Chapter 2. How to Request a Report

Chapter Table of Contents

Chapter 2. How to Request a Report

This chapter teaches you how to use Spectrum Writer control statements to request custom reports.

Spectrum Writer's language is non–procedural, which means you just describe the *result* you want, not the programming steps needed to do it. That means you can produce new reports in a matter of minutes, rather than days or weeks.

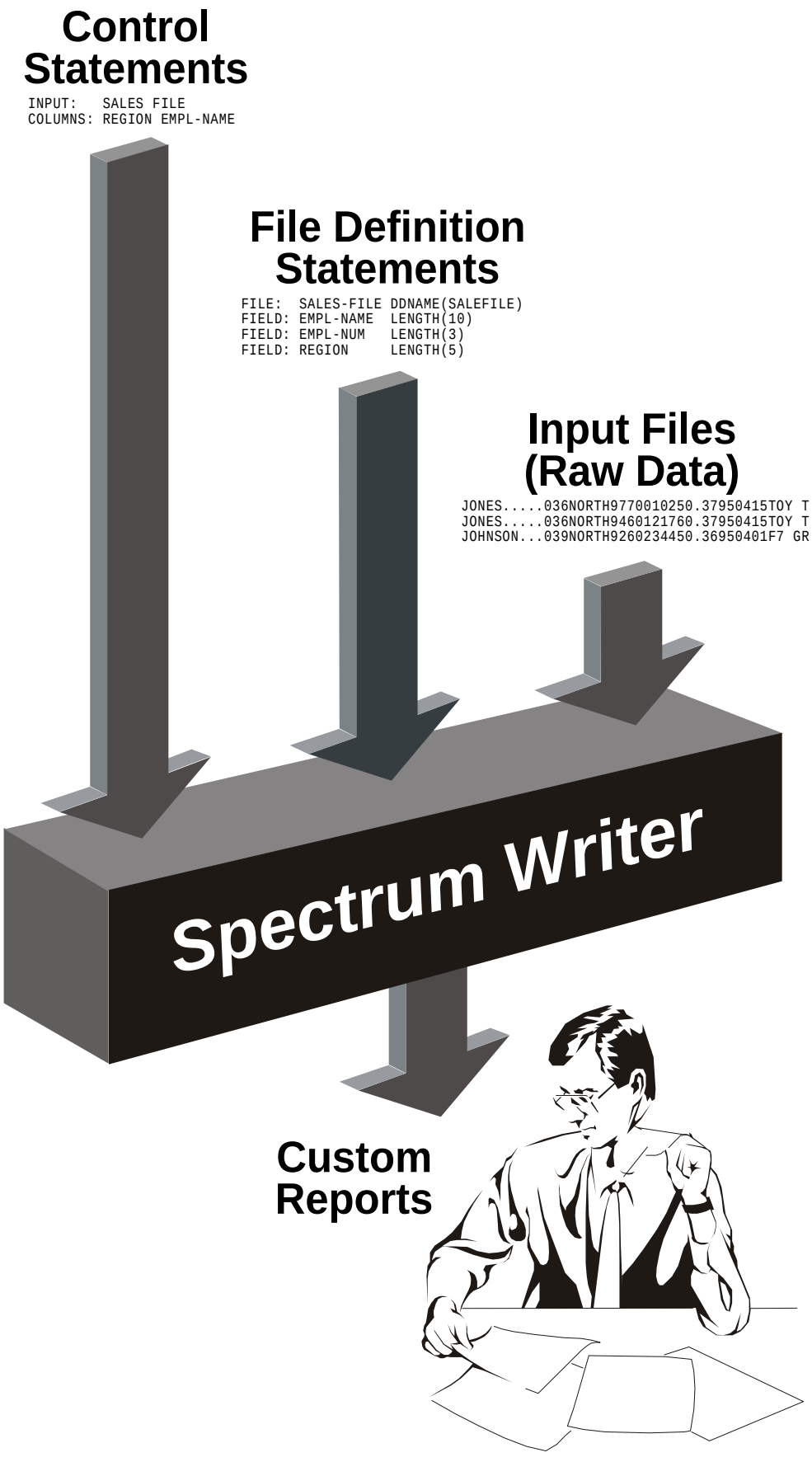
Describe your new report with a few simple "control statements". You can create a report with just *two* control statements. For example:

```
INPUT:    SALES-FILE
COLUMNS:  REGION  EMPL-NAME  SALES-DATE  SALES-TIME  CUSTOMER  AMOUNT  TAX
```

The above statements are all that is needed to produce a complete report with Spectrum Writer. (See **Figure 2** on page 36.)

The box on page 33 lists all of the Spectrum Writer control statements, and tells you which aspect of the report each one deals with. The lessons in this chapter illustrate how to use these control statements.

Once you've written the necessary control statements, submit a batch job to execute Spectrum Writer. Spectrum Writer examines the control statements describing the report you want. It also automatically reads the appropriate "file definition" statements stored in a copy library. (These statements define the input files needed for your report.) Spectrum Writer then accesses the input file(s) and prepares the desired report.


Control Statements
INPUT: SALES FILE
COLUMNS: REGION EMPL-NAME
File Definition Statements
FILE: SALES-FILE DDNAME(SALEFILE)
FIELD: EMPL-NAME LENGTH(10)
FIELD: EMPL-NUM LENGTH(3)
FIELD: REGION LENGTH(5)
Input Files (Raw Data)
JONES.....036NORTH9770010250.37950415TOY T
JONES.....036NORTH9460121760.37950415TOY T
JOHNSON...039NORTH9260234450.36950401F7 GR
Spectrum Writer
Custom Reports

The remainder of this chapter is divided into nine easy lessons that explain how to use Spectrum Writer's control statements to create custom reports. After reading just the first lesson, you will be able to produce useful reports with Spectrum Writer. The other lessons introduce additional control statements, and explain their roles in producing increasingly sophisticated reports. It is not necessary to read all of the other lessons initially. Nor is it necessary to read the lessons in sequential order. Read the summaries below and decide which lessons you need for the kind of reports you want to produce.

Lesson 1. How to Produce a Report in 5 Minutes.
This lesson shows how to produce reports using just two simple control statements — the INPUT and the COLUMNS statements. You will use these two statements for almost every report you request.

Lesson 2. How to Specify Which Records to Include in Your Report.
This lesson shows how to use the INCLUDEIF statement to select which records will appear in your report.

Lesson 3. How to Create Your Own Fields.
This lesson shows you how to create your own fields by performing computations on existing fields. This is done with the COMPUTE statement.

Lesson 4. How to Make Your Own Report Titles.
This lesson introduces the TITLE statement, and shows how you can specify your own report titles.

Lesson 5. Changing the Format of your Report.
This lesson shows how you can customize the appearance of your report. It introduces some of the parms available in the COLUMNS statement. These parms let you change: column headings; column width; and the way dates and numbers are formatted.

Lesson 6. How to Specify the Report Order.
This lesson shows how to sort your reports into whatever order you want. The use of the SORT statement is explained.

Lesson 7. How to Create Control Breaks.
This lesson shows how to break a report up into sections, printing subtotals for each section. The use of the BREAK statement to request such "control breaks" is explained

Lesson 8. How to Create Summary Reports.
This lesson shows you how to turn a report with subtotals into a "summary report."

Lesson 9. How to Use Data from More Than One File.
This lesson shows how easy it is to read records from additional files when producing a report. By adding a single READ statement, you automatically have access to all of the fields from an additional file.

Keep in mind that these lessons show you the most common use of each control statement. Most control statements also have additional features that are not discussed in these lessons. Additional ways to use these control statements are discussed in Chapter 4, "Beyond the Basics." The complete syntax for each control statement is shown in Chapter 10, "Control Statement Syntax."

SPECTRUM WRITER CONTROL STATEMENTS (GROUPED BY FUNCTION)

Statements that Define How Input Data Looks

FILE	Defines a file
FIELD	Defines a field within a file
ASM	Defines a file using an Assembler record layout
COBOL	Defines a file using a Cobol record layout
COMPUTE	Computes a new user-defined field

Statements that Specify the Input Files to Use for a Report

INPUT	Specifies the primary input file
READ	Specifies an auxiliary input file

Statements that Describe the Body of a Report

INCLUDEIF	Specifies which input records to include in the report
COLUMNS	Specifies the report columns and column headings
TITLE	Specifies the report titles
FOOTNOTE	Specifies footnotes at the bottom of each page

Statements that Define the Report Order and Control Breaks

SORT	Specifies report order and, optionally, specifies control break fields
BREAK	Specifies control break processing

Miscellaneous Statements

OPTIONS	Specifies various special options, such as double spacing, or summary reports
NEWOUT	Indicates that subsequent statements will define a new report
COPY	Copies additional control statements for processing

Figure 1. Spectrum Writer Control Statements Used for Making Reports

Lesson 1. How to Produce a Report in 5 Minutes

This lesson teaches you how to produce a complete report using just two simple control statements. These statements are:

- the INPUT statement
- the COLUMNS statement

You only need these two statements to create a report with Spectrum Writer. For example:

```
INPUT:    SALES-FILE
COLUMNS:  REGION  EMPL-NAME  SALES-DATE  SALES-TIME  CUSTOMER  AMOUNT  TAX
```

Figure 2 shows a report created with just these two statements.

How to Use the INPUT Statement

Your company probably has many files stored on its disk drives and magnetic tapes. For example, the personnel department of your company probably has an employee file, containing information about each employee. The accounting department probably has numerous files, such as an accounts receivable file, an accounts payable file, etc. A sales department might have a sales file, with information about sales that have been made, and so forth.

The very first step in requesting a report is to tell Spectrum Writer which one of your company's files has the data needed for your report. Use the INPUT statement to do this. For example:

```
INPUT:  SALES-FILE
```

The above statement tells Spectrum Writer that you want to use a file named SALES–FILE as the input for your report. SALES–FILE is a sample file that we will use for many examples in this manual. The SALES–FILE contains information about the sales made by the employees of an imaginary company. Each record in this file contains data about one sale.

All Spectrum Writer control statements begin in column 1 with the *name* of the statement (for example, INPUT), followed immediately by a *colon*. What follows next will depend on the particular control statement involved. With an INPUT statement, you simply put the name of the file to be used as the input for the report. In the above example, we named the SALES-FILE.

How to Use the COLUMNS Statement

After identifying the input file to use, the next step is to tell Spectrum Writer which *fields* from that file you want to see in your report. Use the COLUMNS statement to do that. Each field named in this statement will appear as one column of data in the report. For example:

```
INPUT:    SALES-FILE
COLUMNS:  REGION  EMPL-NAME  SALES-DATE  SALES-TIME  CUSTOMER  AMOUNT  TAX
```

The COLUMNS statement above tells Spectrum Writer that we want columns in our report that show the sales region, the employee name, the sales date, the sales time, the customer's name, the amount of the sale, and the tax amount.

> **Note:** Normally, reports are a maximum of 132 characters wide. You probably won't be able to fit *all* of a file's fields into that much space. Decide, then, which fields you need to see in your particular report, and put them in the COLUMNS statement. You may specify as many fields as there is room for in the report.

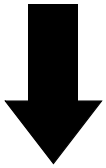
With just the two statements shown above, we have given Spectrum Writer everything it needs to produce a report. The report produced is shown in **Figure 2**.

You now see how easy it is to produce reports with Spectrum Writer. With just two simple statements we have produced an attractive report that has:

- a default **title** containing the name of the input file, as well as the date, time, day of the week, and page number
- the **columns of data** that we requested, appearing in the same order as we requested
- neat, underlined **column headings** identifying each column of data
- date, time and numeric fields that are properly **formatted**
- a **Grand Totals** line which shows totals for each of the numeric columns
- an **item count**, showing the number of records printed in the report

These Control Statements:

```
INPUT:   SALES-FILE
COLUMNS: REGION EMPL-NAME SALES-DATE SALES-TIME CUSTOMER AMOUNT TAX
```



Produce this Report:

```
TUE  05/16/95    8:25 AM     DATA FROM SALES-FILE                  PAGE     1

          EMPL     SALES    SALES
REGION    NAME     DATE     TIME     CUSTOMER          AMOUNT        TAX

SOUTH   JOHNSON    03/12/95 10:25:00 ACE ELECTRICAL        101.38       6.09
WEST    BAKER      03/26/95 12:09:09 JACKS CAFE            137.00       8.22
EAST    MORRISON   03/29/95 15:30:22 STAR MARKET            44.35       2.66
EAST    MORRISON   03/30/95 19:05:41 A1 PHOTOGRAPHY         29.65       1.78
EAST    SIMPSON    04/01/95 08:17:57 EUROPEAN DELI          14.99       0.90
NORTH   JOHNSON    04/01/95 17:02:47 VILLA HOTEL           234.45      14.07
NORTH   JOHNSON    04/05/95 14:33:10 MARYS ANTIQUES          9.98       0.60
WEST    BAKER      04/12/95 14:31:12 JACKS CAFE            135.75       8.15
WEST    THOMAS     04/14/95 15:41:38 YOGURT CITY             9.98       0.60
NORTH   JONES      04/15/95 07:58:32 EZ GROCERY             10.25       0.62
NORTH   JONES      04/15/95 08:01:59 TOY TOWN               121.76       7.31
NORTH   JONES      04/15/95 13:52:41 TOY TOWN                10.25       0.62
SOUTH   JOHNSON    04/16/95 11:48:33 ACME BUILDING          500.00      30.00
EAST    SIMPSON    04/30/95 15:30:21 J & S LUMBER            23.87       1.43

*** GRAND TOTAL (14 ITEMS)                               1,383.66      83.05
```

Remarks:

- this report was produced from just two statements: the INPUT and the COLUMNS statements
- the data used in this report comes from the SALES-FILE
- the seven columns of data in the report correspond to the field names in the COLUMNS statement
- the default column headings used are the field names themselves, broken apart at each dash
- the report has a default title which includes the name of the input file
- the report has a Grand Total line showing totals for the two numeric columns
- the number of items listed in the report is shown
- the JCL used to produce this report is shown on page 413 (OS/390) or page 428 (VSE)

Figure 2. A report produced with just two control statements

Another 5–Minute Report Example

Now let's make another report, this time using a different input file. This time we will request a report from the EMPL–FILE. That's a sample employee file. We will print a simple employee directory from this file. We want the report to have columns showing employee number, last name, first name, sex, social security number, date hired, and their city and state. We only need the following two statements:

```
INPUT:   EMPL-FILE
COLUMNS: EMPL-NUM  LAST-NAME  FIRST-NAME  SEX  SOCIAL-SEC-NUM
         HIRE-DATE  CITY  STATE
```

The INPUT statement above specifies that the input file for our report will be the employee file (EMPL–FILE). The COLUMNS statement specifies the columns of data we want our report to have. Notice that we needed two lines for the COLUMNS statement in this example. You can continue a control statement onto as many lines as you like. Just leave at least 1 blank space at the beginning of each continuation line.

The report produced by the above statements is shown in **Figure 3**.

You have now seen two examples showing just how easy it is to request a report with Spectrum Writer. That's all there is to it! You now know enough to request basic reports from the files at your company. Just specify the file you wish to use in your report with an INPUT statement. And then specify the fields that you want to see in the report with a COLUMNS statement.

Using Your Company's Files

You may be wondering how Spectrum Writer knows the names of *your company's* files and fields. The answer is that your company's files are defined to Spectrum Writer by other control statements that are kept in a Spectrum Writer "copy library." For example, the statements used to define the sample files used in the preceding examples are shown in Appendix F, "Files Used in Examples" (page 648).

For a list of the file names and field names available for you to use, ask your programmer. They can print that information from the Spectrum Writer Copy Library, in a format similar to that shown in Appendix F.

If you already know the name of the *file* to use, you can use a "dummy" run to easily get a list of all of its fields. Just use an INPUT statement with the SHOWFLDS(YES) parm, like this:

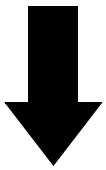
```
INPUT: SALES-FILE  SHOWFLDS(YES)
```

The above statement tells Spectrum Writer to print (in the control statement listing) a list of all of the fields defined for the SALES–FILE.

If a file that you need to use has not yet been defined, see Chapter 6, "How to Define Your Input Files" for information on doing that.

These Control Statements:

```
INPUT:   EMPL-FILE
COLUMNS: EMPL-NUM  LAST-NAME  FIRST-NAME  SEX SOCIAL-SEC-NUM
         HIRE-DATE  CITY  STATE
```



Produce this Report:

```
TUE  05/16/95    8:29 AM        DATA FROM EMPL-FILE                    PAGE     1

                                          SOCIAL
EMPL      LAST            FIRST             SEC       HIRE
NUM       NAME            NAME      SEX     NUM       DATE      CITY          STATE

036  JONES           JERRY            M  012-09-8765 01/31/80 SAN FRANCISCO    CA
037  JOHNSON         THOMAS           M  912-04-0334 06/21/75 SCOTTSDALE       AZ
039  JOHNSON         LINDA            F  004-77-9981 11/25/79 SANTA ROSA       CA
040  MACDONALD       RICHARD          M  889-79-0013 07/04/82 PLEASANTON       CA
041  SIMPSON         TIMOTHY          M  112-05-0456 12/01/82 ARCADIA          CA
042  MORRISON        MICHAEL          M  900-12-0556 11/30/79 GLENDALE         CA
043  CHRISTOPHERSON  MELISSA          F  415-09-0761 08/15/81 PHOENIX          AZ
044  BAKER           VIVIAN           F  878-19-0156 06/04/82 WALNUT CREEK     CA
045  THOMAS          MARTIN           M  776-83-8221 06/04/82 CONCORD          CA

*** GRAND TOTAL(9 ITEMS)
```

Remarks:

- the INPUT statement names the EMPL–FILE as the input file for this report
- the COLUMNS statement specifies which fields to print as columns in the report
- notice that we split the COLUMNS statement onto two lines, with the "continued" line beginning with at least one blank space

Figure 3. An employee directory produced with only two control statements

Summary

Here is a summary of what we learned in this lesson:

- an INPUT statement is needed to tell Spectrum Writer which input file to use for a particular report
- a COLUMNS statement is needed to tell Spectrum Writer what columns of data to print in your report
- by using just these two statements you can produce a complete report

The next lesson will teach you how to limit the records that are included in your report.

To Learn More

To learn more about writing control statements in general, see Chapter 9, "General Syntax Rules." In that chapter you will learn such things as:

- **how long** each control statement can be (page 443)
- how to **continue** control statements onto multiple lines (page 444)

There are some additional features associated with the INPUT and COLUMNS statements which we have not covered in this lesson. Some of these additional features are discussed in Lesson 5, "Changing the Format of your Report" (page 58). Other topics are discussed in Chapter 4, "Beyond the Basics." Some additional features are:

- how to specify your own **column headings** for a report (page 60 and page 130)
- how to make a column in the report **wider or narrower** (page 60 and page 135)
- how to change the way that **numbers, dates and times are formatted** in your report (page 58 and page 137)
- how to make a report column that contains a **literal text** (page 126)
- how to specify the number of **spaces** to leave between columns in your report (page 128)
- how to specify which numeric columns to include in the **Grand Totals** (page 148)
- how to print **multiple report lines** for each input record (page 151)
- how to **print all of the fields** from an input file in your report, without having to name each field individually (see page 158)
- how to produce reports that are **wider than 132 characters** (see page 417 or page 431)

The complete syntax for the INPUT and COLUMNS statements appears in Chapter 10, "Control Statement Syntax" (pages 542 and 498 respectively).

Lesson 2. How to Specify Which Records to Include in Your Report

This lesson teaches you how to select only certain records from the input file for inclusion in your report. The control statement discussed is:

- the INCLUDEIF statement

How to Use the INCLUDEIF Statement

The reports we produced in the previous lesson included all of the records found in the input file. When no INCLUDEIF statement is specified, Spectrum Writer defaults to including every record from the input file. For example, the report on page 36 included all sales from the SALES–FILE. And the report on page 38 listed all of the employees in the EMPL–FILE.

Often you want a report to include only selected records from the input file. Use the INCLUDEIF statement to tell Spectrum Writer to "include" a record in the report only "if" one or more conditions are met.

For example, assume that we want to print another list of sales from the SALES–FILE similar to the one on page 36. But this time we only want to print sales made by the employee named Jones. We would simply add the following INCLUDEIF statement to our other control statements:

```
INCLUDEIF:  EMPL-NAME = 'JONES'
```

The above INCLUDEIF statement tells Spectrum Writer to "include" records from the SALES-FILE "if" the EMPL–NAME field is equal to 'JONES'. Spectrum Writer still reads through the entire SALES–FILE, just like before. But now it examines each record before including it in the report. If the record's EMPL–NAME field contains the value 'JONES', then the record is included in the report. If the EMPL–NAME field contains any other value, then that record is not included in the report. **Figure 4** shows a report produced using the above statement. Only the sales made by Jones appear in that report.

The INCLUDEIF statement may appear anywhere after the INPUT statement. Only one INCLUDEIF statement is allowed per report, but it may contain as many conditions as you like.

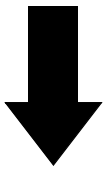
By the way, the INCLUDEIF statement can refer to any of the fields in the input file (as well as any COMPUTE field). You are not limited to just those fields that are listed in the COLUMNS statement.

How to Write Conditional Expressions

The INCLUDEIF statement simply contains a **conditional expression**. The complete rules for writing conditional expressions are explained beginning on page 459. Briefly, a conditional expression contains one or more "conditions," separated with words such as AND and OR. A **condition** usually involves comparing the contents of one field with the

These Control Statements:

```
INPUT:     SALES-FILE
INCLUDEIF: EMPL-NAME = 'JONES'
COLUMNS:   REGION EMPL-NAME SALES-DATE SALES-TIME CUSTOMER AMOUNT TAX
```



Produce this Report:

```
TUE  05/16/95    8:26 AM     DATA FROM SALES-FILE                  PAGE     1

          EMPL     SALES    SALES
REGION    NAME     DATE     TIME     CUSTOMER       AMOUNT        TAX

NORTH  JONES      04/15/95 07:58:32 EZ GROCERY          10.25        0.62
NORTH  JONES      04/15/95 08:01:59 TOY TOWN           121.76        7.31
NORTH  JONES      04/15/95 13:52:41 TOY TOWN            10.25        0.62

*** GRAND TOTAL (3 ITEMS)                              142.26        8.55
```

Remarks:

- the report now includes only those records whose EMPL–NAME field is equal to 'JONES'

Figure 4. Using an INCLUDEIF statement to specify which records to include in a report

contents of another field, or with a literal value. Let's look at some more examples of INCLUDEIF statements and their conditional expressions.

> **Note:** If you are a programmer, you will notice that the syntax for conditional expressions is very similar to the syntax used in "IF statements" in COBOL, PL/1, and BASIC. If you are familiar with any of these languages, you should find it especially easy to write INCLUDEIF statements.

You may want your report to include all records which **do not** contain a certain value. Do this by specifying "not equal" in your condition. For example:

```
INCLUDEIF:  EMPL-NAME ¬= 'JONES'
```

The above statement specifies that the report should include all records from the input file whose EMPL–NAME field is not equal to 'JONES'.

> **Note:** In addition to ¬=, you can also use <> to indicate "not equal," like this:
>
> ```
> INCLUDEIF: EMPL-NAME <> 'JONES'
> ```

You may want to include a record in your report if **either of two conditions** is true. To do this, use an INCLUDEIF statement with two conditions, separated by the word OR. Consider the following statement:

```
INCLUDEIF:  EMPL-NAME = 'JONES'  OR  AMOUNT > 100
```

The above statement states that a record should be included in the report "if the EMPL–NAME field is equal to 'JONES' or if the AMOUNT field is greater than 100." The word OR indicates that records from the input file will be included if either one (or both) of the conditions are true. **Figure 5** shows a report that uses the above statement. All sales listed in that report were either made by Jones or were for an amount over $100.

Notice in the above statement that we enclosed 'JONES' in single quotation marks, while we did not use quotation marks around the 100. That is because EMPL–NAME is a character field, while AMOUNT is a numeric field. Character literals (such as 'JONES') must be enclosed in quotation marks. You can use either single (') or double (") quotation marks. But numeric literals (such as 100), as well as date and time literals, are *not* enclosed in quotation marks. Numeric literals also must not contain commas. (The rules for writing literals are thoroughly explained in "How to Write Literals" on page 448.)

As another example, you may want to include records in your report when **both of two conditions** are true. For example, let's say we want a listing only of sales that were made by Jones and that were also for an amount over $100. For this report, two conditions must both be true: the EMPL–NAME field must be equal to 'JONES' *and* the AMOUNT field must be over 100. Use the word AND to specify that both conditions must be true, like this:

```
INCLUDEIF:  EMPL-NAME = 'JONES'  AND  AMOUNT > 100
```

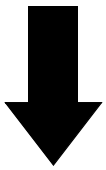
Now as Spectrum Writer reads each record from the input file, it will include a record in the report only "if the EMPL–NAME field is equal to 'JONES' and the AMOUNT field is greater than 100."

Here is an example of including records in a report based on the contents of a **date field**:

```
INCLUDEIF:  SALES-DATE  >  4/15/1995
```

These Control Statements:

```
INPUT:     SALES-FILE
INCLUDEIF: EMPL-NAME = 'JONES'  OR  AMOUNT > 100
COLUMNS:   REGION EMPL-NAME SALES-DATE SALES-TIME CUSTOMER AMOUNT TAX
```



Produce this Report:

```
TUE  05/16/95    8:26 AM     DATA FROM SALES-FILE                  PAGE     1

          EMPL     SALES    SALES
REGION    NAME     DATE     TIME     CUSTOMER        AMOUNT          TAX

SOUTH  JOHNSON    03/12/95 10:25:00 ACE ELECTRICAL      101.38         6.09
WEST   BAKER      03/26/95 12:09:09 JACKS CAFE          137.00         8.22
NORTH  JOHNSON    04/01/95 17:02:47 VILLA HOTEL         234.45        14.07
WEST   BAKER      04/12/95 14:31:12 JACKS CAFE          135.75         8.15
NORTH  JONES      04/15/95 07:58:32 EZ GROCERY           10.25         0.62
NORTH  JONES      04/15/95 08:01:59 TOY TOWN            121.76         7.31
NORTH  JONES      04/15/95 13:52:41 TOY TOWN             10.25         0.62
SOUTH  JOHNSON    04/16/95 11:48:33 ACME BUILDING       500.00        30.00

*** GRAND TOTAL (8 ITEMS)                            1,250.84        75.08
```

Remarks:

- records are included in the report if either the EMPL–NAME field is equal to 'JONES' or the AMOUNT field is greater than 100

Figure 5. Including records in a report if either of two conditions is true

The above statement specifies that records should be included in the report only if their SALES–DATE field contains a date greater than (after) April 15, 1995.

> **Note:** You may be wondering if you need to use a different format for your date literals when you know that a particular date field is stored in a record as a Julian date (YYDDD format.) The answer is no. All date literals in your control statements should be written as MM/DD/YYYY (or MM/DD/YY). Spectrum Writer automatically takes care of any date conversions that may be required. Thus, you test Julian date fields just like all other date fields:
>
> ```
> INCLUDEIF: JULIAN-START-DATE >= 1/1/2000
> ```

Here is an example of including records in a report based on the contents of a **time field**:

```
INCLUDEIF:  SALES-TIME  <  17:00:00
```

The above statement specifies that records should be included in the report only if their SALES–TIME field contains a time less than (before) 17:00:00 (which is 5 PM).

> **Note:** You are allowed to omit the seconds in your time literals, if you prefer. When the seconds are not specified, zero seconds is assumed. Thus, you could also write the above statement this way:
>
> ```
> INCLUDEIF:  SALES-TIME  <  17:00
> ```

If your INCLUDEIF statement contains both the words OR and AND, you should use parentheses to indicate the order in which to perform the comparisons. Consider the following statement:

```
INCLUDEIF:  EMPL-NAME = 'JONES'  OR
            (SALES-DATE > 4/15/1995  AND  SALES-DATE < 4/30/1995)
```

In the above statement, records will be included if the EMPL–NAME field is equal to 'JONES' or if both of the SALES–DATE comparisons are true. The parentheses cause the two SALES–DATE comparisons to be treated as one condition. That condition is true if the SALES–DATE is greater than April 15, 1995 and is less than April 30, 1995.

> **Note:** In addition to the actual words AND and OR, you can also use the symbols "&" and" |", respectively, in your conditional expressions.

Summary

Here is a summary of what we learned in this lesson:

- use the INCLUDEIF statement when you want to include only certain records from the input file in your report
- the INCLUDEIF statement may contain one or more conditions, separated by the words AND or OR
- groups of conditions can be enclosed in parentheses, to indicate the order in which the comparisons should be performed

The next lesson will show you how to compute your own new fields for use in your report.

To Learn More

There are some additional features associated with the INCLUDEIF statement which we have not covered in this lesson. These additional features are discussed in "Conditional Expressions" (page 459). The additional features include:

- how to use the keyword **NOT** (or the symbol ¬) to negate a condition (page 469)
- how to **scan** a character field, to see if a certain text exists *anywhere* within the field (page 460)
- how to specify conditions based on **bit fields** (page 465)
- how to specify a condition based on a field's raw **hexadecimal** value (page 464)
- what to do if you want to specify **date literals** in DD/MM/YY or DD/MM/YYYY format (page 140), like this:

  ```
  INCLUDEIF: SALES-DATE > 15/4/1995
  ```

- how the **KEYRANGE** or **STOPWHEN parms** of the INPUT statement can be used to limit the records included in your run (page 542)

The complete syntax for the INCLUDEIF statement appears in Chapter 10, "Control Statement Syntax" (page 540).

Lesson 3. How to Create Your Own Fields

This lesson teaches you how to create your own fields to use in producing your report. The control statement discussed is:

- the COMPUTE statement

Sometimes the data you need for a report is not contained in the input file. Yet the necessary data might be easily computed from one or more fields which *are* in the input file. In such cases, simply create a new field by using the COMPUTE statement.

Creating Numeric Fields

A COMPUTE statement specifies the name of the new field to create and supplies a *computational expression* to use in assigning a value to that field. The complete rules for computational expressions are discussed in "Computational Expressions" (page 472). Generally, your expression will consist of one or more arithmetic operations performed on numeric fields or numeric literals.

For example, the sample SALES–FILE has numeric fields named AMOUNT and TAX. We can use the COMPUTE statement to create a new field containing the total amount due just by adding those two fields together, like this:

```
COMPUTE:  TOTAL-AMOUNT = AMOUNT + TAX
```

The above statement creates a new field named TOTAL–AMOUNT. It is computed by adding the AMOUNT field and the TAX field together. Now that the TOTAL–AMOUNT field has been created, we can use that field in *any way* that other fields can be used. For example, a computed field can be used: as a column in the body of the report; in the report titles; as a sort field; as a control break field; as part of a conditional expression (in the INCLUDEIF statement); even as an operand in subsequent COMPUTE statements to create other fields.

Figure 6 shows a report that uses the above COMPUTE statement.

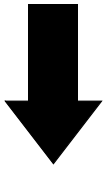
> **Note:** COMPUTE statements normally appear after the INPUT statement. A COMPUTE statement must appear before any other control statement that refers to the field being created. In **Figure 6**, the COMPUTE statement for TOTAL–AMOUNT had to come before the COLUMNS statement, since the COLUMNS statement referred to that field.

You can perform addition, subtraction, multiplication, and division in the COMPUTE statement. Use the +, –, * and / symbols, respectively. You may also use parentheses as needed to indicate the order in which the operations should be performed.

> **Note:** When performing subtraction, always put a **blank space before and after the minus sign**. Otherwise, the minus sign will appear to be part of a field name. Blanks are optional around the other arithmetic operators.

These Control Statements:

```
INPUT:   SALES-FILE
COMPUTE: TOTAL-AMOUNT     = AMOUNT + TAX
COMPUTE: SALES-COMMISSION = TOTAL-AMOUNT * .33
COLUMNS: EMPL-NAME  CUSTOMER  AMOUNT  TAX  TOTAL-AMOUNT  SALES-COMMISSION
```



Produce this Report:

```
TUE  05/16/95    8:26 AM        DATA FROM SALES-FILE                    PAGE     1

   EMPL                                                      TOTAL          SALES
   NAME     CUSTOMER          AMOUNT       TAX             AMOUNT      COMMISSION

JOHNSON    ACE ELECTRICAL        101.38        6.09          107.47         35.4651
BAKER      JACKS CAFE            137.00        8.22          145.22         47.9226
MORRISON   STAR MARKET            44.35        2.66           47.01         15.5133
MORRISON   A1 PHOTOGRAPHY         29.65        1.78           31.43         10.3719
SIMPSON    EUROPEAN DELI          14.99        0.90           15.89          5.2437
JOHNSON    VILLA HOTEL           234.45       14.07          248.52         82.0116
JOHNSON    MARYS ANTIQUES          9.98        0.60           10.58          3.4914
BAKER      JACKS CAFE            135.75        8.15          143.90         47.4870
THOMAS     YOGURT CITY             9.98        0.60           10.58          3.4914
JONES      EZ GROCERY             10.25        0.62           10.87          3.5871
JONES      TOY TOWN              121.76        7.31          129.07         42.5931
JONES      TOY TOWN               10.25        0.62           10.87          3.5871
JOHNSON    ACME BUILDING         500.00       30.00          530.00        174.9000
SIMPSON    J & S LUMBER           23.87        1.43           25.30          8.3490

*** GRAND TOTAL (    14 ITEMS)
                                1,383.66       83.05        1,466.71        484.0143
```

Remarks:

- the column heading used for computed fields is (by default) the field name itself, broken apart at each dash
- computed numeric fields receive Grand Totals just like other numeric fields

Figure 6. Using the COMPUTE statement to create numeric fields

As another example of a creating a numeric field, let's say we wanted to compute a sales commission for each sale. The commission will be 33% of the total value of the sale, including the tax. We could compute the sales commission with the following statement:

```
COMPUTE:  SALES-COMMISSION = TOTAL-AMOUNT * .33
```

This statement creates a new field called SALES–COMMISSION which is computed by multiplying TOTAL–AMOUNT by .33. Notice that we used the result of our previous COMPUTE statement to perform the computation in this statement.

Figure 6 (page 47) shows a report that uses the COMPUTE statement shown above.

In addition to the basic arithmetic operations, there are also a large number of built–in functions that you can use in the COMPUTE statement. These built–in functions allow you to perform more complex mathematical operations on numeric operands. A complete list of built–in functions is found in Appendix D, "Built-In Functions" (page 628).

Creating Character Fields

So far we have been creating numeric fields. Now let's consider how to create your own character fields. There is only one operation used in computing character fields. It is the **concatenation** operation. (Don't let that word scare you if it is new to you. "Concatenating" simply means "stringing together" two or more character fields.) The plus sign (+) is used as the symbol for concatenation. For example:

```
COMPUTE: WHOLE-NAME = LAST-NAME + FIRST-NAME
```

The above statement creates a new field named WHOLE–NAME. It is created by concatenating the contents of the LAST–NAME field and the contents of the FIRST–NAME field. The result is a single field which now contains both the first and last names of the employee. The new field will be 30 bytes long — the combined length of the two operands.

You can also concatenate more than two fields together. For example:

```
COMPUTE: MAILING-CODE = STATE + '-' + EMPL-NUM
```

This example creates a new field called MAILING–CODE which consists of the contents of the STATE field, followed by a dash, followed by the contents of the EMPL–NUM field.

In addition to the concatenation operation, there are also a number of built–in functions that can be used when creating character fields. For example, the #LEFT function can be used to extract the leftmost *n* bytes of a character field. Here is an example of how to use the #LEFT built–in function:

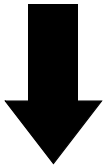
```
COMPUTE: FIRST-INITIAL = #LEFT(FIRST-NAME,1)
```

This statement creates a new character field which consists of only the first character (that is, the leftmost 1 byte) of the FIRST–NAME field.

Figure 7 shows a report that uses each of the above COMPUTE statements.

These Control Statements:

```
INPUT:    EMPL-FILE
COMPUTE:  WHOLE-NAME = LAST-NAME + FIRST-NAME
COMPUTE:  MAILING-CODE = STATE + '-' + EMPL-NUM
COMPUTE:  FIRST-INITIAL = #LEFT(FIRST-NAME,1)
COLUMNS:  EMPL-NUM  WHOLE-NAME  MAILING-CODE  FIRST-INITIAL  CITY  STATE
```



Produce this Report:

```
TUE  05/16/95    8:27 AM   DATA FROM EMPL-FILE                PAGE     1

EMPL             WHOLE               MAILING  FIRST
NUM               NAME                CODE    INITIAL      CITY        STATE

036  JONES          JERRY           CA-036     J    SAN FRANCISCO    CA
037  JOHNSON        THOMAS          AZ-037     T    SCOTTSDALE       AZ
039  JOHNSON        LINDA           CA-039     L    SANTA ROSA       CA
040  MACDONALD      RICHARD         CA-040     R    PLEASANTON       CA
041  SIMPSON        TIMOTHY         CA-041     T    ARCADIA          CA
042  MORRISON       MICHAEL         CA-042     M    GLENDALE         CA
043  CHRISTOPHERSON MELISSA         AZ-043     M    PHOENIX          AZ
044  BAKER          VIVIAN          CA-044     V    WALNUT CREEK     CA
045  THOMAS         MARTIN          CA-045     M    CONCORD          CA

*** GRAND TOTAL (9 ITEMS)
```

Remarks:

- the column heading used for computed fields is (by default) the field name itself, broken apart at each dash

Figure 7. Using the COMPUTE statement to create character fields.

Assigning Values to Fields Based on Conditions

Up until now we have been using "simple" COMPUTE statements. In a **simple COMPUTE statement**, the value of the new field is defined by a single computational expression.

But it is also possible to use conditional logic in a COMPUTE statement. In **conditional COMPUTE statements**, one of several different expressions will be used to assign a value to the new field. The expression that is used will depend on one or more conditions that you specify. Conditional COMPUTE statements can be very powerful tools in producing reports. Here is an example of a conditional COMPUTE statement:

```
COMPUTE: BONUS = WHEN(HIRE-DATE <  1/1/1980)  ASSIGN(TOTAL-SALES * .08)
                 WHEN(HIRE-DATE >= 1/1/1980)  ASSIGN(TOTAL-SALES * .05)
```

The above statement creates a field named BONUS. However, in this example the BONUS field can be computed in one of two ways: for employees hired before January 1, 1980, the bonus is 8 percent of total sales (TOTAL–SALES * .08). But, for employees hired on or after January 1, 1980, the bonus is only 5 percent of total sales (TOTAL–SALES * .05).

When assigning a value to the BONUS field, Spectrum Writer evaluates the conditional expression in each WHEN parm. As soon as a WHEN expression is found that is true, the computational expression from the corresponding ASSIGN parm is used to assign a value to BONUS. (Any remaining WHEN parms are not evaluated.)

You may have as many pairs of WHEN and ASSIGN parms as you like in a COMPUTE statement. If none of the WHEN expressions are true, a value of zero will be assigned to the field. To assign some other value when none of the WHEN parms are true, you may use the ELSE parm. For example:

```
COMPUTE: BONUS = WHEN(HIRE-DATE <  1/1/1980)  ASSIGN(TOTAL-SALES * .08)
                 ELSE                         ASSIGN(TOTAL-SALES * .05)
```

The above statement has the same effect as the previous example, but is a little simpler. It has only one WHEN expression. For employees whose hire date is before January 1, 1980, the bonus will be computed based on 8 percent. For all other cases, the bonus will be computed based on 5 percent.

You may also use conditional COMPUTE statements to create character fields. For example:

```
COMPUTE: TITLE = WHEN(SEX = 'M')  ASSIGN('MR')
                 ELSE             ASSIGN('MS')
```

The statement above creates a new field called TITLE. The contents of TITLE will be "MR" if the SEX field contains an "M", and "MS" otherwise.

Figure 8 shows a report that uses some of the conditional COMPUTE statements just discussed.

When defining character fields with a conditional COMPUTE statement, a value of spaces is assigned if none of the WHEN expressions are true and no ELSE parm is specified.

All of our examples so far have used just a single condition within the WHEN parm. You can, however, use any valid conditional expression within the WHEN parm. The conditional expression can contain as many different conditions as you like, separated with the words AND and OR, and optionally grouped with parentheses. (A conditional expression is the sort of expression that is allowed in the INCLUDEIF statement, as was described in the section

These Control Statements:

```
INPUT:    EMPL-FILE
COMPUTE:  BONUS = WHEN(HIRE-DATE <  1/1/1980)  ASSIGN(TOTAL-SALES * .08)
                  WHEN(HIRE-DATE >= 1/1/1980)  ASSIGN(TOTAL-SALES * .05)
COMPUTE:  TITLE = WHEN(SEX = 'M')  ASSIGN('MR')
                  ELSE             ASSIGN('MS')
COLUMNS:  TITLE  LAST-NAME  FIRST-NAME  SEX  HIRE-DATE  TOTAL-SALES  BONUS
```

Produce this Report:

```
TUE  05/16/95    8:29 AM        DATA FROM EMPL-FILE                      PAGE     1

           LAST            FIRST            HIRE        TOTAL
TITLE      NAME            NAME        SEX  DATE        SALES          BONUS

 MR   JONES           JERRY            M  01/31/80       42,509.89        2,125.4945
 MR   JOHNSON         THOMAS           M  06/21/75       86,999.24        6,959.9392
 MS   JOHNSON         LINDA            F  11/25/79       75,023.55        6,001.8840
 MR   MACDONALD       RICHARD          M  07/04/82        2,560.98          128.0490
 MR   SIMPSON         TIMOTHY          M  12/01/82        8,723.88          436.1940
 MR   MORRISON        MICHAEL          M  11/30/79       98,054.99        7,844.3992
 MS   CHRISTOPHERSON  MELISSA          F  08/15/81       47,665.31        2,383.2655
 MS   BAKER           VIVIAN           F  06/04/82       92,125.89        4,606.2945
 MR   THOMAS          MARTIN           M  06/04/82       60,193.49        3,009.6745

*** GRAND TOTAL (9 ITEMS)                                513,857.22       33,495.1944
```

Remarks:

- the BONUS field is calculated differently, depending on the contents of the HIRE–DATE field
- the value assigned to the TITLE field is based on the contents of the SEX field

Figure 8. Assigning values to computed fields based on conditions

"How to Write Conditional Expressions" on page 40. The complete rules for writing conditional expressions are given in "Conditional Expressions" on page 459.)

Additional examples of COMPUTE statements are shown beginning on page 506.

Summary

Here is a summary of what we learned in this lesson:

- the COMPUTE statement is used to create new fields
- a **simple COMPUTE statement** assigns the result of a single computational expression to a new field
- a **conditional COMPUTE statement** uses one of several different computational expressions, depending on the conditions that you specify

The next lesson will show you how to specify your own report titles.

To Learn More

There are some additional features associated with the COMPUTE statement which we have not covered in this lesson. Some of these additional features are discussed under the COMPUTE statement in Chapter 10, "Control Statement Syntax" (page 506). Other additional features are discussed in Chapter 4, "Beyond the Basics." Examples of additional topics include:

- how to create **date** fields (page 514)
- how to create **time** fields (page 272)
- how to create **bit (boolean)** fields (page 514)
- how to specify how many **decimal places** a numeric or time field should contain (page 511)
- how to specify **column headings** for the fields you create (page 511)
- how to specify how your field should be **formatted** when it is printed in a report (page 510)
- how to specify whether a numeric or time field should be totalled in the **Grand Totals** line at the end of the report (page 148)
- how to **retain the previous value** of a COMPUTE field in certain cases (page 234)

The complete syntax for the COMPUTE statement appears in Chapter 10, "Control Statement Syntax" (page 506).

Lesson 4. How to Make Your Own Report Titles

This lesson teaches you how to specify your own report titles. The control statement discussed is:

- the TITLE statement

How to Use the TITLE Statement

As we've seen in the previous lessons, a TITLE statement is not required to produce a report. If you do not supply a TITLE statement when requesting your report, Spectrum Writer provides a default title.

To specify your own report titles, simply use one or more TITLE statements. For each TITLE statement you supply, Spectrum Writer will print one title line at the top of each page of the report. TITLE statements may appear anywhere after the INPUT statement.

After the word TITLE and the colon, enclosed your desired title text in either single or double quotation marks. For example:

```
TITLE:  'ABC COMPANY -- RECENT SALES'
```

Note: If your title is too big to fit on a single line, you may continue it onto additional lines. See "How to Continue a Control Statement Onto Multiple Lines" (page 444) for information on continuing control statements.

You will probably want to include the date and page number in your titles. Do this by using the special built–in fields named #TODAY and #PAGENUM. (Don't let the pound sign scare you. All of Spectrum Writer's built–in field names begin with this character. This is to help distinguish them from fields in your own files that may have similar names.)

When using #TODAY and #PAGENUM in your TITLE statement, do not enclose them in quotation marks. Anything enclosed in quotation marks is printed *as is* in the title. Anything not within quotation marks must be the name of a field, whose *contents* you want in the title. The words #TODAY and #PAGENUM are the names of built-in fields, whose contents are the system date and the current page number. Here is an example of specifying titles that contain the date and page number:

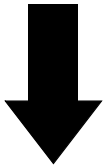
```
TITLE:  'ABC COMPANY -- RECENT SALES'
TITLE:  #TODAY
TITLE:  'PAGE' #PAGENUM
```

The three TITLE statements above result in three title lines in the report. The first title line is the literal text "ABC COMPANY – RECENT SALES". The second title line just contains the current date. The third title line contains the word "PAGE" followed by the page number itself. This third title line illustrates a new point: a TITLE statement can contain more than one item. In this case, it contains one literal text ('PAGE') and one field name (#PAGENUM).

Figure 9 shows a report produced using the above TITLE statements. Notice that the titles are automatically centered over the report.

These Control Statements:

```
INPUT:   SALES-FILE
TITLE:   'ABC COMPANY -- RECENT SALES'
TITLE:   #TODAY
TITLE:   'PAGE' #PAGENUM
COLUMNS: REGION  EMPL-NAME  SALES-DATE  SALES-TIME  CUSTOMER  AMOUNT  TAX
```



Produce this Report:

```
                        ABC COMPANY -- RECENT SALES
                                  12/01/95
                                 PAGE    1

          EMPL     SALES    SALES
REGION    NAME     DATE     TIME     CUSTOMER          AMOUNT      TAX

SOUTH  JOHNSON    03/12/95 10:25:00 ACE ELECTRICAL        101.38      6.09
WEST   BAKER      03/26/95 12:09:09 JACKS CAFE            137.00      8.22
EAST   MORRISON   03/29/95 15:30:22 STAR MARKET            44.35      2.66
EAST   MORRISON   03/30/95 19:05:41 A1 PHOTOGRAPHY         29.65      1.78
EAST   SIMPSON    04/01/95 08:17:57 EUROPEAN DELI          14.99      0.90
NORTH  JOHNSON    04/01/95 17:02:47 VILLA HOTEL           234.45     14.07
NORTH  JOHNSON    04/05/95 14:33:10 MARYS ANTIQUES          9.98      0.60
WEST   BAKER      04/12/95 14:31:12 JACKS CAFE            135.75      8.15
WEST   THOMAS     04/14/95 15:41:38 YOGURT CITY             9.98      0.60
NORTH  JONES      04/15/95 07:58:32 EZ GROCERY            10.25      0.62
NORTH  JONES      04/15/95 08:01:59 TOY TOWN              121.76      7.31
NORTH  JONES      04/15/95 13:52:41 TOY TOWN               10.25      0.62
SOUTH  JOHNSON    04/16/95 11:48:33 ACME BUILDING         500.00     30.00
EAST   SIMPSON    04/30/95 15:30:21 J & S LUMBER           23.87      1.43

*** GRAND TOTAL (14 ITEMS)                             1,383.66     83.05
```

Remarks:

- this report has three title lines, corresponding to the three TITLE statements
- the second title line simply contains the current date (#TODAY)
- the third title line contains the literal word "PAGE" followed by the page number (#PAGENUM)
- all title lines are centered over the report

Figure 9. Using the TITLE statement to specify your own titles

More Date and Time Features

When you use #TODAY in your title, Spectrum Writer formats it in the standard default date format (MM/DD/YY). If you want to **spell out the month name** in the date, specify the LONG1 "display format" after #TODAY, like this:

```
TITLE: #TODAY(LONG1)
```

The above statement would cause, for example, "DECEMBER 1, 1995" to appear in the title, rather than "12/01/95". The report in **Figure 10** uses the LONG1 display format. The use of LONG1 and other display formats is discussed in more detail in "How to Change the Appearance of Items in the Title" (page 165). For a complete list of display formats to choose from when formatting dates in your titles, see Appendix B, "Display Formats" (page 617).

In addition to the current date, you can also use the built–in fields #TIME and #DAYNAME in your TITLE statement. These allow you to print the time of day and the day of the week in your titles.

Figure 10 also illustrates the #TIME built–in field.

How to Align the Title

What if we want just a single title line that contains the date, time and the page number along with our literal text? The following example shows how to do that:

```
TITLE: #TODAY #TIME  /  'ABC COMPANY – RECENT SALES'  /  'PAGE' #PAGENUM
```

Notice that the above TITLE statement contains two **slashes** (/). These are used to separate the title line into three parts. When slashes are not used (as in the previous examples), the whole title is simply *centered* over the report. But when slashes are used, the first part of the title (#TODAY and #TIME, in the case above) is aligned with the *left* edge of the report. The middle part (the literal text) is *centered* over the report. The last part ("PAGE" and #PAGENUM) is aligned with the *right* edge of the report. The use of slashes in the TITLE statement gives you the maximum control over how your title lines look.

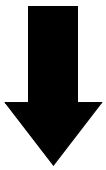
Figure 10 shows a sample report that uses slashes to align a three-part title. **Figure 14** (page 68) illustrates the alignment of a two-part title.

How to Put File Data in the Title

As mentioned earlier, TITLE statement text that is enclosed within quotation marks will appear "as is" in the title. You can also put field names (without quotation marks) in the TITLE statement. A field name can be one of the special built-in fields, such as #PAGENUM. Or it can be the name of a regular field from the input file (or even a COMPUTE field). When inserting the contents of a data field into the title, Spectrum Writer uses the data found in the first record used on the current report page. **Figure 14** (page 68) shows an example of including file data in a title.

These Control Statements:

```
INPUT:   SALES-FILE
TITLE:   #TODAY(LONG1)  #TIME  /  'RECENT SALES'  /  'PAGE'  #PAGENUM
COLUMNS: REGION  EMPL-NAME  SALES-DATE  SALES-TIME  CUSTOMER  AMOUNT  TAX
```



Produce this Report:

```
DECEMBER 1, 1995    8:27 AM      RECENT SALES                      PAGE    1

          EMPL     SALES    SALES
REGION    NAME     DATE     TIME     CUSTOMER        AMOUNT        TAX

SOUTH  JOHNSON    03/12/95 10:25:00 ACE ELECTRICAL      101.38        6.09
WEST   BAKER      03/26/95 12:09:09 JACKS CAFE         137.00        8.22
EAST   MORRISON   03/29/95 15:30:22 STAR MARKET         44.35        2.66
EAST   MORRISON   03/30/95 19:05:41 A1 PHOTOGRAPHY      29.65        1.78
EAST   SIMPSON    04/01/95 08:17:57 EUROPEAN DELI       14.99        0.90
NORTH  JOHNSON    04/01/95 17:02:47 VILLA HOTEL        234.45       14.07
NORTH  JOHNSON    04/05/95 14:33:10 MARYS ANTIQUES       9.98        0.60
WEST   BAKER      04/12/95 14:31:12 JACKS CAFE         135.75        8.15
WEST   THOMAS     04/14/95 15:41:38 YOGURT CITY          9.98        0.60
NORTH  JONES      04/15/95 07:58:32 EZ GROCERY          10.25        0.62
NORTH  JONES      04/15/95 08:01:59 TOY TOWN           121.76        7.31
NORTH  JONES      04/15/95 13:52:41 TOY TOWN            10.25        0.62
SOUTH  JOHNSON    04/16/95 11:48:33 ACME BUILDING      500.00       30.00
EAST   SIMPSON    04/30/95 15:30:21 J & S LUMBER        23.87        1.43

*** GRAND TOTAL (14 ITEMS)                          1,383.66       83.05
```

Remarks:

- the two slashes divide the TITLE statements into three parts
- the first part (the date and time) is left aligned over the report
- the second part (the name of the report) is centered over the report
- the third part (the page number) is right aligned over the report
- the LONG1 "display format" causes the month name to be spelled out in the date

Figure 10. Using slashes to align the different parts of a title

Summary

Here is a summary of what we learned in this lesson:

- use the TITLE statement to specify **your own titles** for a report
- if more than one TITLE statement is used, the title lines print in the **same order** in which the TITLE statements appear
- use Spectrum Writer's built–in fields to include the **date, time, day of the week, and page number** in your titles
- use slashes to separate your title into **left, center, and right** aligned parts

The next lesson will teach you how to customize the formatting of your report.

To Learn More

There are some additional features associated with the TITLE statement which we have not covered in this lesson. Some of these additional features are discussed as topics in Chapter 4, "Beyond the Basics." Some additional features include:

- how to change the way the **dates, times and numbers are formatted** in the title (page 165)
- how to use **any combination** of left aligned, centered, and right aligned title parts (page 168)
- how to include the **jobname** in your title (page 163)
- how to print "footnotes" at the **bottom** of each page of the report (page 175)

The complete syntax for the TITLE statement is given in Chapter 10, "Control Statement Syntax" (page 602).

Lesson 5. Changing the Format of your Report

This lesson teaches you how to specify your own formatting options for a report. The formatting options discussed are:

- display formats
- column headings
- column widths

Using Display Formats

Spectrum Writer provides many "display formats" that you can choose from when displaying fields in a report. A complete list of display formats is found in Appendix B, "Display Formats" (page 617). When no display format is specified (as in most of the examples in the previous lessons), Spectrum Writer uses a default format. To specify your own display format, just place it in parentheses after the appropriate field name. (Do *not* leave a space between the field name and the open parenthesis.) Display formats are allowed in most statements. For example:

```
TITLE:   #TODAY(LONG1)
COLUMNS: SALES-DATE(SHORT3)  SALES-TIME(HH-MM)   AMOUNT(DOLLAR)
```

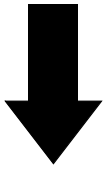
The above statements specify a display format for each field:

- the #TODAY field (in the title) will be formatted in Spectrum Writer's LONG1 format (that is, as MMMMMMMMM DD, YYYY).
- the SALES–DATE column in the report will be formatted in the SHORT3 format (that is, DD MMM YY).
- the SALES–TIME field will be formatted in the HH–MM format (that is, without the seconds). The time will be rounded to the nearest minute and formatted as HH:MM.
- the AMOUNT field will be formatted as a dollar value, with a floating dollar sign

Figure 11 shows a report that illustrates these display formats.

These Control Statements:

```
INPUT:   SALES-FILE
TITLE:   #TODAY(LONG1)  /  'EXAMPLES OF SPECIAL FORMATTING'  /  #PAGENUM
COLUMNS: REGION   EMPL-NAME('SALES PERSON')  SALES-DATE(SHORT3)
         SALES-TIME(HH-MM)  CUSTOMER  AMOUNT(DOLLAR)  TAX(5)
```



Produce this Report:

```
DECEMBER 1, 1995     EXAMPLES OF SPECIAL FORMATTING                    1

                      SALES   SALES
REGION SALES PERSON   DATE    TIME   CUSTOMER        AMOUNT       TAX

SOUTH   JOHNSON     12 MAR 95 10:25 ACE ELECTRICAL          $101.38  6.09
WEST    BAKER       26 MAR 95 12:09 JACKS CAFE              $137.00  8.22
EAST    MORRISON    29 MAR 95 15:30 STAR MARKET              $44.35  2.66
EAST    MORRISON    30 MAR 95 19:06 A1 PHOTOGRAPHY           $29.65  1.78
EAST    SIMPSON     01 APR 95 08:18 EUROPEAN DELI            $14.99  0.90
NORTH   JOHNSON     01 APR 95 17:03 VILLA HOTEL             $234.45 14.07
NORTH   JOHNSON     05 APR 95 14:33 MARYS ANTIQUES            $9.98  0.60
WEST    BAKER       12 APR 95 14:31 JACKS CAFE              $135.75  8.15
WEST    THOMAS      14 APR 95 15:42 YOGURT CITY               $9.98  0.60
NORTH   JONES       15 APR 95 07:59 EZ GROCERY               $10.25  0.62
NORTH   JONES       15 APR 95 08:02 TOY TOWN                 $121.76  7.31
NORTH   JONES       15 APR 95 13:53 TOY TOWN                  $10.25  0.62
SOUTH   JOHNSON     16 APR 95 11:49 ACME BUILDING            $500.00 30.00
EAST    SIMPSON     30 APR 95 15:30 J & S LUMBER              $23.87  1.43

*** GRAND TOTAL (14 ITEMS)                                 $1,383.66 83.05
```

Remarks:

- The display formats (LONG1, SHORT3, HH–MM and DOLLAR) specify how the data is formatted in the report
- The override column heading changes the column heading for the EMPL–NAME field
- The override width parm makes the TAX column only 5 bytes wide
- Changes made to the detail line formatting are also reflected in the Grand Total line

Figure 11. Using override display formats, column headings and column widths

Specifying Column Headings

Another way to customize your report is with override column headings. You remember that Spectrum Writer uses the field name itself as the default column heading. To specify your own column heading, just place the desired text in parentheses after the appropriate field name in the COLUMNS statement. For example:

```
COLUMNS: EMPL-NAME('SALES PERSON')
```

In the above statement, we specified our own column heading for the EMPL–NAME field. As you can see in the report in **Figure 11** (page 59), the EMPL–NAME column now has "SALES PERSON" as its column heading.

> **Note:** To break your column heading text into multiple lines, use the vertical bar (|) as a line separator. For example:
>
> ```
> COLUMNS: EMPL-NAME('SALES|PERSON')
> ```

The above statement would result in a two-line column heading for the EMPL-NAME column. The word SALES would be stacked over the word PERSON.

> **Note:** The vertical bar is the "Shift 1" key on most mainframe terminals. When working at a PC running terminal emulation software, you will probably not see a key with this symbol on it. (The "pipeline" character is *not* the same as the vertical bar.) Some emulator programs use the right–hand square bracket key (]) to send a vertical bar to the mainframe.

Specifying a Column's Width

One other way to customize your report is to specify a column width for a particular column. When no column width is specified, Spectrum Writer chooses a default column width. You may want a larger column width (to hold larger numeric values, for example). Or, you may want a smaller column width (to save space so you can squeeze more columns into your report). Just specify the desired column width in parentheses after the field name. For example:

```
COLUMNS: TAX(5)
```

The above statement tells Spectrum Writer to make the TAX column just 5 bytes wide in the report. This is also illustrated in the report in **Figure 11** (page 59).

Multiple Overrides

You can specify more than one override for a single field. Their order within the parentheses is not important. Just separate the overrides with spaces and/or commas. For example, the following statement specifies an override column heading, display format and column width:

```
COLUMNS:  AMOUNT('AMOUNT OF SALES', DOLLAR, 8)
```

Summary

Here is a summary of what we learned in this lesson:

- use an override **display format** to change the way a field is formatted in a report
- use override **column headings** to change the column headings in a report
- specify a **column width** to change the width of a column in a report
- each of these overrides should be **put in parentheses** after the appropriate field name

The next lesson will teach you how to sort your report into whatever order you want.

To Learn More

There are many additional ways to change the format of your report. Some of these additional features are discussed as topics in Chapter 4, "Beyond the Basics." Some additional formatting features include:

- how to **left-justify, center or right-justify** data within its column (page 146)
- how to **blank out repeating values** in a column(page 144)
- how to **blank out zero values** (page 129)
- how to change the **spacing between columns** in a report (page 128)
- how to use a **character other than the vertical bar** (|) to separate column headings into multiple lines (page 130)
- how to change the default display format for **all fields** in a report (page 562)
- how to format reports using **international conventions** (page 140)

The complete syntax for the COLUMNS statement is given in Chapter 10, "Control Statement Syntax" (page 498).

Lesson 6. How to Specify the Report Order

This lesson teaches you how to sort your report into any order you want. The control statement discussed is:

- the SORT statement

How to Use the SORT Statement

When no SORT statement is specified, Spectrum Writer defaults to printing the report records in their original input file order. For example, the records in the sample SALES–FILE are stored in sales date order. Therefore, the sales reports in the previous lessons (for example, on page 59) all appeared in sales date order. The EMPL–FILE sample file is a VSAM file stored in EMPL–NUM order. Therefore, all previous reports from that file have been in employee number order (for example, the report on page 38).

To print a report in a different order, just add a SORT statement. The SORT statement can appear anywhere after the INPUT statement. Only one SORT statement is allowed per report, but it may contain as many "sort fields" as you like. Spectrum Writer will sort your report on all of the sort fields.

For example, let's request a report from the SALES–FILE and sort it on three fields:

```
SORT:  REGION  EMPL-NAME  SALES-DATE
```

To begin with, the report will be sorted according to the first sort field — REGION. If there are multiple records for the same REGION, then those records will be further sorted using the second sort field, EMPL–NAME. Records having the same value for both the REGION and the EMPL–NAME fields will be further sorted on the third sort field — SALES–DATE.

Figure 12 shows a report produced with the above statement.

By the way, the SORT statement can refer to any of the fields in the input file (as well as any COMPUTE field). You are not limited to just the fields that are listed in the COLUMNS statement.

By default, Spectrum Writer sorts reports into **ascending order** on each sort field. If you want to sort the report into **descending order** for a field, put the DESCENDING parm (or just DESC) in parentheses immediately after the field name. For example, to sort a sales report into reverse employee number order, you could use this SORT statement:

```
SORT:  EMPL-NUM(DESC)
```

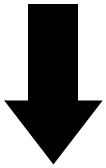
Automatic Sorting

If you prefer, you can let Spectrum Writer *automatically* sort your report for you. To have your report automatically sorted on the first 5 columns of data, simply specify the AUTOSORT option, like this:

```
OPTIONS: AUTOSORT
```

These Control Statements:

```
INPUT:   SALES-FILE
SORT:    REGION  EMPL-NAME  SALES-DATE
TITLE:   'RECENT SALES'
TITLE:   'SORTED BY REGION, EMPLOYEE NAME, AND SALES DATE'
COLUMNS: REGION  EMPL-NAME  SALES-DATE  SALES-TIME  CUSTOMER  AMOUNT  TAX
```



Produce this Report:

```
                                RECENT SALES
                SORTED BY REGION, EMPLOYEE NAME, AND SALES DATE

          EMPL     SALES    SALES
REGION    NAME     DATE     TIME     CUSTOMER          AMOUNT      TAX

EAST   MORRISON   03/29/95 15:30:22 STAR MARKET           44.35      2.66
EAST   MORRISON   03/30/95 19:05:41 A1 PHOTOGRAPHY       29.65      1.78
EAST   SIMPSON    04/01/95 08:17:57 EUROPEAN DELI        14.99      0.90
EAST   SIMPSON    04/30/95 15:30:21 J & S LUMBER         23.87      1.43
NORTH  JOHNSON    04/01/95 17:02:47 VILLA HOTEL         234.45     14.07
NORTH  JOHNSON    04/05/95 14:33:10 MARYS ANTIQUES        9.98      0.60
NORTH  JONES      04/15/95 07:58:32 EZ GROCERY           10.25      0.62
NORTH  JONES      04/15/95 13:52:41 TOY TOWN             10.25      0.62
NORTH  JONES      04/15/95 08:01:59 TOY TOWN            121.76      7.31
SOUTH  JOHNSON    03/12/95 10:25:00 ACE ELECTRICAL      101.38      6.09
SOUTH  JOHNSON    04/16/95 11:48:33 ACME BUILDING       500.00     30.00
WEST   BAKER      03/26/95 12:09:09 JACKS CAFE          137.00      8.22
WEST   BAKER      04/12/95 14:31:12 JACKS CAFE          135.75      8.15
WEST   THOMAS     04/14/95 15:41:38 YOGURT CITY           9.98      0.60

*** GRAND TOTAL (14 ITEMS)                          1,383.66     83.05
```

Remarks:

- the SORT statement causes the report to be sorted on REGION, EMPL–NAME and SALES–DATE

Figure 12. Using a SORT statement to specify the sort order of a report

Summary

Here is a summary of what we learned in this lesson:

- use the SORT statement to **sort your report**
- you can sort on **multiple sort fields**
- you can sort in either **ascending or descending** order

The next lesson will show you how to create control breaks and print subtotals and other statistics in your reports.

To Learn More

There are some additional features associated with the SORT statement which we have not covered in this lesson. Some of these additional features are discussed as topics in Chapter 4, "Beyond the Basics." Some additional features include:

- creating a **control break** with the SORT statement (page 177)
- specifying **control break spacing** with the SORT statement (page 178)
- requesting **totals and statistics** in the SORT statement (page 186)

The complete syntax for the SORT statement is given in Chapter 10, "Control Statement Syntax" (page 595).

Lesson 7. How to Create Control Breaks

This lesson teaches you what control breaks are, and shows how to request them in your report. This lesson also shows how to print totals and other statistics in reports. The control statement discussed is:

- the BREAK statement

How to Use the BREAK Statement

If you are not a programmer, the term "control break" may be new to you. But it is a very simple concept. And as you will see, control breaks can make your reports much more useful.

Consider the result of sorting a report on some field. By sorting the report on a field, we *group together* all the report lines that contain a particular value for that field. For example, in the report in **Figure 12** (page 63) we sorted first of all on the REGION field. As you can see, this caused the report lines to be grouped together by region. All of the report lines for the East region appear together at the beginning of the report. Next come all of the report lines for the North region, and so on. By sorting on the REGION field, we grouped together all of the records for each region.

Often it is desirable to perform special processing whenever one such group of records ends and another group is about to begin. For example, you might want to print a line of totals for the group that just ended. Or, you might want to print a few blank lines before the next group starts printing, or even skip to a new page. This processing is called **control break processing**. A **control break** is said to occur whenever one group of records ends and another group is about to begin. The field that is being grouped (for example, REGION) is called the **control break field** (or often just the **break field**). A control break field *must* also be a sort field, since it is by being sorted that records are grouped together in the first place.

You may designate any sort field as a control break field. Just name the field in a BREAK statement:

```
SORT:  REGION  EMPL-NAME  SALES-DATE
BREAK: REGION
```

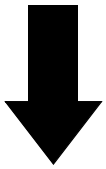
The above statement makes REGION a control break field. Now we will get REGION totals in the report whenever one region finishes printing and another region is about to begin.

After these totals, two blank lines will print. Then the report lines for the next region start to print, and so on.

Figure 13 shows a report that uses the above BREAK statement to produce a control break.

These Control Statements:

```
INPUT:   SALES-FILE
SORT:    REGION  EMPL-NAME  SALES-DATE
BREAK:   REGION
TITLE:   'RECENT SALES'
TITLE:   'TOTALLED BY REGION'
COLUMNS: REGION  EMPL-NAME  SALES-DATE  SALES-TIME  CUSTOMER  AMOUNT TAX
```



Produce this Report:

```
                              RECENT SALES
                           TOTALLED BY REGION

          EMPL     SALES    SALES
REGION    NAME     DATE     TIME     CUSTOMER          AMOUNT      TAX

EAST   MORRISON   03/29/95 15:30:22 STAR MARKET          44.35      2.66
EAST   MORRISON   03/30/95 19:05:41 A1 PHOTOGRAPHY       29.65      1.78
EAST   SIMPSON    04/01/95 08:17:57 EUROPEAN DELI        14.99      0.90
EAST   SIMPSON    04/30/95 15:30:21 J & S LUMBER         23.87      1.43
*** TOTAL FOR EAST  (4 ITEMS)                          112.86      6.77


NORTH  JOHNSON    04/01/95 17:02:47 VILLA HOTEL         234.45     14.07
NORTH  JOHNSON    04/05/95 14:33:10 MARYS ANTIQUES        9.98      0.60
NORTH  JONES      04/15/95 07:58:32 EZ GROCERY           10.25      0.62
NORTH  JONES      04/15/95 13:52:41 TOY TOWN             10.25      0.62
NORTH  JONES      04/15/95 08:01:59 TOY TOWN            121.76      7.31
*** TOTAL FOR NORTH (5 ITEMS)                          386.69     23.22


SOUTH  JOHNSON    03/12/95 10:25:00 ACE ELECTRICAL      101.38      6.09
SOUTH  JOHNSON    04/16/95 11:48:33 ACME BUILDING       500.00     30.00
*** TOTAL FOR SOUTH (2 ITEMS)                          601.38     36.09


WEST   BAKER      03/26/95 12:09:09 JACKS CAFE          137.00      8.22
WEST   BAKER      04/12/95 14:31:12 JACKS CAFE          135.75      8.15
WEST   THOMAS     04/14/95 15:41:38 YOGURT CITY           9.98      0.60
*** TOTAL FOR WEST  (3 ITEMS)                          282.73     16.97



****** GRAND TOTAL (14 ITEMS)                        1,383.66     83.05
```

Remarks:

- REGION is a sort field in this report
- the BREAK statement makes REGION a control break field
- whenever the value of the REGION column changes, a control break occurs
- at each control break a total line prints, followed by two blank lines

Figure 13. Using the BREAK statement to create a control break

How to Specify Control Break Spacing

You can use additional parms in the BREAK statement to customize your control break. For example, you can specify a **break spacing parm**. This parm tells Spectrum Writer what kind of spacing to perform at the control break. By default, Spectrum Writer prints two blank lines at each control break (after the totals line). You can use a spacing parm to request either a different number of blank lines, or to request a page break.

For example, the following statement makes REGION a break field and specifies that 3 blank lines should print at the control break:

```
BREAK:  REGION  SPACE(3)
```

If you want to skip to a new page whenever the contents of the REGION field changes, use the PAGE spacing parm, like this:

```
BREAK:  REGION  SPACE(PAGE)
```

The SPACE(PAGE) parm specifies that, rather than printing 2 blank lines whenever the REGION field changes, the report should skip to a new page.

The report in **Figure 14** illustrates the use of the PAGE spacing parm to request a page break.

How to Print Statistics at a Control Break

You may want to print statistics other than totals at a control break. The total line, as we have seen, prints automatically at control breaks. By supplying the appropriate parm in the BREAK statement, you can also print up to five additional statistical lines at a control break. These additional lines are:

- an **average** line
- a **non–zero average** line (the average of all non–zero values)
- a **maximum** line
- a **minimum** line
- a **non–zero minimum** line (the minimum non–zero value)

The parms that correspond to these statistical lines are:

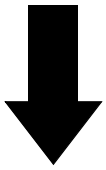
- AVERAGE (or AVG)
- NZAVERAGE (or NZAVG)
- MAXIMUM (or MAX)
- MINIMUM (or MIN)
- NZMINIMUM (or NZMIN)

You can specify as many of these parms as you like in the BREAK statement. The parms may be specified in any order. (The statistic lines in the report, however, will always print in a standard fixed order.) For example:

```
BREAK:  REGION  AVERAGE  MAXIMUM
```

These Control Statements:

```
INPUT:   SALES-FILE
SORT:    REGION  EMPL-NAME  SALES-DATE
BREAK:   REGION  SPACE(PAGE)
TITLE:   'SALES FOR REGION:'  REGION   /   'PAGE'  #PAGENUM
COLUMNS: REGION  EMPL-NAME  SALES-DATE  SALES-TIME  CUSTOMER  AMOUNT  TAX
```



Produce this Report:

```
SALES FOR REGION: EAST                                              PAGE    1

          EMPL     SALES    SALES
REGION    NAME     DATE     TIME     CUSTOMER          AMOUNT        TAX

EAST   MORRISON   03/29/95 15:30:22 STAR MARKET          44.35        2.66
EAST   MORRISON   03/30/95 19:05:41 A1 PHOTOGRAPHY       29.65        1.78
EAST   SIMPSON    04/01/95 08:17:57 EUROPEAN DELI        14.99        0.90
EAST   SIMPSON    04/30/95 15:30:21 J & S LUMBER         23.87        1.43
*** TOTAL FOR EAST  (4 ITEMS)                           112.86        6.77
```

```
SALES FOR REGION: NORTH                                             PAGE    2

          EMPL     SALES    SALES
REGION    NAME     DATE     TIME     CUSTOMER          AMOUNT        TAX

NORTH  JOHNSON    04/01/95 17:02:47 VILLA HOTEL         234.45       14.07
NORTH  JOHNSON    04/05/95 14:33:10 MARYS ANTIQUES        9.98        0.60
NORTH  JONES      04/15/95 07:58:32 EZ GROCERY           10.25        0.62
NORTH  JONES      04/15/95 13:52:41 TOY TOWN             10.25        0.62
NORTH  JONES      04/15/95 08:01:59 TOY TOWN            121.76        7.31
*** TOTAL FOR NORTH (5 ITEMS)                           386.69       23.22
```

```
SALES FOR REGION: SOUTH                                             PAGE    3

          EMPL     SALES    SALES
REGION    NAME     DATE     TIME     CUSTOMER          AMOUNT        TAX

SOUTH  JOHNSON    03/12/95 10:25:00 ACE ELECTRICAL      101.38        6.09
SOUTH  JOHNSON    04/16/95 11:48:33 ACME BUILDING       500.00       30.00

                        (other report lines not shown)
```

Remarks:

- the SPACE(PAGE) parm causes the report to skip to a new page whenever the REGION field changes value
- since each page contains data for only a single region, we chose to include the REGION field in the title
- a single slash (/) in the TITLE statement divides the title into two parts (left- and right-aligned)

Figure 14. A BREAK statement that produces a page break

The BREAK statement above requests that an average line and a maximum line print (in addition to the totals line) whenever the contents of the REGION field changes.

Figure 15 (page 70) shows a sample report that uses the preceding BREAK statement.

How to Produce Multiple Control Breaks

You may designate more than one sort field as a control break field. Spectrum Writer even allows *all* of your sort fields to be control break fields. However, most reports look best when no more than the first two or three sort fields are used as control breaks. The following example makes the first two sort fields control break fields:

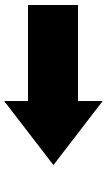
```
SORT:  REGION  EMPL-NAME  SALES-DATE
BREAK: REGION    SPACE(3)
BREAK: EMPL-NAME SPACE(1)
```

In the statements above, we made both REGION and EMPL–NAME control break fields. A control break will occur whenever the REGION field changes values (as in the previous examples). A total line will print for the region, and then 3 blank lines will print. But in this example, the second sort field, EMPL–NAME, is also designated as a control break field. So, a control break will also occur whenever the EMPL–NAME field changes value. A total line will print for the employee, followed by 1 blank line. **Figure 16** shows a sample report that uses the above statements.

Note: When multiple BREAK statements are used, they may appear in any order. However, all BREAK statements must appear after the SORT statement.

These Control Statements:

```
INPUT:   SALES-FILE
SORT:    REGION  EMPL-NAME  SALES-DATE
BREAK:   REGION  AVERAGE  MAXIMUM
TITLE:   'RECENT SALES'
TITLE:   'TOTALLED BY REGION'
COLUMNS: REGION  EMPL-NAME  SALES-DATE  SALES-TIME  CUSTOMER  AMOUNT  TAX
```



Produce this Report:

```
                              RECENT SALES
                           TOTALLED BY REGION

          EMPL     SALES    SALES
REGION    NAME     DATE     TIME     CUSTOMER          AMOUNT      TAX

EAST   MORRISON   03/29/95 15:30:22 STAR MARKET          44.35      2.66
EAST   MORRISON   03/30/95 19:05:41 A1 PHOTOGRAPHY       29.65      1.78
EAST   SIMPSON    04/01/95 08:17:57 EUROPEAN DELI        14.99      0.90
EAST   SIMPSON    04/30/95 15:30:21 J & S LUMBER         23.87      1.43
*** TOTAL FOR EAST  (4 ITEMS)                          112.86      6.77
*** AVERAGE VALUE                                       28.22      1.69
*** MAXIMUM VALUE                                       44.35      2.66


NORTH  JOHNSON    04/01/95 17:02:47 VILLA HOTEL         234.45     14.07
NORTH  JOHNSON    04/05/95 14:33:10 MARYS ANTIQUES        9.98      0.60
NORTH  JONES      04/15/95 07:58:32 EZ GROCERY           10.25      0.62
NORTH  JONES      04/15/95 13:52:41 TOY TOWN             10.25      0.62
NORTH  JONES      04/15/95 08:01:59 TOY TOWN            121.76      7.31
*** TOTAL FOR NORTH (5 ITEMS)                          386.69     23.22
*** AVERAGE VALUE                                       77.34      4.64
*** MAXIMUM VALUE                                      234.45     14.07

                    (other report lines not shown)


****** GRAND TOTAL (14 ITEMS)                        1,383.66     83.05
****** AVERAGE VALUE                                    98.83      5.93
****** MAXIMUM VALUE                                   500.00     30.00
```

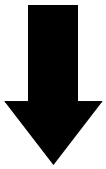
Remarks:

- the AVERAGE and MAXIMUM parms (in the BREAK statement) cause two statistical lines to print (in addition to the totals line) whenever the REGION field changes value
- at the Grand Total, the same statistical lines also print

Figure 15. A report that prints statistical information at control breaks and the Grand Totals

These Control Statements:

```
INPUT:   SALES-FILE
SORT:    REGION  EMPL-NAME  SALES-DATE
BREAK:   REGION  SPACE(3)
BREAK:   EMPL-NAME  SPACE(1)
TITLE:   'SALES TOTALLED BY EMPLOYEE AND REGION'
COLUMNS: REGION  EMPL-NAME  SALES-DATE  SALES-TIME  CUSTOMER  AMOUNT  TAX
```



Produce this Report:

```
                    SALES TOTALLED BY EMPLOYEE AND REGION

          EMPL     SALES    SALES
REGION    NAME     DATE     TIME     CUSTOMER          AMOUNT       TAX

EAST   MORRISON   03/29/95 15:30:22 STAR MARKET          44.35        2.66
EAST   MORRISON   03/30/95 19:05:41 A1 PHOTOGRAPHY       29.65        1.78
*** TOTAL FOR MORRISON   (2 ITEMS)                       74.00        4.44

EAST   SIMPSON    04/01/95 08:17:57 EUROPEAN DELI        14.99        0.90
EAST   SIMPSON    04/30/95 15:30:21 J & S LUMBER         23.87        1.43
*** TOTAL FOR SIMPSON    (2 ITEMS)                       38.86        2.33

****** TOTAL FOR EAST  (4 ITEMS)                        112.86        6.77



NORTH  JOHNSON    04/01/95 17:02:47 VILLA HOTEL         234.45       14.07
NORTH  JOHNSON    04/05/95 14:33:10 MARYS ANTIQUES        9.98        0.60
*** TOTAL FOR JOHNSON    (2 ITEMS)                     244.43       14.67

NORTH  JONES      04/15/95 07:58:32 EZ GROCERY          10.25        0.62
NORTH  JONES      04/15/95 13:52:41 TOY TOWN            10.25        0.62
NORTH  JONES      04/15/95 08:01:59 TOY TOWN           121.76        7.31
*** TOTAL FOR JONES      (3 ITEMS)                     142.26        8.55

****** TOTAL FOR NORTH (5 ITEMS)                        386.69       23.22

                         (other report lines not shown)


********* GRAND TOTAL (14 ITEMS)                     1,383.66       83.05
```

Remarks:

- the two BREAK statements make both REGION and EMP–NAME control break fields
- when the EMPL–NAME field changes, employee totals print, followed by 1 blank line
- when the REGION field changes, region totals print, followed by 3 blank lines
- the employee total line begins with 3 asterisks, while the region total line begins with 6 asterisks, and the Grand Total line has 9 asterisks (indicating the level of the break)

Figure 16. A report with two levels of control breaks

Summary

Here is a summary of what we learned in this lesson:

- use the BREAK statement to specify a **control break field**
- control break fields must also be **sort fields**
- use the SPACE parm to specify your own **spacing** at the control break
- use one or more statistical parms to request that **statistical lines** print at a control break
- you can specify **multiple control breaks** in the same report

The next lesson will show you how to turn reports with control breaks into "summary reports."

To Learn More

There are some additional features associated with the BREAK statement which we have not covered in this lesson. Some of these additional features are discussed as topics in Chapter 4, "Beyond the Basics." Some additional topics include:

- additional control break spacing parms, including one that skips to a **new sheet of paper** (page 178)
- how to print one or more **customized headers at the beginning** of a control break (page 200)
- how to print one or more **customized lines at the end** of a control break (page 188)
- how to **customize the total line** and the other statistical lines (page 182 and page 186)
- how to **suppress the total line** at a control break (page 185)
- how to print only the total lines to produce a **summary report** (page 73 and page 209)
- how to compute **percentages and ratios** that apply to an entire control group (page 202)
- how to customize the **Grand Totals** at the end of the report (page 207)

The complete syntax for the BREAK statement is given Chapter 10, "Control Statement Syntax" (page 481).

Lesson 8. How to Create Summary Reports

This lesson teaches you how to produce summary reports. The control statement discussed is:

- the OPTIONS statement

How to Create a Summary Report

A summary report is one which does not show the detail information for every record included in the report. Instead the detail information is summarized and only the totals are printed in the report.

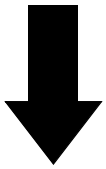
Control breaks are used to create the desired total lines. Consider the report shown earlier on page 66. It is a detail report that lists each sale made in every region. The control break on REGION causes a total line to print after the detail lines for each region have printed. By adding the following statement, we can suppress the detail lines and print just the region totals:

```
OPTIONS: SUMMARY
```

Figure 17 shows a summary report that uses the above statement.

These Control Statements:

```
OPTIONS: SUMMARY
INPUT:   SALES-FILE
SORT:    REGION  EMPL-NAME  SALES-DATE
BREAK:   REGION
TITLE:   'REGIONAL SALES SUMMARY'
COLUMNS: REGION  EMPL-NAME  SALES-DATE  SALES-TIME  CUSTOMER  AMOUNT  TAX
```



Produce this Report:

```
                          REGIONAL SALES SUMMARY

           EMPL     SALES    SALES
REGION     NAME      DATE     TIME     CUSTOMER        AMOUNT        TAX

*** TOTAL FOR EAST  (4 ITEMS)                            112.86       6.77
*** TOTAL FOR NORTH (5 ITEMS)                            386.69      23.22
*** TOTAL FOR SOUTH (2 ITEMS)                            601.38      36.09
*** TOTAL FOR WEST  (3 ITEMS)                            282.73      16.97

****** GRAND TOTAL  (14 ITEMS)                         1,383.66      83.05
```

Remarks:

- this is the same report as on page 66, except for the additional OPTIONS statement
- the SUMMARY parm (in the OPTIONS statement) suppresses the detail report lines, leaving just a summary report
- in summary reports, only the numeric columns are filled in (with total values)

Figure 17. Producing a summary report

Summary

Here is a summary of what we learned in this lesson:

- use the SUMMARY option (in the OPTIONS statement) to **create a summary report**
- a summary report **must have at least one control break field**

The next lesson will show you how to use data from more than one input file in a report.

To Learn More

There are some additional features associated with summary reports which we have not covered in this lesson. Some of these additional features are discussed as topics in Chapter 4, "Beyond the Basics." Examples of additional features include:

- **customizing** the summary lines in your report (page 182)
- **printing statistics** (such as averages, maximums and minimums) in your summary report (page 186)
- creating **multiple levels** of summarization (page 204)
- printing a **limited number of detail records** in each control group, creating reports such as "The Top 3 Sales in Each Region" (page 212)

Lesson 9. How to Use Data from More Than One File

This lesson teaches you how to read records from additional input files for use in your report. The control statement discussed is:

- the READ statement

All of the sample reports produced so far have used data from only one input file. The data has come from the file specified in the INPUT statement, called the **primary input file**. There are times when all of the data needed for a particular report will not be found in just a single file. One of Spectrum Writer's most powerful features is its ability to link to *any number* of additional files to produce a report.

How Auxiliary Input Files Are Processed

Each report is allowed to have only one primary input file, specified in the INPUT statement. When data from additional input files is required to produce a report, a READ statement is used. The READ statement causes a record to be read from another input file, called an **auxiliary input file**. You may have as many READ statements as you like in a single report.

Here is how Spectrum Writer processes the primary and auxiliary input files. Spectrum Writer first reads a single record from the primary input file. (This file is always read *sequentially*, beginning with the first record in the file.) Next, if any auxiliary input files were specified, Spectrum Writer also reads one record from each of those files. (These files are always read *randomly*, using a key.) At this point, Spectrum Writer will have read one record from each of the input files. The fields from *all* of these records are now available for use in producing the report. These fields can be used:

- as columns in the body of the report
- in titles
- as sort fields
- as control break fields
- in conditional expressions
- in calculations
- and in any other way that fields from the primary input file are used

After processing this set of records, Spectrum Writer then repeats the process. Another record is read sequentially from the primary input file. Then random reads are performed to each of the auxiliary input files. This next group of records is then used in making the report, and so on. This process is repeated until there are no more records left in the primary input file.

By simply adding a READ statement to your report request, you automatically make all of the data fields from another file available for use in producing your report.

There is one important thing about auxiliary input files to keep in mind. Since these files are ready randomly, they *must* be keyed files (or DB2 tables). (VSAM files are often keyed files.)

In a keyed file, each record has a unique "key" value associated with it. When a random read is made to such a file, a **read key** must be specified to identify which record to read. What read key should you tell Spectrum Writer use when reading a record from an auxiliary input file? In order to be useful, the auxiliary input record should be somehow related to the primary input record. Usually, the record from the primary input file will contain the key of a corresponding record in the auxiliary input file. That key from the primary input file will be used as the read key.

> **Note:** If you are not familiar with such terms as "keyed files" and "read keys," ask your programmer to help you determine whether a particular file is keyed or not, and also to help you decide what read key to use.

How to Use the READ Statement

Now let's look at a concrete example of how to use the READ statement. Begin by considering **Figure 18**, which shows a simple report that uses only a primary input file (the SALES–FILE). This report shows information about each sale made by an employee.

This report includes columns for two fields that we haven't used in previous examples, so we'll explain them. They are the EMPL–NUM field and the PRODUCT–CODE field. The EMPL–NUM is the employee number of the employee who made the sale. The PRODUCT–CODE is a code that identifies which product was sold to the customer.

Now, let's assume that we need this same report to also show each employee's social security number. The social security number is not available in the SALES–FILE. But it *is* a field in the EMPL–FILE. (See the report on page 38.) In order to produce such a report, we need data from a second input file— the EMPL–FILE.

The EMPL–FILE is a keyed VSAM file. Its key is the 3–byte employee number. The records in the SALES–FILE also contain an employee number, so we can use that field as the "read key" when we read the EMPL–FILE. So, we can make the EMPL–FILE an auxiliary input file by simply adding this statement:

```
READ:  EMPL-FILE   READKEY(EMPL-NUM)
```

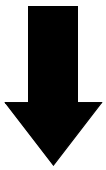
This READ statement tells Spectrum Writer to use the EMPL–NUM field from the records in the SALES–FILE as a key to read an auxiliary record from the EMPL–FILE. All control statements after this READ statement may now refer to the fields in the EMPL–FILE, as well as to those in the SALES–FILE. So, we can now add the SOCIAL–SEC–NUM field from the EMPL–FILE to our COLUMNS statement:

```
READ:     EMPL-FILE  READKEY(EMPL-NUM)
COLUMNS:  EMPL-NAME  SALES-FILE.EMPL-NUM  SOCIAL-SEC-NUM
          SALES-DATE  CUSTOMER  AMOUNT  PRODUCT-CODE
```

Notice that in the above COLUMNS statement we must now prefix the EMPL–NUM field with a record name (SALES–FILE.EMPL–NUM). This is because after the READ statement, EMPL–NUM is no longer a unique field name. A field by that name exists in both the SALES–FILE and the EMPL–FILE. (See Appendix F, "Files Used in Examples" (page 648).) Since the EMPL–NUM will have the same value in both of the records, it doesn't really matter which one we specify in the COLUMNS statement, but we do have to specify a unique name. In this case we specified the EMPL–NUM field from the SALES–FILE. (For more information on using "record names" to qualify field names, see "How to Name the Input File Records" on page 228.)

These Control Statements:

```
INPUT:   SALES-FILE
TITLE:   'RECENT SALES'
COLUMNS: EMPL-NAME  EMPL-NUM  SALES-DATE  CUSTOMER  AMOUNT  PRODUCT-CODE
```



Produce this Report:

```
                        RECENT SALES

   EMPL    EMPL  SALES                                    PRODUCT
   NAME    NUM   DATE     CUSTOMER        AMOUNT          CODE

JOHNSON    037  03/12/95 ACE ELECTRICAL          101.38   952
BAKER      044  03/26/95 JACKS CAFE              137.00   978
MORRISON   042  03/29/95 STAR MARKET              44.35   907
MORRISON   042  03/30/95 A1 PHOTOGRAPHY           29.65   919
SIMPSON    041  04/01/95 EUROPEAN DELI            14.99   916
JOHNSON    039  04/01/95 VILLA HOTEL             234.45   926
JOHNSON    039  04/05/95 MARYS ANTIQUES            9.98   997
BAKER      044  04/12/95 JACKS CAFE              135.75   916
THOMAS     045  04/14/95 YOGURT CITY               9.98   997
JONES      036  04/15/95 EZ GROCERY               10.25   977
JONES      036  04/15/95 TOY TOWN                121.76   907
JONES      036  04/15/95 TOY TOWN                 10.25   977
JOHNSON    037  04/16/95 ACME BUILDING           500.00   976
SIMPSON    041  04/30/95 J & S LUMBER             23.87   916


*** GRAND TOTAL (14 ITEMS)                    1,383.66
```

Remarks:

- all fields used in this report come from the SALES–FILE

Figure 18. A report that uses only the primary input file

Figure 19 shows a report which uses the above statements. The report now has the desired new column showing each employee's social security number. Notice that we also sorted the report on SOCIAL–SEC–NUM. Remember that you can use fields from auxiliary input files in any way that you can use fields from the primary input file.

"One-to-Many" Random Reads

Normally, Spectrum Writer reads just a single record from your auxiliary input file — the record whose key field matches the READKEY value. If you want to use *all of the records* which match the READKEY (or partial READKEY), add the MULTI parm to your READ statement. When the READ statement has the MULTI parm, Spectrum Writer creates and processes "logical input records" by matching the primary input file row with *each* qualifying record from the auxiliary input file. For more information on how the MULTI parm works, see "How to Perform "One–to–Many" Reads" on page 232.

How to Use Multiple READ Statements

You may use as many READ statements in a run as you like. The report in **Figure 20** uses two READ statements. The primary input file is once again the SALES–FILE, which contains one record for each sale made by an employee.

To obtain additional data about the employee who made each sale, we use a READ statement for the EMPL–FILE (just like in the preceding example). The EMPL–NUM field in the SALES–FILE contains the key necessary to read the correct EMPL–FILE record.

To obtain additional information about each *product* sold, a second READ statement names the PRODUCT–FILE as another auxiliary input file. (The PRODUCT–FILE is also described in Appendix F, "Files Used in Examples" on page 648.)

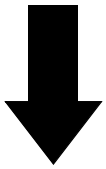
However, there is one minor complication in reading records from this file. The key in the PRODUCT–FILE records is 4 bytes long. It consists of the letter "P" followed by a 3–byte product code. The SALES–FILE does not contain a field which can be used *directly* as the read key to the PRODUCT–FILE. But it does contain the 3–byte PRODUCT–CODE field, which we can use to build the 4–byte read key. A COMPUTE statement is therefore used to create a new field (called PRODKEY) which consists of the letter "P" followed by the product code. This computed field is then used as the read key in the READ statement for the PRODUCT–FILE:

```
COMPUTE: PRODKEY = 'P' + PRODUCT-CODE
READ:    PRODUCT-FILE   READKEY(PRODKEY)
```

By having two READ statements in addition to the INPUT statement, the report now uses data from three input files. Data from all of these files can be used in any of the subsequent control statements. In the report in **Figure 20**, the COLUMNS statement uses two fields from the auxiliary input files. It uses the SOCIAL–SEC–NUM field from the EMPL–FILE and the PRODUCT–DESC field from the PRODUCT–FILE.

These Control Statements:

```
INPUT:   SALES-FILE
READ:    EMPL-FILE  READKEY(EMPL-NUM)
SORT:    SOCIAL-SEC-NUM
TITLE:   'SALES SORTED BY SOCIAL SECURITY NUMBER'
COLUMNS: EMPL-NAME  SALES-FILE.EMPL-NUM  SOCIAL-SEC-NUM
         SALES-DATE  CUSTOMER  AMOUNT  PRODUCT-CODE
```



Produce this Report:

SALES SORTED BY SOCIAL SECURITY NUMBER

EMPL NAME	SALES FILE EMPL NUM	SOCIAL SEC NUM	SALES DATE	CUSTOMER	AMOUNT	PRODUCT CODE
JOHNSON	039	004-77-9981	04/05/95	MARYS ANTIQUES	9.98	997
JOHNSON	039	004-77-9981	04/01/95	VILLA HOTEL	234.45	926
JONES	036	012-09-8765	04/15/95	EZ GROCERY	10.25	977
JONES	036	012-09-8765	04/15/95	TOY TOWN	10.25	977
JONES	036	012-09-8765	04/15/95	TOY TOWN	121.76	907
SIMPSON	041	112-05-0456	04/30/95	J & S LUMBER	23.87	916
SIMPSON	041	112-05-0456	04/01/95	EUROPEAN DELI	14.99	916
THOMAS	045	776-83-8221	04/14/95	YOGURT CITY	9.98	997
BAKER	044	878-19-0156	04/12/95	JACKS CAFE	135.75	916
BAKER	044	878-19-0156	03/26/95	JACKS CAFE	137.00	978
MORRISON	042	900-12-0556	03/30/95	A1 PHOTOGRAPHY	29.65	919
MORRISON	042	900-12-0556	03/29/95	STAR MARKET	44.35	907
JOHNSON	037	912-04-0334	03/12/95	ACE ELECTRICAL	101.38	952
JOHNSON	037	912-04-0334	04/16/95	ACME BUILDING	500.00	976

*** GRAND TOTAL (14 ITEMS) 1,383.66

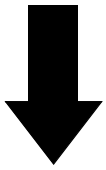
Remarks:

- the READ statement makes the fields from the EMPL–FILE available for use
- the COLUMNS statement includes the SOCIAL–SEC–NUM field from the EMPL–FILE
- we also sorted the report on the SOCIAL–SEC–NUM field from the EMPL–FILE
- the EMPL–NUM field must be prefixed with a record name in the COLUMNS statement, since a field by that name exists in both input files

Figure 19. A report that uses a READ statement to specify an auxiliary input file

These Control Statements:

```
INPUT:    SALES-FILE
READ:     EMPL-FILE      READKEY(EMPL-NUM)
COMPUTE:  PRODKEY = 'P' + PRODUCT-CODE
READ:     PRODUCT-FILE   READKEY(PRODKEY)
TITLE:    'SALES SORTED BY SOCIAL SECURITY NUMBER'
COLUMNS:  EMPL-NAME
          SALES-FILE.EMPL-NUM
          SOCIAL-SEC-NUM
          SALES-DATE
          CUSTOMER
          PRODUCT-CODE
          PRODUCT-DESC
```



Produce this Report:

SALES SORTED BY SOCIAL SECURITY NUMBER

EMPL NAME	SALES FILE EMPL NUM	SOCIAL SEC NUM	SALES DATE	CUSTOMER	AMOUNT	PRODUCT CODE	PRODUCT DESC
JOHNSON	039	004-77-9981	04/05/95	MARYS ANTIQUES	9.98	997	MAILING LABELS
JOHNSON	039	004-77-9981	04/01/95	VILLA HOTEL	234.45	926	DESK CALENDARS
JONES	036	012-09-8765	04/15/95	EZ GROCERY	10.25	977	PAPER CLIPS
JONES	036	012-09-8765	04/15/95	TOY TOWN	10.25	977	PAPER CLIPS
JONES	036	012-09-8765	04/15/95	TOY TOWN	121.76	907	INKPADS
SIMPSON	041	112-05-0456	04/30/95	J & S LUMBER	23.87	916	RED PENS
SIMPSON	041	112-05-0456	04/01/95	EUROPEAN DELI	14.99	916	RED PENS
THOMAS	045	776-83-8221	04/14/95	YOGURT CITY	9.98	997	MAILING LABELS
BAKER	044	878-19-0156	04/12/95	JACKS CAFE	135.75	916	RED PENS
BAKER	044	878-19-0156	03/26/95	JACKS CAFE	137.00	978	HOLE PUNCHERS
MORRISON	042	900-12-0556	03/30/95	A1 PHOTOGRAPHY	29.65	919	GREEN PENS
MORRISON	042	900-12-0556	03/29/95	STAR MARKET	44.35	907	INKPADS
JOHNSON	037	912-04-0334	03/12/95	ACE ELECTRICAL	101.38	952	PENCILS (NO. 1)
JOHNSON	037	912-04-0334	04/16/95	ACME BUILDING	500.00	976	CHAIRS

*** GRAND TOTAL (14 ITEMS) 1,383.66

Remarks:

- all fields from the SALES–FILE, the EMPL–FILE and the PRODUCT–FILE are available for use in the report
- the key to the PRODUCT–FILE is a computed field
- the EMPL–NUM field must be prefixed with a record name in the COLUMNS statement, since a field by that name exists in two input files (SALES–FILE and EMPL–FILE)
- the SOCIAL–SEC–NUM field comes from the EMPL–FILE auxiliary input file
- the PRODUCT–DESC field comes from the PRODUCT–FILE auxiliary input file

Figure 20. A report that uses two READ statements to specify two auxiliary input files

Summary

Here is a summary of what we learned in this lesson:

- the READ statement is used to read records from **auxiliary input files**
- the file named in a READ statement must be a **keyed file** (or a DB2 table)
- you may have as **many READ statements as you like** in a single report

To Learn More

There are some additional features associated with the READ statement which we have not covered in this lesson. Some of these additional features are discussed as topics in Chapter 4, "Beyond the Basics." Some additional features include:

- how to assign a **record name** to the records read from auxiliary input files (page 228)
- how to read multiple records (containing **different keys**) from the same auxiliary input file (page 224)
- how to use **data from one auxiliary input file as the read key** to another auxiliary input file (page 226)
- how to specify **generic keys** and **"KGE" keys** in the READ statement (page 230)
- how to read multiple records (with the **same key or partial key**) from the auxiliary input file (page 232)
- what happens when **no record is found** for a particular read key (page 229)
- how to determine whether the read for a particular key was **successful** or not (page 230)
- how to use the READ statement to obtain data from a **DB2 table or view** (page 393)
- how to use the ONIOERROR option to **increase the severity** of I/O errors on an input file (page 586)

The complete **syntax** for the READ statement, as well as a more detailed **narrative** of how Spectrum Writer assembles input records during the report process, is given in Chapter 10, "Control Statement Syntax" (page 578).