in a control field changes, they may be used to condition operations (such as finding totals) that are to be performed only when all records having the same information in the control field have been read. They may also be used to do total printing or to condition operations that are to be done on only the first record in a control group. Control level indicators always turn on after the first record of a control group is read.

Control level indicators may be used in three different types of specifications: input, calculation, and output-format.

Input Specifications

If a control level indicator is entered in columns 59-60 of this sheet, the field described in columns 53-58 is declared to be a control field. This means that the field on each card read is matched against the same field on the previous card. If the information is not the same, the control level indicator turns on. All lower level indicators turn on when a higher level indicator turns on. For example, if L8 turns on, L1-L7 also turn on.

When a control level indicator is used on the Input sheet in the Field Record Relation columns (63-64), the data from the field named in columns 53-58 is accepted and used only when the control level indicator is on.

If record types without a control field are read, they are treated as if they belong to the same control group as the preceding record. No control level indicator is set for them. Control level indicators may also be used to establish field record relations (see Columns 63-64 in Chapter 7).

Calculation Specifications

When a control level indicator is entered in columns 7-8 of this sheet, it conditions the operation so that it is done only when a control field changes. If any control level indicator appears in columns 9-17, the operation is done only on the first record of a new control group.

A control level indicator may be turned on or off by operation codes SETON and SETOF. However, these operations do not cause all control level indicators lower than the one specified to turn on or off. For example, when L2 is set on, L1 does not automatically turn on.

Output-Format Specifications

Control level indicators entered in columns 23-31 of this sheet specify when output records are to be written:

1. If the control level indicator is entered along with a T in column 15 and no overflow indicator is used, the record is written only after the last record of a control group has been processed,

2. If the indicator is entered along with a *D* in column 15 and no overflow indicator is used, the record is written only after the first record of the new control group has been processed.

3. If the control level indicator is entered along with an overflow indicator, the record is written after the overflow line has been sensed (provided a control break has also occurred).

LO Indicator

The LO indicator is never assigned, but it is always automatically on. Thus, it can be used to condition certain calculation or output operations. LO is used in the same way and for the same purpose as the other control level indicators. However, it is used only when L1-L9 cannot be assigned because the input data records have no field available which can serve as a control field.

LR (Last Record Indicator)

Use the LR indicator to condition all operations that are to be done only at the end of the job. This indicator automatically turns on after the last record of the input file has been processed. When LR turns on, all other control level indicators used also automatically turn on. If LR is on, the job ends after all total records have been written. It is also possible to turn the LR indicator on by a SETON operation. This does not, however, cause all other control level indicators used to turn on. (LR cannot, however, be turned off by a SETOF operation.)

U1-U8 (External Indicators)

Indicators U1-U8 are external indicators. This means they are set prior to processing by an operation control statement. Their setting cannot be changed during processing. Thus, the program has no control over them.

You may use these indicators as file conditioning indicators. They tell whether or not a certain file is to be used for a job. For example, you may have a job which one time requires the use of two output (or input) files and another time the use of only one. Instead of writing two different programs (one using one file, the other two), you can condition a file (in the file description specifications) by an external indicator. When the indicator is on, the file is used; when it is off, the file is not used.

If a file is conditioned by an external indicator, all output data handled by the file must also be conditioned by the same indicator. Any calculation operations which should not be done when the file is not in use should also be conditioned by the same indicator.

In addition to using these indicators as file conditioning indicators, you may use them:

1. To condition calculation operations.

2. To condition output operations.

3. As field record relation indicators (columns 63-64 of input specifications).

LOOK AHEAD

RPG programs process one record at a time. Normally, only the information from the record being processed is available for use. However, the RPG II look ahead feature enables you to use information from records that follow the one being processed. The fields containing the information are called look ahead fields.

Look Ahead Fields

The look ahead feature can be used only with input, update, or combined files. Chained files or demand files may not use the look ahead feature. To use the look ahead feature, you must describe the look ahead fields and reference them as you do normal fields. You can describe one set of look ahead fields per file; the description applies to all records in the file, regardless of their type. (The specifications for describing the fields are given later.)

Look ahead fields always apply to the next record in the file, provided the file is not a combined or update file. Thus, if you wish to use information both before and after the record is selected for processing, you must describe the field twice, once as a look ahead field and once as a normal field.

For combined and update files the look ahead fields apply to the next record in the file only if the current record was not read from that file. Therefore, when you are reading from only one file and the file is a combined or update file, look ahead fields always apply to the current record.

Figure 10.28 shows the processing of three records from two input files, file A and file B. The first record from each file is read (see Figure 10.28, insert A). In Figure 10-28, insert B, record A1 is selected for processing; in Figure 10.28, insert C, record A2; and in Figure 10.28, insert D, record B1. The records available for look ahead during the processing of these records are:

Record Processed	*Records Available*
A1	A2 and B1
A2	A3 and B1
B1	A3 and B2

In general, when the record being processed is from an input file, the next record in the input file is available as are the records which were read, but not selected, from the other files.

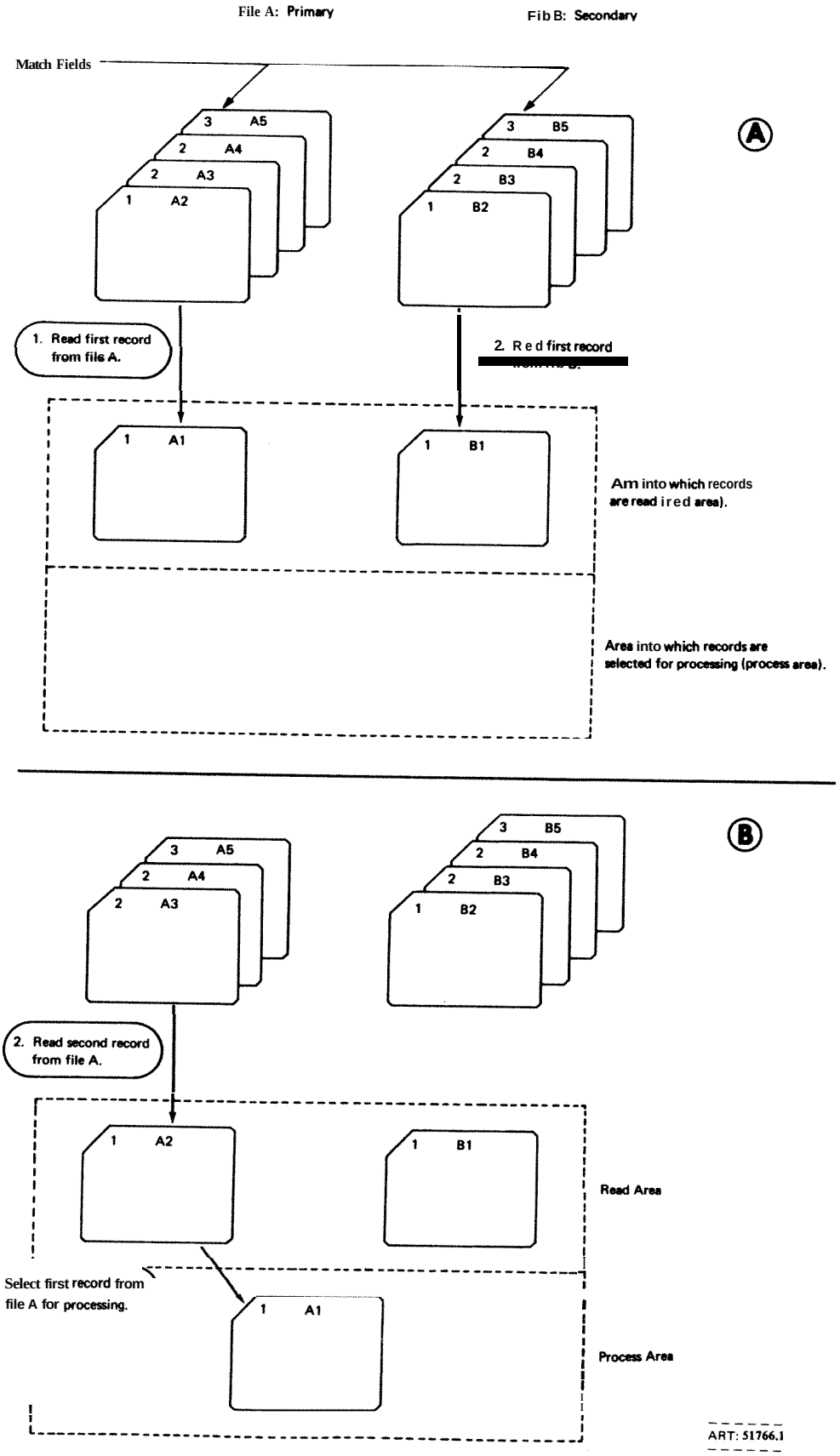


Figure 10-28. Available Records: Two Input Files (Part 1 of 2)

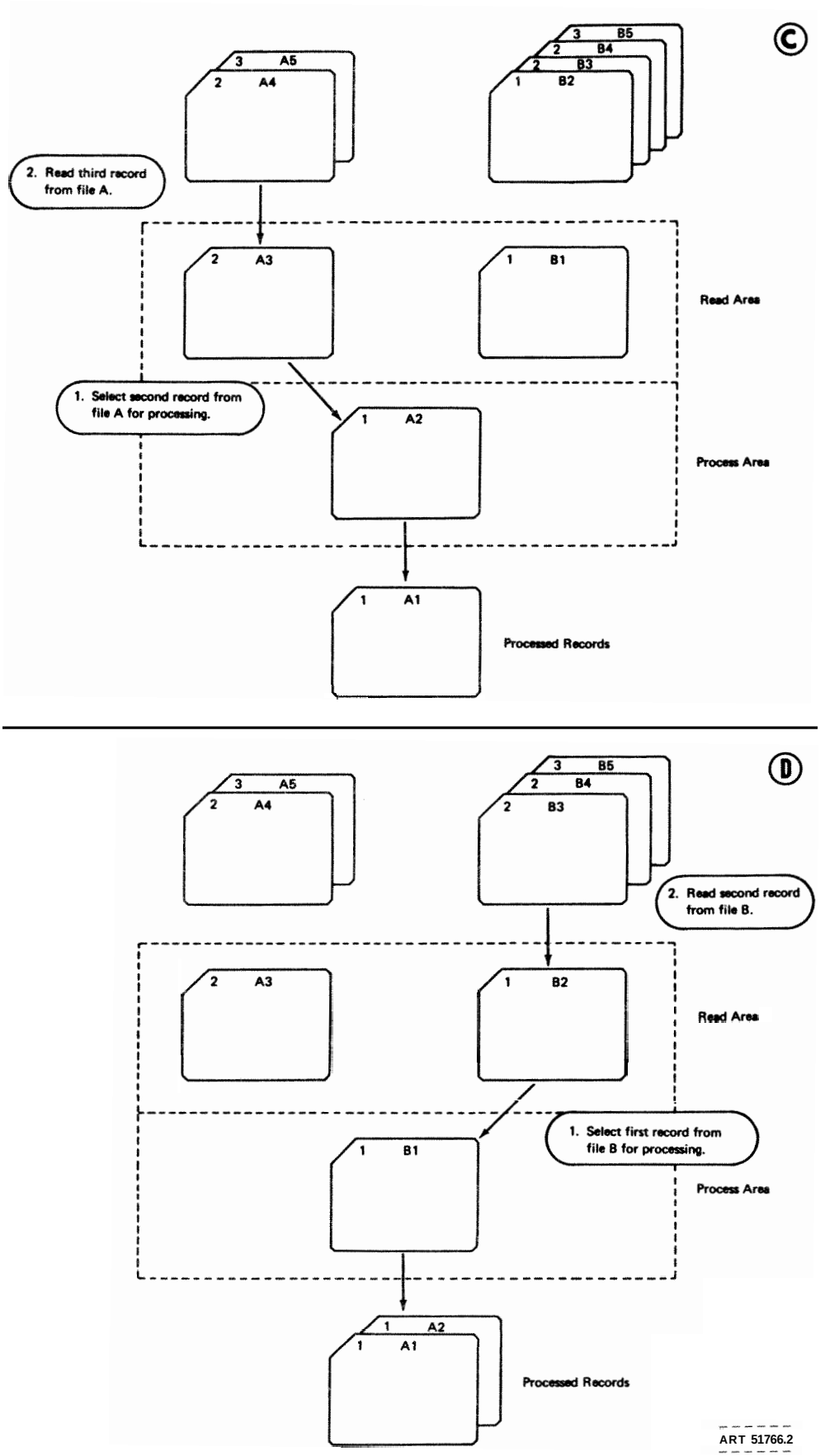


Figure 10-28. Available Records: Two Input Files (Part 2 of 2)

Figure 10.29 shows the same files as Figure 10.28 with one exception: file A is a combined file. The records available for look ahead during the processing of the three records are:

Records *Processed*	*Records* ***Available***
A1	A1 and B1
A2	A2 and B1
B1	A3 and B2

In general, when the record being processed is from a combined or update file, only the records which were read, but not selected, from the other files are available for look ahead. The next record from the combined or update file is not read until after the current record has been processed. Therefore, the next record from the combined or update file is not available for look ahead.

After the last record from a file has been processed, every look ahead field for the file is automatically filled with 9's. For example, a field three record positions long contains 999. The 9's remain in the fields until the job ends. Note also that blank after (*B* in column 39 of the Output-Format sheet) cannot be used with look ahead fields.

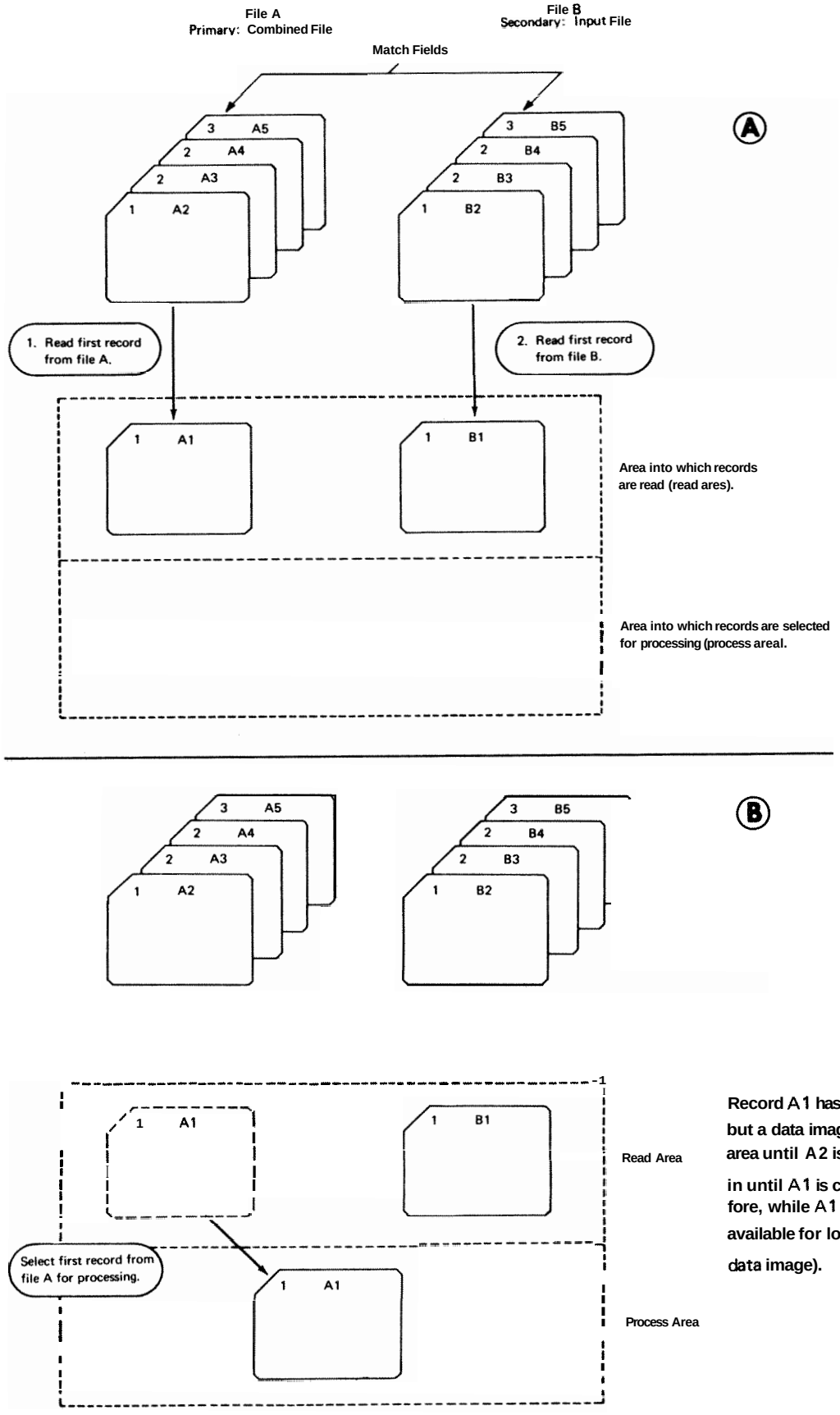


Figure 10-29. Available Records: One Input File, One Combined File (Part 1 of 3)

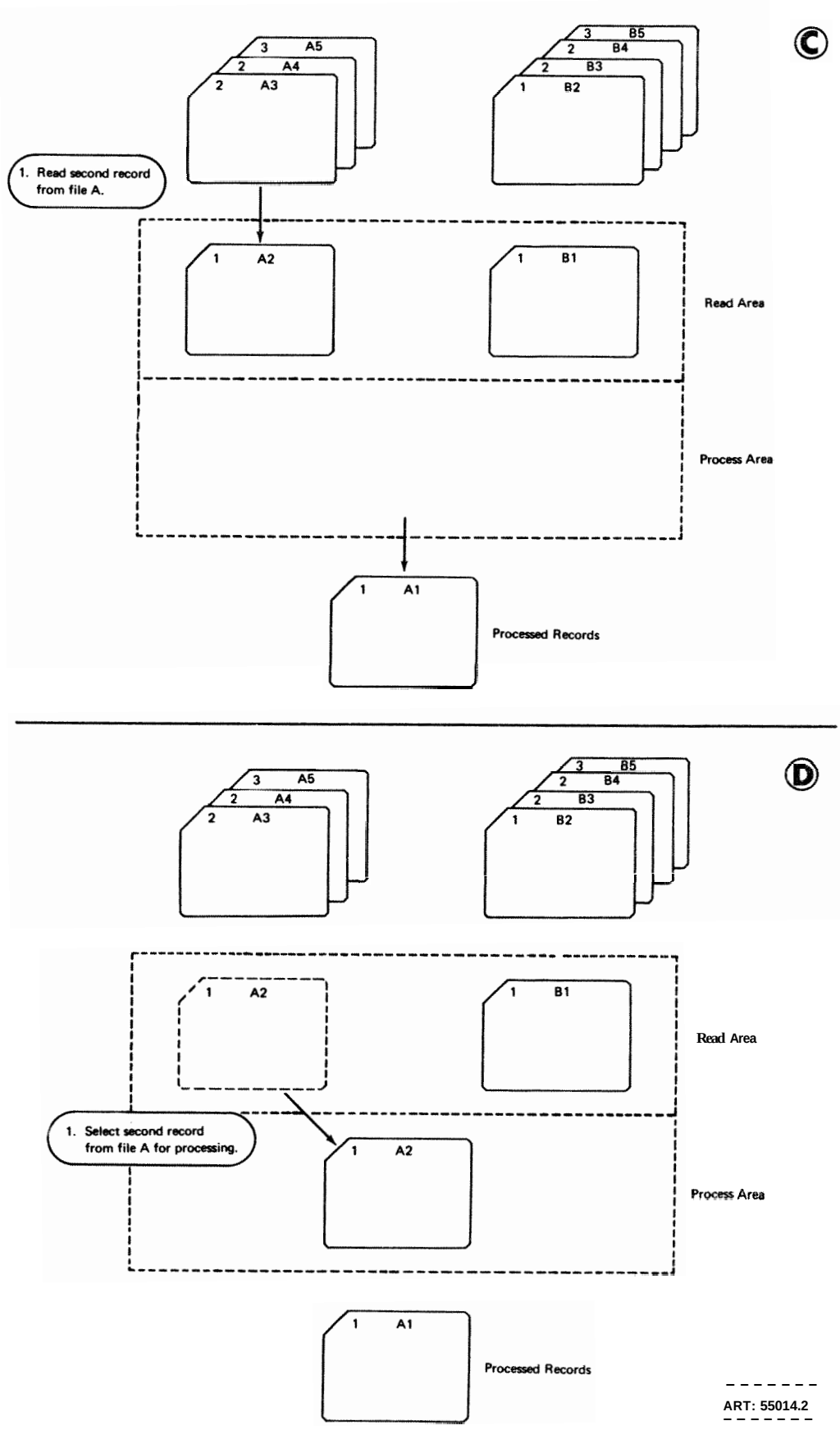


Figure 10-29. Available Records: One Input File, One Combined File (Part 2 of 3)

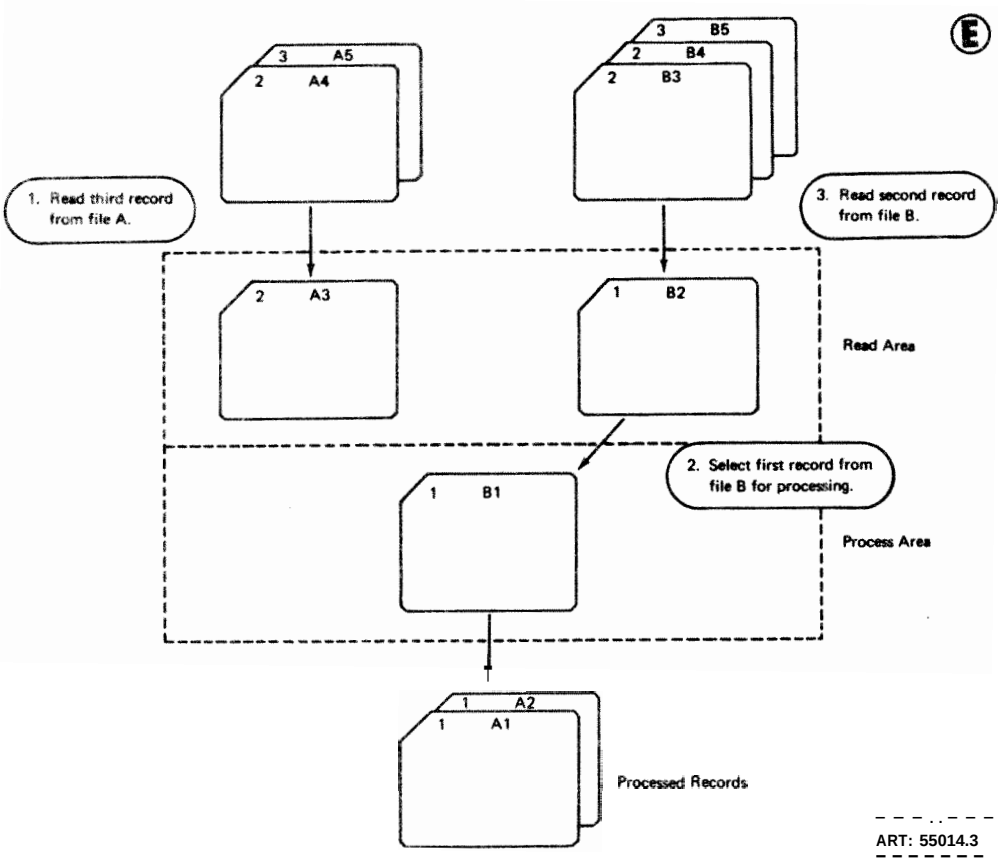


Figure **10-29.** Available **Records: One Input File, One Combined File (Part** *3* **of** *3)*

Specifications

You can describe one set of look ahead fields per file. The description applies to all records in the file, regardless of their type. Look ahead fields must not be described for demand files, and they must not be used as array fields. Describe the fields on the Input sheet before or after the field descriptrons for any of the records in the file. To describe a set of look ahead fields, place ** in columns 19-20 of a line. Leave columns 1-14 and 21-74 blank. Place any alphabetic characters under Sequence in columns 15-16. Describe the look ahead fields on separate lines following the ** line (as in Figure 10-30 Bart 2, insert B). Describe each field as follows:

1. Leave columns 1-43 blank.

2. In columns 44-51, identify the record positions in which the field is located.

3. If the field is numeric, indicate in column **52** the number of digits to the right of the decimal point. If the field is not numeric, leave column **52** blank.

4. In columns **53-58** identify the field by name. If the field is also one of your normal fields, be sure to use a different name here. Use this name to refer to the look ahead field.

5. Leave columns **59-74** blank.

Examples

Figure 10-30 shows a job which reads records from two files. The primary file is named PRIMARY; the secondary file, SECONDRY. If a record from the primary file matches one from the secondary file, the information in positions 1-10 of the secondary file record is placed in positions 31-40 of the primary file record. When there is no match, a 6 is placed in position 1 of the primary file record. The 6 will indicate an unmatched record in the primary file.

Because the primary file record is processed first when it matches a secondary file record, the information from the secondary file record has to be described as a look ahead field.

MULTIFILE PROCESSING

Multifile processing applies to programs that read records from a primary file and one or more secondary files. It is the name given to the methods by which the programs select records for processing. The method used depends upon whether or not match fields are used in the records.

No Match Fields

When no match fields are used, records are selected from one file at a time. When the records from one file have all been processed, the records from the next file are selected. The files are chosen in this order:

1. primary file
2. secondary files in the order in which they are described in the file description specifications.

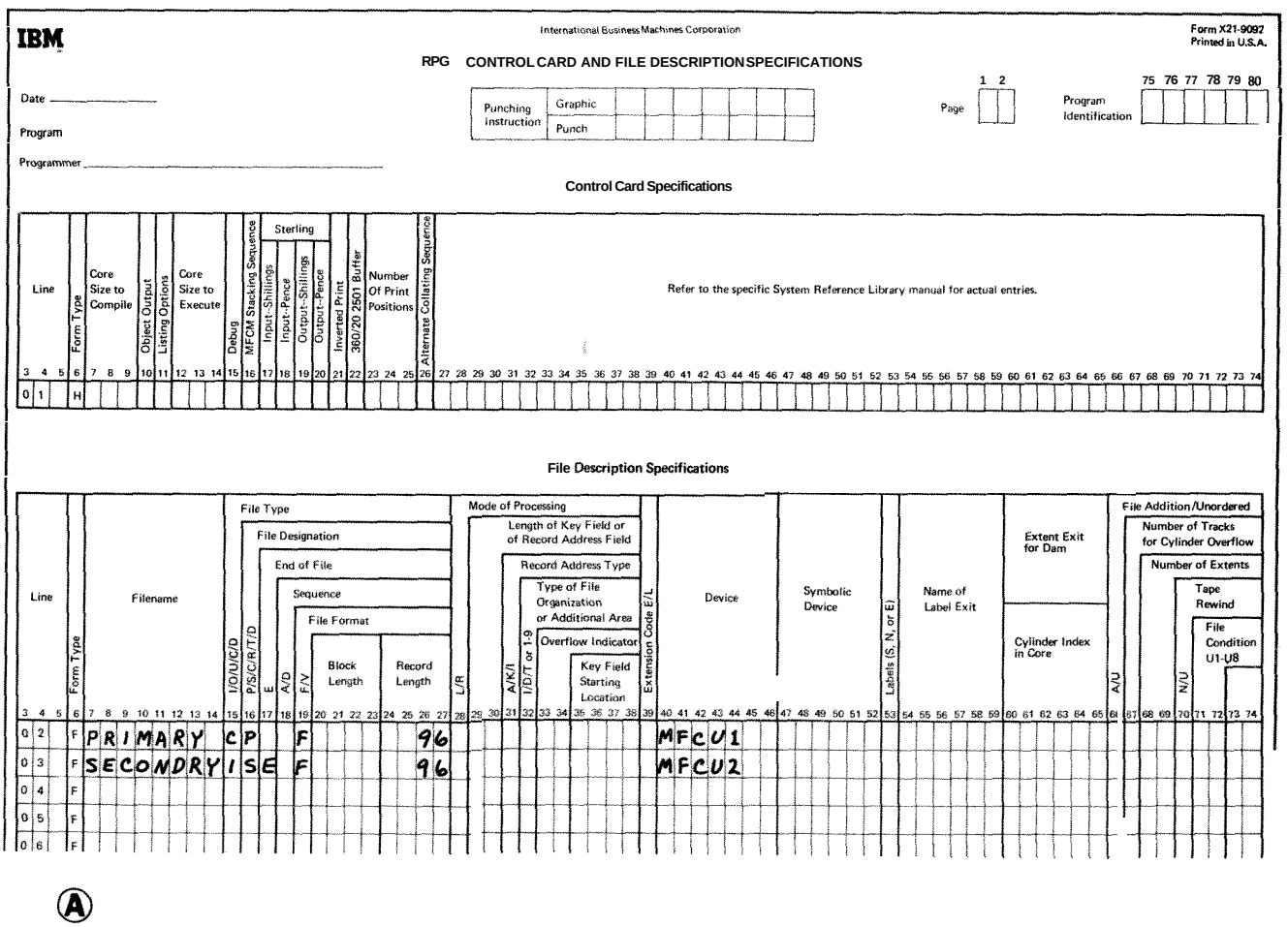
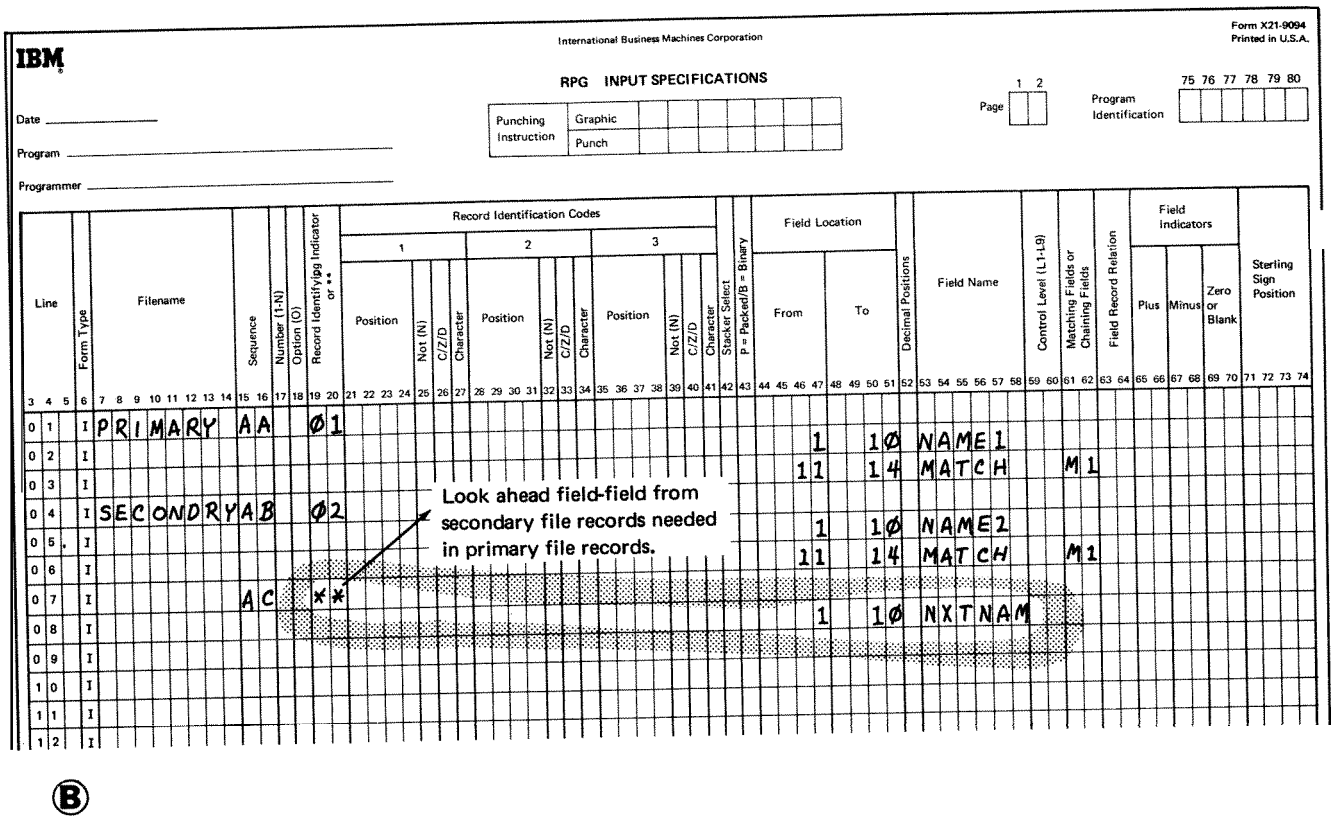
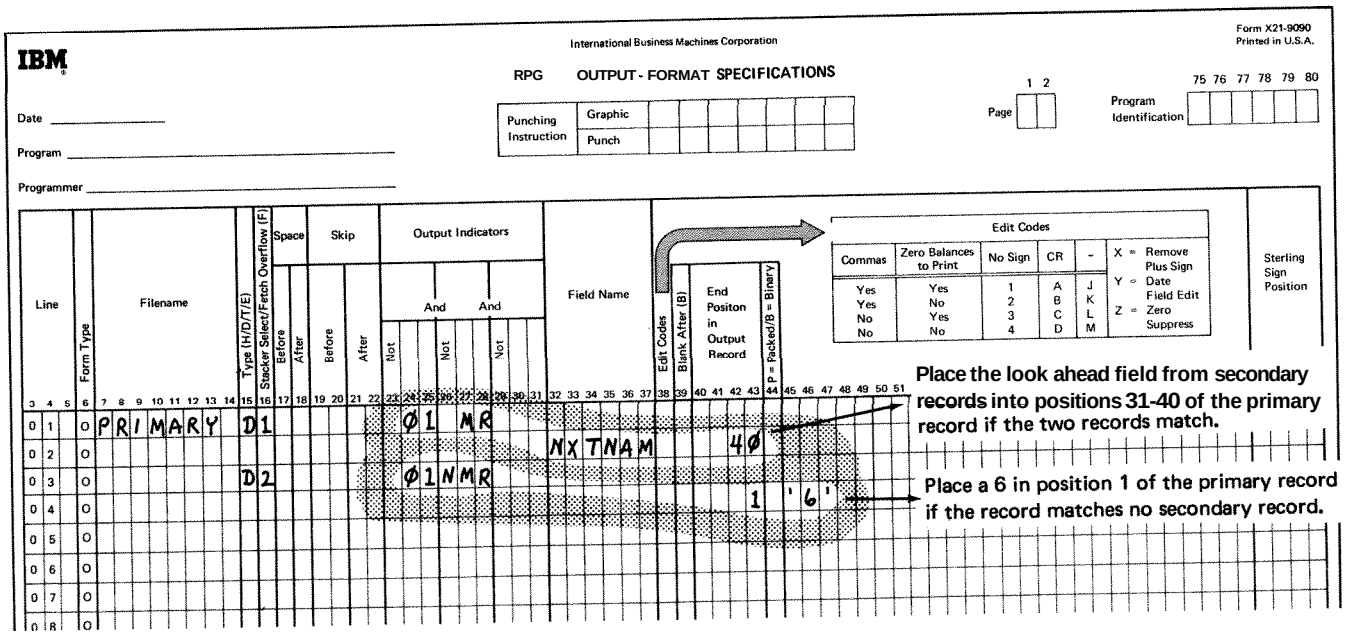


Figure 10-30. Look Ahead Fields (Part 1 of 2)



Ⓑ



Ⓒ

Figure 10-30. Look Ahead Fields (Part *2* of 2)

Match Fields

When match fields are used, records are selected according to the contents of the match fields. One record is read from every file, and the match fields in the records are compared. If the records are in ascending order, the record with the lowest match field is selected for processing. If the records are in descending order, the record with the highest match field is selected.

When a record is selected from a file, the next record from the file is read. At the beginning of the next program cycle, the new record is compared with the records that had not been selected during the previous cycle and one is selected (Figure 10-31).

Records without match fields can be included in the files. Such records are selected before records with match fields. If two or more of the records being compared have no match fields, selection of those records is determined by the priority of the files from which the records came. The priority of files is:

1. primary file
2. secondary files in the order in which they are described in the file description specifications.

When the primary record matches one or more of the secondary records, the MR (matching record) indicator is turned on. The indicator can be used to condition calculations or output for the record that is selected. If one of the matching records must be selected, the selection is determined by the priority of the files from which the records came.

OPERATION CODES

You are able to perform many different types of operations on your data using the RPG II language. Special codes have been set up which indicate the operation to be performed. Usually these are just abbreviations of the name of the operation. You must use these codes to specify the operation to be performed.

Operations may be divided into nine categories; all codes in each category are explained in this section. Examples are also given for many codes. Figure 8-10 (Chapter 8) provides a summary of the operation codes. It also shows what other specifications need to be used with each code.

Arithmetic Operations

Arithmetic operations can be performed only on numeric fields or literals. The result field must also be numeric. For arithmetic operations in which all three fields are used:

1. Factor 1, factor 2, and the result field may all be different fields.
2. Factor 1, factor 2, and the result field may all be the same field.
3. Factor 1 and factor 2 may be the same field but different from the result field.
4. Either factor 1 or factor 2 may be the same as the result field.

The length of any field involved in an'arithmetic operation cannot exceed 15 characters. If the result exceeds 15 characters, characters may be dropped from either or both ends depending on the location of the decimal point. The results of all operations are signed (+,-). Any data placed in the result field replaces the data that was there previously.

Add (ADD)

Factor 2 is added to factor 1. The sum is placed in the result field. Factor 1 and factor 2 are not changed by the operation.

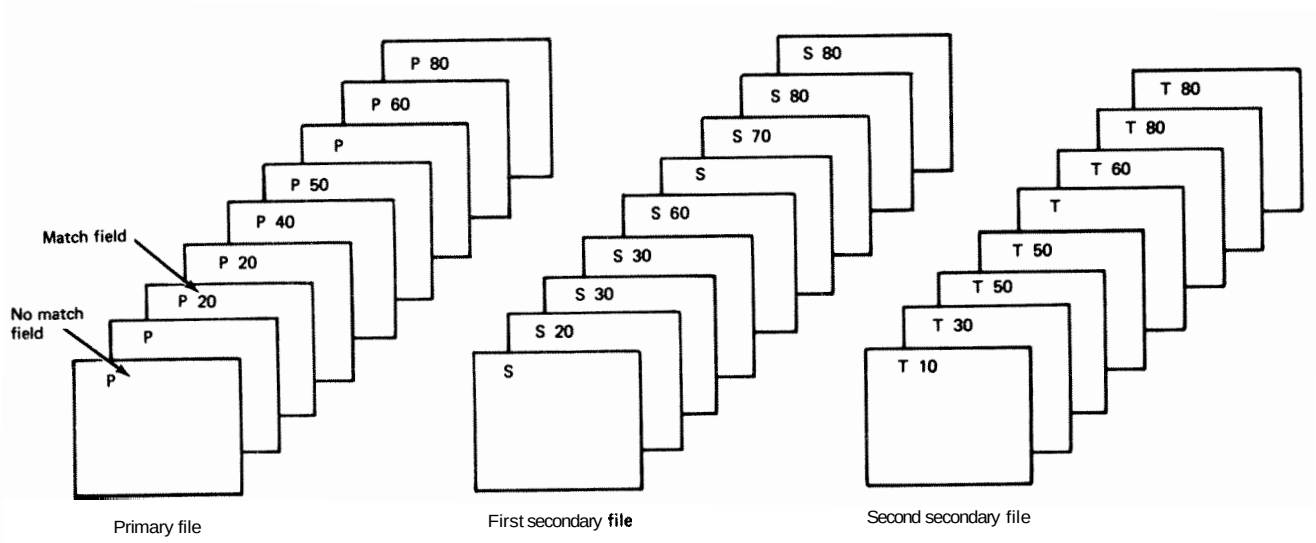
Zero and Add (Z-ADD)

The result field is set to zeros. Then factor 2 is added to the result field and placed in the result field. Factor 1 is not used.

Subtract (SUB)

Factor 2 is subtracted from factor 1. The difference is placed in the result field. Factor 1 and factor 2 are not changed by the operation,

Note: ***Subtracting*** two fields which are the same is a method of setting the result field to zero.



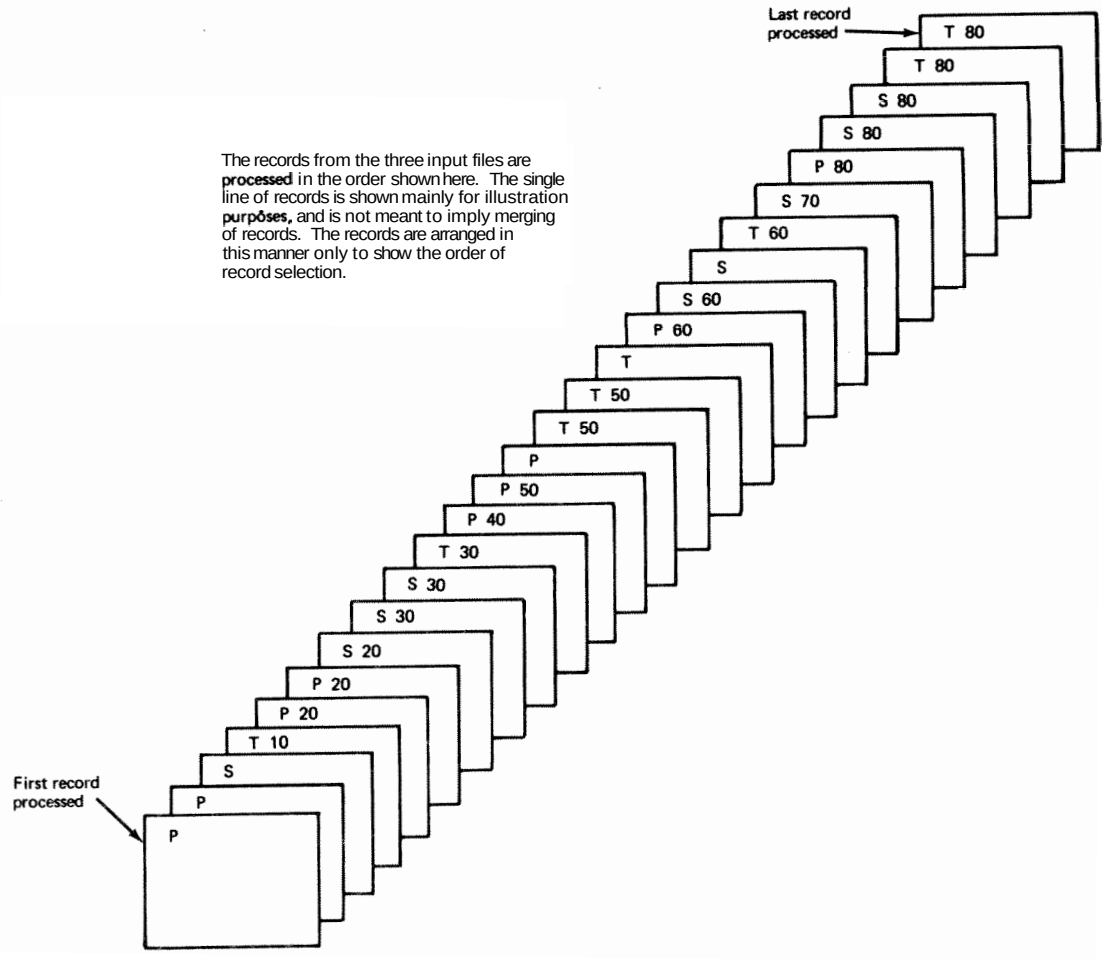


Figure 10-31. Normal Record Selection from Three Files (Part 1 of 3)

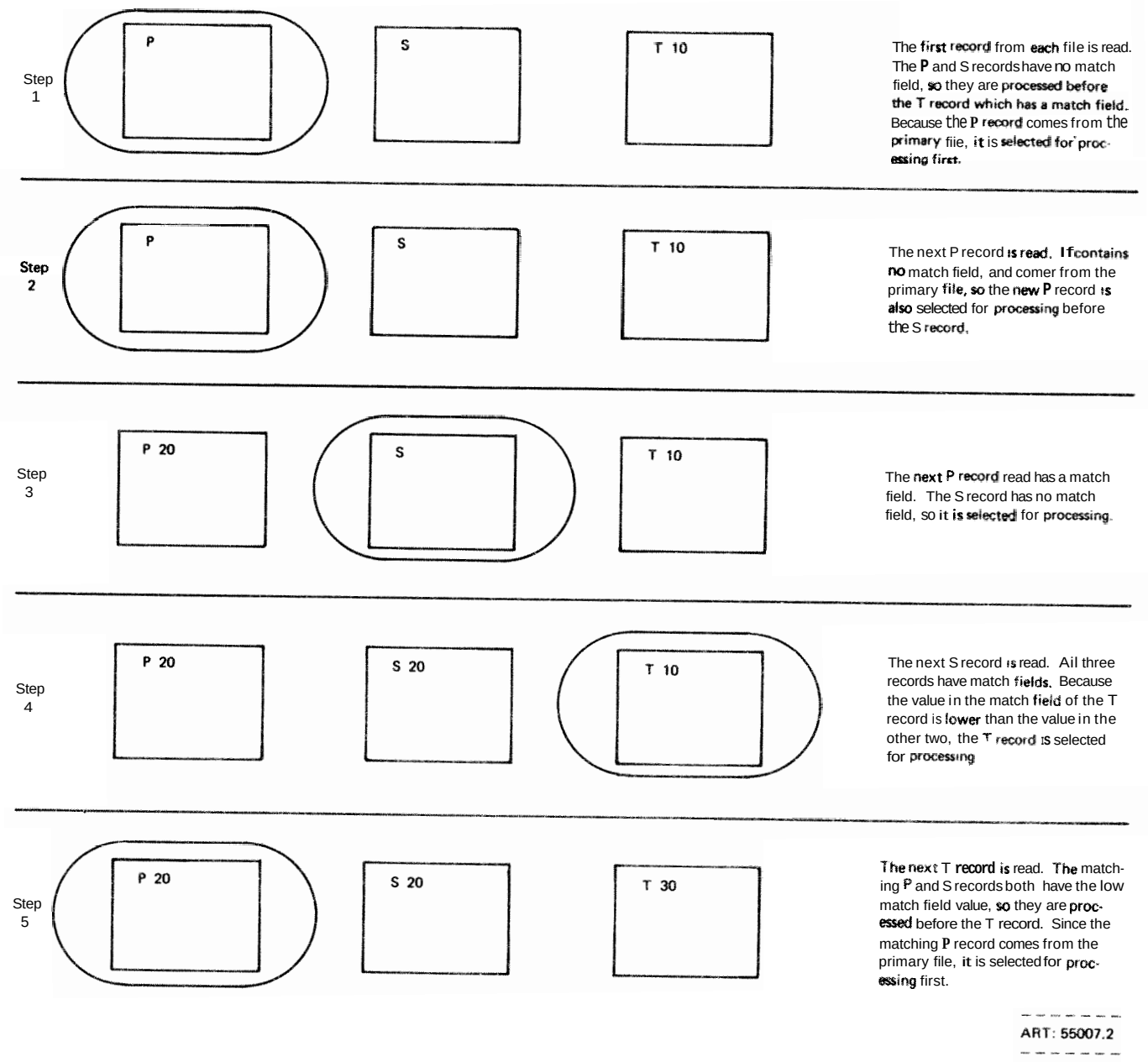


Figure 10-31. Normal Record Selection from Three Files (Part 2 of 3)

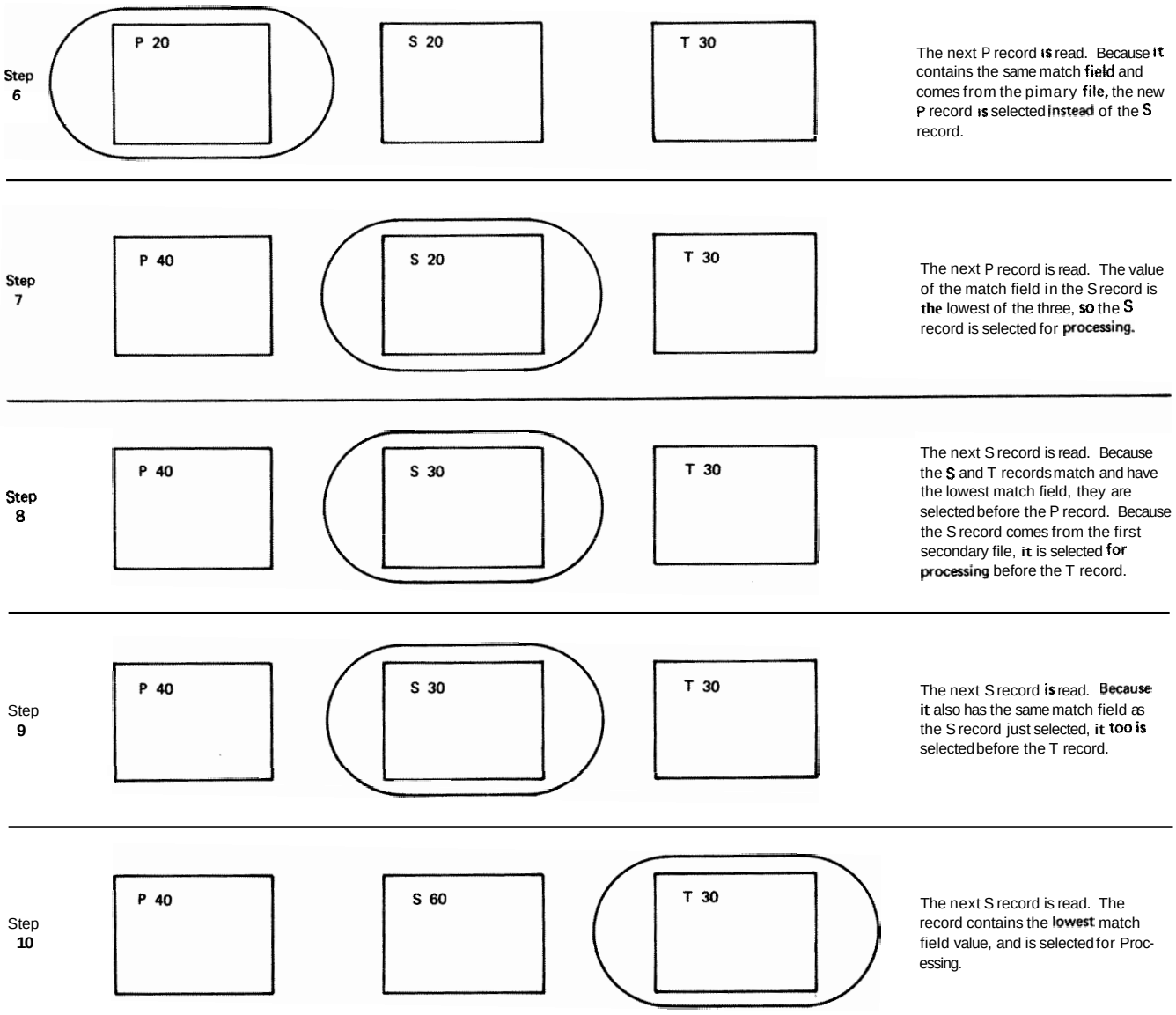


Figure 10-31. Normal Record Selection from Three Files (Part 3 of 3)

Zero and Subtract (Z-SUB)

The result field is set to zeros. Factor 2 is subtracted from the result field and then placed in the result field. This actually places the negative of factor 2 in the result field Factor 1 is not used This operation can be used to change the sign of a field

Multiply (MULT)

Factor 1 is multiplied by factor 2 The product is then placed in the result field Factor 1 and factor 2 are not changed When you use (as a factor) a field which is described as a result field, you must be sure the result field is large enough to hold the product.

Divide (DIV)

Factor 1 (dividend) is divided by factor 2 (divisor). The quotient (result) is placed in the result field. Factor 1 and factor 2 are not changed.

If factor 1 is 0, the result of the divide operation will be 0. Factor 2 cannot be 0. If it is, the job stops immediately. You may continue processing, however, by pressing the start key on the Processing Unit. When processing is continued, the result and remainder are set to zero.

Any remainder resulting from the divide operation is lost unless the move remainder operation is specified as the next operation, If move remainder is the next operation, the result of the divide operation cannot be half adjusted (rounded).

Move Remainder (MVR)

This operation moves the remainder from the previous divide operation to a separate field named under Result Field. Factor 1 and factor 2 must not be used. This operation must immediately follow the divide operation.

The length of the remainder is the same as the length of factor 2 in the divide operation. The remainder has the same number of decimal positions as the adjusted dividend had (the decimal positions are aligned in the remainder). If the result field is shorter than factor 2, the leftmost characters of the remainder are dropped. Figure 10-32 shows the use of the move remainder operation.

Square Root (SQRT)

This operation derives the square root of the field named in factor 2. The square root of factor 2 is placed in the result field. Do not use factor 1.

A whole array can be used in a SQRT operation if factor 2 and Result Field contain array names.

The number of decimal places in the result field may be less than or greater than the number of decimal places in factor 2. However, the result field should not have less than half the number of decimal places in Factor 2.

If the value of the Factor 2 field is negative, the job will halt. You may continue processing by pressing the start key on the Processing Unit. When processing is continued, the Result Field is set to zero.

Crossfoot (XFOOT)

This operation is used only on arrays. It adds all the elements of the array together and puts the sum into a separate field specified as the result field. Factor 1 is not used. Factor 2 contains the name of the array.

Move Operations

Move operations move part or all of factor 2 to the result field. Factor 2 remains-unchanged, but the result field is changed.

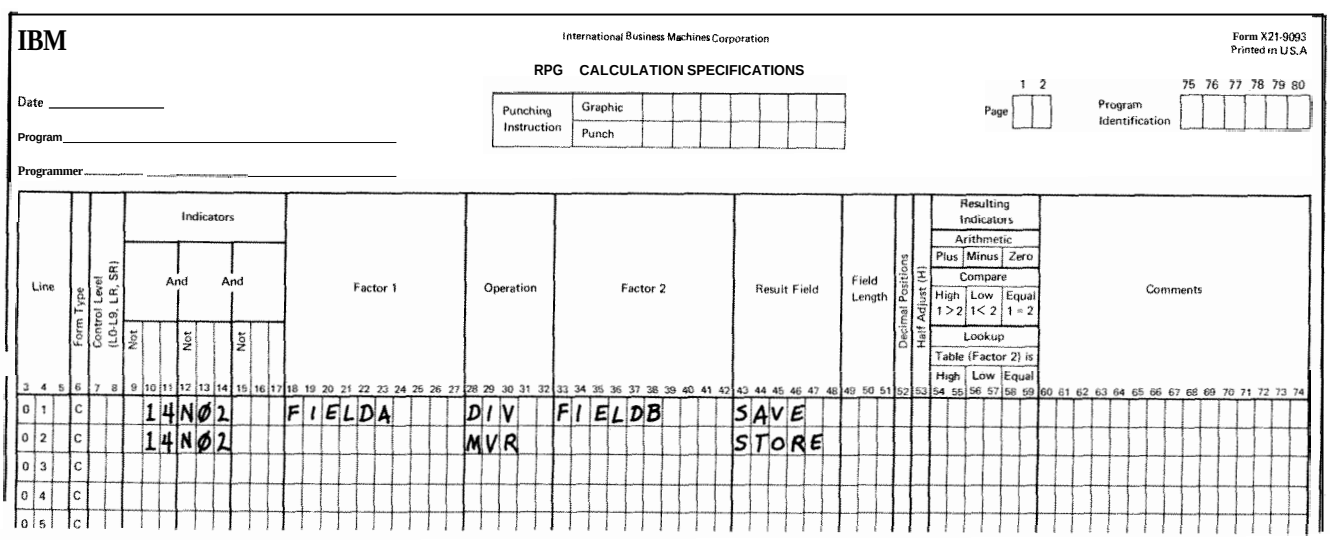


Figure 10-32. Move Remainder Operation

Factor 1 is not used in any move operations. It must always be blank. No resulting indicators may be used. Numeric fields may be changed to alphameric fields and alphameric fields may be changed to numeric fields by the move operations. To change a numeric field to an alphameric field, place the name of the numeric field in the Factor 2 columns and use an alphameric result field. To change an alphameric field to a numeric field, place the name of the alphameric field in the Factor 2 columns and use a numeric result field.

When move operations are specified to move data into numeric fields, decimal positions are ignored. For example, if the data 1.00 is moved into a numeric field with one decimal position, the result is 10.0.

Move (MOVE)

This operation causes characters from factor 2 to be moved to the rightmost position in the result field. Moving starts with the rightmost characters.

If factor 2 is longer than the result field, the excess leftmost characters of factor 2 are not moved. If the result field is longer than factor 2, the characters to the left of the data just moved in are unchanged.

An alphameric field or constant may be changed into a numeric field by moving it into a numeric field When this is specified, the digit portion of each character is converted to its corresponding numeric character and then moved to the result field. Blanks are transferred as zeros. However, the zone portion of the rightmost alphameric character is converted to a corresponding sign and is moved to the rightmost position of the numeric field where it becomes the sign of the field. A numeric field may also be changed into an alphameric field by moving it into an alphameric field. All digits are transferred. The digit and zone of the rightmost character are transferred. The MOVE operation is summarized in Figure 10-33.

Move Left (MOVEL)

This operation causes characters from factor 2 to be moved to the leftmost positions in the result field. Moving begins with the leftmost characters.

If factor 2 is longer than the result field, the excess rightmost characters of factor 2 are not moved. If the result field is longer than factor 2, the characters to the right of the data just moved in are unchanged, In this case the sign of a numeric field is not changed either.

An alphameric field or constant may be changed into a numeric field by moving it into a numeric result field. When this is specified, the digit portion of each character is converted to its corresponding numeric character and then moved into the result field,

Blanks are transferred as zeros. If the rightmost character is moved, the zone is also converted and used as the sign of the field. When the rightmost character is not transferred, the zone is, nevertheless, still transferred and used as the sign of the result field.

A numeric field may also be changed into an alphameric field by moving it into an alphameric field. All digits are transferred. Both digit and zone portions of the rightmost character are transferred if that character is to be moved.

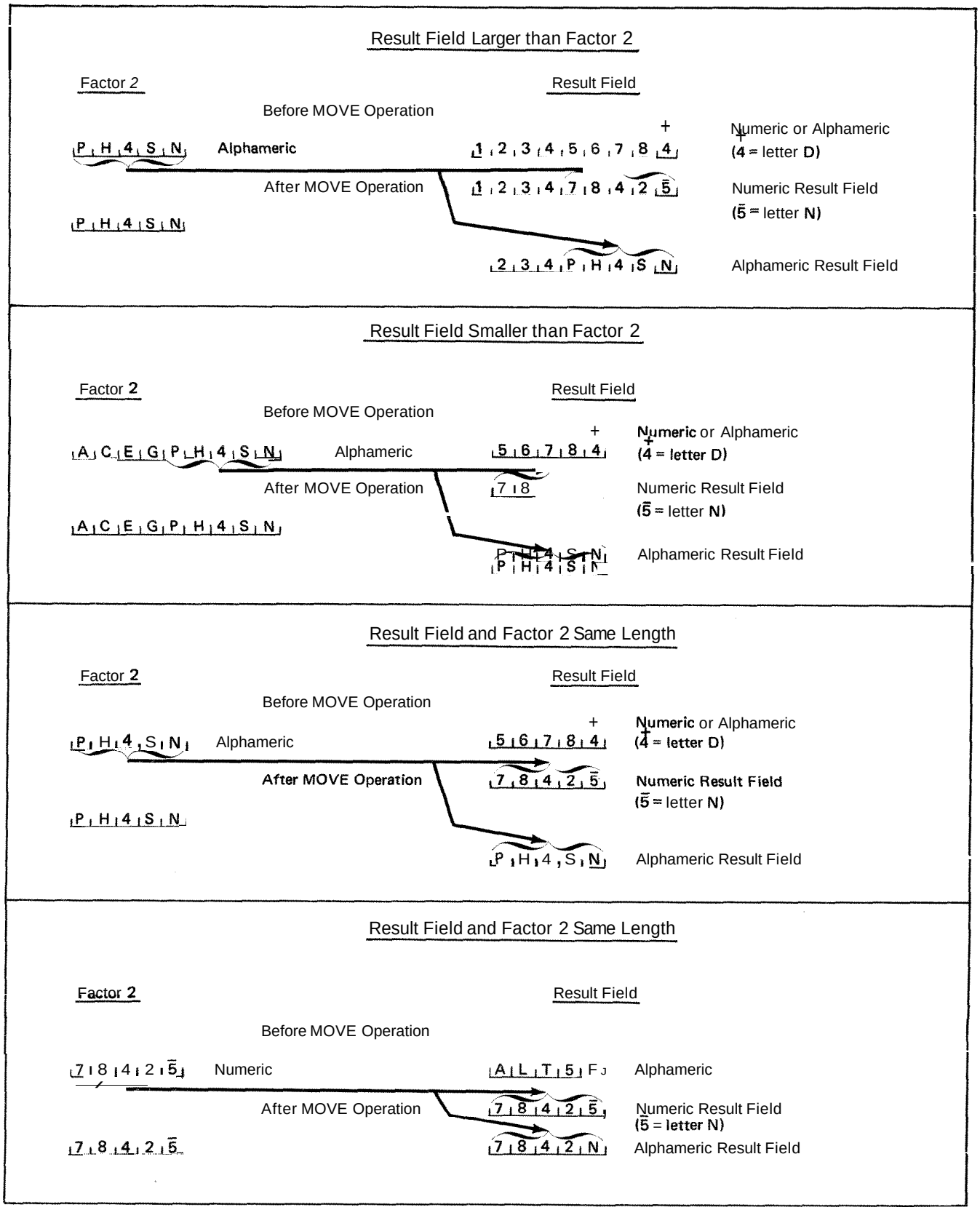


Figure **10-33. MOVE Operations**

A summary of rules for MOVEL transfers are as follows. (See also Figure 10-34.)

1. Factor 2 is the same length as the result field.

   a. Factor 2 and result field numeric: the sign is moved with the rightmost digit.
   b. Factor 2 numeric, result field alphameric: the sign is moved with the rightmost digit. Only digits are moved for other positions.
   c. Factor 2 alphameric, result field numeric: zone and digit portions of rightmost digit are moved. Zones in other positions are not moved.
   d. Factor 2 and result field alphameric: all characters are moved.

2. Factor 2 is longer than the result field.

   a. Factor 2 and result field numeric: the sign from the rightmost position of factor 2 is moved over the rightmost digit of the result field.
   b. Factor 2 numeric, result field alphameric: the result field contains only digits.
   c. Factor 2 alphameric, result field numeric: zone from the rightmost character of factor 2 is moved over the rightmost digit of the result field; other result field positions contain only digits.
   d. Factor 2 and result field alphameric: only the number of characters needed to fill the result field are moved.

3. Factor 2 is shorter than the result field.

   a. Factor 2 either numeric or alphameric, result field numeric: digit portion of factor 2 replaces the contents of the leftmost positions in the result field. The sign in the rightmost position of the result field is not changed.
   b. Factor 2 either numeric or alphameric, result field alphameric: characters in factor 2 replace the equivalent number of leftmost positions in the result field. No change is made in the zone of the rightmost position of the result field.

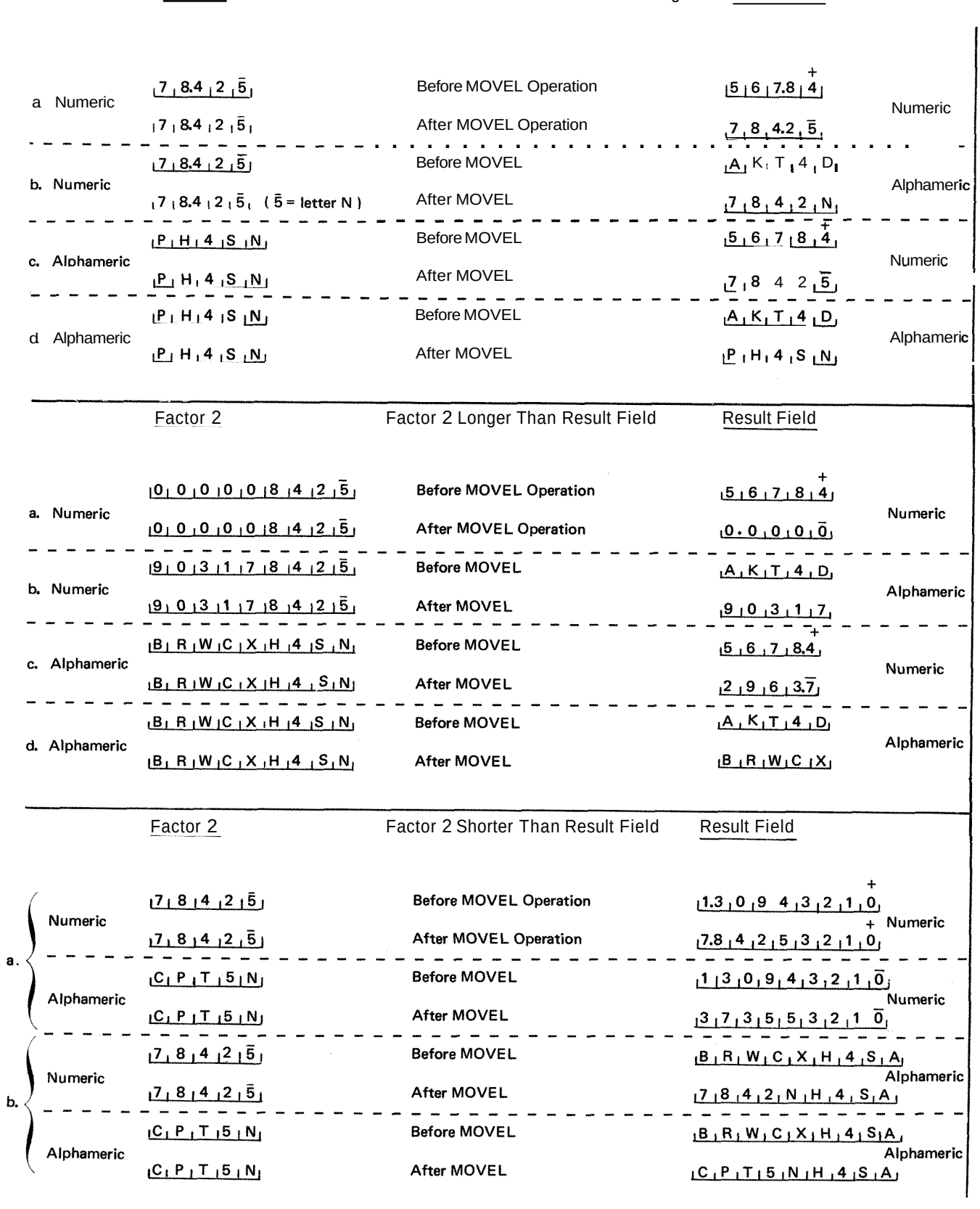


Figure 10-34. MOVEL Operations

Move Zone Operations

These operations are used only to move the zone portion of a character. There are four varieties of the move zone operation (Figure 10-35).

Note: Generally, whenever the word high is used, the field involved must be alphameric; whenever low is used, the field involved may be either alphameric or numeric.

Move High to High Zone (MHHZO)

This operation moves the zone from the leftmost position of factor 2 to the leftmost position of the result field. Factor 2 and the result field must be alphameric.

Move High to Low Zone (MHLZO)

This operation moves the zone from the leftmost position of factor 2 to the rightmost position of the result field. Factor 2 can be only alphameric. The result field may be either alphameric or numeric.

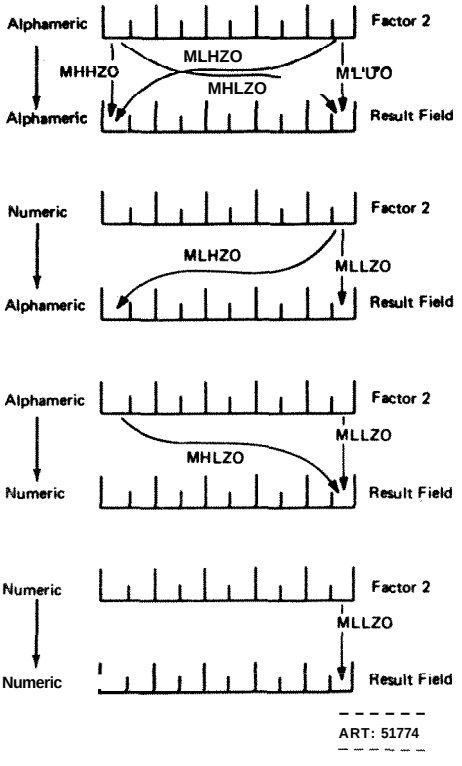


Figure 10-35. Function of Move Zone Operations

Move Low to Low Zone (MLLZO)

This operation moves the zone from the rightmost position of factor 2 to the rightmost position of the result field. Factor 2 and the result field may be either alphameric or numeric.

Move Low to High Zone (MLHZO)

This operation moves the zone from the rightmost position of factor 2 to the leftmost position of the result field. Factor 2 can be numeric or alphameric, but the result field can only be alphameric.

Compare and Testing Operations

These operations test fields for certain conditions. The result of the test is shown by the resulting indicators assigned in columns 54-59. No fields are changed by these operations.

Compare (COMP)

This operation causes factor 1 to be compared with factor 2. As a result of the compare, indicators are turned on as follows:

High	Factor 1 is greater than factor 2.
Low	Factor 1 is less than factor 2.
Equal	Factor 1 equals factor 2.

Factor 1 and factor 2 must either be both alphameric or both numeric.

The fields are automatically aligned before they are compared. If the fields are alphameric, they are aligned to their leftmost character. If one is shorter, the unused positions are filled with blanks (Figure 10-36). The maximum field length for alphameric fields of unequal length which are to be compared is 40 characters. If the fields are of equal length, the maximum is 256 characters.

If the fields which are to be compared are numeric, they are aligned according to the decimal point. Any missing digits are filled in with zeros (Figure 10-37). The maximum field length for numeric fields which are to be compared is 15 digits.

If an alternate collating sequence is defined, alphameric fields are compared according to that sequence.

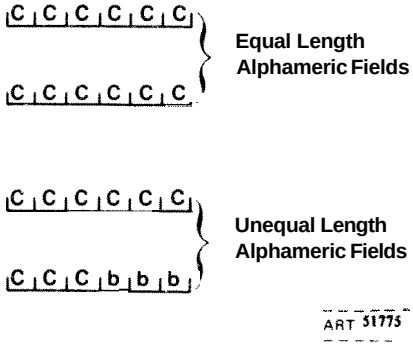


Figure 10-36. Comparison of Alphameric Fields

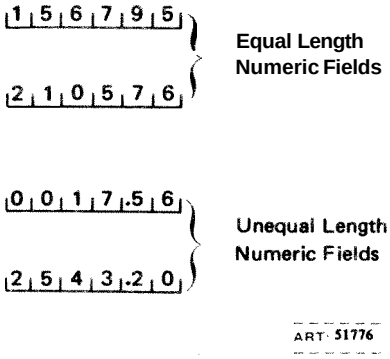


Figure 10-37. Comparison of Numeric **Fields**

Figure 10.38 shows some specifications for compare operations. In specification line 01, the contents of the field SLS67 (1967 sales) are compared with the contents of SLS68. If 1967 sales exceed **1968** sales, resulting indicator 21 turns on; if they are less, resulting indicator 26 turns on; if the two years had equal sales, 30 turns on. In line 03 the alphameric constant **OCTOBER** is compared against the contents of the field named **MONTH** (which must also be defined as alphameric). If the **MONTH** field does not contain the word **OCTOBER,** indicator 13 turns on; if it does, indicator 15 turns on after the compare operation. In line 05 the contents of the field named **GRSPAY** (which must be defined as numeric) is decimal-aligned with numeric constant 1250.00. If the value in field **GRSPAY** is greater than or equal to 1250.00, indicator 04 turns on; if its value is less than 1250.00, indicator 05 turns on. In line 08 the contents of the field NETPAY (which must be defined as numeric) is decimal-aligned with numeric constant 0 and then compared to it, If NETPAY is greater than zero, indicator H1 remains off after the compare operation. If NETPAY is zero or negative, indicator H1 turns on.

IBM

International Business Machines Corporation

Form X21-9093
Printed in U.S.A.

RPG CALCULATION SPECIFICATIONS

Date _____
Program _____
Programmer _____

Punching Instruction: Graphic / Punch

Page 1 2

Program Identification 75 76 77 78 79 80

Line	Form Type	Control Level (L0,L9, LR, SR)	Indicators: And / And — Not	Factor 1	Operation	Factor 2	Result Field	Field Length	Decimal Positions	Half Adjust (H)	Resulting Indicators: Arithmetic (Plus / Minus / Zero); Compare (High 1>2 / Low 1<2 / Equal 1=2); Lookup Table (Factor 2) is (High / Low / Equal)	Comments
01	C			SLS67	COMP	SLS68					21 26 30	
02	C											
03	C			'OCTOBER'	COMP	MONTH					13 13 15	
04	C											
05	C			GRSPAY	COMP	1250.00					04 05 04	
06	C											
07	C			NETPAY	COMP	0					H1 H1	
08	C											
09	C											

Figure 10-38. **Compare** Operations

Test Zone (TESTZ)

This operation tests the zone of the leftmost character in the result field. The result field must be alphameric since this operation can be done only on alphameric characters. Resulting indicators are used to determine the results of the test, The zone portion of characters &, A-I causes the plus indicator to turn on. The zone portion of the characters -, J-R causes the minus indicator to turn on, All other characters, when tested, cause the blank indicator to turn on. Factor 1 and factor 2 are not used in this operation.

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. This allows a space saving ability for binary switches.

Set Bit On (BITONI

This operation code causes specified bits identified in Factor 2, to turn on (set to 1) in a field named as Result Field. In the one position field named in Result Field, one or more of the 8 bits may be turned on. The bits are identified by the numbers 0-7 for each field. The bit numbers must be enclosed by apostrophes. To turn on the first bit in a field, enter '0' in Factor 2, in columns 33-35. To turn on bits 0, 2, and 5 enter '025' in Factor 2 in columns 33-37. From one to eight bits in a field may be turned on with each BITON operation.

The field that contains the eight bits (numbered 0-7) is named in the Result Field. It must be a ***one position alphameric*** field. The field may be an array element, if each element is only one position in length.

The operation code BITON must appear in columns 28-32. Conditioning indicators may be used in columns 7-17, and any entry under Field Length must be ***1.***

Factor 1, Decimal Positions, Half-Adjust, and Resulting Indicators are not used, leave them blank.

Set Bit Off (BITOF)

This operation code causes specified bits, identified in Factor 2, to turn off (set to Ø) in a field named as Result Field.

The operation code BITOF must appear in columns *28-32* All other specifications are the same as those for the BITON operation.

Test Bit (TESTB)

This operation code causes specified bits, identified in Factor 2, to be tested for an on or off condition in the field named as Result Field. The condition of the bits is known by resulting indicators in columns 5459.

The operation code TESTB must appear in columns 28-32. All other specifications are the same as those for BITON and BITOF except for the use of resulting indicators.

At least one resulting indicator must be used with the TESTB operation, and as many as three can be named for one operation. Two indicators may be the same for one TESTB operation, but not three. A resulting indicator has the following meanings for these columns:
Columns 5455: An indicator in these columns is turned on if each bit in Factor 2 is ***off*** **(Ø).**
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.
Columns 58-59: An indicator in these columns is turned on if each bit in Factor 2 is ***on*** (1).

If an array name is used as the Result Field the bits specified in Factor 2 are tested for ***each*** element of the array, and the resulting indicators are set for the array as a whole.

Setting indicators

These operation codes are used to turn indicators off or on. Any indicator to be turned on or off is specified in columns 54-59. The headings in the Resulting Indicators field (Plus or High, Minus or Low, Zero or Equal) have no meaning in these operations. When setting indicators, remember:

1. The following indicators may not be turned on by the SETON operation: 1P, MR, LO, U1-U8.

2. The following indicators may not be turned off by the SETOF operation: 1P, MR, LR, LO, U1-U8.

3. If the LR indicator is turned on by a SETON operation which is conditioned with a control level indicator (columns 7-8 of the Calculation sheet), processing stops after all total output operations are finished. If it is turned on by a SETON operation not so conditioned, processing stops after the next total output operation is completed.

4. If the halt indicators (H1-H9) are set on and not turned off before the detail output operations are complete, the system stops. Processing may be continued by pressing the start key on the Processing Unit once for every halt indicator that is on.

5. Setting on or setting off a control level indicator (L1-L9) does not automatically set any other control level indicator.

6. Indicators L1-L9 and the record identifying indicators are always turned off after detail output operations are completed, regardless of the previous set on or set off operation.

7. Whenever a new record is read, record identifying indicators (01-99) and field indicators are set to reflect conditions on the new record. The setting from any previous SETON or SETOF operation does not apply then.

Set On (SETON)

This operation causes any indicators in columns 54 59 to be turned on.

Set Off (SETOF)

This operation causes any indicators in columns 54 59 to be turned off.

Branching Operations

Operations are normally performed in the order that they appear on the Calculation sheet. There may be times, however, when you do not want the operations performed in the order they are specified. For example, you may wish to:

1. Skip several operations when certain conditions occur.

2. Perform certain operations for several, but not all, record types.

3. Perform several operations over and over again.

Go To (GOTO)

This operation allows you to skip instructions by specifying some other instruction to go to. You may branch to an earlier line or to a later specification line. However, you cannot skip from a calculation that is not conditioned by a control level indicator (columns 7-8) to one that is.

Factor 2 must contain the name of the point to which you wish to go. The name in Factor 2 is called a label. The label can be from 1-6 characters long, and must begin in column 33 with an alphabetic character. The remaining characters can be any combination of alphabetic or numeric characters and special characters. Blanks may not appear between characters in the label. Factor 1 and the result field are not used in this operation. The GOTO operation may be conditioned by any indicators. If it is not conditioned, the operation is always done. See ***Examples*** for use of the GOTO operations.

Tag (TAG)

The operation code names the point to which you are branching in the GOTO operation. Factor 1 contains this label. The name must begin in column 18. The same label may *not* be used for more than one TAG instruction, nor can it be the name of a field used in the program.

Factor 2 and the result field are not used. No indicators may be entered in columns 9-17 for a TAG instruction. Control level indicators must be used, however, if branching is to occur only when the information in a control field has changed. See ***Examples*** for use of the TAG operation.

Examples

Example **I:** Figure 1639 shows how TAG and GOTO may be used to skip operations on certain conditions.

1. **If** the result of the subtraction in line 01 is minus (indicator 10 is on), a branch is taken to RTN1 (routine I) named by the TAG operation code in line 09. Notice that both the GOTO (line 02) and TAG (line 09) are *not* conditioned by control level indicators

2. If the branch is not taken in line 02, the multiplication in line 03 is performed. Then the branch to RTN1 (line 09) must be taken because this branch is not conditioned by indicators.

3. Operations in lines 1 612 are then done. If the operation in line 12 does not turn indicator 15 on, a branch is taken backwards to RTN2 (line 05).

4. Operations then go in the order specified again from lines 06-12. Nothing is done in line 09 since TAG only gives a name. These same operations are performed again and again until 15 does turn on.

5. When 15 is on, the branch to RTN2 is not taken. The TESTZ operation is then performed. **If** this operation causes 20 to turn on, a branch is taken to line 17 (GOTO END). **If** 20 is not on, the operation in line 16 is done.

IBM

International Business Machines Corporation

Form X21-9093
Printed in U.S.A.

RPG CALCULATION SPECIFICATIONS

Date

Program

Programmer

Punching Instruction: Graphic / Punch

Page 1 2

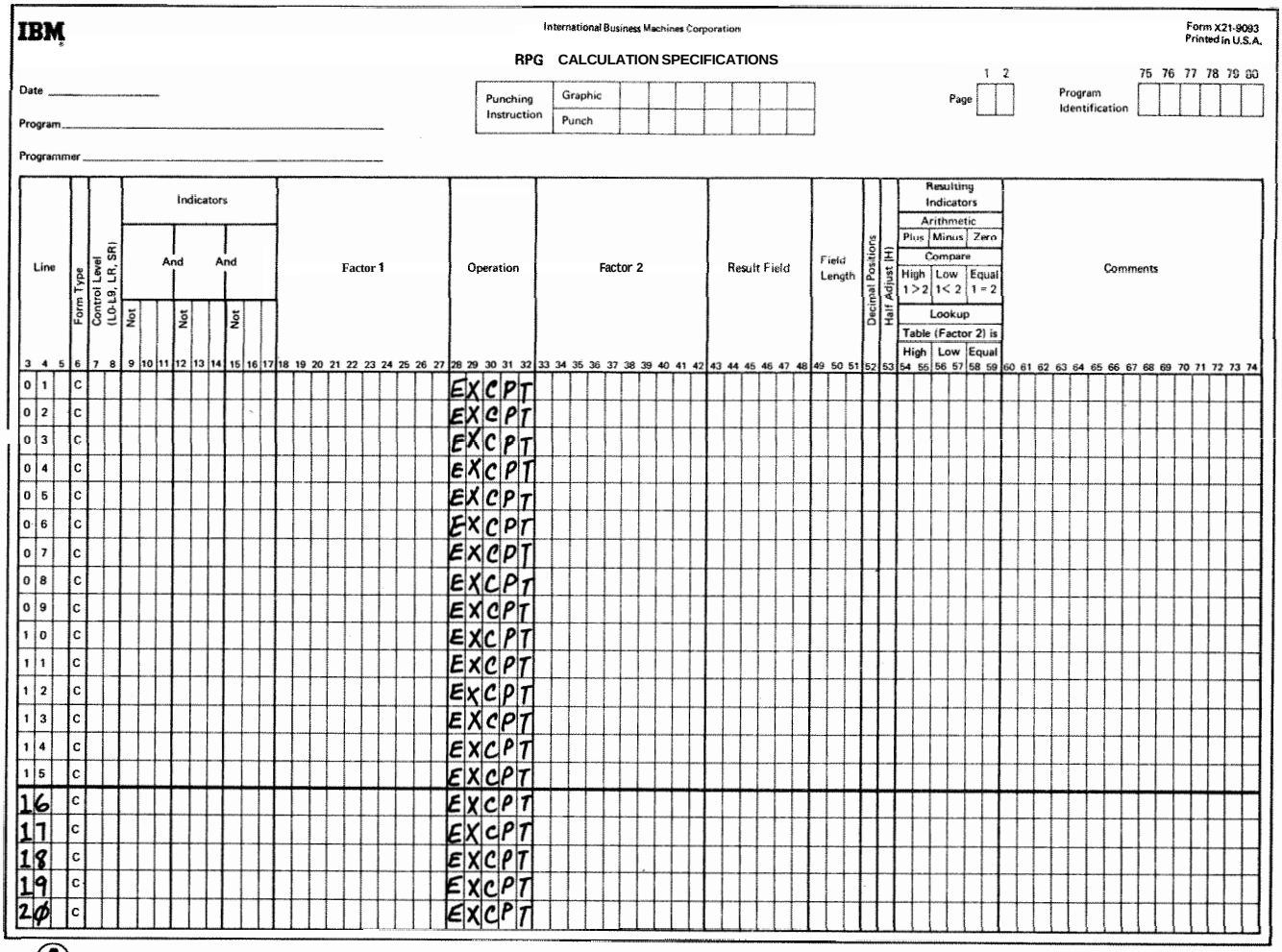
Program Identification 75 76 77 78 79 80

Line	Form Type	Control Level (L0-L9, LR, SR)	Indicators: And / Not	Factor 1	Operation	Factor 2	Result Field	Field Length	Decimal Positions	Half Adjust (H)	Resulting Indicators: Arithmetic (Plus / Minus / Zero); Compare (High 1>2 / Low 1<2 / Equal 1=2); Lookup Table (Factor 2) is (High / Low / Equal)	Comments
01	C			FIELDA	SUB	FIELDB	FIELDB	52			10	
02	C		10		GOTO	RTN1						
03	C			FIELDB	MULT	4	SAVE	82				
04	C				GOTO	RTN1						
05	C			RTN2	TAG							
06	C					Some Calculation Operations						
07	C											
08	C											
09	C			RTN1	TAG							
10	C					Some Calculation Operations						
11	C											
12	C										15	
13	C		N15		GOTO	RTN2						
14	C				TESTZ		FIELDC				20	20
15	C		20		GOTO	END						
	C				MHLZO	FIELDC	FIELDD					
	C			END	TAG							
	C											
	C											
	C											

Figure 10-39. Using **GOTO and TAG (Skipping Operations)**

Example 2: Figure 10-40 shows how TAG and GOTO may be used to eliminate coding when several operations have to be performed again and again

Assume that you wish to make 20 mailing labels for every customer you have, The customer's name and address are found on an input card. Since you wish to



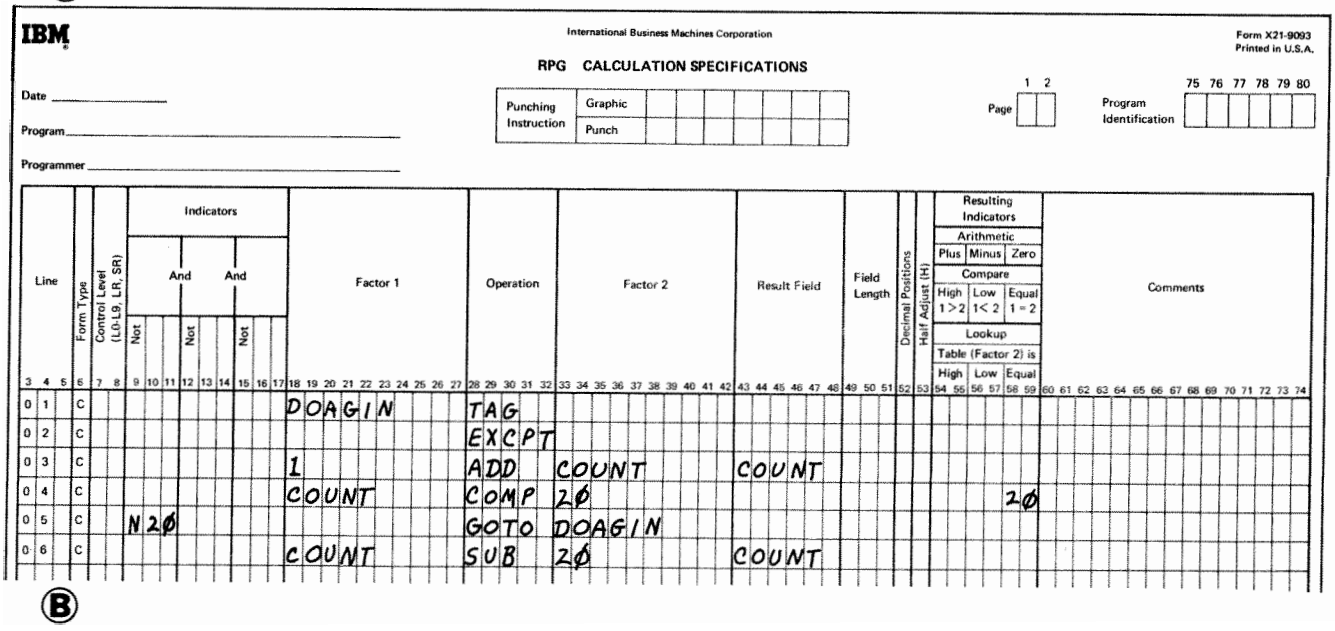


Figure 10-40. **Using GOTO and TAG (Eliminate Duplicate Coding)**

write twenty labels for each card, you have to use exception lines and the operation EXCPT (see EXCPT *operation* in this section for further information).

This can be coded as shown in Figure 10-40, insert A. You have to write the EXCPT operation code for every mailing label. However, by using branching, you can code it all in five lines (see Figure 10-40, insert B). An EXCPT line is printed out. One is added to COUNT in order to keep track of how many times the line has been printed. Then COUNT is compared to 20. If COUNT does not equal 20, a branch is taken back to the beginning (GOTO DOAGIN). If COUNT equals 20, the branch is not taken. Instead 20 is subtracted from the COUNT field so that it will be zero for the next cycle.

Lookup Operations

Lookup operations are used when searching through a table or an array to find a special element.

Lookup (LOKUP)

This operation code causes a search to be made for a particular item in a table or array. The table or array is factor 2. Factor 1 is the search word (data for which you wish to find a match in the table or array named). Factor 1, the search word, may be:

1. An alphameric or numeric constant.
2. A field name.
3. An array element.
4. A table name.

Remember that when a table is named in Factor 1, it refers to the element of the table last selected in a LOKUP operation, not to the whole table.

Resulting indicators are always used in connection with LOKUP. They are used to first indicate the type of search desired and then to reflect the result of the search. A resulting indicator assigned to Equal (columns 58-59) instructs the program to search for an entry in the table or array equal to the search word. The indicator turns on only if such an entry is found. If there are several entries identical to the search word, the first one that is encountered is selected.

An indicator assigned to Low (columns 56-57) instructs the program to locate an entry in the table that is nearest to, yet lower in sequence than, the search word. The first such entry found causes the indicator assigned to Low to turn on.

The indicator assigned to High (columns 5455) instructs the program to find the entry that is nearest to, yet higher in sequence than, the search word. The first higher entry found causes the indicator assigned to High to turn on. In all cases the resulting indicator turns on only if the search is successful.

At least one resulting indicator must be assigned, but no more than two can be used. Resulting indicators can be assigned to Equal and High or Equal and Low. The program searches for an entry that satisfied either condition with Equal given precedence; that is, if no Equal entry can be found, the nearest lower or nearest higher entry is selected. If resulting indicators are assigned both to High and Low, the indicator assigned to Low is ignored. When using the LOKUP operation, remember:

1. The search word and each table or array item must have the same length, the same number of decimal positions (if used), and the same format (alphameric or numeric).

2 You may search on High, Low, High and Equal, or Low and Equal only if your table or array is in sequence,

3. No resulting indicator turns on if the entry searched for is not found.

Using the LOKUP Operation

LOKUP with One Table

When searching a single table, factor 1, factor 2, and at least one resulting indicator must be specified. Conditioning indicators (specified in columns 7-17) may also be used.

Whenever a table item is found that satisfies the type of search being made (Equal, High, Low), a copy of that table item is placed in a special storage area Every time a search is successful, the newly found table item is placed in this area, destroying what was there before. If the search is not successful, no table item is placed in the storage area Instead the area keeps the contents it had before the unsuccessful search.

Resulting indicators are always set to reflect the result of the search. If the indicator is on, showing a successful search, you know that a copy of the item searched for is in the special storage area.

LOKUP with Two Tables

When two related tables are used in a search, only one is actually searched. When the search condition (High, Low, Equal) is satisfied, the corresponding data items from both tables are placed in their respective special storage areas and are made available for use.

Factor 1 must be the search word and factor 2 must name the table to be searched. The result field must name the related table from which data is made available for use. Resulting indicators must also be used. Conditioning indicators (specified in columns 7-17) may be specified if needed.

The two tables involved should be the same length. If the table that is searched is longer than its related table, it is possible to satisfy the search condition. However, there will not be a data table item in the second table which corresponds to the item found in the search table. Unpredictable results may occur.

LOKUP with an Array

The LOKUP specifications for arrays are the same as for tables except that if Factor 2 is an array, the result field cannot be used. In addition if the desired item is found, it is not moved to a special holding area since these holding areas are only associated with tables. Instead, the indicators will reflect only that the desired item is in the array; the programmer does not have ready access to this item

Figure 10-41 shows two LOKUP operations performed with an array. MANNOS, a 2100 element m a y of employee numbers, is read in at execution time from file ARRFILE with six 10 position elements per record; the array elements are in ascending order. Line 01 of the Calculation sheet shows a LOKUP of array MANNOS with the object of finding the element nearest to but higher in sequence than the search word '100336'. If this desired element is found in the array, indicator 20 turns on and the GOTO in line 02 is performed, Notice that the result of this LOKUP indicates only whether or not the desired element exists in the array. Line 05 of the Calculation sheet shows essentially the same LOKUP operation-- indicator 20 will turn on when the fust element higher in sequence than '100336' is found. Note, however, that in this LOKUP operation, the array MANNOS is indexed by the field INX This index field was set to I in line 04 so the LOKW will begin at the first element of MANNOS. If the desired element is found, the number of this element (not its contents) is placed in the field INX In this

IBM

International Business Machines Corporation

Form X21-9091
Printed in U.S.A.

RPG EXTENSION AND LINE COUNTER SPECIFICATIONS

Date _____
Program _____
Programmer _____

Punching Instruction: Graphic / Punch

Page 1 2

Program Identification 75 76 77 78 79 80

Extension Specifications

Line	Form Type	Record Sequence of the Chaining File / Number of the Chaining Field / From Filename	To Filename	Table or Array Name	Number of Entries per Record	Number of Entries Per Table or Array	Length of Entry	P = Packed/B = Binary	Decimal Positions	Table Sequence (A/D)	Table or Array Name (Alternating Format)	Length of Entry	P = Packed/B = Binary	Decimal Positions	Table Sequence (A/D)	Comments
3 4 5	6	7 8 / 9 10 / 11 12 13 14 15 16 17 18	19 20 21 22 23 24 25 26	27 28 29 30 31 32	33 34 35	36 37 38 39	40 41 42	43	44	45	46 47 48 49 50 51	52 53 54	55	56	57	58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74
01	E	ARRFILE		MANNOS	100	2100	6			A						
02	E															

IBM

International Business Machines Corporation

Form X21-9093
Printed in U.S.A.

RPG CALCULATION SPECIFICATIONS

Date _____
Program _____
Programmer _____

Punching Instruction: Graphic

Page 1 2

Program Identification 75 76 77 78 79 80

Line	Form Type	Control Level (LO-L9, LR, SR)	Indicators: And / Not / And / Not / And / Not	Factor 1	Operation	Factor 2	Result Field	Field Length	Decimal Positions	Half Adjust (H)	Resulting Indicators: Arithmetic (Plus / Minus / Zero); Compare (High 1>2 / Low 1<2 / Equal 1=2); Lookup Table (Factor 2) is (High / Low / Equal)	Comments
3 4 5	6	7 8	9 10 11 12 13 14 15 16 17 18	19 20 21 22 23 24 25 26 27	28 29 30 31 32	33 34 35 36 37 38 39 40 41 42	43 44 45 46 47 48	49 50 51	52	53	54 55 56 57 58 59	60 61 62 63 64 65 66 67 68 69 70 71 72 73 74
01	C			'100336'	LOKUPMANNOS						20	
02	C		20		GOTO NEXT							
03	C											
04	C				Z-ADD1		INX	40				
05	C			'100336'	LOKUPMANNOS, INX						20	
06	C		N20		GOTO END							
07	C				MOVE MANNOS, INXSAVE			60				
08	C			NEXT	TAG							
09	C											
10	C											
11	C											
12	C											
13	C											
14	C											

Calculation Operations

Figure 10-41. LOKUP With an Array

way, the actual element which satisfied the **LOKUP** can be used in subsequent calculation operations, as in line 07. If no element was found to satisfy the LOKUP, the field **INX** would be reset to 1. Refer to *Starting the Search at a Particular Array Item* in this section for more information on indexing an array in a **LOKUP** operation.

Examples

Figures 10-42 through 10-45 show the use of the **LOKUP** operation. Figure 10-37, insert **A,** shows the contents of four tables: table A, table B, table C, and table D (loaded at compile time). Each table has five entries.

Figure 10.42, insert B, shows the extension specifications for these tables. Table A and Table B are described separately and are, therefore, entered separately. Tables C and D are related tables and are entered in alternating format on the table input cards. Figure 10.43 shows the order in which the table input cards are loaded into the machine at compile time.

Figure 10-44 shows 15 different **LOKUP** operations using these four tables. The results of these operations are shown in Figure 1045. Figure 10-45 tells if the entry searched for was found, and if so, what indicator is on to tell the result of the search, what table item satisfied the search, what item was taken from a related table (when one is used).

	First Entry	Second Entry	Third Entry	Fourth Entry	Fifth Entry
Table A	01	05	08	32	96
Table B	05.13	02.12	47.15	28.70	15.18
Table C	WWW	NNN	LLL	GGG	AAA
Table D	7	9	3	2	5

Ⓐ

International Business Machines Corporation

RPG EXTENSION AND LINE COUNTER SPECIFICATIONS

Punching Instruction: Graphic / Punch

Page 1 2

Extension Specifications

Table or Array Name	Number of Entries per Record	Number of Entries Per Table or Array	Length of Entry	P = Packed/B = Binary	Decimal Positions	Table Sequence (A/D)	Table or Array Name (Alternating Format)	Length of Entry	P = Packed/B = Binary	Decimal Positions	Table Sequence (A/D)
TABLEA	2	5	2			A					
TABLEB	5	5	4	2							
TABLEC	1	5	3			D	TABLED	1	Ø		

Ⓑ

ART 51780

Figure 10-42. Table Lookup (Tables Used)

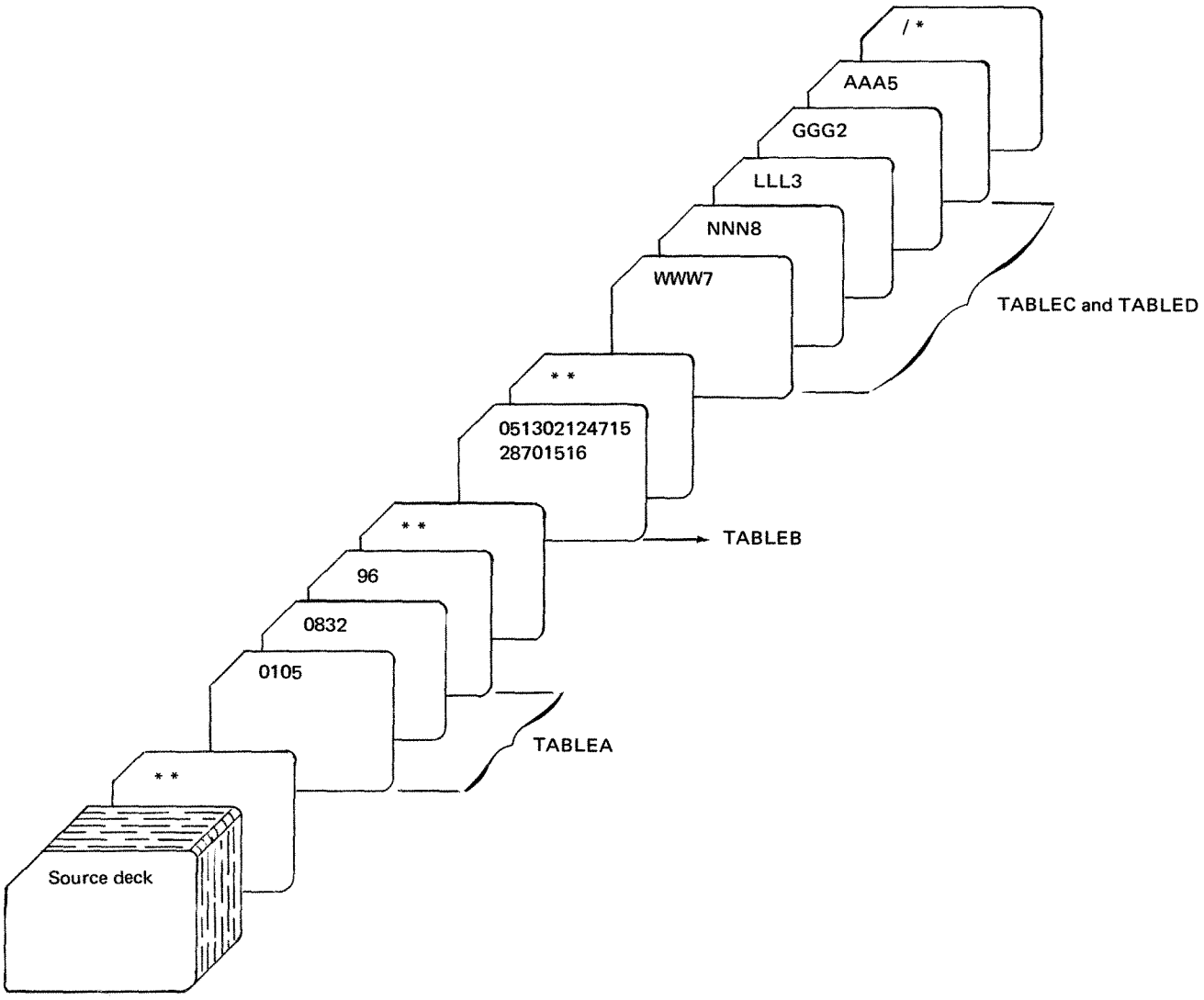
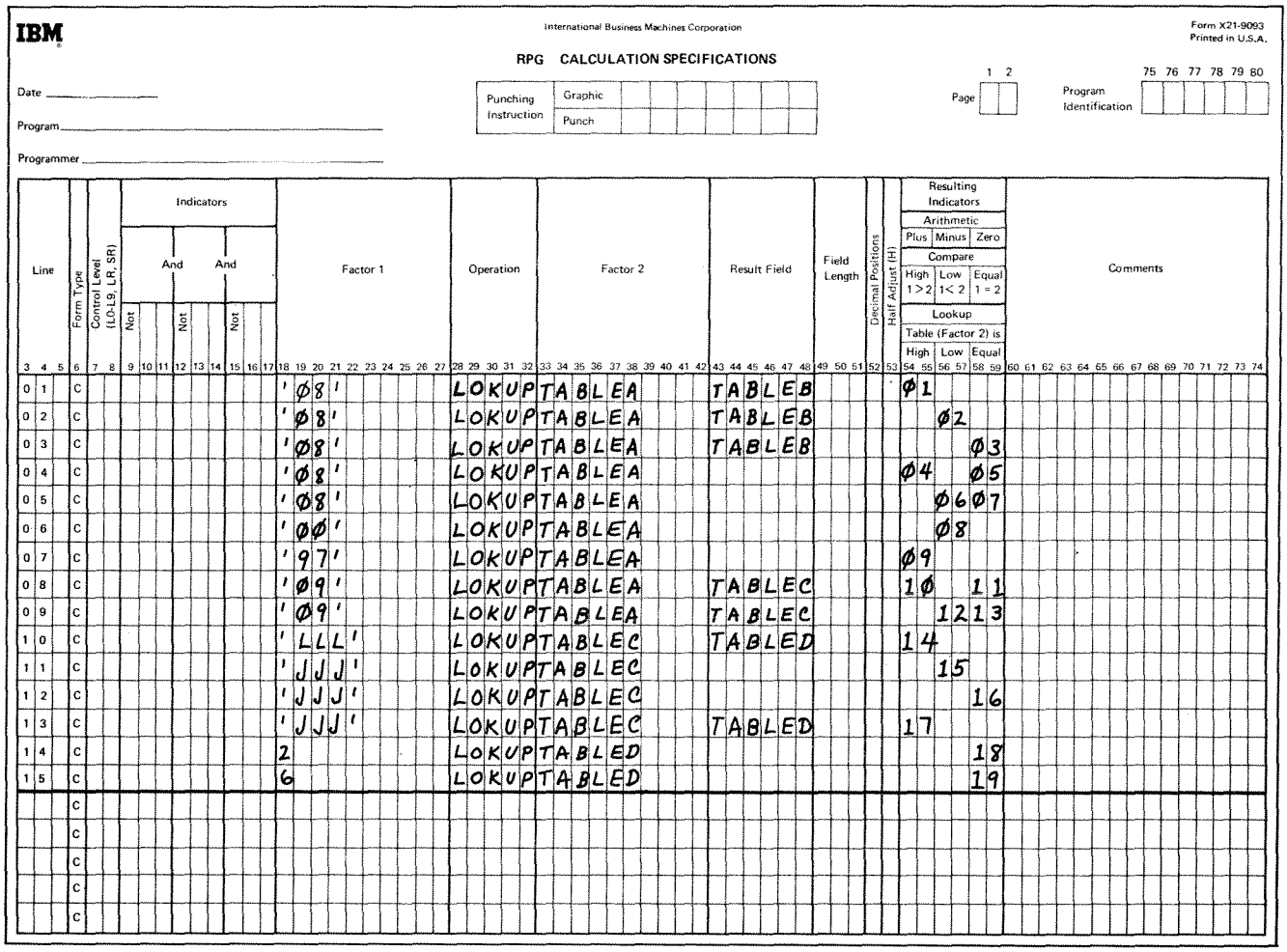


Figure 10-43.. TABLEA, TABLEB, TABLEC, and TABLED (Loaded at Compile Time)



IBM

International Business Machines Corporation

Form X21-9093
Printed in U.S.A.

RPG CALCULATION SPECIFICATIONS

Date _____

Program _____

Programmer _____

Punching Instruction: Graphic / Punch

Page 1 2

Program Identification 75 76 77 78 79 80

Line 3 4 5	Form Type 6	Control Level (L0-L9, LR, SR) 7 8	Indicators: And / Not 9 10 11	Indicators: And / Not 12 13 14	Indicators: And / Not 15 16 17	Factor 1 18 19 20 21 22 23 24 25 26 27	Operation 28 29 30 31 32	Factor 2 33 34 35 36 37 38 39 40 41 42	Result Field 43 44 45 46 47 48	Field Length 49 50 51	Decimal Positions 52	Half Adjust (H) 53	Resulting Indicators: Arithmetic: Plus / Minus / Zero; Compare: High 1>2 / Low 1<2 / Equal 1=2; Lookup: Table (Factor 2) is: High / Low / Equal 54 55 56 57 58 59	Comments 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74
01	C					'08'	LOKUP	TABLEA	TABLEB				01	
02	C					'08'	LOKUP	TABLEA	TABLEB				02	
03	C					'08'	LOKUP	TABLEA	TABLEB				03	
04	C					'08'	LOKUP	TABLEA					04 05	
05	C					'08'	LOKUP	TABLEA					06 07	
06	C					'00'	LOKUP	TABLEA					08	
07	C					'97'	LOKUP	TABLEA					09	
08	C					'09'	LOKUP	TABLEA	TABLEC				10 11	
09	C					'09'	LOKUP	TABLEA	TABLEC				12 13	
10	C					'LLL'	LOKUP	TABLEC	TABLED				14	
11	C					'JJJ'	LOKUP	TABLEC					15	
12	C					'JJJ'	LOKUP	TABLEC					16	
13	C					'JJJ'	LOKUP	TABLEC	TABLED				17	
14	C					2	LOKUP	TABLED					18	
15	C					6	LOKUP	TABLED					19	
	C													
	C													
	C													
	C													
	C													

Figure 10-44. Table LOKUP Operation

Specificetion Line Number	Entry Found	Indicator On	Table Item Satisfying Search Condition	Table Item Used from Related Table
01	yes	01	32	28.70
02	yes	02	05	02.12
03	yes	03	08	47.f5
04	yes	05	08	
05	yes	07	08	
06	no			
07	no			
08	yes	10	32	GGG
09	yes	12	08	LLL
10	yes	14	NNN	8
11	yes	15	GGG	
12	no			
13	yes	17	LLL	3
14	yes	18	2	
15	no			

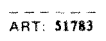


Figure 10-45. Results of LOKUP Operations

Referencing the Table Item Found in a LOKUP Operation

Whenever a table name is used in an operation other than LOKUP, the table name really refers to the data placed in the special table storage area by the last successful search Thus, by specifying the table name in this fashion, you can use data items from a table in calculation operations.

If the table is used as factor 1 in a LOKUP operation, the contents of the special table storage area is used as the search word. In this way a data item from a table can itself become a search word.

The table may also be used as the result field in operations other than the LOKUP operation. In this case the contents of the special table storage area is changed by the calculation operation. The corresponding table item in the table itself is also changed. This is a way in which you can modify the contents of the table by calculation operations (see Figure 10-46).

Starting the Search at a Particular Array Item

It is possible, in order to save processing time, to start the LOKUP search at a particular item in the array. This type of search is indicated by additional entries in columns 33-42, Enter the name of the array to be searched in these columns followed by a comma and a numeric literal or the name of a numeric field (with no decimal positions). The numeric literal or numeric field tells the number of the item at which you wish to start the search (Figure 10-47). This numeric literal or field is known as the index because it points to a certain item in the array. All other columns are used as previously described for the normal lookup operatian.

The search starts at the specified item and continues until the desired item is found or until the end of the array is reached. When an index field is used, an unsuccessful search causes the index field to contain the value of one. If, however, an item is found which satisfies the

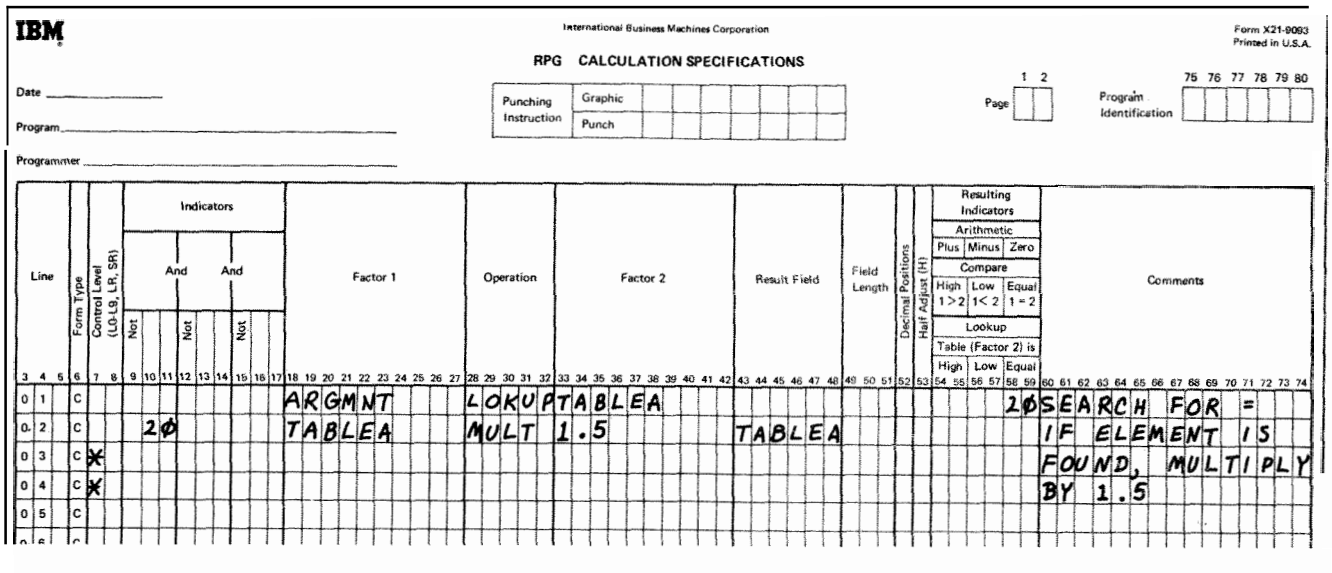


Figure 10-46. Referencing the Table Item Found in a LOKUP Operation

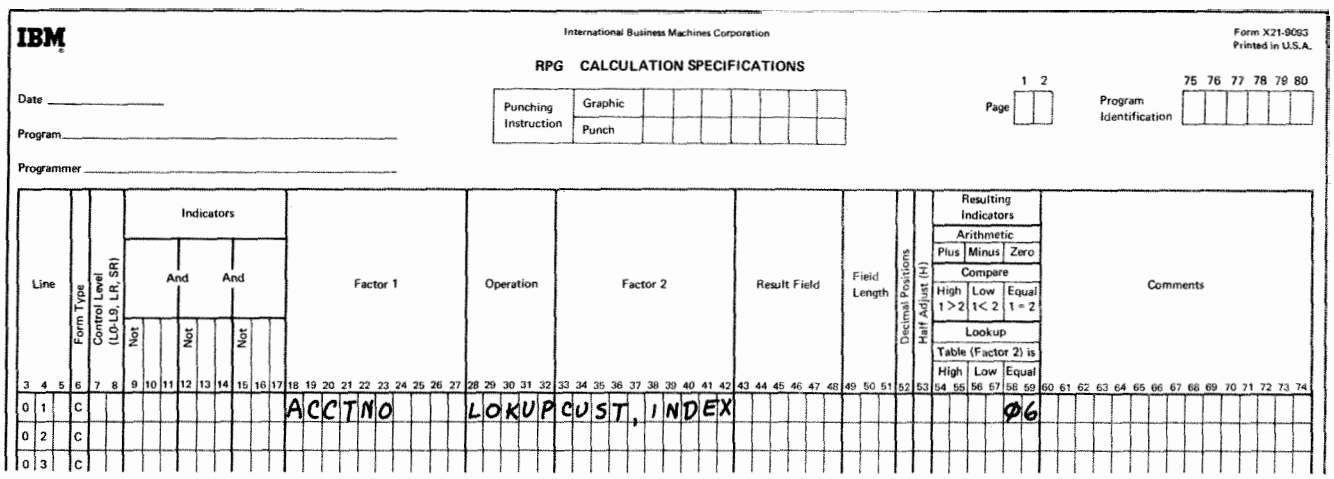


Figure 10-47. Array Lookup: Starting at a Particular Array Item

conditions of the LOKUP operation, the number of that array item (counting from the first item) is placed in the index field. A numeric literal used as an index is not changed to reflect the result of the search.

Note: If a literal or field index for an array is zero, or greater than the number of elements in the array, the following will result:

1. For a literal index a severe error occurs, and compilation will cease.
2. For a field index the job will halt, allowing the operator to cancel or restart the program If the program is restarted, the field index is given a value of one.

Subroutine Operations

These operation codes are only used for subroutines. See Subroutines in this chapter for information on subroutines. All subroutine operation codes must be written in specification Lies following all operations conditioned by control level indicators specified in columns 7-8. Subroutine lines are always identified by an SR in columns 7-8.

Begin Subroutine (BEGSR)

This operation code serves as the beginning point of the subroutine. Factor 1 must contain the name of the subroutine.

End Subroutine (ENDSR)

This operation code must be the last statement of the subroutine. It serves to define the end of the subroutine. Factor 1 may contain a name. This name then serves as a point to which you can branch by a GOTO statement within the subroutine. The ENDSR operation ends the subroutine and automatically causes a branch back to the next statement after the EXSR operation.

Execute Subroutine (EXSR)

This operation causes all the operations in the subroutine to be performed. EXSR may appear anywhere in the program. Whenever it appears, the subroutine is executed. After all operations in the subroutine are done, the operation in the line following the EXSR operation is performed.

This operation may be conditioned by any indicators, meaning the subroutine is executed only when all conditions are satisfied. Factor 2 must contain the name of the subroutine that is to be executed. This same name must appear on a BEGSR instruction.

Programmed Control of Input and Output

Normally a record is read, and calculations are performed on data from that record, Then any data from that input record resulting from calculations on data from that record is written. At this time only the records that have been specified in the output-format specifications will be written or punched Programmed control allows you to have more control concerning which records from the different files are to be read next, when records are to be written, and how many should be written.

Exception (EXCPT)

This operation allows records to be written at the time calculations are being done. Use this primarily when you wish to have a variable number of similar or identical records (either detail or total) written in one program cycle. (Remember that normally only the exact number of records specified in the output-format specifications are written on a file in one program cycle.) For example, you might use EXCPT to produce a variable number of identical mailing labels, to write out contents of a table, or to produce a number of records having the same information punched on them.

When the EXCPT operation is used, EXCPT is entered in columns 28-32, and columns 7-17 may have entries. All other columns must be blank. The line or lines which are to be written out during calculation time are indicated by an E in column 15 of the Output-Format sheet. Exception lines may not be used in a combined file.

Figure 10-48 shows the use of the EXCFT operation to produce a variable number of records having the same information punched on them. Records in the input file have two fields, NAME and COUNT. The NAME field is to be entered into a certain number of records. That number is indicated in the COUNT field.

Every time the operation code EXCPT is performed, the exception record indicated by the *E* in column 15 of the Output-Format sheet is punched. The field CONSEC is used to keep track of the number of records punched, Each time an exception record is written, 1 is added to CONSEC. CONSEC is then compared with COUNT, the field that tells how many records should be punched. If they are not equal (indicator 20 is not on), a branch is taken back to DOAGIN. Another record is written out. One is added to CONSEC and CONSEC is compared to COUNT. If these fields are now equal, another input record is read. If not, the same operations are done again. Whenever CONSEC equals COUNT, enough records have been punched or printed. CONSEC is then subtracted from itself, making it zero. This last operation is necessary so that an accurate count can be kept for the next record.

Force (FORCE)

FORCE statements enable you to select the file from which the next record is to be taken for processing. They apply to primary, secondary, or demand; input, update, or combined files. They are the only means by which records can be read from demand files.

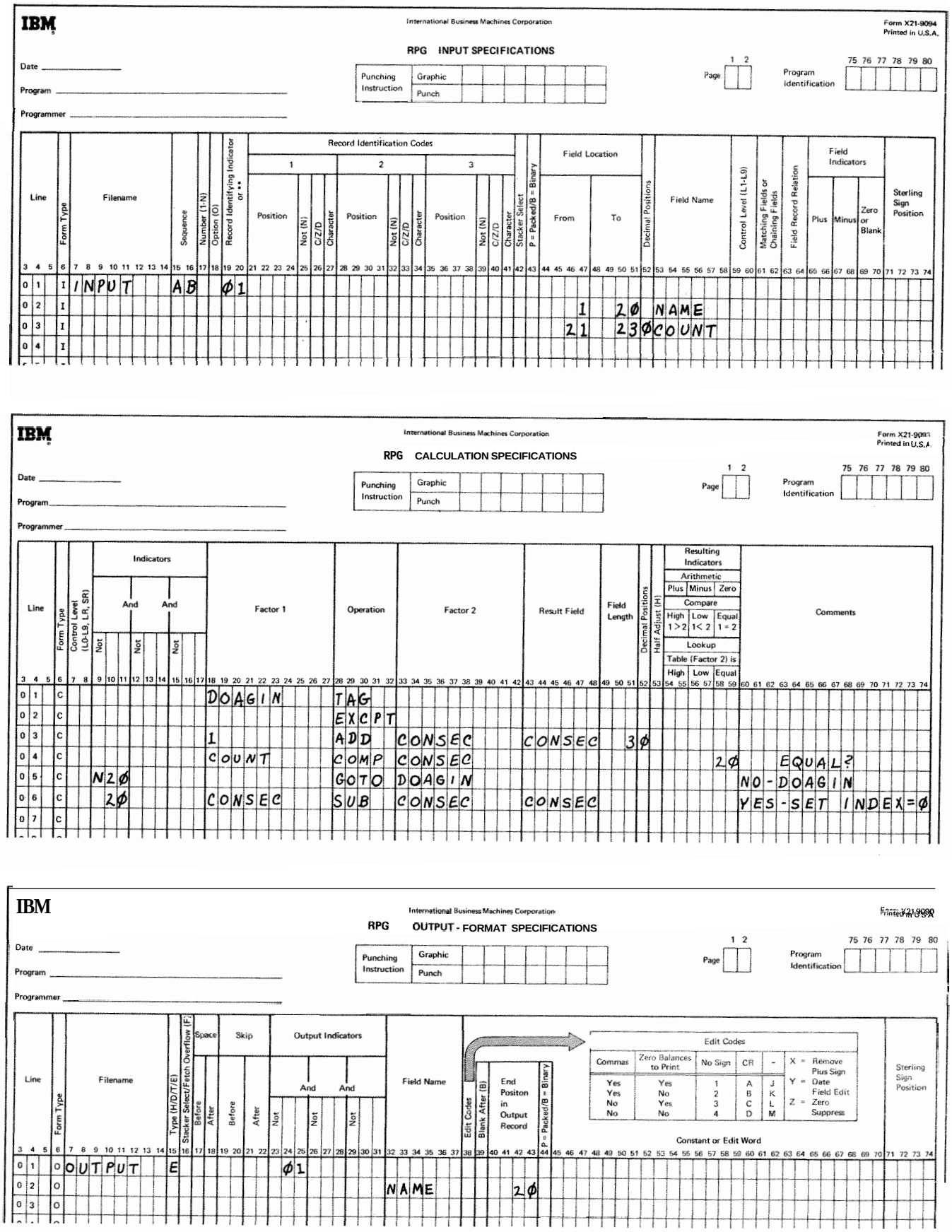


Figure 10-48. EXCPT Operation (Producing a Variable Number of Identical Records)

Factor 2 in a FORCE statement identifies the file from which the next record is to be read. If the statement is executed, the record is read at the start of the next program cycle. If more than one FORCE statement is executed during the same program cycle, all but the last is ignored.

FORCE statements override the multifile processing method by which the program normally selects records However, the first record to be processed is always selected by the normal method. The remaining records can be selected by FORCE statements.

Example

Figure 10-49 shows part of a job which uses FORCE operation codes and look ahead fields to simulate normal record selection. Normal record selection is not used because records in the two secondary files have two match fields, CUST and ITEM, and those in the primary file have only one, CUST. Normal record selection requires all three to have the same number of match fields

Indicators 20-23 and 26-28 are used to determine which file the next record is to be read from. The conditions under which the files are chosen follow. Record 1 means the record from the primary files; record 2 the first secondary file; and record 3, the second secondary file.

Condition	Indicators Set On	File Selected
None of the records match. Record 1 has the lowest CUST field value.	20 and 22	Primary (FIRST)
Record 1 matches record 2 Record 3 has a higher CUST field value.	21 and 22	Primary (FIRST)
Record 1 matches record 3. Record 2 has a higher CUST field value.	20 and 23	Primary (FIRST)
Records 1, 2, and 3 match (CUST field values).	21 and 23	Primary (FIRST)
Record 2 has lower CUST field value than record 1. Record 2 has lower CUST and ITEM fields (together) value than record 3.	26	First secondary (SECOND)

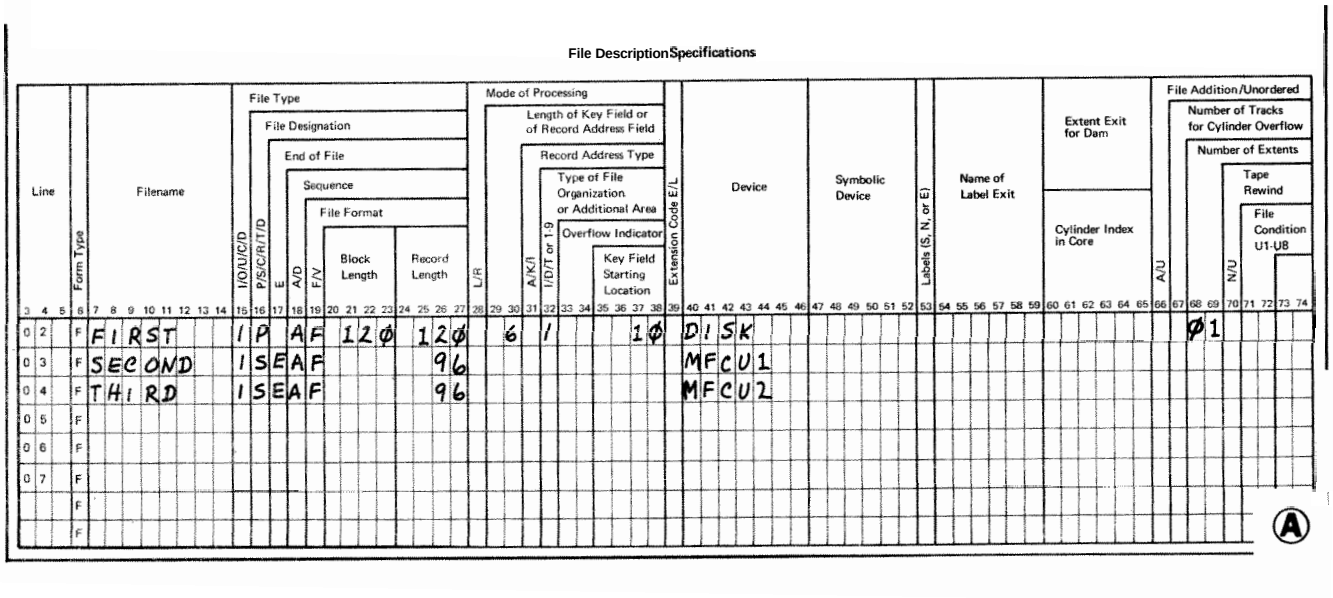
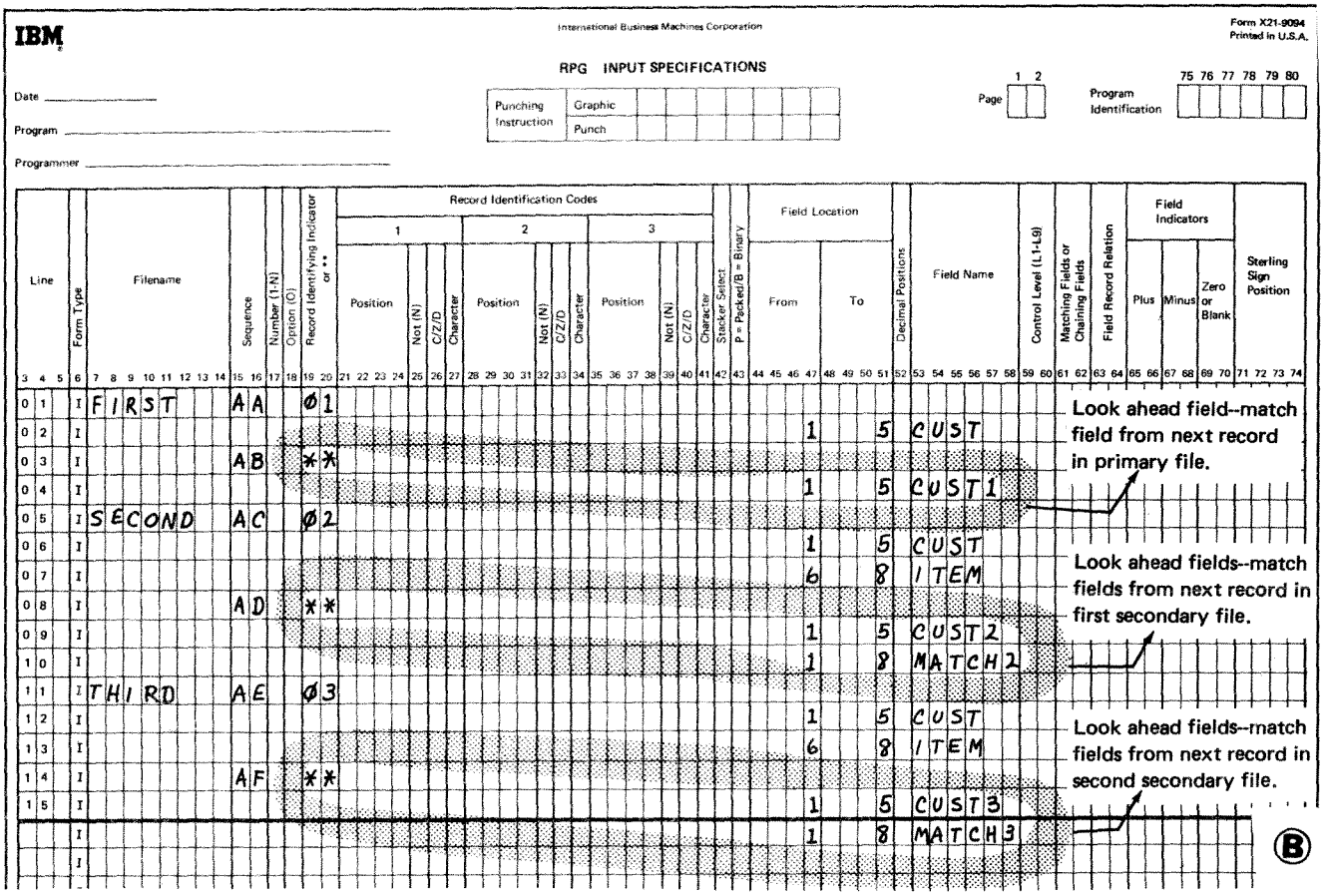


Figure 10-49. FORCE Operation Code (Part 1 of 2)



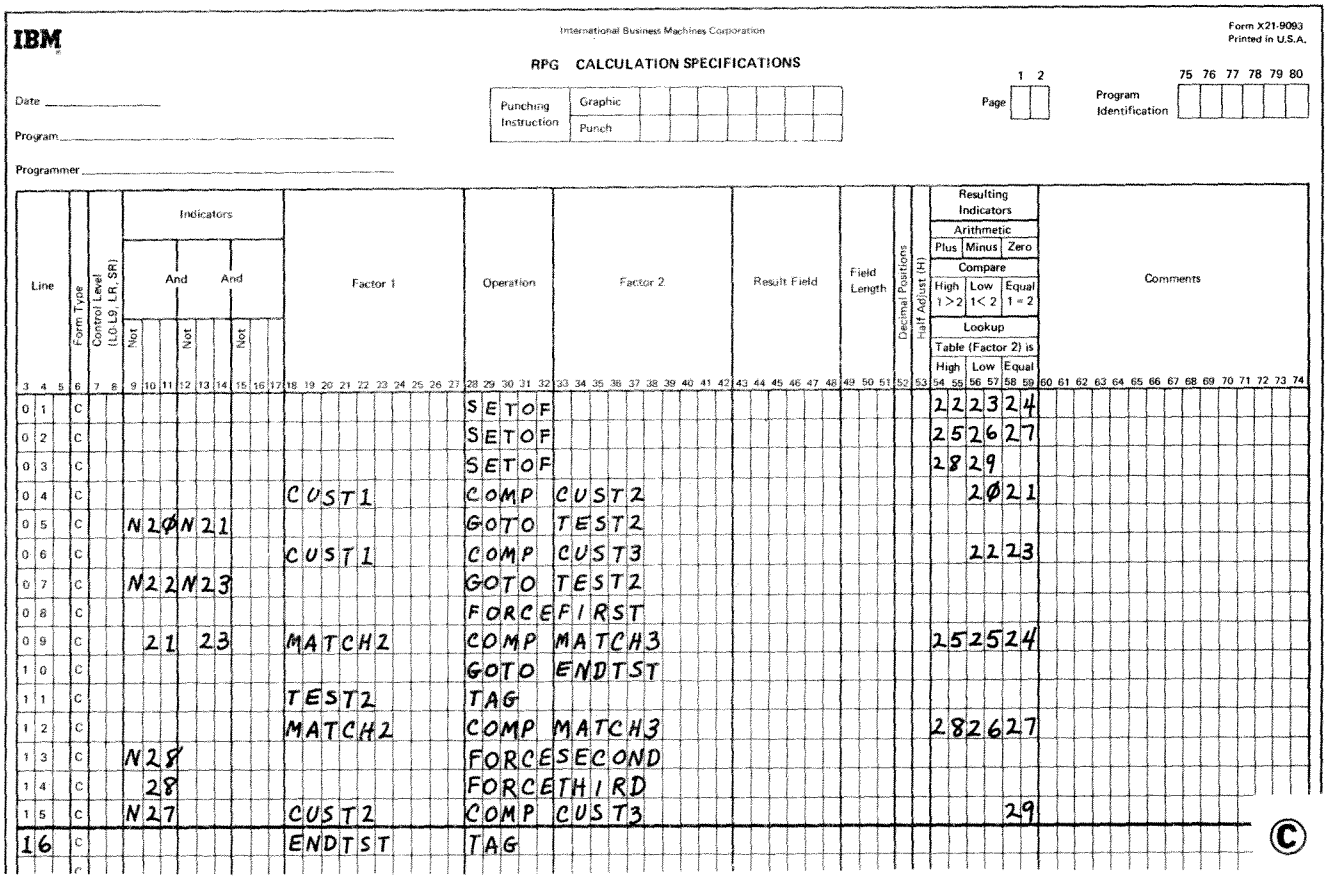


Figure 10-49. FORCE Operation Code (Part 2 of 2)

Condition	*Indicators Set On*	*File Selected*
Record 2 matches record 3 (both CUST and ITEM fields). Record 1 has greater CUST field value.	27	First secondary (SECOND)
Record **3** has lower CUST field value than record 1. Record **3** has lower CUST and ITEM field (together) value than record 2.	28	Second secondary (THIRD)

In addition, indicators 24, 25, and 29 are set to condition calculations which process the record selected.

Condition	*Indicator Set On*
Records 1, 2, and **3** match (CUST fields). Records 2 and **3** match (CUST fields and ITEM fields).	24
Records 1, 2, and **3** match (CUST fields). ITEM fields in records 2 and **3** do not match.	25
CUST field values in records 2 and **3** match; ITEM fields do not, Record 1 has higher CUST field value.	29

All the calculations shown in Figure 10-49, insert C, are needed to determine which record is to be processed next. The operations which are performed upon the data from the input records are not shown. They do, however, precede the calculations shown in Figure 10-49, insert C, and are conditioned by the indicators set during the previous cycle by the calculations shown.

Display (DSPLY)

The display operation allows either or both of the following:

1. A field or array element is printed on the printer-keyboard during program execution without a program halt.

2. A field or array element is printed on the printer-keyboard, and the program halts, allowing that field to be changed.

See Figure 10-50 for coding possibilities and results. Also see Figure 10-51 under CHAIN operation in this chapter for an example using the display operation.

When the display operation is used, DSPLY must appear in columns 28-32, and the filename of the console device under Factor 2. Indicators in columns 7-17 may be specified. Field length, Decimal positions, Half-adjust, and Resulting Indicators (columns 49-59) must be left blank. No input or output specifications are required for this operation, However, the File Description sheet must have entries in columns: 7-14, 15, 19, 24-27, and 40-46 (columns 71-72 are optional).

If data is to be printed but not changed, enter a field name, an array name plus an index, or a literal in Factor 1, and the filename of the console device in Factor 2. Result Field must be blank in this case. The data in Factor 1 will be printed, but not changed, and the program will continue.

If data is to be changed during program execution enter the field name or array name plus index under Result Field, and the filename of the console device under Factor 2. This causes the data to be printed and then blanked out. Immediately after the field or array element is blanked out, the program halts. The operator can now enter data into the blank field or array element via the printer-keyboard. There are several points to remember when this is done:

1. The data entered must be followed by a function key character.

2. Numeric data need not be entered with leading zeros. However, you must be sure to right-justify numeric data when it is keyed in.

3. Similarly, alphameric fields must be left justified by the operator when it is keyed in.

4. Alphameric fields are blanked out. Numeric fields are zeroed out.

Chain (CHAIN)

The chain operation causes a record to be read from a disk file during calculations. This operation allows one record to be read in when the operation code CHAIN appears in columns 28-32 of the Calculation sheet.

Indicators in columns 7-17 may be used, but Result Field, Field Length, Decimal Position, and Half-Adjust (columns 45-53) must be blank. If the chained file is conditioned in the file description specifications by an external indicator, the CHAIN statement must be conditioned by that same external indicator.

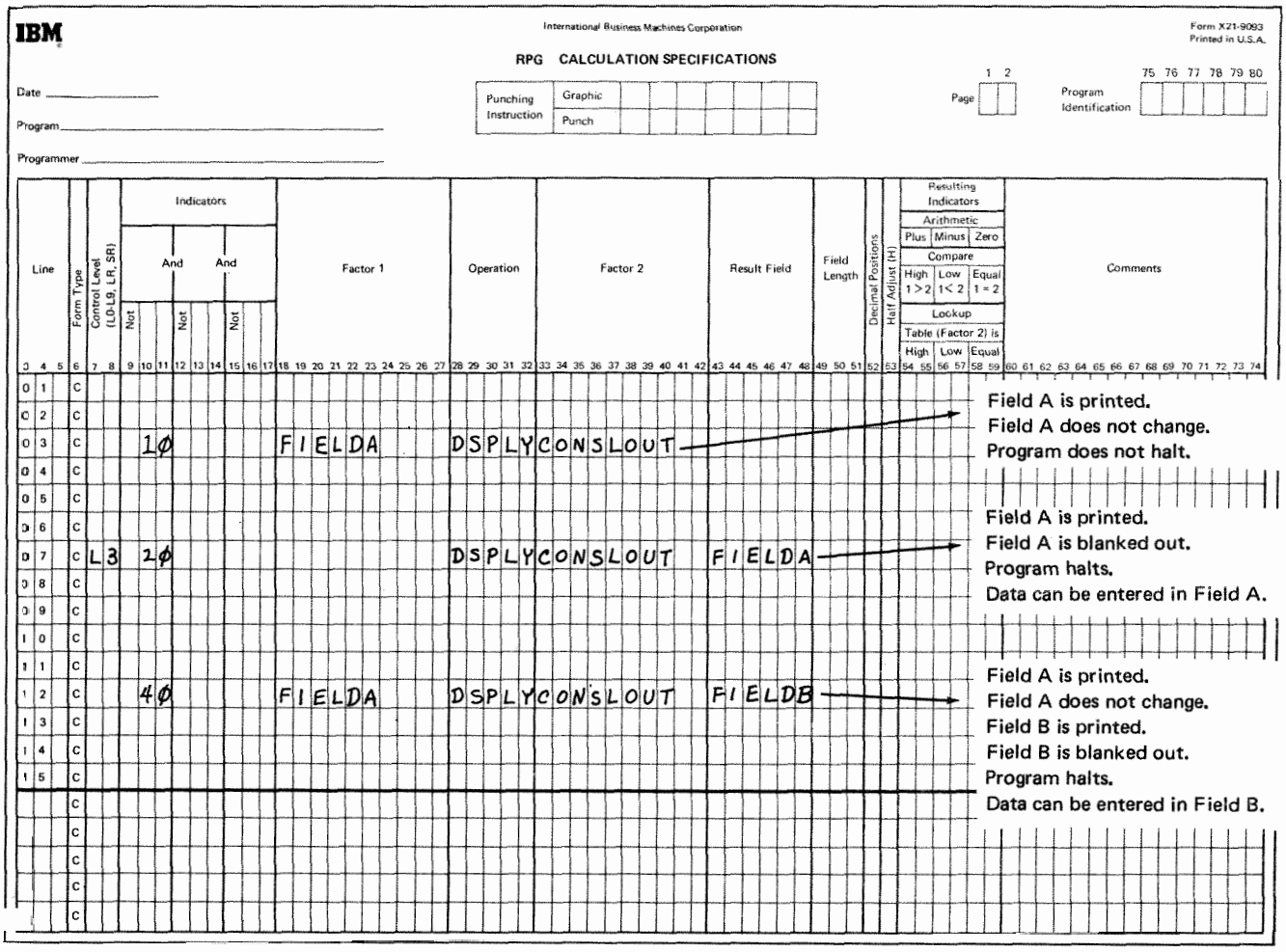


Figure 10-50. Display Operation

Columns 54-55 may contain an entry. If they do, the same entry must be made in columns 56-57. If the record is not found, the indicator specified in these columns will turn on. No output is permitted to a chained update file when the specified record is not found. Columns 58-59 must always be blank for chain operations.

If an indicator is not specified in columns 5457, and the record is not found, the program will halt. Processing can be continued by pressing the start key on the Processing Unit.

The chain operation is used for two purposes:

1 Random processing of an indexed, sequential, or direct file.

2 Loading a direct file.

Random Processing

In order to read a record from a sequential or direct file, the record must be identified by relative record number. To read a record from an indexed file, a record key is used for identification. The relative record number or key can be contained in a field specified for that purpose.

The chain operation requires the code word CHAIN in columns 28-32 of the Calculation sheet. Factor I entries must be a relative record number or key, or the name of a numeric field that contains a relative record number or key. Factor 2 must contain the name of the file from which the record will be read. This Me is called the file that is chained to, or the chained file (see ***Examples, Example 1***).

Direct File **Load**

A direct file load is defined by specifying the disk file to be loaded as a chained output file on the File Description sheet. In the calculation specifications, Factor 1 must contain a relative record number, columns 28-32 must contain the operation code CHAIN, and Factor 2 must contain the name of the direct disk Me to be loaded. The relative record number of the input record defines the record position for each record in the direct disk Me. The relative number can be dl or part of a field in the input records. Such fields are used for record identification of the input records, as well as for the disk records after the disk file is loaded.

When a direct file is loaded you must define the record length and number of records in your file. The system then clears the disk space required for the file with blanks.

When a record is read in, the relative record number is used to chain to the corresponding relative record position in the disk file. The blanks at that record position are read in, and the information contained in the input record is then written on disk, replacing the blanks with data. If a record is missing from the input file when a direct file is loaded, the space reserved for that record in the disk file remains blank (until the proper record is read in later). A direct file is *loaded* by defining it as a chained output file in file description specifications (see *Examples, Example 2*).

Once the direct file is loaded, records are *inserted* or *changed* in the file by defining the direct file as an update file processed consecutively or by the chain operation. (Remember that any file defined as a chained output file will be cleared entirely to blanks before any records are processed.)

Note: The insertion of records in direct disk files is very different from record addition to sequential or indexed files. For sequential disk files, the new record is added in at the first available position at the end of the fie. The same process occurs for an indexed file, except that the record key and disk address are added to the file index. Any new records inserted in a direct disk file already have a space reserved for them. Hence, the record is inserted in its proper place, not merely added to the physical end of the Me.

Examples

***Example* I:** Figure 10-51 shows the coding necessary to chain to and update an indexed file, MASTINV. The CARDIN file consists of cards sorted by item number, each card representing some quantity ordered. Item number is used as a control field. When all the quantities for one item number are added, a control break will occur. At this point in calculations, the master record for that item number must be found and updated. ITEMNO is a field containing the item number of the cards presently being worked on. The chain operation uses ITEMNO to find the master record for that item number. If it is not found, a display operation prints out the item number of the cards. Note that indicator (20) turns on when the records are *not* found.

If the master record is found (20 not on) the total quantity for the item number is subtracted from the quantity on hand. After the total calculations, the QOH field in the master record is updated.

File Description Specifications

Line	Filename	Form Type	I/O/U/C/D	P/S/C/R/T/D	E	A/D	F/V	Block Length	Record Length	L/R	A/K/I	I/D/T or 1-9	Type of File Organization or Additional Area	Overflow Indicator	Key Field Starting Location	Extension Code E/L	Device	Symbolic Device	Labels (S, N, or E)	Name of Label Exit	Extent Exit for Dam	Cylinder Index in Core	A/U	Number of Tracks for Cylinder Overflow	Number of Extents	N/U	Tape Rewind	File Condition U1-U8
02	CARDIN	F	I	P	E	A	F		96								MFCU1											
03	MASTINV	F	U	C			F	120	120	R	9	K	I		1		DISK							Ø1				
04	CONSLOUTD	F					F		9								CONSOLE											
05		F																										
06		F																										
07		F																										
		F																										
		F																										

IBM

International Business Machines Corporation

Form X21-9094
Printed in U.S.A.

RPG INPUT SPECIFICATIONS

Date _____

Program _____

Programmer _____

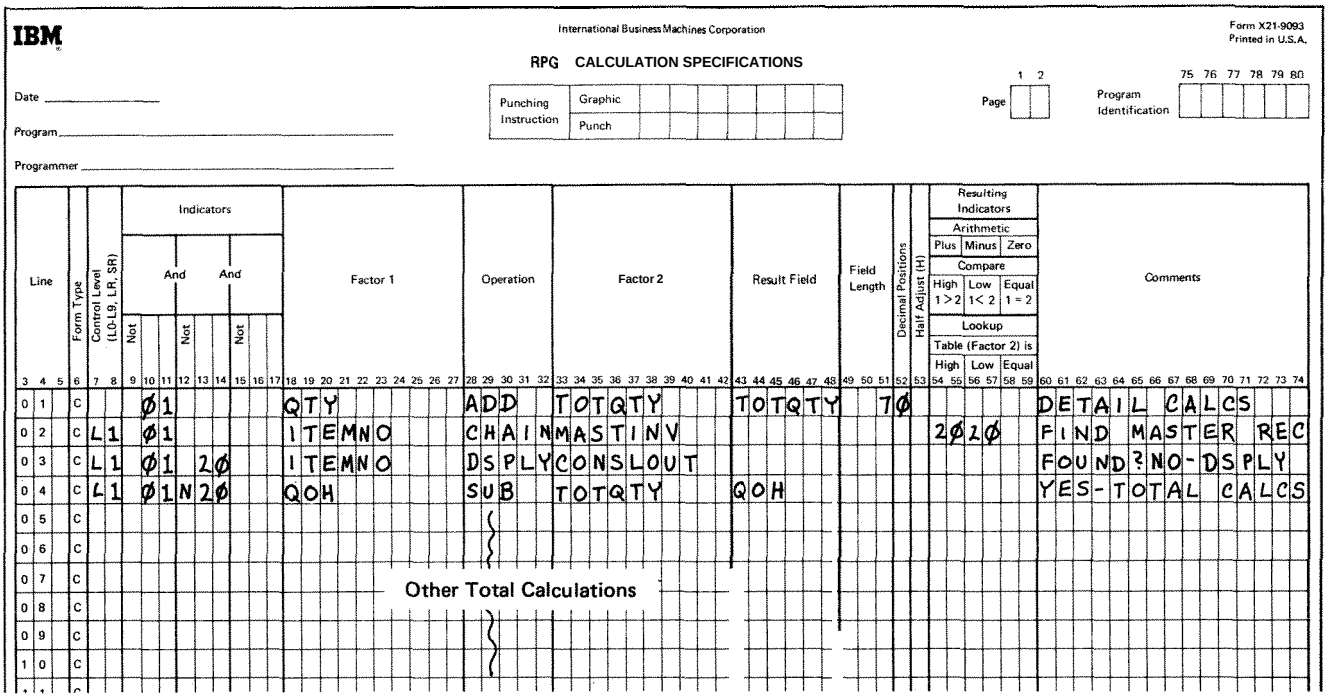
Punching Instruction: Graphic / Punch

Page 1 2

Program Identification 75 76 77 78 79 80

Line	Filename	Form Type	Sequence	Number (1-N)	Option (O)	Record Identifying Indicator or **	Record Identification Codes 1: Position	1: Not (N)	1: C/Z/D	1: Character	2: Position	2: Not (N)	2: C/Z/D	2: Character	3: Position	3: Not (N)	3: C/Z/D	3: Character	Stacker Select	P = Packed/B = Binary	Field Location: From	Field Location: To	Decimal Positions	Field Name	Control Level (L1-L9)	Matching Fields or Chaining Fields	Field Record Relation	Field Indicators: Plus	Field Indicators: Minus	Field Indicators: Zero or Blank	Sterling Sign Position
01	CARDIN	I	AA			Ø1	96		C	X																					
02		I																			21	29		ØITEMNOL	1						
03		I																			3Ø	36		ØQTY							
04	MASTINV	I	BB			Ø2	120		C	1																					
05		I																			1	9		ØITMNO							
06		I																			1Ø	18		ØQOH							
07		I																													

Figure 10-51. **Chain** Operation **(Part 1 of 2)**



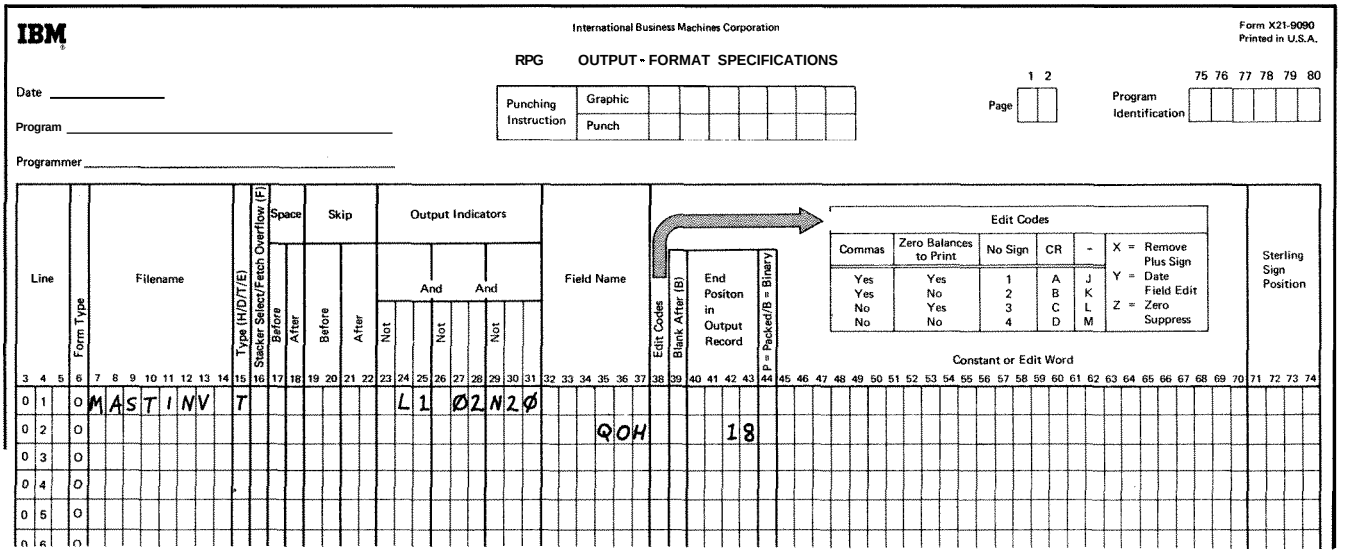
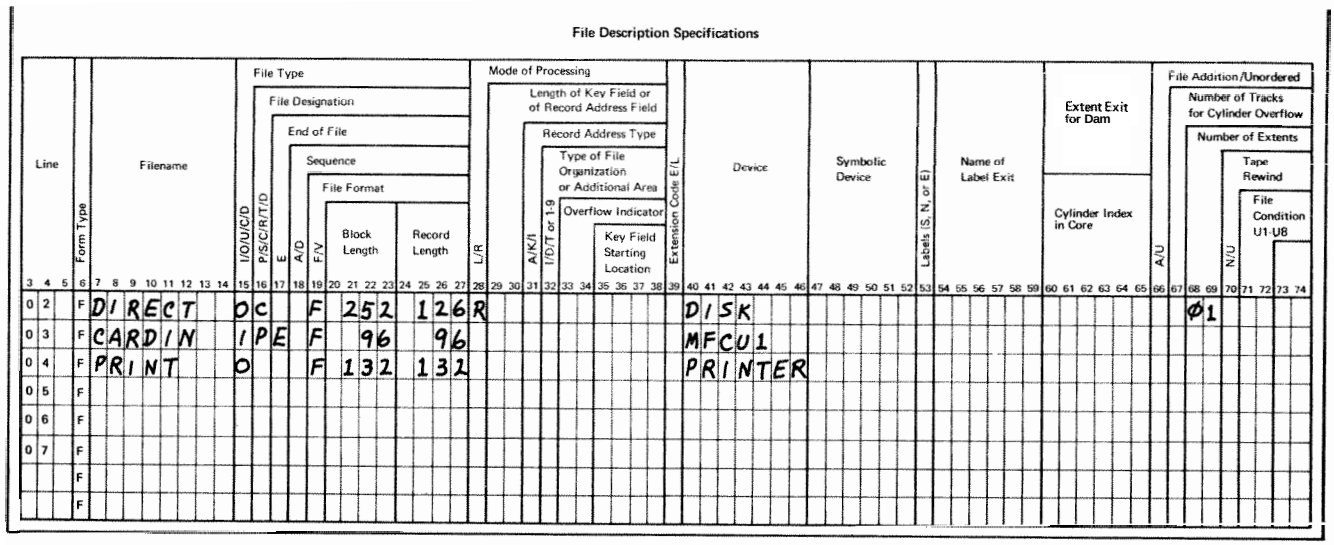


Figure 10-51. Chain Operation (Part 2 of 2)

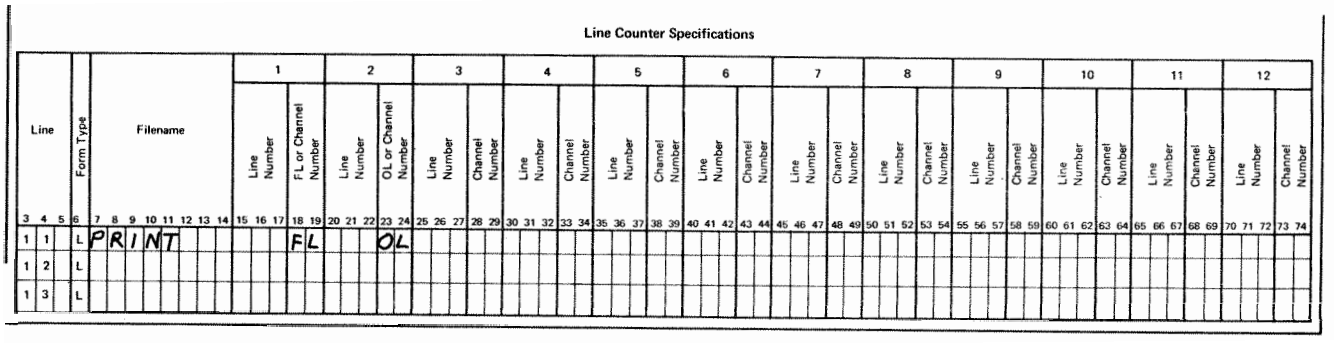
Example 2: Assume that you are loading a direct file with customer information such as: customer number, address, year-to-date sales, balance due, and other data for a master record. Each record is 126 characters long and is identified by customer number. At present there are 200 customers, but you plan to have more in the future so space will be reserved on disk for 250 records altogether.

The direct file of customers is named DIRECT on the File Description sheet (see Figure 1052). This direct file will be loaded from the MFGU. The card file with the customer data is named **CARDIN.** The printer file PRINT enables us to print out any cards that may contain errors or that may be duplicate cards.



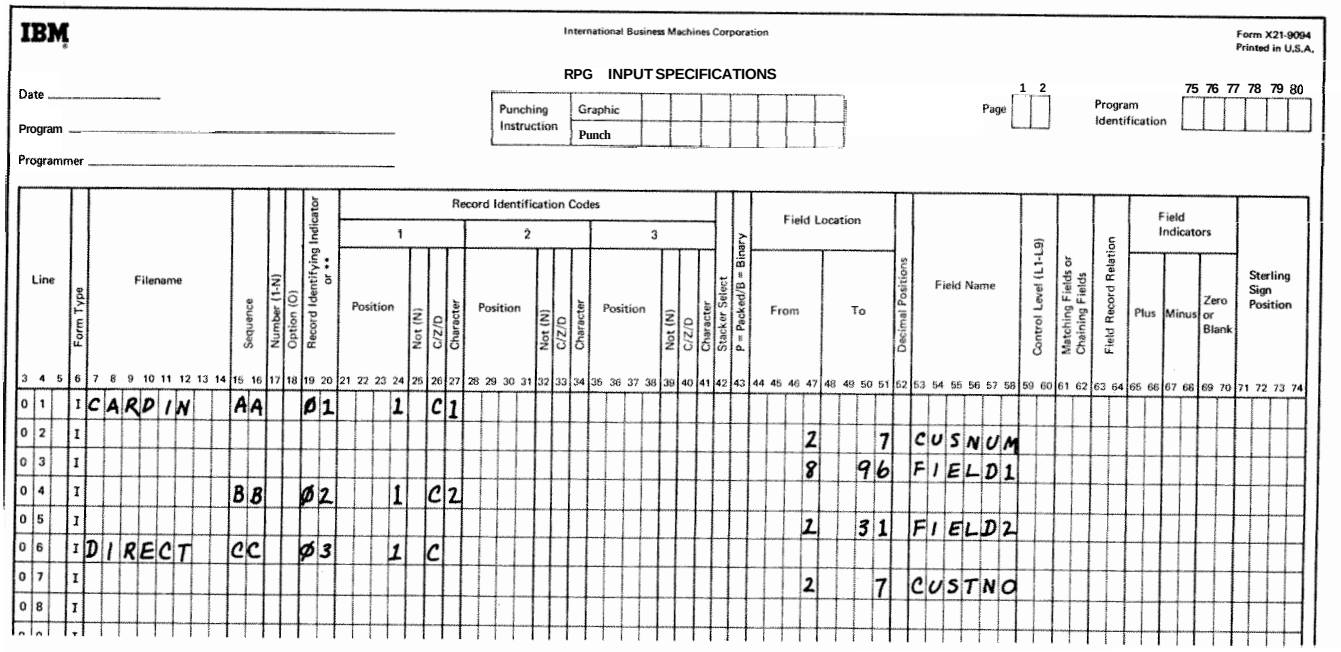
File Description Specifications

Line	Form Type	Filename	I/O/U/C/D	P/S/C/R/T/D	E	A/D	F/V	Block Length	Record Length	L/R	A/K/I	I/D/T or 1-9	Overflow Indicator	Key Field Starting Location	Extension Code E/L	Device	Symbolic Device	Labels (S, N, or El)	Name of Label Exit	Extent Exit for Dam	Cylinder Index in Core	A/U	Number of Tracks for Cylinder Overflow	Number of Extents	N/U	Tape Rewind	File Condition U1-U8
02	F	DIRECT	O	C			F	252	126	R						DISK							Ø1				
03	F	CARDIN	I	P	E		F	96	96							MFCU1											
04	F	PRINT	O				F	132	132							PRINTER											
05	F																										
06	F																										
07	F																										
	F																										
	F																										



Line Counter Specifications

Line	Form Type	Filename	1 Line Number	1 FL or Channel Number	2 Line Number	2 OL or Channel Number	3 Line Number	3 Channel Number	4 Line Number	4 Channel Number	5 Line Number	5 Channel Number	6 Line Number	6 Channel Number	7 Line Number	7 Channel Number	8 Line Number	8 Channel Number	9 Line Number	9 Channel Number	10 Line Number	10 Channel Number	11 Line Number	11 Channel Number	12 Line Number	12 Channel Number
11	L	PRINT		FL		OL																				
12	L																									
13	L																									



IBM

International Business Machines Corporation

Form X21-9094
Printed in U.S.A.

RPG INPUT SPECIFICATIONS

Date _____

Program _____

Programmer _____

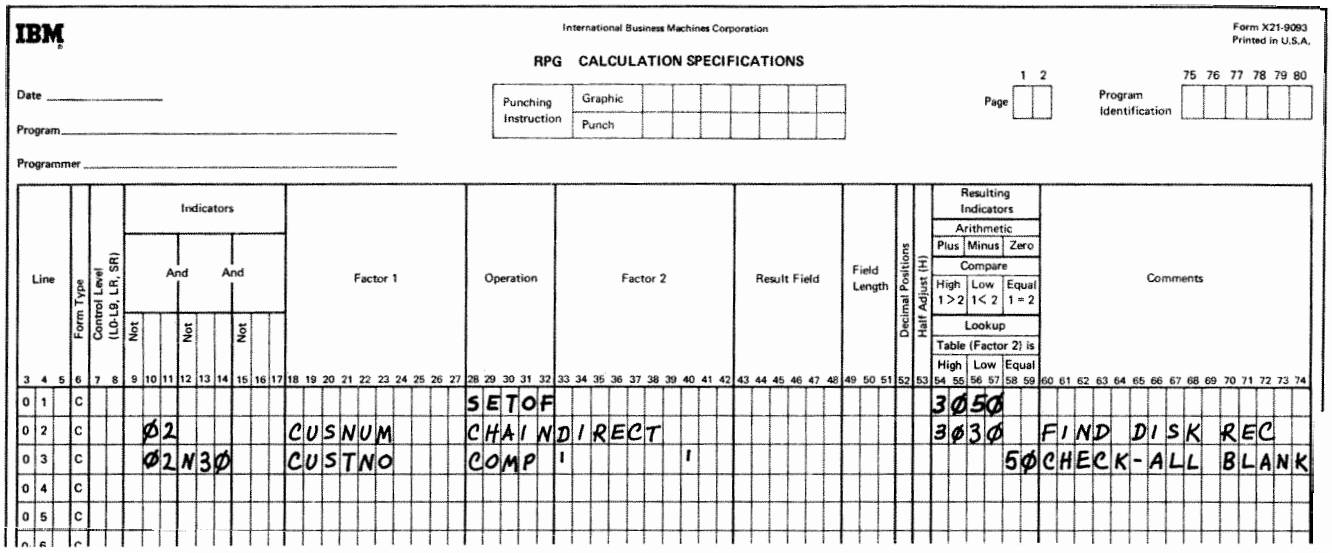
Punching Instruction	Graphic								
	Punch								

Page 1 2

Program Identification 75 76 77 78 79 80

Line	Form Type	Filename	Sequence	Number (1-N)	Option (O)	Record Identifying Indicator or **	Record Identification Codes 1 Position	1 Not (N)	1 C/Z/D	1 Character	2 Position	2 Not (N)	2 C/Z/D	2 Character	3 Position	3 Not (N)	3 C/Z/D	3 Character	Stacker Select	P = Packed/B = Binary	Field Location From	Field Location To	Decimal Positions	Field Name	Control Level (L1-L9)	Matching Fields or Chaining Fields	Field Record Relation	Field Indicators Plus	Field Indicators Minus	Field Indicators Zero or Blank	Sterling Sign Position
01	I	CARDIN	AA			Ø1		1	C	1																					
02	I																				2	7		CUSNUM							
03	I																				8	96		FIELD1							
04	I		BB			Ø2		1	C	2																					
05	I																				1	31		FIELD2							
06	I	DIRECT	CC			Ø3		1	C																						
07	I																				2	7		CUSTNO							
08	I																														

Figure 10-52. Loading a Direct File **(Part 1 of 2)**



IBM

International Business Machines Corporation

Form X21-9093
Printed in U.S.A.

RPG CALCULATION SPECIFICATIONS

Date _____

Program _____

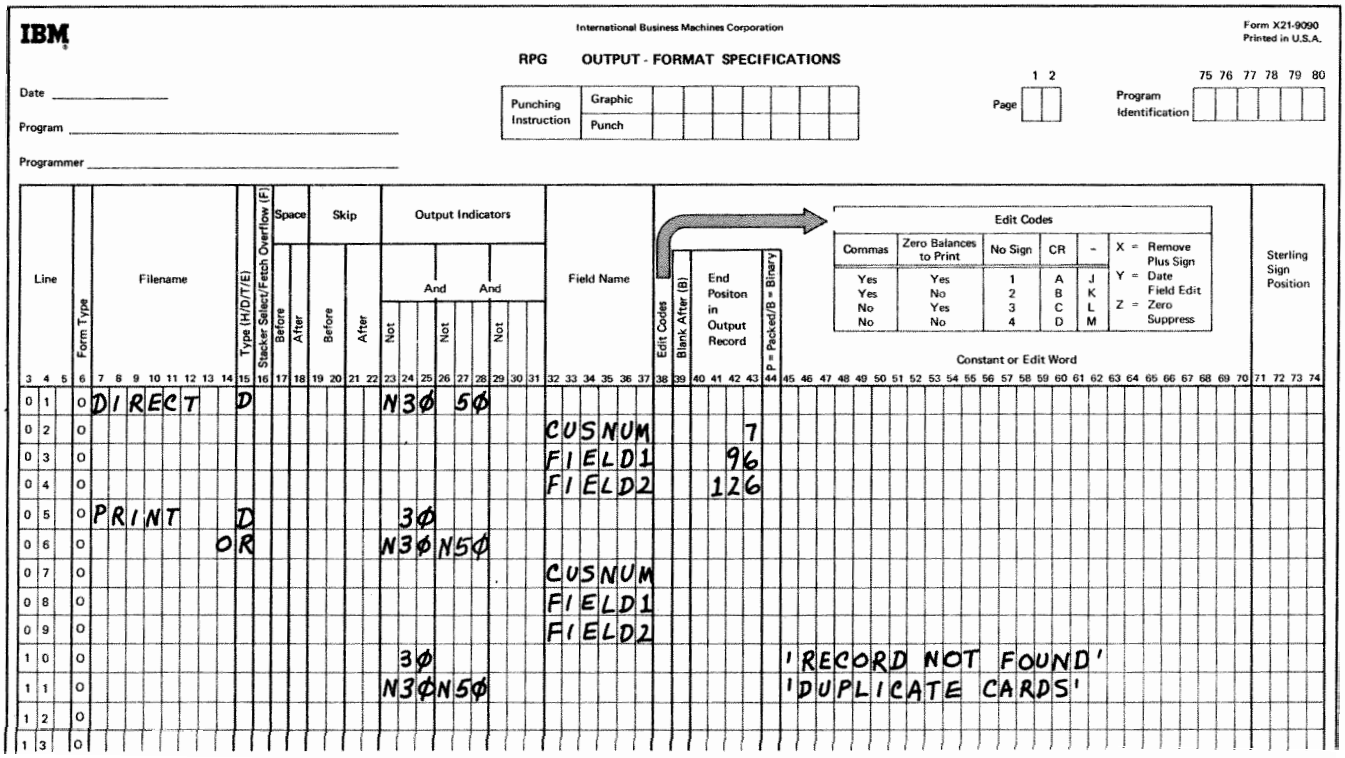
Programmer _____

Punching Instruction: Graphic / Punch

Page 1 2

Program Identification 75 76 77 78 79 80

Line	Form Type	Control Level (LO-L9, LR, SR)	Indicators: And / Not	Factor 1	Operation	Factor 2	Result Field	Field Length	Decimal Positions	Half Adjust (H)	Resulting Indicators: Arithmetic (Plus / Minus / Zero); Compare (High 1>2 / Low 1<2 / Equal 1=2); Lookup; Table (Factor 2) is (High / Low / Equal)	Comments
0 1	C				SETOF						30 50	
0 2	C		02	CUSNUM	CHAIN DIRECT						30 30	FIND DISK REC
0 3	C		02N30	CUSTNO	COMP	1 1					50	CHECK-ALL BLANK
0 4	C											
0 5	C											
0 6	C											



IBM

International Business Machines Corporation

Form X21-9090
Printed in U.S.A.

RPG OUTPUT - FORMAT SPECIFICATIONS

Date _____

Program _____

Programmer _____

Punching Instruction: Graphic / Punch

Page 1 2

Program Identification 75 76 77 78 79 80

Edit Codes

Commas	Zero Balances to Print	No Sign	CR	-	
Yes	Yes	1	A	J	X = Remove Plus Sign
Yes	No	2	B	K	Y = Date Field Edit
No	Yes	3	C	L	Z = Zero Suppress
No	No	4	D	M	

Line	Form Type	Filename	Type (H/D/T/E)	Stacker Select/Fetch Overflow (F)	Space: Before / After	Skip: Before / After	Output Indicators: And / Not	Field Name	Edit Codes	Blank After (B)	End Position in Output Record	P = Packed/B = Binary	Constant or Edit Word	Sterling Sign Position
0 1	O	DIRECT	D				N30 50							
0 2	O							CUSNUM			7			
0 3	O							FIELD1			96			
0 4	O							FIELD2			126			
0 5	O	PRINT	D				30							
0 6	O	OR					N30N50							
0 7	O							CUSNUM						
0 8	O							FIELD1						
0 9	O							FIELD2						
1 0	O						30						'RECORD NOT FOUND'	
1 1	O						N30N50						'DUPLICATE CARDS'	
1 2	O													
1 3	O													

F i r e 10-52. Loading a Direct File (Part 2 of 2)

On the Input sheet there are two record types identified for CARDIN. This is because our disk record is 126 positions long, and two different cards are needed to contain 126 positions of data. The cards each have a code in position 1. The first card is filled with data using positions 2-96; the second card has data in positions 231. Together these data positions total 125 (position 1 of the disk record will be left blank). During the proper output cycle, these two card fields, FIELD 1 and FIELD 2, are written in the appropriate record positions on disk.

Three possibilities will be considered in this program:

1. The CHAIN operation is successful, and the data is written from card to disk

2 The record number in the card has no corresponding record position on disk.

3. The proper record position on disk is found, but already contains data, indicating duplicate input cards.

In the first case, assume that the two input cards are read in for customer 154 (000154 in the card) turning on indicator 0 2 The card field containing the customer number is named CUSNUM. On the Calculation sheet, 02 causes a chain operation using the card field CUSNUM (154) to locate record position 154 in the disk file DIRECT. Assume that the proper record position is found, so 02 must be on and 30 must be off. On 02 and N30 CUSTNO on disk is compared to blanks. CUSTNO is the disk field that will contain the customer number. Remember that this disk field is still all blank at this point. (For a direct load, you may use input specifications to name fields in the file being loaded.)

The compare operation is checking to make sure that CUSTNO is equal to all blanks; that is, checking to see if any data has already been written in CUSTNO. If CUSTNO is all blank indicator 50 turns on, and you can assume the whole disk record is blank and proceed to write the card fields onto disk.

On the Output sheet, indicators N30 (indicating the disk record was found) and 50 (indicating the disk record was blank) are used to condition the loading of the card fields onto disk. The disk record will look like this after the load:

1. disk position 1 is left blank for future use.

2 disk positions 2-7 contain the 6 position customer number from CUSNUM in the input card (in this case 000154).

I disk positions 8-96 contain the data from FIELD1 in the input card

4. disk positions 97-126 contain the data FIELD2 in the input card.

In the second case mentioned, the disk record position is not found. The card field CUSNUM might contain a number for which there is no corresponding record position in the disk file DIRECT. For example, suppose that a card is read in with customer number 260 (000260 in the card), If such a card is processed, this error will be found in calculations; the chain operation will not be successful because the file is only 250 records long (indicator 30 will turn on). On the Output sheet, indicator 30 is used to print out the data from the cards with the error. Also a constant 'RECORD NOT FOUND' is conditioned by indicator 30 to identify the reason for printing out the input cards.

In the third case, the proper disk record is found in the chain operation. However, the disk field CUSTNO is checked for blanks with a compare operation, and CUSTNO has data written in it (it is not equal to blank). Such a case would occur if the cards for customer 154 were read in again, indicating duplicate cards in the card file. If the cards for customer 154 were processed again, CUSTNO would not be blank and indicator 50 in calculations would remain off. On the Output sheet, an output line is conditioned by 30 and N50. The indicators 30 and N50 signify that the proper disk record position (154) was found, but that data was occupying the CUSTNO disk field (the data for customer 154 has already been loaded).

The duplicate cards will not be written on disk but the card fields will be printed with an identifying constant, 'DUPLICATE CARDS'. This constant field is used to distinguish the duplicate card fields from the card fields printed with the constant 'RECORD NOT FOUND'.

Debug Operation

The debug operation is an RPG II function that you may use to help you find errors in a program which is not working properly. This code causes either one or two records to be printed. They contain information which is helpful for finding programming errors,

Debug (DEBUG)

The DEBUG operation code may be placed at any point or at several points in the calculation operations. Whenever it is encountered, either one or two records are printed depending upon the specifications entered. One record contains a list of all indicators which are on at the time the DEBUG code was encountered. The other shows the contents of any one field.

Specifications

Factor 1 is optional: It may contain a literal of 1-8 characters which will identify the particular debug operation. The name entered here is printed on record 1. Factor 2 must contain the name of the output file on which the records are written. The same output filename must appear in Factor 2 for all DEBUG statements in a program. The result field may be a field or array whose contents you wish to appear on record 2. Any valid indicator may be used in columns 7-17. Columns 49-59 must be blank.

The operation code produces results only if the proper entry (1 in column 15) has been made in the control card specifications. If the control card entry has not been made, the operation code DEBUG is treated as a comment. See ***Column*** 15 in Chapter 3 for more information.

Records Printed for Debug

Record 1 is required. It is printed in the following format:

Print Positions	*Information*
1-8	DEBUG=
9-16	Constant entered in Factor 1 (optional)
17-18	Blank
19-34	The words INDICATORS ON=
35-37	The names of all indicators which are on, each separated by a blank.

Record 2 is optional and is printed only when there is a result field. The record is printed in the following format:

Print Positions	*Information*
1-14	The words FIELD VALUE=
15-any position (depending on length of field)	The contents of the result field (up to 256 characters). If the result field is an array, more than one record may be needed to contain the array.

OVERFLOW INDICATORS

When the printer has reached the end of a printed page, RPG II language allows you to do one of three things:

1. Advance to the top of the next page and continue printing.
2. Ignore the fact that the end of the page has been reached and keep right on printing.
3. Print special lines at the bottom of the page and at the top of the new page.

You automatically get the first option by doing nothing. You get the second by assigning an overflow indicator and never using it to condition output lines. You get the third by assigning and using overflow indicators. These three possibilities are described as follows.

1. For every job you do you must determine how many lines will be printed on each page or form. You indicate this by line counter specifications. From these specifications RPG II determines which line is the overflow line. The overflow area is from the line associated with overflow to the end of the form.

   RPG II language is set up so that when the overflow line is sensed, an overflow indicator automatically turns on, and at the appropriate point in the program cycle, the following steps occur:

   a. Detail lines are printed (if this part of the program cycle has not already been completed).
   b. Total lines are printed,
   c. Forms advance to a new page.
   d. The overflow indicator turns off.

Thus you can print detail and total output lines without worrying about what will happen at the end of the page. RPG II takes care of that automatically. All you have to do is set up the correct line counter specifications.

2 If you are not concerned about pages or skipping to new pages and want one continuous listing, you must make an entry that will cause the automatic handling of overflow and advancing of forms to be discontinued. Merely assign an overflow indicator to the printer file in columns **33-34** of a file description specification line. This one entry causes overflow to be ignored. Pages are not taken into consideration.

3. If you are concerned about pages and want certain lines to appear on each page, you first have to assign a certain overflow indicator to the printer file. This is specified in columns **33-34** of a file description specification line (Figure 10-53). Then use this same indicator to condition those lines which you want printed on every page. Usually these lines are total lines which must be printed at the bottom of every page, or heading lines which must be printed at the top of each new page.

Now forms will not automatically advance to a new page. You have to specify a skip to the first printing line on a new page. This skip is usually specified on the first heading line you want printed on the new page (Figure 10-54).

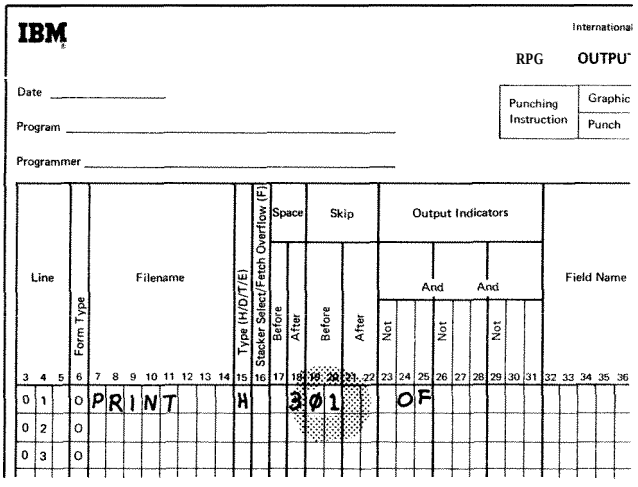


Figure 10-54. Advance Forms to New Page

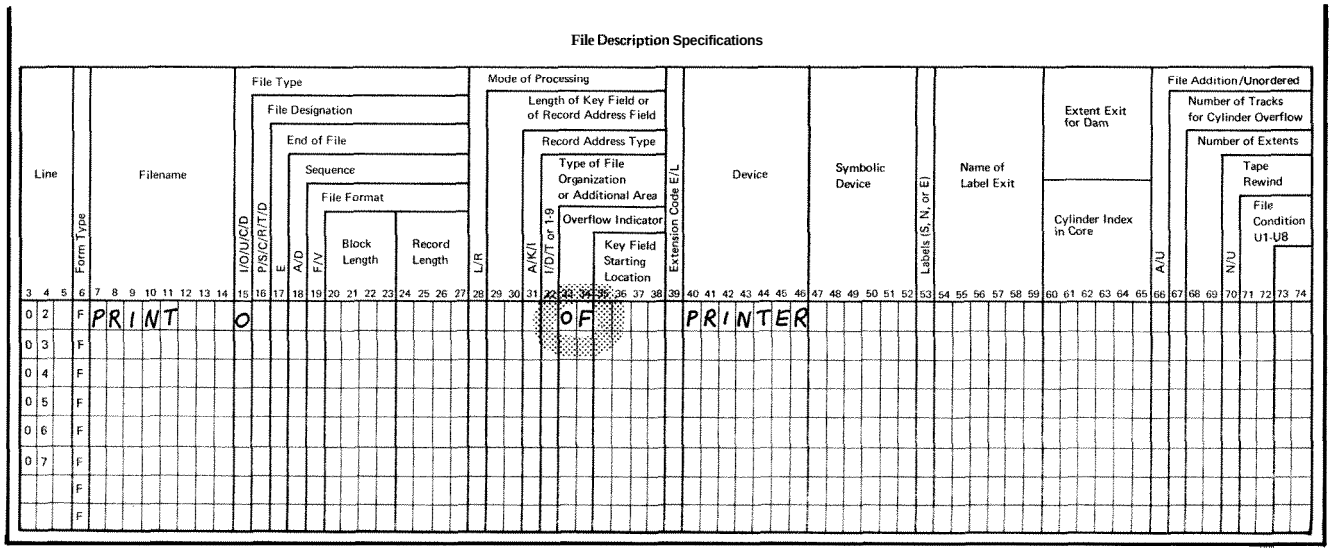


Figure 10-53. Assigning an Overflow Indicator

In the case where you have specified an overflow indicator and are using it to condition output lines, the following steps occur when the overflow line (end of page) has been sensed:

a Detail lines are printed (if that part of the program cycle has not already been completed).
b. Total lines are printed.
c. Total overflow lines are printed if conditioned by the overflow indicator.
d Forms advance to the next page if indicated by the skip specification on a heading line or total line.
e. Headings and detail lines are printed, if conditioned by overflow indicators.

Writing Specifications Using Overflow Indicators

Often you want each page to contain information from only one control group. (Information from one group may require several printed pages, however.) You might also wish each page to have headings identifying the type of information on the page. For these cases you need to use both the control level indicators and the overflow indicators. Together they condition when headings and/or group information are to be printed.

A new page should advance either when the overflow line has been reached (the overflow indicator you assigned is on) or when there is a change in a control field (L indicator is on). You must specify that each indicator causes a new page to be advanced by specifying a skip to the first printing line on a page. If the control level has changed and the overflow condition has occurred at the same time, it is possible to duplicate an output line (one called for by the overflow indicator, the other by the control level indicator). A blank page can also appear in your report as a result.

Figure 10-55 shows the coding necessary for printing headings on every page: fust page, every overflow page, and each new page to be started because of a change in control fields (L2 is on). Line 01 allows the headings to be printed at the top of a new page (skip to 01) only when an overflow occurs (OV is on and L2 is not on).

Line 02 allows printing of headings on the new page only at the beginning of a new control group (L2 is on). This way, duplicate headings caused by both L2 and OV being on at the same time do not occur. Line 02 allows headings to be printed on the first page after the first record is read. This is true because the first record always causes a control break (L2 turns on), if control fields are specified on the record. (If the first record did not have a control field, another OR line would be necessary with a 1P entry in columns 24-25.)

Figure 10-56 shows the necessary coding for the printing of certain fields on every page: a skip to 01 (first line on new page) is done either on an overflow condition or on a change in control level (L2). The NL2 indicator in line 01 prevents the line from printing and skipping twice in the same cycle.

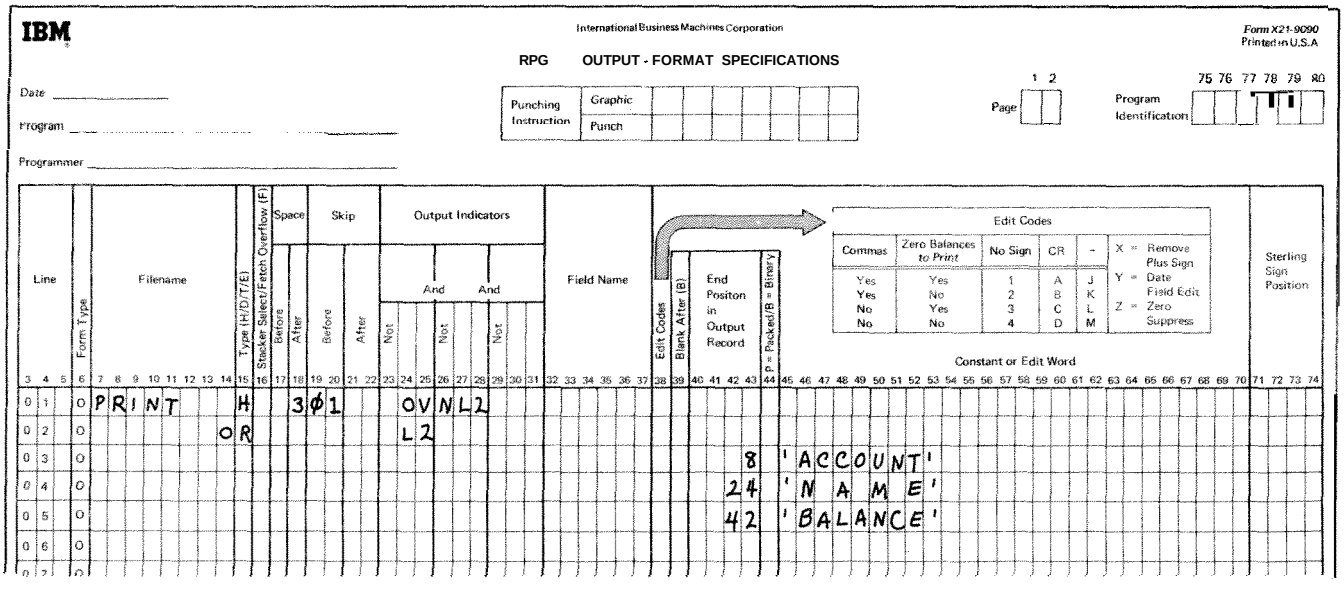


Figure 10-55. Printing Headings on **Every Page**

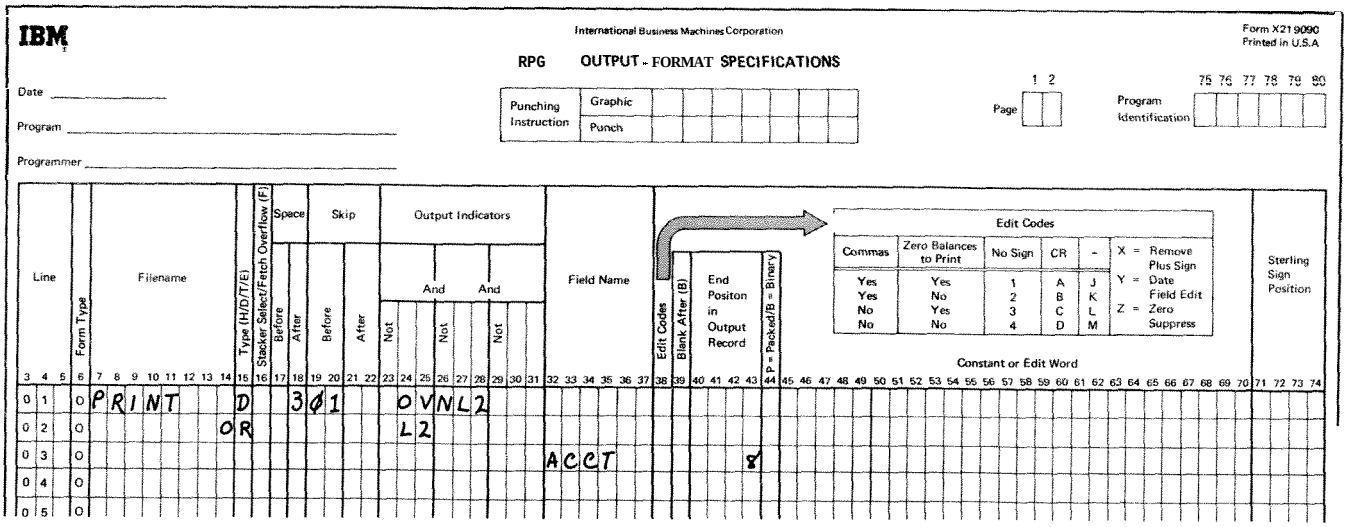


Figure 10-56. Printing Fields on Every Page

Fetching The Overflow Routine

When the overflow line is reached, the same sequence of events always takes place. These were described previously. Briefly, remaining detail lines, total lines, and total overflow lines (lines conditioned by the overflow indicator) are printed on the page even after overflow has occurred. Therefore, you must leave enough room between the overflow line and the actual end of page to have room for all these lines to print.

However, you can run into problems when you do this. For example, if a different number of detail or total lines can be printed each time, you may not have allowed enough room between the overflow line and the end of page to take care of all total lines which will print before the forms advance. Therefore, printing is done on the perforation. You may also have to allow so much room between the overflow line and the end of page that often only half a page is used

To take care of these problems, you may call for the printing of overflow lines and a forms advance any time after the overflow line has been reached Causing overflow lines to be printed ahead of the usual time is known as fetching overflow. When overflow is caused in this way, the following events occur:

1. All total lines conditioned by the overflow indicator are printed.
2. Forms advance to new page when a skip to 01 has been specified in a line conditioned on an overflow indicator.
3. Heading lines conditioned by the overflow indicator are printed.
4. The line that fetched overflow is printed.
5. Any detail and/or total lines left to be printed for that program cycle are printed.

For the printer file, an F in column 16 on the Output-Format sheet specifies that the overflow routine will be fetched. An F can be specified for any total, detail line, or exception line except those conditioned by an overflow indicator.

Figure 10-57 shows the use of a fetched overflow routine (F in column 16). The total lines 03, 09, and 11 can fetch the overflow routine. They do this, however, only if the overflow line has been sensed prior to the printing of one of these lines. If the overflow indicator is turned on before the output line specified in line 03 is printed and if control level indicator L1 is on, forms advance to the new page as specified by the skip entry in the heading line. The heading line and all total lines are printed on the new page. If, however, the printing of the line specified in 03 caused the overflow indicator to turn on, the following happens:

1. The line specified in 05 prints on the same page.
2. The line specified in 07 prints on the same page.
3. The line specified in 09 fetches an overflow (*F* in column 16) and causes the heading line and all total lines (09, 11, 13, and 15) to print on the new page.

If the output lines specified in 09 fetched overflow, line 13 does not fetch a new page again since the overflow indicator is turned off after line 09 fetched overflow. (Remember, a line can fetch overflow only when the overflow indicator is on.) Line 11 fetches overflow only if the output line specified in 09 causes the overflow indicator to turn on.

You should fetch the overflow routine (F in column 16) only when you feel that (1) this line, when printed, could cause overflow and (2) if it did, there would not be enough room left on the page to print the remaining detail and/or total output lines plus lines conditioned by the overflow indicator.

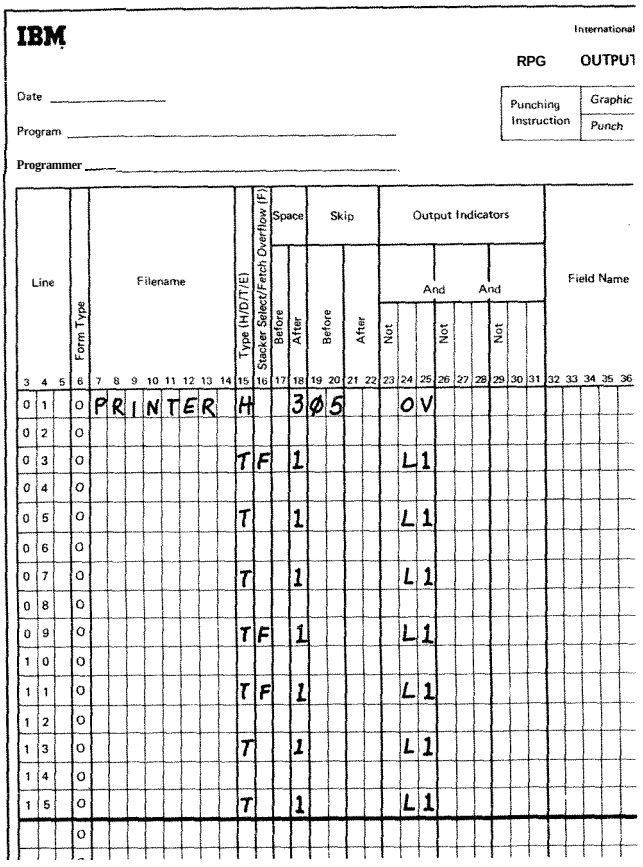


Figure 10-57. Uses of Fetch

Overflow Printing with EXCPT Operation Code

Overflow indicators cannot condition an exception line, but can condition fields within an exception record. The use of the EXCPT operation code with the E in column 15 of the Output-Format sheet causes the fields to be printed during the time calculations are being performed (normally they are printed afterwards). Only the specified fields (identified by an *E* in column 15) are printed at that time. Even though these fields are not printed at the usual time, they still have the same effect on the overflow routine as all other lines. If the overflow line is sensed when an exception field is printed, the overflow indicator turns on as usual.

General Considerations

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

1. Overflow indicators may be turned on and off by the operation codes SETON and SETOF.
2. Spacing past the overflow line causes the overflow indicator to turn on.
3. Skipping past the overflow line to any line on the new page does not turn the overflow indicator on.
4. A skip to a new page specified on a line not conditioned by an overflow indicator causes the overflow indicator to turn off.

Figure 10-58 shows the setting of overflow indicators during the normal overflow routine and during a fetched overflow routine for both normal output and exception output. The left-hand portion of the graph shows when the indicators are on or off in relation to the general program cycle. For example, if, during normal output, a detail line is printed on the line number specified as the overflow line, the overflow indicator turns on. It remains on until the end of the next program cycle. The solid blank lines indicate that the indicator is on. The dashes are used to show a connection between the end of one cycle and the start of the next.

PROGRAM CYCLE

For each record that is processed, the RPG II 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 are written out. First, all calculation operations conditioned by control level indicators (columns 7-8) and all total output operations are done. Second, all calculation operations not conditioned by control level indicators (columns 7-8) and all detail output operations are done.

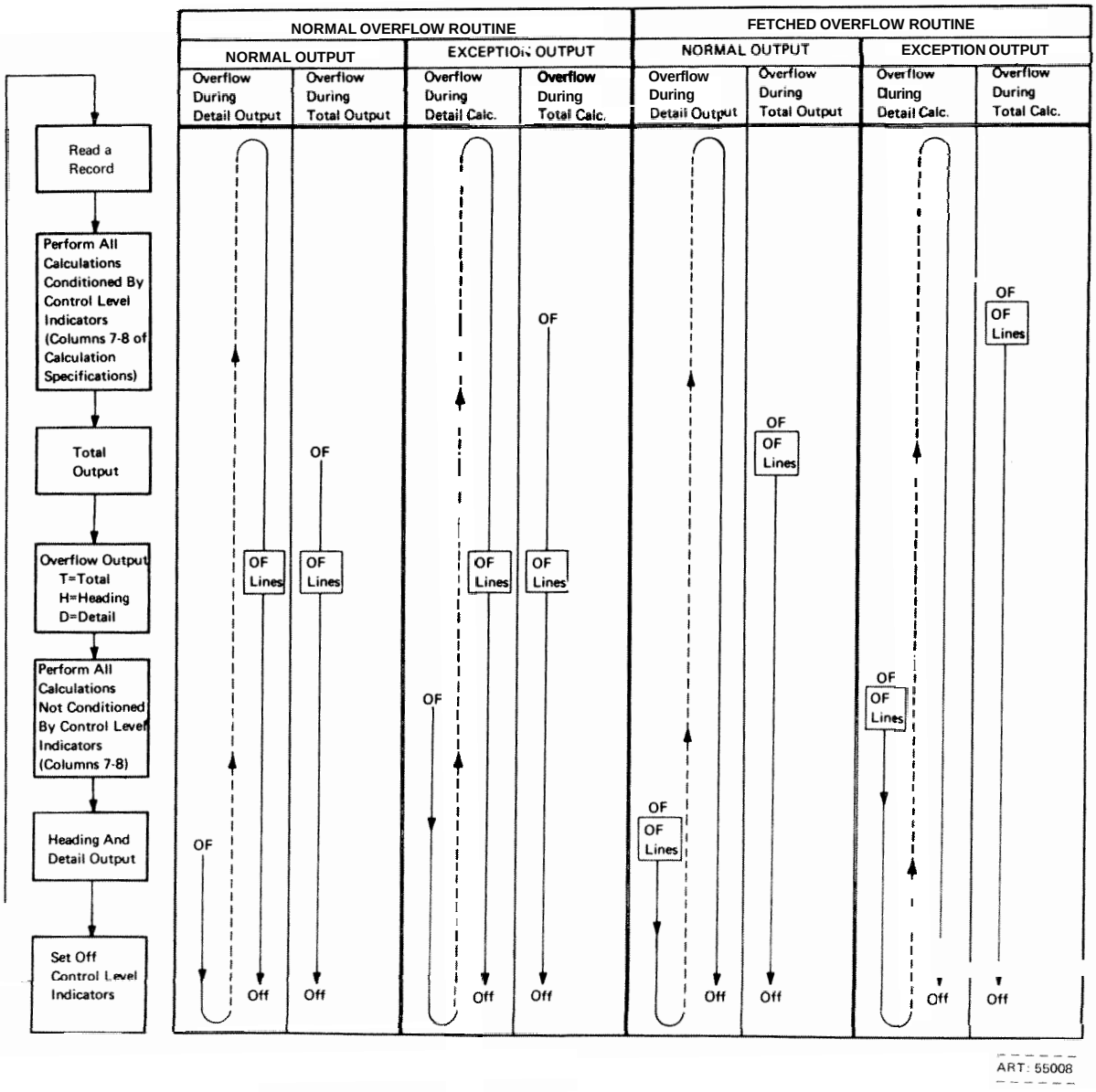


Figure 10-58. Overflow Printing: Setting of the Overflow Indicator

The first instance in time that calculation operations are performed, they are performed on information from records read prior to the record just read. The second instance in time that calculations are performed, they are performed on data from the record just read. 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 record. 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 (shown by control level indicators being turned on), operations may be done using information accumulated from all records in that group. It is at this time then that all calculations conditioned by control level indicators in columns 7-8 are done. Total output operations are also performed at this time. Remember that information on the record read at the beginning of the program cycle is not used in these operations; only information from cards in the previous control group is used.

Calculations are done at the second instance in time when one of the following occurs:

1. All total calculation and total output operations have been completed.

2. The information in the control field has not changed (no total operations have been done).

3. No control fields have been specified (no total operations will be done).

When any one of these three conditions occurs, all calculation operations not conditioned by control level indicators (columns 7-8) are performed. Detail output operations are also performed at this time.

The specific steps taken in one program cycle are shown in Figure 10-59. The item numbers in the following description refer to the numbers in the figure. A program cycle begins with step **1** and continues through step **21.**

1. The object program performs all specified heading and detail output operations whose conditions are satisfied. This does not include specifications that are conditioned by the overflow indicator. The object program also performs a test to determine if the overflow line was encountered when heading and detail records were written. If it was, the overflow indicator turns on. Otherwise, the overflow indicator turns off.

2. The program tests to see if the LR indicator is on. If it is, the program branches to step 14.

3. The object program tests the halt indicators. If the halt indicators are off, the program branches to step 5.

4. The execution of the program is stopped if the halt indicators are on. Execution may be restarted, however, by pressing the start key on the Processing Unit.

5. All record identifying indicators and indicators 1P, LR, L1-L9, H1-H9 are turned off.

6. The program reads the next input record. At the beginning of processing, if it is a multifile job, one input record from each file is read.

7. The program performs a test to determine if the record is an end-of-file record. If an end-of-file condition has occurred, the program branches to step **22.**

8. If end of file has not occurred, the program performs a test to determine if the input records are in the sequence specified for them on the Input sheet. If the sequence is incorrect, the program branches to step **30.** The program also branches to step **30** if nonsequential input records are specified and the record cannot be identified.

9. The object program branches to step **25** if matching fields are specified.

10. The record identifying indicator specified for the current record type turns on. Data from the current record type is not available for processing until step **20.**

11. The object program performs a test to determine if a control break has occurred (the contents of this control field are not equal to the contents of a previously stored control field). If a control break has not occurred, the program branches to step 13.

12. If a control break has occurred, the appropriate control level indicators turn on.

13. If this is the fust program cycle, the program bypasses the calculations and the output specifications conditioned by control level indicators and branches to step 16.

    If control fields are specified, calculations conditioned by control level indicators and total output lines are bypassed for all records read until the first record

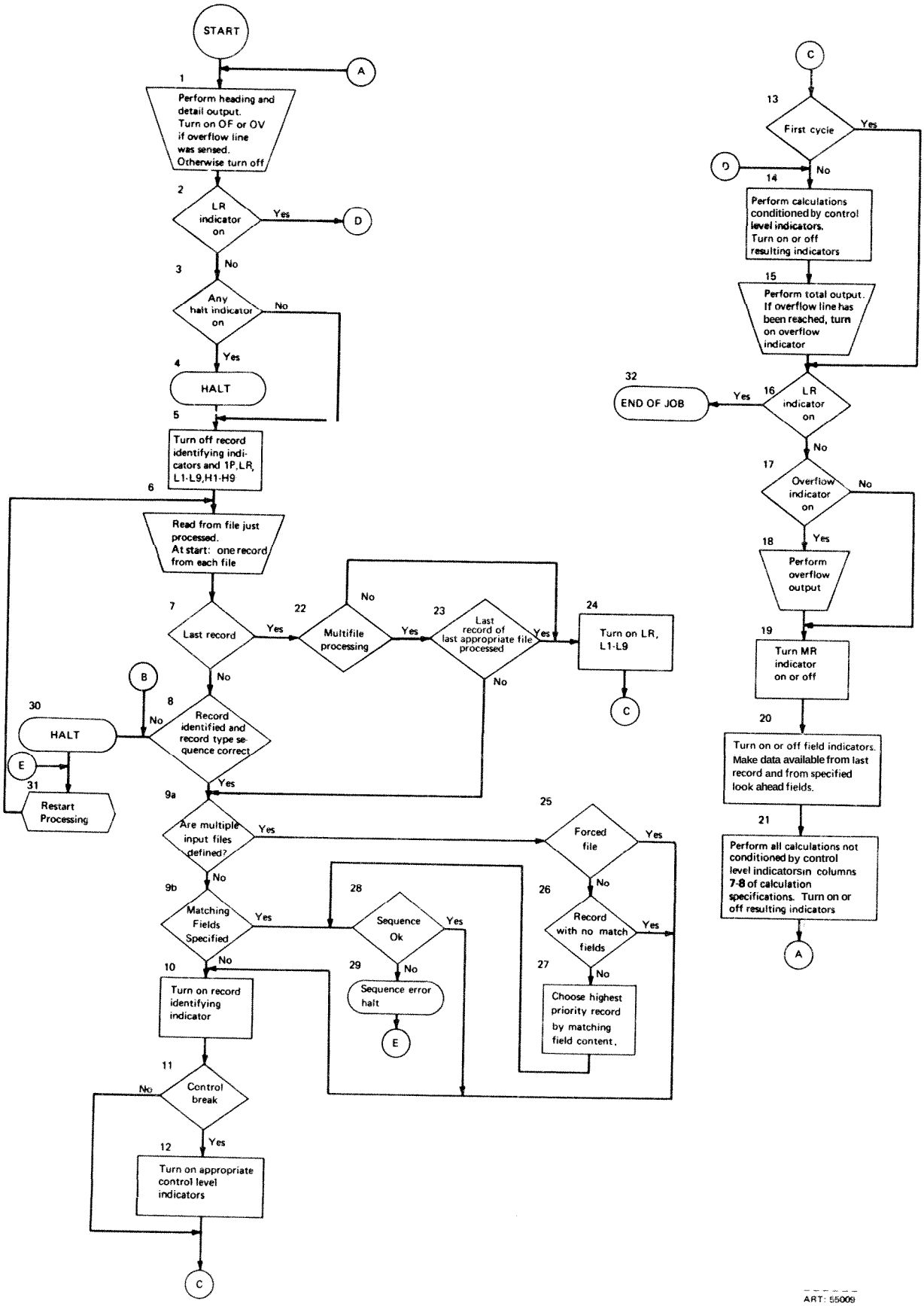


Figure 10-59. Program Cycle

that contains control field information has been processed. This applies also to the calculations specified with an LO indicator.

14. All calculations conditioned by control level indicators (columns 7-8 of calculation specifications) are performed and resulting indicators are turned on or off as specified.

15. All total output that is not conditioned by an overflow indicator is performed. The program performs a test to determine if an overflow condition has occurred. If an overflow condition has occurred at any time during this cycle, the overflow indicator turns on.

16. The program performs a test to determine if the last record indicator (LR) is on. If the indicator is on, the program branches to step 32.

17. The program performs a test to determine if the overflow indicator is on. If no overflow indicator is on, the program branches to step 19.

18. The specified overflow output is performed. If no overflow output is specified, and no overflow indicator is used in file description specifications, the program performs an automatic skip to line 06 of the next page in the printer.

19. The MR indicator turns on if this is a multifile job and the record to be processed is a matching record. Otherwise, the MR indicator turns off.

20. Field indicators are turned on or off as specified. Data from the last record read and from specified look ahead fields is made available for processing.

21. Any calculations not conditioned by control level indicators (columns 7-8 of the calculation specifications) are performed, and resulting indicators are turned on or off as specified. Processing continues with step I.

22. If only one input file is to be processed, the program continues with step 24.

23. The program performs a test to determine if the processing of all the files to be checked for end of Me has been completed. If not, the program branches to step 27.

24. All control level indicators (L1-L9), and last record indicator (LR) are turned on and processing continues with step 13.

25. The program performs a test to determine if the sequence of matching fields is correct. If the sequence is incorrect, the program branches to step 30.

26. The program performs a test to determine if more than one file is to be processed. If only one input file is to be processed, the program branches to step 10.

27. The contents of the matching fields are compared. If the contents are equal, the program branches to step 29.

28. The program determines the contents of the new matching fields.

29. The program determines the next file to be processed and branches to step 10.

30. The execution of the program is discontinued because of a sequence or record type error.

31. Restart processing after eliminating the error condition.

32. End of job occurs.

STERLING

The RPG II language is able to handle British sterling data. The use of sterling data, however, must be indicated to the RPG II compiler. This requires special control card specifications, input specifications, and output-format specifications.

System/3 can process pence data only. Input data, however, may be in pounds, shillings, pence, and pence decimals. RPG II automatically converts the sterling amounts in the input field into pence so that processing can be done. All records are punched or printed in pence unless otherwise indicated by certain specifications.

Since sterling requires the use of special entries in three different types of specifications, each type will be considered separately. A column by column description is used However, only those columns affected by the use of sterling are described. Those that are not described have the same entries as described in the main sections.

CONTROL CARD SPECIFICATIONS (COLUMNS 17-20)

Entry	***Explanation***
0	Records are only printed, not punched.
1	Indicates IBM format.
2	Indicates BSI format.

Use columns 17-20 to indicate the format in which the input data is punched on the card. Two forms are available, IBM or BSI, for data recorded on punched cards. These two formats allow variations in the number of card columns used for shilliig and pence fields. As you read about entries in columns 17-20 refer to Figure 10-60 which shows sterling data punched in various formats.

Sterling Amount : £ : 15 : 10 . 5
(one decimal position, unsigned)

Format	Pounds (£)	Shillings (s)	Pence (d)	Decimals
IBM/IBM	0	15	-	5
IBM/BSI	0	15	&	5
BSI/BSI	0	E	&	5
BSI/IBM	0	E	-	5

ART. 51793

Figure 10-60. Sterling Formats for Punched Output Records

Column 17 (Input Shilling Field)

IBM — Two columns are used in the shilling field The field may contain a number from 00-19.

BSI — One column is used in the shilling field. Because this one column shilling field may contain a maximum value of 19, there must be a way of representing a two digit number in a one column field. The following characters are used to do this:

0-9	0-9 shillings.
&	10 shillings.
A-I	11-19 shillings.

Column 18 (Input Pence Field)

IBM — One column is used in the pence field. The following punches are used to punch pence data into the card:

0-9	0-9 pence.
- (minus)	10 pence.
&	11 pence.

BSI — One column is used in the pence field. The following are used to punch pence data in the BSI format:

0-9	0-9 pence.
&	10 pence.
- (minus)	11 pence.

Column 19 (Output Shilling Field)

See column I 7 for details on formats.

Column 20 (Output Pence Field)

See column 1 7 for details on formats.

When using sterling, remember:

1. It is possible to combine the two formats (see Figure 10-60). For example, the shilling field may be in IBM format and the pence field in BSI format.
2. Sterling fields written on the printer are not in IBM or BSI format. Instead they are always in print format which consists of two shilling positions and two pence positions in addition to a maximum of three decimal positions and nine pound positions.

INPUT SPECIFICATIONS

Columns 1-43

See *Chapter* 4 for information concerning columns 1-43.

Columns 44-51 (Field Location)

Columns 44-51 are used to indicate the location of the sterling field on the card. Entries in these columns are the same for fields containing sterling data as for fields not containing sterling data. Keep in mind, however, that the total length of any sterling field before and after conversion to pence must not be greater than 15 characters. (The RPG II compiler converts all fields to pence.) See *Columns* 44-51 in Chapter 7 for correct entries,

The field length includes pounds, shillings, pence, and decimal positions. The field length must be large enough to include at least one pounds position, but no more than nine. The number of positions in the shilling and pence fields is determined by the type of format used (see *Columns* 17-20 in Chapter 3). Figure 10.48 shows the maximum size of sterling fields for all formats.

Column 52 (Decimal Positions)

Use column 52 to indicate the number of decimal positions in the pence field. The maximum number of positions is three. Therefore, you may enter any number from 0 to 3 in this column.

Columns 53-58 (Field Name)

Use columns 53-58 to name your sterling field. Remember that the same name cannot be used for both a sterling field and a decimal field. See *Columns 53-58* in Chapter 7 for rules on forming field names,

Columns 59-62

Columns 59-62 may not be used with sterling fields. Leave them blank.

Columns 63-70

See *Chapter* 7 for information concerning columns 63-70.

Columns 71-74 (Sterling Sign Position)

Use columns 71-74 to indicate the position of the sign in the sterling field. Normally, when there are decimal positions, the sign is in the rightmost decimal position of the pence field (see *Example* 1). The sign of the field is found in the rightmost character of the pounds field, however, when there are no decimal positions (see *Example* 2).

The sign need not appear in these standard positions. In fact, the sign does not even need to be within the field. However, the sign position, wherever it is, must not only contain a zone entry but also a valid digit entry to ensure that the sign position will be recognized.

Enter an *S* in column 74 when the sign is in the standard position. However, when the sign is not in the standard position, enter the number of the record position (1-4,096) in which the sign is found. The number entered must end in column 74.

Example 1: Figure 10.61, insert **A** shows that the correct position of the sign when decimals are used is in the rightmost decimal position of the pence field. Notice that the minus sign combined with a 5 (the number in the last decimal position) punched out as an *N*.

Example 2: Figure 10.61, insert **B** shows that the correct position of the sign, when decimals are not used, is in the rightmost pound position. Notice that the minus sign, combined with a **1** (number in the rightmost pound position), punches out as a **J**.

OUTPUT SPECIFICATIONS

Columns 1-37

See *Chapter* 9 for information on columns 1-37.

Column 38 (Edit Codes)

The RPG II compiler automatically causes zero suppression of the leftmost digits of the shilling and pence fields. However, if you wish the pounds field to be zero suppressed you must specify editing. A *Z* in column 38 causes the pound portion of the field named in columns 32-37 to be zero suppressed, It also removes the sign of the field before the field is printed.

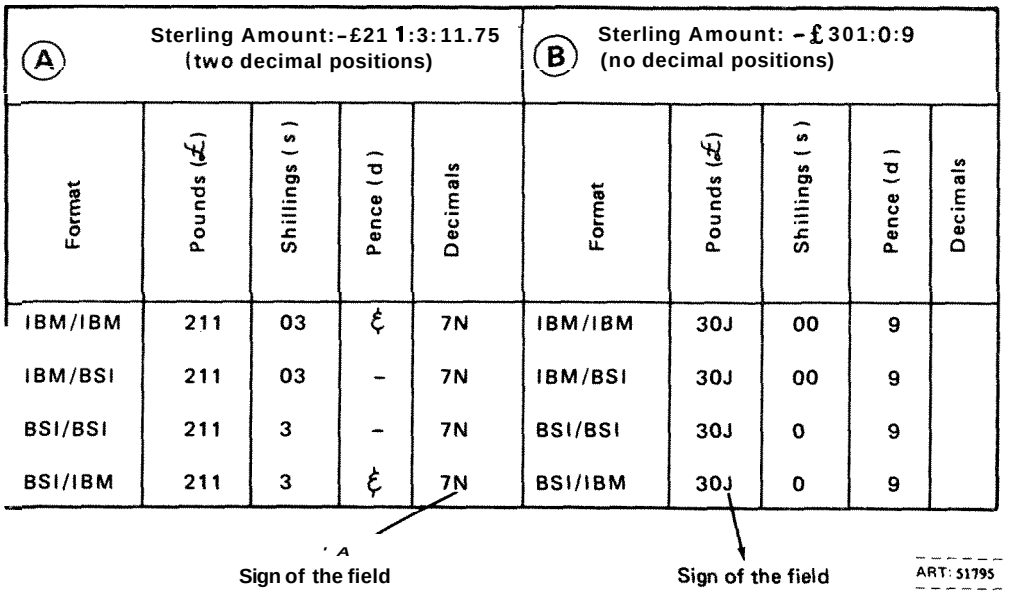


Figure 10-61. Sterling Amounts in All Available Formats

Example: After conversion from pence to pounds, shillings, and pence, the field containing a value of 001040201 (00104 pounds, 02 shillings, and 01 pence) is printed as 1040201 if zero suppression has been specified. If zero suppression has not been specified, the field prints out as 00104 2 1.

Column 39 (Blank After)

See ***Chapter 9*** for further information.

Columns 40-43 (End Position in Output Record)

Use columns 40-43 to indicate the end position of the field on the output record. The formats (IBM or BSI) which were specified on the control card are not used on printed output. Printed output requires two positions for pence, two positions for shillings, from one to nine positions for pounds, and from zero to three positions for decimals. Keep this in mind so that you are sure to allow enough room on the record for the entire field See ***Columns 40-43*** in Chapter ***9*** for correct specifications. For output devices other than the printer, the length required depends on the format used (see ***Columns 40-43*** in Chapter ***3).***

Column 44

Column 44 is not used. Leave it blank.

Columns 45-70 (Constant or Edit Word)

If edit code Z is not used, columns 45-70 may be used to edit an output field. Each edit word used is composed of three sections or fields: the pounds field, the shillings field, and the pence field. When using edit words, you may use:

1. Floating and fixed pound signs.
2. Zero suppression of the pounds field.
3. CR and minus (-) symbols.
4. Asterisk fill.
5. An ampersand to cause a blank in the edit word
6. Any constant information.

When editing sterling fields, remember:

1. An edit word must be enclosed by single quotes.

2. Two positions must be allowed for the shillings field in every edit word. Two positions must be allowed for the pence field.

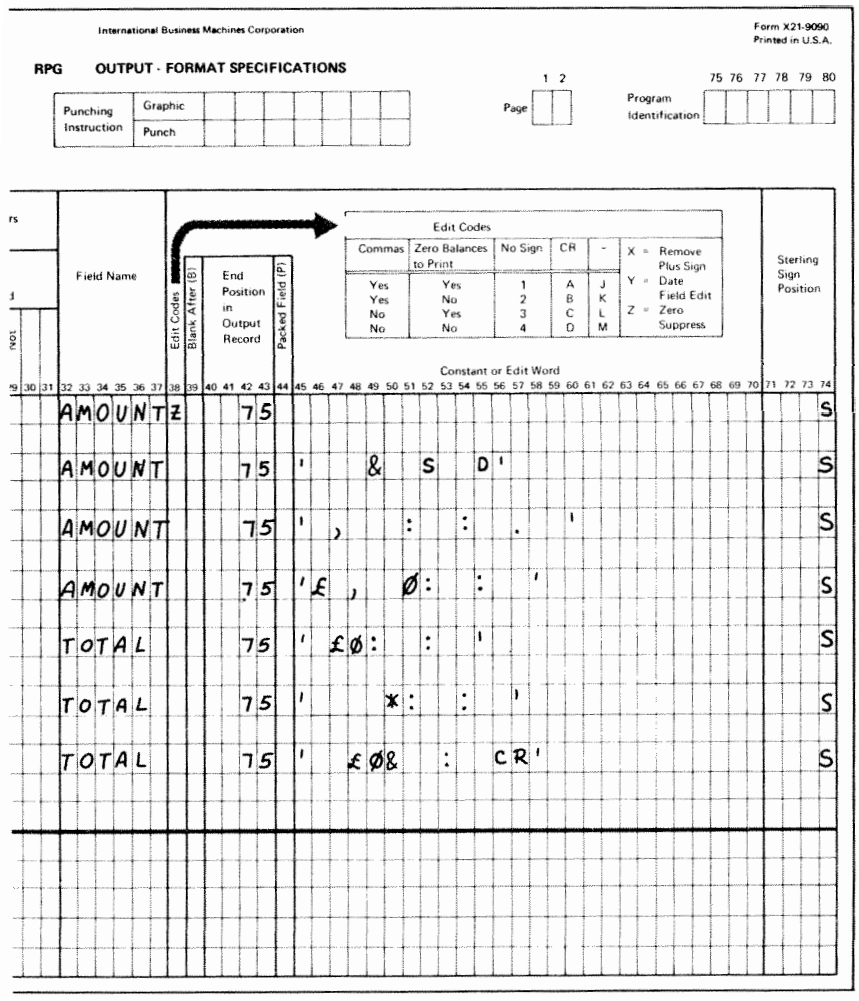
3. At least one character should be inserted between the pounds and shillings fields and the shillings and pence fields in order to separate them. Any character except a blank may be used to separate the shillings and pence fields. A comma, however, is permitted within the pounds field and a decimal point is permitted within the pence field.

4. Zeros in the pounds field may be suppressed by putting a zero suppression zero in the edit word. The shillings and pence fields are always zero suppressed automatically.

5. When specifying the floating pound sign, there must be at least one pound field position preceding the shillings field and following the pound sign.

6. Asterisk fill, if desired, must be specified by placing an asterisk in the pounds field. This causes the pounds field to fdl with asterisks.

Figure 10-62 shows valid examples of editing a sterling field. £ denotes the pound sign. *S* the shilling sign, and d the pence sign. See *Editing* in this chapter for more information on edit words.



Column 75

Unedited Field	Edited Field
00N0703	5 7 3
7551604	755 16S 4D
6754121155	6,754:12:11.55
000M0510	£ 4: 5:10
061406	£6:14: 6
0005N0604	***55: 6: 4
00J1304	£1 13: 4CR
0021304	£2 13: 4

ART: S1796

Figure 10-62. Edit Words for Sterling Fields

Columns 71-74 (Sterling Sign Position)

For printed output records, column 74 must contain an *S* if the pence field is to be converted to pounds, shillings, and pence before it is printed. It may not contain a numeric entry. If blank, the field is printed in pence.

For punched card or disk output, the same entries are used as on the input specifications. An *S* is entered in column 74 when the sign is to appear in the standard position. When the sign is not in the standard position, columns 73-74 must contain the number of the column (1-96) in which the sign is to appear.

SUBROUTINES

A subroutine is a routine that is part of another main routine. A routine is something done over and over again. A program can be called a routine because the instructions in a program are done again and again (the program cycle). A subroutine is a group of instructions in that main routine (program) which may be done several times in one program cycle.

Sometimes it is necessary to write a program which at several points does the same operations. Instead of having to write these instructions every time they are needed, it is easier and less time consuming if they can be written just once and then referred to each time they are needed. You can do this by writing a subroutine which then consists of all those operations you have to do at several points in your program.

You might also have to do the same sequence of operations in several different programs. Instead of writing these specifications in each program, you can code the operations once as a subroutine. You then include this subroutine in as many different programs as you wish.

Coding Subroutines

Subroutines are coded and used on the Calculation sheet. They are entered after all other calculation operations. Every subroutine must have a name. But no two subroutines used in the same program may have the same name.

Enter the name of the subroutine in Factor 1, and on the same line enter the operation code BEGSR (line 10 of Figure 10-63). The subroutine name can be 1-6 characters long and must begin in column 18 with an alphabetic character. The remaining characters can be any combination

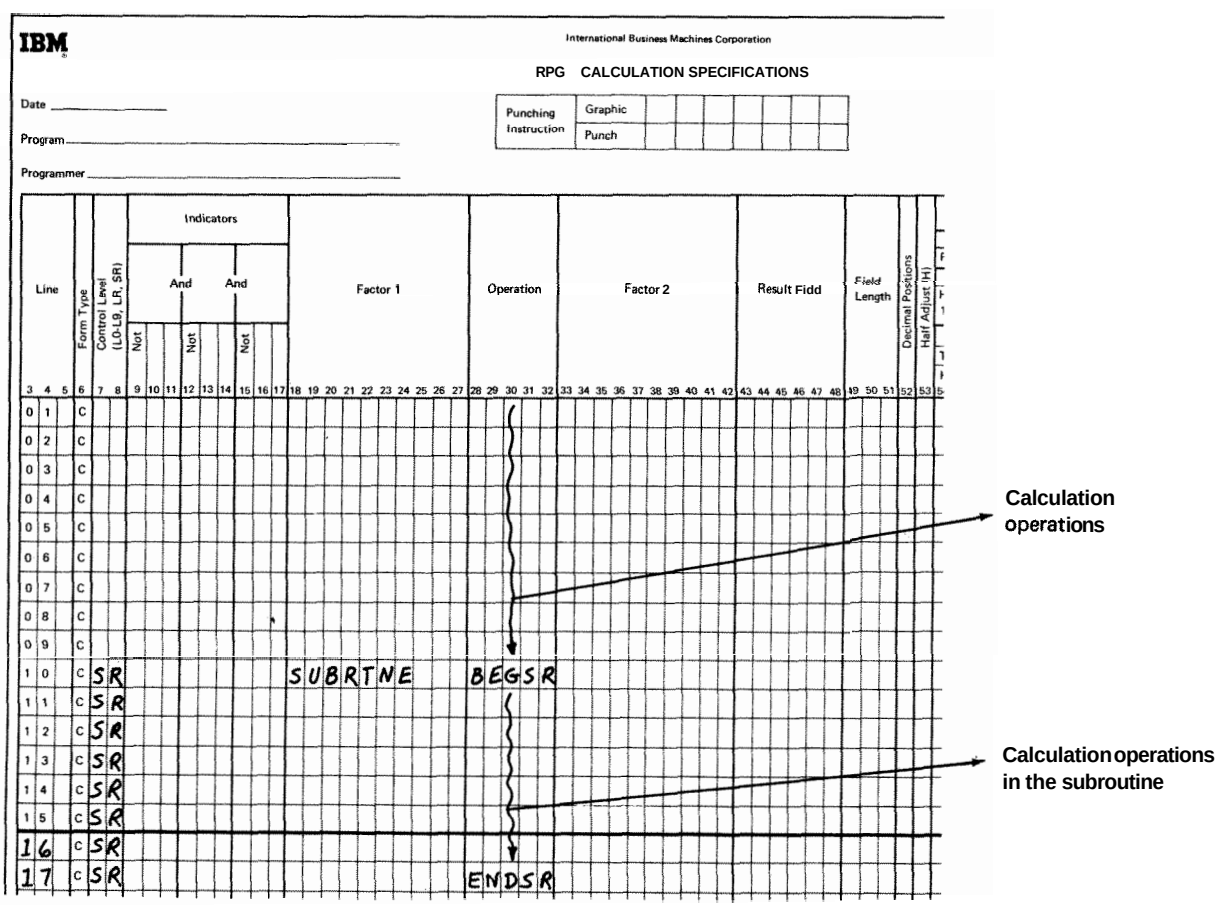


Figure 10-63. Subroutine Lines (SR)

of alphabetic or numeric characters (no special characters). Blanks may not appear between characters in the name.

Each specifications line within the subroutine must have SR in columns 7-8 to identify it as a subroutine line (see Figure 1863). The last statement of the subroutine is indicated by the operation code ENDSR (see line 17 of Figure 1863). Factor 1 of the ENDSR statement may contain a name. This name indicates the point to which a GOTO within the subroutine can branch (Figure 10.64).

The subroutine, even though specified last on the Calculation sheet, may be performed at any point in the calculation operations. Whenever the subroutine is to be used, enter the operation code EXSR (execute subroutine). The name of the subroutine to be used must also be entered in Factor 2 (lines 04 and 08 of Figure 10-65). Using the EXSR operation is known as calling a subroutine.

The operation code EXSR causes the operations in the subroutine named in Factor 2 to be performed. After all calculation operations in the subroutine are done, the next operation after the EXSR is performed. For example, when the EXSR operation (see line 04 of Figure 10.65 is encountered, all subroutine operations (lines 11-15) are done. Then the operation in line 05 is performed.

Indicators may be used with EXSR code to condition when the subroutine should be executed. Any valid indicator may be used in columns 7-17. If no indicators are used, the subroutine is always executed.

All possible RPG II operations may be performed within a subroutine. Operations within the subroutine may be conditioned by any valid indicator in columns 9-17 (see Figure 10.65). Since SR must appear in columns 7-8, control level indicators cannot be used in these columns.

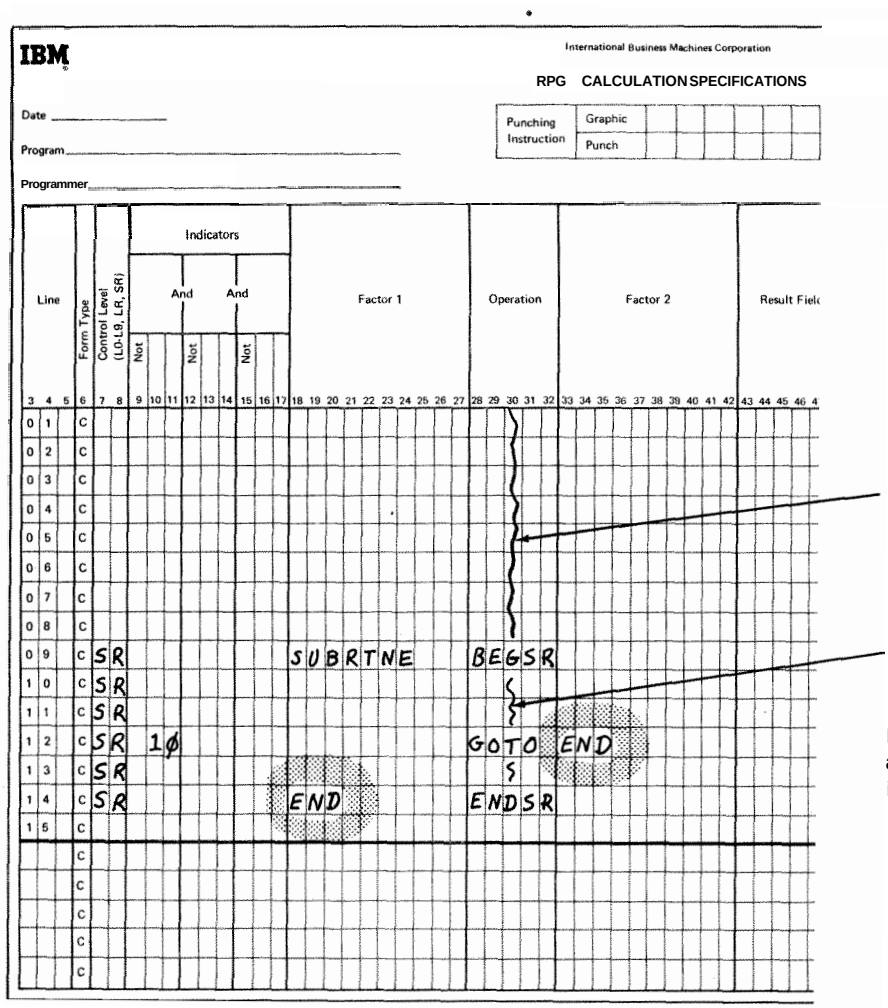


Figure 10-64. Subroutines (ENDSR)

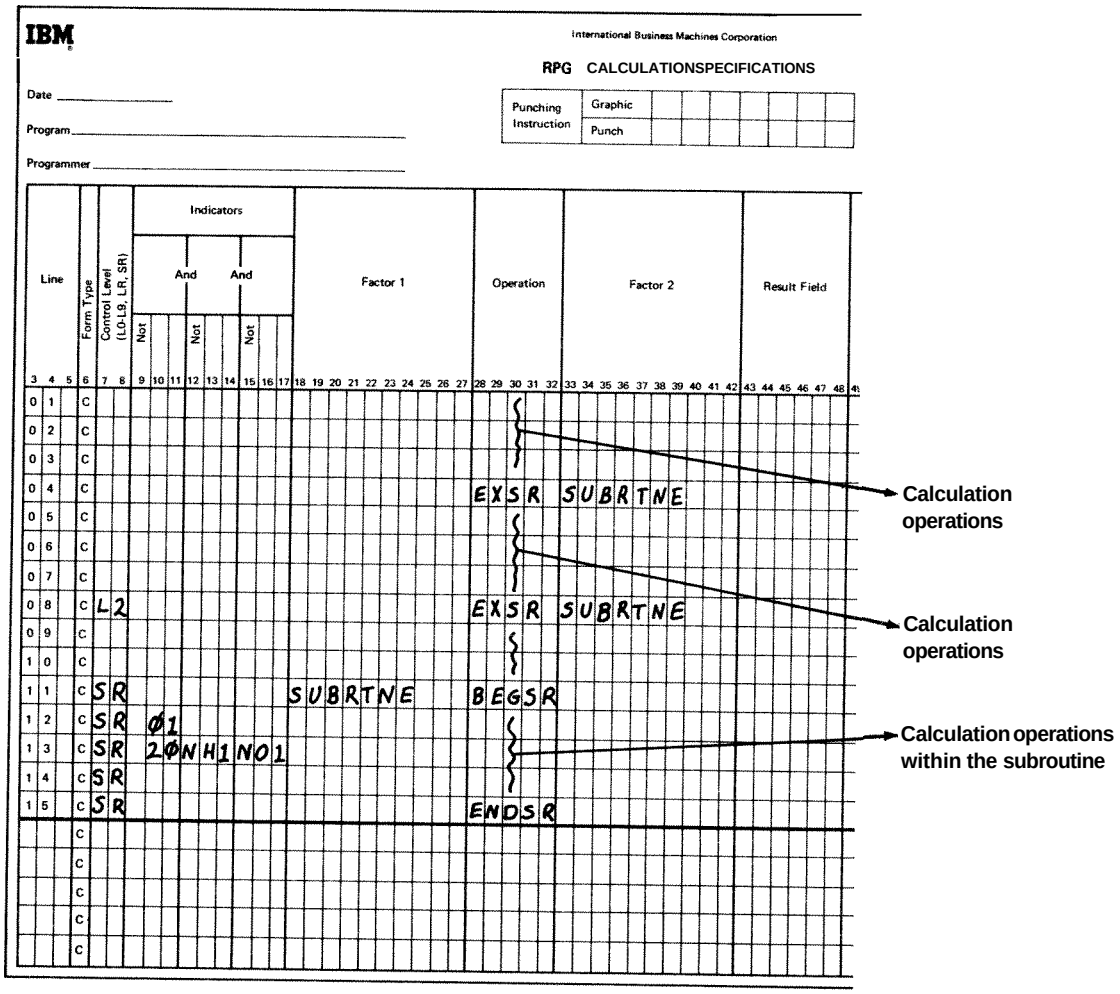


Figure 10-65. Subroutines(EXSR)

This means that individual operations within the subroutine cannot be conditioned by a control level indicator used in columns 7-8. However, entire subroutines can be conditioned by control level indicators. This can be done by using the control level indicator with the EXSR operation (see line 08 of Figure 10.65).

Fields used in the subroutine may be defined either inside or outside the subroutine. In either case, they can be used by both the main routine and the subroutine.

You may use as many subroutines in your main program as you wish. However, you cannot write a subroutine within a subroutine. This means that within one sub routine you cannot have the BEGSR and ENDSR operation codes. One subroutine may call another subroutine, however. In other words, within a subroutine you may have an EXSR operation (Figure 10.66).

Subroutines need not be defined in the order in which they are used. However, you must make certain that each one has a different name and a BEGSR and ENDSR operation code.

When you use a GOTO statement in a subroutine, you may only branch to another statement in that same subroutine. Branching (GOTO) to a statement in another subroutine or outside of a subroutine causes an error condition. You cannot use a GOTO from outside the subroutine to a statement within the subroutine either. Figure 10.67 shows the correct use of GOTO and TAG within a subroutine.

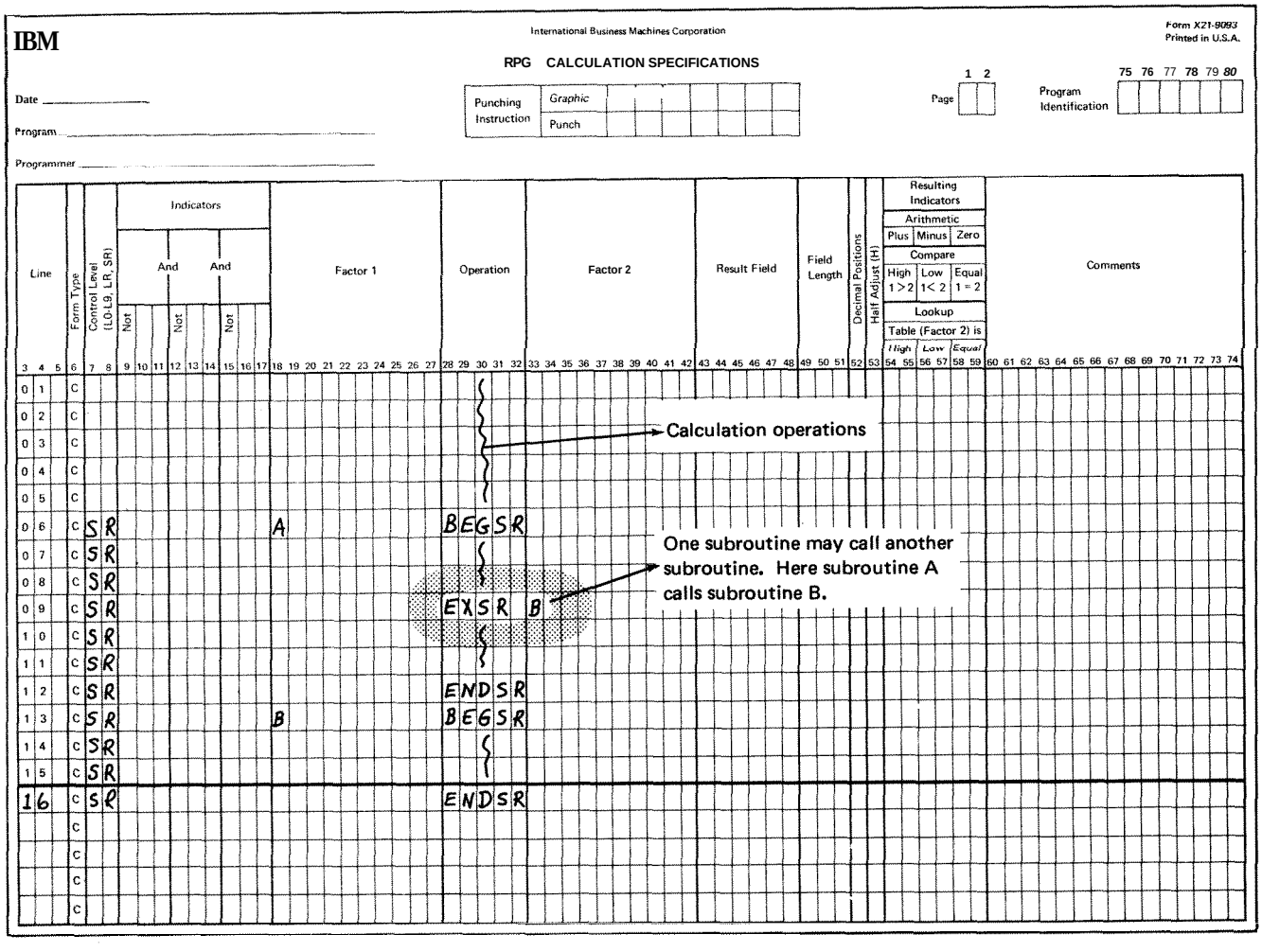


Figure 10-66. Subroutines: Calling Another Subroutine

Use of One Subroutine in Many Different Programs

When you wish to do the same operations in many different programs, you may use a subroutine to eliminate duplicate coding in each program Merely code these operations once, punch them on cards, and use this subroutine deck along with your main program deck.

Whenever you code a subroutine to be used in several different programs, remember:

1. When you call the subroutine in your main program (EXSR operation code), you must use the correct name of the subroutine in Factor 2.
2. All fields that will be used both by the subroutine and the main routine must be named the same in each routine, For example, if both the main routine and the subroutine used data from the field called COST on the input card, that field must be named COST in both routines. Keep in mind that the COST field also has the same characteristics (length, decimal positions) in both the main routine and the subroutine.

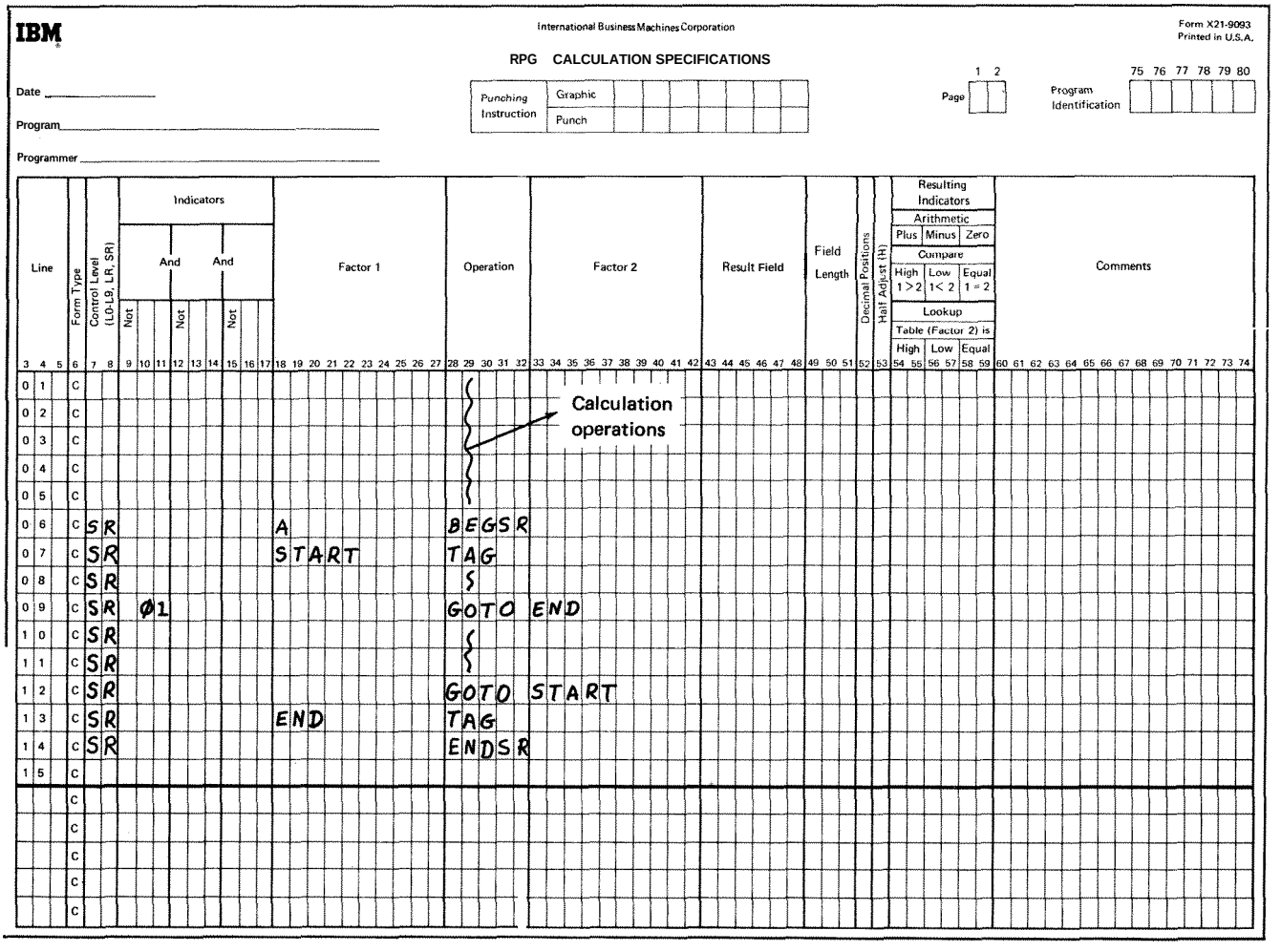


Figure 10-67. GOTO and TAG Within a Subroutine

C21-7504-0

IBM System/3 Printed in USD C21-7504-0

International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, New York 10601
(USA only)

IBM Wortd Trade Corporation
821 United Nations Plaza, New York, New York 10017
(International)

READER'S COMMENT FORM

C21-7504-0

Your answers to the questions on this sheet will help us produce better manuals for your use. If any of your answers require comments, or if you have additional information you think would be helpful, please use the space provided. All comments and suggestions become the property of IBM.

	Yes	No
1. Is the manual easy to read?		
2. Is any of the information unclear?		
3. Is additional information needed?		
4. Is any of the information unnecessary?		
5. Did you read the Preface?		
6. Did you use the Table of Contents?		
7. Did you use the Index? *		
8. Did you take the tests? *		

* Not included in all manuals

9. How did you use the manual:

Instructor in a class _____
Student in a class _____
Reference material _____
Self-Training _____
Other (Explain) _____

Have you had previous computer or programming training? _____

What is your present job? _____

What business is your company engaged in? _____

COMMENTS

● Thank you for your cooperation. No postage necessary if mailed in the U.S.A.

C21—7504—0

YOUR COMMENTS, PLEASE. . .

Your answers to the questions on the back of this form, together with your comments, will help us produce better publications for your use. Each reply will be carefully reviewed by the persons responsible for writing and publishing this material. All comments and suggestions become the property of IBM.

Note: Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.

Cut Along Line

Fold

Fold

FIRST CLASS
PERMIT NO. 387
ROCHESTER, MINN.

BUSINESS REPLY MAIL

NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

POSTAGE WILL BE PAID BY . . .

IBM Corporation
Systems Development Division
Development Laboratory
Rochester, Minnesota 55901

Attention: Programming Publications, Dept. 425

Fold

Fold

IBM System/3 Printed in USA C21—7504—0

IBM

International Business Machines Corporation
Data Processing Division
112 East Post Road White Plains, N.Y.10601
'USA Only

IBM World Trade Corporation
821 United Nations Plaza New York, New York 10017
{International