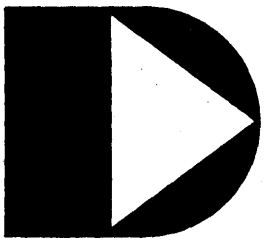
REPORT PROGRAM GENERATOR II
RPGPLUS (VERSION 1)
RPGII (VERSION 4)
User's Guide

April, 1978

Model Code No. 50325

DATAPOINT CORPORATION



The leader in dispersed data processing ™

REPORT PROGRAM GENERATOR II
RPGPLUS (VERSION 1)
RPGII (VERSION 4)

User's Guide

Version 4

April, 1978

Model Code No. 50021

PREFACE

This manual applies to the following versions of Datapoint's RPG II systems: RPGPLUS version 1, and RPGII version 4.* This manual is to be used as a source language reference and operation guide for all Datapoint RPG II systems. The RPG II systems are specifically RPGII and RPGPLUS. RPGII will operate on 1100 and 2200 systems as well 5500 and 6600 systems. RPGPLUS processes an enhanced version of the Datapoint RPG II source language to produce relocatable object code for 5500 and 6600 systems.

* RPGII version 4 contains all features and capabilities of RPG5500 version 1 and RPGII version 3.

TABLE OF CONTENTS

CHAPTER 1. INTRODUCTION

RPG II is a language oriented for business data processing. This document specifies RPG II for Datapoint systems. Source programs and data can be prepared using RPGPREP, CRPGPREP, or DATAFORM II.

After preparing the source program, one of the RPG II compilers must be run to produce the object program. RPGII object program output is directly executable under DOS, whereas the object program produced by RPGPLUS must be subsequently link edited (using the LINK utility) to produce a self-contained program which executes under DOS.

RPG source programs are rigidly formatted. In the subsequent text, source records will be referred to as if they were 80-column card images.

1.1 Installing The RPG II Compiler

Complete instructions for installating RPGPLUS are given in Appendix A; installation instructions for RPGII are given in Appendix F. The appendix also contains detailed instructions for compiling programs and indexing ISAM (Indexed Sequential Access Method) files.

1.2 Required Utility Programs

The following utilities at the indicated version/revision levels (or higher) are required for full utilization of RPG II:

RPGPREP	RPG Program Preparation Utility (DOS).
CRPGPREP	RPG Program Preparation Utility (Cassette).
INDEX	DOS Index Utility.
REFORMAT	DOS Reformat Utility.
SORT	Disk Operating System Sort.
LINK	DOS Linking Editor for Relocatable Modules (required for RPGPLUS only)

Note: Records output using the DOS SORT utility will be in ASCII sequence, unless specified otherwise.

1.3 Definition of Terms

EBCDIC (Extended Binary-Code-Decimal Interchange Code) Notation: The 256-character machine code used inside the Datapoint RPG II system. Files are automatically translated to EBCDIC when read, and from EBCDIC to ASCII (if necessary) when written.

Alphabetic Characters: The 26 alphabetic EBCDIC characters and the three EBCDIC characters '.', '$', and '#'.

Numeric Characters: The EBCDIC characters 0-9.

Special Characters: The 217 EBCDIC characters not defined as alphabetic or numeric.

Alphanumeric Characters: Any of the 256 EBCDIC characters.

Alphanumeric Fields: All fields for which a decimal-positions specification has not been made in the appropriate column of the specifications forms. Alphanumeric fields can contain alphabetic, numeric, or special characters.

Numeric Fields: All fields having a decimal-positions specification in the appropriate columns of the specifications forms.

Valid Datapoint RPG II Names: The following rules apply to names used in RPG II programs:

RPG II filenames can be from 1-8 characters long; RPG II field names can be from 1-6 characters long.

The first character of either a filename or a field name must be alphabetic (see preceding definitions of alphabetic characters). The remaining characters can be any combination of alphabetic and numeric characters (special characters are not allowed).

Blanks cannot appear between characters in the name.

1.4 General Datapoint RPG II Program Logic

Every Datapoint RPG II object program has the same general program logic. This logic is based on the processing cycle performed for each input record read. Every program cycle involves three basic steps.

1. Reading information (input).

2. Performing calculations (processing).

3. Recording results (output).

In the RPG II cycle, calculation and output can occur at two different times in the cycle: total time and detail time.

1.4.1 Operations at Total Time

Total calculation and output are normally performed on data accumulated for a group of related records which form a control group. When the fields of a record which determine the control group change, a control break occurs indicating a new control group is starting. When a break occurs (shown by control level indicators being turned on), calculation and output operations are performed using information accumulated from all records in the previous control group.

1.4.2 Operations at Detail Time

Detail calculations and detail output are normally performed for individual data records. Total operations are performed before detail operations. Thus a record which causes a control break is processed after the total operations for the previous control group.

1.4.3 General Program Cycle

An RPG object program proceeds through three major steps:

1. Initialization

2. Processing Cycle

3. Termination

Initialization consists of: loading all necessary tables, clearing all working areas, and opening all files for the processing cycle.

The processing cycle consists of: writing all heading and detail records, reading the next input record and identifying it, performing total operations if a control break occurs, and then calculating all results from the record previously read. This cycle repeats until the last record is processed.

The first and last cycles are somewhat different from the normal cycle. Before the first record is read, detail output conditioned by the 1P indicator and unconditioned detail output is performed. Total processing is bypassed until the cycle after the first control break.

After the last record has been read, the last record indicator (LR) is turned on, as well as all control levels. After total processing has been performed, the normal cycle is aborted and the termination routines are processed.

Termination consists of: writing all necessary tables and closing all files.

A detailed description of the object program logic is found in Appendix K.

1.5 Source File Order and Program Specifications

The RPG source file consists of a source program, optionally followed by compile-time tables and arrays. The source file contains up to seven sections which can be coded on seven standard RPG specification sheets and entered using the RPGPREP utility program. The seven forms are:

1. Header Specification. This specification contains control information for the compiler.

2. File Description Specification. This specification contains file description information about each file used in the program.

3. Extension Specification. This specification contains extension information about each table or array used in the program.

4. Line Counter Specification. This specification contains information about the number of lines to be printed on each

form.

5. Input Specification. This specification contains information describing the records read by the program.

   NOTE: In all versions of RPGII, the order of references to files in the Input Specifications must correspond to the order of occurrence of file declarations in the File Specifications. This is not a restriction in RPGPLUS.

6. Calculation Specification. This specification describes all calculations performed by the program.

7. Output Format Specification. This specification describes the format of all records written by the program.

The first part of the source program may end with: a normal EDIT end-of-file, a record containing '/*b' in columns 1-3, or a record containing '**b' in columns 1-3 (where b indicates a blank). The first two cases signify the end of the source program; the last case signifies that a user library inclusion and/or compile-time tables follow the source program.

If code from a user library is required, (see Appendices E and I for details), a library inclusion record must immediately follow the '**b' record. A user library is required for SPECIAL input-output devices and non-standard tape labels. The EXIT operation (see Chapter 8) also requires a user library. The format of a library inclusion is: '*LIBRARY' in columns 1-8, followed by one or more blanks, followed by a DOS file name. If no extension is supplied, the extension 'REL' is assumed by RPGPLUS; RPGII uses 'RPG' as the default. If compile-time tables are also used, a '**b' record should follow the library inclusion. Compile-time tables follow the first '**b' record if no library is included, or the second '**b' record if one is used. Each compile-time table must appear in the order specified on the extension sheets, and must be separated by '**b' records. See Chapter 5 for a description of tables and arrays.

After any of the preceding optional sections have appeared, the source file should terminate with either an EDIT end-of-file or a record containing '/*b' in columns 1-3.

CHAPTER 2. COMMON FIELDS ON SOURCE

This chapter defines entries 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).

2.1 Columns 1-2 (Page)

Entry	Explanation
01-99	Page number.

Columns 1-2 are for numbering the specification sheets used in a job. You can use more than one of each sheet, but all sheets of the same type must be kept together. When all the specifications sheets are filled out, arrange them in the following order and number them in ascending sequence:

1. Header Card.
2. File Description.
3. Extension.
4. Line Counter.
5. Input.
6. Calculation.
7. Output Format.

2.2 Columns 3-5 (Line)

Entry	Explanation
Any numbers	Line numbers.

Columns 3-5 are used to number the lines on each sheet. Columns 3-4 contain preprinted line numbers, so in most cases line numbering is already done. The unnumbered lines below the preprinted numbers can be used for additional lines or to insert a line between two other completed lines. Any other lines on the sheets can be skipped. The line numbers used need not be consecutive, but should be in ascending order. Line numbers are optional. Note: RPGPREP automatically supplies line numbers. Column 5 is always set to zero to allow later insertion of up to nine new lines.

2.3 Column 6 (Form Type)

Entry	Explanation
H	Header.
F	File Description Specifications.
E	Extension Specifications.
L	Line Counter Specifications.
I	Input Specifications.
C	Calculation Specifications.
O	Output Format Specifications.

Column 6 contains a code for each type of source statement.

2.4 Column 7 (Comments)

Entry	Explanation
*	Comment line.

You may want to write comments to help you understand or remember what is being done in a certain section of coding. Datapoint RPG

II allows an entire line to be used 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.

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

2.5 Columns 75-80 (Program Identification)

Columns 75-80 on all source program cards may contain any characters. These columns may use the program name, or the columns 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.

Note: Any entry made in these columns on the header specification will be automatically placed into these columns of each source statement by RPGPREP.

CHAPTER 3. HEADER SPECIFICATION

One header line is required for every program. It provides information about your program and your system to the RPG II compiler. Without this information your source program cannot be translated into an RPG II object program.

3.1 Columns 1-2 (Page) and 3-5 (Line)

The header line must always be line "010" (columns 3-5). Refer to Chapter 2 for additional details.

3.2 Column 6 (Form Type)

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

3.3 Columns 7-9

Columns 7-9 are not used. The program is compiled in the available core storage.

3.4 Column 10 (Object Output)

Column 10 is checked and if it contains neither C, D, or blank, a warning is produced. The program identification is ignored. No object program is produced when severe (terminal) errors are present in the source statements.

3.5 Column 11 (Listing Options)

Entry	Explanation
Blank	1. The object program is produced (if no severe errors are found). 2. The program listing is printed.
B	1. The object program is produced (if no severe errors are found).

2. The program listing is not printed.

Column 11 provides for listing options at the time your source program is compiled. If any severe errors are found during compilation, the system halts after completing the listing (provided a listing is to be printed).

The blank entry is the usual case, producing an object program (if no severe errors are found) and a source program listing. The program listing consists of the source program, error messages, and a core map. The core map lists such information as relative addresses of fields, constants, and I/O areas. The core map is printed only if the program is successfully compiled. The B entry means that no program listing is printed; however, an object program is produced.

3.6 Columns 12-14 (Core Size to Execute)

Core Size to Execute is documentary only in Datapoint RPG and does not affect program execution in any way!

3.6.1 Column 12

Entry	Explanation
Blank,	No additional 256-byte increments are needed.
Q	One additional 256-byte increment is needed.
H	Two additional 256-byte increments are needed (512 bytes).
T	Three additional 256-byte increments are needed (768 bytes).

Column 12 may be used to specify additional 256-byte increments of storage. These increments document an extra 1/4K, 1/2K or 3/4K of storage to be required in addition to the storage specified in columns 13-14.

3.6.2 Columns 13-14

Entry	Explanation
Blank	The core storage required for object program execution is the same as that used to compile the program.
01-13	The core storage required for program execution (if different from core storage available for object program generation).

Use columns 13-14 to specify some multiple of 1K bytes of storage (K=1024). Columns 13-14 document the core storage required for program execution. The entry must end in column 14.

3.7 Column 15 (Debug)

Entry	Explanation
Blank	DEBUG operation is not performed.
1	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.

3.8 Columns 16-25

Columns 16-25 are not used. Leave them blank.

3.9 Column 26 (Alternate Collating Sequence)

Entry	Explanation
Blank	Normal (EBCDIC) collating sequence is used.
A	ASCII collating sequence is used.

Use column 26 if job collating sequence is in ASCII. Sequence checking, comparisons and indexed disk files will be processed in ASCII sequence.

Using the ASCII collating sequence is recommended since it is the normal sequence used by other Datapoint software. The collating sequence used in an RPG program must agree with the collating sequence used to INDEX or SORT files.

3.10 Columns 27-74

Columns 27-74 are not used. Leave them blank.

3.11 Columns 75-80 (Program Identification)

See Chapter 2.

CONTROL CARD SPECIFICATION

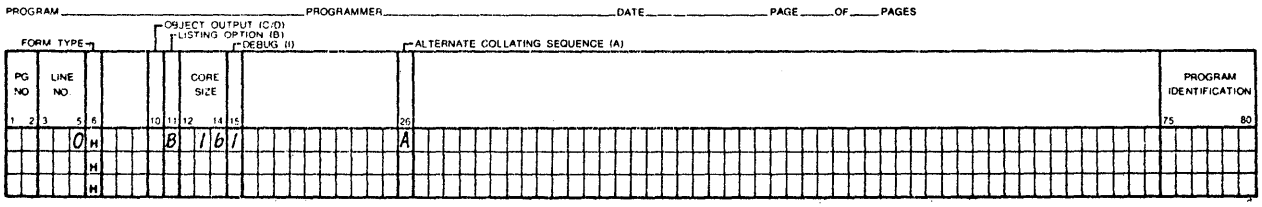


Figure 3-1. Example of a Header Specification.

CHAPTER 4. FILE DESCRIPTION SPECIFICATIONS

File description specifications are required for every file used by a program. Write these specifications on the File Description Sheet. At least one line is needed to describe a file. Datapoint RPG will allow a maximum of 24 file description entries.

4.1 Columns 1-2 (Page) and 3-5 (Line)

Refer to Chapter 2.

4.2 Column 6 (Form Type)

An F must appear in column 6.

4.3 Columns 7-14 (File Name)

Use columns 7-14 to assign a unique filename to every file used in your program except compile-time table and array files, which must not be named on the File Description Sheet. (Compile time tables and arrays are described on the Extension Sheet). The filename can be from 1-8 characters long, must begin in column 7, and must be a valid RPG II name. The filename can be the same as a field name.

Pre-execution time table and array files are described on the File Description Sheet.

The entry in this field may be the same as the DOS file name used when executing this program or it may be completely internal to the program. See Column 53 (Continuation Code) for details.

When assigning file names for processing existing indexed files, the file name should refer to the index to be used. The associated data file will be selected whenever the index is referenced.

4.4 Column 15 (File Type)

Entry	Explanation
I	Input file
O	Output file
U	Update file
D	Display file

Use column 15 to identify the way in which your program uses the file. All input file descriptions must preceed other file descriptions.

4.4.1 Input 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 except table and array files must be further described on the Input Sheet. Table and array files must be further described on the Extension Sheet.

4.4.2 Output Files

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

4.4.3 Update Files

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 Sheet and Output-Format Sheet. A chained file or a demand file may be updated at detail time, at total time or exception time. All other disk files can be updated only at detail time during the same program cycle that reads the record.

4.4.4 Display Files

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 display 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 keyboard-display (CONSOLE). See Operation Codes, DSPLY in Chapter 8 for more information.

4.5 Column 16 (File Designation)

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

Use column 16 to further identify the use of input, update, and chained files. Leave the column blank for display files and all output files except chained and table output files.

4.5.1 Primary Files

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 or update 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. The primary file description must be the first file description entry.

4.5.2 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, or update file. Secondary files are processed in the order in which they are written in the file description specifications, except when matching records (MR) or when the FORCE operation is used. Note that table, chained, and demand files are not involved in record selection in multifile processing.

4.5.3 Chained Files

A chained file is a disk file that is read or written randomly via the CHAIN operation code. A chained file can be an input, output, or update file. If it is output and indexed, the A option must be specified in column 66. This is because a chained, indexed file must have an index built using the INDEX Utility program. If this has been done, the file already exists and any records written are appended to the existing file, and inserted into the index.

4.5.4 Record Address Files

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

Record address files contain binary relative record addresses and are called ADDROUT (address output) files. They are disk files produced by the DOS 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.

4.5.5 Table or Array Files

A table or array file is an input or output file that contains table or array entries. The entries can be read into the program from a table input file immediately before execution of the program. Only pre-execution time tables or arrays are described on the File Description Sheet. However, both pre-execution and compile time tables and arrays must be described

in the Extension Sheet.

A table or array output file (written after LR output) can be defined and used as a normal output file and does not require an entry in column 16. If the only output to the file is tables and arrays, the file should be designated as a table output file.

Table and array files are not involved in record selection and processing. They are only a means of supplying entries for tables or arrays used by the program. When table or array files are read during the execution of the program, the program reads all the entries from the table or array files before it begins record processing. See Chapter 5 for additional information.

4.5.6 Demand Files

Demand files can be input or update files. The READ operation code must be used on the Calculation Sheet in order to read consecutively from a demand file. Demand files can only be processed consecutively. See Operation Codes, READ in Chapter 8 for a discussion of processing demand files.

4.6 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 and update files that are used as primary or secondary files.

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

4.7 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, and all input files except table, array, chained, and demand files. Leave column 18 blank for output, display, table or array files, and chained files. 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 matching fields containing the sequence information. The proper collating sequence for sequence checking (EBCDIC or ASCII) is determined by column 26 in the Control Card Specification.

Sequence checking is required when matching fields are used in the records from the file. When a record from a matching input file is out of sequence, the program halts, and the operator has three options:

1. Bypass the record out of sequence and read the next record from the same file (BYPASS option).

2. Bypass the record out of sequence, turn on the LR indicator and perform all end-of-job and final total procedures (CANCEL option).

3. Immediately discontinue program execution (ABORT option).

See Appendix A (RPGPLUS) or F (RPGII) for a detailed description of operating procedures.

4.8 Column 19 (File Format)

Entry	Explanation
F	Fixed-length records.
V	Variable-length records.

In Datapoint RPG II there are two types of file organizations, fixed and variable. Disk and cassette files may be either; files on other devices must be fixed. For cassette input either 'F' or 'V' may be used, with no effect on program execution.

4.8.1 Fixed Format Files

These files have definite record length and are not subject to special processing. Disk files to be updated or processed randomly must be fixed format. See Appendix M for further details about fixed-format disk files. Cassette output file records will be written without space compression if an 'F' is used in column 19.

4.8.2 Variable Format Files

These files have a maximum record length and are compatible with Datapoint software using the sequential record format such as the general purpose editor program EDIT. On input, blanks are expanded; on output, blanks are compressed. See Appendix M for further details about variable-format disk files. Cassette output files will be written with space compression if a 'V' is used in column 19.

4.9 Columns 20-23 (Block Length)

Entry	Explanation
Number	Length of block.
Blank	Default block length.

These columns have differing interpretations depending on the device assigned for the file (see Columns 40-46). If an entry is specified, it must end in column 23. Leading zeros may be omitted.

4.9.1 Disk Files

The block length for fixed-format disk files may be a multiple of the record length. This is allowed for language compatibility, however, Datapoint RPG will always assign the most efficient block length. For variable-format disk files these columns must either be blank or equal to the record length.

4.9.2 Tape Files

These columns may either be blank or contain a multiple of the record length. In order to properly process tape input files, the block length entry used must be the same as the block length used when the tape file was written.

4.9.3 Other Files

These columns must either be blank or contain the record length.

4.10 Columns 24-27 (Record Length)

Entry	Explanation
Number	The number of characters used in each record (limited by the device used).

Use columns 24-27 to indicate the length of the records in the file. For variable-format files the record length defines the maximum size of a record. The actual size is determined by the data read or written. For fixed-format files, information is transferred in units of the record length. All of the records in one file must be the same length. (For update files, the length of a record after it is updated must be the same as before it was updated). The maximum record length allowed and the size of the I/O area assigned depend upon the device assigned to the file. The record length specified may be shorter than the maximum length for the device. The entry placed in these columns must end in column 27. Leading zeros can be omitted.

4.11 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 processed. Only indexed disk files can be processed sequentially by key or within limits. Disk files that are indexed, chained or controlled by an ADDROUT file can be processed randomly. All other files must be processed consecutively.

Column 31 is used to further identify the processing method. See column 31 (Record Address Type) in this chapter.

4.11.1 Consecutive Method

The consecutive method applies to all 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 (fixed-format) files are read as though the records were there. (Such spaces are filled with blanks).

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.

4.11.2 By ADDROUT File

An ADDROUT (address output) file is a record address file produced by the DOS SORT Program. It is a file of 3-byte disk records containing binary relative record addresses of records in a disk file. RPG II locates and reads the record at the specified 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.

4.11.3 Sequential By Key

Processing sequentially by key applies only to indexed disk files that are used as primary, secondary or demand files.

Records are read in ascending key sequence established when the file was INDEXed by the INDEX utility program (See Appendices A/F). The alternate collating sequence option (Column 26 in the Header Specification) must agree with the option used when INDEXing the file (EBCDIC or ASCII). 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.

4.11.4 Sequential Within Limits

Sequential within limits processing is accomplished using the SETLL operation code during calculations. The SETLL operation code is used to establish a lower limit for sequentially processing primary, secondary, or demand files. The upper limit (if not end-of-file) must be checked using the COMP operation code.

When using SETLL with primary and secondary files, care should be exercised to discard the first record read (as part of the normal input cycle) prior to entering calculations and executing the first SETLL. (See Operation Codes, SETLL in Chapter 8).

4.11.5 Random Method

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.

When processing fixed blocked disk files "directly" (without using an ISAM index), relative record numbers are 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. (See Operation Codes, CHAIN in Chapter 8).

For indexed files, record keys must be used to locate the records. A record key is the information used to match unique data in a field in each record that is used to identify that record. Record key fields are defined when a fixed blocked disk file is indexed with the INDEX utility program (See Appendices A/F).

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

4.12 Columns 29-30 (Length of Key)

Entry	Explanation
Number	Length of record key or ADDROUT file record

Columns 29-30 apply only to indexed disk files and record address files. Enter:

1. The length of the record keys in indexed files.

2. The length of the records in ADDROUT files.

All of the key fields in the records in an indexed file must be the same length. The maximum is 99 bytes. All of the records in an ADDROUT file have a length of three. A leading zero is not required.

4.13 Column 31 (Record Address Type)

Entry	Explanation
A	Record keys in unpacked format are used in processing indexed files.
I	The file is being processed by means of an ADDROUT file or the file is an ADDROUT file.
Blank	1. 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 indicates the way in which records in a disk file are identified.

4.14 Column 32 (File Organization)

Entry	Explanation
I	Indexed file.
T	ADDROUT file.
Blank	Sequential file or direct file.
1-9	Additional I/O areas (ignored).

Use column 32 to identify indexed and ADDROUT files. See Column 28, Mode of Processing for further details. A digit is allowed in this column for compatibility with other RPG systems, but has no effect on the execution of the program.

4.15 Columns 33-34 (Overflow Indicator)

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 the output file 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 the output file assigned to the printer. Note that only one overflow indicator can be assigned to a file. Do not assign overflow indicators to a console file.

4.15.1 Overflow Indicator

Overflow indicators are used only with printer files, 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). The same indicator must 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 is a line beyond the overflow line, the overflow indicator is turned on and remains on until all overflow lines are printed. However, if a skip or space 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.

If an overflow indicator is used as a conditioning indicator, it indicates that output is to be performed at overflow time. This applies whether or not the line conditioned by the indicator is in an AND or OR relationship with other 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.

4.16 Columns 35-38 (Key Field Starting Location)

Entry	Explanation
1-255	Record position in which the key field begins.

Columns 35-38 apply to indexed files only. An entry must be made in these columns for an indexed disk file. Enter the location in which the key field begins in indexed file records.

The number entered must end in column 38. Leading zeros can be omitted.

4.17 Column 39 (Extension Code)

E	Extension specifications further describe the file.
L	Line counter specifications further describe the file.

Column 39 applies to (1) table and array files that are to be read during program execution, (2) record address files, and (3) the output file assigned to the printer. Output files that are assigned to the printer can be described on the Line Counter Sheet. Tables, array, and record address files must be described on the Extension Sheet. If tables are output to the printer 'E' should not be used.

4.18 Column 40-46 (Device)

Entry	Explanation
PRINTER	Printer.
CONSOLE	Keyboard Display.
DISK	Disk.
READER	Card Reader.
TAPE	Industry-compatible 9 track tape unit.
CASSET1	Rear tape cassette.
CASSET2	Front tape cassette.

LOADER Pseudo-device for use with the DOS CHAIN command

SPECIAL Special input/output device not supported by Datapoint RPG II. (See Appendix F).

These columns are used to specify the input/output device to be used for the file. All entries begin in column 40. The devices that can be used depend upon the type of file access specified.

AVAILABLE DEVICES

FILE	MEDIA	DEVICES	MAXIMUM BLOCK LENGTH
Primary or Secondary Input	Disk	DISK (fixed or variable format)	9999
	Cards	READER	80
	Tape(800 BPI)	TAPE	1057
	Tape(1600 BPI)	TAPE	2048
	Cassette	CASSET1 or CASSET2	249
	Keyed In	CONSOLE	78
Chained Input Files	Disk	DISK (fixed format)	9999
Update Files (Primary, Secondary, or Chained)	Disk	DISK (fixed format)	9999
Output Files	Disk	DISK, (fixed or variable format)	9999
	Tape(800 BPI)	TAPE	1057
	Tape(1600 BPI)	TAPE	2048
	Cassette	CASSET1 or CASSET2	249
	Printed	PRINTER	132
	CRT	CONSOLE	80
Display File	CRT	CONSOLE	80
DOS Chaining	none	LOADER	80
Special Files	unknown	SPECIAL	9999

4.18.1 Use of the LOADER Device for Program Chaining

This device may be used to construct a command line to be executed by the DOS command processor after the normal end of an RPG II object program. The LOADER must be used as an output file, and the command line can typically be written during the last total cycle (LR-time). The simplest use of the LOADER is to invoke some other program, for example, the DOS SORT. By constructing a sequence of DOS commands on some disk file and then writing "CHAIN file" on the LOADER, where "file" is the DOS name of the command file, more complex sequences may be realized.

Only one record may be written on the LOADER during the execution of the object program. If the object program aborts, rather than concluding normally, the command line is ignored.

4.18.2 SPECIAL Device Support

You can process files using devices not supported by Datapoint RPG II. To do this, you must indicate that the file will be handled by a SPECIAL device (SPECIAL in columns 40-46 of the File Description Sheet). You must also supply a subroutine to perform the I/O operations required to transfer data between the SPECIAL device and core storage (subroutine name in columns 54-59 of the File Description Sheet).

The following can be used with SPECIAL files:

FORCE operation code.
READ operation code.

The following cannot be used with SPECIAL files:

CHAIN operation code.
Spacing and skipping.

SPECIAL files can only be processed consecutively. See Appendix E for the conventions used by RPGPLUS to call the input-output subroutine; Appendix I gives the RPGII conventions.

4.19 Columns 47-52

These columns are not used and should be left blank.

4.20 Columns 53-65 (Continuation Lines)

4.20.1 Column 53 (Continuation Code)

Entry	Explanation
A	Assign disk file name at run-time.
D	Disk file name defined at compile-time.
S	Standard labels are used.
N	Non-standard labels are used.
U	No labels are used.
K	Continuation Record.

If the file being defined is a disk file, column 53 may contain 'A' or 'D'. (If it is left blank, 'A' is assumed.) When the 'A' entry is used, the object program will ask for the DOS external file name to assign to the current internal file. When the 'D' entry is used, the external file name is assumed to be the same as the internal file name. See Appendix A and the 'EXTDRV' entry in columns 54-59 for additional information.

Column 53 must contain 'S', 'N', or 'U' if the file is a tape file. If non-standard labels are being used, columns 54-59 must contain the name of the user-supplied subroutine for processing the labels. (See Appendices E/I for calling conventions.)

If the preceding File Description line describes an ASCII tape file, column 53 must contain 'K' and columns 54-59 must contain 'ASCII'.

If the preceding File Description line describes a disk or printer file, column 53 may contain a 'K', in which case columns 54-65 contain additional file specifications. See the following discussion.

4.20.2 Columns 54-59 (Continuation Option)

Entry	Explanation
EXTDRV	Extension and/or drive for disk file (Used to specify the extension and/or disk drive assigned for defined DISK files (see Column 53 above).
MAXSEC	Maximum number of sectors for new disk files.
LOCAL	The Local printer is used at object (execution) time.
SERVO	The Servo printer is used at object (execution) time. (The DEVICE in the preceding File Description line must be PRINTER).
ASCII	TAPE file written in ASCII.
800 or 1600	Tape density(if not given for a tape file 800bpi is assumed).

4.20.2.1 Columns 54-59 (Name of Label Exit)

Entry	Explanation
RPG name	Name of the user-written subroutine which will perform the I/O operation for a SPECIAL device, or which will process non-standard tape labels.
Blank	No SPECIAL device or non-standard labels are being used.

Columns 54-59 must contain an entry for each data file assigned to a SPECIAL device or to a TAPE file with non-standard labels. These columns are used to specify the subroutine which will perform the input/output operations for a file assigned to a SPECIAL device or for non-standard tape label processing. The subroutine name entered in columns 54-59 can be from one to six characters long, and must be a valid RPG II name.

4.20.2.2 Columns 60-62 (Extension)

Use these columns on a continuation card with the 'EXTDRV' entry in columns 54-59 to specify the extension to be used for a (compile-time) defined disk file. Datapoint RPG assumes an extension of 'TXT' for all disk files except ISAM files. For ISAM files, an extension of 'ISI' is assumed.

4.20.2.3 Columns 63-65 (Drive)

Use these columns to specify the drive for a compile-time defined disk file. Single-digit drive numbers are specified by 'DR0' through 'DR9' or by 'D00' through 'D09'. Double-digit drive numbers are specified by 'D00' through 'D15'.

Either entry (EXT or DRV) can be specified, or left blank. If an entry is left blank, '/TXT' is assumed for the extension, and any drive, starting with ':DR0', is assumed for the drive. Do not specify the initial '/' on the extension, or the ':' on the drive. See the DOS manuals for further details.

4.20.2.4 Columns 60-65 (Number of Sectors)

These columns may be specified on a continuation card with the 'MAXSEC' entry in columns 54-59 to specify the LRN limit to be used for any disk file which may be created by the object program. This entry no longer has any effect in Datapoint RPG.

4.20.2.5 LOCAL/SERVO Continuation Option

For files assigned to the printer, you may specify whether a local or servo printer will be used at execution time by means of a continuation card. If no specification is given local printer is assumed.

4.20.2.6 ASCII Continuation Option

The ASCII continuation option is used for a tape file written in ASCII. If the tape file is in ASCII format this option must be used.

4.20.2.7 Tape Density Continuation Option

The tape density continuation option is used to give the density of the tape being used. If the density of a tape file is not given 800bpi is assumed. This option may be used in conjuction with the ASCII option.

4.21 Column 66 (File Addition)

Entry	Explanation
A	New records are to be added to the file.

Column 66 must contain an 'A' when new records are to be added to an existing consecutive or indexed disk file.

Records added to a consecutive file are added to the end of the file. To add records to a sequential file, the file must be an output file (O in column 15 of the File Description).

Records added to an indexed file are added to the end of the file and the index used for the operation is updated to refect the addition. New records may be added in any order and will be indexed into the proper sequence. To add records to an indexed file, the file must be an output or update file (O or U in column 15 of the File Description).

If an indexed file has more than one index (indexed on more than one key using the INDEX utility) the new records can not be accessed using the other indices until the other indices have been updated using the INDEX utility! (See Appendix A).

Column 66 should be blank for direct files.

4.22 Columns 67-70

These columns are ignored.

4.23 Columns 71-72 (File Conditioning Indicator)

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 primary and secondary input (excluding table input files), update, and output files. If an output file is conditioned by an external indicator which is off, records will not be written on that file. Any calculation operation which should not be done when the file is not in use should also be conditioned by the same indicator. When the indicator is off, the file is treated as though the end of file had been reached; that is, no records can be read from or written into the file.

4.23.1 U1-U8 (External Indicators)

Indicators U1-U8 are external indicators. This means they are set during start-up. 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.

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 Sheet).

4.24 Columns 73-74

Columns 73-74 are not used.

4.25 Columns 75-80 (Program Identification)

See Chapter 2.

FILE DESCRIPTION SPECIFICATION

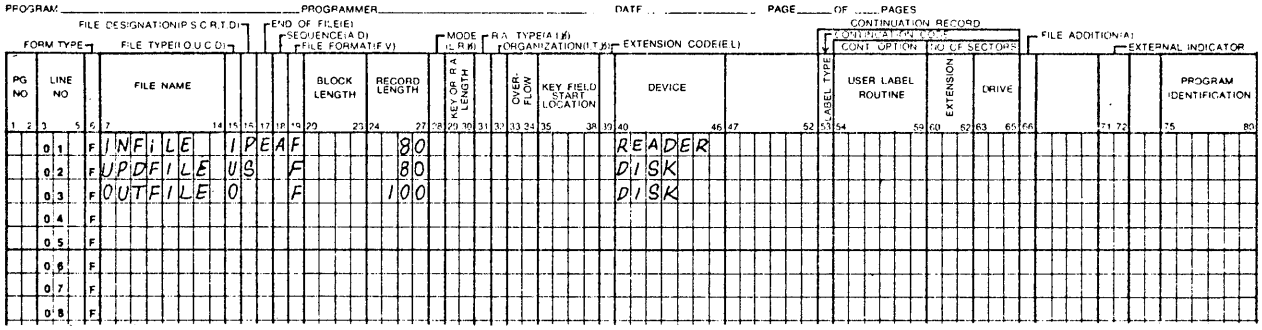


Figure 4-1. Example of simple File Description Specification.

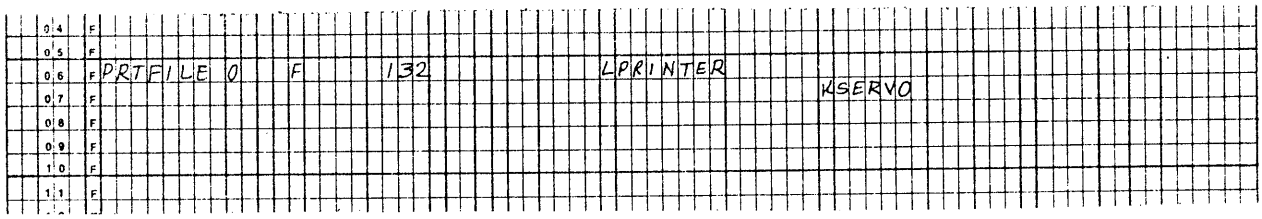


Figure 4-2. Example of continuation line usage in File Description Specification.

CHAPTER 5. EXTENSION SPECIFICATIONS

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 Sheet.

Pre-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 46-57 of the same line as the first.

Record address files require entries on the Extension Sheet in columns 11-26.

5.1 Columns 1-2 (page) and 3-5 (Line)

See Chapter 2.

5.2 Column 6 (Form Type)

An E must appear in column 6.

5.3 Columns 7-10

Columns 7-10 are not used.

5.4 Columns 11-18 (From Filename)

Entry	Explanation
Record Address Filename	The name of the record address file defined on the File Description Specification Sheet.
Table or Array Filename	Table or array file loaded at pre-execution time.
Blank	1. Table or array loaded at compilation time if an entry appears in Number of Entries

per Record (columns 33-35).

2. Array loaded at execution time (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.

Leave columns 11-18 blank for compile time tables or arrays, or for arrays loaded via input or calculations specifications (execution time array). These columns must contain the table or array filename of every pre-execution time table or array used in your program.

5.5 Columns 19-26 (To Filename)

Entry	Explanation
Name of an input or update file	The file processed via the record address file name under From Filename.
Name of an output file	The output file on which a table or array is to be written at end of job.

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 name of the primary or secondary file that contains the data records to be processed must be entered in To Filename, columns 19-26.

If you wish a table or array to be written, 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. Execution time arrays cannot be written at end-of-job. Leave columns 19-26 blank for execution time arrays or if you do not want the table or array written.

If a table or array is to be written, it is automatically written

at the end of the job after all other records have been written.

Since the table or array will be written in the same format in which it 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 8). Tables or arrays will be written under RPG II control only after all records have been processed (Last Record indicator is on). Note: If a table or array is to be written to a printer file at the end of a job, the last Output Format Specification should be a space or skip to the line at which table or array output should begin.

5.6 Columns 27-32 (Table or Array Name)

Entry	Explanation
Table or Array	Name of a 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 one to six characters long and must begin in column 27, and must be a valid RPG II name. If alternating tables or arrays are being described, this must name the table or array whose entry is first on the input record.

5.6.1 Table Name

Every table used in your program must be given a name from three to six characters long 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. Factor 2 on the Calculation Sheet can contain the name of a table to be searched and the result field can contain the name of another table from which an associated function is to be obtained. When the table name is used in Factor 2 or Result Field (on the Calculation Sheet) with the 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. If the table name is used before any successful look-ups are performed, the first table item is referenced. See Operation Codes, LOKUP, in Chapter 8 for more information.

Tables are processed in the same order as they are specified on the Extension Sheet. Therefore, if you have more than one table, remember the tables are to be loaded in the same order as they appear on the sheet.

Tables cannot be used with an index.

5.6.2 Array Name

Every array used in your program must be given a name from one to six characters long. An array name cannot begin with the letters TAB. This array name is used throughout the program. Different results are obtained if the array name is used with an index or without an index. When used with an index, a particular element of the array is referenced. An array name used unindexed refers to the whole array.

An index is a numeric field or literal with zero decimal positions. When used to select an array element, the value of the index must not be negative, zero, or greater than the number of elements in the array. An indexed array reference is written as: array-name,index (note that the name and index are separated by a comma).The length of an array name is limited by its use. In input, output and the result field of calculations, the array element (array-name, index) is limited to six positions; in factor 1 and factor 2, to ten positions.

On input or output an entire array may be read or written to a single field or the array may be processed element by element. An indexed reference is treated like a normal field during calculation. An unindexed reference refers to the entire array. An entire array may not be used with: COMP, DSPLY, TESTZ, TESTB, BITON, or BITOF. Otherwise the following rules apply:

1. When all operands are arrays, the operation is performed element by element until the shortest array is processed.

2. When one operand and the result field are arrays, and the other operand is a field or literal, the operation is repetitively performed using the same field or literal.

3. Except for XFOOT and LOKUP, neither operand can be an array name unless the result field is an array name, and resulting indicators may not be used.

5.7 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. The number entered must end in column 35. 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, each table input record must contain the corresponding items from each table written in alternating form. These table items are considered as one entry. Corresponding items from related tables must be on the same record. If there is room, comments may be entered on table input records in columns following table entries.

When loading an array the following must be considered:

1. To load a pre-execution time array, the array 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 execution time array (via the input and/or calculation specifications), the From Filename (columns 11-18) and the Number of Entries per Record (columns 33-35) must be blank.

5.8 Columns 36-39 (Number of Entries/Table)

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 alternating tables. If alternating tables are described, corresponding table items are considered one entry. 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. A table or array that is not full is known as a short table or array.

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

5.9 Columns 40-42 (Length of Entry)

Entry	Explanation
1-256	Length of a table or array 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.

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 alphanumeric items. For alphanumeric items, blanks may be added either before or after the item.

If two related tables or arrays are described on one Extension Sheet, the entry in columns 40-42 applies to the table whose item appears first on the record.

5.10 Column 43 (Packed or Binary Field)

Entry	Explanation
Blank	Data for table or array is in IBM-compatible numeric format or is alphanumeric.
-	Data for table or array is in Databus-compatible format.

For a complete discussion of data representation, see Column 43, Packed or Binary Field in Chapter 7.

5.11 Column 44 (Decimal Positions)

Entry	Explanation
Blank	Alphanumeric 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 numeric table or array have no decimal positions, enter a 0.

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

5.12 Column 45 (Sequence)

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

Use column 45 to describe the sequence (ascending or descending) of the data in a table or array. Execution time arrays are not checked for sequence, but column 45 must contain an entry if high or low LOKUP is to be used.

When an entry is made in column 45, the table or array is checked for the specified sequence. If a pre-execution time table or array is out of sequence, an error occurs and the program halts immediately. The program can be restarted from the point where it halted if you do not want to correct the out-of-sequence condition; otherwise program execution must be restarted from the beginning.

Compile-time tables or arrays are sequenced checked at compile-time and a diagnostic is issued if they are improperly sequenced.

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 alternating tables or 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, LOKUP in Chapter 8 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. In checking for sequence, an equal condition (identical entries) is considered valid. This allows you to pad the beginning of the table with zeros or blanks, or to pad the end of the table with 9's (assuming EBCDIC, ascending sequence).

5.13 Columns 46-57 (Alternate Table/Array Specification)

Use columns 46-57 only when describing a second table or array which is entered in alternating format 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. An alternating array cannot be described with an execution time array. See the previous discussion on those columns for information about correct specifications.

5.14 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.

5.15 Columns 75-80 (Program Identification)

See Chapter 2.

EXTENSION SPECIFICATION

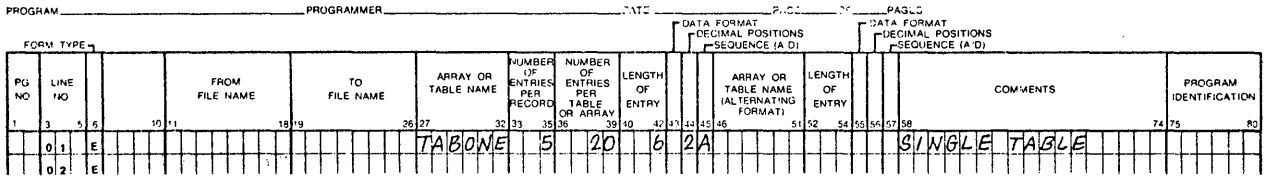


Figure 5-1. Example of a single table description on Extension Specification.

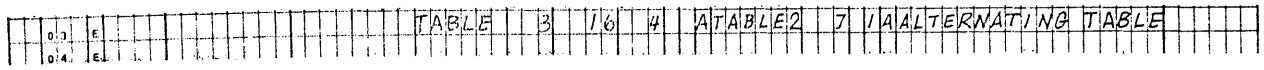


Figure 5-2. Example of alternating tables on Extension Specification.

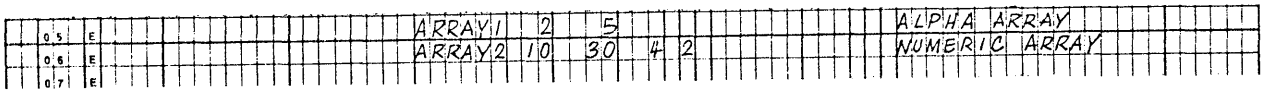


Figure 5-3. Example of an array description on Extension Specification.

CHAPTER 6. LINE COUNTER SPECIFICATIONS

Line counter specifications should be used for the printer file (except the keyboard-display) in your program. Line counter specifications indicate at what line overflow occurs and the length of the form used in a printer. Both of these entries must be specified on the Line Counter Sheet (Figure 6-1). Line counter specifications may be omitted if overflow indicators are not used with untitled 66-line forms.

6.1 Columns 1-2 (Page) and Columns 3-5 (Line)

See Chapter 2.

6.2 Column 6 (Form Type)

An L must appear in column 6.

6.3 Columns 7-14 (Filename)

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

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

6.4 Columns 15-17 (Lines per Page)

Entry	Explanation
1-99	Number of printing lines per form or page.

Columns 15-17 specify the exact number of lines on the form or page to be used. The entry must end in column 17. Leading zeros may be omitted.

6.5 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.

6.6 Columns 20-22 (Line Number of Overflow Line)

Entry	Explanation
1-99	A line number from 1-99 is the overflow line.

Columns 20-22 specify the line number that is the overflow line. The entry must end in column 22. Leading zeros may be omitted.

When the destination line of a space, skip, or print operation is a line beyond the overflow line you have specified (but not beyond the form length), 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.

6.7 Columns 23-24 (Overflow Line)

Entry	Explanation
OL	Overflow line

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

6.8 Columns 25-74

Columns 25-74 are not used, and should be left blank.

6.9 Columns 75-80 (Program Identification)

See Chapter 2.

LINE COUNTER SPECIFICATION

PROGRAM _____ PROGRAMMER _____ DATE _____ PAGE ____ OF ____ PAGES

FORM TYPE	PG NO (1-2)	LINE NO (3-5)		FILENAME (7-14)	FORM LENGTH (15-17)	FL (18)	OVERFLOW LINE NUMBER (20-22)	OL (23)		PROGRAM IDENTIFICATION (75-80)
L			L	PRTFILE	50	FL	44	OL		
L			L							
L			L							
L			L							
L			L							

Figure 6-1. Example of Line Counter Specification.

CHAPTER 7. INPUT SPECIFICATIONS

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

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

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

The specifications are written on the Input Sheet. Field description entries must not appear on the same line as file or record type identification entries.

7.1 Columns 1-2 (Page) and 3-5 (Line)

See Chapter 2.

7.2 Column 6 (Form Type)

An I must appear in column 6.

7.3 Columns 7-14 (Filename)

Columns 7-14 identify the input or update file being described. The filename must begin in column 7 and conform to RPG II naming specifications. Use the same filename given in the file description specifications. The name of every input or update file (except table input files) 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 records in that file. If the filename is omitted, the last input filename entered is assumed to be the file being described. All records and fields for one file must be completely described before another file can be described.

In RPGPLUS, the order in which files are described in the File Specifications need not be the same as the order in which they are described in the Input Specifications. However, in RPGII, files

must be described in the Input Specifications in the same order as they were described in the File Specifications.

7.4 Columns 15-16 (Sequence)

Entry	Explanation
Any two Alphabetic characters	No check for special sequence.
Any two-digit number	Check for special sequence

Columns 15-16 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. Alphabetic characters must be used for chained files and look ahead records. Within one file, record types having alphabetic and numeric sequence entries can be specified for the same file, but all alphabetic entries must be before the numeric entries.

Use columns 15-16 to assign sequence numbers to different types of records within a file. A job may require that one record type (identified by a record identification code) must appear before another record type within a sequenced group. For example, a name record may be needed before an address record. A record identification code must be provided for each type of record and the record types must be numbered 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. Gaps in sequence numbers are allowed, but the numbers used must be kept in ascending order. The first sequence number must be 01. 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. A record type out of sequence causes the program to stop. Program execution may be resumed, but the record that causes the halt is bypassed and the next record is read from the same file.

Records in an AND or OR line cannot have a sequence entry in these columns. The entry in these columns from the previous line also applies to the AND or OR line.

7.5 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.

Use column 17 only if sequence checking is to be done (columns 15-16 contain numbers). Often, when sequence checking, there may be more than one record of a particular type within the sequenced group, 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.

AND or OR lines (columns 14-16 have the letters AND or 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 21-41 and Columns 53-58 in this chapter for more information on OR lines).

7.6 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. If all record types are

optional, no sequence errors will be found.

AND or OR lines should not have an entry in this column. The entry in this column on the previous line also applies to this line. (See Columns 21-41 in this chapter for more information on AND lines; see Columns 53-58 for more information on OR lines).

7.7 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 fields.

Columns 19-20 may be used for two purposes:

1. Specifying record identifying indicators.
2. Indicating look-ahead fields.

7.7.1 Record Identifying Indicator

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, unless chained files or demand files are being used, in which case several may be on at the same time.) This indicator signals throughout the rest of the program cycle

which record type has just been selected. A record identifying indicator need not be assigned if you are not concerned about different record types.

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.

When a control level indicator used as a record identifying indicator turns on to reflect the type of record read, only that one control level indicator turns on. All lower level indicators remain off.

The same indicator may be assigned to two or more different record types provided the same operations are to be performed on these types. This can be done 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. Record identifying 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).

7.7.2 Look Ahead Fields

Use asterisks in columns 19-20 to indicate that fields named in columns 53-58 on the following specifications lines 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 file. In update files, the look-ahead field is for the record currently in process.

Two of the uses for look-ahead fields are:

1. Determining when the last card of a control group is being processed.

2. Extending the RPG II matching record capability.

Look-ahead fields can be used with input and update files. They cannot be specified for chained or demand files. 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 cannot be altered in the program (they cannot be used as a result field or blanked after).

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 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 an update file, look-ahead fields always apply to the current record.

At end of file, all look ahead fields for that file are filled with nines.

INPUT SPECIFICATION

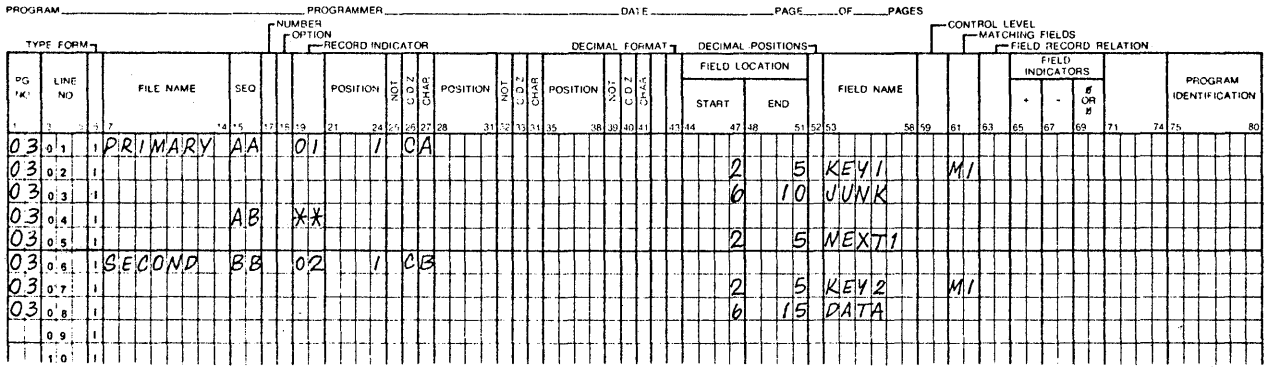


Figure 7-1. Example of look-ahead specification.

7.8 Columns 21-41 (Record Identification Codes)

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

When there are 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. The first record identifying character should be identified in columns 21-27, the second in columns 28-34, and so forth.

You may specify AND or OR lines in any combination 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.

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.

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. If a data record meets the requirements of more than one of the record types, it will belong to the first record type for which it qualifies. When all records are to be processed alike regardless of their type, or if there is only one type, leave columns 21-41 blank.

Note: If none of the identifying codes you have specified is found on a record, processing stops. You may continue. However, the record that caused the halt is not processed, and the next record in that file is read.

7.8.1 Position

Entry	Explanation
Blank	No record identificaton code is needed.
1-4096	Record position of 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. Leading zeros can be omitted.

7.8.2 Not

Entry	Explanation
Blank	Record ID code is present in the specified column.
N	Record ID code is not present in the specified column.

Use an 'N' in columns 25, 32, or 39 to indicate that a certain character should NOT be present in the specified position.

7.8.3 C/Z/D

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

Use columns 26, 33, or 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. When establishing record identifying codes, remember that many characters have either the same zone or the same digit portion.

7.8.4 Character

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

7.8.5 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.

7.8.6 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. Write the word OR in columns 14-15 to indicate an OR line.

INPUT SPECIFICATION

PROGRAM	PROGRAMMER		PAGES

PG NO	LINE NO		FILE NAME	SEQ	NUMBER OPTION	RECORD INDICATOR	POSITION	NOT	C.D.Z	CHAR	POSITION	NOT	C.D.Z	CHAR	POSITION	NOT	C.D.Z	CHAR	DECIMAL FORMAT	DECIMAL POSITIONS	FIELD LOCATION START	FIELD LOCATION END	FIELD NAME	CONTROL LEVEL	MATCHING FIELDS	FIELD RECORD RELATION	FIELD INDICATORS +	FIELD INDICATORS -	FIELD INDICATORS # OR blank	PROGRAM IDENTIFICATION
	01	I	INFILE	AA		01	22		CT		23		D2		24	N	Z													
	02	I		OR			22		CT		23		D3																	
	03	I																												
	04	I																												
	05	I	INFILE2	BC		02	01		CA		02		CB		03		CC													
	06	I		AND			04		CD		05		CE																	
	07	I																												

Figure 7-2. Example of record identification line in Input Specification.

7.9 Column 42

Column 42 is not used and should be left blank.

7.10 Column 43 (Packed or Binary Field)

Entry	Explanation
Blank	Field is in IBM-compatible decimal format, or is alphameric.
-	Field is in Datapoint-compatible decimal format.

Column 43 is used to indicate that a numeric field is in Datapoint-compatible format. Fields in this form will be converted to IBM-compatible form for use within the RPG II program.

An array which is read in Datapoint format should have an entry in column 43 of the Input Sheet. In this case the From and To columns of the Input Sheet should define the position the array occupies in the record. The array element length is defined on the Extension Sheet.

7.10.1 IBM Compatible Format

In this format, numeric input data is represented without an explicit decimal point and with the sign superimposed over the right-most digit. Leading zeroes may be replaced with blanks. This is the form RPG normally processes.

7.10.2 Datapoint Compatible Format

In this format numeric input data must be represented with an explicit decimal point unless the field contains no decimal positions. In this case the decimal point may or may not be present. If the number is negative, the character preceding the first digit (or the decimal point) must be a minus sign. Since RPG does not internally store the decimal point, the internal field assigned is one byte less than the external field size with the same number of decimal positions.

INPUT FIELD	SIZE	INTERNAL FIELD	SIZE
A. 1234.	5.0	1234	4.0
B. -123.	5.0	012L	4.0
C. -1234	5.0	123M	4.0
D. 01234	5.0	1234	4.0
E. 12345	5.0	illegal	4.0
F. 12.34	5.2	12^34	4.2
G. -1.23	5.2	01^2L	4.2

Caret (^) implies assumed decimal point.

Example E shows that since the field contains five digits (no decimal point or sign is present), it cannot be converted to a four digit internal number. An attempt to read in a field larger than will fit will cause an error message to be displayed.

7.11 Columns 44-51 (Field Location)

Entry	Explanation
Two numbers of 1-4 digits	Beginning (From) and end (To) of a field.

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 less than the number entered in To (columns 48-51).

It is not necessary that the From and To columns specify a whole array. A portion of an array may be read in; however, the array will be read in from element 1 up to as many elements as will fit in the numbers specified in the From and To columns.

DATABUS compatible input fields must take at least two positions.

The maximum field length for a numeric field is 15 positions. The maximum field length for an alphanumeric field is 256 characters.

Entries in these columns must end in columns 47 and 51. Leading zeros may be omitted.

7.12 Column 52 (Decimal Positions)

Entry	Explanation
Blank	Alphanumeric 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 positions, enter a 0. If a field is to be used in arithmetic operations or is to be edited, it must be numeric. The number of decimal positions must be less than or equal to the field length.

7.13 Columns 53-58 (Field Name)

Entry	Explanation
1-6 alphanumeric characters	Field name, array name, or array element.
PAGE	Special word.

Use columns 53-58 to name a field, array, or array element found on the input records. If you are referencing an array, additional entries may be needed in these columns. Use this name throughout the program whenever this field is referred to. Indicate the names of the fields for all types of records. However, you need name only the fields that are used.

7.13.1 Field Names

A field name can be from one to six characters long, must begin in column 53, and must be a valid RPG II name.

All fields in one type of record should have different names. If two or more fields of 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. Storage is only reserved for each unique field name defined.

Fields that are used in arithmetic operations or fields that are edited or zero suppressed (see Column 38 and Columns 45-70 in Chapter 9) 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.

7.13.2 Field Names in 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.

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.

2. Two or more record types have some fields which are identical and some fields which differ in location, length, or type of data.

Write the word OR in columns 14 and 15 to indicate an OR line. If there are several AND or OR lines, field description lines start after the last record identification line.

7.13.3 Special Word 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.

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 (up to 15

positions), but must have zero decimal positions specified. 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.

7.14 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 or demand 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 and 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 selected, the data in the control field is compared with data in the same control field from the previously selected record. When a control break occurs, the control level indicator turns on. Operations conditioned by the control level indicators are then done before processing the record which caused the control break, since that record is the first record of a new control group.

7.14.1 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.

7.15 Columns 61-62 (Matching Fields)

Entry	Explanation
M1-M9	Any matching level.

Use columns 61-62 to specify matching fields and sequence checking.

An entry in columns 61-62 indicates:

1. Matching fields and sequence checking when you have two or more input or update files with match fields.

2. Only sequence checking when you have just one input or update file.

7.15.1 Matching Fields

Make an entry in columns 61-62 when you wish to compare records from two or more input or update 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. Matching is allowed with primary and secondary files only.

7.16 Columns 63-64 (Field Record Relations)

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.
H1-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.

4. When used with match or control fields, the field record relation indicator must match a record identifying indicator for this file.

5. When any match value (M1-M9) is specified without field record relation, all match values used must be specified once without field record relation. If all match fields are not common to all records, a dummy match field should be used.

7.16.1 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, there must be some way of relating a field to a certain record. To do this, place in columns 63-64 the record type in which the field is found.

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, U1-U8, or MR entry in columns 63-64.

7.16.2 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.

7.16.3 External Indicators (U1-U8)

These columns may also be used to condition a specification by an external indicator (U1-U8). The external indicator, which is set prior to processing, conditions whether a field is to be used in the program. When the indicator is on, the field is read; when the indicator is off, the field is not read.

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.

7.16.4 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 of the input record specifications.

7.17 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 to test a field for a condition of either plus, minus, zero, or blank. The indicator specified turns on if the condition is true for the input record; it remains off or turns off if the condition is not true for the input record. These indicators may then be used to control certain calculation or output operations.

The three conditions which may be checked 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 alphanumeric field is all blanks.

A numeric field which is all blanks will turn on an indicator specified for all zeros. However, if an alphanumeric field is all zeros, the field will NOT turn on an indicator specified for all blanks.

7.17.1 Halt Indicators

Specify any halt indicator (H1-H9) in columns 65-70 to check for an error condition in your data. For example, if a field should not be zero, you may 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.

7.18 Columns 71-74

These columns are not used and should be left blank.

7.19 Columns 75-80 (Program Identification)

See Chapter 2.

INPUT SPECIFICATION

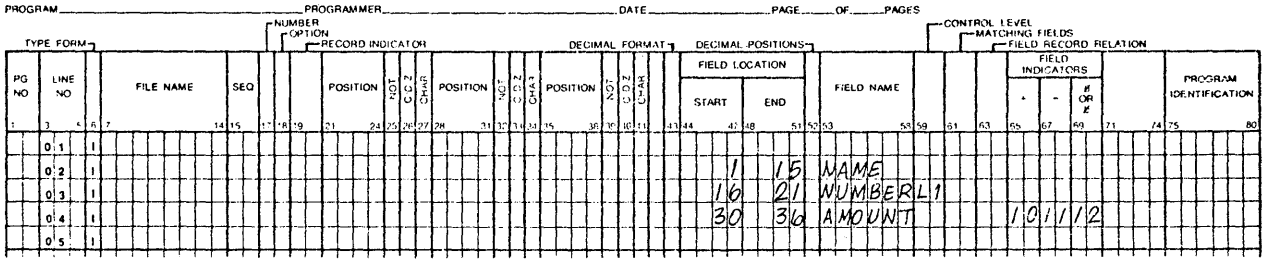


Figure 7-3. Example of field specifications on Input Specification.

INPUT SPECIFICATION

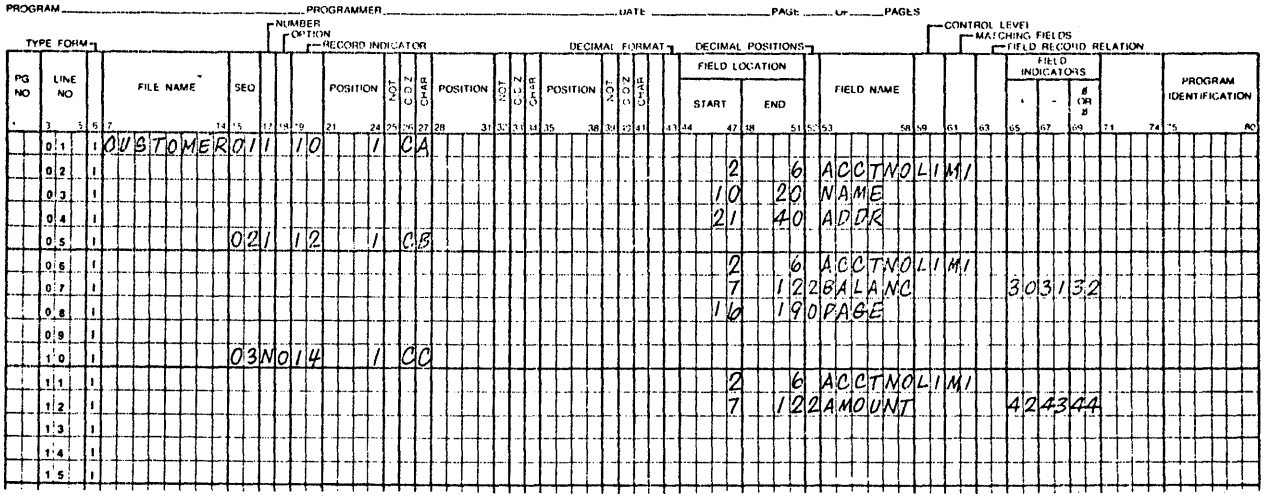


Figure 7-4. Example of Input Specification.

CHAPTER 8. CALCULATION SPECIFICATIONS

Calculation specifications describe the calculations to be performed on the data and the order in which they are to be performed. Each calculation specification can be divided into three parts that indicate:

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 (column 28-32) is to be performed on the data in columns 18-27 and/or columns 33-42. 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, and, if applicable, where the result is to be placed.

3. What tests are to be made on the results of the operation (columns 54-59). The indicators entered here signal the nature of the result of the operation and may serve to condition other operations.

8.1 Columns 1-2 (Page) and 3-5 (Line)

See Chapter 2.

8.2 Column 6 (Form Type)

A C must appear in column 6.

8.3 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 or when an

indicator is set on (L0 is always on).

LR	Calculation operation is done after the last record has been processed or after the LR indicator has been set on by a SETON operation.
SR	Calculation operation is part of a subroutine.
AN, OR	Establishes AND and OR relationships between lines of indicators.

If columns 7-8 are blank, the operation specified on the same line is done every time a record is read, provided indicators in columns 9-17 of that line or AN/OR lines associated with that line allow it.

Calculations must be specified in the following order:

1. Detail (blank in columns 7-8).
2. Total (L0 or L1-L9 in columns 7-8).
3. Last record (LR in columns 7-8). LR calculations must appear after L1-L9 calculations.
4. Subroutine (SR in columns 7-8).

AN/OR lines can appear within any of the above calculations. When AN/OR lines are used, the operation and its operands are entered on the last line of the AN/OR group.

8.4 Columns 9-17 (Indicators)

Entry	Explanation
Blank	Operation is performed for every record read unless columns 7-8 contain L0, L1-L9 or SR.
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. From one to three indicators may be used on a line. By using AN or OR entries in columns 7-8, many indicators can be used to condition one operation.

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 (columns 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.

Unless otherwise specified, any calculation operation may be conditioned by indicators.

8.5 Columns 18-27 and Columns 33-42 (Factor 1 & 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 which can be used are:

1. The name of any field that has been defined.
2. Any alphanumeric or numeric literal.
3. Any subroutine, table or array name, or an array element.
4. Any data field names (UDATE, UMONTH, UDAY, UYEAR).
5. The special name, PAGE.
6. A label or a TAG, BEGSR, or ENDSR operation (Factor 1 only).
7. A filename for a CHAIN, DEBUG, DSPLY, READ, 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 depend 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.

8.6 Literals

A literal is the actual data used in an operation rather than the field name representing that data. A literal may be either alphanumeric or numeric.

1. Any combination of characters may be used in an alphanumeric literal. Blanks are also valid.

2. Alphanumeric literals must be enclosed by apostrophes (').

3. The maximum length of an alphanumeric literal is eight characters excluding the two enclosing 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. Alphanumeric literals may not be used for arithmetic operations.

Consider the following rules when using a numeric literal:

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 (').

8.7 Columns 28-32 (Operation)

Use columns 28-32 to specify the kind of operation to be performed using Factor 1, Factor 2, and/or the Result Field and resulting indicators. The operation code must begin in column 28. A special set of operation codes have been defined which must be used to indicate the type of operation desired. Every operation code used requires certain entries on the same specification line. For further information on the operations that can be performed, see Operation Codes in this chapter.

The operations are performed in the order specified on the Calculation Sheet.

All operations conditioned by control level indicators in columns 7-8 must follow those that are not conditioned by control level indicators. All operations which are part of a subroutine (SR in column 7-8) must follow all other calculations in a program.

8.8 Columns 43-48 (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 on extension specifications, input specifications, or elsewhere in the calculation specifications.

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 alphanumeric. A field used in arithmetic operations or numeric compares, 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.

8.9 Columns 49-51 (Field Length)

Entry	Explanation
Blank	Alphanumeric or numeric field described elsewhere.
1-256	Result Field length.

Use columns 49-51 to give the result field length for any result field. 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, an error during execution results.

Numeric fields have a maximum length of 15 characters. Alphanumeric 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.

8.10 Column 52 (Decimal Positions)

Entry	Explanation
Blank	Alphanumeric or numeric field described elsewhere.
0-9	Number of decimal places in a numeric result field.

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 zero.

This column must be left blank if the result field is alphanumeric. It may also be left blank if the result field is numeric but has been previously described in the Extension, Input, or Calculation Specifications. In this case, Field Length (columns 49-51) must also be blank.

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 specified is smaller than the number that results from the operation, the rightmost digits (lowest order, or least significant, decimal places) are dropped.

8.11 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). In essence, 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.

The half adjust entry is allowed only with arithmetic operations. This entry cannot be specified for a DIV operation followed by an

MVR operation.

8.12 Columns 54-59 (Resulting Indicators)

Entry	Explanation
01-99	Any numeric indicator.
H1-H9	Any halt indicator.
L1-L9	Any control level indicator.
LR	Last record indicator.
OA-OG, OV	Any overflow indicator (if specified on File Description Sheet).

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.

3. To specify which indicators to SETON or SETOF.

4. To indicate end of file for the READ operation code.

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. 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 can test for more than one of the conditions.

8.12.1 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 table or array lookup operation.
4. The results of a CHAIN (not found), TESTB (all 0's), or TESTZ operation (for characters '&' and 'A-I').

8.12.2 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 table or array lookup operation.
4. The results of a TESTB (mixed), or TESTZ operation (for characters '}','-', and 'J-R').

8.12.3 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 1 in a table or array lookup operation.
4. The results of a READ (end of file), TESTB (all ones), or TESTZ operation (neither of the preceding).

8.13 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.

8.14 Columns 75-80 (Program Identification)

See Chapter 2.

8.15 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.

8.16 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 digits. If the result exceeds 15 digits, digits may be dropped from the right end of the fractional part of the result. Too many digits in the integer part of the result causes an

execution error diagnostic. The results of all operations are signed (+, -). Any data placed in the result field replaces the data that was there previously.

8.16.1 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.

8.16.2 Zero and Add (Z-ADD)

Factor 2 is added to a field of zeros, and the sum is placed in the Result Field.

8.16.3 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.

8.16.4 Zero and Subtract (Z-SUB)

Factor 2 is subtracted from a field of zeros. The difference is placed in the Result Field. This actually places the negative of Factor 2 in the Result Field. This operation can be used to change the sign of a field. Factor 1 is not used.

8.16.5 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 the Result Field, you must be sure the Result Field is large enough to hold the product.

8.16.6 Divide (DIV)

Factor 1 (dividend) is divided by Factor 2 (divisor). The result (quotient) 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 and a halt code is displayed. If 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).

8.16.7 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 and should be conditioned by the same indicators. The maximum length of the remainder is 15, including decimal positions. The number of significant decimal positions is the greater of:

1. The number of decimal positions in Factor 1 of the previous divide operation.

2. The sum of the decimal positions in Factor 2 and the Result Field of the previous divide operation.

The maximum whole number positions in the remainder is equal to the whole number positions in Factor 2 of the previous divide operation.

8.16.8 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. Factor 1 is not used.

Factor 2 and the Result Field can be numeric fields up to fifteen digits long overall, including up to nine decimal places.

For every digit left of the decimal place in the Result Field,

there should be two digits left of the decimal place in Factor 2; for every digit right of the decimal place in the Result Field, there should be two digits right of the decimal place in Factor 2.

A whole array can be used in a SQRT operation if Factor 2 and Result Field contain array names. In this case, the square root of each element of the array named in Factor 2 will be placed in the corresponding element of the array named in the Result Field.

When using the SQRT operation, remember:

1. The Result Field (root) is automatically half-adjusted.

2. The Result Field length must be greater than or equal to the decimal positions entry.

3. Factor 2 cannot be a negative number. A negative number causes a halt.

8.16.9 Crossfoot (XFOOT)

This operation is used only on arrays with numeric elements. 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. You can half-adjust the total in the Result Field and use resulting indicators if you wish.

If the Result Field is an element of the same array used in Factor 2, the value of that element prior to the XFOOT operation is used in arriving at a total.

CALCULATION SPECIFICATIONS

PROGRAM _____ PROGRAMMER _____ DATE _____ PAGE _____ OF _____ PAGES

FORM TYPE	PG NO	LINE NO		CONTROL LEVEL / INDICATORS	FACTOR 1	OPERATION	FACTOR 2	RESULT FIELD	FIELD LENGTH	DECIMAL POSITIONS	HALF ADJUST	RESULTING INDICATORS: COMPARE HI 1>2	LO 1<2	EQ 1=2	COMMENTS	PROGRAM IDENTIFICATION
		01	C		VALUEA	ADD	VALUEB	VALUEC	5	0						
		02	C		VALUEC	SUB	VALUED	VALUEE	5	0						
		03	C			Z-ADD	3.5	VALUEF	3	1						
		04	C			Z-SUB	10	VALUEG	3	0						
		05	C		VALUEF	MULT	VALUED	VALUEH	8	1	H					
		06	C		VALUEC	DIV	VALUEG	VALUEI	8	3						
		07	C			MVR		VALUEJ	3	0						
		08	C			SQRT	VALUEK	VALUEL	4	2						
		09	C			XFOOT	ARRAY1	TOTAL	10	2						
		10	C													
		11	C													
		12	C													
		13	C													
		14	C													
		15	C													
		16	C													
		17	C													
		18	C													
		19	C													
		20	C													
		21	C													
		22	C													
		23	C													
		24	C													
		25	C													
			C													
			C													
			C													
			C													
			C													

Figure 8-1. Example of arithmetic operations.

8.17 Move Operations

Move operations move part or all of Factor 2 to the Result Field. Factor 2 remains unchanged. 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 alphanumeric fields and alphanumeric fields may be changed to numeric fields by the move operations. To change a numeric field to an alphanumeric field, place the name of the numeric field in Factor 2 and use an alphanumeric result field. To change an alphanumeric field to a numeric field, place the name of the alphanumeric field in Factor 2 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.

Note: Databus-format numeric fields are stored within RPG with the sign superimposed over the low-order digit (see internal field examples in Chapter 7, Column 43).

8.17.1 Move (MOVE)

This operation causes characters from Factor 2 to be moved to the rightmost positions in the result field. Moving starts with the rightmost character.

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 alphanumeric field or constant may be changed 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 alphanumeric 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 alphanumeric field by moving it into an alphanumeric field. All digits are transferred. In addition the zone of the rightmost character is transferred with its digit.

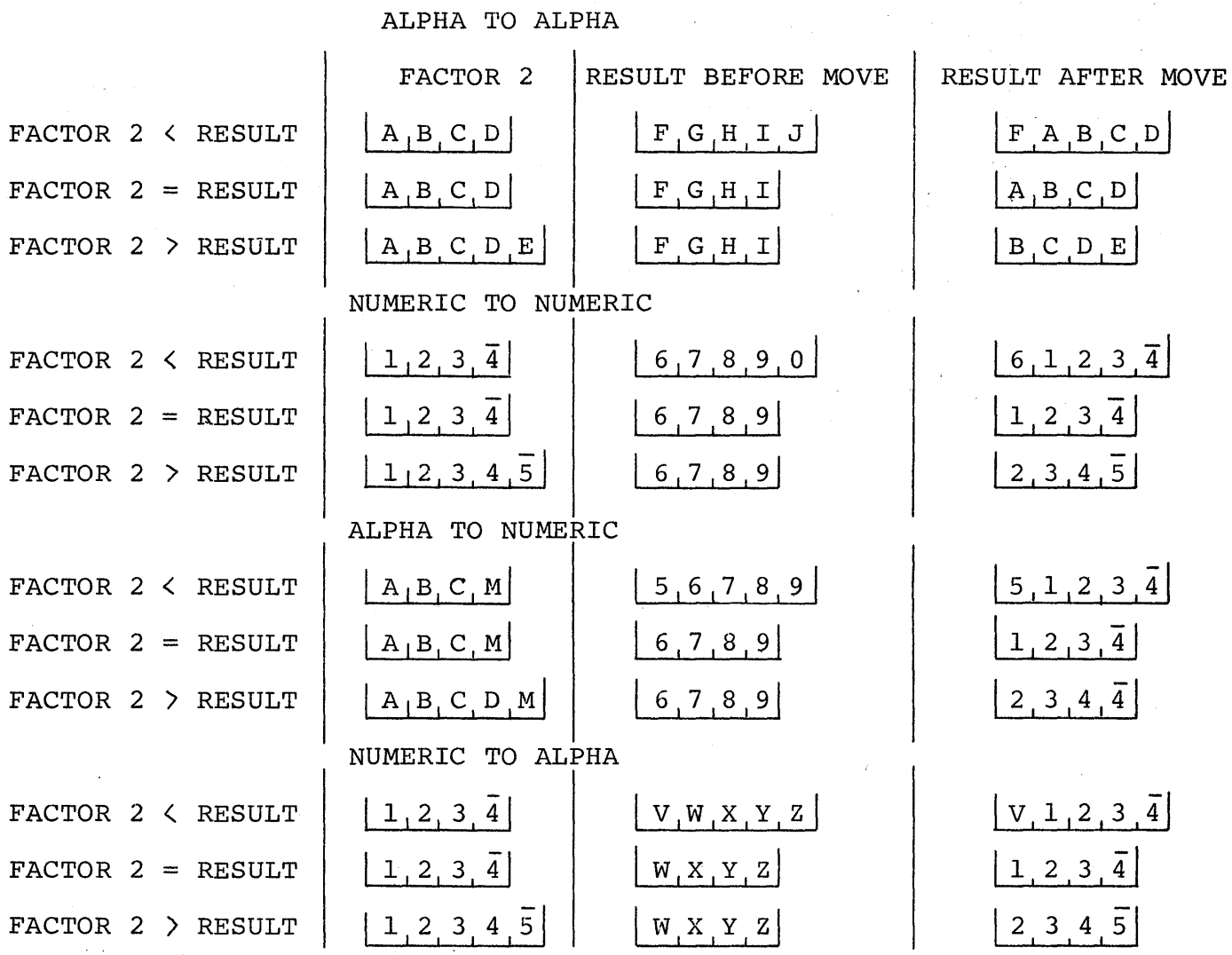


Figure 8-2. Diagram of MOVE instruction.

8.17.2 Move Left (MOVEL)

This operation causes characters from Factor 2 to be moved to the leftmost position in the Result Field. Moving begins with the leftmost character.

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 alphanumeric 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 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 alphanumeric field by moving it into an alphanumeric field. All digits are transferred. Both digit and zone portions of the rightmost character are transferred if that character is to be moved.

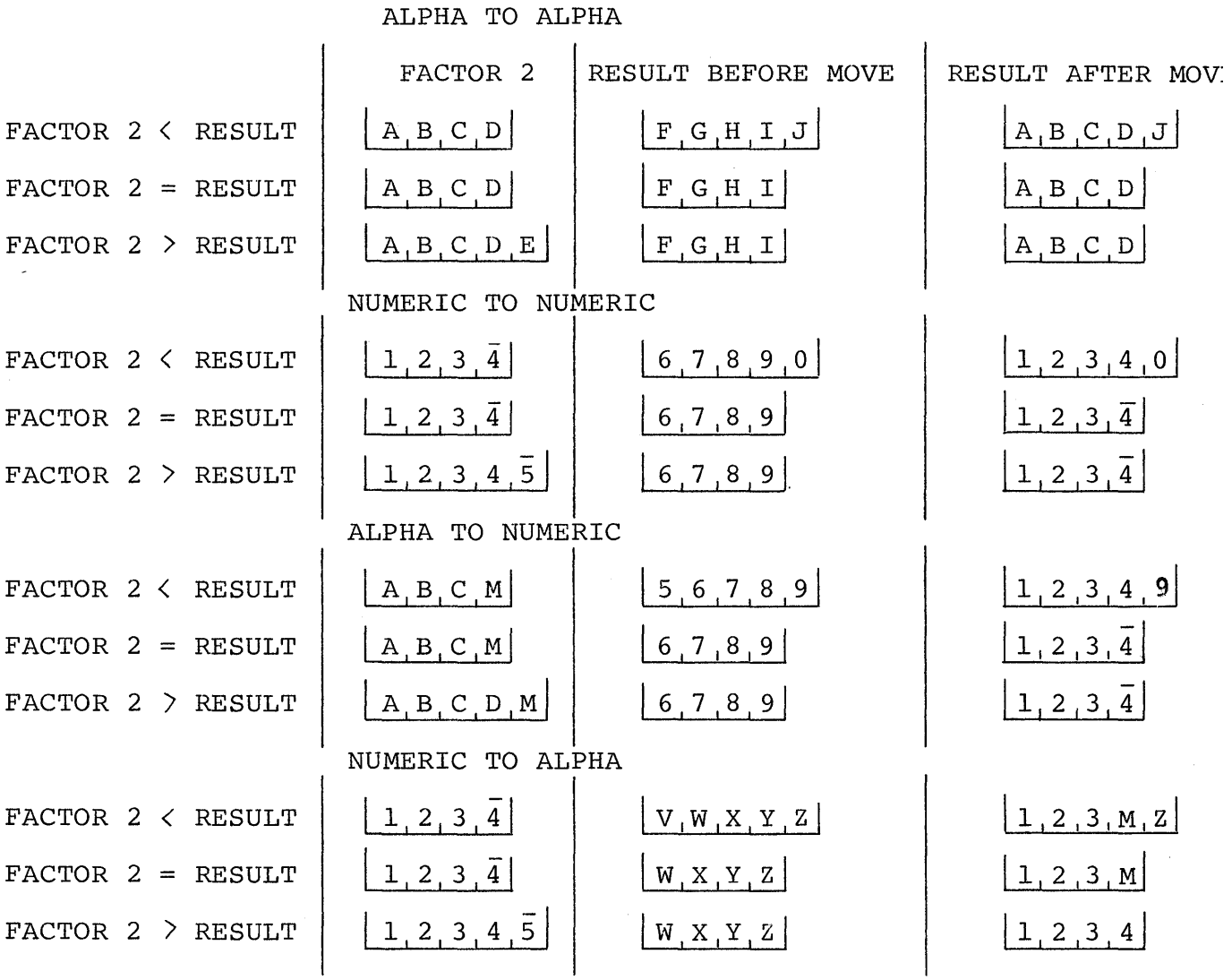


Figure 8-3. Diagram of MOVEL instruction.

8.17.3 Move Array (MOVEA)

The move array operation, valid ONLY in RPGPLUS, moves characters from Factor 2 to the Result Field, starting at the left-most position of each operand. The number of characters moved is the smaller of the lengths of Factor 2 and the Result Field. If Factor 2 is longer than the Result Field, the right-most characters of Factor 2 are not moved. If Factor 2 is shorter than the Result Field, the right-most characters of the Result Field are left unchanged. Both operands must be described as alphanumeric.

With MOVEA it is possible to move multiple contiguous elements of an array, a whole array, or a field to multiple contiguous elements of an array, a whole array, or a field. If Factor 2 or the Result Field contain the name of an array, the move starts at the first element of the array. If they contain an indexed array element, the move starts at the element specified. The MOVEA operation terminates when the end of the shorter operand is reached. In the case of MOVEA with arrays, this may be in the middle of an array element.

8.18 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.

Note: Generally, whenever the word high is used in a move zone operation, the field involved must be alphanumeric; whenever low is used, the field involved may be either alphanumeric or numeric.

8.18.1 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 alphanumeric.

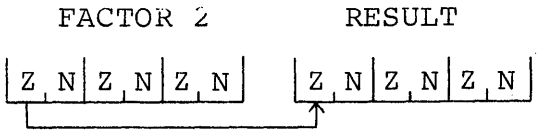


Figure 8-4. Diagram of MHHZO instruction.

8.18.2 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 alphanumeric. The Result Field may be either alphanumeric or numeric.

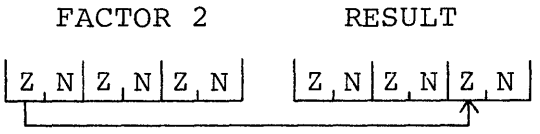


Figure 8-5. Diagram of MHLZO instruction.

8.18.3 Move Low to Low Zone (MLLZO)

This operation moves the zone from the rightmost position of Factor 2 to the rightmost position to the Result Field. Factor 2 and the Result Field may be either alphanumeric or numeric.

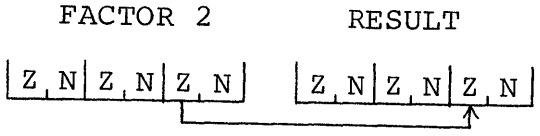


Figure 8-6. Diagram of MLLZO instruction.

8.18.4 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 alphanumeric, but the Result Field can only be alphanumeric.

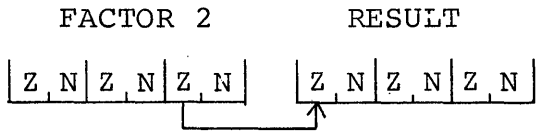


Figure 8-7. Diagram of MLHZO instruction.

8.19 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.

8.19.1 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 alphanumeric or both numeric. A field may be compared to another field or a literal.

The fields are automatically aligned before they are compared. If the fields are alphanumeric, they are aligned to their leftmost character. If one is shorter, the unused positions are filled with blanks.

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. The maximum field length for numeric fields which are to be compared is 15 digits.

NUMERIC COMPARE

	VALUE	COMPARE VALUE
FACTOR 1	123.45	123.450
FACTOR 2	42.763	042.763

ALPHANUMERIC COMPARE

FACTOR 1	ABC	ABCØØ
FACTOR 2	VWXYZ	VWXYZ

Figure 8-8. Diagram of COMP instruction.

8.19.2 **Test Zone (TESTZ)**

This operation tests the zone of the leftmost character in the result field (see ASCII to EBCDIC Translation Table in Appendix L). The Result Field must be alphanumeric since this operation can be done only on alphanumeric characters. Resulting indicators are used to determine the results of the test. The zone portion of characters & and A-I causes the plus indicator to turn on. The zone portion of the characters } (bracket), - (minus), and 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.

8.20 **Binary Field Operations**

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

In binary field operations, the operation code, BITON, BITOF, or TESTB must appear in columns 28-32. Factor 2 can contain:

Bit number 0-7: One or more bits (maximum of eight) may be set on, set off, or tested per operation. The bits are numbered from left (most significant) to right (least

significant) and are enclosed in apostrophes. The order of specification of the bits is not restricted. For example, to specify the first bit in a field, enter '0' in Factor 2 (columns 33-35). To specify bits 0, 2, and 5, enter '025' in Factor 2 (columns 33-37). Bits not specified in Factor 2 are not changed.

Field Name: The name of a one-position, alphanumeric field or table or array element can be entered. In this case, the bits which are on in the field or array element are set on, set off, or tested in the Result Field; bits which are not on are not affected.

Any field named in Factor 2 or the Result Field must be a one-position, alphanumeric field (no entries in the decimal positions columns on the Input or Calculation Sheet).

8.20.1 **Set Bit On (BITON)**

This operation code causes bits identified in Factor 2 to turn on (set to one) in a field named as the Result Field. The operation code BITON must appear in columns 28-32. Conditioning indicators can be used in columns 7-17. Any entry under Field Length must be 1. See the preceding discussion in Binary Field Operations.

Factor 1, Decimal Positions, Half-adjust, and Resulting Indicators are not used with the BITON operation.

8.20.2 **Set Bit Off (BITOF)**

This operation code causes bits identified in Factor 2 to turn off (set to zero) in a field named as the Result Field.

The operation code BITOF must appear in columns 28-32. All other specifications are the same as those for the BITON operation.

8.20.3 **Test Bit (TESTB)**

This operation code causes bits identified in Factor 2 to be tested for an on or off condition in the field named as the Result Field. The condition of the bits is known by resulting indicators in columns 54-59. All other specifications are the same as those for BITON and BITOF.

At least one resulting indicator must be used with the TESTB operation; as many as three can be named for one operation. Two indicators may be the same for one TESTB operation, but not three. If Factor 2 contains bits which are all off, no resulting indicators are turned on. A resulting indicator has the following meanings for these columns:

Columns 54-55: An indicator in these columns is turned on if each bit specified in Factor 2 is off (0) in the Result Field.

Columns 56-57: An indicator in these columns is turned on if two or more bits were tested and found to be of mixed status; that is, some bits on and other bits off. It is the programmer's responsibility to ensure that the field named in Factor 2 contains more than one bit which is on if an indicator appears in columns 56-57.

Columns 58-59: An indicator in these columns is turned on if each bit specified in Factor 2 is on (1) in the Result Field.

8.21 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, 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 after halting once for every halt indicator that is on.

5. Setting on or setting off a control level indicator (L1-L9)

does not automatically set on the lower control level indicators.

6. Indicators L1-L9 and the record identifying indicators are always turned off after detail output operations are completed, regardless of the previous SETON or SETOF operation.

7. Whenever a new record is read, record identifying indicators are set to reflect conditions on the new record. The setting from any previous SETON or SETOF operation does not apply then.

8.21.1 Set On (SETON)

This operation causes any indicators in columns 54-59 to be turned on.

8.21.2 Set Off (SETOF)

This operation causes any indicators in columns 54-59 to be turned off.

8.22 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.

8.22.1 Go To (GOTO)

This operation allows you to skip instructions by specifying some other instruction to go to (see TAG). You may branch to an earlier line or to a later specification line. However, you cannot skip from a calculation that is conditioned by a control level indicator (columns 7-8) to one that is not. Neither can you branch from a calculation within a subroutine to a calculation outside of that subroutine, or vice versa.

When using the GOTO command to transfer control from a detail calculation to a total calculation, care must be taken, to avoid creating an infinite loop. (See Appendix K for normal sequencing.) The 'L' indicators will not be implicitly set when such a jump is performed.

Factor 2 must contain the name of the point to which you wish to go. 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.

8.22.2 Tag (TAG)

This 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.

Factor 2 and the Result Field are not used. No indicators may be entered in columns 9-17 for a TAG instruction. A control level (see Columns 7-8) may be used, however, if the TAG occurs in total calculations.

8.23 Lookup Operations

Lookup operations are used when searching through a table or an array to find a special element.

8.23.1 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 alphanumeric 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 54-55) 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 satisfies 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 and the same format (alphanumeric or numeric), but need not have the same alignment.

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.

8.23.2 Using 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.

8.23.3 Using 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 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, the search

stops at the end of the shorter table.

8.23.4 Referencing the Table Item Found

Whenever a table name is used in an operation other than LOKUP, the table name really refers to the data placed in the special 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 storage area are 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 storage area are 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.

8.23.5 Using 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, the indicators reflect only that the desired item is in the array; the programmer does not have ready access to this item.

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 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.

8.24 Subroutine Operations

These operation codes are only used for subroutines. All subroutine operation codes must be written in specification lines following all detail and total calculations. Subroutine lines are always identified by an SR in columns 7-8.

8.24.1 Begin Subroutine (BEGSR)

This operation code serves as the beginning point of the subroutine. Factor 1 must contain the name of the subroutine.

8.24.2 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.

8.24.3 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.

8.25 Programmed Control of Input and Output

The normal Datapoint RPG II processing cycle is as follows:

1. A record is read.
2. Calculations are performed.
3. Records are written.

The normal program cycle can be altered to allow input and output operations during calculations. The following operations provide this capability:

-- Exception (EXCPT)

-- Force (FORCE)

-- Display (DSPLY)

-- Read (READ)

-- Chain (CHAIN)

-- Set lower limit (SETLL)

8.25.1 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, or to write out contents of a table.

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.

8.25.2 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 or secondary, input or update files.

Factor 2 in a FORCE statement identifies the file from which the next record is to be selected. If the statement is executed, the record is selected 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 should not be specified at total time.

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. When end-of-file is encountered on a forced file, a record will not be

retrieved from the file; normal records selection will determine which record is to be processed.

8.25.3 Display (DSPLY)

The display operation allows either or both of the following:

1. A field, table element, array element, or literal up to 80 characters long is displayed on the keyboard-display during program execution without a program halt.

2. A field, table element, literal, or array element up to 80 characters long is displayed on the keyboard-display, and the program halts, allowing that field to be changed.

A literal may not be changed with display.

Factor 2 in a DSPLY statement identifies the file used for the display operation and must be defined with a D in column 15 of the File Description Specification. The device used must be the CONSOLE.

Fields Defined	Data Displayed	Keyboard
Factor 1	DSPLY <factor-1>	none
Result Field	DSPLY <result-field>	data to replace
Factor 1 and Result Field	DSPLY <factor-1> <result-field>	data to replace result-field

There are several points to remember if you wish to enter data during program execution:

1. Numeric data must be entered in Databus format. To key a negative field, the minus sign is keyed and then the field is keyed. The entry will be automatically aligned on its decimal point before it is stored in the result field. The result field must be defined with one extra digit position to the left of the decimal point if a sign is to be entered.

2. Alphanumeric result fields will be left-justified after all characters are keyed. If the number of characters entered is

less than the result field size the field is right filled with blanks.

3. If no characters are entered or the space bar is not depressed, the result field will not be changed.

4. Numeric fields are displayed in Databus format. The field definition must contain enough integer positions to allow room for a minus sign to the left of the first significant digit or a NUMERIC FIELD ERROR may occur when displaying negative numbers.

8.25.4 Read (READ)

The READ operation is used to call for immediate input from a demand file during the calculations in the program cycle. This operation differs from the FORCE operation because FORCE specifies input on the next program cycle, not the present one. The READ operation is similar to the CHAIN operation, except that the READ file is processed sequentially and the CHAIN file is processed randomly.

The operation code READ must appear in columns 28-32. Factor 2 contains the name of the file from which a record will be read immediately. An indicator should be used in columns 58-59. An indicator specified in these columns will turn on after a READ operation in which an end-of-file condition is reached. An attempt to read past end-of-file will cause an error message to be displayed. If columns 58-59 are blank, a halt will occur on an end-of-file condition and on subsequent READ operations after the end-of-file condition is reached. Indicators may be specified in columns 7-17.

Note: When the program is reading from several demand files during the same RPG II cycle, record identifying indicators assigned to the demand files will remain on throughout the cycle if the previous READ operations were executed successfully.

The following files can appear as Factor 2 in a READ operation (all must be designated demand files with a D in column 16 of the File Description Sheet):

Files processed consecutively and specified as input or update files.

Console files specified as input files.

When using the READ operation for demand files remember these points:

1. Demand files can only be processed by the READ operation.

2. Control levels, matching fields, and look-ahead fields are not allowed with demand files.

3. Numeric sequence testing on the Input Sheet is not allowed for demand files.

4. The MR indicator may not be entered in columns 63-64 (Field Record Relation) on the Input Sheet.

5. When a demand file is conditioned by a U1-U8 indicator which is not on, no records will be read from that file and the end-of-file indicator (in columns 58-59 of the Calculations Sheet) will not turn on.

8.25.5 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 43-53) must be blank. File conditioning indicators (U1-U8) can be used to condition a chained file.

Factor 1 must contain:

1. Relative record number of record to be read.

2. Key of indexed file record to be read.

Relative record number must be a numeric field with no decimal positions. Relative record numbers start from 1.

Keys must be alphanumeric fields. If the length is not the same as keys in the file, the shorter key will be extended with blanks for comparison.

Factor 2 must contain the name of a CHAIN file.

Columns 54-55 should contain an entry. If the record is not

found, the indicator specified in these columns will turn on. Columns 56-59 must always be blank for chain operations.

If an indicator is not specified in columns 54-55, and the record is not found, the program will halt and display a chaining error message. The options given are to end the job or bypass the remainder of the current cycle and begin a new cycle. If LR processing has already been initiated, the bypass-and-begin-new-cycle option is not allowed. If the controlled cancel option is taken, files are closed, but the rest of the LR processing does not occur. When a record is beyond the range of the file, the options to end the job or bypass the remainder of the current cycle are given.

CHAINING RECORD PROCESSING

FILE TYPE	RECORD PRESENT	ADD FILE	CHAIN INDICATOR	ACTION
DIRECT INPUT	Yes		Off	A
	No*		On	B
	No			1
DIRECT OUTPUT	Yes		Off	D
	No*		Off	E
	No		On	F
DIRECT UPDATE	Yes		Off	C
	No*		On	E
	No		On	F
INDEXED INPUT	Yes		Off	A
	No		On	B
INDEXED OUTPUT	Yes			2
	No	Yes	On	F
	No	No		1
INDEXED UPDATE	Yes		Off	C
	No	Yes	On	F
	No	No		1

*Fixed format

ACTION CODES:

A Record returned.

B Blank record returned.

C Existing record updated.

D Existing record overwritten.

E Record written in formatted disk space.

F File extended and new record written.

1 CHAINING ERROR.

2 DUPLICATE KEY error.

8.25.6 Set Lower Limits Operation (SETLL)

This operation allows the lower limit to be set during calculations when processing indexed files sequentially by key. It may be used with indexed input, update and demand files. When used with input and update files, care should be exercised to properly process records already read during the input cycle prior to executing calculations. The SETLL operation may be executed as many times as desired prior to reaching the end of the input file. SETLL will select a new starting record for input during the next input cycle.

Factor 1 must contain an alphanumeric field name or literal representing the value of the lower limit being set. The length of the field or literal does not have to be equal to the length of the key for the file named in Factor 2. The shorter key will be extended with blanks before comparison is made.

Factor 2 must contain the name of the file for which the lower limit is to be set. If a read is performed to the file prior to the first SETLL instruction the record with the lowest key in the file is read.

The Mode of Processing entry (column 28) in the File Description must contain an L for processing within limits.

8.26 Audio Output Operations

Operation codes are provided to allow audible signals to be given to alert the operator of conditions requiring operator intervention. These signals can also be used for debugging and timing.

8.26.1 Beep (BEEP)

The BEEP operation code causes the Datapoint computer to emit an audible beep.

8.26.2 Click (CLICK)

The CLICK operation code causes the Datapoint computer to emit an audible click.

8.27 Debug Operations

The debug operation is an RPG II function that may be used to help find errors in a program which is not working properly. This code causes one or more records to be written containing information helpful for finding programming errors.

8.27.1 Debug (DEBUG)

The DEBUG operation code may be placed at any point or at several points in the calculation operations. Whenever it is encountered, one or more records are written 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.

8.27.2 Debug Specifications

Factor 1 is optional. It may contain a literal or field name which identifies the particular debug operation. The literal or the value of the field named here is written 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, table element, array element, or whole array whose contents you want to write on record 2. Any valid indicators may be used in columns 7-17. Columns 49-59 must be blank.

Numeric fields are displayed in Databus format. The field definition must contain enough integer positions to allow room for a minus sign to the left of the first significant digit or a NUMERIC FIELD ERROR may occur when displaying negative numbers.

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 by the compiler.

8.28 EXIT and RLABL Operations

Linkage from Datapoint RPG II to Assembler language subroutines is accomplished through the RPG II EXIT and RLABL operations.

8.28.1 EXIT Operation

The EXIT operation code is used to designate a point in the RPG II calculation specifications at which control is to be passed to a pre-processed, external subroutine. The rules for use of the EXIT operation in RPG II calculation specifications are as follows:

1. Operation EXIT.

2. Factor 1 blank.

3. Factor 2 contains the name of subroutine to which control is to pass.

4. Result Field blank.

5. Resulting Indicators blank.

The EXIT operation can be conditioned by Control Level entries (columns 7-8) and Indicator entries (columns 9-17). If not conditioned by control level entries, the EXIT operation occurs at detail calculation time.

The EXIT operation generates a CALL to the subroutine named in Factor 2.

8.28.2 RLABL Specification

Through the RLABL operation, a field, table, or array defined in the RPG II program can be referenced by the subroutine to which the EXIT operation gives control. The rules for use of RLABL in RPG II calculation specifications are as follows:

1. Operation RLABL.

2. Result Field contains field, table, or array name.

3. Field Length contains the length of the field (optional).

4. Decimal Positions contains the decimal indication (optional).

The RLABL specifications must immediately follow the EXIT specifications for the subroutine which references the RPG II field. A name defined by a TAG, BEGSR, or ENDSR specification cannot be used in an RLABL specification.

8.28.3 Referencing Fields

If the result field of the RLABL refers to a field, a four-byte DC is generated containing: the number of decimal positions in the field or 0 (first byte), the field length (second byte), and the address (third and fourth bytes) of the left-most byte of the field. A numeric field has normal zones (octal values 360-371) over all positions except the last (right-most), which contains a character in the range 300 to 311 if the number is positive or a character in the range 320 to 321 if the number is negative. If the subroutine generates numeric results, the user should ensure that all zero values generated have a positive sign.

8.28.4 Referencing Tables and Arrays

If the result field of the RLABL refers to a table or array, the two-byte address of the Table Description Block (TDB) is generated. See Appendix C for its format.

8.28.5 Referencing Indicators

An assembler subroutine may reference indicators in the RPG II program to which it is linked. This is done by entering INxx in the Result Field of an RLABL specification. The xx represents the indicator to be referenced. For example, if MR is to be tested, INMR must be entered in the Result Field of the RLABL specification.

The object code generated is the two-byte address of the indicator. An indicator byte contains zero when the indicator is off. It is non-zero (and normally 377) when it is on.

Note: Two-byte addresses are generated in the standard order: least-significant-byte, then most-significant-byte.

CHAPTER 9. OUTPUT FORMAT SPECIFICATION

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.

2. Field description entries (columns 32-74) which indicate the position and the format of data on the output record.

Write the specifications on the Output Format Sheet. The field description entries start one line lower than record description entries.

9.1 Columns 1-2 (Page) and Columns 3-5 (Line)

See Chapter 2.

9.2 Column 6 (Form Type)

An O must appear in column 6.

9.3 Columns 7-14 (Filename)

Use columns 7-14 to identify the file to which records are to be written. 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.

9.4 Column 15 (Type)

Entry	Explanation
H	Heading records.
D	Detail records.
T	Total records.

E	Exception Records (records to be written during calculaton time).

Use column 15 to indicate the type of record that is to be written. Either enter the records for each file in this order: heading, detail, total, and exception, or enter all heading records for all output files, then, all detail records for all output files, etc.

9.5 Columns 16-18 (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 output or update file. The output device must be DISK and an A must be coded in column 66 of the File Description Specification for the file to which the record is to be added.

The entry in Columns 16-18 is optional and is not required for adding records to the file.

9.6 Column 16 (Fetch Overflow)

Entry	Explanation
Blank	Overflow not fetched.
F	Fetch overflow.

Column 16 may be used to indicate that the overflow routine can be used at this point for a printer file. 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.

The fetched overflow routine automatically causes forms to advance.

F may be used in an OR line if you want that line to condition a record with the overflow indicator.

9.7 Columns 17-22 (Space/Skip)

Columns 17-22 are used to specify spacing and line skipping for a printer file. 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. Only space before and space after can be specified on output for the display.

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.

9.7.1 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. If the destination of a space operation is a line beyond the overflow line (but not on a new page), the overflow indicator turns on and remains on until all overflow lines are printed.

Note: The display will always roll up one line before output. Therefore, a space before entry of blank, zero, or one will result in a single space before output.

9.7.2 Columns 19-22 (Skip)

Entry	Explanation
01-99	Lines 1-99.

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 (columns 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. If the destination of a skip operation is a line beyond the overflow line (but not on a new page), the overflow indicator is turned on and remains on until all overflow lines are printed. The destination line of a skip operation must not be beyond the form length defined on the Line Counter Specification.

9.8 Columns 23-31 (Output Indicators)

Entry	Explanation
01-99	Any resulting indicator, field indicator, or record identifying indicator previously specified.
L1-L9	Any control level indicators previously specified.

H1-H9	Any halt indicators previously specified.
U1-U8	Any external indicator set during program initialization.
OA-OG, OV	Any overflow indicator previously assigned to this file.
MR	Matching record indicator.
LR	Last record indicator.
1P	First page indicator.
L0	Level zero indicator.

Use output indicators to give the conditions under which output operations are to be done. When you use an indicator to condition an entire line of print, place it on the line which specified the type of record. Place an indicator which conditions when a field is to be printed on the same line as the field name.

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 should be conditioned by all negative indicators (at least one of the indicators used should be positive). If all negative indicators condition a heading or detail operation, the operation is performed at the beginning of the program cycle when 1P lines are written. The overflow indicators may not be specified on an E (exception output) line.

Warning: When defining records of update files, avoid writing multiple records on one cycle, since results are unpredictable.

In Datapoint RPG II, all total lines conditioned by LR will be performed last.

9.8.1 AND and OR Lines

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.

Output indicators may also be in an OR relationship. If either or both of the OR conditions are met, the output operation will be done. OR lines are indicated by the word OR in columns 14-15. Both AND or OR lines may be used together to condition an entire output line. AND and OR lines cannot be used to condition a field.

The use of an L0-L9 indicator in an OR relationship with an LR indicator can result in the specified operation being done twice when LR is on. One operation is performed during LR processing and the other at detail or total time.

9.8.2 External Indicators

A file named in the Output Format specifications may be conditioned by an external indicator in the file description specifications. External indicators can also be used to condition a record or field. No output can occur to a file if it is conditioned by an external indicator and that indicator is off.

9.8.3 Control Level Indicators

Control level indicators entered in columns 23-31 of this sheet specify when output records or fields 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).

9.8.4 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 has been passed.

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 overflow 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.

9.8.5 First Page Indicator

The first page (1P) indicator is usually used to allow printing on the first page. It may also be used in an OR relationship with the overflow indicator to allow printing on every page. 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, columns 45-70.

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. All lines conditioned by the 1P indicator are written out even before the first record from any input file is processed. Therefore, do not condition output fields (except PAGE and UDATE) which are based upon data from input records by the 1P indicator. Calculation operations cannot be conditioned by the 1P indicator.

9.9 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 the program.

2. The special words PAGE, 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 later fields overlap the first fields specified, the data which is overlayed is lost.

In IBM-compatible format, the sign (+ or -) of a numeric field is in the units position (rightmost digit). A minus sign in the units position prints as a letter unless the field is edited (see Column 38 and Column 44 in this chapter).

9.9.1 PAGE

PAGE is a special word which causes automatic numbering of pages. Enter the word PAGE in these columns if you wish pages (or an individual record) to be numbered. When a PAGE field is named in these columns without being defined elsewhere, it is assumed to be a four-position numeric field with no decimal positions.

However, a PAGE field can be defined in input or calculation

specifications and may be up to 15 positions long. A PAGE field, when explicitly defined, must be defined with zero decimal positions. Leading zeros are suppressed, and the sign is not printed in the rightmost position unless an edit word or edit code is specified. The page number starts with 1 unless otherwise specified, and one is automatically added each time the PAGE field is written.

It is possible at any point in the job to restart the page numbering sequence. To do this, set the PAGE 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. A PAGE field will always be printed even though the field is conditioned by an indicator. If the indicator is on, the PAGE field is set to zero, and one is added before it is written. Remember that one is always added to the PAGE field before it is written.

9.9.2 Date Field

Often you want the date to appear on the printed report or output record. Use special words UDATE, UMONTH, UDAY, and UYEAR to get the date field you desire. The following rules apply to the date field:

1. UDATE gives a six-character numeric date field in the format: mmddyy (d, m, and y are the day, month and year positions in the UDATE field).

   The edited date field is eight characters long, in the format: MM/DD/YY.

2. UDAY may be used for days only, UMONTH for months only, and UYEAR for years only.

3. These fields may not be changed by any operations specified in the program.

9.10 Column 38 (Edit Codes)

Use column 38 to:

1. Suppress leading zeros for a numeric field.

2. Omit a sign from the low order position of a numeric field.

3. Punctuate a numeric field without setting up an edit word.

A table summarizing the edit codes that can be used is printed below. Each edit code punctuates differently. If an edit code is used in column 38, columns 45-70 must be blank unless asterisk fill or a floating dollar sign is required ('*' or '$' entered in columns 45-47). If an edit code is used to punctuate an array, two spaces are left between elements of the array to the left of each element.

Normally, when an edit code is used in column 38, defining an edit word in columns 45-70 is not allowed; however, there are two exceptions:

1. If leading zeros are to be replaced by asterisks, enter '*' in columns 45-47 of the line containing the edit code.

2. If a dollar sign is to appear before the first digit in the field (floating dollar sign), enter '$' in columns 45-47 of the line containing the edit code.

Asterisk fill and floating dollar sign are not allowed with X, Y and Z edit codes.

It is also possible to have a dollar sign appear before the asterisk fill (fixed dollar sign). This is done in the following way:

1. Place a dollar sign constant one space before the beginning of the edited field (on another output specification line).

2. Place '*' in column 45-47 of the line containing the edit code.

Summary of Edit Codes

Commas	Zero 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

9.11 Column 39 (Blank After)

Entry	Explanation
Blank	Field is not to be reset (blanked or zeroed) after writing.
B	Field is to be reset (blanked or zeroed) after writing.

Use column 39 to reset a field to zeros or blanks. Numeric fields are set to zero and alphanumeric fields are set to blanks. This column must be blank for Look-Ahead fields, date fields (UDATE, UDAY, UMONTH, UYEAR), and constants.

Resetting fields to zeros is useful when accumulating totals for each control group. After finding the total for one group and printing it, you want to start accumulating and printing totals for the next group. However, the total field should always 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 to be used for output more than once, 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.

If a field name specified with Blank After is a table name, the element of the table looked up last will be blanked or zeroed.

9.12 Columns 40-43 (End Position in Output Record)

Use columns 40-43 to indicate the location on the output record of the field or constant that is to be written. Enter only the number of the ending position (the rightmost character) in the field or constant. Be sure to allow enough space (as indicated by end position entries) on the output record to hold edited fields.

9.13 Column 44 (Packed or Binary Field)

Entry	Explanation
Blank	Field is IBM-compatible numeric or alphanumeric data.
-	Field is Datapoint-compatible numeric data.

Column 44 should contain a D for output in Datapoint format. If used, the field must not be edited. Since a decimal point is automatically inserted when the field is written, an additional column must be allowed for the period. A minus sign will always be placed to the left of the most significant digit. There must be room for both a decimal point and a sign if the output field is negative.

Examples

Internal field	Size	Output field	Size
1234	4.0	1234.	5
012L	4.0	-123.	5
123M	4.0	illegal	5
001K	4.0	-12.	5

9.14 Columns 45-70 (Constant or Edit Word)

Use columns 45-70 to specify a constant or an edit word.

9.14.1 Constant

A constant is any unchanging information that is entered by a specification. Constants are usually words used for report headings or column headings.

The following rules apply to constants:

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 O'CLOCK appears as a constant it must be coded O''CLOCK.

4. 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.

9.14.2 Edit Word

An edit word gives more flexibility in punctuating a numeric field than an edit code. You directly specify whether commas, decimal points, and zero suppression are needed, whether the negative sign should print, whether the output is dollars and cents, and whether a dollar sign and leading asterisks are wanted. Constants can be used within edit words.

The following rules apply to edit words:

1. Column 38 (Edit Codes) must not be used.

2. Columns 32-37 (Field Name) must contain the name of a numeric field.

3. Columns 40-43 (End Position in Output Record) must contain an entry.

4. An edit word must be enclosed in apostrophes. Enter the

leading apostrophe in column 45. The edit word itself must begin in column 46.

5. Any printable character is valid, but certain characters in certain positions have special uses (see Editing Considerations in the following text).

6. An edit word cannot be longer than 24 characters.

7. The number of replaceable characters in the edit word must be equal to the length of the field to be edited. (See Editing Considerations in the following text for a discussion of replaceable characters.)

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

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

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

9.14.3 Editing Considerations

Always leave exactly enough room on the output file for the edited field. If the field to be edited is seven characters long on the input record, the possible insertion of editing characters may well require the output field length to be greater 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 (see following Note). The replaceable characters are:

Character	Use
0	Zero Suppression.
*	Asterisk fill.
Blank	Blank.

$ Floating dollar sign (if it appears immediately to the left of zero suppression).

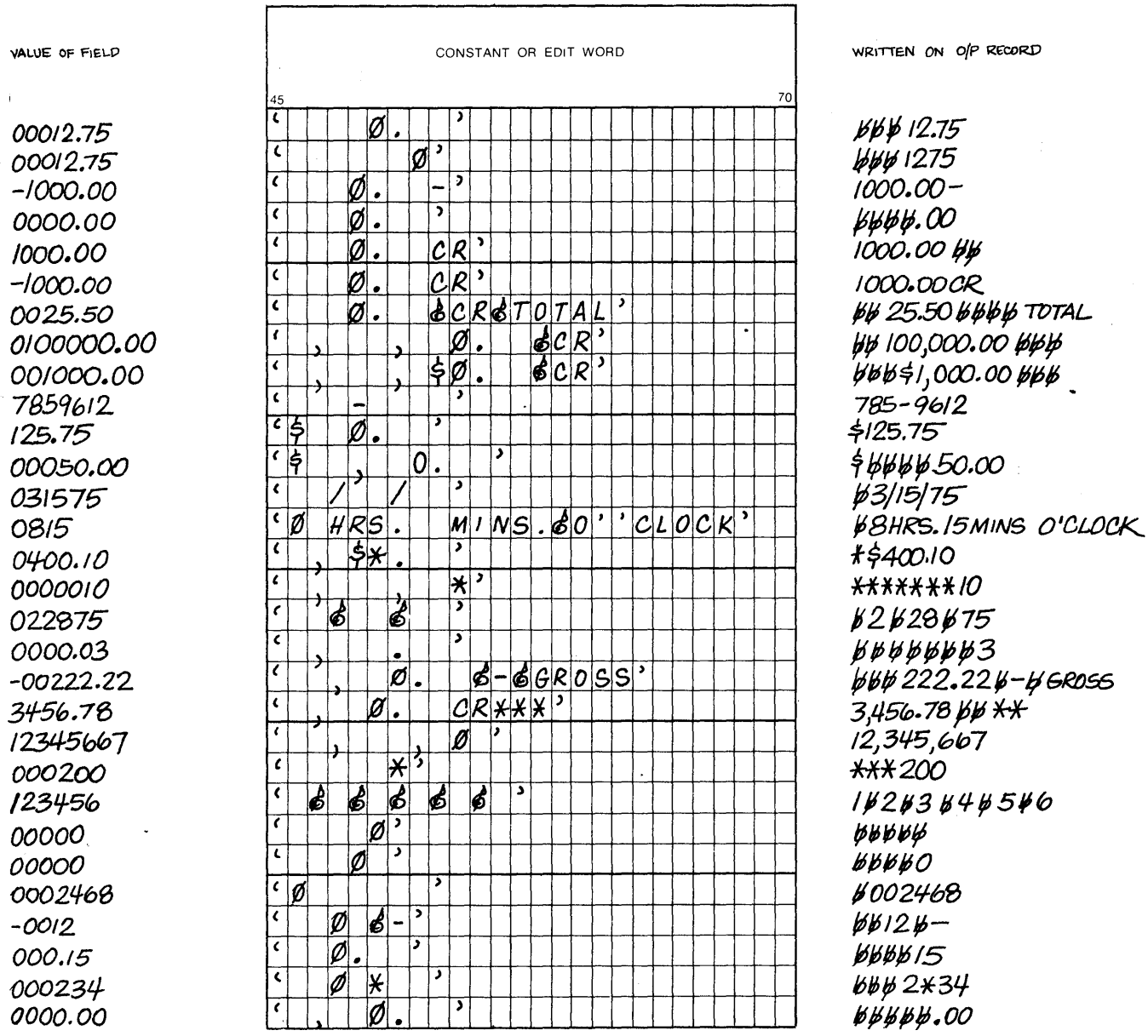
A fixed dollar sign, decimal points, floating dollar sign, commas, ampersands (representing blanks), negative signs (- or CR) and constant information are not replaceable characters.

Note: There are two exceptions to the rule that the number of replaceable characters in the edit word must be equal to the length of the field to be edited. The exceptions are:

1. 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.

2. An extra space can be left in the edit word if the first character in the edit word is a zero. In this case, the field to be edited will not be zero suppressed, but all other specified editing will be performed.

If it is necessary to show a negative number, a sign must be included in the edit word. Either the minus sign (-) or the letters CR may be used. 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.



Note all decimal points appearing under VALUE OF FIELD are implied and do not occupy a byte of memory.

Figure 9-1. Examples of edit words.

9.15 Columns 71-74

These columns are not used and should be left blank.

9.16 Columns 75-80 (Program Identification)

Refer to Chapter 2.

OUTPUT FORMAT SPECIFICATION

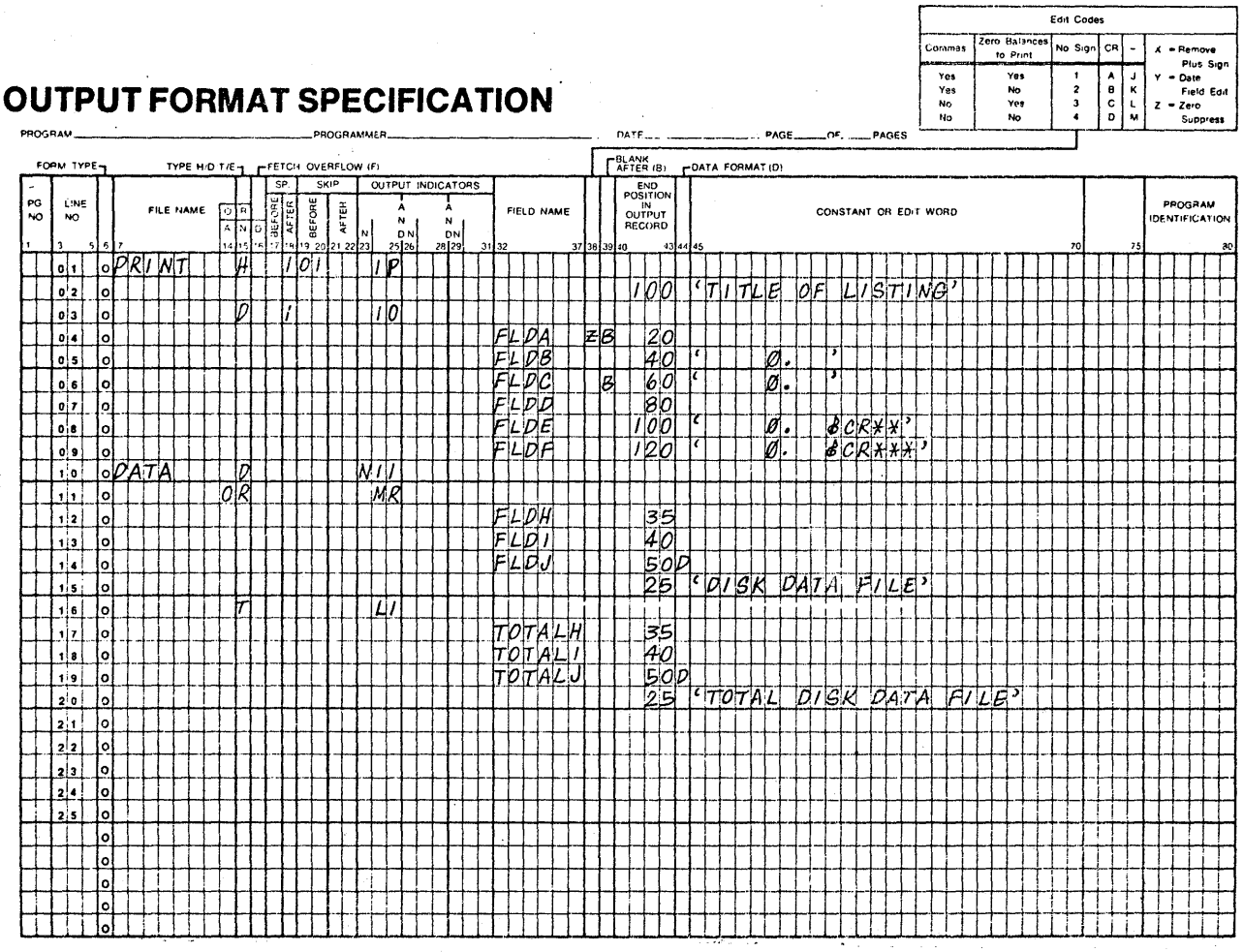


Figure 9-2. Examples of Output Specifications.

APPENDIX A. GENERATION AND USE OF RELOCATABLE RPGPLUS

The collection of files which comprise the Datapoint RPGPLUS system are distributed on a serialized disk pack.

A.1 Compiling an RPGPLUS Program

An RPG II source file is compiled by the RPGPLUS compiler by keying in a command with the following format:

```
RPG srcfil (,objfil)(;(L)(P)(D)(O)(S)(X)(A)(F)(W)(B))
```

where the file names are in standard DOS format. If no object file is specified, the name "srcfil" will be substituted: thus the command file would be "srcfil/CMD". TXT extension is assumed for the source file, which can be in EDIT or DATAFORM format and CMD is the default object file extension. Additional default extensions are PRT for a listing file, REL for the file containing relocatable object code, TXC for the link control file and SYS for the RPG work files. The standard RPGPLUS work file names are suffixed by either a blank or the alphabetized unique character assigned by the Partition System to the partition in which the compiler is running. This character replaces the 'u' shown in file names in the following text. The option characters are used as follows:

```
L - List source program and storage map on printer
P - List source program and storage map on disk
D - List source program and storage map on screen

O - List generated object code      (requires L, P or D)
S - List symbol table               (requires L, P or D)
X - List cross references           (requires L, P or D)

A - Automatic link edit

F - Rename intermediate files (REL, TXC)
W - Rename work files (RPGWRKnu/SYS)
B - Specify new buffer area upper bound
```

For normal use of the compiler, the L or P and the A options are all that are required. The O option will cause listings of 15 to over 100 pages to be produced, and exists for use by maintenance personnel. If the A option is specified and fatal errors occur,

linking will be supressed.

Use of the L,P,F,W and B options will cause the compiler to request additional data. These requests will be made in the following order (with * representing the blinking cursor):

The W option (rename work files) displays:

```
Work file specification:
Work file 0: RPGWRK0u/SYS *
Work file 1: RPGWRK1u/SYS *
```

The second and third messages are requests for new file names; the default names are those shown. These two files are created and then killed each time the RPG compiler is used.

The B option (reset upper bound of buffer area) displays:

```
Buffer allocation up to: ddddd (oooooo) *
```

giving the current limit in both decimal and octal and accepting a new value in either number system (leading 0 for octal); the default value is the one shown.

The F option (rename intermediate files) displays:

```
ENTER INTERMEDIATE OBJECT FILE NAMES (R,L): *
```

and accepts names for the Relocatable object code and Link control files in the format: (relfil)(,lnkfil). These two files have defaults of "objfil/REL" and "objfil/TXC" respectively. The F option allows these files to be saved for future use; if it is not specified, RPGOBJTu/REL and RPGLINKu/TXC are used and LINK must be performed before the next compilation.

The P option (list on disk) displays:

```
LIST ON FILE: *
```

requesting a print file name. The default name and extension for this file are "srcfil" and PRT.

The L and P options (listing requested) display:

```
ENTER HEADING: *
```

and accept a heading to be displayed at the top of each page.

Having obtained any optional data it needs, the compiler will process the source file, and produce a relocatable object file (RPGOBJTu/REL) and a link control file (RPGLINKu/TXC). During the processing, the top right-hand corner of the display will contain the RPGPLUS version number and the message "Phase: XX", where XX are two alphabetic characters. These two characters indicate which compilation phase is being executed at any particular time. There are over 30 separate phases, each corresponding to a particular portion of the compilation process. The names of the phases and their functions are described in Appendix C.

A.2 Linking a Compiled RPGPLUS Program

After a relocatable RPGPLUS program has been compiled, it must be linked with the RPG object library (and user library, if specified) to produce an executable command file. This may be done by using the A option or by typing the command:

```
LINK RPGLINKu;F
```

WARNING: Normally LINKing must be done before another RPGPLUS program is compiled since the compiler will overwrite the two files each time it is used; if LINKing is to be deferred, the F option (see above) can be used to rename these files.

A.3 Running a Linked RPGPLUS Program

After compiling the source file and linking the relocatable object file, the resulting command file is executed by merely calling for it from the command interpreter. For example suppose the source file TEST/TXT were compiled by the command:

```
RPG TEST;L
```

and linked using:

```
LINK RPGLINKu;F
```

or both compiled and linked by the command:

```
RPG TEST;LA
```

Then the object file could be executed by the command:

```
TEST
```

The object file can not accept parameters from the command line; all necessary interaction with the user is done under object program control.

A.3.1 DATE Field

If any of the special words: UDATE, UDAY, UMONTH, or UYEAR were used in the source program, the object program will ask for the date, which should be entered as MM/DD/YY. For example, Sept. 5, 1973, is entered as 09/05/73.

A.3.2 External Indicators

If any of the external indicators, U1 to U8, were used in the source program, the object program will ask for their values at the beginning of execution. The values must be entered in binary, with a 0 setting the indicator off and a 1 setting it on, and in the following order:

U1 U2 U3 U4 U5 U6 U7 U8

Values for indicators not used are not required if there are no used indicators with a higher number. For example, if U1 were the only external indicator used, a valid response is either:

0 or 1.

If U1 and U2 were the only external indicators used, the valid responses are:

00, 01, 10, or 11.

However, if U1 and U3 were used, and no others, the response must be of the form:

0x0, 0x1, 1x0, or 1x1

where x is any character.

A.3.3 Opening Files

Each file opened by the object program causes an opening message to be displayed. In the case of assignable disk files, a message will be displayed, and then the program will wait for a file name to be entered. This name should be in standard DOS format (TXT extension is assumed for data files, ISI extension is assumed for indexed files if extension is not given). If a defined disk file does not exist, an error message will be displayed and the program will then ask for a name as for an assignable file.

A.3.4 Indexing ISAM (Indexed) Files

Indexed files are created in exactly the same format as any fixed format disk file. The data structure is identical and may be processed, disregarding the index, as a simple fixed format file. To permit processing as an indexed file, the INDEX utility is used to create a separate index file. The file is indexed by typing:

```
INDEX datafile(,indexfile);(E)aaa-bbb
```

All parameters within the parenthesis are optional. File names are in standard DOS format. If the indexfile name parameter is omitted, an index will be created with the name "datafile/ISI". The "E" parameter indicates that the index is in EBCDIC collating sequence. If the "E" is omitted the index will be created in ASCII sequence. The parameter aaa (1-255) is the position of the first character in the key and bbb (1-353) is the position of the end of the key.

The indexfile name should be referenced in the File Description Specifications any time the file is used as an indexed input, update or add file in an RPG program.

A.3.5 Console Input Files

When entering data from the keyboard as an input file, end-of-file may be entered by depressing the DISPLAY key and the ENTER key simultaneously. This eliminates the necessity of coding an end of file character to set the LR indicator when keyboard input files are used directly (not in DOS CHAIN). When keyboard input is used with DOS CHAIN, a record which sets the LR indicator should be used to terminate processing.

APPENDIX B. RPGPLUS REFERENCE TABLES

General System Organization

The first part of the appendix lists the various components of the RPGPLUS system and gives a brief description of the function performed by each phase of the compiler. The two-letter abbreviations appear during compilation in the upper right-hand corner of the display. An RPG compilation is passed through the following phases:

1. Interface Program - common data and code.
2. Enter Phases - read, list, and compress source.
3. Assign Phases - allocate data storage.
4. Diagnostic Phases - finish error checking.
5. Generate Phases - generate object program operations for input, compute, output.
6. Assembly Phases - assemble object text in relocatable form.

Enter Phase Summary

AA - Initialize system, read control card, list, compress and diagnose.

AD - Process file-descriptions - compress information, writing part of card on disk, building file-name table and in-core compression table with the rest of the information.

AE - Process file-extension specifications - compress and write on disk.

AF - Process line-counter specifications.

AG - Process input specifications, generating record and field compressions.

AK - Process calculation specifications - read, list, diagnose and compress records.

AM - Process output specifications, generating record and field compressions.

AZ - Process user library inclusion, and compile-time tables.

BA - Initialize symbol name table.

BF - Assign token numbers to TAG names.

BG - Assign token numbers to subroutine names (BEGSR).

BH - Assign token numbers to table and field names.

BQ - Reorganize symbol name table.

Assign Phase Summary

CA - Assign indicator storage.

CB - Generate indicator table for 'DEBUG' operations.

CC - Define
and%%
%%%
%%%%%%%%%%%%%%
%%%table
and field storage, generate table storage.

CH - Scan input, calculation and output compressions, move definitions to compression records.

CI - Generate field storage.

CK - Scan calculation and output compressions, define literals and edit masks in core table.

CL - Move literal definitions into compression records.

CV - Generate literal definitions.

Diagnostic Phase Summary

FC - Diagnose file-descriptions.

FG - Diagnose calculation specifications, check use of table and arrays in calculations, check arithmetic precision.

FK - Diagnose and preprocess input specifications.

FL - Diagnose and preprocess output specifications.

FM - Diagnose file referencing errors.

GA - List error notes in order by line number.

GC to GI - List error texts as needed.

GX - End error text listing, call phase ZZ if fatal error occurred.

Generation Phase Summary

Input Generation Phases:

MB - Generate control field compare.

ME - Generate control field moves.

MG - Generate match field moves.

MK - Generate input field processors.

MP - Generate input, record-tests, and select routines.

MQ - Generate input mainline, end test, and file select sequences.

Calculation Generation Phases:

PA - Generate arithmetic and character (byte) sequences.

PX - Generate detail calculation mainline.

PY - Generate total calculation mainline.

PZ - Generate RPG subroutines.

Phases PX, PY, and PZ generate calculation control code and the code for other operators.

Output Generation Phases:

SA - Generate output field processors.

SO - Generate heading/detail output mainline.

SP - Generate total output mainline.

SQ - Generate overflow output processor.

SR - Generate exception output processor.

SS - Generate put routines.

Final Generation Phases:

UA - Generate file description blocks.

UF - Generate open mainline.

UG - Generate end mainline.

UU - Link generated segments with the fixed library.

Assembly Phases:

WA - Initialize assembly, set up symbol table.

WC - Assembly pass 1 - define internal symbols.

WE - Assembly pass 2 - generate relocatable object code.

WG - Assembly print pass - list object code.

WK - Finish assembly and generate /TXC file.

WS - Cross-reference sort.

WX - Cross-reference listing.

ZZ - RPG Close phase.

Format of File Description Blocks

A partial description of the format of a File Description Block (FDB) is given symbolically in this section. For a complete and absolute listing look at the beginning of an RPGPLUS object listing (O and list options). The symbols defined here have the format 'PxxxxFDB' where the suffix 'FDB' shows the symbol pertains to FDBs and 'P' is a prefix indicating the type of symbol. The prefix 'D' is used for a displacement, 'M' for a mask, 'L' for a length and 'V' for a value. Thus 'DRLENFDB' is the offset of the 2-byte record length field within the FDB, 'MEOFFDB' is the mask used to set the end-of-file flag, 'LCOMFDB' is the length of the common part of an FDB and 'VOPNFDB' is the value of byte passed to a special device driver signifying the OPEN function.

Two-byte quantities such as the record length and the file work area address are stored in standard LSB,MSB form; Boolean (bit) values are represented as 1 = true and 0 = false.

Name	Purpose
DFLAGFDB	Displacement of the three-byte flag field.
	------ First Flag Byte -------
MRREQFDB	Read request pending.
MCLOSFDB	File is closed.
MEOFFDB	File is at end-of-file.
MADDFDB	Append file.
MPUTFDB	Put pending - output record ready for update or chained output.
MDBUFFDB	Buffer dirty - buffer output pending (update files).
MERRFDB	Error (invalid key).
MRPRFFDB	Record is present in buffer.
	------ Second Flag Byte -------

MCHNFDB	CHAIN file.
MWACFDB	Output/update file.
MRACFDB	Input/update file.
MERFFDB	Embedded record (update file).
MAPPFDB	Record appended (don't read ahead).
	------ Third Flag Byte -------
MISAFDB	ISAM file.
MRAFFDB	File processed by ADDROUT file.
MSBCFDB	Suppress buffer clear.
MASTPFDB	ASCII tape.
MHIDNFDB	High density tape.
MFLMKFDB	File mark seen.
	------ Basic Description ------
DRLENFDB	Logical record length (2 bytes).
DBLKFFDB	Blocking factor (2 bytes).
DFWAFDB	File work area address (2 bytes).
DFNAMFDB	File name address (2 bytes).
LFNAMFDB	Length of file name.

Format of Table Description Blocks

The format of table description blocks is given symbolically in this section. Absolute values follow the FDB definitions in an RPGPLUS object listing. Conventions are the same as for file description blocks.

Name	Purpose
DBASETDB	Address of table/array storage area (2 bytes).
DNENTDB	Number of elements (2 bytes).
DELENTDB	Element length (1 byte).
DCENOTDB	Index of current entry (2 bytes).
DCEADTDB	Address of current entry (2 bytes).

APPENDIX C. RPGPLUS COMPILE TIME MESSAGES

In the following itemization of possible RPGPLUS compile-time messages, column 1 is the message number, column 2 distinguishes between warning (W) and fatal (F) errors, column 3 identifies the type of specification to which the flagged statement belongs (Header, File, Extension, etc.), and column 4 gives the text of the message.

No.	W/F	Card	Text
1	W	H	CC7-9 SHOULD BE BLANK.
2	W	H	INVALID CORE SIZE, CC12-14.
3	W	H	INVALID DEBUG CODE, C15.
4	W	H	CC16-25 SHOULD BE BLANK.
5	F	F	INVALID OR BLANK FILE NAME, CC7-14.
6	F	F	INVALID OR BLANK FILE TYPE, C15.
7	F	F	INVALID OR BLANK FILE DESIGNATION, C16.
8	W	F	INVALID PROCESS TO END OF FILE ENTRY, C17.
9	W	F	SEQUENCE ENTRY INVALID, OR SPECIFIED WITH FILE TYPE NOT PRIMARY OR SECONDARY, C18.
10	W	F	INVALID FORMAT ENTRY, C19.
11	W	F	INVALID BLOCK LENGTH, CC20-23.
12	F	F	INVALID RECORD LENGTH, CC24-27.
13	W	F	INVALID MODE OF PROCESSING ENTRY, C28.
14	W	F	CC29-31 SHOULD BE BLANK.
15	W	F	INVALID OVERFLOW ENTRY, CC33-34.
16	W	F	OVERFLOW SPECIFIED WITH DEVICE OTHER THAN PRINTER, CC33-34.

17	W	F	CC35-38 SHOULD BE BLANK.
18	W	F	INVALID EXTENSION OR LINE COUNTER ENTRY, C39.
19	W	F	LINE COUNTER SPECIFIED WITH DEVICE OTHER THAN PRINTER, C39.
20	F	F	INVALID DEVICE CODE, CC40-46.
21	W	F	CC47-52 SHOULD BE BLANK.
22	W	F	CC53-59 SHOULD BE BLANK UNLESS DEVICE SPECIFIED AS 'SPECIAL'.
23	W	F	INVALID 'K' ENTRY, C53.
24	W	F	INVALID UNLESS SPECIAL DEVICE NAME, CC54-59.
25	W	F	INVALID NAME FOR SPECIAL DEVICE I/O ROUTINE, CC54-59.
26	W	F	CC60-65 SHOULD BE BLANK.
27	W	F	C66 SHOULD BE BLANK.
28	W	F	INVALID ADDITIONS ENTRY, C66.
29	W	F	CC67-69 SHOULD BE BLANK.
30	W	F	C70 SHOULD BE BLANK UNLESS TAPE DEVICE.
31	W	F	INVALID REWIND OPTION, C70.
32	W	F	INVALID FILE CONDITION ENTRY, CC71-72.
33	W	F	CC73-74 SHOULD BE BLANK.
34	W	E	CC7-10 SHOULD BE BLANK.
35	F	E	INVALID OR UNRECOGNIZABLE 'FROM' FILE NAME, CC11-18.
36	F	E	INVALID OR UNRECOGNIZABLE 'TO' FILE NAME, CC19-26.
37	F	F	CHAINED, INDEXED OUTPUT FILE C66 MUST BE A.
38	F	E	INVALID TABLE/ARRAY NAME, CC27-32.

39	F	E	INVALID NUMBER OF ENTRIES PER RECORD, CC33-35.
40	F	E	INVALID NUMBER OF ENTRIES FOR TABLE/ARRAY, CC36-39.
41	F	E	INVALID LENGTH OF ENTRY, CC40-42 OR CC52-54.
42	W	E	INVALID FORMAT ENTRY, C43 OR C55 - IGNORED.
43	F	E	INVALID DECIMAL POSITIONS ENTRY, C44 OR C56.
44	W	E	INVALID SEQUENCE ENTRY, C45 OR C57 - IGNORED.
45	F	E	INVALID ALTERNATE TABLE NAME, CC46-51.
46	F	E	EXECUTION-TIME TABLES NOT ALLOWED.
47	F	E	ALTERNATE TABLE SPECIFICATION NOT ALLOWED WITH EXECUTION-TIME ARRAYS.
48	F	E	'FROM' FILE MUST BE AN INPUT-TABLE FILE.
49	F	E	'TO' FILE MUST BE ORDINARY OUTPUT OR OUTPUT-TABLE FILE.
50	F	E	'FROM' FILE TOO SHORT FOR TABLE RECORD.
51	F	E	'TO' FILE TOO SHORT FOR TABLE RECORD.
52	F	E	MORE THAN 8192 BYTES OF TABLE STORAGE ALLOCATED.
53	W	H	INVALID COLLATE SEQUENCE ENTRY, CC26.
54	W	H	CC27-74 SHOULD BE BLANK.
55	F	L	INVALID OR UNDEFINED FILE NAME, CC7-14.
56	F	L	FILE MUST BE ASSIGNED TO THE PRINTER.
57	F	L	FORM LENGTH, CC15-17, INVALID OR >99.
58	W	L	CC18-19 SHOULD CONTAIN 'FL'.
59	F	L	OVERFLOW LINE, CC20-22, INVALID OR >99.
60	W	L	CC23-24 SHOULD CONTAIN 'OL'.
61	W	L	CC25-74 SHOULD BE BLANK.

62	F	L	OVERFLOW LINE IS GREATER THAN FORM LENGTH.
63	F	L	MULTIPLE LINE COUNTER SPECIFICATION LINES.
64-65			UNASSIGNED.
66	F	I	MIXED RECORD AND FIELD DATA, CC7-42, 43-74.
67	W	I	CC7-14 SHOULD BE BLANK FOR 'AND' AND 'OR' CARDS.
68	F	I	FILE NAME, CC7-14, NOT SPECIFIED IN FILES SECTION.
69	F	I	INVALID SEQUENCE ENTRY, CC15-16.
70	W	I	INVALID SEQUENCE ENTRY, C17.
71	W	I	INVALID OPTIONAL SEQUENCE ENTRY, C18.
72	W	I	INVALID RECORD INDENTIFYING INDICATOR, CC19-20.
73	F	I	INCOMPLETE RECORD INDENTIFYING CODE, CC25-27, CC32-34, OR CC39-41.
74	F	I	INVALID POSITION ENTRY, CC21-24, 28-31 OR 35-38.
75	W	I	INVALID NOT ENTRY, C25, 32 OR 39.
76	W	I	INVALID C/Z/D ENTRY, C26, 33 OR 40.
77	W	I	STACKER SELECT NOT IMPLEMENTED, C42.
78	F	I	INVALID FORMAT ENTRY, C43.
79	F	I	INVALID 'FROM' LOCATION ENTRY, CC44-47.
80	F	I	INVALID 'TO' LOCATION ENTRY, CC48-51.
81	F	I	NEGATIVE OR ZERO FIELD LENGTH, CC44-51.
82	F	I	INVALID DECIMAL POSITION ENTRY, C52.
83	F	I	INVALID FIELD NAME, CC53-58.
84	F	I	INVALID CONTROL LEVEL ENTRY, CC59-60.
85	F	I	INVALID MATCH FIELD ENTRY, CC61-62.
86	F	I	INVALID FIELD RECORD RELATION ENTRY, CC63-64.

87	F	I	INVALID FIELD INDICATOR(S), CC65-70.
88	W	I	CC71-74 SHOULD BE BLANK.
89	F	I	RECORD DESCRIPTION ILLEGAL.
90	F	C	INVALID ENTRY IN CC7-8.
91	F	C	INVALID 'NOT' ENTRY IN CC9, 12, 15.
92	F	C	INVALID INDICATOR IN CC10-11, CC13-14, OR CC16-17.
93	F	C	INVALID FACTOR 1 ENTRY, CC18-27.
94	F	C	UNRECOGNIZABLE OPERATION, CC28-32.
95	F	C	INVALID FACTOR 2 ENTRY, CC33-42.
96	F	C	INVALID RESULT FIELD, CC43-48.
97	W	C	HALF-ADJUST ENTRY IN C53 UNRECOGNIZABLE OR NOT ALLOWED.
98	F	C	INVALID RESULT INDICATOR ENTRY, CC54-59.
99	F	C	INVALID LENGTH IN CC49-51.
100	F	C	DECIMAL POSITIONS INVALID, C52.
101	F	C	CC49-52 SHOULD BE BLANK IF NO RESULT FIELD IS SPECIFIED.
102	F	C	INVALID FILE NAME IN CC33-42.
103	F	C	INVALID LITERAL SPECIFICATION.
104	F	C	INVALID BIT MASK, CC33-42.
105	F	C	CONDITION INDICATORS NOT ALLOWED WITH 'TAG', 'RLABL', 'BEGSR', OR 'ENDSR' OPERATIONS.
106	F	F	INVALID KEY LENGTH, CC29-30.
107	F	F	INVALID KEY STARTING POSITION, CC35-38.
108	W	F	ISAM OUTPUT FILE MUST BE "INDEXED" AFTER CREATION.

109	F	F	SEQUENTIAL WITHIN LIMITS VALID ONLY ON INDEXED FILES, C28.
110	F	O	RECORD AND FIELD DATA IN SAME LINE.
111	F	O	INVALID OR UNSPECIFIED FILE NAME, CC7-14.
112	F	O	INVALID 'AND' OR 'OR' ENTRY, CC14-16.
113	W	O	CC17-22 SHOULD BE BLANK ON 'AND' OR 'OR' LINES.
114	F	O	INVALID LINE TYPE, C15.
115	W	O	INVALID FETCH OVERFLOW ENTRY, C16.
116	W	O	INVALID 'SPACE BEFORE' ENTRY, C17.
117	W	O	INVALID 'SPACE AFTER' ENTRY, C18.
118	W	O	INVALID 'SKIP BEFORE' ENTRY, CC19-20.
119	W	O	INVALID 'SKIP AFTER' ENTRY, CC21-22.
120	W	O	INVALID 'NOT' ENTRY, C23, 26, OR 29.
121	F	O	INVALID FIELD CONDITIONING INDICATOR.
122	F	O	INVALID FIELD NAME.
123	F	O	INVALID EDIT CODE, C38.
124	W	O	C38 SHOULD BE BLANK IF NO FIELD NAME SPECIFIED.
125	W	O	INVALID 'BLANK AFTER' ENTRY, C39.
126	F	O	INVALID END POSITION ENTRY, CC40-43.
127	W	O	INVALID DATABUS FORMAT ENTRY, C44.
128	W	O	C44 SHOULD BE BLANK IF NO FIELD NAME SPECIFIED.
129	W	O	LITERAL NOT STARTED WITH A QUOTE, C45.
130	F	O	EMBEDDED SINGLE QUOTE, CC45-70.
131	F	O	REMAINDER OF CC45-70 NOT BLANK AFTER LITERAL.
132	W	O	LITERAL TOO LONG, CC70-80.

133	W	O	INVALID COMBINATIONS OF $ AND * IN CC45-47.
134	W	O	CC45-47 NOT $, *, OR BLANK.
135	W	O	INVALID EDIT WORD.
136	W	O	CC71-74 SHOULD BE BLANK.
137	W	H	MULTIPLE HEADER CARDS.
138	W	H	INVALID ENTRY, C10.
139	W	H	INVALID ENTRY, C11.
140	W	F	FILE NAME TABLE FULL.
141	W	F	DUPLICATE FILE NAME, CC7-14.
142	W	F	INVALID TYPE FOR GIVEN DEVICE CODE, C15.
143	F	F	INVALID DESIGNATION FOR GIVEN DEVICE CODE, C16.
144	W	F	INVALID FILE FORMAT FOR THIS DEVICE, C19.
145	W	F	BLOCKING FACTOR GREATER THAN 255, CC20-27.
146	W	F	BLOCK LENGTH LESS THAN RECORD LENGTH, CC20-27.
147	W	F	BLOCK LENGTH NOT A MULTIPLE OF RECORD LENGTH, CC20-27.
148	F	F	BLOCK LENGTH GREATER THAN ALLOWED FOR GIVEN DEVICE, CC20-23.
149	W	F	BLOCK LENGTH NOT EQUAL TO RECORD LENGTH, CC20-27.
150	F	F	NO FILE DESCRIPTION SPECIFICATIONS.
151	F	F	NO PRIMARY OR SECONDARY FILE SPECIFIED.
152	W	F	SECONDARY FILE PRECEDES PRIMARY FILE.
153	W	F	MULTIPLE PRIMARY FILES. SECONDARY ASSUMED.
154	W	F	AN EXTENSION, C39, MUST BE SPECIFIED FOR TABLE FILES.

155	W	F	EXTENSION, C39, INVALID WITH GIVEN DEVICE OR NON-TABLE FILE.
156	F	F	DEVICE ASSIGNED TO MORE THAN ONE FILE.
157	W	F	CONDITION INDICATOR, CC71-72, INVALID FOR TABLE FILE.
158	W	F	FILE NAME ASSIGNED BUT NEVER USED IN PROPER SECTION.
159	F	F	SEQUENCE, C18, INVALID WITH NO MATCH FIELDS.
160	W	F	SEQUENCE, C18, MUST BE SPECIFIED WITH MATCH FIELDS.
161	F	F	EXTENSION OR LINE COUNTER SPECIFICATION MISSING, C39.
162	W	F	EXTENSION OR LINE COUNTER SPECIFICATION FOUND FOR THIS FILE, BUT C39 IS NOT 'E' OR 'L'.
163	F	F	OUTPUT REFERENCE REQUIRED FOR UPDATE FILE.
164	W	F	CC7-52 ON CONTINUATION CARD SHOULD BE BLANK.
165	F	F	CC54-59 ON CONTINUATION CARD NOT EQUAL 'ASCII'.
166	W	F	CC60-74 ON CONTINUATION CARD SHOULD BE BLANK.
167	F	F	TAPE RECORD LENGTH LESS THAN 18.
168	F	F	ADDITIONS INVALID FOR FILE OR DEVICE, C66.
169	W	F	ALL PRIMARY AND SECONDARY FILES CONDITIONED.
170	F	F	CALCULATION REFERENCE REQUIRED FOR CHAIN OR DEMAND FILES.
171			UNASSIGNED.
172	W	F	INVALID ENTRY IN C53. 'A' ASSUMED.
173	W	F	INVALID LABEL EXIT.
174	W	F	C53 SHOULD BE BLANK.
175	W	F	CC60-65 SHOULD BE BLANK ON ASCII CARD.

176	F	F	UNRECOGNIZABLE DISK CONTINUATION OPTION IN CC54-59.
177	F	F	UNRECOGNIZABLE ENTRY IN CC60-65.
178			UNASSIGNED.
179	W	F	VARIABLE BLOCKING INVALID FOR THIS FILE TYPE, FIXED BLOCKING ASSUMED.
180	W	F	CC31-32 SHOULD CONTAIN 'I ' OR 'AI' FOR RANDOM PROCESSING.
181	F	F	CC29-32 SHOULD CONTAIN ' 3IT' FOR ADDROUT FILES.
182	F	F	ADDROUT FILES MUST BE FIXED-FORMAT, UNBLOCKED FILES WITH RECORD LENGTH EQUAL TO 3.
183	F	F	THIS FILE MUST BE CONTROLLED BY AN ADDROUT FILE.
184	F	F	ADDROUT FILE MUST CONTROL A PRIMARY/SECONDARY FILE.
185	F	F	CORRESPONDING ADDROUT AND PRIMARY/SECONDARY FILES MUST HAVE THE SAME EXTERNAL INDICATOR CONDITION.
186	F	F	UNRECOGNIZABLE PRINTER CONTINUATION.
187	F	F	RECORD ADDRESS TYPE SHOULD BE 'A' OR 'I', CC31.
188	F	F	UNRECOGNIZABLE RECORD ADDRESS TYPE, CC31.
189	F	F	UNRECOGNIZABLE FILE ORGANIZATION, C32.
190	W	F	CC29-30 SHOULD BE BLANK FOR FILE PROCESSED BY ADDROUT FILE.
191			UNASSIGNED
192	F	E	ISAM FILE CAN'T BE CONTROLLED BY TAG FILE.
193	F	E	NO DATA FOR COMPILE TIME TABLE.
194	F	E	MORE THAN ONE ADDROUT FILE CONTROLS THIS PRIMARY/SECONDARY FILE.
195	F	E	MORE THAN ONE PRIMARY/SECONDARY FILE IS CONTROLLED BY THIS ADDROUT FILE.
196	F	E	'FROM' FILE MUST BE AN ADDROUT FILE.

197	F	E	'TO' FILE MUST BE A RANDOMLY-PROCESSED PRIMARY/SECONDARY FILE.
198	F	E	TOO MANY TABLES DEFINED.
199	F	E	MULTIPLE TABLE DEFINITIONS.
200	F	I	NO INPUT SPECIFICATION SECTION.
201	F	I	FIELD PRECEDES FIRST RECORD.
202	F	I	FILE ASSIGNED IS NOT AN INPUT OR UPDATE FILE.
203	F	I	MORE THAN 256 INPUT RECORD TYPES SPECIFIED.
204	F	I	UNASSIGNED.
205	F	I	CONTROL LEVEL SPECIFICATION INVALID WITH FILE TYPE.
206	F	I	MATCH FIELD SPECIFICATION INVALID WITH FILE TYPE.
207	F	I	MORE THAN 255 RECORD ID TESTS FOR THIS RECORD.
208	F	I	FIRST LINE IS AN 'AND' OR 'OR' LINE.
209	F	I	MULTIPLY DEFINED FIELD.
210	F	I	LENGTH OF CONTROL FIELDS GREATER THAN 255 BYTES.
211	F	I	LENGTH OF MATCH FIELDS GREATER THAN 255 BYTES.
212	F	I	MORE THAN 32 'AND' LINES.
213			UNASSIGNED.
214	W	I	'AND' LINE FOLLOWS LINE WITHOUT RECORD ID CODES.
215	W	I	NO FIELDS DESCRIBED FOR PREVIOUS RECORD.
216	W	I	NUMERIC SEQUENCE ENTRIES NOT IN ORDER, OR FIRST ENTRY NOT EQUAL 01.
217	F	I	CC17-18 SHOULD BE BLANK FOR ALPHABETIC SEQUENCE.
218	W	I	CC17-20 SHOULD BE BLANK FOR 'AND' LINES.
219	W	I	CC17-18 SHOULD BE BLANK FOR 'OR' LINES.

220	F	I	LENGTH OF NUMERIC FIELD GREATER THAN 15, OR LENGTH OF ALPHABETIC FIELD GREATER THAN 256.
221	W	I	DECIMAL POSITION ENTRY INVALID FOR ARRAY.
222	F	I	NUMBER OF DECIMAL POSITIONS EXCEEDS FIELD LENGTH.
223	F	I	TABLE NAME INVALID FOR A FIELD NAME.
224	F	I	'AND' LINES INVALID WITH LOOK-AHEAD RECORD.
225	F	I	CC17-18, 21-42, AND 59-74 INVALID WITH LOOK-AHEAD.
226	F	I	FIELD LOCATION ENTRIES EXCEED RECORD LENGTH.
227	F	I	FIELD NAME IS A RESERVED WORD OTHER THAN 'PAGE'.
228	F	I	CONTROL AND MATCH SPECIFICATIONS INVALID FOR ARRAYS.
229	F	I	LOOK-AHEAD INVALID WITH CHAIN OR DEMAND FILES OR WITH THIS DEVICE.
230	F	I	NO FIELDS SPECIFIED FOR LOOK-AHEAD RECORD.
231	F	I	ARRAY LENGTH EXCEEDS OR IS NOT A MULTIPLE OF LENGTH IN EXTENSION SPECIFICATION.
232	F	I	INCONSISTENT LENGTHS FOR CONTROL OR MATCHING FIELDS OF ONE LEVEL.
233	F	I	INVALID SPLIT CONTROL FIELD SPECIFICATION.
234	W	I	CONTROL OR MATCHING FIELDS SPECIFIED AS ALPHA AND NUMERIC.
235	F	I	ALL VALID MATCH LEVELS WERE NOT REFERENCED IN THE LAST RECORD GROUP.
236	F	I	CONTROL OR MATCH FIELDS WITHOUT FIELD RECORD RELATION MUST PRECEDE THOSE WITH FIELD RECORD RELATION.
237	F	I	CONTROL OR MATCH FIELDS WITH FIELD RECORD RELATION MUST BE GROUPED BY FIELD RECORD RELATION.
238	F	I	FIELD RECORD RELATION INDICATOR USED IMPROPERLY WITH CONTROL OR MATCH FIELDS.

239	W	I	INDICATOR ASSIGNED BUT NOT USED.
240	F	I	INDICATOR USED, BUT NOT ASSIGNED.
241	F	I	FIELD LENGTH NOT MULTIPLE OF TABLE ENTRY LENGTH.
242	F		INDEX FIELD NOT NUMERIC OR DECIMAL POSITIONS > 0.
243	F		LITERAL INDEX OUT-OF-BOUNDS.
244	F		CONFLICT IN TAPE DENSITY.
245	F		FILE ORGANIZATION SPECIFICATION SHOULD BE BLANK, CC32.
246	F		FILE ORGANIZATION SHOULD BE 'I' OR 'T', CC32.
247	F		RECORD ADDRESS TYPE AND FILE ORGANIZATION ARE INCOMPATIBLE, CC31-32.
248-249			UNASSIGNED.
250	F	C	INVALID FILE FOR FORCE.
251	F	C	INVALID FILE FOR READ.
252	F	C	INVALID CHAINING FIELD.
253	F	C	INVALID FILE IN CHAIN.
254	F	C	DEBUG FILE NOT OUTPUT FILE.
255	W	C	DEBUG OPERATIONS IN PROGRAM IGNORED.
256	W	C	DEBUG OPTION WITHOUT DEBUG OPERATION.
257	F	C	DIFFERENT DEBUG FILES.
258	F	C	INVALID FILE FOR DSPLY.
259	W	C	CALCULATIONS CONSIST ONLY OF SUBROUTINES.
260	F	C	SUBROUTINE MUST BEGIN WITH 'BEGSR' OPERATION.
261	F	C	TOTAL OR DETAIL RECORD OUT OF SEQUENCE.
262	F	C	ARRAY IMPROPERLY USED IN RESULT FIELD.

263	F	C	FACTOR 1 OR 2 MAY NOT BE AN ARRAY UNLESS RESULT FIELD IS.
264	F	C	RECORD LENGTH FOR DEBUG FILE IS TOO SMALL.
265	F	C	FACTOR 1 IN 'DEBUG' SHOULD BE LESS THAN NINE BYTES LONG.
266	F	C	SUBROUTINE MUST END WITH 'ENDSR'.
267	F	C	RESULT FIELD MUST BE ALPHANUMERIC.
268	F	C	FACTOR 2 MUST BE ALPHANUMERIC.
269	F	C	FACTORS 1 & 2 MUST HAVE SAME TYPE.
270	F	C	BIT OPERATIONS TAKE SINGLE-BYTE FIELDS.
271	F	C	FACTOR 2 IN 'LOKUP' MUST BE A TABLE OR ARRAY.
272	F	C	CORRESPONDING TABLE MAY NOT BE USED WITH ARRAY LOOK-UP.
273	F	C	RESULT FIELD IN LOKUP MUST BE A TABLE.
274	F	C	FACTOR 1 MUST HAVE SAME LENGTH AS FACTOR 2 IN LOOK-UP.
275	F	C	'BEGSR' IN MIDDLE OF SUBROUTINE.
276	F	C	'RLABL' MUST IMMEDIATELY FOLLOW 'EXIT'.
277	F	C	INVALID LABEL OPERAND.
278	F	C	'BEGSR' OR 'ENDSR' IN DETAIL OR TOTAL RECORDS.
279	F	C	FACTOR 1 MUST BE NUMERIC.
280	F	C	FACTOR 2 MUST BE NUMERIC.
281	F	C	RESULT FIELD MUST BE NUMERIC.
282	W	C	HALF-ADJUST NOT NEEDED, ENTRY ASSUMED BLANK.
283	W	C	COMPUTED RESULT MAY OVERFLOW RESULT FIELD.
284	F	C	FACTOR 2 IN 'XFOOT' MUST BE AN ARRAY.

285	F	C	'MVR' MUST FOLLOW 'DIV'.
286	F	C	HALF-ADJUST ON PREVIOUS 'DIV' ILLEGAL WITH 'MVR'.
287	F	C	FACTOR 2 NOT A PROCESS WITHIN LIMITS FILE.
288	F	C	FACTOR 1 IS NOT A VALID KEY.
289	F	C	INVALID USE OF AND/OR LINE.
290	F	C	PRECEDING LINE SHOULD HAVE AN OP-CODE OR THIS LINE SHOULD BE AN AND/OR LINE.
291	F	C	FACTOR 2 AND RESULT FIELD CAN NOT BE SAME ARRAY.
292	F	C	TABLE OR ARRAY MUST BE EITHER ASCENDING OR DESCENDING FOR HIGH OR LOW 'LOOKUP'.
293-327			UNASSIGNED
328	W	O	DATAPOINT COMPATIBLE FIELD SHOULD BE NUMERIC.
329	F	O	NEITHER FIELD NAME OR LITERAL IS PRESENT.
330	F	O	'AND' OR 'OR' LINE NOT PRECEDED BY RECORD LINE.
331	W	O	SPACE AND SKIP INVALID WITH DEVICE OTHER THAN CONSOLE OR PRINTER.
332	W	O	SKIP ENTRY GREATER THAN FORM LENGTH.
333	W	O	FETCH OVERFLOW INVALID FOR DEVICE OTHER THAN PRINTER.
334	F	O	OVERFLOW INDICATOR INVALID FOR EXCEPTION LINE.
335	W	O	FETCH OVERFLOW INVALID WITH OVERFLOW INDICATORS.
336	F	O	OVERFLOW INDICATOR USED IS NOT ASSIGNED TO THIS FILE.
337	W	O	1P INDICATOR INVALID ON TOTAL OR EXCEPTION LINES.
338	W	O	FETCH OVERFLOW INVALID WITH 1P INDICATOR.
339	W	O	SPACE BEFORE OF 0 INVALID FOR CONSOLE.
340	F	O	INVALID INDICATORS USED WITH 1P INDICATOR.

341	F	O	END POSITION GREATER THAN RECORD LENGTH.
342	F	O	LENGTH OF ARRAY, ELEMENT, OR FIELD EXCEEDS RECORD LENGTH.
343	F	O	END POSITION TOO LOW.
344	W	O	ALL INDICATORS MISSING OR NEGATIVE IN PREVIOUS RECORD.
345	W	O	ALL INDICATORS MISSING ON THIS LINE.
346	F	O	INVALID EDIT WORD SIZE.
347	F	O	EDIT CODE INVALID WITH ALPHA FIELD OR CONSTANTS OTHER THAN $ OR *.
348	F	O	CONSTANT INVALID WITH EDIT CODES X, Y OR Z.
349	F	O	INVALID FIELD LENGTH FOR Y EDIT CODE.
350	F	O	DECIMAL POSITIONS INVALID WITH Y EDIT CODE.
351	F	O	INVALID FILE TYPE FOR OUTPUT RECORD.
352	W	O	BLANK AFTER INVALID WITH RESERVED WORD OTHER THAN 'PAGE'.
353	F	O	MORE THAN 32 'AND' OR 'OR' LINES.
354	W	O	BLANK AFTER SPECIFIED FOR A CONSTANT.
355	F	O	ARRAY INDEX EXCEEDS NUMBER OF ELEMENTS.
356	W	O	BLANK AFTER INVALID WITH LOOK-AHEAD.
357	W	O	INDICATOR ASSIGNED BUT NEVER USED.
358	F	O	INDICATOR USED BUT NEVER ASSIGNED.
359	F	O	FIELD NAME USED BUT NOT DEFINED.
360	F	O	TABLE OR ARRAY NAME USED AS INDEX.
361	F	O	NUMBER OF DECIMAL POSITIONS EXCEEDS FIELD LENGTH.
362	W	O	L0-L9 IN 'OR' RELATIONSHIP WITH LR.

363	F	O	ADDITIONS INVALID WITH 'AND' OR 'OR' LINES.
364	W	O	FOR ADD FILES, EACH RECORD MUST HAVE 'ADD' IN CC16-18.
365	F	O	ADDITIONS INVALID WITH FILES EXCEPT SEQUENTIAL DISK FILES.
366	F	O	'T' IN C15, OR E WITH L0-L9 INVALID WITH UPDATE FILES.
367	F	O	FIELD LINE PRECEEDS FIRST RECORD LINE.
368			UNASSIGNED.
369	F	O	MORE THAN 255 OUTPUT RECORD TYPES SPECIFIED.
370	F	O	NO OUTPUT SPECIFICATION SECTION FOUND.
371	F	O	RECORDS MUST BE IN SAME ORDER AS FILES.
372	F	O	H, D, T AND E LINES MUST BE IN ORDER.
373	F	O	FIELD LENGTH DOES NOT CORRESPOND TO NUMBER OF REPLACEABLE CHARACTERS IN EDIT WORD.
374	W	O	EXCEPT RECORD WITHOUT 'EXCPT' OPERATION.
375	F	O	EDIT CODE INCOMPATIBLE WITH OPTIONS USED IN CC45-47.
376	F	O	NO REPLACEABLE CHARACTERS IN EDIT WORD.
377	F	O	FILE IS NOT A DISK ADD FILE, CC16-18.
378-379			UNASSIGNED.
380	W		SEQUENCE NUMBERING ERROR IN SOURCE RECORDS, CC1-5. (Note: a sequence error will occur if a blank record is present in the source code.)
381	F		RECORD TYPE OUT OF SEQUENCE IN SOURCE RECORDS, C6.
382	F		INVALID CHARACTERS IN (MAIN) OPERAND NAME.
383	F		INVALID CHARACTERS IN INDEX OF OPERAND.
384	F		INDEX IS INVALID WITH THIS OPERAND.

385-399		UNASSIGNED.
400	F	SEQUENCE ERROR IN COMPILE-TIME TABLE OR ARRAY.
401	F	NUMERIC FIELD ERROR IN COMPILE-TIME TABLE OR ARRAY.
402	W	END OF FILE FOLLOWS '**b' RECORD.
403	F	NO FILE NAME IN LIBRARY INCLUSION RECORD.
404	F	USER LIBRARY FILE DOES NOT EXIST.
405	W	DELIMITER CARD FORMAT ERROR.
406	W	INSUFFICIENT DATA FOR TABLE OR ARRAY.
407	F	EXCESS DATA FOR TABLE OR ARRAY.
408	F	ALTERNATE TABLE BUFFER FULL.
409	W	NO '**b' RECORD FOLLOWS LIBRARY INCLUSION RECORD.
410	W	NO COMPILE-TIME TABLE/ARRAY FOR DATA.
411	W	INVALID LIBRARY FILE NAME.
412	F	COMPILE-TIME TABLE/ARRAY DATA RECORD LENGTH > 80.

APPENDIX D. RPGPLUS OBJECT (EXECUTION) TIME MESSAGES

During the execution of an RPGPLUS object program, messages will be displayed on the screen either to request input from the user or merely to inform him of certain actions being performed. In addition, error messages will be displayed if abnormal situations are encountered. The following list of messages are all those which could possibly occur during an RPGPLUS object program execution. The list includes an explanation of each message, special action taken by the object program after displaying the message, and an explanation of the response from the user, if necessary. RPGPLUS accepts a single character response (R,B,C,or A for Resume, Bypass, Cancel, or Abort, respectively) from the operator only; this response cannot be incorporated into a CHAIN file. This appendix lists unnumbered general messages followed by numbered diagnostics.

RESUME/BYPASS/CANCEL/ABORT
BYPASS/CANCEL/ABORT
CANCEL/ABORT

Explanation: General error messages displayed after other error messages to give the user an option as to what action should be taken by the program.

Program Action: Wait for input.

User Response: Up to four different responses are allowed. Typing an 'R' will cause the program to Resume execution at the point where the error occured. Typing a 'B' will cause the program to Bypass the current cycle and read the next record. Typing a 'C' will Cancel program execution and close all the files. Typing an 'A' will immediately return control to the operating system without closing any files.

In the following descriptions, those errors permitting Resume or Bypass will be marked with a parenthesized 'R' or 'B' following the Object Time Message number. Messages not so marked must be processed by CANCELling or ABORTing program

execution.

OPEN xxxxxxxx AS yyyyyyyy FILE:

Explanation: General message displayed during the file opening sequence for every file described in the program. The program name for the file will appear in place of the x's and the type of file will appear in place of the y's. This message will be followed by one of the following: blinking cursor (for an assignable disk file), the file name defined in the source program, or an asterisk followed by the device being used (e.g. "*SERVO PRINTER").

Program Action: Wait for response if assignable disk file.

User Response: If the file is an assignable disk file, enter the file name. The default extension is "TXT"; "search all drives" will be assumed if no drive is specified.

(NO SUCH FILE)

Explanation: Error message displayed if, when naming an input file during the file opening sequence, the named file does not exist.

Program Action: The request for a file name will be repeated.

User Response: The name of an existing file should be entered.

(BAD FILE SPEC)

Explanation: Error message displayed if a specification of a file during the file opening sequence is incorrect.

Program Action: The request for a file name will be repeated.

User Response: A correct file name should be entered.

DSPLY

Explanation: Message displayed whenever the DSPLY operation is executed.

User Response: If blinking cursor displayed, enter new value for quantity shown.

ENTER EXTERNAL INDICATOR SETTING IN BINARY

Explanation: Message displayed if any of the external indicators, U1 to U8, were used in the source program.

Program Action: Wait for input.

User Response: The values of the external indicators used in the program should be entered. Detailed formatting can be found in Appendix A.

ENTER DATE AS MM/DD/YY

Explanation: Message displayed if any of the special names: UDATE, UDAY, UMONTH or UYEAR were used in the source program.

Program Action: Wait for input.

User Response: The desired date should be entered. The format of the date is fixed, in that January 24, 1977 would be entered as 01/24/77.

FORMAT CORRECT, CONTINUE?

Explanation: Message displayed if the format of the tape header labels was correct.

Program Action: Display labels then wait for response.

User Response: If the correct tape is mounted, a "Y" should be

entered so processing can continue. If the wrong tape is mounted, an "N" should be entered to stop program execution.

ERROR HALT n

Explanation: Message displayed at the end of a cycle if Halt indicator n (H1 - H8) is found on.

001(B): MULTIPLE WRITES TO LOADER

Explanation: Error message displayed if more than one record is written to device LOADER.

002: LOADER OVERLAY MISSING

Explanation: Error message displayed if the source program has specified the LOADER device and the loader object file (RPGLDR/OV1) does not exist.

User Response: The file RPGLDR/OV1 should be re-installed from the RPGPLUS generation tapes and the program re-run.

003(B): NON-DIGIT IN CONVERSION TO BINARY

Explanation: Error message displayed if a field or literal used as an array index of a record address of a CHAINed file is less than zero or greater than 65535.

004(B): DATABUS INPUT ERROR

Explanation: Error message displayed if the format of a number being read from an input file is not in correct DATABUS format.

005(B): NUMERIC FIELD ERROR

Explanation: Error message displayed if a character in a numeric field is not a digit.

006(B): RESULT OVERFLOW

Explanation: Error message displayed when the result of an arithmetic operation is too large to fit in the field specified.

007(B): DSPLY FIELD TOO LONG

Explanation: Error message displayed if the length of a field being displayed in the DSPLY operation is greater than 80.

008(B): DIVIDE BY ZERO

Explanation: Error message displayed upon an attempt to divide by zero.

009(R): SQUARE ROOT IMAGINARY

Explanation: Error message displayed if an attempt is made to take the square root of a negative number.

010(B): INVALID INDEX

Explanation: Error message displayed if an array index is less than 1 or greater than the number of elements in the array.

011(R): NO DATA FOR TABLE/ARRAY LOAD

Explanation: Error message displayed if there is no data for a pre-execution time table or array.

Program Action: Leave table empty.

012(R): TOO MUCH DATA FOR TABLE/DATA LOAD

Explanation: Error message displayed if a pre-execution time table or array has been entirely filled and there is another record of data for that table or array.

Program Action: Ignore excess data records.

013: SEQUENCE ERROR IN TABLE/ARRAY LOAD

Explanation: Error message displayed if the sequence of data being read into a pre-execution time table or array is not as specified on the Extension Specs.

014(B): OPTION TEST LOOP

Explanation: Error message displayed if all record types for an input file are described as optional and the current input record is not identifiable.

015(B): MATCH SEQUENCE ERROR

Explanation: Error message displayed if the sequence of data in any specified match fields is not as specified on the File description specs.

016(B): RID TESTS FAILED

Explanation: Error message displayed if a record in an input file can not be identified; i.e. does not match any of the Record IDentifying codes on the Input specification.

017(R): INVALID EXTERNAL INDICATOR SETTING

Explanation: Error message displayed if the external indicators entered were not a valid 0 or 1.

Program Action: Request for external indicator setting repeated.

User Response: Enter correct setting.

018(R): INVALID USER DATE ENTERED

Explanation: Error message displayed if incorrect date entered - must have form mm/dd/yy where mm (month), dd (day) and yy (year) are two-digit numbers.

Program Action: Request for date repeated.

User Response: Enter date in correct format.

026: NO WRITE RING FOR OUTPUT

Explanation: Error message displayed if magnetic tape to be written on does not have a write ring.

028: END OF TAPE FOUND

Explanation: Error message displayed if the physical end of a magnetic tape is encountered.

029(B): RECURRING PARITY ERROR

Explanation: Error message displayed if bad parity persists during an attempt to read tape.

030(B): BAD TAPE FOUND DURING WRITE

Explanation: Error message displayed if it is not possible to write a tape block correctly.

031: FDBS DO NOT MATCH IN TAPE CLOSE

Explanation: Error message displayed if the FDB address supplied to tape close does not match that stored by the open routine.

032: NO VOLUME 1 LABEL FOUND

Explanation: Message displayed if the VOL1 label is either missing or invalid.

033: MISSING OR INVALID HDR1 LABEL

Explanation: Error message displayed if the HDR1 label on an input tape is either missing or invalid.

034: MISSING OR INVALID HDR2 LABEL

Explanation: Error message displayed if the HDR2 label on an input tape is either missing or invalid.

035: RECORD FORMAT NOT FIXED

Explanation: Error message displayed if format specified in tape labels is not Fixed ("F").

036: WRONG BLOCK SIZE IN LABEL

Explanation: Error message displayed if the block length in the HDR2 label of an input tape is not the same as that specified in the source program.

037: WRONG RECORD SIZE IN LABEL

Explanation: Error message displayed if the record length in the HDR2 label of an input tape is not the same as that specified in the source program.

038: OPERATOR ABORT REQUESTED

Explanation: Message displayed when operator responds negatively to message asking if tape label format is correct.

039: NO EOF1 LABEL FOUND

Explanation: Error message displayed if the EOF1 label is missing or invalid on an input tape.

040: BLOCK COUNT NOT EQUAL TO COUNT IN LABEL

Explanation: Error message displayed if the block count in the EOF1 label is not equal to the number of blocks read from an input tape.

041: INVALID TAPE DENSITY IN LABEL

Explanation: Error message displayed if tape density specified in HDR2 does not match that defined in source program.

042: INVALID LABEL RECORD LENGTH

Explanation: Error message displayed if the length of a tape label record was incorrect.

051: INVALID OPENING OF CASSETTE FILE

Explanation: Error message displayed if a file has already been opened using this cassette drive.

052(B): READ PARITY ERROR

Explanation: Error message displayed if a parity fault occurred while reading a cassette file.

053(B): WRITE PARITY ERROR

Explanation: Error message displayed if a parity fault occurred while writing a cassette file.

054: CAN NOT POSITION TAPE

Explanation: Error message displayed if there is no file zero on a cassette tape.

055: END OF CASSETTE TAPE

Explanation: Error message displayed if the physical end of a cassette tape is encountered.

057(B): FILE FORMAT ERROR

Explanation: Error message displayed if disk file has format error - EOR missing or prior record not terminated by EOS or deletion character.

058(B): SHORT INPUT RECORD

Explanation: Error message displayed if record read from Fixed disk file is shorter than specified in the source program.

059(B): LONG INPUT RECORD

Explanation: Error message displayed if record read from Fixed disk file is longer than specified in the source program.

062(B): DR$ READ ERROR

Explanation: Error message displayed when an attempt to read a sector of an index file fails.

063(B): FORMAT ERROR IN EXTENDING DIRECT FILE

Explanation: Error message displayed when the point to which a direct (CHAIN by record number) file comes before the last record of the file.

064: ISAM FILE KEY DOES NOT MATCH FDB SPECIFCATION

Explanation: Error message displayed when key description in index file does not match that given in the source program.

065: OPEN ERROR IN ISAM DATA FILE

Explanation: Error message displayed when data file named in index file cannot be opened or has invalid name.

077: WRITE PROTECTED

Explanation: Error message displayed if an attempt is made to write on a disk file which is write protected.

078: DELETE PROTECTED

Explanation: Error message displayed if an attempt is made to shorten or delete a disk file which is delete protected.

079: FILE SPACE FULL

Explanation: Error message displayed if an attempt is made to allocate space to a disk file when either the disk is full or no more segment descriptor slots are available for the file.

080: DRIVE OFF LINE

Explanation: Error message displayed if an attempt is made to access a disk drive which is either physically absent or off line.

081(B): CHAINING ERROR

Explanation: Error message displayed when attempt is made to CHAIN to a non-existent record and the source program has not specified an indicator to set.

082: INVALID BUFFER ADDRESS

Explanation: Error message displayed if the buffer address in a record address (ADDROUT) file is invalid.

083: PARITY ERROR IN INDEX - REINDEX

Explanation: Error message displayed if a parity error is found in the index of an ISAM file.

084(B): DUPLICATE KEY

Explanation: Error message displayed if the program attempts to add a record to an indexed file and a record already exists with the same key.

085(B): MULTIPLE UPDATES IN SAME CYCLE

Explanation: Error message displayed if the program attempts to write two or more records onto an Update file during a single cycle.

086(B): MULTIPLE CHAINED OUTPUT IN SAME CYCLE

Explanation: Error message displayed if the program attempts to write two or more chained output records during a single cycle.

087(B): READ AT EOF OR FROM CLOSED FILE

Explanation: Error message displayed if the READ operation was attempted on a file which was at end-of-file or was closed and the source did not specify an indicator to set.

APPENDIX E. RPGPLUS USER ASSEMBLY LANGUAGE FACILITIES

E.1 The RPGPLUS Library Facility

An integral component of the RPGPLUS compiler system is the LIBRARY facility. This facility includes the system library file RPGOLIB/REL and one optional user library file (<user lib>/REL specified by the *LIBRARY <user lib> line in the source code). These libraries contain relocatable code sequences which can be selectively included in an RPGPLUS object program, depending upon the particular operations specified in the source program. SNAP/2 is used to transform a library source file into a relocatable object file; LIB may be used to combine such files into a larger library.

The USER LIBRARY facility allows for user written routines to be assembled into an RPGPLUS object program: SPECIAL device drivers, user label processors, and routines referenced by the EXIT operation. The calling sequences generated for these features will be described at the end of this appendix.

A library file is partitioned into SEGMENTS, each of which can be included separately into an RPGPLUS object program. Segment inclusion is done on the basis of ENTRY POINTS in the segment and undefined symbols in the object code. In other words, during object program LINKing, a segment will be included if at least one of its entry points corresponds to an undefined symbol in the dictionary. When such a segment is found, it becomes part of the object code and treated exactly as if it were code directly generated by the compiler. Any undefined symbols it may have, if not already defined previously, will then cause additional library segments to be included. In this way, a hierarchy of segments can be included in the object code, depending upon the particular operation specified in the source program.

E.2 RPGPLUS Calling Sequences to User Subroutines

The RPGPLUS system will generate calls to user subroutines when any of the following language features is invoked:

1. SPECIAL files,
2. non-standard tape labels, or
3. EXIT operations.

In each case the name of the subroutine must be given in special columns as follows:

1. cc54-59 - Label Exit - in the File Description Specifications, and
2. cc33-38 - Factor 2 - in the Calculation Specifications.

The compiler will prefix each user subroutine name with the characters 'X$' to distinguish it from entry-points in the system library.

E.3 SPECIAL Device Drivers

Each file in the RPG object program is described by a table called the File Description Block (FDB). The format of this table and values of associated symbols are given in Appendix B. A SPECIAL device subroutine will be called: to open the file, to read from it, to write on it, or to close it. In all cases the subroutine will be called with an operation code in A and the address of the FDB in HL. When the file has been successfully opened, the closed-file flag MCLOSFDB in the file's FDB must be cleared; if the file is an input file, the End-Of-File flag MEOFFDB should be set when the end of the file has been found.

The values of the operation codes can be found by compiling an RPG II program with an object listing (O and list options) and looking for the following symbols (defined near the beginning):

Value of A	Operation
VOPNFDB	Open
VGETFDB	Input
VPUTFDB	Output
VCLSFDB	Close

Thus a simple input driver which requires no open or close actions

might start like this (remember the 'X$' prefix for subroutine names):

```
MYDRIVER  PROG                          Begin library segment
CODE      ORG       0
DATA      ORG       0,P
          INC       IFDBDEF/EPT         FDB definitions
.
          USE       DATA
WORKAREA                                Base of data page
CURRFDB   SK        2                   Address of FDB
          ...                           Additional data
.
          USE       CODE
X$MYDRV:  CP        VGETFDB             GET operation?
          JFZ       CHKOPEN               No, check for OPEN
.
          PUSH      XA                  Save (X)
          LX        WORKAREA>8          Set page
          DPS       HL,CURRFDB          Save FDB address
.
          ...                           Read record
.
          POP       XA                  Restore (X)
          RET                           EXIT
.
* End of file seen
.
ENDOFILE  LFII      HL,DFLAGFDB,CURRFDB Address flag byte
          LAM                             & fetch it
          OR        MEOFFDB             Set end of file
          LMA                             & update
.
          LFII      HL,DRLENFDB,CURRFDB Address record length
          DL        BC,HL                 & fetch it
.
          LFII      HL,DFWAFDB,CURRFDB  Point to FWA address
          DL        HL,HL                 & fetch it
.
          POP       XA                  Restore (X)
          JMP       BLNSETL             EXIT clearing FWA
.
.
* Operation is not GET
.
CHKOPEN   CP        VOPNFDB             OPEN function?
          RFZ                             No, EXIT
```

```
        LAM                             Yes, get flag byte
        ND      -1.XOR.MCLOSFDB         Clear file closed
        LMA                             Update FDB
        RET                             EXIT

        END                             End library segment
```

E.4 Non-standard Tape Labels

A non-standard label routine is called with the operation in A, the tape data page address MSB in X, and the tape FDB address stored at TAPFDB in the data page. The operation code is 0 for open (i.e., header labels) and not 0 for close (i.e., trailer labels). The tape will be positioned before the first label record in each case and it is the responsibility of the label routine to properly position the tape at the beginning of data (for headers) or before the last tape mark (for trailers). The standard tape I/O routines TAPEREAD and TAPEWRIT may be used to read and write tape labels. They both require the address of a label record buffer in HL and its length in DE. TAPEREAD will read up to the specified number of characters and then store the character EOTXT to mark the end of the record; it will return the False Zero condition if the read was not successful. TAPEREAD will also set the flag MFLMKFDB in the tape FDB whenever it sees a file mark. The subroutine TAPINIT must be called before reading a set of labels; before calling TAPEWRIT, call CHKTAPE to wait for a previous write to finish. Additional subroutines are WRITFMRK which writes a file mark (call CHKTAPE also) and TAPRESET to clear one. For example:

```
MYLABELS  PROG
CODE      ORG     0
WORK      ORG     0
          INC     IFDBDEF/EPT           FDB definitions
LLABLUSR  EQU     ??                    User label length
.
          USE     CODE
X$MYLABL: ORA     Opening file?
          JFZ     TRLRLABL                No
.
          LFII    HL,DFLAGFDB+1,TAPFDB    Yes
          LAM                           Get flag byte
          ND      MWACFDB               Output?
          JFZ     OUTHEADR                Yes
.
```

```
* Process header labels of input file
.
          CALL      TAPINIT             Initialize read
          ...
          CALL      GETLABEL            Get a label record
          ...
          RET                           EXIT
.
* Process header labels of output file
.
OUTHEADR  ...
.
* Process trailer labels
.
TRLRLABL  ...
.
.
* Subroutine to read a label record
.
GETLABEL  HL        LABLBUFR            Address label
          DE        LLABLUSR            Load length
          CALL      TAPEREAD            Read a record
          JFZ       LABELERR            If read failed
.
          HL        ROLLUP
          CALL      DSPLY$              Rollup screen
          HL        LABLBUFR            Address label
          JMP       DSPLY$              EXIT displaying label
.
LABELERR  ...                           Error routine
.
* Subroutine to write a label record
.
PUTLABEL  CALL      CHKTAPE             Wait for prior write
          HL        LABLBUFR            Address label
          DE        LLABLUSR            Load length
          JMP       TAPEWRIT            EXIT writing record
.
* Allocate label buffer(s)
.
          USE       WORK
LABLBUFR  SK        LLABLUSR+1
.
          END
```

APPENDIX F. GENERATION AND USE OF RPGII

F.1 RPGII Generation For Cartridge Disks

The collection of files which comprise the Datapoint RPGII system are distributed on five cassettes, labeled RPGII CASSETTE 1, RPGII CASSETTE 2, RPGII CASSETTE 3, RPGII CASSETTE 4, and RPGII CASSETTE 5. The first two cassettes are to be MIN-ed onto a disk. The contents of the last three cassettes must be catalogued under DOS using the RPGII Generator. This is done by running the CHAIN RPGIIOUT/CHN which is on CASSETTE 1. In order for the system generation to be successful, there must be at least 50 free files and 1800 free sectors on the drive specified for generation. The actual number of free files and sectors can be determined by running the FREE utility program.

Once it has been determined that there is enough space for the RPGII system, place CASSETTE 1 in the front deck, and type the command: MIN ;AO:DRn. (Where n is the number of the disk where the RPGII system is to be placed.) Use the same command for CASSETTE 2. Now to unload CASSETTES 3, 4, and 5 type in the command CHAIN RPGIIOUT/CHN;ALL,D1=n,D2=n,D3=n,DISK. (Where n is the same number as for CASSETTES 1 AND 2.) The CHAIN will ask that CASSETTE 3 be mounted in the front deck first, then 4, and last CASSETTE 5. Upon completion of CASSETTE 5, the RPGII system generation is complete and ready for operation.

F.2 Selective Generation of RPGII

If, as sometimes happens, a single RPGII system file gets destroyed, it can be selectively recovered by one of two procedures, depending upon the type of file. RPGGEN has the facility to bypass a tape if "*" is keyed in when a cassette is asked for. This is useful for recovering one of the library files from CASSETTES 3, 4 or 5. An overlay file or the pre-processor can be recovered by performing:

MIN and selecting the file that is desired. The following table gives the cassette names and file numbers for single file recovery.

File	Cassette	Recovery
RPGALIB	3	CHAIN RPGIIOUT/CHN;3,D1=n,DISK
RPGBLIB	4	CHAIN RPGIIOUT/CHN;4,D2=n,DISK
RPGCLIB	5	CHAIN RPGIIOUT/CHN;3,D3=n,DISK
RPG/CMD	1	MIN
RPGIIOUT/CHN	1	MIN
RPG/OIG	1	MIN
RPGPRTLB/REL	1	MIN
RPG/OAA	1	MIN
RPG/OAD	1	MIN
RPG/OAE	1	MIN
RPG/OAG	1	MIN
RPG/OAK	1	MIN
RPG/OAM	1	MIN
RPG/OAZ	1	MIN
RPG/OCA	1	MIN
RPG/OCG	1	MIN
RPG/OCK	1	MIN
RPG/OFC	1	MIN
RPG/OFK	1	MIN
RPG/OGA	1	MIN
RPG/OGB	1	MIN
RPG/OGD	1	MIN
RPG/OGF	1	MIN
RPG/OGH	1	MIN
RPG/OGK	1	MIN
RPG/OGM	1	MIN
RPG/OGO	1	MIN

```
RPG/OMB         2       MIN
RPG/OMP         2       MIN
RPG/OPA         2       MIN
RPG/OPX         2       MIN
RPG/OSA         2       MIN
RPG/OSO         2       MIN
RPG/OUA         2       MIN
RPG/OUU         2       MIN
RPG/OWA         2       MIN
RPG/OWL         2       MIN
RPG/OWM         2       MIN
RPG/OWS         2       MIN
RPG/OWU         2       MIN
RPG/OWX         2       MIN
RPG/OZZ         2       MIN
RPGPP/CMD       2       MIN
RPGLDR/OV1      2       MIN
RPGISA/OV1      2       MIN
RPG/OPL         2       MIN
RPG/OPF         2       MIN
RPG/OPS         2       MIN
```

F.3 RPGII Generation For Diskette Systems

Diskette systems will be distributed on diskettes rather than cassettes. A diskette user will receive four diskettes labeled RPGII DISKETTE 1, RPGII DISKETTE 2, RPGII DISKETTE 3, RPGII DISKETTE 4. These diskettes should be placed on drives 0, 1, 2, and 3 respectively.

In addition to the RPGII files, the user will need to build the following DOS utilities on DISKETTE 4:

SORT/CMD and SORT/OV1
INDEX/CMD and INDEX/OV1
REFORMAT/CMD

All utilities must be the proper version for DOS.C. RPGPREP/CMD will be included on DISKETTE 4.

Use the diskette on drive zero for RPG II source and object programs and other user desired utilities (such as EDIT, LIST, etc). The RPG II compiler will use the diskette on drive zero for work files. Note that four drives are required for an RPG II compilation on diskette.

F.4 Compiling an RPG II Program

An RPG II source file is compiled by the RPG II compiler by keying in a command with the following format:

```
RPG srcfil (,objfil)(;(L)(O)(F)(I)(D)(S)(X))
```

where the file names are in standard DOS format. If no object file is specified, the file "srcfil/CMD" will be produced. TXT extension is assumed for the source file, which can be in EDIT or DATAFORM format, and CMD is the default object file extension. The option characters are used as follows:

```
L - List source program and storage map
O - List generated object code      (requires L)
F - List code under false IF's      (requires L,O)
G - List all object bytes           (requires L,O)
I - List included library routines  (requires L,O)
D - Display object code
S - List symbol table               (requires L)
X - List cross references           (requires L)
```

For normal use of the compiler, the L option is all that is required. The O and I options will cause listings of 30 to over 100 pages to be produced, and exist for use by maintenance personnel.

After the RPG II compiler has been invoked, it will ask for a heading line if the L option has been specified. A page heading should be keyed in, terminated with ENTER. This heading will then appear on the top line of every page of the listing. The compiler will then process the source file, and produce an executable object file. During the processing, the top right-hand corner of the display will contain the message "PHASE XX", where XX are two alphabetic characters. These two characters indicate which compilation phase is being executed at any particular time. There are over 40 separate phases, each corresponding to a particular portion of the compilation process. The names of the phases and their functions are described in Appendix G.

Additionally, there are a number of PAUSE points defined in the compiler. These occur before displaying each error note, error text, IDENT, and object code line. At each of these PAUSE points, if the DISPLAY key is depressed, execution will be temporarily suspended until it is released. If the KEYBOARD key is depressed, the machine will BEEP and stop execution. This is another facility

for maintenance personnel, and as such is not required for normal compiler operation. The compiler can be restarted from this suspended condition by pressing ENTER.

F.5 Running a Compiled RPG II Program

The object code generated by RPGII is totally compatible with the instruction set of the Datapoint 1100/2200/5500/6600 series of systems. After compiling the source file, the resulting object file is executed by merely calling for it from the command interpreter. For example suppose the source file TEST/TXT were compiled by the command:

```
RPG TEST;L
```

Then the object file could be executed by the command:

```
TEST
```

The object file can not accept parameters from the command line; all necessary interaction with the user is done under object program control.

F.5.1 DATE Field

If any of the special words: UDATE, UDAY, UMONTH, or UYEAR were used in the source program, the object program will ask for the date, which should be entered as MM/DD/YY. For example, Sept. 5, 1973, is entered as 09/05/73.

F.5.2 External Indicators

If any of the external indicators, U1 to U8, were used in the source program, the object program will ask for their values at the beginning of execution. The values must be entered in binary, with a 0 setting the indicator off and a 1 setting it on, and in the following order:

```
U1 U2 U3 U4 U5 U6 U7 U8
```

Values for indicators not used are not required if there are no used indicators with a higher number. For example, if U1 were the only external indicator used, a valid response is either:

0 or 1.

If U1 and U2 were the only external indicators used, the valid responses are:

00, 01, 10, or 11.

However, if U1 and U3 were used, and no others, the response must be of the form:

0x0, 0x1, 1x0, or 1x1

where x is any character.

F.5.3 Opening Files

Each file opened by the object program causes an opening message to be displayed. In the case of assignable disk files, a message will be displayed, and then the program will wait for a file name to be entered. This name should be in standard DOS format (TXT extension is assumed for data files, ISI extension is assumed for indexed files if extension is not given). If a defined disk file does not exist, an error message will be displayed and the program will then ask for a name as for an assignable file.

F.5.4 Indexing ISAM (Indexed) Files

Indexed files are created in exactly the same format as any fixed format disk file. The data structure is identical and may be processed, disregarding the index, as a simple fixed format file. To permit processing as an indexed file, the INDEX utility is used to create a separate index file. The file is indexed by typing:

```
INDEX datafile(,indexfile);(E)aaa-bbb
```

All parameters within the parenthesis are optional. File names are in standard DOS format. If the indexfile name parameter is

omitted, an index will be created with the name "datafile/ISI". The "E" parameter indicates that the index is in EBCDIC collating sequence. If the "E" is omitted the index will be created in ASCII sequence. The parameter aaa (1-255) is the position of the first character in the key and bbb (1-353) is the position of the end of the key.

The indexfile name should be referenced in the File Description Specifications any time the file is used as an indexed input, update or add file in an RPG program.

F.5.5 Console Input Files

When entering data from the keyboard as an input file, end of file may be entered by depressing the DISPLAY key and the ENTER key simultaneously. This eliminates the necessity of coding an end of file character to set the LR indicator when keyboard input files are used directly (not in DOS CHAIN). When keyboard input is used with DOS CHAIN a record which sets the LR indicator should be used to terminate processing.

APPENDIX G. RPGII REFERENCE TABLES

General System Organization

The first part of the appendix lists the various components of the RPG II system and gives a brief description of the function performed by each phase of the compiler. The two-letter abbreviations appear during compilation in the upper right-hand corner of the display. An RPG compilation is passed through the following phases:

1. Interface Program - common data and code.
2. Enter Phases - read, list, and compress source.
3. Assign Phases - allocate data storage.
4. Diagnostic Phases - finish error checking.
5. Generate Phases - generate object program operations for input, compute, output.
6. Assembly and Library Phases - assemble object text, include necessary routines from library.

Enter Phase Summary

AA - Initialize system, read control card, list, compress and diagnose.

AD - Process file-descriptions - compress information, writing part of card on disk, building file-name table and in-core compression table with the rest of the information.

AE - Process file-extension specifications - compress and write on disk.

AF - Process line-counter specifications.

AG - Process input specifications, generating record and field compressions.

AK - Process calculation specifications - read, list, diagnose and

compress records.

AM - Process output specifications, generating record and field compressions.

AZ - Process user library inclusion, and compile-time tables.

Assign Phase Summary

CA - Assign indicator storage.

CB - Generate indicator table for 'DEBUG' operations.

CC - Define and assign control field storage.

CD - Assign file working areas.

CE - Define and assign match field storage.

CG - Scan extension, input and calculation compressions, define table and field storage, generate table storage.

CH - Scan input, calculation and output compressions, move definitions to compression records.

CI - Generate field storage.

CK - Scan calculation and output compressions, define literals and edit masks in core table.

CL - Move literal definitions into compression records.

CV - Generate literal definitions.

Diagnostic Phase Summary

FC - Diagnose file-descriptions.

FG - Diagnose calculation specifications, check use of table and arrays in calculations, check arithmetic precision.

FK - Diagnose and preprocess input specifications.

FL - Diagnose and preprocess output specifications.

FM - Diagnose file referencing errors.

GA - List error notes in order by line number.

GB to GO - List error texts as needed.

GX - End error text listing, call phase ZZ if fatal error occurred.

Generation Phase Summary

Input Generation Phases:

MB - Generate control field compare.

ME - Generate control field moves.

MG - Generate match field moves.

MK - Generate input field processors.

MP - Generate input, record-tests, and select routines.

MQ - Generate input mainline, end test, and file select sequences.

Calculation Generation Phases:

PA - Generate arithmetic and character (byte) sequences.

PX - Generate detail calculation mainline.

PY - Generate total calcutation mainline.

PZ - Generate RPG subroutines.
Phases PX, PY, and PZ generate calculation control code and the code for other operators.

Output Generation Phases:

SA - Generate output field processors.

SO - Generate heading/detail output mainline.

SP - Generate total output mainline.

SQ - Generate overflow output processor.

SR - Generate exception output processor.

SS - Generate put routines.

Final Generation Phases:

UA - Generate file description blocks.

UF - Generate open mainline.

UG - Generate end mainline.

UU - Generate assembly parameters.

UV - Link generated segments with the fixed library.

Assembly Phases:

WA - Initialize assembly, set up symbol table.

WM - Assembly passes 1 and 2.

WL - Scan library and include referenced segments.

WS - Sort dictionary.

WU - Dictionary listing.

WX - Cross reference sort and listing.

ZZ - RPG Close phase.

RPG Preprocessor:

(PP) - Preprocess ASM text to object text.

Format of File Description Blocks

A partial description of the format of a file description block is given symbolically in this section. For a complete and absolute listing look at an assembly listing of the second object program segment. All symbols defined below have blank qualification, and are input to the pre-processor as ':XXXXXX'. Symbols beginning with the letter 'D' are byte displacements relative to the start of an FDB entry. Symbols beginning with 'M' are masks for parts of a byte. For example the displacement of the two-byte record length is 'DRLNFD', and the mask for the end-of-file flag is 'MEOFFD'.

Name	Purpose
DFLGFD	Displacement of the three-byte flag field.
	------First Flag Byte------
MEOFFD	End of file.
MCLSFD	File closed.
MPUTFD	Output record ready for update or chained output file. Output pending.
MPBFFD	Buffer output pending (update files).
MADDFD	ADD file.
	------Second Flag Byte------
MCHNFD	CHAIN file.
MWACFD	Output/Update file.
MRACFD	Input/Update file
	------Third Flag Byte--------
MISAFD	Indexed (ISAM) File.
	------Basic Description------
DRLNFD	Logical record length (two-bytes - MSB,LSB).
DBLFFD	Blocking factor (one-byte).

DFWAFD Address of File (record) Working Area (two-bytes - LSB,MSB).

DFNMFD Address of internal file name (LSB,MSB)

Format of Table Description Blocks

The format of table description blocks is given symbolically in this section. For an absolute listing look at the assembly listing for the segment named 'COMMON TABLE PROCESSOR'. The conventions are the same as for file description blocks.

Name	Purpose
DTHATD	Address of table/array storage area (two-bytes - LSB,MSB).
DFLDTD	Address of current selected entry (LSB,MSB).
DNELTD	Number of elements (two-bytes - MSB,LSB).
DELNTD	Length of an element (one-byte).
DCENTD	Index of current entry (two-bytes - MSB,LSB).
DCEATD	Address of current entry (two-bytes - LSB,MSB)

Note: By convention, addresses are stored in the least-significant, then the most-significant byte order, whereas two-byte binary numbers are stored in the most-significant, least-significant order. For Boolean (bit) values, true = 1 and false = 0.

APPENDIX H. RPGII COMPILE TIME MESSAGES

No.	W/F	Card	Text
1	W	H	CC7-9 SHOULD BE BLANK.
2	W	H	INVALID CORE SIZE, CC12-14.
3	W	H	INVALID DEBUG CODE, C15.
4	W	H	CC16-25 SHOULD BE BLANK.
5	F	F	INVALID OR BLANK FILE NAME, CC7-14.
6	F	F	INVALID OR BLANK FILE TYPE, C15.
7	F	F	INVALID OR BLANK FILE DESIGNATION, C16.
8	W	F	INVALID PROCESS TO END OF FILE ENTRY, C17.
9	W	F	SEQUENCE ENTRY INVALID, OR SPECIFIED WITH FILE TYPE NOT PRIMARY OR SECONDARY, C18.
10	W	F	INVALID FORMAT ENTRY, C19.
11	W	F	INVALID BLOCK LENGTH, CC20-23.
12	F	F	INVALID RECORD LENGTH, CC24-27.
13	W	F	INVALID MODE OF PROCESSING ENTRY, C28.
14	W	F	CC29-31 SHOULD BE BLANK.
15	W	F	INVALID OVERFLOW ENTRY, CC33-34.
16	W	F	OVERFLOW SPECIFIED WITH DEVICE OTHER THAN PRINTER, CC33-34.
17	W	F	CC35-38 SHOULD BE BLANK.
18	W	F	INVALID EXTENSION OR LINE COUNTER ENTRY, C39.
19	W	F	LINE COUNTER SPECIFIED WITH DEVICE OTHER THAN PRINTER, C39.

20	F	F	INVALID DEVICE CODE, CC40-46.
21	W	F	CC47-52 SHOULD BE BLANK.
22	W	F	CC53-59 SHOULD BE BLANK UNLESS DEVICE SPECIFIED AS 'SPECIAL'.
23	W	F	INVALID 'K' ENTRY, C53.
24	W	F	INVALID UNLESS SPECIAL DEVICE NAME, CC54-59.
25	W	F	INVALID NAME FOR SPECIAL DEVICE I/O ROUTINE, CC54-59.
26	W	F	CC60-65 SHOULD BE BLANK.
27	W	F	C66 SHOULD BE BLANK.
28	W	F	INVALID ADDITIONS ENTRY, C66.
29	W	F	CC67-69 SHOULD BE BLANK.
30	W	F	C70 SHOULD BE BLANK UNLESS TAPE DEVICE.
31	W	F	INVALID REWIND OPTION, C70.
32	W	F	INVALID FILE CONDITION ENTRY, CC71-72.
33	W	F	CC73-74 SHOULD BE BLANK.
34	W	E	CC7-10 SHOULD BE BLANK.
35	F	E	INVALID OR UNRECOGNIZABLE 'FROM' FILE NAME, CC11-18.
36	F	E	INVALID OR UNRECOGNIZABLE 'TO' FILE NAME, CC19-26.
37	F	F	CHAINED, INDEXED OUTPUT FILE C66 MUST BE A.
38	F	E	INVALID TABLE/ARRAY NAME, CC27-32.
39	F	E	INVALID NUMBER OF ENTRIES PER RECORD, CC33-35.
40	F	E	INVALID NUMBER OF ENTRIES FOR TABLE/ARRAY, CC36-39.
41	F	E	INVALID LENGTH OF ENTRY, CC40-42 OR CC52-54.
42	W	E	INVALID FORMAT ENTRY, C43 OR C55 - IGNORED.

43	F	E	INVALID DECIMAL POSITIONS ENTRY, C44 OR C56.
44	W	E	INVALID SEQUENCE ENTRY, C45 OR C57 - IGNORED.
45	F	E	INVALID ALTERNATE TABLE NAME, CC46-51.
46	F	E	EXECUTION-TIME TABLES NOT ALLOWED.
47	F	E	ALTERNATE TABLE SPECIFICATION NOT ALLOWED WITH EXECUTION-TIME ARRAYS.
48	F	E	'FROM' FILE MUST BE AN INPUT-TABLE FILE.
49	F	E	'TO' FILE MUST BE ORDINARY OUTPUT OR OUTPUT-TABLE FILE.
50	F	E	'FROM' FILE TOO SHORT FOR TABLE RECORD.
51	F	E	'TO' FILE TOO SHORT FOR TABLE RECORD.
52	F	E	MORE THAN 8192 BYTES OF TABLE STORAGE ALLOCATED.
53	W	H	INVALID COLLATE SEQUENCE ENTRY, CC26.
54	W	H	CC27-74 SHOULD BE BLANK.
55	F	L	INVALID OR UNDEFINED FILE NAME, CC7-14.
56	F	L	FILE MUST BE ASSIGNED TO THE PRINTER.
57	F	L	FORM LENGTH, CC15-17, INVALID OR >99.
58	W	L	CC18-19 SHOULD CONTAIN 'FL'.
59	F	L	OVERFLOW LINE, CC20-22, INVALID OR >99.
60	W	L	CC23-24 SHOULD CONTAIN 'OL'.
61	W	L	CC25-74 SHOULD BE BLANK.
62	F	L	OVERFLOW LINE IS GREATER THAN FORM LENGTH.
63	F	L	MULTIPLE LINE COUNTER SPECIFICATION LINES.
64-65			UNASSIGNED.
66	F	I	MIXED RECORD AND FIELD DATA, CC7-42, 43-74.

67	W	I	CC7-14 SHOULD BE BLANK FOR 'AND' AND 'OR' CARDS.
68	F	I	FILE NAME, CC7-14, NOT SPECIFIED IN FILES SECTION.
69	F	I	INVALID SEQUENCE ENTRY, CC15-16.
70	W	I	INVALID SEQUENCE ENTRY, C17.
71	W	I	INVALID OPTIONAL SEQUENCE ENTRY, C18.
72	W	I	INVALID RECORD INDENTIFYING INDICATOR, CC19-20.
73	F	I	INCOMPLETE RECORD INDENTIFYING CODE, CC25-27, CC32-34, OR CC39-41.
74	F	I	INVALID POSITION ENTRY, C24, 31 OR 38.
75	W	I	INVALID NOT ENTRY, C25, 32 OR 39.
76	W	I	INVALID C/Z/D ENTRY, C26, 33 OR 40.
77	W	I	STACKER SELECT NOT IMPLEMENTED, C42.
78	F	I	INVALID FORMAT ENTRY, C43.
79	F	I	INVALID FROM LOCATION ENTRY, CC44-47.
80	F	I	INVALID 'TO' LOGATION ENTRY, CC48-51.
81	F	I	NEGATIVE FIELD LENGTH, CC44-51.
82	F	I	INVALID DECIMAL POSITION ENTRY, C52.
83	F	I	INVALID FIELD NAME, CC53-58.
84	F	I	INVALID CONTROL LEVEL ENTRY, CC59-60.
85	F	I	INVALID MATCH FIELD ENTRY, CC61-62.
86	F	I	INVALID FIELD RECORD RELATION ENTRY, CC63-64.
87	F	I	INVALID FIELD INDICATOR(S), CC65-70.
88	W	I	CC71-74 SHOULD BE BLANK.
89	F	I	RECORD DESCRIPTION ILLEGAL.
90	F	C	INVALID ENTRY IN CC7-8.

91	F	C	INVALID NOT ENTRY IN CC9, 12, 15.
92	F	C	INVALID INDICATOR IN CC10-11, CC13-14, OR CC16-18.
93	F	C	INVALID FACTOR 1 ENTRY, CC18-27.
94	F	C	UNRECOGNIZABLE OPERATION, CC28-32.
95	F	C	INVALID FACTOR 2 ENTRY, CC33-42.
96	F	C	INVALID RESULT FIELD, CC43-48.
97	W	C	HALF-ADJUST ENTRY IN C53 UNRECOGNIZABLE OR NOT ALLOWED.
98	F	C	INVALID RESULT INDICATOR ENTRY, CC54-59.
99	F	C	INVALID LENGTH IN CC49-51.
100	F	C	DECIMAL POSITIONS INVALID, C52.
101	F	C	CC49-52 SHOULD BE BLANK IF NO RESULT FIELD IS SPECIFIED.
102	F	C	INVALID FILE NAME IN CC33-42.
103	F	C	INVALID LITERAL SPECIFICATION.
104	F	C	INVALID BIT MASK, CC33-42.
105	F	C	CONDITION INDICATORS NOT ALLOWED WITH 'TAG', 'RLABL', 'BEGSR', OR 'ENDSR' OPERATIONS.
106	F	F	INVALID KEY LENGTH, CC29-30.
107	F	F	INVALID KEY STARTING POSITION, CC35-38.
108	W	F	ISAM OUTPUT FILE MUST BE "INDEXED" AFTER CREATION.
109	F	F	SEQUENTIAL WITHIN LIMITS VALID ON INDEXED FILES, C28.
110	F	O	RECORD AND FIELD DATA IN SAME LINE.
111	F	O	INVALID OR UNSPECIFIED FILE NAME, CC7-14.
112	F	O	INVALID 'AND' OR 'OR' ENTRY, CC14-16.

113	W	O	CC17-22 SHOULD BE BLANK ON 'AND' OR 'OR' LINES.
114	F	O	INVALID LINE TYPE, C15.
115	W	O	INVALID FETCH OVERFLOW ENTRY, C16.
116	W	O	INVALID 'SPACE BEFORE' ENTRY, C17.
117	W	O	INVALID 'SPACE AFTER' ENTRY, C18.
118	W	O	INVALID 'SKIP BEFORE' ENTRY, CC19-20.
119	W	O	INVALID 'SKIP AFTER' ENTRY, CC21-22.
120	W	O	INVALID 'NOT' ENTRY, C23, 26, OR 29.
121	F	O	INVALID FIELD CONDITIONING INDICATOR.
122	F	O	INVALID FIELD NAME.
123	F	O	INVALID EDIT CODE, C38.
124	W	O	C38 SHOULD BE BLANK IF NO FIELD NAME SPECIFIED.
125	W	O	INVALID 'BLANK AFTER' ENTRY, C39.
126	F	O	INVALID END POSITION ENTRY, CC40-43.
127	W	O	INVALID DATABUS FORMAT ENTRY, C44.
128	W	O	C44 SHOULD BE BLANK IF NO FIELD NAME SPECIFIED.
129	W	O	LITERAL NOT STARTED WITH A QUOTE, C45.
130	F	O	EMBEDDED SINGLE QUOTE, CC45-70.
131	F	O	REMAINDER OF CC45-70 NOT BLANK AFTER LITERAL.
132	W	O	LITERAL TOO LONG, CC70-80.
133	W	O	INVALID COMBINATIONS OF $ AND * IN CC45-47.
134	W	O	CC45-47 NOT $, *, OR BLANK..
135	W	O	INVALID EDIT WORD.
136	W	O	CC71-74 SHOULD BE BLANK.

137	W	H	MULTIPLE HEADER CARDS.
138	W	H	INVALID ENTRY, C10.
139	W	H	INVALID ENTRY, C11.
140	W	F	FILE NAME TABLE FULL.
141	W	F	DUPLICATE FILE NAME, CC7-14.
142	W	F	INVALID TYPE FOR GIVEN DEVICE CODE, C15.
143	F	F	INVALID DESIGNATION FOR GIVEN DEVICE CODE, C16.
144	W	F	INVALID FILE FORMAT FOR THIS DEVICE, C19.
145	W	F	BLOCKING FACTOR GREATER THAN 255, CC20-27.
146	W	F	BLOCK LENGTH LESS THAN RECORD LENGTH, CC20-27.
147	W	F	BLOCK LENGTH NOT A MULTIPLE OF RECORD LENGTH, CC20-27.
148	F	F	BLOCK LENGTH GREATER THAN ALLOWED FOR GIVEN DEVICE, CC20-23.
149	W	F	BLOCK LENGTH NOT EQUAL TO RECORD LENGTH, CC20-27.
150	F	F	NO FILE DESCRIPTION SPECIFICATIONS.
151	F	F	NO PRIMARY OR SECONDARY FILE SPECIFIED.
152	W	F	SECONDARY FILE PRECEDES PRIMARY FILE.
153	W	F	MULTIPLE PRIMARY FILES. SECONDARY ASSUMED.
154	W	F	AN EXTENSION, C39, MUST BE SPECIFIED FOR TABLE FILES.
155	W	F	EXTENSION, C39, INVALID WITH GIVEN DEVICE OR NON-TABLE FILE.
156	F	F	DEVICE ASSIGNED TO MORE THAN ONE FILE.
157	W	F	CONDITION INDICATOR, CC71-72, INVALID FOR TABLE FILE.

158	W	F	FILE NAME ASSIGNED BUT NEVER USED IN PROPER SECTION.
159	F	F	SEQUENCE, C18, INVALID WITH NO MATCH FIELDS.
160	W	F	SEQUENCE, C18, MUST BE SPECIFIED WITH MATCH FIELDS.
161	F	F	EXTENSION OR LINE COUNTER SPECIFICATION MISSING, C39.
162	W	F	EXTENSION OR LINE COUNTER SPECIFICATION FOUND FOR THIS FILE, BUT C39 IS NOT 'E' OR 'L'.
163	F	F	OUTPUT REFERENCE REQUIRED FOR UPDATE FILE.
164	W	F	CC7-52 ON CONTINUATION CARD SHOULD BE BLANK.
165	F	F	CC54-59 ON CONTINUATION CARD NOT EQUAL 'ASCII'.
166	W	F	CC60-74 ON CONTINUATION CARD SHOULD BE BLANK.
167	F	F	TAPE RECORD LENGTH LESS THAN 18.
168	F	F	ADDITIONS INVALID FOR FILE OR DEVICE, C66.
169	W	F	ALL PRIMARY AND SECONDARY FILES CONDITIONED.
170	F	F	CALCULATION REFERENCE REQUIRED FOR CHAIN OR DEMAND FILES.
171			UNASSIGNED.
172	W	F	INVALID ENTRY IN C53. 'A' ASSUMED.
173	W	F	INVALID LABEL EXIT.
174	W	F	C53 SHOULD BE BLANK.
175	W	F	CC60-65 SHOULD BE BLANK ON ASCII CARD.
176	F	F	UNRECOGNIZABLE DISK CONTINUATION OPTION IN CC54-59.
177	F	F	UNRECOGNIZABLE ENTRY IN CC60-65.
178			UNASSIGNED.
179	W	F	VARIABLE BLOCKING INVALID FOR THIS FILE TYPE, FIXED BLOCKING ASSUMED.

180	W	F	CC31-32 SHOULD CONTAIN 'I ' OR 'AI' FOR RANDOM PROCESSING.
181	F	F	CC29-32 SHOULD CONTAIN ' 3IT' FOR ADDROUT FILES.
182	F	F	ADDROUT FILES MUST BE FIXED-FORMAT, UNBLOCKED FILES WITH RECORD LENGTH EQUAL TO 3.
183	F	F	THIS FILE MUST BE CONTROLLED BY AN ADDROUT FILE.
184	F	F	ADDROUT FILE MUST CONTROL A PRIMARY/SECONDARY FILE.
185	F	F	CORRESPONDING ADDROUT AND PRIMARY/SECONDARY FILES MUST HAVE THE SAME EXTERNAL INDICATOR CONDITION.
186	F	F	UNRECOGNIZABLE PRINTER CONTINUATION.
187	F	F	RECORD ADDRESS TYPE SHOULD BE 'A' OR 'I', CC31.
188	F	F	UNRECOGNIZABLE RECORD ADDRESS TYPE, CC31.
189	F	F	UNRECOGNIZABLE FILE ORGANIZATION, C32.
190	W	F	CC29-30 SHOULD BE BLANK FOR FILE PROCESSED BY ADDROUT FILE.
191			UNASSIGNED
192	F	E	ISAM FILE CAN'T BE CONTROLLED BY TAG FILE.
193	F	E	NO DATA FOR COMPILE TIME TABLE.
194	F	E	MORE THAN ONE ADDROUT FILE CONTROLS THIS PRIMARY/SECONDARY FILE.
195	F	E	MORE THAN ONE PRIMARY/SECONDARY FILE IS CONTROLLED BY THIS ADDROUT FILE.
196	F	E	'FROM' FILE MUST BE AN ADDROUT FILE.
197	F	E	'TO' FILE MUST BE A RANDOMLY-PROCESSED PRIMARY/SECONDARY FILE.
198	F	E	TOO MANY TABLES DEFINED.
199	F	E	MULTIPLE TABLE DEFINITIONS.
200	F	I	NO INPUT SPECIFICATION SECTION.

201	F	I	FIELD PRECEDES FIRST RECORD.
202	F	I	FILE ASSIGNED IS NOT AN INPUT OR UPDATE FILE.
203	F	I	MORE THAN 256 INPUT RECORD TYPES SPECIFIED.
204	F	I	INPUT RECORDS MUST BE DESCRIBED IN SAME ORDER AS FILES.
205	F	I	CONTROL LEVEL SPECIFICATION INVALID WITH FILE TYPE.
206	F	I	MATCH FIELD SPECIFICATION INVALID WITH FILE TYPE.
207	F	I	MORE THAN 255 RECORD ID TESTS FOR THIS RECORD.
208	F	I	FIRST LINE IS AN 'AND' OR 'OR' LINE.
209	F	I	MULTIPLY DEFINED FIELD.
210	F	I	LENGTH OF CONTROL FIELDS GREATER THAN 255 BYTES.
211	F	I	LENGTH OF MATCH FIELDS GREATER THAN 255 BYTES.
212	F	I	MORE THAN 32 'AND' LINES.
213			UNASSIGNED.
214	W	I	'AND' LINE FOLLOWS LINE WITHOUT RECORD ID CODES.
215	W	I	NO FIELDS DESCRIBED FOR PREVIOUS RECORD.
216	W	I	NUMERIC SEQUENCE ENTRIES NOT IN ORDER, OR FIRST ENTRY NOT EQUAL 01.
217	F	I	CC17-18 SHOULD BE BLANK FOR ALPHABETIC SEQUENCE.
218	W	I	CC17-20 SHOULD BE BLANK FOR 'AND' LINES.
219	W	I	CC17-18 SHOULD BE BLANK FOR 'OR' LINES.
220	F	I	LENGTH OF NUMERIC FIELD GREATER THAN 15, OR LENGTH OF ALPHABETIC FIELD GREATER THAN 255.
221	W	I	DECIMAL POSITION ENTRY INVALID FOR ARRAY.
222	F	I	NUMBER OF DECIMAL POSITIONS EXCEEDS FIELD LENGTH.
223	F	I	TABLE NAME INVALID FOR A FIELD NAME.

224	F	I	'AND' LINES INVALID WITH LOOK-AHEAD RECORD.
225	F	I	CC17-18, 21-42, AND 59-74 INVALID WITH LOOK-AHEAD.
226	F	I	FIELD LOCATION ENTRIES EXCEED RECORD LENGTH.
227	F	I	FIELD NAME IS A RESERVED WORD OTHER THAN 'PAGE'.
228	F	I	CONTROL AND MATCH SPECIFICATIONS INVALID FOR ARRAYS.
229	F	I	LOOK-AHEAD INVALID WITH CHAIN OR DEMAND FILES OR WITH THIS DEVICE.
230	F	I	NO FIELDS SPECIFIED FOR LOOK-AHEAD RECORD.
231	F	I	ARRAY LENGTH EXCEEDS OR IS NOT A MULTIPLE OF LENGTH IN EXTENSION SPECIFICATION.
232	F	I	INCONSISTENT LENGTHS FOR CONTROL OR MATCHING FIELDS OF ONE LEVEL.
233	F	I	INVALID SPLIT CONTROL FIELD SPECIFICATION.
234	W	I	CONTROL OR MATCHING FIELDS SPECIFIED AS ALPHA AND NUMERIC.
235	F	I	ALL VALID MATCH LEVELS WERE NOT REFERENCED IN THE LAST RECORD GROUP.
236	F	I	CONTROL OR MATCH FIELDS WITHOUT FRR MUST PRECEDE THOSE WITH FRR.
237	F	I	CONTROL OR MATCH FIELDS WITH FRR MUST BE GROUPED BY FRR.
238	F	I	FIELD RECORD RELATION INDICATOR USED IMPROPERLY WITH CONTROL OR MATCH FIELDS.
239	W	I	INDICATOR ASSIGNED BUT NOT USED.
240	F	I	INDICATOR USED, BUT NOT ASSIGNED.
241	F	I	FIELD LENGTH NOT MULTIPLE OF TABLE ENTRY LENGTH.
242	F		INDEX FIELD NOT NUMERIC OR DECIMAL POSITIONS > 0.
243	F		LITERAL INDEX OUT-OF-BOUNDS.

244	F		CONFLICT IN TAPE DENSITY.
245	F	F	FILE ORGANIZATION SHOULD BE BLANK, CC32.
246	F	F	FILE ORGANIZATION SHOULD BE 'I' OR 'T', CC32.
247	F	F	RECORD ADDRESS TYPE AND FILE ORGANIZATION ARE INCOMPATIBLE, CC31-CC32.
248-249			UNASSIGNED.
250	F	C	INVALID FILE FOR FORCE.
251	F	C	INVALID FILE FOR READ.
252	F	C	INVALID CHAINING FIELD.
253	F	C	INVALID FILE IN CHAIN.
254	F	C	DEBUG FILE NOT OUTPUT FILE.
255	W	C	DEBUG OPERATIONS IN PROGRAM IGNORED.
256	W	C	DEBUG OPTION WITHOUT DEBUG OPERATION.
257	F	C	DIFFERENT DEBUG FILES.
258	F	C	INVALID FILE FOR DSPLY.
259	W	C	CALCULATIONS CONSIST ONLY OF SUBROUTINES.
260	F	C	SUBROUTINE MUST BEGIN WITH 'BEGSR' OPERATION.
261	F	C	TOTAL OR DETAIL RECORD OUT OF SEQUENCE.
262	F	C	ARRAY IMPROPERLY USED IN RESULT FIELD.
263	F	C	FACTOR 1 OR 2 MAY NOT BE AN ARRAY UNLESS RESULT FIELD IS.
264	F	C	RECORD LENGTH FOR DEBUG FILE IS TOO SMALL.
265	F	C	FACTOR 1 IN 'DEBUG' SHOULD BE LESS THAN NINE BYTES LONG.
266	F	C	SUBROUTINE MUST END WITH 'ENDSR'.
267	F	C	RESULT FIELD MUST BE ALPHANUMERIC.

268	F	C	FACTOR 2 MUST BE ALPHANUMERIC.
269	F	C	FACTORS 1 & 2 MUST HAVE SAME TYPE.
270	F	C	BIT OPERATIONS TAKE SINGLE-BYTE FIELDS.
271	F	C	FACTOR 2 IN 'LOKUP' MUST BE A TABLE OR ARRAY.
272	F	C	CORRESPONDING TABLE MAY NOT BE USED WITH ARRAY LOOK-UP.
273	F	C	RESULT FIELD IN LOKUP MUST BE A TABLE.
274	F	C	FACTOR 1 MUST HAVE SAME LENGTH AS FACTOR 2 IN LOOK-UP.
275	F	C	'BEGSR' IN MIDDLE OF SUBROUTINE.
276	F	C	'RLABL' MUST IMMEDIATELY FOLLOW 'EXIT'.
277	F	C	INVALID LABEL OPERAND.
278	F	C	'BEGSR' OR 'ENDSR' IN DETAIL OR TOTAL RECORDS.
279	F	C	FACTOR 1 MUST BE NUMERIC.
280	F	C	FACTOR 2 MUST BE NUMERIC.
281	F	C	RESULT FIELD MUST BE NUMERIC.
282	W	C	HALF-ADJUST NOT NEEDED, ENTRY ASSUMED BLANK.
283	W	C	COMPUTED RESULT MAY OVERFLOW RESULT FIELD.
284	F	C	FACTOR 2 IN 'XFOOT' MUST BE AN ARRAY.
285	F	C	'MVR' MUST FOLLOW 'DIV'.
286	F	C	HALF-ADJUST ON PREVIOUS 'DIV' ILLEGAL WITH 'MVR'.
287	F	C	FACTOR 2 NOT A PROCESS WITHIN LIMITS FILE.
288	F	C	FACTOR 1 IS NOT A VALID KEY.
289-327			UNASSIGNED
328	W	O	DATAPOINT COMPATIBLE FIELD SHOULD BE NUMERIC.

329	F	O	NEITHER FIELD NAME OR LITERAL IS PRESENT.
330	F	O	'AND' OR 'OR' LINE NOT PRECEDED BY RECORD LINE.
331	W	O	SPACE AND SKIP INVALID WITH DEVICE OTHER THAN CONSOLE OR PRINTER.
332	W	O	SKIP ENTRY GREATER THAN FORM LENGTH.
333	W	O	FETCH OVERFLOW INVALID FOR DEVICE OTHER THAN PRINTER.
334	F	O	OVERFLOW INDICATOR INVALID FOR EXCEPTION LINE.
335	W	O	FETCH OVERFLOW INVALID WITH OVERFLOW INDICATORS.
336	F	O	OVERFLOW INDICATOR USED IS NOT ASSIGNED TO THIS FILE.
337	W	O	1P INDICATOR INVALID ON TOTAL OR EXCEPTION LINES.
338	W	O	FETCH OVERFLOW INVALID WITH 1P INDICATOR.
339	W	O	SPACE BEFORE OF 0 INVALID FOR CONSOLE.
340	F	O	INVALID INDICATORS USED WITH 1P INDICATOR.
341	F	O	END POSITION GREATER THAN RECORD LENGTH.
342	F	O	LENGTH OF ARRAY, ELEMENT, OR FIELD EXCEEDS RECORD LENGTH.
343	F	O	END POSITION TOO LOW.
344	W	O	ALL INDICATORS MISSING OR NEGATIVE IN PREVIOUS RECORD.
345	W	O	ALL INDICATORS MISSING ON THIS LINE.
346	F	O	INVALID EDIT WORD SIZE.
347	F	O	EDIT CODE INVALID WITH ALPHA FIELD OR CONSTANTS OTHER THAN $ OR *.
348	F	O	CONSTANT INVALID WITH EDIT CODES X, Y OR Z.
349	F	O	INVALID FIELD LENGTH FOR Y EDIT CODE.

350	F	O	DECIMAL POSITIONS INVALID WITH Y EDIT CODE.
351	F	O	INVALID FILE TYPE FOR OUTPUT RECORD.
352	W	O	BLANK AFTER INVALID WITH RESERVED WORD OTHER THAN 'PAGE'.
353	F	O	MORE THAN 32 'AND' OR 'OR' LINES.
354	W	O	BLANK AFTER SPECIFIED FOR A CONSTANT.
355	F	O	ARRAY INDEX EXCEEDS NUMBER OF ELEMENTS.
356	W	O	BLANK AFTER INVALID WITH LOOK-AHEAD.
357	W	O	INDICATOR ASSIGNED BUT NEVER USED.
358	F	O	INDICATOR USED BUT NEVER ASSIGNED.
359	F	O	FIELD NAME USED BUT NOT DEFINED.
360	F	O	TABLE OR ARRAY NAME USED AS INDEX.
361	F	O	NUMBER OF DECIMAL POSITIONS EXCEEDS FIELD LENGTH.
362	W	O	L0-L9 IN 'OR' RELATIONSHIP WITH LR.
363	F	O	ADDITIONS INVALID WITH 'AND' OR 'OR' LINES.
364	W	O	FOR ADD FILES, EACH RECORD MUST HAVE 'ADD' IN CC16-18.
365	F	O	ADDITIONS INVALID WITH FILES EXCEPT SEQUENTIAL DISK FILES.
366	F	O	'T' IN C15, OR E WITH L0-L9 INVALID WITH UPDATE FILES.
367	F	O	FIELD LINE PRECEEDS FIRST RECORD LINE.
368			UNASSIGNED.
369	F	O	MORE THAN 255 OUTPUT RECORD TYPES SPECIFIED.
370	F	O	NO OUTPUT SPECIFICATION SECTION FOUND.
371	F	O	RECORDS MUST BE IN SAME ORDER AS FILES.

372	F	O	H, D, T AND E LINES MUST BE IN ORDER.
373	F	O	FIELD LENGTH DOES NOT CORRESPOND TO NUMBER OF REPLACEABLE CHARACTERS IN EDIT WORD.
374	W	O	EXCEPT RECORD WITHOUT 'EXCPT' OPERATION.
375	F	O	EDIT CODE INCOMPATIBLE WITH OPTIONS USED IN CC45-47.
376	F	O	NO REPLACEABLE CHARACTERS IN EDIT WORD.
377	F	O	FILE IS NOT A DISK ADD FILE, CC16-18.
378-379			UNASSIGNED.
380	W		SEQUENCE NUMBERING ERROR IN SOURCE RECORDS, CC1-5. SEE NOTE BELOW.
381	F		RECORD TYPE OUT OF SEQUENCE IN SOURCE RECORDS, C6.
382	F		INVALID CHARACTERS IN (MAIN) OPERAND NAME.
383	F		INVALID CHARACTERS IN INDEX OF OPERAND.
384	F		INDEX IS INVALID WITH THIS OPERAND.
385-399			UNASSIGNED.
400	F		SEQUENCE ERROR IN COMPILE-TIME TABLE OR ARRAY.
401	F		NUMERIC FIELD ERROR IN COMPILE-TIME TABLE OR ARRAY.
402	W		END OF FILE FOLLOWS '**b' RECORD.
403	F		NO FILE NAME IN LIBRARY INCLUSION RECORD.
404	F		USER LIBARY FILE DOES NOT EXIST.
405	W		DELIMITER CARD FORMAT ERROR.
406	W		SHORT COMPILE-TIME TABLE OR ARRAY.
407	F		TABLE/ARRAY FILLED.
408	F		ALTERNATE TABLE BUFFER FULL.
409	W		NO '**b' RECORD FOLLOWS LIBRARY INCLUSION RECORD.

410 W NO COMPILE-TIME TABLE/ARRAY FOR DATA.

411 W INVALID LIBRARY FILE NAME.

412 F COMPILE-TIME TABLE/ARRAY DATA RECORD LENGTH > 80.

***NUMBER FOR CONVERSION WON'T FIT-This is an indication
that the RPG II Compiler has failed. Report problem to Datapoint.

NOTE: A SEQUENCE ERROR WILL OCCUR IF A BLANK RECORD IS PRESENT IN THE SOURCE CODE.

APPENDIX I. RPGII OBJECT (EXECUTION) TIME MESSAGES

During the execution of an RPG II object program, messages will be displayed on the screen either to request input from the user or merely to inform him of certain actions being performed. In addition, error messages will be displayed if abnormal situations are encountered. The following list of messages are all those which could possibly occur during an RPG II object program execution. The list includes an explanation of each message, the action taken by the object program after displaying the message, and an explanation of the response from the user, if necessary. Note that any data typed in by the user must be terminated with the ENTER key.

OTM01: *** INVALID INPUT

Explanation: Error message displayed if the data given in response to either the DATE message (OTM23) or the INDICATOR message (OTM24) is not in the correct format.

Program Action: The appropriate request for data is made again.

User Response: Re-enter the data requested.

OTM02: ABSENT RECORD AND NO INDICATOR

Explanation: Error message displayed if a record was not found during a CHAIN operation and no indicator was specified in columns 54-55.

Program Action: Display message OTM10.

OTM03: ASC/BIN ERROR

Explanation: Error message displayed whenever an ASCII numeric string, being converted to an internal binary number, contains either no digits or more than 6 digits.

Program Action: Display message OTM10.

OTM04: ATTEMPT TO READ PAST END OR FROM CLOSED FILE

Explanation: Error message displayed if the READ operation was attempted on a file which was at end-of-file or was closed.

Program Action: Display message OTM10.

OTM05: (BAD FILE SPEC)

Explanation: Error message displayed if a specification of a file during the file opening sequence is incorrect.

Program Action: A file name will be asked for again.

User Response: A correct file name should be entered.

OTM06: BAD RECORD NUMBER

Explanation: Error message displayed if a disk access is attempted to a record number less than zero or to a record number above the currently allocated space.

Program Action: Display the program name of the file and then message OTM10.

OTM07: BAD TABLE OR ARRAY SEQUENCE

Explanation: Error message displayed if the sequence of data being read into a pre-execution time table or array is not as specified on the Extension Specs.

Program Action: Display message OTM10.

OTM08: BIN/ASC ERROR

Explanation: Error message displayed whenever an ASCII numeric string, destined to receive a converted binary value, is of zero length.

Program Action: Display message OTM10.

OTM09: BIN/DEC ERROR

Explanation: Error message displayed if the length of a field being used to store an index after an array LOKUP operation is not long enough to contain the result.

Program Action: Display message OTM10.

OTM10: BYPASS/CANCEL/ABORT

Explanation: General error message displayed after many other specific error messages to give the user the option as to what action should be taken by the program.

Program Action: Wait for input.

User Response: Three different responses are allowed. Typing a 'B' will cause the program to bypass the current cycle and read the next record. Typing a 'C' will cancel program execution and close all the files. Typing an 'A' will immediately return control to the operating system without closing any files.

OTM11: CANCEL/ABORT

Explanation: Similar to OTM10. Displayed in place of OTM10 if the program is in the last cycle.

Program Action: Wait for input.

User Response: Same as for OTM10 and OTM55, except the 'R' and 'B' responses are not valid.

OTM12: CANNOT LOAD ISAM OVERLAY

Explanation: Error message displayed if the program tries to load the overlay RPGISA/OV1 to add records to an indexed file, and is unable to do so.

Program Action: Execution is terminated.

User Response: The file RPGISA/OV1 should be re-installed from the RPG II generation tapes and the program re-run.

OTM13: CHAINING ERROR

Explanation: Error message displayed if Factor-1 used in a CHAIN operation contains invalid data. See the CHAINED RECORD PROCESSING table in Chapter 8.

Program Action: Display the progam name of the file and then message OTM10.

OTM14: DATABUS INPUT ERROR

Explanation: Error message displayed if the format of a number being read from an input file is incorrect (not correct Databus format).

Program Action: Display message OTM10.

OTM15: DEBUG:

Explanation: Message written on an output file whenever the DEBUG operation is executed.

Program Action: A series of records are written on the output file, according to the format of the DEBUG operation.

OTM16: DEC/BIN ERROR

Explanation: Error message displayed if a field or literal, used as an array index or a record address of a Chain file, is less than zero or greater than 65535.

Program Action: Display message OTM10.

OTM17: DELETE PROTECT

Explanation: Error message displayed if an attempt is made to shorten a disk file which is delete protected.

Program Action: Display the program name of the file and then message OTM10.

OTM18: DRIVE OFF LINE

Explanation: Error message displayed if an attempt is made to access a disk drive which is either physically absent or off line.

Program Action: Display the program name of the file and then message OTM10.

OTM19: DSPLY

Explanation: Message displayed whenever the DSPLY operation is executed.

Program Action: The contents of one or two fields are subsequently displayed, depending upon the format of the DSPLY statement.

User Response: If the cursor is on after the contents of the last field are displayed, the program is waiting for data to be entered by the user. This data will become the contents of the Result field used in the DSPLY operation (last field shown before cursor).

OTM20: DSPLY FIELD TOO LARGE

Explanation: Error message displayed if the length of a field being displayed in the DSPLY operation is greater than 80.

Program Action: Display message OTM10.

OTM21: DUPLICATE KEY

Explanation: Error message displayed if the program attempts to add a record to an indexed file and a record already exists with the same key.

Program Action: Display the program name of the file and then the message OTM10.

OTM22: END OF TAPE

Explanation: Error message displayed if the end of the tape is encountered while reading or writing a Cassette file.

Program Action: Display the program name of the file and then message OTM10.

OTM23: ENTER DATE AS MM/DD/YY

Explanation: Displayed if any of the special words: UDATE, UDAY, UMONTH, or UYEAR were used in the source program.

Program Action: Wait for input.

User Response: The desired date should be typed in. The format of the date is fixed, in that Sept. 5, 1973 should be entered as 09/05/73.

OTM24: ENTER EXTERNAL INDICATOR SETTING IN BINARY

Explanation: Displayed if any of the external indicators, U1 to U8, were used in the source program.

Program Action: Wait for input.

User Response: The values of the external indicators used in the program should be typed in. Detailed formatting information can be found in Appendix F.

OTM25: EOF AND NO INDICATOR

Explanation: Error message displayed if a READ operation encounters an end-of-file condition and no indicator was specified in columns 58-59.

Program Action: Display message OTM10.

OTM26: ERROR HALT n

Explanation: Message displayed at the end of a cycle if Halt indicator n is found on.

Program Action: Display message OTM10.

OTM27: FILE FORMAT ERROR

Explanation: Error message displayed if the format of a fixed format disk file does not match the program specifications of the file.

Program Action: Display the program name of the file and then message OTM10.

OTM28: FILE SPACE FULL

Explanation: Error message displayed if an attempt is made to allocate space to a disk file when either the disk is full or no more segment descriptor slots are available for the file.

Program Action: Display the program name of the file and then message OTM10.

OTM29: FORMAT CORRECT. CONTINUE?

Explanation: Message displayed if the format of the tape header labels was correct.

Program Action: Display label (OTM34) and wait for input.

User Response: If the correct tape is mounted, a "Y" should be typed in so processing can continue. If the incorrect tape is mounted, an "N" should be typed in, which will stop program execution.

OTM30: FORMAT ERROR, TRY AGAIN?

Explanation: Error message displayed if the format of a number being entered during a DSPLY operation is not correct (Incorrect Databus format).

Program Action: Wait for input.

User Response: A "Y" should be typed in if the user wishes to re-enter the data. A "N" should be typed if the user does not wish to try again, in which case OTM50 is displayed and entered.

OTM31: ILLEGAL FORMAT IN LABEL

Explanation: Error message displayed if the format in the HDR2 label of an input tape is not "F".

Program Action: Display message OTM10.

OTM32: INVALID BUFFER ADDRESS

Explanation: Error message displayed if the buffer address in a Record Address file is invalid.

Program Action: Display the program name of the file and then message OTM10.

OTM33: INVALID INDEX

Explanation: Error message displayed if an array index is less than 1 or greater than the number of elements in the array.

Program Action: Display message OTM55. If the response to this message is Resume, the first element of the array is accessed.

OTM34: LABELS:

Explanation: Message displayed prior to display of tape labels during header label tape processing.

Program Action: Display header labels.

OTM35: MATCH SEQUENCE ERROR

Explanation: Error message displayed if the sequence of data in any specified match fields is not as specified on the File Description Specs.

Program Action: Display message OTM10.

OTM36: MORE CARDS?

Explanation: Message displayed when the card reader hopper becomes empty.

Program Action: Wait for input.

User Response: If more cards are to be processed, they should be put into the hopper and a "Y" typed in. If there are no more cards to be processed, an "N" should be typed in, which will cause an end-of-file condition on the reader.

OTM37: MULTIPLE CHAINED OUTPUT IN SAME CYCLE

Explanation: Error message displayed if the program attempts to write two or more chained output records into the same file during a single cycle.

Program Action: Display the program name of the file and then the message OTM10.

OTM38: MULTIPLE LOADER OUTPUT

Explanation: Error message displayed if more than one record is written to the LOADER device.

Program Action: Display message OTM10.

OTM39: MULTIPLE UPDATE IN SAME CYCLE

Explanation: Error message displayed if the program attempts to write two or more records onto an Update file during a single cycle.

Program Action: Display message OTM10.

OTM40: NO DATA FOR TABLE OR ARRAY

Explanation: Error message displayed if there is no data for a pre-execution time table or array.

Program Action: Display message OTM55. If the response to this message is Resume, the table will remain empty and the object program will resume execution.

OTM41: NO HDR1 LABEL - NO PROBLEM

Explanation: Message displayed if the HDR1 label is not present on an output tape.

Program Action: The program will supply a dummy HDR1 label and processing will continue.

OTM42: NO LOADER OVERLAY

Explanation: Error message displayed if the source program has specified the LOADER device and the loader object file (RPGLDR/OV1) does not exist.

Program Action: Execution is terminated.

User Response: The file RPGLDR/OV1 should be re-installed from the RPG II generation tapes and the progam re-run.

OTM43: (NO SUCH FILE)

Explanation: Error message displayed if, when naming an input file during the file opening sequence, the named file does not exist.

Program Action: A file name will be asked for again.

User Response: The name of an existing file should be typed in.

OTM44: NON-ZERO BLOCK COUNT

Explanation: Error message displayed if the block count in the HDR1 label of an input tape is not zero.

Program Action: Display message OTM10.

OTM45: (NOT READY)

Explanation: Message displayed when the program attempts to open either the tape unit or card reader and the device is not in a ready condition.

Program Action: Wait for device to become ready.

User Response: The device should be made ready.

OTM46: NUMERIC FIELD ERROR

Explanation: Error message displayed if a character in a numeric field is not a digit.

Program Action: Display message OTM10.

OTM47: OPEN XXXXXXXX AS YYYYYY FILE:

Explanation: General message displayed during the file opening sequence for every file described in the program. The program name for the file will appear in place of the X's and the type of file will appear in place of the Y's.

Progam Action: Depending upon the type of the device specified for the file or, if a DISK file, whether it has an assignable or defined name, the program will either supply a name after this message or wait for input.

User Response: If the file is an assignable DISK file the appropriate file name should be typed in. An extension of "TXT" and "all drives" will be assumed if neither the extension nor drive number is supplied.

OTM48: OPT TEST LOOP

Explanation: Error message displayed if all record types for an input file are described as optional and the current input record is not identifyable.

Program Action: Display message OTM10.

OTM49: PARITY ERROR IN INDEX. RE-INDEX

Explanation: Error message displayed if a parity is found in the index of an indexed (ISAM) file.

Program Action: Display the program name of the file and then the message OTM10.

User Response: When job terminates, use the INDEX utility to re-index the file.

OTM50: RE-TRY DECLINED

Explanation: Message displayed if the response to message OTM30 was 'N'.

Program Action: Display message OTM10.

OTM51: READ PARITY

Explanation: Error message displayed if a parity fault occurred while reading a disk file.

Program Action: Display the program name of the file and then message OTM10.

OTM52: READER CHECK!

Explanation: Error message displayed if a card reader error is detected.

Program Action: Display message OTM56.

OTM53: RECORD FORMAT ERROR

Explanation: Error message displayed if the physical file number or logical record number in a disk record do not match the entries in the logical file table.

Program Action: Display the program name of the file and then message OTM10.

OTM54: RESULT OVERFLOW

Explanation: Error message displayed if the result of an arithmetic operation is too large for the result field.

Program Action: Display message OTM10.

OTM55: RESUME/BYPASS/CANCEL/ABORT

Explanation: General error message displayed after many other specific error messages to give the user the option as to what action should be taken by the program.

Program Action: Wait for input.

User Response: Four different responses are allowed. Typing an 'R' will cause the program to Resume execution at the point where the error occurred. Typing a 'B' will cause the program to Bypass the current cycle and read the next record. Typing a 'C' will Cancel program execution and close all the files. Typing an 'A' will immediately return control to the operating system without closing any files.

OTM56: RESUME/CANCEL/ABORT

Explanation: Displayed after message OTM52. Check card reader indicator lights. If STACKER and DATA lights are both off, an invalid punch combination has been detected in the last card in the stacker. If the STACKER light is on, the stacker is full. If the DATA light is on, the last card in the stacker was read incorrectly.

Program Action: Wait for input.

User Response: Remove the last card stacked, correct it if necessary and insert it in front of the cards in the hopper. Type 'R' to resume. Type 'C' to cancel program execution or 'A' to abort if the error cannot be corrected.

OTM57: RID TESTS FAILED

Explanation: Error message displayed if a record in an input file can not be identified; i.e. does not match any of the Record Identifying Codes on the Input Specifications.

Program Action: Display message OTM10.

OTM58: SQRT OF NEGATIVE NUMBER

Explanation: Error message displayed if the SQRT operation was attempted on a number less than zero.

Program Action: Display message OTM10.

OTM59: TAPE BLOCK COUNT BAD

Explanation: Error message displayed if the block count in the EOF1 label of an input tape is not the same as the number of blocks processed by the program.

Program Action: Display message OTM10.

OTM60: TAPE PROCESSING ABORTED

Explanation: Message displayed if the response to OTM29 was a "N".

Program Action: Display message OTM11.

OTM61: TOO MUCH DATA FOR TABLE OR ARRAY

Explanation: Error message displayed if a pre-execution time table or array has been entirely filled and there is another record of data for that table or array.

Program Action: Display message OTM56. If the response to this message is Resume, the offending record will be ignored and the table loading process will continue.

OTM62: UNFINDABLE FILE

Explanation: Error message displayed if there is no file zero on a Cassette tape.

Program Action: Display the program name of the file and then message OTM10.

OTM63: WRITE PARITY

Explanation: Error message displayed if a parity fault occurred while writing a disk file.

Program Action: Display the program name of the file and then message OTM10.

OTM64: WRITE PROTECT

Explanation: Error message displayed if an attempt is made to write on a disk file which is write protected.

Program Action: Display the program name of the file and then message OTM10.

OTM65: WRONG BLOCK LENGTH

Explanation: Error message displayed if the block length in the HDR2 label of an input tape is not the same as that specified in the source program.

Program Action: Display message OTM10.

OTM66: WRONG RECORD LENGTH

Explanation: Error message displayed if the record length in the HDR2 label of an input tape is not the same as that specified in the source program.

Program Action: Display message OTM10.

OTM67: 3 FILES

Explanation: Error message displayed if only three disk files were specified in the program, but more than three are being used during program execution.

Program Action: Display message OTM10.

APPENDIX J. RPGII USER ASSEMBLY LANGUAGE FACILITIES

J.1 The RPG II Library Facility

An integral component of the RPG II compiler system is the LIBRARY facility. This facility includes the three system library files, RPGALIB/RPG, RPGBLIB/RPG and RPGCLIB/RPG, the library pre-processor, RPGPP, and, optionally, one user library file. The libraries contain fixed code sequences which can be selectively included in an RPG II object program, depending upon the particular operations specified in the source program. The pre-processor transforms a library file from the standard text (assembler) format into a format compatible with the RPG II compiler. The USER LIBRARY facility allows for user written routines to be assembled into an RPG II object program: SPECIAL device drivers, user label processors, and routines referenced by the EXIT operation. The calling sequences generated for these features will be described at the end of this appendix.

A library file is partitioned into SEGMENTS, each of which can be included separately into an RPG II object program. A segment inclusion is done on the basis of ENTRY POINTS in the segment and undefined symbols in the object code. In other words, during the library inclusion phase of the compiler, a segment will be included if at least one of its entry points corresponds to an undefined symbol in the main dictionary. When such a segment is found, it becomes part of the object code and treated exactly as if it were code directly generated by the compiler. Any undefined symbols it may have, if not already defined previously, will then cause additional library segments to be included. In this way, a hierarchy of segments can be included in the object code, depending upon the particular operation specified in the source program.

All symbols in an RPG II object program are QUALIFIED, in that they have the form:

<A'CHAR>:<SYMBOL> or <SYMBOL>

where A'CHAR is any alphabetic character. This effectively increases the length of a symbol to 7 characters and reduces the possibility of conflicts. The compiler uses this facility to partition the object symbols into a number of categories,

depending on the particular function the symbol is involved with.

J.2 The RPG II Pre-processor

As discussed previously, the pre-processor, RPGPP, translates a text file to library format. Its input is substantially like that of ASSEMBLER 5, with some additions and deletions consistent with generating a library file. The start of an independent segment is denoted with the IDENT directive, which must be the first statement in the input file. The end of a segment is the next IDENT directive, or the end of the file. In other words, IDENT directives partition the file into independent segments. A segment ENTRY POINT is declared by terminating a label with an asterisk. There may be any number of entry points, but there should be at least one. If not, there is no way for the segment to be included in the object program.

The following table lists the additional directives accepted by the pre-processor.

Pre-processor Directives

IDENT Define the start of a segment, which continues until the next IDENT directive or the end of the file. It must be the first statement in the input file. The label and expression fields are ignored. The system libraries use the expression field to denote the hierarchical level of the segment, where 0 means a primitive level and higher numbers mean more general levels. This denotation, however, is for documentation purposes only.

QUAL QUALIFY. Defines the current qualification character. The expression field can contain any alphabetic character, or can be blank. The current qualification character is that character which is affixed to every symbol (during pre-processing) which is not explicitly qualified. For example, if X were the current qualification character, the statement:

```
LABEL      MSA   *COUNT
```

would be transformed into:

```
X:LABEL    MSA   *X:COUNT
```

and this latter form would appear in the object code. However, this automatic qualification can be inhibited on a per-symbol basis by explicitly qualifying a symbol with "<CHAR>:". For example:

```
LABEL      MSA   *Y:COUNT
```

would be transformed into:

```
X:LABEL    MSA   *Y:COUNT
```

if X were the current qualification. The initial current qualification character in the pre-processor is X. It is recommended that any user library routines be restricted to X, B, or Q qualification, so as not to conflict with RPG II system symbols.

PPLIST PRE-PROCESSOR LIST control. This directive can accept the L or I flags (as in the LIST directive) and controls the listing during pre-processing.

EBCDIC Sets the mode to EBCDIC, whereby all string characters are transformed into their EBCDIC equivalent values.

ASCII Sets the mode to ASCII, whereby all string characters are transformed into their ASCII equivalent values.

The following ASSEMBLER 5 directives are illegal as input to the pre-processor:

SET, LOC, ORG, USE, END.

The function of the pre-processor is to translate text files to library files. The syntax of the source code and the opcode specifications are checked for validity, and some directives are evaluated. In particular, an INCLUDE directive is evaluated during pre-processing and the contents of the included file put in the library file. Also, the IF and LIST directives have no effect during pre-processing. The '*' terminator for a label, as mentioned previously, declares that label as an ENTRY POINT of the segment in which it is located. This is a pre-processor evaluation; the '*' is ignored by the compiler. All macros and any directive or construct not mentioned in this section is exactly the same as in ASM 5, and is evaluated by the compiler.

J.3 RPG II Calling Sequences to User Subroutines

The RPG II system will generate calls to user subroutines when any of the following language features is invoked:

1. SPECIAL files,
2. non-standard tape labels, or
3. EXIT operations.

In each case the name of the subroutine must be given in special columns as follows:

1. cc54-59 - Label Exit - in the File Description Specifications, and
2. cc33-38 - Factor 2 - in the Calculation Specifications.

The compiler always uses 'X'-qualified symbols to refer to a user subroutine. We will now discuss the calling sequence used for each feature.

J.3.1 SPECIAL Device Drivers

All files in the RPG object program are described by a table entry called the File Description Block (FDB). The format of this table is given in Appendix G, and in the second object program segment. A SPECIAL device subroutine will be called: to open the file, to read from it, to write on it, or to close it. In all cases the subroutine will be called with an operation code in A and the address of the FDB in HL. The user must clear the file-closed flag (MCLSFDB) upon OPEN, set the end-of-file flag (MEOFFD) when the end of file has been read, and reset the file-closed flag when called with the CLOSE function. The operation codes are as follows:

Value of A	Operation
0	Open
1	Input
2	Output
3	Close

Thus a simple input driver which requires no open or close actions might start like this (remember that 'X'-qualification is the default):

```
         IDENT                  SPECIAL DRIVER SUBR
         QUAL   X
SPCL     ORA                    OPEN OPERATION?
         JTZ    SETOPN            YES
         CP     3               CLOSE?
         JTZ    SETCLS            YES
         LDH                    NO< ASSUME INPUT OPERATION
         LEL                    COPY FDB ADDR TO (DE)
         ...
*
SETOPN   LAM                    GET FLAG BYTE
         ND     -1.XOR.MCLSFD     & CLEAR CLOSED
         LMA                       & UPDATE FLAG
         RET                    EXIT
*
SETCLS   LAM                    GET FLAG BYTE
         OR     MCLSFD            & SET CLOSED
         LMA                       & UPDATE FLAG
         RET                    EXIT
```

J.3.2 Non-standard Tape Labels

A non-standard label routine is called with the operation in A and the FDB address in HL. The standard tape drivers may be used for input-output; use the LOI options with some tape program to get a listing. The operation code is 0 for open (i.e., header labels) and 1 for close (i.e., trailer labels). The tape will be positioned before the first label record in each case and it is the responsibility of the label routine to properly position the tape at the beginning of data (for headers) or before the last tape mark (for trailers).

APPENDIX K. DETAILED RPG OBJECT FLOW (COMMON)

The flowcharts in this appendix give the object program flow. The numbers beside each box refer to the description which follows. Initialization is covered in steps 1 through 4; the program cycle, 5 through 38; and termination, 39 and 40.

K.1 Initialization

1. Start: For each external indicator (U1-U8) used, the program requests the setting from the keyboard.

2. For each table input file in use, the program opens the file, and then reads all the tables on that file into memory.

3. All files used during normal processing are opened/prepared. The names of files assigned to the disk may be keyed in. If the program is unable to open all files, DOS is reloaded.

4. All working areas are cleared to blank/zero. Indicators are set to off except 1P. Records are requested for all input/update files.

K.2 Program Cycle

5. Cycle: All heading and detail records whose output condition is true are written. The overflow indicator and switch are turned on if necessary and overflow output is performed if fetched during detail output.

6. The overflow switch is copied to the overflow indicator (see also step 35).

7. Bypass: If any halt indicator (H1-H9) is on, a halt code is set, and the corresponding indicator is turned off. Control then goes to step 18 (Error). If all halt indicators are off this step is skipped.

8. All record identifying indicators are turned off, as well as control level indicators.

9. If the last record indicator (LR) is on, a branch is taken to step 30 (Last Cycle).

Input:

The input steps 10 through 17 are performed for all input/update files. The exit to Select is taken only for the last file.

10. If no record request is pending for this file, go to step 10 for the next file or go to step 21 (Select) if this is the last input file. Note that on the first cycle, requests are pending for all input files.

11. Go to step 17 if this file is at end-of-file.

12. Read the next record from this file.

13. Go to step 18 (Error) if an Input/Output error occurred.

14. Go to step 17 if file now at end.

15. Try to identify the record type and check the type sequence.

16. Go to step 18 (Error) if type or type sequence error.

17. Turn off the record request. Control goes to the next input sequence or to step 21 (Select).

18. Error: The code for the halt is displayed and the program requests whether to bypass, cancel, or abort.

19. If the halt is to be bypassed, go to step 7 (Bypass).

20. If the halt is to cause cancellation, go to step 30 (Last Cycle). Otherwise abort to DOS.

Select:

Depending on the program, one of four select routines is compiled. The others are subsets of matching with multiple input files.

21. If all input files which must be at end-of-file are at their end, go to step 30 (Last Cycle).

22. If a valid FORCE has been executed, make that file current; reset FORCEing.

23. If the current record of the currently selected file has no matching fields go to step 26.

24. Select the record and file with matching fields with the highest priority (lowest in sequence if ascending).

25. Request next record and go to step 18 (Error) if selected record is not in sequence.

26. Turn on the record identifying indicator of the selected record.

27. Test Break: If the current record contains control fields, a test for control break occurs. If no control break, go to step 33 (Test End).

28. Turn on indicators for all appropriate control levels. Reset switch to bypass totals (takes effect next cycle).

29. Go to step 33 (Test End) until after the first control break has been found.

30. Last Cycle: Turn on LR indicator and all control level indicators (L1-L9).

31. Perform all total calculations whose enabling conditions are met. Since I/O can be performed, set indicators as needed.

32. Perform all enabled total output, process overflow as in step 5.

33. Test End: Go to step 39 (End-of-Job) if LR is on.

34. Go to step 36 if the overflow indicator is off.

35. Turn off the overflow switch. Perform overflow output only if overflow not previously fetched.

36. Turn on the MR indicator if the primary file matches some secondary file.

37. Move current input record to fields, setting resulting indicators. Request new read for current file.

38. Perform detail calculations as in step 31. Go to step 5

(Cycle).

K.3 Termination

39. EOJ (End-of-Job): Close all files except table output files.

40. For each table output file, write out all tables and arrays requested for the file, and then close the file. Exit to DOS after all table output.

APPENDIX L. COMMON REFERENCE TABLES

Standard Tape Label Format

Name	Description
VOL1	Volume Label
HDR1	Header Label 1
HDR2	Header Label 2
	File Mark
	Data
	File Mark
EOF1	Trailer Label 1
EOF2	Trailer Label 2
	File Mark
	File Mark

Format of Unlabeled RPG Tape

```
        .
        .
Data Blocks
        .
        .
        .
File Mark
File Mark
```

Reading a tape with a file mark on the front will cause an EOF condition at the beginning of the file. A user label routine could read the file mark and ignore it.

Volume Label Format

Columns	Contents	Description
1-4	VOL1	Label Identifier
5-80	Arbitrary	Volume Identification

Format of Label 1

1-3	HDR	Header Label Identifier
	EOF	Trailer Label Identifier
4	1	Label Number
5-54	Arbitrary	Data Identification
55-60	Number	Block Count (Trailer)
	000000	Zero (Header)
61-80	Arbitrary	Data Identification

Format of Label 2

1-3	HDR	Header Label Identifier
	EOF	Trailer Label Identifier
4	2	Label Number
5	F	Fixed Format
6-10	Number	Block Length
11-15	Number	Record Length
16	2	800 BPI Density

	3	1600 BPI Density
17	0	No Volume Switch
18-34	Arbitrary	Job Step Identification
35-80	Blank	

ASCII to EBCDIC Translation Table

MSH	40	60	100	120	140	160	200	220	240	260	300	320	340	360
LSH														
000	SP	0	@	P	`	p	40	60	101	121	104	124	144	164
001	!	1	A	Q	a	q	41	61	102	122	105	125	145	165
002	"	2	B	R	b	r	42	62	103	123	106	126	146	166
003	#	3	C	S	c	s	43	63	110	130	107	127	147	167
004	$	4	D	T	d	t	44	64	111	131	212	232	252	272
005	%	5	E	U	e	u	45	65	142	160	213	233	253	273
006	&	6	F	V	f	v	46	66	143	161	214	234	254	274
007	'	7	G	W	g	w	47	67	150	162	215	235	255	275
010	(	8	H	X	h	x	50	70	151	163	216	236	256	276
011	)	9	I	Y	i	y	51	71	200	170	217	237	257	277
012	*	:	J	Z	j	z	52	72	220	171	312	332	352	372
013	+	;	K	[	k	{	53	73	240	241	313	333	353	373
014	,	<	L	\	l	|	54	74	264	270	314	334	354	374
015	-	=	M	]	m	}	55	75	265	271	315	335	355	375
016	.	>	N	^	n	~	56	76	266	340	316	336	356	376
017	/	?	O	_	o	DEL	57	77	267	341	317	337	357	377

EBCDIC to ASCII Translation Table

MSH	40	60	100	120	140	160	200	220	240	260	300	320	340	360
LSH														
000	200	220	SP	&	-	265	251	252	253	[	{	}	276	0
001	201	221	240	260	/	266	a	j	273	^	A	J	277	1
002	202	222	241	261	245	267	b	k	s	]	B	K	S	2
003	203	223	242	262	246	270	c	l	t	\	C	L	T	3
004	204	224	300	320	340	360	d	m	u	254	D	M	U	4
005	205	225	301	321	341	361	e	n	v	255	E	N	V	5
006	206	226	302	322	342	362	f	o	w	256	F	O	W	6
007	207	227	303	323	343	363	g	p	x	257	G	P	X	7
010	210	230	243	263	247	271	h	q	y	274	H	Q	Y	8
011	211	231	244	264	250	272	i	r	z	275	I	R	Z	9
012	212	232	`	!	DEL	:	304	324	344	364	312	332	352	372
013	213	233	.	$	,	#	305	325	345	365	313	333	353	373
014	214	234	<	*	%	@	306	326	346	366	314	334	354	374
015	215	235	(	)	_	'	307	327	347	367	315	335	355	375
016	216	236	+	;	>	=	310	330	350	370	316	336	356	376
017	217	237	|	^	?	"	311	331	351	371	317	337	357	377

NOTE: MSH= Most Significant Half
LSH= Least Significant Half

For example: MSH (240) + LSH (013) = 0253

Values between 000 and 037 map into themselves.

APPENDIX M. COMMON INPUT/OUTPUT DEVICE INTERFACES

This section contains information about each of the I/O devices supported by the RPG II system, including what, if any, actions must be performed by the user for any particular device.

PRINTER

The printer must be correctly positioned at top of form prior to executing programs that use the printer as an output device. The printer will remain at this initial position when it is opened at the beginning of object program execution. When the printer is closed at the completion of execution, it will skip to top-of-form. Depending upon the form length used in the RPG program, a "soft" top-of-form will be issued in place of a "hard" top-of-form.

CONSOLE

When the keyboard is used as an input device and the object program requires input, a "*" will be displayed, followed by the blinking cursor. The input record should then be entered, with column 1 of the record being the character position just after the "*". End-of-file for this device is signaled by simultaneously depressing the DISPLAY key while entering a null record with the ENTER key.

DISK

Disk files may be organized with either variable or fixed length records.

Each file sector contains 3 system bytes followed by up to 250 data and control bytes and a byte containing 003 to indicate the logical end of the sector. Logical records are terminated with an 015 byte and are packed contiguously into sectors, spanning to the next sector when the 250th data byte is filled.

Fixed format files contain records of equal length, with no compression of spaces. This organization is necessary to allow the files to be randomly processed and updated.

Variable format is the standard GEDIT format with compressed blanks and variable length records. Contiguous blanks are represented by an 011 byte followed by a byte containing the count of the number of spaces compresed (2-255 in binary).

RPG disk record formats are compatible with records written by other DOS programs, however, caution should be excercised in selecting a record length that does not exceed the maximum record size accepted by other DOS programs that may be handling the same data. When creating fixed format records with Databus or Datashare, they must be written using the physically sequential access method if they are to be processed by RPG as output, update or direct files. The one exception to this rule is 249 character records written using the physically random access method.

CARD READER

If the card reader is specified as an input device in an RPG II program, the object program will try to start it up during the open sequence. If the power to the reader is off or the hopper is empty at this time, the program will BEEP and display "NOT READY" until power is on and cards are in the hopper.

After one or more cards have been read by the program, and the hopper becomes empty, the message "MORE CARDS (Y OR N)" will be displayed. Place the last card in front of any additional cards and ready the card reader. If there is a hardware error in the card reader, the message "CLEAR READER ERROR" will be displayed. When the reader is made ready, the message is removed from the screen and cards continue to be read. To indicate an end-of-file condition on the card reader, an end-of-file card must be placed at the end of the deck. This card contains a multi-punch in card column one of 1-2-3-4-5-6-7-8-9.

TAPE

RPG II supports 9-track industry-compatible magnetic tape as either an input or output device. During the open sequence, the RPG II object program will interrogate the tape unit to determine if it is ready for operation. If the deck is not in service, "NOT READY" will be displayed and a BEEP will sound until the deck is in service. (To cause the deck to be in service, the tape must be loaded, the disable switch must be in RUN, and the REMOTE button must be pressed). In addition, if the TAPE is an output device, the presence of a write ring is checked for, and an error message is displayed if one is not on the reel.

The user has the option, in his source program, to specify one of three tape label options: unlabeled tape, user labels, or IBM standard labels. The procedures for invoking one of these options and for processing user labels are discussed in other sections of this manual. This portion discusses the IBM standard label option. The tape will always be rewound when the file is opened. However, the tape is not rewound when the file is closed.

TAPE INPUT

Upon opening the tape, the volume label (VOL1) and the two header labels (HDR1, HDR2) are read and displayed. The first four characters of each one are checked, the block count in HDR1 is checked for zero, and HDR2 is checked for "F" format and the correct block length and record length. If all checks are successful, the user is asked if the program should continue.

Upon encountering end-of-file, the first trailer label (EOF1) is read. The first four characters are checked and the block count is compared against the internal block count. The second trailer label is not checked.

TAPE OUTPUT

Upon opening the tape, the volume label (VOL1) is read and displayed. If it is not present, or if the first four characters are not "VOL1", tape processing is aborted. Next a read of the first header label (HDR1) is attempted. If it does not exist, a dummy HDR1 label is generated and displayed. The user is then asked if the correct tape is mounted. If "Y" is the response, "HDR1" is displayed and the user is asked to key in data for columns 5 to 80. After the data is entered, a block count of zero is put in columns 55 to 60, but all other columns can have arbitrary information in them. Next a new HDR2 label is generated with the current format information and then displayed. The user may now enter any data in the job-step identification field (columns 18 to 34). The tape is now rewound, the VOL1 label re-written, the new header labels are written, and a file mark written. The tape is now positioned for normal output operations.

Upon closing the tape, the current block is written on the tape. If the block is not full, dummy blank records are generated so that all data blocks will be of the same length. A file mark is then written, the EOF1 and EOF2 labels are generated and written, followed by two file marks.

CASSETTES

Both cassette decks may be used as RPG II I/O devices. The rear deck is designated CASSET1 and the front deck CASSET2. Either may be used as input or output, independently of each other. The data on a cassette is assumed to be a standard source file (file 0) in GEDIT format, with compressed blanks and variable length logical records.

APPENDIX N. CODING SHEET SUMMARY (COMMON)

N.1 Common Fields

Columns 1- 2: (Page)

Columns 3- 5: (Line)

Page/line number must be in strictly ascending order, if used.

Column 6: (Form Type)

H	Header (control) card.
F	File description specifications.
E	Extension specifications.
L	Line counter specifications.
I	Input record formats.
C	Calculations.
O	Output record formats.

Column 7: (Comments)

*	Comment line identifier.

Columns 75-80: (Program Identification)

xxxxxx	Alphanumeric consisting of any 6 characters.

N.2 Header Specification

Column 6: (Form Type)

H	To identify this source line type.

Column 10: (Object Output)

C	
D	
blank	

Column 11: (Listing Options)

B	Suppress listing.
blank	List source code.

Columns 12-14: (Core Size to Execute)
Size of memory required for object program.

Columns 13-14: Number of 1K blocks: (1K = 1024).

0 or blank	If same as that used to compile the program.
01-13	For 01K-13K bytes, if the object program is to be executed on a system with less memory than the compilation system.

Column 12: (Number of additional quarter blocks)

0 or blank	If none.
Q	If one additional Quarter block (256 bytes) is needed.
H	If two quarter blocks are needed (i.e., a Half block (512 bytes)).
T	(768 bytes).

Column 15: (Debug)

1	To perform DEBUG operation (calculations).
blank	To ignore DEBUG commands.

Column 26: (Alternate Collating Sequence)

A	ASCII sequence.
blank	EBCDIC sequence.

N.3 File Description Specifications

Column 6: (Form Type)

F	File Specification statement type identification.

Columns 7-14: (File Name)

file name	Name to be used for this file throughout the program.

Column 15: (File Type)

I	Input.
O	Output.
U	Update.
D	Display.

Column 16: (File Designation)

blank	This column must be blank for display and non-chained output files.
P	Primary -- there must be exactly one primary input/update file.
S	Secondary.
C	Chained.
R	Record address file.
T	Table or array file (see Extension specs).
D	Demand file.

Column 17: (End of File)

E or blank for all files	If all records of the file(s) must be processed before the program can be terminated.
blank for some files	If all records of the file need not be read.
blank for all files	If all files must be read to end-of-file.

Column 18: (Sequence)

A	Records are in ascending order.
D	Records are in descending order.
blank	No sequence checking.

Column 19: (File Format)

F	Fixed length records: record length specified will be that of all records.
V	Variable length records: maximum length will be given.

Columns 20-23: (Block Length)

nnnn	Length of block.
blank	Default length.

maxima are:	For:
78	Console input.
80	Console output.
132	Printer.
80	Card reader.
9999	Disk.
249	Cassette.
1057	800 BPI Tape.
2048	1600 BPI Tape.
80	Loader.
9999	Special.

Columns 24-27: (Record Length)

nnnn	Record length (number of bytes per record).

Column 28: (Mode of Processing)

L	Sequential within limits. (Must be indexed file).
R	Random. (Chained disk files and files processed by ADDROUT).
blank	Sequential. (Non-disk files must be processed sequentially).

Columns 29-30: (Length of Key)

nn	Length of indexed file keys or ADDROUT file record.

Column 31: (Record Address Type)

A	Unpacked indexed file keys.
I	File is an ADDROUT file or is processed by one.
blank	All other files.

Column 32: (File Organization)

I	Indexed file.
T	ADDROUT file.
blank	Any other type of file.
1-9	Additional I/O areas (ignored).

Columns 33-34: (Overflow Indicator)

OV or OA-OG	The indicator to be used.
blank	None used.

Columns 35-38: (Key Field Starting Location)

nnnn	Key field starting location for indexed files.

Column 39: (Extension Code)

E	On the Extension form.
L	On the Line counter form.
blank	Neither needed.

Columns 40-46: (Device)

PRINTER	Printer (default is LOCAL).
CONSOLE	Keyboard display.
DISK	Disk.
READER	Card reader.
TAPE	Industry-compatible tape unit.
CASSET1	Rear tape cassette.
CASSET2	Front tape cassette.
LOADER	Pseudo device for use with the DOS CHAIN command.
SPECIAL	Special input/output device supported only through user assembly language coding.

Column 53: (Continuation Code)

A	Assign disk file name at run-time.
D	Internal name is to be used externally as well.
S	Standard tape labels are used.
N	Non-standard tape labels are used.
U	Tapes are unlabeled.
K	Continuation line, columns 54-59 must be filled.
blank	Defaults to same as "A". (Assign disk file name at run-time).

Columns 54-59: (Continuation Option)

EXTDRV	Extension and/or drive for disk file.
MAXSEC	Maximum number of sectors for new disk files.
LOCAL	The local printer is used at object-time.
SERVO	The servo printer is to be used at object-time.
ASCII	Tape file is written in ASCII.
800 or 1600	Tape Density (800 is assumed if not given).
RPG name	Name of user written subroutine which will perform I/O for a Special device or process non-standard labels.
blank	Neither a special device, nor non-standard labels are being used.

Columns 60-62: (Extension)

extension	File name extension to be used for a disk file.

Columns 63-65: (Drive)

drive	Drive to be selected for disk file (DR0-DR1 or D00-D15).

Columns 60-65: (Number of Sectors)

nnnnnn	With MAXSEC, to specify LRN limit.

Column 66: (File Addition)

A	If new records are to be added to the file (APPEND file).

Columns 71-72: (File Conditioning Indicator)

U1-U8	If the file is to be used only when the specified external indicator is on.

N.4 Extension Specifications

Column 6: (Form Type)

E	Form type.

Columns 11-18: (From Filename)

table or array file name	If the table or array is to be loaded at pre-execution time (From file name).
blank	Table or array loaded at compile time if number of entries per record is specified, otherwise at execution time.
record address file name	If record address file is defined.

Columns 19-26: (To Filename)

name of output file	If a table or an array is to be written out at the end of a program (To file name).
name of an input or update file	If file processed via a record address file specified in columns 11-18.

Columns 27-32: (Table or Array Name)

TABxxx	Table name, xxx being from 1-3 alphanumeric characters.
array name	An array name must not begin with "TAB".

Columns 33-35: (Number of Entries per Record)

1-999	Exact number of table or array entries per input record, if loaded at compile or pre-execution time. Use the sum of the contents of columns 40-42 and 52-54 (entry lengths) in computing this number.

Columns 36-39: (Number of Entries/Table)

1-9999	Maximum number of table or array entries. This number is limited by the size of object time memory as the whole table is kept in main memory.

Columns 40-42: (Length of Entry)

1-256	Length of a table or array entry.

Column 43: (Packed or Binary Field)

-	If an execution time array is in Databus-compatible format.

Column 44: (Decimal Positions)

0-9	Number of decimal positions for numeric field.
blank	Alphanumeric.

Column 45: (Sequence)

A	Ascending order.
D	Descending order.
blank	No particular order.

Columns 46-57: (Alternate Table/Array Specification)
Entries to this table or those of the array are paired with those of the table described in columns 27-45.

Columns 46-51: (Alternate Table or Array Name)
(see Columns 27-32)

Columns 52-54: (Alternate Length of Entry)
(see Columns 40-42)

Column 55: (Alternate Packed or Binary Field)
(see Column 43)

Column 56: (Alternate Decimal Position)
(see Column 44)

Column 57: (Alternate Sequence)
(see Column 45)

Columns 58-74: (Comments)

text	Any programmer comments

N.5 Line Counter Specifications

Column 6: (Form Type)

L	Form type.

Columns 7-14: (Filename)

file name	Name of a PRINTER file.

Columns 15-17: (Lines per Page)

1-99	Number of printing lines available per page.

Columns 18-19: (Form Length)

FL	Previous entry is Form Length.

Columns 20-22: (Line Number of Overflow Line)

1-99	Number of the line at which the overflow indicator is to be set on. (Allow additional lines to complete groups of detail records and for totals).

Columns 23-24: (Overflow Line)

OL	Previous entry is Overflow Line.

N.6 Input Record Descriptions

N.6.1 Record Type Definition

Column 6: (Form Type)

I	Form type.

Columns 7-14: (Filename)

file name	Name of input file containing the record described.

Columns 15-18: (Record Order)

Columns 15-16: (Sequence)

aa	Any two alpha characters if no check for sequence is to be made.
01-99	If records must be in order by type: first type must be 01, subsequent types must be defined in ascending order.

Column 17: (Number)

1	Only one.
N	More than one.

Column 18: (Option)

O	Record type is optional.
blank	Record type is required.

Columns 19-20: (Record Identifying Indicator)

01-99, L0-L9, LR or H1-H9	Only one RID is on at a time.

Columns 21-41: (Record Identification Codes)

This set of columns can contain up to three codes whose presence is to be ANDed together per line.

Columns 21-24, 28-31, 35-38: (Position)

1-256	Position (column) in record of code.
blank	No (additional) code needed.

Columns 25, 32, and 39: (Not)

blank	Code is present in this type of record.
N	Code is Not present.

Columns 26, 33, and 40: (C/Z/D)

C	Entire Character is code.
Z	Only Zone portion is used.
D	Digit part only.

Columns 27, 34, and 41: (Character)

Any character at all to indentify this record type.

N.6.2 And/Or Line

Column 6: (Form Type)

I	Form type.

Columns 14-16: (Logical relation)

AND	To AND the result of tests specified on this line.
OR	To (inclusive) OR the result.

Columns 21-41: (Record Identification Codes)

Additional codes as specified above for these columns.

N.6.3 Field Definitions

Columns 6: (Form Type)

I	Form type.

Column 43: (Datapoint-compatible Format)

-	If data in Datapoint-compatible decimal format.

Columns 44-51: (Field Location)

Field location, up to 256 positions if alphanumeric (final - initial + 1 <= 256).

Columns 44-47: (From)

1-9999	Initial position (column).

Columns 48-51: (To)

1-9999 (File-dependent).	Final position.

Column 52: (Decimal positions)

blank	Alphanumeric field.
0-9	Number of decimal places (numeric field).

Columns 53-58: (Field Name)

field name,
array name or
array element

PAGE	To input initial page number.

Columns 59-60: (Control Level)

L1-L9	A control level indicator.
blank	If none needed.

Columns 61-62: (Matching Fields) -- If fields match Matching Record indicator is set on.

M1-M9	Matching field number.
blank	If none used.

Columns 63-64: (Field record relation (FRR))

01-99	Field only used if indicator is on.
L1-L9	Data only used at previously specified control level.
MR	Matching Record permits acceptance of data from this field.
U1-U8	Field used only when this external indicator is on.
H1-H9	Relates field to a record type with a halt indicator in columns 19-20.
blank	If none needed.

Columns 65-70: (Field Indicators)

01-99	Field status indicator set.
H1-H9	Halt indicator set.
blank	None used.

Columns 65-66: (Plus)

(+)	Indicator is to be set on when numeric field is greater than zero.

Columns 67-68: (Minus)

(-)	Indicator set when contents of numeric field is less than zero.

Columns 69-70: (Zero)

(0)	Set if numeric field equal to zero.
(blank)	Set if numeric or alphanumeric field is blank.

N.7 Calculation Specifications

Column 6: (Form Type)

C	Form type.

Columns 7-8: (Control Level)

	Calculation selected:
L0, L1-L9	At control break (L0 is always on).
LR	When Last Record read.
SR	As part of a subroutine.
blank	Otherwise.
AN	This line of indicators is to be ANded with the previous one.
OR	This line is to be ORed with the previous line. Note: The line containing the operation code is the last in a series of ANds and ORs.

Columns 9-17: (Indicators) performance of the operation.

blank	If operation is to be done for every record (but see columns 7-8).

Columns 10-11, 13-14, and 16-17: Indicator

01-99, L1-L9, LR, MR, H1-H9, U1-U8, OA-OG, or OV	If performance of operation is dependent on the specified indicator.

Columns 9, 12, and 15: Negation relation

N	Operation performed when indicator is Not on.
blank	When indicator is on.

Columns 18-27: (Factor 1)
Columns 33-42: (Factor 2)

field name,
table name,
array name,
array element,
literal,
PAGE, UDATE,
UMONTH, UDAY,
or UYEAR

a label for one of the following operations (factor 1 only):

- TAG
- BEGSR
- ENDSR

a file name for one of the explicit I/O operations (factor 2 only):

- CHAIN
- DEBUG
- DSPLY
- READ
- FORCE

the name of the external subroutine (factor 2 only) called by the EXIT command.

Columns 28-32: (Operation)

Arithmetic operations:

Perform an operation using factor 1 and factor 2 to give the result.

ADD	ADD factor 1 to factor 2.
Z-ADD	ADD factor 2 to a field of Zeros.
SUB	SUBtract factor 2 from factor 1.

Z-SUB — SUBtract factor 2 from a field of zeros to yield the result (the negative of factor 2).

MULT — MULTiply factor 1 by factor 2. Note that the length of the result could be up to the sum of the lengths of the two factors.

DIV — DIVide factor 1 by factor 2 storing the quotient in the specified result field: factor 2 may not be zero.

MVR — Move the Remainder of the previous DIVide to the result field.

SQRT — Place the SQuare RooT of factor 2 in the result field: factor 2 may not be negative.

XFOOT — Crossfoot: the result is the sum of the elements of the array named as factor 2.

Comparison and test:

COMP — COMPare factor 1 to factor 2 and as a result set indicators showing whether factor 1 is greater than, less than, or equal to factor 2.

TESTZ — Test zone of the leftmost character of the result field setting indicators as follows:

+ if &, or A-I (12 zone)
- if -, } or J-R (11 zone)
0 otherwise.

Binary field operations:

Using bit positions specified in factor 2, operate on 1 or more bits of the result field. Factor 2 may be a string of decimal digits each specifying a position or it may be the name of a one-position mask having a bit on wherever a result field bit is to be operated on.

BITON	Set specified BIT(s) ON.
BITOF	Set BIT(s) OFf.
TESTB	TEST Bit(s) and set indicators to show if the bits specified were all zero, of mixed values, or were all ones, respectively.

Indicator set and reset:

SETON	SET indicators listed in columns 54-59 ON.
SETOF	SET listed indicators OFF.

Moves:

(Move the contents of the field named as factor 2 to the result field, ignoring decimal points).

MOVE	MOVE characters of factor 2 starting with the rightmost position.
MOVEL	MOVE characters, starting with the leftmost position.
MOVEA	MOVE characters, starting with leftmost position, ignoring array element boundaries.
MHHZO	Move High to High ZOne: move the leftmost zone only from factor 2 to the result field.
MHLZO	Move High to Low ZOne: from the leftmost position of factor 2 to the rightmost position of the result.
MLLZO	Move Low to Low ZOne: rightmost to rightmost.
MLHZO	Move Low to High ZOne: rightmost to leftmost.

Branching:

GOTO	Factor 2 names the label (factor 1) of the instruction with which to resume computation. (GO TO <label>).
TAG	TAG (label) a location in the calculations with the name given as factor 1.

Table lookup:

LOKUP	LOoK UP factor 1 in table or array named as factor 2. The result field contains the name of thee alternating table or array; the resulting indicators give the results of the search.

Subroutine operations:

BEGSR	BEGin a SubRoutine whose name is contained in factor 1.
ENDSR	END a SubRoutine and return to the command following the EXSR which caused this to be executed: factor 1 may contain a label (TAG).
EXSR	EXecute the SubRoutine named as factor 2.

Programmed control of Input/Output:

Note: normal program cycle is:
1. Read.
2. Calculate.
3. Write.

EXCPT	Write exception records (identified by an E in column 15 of the output format description).

DSPLY	Display on the console (file named as factor 2) the data specified as either factor 1 and or as the result field or both; if the result field is used, the machine will wait for input from the keyboard to enter into the named field or array: if only the enter key is pressed then no change will be made.
DEBUG	Output data for DEBUGging the program: write the data specified in the factor 1 and result fields into the file named as factor 2; also list those indicators which are on. Column 1 of the Header card must contain a 1 for this to be executed.
FORCE	FORCE the file specified as factor 2 to be read on the next program cycle.
READ	READ a record from a demand file during the current cycle. Note -- record identifying indicators are not automatically turned off until the end of the cycle. The resulting indicator designated in columns 58-59 will be turned on if the end of the file has been found.
CHAIN	Read or write a record of a CHAINed file (factor 2). Columns 54-55 should contain an indicator to be turned on if the record specified as factor 1 is not found.
SETLL	Set lower limit (factor 1) for processing sequential indexed file (factor 2).
BEEP	Emit an audible BEEP.
CLICK	Emit an audible CLICK.

External subroutine access:

The following two commands are used to generate a call to an external (assembly language) subroutine and to pass an argument address vector to it. (See Appendices E or I of the DATAPOINT RPG User's Guide for further information.)

EXIT	EXIT to (call) the pre-processed external subroutine named as factor 2.
RLABL	Pass the LABEL (data name) in the Result field to the subroutine EXITed to. The result field contains the name of a field, table or array or it contains INxx where xx is an indicator; the field length and number of decimal positions may be specified. The RLABL command must immediately follow the EXIT instruction.

Columns 43-48: (Result Field)

field name
table name
array name
array element

Columns 49-52: Result field definition (blank if previously defined)

Columns 49-51: (Field Length)

1-256	Field length.

Column 52: (Decimal Positions)

0-9	Number of decimal positions, if field is numeric.
blank	If field is alphanumeric.

Column 53: (Half Adjust)

H	To Half-adjust (round) result.
blank	Truncate result.

Columns 54-59: (Resulting Indicators)

01-99, H1-H9, L1-L9,LR, OA-OG or OV	Indicators to be set depending upon the result of the operation.

Columns 54-55: (Plus or High)

(+, >) Set if result positive, greater than, or higher.

Columns 56-57: (Minus or Low)

(-, <) Result negative, less than, or lower.

Columns 58-59: (Zero or Equal)

(0, =) Zero, equal, or successful match.

Columns 60-74: (Comments)

Comments

N.8 Output Format Specifications

N.8.1 Record Type Definition

Column 6: (Form Type)

O	Form type.

Columns 7-14: (File Name)

file name	Name of file being described.
blank	For subsequent record descriptions.

Column 15: (Type)

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

Columns 16-18: (Add a Record)

ADD	To add a record to a sequential or indexed disk file (optional).

Column 16: (Fetch Overflow)

F	To perform (Fetch) the printer page overflow routine if at the end of the page.
blank	Otherwise: overflow type printing done at normal point in the cycle.

Column 17: (Space)

space code	Space before printing.

Column 18: (Space)

space code	Space after printing.
0	Zero lines.
1 or blank	One line.
2	Two lines.
3	Three lines.

Columns 19-20: (Skip)

01-99	Skip to specified line of page before printing, next page if number less than or equal to current one, suppressing overflow printing.
blank	No skip before printing.

Columns 21-22: (Skip)

01-99	Skip after printing.
blank	No skip after printing.

(If both 19-20 and 21-22 blank, skip one after printing).

Columns 24-25, 27-28, and 30-31: (Output Indicators)

01-99, L1-L9, H1-H9, U1-U8, OA-OG, OV, MR,	
LR,	(On only at end of program during total output).
1P, or LO	(On only at the very beginning of the program).

Columns 23, 26, and 29: (Negation Relation)

N	If output is to occur when indicator is Not on.
blank	Otherwise.

N.8.2 And/Or Line

Column 6: (Form Type)

O	Form type.

Columns 14-16: (Logical Relation)

AND	AND indicator lines.
OR	OR lines.

Columns 23-31: Indicator specifications (see above).

N.8.3 Record Formats and Field Editing

Column 6: (Form Type)

O	Form type.

Columns 23-31: Indicator specifications (see above)

(blank If field always used)

Columns 32-37: (Field Name)

field name	Name of a field previously used in the program.
table name	
array name	
array element	
PAGE	
UDATE, UDAY,	UMONTH, or UYEAR
blank	To write a constant of value given in columns 45-70.

Column 38: (Edit codes)

blank	No editing, or use specified edit word (see below).

No Sign	CR	-	Commas	Zeros to print
1	A	J	Yes	Yes
2	B	K	Yes	No
3	C	L	No	Yes
4	D	M	No	No

X	Remove plus sign.
Y	Date field (insert "/"'s).
Z	Zero suppress.

Column 39: (Blank After)

B	Blank or zero field after writing it.
blank	Do not destroy it.

Columns 40-43: (End Position in Output Record)

1-9999	End position of field in output record (output file limited).

Column 44: (Packed or Binary Field)

-	Field is Datapoint-compatible numeric.
blank	Otherwise.

Columns 45-47: (Edit Codes)

blank	No additional editing (See Column 38).
'*'	Replace leading zeroes with asterisks.
'$'	Floating dollar sign. (These are used in conjuction with Column 38 edit codes 1-M.)

Columns 45-70: (Constant or Edit Word)

Constant	Constant to be printed, columns 32-37 (field name) must be blank.
Edit Word	To be used with numeric field named in columns 32-37 (field name).
edit mask	Mask to control editing of the field within the Edit Word when column 38 is blank:

Mask:	Function:
$	Fixed or floating dollar sign.
*	Replace leading zeroes by asterisks.
0	End suppression of leading zeroes.
&	Replace this character with a space.
- or CR	Write sign if negative.
blank	Replace with the next consecutive digit of the field.
other	Write this character here -- may not begin an edit mask.

ge B-2 Assign Phase Summary

reads:

- Define
  and %%
  %%%
  %%%%%%%%%%%%%%%%%%%%%%%
  %%table
  and field storage, generate table storage.

ould read:

- Define and assign control field storage.

- Assign file working areas.

- Define and assign match field storage.

- Scan extension, input and calculation compressions, define table and field storage, generate table storage.

ge N-10

reads:

lumn 43: (Packed or Binary Field)

- If an execution time array is in Databus - compatible format.

ould read:

lumn 43: (Packed or Binary Field)

- If an execution time array is in Databus - compatible format and has no decimal point in the input.

D If an execution time array is in Databus - compatible format and has a decimal point in the input.

ge N-15

reads:

lumn 43: (Datapoint - compatible format)

- If data in Datapoint - compatible decimal format.

Addendum to RPG User's Guide
Model #50325

April, 1978

Page 5-6, paragraph 5.10 Column 43 (Packed or Binary Field)

As reads:

Entry	Explanation
-	Data for table or array is in Databus - compatible format.

Should read:

Entry	Explanation
-	Data for table or array is in Databus - compatible format and has no decimal point in the input.
D	Data for table or array is in Databus - compatible format and has a decimal point in the input.

Page 7-10, paragraph 7.10 Column 43 (Packed or Binary Field)

As reads:

Entry	Explanation
Blank	Field is in IBM - compatible decimal format, or is alphanumeric.
-	Field is in Datapoint - compatible decimal format.

Should read:

Entry	Explanation
Blank	Field is in IBM - compatible decimal format, or is alphanumeric.
-	Field is in Datapoint - compatible decimal format and has no decimal point in the input.
D	Field is in Datapoint - compatible decimal format and has a decimal point in the input.

Should read:

Column 43: (Datapoint - compatible format)

- If data is in Datapoint - compatible decimal format and has no decimal point in the input.

D If data is in Datapoint - compatible decimal format and has a decimal point in the input.