READ Statement

(Specifies an auxiliary input file for the report.)

```
READ: filename
 READKEY(fieldname)    key (or partial key) of VSAM rec(s) to read
 WHERE(search–condition)     defines row(s) of DB2 table to read
 [ GENERIC ]                      readkey is generic - VSAM only
 [ KGE ]                  key greater than or equal - VSAM only
 [ MULTI ]                             read all matching records
 [ SHOWFLDS(YES/NO) ]        print a list of all fields in this file?
```

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

OPTIONS Statement

(Specifies various report options. Partial list only.)

```
OPTIONS: [ SUMMARY ] print summary report (suppress detail lines)
 [ PC ]     format report as a comma-delimited file for use on a PC
 [ HTML[('html-title')] ]                  format report as HTML
 [ MAXINPUT(nnnn) ] [ MAXINCLUDE(nnnn) ]    useful for testing
 [ HEADINGSEP('/')    change the column-heading separator char
 [ CENTURY(nn/50) ] specify the cutoff year for century windowing
 [ FORMAT(disp-fmt, disp-fmt, ...) ]  global format override
```

Run JCL

```
//STEP1    EXEC PGM=SPECTWTR
//STEPLIB   DD DSN=XXXXXX.LOADLIB,DISP=SHR Steplib, if needed
//SWCOPY    DD DSN=XXXXXX.COPYLIB,DISP=SHR       Copy Library
//SWLIST    DD SYSOUT=*                         Control Listing
//SWOUTPUT DD SYSOUT=*                     Report or Output File
//SYSOUT    DD SYSOUT=*                    Sort Control Listing
//SORTWK01 DD UNIT=SYSDA,SPACE=(CYL,(5,1))           Sort Work
//SORTWK02 DD UNIT=SYSDA,SPACE=(CYL,(5,1))
//SORTWK03 DD UNIT=SYSDA,SPACE=(CYL,(5,1))
//XXXXXXXX DD DSN=XXXXXX.XXXXXX,DISP=SHR              Input File
//SYSIN     DD *                           Control Statements
INPUT:   filename
TITLE:   'your title'
COLUMNS: fieldname fieldname fieldname ...
//
```

Converting a COBOL Record Layout

```
//STEP1    EXEC PGM=SPECTWTR
//COBLIB    DD DSN=XXXXX.COPYLIB,DISP=SHR  COBOL Copy Lib
//SWLIST    DD SYSOUT=*                      Control Listing
//SWOUTPUT DD SYSOUT=*                  Report  (will be empty)
//FLDSOUT   DD DSN=XXXXX.XXXXX,  FIELD stmts are written  here
//             DISP=(,CATLG),UNIT=SYSDA,SPACE=(TRK,10)
//SYSIN     DD *                        Control Statements
FILE: DUMMY
COBOL: OUTDDN(FLDSOUT)
COPY: COBSALES
//
```

The above example converts a COBOL record layout named COBSALES to FIELD statements and writes them out to a file. Edit the resulting file. Manually add a FILE statement, and set the correct datatypes for any date or time fields.

Display Formats

*Complete list is in Appendix B. Remember, the way data is formatted in the report is determined by **display formats** (not by a field's **datatype** in the FIELD statement.)*

Display Format	Example	Notes
Display Formats for Numeric Fields		
NUMERIC	-1,234.56	**Default** format for all numeric fields (regardless of their datatype.) Floating negative sign. Leading zero suppression. Commas inserted.
NOCOMMA	-12345.56	Same as NUMERIC except that no commas are inserted. Useful in comma delimited files.
DOTSEP	-1.234,56	Same as NUMERIC except that dots are used as thousands separators, and a comma indicates decimal location.
DISPLAY	001234.5M	No punctautian except for a decimal. No leading zero suppression. Sign is in the zone portion of last byte. Same as COBOL USAGE DISPLAY.
PIC'999999.9' PIC'$$,$$9.99'	001234.6 $1,234.56	User picture (similar to COBOL pictures.)
COMP	X'04D2'	Signed binary. Useful for mainframe output files. Same as COBOL USAGE COMP.
COMP-3	X'01234C'	Signed packed. Useful for mainframe output files. Same as COBOL USAGE COMP-3.
Display Formats for Date Fields		
MM-DD-YY	12/31/05	**Default** format for all dates.
DD-MM-YY	31/12/05	
YY-MM-DD	05/12/31	
MM-DD-YYYY	12/31/2005	
SHORT1	DEC 31, 2005	
SHORT2	31 DEC 2005	
SHORT3	31 DEC 05	
LONG1	DECEMBER 31, 2005	
LONG2	31 DECEMBER 2005	
LONG3	31 DECEMBER 05	
Q-MM-DD-YY	"12/31/05"	Useful for comma delimited output files.
P-YYDDD	X'05365C'	Packed julian. Useful for mainframe output files. Same as COBOL S9(5) COMP-3.
Note: to format dates with dashes instead of slashes (12-31-05), add an OPTION: DATEDELIM('-') statement.		
Display Formats for Time Fields		
HH-MM-SS	23:30:59	**Default** format for all times.
HH-MM-SS-AMPM	11:30:59 PM	12-hour clock with AM/PM.
HH-MM	23:31	Seconds rounded out.
HH-MM-AMPM	11:31 PM	
Note: to format times with dots instead of colons (23.30.59), add an OPTION: TIMEDELIM('.') statement.		

fold here so that this line is on the outside

Spectrum Writer OS/390 Quick Reference

Simplified Control Statement Syntax and Other Useful Information

FILE Statement

(Defines one input file or DB2 table.)

```
FILE: filename          user-friendly name to assign to this file
 DDNAME(ddname)               DDNAME used for file in run JCL
 [ LRECL(nnnnn/1000) ]    size of largest potential record in file
 [ TYPE(SEQ/VSAM/DB2) ]                           type of file
 [ DB2NAME('[qualifier.]name') ]  DB2 table name - DB2 only
```

Example: `FILE: PAYROLL DDNAME(PAYROLL)`

FIELD Statement

(Defines one field in an input file.)

```
FIELD: fieldname        user-friendly name to assign to this field
 [ LENGTH(nnnnn) ]          length in record (in bytes, not digits)
 [ TYPE(datatype/CHAR) ]             type of data (see list below)
 [ DECIMALS(nn/0) ]             for numeric and time fields only
 [ COLUMN/DISP(*/nnnnn/fldname [+/- nnnn]) ] location in rec
 [ HEADING('heading1|heading2...') ]            column heading
 [ FORMAT(display-format) ]       how to format (see list on left)
 [ ACCUM/NOACCUM ]          should field appear in Grand Totals?
```

Examples:

```
FIELD: LAST-NAME LEN(20) HEAD('SURNAME')
FIELD: SALE-AMOUNT COL(110) LEN(5) TYPE(COMP-3) DEC(2)
FIELD: SALE-DATE TYPE(YYYYMMDD)
FIELD: NUM-SALE-MONTH COL(SALE-DATE+4) LEN(2) TYPE(NUM)
FIELD: DEPARTMENT-NUM COL(*+2) LEN(2) TYPE(COMP)
       NOACCUM  FORMAT(PIC'9-999')
```

Data Types

(Complete list is in Appendix A.)

Datatype	Example	Notes
CHARACTER	JOHNSON 001	As-is text - cannot be totaled, even if all chars are numeric.
NUMERIC	001 1 -1,234.56	Character numeric data. Like COBOL PIC 9999 Is totaled by default.
NUM-SLD	123D 123M	Numeric, with sign in last digit. COBOL PIC S9999.
COMP	X'7FFF'	Signed binary num (1-8 bytes)
COMP-3	X'123C'	Signed packed num (1-16 bytes)
YYYYMMDD	20031231	Character date (8 bytes)
YYMMDD	031231	Character date (6 bytes)
YYDDD	03365	Character Julian date (5 bytes)
H-YYYYMMDD	X'20031231'	Hex date (4 bytes)
H-YYMMDD	X'031231'	Hex date (3 bytes)
P-YYDDD	X'03365C'	Packed Julian date (5 bytes)
HHMMSS	133059	Character time (6 bytes)
H-HHMMSS	X'133059'	Hex time (3 bytes)
HHMM	1401	Character time (4 bytes)
H-HHMM	X'1401'	Hex time (2 bytes)

1-800-572-5517 ♦ www.pacsys.com

INPUT Statement

(Names the primary input file for the report. Required.)

```
INPUT: filename
 [ KEYRANGE('begin' ['end']) ] limit records read - VSAM only
 [ WHERE(search–condition) ]    limit rows retrieved - DB2 only
 [ SHOWFLDS(YES/NO) ]          print a list of all fields in this file?
```

Example: `INPUT: SALES-FILE`

COLUMNS Statement

(Names the fields desired as columns in the report.)

```
COLUMNS:  [n] item [n] item [n] item ...
```

n is optional and specifies the number of blank spaces wanted before the next column. Each **item** is either a **fieldname** or a **literal text** (in quotes). To customize an item's appearance, follow it with a parm list in parentheses. (No space is allowed before the open parenthesis.) Parms may appear in any order.

```
fieldname[( [ nnn ]                                 override column width
 [ 'heading1|heading2...' ]                       override column heading
 [ display-format ]                     override format (see list on reverse)
 [ BIZ ]                                                     blank if zero
 [ LEFT/CENTER/RIGHT ]                       how to justify contents of field
 [ NOREPEAT/NOREPEATPAGE ]                        blank out repeating values
 [ ACCUM/NOACCUM ] )]         include this column in the Grand Totals?

 'literal'[( [ nnn ]                                override column width
 [ 'heading1|heading2...' ]  )]                            column heading
```

Example:
```
COLUMNS: REGION EMPL-NAME SALES-DATE(SHORT1)
         AMOUNT(PIC'$$,$$9') TELEPHONE(BIZ)
         'CODE=' 0 STATUS-CODE('S|T|A|T|U|S' 1)
         SSN(PIC'999-99-9999' NOACCUM)
```

TITLE (also FOOTNOTE) Statement

(Defines a title or footnote for report pages.)

```
TITLE: print–expr  [/ print–expr]  [/ print–expr]
```

Up to 3 print-expressions, separated by slashes, define the left-aligned, centered, and right-aligned parts of the title.

Each print-expression is just like a COLUMNS statement :

```
[n] item [n] item [n] item ...
```

Each **item** is either a **literal text** in quotes, or a **fieldname,** optionally followed by a parm list in parentheses.

```
fieldname[( [ nnn ]                                      overwide width
 [ display-format ]                     override format (see list on reverse)
 [ BIZ ]                                                     blank if zero
 [ LEFT/CENTER/RIGHT ]  )]                  how to justify contents of field
```

Built-In Fields Useful in the TITLE Statement

#TODAY	*system date*
#**HHMMSS**	*system time of day*
#**PAGENUM**	*current page number*

Example:
```
TITLE: 'DATE:' #TODAY(LONG1) #HHMMSS(HH-MM-AMPM)
     / 'SALES REPORT FOR REGION:' REGION
     / 'PAGE:' #PAGENUM(PIC'ZZ9')
```

SORT Statement

(Defines sort order of report.)

```
SORT: fieldname[(ASC/DESC)]                            primary sort field
      fieldname[(ASC/DESC)] ...                      secondary sort field
      ...                                          additional sort fields
      [ #EQUALS ]     keep equal records in original relative order
```

Example: `SORT: REGION STATE CITY`

BREAK Statement

(Defines one level of control break, or customizes Grand Totals.)

The break field **must** be a sort field (named in the SORT statement.) Or, use #GRAND to customize the report's Grand Totals.

```
BREAK: fieldname/#GRAND          break field (or Grand Total break)
 [ TOTAL[(print–expression) ]/NOTOTAL ]               print total line?
 [ AVERAGE[(print–expression)] ]                    print average line
 [ MAXIMUM[(print–expression)] ]                   print maximums line
 [ MINIMUM[(print–expression)] ]                   print minimums line
 [ FOOTING(print–expression) ... ]                print footing line(s)
 [ HEADING(print–expression) ... ]                print heading line(s)
 [ REPEAT ]                               repeat heading(s) on each page
 [ SPACE(n/2/PAGE/PAGE1) ]           blank lines to print, or new page
```

Each print-expression is just like a COLUMNS statement :

```
[n] item [n] item [n] item ...
```

Each **item** is either a **literal text** in quotes, or a **fieldname,** optionally followed by a parm list in parentheses.

```
fieldname[( [ nnn ]                                      overwide width
 [ display-format ]                     override format (see list on reverse)
 [ BIZ ]                                                     blank if zert
 [ LEFT/CENTER/RIGHT ]                      how to justify contents of field
 [ TOTAL/AVERAGE/MAX/MIN/NZAVG/NZMIN ]  )]                   value to prin
```

Note: If TOTAL, AVERAGE, etc. is specified for a field, then the total value (or average value, etc.) of that field for the whole control group is printed. If omitted, the value of the field from the first record (if HEADING parm) or last record (all other parms) in the control group is printed.

Built-In Fields Available in the BREAK Statement

#ITEMS	*the number of items in the control group*
#COUNTER	*the running total number of items in the report so far*

Example:
```
BREAK: REGION SPACE(PAGE)
 HEADING('SALES IN' REGION 'REGION FOLLOW')
 FOOTING('AVERAGE SALE IN REGION =' AMOUNT(AVG))
 FOOTING('MAXIMUM SALE IN REGION =' AMOUNT(MAX))
```

INCLUDEIF Statement

(Specifies which input records to include in report.)

```
INCLUDEIF: conditional-expression
```

A conditional expression is one or more **tests**, separated by OR (|) or AND (&), optionally grouped within parentheses. There are two types of tests.

1. Comparison test: `[NOT] operand1 operator operand2`
2. Bit test: `[NOT] bit-fieldname`

fold here so that this line is on the outside

In a comparison test, each operand can be either a fieldname or a literal value. After the first test, *operand1* and the *operator* are optional. When omitted, operand1 and/or the operator from the previous test are used. Preceding a test with NOT reverses the result of the test.

List of Operators for Comparison Tests

=	is equal to	**<>** *or* **¬=**	is not equal to
<	is less than	**>**	is greater than
<=	is less than or equal to	**>=**	is greater than or equal to
:	"contains"	**¬:**	does not "contain"

Note: "contains" means that the full text of character operand2 is contained somewhere within character operand1.

Examples:
```
INCLUDEIF: AMOUNT > 99.99 AND REGION <> "WEST"
INCLUDEIF:  PART-TIME     if PART-TIME is defined as a bit field
INCLUDEIF: (SALES-DATE >= 1/1/2002 AND <= 12/31/2004) &
           (REGION = "SOUTH" OR "NORTH" OR "EAST")
```

COMPUTE Statement

(Creates a new field.)

Simple format:

```
COMPUTE: result-name = computational-expression
```

Conditional format:

```
COMPUTE: result-name =
    WHEN(conditional-expr) ASSIGN(computational-expr)
  [ WHEN(conditional-expr) ASSIGN(computational-expr) ]
    ...
  [ ELSE                   ASSIGN(computational-expr) ]
```

Conditional expressions are described above under the INCLUDEIF statement. The syntax of a computational expression is:

```
operand [operator operand] [operator operand] ...
```

Operands can be: fieldnames, literal values or built-in functions. **Operators** are the standard +, -, *, and / for numeric operands, and + for character concatenation.

Note: computed fields may be used *anywhere* that a real field from a file can be used-- as a sort field, break field, read key, in the INCLUDEIF statement, as an operand in another COMPUTE statement, and so on.

Examples:
```
COMPUTE: TAX = AMOUNT * .08
COMPUTE: TAX = WHEN(REGION="SOUTH") ASSIGN(AMOUNT * .07)
               WHEN(REGION="NORTH") ASSIGN(AMOUNT * .06)
               ELSE                 ASSIGN(AMOUNT * .08)
```