UNISYS

REPORTER III Report Language

Operations Reference Manual

Relative to Release Level 2.0

Priced Item

January 1985

Printed in U S America
1177185

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

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

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

Comments or suggestions regarding this document should be submitted on a Field Communication Form (FCF) with the Class specified as "2" (System Software), the Type specified as "1" (F.T.R.), and the Product specified as the seven-digit form number of the manual (for example, "1177185"). The FCF should be sent to the following address: Unisys Corporation, Product Assurance and Support, 19 Morgan, Irvine CA 92718-9958 USA.

UNISYS

Product Information Announcement

○ New Release ○ Revision ● Update ○ New Mail Code

Title

REPORTER III Report Language Operations Guide

This Product Information Announcement announces the release of Update 3 to the January 1985 publication of the *REPORTER III Report Language Operations Guide*. This update is relative to the A Series Mark 4.0.2 System Software Release, dated October 1993.

This update documents enhancements to REPORTER III to process and generate COBOL85 programs. These enhancements are being made to support users migrating from V Series systems to A Series systems.

Remove	Insert
	iiA through iiD
xi through xii	xi through xii
1–1 through 1–2	1–1 through 1–2
2–17 through 2–18	2–17 through 2–18B
4–35 through 4–36	4–35 through 4–36
4–47 through 4–48	4–47 through 4–48
4–79 through 4–80	4–79 through 4–80
4–137 through 4–138	4–137 through 4–138
4–161 through 4–162	4–161 through 4–162
4–179 through 4–182B	4–179 through 4–182B
4–241 through 4–242	4–241 through 4–242
4–333 through 4–334	4–333 through 4–334
6–1 through 6–16	6–1 through 6–16
A–1 through A–4	A–1 through A–4
C–25 through C–26	C–25 through C–26B
C–39 through C–40	C–39 through C–40

Changes are indicated by vertical bars in the margins of the replacement pages.

Retain this Product Information Announcement as a record of changes made to the base publication.

To order additional copies of this document

- United States customers, call Unisys Direct at 1-800-448-1424.
- All other customers, contact your Unisys Sales Office.
- Unisys personnel, use the Electronic Literature Ordering (ELO) system.

Announcement only:

Announcement and attachments:
AS244

System: REPORTER III
Release: Mark 4.0.2 October 1993

Part number: 1177185–003

UNISYS **Publication Change Notice (PCN)**

Date	Form—PCN number
10/25/88	1177185-002

Title: REPORTER III Report Language Operations Guide
(Relative to the Mark 2.4 System Software Release)

Description

This PCN provides revisions to the REPORTER III Report Language Operations Guide, relative to the 2.4 Software Release.

Revisions to the text are indicated by black vertical bars on the affected pages.

Replace These Pages	Add These Pages
iii thru xi	4-92A
4-1	4-94A
4-17 thru 4-19	4-98A
4-23 thru 4-25	4-148A
4-91 thru 4-99	4-182A
4-145 thru 4-147	4-242A
4-153 thru 4-155	4-312A
4-163 thru 4-169	C-40A
4-181	
4-187	
4-241	
4-297	
4-311	
4-341 thru 4-347	
C-7	
C-21	
C-27	
C-39	
Index 1 thru 9	

Retain the PCN cover sheet as a record of changes made to the basic publication.

Distribution Codes SC, SD, SE

1177185-002
Printed in U S America

JNISYS Publication Change Notice (PCN)

te

01/20/86

Form—PCN number

1177185-001

le

EPORTER III Report Language User's Guide (January 1985)

scription

iis PCN provides revisions to the REPORTER III Report Language User's Guide, elative to the 2.1 Software Release. Revisions to the text are indicated by lack vertical bars on the affected pages.

Replace These Pages	Add This Page
3-1	3-2A
3-3	
4-91	
4-151	
4-177 thru 4-179	
4-225	
4-271	
6-9	
6-23	
6-35	
A-3	
B-1 thru B-3	
B-7 thru B-9	
C-25	
C-37	

tain this PCN cover sheet as a record of changes made to the basic publication.

Printed in U S America

Page Status

Page	Issue
iiA through iiC	–003
iiD	Blank
iii through x	–002
xi	–003
xii	Blank
1–1 through 1–2	–003
1–3 through 1–5	–000
1–6	Blank
2–1 through 2–17	–000
2–18 through 2–18A	–003
2–18B	Blank
2–19 through 2–35	–000
2–36	Blank
3–1	–000
3–2 through 3–2A	–001
3–2B	Blank
3–3 through 3–4	–001
3–5 through 3–19	–000
3–20	Blank
4–1 through 4–2	–002
4–3 through 4–16	–000
4–17 through 4–20	–002
4–21 through 4–22	–000
4–23 through 4–26	–002
4–27 through 4–35	–000
4–36	–003
4–37 through 4–46	–000
4–47	–003
4–48 through 4–79	–000
4–80	–003
4–81 through 4–90	–000
4–91 through 4–92	–002
4–92A through 4–92B	–002
4–93 through 4–94	–002
4–94A through 4–94B	–002
4–95 through 4–98	–002
4–98A through 4–98B	–002
4–99 through 4–100	–002
4–101 through 4–137	–000
4–138	–003
4–139 through 4–144	–000
4–145 through 4–148A	–002

continued

continued

Page	Issue
4–148B	Blank
4–149 through 4–150	–000
4–151	–001
4–152	–000
4–153 through 4–156	–002
4–157 through 4–160	–000
4–161	–003
4–162	–000
4–163 through 4–170	–002
4–171 through 4–176	–000
4–177 through 4–178	–001
4–179	–003
4–180 through 4–181	–000
4–182 through 4–182A	–003
4–182B	–002
4–183 through 4–186	–000
4–187 through 4–188	–002
4–189 through 4–224	–000
4–225	–001
4–226 through 4–240	–000
4–241	–002
4–242	–003
4–242A	–002
4–242B	Blank
4–243 through 4–271	–000
4–272	–001
4–273 through 4–296	–000
4–297 through 4–298	–002
4–299 through 4–310	–000
4–311 through 4–312A	–002
4–312B	Blank
4–313 through 4–332	–000
4–333	–003
4–334 through 4–340	–000
4–341 through 4–348	–002
4–349	–000
4–350	Blank
5–1 through 5–15	–000
5–16	Blank
6–1	–000
6–2	–003
6–3 through 6–4	–000
6–5 through 6–6	–003
6–7	–000
6–8 through 6–9	–003
6–10	–001
6–11	–003
6–12	–000
6–13 through 6–14	–003
6–15	–000

continued

continued

Page	Issue
6–16	–003
6–17 through 6–23	–000
6–24	–001
6–25 through 6–34	–000
6–35	–001
6–36 through 6–37	–000
6–38	Blank
A–1 through A–3	–003
A–4	Blank
B–1	–001
B–2	–000
B–3	–001
B–4 through B–7	–000
B–8 through B–9	–001
B–10 through B–11	–000
B–12	Blank
C–1 through C–6	–000
C–7 through C–8	–002
C–9 through C–20	–000
C–21 through C–22	–002
C–23 through C–24	–000
C–25 through C–26A	–003
C–26B	Blank
C–27 through C–28	–002
C–29 through C–36	–000
C–37	–001
C–38	–000
C–39 through C–40	–003
C–40A	–002
C–40B	Blank
C–41 through C–43	–000
C–44	Blank
D–1 through D–4	–000
E–1 through E–7	–000
E–8	Blank
1 through 9	–002
10	Blank

Unisys uses an 11-digit document numbering system. The suffix of the document number (1234 5678–*xyz*) indicates the document level. The first digit of the suffix (*x*) designates a revision level; the second digit (*y*) designates an update level. For example, the first release of a document has a suffix of –000. A suffix of –130 designates the third update to revision 1. The third digit (*z*) is used to indicate an errata for a particular level and is not reflected in the page status summary.

TABLE OF CONTENTS

INTRODUCTION

This manual defines the report language of the REPORT writER III (REPORTER III) system and its use in the system. The report language is used in conjunction with the vocabulary language of the system to prepare reports reflecting information in the user data files or data bases. Before a report can be produced, a vocabulary of terms that describe pertinent information in the data files or data bases must be designed and created. When this vocabulary exists, a report-language specification which uses it can be designed and processed to produce the desired report.

This manual is directed to all users of the report language. It presents basic information about the language and system to the first-time or occasional user, while it is arranged to allow easy reference by the more experienced user. It furnishes all users with the detailed information they need to design and specify their reports. It is anticipated that as the user gains experience in the report language, the REPORTER III Reference Card will be increasingly relied on when composing report-language specifications.

The contents of the manual are organized as follows:

Section 1	Section 1 of this manual provides a general description of REPORTER III, including features of the system.
Section 2	Section 2 presents basic information about the report language.
Section 3	Section 3 is a guide for designing and specifying reports.
Section 4	Detailed explanations of all language statements, clauses, language constructs, and functions used to design and prepare reports are provided in Section 4, in alphabetical order.
Section 5	Examples of report-language specifications and the reports produced are presented in Section 5.

Section 6	Section 6 contains information on system operations associated with report preparation.
Appendices	The four-phase system process of report preparation is depicted and discussed briefly in Appendix A. Limits and defaults associated with report-language specification are indicated in Appendix B. Error and warning messages associated with system analysis of report-language specifications are listed and explained in Appendix C. Exceptions also are listed and explained, separately, in Appendix C. Appendix D presents a list of all reserved words of the report language. Appendix E is a glossary of terms used within this manual.

The following manuals explain the use of the REPORTER III system and the optional On-Line REPORTER III module.

Operations Guide, REPORTER III Vocabulary Language (Relative to 2.4 Software Release), form 1177177.

Reference Card, REPORTER III (Relative to 2.0 Software Release), form 1177318.

Capabilities Manual, REPORTER III (Relative to 2.0 Software Release), form 1177300.

User's Guide, On-Line REPORTER III (Relative to 2.0 Software Release), form 1177151.

Note that the "REPORTER III Vocabulary Language Operations Guide" was formerly titled "REPORTER III Vocabulary Language User's Guide." Subsequent references to the guide do not reflect its new title.

The REPORTER III System is designed for use with the following:

1. A Series of Systems.
2. B 1000 Series of Systems.
3. B 2000/B 3000/B 4000 Series of Systems.
4. B 5000/B 6000/B 7000 Series of Systems.

Note that throughout this guide, instructions for A Series also pertain to B 5000/B 6000/B 7000 Series of Systems. Also note that when the word "COBOL" is used, it refers to the appropriate ANSI-74 COBOL for each system. If you are using an A Series System, the term "COBOL" also refers to ANSI-85 COBOL.

SECTION 1

SYSTEM DESCRIPTION

The REPORTER III System provides an effective means for creating a wide variety of management reports reflecting information maintained on a computer system in the Unisys A Series, and B 1000 through B 7000 Series of Systems.

SYSTEM FEATURES

REPORTER III greatly facilitates the retrieval, analysis, and reporting of data by using the power of the computer system to perform tasks such as the following:

1. Select data on the basis of simple to complex criteria.

2. Match records on the basis of data field values.

3. Sort data as specified in ascending and/or descending order according to multiple keys.

4. Automatically age data, using a variety of date formats.

5. Determine statistics including count, total, average, maximum, minimum, sum squares, mean squares, variance, and standard deviation.

6. Handle multilevel control breaks and ranges, and give summary statistics for each control break or range.

7. Automatically format information for printed reports.

8. Automatically schedule phases of report processing, while providing override options.

9. Create one or more files of extracted data for subsequent processing or reporting.

An important aspect of REPORTER III is that its features can be used in auditing applications. The system is applicable to both internal and external auditing activities. REPORTER III provides the auditor with an effective means to test and evaluate the information maintained on the computer system. It greatly aids the auditor's analysis by using the power of the computer system to perform such tasks as the following:

1. Test derived data items (extensions) and footings.

2. Select and print audit samples.

3. Examine records for completeness, consistency, and valid conditions.

4. Summarize data.

5. Compare duplicate or related data for correctness and consistency.

6. Compare audit data with computerized records.

7. Extract information for subsequent processing and evaluation.

8. Print confirmation letters.

The REPORTER III System can produce multiple reports in one pass of the input data. Also, the system enables you to control the formatting of reports as well as output information on preprinted forms, if desired.

REPORTER III can report information from data files contained on magnetic tape, disk, and punched cards. The system also can report information from DMS II data bases.

LANGUAGE INTERFACE TO SYSTEM

The REPORTER III System includes two free-form languages:

1. Vocabulary Language - a language designed to create a vocabulary (dictionary) of descriptions and definitions of the data to be reported. The vocabulary language can accept an independent description, or it can use existing COBOL record descriptions and/or data-base directories.

2. Report Language - a report-description language which enables the specification of a wide variety of reports. The language is easily used by non-programmers.

REPORTER III generates a COBOL program tailored to your exact reporting requirements. Note that whenever "COBOL" is referred to, it means the appropriate ANSI-74 COBOL or ANSI-85 COBOL for A Series systems or the appropriate ANSI-74 COBOL for any system other than A Series.
The generative approach

speeds recurrent reporting, and it also provides the flexibility needed for cost-effective one-time reporting.

REPORT LANGUAGE AND SYSTEM OPERATION

The REPORTER III report language provides a method for quickly specifying reports based on information contained in the data files or data bases. With appropriate selection, grouping, ordering, and summarizing, the information is furnished in a useful report form by the system. Information can be furnished as a printed report, or it can easily be extracted in machine-readable form to a separate file for further processing.

The report language is free-form, "English-like," and concise. Little writing is necessary to prepare the report-language specification, which describes the contents of the report. Yet the report-language statements are readable and self-documenting. The language includes many default features and thus does not require that you specify each option. Each data item of information is specified by name, and you are not required to know its size or format characteristics. The format of the report, whether simple or complex, is determined by REPORTER III, unless you choose to specify the format requirements.

The report-language specification which you have written (also referred to as the "report specification") is input to the Report Language Analysis Program (RP3REP), which analyzes your request and prints a listing of the specification as given. If the request is invalid, the errors are indicated by means of appropriate messages, and you must correct the mistakes and resubmit the specification. If the request is valid, a parameter file is produced which represents your specific request complete with "physical" information obtained from the referenced vocabulary. The report-program generator (RP3GEN) then is run automatically to generate a COBOL source program based on the input parameter file. The generated COBOL source program is then compiled and executed automatically, and the desired report is produced.

The generated report program can be treated either as a one-time-only report program or as a recurring report program which is run as often as required without regeneration.

A more detailed explanation of system operation, together with a diagram representing the report-preparation process, is contained in Appendix A.

EXAMPLE REPORT-LANGUAGE SPECIFICATION

The following is an example of a report-language specification, shown together with the job control language statements needed to produce the report. The control statements are preceded by a question mark (?). This example assumes input of the specification as a card deck.

```
?EXECUTE RP3REP
?DATA RP3CRD
      VOCABULARY IS "CLIENT".
      INPUT ACCTS-RECV.
      SELECT BALANCE-DUE GREATER THAN CREDIT-LIMIT.
      TITLE "ACCOUNTS OVERDRAWN".
      REPORT CUST-NO, BRANCH, CREDIT-LIMIT, BALANCE-DUE.
      SUMMARIZE FOOTING BALANCE-DUE.
?END
```

The six report language statements in this specification are explained below. The specification is designed to provide a report containing information reflecting all accounts which are overdrawn.

The VOCABULARY statement identifies the vocabulary of names used in this report language specification to describe the information in the data base. The vocabulary is created by the program RP3VOC. (Figure 2-7 lists the vocabulary identified here.)

The INPUT statement specifies that all accounts receivable information be input. (ACCTS-RECV is a DMS II data set which can be thought of as a file.)

The SELECT statement specifies that only those accounts in which the balance due is greater than the customer credit limit be reported.

The TITLE statement describes the title to be placed at the top of each page of the report.

The REPORT statement specifies that the customer number, branch number, credit limit, and balance due be reported for each selected customer account.

The SUMMARIZE statement specifies that the balance due column be footed. This gives the total balance due of all overdrawn accounts.

Figure 1-1 shows the report produced from this specification. The requested information has been formatted automatically into a readable report.

PAGE 1

ACCOUNTS OVERDRAWN

CUST NO	BRANCH	CREDIT LIMIT	BALANCE DUE
001302	0013	1000	$ 1311.80
002117	0020	1000	$ 1560.00
051231	0013	500	$ 732.00
081380	0008	2000	$ 2016.80
100500	0034	3000	$ 3810.21
101137	0130	500	$ 511.37
102800	0130	5000	$ 6387.50
120060	0038	500	$ 643.70

SUMMARIES FOR FINAL
TOTAL $16973.38

Figure 1-1. Sample Report

Other examples of report-language specifications are provided in Sections 2 and 5.

SECTION 2

BASIC INFORMATION ABOUT THE REPORT LANGUAGE

The REPORTER III report language is a high-level language based on English and composed of characters, words, and statements. This section documents basic elements of the report language and the method used to define the language, and presents some basic terminology used for data identification. It also describes the various types of vocabulary names which can be specified in appropriate contexts in the report language. Finally, sample vocabularies used in examples throughout this manual to illustrate various language constructs and statements are presented.

For brief explanations of basic language constructs and functions used in defining information in language statements, refer to Section 3. Detailed explanations of the constructs and functions are provided in Section 4.

CHARACTER SET

The report language character set consists of the digits 0 through 9, the letters A through Z, the blank or space, and the following symbols:

Symbol	Definition
*	asterisk or multiplication sign
@	at sign
[	bracket, left
]	bracket, right
:	colon
,	comma
$	dollar sign
=	equal sign
>	greater than symbol
<	less than symbol
-	minus sign or hyphen
#	number sign
(	parenthesis, left
)	parenthesis, right
%	percent sign
.	period or decimal point
+	plus sign
"	quotation mark
;	semicolon
/	slash or division sign

SPECIFICATION FORM

The report language can be written on any COBOL-compatible coding form to facilitate keypunching or data entry of card images. You write the free-form statements between columns 8 and 72, using one or more spaces or appropriate punctuation to delimit elements of the language. Columns 1 through 6 can be used for sequence numbers. Columns 73 through 80 are available for optional use, such as identification, remarks, etc. The sequence number fields can be used later to indicate the sequence numbers corresponding to source language statements which are to be updated. Column 7 is not used.

COMMENT INDICATOR

A percent sign (%) placed in columns 8 through 72 indicates that the characters which follow the sign are a part of a user comment. These comments are not part of the report language and, therefore, need not follow the rules of the language. They are useful for documentation of the language specifications.

Examples:

```
% AN EXAMPLE OF THE USE OF COMMENTS.
REPORT A,B, % COMMENTS MAY EVEN
C,D AS "VALUE", % BE USED INSIDE
E as "%", F, G. % A CONSTRUCT.
```

NOTE

A percent sign placed between quotes, as in the string "%", is not interpreted as a comment.

DEFINITION OF WORDS

A word is a combination of not more than 30 characters which can consist of the alphabetic characters A through Z, the numeric characters 0 through 9, and the hyphen (-). A word must contain at least one alphabetic character. It cannot begin or end with a hyphen.

Report specifications are constructed with English-like statements composed of report language words, symbols, and punctuation.

RESERVED WORDS

Reserved words are English words and their abbreviations which are system-defined as part of the report language. A list of the reserved words in the report language is contained in Appendix D. Although reserved words can be used in some statements in which they do not function as reserved words, such usage is discouraged and should be avoided because it can cause subsequent errors. Reserved words can be keywords or optional words.

Keywords

The reserved-word category of keywords includes the words or portions of words required to complete the meaning of statements and entries. The category also includes words or portions of words that have a specific functional meaning. In the statement A IS GREATER THAN B or A GREATER B, the keyword is GREATER.

Optional Words

Optional words are reserved words included in the report language to improve the readability of the statement formats. The optional words may be included or omitted by the user. For example, A IS GREATER THAN B is equivalent to A GREATER B; the inclusion or omission of the words IS and THAN does not affect the logic of the statement.

NAMES

A <name> is a user-defined word. The <name> is defined as part of the report language by its existence in a referenced vocabulary or its definition in the report language specification. In general, <name>s should not be chosen which are identical to reserved words in the report language because, in certain contexts, such <name>s might be mistaken for the reserved words.

METHOD OF LANGUAGE DEFINITION

The report language is defined by rules of syntax and semantics. The syntactic rules determine the structure of valid report-language statements. The semantic rules determine which report-language statements have valid meanings and what those meanings are.

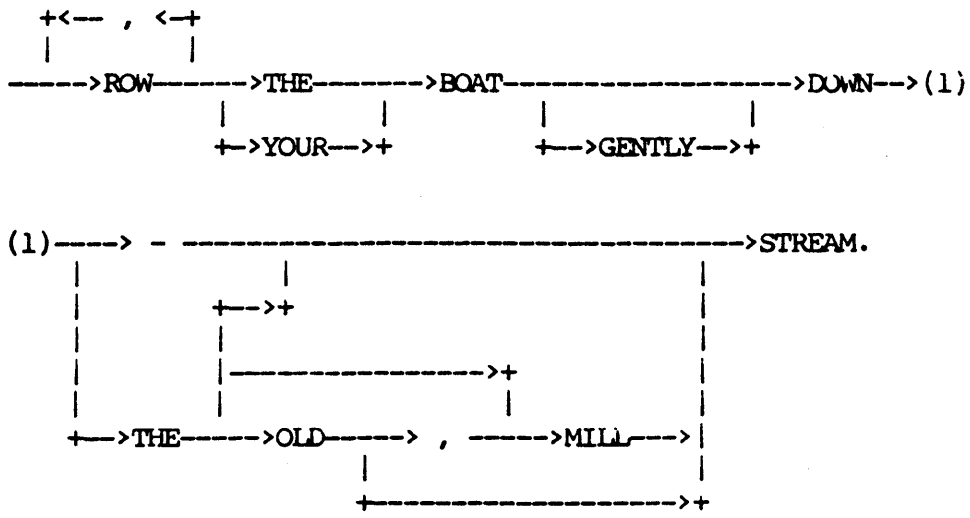
SYNTAX DIAGRAMS

The syntactic rules of the report language are described by syntax diagrams constructed of words and arrows. A syntactically correct statement is produced by tracing any path along the direction of the arrows in a given syntax diagram. Words and symbols are written as they are encountered along the line paths.

The syntax diagrams occasionally must be continued on new line(s). In this case, the break in an arrow is shown by means of connectors using the same number at each end of the break.

An example syntax diagram (with connectors) is shown below.

Example:

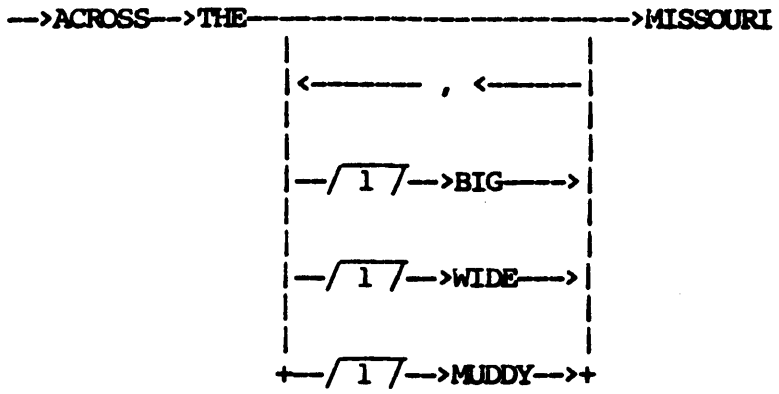


Valid productions from this syntax diagram include:

ROW THE BOAT DOWN-STREAM.
ROW, ROW, ROW YOUR BOAT GENTLY DOWN THE STREAM.
ROW, ROW, ROW, ROW THE BOAT DOWN THE OLD STREAM.
ROW YOUR BOAT DOWN THE MILL STREAM.
ROW THE BOAT DOWN THE OLD, MILL STREAM.

A bridge over a number indicates that the path can be traced a maximum number of times specified by the number under the bridge.

Example:



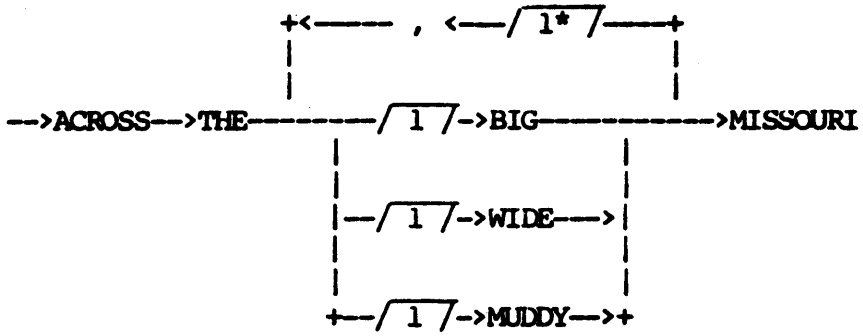
Valid productions from this syntax diagram include:

ACROSS THE MISSOURI
ACROSS THE BIG MISSOURI
ACROSS THE MUDDY, WIDE MISSOURI
ACROSS THE BIG, WIDE, MUDDY MISSOURI

but do not include:

ACROSS THE BIG, BIG MISSOURI
ACROSS THE WIDE, BIG, WIDE MISSOURI

A bridge over a number with an asterisk must be traced at least once, but not more than the maximum number of times specified by the number.



Valid productions from this syntax diagram include:

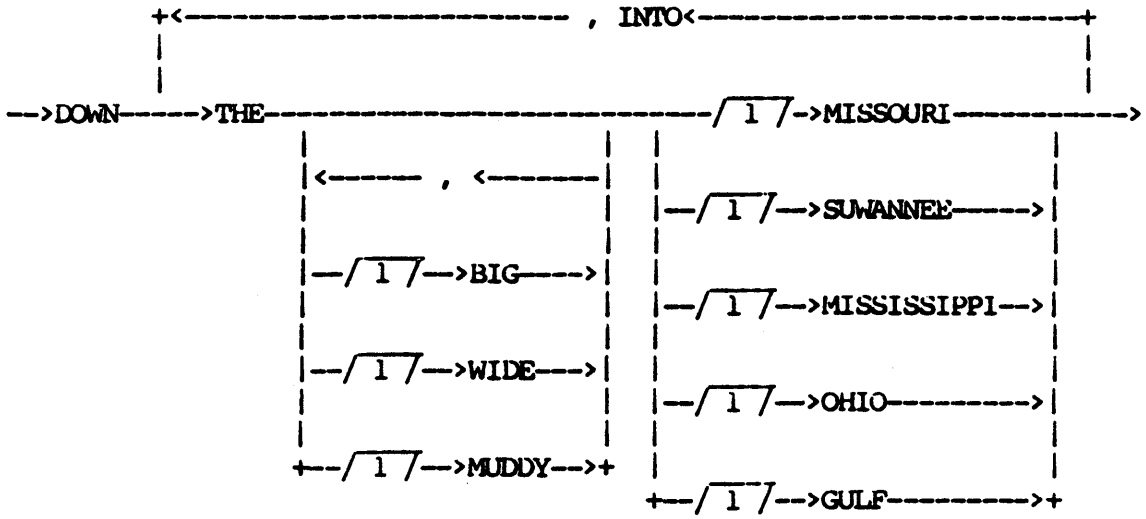
```
ACROSS THE BIG, WIDE MISSOURI
ACROSS THE WIDE, MUDDY MISSOURI
```

but do not include:

```
ACROSS THE BIG MISSOURI
ACROSS THE BIG, WIDE, MUDDY MISSOURI
ACROSS THE BIG, BIG MISSOURI
```

Number bridges usually are used with loops, as in the preceding examples. Once a loop is exited, all number bridges within that loop are "reset" for the next entry into the loop.

Example:



Valid productions from this syntax diagram include:

```
DOWN THE MISSOURI
DOWN THE BIG OHIO
DOWN THE BIG OHIO, INTO THE BIG, WIDE MISSISSIPPI,
   INTO THE BIG, WIDE, MUDDY GULF
```

but do not include:

DOWN THE BIG OHIO, INTO THE MUDDY, MUDDY MISSISSIPPI
DOWN THE BIG OHIO, INTO THE MUDDY OHIO

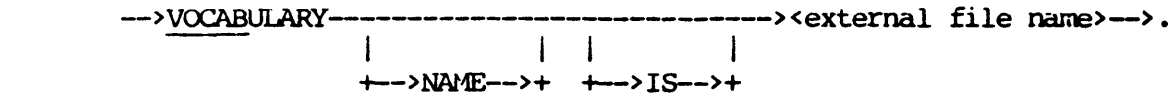
RESERVED WORDS

Words in uppercase letters are the reserved words in the language; these words must be written exactly as shown in the syntax diagrams.

Keywords are the reserved words required to complete the meaning of language statements. Some keywords can be abbreviated by omitting letters from the end of the word. The minimum required abbreviation is underlined in the syntax diagram. If a keyword must be used in its entirety, the entire word is underlined in the diagram.

Optional words are reserved words having no semantic meaning. Optional words are included in the language to improve the readability of the statement formats. They are never underlined in the syntax diagrams and, if used, cannot be abbreviated.

Example:



The reserved words are VOCABULARY, NAME, and IS. VOCABULARY is a keyword and can be abbreviated as VOCAB, VOCABU, VOCABUL, VOCABULA, or VOCABULAR. NAME and IS are optional words and must be spelled out if they are used.

PUNCTUATION

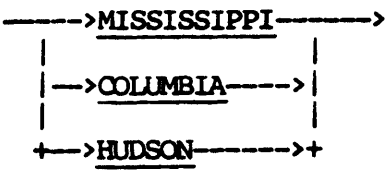
Special characters such as the parenthesis, colon, period, and comma must be written as they appear in the syntax diagrams. Keywords, optional words, names, numeric constants, and character-string constants must be separated from each other by a blank or a special character. Wherever a blank is required, several blanks optionally can be used. Blanks optionally can be placed around special characters for readability. A blank immediately before a period is not required except when a period immediately follows a numeric literal.

SYNTACTIC VARIABLES

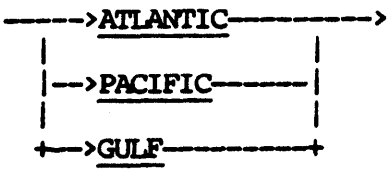
Phrases set off by angle brackets (<>) are syntactic variables which represent information to be supplied by you. A particular variable can represent a simple language element, such as an integer, character, string, or name; or it can represent a relatively complicated language construct, such as a Boolean expression. These variables are defined either by verbal description or by syntax diagrams of their own. In either case, any valid language element or construct derived from the syntactic variable can be inserted into any diagram in place of the variable.

Example:

<river>:



<ocean>:



In the first diagram, "river" and "ocean" are syntactic variables. Both are defined by the diagrams that follow. Syntactically valid productions include:

```
DOWN THE MISSISSIPPI INTO THE GULF
DOWN THE COLUMBIA INTO THE PACIFIC
DOWN THE COLUMBIA INTO THE GULF
DOWN THE MISSISSIPPI INTO THE PACIFIC
```

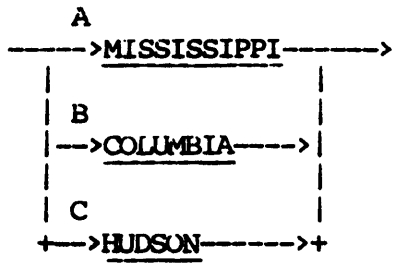
SEMANTIC RULES

Semantic rules are linked to the syntax diagrams by means of letters which label the critical paths. The letters reference paragraphs which explain the meaning associated with the corresponding syntax path, and explain additional rules associated with a choice in paths. These rules can make certain syntactically correct statements invalid.

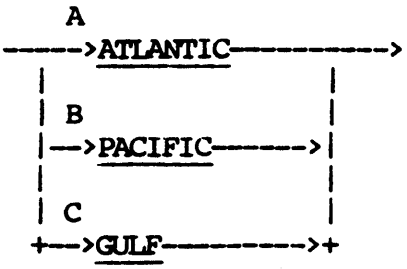
Example:

```
-->DOWN THE <river> INTO THE <ocean>--->
```

<river>:



<ocean>:



The semantic rules are written so that they prompt you to choose the correct paths and supply the appropriate information. Thus, by tracing the syntax diagram paths and reading the "path prompts," you supply the information for the report specifications which describe the application.

If the example above were actual syntax (such as the syntax described in Section 4), the semantic rules for <river> would be explained in accompanying paragraphs labeled A, B, and C. The semantic rules for

<ocean> would be explained in accompanying paragraphs labeled A, B, and C. These paragraphs would explain that the choice of river must correctly match the choice of ocean so that the river flows into the correct ocean. Semantically valid productions include:

DOWN THE MISSISSIPPI INTO THE GULF
DOWN THE HUDSON INTO THE ATLANTIC
DOWN THE COLUMBIA INTO THE PACIFIC

but do not include:

DOWN THE MISSISSIPPI INTO THE PACIFIC
DOWN THE COLUMBIA INTO THE ATLANTIC

TERMINOLOGY FOR DATA IDENTIFICATION

In specifying the semantic rules of the report language, various terms are used consistently to identify data or information. These terms are defined in the following paragraphs.

DATA ITEM OR ITEM

A data item or item is an elementary item of information. In the report language, data items are referenced by data names, numeric literals, character strings, or expressions. Examples of data items are the following: the number of parts on order, the name of the individual, the total balance of aged accounts, and the percent utilization factor.

Data items are numeric, string, or Boolean. Numeric data items have numeric values; string items have character values, and Boolean items have values of TRUE or FALSE.

GROUP

A group is a collection of related data items which could be thought of as a single data item. For example, a date can be a group consisting of the data items month, day, and year. Or an address can be a group consisting of the first, second, and third lines of the address. Groups referenced in the report language are considered string-valued data items.

RECORD

A record is a collection of related data items and/or groups. The term "record" is used to mean a record, list element, or member, depending on the term which is appropriate for the type of data structure under consideration. A record might include all information pertaining to a client's account, such as customer name, account number, and current balance. A record might consist of information pertaining to an employee, such as name, age, sex, salary, and job grade.

DATA STRUCTURE

A data structure is a collection of records. Access to information contained in data structures is indicated in a report specification by the INPUT statement. A data structure might contain information on all client accounts, or it might contain records for all June order transactions. Valid data structures for the report language include system files that can be described in COBOL, and DMS II data sets.

DATA BASE

A data base is a collection of one or more data structures and the relationships between them.

LOGICAL RECORD

A logical record is a record or a group of associated records which contains all pertinent information about a single entity. Records comprising a logical record can come from a single data structure, multiple data structures, or a user input routine. Certain information within a logical record can contain null values indicating the absence of related information. The INPUT statement in a report specification specifies what information each logical record contains and how this information is accessed.

As an example, a logical record may consist of information about a single transaction, including pertinent information about the account to which the transaction applies. As another example, each logical record may consist of information gathered from various sources about a single employee. In addition to containing such information as name and salary from the payroll master file, the logical record could contain employee address information obtained from the address file. Information about the employee's job grade and department might be obtained from records in separate but related files. Of course, job grade and department

information within the logical record probably would be identical for many employees (see Figure 2-1).

A logical record as defined by the INPUT statement is an underlying concept behind the report language description. Specifications can be given to extend, select, sample, group, order, summarize, extract, and report the information contained in the logical records.

Logical records can be viewed also as rows within a large table (see Figure 2-1). A column in this table would represent a single item. The table is first defined by the INPUT statement in the report specification. Subsequent language statements in the report specification can extend, select, sample, group, order, summarize, extract, and report the information contained in the rows of the table.

A logical record can be accessed only once since only one pass is made through the specified data per report specification when the report program is run. In terms of preparing a report specification, this means that once you have specified access to a logical record, you cannot specify access to that logical record again.

Thus, for example, the following combination of physical records within a particular logical record cannot be reflected in a report specification because such a combination would involve more than one pass through the specified data:

```
PHYSICAL RECORD 1   PHYSICAL RECORD 2
PHYSICAL RECORD 1   PHYSICAL RECORD 2
PHYSICAL RECORD 3   PHYSICAL RECORD 4
PHYSICAL RECORD 3   PHYSICAL RECORD 4
```

The term "row" is substituted for the term "logical record" in various parts of this manual.

NAME	SALARY	AGE	ADDRESS	JOB GRADE	DEPARTMENT	DEPT.NO.
DOE, JOHN	532.00	21	101 HIGH ST.	3	EDP	0611
SMITH, JOE	1,100.00	27	332 MADISON	7	EDP	0611
BAKER, BILL	632.05	27	1222 3RD ST.	7	ACCT	0232
BAKER, SUE	711.00	23	27 RIVER RD.	7	ACCT	0232
JAMES, HELEN	883.00	37	1001 51 ST.	5	EDP	0611
JONES, BOB	930.00	41	531 PEACH ST.	5	MKTG	0110
KELLY, MARY	650.30	33	011 3RD AVE.	5	EDP	0611
NEWMAN, KEN	1,232.00	52	1311 CYPRESS	7	MKTG	0110
SMITH, SUE	766.45	24	303 WEST 8TH	7	MKTG	0110
HILL, DON	811.50	32	200 EAST 3RD	7	STAFF	0777
JONES, KAREN	950.61	30	17 VALLEY DR.	5	STAFF	0777

Figure 2-1. Logical Records for Personnel

CONTROL-BREAK ITEM

A control-break item is a data item used to group the information to be reported. All logical records which contain the same value of the control-break item are grouped together for purposes of reporting the information and calculating statistics regarding the information. For example, all transactions can be grouped by account number. For purposes of reporting, the information about a transaction can then be listed under the appropriate account-number heading. In addition, summaries such as the average transaction amount for each account can be obtained.

A control-break item can be subordinate to another control-break item which, in turn, can be subordinate to yet another control-break item. (The limit on the number of levels of control-break items is nine.) Information is then hierarchically grouped based on values of the control-break items. For example, transactions can be grouped based on account number, and accounts can be grouped based on account type. Thus for each account type there are many account numbers and, for each

account number, there are many different transactions. In Figure 2-2, the personnel data shown in Figure 2-1 is grouped by department and job grade.

DEPARTMENT	DEPT.NO.	GRADE	NAME	SALARY	AGE	ADDRESS
ACCT	0232	7	BAKER, BILL	632.05	27	1222 3RD ST.
ACCT	0232	7	BAKER, SUE	711.00	23	27 RIVER RD.
EDP	0611	3	DOE, JOHN	532.00	21	101 HIGH ST.
EDP	0611	5	JAMES, HELEN	883.00	37	1001 51 ST.
EDP	0611	5	KELLY, MARY	650.30	33	011 3RD AVE.
EDP	0611	7	SMITH, JOE	1,100.00	27	332 MADISON
MKTG	0110	5	JONES, BOB	930.00	41	531 PEACH ST.
MKTG	0110	7	NEWMAN, KEN	1,232.00	52	1311 CYPRESS
MKTG	0110	7	SMITH, SUE	766.45	24	303 WEST 8TH
STAFF	0777	5	JONES, KAREN	950.61	30	17 VALLEY DR.
STAFF	0777	7	HILL, DON	811.50	32	200 EAST 3RD

Figure 2-2. Logical Records for Personnel Grouped by Control Breaks

RANGE-BREAK ITEM

A range-break item is a special control-break item which is based on specified value ranges of a particular data item. This data item is referred to as a ranged item. All logical records which contain values of the ranged item within specified limits are grouped together for purposes of reporting the information and calculating statistics on the information. To accomplish this, the information is ordered in ascending sequence based on the ranged item. For example, all transactions can be grouped based on transaction amount such that all transactions within the following ranges are grouped together: 0 to 1000 dollars, 1000 to 5000 dollars, 5000 to 10,000 dollars, and over 10,000

dollars. A count of all transactions for each defined amount range can be obtained.

The range-break item itself is a string item whose values indicate a lower- and upper-limit of the ranged item. A range-break item value can be assigned to each logical record to indicate the range to which it belongs. In Figure 2-3, the personnel data shown in Figure 2-1 is grouped based on age, such that all employees within the following age groups are grouped together: 20 to 29, 30 to 39, 40 to 49, and 50 to 59.

DEPARTMENT	DEPT.NO.	JOB GRADE	NAME	SALARY	AGE	ADDRESS	RANGE
EDP	0611	3	DOE, JOHN	532.00	21	101 HIGH ST.	20-29
ACCT	0232	7	BAKER, SUE	711.00	23	27 RIVER RD.	20-29
MKTG	0110	7	SMITH, SUE	766.45	23	303 WEST 8TH	20-29
EDP	0611	7	SMITH, JOE	1,100.00	27	322 MADISON	20-29
ACCT	0232	7	BAKER, BILL	632.05	27	1222 3RD ST.	20-29
STAFF	0777	5	JONES, KAREN	950.61	30	17 VALLEY DR.	30-39
STAFF	0777	7	HILL, DON	811.50	32	200 EAST 3RD	30-39
EDP	0611	5	KELLY, MARY	650.30	33	011 3RD AVE.	30-39
EDP	0611	5	JAMES, HELEN	883.00	37	1001 51 ST.	30-39
MKTG	0110	5	JONES, BOB	930.00	41	531 PEACH ST.	40-49
MKTG	0110	7	NEWMAN, KEN	1,232.00	52	1311 CYPRESS	50-59

Figure 2-3. Logical Records for Personnel Grouped by Range Break

INPUT DATA ITEM

An input data item is an item which is part of the information to be input for the report. For a single-report specification or the <input section> of a multiple-report specification, input data items are those described in the specified vocabulary; these data items are defined by reference to the appropriate data structures in the INPUT statement and by extensions. For the <report section> of a multiple-report specification, input data items are defined in the <input section> by the INPUT statement and in the <report section> by extensions.

ACCEPTED DATA ITEM

An accepted data item is a data item which is defined in the ACCEPT statement. This data item serves as a run-time parameter for report specification. The actual value of the data item is read in by the generated report program before any processing begins.

DERIVED DATA ITEM

A derived data item is a data item which is described in terms of other data items via an <item desc>. The value of this data item is "derived" from input items, accepted items, constants, or other derived data items. Combinations of arithmetic operations, logical operations, and REPORTER III intrinsic functions can be used to compute the derived data item. A derived data item represents an extension of the logical record.

STATISTICAL DATA ITEM

A statistical data item is a derived data item which is defined in terms of one or more statistical functions or other statistical data items. For example, the company payroll, when derived as the total salary of all employees, is a statistical data item.

NONSTATISTICAL DATA ITEM

A nonstatistical data item is an input data item, an accepted data item, or a derived data item which is not defined in terms of any statistical function or statistical data item.

SUMMARY ITEM

A summary item is an item which summarizes a group of information. It can be statistical or nonstatistical. A summary item for a particular control break has only one value for each control-break value. For example, when reporting on company personnel records grouped by department, the department location can be thought of as a summary item related to the department control break. Also, the average salary for each department can be derived as a statistical item and as a summary item related to the department control break.

VOCABULARY NAMES

Information within the data base is referenced in the report language by names which are defined within a given vocabulary. Vocabularies are constructed from vocabulary language (RP3VOC) specifications. A particular vocabulary is supplied to the report language processor by the VOCABULARY statement. Only names within the supplied vocabulary can then be used in the report specification.

A listing of the vocabulary supplies the names and description of entries in the vocabulary. Vocabulary entries describe data items, groups, records, and data structures within the data base. In addition, vocabulary entries can describe accessing techniques, specific data-item values, macros, and input routines. (See Figures 2-4 through 2-9 for examples of vocabulary listings.) Certain names within the vocabulary may be duplicated, in which case they must be qualified by other names to identify them uniquely. (This qualification is governed by the rules of COBOL.) A proper qualification is given in the vocabulary listing for duplicate names. Listed and described briefly below are the types of names which can appear in a vocabulary listing and thus be specified in appropriate contexts in the report language.

DATA-ITEM NAME

A data-item name refers to a data item described in the vocabulary (vocabulary item). A data-item name can be qualified and subscripted. (Refer to the explanation of <data name> in Section 4.)

GROUP NAME

A <group name> refers to a group of data items described in the vocabulary. A <group name> can be qualified and subscripted. (Refer to the explanation of <data name> in Section 4.)

When referenced in the report language, a group is treated as a string-type item. When using a vocabulary created prior to the 2.50 software release, all group items are assigned a default character length that can be changed by the Process-option statement SET STRING-SIZE. (Refer to the explanation of the Process-option statement in Section 4.)

For a vocabulary created using the 2.50 or later software release, the size of the group item reflects the size of data items subordinate to that particular group item. Both the size of the group item and the release level used to create the vocabulary can be found in the vocabulary report.

MACRO NAME

A <macro name> refers to a permanent macro included in the vocabulary. A permanent <macro name> can be referenced any number of times in the report specification once the VOCABULARY statement is given. If formal parameters were defined for the macro, the values of the actual parameters must be specified whenever the macro is referenced. The appropriate macro text replaces each reference to the permanent <macro name>.

CONDITION NAME

A <condition name> refers to a COBOL 88-level <condition name>, which represents one or more specific values of a data item. A <condition name> can be qualified. In the report language (except when using the PRESELECT clause), a <condition name> may be referred to only in the following context:

<data name> = <condition name>

or

<data name> NOT = <condition name>

For example, the <data name> THIS-MONTH identifies the 12 months of a year with subordinate <condition name>s defined as JANUARY through DECEMBER having assigned values of 01 to 12. The following statements use the <condition name>s present in the vocabulary.

```
SELECT THIS-MONTH = FEBRUARY OR THIS-MONTH = MAY.
REPORT THIS-MONTH, AMOUNT.
```

The preceding statements result in the month being made available for reporting and the proper internal code being compared for selection. An example of the output resulting from the statements follows.

Example:

```
THIS MONTH        AMOUNT

FEBRUARY          10.50
FEBRUARY           2.75
MAY               89.96
MAY               12.24
MAY               18.95
.                 .
.                 .
.                 .
```

RECORD NAME

A record name refers to a COBOL 01 level record which contains data items at subordinate levels. A record name appears in the vocabulary listing as a group name and can be used as such.

FILE NAME

A <file name> refers to a COBOL-described standard card, tape, or disk file. The <file name> represents the internal name rather than the external name of the file.

LINK NAME

A <link name> refers to an A Series DMS II-defined link item. The <link name> can be qualified.

DATA-BASE NAME

A <data-base name> is the name of a DMS II data base.

DATA-SET NAME

A <data-set name> refers to a DASDL-defined (Data And Structure Definition Language-defined) DMS II data set. A <date-set name> can be qualified.

SET NAME

A <set name> refers to a DASDL-defined DMS II set or subset which spans a data set. A <set name> can be qualified.

INPUT-PROCEDURE NAME

An <input-procedure name> designates a user-supplied COBOL input procedure. An input procedure is defined to RP3VOC by the user in order to access data structures not handled by standard REPORTER III-supplied input routines. One call on the input procedure returns one record. That record contains data from one or more records accessed from one or more data structures by the procedure.

VOCABULARY EXAMPLES

Figures 2-4 through 2-9 show listings of six sample vocabularies used in examples throughout this manual to illustrate various language constructs and statements. It is assumed that RP3VOC was used to create these vocabularies. All examples in the manual that use one of these six vocabularies identify the name of the vocabulary. Section 5 describes REPORTER III applications that use the information described by the vocabularies.

While most of the vocabulary listing is self-explanatory, various entries require some additional explanation. Explanations of these entries follow.

LEVEL describes the hierarchical level of an element. That is, it describes which elements are actually part of other elements. For example, items in many instances are described as part of a group.

The level of an element in relation to other elements is indicated by a number (level number). Elements assigned the same number are at the same hierarchical level. Elements assigned larger numbers are at lower levels and are actually part of the immediate preceding element assigned a smaller number. For example, in Figure 2-6 PART-NO-SOLD and LAST-SALE are at the same level; MONTH-SOLD is part of LAST-SALE, and LAST-SALE is part of PART-SALES-REC.

SUBSCRIPTS indicates the number of subscripts required for an element. If an <item name> or <group name> requires no subscripts, the entry in this column is blank.

LENGTH describes the number of characters required on the listing to print the item.

SAMPLE VOCABULARY 1: "VOCEMP"

Figure 2-4 lists the vocabulary for an employee file which is maintained by the Personnel Department. There is one record in the file for each employee. The entries in each record used are as follows:

Name	Description
EMPYE-NO	Employee number
EMPYE-NAME	Employee name
JOB-GRADE	Code for employee's job level
DEPT-NO	Department number
CHG-CODE	Charge code
EMPLOYMENT-YR	Year of employment
EMPYE-AGE	Age of employee
SEX	Sex of employee
SALARY	Salary of employee
FEDERAL-TAX	Federal income tax withheld
STATE-TAX	State income tax withheld
SOC-SEC-TAX	Social Security tax withheld
EMPYE-NET-PAY	Employee's net pay

```
RP3 N.NN                    V O C A B U L A R Y                    DD MMM YY
                                 "VOCEMP"                          PAGE    1

SYSTEM FILES.

FILE EMPYE-FILE,
     ORGANIZATION IS SEQUENTIAL,
     TOTAL-POPULATION IS DEFAULTED TO 9999.

       NAME WITH                                                   EDITING
LEVEL  QUALIFIERS          SUBSCRIPTS  TYPE            LENGTH      PICTURE

  1    EMPYE-RECORD                   GROUP
  2    EMPYE-NO                       ITEM NUMERIC         6      9(6)
  2    EMPYE-NAME                     ITEM STRING         10      X(30)
  2    EDUCATION-LEVEL                ITEM NUMERIC         2      99
  2    JOB-GRADE                      ITEM NUMERIC         2      99
  2    DEPT-NO                        ITEM NUMERIC         4      9(4)
  2    CHG-CODE                       ITEM NUMERIC         6      9(6)
  2    EMPLOYMENT-YR                  ITEM NUMERIC         2      99
  2    EMPYE-AGE                      ITEM NUMERIC         2      99
  2    SEX                            ITEM NUMERIC         6      9
       MALE                           CONDITION
       FEMALE                         CONDITION
  2    SALARY                         ITEM NUMERIC        10      $Z(6).99
  2    FEDERAL-TAX                    ITEM NUMERIC        10      $Z(6).99
  2    STATE-TAX                      ITEM NUMERIC        10      $Z(6).99
  2    SOC-SEC-TAX                    ITEM NUMERIC        10      $Z(6).99
  2    EMPYE-NET-PAY                  ITEM NUMERIC        10      $Z(6).99
```

Figure 2-4. Vocabulary Listing for "VOCEMP"

SAMPLE VOCABULARY 2: "VOCAST"

Figure 2-5 lists the vocabulary for a fixed-asset master file which would be available at the end of each year and as of the current balance sheet date. There is one record in the file for each fixed asset. The entries in each record are as follows:

Name	Description
ASSET-NO	Asset number
ASSET-DESC	Asset description
ASSET-TYPE	Asset type code
DEPT-NO	Associated department number
LOC-CODE	Location code
ACQUISITION-YR	Year of acquisition
ASSET-LIFE	Asset useful life in years
TDM	Tax depreciation method
COST	Original asset cost
ACCUM-DEPRE-BOOK	Accumulated depreciation, beginning book
DEPRE-YTD-BOOK	Depreciation, year-to-date book
ACCUM-DEPRE-TAX	Accumulated depreciation, beginning tax

RP3 N.NN V O C A B U L A R Y DD MMM YY
"VOCAST" PAGE 1

SYSTEM FILES.

FILE ASSET-FILE,
ORGANIZATION IS SEQUENTIAL,
TOTAL-POPULATION IS DEFAULTED TO 9999.

LEVEL	NAME WITH QUALIFIERS	SUBSCRIPTS	TYPE	LENGTH	EDITING PICTURE
1	ASSET-RECORD		GROUP		
2	ASSET-NO		ITEM NUMERIC	6	9(6)
2	ASSET-DESC		ITEM STRING	30	X(30)
2	ASSET-TYPE		ITEM NUMERIC	2	99
2	DEPT-NO		ITEM NUMERIC	4	9(4)
2	LOC-CODE		ITEM NUMERIC	6	9(6)
2	ACQUISITION-YR		ITEM NUMERIC	2	99
2	ASSET-LIFE		ITEM NUMERIC	2	99
2	TDM		ITEM NUMERIC	15	9
	STRAIGHT-LINE		CONDITION		
	DOUBLE-DECL-BAL		CONDITION		
2	COST		ITEM NUMERIC	10	$Z(6).99
2	ACCUM-DEPRE-BOOK		ITEM NUMERIC	10	$Z(6).99
2	DEPRE-YTD-BOOK		ITEM NUMERIC	10	$Z(6).99
2	ACCUM-DEPRE-TAX		ITEM NUMERIC	10	$Z(6).99
2	DEPRE-YTD-TAX		ITEM NUMERIC	10	$Z(6).99

Figure 2-5. Vocabulary Listing for "VOCAST"

SAMPLE VOCABULARY 3: "INVENT"

Figure 2-6 lists the vocabulary for a client's inventory file. Each record in the inventory file corresponds to a part.

The entries for each record are as follows:

Name	Description
PART-NO	Part number
DEPARTMENT	Manufacturing department number
PART-DESC	Part description
MATERIAL-SOURCE-CODE	Material source code
QUANTITY	Inventory quantity
UNIT-COST	Cost per part
BIN-NO	Bin number

The file of part sales for the year is related to the inventory file, and is also contained in the vocabulary. Each of these records contains cumulative sales information on a particular part. The part number relates the inventory record for a particular part to the sales record for that part.

The entries in each sales record are as follows:

Name	Description
PART-NO-SOLD	Part number
DEPARTMENT-NO	Manufacturing department number
QUANTITY-SOLD	Total quantity sold
LAST-SALE	Date of last sale
MONTH-SOLD	Month of last sale
DAY-SOLD	Day of last sale
YR-SOLD	Year of last sale

```
RP3 N.NN                V O C A B U L A R Y                DD MMM YY
                              "INVENT"                     PAGE    1

PERMANENT MACROS (REPLACE STATEMENTS).

MACRO PART-INVENT-SALES-INFO.
        TEXT:  PARTMF, YTDSMF AT PART-NO #
```

```
RP3 N.NN                V O C A B U L A R Y                DD MMM YY
                              "INVENT"                     PAGE    2

SYSTEM FILES.

FILE PARTMF,
     ORGANIZATION IS INDEXED,
     RECORD KEY IS PART-NO,
     TOTAL-POPULATION IS 1000000.

                                SUB-                                  EDITING
LEVEL  NAME WITH QUALIFIERS     SCRIPTS  TYPE          LENGTH         PICTURE

  1    PART                             GROUP
  2    PART-NO                          ITEM STRING        6        X(6)
  2    DEPARTMENT                       ITEM NUMERIC       4        9(4)
  2    PART-DESC                        ITEM STRING       20        X(20)
  2    MATERIAL-SOURCE-CODE             ITEM STRING        1        X(1)
  2    QUANTITY                         ITEM NUMERIC       5        9(5)
  2    UNIT-COST                        ITEM NUMERIC       6        Z(2)9.99
  2    INVENT-DOLLAR-VALUE              ITEM NUMERIC      11        Z(7)9.99
  2    BIN-NO                           ITEM NUMERIC       3        9(3)

FILE YTDSMF,
     ORGANIZATION IS INDEXED,
     RECORD KEY IS PART-NO-SOLD,
     TOTAL-POPULATION IS 1000000.

                                SUB-                                  EDITING
LEVEL  NAME WITH QUALIFIERS     SCRIPTS  TYPE          LENGTH         PICTURE

  1    PART-SALES-REC                   GROUP
  2    PART-NO-SOLD                     ITEM STRING        6        X(6)
  2    DEPARTMENT-NO                    ITEM NUMERIC       4        9(4)
  2    QUANTITY-SOLD                    ITEM NUMERIC       5        9(5)
  2    LAST-SALE                        GROUP
  3    MONTH-SOLD                       ITEM NUMERIC       2        99
  3    DAY-SOLD                         ITEM NUMERIC       2        99
  3    YEAR-SOLD                        ITEM NUMERIC       2        99
```

Figure 2-6. Vocabulary Listing "INVENT"

SAMPLE VOCABULARY 4: "CLIENT"

Figure 2-7 lists the vocabulary for three DMS II data sets: an accounts receivable data set, a name and address data set, and an invoice data set. (A data set is a DMS II data structure which can be considered a file.) Each record in the accounts receivable data set corresponds to a single customer account and contains the following entries:

Name	Description
BRANCH	Branch number
CUST-NO	Customer number
CREDIT-LIMIT	Customer credit limit in dollars
SALES-PERSON-CD	Salesperson code
DISCOUNT-PERCENT	Cash discount percentage
BILLING-TERMS-CODE	Terms of billing code
DATE-LAST-PAYMT	Date of last payment
DATE-LAST-PURCH	Date of last purchase
BALANCE-DUE	Total balance due
CURRENTLY-DUE	Amount currently due
THIRTY-DAYS-DUE	Amount 30 days due
SIXTY-DAYS-DUE	Amount 60 days due
NINETY-DAYS-DUE	Amount 90 days due

Each record in the name and address data set contains information on a particular customer invoice and contains the following entries:

Name	Description
CUSTOMER-NUMBER	Customer number
BRANCH-NUMBER	Branch number
CUSTOMER-NAME	Customer's full name
STREET-ADDRESS	Street address
CITY-STATE	City and state
ZIP-CODE	Zip code
RISK-RATING	Risk rating code

Each record in the invoice data set deals with a particular customer invoice and contains the following entries:

Name	Description
INVOICE-NO	Invoice number
INV-CUST-NO	Customer number
DUE-DATE	Due date
TERMS-BILLING-CODE	Terms of billing code
TOTAL-AMOUNT	Total amount of invoice
AMOUNT-PAID	Total amount paid
CASH-DISCOUNT	Cash discount taken
DATE-LAST-PAYMT	Date of last payment
CHECK-NO	Check number
INV-BRANCH-NO	Branch number

Each accounts receivable record is associated with one or more invoices via the DMS II-maintained set INVOICES. Related customer name and address information can be obtained for each accounts receivable record or invoice record via appropriate lookup in the DMS II-maintained set NUMBER-SET.

```
RP3 N.NN                    V O C A B U L A R Y                    DD MMM YY
                                  "CLIENT"                         PAGE    2

DMSII DATA BASE CUST-ACCT-INFO.

DATA SET ACCTS-RECV,
     TOTAL-POPULATION IS 200.

     SET ACCTS-SET,
         SPANS DATA SET ACCTS-RECV,
         KEY IS CUST-NO.

     SET INVOICES,
         SPANS DATA SET INVOICE-INFO,
         KEY IS INVOICE-NO.

                                  SUB-                                 EDITING
LEVEL  NAME WITH QUALIFIERS       SCRIPTS  TYPE            LENGTH      PICTURE

  2    BRANCH                             ITEM NUMERIC       4        9(4)
  2    CUST-NO                            ITEM NUMERIC       6        9(6)
  2    CREDIT-LIMIT                       ITEM NUMERIC       5        Z(4)9
  2    SALES-PERSON-CD                    ITEM NUMERIC       5        9(5)
  2    DISCOUNT-PERCENT                   ITEM NUMERIC       2        99
  2    BILLING-TERMS-CODE                 ITEM NUMERIC       1        9
  2    DATE-LAST-PAYMNT                   ITEM NUMERIC       6        9(6)
  2    DATE-LAST-PURCH                    ITEM NUMERIC       6        9(6)
  2    BALANCE-DUE                        ITEM NUMERIC       9        $Z(4)9.99
  2    CURRENTLY-DUE                      ITEM NUMERIC       9        $Z(4)9.99
  2    THIRTY-DAYS-DUE                    ITEM NUMERIC       9        $Z(4)9.99
  2    SIXTY-DAYS-DUE                     ITEM NUMERIC       9        $Z(4)9.99
  2    NINETY-DAYS-DUE                    ITEM NUMERIC       9        $Z(4)9.99
```

Figure 2-7. Vocabulary Listing for "CLIENT"
(Sheet 1 of 2)

```
RP3 N.NN                    V O C A B U L A R Y                    DD MMM YY
                                 "CLIENT"                          PAGE    3

DMSII DATA BASE CUST-ACCT-INFO (CONTINUED)

DATA SET CUST-INFO,
     TOTAL-POPULATION IS 200.

     SET NAME-SET,
          SPANS DATA SET CUST-INFO,
          KEY IS CUSTOMER-NUMBER.

     SET NUMBER-SET,
          SPANS DATA SET CUST-INFO,
          KEY IS CUSTOMER-NUMBER.

                                SUB-                                    EDITING
LEVEL   NAME WITH QUALIFIERS    SCRIPTS  TYPE            LENGTH         PICTURE

  2     CUSTOMER-NUMBER                  ITEM NUMERIC      6         9(6)
  2     BRANCH-NUMBER                    ITEM NUMERIC      4         9(4)
  2     CUSTOMER-NAME                    ITEM STRING      20         X(20)
  2     STREET-ADDRESS                   ITEM STRING      20         X(20)
  2     CITY-STATE                       ITEM STRING      20         X(20)
  2     ZIP-CODE                         ITEM NUMERIC      5         9(5)
  2     RISK-RATING                      ITEM NUMERIC      1         9

DATA SET INVOICE-INFO,
     TOTAL-POPULATION IS 5000.

     SET INVOICE-NOS,
          SPANS DATA SET INVOICE-INFO,
          KEY IS INVOICE-NO.

                                SUB-                                    EDITING
LEVEL   NAME WITH QUALIFIERS    SCRIPTS  TYPE            LENGTH         PICTURE

  2     INVOICE-NO                       ITEM NUMERIC      6         9(6)
  2     INV-CUST-NO                      ITEM NUMERIC      6         9(6)
  2     DUE-DATE                         ITEM NUMERIC      6         9(6)
  2     TERMS-BILLING-CODE               ITEM NUMERIC      1         9
  2     TOTAL-AMOUNT                     ITEM NUMERIC      8         $Z(3)9.99
  2     AMOUNT-PAID                      ITEM NUMERIC      8         $Z(3)9.99
  2     CASH-DISCOUNT                    ITEM NUMERIC      8         $Z(3)9.99
  2     DATE-LAST-PAYMT                  ITEM NUMERIC      6         9(6)
  2     CHECK-NO                         ITEM NUMERIC      8         9(8)
  2     INV-BRANCH-NO                    ITEM NUMERIC      4         9(4)
```

Figure 2-7. Vocabulary Listing for "CLIENT"
(Sheet 2 of 2)

SAMPLE VOCABULARY 5: "CUSTV"

Figure 2-8 lists the partial vocabulary for a magnetic tape file that contains customer remittances received during the months of January and February. This file is maintained in sequence by account number. Each record represents a single remittance and contains the following entries:

Name	Description
REMIT-ACCT-NO	Account number
CUST-NAME	Customer name
DATE-RECD	Date received
YR-MO-RECD	Year and month received
DAY-RECD	Day received
NET-PAYMT	Net payment
DISCNT-TAKEN	Discount taken
CHK-NO	Check number

The file of remittances is used to update an open accounts receivable file which is also maintained in ascending account-number order. Each record of the accounts receivable file contains the following entries as well as other account information:

Name	Description
ACCT-NO	Account number
BALANCE	Account balance
OLD-BALANCE	Previous balance at month end

```
RP3 N.NN                    V O C A B U L A R Y                    DD MMM YY
                                  "CUSTV"                          PAGE    1

SYSTEM FILES.

FILE REMIT-FILE,
     ORGANIZATION IS SEQUENTIAL,
     TOTAL-POPULATION IS 1000.

      NAME WITH                                                    EDITING
LEVEL QUALIFIERS        SUBSCRIPTS  TYPE            LENGTH         PICTURE

  1   REMIT-RECORD                  GROUP
  2   REMIT-ACCT-NO                 ITEM NUMERIC       3         9(3)
  2   CUST-NAME                     ITEM STRING       20         X(20)
  2   DATE-RECD                     GROUP
  3   YR-MO-RECD                    ITEM NUMERIC       4         9(4)
  3   DAY-RECD                      ITEM NUMERIC       2         9(2)
  2   NET-PAYMNT                    ITEM NUMERIC      10         $Z(5)9.99
  2   DISCNT-TAKEN                  ITEM NUMERIC       8         $Z(3)9.99
  2   CHK-NO                        ITEM NUMERIC       8         9(8)


FILE ACCT-FILE,
     ORGANIZATION IS SEQUENTIAL,
     TOTAL-POPULATION IS 200.

      NAME WITH                                                    EDITING
LEVEL QUALIFIERS        SUBSCRIPTS  TYPE            LENGTH         PICTURE

  1   ACCT-RECORD                   GROUP
  2   ACCT-NO                       ITEM NUMERIC       3         9(3)
  2   BALANCE                       ITEM NUMERIC      12         $Z(7)9.99
  2   OLD-BALANCE                   ITEM NUMERIC      12         $Z(7)9.99
```

Figure 2-8. Vocabulary Listing for "CUSTV"

SAMPLE VOCABULARY 6: "SHIPV"

Figure 2-9 lists the vocabulary for a detail transaction file of accounts receivable and accounts payable for a shipping company. Each record of the file represents a transaction and contains the following entries:

Name	Description
NUMBER	Freight bill number
DATE-OF-BILLING	Date of billing
MONTH-OF-BILLING	Month of billing
DAY-OF-BILLING	Day of billing
YEAR-OF-BILLING	Year of billing
POINTS-OF-SHIPMENT	Codes for shipment origination points
POINTS-OF-DESTINATION	Codes for shipment destination points
RECEIPT-NO	Receipt number or disbursement check number
ACCOUNT-NO	Account number
TRANSACTION-TYPE	Transaction type (1=receivable; 0=payable)
TRANSACTION-CODE	Transaction code
AMOUNT-OWED	Dollar amount owed

```
RP3 N.NN                    V O C A B U L A R Y                    DD MMM YY
                                 "SHIPV"                           PAGE    1

SYSTEM FILES.

FILE ACCT-TRANS,
      ORGANIZATION IS INDEXED,
      RECORD KEY IS NUMBER,
      TOTAL-POPULATION IS DEFAULTED TO 9999.
```

LEVEL	NAME WITH QUALIFIERS	SUB-SCRIPTS	TYPE	LENGTH	EDITING PICTURE
1	BILL		GROUP		
2	NUMBER OF BILL		ITEM STRING	8	X(8)
2	DATE-OF-BILLING		GROUP		
3	MONTH-OF-BILLING		ITEM NUMERIC	2	99
3	DAY-OF-BILLING		ITEM NUMERIC	2	99
3	YEAR-OF-BILLING		ITEM NUMERIC	2	99
2	POINTS OF SHIPMENT SUB 1: MAX VALUE IS 5	1	ITEM NUMERIC	8	9(8)
2	POINTS-OF-DESTINATION		ITEM NUMERIC	8	9(8)
2	RECEIPT-NO		ITEM NUMERIC	6	9(6)
2	ACCOUNT-NO OF BILL		ITEM NUMERIC	4	9(4)
2	TRANSACTION-TYPE		ITEM NUMERIC	1	9
2	TRANSACTION-CODE		ITEM NUMERIC	1	9
2	AMOUNT-OWED		ITEM NUMERIC	10	-$(5).99

Figure 2-9. Vocabulary Listing for "SHIPV"

SECTION 3

DESIGNING REPORTS

This section is intended to help you determine the REPORTER III report language statements you need, as well as the logical order of the statements, to design and specify report(s). Brief explanations of "basic language constructs" and "functions" often used in constructing report language statements are included in this section. Detailed explanations of all report language statements, clauses, basic language constructs, and functions are provided in Section 4.

REPORT SPECIFICATION

A REPORTER III report specification describes one or more reports. A report is defined in general to mean one of the following:

1. An automatically formatted printed report.
2. A user-formatted printed report.
3. A machine-readable extract file.

The syntax diagram presented in Figure 3-1 shows the general processes involved in use of REPORTER III report language statements to design and specify either a single report or multiple reports.

In designing report(s), you use report language statements to define what information is to be reported and what types of reports are to be produced.

First you indicate the password, if any is needed to reference the required vocabulary. Next you reference the vocabulary (created by the RP3VOC program) which contains the description of the file(s) to be used in specifying the desired report(s).

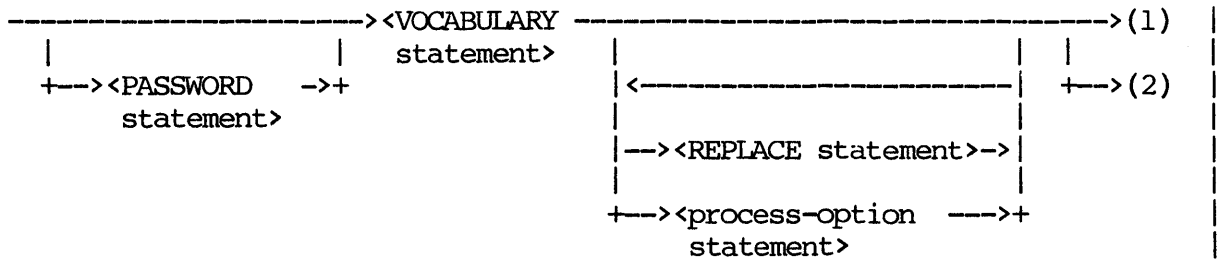


Figure 3-1. General Syntax Diagram for REPORTER III Report Specification (Sheet 1 of 4)

For Single Report:

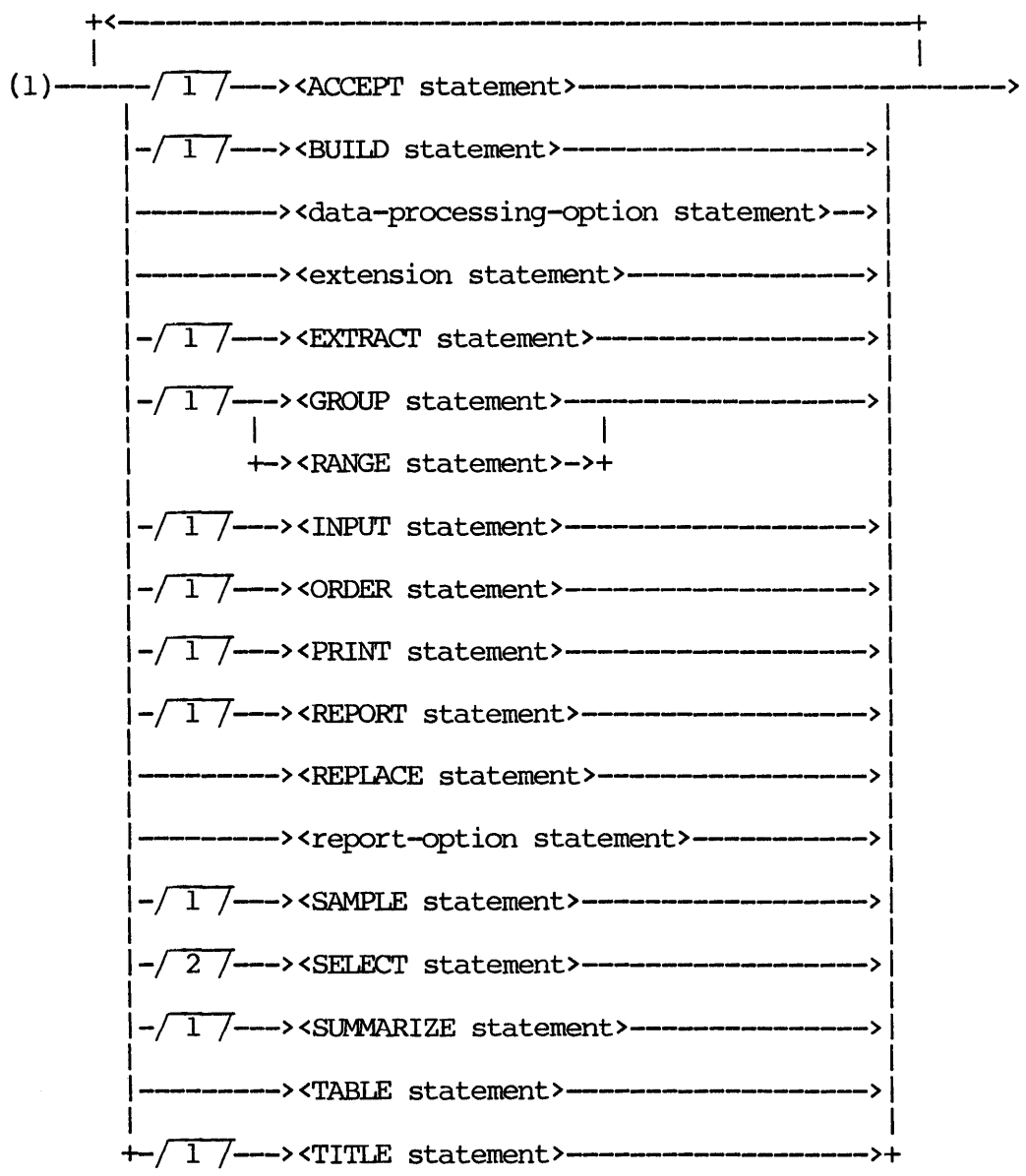


Figure 3-1. General Syntax Diagram for REPORTER III Report Specification (Sheet 2 of 4)

For Multiple Reports:

Input Section:

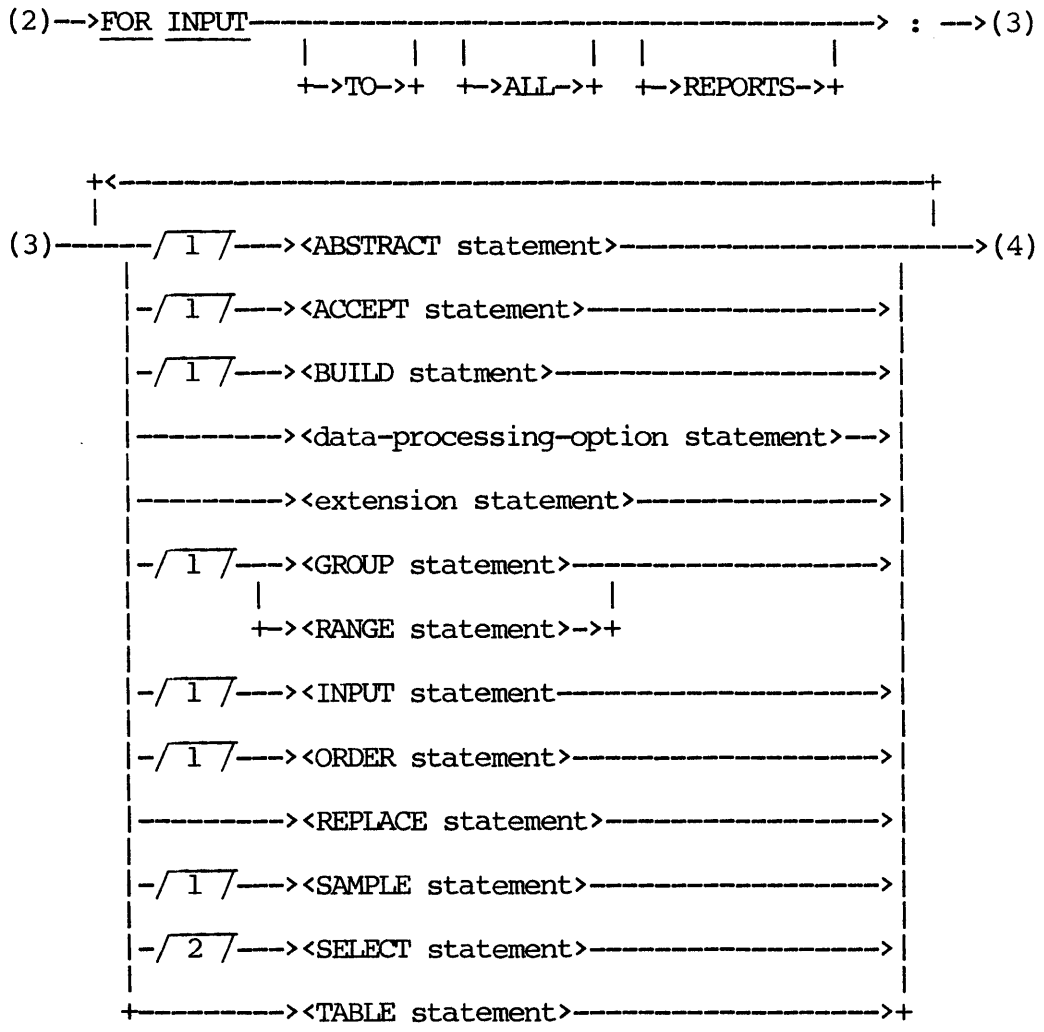


Figure 3-1. General Syntax Diagram for REPORTER III Report Specification (Sheet 3 of 4)

Report Section(s):

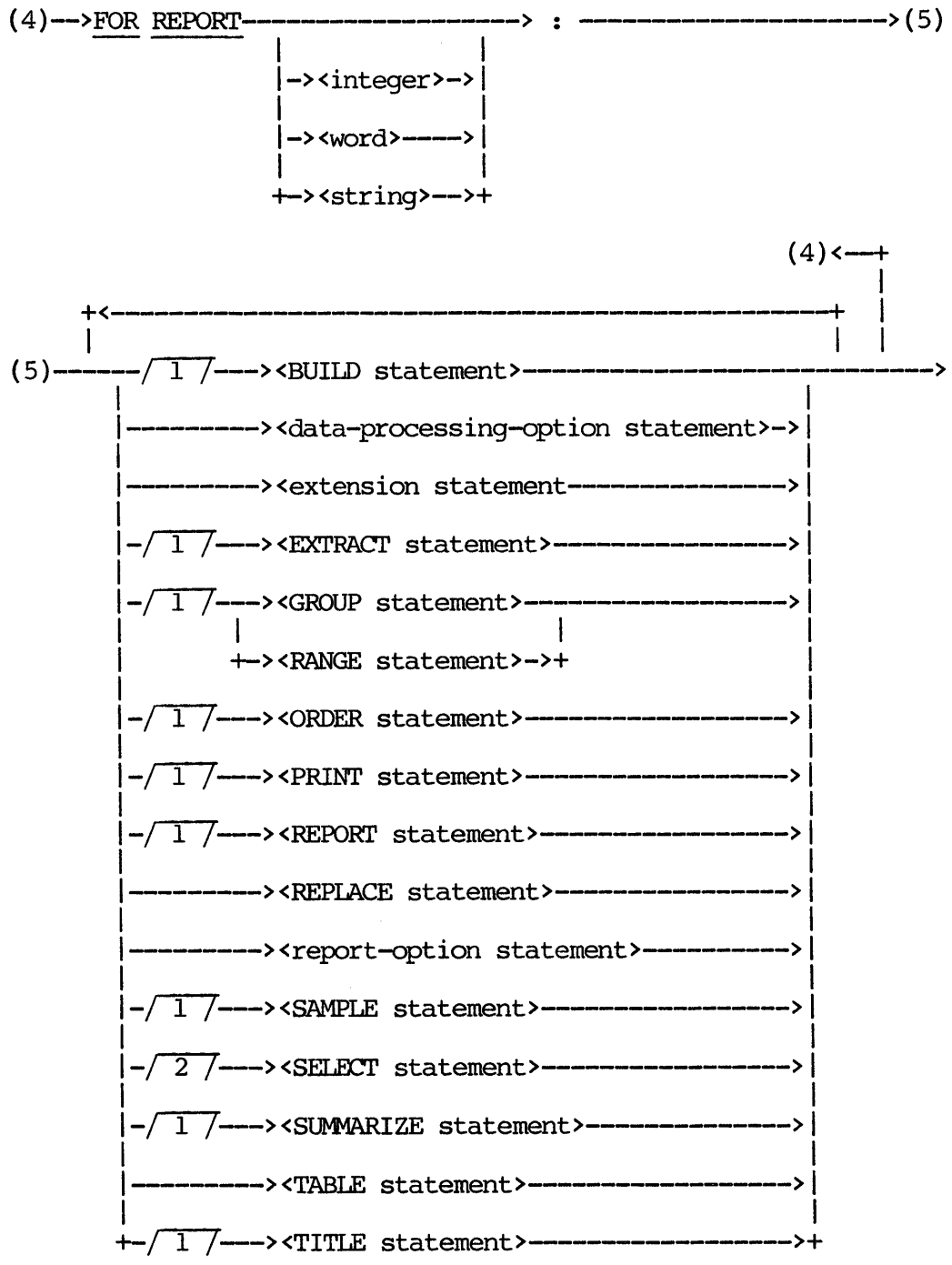


Figure 3-1. General Syntax Diagram for REPORTER III Report Specification (Sheet 4 of 4)

After you reference the vocabulary and before you enter your report specification, you have the option to enter the following:

1. Process-option statement(s), which allow you to change system default values and/or actions for the report(s).

2. REPLACE statement(s), which allow you to provide instructions to RP3REP to replace specified names by corresponding designated portions of text in all places where those names occur.

Then you enter the report specification, (using language statements such as INPUT, EXTENSION, SELECT, SAMPLE, GROUP, ORDER, SUMMARIZE, PRINT, EXTRACT) which allow you to manipulate the information and to generate the desired report(s).

When you have finished specifying the report(s), you input your report specification to the Report Language Analysis Program (RP3REP). (The vocabulary which you have referenced must be on disk or disk pack when the RP3REP program is run.) If you have not made any syntax errors in designing your report(s), a source program based on the parameters produced by RP3REP is generated by the RP3GEN program, the source program is compiled, and then the object code is executed to produce the desired report(s). Otherwise, the REPORTER III System displays messages indicating the errors you must correct before processing can continue.

SINGLE-REPORT SPECIFICATION

When you specify a single report, one pass is made through the data, and one report is produced.

In designing a single report, you use those language statements necessary to produce the desired report (see portion of Figure 3-1 under the heading "For Single Report"). A single-report specification cannot consist of more than one sort; that is, logical records which are to be reported are grouped (via control-break items) and/or ordered in only one sequence.

MULTIPLE-REPORT SPECIFICATION

When you specify multiple reports, one pass is made through the data, and one or a number of reports are produced.

First you select the logical record(s)(through the INPUT statement) and/or design other processing (through other language statements) that will be common to all the reports to be created; this constitutes the Input Section specification (see Input Section portion of Figure 3-1). This information is to be input to each of the Report Sections. The Input Section specification cannot consist of more than one sort; that is, logical records which are to be input are grouped (via control-break items) and/or ordered in only one sequence.

Then you design each report individually, using those language statements necessary to produce the desired report (see Report Section portion of Figure 3-1). As shown in the Report Section portion of the figure, all language statements pertaining to a particular report must be specified before the statements pertaining to another report are specified. Language statements contained in a particular Report Section have no relation to any other Report Sections. Each Report Section describes one report to be produced from the information [logical record(s)] defined by the INPUT statement in the Input Section.

All names and identifiers defined in a Report Section [e.g., macro names, derived data item names (known as "extensions"), and table identifiers] can only be referenced in the statements within that section. These names and identifiers can be identical to those defined in other Report Sections; no conflict arises since the names and identifiers defined in a Report Section are local to that section.

The specification for each Report Section cannot consist of more than one sort; that is, logical records which are to be reported in a particular Report Section are grouped (via control-break items) and/or ordered in only one sequence.

For the purpose of identification, each report designed can be referenced by report number (integer, word, or string).

Example:

```
        .
        .
        .
    FOR REPORT 1:
        .
        .
        .
    FOR REPORT TWO:
```

.
.
.
FOR REPORT "ON SALARY TOTALS":
.
.
.

It is noted that a single-report specification which contains an INPUT statement is identical to a multiple-report specification in which the Input Section contains only an INPUT statement and possibly an ACCEPT statement, and the Report Section contains all the other statements of that particular single-report specification.

SPECIFICATION CONSTRAINTS

A number of constraints with respect to report language statement specification are noted below.

ORDER OF LANGUAGE STATEMENTS

In specifying the language statements needed either in a single-report specification or in the Input Section of a multiple-report specification, you can give the statements in any order you desire, provided you meet the following constraints:

1. You must define an input data structure in the INPUT statement before you can reference data items in that structure.

2. You must define a name before you can reference it. This applies to a <macro name>, a derived data item name, an accepted data item name, a control-break name, and a <column> name.

3. You must define a table before you can reference it.

In specifying the language statements needed in the Report Section of a multiple-report specification, you are constrained by items 2 and 3 above (item 1 does not apply).

CONTROL-BREAK ITEMS AND ORDERING KEYS

Control-break items and ordering keys specified in the Input Section are applicable only to the specifications within the Input Section. If these same control-break items and ordering keys are required within a Report Section, they must be redefined as such within the particular Report Section, and the SUPPRESS SORT feature may be used (refer to the SUPPRESS-SORT statement in Section 4).

Control-break items specified in a Report Section are applicable only to the report described by that section.

ABSTRACT STATEMENT

An ABSTRACT statement can be used only in the Input Section of a multiple-report specification (even if only one report is to be produced).

LANGUAGE STATEMENTS

The language statements which appear in the general syntax diagram (Figure 3-1) are listed alphabetically and explained briefly in the following table. All these statements, as well as all clauses which can be contained within them, are explained individually and in detail in Section 4.

You can use the following table as a guideline in determining language statements which you need to design your report(s). You then can consult the detailed discussions of the language statements and their components in Section 4 to determine the specific construction of each statement.

Language Statement	Explanation
ABSTRACT statement	Specifies that only summary information for a previously defined referenced data item is to be input, as a single logical record, to every report. The referenced item is declared a control-break item if control breaks have not been specified previously in another statement.

ACCEPT statement

Specifies data items which are to be parameters to the generated report program. The specified values of the data items will be entered at run time and will remain constant throughout the production of one report (or one set of reports, in the case of a multiple-report specification). The ACCEPT statement can be used to produce different versions of a report from one run of a report program.

BUILD statement

Provides the means to define new data items by isolating portions of data names or joining data names. These new data items subsequently can be referenced where appropriate in the report specifications.

Data-processing-option statement

Provides the means to override default values and actions with nondefault values and actions related to the grouping and arrangement of records in the generated report program. There are three types of data-processing-option statements, each of which is explained in detail in Section 4:

1. SUPPRESS SORT statement.
2. SET SORT BLOCKING statement.
3. SET SORT SIZE statement.

For a brief explanation of each type, refer to the DATA-PROCESSING-OPTION statement in Section 4.

Extension statement

Defines a new data item to be derived from already-existing data items. Once defined, the new data item can be subsequently referenced where appropriate in the report specifications.

EXTRACT statement

Extracts information to a new file in machine-readable format. This extracted information can then be printed or reported through another report created by the REPORTER III System, or it can be made available for other processing as needed.

GROUP statement — Specifies control-break items which are used to group the information being reported and provide a basis for summarization. Control breaks for the report, or section, must not have been specified previously in another statement.

INPUT statement — Describes what DMS II data base and/or system files are to be used and the method of access. Defines the content of a logical record.

ORDER statement — Specifies how information is to be arranged (ordered). The ordering is subordinate to any control-break grouping specified elsewhere, and it must be consistent with any ordering described elsewhere.

PASSWORD statement — Supplies the password which enables the specified vocabulary to be referenced. This statement is required only if a password was established for the vocabulary in the RP3VOC specifications. If the statement is required and the password is not entered correctly, the Report Language Analysis Program (RP3REP) will terminate with an appropriate error message when it is run.

PRINT statement — Specifies the layout and content of a user-formatted report. This statement is useful for printing special forms, such as confirmation letters and mailing labels, and for printing other reports where you need the data printed at an exact location on the page.

Process-option statement — Enables you to change the default operation of the REPORTER III System, (to change a default processing option). There are five types of process-option statements, each of which is explained in detail in Section 4:

1. COMBINE statement.
2. Process-option ASSIGN statement.

3. Process-option SAVE statement.

4. Process-option SET statement.

5. Process-option SUPPRESS statement.

For a brief explanation of each type, refer to PROCESS-OPTION STATEMENT in Section 4.

RANGE statement Defines a range-break item as the one and only control-break item. Specifies that information be grouped according to given value ranges of a given data item, to provide a basis for reporting and summarization.

REPLACE statement Enables a specified name to be replaced by a portion of language text, consisting of characters, words, and/or phrases, in all places where the name occurs. Provides a convenient shorthand technique for expanding a REPORTER III report language statement.

Report-option statement Enables you to override a default value or action with a specified value or action for the particular report. There are four types of report-option statements, each of which is explained in detail in Section 4:

1. ASSIGN LISTING statement.
2. Report-option SET statement.
3. Report-option SUPPRESS statement.
4. SAVE LISTING statement.

For a brief explanation of each type, refer to REPORT-OPTION STATEMENT in Section 4.

REPORT statement Specifies the layout of an automatically formatted report and the data items to be included in the report. This statement provides control-break headings and column listings, if desired.

SAMPLE statement	Causes only certain selected logical records ("samples") to be made available for reporting. The statement can be designed to make the selection either systematic or random, depending upon the needs for the report. Also, the sampling can be stratified by using a strata-defined Boolean expression in the sample description.
SELECT statement	Causes input logical records either to be made available for reporting or to be suppressed. If no SELECT statement and no SAMPLE statement are used, all logical records that were input are available for reporting (no information is suppressed).
SUMMARIZE statement	Specifies what statistical summaries and column footings are to be included at the end of control-break groupings and/or at the end of the report.
TABLE statement	Defines a conversion table which relates data values to equivalent or corresponding values. Once the table is defined, you can use an ENTRY function to obtain the corresponding table-defined values for particular values of a data item.
TITLE statement	Specifies page title information for the report. Titles can consist of character strings, time of report, date of report, and/or data items.
VOCABULARY statement	Identifies the vocabulary files to be used for the desired report. This statement is always required in a REPORTER III report specification. The vocabulary files must be created by the RP3VOC Program.

BASIC LANGUAGE CONSTRUCTS

In most REPORTER III report language statements, you use basic language constructs in defining information within the statements. The constructs are listed alphabetically and explained briefly in the following table. All these constructs are explained individually and in detail in Section 4.

Language Construct	Explanation
<COBOL picture>	Optionally used to describe how an item is to be printed.
<data name>	References an item or group. It is a <name> or a <name> qualified by other <name>s. It possibly can be subscripted by <integer>s and/or other <data name>s. It possibly can be associated with a designated control-break item.
<expression>	Describes one of the following in terms of operands and operators: 1. An arithmetic expression, which specifies a numeric value. 2. A string expression, which specifies a string value. 3. A Boolean expression, which specifies a Boolean value (TRUE or FALSE) or a numeric value (1 for TRUE and 0 for FALSE). Can be statistical or nonstatistical.
<external file name>	Identifies a file to the MCP (Master Control Program). Contains <identifier>s, each being a <string> enclosed in quotation marks.
<literal>	Represents a data item which has a value identical to the value being described. There are two classes of literals:

1. Numeric literals, or <number>s.

2. Nonnumeric literals, or <string>s.

<relational operator> Specifies the criteria for comparison (for example, IS NOT EQUAL T OR GREATER THAN).

FUNCTIONS

In many REPORTER III report language statements, you use a function(s) to define the value of a data item which is based on the value(s) of other data item(s). The functions (and the component clauses <date format> and <stat parameters>) are listed alphabetically and explained briefly in the following table. All these functions (and the component clauses) are explained individually and in detail in Section 4.

Function	Explanation
<AGE function>	A nonstatistical function used to calculate the time interval (the number of boundaries crossed) between two dates, in terms of DAYS, WEEKS, MONTHS, QUARTERS, or YEARS.
<date-convert function>	A nonstatistical function used to convert a data item stored within a particular <date format> to a data item representing the same date, but in a different format.
<date format>	A clause which specifies how a particular date is coded so that it can be accessed properly by the REPORTER III System.
<ENTRY function>	A nonstatistical function used to convert values of a data item to alternate item values according to a table previously defined by a TABLE statement.

<statistical function>	Describes a statistical item which summarizes a group of logical records which were input and possibly extended, selected from input, sampled, and/or selected from the sample. A statistic can be specified as a count, running count, total, running total, average, maximum, minimum, sum of squares, mean square, variance, or standard deviation.
<stat parameters>	Specifies the parameters of the statistical function.

EXAMPLE REPORT SPECIFICATIONS

An example of a single-report specification and an example of a multiple-report specification are presented below. Each example is accompanied by an explanation of the specification and an illustration of the report produced. The vocabulary used in each example is described in Section 2. Other examples of report specifications are presented in Section 5.

EXAMPLE SINGLE-REPORT SPECIFICATION

A printed report listing all assets acquired during the year 1985 with a cost in excess of $5,000 is required for vouching and inspection. The following REPORTER III report specification, using the vocabulary "VOCAST", is constructed to reflect this information:

```
VOCABULARY IS "VOCAST".
INPUT ASSET-FILE.
SELECT ACQUISITION-YR = 85 AND COST > 5000.00.
REPORT ASSET-NO, ASSET-DESC, ASSET-TYPE, DEPT-NO,
   ASSET-LIFE, COST.
```

In this specification, first the vocabulary "VOCAST" is referenced. Then the following is done:

1. The logical records in the data base corresponding to ASSET-FILE are specified as input required for the report specification.

2. Then all logical records corresponding to both ACQUISITION-YR = 85 (acquisition year is 1985) and COST > 5000.00 (cost is greater than $5,000) from the input physical record ASSET-FILE are specified to be made available for reporting (selected).

3. Finally, the data items ASSET-NO (asset number), ASSET-DESC (asset description), ASSET-TYPE (asset type code), DEPT-NO (associated department number), ASSET-LIFE (asset useful life in years), and COST (original asset cost) from the logical records which meet the select criteria are specified to be reported (printed) in an automatically formatted report.

Processing the above specification produces the required report, which is shown in Figure 3-2. Note that the column headings are the data names.

PAGE 1

ASSET NO	ASSET DESC	ASSET TYPE	DEPT NO	ASSET LIFE		COST
007130	V2000 TERMINAL	03	1122	05	$	5205.00
009200	1982 FORD VAN	01	1501	10	$	5102.80
009801	TEMPORARY STORAGE BUILDING	03	1203	20	$	6179.25

Figure 3-2. Example Single Report

EXAMPLE MULTIPLE-REPORT SPECIFICATION

Two printed reports are required, each giving the total of the remittances received each day during January 1985 and the grand total for the month. One of the reports lists detailed information about each remittance received, while the other report lists only the required summary information. The following REPORTER III report specification, using the vocabulary "CUSTV", is constructed to reflect this information:

```
VOCABULARY IS "CUSTV".
FOR INPUT TO ALL REPORTS:
   INPUT REMIT-FILE.
   SELECT YR-MO-RECD = 8501
   GROUP BY DAY-RECD.  % THIS STATEMENT SAVES AN
                       % ADDITIONAL SORT
FOR REPORT 1:
   TITLE "JANUARY REMITTANCES".
   REPORT BY DAY-RECD LISTING REMIT-ACCT-NO,CUST-NAME,
      NET-PAYMNT, DISCNT-TAKEN, CHK-NO.
   SUMMARIZE FOOTING NET-PAYMNT.
   SUPPRESS SORT.  % INPUT INFO ALREADY IN
                   % CORRECT SEQUENCE
FOR REPORT 2:
   TITLE "SUMMARY OF" / "JANUARY REMITTANCES".
   REPORT FOR EACH DAY-RECD: TOTAL-REMITTANCES WHICH
      IS TOTAL(NET-PAYMNT).
   SUMMARIZE FOOTING TOTAL-REMITTANCES.
   SUPPRESS SORT.  % INPUT INFO ALREADY IN
                   % CORRECT SEQUENCE
```

In this specification, first the vocabulary "CUSTV" is referenced. Next information is specified as input to each report as follows:

1. The physical records corresponding to REMIT-FILE is specified as input.

2. Next all logical records corresponding to YR-MO-RECD = 8501 (year and month received is 1985 January) from the input physical records REMIT-FILE are made available for reporting (selected).

3. Then the control-break heading DAY-RECD (day received) is specified to group the data items to be reported.

Then the first report (REPORT 1) is designed as follows:

1. It is titled "JANUARY REMITTANCES".

2. All data items corresponding to the control-break heading DAY-RECD and the following columns are specified for inclusion in the report: REMIT-ACCT-NO (account number), CUST-NAME (customer name), NET-PAYMNT (net payment), DISCNT-TAKEN (discount taken), and CHK-NO (check number).

3. Total net payment for each day reported and for all days reported, respectively, are included in the report through the SUMMARIZE statement.

4. The SUPPRESS SORT statement is used because the information is already in the correct sequence.

Finally, the second report (REPORT 2) is designed as follows:

1. The report is titled "SUMMARY OF JANUARY REMITTANCES."

2. The total amount for each data item corresponding to the control-break heading NET-PYMNT within each control-break heading DAY-RECD is specified for inclusion in the report. These total amounts correspond to the control-break heading TOTAL-REMITTANCES.

3. The sum of the total amounts corresponding the TOTAL-REMITTANCES are included in the report through the SUMMARIZE statement.

4. The SUPPRESS SORT statement is used because the information is already in the correct sequence.

Processing the above specification produces the required report, which is shown in Figure 3-3.

```
                    JANUARY REMITTANCES                          PAGE 1

DAY RECD:  01

       REMIT     CUST NAME          NET PAYMENT          DISCNT      CHK NO
       ACCT                                              TAKEN
         NO

         112   JOE BROWN          $    211.00    $     20.50    31811510
         123   SUSAN KELLY        $    501.00    $     50.10    33355560
         111   JOHN D. DOE        $    123.10    $      0.00    11166000
           .
           .
           .

SUMMARIES FOR DAY RECD:  01
TOTAL                            $   8113.03

DAY RECD:  02

       REMIT     CUST NAME          NET PAYMENT          DISCNT      CHK NO
       ACCT                                              TAKEN
         NO

         203   BILL M. SMITH      $     50.00    $      0.00    11351000
         501   FRED ADAMS         $     13.13    $      1.00    55661111
         100   MARY P. JONES      $    506.00    $     56.83    10123450
           .
           .
           .

SUMMARIES FOR DAY RECD:  02
TOTAL                            $   1776.53
           .
           .
           .

SUMMARIES FOR FINAL
TOTAL                            $ 66010.80
```

```
                    SUMMARY OF
                JANUARY REMITTANCES                              PAGE 1

                  DAY          REMITTANCES
                 RECD

                   01        $   8113.03
                   02        $   1776.53
                    .                .
                    .                .
                    .                .

      SUMMARIES FOR FINAL    $ 66010.80
      TOTAL
```

Figure 3-3. Example Multiple Report

SECTION 4

REPORT LANGUAGE STATEMENTS

The language statements used to design and prepare reports through the REPORTER III System are each explained in detail in this section. The clauses, basic language constructs, and functions which can be contained as syntactic variables within report language statements are presented in alphabetical order.

Each statement and clause and most of the constructs and functions are depicted by syntax diagrams. If a syntax diagram contains more than one path, each path in the diagram is identified by an alphabetic character and is discussed in the text labeled by that alphabetic character. Examples are presented for various paths. If an example is complex, the part of the example which illustrates the particular path may be underlined.

For definitions of various terms used consistently in the semantic rules of the report language to identify data or information, refer to Section 2, under "Terminology for Data Identification." For brief descriptions of the various types of vocabulary names which can be specified in appropriate contexts in the report language, refer to Section 2 under "Vocabulary Names."

ABBREVIATED MONTH OPTION

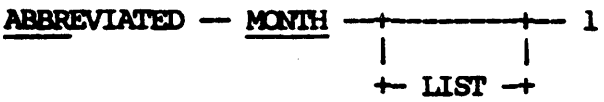
The process option, SET ABBREVIATED MONTH, allows changing the 3-character month strings to an alternate language. The default language is English. The table shown below shows the 3-character abbreviations for months in other languages.

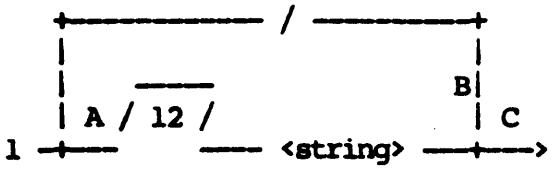
Month	English (default)	French	German	Italian	Spanish
1	JAN	JAN	JAN	GEN	ENE
2	FEB	FEV	FEB	FEB	FEB
3	MAR	MAR	MAR	MAR	MAR
4	APR	AVR	APR	APR	ABR
5	MAY	MAI	MAI	MAG	MAY
6	JUN	JUN	JUN	GIU	JUN
7	JUL	JUL	JUL	LUG	JUL
8	AUG	AOU	AUG	AGO	AGO
9	SEP	SEP	SEP	SET	SEP
10	OCT	OCT	OKT	OTT	OCT
11	NOV	NOV	NOV	NOV	NOV
12	DEC	DEC	DEZ	DIC	DIC

The 3-character month string is used for the MMM part of the DDMMMYY-DATE and DDMMMYYYY-DATE clauses in the TITLE statement and PRINT insert clause. The 3-character month string is also used in the DDMMMYY-DATE and DDMMMYYYY-DATE output of the Date Convert function.

To avoid entering the ABBREVIATED MONTH OPTION for every report specification in which a different, nondefault language is desired, it is recommended that this option be stored in the vocabuary with the REPLACE statement, and then the name given in the REPLACE statement be used when the option is desired.

The syntax for the ABBREVIATED MONTH option is as follows:





The paths of this syntax diagram are explained below:

Path	Explanation
A	This path must be taken 12 times. The string given can not exceed 3 characters in length.
B	Take this path until all 12 abbreviated month strings have been entered.
C	Take this path after all 12 abbreviated month strings have been given.

Example:

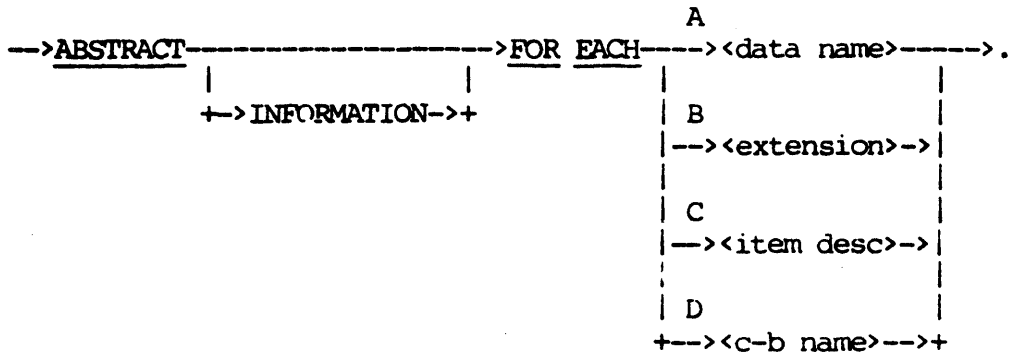
```
SET ABBR MONTH LIST          % TO ITALIAN
    "GEN"/"FEB"/"MAR"/"APR"/
    "MAG"/"GIU"/"LUG"/"AGO"/
    "SET"/"OTT"/"NOV"/"DIC".
```

ABSTRACT STATEMENT

The ABSTRACT statement suppresses detailed information and specifies that only summary information for a designated data item or a designated previously defined control-break item be input to all reports. The ABSTRACT statement can only be used in the INPUT section in a multiple report specification.

The ABSTRACT statement essentially specifies that information within multiple logical records associated with a particular control-break item value be abstracted into a single logical record. This logical record contains summary information only, but is treated as a detailed record for report specification purposes within each Report Section. Summary information for a control-break item consists of items related to the control-break item which have a constant value for each value of the control-break item. These items usually are derived statistical data items, but they also can be nonstatistical data items. If you use the ABSTRACT statement, you can reference only summary items which are related to the specified control-break item in the subsequent Report Sections.

The syntax for the ABSTRACT statement is as follows:



The paths of this syntax diagram are explained below:

Path	Explanation
A-C	Take one of these path, whichever applies, to define a data item as a control-break item. The item must be nonstatistical, and control break must not have been specified previously in another statement.

Example: ("CLIENT")

```
VOCABULARY IS "CLIENT".
FOR INPUT TO ALL REPORTS:
   INPUT ACCT-INVOICE-INFO.
   SUPPRESS SORT.
   ABSTRACT FOR EACH CUST-NO.
   TOTAL-INV-AMT IS TOTAL(TOTAL-AMOUNT FOR
      EACH CUST-NO) NUM(6,2).
   TOTAL-INV-AMT-PAID IS TOTAL(AMOUNT-PAID
      FOR EACH CUST-NO) NUM(6,2).

FOR REPORT 1:
      .
      .
      .
```

Input information for each report in the example consists of summary information related to CUST-NO (that is, information related to a customer account). Detailed invoice information is suppressed and does not appear in any of the specified reports. The summary information for each CUST-NO consists of the following summary items: all items within the accounts receivable record (such as BRANCH, CUST-NO, CREDIT-LIMIT); the derived items (extensions) TOTAL-INV-AMT, TOTAL-INV-AMT-PAID; and INV-CUST-NO and INV-BRANCH-NO (if these items have constant values for each CUST-NO value).

D Take this path to reference a previously defined control-break item.

Example: ("INVENT")

```
VOCABULARY IS "INVENT".
FOR INPUT:
   INPUT PARTMF.
   GROUP BY DEPARTMENT.
   DEPT-INVENT-VALUE IS TOTAL(INVENT-DOLLAR-
      VALUE FOR EACH DEPARTMENT) NUM(9,2).
   ABSTRACT INFORMATION FOR EACH DEPARTMENT.
FOR REPORT:
   SELECT DEPT-INVENT-VALUE > 50000.00.
   REPORT DEPARTMENT, DEPT-INVENT-VALUE.
```

The information described in the <input section> as input to the report consists of a logical record for each department. The logical record contains the summary items DEPARTMENT and DEPT-INVENT-VALUE. The following report is produced:

PAGE 1

DEPARTMENT	DEPT INVENT VALUE
1031	55010.03
1120	73176.20
1300	120177.88
.	.
.	.
.	.

ACCEPT STATEMENT

The ACCEPT statement specifies data items which are parameters to the generated report program. These data items are given values once at the start of the generated report program run, and the values remain constant throughout the run. These data items can be referenced by <name> in other report language statements to control selection, titles, sampling limits, or extension calculations.

Accepted data values can come from the operator display terminal (known as ODT or SPO), card reader, or tape or disk file. The origin of the accepted data is determined by the Processing-option ASSIGN statement. The accepted data values are supplied in free-form format, with each value separated by one or more spaces. If the ACCEPT statement is present, the generated report program asks the user for values of the data items specified by the Process-option ASSIGN statement. After a report is produced, based on the accepted values, the generated report program asks for a new set of accepted data item values and produces another report. This continues until an END is recognized in the input stream or an EOF (End-of-File) occurs.

When the processing mode is set to ON-LINE and the ASSIGN LISTING statement has not been specified or has been set specifically to TERMINAL BACKUP, only one set of accepted data item values is processed and only one report (or one set of reports, in the case of a multiple-report specification) is produced.

Example:

```
ACCEPT USER-NAME STRING(15)
       .
       .
       .
TITLE "REPORT FOR" USER-NAME.
```

The report program reads a 15-character string value for USER-NAME and produces a report. This continues until an END is detected. This USER-NAME might then be used in a TITLE statement to identify the person who is to receive the report.

The following input data items would produce two identical reports, except for the values for USER-NAME.

Example:

```
"   JOHN DOE     "
"  FRANK ADAMS   "
END
```

You can, optionally, combine input data items. You are limited to 80 characters per entry, and you cannot split a data item between two entries. The following would also produce two identical reports, except for the values for USER-NAME.

Example:

```
"   JOHN DOE     " " FRANK ADAMS   " END
```

USE OF ACCEPT STATEMENT ON B 2000/B 3000/B 4000 SERIES

On B 2000/B 3000/B 4000 Series of systems, the following are conditions that affect the use of the ACCEPT statement when you specify data associated with the ACCEPT statement from the ODT (SPO).

- If you enter a string with embedded blanks (spaces), you must include an at sign (@) immediately before and after the string.
- If you are specifying data items for the ACCEPT statement in combinations, you are limited to 60 characters per entry.

The following example shows how to enter such an ACCEPT statement in a report specification.

Example:

Suppose that a report specification includes the following statements:

```
ASSIGN ACCEPTED-DATA TO SPO.
ACCEPT NUMBER1, STRING1 STRING, SUB NUM(2).
```

Supposing that the data items to be input are 12 for NUMBER1, REPORTER III for STRING1, and 7 for SUB, the following are three valid methods of entering the data:

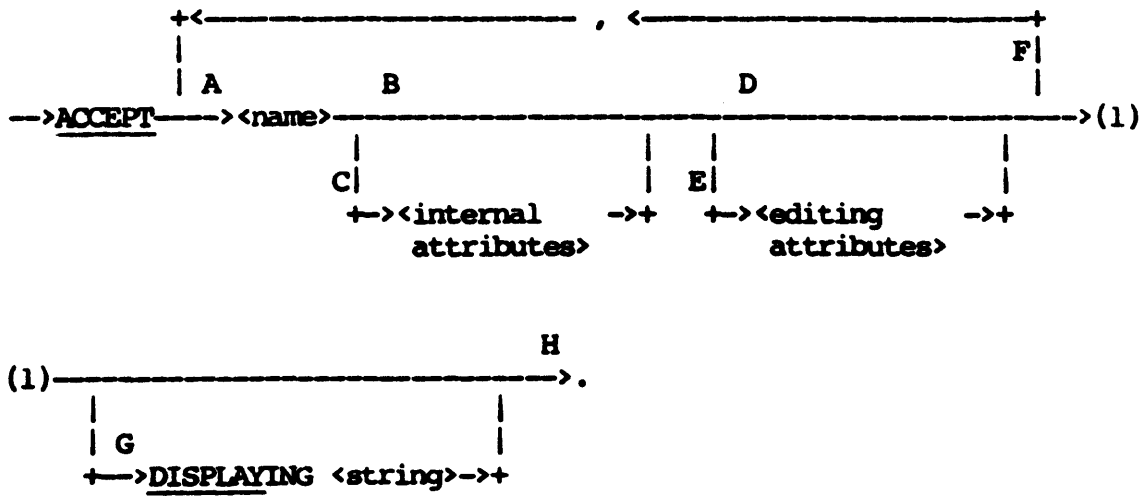
1. 12
   "@REPORTER III@"
   7
   END

2. 12 "@REPORTER III@" 7 END

3. "12 @REPORTER III@ 7 END"

If choice 1 above is used, the values will be transmitted one at a time.

The syntax for the ACCEPT statement is as follows:



The paths of this syntax diagram are explained below:

Path	Explanation

A The <name> which you give here defines an accepted data name. The <name> must be unique; that is, it cannot be a name defined as a vocabulary name or a derived data name. Once defined, the <name> can be used as a <data name> where appropriate in the report specifications.

A value is assigned to this <name> from some external device at the beginning of the generated report program run. The value of the data item remains constant throughout the run. The value is read off the external device by a free-form scanner and correctly aligned to the internal size of the data item. The value must correspond to the type of the accepted data item. String values having no embedded blanks are accepted with or without surrounding quotes. Numeric item values are specified as <number>s and may contain decimal points. Boolean item values are accepted as a 0 or 1 digit.

Example:

```
ACCEPT CODE-ID.
```

CODE-ID references an accepted data item. In this case, no <internal attributes> were given; thus CODE-ID references a signed-numeric item with size equal to the INTEGER-SIZE and FRACTION-SIZE.

B Take this path to specify default internal attributes. The default internal attributes are the following:

1. The type is signed numeric.

2. INTEGER-SIZE and FRACTION-SIZE are used for the size of the numeric item. The default for INTEGER-SIZE is 12 digits. The default for FRACTION-SIZE is 5 places after the decimal point. Refer to Process-option SET statement or Internal Attributes if values other than the default values are desired.

C Take this path to specify internal attributes. The internal attributes can specify type as well as size.

Example:

```
ACCEPT CODE-ID NUM(7).
```

Internal attributes are used to specify that CODE-ID is an unsigned numeric with seven integer digits.

D Take this path to specify default editing. You must take this path if the accepted data item is not to be printed. A default editing picture is derived from the internal attribute of the item. Specification of default editing is illustrated by the example for path C.

E Take this path to specify an editing picture, other than the default, to be used when the data item is to be printed.

Example:

```
ACCEPT CODE-ID NUM(5) WITH PIC "99=999".
```

CODE-ID is printed under the COBOL editing picture "99=999".

F Take this path as many times as necessary to specify all accepted data items. The items specified are accepted as a set at the start of the generated report program run. The order and number of data values read in must correspond to the order and number of items specified in the ACCEPT statement. A free-form scanner is used to scan out the values and place the values correctly aligned into the data item based on its internal attributes. After the specified report (or reports) is produced, another set of data values is read in and assigned to the accepted data items, and another report is produced. This continues until End-of-File or "END" is recognized on the external device. An error exists if End-of-File or "END" is recognized in the middle of a group of accepted data values.

Example:

```
ACCEPT LAST-NAME STRING(20),SOC-SEC-NR NUM(9)
   WITH PIC "999=99=9999".
```

Two values are accepted before each report is produced. LAST-NAME is a 20-character-long string, while SOC-SEC-NR is a 9-digit number. A <string> and a <number> are read off the external device for each report until an END or EOF halts the report process. The following is accepted data for this example:

1. For one set of data:

```
BETHLEHEM          3420091099
```

2. Using quotes:

```
"LINN-HIPSHER"     953992101
```

3. For multiple data sets across multiple records with use of the END option:

```
BETHLEHEM          342009199    1st set

"LINN-HIPSHER"     953992101    2nd set

NEWMANS-CARDIGAN                3rd set
210007615
END
```

G This path allows up to 55 characters of information to be displayed on the operator display terminal (ODT or SPO). The string must be entirely contained on one line.

Example:

```
ACCEPT CODE-ID NUM(5) DISPLAYING
  "ENTER 5 DIGIT CODE. WHEN FINISHED, ENTER END".
```

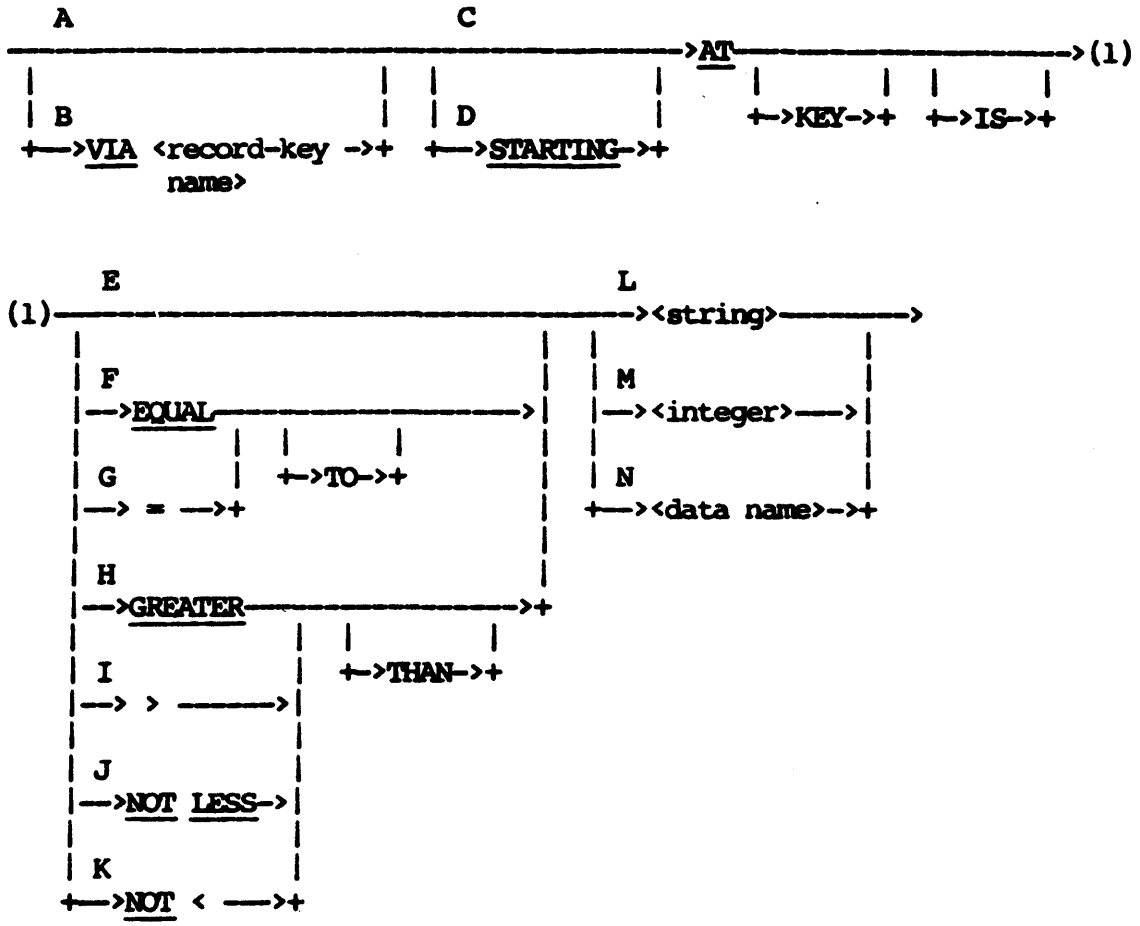
If this path is not taken, the default of "ENTER DATA" will be displayed. The DISPLAYING option is ignored if ASSIGN ACCEPTED-DATA is set to anything other than ODT or SPO.

H Take this path after you have specified all accepted data items.

ACCESS CLAUSE

The Access Clause specifies random access to a system file via a key, or sequential access to a system file beginning at the key value which satisfies a given criterion.

The syntax for the Access Clause is as follows:



The paths of this syntax diagram are explained below:

Path	Explanation
A	This path must be taken for ORGANIZATION RELATIVE and ORGANIZATION SEQUENTIAL system files. For ORGANIZATION INDEXED system files, taking this path implicitly declares access via the prime record key.

Example:

```
INPUT IDXPERSONNEL AT KEY "17881120".
```

IDXPERSONNEL is a file that has ORGANIZATION INDEXED. The single personnel record with prime RECORD KEY value equal to 17881120 is returned.

Example:

```
INPUT ACCT-FILE AT KEY EQUAL TRANS-ACCT-NO.
```

The record in the sequential file ACCT-FILE with actual key equal to the value of an accepted TRANS-ACCT-NO is accessed.

B This path is taken only for ORGANIZATION INDEXED system files to specify access via an ALTERNATE RECORD KEY, or to declare explicitly access via the prime RECORD KEY. Any qualifiers given to the <record-key name> are ignored. The data item referenced by <record-key name> must be declared as a primary RECORD KEY, or as an ALTERNATE RECORD KEY.

Example:

```
INPUT IDXPERSONNEL VIA JOB-TITLE AT
   KEY = "ENGINEER".
```

The first record in the ORGANIZATION INDEXED file IDXPERSONNEL with ALTERNATE RECORD KEY JOB-TITLE equal to ENGINEER is accessed.

NOTE

To access all records with the same key, use path D and a SELECT statement.

C Take this path to access randomly a specific record.

No duplicates are returned when you access an ORGANIZATION INDEXED file via an ALTERNATE RECORD KEY with duplicates. Only the first record satisfying the key value is accessed.

Example:

```
INPUT ACCT-TRANS AT "50000000".
```

The first record with account number equal to 50,000,000 is accessed.

D Take this path if records are to be accessed sequentially starting at a key value which satisfies a certain condition. For ORGANIZATION SEQUENTIAL system files, unused record positions are accessed until an END OF FILE or INVALID KEY condition occurs.

Example:

```
INPUT ACCT-FILE STARTING AT KEY GREATER
   TRANS-ACCT-NO.
```

All records including unused ones in the system file ACCT-FILE with keys greater than the value of TRANS-ACCT-NO are accessed.

Example: ("SHIPV")

```
INPUT ACCT-TRANS STARTING AT KEY
   NOT < "50000000".
```

The above statement results in the input of all account transaction records with a key greater than or equal to 50,000,000.

Example:

```
INPUT IDXPERSONNEL VIA JOB-TITLE STARTING AT
   KEY =  "ENGINEER".
SELECT JOB-TITLE = "ENGINEER".
REPORT . . .

      .
      .
      .
```

In this example, all records from the ORGANIZATION INDEXED file IDXPERSONNEL starting with JOB-TITLE equal to ENGINEER are input. From the records input, the SELECT statement selects only those records with JOB-TITLE equal to ENGINEER. This method allows all the records having ALTERNATE KEY JOB-TITLE WITH DUPLICATES to be made available for reporting.

E-G Taking any of these paths specifies an "equal to" condition.

H-I These paths can be taken only if STARTING has been specified previously. Taking either of these paths specifies that all records with a key value greater than the specified key value are to be input sequentially.

J-K These paths can be taken only if STARTING has been specified previously. Taking either of these paths specifies that all records with a key value greater than or equal to the specified key value are to be input sequentially.

L This path is taken only for ORGANIZATION INDEXED system files to specify a string key value.

Example:

```
INPUT PERS-FILE AT KEY "SMITH, JOHN".
```

M This path is taken only for ORGANIZATION RELATIVE files to specify an unsigned integer key value.

N Take this path to reference a data item which is to supply the key value. The data item must be an accepted data item or an input item contained within a previously input data structure at the same input level. For an ORGANIZATION INDEXED file, the data item must be a string-type item. For an ORGANIZATION RELATIVE or ORGANIZATION SEQUENTIAL file, it must be a numeric-type item.

Example:

```
ACCEPT EMPLOYEE-NUM NUM(7).

INPUT PERS-FILE AT KEY = EMPLOYEE-NUM.

      .
      .
      .
```

This INPUT statement specifies that the employee record with key equal to accepted EMPLOYEE-NUM is to be input.

Example:

```
INPUT PERS-FILE, ADDR-INFO AT KEY
   EQUAL TO NAME.
```

NAME references an item of PERS-FILE. ADDR-INFO is an ORGANIZATION INDEXED file with the prime RECORD KEY equal to ADDR-NAME. The file ADDR-INFO contains information for company employees as well as customers. The above statement specifies the following: for each PERS-FILE record, the data found in NAME will be used to access ADDR-INFO, and a record containing mailing address information for the employee is input.

AGE FUNCTION

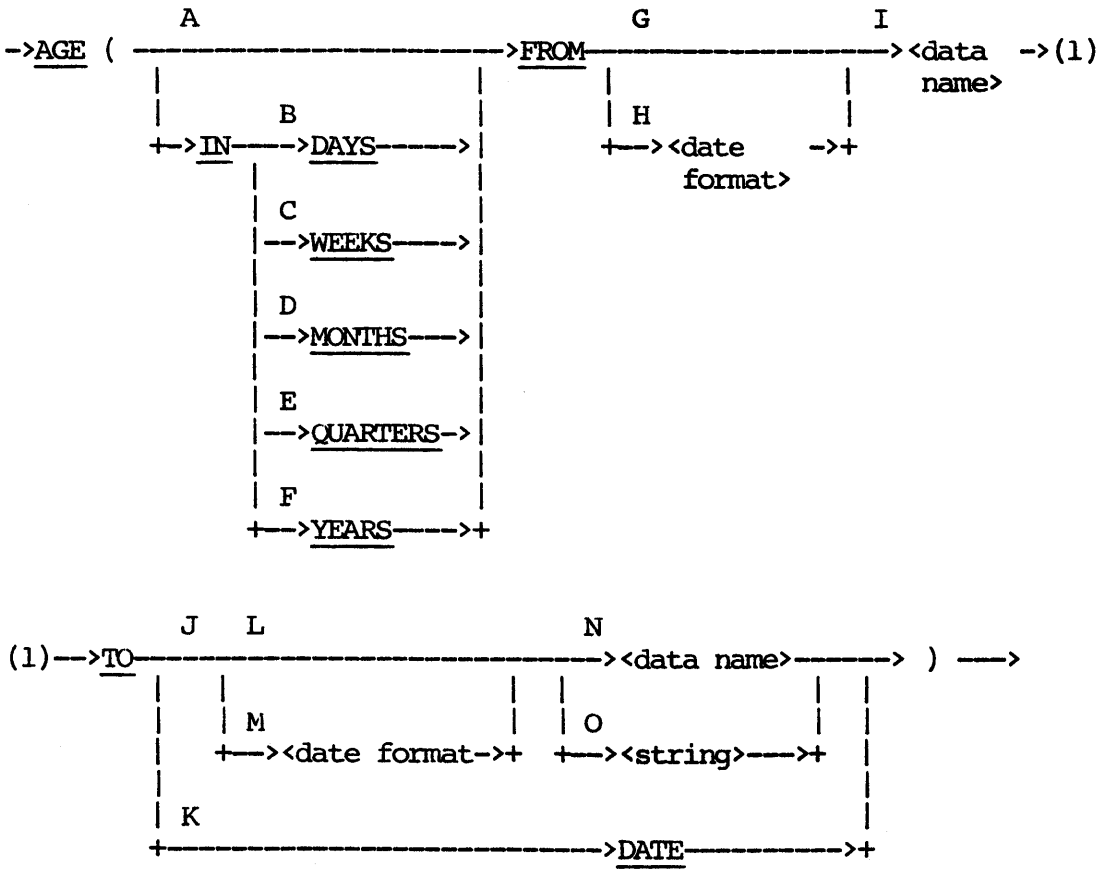
The AGE Function is a nonstatistical function used to calculate the time interval (age) between two dates. The AGE Function enables aging to be done (enables the time interval to be determined) in terms of DAYS, WEEKS, MONTHS, QUARTERS, and YEARS. The time interval is defined as the number of boundaries crossed between the two dates. The age is returned as an 8-digit, signed integer item (S9(8)).

Example:

```
AGE(FROM DATE-LAST-PAYMT TO DUE-DATE)
```

DATE-LAST-PAYMT and DUE-DATE are two data items in the nondelimited MMDDYY format. If the year format is not specified, a 2-digit year is assumed. The AGE Function returns the number of days between the two dates.

The syntax for the AGE Function is as follows:



The paths of this syntax diagram are explained below:

Path	Explanation

A Take this path to specify implicitly that aging be done in terms of days. The example presented above illustrates default aging in terms of days.

B Take this path to specify explicitly that aging be done in terms of days. The results obtained are identical to those obtained in path A above.

Example: ("CLIENT")

```
AGE (IN DAYS FROM DATE-LAST-PAYMT TO DUE-DATE)
```

To obtain the decimal part of the year which the time interval represents, the AGE function in the above example can be used in the following Extension statement:

```
YR IS AGE (IN DAYS FROM DATE-LAST-PAYMT
   TO DUE-DATE)/365.
```

C Take this path to specify that aging be done in terms of weeks. The difference between dates is calculated in terms of 7-day intervals.

Example: ("CLIENT")

```
AGE(IN WEEKS FROM DATE-LAST-PAYMT TO DUE-DATE)
```

D Take this path to specify that aging be done in terms of months. The value returned indicates the difference in month boundaries between dates (for example, 12/28/82 to 1/02/83 has one month boundary crossed).

Example:

```
AGE(IN MONTHS FROM DATE-LAST-PAYMT TO DUE-DATE)
```

E Take this path to specify that aging be done in terms of quarters. The value returned indicates the number of quarter boundaries between dates.

Example: ("CLIENT")

```
AGE(IN QUARTERS FROM DATE-LAST-PAYMT TO DUE-DATE)
```

F Take this path to specify that aging be done in terms of years. The value returned indicates the difference in year boundaries between dates. For example, 12/28/82 to 1/02/83 has one year boundary crossed.

Example ("CLIENT")

```
AGE IN YEARS FROM DATE-LAST-PAYMT TO DUE-DATE)
```

For a sample of how the decimal part of a year which a time interval represents can be determined, refer to the information following the example for path B.

G Take this path to specify implicitly that the FROM date (the earlier date) is coded in the default nondelimited MMDDYY format. All the examples above illustrate this default option.

H Take this path to specify explicitly the format of the earlier date. The format types available are MMDDYY, MMDDYYYY, YYDDD, YYYYDDD, YYMMDD, YYYYMMDD, DDMMYY, DDMMYYYY, and DDDDD.

Example:

```
AGE(FROM YYDDD INVOICE-DATE TO CURRENT-DATE)
```

In this case, INVOICE-DATE is stored in the YYDDD format, while CURRENT-DATE is stored in the MMDDYY format. The function calculates the age in terms of days.

I Take this path to specify the earlier of the two dates. If this date is not the earlier, the result of the AGE Function is negative. The <data name> must reference an item or group which represents a date in a valid format. (Refer to DATE FORMAT in this section for a full explanation of valid formats.) In the example for path H, INVOICE-DATE is the name of the earlier of the two dates.

J Take this path to specify a <date name> or <string> as the later date.

Example:

AGE(FROM OLD-DATE TO NEW-DATE)

Both OLD-DATE and NEW-DATE are data items coded in nondelimited DDMMYY format. The age is calculated in terms of days.

K Take this path to specify that the earlier date be aged to the current date as maintained by the system.

Example:

AGE(FROM INVOICE-DATE TO DATE)

L Take this path if the later date is coded in the default format of nondelimited DDMMYY. The example for path J illustrates this option.

M Take this path to specify a particular format for the later date. The formats available are DDMMYY, DDMMYYYY, YYDDD, YYYYDDD, YYMMDD, YYYYMMDD, DDMMYY, DDMMYYYY, and DDDDD.

Example:

AGE(FROM OLD-DATE TO YYYYDDD NEW-DATE)

In this case, OLD-DATE is coded in the DDMMYY format, while NEW-DATE is coded in the YYYYDDD format. Note that NEW-DATE has a 4-digit year. The function calculates the age in terms of days.

N Take this path to specify a <data name> which represents the later date. This <data name> must reference an item or group which represents a date in valid format. (Refer to DATE FORMAT in this section for a full explanation of valid date formats.) In the example for path M, NEW-DATE is the name of the later of the two dates.

O Take this path to specify the later date as a string constant. The <string> is interpreted as a date according to the format specified by paths L or M.

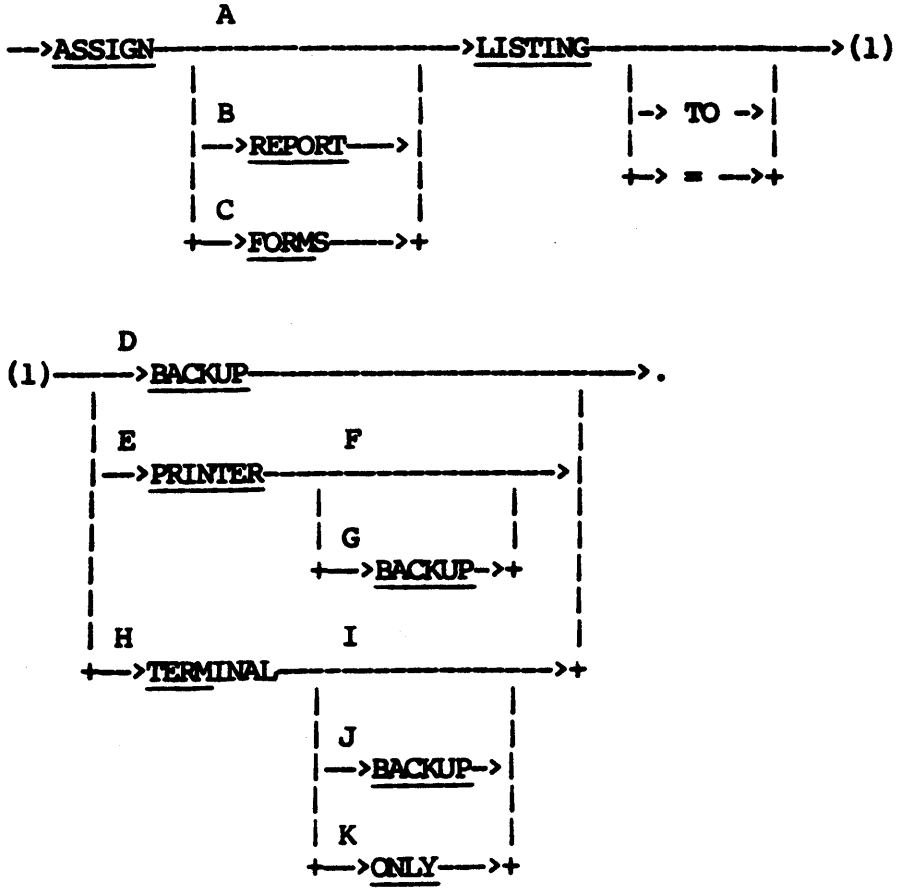
ASSIGN LISTING STATEMENT

The ASSIGN LISTING statement forces the automatically formatted or user-formatted report listing produced by the generated report program to a particular hardware device.

Example:

```
ASSIGN REPORT LISTING TO PRINTER BACKUP.
```

The syntax for the ASSIGN LISTING statement is as follows:



The paths of this syntax diagram are explained below:

Path	Explanation
A	Take this path to force the automatically formatted report to a particular hardware device.
B	This path is identical in meaning to path A.
C	Take this path to force the user-formatted report to a particular hardware device. This path applies only to reports specified with the PRINT statement. Example: ASSIGN FORM LISTING TO BACKUP.
D	Take this path to force the specified listing to a backup device. This device is dependent on the processing mode of the generated report program. (Refer to the Process-option SET statement in this section.) If the mode is BATCH, BACKUP means PRINTER BACKUP and the hardware device is disk or tape (a function of the system software). If the mode is ON-LINE, BACKUP means TERMINAL BACKUP, and the listing is created as a disk file which is accessible through On-Line REPORTER III.
E	Take this path to assign the listing to printer or printer backup.
F	Take this path to specify that the listing be sent to the printer. This is the default assignment for the BATCH processing mode. This option only allows the listing to be sent to the printer. Whether the listing actually goes to the printer or printer backup is a function of the system software. Example: ASSIGN REPORT LISTING TO PRINTER.

G Take this path to specify that the listing be sent to printer backup.

Example:

```
ASSIGN REPORT LISTING = PRINTER BACKUP.
```

H Take this path to assign the report listing to a terminal device.

I Take this path to specify that the listing be sent to the terminal. This path should only be taken if the processing mode is ON-LINE.

Example:

```
ASSIGN LISTING TO TERM.
```

The report listing is sent to the terminal via On-Line REPORTER III.

J Take this path to assign the listing to terminal backup. For the ON-LINE mode, this is the default destination of listing.

If accepted data is used in the report-generation process, only one set of accepted data values is processed and only one report (or one set of reports, in the case of a multiple-report specification) is produced.

Example:

```
ASSIGN FORMS LISTING = TERM BACKUP.
```

K Take this path when the processing mode is ON-LINE to assign the listing to terminal only. Consequently, it is not possible to get a printer listing of the report. It is suggested that TERMINAL ONLY be used for small reports only. This path is illegal if processing mode is BATCH.

Example:

```
ASSIGN REPORT LISTING TO TERMINAL ONLY.
```

This report listing is sent to the terminal via On-Line REPORTER III.

BASE-DATE OPTION

The SET BASE-DATE process option allows you to set the base date to a Julian date other than the default. The default BASE-DATE is January 1, 1900.

In some companies, most notably financial institutions, dates used for opening accounts, closing accounts, and other account transactions are in the format DDDDD. The value DDDDD represents the number of days from a fixed date in the century, called the BASE-DATE.

For example, suppose a savings account was opened on January 1, 1900, and closed on January 2, 1900. The DDDDD open date would be 0, the DDDDD close date would be 1. To compute the interest, the open date is subtracted from the close date and multiplied times the daily interest.

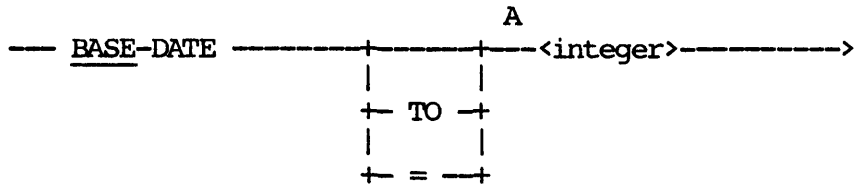
If a savings account was opened September 14, 1983, then the DDDDD would be 30571. This number is 365 times 83 (the number of years from the BASE-DATE) plus 20 (the number of days added for leap years) plus 257 (number of the day) minus 1 (the offset for the BASE-DATE being January 1).

If your company uses a BASE-DATE other than January 1, 1900 in its calculation of dates, then this option allows you to set a BASE-DATE other than the default. Other popular BASE-DATEs are January 1, 1957 (specified as 1957001), and January 1, 1930 (specified as 1930001).

The BASE-DATE is used in conjunction with the DATE-CONVERT FUNCTION using DDDDD-DATE as an output and/or DDDDD as the date format of the input. The BASE-DATE is also used in the AGE function when DDDDD is specified as the FROM data-name and/or the TO data-name.

To avoid entering the BASE-DATE option for every report specification which uses the DDDDD format, it is recommended that this option be stored in the vocabuary with the REPLACE statement and the name given in the REPLACE statement be used when the option is desired.

The syntax for the BASE-DATE option is as follows:



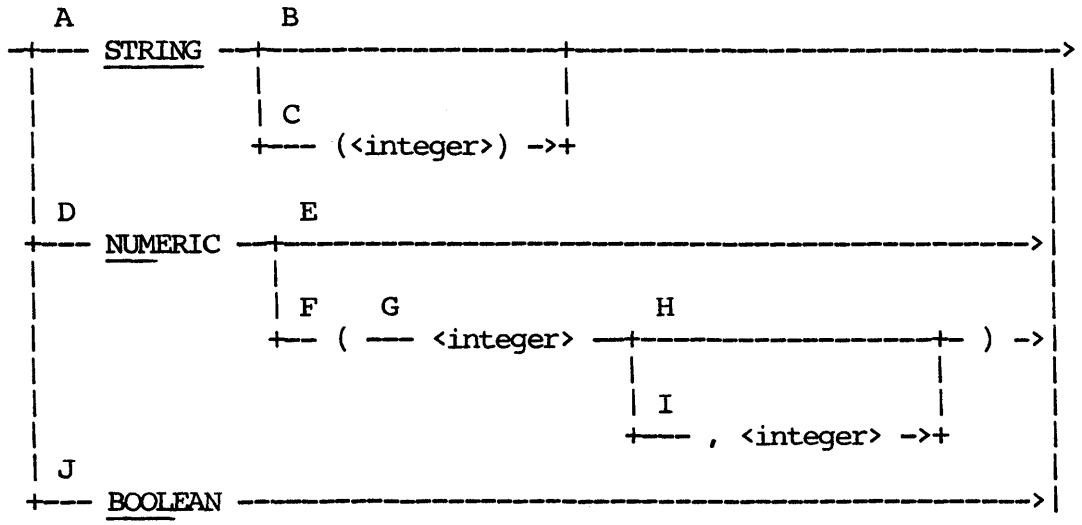
The paths of this syntax diagram are explained below:

Path	Explanation
A	Take this path to specify that date which is to be used as the BASE-DATE. The date format is YYYYDDD or YYDDD, where YYYY represents a 4-digit year, YY represents a 2-digit year, and DDD represents a 3-digit day of the year. REPORTER III will accept either year format (YYYY or YY) and automatically detects the format being used. The actual format used by REPORTER III is YYYYDDD. If the format used for the BASE-DATE is YYDDD, REPORTER III will automatically add the current century to the date. A valid DDD field must be from 1 to 365 for non-leap years and 1 to 366 for leap years. Examples: SET BASE-DATE TO 52001. The BASE-DATE has been changed to January 1, 1952. SET BASE-DATE TO 1880001 The BASE-DATE has been changed to January 1, 1880.

BUILD INTERNAL ATTRIBUTES CLAUSE

The BUILD Internal Attributes clause declares the type and size of the data name that is output from the BUILD statement.

The syntax for the BUILD Internal Attributes clause is as follows:



The paths of this syntax diagram are explained below:

Path	Explanation
A	This path is taken to specify a string-type output for the BUILD name.
B	Take this path if the default string length is going to be used for the BUILD output name being specified. The default length is STRING-SIZE. STRING-SIZE can be set by the Process-option SET statement. If the STRING-SIZE is not set, the system default is used. Refer to Appendix B for more information about system defaults.

C This path is taken to specify a non-default length for the string-type output for the BUILD name. The integer specifies the length of the BUILD name in number of characters.

If the integer value is larger than the value of STRING-SIZE, truncation will result.

Example: ("SHIPV")

```
BUILD
   BILL-NUMBER-PREFIX STRING (2),
   BILL-NUMBER-ONLY  NUM (6)
FROM
   NUMBER-OF-BILL.
```

In this example, the BUILD name of BILL-NUMBER-PREFIX is a two-character string which represents the first two character positions of NUMBER-OF-BILL.

D This path is taken to specify a BUILD name of unsigned numeric type.

E Take this path if the default size for a numeric BUILD name is used. The path specifies an unsigned numeric BUILD name. The default size is determined by INTEGER-SIZE and FRACTION-SIZE, which can both be set by the Process-option SET statement. Refer to Appendix B for information about the system defaults.

F This path is taken to specify the size of the numeric BUILD name.

G The integer specified by taking this path indicates the number of digits in the integer part of the numeric BUILD name.

Example: ("SHIPV")

```
BUILD
   SKIP 2,
   BILL-NUMBER-ONLY NUM (6)
FROM
   NUMBER-OF-BILL.
```

In this example, the BUILD name of BILL-NUMBER-ONLY is a six-character integer. After skipping two character positions of NUMBER-OF-BILL, BILL-NUMBER-ONLY refers to the next six character positions.

H This path is taken to specify an integer-only numeric BUILD name. Path G provides an example.

I This path specifies a fractional part for the BUILD name being defined. The integer specifies the size of the fraction in number of digits. The decimal point is implied and occupies no character positions.

Example:

```
INCHES IS CENTIMETERS * 0.3937 NUM (6,4).

BUILD
   INCHES-WITH-LESS-PRECISION  NUM (6,2),
   SKIP 2
FROM
   INCHES.
```

In this example, the BUILD name INCHES-WITH-LESS-PRECISION is a real number with six positions before the decimal and two positions after. The decimal point is implied and occupies no position.

J This path is taken to specify a Boolean-type BUILD name. Boolean items are represented internally as numeric 1 or 0 and printed as TRUE or FALSE, respectively. A Boolean item occupies one character position.

The following list shows the results that correspond to the character position referenced by a BUILD name Boolean item.

TYPE	VALUE	RESULT
BOOLEAN	0	FALSE
	1	TRUE
NUMERIC	0	FALSE
	1	TRUE
	2 through 9	FALSE

Caution should be exercised when using the Boolean-type BUILD name to reference a string character position, because hardware and software changes could cause unpredictable results.

The following hexadecimal values for string character positions should cause a result of TRUE; all others should cause a result of FALSE.

HEXADECIMAL VALUE	GRAPHIC EQUIVILENT (U.S. CHARACTER SET)
01	SOH (1)
11	DC1 (1)
21	
31	
41	
51	
61	/
71	
81	a
91	j
A1	cent sign (only in some character sets)
B1	
C1	A
D1	J
E1	
F1	1

(1) refers to a data-communication character in EBCDIC form.

BUILD STATEMENT

The BUILD statement is used to create new items or redefine existing items in the following ways:

1. Concatenating data names.
2. Isolating portions of data names.
3. Converting the types of data names.

For a single report, only one BUILD statement is allowed. For multiple reports, one BUILD statement per section is allowed.

The data name used as input to a BUILD statement may be a vocabulary item, a derived item, or an accepted item. In the case of multiple reports, the input data name may be a name from a BUILD statement in the Input Section. Refer to the explanation of Path I (in the syntax diagram that follows) for more information about the allowable input data names.

Computational, Boolean, or A Series binary data names are converted to character form when new names are created with the BUILD statement.

CAUTION

> For A Series users, the value of the output data name is unpredictable if the character position or positions of an input data name are referred to as numeric and contain the hexadecimal value of A through F.

The following examples show how the BUILD statement and the SKIP clause are used to create new items. The syntax diagram and path prompts for the BUILD statement follow the examples.

The first example shows how three existing items can be joined to create one new item.

Example: ("SHIPV")

```
BUILD
   CAL-DATE-OF-BILLING NUM (6)
FROM
```

```
    MONTH-OF-BILLING,
    DAY-OF-BILLING,
    YEAR-OF-BILLING.
```

In this example, the new item CAL-DATE-OF-BILLING was created by joining the vocabulary items MONTH-OF-BILLING, DAY-OF-BILLING, and YEAR-OF-BILLING. The original data names used to build the new item cannot be used with the Date-Convert function or the AGE function unless a complicated extension is created. However, the new item can now be used in these functions.

The next example shows how to use the SKIP clause to skip a specified number of characters in an item being built.

Example: ("SHIPV")

```
BUILD
   SKIP 2,
   BILL-NUMBER-ONLY NUM (6)
FROM
   NUMBER-OF-BILL.
```

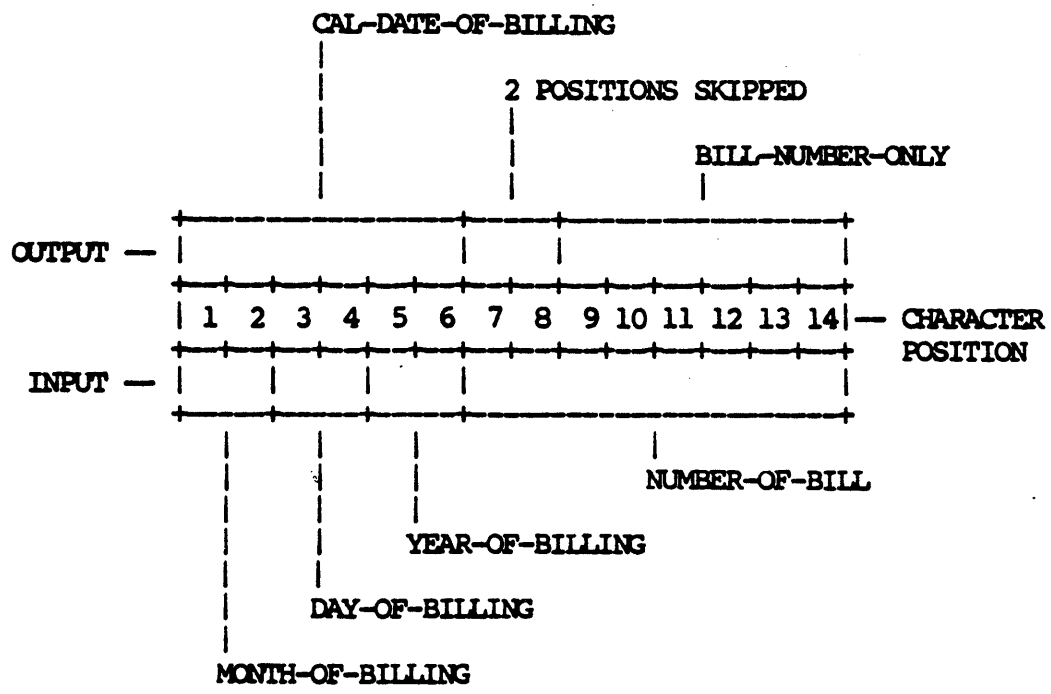
In this example, BILL-NUMBER-ONLY uses only the last six positions of NUMBER-OF-BILL. The first 2 positions of NUMBER-OF-BILL have been skipped.

The third example combines the first and second examples.

Example: ("SHIPV")

```
BUILD
   CAL-DATE-OF-BILLING NUM (6),
   SKIP 2,
   BILL-NUMBER-ONLY NUM (6)
FROM
   MONTH-OF-BILLING,
   DAY-OF-BILLING,
   YEAR-OF-BILLING,
   NUMBER-OF-BILL.
```

The following diagram represents the character layout of the items combined in the third example:



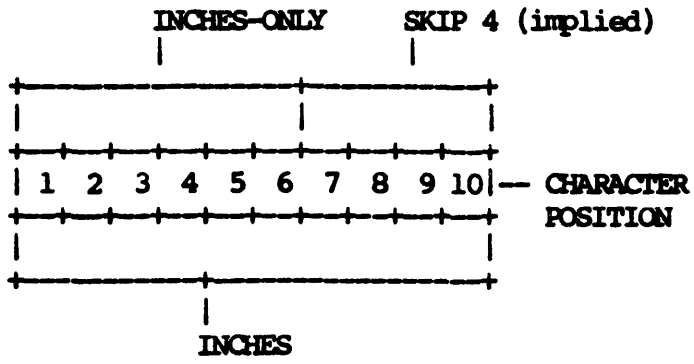
The following example shows how truncating is done to represent a number as an integer only.

Example:

```
INCHES IS CENTIMETERS * 0.3937  NUM (6,4).

BUILD
   INCHES-ONLY NUM(6)
FROM
   INCHES.
```

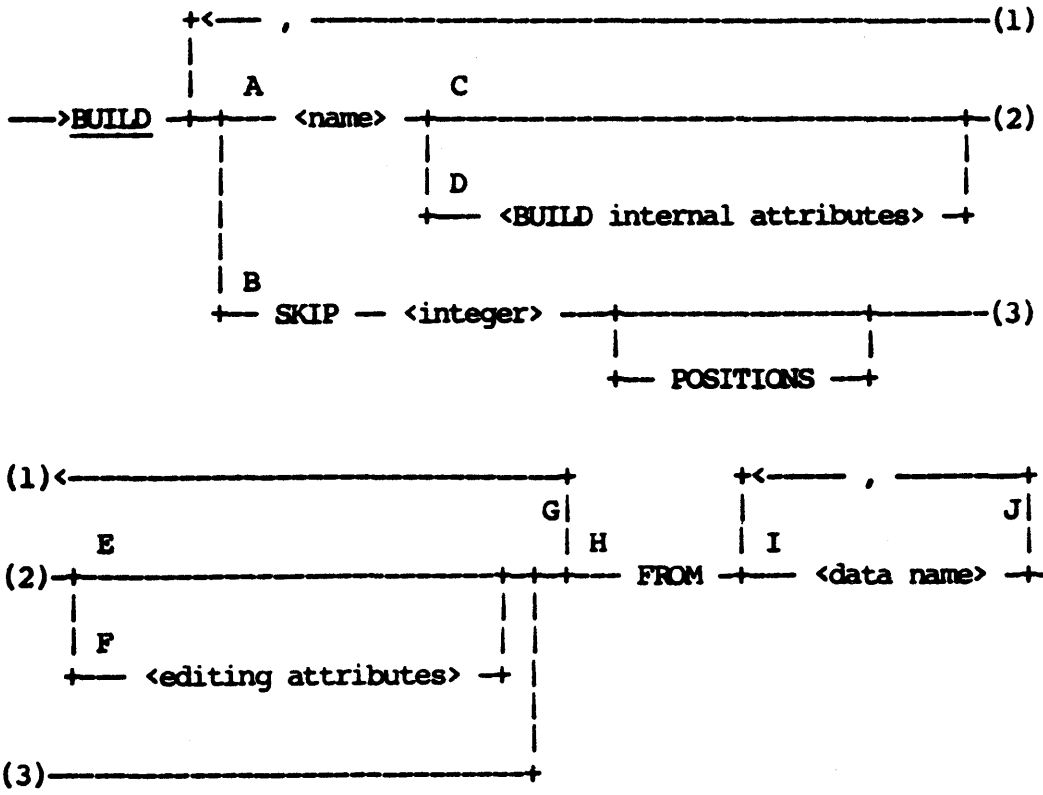
In this example, INCHES occupies six positions before the decimal point and four positions after. Decimal points are implied and occupy no character positions. INCHES-ONLY only refers to the first six positions, so RP3REP automatically fills the unused space with an implied skip of 4.

The following represents the character layout for the preceding example:



INCHES has an implied decimal point between character positions 6 and 7.

The syntax for the BUILD statement is as follows:



The paths of this syntax diagram follow.

Path	Explanation

A The name assigned here defines a BUILD output data name. The name must be unique; that is, it cannot be a name defined as a vocabulary name or a derived data name. Once defined, a BUILD output data name can be used in the report specifications in the same manner as any other data item.

B This path is taken in order to skip a specified number of character positions.

Example: ("SHIPV")

```
BUILD
   SKIP 2 POSITIONS,
   BILL-NUMBER-ONLY NUM (6)
FROM
   NUMBER-OF-BILL.
```

In this example, the first two positions of NUMBER-OF-BILL are bypassed.

C Take this path to specify the following default internal attributes:

- A default type of unsigned numeric.
- A default item size of INTEGER-SIZE or FRACTION-SIZE. Alternatively, the item size can be set with the Process-option SET statement.

D This path is taken to specify the BUILD Internal Attributes for the type and size of items. Refer to the explanation of the BUILD Internal Attributes clause in this section for more information.

Example: ("SHIPV")

```
BUILD
   SKIP 2,
   BILL-NUMBER-ONLY NUM (6)
FROM
   NUMBER-OF-BILL.
```

In this example, BUILD Internal Attributes are used to specify that BILL-NUMBER-ONLY is a numeric item containing six integer positions.

E Take this path to specify default editing. A default editing picture is derived from the <BUILD internal attributes> specified for the item. The example for path D illustrates the specification of default editing. For that example, the default editing picture is Z(5)9.

F This path is taken to specify an editing picture other than the default, which is used when the data item is printed.

Example: ("SHIPV")

```
BUILD
   SKIP 2,
   BILL-NUMBER-ONLY NUM (6) WITH PIC "9(6)"
FROM
   NUMBER-OF-BILL.
```

BILL-NUMBER-ONLY will be printed with the COBOL editing picture "9(6)".

G Take this path as many times as necessary to specify all of the BUILD output names and skips.

H This path should be taken after you have specified all of the BUILD output names and skips.

I This path is taken to specify the data name that will be input to the BUILD statement. The data name must be known to RP3REP prior to being used in the BUILD statement.

Depending on what kind of item you are specifying, the item must have already been included in the report specifications according to the following guidelines:

- A vocabulary item must have been specified through an INPUT statement.
- An accepted item must have been specified in the ACCEPT statement.
- For a derived item, the extension which derives the name must be specified prior to the BUILD statement which uses it.

If the data name is a vocabulary item, it cannot be a group item. The number of character positions used by a vocabulary item is given in the Vocabulary Dictionary listing under the column heading STORAGE LENGTH.

Each data name for an accepted data item or derived data item (extension) must be unique and can be used only once as input in each BUILD statement. If it is necessary to use the same data name more than once, an extension can be used to duplicate the data name. The name of the extension can then be used in the BUILD statement. The extension must be defined prior to the BUILD statement.

The number of character positions used by an accepted or derived item is the value given in the <internal attributes> clause for that item. If an <internal attribute> was not specified, then the number of character positions is determined as follows:

- For a string item, the current setting of STRING-SIZE is used.
- For a numeric item, the sum of the current settings of INTEGER-SIZE and FRACTION-SIZE is used.
- Boolean items have a fixed size of one character.

If a vocabulary, accepted, or derived item is a signed numeric item, only the absolute value will be used. For example, -52.34 is treated as 52.34.

Certain restrictions in using the BUILD statement apply to derived data names which are statistical functions. For a single report, this type of data name cannot be used in the BUILD statement. For multiple reports, this type of data name cannot be used in the BUILD statement in the following situations:

- In the same Input Section in which the data name was created.
- In the same Report Section in which the data name was created.

However, derived data names which are statistical functions created in the Input Section can be used in the BUILD statement of a Report Section if the derived data name has been reassigned to a new data name in the Report Section. Consider the following example:

Example: ("SHIPV")

```
FOR INPUT:
  INPUT ACCT-TRANS.

  AVG-AMOUNT-OWED IS                  % A STATISTICAL FUNCTION
     AVG(AMOUNT-OWED) NUM (6,2).      % IN THE INPUT SECTION
```

FOR REPORT 1:

```
REP-1-AVG-AMOUNT-OWED IS  % THE INPUT SECTION STATISTICAL
   AVG-AMOUNT-OWED.       % DATA NAME IS REASSIGNED TO A
                          % NEW NAME IN THE REPORT SECTION
BUILD
   THOUSANDS-AVG-OWED  NUM (3),
   HUNDREDS-AVG-OWED   NUM (3),
   CENTS-AVG-OWED      NUM (2)
FROM
   REP-1-AVG-AMOUNT-OWED.        % THE NEW NAME IS USED AS
                                 % A BUILD INPUT DATA NAME
```

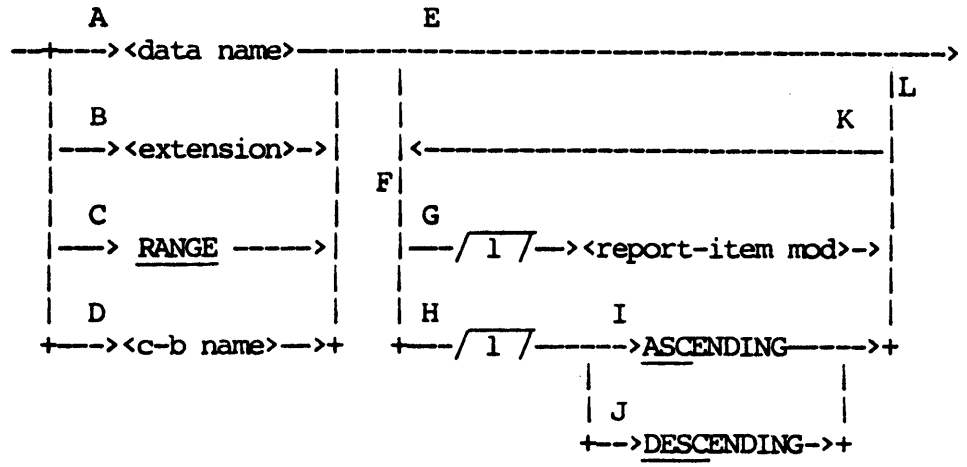
J Take this path as many times as necessary to specify all of the BUILD input names. The total size of all the data names must be equal to or larger than the total size of all the BUILD output names and skips.

K This path should be taken after all of the BUILD input data names have been specified.

C-B-HEADING DESC

C-b-heading Desc describes a control-break heading which is printed for each value of the referenced item.

The syntax for C-b-heading Desc is as follows:



The paths of this syntax diagram are explained below:

Path	Explanation

A Take this path to specify that a control-break heading be printed for each value of <data name>. The appearance of <data name> in this clause identifies it as a control-break name, and information is grouped based on this item. The <data name> must reference a nonstatistical data item. Control breaks for the report must not have been defined previously in another statement.

Example: ("INVENT")

```
REPORT BY DEPARTMENT LISTING PART-NO, QUANTITY.
```

DEPARTMENT is a data item defined to be a control-break item. The value is printed in a control-break heading each time the value changes.

B Take this path to define an <extension> and specify that a control-break heading be printed for each value of the derived data item. The <data name> defined within the <extension>

becomes a control-break name, and information is grouped based on this item. The derived data item must be nonstatistical. Control breaks for the report must not have been defined previously in another statement.

Example: ("INVENT")

```
REPORT BY QUANTITY-HUNDREDS WHICH IS
   QUANTITY / 100 LISTING PART-NO, UNIT-COST.
```

QUANTITY-HUNDREDS is defined to be a control-break item. The value (equal to QUANTITY/100) is printed in a control-break heading each time it changes.

C Take this path to specify that summary information be printed for each value of a range-break item defined previously by a RANGE statement.

Example: ("CLIENT")

```
RANGE BY BALANCE-DUE FROM 0 BY 1000.00 TO 5000.00
   .
   .
   .
REPORT BY RANGE AS "BALANCE DUE" LISTING CUST-NO,
   CREDIT LIMIT, BALANCE-DUE.
```

The RANGE statement defines a single control-break item which is referenced by RANGE. The following report would be produced:

```
BALANCE DUE:  $    0.00 - $ 1000.00
      CUST          CREDIT          BALANCE
        NO           LIMIT              DUE

    001231            1000       $     0.00
    034100             500       $    20.00
    510666             500       $   121.00
       .               .              .
       .               .              .
       .               .              .
```

BALANCE DUE: $ 1000.00 - $ 2000.00

CUST NO	CREDIT LIMIT	BALANCE DUE
107731	5000	$ 1103.00
133320	3000	$ 1257.23
554140	1000	$ 1802.70
.	.	.
.	.	.
.	.	.

D Take this path to specify that a control-break heading be printed for each value of a previously defined control-break item; <c-b name> must reference this item.

When control-break headings are used in conjunction with the GROUP statement, the following restrictions are imposed on the specification of control-break headings:

1. The first control-break heading must be for the first (highest-level) control break declared in the GROUP statement.

2. The last control-break heading need not be for the last (lowest-level) control break in the GROUP statement. However, all intermediate control breaks must be used as control-break headings in the same sequence as they appeared in the GROUP statement.

The following is an example of a valid application using the GROUP statement and REPORT statement:

```
GROUP BY BRANCH, BY CREDIT-LIMIT, BY CUST-NO.
REPORT BY BRANCH; BY CREDIT-LIMIT
   LISTING CUSTOMER-NAME.
```

The following is an example of an invalid application using the GROUP statement and REPORT statement.

```
GROUP BY BRANCH, BY CREDIT-LIMIT, BY CUST-NO.
REPORT BY BRANCH; BY CUST-NO LISTING
CUSTOMER-NAME.
```

E Take this path if default control-break ordering and default printing of the control-break headings is desired. Control-break values are sorted and printed in ascending order by default. The control-break value is also printed by default in each heading, and a default identifier is supplied which is used on the control-break name. (Refer to discussion of DEFAULT IDENTIFIERS under COLUMN DESC in Section 4.)

F Take this path to specify explicitly control-break ordering and/or to specify nondefault printing of control-break headings. The nondefault printing options are as follows:

1. Change the default control-break heading identifier.

2. Specify literal string prefixes and/or suffixes for the control-break item value.

3. Specify conditional control-break value suppression.

G Take this path to specify nondefault printing of the control-break heading. The nondefault options available are listed in the explanation of path F.

Example: ("CLIENT")

```
REPORT BY INV-CUST-NO
   AS "INVOICE CUSTOMER NUMBER"
      LISTING INVOICE-NUMBER, TOTAL-AMOUNT.
```

Within the control-break heading, the value of INV-CUST-NO is identified as

INVOICE CUSTOMER NUMBER: 001179

and not as

INV CUST NO: 001179

H Take this path to specify explicitly the order in which control-break values are to appear in the report. This path cannot be taken if path C was taken, and a range-break item was referenced. The default is to sort and print control-break values in ascending order.

I Taking this path explicitly specifies ascending order for printing control-break values.

Example: ("CLIENT")

```
REPORT BY INV-CUST-NO ASCENDING LISTING
   INVOICE-NO, TOTAL-AMOUNT.
```

Invoices are grouped by customer number. Customer numbers appear in ascending order in the report.

J Taking this path specifies descending order for printing control-break values.

Example: ("CLIENT")

```
REPORT BY INV-CUST-NO DESCENDING LISTING
   INVOICE-NO, TOTAL-AMOUNT.
```

Invoices are grouped by customer number. Customer numbers appear in descending order in the report.

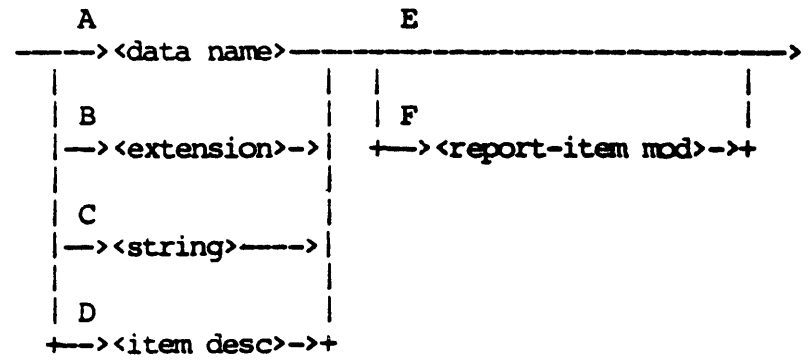
K Take this path to specify both explicit ordering and nondefault printing of control-break values.

L Take this path after all nondefault specifications are made.

C-B-SUBHEADING DESC

C-b-subheading Desc describes an item to be included in the control-break heading which is related to the control-break item.

The syntax for C-b-subheading Desc is as follows:



The paths of this syntax diagram are explained below:

Path	Explanation

A Take this path to specify that the value of a <data name> associated with the control-break item be included in a control-break heading. The value of the <data name> is printed on its own separate line within the control-break heading and is indented five spaces.

Example:

```
REPORT BY CUST-NO INCLUDING CUSTOMER-NAME
   LISTING INVOICE-NO, TOTAL-AMOUNT.
```

The control-break heading for CUST-NO includes the value of CUSTOMER-NAME. This value is written on a separate line in the heading and is identified by "CUSTOMER-NAME:".

B Take this path to define an <extension> associated with the control-break item and to specify that the value of the <extension> be included in a control-break heading. The value of the derived item is printed on its own separate line within the control-break heading and is indented five spaces.

Example: ("CLIENT")

```
REPORT BY CUST-NO INCLUDING UNUSED-CREDIT IS
   CREDIT-LIMIT - BALANCE-DUE LISTING
      INVOICE-NO, TOTAL-AMOUNT.
```

The following shows the control-break heading as it appears in a portion of the report:

```
                .
                .
                .
CUST NO:  001172
    UNUSED CREDIT:  83.27

        INVOICE                    TOTAL
            NO                    AMOUNT

        071310                  $ 131.20
                .
                .
                .
```

C Take this path to specify a <string> to be printed in the control-break heading. The <string> must be 30 characters or less and is printed on its own line, indented five spaces, within the heading.

Example: ("CLIENT")

```
REPORT BY CUST-NO INCLUDING
   "INVOICES ON ABOVE ACCOUNT ARE" LISTING
      INVOICE-NO, TOTAL-AMOUNT.
```

The control-break heading appears as follows in a portion of the report:

```
          .
          .
          .
CUST NO:  001172
    INVOICES ON ABOVE ACCOUNT ARE

                    INVOICE                    TOTAL
                      NO                      AMOUNT

                    051732                  $ 230.00
                       .
                       .
                       .

          .
          .
          .
```

D Take this path to specify that the value of an <item desc> be included in a control-break heading. The value of the <item desc> is printed on its own separate line within the control-break heading and is indented five spaces.

Example: ("CLIENT")

```
REPORT BY CUST-NO INCLUDING
   CREDIT-LIMIT - BALANCE-DUE NUM(5,2)
      LISTING INVOICE-NO, TOTAL-AMOUNT.
```

The control-break heading appears as follows in a portion of the report:

```
CUST NO:  001172
     CREDIT LIMIT - BALANCE DUE:      83.27

        INVOICE                    TOTAL
          NO                      AMOUNT

        071310                  $ 131.20
           .
           .
           .

     .
     .
     .
```

E Take this path if default printing is desired for the data item. The data item then is always printed by default and is identified in the control-break heading by a default identifier. (Refer to DEFAULT IDENTIFIERS under COLUMN DESC in Section 4.)

F Take this path to specify nondefault printing of the data item value. The nondefault printing options are the following:

1. Change the data item identifier.

2. Specify character string prefixes and/or suffixes.

3. Specify conditional data item value suppression.

Example: ("CLIENT")

```
REPORT BY CUST-NO INCLUDING DISCOUNT-PERCENT
   AS "DISCOUNT" SUFFIXED BY "%" LISTING
      INVOICE-NO, TOTAL-AMOUNT.
```

The control-break heading appears as follows in a portion of the report:

```
        .
        .
        .
CUST NO:  001172
   DISCOUNT:  05%
                        INVOICE                  TOTAL
                           NO                   AMOUNT

                        071310                $ 130.50
```

COBOL PICTURE

A COBOL Picture optionally is used to describe how an item is to be printed. When specified as a string in REPORTER III, a COBOL Picture must have no blanks after the first quote and no blanks before the ending quote. It must be a valid COBOL editing picture. The specification is taken as provided and used in the generated program; that is, no syntax checking is performed by the Report Language Analysis Program.

The following characters are used in forming COBOL editing pictures:

Character	Definition
$	Dollar sign
*	Asterisk (check protect)
,	Comma
.	Actual decimal point
B	Space
0	Zero
+	Plus
-	Minus
CR	Credit
DB	Debit
Z	Zero suppress
/	Slash

An ANSI-74 or ANSI-85 COBOL reference manual should be consulted for the exact rules for forming COBOL editing pictures.

Examples:

```
"$(5)Z.99"
"ZZZ,ZZZ.99"
"X(10)"
"B(5)"
"99/99/99"
```

COLUMN DESC

The Column Desc clause specifies the content of a column to be listed in the report.

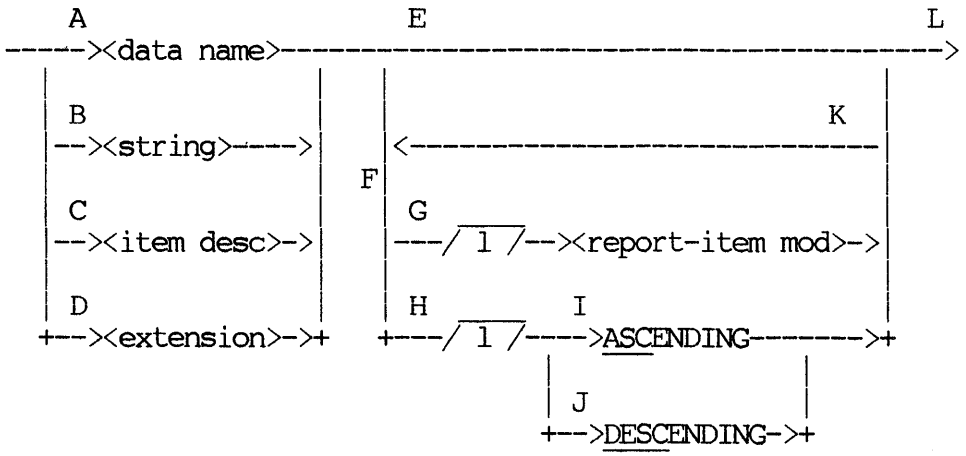
Example: ("INVENT")

```
REPORT PART-NO, QUANTITY.
```

This produces the following type of report:

PART NO	QUANTITY
164310	00023
164320	00018
164322	00130
.	.
.	.
.	.

The syntax for Column Desc is as follows:



The paths of this syntax diagram are explained below:

Path	Explanation

A Take this path to specify that a column consisting of the values of a <data name> be printed. The appearance of the <data name> in this clause identifies it as a column name for purposes of footing. The <data name> provides a default heading for the column.

Example: ("INVENT")

```
REPORT PART-NO.
```

This specifies that one column of information be listed. Each entry in the column represents a part and is given the part number of that part.

Each column name must be unique; that is, a <data name> can be used only once as a column name. If it is desired to report the same data item in more than one column, this can be done by reporting an extension consisting of the data item.

Example:

```
REPORT NAME, SALARY, NEW-NAME IS NAME AS "NAME".
```

In this example, the first and third columns contain identical values and have identical headings.

B Take this path to specify a constant string of characters to be printed in a column. The string must be 30 characters or less. The column is unnamed and has no identifying heading. This allows the introduction of constant values into each line of the listing or the shifting of columns right or left by including an appropriate number of spaces in the string.

Example:

```
REPORT CUST-NO, "ACCOUNT OF", CUSTOMER-NAME.
```

This produces the following type of report:

```
  CUST                              CUSTOMER
    NO                              NAME

117680      ACCOUNT OF              JOHN Q. DOE
128200      ACCOUNT OF              HELEN SMITH
171111      ACCOUNT OF              JACK ADAMS
                 .
                 .
                 .
```

C Take this path to specify that a column consisting of the item values described by <item desc> be listed. There is no column name associated with the column. The <item desc> itself, if less than 30 characters, provides a default heading for the column.

Example: ("INVENT")

```
REPORT PART-NO, UNIT-COST * QUANTITY NUM(8,2).
```

The underlined portion of the REPORT statement specifies a column of data to be printed consisting of the unit cost values times the quantity for each part. The following type of report is produced:

```
  PART          UNIT COST
    NO        * QUANTITY

119017            1671.03
121130            1833.01
132261             926.63
   .                  .
   .                  .
   .                  .
```

D Take this path to define an extension and to specify that a column consisting of the values of this extension be listed. The <data name> defined within the extension becomes a column name and provides a default heading for the column.

Example: ("INVENT")

```
REPORT PART-NO, VALUE-ON-HAND IS
   UNIT-COST * QUANTITY.
```

The underlined portion specifies a column of data to be printed consisting of the value of all units of a particular part number in stock. The column is named VALUE-ON-HAND. The following type of report is produced:

PART NO	VALUE ON HAND
164380	2131.00
170111	1100.15
215010	250.80
.	.
.	.
.	.

E Take this path if the information to be reported (the information in this column) is not to be ordered based on this data item, and if default column printing is acceptable. Each value of the data item appears in the column. A column heading is formed from the default identifier. (Refer to DEFAULT IDENTIFIERS, which follows the path explanations for COLUMN DESC.) If the default identifier is oversized with respect to the width of the data item in the column, an attempt is made to reduce the size of the heading by separating the default identifier on space boundaries and by generating up to a 3-line heading.

F Take this path to specify ordering of information based on this item and/or nondefault column printing. The nondefault printing options are as follows:

1. Change the column heading.
2. Specify item value prefixes and/or suffixes.
3. Specify suppression of certain values in the column.

G Take this path to specify nondefault column printing. The nondefault options available are listed under the explanation of path F above.

Example: ("VOCAST")

```
REPORT ASSET-NO AS "ASSET NUMBER", COST.
```

This illustrates a nondefault column heading. The first column is headed ASSET NUMBER instead of the default heading ASSET NO.

H Take this path to specify that information to be reported be ordered based on this data item. This path cannot be taken if a <row desc> clause was previously specified in the REPORT statement. The ordering specified here is subordinate to any ordering of control-break groups. Also, if ordering is specified for other columns, the ordering specified first is considered the highest level for columns. Since only one sort is done for grouping and ordering, the ordering specified here must be consistent with the ordering specified elsewhere.

Example:

```
REPORT LAST-NAME ASC, FIRST-NAME ASC, SALARY.
```

Information is ordered based on FIRST-NAME for those individuals with identical last names.

I Take this path to specify that information be ordered in ascending sequence based on this column of information.

Example: ("CUSTV")

```
REPORT CUST-NAME ASCENDING, NET-PAYMT.
```

In this case, information reported on customer remittances appears in alphabetic order based on the customer's name.

J DESCENDING specifies that information be ordered based on this item in descending sequence.

Example: ("VOCEMP")

```
REPORT EMPYE-NAME, SALARY DESC.
```

In this case, information on employees is reported in descending order based on SALARY.

K Take this path to specify both ordering of information based on the column and nondefault printing of the column.

L Take this path after all specifications for the column are made.

DEFAULT IDENTIFIERS

A default identifier is used to identify an item in a report when no "AS clause" is specified to identify the item (refer to Report-item Mod in this section for an explanation of "AS clause"). When the values of the item are printed in a column, the default identifier serves as the column heading. The description of the item in a REPORT statement or SUMMARIZE statement determines the default identifier.

If the item is referenced by a <data name>, the default identifier is formed from that <data name> as follows:

1. One space is placed between all elements of the <data name>, except after a "(" or "[" and before a ")" or "]".

2. Every hyphen within a <name> which is not adjoined to other hyphens is replaced by a blank (for example, A-B-C-D becomes A B C D). If more than one hyphen is contained contiguously in a <name>, the first hyphen and every alternate adjoining hyphen is replaced by a blank (for example, A-----B becomes A - - B).

If the preceding would result in an identifier longer than 30 characters, only the first 27 characters are used and an ellipsis (...) forms the last three characters of the identifier.

Examples:

Data Name	Default Identifier
COST	COST
ACCOUNT-NO OF BILL	ACCOUNT NO OF BILL
PART-DESCRIPTION OF UNIT OF CABINET	PART DESCRIPTION OF UNIT OF...
POINTS-OF-SHIPMENT[1]	POINTS OF SHIPMENT [1]

If the item is specified as an <extension>, the default identifier is formed from the derived data item name by replacing any hyphens with blanks.

If the item is a <string>, no default identifier is created.

If the item is described via an <item desc> that contains an <expression> which is a <number> or <string>, no default identifier is created. Otherwise, the default identifier is formed from the <item desc> as follows:

1. Any <internal attributes> and/or <editing attributes> specifications are deleted.

2. One space is placed between all elements of the <item desc>, except after a "(" or "[" or before a ")" or "]".

3. Every hyphen within a <name> which is not adjoined to other hyphens is replaced by a blank (for example, A-B-C-D becomes A B C D). If more than one hyphen is contained contiguously in a <name>, the first hyphen and every alternate adjoining hyphen is replaced by a blank (for example, A-----B becomes A - - B).

If the preceding would result in an identifier longer than 30 characters, only the first 27 characters are used and an ellipsis (...) forms the last three characters of the identifier.

Examples:

Item Desc	Default Identifier
UNIT-COST * QUANTITY NUM(6,2)	UNIT COST * QUANTITY
AVG (COST WHERE MONTH = 03)	AVG (COST WHERE MONTH = 03)
TOTAL (BALANCE-DUE WHERE THIRTY-DAYS-DUE > 0)	TOTAL (BALANCE DUE WHERE TH...

COLUMNS

The Columns clause specifies the content of the columns to be listed on the report. Each line of information in the columns (a detail line) can represent a logical record or summary information related to each value of a control-break item.

Example: ("INVENT")

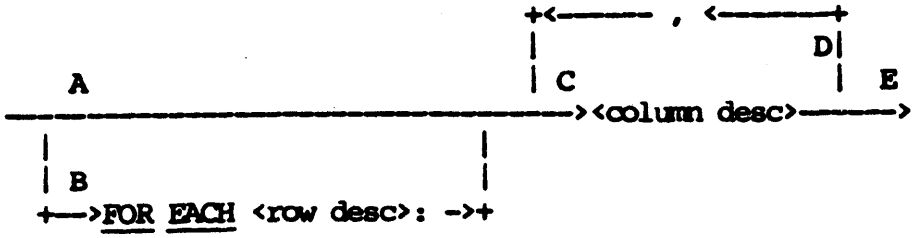
```
REPORT PART-NO, QUANTITY.
```

This produces the following type of report:

PART NO	QUANTITY
164310	00023
164320	00018
164322	00130
.	.
.	.
.	.

Each line represents one logical record.

The syntax for the Columns clause is as follows:



The paths of this syntax diagram are explained below:

Path	Explanation
A	Take this path if information from each logical record is to be listed.
B	Take this path to specify that the columns are to consist of summary information for each value of a control-break item.

Summary information for control-break items consists of items which are related to the control-break item and have a constant value for each value of the control-break item. In many instances, summary items are derived statistical items; but they also can be nonstatistical items. If path B is used, all items reported as columns must be summary items related to the specific control break. Nonstatistical items which are reported as columns are implicitly declared summary items for the specified control break.

The first column printed is specified by <row desc> and contains the values of the control-break item.

<Row desc> defines the default scope for any statistical functions which are specified in subsequent <column desc>s.

Example: ("VOCEMP")

```
RANGE BY EMPYE-AGE FROM 20 BY 5.
   .
   .
   .
REPORT FOR EACH RANGE:  COUNT, AVG(FEDERAL-TAX).
```

The underlined portion specifies that each line or row in the column listing consists of summary information for a particular age range. The following type of report is produced:

RANGE	COUNT	AVG (FEDERAL TAX)
20 - 25	5	$ 751.20
25 - 30	11	$ 843.72
30 - 35	3	$ 1022.80
.	.	.
.	.	.
.	.	.

C This path describes a column of data to be printed in the report. Each column is headed at the top of each page by an appropriate identifier.

D Take this path as many times as necessary to describe all columns which are to be reported.

Columns are printed left to right in the same order as they are specified in the REPORT statement. If more columns are specified than can fit on a page, a line overflow results, unless PAGE-OVERFLOW was set. (Refer to the Report-option SET statement in this section.)

(For discussions of line overflow and page overflow in connection with columns, refer to LINE OVERFLOW and PAGE OVERFLOW, respectively, which follow the path explanations for COLUMNS.)

The following is an example of a multiple-column specification using many nondefault options.

Example:

```
REPORT NAME ASCENDING WHERE CHANGES, AMOUNT, QTY
   WHERE QTY > 0 AS "QUANTITY", "CASES AT"
      WHERE QTY > 0, COST IS AMOUNT * QTY
      AS "ORDER COST", ITEM-NR AS "ITEM"
      PREFIXED BY "#".
```

The preceding statement produces the following report:

NAME	AMOUNT	QUANTITY	ORDER COST	ITEM
ADAMS, JOHN	$10.50	5 CASES AT	52.50	#17
	$12.22	16 CASES AT	73.32	#62
	$13.05	1 CASES AT	13.05	#45
AMON, CHARLES	$ 5.00	4 CASES AT	20.00	#11
	$ 2.00		0.00	#41
BAKER, FRED	$21.10	2 CASES AT	42.50	#26

E Take this path after all columns are specified.

LINE OVERFLOW

If the PAGE-WIDTH is insufficient to print all columns specified in the REPORT statement, by default each line of the listing is continued onto the necessary number of lines. In the same way, corresponding detail headings are continued onto multiple lines. However, the individual headings are not split onto multiple lines in the usual fashion.

The following is an example of line overflow. It shows the result of the report option SET PAGE-OVERFLOW = FALSE (the default). It is useful to compare this example with the example showing the use of page overflow in connection with the same REPORT statement (refer to PAGE OVERFLOW, which follows).

Example

```
REPORT A,B,C,D,E,F,G,H,I,J,K,L.
```

The preceding statement produces the following page layout, which reflects line overflow. In this illustration, the capital letters (A through L) represent headings, and the lowercase letters (a through l) represent data.

```
     PAGE 1

A B C D E
F G H I J
K L

a b c d e
f g h i j
k l

a b c d e
f g h i j
k l


     PAGE 2

A B C D E
F G H I J
K L

a b c d e
f g h i j
k l


     etc.
```

The following is an example of line overflow with a control break. It is useful to compare this example with the example showing the use of page overflow in connection with the same REPORT statement (refer to PAGE OVERFLOW, which follows).

Example:

```
REPORT BY STATE LISTING A,B,C,D,E,F,G,H,I,J,K,L,M.
```

The preceding statement produces the following page layout, which reflects line overflow with a control break. In this illustration, the capital letters (A through M) represent headings, and the lowercase letters (a through m) represent data.

PAGE 1

STATE: ALABAMA

A B C D E
F G H I J
K L M

a b c d e
f g h i j
k l m

a b c d e
f g h i j
k l m

PAGE 2

STATE: ALASKA

A B C D E
F G H I J
K L M

a b c d e
f g h i j
k l m

etc.

NOTE

By default, VERTICAL-SPACING is set to 2 when line overflow occurs. Setting VERTICAL-SPACING to 1 when line overflow is expected results in paper saving at the expense of some report readability.

PAGE OVERFLOW

If more columns are specified than can fit on a page and if the PAGE-OVERFLOW option is set to TRUE (refer to Report-Option SET statement in this section), page overflow occurs. In this case, one logical page of information is printed on several physical pages.

The following is an example of page overflow with the report option SET PAGE-OVERFLOW = TRUE.

Example:

```
REPORT A,B,C,D,E,F,G,H,I,J,K,L.
```

The preceding statement produces the following page layout, which reflects page overflow. In this illustration, the capital letters (A through L) represent headings, and the lowercase letters (a through l) represent data.

```
        PAGE 1.1                    PAGE 1.2

A B C D E                   F G H I J
a b c d e                   f g h i j
a b c d e                   f g h i j
a b c d e                   f g h i j


        PAGE 1.3                    PAGE 2.1

K L                         A B C D E
k l                         a b c d e
k l                         a b c d e
k l                         a b c d e

                    etc.
```

The following is an example of page overflow with a control break and the report option SET PAGE-OVERFLOW = TRUE.

Example:

```
REPORT BY STATE LISTING A,B,C,D,E,F,G,H,I,J,K,L,M.
```

The preceding statement produces the following page layout, which reflects page overflow with a control break. In this illustration, the capital letters (A through M) represent headings, and the lowercase letters (a through m) represent data.

PAGE 1.1

STATE: ALABAMA
A B C D E
a b c d e
a b c d e
a b c d e

PAGE 1.2

STATE: ALABAMA
F G H I J
f g h i j
f g h i j
f g h i j

PAGE 1.3

STATE: ALABAMA
K L M
k l m
k l m
k l m

PAGE 2.1

STATE: ALASKA
A B C D E
a b c d e
a b c d e
a b c d e

etc.

COMBINE STATEMENT

The COMBINE statement enables the language statements in the report specification which follow this statement to be merged with a file designated by <external file name>. This file must be a disk file of report language statements.

The COMBINE statement is useful for merging frequently used macros into the report specification. These macros are defined by <REPLACE statement>s in the designated disk file.

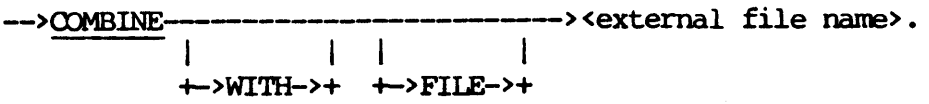
Sequence numbers in the first six positions of each record determine the location of each statement in the combined file. If the first six positions of a record are blank, the sequence number for that record is 0. If the sequence number of a record which you entered following the COMBINE statement is the same as a sequence number in the designated file, the record which you entered is used in place of the corresponding record in the designated file.

If a SAVE SPECIFICATIONS statement follows the COMBINE statement, the saved file is the file resulting from the merge with the specified disk file.

Examples:

```
COMBINE "MACROS".
COMBINE WITH FILE "AUDSPC".
```

The syntax for the COMBINE statement is as follows:

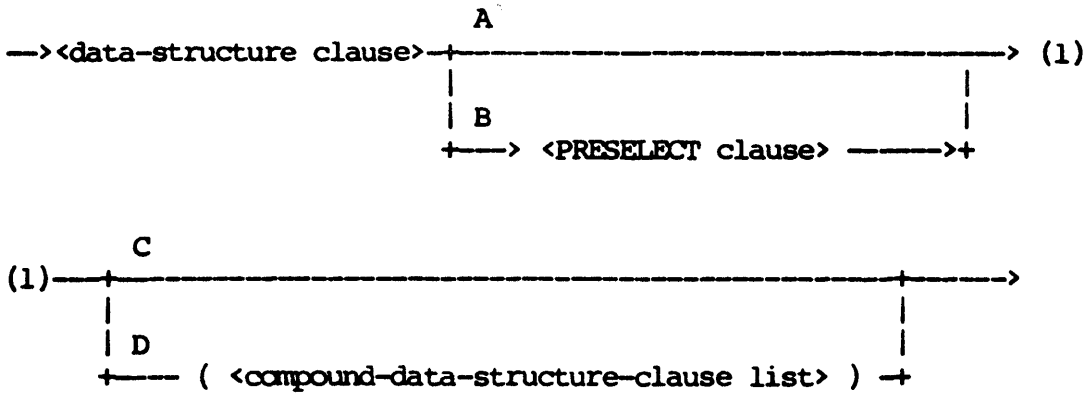


COMPOUND-DATA-STRUCTURE CLAUSE

The Compound-data-structure clause can be used to do the following:

1. To specify access to a single data structure only.
2. To specify one-to-many access to a list of associated data structures.
3. To preselect (using the PRESELECT clause) physical records to be input to the logical record. Preselecting can be done for a single data structure or for a list of associated data structures.

The syntax for the Compound-data-structure clause is as follows:



The paths of this syntax diagram are explained below:

Path	Explanation
A	Take this path if all physical records read from the data-structure specified in the <data-structure clause> are to be input to the logical record.
B	Take this path to pass into the logical record only those physical records which result in a true condition when checked against the PRESELECT Boolean expression. Refer to "PRESELECT clause" later in this section for further information.

C Take this path when no data structure is to be multi-accessed (when it is not necessary to access multiple records of any data structure for each record accessed from the data structure specified).

Example:

```
INPUT PARTS, SUPPLIER.
```

In this example, one record from PARTS is read, and then one record from SUPPLIER is read. The one-to-one access order continues until both files are exhausted. No one-to-many access from either of these files is specified.

D Take this path to access data structures in a one-to-many manner. For each record of the data structure specified, multiple records of one or more data structures specified in path B are accessed. Appropriate clauses for the data structures specified in path B can be used to define how the one-to-many access should be done. The parentheses of path B indicate a nesting of input levels; thus there is a hierarchical order in accessing the data structures. All Compound-data-structure clauses that are separated by commas are considered to be at the same input level, while those enclosed in a set of parentheses are considered to be at an inner input level. A nesting of parentheses in an INPUT statement can define several input levels. INPUT statements containing no parentheses have one level of input. In many applications, INPUT statements consist of one or two input levels.

In general, data structures occurring at the same input level are accessed in order from left to right. In the case of a data structure having an associated <compound-data-structure-clause list>, for each single record of the outer-level structure accessed, the data structures at the next inner-level are accessed until exhausted. A logical record in this case consists of one record from each data structure at each level. When an inner-level structure is multi-accessed for one record at an outer level, each of the logical records reflects the following: the record for the outer-level stays the same, while the record for the inner-level structure varies.

Example:

```
INPUT DEPT-FILE (EMP-NAME).
```

For each department name in DEPT-FILE, all employee names are to be reported. In this example, each logical record contains a department name and one employee name. The department name remains static, while the employee name varies with each logical record, until all employee names are used.

Example: ("CUSTV")

```
INPUT ACCT-FILE(REMIT-FILE MATCHING
   REMIT-ACCT-NO WITH ACCT-NO).
```

For each record of ACCT-FILE, all records of REMIT-FILE whose REMIT-ACCT-NO matches ACCT-NO are input.

Example:

```
INPUT PARTS (SUPPLIERS VIA SUP-CH(LOCATION
   VIA ADDR), QTY-ON-HAND VIA QTY-LK).
```

In this example, for each PARTS record, all associated SUPPLIERS records and QTY-ON-HAND records are input; for each SUPPLIERS record, all associated LOCATION records are accessed. Each logical record consists of one record each from PARTS, SUPPLIERS, LOCATIONS, and QTY-ON-HAND.

COMPOUND-DATA-STRUCTURE-CLAUSE LIST

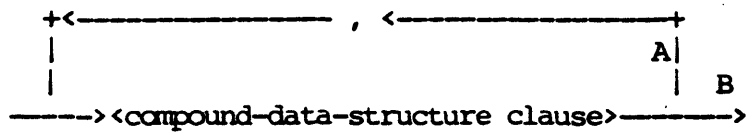
The Compound-data-structure-clause List specifies which data structures from the vocabulary are to be accessed, and the order in which they are to be accessed; all the data structures must be from the same vocabulary.

Example: ("INVENT")

```
INPUT PARTMF.
```

A single data structure is accessed, PARTMF. All information within this data structure can be reported.

The syntax for the Compound-data-structure-clause List is as follows:



The paths of this syntax diagram are explained below:

Path	Explanation
A	You can specify more than one <compound-data-structure clause> in the INPUT statement by taking this path. The data structures identified by the <compound-data-structure clause>s are accessed in a one-to-one corresponding fashion in the order in which they are named, provided the <compound-data-structure clause>s are separated by commas. Access to a data structure in a one-to-one fashion can be dependent or independent. Dependent one-to-one accessing is accomplished by appropriate clauses which specify how the data structure is to be accessed from a previously-specified data structure. Independent accessing involves accessing records from independent data structures, in turn, from left to right.

Example:

```
INPUT PARTS,MASTER-UNIT,SUPPLIER.
```

In this example, one record from PARTS is read, then one from MASTER-UNIT, and then one from SUPPLIER. These three records are now viewed as composing one logical record. The process is repeated until all three data structures are exhausted.

Whenever an attempt is made to access a data structure which is already exhausted, and other specified data structures at the same input level are not yet exhausted, REPORTER III-defined null values are returned for the record entries of the exhausted data structure. The null values to be used in such cases can be specified by a Process-option SET statement.

Example:

```
INPUT SYS-FILE, DMS2DS.
```

This example illustrates that different types of data structures can be specified in the INPUT statement. In this example, a system file (SYS-FILE) and DMS II data set (DMS2DS) are accessed in a one-to-one independent manner and in the order in which they are specified. A logical record is now composed of one physical record from SYS-FILE and one physical record from DMS2DS (in other words, the first logical record consists of the first physical record encountered in SYS-FILE and the first physical record of DMS2DS; the second logical record consists of the second physical record of SYS-FILE and the second physical record of DMS2DS). This matching continues until both files are exhausted. If one file is exhausted before the other, the matching continues with nulls being added in place of the exhausted file.

Example:

```
INPUT MASTER, SYS-FILE AT DIRECT-KEY,
   DS1(DS2 VIA SUBSET1),
      DMS-DS1 FROM SYS-FILE VIA SET1 AT
      KEY = X (DMS-DS2(DMS-DS3)).
```

This example is designed to illustrate how different types of data structures can be accessed together in a complex fashion.

SYS-FILE is a system file associated with another system file, MASTER, through use of a direct key called DIRECT-KEY. This is an example of one-to-one accessing (notice the use of the comma in this relationship). DS2 is a DMS II disjoint data set which is accessed in a one-to-many relationship with DS1 (another disjoint data set) via a subset SUBSET1. This relationship is established by the use of the parentheses. DMS-DS1 is a disjoint data set, and DMS-DS2 and DMS-DS3 are unordered embedded data sets within DMS-DS1 (if they had been embedded standard data sets, they would have required access via a set). X is a data item in SYS-FILE, and KEY is a data item in DMS-DS1. The data item KEY also is used as the key for the set SET1 which spans DMS-DS1.

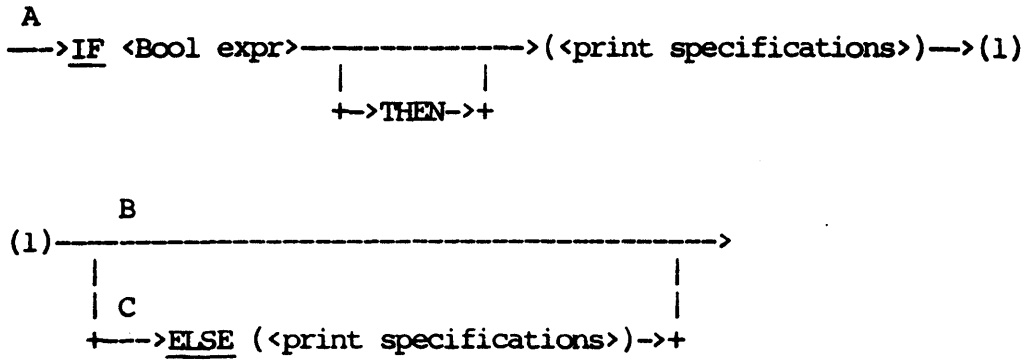
For each record of MASTER accessed, a record of SYS-FILE is accessed using DIRECT-KEY, a record of DS1 is accessed, and a record of DMS-DS1 is accessed using the data item X and the set SET1. For each DS1 record, all DS2 records associated by SUBSET1 are retrieved. For each DMS-DS1 record, all related records of DMS-DS2 are accessed; and for each record of DMS-DS2, all related records of DMS-DS3 are accessed.

B Take this path after you have specified all compound-data-structure clauses.

CONDITIONAL PRINT SPECIFICATION

The Conditional Print Specification establishes the conditions and layout for the printing of a report segment

The syntax for the Conditional Print Specification is as follows:



The paths of this syntax diagram are explained below:

Path	Explanation
A	Take path A to specify a Boolean expression and the <print specifications> to be met if the value of the Boolean expression is TRUE.
B	Take path B if you do not desire to specify print specifications to be met if the value of the Boolean expression is FALSE. For path B, if the value of the given <Bool expr> is TRUE, the <print specifications> given in parenthesis are used to describe the layout of a segment of the report. The print specifications determine the next available print position. For path B, if the value of the given <Bool expr> is FALSE, the <print specifications> given in parentheses are not used, and the next available print position remains unchanged.

Example: ("SHIPV")

```
IF POINTS-OF-SHIPMENT[2]
   NEQ 0 (NL, [POINTS-OF-SHIPMENT[2]] AT 15)
```

If a second point of shipment exists, it is printed as indicated. If not, no printing is specified.

C Take path C if you desire to specify print specifications to be met if the value of the Boolean expression is FALSE.

For path C, if the value of the <Bool expr> is FALSE, the <print specifications> given in the ELSE clause are used to describe the layout of the report segment. These <print specifications> determine the next available print position.

Example: ("VOCEMP")

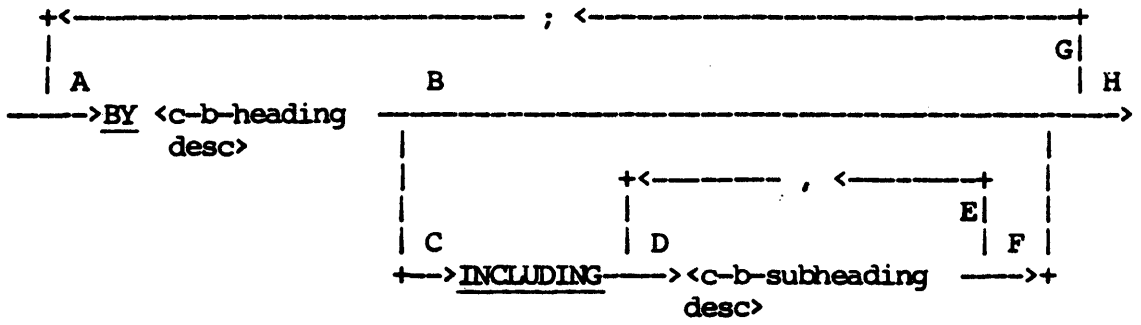
```
IF SALARY < 12000 THEN ([SALARY] IN 10)
   ELSE (10 SPACES)
```

If the employee salary is greater than or equal to 12000 dollars, 10 spaces are printed.

CONTROL-BREAK HEADINGS

Each occurrence of a control break is defined by a Control-break Heading. Control breaks organize information in hierarchical order based on control-break heading definitions.

The syntax for Control-break Headings is as follows:



The paths of this syntax diagram are explained below:

Path	Explanation
A	This clause specifies that a heading be printed for each value of the control-break item described by <c-b-heading desc>. <C-b-heading desc> defines a referenced item as a control-break item if the control-breaks for the report have not been defined in a previous statement. As a minimum, the control-break heading contains the value of the control-break item described in <c-b-heading desc>. It can also contain summary information.

Example: ("CLIENT")

```
REPORT BY INV-CUST-NO LISTING DUE-DATE,
   TOTAL-AMOUNT.
```

INV-CUST-NO is defined as a control-break item. A header is printed for each value of INV-CUST-NO. The listing produced by this specification follows.

INV CUST NO: 001718

DUE DATE	TOTAL AMOUNT
031776	$ 12.83
031776	$ 16.74
040176	$ 101.18

INV CUST NO: 001720

DUE DATE	TOTAL AMOUNT
031976	$ 23.70
041076	$ 103.99
050376	$ 27.04

B Take this path if no information other than the value of the control-break item is desired in the control-break heading.

C Take this path to include summary information in the control-break heading. All items printed in the control-break heading must be summary items related to the control break. Nonstatistical items reported in the control break are implicit summary items.

Example: ("CLIENT")

```
REPORT BY CUST-NO INCLUDING CUSTOMER-NAME
  LISTING INVOICE-NO, TOTAL-AMOUNT.
```

The customer name is a summary item related to the customer number, and is printed with the customer number in each control-break heading.

D <C-b-subheading desc> specifies a line to be printed in the control-break heading which contains the value of the item described in <c-b-subheading desc>. The line is printed beneath the control-break value and indented five spaces.

Example: ("CLIENT")

```
REPORT BY CUST-NO INCLUDING CUSTOMER-NAME
  LISTING INVOICE-NO, TOTAL-AMOUNT.
```

The following are examples of Control-break Headings produced in the body of the report from these instructions:

```
CUST NO.:  073121
   CUSTOMER NAME:  JOHN Q. DOE

          INVOICE              TOTAL
             NO               AMOUNT

          003111            $  73.20
          003202            $  15.70


CUST NO.:  073200
   CUSTOMER NAME:  JACK ADAMS

          INVOICE              TOTAL
             NO               AMOUNT

          004300            $  80.00
          005112            $  27.10
```

E Take this path as many times as necessary to specify all additional information desired in the control-break heading. Data items are printed vertically in the heading in the order specified by the <c-b-subheading desc> clauses.

Example:

```
REPORT BY ACCT-NO INCLUDING NAME, ADDRESS, PHONE
   LISTING TRANSACTION-AMT, TRANSACTION-DATE.
```

The following is an example of a Control-break Heading produced for a report by the statement above:

```
ACCT-NO: 01127
   NAME:  JOHN Q. DOE
   ADDRESS:  507 E. PINE ST., BAY CITY, MICH.
   PHONE: 555-1212
```

F Take this path after all information for a control-break heading is specified.

G Take this path as many times as necessary to specify all control-break headings.

Each occurrence of <c-b-heading desc> can define a control break. Control breaks organize information hierarchically based on the order in which they are defined in Control-break Headings, as follows:

1. The information to be reported is first organized based on the values of the control break specified first (outer-level control breaks).

2. Then, for each value of an outer-level control break, the information is organized based on the values of the control break specified next (inner-level control breaks).

Columns of detail information are listed within only the innermost-level control break.

Example: ("VOCAST")

```
REPORT BY DEPT-NO; BY ACQUISITION-YR;
   BY ASSET-TYPE LISTING ASSET-NO, COST.
```

This statement produces the following report segment:

```
DEPT NO: 1311
ACQUISITION YR: 65
ASSET TYPE: 07

                ASSET          COST
                  NO

                511330·        $ 510.23
                512000         $ 120.80
                   .              .

ASSET TYPE: 11

                ASSET          COST
                  NO

                411630         $ 83.70
                   .              .
```

ACQUISITION YR: 66
ASSET TYPE: 07

ASSET NO	COST
333120	$ 71.10
.	.
.	

DEPT NO: 1312
ACQUISITION YR: 65
ASSET TYPE: 08

ASSET NO	COST
610000	$ 606.01
.	.

ASSET TYPE: 11

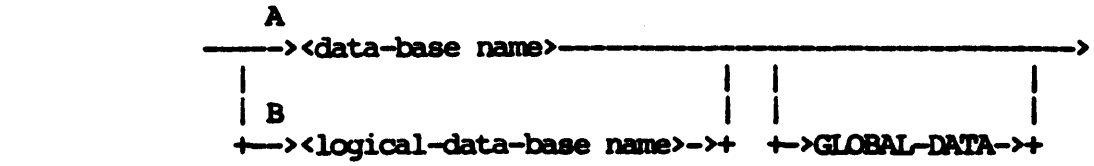
ASSET NO	COST
311010	$ 20.00
.	.

H Take this path after all Control-break Headings are specified.

DATA-BASE CLAUSE

The Data-base Clause is used to input global data from an A Series DMS II data base or logical data base. It can be used only with data bases that have global data, and it can be specified at any input level.

The syntax for the Data-base Clause is as follows:



The paths of this syntax diagram are explained below:

Path	Explanation

A Take this path to specify the name of the data base from which global data is to be accessed.

Example:

```
INPUT UNIVERSITY GLOBAL-DATA.
```

Global data from the UNIVERSITY data base is read (see Figure 4-1).

B Take this path to specify the name of a logical data base from which global data is to be accessed.

Example:

```
INPUT UNIV.
```

Global data from the logical data base UNIV is read.

```
NOSOFSTUDENTS NUMBER (10);
NOSOFCOURSES NUMBER (5);
UNIV-COURSES DATA SET "MAIN FILE" (

    CRS-NAME GROUP (
            DEPARTMENT ALPHA (2) ;
            LEVEL NUMBER (3) ;
            CRS-NO NUMBER (4)) REQUIRED ;
    NOPROF NUMBER (2) ;
    CNTOFCRS COUNT (300) ;
    DAYS-OF-WEEK FIELD (
            MON BOOLEAN ;
            TUES BOOLEAN ;
            WEDS BOOLEAN
            THURS BOOLEAN ;
            FRI BOOLEAN ;
            SAT BOOLEAN) ;
    BUILDING NUMBER (3) ;
    ROOM ALPHA (2) NULL IS "NO" ;
    COURSENAME ALPHA (24) ;
    FLAG-BITS FIELD (12) ;
    HOURSCRDT NUMBER (4) ;
    CLASS-SIZE NUMBER (2) ;
    PROFESSOR IS IN UNIV-PERSONNEL COUNTED
    OCCURS 3 TIMES;
    BOOKS UNORDERED DATA SET (

        LC NUMBER (9) ;
        TITLE ALPHA (60) NULL IS BLANKS ;
        AUTHR ALPHA (30)
            )
            BUFFERS = 1 + 1 PER USER,
            AREAS = 10
            AREASIZE = 500,
            POPULATION = 5 ,
            BLOCKSIZE = 5
            ;

    BOK SUBSET OF BOOKS UNORDERED LIST DATA LC;
    STUDENTS DATA-SET   (

        LAST-NAME ALPHA (15) REQUIRED ;
        FIRST-NAME ALPHA (10) REQUIRED ;
        )
        POPULATION = 300
        ;

    STUDSET SET OF STUDENTS
        KEY IS (LAST-NAME ASCENDING, FIRST-NAME) I-S DUPLICATES
        LOADFACTOR = 75  TABLESIZE = 12  AREAS = 100
        )   % END RECORD DESCRIPTION  OF UNIV-COURSES DATA SET
        POPULATION = 1000
VERIFY (HOURSCRDT GTR 0 AND CLASS-SIZE LEQ 60) AND NOPROF NEQ 0;

UNIV-C-SET SET OF UNIV-COURSES
    KEY IS CRS-NAME DESCENDING I-S NO DUPLICATES;
UNIV-PERSONNEL DATA SET
POPULATION = 997

(
    USC-COUNT COUNT (100)
    NAME GROUP (
        LASTNAME ALPHA (15) ;
        FIRSTNAME ALPHA (10)) REQUIRED ;
    SEX BOOLEAN ;
    AGE NUMBER (2) NULL IS HIGH-VALUE ;
    SSNUM NUMBER (9) REQUIRED UNIQUE ;
    DPT ALPHA (4) ;
    RANK ALPHA (1) ;
    SALARY NUMBER (S7,2)   INITIALVALUE IS LOW-VALUE;
    COURSES IS IN UNIV-COURSES COUNTED OCCURS 8 TIMES;
    SUPR ID IN UNIV-PERSONNEL WITH NO PROTECTION
                                                    );

SS-U-P SET OF UNIV-PERSONNEL
    KEY IS SSNUM I-S NO DUPLICATES ;
U-P-SET SET OF UNIV-PERSONNEL
    KEY IS NAME INDEX SEQUENTIAL DUPLICATES;
```

Figure 4-1. A Series Data Structure Definition Language (DASDL)

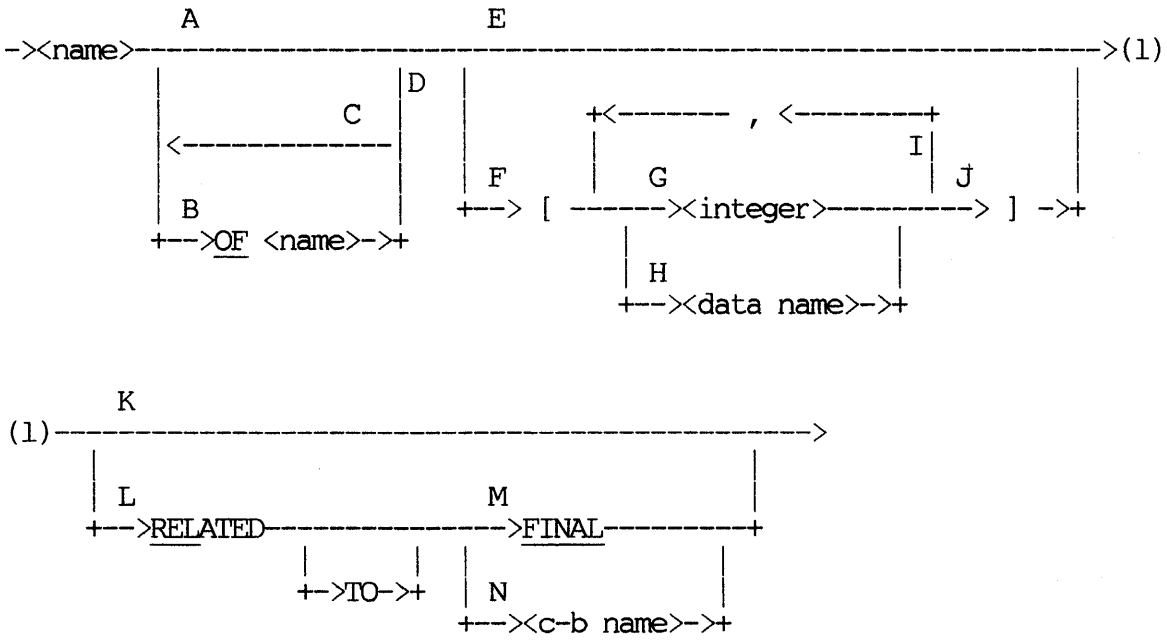
DATA NAME

A Data Name references an item or group. It is a <name>, or a <name> qualified by other <name>s, and possibly subscripted by <integer>s and other <data name>s. A <name> possibly can be associated with a particular control-break item.

Examples:

```
ASSET-NO
INVENT-DOLLAR-VALUE OF CUST-ACCOUNT
LAST-SALE [25]
BALANCE-DUE [BRANCH]
BRANCH REL FINAL
```

The syntax for Data Name is as follows:



The paths of this syntax diagram are explained below.

Path	Explanation
A	If the <name> which references the data item or group is unique, no qualification is necessary and this path is taken. The <name>s of data items and groups appearing in the vocabulary which do not require qualification are not qualified when they appear in the vocabulary listing. The <name>s of accepted data items and derived data items must never be qualified.
B	Take this path to qualify a <name> by another <name> in order to identify it uniquely. Any <name>s requiring qualification because of duplication within the vocabulary are properly qualified on the vocabulary listing. The qualification of <name>s is governed by the rules of COBOL. The qualification of <name>s appearing in the vocabulary is always acceptable, but may not always be necessary. As a minimum, a data item or group used in the vocabulary need only be qualified to make it unique with the data structure specified in the INPUT statement. For example, if the data name TEST-QUAL-NAME is defined under the files DATA-FILE-A and DATA-FILE-B in the vocabulary and only DATA-FILE-B is used in the report specification, then TEST-QUAL-NAME need not be qualified in the report specification. As long as the data name is unique to the data structures specified in the INPUT statement, the qualification shown in the vocabulary listing need not be used. There is one major exception to the preceding rule: Data names used in the INPUT statement (Matching, Preselect, and so on) must be qualified as shown in the vocabulary report.
C	Take this path to specify additional qualifiers. Example: PART-NO OF UNIT OF CABINET
D	Take this path when all appropriate name qualifiers are supplied.
E	If the <name> given refers to a single item or group as opposed to an array of elements, no subscripting is specified, and this path is taken. The vocabulary listing identifies data items and group items requiring subscripts by noting the number required. The <name>s of accepted data items and derived data items are never subscripted.

F Take this path to specify a single element from an array of elements. The <name>s of data items and groups requiring subscripts are noted in the vocabulary, and the number of subscripts required is provided. Subscripts are specified within the left and right brackets, [ ], and not within parentheses as in COBOL. COBOL index variables cannot be used as subscripts. A <data name> for which subscripts are provided is referred to as a subscripted <data name>.

It is the user's responsibility to ensure that a subscript is within the bounds of the array if the subscripted <data name> is either a DMS II item specified in the INPUT statement or a non-DMS II item specified in any REPORTER III statement.

If a subscript value is not within the bounds of the array, a run-time error (INVALID INDEX or INVALID SUBSCRIPT) could occur during execution of the generated report program.

For the remaining cases, REPORTER III will check subscript values automatically. If an invalid subscript is detected at run-time, the array element is handled as a null item.

Example: ("SHIPV")

```
POINTS-OF-SHIPMENT[1]
```

G Take this path to specify an <integer> as a subscript. The integer must be within the bounds of the array.

Examples:

```
ADDRESS-LINE[1]
STATUS-FLAG[5]
```

H Take this path to specify a <data name> as a subscript. This <data name> must not itself be subscripted, and it must not contain a RELATED TO clause. If the <data name> references a derived data item, it must be a nonstatistical data item defined in the <input section> of a multiple-report specification. The value of the data item referenced by the <data name> must be within the bounds of the array.

Examples:

```
ADDRESS-LINE[I]
POINTS-OF-SHIPMENT[ORIGIN]
```

I Take this path to specify additional subscripts. To reference the desired array element properly, you must take this path the number of times required to specify the number of subscripts identified in the vocabulary listing for the item or group.

J Take this path after you have specified all subscripts.

Example:

```
TABLE[2,1,3]
```

K This path is taken for most <data name>s. Taking this path specifies that the data item is not to be declared explicitly as a summary item. Declaring a data item explicitly as a summary item is only necessary when you define certain statistical data items based on the data item. In many cases, a <data name> is declared implicitly as a summary item by the context in which it is used.

Example:

```
REPORT FOR EACH DEPARTMENT:  PAYROLL, ...
```

Used in this context, PAYROLL is implicitly associated with DEPARTMENT. It is assumed that only one value of PAYROLL exists for each value of DEPARTMENT.

L Take this path to specify explicitly that a data item be associated with FINAL or a control-break item. The RELATED clause explicitly declares a nonstatistical item as a summary item and thus specifies that one and only one value of the data item is associated with all information reported on each value of a previously-defined control-break item. The RELATED clause is implied for control-break items and thus need never be specified for control-break items.

Examples: ("CLIENT")

```
CREDIT-LIMIT RELATED TO CUST-NO
BALANCE-DUE REL TO CUST-NO
```

Specifying the RELATED clause is important only when you use a summary item in a statistical expression in two distinct contexts.

If a statistical function is taken on a summary item, either statistical or nonstatistical, the RELATED clause is used to indicate that the item in this context is to be treated specifically as a summary item. That is, the statistic is accumulated only once for each unique value of the summary item rather than accumulated for each reported logical record. (Refer to STAT PARAMETERS in this section.) When the RELATED clause is used in this context for a nonstatistical data item, it also explicitly declares the item as a summary item.

Example:

```
TOTAL (BUDGET REL TO DEPARTMENT)
```

BUDGET is used as a summary item for the purpose of computing the requested total. The REL clause also explicitly declares BUDGET as a summary item if BUDGET is a nonstatistical data item.

If a statistical expression defines an item which is to be used as a summary item, all nonstatistical summary items within the expression must be declared as such. The RELATED clause is used to declare explicitly a nonstatistical data item as a summary item if the item was not declared previously either implicitly or explicitly as a summary item.

Example:

```
SUMMARIZE FOR EACH DEPARTMENT:
   TOTAL(SALARY)/BUDGET REL TO DEPARTMENT.
```

TOTAL(SALARY)/BUDGET REL TO DEPARTMENT is a statistical expression which defines an item used as a summary item. Therefore, BUDGET, which is a nonstatistical summary item, must be declared as a summary item.

M Take this path to specify that the data item has a constant value and thus to summarize all reported information.

Example:

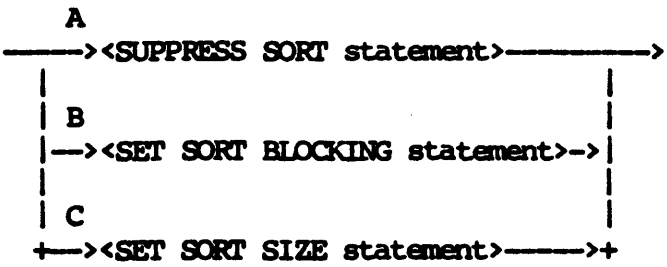
```
COMPANY-ASSETS REL TO FINAL
```

N Take this path to specify that the data item has a constant value for each value of the referenced control-break item.

DATA-PROCESSING-OPTION STATEMENT

A Data-processing-option statement can be used to control the processing of information. Appropriate defaults are taken if you do not specify a data-processing option.

The syntax for the Data-processing-option statement is as follows:



The paths of this syntax diagram are explained below:

Path	Explanation
A	Take this path to suppress the sort required to group and order information in the manner described. This is done if you know that the information is already in the required sequence.
B	Take this path to set the blocking factor of the internal sort file. This provides control of the core usage and efficiency of the sort.
C	Take this path to set the size of the internal sort file. This provides control of disk allocation for this file.

DATA-SET CLAUSE

The Data-set Clause is used to specify input of a A Series DMS II data set or selected members of data sets based on keys of retrieval. You can input embedded data sets, provided the data set in which they are embedded has been specified previously. However, specification of a data set does not cause automatic access to its embedded data sets; to access an embedded data set, you must name it explicitly.

Example:

```
INPUT UNIV-COURSES.
```

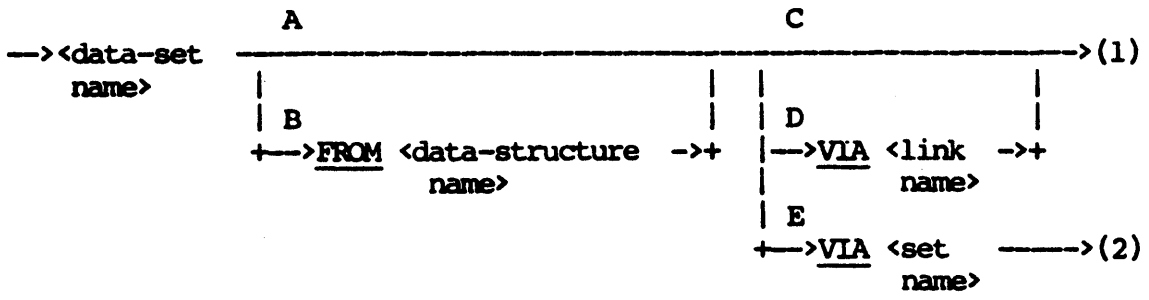
All members of the data set UNIV-COURSES are input (see Figure 4-1).

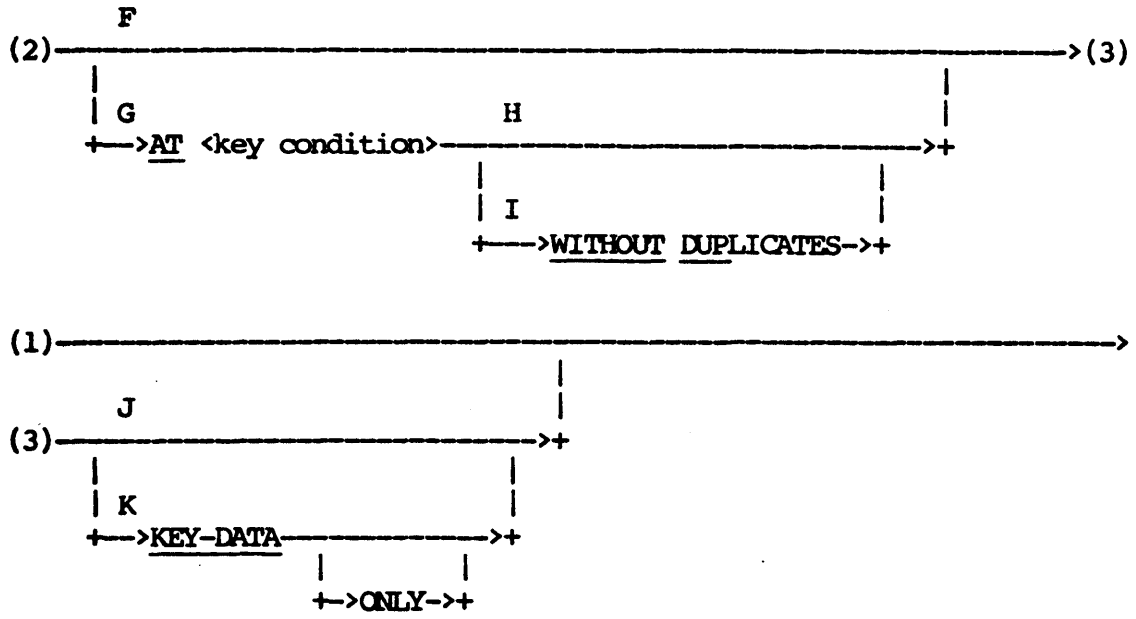
Example: ("CLIENT")

```
INPUT ACCTS-RECV.
```

All members (accounts) of the data set ACCTS-RECV are input.

The syntax for the Data-set Clause is as follows:





The paths of this syntax diagram are explained below:

Path	Explanation

A Take this path if it is not necessary to clarify the one-to-one dependence of the data set on another input data structure.

B The FROM clause must be used whenever the following two conditions exist:

1. The data set is to be accessed in a one-to-one manner from another data structure.

2. The specified access to that data set depends on the value of a data item in the other data structure.

The name of the data structure containing the data item must be specified. This type of dependence is used when a key in a set spanning the data set is compared against the value of a data item in another data structure.

Example:

```
INPUT UNIV-COURSES, UNIV-PERSONNEL FROM
  UNIV-COURSES VIA SP-SET AT SP-KEY = PROF.
```

In this example, UNIV-COURSES and UNIV-PERSONNEL are two disjoint data sets. SP-SET spans UNIV-PERSONNEL with symbolic

key SP-KEY. If PROF is a data item of UNIV-COURSES, the FROM clause is needed to indicate the dependence of UNIV-PERSONNEL on the value of PROF in UNIV-COURSES. All members of UNIV-COURSES are accessed sequentially; for each UNIV-COURSES record, the corresponding UNIV-PERSONNEL record is accessed (see Figure 4-2).

Example:

```
INPUT FILE1, DATASET2 FROM FILE1 VIA SET2
   AT KEY2 = FLD-IN-FILE1.
```

In this example, a system file, FILE1, and a DMS II data set, DATASET2, are accessed in a one-to-one manner. The access to DATASET2 depends on the results of the access to FILE1, as the key KEY2 of set SET2 is compared against the value of a data item in FILE1.

C Take this path if the specified data set is not to be accessed via a set or a link. An embedded, standard data set must be accessed via a set or link.

Example:

```
INPUT UNIV-COURSES.
```

All members of the data set UNIV-COURSES are accessed.

Example:

```
INPUT UNIV-COURSES VIA UNIV-C-SET (BOOKS).
```

For each member of the data set UNIV-COURSES, all members of the embedded data set BOOKS are accessed.

D Take this path if the specified data set is to be accessed via a link. The specified <link name> must name a link embedded in a data set which has been specified previously in the INPUT statement.

Example:

```
INPUT UNIV-COURSES, UNIV-PERSONNEL
   VIA PROFESSOR[1].
```

In this example, all members of the data set UNIV-COURSES are accessed. For each member of UNIV-COURSES, the link PROFESSOR[1] is used to access a member of the data set UNIV-PERSONNEL.

E Take this path if the specified data set is to be accessed through a spanning set.

Example:

```
INPUT UNIV-COURSES VIA UNIV-C-SET.
```

In this example, all members of the data set UNIV-COURSES are accessed using the spanning set UNIV-C-SET.

Example: ("CLIENT")

```
INPUT CUST-ACCT-INFO (INVOICE-INFO
   VIA INVOICES).
```

All INVOICE-INFO records of the embedded, spanning subset INVOICES are accessed for each CUST-ACCT-INFO record accessed.

F Take this path if retrieval keys are not to be used to access specific members of the data set. In this case, all the data set members within the given set are accessed.

G The AT clause allows selective retrieval of members of the specified data set using the keys associated with the spanning set. A <key condition> must be a valid A Series DMS II selection expression for the specified set.

Example:

```
INPUT UNIV-PERSONNEL VIA UNIV-C-SET
   AT CRS-NAME = "PY0030510".
```

All records for the graduate-level course Psychology 510 are accessed.

Example:

```
INPUT UNIV-PERSONNEL VIA SS-V-P
   AT SSNUM = 499502642, UNIV-COURSES
      VIA COURSES[1] (STUDENTS VIA STUDSET).
```

In this example, the member of UNIV-PERSONNEL whose social security number is 499-50-2642 is accessed using the set SS=V=P. The link, COURSES[1], is used to obtain a member of UNIV-COURSES. Finally, all the students for the accessed member of UNIV-COURSES are obtained using the set STUDSET.

H Take this path if all records which satisfy the key condition are to be input.

Example:

```
INPUT STUDENT VIA STUDSET AT LAST-NAME = "JONES".
```

All students whose last name is JONES are accessed.

I The WITHOUT DUPLICATES clause suppresses the input of duplicate records which satisfy the key condition.

Example:

```
INPUT STUDENT VIA STUDSET AT LAST-NAME = "JONES"
   AND FIRST-NAME = "JOHN" WITHOUT DUPLICATES.
```

The STUDENT record for the first student whose name is JOHN JONES is retrieved using the set STUDSET.

J Take this path if the data set itself is to be accessed.

K The KEY-DATA ONLY clause is valid for data sets that have key data; it provides a means of accessing only the key data in the set tables. The data set must be given because the information from the set tables is transferred from the user work area for the data set. However, the data set itself is not accessed. Therefore, items of the data set which are not key data must not be referenced in the report language statement.

Example:

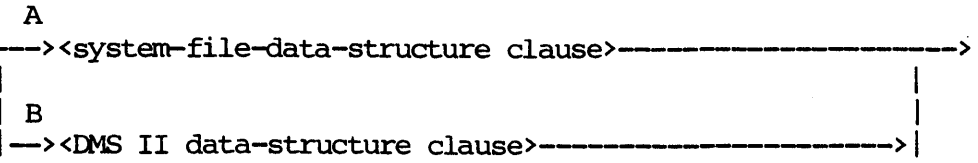
```
INPUT BOOKS VIA BOX KEY-DATA ONLY.
```

In this example, all key data associated with the set BOX is input.

DATA-STRUCTURE CLAUSE

The Data-structure Clause is used to specify the name of a data structure whose records are to be accessed, as well as the method to be used in accessing it.

The syntax for the Data-structure Clause is as follows:



The paths of this syntax diagram are explained below:

Path	Explanation
A	Take this path to access a system file.
B	Take this path if a DMS II data structure clause is to be input.

DATE-CONVERT FUNCTION

A Date-convert Function is a nonstatistical function used to convert a data item or group stored with a particular <date format> to a numeric type data item that represents the same date, but a different format. (The DDMMMYY-DATE format is an exception; it is changed to a string type data item.) All date conversion is done using a 4-digit year. If the date format specified is for two digits, the current century is added to the 2-digit year. When converting to a format with a 2-digit year, the century is stripped from the year.

Example:

```
YYDDD-DATE (MMDDYY INVOICE-DATE)
```

INVOICE-DATE is a data item or group which is coded in the MMDDYY format with a 2-digit year. The function converts this to a YYDDD representation.

When processing date conversions, all dates are converted to 4-digit years if necessary. When a date contains a 2-digit year, the date is converted to a 4-digit year by adding the current century to the year specified in the date.

When mixing dates containing 2- and 4-digit years, care must be taken that the size of the result data name is large enough to hold a date with a 4-digit year. For example, the following syntax will not work as intended because the size of NEW-DATE is not large enough to hold the converted date with a 4-digit year:

```
NEW-DATE IS DDMMYYYY (YYDDD OLD-DATE) NUM (6).
```

The above syntax will work correctly if the new date format is DDMMYY. The correct syntax for a 4-digit year is the following:

```
NEW-DATE IS DDMMYYYY (YYDDD OLD-DATE) NUM (8).
```

REPORTER III may not detect all problems of this type, especially if the result is used later in another calculation. The most common errors are the following:

1. For a 2-digit year, when the year is always the current century.

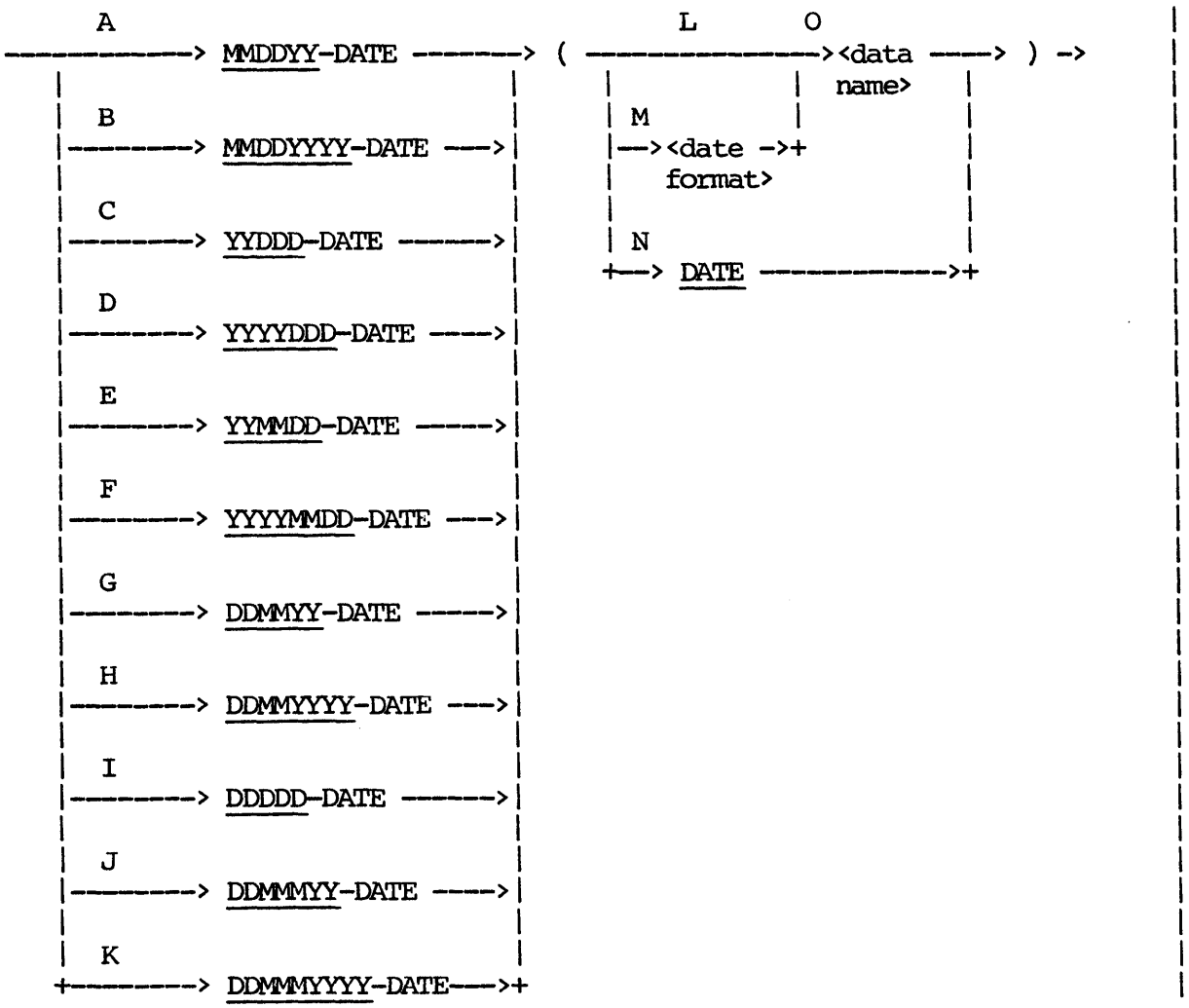
2. For a 4-digit year, when the current century is repeated twice within the year.

These erroneous dates may not be detected unless a date such as a leap year causes an invalid date exception for that year.

NOTE

In the present description of the Date-convert Function, words signifying the type of date format (for example, JULIAN) are no longer indicated. Only letters signifying the actual date format (for example, YYDDD) are indicated. However, REPORTER III will still recognize the words signifying the type of date format (for example, JULIAN) if used in a report specification.

The syntax for the Date-convert Function is as follows:



The paths of this syntax diagram are explained below. In the following path descriptions, DD or DDD refers to days, MM or MMM refers to months, and YY or YYYY refers to years.

Path	Explanation
A	Take this path to specify that the data item or group be converted to a 6-digit integer in the date format MMDDYY. Example: MMDDYY-DATE (YYDDD INVOICE-DATE) INVOICE-DATE is coded in the format YYDDD. This function converts INVOICE-DATE to the format MMDDYY.
B	Take this path to specify that the data item or group be converted to an 8-digit integer in the date format MMDDYYYY. This format uses a 4-digit year. Example: MMDDYYYY-DATE (YYDDD INVOICE-DATE) INVOICE-DATE is coded in the format YYDDD. This function converts INVOICE-DATE to the format MMDDYYYY. Since the date format for INVOICE-DATE contains a 2-digit year, the current century is added to the year before the conversion takes place.
C	Take this path to specify that the data item or group be converted to a 5-digit integer in the date format YYDDD. Example: YYDDD-DATE (MMDDYY INVOICE-DATE) INVOICE-DATE is coded in the format MMDDYY. This function converts the INVOICE-DATE to the format YYDDD.
D	Take this path to specify that the data item or group be converted to a 7-digit integer in the date format YYYYDDD. This format uses a 4-digit year. Example: YYYYDDD-DATE (MMDDYY INVOICE-DATE)

INVOICE-DATE is coded in the format MMDDYY. This function converts INVOICE-DATE to the format YYYYDDD. Since the date format for INVOICE-DATE contains a 2-digit year, the current century is added to the year before the conversion takes place.

E Take this path to specify that the data item or group be converted to a 6-digit integer in the date format YYMMDD.

Example:

YYMMDD-DATE (YYDDD INVOICE-DATE)

INVOICE-DATE is coded in the format YYDDD. This function converts INVOICE-DATE to the format YYMMDD.

F Take this path to specify that the data item or group be converted to an 8-digit integer in the date format YYYYMMDD. This format uses a 4-digit year.

Example:

YYYYMMDD-DATE (YYDDD INVOICE-DATE)

INVOICE-DATE is coded in the format YYDDD. This function converts INVOICE-DATE to the format YYYYMMDD. Since the date format for INVOICE-DATE contains a 2-digit year, the current century is added to the year before the conversion takes place.

G Take this path to specify that the data item or group be converted to a 6-digit integer in the date format DDMMYY.

Example:

DDMMYY-DATE (YYDDD INVOICE-DATE)

INVOICE-DATE is coded in the format YYDDD. This function converts INVOICE-DATE to the format DDMMYY.

H Take this path to specify that the data item or group be converted to an 8-digit integer in the date format DDMMYYYY. This format uses a 4-digit year.

Example:

DDMMYYYY-DATE (YYDDD INVOICE-DATE)

INVOICE-DATE is coded in the format YYDDD. This function converts INVOICE-DATE to the format DDMMYYYY. Since the date format for INVOICE-DATE contains a 2-digit year, the current century is added to the year before the conversion takes place.

I Take this path to specify that the data item or group be converted to a 5-digit integer in the date format DDDDD. The DDDDD format consists of the number of days from the BASE-DATE not including the BASE-DATE.

The default BASE-DATE is January 1, 1900 (YYYYDDD format of 1900001).

Example:

DDDDD-DATE (MMDDYY INVOICE-DATE)

If INVOICE-DATE, coded in the date format MMDDYY, was 091483, then the result returned would be 30571. This is computed as follows:

365 * 83 = 30295	The number of days in the year times the number of years. The default BASE-DATE year field is 1900. The year field of INVOICE-DATE is 83 giving 83 years. Since the date format for INVOICE-DATE contains a 2-digit year, the current century is added before any calculations take place.
30295 + 20 = 30315	Days added for the number of leap years which have occurred from the BASE-DATE.
30315 + 257 = 30572	The day of the year which represents September 14, 1983 (09/14/83).
30572 - 1 = 30571	Subtract the number of days represented by the DDD field of the BASE-DATE. The default BASE-DATE is YYYYDDD format of 1900001 (January 1, 1900).

J Take this path to specify that the data item is to be converted to a 7-character string in the date format DDMMMYY, where DD is a 2-digit field representing the day of the month from 1 to 31, MMM is a 3-character string representing the month, and YY is a 2-digit field representing the year from 00 to 99.

Example:

DDMMMYY-DATE (YYDDD INVOICE-DATE)

If INVOICE-DATE, coded in the format YYDDD, was 83257, then the result would be 14SEP83.

Refer to the Abbreviated Month List option if the 3-character month field is desired in a language other than English.

K Take this path to specify that the data item be converted to a 9-character string in the date format DDMMMYYYY, where DD is a 2-digit field representing the day of the month from 1 to 31, MMM is a 3-character string representing the month, and YYYY is a 4-digit field representing the year from 0000 to 9999.

Example:

DDMMMYYYY-DATE (YYDDD INVOICE-DATE)

If INVOICE-DATE, coded in the format YYDDD, was 83257, then the result would be 14SEP1983. Since the date format for INVOICE-DATE contained a 2-digit year, the current century is added to the year before the conversion takes place.

Refer to the Abbreviated Month List option if the 3-character month field is desired in a language other than English.

L Take this path if the date to be converted is coded in the default format of MMDDYY without delimiters. The default assumes a 2-digit year.

Example:

YYDDD-DATE (INVOICE-DATE)

M Take this path to specify explicitly the format under which the date to be converted is currently coded. (Refer to DATE FORMAT in this section for a full explanation of date formats.) Use this path to convert dates with 4-digit years.

Example:

```
YYDDD-DATE (DDMMYYYY INVOICE-DATE)
```

In this example, the date to be converted (INVOICE-DATE) is coded in the format DDMMYYYY and contains a 4-digit year.

N Take this path if the date to be converted is the system date, which is coded in the format YYMMDD.

Example:

```
DATE-TODAY IS DDMMYY (DATE)
```

O Take this path to specify the name of the date to be converted. The <data name> must represent either a date item or group coded in the specified <date format>. (Refer to DATE FORMAT in this section for a full explanation of acceptable formats.)

In the examples presented above, INVOICE-DATE is the name of the date to be converted.

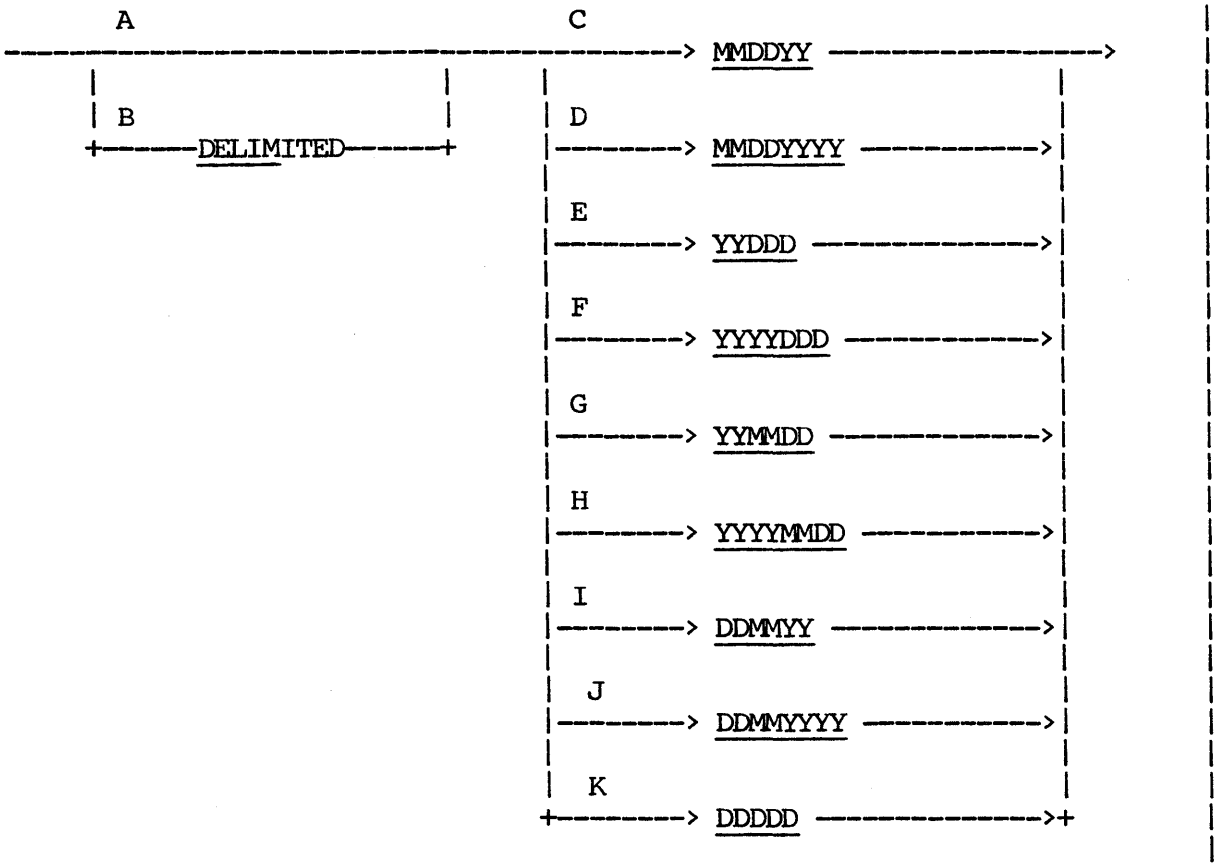
DATE FORMAT

Dates to be converted or aged consist of digits and optional delimiters. The digits and delimiters can be stored in the following ways: string or alphanumeric; numeric display; and numeric computational. The Date Format clause specifies how a particular date is coded so that it can be accessed properly by the REPORTER III System.

NOTE

In the present description of the Date Format clause, words signifying the type of date format (for example, JULIAN) are no longer indicated. Only letters signifying the actual date format (for example, YYDDD) are indicated. However, REPORTER III will still recognize the words signifying the type of date format (for example, JULIAN) in a report specification.

The syntax for the Date Format clause is as follows:



The paths of this syntax diagram are explained below.

Path	Explanation

A Take this path if the date does not have delimiters to separate the fields. In this case, the date consists solely of digits.

B Take this path if the date is coded with delimiters to separate the fields comprising the date. The delimiter used can be any character (usually a slash or a hyphen) and is ignored by the REPORTER III System. DELIMITED is used so that the system can properly locate the date fields. The date can be a string item or a group of display data items. It is assumed that each delimiter is only one character in length.

WARNING

This path cannot be used in conjunction with path K. See path K for additional explanation.

C Take this path if the date is coded in the format MMDDYY. The nondelimited case consists of six digits in the format MMDDYY. The delimited case consists of eight characters in the format MM/DD/YY. MM must be 01 through 12; DD must be 01 through 31; YY must be 00 through 99, standing for the years 00 through 99 in the current century.

Example:

The date February 20, 1988 is expressed as follows:

Nondelimited: 022088
Delimited: 02/20/88

D Take this path if the date is coded in the format MMDDYYYY. The nondelimited case consists of eight digits in the format MMDDYYYY. The delimited case consists of ten characters in the format MM/DD/YYYY. MM must be 01 through 12; DD must be 01 through 31; YY must be 0000 through 9999. This date format uses a 4-digit year.

Example:

The date February 20, 1988 is expressed as follows:

Nondelimited: 022088
Delimited: 02/20/1988

E Take this path if the date is coded in the format YYDDD. The nondelimited case consists of five digits in the format YYDDD. The delimited case consists of six characters in the format YY/DDD. YY must be 00 through 99, standing for the years 00 through 99 in the current century; DDD must be 001 through 366, standing for the number of days since the beginning of the year.

Example:

The date February 20, 1988 is expressed as follows:

Nondelimited: 88051
Delimited: 88/051

F Take this path if the date is coded in the format YYYYDDD. The nondelimited case consists of seven digits in the format YYYYDDD. The delimited case consists of eight characters in the format YYYY/DDD. YYYY must be 0000 through 9999; DDD must be 001 through 366, standing for the number of days since the beginning of the year. This date format expects a 4-digit year.

Example:

The date February 20, 1988 is expressed as follows:

Nondelimited: 1988051
Delimited: 1988/051

G Take this path if the date is coded in the format YYMMDD. The nondelimited case consists of six digits in the format YYMMDD. The delimited case consists of eight characters in the format YY/MM/DD. YY must be 00 through 99, standing for the years 00 through 99 in the current century; MM must be 01 through 12; DD must be 01 through 31.

Example:

The date February 20, 1988 is expressed as follows:

Nondelimited: 880220
Delimited: 88/02/20

H Take this path if the date is coded in the format YYYYMMDD. The nondelimited case consists of eight digits in the format YYYYMMDD. The delimited case consists of ten characters in the format YYYY/MM/DD. YYYY must be 0000 through 9999; MM must be 01 through 12; DD must be 01 through 31. This date format expects a 4-digit year.

Example:

The date February 20, 1988 is expressed as follows:

Nondelimited: 19880220
Delimited: 1988/02/20

I Take this path if the date is coded in the format DDMMYY. The nondelimited case consists of six digits in the format DDMMYY. The delimited case consists of eight characters in the format DD/MM/YY. DD must be 01 through 31; MM must be 01 through 12; YY must be 00 through 99, standing for the years 00 through 99 in the current century.

Example:

The date February 20, 1988 is expressed as follows:

Nondelimited: 200288
Delimited: 20/02/88

J Take this path if the date is coded in the format DDMMYYYY. The nondelimited case consists of eight digits in the format DDMMYYYY. The delimited case consists of ten characters in the format DD/MM/YYYY. DD must be 01 through 31; MM must be 01 through 12; YYYY must be 0000 through 9999. This date format expects a 4-digit year.

Example:

The date February 20, 1988 is expressed as follows:

Nondelimited: 20021988
Delimited: 20/02/1988

K Take this path if the date is coded in the format DDDDD. This date is nondelimited and consists of five digits in the format DDDDD, standing for the number of days from the BASE-DATE. Either the word CENTURY or DDDDD may be used. For example, if the date of September 14, 1983 is used with the default BASE-DATE of January 1, 1900, the DDDDD date format would be:

30571

Refer to the BASE-DATE option and the Date-Convert function for additional information.

DMS II DATA-STRUCTURE CLAUSE

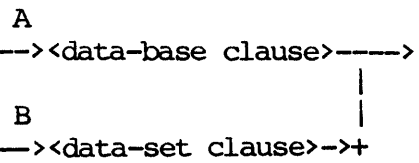
DMS II Data-structure Clauses allow access to various systems' data sets. For the A Series of systems, the Data-structure Clause also enables input of A Series DMS II data base global data.

You should be familiar with the structure of a data base before using DMS II Data-structure clauses. Constructs defined in the Data and Structure Definition Language (DASDL) can affect retrieval of data from the data base. These constructs are not accessible to REPORTER III. For example, when an automatic subset defined in the DASDL uses a WHERE clause to specify the conditions by which data is to be retrieved from the data base, REPORTER III cannot access the WHERE clause. However, the report generated by REPORTER III is affected by the conditions of the WHERE clause and may not contain all the information you need.

A SERIES OF SYSTEMS

For the A Series of Systems, the A Series DMS II Data-structure Clause enables input of A Series DMS II data base global data or records from an A Series DMS II data set. All DMS II data structures referenced in the INPUT statement must belong to the same data base or logical data base.

The syntax for the A Series DMS II Data-structure Clause is as follows:



The paths of this syntax diagram are explained below:

Path	Explanation
A	Take this path to specify input of global data from a A Series DMS II data base or logical data base.
B	Take this path to specify input of an A Series DMS II data set.

B 1000 SERIES OF SYSTEMS

For the B 1000 Series of Systems, the B 1000 DMS II Data-structure Clause enables access to a B 1000 DMS II data set. All data sets referenced in the INPUT statement must belong to the same data base. You can input embedded data sets, provided the data set in which they are embedded has been specified previously. However, specification of a data set does not cause automatic access to its embedded data sets; to access an embedded data set, you must name it explicitly.

Example: ("CLIENT")

```
    INPUT ACCTS-RECV.
```

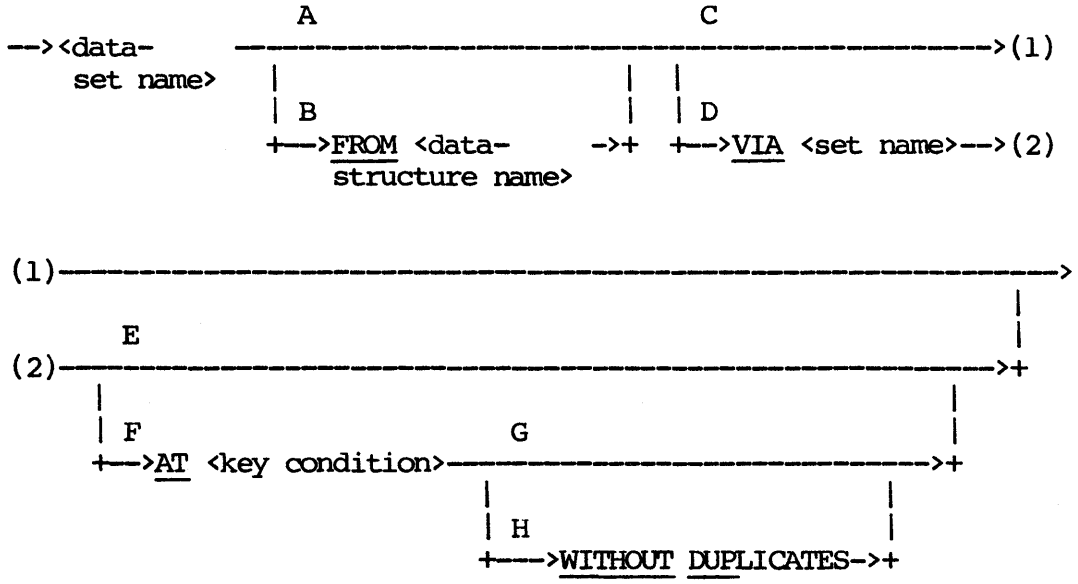
All members (that is, all accounts) of the data set ACCTS-RECV are input.

Example:

```
    INPUT UNIV-COURSES.
```

All members of the data set UNIV-COURSES are accessed.

The syntax for the B 1000 DMS II Data-structure Clause is as follows:



The paths of this syntax diagram are explained below:

Path	Explanation
A	Take this path if it is not necessary to clarify the one-to-one dependence of the data set on another input data structure.
B	The FROM clause must be used whenever the following two conditions exist: 1. The data set is to be accessed in a one-to-one manner from another data structure. 2. The specified access to that data set depends on the value of a data item in the other data structure. The name of the data structure containing the data item must be specified. This type of dependence is used when a key in a set spanning the data set is compared against the value of a data item in another data structure. Example: INPUT UNIV-COURSES, UNIV-PERSONNEL FROM UNIV-COURSES VIA SP-SET AT SP-KEY = PROF. In this example, UNIV-COURSES and UNIV-PERSONNEL are two disjoint data sets. SP-SET spans UNIV-PERSONNEL with symbolic key SP-KEY. If PROF is a data item of UNIV-COURSES, the FROM clause is needed to indicate the dependence of UNIV-PERSONNEL on the value of PROF in UNIV-COURSES. All members of UNIV-COURSES are accessed sequentially; for each UNIV-COURSES record, the corresponding UNIV-PERSONNEL record which satisfies the key condition is accessed (see Figure 4-2).

```
00000100   %THIS DASDL PROGRAM GIVES EXAMPLES
00000150   %OF THE VARIOUS CONSTRUCTS USED IN
00000200   %DASDL TO DESCRIBE A DATA BASE
00000300   PARAMETERS(
00000400       BUFFERS = 10    );
00000600   UNIV-COURSES DATA SET "MAIN FILE" (
00000700       CRS-NAME GROUP (
00000800           DEPARTMENT ALPHA (2);
00000900           LEVEL NUMBER(3);
00001000           CRS-NO NUMBER(4);
00001100       NOPROF NUMBER (2);
00001200       DAYS-OF-WEEK GROUP (
00001300           MON NUMBER(1);
00001400           TUES NUMBER(1);
00001500           WEDS NUMBER(1);
00001600           THURS NUMBER(1);
00001700           FRI NUMBER(1);
00001800           SAT NUMBER(1);
00001900       BUILDING NUMBER(3);
00002000       ROOMNUMBER ALPHA(2);
00002100       COURSENAME ALPHA (24);
00002200       FLAG-BITS ALPHA(12);
00002300       HOURSCRDT NUMBER(4);
00002400       CLASS-SIZE NUMBER(2);
00002500       PROFESSOR SUBSET OF UNIV-PERSONNEL, POPULATION = 3;
00002600       BOOKS UNORDERED DATA SET(
00002700           LC NUMBER(9);
00002800           TITLES ALPHA(60);
00002900           AUTHR ALPHA(30);
00003000       STUDENTS SUBSET OF MSF KEY IS
00003100           (LNAME,FNAME)DUPLICATES,
00003200           POPULATION = 300)
00003700       POPULATION = 1000;
00003800       UNIV-C-SET ORDERED SET OF UNIV-COURSES KEY IS
00003850           (CRS-NAME);
00003900   UNIV-PERSONNEL DATA SET(
00004000       NAME GROUP(
00004100           LASTNAME ALPHA(15);
00004200           FIRSTNAME ALPHA(10);
00004300       SEX NUMBER(1);
00004400       AGE NUMBER(2);
00004500       SSNUM NUMBER(9);
00004600       DPT ALPHA(4);
00004700       RANK ALPHA(1);
00004800       SALARY NUMBER(S7.2);
00004900       COURSES SUBSET OF UNIV-COURSES, POPULATION = 8;
00005000       ADDRES SUBSET OF ADR;
00005100       SUPR SUBSET OF UNIV-PERSONNEL);
00005200       SS-U-P ORDERED SET OF UNIV-PERSONNEL KEY IS
00005250           (SSNUM);
00005300       U-P SET ORDERED SET OF UNIV-PERSONNEL KEY IS
00005350           (LASTNAME,FIRSTNAME) DUPLICATES;
00005400   MSF DATA SET(
00005500       SSNO NUMBER(9);
00005600       NONAM NUMBER(1);
00005700       LNAME ALPHA(30);
00005800       MNAME ALPHA(30);
00005900       FNAME ALPHA(30);
```

Figure 4-2. B 1000 Series Data Structure Definition Language (DASDL) (Sheet 1 of 2)

```
00006000        CAMPUS-ADDRESS GROUP(
00006100            DORM ALPHA(6);
00006200            ROOM NUMBER(4);
00006300            POBOX NUMBER(4);
00006400            PHONE NUMBER(7);
00006500        ND NUMBER(2);
00006600        DEGREE ALPHA(4) OCCURS 6 TIMES;
00006700        TOTHRS NUMBER(3);
00006800        TOTOP NUMBER(3);
00006900        GRADE-POINT-AVG NUMBER(3.2);
00007000        MJR NUMBER(3);
00007100        AMJR ALPHA(18);
00007200        SSEX NUMBER(1);
00007300        SAGE NUMBER(2);
00007400        HOME-ADDRESS SUBSET OF ADR;
00007500        QUARTER ORDERED DATA SET(
00007600            QTR ALPHA(4);
00007700            QTTHRS NUMBER(2);
00007800            QTROP NUMBER(2);
00007900            CORSES ORDERED DATA SET(
00008000                TYPECOURSE NUMBER(1);
00008100                YR NUMBER(2);
00008200                Q NUMBER(2);
00008300                GCRS SUBSET OF UNIV-COURSES;
00008400                GGD ALPHA(2);
00008500                TITLE-OF-PAPER ALPHA(30);
00008600                PPRGD ALPHA(2);
00008700                POPULATION = 4;
00008800                CSET ACCESS TO CORSES KEY IS
00008850                    (TYPECOURSE) DUPLICATES)
00009000                POPULATION = 5000;
00009100                QSET ACCESS TO QUARTER KEY IS (QTR));
00009200            MSFSET ORDERED SET OF MSF KEY IS (SSNO);
00009300    ADR DATA SET(
00009400        FACULTY-STUDENT NUMBER(1);
00009500        SNO NUMBER(9);
00009600        ADLN ALPHA(54) OCCURS 9 TIMES
00009700        ZIPC NUMBER(5)
00009800        PHON NUMBER(10);
00009900        SSAD ORDERED SET OF ADR KEY IS (SNO);
00010500    BOOKS(
00010600        AREASIZE = 500,
00010650        TYPE = UNORDERED LIST
00010700        BLOCKSIZE = 5);
00010800    BOOKFILE STORAGE FOR BOOKS(
00010850        TITLE=UNIV/LIBRARY,
00010900        AREAS = 10);
00011000    UNIV-C-SET(
00011100        TABLESIZE = 12;
00011150        AREASIZE = 10,
00011200        TYPE = INDEX SEQUENTIAL,
00011300        LOADFACTOR = 9);
00011400    UNIV-PERSONNEL(
00011450        PRIME,
00011500        POPULATION = 997);
00011600    INITIALIZE;
            $FILE STRUCTURE
```

Figure 4-2. B 1000 Series Data Structure Definition Language (DASDL) (Sheet 2 of 2)

C Take this path if the specified data set is not to be accessed via a set. An embedded, standard data set must be accessed via a set.

Example:

```
INPUT UNIV-PERSONNEL.
```

All members of the data set UNIV-PERSONNEL are obtained.

Example:

```
INPUT UNIV-COURSES(BOOKS VIA BOOKSET).
```

All members of the data set UNIV-COURSES are accessed. For each member of UNIV-COURSES, all related members of the embedded data set BOOKS are accessed.

D Take this path if the specified data set is to be accessed through a spanning set.

Example:

```
INPUT UNIV-COURSES VIA UNIV-C-SET.
```

In this example, all members of the data set UNIV-COURSES are accessed through the spanning set UNIV-C-SET.

Example: ("CLIENT")

```
INPUT CUST-ACCT-INFO(INVOICE-INFO
  VIA INVOICES).
```

All INVOICE-INFO records of the embedded, spanning subset INVOICES are accessed for each CUST-ACCT-INFO record accessed.

E Take this path if retrieval keys are not to be used to access specific members of the data set. In this case, all the data set members within the given set are accessed.

F The AT clause allows selective retrieval of members of the specified data set using the keys associated with the spanning set. A <key condition> must be a valid B 1000 DMS II selection expression for the specified set, including the generalized selection expression.

Example:

```
INPUT UNIV-COURSES VIA UNIV-C-SET AT
   DEPARTMENT = "PY" AND LEVEL = 3 AND
      CRS-NO = 510.
```

The record for the graduate level course Psychology 510 is accessed.

Example:

```
INPUT STUDENTS VIA STUDSET AT
   L-NAME = "JONES".
```

All students whose last name is JONES are accessed.

G Take this path if all records which satisfy the key condition are to be input.

H The WITHOUT DUPLICATES clause suppresses the input of duplicate records which satisfy the key condition.

Example:

```
INPUT STUDENTS VIA STUDSET AT L-NAME = "JONES"
  WITHOUT DUPLICATES.
```

The first STUDENT record with L-NAME equal to JONES is retrieved through the set STUDSET.

B 2000/B 3000/B 4000 SERIES OF SYSTEMS

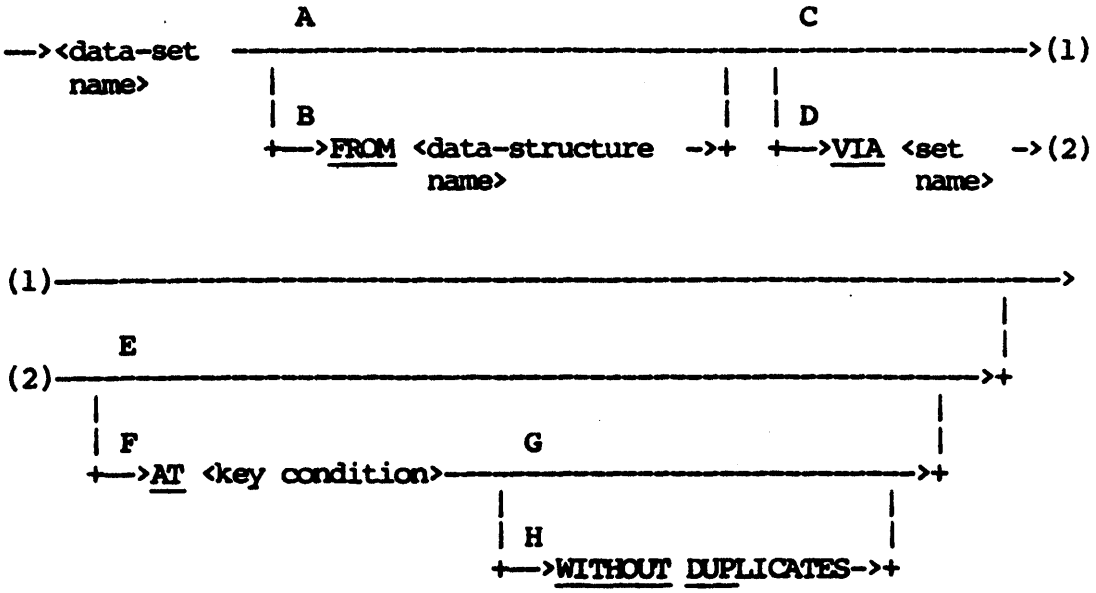
For the B 2000/B 3000/B 4000 Series of Systems, the B 2000/B 3000/B 4000 DMS II Data-structure Clause enables access to a B 2000/B 3000/B 4000 DMS II data set. All data sets referenced in the INPUT statement must belong to the same DMS II data base. You can input embedded data sets, provided the data set in which they are embedded has been specified previously. However, specification of a data set does not cause automatic access to its embedded data sets; to access an embedded data set, you must name it explicitly.

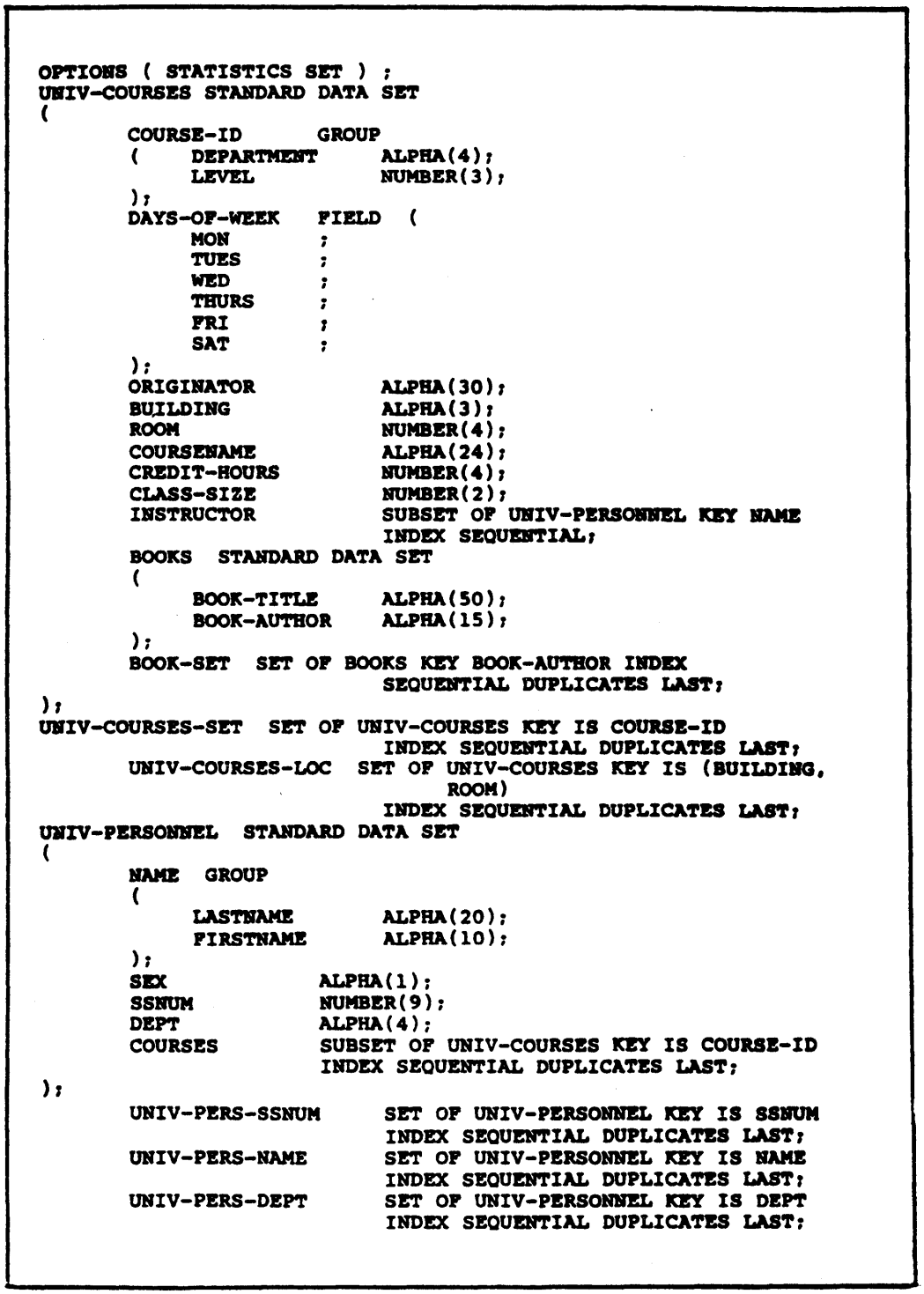
Example:

```
INPUT UNIV-COURSES.
```

A listing of the Data Structure Definition Language (DASDL) used in the example data base is presented in Figure 4-3.

The syntax for the B 2000/B 3000/B 4000 DMS II Data-structure Clause is as follows:





```
OPTIONS ( STATISTICS SET ) ;
UNIV-COURSES STANDARD DATA SET
(
       COURSE-ID      GROUP
       (    DEPARTMENT      ALPHA(4);
            LEVEL           NUMBER(3);
       );
       DAYS-OF-WEEK   FIELD  (
            MON        ;
            TUES       ;
            WED        ;
            THURS      ;
            FRI        ;
            SAT        ;
       );
       ORIGINATOR           ALPHA(30);
       BUILDING             ALPHA(3);
       ROOM                 NUMBER(4);
       COURSENAME           ALPHA(24);
       CREDIT-HOURS         NUMBER(4);
       CLASS-SIZE           NUMBER(2);
       INSTRUCTOR           SUBSET OF UNIV-PERSONNEL KEY NAME
                            INDEX SEQUENTIAL;
       BOOKS  STANDARD DATA SET
       (
            BOOK-TITLE      ALPHA(50);
            BOOK-AUTHOR     ALPHA(15);
       );
       BOOK-SET  SET OF BOOKS KEY BOOK-AUTHOR INDEX
                            SEQUENTIAL DUPLICATES LAST;
);
UNIV-COURSES-SET  SET OF UNIV-COURSES KEY IS COURSE-ID
                            INDEX SEQUENTIAL DUPLICATES LAST;
       UNIV-COURSES-LOC  SET OF UNIV-COURSES KEY IS (BUILDING,
                                 ROOM)
                            INDEX SEQUENTIAL DUPLICATES LAST;
UNIV-PERSONNEL  STANDARD DATA SET
(
       NAME  GROUP
       (
            LASTNAME        ALPHA(20);
            FIRSTNAME       ALPHA(10);
       );
       SEX             ALPHA(1);
       SSNUM           NUMBER(9);
       DEPT            ALPHA(4);
       COURSES         SUBSET OF UNIV-COURSES KEY IS COURSE-ID
                       INDEX SEQUENTIAL DUPLICATES LAST;
);
       UNIV-PERS-SSNUM      SET OF UNIV-PERSONNEL KEY IS SSNUM
                            INDEX SEQUENTIAL DUPLICATES LAST;
       UNIV-PERS-NAME       SET OF UNIV-PERSONNEL KEY IS NAME
                            INDEX SEQUENTIAL DUPLICATES LAST;
       UNIV-PERS-DEPT       SET OF UNIV-PERSONNEL KEY IS DEPT
                            INDEX SEQUENTIAL DUPLICATES LAST;
```

Figure 4-3. B 2000/B 3000/B 4000 Series Data Structure Definition Language (DASDL)

The paths of this syntax diagram are explained below:

Path	Explanation

A Take this path if it is not necessary to clarify the one-to-one dependence of the data set on another input data structure.

B The FROM clause must be used whenever the following two conditions exist:

1. The data set is to be accessed in a one-to-one manner from another data structure.

2. The specified access to the data set depends on the value of a data item in the other data structure.

The name of the data structure containing the data item must be specified. This type of dependence is used when a key in a set spanning the data set is compared against the value of a data item in another data structure.

Example:

```
INPUT UNIV-COURSES, UNIV-PERSONNEL
   FROM UNIV-COURSES VIA UNIV-PERS-NAME
      AT NAME = ORIGINATOR.
```

In this example, UNIV-COURSES and UNIV-PERSONNEL are two disjoint data sets. UNIV-PERS-NAME spans UNIV-PERSONNEL with symbolic key NAME. If ORIGINATOR is a data item of UNIV-COURSES, the FROM clause is needed to indicate the dependence of UNIV-PERSONNEL on the value of ORIGINATOR in UNIV-COURSES. All members of UNIV-COURSES are accessed sequentially; for each UNIV-COURSES record, the corresponding UNIV-PERSONNEL record which satisfies the key condition is accessed.

C Take this path if the specified data set is not to be accessed via a set. An embedded standard data set must be accessed via a set.

Example:

```
INPUT UNIV-PERSONNEL.
```

All members of the data set UNIV-PERSONNEL are input.

Example:

```
INPUT UNIV-COURSES (BOOKS VIA BOOK-SET).
```

All members of the data set UNIV-COURSES are accessed. For each member of UNIV-COURSES, all related members of the embedded data set BOOKS are accessed.

D Take this path if the specified data set is to be accessed through a spanning set.

Example:

```
INPUT UNIV-COURSES VIA UNIV-COURSES-SET.
```

In this example, all members of the data set UNIV-COURSES are accessed using the spanning set UNIV-COURSES-SET.

Example:

```
INPUT CUST-ACCT-INFO (INVOICE-INFO
   VIA INVOICES).
```

All INVOICE-INFO records of the embedded, spanning subset INVOICES are accessed for each CUST-ACCT-INFO record accessed.

E Take this path if retrieval keys are not to be used to access specific members of the data set. In this case, all the data set members with the given set are accessed.

F The AT clause allows selective retrieval of members of the specified data set using keys associated with the spanning set. A <key condition> must be a valid B 2000/B 3000/B 4000 DMS II selection expression for the specified set.

Example:

```
INPUT UNIV-COURSES VIA UNIV-COURSES-LOC AT
   BUILDING = "PSY" AND ROOM = 214.
```

All courses held in PSY 214 are input.

Example:

```
INPUT STUDENTS VIA STUDENT-SET AT
   L-NAME = "JONES".
```

All students whose last name is JONES are accessed.

G Take this path if all records which satisfy the key condition are to be input.

H The WITHOUT DUPLICATES clause suppresses the input of duplicate records which satisfy the key condition.

Example:

```
INPUT STUDENTS VIA STUDENT-SET AT
   L-NAME = "JONES" WITHOUT DUPLICATES.
```

The first STUDENTS record with L-NAME equal to JONES is retrieved using the set STUDENT-SET.

EDITING ATTRIBUTES

The Editing Attributes specify a COBOL editing picture which defines how a data item is to appear when printed. This picture overrides the default editing picture. The editing picture specified is used throughout the report to print the data item value. The COBOL picture must be enclosed in quotes with no leading blanks. It is not syntax-checked; therefore, you must ensure that no COBOL syntax error is generated by the supplied picture. If the editing picture is smaller than the storage picture, truncation can result.

Example:

```
WITH PICTURE "$Z(4)9.99"
```

The associated data item is printed with a COBOL editing picture of $Z(4)9.99.

The syntax for the Editing Attributes specification is as follows:

```
--->WITH PICTURE "<COBOL picture>" --->
```

Note that if the vocabulary being used has the option DECIMAL-POINT IS COMMA set, then the picture must reflect this to ensure proper decimal point alignment and prevent possible syntax errors when compiling the report program.

Example:

```
WITH PIC "Z.ZZ9,99".
```

ENTRY FUNCTION

The ENTRY Function is a nonstatistical function used to convert values of a data item to alternate item values according to a table which was previously defined via a TABLE statement. The value of the function is determined by matching the value of a referenced data item with a table entry and returning the corresponding conversion value defined by the table.

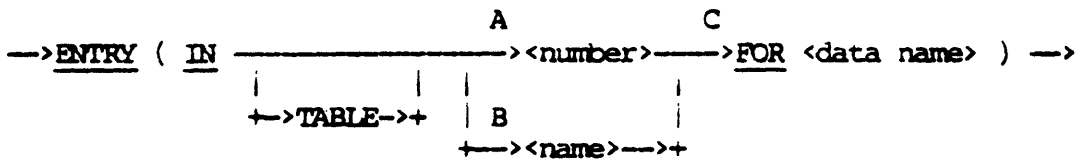
Example:

```
ENTRY (IN TABLE 03 FOR LOC-CODE)
```

Table 03 was previously defined as follows: if LOC-CODE has the value C, the function has the value "CHICAGO".

The item attributes described by the ENTRY Function are determined by the characteristics of the conversion values specified in the referenced table. The conversion values in the example above are defined by the table. They are <string>s having a maximum length of 11 characters. For this reason, the item described by the ENTRY Function is a string-type item with an item value 11 characters in length.

The syntax for the ENTRY Function is as follows:



The paths of this syntax diagram are explained below.

A If a table identified by a <number> is to be used for conversion, it is referenced by taking this path. The <number> must be given exactly as it was in the related TABLE statement.

B If a table identified by a <name> is to be used for conversion, it is referenced by taking this path.

Example:

ENTRY (IN CITIES FOR LOC-CODE)

In this example, the table being referenced must have been named previously as CITIES in a TABLE statement.

Example:

ENTRY (IN TABLE A FOR X)

C Take this path to reference the data item whose value is to be converted based on the referenced table.

Example:

ENTRY (IN TABLE 03 FOR LOC-CODE)

EXPRESSIONS

An <expression> refers to an arithmetic expression, a string expression, or a Boolean expression. An arithmetic expression specifies a numeric value, a string expression specifies a string value, and a Boolean expression specifies a Boolean value (TRUE or FALSE). An expression can be statistical or nonstatistical (refer to STATISTICAL EXPRESSION and NONSTATISTICAL EXPRESSION in this section).

ARITHMETIC EXPRESSION

An arithmetic <expression> in its simplest form is one of the following:

1. A <data name> which references a numeric-value data item.
2. A <number>.
3. An intrinsic function or <ENTRY function> which describes a numeric value.

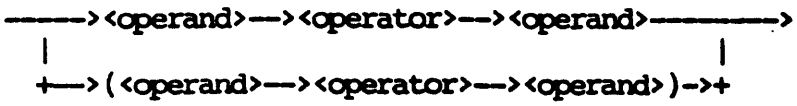
Examples:

```
COST
10
SALARY
AVERAGE(SALARY)
22.57
AGE(FROM INVOICE-DATA TO DATE)
DISC-TABLE OF PART-REC[1,2]
COUNT
```

A basic arithmetic expression describes a numeric operation to be performed on two numeric data items. The two numeric items are called operands. The numeric operation is expressed by one of the following symbolic operators.

Operator	Meaning
+	Addition
-	Subtraction
*	Multiplication
/	Division
**	Exponentiation
MOD	Quotient remainder
DIV	Quotient integer

The syntax for a basic arithmetic expression is as follows:



The arithmetic operators for addition (+) and subtraction (-) must be preceded and followed by at least one blank.

Examples:

```
SALARY/12
QUANTITY*UNIT-COST
THIRTY-DAYS-DUE + SIXTY-DAYS-DUE
(QUANTITY*UNIT-COST)
(X - Y)
A + B * 1.33
AGE (IN WEEKS FROM ORDER-DATE TO DATE) DIV 4
```

The <operand> of an arithmetic expression can be a Boolean expression (refer to the discussion under the heading BOOLEAN EXPRESSION). When this is the case, the value of the Boolean expression is interpreted as 1 if it is TRUE, and 0 if it is FALSE. The Boolean expression must be enclosed in parentheses.

Examples:

```
5*(DEGREE = BS)
CREDIT IS AMOUNT * (AMOUNT > 0)
```

A complex arithmetic expression results when the <operand> of a basic arithmetic expression is itself a basic arithmetic expression. That is, basic arithmetic operations can be combined into a complex arithmetic expression.

Examples:

```
THIRTY-DAYS-DUE + SIXTY-DAYS-DUE + NINETY-DAYS-DUE
MONTHLY-SALARY * 12 + BENEFITS
(X - Y)*Z
10 * YEARS-OF-EXPERIENCE + 5 * (DEGREE = BS) +
  10 * (DEGREE = MS)
(TOTAL(SALARY) + TOTAL(BENEFITS)) / 12
TOTAL(SALARY FOR EACH DEPT)/BUDGET REL TO DEPT * 100
```

Parentheses can be used in a complex arithmetic expression to ensure the proper sequence of the operations. Operations within innermost parentheses are performed first. If parentheses are not used, the arithmetic operators have an assigned hierarchy which determines the sequence in which the operations are performed. Operators of highest hierarchy are performed first. Operators having the same hierarchy are performed in the order in which they are specified (left to right). The hierarchy of arithmetic operators is as follows:

Hierarchy		Symbol	Definition
Highest	1	**	Exponentiation
	2	*	Multiplication
		/	Division
		MOD	Quotient remainder
		DIV	Quotient integer
Lowest	3	+	Addition
		-	Subtraction

The following two examples are equivalent:

```
A + B - C * ((D - 4) * E)/3 -X

((A + B) - C * ((D - 4) * E)) - X
           ----------------
                  3
```

STRING EXPRESSION

A string <expression> is one of the following:

1. A <data name> which references a string-valued data item.
2. A <string>.
3. An intrinsic function which describes a string value.

Examples:

```
CUSTOMER-NAME
"LBS."
PART-DESC
"          "
JANUARY-DATE(ORDER-DATE)
MATERIAL-SOURCE-CODE
```

BOOLEAN EXPRESSION

A Boolean expression, <Bool expr>, describes a condition which can be evaluated as TRUE (numeric value 1) or FALSE (numeric value 0).

Simple Boolean Expression

A Boolean expression in its simplest form is a <data name> which references a Boolean item.

Example:

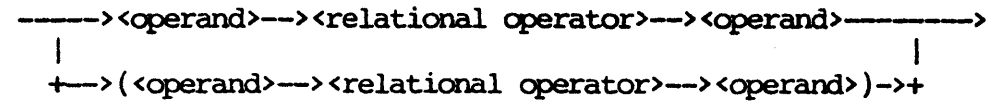
```
LARGE-ACCOUNT
```

LARGE-ACCOUNT is a Boolean-type item. It can be an accepted or derived item defined as BOOLEAN or a DMS II Boolean vocabulary item.

Basic Boolean Expression

A basic Boolean expression compares two data items or specifies a special test to be performed on the value of a data item. Only data items of the same type (that is, numeric, string, or Boolean) may be compared.

The syntax of a basic Boolean expression is as follows:



In most cases, the left <operand> is a <data name>, and the right <operand> is a <data name> or <literal>.

Examples:

```
COST > 1000.00
BRANCH-NUMBER = ACCEPTED-BR-NO
PART-NO = 103888
CREDIT-LIMIT GREATER THAN 5000
CUSTOMER-NAME = "JOHN Q. DOE"
```

In general, either <operand> can be an arithmetic <expression>.

Examples:

```
BALANCE-DUE = (CURRENTLY-DUE + THIRTY-DAYS-DUE
   + SIXTY-DAYS-DUE + NINETY-DAYS-DUE)
(QUANTITY*UNIT-COST) EQUAL INVENT-DOLLAR-VALUE
(X/Y + Z) LEQ 10
AGE(FROM INVOICE-DATE TO DATE) > 90
AVERAGE(SALARY) < 10000
```

The values TRUE, FALSE, and NULL are figurative constants which can be used in certain instances as the right <operand>. TRUE and FALSE can be used to test Boolean items for value 1 and value 0, respectively; however, use of a simple Boolean expression is recommended. NULL is used to check DMS II items for a NULL condition. When a figurative constant is used, the <relational operator> must specify an equal or not-equal relation.

Examples:

```
FLAG-BIT = TRUE (equivalent to:  FLAG-BIT)
STUD-NO NOT = NULL
EXCEPTION-FLAG IS EQUAL FALSE
```

A <condition name> can be used as the right <operand> to perform conveniently value tests on certain data items. The left <operand> must be the <data name> for which the condition was defined. The <relational operator> must specify an equal or not-equal relation.

Examples:

```
SEX = MALE
COLOR = RED OF COLOR
```

ALPHABETIC or NUMERIC can be specified as the right <operand> in order to perform a class test on the data item specified by the left <operand>. The NUMERIC class test determines if the item value consists entirely of the characters 0, 1, 2, 3, 4, ..., 9, with or without an operational sign. The ALPHABETIC class test determines if the item value consists entirely of the characters A, B, C, ..., Z, and space. The NUMERIC class test cannot be used on a string-type item. The ALPHABETIC class test cannot be used on a numeric-type item. The <relational operator> for a class test must specify an equal or not-equal relation.

Examples:

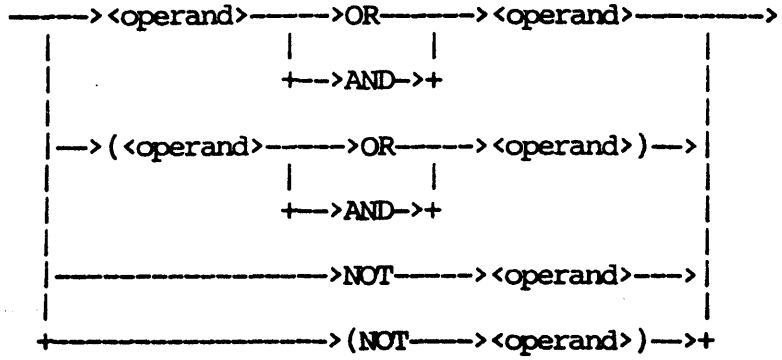
```
PART-NO = NUMERIC
CUSTOMER-NAME NOT = ALPHABETIC
```

The right <operand> of a basic Boolean expression can be a pattern. A discussion of pattern matching is contained under the heading "Pattern Matching," which immediately follows the discussion under "Complex Boolean Expression" below.

Complex Boolean Expression

A complex Boolean expression describes logical operations to be performed on simple or basic Boolean expressions. The logical operations are NOT (negation), AND (conjunction), and OR (disjunction).

The syntax for a complex Boolean expression is as follows:



An <operand> can be a basic Boolean expression, or it can itself be a complex Boolean expression.

Examples:

```
SEX = MALE AND AGE > 18
STRATA-1 OR STRATA-2
(BALANCE-DUE > 5000.00) AND (BALANCE-DUE < 10000.00)
[(X > A + B) AND Y < 10] OR Z = 20
```

A complex Boolean expression can involve logical, relational, and arithmetic operators. Parentheses can be used to ensure the proper sequence of operations. If parentheses are not used, the following assigned hierarchy of operators determines the sequence in which the operations are performed. Operators of highest hierarchy are performed first. Operators having the same hierarchy are performed in the order in which they are specified (that is, from left to right). The hierarchy of arithmetic operators is as shown in the following list.

Hierarchy	Symbol	Definition
Highest 1	**	Exponentiation
2	*	Multiply
	/	Divide
	MOD	Quotient remainder
	DIV	Quotient integer
3	+	Add
	-	Subtract
4		All relational operators
5	NOT	Negation
6	AND	Conjunction
Lowest 7	OR	Disjunction

Example:

```
A > B + C AND D < 10
```

is equivalent to:

```
[A > (B + C)] AND (D < 10)
```

Pattern Matching

The right <operand> of a basic Boolean expression can be a pattern. A pattern describes a pattern of characters to be matched against a string value. Only string-type data items can be tested for pattern match, and only tests for equal (=) or not equal (NOT =) are allowed. All patterns are enclosed in brackets. A pattern can be a maximum of 55 characters in length. The various forms of patterns are specified as follows:

1. A single specific string is specified in quotation marks inside the brackets. This indicates that the significant characters of the item value must match this string. Trailing blanks in the item value are not considered as significant.

Example:

```
WORD = ["SAMPLE"]
```

The pattern match in this example indicates that the significant characters of WORD must be SAMPLE.

2. A hyphen (-) in a pattern indicates that any number of characters may be present.

Example:

```
WORD = ["S" - "M"]
```

The pattern match in this example indicates that the first character of WORD is an S, and the character M is the last significant or nonblank character in the item value.

Example:

```
WORD = ["S" - "M" -]
```

The pattern match in this example indicates that the first character of the item value is an S, and that this first character is followed by an M somewhere within the string value.

3. A single number in a pattern specifies the exact number of characters which are ignored in the item value after the first string and before the next string.

Example:

```
WORD = ["S" 1 "M" -]
```

The pattern match in this example indicates that the first character of WORD is an S and the third character is M.

4. Two numbers separated by a comma represent the minimum and the maximum number, respectively, of characters which are ignored in the item value after the first string and before the next string.

Example:

```
WORD = ["S" 1, 3 "M"]
```

The pattern match in this example indicates the following: the first character of WORD is S, and at least one but not more than three characters must exist before M. The following values represent match and no-match conditions for the example:

Match	No-Match
SAM	SHERMAN
SLIM	HIM
STEAM	STREAM

5. A number of specifications of the type described in items 2, 3, and 4 above may be given in one pattern.

Example:

```
VEHICLE-LICENSE = ["L" 1 "J" - "6" -]
```

The pattern match in this example indicates the following: the first character is L, the third character is J, and a 6 appears elsewhere in the vehicle license number.

EXTENSION

An Extension (also known as a derived data item) is used to define a new data item to be derived from other data items. The derived data item is calculated for each logical record and "extends" the logical record with the new data value. Once a derived data item is defined, you can reference it subsequently where appropriate in the report language statements.

Example: ("INVENT")

```
PART-SALES-COST IS UNIT-COST * QUANTITY-SOLD
```

This describes a new data item PART-SALES-COST which is calculated for each part.

The <name> in the Extension clause is defined by the user. It must be unique for a specified report, that is, it must not duplicate any <name> defined in the vocabulary, or any other derived data item. You can use this <name> as a <data name> in other statements to reference the derived data item after the <name> has been defined in the Extension clause.

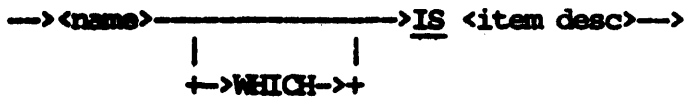
The <item desc> clause describes various aspects of the derived data item. It specifies what the value of the extension will be composed of, what the physical attributes of the extension will be, and what format will be used when the item is printed.

An Extension can be part of an Extension statement or it can be part of another report language statement. The following illustrates an Extension specified in a REPORT statement:

Example: ("CLIENT")

```
REPORT CUST-NO, OVERDRAWN WHICH IS
   BALANCE-DUE - CREDIT-LIMIT NUM(5,2)
      WITH PICTURE "Z(4)9.99".
```

The syntax for the Extension clause is as follows:



EXTENSION STATEMENT

An Extension statement is used to define an <extension> (a derived data item) independent of its use in other report language statements.

Example:

```
PART-SALES-COST IS UNIT-COST * QUANTITY-SOLD.
```

The above Extension statement defines the derived data item, PARTS-SALES-COST. This item now can be referenced in other report language statements.

The syntax for the Extension statement is as follows:

```
------------------------> <extension>.
```

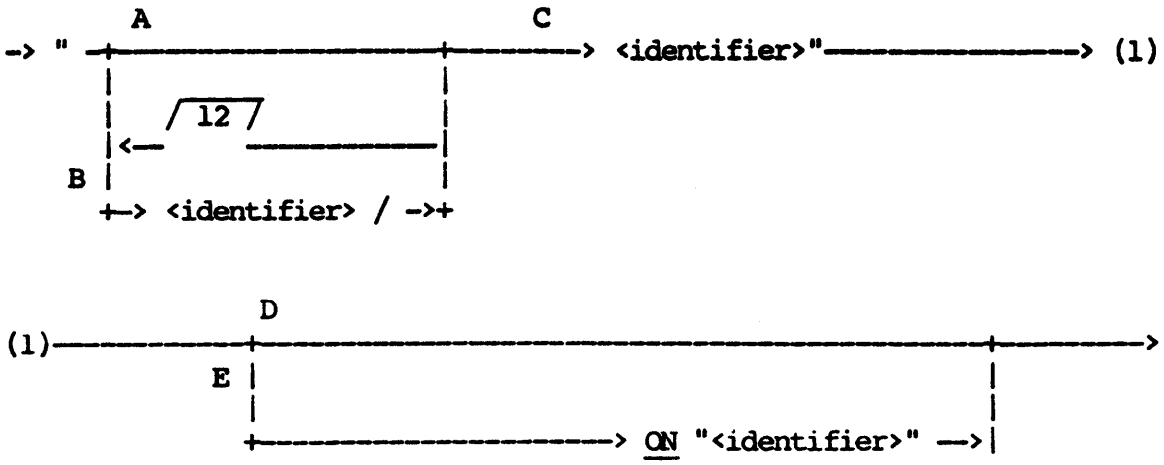
EXTERNAL FILE NAME

An External File Name identifies a file to the MCP. It contains <string>(s) used as <identifier>(s). Optionally, it also can contain a disk pack (cartridge) <identifier> clause.

A SERIES OF SYSTEMS

For the A Series of Systems, an External File Name contains a series of <identifier>s (file <identifier>, file directory <identifier>s, and volume <identifier>s). An optional disk pack name can be specified in an External File Name in certain contexts. An <identifier> can be up to 17 characters long. A maximum of 30 characters (excluding the optional ON clause) is permitted for an External File Name, including slashes.

The syntax of External File Name for the A Series of Systems is as follows:



The paths of this syntax diagram are explained below:

Path	Explanation
A	Take this path if no file directory <identifier>s and no volume <identifier>s are required.
B	Take this path to supply file directory <identifier>s and optional volume <identifier>s.

C Take this path to specify the <identifier> which represents the file name.

D Take this path if no pack name is to be specified.

Examples:

"VOCAB1"
"SAVINGS/NAMEANDADDRESS"
"A/B/C/D/E/F"

E Take this path to specify a pack name. The ON option can only be used in the EXTRACT statement, VOCABULARY statement, and the process option ASSIGN ACCEPTED-DATA TO DISK statement. Other file assignments to disk pack can be done via proper file equation (refer to the A Series Work Flow Language (WFL) Reference Manual).

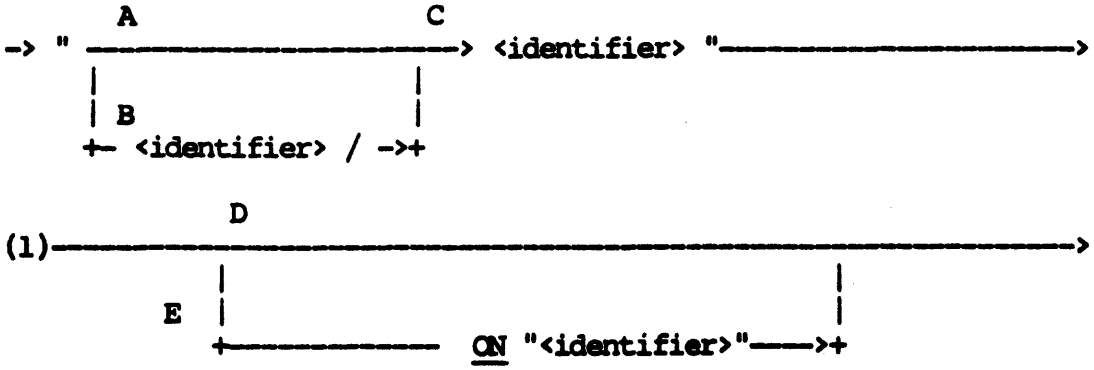
Example:

"PERSONNEL/FILE" ON "EMPLOYEES"

B 1000 SERIES OF SYSTEMS

For the B 1000 Series of Systems, an External File Name contains up to three <identifier>s. An <identifier> can be up to 10 characters long. The use of special characters as part of the <identifier> is not recommended. (Do not use periods in the <external file name> part of a SAVE OBJECT AS clause.)

The syntax of External File Name for the B 1000 Series of Systems is as follows:



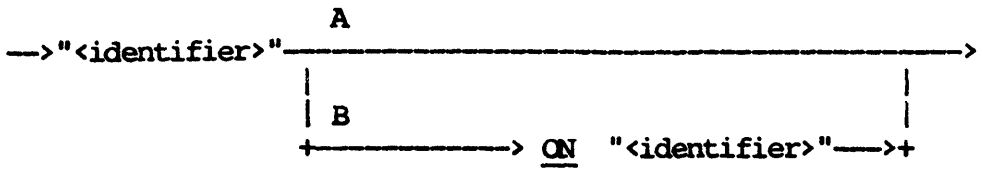
The paths of this syntax diagram are explained below:

Path	Explanation
A	Take this path if no family name is to be specified.
B	Take this path to specify an <identifier> which represents the family name.
C	Take this path to specify the <identifier> which represents the file name.
D	Take this path if no pack-ID is to be specified. Example: "VOCAB1" "PAYROLL10/TSTVCB"
E	Take this path to specify the <identifier> which represents the pack-ID. The pack-ID cannot be used with any of the following specifications: SAVE REPORT, SAVE FORMS, and SAVE EXCEPTIONS LISTING. Examples: "VOCAB" ON "SYSTEMFILE" "EMPLOYEES/REPORT" ON "PAYROLL-74"

B 2000/B 3000/B 4000 SERIES OF SYSTEMS

For the B 2000/B 3000/B 4000 Series of Systems, an External File Name contains one or two <identifier>s. An <identifier> can be up to six characters long and cannot contain any special characters (space, comma, period, slash, hyphen, and semicolon). The <identifier> specified first is the file name.

The syntax of External File Name for the B 2000/B 3000/B 4000 Series of Systems is as follows:



The paths of this syntax diagram are explained below:

Path	Explanation
A	Take this path if no PACK-ID is to be specified. Examples: "VOCAB1" "12JA74"
B	Take this path to specify the <identifier> which represents the PACK-ID. A PACK-ID can only be used in the EXTRACT statement, VOCABULARY statement, and the process option ASSIGN ACCEPTED-DATA TO DISK statement. Example: "STAFF" ON "SYSTEM"

EXTRACT-ITEM DESC

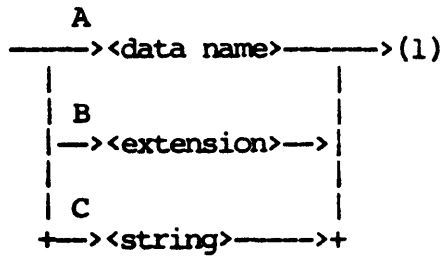
The Extract-item Desc is used to specify a data item to be written to the extract file and the manner in which it is to be written.

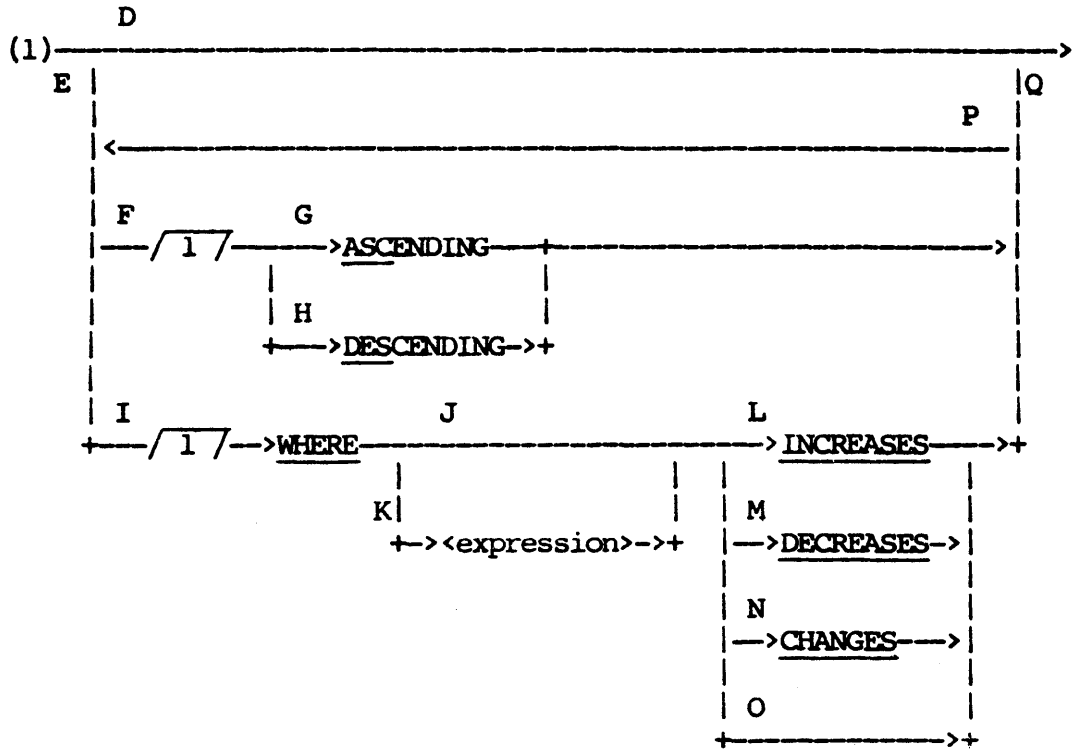
Example: ("INVENT")

```
EXTRACT PART-NO, PART-DESC, QUANTITY TO EF-FILE.
```

In this example, PART-NO, PART-DESC, and QUANTITY specify which data item is to be extracted.

The syntax for Extract-item Desc is as follows:





The paths of this syntax diagram are explained below:

Path	Explanation
A	Take this path to specify the name of the data item to be extracted. Subscripting cannot be used if EXTRACT WITH VOCABULARY has been specified previously. You can use an extension to extract a subscripted data name (refer to path B). The same <data name> cannot be extracted to the same file more than once. An extension can be used for the second and subsequent occurrences of the <data name>.
B	Take this path to define an <extension> and to specify that its value is to be extracted.

Example: ("CLIENT")

```
EXTRACT CUSTOMER-NAME, CREDIT-CODE IS
   CREDIT-LIMIT/100 NUM(2) TO EF-CREDIT.
```

The derived data item, CREDIT-CODE, is extracted as the second field of the extract record. The value of the <extension>, CREDIT-LIMIT/100, is written to the file with a storage picture of 9(2) COMPUTATIONAL.

Example: ("SHIPV")

```
EXTRACT NUMBER OF BILL, FIRST-POINT IS
   POINTS-OF-SHIPMENT [1] TO EF-NEW.
```

In this example, the first element of the subscripted data item, POINTS-OF-SHIPMENT, is extracted as FIRST-POINT.

NOTE

Use of subscripting in this example is allowed because the EXTRACT WITH VOCABULARY was not used.

C Take this path to specify a <string> to be extracted on every record of the file. The <string> is given an arbitrary name and an internal storage picture to match the length specified. The maximum length allowed for an extracted <string> is 30 characters.

Example: ("CLIENT")

```
EXTRACT CUSTOMER-NAME, "    ", BALANCE-DUE
   TO EF-NEW.
```

The second field on each record is four blank characters.

D Take this path if all data item values are to be extracted and the information to be extracted requires no ordering based on this item.

E Take this path if not all values of the data item are to be extracted or if the values of this data item are to be extracted in a particular order.

F Take this path to specify that the information to be extracted is to be ordered based on this item. You cannot take this path if the item is statistical or if the FOR EACH clause of the EXTRACT statement has been taken previously. If ordering is specified for more than one extract item, the extract item for which ordering is specified first is considered the major ordering key. The ordering specified here is subordinate to any grouping of information via control breaks. In addition, the ordering must be consistent with the ordering specified elsewhere.

G Take this path to specify that the values of this data item be extracted in ascending order.

Example: ("CLIENT")

```
EXTRACT CUSTOMER-NAME ASCENDING, CREDIT-LIMIT
   TO EF-FILE.
```

In this example, records are extracted to the EF-FILE so that the customer names are in alphabetic order.

H Take this path to specify that the values of this data item be extracted in descending order.

Example: ("CLIENT")

```
EXTRACT CREDIT-LIMIT DESCENDING, CUSTOMER-NAME
   TO EF-FILE.
```

In this example, records are extracted to the EF-FILE such that the credit limits are in order, largest to smallest.

I Take this path to specify a condition under which the values of the data item are to be extracted. If the condition evaluates FALSE, NULL-VALUES is extracted instead of the data item value. If the condition evaluates TRUE, the data item value is extracted. The expression is evaluated for each value of the data item.

Example: ("CLIENT")

```
EXTRACT CUSTOMER-NAME, CREDIT-LIMIT
   WHERE CREDIT-LIMIT < 10000 TO EF-FILE.
```

In this example, all the customer names will be extracted. The underlined portion specifies that the CREDIT-LIMIT data item value is to be placed into the extract record only if the value is less than $10,000. If it is greater than or equal to $10,000, NULL-VALUES is extracted instead.

J Take this path as a shorthand method to specify implicitly an <expression> whose value is identical to the data item being extracted. This path must not be used with path O. Path L, M, or N must be used to qualify the data value to form a condition.

Example: ("INVENT")

```
EXTRACT PART-DESC WHERE CHANGES, QUANTITY-SOLD
   TO EF-SALES.
```

In this example, only new values of PART-DESC are written to the extract files. If the value of PART-DESC is identical to the previous value, null-values are extracted.

K Take this path to specify an <expression> used in defining the conditions for extracting the data item. A Boolean <expression> can be given to define the condition, or a string or numeric <expression> can be used in conjunction with paths L through N to define the condition.

Example: ("CLIENT")

```
EXTRACT CUSTOMER-NAME, CREDIT-LIMIT WHERE
   CREDIT-LIMIT < 10000 TO EF-FILE.
```

In this example, a Boolean <expression> is given which completely defines the condition.

L Take this path in conjunction with a non-Boolean <expression> to specify that the data value be extracted only if the value of the <expression> increases.

M Take this path in conjunction with a non-Boolean <expression> to specify that the data value be extracted only if the value of the <expression> decreases.

N Take this path in conjunction with a non-Boolean <expression> to specify that the data value be extracted only if the value of the <expression> has changed.

O Take this path only if the <expression> given in path K is Boolean. A Boolean <expression is sufficient to express the condition. This path is illegal if an arithmetic or string <expression> is specified in path K. The example for path K illustrates the use of this option with a Boolean <expression>.

P Take this path to specify both ordering, based on the item, and conditional extraction. Conditional extraction of an item occurs based on the ordering of the data values as specified.

Example: ("CLIENT")

```
EXTRACT CUSTOMER-NAME ASCENDING WHERE CHANGES
   TO EF-FILE.
```

In this example, duplicate customer names are grouped together via the ASCENDING specification. Then all duplicate names except the first are nulled by the WHERE CHANGES clause.

Q Take this path after ordering and/or conditional extraction are specified.

EXTRACT STATEMENT

The EXTRACT statement is used to extract information to a machine-readable file for subsequent processing. The file can be a magnetic tape, disk, disk pack, or card file. Extracted information can be used subsequently to generate confirmations, print mailing labels, and so forth. In addition, the information extracted can be reported using the REPORTER III System. To assist in this, a vocabulary describing the extracted information optionally can be produced automatically. Only reportable information is extracted.

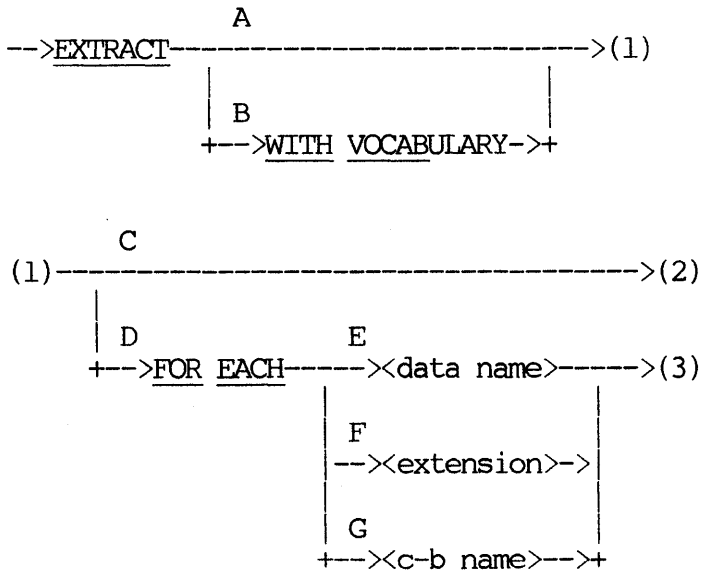
Example: ("VOCAST")

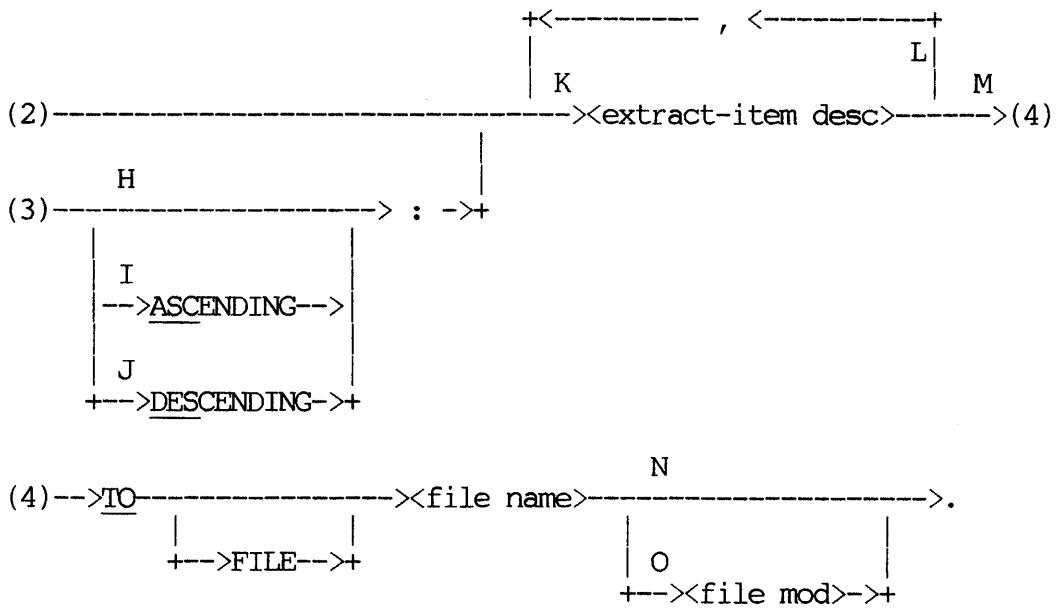
```
    EXTRACT ASSET-NO, ASSET-DESC, COST TO EF-FILE.
```

A disk file called EF-FILE is created with each record containing the asset number, asset description, and cost of a reported asset. The data items are written to the file using their internal storage picture.

A detailed example of EXTRACT statements used in report specifications for inventory purposes is presented later (after the explanations of the paths in the syntax diagram).

The syntax for the EXTRACT statement is as follows:





The paths of this syntax diagram are explained below:

Path	Explanation

A Take this path if the ability to report information from the extract file using the REPORTER III System is not required or if a vocabulary for this file is to be built independently.

Example: ("CLIENT")

```
EXTRACT CUSTOMER-NAME, STREET-ADDRESS,
   CITY-STATE, ZIP-CODE TO MAILING-LABELS.
```

This information is extracted to a file which is input to a program that creates mailing labels.

B If you take this path, RP3VOC specifications are required for the extract file. RP3VOC is executed, thus creating an extract vocabulary which can be used to report the information in the extracted file. The Process-option SAVE statement can be used to assign names to the extract vocabulary files. If the original vocabulary requires a password, the same password is required for the extract vocabulary. Only one extract vocabulary is created for all files extracted in one report specification.

If the original vocabulary has the statement "SET LANGUAGE TO COBOL85", the same statement is added to the extract vocabulary.

Example: ("VOCAST")

```
EXTRACT WITH VOCABULARY ASSET-NO, ASSET-DESC,
   COST TO EF-FILE.
```

This example is nearly identical to the initial example given for the EXTRACT statement. The only difference is that the extract vocabulary statement in this example is built, which includes a description of the file EF-FILE. The information within the file then can be reported subsequently using the REPORTER III System.

C Take this path if one record is to be written to the extract file for each logical record reported. The example for path B illustrates this option.

D Take this path to specify that a record of summary information be written to the extract file for each value of a control-break item. The specified control-break item is the first item written to each record of the extract file.

Summary information for a control-break item consists of items which are related it and have a constant value for each of its values. In many instances, these items are derived statistical items; but they also can be nonstatistical items. If path D is taken, all items extracted must be summary items for the specified control break. Nonstatistical items which are extracted are implicitly declared as summary items for the specified control break.

Path D defines the default scope for any statistical functions which are specified in subsequent <extract-item desc>s (refer to path K).

Example: ("VOCAST")

```
EXTRACT FOR EACH DEPT-NO: TOTAL-ASSET-VALUE IS
   TOTAL(COST) TO DEPT-FILE.
```

For each value of DEPT-NO, a summary record is written containing the department number and the total cost of all department assets. If control breaks have not been specified previously, the above statement defines DEPT-NO as a control-break name. Information is grouped accordingly.

E Take this path to specify that summary information be written to an extract record for each value of a data item referenced by <data name>. The item must be nonstatistical and is defined to be a control-break item by its reference here. Control breaks

for the report must not have been specified previously in another statement. (Refer to the example for path D.)

F Take this path to specify that summary information be written to an extract record for each value of a data item defined by <extension>. The derived item must be nonstatistical. The item is defined to be a control-break item; control breaks for the report must not have been specified previously in another statement.

G Take this path to specify that summary information be written to an extract record for each value of a previously defined control-break item; <c-b name> must reference this item.

Example: ("INVENT")

```
RANGE BY INVENT-DOLLAR-VALUE FROM 0 BY 1000.
   .
   .
   .
EXTRACT FOR EACH RANGE:  NO-OF-PARTS IS COUNT,
   TOTAL-VALUE IS TOTAL (INVENT-DOLLAR-VALUE),
      AVG-VALUE IS AVG(INVENT-DOLLAR-VALUE)
      TO FILE INVENT-SUMMARIES.
```

RANGE references a previously-defined control-break item. A summary record is written to the extract file for each range-break grouping.

H Take this path when you desire no particular ordering based on the control-break item, or if you have already specified this ordering.

You must take this path if you have taken path G and <c-b name> references a range-break item. Unless suppressed, a sort is done to properly group the logical records based on the control-break item. The default ordering of information is ascending based on the control-break item.

I Take this path to specify explicitly that the summary records be written to the extract file in ascending sequence based on control-break item values.

Example: ("VOCAST")

```
EXTRACT FOR EACH DEPT-NO ASC:
   TOTAL-ASSET-VALUE IS TOTAL(COST)
      TO DEPT-FILE.
```

Information is extracted in ascending sequence of department numbers.

J Take this path to specify that the summary records be written to the extract file in descending sequence based on control-break item values.

Example: ("VOCAST")

```
EXTRACT FOR EACH DEPT-NO DESC: TOTAL-ASSET-VALUE
   IS TOTAL(COST) TO DEPT-FILE.
```

Information is extracted in descending department number sequence.

K An <extract-item desc> specifies a data item to be written into one field of the extract file record. The data item is written in its storage picture format. If you have taken path B, you make an entry in the extract vocabulary for this item. The entry includes the item name, if any, and its internal and editing attributes.

Example: ("INVENT")

```
EXTRACT PART-DESC TO EF-SELECTED-PARTS.
```

This specifies a record containing only the item PART-DESC. Each value for PART-DESC is written on a separate record in the disk file EF-SELECTED-PARTS.

L Take this path as many times as necessary to specify all items to be extracted to a particular file. Any number of items can be extracted to a file. A record for the file contains all items extracted in the same order in which they were specified in the EXTRACT statement.

Example: ("INVENT")

```
EXTRACT PART-NO, PART-DESC, QUANTITY
   TO EF-FILE.
```

Each record in EF-FILE contains a field for PART-NO, PART-DESC, and QUANTITY, in that order.

M Take this path after you have specified all data items to be extracted. <File name> specifies the internal file name of the file which contains the extracted data records. This file name

is entered into the extract vocabulary if you have taken path B, and it is used to reference the file being extracted in other report specifications and/or vocabulary specifications. It is recommended that the internal file name be prefixed "EF-" to ensure that no COBOL reserved words are used illegally. If this prefix is not used, a warning message is issued.

Example: ("INVENT")

```
EXTRACT PART-NO, PART-DESC,
   QUANTITY TO EF-FILE.
```

EF-FILE is the internal name for the extract file (the external name is also EF-FILE by default).

N If you take this path, default attributes are assigned to the extract file. The default file attributes are the following:

1. The external file name is identical to the internal file name specified in path M above.
2. The hardware device for the file is disk.
3. The blocking factor is 10 records per block for a non-PUNCH file and one record per block for a PUNCH file.
4. The assumed file total population is 9999.

O Take this path to specify nondefault file attributes. The nondefault attributes which can be specified are the following:

1. External file name.
2. Hardware device.
3. Blocking factor.
4. File total population.

Example: ("INVENT")

```
EXTRACT PART-NO, QUANTITY TO EF-CONTROL;
   HARDWARE IS PUNCH.
```

In this case, the records are extracted to a card punch instead of disk.

EXAMPLE OF EXTRACT STATEMENTS

The following is an example of a REPORTER III report specification which could be used for a year-end inventory check:

```
      .
      .
SAVE EXTRACT-VOCAB AS "INVTR1" AND "INVTR2".
INPUT INVENT.
EXTRACT WITH VOCAB PART-NO, "   ", BIN-NO ASC, "   ",
   QUANTITY, "   ", ACTUAL-QUANTITY IS 0 NUM(5)
      WITH PIC "ZZZZZ" TO CARD-CONFIRM HARDWARE IS
      PUNCH ID IS "CONINV".
      .
      .
```

The "SAVE EXTRACT-VOCAB AS" is used to assign names to the vocabulary files created by the "EXTRACT WITH VOCAB". If the SAVE statement were not used, the names "V1nn" and "V2nn" would be used, where "nn" is a random number generated by RP3REP at run time.

A string of 3 spaces is specified between PART-NO, BIN-NO ASC, QUANTITY, and ACTUAL-QUANTITY. The extracted output will be in BIN-NO ascending order. ACTUAL-QUANTITY is an <extension>, and is being used as a place holder for the cards to be punched. ACTUAL-QUANTITY size "NUM(5)" will be kept in the extracted vocabulary for later use. The <file name> CARD-CONFIRM will become the <data-structure clause> in the INPUT statement for the next specification.

The cards punched for the above specification will be in bin number order and could be used to verify the quantity and part. The space available in the area of ACTUAL-QUANTITY could be used to punch in the actual number of parts found. These cards then could be used for input information to the following specification:

```
VOCAB IS "INVTR1".
INPUT CARD-CONFIRM.
REPORT PART-NO WHERE QUANTITY NEQ ACTUAL-QUANTITY,
   BIN-NO, QUANTITY AS "QUANTITY ON FILE",
      ACTUAL-QUANTITY.
EXTRACT PART-NO, BIN-NO, QUANTITY,
   ACTUAL-QUANTITY TO INVENTORY-INFO HARDWARE IS
      DISK ID IS "INVINF".
```

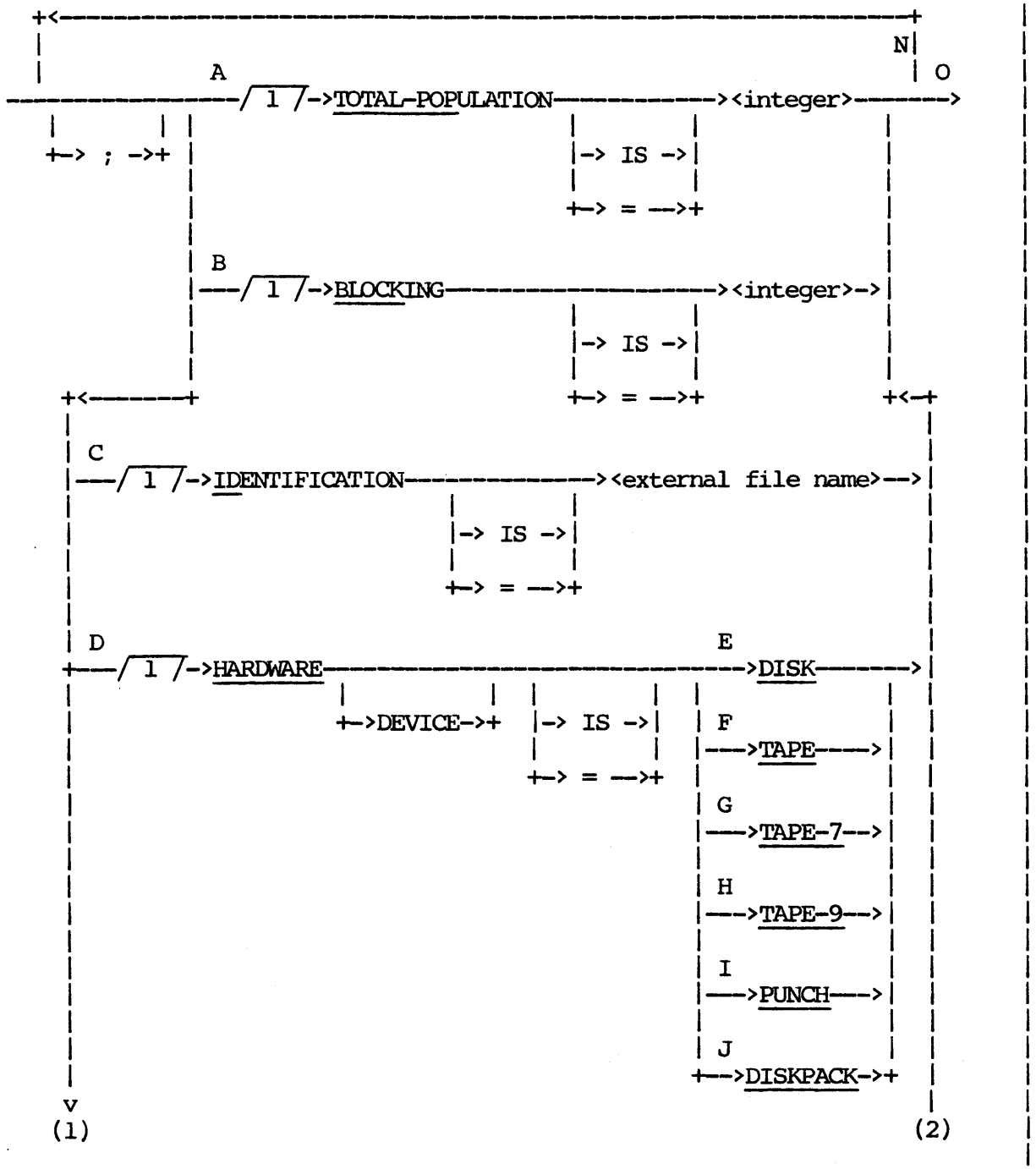
The above specification uses the vocabulary created by processing of the previous specification. The <file name> of the previous specification's EXTRACT statement is used as the <data-structure clause> of the INPUT statement of the above specification. A report will be created listing only those part numbers whose actual quantity did not equal the quantity on file.

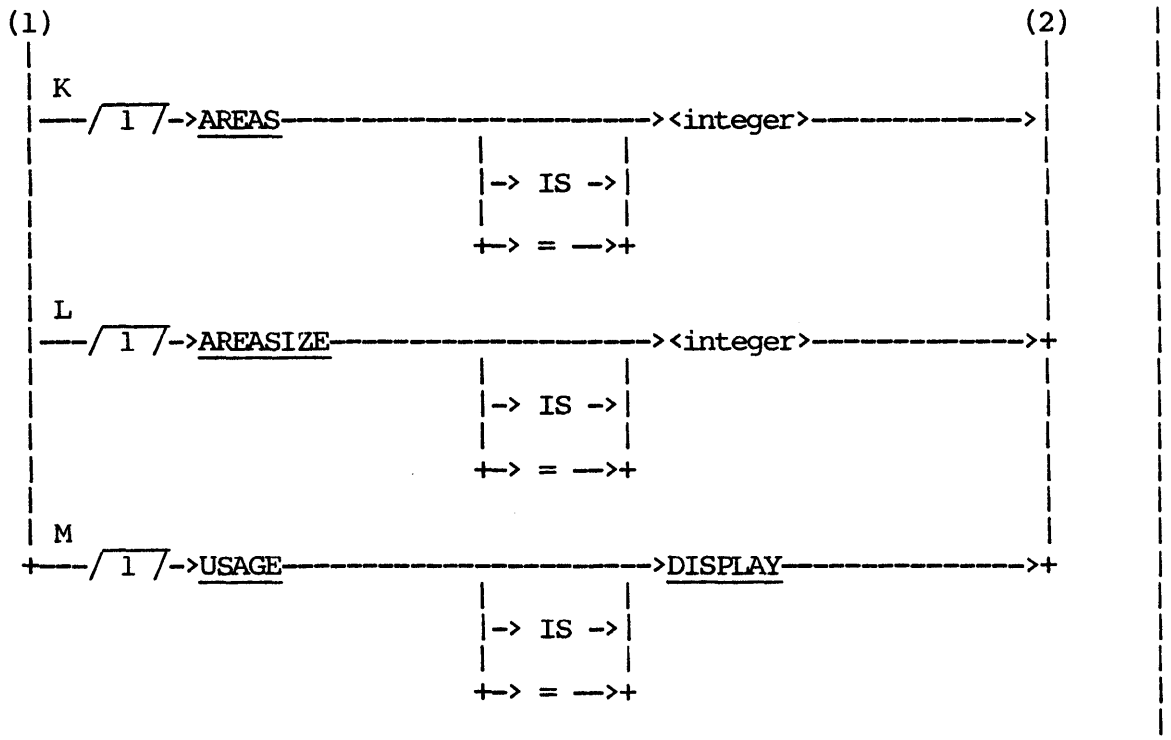
The EXTRACT statement in the above specification shows how the information read from the cards could be put into a disk file. This information then could be used by another program to make any corrections to the master files.

FILE MOD

File Mod is used to specify nondefault attributes for the extract file.

The syntax for File Mod is as follows:





The paths of this syntax diagram are explained below:

Path	Explanation

A This path is taken to specify a total population attribute for the file and is relevant only if WITH VOCABULARY has been specified previously. It does not limit the file in any way. The total population attribute is used when reporting on the extracted file to calculate the number of spaces to allow for printing statistical summaries. If you do not specify an integer, the total population of the extracted file defaults to 9999.

Example:

```
EXTRACT CUSTOMER-NAME TO EF-NEW;
   TOTAL-POP = 1000.
```

The EF-NEW file is declared to contain 1000 records. If EF-NEW is reported on, four spaces are allocated for COUNT.

B Take this path to specify a blocking factor for the extract file. This has meaning only for DISK, TAPE, or DISKPACK files. The default blocking factor is 10 for these files.

Example:

```
EXTRACT CUSTOMER-NAME, BALANCE-DUE TO
   EF-CONFIRM; BLOCKING IS 5.
```

C Take this path to specify a different external name for the file. The default external name is the same as the internal name given in the "TO" clause of the EXTRACT statement (refer to the EXTRACT statement in this section). The name specified in the example below is the one that identifies the file to the MCP. If DISKPACK is chosen as the hardware device, you must follow this path to give the pack identifier to the file.

Example:

```
EXTRACT CUSTOMER-NAME, BALANCE-DUE TO
   EF-CONFIRM ID IS "BALDUE".
```

File EF-CONFIRM is known externally as "BALDUE".

D Take this path to specify the hardware device for the extracted file. The default is DISK if no hardware specification is given.

E Take this path to document explicitly the default of DISK as the hardware device.

Example:

```
EXTRACT CUSTOMER-NAME, BALANCE-DUE TO
   EF-CONFIRM; HARDWARE IS DISK.
```

The underlined portion serves as documentation of the default hardware assignment of DISK.

F-H These paths are taken to specify magnetic tape (TAPE, TAPE-7, or TAPE-9) as the hardware device for the extracted file. If you specify TAPE, TAPE-7, or TAPE-9, all items are extracted with display usage. TAPE-7 and TAPE-9 are identical to TAPE.

Example:

```
EXTRACT CUSTOMER-NAME, BALANCE-DUE TO
   EF-CONFIRM EG-CONFIRM HARDWARE TAPE.
```

In this example, the file is extracted to tape instead of disk.

I Take this path to specify card punch as the hardware device for the extracted file. You must ensure that the total extract record size is less than or equal to 80 characters. All numeric items are extracted in display mode if PUNCH is specified.

Example:

```
EXTRACT CUSTOMER-NAME, BALANCE-DUE TO
   EF-NEW HARDWARE IS PUNCH.
```

In this example, the records are extracted to the card punch.

J Take this path to specify DISKPACK as the hardware device for the extracted file. If you use this path, a pack name must be given via the ID clause (refer to path B).

Example:

```
EXTRACT CUSTOMER-NAME, BALANCE-DUE TO
   EF-CONFIRM;ID = "BALDUE" PACK "ACCTCL";
      HARDWARE IS DISKPACK.
```

K Take this path to specify the number of areas for the extracted file. This applies only to DISK and DISKPACK files. Refer to Appendix B for default and limit values.

L Take this path to specify the AREASIZE of the extracted file. In this case, the total number of records in the file will be the product of the number of areas and the AREASIZE. AREASIZE must be a multiple of the BLOCKSIZE if path A has been taken. Refer to Appendix B for default and limit values.

Example:

```
EXTRACT CUSTOMER-NAME, BALANCE-DUE TO
   EF-CONFIRM; AREA IS 25; AREASIZE IS 10000.
```

M Take this path to specify that all numeric items in the extract file are to have a usage of DISPLAY rather than the default of COMPUTATION. This path affects only extract files with a hardware device of DISK or DISKPACK. Note that because the sign of a signed field is encoded on the first digit as a high order bit, the plus (+) or minus (-) symbol and the first digit might not be displayed correctly.

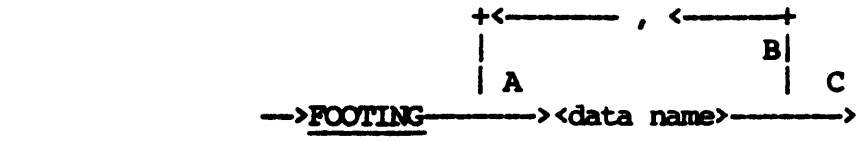
N Take this path until you have specified all desired attributes for the file.

- Take this path when you have specified all desired attributes for the file.

FOOTINGS

The Footings clause specifies which columns are to be footed (in other words, totaled).

The syntax for the Footings clause is as follows:



The paths of this syntax diagram are explained below:

Path	Explanation
A	Take this path to foot a previously specified column. <Data name> must be specified as a column name in the REPORT statement. If print suppression is specified for this column, the footing of the column reflects only those values actually printed in the column.

Example: ("INVENT")

```
REPORT PART-NO, INVENT-DOLLAR-VALUE
                .
                .
                .
SUMMARIZE FOOTING INVENT-DOLLAR-VALUE.
```

The SUMMARIZE statement specifies that the column for INVENTORY-DOLLAR-VALUE is to be footed. The report appears as follows:

```
                PART          INVENT
                 NO           DOLLAR
                               VALUE

                113110         131.00
                203171         701.63
                  .              .
                  .              .
                  .              .

SUMMARIES FOR FINAL
TOTAL                        89131.50
```

B Take this path as many times as necessary to specify all columns to be footed. The column names can be specified in any order.

Example: ("CLIENT")

```
REPORT BY BRANCH
    LISTING CUST-NO, CREDIT-LIMIT, BALANCE-DUE.
            .
            .
            .
SUMMARIZE FOOTING CREDIT-LIMIT, BALANCE-DUE.
```

This produces the following type of report.

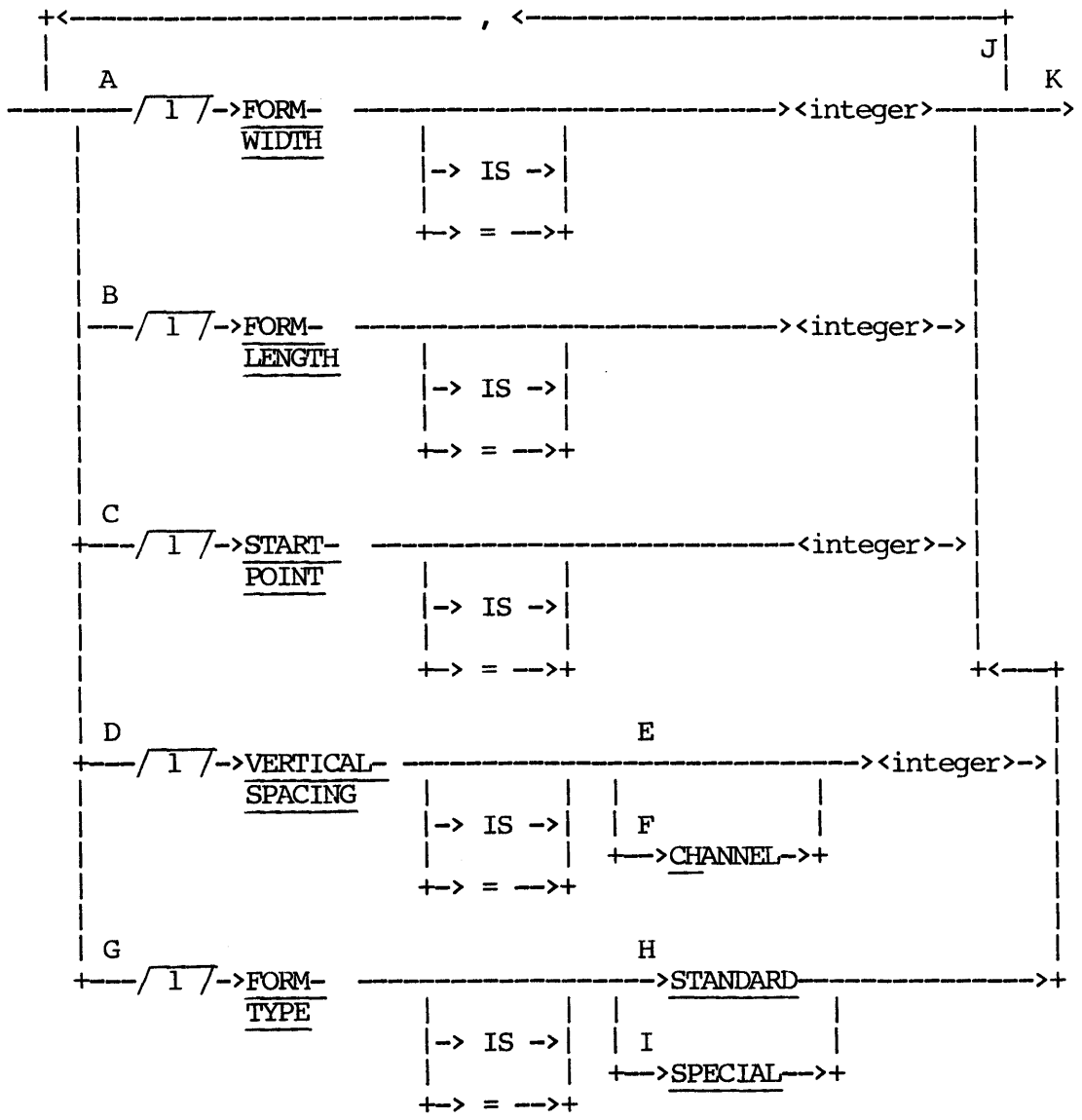
```
                .
                .
                .
BRANCH:  0013

         CUST          CREDIT        BALANCE
           NO           LIMIT            DUE

        30334            5000      $  1731.00
        30714            1000      $   763.00
          .                .            .
          .                .            .
          .                .            .

SUMMARIES FOR BRANCH:  0013
TOTAL                 113000      $ 93100.80

            .
            .
            .
SUMMARIES FOR FINAL
TOTAL                 987700      $871310.93
```

C Take this path when all columns to be footed are specified.

FORM ATTRIBUTES

The Form Attributes specification is used to define any of the following attributes: form width, form length, the first printable position of the form, the number of lines between forms, and form type. These attributes only apply to the PRINT statement.

The syntax for the Form Attributes specification is as follows:



If you do not specify a form attribute, the corresponding default form attribute is assumed, as follows:

FORM-WIDTH	132
FORM-LENGTH	54
START-POINT	1
VERTICAL-SPACING	CHANNEL 1
FORM-TYPE	STANDARD

The paths of this syntax diagram are explained below:

Path	Explanation

A Take this path to specify the number of horizontal print positions from the start point (refer to path C) to the right edge. Subsequent <print specifications> given in the PRINT statement describe the layout of information on the form in relative fashion from the first printable position on a line (one) to the last printable position on the line (FORM-WIDTH).

B Take this path to specify the total number of lines which can be printed on the form. Subsequent <print specifications> given in the PRINT statement describe the layout of information on the form in relative fashion from the first printable line (one) to the last printable line (FORM-LENGTH).

C Take this path to specify the number of print positions from the left where the first printable position of the form begins (position one). You can adjust START-POINT and FORM-WIDTH to provide for a standard left margin.

D VERTICAL-SPACING is the number of lines between the last printable line on a form and the first printable line on the next form. The attribute may be given optionally as a channel number which corresponds to the first printable line on a form.

E Take this path if the <integer> represents the number of lines between forms.

F Take this path if the <integer> represents a channel number.

G Take this path if you desire to specify whether the form is an installation standard or a special form which requires special operator load instructions. If you do not take this path, the default form type (installation standard: STANDARD) is assumed by the system.

H If you specify STANDARD, no load/unload messages are given by the system.

I If you specify SPECIAL, the forms output file, or listing, is assigned the system forms attribute.

For the A Series and B 2000-B 4000 line of computers, special instructions to the operator are issued by the system when the loading of forms on the printer is required. Provision is also made to inform the operator to unload forms when output is completed.

For the B 1000 line of computers, special instructions to the operator are issued by the program when the loading of forms on the printer, and the unloading of forms from the printer, respectively, is required. The operator takes the appropriate action and then issues a <mix #>AX message to continue program execution.

NOTE

If the printer file is designated to go to backup, no forms messages will be issued.

J Take this path to specify additional form attributes.

K Take this path after all nondefault attributes are given.

Example:

```
PRINT (START-POINT = 11, FORM-WIDTH = 30,
   FORM-LENGTH = 8, FORM-TYPE = SPECIAL)... .
```

Form Attributes are given in the above PRINT statement to describe 1-up mailing labels. The left margin is 10 positions wide. Information can be placed into positions 1 through 30 on each of lines 1 through 8. The first line of each label corresponds, by default, to channel one. Special instructions are given to the operator to load and unload the labels.

FULL-LENGTH MONTH OPTION

The process option SET FULL-LENGTH MONTH allows you to change the 10-character month strings to an alternate language. The default language is English. The table shown below shows the 10-character full-length names for months in other languages.

Month	English (DEFAULT)	French	German
1	" JANUARY "	" JANVIER "	" JANUAR "
2	" FEBRUARY "	" FEVRIER "	" FEBRUAR "
3	" MARCH "	" MARS "	" MAERZ "
4	" APRIL "	" AVRIL "	" APRIL "
5	" MAY "	" MAI "	" MAI "
6	" JUNE "	" JUIN "	" JUNI "
7	" JULY "	" JUILLET "	" JULI "
8	" AUGUST "	" AOUT "	" AUGUST "
9	"SEPTEMBER "	"SEPTEMBRE "	"SEPTEMBER "
10	" OCTOBER "	" OCTOBRE "	" OKTOBER "
11	" NOVEMBER "	" NOVEMBRE "	" NOVEMBER "
12	" DECEMBER "	" DECEMBRE "	" DEZEMBER "

	Italian	Spanish
1	" GENNAIO "	" ENERO "
2	" FEBBRAIO "	" FEBRERO "
3	" MARZO "	" MARZO "
4	" APRILE "	" ABRIL "
5	" MAGGIO "	" MAYO "
6	" GIUGNO "	" JUNIO "
7	" LUGLIO "	" JULIO "
8	" AGOSTO "	" AGOSTO "
9	"SETTEMBRE "	"SEPTIEMBRE"
10	" OTTOBRE "	" OCTUBRE "
11	" NOVEMBRE "	"NOVIEMBRE "
12	" DICEMBRE "	"DICIEMBRE "

The 10-character month strings are used for the month portion of the MONTH-DD-YYYY-DATE and DD-MONTH-YYYY-DATE clauses in the TITLE statement and the PRINT insert clause. When the date is formatted, the month is followed by a single space or comma, depending on the format used. Although the above tables show the month names centered in each field, both leading and trailing spaces are ignored when the date is formatted.

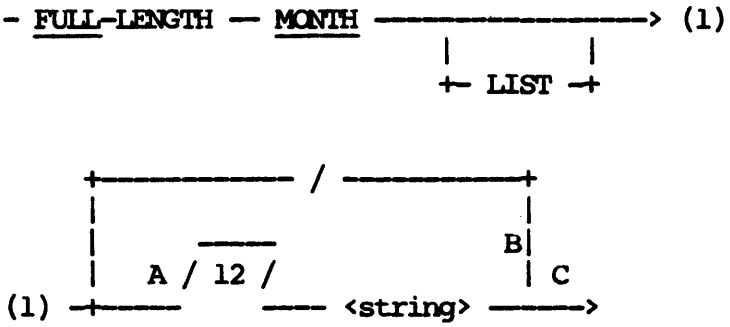
For example, the dates September 12, 1988 and May 3, 1988 format as follows:

Date Format	September 12	May 3
MONTH-DD-YYYY	SEPTEMBER 12, 1988	MAY 03, 1988
DD-MONTH-YYYY	12 SEPTEMBER, 1988	03 MAY, 1988

Although the dates differ in length, REPORTER III considers all dates to be 19 characters in length for formatting purposes.

To avoid entering the FULL-LENGTH MONTH option for every report specification in which a different, nondefault language is desired, it is recommended that this option be stored in the vocabulary with the REPLACE statement. Then the name given in the REPLACE statement can be used when the option is desired.

The syntax for the FULL-LENGTH month option is as follows:



The paths of this syntax diagram are explained below:

Path	Explanation
A	This path must be taken 12 times. The string given cannot exceed 10 characters.
B	Take this path until all 12 month strings have been entered.

C Take this path after all 12 full-length month strings have been given.

Example:

```
SET FULL-LENGTH  MONTH          % TO SPANISH       MONTHS
"  ENERO    " / " FEBRERO  " / "  MARZO    " /  % 1  2  3
"  ABRIL    " / "   MAYO   " / "  JUNIO    " /  % 4  5  6
"  JULIO    " / "  AGOSTO  " / "SEPTIEMBRE" /  % 7  8  9
" OCTUBRE   " / "NOVIEMBRE " / "DICIEMBRE " .  % 10 11 12
```

GROUP STATEMENT

The GROUP statement is used to specify control-break items. These items are used to group the information being reported and provide a basis for preparing summaries.

Control breaks for a single report, or a section of a multiple report, cannot have been previously specified in another statement. Items specified as control-breaks in one section of a multiple report are not considered control-break items in another section.

Example: ("VOCEMP")

```
GROUP BY JOB-GRADE.
```

In this example, logical records, or employees, are grouped by job grade. Summaries of employee data then can be specified within the JOB-GRADE groupings.

Example: ("INVENT")

```
        .
        .
        .
GROUP BY DEPARTMENT.
SUMMARIZE FOR EACH DEPARTMENT TOTAL(QUANTITY),
   TOTAL(INVENT-DOLLAR-VALUE).
        .
        .
        .
```

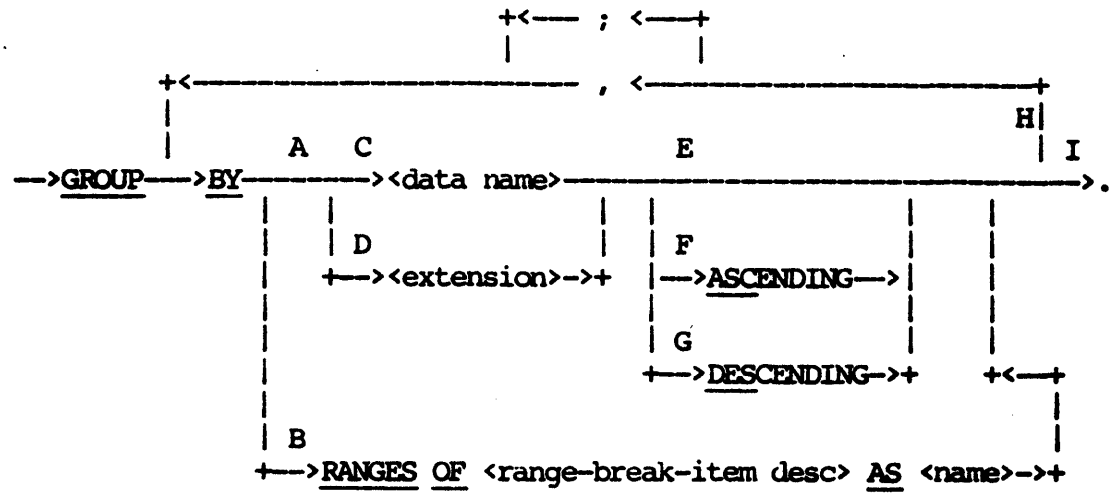
In this example, parts are grouped by department, total quantity, and inventory dollar value, and are obtained for each department. (Refer to the SUMMARIZE statement in this section.)

The following type of report would result:

```
        .
        .
        .
```

```
SUMMARIES FOR DEPARTMENT:  0134
TOTAL(QUANTITY):  35
TOTAL (INVENT DOLLAR VALUE):  3115.05

SUMMARIES FOR DEPARTMENT:  0135
TOTAL(QUANTITY):  60
TOTAL (INVENT DOLLAR VALUE):  4063.80
      .
      .
      .
```

The syntax for the GROUP statement is as follows:



The paths of this syntax diagram are explained below:

Path	Explanation
A	Take this path to define a data item as a control-break item. Information pertaining to each value of a data item is grouped together.
B	Take this path to define a range-break item as a control-break item. (This path cannot be taken more than once for the same range-break item.) Information is grouped based on specified value ranges of a particular data item. The range-break item is referenced in other statements by a <c-b name> which is specified here as <name>. The given <name> must be unique for the specified report, that is, it must not duplicate any other <name> defined in the vocabulary or defined previously by you.

Example: ("VOCEMP")

```
GROUP BY RANGES OF EMPYE-AGE FROM
   20 BY 5 TO 80 AS AGE-INTERVAL.
      .
      .
      .
REPORT FOR EACH AGE-INTERVAL:  COUNT,
   TOTAL(SALARY).
```

Information pertaining to each age range is grouped together. Statistics are calculated and printed for each age range (refer to the REPORT statement in this section).

C Take this path to specify a nonstatistical data item referenced by <data name> as a control-break item. The <data name> is identified as a control-break name, and you can use it as a <c-b name> in other statements to reference a control-break grouping of information.

Example: ("VOCAST")

```
GROUP BY ASSET-TYPE.
```

ASSET-TYPE is an input data item. The input information is grouped according to ASSET-TYPE. This brings all similar assets together for purposes of summarization and reporting.

D Take this path to define an <extension> and specify a control break at the same time. The <data name> defined by the <extension> becomes a control-break name. The value of the control-break item is defined by the <extension>. The derived data item must be nonstatistical. You can use the <data name> in other statements as a <c-b name> to reference a control-break grouping of information.

Example: ("CLIENT")

```
GROUP BY CREDIT-CODE IS CREDIT-LIMIT / 100.
```

CREDIT-CODE is defined as a derived data item. Its value is CREDIT-LIMIT/100. This value is used to group the logical records by identical CREDIT-CODE values.

E Take this path when no particular ordering based on the control-break item is desired. Unless suppressed, a sort is performed to properly group the logical records based on the control-break item. The default ordering of information is

ascending based on the control-break item. The examples for paths C and D use this option.

F Take this path to specify explicitly ascending order for control-break item values.

Example: ("VOCAST")

```
GROUP BY ASSET-TYPE ASCENDING.
```

G Take this path to specify descending order for control-break item values.

Example: ("CLIENT")

```
GROUP BY BRANCH DESCENDING.
```

The information is grouped based on BRANCH. Branch numbers are in descending sequence.

H Take this path as many times as necessary to specify all control-break items. If more than one control-break item is given, information is grouped hierarchically based on the order in which the control-break items are specified. For example, if two control-break items are specified, information is first organized based on the values of the control break specified first (that is, the outer-level control break); for each value of this control break, the information is then organized based on the values of the control break specified next (that is, the inner-level control break).

NOTE

The limit on the number of levels of control-break items is nine.

Example: ("VOCAST")

```
GROUP BY DEPT-NO, BY ASSET-TYPE.
```

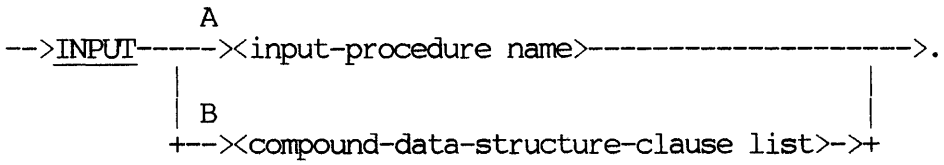
Assets are organized by similar department number and then are organized by similar type for each department.

I Take this path after you have specified all control breaks.

INPUT STATEMENT

The INPUT statement specifies access to one or more data structures or specifies the use of a user-written <input-procedure name> to obtain the required information for the report. The information specified by the INPUT statement is organized into a set of logical records. A logical record can be a single record from a single data structure (for example, a sequential tape file) or it can be a number of related records accessed from one or more data structures. Once a data structure is specified for INPUT, its data items can be referenced elsewhere in the report specification.

The general syntax for the INPUT statement is as follows:



The paths of this syntax diagram are explained below:

Path	Explanation
A	Take this path if you desire to report on information accessed by an input procedure. Input is supplied by the specified procedure, which is user-developed in COBOL and stored by RP3VOC in the vocabulary. The COBOL procedure must provide a complete logical record each time control is returned by the procedure. Once you specify an <input-procedure name> in the INPUT statement, reference can be made to all non-77 WORKING-STORAGE data items associated with the input procedure and to all data items within the vocabulary-defined data structures used by the input procedure. Refer to the Vocabulary Language Operations Guide for additional information on the input procedure.

Example:

INPUT GET-A-RECORD.

GET-A-RECORD is a vocabulary-defined <input-procedure name>. Each time the generated report program calls the procedure GET-A-RECORD, one logical record of information is returned.

B Take this path if REPORTER III-supplied input routines are to be used. These routines allow access to data records in any of the data structures known to the vocabulary. The <compound-data-structure- clause list> identifies all data structures to be accessed and how they are to be accessed. Information within these data structures then can be reported.

Example: ("VOCAST")

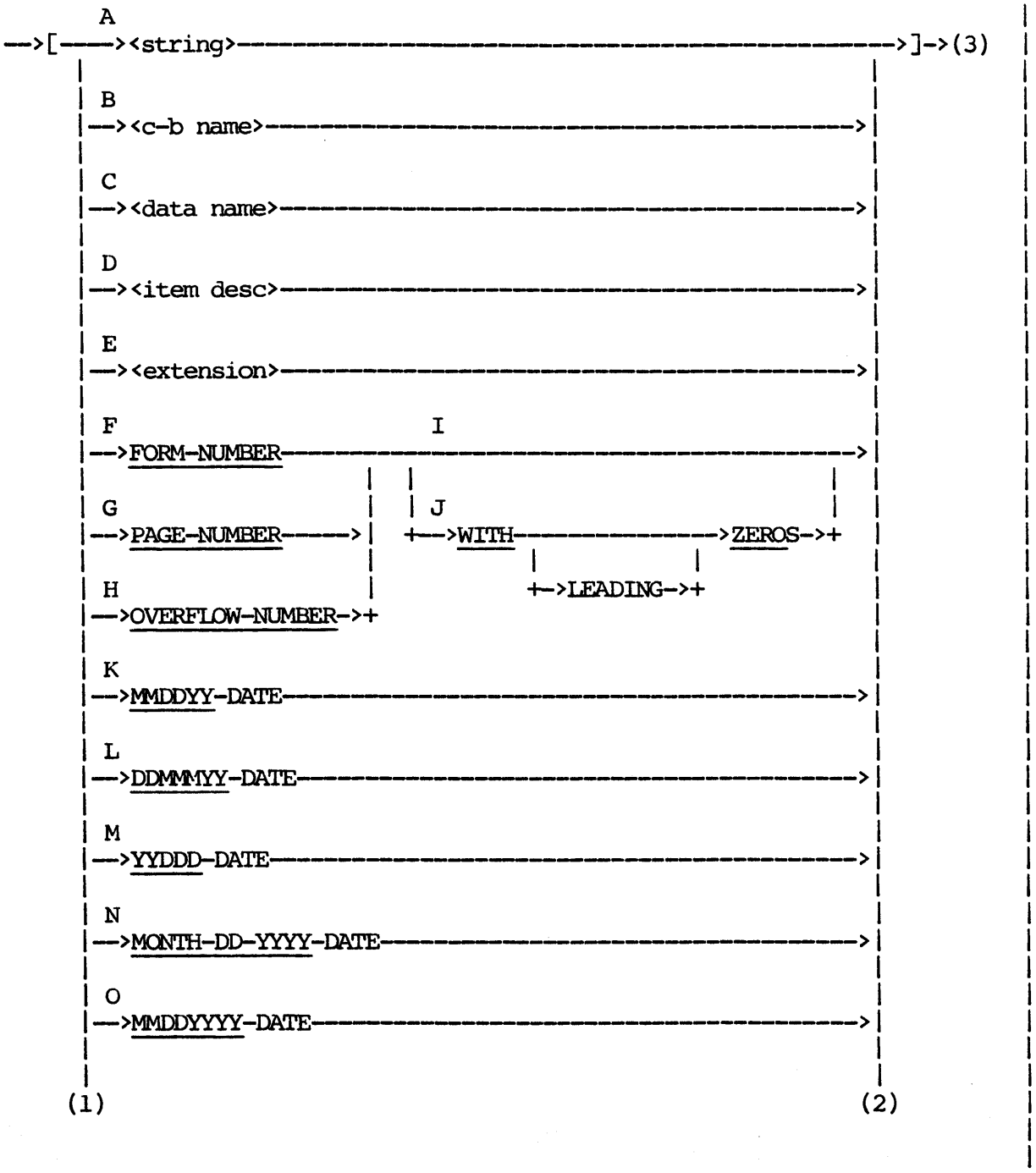
INPUT ASSET-FILE.

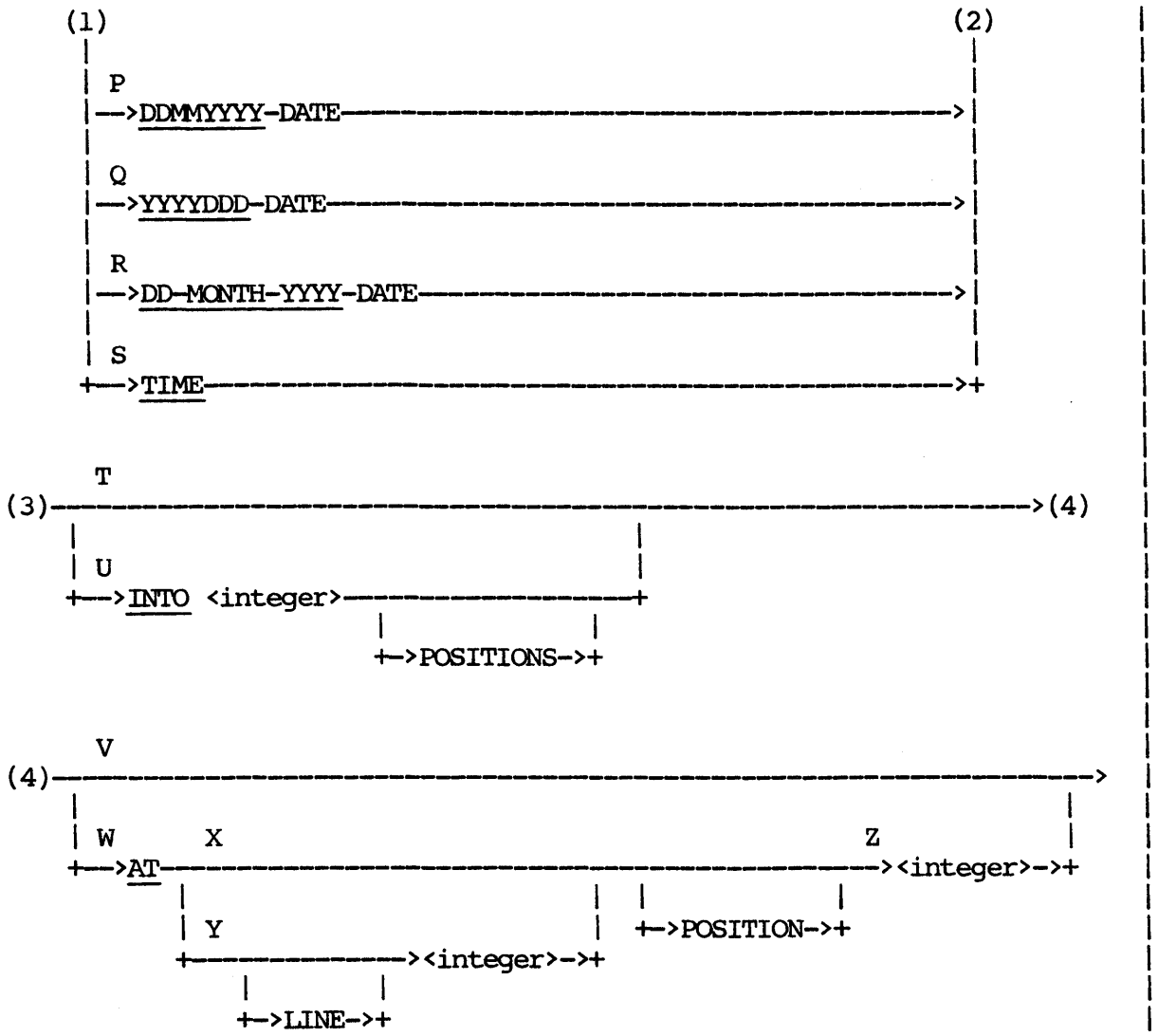
ASSET-FILE is a <file name>. All records in the file are input to the report. In this case, each record of the file corresponds to a logical record.

INSERT

Insert specifies the printing of an item value, a constant string of characters, a form number, or a line number.

The syntax for Insert is as follows:





The paths of this syntax diagram are explained below:

Path	Explanation
A	Take this path to specify insertion of a string of characters. Specification of a string insertion is identical to the print specification of a string with the following exceptions: 1. A string insertion can cause line overflow. 2. A line number and/or position number can be given for a string insertion.

Example:

```
["XYZ SAVINGS & LOAN"]
```

B Take this path to specify insertion of a control-break item value.

Examples: ("INVENT")

```
[BIN-NUMBER]
[RANGE OF INVENT-DOLLAR-VALUE]
```

C Take this path to specify insertion of an item represented by <data name>.

Example: ("CLIENT")

```
[STREET-ADDRESS]
```

D Take this path to specify insertion of a derived data item described by <item desc>.

Examples:

```
[TOTAL (COST)]
[INVENT-DOLLAR-VALUE]
[BALANCE - OLD-BALANCE NUM(8,2)]
```

E Take this path to specify the printing of a derived data item described by <extension>.

Example: ("INVENT")

```
[TOTAL-COST WHICH IS
   QUANTITY * UNIT-COST NUM(8,2)]
```

F Take this path to specify insertion of the form number. FORM-NUMBER refers to an 8-digit unsigned integer.

Example:

```
[FORM-NUMBER]
```

G Take this path to specify insertion of a page number on each physical page, beginning with 1. PAGE-NUMBER references an 8-digit unsigned integer that is incremented by 1 for each subsequent physical page.

H Take this path to specify insertion of an overflow form number, an 8-digit unsigned integer. It is 1 for all non-overflow forms. It is 2 for a first overflow form and is incremented by 1 for each additional overflow form.

Example:

```
PRINT CONFIRM-STATEMENT AS FOLLOWS: ...

   FOR OVERF FORM HEADER "PAGE",
      [OVERFLOW-NUMBER] INTO 2,
      "OF INVOICE STATEMENT";
   FOR OVERFLOW FORM FOOTNOTE
      "CONTINUED ON ATTACHED STATEMENT";
   .
   .
   .
```

CONFIRM-STATEMENT references a macro which describes a positive confirmation statement. All attached statements are numbered sequentially.

I Take this path to edit the form number, page number, or overflow form number, replacing leading zeros with blanks.

J Take this path to edit the form number, page number, or overflow form number, retaining leading zeros.

K-R Take one of these paths, whichever is appropriate, to print the current system date in the desired format on the forms. The keyword determines the format. Keywords and their corresponding formats are indicated in the following table. For illustrative purposes, the assumed date is October 25, 1982.

Path	Keyword	Example
K	MMDDYY-DATE	10/25/82
L	DDMMMYY-DATE	25OCT82
M	YYDDD-DATE	82299
N	MONTH-DD-YYYY-DATE	OCTOBER 25, 1982
O	MMDDYYYY-DATE	10/25/1982
P	DDMMMYYYY-DATE	25OCT1982
Q	YYYYDDD-DATE	1982299
R	DD-MONTH-YYYY-DATE	25 OCTOBER, 1982

Example:

```
[DDMMMYY-DATE]
```

For the date October 25, 1982, this inserts the date "25OCT82" on each page.

NOTE

In the present description of Insert, words signifying the type of date format (for example, JULIAN-DATE) are no longer indicated in keywords. Instead, letters indicating the actual date format (for example, YYDDD-DATE) are indicated in the keywords. However, REPORTER III will still recognize keywords with words signifying the type of date format (for example, JULIAN-DATE) if they are used in a report specification.

S Take this path to print the time the report is run on the forms in the format HH:MM:SS (hours, minutes, seconds).

T Take this path to insert the string, item, or form number into the number of positions equal to its edited length.

U Take this path to specify insertion of the string, item, form number, and so forth into the designated number of positions. For form number, page number, and overflow form number, you can take this path to limit the number of digits printed. For all others, an error results if the designated number of positions is less than the edited length of the string or item. If the number of positions is greater than the edited length, numeric values and date format YYDDD or YYYYDDD are right-justified, and string values and other date and time values are left-justified.

Example: ("INVENT")

```
[INVENT-DOLLAR-VALUE] IN 15
```

The original length of INVENT-DOLLAR-VALUE is 11. Therefore, the item value begins in position 5 of the 15-character insert field.

V If you take this path, the insertion begins at the next available print position.

Example:

```
PRINT AS FOLLOWS:
   FOR EACH FORM HEADER 25 S, "-",
      [PAGE-NUMBER] IN 4, 3 S, "-", BL;
```

The page number is inserted into 4 positions immediately following the "-".

W Take this path to select the exact position of the insertion. This path can be taken only when a NEW FORM, NEW LINE, or BLANK LINE print specification has been previously provided and a <data name> <string> print specification does not immediately precede this Insert clause.

If a conditional print specification immediately precedes this Insert clause, Path W may only be taken if the ELSE clause of the conditional print specification is used. In addition, a print specification that follows the ELSE clause must reference the same amount of information as the print specification that precedes the ELSE clause.

Example 1: ("CLIENT")

```
PRINT (FORM-WIDTH IS 40, FORM-LENGTH IS 8)
  FOR EACH RECORD
    NF,
    [CUSTOMER-NAME] AT LINE 1 POSITION 1,
    [CUSTOMER-NUMBER] AT LINE 1 POSITION 32,
    [STREET-ADDRESS] AT LINE 2 1,
    [CITY-STATE] AT 3 1,
    [ZIP-CODE] AT 3 22.
```

This example shows how mailing labels might be created using the PRINT statement with the Insert clause. The customer's name is inserted at the first line and the first position. The customer's account number is also on the first line starting at position 32. The customer's address is inserted on the second line and the first position. The city and state are inserted on the third line and the first position. The zip code is also on the third line, and it starts at position 22.

Example 2: ("CLIENT")

```
PRINT FOR EACH RECORD
  NL,
  [CUSTOMER-NAME] AT POSITION 1,
  [CUSTOMER-NUMBER] AT 32,
  IF BRANCH-NUMBER = 10
      (1 S, "HOME OFFICE ACCT")
  ELSE
      (1 S, "                "),
  [RISK-RATING] AT 56.
```

This example shows the use of the AT clause within the Insert clause following a conditional print statement. The ELSE clause of the conditional print statement is used, and the print specification following the ELSE clause references the same

amount of information as the print specification preceding the ELSE clause.

X Take this path if the insertion begins, not in the next available print position, but somewhere to the right on the same line.

Example:

```
PRINT AS FOLLOWS:
   FOR EACH RECORD NEW LINE,[NAME],
      [SALARY] AT 40, ... .
```

In the user-formatted report, the employee salary begins following the employee name in position 40 of a line.

Y Take this path to specify the line number of the insertion. This path can be taken only when a NEW FORM print specification has been given previously in the print specifications, and if path E was not taken. The line number must be between 1 and FORM-LENGTH (minus the number of lines allocated for form footnotes).

Example: ("CLIENT")

```
[STREET-ADDRESS] AT LINE 3 POSITION 1
```

Z Take this path to give the position number for the insertion. The <integer> specified must be between 1 and FORM-WIDTH and at or beyond the next available print position.

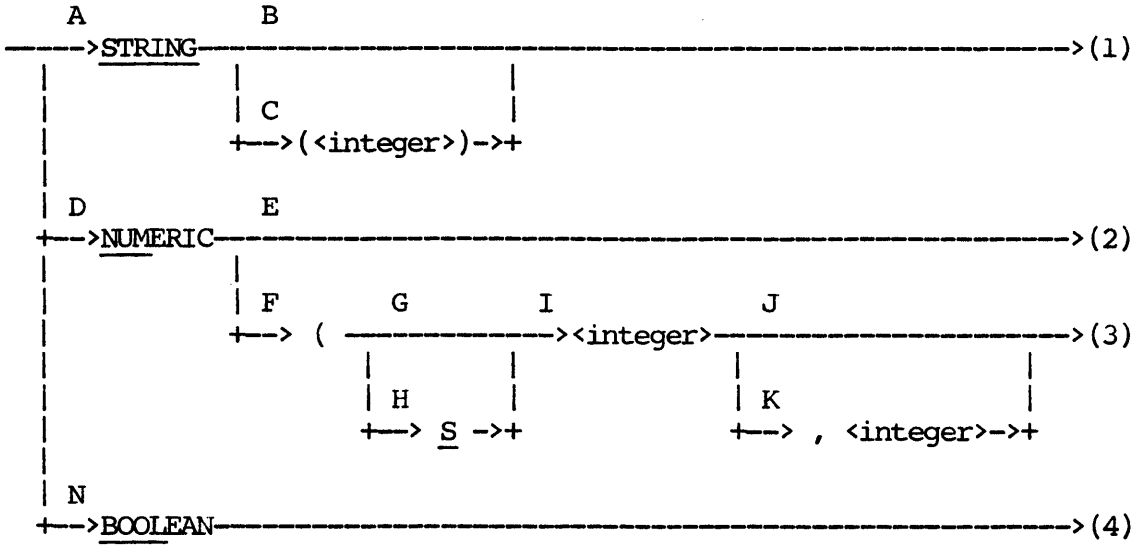
Example: ("CLIENT")

```
[STREET-ADDRESS] AT LINE 3 POSITION 1
```

INTERNAL ATTRIBUTES

The Internal Attributes specification declares the type and size of the data item. When used in an Item Desc, the internal-attribute type must match the expression type given in the Item Desc with the exception that a NUMERIC type can be given with a Boolean expression. When used in an ACCEPT statement, the internal-attribute type defines the type of item to be accepted.

The syntax for the Internal Attributes specification is as follows:



The paths of this syntax diagram are explained below:

Path	Explanation

A Take this path to specify a <string> type item.

Example:

```
STRING
```

This declares the data item to be a <string>.

B Take this path if the default <string> length is to be used for this item. The default length is STRING-SIZE. STRING-SIZE can be set by the Process-option SET statement. If STRING-SIZE is not set, it defaults to 30.

Example:

```
ACCEPT USER-ID STRING.
```

The length of the item USER-ID is STRING-SIZE.

C Take this path to specify a nondefault length for the <string> item. The <integer> specifies the length of the item in number of characters. If necessary, a string value is truncated or extended with spaces by the system when it is assigned to the item.

Example:

```
STRING(10)
```

This declares the data item is a <string> of 10 characters.

D Take this path to specify a numeric-type data item.

Example:

```
NUMERIC
```

The data item is declared to be signed numeric.

E Take this path if the default size for a numeric data item is to be used. This path also specifies a signed numeric data item. The default size for numeric data items is determined by INTEGER-SIZE and FRACTION-SIZE. Both INTEGER-SIZE and FRACTION-SIZE can be set by the Process-option SET statement.

F Take this path to specify the size of the item.

Example:

```
NUM(8)
```

This specifies the data item to be an unsigned numeric integer of eight digits.

G Take this path to specify a numeric data item that is unsigned. The value of an unsigned numeric item may only be positive. The example for path F illustrates this specification.

H Take this path to specify a signed numeric data item. Signed numeric items have both positive and negative values. A space must separate the "S" from the following <integer>.

Example:

```
NUM(S 8)
```

This specifies the data item to be a signed 8-digit number.

I The <integer> given here specifies the number of digits in the integer part of the numeric item.

J Take this path to specify an integer-only numeric item. In this case, the value contains no fractional digits, as the examples for paths F and H illustrate.

K Take this path to specify a fractional part for the data item being defined. The <integer> specifies the size of the fraction in terms of number of digits.

Example:

```
NUM(S 5,2)
```

This specifies the data item to be a signed 7-digit numeric item with five integer digits and two fractional digits.

L Take this path if the value assigned to the item is to be truncated to the size of the item. Truncation involves the elimination of excessive fractional digits.

M Take this path to specify that any value assigned to the item be rounded to the fractional size of the item.

Example:

```
NUMERIC (S 8,2) ROUNDED
```

This example specifies a signed 10-digit number with eight integer digits and two fractional digits. Values are rounded to two fractional digits (in other words, cents) when assigned to the item.

NOTE

Numbers are limited to a maximum of 18 digits. This includes the fractional portion of the number and the sign position. In each of the following three examples, the maximum number of digits allowed is specified:

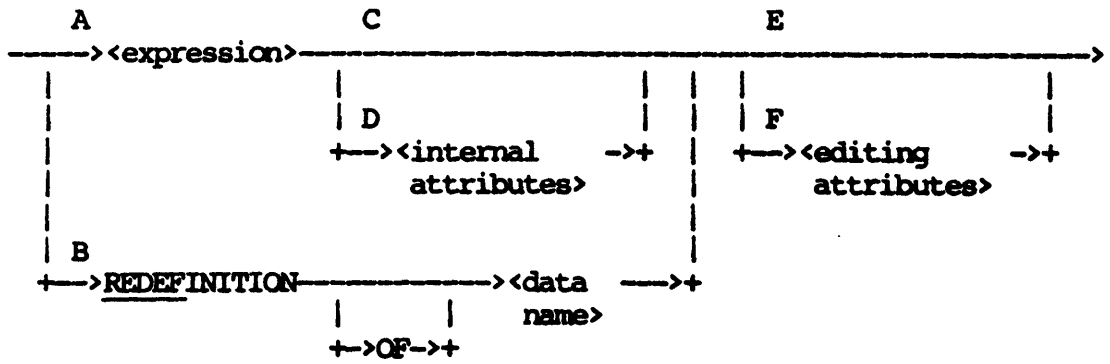
```
NUM(11,7)
NUM(18)
NUM(0,18)
NUM(S, 17)
```

N Take this path to specify a Boolean-type data item. Boolean items are represented internally as a 1 or 0 and printed as TRUE or FALSE, respectively.

ITEM DESC

The Item Desc specification describes a new data item. Item Desc is used in the Extension clause to define the value, item attributes, editing picture, and total population of the new data item.

The syntax for the Item Desc specification is as follows:



The paths of this syntax diagram are explained below:

Path	Explanation
A	Take this path to define the value of the data item by an <expression>. For every logical record processed, a value is calculated for the <expression> and assigned as the value of the Item Desc. Example: OVER-DUE IS SIXTY-DAYS-DUE + NINETY-DAYS-DUE. The value of OVER-DUE is the sum of two data items.
B	If a vocabulary data item to be reported has a name identical to that of a REPORTER III function, take this path to define an identical item with a different name. The derived data item has the same value and internal attributes as the item referenced. Editing-attributes are also the same if not given explicitly.

Example:

ACCT-AGE IS REDEF OF ACCOUNT-AGE

ACCT-AGE now references the same information as does ACCOUNT-AGE.

C Take this path to assign appropriate default <internal attributes> to the item.

Default <internal attributes> are determined by the type and form of the <expression>:

1. If the <expression> is a data name, the default internal attributes are identical to the internal attributes of the referenced data item.

2. If the <expression> is a REPORTER III function (for example, the <AGE function>), the default internal attributes are the attributes of the value returned by the function (the internal attribute returned for the <AGE function> is a 5-digit signed integer).

3. If the <expression> is other than a data name or REPORTER III function and the value of the <expression> is numeric, INTEGER-SIZE and FRACTION-SIZE are used to set the <internal attributes> for a signed numeric item (refer to the Process-option SET statement in this section).

4. If the <expression> is a string, the length of the string is used to set the <internal attributes> for the string item.

5. If the <expression> is Boolean, an unsigned digit (0 = FALSE, 1 = TRUE) is used internally to store the value of the Boolean item.

D Take this path to specify explicitly <internal attributes> for the item. The specified type of the item must match the type of the <expression>, but the item size can be set to any value.

Example: ("CLIENT")

OVER-DUE IS SIXTY-DAYS-DUE + NINETY-DAYS-DUE
NUMERIC(5,2)

The underlined portion specifies that OVER-DUE is to reference a 7-digit unsigned numeric item. Five digits are integers, and two digits are fractions.

E Take this path to assign appropriate <editing attributes> to the item.

If path B was taken, the default editing attributes are identical to those of the item being redefined. If path A was taken, the default editing picture is determined by the internal item attributes with the following two exceptions:

1. If the <expression> is a data name and path C was taken, the default editing picture is the editing picture of the referenced data item.

2. If the <expression> is a REPORTER III statistical function other than COUNT or RUNNING-COUNT and path C was taken, the default editing picture is the expanded editing picture of the data item being summarized.

The editing picture is based on the maximum TOTAL-POPULATION of the data structures specified in the INPUT statement. (Refer also to the Report-option SET statement in this section.)

NOTE

A Boolean item is reported as TRUE or FALSE with a default editing picture of X(5).

Example: ("INVENT")

```
INVENTORY-VALUE IS INVENT-DOLLAR-VALUE.
```

This statement creates an INVENTORY-VALUE item which is identical in value and attributes to the INVENT-DOLLAR-VALUE item.

F Take this path to specify a nondefault editing picture for the item.

Example: ("CLIENT")

```
OVER-DUE IS SIXTY-DAYS-DUE + NINETY-DAYS-DUE

  WITH PIC "$Z(4)9.99".
```

The underlined portion specifies an editing picture to be used for the value of OVER-DUE.

NOTE

If the vocabulary being used has the option DECIMAL-POINT IS COMMA set, then the picture must reflect this option to ensure the proper decimal-point alignment.

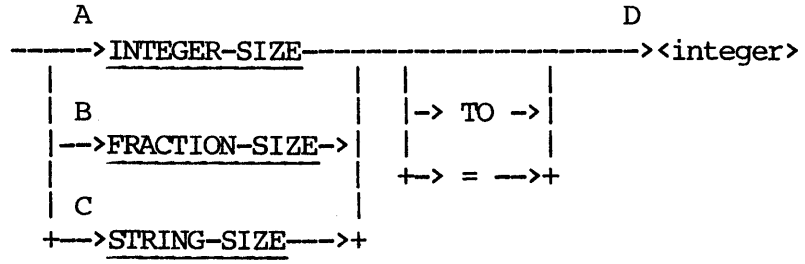
Example:

```
WITH PIC "Z.ZZ9,99".
```

ITEM SIZE

Item Size is used to specify the nondefault size of intermediate values resulting from calculation.

The syntax for Item Size is as follows:



The paths of this syntax diagram are explained below:

Path	Explanation
A	Take this path to specify the number of digits to be maintained in the integer portion of intermediate result values. The number specified must be between 1 and 18 for B 1000 systems, and must be between 1 and 23 for A Series and B 2000/B 3000/ B 4000 systems. The default is 12. In addition, INTEGER-SIZE plus FRACTION-SIZE must not be greater than 18 digits for B 1000 systems, and must not be greater than 23 digits for A Series and B 2000/B 3000/B 4000 systems. Example: SET INTEGER-SIZE TO 10.
B	Take this path to specify the number of digits to be maintained in the fractional portion of intermediate result values. Thus, all intermediate results are truncated to this number of fractional digits. The number specified must be between 0 and 18 for B 1000 systems, and must be between 0 and 23 for A Series and B 2000/B 3000/B 4000 systems. The default is 5. In addition, INTEGER-SIZE plus FRACTION-SIZE must not be greater than 18 digits for B 1000 systems, and must not be greater than 23 digits for A Series and B 2000/B 3000/B 4000 systems. Example: SET FRACTION-SIZE TO 0.

C Take this path to specify the number of characters to be maintained for intermediate string results. Where intermediate results are required for string manipulations, strings are truncated to the right of this number of characters. When using vocabularies created with software releases earlier than 2.50, this path also specifies the size of referenced groups. The default is 30.

Example:

SET STRING-SIZE TO 20.

D Take this path to specify the integer value to be assigned with respect to path A, B, or C, whichever was taken. Specify this integer in terms of characters or digits, whichever is applicable.

LITERALS

A <literal> represents a data item which has a value identical to the value being described. There are two classes of literals: numeric literals, or <number>s, and nonnumeric literals, or <string>s.

NUMBERS.

A <number> is defined as an element composed of characters chosen from the digits 0 through 9, a plus sign (+) or the minus sign (-), and the decimal point (.) or comma (,) if the vocabulary being used has the option DECIMAL-POINT IS COMMA set. The rules for the formation of a <number> are the following:

1. A <number> can contain only one sign character and one decimal point.

2. A <number> must contain at least one digit.

3. The sign of a <number> must appear as the left-most character. If a sign is not present, the <number> is defined as a positive value.

4. A <number> cannot exceed 18 characters for the B 1000 and B 2000/B 3000/B 4000 Series of systems and 22 for the A Series of systems, including the optional sign (+ or -) character.

5. The decimal point cannot appear as the right-most character of the <number>. The absence of a decimal point denotes an <integer>.

NOTE

The form 25. is incorrect, but either 25.0 or 25 is a correct representation for a <number>.

6. A <number> cannot contain more than six characters to the right of the decimal point.

The following are example numbers:

Without DECIMAL POINT IS COMMA Set	With DECIMAL POINT IS COMMA Set
132470	13247
.005	,005
+1.808	+1,808
-.0968	-,0968
7894.64	7894,64
10	10

INTEGERS

An <integer> is a number which contains no decimal point.

STRINGS

A <string> can contain any allowable character with the exception of the quotation mark. The beginning and end of the <string> is denoted by a quotation mark. Any character enclosed by the quotation marks is a part of the <string>. A maximum of 55 characters can be used in a <string>.

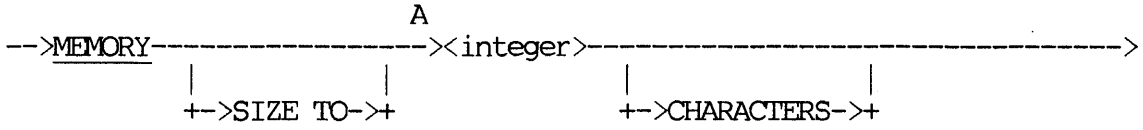
Examples:

```
"THIS IS A SAMPLE STRING -/$."
"ACTUAL SALES FIGURE"
"-1234.567"
```

MEMORY SIZE

The process statement SET MEMORY SIZE allows you to increase the amount of memory used by the generated report program while it is executing. This option is only necessary when very large report specifications have been developed or a large amount of data is to be sorted. The type of system plays a factor in how the extra memory is used.

The syntax for the MEMORY SIZE option is as follows:



Path	Explanation
A	Take this path to specify the amount of memory to be used by the object program created by the REPORTER III system. The actual amount of memory used depends on the system type. The maximum value allowed for <integer> is 999999.

A Series systems use the MEMORY SIZE option in conjunction with the sorting of input data. If SORT has been suppressed, this statement has no effect. If no memory size is specified, a default of 12,000 words (72,000 characters) is used by the COBOL compiler. Increasing the default memory used by the sorting process can possibly increase performance. Specifying a number less than 72000 will not decrease the memory used for sorting.

For V Series systems, increasing the memory size allows large report programs to be compiled. The use of the MEMORY SIZE option might be required for report specifications that contain many multiple reports, that use the PRINT statement extensively, and that use multiple files or a large data base. If the syntax error "MEMORY SIZE >= 300K DIGITS REQUIRED" is received during the compilation of a report program, enter "SET MEMORY SIZE TO 15000" to allow the compilation. Increasing the memory size when not required causes excessive memory to be used. However, this excessive memory can be an advantage when large amounts of data are sorted.

For B 1000 systems, increasing the memory size increases the amount of dynamic memory used by the report program. In some cases this increase can improve the performance of the report program.

For additional information, refer to the appropriate ANSI-74 or ANSI-85 COBOL reference manual for your system.

NONSTATISTICAL EXPRESSION

A nonstatistical expression is an expression which does not contain any statistical functions or statistical data items. A nonstatistical expression can be of the type arithmetic, string, or Boolean.

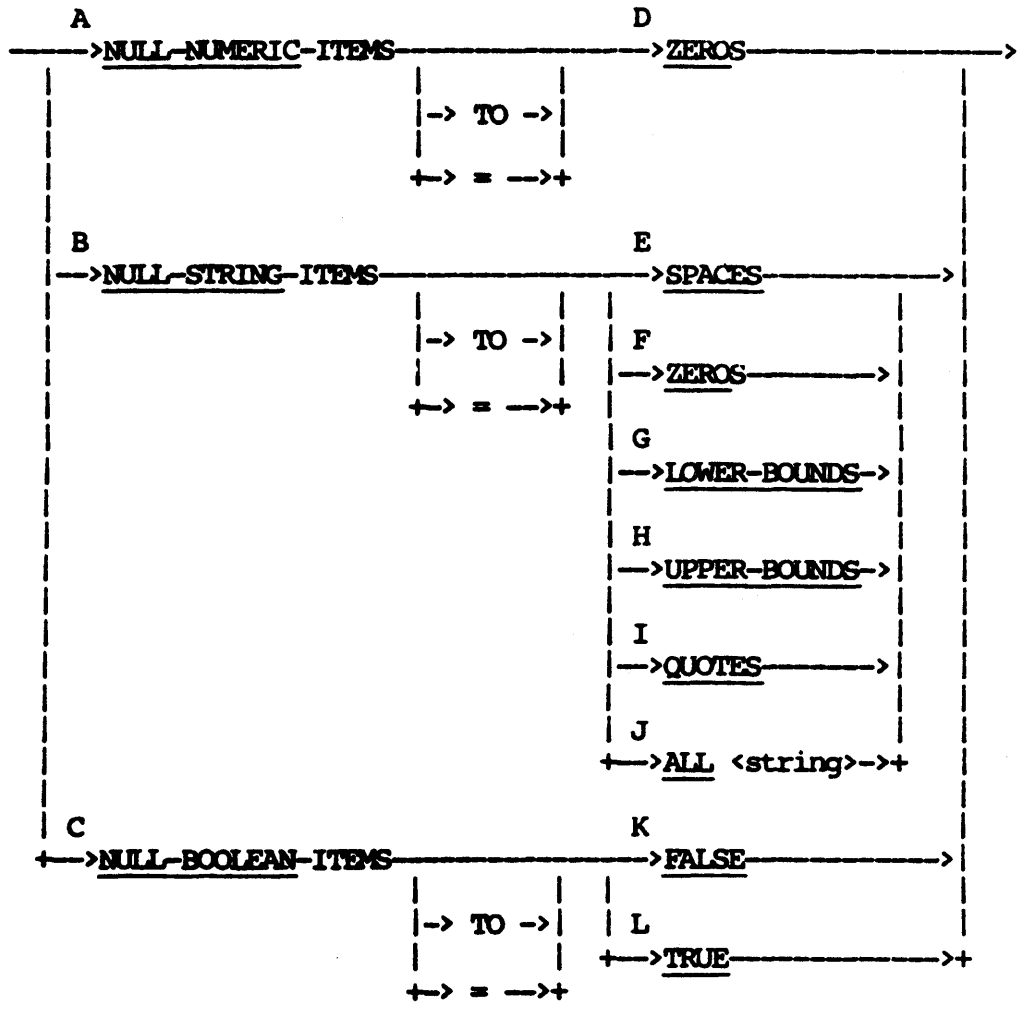
Example:

```
QUANTITY * UNIT-COST
AGE (FROM ORDER-DATE TO DATE) > 30
```

NULL SPEC

Null Spec is used to specify a nondefault value for null items.

The syntax for Null Spec is as follows:



The paths of this syntax diagram are explained below:

Path	Explanation
A	Take this path to specify the value to be assigned to NULL numeric data items for purposes of reporting and processing. The default value is ZEROS.

B Take this path to specify the value to be assigned to NULL string items for purposes of reporting and processing. The default value is SPACES.

C Take this path to specify the value to be assigned to NULL Boolean items for purposes of reporting and processing. The default value is FALSE.

NOTE

This option (path C) applies only to A Series of Systems.

D Take this path to specify the default NULL-NUMERIC-ITEM value of zero.

Example:

```
SET NULL-NUMERIC-ITEMS TO ZEROS.
```

E Take this path to specify the default NULL-STRING-ITEM value of all blanks.

Example:

```
SET NULL-STRING-ITEM TO SPACES.
```

F Take this path to specify that character zeros be used as the NULL-STRING-ITEM value.

Example:

```
SET NULL-STRING TO ZEROS.
```

G Take this path to specify that the lowest value allowable within the usage of the data item be used as the NULL-STRING-ITEM value.

Example:

```
SET NULL-STRING TO LOWER-BOUNDS.
```

H Take this path to specify that the highest value allowable within the usage of the data item be used as the NULL-STRING-ITEM value.

Example:

```
SET NULL-STRING TO UPPER-BOUNDS.
```

I Take this path to specify that a character string consisting of all quotes (") be used as the NULL-STRING-ITEM value.

Example:

```
SET NULL-STRING-ITEM TO QUOTES.
```

J Take this path to specify that the given string be used as the NULL-STRING-ITEM value. The string is repeated as many times as is necessary to fill the data item. The maximum number of characters allowed in the string is 51.

Example:

```
SET NULL-STRING = ALL "*".
```

Any null string items are printed as a string of asterisks in the generated report program.

K Take this path to specify that the value FALSE be assigned as the NULL-BOOLEAN-ITEM value. The default value is FALSE.

Example:

```
SET NULL-BOOLEAN-ITEM TO FALSE.
```

L Take this path to specify that the value TRUE be assigned as the NULL-BOOLEAN-ITEM value.

Example:

```
SET NULL-BOOLEAN TO TRUE.
```

ORDER STATEMENT

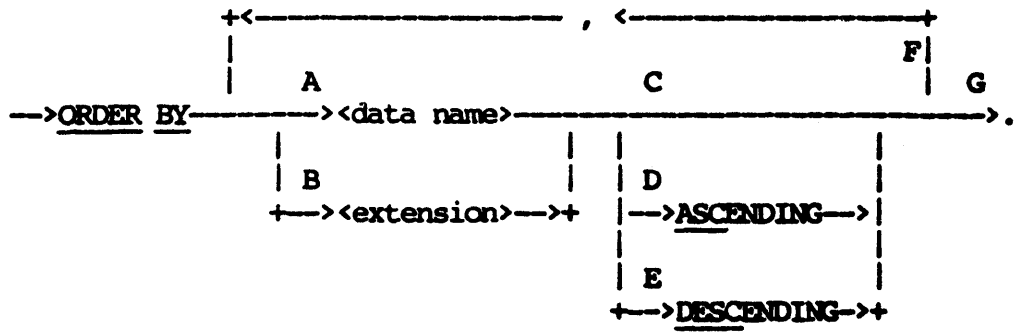
The ORDER statement specifies how logical records are to be ordered based on one or more ordering keys. The ordering is subordinate to a control-break grouping specified elsewhere and must be consistent with any ordering described elsewhere (for example, the ordering described in a REPORT statement).

Example: ("VOCEMP")

```
    ORDER BY EMPYE-NO.
```

This statement specifies that reported information is to be ordered by employee number in ascending sequence.

The syntax for the ORDER statement is as follows:



The paths of this syntax diagram are explained below:

Path	Explanation
A	Take this path to specify that ordering be based on the item referenced by <data name>. The item referenced must be a nonstatistical data item. Example: ("VOCEMP") ORDER BY EMPYE-NO. EMPYE-NO references a nonstatistical data item.

B Take this path to define an <extension> and specify that ordering be based on the derived data item. The item must be nonstatistical.

Example: ("SHIPV")

```
ORDER BY DAY-OF-WEEK WHICH IS
   -1 * AGE(FROM DATE-OF-BILLING TO
      YYDDD 00001) MOD 7 + 1.
```

Transactions are ordered such that DAY-OF-WEEK increases from 1 (Monday) to 7 (Sunday).

C Take this path to specify the default ordering sequence of ASCENDING.

D Take this path to specify ascending sequence.

Example: ("VOCAST")

```
ORDER BY ASSET-NO ASCENDING.
```

E Take this path to specify descending sequence.

Example: ("CLIENT")

```
ORDER BY TOTAL-AMOUNT DESC.
```

Invoices are ordered in descending sequence of total invoice amount.

F Take this path as many times as necessary to specify multiple ordering keys. If more than one ordering key is specified, the first key given is the major key; subsequent keys given are subordinate minor keys. Minor keys are used to order logical records whenever a major key has duplicate values. A maximum of 9 keys can be specified.

Example:

```
ORDER BY LAST-NAME, FIRST-NAME.
```

Records are ordered by FIRST-NAME when LAST-NAMEs are identical.

G Take this path when all ordering keys are specified.

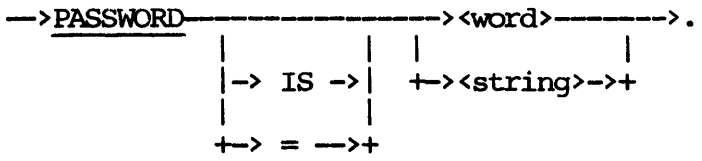
PASSWORD STATEMENT

The PASSWORD statement supplies the password which enables the specified vocabulary to be referenced. This statement is required only if a password was established for the vocabulary by RP3VOC. The password must be given here exactly as it was provided in the RP3VOC run which created the vocabulary. A password can be either a <word> or <string> specification. If the statement is required and you do not enter the password correctly, the REPORTER III Report Language Analysis Program will terminate abnormally when it is run.

Examples:

```
PASSWORD = "QQSV".
PASSWORD = HELLO.
```

The syntax for the PASSWORD statement is as follows:



PRESELECT BOOLEAN EXPRESSION

The Boolean expression that is part of the PRESELECT clause specifies a condition to be evaluated as true or false. As an INPUT statement containing a PRESELECT clause is executed, a data structure is accessed and each record is checked against the PRESELECT Boolean expression. If a record does meet the PRESELECT condition, the record is passed to the logical record used to generate a report.

The PRESELECT Boolean expression can be simple, basic, or complex, and is similar to the Boolean expressions explained under "Expressions" in this section. However, RP3REP does only minimal syntax checking, so the user must be sure to use a PRESELECT Boolean expression which is a valid COBOL74 conditional expression that includes valid relational operators. Refer to the COBOL74 Reference Manual for additional information. Alternatively, REPORTER III relational operators can be used, as shown in the list of valid operators appearing later in this subsection.

SIMPLE BOOLEAN EXPRESSION

A simple Boolean expression is a condition name or a DMSII data name which is of type Boolean. When a condition name is used in a PRESELECT clause, it cannot be equated to a data name.

The following two report specifications show the difference between using a condition name in a SELECT statement and using one in a PRESELECT clause.

Example: ("VOCEMP")

Use in a SELECT Statement:

```
INPUT EMPYE-FILE.
SELECT SEX = MALE.
REPORT EMPYE-NAME ASC, EDUCATION-LEVEL, SALARY.
```

Use in a PRESELECT Clause:

```
INPUT EMPYE-FILE PRESELECT MALE;.
REPORT EMPYE-NAME ASC, EDUCATION-LEVEL, SALARY.
```

BASIC AND COMPLEX BOOLEAN EXPRESSIONS

A basic Boolean expression compares two data items or specifies a condition to be met. Only data items of the same type can be compared.

A complex Boolean expression describes logical operations to be performed on simple or basic Boolean expressions. The logical operators are NOT, AND and OR.

When a basic or complex Boolean expression is used in a PRESELECT clause, either the left or right operand can be a data name, a condition name, a string literal, or a number. An operand which is a data name must be taken from a previously referenced data structure or an accepted data item. The data name cannot be a derived data item.

The following are examples of basic Boolean expressions:

```
COST > 1000.00

or

1000.00 < COST
```

The class tests TRUE, FALSE, and NULL cannot be used in basic and and complex PRESELECT Boolean expressions. However, the following class tests can be used:

- ALPHABETIC
- NUMERIC
- POSITIVE
- NEGATIVE
- ZERO

The following conventions also apply to basic and complex PRESELECT Boolean expressions:

1. Pattern matching is not allowed in the PRESELECT Boolean expression, although it is allowed in the SELECT statement.

2. The optional words IS, TO, and THAN are ignored if they appear in the PRESELECT Boolean expression.

3. Parentheses can be used to ensure the proper sequence of operations.

Valid Operators

The following table lists the arithmetic, relational, and logical operators that are valid for use in basic and complex PRESELECT boolean expressions. The operators MOD and DIV cannot be used in a PRESELECT Boolean expression.

Operator	Meaning
+	Addition
-	Subtraction
*	Multiplication
/	Division
**	Exponentiation
=	Equal to
EQUALS	
<	Less than
LESS	
LT	
LSS	
>	Greater than
GREATER	
GTR	
LEQ	Less than or equal to; equivalent to NOT GREATER THAN
GEQ	Greater than or equal to; equivalent to NOT LESS THAN
NEQ	Not equal
NOT	Negation
AND	Conjunction
OR	Disjunction

PRESELECT CLAUSE

The PRESELECT clause is an option that can be used with the data-structure clause to specify a selection condition in the INPUT statement of a report specification.

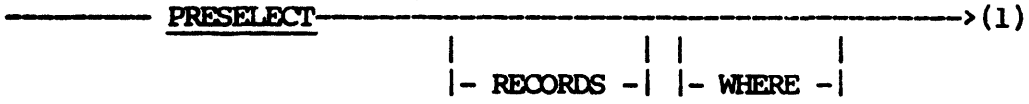
CAUTION

> Read the following pages carefully before using the PRESELECT clause. Improper use could adversely affect the results of the INPUT statement.

The PRESELECT clause can be used most advantageously with multiple data structures that are related to each other. Examples are data structures that are accessed in a one-to-many manner, and a data structure that bears a key into a related data structure that is being accessed.

Using the PRESELECT clause reduces the number of data-structure accesses that have to be done, thereby reducing the I/O time used.

The syntax for the PRESELECT clause is as follows:



The PRESELECT clause includes the PRESELECT Boolean expression, which yields a true or false condition. If the Boolean expression is true, then the selected record is input to the logical record. The record is passed to either the Report Section or the Input Section, depending on whether the ORDER statement is present and/or any control breaks are referenced. If the Boolean expression is false, the logical record is discarded.

When the PRESELECT clause is used with multiple data structures, the physical records of all the data structures will only be moved to the logical record when the PRESELECT Boolean expression yields a true condition. If the condition is false, the normal left-to-right accessing of the data structure stops at the data structure just read, and the next record is accessed from the leftmost data structure at that

level. See "PRESELECT Boolean expression" for more information on its use.

The optional words RECORDS and WHERE are used merely to render the PRESELECT clause more like English in the report specification.

SINGLE DATA-STRUCTURE ACCESS

The PRESELECT clause can be used in the INPUT statement of a report specification when accessing a single data structure such as a file, a data set, or a data base. Using the PRESELECT clause in a single data structure can help to prevent data such as invalid numbers or dates with zero values from being passed to the report program, where exceptions can occur. (For more information on data structures, see the Data-structure Clause in this section.)

After the report program reads a record from the data structure, it checks the result of the PRESELECT Boolean expression. If the condition is true, the requested data items from the data structure are passed from the physical record to the logical record for subsequent processing. If the condition is false, the next record is accessed and the PRESELECT Boolean result is again checked. This process continues until the end of the data structure is reached. Only those physical records which caused a PRESELECT Boolean condition of true are used to create the logical record for the report.

ONE-TO-ONE DATA-STRUCTURE ACCESS

The PRESELECT clause can be used in the INPUT statement of a report specification to indicate one-to-one access of a data structure. In this kind of access, one record of a data structure is accessed for each record of a related or unrelated data structure.

Using the PRESELECT clause in one-to-one accesses can reduce the number of physical records accessed if the access of each stucture is based on the access of a previous structure. This one-to-one dependency is shown in the two examples that follow.

Example 1:

A report that lists managers and their phone numbers is needed. The names of all employees, their employee numbers, and their ranks are kept in a single data structure called EMPLOYEE. The phone numbers are kept

in an indexed system file called PHONE-FILE, with EMPLOYEE NUMBER as the primary key. For this report, the following report specification could be used:

```
INPUT EMPLOYEE, PHONE-FILE AT KEY = EMP-NUM.
SELECT EMPLOYEE-RANK > 9. % MANAGERS ARE 10 AND ABOVE
REPORT EMPLOYEE-NAME AS "NAME OF MANAGER" ASC, PHONE-NUMBER.
```

This report specification causes a read of the PHONE-FILE for each read of the EMPLOYEE data structure. If the report specification is changed to include the PRESELECT clause in the INPUT statement, the report specification might look like the following:

```
INPUT EMPLOYEE PRESELECT EMPLOYEE-RANK > 9;
      , PHONE-FILE AT KEY = EMP-NUM.
REPORT EMPLOYEE-NAME AS "NAME OF MANAGER" ASC, PHONE-NUMBER.
```

Use of the PRESELECT clause will cause a read of the PHONE-FILE to occur only when the EMPLOYEE data structure's EMPLOYEE-RANK field is greater than 9, thereby reducing the number of accesses to the PHONE-FILE data structure.

Example 2:

Using the same entities as in the previous Example 1, assume that PHONE-FILE is a DMSII data structure. The report specification without the PRESELECT clause might look like the following:

```
INPUT EMPLOYEE, PHONE-FILE FROM EMPLOYEE VIA EMP-NUM-SET
      AT PHONE-EMP-NUM = EMP-NUM.
SELECT EMPLOYEE-RANK > 9;
REPORT EMPLOYEE-NAME AS "NAME OF MANAGER" ASC, PHONE-NUMBER.
```

Using the PRESELECT clause to reduce the accesses to PHONE-FILE, the report specifcation could be written as follows:

```
INPUT EMPLOYEE  PRESELECT EMPLOYEE-RANK > 9;
       PHONE-FILE FROM EMPLOYEE VIA EMP-NUM-SET
          AT PHONE-EMP-NUM = EMP-NUM.
REPORT EMPLOYEE-NAME AS "NAME OF MANAGER" ASC, PHONE-NUMBER.
```

Example 3:

The PRESELECT clause may be used to produce a result similar to that of the SELECT statement, with better I/O time and fewer data-structure accesses. However, you must be careful in constructing the PRESELECT clause so a SELECT statement is not converted incorrectly.

This example shows the use of a SELECT statement, the correct use of the PRESELECT clause, and an incorrect conversion to the PRESELECT clause.

The example involves four data structures which are accessed in the one-to-one manner with no structure depending on a structure to its left. These data structures are named A, B, C, and D, and they contain the fields A-1, B-1, C-1, and D-1 respectively. Data structures A and C each contain three records having the values A-1-1 through A-1-3 and C-1-1 through C-1-3 respectively. Data structures B and D each contain four records having the values B-1-1 through B-1-4 and D-1-1 through D-1-4 respectively.

Specifying One-to-One Access

The following report specification contains an INPUT statement indicating that a one-to-one access of the A, B, C, and D data structures will occur:

```
INPUT A, B, C, D.
REC-COUNT IS RUNNING-COUNT.
REPORT A-1, B-1, C-1, D-1, REC-COUNT.
```

The report generated from the preceding report specification would contain the following records:

A-1	B-1	C-1	D-1	REC-COUNT
A-1-1	B-1-1	C-1-1	D-1-1	1
A-1-2	B-1-2	C-1-2	D-1-2	2
A-1-3	B-1-3	C-1-3	D-1-3	3
	B-1-4		D-1-4	4

The generated program accesses the data structures from left to right. When data structures A and C have no records remaining, null values are placed into the logical record instead of the values A-1-4 and C-1-4 respectively. This process continues until all of the data structures

are out of records. The INPUT statement used in the example results in all data structures being at the same level, which is expressed as level 1.

Using the SELECT statement

If the report needs only information that meets the conditions A-1 = "A-1-2" and C-1 = "C-1-2", the following SELECT statement could be added to the report specification:

```
INPUT A, B, C, D.
SELECT A-1 = "A-1-2" AND C-1 = "C-1-2".
REC-COUNT IS RUNNING-COUNT.
REPORT A-1, B-1, C-1, D-1, REC-COUNT.
```

This report specification causes all the records in the data structures A, B, C, and D to be passed into the logical record before the selection conditions are applied. The following record appears in the report:

A-1	B-1	C-1	D-1	REC-COUNT
A-1-2	B-1-2	C-1-2	D-1-2	1

Using the PRESELECT Clause

To achieve the same result as the INPUT statement containing the SELECT statement, the report specification should be written as follows:

```
INPUT A, B, C, D PRESELECT A-1 = "A-1-2" AND
                           C-1 = "C-1-2";.
REC-COUNT IS RUNNING-COUNT.
REPORT A-1, B-1, C-1, D-1, REC-COUNT.
```

This is not a highly efficient use of the PRESELECT clause; however, the report produced will look identical to the one produced using the SELECT statement.

Incorrect Conversion Using the PRESELECT Clause

Care should be exercised when using the PRESELECT clause to convert a SELECT statement. The following information explains how an attempt to convert the SELECT statement into PRESELECT clauses could result in the incorrect INPUT statement that follows.

```
INPUT A PRESELECT RECORDS WHERE A-1 = "A-1-2";,
      B, C PRESELECT C-1 = "C-1-2";,  D.
REC-COUNT IS RUNNING-COUNT.
REPORT A-1, B-1, C-1, D-1, REC-COUNT.
```

The selection conditions specified by the PRESELECT clauses are the same as the ones specified by the SELECT statement. However, the placement of the selection conditions within the INPUT statement skews the reading of the files, so that the desired record is never found. Instead, this INPUT statement causes the generated report program to read the physical records and form the logical records as shown in the following chart:

A	B	C	D	LOGICAL RECORD
A-1-1				(no logical record formed)
A-1-2	B-1-1	C-1-1		(no logical record formed)
A-1-3				(no logical record formed)
	B-1-2	C-1-2	D-1-1	NULL B-1-2 C-1-2 D-1-1
	B-1-3	C-1-3		(no logical record formed)
	B-1-4		D-1-2	NULL B-1-4 NULL D-1-2
			D-1-3	NULL NULL NULL D-1-3
			D-1-4	NULL NULL NULL D-1-4

The logical records shown in the chart are completely different from the logical records that would result from the SELECT statement. The following list describes in detail the actions caused by the INPUT statement containing the PRESELECT clauses:

1. Data structure A would be accessed.

2. The data from A-1 would be checked against the PRESELECT Boolean expression and would cause a false condition.

3. The false result causes the next record from data structure A to be accessed and checked. This time the result of the PRESELECT Boolean check is true.

4. The next data stucture, named B in this example, is accessed.

5. Continuing from left to right, data stucture C is accessed and the PRESELECT Boolean condition is false.

6. The false result causes a return to the leftmost structure of the level. In this case, all structures are at level 1, so the leftmost structure is data structure A.

7. Data structure A is accessed and a check of the PRESELECT Boolean expression results in a false condition.

8. The next access of A finds A exhausted of records.

9. With no records in A, B is accessed, then C.

10. The data from C is checked against the PRESELECT Boolean expression and results in a true condition.

11. The true condition allows the left-to-right accessing to continue to data structure D.

12. The completion of access to D results in the first logical record being built, with a null value added for the A data structure.

13. Data structure B is accessed, then C. The data from C results in a false condition when checked against the PRESELECT Boolean expression.

14. The false result causes a return to the left so that the next record in B is accessed. C is found exhausted of records, so D is accessed and the next logical record is formed with null values for A and C.

15. The next access of B finds it exhausted of records. So D is accessed and the next logical record is formed with null values for the A, B, and C data structures.

16. D is accessed and another logical record is formed.

17. The next access finds all of the data structures exhausted of records, so accessing is discontinued.

The report produced looks completely different than the one specified and produced using the SELECT statement. The following illustration shows the report produced using the INPUT statement containing the PRESELECT clauses.

A-1	B-1	C-1	D-1	REC-COUNT
	B-1-2	C-1-2	D-1-1	1
	B-1-4		D-1-2	2
			D-1-3	3
			D-1-4	4

ONE-TO-MANY DATA-STRUCTURE ACCESS

The PRESELECT clause can be used in the INPUT statement of a report specification to indicate one-to-many access. In this kind of access, one record of a data structure is accessed for each record of a related or unrelated data structure.

The following examples explain how the SELECT statement and the PRESELECT clause can be used in an INPUT statement that specifies one-to-many access of three data structures called A, B, and C. These data structures contain the fields A-1, B-1, and C-1 respectively. Each of the data structures contains three records with the values A-1-1 through A-1-3, B-1-1 through B-1-3, and C-1-1 through C-1-3 respectively.

Example:

The following report specification indicates one-to-many access of the data structures A, B, and C.

```
INPUT A(B(C)).
REC-COUNT IS RUNNING-COUNT(C-1).
REPORT A-1, B-1, C-1, REC-COUNT.
```

The report generated by the preceding report specification would contain the following information:

A-1	B-1	C-1	REC-COUNT
A-1-1	B-1-1	C-1-1	1
A-1-1	B-1-1	C-1-2	2
A-1-1	B-1-1	C-1-3	3
A-1-1	B-1-2	C-1-1	4
A-1-1	B-1-2	C-1-2	5
A-1-1	B-1-2	C-1-3	6
A-1-1	B-1-3	C-1-1	7
A-1-1	B-1-3	C-1-2	8
A-1-1	B-1-3	C-1-3	9
A-1-2	B-1-1	C-1-1	10
A-1-2	B-1-1	C-1-2	11
.	.	.	.
.	.	.	.
A-1-3	B-1-3	C-1-2	26
A-1-3	B-1-3	C-1-3	27

The INPUT statement "INPUT A(B(C))" results in data structure A being at level 1, B at level 2, and C at level 3. Level 3 is considered the innermost level because it provides more input to the report than either of the other levels used in the example. Level 1 is considered to be the outermost level.

The generated program accesses the data structures from left to right. The first time through, data stucture A is accessed, then B, then C. This results in the formation of the first logical record. One-to-many access causes the next read from the leftmost structure of the innermost level. In this example, level 3 is both the innermost level and the leftmost structure at that level.

The data structures in outer levels are not accessed. This read of level 3 forms the next logical record. The innermost level will continue to be accessed, forming logical records until the data structure is exhausted of records.

When the innermost level is exhausted of records, the generated program accesses the next record from the next outer level. The innermost level is restored to the first record and that record is accessed, forming the next logical record. As each level runs out of records, the next record from the next outer level is accessed until level 1, the outermost level, is exhausted.

In this example, data structure A would input three records, B would input three records three times, and C would input three records nine times.

Using the SELECT statement

If the report needs only information that meets the conditions A-1 = "A-1-2" and B-1 = "B-1-1", then the following SELECT statement could be added to the report specification:

```
INPUT A(B(D)).
SELECT A-1 = "A-1-2" AND B-1 = "B-1-1".
REC-COUNT IS RUNNING-COUNT.
REPORT A-1, B-1, C-1, D-1, REC-COUNT.
```

This SELECT statement is tested only after each logical record is formed. It does not reduce the number of accesses to the data structures. However, the number of accesses could be reduced by using the PRESELECT clause.

Using the PRESELECT Clause

The INPUT statement containing the PRESELECT clause could be written as follows:

```
INPUT A PRESELECT RECORDS WHERE A-1 = "A-1-2";
      (B PRESELECT B-1 = "B-1-1";
       (C)).
REC-COUNT IS RUNNING-COUNT.
REPORT A-1, B-1, C-1, D-1, REC-COUNT.
```

This statement would cause the following actions to occur:

1. Data structure A is accessed.
2. The data from field A-1 is checked against the PRESELECT Boolean expression and causes a false condition.
3. The false result causes the next record from data stucture A to be accessed and checked. This time the condition is true.
4. The true result causes the generated program to advance to the next data structure to the right, which is B in this example.

5. The PRESELECT Boolean check of data from data structure B causes a true condition.

6. The true result causes data structure C to be read, and the first logical record to be formed. The logical record would appear as follows:

A-1	B-1	C-1	REC-COUNT
A-1-2	B-1-1	C-1-1	1

7. After data structure C is exhausted of records, the next record from data structure B is accessed and the PRESELECT Boolean check yields a false result.

8. The false condition causes the next record in data structure B to be accessed. That record also yields a false result when checked.

9. The next access of B finds the file exhausted of records, so the next record in A is accessed.

10. The PRESELECT Boolean check for A yields a false result.

11. The false result causes the next access of A, which finds the data structure exhausted. The logical record created would contain the following information:

A-1	B-1	C-1	REC-COUNT
A-1-2	B-1-1	C-1-1	1
A-1-2	B-1-1	C-1-2	2
A-1-2	B-1-1	C-1-3	3

The following chart compares the physical records accessed in this example with the logical records that are formed.

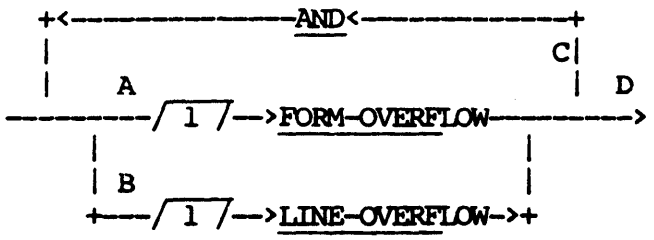
A	B	C	LOGICAL RECORD		
A-1-1					
A-1-2	B-1-1	C-1-1	A-1-2	B-1-1	C-1-1
		C-1-2	A-1-2	B-1-1	C-1-2
		C-1-3	A-1-2	B-1-1	C-1-3
	B-1-2				
	B-1-3				
A-1-3					

The preceding example shows that using the PRESELECT clause reduces the number of data-structure accesses required to create the correct logical record.

PRINT EXCEPTIONS

Print Exceptions is used to specify that form overflow and/or line overflow are to be noted as errors or exceptions.

The syntax for Print Exceptions is as follows:



The paths of this syntax diagram are explained below:

Path	Explanation
A	If you take this path, form overflow results in an error or exception. Form overflow results when a print specification other than a NEW FORM specification causes printing to overflow a form.

Example:

```
PRINT NOTING FORM-OVERFLOW FOR EACH FORM HEADER
   ["C U S T O M E R   A C T I V I T Y"]
      AT LINE 4 POSITION 14,
   ["PAGE"] AT 55,
      .
      .
      .
```

Form overflow occurs when the print specification causes printing to go beyond the last printable line of the form.

B If you take this path, line overflow results in an error or exception. Line overflow occurs when a print specification other than a NEW FORM, NEW LINE, or BLANK LINE specification causes printing to overflow a line.

Example:

```
PRINT AS FOLLOWS NOTING LINE-OVERFLOW:
   FOR EACH EMPLOYEE-NO BLANK LINE,
      [EMPLOYEE-NAME], 2 BL;
   FOR EACH RECORD 2 SPACES, [PREV-SUPERVISOR].
```

Each logical record contains information on an employee's previous job within the company. If an employee has had more than six previous jobs within the company, a line overflow exception occurs as a result of the above PRINT statement. (Default FORM-WIDTH is 132, and PREV-SUPERVISOR references a name which is 20 characters in length.)

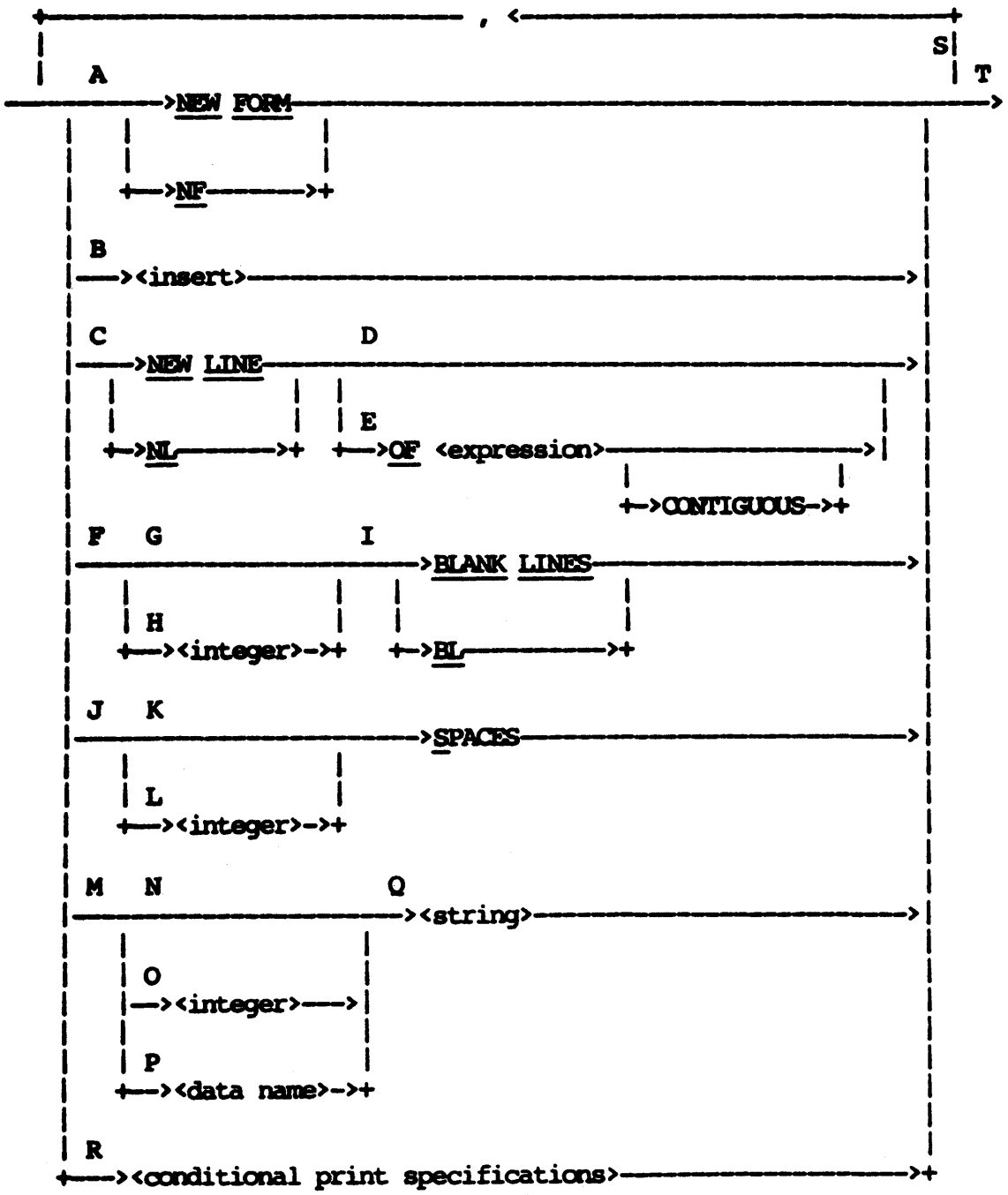
C Take this path to specify that both form overflow and line overflow are to be noted as errors or exceptions.

D Take this path after you have specified all print exceptions.

PRINT SPECIFICATIONS

Print Specifications are used to specify where and/or how information is to be printed.

The syntax for Print Specifications is as follows:



The paths of this syntax diagram are explained below:

Path	Explanation

A NEW FORM, or NF, specifies that printing begins on a new form (new page). Appropriate form headers and footnotes which you specified are printed. The next available print position is 1. If printing already begins on a new form as a result of previous print specifications, no form is skipped as a result of this specification.

Example:

NF, NF

The above print specifications do not result in a form skip.

Example:

NF, BLANK LINES, NF

The above print specifications result in a form which is blank except for appropriate form headers and footnotes which are specified.

B An <insert> specifies the printing of an item, a constant string, a form number, a page number, a date, or a time. The entity inserted must not overflow a line. If the entity does not fit properly on the line, an error or exception results.

The next available print position is the position immediately following the insert.

Example: ("CLIENT")

[CUSTOMER-NAME]

The above print specification causes the customer name to be inserted properly on the form.

C NEW LINE, or NL, specifies that printing begins on a new line. The next available print position is 1.

D Take this path if a series of adjoining (contiguous) lines on the same form need not be assured. If printing already begins on a new line as a result of previous print specifications, no line is skipped as a result of this specification.

Appropriate form headers and footnotes which were specified are printed if printing must be continued onto a new form.

This occurs when LINE-NUMBER > FORM-LENGTH - (number of lines allocated for form footnotes).

Example: ("VOCAST")

```
PRINT AS FOLLOWS:

   FOR EACH RECORD NL, [ASSET-DESC], [COST]
      AT 40, [ASSET-LIFE] AT 59; ...
```

The above PRINT statement specifies that information for each logical record is to begin on a new line.

E Take path E if a series of adjoining (contiguous) lines on the same form is required. The <expression> specifies the number of adjoining lines; it must be numeric and is truncated, if necessary, to an integer value.

If printing already begins on a new line as a result of previous print specifications and if the specified number of adjoining lines are available on the current form, no form is skipped as a result of this specification.

Appropriate form headers and footnotes which were specified are printed if printing must be continued onto a new form. This occurs when there are less than <expression> blank lines left on the form before any footnotes.

Example: ("INVENT")

```
PRINT AS FOLLOWS: ...

   FOR EACH DEPARTMENT SUM NL OF 3, BL,
      "TOTAL QUANTITY -", [TOTAL(QUANTITY)], NL,
      "TOTAL VALUE    -",
      [TOTAL(INVENT-DOLLAR-VALUE)]
```

In the above statement, the "NL OF 3" specification assures that the department summary is never split across forms.

F Take this path to specify the printing of one or more blank lines. Following this specification, the next available print position is 1.

If printing must be continued onto a new form as a result of this specification, appropriate form headers and footnotes which were specified are printed. Specified blank lines which cannot be printed at the bottom of a form because of overflow are suppressed when printing continues at the top of the overflow form.

Example:

```
PRINT AS FOLLOWS: ...

   FOR EACH RECORD BLANK LINE, [NAME], [AGE] AT
      35, [SALARY] AT 45; ... .
```

The above PRINT statement specifies that two lines are printed for each employee record; the first line is blank. If the BLANK LINE specification causes form overflow, printing continues on the next form, and the blank line is not printed.

G Take this path to specify one blank line.

H Take this path to specify <integer> blank lines.

I BLANK LINES and BL are synonymous.

J Take this path to specify the printing of one or more spaces, beginning at the available print position. The next available print position following this specification is the position immediately following the last space printed.

If necessary, printing continues onto a new line as a result of this specification. Additional printing of spaces on the overflow line is suppressed, and the next available print position is 1. If printing must continue onto a new form, appropriate form headers and footnotes which are specified are printed.

Example: ("INVENT")

```
PRINT(FORM-WIDTH = 32, FORM-LENGTH = 8): NF, BL,
   8 S, "PART INFORMATION",... .
```

The above statement specifies that PART INFORMATION appears on the second line of the form, eight spaces from the left.

K Take this path to specify one space.

L Take this path to specify <integer> spaces.

M Take this path to specify the printing of a string of characters, beginning at the next available print position. The next available print position following this specification is the position immediately following the last printed character.

Printing continues onto a new line if necessary to print all specified characters. If printing must continue onto a new form, appropriate form headers and footnotes which are specified are printed.

N If you take this path, the specified <string> is printed once.

Example:

```
PRINT(FORM-WIDTH = 32, FORM-LENGTH = 8): NF, BL,
   8 S, "PART INFORMATION", ... .
```

PART INFORMATION is printed on the second line of the form. The next available print position is position 25 of line 2.

O If you take this path, the specified <string> is repeated <integer> times.

Example: ("VOCAST")

```
PRINT AS FOLLOWS: ...

   FOR FINAL SUMMARY NL, 35 SPACES, 14 "-", NL,
      [TOTAL(COST)] IN 15 AT 35.
```

The underlined portion of the above statement underlines a column of figures which is totaled.

P If you take this path, the value of the referenced data item specifies the number of times the <string> is repeated. The data item must be numeric and is truncated, if necessary, to an integer value.

Example: ("VOCEM")

```
      .
      .
      .

   GROUP BY RANGES OF EMPYE-AGE FROM 20 BY 5 TO 70
      AS AGE-RANGE.
   NO-OF-EMPLOYEES IS COUNT(FOR EACH AGE-RANGE).
   RELATIVE-VALUE IS (NO-OF-EMPLOYEES/
      MAX(NO-OF-EMPLOYEES)) * 50.
   PRINT AS FOLLOWS: ...
      FOR EACH AGE RANGE SUMMARY NL, (AGE-RANGE),
         2 S, [NO-OF-EMPLOYEES],
         2 S, RELATIVE-VALUE "*".
```

The above PRINT statement results in a histogram, or bar graph, type report. From 0 to 50 "*"s are printed for each range of age.

Q Take this path to specify the string of characters which is to appear on the form.

R Take this path to describe the layout of a segment of the report which varies according to a stated condition.

Example:

```
   PRINT: ...
      FOR EACH RECORD NL, [NAME], 5 SPACES,
         IF SALARY < 2000
         THEN ([SALARY] INTO 10 POSITIONS)
         ELSE (10 SPACES), 5 SPACES,
         [AGE]; ... .
```

The <conditional print specification> in the above statement causes suppression of salaries within the report which are 2,000 dollars or over.

Example: ("SHIPV")

```
REPLACE PRINT-P-O-S(I)
      BY IF POINTS-OF-SHIPMENT(I)
   NEQ 0 (NL, [POINTS-OF-SHIPMENT[I]] AT 15) #.
.
.
.
PRINT:  ...
   FOR EACH RECORD NL, [NUMBER OF BILL],
   [POINTS-OF-SHIPMENT[1]]
      AT 15, [POINT-OF-DESTINATION] AT 30,
      PRINT-P-O-S(2),
      PRINT-P-O-S(3),
      PRINT-P-O-S(4); ...  .
```

The macro PRINT-P-O-S represents a <conditional print specification>. The macro is used three times in the above PRINT statement to provide for the possible printing of three additional points of shipment. The report appears as follows:

NUMBER	SHIPMENT POINTS	DESTINATION	Specifications for header not given
00010211	02399110 02398112	02399350	RECORD1
00010212	02397766	02397660	RECORD2
00010213	02391000 02398113 02391177 02391203	11396770	RECORD3
	.	.	.
	.	.	.
	.	.	.

S Take this path to give additional print specifications.

T Take this path after all print specifications for the portion of the report are given.

PRINT STATEMENT

The PRINT statement specifies how information is to be printed onto forms. The statement facilitates the printing of special forms such as confirmation letters, mailing labels, or Burroughs Mailer Sets. The statement also provides a very flexible means to specify a user-formatted report which is printed on a standard form or displayed on a terminal device. For this latter use, a page or a screen is simply treated as a single form.

The structure of the PRINT statement allows the definition of the form to be given within the statement or conveniently specified via a macro and merely referenced within the statement.

Example: ("CLIENT")

```
PRINT MP-1 AS FOLLOWS:

   FOR EACH RECORD NEW FORM, 2 BLANK LINES,
                       [CUSTOMER-NAME],
                   NEW LINE, [STREET-ADDRESS],
                   NEW LINE, [CITY-STATE],
                   NEW LINE, [ZIP-CODE].
```

The above statement specifies the printing of standard Burroughs 1-up mailing labels. MP-1 references a macro which defines the characteristics of the mailing labels. One mailing label is printed as described for each reported logical record or customer. The statement could be specified in a more abbreviated manner as follows:

```
PRINT MP-1: NF, 2 BL, [CUSTOMER-NAME], NL,
   [STREET-ADDRESS], NL, [CITY-STATE], NL,
      [ZIP-CODE].
```

Example: ("CLIENT")

```
PRINT NEGATIVE-CONFIRM:

   FOR EACH RECORD NEW FORM,
      [CUSTOMER-NAME] AT LINE 1 POSITION 1,
      [STREET-ADDRESS] AT LINE 3 POSITION 1,
      [CITY-STATE] AT 5 1,
      [ZIP-CODE] AT 7 1,
      ["03/31/77"] AT 15 2,
      ["MR. JOHN DOE, VICE PRESIDENT"] AT 24 1,
      ["TOTAL BALANCE DUE - "] AT 26 4,
      [BALANCE-DUE] AT 26 24,
      ["XYZ SAVINGS & LOAN"] AT 28 1,
      ["702 MAIN"] AT 30 1,
      ["IRVINE, CALIF.  92714"] AT 32 1.
```

The above statement specifies the printing of negative confirmation letters. NEGATIVE-CONFIRM references a macro which defines both attributes and text of a multipurpose negative confirmation. One confirmation letter is printed as described for each logical record.

Example: ("INVENT").

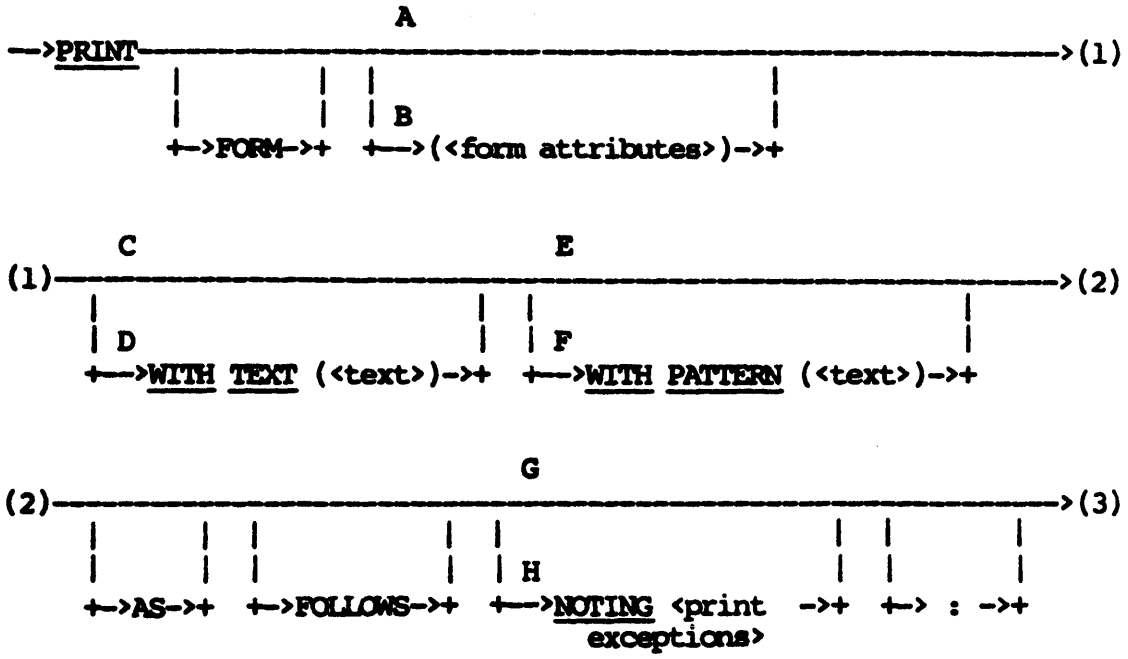
```
GROUP BY BIN-NO.
    .
    .
    .

PRINT AS FOLLOWS:

   FOR EACH BIN-NO HEADER 2 BLANK LINES,
       "*** BIN ", [BIN-NO], 1 BLANK LINE;
   FOR EACH RECORD 6 SPACES, [PART-NO].
```

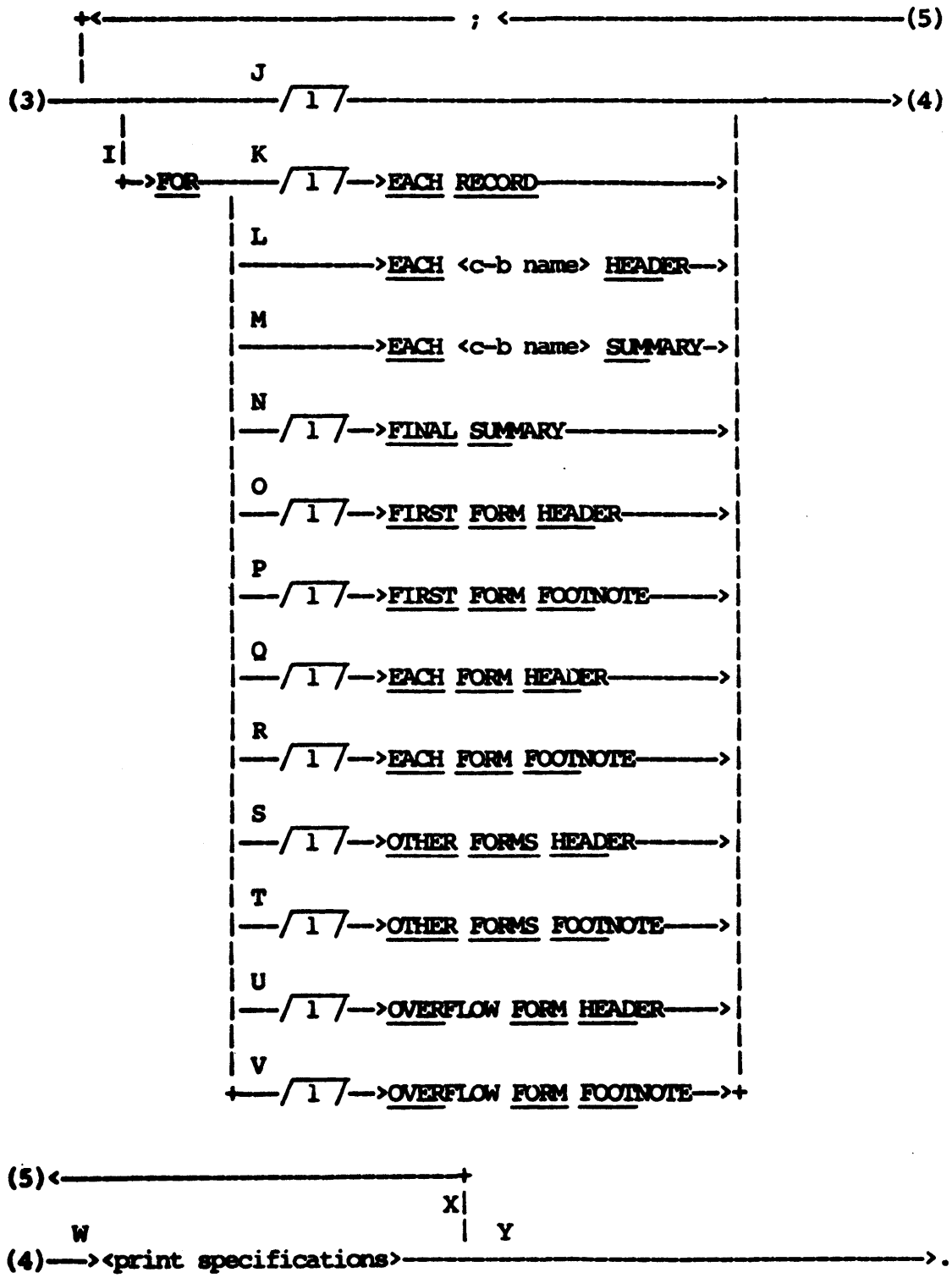
A report of the type shown in Figure 4-4 is required. The above PRINT statement formats the report exactly as required. A standard default form is used.

```
*** BIN 001

     013221   053111   532100   711300   123456   175123   177775   332117   176631   866631   188000
     013005   030052   411002   110555   777811   000888   431122   332180   187766   311000   226601
     001322   554411

*** BIN 002

     111000   773116   881100   117755

*** BIN 003

     000556   077731   899100   877766   110000   001330   222111   007777   886611   337703   821150
     001320   667001   336600

      .
      .
      .
```

Figure 4-4. A User-Formatted Report

The syntax for the PRINT statement is as follows:




; <
(5)
J
(3)
1
(4)
I
K
FOR
1
EACH RECORD
L
EACH <c-b name> HEADER
M
EACH <c-b name> SUMMARY
N
1
FINAL SUMMARY
O
1
FIRST FORM HEADER
P
1
FIRST FORM FOOTNOTE
Q
1
EACH FORM HEADER
R
1
EACH FORM FOOTNOTE
S
1
OTHER FORMS HEADER
T
1
OTHER FORMS FOOTNOTE
U
1
OVERFLOW FORM HEADER
V
1
OVERFLOW FORM FOOTNOTE
(5)
X
W
Y
(4)
<print specifications>
.

The paths of this syntax diagram are explained below:

Path	Explanation

A Take this path if no change in default form attributes is required. The following default form attributes are assumed. Refer to FORM ATTRIBUTES in this section for a complete explanation of these attributes.

Defaults:

FORM-WIDTH	132
FORM-LENGTH	54
START-POINT	1
VERTICAL-SPACING	CHANNEL 1
FORM-TYPE	STANDARD

B Take this path to change one or more default form attributes. Any attribute which is not given explicitly is assumed to have its default value.

Example:

```
PRINT FORM (START-POINT = 10,
   FORM-WIDTH = 65,
   FORM-TYPE = SPECIAL)...
```

The above attributes are given to describe the form suitable for printing 8 1/2 x 11 inch confirmation letters. Implicit attributes are FORM-LENGTH = 54 and VERTICAL-SPACING = CHANNEL 1.

C Take this path when the desired standard text is preprinted on each form or when no standard text is desired for each form. In the latter case, all text printed onto a form is supplied via the <print specifications>.

D Take this path to specify a standard text for each form. Specifying standard text for a form is similar to designing a preprinted form; the given text is preprinted as described onto each form by the generated report program.

Figure 4-5 shows the use of this feature for the composition of a confirmation form letter.

```
PRINT FORM(START-POINT=10, FORM-WIDTH=65, FORM-TYPE=SPECIAL) WITH
      TEXT(
10 "DEAR SIR:",
12 "OUR AUDITORS ARE MAKING THEIR REGULAR EXAMINATION OF OU"
                                                        "R          ",
13 "ACCOUNTS.  IN REGARD TO THAT EXAMINATION, YOUR ACCOUNT "
                                                        "HAS BEEN   ",
14 "SELECTED FOR CONFIRMATION.  PLEASE COMPARE THE INFORMAT"
                                                        "ION STATED ",
15 "BELOW TO YOUR RECORDS AS OF          .  TRANSACTIONS SIN"
                                                        "CE THAT    ",
16 "DATE ARE NOT TO BE CONSIDERED.  IF THE INFORMATION IS N"
                                                        "OT IN      ",
17 "AGREEMENT WITH YOUR RECORDS, PLEASE STATE THE DIFFERENC"
                                                        "E ON THE   ",
18 "REVERSE SIDE AND RETURN THIS FORM DIRECTLY TO OUR AUDIT"
                                                        "ORS.       ",
20 "IF CORRECT, NO REPLY IS NECESSARY.",
22 "VERY TRULY YOURS,"
        ) AS FOLLOWS:  FOR EACH RECORD NEW FORM,
[CUSTOMER-NAME] AT 1 1, [STREET-ADDRESS] AT 3 1, [CITY-STATE] AT
5 1, [ZIP-CODE] AT 7 1, ["03/31/83"] AT 15 29,
["MR. JOHN DOE, VICE PRESIDENT"] AT 24 1, ["TOTAL BALANCE DUE - "
] AT 26 4, [BALANCE-DUE] AT 26 24, ["XYZ SAVINGS & LOAN"] AT 28
1, ["702 MAIN"] AT 30 1, ["IRVINE, CALIF.   92714"] AT 32 1.
```

Figure 4-5. PRINT Statement (with Specified Text) for Confirmation Form Letter

E Take this path when no form pattern is to be defined.

F Take this path when a form pattern is to be defined. A form pattern is a dummy-form image that is printed before the actual printing of forms to help you determine proper alignment of the forms on the printer. The number of times this form pattern is printed is given in the SET FORM-PATTERNS specification. If SET FORM-PATTERNS is not specified and pattern text is given, no pattern printing occurs. If SET FORM-PATTERNS is specified and pattern text is not given, an error results.

G Take this path if line and/or form overflow resulting from subsequent <print specifications> in path I is not to be flagged as errors or exceptions. Subsequent <print specifications> describe printing in a top-down, left-to-right manner. When print specifications result in printing beyond the right-most position of the form, printing continues on position 1 of the next line. When print specifications result in printing beyond the last available line on the form, printing continues on position 1 of the next available line or the top of the next form.

In many cases, the type of <print specifications> given guarantees that line overflow and form overflow cannot occur.

H Take this path to specify that line and/or form overflow is to be detected as an error or noted as an exception. In many cases, overflow can be detected by an analysis of the PRINT statement and flagged as an error. If overflow results in a run-time exception, printing is discontinued.

I Take this path to specify how one or more portions of the user-formatted report are to appear.

Example: ("CLIENT")

```
PRINT MP-1 AS FOLLOWS:

   FOR EACH RECORD NEW FORM, 2 BLANK LINES,
                       [CUSTOMER-NAME],
                   NEW LINE, [STREET-ADDRESS],
                   NEW LINE, [CITY-STATE],
                   NEW LINE, [ZIP-CODE].
```

In the above statement, the portion of the report corresponding to each logical record is the only portion specified. As specified, this portion always begins on a new form and corresponds to a single form.

Example: ("INVENT")

```
PRINT AS FOLLOWS:
   FOR EACH BIN-NO HEADER 2 BLANK LINES,
      "*** BIN ", [BIN-NO], 1 BLANK LINE;
   FOR EACH RECORD 6 SPACES, [PART-NO].
```

The above PRINT statement specifies the printing of a report consisting of two portions: a control-break header for each bin, and a portion for each logical record or part. (The specified report is shown in Figure 4-5.)

J Take this path (or path K) to describe how information is to appear for each logical record reported. Path J implicitly specifies a "FOR EACH RECORD" clause and cannot be taken if the "FOR EACH RECORD" option (path K) was taken previously.

Example: ("CLIENT")

```
PRINT MP-1: NF, 2 BL, [CUSTOMER-NAME], NL,
   [STREET-ADDRESS], NL, [CITY-STATE], NL,
      [ZIP-CODE].
```

The print specifications given in the above PRINT statement are assumed applicable to each logical record.

K Take this path (or path J) to describe how information is to appear for each logical record reported. Path K explicitly specifies a "FOR EACH RECORD" clause and cannot be taken if the implicit "FOR EACH RECORD" option (path J) was taken previously.

Example: ("CLIENT")

```
PRINT MP-1: FOR EACH RECORD NF, 2 BL,
   [CUSTOMER NAME], NL, [STREET-ADDRESS], NL,
      [CITY-STATE], NL, [ZIP CODE].
```

The above PRINT statement is identical in meaning to the PRINT statement given in the example for path J.

L Take this path to specify the printing of a header containing information related to a control-break item. <c-b name> must reference a previously defined control-break item.

The specified <c-b name> defines the default scope for statistical functions specified in the subsequent <print specifications>.

All items printed via the <print specifications> for the header must be summary items related to the specified control-break item. Nonstatistical items printed in the header are declared implicitly as summary items.

Example: ("INVENT")

```
PRINT AS FOLLOWS:

   FOR EACH BIN-NO HEADER 2 BL, "*** BIN ",
      [BIN-NO], BL;
   FOR EACH RECORD 6 S, [PART-NO].
```

A 4-line header is specified for each bin number in the above PRINT statement (see Figure 4-4).

M Take this path to specify the printing of summary information related to a control-break item. <c-b name> must reference a previously defined control-break item.

The specified <c-b name> defines the default scope for statistical functions specified in the subsequent <print specifications>.

The summary is printed after the last logical record related to each value of the control-break item is printed, and after all summaries for any lower-level control breaks are printed.

Each item printed via the <print specifications> for the summary must be a summary item related to the specified control-break item. Nonstatistical items printed in the summary are declared implicitly as summary items. Using the <c-b name> in the summary is strongly discouraged; the value contained in the <c-b name> can change before the summary is printed.

Example: ("CLIENT")

```
PRINT:

   FOR EACH INV-CUST-NO SUMMARY NEW LINE,
      "TOTAL", 30 SPACES, [TOTAL (AMOUNT-PAID)]
      IN 10;
   FOR EACH RECORD:  NEW LINE, [INVOICE-NO],
      24 SPACES, [AMOUNT-PAID] IN 10;... .
```

In the above PRINT statement, a 1-line summary is specified for each customer number. The line contains the total amount paid for all listed invoices placed appropriately under the amount-paid column.

N Take this path to specify the printing of a summary for all reported information. The summary is printed after the last logical record is printed and after any specified control-break summaries are printed.

All items printed via the <print specifications> for the FINAL summary must be summary items related to all information reported. Nonstatistical items printed in the FINAL summary are declared implicitly as summary items.

Example: ("INVENT")

PRINT AS FOLLOWS:

```
FOR FINAL SUMMARY NF, 11 S, "QUANTITY", 3 S,
   "DOLLAR VALUE", BL, "TOTAL", 4 S,
   [TOTAL(QUANTITY)] IN 9, 1 S,
   [TOTAL(INVENT-DOLLAR-VALUE)] IN 15, BL,
   "AVERAGE", 2 S, AVERAGE(QUANTITY)] IN 9,
   [AVERAGE(INVENT)-DOLLAR-VALUE)] IN 15.
```

In the above PRINT statement, a final summary is specified which appears on the last form as follows:

	QUANTITY	DOLLAR VALUE
TOTAL	51662	317721.06
AVERAGE	13	71.28

- Take this path to describe the printing of a header for the first form. The FIRST FORM HEADER always begins at the top of the first form; a NEW FORM print specification is implicit and must not be given. You cannot take this path if the EACH FORM HEADER option was specified previously.

  A header cannot continue onto a new form. That is, the <print specifications> for the header cannot cause form overflow, and it cannot contain a NEW LINE specification or a NEW FORM specification which would continue the header onto another form.

  The values of any items which appear in the header are obtained from the first reported logical record.

  Example: ("INVENT")

  PRINT:

  ```
  FOR FIRST FORM HEAD 52 S, "LIST OF
     PART NUMBERS BY BIN", 3 BL; ... .
  ```

  The above PRINT statement specifies a report title which is centered at the top of the first page.

P Take this path to specify a footnote for the first form. The footnote appears on the very bottom lines of the first form. You cannot take this path if the EACH FORM FOOTNOTE option was specified previously.

A footnote always begins on a new line; a NEW LINE print specification is implicit and need not be given. <print specifications> for a footnote cannot contain any of the following:

1. NEW FORM specification.

2. NEW LINE specification which would continue the footnote onto a new form.

3. <Data name> <string> specification.

4. <Conditional print specification>.

The footnote begins on LINE-NUMBER = FORM-LENGTH - (number of lines specified for the footnote) + 1, and it continues to the last position on the form.

The values of any items which appear in the footnote are taken from the logical record being printed at the time a print specification or form overflow causes the form footnote to be printed.

Example: ("CLIENT")

```
PRINT:

   ... FOR FIRST FORM FOOT "* - 'RISK RATING'
       INDICATES" "THE CUSTOMER CREDIT RISK (1 IS
       LOW RISK," "5 IS HIGH RISK)";... .
```

The above PRINT statement specifies a 1-line footing which appears on the last line of the first form.

Q Take this path to describe the printing of a header for each form. The EACH FORM HEADER always begins at the top of each form; a NEW FORM print specification is implicit and need not be given. You cannot take this path if the FIRST FORM HEADER option or OTHER FORMS HEADER option was specified previously.

A header cannot continue onto a new form. That is, the <print specifications> for the header cannot cause form overflow, and it cannot contain a NEW LINE specification or a NEW FORM print specification which would continue the header onto another form.

The values of any data names, extensions, and so forth, which appear in the form header reflect the values contained in the logical record which caused a new form to be started (they reflect the values used in the last logical record in the previous form).

Example: ("INVENT")

```
PRINT:

   FOR EACH FORM HEAD 52 S, "LIST OF PART NUMBERS
      BY BIN", 3 BL;... .
```

The above PRINT statement specifies a report title centered on the top of each page.

R Take this path to specify a footnote for each form. The footnote appears on the very bottom lines of each form. You cannot take this path if the FIRST FORM FOOTNOTE option or OTHER FORMS FOOTNOTE option was specified previously.

A footnote always begins on a new line; a NEW LINE print specification is implicit and need not be given. <print specifications> for a footnote cannot contain any of the following:

1. NEW FORM specification.

2. NEW LINE specification which would continue the footnote onto another form.

3. <Data name> <string> specification.

4. <Conditional print specification>.

The footnote begins on LINE-NUMBER = FORM-LENGTH - (number of lines specified for the footnote) + 1, and continues to the last position on the form.

The values of any items which appear in the footnote are taken from the logical record being printed at the time a print specification or form overflow causes the form footnote to be printed.

Example: ("CLIENT")

```
PRINT:

   ... FOR EACH FORM FOOT "* - 'RISK RATING'
       INDICATES" "THE CUSTOMER CREDIT RISK (1 IS
       LOW RISK," "5 IS HIGH RISK)";... .
```

The above PRINT statement specifies a 1-line footing on the last line of each form.

Take this path to describe the printing of a header for each form other than the first. The OTHER FORMS header always begins at the top of a form; a NEW FORM print specification is implicit and must not be given. This path cannot be taken if the EACH FORM HEADER option was specified previously.

A header cannot continue onto a new form. That is, the <print specifications> for the header cannot cause form overflow, and it cannot contain a NEW FORM print specification or a NEW LINE specification which would continue the header onto another form.

The values of any data names, extensions, and so forth, which appear in the form header reflect the values contained in the logical record which caused a new form to be started (they reflect the values used in the last logical record in the previous form).

Example: ("INVENT")

```
PRINT: ...

   FOR OTHER FORMS HEADER 52 S, "LIST OF PART
      NUMBERS BY BIN", NL, 60 S, "(CONTINUED)",
      3 BL;... .
```

The above PRINT statement specifies a report title centered at the top of each page other than the first.

T Take this path to specify a footnote for each form other than the first. The footnote appears on the very bottom lines of the form. You cannot take this path if the EACH FORM FOOTNOTE option was taken previously.

A footnote always begins on a new line; a NEW LINE print specification is implicit and need not be given. <print specifications> for a footnote cannot contain any of the following:

1. NEW FORM specification.

2. NEW LINE specification which would continue the footnote LINE specification onto another form.

3. <Data name> <string> specification.

4. <Conditional print specification>.

The footnote begins on LINE-NUMBER = FORM-LENGTH - (number of lines specified for the footnote) + 1, and continues to the last position on the form.

The value of any items which appear in the footnote are taken from the logical record being printed at the time a print specification or form overflow causes the form footnote to be printed.

U Take this path to specify a header for each form onto which overflow is continued. A form overflows when a print specification other than an unconditional NEW FORM specification causes printing to overflow the form and continue at the top of the next form. The header resulting from the OVERFLOW FORM HEADER specification appears immediately after any header resulting from a FIRST FORM HEADER, EACH FORM HEADER, or OTHER FORMS HEADER specification at the top of the form onto which printing is continued.

A form header cannot continue onto a new form. That is, the <print specifications> for the form header cannot cause form overflow, and it cannot contain a NEW FORM specification or a NEW LINE specification which would continue the header onto another form.

The values of any items which appear in the header are obtained from the logical record which is being printed at the time a NEW FORM print specification or form overflow causes the header to be printed.

Example: ("CLIENT")

```
PRINT CONFIRM-STATEMENT AS FOLLOWS:

   FOR EACH CUSTOMER-NUMBER HEAD NEW FORM,...;
   FOR EACH RECORD NEW LINE,...;
   FOR OVERF FORM FOOT "ACCOUNT INVOICES
      CONTINUED" "ON ATTACHED STATEMENT";
   FOR OVERF FORM HEAD "INVOICE NUMBER", 2 S,
      "DUE DATE", 2 S, "TOTAL AMOUNT", 2 S,
      "AMOUNT PAID", 2 S, "LAST PAYMENT", 1 BL.
```

The above PRINT statement is identical to the PRINT statement given in the example for the OVERFLOW FORM FOOTNOTE option (path V) except for the specification of an overflow form header. The header supplies detail column headers which are printed on any attached statements (overflow form).

V Take this path to specify a footnote for each form which overflows. A form overflows when a print specification other than an unconditional NEW FORM specification causes printing to overflow the form and continue at the top of the next form. The footnote resulting from the OVERFLOW FORM FOOTNOTE specification appears on the very bottom lines of the form containing the overflow before any footnote resulting from a FIRST FORM FOOTNOTE, EACH FORM FOOTNOTE, or OTHER FORMS FOOTNOTE specification.

A footnote always begins on a new line; a NEW LINE print specification is implicit and need not be given. <print specifications> for a footnote cannot contain any of the following:

1. NEW FORM specification.
2. NEW LINE specification which would continue the footnote onto another form.
3. (Data name> <string> specification.
4. <Conditional print specification>.

The footnote begins on LINE-NUMBER = FORM-LENGTH - (number of lines specified for the footnote) - (numbers of lines specified for any footnote resulting from a FIRST FORM FOOTNOTE, EACH FORM FOOTNOTE, or OTHER FORMS FOOTNOTE specification) + 1. The number of lines required to print all appropriate footnotes are pre-allocated on each form.

The values of any items which appear in the footnote are taken from the logical record being printed at the time a print specification or form overflow causes the form footnote to be printed.

Example: ("CLIENT")

```
PRINT CONFIRM-STATEMENT AS FOLLOWS:

   FOR EACH CUSTOMER-NUMBER HEAD NEW FORM,...;
   FOR EACH RECORD NEW LINE,...;
   FOR OVERF FORM FOOT "ACCOUNT INVOICES
      CONTINUED" "ON ATTACHED STATEMENT".
```

The above PRINT statement specifies the printing of a positive confirmation letter which includes a listing of account balance details. CONFIRM-STATEMENT references a macro which provides the attributes and standard text of the form letter. Space is provided on the last half of the form letter for a detail listing. On occasion, details are continued onto a separate form. When this occurs, the specified OVERFLOW FORM FOOTNOTE provides a proper notice to the customer.

W <print specifications> describe, in a top-down, left-to-right, and abbreviated manner, what information is to be placed onto the form and how it should appear. A portion of the report is described which is printed just once (FIRST FORM HEADER) or is printed a number of times depending on the number of logical records reported and/or the control breaks specified (for example, <c-b name> HEADER or RECORD display).

Example: ("CLIENT")

```
PRINT MP-1 AS FOLLOWS:

FOR EACH RECORD NEW FORM, 2 BLANK LINES,
                   [CUSTOMER NAME],
                NEW LINE, [STREET-ADDRESS],
                NEW LINE, [CITY-STATE],
                NEW LINE, [ZIP-CODE].
```

The above <print specifications> describe how a portion of the report appears for each logical record reported. This portion is repeated once for each logical record reported; it corresponds to one mailing label form, and appears as shown in Figure 4-6. (MP-1 references a macro which sets START-POINT = 11, FORM-WIDTH = 30, FORM-LENGTH = 8, and FORM-TYPE = SPECIAL.)

```
JOHN C. DOE
107 S. MAIN ST
IRVINE, CA.
92714
```

Figure 4-6. Sample Mailing Label

X Take this path to give <print specifications> for additional segments of the report.

Y Take this path after all portions of the user-formatted report have been described.

Example: ("INVENT")

```
ACCEPT PART-NUMBER.  INPUT PARTMF USING PARTAM AT
   PART-NO = PART-NUMBER.
PRINT (FORM-WIDTH = 32, FORM-LENGTH = 8) :
   NF, BL, 8 S, "PART INFORMATION", 1 BL,
      "DESC: ", [PART-DESC], NL,
      "QUANTITY ON HAND:", [QUANTITY], NL,
      "BIN NUMBER: ", [BIN-NO].
```

The above PRINT statement specifies the desired layout for a TD730 screen. One screen displays information on a particular part as shown in Figure 4-7.

```
                    PART INFORMATION

          DESC:  1977 C LF BUMPER GRD
          QUANTITY ON HAND:  00021
          BIN NUMBER:  017
```

Figure 4-7. Sample TD 730 Output

Example: ("VOCAST")

PRINT AS FOLLOWS:

```
FOR EACH FORM HEADER 25 S, "-", [PAGE-NUMBER]
   IN 4, 3 S, "-", BL;
FOR EACH RECORD NL, [ASSET-DESC], [COST] AT
   40, [ASSET-LIFE] AT 59;
FOR EACH DEPT-NO HEAD NF,19 S, "DEPARTMENT
   NUMBER", [DEPT-NO], 2 BL, "ASSET DESC",
   ["COST"] AT 46, ["LIFE"] AT 57, BL;
FOR OVERF FORM HEADER "ASSET DESC", ["COST"]
   AT 46, ["LIFE"] AT 57, BL;
FOR FINAL SUMMARY NL, ["----------"] AT 36,
   NL, [TOTAL(COST)] IN 15 AT 35.
```

The above PRINT statement specifies a report consisting of five distinct portions (see Figure 4-8).

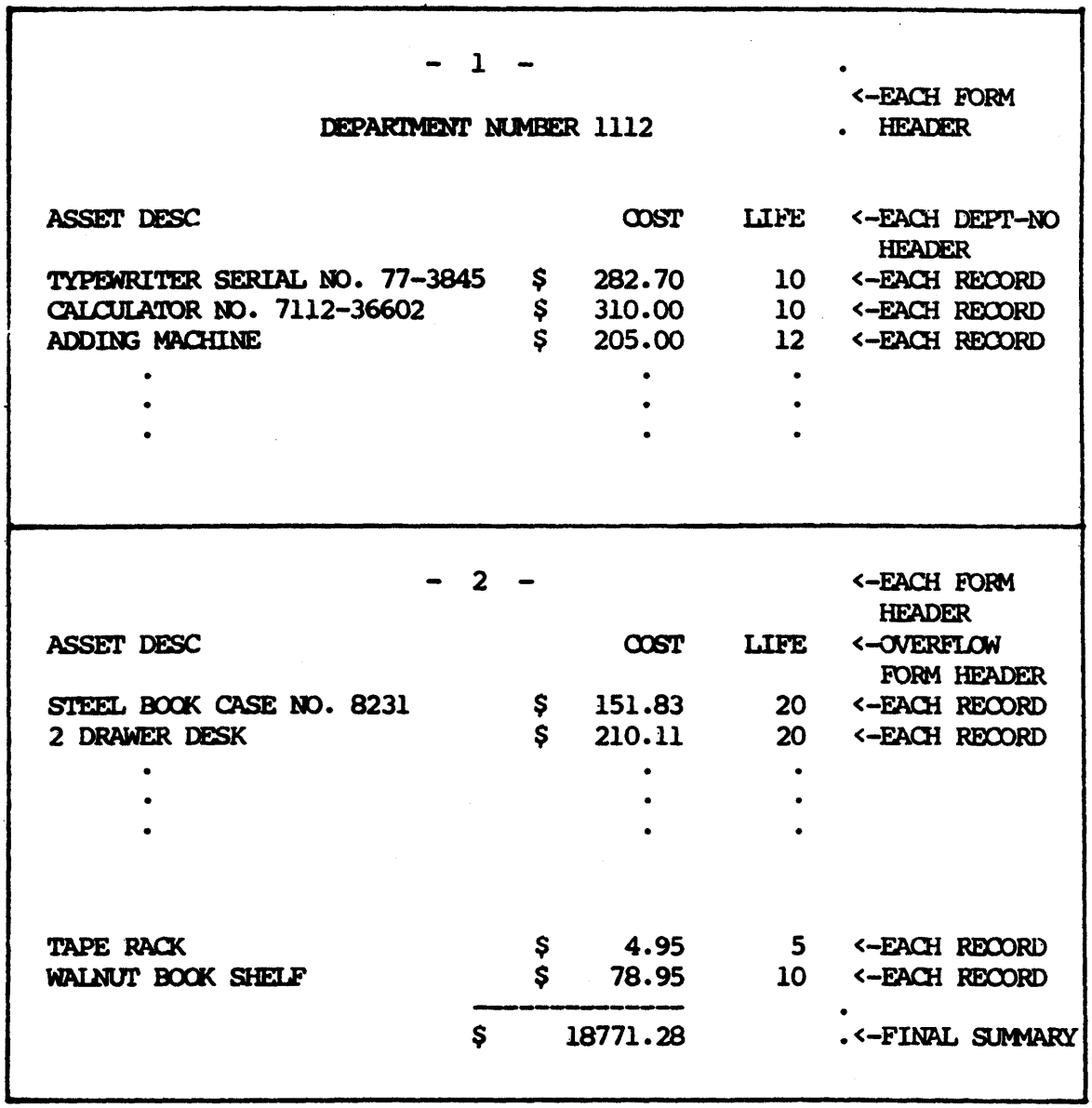


Figure 4-8. Portions of a User-Formatted Report

PROCESSING MODE

The Processing Mode clause is used to change the default processing mode of the generated report program. The default values are determined as follows:

1. Batch. The generated report program is to be run as an independent batch program. Batch mode is the default whenever the Report Language Analysis Program is run from the card reader or through the operator display terminal (ODT or SPO).

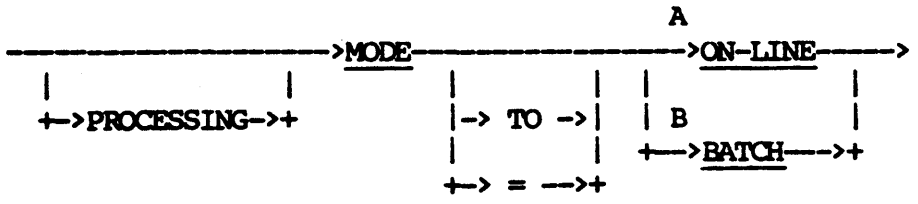
2. On-line. The generated report program is to be run under the control of On-Line REPORTER III. On-line mode is the default whenever the Report Language Analysis Program is run from a terminal via On-Line REPORTER III.

In either batch or on-line mode, the report listings or forms listings can be sent to printer, printer backup, or terminal backup. In on-line mode, the report listings or forms listings can also be sent to the terminal. The exceptions listing can be sent to printer, printer backup, or terminal backup in either mode.

NOTE

A terminal backup file is a disk file that is only accessible via On-Line REPORTER III. You can view it at a terminal or print it on a line printer using On-Line REPORTER III.

The syntax for the Processing Mode clause is as follows:



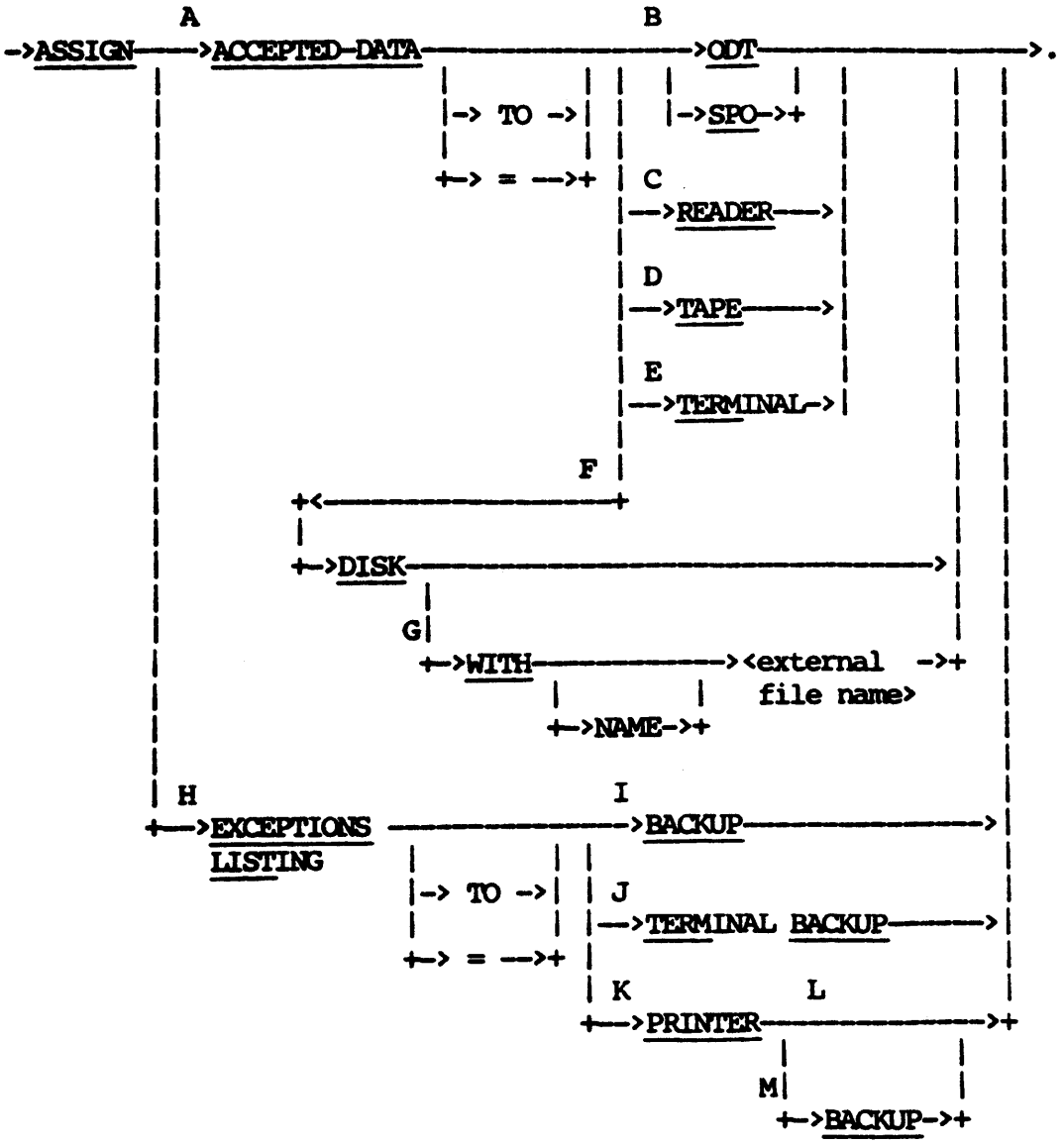
The paths of this syntax diagram are explained below:

Path	Explanation
A	Take this path to specify that the processing mode of the generated report program is on-line. The default hardware device for the report listings, the forms listings, and the exceptions listing is terminal backup. You can override these defaults with the appropriate Report-option Statement(s). Example: SET PROCESSING MODE TO ON-LINE. The execution of the generated report program is to be controlled by On-Line REPORTER III.
B	Take this path to specify that the processing mode of the generated report program is batch. The default hardware device for the report, forms, or exceptions listing is the printer. Example: SET MODE = BATCH. The generated report program is to be an independent batch program.

PROCESS-OPTION ASSIGN STATEMENT

You can use the Process-option ASSIGN statement for either of the following purposes: to specify explicitly the source for input of accepted data to a particular generated report program; to specify explicitly the output device/means to be used for the exceptions listing produced by a particular generated report program.

The syntax for the Process-option ASSIGN statement is as follows:



The paths of this syntax diagram are explained below:

Path	Explanation

A The ASSIGN ACCEPTED-DATA statement specifies the source for input of accepted data to the generated report program. All accepted data is free-form with each field separated by one or more spaces. Data described to be a string must be enclosed in quotes ("). This statement is valid only when used with an ACCEPT statement. The default device for assigning accepted data is READER if PROCESSING MODE is BATCH, and TERMINAL if PROCESSING MODE is ON-LINE. (Refer to Process-option SET statement in this section.)

Example:

```
ASSIGN ACCEPTED-DATA TO ODT.
```

The accepted data for the required report is to be entered through the operator display terminal.

B Take this path to specify that the operator display terminal (ODT) be the hardware device for accepted data. This device is also known as the console printer (SPO). You can use either ODT or SPO in this specification. All accepted data is input via the operator display terminal keyboard. To specify the end-of-data condition, enter <mix#>AX END.

C Take this path to specify that the accepted data be input via the card reader. The first card must have an invalid character in column 1, followed by the words DATA RP3DAT. The last card must have an invalid character in column 1, followed by the word END.

Example:

```
ASSIGN ACCEPTED-DATA = READER.
```

D Take this path the specify that the accepted data be input via a tape drive.

```
ASSIGN ACCEPTED-DATA TO TAPE.
```

E Take this path to specify that the accepted data be entered from a terminal. Use of this path is valid only if the processing mode is ON-LINE.

F Take this path when the accepted data is to be input from a disk or disk pack file. This file must be an 80-character DATA type file. Other file types, such as CANDE sequence or COBOL, would input the sequence numbers as part of the accepted data and thereby would result in errors.

G Take this path to specify an <external file name> for the accepted data file assigned to disk. If this path is not taken, the default file name will be RP3DAT on disk.

Example:

```
ASSIGN ACCEPTED-DATA DISK WITH NAME "PGMAXD"
   PACK "USER3".
```

H The ASSIGN EXCEPTIONS LISTING statement causes the exceptions listing produced by the generated report program to be assigned to the specified output device/means.

Example:

```
ASSIGN EXCEPTIONS LISTING TO BACKUP.
```

I Take this path to assign the exceptions listing to backup. The type of backup is dependent on the processing mode. If the processing mode is BATCH, BACKUP means printer backup. If the processing mode is ON-LINE, BACKUP means terminal backup.

J Take this path to assign the exceptions listing specifically to terminal backup. When processing is in the ON-LINE mode, terminal backup is the default.

Example:

```
ASSIGN EXCEPTIONS LIST TO TERM BACKUP.
```

K Take this path if you desire to specify that the exceptions listing be assigned to the printer or to printer backup.

L Take this path to specify that the exceptions listing be assigned to the printer. It is a function of the system software to determine whether the listing is sent to the printer or to printer backup. When processing in the BATCH mode, printer is the default.

Example:

```
ASSIGN EXCEPTIONS LIST PRINTER.
```

M Take this path to specify that the exceptions listing be assigned to printer backup.

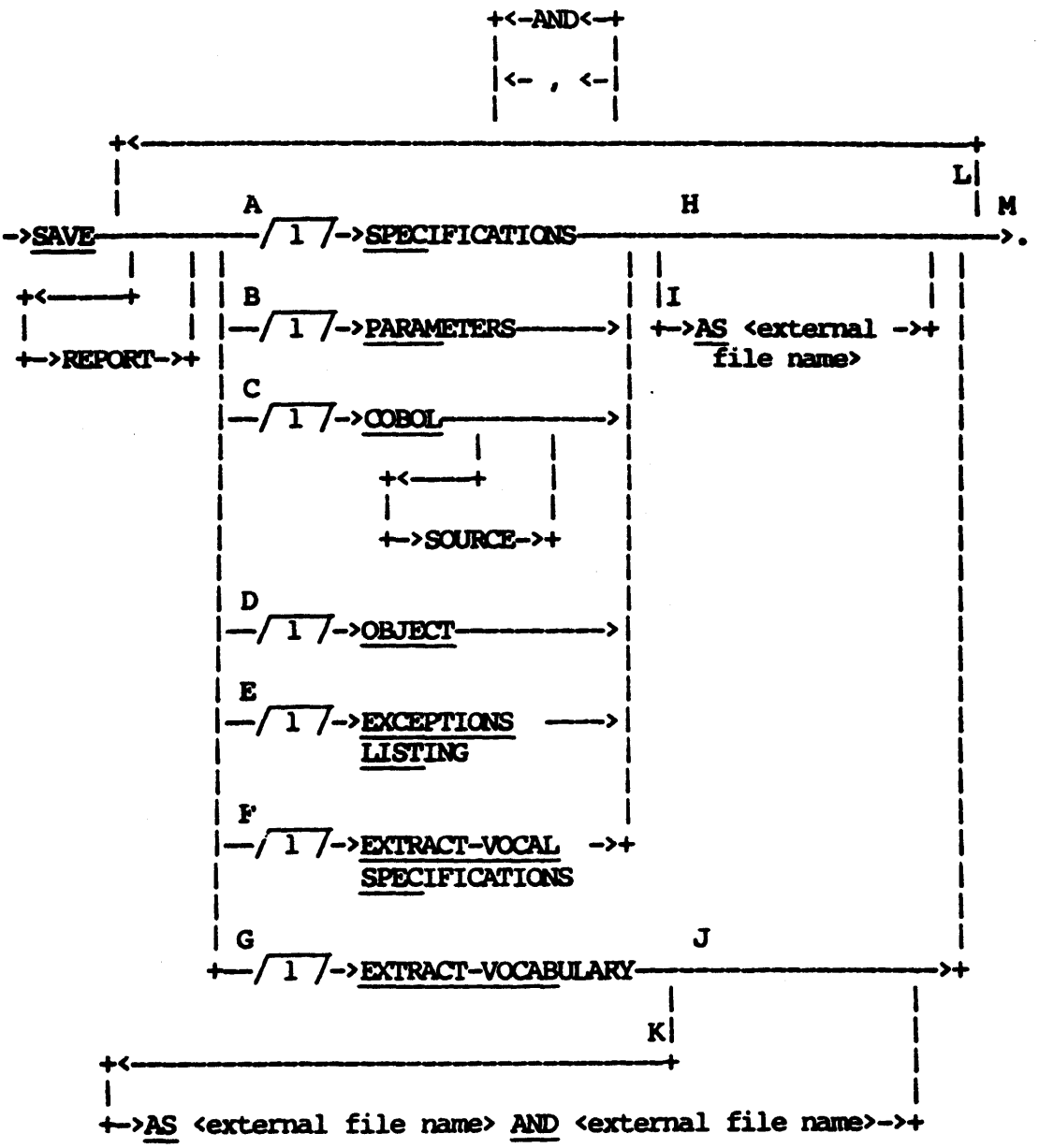
Example:

```
ASSIGN EXCEPTIONS LISTING = PRINTER BACKUP.
```

PROCESS-OPTION SAVE STATEMENT

You can use the Process-option SAVE statement to cause certain files to be saved and stored on disk. You optionally can specify <external file name>s for these files.

The syntax for the Process-option SAVE statement is as follows:



The paths of this syntax diagram are explained below.

Path	Explanation

A Take this path to specify that all subsequent statements in the specification be saved on disk. All statements following this statement in the specification are (re)sequenced and saved. This file can be referenced in subsequent report specifications with a COMBINE statement. If this path is not taken, the specification disk file will not be created.

Example:

```
SAVE SPECIFICATIONS.
```

B Take this path to specify that the parameters produced by the Report Language Analysis Program be saved on disk. Normally this parameter file is removed after being processed by the Report Program Generator. By saving the parameter file and suppressing GENERATION, the Report Program Generator can be run independently of the Report Language Analysis Program. (Refer to the appropriate discussion in Section 6 for details.)

C Take this path to specify that the generated COBOL source report program be saved on disk in a state ready for subsequent use. If this option is not specified, the COBOL source file remains on disk after compilation, but it is not resequenced.

Example:

```
SAVE COBOL.
```

D Take this path to specify that the report program created by compiling the generated COBOL source program be saved on disk. If this option is not specified, the compiled program is removed by default.

Example:

```
SAVE OBJECT.
```

E Take this path in conjunction with path I to assign an external file name to the exceptions listing produced by the generated report program. Exceptions are run-time errors which can occur during the input, processing, and output of data for a report. The exceptions listing will be produced when at least one exception error occurs.

Assigning an external file name to the exceptions listing will have no effect on the operation of the report program while the internal file is assigned to the printer. Reassigning the exceptions listing to another hardware device, such as disk, through a system file equate statement gives more meaning to this option.

This option is primarily used when the processing mode is set to on-line. Using this path in conjunction with the process-option ASSIGN statement (ASSIGN EXCEPTIONS LISTING TO TERMINAL BACKUP) allows the user to assign a meaningful file name to the exceptions listing accessible through the On-Line REPORTER III product.

Using this path without taking path I is meaningless.

Example:

SAVE EXCEPTIONS LISTING AS "EXCP01".

The exceptions listing file is assigned an external file name of EXCP01. If the exceptions listing had been assigned to TERMINAL BACKUP, the file EXCP01 would be accessible through the On-Line REPORTER III product.

F Take this path to specify that the vocabulary specifications generated for files extracted "WITH VOCABULARY" (refer to EXTRACT Statement in this section) be saved on disk. If this option is not specified, the vocabulary specifications are removed by default.

Example:

SAVE EXTRACT-VOCAL SPECIFICATIONS.

G Take this path to specify that the vocabulary files created by the RP3VOC program based on the EXTRACT WITH VOCABULARY statements be saved on disk. Since these files are saved by default, this path is used primarily in conjunction with path K to name the files.

Example:

SAVE EXTRACT-VOCABULARY AS "NEWVOC" AND "NEWVC2".

H Take this path to specify that for paths A thru F the default <external file name> is satisfactory (refer to Appendix B: DEFAULTS AND LIMITS).

I Take this path to specify explicitly the <external file name> for the file to be saved. B 1000 system users must not use a period as part of the <external file name> of a SAVE OBJECT AS <external file name> statement.

Examples:

```
SAVE SPEC AS "RP3SPC".
```

The remaining specifications in the deck are resequenced and saved in the disk file AUDSPC.

```
SAVE OBJECT AS "RP3OBJ".
```

The generated report program is saved as a disk file named RP3OBJ.

J Take this path if you desire default external file names for the extract vocabulary files selected in path E (refer to Appendix B: LIMITS AND DEFAULTS).

K Take this path to specify two nondefault <external file name>s to identify the extract vocabulary files.

Example:

```
SAVE EXTRACT-VOCAB AS "RP3VC1" AND "RP3VC2".
```

The extracted vocabulary files are saved as "RP3VC1" and "RP3VC2". The name of the vocabulary is "RP3VC1".

L Take this path to specify additional options in the same statement.

Examples:

```
SAVE SPEC AS "RP3SPC" OBJECT AS "RP3OBJ".
SAVE COBOL AS "RP3COB", OBJECT AS "RP3OBJ", SPEC
   AS "RP3SPC" AND EXTRACT-VOCAL AS "RP3VOC".
```

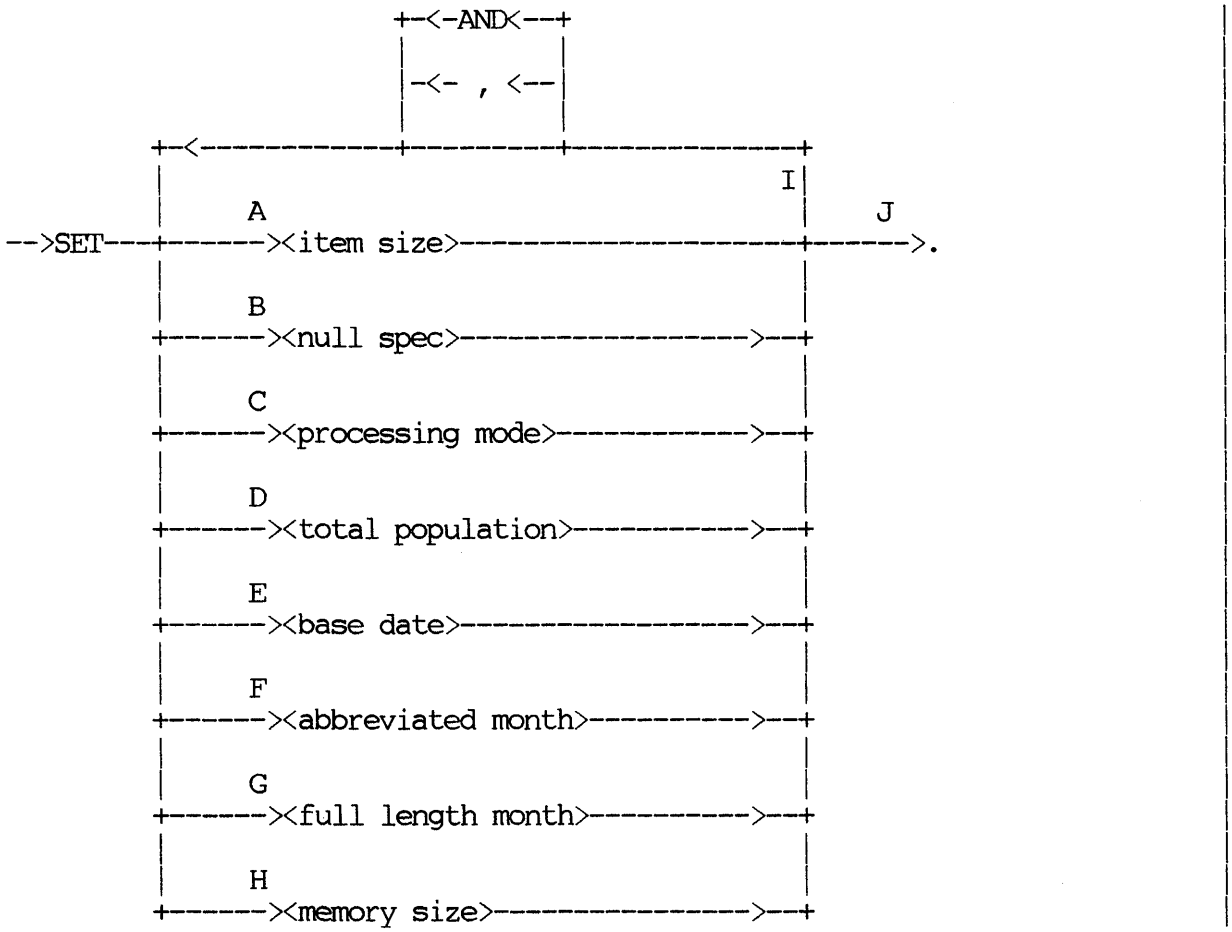
The second example above assumes the presence of an EXTRACT statement in the report specification.

M Take this path after all desired options in the same statement are specified.

PROCESS-OPTION SET STATEMENT

The Process-option SET statement enables you to assign nondefault values to certain processing attributes. If the default values are satisfactory, this statement is not needed.

The syntax for the Process-option SET statement is as follows:



The paths of this syntax diagram are explained below:

Path	Explanation
A	Take this path to change the default size of intermediate values resulting from the calculation of expressions and statistics within the generated report program. The size of the intermediate result values affects the accuracy of intermediate,

and therefore final results; it also provides a default size for derived and accepted data items.

If the size specified for intermediate values is not large enough for the integer portion of the result of a calculation, an exception is noted by the generated report program; if it is not large enough for the fractional portion, truncation of least significant digits occurs.

B Take this path to change the default values of null items for purposes of reporting or processing. Null-valued items result from certain retrievals as specified in the INPUT statement, or from exceptional data processing (for example, dividing by zero causes an exception to be noted and a null-valued result).

C Take this path to change the default processing mode of the generated report program. This path must not be taken unless On-Line REPORTER III is being used.

D Take this path to provide an estimate of the maximum number of logical records which are initially selected and/or sampled for reporting. This estimate is useful in determining an optimal field size for the printing of certain statistical values.

E Take this path to change the default base date.

F Take this path to change the default month abbreviations from English to another language. The month abbreviations are used in the TITLE statement, the PRINT insert clause, and the date format DDMMMYY-DATE.

G Take this path to change the default month names from English to another language. The month names are used in the TITLE statement and the PRINT insert clause.

H Take this path to increase the memory size for the generated COBOL program object code. This clause is necessary only for very large report specifications or where a large amount of information is being sorted.

I Take this path to specify any additional process SET options in the same statement. The same option should not be specified more than once in the same report specification; however, if this occurs, the last specification is used by REPORTER III.

J Take this path after all desired process SET options in the same statement are specified.

PROCESS-OPTION STATEMENT

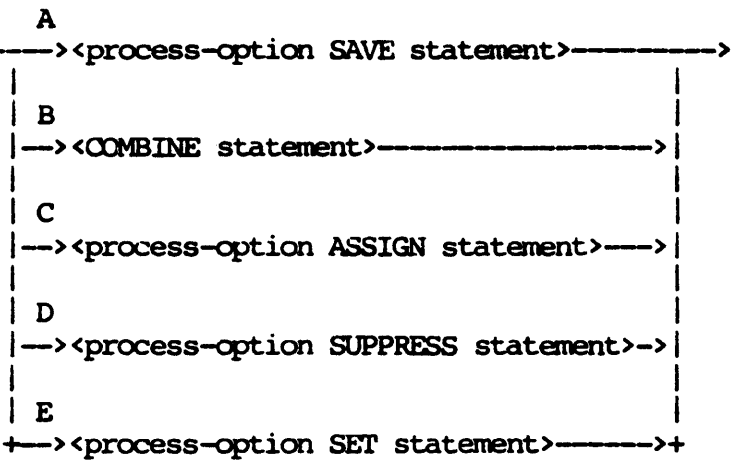
The process options enable you to vary the default operation of the REPORTER III System. The basic default operation of REPORTER III on the A Series Systems, the B 2000/B 3000/B 4000 System, and the B 1000 System, respectively, is described under the appropriate AUTOMATIC PROCEDURE heading in Section 6.

Example:

```
SUPPRESS COMPILE.
```

This specification halts the automatic compilation of the generated program.

The syntax for the Process-option statement is as follows:



The paths of this syntax diagram are explained below:

Path	Explanation
A	Take this path to name explicitly and/or save certain work files, intermediate result files, and output files.

Example:

```
SAVE OBJECT AS "MYPROG".
```

This statement specifies that the compiled, generated program is to be saved on disk as the file "MYPROG".

B Take this path to specify the merging of subsequent specifications with existing report specifications.

Example:

```
COMBINE WITH "ARSPEC".
```

In this case, the specification input is taken as patches to be merged with the report specifications in the disk file "ARSPEC".

C Take this path to specify that accepted data be taken from a different device than the card reader, or that the exception listing is to go to a particular hardware device.

Example:

```
ASSIGN ACCEPTED-DATA TO SPO.
```

In this case, the accepted data is taken from the console printer instead of the card reader.

D Take this path to alter the automatic initiation of the various processes of the REPORTER III System, or to suppress various listings.

Example:

```
SUPPRESS GENERATION.
```

In this case, the specifications are scanned and syntax-checked, but no program is generated.

E Take this path to change the default values of various processing parameters.

Example:

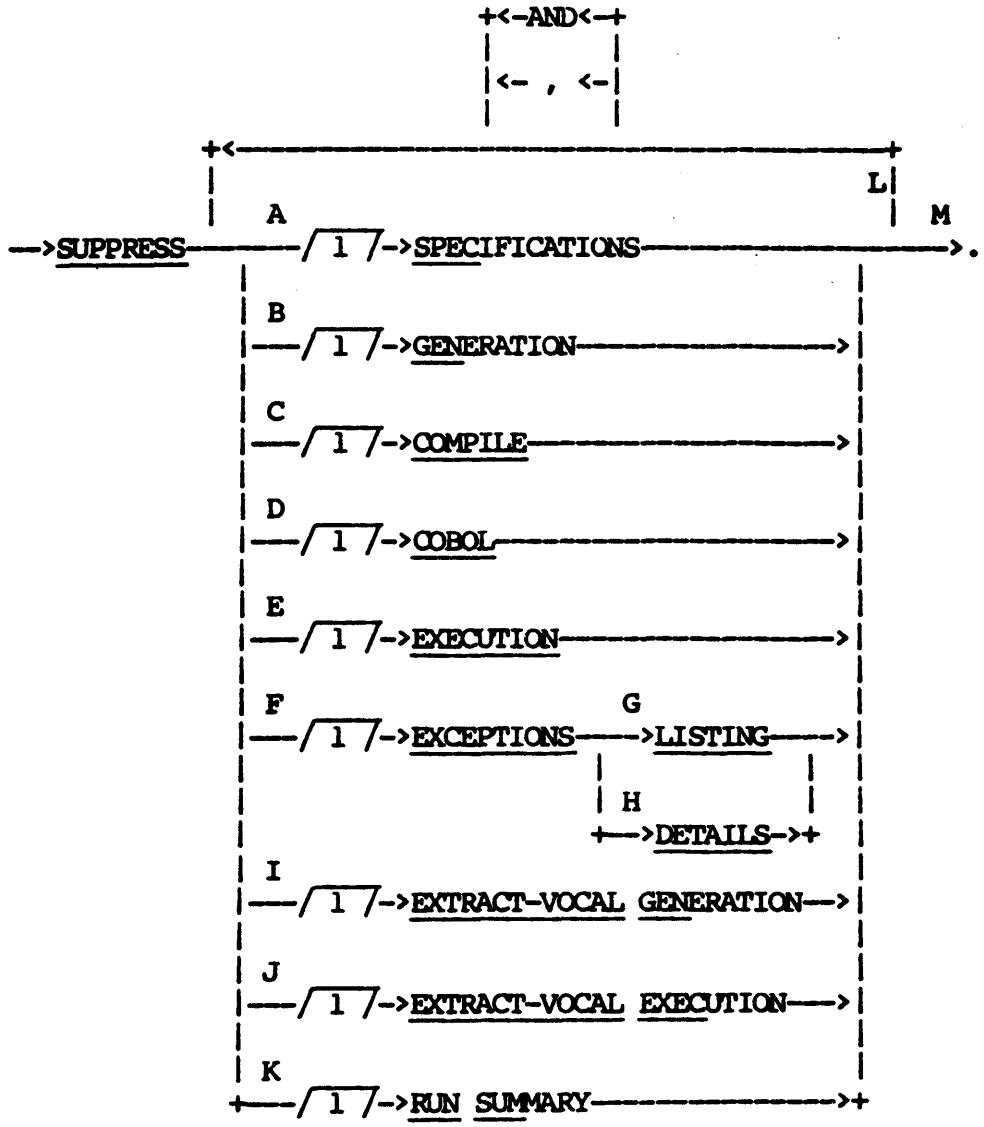
```
SET NULL-STRING TO ALL "*".
```

In this case, null string-data items are filled with asterisks instead of the default spaces (blanks).

PROCESS-OPTION SUPPRESS STATEMENT

The Process-option SUPPRESS statement enables selective suppression of unwanted actions during and after the Report Language Analysis Program execution. None of the actions are suppressed by default.

The syntax for the Process-option SUPPRESS statement is as follows:



The paths of this syntax diagram are explained below.

Path	Explanation
A	Take this path to specify the suppression of SPECIFICATIONS (the listing of report statements and diagnostics). Example: SUPPRESS SPEC.
B	Take this path to specify the suppression of GENERATION (the report program generation). This option is useful for a syntax-only check of the report specification. Subsequent action, if desired, is done manually. Refer to the appropriate portion of Section 6 for information on executing the Report Program Generator. Example: SUPPRESS GENERATION.
C	Take this path to specify the suppression of COMPILE (the compilation of the generated report program). Refer to the appropriate portion of Section 6 for information on manual compilation of the generated report. Example: SUPPRESS COMPILE.
D	Take this path to specify the suppression of COBOL (the listing of the generated program produced during the compile). Example: SUPPRESS COBOL.
E	Take this path to specify the suppression of EXECUTION (the suppression of execution of the compiled generated report program). The program is not saved by default. If the generated report program is to be executed at a future time, you must also specify a SAVE OBJECT statement (refer to Process-option SAVE statement in this section). Example: SUPPRESS EXECUTION.

F Take this path if all or part of the information in the exceptions listing is to be suppressed. An exceptions listing contains detailed information related to individual exceptions. It also contains a count of the total number of exceptions encountered.

G Take this path if the entire exceptions listing is to be suppressed. Exceptions checking is done in the generated report program, but no exceptions report is produced. A count of the number of exceptions is given in the run summary report if the run summary is not suppressed. If this path is taken, an ASSIGN EXCEPTIONS LISTING or SAVE EXCEPTIONS LISTING specification is ignored.

Example:

```
SUPPRESS EXCEPTIONS LISTING.
```

H Take this path if the detail information in the exceptions listing is to be suppressed. Only a count of the number of exceptions is reported.

Example:

```
SUPPRESS EXCEPTIONS DETAILS.
```

I Take this path to specify the suppression of EXTRACT-VOCAL GENERATION (the creation of new vocabulary specifications for extract files). This path is used only in conjunction with the EXTRACT WITH VOCABULARY specification.

Example:

```
SUPPRESS EXTRACT-VOCAL GEN.
```

J Take this path to specify the suppression of EXTRACT-VOCAL EXECUTION (the automatically initiated execution of the RP3VOC Program). This path is used only in conjunction with the EXTRACT WITH VOCABULARY specification. It allows the RP3VOC specification deck for the extract vocabulary to be built on disk but prevents the execution of this deck. These specifications can be executed subsequently with the proper file equation.

Example:

```
SUPPRESS EXTRACT-VOCAL EXEC.
```

K Take this path to specify the suppression of all run-summary reports. If this path is not taken, a run summary is reported as usual for each report section at the end of the REPORT LISTING RL<nnrr>. A run summary includes the number of records processed by the input section and report section, and the number of exceptions detected.

Example:

```
SUPPRESS RUN SUMMARY.
```

NOTE

If both exceptions listing and run summary are suppressed, the user is not notified of any exceptions that might invalidate the reports.

L Take this path to specify additional suppress options in the same statement.

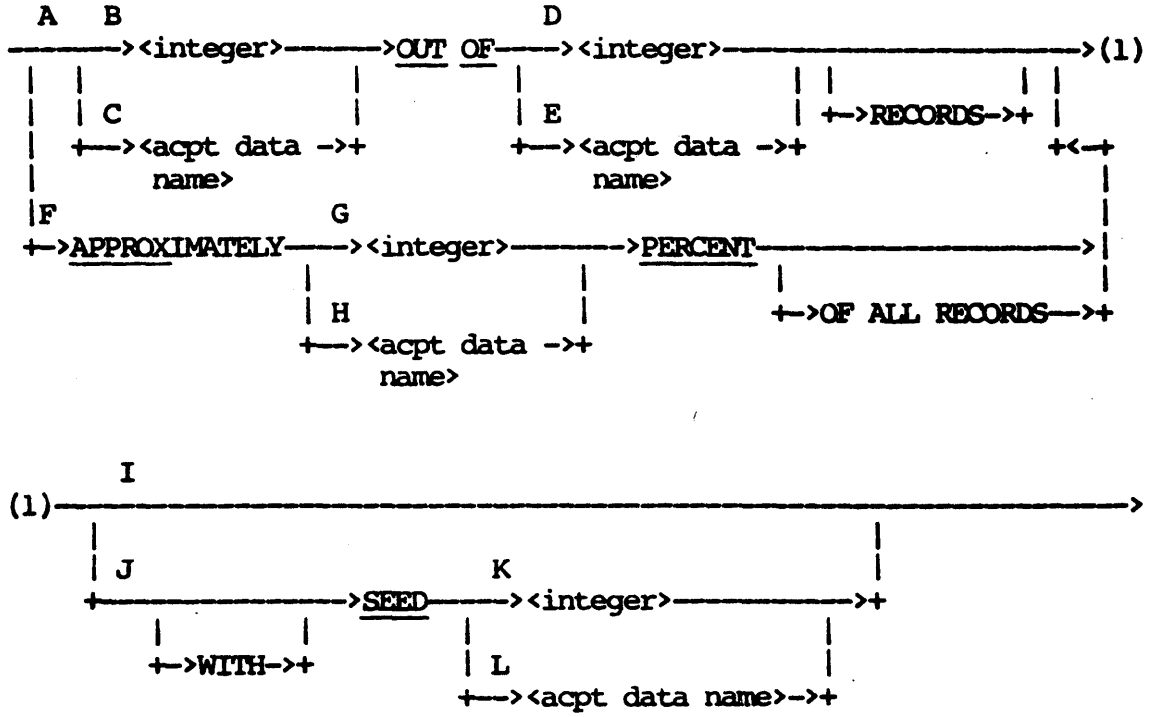
```
SUPPRESS SPECIFICATIONS EXECUTION.
```

M Take this path after all desired suppress options in the same statement have been specified.

RANDOM-SAMPLE DESC

The Random-sample Desc is used to describe a random sample.

The syntax for the Random-sample Desc is as follows:



The paths of this syntax diagram are explained below:

Path	Explanation
A	Take this path to specify the exact sample size or number of records to be sampled randomly out of a known strata population.

Example:

```
SAMPLE RANDOMLY 100 OUT OF 5107 RECORDS.
```

This sample consists of 100 records taken randomly out of a predetermined population of 5107 records.

B-C The sample size can be specified by an <integer> (path B) or an accepted data item (path C). It must be a positive integer less than or equal to the strata population.

D-E The strata population can be specified by an <integer> (path D) or an accepted data item (path E). It must be a positive integer. It must be predetermined and supplied as accurately as possible to assure a resulting sample which is truly random. If unknown, the strata population can be determined by appropriate report specifications.

Example:

```
ACCEPT SAMPLE-SIZE NUM(5).
    .
    .
    .
SAMPLE CREDIT-ACCTS RANDOMLY SAMPLE-SIZE OUT OF
       10564 RECORDS.
```

F Take this path to specify an approximate sample size as a percentage of all records in the strata. The percent specified actually represents the probability of any one record in the strata being selected for the sample.

Example:

```
SAMPLE RANDOMLY APPROXIMATELY 5 PERCENT OF ALL
   RECORDS.
```

The sample consists of approximately 5 percent of all records in the strata. Each record has a 5 percent chance of being selected.

G-H The approximate percentage of records to be sampled can be specified as an <integer> (path G) or an accepted data item (path H). It must be a positive integer.

I Take this path if a seed is not to be specified for the REPORTER III random number generator. A default seed is used which is randomly generated by the report program for each run (and for each set of accepted data when supplied). This seed is printed in the run summary.

Example:

```
SAMPLE RANDOMLY APPROX 2 PERCENT.
```

A default seed is generated for each run of the generated report program. Therefore, a different sample is produced for each run.

J Take this path to specify a seed. The report program then produces the same sample each time it is run.

Example:

```
SAMPLE RANDOMLY 100 OUT OF 827 WITH SEED 123.
```

The specified seed of 123 is always used by the generated report program.

K-L The seed can be specified by an <integer> (path K) or an accepted data item (path L). It must be a positive integer.

RANGE-BREAK-ITEM DESC

The Range-break-item Desc is used to categorize information. You use it by stipulating value ranges (range breaks or control breaks) for a particular data item, referred to as a ranged item. You then can obtain the item values of logical records in the stipulated ranges.

The Range-break-item Desc defines a range-break item. The range-break-item is a string that denotes the lower and upper range limits of a ranged data item (for example, where age is the ranged item, age values are organized into range-break groups: 20-29, 30-39, and so forth) Once defined, the range-break item is referenced by <c-b name> in other report language statements.

Ranges must be specified in increasing sequence value, and they need not be contiguous or of equal intervals. Specified range groups always include the lower limit and exclude the upper limit value.

Example:

```
1000 TO 6000
6000 TO 10000
```

The first group contains all numbers from 1000 up to but not including 6000. The value 6000 as the upper limit is excluded from the first group, but is included in the second group as the lower limit. Thus the second group includes numbers from 6000 up to but not including the upper limit 10000.

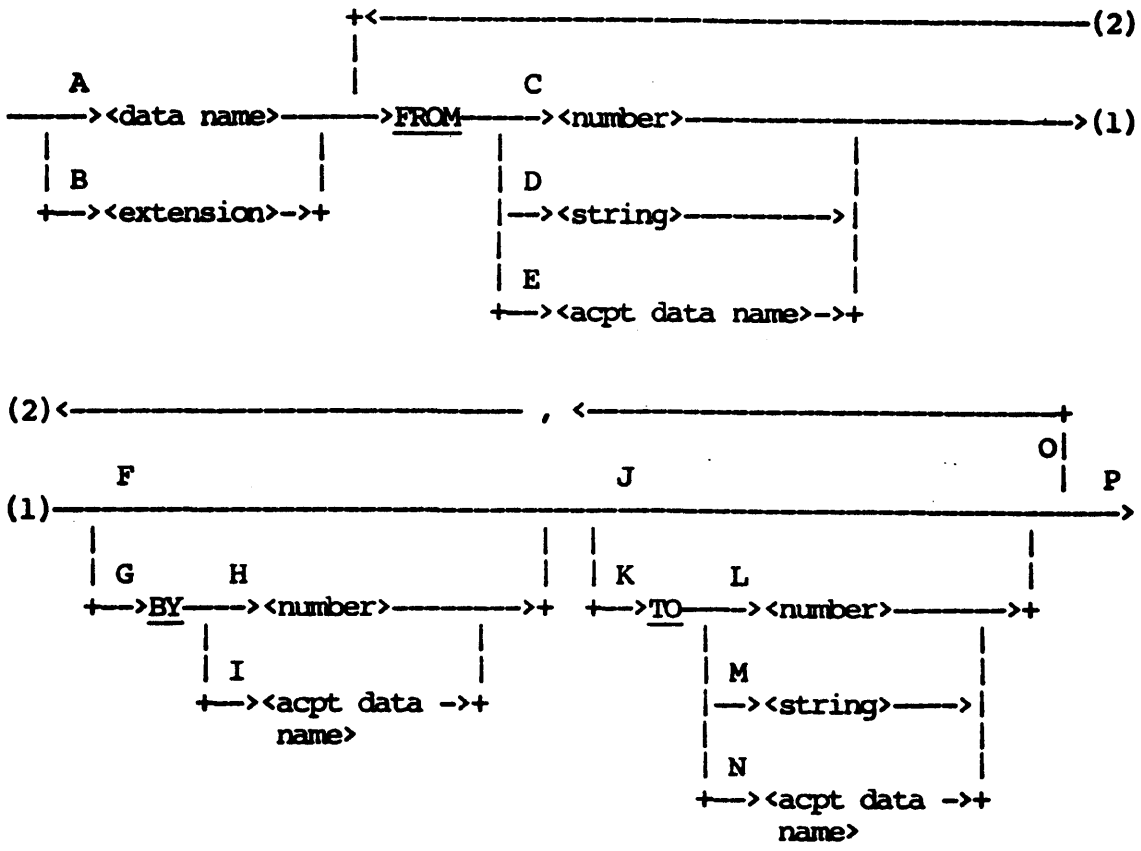
You specify the range limits in numbers, strings, or accepted data names, corresponding to the type of ranged item. The ranges appear on reports in edited format, separated by a hyphen, and preceded and followed by a blank.

Example: ("CLIENT")

```
RANGE BY CREDIT-LIMIT FROM 1000 BY 1000 TO 10000
```

The values of CREDIT-LIMIT are used to organize the logical records (accounts) into nine range-break groups (accounts having limits of 1000 to 2000, 2000 to 3000, 3000 to 4000, ... 9000 to 10000). These groups can be used to gather and report statistical summaries.

The syntax for Range-break-item Desc is as follows:



The paths of this syntax diagram are explained below:

Path	Explanation
A	Take this path to specify a <data name> item to be ranged. The <data name> must reference a nonstatistical data item. Specified values of the item are organized into range- or control-break groupings. Unless suppressed, information is ordered in ascending sequence based on the ranged item. This ordering is subordinate to any higher level control-break ordering. Example: ("CLIENT") RANGE BY CREDIT-LIMIT FROM 1000 BY 1000 TO 10000.

CREDIT-LIMIT refers to a data item whose values are organized into nine range-break groupings, each of which represents a $1000 span in CREDIT-LIMIT.

B Take this path when range- or control-break groupings are based on the value ranges of a derived data item (extension). The extension that you define here must be nonstatistical.

Unless suppressed, information is ordered into ascending sequence based on the ranged item. This ordering is subordinate to any higher-level control-break ordering.

Example: ("CLIENT")

```
RANGE BY CREDIT-CODE IS CREDIT-LIMIT/1000
   FROM 1 BY 5 TO 99.
```

The values of CREDIT-CODE, which are CREDIT-LIMIT/1000, are used to organize the accounts being reported into 20 range-break groups.

C Take this path to specify a numeric lower limit for a range-break grouping. Numeric limits can be used only if the ranged item is a numeric data item.

D Take this path to assign a string lower limit for a range-break grouping. String limits can be used only with a string data item.

Example: ("CLIENT")

```
RANGE BY CUSTOMER-NAME FROM "A" TO "F",
   FROM "F" TO "J", FROM "J" TO "N",
      FROM "N" TO "R", FROM "R" TO "V",
      FROM "V" TO "Z".
```

The first range group consists of all customer names from the lowest value "A...", through the highest value "E..." (i.e, to the value "F" but not including it). "F" is the lower limit included in next group, "F" TO "J".

E Take this path to reference an accepted data item as a lower limit. It must be the same type, string or numeric, as the ranged item.

F Take this path to specify string limits (when you have taken path D), or if all numeric limits are not equal spans or are not contiguous.

G Use this path as a shorthand way to set numeric range limits. The value of the integer in the BY option is added repeatedly to the lower-limit value to form successive ranges. Since range groups include the lower limit value and exclude the upper limit value, one number can be the upper limit of one group and the lower limit of another.

Example: ("CLIENT")

```
GROUP BY RANGES OF CREDIT-LIMIT FROM 1000 BY 5000
   TO 21000 AS CREDIT-LIMITS.
```

The statement generates the following four range-break groups:

1. 1000 to 6000
2. 6000 to 11000
3. 11000 to 16000
4. 16000 to 21000

J The highest value for the upper limit is set when you take this path. The grouping includes all values of the ranged item from the lower limit up to and including the maximum value that can fit into the storage picture (internal size) for the ranged item.

Example: ("CLIENT")

```
RANGE BY CREDIT-LIMIT FROM 0 BY 1000
```

This instruction provides 100 groupings, since the internal size for CREDIT-LIMIT is five digits (0-1000, 1000-2000, ... 98000-99000, 99000-99999+). The last grouping contains the value 99999.

K Take this path to specify an upper limit for a range grouping.

L Take this path to specify a numeric upper limit when the ranged item is numeric.

Example: ("CLIENT")

```
RANGE BY CREDIT-LIMIT FROM 1000 TO 3000,
   FROM 3000 TO 5000
```

This specifies range groups based on CREDIT-LIMIT. Accounts having credit limits of 3000 are not included in the first grouping, but are included in the second.

M Take this path to designate a string upper limit for a string ranged item.

Example: ("CLIENT")

```
RANGE BY CUSTOMER-NAME FROM "A" TO "F",
   FROM "F" TO "J", FROM "J TO "N",
      FROM "N" TO "R", FROM "R" TO "V",
      FROM "V" TO "Z".
```

All accounts for customer names beginning with A through E are included in the first range group; F through I are included in the second, and so forth.

N Take this path to reference an accepted data item as the upper limit. The type of the item, string or numeric, must match the ranged item.

O Take this path as many times as necessary to specify all range limits; any number of limits can be used. Range limits must be specified in increasing value. You can use the BY clause as a shorthand method of denoting two or more contiguous ranges. The range limits you assign in each FROM-BY-TO option need not be contiguous; the ranges between the ones you specify are implied, and all ranges will be identified in the report.

Example: ("CLIENT")

```
RANGE BY CUSTOMER-NAME FROM "A" TO "F",
                       FROM "F" TO "K",
                       FROM "K" TO "P",
                       FROM "P" TO "U",
                       FROM "U".
```

Accounts above are organized into five contiguous range groups based on the customer name limits supplied. Four categories are specified, and one is implied ("U" to highest value).

Example: ("CLIENT")

```
RANGE BY CREDIT-LIMIT FROM 1000 TO 3000,
   FROM 3000 TO 5000, FROM 7000 TO 10000,
```

The underlined portion actually describes six numeric range groups based on credit limit. Since the ranges are not contiguous, the range limits before, between, and after the specified limits are implied (0 TO 1000, 5000 TO 7000, 10000 TO highest value). Values of CREDIT-LIMIT within these implied range limits also will appear on the printed report.

Example: ("CLIENT")

```
RANGE BY BALANCE-DUE FROM 0 TO 5000.00,
   FROM 5000.00 BY 1000.00 TO 50000.00,
      FROM 50000.00.
```

Accounts in the above example are ranged based on balance-due into 48 categories: 47 are specified, and one is implied (50000.00 to highest value).

P Take this path when all range limits have been specified.

RANGE STATEMENT

The RANGE statement is used to define a range-break item as the only control-break item. Ranges for the particular data item are specified so that information can be grouped as required for reports and summaries. The defined range-break item is referenced in other statements by a <c-b name> or RANGE.

Example:

```
RANGE BY AGE FROM 20 BY 5 TO 80.
    .
    .
    .
REPORT FOR EACH RANGE:  COUNT, TOTAL(SALARY).
```

Information about each age range is grouped together. Statistics are calculated and printed for each age range. (Refer to the REPORT statement in this section.)

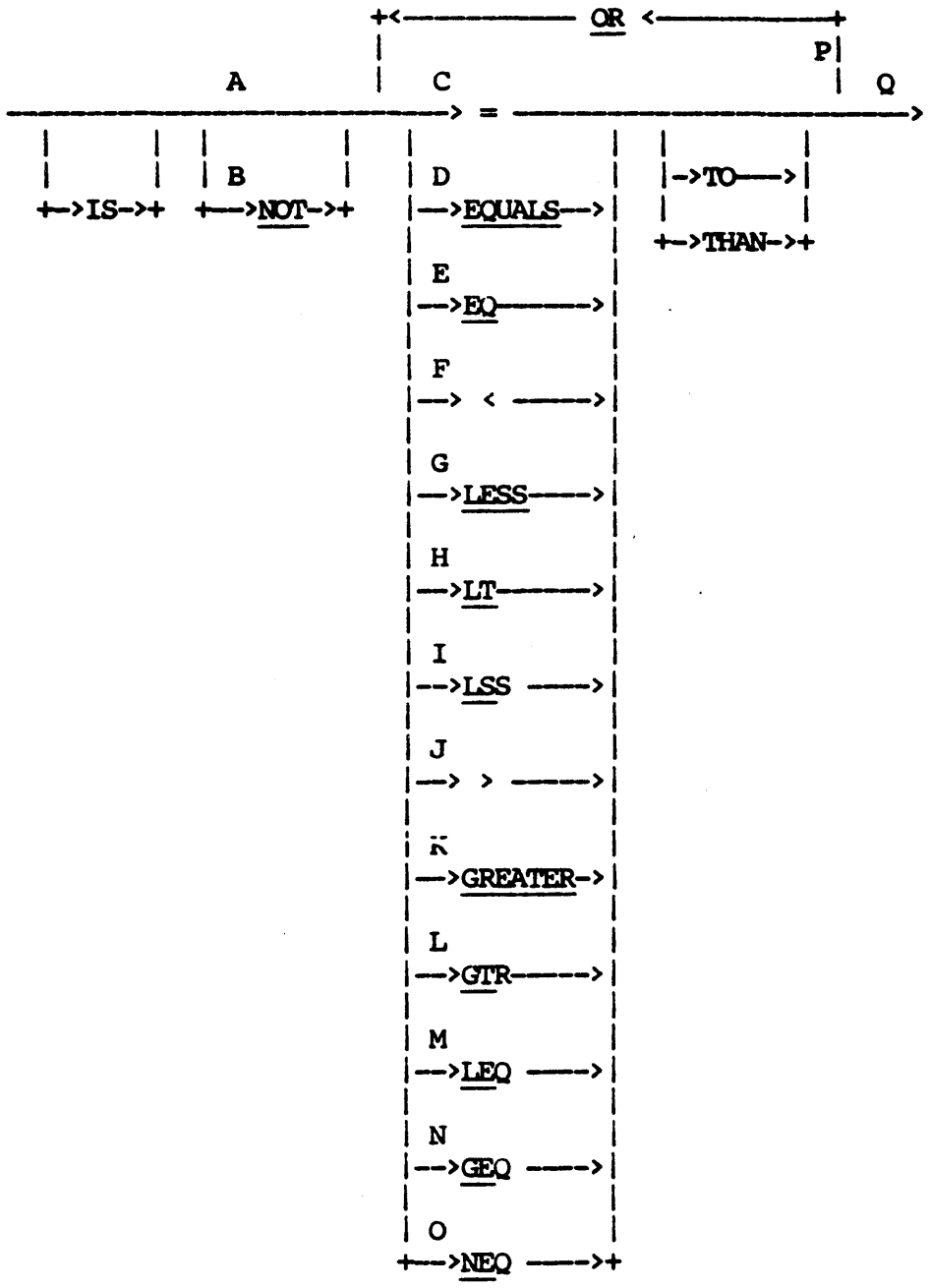
The syntax for the RANGE statement is as follows:

```
-->RANGE BY <range-break-item desc>.
```

RELATIONAL OPERATOR

A Relational Operator specifies the criteria for comparison.

The syntax for Relational Operator is as follows:



NOTE

When two string values of unequal length are compared, the smaller is extended with spaces (for purposes of comparison only) until its length is equal to the larger.

The paths of this syntax diagram are explained below:

Path	Explanation
A	Take this path if the subsequently specified relation is to be tested.
B	NOT is specified if the opposite of the subsequently specified relation is to be tested.
C-E	Take one of these paths to specify an equal relation.
F-I	Take one of these paths to specify a less-than relation.
J-L	Take one of these paths to specify a greater-than relation.
M	Take this path to specify a less-than-or-equal relation.
N	Take this path to specify a greater-than-or-equal relation.
O	Take this path to specify a not-equal relation.
P	Take this path to specify an alternate relation.

Q Take this path when the relation is fully specified.

Examples:

```
=
NOT =
>
GREATER THAN OR EQUAL
IS EQUAL TO
LEQ
```

REPLACE STATEMENT

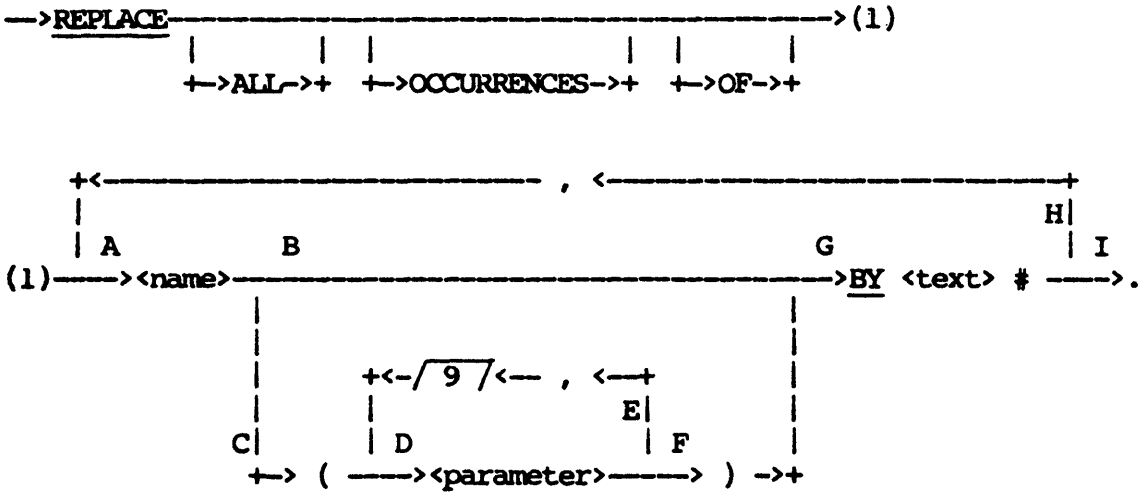
The REPLACE statement is used to define a name that represents a portion of language text consisting of characters, words, or phrases. This name can then be used to reference the text at any appropriate point within the specification. This provides a convenient shorthand technique for expanding a REPORTER III report language statement.

Example:

```
REPLACE ALL OCCURRENCES OF WRITE BY
   EXTRACT WITH VOCABULARY #.
```

This defines WRITE as a text replacement name. All occurrences of the word WRITE are replaced by the phrase "EXTRACT WITH VOCABULARY". If "WRITE A,B,C to EF-FILE." is specified, the Report Language Analysis Program reads "EXTRACT WITH VOCABULARY A,B,C TO EF-FILE". Thus a new command called WRITE has been created for use within the specification.

The syntax for the REPLACE statement is as follows:



The paths of this syntax diagram are explained below:

Path	Explanation

A Take this path to define a <name> to represent the <text> of the REPLACE statement. You then can use this <name> subsequently to reference the <text>. The <name>, sometimes referred to as a macro name, must not be the same as another macro name. (Macro names defined in one Report Section can be identical to macro names defined in other Report Sections.)

Example: ("INVENT")

```
REPLACE ALL OCCURRENCES OF TIMES BY * #.
```

This defines TIMES to represent the text "*". Now the following statement can be specified:

```
TOTAL-COST IS QUANTITY TIMES UNIT-COST.
```

This is interpreted by the Report Language Analysis Program as if "TOTAL-COST IS QUANTITY * UNIT-COST" was specified.

B Take this path if the text which is to replace the macro name at every occurrence of the macro name is to be unaltered. In this case, the same text replaces all occurrences of the macro name as the language statements are scanned. The example given for path A shows this path choice: "*" replaces "TIMES" wherever "TIMES" occurs.

C Take this path if the text represented by the macro name is to be modified each time it replaces the macro name. In this case, the text is modified as specified before it replaces an occurrence of the macro name.

Example:

```
REPLACE AGED (X) BY AGE(IN WEEKS FROM
   X TO DATE) / 4#.
```

The text associated with "AGED" is altered each time "AGED" is referenced. For instance, if

```
REPORT BILLING-AGE IS AGED(INVOICE-DATE).
```

is specified, REPORTER III reads:

```
REPORT BILLING-AGE IS AGE (IN WEEKS FROM
   INVOICE-DATE TO DATE ) / 4.
```

The macro as defined here performs as a new function definition.

D The mechanism which you use to specify how the text is to be modified consists of formal parameters associated with the macro name. A formal parameter is a word that is a place holder for an actual parameter. The actual parameter is the text you enter in a subsequent language statement(s) as a substitute for the formal parameter. Take path D to specify the formal parameter. You can use the formal parameter one or more times in the body of the text associated with the macro name.

At the time the macro name is referenced, the actual parameter (the desired text) replaces the formal parameter. When the text replaces the macro name, the supplied actual parameter replaces the formal parameter.

Example:

```
REPLACE TYPE1(N) BY N WHERE TYPE = 1#.
```

When the macro name TYPE1 is referenced, an actual parameter must be supplied for N.

```
REPORT NAME, TYPE1(YEAR), TYPE1(NAME OF SUPR).
```

This statement is expanded to:

```
REPORT NAME, YEAR WHERE TYPE = 1, NAME OF SUPR
   WHERE TYPE = 1.
```

E Take this path up to nine times to specify a maximum of ten formal parameters for the macro name. When the macro name is referenced in a subsequent language statement(s), the actual parameters (which you supply) are assigned to the formal parameters in a left to right manner. You must specify one actual parameter for each formal parameter each time the macro name is referenced.

Example:

```
REPLACE TAX(INC-RATE,CAP-GAIN-R) BY SALARY *
   INC-RATE + CAP-GAIN-R * CAPITAL-GAIN #.
```

If TAX is referenced subsequently as

```
REPORT INCOME-TAX IS TAX(.15,.0500 * VALUE),NAME.
```

.15 replaces INC-RATE, and .0500 * VALUE replaces CAP-GAIN-R. The following equivalent statement results:

```
REPORT INCOME-TAX IS SALARY * .15 + .0500 *
   VALUE * CAPITAL-GAIN, NAME.
```

The same macro could be referenced elsewhere as follows:

```
ACCEPT VALUE-1, VALUE-2.
SELECT (TAX(VALUE-1,VALUE-2)) > 500.00.
```

In this case, VALUE-1 replaces INC-RATE and VALUE-2 REPLACES CAP-GAIN-R. The SELECT statement is expanded to:

```
SELECT (SALARY * VALUE-1 + VALUE-2 *
   CAPITAL-GAIN) > 500.00.
```

F Take this path after you have specified all formal parameters.

G Take this path to define the <text> that is to replace the macro name. The <text> consists of all characters after "BY" before the terminating "#". The <text> cannot contain a "#". The "#" is reserved for indicating the end of the <text>. No syntax checking is performed on the <text> as the REPLACE statement is scanned. Syntax checking occurs as the <text> replaces each occurrence of a macro name.

The text can contain references to the formal parameters defined in path D. The actual parameters (which you specify in subsequent language statements) replace the corresponding formal parameters when the text replaces the macro name.

The text also can contain references to other macro names. These macro name references are not expanded into the associated text at the time the REPLACE statement is scanned, but they are expanded at the time the macro name is referenced. Macros can be nested up to 10 deep. The following example illustrates nested macros. (Simple and parameterized macro texts have been illustrated previously.)

Example: (nested macros)

```
REPLACE A BY B + C #.
REPLACE B BY D * E#.
REPLACE C BY F / G #.
```

If you specify "REPORT T IS A,X,Y." after specifying the above macros, the following replacement steps are taken:

```
REPORT T IS A,X,Y.
REPORT T IS B + C,X,Y.
REPORT T IS D * E + C,X,Y.
REPORT T IS D * E + F / G,X,Y.
```

Thus the original REPORT statement is equivalent to:

```
REPORT T IS D * E + F / G,X,Y.
```

NOTE

Macro names cannot reference themselves. Specifications such as those shown in the examples below must be avoided because they result in an infinite series of references.

Examples:

```
REPLACE A BY A + B#.
```

The above statement is flagged when A is encountered in the text of the macro. Such a REPLACE statement must be avoided.

```
REPLACE A BY B#.
REPLACE B BY A#.
```

If the above statements are specified, an error is flagged if either A or B is referenced because the limit of 10 nested macros is exceeded. Such REPLACE statements must be avoided.

NOTE

If a period appears before the "#" in a REPLACE statement, problems can arise if, when you subsequently reference the macro name, you use a period after the macro name.

Example:

```
REPLACE ONE BY 1.#.
SELECT COST > ONE.
```

Using the above SELECT statement results in

```
SELECT COST > 1..
```

and thus causes an error.

H Take this path as many times as necessary to define additional macro names in the same REPLACE statement as you require. (You optionally can define each macro name in a separate REPLACE statement.)

Example:

```
REPLACE PLUS BY +#, INTERVAL (X,Y) BY
   AGE(IN DAYS FROM X TO Y) /30#.
```

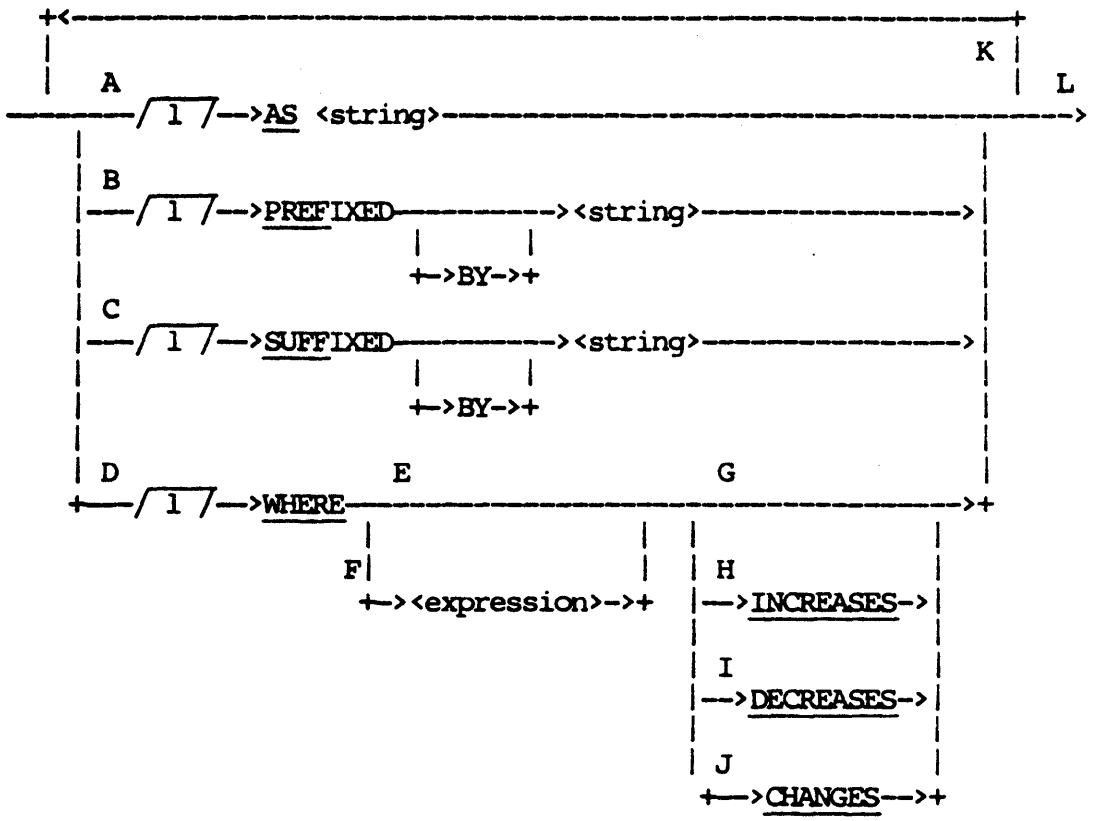
In this case, two macros, PLUS and INTERVAL, are defined.

I Take this path after you have completed all macro definitions.

REPORT-ITEM MOD

The Report-item Mod is used to specify nondefault printing of control-break headings and columns.

The syntax for Report-item Mod is as follows:



The paths of this syntax diagram are explained below:

Path	Explanation
A	Take this path to give a nondefault identifier, or heading, for a data item. The <string> given must be 30 characters or less. If a specified heading is larger than the column size, an attempt is made to separate the heading on space boundaries to form a multiple-line heading.

Example: ("INVENT")

```
REPORT PART-NO AS "PART #", UNIT-COST AS
   "COST PER UNIT".
```

In this example, two columns are specified. The first is headed by "PART #". The second is headed by "COST PER UNIT".

B-C These paths are used to place characters next to a data value in the listing without any intervening spaces. PREFIXED places the <string> in front of the data value, while SUFFIXED places the <string> behind the data value. In both cases, the <string> must be 30 characters or less.

Example:

```
REPORT ORDER-NO PREFIXED BY "#", AMOUNT
   SUFFIXED BY " LBS."
```

This produces the following report:

```
ORDER          AMOUNT
  NO

#0021          38 LBS.
#0117           4 LBS.
#0235         582 LBS.
```

D Take this path to suppress printing of certain data values under specified conditions. The data value is printed whenever the condition specified by the following paths is met. If the condition is not met, blanks are printed instead of data value.

Example:

```
REPORT NAME, SALARY WHERE SALARY < 20000.
```

In this case, two columns are printed as in the following listing. NAME is always printed, but the corresponding SALARY entry is left blank if the salary is greater than or equal to $20,000.

NAME	SALARY
JOE SMITH	$12111.00
BOB ADAMS	
JANE PHILLIPS	$18000.00
JIM TAYLOR	
MARY JENKINS	$13500.00

E This path is a shorthand method for specifying implicitly an <expression> which is equal to the value of the data item reported in this clause. You must use path H, I, or J with this path.

Example: ("CUSTV")

```
REPORT CUST-NAME ASC WHERE CHANGES, NET-PAYMT.
```

The column for CUST-NAME contains a customer name only when the customer name for a remittance differs from the customer name for the previous remittance; that is, duplicate customer names other than the first are suppressed.

F Take this path to specify an <expression> which defines the condition under which data items are to be printed. The <expression> can be a Boolean, numeric, or string value. If the <expression> is Boolean (the result is either true or false), path G must be taken. If the <expression> is a numeric or string value, path G is illegal, and the <expression> must be supplemented by paths H, I, or J. The <expression> can consist of any type of data items.

G Take this path if the <expression> given in path F above is Boolean, and therefore determines the printing condition.

Example: ("VOCAST")

```
REPORT ASSET-DESC, ASSET-LIFE WHERE
   ASSET-LIFE < 70, COST.
```

In this case, three columns are printed. The ASSET-LIFE column is blank when ASSET-LIFE is greater than or equal to 70.

H Take this path to specify printing only when the value of the <expression> given in path F or implied by path E increases from the previous value. The <expression> must be either a numeric or string value.

I Take this path to specify printing only when the value of the <expression> given in path F or implied by path E decreases from the previous value. The <expression> must be either a numeric or string value.

J Take this path to specify printing only when the value of the <expression> given in path F or implied in path E differs from the previous value. The <expression> must be either a numeric or string value.

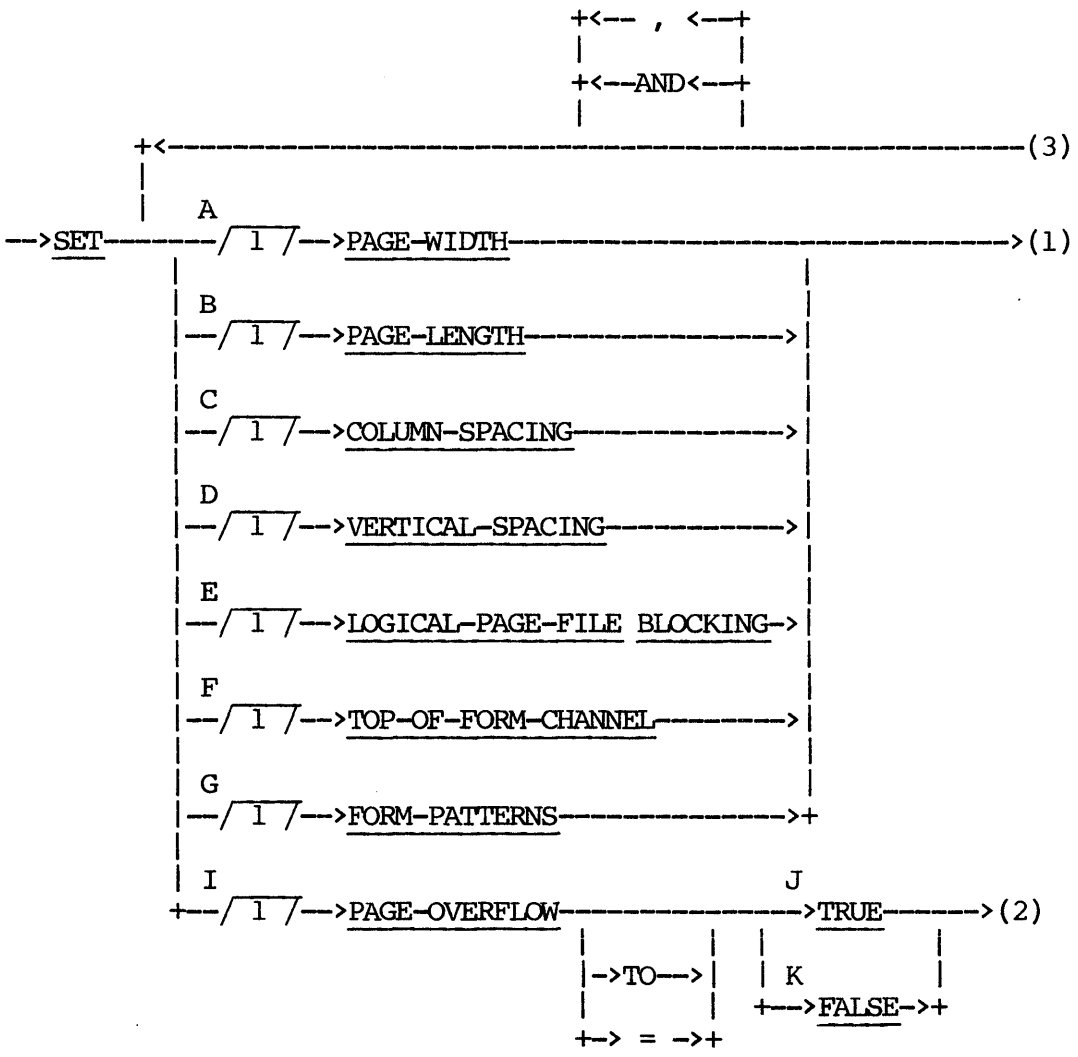
K Take this path as many times as necessary to specify all nondefault options. The options can be used in any combination independent of one another.

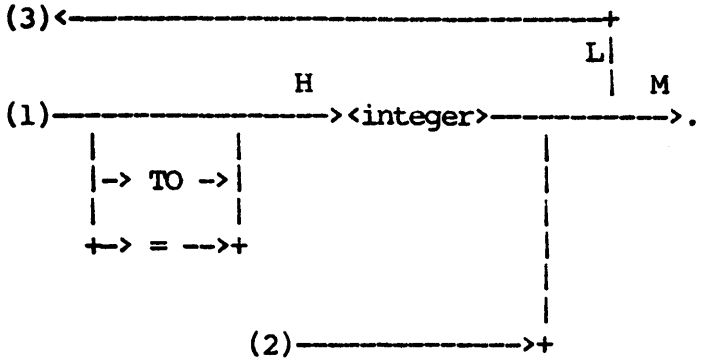
L Take this path after all options are specified.

REPORT-OPTION SET STATEMENT

The Report-option SET statement enables certain formatting attributes to be assigned nondefault values. These formatting attributes only apply to the REPORT statement. If the default values are satisfactory, this statement is not necessary.

The syntax for the Report-option SET statement is as follows:





The paths of this syntax diagram are explained below:

Path	Explanation

A This path is taken to specify a nondefault PAGE-WIDTH, the number of horizontal print positions allocated for the report. The default value is 132. The minimum PAGE-WIDTH that may be specified is 32.

Example:

```
SET PAGE-WIDTH TO 80.
```

The generated report listing is 80 characters across.

B Take this path to specify the maximum number of printed lines per page for the report. The default value is 54.

Example:

```
SET PAGE-LENGTH TO 72.
```

The generated report listing contains up to 72 lines per page.

C Take this path to specify a nondefault COLUMN-SPACING, the minimum number of print positions between columns of a report. If this number of positions cannot be achieved on a page, line or page overflow results. The default value is 1.

Example:

```
SET COLUMN-SPACING TO 5.
```

The minimum number of spaces between columns of information in the generated report is 5.

D Take this path to specify a nondefault VERTICAL-SPACING, the number of lines advanced before printing the next line. This pertains only to the lines of a column listing. The default value is 1 (single spacing) if line overflow does not occur; it is 2 (double spacing) if line overflow does occur in the report. The only allowable values are 1, 2, or 3.

Example:

```
SET VERTICAL-SPACING TO 2.
```

The information listed in the columns for the report in this example is double-spaced.

E Take this path to specify the blocking factor for the internal logical page work file; the path is meaningful only if page overflow occurs. The integer specifies the number of records in a block. If this option is not specified, the value selected at vocabulary creation time is used.

Example:

```
SET LOGICAL-PAGE-FILE BLOCKING TO 6.
```

F This option is given to override the VERTICAL-SPACING form attribute, either given or defaulted in the PRINT statement. The <integer> given provides the channel number which corresponds to the first printable line on a form.

The TOP-OF-FORM-CHANNEL option provides a convenient means to specify the appropriate channel number when VERTICAL-SPACING was inappropriately given in a macro.

Example:

```
SET TOP-OF-FORM-CH TO 3.
```

This statement specifies that channel 3 corresponds to top-of-form.

G If this option is given, a form pattern must be defined through the WITH PATTERN specification within a PRINT statement. At this point, <integer> form patterns are printed for operator alignment before the printing of actual forms begins.

Example:

```
SET FORM-PATTERNS TO 10.
PRINT....WITH PATTERN....
```

Ten form patterns are written to the forms file before the first actual form is printed.

H Take this path to specify the integer value for the previously specified option. (Appendix B lists the acceptable values for these options.)

I Take this path if you desire to specify whether or not PAGE-OVERFLOW is to be used as a formatting method. PAGE-OVERFLOW is a formatting method that can be used when the width of the report exceeds the width of a page. See the examples given in the subsection titled Columns for the difference in setting the option to TRUE or FALSE.

J Take this path to specify that PAGE-OVERFLOW is to be used.

Example:

```
SET PAGE-OVERFLOW TO TRUE.
```

K Take this path to specify explicitly that PAGE-OVERFLOW is not to be used. Line overflow is used when required. This is the default.

L Take this path to specify additional nondefault options in the same statement.

Examples:

```
SET PAGE-WIDTH = 80,
    PAGE-LENGTH = 72.
SET PAGE-LENGTH = 50,
    COLUMN-SPACING = 10
    AND VERTICAL-SPACING TO 2.
```

M Take this path when all desired report SET options are specified.

REPORT-OPTION STATEMENT

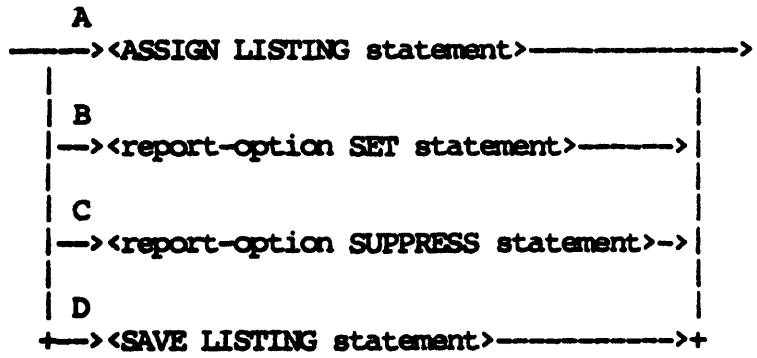
The Report-option statement enables you to override various default values and actions unique to each report.

Example:

```
SAVE LISTING AS "JUDREP".
```

The listing is labeled "JUDREP" instead of the default label "RLnnrr", where nn is an assigned report ID number, and rr is the report number.

The syntax for the Report-option statement is as follows:



The paths of this syntax diagram are explained below:

Path	Explanation
A	This path allows the report or forms listing to be assigned to a particular device. Example: `ASSIGN LISTING TO PRINTER BACKUP.` In this example, the generated report program produces the report on a printer backup file.

B This path allows various formatting limits to be set to nondefault values.

Example:

```
SET PAGE-WIDTH TO 60.
```

In this example, the generated program produces a report only 60 characters wide.

C Take this path to suppress the automatic generation of page numbers in the report produced by the generated program.

Example:

```
SUPPRESS PAGE-NUMBERS.
```

D Take this path to specify a nondefault name for the report or forms listing produced by the generated report program.

Example:

```
SAVE LISTING AS "JUDREP".
```

REPORT-OPTION SUPPRESS STATEMENT

The Report-option SUPPRESS statement suppresses the printing of page numbers at the top of each page of the report.

Example:

```
SUPPRESS PAGE-NUMBERS.
```

The syntax for the Report-option SUPPRESS statement is as follows:

```
-->SUPPRESS PAGE-NUMBERS.
```

REPORT STATEMENT

The REPORT statement is used to specify the layout of an automatically formatted report and the data items to be included in it.

Example: ("VOCAST")

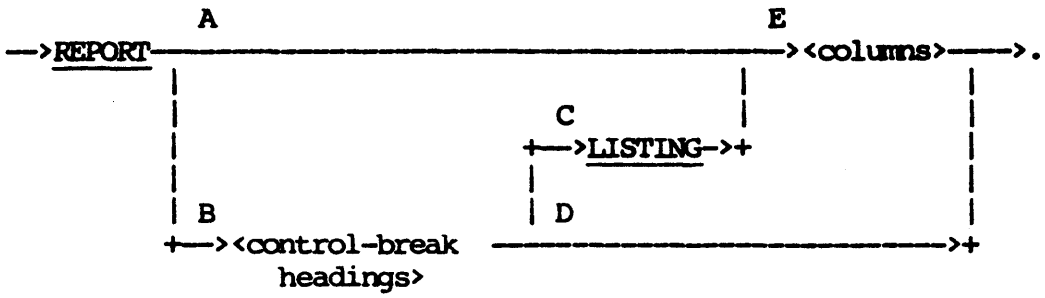
```
REPORT ASSET-NO, ASSET-DESC, LOC-CODE, COST.
```

In this example, the data items ASSET-NO, ASSET DESC, LOC-CODE, and COST are printed as columns of data with appropriate headings (see Figure 4-9).

ASSET NO	ASSET DESC	LOC CODE		COST
003118	STEEL BOOK CASE NO. 5587	122113	$	151.83
003119	STEEL BOOK CASE NO. 6187	122214	$	210.11
002000	TYPEWRITER SERIAL NO. 77-3115	122113	$	322.05
002180	CALCULATOR NO. 7112-36602	112220	$	310.00
005571	TYPEWRITER SERIAL NO. 77-3845	112214	$	282.70

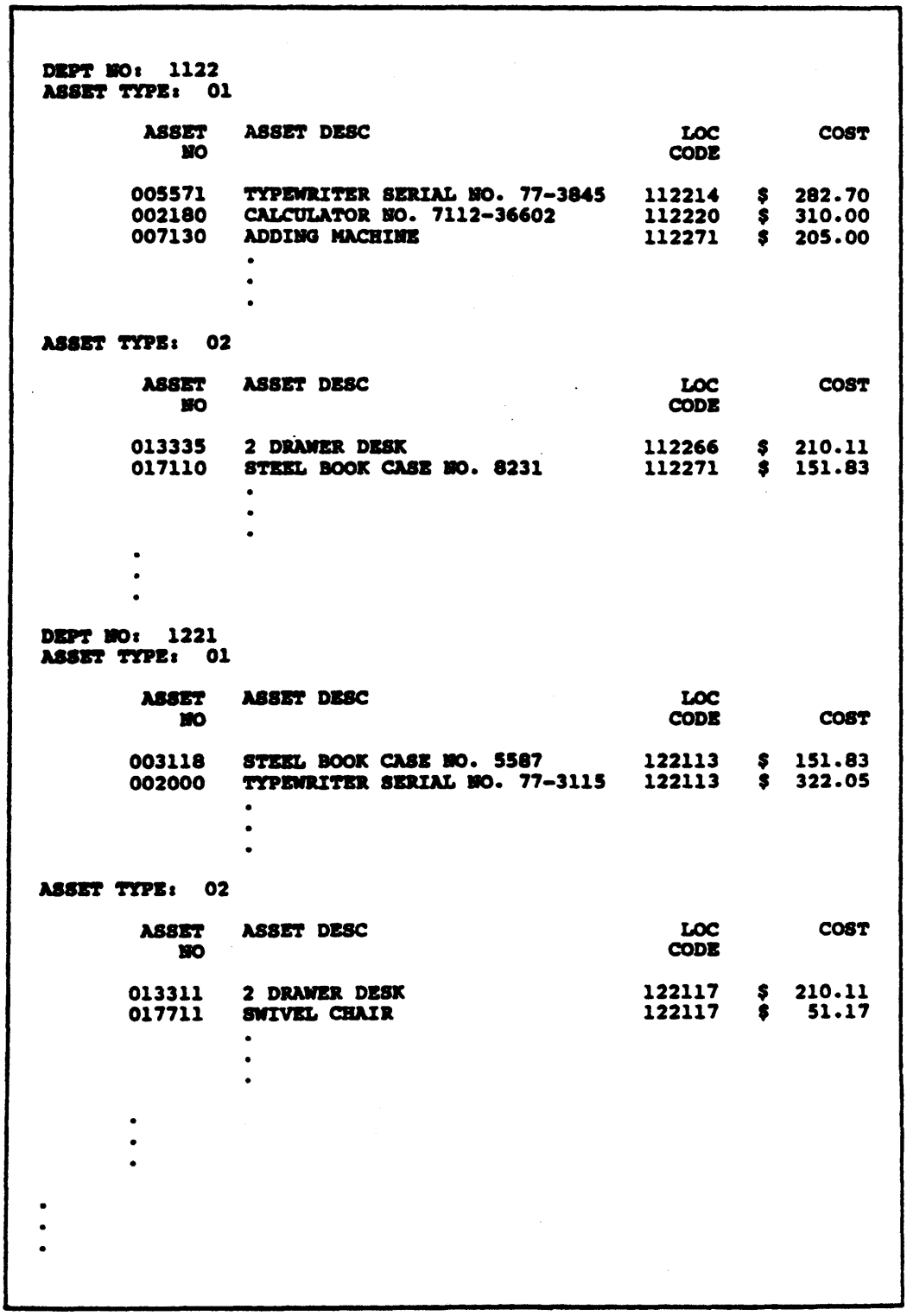
Figure 4-9. Sample Report Output

The syntax for the REPORT statement is as follows:



The paths of this syntax diagram are explained below:

Path	Explanation
A	Take this path to specify printing of only columns of data in the report, omitting control-break headings. The information in the report will not be printed in sections grouped by control-break headings.
B	Take this path if the report is to be printed in sections based on the values of one or more control-break items. <Control-break headings> introduce the information in each section. You can specify one or more <control-break headings> for each section created when a control-break item changes. You also have the option of printing associated information with each control-break item. Example: ("VOCAST") REPORT BY DEPT-NO; BY ASSET-TYPE LISTING ASSET-NO, ASSET DESC, LOC-CODE, COST. The underlined portion specifies two control breaks. Each time the value of ASSET-TYPE changes, a heading containing this new value is printed. Each time the value of DEPT-NO changes, a heading containing the new value of DEPT-NO is printed and a heading containing the value of the first ASSET-TYPE for the new DEPT-NO is printed (see Figure 4-9).
C	After specifying <control-break headings>, take this path to specify the content of the columns to be printed. For each value of the lowest level control break, new column headings are printed and new detail columns are started. Example: ("VOCAST") REPORT BY DEPT-NO; BY ASSET-TYPE LISTING ASSET-NO, ASSET-DESC, LOC-CODE, COST. After each control-break heading for ASSET-TYPE, columns for ASSET-NO, ASSET-DESC, LOC-CODE, and COST are started (see Figure 4-10).



```
DEPT NO:  1122
ASSET TYPE:  01

        ASSET    ASSET DESC                          LOC          COST
          NO                                        CODE

       005571    TYPEWRITER SERIAL NO. 77-3845     112214    $  282.70
       002180    CALCULATOR NO. 7112-36602         112220    $  310.00
       007130    ADDING MACHINE                    112271    $  205.00
                 .
                 .
                 .

ASSET TYPE:  02

        ASSET    ASSET DESC                          LOC          COST
          NO                                        CODE

       013335    2 DRAWER DESK                     112266    $  210.11
       017110    STEEL BOOK CASE NO. 8231          112271    $  151.83
                 .
                 .
                 .
       .
       .
       .

DEPT NO:  1221
ASSET TYPE:  01

        ASSET    ASSET DESC                          LOC
          NO                                        CODE         COST

       003118    STEEL BOOK CASE NO. 5587          122113    $  151.83
       002000    TYPEWRITER SERIAL NO. 77-3115     122113    $  322.05
                 .
                 .
                 .

ASSET TYPE:  02

        ASSET    ASSET DESC                          LOC          COST
          NO                                        CODE

       013311    2 DRAWER DESK                     122117    $  210.11
       017711    SWIVEL CHAIR                      122117    $   51.17
                 .
                 .
                 .

       .
       .
       .

.
.
.
```

Figure 4-10. Sample Report Listing

D Take this path if no column listings are desired.

Example: ("VOCAST")

```
REPORT BY DEPT-NO; BY ASSET-TYPE.
```

This specifies a heading for each department and a heading for each asset type per department. No columns of detailed items are produced (see Figure 4-11).

```
          .
          .
          .
DEPT NO:  1122
ASSET TYPE:  01
ASSET TYPE:  02
ASSET TYPE:  03
          .
          .
          .
DEPT NO:  1221
ASSET TYPE:  01
ASSET TYPE:  02
ASSET TYPE:  07
          .
          .
          .
```

Figure 4-11. Listing by Department Number

E The <columns> clause specifies the number and content of columns to be printed. Headings for these columns can be either implied or specifically defined.

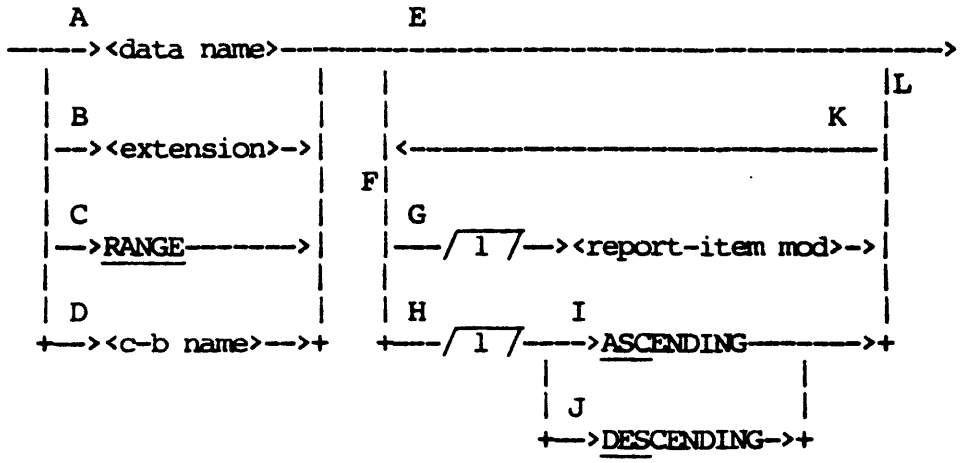
Example: ("VOCAST")

```
REPORT ASSET-NO, ASSET-DESC, LOC-CODE, COST.
```

Four columns are defined, each line of information representing the values of the items ASSET-NO, ASSET-DESC, LOC-CODE, and COST within one logical record (see Figure 4-9).

Row Desc defines the first column and specifies that each line of columns represents summary information related to a control-break item value. The referenced item is defined as a control-break item if control breaks for the report have not been defined previously in another statement.

The syntax for the Row Desc clause is as follows:



The paths of this syntax diagram are explained below:

Path	Explanation
A	Take this path to specify that summary information be printed for each value of a data item referenced by <data name>. The item must be nonstatistical and is defined to be a control-break item by its reference here. Control breaks must not have been specified previously for the report in another statement. If any <c-b-heading desc>s were specified previously in the REPORT statement, the control-break item referenced by <data name> must be the lowest-level control break. Example: ("VOCAST") REPORT FOR EACH ASSET-TYPE: AVG(COST) NUM(6,2).

Each line in the listing consists of an asset type and the average asset cost for that asset type. The following type of listing would be produced:

```
ASSET          AVG
 TYPE        (COST)

   01         88.71
   02         17.33
   03         62.80
    .           .
    .           .
    .           .
```

B Take this path to specify that summary information be printed for each value of a data item defined by <extension>. The derived data item must be nonstatistical. The item is defined to be a control-break item; control breaks for the report must not have been specified previously in another statement.

If any <c-b-heading desc>s have been specified previously in the REPORT statement, the control-break item defined by the <extension> must be the lowest-level control-break item.

Example: ("VOCAST")

```
REPORT FOR EACH COST-RANGE WHICH IS
   COST/100 NUM(6) ROUNDED: COUNT.
```

C Take this path to specify that summary information be printed for each value of a range-break item defined previously by a RANGE statement.

Example: ("INVENT")

```
RANGE BY INVENT-DOLLAR-VALUE FROM 0 BY 1000.
    .
    .
    .
REPORT FOR EACH RANGE: COUNT,
   TOTAL(INVENT-DOLLAR-VALUE),
      AVG(INVENT-DOLLAR-VALUE).
```

The following report is produced:

RANGE	COUNT	TOTAL (INVENT DOLLAR VALUE)	AVG (INVENT DOLLAR VALUE)
0.00 - 1000.00	24	11310.15	471.25
1000.00 - 2000.00	12	15960.00	1330.00
2000.00 - 3000.00	10	21020.83	2101.08
.	.	.	.
.	.	.	.
.	.	.	.

D Take this path to specify that summary information be printed for each value of a control-break item previously defined in a GROUP statement. <c-b name> must reference this item.

If no <c-b-heading desc>s were specified in the REPORT statement, <c-b name> must reference the highest-level control-break item in the GROUP statement.

If any <c-b-heading desc>s were specified previously in the REPORT statement, <c-b name> must reference the next lower-level control-break item, based on the order in the GROUP statement (all control breaks higher than <c-b name> must have been used as control-break headings).

Example: ("VOCEMP")

```
GROUP BY EDUCATION-LEVEL, BY JOB-GRADE,
   BY DEPT-NO.
        .
        .
        .
REPORT BY EDUCATION-LEVEL LISTING FOR EACH
   JOB-GRADE: AVG(SALARY), MAX(SALARY),
      MIN(SALARY).
```

The following report is produced:

EDUCATION LEVEL: 10

JOB GRADE	AVG (SALARY)	MAX (SALARY)	MIN (SALARY)
01	$ 8711.80	$ 9830.00	$ 6520.00
02	$ 9500.50	$ 11800.80	$ 8005.70
03	$ 10111.71	$ 13170.00	$ 8123.80
.	.	.	.
.	.	.	.
.	.	.	.

E Take this path if path A or B was taken and default control-break ordering and default printing of the control-break item values is desired, or if path C or D was taken and default printing of the control-break item values is desired. By default, control-break item values are sorted and printed in ascending sequence.

Each control-break item value or each value range appears in the first column. A default identifier based on the control-break item or range-break item name supplies the column heading.

F Take this path to specify ordering of information based on the control-break item and/or to specify nondefault column printing. The nondefault printing options are as follows:

1. Change the column heading.
2. Specify strings for prefixes and/or suffixes.
3. Specify suppression of certain values in the column.

G Take this path to specify nondefault column printing. Path E above lists the nondefault options available.

Example: ("VOCAST")

```
REPORT FOR EACH RANGE
   AS "ASSET COST LIMITS": COUNT.
```

The first column of the listing is headed by "ASSET COST LIMITS".

H Take this path to specify explicitly the order in which control-break values are to be printed. This path cannot be taken if path C was taken or if path D was taken and a range-break item was referenced.

I Take this path to specify explicitly that information be ordered in ascending sequence based on the control-break item.

J Take this path to specify that information be ordered in descending sequence based on the control-break item.

Example: ("VOCAST")

```
REPORT FOR EACH ASSET-TYPE DESCENDING
   AVG(COST), COUNT.
```

In this case, the rows of summary information are printed such that asset type is in descending sequence.

K Take this path to specify both explicit ordering and nondefault printing of the column.

L Take this path after all nondefault specifications are made.

SAMPLE STATEMENT

The SAMPLE statement provides the capability of sampling selected input information (that information which was input or selected from input). If a SAMPLE statement is specified, only sampled information is reported (grouped, ordered, summarized, printed, and/or extracted).

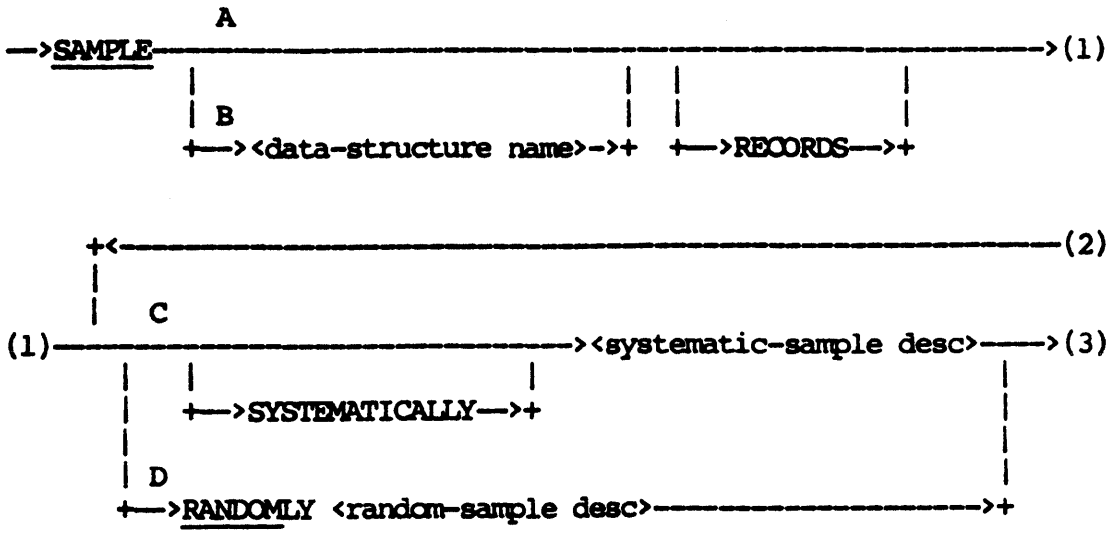
Sampling can be done systematically or randomly based on logical records or records of a particular data structure. Sampling also can be stratified by using a strata-defining Boolean expression in the sample description. If no SELECT INPUT statement is specified, the sample is drawn from the input information. If a SELECT INPUT statement is provided, the sample is drawn from the selected information.

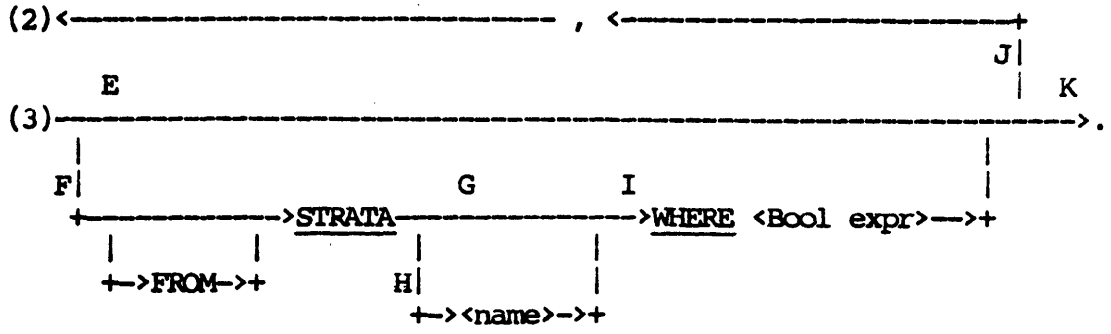
Example:

```
SAMPLE EVERY 10 RECORDS.
```

The resulting sample consists of every tenth record from the total population of input or selected input logical records.

The syntax for the SAMPLE statement is as follows:





The paths of this syntax diagram are explained below:

Path	Explanation

A Take this path if the sampling is to be based on logical records rather than one of a number of data structures specified in the INPUT statement.

Example:

```
SAMPLE EVERY 10 RECORDS.
```

This sample consists of every tenth logical record.

B Take this path if the sampling is to be based on records accessed from a particular input data structure. This path can only be taken if the SAMPLE statement is given in the Input Section of a multiple-report specification.

Example:

```
INPUT BRANCHES(ACCOUNTS(INVOICES)).
SAMPLE ACCOUNTS SYSTEMATICALLY EVERY 10.
```

In this example, every tenth ACCOUNTS record accessed becomes part of the sample. For each of the sampled ACCOUNTS records, all related INVOICES are accessed and the resulting logical records are included in the sample.

C Take this path if records are to be sampled systematically. The <systematic-sample desc> describes how the systematic sample is taken.

Example:

```
SAMPLE EVERY 10 RECORDS.
```

Every tenth logical record is included in the sample.

D Take this path if records are to be sampled randomly. The <random-sample desc> describes how the random sample is to be taken.

Example:

```
SAMPLE RANDOMLY APPROX 10 PERCENT OF ALL RECORDS.
```

The sample consists of approximately 10 percent of all logical records input or selected from input.

E Take this path if stratified sampling is not needed. The default strata from which the sample is drawn is assumed to be all selected-input information. If path A was taken, the exact population to be sampled is the number of logical records which were input or selected from input via a SELECT INPUT statement. If path B was taken, the exact population to be sampled is the number of records input and selected via a SELECT INPUT statement from the specified data structure.

F Take this path if stratified sampling is desired. The strata consists of all selected input information satisfying the defining Boolean expression. This strata is sampled as described. Only the sampled information is available for reporting.

Example: ("VOCAST")

```
SAMPLE EVERY 10 RECORDS
FROM STRATA WHERE COST > 5000.00.
```

Approximately 10 percent of all assets whose cost is greater than 5000.00 dollars are sampled.

G Take this path if the strata is not named.

H Take this path to name the strata for later reference. The <name> specified then can be used to identify the sampled information (logical records) from the strata. The <name> can be referenced as a data item which is 1 (TRUE) if the logical record is in the strata or 0 (FALSE) if it is not in the strata.

Example:

```
SAMPLE EVERY 100 RECORDS FROM STRATA
       SMALL-ACCOUNTS WHERE BALANCE < 10000,
   EVERY 10 RECORDS FROM STRATA BIG-ACCOUNTS
        WHERE BALANCE GEQ 10000.
                .
                .
                .
SELECT SAMPLED INFORMATION WHERE
   SMALL-ACCOUNTS = TRUE.
```

The above SELECT statement selects only sampled information from the strata SMALL-ACCOUNTS.

I The <Bool expr> defines the strata to be sampled. If path B was taken, the <Bool expr> can reference only accepted data items and data items contained in the specified data structure. The Boolean expression must be nonstatistical.

Example: ("CLIENT")

```
SAMPLE ACCTS-RECV EVERY 4 RECORDS
   STRATA WHERE NINETY-DAYS-DUE > 10000.00.
```

In this example, the strata population consists of all ACCTS-RECV records satisfying the condition that BALANCE NINETY DAYS DUE is over 10,000 dollars. Every fourth such record is included in the sample.

J Take this path to specify additional sample descriptions and strata specifications. Information which is not part of any defined STRATA is not sampled and therefore unavailable for reporting.

Example: ("CLIENT")

```
SAMPLE SYSTEMATICALLY EVERY 100 RECORDS FROM
   STRATA WHERE BALANCE-DUE < 5000,
   RANDOMLY APPROX 15 PERCENT OF ALL RECORDS FROM
       STRATA WHERE BALANCE-DUE GEQ 5000 AND
            BALANCE-DUE < 10000,
   RANDOMLY 51 OUT OF 514 RECORDS FROM STRATA
       WHERE BALANCE-DUE GEQ 10000.
```

This sample consists of every 100th account for which BALANCE-DUE is less than 5000, approximately 15 percent of all accounts for which BALANCE-DUE is greater than or equal to 5000 but less than 10000, and 51 out of 514 accounts for which BALANCE-DUE is at least 10000.

K Take this path when all desired sample descriptions and strata are specified.

SAVE LISTING STATEMENT

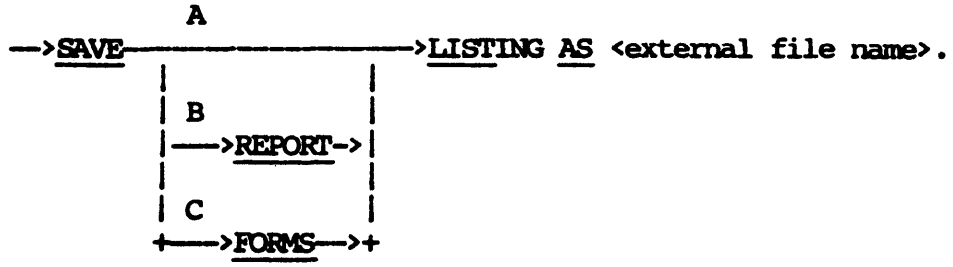
The SAVE LISTING statement causes the automatically formatted or user-formatted report listing produced by the generated report to be named <external file name> if assigned to an appropriate hardware device. (Refer to the ASSIGN LISTING statement in this section.)

Example:

```
SAVE REPORT LISTING AS "EMPPAY".
```

The automatically formatted report listing is labeled "EMPPAY".

The syntax for the SAVE LISTING statement is as follows:



The paths of this syntax diagram are explained below:

Path	Explanation
A	Take this path to identify the automatically formatted report listing.
B	This path is identical in meaning to path A.
C	This path is taken to identify the user-formatted report listing and applies only to reports specified with the PRINT statement.

Example:

```
SAVE FORMS LISTING AS "FML01".
```

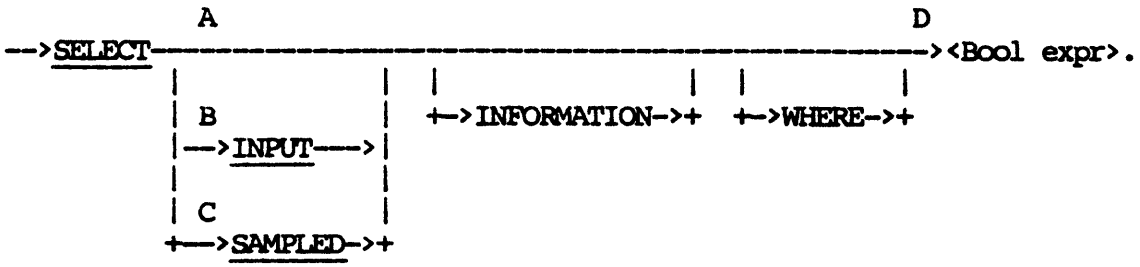
The user-formatted report is labeled "FML01".

SELECT STATEMENT

The SELECT statement describes what information (logical records) is to be reported. If no SELECT statement is provided, all logical records which were input and possibly sampled are reported; no information is suppressed.

The SELECT statement should be contrasted to the action of the various WHERE clauses in Report-Item Mod (refer to REPORT-ITEM MOD in this section). The WHERE clauses suppress data items from a specific application, while the SELECT statement suppresses entire logical records from all applications.

The syntax for the SELECT statement is as follows:



The paths of this syntax diagram are explained below:

Path	Explanation
A	Take this path to specify selection from the information input. Logical records suppressed by this option are not available for sampling, reporting, extracting, or summarizing. Only one SELECT from input information can be specified in a single-report specification, Input Section, or Report Section.

Example: ("VOCAST")

```
SELECT COST > 100.00.
```

In this case, all logical records, or assets, whose cost is less than or equal to $100 are suppressed and are not sampled, extracted, reported, or summarized.

B Take this path to document the fact that selection is from the information input. The specification of INPUT here produces exactly the same results as taking path A above. Only one SELECT from input information can be given in a single-report specification, Input Section, or Report Section.

Example: ("VOCAST")

```
SELECT INPUT INFORMATION WHERE COST > 100.00.
```

This example produces the same results as the example in path A above.

C Take this path to specify selection from the information sampled; that is, specific information is selected from the sample taken. Logical records suppressed by this option are not available for reporting, extracting, or summarizing. Only one SELECT from sampled information can be specified in a single-report specification, Input Section, or Report Section.

If a SAMPLE statement is not specified, this path specifies the same selection as paths A and B.

Example: ("VOCAST")

```
SELECT SAMPLED INFORMATION WHERE COST > 100.00.
```

In this case, all sampled assets whose cost is less than or equal to $100 are suppressed and are not extracted, reported, or summarized.

D This path specifies the Boolean expression which defines the selection criteria. The expression can be composed of any data item or constant (strings or numbers), but must be a nonstatistical Boolean expression. The expression is evaluated for each logical record, and the result (TRUE or FALSE) determines if the logical record is to be selected.

Examples: ("VOCAST")

```
SELECT ASSET-TYPE = 23.
SELECT (ACQUISITION-YR > 74) OR
   (COST > 5000.00).
```

The above are legal nonstatistical Boolean expressions.

Example: ("INVENT")

```
SELECT SAMPLE WHERE QUANTITY IS GREATER THAN 10
AND YR-SOLD IS LESS THAN 75.
```

The above is a legal Boolean expression.

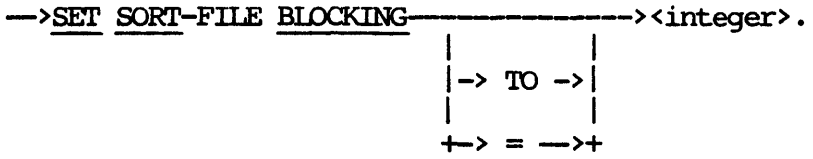
SET SORT BLOCKING STATEMENT

The SET SORT BLOCKING statement is used to change the default blocking factor for the internal sort file. The integer is the number of records in a block. If you do not specify this option, the blocking factor is the value as defaulted or specified by RP3VOC. If a sort is not required, this statement is meaningless.

NOTE

The SET SORT BLOCKING statement is a Data-processing-option statement.

The syntax for the SET SORT BLOCKING statement is as follows:



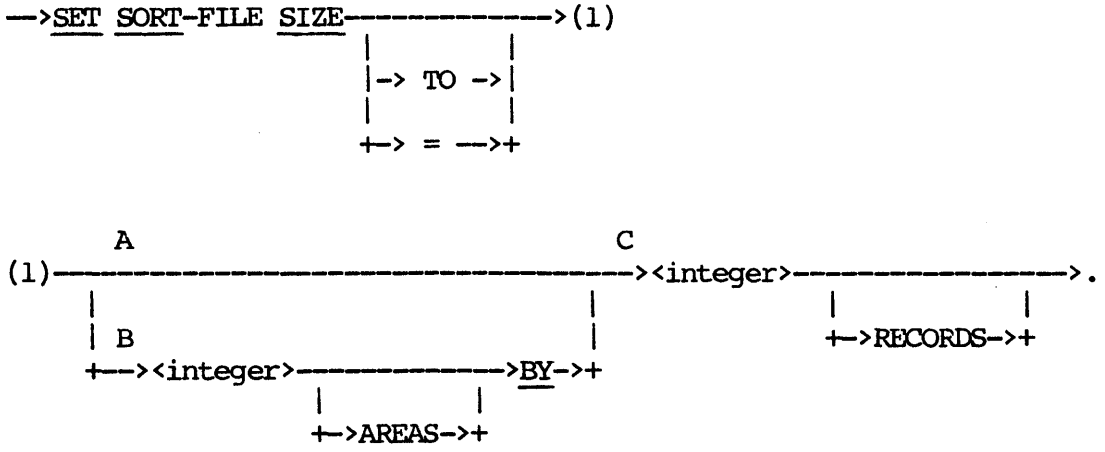
SET SORT SIZE STATEMENT

The SET SORT SIZE statement is used to change the default size of the internal sort file. If you do not use this option, the size is supplied by RP3VOC, either as given in RP3VOC or as defaulted. If a sort is not required, this statement is meaningless.

NOTE

The SET SORT SIZE statement is a Data-processing-option statement.

The syntax for the SET SORT SIZE statement is as follows:



The paths of this syntax diagram are explained below:

Path	Explanation
A	Take this path to specify the sort-file size in terms of records only; only one area is allocated.
B	Take this path to specify the sort-file size in terms of areas and records. The <integer> specifies the number of areas to be allocated. It is advisable that you take this path if you anticipate that the sort file is large, or if the amount of available disk is limited.

C If you have taken path A, the <integer> must specify, as a minimum, the total number of records to be sorted. If you have taken path B, the <integer> specifies the number of records in each area. This integer should be a multiple of the blocking factor (refer to the SET SORT BLOCKING statement in this section).

Example:

```
SET SORT-FILE SIZE TO 300 RECORDS.
```

One area sufficient in size for 300 records is allocated for the sort file.

Example:

```
SET SORT SIZE TO 100 BY 100.
```

This specifies that the sort file can accommodate, at most, 10,000 records. Space is allocated in chunks of 100 records at a time.

STATISTICAL EXPRESSION

A statistical expression is an expression which contains one or more statistical functions and/or statistical data items. A statistical expression can be of the type arithmetic, string, or Boolean.

Example:

```
TOTAL (SALARY) / 12
MAX (AGE)
COUNT > 25
```

Example:

```
TOTAL-PAYROLL
TOTAL-PAYROLL + TOTAL-BENEFITS
TOTAL-PAYROLL / 12
```

The preceding expressions are statistical if TOTAL-PAYROLL and TOTAL-BENEFITS reference derived data items defined as follows (refer to EXTENSION in this section):

```
TOTAL-PAYROLL IS TOTAL(SALARY)
TOTAL-BENEFITS IS TOTAL(BENEFITS)
```

STATISTICAL FUNCTION

A Statistical Function is used to describe a statistical item which summarizes a group of logical records which were input and possibly extended, selected from input, sampled, and/or selected from the sample. A statistic may be specified as a count, running count, total, running total, average, maximum, minimum, sum of squares, mean square, variance, or standard deviation.

The scope of a Statistical Function is the group of information (logical records) to be summarized. Statistics can summarize all information being reported or summarize the information related to each value of a control-break item. The scope of the statistic is specified explicitly in the Statistical Function or is inferred by the context in which the function appears.

Example:

```
INPUT STUDENT-LAB-FEE.

GRAND-TOT IS TOTAL(ACCOUNT-BALANCE) NUM(5,2).
SELECT DEPARTMENT > 10.
REPORT BY DEPARTMENT LISTING STUDENT-NAME,
   YEAR-IN-SCHOOL, ACCOUNT-BALANCE.

SUMMARIZE FOR EACH DEPARTMENT
   DEPT-TOT IS TOTAL (ACCOUNT-BALANCE) NUM(5,2),
   PERCENT-OF-GRAND-TOT IS (DEPT-TOT /
      GRAND-TOT * 100) NUM(2,1),
   GRAND-TOT;
```

In this example, GRAND-TOT appears between the INPUT statement and the REPORT statement. The scope of GRAND-TOT will be all of the selected records. While DEPT-TOT uses the same data name (ACCOUNT-BALANCE) as GRAND-TOT, the scope of DEPT-TOT is only all the records associated with a particular department.

The following would be the resulting output:

```
DEPARTMENT:     30

      STUDENT NAME                YEAR IN SCHOOL    ACCOUNT BALANCE

      JANE DOE                          4              21.00
      SUSAN JONES                       3              13.00

 SUMMARIES FOR DEPARTMENT:     30

 DEPT TOT:                34.00
 PERCENT OF GRAND TOT:     42.8
 GRAND TOT                79.44
```

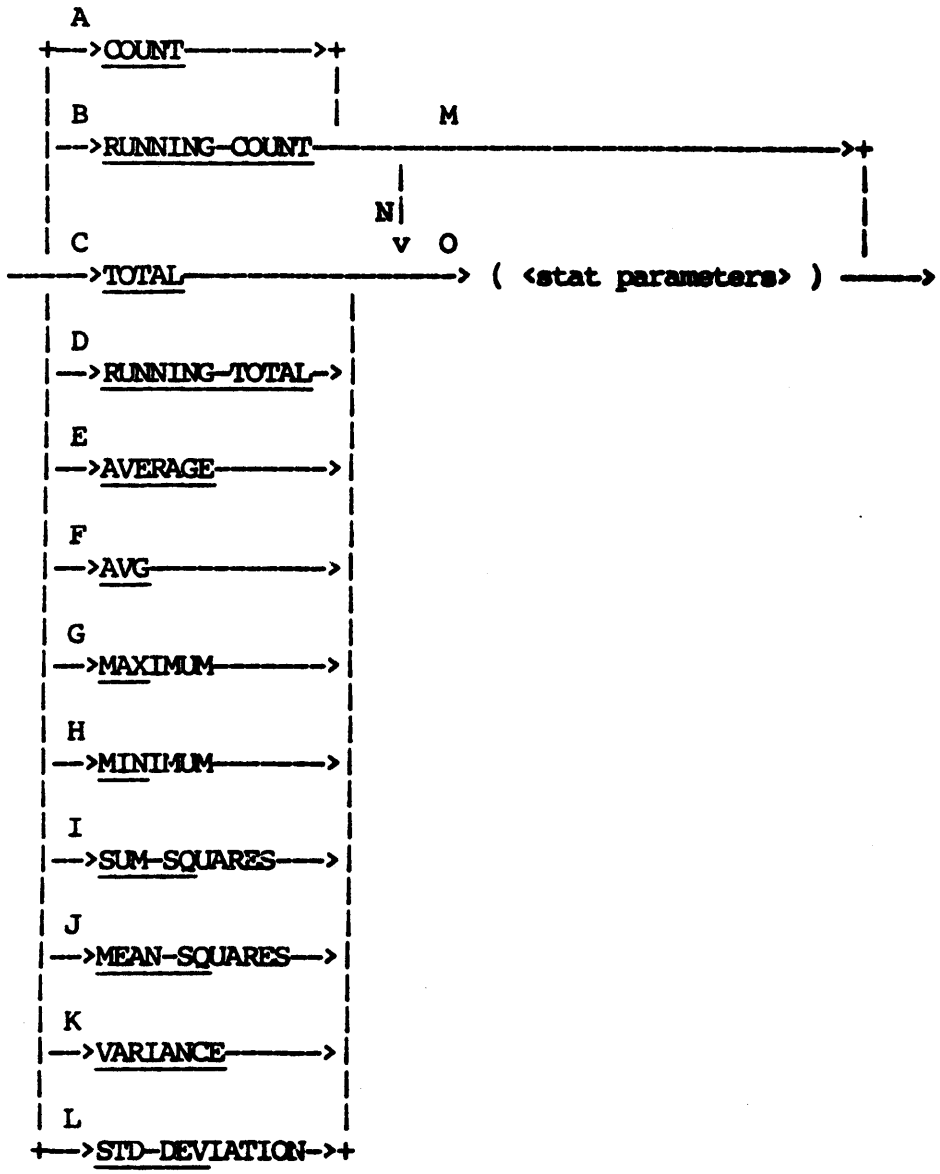
Independent of scope, statistics can be conditional, based on a specified Boolean expression.

Example: ("VOCAST")

```
TOTAL(COST)
```

The preceding Statistical Function represents the total cost of assets.

The syntax for the Statistical Function follows.



The paths of this syntax diagram are explained below:

Path	Explanation
A	Use COUNT to specify the number of logical records or the number of values for the specified data item within the logical records. The value returned is an unsigned integer of INTEGER-SIZE.

Example: ("VOCAST")

COUNT

The preceding statistical function can specify the number of selected assets.

B Use RUNNING-COUNT to provide a running count of logical records or values for the specified data item within the logical records. The value returned is an unsigned integer of INTEGER-SIZE.

Example: ("VOCAST")

RUNNING-COUNT

The above statistical function can be used to provide a unique number for each selected asset.

C Use TOTAL to specify the sum of values for the specified data item within the logical records. The value returned is a signed numeric of INTEGER-SIZE and FRACTION-SIZE.

Example: ("VOCAST")

TOTAL (COST)

The above statistical function specifies the total cost of assets.

D Use RUNNING-TOTAL to provide a running total for the specified data item within the logical records. The value returned is a signed numeric of INTEGER-SIZE and FRACTION-SIZE.

Example: ("VOCAST")

RUNNING-TOTAL (COST)

The above statistical function specifies the running total or accumulation of selected asset cost.

E-F Take one of these paths to specify the average (TOTAL/COUNT) of values for the specified data item within the logical records. The value returned is a signed numeric of INTEGER-SIZE and FRACTION-SIZE.

Example: ("VOCAST")

AVERAGE(COST)

The above statistical function specifies the average cost of assets.

G Use MAXIMUM to specify the largest value of the specified data item within the logical records. The attributes of the value returned are identical to the specified data item.

Example: ("VOCAST")

MAXIMUM(COST)

The above statistical function specifies the maximum cost of an asset.

H Use MINIMUM to specify the smallest value of the specified data item within the logical records. The attributes of the value returned are identical for the specified data item.

Example: ("VOCAST")

MINIMUM(COST)

The above statistical function specifies the minimum cost of an asset.

I Use SUM-SQUARES to specify the sum of the squared values of the specified data item within the logical records. The value returned is a signed numeric of INTEGER-SIZE and FRACTION-SIZE.

Example: ("VOCAST")

SUM-SQUARES(COST)

The above statistical function specifies the sum of the squared cost of assets.

J Use MEAN-SQUARES to specify the average of the sum of squared values (SUM-SQUARES/COUNT) of the specified data item within the logical records. The value returned is a signed numeric of INTEGER-SIZE and FRACTION-SIZE.

Example: ("VOCAST")

MEAN-SQUARES(COST)

The above statistical function specifies the mean squares cost of assets.

K Use VARIANCE to specify the variance values for the specified data item within the logical records, for example, MEAN-SQUARES - (AVERAGE ** 2). The value returned is a signed numeric of INTEGER-SIZE and FRACTION-SIZE.

Example: ("VOCAST")

VARIANCE(COST)

The preceding statistical function specifies the cost variance of assets.

L Use STD-DEVIATION to specify the standard deviation (the square root of VARIANCE of the specified data item within the logical records). The value returned is a signed numeric of INTEGER-SIZE and FRACTION-SIZE.

Example: ("VOCAST")

STD-DEVIATION(COST):

The above statistical function specifies the standard deviation of asset cost.

M Take this path if each logical record is to be counted. The scope of the statistic is all logical records reported or specified by the context in which COUNT is specified.

Example: ("INVENT")

SUMMARIZE FOR EACH DEPARTMENT COUNT.

The count of the number of parts reported for each department is obtained.

N Take this path to specify a condition on which to base the count, to specify the scope explicitly, and/or to specify a count of the number of different control-break values.

Example: ("INVENT")

COUNT (FOR EACH DEPARTMENT)

This specifies, independent of context, that a count is taken of parts for each department.

Example: ("INVENT")

COUNT(WHERE QUANTITY < 100)

This specifies that a count is taken of only those parts where quantity on hand is less than 100.

- Take this path to specify the data items on which the statistic is calculated, the condition on which the statistic is based, and/or the scope over which the statistic is calculated.

Example: ("VOCAST")

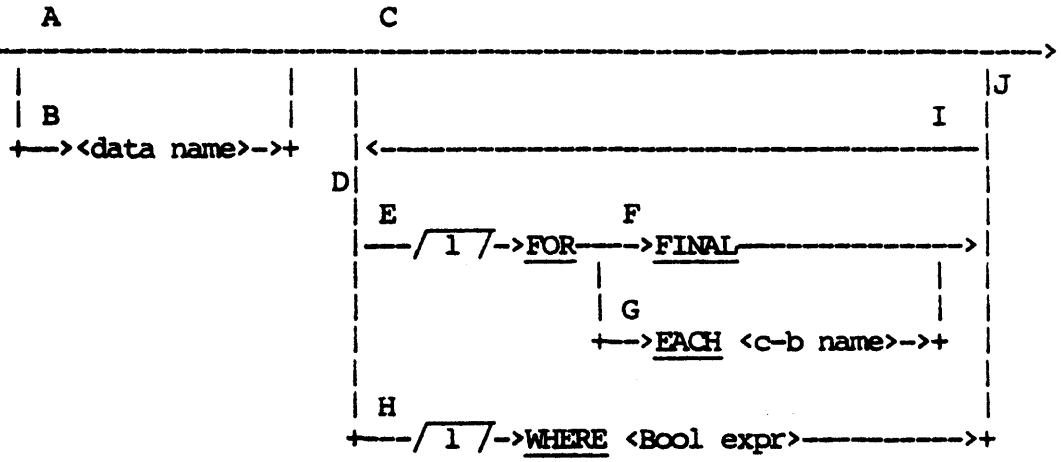
TOTAL(COST)

In this case, the data item COST is totaled. No condition is specified, since the total cost for all asset records is taken. Depending on the context in which TOTAL(COST) is specified, this total can be obtained for all assets being reported or for each value of a control-break item (for example, DEPT-NO) or each range group.

STAT PARAMETERS

Stat Parameters is used to specify the parameters of the statistical function.

The syntax for Stat Parameters is as follows:



The paths of this syntax diagram are explained below.

Path	Explanation

A Take this path only if the COUNT or RUNNING COUNT function is used, and if you desire a count or running count of the number of logical records.

Example: ("CLIENT")

```
COUNT(WHERE NINETY-DAYS-DUE > 0).
```

Only those logical records (accounts) over ninety days due are counted.

B Take this path to specify the <data name> which references the data item to be summarized.

Example: ("INVENT")

```
AVERAGE(INVENT-DOLLAR-VALUE)
```

INVENT-DOLLAR-VALUE refers to a numeric data item.

AVERAGE(INVENT-DOLLAR-VALUE) specifies that the average value of this item is calculated.

If information is grouped by control breaks and the data item specified is a control-break item or is related to a control-break item via a RELATED TO clause, only one value of the data item is calculated into the statistics for each value of the control-break item.

Example:

```
AVERAGE(JOB-GRADE)
AVERAGE(JOB-GRADE RELATED TO JOB-DESC)
COUNT(JOB-DESC)
```

If employee records are grouped by the control-break item, JOB-DESC, the first statistical function specifies the average job grade of employees. However, the second statistical function specifies the average job grade in relation to job descriptions. In this context, JOB-GRADE is used as a summary item. The third statistical function counts the number of different job description values reported, but not the number of logical records (employees). Here again, JOB-DESC is used as a summary item.

C Take this path if the statistic is not based on any condition and if the scope of the statistic is determined by the context in which it appears. If the context does not imply a scope, the default scope is all logical records reported (FOR FINAL).

Example: ("VOCAST")

```
TOTAL(COST)
```

Assets costs are totaled. The scope of the statistic is determined by the statement in which it appears.

Example: ("VOCAST")

```
TOTAL-ASSET-COST IS TOTAL(COST).
```

TOTAL-ASSET-COST references the total cost of all assets reported. The default scope is FOR FINAL.

Example: ("VOCAST")

```
REPORT FOR EACH DEPT-NO:   TOTAL(COST).
```

Here the total cost of all assets reported for a particular department number is reported. The default scope is implied from the context.

D Take this path to specify a condition on which to base the statistic and/or explicitly specify the scope of the statistic.

E Take this path to specify explicitly the scope of the statistic. This specification overrides the default scope implied by the context in which the statistical function is specified.

F Take this path to specify that this statistic is to summarize all information being reported.

Example: ("VOCAST")

TOTAL(COST FOR FINAL)

This specifies the total cost of all assets reported. A single value is obtained.

G Take this path to specify that the statistic is to summarize information for each value of the specified control-break item. <c-b name> must refer to a previously defined control-break item.

Example: ("VOCAST")

TOTAL(COST FOR EACH DEPT-NO)

This specifies the total cost of all assets reported for each department number. A number of values is then obtained.

Example: ("VOCAST")

TOTAL(COST FOR EACH RANGE)

The above specifies the total cost of all assets reported for each value range. (Refer to RANGE statement in this section.)

H Take this path to specify a condition on which the statistic is to be based. The Boolean expression specifies a condition which is evaluated for each logical record. If this condition is evaluated as TRUE, the logical record is figured into the statistic. If the condition is evaluated as FALSE, the logical record is not figured into the statistic.

Example: ("CLIENT")

AVERAGE(BALANCE-DUE WHERE NINETY-DAYS-DUE >0)

The average balance due of all accounts with unpaid balances over ninety days due is obtained.

NOTE

If several statistics with the same WHERE clause are specified for a report, it is recommended that the WHERE condition be defined as a Boolean extension and referenced in each WHERE clause.

CAUTION

Use care when specifying and using statistics with a WHERE clause. Under certain conditions, the results obtained from a statistic are invalid.

For example, the statistical function MAXIMUM returns a low value if used with a WHERE clause in which the Boolean expression is never true, because the actual value returned is based on the size of the data field used in the statistical function and on whether or not the data field is signed. A data field defined as 9999.99, for instance, has a low value of 0, while a data field defined as S99 has a low value of -99.

In addition, moving the low value result of a statistical item from a signed data field to an unsigned data field produces invalid results because the sign is lost. Moreover, if digits are inadvertently truncated in the moving process, additional errors can occur.

To guarantee correct results, do not assign internal attributes to the data field receiving a statistical value.
REPORTER III automatically maintains both the sign and the size of a data field throughout a report.

I Take this path to specify both the scope of the statistic and a condition on which to base the statistic.

Example: ("VOCEMP")

```
TOTAL(SALARY FOR EACH DEPT-NO
   WHERE EMPYE-AGE > 30).
```

This statistic specifies the total salary for each department of all employees whose age is greater than 30 years.

J Take this path after the scope and/or condition is specified.

SUMMARIZE STATEMENT

The SUMMARIZE statement is used to specify statistical summaries and column footings which are included at the end of specified control-break groupings and/or at the end of the entire report (final summaries). The SUMMARIZE statement is omitted if no such statistical summaries and no column footings are desired.

Example: ("INVENT")

```
REPORT PART-NO, INVENT-DOLLAR-VALUE.
          .
          .
          .
SUMMARIZE FOOTING INVENT-DOLLAR-VALUE INCLUDING
    AVERAGE(INVENT-DOLLAR-VALUE).
```

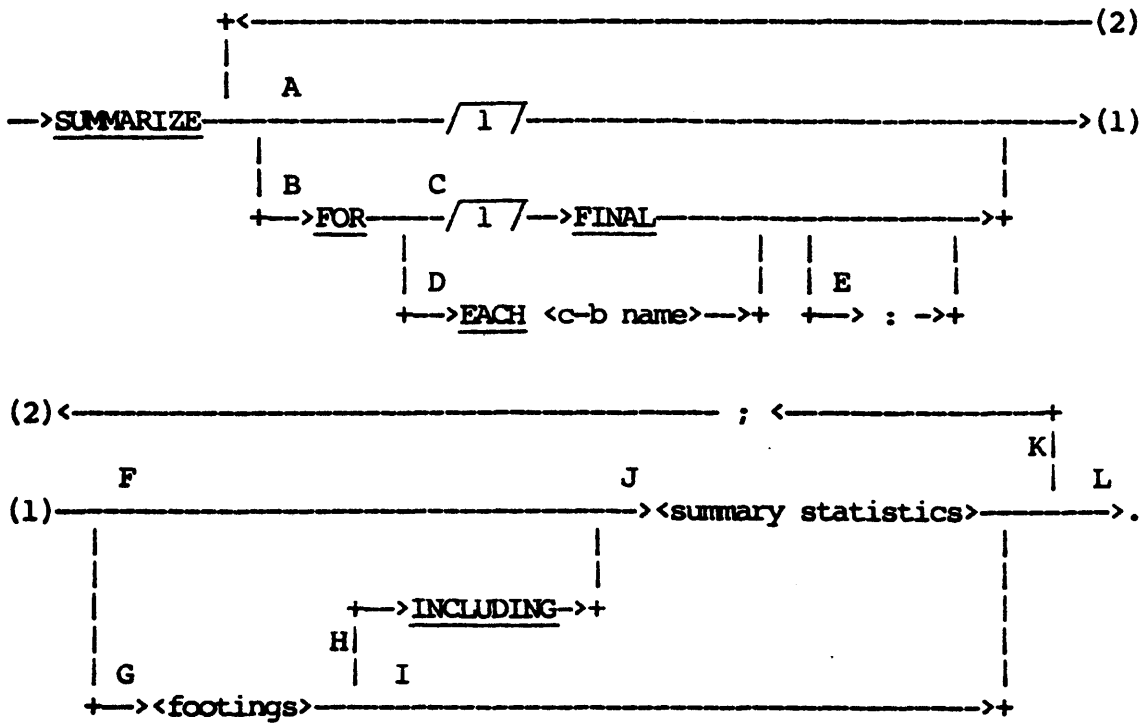
In this example, it is assumed that two columns are reported, PART-NO and INVENT-DOLLAR-VALUE. No control breaks are given, so the report appears as follows:

PART NO	INVENT DOLLAR VALUE
113100	131.00
203171	701.63
311711	333.80
.	.
.	.
.	.

```
SUMMARIES FOR FINAL
TOTAL                              89131.50
AVERAGE (INVENT DOLLAR VALUE): 521.65
```

Notice that the column for inventory dollar value is footed with the total of all values appearing in the column. In addition, the average inventory dollar value of all assets reported is obtained.

The syntax for the SUMMARIZE statement is as follows:



The paths of this syntax diagram are explained below:

Path	Explanation
A	Take this path to specify conveniently that footings be printed at the end of the report and for all previously specified control-break groupings. If <summary statistics> are specified, they are printed only at the end of the report as FINAL summaries (refer to path C).

Example: ("INVENT")

```
REPORT BY DEPARTMENT LISTING PART-NO,
   INVENT-DOLLAR-VALUE.
        .
        .
        .
SUMMARIZE FOOTING INVENT-DOLLAR-VALUE.
```

The column for INVENTORY-DOLLAR-VALUE is footed for each DEPARTMENT grouping and at the end of the report. The above SUMMARIZE statement is equivalent to the statement that follows (refer to paths C and D).

```
SUMMARIZE FOR EACH DEPARTMENT FOOTING
   INVENT-DOLLAR-VALUE;
      FOR FINAL FOOTING INVENT-DOLLAR-VALUE.
```

Each SUMMARIZE statement generates a report that appears as follows:

```
DEPARTMENT:  1031

                  PART              INVENT
                    NO              DOLLAR
                                     VALUE

                023111              170.20
                178110               83.50
                   .                   .
                   .                   .
                   .                   .
SUMMARIES FOR DEPARTMENT:  1031
TOTAL                             83321.13


DEPARTMENT:  1120

                  PART              INVENT
                    NO              DOLLAR
                                     VALUE

                071310               13.10
                500300              510.00
                   .                   .
                   .                   .
                   .                   .

SUMMARIES FOR DEPARTMENT:  1120
TOTAL                              7100.20

                   .                   .
                   .                   .
                   .                   .

SUMMARIES FOR FINAL
TOTAL                            317121.80
```

Example: ("VOCAST")

```
REPORT BY DEPT-NO; BY LOC-CODE LISTING
   ASSET-NO, COST.
SUMMARIZE AVERAGE(COST).
```

The average asset cost of all assets is calculated and printed as a final summary. The above SUMMARIZE statement is equivalent to the following statement (refer to path C):

```
SUMMARIZE FOR FINAL: AVERAGE(COST).
```

As shown in the preceding example, path A defines a default scope of FINAL for the specified statistical functions. The scope of a statistical function is defined as the group of information which is to be summarized.

If path A is taken and columns consist of summary information for a particular control break, specified footings only appear in the report for all higher-level breaks and FINAL.

Example: ("INVENT")

```
REPORT FOR EACH DEPARTMENT: COUNT,INVENTORY-VALUE
   WHICH IS TOTAL(INVENT-DOLLAR-VALUE).
SUMMARIZE FOOTING INVENTORY-VALUE.
```

The INVENTORY-VALUE column is footed only at the end of the report.

B Take this path to specify that summary statistics and/or footings be calculated and printed at the end of the report as final summaries or for each value of a specified control-break item. The FOR option defines the default scope for the specified statistical functions and footings. (Refer to FOR option for Statistic Function in this section.) The scope of a statistic or footing is defined as the group of information which is to be summarized.

C Take this path to specify summary statistics and footings that summarize all information being reported and appear at the end of the report as SUMMARIES FOR FINAL.

If path C is taken, all items reported as final must be statistical summary items related to all information being reported.

Example: ("VOCAST")

```
SUMMARIZE FOR FINAL AVERAGE(SALARY).
```

The above statement specifies that the average salary of all employees reported is to be calculated. The value appears on the printed reports as follows.

```
SUMMARIES FOR FINAL
AVERAGE(SALARY): $ 17130.87
```

D Take this path to specify summary statistics and/or footings which summarize information related to each value of a previously defined control-break item. Path D can only be specified once for a given <c-b name>. These summaries appear in the report after all specified information pertaining to each control-break value is printed. The specific control-break summaries are identified as follows:

```
SUMMARIES FOR <identifier>:  <c-b value>.
```

The default <identifier> is based on the control-break name. (Refer to DEFAULT IDENTIFIERS under COLUMN DESC in this section.)

If path D is taken, all items reported as summary statistics for the specified control break must be statistical summary items related to the control break.

Example: ("VOCAST")

```
SUMMARIZE FOR EACH DEPT-NO AVERAGE(COST).
```

The above statement specifies that the average cost of all assets reported for each department number is calculated and printed. The values appear for each department in the following format.

```
SUMMARIES FOR DEPT NO:  0134
AVERAGE(COST):  $   123.20
```

Example: ("CLIENT")

```
GROUP BY RANGES OF CREDIT-LIMIT FROM 1000
   TO 1000 AS CREDIT-RANGE.
     .
     .
     .
SUMMARIZE FOR EACH CREDIT-RANGE:
   AVERAGE(BALANCE-DUE).
```

The above SUMMARIZE statement specifies that the average balance due is to be calculated and reported for each value range of credit limit. The average balance due appears for each control-break grouping in the following format.

```
SUMMARIES FOR CREDIT RANGE:   2000 -  3000
AVERAGE (BALANCE DUE):  $ 1115.26
```

E The colon can be used for documentation clarity when no footings are desired and summary statistics follow. The colon has no semantic meaning.

F Take this path if no footings are desired and only summary statistics are specified.

Example: ("TRUCKV")

```
SUMMARIZE TOTAL(AMOUNT-OWED).
```

G Take this path to specify column footings for the report. The use of this path presupposes the specification of columns in the REPORT statement. The values appearing in the specified columns are totaled. The totals are placed under the columns.

H If summary statistics are desired in addition to column footings, this path is taken. The footings are printed underneath the columns, while the summary statistics are printed in a vertical format.

Example: ("INVENT")

```
REPORT BY DEPARTMENT LISTING PART-NO,
   INVENT-DOLLAR-VALUE.
            .
            .
            .
SUMMARIZE FOR EACH DEPARTMENT FOOTING
   INVENT-DOLLAR-VALUE
      INCLUDING AVERAGE(INVENT-DOLLAR-VALUE).
```

This produces the following type of report.

```
DEPARTMENT:  SUB-ASSEMBLY

                    PART              INVENT
                     NO               DOLLAR
                                       VALUE

                   071310              13.10
                   500300             510.00
                     .                   .
                     .                   .
                     .                   .
SUMMARIES FOR DEPARTMENT:  SUB-ASSEMBLY

TOTAL                                 7100.20
AVERAGE(INVENT DOLLAR VALUE):  187.52
```

I Take this path if only footings are desired.

J Take this path to specify summary statistics. Summary statistics are printed in a vertical format and are appropriately identified in the report listing.

Example: ("CLIENT")

```
SUMMARIZE MAXIMUM(CREDIT-LIMIT).
```

This specifies that the maximum credit limit is calculated and printed for reported accounts.

K Take this path as many times as necessary to specify all final and control-break summaries. Any previously defined control break can be mentioned in the SUMMARIZE statement (refer to path D above).

Example: ("CLIENT")

```
REPORT BY INV-BRANCH-NO; BY INV-CUST-NO LISTING
   INVOICE-NO, TOTAL-AMOUNT.
            .
            .
            .
SUMMARIZE FOR EACH INV-CUST-NO
   FOOTING TOTAL-AMOUNT;
   FOR EACH INV-BRANCH-NO: MAXIMUM(TOTAL-AMOUNT);
   FOR FINAL:  AVG(TOTAL-AMOUNT),
               AVG(AMOUNT-PAID).
```

The following report results:

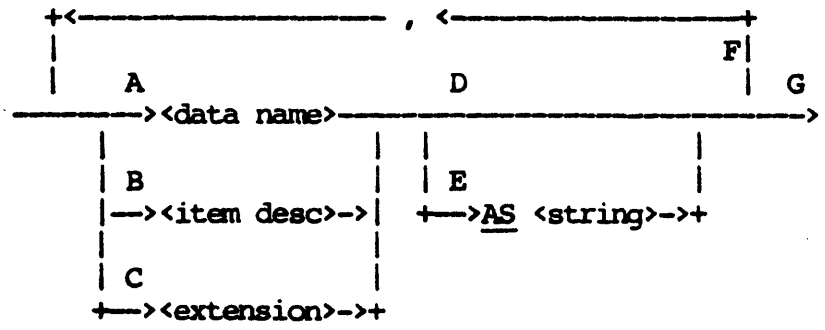
```
INV BRANCH NO:  0012
INV CUST NO:  031748

                INVOICE                 TOTAL
                   NO                  AMOUNT


                033712                $ 131.05
                041350                $  73.03
                   .                     .
                   .                     .
                   .                     .
SUMMARIES FOR INV CUST NO:  031748
TOTAL                                 $2100.20
                   .
                   .
                   .

SUMMARIES FOR INV BRANCH NO:  0012
MAXIMUM (TOTAL AMOUNT):  $237.80
                   .
                   .
                   .

SUMMARIES FOR FINAL
AVG (TOTAL AMOUNT):  $83.50
AVG (AMOUNT PAID):   $12.65
```

L Take this path when all summary information is specified.

SUMMARY STATISTICS

Summary Statistics specifies what summary statistical items are to be included in a report.

The syntax for Summary Statistics is as follows:



The paths of this syntax diagram are explained below:

Path	Explanation

A Take this path to reference a statistical data item to be included in the summary statistics. The <data name> must not have been specified previously as a summary statistic within the SUMMARIZE statement.

Example:

```
TOTAL-COST IS TOTAL(COST FOR EACH DEPT-NO)
   NUM(8,2)
      .
      .
      .
SUMMARIZE FOR EACH DEPT-NO: TOTAL-COST.
```

The total cost of assets for each department is printed as summary information for each department number.

B Take this path to describe a statistical data item to be included in the summary statistics. The item described must be statistical.

Example: ("VOCAST")

```
SUMMARIZE FOR EACH DEPT-NO:
   TOTAL(COST) NUM(8,2)
```

The total cost of assets for each department is printed as summary information for each department number.

C Take this path to define an extension and specify that the derived data item be included in the summary statistics. The derived data item must be statistical.

Example: ("CLIENT")

```
SUMMARIZE FOR EACH BRANCH:

        BAL30 IS TOTAL(THIRTY-DAYS-DUE).
```

The total balance thirty days due (BAL30) is printed as summary information for each branch.

D If this path is taken, a default identifier based on the <item desc> or derived data item name is used in the report to identify the value of the summary statistic (Refer to DEFAULT IDENTIFIERS under COLUMN DESC in this section.)

E Take this path to specify a string of characters to be used to identify the statistical value printed.

The <string> specified here must be 30 characters or less in length.

Example: ("INVENT")

```
TOTAL(QUANTITY) AS "STOCK ON HAND".
```

The total quantity of parts is identified in the report as follows:

```
STOCK ON HAND:  218195
```

F Take this path as many times as necessary to specify all statistical summaries desired. Summaries are listed vertically in the report in the same sequence as they are specified.

Example: ("INVENT")

```
SUMMARIZE FOR EACH DEPARTMENT:
   MAX(INVENT-DOLLAR-VALUE),
   MIN(INVENT-DOLLAR-VALUE),
   TOTAL(INVENT-DOLLAR-VALUE),
   AVG(QUANTITY).
```

Summaries for each department are printed as follows:

```
                .
                .
SUMMARIES FOR DEPARTMENT:  ENGINEERING

MAX (INVENT DOLLAR VALUE):  707.50
MIN (INVENT DOLLAR VALUE):   10.70
TOTAL (INVENT DOLLAR VALUE): 107100.50
AVG (QUANTITY):  20
                .
                .
                .
```

Example: ("CLIENT")

```
SUMMARIZE FOR EACH BRANCH:
    BAL30 IS TOTAL(THIRTY-DAYS-DUE)
        WITH PIC "$ZZZZ9.99",
    BAL60 IS TOTAL(SIXTY-DAYS-DUE),
        WITH PIC "$ZZZZ9.99",
    BAL90 IS TOTAL(NINETY-DAYS-DUE),
        WITH PIC "$ZZZZ9.99",
    BAL30 + BAL60 + BAL90 AS "TOTAL".
```

Summaries for each branch are printed as follows:

```
                .
                .
                .

    SUMMARIES FOR BRANCH:  0012

    BAL30:              $ 1320.80
    BAL60:              $  987.26
    BAL90:              $  220.40
    TOTAL:                2528.46000
                .
                .
                .
```

G Take this path after all summary statistics desired are provided.

SUPPRESS SORT STATEMENT

The SUPPRESS SORT statement suppresses the sort normally required to group and order information in the specified manner. A SUPPRESS SORT should only be specified if you know that the information as input results in logical records grouped by control break and/or ordered in the sequence required and specified in the report specification.

Example: ("CUSTV")

```
INPUT REMIT-FILE.
      .
      .
      .
REPORT BY REMIT-ACCT-NO LISTING CUST-NAME,
   NET-PAYMT.
SUPPRESS SORT.
```

Since the records of REMIT-FILE are maintained in sequence based on REMIT-ACCT-NO, all remittances are already grouped together by account number; therefore, a listing of customers can be obtained by account number without a sort.

Example: ("CLIENT")

```
INPUT ACCT-INVOICE-INFO.
SUPPRESS SORT.
GROUP BY CUST-NO.
      .
      .
      .
```

The input information (the invoices) are already grouped correctly based on CUST-NO by nature of the access specified; therefore, sorting is not necessary.

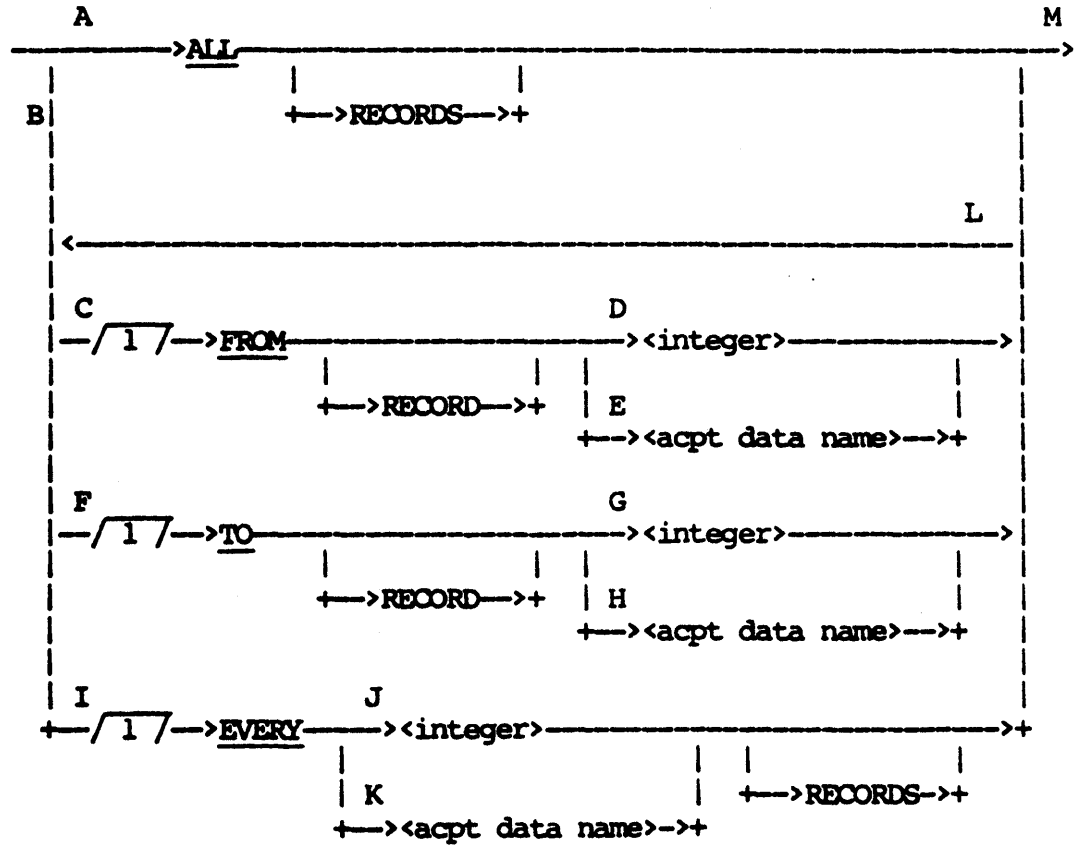
The syntax for the SUPPRESS SORT statement is as follows:

—>SUPPRESS SORT.

SYSTEMATIC-SAMPLE DESC

The Systematic-sample Desc is used to describe a systematic sampling.

The syntax for the Systematic-sample Desc is as follows:



The paths of this syntax diagram are explained below:

Path	Explanation
A	Take this path if all records are to be included in the sample.

Example:

```
SAMPLE ALL RECORDS FROM STRATA WHERE
   BALANCE > 10000.
```

This sample consists of all records with BALANCE greater than 10000.

B Take this path to specify the first record to be sampled, the last record to be considered, and/or the sampling interval.

C Take this path to specify the first record to be sampled. The default is the first record as determined by the sampling interval.

D-E The starting record can be specified through path D using an integer, or through path E using an accepted data name.

Example:

```
SAMPLE FROM RECORD 5.
```

This sample consists of the fifth record and every record after it.

F Take this path to specify the last record to be considered. The default is the last record processed.

G-H The last record can be specified through path G using an integer or through path H using an accepted data name.

Example:

```
ACCEPT START NUM(2).
SAMPLE FROM START TO RECORD 100 EVERY 10.
```

If the value of START is 5, the sample consists of the 5th, 15th, 25th, . . . 85th, and 95th records.

I This path is used to specify the sampling interval. The default interval is 1.

J-K The sample interval can be specified through path J using an integer, or through path K using an accepted data name.

Example:

```
ACCEPT INTERVAL NUM(3).
SAMPLE FROM RECORD 1 EVERY INTERVAL
    FROM STRATA WHERE BALANCE > 10000.
```

Suppose the value of INTERVAL is 10. Then among all the records with BALANCE greater than 10000, the 1st, 11th, 21st, 31st, . . . records are included in the sample.

L Take this path if an additional description for sampling is required.

M Take this path when all desired descriptions for the sample are specified.

SYSTEM-FILE-DATA-STRUCTURE CLAUSE

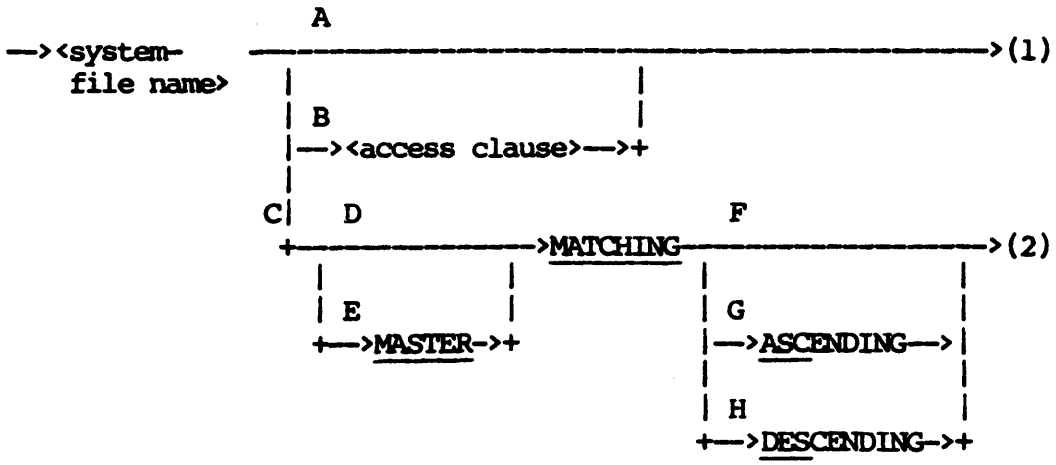
The System-file-data-structure clause allows an ORGANIZATION SEQUENTIAL system file to be accessed sequentially in physical order or randomly via an actual key, or matched with another data structure based on common data fields. The clause allows a system file that is ORGANIZATION INDEXED to be accessed sequentially in ascending key order or randomly via a symbolic record key. The clause also allows a system file that is ORGANIZATION RELATIVE to be accessed sequentially in ascending relative record number order or randomly via a relative record key.

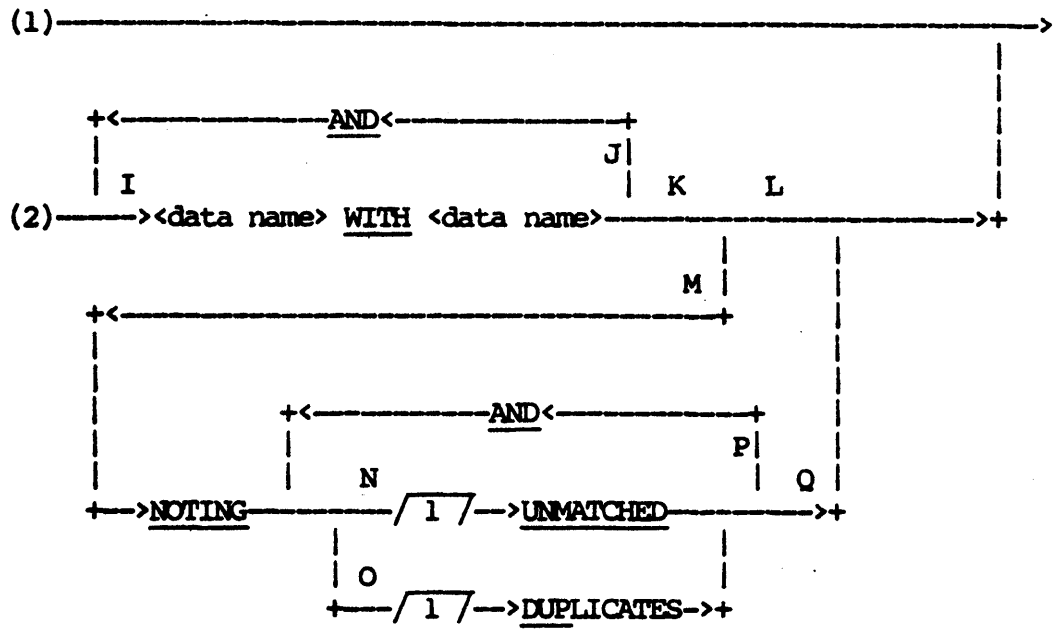
Example: ("CUSTV")

```
INPUT REMIT-FILE.
```

All records in the system file, REMIT-FILE, are accessed sequentially. Each logical record corresponds to a remittance record of the file.

The syntax for the System-file-data-structure clause is as follows:





The paths of this syntax diagram are explained below:

Path	Explanation
A	Take this path if all records of the system file are to be accessed sequentially, not using an ACTUAL KEY, symbolic record key, or matching algorithm.

Example: ("VOCAST")

```
INPUT ASSET-FILE.
```

In this example, the file name ASSET-FILE is accessed sequentially until the file is exhausted.

Example:

```
INPUT PARTS, SUPPLIER.
```

In this example, a record is read from file PARTS and file SUPPLIER in turn, from left to right, until an End-of-File condition is obtained on both files. Thus, a logical record consists of a record for PARTS and a record for SUPPLIER. If one of the files becomes exhausted before the other, all input ceases from this file, and any data items associated with the file are considered to have NULL values.

Example:

```
INPUT ACCT-FILE(INVOICES).
```

All account records are accessed. For each ACCT-FILE record, all INVOICE records are accessed. A logical record contains an account-file record and an invoice record. Each account-file record is linked together with every invoice record in the set of logical records.

B This path allows random access to a specific record given a key value. In addition, it permits sequential access of records starting at a key value which satisfies a given criterion.

Example:

```
INPUT TRANS-FILE, ACCT-FILE AT TRANS-ACCT-NO.
```

In this example, TRANS-ACCT-NO is a data item of TRANS-FILE. With each TRANS-FILE record accessed, the value of TRANS-ACCT-NO is the key value of the ACCT-FILE record to be accessed directly.

C Take this path if records of the system file are to be matched with those of another data structure based on one or more common data items. The system file organization must be sequential. The data structures to be matched are often described as master (or primary) and slave (or secondary). They must be input with their matching key values coming in the same ascending or descending sequence. Either data structure can have duplicates (records with the same matching key values). Usually the master contains no duplicates.

Unmatched records of the specified system file are not input. They can be ignored or noted as exceptions. Each unmatched record of the related data structure is paired with a null record of the system; that is, data items associated with the system file are considered to have null values.

Duplicates (records with duplicate key values) in the specified system file also can be noted as exceptions.

Examples which follow illustrate how records are matched by showing key values within the input records along with the key values within the logical records that result from the match. Duplicate keys are distinguished using subscripts, for example, 101-1, 101-2, and so forth.

D Take this path if the system file is to be considered the slave matching file. The master matching data structure must already have been specified at the same or immediate outer input level. In matching at the same level, no record in the master data structure is matched to more than one record in the slave system file (matching is done in a one-to-one manner). In matching between inner and outer levels, duplicates in the slave system file are matched with the first matching record in the related master data structure, and any duplicate master records after the first have no matched slaves (matching is done in a one-to-many manner).

Example: ("CUSTV")

```
INPUT ACCT-FILE, REMIT-FILE MATCHING
   REMIT-ACCT-NO WITH ACCT-NO.
```

The input records before matching, the resulting logical records after matching, the unmatched REMIT-FILE records, and the duplicate REMIT-FILE records appear as follows:

Before Matching		After Matching		REMIT-FILE Records	
ACCT-FILE	REMIT-FILE	ACCT-FILE	REMIT-FILE	Unmatched	Duplicate
101	101-1	101	101-1	320-1	101-2
213	101-2	213	213	320-2	320-2
417-1	213	417-1	417-1		417-2
417-2	320-1	417-2	null		417-3
580	320-2	580	null		
	417-1				
	417-2				
	417-3				

Example: ("CUSTV")

```
INPUT REMIT-FILE, ACCT-FILE MATCHING
   ACCT-NO WITH REMIT-ACCT-NO.
```

In the following example, REMIT-FILE is considered the master and ACCT-FILE the slave.

Before Matching		After Matching		ACCT-FILE Records	
REMIT-FILE	ACCT-FILE	REMIT-FILE	ACCT-FILE	Unmatched	Duplicate
101-1	101	101-1	101	580	417-2
101-2	213	101-2	null		
213	417-1	213	213		
320-1	417-2	320-1	null		
320-2	580	320-2	null		
417-1		417-1	417-1		
417-2		417-2	null		
417-3		417-3	null		

Example: ("CUSTV")

```
INPUT ACCT-FILE(REMIT-FILE MATCHING
   REMIT-ACCT-NO WITH ACCT-NO).
```

This example illustrates the type of matching which is probably the most common. Unmatched master records are accepted as input to the report. Unmatched slave records are not input but can be noted as an exception (refer to path option M). The input records before matching, the resulting logical records, the unmatched records, and the duplicate records appear as follows:

Before Matching		After Matching		REMIT-FILE Records	
ACCT-FILE	REMIT-FILE	ACCT-FILE	REMIT-FILE	Unmatched	Duplicate
101	101-1	101	101-1	320-1	101-2
213	101-2	101	101-2	320-2	320-2
417-1	213	213	213		417-2
417-2	320-1	417-1	417-1		417-3
580	320-2	417-1	417-2		
	417-1	417-1	417-3		
	417-2	417-2	null		
	417-3	580	null		

Example: ("CUSTV")

```
INPUT REMIT-FILE(ACCT-FILE MATCHING
   ACCT-NO WITH REMIT-ACCT-NO).
```

In the following example, the ACCT-FILE is treated as the slave. The records appear as follows.

Before Matching		After Matching		ACCT-FILE Records	
REMIT-FILE	ACCT-FILE	REMIT-FILE	ACCT-FILE	Unmatched	Duplicate
101-1	101	101-1	101	580	417-2
101-2	213	101-2	null		
213	417-1	213	213		
320-1	417-2	320-1	null		
320-2	580	320-2	null		
417-1		417-1	417-1		
417-2		417-1	417-2		
417-3		417-2	null		
		417-3	null		

E Take this path if the system file is to be considered the master matching file. The related slave data structure must have been specified previously at the same input level. Duplicate records in the slave data structure are matched with the first matching record in the master system file. Duplicate master records, other than the first, are unmatched.

Example: ("CUSTV")

```
INPUT REMIT-FILE, ACCT-FILE MASTER MATCHING
   ACCT-NO WITH REMIT-ACCT-NO.
```

This example illustrates a common type of matching which is used when you want to accept any unmatched slave records as input to the report. Unmatched master records can be ignored or treated as an exception (refer to path option M). Duplicate master records also can be noted as exceptions.

The input records before matching, the logical records after matching, the unmatched records, and the duplicate records appear as follows:

Before Matching		After Matching		ACCT-FILE Records	
REMIT-FILE	ACCT-FILE	REMIT-FILE	ACCT-FILE	Unmatched	Duplicate
101-1	101	101-1	101	417-2	417-2
101-2	213	101-2	101	580	
213	417-1	213	213		
320-1	417-2	320-1	null		
320-2	580	320-2	null		
417-1		417-1	417-1		
417-2		417-2	417-1		
417-3		417-3	417-1		

Example: ("CUSTV")

```
INPUT ACCT-FILE, REMIT-FILE MASTER MATCHING
   REMIT-ACCT-NO WITH ACCT NO.
```

In this example, the REMIT-FILE is treated as the master. The records appear as follows:

Before Matching REMIT-FILE		After Matching		REMIT-FILE Records	
ACCT-FILE	REMIT-FILE	ACCT-FILE	REMIT-FILE	Unmatched	Duplicate
101	101-1	101	101-1	101-2	101-2
213	101-2	213	213	320-1	320-2
417-1	213	417-1	417-1	320-2	417-2
417-2	320-1	417-2	417-1	417-2	417-3
580	320-2	580	null	417-3	
	417-1				
	417-2				
	417-3				

F-H You can specify the sequential ordering of the matching key values through path F (ascending by default), path G (ascending), or path H (descending).

If the key values are not in the specified sequential ordering, the generated program discontinues processing with an exception condition.

I Take this path to specify matching keys. The first <data name> must be the name of a <data name> in the specified system file, and the second <data name> must be the name of a data item in the related data structure.

Both <data name>s must be qualified to the full extent shown in the vocabulary report.

J You can specify additional matching keys by taking this path. All keys specified must match in order for the records to be considered matched. Records in the file must be ordered such that the first matching key specified is the major ordering key and subsequent matching keys specified are minor ordering keys.

Example:

```
INPUT ACCT-FILE (TRANS-FILE MATCHING
   TRANS-ACCT-NO WITH ACCT-NO AND
      TRANS-BANK-NO WITH ACCT-BANK-NO).
```

In this example, TRANS-ACCT-NO and TRANS-BANK-NO are data items in TRANS-FILE. ACCT-NO and ACCT-BANK-NO are data items in ACCT-FILE.

K Take this path if no more matching keys are to be specified.

L Take this path if unmatched records and duplicate records in the system file are to be ignored.

M Take this path if unmatched records and/or duplicate records in the specified system file are to be noted as exceptions. Pertinent information about the records, such as the values of their actual keys and given matching keys, are noted on the exceptions listing.

N Take this path if unmatched records in the specified system file are to be noted as exceptions.

O Take this path if duplicate records in the specified system file are to be noted as exceptions.

P Take this path to specify an additional noting option.

Example: ("CUSTV")

```
INPUT ACCT-FILE, REMIT-FILE MATCHING
   REMIT-ACCT-NO WITH ACCT-NO NOTING
      UNMATCHED AND DUPLICATES.
```

Q Take this path when all noting options are specified.

TABLE STATEMENT

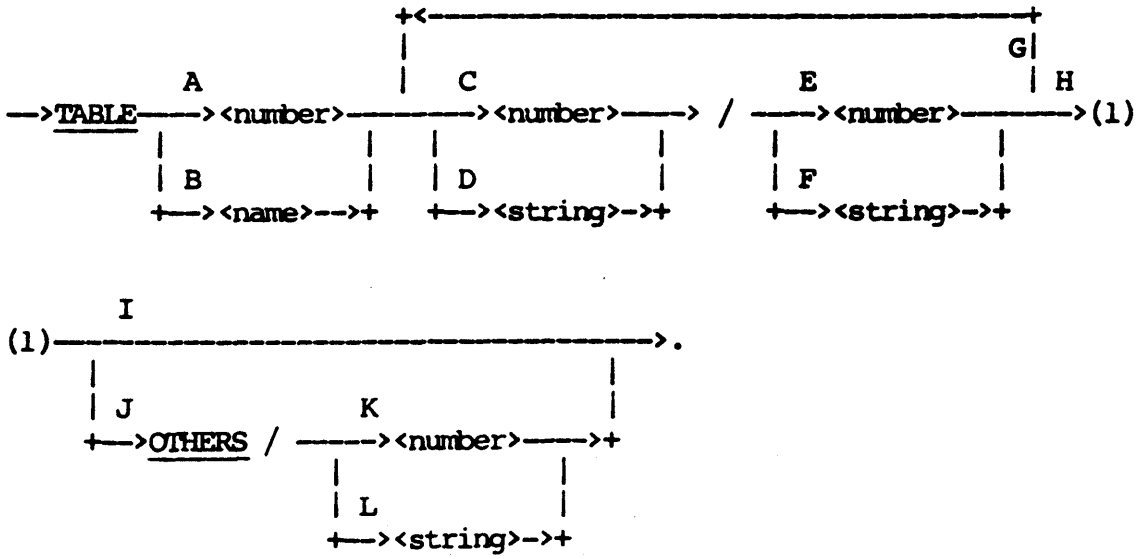
The TABLE statement is used to create a conversion table which relates certain data values to equivalent or corresponding values. Once a table is defined, an <ENTRY function> can be used to obtain the corresponding table-defined values for particular values of a referenced data item.

Example:

```
TABLE 03
    "N" / "NEW YORK"
    "T" / "TULSA"
    "D" / "DETROIT"
    "S" / "SALT LAKE CITY".
```

This table can be referenced in an <ENTRY function> to obtain from the coded values of particular items "N", "T", "D", and "S", the equivalent values: "NEW YORK", "TULSA", "DETROIT", and "SALT LAKE CITY", respectively.

The syntax for the TABLE statement is as follows:



The paths of this syntax diagram are explained below:

Path	Explanation

A A <number> can be used to identify the table by taking this path. This <number> would then be specified to reference the table in an <ENTRY function>. The <number> can contain up to 10 digits without a decimal point, or nine digits with a decimal point.

To use the table defined in the previous example to convert a city to a city name, the following <ENTRY function> could be given:

Example:

```
ENTRY (IN TABLE 03 FOR LOC-CODE)
```

B A <name> can be used to identify the table. The <name> could then be specified by referencing this table in an <ENTRY function>. The <name> can be up to 10 characters in length.

Example:

```
TABLE CITY-CODES

 "N" / "NEW YORK"
 "T" / "TULSA"
 "D" / "DETROIT"
 "S" / "SALT LAKE CITY".
```

To use this table to convert a city code to the actual name of the city, the <ENTRY function> would reference the table as follows:

```
ENTRY (IN TABLE CITY-CODES FOR LOC-CODE)
```

C Take this path if the values to be converted are numeric. The <number> represents one possible value for the data item referenced in an <ENTRY function>.

Example:

```
TABLE 04
    1 / "MALE"
    2 / "FEMALE".
```

If SEX references a numeric data item which has a value of 1 in a particular record, reference to table 04 and SEX in the <ENTRY function> returns the value of "MALE" for this record.

D Take this path if the values to be converted are of type <string>. The <string> represents one possible value for the data item referenced in an <ENTRY function>.

Example:

```
TABLE 04
    "M" / "MALE"
    "F" / "FEMALE".
```

If SEX references a string data item which has a value of "M" in a particular record, reference to SEX in the <ENTRY function> returns the value "MALE" for this record.

E Take this path if the data value given in path C or D is to be converted to a numeric value. The <number> provides one possible value for an <ENTRY function> which references the table.

Example:

```
TABLE 1
    "0" / 0
    "1" / 1
    "2" / 2
        .
        .
        .
    "9" / 9.
```

If the string item referenced in an <ENTRY function> has a value "0", the function returns the numeric value 0.

F Take this path if the data value in path C or D is to be converted to a string value. The <string> provides one possible value for an <ENTRY function> which references the table (refer to the example for path D).

G By taking this path, an additional table entry is specified. Any number of entries can be given. The types of values, string or numeric, that are specified on the left or right side of the slash must be consistent; that is, once path C, D, E, or F is taken, that same path must be taken for every subsequent table entry specified.

H Take this path when all table entries are specified.

I If this path is taken and the value of a data item referenced in an <ENTRY function> does not have a matching value in the conversion table, the function returns a null value.

Example:

```
TABLE 04
    1 / "MALE"
    2 / "FEMALE".
```

If a data item specified in an <ENTRY function> referencing table 04 has a value other than 1 or 2, the function returns a null value. (The default null value for string items is spaces.)

J If this path is taken and the value of a data item referenced in an <ENTRY function> does not have a matching value in the conversion table, the function returns the value corresponding to the "OTHERS" entry.

Example:

```
 TABLE A
     0 / "F"
     1 / "T"
OTHERS / "?".
```

The character "?" is returned if a data item referenced in an <ENTRY function> using table A has a value other than 0 or 1.

K Take this path to specify a numeric value for the OTHERS entry. This path must be taken if path E was taken previously.

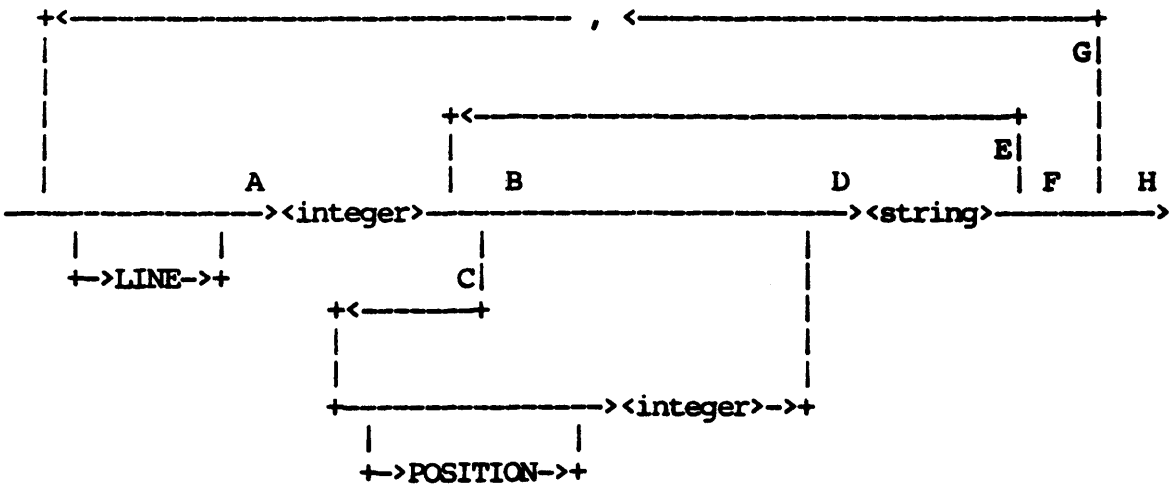
L Take this path to specify a string value for the OTHERS entry. This path must be taken if path F was taken previously.

TEXT

In a WITH TEXT specification (within a PRINT statement), Text specifies standard form text which is to be placed on all forms except overflow forms. Standard form text can be overwritten (purposely or inadvertently) by subsequent Print Specifications.

In a WITH PATTERN specification (within a PRINT statement), Text defines a dummy-form image called a form pattern, which is printed out a specified number of times before actual form printing. The SET FORM-PATTERNS specification (within a Report-option SET statement) specifies the number of times a form pattern is to be printed. This assists in aligning the forms on the printer.

The syntax for Text is as follows:



The paths of this syntax diagram are explained below:

Path	Explanation
A	<Integer> is the line number (1 to FORM-LENGTH) on which the subsequently given text is to appear. The text is specified as one or more <string>s. The next available print position at this point is position one of the specified line.

B Take this path to specify that the <string> is to start at the next available print position.

C Take this path to specify the position on the current line on which the <string> is to be placed. <Integer> cannot be less than the next available print position. If <integer> is greater than the next available print position, an appropriate number of blanks are generated to achieve the desired spacing.

D <String> supplies all or a portion of the text for a particular line. The length of the string cannot exceed FORM-WIDTH, STRING-SIZE, or 55 characters.

E Take this path to specify additional <string>s that add to text for a particular line. The next available print position is now defined as the first character position past the end of the <string>. Therefore a <string> <string> specification results in the two strings being linked together. The total number of text characters given for a particular line cannot exceed FORM-WIDTH.

F Take this path when you have specified all text for a particular line.

G Take this path to specify text for another line. You must give the line numbers in ascending sequence.

H Take this path after you have specified all standard text. See Figure 4-5 for an example of Text specification.

TITLE STATEMENT

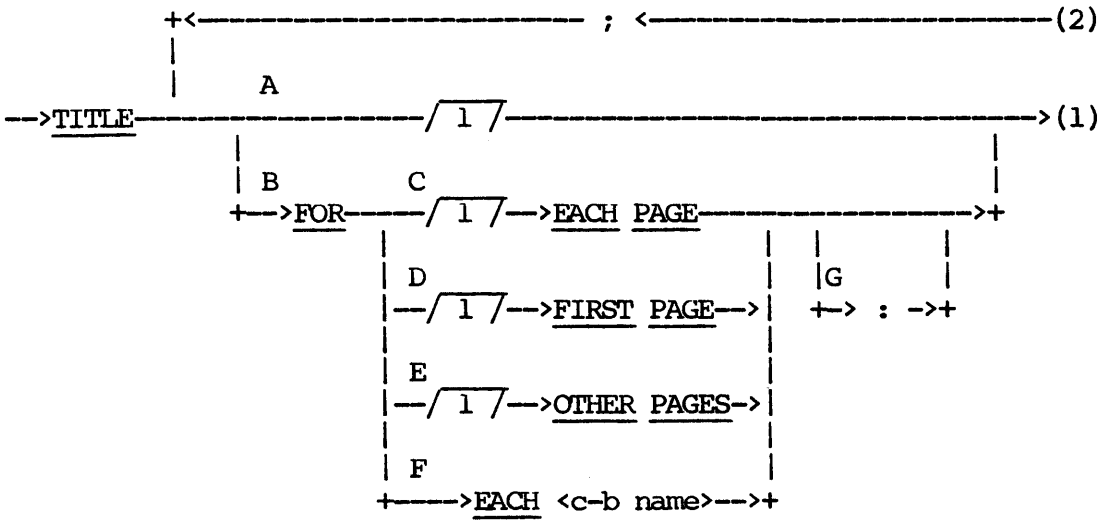
The TITLE statement specifies page title information for reports generated by the REPORT statement. This title information is included in the job summary report. Titles can consist of character strings, time of report, date of report, and data items.

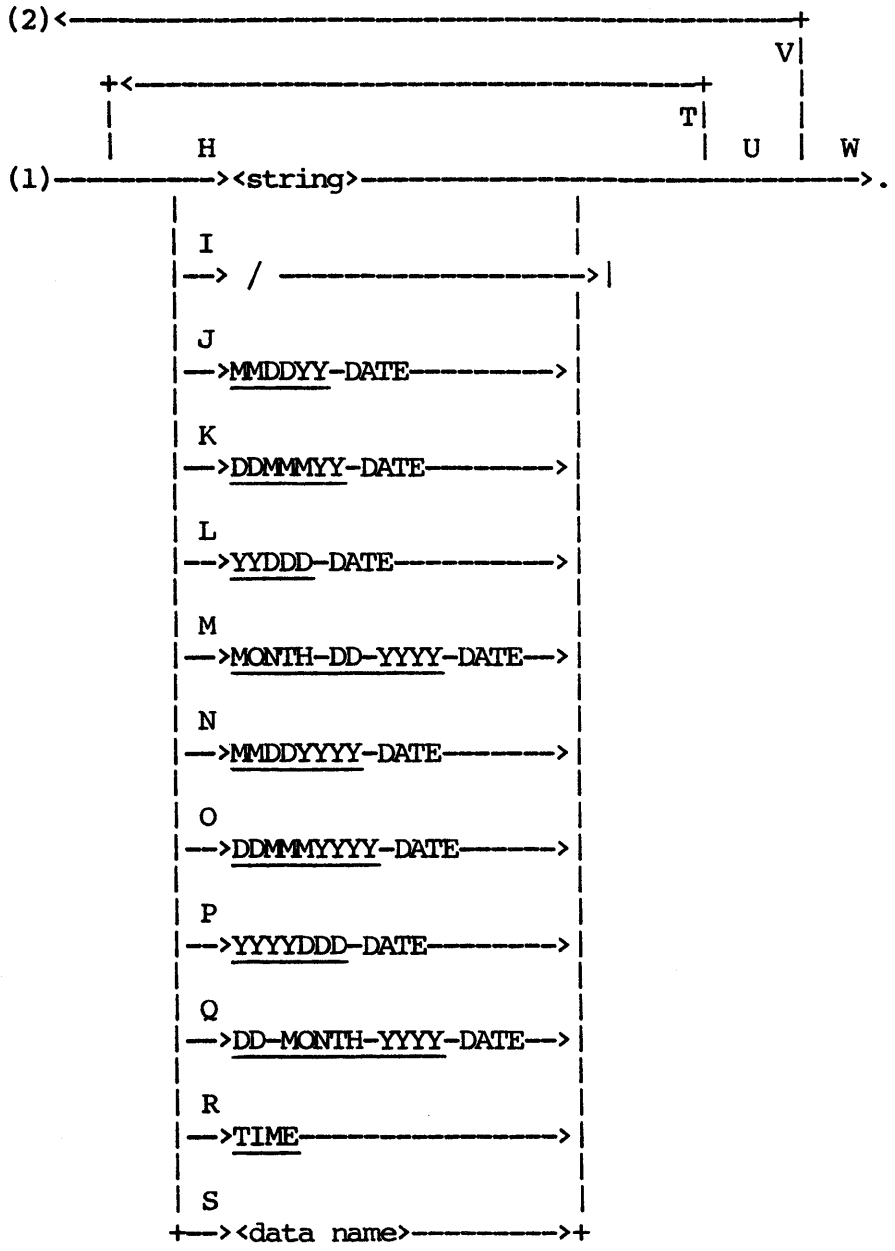
Example:

```
TITLE "SAMPLE RESULTS -- STRATA 1".
```

The report is titled "SAMPLE RESULTS -- STRATA 1". The title is centered on the top of every page of the report.

The syntax for the TITLE statement is as follows:





The paths of this syntax diagram are explained below:

Path	Explanation
A	Take this path if the same title is to be printed on each and every page. This path is identical in function to the path C option. Once this path is used, subsequent use of paths C, D, and E is illegal.

B Take this path to:

1. Specify explicitly that the title is to appear on each page (as in path A).

2. Specify a title for the first page of a report.

3. Specify a title for all other pages but the first.

4. Specify a title to appear for each new control-break value.

C The EACH PAGE option is identical in function to the default option described in path A above. Once path C is used, subsequent use of paths A, D, and E is illegal.

Example:

```
TITLE FOR EACH PAGE: "SAMPLE RESULTS".
```

The string "SAMPLE RESULTS" appears centered at the top of every page in the report.

D Take this path to specify a title which appears only on the first page of a report. Once this path is used, subsequent use of paths A and C is illegal.

Example:

```
TITLE FOR FIRST PAGE "PARTS INVENTORY REPORT".
```

The title "PARTS INVENTORY REPORT" appears centered at the top of the first page of the report only.

E Take this path to specify a title which appears on all pages of a report but the first. This option can be used in conjunction with the FIRST PAGE option described in path D above (refer to option V). Once path E is used, subsequent use of paths A and C is illegal.

Example:

```
TITLE FOR OTHER PAGES "PARTS (CONTD)".
```

The title "PARTS (CONTD)" appears centered at the top of all pages of the report except the first.

F Take this path to specify that information for a specific control-break grouping begin on a new page with an appropriate title. The <c-b name> must be a <data name> which was previously declared as a control-break name. This title is printed in addition to any other <c-b name>, EACH PAGE, OTHER PAGE, or FIRST PAGE title.

Example: ("INVENT")

```
TITLE FOR EACH DEPART-NO "DEPARTMENT SALES".
```

Information pertaining to a particular DEPARTMENT begins on a new page and is titled "DEPARTMENT SALES".

G The colon is optional and is used only for clarity.

H Take this path to specify a character string within the title. The string can be up to 55 characters in length. (More than one string can be used to generate titles longer than 55 characters. Refer to path T.)

Example:

```
TITLE "ACCOUNTS OVERDRAWN".
```

A title composed of the character string "ACCOUNTS OVERDRAWN" is centered on each page.

I A slash is used to indicate that subsequent title information begins on a new line.

Example:

```
TITLE "REPORT ON"/"PASSBOOK"/"ACCOUNTS".
```

The following title appears at the top of each page:

```
REPORT ON
PASSBOOK
ACCOUNTS
```

NOTE

A title can be composed by specifying a list of title elements (refer to path T).

J-Q These paths are taken to include the date of the report in the title in various formats. The format is determined by the choice of key word. The value for the date is obtained from the operating system.

The following table gives the formats corresponding to the different key words. For illustrative purposes, the assumed date is October 25, 1982.

Path	Key Word	Example
J	MMDDYY-DATE	10/25/82
K	DDMMMYY-DATE	25OCT82
L	YYDDD-DATE	82299
M	MONTH-DD-YYYY-DATE	OCTOBER 25, 1982
N	MMDDYYYY-DATE	10/25/1982
O	DDMMMYYYY-DATE	25OCT1982
P	YYYYDDD-DATE	1982299
Q	DD-MONTH-YYYY-DATE	25 OCTOBER, 1982

Example:

```
TITLE "REPORT FOR" DDMMMYY-DATE.
```

For the date October 25, 1982 this statement produces the title "REPORT FOR 25OCT82" on each page of the report.

In the original version of this guide, the date format keywords attempted to describe the date format. The wordage used did not match the wordage used in the date functions. In the present version of the guide, additional wordage has been added. These keywords describe the exact format of the date. Although not shown here, the old format keywords are still valid.

R Take this path to include the time the report is run in the title format HH:MM:SS (hours, minutes, seconds). The time is obtained from the operating system.

Example:

```
TITLE "ACTIVITY REPORT" TIME.
```

This produces the title "ACTIVITY REPORT 10:11:52" on each page (assuming the time is 10:11:52 AM).

S Take this path to specify that the value of a data item be included in a title. This capability facilitates the inclusion of accepted data into the title. In addition, it allows information from the first logical record reported on a page to be included in the titles appearing at the top of the page. No check is made to ensure that the information reported in a page title is applicable to all information reported on the page.

Example:

```
ACCEPT USER-NAME STRING(10).
    .
    .
    .
TITLE "REPORT ON" USER-NAME.
```

Assuming that the value accepted for USER-NAME is "JOHN DOE", the TITLE statement produces the title "REPORT ON JOHN DOE" on each page.

Example: ("INVENT")

```
REPORT PART-NO ASCENDING, PART-DESC, DEPARTMENT,
   QUANTITY, INVENT-DOLLAR-VALUE.
    .
    .
    .
TITLE FOR EACH PAGE:  "LISTING OF PARTS FROM NO." PART-NO.
```

The title printed on each page identifies the part number of the first part listed on the page.

T Take this path as many times as necessary to specify additional components which make up the title. Each component is separated by one blank when the title is printed.

Example:

```
TITLE "REPORT"   "ON"  MMDDYYYY-DATE   "SALES" / TIME.
```

For 10:11:52 AM on October 25, 1982, this statement produces on each page:

```
REPORT ON 10/25/82 SALES
       10:11:52
```

U Take this path after the last title component is specified.

V Take this path to specify additional titles by choosing options specified by paths A and B.

Example:

```
TITLE FOR EACH PAGE: "REPORT OF TRANSACTIONS";
      FOR EACH ACCT-NO:  "ACCOUNT AS OF" MONTH-DD-YYYY.
```

This statement produces the titles:

```
    REPORT OF TRANSACTIONS
ACCOUNT AS OF OCTOBER 25, 1982
```

on pages where ACCT-NO changes value. On pages where ACCT-NO does not change value, only the following appears on the top of the page:

```
    REPORT OF TRANSACTIONS
```

Example:

```
TITLE FOR FIRST PAGE: "Sample Results";
      FOR OTHER PAGES: "Results Continued".
```

This produces the title "Sample Results" on the first page only. All other pages except the first have the title "Results Continued". Note that your printer must be capable of printing lower case letters in order for this to work correctly.

W Take this path when all desired titles are specified.

TOTAL-POPULATION

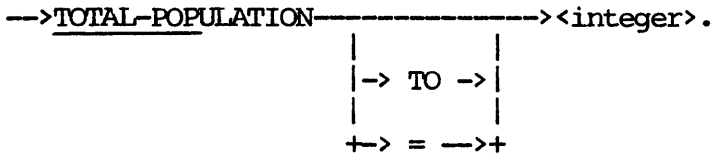
The TOTAL-POPULATION clause is used to establish an estimate of the number of logical records which are to be selected and/or sampled initially for reporting. Its purpose is to determine how much space to allow for the printing of statistical summaries, such as TOTAL and VARIANCE. That is, it is used to set default editing attributes for certain statistical items. If you do not specify this option, the TOTAL-POPULATION is set to the maximum TOTAL-POPULATION associated in RP3VOC with the data structures specified in the INPUT statement. TOTAL-POPULATION should not be confused with the meaning of the POPULATION used in DMS II.

Example:

```
SET TOTAL-POP TO 100.
```

This statement specifies that up to 100 logical records are expected to be selected and/or sampled initially.

The syntax for the TOTAL-POPULATION clause is as follows:



VOCABULARY STATEMENT

The VOCABULARY statement is required to identify the vocabulary files to be used for the desired report. This statement is always required in a REPORTER III report specification. The vocabulary files must be created by the RP3VOC Program.

In the VOCABULARY statement, the <external file name> is used to specify the name of the vocabulary. You must specify the <external file name> exactly as it appears in the second line of the title for the vocabulary listing. This name is the same as the first <external file name> specified in the VOCABULARY statement in RP3VOC.

Example: ("INVENT")

```
    VOCABULARY "INVENT".
```

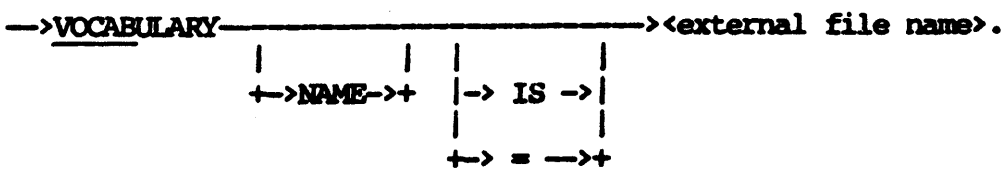
The vocabulary "INVENT" is used for the desired report.

Example: ("VOCEMP")

```
    VOCAB NAME IS "VOCEMP".
```

The vocabulary "VOCEMP" is used for the desired report.

The syntax for the VOCABULARY statement is as follows:



SECTION 5

EXAMPLES OF REPORTS

This section presents examples which demonstrate the use of REPORTER III in a number of applications. The report specifications in these examples use the vocabularies given in Section 2. The kinds of report language statements contained in the examples are explained in detail in Section 4.

EXAMPLE 1

A verification of inventory extensions and a total of any exceptions are required. The following report specification is written using the vocabulary "INVENT".

```
VOCABULARY IS "INVENT".
INPUT PARTMF.
EXTENDED-COST IS QUANTITY * UNIT-COST WITH
   PIC "Z(7)9.99".
SELECT INVENT-DOLLAR-VALUE NOT EQUAL EXTENDED-COST.
REPORT PART-NO, PART-DESC, DEPARTMENT, QUANTITY,
   UNIT-COST, INVENT-DOLLAR-VALUE, EXTENDED-COST.
SUMMARIZE FOOTING INVENT-DOLLAR-VALUE, EXTENDED-COST.
TITLE FOR FIRST PAGE "INVENTORY RECORDS IN ERROR".
```

The report obtained from this report specification is shown in Figure 5-1.

PAGE 1

INVENTORY RECORDS IN ERROR

PART NO	PART DESC	DEPARTMENT	QUANTITY	UNIT COST	INVENT DOLLAR VALUE	EXTENDED COST
54851	FENDER SKIRT	0310	04000	1.15	4050.00	4600.00
59160	QTR. PANEL MOLDING	0400	02013	0.80	1813.13	1610.40
50311	L. PARK. LIGHT LENS	0310	04000	1.15	200.00	234.30
		.				
		.				
		.				
SUMMARIES FOR FINAL:						
TOTAL					6063.83	6444.70

Figure 5-1. Report for Example 1

EXAMPLE 2

Using a client's file of sales for the year, an analysis of inventory is required to compute inventory turnover and identify parts on hand that have had no sales for the year. The following report specification is written using the vocabulary "INVENT".

```
VOCABULARY IS "INVENT".
INPUT PART-INVENT-SALES-INFO.
TITLE FOR FIRST PAGE "SLOW MOVING INVENTORY".
ANNUAL-TURNOVER IS QUANTITY-SOLD/QUANTITY
   NUMBER(5,2).
SELECT ANNUAL-TURNOVER LESS THAN 4.00.
REPORT DEPARTMENT AS "DEPT NO", PART-NO, QUANTITY,
   INVENT-DOLLAR-VALUE, QUANTITY-SOLD
      AS "Y.T.D.SALES", MONTH-SOLD
      YEAR-SOLD, ANNUAL-TURNOVER,
      "NO Y.T.D. SALES" AS
      "COMMENTS" WHERE QUANTITY-SOLD = 0.
```

PART-INVENT-SALES-INFO is a permanent macro in the referenced vocabulary. The macro text specifies access to inventory records and, for each inventory record, access to a related sales record. The report obtained from this report specification is shown in Figure 5-2.

PAGE 1

SLOW MOVING INVENTORY

DEPT NO	PART NO	QUANTITY	INVENT DOLLAR VALUE	Y.T.D. SALES	MONTH SOLD	YEAR SOLD	ANNUAL TURNOVER	COMMENTS
0021	412131	09000	1188.90	27270	4	82	3.03	
0031	428000	11300	3813.75	0			0.00	NO Y.T.D. SALES
0040	531002	13500	4884.30	15660	2	82	1.16	
0021	580058	14800	5998.44	0			0.00	NO Y.T.D. SALES
0132	670800	4400	2134.44	0			0.00	NO Y.T.D. SALES
0230	680000	16200	14839.00	57996	7	82	3.58	

Figure 5-2. Report for Example 2

EXAMPLE 3

Inventory items are sampled statistically for subsequent verification of unit costs. Unit costs for the sampled items are compared with appropriate documents (such as purchase invoices) to determine the proper handling of rebates, discounts, duty, freight, and insurance. The corrected unit cost values are then input to obtain a statistical evaluation of the recorded amount.

The following report specification is written to obtain the desired sample. The vocabulary "INVENT" is used.

```
VOCABULARY IS "INVENT".
SAVE EXTRACT-VOCABULARY AS "SAMPLE" AND "SAMP2".
INPUT PARTMF.
SAMPLE EVERY 100.
EXTRACT WITH VOCABULARY PART-NO, DEPARTMENT,
   BIN-NO, UNIT-COST,
   ACTUAL-UNIT-COST IS UNIT-COST NUM(3,2) TO
            FILE CONFIRM; HARDWARE IS DISK.
```

A disk file is created which represents the sampled inventory items, and a vocabulary is automatically created for this file. The listing of the vocabulary is given in Figure 5-3.

```
RP3 N.NN                    V O C A B U L A R Y                    DD MMM YY
                                "SAMPLE"                             PAGE 1

SYSTEM FILES.

FILE CONFIRM,
     TOTAL-POPULATION IS DEFAULTED TO 9999.

LEVEL  NAME WITH          SUBSCRIPTS  TYPE          LENGTH  EDITING
       QUALIFIERS                                          PICTURE

  1    QEXT-RECORD                   GROUP
  2    PART-NO                       ITEM STRING       6  X(6)
  2    DEPARTMENT                    ITEM NUMERIC      4  9(4)
  2    BIN-NO                        ITEM NUMERIC      3  9(3)
  2    UNIT-COST                     ITEM NUMERIC      6  Z(2)9.99
  2    ACTUAL-UNIT-COST              ITEM NUMERIC      6  Z(2)9.9(2)
```

Figure 5-3. Vocabulary for CONFIRM File

After the examination of relevant records, the disk file CONFIRM is updated where necessary to reflect the ACTUAL-UNIT-COST as determined from the audit examination. The following report specification is written to obtain an analysis of the findings.

```
VOCABULARY IS "SAMPLE".
INPUT CONFIRM.
REPORT PART-NO, DEPARTMENT, BIN-NO,
   UNIT-COST, ACTUAL-UNIT-COST,
   DISCREPANCY IS UNIT-COST - ACTUAL-UNIT-COST
   WITH PICTURE "-Z(2)9.99".
SUMMARIZE FOOTING UNIT-COST, ACTUAL-UNIT-COST,
   DISCREPANCY INCLUDING AVERAGE(DISCREPANCY),
      STD-DEV(ACTUAL-UNIT-COST).
TITLE "ANALYSIS OF UNIT-COST VERIFICATION".
```

The report obtained from this report specification is shown in Figure 5-4.

PAGE 1

ANALYSIS OF UNIT-COST VERIFICATION

PART NO	DEPARTMENT	BIN NO	UNIT COST	ACTUAL UNIT COST	DISCREPANCY
59160	0400	113	0.80	0.80	0.00
60300	0420	201	1.12	1.20	- 0.08
61200	0400	512	2.57	2.57	0.00
63777	0118	002	5.12	5.06	0.06
			.		
			.		
			.		
SUMMARIES FOR FINAL					
			715.60	730.08	- 14.48

AVERAGE (DISCREPANCY): - 0.03
STD-DEV (ACTUAL UNIT COST): 0.31

Figure 5-4. Report for Example 3

EXAMPLE 4

Aged balances in an accounts receivable file are to be listed, footed, and crossfooted. Any crossfoot exceptions are to be noted. The following report specification is written using the vocabulary "CLIENT".

```
VOCABULARY IS "CLIENT".
INPUT ACCTS-RECV.
EXT-BAL IS CURRENTLY-DUE
   + THIRTY-DAYS-DUE
   + SIXTY-DAYS-DUE
   + NINETY-DAYS-DUE.
REPLACE BALANCES BY CURRENTLY-DUE, THIRTY-DAYS-DUE,
   SIXTY-DAYS-DUE, NINETY-DAYS-DUE, BALANCE-DUE,
      EXT-BAL #.
REPORT BRANCH, CUST-NO, BALANCES,
   " EXCEPTION" WHERE BALANCE-DUE NOT EQUAL
      TO EXT-BAL.
SUMMARIZE FOOTING BALANCES.
```

The report obtained from this report specification is shown in Figure 5-5.

BRANCH	CUST NO	CURRENTLY DUE	THIRTY DAYS DUE	SIXTY DAYS DUE	NINETY DAYS DUE	BALANCE DUE	EXT BAL	
0110	111231	$ 21.00	$ 81.00	$ 3.00	$ 11.00	$ 116.00	116.00000	
0110	128777	$ 118.20	$ 11.50	$ 0.00	$ 128.70	$ 128.70	129.70000	EXCEPTION
0213	235531	$ 23.77	$ 2.00	$ 40.66	$ 5.00	$ 71.43	71.43000	
1300	100511	$ 11.60	$ 15.70	$ 20.00	$ 0.00	$ 51.30	47.30000	EXCEPTION
1300	222411	$ 60.00	$ 110.00	$ 30.00	$ 6.30	$ 206.30	206.30000	
		.	.	.	.	.	.	
		.	.	.	.	.	.	
		.	.	.	.	.	.	
.								
.								
.								
SUMMARIES FOR FINAL:								
TOTAL		$ 11380.73	$ 6420.00	$ 840.50	$ 120.51			

Figure 5-5. Report for Example 4

EXAMPLE 5

Negative confirmations are required on selected accounts as follows:

1. Confirmations on all accounts over $10,000 and/or over 90 days old.

2. Confirmations on 25 percent of all accounts between $2,000 and $10,000 which are less than 90 days old.

3. Confirmations on 5 percent of all other accounts.

Figure 5-6 shows a Burroughs Mailer Set for negative confirmation utilized by the CPA firm of Adam, Doe, Jones, Smith & Co. This form is to be used for this application. A disk file called "MACROS" (loaded from tape) contains common macros frequently used by the auditors of Adam, Doe, Jones, Smith, & Co. The following macro with comments on its use is included in the file "MACROS".

```
REPLACE NEG-CONF (P1, P2, P3, P4, P5, P6, P7, P8, P9, P10) BY
   (FORM-WIDTH = 70, FORM-LENGTH = 33, FORM-TYPE = SPECIAL):
   NF, [P1] IN 30 AT 5 40, [P2] IN 30 AT 6 40, [P3] IN 30 AT 7 40,
       [P4] IN 10 AT 21 50, [DATE] IN 8 AT 21, 67,
       [P5] AT 23, 11, " - ", [P6], [P7] IN 30 AT 25 11,
       [P8] IN 30 AT 26, 11, [P9] IN 30 AT 27 11,
       [P10] IN 30 AT 28 11#.

%  NEG-CONF MAY BE USED IN A PRINT STATEMENT TO SPECIFY THE
%  PRINTING OF A STANDARD A,D,J,S&CO. NEGATIVE CONFIRMATION
%  MAILER SET.  SPECIFICATION MUST BE AS FOLLOWS:
%
%  PRINT NEG-CONF (<P1>, <P2>, <P3>, <P4>, <P5>, <P6>, <P7>,
%                  <P8>, <P9>, <P10>).
%
%  <P1>, <P2>...<P10> ARE PARAMETERS SUPPLIED AS FOLLOWS:
%
%     <P1>  - A <STRING> WHICH REPRESENTS CLIENT'S NAME
%     <P2>  - A <STRING> WHICH REPRESENTS CLIENT'S 1ST ADDRESS LINE
%     <P3>  - A <STRING> WHICH REPRESENTS CLIENT'S 2ND ADDRESS LINE
%     <P4>  - A <DATA-NAME> WHICH REFERENCES THE ACCOUNT NUMBER
%     <P5>  - A <STRING> WHICH DESCRIBES THE AMOUNT TO BE CONFIRMED
%     <P6>  - A <DATA-NAME> WHICH REFERENCES THE AMOUNT
%     <P7>  - A <DATA-NAME> WHICH REFERENCES THE CUSTOMER NAME
%     <P8>  - A <DATA-NAME> WHICH REFERENCES THE CUSTOMER 1ST
%             ADDRESS LINE
%     <P9>  - A <DATA-NAME> WHICH REFERENCES THE CUSTOMER 2ND
%             ADDRESS LINE
```

```
%      <P10> - A <DATA NAME> WHICH REFERENCES THE CUSTOMER 3RD
%              ADDRESS LINE
```

The following report specification is written to produce the confirmations. The vocabulary "CLIENT" is used.

```
000100 VOCABULARY IS "CLIENT".
000200 COMBINE "MACROS".
000300 INPUT ACCT-CUST-INFO.
000400 SAMPLE ALL RECORDS FROM STRATA STRATA-1
000500    WHERE BALANCE-DUE GTR 10000 OR
000600       NINETY-DAYS-DUE GTR 0,
000700       EVERY 4 RECORDS FROM STRATA STRATA-2
000800    WHERE STRATA-1 = FALSE AND BALANCE-DUE
000900       GTR 2000,
001000       EVERY 20 RECORDS FROM STRATA
001100    WHERE NOT STRATA-1 OR STRATA-2.
001200 PRINT NEG-CONF("ABC SAVINGS & LOAN ASSOC.",
001300    "107 MAIN",
001400    "IRVINE, CA. 92714",
001500    CUST-NO,
001600    "CURRENT BALANCE DUE", BALANCE-DUE,
001700    CUSTOMER-NAME, STREET-ADDRESS, CITY-STATE,
001800    ZIP-CODE).
```

NOTE

A sequence number on the card immediately following the COMBINE statement is required for proper merge.

REMOVE THIS FLAP BEFORE MAILING

AUDITORS CONFIDENTIAL CONFIRMATION

Our auditors, Adams, Doe, Jones, Smith & Co., 718 North Wilson Street, Los Angeles, California 90071, are conducting the regular examination of our accounts. They have selected your account at random for confirmation of the information printed below. While you may have other accounts with our Association, please confirm this particular account only. The information presented is as of the date indicated; any transactions after this date should not be considered for this confirmation. If the information is not in agreement with your records, please explain the differences on the reverse side and return this notice directly to our auditors, not to the Association.

IF CORRECT, NO REPLY IS NECESSARY

DO NOT MAIL TO THE ASSOCIATION

ACCOUNT NUMBER	DATE

Figure 5-6. Negative Confirmation Mailer Set

EXAMPLE 6

A report on accounts receivable in excess of $10,000 is required. Remittances received for these accounts during the previous two months, January and February, are to be summarized for verification with current balances. The following report specification is written using the vocabulary "CUSTV".

```
VOCABULARY IS "CUSTV".
INPUT ACCT-FILE(REMIT-FILE MATCHING
   REMIT-ACCT-NO WITH ACCT-NO).
SELECT BALANCE > 10000.
REPORT BY ACCT-NO INCLUDING % ACCOUNT
   BALANCE, OLD-BALANCE      % INFO
   LISTING
      CUST-NAME WHERE CHANGES AS      % REMIT-
      "CUSTOMER NAME", YR-MO-RECD,    % TANCE
      DAY-RECD, NET-PAYMNT, CHK-NO.   % INFO
SUMMARIZE FOR EACH ACCT-NO FOOTING NET-PAYMENT.
SUPPRESS SORT.
```

The report obtained from this report specification is shown in Figure 5-7.

ACCT NO: 010

BALANCE: $ 13000.50
OLD BALANCE: $ 14000.00

CUSTOMER NAME	YR MO RECD	DAY RECD		NET PAYMNT	CHK NO
JOE M. SMITH	8201	04	$	50.00	88166001
	8201	18	$	550.00	88166002
	8201	15	$	399.50	88166006

SUMMARIES FOR ACCT NO: 010

TOTAL	$	999.50	

ACCT NO: 015

BALANCE: $ 10310.83
OLD BALANCE: $ 11000.00

CUSTOMER NAME	YR MO RECD	DAY RECD		NET PAYMNT	CHK NO
SUSAN C. ROBERTS	8201	05	$	100.00	30115008
	8202	02	$	50.00	30115016
	8202	10	$	20.00	30115018
	8202	20	$	540.00	30115020

SUMMARIES FOR ACCT NO: 015

TOTAL	$	710.00	

Figure 5-7. Report for Example 6

EXAMPLE 7

Receivables are to be aged and listed by month. The number of outstanding receivables and the total dollar value for each month are to be shown. In addition, a specific count and total of the outstanding receivables over $5,000 is required. The following report specification is written using the vocabulary "SHIPV".

```
PASSWORD IS "S2175".
VOCABULARY IS "SHIPV".
SAVE OBJECT AS "AGEREP".
INPUT ACCT-TRANS.
TITLE "ACCOUNTS RECEIVABLES"/ "BY MONTH,"
   /"REPORT --" DATE.
SELECT TRANSACTION-TYPE = 1.  % RECEIVABLES
REPORT BY MONTH-OF-BILLING LISTING
   DAY-OF-BILLING ASCENDING, NUMBER OF BILL,
      ACCOUNT-NO OF BILL, AMOUNT-OWED.
SUMMARIZE FOR EACH MONTH-OF-BILLING:
   COUNT, TOTAL(AMOUNT-OWED);
   FOR FINAL:
      COUNT(WHERE AMOUNT-OWED > 5000),
      TOTAL(AMOUNT-OWED WHERE AMOUNT-OWED > 5000)
      AS "VALUE OF RECEIV. OVER $5000".
```

A program is generated and saved as "AGEREP". Then the program produces the report as shown in Figure 5-8.

```
                    ACCOUNTS RECEIVABLE                          PAGE 1
                         BY MONTH,
                    REPORT -- 04/29/82

MONTH OF BILLING:  01

          DAY          NUMBER          ACCOUNT          AMOUNT
           OF            OF           NO OF            OWED
      BILLING          BILL            BILL

           01      11001700            6700          $211.00
           01      20550010            1666           $76.00
           02      10007755            1667          $500.20
                   .
                   .
                   .

SUMMARIES FOR MONTH OF BILLING:  01
COUNT:  93
TOTAL (AMOUNT OWED):  $18420.17

MONTH OF BILLING:  02

          DAY          NUMBER          ACCOUNT          AMOUNT
           OF            OF           NO OF            OWED
      BILLING          BILL            BILL

           01      87100100            6523           $17.00
           03      81106000            6700          $523.50
           04      12301070            5000          $117.00
                   .
                   .
                   .

SUMMARIES FOR MONTH OF BILLING:  02
COUNT:  78
TOTAL (AMOUNT OWED):  $12782.14
                   .
                   .
                   .

SUMMARIES FOR FINAL
COUNT (WHERE AMOUNT OWED > 5000:  5)
VALUE OF RECEIV. OVER $5000:  $37113.05
```

Figure 5-8. Report for Example 7

EXAMPLE 8

A frequency report is desired which summarizes account balances in the file in increments of $25.00 from a starting point of -$400.00 to a terminating point of $850.00. The following report specification is written using the vocabulary "SHIPV".

```
VOCABULARY IS "SHIPV".
INPUT ACCT-TRANS.
GROUP BY RANGES OF AMOUNT-OWED FROM -400.00 BY
   25.00 TO 850.00 AS RANGE-OF-AMOUNT-OWED.
REPORT FOR EACH RANGE-OF-AMOUNT-OWED:
   COUNT, TOTAL(AMOUNT-OWED).
```

The report obtained from this report specification is shown in Figure 5-9.

RANGE OF AMOUNT OWED			COUNT	TOTAL (AMOUNT OWED)
- $400.00	-	- $375.00	2	- $790.20
- $375.00	-	- $350.00	2	- $723.80
- $350.00	-	- $325.00	3	- $985.68
		.		
		.		
		.		
- $25.00	-	$.00	27	- $271.50
$.00	-	$25.00	151	$1500.61
$25.00	-	$50.00	68	$2143.70
		.		
		.		
		.		
$775.00	-	$800.00	15	$11475.00
$800.00	-	$825.00	20	$16173.86
$825.00	-	$850.00	6	$4976.21

Figure 5-9. Report for Example 8

SECTION 6

SYSTEM OPERATIONS ASSOCIATED WITH REPORT PREPARATION

This section contains information on system operations associated with REPORTER III report preparation on each of the following:

1. A Series of Systems.
2. B 1000 Series of Systems.
3. B 2000/B 3000/B 4000 Series of Systems.

The information is presented separately (under a different main paragraph heading) for each of the three series of systems.

A SERIES OPERATIONS

The following subjects are discussed below with respect to use of the REPORTER III System on a computer in the A Series of Systems:

1. The files which must be present on disk before a report can be prepared.

2. The sequence of statements needed to execute the entire process from input of the report specification to output of the printed report.

3. The types of procedures you can use to execute the report preparation process:

   a. Automatic execution procedure.
   b. User-controlled execution procedure.
   c. Execution using Work Flow Language (WFL).

FILES REQUIRED FOR EXECUTION

The REPORTER III System contains the following four files for specifying and generating a report program. The files are supplied on the REPORTER III System tape and must be present on disk when the report specification is analyzed and a report program is generated.

1. RP3REP -- contains the object code for the Report Language Analysis Program.

2. RP3GRM -- contains the report language syntax descriptions used by RP3REP.

3. RP3GEN -- contains the object code for the Report Program Generator.

In addition to the above files, other files must be present on disk or disk pack, as follows:

1. Vocabulary files generated by RP3VOC describing the files to be accessed by the generated program must be present when RP3REP is run. The names of the two vocabulary files needed are given in the output from the RP3VOC execution as follows:

   FILES NEEDED TO RUN REPORTER III ARE:
   <file name 1> AND <file name 2>.

2. The COBOL74 compiler or COBOL85 compiler must be present to compile the generated program, even when DMS II structures are being accessed.

3. If DMS II files are to be accessed, the program DATABASE/INTERFACE as well as the description file DESCRIPTION/<data-base name> must be on disk or disk pack.

INPUT REQUIRED FOR EXECUTION

The process of REPORTER III report specification analysis, program generation, program compilation, program execution, and report production is automatic. You need input only one simple sequence of statements to execute the entire process from specifications input to printed report output. The input execution deck consists of the following instructions, in the order presented:

1. RUN statement
2. FILE statement(s)
3. DATA statement
4. Report-specification file
5. ?END statement

Descriptions of each of these five instructions follow. If you are using card input, the RUN, FILE, and DATA statements must be preceded by an invalid punch.

RUN Statement

The RUN statement instructs the Master Control Program (MCP) to schedule the RP3REP program for execution of specification parsing. The syntax is as follows:

```
RUN RP3REP
```

FILE Statement(s)

The FILE statement is optional. It allows you to change the <external file name> and/or hardware device for one or more of the files used by RP3REP. Any number of FILE statements can be included. (Refer to the A Series Work Flow Language (WFL) Reference Manual for the syntax of the FILE statement.)

Each file used has two names associated with it:

1. <internal file name>, which is the name by which a program refers to the file.

2. <external file name>, which is the name by which the MCP refers to the file.

The two names can be the same, but this is not a requirement. To change file names at execution time, you need to "equate" the <internal file name> to the new <external file name> through use of a FILE statement. A hardware device different from the one normally used to contain the file can be specified together with the new <external file name>.

To use a FILE statement, you must know the <internal file name> of each file that is to be equated. Table 6-1 gives the <internal file name>s, the default values for the <external file name>s, the expected hardware device, and a description of each file used by the Report Language Analysis Program (RP3REP).

You can use the following FILE statement to change the name of the RP3CRD input file:

```
FILE RP3CRD (TITLE = MYSPECS, KIND = DISK)
```

Table 6-1

RP3REP File Equates for A Series of Systems

Internal File Name	Default External File Name and Kind	Description
RP3GRM	RP3GRM, disk	Grammar definition file.
RP3CRD	RP3CRD, card reader	Input report specification file from the card reader.
RP3PRT	RP3PRT, printer	File where the report specification is printed (and error messages, if any, are listed).

DATA Statement

The DATA statement is placed immediately before the report specification for card decks or Work Flow Language (WFL) decks (the statement is not used if input of the report specification is from disk). The two syntax formats for the statement are as follows:

1. DATA RP3CRD
2. EBCDIC RP3CRD

NOTE

RP3REP does not accept BCL-coded input.

Report-Specification File

The report-specification file consists of all the language statements which you have constructed in designing your report(s), arranged in the appropriate order. This file is placed between the DATA statement and the ?END statement; however, if the report-specification file is a disk file, the DATA statement and ?END statement are not used. The file must be structured similarly to the file containing COBOL source code. The sequence numbers may appear in columns 1 through 6, column 7 is blank, and the language statements of the report specification appear in columns 8 through 72.

?END Statement

The ?END statement, which signals the end of your input, must be the last statement in the input execution deck. The syntax is as follows:

```
?END
```

EXECUTION PROCEDURE

Before you can process your report specification, you must do the following (if not already done): load the REPORTER III System programs and files and the pertinent vocabulary files to disk. Note that the COBOL74 compiler or COBOL85 compiler must be accessible for processing. Note also that if a DMS II data base is to be used, the files DATABASE/INTERFACE and DESCRIPTION/<data base name> must be accessible for processing.

The REPORTER III System has the capability to proceed automatically from specification input to printed report output. However, the system also allows you to process various phases of the report preparation manually. These capabilities, as well as a Work Flow Language (WFL) capability that can be used in connection with REPORTER III on the A Series of Systems, are discussed below.

Automatic Execution Procedure

The procedure for the default automatic execution of the REPORTER III report preparation process is as follows:

1. If you input an execution deck from the card reader or start a WFL job deck, the RP3REP execution can result in the creation of two disk files: a report-specification file if SAVE SPECIFICATIONS is specified, and a parameter file if no errors are encountered. The parameter file will be removed when RP3GEN finishes unless SAVE PARAMETERS is specified.

   If the report-specification file is a disk file and RP3REP is run through the operator display terminal (ODT) or a WFL job deck, the disk file will not be removed; hence no SAVE SPECIFICATIONS statement is required.

2. When RP3REP terminates and there are no errors in the report specification, RP3REP automatically starts RP3GEN.

3. When RP3GEN terminates, the COBOL source program is passed to the COBOL74 compiler or COBOL85 compiler automatically.

4. At the end of compilation, the report program is executed automatically and the report is produced.

If a Process-option SAVE statement has been used to save the object program and no name was specified in the statement for the object program, the program is named "RO<nn>" permanently. If a Process-option SAVE statement has not been used to save the object program, the program is named RO<nn> temporarily.

Following is an example of running RP3REP through the ODT; the report specification is in a disk file called NEW/REPSPC, and it does not contain any Process-option SUPPRESS statements:

```
RUN RP3REP; FILE RP3CRD (TITLE = NEW/REPSPC,
   KIND=DISK)
```

User-Controlled Execution Procedure

By using the Process-option SUPPRESS statement, you can halt the automatic initiation of the various processes of REPORTER III report preparation. By using the Process-option SAVE statement, you can name and preserve the various intermediate files used to pass information from one process to the next. You then must run separate execute decks for each process using file equates to specify the names for the intermediate files.

A program-by-program summary of execution decks and presentation of file equates for each process of REPORTER III report preparation.

RP3REP Program:

RP3REP is the first process in report preparation. It parses the report specification and produces a parameter file for RP3GEN. If no name was assigned to the parameter via the Process-option SAVE statement, the file name is "RP<nn>", where <nn> is a random number assigned by RP3REP. This number is called the ID number, and it remains constant through all processes. The execution deck setup is as follows:

1. RUN RP3REP
2. FILE ---
3. DATA RP3CRD
4. Report-specification file
5. ?END

Table 6-1 lists the file equates for files used by RP3REP.

RP3GEN Program:

After RP3REP, RP3GEN is run to generate a source program based on the parameters produced by RP3REP. It uses the parameter file from RP3REP and the vocabulary files from RP3VOC to produce a source program. The execution deck setup is as follows:

1. RUN RP3GEN
2. FILE ---
3. ?END

Table 6-2 lists the file equates for files used by RP3GEN.

The example below shows part of a WFL deck in which the scope of the usercode family is DEVPACK and Disk but the vocabulary file TSTVII resides on a disk pack called COMPACK.

EXAMPLE:

```
RUN RP3GEN;
FILE RP3PAR (TITLE = DOC/INPUT/PARAM);
FILE APDISK (TITLE = TSTVII ON COMPACK);
```

In this example, the last file-equate card would not be necessary if the vocabulary file resided on DISK or DEVPACK.

Table 6-2

RP3GEN File Equates for
A Series of Systems

Internal File Name	Default External File Name and Kind	Description
RP3PAR	RP3PAR, disk	Input parameters from RP3REP. Must be file-equated to this file.
RP3PRN	RP3PRN, printer	Error messages listing.
APDISK	<vocabulary name>, disk	Input COBOL library. Must be file-equated if vocabulary is on disk pack or outside the scope of the usercode family.

COBOL74 Compiler or COBOL85 Compiler:

After RP3GEN, the COBOL74 compiler or COBOL85 compiler is run to compile the generated program. If no name was assigned to the generated source via the Process-option SAVE statement, the file name is "RC<nn>", where <nn> is the RP3REP ID number. There are two possible situations:

1. COBOL source is saved, source was compiled, and object program is saved.

2. COBOL source is saved and source was compiled, but object program is not saved.

The compile deck is set up as follows:

```
COMPILE <name> WITH COBOL74 LIBRARY;
COMPILER FILE CARD (TITLE = <source name>,
   KIND=DISK);
COMPILER FILE NEWSOURCE (TITLE = <source name>,
   KIND=DISK)
```

Generated Program Object Code:

The generated program object code is executed to produce the desired report. The input for the object program is the data structures to be reported on. The output consists of one or more report and/or extract files. Table 6-3 lists the file equates for the object program; ss represents the section number, rr represents the report section number, mmmm represents the system mix number for the report program, and nn represents the RP3REP ID number.

If the generated report program encounters an exception while creating the report, one of the following conditions apply:

- If the exception is fatal, the exception is reported in the exception report. If the exception report was suppressed in the report specification, a message is displayed on an operator display terminal and the STATUS task attribute for the report program is set to minus one (-1), which causes the program to be discontinued.

  Examples of fatal exceptions include DMS II open errors and errors that occur while a data file is being read.

- If the report program encounters exceptions that are not fatal, the TASKVALUE task attribute for the report program is set to minus one (-1) before the program terminates normally.

Both of these task attributes can be tested in a WFL deck to determine the terminating status of the report program. (Refer to "Execution Using Work Flow Language (WFL)" later in this section.)

Table 6-3

ROnn File Equates for
A Series of Systems

Internal File Name	Default External File Name and Kind	Description
QEXTss	RXnnmmmmss, disk	Internal utility file. Default number of AREAS is 1000. Default AREASIZE is 1000. Value of AREAS and AREASIZE may be changed by the data-processing option SET SORT SIZE statement. Default number of records per block is 10. Value of number of records per block may be changed in the vocabulary by the SET SORT FILE BLOCKING statement or by the data-processing option SET SORT BLOCKING statement.
QPRTrr	RLnnrr, printer	Contains an automatically formatted, printed report and/or a run summary report. External file name may be changed with the report option SAVE REPORT LISTING AS statement.
QEXCEPTIONS	REnn, printer	Contains a notation of exception conditions for all reports. External file name my be changed with the process option SAVE EXCEPTIONS LISTING AS statement.
QPAGrr	QPAGrr, disk	Used for page overflow.

QFRMrr	RFnnrr, printer	Contains a user-formatted report from PRINT statement. External file name may be changed with the report option SAVE FORMS LISTING AS statement.
QRANss	QRANss, disk	Contains random numbers used for sampling.
RP3DAT	RP3DAT, reader	File which contains accepted data that is not from ODT. Kind may be changed by the process option ASSIGN ACCEPTED - DATA. External file name may be changed if ACCEPTED-DATA is assigned to disk.

Execution Using Work Flow Language (WFL)

A Work Flow Language (WFL) capability can be used in connection with the REPORTER III System on the A Series of Systems to facilitate user control of the REPORTER III System and take maximum advantage of the Master Control Program (MCP). The use of WFL is optional (even with WFL, the default progression RP3REP-RP3GEN-COBOL compile-execution can still be controlled internally via Process-option SUPPRESS statements if you so desire). You should consider the advantages of WFL since the transfer to it is a simple process.

The most obvious benefit of WFL is that all tasks within a job are run from the same job stack. This means that all listings appear consecutively on the same printer upon completion of the entire job. Thus, the traditional problem of printer "roulette" is avoided and system messages, task statistics, and so forth, appear conveniently together in a single job summary.

Under WFL, you have the freedom to give task attributes, (for example, PRIORTY or OPTION), either globally or for each step during generation. You also can control attributes for any or all files via FILE statements for each step.

Finally, WFL permits the use of the REPORTER III System in computer installations which require charge codes. Different charge codes and user codes can be associated with each task, as required.

Aside from the need to prepare and keep a small Work Flow deck, there are no disadvantages in using WFL. The same Workflow deck is usable for all REPORTER III runs with no changes except for the <external file name>s unique to each run.

In order for the REPORTER III System to be controlled externally by your WFL rather than being initialized internally, the following option statements must appear in the report-specification file:

```
SUPPRESS GENERATION, COMPILE.
SAVE PARAMETERS AS <external file name 1>,
   COBOL AS <external file name 2>.
```

Example:

```
SUPPRESS GENERATION, COMPILE.
SAVE PARAMETERS AS "PARAM"/"397X",
   COBOL AS "COB"/"397X".
```

To avoid having to enter an option statement for each report run under WFL, you can add the following macro example to each pertinent RP3VOC vocabulary:

```
REPLACE ALL OCCURRENCES OF WFLCONTROL(A,B) BY
   SUPPRESS GENERATION, COMPILE.
SAVE PARAMETERS AS A, COBOL AS B #.
```

You can set up the desired conditions for WFL use with minimal effort by invoking the macro in the report specification as follows:

```
WFLCONTROL(<external file name 1>,
   <external file name 2>).
```

Example:

```
WFLCONTROL ("FRIENDS"/"RELATIVES",
   "TAX"/"ASSESSMENT"/"REPORT").
```

In this example, the file FRIENDS/RELATIVES contains the parameter file produced by RP3REP. The file TAX/ASSESSMENT/REPORT contains the COBOL program generated by RP3GEN.

The following suggested WFL deck indicates a Work Flow Language specification which should be sufficient to handle the majority of cases. This Work Flow specification involves a simple sequential control structure, with each task starting if the previous task was completed successfully; otherwise, an error message is displayed and the job is terminated. Users with Work Flow Language familiarity are able to supply the optional statement images if they desire; they can tailor the WFL deck to their exact needs. Although physical layout and indentation are matters of preference, the method shown in the following examples does enhance readability. This method also isolates the <external file name>s, enabling changes with a minimum of effort (since these are the only elements that change from run to run).

```
BEGIN JOB <job name>; %GENERALIZED REPORTER III SYSTEM WORKFLOW
 USER=<user code>/<password>;% EXAMPLE
 CLASS=<#>; %NORMALLY ONLY IF MULTIPLE JOB QUEUES ENFORCED AT
            %YOUR INSTALLATION
 CHARGE=<charge information>; %IF REQUIRED AT YOUR INSTALLATION
 TASK MYTASK; %REQUIRED BY THE RP3REP IN THIS EXAMPLE
<global task-attribute cards>; %IF DESIRED - SEE WFL MANUAL
%
    RUN RP3REP [MYTASK]; %"MYTASK" WILL CONTROL LOGIC FLOW
    DATA RP3CRD% INPUT SPECIFICATIONS TO RP3REP FOLLOW
        <pertinent password and vocabulary statements>
        WFLCONTROL(<external file name 1>,<external file name 2>).
        <additional report specifications>
?   IF MYTASK (VALUE) = 1 THEN %WE CAN CALL RP3GEN
%
    BEGIN%
        RUN RP3GEN;% NEEDN'T ASSOCIATE TASK WITH THIS STAGE
        <task attribute cards>;% OPTIONAL
        FILE RP3PAR (TITLE=<external file name 1>);
        %CHANGES MADE WITH MINIMAL DIFFICULTY THIS WAY.
%
%   USE "WITH COBOL85" INSTEAD                                    |
%   OF "COBOL74", AS SHOWN BELOW,
%   IF COBOL85 IS INVOLVED.                                       |
%
        COMPILE <external file name> WITH COBOL74 LIBRARY GO;
        <compiler task-attribute cards>;%OPTIONAL
        COMPILER FILE CARD (KIND=DISK, TITLE=
            <external file name 2>);
        COMPILER FILE NEWSOURCE (KIND=DISK,
            TITLE=<external file name 2>;
    END;
?END JOB.
```

If "LIBRARY GO" is not specified in the COMPILE statement of the WFL deck, the default is compile and go (immediately begin execution), and the compiled object is not saved.

The following example is an actual working deck based on the model. Generalities, such as <external file name>s, are given specific values in this example.

Example:

```
BEGIN JOB REPORT/SYSTEM/WFL;%SPECIFIC WFL EXAMPLE
 USER=QQSV123/PASSWORD;
 CHARGE=AR/DEVELOPMENT/095;
 PRIORITY=55;
 TASK MYTASK;
    RUN RP3REP[MYTASK];
    DATA RP3CRD
        PASSWORD IS IRVINE.
        VOCABULARY= "TSTVII".
        WFLCONTROL("PARAMETERFILE","SOURCEFILE").
        .
        .   (remainder of report specification goes here)
        .
?   IF MYTASK(VALUE) = 1 THEN
    BEGIN
        RUN RP3GEN;
        FILE RP3PAR (TITLE= PARAMETERFILE);
        COMPILE FILEAUDIT/VERIFY COBOL74 LIBRARY GO;
        COMPILER FILE CARD (KIND=DISK, TITLE=SOURCEFILE);
    END;
?END JOB.
```

A WFL deck which might be used if the report specification is on disk is shown below; first the WFL deck is shown, and then the contents of the specification DOE/INPUT/REPSPC are shown.

WFL deck:

```
BEGIN JOB DOE/TRY/WFLCONTROL;
 USER= REPORTER ;
 CLASS = 5;
 TASK MYTASK;

   RUN RP3REP [MYTASK];
   FILE RP3CRD(TITLE=DOE/INPUT/REPSPC,KIND=DISK);

   IF MYTASK (VALUE) = 1 THEN
```

```
   BEGIN

     RUN RP3GEN;
     FILE RP3PAR(TITLE=DOE/INPUT/PARAM);

     COMPILE DOE/INPUT/OBJECT WITH COBOL74 LIBRARY GO;
     COMPILER FILE CARD (KIND=DISK, TITLE =
        DOE/INPUT/SOURCE);
     COMPILER FILE NEWSOURCE (KIND=DISK, TITLE =
        DOE/INPUT/SOURCE);
   END;
?END JOB.
```

Specification file DOE/INPUT/REPSPC:

```
000100   VOCAB "DOE"/"INPUT"/"V1".
000200
000300   WFLCONTROL (
000350                  "DOE"/"INPUT"/"PARAM"
000400               ,  "DOE"/"INPUT"/"SOURCE").
000500   SAVE OBJECT AS "DOE"/"INPUT"/"OBJECT".
000510
000520
000600   INPUT ACCESS.
000700   REPORT STUDENT-NAME, STUDENT-ADDRESS, YEAR-IN-SCHOOL.
```

B 1000 OPERATIONS

The following subjects are discussed below with respect to use of the REPORTER III System on a computer in the B 1000 Series of Systems.

1. The files which must be present on disk before a report can be prepared.

2. The sequence of statements needed to execute the entire process from input of the report specification to output of the printed report.

3. The types of procedures you can use to execute the report preparation process:

   a. Automatic execution procedure.
   b. User-controlled execution procedure.

FILES REQUIRED FOR EXECUTION

The REPORTER III System contains the following three files for specifying and generating a report program. The files are supplied on the REPORTER III System tape and must be present on disk when the report specification is analyzed and a report program is generated.

1. RP3REP -- contains the object code for the Report Language Analysis Program.

2. RP3GRM -- contains the report language syntax descriptions used by RP3REP.

3. RP3GEN -- contains the object code for the Report Program Generator.

In addition to the above files, other files must be present on disk or disk pack, as follows:

1. Vocabulary files generated by RP3VOC describing the files to be accessed by the generated program must be present when RP3REP is run. The names of the two vocabulary files needed are given in the output from the RP3VOC execution as follows:

   FILES NEEDED TO RUN REPORTER III ARE:
   <file name 1> AND <file name 2>.

2. The COBOL74 compiler must be present to compile the generated program.

3. If DMS II files are to be accessed, the dictionary and library files created by DASDL must be on disk or disk pack:

   a. <data-base name>/DICTIONARY
   b. #<data-base name>/<disjoint-data-set name>

INPUT REQUIRED FOR EXECUTION

The process of REPORTER III report specification analysis, program generation, program compilation, program execution, and report production is automatic. You need input only one simple sequence of statements to execute the entire process from specification input to printed report output. The input execution deck consists of the following instructions, in the order presented:

1. ?EXECUTE statement
2. ?FILE statement(s)
3. ?DATA statement
4. Report-specification file
5. ?END statement

Descriptions of each of these five instructions follow.

?EXECUTE Statement

The ?EXECUTE statement instructs the Master Control Program (MCP) to schedule the RP3REP program for execution of specification parsing. The syntax is as follows:

```
?EXECUTE RP3REP
```

?FILE Statement(s)

The ?FILE statement is optional. It allows you to change the <external file ID> and/or hardware device for one or more of the files used by RP3REP. Any number of ?FILE statements can be included.

The syntax is as follows:

```
?FILE <internal file name> NAME <external file ID>
  <output device>;
```

Each file used has two names associated with it:

1. <internal file name>, which is the name by which a program refers to the file.

2. <external file ID>, which is the name by which the MCP refers to the file.

The two names can be the same, but this is not a requirement. To change file names at execution time, it is necessary to "equate" the <internal file name> with the new <external file ID> through use of a ?FILE statement. A hardware device different from the one normally employed for a file can be specified together with the new <external file ID>.

To use a ?FILE statement, you must know the <internal file name> of each file that is to be equated. Table 6-4 gives the <internal file name>s, the default values for the <external file ID>s, and a description of each file used by the Report Language Analysis Program, RP3REP.

Table 6-4

RP3REP File Equates for
B 1000 Series of Systems

Internal File Name	Default External File Name and Kind	Description
RP3GRM	RP3GRM, disk	Grammar definition file.
RP3CRD	RP3CRD, card reader	Input report-specification file from the card reader.
RP3PRT	RP3PRT, printer	File where the report specification is printed (and

error messages, if any, are listed).

Example:

The name of the RP3GRM file is to be changed to IMS001. The ?FILE statement for the change is as follows:

```
?FILE RP3GRM NAME IMS001;
```

If a disk file is to be equated to RP3CRD, the word DEFAULT or DEF must be included after the <external file name> to match the block and record sizes of the disk file to RP3CRD.

Example:

The name of the input file is to be changed to PAYROLL/REPORT/EMPLOYEES, and the input medium is to be changed to disk cartridge. The ?FILE statement for the change is as follows:

```
?FILE RP3CRD NAME PAYROLL/REPORT/EMPLOYEES
   DISK.CARTRIDGE DEF;
```

It is possible to change the hardware medium for one or more files. For example, if you want the output file RP3PRT to be sent to printer backup disk without changing the name of the file, the ?FILE statement for the change in output medium is as follows:

```
?FILE RP3PRT NAME RP3PRT BACKUP.DISK NO HARDWARE;
```

NOTE

For further information, refer to the B 1000 Systems Software Operations Guide.

?DATA Statement

The ?DATA statement is placed immediately before the report specification for card decks or pseudo card decks (the statement is not used if input of the report specification is from disk).

The statement names the report specification for the MCP and tells the computer the character set to be used on the input execution deck. The syntax is as follows:

```
?DATA RP3CRD
```

NOTE

> If cards are to be used, only cards punched in EBCDIC can be input to RP3REP.

Report-Specification File

The report-specification file consists of all the language statements which you have constructed in designing your report(s), arranged in the appropriate order. This file is placed between the ?DATA statement and the ?END statement. The file must be structured similarly to the file containing COBOL source code. The sequence numbers appear in columns 1 through 6, column 7 is blank, and the language statements of the report specification appear in columns 8 through 72.

?End Statement

The ?END statement, which signals the end of your input, must be the last statement in the input execution deck. The syntax is as follows:

```
?END
```

EXECUTION PROCEDURE

Before you can process your report specification, you must do the following (if not already done): load the REPORTER III Systems programs and files and the pertinent vocabulary files on disk or diskpack. Note that the COBOL74 compiler must be accessible for processing. Note also

that if a DMS II data base is to be used, the DMS II dictionary and library files must be accessible for processing.

The REPORTER III System has the capability to proceed automatically from specification input to printed report output. However, the system also allows you to process various phases of the report preparation manually. These capabilities are discussed below.

Automatic Execution Procedure

The procedure for the default automatic execution of the REPORTER III report preparation process is as follows:

1. You input the execution deck (from the card reader or terminal, whichever applies) to begin execution of RP3REP. The following message is displayed:

   RP3REP = <mix #> BOJ

2. When RP3REP terminates and there are no errors in the report specification, RP3REP automatically starts RP3GEN. The following message is displayed:

   RP3GEN = <mix #> BOJ.
   RP3REP = <mix #> EOJ.

3. When RP3GEN terminates, COBOL compilation is started automatically. The following message is displayed:

   COBOL74: <COBOL-program> = <mix #> BOJ.
   RP3GEN = <mix #> EOJ.

4. At the end of compilation, the report program is executed automatically. The following message is displayed:

   <COBOL-program> = <mix #> BOJ.
   COBOL74: <COBOL-program> = <mix #> EOJ.

5. When the report is produced, the report program terminates and the following message is displayed:

   <COBOL-program> = <mix #> EOJ.

If a Process-option SAVE statement has been used to save the object program and no name was specified in the statement for the object program, the program is named "RO<nn>" permanently. If a Process-option

SAVE statement has not been used to save the object program, the program is named RO<nn> temporarily.

The example below shows RP3REP being executed through the operator display terminal (OUT). The report specification is contained in a disk file called NEW/REPSPC. The specification does not contain any Process-option SUPPRESS statements.

```
EX RP3REP FI RP3CRD
   NAME NEW/REPSPC DISK DEF
```

User-Controlled Execution Procedure

By using the Process-option SUPPRESS statement, you can halt the automatic initiation of the various processes of REPORTER III report preparation. By using the Process-option SAVE statement, you can name and preserve the various intermediate files used to pass information from one process to the next. You then must run separate execute decks for each process using file equates to specify the names of the intermediate files.

A program-by-program summary of execution decks and presentation of file equates for each process of REPORTER III report preparation follow.

RP3REP Program:

RP3REP is the first process in report preparation. It parses the report specification and produces a parameter file for RP3GEN. If no name was assigned to the parameter via the Process-option SAVE statement, the file name is "RP<nn>", where <nn> is a random number assigned by RP3REP. This number is called the ID number, and it remains constant through all processes. The execution deck setup is as follows:

1. ?EX RP3REP
2. ?FILE ---
3. ?DATA RP3CRD
4. Report-specification file
5. ?END

Table 6-4 lists the file equates for files used by RP3REP.

RP3GEN Program:

After RP3REP, RP3GEN is run to generate a source program based on the parameters produced by RP3REP. It uses the parameter file from RP3REP and the vocabulary files from RP3VOC to produce a source program. The execution deck setup is as follows:

1. ?EX RP3GEN
2. ?FILE ---

Table 6-5 lists the file equates for files used by RP3GEN.

Table 6-5

RP3GEN File Equates for
B 1000 Series of Systems

Internal File Name	Default External File Name and Kind	Description
RP3PAR	RP3PAR, disk	Input parameters from RP3REP must be file-equated to this file.
RP3PRN	RP3PRN, printer	Error messages listing.

COBOL74 Compiler:

After RP3GEN, the COBOL74 compiler is run to compile the generated program. If no name was assigned to the generated source via the Process-option SAVE statement, the file name is "RC<nn>", where <nn> is the RP3REP ID number. There are two possible situations:

1. COBOL source is saved, source was compiled, and object program is saved.
2. COBOL source is saved and source was compiled, but object program is not saved.

In situations 1 and 2, the compiler has resequenced the source and removed the control card. If the source program is not compiled, or if it is compiled without NEWSOURCE being file-equated to the source, the source file's sequence numbers will be somewhat random or nonexistent and the source file will contain a compiler control card as the first card. (Refer to the COBOL ANSI-74 manual for information on running the compiler.) The compile deck would be set up as follows:

```
CO <name> WI COBOL74 LI FI CARD NAME
   <source name> DISK DEF;
      FI NEWSOURCE NAME <source name>
```

Generated Program Object Code:

The generated program object code is executed to produce the desired report. The input for the object program is the data structures to be reported on. The output consists of one or more report and/or extract files. Table 6-6 lists the file equates for the object program; ss represents the section number, rr represents the report section number, mm represents the seconds portion of the system time when the report program was executed, and nn represents the RP3REP ID number.

Table 6-6

ROnn File Equates for
B 1000 Series of Systems

Internal File Name	Default External File Name and Kind	Description
QEXTss	RXnnmmss, disk	Internal utility file. Default number of AREAS is 100. Default AREALENGTH is compile time default. Value of AREAS and AREALENGTH may be changed by the data-processing option SET SORT SIZE statement. However, the number of records given in the SET SORT SIZE statement must be a multiple of number of records per block. Default number of records per block is 10.

		Value of number of records per block may be changed in the vocabulary by the SET SORT FILE BLOCKING statement or by the data-processing option SET SORT BLOCKING statement.
QPRTrr	**RLnnrr, printer**	Contains an automatically formatted, printed report and/or a run summary report. External file name may be changed with the report option SAVE REPORT LISTING AS statement.
QEXCEPTIONS	**REnn, printer**	Contains notations of exception conditions for all reports. External file name may be changed with the process option SAVE EXCEPTIONS LISTING AS statement.
QPAGrr	**QPAGrr, disk**	Used for page overflow.
QFRMrr	**RFnnrr, printer**	Contains a user-formatted report from PRINT statement. External file name may be changed with the report option SAVE FORMS LISTING AS statement.
QRANss	**QRANss, disk**	Contains random numbers used for sampling.
RP3DAT	**RP3DAT, reader**	File which contains accepted data if not from the ODT. Kind may be changed by the process option ASSIGN ACCEPTED-DATA. External file name may be changed if ACCEPTED-DATA is assigned to disk.

B 2000/B 3000/B 4000 OPERATIONS

The following subjects are discussed below with respect to use of the REPORTER III System on a computer in the B 2000/B 3000/B 4000 Series of Systems.

1. The files which must be present on disk before a report can be prepared.

2. The sequence of statements needed to execute the entire process from input of the report specification to output of the printed report.

3. The types of procedures you can use to execute the report preparation process:

   a. Automatic execution procedure.
   b. User-controlled execution procedure.

FILES REQUIRED FOR EXECUTION

The REPORTER III System contains the following four files for specifying and generating a report program. These files are supplied on the REPORTER III System tape and must be present on disk when the report specification is analyzed and a report program is generated.

1. RP3REP — contains the object code for the Report Language Analysis Program.

2. RP3GRM — contains the report language syntax descriptions used by RP3REP.

3. RP3GEN — contains the object code for the Report Program Generator.

In addition to the above files, other files must be present on disk or disk pack, as follows:

1. Vocabulary files generated by RP3VOC describing the files to be accessed by the generated program must be present when RP3REP is run. The names of the two required vocabulary files are given in the output from the RP3VOC execution as follows:

   FILES NEEDED TO RUN REPORTER III ARE:
   <file name 1> AND <file name 2>.

2. The COBOL compiler must be present to compile the generated program.

3. If DMS II data structures are to be accessed, the data base files created by DASDL must be present.

INPUT REQUIRED FOR EXECUTION

The process of REPORTER III report specification analysis, program generation, program compilation, program execution, and report production is automatic. You need input only one simple sequence of statements to execute the entire process from specification input to printed report output. The input execution deck consists of the following instructions, in the order presented:

1. ?EXECUTE statement
2. ?FILE statement(s)
3. ?DATA statement
4. Report-specification file
5. ?END statement

Descriptions of each of these five instructions follow.

?EXECUTE Statement

The ?EXECUTE statement instructs the Master Control Program (MCP) to schedule the RP3REP program for execution of specification parsing.

For card input of the report specification to RP3REP, the syntax is as follows:

```
?EXECUTE RP3REP
```

For disk input of the report specification to RP3REP, the syntax is as follows:

```
?EXECUTE RP3REP VA 0 3
```

?FILE Statement(s)

The ?FILE statement is optional. It allows you to change the <external file ID> and/or hardware device for one or more files used by RP3REP. Any number of ?FILE statements can be included. The syntax is as follows:

```
?FILE <internal file name> = <external file ID> <backup
   device>
```

Each file has two names associated with it:

1. <internal file name>, which is the name by which a program refers to the file.

2. <external file ID>, which is the name by which the MCP refers to the file.

The two names can be the same, but this is not a requirement. To change file names at execution time, you need to "equate" the <internal file name> to the new <external file ID> through use of a ?FILE statement. A backup output device other than the one normally employed for a file can be specified together with the new <external file ID>.

To use a ?FILE statement, you must know the <internal file name> of each file that is to be equated. Table 6-7 gives the <internal file name>s, the default values for the <external file ID>s, and a description of each file used by the Report Language Analysis Program (RP3REP).

Table 6-7

RP3REP File Equates for B 2000/
B 3000/B 4000 Series of Systems

Internal File Name	Default External File Name and Kind	Description
RP3GRM	RP3GRM, disk	Grammar definition file.
RP3CRD	RP3CRD, card reader	Input report-specification file from the card reader.
RP3PRT	RP3PRT, printer	File where the report specification is printed (or error messages, if any, are listed).
RP3DSK	RP3DSK, disk	Input report-specification file from disk. Use VA 0 3 in ?EXECUTE statement to specify input from disk.

Example:

The name of the RP3GRM file is to be changed to IMS001. The ?FILE statement for the change is as follows:

```
?FILE RP3GRM = IMS001
```

It is possible to specify that a backup file be used instead of the physical hardware device. For example, if you want the output file RP3PRT to be sent to printer backup disk without changing the name of the file, the ?FILE statement for the change in the output medium is as follows:

```
?FILE RP3PRT = RP3PRT PRINT DISK
```

NOTE

For further information, refer to the B 2000/B 3000/B 4000 Software Operations Guide.

?DATA Statement

The ?DATA statement is placed immediately before the report specification for card decks or pseudo card decks (the statement is not used if input of the report specification is from disk). This statement names the report specification for the MCP and tells the computer the character set to be used on the input execution deck. The syntax is as follows:

```
?DATA RP3CRD
```

NOTE

If cards are to be used, only cards punched in EBCDIC can be input to RP3REP.

Report-Specification File

The report-specification file consists of all the language statements which you have constructed in designing your report(s), arranged in the appropriate order. This file is placed between the ?DATA statement and the ?END statement. The file must be structured similarly to the file containing COBOL source code. The sequence numbers appear in columns 1 through 6, column 7 is blank, and the language statements of the report specification appear in columns 8 through 72.

?END Statement

The ?END statement, which signals the end of your input, must be the last card in the input execution deck. The syntax is as follows:

```
?END
```

EXECUTION PROCEDURE

Before you can process your report specification, you must do the following (if not already done): load the REPORTER III System programs and files and the pertinent vocabulary files on disk. Note that the COBOL compiler must be accessible for processing.

The REPORTER III System has the capability to proceed automatically from specification input to printed report output. However, the system also allows you to process various phases of the report preparation manually. These capabilities are discussed below.

Automatic Execution Procedure

The procedure for the default automatic execution of the REPORTER III report preparation process is as follows:

1. You input the execution deck (from the card reader or terminal, whichever applies) to begin execution of RP3REP. The following message is displayed on the ODT:

   ```
   BOJ RP3REP
   ```

2. When RP3REP terminates and there are no errors in the report specification, RP3REP automatically starts RP3GEN. The following message is displayed:

   ```
   BOJ RP3GEN
   EOJ RP3REP
   ```

3. When RP3GEN terminates, COBOL compilation is started automatically. The following message is displayed:

   ```
   BOJ <COBOL-program>/COBOL
   EOJ RP3GEN
   ```

4. At the end of compilation, the report program is executed automatically. The following message is displayed:

   ```
   EOJ <COBOL-program>/COBOL
   BOJ <COBOL-program>
   ```

5. When the report is produced, the report program terminates and the following message is displayed:

   ```
   EOJ <COBOL-program>
   ```

If a Process-option SAVE statement has been used to save the object program and no name was specified in the statement for the object program, the program is named "RO<nn>" permanently. If a Process-option SAVE statement has not been used to save the object program, the program is named RO<nn> temporarily.

The example below shows RP3REP being executed through the operator display terminal (ODT). The report specification is contained in a disk file called NEWSPC. The specification does not contain any Process-option SUPPRESS statements.

```
EX RP3REP VA 0 3 FILE RP3DSK NEWSPC.
```

User-Controlled Execution Procedure

By using the Process-option SUPPRESS statement, you can halt the automatic initiation of the various processes of report preparation. By using the Process-option SAVE statement, you can name and preserve the various intermediate files used to pass information from one phase to the next. You then must run separate execute decks for each process using file equates to specify the names for the intermediate files.

A program-by-program summary of execution decks and presentation of file equates for each process of REPORTER III report preparation follow.

RP3REP Program:

RP3REP is the first process in report preparation. It parses the report specification and produces a parameter file for RP3GEN. If no name was assigned to the parameter via the Process-option SAVE statement, the file name is "RP<nn>", where <nn> is a random number assigned by RP3REP. This number is called the ID number, and it remains constant through all processes. The execution deck setup is as follows:

1. ?EX RP3REP
2. ?FILE ---
3. ?DATA RP3CRD
4. Report specification
5. ?END

RP3REP can take the input report specification from disk instead of cards. To accomplish this, the following execution deck setup is used:

1. ?EX RP3REP VA 0 3
2. ?FILE ---

In this case, input report specification is taken from the file RP3DSK instead of RP3CRD.

Table 6-7 lists the file equates for files used by RP3REP.

RP3GEN Program:

After RP3REP, RP3GEN is run to generate a source program based on the parameters produced by RP3REP. It uses the parameter file from RP3REP and the vocabulary files from RP3VOC. The execution deck setup is as follows:

1. ?EX RP3GEN
2. ?FILE ---

Table 6-8 lists the file equates for files used by RP3GEN.

The example below shows part of a execution deck in which the vocabulary file COMVC1 is on a disk pack called COMPACK.

Example:

```
?EX RP3GEN;
?FILE RP3PAR = MYPAR;
?FILE APDISK = COMPCK/COMVC1 DPK
```

In this example, the last file-equate card is not needed if the vocabulary files reside on disk.

Table 6-8

RP3GEN File Equates for B 2000/
B 3000/B 4000 Series of Systems

Internal File Name	Default External File Name and Kind	Description
RP3PAR	RP3PAR, disk	Input parameters from RP3REP. Must be file-equated to this file.
RP3PRN	RP3PRN, printer	Error messages listing.
APDISK	<vocabulary name>, disk	Input COBOL library must be file-equated if vocabulary is not on disk.

COBOL Compiler:

After RP3GEN, the COBOL compiler is run to compile the generated program. If no name was assigned to the generated source via the Process-option SAVE statement, the file name is "RC<nn>", where <nn> is the RP3REP ID number. There are two possible situations:

1. COBOL source is saved, source was compiled, and object program is saved.

2. COBOL source is saved and source was compiled, but object program is not saved.

In situations 1 and 2, the compiler has resequenced the source and removed the control card. If the source program is not compiled, or if it is compiled without NEWSOU being file-equated to the source, the source file's sequence numbers will be somewhat random or nonexistent and the source file will contain a compiler card as the first card. (Refer to the COBOL ANSI-74 manual for information on running the compiler.)

To resume standard REPORTER III processing, the COBOL compiler must be invoked as follows:

```
?COMPILE ROnn COBOL IN 0 4 UA "D   "
?FILE CARD RCnn
?FILE NEWSOU = RCnn
```

The file names ROnn and RCnn are the default names for the object code and the source code, respectively, and can be changed as needed.

Generated Program Object Code:

The generated program object is executed to produce the desired report. The input for the object program is the data structures to be reported on. The output consists of one or more report and/or extract files. Table 6-9 lists the file equates for the object program; ss represents the section number, rr represents the report section number, mm represents the system mix number for the report program, and nn represents the RP3REP ID.

Table 6-9

ROnn File Equates for B 2000/
B 3000/B 4000 Series of Systems

Internal File Name	Default External File Name and Kind	Description
QEXTss	RXmmss, disk	Internal utility file. Default number of AREAS is 20. Default AREASIZE is 5000. Value of AREAS and AREASIZE may be changed by the data-processing option SET SORT SIZE statement. Default number of records per block is 10. Value of number of records per block may be changed in the vocabulary by the SET SORT FILE BLOCKING statement or by the data-processing option SET SORT BLOCKING statement.
QPRTrr	RLnnrr, printer	Contains an automatically formatted, printed report and/or a run summary report. External file name may be changed with the report option SAVE REPORT LISTING AS statement.
QEXCEPTIONS	REnn, printer	Contains a notation of exception conditions for all reports. External file name may be changed with the process option SAVE EXCEPTIONS LISTING AS statement.
QPAGrr	QPAGrr, disk	Used for page overflow.

QFRMrr	RFnnrr, printer	Contains a user-formatted report from PRINT statement. External filename may be changed with the report option SAVE FORMS LISTING AS statement.
QRANss	QRANss, disk	Contains random numbers used for sampling.
RP3DAT	RP3DAT, reader	File which contains accepted data that is not from ODT. Kind may be changed by the process option ASSIGN ACCEPTED-DATA. External file name may be changed if ACCEPTED-DATA is assigned to disk. If ACCEPTED-DATA is assigned to disk, the file must be BLOCK 1.

APPENDIX A

REPORTER III SYSTEM FLOW

A REPORTER III report-language specification (generally referred to as a report specification) describes the information to be reported and the manner in which it is to be reported. (Instructions and formats for processing a report specification are presented in Section 6, System Operations Associated with Report Preparation.)

A diagram representing the process performed by the REPORTER III System to prepare a report(s) based on a report specification is presented in Figure A-1.

REPORTER III report preparation consists of four processes. After the first process, each process is executed automatically unless an error has occurred in the previous process, or unless the user inhibits automatic execution by using a Process-option SUPPRESS statement in the report specification.

The first process involves the RP3REP program. During this process, the report specification is analyzed for syntax and semantics errors. If errors are encountered, a diagnostic report of the problem is produced and the automatic process is terminated. A listing of the report specification is produced whether or not errors occurred. If no errors are encountered, a parameter file is created to be used in the second process.

The second process involves the RP3GEN Program. During this process, the parameter file created in the first process and the vocabulary are used to generate a COBOL source program. On A Series systems, the COBOL source program can be COBOL74 or COBOL85. On all other systems, only COBOL74 source programs are generated.

The third process is compilation of the source program code with the appropriate compiler:

1. COBOL74 or COBOL85 compiler, if a computer in the A Series of Systems is being used.

2. COBOL74 compiler, if a computer in the B 1000 Series of Systems is being used.

3. COBOL compiler, if a computer in the B 2000/ B 3000/B 4000 Series of Systems is being used.

Compilation of the source program code produces an executable object program.

The fourth process is execution of the object program produced in the third process. The output is the desired report.

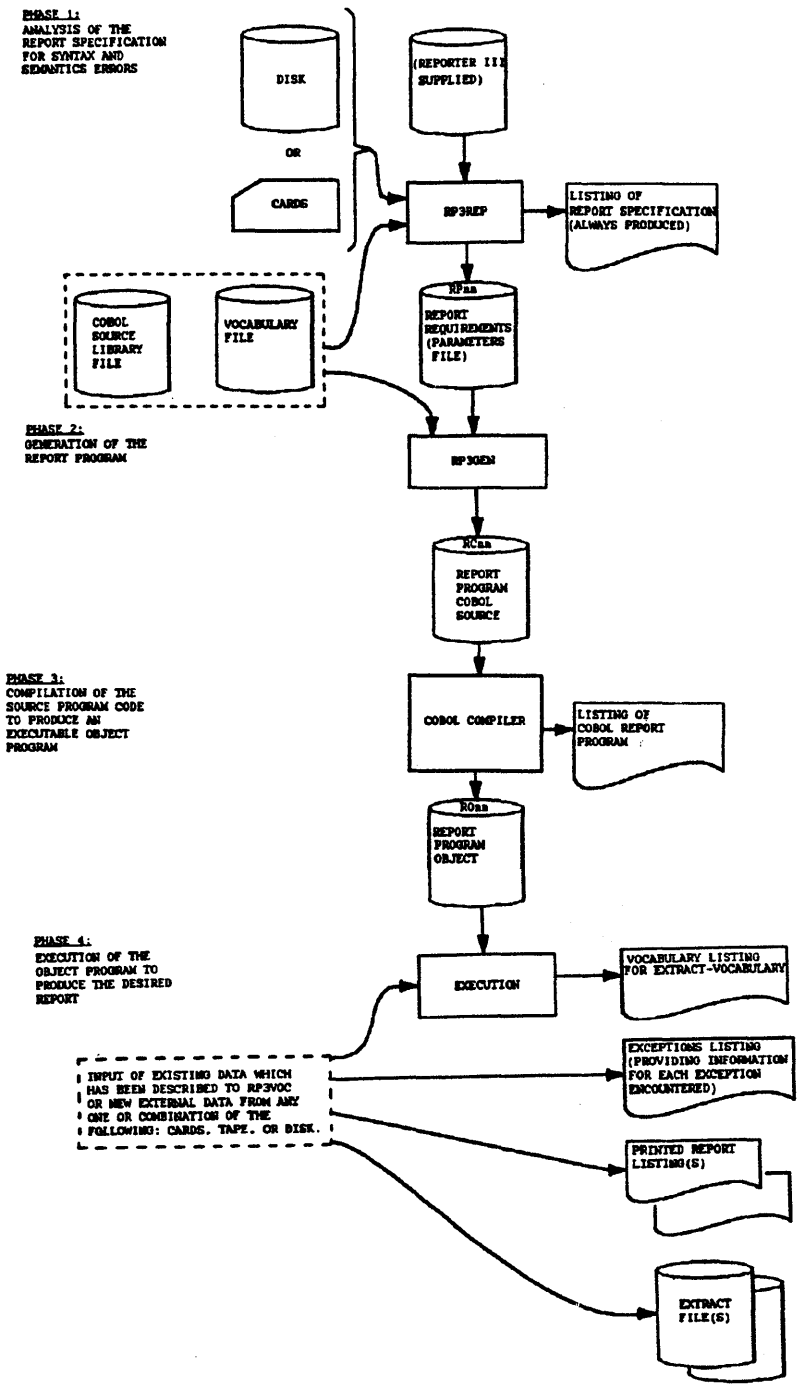


Figure A-1. REPORTER III Report Preparation Process

APPENDIX B

LIMITS AND DEFAULTS

The REPORTER III report language limits and defaults are indicated in this appendix. The limits are restrictions in terms of quantity or size. The defaults are values assigned by the system when values are not specified explicitly by the user.

ITEMS, FUNCTIONS, AND CONSTRUCTS

The report language limits for items, functions, and basic language constructs are indicated below.

ITEMS

A maximum of 500 unique items can be defined or referenced in a report specification; <expression>s which are defined internally by the system as items are included in this limit.

FUNCTIONS

A maximum of 999 functions can be given in a report specification.

BASIC LANGUAGE CONSTRUCTS

Limits are indicated for each basic language construct identified below. The constructs are presented alphabetically.

COBOL Picture

The maximum length of a <COBOL picture>, which is an editing picture, is 30 characters. The maximum number of digits allowed in a numeric picture clause is 23 for the A Series and B 2000/B 3000/B 4000 Series of Systems. The maximum number is 18 for the B 1000 Series of Systems.

Data Name

The maximum length of a <data name>, including subscripts and qualifiers, is 60 characters. However, in determining the length of a <data name>, 11 characters must be counted for each subscript which is a derived item, rather than the length of the <name> which references the derived item.

The maximum number of qualifiers in a <data name> is four.

The number of subscripts specified for a <data name> must be identical to the number of subscripts identified in the vocabulary listings for that <data name>. It is the user's responsibility to ensure that a subscript is within the bounds of the array if the subscripted <data name> is either a DMS II item specified in the INPUT statement or a non-DMS II item specified in any report language statement. If a subscript value is not within the bounds of the array, a run-time error (for example, INVALID INDEX or INVALID SUBSCRIPT) could occur during the execution of the generated report program. For the remaining cases, REPORTER III would check subscript values automatically. If an invalid subscript is detected at run time, the array element is handled as a null item.

Expression

Parentheses can only be nested 30 deep. A maximum of 999 operators per <expression> and 999 <expression>s per report specification is allowed. A maximum of three passes through the logical record to compute statistical expressions is allowed.

External File Name

For the A Series of Systems, the maximum length of an <external file name> is 30 characters, including the slashes. The maximum length of a name following the ON clause is 17 characters.

For the B 1000 Series of Systems, the maximum length of an <external file name> is 21 characters, including the slashes. If no ON clause is given, the maximum length of the string is 10 characters. The total length, including all elements of the <external file name>, cannot exceed 28 characters.

For the B 2000/B 3000/B 4000 Series of Systems, the maximum length of a <string> in an <external file name> is six characters. The maximum length of the string in the ON clause, if given, is six characters.

Literal

The maximum length of a numeric literal used for arithmetic operations, including the sign, is 23 characters for the A Series and B 2000/B 3000/B 4000 Series, and 18 for the B 1000 Series.

The maximum length of a string is 55 characters, excluding the quotation marks.

A maximum of 99 strings can be used in a report specification.

REPORT LANGUAGE STATEMENTS

The limits and defaults which apply to particular REPORTER III report language statements are presented in the following table. The statements are listed alphabetically.

Statement	Limit/Default
ACCEPT statement	The default for item type is NUMERIC. The maximum length for each NUMERIC item type, including the sign, is 23 characters for the A Series and B 2000/B 3000/B 4000 Series, and 18 characters for the B 1000 Series. If no lengths are specified in the statement, the default values are those specified by the SET-STRING-SIZE, INTEGER-SIZE, and FRACTION-SIZE options. The DISPLAY limit for an ACCEPT statement is 55 characters.
BUILD statement	The default item type for the BUILD statement is NUMERIC. The item is unsigned with INTEGER-SIZE and FRACTION-SIZE as system default values or the value set in the Process-option SET statement.

	If NUMERIC or STRING is used as the data type for the BUILD statement without specifying the size, the system default value or the value given in the Process-option SET statement will be used.
EXTRACT FILE AREASIZE statement	Default is 5000 records. For the A Series of Systems, the maximum number of records per area is 1,048,575. For the B 1000 Series of Systems, the product of the number of areas and the AREASIZE cannot exceed 16,777,216. For the B 2000/B 3000/B 4000 Series of Systems, the product of the number of areas and the AREASIZE cannot exceed 99,999,999.
EXTRACT FILE AREA statement	Default is 20. For the A Series of Systems, the maximum number of areas is 1000. For the B 1000 Series of Systems, the maximum number of areas is 105. For the B 2000/B 3000/B 4000 Series of Systems, the maximum number of areas is 100. For all the systems, the minimum number of areas is one.
EXTRACT statement	Defaults are the following: HARDWARE DEVICE = DISK; ID = <file name>; TOTAL-POP = 9999.
GROUP statement	The maximum number of control breaks is nine.
INPUT statement	The maximum number of input levels is nine. The maximum number of data structures (files) allowed is nine.
ORDER statement	The maximum number of ordering keys which you can specify is nine.

PRINT statement	For form width, the default is 132 and the maximum also is 132. For form length, the default is 54 and the maximum is 9999. For a <conditional print specification>, the maximum level of nesting is 20.
RANGE statement	The maximum number of FROM...BY...TO groups which can be specified is 99.
REPLACE statement	A maximum of 50 unique macros can be used to describe a report. The maximum number of parameters is 10. The maximum number of nested macro definitions is 10.
REPORT statement	The maximum number of items which can be reported is 99. The maximum number of control breaks is nine. The maximum length of the <string> in the AS clause is 30 characters, excluding quotation marks. The maximum width of all reported items is 660 characters.
SAMPLE statement	A maximum of 99 strata can be specified. Refer to Table B-1 for parameter limits which apply to the SAMPLE statement.
SUMMARIZE statement	The maximum length of the <string> in the AS clause is 30 characters, excluding quotation marks.
TABLE statement	A table number can contain a maximum of 10 digits (nine digits plus decimal point). A table name can contain a maximum of 10 characters. The maximum number of table entries is 999. The maximum number of tables which can be defined in a single-report specification is nine. The number of tables defined in the Input Section plus the number of tables defined in a Report Section of a multiple-report specification cannot exceed nine.

TITLE statement	The maximum length of a <string>, excluding quotation marks, in a TITLE statement is 55. Concatenation can be used to obtain longer titles. The maximum number of elements per title is 99. The maximum number of lines per title is PAGE-LENGTH. The maximum number of specified title lines, excluding skipped lines, for all titles is 50.

Table B-1

Parameter Limits for
SAMPLE Statement

Parameters	Limits	
Sample size	1 to 99,999,999	
Sample size percent	1 to 100	
Strata population	Sample size to 99,999,999	
Seed	0 to 99,999,999	
Sum of all strata population	A Series:	1,080,000,000
	B 1000:	108,000,000
	B 2000/B 3000/B 4000:	108,000,000

PROCESS-OPTION STATEMENTS

The limits and defaults which apply to the Process-option statements are presented in the following table. The statements are listed alphabetically.

Statement	Limit/Default
ASSIGN statement	The default hardware device for accepted data is READER if PROCESSING MODE is BATCH, and TERMINAL if it is ON-LINE. The default <external file name> is RP3DAT. The exceptions listing is assigned to PRINTER if PROCESSING MODE is BATCH; it is assigned to TERMINAL BACKUP if PROCESSING MODE is ON-LINE.
SAVE statement	If no Process-option SAVE statement is used, the following actions result by default: SPECIFICATIONS are not saved. PARAMETERS are not saved. COBOL SOURCE is saved. OBJECT is not saved. EXCEPTIONS LISTING is saved if number of exceptions > 0. EXTRACT-VOCAL SPECIFICATIONS are not saved. EXTRACT-VOCABULARY is saved if it has been generated. If <external file name> is not provided by means of an AS phrase, the following standard names are used: SPECIFICATIONS "RSnn" PARAMETERS "RPnn" COBOL SOURCE "RCnn" OBJECT "ROnn" EXCEPTIONS LISTING "REnn" EXTRACT-VOCAL SPECIFICATIONS "EVnn" EXTRACT-VOCABULARY: <external file name 2> "V1nn" <external file name 3> "V2nn"

NOTE

The "nn" in the preceding names represents an identification number assigned when the Report Language Analysis Program (RP3REP) is run and is based on the TIME register.

SET statement — Refer to Table B-2 for limits and defaults which apply to the Process-option SET statement.

SUPPRESS statement — Actions which are not specifically suppressed will be initiated.

TOTAL-POP — Maximum is TOTAL-POP of the referenced data structures.

Table B-2

Limits and Defaults for
Process-option SET Statement

Statement	Default	Limit
ABBREVIATED MONTH LIST		The language is English. The values 1 through 12 correspond to months JAN through DEC.
FULL-LENGTH MONTH LIST		The language is English. The values 1 through 12 correspond to the months JANUARY through DECEMBER.
INTEGER-SIZE	12	For A Series and B 2000/B 3000/ B 4000 Series systems: 0 to 23 minus FRACTION-SIZE. For B 1000 Series systems: 1 to 18 minus FRACTION-SIZE.

FRACTION-SIZE	5	For A Series and B 2000/B 3000/ B 4000 Series systems: 0 to 23 minus INTEGER-SIZE. For B 1000 Series systems: 0 to 18 minus INTEGER-SIZE.
STRING-SIZE	30	PAGE-WIDTH
NULL-NUMERIC	0	---
NULL-STRING	SPACES	---
NULL-BOOLEAN	FALSE	---
MODE	BATCH (ON-LINE if Report Language Analysis Program (RP3REP) is run via On-Line REPORTER III).	---

DATA-PROCESSING-OPTION STATEMENTS

The limits and defaults which apply to the Data-processing-option statements are presented in the following table:

Statement	Limit/Default
SET Statement	The SORT-FILE BLOCKING and SORT-FILE SIZE default values are as specified in the REPORTER III Vocabulary Language Program (RP3VOC). For the A Series systems, the maximum SORT-FILE size is 90 million words (540 million characters).
SUPPRESS statement	Unless suppressed, a sort is done to group and/or order the information in the specified sequence.

REPORT-OPTION STATEMENTS

The limits and defaults which apply to the Report-option statements are presented in the following table. The statements are listed alphabetically.

Statement	Limits/Defaults
ASSIGN statement	If PROCESSING MODE is BATCH, the forms and report listings are assigned to PRINTER. If PROCESSING MODE is ON-LINE, they are assigned to TERMINAL BACKUP.
SAVE statement	The <external file name> for the automatically formatted report and run summary report listing is, by default, "RLnnrr". The <external file name> for the user-formatted report listing is, by default, "RFnnrr". The "nnrr" portion of the file name is a 4-digit number where nn is the report ID number and rr is the report number.
SET statement	The limits and defaults for the various formatting attributes of the Report-option SET statement are indicated in Table B-3.
SUPPRESS statement	Pages in the printed report are numbered unless this action is suppressed.

Table B-3

Limits and Defaults for
Report-option SET Statement

Formatting Attribute	Default	Limit
PAGE-WIDTH	132	32-132
PAGE-LENGTH	54	8-99
COLUMN-SPACING	1	0-30
VERTICAL-SPACING	1 if page overflow or no overflow. 2 if line overflow.	1-3
BLOCKING	As specified in REPORTER III Vocabulary Language Program (RP3VOC).	---
TOP-OF-FORM-CH	VERTICAL-SPACING	1-11
FORMS-PATTERNS	0	1-99
PAGE-OVERFLOW	FALSE	---

MULTIPLE-REPORT SPECIFICATION

A multiple-report specification can contain a maximum of one Input Section and 98 Report Sections.

APPENDIX C

ERROR AND WARNING MESSAGES AND EXCEPTIONS LISTINGS

This appendix presents the possible error and warning messages which can be issued by the REPORTER III Report Language Analysis Program (RP3REP) when it analyzes the syntax and semantics of the report language statements of a report specification. This appendix also presents the possible exceptions which can be encountered and noted during the execution of a generated report program.

ERROR AND WARNING MESSAGES

When the Report Language Analysis Program (RP3REP) has completed its run, all error and warning messages reflecting its analysis of syntax and semantics are noted on the output listing of the report specification. On the output listing, each error or warning message immediately follows the line containing the portion of the language statement to which the message applies. When an asterisk (*) appears in an error message, it appears either directly beneath the portion of the language statement that caused the error, or directly beneath the portion of the language statement immediately following the portion that caused the error.

A message which appears on the output listing preceded by the word "ERROR" is an "error message." It states the reason that processing of the pertinent language statement by the RP3REP program has discontinued. RP3REP then searches for the period in that statement and proceeds to process the next language statement.

A message which appears on the output listing preceded by the word "WARN" is a "warning message." It indicates an abnormal condition encountered by the RP3REP program in processing the pertinent language statement. Processing of the language statement is not discontinued when a "warning message" is issued.

The RP3REP program can detect only one error in a language statement in a given run. If a detected error in the statement is corrected and if the statement contains additional errors which are undetected and uncorrected, the next error in the statement will be detected by RP3REP when it is run to analyze the given report specification again.

The following is a list of error messages not prefixed by a number. An explanation is provided for each message.

****ERROR** NOT ENOUGH AVAILABLE SPACE ON PAGE TO PRINT THIS DATA ITEM**

The length of the data item plus indentation is greater than the PAGE-WIDTH.

ILLEGAL QUALIFICATION

This error is the result of a specified qualification which is hierarchically incorrect.

INVALID KEY WHILE READING VOCAF

This error is internal to the REPORTER III System. It is followed by the message PLEASE SEND ALL INFORMATION REQUIRED TO DUPLICATE THIS PROBLEM TO BURROUGHS.

INVALID TYPE OF QUALIFICATION

This error is the result of a specified qualification which is hierarchically incorrect.

NAME IS NOT FOUND IN VOCABULARY

The name was not an accepted or derived item name and was not in the vocabulary.

NAME IS NOT UNIQUE

The vocabulary item and/or its qualifiers is/are not unique.

The following is a list of the numbered error messages and warning messages. An explanation is provided for each message. Message numbers which are not in use are so-indicated in the list. On an output listing, the 3-digit number of a numbered message, which appears on the left side of the message, is preceded by the word "ERROR" or "WARN".

000 NAME IS TOO LONG

The length of a <data name> or <name> with qualifiers exceeds the allowable limit (refer to Appendix B: Limits and Defaults), or a <name> is longer than 30 characters.

001 NAME EXPECTED

A <data name> was expected, but none was given.

002 INVALID SUBSCRIPT

The subscript is not a valid <data name> or <integer>.

003 RIGHT BRACKET EXPECTED

A right bracket was expected at this point in the statement, but none was given.

004 INVALID NAME

This error indicates the presence of a previous error.

005 INVALID NUMBER OF SUBSCRIPTS

The number of subscripts given does not agree with the number defined in the vocabulary.

006 NAME HAS ALREADY BEEN DEFINED

The new name you are attempting to define already exists in the vocabulary as an extension, or in a BUILD statement as a name.

007 NAME NOT IN VOCABULARY AND NOT DEFINED AS ACCEPTED OR DERIVED ITEM

The name which you have given was not recognized as an accepted or derived item name or as a name within the vocabulary.

008 INVALID STATISTIC DESCRIPTION

Syntax is incorrect. Refer to STATISTIC DESC in Section 4 for correct syntax of the <statistical function> clause.

009 MISSING KEYWORD: 'PAGE(S)'

In the TITLE statement, reference is made to "EACH", "FIRST", or "OTHER" without the word "PAGE" or "PAGES" following.

010 PREVIOUS <NAME> UNDEFINED, 'IS' REQUIRED FOR DEFINITION

The previous <name> has not been defined. An "IS" is required at this point for derived item definition.

011 TITLE TOO LONG FOR 1 LINE--TITLE LINE #<nn>

The <n>th line of title information which you specified in the TITLE statement is too long to fit on one line. The length of a given title line is restricted to 132 characters or to the PAGE-WIDTH setting if that option was specified. If a title length greater than one line is desired, a slash (/) must be used as explained under TITLE STATEMENT in Section 4.

012 INVALID USE OF STATISTICAL FUNCTION

The expression cannot be statistical; therefore, it cannot contain a statistical function.

013 INTERNAL ERROR, CONTACT BURROUGHS

This error occurs only when the Report Language Analysis Program detects internal inconsistencies. These are due to system failures. The dump and appropriate listings should be sent to Burroughs. The vocabulary files and the Reporter specification, and any other information necessary to reproduce the problem should be sent on a tape to Burroughs.

014 INVALID TITLE STRUCTURE

The syntax of the TITLE statement needs to be corrected; it should be checked for an invalid option.

015 NAME NOT IN INPUT STRUCTURES

The <data name> is not in any of the data structures specified in the INPUT statement. The vocabulary listing should be checked.

016 INVALID <DATA-NAME> OR CONDITION NAME

A name in an expression must reference an item or a condition.

017 INVALID OPERAND TYPE

A number, literal, condition, or pattern cannot reside on the left side of a relational operator.

018 TOO MANY RIGHT PARENS

The right parenthesis does not correspond to a left parenthesis.

019 LIMIT ON TITLE LINES EXCEEDED

The limit on the number of title lines was exceeded. Refer to Appendix B, Limits and Defaults.

020 EXPRESSION TOO LONG

The expression cannot be handled because of the number, nature, and/or order of operations involved in it. You should break up the expression using derived data items (extensions).

021 INVALID TYPE OF SUBSCRIPT ITEM

The type in a subscript item must be numeric.

022 EQUAL OR NOT EQUAL OPERATOR REQUIRED

Certain operands, such as TRUE, FALSE, patterns, and conditions, require either "=" or "NOT =", or "EQUAL" or "NOT EQUAL" relational operators.

023 TOO MANY LEFT PARENS OR RIGHT PARENS EXPECTED

The number of left parentheses does not match the number of right parentheses.

024 INVALID EXPRESSION

The <expression> is invalid. Refer to the explanation of expression formation under EXPRESSIONS in Section 4.

025 INVALID USE OF CONDITION IN EXPRESSION

The condition must be related to a <data name> by residing on the right side of the relational operator. The relational operator must be "=" or "NOT =" (or "EQUAL" or "NOT EQUAL").

026 INVALID MIXTURE OF OPERAND TYPES IN EXPRESSION

Operand types in an <expression> must be compatible with the operations performed on them. Generally, operands of different types (numeric, alpha, or Boolean) should not be mixed. Care should be taken when using <number> or <string> operands.

027 INVALID ENTITY IN PATTERN

The pattern match condition was specified incorrectly. Refer to the explanation of the pattern match operand under EXPRESSIONS in Section 4.

028 STRING IS TOO LONG OR EMPTY STRING SPECIFIED

The <string> is too long, or no value [and/or blank space(s)] has been specified as the <string>. The acceptable <string> lengths and the resulting defaults are presented in Appendix B.

029 INVALID KEYWORD

An unrecognized keyword was given.

030 INVALID SELECT EXPRESSION

The SELECT statement needs to be corrected. The pattern match conditions, if any, may be in error, or the Boolean and arithmetic expressions may not be correct syntactically.

031 MISSING PERIOD -- ONE HAS BEEN ASSUMED

A period (.) was expected at this point in the statement, but none was given. The system has assumed a period at this point.

032 SELECT ALREADY SPECIFIED

More than one SELECT INPUT statement or more than one SELECT SAMPLED statement was specified. Only one of each is allowed in a single-report specification, in the Input Section of a multiple-report specification, or in the Report Section of a multiple-report specification.

033 UNRECOGNIZED OPTION SPECIFICATION

One of the option statements contains a reserved word not recognized by the system.

034 EXTERNAL-FILE-NAME TOO LONG

On the A Series of Systems, the maximum length of a particular file-name identifier is 17 characters. Excluding the ON clause, a maximum of 30 characters is allowed for an <external file name> including quotes and slashes.

On B 1000 Series of Systems, the maximum length of each part of the file identifier is 10 characters.

On the B 2000/B 3000/B 4000 Series of Systems, the maximum length of a file identifier is six characters.

035 MEMORY SIZE OUTSIDE ALLOWABLE RANGE

The integer value specified in the MEMORY SIZE option exceeds the maximum value allowed by the system. The maximum value for all systems is 9999999.

036 INTEGER EXPECTED

An <integer> was expected at this point in the statement, but none was given.

037 WORD OR STRING EXPECTED

A <word> or <string> was expected at this point in the statement, but none was given.

038 INVALID EXTERNAL-FILE-NAME

The format of the <external file name> was not valid.

039 CONSTRUCT NOT IMPLEMENTED

This construct is currently not within the scope of defined capabilities, or it is no longer an implemented construct. If the message is a warning (i.e., if WARN precedes the message number), the construct is ignored.

040 'SAVE SPECIFICATIONS' ALREADY SEEN

More than one SAVE SPECIFICATIONS statement was given; only one is allowed.

041 MISSING PERIOD

A period (.) was expected, but none was given.

042 VOCABULARY OR PASSWORD STATEMENT EXPECTED

A VOCABULARY statement or a PASSWORD statement was expected at this point in the report specification, but none was given. This message can occur if the statement does not start in column 8 or after.

043 VOCABULARY NAME EXPECTED

A valid vocabulary <external file name> was expected at this point in the statement, but none was given.

044 INVALID PASSWORD

The proper password was not given. The PASSWORD statement may be missing or not given in the proper sequence; an incorrect password may have been given; or the password given may be too long.

045 COMBINE ALREADY SEEN

More than one COMBINE statement was given; only one is allowed.

046 LIMIT ON NUMERIC LITERALS EXCEEDED

The limit on the number of <number>s allowed in an expression was exceeded. Refer to Appendix B: Limits and Defaults.

047 LIMIT ON NUMBER OF OPERATORS EXCEEDED

The limit on the number of operators in an expression was exceeded. Refer to Appendix B: Limits and Defaults.

048 LIMIT ON QUALIFICATION EXCEEDED

The limit on the number of qualifiers in a <data name> was exceeded. The maximum number of qualifiers permitted is indicated in Appendix B, Limits and Defaults.

049 INCOMPATIBLE SORT SPECIFICATION

Once a statement has specified ordering and/or control-break groupings, other statements can repeat the sorting specifications for purposes of clarity, but they cannot specify ambiguous sorting specifications.

050 CONTROL BREAKS ALREADY DEFINED

Control-break items were already defined by a previous statement. Their definition here is invalid.

051 INVALID CONSTRUCT IN EXTRACT STATEMENT

An invalid construct was specified in the EXTRACT statement. Refer to the discussion under EXTRACT STATEMENT in Section 4.

052 SUMMARY STATISTIC ALREADY SPECIFIED

The summary statistic item was already specified in the SUMMARIZE statement.

053 LIMIT ON REFERENCED ITEMS EXCEEDED

The limit on total number of <data names>, reported <strings>, and <statistic functions> referenced was exceeded. Refer to Appendix B, Limits and Defaults.

054 INVALID EXPRESSION

This error indicates the presence of a previous error.

055 STRING EXPECTED

A <string> was expected at this point, but none was given.

056 INVALID USE OF DERIVED ITEM

Derived items cannot be referenced in an INPUT statement.

057 INVALID WHERE CLAUSE

The WHERE <expression> does not evaluate to a proper Boolean value, or an error was present in the <expression>.

058 UNRECOGNIZED GROUP STATEMENT CONSTRUCT

An error exists in the syntax of the GROUP statement. Refer to discussion under GROUP STATEMENT in Section 4.

059 LIMIT ON CONTROL BREAKS EXCEEDED

The maximum number of permissible control breaks, which is nine, has been exceeded.

060 'TO' NOT SEEN IN EXTRACT STATEMENT

The EXTRACT statement must contain the word "TO" followed by a <file name>.

061 INVALID USE OF DERIVED ITEM

A derived item cannot be referenced at this point in the specification.

062 RIGHT PAREN EXPECTED

A right parenthesis, ")", was expected at this point in the statement, but none was given.

063 REPORT STATEMENT MUST BE PREVIOUSLY SPECIFIED

Columns can be footed only if they were defined for the report previously via a REPORT statement. In summary-only reporting, footing is not possible.

064 SCOPE DEFINED IN SUMMARIZE STATEMENT ALREADY SPECIFIED

Only one specification of footings and/or summary statistics for FINAL, in addition to specifications of footings and/or summary statistics for every control break, is permitted in the SUMMARIZE statement.

065 'EACH' EXPECTED, NOT FOUND

The syntax requires "FOR EACH <c-b name>".

066 COLON EXPECTED

A colon was expected at this point, but none was given.

067 INVALID CONTROL BREAK NAME

The <data name> specified was not a valid control-break name specified in a REPORT statement, a GROUP statement, or a RANGE statement.

068 ITEM MUST BE STATISTICAL

The item referenced must be statistical.

069 INVALID DUPLICATE REPORT ITEM

A <data item> cannot appear more than once as a report or group item. However, it may be redefined by an <extension>.

070 INCOMPLETE SPECIFICATION

The report specification is incomplete: a period may have been left off the last statement, or the last statement may be missing.

071 INVALID RANGE ITEM

The range item must not be a BOOLEAN item, and it cannot be specified previously as a range item or ordering key.

072 UNRECOGNIZABLE SUMMARIZE SYNTAX

An invalid SUMMARIZE statement syntax was encountered.

073 'FROM' EXPECTED

The word "FROM" was expected at this point in the statement, but none was given.

074 INVALID ACCEPTED-DATA-NAME

The <data name> must reference an accepted data item which was defined previously in the ACCEPT statement.

075 TYPE OF RANGE LIMIT DOES NOT MATCH RANGE ITEM

The range limit and range item must correspond in type (for example, both must be string, or both must be numeric).

076 SUMMARY LEVEL OF COLUMNS NOT COMPATIBLE

When columns represent summary information for a particular control break, footings can be specified only for higher-level control breaks.

077 DATA-NAME DOES NOT REFERENCE DATA ITEM

A <data name> used in the current context must reference a data item.

078 ALL 12 MONTHS ARE REQUIRED FOR THE ABBREVIATED MONTH LIST

The process option SET ABBREVIATED MONTH LIST requires that all 12 months be given.

079 LEFT PAREN EXPECTED

A left parenthesis "(" was expected at this point in the statement, but none was given.

080 UNRECOGNIZED INPUT CONSTRUCT

An invalid INPUT statement syntax was encountered. Refer to explanation of syntax under INPUT STATEMENT in Section 4.

081 (NOT IN USE)

082 INVALID FILE NAME

The <file name> referenced at this point was not a valid file name within the vocabulary.

083 INVALID INPUT DATA STRUCTURE NAME

A valid <file name>, data set name, or <input procedure> name was not given.

084 INVALID SELECTION EXPRESSION

The current <expression> is not a valid DMS II selection expression: the <data name>s must be valid DMS II keys, and the <expression> following "AT" must be a Boolean expression.

085 ILLEGAL VALUE SPECIFIED

An illegal value was specified. Refer to Appendix B, Limits and Defaults.

086 INVALID KEY ITEM

The key item is not the required type, or it is not an accepted item or an item in a data structure input previously at the same level as the system file.

087 LIMIT ON MAX DATA STRUCTURES EXCEEDED

The maximum number of data structures specifiable in the INPUT statement was exceeded. Refer to Appendix B, Limits and Defaults for the limit on the number of data structures specifiable in the INPUT statement.

088 BASE-DATE INTEGER GIVEN WAS LONGER THAN 5 DIGITS. A JULIAN FORMAT (YYDDD) WAS EXPECTED

The process option SET BASE-DATE requires that the integer given for the date be in Julian format which is YYDDD.

089 DDD PART OF THE JULIAN DATE GIVEN FOR THE BASE-DATE WAS NOT VALID

When the process option SET BASE-DATE is given, the Julian date is checked to verify that the DDD part is greater than 0 and not more than 366 for leap years and not more than 365 for non-leap years.

090 PICTURE NOT EXPANDABLE — TRUNCATION MAY RESULT

While attempting to expand a picture for summary reporting (notably, footed items), the size of the picture exceeded the maximum size allowed (see Appendix B, Limits and Defaults). The expanded picture is determined from the picture size of the original item, the integer size and fraction size, and the TOTAL-POPULATION settings.

091 OVERLAP OF TITLE SPECIFICATION

You cannot specify a title with the EACH PAGE clause if you specify a title(s) with the FIRST PAGE clause and/or OTHER PAGES clause. Either the EACH PAGE clause or the FIRST PAGE clause and/or OTHER PAGES clause must be used. Refer to explanation under TITLE STATEMENT in Section 4.

092 MACRO ERROR

An error has occurred in a macro definition or a macro call. The following are the specific error messages which can occur in association with the 092 MACRO ERROR message:

DUPLICATE FORMAL PARAMETER NAME

The parameters in the parameter list must be unique among themselves.

EXTRANEOUS #

An extraneous pound sign (#) was encountered.

IDENTIFIER PREVIOUSLY DECLARED

Macro names must be unique.

ILLEGAL IDENTIFIER

Literals and special characters cannot be used as macro names or formal parameters.

ILLEGAL MACRO INVOKE

A macro cannot define itself (for example, "REPLACE A BY A + B#." is invalid).

INCORRECT NO. OF ACTUAL PARAMETERS

The number of actual parameters in the macro call does not correspond to the number of formal parameters in the macro definition.

LEFT PAREN EXPECTED - '('

A left parenthesis was expected, but none was given.

MACRO DEFINE LIMIT EXCEEDED

The maximum number of active macros (i.e., macros in use), both temporary and permanent at any given point, is 50.

MACRO NESTING TOO DEEP

Macros can be defined to a nesting level 10 deep.

MACRO PARAMETER STACK LIMIT EXCEEDED

Macros can be nested 10 deep, and each macro can have a maximum of 10 parameters, thus permitting a parameter stack limit of 100.

MISSING KEYWORD - 'BY'

The word "BY" must separate the macro name from the macro text.

MISSING RIGHT PARENTHESIS - ')'

Left and right parentheses must correspond. A right parenthesis must mark the end of the list of formal or actual parameters.

TOO MANY FORMAL PARAMETERS

A maximum of 10 formal parameters is permitted.

NOT ALLOWED AS ACTUAL PARAMETER VALUE

The pound sign (#) cannot be used within an actual parameter.

093 ITEM MUST NOT BE STATISTICAL

A nonstatistical item must be given.

094 ALL 12 MONTHS ARE REQUIRED FOR THE FULL-LENGTH MONTH LIST

When the process option SET FULL-LENGTH MONTH LIST is used, all 12 months must be given.

095 INPUT STRUCTURE ALREADY SPECIFIED

An input data structure can be specified only once. Therefore, a statement such as "INPUT FILE1, FILE2, FILE1." is not permitted.

096 LIMIT ON CUMULATIVE LENGTH OF ITEM HEADERS EXCEEDED

The internal buffer, which holds the headers for items being reported, overflowed.

097 INVALID NULL SPECIFICATION

An invalid null specification has been made. Refer to the syntax under PROCESS-OPTION SET STATEMENT in Section 4.

098 INVALID ITEM LENGTH

The size of the item as specified in <internal attributes> is invalid.

099 INCOMPATIBLE RELEASE VERSION

The release version of the program which created the vocabulary is incompatible with the release version of the REPORTER III Report Language Analysis Program. The last release letter should be consulted.

100 # NOT FOUND IN MACRO

The pound sign (#) marks the end of a macro definition. One was not detected; therefore, all subsequent report specifications were taken as macro text.

101 STATEMENT ALREADY SEEN

Only one statement of this kind is allowed. One such statement was already specified.

102 INVALID EXTRACT ITEM

The extract item must be a nonsubscripted vocabulary item, an accepted data item, a derived data item, or a <string>.

103 INVALID HARDWARE DEVICE

An invalid hardware device was specified. Check for possible spelling errors. Also refer to the syntax for the EXTRACT statement in Section 4.

104 EXPRESSION MUST NOT BE STATISTICAL

A nonstatistical expression must be given.

105 UNRECOGNIZED REPORT STATEMENT CONSTRUCT

The syntax for the REPORT statement is invalid. Refer to the syntax diagram under REPORT STATEMENT in Section 4.

106 INVALID DUPLICATE EXTRACT ITEM

A <data name> cannot appear more than once as an extract item. It can be redefined as a derived <data name> as explained under EXTRACT STATEMENT in Section 4.

107 'ASC' OR 'DES' ALREADY SEEN -- IGNORED

More than one sorting sequence was specified for an item. The second and succeeding sorting commands are ignored.

108 FILE-NAME NOT PREFIXED BY 'EF-'; SYNTAX ERRORS MAY OCCUR

Under certain circumstances, a file name which was not prefixed by an 'EF-' can result in a syntax error. For example, a COBOL syntax error occurs if a file name is the same as a COBOL reserved word, a REPORTER III reserved word, or a name in the vocabulary input to this report.

109 HARDWARE AND ID NOT COMPATIBLE -- HARDWARE WILL BE CHANGED

Hardware and ID specification can become incompatible if, for example, hardware device DISKPACK is specified and the ID does not specify PACK, or if hardware device DISK is specified and the ID specifies PACK.

110 INVALID 'FOR' STATEMENT GIVEN

The FOR statement requires either the word INPUT or REPORT and a terminating colon (:).

111 SORTING SPECIFIED WITH HARDWARE DEVICE PUNCH

When the hardware device specified is PUNCH, sorting is not permitted in an EXTRACT statement specified in a Report Section.

For B 1000 Systems:
112 KEY NAME ALREADY SPECIFIED

In a B 1000 DMS II selection expression, a key name can be assigned a value only once.

For B 1000 and B 2000/B 3000/B 4000 Systems:
113 INFORMATION FROM ONE AND ONLY ONE DATABASE MAY BE INPUT

The INPUT statement that was given violates the rule that only one DMS II data base can be input at one time.

114 OBJECT OF A VIA CLAUSE MUST BE A SET

If a DMS II VIA clause is used, it must refer to a DMS II set or link. The DMS II VIA clause here does not make such a reference. (Note that links are not applicable in B 1000 DMS II.)

115 A STANDARD EMBEDDED DATA SET MUST HAVE A VIA CLAUSE

The "INPUT" of a standard embedded DMS II data set was specified without the required VIA clause. DMS II requires that standard embedded-data sets be accessed via a link or spanning set. (Note that the term "standard" means "ordered" and that links are not applicable in B 1000 DMS II.)

116 SET DOES NOT SPAN THE DATASET GIVEN

Input was specified for a DMS II data set VIA a set which does not span the data set.

For A Series Systems:
117 LINK DOES NOT REFERENCE THE DATASET GIVEN

Input was specified for a DMS II data set VIA a link which does not reference the data set.

For A Series Systems:
118 SET DOES NOT CONTAIN KEY DATA

Input of a DMS II data set VIA a set with "KEY-DATA ONLY" was specified, but the set named has no key data.

119 CLAUSE INCOMPATIBLE WITH FILE ORGANIZATION

The clause cannot be used to describe access to the referenced system file.

120 PAGE-LENGTH TOO SMALL TO PRINT REPORT

The specified (or defaulted) page length is not large enough to print the report because one of the following conditions exists:

1. The number of specified TITLE lines, control-break heading lines, and/or column heading lines makes it impossible to print one line of column information on a page.

2. The number of specified TITLE lines and/or column heading lines makes it impossible to print all summary information for a control break or for FINAL on a page.

This error condition can be corrected by increasing PAGE-LENGTH or decreasing the number of TITLE lines and/or heading lines.

121 RESIDENT DATA SET NOT SPECIFIED AMONG PRIOR INPUT DATA STRUCTURES

Input of a DMS II data set was specified in a manner such as to make it dependent on a data set which was not among the data structures whose input was specified previously. This can occur in three situations in which access to the two data sets in a one-to-one corresponding fashion is specified:

1. The data set is embedded in the missing data set.

2. Input of the data set is to be VIA a link embedded in the missing data set.

3. Input of the data set is to be VIA a subset embedded in the missing data set.

122 ONLY ONE BUILD STATEMENT IS ALLOWED PER SECTION

For a single report, only one BUILD statement is allowed. For multiple reports, one BUILD statement is allowed in the Input Section and one in each Report Section.

123 A USER DEFINED NAME OR 'SKIP' WAS EXPECTED

At this point within the BUILD statement, something other than a name or the word SKIP was found. Compare your syntax for the BUILD statement with the syntax diagram for the statement.

124 PREFIX OR SUFFIX ALREADY SEEN FOR THIS ITEM

The PREFIX or SUFFIX clause has already been specified for this report item. Combine the separate PREFIX or SUFFIX clauses into a single PREFIX or SUFFIX clause.

125 MISSING KEYWORD 'DUPLICATES'

The AT clause of the DMSII structure clause is being used. The keyword WITHOUT was found, but the keyword DUPLICATES was not found.

For B 1000 Systems Only:

126 AT SYMBOLIC KEY EXPRESSION CLAUSE REQUIRED

An attempt is being made to access a data set via a retrieval (index random) set. Retrieval sets require use of the AT clause.

127 DATA NAME IS NOT AN ELEMENTARY ITEM

The data-name referenced in the FROM clause of the BUILD statement is not an elementary item. Refer to the vocabulary listing to determine which items are elementary items.

128 MORE THAN 12 MONTHS WERE GIVEN FOR THE ABBREVIATED MONTH LIST

RP3REP found more than 12 months listed while loading the months in the process option SET ABBREVIATED MONTH LIST.

129 1-1 RESIDENT STRUCTURE NOT AT CURRENT LEVEL

Access to a DMS II embedded data set was specified without previously specifying access to the owner data set at the same level of input.

130 TOTAL SIZE OF DATA-NAMES IS LARGER THAN TOTAL SIZE OF NAMES AND SKIPS. ADDED FILLER OF <integer>

The total size of all the data names referenced is larger than the total size of all of the names and SKIPs used in the BUILD statement. A FILLER (an implied SKIP clause) of <integer> size is being added to match the total size of the data names.

131 TOTAL SIZE OF THE DATA-NAMES IS LESS THAN TOTAL SIZE OF NAMES AND SKIPS. THE DIFFERENCE IS <integer>

The total size of all the data names referenced was less than the total size of all of the names and SKIPs used. The integer supplied in the message is the difference between the two sizes.

132 INVALID USAGE SPECIFIED

The usage specified in the USAGE IS clause of the EXTRACT statement must be DISPLAY.

133 STRING GIVEN FOR ABBREVIATED MONTH LIST WAS GREATER THAN 3 CHARACTERS

While loading the months in the process option SET ABBREVIATED MONTH LIST, RP3REP found an abbreviated month string which was longer than 3 characters.

134 MORE THAN 12 MONTHS WERE GIVEN FOR THE FULL-LENGTH MONTH LIST

RP3REP found more than 12 months listed while loading the months in the process option SET FULL-LENGTH MONTH LIST.

135 STRING GIVEN FOR FULL-LENGTH MONTH LIST WAS GREATER THAN 10 CHARACTERS

While loading the months in the process option SET FULL-LENGTH MONTH LIST, RP3REP found a month string which was longer than 10 characters.

136 DATA-NAME HAS SIGN. THE ABSOLUTE VALUE WILL BE USED.

The data-name in the FROM clause of the BUILD statement is a signed, elementary numeric item. If numeric items of this type are used in the FROM clause, only the absolute value of the numeric item will be available to the BUILD name that precedes the FROM clause.

137 - 138 (NOT IN USE)

139 SEMICOLON EXPECTED - ';'

A semicolon is needed to terminate the PRESELECT clause.

140 (NOT IN USE)

141 DATA-NAME CANNOT BE A DERIVED ITEM

The data name given in the PRESELECT clause was a derived item. Only vocabulary data-items from a previously referenced data structure and accepted data-items may be used in the PRESELECT clause.

142 EXPRESSION MUST BE BOOLEAN

The expression must be a Boolean expression.

143 THIS ITEM CANNOT BE USED IN PRESELECT CLAUSE

The data name used in the PRESELECT clause was not a data-item name, group name, condition name, or an accepted data-item.

144 - 145 (NOT IN USE)

146 INVALID 'FOR' EXPRESSION

The FOR expression is not valid and cannot be evaluated properly.

147 DATA-NAME IS STATISTICAL. IT MAY NOT BE USED HERE.

The data name in the FROM clause of the BUILD statement was a statistical item. For a single report, statistical items may not be used. For multiple reports, statistical items in the Input section may be used in a Report Section if the data name has been redefined. Refer to the explanation of the BUILD statement in Section 4 for the recommended procedure.

148 CONFLICT IN IMPLICIT AND EXPLICIT FROM

One-to-one corresponding access to two data structures related by an association was specified such that access to the second data structure depends on the results of the access to the first data structure. The FROM clause given was optional, since the system was able to determine the data structure "depended on" from the association used. The resident data structure (the one "depended on") which was given in the FROM clause did not match the resident determined by the system. The REPORTER III System determines residents according to the following table based on the type of the second (associated) data structure and the kind of association given:

Type of Second Data Structure	Type of Association	Resident Data Structure Determined by REPORTER III
DMS II data set	Link (reference)	The data set in which the link is embedded.
DMS II data set	Embedded set	The data set in which the set is embedded.
DMS II embedded data set	Embedding	The data set in which the data set is embedded.

149 - 152 (NOT IN USE)

153 NAME NOT A KEY OF THIS SET

In a B 1000 DMS II selection expression, a name was given that was not a valid key name for this set.

154 DUPLICATE NAME FOUND IN BUILD STATEMENT

A duplicate name was found after the FROM clause of the BUILD statement. A data name for an accepted data item or a derived data item (extension) can be used only once in a BUILD statement. To use a data name more than once, duplicate the data item using an extension. The extension must be defined before the BUILD statement.

155 NOT ENOUGH AVAILABLE SPACE ON A PAGE TO PRINT THIS ITEM: <data name>

The specified or defaulted PAGE-WIDTH is not sufficient to allow the printing of the identifier and the value of the referenced item. Either the identifier should be abbreviated, or the PAGE-WIDTH should be increased.

156 ITEM USAGE MUST BE DISPLAY OR COMPUTATIONAL

The NUMERIC class test requires the item to be DISPLAY or COMPUTATIONAL.

157 AGE INTERVAL EXPECTED, NONE FOUND

Aging can be done in DAYS, WEEKS, MONTHS, QUARTERS, or YEARS only.

158 INVALID <DATA-NAME>

The name specified does not reference either an elementary item, a group item, or a derived item.

159 VOCABULARY ELEMENT CANNOT BE ACCESSED

The element or field within the element was secured; therefore it is inaccessible.

160 DATA STRUCTURE OR INPUT PROCEDURE NAME EXPECTED

In the INPUT statement, an <input procedure> name or compound data structure clause list must be specified following the word INPUT. This error occurs if neither of these is found. In addition, a compound data structure clause must be composed of data structure clauses, each beginning with the name of a data structure.

This error occurs if a data structure clause is expected and no data structure name is found.

161 KEY ITEM NAME REQUIRED

If a system file is to be accessed using an actual key, the name of the key must follow immediately the word VIA in the system file data structure clause.

162 VOCABULARY KEY ITEM NOT WITHIN PROPER DATA STRUCTURE

If a system file is to be accessed using an actual or symbolic key, the key must be a data item in a data structure previously specified at the same input level as the system file.

163 INVALID VALUE SPECIFIED FOR INTEGER-SIZE, FRACTION-SIZE

AND/OR STRING-SIZE

This error occurs if either or both of the following constraints are not met:

1. The value of STRING-SIZE cannot be set to 0.

2. For A Series and B 2000/B 3000/B 4000 Series systems, the value of INTEGER-SIZE + FRACTION-SIZE cannot be set to 0 or to a value greater than 23. For B 1000 Series systems, the value of INTEGER-SIZE + FRACTION-SIZE cannot be set to 0 or to a value greater than 18.

164 THIS CLAUSE MAY NOT BE REPEATED

This clause or option cannot be specified more than once within the current statement, description, or clause.

165 INTEGER OR ACCEPTED DATA NAME IS EXPECTED

An integer or accepted data name must be specified.

166 INVALID TYPE OR USAGE

The item has an item type, numeric, or string which conflicts with its usage.

167 MISSING OR MISSPELLED KEY WORD

A keyword is missing or misspelled. Refer to appropriate syntax diagram.

168 ITEM NOT IN DATA STRUCTURE BEING SAMPLED

When a data structure is being sampled, an item in the strata Boolean expression either must be an accepted data item or must belong to the data structure being sampled.

169 THE FOLLOWING REAL NUMBER WILL BE CORRECTED TO CONFORM TO DECIMAL-POINT IS COMMA STANDARDS <integer part>.<faction part>

The vocabulary being used was found to have the option DECIMAL-POINT IS COMMA set. To use DECIMAL-POINT IS COMMA, the fractional part of a real number must be separated from the integer part by a comma. The decimal point will be changed to a comma before the number is passed to RP3GEN.

170 INVALID EDITING ATTRIBUTE

The only valid <editing attribute> is WITH PICTURE.

171 LIMIT ON NUMBER OF REPORT ITEMS EXCEEDED

The maximum number of report items which can be specified was exceeded. Refer to Appendix B, Limits and Defaults.

172 INVALID TYPE SPECIFICATION

An attribute type (string or numeric) was specified which did not match the type of the resulting expression.

173 ITEM SIZE EXCEEDS GENERATED REPORT PROGRAM CAPACITY

The value specified for the attribute size is greater than the maximum capacity the generated report program can handle. Process options INTEGER-SIZE, FRACTION-SIZE, and STRING-SIZE can be reset to increase program capacity.

174 MISSING KEYWORD - 'BY'

The keyword "BY" was expected at this point in the statement, but it was not found.

175 LIMIT ON SUB-EXPRESSIONS EXCEEDED

The number of sub-expressions allowed was exceeded. You can use extensions to eliminate the nesting of expressions required.

176 EXPRESSION DOES NOT DESCRIBE AN APPROPRIATE SUMMARY ITEM

One or more items within the expression either are not appropriate summary items or require "RELATED TO" clauses to declare them explicitly to be appropriate summary items.

177 STAT PASSES REQUIRED EXCEEDS LIMIT

The number of passes through the data required to calculate the statistical expression exceeds the limit (refer to Appendix B, Limits and Defaults). The expression can be obtained by placing it in the Report Section of a multiple-report specification, or by one or more applications of the EXTRACT WITH VOCABULARY capability.

178 FOOTED ITEM MUST BE A COLUMN

An item cannot be footed unless it is reported as a column and does not have any editing attributes or internal attributes assigned except through an Extension statement.

179 LIMIT ON NUMBER OF FUNCTIONS EXCEEDED

The maximum number of functions which can be specified in a report specification was exceeded. Refer to Appendix B, Limits and Defaults.

180 ILLEGAL 'REPORT FOR EACH RANGE' STATEMENT

The specified "REPORT FOR EACH RANGE" statement is invalid.

181 'REL TO' CLAUSE INCOMPATIBLE WITH 'FOR EACH'

The accumulation required for the statistic is incompatible with the scope of the statistic.

182 CONSTRUCT MAY ONLY BE SPECIFIED ONCE

The syntax only allows the construct to be given once; multiple specification of the construct is invalid.

183 <DATA-NAME> REQUIRED FOR ITEM TO BE SUMMARIZED

The statistical function requires the specification of a <data name> after the left parenthesis.

184 STATISTIC MAY ONLY SUMMARIZE NUMERIC ITEM

Item must be numeric. Only COUNT, MAXIMUM, and MINIMUM can be taken on string or Boolean items.

185 INVALID OR MISSING MATCH KEY DATA-NAME

If a system file is to be input matching another input data structure, the name of the match key <data item> must follow the word MATCHING, DESCENDING, or ASCENDING. When multiple match keys are used, keys specified after the first key must follow the AND which joins the match key clauses. Each match key must be a data item in the specified system file.

186 MISSING 'WITH' CLAUSE

In specifying input of a system file matching another input data structure, each match key specified must be followed by the word WITH and the name of the matching data item in the related data structure.

187 INVALID OR MISSING RELATED MATCH KEY DATA-NAME

In specifying match keys to be used to input a system file, a related match key corresponding to each key in the matched system file must be specified following the match key <data name> and the word WITH. All related match keys must be data items in the data structure with which the system file is being matched. Note the following:

1. If MATCHING was specified, this related data structure must be a data structure previously specified at the same or immediate outer input level.

2. If MASTER MATCHING was specified, this related data structure must be a data structure previously specified at the same input level.

188 MISSING KEYWORD — 'UNMATCHED' OR 'DUPLICATES'

To specify that unmatched or duplicate records are to be noted as exceptions, the word NOTING must be followed by the keyword UNMATCHED or DUPLICATES, respectively.

189 MATCHING CLAUSE IS REQUIRED WHEN 'MASTER' HAS BEEN SPECIFIED FOR SYSTEM FILE

To specify that a system file is to be input as the master matching file, the keyword MASTER and the remainder of the matching clause is required. MASTER cannot be specified alone.

190 INVALID TABLE IDENTIFIER - NUMBER OR NAME REQUIRED

The table name was not a <name> or <number>.

191 MISSING TABLE ENTRIES

"OTHERS" or "." (a period) was encountered prior to at least one defined table entry.

192 SLASH (/) REQUIRED

A slash must appear between left and right items specified in the TABLE statement.

193 INVALID TABLE ENTRY - NUMBER OR STRING REQUIRED

The left or right entry in the TABLE statement was not a <string> or <number>.

194 TABLE IDENTIFIER ALREADY EXISTS OR IS > 10 CHARACTERS

A table identifier for the table already appeared, or the table name was too long.

195 NUMBER OF TABLES EXCEEDS LIMIT OF 9

The maximum number of tables allowed in a report specification is nine.

196 TABLE REFERENCED DOES NOT EXIST

The table name appearing in the ENTRY function was not defined. A table must be defined before it is referenced.

197 TABLE ENTRY TYPE MISMATCH

This message can reflect any of the following errors:

1. The table entry items on the left side of the table are not all the same type.

2. The table entry items on the right side of the table are not all the same type

3. The item appearing in the <ENTRY function> to be matched during table referencing is not the same type as the left table entries.

198 NUMBER OF ENTRIES IN TABLE EXCEEDS 999

The table size limit was exceeded.

199 ASSIGNMENT TO TERMINAL INVALID IN BATCH MODE

A listing was assigned to terminal, or accepted data was assigned to terminal, but the processing mode of the generated report program is batch. Therefore, the assignment is invalid.

200 A LISTING ASSIGNMENT TO TERMINAL ALREADY GIVEN

One and only one listing can be assigned directly to a terminal.

201 SUBSCRIPT ITEM CANNOT BE STATISTICAL — <data name>

The <data name> references a statistical item which was used as a subscript. This is not allowed.

202 DERIVED ITEM DEFINED BUT NEVER USED — <data name>

The <data name> references an item which was defined but never used.

For example, a derived item X used only in COUNT(X) is considered as never used because COUNT(X) is calculated exactly as COUNT. X is not used or needed for the calculation. Similarly, any derived item

used only for calculating X is not used or needed for calculating COUNT(X).

203 FILE NOT ASSIGNED TO RANDOM ACCESS DEVICE

A VIA clause was given for a file which can only be accessed sequentially.

204 STRING VALUE MAY BE TRUNCATED OR EXTENDED WITH SPACES TO nnn CHARACTERS

The group is to be treated as a string type item of length nnn. The group value may be truncated or extended with spaces to nnn characters, depending on its actual size. The value nnn is the specified or defaulted STRING-SIZE. It can be reset (refer to PROCESS-OPTION SET STATEMENT in Section 4).

Except in the case where the group is used as a system file match key, group values are not truncated. That is, they are not truncated when used to access data structures as specified in the INPUT statement.

205 UNRECOGNIZED FORM ATTRIBUTE

The only allowable form attributes are FORM-WIDTH, FORM-LENGTH, START-POINT VERTICAL-SPACING, or FORM-TYPE.

206 LIMIT ON NUMBER OF NESTED <PRINT SPECIFICATIONS> EXCEEDED

A maximum of 20 levels of nesting is allowed in the <conditional print specifications> of the PRINT statement.

207 PRINT SPECIFICATION INVALID IN GIVEN CONTEXT

Certain <print specifications> cannot be used in HEADERS and FOOTNOTES; i.e., NF cannot be used in a FIRST FORM HEADER, and <data name> <string> cannot be used in an OTHER FORM FOOTNOTE.

208 UNRECOGNIZED PRINT SPECIFICATION

The <print specifications> given are unrecognizable.

209 INTEGER MUST BE BETWEEN 1 AND 132 INCLUSIVE

The specified integer must be a number between 1 and 132 inclusive.

210 INTEGER MUST BE BETWEEN 1 AND 9999 INCLUSIVE

The specified integer must be a number between 1 and 9999 inclusive.

211 INTEGER MUST BE BETWEEN 0 AND 99 INCLUSIVE

The specified integer must be a number between 0 and 99 inclusive.

212 ILLEGAL CHANNEL NUMBER

The channel numbers allowable are 1 through 11.

213 ILLEGAL LINE NUMBER

Line numbers must be greater than zero and less than or equal to the form length given in the Form Attributes specification.

214 STRING SIZE EXCEEDS FORM WIDTH

The size of the string being built in the <text> is greater than the form width.

215 STATEMENT INVALID WHEN USED WITH EACH FORM HEADER

If you have specified EACH FORM HEADER, you cannot use FIRST FORM HEADER or OTHER FORM HEADER subsequently.

216 STATEMENT INVALID WHEN USED WITH EACH FORM FOOTNOTE

If you have specified EACH FORM FOOTNOTE previously, you cannot use FIRST FORM FOOTNOTE or OTHER FORM FOOTNOTE.

217 EACH FORM HEADER ILLEGAL WITH FIRST OR OTHER FORM HEADER

If you have specified FIRST FORM HEADER or OTHER FORM HEADER previously, you cannot use EACH FORM HEADER.

218 EACH FORM FOOTNOTE ILLEGAL WITH FIRST OR OTHER FORM FOOTNOTE

If you have specified FIRST FORM FOOTNOTE or OTHER FORM FOOTNOTE previously, you cannot use EACH FORM FOOTNOTE.

219 SPECIFICATION RESULTS IN FORM OVERFLOW

The <print specification> results in form overflow. Either NOTING FORM OVERFLOW was indicated or form overflow is strictly prohibited (such as form overflow occurring while some HEADER or FOOTNOTE is being defined).

220 INTEGER GIVEN IN -INTO- SPEC OVERFLOWS THE LINE OR ILLEGALLY TRUNCATES THE ITEM

An INTO nn POSITIONS specification was given that was less than the size of the item's editing picture (whether user-defined or default), or there were not enough positions left in the line to do the insert. If INTO nn POSITIONS is not specified, an INTO nn POSITIONS is supplied internally, with nn equal to the size of the item's editing picture.

221 CURRENT PRINT POSITION UNDEFINED, -AT- SPEC ILLEGAL

The current print position must be defined before an AT LINE POSITION can be used. A <data name> <string> and most cases of <conditional print specification> result in the current print position becoming undefined.

222 THE -AT- SPEC INDICATES ILLEGAL LINE AND/OR POSITION NUMBERS

An AT LINE POSITION of an <insert> specifies insertion before the current line number, or before the current position number, or both.

223 DATA NAME IS NOT A BOOLEAN

The <data name> specified is not a BOOLEAN.

224 CONDITIONAL PRINT SPEC ILLEGAL IN A FOOTNOTE

A <conditional print specification> cannot be made in a footnote.

225 STRING, INTEGER OR COMMA EXPECTED

A <string>, an <integer>, or a comma is expected at this point in the specification.

226 ILLEGAL POSITION GIVEN

In a <text> specification, the position given is beyond the form length or before the current position.

227 PATTERNS WILL NOT BE PRINTED. MISSING EITHER "SET FORM-PATTERNS" OR "WITH PATTERN" CONSTRUCTS

For patterns to be printed, the following two specifications must be made:

SET FORM-PATTERNS TO nn —to indicate the number to print.

WITH PATTERN (<text>) —to define what a pattern looks like.

228 LINE OVERFLOW DETECTED

The noting of LINE-OVERFLOW was specified, and line overflow was found.

229 OVERFLOW FOOTNOTE RUNS INTO FIRST FORM FOOTNOTE

The end of the overflow form footnote writes into the first form footnote area.

230 OVERFLOW FOOTNOTE RUNS INTO OTHER FORMS FOOTNOTE

The end of the overflow form footnote writes into the other form footnote area.

231 OVERFLOW HEADER RUNS INTO PAGE HEADINGS

The overflow header begins in the page heading area.

232 AN -AT LINE- SPEC ILLEGALLY INSERTS INTO THE HEADER AREA

An -AT LINE- specification inserts into the area reserved for page headings.

233 AN -AT LINE- SPEC ILLEGALLY INSERTS INTO THE FOOTNOTE AREA

An -AT LINE- specification inserts into the area reserved for page footnotes.

234 UNEXPECTED END OF EXPRESSION FOUND

This error can be caused by the syntax errors found in the expression (for example, invalid/misspelled data names or operations).

235 FORM WIDTH PLUS START POINT IS GREATER THAN 132

The maximum line size for forms is 132 character positions. The line size for forms consists of the form width plus the start point. You can apportion the character positions as you desire, but the total number must be 132 or less.

236 HEADER AND FOOTNOTE AREAS OVERLAP

The area reserved for headers overlaps into the area reserved for footnotes. There is the possibility of a run-time exception.

237 NUMBER OF RECORDS PER AREA MUST BE MULTIPLE OF SORT-FILE BLOCKING.

RP3REP cannot determine if the number of records given in the SET SORT-FILE SIZE statement is a multiple of the number which will be used for SORT-FILE BLOCKING. If the SET SORT-FILE BLOCKING is not used, then the SORT-FILE BLOCKING will be determined from the value found in the vocabulary. Refer to the SET SORT-FILE BLOCKING statement in the Vocabulary Language User's Guide and the data-processing option SET SORT BLOCKING statement.

If the default value is used, then the number of records given in the SET SORT-FILE SIZE statement must be multiples of 10; for example, 10, 20, 30.

238 - 240 (NOT IN USE)

241 MAXIMUM NUMBER OF DIGITS ALLOWED IN PICTURE CLAUSE
EXCEEDED ITEM NAME: PICTURE:

In numeric picture clauses, a maximum number of 23 digits is allowed for the A Series and B 2000/B 3000/B 4000 Series systems, and a maximum number of 18 digits is allowed for the B 1000 Series systems. The numeric picture clause of the item to be printed has exceeded this limit.

242 UNRECOGNIZED INSERT SPECIFICATION - 'ZERO' EXPECTED

The <insert> specifications FORM-NUMBER WITH, PAGE-NUMBER WITH, or OVERFLOW-NUMBER WITH must be followed by ZERO, ZEROS, LEADING ZERO, or LEADING ZEROS.

243 (NOT IN USE)

244 (NOT IN USE)

245 DMS II FIELD ITEMS CAN ONLY BE USED IN QUALIFICATION

DMS II FIELD items cannot be reported on or used in calculations.

246 INVALID. REPLACED BY '=' CONDITION

A GREATER THAN or NOT LESS THAN condition was specified in the <access clause>, but STARTING was not specified previously. STARTING must be specified for these conditions to be valid.

247 1-TO-1 ACCESSING IMPLIED. 'STARTING' IGNORED; ONLY
FIRST RECORD SATISFYING CONDITION IS INPUT

A one-to-one access order is to be used. The first record which satisfies the specified condition is to be accessed.

248 SUBSCRIPT IS OUT OF BOUNDS - LESS THAN 1 OR EXCEEDS
MAXIMUM VALUE

An <integer> which is used as a subscript is not within the bounds of the array.

249 SUBSCRIPT MUST BE WITHIN THE BOUNDS OF THE ARRAY

A subscripted <data name> is specified in the INPUT statement. You must ensure that the subscript is within the bounds of the array.

250 THE NUMBER OF CONTROL BREAKS EXCEEDS LIMIT

The number of control breaks declared inclusively by the GROUP or RANGE statement and the REPORT statement has exceeded the limit of nine.

251 REPORTER LIMIT FOR SPECIAL DMS II ITEMS EXCEEDED
ITEM NAME:
DATA SET:

This message applies to REPORTER III on the A Series of Systems when DMS II control items (count, record type, and population) are referenced. In the COBOL report program, these items must be referenced using the format <data-set name> (<control-item name>). The maximum length of <data-set name> (<control-item name>), including all qualifiers, is 60 characters.

252 THE NUMBER OF ORDERING KEYS EXCEEDS LIMIT

The number of ordering keys has exceeded the limit of nine.

253 ONLY COBOL74 CODE IS GENERATED, NO ACTION IS TAKEN

For the REPORTER III System, only COBOL74 code is generated. The GENERATE statement cannot be used with REPORTER III.

254 THE ABSTRACT STATEMENT IS ONLY ALLOWED IN THE
INPUT SECTION OF A MULTIPLE REPORT

The ABSTRACT statement was found in either a single report or in a Report Section of a multiple report.

255 ONLY 'TAPE' WILL BE GENERATED FOR TAPE-7 AND TAPE-9

The specification 'TAPE-7' and 'TAPE-9' in the FILE-MOD clause of the EXTRACT statement will cause 'TAPE' to be generated in the report program source code.

256 LIMIT ON FRACTION LENGTH EXCEEDED, 6 DIGITS ONLY

The maximum number of digits after the decimal point allowed in a fraction number is six.

257 UPPER-BOUNDS/LOWER-BOUNDS ARE NOT AVAILABLE

The constructs UPPER-BOUNDS and LOWER-BOUNDS are invalid constructs in ANSI-74 and ANSI-85 COBOL when numeric data items are being referenced.

258 DECIMAL-POINT-IS-COMMA HAS BEEN SPECIFIED

DECIMAL-POINT-IS-COMMA has been specified in the vocabulary. All pictures defined for derived items and editing pictures for existing data items must use commas in place of decimal points and periods in place of commas for all numeric items.

259 (NOT IN USE)

260 (NOT IN USE)

EXCEPTIONS

Exceptions encountered during the execution of a generated report program are noted by the REPORTER III System. These exceptions reflect abnormalities in input data values in the report specification which did not suspend execution of the report program but which did affect the information produced in the report. The exceptions are recorded on an Exceptions Listing, the contents of which is discussed below.

For each exception, information about the cause of the exception and the program environment at the time it happened is given. Exception numbers are assigned to the exceptions in numerical order as they occur. The exception type and exception message identify the exception condition. Up to two "buckets" of information may be given for each exception to provide further clarification. Each of these exception buckets may contain a value relevant to the specific exception, an additional message, or other evidence of the condition that caused the exception. The call number for an exception provides an index which can be used to locate the source of the exception call in the COBOL listing of the generated report program.

In the case of multiple reports, a report number is given if the exception applies to one of the reports but not to all of the reports.

A list of data item values relevant to the processing environment at the time of the exception may be provided. A standard editing picture based on INTEGER-SIZE, FRACTION-SIZE, and STRING-SIZE is used for printing item values. (Refer to the discussion under PROCESS-OPTION SET STATEMENT in Section 4.) However, for string item values, a maximum of 55 characters is printed.

The following is a list of the types of exceptions. The type number, message, and exception bucket description(s) (if any) are presented for each type of exception.

Type 1 INVALID KEY ON WRITE

Bucket 1 contains the file name.

Bucket 2 contains the value of the actual key for a random file.

Type 2 INVALID KEY ON READ

Bucket 1 contains the file name.

Bucket 2 contains the value of the actual key for a random file.

Type 3 DMSII ERROR: DMSTATUS = <category mnemonic>

(<Category mnemonic> is defined in DMS II.)

For the A Series and the B 2000/B 3000/B 4000 Series of Systems, Buckets 1 and 2 contain the DMS II numeric attributes DMSTATUS(DMSTRUCTURE) and DMSTATUS(DMERRORTYPE), respectively, to further identify the error source and subcategory.

For B 1000 Series of Systems, Bucket 1 gives the data structure name.

Type 4 (NOT IN USE)

Type 5 ERROR IN CALCULATING RANGE LIMITS

Bucket 1 gives additional information as follows:

"UPPER LIMIT ERROR" indicates an ON SIZE ERROR while calculating the upper range limit. The limit is set by default to HIGH-VALUES.

"INCREMENT ERROR" indicates an ON SIZE ERROR while setting the range increment. The default increment is then zero.

"LOWER LIMIT ERROR" indicates an ON SIZE ERROR while calculating the lower limit. A default value of zero is used for the lower limit.

Type 6 ERROR IN CALCULATING STATISTICS

An ON SIZE ERROR occurred during the calculation. The function being calculated is assigned NULL values.

Bucket 1 gives the name of the function being calculated.

Bucket 2 gives the type of statistics for which the function primarily is used. Statistics whose calculations depend on this function are affected.

If the function was calculated at a control break, the data items listed reflect the first logical record for the next control-break grouping. If it was calculated on FINAL, the data items listed reflect the last logical record reported.

Type 7 ERROR IN CALCULATING EXPRESSION

An ON SIZE ERROR was encountered while evaluating an expression. The result of the expression is assigned NULL values.

Bucket 1 gives the name of the item which is to contain the result of the calculation. It can be a <data name> referenced in the report specification or an <internal data name> used by the generated program to reference the result of the expression. The data items listed reflect the logical record which was processing.

Type 8 ERROR IN ACCEPTED DATA

Bucket 1 gives further explanation of the error.

Bucket 2 contains the erroneous data being scanned (up to 60 characters). For accepted data assigned to a hardware device other than the system ODT, the erroneous data should be corrected and the report program run again. For erroneous data accepted from the system ODT, this exception is not noted; instead an appropriate message is displayed and an ACCEPT is issued requesting an on-site correction.

Type 9 UNMATCHED RECORD

If the matching capability was used to input a system file, and if "NOTING UNMATCHED" was specified, a Type 9 exception occurs when an unmatched record was encountered.

Bucket 1 gives the name of the system file containing the unmatched record.

The data items listed reflect the unmatched record and the record for which a match was sought.

Type 10 MATCH KEY OUT OF ORDER

A key used to input a system file with the matching capability has a value which is out of order. It can be a key of the matched system file or its related data structure.

Bucket 1 gives the name of the key.

Bucket 2 gives the previous value of the key, i.e., the value before the fault.

The data items listed reflect the record out of sequence.

Type 11 INVALID DATE ENCOUNTERED IN DATA

Bucket 1 gives further explanation of the error.

Bucket 2 contains the invalid date. The format of the date is either CALENDAR or JULIAN as specified by the message in Bucket 1.

Whenever aging or data conversion is specified and invalid dates are encountered, this exception occurs. Invalid dates in GREGORIAN, MILITARY, or CALENDAR format are displayed in CALENDAR format. Invalid dates are dates where the month is not between 1 and 12, and/or the day is not between 1 and the last day (numerically) in the month.

After the invalid date is displayed in the exception file, the working copy of the date is changed to January 1, 1999 and processing continues.

Type 12 PROGRAM CAPACITY EXCEEDED

INTEGER-SIZE, which is used to determine the size of program accumulators and intermediate results, is too small. As a result, an ON SIZE ERROR occurred.

Bucket 1 contains the COBOL data name involved.

Type 13 DUPLICATE RECORD IN MATCHED FILE

If the matching capability was used to input a system file, and if "NOTING DUPLICATES" was specified, a Type 13 exception occurs when a duplicate record is encountered.

Bucket 1 gives the name of the system file containing the duplicate record.

The data items listed reflect the duplicate record.

Type 14 ERROR IN RANDOM SAMPLING

An illegal value pertaining to a parameter was detected. Refer to Appendix B, Limits and Defaults.

Bucket 1 indicates which parameter was illegal and gives the reason.

Bucket 2 gives the actual value of the parameter indicated by Bucket 1.

Type 15 INTERNAL SYSTEM ERROR

This exception indicates the occurrence of a REPORTER III internal system error. You should contact the local Burroughs systems representative and save all listings and associated documentation related to this run.

Bucket 1 indicates the type of system error encountered.

Bucket 2 gives pertinent information relating to that specific error.

Type 16 SUMMARY ITEM VALUE INCONSISTENT

The following was detected: an item explicitly or implicitly related to a control break has more than one value associated with a given value of the control break.

Bucket 1 gives the name of the item.

Bucket 2 gives the inconsistent value detected.

Type 17 FORM PRINTING EXCEPTION

An exception condition was detected during the printing of forms (user-formatted report).

Buckets 1 and 2 give further explanation and additional information about the exception.

APPENDIX D

RESERVED WORDS

All reserved words of the REPORTER III report language are listed below in alphabetic order. Caution must be exercised when using report language reserved words as names in a report specification for the following reasons:

1. In certain contexts, such names can be confused as reserved words and cause syntax errors to be generated.

2. When a reserved word is used as a macro name, the word subsequently is no longer recognized as a reserved word.

3. If a name is specified which is the same as a subsequently specified keyword, that keyword becomes functionally inoperative.

If a vocabulary name is identical to a report language reserved word, it can be renamed by means of an Extension, for example, ACCT-AGE IS REDEF OF AGE.

If the beginning portion of a reserved word is underlined in the following list, that portion of the word is also a reserved word, as is any portion of the word which includes the underlined portion and adjacent character(s).

In the case of some reserved words, whether or not they can be abbreviated depends on where they are used (for example, "SPACES" can be abbreviated when used in the "NULL SPEC" syntax but not when used in the "PRINT SPECIFICATIONS" syntax). To avoid confusion, such reserved words are not underlined in the following list. All reserved words which cannot be abbreviated under any circumstances also are not underlined in the following list.

ABBREVIATED
ABSTRACT
ACCEPT
ACCEPTED-DATA
AGE
ALL
AND
APPROXIMATELY
AREAS
AREASIZE
AS
ASCENDING
ASSIGN
AT
AVERAGE
AVG

BACKUP
BASE-DATE
BL
BLANK
BLOCKING
BUILD
BY

CALENDAR-DATE
CENTURY-DATE
CHANGES
CHANNEL
COBOL
COLUMN-SPACING
COMBINE
COMPILE
CONTIGUOUS
COUNT

DATE
DAYS
DDDDD-DATE
DDMMMYY-DATE
DDMMYY-DATE
DECREASES
DELIMITED
DESCENDING
DEVICE
DISK
DISKPACK
DISPLAYING
DIV
DUPLICATES
DUPS

EACH
ELSE
ENTRY
EQL
EQUALS
EVERY
EXCEPTIONS
EXECUTION
EXTRACT
EXTRACT-VOCABULARY
EXTRACT-VOCAL

FALSE
FILE
FINAL
FIRST
FOLLOWS
FOOTING
FOOTNOTE
FOR
FORMS
FORM-LENGTH
FORM-NUMBER
FORM-PATTERNS
FORM-OVERFLOW
FORM-TYPE
FORM-WIDTH
FRACTION-SIZE
FROM
FULL-LENGTH

GENERATION
GEQ
GLOBAL-DATA
GREATER
GREGORIAN-DATE
GROUP
GROUPING
GTR

HARDWARE
HEADER

IDENTIFICATION
IF
IN
INCLUDING
INCREASES
INFORMATION
INPUT
INTEGER-SIZE
INTERCHANGE
INTO
IS

JULIAN-DATE

KEY
KEY-DATA

LEADING
LEQ
LESS
LINES
LINE-NUMBER
LINE-OVERFLOW
LIST
LISTING
LOGICAL-PAGE-FILE
LOWER-BOUNDS
LSS
LT

MASTER
MATCHING
MAXIMUM
MEAN-SQUARES
MILITARY-DATE
MINIMUM
MMDDYY-DATE
MOD
MODE
MONITOR
MONTH
MONTHS

NAME
NEQ
NEXT
NEW
NF
NL
NOT
NOTING
NULL
NULL-BOOLEAN-ITEMS
NULL-NUMERIC-ITEMS
NULL-STRING-ITEMS
NUMERIC

OBJECT
OCCURRENCES
OF
OFF
ON
ON-LINE
ONLY
OR
ORDER
OTHERS
OUT
OVERFLOW
OVERFLOW-NUMBER

PACK-ID
PACKNAME
PAGES
PAGE-LENGTH
PAGE-NUMBERS
PAGE-OVERFLOW
PAGE-WIDTH
PARAMETERS
PASSWORD
PATTERN
PERCENT
PICTURE
POSITION
POSITIONS
PREFIXED
PRESELECT
PRINT
PRINTER
PROCESSING
PUNCH

QUARTERS
QUOTES

RANDOMLY
RANGE
RANGES
READER
RECORDS
REDEFINITION
RELATED
REPLACE
REPORTS
ROUNDED
RUN
RUNNING-COUNT
RUNNING-TOTAL

SAMPLED
SAVE
SEED
SELECT
SEMANTICS

SET
SIZE
SORT
SORT-FILE
SOURCE
SPACES
SPECIAL
SPECIFICATIONS
SPO
STANDARD
START-POINT
STARTING
STD-DEVIATION
STRATA
STRING
STRING-SIZE
SUFFIXED
SUM-SQUARES
SUMMARIZE
SUMMARY
SUPPRESS
SYNTAX
SYSTEMATICALLY

TABLE
TAPE
TERMINAL
TEXT
THAN
THEN
TIME
TITLE
TO
TOP-OF-FORM-CHANNEL
TOTAL
TOTAL-POPULATION
TRUE

UNMATCHED
UNSORTED
UNTIL
UPPER-BOUNDS

VALUE
VARIANCE
VERTICAL-SPACING
VIA
VOCABULARY

WEEKS
WHERE
WHICH
WITH
WITHOUT

YEARS
YYDDD-DATE
YYMMDD-DATE

ZEROS

APPENDIX E

GLOSSARY

A glossary of terms used within this manual is provided in this appendix. The terms are listed in alphabetic order, and a description is given for each term.

Term	Description
Accepted Data Item	A data item defined in the ACCEPT statement. This data item serves as a run-time parameter for report specification. Its actual value is read in by the generated report program before processing begins.
Control-Break Item	A data item used to group the information to be reported. All logical records which contain the same value of the control-break item are grouped together for purposes of reporting the information and calculating statistics on the information.
Data Base	A collection of one or more data structures and the relationships between them.
Data Item	An elementary item of information, e.g., account balance. Data items in REPORTER III are referenced by <data name>s, <number>s, character <string>s, or <item desc>s.
Data Structure	A collection of records, e.g., a master accounts file. Access to information in data structures is specified in the report language by the INPUT statement. Valid data structures for REPORTER III include system files describable in COBOL and DMS II data set.

Derived Data Item

A data item described in terms of other data items via an <item desc>. The value of this data is derived from input items, accepted items, constants, or other derived data item. Combinations of arithmetic operations, logical operations, and REPORTER III intrinsic functions can be used to compute the derived item. A derived data item represents an extension of the logical record.

Editing Picture

A COBOL construct which describes how the value of an item of information is to be printed.

Exception

An unusual condition detected at run-time by the generated report program which warrants notification of the user.

Footing

The total of a printed column appearing beneath the printed column.

Form Overflow

A condition which occurs when a <print specification> given in the PRINT statement other than an unconditional NEW FORM specification causes printing to overflow the form, i.e., to go beyond the last printable line.

Group

A collection of related data items which could be viewed as a single data item, e.g., data comprising month, day, and year.

Input Data Item

A data item which is part of the information to be used for the report. For a single-report specification, input data items are described in the vocabulary and are defined by reference to the appropriate data structures in the INPUT statement. For a multiple-report specification, input data items are defined in the Input

Section by the INPUT statement and by extensions.

Input Procedure

User-written COBOL code defined to RP3VOC and included in a vocabulary, which is referenced in the INPUT statement of a report specification to provide access to data structures in a manner which is not provided by REPORTER III-generated input routines.

Item

An abbreviated term for data item. (Refer to Data Item in this glossary.)

Keyword

A reserved word required to complete the meaning of a language statement. A keyword has a specific functional, i.e., semantic, meaning.

Logical Record

A record or group of associated records which contains all pertinent information about a single entity, e.g., an account. Records comprising a logical record can come from a single data structure, multiple data structures, or a user-written procedure. Certain information within a logical record can contain null values indicating the absence of related information. The INPUT statement in a report specification specifies what information each record contains and how this information is accessed.

Macro

A definition which allows a name to represent a portion of language text. Reference to the macro name causes the defined language text to be substituted for the name. Macros can exist in the vocabulary or be defined in the report specifications via the REPLACE statement.

Nonstatistical Data Item

An input data item, accepted data item, or derived data item which is not defined in terms of any statistical function or statistical data item.

Nonstatistical Expression

An expression which does not contain any statistical functions or statistical data items. A nonstatistical expression can be of type arithmetic, string, or Boolean.

One-to-Many Access

One-to-many access is specified in the INPUT statement. In one-to-many access, multiple records of a data structure are accessed for each record of a related data structure.

One-to-One Access

One-to-one access is specified in the INPUT statement. In one-to-one access, one record of a data structure is accessed for each record of an unrelated data structure (independent access) or related data structure (dependent access).

Optional Word

A reserved word included in the language to improve the readability of a statement. The user may include or omit an optional word.

Range-Break Item

A control-break item which categorizes information into range groupings based on where an item within the logical record, i.e., the ranged item, falls into specified value ranges. Limits for these ranges and the ranged item itself are defined in the GROUP statement or RANGE statement.

Ranged Item

A ranged item is specified in a GROUP statement or RANGE statement together with associated value ranges. The values of a ranged item determine the values of the corresponding range-break item which categorizes information into

range groupings for purposes of reporting and summarization.

Record — A collection of related data items and/or groups (e.g., an account record). The term record is used to mean a record, list element, or member, depending on the term which is appropriate for the type of data structure under consideration.

Report — A report can be one of the following: an automatically formatted report (used for the REPORT, TITLE, and SUMMARIZE statements and the run summary), a user-formatted report (corresponding to the PRINT statement), a machine-readable extract file (specified using the EXTRACT statement), or an exception report (containing exceptions encountered).

Reported Information — Information which is to appear in a report.

Reserved Word — An English word or an abbreviation thereof which is part of the REPORTER III language.

Scope — The scope of a statistic is the group of information to be summarized. The scope is all reported information, e.g., FINAL, or the information associated with a particular control-break item value.

Seed — A statistical item used in generating random numbers. The seed is used in the random number function. The same seed should be specified when identical random numbers are desired for each run of the report program.

Semantics

Rules that describe which syntactically legal REPORTER III statements have valid meanings, and what those meanings are.

Statistical Data Item

A derived data item which is defined in terms of one or more statistical functions or other statistical data items. For example, the company payroll, when derived as the total salary of all employees, is a statistical data item.

Statistical Expression

An expression which contains one or more statistical functions or statistical data items. A statistical expression can be of type arithmetic, string, or Boolean.

Storage Picture

A COBOL construct which defines the internal attributes, and thus determines machine storage allocation for an item of information.

Strata

Strata is used in the singular sense to mean a subpopulation, or subset, of information to be sampled. Strata is used in the plural sense to mean more than one subpopulation, or subset, of information to be sampled.

Summary Information

Information related to a control-break item, or the sum total of all information reported. Summary information consists of one or more summary items.

Summary Item

A summary item contains summary information. A summary item has a unique value for a particular control-break item value or for all reported information.

Summary Statistics

Those statistical items which are printed via the SUMMARIZE statement.

Syntax	A set of rules that describe legally constructed REPORTER III language statements.
Terminal Backup	A terminal backup file is a disk file (analogous to a printer backup file) created by the generated report program. It is accessible only by On-Line REPORTER III and can be paged (viewed) at a terminal or printed on a line printer at the user's request.
Vocabulary	A dictionary of data descriptions and definitions created by RP3VOC and referenced by REPORTER III report-language specifications via the VOCABULARY statement.
Word	A combination of not more than 30 characters which can consist of the alphabetic characters A thru Z, the numeric characters 0 thru 9, and the hyphen (-). A word must contain at least one alphabetic character and cannot begin or end with a hyphen.

INDEX