
Enterprise PeopleTools 8.50 PeopleBook: SQR Language Reference for PeopleSoft

September 2009

Appendix B

Using SQR Sample Programs 443

Understanding SQR Samples 443

 SQR Sample Programs 443

Index 449

- How to manage the PeopleSoft online library including full-text searching and configuring a reverse proxy server.
- Understanding documentation integration and how to integrate customized documentation into the library.
- Glossary of useful PeopleSoft terms that are used in PeopleBooks.

You can find this companion PeopleBook in your PeopleSoft online library.

- The PROCEDURE section performs the tasks to produce the report.

```
BEGIN-PROCEDURE {procedure_name}
  {procedure commands}...
END-PROCEDURE
```

SQR Syntax Conventions

The following table describes the SQR syntax conventions:

<i>Syntax Convention</i>	<i>Description</i>
{ }	Braces enclose required items.
[]	Square brackets enclose optional items.
...	Ellipses indicate that the preceding parameter can be repeated.
	A vertical bar separates alternatives inside brackets, braces, or parentheses.
!	An exclamation point begins a single-line comment that extends to the end of the line. Each comment line must begin with an exclamation point.
'	A single quote starts and ends a literal text constant or any argument with more than one word. Important! If you are copying code directly from the sample programs, change the slanted quotes to regular quotes; otherwise, you will receive an error message.
,	A comma separates multiple arguments.
()	Parentheses must enclose an argument or element.
UPPERCASE	SQR commands and arguments are uppercase within the text, but lowercase in the sample programs. (Note that these commands are <i>not</i> case sensitive.)
<i>Variable</i>	Information and values that you must supply appear in <i>variable</i> style.

Flag	Description
-BURST:{xx}	<p>-BURST:T generates the table of contents file only.</p> <p>-BURST:S generates the report output according to the symbolic table of contents entries that are set in the program with the <i>level</i> argument of the TOC-ENTRY command. In -BURST:S[{1}], {1} is the level on which to burst. The -BURST:S setting is equivalent to -BURST:S1.</p> <p>See <i>Enterprise PeopleTools 8.50 PeopleBook: SQR for PeopleSoft Developers</i>, "Generating and Publishing HTML from an SQR Program," Bursting Reports.</p> <p>Note. -BURST:P and -BURST:S require -PRINTER:EH or -PRINTER:HT.</p> <p>The page range selection feature of -BURST:P requires -PRINTER:HT.</p> <p>-BURST:T requires -PRINTER:HT.</p>
-C	(Microsoft Windows) Specifies that the Cancel dialog box appears while the program runs so that you can easily stop the program.
-CB	(Microsoft Windows, Callable SQR) Forces the communication box to use.
-Dnn	<p>(Non-Microsoft Windows) Displays the report output on the terminal while it is being written to the output file. The value for <i>nn</i> is the maximum number of lines to display before pausing. If no number is entered after -D, the display scrolls continuously.</p> <p>Note. The printer type must be LP; otherwise, the display is ignored. If the program is producing more than one report, the display is for the first report only.</p>
-DBdatabase	(Sybase) Forces the SQR program to use the specified database, which overrides any USE command in the SQR program.
-DEBUG [xxx]	See Chapter 2, "SQR Command Reference," #DEBUG, page 71.
-DNT: {xx}	See Chapter 2, "SQR Command Reference," DECLARE-VARIABLE, page 108.

Flag	Description
-E[<i>file</i>]	Directs error messages to the named file or to the default file program.err. If no errors occur, no file is created.
-EH_APPLETS: <i>dir</i>	Specifies the directory location of the enhanced HTML applets. If you include an applet, SQR for PeopleSoft must know where it resides. SQR for PeopleSoft usually checks for the applet in a default directory; the default directory for these applets is IMAGES. Note. This flag is applicable only when you specify the -PRINTER:EH or -PRINTER:EP flag.
-EH_BQD	Generates a {report}.bqd file from the report data. Also associates a query format file (BQD) icon with {report}.bqd in the navigation bar. Note. This flag is applicable only when you specify the -PRINTER:EH or -PRINTER:EP flag.
-EH_BQD: <i>file</i>	Associates the BQD icon with the specified file. Note. This flag is applicable only when you specify the -PRINTER:EH or -PRINTER:EP flag.
-EH_BROWSER: <i>xx</i>	Generates HTML, determines the browser, and displays HTML. When this flag is set to ALL, SQR for PeopleSoft generates frame.html, which contains JavaScript to determine the browser on the user's machine (that is, the person reading the report, not the person writing it). When this flag is set to BASIC, SQR for PeopleSoft generates HTML that is suitable for all browsers. When this flag is set to IE, SQR for PeopleSoft generates HTML that is designed for Microsoft Internet Explorer. When this flag is set to NETSCAPE, SQR for PeopleSoft generates HTML that is designed for Netscape. Note. This flag is applicable only when you specify the -PRINTER:EH or -PRINTER:EP flag.

<i>Flag</i>	<i>Description</i>
-EH_CSV	Generates a {report}.csv file from the report data. Note. This flag is applicable only when you specify the -PRINTER:EH or -PRINTER:EP flag.
-EH_CSV:file	Associates the CSV icon with the specified file. Note. This flag is applicable only when you specify the -PRINTER:EH or -PRINTER:EP flag.
-EH_CSVONLY	Creates a .csv file, but does not create an HTML file. Note. This flag is applicable only when you specify the -PRINTER:EH or -PRINTER:EP flag.
-EH_FULLHTML:xx	Specifies the level of the generated enhanced HTML code. This can be 30, 32, or 40. Note. For upward compatibility, a value of TRUE is equivalent to 40 and FALSE is 30.
-EH_Icons:dir	Specifies the directory in which the HTML should find the referenced icons. Note. This flag is applicable only when you specify the -PRINTER:EH or -PRINTER:EP flag.
-EH_IMAGES:dir	Specifies the directory path for the .gif files that are used by the navigation bar. Note. This flag is applicable only when you specify the -PRINTER:EH or -PRINTER:EP flag.
-EH_KEEP	Copies (does not move) the files when used in conjunction with -EH_ZIP. Note. This flag is applicable only when you specify the -PRINTER:EH or -PRINTER:EP flag.

Flag	Description
-EH_LANGUAGE:xx	<p>Sets the language that is used for the HTML navigation bar. You can specify English, French, German, Portuguese, Spanish, Japanese, Simplified Chinese, or Korean.</p> <p>Note. This flag is applicable only when you specify the -PRINTER:EH or -PRINTER:EP flag.</p>
-EH_PDF	<p>Associates a PDF icon with {report}.pdf in the navigation bar.</p> <p>Note. This flag is applicable only when you specify the -PRINTER:EH or -PRINTER:EP flag.</p>
-EH_Scale:{nm}	<p>Sets the scaling factor to a value from 50 to 200.</p> <p>Note. This flag is applicable only when you specify the -PRINTER:EH or -PRINTER:EP flag.</p>
-EH_XML:file	<p>Associates the XML icon with the specified file.</p> <p>Note. This flag is applicable only when you specify the -PRINTER:EH or -PRINTER:EP flag.</p>
-EH_ZIP[:file]	<p>Moves the generated files to the specified file or {report}.zip if {file} is not specified.</p> <p>Note. This flag is applicable only when you specify the -PRINTER:EH or the -PRINTER:EP flag.</p>

Flag	Description
-F[file directory]	<p>Overrides the default output file name, program.lis. The default action places the program.lis file in the same directory as the program.sqr file. To use the current directory, specify -F without an argument. To change the name of the output file, specify -F with the new name. If the new name does not specify a directory, the file is created in the current directory. The output file is not created until data is actually printed on the page. If no data is printed, no output file is created.</p> <p>Specify these file names and directories for different operating systems:</p> <p>UNIX/Linux</p> <p>Directory character is /</p> <p>-F\$HOME/reports/</p> <p>IBM MVS</p> <p>Directory character is (</p> <p>-FDSN:SQR.REPORTS(</p>
-GPRINT=YES NO	(IBM MVS) -GPRINT=YES includes control characters in the first column of each record of the SQR report output file. -GPRINT=NO omits the control characters.
-Idir_list	<p>Specifies the list of directories that SQR for PeopleSoft searches when processing the #INCLUDE directive when the include file does not exist in the current directory and no path is specified for the file. The directory names must be separated by commas (,) or semicolons (;).</p> <p>For UNIX/Linux-based systems, if your shell uses semicolons as command delimiters, you must precede each semicolon with a backslash (\). Always append the directory character to the end of each directory name. See the -F flag for a list of directory characters, sorted by operating system.</p>
-ID	(Non-Microsoft Windows) Displays the copyright banner on the console.
-KEEP	See <i>Enterprise PeopleTools 8.50 PeopleBook: SQR for PeopleSoft Developers</i> , "Printing with SQR."

Flag	Description
-LL{s d}{c i}	<p>Specifies the following LOAD-LOOKUP values:</p> <ul style="list-style-type: none"> • s: SQR • d: DB • c: Case sensitive • i: Case insensitive <p>See Chapter 2, "SQR Command Reference," LOAD-LOOKUP, page 174.</p>
-Mfile	<p>Defines a startup file containing sizes to assign to internal parameters—extremely small, large, or complex reports. <i>Mfiles</i> are text files that have individual switches in the INI files that are unique to a run.</p>
-NOLIS	<p>Prevents the creation of .lis files, creating .spfs instead.</p>
-O[file]	<p>Directs log messages to the specified file or to program.log if no file is specified. By default, the sqr.log file is created in the current working directory.</p>
-olim	<p>Displays the SQR resources that are used by the SQR report.</p> <ul style="list-style-type: none"> • Limit is the programmatic limit, the maximum amount of memory that can be allocated to the variable. • Defined is the value defined in the pssqr.ini file; this is a user-defined value. • Actual is the actual amount of memory/space used by the variable at runtime.
-P	<p>(IBM MVS) Suppresses printer control characters from column 1.</p>
-PB	<p>(Informix) Causes column data to preserve trailing blanks.</p>

<i>Flag</i>	<i>Description</i>
-PRINTER:xx	<p>Uses printer type xx when creating output files. The xx represents:</p> <p>EH: Enhanced HTML</p> <p>-PRINTER:EH</p> <p>EP: Enhanced HTML or PDF</p> <p>-PRINTER:EP</p> <p>HP: HP LaserJet</p> <p>-PRINTER:HP</p> <p>HT: HTML 2.0</p> <p>-PRINTER:HT</p> <p>LP: Line printer</p> <p>-PRINTER:LP</p> <p>PD: PDF</p> <p>-PRINTER:PD</p> <p>PS: PostScript</p> <p>-PRINTER:PS</p> <p>WP: Microsoft Windows</p> <p>-PRINTER:WP</p> <p>LP, HP, and PS produce files .lis files. EH and HT produce .htm files. HT produces version 2.0 HTML files with the report content inside <PRE></PRE> tags. EH produces reports in which content is fully formatted with version 3.0 or 3.2 HTML tags. On Microsoft Windows systems, WP sends the output to the default Microsoft Windows printer. To specify a Microsoft Windows printer that is not the default, enter -PRINTER:WP:{printer name}, where {printer name} is the name that is assigned to your printer. For example, to send output to a Microsoft Windows printer named New Printer, use -PRINTER:WP:NewPrinter. If the printer name has spaces, enclose the entire argument in quotes. To create an .spf file also, use -KEEP.</p>
-RS	<p>Saves the program in a runtime file. The program is scanned, compiled, and checked for correct syntax. Queries are validated and compiled. The executable version is saved in a file named program.sqt.</p> <p>Note. SQR for PeopleSoft does not prompt ASK variables after compilation.</p>

<i>Flag</i>	<i>Description</i>
-RT	Uses the runtime file that is saved with the -RS flag. This skips all syntax and query checking, and processing begins immediately. Note. SQR for PeopleSoft does not prompt ASK variables after compilation.
-S	Requests that the status of all cursors be displayed at the end of the report run. Status includes the text of each SQL statement, the number of times each was compiled and run and the total number of rows that were selected. The output appears directly on the screen. This information can be used for debugging SQL statements, enhancing performance, and tuning.
-Tnn	Specifies that you want to test your report for <i>nn</i> pages. To save time during testing, SQR for PeopleSoft ignores all ORDER BY clauses in SELECT statements. If the program is producing more than one report, SQR for PeopleSoft stops after producing the specified number of pages defined for the first report.

Flag	Description
-T{B}	<p>(Microsoft Windows, IBM DB2, Sybase CT-Lib, and ODBC) Trims trailing blanks from database character columns.</p> <p>If the TB flag is set in DB2 database environment, SQR trims the all-blanks fields (fields that contain only spaces) to NULL values in the SQR buffers.</p> <p>Using the TB flag on IBM MVS and DB2 have no effect IBM MVS and DB2 prevent SQR for PeopleSoft from removing trailing blanks from database character columns.</p> <p>Note. The -TB flag only has an effect if SQR is connecting to either a DB2, Sybase CT-Lib, or ODBC (MSS) database. Confusingly, the behavior of the -TB command-line flag varies depending on the platform. If you are using one of the above mentioned databases and are running SQR on z/OS, the -TB flag will behave as follows:</p> <p>If you do not use the -TB flag, trailing blanks are trimmed.</p> <p>If you do use the -TB flag, trailing blanks are not trimmed.</p> <p>If you are running SQR on any other platform, the behavior of -TB is the opposite:</p> <p>If you do not use the -TB flag, trailing blanks are not trimmed.</p> <p>If you do use the -TB flag, trailing blanks are trimmed.</p>
-T{Z}	(IBM MVS and DB2) Prevents SQR for PeopleSoft from removing trailing zeros from the decimal portion of numeric columns.
-Vserver	(Sybase) Uses the named server.
-XB	(Non-Microsoft Windows) Suppresses the SQR banner and the <i>SQR.... End of Run</i> message.
-XC	(Callable SQR) Suppresses the database commit when the report has finished running.
-XCB	(Microsoft Windows) Does not use the communication box. Requests for input are made in Microsoft Windows dialog boxes.

Flag	Description
-XI	Prevents user interaction during a program run. If an ASK or INPUT command requires user input, an error is produced and the program ends.
-XL	Prevents SQR for PeopleSoft from signing in to the database. Programs that you run in this mode cannot contain SQL statements. -XL enables you to run SQR for PeopleSoft without accessing the database. You still must supply at least an empty slash on the command line as a placeholder for the connectivity information. For example: <code>sqr myprog / -xl</code>
-XLFF	Prevents a trailing form feed.
-XMB	(Microsoft Windows) Disables the error message display so that you can run a program without interruption from error message boxes. Error messages are sent to an .err file. See the -E flag.
-XNAV	Prevents SQR for PeopleSoft from creating the navigation bar in .htm files that are generated with -PRINTER:HT. This occurs when only a single .htm file is produced. Multiple .htm files that are generated from a single report always contain the navigation bar.
-XP	(Sybase) Prevents SQR for PeopleSoft from creating temporary stored procedures. <u>See Chapter 2, "SQR Command Reference," BEGIN-SELECT, page 51.</u>
-XTB	Preserves the trailing blanks in a .lis file.
-XTOC	Suppresses the table of contents for the report. This flag is ignored when you specify either -PRINTER:EH or -PRINTER:HT.
-ZIF{file}	Sets the full path and name of the SQR initialization file, sqr.ini.
-ZIV	Invokes the SPF Viewer after generating the program.spf file. This flag implicitly invokes the -KEEP flag to create program.spf. In the case of multiple output files, only the first report file is passed to the viewer.

The usual SQR rules for variable assignment apply to list access. Assignment is prohibited only between date and numeric types. Assignment of a numeric column to a string variable returns the string representation of the numeric value; assignment of a date variable to a string variable returns the default-edit-mask representation of the date.

Literals

Literals are text or numeric constants:

- A single quote begins and ends a text literal. For example:

```
'Hello'
```

- Numerals 0–9 begin numeric literals.

Numerals that include digits with an optional decimal point and leading sign are acceptable numeric literals. For example, -543.21. Numeric literals can also be expressed in scientific form. For example, 1.2E5.

Sample Reports

For an overview of how an SQR report looks, view the sample reports that are stored in the SQR for PeopleSoft directory <PS_HOME>\bin\sqr\<database_platform>\SAMPLE (or SAMPLEW, for Windows). You can modify these reports to meet your needs.

Chapter 2

SQR Command Reference

This chapter describes and demonstrates each command in the SQR lexicon.

SQR Command Overview

The commands in this section follow the conventions listed in the section "SQR Syntax Conventions" in the previous chapter and use the abbreviations described in the following table:

Warning! If you are copying code directly from the examples in the PDF file, make sure you change the slanted quotes to regular quotes or you will receive an error message.

<i>Abbreviation</i>	<i>Description</i>	<i>Example</i>
txt_col	Text column retrievable from a database.	&address
num_col	Numeric column retrievable from a database.	&price
date_col	Date or datetime column retrievable from a database.	&date1
txt_var	String variable defined in a program.	\$your_name
num_var	Numeric variable defined in a program.	#total_cost
date_var	A variable explicitly defined as a date variable.	\$date1
any_lit	A literal of any type.	'abc' 12
any_var	A variable of any type.	\$string #number \$date

Abbreviation	Description	Example
any_col	A column of any type.	&string &number &date
txt_lit	Text literal defined in a program.	'Company Confidential'
num_lit	Numeric literal defined in a program.	12345.67
int_lit	Integer literal defined in a program.	12345
nn	Integer literal used as an argument to a command.	123
position	The position qualifier, which consists of the line, column, and length specification. The minimum position, (), means to use the current line and column position on the page for the length of the field being printed.	(5,10,30)

SQR Commands

The following sections discuss the SQR commands in alphabetical order.

ADD

Syntax

```
ADD{src_num_lit | _var |
_col} TO dst_num_var
[ROUND=nn]
```

Description

Adds one number to another.

The source value is added to the destination variable and the result is placed in the destination. The source is always first and the destination is always second.

Example

The following example illustrates the ALTER-COLOR-MAP command:

```
begin-setup
  declare-color-map
    light_blue = (193, 222, 229)
  end-declare
end-setup

begin-program
  alter-color-map name = 'light_blue' value = (193, 233, 230)

  print 'Yellow Submarine' ()
    foreground = ('yellow')
    background = ('light_blue')

  get-color print-text-foreground = ($print-foreground)
  set-color print-text-foreground = ('purple')
  print 'Barney' (+1,1)
  set-color print-text-foreground = ($print-foreground)
end-program
```

See Also

DECLARE-COLOR-MAP, SET-COLOR, GET-COLOR

Keyword	Value
MONEY-EDIT-MASK	SQR uses the same default as the NUMBER-EDIT-MASK keyword.
DATE-EDIT-MASK	SQR uses the default database date format. See the Date Time section for more details.
INPUT-DATE-EDIT-MASK	SQR uses a default date edit mask with the INPUT command. See the Sample Date Edit Masks table for a listing of the date edit mask.
MONEY-SIGN	'\$'
MONEY-SIGN-LOCATION	LEFT
THOUSAND-SEPARATOR	','
DECIMAL-SEPARATOR	'.'
DATE-SEPARATOR	'/'
TIME-SEPARATOR	':'
EDIT-OPTION-NA	'NA'
EDIT-OPTION-AM	'am'
EDIT-OPTION-PM	'pm'
EDIT-OPTION-BC	'bc'
EDIT-OPTION-AD	'ad'
DAY-OF-WEEK-CASE	EDIT
DAY-OF-WEEK-FULL	('Sunday', 'Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday')
DAY-OF-WEEK-SHORT	('Sun', 'Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat')

<i>Argument</i>	<i>Values</i>	<i>Description</i>
DATA-LABELS	Yes No	If NewGraphics is set to Yes, use this argument to specify whether SQR prints the numeric value above the individual data points. Specify NO to suppress the numeric values.
COLOR-PALETTE	palette_name	If NewGraphics is set to Yes, use this argument to specify the name of the color palette to be used to color the individual data points in each chart (for example, bar, slice, point). A valid SQR color-palette must be defined to use COLOR-PALETTE.
ITEM-COLOR	ChartBackground ChartForeground HeaderBackground HeaderForeground FooterBackground FooterForeground LegendBackground LegendForeground ChartAreaBackground ChartAreaForeground PlotAreaBackground	If NewGraphics is set to Yes, use this argument to specify the color of an individual item in a chart. Specify a chart item and a valid (r,g,b) color to set the color of the chart item.
TITLE	NONE text	Specifies a title for the chart. That text is placed at the top of the chart.
SUB-TITLE	NONE text	Specifies a subtitle for the chart. That text is placed below the title regardless of whether TITLE is specified.
FILL	GRAYSCALE COLOR CROSS-HATCH NONE	Specifies the type of filling that is applied to the shapes (bars, pie-segments, and so on) in the chart. GRAYSCALE varies the density of black dots. COLOR sends color instructions to the current printer. If the current printer does not support color, then color can appear in a GRAYSCALE fashion. CROSSHATCH uses patterns to fill the shapes representing each data set. With NONE, all graph shapes are filled with white.
3D-EFFECTS	YES NO	Specifies whether the chart depth appears with 3-D effects. If this argument is set to NO, the chart is displayed in the default 2D mode.

Argument	Values	Description
BORDER	YES NO	If this argument is set to YES, a border is drawn around the chart. If it is set to NO, no border is displayed around the chart.
POINT-MARKERS	YES NO	Specifies whether point markers appear on line charts. If this argument is set to YES, point markers appear on line charts. If it is set to NO, point markers do not appear.
TYPE	LINE PIE BAR, STACKED-BAR 100%-BAR OVERLAPPED-BAR FLOATING-BAR HISTOGRAM AREA STACKED-AREA 100%-AREA XY-SCATTER-PLOT HIGH-LOW-CLOSE	Specifies the type of chart. <i>See Enterprise PeopleTools 8.50 PeopleBook: SQR for PeopleSoft Developers, "Using Business Charts."</i>
LEGEND	YES NO	Specifies whether to display a legend.
LEGEND-TITLE	NONE text	Specifies the title for the legend. If this argument is set to NONE, no title is displayed in the legend box.
LEGEND-PLACEMENT	CENTER-RIGHT CENTER-LEFT UPPER-RIGHT UPPER-LEFT UPPER-CENTER LOWER-RIGHT LOWER-LEFT LOWER-CENTER	Places the legend in the specified location on the chart. The first portion of the placement parameter (CENTER, UPPER, or LOWER) is the vertical position, and the second portion (RIGHT, LEFT, or CENTER) is the horizontal.
LEGEND-PRESENTATION	INSIDE OUTSIDE	Specifies where the legend appears on the chart. If this argument is set to INSIDE, the legend is presented inside the area defined by the two axes. If it is set to OUTSIDE, the legend is presented within the chart border, but outside of the region represented by the two axes.
PIE-SEGMENT-QUANTITY-DISPLAY	YES NO	Specifies whether quantity is presented for each pie segment. If this argument is set to YES, the quantity is presented.
PIE-SEGMENT-PERCENT-DISPLAY	YES NO	Specifies whether the percent-of-total number is presented for each pie segment. If this argument is set to YES, the percent-of-total figures is presented.

<i>Argument</i>	<i>Values</i>	<i>Description</i>
PIE-SEGMENT-EXPLODE	NONE MAX MIN USE-3RD-DATA-COLUMN	Controls which pie segments are exploded (selected) within the pie chart. MAX selects the largest segment. MIN selects the smallest segment. USE-3RD-DATA-COLUMN uses the third field in the DATA-ARRAY to determine which pie segments are exploded. This third field should be a CHAR and values of YES or Y indicate that the segment should be exploded.
X-AXIS-LABEL or Y-AXIS-LABEL	NONE text	Specifies a line of text to be displayed below (or alongside) the tick-mark labels on the axis.
X-AXIS-MIN-VALUE	AUTOSCALE number	Specifies the minimum value on the axis. If data values exist that are less than X-AXIS-MIN-VALUE, they are not displayed. AUTOSCALE directs SQR to calculate an appropriate minimum value.
Y-AXIS-MIN-VALUE	AUTOSCALE number	Specifies the minimum value on the axis. If data values exist that are less than Y-AXIS-MIN-VALUE, then they are not displayed. AUTOSCALE directs SQR to calculate an appropriate minimum value.
X-AXIS-MAX-VALUE	AUTOSCALE number	Specifies the maximum value on the axis. If data values exist that are greater than X-AXIS-MAX-VALUE, they are not displayed. AUTOSCALE directs SQR to calculate an appropriate maximum value.
Y-AXIS-MAX-VALUE	AUTOSCALE number	Specifies the maximum value on the axis. If data values exist that are greater than Y-AXIS-MAX-VALUE, they are not displayed. AUTOSCALE directs SQR to calculate an appropriate maximum value.
X-AXIS-SCALE or Y-AXIS-SCALE	LOG LINEAR	Specifies the scale for the axis. LOG specifies a logarithmic scale for the axis. Otherwise, the scale is linear.

Argument	Values	Description
X-AXIS-MAJOR-TICK- MARKS	YES NO	Specifies whether to display tick-marks for major increments on the X-axis. If this argument is set to YES, tick-marks appear on the axis between X-AXIS-MIN-VALUE and X-AXIS-MAX-VALUE, according to the X-AXIS-SCALE setting spaced by X-AXIS-MAJOR-INCREMENT.
Y-AXIS-MAJOR-TICK- MARKS	YES NO	Specifies whether to display tick-marks for major increments on the Y-axis. If this argument is set to YES, tick-marks appear on the axis between Y-AXIS-MIN-VALUE and Y-AXIS-MAX-VALUE, according to the Y-AXIS-SCALE setting spaced by Y-AXIS-MAJOR-INCREMENT.
X-AXIS-MINOR-TICK- MARKS	YES NO	Specifies whether to display tick-marks for minor increments on the X-axis. If this argument is set to YES, tick-marks appear on the axis between X-AXIS-MIN-VALUE and X-AXIS-MAX-VALUE, according to the X-AXIS-SCALE setting spaced by X-AXIS-MINOR-INCREMENT.
Y-AXIS-MINOR-TICK- MARKS	YES NO	Specifies whether to display tick-marks for minor increments on the Y-axis. If this argument is set to YES, tick-marks appear on the axis between Y-AXIS-MIN-VALUE and Y-AXIS-MAX-VALUE, according to the Y-AXIS-SCALE setting spaced by Y-AXIS-MINOR-INCREMENT.
X-AXIS-MAJOR-INCREMENT or Y-AXIS-MAJOR-INCREMENT	AUTOSCALE <i>number</i>	Specifies, for SQR, the increment used for spacing the major tick-marks on the axis. AUTOSCALE directs SQR to determine an appropriate increment.
X-AXIS-MINOR-INCREMENT or Y-AXIS-MINOR-INCREMENT	<i>number</i>	Specifies, for SQR, the increment used for spacing the minor tick-marks on the axis. These arguments must be set for the X-AXIS- MINOR-TICK-MARKS and the Y-AXIS- MINOR-TICK-MARKS to appear.
X-AXIS-TICK-MARK-PLACEMENT or Y-AXIS-TICK-MARK-PLACEMENT	INSIDE OUTSIDE BOTH	Specifies where to place the tick-marks on the axis. INSIDE (or OUTSIDE) directs SQR to place the tick-marks on the inside (or outside) of the axis only. BOTH directs SQR to draw the tick-marks such that they appear on both sides of the axis.

<i>Argument</i>	<i>Values</i>	<i>Description</i>
X-AXIS-GRID or Y-AXIS-GRID	YES NO	Specifies whether a grid line is drawn for each major tick-mark on the axis. If this argument is set toYES, a dashed grid line is drawn for each major tick-mark. If it is set to NO, no grid line is drawn on the axis.

Example

This code example declares a basic sales chart using DECLARE-CHART. For each region, the SUB-TITLE, DATA-ARRAY, and other elements are overridden to provide the chart with the specific features desired.

```

begin-setup

declare-chart base_sales_chart
chart-size           = (30, 20 )
title                = 'Quarterly Sales'
sub-title            = none
fill                 = color
3d-effects           = yes
type                 = stacked-bar
legend-title         = 'Product'
x-axis-grid          = yes
end-declare

end-setup

begin-program

print-chart base_sales_chart
sub-title            = 'Region I'
data-array           = reg1_sales
data-array-row-count = #rows_reg1
data-array-column-count = 2
y-axis-max-value     = #max_of_all_regions
y-axis-min-value     = #min_of_all_regions
legend               = no

print-chart base_sales_chart
sub-title            = 'Region II'
data-array           = reg2_sales
data-array-row-count = #rows_reg2
data-array-column-count = 2
y-axis-max-value     = #max_of_all_regions
y-axis-min-value     = #min_of_all_regions
legend               = no

end-program

begin-procedure chart_region_sales ($sub, $ary,
                                   #rows, #cols,
                                   #max_of_all_regions,
                                   #min_of_all_regions)

print-chart base_sales_chart (20, 15 )
sub-title            = $sub
data-array           = all sales
data-array-row-count = #rows
data-array-column-count = #cols
data-array-column-labels = ('Q1', 'Q2', 'Q3', 'Q4' )
y-axis-max-value     = #max_of_all_regions
y-axis-min-value     = #min_of_all_regions
chart-size           = (50, 30)

end-procedure

```

See Also

The PRINT-CHART command.

DECLARE-COLOR-MAP

Syntax

In the SETUP section:

```
DECLARE-COLOR-MAP
    color_name = ({rgb})
color_name = ({rgb})
.
.
.
END-DECLARE
```

Description

Defines colors in an SQR report.

The DECLARE-COLOR-MAP command in the BEGIN-SETUP section defines or redefines colors in an SQR report. You can define an endless number of entries.

Parameters

Parameter	Description
color_name	A <i>color_name</i> is composed of alphanumeric characters (A–Z, 0–9), the underscore (_) character, and the hyphen (-) character. It must start with an alphabetical (A–Z) character and is not case-sensitive. The name none is reserved and cannot be assigned a value. A name in the format (RGBredgreenblue) cannot be assigned a value. The name default is reserved and cannot be assigned a value. Default is used during execution when a referenced color is not defined in the runtime environment.

The DECLARE-LAYOUT command selects the proper fonts. In addition, the parameter interacts with PAPER-SIZE as follows:

- When you do not specify ORIENTATION=LANDSCAPE or the PAPER-SIZE dimensions, SQR creates a page with the dimensions set to 11 inch by 8.5 inch. This results in a page of 100 columns by 45 lines with 0.5 inch margins.
- When you specify PAPER-SIZE=(*paper_name*), the page orientation is set according to the *paper_name* specified. If you also specify ORIENTATION and the value differs from the PAPER-SIZE value, the ORIENTATION value overrides the PAPER-SIZE value.
- When you specify PAPER-SIZE=(*page_width*, *page_depth*), SQR *does not* swap the page width and page depth if ORIENTATION=LANDSCAPE.

Parameters

Parameter	Description
layout_name	A unique layout name to be used for referencing the layout and its attributes.
uom (unit of measure)	An optional suffix that denotes the unit of measure to apply to the preceding value.
paper_name	An option of PAPER-SIZE. This name is associated with predefined dimensions.

This table lists valid unit of measure suffixes:

Suffix	Meaning	Definition
dp	decipoint	0.001388 inch
pt	point	0.01388 inch
mm	millimeter	0.03937 inch
cm	centimeter	0.3937 inch
in	inch	1.0000 inch

This table lists valid paper names for the *paper_name* parameter.

Name	Width	Depth	Orientation
Letter	8.5 in	11 in	Portrait

Name	Width	Depth	Orientation
Legal	8.5 in	14 in	Portrait
A4	8.27 in	11.69 in	Portrait
Executive	7.25 in	10.5 in	Portrait
B5	7.17 in	10.12 in	Portrait
Com-10	4.125 in	9.5 in	Landscape
Monarch	3.875 in	7.5 in	Landscape
DL	4.33 in	8.66 in	Landscape
C5	6.378 in	9.016 in	Landscape

This table describes the arguments of the DECLARE-LAYOUT command:

Argument	Choice or Default UOM	Default Value	Description
PAPER-SIZE	inches	8.5 in, 11 in	Physical size of the page. The first parameter is the width of the page. The second parameter is the depth or length. It may also be a predefined name. (See the table of valid paper names.) Note that when ORIENTATION=LANDSCAPE, the default values are 11 in, 8.5 in.
FORMFEED	YES, NO	YES	Specifies whether formfeeds are to be written at the end of each page.

Argument	Choice or Default UOM	Default Value	Description
ORIENTATION	PORTRAIT, LANDSCAPE	PORTRAIT	Portrait pages are printed vertically. Landscape pages are printed horizontally. Printing in landscape for the printer type HPLASERJET requires landscape fonts.
LEFT-MARGIN	inches	0.5 in	Amount of blank space to leave at the left side of the page.
TOP-MARGIN	inches	0.5 in	Amount of blank space to leave at the top of the page.
RIGHT-MARGIN	inches	0.5 in	Amount of blank space to leave at the right side of the page. If you specify LINE-WIDTH or MAX-COLUMNS, you cannot use this parameter.
LINE-WIDTH	inches	7.5 in	Length of the line. If you specify RIGHT-MARGIN or MAX-COLUMNS, you cannot use this parameter.
MAX-COLUMNS	NA (not applicable)	75	Maximum number of columns in a line. If you specify RIGHT-MARGIN or LINE-WIDTH, you cannot use this parameter.
BOTTOM-MARGIN	inches	0.5 in	Amount of blank space to leave at the bottom of the page. If you specify PAGE-DEPTH or MAX-LINES, you cannot use this parameter.
PAGE-DEPTH	inches	10 in	Depth of the page. If you specify BOTTOM-MARGIN or MAX-LINES, you cannot use this parameter.

<i>Argument</i>	<i>Choice or Default UOM</i>	<i>Default Value</i>	<i>Description</i>
MAX-LINES	NA	60	Maximum number of lines printed on the page. If you specify PAGE-DEPTH or BOTTOM-MARGIN, you cannot use this parameter.
LINE-HEIGHT	points	12 pt	Size of each SQR line on the page. There are 72 points per inch. If LINE-HEIGHT is not specified, it follows the value for POINT-SIZE, if specified. The default value of 12 points yields 6 lines per inch. For the printer type LINEPRINTER, this value is used only to calculate the TOP-MARGIN and BOTTOM-MARGIN (for example, not in computing the position on the page).
CHAR-WIDTH	points	7.2 pt	Size of each SQR horizontal character column on the page (for example, the distance between the locations (1, 12) and (1, 13)). For the printer type LINEPRINTER, this value is used only to calculate the TOP-MARGIN and BOTTOM-MARGIN (not in computing the position on the page).

Example

This example illustrates the ability to specify these parameters using a different measurement system, such as metric:

```

!
declare-layout my-layout      ! Results in:

paper-size=(a4)              !  paper-size=(210mm, 297mm)

left-margin=12.7 mm         !   top-margin=12.7mm

right-margin=25.4 mm        !   left-margin=12.7mm
end-declare                 !   right-margin=25.4mm

                             !   bottom-margin=12.7mm

                             !   orientation=portrait

                             !   columns=67

                             !   lines=64

```

This example changes the page dimensions and also changes the left and right margins to be 1 inch:

```

!
declare-layout large-paper   ! Results in:

paper-size=(14, 11)         !  paper-size=(14in, 11in)

left-margin=1               !   top-margin=0.5in

right-margin=1              !   left-margin=1.0in
end-declare                 !   right-margin=1.0in

                             !   bottom-margin=0.5in

                             !   orientation=portrait

                             !   columns=120

                             !   lines=60

```

This example retains the default page dimensions and changes the left and right margins to be 1 inch:

```

declare-layout default      ! Results in:

left-margin=1               !  paper-size=(8.5in, 11in)

right-margin=1              !   top-margin=0.5in
end-declare                 !   left-margin=1.0in

                             !   right-margin=1.0in

                             !   bottom-margin=0.5in

                             !   orientation=portrait

                             !   columns=65

                             !   lines=60

```

This example changes the orientation to landscape; the default page dimensions (8.5in and 11in) are swapped, the columns and rows are recalculated, and all other values remain the same:

```

declare-layout default      ! Results in:

orientation=landscape      !   paper-size=(11in, 8.5in)
end-declare                !   top-margin=0.5in

                           !   left-margin=0.5in
                           !   right-margin=0.5in
                           !   bottom-margin=0.5in
                           !   orientation=landscape
                           !   columns=100
                           !   lines=45

```

This example changes the orientation to landscape; the default page dimensions (8.5in and 11in) are swapped, and the top margin is set to 1 inch:

```

declare-layout my_landscape ! Results in:

orientation=landscape      !   paper-size=(11in, 8.5in)

top-margin=1              !   top-margin=1.0in
end-declare                !   left-margin=0.5in

                           !   right-margin=0.5in
                           !   bottom-margin=0.5in
                           !   orientation=landscape
                           !   columns=100
                           !   lines=43

```

This example illustrates how to specify the page dimensions using one of the predefined names (note that the orientation has also changed because this example is an envelope):

```

declare-layout envelope    ! Results in:

paper-size=(com-10)        !   paper-size=(4.125in, 9.5in)
end-declare                !   top-margin=0.5in

                           !   left-margin=0.5in
                           !   right-margin=0.5in
                           !   bottom-margin=0.5in
                           !   orientation=landscape
                           !   columns=85
                           !   lines=18

```

See Also

DECLARE-REPORT

DECLARE-PRINTER**Syntax**

```

DECLARE-PRINTER printer_name
[FOR-REPORTS=(report_name1[,report_namei]...)]
[TYPE=printer_type_lit]
[INIT-STRING=initialization_string_txt_lit]
[RESET-STRING=reset_string_txt_lit]
[COLOR=color_lit]
[POINT-SIZE=point_size_num_lit]
[FONT-TYPE=font_type_int_lit]
[SYMBOL-SET=symbol_set_id_lit]
[STARTUP-FILE=file_name_txt_lit]
[PITCH=pitch_num_lit]
[FONT=font_int_lit]
[BEFORE-BOLD=before_bold_string_txt_lit]
[AFTER-BOLD=after_bold_string_txt_lit]
END-DECLARE

```

Description

Overrides the printer defaults for the specified printer type.

Each printer has a set of defaults as listed in the DECLARE-PRINTER Command Arguments table. The DECLARE-PRINTER command overrides these defaults.

Use the DECLARE-PRINTER command in the SETUP section to define the characteristics of the printer or printers to be used. If you need to change some of the arguments depending on the runtime environment, you can use the ALTER-PRINTER command in any part of the program except the PROGRAM and SETUP sections.

A program can contain no more than one DECLARE-PRINTER command for each printer type for each report. If you do not provide a printer declaration, the default specifications are used. The default printer attributes can be overridden by providing a DECLARE-PRINTER specification for each printer. The names are:

- DEFAULT-LP for line printer.
- DEFAULT-HP for HP LaserJet.
- DEFAULT-HT for HTML.
- DEFAULT-PS for PostScript

This table lists the arguments, provides the possible choices or measure, lists the default values, and describes the arguments.

<i>Argument</i>	<i>Choice or Measure</i>	<i>Default</i>	<i>Description</i>
FOR-REPORTS	NA (Not Applicable)	ALL	The name of the reports that use this printer definition. The default is ALL, for all reports. This argument is required only for a program with multiple reports. If you are writing a program that produces only a single report, you can ignore this argument.
TYPE	LINEPRINTER, POSTSCRIPT, HPLASERJET, HTML, LP, PS, HP, HT	LP	The output type specific to each printer. LINEPRINTER (LP) files can be viewed by a text editor. POSTSCRIPT (PS) files require you to know PostScript to understand what will be shown on the printer. HPLASERJET (HP) files are binary files and cannot be edited or viewed. HTML (HT) files can be viewed by a browser.
INIT-STRING	NA	(none)	Sends control or other characters to the printer at the beginning of the report. This parameter is designed primarily for the LINEPRINTER and has limited use with other printer types. Specify nondisplay characters by placing their decimal values inside angle brackets. For example, <27> is the ESC or escape character.

<i>Argument</i>	<i>Choice or Measure</i>	<i>Default</i>	<i>Description</i>
RESET-STRING	NA	(none)	Sends control or other characters to the printer at the end of the report. This parameter is designed primarily for the LINEPRINTER and has limited use with other printer types. Specify nondisplay characters by placing their decimal values inside angle brackets. For example, <27> is the ESC or escape character.
COLOR	Yes, No	No	Specifies whether this printer can print in color.
POINT-SIZE	points	12	This argument does not apply to LINEPRINTER printers. The beginning size of the selected font.
FONT-TYPE	PROPORTIONAL, FIXED	Depends on the font	This argument applies only to HPLASERJET printers and needs to be specified only for font types not defined in the Fonts Available for HP LaserJet Printers in SQR table.
SYMBOL-SET	HP defined sets	0U	This argument applies only to HPLASERJET printers. The default value of 0U is for the USASCII symbol set. For a complete list of the symbol sets, see the <i>HP LaserJet Technical Reference Manual</i> .

<i>Argument</i>	<i>Choice or Measure</i>	<i>Default</i>	<i>Description</i>
STARTUP-FILE	file name	POSTSCRI.STR	This argument applies only to POSTSCRIPT printers. This argument is used to specify an alternate startup file. Unless otherwise specified, the default startup file is located in the directory pointed to by the environment variable SQDIR.
PITCH	characters/inch	10	This argument is for HPLASERJET printers only. If you specify a fixed pitch font, you should also indicate the pitch.

<i>Argument</i>	<i>Choice or Measure</i>	<i>Default</i>	<i>Description</i>
FONT	font_number	3	<p>This is the font number of the typeface to use. For HPLASERJET printers, this is the typeface value as defined by Hewlett-Packard. For a complete list of the typeface numbers, see the <i>HP LaserJet Technical Reference Manual</i>.</p> <p>For POSTSCRIPT printers, SQR supplies a list of fonts and arbitrary font number assignments in the file POSTSCRI.STR. The font numbers are the same as those for HP LaserJet printers, wherever possible, so that you can use the same font number for reports to be printed on both types of printers. You can modify the font list in POSTSCRI.STR to add or delete fonts. Read the POSTSCRI.STR file for instructions. The Fonts Available for HP LaserJet Printers in SQR table lists the fonts available in SQR internally. The Fonts Available for PostScript Printers table lists the fonts available in the SQR POSTSCRI.STR file.</p>

Argument	Choice or Measure	Default	Description
BEFORE-BOLD	any string	(none)	<p>The BEFORE-BOLD and AFTER-BOLD arguments are for LINEPRINTER printers only. They specify the character string to enable or disable boldfacing. If the string contains blank characters, enclose it in single quote marks ('...'). To specify nonprintable characters, such as ESC, enclose the decimal value inside angle brackets as shown: BEFORE-BOLD=<27>[r ! Turn on bold AFTER-BOLD=<27>[u ! Turn it off</p> <p>These arguments work with the BOLD argument of the PRINT command.</p>
AFTER-BOLD	any string	(none)	See BEFORE-BOLD.

This table lists the fonts available in SQR for use with the FONT argument for HPLASERJET printer types.

Value	Typeface	Style
0	Line printer	Fixed
1	Pica	Fixed
2	Elite	Fixed
3	Courier	Fixed
4	Helvetica	Proportional
5	Times Roman	Proportional
6	Letter Gothic	Fixed
8	Prestige	Fixed

Value	Typeface	Style
11	Presentations	Fixed
17	Optima	Proportional
18	Garamondi	Proportional
19	Cooper Black	Proportional
20	Coronet Bold	Proportional
21	Broadway	Proportional
22	Bauer Bodini Black Condensed	Proportional
23	Century Schoolbook	Proportional
24	University Roman	Proportional

The font you choose—in orientation, typeface, and point size—must be an internal font, available in a font cartridge, or downloaded to the printer.

For fonts not listed in the Fonts Available for HP LaserJet Printers in SQR table, you must indicate the font style using the FONT-TYPE argument to ensure that the correct typeface is selected by the printer.

This table lists the fonts available in SQR for use with the FONT argument for PostScript printer types:

Value	Typeface	Boldface Type Available
3	Courier	Y
4	Helvetica	Y
5	Times Roman	Y
6	Avant Garde Book	NA (Not applicable)
8	Palatino Roman	Y
11	Symbol	NA

<i>Value</i>	<i>Typeface</i>	<i>Boldface Type Available</i>
12	Zapf Dingbats	NA
17	Zapf Chancery Medium Italic	NA
18	Bookman Light	NA
23	New Century Schoolbook Roman	Y
30	Courier Oblique	Y
31	Helvetica Oblique	Y
32	Times Italic	Y
33	Avant Garde Demi	NA
34	Avant Garde Book Oblique	NA
35	Avant Garde Demi Oblique	NA
36	Palatino Oblique	Y
37	New Century Schoolbook Italic	Y
38	Helvetica Narrow	Y
39	Helvetica Narrow Oblique	Y
40	Bookman Demi	NA
41	Bookman Light Italic	NA
42	Bookman Demi Italic	NA

Other type faces can be added to the POSTSCRI.STR file.

Different fonts are available in SQR for Windows when you are printing with Microsoft Windows printer drivers (using the -PRINTER:WP command-line flag). When you use the -PRINTER:WP flag, your report is sent directly to the default Microsoft Windows printer. To specify a nondefault Microsoft Windows printer, enter -PRINTER:WP:{printer name}. The {printer name} is the name assigned to your printer. For example, to send output to a Microsoft Windows printer named *NewPrinter*, you would use -PRINTER:WP:*NewPrinter*. If your printer name has spaces, enclose the entire argument in quotes.

Fonts are specified by number in the FONT qualifier of the ALTER-PRINTER command.

This table lists the fonts available when you are printing with Microsoft Windows printer drivers:

<i>Value</i>	<i>Windows Font/Name</i>	<i>Style</i>
3	Courier New	Fixed
300	Courier New	Bold
4	Arial	Proportional
400	Arial	Bold
5	Times New Roman	Proportional
500	Times New Roman	Bold
6	AvantGarde	Proportional
8	Palatino	Proportional
800	Palatino	Bold
11	Symbol	Proportional

Note. Fonts 6, 8, and 800 are not supplied with Microsoft Windows. You can get these fonts by purchasing the ADOBE Type Manager (ATM). The advantage of using ATM fonts is the compatibility for PostScript printer fonts. The Symbol font uses the SYMBOL_CHARSET instead of the usual ANSI_CHARSET character set. You can add more fonts by editing the appropriate Fonts section in the sqr.ini file.

See [Chapter 7, "Using the PSSQR.INI File and the PSSQR Command Line,"](#) page 301.

Parameters

<i>Parameter</i>	<i>Description</i>
printer_name	A unique name to be used for referencing a printer definition and its attributes.

Note. The DECLARE-PRINTER Command Arguments table describes the other arguments of the DECLARE-PRINTER command. The table lists the options, default values, and description of each of the arguments.

Example

The following example illustrates the DECLARE-PRINTER command:

```
declare-printer HP-definition      ! Default HP definition
type=HP                          ! for all reports
font=4                           ! Helvetica
symbol-set=12U                   ! PC-850 Multilingual
end-declare
declare-printer PS-Sales          ! PS definition
for-reports=(sales)              ! for the Sales report
type=PS
font=5                           ! Times-Roman
end-declare
```

See Also

ALTER-PRINTER, DECLARE-REPORT

DECLARE-PROCEDURE

Syntax

```
DECLARE-PROCEDURE
[FOR-REPORTS=(report_name1[,report_namei]...)]
[BEFORE-REPORT=procedure_name[(arg1[,argi]...)]]
[AFTER-REPORT=procedure_name[(arg1[,argi]...)]]
[BEFORE-PAGE=procedure_name[(arg1[,argi]...)]]
[AFTER-PAGE=procedure_name[(arg1[,argi]...)]]
END-DECLARE
```

Description

Declares procedures that are triggered when a specified event occurs.

The DECLARE-PROCEDURE command can be used to define SQR procedures that are to be invoked before or after a report is printed or before the beginning or end of each page.

Issue the DECLARE-PROCEDURE in the SETUP section. For multiple reports, you can use the command as often as required to declare procedures required by all the reports. If you issue multiple DECLARE-PROCEDURE commands, the last one takes precedence. In this way, you can use one command to declare common procedures for ALL reports and others to declare unique procedures for individual reports. The referenced procedures can accept arguments.

If no FOR-REPORTS are specified, ALL is assumed. Initially, the default for each of the four procedure types is NONE. If a procedure is defined in one DECLARE-PROCEDURE for a report, that procedure is used unless NONE is specified.

Use the USE-PROCEDURE command to change the procedures to be used at runtime. To disable a procedure, specify NONE in the USE-PROCEDURE statement.

Parameters

<i>Parameter</i>	<i>Description</i>
FOR-REPORTS	Specifies one or more reports that use the given procedures. This argument is required only for a program with multiple reports. If you are writing a program that produces only a single report, you can ignore this argument.
BEFORE-REPORT	Specifies a procedure to be executed at the time of the execution of the first command that causes output to be generated (PRINT). It can be used, for example, to create a report heading.
AFTER-REPORT	Specifies a procedure to be executed just before the report file is closed at the end of the report. It can be used to print totals or other closing summary information. If no report was generated, the procedure does not execute.
BEFORE-PAGE	Specifies a procedure to be executed at the beginning of every page, just before the first output command for the page. It can be used, for example, to set up page totals.
AFTER-PAGE	Specifies a procedure to be executed just before each page is written to the file. It can be used, for example, to display page totals. You can optionally specify arguments to be passed to any of the procedures. Arguments can be any variable, column, or literal.

Example

The following example illustrates the DECLARE-PROCEDURE command:

```

declare-procedure           ! These procedures will
before-report=report_heading ! be used by all reports

after-report=report_footing
end-declare
declare-procedure           ! These procedures will
for-reports=(customer)      ! be used by the customer

before-page=page_setup      ! report

after-page=page_totals
end-declare

```

See Also

USE-PROCEDURE

DECLARE-REPORT

Syntax

```

DECLARE-REPORT report_name
[TOC=toc_name]
[LAYOUT=layout_name]
[PRINTER-TYPE=printer_type]
END-DECLARE

```

Description

Defines reports and their attributes.

Issue the DECLARE-REPORT in the SETUP section.

You can use the DECLARE-REPORT command to declare one or more reports to be produced in the application.

You must use this command when developing applications to produce more than one report.

Multiple reports can share the same layout and the same printer declarations or each report can use its own layout or printer definitions if the report has unique characteristics.

When you are printing multiple reports, unless report names are specified using the -F command-line flag, the first report declared is generated with the name of *program.lis*, where *program* is the application name.

Additional reports are generated with names conforming to the rules dictated by the OUTPUT-FILE-MODE setting in the *sqr.ini* file.

When the -KEEP or -NOLIS flags are used, the first intermediate print file (.spf file) is generated with a name of *program.spf* and additional reports are generated with names conforming to the rules dictated by the OUTPUT-FILE-MODE setting in the *sqr.ini* file.

Parameters

<i>Parameter</i>	<i>Description</i>
report_name	Specifies the name of the report.
TOC	Specifies the name of the table of contents for this report.
LAYOUT	Specifies the name of the layout for this report. If no layout is specified, the default layout is used.
PRINTER-TYPE	Specifies the type of printer to be used for this report. If no printer type is specified, the default, LINEPRINTER, is used for this report. If no DECLARE-PRINTER is specified, DEFAULT-LP is used. Valid values for PRINTER-TYPE are HT, HP, PD, PS, LP, HTML, HPLASERJET, POSTSCRIPT, and LINEPRINTER.

Example

The following example illustrates the DECLARE-REPORT command:

```

declare-layout customer_layout
    left-margin
    right-margin
end-declare

declare-layout summary_layout
    orientation=landscape
end-declare

declare-report customer_detail

toc=detailed

layout=customer_layout

printer-type=postscript
end-declare
declare-report customer_summary

layout=summary_layout

printer-type=postscript
end-declare
.
.
.
use-report customer_detail

...print customer detail...
use-report customer_summary

...print customer summary...

```

See Also

USE-REPORT, DECLARE-LAYOUT, DECLARE-PRINTER, DECLARE-TOC

DECLARE-TOC**Syntax**

```
DECLARE-TOC toc_name
[FOR-REPORTS=(report_name1[,report_namei]...)]
[DOT-LEADER=YES|NO]
[INDENTATION=position_count_num_lit]
[BEFORE-TOC=procedure_name[(arg1[,argi]...)]]
[AFTER-TOC=procedure_name[(arg1[,argi]...)]]
[BEFORE-PAGE=procedure_name[(arg1[,argi]...)]]
[AFTER-PAGE=procedure_name[(arg1[,argi]...)]]
[ENTRY=procedure_name [(argi [,argi] ...)]]
END-DECLARE
```

Description

Defines the table of contents and its attributes.

Use DECLARE-TOC in the SETUP section.

You can use the DECLARE-TOC command to declare one or more tables of contents for the application.

A table of contents can be shared between reports.

Parameters

<i>Parameter</i>	<i>Description</i>
toc_name	Specifies the name of the table of contents.
FOR-REPORTS	Specifies one or more reports that uses this table of contents.
DOT-LEADER	Specifies whether a dot leader precedes the page number. The default setting is NO.
INDENTATION	Specifies the number of spaces by which each level is indented. The default setting is 4.
BEFORE-TOC	Specifies a procedure to be executed before before the application generates the table of contents. If no table of contents is generated, the procedure does not execute.
AFTER-TOC	Specifies a procedure to be executed after the application generates the table of contents. If no table of contents is generated, the procedure does not execute.

<i>Parameter</i>	<i>Description</i>
BEFORE-PAGE	Specifies a procedure to be executed at the start of every page.
AFTER-PAGE	Specifies a procedure to be executed at the end of each page.
ENTRY	<p>Specifies a procedure that is executed to process each table of contents entry (instead of SQR doing it for you). When this procedure is invoked, the following SQR-reserved variables are populated with data about the TOC entry:</p> <p>#SQR-TOC-LEVEL contains the level.</p> <p>#SQR-TOC-TEXT contains the text.</p> <p>#SQR-TOC-PAGE contains the page number.</p> <p>These are global variables. If the procedure is local, you must precede it with an underscore (for example, #_sqr-toc-page). These three SQR-reserved variables are valid only within the scope of the ENTRY procedure. They can be referenced outside the scope, but their contents are undefined.</p>

Example

The following example illustrates the DECLARE-TOC command:

```
begin-setup
  declare-toc common
    for-reports=(all)
    dot-leader=yes
    indentation=2
end-declare
end-setup
.
.
.
toc-entry level=1 text=$Chapter
toc-entry level=2 text=$Heading
.
.
```

See Also

BEGIN-FOOTING, BEGIN-HEADING, DECLARE-REPORT, TOC-ENTRY

DECLARE-VARIABLE

Syntax

```
DECLARE-VARIABLE
[DEFAULT-NUMERIC={DECIMAL[(prec_lit)]|FLOAT|INTEGER}]
[DECIMAL[(prec_lit)]num_var[(prec_lit)] [num_var
[(prec_lit)]... ]
[FLOAT num_var[num_var]... ]
[DATE date_var[date_var]... ]
[INTEGER num_var[num_var]... ]
[TEXT string_var[string_var]... ]
END-DECLARE
```

Description

Enables the user to explicitly declare a variable type.

You can set the default numeric type externally, using the -DNT command-line flag or the DEFAULT-NUMERIC setting in the Default-Settings section of the sqr.ini file. However, the setting in the DECLARE-VARIABLE command takes precedence over all other settings. If the command has not been used, then the -DNT command-line flag takes precedence over the setting in the sqr.ini file.

In addition to FLOAT, INTEGER, and DECIMAL, you can set DEFAULT-NUMERIC in the sqr.ini file and -DNT command-line flag to V30. With V30, the program acts in the same manner as in pre-version 4.0 releases; that is, all variables are FLOAT. V30 is not a valid setting for the DEFAULT-NUMERIC setting in the DECLARE-VARIABLE command.

The DECLARE-VARIABLE command enables the user to determine the type of variables to use. This command can appear only in the SETUP section or as the first statement of a local procedure. The placement of the command affects its scope. When used in the SETUP section, it affects all variables in the entire program. Alternately, when it is placed in a local procedure, its effect is limited to the scope of the procedure. If the command is in both places, the local declaration takes precedence over the SETUP declaration.

In addition to declaring variables, this command enables the default numeric type to be specified using the DEFAULT-NUMERIC setting as FLOAT, INTEGER, or DECIMAL. When dealing with money or where more precision is required, use the DECIMAL qualifier.

The DECLARE-VARIABLE command, the -DNT command-line flag, and the DEFAULT-NUMERIC setting in the sqr.ini file affect the way numeric literals are typed. If V30 is specified, then all numeric literals are FLOAT (just as in pre-version 4.0 releases); otherwise, the use or lack of a decimal point determines the type of the literal as either FLOAT or INTEGER, respectively. Finally, not specifying the DECLARE-VARIABLE command, the -DNT command-line flag, and the DEFAULT-NUMERIC setting in the sqr.ini file is the same as specifying V30.


```

begin-setup
  declare-variable
    default-numeric=float
    decimal   #decimal(10)
    integer   #counter
    date      $date
  end-declare
end-setup
.
.
let $date = strtodate('Jan 01 2004','Mon DD YYYY')
print $date (1,1)
position (+2,1)

let #counter = 0
while #counter < 10
  let #decimal = sqrt(#counter)
  add 1 to counter
  print #decimal (+1,1) 9.999999999
end-while

do sub1($date, 'day', 10)

do sub2

.
.
begin-procedure sub1(:$dvar, $units, #uval)
declare-variable
  date $dvar
  integer #uval
end-declare
let $dvar = dateadd($dvar, $units, #uval)
print $dvar (+1,1)
position (+2,1)
end-procedure
.
.
begin-procedure sub2 LOCAL
declare-variable
  date $mydate
end-declare
let $mydate = dateadd($_date, 'year', 5)
print $mydate (+1,1)
position (+2,1)
end-procedure
.
.

```

#DEFINE

Syntax

```
#DEFINE substitution_variablevalue
```

Description

Declares a value for a substitution variable within the body of the report (rather than using the ASK command).

#DEFINE is useful for specifying constants such as column locations, printer fonts, or any number or string that is used in several locations in the program. When the value of the number or string must be changed, you need only change the #DEFINE command. All references to that variable change automatically, which makes modifying programs much simpler.

If the ASK command is used to obtain the value of a substitution variable that has already been defined, ASK uses the previous value and the user is not prompted. This enables you to predefine some variables and not others. When the report runs, ASK requests values for only those variables that have not had a value assigned.

You can use #DEFINE commands inside an include file. This is a method of gathering commonly used declarations into one place, and reusing them for more than one report.

The value in the #DEFINE command can have embedded spaces and does not need to be enclosed within quotes. The entire string is used as is.

The #DEFINE command cannot be broken across program lines.

Parameters

<i>Parameter</i>	<i>Description</i>
substitution_variable	The variable to be used as the substitution variable. The substitution variable is used to substitute any command, argument, or part of a SQL statement at compile time.
Value	The value to be substituted.

Example

This code example defines several constants:

```
#define page_width 8.5
#define page_depth 11
#define light LS^10027
#define bold LS^03112
#define col1 1
#define col2 27
#define col3 54
#define order_by state, county, city, co_name
```

This code example from a report uses the definitions from the preceding example:

```

begin-setup

declare-printer contacts

    type=hp

    paper-size=({page_width}, {page_depth})

end-declare
end-setup
begin-heading 5
    print 'Company Contacts'      (1,1) center
    print 'Sort: {order_by}'      (2,1) center
    print 'Company'               (4,{col1})
    print 'Contact'               (4,{col2})
    print 'Phone'                 (4,{col3})
end-heading
begin-procedure main
begin-select
company      (1,{col1})
    print '{bold}'      (0,{col2})      ! Print contact in boldface.
contact      ( )
    print '{light}'      ( )           ! Back to lightface.
phone        (0,{col3})                ! Note: There must be enough
    next-listing                    ! space between col2
from customers                    ! and col3 for both
order by {order_by}                ! font changes and the
end-select                        ! contact field.
end-procedure

```

See Also

ASK

DISPLAY

Syntax

```

DISPLAY {any_lit|_var|_col}
[[:$]edit_mask|NUMBER|MONEY|DATE][NOLINE]

```

Description

Displays the specified column, variable, or literal.

The DISPLAY command can display data to a terminal. The data is displayed to the current location on the screen. If you want to display more than one field on the same line, use NOLINE on each display except the last.

Dates can be contained in a date variable or column, or a string literal, column, or variable. When a date variable or column is displayed without an edit mask, the date appears in the following manner:

- For DATETIME columns and SQR DATE variables, SQR uses the format specified by the SQR_DB_DATE_FORMAT setting.

If this has not been set, SQR uses the first database-dependent format as listed in the Default Database Formats table.

- For DATE columns, SQR uses the format specified by the SQR_DB_DATE_ONLY_FORMAT setting.

If this has not been set, SQR uses the format listed in the DATE Column Formats table.

- For TIME columns, SQR uses the format specified by the SQR_DB_TIME_ONLY_FORMAT setting.

If this has not been set, SQR uses the format listed in the TIME Column Formats table.

When the program displays a date in a string literal, column, or variable using EDIT or DATE, the string uses the format specified by the SQR_DB_DATE_FORMAT setting, one of the database-dependent formats as listed in the Default Database Formats table, or the database-independent format SYYYYMMDD[HH24[MI[SS[NNNNN]]]].

If you require more control over the display, use the SHOW command.

Parameters

<i>Parameter</i>	<i>Description</i>
any_lit _var _col	The text, number, or date to be displayed.
edit_mask	Causes the field to be edited before being displayed. For additional information regarding edit masks, see the PRINT command.
NUMBER	Indicates that any_lit _var _col is to be formatted using the NUMBER-EDIT-MASK of the current locale. This option is not valid with date variables.
MONEY	Indicates that any_lit _var _col is to be formatted using the MONEY-EDIT-MASK of the current locale. This option is not valid with date variables.
DATE	Indicates that any_lit _var _col is to be formatted using the DATE-EDIT-MASK of the current locale. This option is not valid with numeric variables. If DATE-EDIT-MASK has not been specified, the date is displayed using the default format for that database (see the Default Database Formats table).
NOLINE	Suppresses the carriage return after the field is displayed.

Example

The following segments illustrate the various features of the DISPLAY command:

```
!
! Display a string using an edit mask
!
display '123456789' xxx-xx-xxxx
```

Produces the following output:

123-45-6789

```
!
! Display a number using an edit mask
!
display 1234567.89 999,999,999.99
```

Produces the following output:

1,234,567.89

```
!
! Display a number using the default edit mask (specified in SQR.INI)
!
display 123.78
```

Produces the following output:

123.780000

```
!
! Display a number using the locale default numeric edit mask
!
alter-locale number-edit-mask = '99,999,999.99'
display 123456.78 number
```

Produces the following output:

123,456.78

```
!
! Display a number using the locale default money edit mask
!
alter-locale money-edit-mask = '$$, $$$, $$$9.99'
display 123456.78 money
```

Produces the following output:

\$123,456.78

```
!
! Display a date column using the locale default date edit mask
!
begin-select
dcol
  from tables
end-select
alter-locale date-edit-mask = 'DD-Mon-YYYY'
display &dcol date
```

Produces the following output:

01-Jan-2004

```
!
! Display two values on the same line
!
display 'Hello' noline
display ' World'
```

Produces the following output:

```
Hello World

!
! Display two values on the same line with editing of the values
!
alter-locale money-edit-mask = '$$, $$$, $$9.99'
let #taxes = 123456.78
display 'You owe ' noline
display #taxes money noline
display ' in back taxes.'
```

Produces the following output:

You owe \$123,456.78 in back taxes.

See Also

- The SHOW command for information about screen control.
- The LET command for information about copying, editing, or converting fields.
- The EDIT parameter of the PRINT command for a description of the edit masks.
- The ALTER-LOCALE command for a description of the arguments NUMBER-EDIT-MASK, MONEY-EDIT-MASK, and DATE-EDIT-MASK.

DIVIDE

Syntax

```
DIVIDE {src_num_lit|_var|_col} INTO dst_num_var
[ON-ERROR={HIGH|ZERO}][ROUND=nn]
```

Description

Divides one number into another.

The source field is divided into the destination field and the result is placed in the destination. The source is always first, the destination always second.

When dealing with money-related values (dollars and cents), use decimal variables rather than float variables. Float variables are stored as double-precision floating-point numbers, and small inaccuracies can appear when dividing many numbers in succession. These inaccuracies can appear due to the way different hardware and software implementations represent floating point numbers.

Parameters

Parameter	Description
src_num_lit _var _col	Divided into the contents of dst_num_var.

<i>Parameter</i>	<i>Description</i>
dst_num_var	Contains the result after execution.
ON-ERROR	Sets the result to the specified number when a division by zero is attempted. If ON-ERROR is omitted and a division by zero is attempted, SQR halts with an error message.
ROUND	Rounds the result to the specified number of digits to the right of the decimal point. For float variables, this value can be from 0 to 15. For decimal variables, this value can be from 0 to the precision of the variable. For integer variables, this argument is not appropriate.

Example

The following example illustrates the DIVIDE command:

```
divide 37.5 into #price ! #price / 37.5
divide &rate into #tot on-error=high
divide #j into #subtot on-error=zero
```

Note. In the preceding example, High is the maximum value and zero is the lowest value.

See Also

ADD

The LET command for a discussion of complex arithmetic expressions.

DO

Syntax

```
DO procedure_name[(arg1[, argi]...)]
```

Description

Invokes the specified procedure.

When the procedure ends, processing continues with the command following the DO command. You can use arguments to send values to or receive values from a procedure.

Arguments passed by a DO command to a procedure must match in number:

- Database text columns, string variables, and literals can be passed to procedure string or date arguments.
- Database numeric columns, numeric variables, and numeric literals can be passed to procedure numeric arguments.

#ELSE

Syntax

```
#ELSE
```

Description

Compiles the code following the #ELSE command when a preceding #IF, #IFDEF, or #IFNDEF command is FALSE. (#ELSE is a compiler directive that works with the #IF, #IFDEF, and #IFNDEF compiler directives.)

See Also

The #IF, #IFDEF, and #IFNDEF commands for a description of each compiler directive.

ELSE

Syntax

```
ELSE
```

Description

ELSE is an optional command in an IF command.

See Also

The IF command for a description and example.

ENCODE

Syntax

```
ENCODE src_code_string_lit INTO dst_txt_var
```

Description

Assigns a nondisplay or display character to a string variable.

The ENCODE command can define nondisplay characters or escape sequences sent to an output device. These characters or sequences can perform complex output device manipulations. The ENCODE command also displays characters not in the keyboard. If your keyboard does not have the euro symbol, use the ENCODE feature to create a string variable for it.

The encode characters can be included in a report at the appropriate location using a PRINT or PRINT-DIRECT command.

Unicode (UCS-2) code points from <1> to <65535> can be defined in ENCODE command.

Parameters

<i>Parameter</i>	<i>Description</i>
src_code_string_lit	Specifies a string of characters to be encoded and placed in <i>dst_txt_var</i> .
dst_txt_var	Contains the result after execution.

Example

The following example illustrates the ENCODE command:

```
encode '<27>L11233' into $bold           ! Code sequence to turn bold on.
print $bold ( ) code-printer=lp
```

See Also

The chr function described in the Miscellaneous Functions table under the LET command.

PRINT, PRINT-DIRECT

END-DECLARE, END-DOCUMENT, END-EVALUATE, END-FOOTING, END-HEADING

Syntax

```
END-DECLARE
END-DOCUMENT
END-EVALUATE
END-FOOTING
END-HEADING
```

Description

Completes a section or paragraph.

The END-DECLARE command completes a paragraph started with:

```
DECLARE-CHART  
DECLARE-IMAGE  
DECLARE-LAYOUT  
DECLARE-PRINTER  
DECLARE-PROCEDURE  
DECLARE-REPORT  
DECLARE-VARIABLE
```

Other END- commands complete the corresponding BEGIN- command:

```
BEGIN-DOCUMENT  
EVALUATE  
BEGIN-FOOTING  
BEGIN-HEADING
```

Each command must begin on its own line.

Example

The following example illustrates the BEGIN-FOOTING and END-FOOTING commands:

```
begin-footing 2  
    print 'Company Confidential' (1) center  
end-footing
```

See Also

DECLARE-paragraph, BEGIN-section

#END-IF, #ENDIF

Syntax

```
#END-IF
```

Description

Ends an #IF, #IFDEF, or #IFNDEF command. (#END-IF is a compiler directive.)

#ENDIF (without the hyphen) is a synonym for #END-IF.

Example

The following example illustrates the #END-IF compiler directive:

```
#ifdef debuga  
    show 'DebugA: #j = ' #j edit 9999.99  
    show 'Cust_num = ' &cust_num  
#end-if
```

See Also

The #IF, #IFDEF, and #IFNDEF commands for a description of each compiler directive.

END-IF**Syntax**

```
END-IF
```

Ends an IF command.

See Also

The IF command for a description and example.

END-PROCEDURE, END-PROGRAM, END-SELECT, END-SETUP, END-SQL, END-WHILE**Syntax**

```
END-PROCEDURE  
END-PROGRAM  
END-SELECT  
END-SETUP  
END-SQL  
END-WHILE
```

Description

Completes the corresponding section or paragraph.

Each END- command completes the corresponding BEGIN- command:

```
BEGIN-PROCEDURE  
BEGIN-PROGRAM  
BEGIN-SELECT  
BEGIN-SETUP  
BEGIN-SQL  
WHILE
```

Each command must begin on its own line.

Example

The following example illustrates the END-PROGRAM command:

```
begin-program
  do main
end-program
```

See Also

BEGIN-section, WHILE

EVALUATE

Syntax

```
EVALUATE {any_lit|_var|_col}
```

This command is equivalent to case/switch in C or Java. The general format of an EVALUATE command is:

```
EVALUATE {any_lit|_var|_col}
WHEN comparison_operator {any_lit|_var|_col}
  SQR_Command...
  [BREAK]
[WHEN comparison_operator {any_lit|_var|_col}
  SQR_Command...
  [BREAK]]
[WHEN-OTHER
  SQR_Command...
  [BREAK]]
END-EVALUATE
```

Description

Determines the value of a column, literal, or variable and takes action based on that value.

The EVALUATE command is useful for branching to different commands depending on the value of a specified variable or column.

EVALUATE commands can be nested.

Evaluating a date variable or column with a string results in a date comparison (chronological, not a byte-by-byte comparison as is done for strings). The string must be in the proper format as follows:

- For DATETIME columns and SQR DATE variables in the format specified by the SQR_DB_DATE_FORMAT setting, SQR uses one of the database-dependent formats (see the Default Database Formats table), or the database-independent format 'SYYYYMMDD[HH24[MI[SS[NNNNNN]]]]'.
- For DATE columns, SQR uses the format specified by the SQR_DB_DATE_ONLY_FORMAT setting, or the format listed in the DATE Column Formats table.
- For TIME columns, SQR uses the format specified by the SQR_DB_TIME_ONLY_FORMAT setting, or the format as listed in the TIME Column Formats table.

Parameters

<i>Parameter</i>	<i>Description</i>
any_lit _var _col	Specifies a text or numeric column; a text, numeric, or date variable; or a text or numeric literal to be used in the evaluation. In short, this is an evaluation argument.
comparison_operator	Any valid comparison operator. See the list of operators in the #IF command. See Chapter 2, "SQR Command Reference," #IF, page 135 .
WHEN	Specifies the evaluation expression. The evaluation argument is compared with the argument, beginning from the first WHEN. If the expression is TRUE, SQR processes the commands after the WHEN. If the expression is FALSE, SQR processes the next WHEN expression. Each WHEN must be on its own line. If more than one WHEN expression appears directly before a set of commands, any one of them, if TRUE, causes the commands to execute.
BREAK	Causes an immediate exit of the EVALUATE command. Use BREAK at the end of a set of commands.
WHEN-OTHER	Signifies the start of default commands to be processed if all other WHEN expressions are FALSE. WHEN-OTHER must appear after all other WHEN expressions.

Example

The following example illustrates the EVALUATE command:

```

evaluate &code
  when = 'A'
    move 1 to #j
    break
  when = 'B'
  when = 'C'
    move 2 to #j      ! Will happen if &code is B or C.
    break
  when > 'D'
    move 3 to #j      ! Move 3 to #j and continue checking.
  when > 'H'
    add 1 to #j       ! Add 1 to #j and continue checking.
  when > 'W'
    add 2 to #j
    break
  when-other
    if isnull (&code)
      do null_code
    else
      move 0 to #j    ! Unknown code.
    end-if
    break
end-evaluate

```

See Also

The commands IF and LET for comparison operators.

EXECUTE (Sybase and Microsoft SQL Server)

Syntax

```
EXECUTE [-XC][ON-ERROR=procedure[(arg1[,argi]...)]]  
[DO=procedure[(arg1[,argi]...)]]  
[@#status_var=]stored_procedure_name  
[[@param=]{any_col|_var|_lit}[,...]]  
[INTO any_coldata_type[(length_int_lit)]  
[,...]][WITH RECOMPILE]
```

The syntax of this command generally follows that of the Sybase Transact-SQL EXECUTE command, with the exception of optional arguments and the INTO argument.

Description

Executes a stored procedure in Sybase or Microsoft SQL Server database.

If the stored procedure specified in **stored_procedure_name** contains a SELECT query, the EXECUTE command must specify an INTO argument to process the values from the query. If no INTO argument is specified, then the values from the query are ignored.

EXECUTE retrieves just the first row when the following instances are true:

- The DO procedure is not specified.
- The stored procedure, **stored_procedure_name** selects one or more rows.
- An INTO argument is specified.

This is useful for queries returning a single row.

Parameters

Parameter	Description
-XC	(Sybase only) Specifies that the EXECUTE command share the same connection as the DO=procedure it can invoke. This argument is required to share Sybase temporary tables.

<i>Parameter</i>	<i>Description</i>
ON-ERROR	<p>Declares an SQR procedure to execute if an error occurs. If ON-ERROR is omitted and an error occurs, SQR halts with an error message. For severe errors (for example, passing too few arguments) SQR halts, even if an error procedure is specified.</p> <p>You can specify arguments to be passed to the ON-ERROR procedure. Arguments can be any variable, column, or literal.</p>
DO	<p>Specifies an SQR procedure to execute for each row selected in the query. Processing continues until all rows have been retrieved. You can specify arguments to be passed to the procedure. Arguments can be any variable, column, or literal.</p>
@# <i>status_variable</i>	Returns the procedure status in the specified numeric variable. The status is returned only after selected rows are retrieved.
<i>stored_procedure_name</i>	Names the stored procedure to execute.
@ <i>param</i>	Names the parameter to pass to the stored procedure. Parameters can be passed with or without names. If used without names, they must be listed in the same sequence as defined in the stored procedure.
<i>any_lit _var _col</i>	Specifies the value passed to the stored procedure. It can be a string, numeric, or date variable, a previously selected column, a numeric literal, or a string literal.
OUT[PUT]	Indicates that the parameter receives a value from the stored procedure. The parameter must be a string, numeric, or date SQR variable. Output parameters receive their values only after rows selected have been retrieved. If you specify multiple output parameters, they must be in the same sequence as defined in the stored procedure.
INTO	<p>Indicates where to store rows that are retrieved from the stored procedure's SELECT statement. The INTO argument contains the names of the columns with data types and lengths (if needed). You must specify the columns in the same sequence and match the data type used in the stored procedure's SELECT statement.</p> <p>If the stored procedure contains more than one SELECT query, only the first query is described with the INTO argument. Rows from subsequent queries are ignored.</p>
WITH RECOMPILE	Causes the query to recompile each time it is executed rather than using the plan stored with the procedure. Normally, this is not required or recommended.

Example

The following code example invokes the stored procedure **get_total** with two parameters: a string literal and a string variable. The result from the stored procedure is stored in the variable *#total*.

```
execute get_total 'S. Q. Reporter' $State #Total Output
```

The following code example invokes the stored procedure **get_products** with two parameters. The stored procedure selects data into five column variables. The SQR procedure **print_products** is called for each row retrieved. The return status from the stored procedure is placed in the variable *#proc_return_status*.

```
execute do=print_products
  @#proc_return_status=
  get_products
  @prodcode=&code,          @max=#maximum
  INTO &prod_code          int,
      &description        char (45),
      &discount            float,
      &restock             char,
      &expire_date         datetime

begin-procedure print_products
print &prod_code           (+1,1)
print &description        (+5,45)
print &discount            (+5) edit 99.99
print &restock             (+5) match Y 0 5 Yes N 0 5 No
print &expire_date         (+5,) edit 'Month dd, yyyy'
end-procedure
```

EXIT-SELECT

Syntax

EXIT-SELECT

Description

Exits a SELECT paragraph immediately.

EXIT-SELECT causes SQR to jump to the command immediately following the END-SELECT command.

Use EXIT-SELECT when you need to end a query before all rows have been retrieved.

Example

The following example illustrates the EXIT-SELECT command:

```
begin-select
cust_num, co_name, contact, city, state, zip, employees
  add &employees to #tot_emps
  if #tot_emps >= 5000
    exit-select          ! Have reached required total emps.
  end-if
  do print_company
from customers order by employees desc
end-select
```

See Also

BEGIN-SELECT

EXTRACT

Syntax

```
EXTRACT {dst_txt_var|date_var} FROM
{{src_txt_lit|_var|_col}|{src_date_var|_col}}
{start_num_lit|_var}{length_num_lit|_var}
```

Description

Copies a portion of a string into a string variable.

You must specify the starting location of the string as an offset from the beginning of the string and its length. An offset of 0 (aero) begins at the leftmost character; an offset of 1 begins one character beyond that, and so on.

If the source is a date variable or column, it is converted to a string before the extraction according to the following rules:

- For DATETIME columns and SQR DATE variables, SQR specifies the SQR_DB_DATE_FORMAT setting.

If this has not been set, SQR uses the first database-dependent format as listed in the Default Database Formats table.
- For DATE columns, SQR uses the format specified by the SQR_DB_DATE_ONLY_FORMAT setting.

If this has not been set, SQR uses the format listed in the DATE Column Formats table.
- For TIME columns, SQR uses the format specified by the SQR_DB_TIME_ONLY_FORMAT setting.

If this has not been set, SQR uses the format as listed in the TIME Column Formats table.

If the destination is a date variable, the string extracted from the source must be in one of the following formats:

- The format specified by the SQR_DB_DATE_FORMAT setting.
- One of the database-dependent formats (see the Default Database Formats table).
- The database-independent format 'SYYYYMMDD[HH24[MI[SS[NNNNNN]]]]'.

Parameters

Parameter	Description
dst_txt_var date_var	Specifies a text or date variable into which the extracted string is placed.
{src_txt_lit _var _col} {src_date_var _col}	Specifies a text or date variable, column, or literal from which the string is to be extracted.

Parameter	Description
start_num_lit_var	Specifies starting location of the string to be extracted.
length_num_lit_var	Specifies length of the string to be extracted.

Example

The following example illustrates the EXTRACT command:

```
extract $state from $record 45 2
extract $foo from "SQR Rocks" 0 4 ! $foo='SQR'

code from &phone 0 3
extract $zip_four from &zip 5 4
extract $rec from $tape_block #loc #rec_len
```

Note. Oracle recommends that you not use the EXTRACT command when processing strings.

See Also

The substr function described in Miscellaneous Functions table under the LET command.

FIND

The PRINT command for information about default date and time formats

FIND

Syntax

```
FIND {{obj_txt_lit|_var|_col}}|{date_var|_col}} IN
{{src_txt_var|_col}}|{date_var|_col}}
{start_int_lit|_var} dst_location_int_var
```

Description

Determines the location of a character sequence within a string.

FIND searches the specified string for a character sequence and, if the string is found, returns its location as an offset from the beginning of the specified string. If the sequence is not found, FIND returns -1 in *dst_location_int_var*.

You must specify an offset from which to begin the search and supply a numeric variable for the return of the location.

If the source or search object is a date variable or column, it is converted to a string before the search according to the following rules:

- For DATETIME columns and SQR DATE variables, SQR uses the format specified by the SQR_DB_DATE_FORMAT setting.

If this has not been set, SQR uses the first database-dependent format as listed in the Default Database Formats table.

- For DATE columns, SQR uses the format specified by the SQR_DB_DATE_ONLY_FORMAT setting.
If this has not been set, SQR uses the format listed in the DATE Column Formats table.
- For TIME columns, SQR uses the format specified by the SQR_DB_TIME_ONLY_FORMAT setting.
If this has not been set, SQR uses the format as listed in the TIME Column Formats table.

Parameters

<i>Parameter</i>	<i>Description</i>
{obj_txt_lit _var _col} {date_var _col}	Specifies a text variable, column, or literal that is to be sought in <i>src_txt_var _col</i> .
{src_txt_var _col} {date_var _col}	Specifies a text variable or column to be searched.
start_int_lit _var	Specifies the starting location of the search.
dst_location_int_var	Specifies the returned starting location of the leftmost character of the matching text in { <i>src_txt_var _col date_var _col</i> }.

Example

The following example illustrates the FIND command:

```
find 'aw.2' in &code5 0 #loc
find ',' in &name 0 #comma_loc
if #comma_loc = -1
    ...comma not found...
```

See Also

The instr function described in the Miscellaneous Functions table under the LET command.

EXTRACT

The PRINT command for information about default date and time formats.

GET

Syntax

```
GET dst_any_var...FROM src_array_name(element)
[field[(occurs)]]...
```

Description

Retrieves data from an array and places it into a date, string, or numeric variable.

Parameters

<i>Parameter</i>	<i>Description</i>
<i>dst_any_var</i>	<p>Date, string, or numeric variables (not database columns) can be destination variables. Numeric variables (decimal, float, integer) are copied from number fields. String variables are copied from the char, text, or date field. Date variables are copied from the char, text, or date field.</p> <p>When a date field is copied to a string variable, SQR converts the date to a string in the format specified by the SQR_DB_DATE_FORMAT setting. If this has not been set, SQR uses the first database-dependent format listed in the Default Database Formats table.</p> <p>If the destination is a date variable, the string extracted from the source must be in the format specified by the SQR_DB_DATE_FORMAT setting, or one of the database-dependent formats (see the Default Database Formats table), or the database-independent format 'SYYYYMMDD[HH24[MI[SS[NNNNNN]]]]'.</p>
<i>src_array_name</i> (<i>element</i>)	If the array's field names are listed, SQR takes the values from the fields and occurrences specified. If the array's field names are not listed, the values are taken from consecutively defined fields in the array.
<i>field</i> [(<i>occurs</i>)]	Array element and field occurrence numbers can be numeric literals (such as 123) or numeric variables (such as #j). If no field occurrence is stated, occurrence zero is used.

Example

The following code example copies *\$name*, *\$start_date*, and *#salary* from the first three fields in the *#j*th element of the *emps* array:

```
get $name $start_date #salary from emps(#j)
```

The following code example copies *#city_tot* and *#county_tot* from the fields *cities* and *counties* in the *#j*th element of the *states* array:

```
get #city_tot #county_tot from states(#j) cities counties
```

The following code example copies *\$code* from the *#jth* occurrence of the *code* field in the *#nth* element of the *codes* array:

```
get $code from codes(#n) code(#j)
```

See Also

The PUT command for information about moving data into an array.

GET-COLOR

Syntax

```
GET-COLOR
[ PRINT-TEXT-FOREGROUND= ( { color_name_var | { rgb } } ) ]
[ PRINT-TEXT-BACKGROUND= ( { color_name_var | { rgb } } ) ]
```

Description

Retrieves the current colors.

The GET-COLOR command is allowed wherever the PRINT command is allowed. If the requested color settings do not map to a defined name, the name is returned as *RGBredgreenblue*, where each component is a three-digit number—for example, *RGB127133033*. You can use this format wherever you use a color name. The color name 'none' is returned if no color is associated with the specified area.

Parameters

Parameter	Description
PRINT-TEXT-FOREGROUND	Defines the color in which the text prints.
PRINT-TEXT-BACKGROUND	Defines the color to print as a background behind the text.
{ color_name_var }	A color_name is composed of alphanumeric characters (A–Z, 0–9), the underscore (_) character, and the hyphen (-) character. The name must start with an alphabetical (A–Z) character and is not case-sensitive. The name 'none' is reserved and cannot be assigned a value. A name in the format (RGBredgreenblue) cannot be assigned a value. The name 'default' is reserved and can be assigned a value. 'Default' is used during execution when a referenced color is not defined in the runtime environment.

Parameter	Description
<code>{rgb}</code>	<p><code>red_lit / _var / _col, green_lit / _var / _col, blue_lit / _var / _col</code> where each component is a value in the range of 000 to 255. In the BEGIN-SETUP section, only literal values are allowed.</p> <p>The default colors implicitly installed with SQR include:</p> <p>black=(0,0,0) white=(255,255,255) gray=(128,128,128) silver=(192,192,192) red=(255,0,0) green=(0,255,0) blue=(0,0,255) yellow=(255,255,0) purple=(128,0,128) olive=(128,128,0) navy=(0,0,128) aqua=(0,255,255) lime=(0,128,0) maroon=(128,0,0) teal=(0,128,128) fuchsia=(255,0,255)</p>

Example

The following example illustrates the GET-COLOR command:

```
begin-setup
  declare-color-map
    light_blue = (193, 222, 229)
  end-declare
end-setup

begin-program
  alter-color-map name = 'light_blue' value = (193, 233, 230)

  print 'Yellow Submarine' ( )
    foreground = ('yellow')
    background = ('light_blue')

  get-color print-text-foreground = ($print-foreground)
  set-color print-text-foreground = ('purple')
  print 'Barney' (+1,1)
  set-color print-text-foreground = ($print-foreground)
end-program
```

See Also

DECLARE-COLOR-MAP, ALTER-COLOR-MAP, SET-COLOR

GOTO

Syntax

```
GOTO  label
```

Description

Skips to the specified label.

Labels must end with a colon (:) and can appear anywhere within the same section or paragraph as the GOTO command.

Parameters

<i>Parameter</i>	<i>Description</i>
label	Specifies a label within the same section or paragraph.

Example

The following example illustrates the GOTO command:

```
begin-select
price
  if &price < #old_price
    goto next
  end-if
  print &price (2,13,0) edit 999,999.99
  ...
next:
  add 1 to #count
from products
end-select
```

GRAPHIC BOX, GRAPHIC HORZ-LINE, GRAPHIC VERT-LINE

Syntax

The GRAPHIC commands have the following syntax:

```

GRAPHIC ({line_int_lit|_var},{column_int_lit|_var},
{width_int_lit|_var})

BOX {depth_int_lit|_var}
[rule_width_int_lit|_var[shading_int_lit|_var]]

GRAPHIC ({line_int_lit|_var},{column_int_lit|_var},
{length_int_lit|_var})

HORZ-LINE [rule_width_int_lit|_var]

GRAPHIC ({line_int_lit|_var},{column_int_lit|_var},
{length_int_lit|_var})

VERT-LINE [rule_width_int_lit|_var]

```

Description

Draws a box or line.

After GRAPHIC commands execute, SQR changes the current print location to the starting location of the graphic. This is different from the way the PRINT command works.

The GRAPHIC command has the following variations:

- BOX
- HORZ-LINE
- VERT-LINE

The following sections describe the individual GRAPHIC commands:

Parameters

<i>Parameter</i>	<i>Description</i>
BOX	BOX draws a box of any size at any location on the page. Boxes can be drawn with any size rule and can be shaded or left empty.
<i>width and depth</i>	The width is the horizontal size in character columns; depth is the vertical size in lines. The top left corner of the box is drawn at the line and column specified. The bottom right corner is calculated using the width and depth. You can specify relative placement with (+), (-), or numeric variables, as with regular print positions.
<i>rule_width</i>	The default rule width is 2 decipoints (an inch has 720 decipoints). The top horizontal line is drawn just below the base of the line above the starting point. The bottom horizontal line is drawn just below the base of the ending line. Therefore, a one-line deep box surrounds a single line.
<i>shading</i>	A number between 1 and 100, specifying the percentage of shading to apply. 1 is very light, and 100 is black. If no shading is specified, the box is blank. Specify a rule-width of zero if you do not want a border.

Parameter	Description
HORZ-LINE	HORZ-LINE draws a horizontal line from the location specified, for the length specified. Horizontal lines are drawn just below the base.
<i>rule_width</i>	The default rule width is 2 decipoints.
VERT-LINE	VERT-LINE draws a vertical line from the location specified for the length (in lines) specified. Vertical lines are drawn just below the base line of the line position specified to just below the base line of the line reached by the length specified. To draw a vertical line next to a word printed on line 27, position the vertical line to begin on line 26, for a length of 1 line.
<i>rule_width</i>	The default rule width is 2 decipoints.

Example

The following code example shows the GRAPHIC BOX command:

```
graphic (1,1,66) box 58 20      ! Draw box around page
graphic (30,25,10) box 10      ! Draw a 10-characters-wide-by-10- characters-long⇒
  box
graphic (1,1,66) box 5 0 8      ! Draw 5 line shaded box (without border)
graphic (50,8,30) box 1        ! Draw box around 1 line
```

The following code example shows the GRAPHIC HORZ-LINE command:

```
graphic (4,1,66) horz-line 10   ! Put line under page heading
graphic (+1,62,12) horz-line    ! Put line under final total
```

The following code example shows the GRAPHIC VERT-LINE command:

```
graphic (1,27,54) vert-line     ! Draw lines between columns
graphic (1,52,54) vert-line
graphic (3,+2,4) vert-line 6    ! Red line the paragraph
```

See Also

The ALTER-PRINTER and DECLARE-PRINTER commands for information about setting and changing the FONT, FONT-TYPE, POINT-SIZE, and PITCH.

#IF

Syntax

```
#IF {txt_lit|num_lit}comparison_operator
{txt_lit|num_lit}
```

Description

Indicates that the commands following are to be compiled when the expression is TRUE. (#IF is a compiler directive.)

SQR has five compiler directives that enable different pieces of SQR code to be compiled, depending on the existence or value of substitution variables (not program variables, such as string, numeric, or date).

Substitution variables defined automatically for each -DEBUGxxx letter can also be used with the #IF, #IFDEF, and #IFDEF directives. They can turn entire sections of an SQR program on or off from the command line, depending on the -DEBUGxxx flag.

You can nest #IF, #IFDEF, and #IFDEF directives to a maximum of 10 levels.

The #IF, #IFDEF, and #IFDEF directives cannot be broken across program lines.

The following table lists the compiler directives.

<i>Directive</i>	<i>Example</i>	<i>Description</i>
#IF	#IF {option}='A'	Compiles the commands following the #IF directive if the substitution variable option is equal to 'A'. The test is not case-sensitive. Only one simple expression is allowed per #IF command.
#ELSE	#ELSE	Compiles the commands following the #ELSE directive when the #IF expression is FALSE.
#ENDIF	#ENDIF	Ends the #IF directive. #ENDIF can also be typed #END-IF (with a hyphen).
#IFDEF	#IFDEF option	Compiles the commands following the #IFDEF directive if the substitution variable option is defined.
#IFNDEF	#IFNDEF option	Compiles the command following the #IFNDEF directive if the substitution variable option is not defined.

Parameters

<i>Parameter</i>	<i>Description</i>
txt_lit num_lit	Any text or numeric literal.
comparison_operator	Any of the comparison operators as follows: =Equal !=Not Equal <>Not Equal <Less than >Greater than <=Less than or equal >=Greater than or equal

Example

The following example illustrates the #IF compiler directive:

```

begin-setup
  ask type 'Use Male, Female or Both (M,F,B)'
end-setup
begin-procedure Main
#if {type} = 'M'
  ...code for M here
#else
#if {type} = 'F'
  ...code for F here
#else
#if {type} = 'B'
  ...code for B here
#else
  show 'M, F or B not selected.   Report not created.'
  stop
#endif    ! for B
#endif    ! for F
#endif    ! for M

#ifdef  debug
  show 'DEBUG:  Cust_num = '  &cust_num  edit  099999
#endif

#ifdef  debugB                ! DebugB turned on with -DEBUGB on
  do test_procedure          ! SQR command line.
#endif

```

See Also

The #DEBUG command for information about the -DEBUG command-line flag.

IF

Syntax

```
IF logical_expression
```

IF commands have the following structure:

```
IF logical_expression  
SQR Command...  
[ELSE  
SQR Command...]  
END-IF
```

Description

Executes commands depending on the value of a condition.

The expression is evaluated as a logical TRUE or FALSE. A value or expression that evaluates to nonzero is TRUE.

Each IF command must have a matching END-IF command.

IF commands can be nested.

Comparing a date variable or column with a string results in a date comparison (chronological, not a byte-by-byte comparison as is done for strings). The string must be in the proper format as follows:

- For DATETIME columns and SQR DATE variables, SQR uses the format specified by the SQR_DB_DATE_FORMAT setting, one of the database-dependent formats (see the Default Database Formats table), or the database-independent format 'SYYYYMMDD[HH24[MI[SS[NNNNNN]]]]'.
- For DATE columns, SQR uses the format specified by the SQR_DB_DATE_ONLY_FORMAT setting, or the format listed in the Date Column Formats table.
- For TIME columns, SQR uses the format specified by the SQR_DB_TIME_ONLY_FORMAT setting, or the format as listed in the Time Column Formats table.

Parameters

<i>Parameter</i>	<i>Description</i>
logical_expression	Any valid logical expression. See the LET command for a description of logical expressions.

Example

The following example illustrates the IF command:

```
if &price > &old_price and instr(&code, 'M', 1) > 0
  add 1 to #price_count
  if #price_count > 50
    show 'More than 50 prices found.' noline
    input $x 'Continue? (Y/N)'
    if upper($x) = 'N'
      stop
    end-if
  end-if
else
  add 1 to #old_price_count
end-if
if #rows      ! Will be TRUE if #rows is non-zero.
  do print-it
end-if

if $date1 > 'Apr 21 2004 23:59'
  do past_due
end-if
```

See Also

The LET command for a description of logical expressions.

EVALUATE

#IFDEF

Syntax

```
#IFDEF substitution_variable
```

Description

Indicates that the following commands are to be compiled when the substitution variable has been declared by an ASK or #DEFINE command, or by the -DEBUG flag on the SQR command line. (#IFDEF is a compiler directive.)

Parameters

<i>Parameter</i>	<i>Description</i>
substitution_variable	The variable to be used as the substitution variable.

See Also

The #IF command for a description of each compiler directive.

#IFNDEF

Syntax

```
#IFNDEF substitution_variable
```

Description

Indicates that the following commands are to be compiled when the substitution variable has not been declared by an `ASK` or `#DEFINE` command, or by the `-DEBUG` flag on the SQR command line. (`#IFNDEF` is a compiler directive.)

Parameters

<i>Parameter</i>	<i>Description</i>
<code>substitution_variable</code>	The variable to be used as the substitution variable.

See Also

The `#IF` command for a description of each compiler directive.

#INCLUDE

Syntax

```
#INCLUDE filename_lit
```

Description

Includes an external source file in the SQR report specification.

You may want to keep commonly used routines in a single file and reference or *include* that file in programs that use the routine. For example, you might have a set of `#DEFINE` commands for different printers to control initialization, font changes, and page size declarations. You can reference the appropriate include file depending on which printer you want to use.

Include files can be nested up to four levels. Variable substitution scanning takes place before the `#INCLUDE` command is processed. This enables you to substitute all or part of the include file name at runtime, adding flexibility to controlling which file is included for the run.

Parameters

<i>Parameter</i>	<i>Description</i>
filename_lit	A file name that is valid for the platform on which the application is to be compiled.

Example

The following example illustrates the #INCLUDE command:

```
#include 'gethours.dat'      ! Common procedure.
#include 'XYZheader.dat'     ! Common report heading for XYZ Company.
#include 'printer{num}.dat'  ! Include printer definitions for
                             ! printer {num}, which is passed
                             ! on the command line:
                             ! SQR REPLA SAM/JOE 18
                             ! where 18 is the arbitrary number
                             ! assigned your printer
                             ! definition file, 'printer18.dat'.
                             ! The report would contain the
                             ! command: ASK num
                             ! in the SETUP section, preceding
                             ! this #include statement.
```

INPUT

Syntax

```
INPUT input_var[MAXLEN=nn][prompt]
[TYPE={CHAR|TEXT|NUMBER|INTEGER|DATE}]
[STATUS=num_var][NOPROMPT][BATCH-MODE]
[FORMAT={txt_lit|_var|_col}]
```

Description

Accepts data entered by the user at a terminal.

Use MAXLEN to prevent the user from entering data that is too long. If an INSERT or UPDATE command references a variable for which the length is greater than the length defined in the database, the SQL is rejected and SQR halts. If the maximum length is exceeded, the terminal beeps (on some systems, this may cause the screen to flash instead).

If *prompt* is omitted, SQR uses the default prompt. Enter [*\$|#*]*var*: . In either case, a colon (:) and two spaces are added to the prompt.

Specifying TYPE causes data type checking to occur. If the string entered is not the type specified, the terminal beeps and an error message is displayed. The INPUT command is then re-executed. If TYPE=DATE is specified, then *input_var* can be a date or text variable; however, TYPE=DATE is optional if *input_var* is a date variable. If a numeric variable is used, it is validated as a numeric variable. The types CHAR, TEXT, and DATE are invalid types. The data types supported are described in the following table:

Data Type	Description
CHAR, TEXT	Any character. This is the default datatype.
NUMBER	A floating point number in the format [+ -]9999.999[E[+ -]99]
INTEGER	An integer in the format [+ -]99999
DATE	A date in one of the following formats: <ul style="list-style-type: none"> MM/DD/YYYY [BC AD] [HH:MI[:SS[.NNNNNN]] [AM PM]] MM-DD-YYYY [BC AD] [HH:MI[:SS[.NNNNNN]] [AM PM]] MM.DD.YYYY [BC AD] [HH:MI[:SS[.NNNNNN]] [AM PM]] SYYYMMDD[HH24[MI[SS[NNNNNN]]]]

Specifying STATUS causes the INPUT command to finish regardless of what the user enters. No error message is displayed. A nonzero error code is stored in the indicated numeric variable if the length or datatype entered is incorrect.

The following table lists the values of the INPUT command's STATUS argument:

Status Value	Indicates
0	Successful.
1	Bad type (did not match the datatype of TYPE).
2	Too long (longer than MAXLEN or the input for an INTEGER variable is < -2147483648 or > +2147483647).
3	No arguments remain on the command line. The command was ignored.

By using NOPROMPT and STATUS with the SHOW command, you can write a sophisticated data entry routine.

FORMAT can be used only with dates. It can be a date edit mask or the keyword DATE. Use the keyword DATE if the date must be in the format as specified with the INPUT-DATE-EDIT-MASK setting for the current locale. If FORMAT has not been set, use a database-independent format for the data as listed in the datatypes table.

Parameters

<i>Parameter</i>	<i>Description</i>
input_var	Specifies a text, numeric, or date variable for the input data.
MAXLEN	Specifies the maximum length for the data.
prompt	Specifies the prompt displayed to the user.
TYPE	Specifies the datatype required for the input.
STATUS	Specifies a numeric variable for a return status code.
NOPROMPT	Prevents the prompt from being displayed before the INPUT command is processed.
BATCH-MODE	If BATCH-MODE is specified and no more arguments are in the command line, a value of 3 is returned in the STATUS variable and the user is not prompted for input.
FORMAT	Specifies the format for entering a date. The Date Edit Format Codes table lists date edit format codes.

Example

The following example shows several INPUT commands:

```
input $state maxlen=2 'Please enter state abbreviation'
input #age 'Enter lower age boundary' type=integer
input $start_date 'Enter starting date for report' type=date
input $date_in format='Mon dd yyyy'
input $date format=date
```

The following example shows another INPUT command:

```
show clear-screen (5,32) reverse 'CUSTOMER SUMMARY' normal
Try_again:
show (12,20) 'Enter Start Date: ' clear-line
input $start-date noprompt status=#istat type=date
if #istat != 0
    show (24,1) 'Please enter date in format DD-MON-YY' beep
    goto try_again
end-if
show (24,1) clear-line           ! Clear error message line.
```

The following example illustrates the use of the BATCH-MODE option:

```

begin-program
while (1)
  input $A status=#stat batch-mode
  if #stat = 3
    break
  else
    do procedure ($a)
  end-if
end-while
end-program

```

See Also

ALTER-LOCALE

The INPUT-DATE-EDIT-MASK setting in the chapter "Using the PSSQR.INI File and the PSSQR Command Line."

LAST-PAGE

Syntax

```
LAST-PAGE position [pre_txt_lit[post_txt_lit]]
```

Description

Places the last page number on each page, as in *page n of m*.

The text strings specified in *pre_txt_lit* and *post_txt_lit* are printed immediately before and after the number.

Using LAST-PAGE causes SQR and SQRT to delay printing until the last page has been processed so that the number of the last page is known.

Parameters

<i>Parameter</i>	<i>Description</i>
position	Specifies the position for printing the last page number. See the POSITION command for a description of the <i>position</i> parameter.
pre_txt_lit	Specifies a text string to be printed before the last page number.
post_txt_lit	Specifies a text string to be printed after the last page number.

Example

The following example illustrates the LAST-PAGE command:

```
begin-footing 1
  page-number      (1,37) 'Page '      ! Will appear as
  last-page        ()  ' of ' '.'      ! "Page 12 of 25."
end-footing
```

See Also

PAGE-NUMBER, BEGIN-HEADING, BEGIN-FOOTING

LET

Syntax

```
LET dst_var=expression
```

Description

Assigns the value of an expression to a string, numeric, or date variable.

Valid expressions are formed as a combination of operands, operators, and functions. String, numeric, date, and array field operands can be used in an expression and embedded functions. SQR supports a standardized set of mathematical operators and logical comparison operators working within a carefully defined set of precedence rules. SQR also provides a rich set of mathematical, string, date, and file manipulation functions along with a number of special purpose utility functions. All combined, the SQR expression provides a powerful tool that can be tailored to suit any information processing need. The following detail outlines the specific behavior of each expression component: (1) the operand, (2) the operator, and (3) the function.

Parameters

<i>Parameter</i>	<i>Description</i>
<code>dst_var</code>	A string, numeric, or date variable or array field to which the result of the expression is assigned.
<code>expression</code>	The expression to evaluate.

Operands

Operands form the backbone of an SQR expression. Operands do not have to be the same type. You can combine string, numeric, and array field operands to form a valid expression. SQR performs a sequence of automatic operand conversions when it evaluates expressions that contain dissimilar operand types. As the expression is evaluated, operands of lower precision are converted to match the operand of higher precision. Consider the following code example:

```
let  #answer = #float * #decimal / #integer
```

Because the *multiply* and *divide* operators are equal in precedence, the expression is evaluated as $(\#float * \#decimal) / \#integer$. Working from the inside out, the *#float* variable is converted to a decimal type where a multiply is performed yielding the simplified expression $(\#decimal) / \#integer$. SQR now converts the *#integer* operand to a decimal type before performing the final divide. When finished with the expression evaluation, SQR converts the result to match the type of the *#answer* variable.

Converting operands of lower precision to operands of higher precision preserves the number of significant digits. The number of significant digits is not lost when an integer is converted to float or decimal. In a similar manner, the number of significant digits is preserved when floating point operands are converted to the decimal type. The number of significant digits is sacrificed only when the final result is converted to match the type of the *#answer* variable and this variable is less precise than the highest of the operands being evaluated. In the example, precision is not lost if the *#answer* is declared as a decimal type. SQR considers integer variables as the lowest in the precision hierarchy, followed by float and then decimal.

Here are a few simple expression examples:

```
let #discount = round (&price * #rate / 100, 2)
let $name = $first_name || ' ' || $last_name
let customer.total (#customer_id) =

    customer.total (#customer_id) + #invoice_total
if not range(upper($code), 'A', 'G')
    ...processing when out of range...
let store.total (#store_id, #qtr) =

    store.total (#store_id, #qtr) + #invoice_total
let $date1 = strtodate ('Apr 10 2004', 'MON DD YYYY')
```

The following sections list operators and functions supported in expressions.

Operators

Operators of the same precedence are processed in the sequence in which they appear in the expression, from left to right. Use parentheses to override the normal precedence rules. All numeric types (decimal, float, integer) are supported for all operators.

This table lists operators in descending order of precedence (operators listed in the same row within the table have the same precedence):

Operator	Explanation
	Concatenate two strings or dates
+, -	Sign prefix (positive or negative)
^	Exponent
*, /, %	Multiply, divide, remainder: $a \% b = \text{mod}(a,b)$ for integers

Operator	Explanation
+, -	Plus, minus Note. SQR distinguishes between a sign prefix and arithmetic operation by the context of the expression.
>, <, >=, <=, <>, !=, =	Comparison operators: greater than, less than, greater or equal to, less than or equal to, not equal to (!= or <>), equal to.
not	Logical NOT
and	Logical AND
or, xor	Logical OR, XOR (exclusive OR)

Functions

This section lists numeric, file-related, and miscellaneous functions. The functions are listed in alphabetical order.

Function arguments are enclosed in parentheses and can be nested. Arguments referenced as x, y, or z indicate the first, second, or third argument of a function. Otherwise, functions take a single argument or no arguments. All arguments are evaluated before a function is evaluated.

Not all functions support all numeric types (decimal, float, integer). Certain functions do not support the decimal type directly, but convert input decimal operands to the float type before the function is evaluated. The following table annotates the functions that directly support the decimal type and the ones that do not.

Use parentheses to override the normal precedence rules.

This table describes numeric functions:

Function	Description
abs	<p>Returns the absolute value of <i>num_value</i>. This function returns a value of the same type as <i>num_value</i>.</p> <p>Syntax:</p> <pre>dst_var = abs(num_value)</pre> <p><i>num_value</i> = decimal, float, or integer literal, column, variable, or expression.</p> <p><i>dst_var</i> = decimal, float, or integer variable.</p> <p>Example:</p> <pre>let #dabsvar = abs(#dvar)</pre>

Function	Description
acos	<p>Returns the arccosine of <i>num_value</i> in the range of 0 to π radians. The value of <i>num_value</i> must be between -1 and 1. This function returns a float value.</p> <p>Syntax:</p> <pre>dst_var = acos(num_value)</pre> <p><i>num_value</i> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float.</p> <p><i>dst_var</i> = decimal, float, or integer variable.</p> <p>Example:</p> <pre>let #facosvar = acos(#fvar)</pre>
asin	<p>Returns the arcsine of <i>num_value</i> in the range of $-\pi/2$ to $\pi/2$ radians. The value of <i>num_value</i> must be between -1 and 1. This function returns a float value.</p> <p>Syntax:</p> <pre>dst_var = asin(num_value)</pre> <p><i>num_value</i> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float.</p> <p><i>dst_var</i> = decimal, float, or integer variable.</p> <p>Example:</p> <pre>let #fasinvar = asin(#fvar)</pre>
atan	<p>Returns the arctangent of <i>num_value</i> in the range of $-\pi/2$ to $\pi/2$ radians. The value of <i>num_value</i> must be between -1 and 1. This function returns a float value.</p> <p>Syntax:</p> <pre>dst_var = atan(num_value)</pre> <p><i>num_value</i> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float.</p> <p><i>dst_var</i> = decimal, float, or integer variable.</p> <p>Example:</p> <pre>let #fatanvar = atan(#fvar)</pre>

<i>Function</i>	<i>Description</i>
ceil	<p>Returns a value representing the smallest integer that is greater than or equal to <i>num_value</i>. This function returns a value of the same type as <i>num_value</i>.</p> <p>Syntax:</p> <pre>dst_var = ceil(num_value)</pre> <p><i>num_value</i> = decimal, float, or integer literal, column, variable, or expression.</p> <p><i>dst_var</i> = decimal, float, or integer variable.</p> <p>Example:</p> <pre>let #fceilvar = ceil(#fvar)</pre>
cos	<p>Returns the cosine of <i>num_value</i>. This function returns a float value.</p> <p>Syntax:</p> <pre>dst_var = cos(num_value)</pre> <p><i>num_value</i> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float.</p> <p><i>dst_var</i> = decimal, float, or integer variable.</p> <p>Example:</p> <pre>let #fcosvar = cos(#fvar)</pre>
cosh	<p>Returns the hyperbolic cosine of <i>num_value</i>. This function returns a float value.</p> <p>Syntax:</p> <pre>dst_var = cosh(num_value)</pre> <p><i>num_value</i> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float.</p> <p><i>dst_var</i> = decimal, float, or integer variable.</p> <p>Example:</p> <pre>let #fcoshvar = cosh(#fvar)</pre>

<i>Function</i>	<i>Description</i>
deg	<p>Returns a value expressed in degrees of <i>num_value</i>, which is expressed in radians. This function returns a float value.</p> <p>Syntax:</p> <pre>dst_var = deg(num_value)</pre> <p><i>num_value</i> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float.</p> <p><i>dst_var</i> = decimal, float, or integer variable.</p> <p>Example:</p> <pre>let #fdegvar = deg(#fvar)</pre>
e10	<p>Returns the value of 10 raised to <i>num_value</i>. This function returns a float value.</p> <p>Syntax:</p> <pre>dst_var = e10(num_value)</pre> <p><i>num_value</i> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float.</p> <p><i>dst_var</i> = decimal, float, or integer variable.</p> <p>Example:</p> <pre>let #fe10var = e10(#fvar)</pre>
exp	<p>Returns the value of e raised to <i>num_value</i>. This function returns a float value.</p> <p>Syntax:</p> <pre>dst_var = exp(num_value)</pre> <p><i>num_value</i> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float.</p> <p><i>dst_var</i> = decimal, float, or integer variable.</p> <p>Example:</p> <pre>let #fexpvar = exp(#fvar)</pre>

Function	Description
floor	<p>Returns a value representing the largest integer that is less than or equal to <i>num_value</i>. This function returns a value of the same type as <i>num_value</i>.</p> <p>Syntax:</p> <pre>dst_var = floor(num_value)</pre> <p><i>num_value</i> = decimal, float, or integer literal, column, variable, or expression.</p> <p><i>dst_var</i> = decimal, float, or integer variable.</p> <p>Example:</p> <pre>let #ffloorvar = floor(#fvar)</pre>
log	<p>Returns the natural logarithm of <i>num_value</i>. This function returns a float value.</p> <p>Syntax:</p> <pre>dst_var = log(num_value)</pre> <p><i>num_value</i> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float.</p> <p><i>dst_var</i> = decimal, float, or integer variable.</p> <p>Example:</p> <pre>let #flogvar = log(#fvar)</pre>
log10	<p>Returns the base-10 logarithm of <i>num_value</i>. This function returns a float value.</p> <p>Syntax:</p> <pre>dst_var = log10(num_value)</pre> <p><i>num_value</i> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float.</p> <p><i>dst_var</i> = decimal, float, or integer variable.</p> <p>Example:</p> <pre>let #flog10var = log10(#fvar)</pre>

Function	Description
mod	<p>Returns the fractional remainder, <i>f</i>, of <i>x_value</i>/<i>y_value</i> such that $x_value = i * y_value + f$, where <i>i</i> is an integer, <i>f</i> has the same sign as <i>x_value</i>, and the absolute value of <i>f</i> is less than the absolute value of <i>y_value</i>. The arguments are promoted to the type of the greatest precision and the function returns a value of that type.</p> <p>Syntax:</p> <pre>dst_var = mod(x_value, y_value)</pre> <p><i>x_value</i> = decimal, float, or integer literal, column, variable, or expression.</p> <p><i>y_value</i> = decimal, float, or integer literal, column, variable, or expression.</p> <p><i>dst_var</i> = decimal, float, or integer variable.</p> <p>Example:</p> <pre>let #fmodvar = mod(#fxvar, #fyvar)</pre>
power	<p>Returns the value of <i>x_value</i> raised to the power of <i>y_value</i>. This function returns a float value.</p> <p>Syntax:</p> <pre>dst_var = power(x_value, y_value)</pre> <p><i>x_value</i> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float.</p> <p><i>y_value</i> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float.</p> <p><i>dst_var</i> = decimal, float, or integer variable.</p> <p>Example:</p> <pre>let #fpowervar = power(#fxvar, #fyvar)</pre>

Function	Description
rad	<p>Returns a value expressed in radians of <i>num_value</i>, which is expressed in degrees. This function returns a float value.</p> <p>Syntax:</p> <pre>dst_var = rad(num_value)</pre> <p><i>num_value</i> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float.</p> <p><i>place_value</i> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float.</p> <p><i>dst_var</i> = decimal, float, or integer variable.</p> <p>Example:</p> <pre>let #fradvar = rad(#fvar)</pre>
round	<p>Returns a value that is <i>num_value</i> rounded to <i>place_value</i> digits after the decimal separator. This function returns a value of the same type as <i>num_value</i>.</p> <p>Syntax:</p> <pre>dst_var = round(num_value, place_value)</pre> <p><i>num_value</i> = decimal, float, or integer literal, column, variable, or expression.</p> <p><i>place_value</i> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float.</p> <p><i>dst_var</i> = decimal, float, or integer variable.</p> <p>Example:</p> <pre>let #frndvar = round(#fvar, #fplace) => (#x, #y)</pre>
sign	<p>Returns a -1, 0, or +1 depending on the sign of <i>num_value</i>. This function returns a float value.</p> <p>Syntax:</p> <pre>dst_var = sign(num_value)</pre> <p><i>num_value</i> = decimal, float, or integer literal, column, variable, or expression.</p> <p><i>dst_var</i> = decimal, float, or integer variable.</p> <p>Example:</p> <pre>let #fsignvar = sign(#fvar)</pre>

Function	Description
<code>sin</code>	<p>Returns the sine of <i>num_value</i>. This function returns a float value.</p> <p>Syntax:</p> <pre><i>dst_var</i> = sin(<i>num_value</i>)</pre> <p><i>num_value</i> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float.</p> <p><i>dst_var</i> = decimal, float, or integer variable.</p> <p>Example:</p> <pre>let #fsinvar = sin(#fvar)</pre>
<code>sinh</code>	<p>Returns the hyperbolic sine of <i>num_value</i>. This function returns a float value.</p> <p>Syntax:</p> <pre><i>dst_var</i> = sinh(<i>num_value</i>)</pre> <p><i>num_value</i> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float.</p> <p><i>dst_var</i> = decimal, float, or integer variable.</p> <p>Example:</p> <pre>let #fsinhvar = sinh(#fvar)</pre>
<code>sqrt</code>	<p>Returns the square root of <i>num_value</i>. This function returns a float value.</p> <p>Syntax:</p> <pre><i>dst_var</i> = sqrt(<i>num_value</i>)</pre> <p><i>num_value</i> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float.</p> <p><i>dst_var</i> = decimal, float, or integer variable.</p> <p>Example:</p> <pre>let #fsqrtvar = sqrt(#fvar)</pre>

<i>Function</i>	<i>Description</i>
tan	<p>Returns the tangent of <i>num_value</i>. This function returns a float value.</p> <p>Syntax:</p> <pre>dst_var = tan(num_value)</pre> <p><i>num_value</i> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float.</p> <p><i>dst_var</i> = decimal, float, or integer variable.</p> <p>Example:</p> <pre>let #ftanvar = tan(#fvar)</pre>
tanh	<p>Returns the hyperbolic tangent of <i>num_value</i>. This function returns a float value.</p> <p>Syntax:</p> <pre>dst_var = tanh(num_value)</pre> <p><i>num_value</i> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float.</p> <p><i>dst_var</i> = decimal, float, or integer variable.</p> <p>Example:</p> <pre>let #ftanhvar = tanh(#fvar)</pre>
trunc	<p>Returns a value that is <i>num_value</i> truncated to <i>place_value</i> digits after the decimal separator. This function returns a value of the same type as <i>num_value</i>.</p> <p>Syntax:</p> <pre>dst_var = trunc(num_value, place_value)</pre> <p><i>num_value</i> = decimal, float, or integer literal, column, variable, or expression.</p> <p><i>place_value</i> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float.</p> <p><i>dst_var</i> = decimal, float, or integer variable.</p> <p>Example:</p> <pre>let #ftruncvar = trunc(#fvar, #fplace)</pre>

The transcendental functions sin, cos, tan, sinh, cosh, and tanh take their arguments in radians. The functions asin, acos, and atan return radian values. To convert from radians to degrees or degrees to radians, use the rad or deg functions as follows:

```
let #x = sin(rad(45))           ! Sine of 45 degrees.
let #y = deg(asin(#x))         ! Convert back to degrees.
```

If arguments or intermediate results passed to a numeric function are invalid for that function, SQR halts with an error message.

For example, passing a negative number to the `sqrt` function causes an error. Use the `cond` function described in the Miscellaneous Functions table to prevent division by zero or other invalid function or operator argument values.

The following table lists file-related functions. These functions return 0 (zero) when successful; otherwise, they return the system error code.

Function	Description
delete	<p>Deletes the file <i>filename</i>. The function returns either a 0 (zero) to indicate success or the value returned from the operating system to indicate an error.</p> <p>Syntax:</p> <pre>stat_var = delete(filename)</pre> <p><i>filename</i> = text literal, column, variable, or expression.</p> <p><i>stat_var</i> = decimal, float, or integer variable.</p> <p>Example:</p> <pre>let #fstatus = delete(\$filename)</pre>
exists	<p>Determines whether the file <i>filename</i> exists. The function returns either a 0 (zero) to indicate success or the value returned from the operating system to indicate an error.</p> <p>Syntax:</p> <pre>stat_var = exists(filename)</pre> <p><i>filename</i> = text literal, column, variable, or expression.</p> <p><i>stat_var</i> = decimal, float, or integer variable.</p> <p>Example:</p> <pre>let #fstatus = exists(\$filename)</pre>
rename	<p>Renames <i>old_filename</i> to <i>new_filename</i>. The function returns either a 0 (zero) to indicate success or the value returned from the operating system to indicate an error.</p> <p>Syntax:</p> <pre>stat_var = rename(old_filename, new_filename)</pre> <p><i>old_filename</i> = text literal, column, variable, or expression.</p> <p><i>new_filename</i> = text literal, column, variable, or expression.</p> <p><i>stat_var</i> = decimal, float, or integer variable.</p> <p>Example:</p> <pre>let #fstatus = rename(\$old_filename, \$new_→ filename)</pre>

The following table lists miscellaneous functions. These functions return a string value unless otherwise indicated.

In these functions where a string argument is expected and a date variable, column, or expression is entered, SQR converts the date to a string according to the following rules:

- For DATETIME columns and SQR DATE variables, SQR uses the format specified by the SQR_DB_DATE_FORMAT setting.

If this has not been set, SQR uses the first database-dependent format as listed in the Default Database Formats table.

- For DATE columns, SQR uses the format specified by the SQR_DB_DATE_ONLY_FORMAT setting.

If this has not been set, SQR uses the format listed in the DATE Column Formats table.

- For TIME columns, SQR uses the format specified by the SQR_DB_TIME_ONLY_FORMAT setting.

If this has not been set, SQR uses the format as listed in the TIME Column Formats table.

Except where noted in an individual function, if a string variable, column, or expression is entered where a date argument is expected, the string must be in the format specified by the SQR_DB_DATE_FORMAT setting, one of the database-dependent formats listed in the Default Database Formats table, or the database-independent format 'YYYYMMDD[HH24[MI[SS[NNNNNN]]]]'.

<i>Function</i>	<i>Explanation</i>
array	<p>Returns a pointer to the starting address of the specified array field. The value returned from this function can be used only by a user-defined function. See the routine printarray in the file UFUNC.C for complete instructions on how to use this function.</p> <p>Syntax:</p> <pre>array_var = array(array_name, field_name) array_name = text literal, column, variable, or expression field_name = text literal, column, variable, or expression array_var = text variable</pre> <p>Example:</p> <pre>let #fstatus = printarray(array('products', => 'name'), 10, 2, 'c')</pre>
ascii	<p>Returns the numeric value for the first character in <i>str_value</i>. This function returns a float value.</p> <p>Syntax:</p> <pre>ascii_var = ascii(str_value) str_value = date or text literal, column, variable, or expression ascii_var = decimal, float, or integer variable</pre> <p>Example:</p> <pre>let #fascii = ascii(\$filename)</pre>

Function	Explanation
asciiic	<p>Returns the numeric value for the first character (rather than byte) of the specified string.</p> <p>Syntax:</p> <pre>ascii_var = asciiic(str_value)</pre> <p><i>str_value</i> = date or text literal, column, variable, or expression</p> <p><i>ascii_var</i> = decimal, float, or integer variable</p> <p>Example:</p> <pre>let #fascii = asciiic(\$filename)</pre>
chr	<p>Returns a string that is composed of a character with the numeric value of <i>num_value</i>.</p> <p>Syntax:</p> <pre>dst_var = chr(num_value)</pre> <p><i>num_value</i> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float.</p> <p><i>dst_var</i> = text variable.</p> <p>Example:</p> <pre>let \$svar = chr(#num)</pre>
cond	<p>Returns <i>y_value</i> if the <i>x_value</i> is nonzero; otherwise, returns <i>z_value</i>. If <i>y_value</i> is numeric, the <i>z_value</i> must also be numeric; otherwise, date and textual arguments are compatible. If either the <i>y_value</i> or <i>z_value</i> is a date variable, column, or expression, a date is returned. The return value of the function depends on which value is returned.</p> <p>Syntax:</p> <pre>dst_var = cond(x_value, y_value, z_value)</pre> <p><i>x_value</i> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float.</p> <p><i>y_value</i> = Any literal, column, variable, or expression</p> <p><i>z_value</i> = Any literal, column, variable, or expression</p> <p><i>dst_var</i> = Any variable</p> <p>Example:</p> <pre>let #avg = #total / cond(&rate != 0, &rate, 1)</pre>

Function	Explanation
dateadd	<p>Returns a date after adding (or subtracting) the specified units to the <i>date_value</i>.</p> <p>Syntax:</p> <pre>dst_var = dateadd(date_value, units_value, ⇒ quantity_value)</pre> <p><i>date_value</i> = date variable or expression.</p> <p><i>units_value</i> = text literal, column, variable, or expression. Valid units are 'year', 'quarter', 'week', 'month', 'day', 'hour', 'minute', and 'second'.</p> <p><i>quantity_value</i> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to float.</p> <p><i>dst_var</i> = date variable</p> <p>Example:</p> <pre>let \$date = dateadd(\$startdate, 'day', 7.5)</pre>
datediff	<p>Returns the difference between the specified dates expressed in <i>units_value</i>. The function returns a float value. The result can be negative if the first date is earlier than the second date.</p> <p>Syntax:</p> <pre>dst_var = datediff(date1_value, date2_value, ⇒ units_value)</pre> <p><i>date1_value</i> = date variable or expression.</p> <p><i>date2_value</i> = date variable or expression.</p> <p><i>units_value</i> = text literal, column, variable, or expression. Valid units are 'year', 'quarter', 'week', 'month', 'day', 'hour', 'minute', and 'second'</p> <p><i>dst_var</i> = decimal, float, or integer variable.</p> <p>Example:</p> <pre>let #diff = datediff(\$date1, \$date2, 'hour')</pre>
datenow	<p>Returns the current local date and time from the client machine.</p> <p>Syntax:</p> <pre>dst_var = datenow()</pre> <p><i>dst_var</i> = date variable</p> <p>Example:</p> <pre>let \$date = datenow()</pre>

Function	Explanation
datetostr	<p>Converts the date <i>date_value</i> to a string in the format <i>format_mask</i>.</p> <p>Syntax:</p> <pre>dst_var = datetostr(date_value [, format_mask])</pre> <p><i>date_value</i> = date variable or expression.</p> <p><i>format_mask</i> = text literal, column, variable, or expression. The keyword DATE can be used to specify the DATE-EDIT-MASK setting from the current locale. If this argument is not specified, the format specified by the SQR_DB_DATE_FORMAT setting is used. If this has not been set, the first database-dependent format listed in the Default Database Formats table is used.</p> <p><i>dst_var</i> = text variable</p> <p>Example: let \$formdate = datetostr(\$date, 'Day Mon DD, YYYY') let \$localedate = datetostr(\$date, DATE)</p>
edit	<p>Formats <i>source_value</i> according to <i>edit_mask</i> and returns a string containing the result.</p> <p>Syntax:</p> <pre>dst_var = edit(source_value, edit_mask)</pre> <p><i>source_value</i> = Any literal, column, variable, or expression</p> <p><i>edit_mask</i> = text literal, column, variable, or expression</p> <p><i>dst_var</i> = text variable</p> <p>Example:)</p> <pre>let \$phone = edit(&phone, '(xxx) xxx-xxxxx')=> let \$price = edit(#price, '999.99')=> let \$today = edit(\$date, 'DD/MM/YYYY')</pre>
getenv	<p>Returns the value of the specified environment variable. If the environment variable does not exist, an empty string is returned.</p> <p>Syntax:</p> <pre>dst_var = getenv(env_value)</pre> <p><i>env_value</i> = text literal, column, variable, or expression</p> <p><i>dst_var</i> = text variable</p> <p>Example:</p> <pre>let \$myuser = getenv('USER')</pre>

Function	Explanation
instr	<p>Returns the numeric position of <i>sub_value</i> in <i>source_value</i> or 0 (zero) if not found. The search begins at offset <i>offset_value</i>. This function returns a float value.</p> <p>Syntax:</p> <pre>dst_var = instr(source_value, sub_value, offset_value)</pre> <p><i>source_value</i> = date or text literal, column, variable, or expression.</p> <p><i>sub_value</i> = text literal, column, variable, or expression.</p> <p><i>offset_value</i> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to integer.</p> <p><i>dst_var</i> = decimal, float, or integer variable.</p> <p>Example:</p> <pre>let #offset = instr(&description, 'auto', 10)</pre>
instrb	<p>Performs the same functionality as the instr function except that the starting point and returned value are expressed in bytes rather than in characters.</p> <p>Syntax:</p> <pre>dst_var = instrb(source_value, sub_value, offset_value)</pre> <p><i>source_value</i> = date or text literal, column, variable, or expression.</p> <p><i>sub_value</i> = text literal, column, variable, or expression.</p> <p><i>offset_value</i> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to integer.</p> <p><i>dst_var</i> = decimal, float, or integer variable.</p> <p>Example:</p> <pre>let #offset = instrb(&description, 'auto', 10)</pre>
isblank	<p>Returns a value of 1 (one) if <i>source_val</i> is an empty string, null string, or composed entirely of white-space characters; otherwise, returns a value of 0 (zero).</p> <p>Syntax:</p> <pre>dst_var = isblank(source_value)</pre> <p><i>source_value</i> = date or text literal, column, variable, or expression</p> <p><i>dst_var</i> = decimal, float, or integer variable</p> <p>Example:</p> <pre>let #blank = isblank(&description)</pre>

<i>Function</i>	<i>Explanation</i>
isnull	<p>Returns a value of 1 (one) if source_val is null; otherwise, returns a value of 0 (zero).</p> <p>Syntax:</p> <pre>dst_var = isnull(source_value)</pre> <p>source_value = date or text literal, column, variable, or expression</p> <p>dst_var = decimal, float, or integer variable</p> <p>Example:</p> <pre>let #null = isnull(\$date)</pre>
length	<p>Returns the number of characters in source_value.</p> <p>Syntax:</p> <pre>dst_var = length(source_value)</pre> <p>source_value = date or text literal, column, variable, or expression</p> <p>dst_var = decimal, float, or integer variable</p> <p>Example:</p> <pre>let #length = length(&description)</pre> <p>Note. Oracle recommends that you use either the lengthp or lengtht function instead of the length function.</p>
lengthb	<p>(Multibyte versions of SQR only) Has the same functionality as the length function except that the return value is expressed in bytes rather than in characters.</p> <p>Syntax:</p> <pre>dst_var = lengthb(source_value)</pre> <p>source_value = date or text literal, column, variable, or expression</p> <p>dst_var = decimal, float, or integer variable</p> <p>Example:</p> <pre>let #length = lengthb(&description)</pre> <p>Note. Oracle recommends that you use either the lengthp or lengtht function instead of the lengthb function.</p>
lengthp	<p>Returns the length of a given string in print positions.</p> <p>Syntax:</p> <pre>dst_var = lengthp(source_value)</pre> <p>source_value = date or text literal, column, variable, or expression</p> <p>dst_var = decimal, float, or integer variable</p> <p>Example:</p> <pre>let #printlen = lengthp(&string)</pre>

<i>Function</i>	<i>Explanation</i>
lengtht	<p>Returns the length of a given string in bytes when converted (transformed) to a specified encoding.</p> <p>Syntax:</p> <pre>dst_var = lengtht(source_value, encoding_value)</pre> <p><i>source_value</i> = date or text literal, column, variable, or expression</p> <p><i>encoding_value</i> = text literal, column, variable, or expression</p> <p><i>dst_var</i> = decimal, float, or integer variable</p> <p>Example:</p> <pre>let #sjislen = lengtht(\$string, 'Shift-JIS')</pre>
lower	<p>Converts the contents of <i>source_value</i> to lowercase and returns the result.</p> <p>Syntax:</p> <pre>dst_var = lower(source_value)</pre> <p><i>source_value</i> = date or text literal, column, variable, or expression</p> <p><i>dst_var</i> = text variable</p> <p>Example:</p> <pre>let \$lower = lower(&description)</pre>
lpad	<p>Pads the <i>source_value</i> on the left to a length of <i>length_value</i> using <i>pad_value</i> and returns the result.</p> <p>Syntax:</p> <pre>dst_var = lpad(source_value, length_value, pad_value)</pre> <p><i>source_value</i> = date or text literal, column, variable, or expression.</p> <p><i>length_value</i> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to integer.</p> <p><i>pad_value</i> = text literal, column, variable, or expression.</p> <p><i>dst_var</i> = text variable.</p> <p>Example:</p> <pre>let \$lpad = lpad(\$notice, 25, '.')</pre>

Function	Explanation
ltrim	<p>Trims characters in <i>source_value</i> from the left until a character is not in <i>set_value</i> and returns the result.</p> <p>Syntax:</p> <pre>dst_var = ltrim(source_value, set_value)</pre> <p><i>source_value</i> = date or text literal, column, variable, or expression</p> <p><i>set_value</i> = text literal, column, variable, or expression</p> <p><i>dst_var</i> = text variable</p> <p>Example:</p> <pre>let \$ltrim = ltrim(&description, '.')</pre>
nvl	<p>Returns <i>y_value</i> if the <i>x_value</i> is null; otherwise, returns <i>x_value</i>. If <i>x_value</i> is numeric, <i>y_value</i> must also be numeric; otherwise, date and textual arguments are compatible. In any case, the <i>x_value</i> determines the type of expression returned. The return value of the function depends on which value is returned.</p> <p>Syntax:</p> <pre>dst_var = nvl(x_value, y_value)</pre> <p><i>x_value</i> = Any literal, column, variable, or expression</p> <p><i>y_value</i> = Any literal, column, variable, or expression</p> <p><i>dst_var</i> = Any variable</p> <p>Example:</p> <pre>let \$city = nvl(&city, '-- not city --')</pre> <p>If <i>x_value</i> is a date and <i>y_value</i> is textual, <i>y_value</i> is validated according to the following rules:</p> <p>For DATETIME columns and SQR DATE variables, SQR uses the format specified by the SQR_DB_DATE_FORMAT setting, one of the database-dependent formats (see the Default Database Formats table), or the database-independent format 'SYYYMDD[HH24[MI[SS[NNNNN]]]]'.</p> <p>For DATE columns, SQR uses the format specified by the SQR_DB_DATE_ONLY_FORMAT setting, or the format listed in the DATE Column Formats table.</p> <p>For TIME columns, SQR uses the format specified by the SQR_DB_TIME_ONLY_FORMAT setting, or the format as listed in the TIME Column Formats table.</p>

Function	Explanation
range	<p>Returns a value of 1 (one) if <i>x_value</i> is between <i>y_value</i> and <i>z_value</i>; otherwise, returns a value of 0 (zero). If the first argument is text or numeric, the other arguments must be of the same type. If the first argument is a date, the remaining arguments can be dates, text, or both. You can also perform a date comparison on a mix of date and text arguments, for example, where <i>x_value</i> is a date and <i>y_value</i> and <i>z_value</i> are text arguments. In a comparison of this sort, <i>y_value</i> must represent a date that is earlier than that of <i>z_value</i>.</p> <p>Syntax:</p> <pre>dst_var = range(x_value, y_value, z_value)</pre> <p><i>x_value</i> = Any literal, column, variable, or expression <i>y_value</i> = Any literal, column, variable, or expression <i>z_value</i> = Any literal, column, variable, or expression <i>dst_var</i> = decimal, float, or integer variable</p> <p>Example:</p> <pre>let #inrange = range(&grade, 'A', 'D')⇒ let #inrange = range(\$date, \$startdate,⇒ \$enddate)⇒ let #inrange = range(\$date, \$startdate, '15-Apr-⇒ 04')⇒ let #inrange = range(#price, #low, #high)</pre> <p>If <i>x_value</i> is a date and <i>y_value</i> and/or <i>z_value</i> is textual, then <i>y_value</i> and/or <i>z_value</i> is validated according to the following rules:</p> <p>For DATETIME columns and SQR DATE variables, SQR uses the format specified by the SQR_DB_DATE_FORMAT setting, one of the database-dependent formats (see the Default Database Formats table), or the database-independent format 'SYYYMM-MDD[HH24[MI[SS[NNNNNN]]]]'.</p> <p>For DATE columns, SQR uses the format specified by the SQR_DB_DATE_ONLY_FORMAT setting, or the format listed in the table DATE Column Formats.</p> <p>For TIME columns, SQR uses the format specified by the SQR_DB_TIME_ONLY_FORMAT setting, or the format as listed in table TIME Column Formats.</p>

Function	Explanation
replace	<p>Inspects the contents of <i>source_value</i> and replaces all occurrences of <i>from_string</i> with <i>to_string</i> and returns the modified string.</p> <p>Syntax:</p> <pre>dst_var = replace(source_value, from_string, to_string)</pre> <p><i>source_value</i> = date or text literal, column, variable, or expression <i>from_string</i> = text literal, column, variable, or expression <i>to_string</i> = text literal, column, variable, or expression <i>dst_var</i> = text variable</p> <p>Example:</p> <pre>let \$replaced = replace(\$paragraph, 'good', 'excellent')</pre>
roman	<p>Returns a string that is the character representation of <i>source_value</i> expressed in lowercase roman numerals.</p> <p>Syntax:</p> <pre>dst_var = roman(source_value)</pre> <p><i>source_value</i> = text literal, column, variable, or expression <i>dst_var</i> = text variable</p> <p>Example:</p> <pre>let \$roman = roman(#page-count)</pre>
rpadd	<p>Pads the <i>source_value</i> on the right to a length of <i>length_value</i> using <i>pad_value</i> and returns the result.</p> <p>Syntax:</p> <pre>dst_var = rpadd(source_value, length_value, pad_value)</pre> <p><i>source_value</i> = date or text literal, column, variable, or expression <i>length_value</i> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to integer. <i>pad_value</i> = text literal, column, variable, or expression <i>dst_var</i> = text variable</p> <p>Example:</p> <pre>let \$rpadd = rpadd(\$notice, 25, '.')</pre>

Function	Explanation
rtrim	<p>Trims characters in <i>source_value</i> from the right until a character is not in <i>set_value</i> and returns the result.</p> <p>Syntax:</p> <pre>dst_var = rtrim(source_value, set_value)</pre> <p><i>source_value</i> = date, or text literal, column, variable, or expression</p> <p><i>set_value</i> = text literal, column, variable, or expression</p> <p><i>dst_var</i> = text variable</p> <p>Example:</p> <pre>let \$rtrim = rtrim(&description, '.')</pre>
strtodate	<p>Converts the string <i>source_value</i> in the format <i>format_mask</i> to a date type.</p> <p>Syntax:</p> <pre>dst_var = strtodate(source_value [, format_mask])</pre> <p><i>source_value</i> = text literal, column, variable, or expression.</p> <p><i>format_mask</i> = text literal, column, variable, or expression that describes the exact format of the <i>source_value</i>. The keyword DATE can be used to specify the DATE-EDIT-MASK setting from the current locale. If this argument is not specified, then <i>source_value</i> must be in the format specified by the SQR_DB_DATE_FORMAT setting, one of the database-dependent formats (see the Default Database Formats table), or the database-independent format 'SYYYMMDD[HH24[MI[SS[NNNNNN]]]]'. Valid format codes are specified in the Date Edit Format Codes table.</p> <p><i>dst_var</i> = date variable</p> <p>Example: let \$date = strtodate(\$str_date, 'Mon DD, YYYY') let \$date = strtodate(\$str_date, DATE)</p>

Function	Explanation
substr	<p>Extracts the specified portion <i>source_value</i>. The extraction begins at <i>offset_value</i> (origin is 1) for a length of <i>length_value</i> characters.</p> <p>Syntax:</p> <pre>dst_var = substr(source_value, offset_value, => length_value)</pre> <p><i>source_value</i> = date or text literal, column, variable, or expression.</p> <p><i>offset_value</i> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to integer.</p> <p><i>length_value</i> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to integer.</p> <p><i>dst_var</i> = text variable.</p> <p>Example:</p> <pre>let \$piece = substr(&record, 10, #len)</pre> <p>Note. Oracle recommends that you use either the substrp or substrtr function instead of the substr function.</p>
substrb	<p>Has the same functionality as the substr function except that the starting point and length are expressed in bytes rather than in characters.</p> <p>Syntax:</p> <pre>dst_var = substrb(source_value, offset_value, => length_value)</pre> <p><i>source_value</i> = date or text literal, column, variable, or expression.</p> <p><i>offset_value</i> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to integer.</p> <p><i>length_value</i> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to integer.</p> <p><i>dst_var</i> = text variable.</p> <p>Example:</p> <pre>let \$piece = substrb(&record, 10, #len)</pre> <p>Note. Oracle recommends that you use either the substrp or substrtr function instead of the substrb function.</p>

<i>Function</i>	<i>Explanation</i>
substrp	<p>Returns a substring of a given string starting at a specified print position into the string and of a specified print length.</p> <p>Syntax:</p> <pre>dst_var = substrp(source_value, offset_value, => length_value)</pre> <p><i>source_value</i> = date or text literal, column, variable, or expression.</p> <p><i>offset_value</i> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to integer.</p> <p><i>length_value</i> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to integer.</p> <p><i>dst_var</i> = decimal, float, or integer variable.</p> <p>Example:</p> <pre>let \$sub = substrp(&string, #printpos, #printlen)</pre>
substrt	<p>Returns a substring of a given string starting at a specified byte and byte length in a given encoding.</p> <p>Syntax:</p> <pre>dst_var = substrt(source_value, offset_value, => length_value, encoding_value)</pre> <p><i>source_value</i> = date or text literal, column, variable, or expression.</p> <p><i>offset_value</i> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to integer.</p> <p><i>length_value</i> = decimal, float, or integer literal, column, variable, or expression. The value is always converted to integer.</p> <p><i>encoding_value</i> = text literal, column, variable, or expression.</p> <p><i>dst_var</i> = text variable.</p> <p>Example:</p> <pre>let \$sjisPrep = substrt(&string, 1, 10, 'Shift->JIS')</pre>
to_char	<p>Converts <i>source_value</i> to a string, using maximum precision.</p> <p>Syntax:</p> <pre>dst_var = to_char(source_value)</pre> <p><i>source_value</i> = decimal, float, or integer literal, column, variable, or expression</p> <p><i>dst_var</i> = text variable</p> <p>Example:</p> <pre>let \$string = to_char(#number)</pre>

<i>Function</i>	<i>Explanation</i>
<code>to_multi_byte</code>	<p>Converts the specified string as follows: Any occurrence of a double-byte character that also has a single-byte representation (numerals, punctuation, roman characters, and katakana) is converted.</p> <p>Syntax:</p> <pre>dst_var = to_multi_byte (source_value)</pre> <p><i>source_value</i> = date or text literal, column, variable, or expression</p> <p>Example:</p> <pre>let \$multi = to_multi_byte (&text)</pre>
<code>to_number</code>	<p>Converts <i>source_value</i> to a number. This function returns a float value.</p> <p>Syntax:</p> <pre>dst_var = to_number(source_value)</pre> <p><i>source_value</i> = date or text literal, column, variable, or expression</p> <p><i>dst_var</i> = decimal, float, or integer variable</p> <p>Example:</p> <pre>let #value = to_number(\$number)</pre>
<code>to_single_byte</code>	<p>Converts the specified string as follows: For SJIS, EBCDIK290, and EBCDIK1027, any occurrence of a single-byte character that also has a multi-byte representation (numerals, punctuation, roman characters, and katakana) is converted. This function also converts a sequence of kana characters followed by certain grammatical marks into a single-byte character that combines the two elements. For all other encodings, the string is not modified.</p> <p>Syntax:</p> <pre>dst_var = to_single_byte(source_value)</pre> <p><i>source_value</i> = date or text literal, column, variable, or expression</p> <p>Example:</p> <pre>let \$single = to_single_byte (&text)</pre>

<i>Function</i>	<i>Explanation</i>
translate	<p>Inspects the contents of <i>source_value</i> and converts characters that match those in <i>from_set</i> to the corresponding character in <i>to_set</i> and returns the translated string.</p> <p>Syntax:</p> <pre>dst_var = translate(source_value, from_set, to_set)</pre> <p><i>source_value</i> = date or text literal, column, variable, or expression <i>from_set</i> = text literal, column, variable, or expression <i>to_set</i> = text literal, column, variable, or expression <i>dst_var</i> = text variable</p> <p>Example:</p> <pre>let \$translated = translate(edit(&price,⇒ '999,999.99'),' ',' ','')</pre>
transform	<p>Returns a Unicode string that is a specified transform of a given string.</p> <p>Syntax:</p> <pre>dst_var = transform(source_value, transform_value)</pre> <p><i>source_value</i> = date or text literal, column, variable or expression <i>transform_value</i> = text literal, column, variable, or expression <i>dst_var</i> = text variable</p> <p>Example:</p> <pre>let \$hiragana = transform(\$string, 'ToHiragana')</pre>
unicode	<p>Returns a Unicode string from the string of hexadecimal values provided.</p> <p>Syntax:</p> <pre>dst_var = unicode(source_value)</pre> <p><i>source_value</i> = text literal, column, variable or expression <i>dst_var</i> = text variable</p> <p>Example:</p> <pre>let \$uniStr = unicode('u+5e73 u+2294')</pre>
upper	<p>Converts the contents of <i>source_value</i> to uppercase and returns the result.</p> <p>Syntax:</p> <pre>dst_var = upper(source_value)</pre> <p><i>source_value</i> = date or text literal, column, variable, or expression <i>dst_var</i> = text variable</p> <p>Example:</p> <pre>let \$upper = upper(&description)</pre>

<i>Function</i>	<i>Explanation</i>
wrapdepth	<p>Returns the number of print lines required by <i>source_value</i>. See the PRINTWRAP command for detailed descriptions of the parameters to this function. This function returns a float value.</p> <p>Syntax:</p> <pre>dst_var = wrapdepth(source_value, wrap_width, ⇒ line_height, on, strip)</pre> <p><i>source_value</i> = text literal, column, variable, or expression</p> <p><i>wrap_width</i> = decimal, float, or integer literal, column, variable, or expression</p> <p><i>line_height</i> = decimal, float, or integer literal, column, variable, or expression</p> <p><i>on</i> = text literal, column, variable, or expression</p> <p><i>strip</i> = text literal, column, variable, or expression</p> <p><i>dst_var</i> = decimal, float, or integer variable</p> <p>Example:</p> <pre>let #depth = wrapdepth(&description,40,1,'<13>','')</pre>

Writing Custom Functions

In addition to using the preceding built-in functions, you can write your own functions in C, using the supplied source file UFUNC.C.

You can pass any number of arguments to your function and values can be returned by the function or passed back in variables.

After editing and recompiling UFUNC.C, you must relink SQR.

The following step-by-step example shows how to add a user-defined function to SQR so that it can be invoked using the LET, IF, or WHILE command.

The example adds the C function random, which returns a random number. The function accepts a parameter that is used as the seed to start a new sequence of numbers. If the seed is zero, then the same sequence is used.

When adding functions to UFUNC, remember the following considerations:

- String functions require the following arguments:
 - (int) Number of arguments.
 - (char *) or (double *) Array of argument pointers, to either char[] or double.
 - (char *) Address for result string. If unchanged, function returns a NULL string.
 - (int) Maximum length of result string, in bytes.

- Numeric functions require the following arguments:
 - (int) Number of arguments.
 - (char *) or (double *) Array of argument pointers, to either char[] or double.
 - (double *) Address for result numeric value. If unchanged, function returns zero.

To add the random function to SQR, add the following modifications to the UFUNC.C file that was provided with SQR:

- Add the prototype for the random function: static void random CC_ARGS((char *, char *));
- Add the function name to the declaration list. The name of the function called from SQR is random. The return type is *n* for numeric. The number of arguments passed is 1, and the argument type is *n* for numeric. The function name in UFUNC.C is random. The characters PVR must be entered before the function name.

<i>Name</i>	<i>Return_type</i>	<i>Number of Arguments</i>	<i>Arg_Types</i>	<i>Function</i>
"max",	'n',	0,	"n",	PVR max,
"max",	'n',	0,	"n",	PVR max,
"split",	'n',	0,	"C",	PVR split,
"printarray",	'n',	4,	"cnnc",	PVR printarray,
"random",	'n',	1,	"n",	PVR random,

```
/* Last entry must be NULL -- do not change */
", '\0', 0, "", 0
};
```

At the end the of UFUNC.C file, add the following random routine. The routine name must be lowercase; however, in your SQR program it can be referenced in either uppercase or lowercase.

```
static void random CC_ARGL((argc, argv, result))
CC_ARG(int, argc)           /* The number arguments passed */
CC_ARG(double *, argv[])    /* The argument list */
CC_LARG(double *, result)   /* Where to store result */
{
  if (*argv[0] != 0)
    srand(*argv[0]);
  *result = rand();
  return;
}
```

After these modifications, recompile UFUNC.C and relink SQR. See the programmer's reference manual for details about your particular machine.

This is a simple SQR program that uses the random function:

```

begin-program
  do get-random-number
  do process-calculations
end-program

begin-procedure
  let #seed = 44
  let #ran = random(#seed)
end-procedure

begin-procedure process-calculations
.
.
.

```

Example

These examples show some complex expressions:

```

let #j = ((#a + #b) * #c) ^ 2
if #j > 2 and sqrt(#j) < 20 or #i + 2 > 17.4
while upper(substr(&descrip,1,#j+2)) != 'XXXX'

and not isnull(&price)
let #len = length(&fname || &initial || &lname) + 2
let $s = edit(&price * &rate, '99999.99')
let summary.total(#j) = summary.total(#j) + (&price * &rate)
if summary.total(#j) > 1000000
let store.total (#store_id, #dept)

    = store.total (#store_id, #dept) + #total
let #diff = datediff(datenow(), strtodate('1995','YYYY'),'day')
let $newdate = dateadd(datenow(),'month',50)
let $date1 = datetostr(strtodate(&sale_date), 'Day Month DD, YYYY')

```

SQR analyzes LET, IF, and WHILE expressions when it compiles your code and saves the result in an internal format so that repetitive execution is at maximum speed.

LOAD-LOOKUP

Syntax

In the SETUP section:

```

LOAD-LOOKUP
NAME=lookup_table_name
TABLE=database_table_name
KEY=key_column_name
RETURN_VALUE=return_column_name
[ROWS=initial_row_estimate_int_lit]
[EXTENT=size_to_grow_by_int_lit]
[WHERE=where_clause_txt_lit]
[SORT=sort_mode]
[QUIET]
[SCHEMA=schema_name]
[PROCEDURE=procedure_name]
[COMMAND=command_string]
[GETDATA=getdata_string]
[PARAMETERS=(...)]
[FROM-ROWSET=(m,n,-n,m-,all)]
[FROM-PARAMETER=rowset_name]

```

In the body of the report:

```

LOAD-LOOKUP
NAME=lookup_table_name
TABLE=database_table_name
KEY=key_column_name
RETURN_VALUE=return_column_name
[ROWS=initial_row_estimate_lit|_var|_col]
[EXTENT=size_to_grow_by_lit|_var|_col]
[WHERE=where_clause_txt_lit|_var|_col]
[SORT=sort_mode]
[QUIET]
[SCHEMA=schema_name]
[PROCEDURE=procedure_name]
[COMMAND=command_string]
[GETDATA=getdata_string]
[PARAMETERS=(...)]
[FROM-ROWSET=(m,n,-n,m-,all)]
[FROM-PARAMETER=rowset_name]

```

Description

Loads an internal table with columns from the database. Enables a quick search using the LOOKUP command.

Use the LOAD-LOOKUP command in conjunction with one or more LOOKUP commands.

LOAD-LOOKUP retrieves two columns from the database, the KEY field and the RETURN_VALUE field. Rows are ordered by KEY and stored in an array.

LOAD-LOOKUP commands specified in the SETUP section are always loaded and cannot reference variables for the ROWS, EXTENT, and WHERE arguments.

When you use the LOOKUP command, SQR searches the array (with a binary search) to find the RETURN_VALUE corresponding to the KEY referenced in the lookup.

Usually this type of lookup can be done with a database join, but joins take substantially longer. However, if your report is small and the number of rows to be joined is small, a lookup table can be slower because the entire table has to be loaded and sorted for each report run.

By default, SQR lets the database sort the data. This works fine if the database and SQR both use the same character set and collating sequence. The **Sort** argument enables you to specify the sorting method if this is not true. Additionally, if the machine that SQR is running on is faster than the machine that the database is running on, letting SQR perform the sort could decrease the execution time of the report.

The only limit to the size of a lookup table is the amount of memory your computer has available. You could conceivably load an array with many thousands of rows. The binary search is performed quickly regardless of how many rows are loaded.

Except for the amount of available memory, the number of lookup tables that can be defined is unlimited.

Parameters

<i>Parameter</i>	<i>Description</i>
NAME	The name of the lookup table. The array name is referenced in the LOOKUP command.
TABLE	The name of the table in the database, where the KEY and RETURN_VALUE columns or expressions are stored.
KEY	The name of the column that is used as the key in the array that is used for looking up the information. Keys can be character, date, or numeric data types. If the key is numeric, SQR permits only integers with 12 digits or fewer for the KEY column. Keys can be any database-supported expression. See the RETURN_VALUE argument.
RETURN_VALUE	The name of the column (expression) that is returned for each corresponding key. The following example is for ORACLE. See your database manual for the correct syntax. RETURN_VALUE='name "-" country "-" population'
ROWS	The initial size of the lookup table. This argument is optional, and if it is not specified, a value of 100 is used.
EXTENT	The amount to increase the array when it becomes full. This argument is optional, and if it is not specified, a value of 25% of the ROWS value is used.
WHERE	A WHERE clause used to select a subset of all the rows in the table. If it is specified, the selection begins after the word WHERE. The WHERE clause is limited to 255 characters.
Sort	The sorting method to be used. The following values are permitted: DC: Database sorts data, case-sensitive sort DI: Database sorts data, case-insensitive sort SC: SQR sorts data, case-sensitive sort SI: SQR sorts data, case-insensitive sort The default is SC or the method specified by the -LL command-line flag. The DI method is applicable only to databases that provide this feature and have been installed in that manner.

<i>Parameter</i>	<i>Description</i>
QUIET	Suppresses the message <i>Loading lookup array...</i> when the command executes. The warning message stating the number of duplicate keys found is also suppressed.

Example

The following command loads the array *states* with the columns *abbr* and *name* from the database table *stateabbrs*, where country is *USA*.

```
load-lookup
name=states
rows=50
table=stateabbrs
key=abbr
return_value=name
where=country='USA'
```

The preceding array is used in the example for the LOOKUP command to retrieve the full text of a state name from the abbreviation.

The following example uses the LOOKUP command to validate data entered by a user using an INPUT command:

```
get_state:
input $state 'Enter state abbreviation'
uppercase $state
lookup states $state $name
if $name = '' ! Lookup didn't find a match
  show 'No such state.'
  goto get_state
end-if
```

Surround any command argument with embedded spaces by single quotes, as shown in the following example:

```
where='country=''USA'' and region = ''NE'''
```

The entire WHERE clause is surrounded by quotes. The two single quotes around USA and NE are translated to one single quote in the SQL statement.

The following example uses joins in a LOAD-LOOKUP command by including two tables in TABLE and the join in WHERE:

```
load-lookup

name=states

rows=50

sort=sc

table='stateabbrs s, regions r'

key=abbr

return_value=name

where='s.abbr = r.abbr and r.location = ''ne'''
```

LOOKUP

Syntax

```
LOOKUP lookup_table_name{key_any_lit|_var|_col}
{ret_txt_var|_date_var}
```

Description

Searches a lookup table (an array) for a key value and returns the corresponding text string.

Speeds up processing for long reports. For example, if you want to print the entire state name rather than the abbreviation, you could use LOAD-LOOKUP and then LOOKUP to do this.

Parameters

Parameter	Description
lookup_table_name	Specifies the lookup table. This table must be previously loaded with a LOAD-LOOKUP command.
key_any_lit _var _col	The key used for the lookup.
ret_txt_var _date_var	A string variable into which the corresponding value is returned.

Example

The following example works in conjunction with the example for the LOAD-LOOKUP command:

```
lookup states &state_abbr $state_name
```

This example searches the states lookup table for a matching *&state_abbr* value; if the value is found, the example returns the corresponding state name in *\$state_name*. If it is not found, a null is placed in *\$state_name*.

See Also

The LOAD-LOOKUP command.

LOWERCASE

Syntax

```
LOWERCASE txt_var
```

Description

Converts the contents of a text variable to lowercase.

Parameters

<i>Parameter</i>	<i>Description</i>
<i>txt_var</i>	Specifies a text variable to be converted to lowercase.

Example

The following example illustrates the LOWERCASE command:

```
input $answer 'Type EXIT to stop'
lowercase $answer      ! Allows user to enter
                        ! upper or lowercase.
if $answer = 'exit'
  ...etc...
```

See Also

The lower function listed in the Miscellaneous Functions table.

MBTOSBS

Syntax

```
MBTOSBS {txt_var}
```

Description

Converts a double-byte string to its single-byte equivalent.

This command converts the specified string as follows: any occurrence of a double-byte character that also has a single-byte representation (numerals, punctuation, roman characters, and katakana) is converted.

Parameters

<i>Parameter</i>	<i>Description</i>
txt_var	Specifies the string to be converted.

See Also

The TO_SINGLE_BYTE function of the LET command.

MOVE

Syntax

```
MOVE {src_any_lit|_var|_col} TO dst_any_var  
[[:$]format_mask|NUMBER|MONEY|DATE]
```

Description

Moves the source field to the destination field. Optionally, you can reformat the field using the format_mask argument. Source and destination fields can be different types, numeric, text, or date. MOVE is also useful for converting from one type to another; however, date and numeric variables are incompatible.

When a date variable or column is moved to a string variable, the date is converted according to the following rules:

- For DATETIME columns and SQR DATE variables, SQR uses the format specified by the SQR_DB_DATE_FORMAT setting.
If this has not been set, SQR uses the first database-dependent format as listed in the Default Database Formats table.
- For DATE columns, SQR uses the format specified by the SQR_DB_DATE_ONLY_FORMAT setting.
If this has not been set, SQR uses the format listed in the DATE Column Formats table.
- For TIME columns, SQR uses the format specified by the SQR_DB_TIME_ONLY_FORMAT setting.
If this has not been set, SQR uses the format as listed in the TIME Column Formats table.

Finally, as this example shows, the edit mask can be contained in a string variable.

Parameters

<i>Parameter</i>	<i>Description</i>
src_any_lit _var _col	Specifies any source column, variable, or literal. Note that a date can be stored in a date variable or column, or a string literal, column, or variable. When you are using a date format_mask or the keyword DATE with the MOVE command, the source, if a string literal, column, or variable, must be in the format specified by the SQR_DB_DATE_FORMAT setting, one of the database-dependent formats as listed in the Default Database Formats table, or the database-independent format 'YYYYMMDD[HH24[MI[SS[NNNNNN]]]]'.
dst_any_var	Specifies a destination variable.
format_mask	Specifies an optional format mask. For additional information regarding edit masks, see the PRINT command.
NUMBER	Indicates that src_any_lit _var _col is to be formatted using the NUMBER-EDIT-MASK from the current locale. This option is not legal with date variables. (See the ALTER_LOCALE command.)
MONEY	Indicates that src_any_lit _var _col is to be formatted using the MONEY-EDIT-MASK from the current locale. This option is not legal with date variables. (See the ALTER_LOCALE command.)
DATE	Indicates that src_any_lit _var _col is to be formatted using the DATE-EDIT-MASK from the current locale. This option is not legal with numeric variables. (See the ALTER_LOCALE command.)

Example

The following example illustrates the various features of the MOVE command:

```
!
! Convert a string in place
!
move '123456789' to $ssn
move $ssn to $ssn xxx-xx-xxxx
show '$SSN = ' $ssn
```

Produces the following output:

```
$SSN = 123-45-6789
```

```
!
! Convert a number to a string using an edit mask
!
move 1234567.89 to #value
move #value to $value 999,999,999.99
show '$Value = ' $value
```

Produces the following output:

```
$Value = 1,234,567.89

!
! Convert a number to a string using a variable edit mask
!
move 123 to #counter
move '099999' to $mask
move #counter to $counter :$mask
show '$Counter = ' $counter
```

Produces the following output:

```
$Counter = 000123

!
! Convert a number to a string using the default edit mask
!
! SQR, by default, outputs six digits of precision.
! If you require more or less precision, specify an edit mask.
!
move 123.78 to #defvar
move #defvar to $defvar
show '$DefVar = ' $defvar
```

Produces the following output:

```
$DefVar = 123.780000

!
! Convert the number to a string using the locale default
! numeric edit mask
!
alter-locale number-edit-mask = '99,999,999.99'
move 123456.78 to #nvar
move #nvar to $nvar number
show '$NVar = ' $nvar
```

Produces the following output:

```
$NVar = 123,456.78

!
! Convert the money value to a string using the locale default
! money edit mask
!
alter-locale money-edit-mask = '$9,999,999.99'
move 123456.78 to #mvar
move #mvar to $mvar money
show '$MVar = ' $mvar
```

Produces the following output:

```

$MVar = $ 123,456.78

!
! Convert the date column to a string using the locale default
! date edit mask
!
begin-select
dcol
  from tables
end-select
alter-locale date-edit-mask = 'Mon-DD-YYYY'
move &dcol to $dvar date
show '$DVar = ' $dvar

```

Produces the following output:

```

$DVar = Jan-01-2004

!
! Reset date to first day of the month
! ($date1 and $date2 have been defined as date variables)
!
let $date1 = datenow()
move $date1 to $date2 'MMYYYY'
show '$Date2 = ' $date2 edit 'MM/DD/YY HH:MI'

```

Produces the following output if the report was run in October of 2004:

```

$Date2 = 10/01/04 00:00

!
! Convert date to a string
! ($date1 has been defined as a date variable)
!
move $date1 to $str_date 'DD-MON-YYYY'
show '$Str_Date = ' $str_date

```

Produces the following output:

```

$Str_Date = 01-DEC-2004

!
! Convert string (in partial format of SYYYYMMDDHHMISSNNN) to a
! date
!
move '20041129' to $date1
show '$Date1 = ' $date1 edit 'Mon DD YYYY HH:MI'

```

Produces the following output:

```

$Date1 = Nov 29 2004 00:00

```

See Also

The LET command for information about copying, editing, or converting fields.

The EDIT parameter of the PRINT command for a description of the edit masks.

The ALTER-LOCALE command for a description of the arguments NUMBER-EDIT-MASK, MONEY-EDIT-MASK, and DATE-EDIT-MASK.

The PRINT command regarding the default date-time components as a result of moving an incomplete date to a date variable.

MULTIPLY

Syntax

```
MULTIPLY {src_num_lit|_var|_col} TIMES dst_num_var
[ROUND=nn]
```

Description

Multiplies the first field by the second and places the result into the second field.

When dealing with money-related values (dollars and cents), use decimal variables rather than float variables. Float variables are stored as double-precision floating-point numbers, and small inaccuracies can appear when the program multiplies many numbers in succession. These inaccuracies can appear due to the way different hardware and software implementations represent floating point numbers.

Parameters

<i>Parameter</i>	<i>Description</i>
src_num_lit _var _col	Specifies a numeric source column, variable, or literal.
dst_num_var	Specifies a destination numeric variable.
ROUND	Rounds the result to the specified number of digits to the right of the decimal point. For float variables, this value can be from 0 to 15. For decimal variables, this value can be from 0 to the precision of the variable. For integer variables, this argument is not appropriate.

Example

The following example illustrates the MULTIPLY command:

```
multiply &quantity times #cost
multiply 1.5 times #result
```

See Also

The ADD command for more information.

The LET command for a discussion of complex arithmetic expressions.

NEW-PAGE**Syntax**

```
NEW-PAGE [erase_from_line_num_lit / _var / _col]
```

Description

Writes the current page and begins a new one.

For line printers, this command can optionally erase the old page starting at a specified line. After this action is performed, the location on the page is unchanged—that is, the value of #current-line is the same. The default action is to erase the entire page and reset #current-line to its initial value for the page.

In reports in which an overflow page is needed, sometimes retaining information from the first page on succeeding pages is useful.

Each NEW-PAGE occurrence adds a form feed character to the output file unless you specify FORMFEED=NO in the DECLARE-LAYOUT for this program in the SETUP section.

Note. A NEW-PAGE automatically occurs if page overflow is detected. Tabular reports do not require explicit NEW-PAGE commands; use NEXT-LISTING instead.

Parameters

<i>Parameter</i>	<i>Description</i>
<i>erase_from_line_num_lit</i> <i>_var</i> <i>_col</i>	Specifies a numeric column, variable, or literal for line printers.

Example

The following example illustrates the NEW-PAGE command:

```
! Write current page, then erase it
! beginning at line 5.
new-page 5
```

NEW-REPORT

Syntax

```
NEW-REPORT {report_filename_txt_lit|_var|_col}
```

Description

Closes the current report output file and opens a new one with the specified file name.

This command is normally used with single reports only. When used with multiple report declarations, this command affects the *current* report only.

The internal page counter is reset to 1 when NEW-REPORT is executed.

Note. SQR does not create a report output file until the first page is completed. NEW-REPORT might not create a new file, for example, if no data is selected and nothing is printed on the page.

Parameters

Parameter	Description
{report_filename_txt_lit _var _col}	Specifies a new file name.

Example

The following example shows the NEW-REPORT command:

```
new-report 'rep2a.lis'
new-report $next-file
```

Assign the report file name within an SQR report by issuing the NEW-REPORT command before printing. You might even prompt for the file name to use, as shown in the following example:

```
begin-report
  input $file 'Enter report file name'
  new-report $file
  ...
```

After execution of this command, the reserved variable *\$sqr-report* is updated to reflect the new report name.

See Also

DECLARE-REPORT, USE-REPORT

The -F command-line flag.

NEXT-COLUMN

Syntax

```
NEXT-COLUMN [ AT-END={NEWLINE|NEWPAGE} ]  
[ GOTO-TOP={num_lit|_var|_col} ]  
[ ERASE-PAGE={num_lit|_var|_col} ]
```

Description

Sets the current position on the page to the next column defined with the COLUMNS command.

Parameters

Parameter	Description
AT-END	Takes effect if the current column is the last one defined when NEXT-COLUMN is invoked.
GOTO-TOP	Causes the current line in the next column to be num_lit _var _col. This argument is useful when you are printing columns down the page.
ERASE-PAGE	Specifies where to begin erasing the page when an AT-END=NEWPAGE occurs.

Example

The following example prints columns across the page:

```
columns 10 50                                ! Define two columns  
begin-select  
name      (0,1,20)  
phone     (0,+3,0) edit (xxx)xxxx-xxxx  
  next-column at-end=newline                ! Print names  
                                              ! across the page  
from phonelist                               ! within two columns.  
order by name  
end-select
```

The following example prints columns down the page:

```
columns 10 50
move 55 to #bottom_line
begin-select
name      (0,1,20)
phone     (0,+3,0) edit (xxx)xxxx-xxxx
  if #current-line >= #bottom_line
    next-column goto-top=1 at-end=newpage
  else
    position (+1,1)
  end-if
from phonelist
order by name
end-select
```

See Also

COLUMNS, USE-COLUMN

NEXT-LISTING

Syntax

```
NEXT-LISTING[NO-ADVANCE]
[SKIPLINES={num_lit|_var|_col}]
[NEED={num_lit|_var|_col}]
```

Description

Ends the current set of detail lines and begins another.

NEXT-LISTING is used in tabular reports. This command causes a new vertical offset in the page to begin.

After NEXT-LISTING is executed, line 1 is reset one line below the deepest line previously printed in the page body. That is, if you then write PRINT (1, 5), the string is printed on the next available line starting in column 5. Note that the SQR-reserved variable #current-line still reflects the actual line number within the page body.

The value of SKIPLINES must be a nonnegative integer. If it is less than 0 (zero), then 0 is assumed.

The value of NEED must be an integer greater than 0. If it is less than or equal to 0, then 1 is assumed.

Parameters

Parameter	Description
NO-ADVANCE	Suppresses any line movement when no printing has occurred since the previous NEXT-LISTING or NEW-PAGE. The default increments the line position even when nothing was printed.

<i>Parameter</i>	<i>Description</i>
SKIPLINES	Causes the specified number of lines to be skipped before setting up the new offset.
NEED	Specifies the minimum number of lines needed to begin a new listing or set of detail lines. If this number of lines does not exist, a new page is started. You can use NEED to prevent a group of detail lines from being broken across two pages.

Example

The following example shows the NEXT-LISTING command:

```
begin-select
cust_num  (1,1)      edit 0999999      ! Each detail group prints
city      (,+3)      ! starting on line 1 since
name      (2,10,30)  ! NEXT-LISTING keeps
address    (,+2)      ! moving line 1 down the
  next-listing skiplines=1 need=2      ! page. NEED=2 keeps 2
from customers order by cust_num      ! line detail groups from
end-select           ! breaking across pages.
```

Note. The NEXT-LISTING command automatically issues a Use-Column 1 command if columns are active.

OPEN

Syntax

```
OPEN {filename_lit|_var|_col} AS
{filenum_num_lit|_var|_col}
{FOR-READING|FOR-WRITING|FOR-APPEND}
{RECORD=length_num_lit[:FIXED|:FIXED_NOLF|:VARY]}
[STATUS=num_var]
```

Description

Opens an operating system file for reading or writing. After a file is opened, it remains open until explicitly closed by the CLOSE command. A maximum of 256 files can be opened at one time.

Parameters

<i>Parameter</i>	<i>Description</i>
{filename_lit _var _col}	Specifies the file name. The file name can be literal, variable, or column. This makes prompting for a file name at runtime easy.

Parameter	Description
{ filenum_num_lit _var _col }	Specifies a number that identifies the file in the application. All file commands use the file number to reference the file. File numbers can be numeric variables and literals. The number can be any positive integer less than 64,000.
FOR-READING	When a file is opened for reading, SQR procures all data sequentially. SQR does not allow for random access of information.
FOR-WRITING	When a file is opened for writing, a new file is created. If a file of the same name already exists, it can be overwritten (this depends on the operating system).
FOR-APPEND	When a file is opened in append mode, the current file contents are preserved. All data written is placed at the end of the file. SQR creates the file if one does not already exist. For existing files, make sure that the attributes used are the same as those used when the file was created. Failure to do this may produce unpredictable results.
RECORD	For the VARY file type, this is the maximum size for a record. For the FIXED file type, this is the size of each record without the line terminator. For the FIXED_NOLF file type, this is the size of each record.
FIXED	This file type assumes that all records contained within the file are the same length. Terminate each record by a line terminator (system-dependent). You can use this file type when writing or reading binary data.
FIXED_NOLF	This file type specifies that all records contained within the file are the same length with no line terminators. When writing records, SQR pads short records with blank characters to ensure each record is the same length. This file type can be used when SQR is writing or reading binary data.
VARY	This file type specifies that the records can be of varying length. Each record is terminated by a line terminator (system-dependent). Only records containing display characters (no binary data) can be used safely. When SQR is reading records, any data beyond the maximum length specified is ignored. This is the default file type.
STATUS	Sets the numeric variable to zero if the OPEN command succeeds and to -1 if it fails. Without the STATUS argument, a failure on OPEN causes SQR to halt. By using a STATUS variable, you can control what processing should occur when a file cannot be opened.

Example

This section contains two examples: a regular open command and an expanded command:

```

open 'stocks.dat' as 1  for-reading record=100
open 'log.dat'     as 5  for-writing  record=70
open $filename    as #j for-append  record=80:fixed
open $filename    as 2  for-reading record=80:fixed_nolf

open $filename    as 6  for-reading record=132:vary status=#filestat
if #filestat != 0
    ... error processing ...
end-if

```

An encoding directive added to the OPEN command allows differently encoded files to be managed in a single run of SQR. When no ENCODING is specified on the OPEN command, SQR uses the file input or output encoding specified in the INI file unless the file is UCS-2 encoded and auto-detection of UCS-2 files is enabled. The complete syntax of the OPEN command is:

```
OPEN {filename_lit | _var | _col} AS {filenum_num_lit |
_var | _col}
{ FOR-READING | FOR-WRITING | FOR-APPEND }
{ RECORD = length_num_lit[:FIXED | :FIXED_NOLF |
:VARY]}
[ STATUS = num_var ]
[ ENCODING = { _var | _col | ASCII | ANSI | SJIS | JEUC
| EBCDIC |
EBCDIK290 | EBCDIK1027 | UCS-2 | UTF-8 | others... }

```

The ENCODING directive is allowed only when you are converting to Unicode internally.

See Also

The READ, WRITE, and CLOSE commands for information about using files.

PAGE-NUMBER

Syntax

```
PAGE-NUMBER position[pre_txt_lit[post_txt_lit]]
```

Description

Places the current page number on the page.

The text specified in *pre_txt_lit* and *post_txt_lit* is printed immediately before and after the number.

Parameters

<i>Parameter</i>	<i>Description</i>
position	Specifies the position of the page number. See the POSITION command for a description and examples of the <i>position</i> parameter.
pre_txt_lit	Specifies a text string to be printed before the page number.
post_txt_lit	Specifies a text string to be printed after the page number.

Example

The following example shows the PAGE-NUMBER command:

```
begin-footing 1
  page-number      (1,37) 'Page '      ! Will appear as
  last-page        ( ) ' of ' ' .'      ! "Page 12 of 25."
end-footing
```

See Also

LAST-PAGE

POSITION

Syntax

```
POSITION position
[@document_marker[COLUMNS{num_lit|_var|_col}
num_lit|_var|_col]. . . ]
```

Description

Sets the current position on a page.

Parameters

Parameter	Description
position	A position qualifier consisting of (<i>line, column, width</i>), where <i>column</i> and <i>width</i> are numbers that denote characters and spaces. <i>Line, column, and width</i> are all optional. If line or column is omitted, the print position is set by default to the current position, which is the position following the last printed item. Width is set by default to the width of the text that is being printed. A plus sign or minus sign indicates relative positioning in SQR. A plus sign moves the print position forward from the current position, and a minus sign moves it back.
@document_markerg	References a location defined in a document paragraph. In this case, the position used is the location of that marker in the text of the document.
COLUMNS	Defines columns beginning at the location of the document marker. The columns defined are relative to the position of the document marker. When COLUMNS is used, the entire command cannot be broken across more than one program line.

Example

The following example shows the POSITION command:

```

position (12,5)           ! Set current position to line 12, column 5.
position (+2,25)          ! Set position 2 lines down, at 25th column.
position () @total_location ! Set position to document
print #total () edit 999,999,999 ! marker @total_location.
position () @name_loc columns 1 30
print name ()             ! Columns are defined at @name_loc and
next-column              ! 29 characters to the right of @name_loc
print title ()

```

See Also

The COLUMNS command for more information.

Enterprise PeopleTools 8.50 PeopleBook: SQR for PeopleSoft Developers, "Creating Form Letters"

PRINT

Syntax

```
PRINT {any_lit|_var|_col} position[format_command[format_cmd_params]...]...
```

Description

Puts data on the page at a specified position.

See *Enterprise PeopleTools 8.50 PeopleBook: SQR for PeopleSoft Developers, "Changing Fonts."*

Parameters

Parameter	Description
{any_lit _var _col}	Specifies the data to be printed. Dates can be contained in a date column or variable, or in a string literal, column, or variable. When you are using EDIT or DATE with the PRINT command, a date in a string literal, column, or variable must be in an acceptable format. See the description of <i>EDIT</i> for further details.
position	A position qualifier consisting of (<i>line</i> , <i>column</i> , <i>width</i>), where <i>column</i> and <i>width</i> are numbers that denote characters and spaces. <i>Line</i> , <i>column</i> , and <i>width</i> are all optional. If line or column is omitted, the print position is set by default to the current position, which is the position following the last printed item. Width is set by default to the width of the text that is being printed. Position can be relative. See the POSITION command for a full description and examples of relative positioning.
format_command [format_cmd_params]	Specifies optional formatting commands and parameters.

Format Commands

The PRINT command has the following format commands::

- BACKGROUND
- BOLD
- BOX
- CENTER
- CODE-PRINTER
- DATE
- EDIT
- FILL
- FOREGROUND
- MONEY
- NOP
- NUMBER
- ON-BREAK
- SHADE
- UNDERLINE
- WRAP

Note. The SHADE and BOX format options are not supported for output to HTML format.

Some of these format commands can be used in combination with others and some are mutually exclusive. The following tables show which can be used together. An X indicates that they can be used together.

	BACKGROUND/ FOREGROUND	BOLD	BOX	CENTER	CODE- PRINTER	DATE/ EDIT/ MONEY/ NUMBER
BACKGROU ND/ FOREGROU ND		X		X		X
BOLD	X		X	X		X
BOX	X	X		X		X

	BACKGROUND/ FOREGROUND	BOLD	BOX	CENTER	CODE- PRINTER	DATE/ EDIT/ MONEY/ NUMBER
CENTER	X	X	X			X
CODE- PRINTER						
DATE/ EDIT/ MONEY/ NUMBER	X	X	X	X		
FILL	X	X	X	X		
MATCH	X	X	X	X		
NOP	X	X	X	X	X	X
ON-BREAK	X	X	X	X		X
SHADE	X	X	X	X		X
UNDERLINE	X	X		X		X
WRAP	X	X	X			

	FILL	MATCH	NOP	ON- BREAK	SHADE	UNDERLINE	WRAP
BACKGROU ND/ FOREGROU ND	X	X	X	X	X		X
BOLD	X	X	X	X	X	X	X
BOX	X	X	X	X	X		X
CENTER	X	X	X	X	X	X	

	<i>FILL</i>	<i>MATCH</i>	<i>NOP</i>	<i>ON-BREAK</i>	<i>SHADE</i>	<i>UNDERLINE</i>	<i>WRAP</i>
CODE-PRINTER			X				
DATE/ EDIT/ MONEY/ NUMBER			X	X	X	X	
FILL			X	X	X	X	X
MATCH			X	X	X	X	
NOP	X	X		X	X	X	X
ON- BREAK	X	X	X		X	X	X
SHADE	X	X	X	X		X	X
UNDERLINE	X	X	X	X	X		X
WRAP	X		X	X	X	X	

The following sections describe the PRINT format commands.

BOLD

BOLD causes the string or number to print in **bold** type.

For HP LaserJet printers, the appropriate boldface font must be loaded in the printer.

For PostScript printers, the appropriate boldface must be defined in the PostScript startup file, postscri.str.

See the DECLARE-PRINTER Command Arguments table for information about which fonts can be boldfaced.

For line printers, when the BEFORE-BOLD and AFTER-BOLD arguments on the DECLARE-PRINTER command are used, the specified strings are added before and after the data to be boldfaced. If BEFORE-BOLD and AFTER-BOLD are not specified, BOLD has no effect.

For example:

```
print &name (+1, 20) bold
print 'Your account is in arrears' (1,1) bold
```

BOX

BOX draws a one-line-deep graphical box around the printed data. This option has no effect for line printers.

For example:

```
print &grand_total (+5, 20) box
print 'Happy Birthday !!' (1,1) box
```

Note. For HP LaserJet printers using proportional fonts, BOX and SHADE are not able to determine the correct length of the box because it varies with the width of the characters printed. BOX and SHADE work well with fixed-pitch fonts and with all PostScript fonts.

Note. The BOX format option is not supported for output to HTML format.

CENTER

CENTER centers the field on a line. The position qualifier for column is ignored. For example:

```
print 'Quarterly Sales' (1) center
```

CODE-PRINTER

CODE-PRINTER has the following syntax:

```
CODE-PRINTER = printer_type
```

Valid values for *printer_type* are HT, HP, PS, LP, HTML, HPLASERJET, POSTSCRIPT, and LINEPRINTER.

CODE-PRINTER

Adds nondisplay characters to the program for the purpose of sending a sequence to the printer. CODE-PRINTER causes the string to be placed *behind* the page buffer, rather than within it, so alignment of printed data is not affected by the white space consumed by the nondisplay characters. Only strings can be printed using CODE-PRINTER.

Because the report might be printed on different types of printers, you should specify for which type this data is to be used. The report is ignored if it is printed to a different type. If necessary, you can send a different sequence to another type with a second PRINT statement.

For example:

```
encode '<27>[5U' into $big_font
encode '<27>[6U' into $normal_font
...
print $big_font (0, +2) code-printer=lp
print &phone () edit '(xxx) xxx-xxxx'
print $normal_font () code-printer=lp
```

In the previous example, the two CODE-PRINTER arguments put the *\$big_font* and *\$normal_font* sequences into the output, without overwriting any data in the page buffer. Sequences printed with the CODE-PRINTER argument are positioned using the regular line and column positioning. However, unlike the PRINT command, the current print location after execution is the beginning location where the CODE-PRINTER string was placed. Multiple coded strings printed using the same line and column location appear in the output in the same sequence in which they were printed.

DATE

You cannot use DATE with numeric columns or variables. DATE indicates that the field is to be formatted using the DATE-EDIT-MASK from the current locale. (See the ALTER_LOCALE command.) If this entry is not set, then the date is printed according to the following rules shown in the Date table.

<i>Column Type</i>	<i>Default Mask</i>	<i>If not set</i>
DATETIME	SQR_DB_DATE_FORMAT	See the Default Database Formats table for the format that is used.
DATE	SQR_DB_DATE_ONLY_FORMAT	See the DATE Column Formats table for the format that is used.
TIME	SQR_DB_TIME_ONLY_FORMAT	See the TIME Column Formats table for the format that is used.

EDIT

EDIT has the following syntax:

```
EDIT edit_format
```

EDIT causes each field to be edited before it is printed. The three types of edits are::

- Text edit
- Numeric edit
- Date edit

The following table lists the text edit format characters:

<i>Character</i>	<i>Description</i>
X	Use character in field.
B	Insert blank.
~ (tilde)	Skip character in field.
R[n]	Reverse sequence of string, for languages such as Hebrew. The optional number indicates right justification within length indicated.

Any other character, such as punctuation, in a text edit mask is treated as a constant and is included in the edited field.

The characters 8, 9, 0, V, and \$ are illegal in a text edit mask because they are used to indicate that the mask is for a numeric edit.

The following table lists the numeric edit format characters:

<i>Character</i>	<i>Description</i>
8	Digit, zero fill to the right of the decimal point, trim leading blanks (left-justify the number).
9	Digit, zero fill to the right of the decimal point, space fill to the left.
0	Digit, zero fill to the left.
\$	Dollar sign, optionally floats to the right.
B	Treated as a 9, but if a value is zero, the field is converted to blanks.
C	Entered at the end of the mask, causes the comma and period characters to be transposed when the edit occurs. This is to support monetary values for which periods delimit thousands and commas delimit decimals. (Example: 1.234,56).
E	Scientific format. The number of 9s after the decimal point determines the number of significant digits displayed. The E can be uppercase or lowercase; the display follows the case of the mask.
V	Implied decimal point.
MI	Entered at the end of the mask, causes a minus to appear at the right of the number.
PR	Entered at the end of the mask, causes angle brackets (< >) to be displayed around the number if the number is negative.
PS	Entered at the end of the mask, causes parentheses to be displayed around the number if the number is negative.
PF	Entered at the end of the mask, causes floating parentheses to be displayed around the number if the number is negative.

Character	Description
NA	Entered at the end of the mask, causes NA to be displayed if the numeric column variable is null. The case of NA follows that of the mask.
NU	Entered at the end of the mask, causes blanks to be displayed if the numeric column variable is null.
.	Decimal point.
,	Comma.

Characters other than those listed in the Numeric Edit Format Characters table are illegal for numeric edit masks and cause errors during processing.

The following table shows sample edit masks and resulting fields:

Mask	Value	Display
999.99	34.568	34.57
9,999,999V9999	123,456.7890	123,4567890
8,888,888.888	123,456.789	123,456.789
9,999	1234	1,234
9,999	123	123
09999	1234	01234
9999	-123	-123
9999	-1234	****
9999	12345	****
9999mi	-123	123-

<i>Mask</i>	<i>Value</i>	<i>Display</i>
9999pr	-123	< 123>
999999ps	-123	(123)
999999pf	-123	(123)
9999na	(null)	NA
9999nu	(null)	(blank)
\$\$9,999.99c	1234.56	\$1.234,56
\$\$9,999.99	1234.56	\$1,234.56
\$\$9,999.99	12.34	\$ 12.34
\$\$\$,\$\$9.99	12.34	\$12.34
9.999e	123456	1.235e+05
B9,999	0	(blank)
B9,999	12345	12,345
(xxx)bxxx-xxxx	2169910551	(216) 991-0551
xxx-xx-xxxx	123456789	123-45-6789
~~xx~xx	ABCDEFGHJIJ	CDFG
r10	ABCDEFGF	GFEDCBA

The following example shows some uses of edit masks:

```

print #total (7,55,0) edit $999,999.99 ! $ 12,345.67
print #total (7,55,0) edit $$$9,999.99 ! $12,345.67
print #total (7,55,0) edit 999,999.99pr ! < 12,345.67>(if neg)
print #comm (7,55,0) edit b99,999.99 ! Blank if zero
print #cnum (16,1,0) edit 099999 ! 001234
print #cat (5,10,0) edit 9.999E ! 1.235E+04
print #phone (16,60,0) edit (xxx)bxxx-xxxx ! (216) 397-0551
print #total (7,55,0) edit fff9,999.99 ! Dollar-Symbol f

```

The following table lists the date edit format codes:

Character	Description
YYY YY Y	Last 3, 2, or 1 digits of the year. On input, for calculating the 4-digit year, the current century, decade, or both are used. For example, a 9 using the Y mask would result in the year 1999 if the current year is in the 1990s.
YYYY SYYYY	4-digit year, S prefixes BC dates with "-".
RR	Last 2 digits of year; for years in other centuries. See the Date Edit Format Code-RR table.
CC or SCC	Century; S prefixes BC dates with "-".
BC AD	BC/AD indicator.
Q	Quarter of year (1,2,3,4; JAN-MAR=1).
RM	Roman numeral month (I-XII; JAN=I).
WW	Week of year (1-53) where week 1 starts on the first day of the year and continues to the seventh day of the year.
W	Week of the month (1-5) where week 1 starts on the first day of the month and ends on the seventh.
DDD	Day of year (1-366).
DD	Day of month (1-31).
D	Day of week (1-7). Sunday is the first day of the week.

<i>Character</i>	<i>Description</i>
DAY	Name of day.
DY	Abbreviated name of day.
ER	Japanese Imperial Era. Returns the name of the Japanese Imperial Era in the appropriate kanji (Heisei is the current era).
EY	Year of the Japanese Imperial Era. Returns the current year within the Japanese Imperial Era. Note. The common Japanese date format is 'YYYY<nen>MM<gatsu>DD<nichi>' where <nen>, <gatsu>, and <nichi> are the kanji strings for year, month, and day respectively.
J	Julian day; the number of days since Jan 1, 4713 BC. Numbers specified with <i>J</i> must be integers.
AM PM	Meridian indicator.
HH	Assumes 24-hour clock unless meridian indicator is specified.
HH12	Hour of day (1–12).
HH24	Hour of day (0–23).
SSSSS	Seconds past midnight (0–86399).
N NN NNN NNNN NNNNN NNNNNN	Fractions of a second. Precise to microseconds; however, for most hardware and databases, this much accuracy is not attainable.
MONTH	Name of month.
MON	Abbreviated name of month.
MM	Month (01–12; JAN=01).

Character	Description
MI	Minute (0–59).
SS	Second (0–59).
	Used to concatenate different masks.

Note. If the last two digits of the year are between 00 and 49, the return date is in the current century. If the last two digits of the year are between 50 and 99, the return date is in the century after the current one.

Last 2 Digits of Current Year	2-Digit Year is 00–49	2-Digit Year is 50–99
00–49	The return date is in the current century.	The return date is in the century before the current one.
50–99	The return date is in the century after the current one.	The return date is in the current century.

All masks can be used by the strtodate function except for CC, SCC, Q, W, and WW.

A backslash forces the next character into the output from the mask. For example, a mask of *The cu\rre\nt\mo\nth is Month* results in the output string of *The current month is January*. Without the backslashes the output string would be *The cu04e7t january is January*.

A vertical bar can be used as a delimiter between format codes; however, in most cases the bar is not necessary. For example, the mask 'YYYY|MM|DD' is the same as 'YYYYMMDD'.

Any other character, such as punctuation, in a date edit mask is treated as a constant and is included in the edited field. If the edit mask contains spaces, it must be enclosed in single quotes (').

The masks MON, MONTH, DAY, DY, AM, PM, BC, AD, and RM are case-sensitive and follow the case of the mask entered. For example, if the month is January, the mask Mon yields *Jan* and MON yields *JAN*.

All other masks are case-insensitive and can be entered in either uppercase or lowercase. In addition, National Language Support is provided for the following masks: MON, MONTH, DAY, DY, AM, PM, BC, and AD.

See the ALTER-LOCALE command in the SQR Samples section for additional information.

If the value of the date field being edited is *Mar 14 2004 9:35*, the edit masks produce the results in the following table.

Edit Mask	Result
dd/mm/yy	14/03/04
DD-MON-YYYY	14-MAR-2004

Edit Mask	Result
'Month dd, YYYY'	March 14, 2004
MONTH-YYYY	MARCH-2004
HH:MI	09:35
'HH:MI PM'	09:35 AM
YYYYMMDD	20040314
MM.DD.YYYY	03.14.2004
Mon	Mar
DD D DDD	143073

In addition to being used with the EDIT argument, edit masks can be used with the MOVE, CONCAT, DISPLAY, and SHOW commands, and with the edit function of the LET command. You edit the field using the supplied mask before storing or displaying it.

When a date with missing date components, time components, or both is displayed or printed, the defaults are as follows:

- The default year is the current year.
- The default month is the current month.
- The default day is one.
- The default time is zero (00:00:00.000000).

For example, assuming today is September 7, 2004, the following assignment would produce an equivalent date-time of September 1, 2004 13:21:00.000000:

```
let $date1 = strtodate('13:21','HH:MI')
```

Edit masks can be changed dynamically by storing them in a string variable and referencing the variable name preceded by a colon (:). For example:

```
move '$999,999.99' to $mask
print #total (5,10) edit :$mask
show #total edit :$mask
```

When a date that is stored in a string literal, column, or variable is printed with an edit mask, it must be in one of the following formats:

- The format specified by the environment variable `SQR_DB_DATE_FORMAT`, or the corresponding setting in the `sqr.ini` file.
- One of the database-dependent formats, as listed in the Default Database Formats table.
- The database-independent format, 'YYYYMMDD[HH24[MI[SS[NNNNNN]]]]'.

When a date column or variable is printed without an edit mask, the date is printed in the format specified by the environment variable `SQR_DB_DATE_FORMAT` or the corresponding setting in the `sqr.ini` file. If this has not been set, then the date is printed in the primary database format (the first entry) listed in the Default Database Formats table.

This applies to `DISPLAY`, `MOVE`, and `SHOW` commands as well as `PRINT`.

The following table lists default date formats for each database.

Database	Default Database Formats
Oracle	DD-MON-YY
Informix	YYYY-MM-DD HH:MI:SS.NNN MM/DD/YYYY MM-DD-YYYY MM.DD.YYYY
ODBC	MM-DD-YYYY
DB2	YYYY-MM-DD-HH:MI:SS.NNNNNN YYYY-MM-DD
Sybase	MON DD YYYY HH:MIPM MON DD YYYY [HH:MI[:SS[:NNN]]][PM] MON DD YYYY [HH:MI[:SS[:.NNN]]][PM] YYYYMMDD [HH:MI[:SS[:.NNN]]PM] YYYYMMDD [HH:MI[:SS[:.NNN]]PM]

Database	DATE Column Formats
DB2	YYYY-MM-DD
Informix	MM/DD/YYYY
ODBC	DD-MON-YYYY

Database	TIME Column Formats
DB2	HH24.MI.SS

<i>Database</i>	<i>TIME Column Formats</i>
ODBC	HH24:MI:SS

FILL

FILL fills the page with the specified character or string as indicated by the print position and length.

The following example prints a line of stars and then a line of dashes followed by stars:

```
print  '*'   (1,1,79)  fill      ! Fill line with '*'s
print  '-*'   (+1,20,40) fill    ! Fill with '-*' characters.
```

FOREGROUND/BACKGROUND

When you specify a color on the PRINT command, its scope is that of the PRINT command. If you do not define the specified color name, then the setting for *default* is used. Use the color name *none* to turn off color for the specified area.

```
PRINT {any_lit|_var|_col}
[FOREGROUND =({color_name_lit|_var|_col}|{rgb})]
[BACKGROUND =({color_name_lit|_var|_col}|{rgb})]
```

MATCH

MATCH has the following syntax:

```
MATCH match_text { line_num_lit|_var|_col }
{ column_num_lit|_var|_col } print_text ...
```

MATCH compares a field to a list of key values and if a match is found, prints the corresponding string at the specified line and column. If the match_text contains white space, it must be enclosed in single quotes ('). Any number of match texts can be tested, but each must have its own line, column, and print_text. If a match is not found, the unmatched field is printed at the position specified in the parentheses. Line and column positions for each matched string are treated as fixed or relative positions depending on the type of positioning used in the position qualifier for the PRINT command. For example:

```
print  &type_buyer  (20,12)  match
A 20 12 Casual
B 20 22 Impulsive
C 21 12 Informed
D 21 22 Choosey
```

To use relative line and fixed-column positioning, for example, you could use the following example:

```
print  $state  (0,25)  match
OH 0 25 Ohio
MI 0 37 Michigan
NY 0 25 'New York'
```

The column positions are treated as fixed locations due to the fixed 25 position declared in parentheses.

MONEY

MONEY indicates that the column or variable is to be formatted using the MONEY-EDIT-MASK from the current locale (See the ALTER_LOCALE command). This can be used only with a numeric column or variable.

NOP

NOP suppresses the print command, causing *no operation* to be executed. This argument is useful for temporarily preventing a field from printing. For example:

```
print &ssn (1,1) nop
  Hide the social security number.
```

NUMBER

NUMBER indicates that the column or variable is to be formatted using the NUMBER-EDIT-MASK from the current locale (See the ALTER-LOCALE command). This argument can be used only with a numeric column or variable.

ON-BREAK

ON-BREAK has the following syntax:

```
ON-BREAK [PRINT={ALWAYS|CHANGE|CHANGE/TOP-PAGE|NEVER}]
[SKIPLINES={num_lit|_var|_col}]
[PROCEDURE=procedure_name[(arg1[ ,argi]...)] ]
[AFTER=procedure_name[(arg1[ ,argi]...)] ]
[BEFORE=procedure_name[(arg1[ ,argi]...)] ]
[SAVE=txt_var]
[LEVEL=nn]
[SET=nn]
```

ON-BREAK causes the specified action in a tabular report when the value of a field changes (a break occurs). The default action prints the field only when its value changes (PRINT=CHANGE). ON-BREAK has the following qualifiers:

- PRINT specifies when the break field is printed.
 - ALWAYS duplicates the break field for each detail group.
 - CHANGE prints the value only when it changes.
This is the default.
 - CHANGE/TOP-PAGE prints the value both when it changes and at the top of each new page.
 - NEVER suppresses printing.
- SKIPLINES specifies how many lines to skip when the value changes.
- PROCEDURE specifies the procedure to be invoked when the value changes.

This qualifier cannot be used with either the AFTER or BEFORE qualifier.

- AFTER and BEFORE procedures specify procedures to invoke either after or before the value changes.

If no rows are fetched, neither procedure is executed. AFTER and BEFORE can be used only within a SELECT paragraph.

- The sequence of events is shown here:

- SAVE indicates a string variable in which the previous value of a break field is stored.
- LEVEL specifies the level of the break for reports containing multiple breaks.

For example, a report sorted by state, county, and city might have three break levels: state is level 1 (the most major), and city is level 3 (the most minor). When a break occurs, other breaks with equal or higher level numbers are cleared. The level number also affects the sequence in which AFTER and BEFORE procedures are processed.

- SET assigns a number to the set of leveled breaks in reports with more than one set of independent breaks.

Following is the sequence of events for a query containing ON-BREAK fields:

1. Any BEFORE procedures are processed in ascending LEVEL sequence before the first row of the query is retrieved.
2. When a break occurs in the query, the following events occur.:
 - AFTER procedures are processed in descending sequence from the highest level to the level of the current break field.
 - SAVE variables are set with the new value.
 - BEFORE procedures are processed in ascending sequence from the current level to the highest level break.
 - Any breaks with the same or higher level numbers are cleared so that they do not break on the next value.
 - If a PROCEDURE has been declared, the procedure is invoked.
 - If SKIPLINES was specified, the current line position is advanced.
 - The value is printed (unless PRINT=NEVER was specified).

3. After the query finishes (at END-SELECT) any AFTER procedures are processed in descending level sequence, for example:

```
begin-select
state  (+1,1,2)  on-break level=1  after=state-tot  skiplines=2
county (+,2,14) on-break level=2  after=cnty-tot   skiplines=1
city   (+,2,14) on-break level=3  after=city-tot
...
end-select
```

The breaks are processed as follows:

- When city breaks, the city-tot procedure is executed.
- When county breaks, first the city-tot procedure is executed, then the cnty-tot procedure is executed.
- When state breaks, the city-tot, cnty-tot, and state-tot procedures are processed in that sequence.

If any BEFORE breaks were indicated, they would also be processed automatically, after all of the AFTER breaks and in sequence from lower to higher level numbers, for example:

```
begin-select
state  (+1,1,2) on-break level=1 before=bef-state after=state-tot
county (+,2,14) on-break level=2 before=bef-cnty after=cnty-tot
city   (+,2,14) on-break level=3 before=bef-city after=city-tot
...
end-select
```

Now when state breaks, this sequence of procedures is executed:

- City-tot
- Cnty-tot
- State-tot
- Bef-state
- Bef-cnty
- Bef-city

When program flow enters the query at BEGIN-SELECT, the three BEFORE procedures are executed in sequence:

- Bef-state
- Bef-cnty
- Bef-city

After the last row is retrieved, at END-SELECT, the three AFTER procedures are executed in sequence:

- City-tot
- Cnty-tot
- State-tot

The SAVE qualifier saves the previous break value in the specified string variable for use in an AFTER procedure. You may want to print the previous break field with a summary line:

```
print &state (+1,1) on-break after=state-tot save=$old-state
```

The SET qualifier enables you to have subreports with leveled breaks. Because the ON-BREAKs are separated into sets, the associated leveled breaks in each set do not interfere with each other.

```
begin-select
state (+1,1,2) on-break set=1 after=state-tot level=1
```

SET=1 associates this leveled break with other breaks having the same set number.

SHADE

Draws a one-line deep, shaded graphical box around the printed data. For line printers, this argument has no effect.

```
print 'Company Confidential' (1,1) shade
print &state (+2, 40) shade
```

Note. For HP LaserJet printers using proportional fonts, BOX and SHADE are not able to determine the correct length of the box because it varies with the width of the characters printed. BOX and SHADE work well with fixed-pitch fonts and with all PostScript fonts.

Note. The SHADE format option is are not supported for output to HTML format.

UNDERLINE

UNDERLINE prints the specified data with underlined characters. For line printers, UNDERLINE generates backspace and underscore characters, which emulates underlining. For example:

```
print &name (+1, 45) underline
print 'Your account is in arrears' (1,1) underline
```

WRAP

WRAP wraps text at word spaces. Additional text is moved to a new line. WRAP has the following syntax:

```
WRAP {line_length_lit|_var|_col}
{max_lines_lit|_var|_col}[KEEP-TOP]
[STRIP=strip_chars][ON=break_chars][R]
[LINE-HEIGHT={line_height_lit|_var|_col}]

line_length_lit|_var|_col
```

WRAP specifies the maximum paragraph width in characters.

Note. After a string wraps, the current position is one character to the right of the last character in the column. When a string ends on the last position of a line, an implicit line feed causes the new current position to be the first character of the following line. In the SETUP section, use the DECLARE-LAYOUT command to make the page width one character wider than the right edge of the wrapped text to avoid generating an implicit line feed.

In this example, the line position is 1 for each of the three wrapped fields: note1, note2, and note3.

```

print &comment (48,20,0) wrap 50 3 ! Paragraph is 50
                                   ! characters wide,
                                   ! with a maximum
                                   ! depth of 3 lines.

print &note1      (1,20,30) wrap 30 4
print &note2      (1,+2,30) wrap 30 4
print &note3      (1,+2,30) wrap 30 4

```

The current print position after a wrap occurs at the bottom right edge of the wrapped paragraph. To continue printing on the same line, you must use a fixed line number for the next field.

```
max_lines_lit|_var|_col
```

Specifies the maximum paragraph depth in lines. Usually, the line length and maximum lines are indicated with numeric literals. However, WRAP can also reference numeric variables or columns. This is useful when you want to change the width or depth of a wrapped paragraph during report processing. The numeric variable can optionally be preceded by a colon (:), for example:

```

print $comments (1,30) wrap #wrap_width 6
print $message (5,45) wrap #msg_wid #msg_lines

```

KEEP-TOP retains the current line position except if a page break occurs, in which case, line 1 is used as the current line position. The default action is to set the next print position at the bottom of the wrapped data.

In the following example, the column &resolution prints on the same line as the first line of the column &instructions:

```

print &phone (+1,10) edit '(xxx) xxx-xxxx'
print &instructions (+1,10,30) wrap 6 30 keep-top
print &resolution (0,+3,25)

```

The STRIP and ON arguments affect which characters are to be converted before wrapping, and which characters force a wrap to occur.

- Characters in the STRIP string argument are converted to spaces before the wrap occurs.
- Characters in the ON string argument cause a wrap at each ON character found. The ON character is not printed.

Both arguments accept regular characters and nondisplay characters for which decimal values are surrounded by angled brackets, <nn>, where *nn* is a decimal number between 1 and 255, representing the character in the current character set of the operating system. For example, to print a long data type that contains embedded carriage returns, the setup would be:

```
print &long_field (5,20) wrap 42 30 on=<13>
```

The paragraph wraps at each carriage return, rather than at the usual word boundaries. If the ON character is not found within the width specified for the paragraph, the wrap occurs at a word space. The following example converts the STRIP characters to spaces before wrapping on either a line feed <10> or a space (the default):

```
print &description (20,10) wrap 50 22 strip=/\^@<13> on=<10>
```

WRAP can also be used to print reversed characters, for support of languages such as Hebrew. An R after the length and max_lines arguments causes the field to be reversed before the wrap takes place. In addition, the entire paragraph is right-justified within the length indicated.

```
! Reverse wrap, in 30 character field.
print &comment (2,35) wrap 30 5 r
print $notes (1,50) wrap 50 7 r
```

LINE-HEIGHT specifies the number of lines to skip between each line of the wrapped data. By default, a value of 1 (single space) is assumed. The following example prints the comment column with one blank line between each printed line for a maximum of four printed lines:

```
print &comment (1,1) wrap 40 4 line-height = 2           ! Double space text
```

See Also

The LET command for information about copying, editing, or converting fields.

The ALTER-LOCALE command for a description of the arguments NUMBER-EDIT-MASK, MONEY-EDIT-MASK, and DATE-EDIT-MASK.

DISPLAY, SHOW

PRINT-BAR-CODE

Syntax

```
PRINT-BAR-CODE position
{TYPE={bar_code_type_num_lit|_var|_col}}
{HEIGHT={bar_code_height_num_lit|_var|_col}}
{TEXT={bar_code_txt_lit|_var|_col}}
[CAPTION={bar_code_caption_txt_lit|_var|_col}]
[CHECKSUM={checksum_lit}]
```

Description

Prints industry standard bar codes. SQR supports the bar code types listed in the following table.

Type	Description	Text Length	Text Type	CHECKSUM RECOGNIZED
1	UPC-A	11, 13, or 16	9	
2	UPC-E	11, 13, or 16	9	
3	EAN/JAN-13	12, 14, or 17	9	
4	EAN/JAN-8	7, 9, or 12	9	
5	3 of 9 (Code 39)	1 to 30	9, X, p	y

<i>Type</i>	<i>Description</i>	<i>Text Length</i>	<i>Text Type</i>	<i>CHECKSUM RECOGNIZED</i>
6	Extended 3 of 9	1 to 30	9, X, x, p, c	y
7	Interleaved 2 of 5	2 to 30	9	y
8	Code 128	1 to 30	9, X, x, p, c	
9	Codabar	1 to 30	9	y
10	Zip+4 Postnet	5, 9, or 11	9	
11	MSI Plessey	1 to 30	9	y
12	Code 93	1 to 30	9, X, p	y
13	Extended 93	1 to 30	9, X, x, p	y
14	UCC-128	19	9	
15	HIBC	1 to 30	9	y

Parameters

<i>Parameter</i>	<i>Description</i>
position	Specifies the position of the upper left corner. Position parameters can be relative. See the POSITION command for examples of relative positioning. Document markers are not allowed. After the statement executes, the current position is returned to this location; however, the next listing line is the next line below the bottom of the bar code. (This is different from the way the PRINT command works.)
TYPE	Specifies the type of bar code to be printed. Types are shown in the Bar Code Types table.
HEIGHT	Specifies the height of the bar code in inches. The height must be between 0.1 and 2 inches. The code prints to the nearest one-tenth of an inch. For Zip+4 Postnet, the height of the bar code is fixed. The height should be between 0.2 and 2.0 for Zip+4 Postnet. If it is less than 0.2, the bar code extends above the position specified.

<i>Parameter</i>	<i>Description</i>
TEXT	Specifies the text to be encoded and printed. The number and type of text characters permitted or required depends on the bar code type. See the Bar Code Types table for specifications.
CAPTION	Specifies optional text to be printed under the bar code in the current font. SQR attempts to center the caption under the bar code; however, for proportional fonts this may vary slightly. CAPTION is not valid for Zip+4 Postnet. If specified, it is ignored.
CHECKSUM	Specifies an optional check sum to be computed and printed within the bar code. Valid values are YES and NO, where NO is the default. Note. Some barcode types ignore the CHECKSUM qualifier. See the Bar Code Types table for those barcode types for which CHECKSUM is relevant.

Example

This example shows how to use the PRINT-BAR-CODE command to create a UPC-A bar code:

```
begin-program
  print 'Sample UPC-A Barcode' (1,1)
  print-bar-code (3,1)
    type=1                      ! UPC-A
    height=0.3
    text='01234567890'
    caption='0 12345 67890'
end-program
```

This example shows how to use the PRINT-BAR-CODE command to create a ZIP+4 Postnet code:

```
begin-program
  print 'Sample Zip+4 Postnet' (1,1)
  print 'John Q. Public' (3,1)
  print '1234 Main Street' (4,1)
  print 'AnyTown, USA 12345-6789' (5,1)
  print-bar-code (7,1)
    type=10
    height=0.2
    text='12345678934'
end-program
```

Note. SQR does not check bar code syntax. See your bar code documentation for the proper formatting of certain bar codes.

PRINT-CHART

Syntax

```

PRINT-CHART[chart_name]position
DATA-ARRAY-ROW-COUNT={x_num_lit|_var|_col}
DATA-ARRAY-COLUMN-COUNT={x_num_lit|_var|_col}
DATA-ARRAY=array_name
[DATA-LABELS=data_labels_lit | _var | _col]
[COLOR-PALETTE=color_palette_lit | _var | _col]
[ITEM-COLOR=(Chart_item_keyword, txt_lit | var | (r,g,b))]
[DATA-ARRAY-COLUMN-LABELS={NONE|array_name|
{({txt_lit|_var}[, {txt_lit|_var}]...))}]
[CHART-SIZE=(chart_width_num_lit|_var,chart_depth_num_lit|_var)]
[TITLE={title_txt_lit|_var|_col}]
[SUB-TITLE={subtitle_txt_lit|_var|_col}]
[FILL={fill_lit|txt_var|_col}]
[3D-EFFECTS={3d_effects_lit|txt_var|_col}]
[BORDER={border_lit|txt_var|_col}]
[POINT-MARKERS={point_markers_lit|txt_var|_col}]
[TYPE={chart_type_lit|txt_var|_col}]
[LEGEND={legend_lit|txt_var|_col}]
[LEGEND-TITLE={legend_title_txt_lit|_var|_col}]
[LEGEND-PLACEMENT={legend_placement_lit|txt_var|_col}]
[LEGEND- PRESENTATION={legend_presentation_lit|txt_var|_col}]
[PIE-SEGMENT-QUANTITY-DISPLAY=      {pie_segment_quantity_display_lit|txt_var|_col
}]
[PIE-SEGMENT-PERCENT- DISPLAY={pie_segment_percent_display_lit
|txt_var|_col}]
[PIE-SEGMENT-EXPLODE={pie_segment_explode_lit
|txt_var|_col}]
[X-AXIS-LABEL={x_axis_label_txt_lit|_var|_col}]
[X-AXIS-MIN-VALUE={x_axis_min_value_lit|_num_lit|_var
|_col}]
[X-AXIS-MAX-VALUE={x_axis_max_value_lit|_num_lit|_var
|_col}]
[X-AXIS-SCALE={x_axis_scale_lit|txt_var|_col}]
[X-AXIS-MAJOR-TICK-MARKS={x_axis_major_tick_marks_lit
|txt_var|_col}]
[X-AXIS-MINOR-TICK-MARKS={x_axis_minor_tick_marks_lit
|txt_var|_col}]
[X-AXIS-MAJOR-INCREMENT={x_axis_major_increment_lit
|_num_lit|_var|_col}]
[X-AXIS-MINOR-INCREMENT={x_axis_minor_increment_lit
|_num_lit|_var|_col}]
X-AXIS-TICK-MARK-PLACEMENT=
{x_axis_tick_mark_placement_lit|txt_var|_col}]
[X-AXIS-GRID={x_axis_grid_lit|txt_var|_col}]
[Y-AXIS-LABEL={y_axis_label_lit|txt_var|_col}]
[Y-AXIS-MIN-VALUE={y_axis_min_value_lit|_num_lit
|_var|_col}]
[Y-AXIS-MAX-VALUE={y_axis_max_value_lit|_num_lit
|_var|_col}]
[Y-AXIS-SCALE={y_axis_scale_lit|txt_var|_col}]
[Y-AXIS-MAJOR-TICK-MARKS={y_axis_major_tick_marks_lit
|txt_var|_col}]
[Y-AXIS-MINOR-TICK-MARKS={y_axis_minor_tick_marks_lit
|txt_var|_col}]
[Y-AXIS-MAJOR-INCREMENT={y_axis_major_increment_lit
|_num_lit|_var|_col}]

```

```
[Y-AXIS-MINOR-INCREMENT={y_axis_minor_increment_lit
|_num_lit|_var|_col}]
[Y-AXIS-TICK-MARK-PLACEMENT=
  {y_axis_tick_mark_placement_lit|txt_var|_col}]
[Y-AXIS-GRID={y_axis_grid_lit|txt_var|_col}]
```

Note. If you do not define CHART-SIZE with this command, you must define it with DECLARE-CHART.

Description

Prints a chart. Only PostScript printers or HP printers that support HPGL (generally, this is HPLaserJet 3 and higher) render chart output.

The PRINT-CHART command directs SQR to generate a chart according to the named chart, if any, and the overridden attributes, if any.

Note. PRINT-CHART can be used without referencing a named chart if all required attributes for the DECLARE-CHART are supplied in addition to all its required parameters. The PRINT-CHART command directs SQR to display the chart on the current page using the attribute values at the moment the command is executed. Manipulation of chart attribute values has no effect on the appearance of the chart after the PRINT-CHART command has been executed. For example, if you execute a PRINT-CHART with TITLE=\$ttl and \$ttl='Encouraging Results', and then change the value of \$ttl to 'Discouraging Results' immediately afterward, then the chart is printed with first value, 'Encouraging Results'. PRINT-CHART expects the DATA-ARRAY to be organized in a particular way. See the Chart Array Field Types (fewer than four fields) table for details. PRINT-CHART fills the area defined by CHART-SIZE as much as possible while maintaining an aesthetically pleasing ratio of height to width. In cases in which the display area is not well suited to the chart display, the chart is centered within the specified region, and the dimensions are scaled to accommodate the region. Do not be alarmed if the chart does not fit exactly inside the box you have specified. It means that SQR has accommodated the shape of the region to provide the best looking chart possible. Chart commands used to send output to a line printer are ignored. Only PostScript printers or HP printers that support Hewlett Packard's HPGL (generally, this is HP LaserJet model 3 and higher) render chart output. If you attempt to print a chart to a LaserJet printer that does not support HPGL, the HPGL command output might become part of your output, leaving one or more lines of meaningless data across your report.

All the attributes defined for DECLARE-CHART are valid for the PRINT-CHART command. PRINT-CHART has five additional parameters. The position of the chart is described using the first parameter. The data that supports the chart is defined in the additional attributes: DATA-ARRAY, DATA-ARRAY-ROW-COUNT, DATA-ARRAY-COLUMN-COUNT, and DATA-ARRAY-COLUMN-LABELS.

As mentioned, each chart type meets a specific organizational requirement. The Chart Array Field Types (fewer than four fields) table describes these requirements.

Note. If the first field in the array designated by DATA-ARRAY is of type CHAR, then the value on the x-axis is the contents of that column. If the first field is not of type CHAR, then the value of the x-axis is the row number of the array designated by DATA-ARRAY, beginning with 1. Pie charts show the character value in the legend area. Histograms show the character value on the y-axis. XY-Scatter charts do not use the character value and none is needed in the array.

Note. If a PIE chart contains many small slices, the user must set the PIE-SEGMENT-QUANTITY-DISPLAY argument, the PIE-SEGMENT-PERCENT-DISPLAY argument, or both to NO to prevent the values from one slice overwriting the values of another slice.

Chart Type	Field 0	Field 1	Field 2	Field 3
PIE	Type=char Pie segment labels, the names associated with each segment	Type=num The value associated with each pie segment	(Optional) Type=char Pie segment explode flag setting, 'Y' or 'N'	
LINE BAR STACKED-BAR 100%-BAR OVERLAPPED-BAR HISTOGRAM AREA STACKED-AREA 100%-AREA	Type=char X-Axis values	Type=num Series 1 Y-Axis values	(Optional) Type=num Series 2 Y-Axis values	(Optional) Type=num Series 3... Y-Axis values
XY-SCATTER-PLOT	Type=num Series 1 X-Axis values	Type=num Series 1 Y-Axis values	(Optional) Type=num Series 2 X-Axis values	(Optional) Type=num Series 2 ... Y-Axis values
FLOATING-BAR	Type=char X-Axis values	Type=num Series 1 Y-Axis offset	Type=num Series 1 Y-Axis duration	(Optional) Type=Num Series 2 ... Y-Axis offset

Chart Type	Field 0	Field 1	Field 2	Field 3	Field 4
HIGH-LOW-CLOSE	Type=char X-Axis values	Type=num High value	Type=num Low value	Type=num Closing value	(Optional) Type=num Opening value

Parameters

Parameter	Description
chart_name	Specifies the name of the chart from the DECLARE-CHART command. This name is not necessary if you specify the CHART-SIZE and all other pertinent attributes in the PRINT-CHART command.
Position	(row, column) Specifies the position of the upper left corner. Position parameters can be relative. See the POSITION command for examples of relative positioning. Document markers are not allowed. After execution, the current position is returned to this location; however, the next listing line is the next line below the bottom of the chart area. (This is different from the way the PRINT command works.)

<i>Parameter</i>	<i>Description</i>
DATA-ARRAY	Specifies the name of the array containing the data to be plotted. This must be the name of an array defined with CREATE-ARRAY.
DATA-ARRAY- ROW-COUNT	Specifies the number of rows or sets of data to be used from the DATA-ARRAY. If the DATA-ARRAY has a greater number of rows, only DATA-ARRAY-ROW-COUNT is included in the chart.
DATA-ARRAY- COLUMN-COUNT	Specifies the number of columns to be used from the DATA-ARRAY. If the DATA-ARRAY has a greater number of columns, only DATA-ARRAY-COLUMN-COUNT is included in the chart.
DATA-ARRAY- COLUMN-LABELS	Specifies labels for each Y-Axis value of the data set (fields) in DATA-ARRAY. These labels are displayed in the legend box. Column labels are ignored for pie charts. See the Chart Array Field Types (fewer than four fields) table for applicable fields for each type of chart. For definitions of the other arguments, see the DECLARE-CHART Command Arguments table.

Example

In this example, a pie chart is printed without explicit reference to a chart declared with DECLARE-CHART. All necessary arguments must be supplied in PRINT-CHART.

```
.
.
create-array
    name=unit_sales
    size=12
    field=product:char
    field=units:number
    field=explode:char
.
.
print-chart (15, 20)
    title                = 'Green City Store Sales'
    sub-title            = '(Second Quarter)'
    chart-size           = (50, 28)
    type                 = pie
    data-array           = unit_sales
    data-array-column-count = 3
    data-array-row-count = 7
    3d-effects           = yes
    fill                 = color
```

See Also

DECLARE-CHART

PRINT-DIRECT

Syntax

```
PRINT-DIRECT
[ NOLF ]
[ PRINTER={LINEPRINTER|POSTSCRIPT|HPLASERJET|HTML|LP|PS|HP|HT} ]
{txt_lit|_var|_col}...
```

Description

Writes directly to the print output file without using the SQR page buffer.

PRINT-DIRECT can be used for special applications that cannot be accomplished directly with PRINT commands, such as initializing a page with graphics or other special sequences. Because this text is often printer-dependent and because the report can be printed on different types of printers that require different control characters, you can use the PRINTER qualifier to specify the printer type. If no PRINTER qualifier is specified, the command applies to all printer types.

When using PRINT-DIRECT in conjunction with PRINT commands, be aware that the SQR page buffer is copied to the output file only when each page is full or when a NEW-PAGE command is issued. One approach is to use PRINT-DIRECT commands inside a BEFORE-PAGE or AFTER-PAGE procedure (declared with the DECLARE-PROCEDURE command) so that they are coordinated with the information coming out of the page buffer.

Parameters

<i>Parameter</i>	<i>Description</i>
NOLF	Specifies that no carriage return and line feed is to be printed. By default, printed text is followed by a carriage return and line feed character.
PRINTER	Specifies the type of printer to which this text applies.
{txt_lit _var _col}	The text to be printed.

Example

The following example shows the PRINT-DIRECT command:

```
print-direct printer=ps '%%Page: ' $page-number
print-direct nolf printer=lp reset
```

PRINT-IMAGE

Syntax

```
PRINT-IMAGE[ image_name ] position
[ TYPE={ image_type_lit | _var | _col } ]
[ IMAGE-SIZE=( width_num_lit | _var | _col , height_num_lit
| _var | _col ) ]
[ SOURCE={ file_name_txt_lit | _var | _col } ]
```

Note. If TYPE, IMAGE-SIZE, and SOURCE are not defined in PRINT-IMAGE, they must be defined in DECLARE-IMAGE.

Description

Prints an image.

The PRINT-IMAGE command can be placed in any section of a report with the exception of the SETUP section. The image file pointed to can be any file of the proper format.

PRINT-IMAGE can be used without referencing a named image if all required attributes for the DECLARE-IMAGE are supplied in addition to all its required parameters.

Parameters

<i>Parameter</i>	<i>Description</i>
image_name	Specifies the name of an image specified by a DECLARE-IMAGE.
position	(row, column) Specifies the position of the upper left corner. Position parameters can be relative. See the POSITION command for examples of relative positioning. Document markers are not allowed. After execution, the current position is returned to this location. (This is different from the way the PRINT command works.)
TYPE	Specifies the image type. Types can be EPS-FILE, HPGL-FILE, GIF-FILE, JPEG-FILE, or BMP-FILE (for Microsoft Windows).
IMAGE-SIZE	Specifies the width and height of the image.
SOURCE	Specifies the name of a file containing the image.

Example

For PostScript:

```
print-image office-signature (50, 20)
print-image (50, 20)
  type          = eps-file
  source        = 'sherman.eps'
  image-size = (10, 3)
```

For Microsoft Windows:

```
print-image company-logo (+21, 25)
  type=bmp-file
  source='m:\logos\gustavs.bmp'
  image-size=(75,50)
```

See Also

DECLARE-IMAGE

PUT

Syntax

```
PUT {src_any_lit|_var|_col}...
INTO dst_array_name(element)[field[(occurs)]]...
```

Description

Moves data into an array.

Columns retrieved from the database and SQR variables or literals can be moved into an array. The array must have been created previously using the CREATE-ARRAY command.

Considerations for Using PUT

When a date variable or column is moved into a text or char array field, the date is converted to a string according to the following rules:

- For DATETIME columns and SQR DATE variables, SQR uses the format specified by the SQR_DB_DATE_FORMAT setting.
If this has not been set, SQR uses the first database-dependent format as listed the Default Database Formats table.
- For DATE columns, SQR uses the format specified by the SQR_DB_DATE_ONLY_FORMAT setting.
If this has not been set, SQR uses the format listed in the DATE Column Formats table.
- For TIME columns, SQR uses the format specified by the SQR_DB_TIME_ONLY_FORMAT setting.
If this has not been set, SQR uses the format as listed in the TIME Column Formats table.

When a string variable, column, or literal is moved to a date array field, the string must be in the format specified by the SQR_DB_DATE_FORMAT setting, one of the database-dependent formats as listed in the DATE Column Formats table is used, or the database-independent format 'SYYYYMMDD[HH24[MI[SS[NNNNNN]]]]' is used.

```
dst_array_name(element)
```

If array fields are listed, data is placed into each field in the sequence it is listed, in the occurrence specified for that field.

If array fields are not listed, data is placed into consecutive fields in the order in which they were defined in the CREATE-ARRAY command; data is copied into occurrence zero of each field of the element specified in the array.

```
field [ ( occurs ) ]
```

Array element and field occurrence numbers can be numeric literals (123) or numeric variables (#j).

If no occurrence is specified, occurrence zero is used.

Parameters

Parameter	Description
src_any_var	The source variable or literal to be moved into the array. Numeric variables, literals, and database columns can be put into number (decimal, float, integer) fields. String variables, literals, and database columns can be put into char, text, or date fields. Date variables can be put into date, char, or text fields.

Example

In the following example, the four variables &name,#count,\$date1, and \$code are placed in the first four fields defined in the names array. The data is put into the #jth element of the array.

```
put &name #count $date1 $code into names(#j)
```

The following command places #j2,#j3, and #j4 into the zero through 2nd occurrences of the tot field in the #jth element of the totals array.

```
put #j2 #j3 #j4 into totals(#j) tot(0) tot(1) tot(2)
```

The following command copies #count into the #j2th occurrence of the count field in the #jth element of the states array.

```
put #count into states(#j) count(#j2)
```

READ

Syntax

```
READ {filenum_lit|_var|_col} INTO {any_var:length_int_lit}...[STATUS=
status_num_var]
```

Description

Reads the next record of a file into the specified variables.

Text and binary data are parsed according to the following criteria:

- Text data is any string of characters. The length of the variable name indicates how many characters to place in the variable.

After text is transferred, trailing blanks in the variable are omitted.

- If the field was written as a date variable, then it can be read into a date variable or text variable.

When reading a date into a date variable, it must be in the format specified by the `SQR_DB_DATE_FORMAT` setting, one of the database-dependent formats as listed in the DATE Column Formats table is used, or the database-independent format 'YYYYMMDD[HH24[MI[SS[NNNNNN]]]' is used.

- Binary numbers can be 1, 2, or 4 bytes in length.

They must be read into numeric variables. Note that the bytes making up the binary number must be in the standard sequence expected by your operating system.

- When the program is reading binary data, the file must be opened with the `FIXED` or `FIXED-NOLF` qualifier.
- Only the integer portion of the number is represented with binary numbers.

To maintain the decimal portion of the number, convert the number to a string variable.

- If you use binary numbers, the file is not portable across platforms.

Different hardware represents binary numbers differently.

The total length indicated for the variables must be less than or equal to the length of the record being read.

If no more records exist to read, the `#end-file` reserved variable is set to 1; otherwise, it is set to 0 (zero). Your program should check this variable after each `READ` command.

If `STATUS` is specified, SQR returns 0 if the read is successful; otherwise, it returns the value of `errno`, which is system-dependent.

Parameters

<i>Parameter</i>	<i>Description</i>
<code>filenum_lit _var _col</code>	Specifies the number assigned in the <code>OPEN</code> command to the file to be read.
<code>any_var :length_int_lit</code>	Specifies one or more variables into which data from the record read are to be put. <i>length_int_lit</i> specifies the length of each field of data.
<code>STATUS</code>	Specifies an optional variable into which a read status is returned.

Example

The following example shows several READ commands:

```
read 1 into $name:30 $addr:30 $city:20 $state:2 $zip:5
read 3 into $type:2 #amount:2 #rate:1 $code:5 $date:11
read #j into #sequence:2 $name:20 $title:15
```

The following example shows a READ command that reads two dates. One is loaded into a date variable; the other is loaded into a string variable, which is then converted to a date using the strtodate function.

```
.
.
.
declare-variable
  date $date1 $date2
  text $text
end-declare
.
.
.
read 4 into $date1:18 $text1:18
let $date2 = strtodate($text1,'SYYYYMMDDHHMISSNNN')
           or
let $date2 = strtodate($text1)
```

The following example shows a READ command with an INSERT loop:

```
begin-sql
  begin transaction
end-sql

while 1      ! Infinite loop, exited by BREAK, below.
  read 10 into $company:40 $parent:30 $location:50
  if #end-file
    break    ! End of file reached.
  end-if
  begin-sql
    insert into comps (name, parent, location)
      values ($company, $parent, $location)
  end-sql
  add 1 to #inserts
  if #inserts >= 100
    begin-sql
      end transaction;
      begin transaction
    end-sql
    move 0 to #inserts
  end-if
end-while

begin-sql
  end transaction
end-sql
```

See Also

The OPEN, CLOSE, and WRITE commands for information about files.

ROLLBACK

Syntax

ROLLBACK

Description

An automatic rollback is performed whenever SQR ends due to program errors. ROLLBACK is useful in testing or in certain error conditions.

ROLLBACK is an SQR command and should not be used inside an SQL paragraph.

Note. The ROLLBACK command can be used with Oracle, DB2, Informix, and ODBC(Microsoft SQL Server is accessible only with SQR Server for ODBC). For Sybase and Microsoft SQL Server, use BEGIN TRANSACTION and ROLLBACK TRANSACTION within SQL paragraphs as in the following example. See the COMMIT command for an example of ROLLBACK.

Example

The following example shows the ROLLBACK command:

```
if #error-status = 1
    rollback
    stop
end-if
```

See Also

The COMMIT command.

SBTOMBS

Syntax

SBTOMBS { *txt_var* }

Description

Converts a single-byte character into a multibyte equivalent.

This command converts the specified string as follows: Any occurrence of a single-byte character that also has a multibyte representation (numerals, punctuation, roman characters, and katakana) is converted. This command also converts a sequence of a kana character followed by certain grammatical marks into a single multibyte character, which combines the two elements.

Parameters

<i>Parameter</i>	<i>Description</i>
txt_var	Specifies the string to be converted.

See Also

The TO_MULTI_BYTE function of the LET command.

SECURITY

Syntax

```
SECURITY
[SET=(sid [,sid]...)]
[APPEND=(sid [,sid]...)]
[REMOVE=(sid [,sid]...)]
[MODE=mode]
```

Description

Enables you to mark sections of a report for security purposes.

The SECURITY command can be repeated as many times as desired for the current report. After the SECURITY command is executed, all subsequent commands for the current report are constrained by the designated *sids* until the report ends or another SECURITY command executes.

You can use the SECURITY command wherever you use the PRINT command.

Parameters

<i>Parameter</i>	<i>Description</i>
SET	Sets the list of security IDs for subsequent commands. The previous list of security IDs is replaced by the specified security IDs. This argument is optional and can be used only once.
<i>sid</i>	Can be any string literal, column, or variable. The value is case-sensitive.
APPEND	Appends the specified security IDS to the current list. This argument is optional and can be used multiple times.
REMOVE	Removes the specified security IDS from the current list. This argument is optional and can be used multiple times.

<i>Parameter</i>	<i>Description</i>
MODE	Used to enable (reactivate) or disable (suspend) the security feature for the current report. This argument is optional and can be used only once.
<i>mode</i>	Can be any string literal, column, or variable. The value is not case-sensitive and can be either ON or OFF.

Example

The following example shows the SECURITY command:

```

Begin-Report
  Security Set=('Directors', 'Vice-Presidents')
    .
    .           ! Only Directors and VPS can see this
    .
  Security Remove=('Directors')
    .
    .           ! Only VPS can see this
    .
  Security Mode='Off'
    .
    .           ! Anybody can see this
    .
  Security Mode='On' Append=('Managers')
    .
    .           ! Only VPs and Managers can see this
    .
  Security Append=('Engineers')
    .
    .           ! Only VPs, Managers, and Engineers can see this
    .
End-report

```

SET-COLOR

Syntax

```

SET-COLOR
  [PRINT-TEXT-FOREGROUND=( {color_name_lit | _var | _col | {rgb} } ) ]
  [PRINT-TEXT-BACKGROUND=( {color_name_lit | _var | _col | {rgb} } ) ]

```

Description

Defines default colors.

The SET-COLOR command is allowed wherever the PRINT command is allowed. If the specified color name is not defined, SQR uses the settings for the color name 'default.' Use the color name 'none' to turn off color for the specified area.

Parameters

<i>Parameter</i>	<i>Description</i>
PRINT-TEXT- FOREGROUND	Defines the color in which the text is printed.
PRINT-TEXT- BACKGROUND	Defines the color to print as a background behind the text.
{ <i>color_name_lit</i> <i>_var</i> <i>_col</i> }	A <i>color_name</i> is composed of alphanumeric characters (A–Z, 0–9), the underscore (_) character, and the hyphen (-) character. It must start with an alphabetical (A–Z) character and is case-insensitive. The name 'none' is reserved and cannot be assigned a value. A name in the format (RGBredgreenblue) cannot be assigned a value. The name 'default' is reserved and can be assigned a value. 'Default' is used during execution when a referenced color is not defined in the runtime environment.
{ <i>rgb</i> }	<p><i>red_lit</i> <i>_var</i> <i>_col</i>, <i>green_lit</i> <i>_var</i> <i>_col</i>, <i>blue_lit</i> <i>_var</i> <i>_col</i> where each component is a value in the range of 000 to 255. In the BEGIN-SETUP section, only literal values are allowed.</p> <p>The default colors implicitly installed with SQR include:</p> <p>black=(0,0,0) white=(255,255,255) gray=(128,128,128) silver=(192,192,192) red=(255,0,0) green=(0,255,0) blue=(0,0,255) yellow=(255,255,0) purple=(128,0,128) olive=(128,128,0) navy=(0,0,128) aqua=(0,255,255) lime=(0,128,0) maroon=(128,0,0) teal=(0,128,128) fuchsia=(255,0,255)</p>

Example

The following example shows the SET-COLOR command:

```

begin-setup
  declare-color-map
    light_blue = (193, 222, 229)
  end-declare
end-setup

begin-program
  alter-color-map name = 'light_blue' value = (193, 233, 230)

  print 'Yellow Submarine' ( )
    foreground = ('yellow')
    background = ('light_blue')

  get-color print-text-foreground = ($print-foreground)
  set-color print-text-foreground = ('purple')
  print 'Barney' (+1,1)
  set-color print-text-foreground = ($print-foreground)
end-program

```

See Also

DECLARE-COLOR-MAP, ALTER-COLOR-MAP, GET-COLOR

SET-GENERATIONS

Syntax

```
SET-GENERATIONS=(dimension,hierarchy,dimension,hierarchy,dimension,hierarchy,
...,...)
```

Description

Specifies dimension hierarchy for the previously declared dimension.

Returns the set of members in the dimension 'product' that are at the 5th generation in the dimension's hierarchy. (Returns all 'Brand Name' members (Generation Level 5) under the product hierarchy of 'all products.drink.alcoholic beverages.beer and wine.' This would increase the result set to a list of beers and wines.) Returns the set of members in the dimension 'time' that are at the 1st generation deep into the dimension. (Returns all 'Year' members (Generation Level 1) under the time hierarchy of '2004.Q1.2'. This reduces result set to '2004'.)

Example

The following example shows the SET-GENERATIONS command:

```
set-generations=('product',5,'time',1 )
```

SET-LEVELS

Syntax

```
Set-levels=(dimension, level, dimension, level,...,...)
```

Description

Extends the dimension hierarchy for the previously declared dimension.

Set-levels used with only the previous 'set-members' returns all members under the product hierarchy and the next two generations (Product SubCategory and Brand Name) for the product hierarchy of all products.drink.alcoholic beverages.beer and wine'. Set-levels used with the previous 'set-members' and 'set-generations' returns all members for generation levels 5 through 7 under the product hierarchy of all products.drink.alcoholic beverages.beer and wine.'

Example

The following example shows the SET-LEVELS command:

```
set-levels=('product',2 )
```

SET-MEMBERS

Syntax

```
set-members=(dimension, hierarchy, dimension, hierarchy,..., ...)
```

Description

Returns the set of members in a dimension, level, or hierarchy for which name is specified by a string.

Example

Returns the set of members in the dimension 'product' at the specific hierarchy of 'all products', at a specific level of 'drink', at a specific level of 'alcoholic beverages', at a specific level of 'beer and wine'. Returns the set of members in the dimension 'time' at the specific hierarchy of '2004', at the specific level of 'Q1', at the specific level of '2'.

```
set-members=('product','all products.drink.alcoholic beverages.beer⇒  
and wine','time','2004.Q1.2')
```

SHOW

Syntax

```
SHOW[cursor_position]  
[CLEAR-SCREEN | CS | CLEAR-LINE | CL] [any_lit | _var | _col]  
[EDITedit_mask | NUMBER | MONEY | DATE] [BOLD] [BLINK]  
[UNDERLINE] [REVERSE] [NORMAL] [BEEP] [NOLINE] . . .
```

Description

Displays one or more variables or literals on the screen. In addition, cursor control is supported for ANSI terminals.

Any number of variables and screen positions can be used in a single command. Each one is processed in sequence.

Screen locations can be indicated by either fixed or relative positions in the format (A,B), where A is the line and B is the column on the screen. A, B, or both can also be numeric variables. Relative positions depend on where the previous SHOW command ended. If the line was advanced, the screen cursor is usually immediately to the right of the previously displayed value and one line down.

Fixed or relative cursor positioning can be used only within the boundaries of the terminal screen. Scrolling off the screen using relative positioning, for example (+1,1), is not supported. Instead, use a SHOW command without any cursor position when you want to scroll. Also, you cannot mix SHOW and DISPLAY commands while referencing relative cursor positions.

The SHOW command does not advance to the next line if a cursor location (...), CLEAR-SCREEN, CLEAR-LINE, or BEEP is used. (A SHOW command without any of these arguments automatically advances the line.) To add a line advance, add (+1,1) to the end of the line or use an extra empty SHOW command.

Only ANSI terminals are supported for cursor control, screen blanking, line blanking, and display characteristics.

Dates can be contained in a date variable or column, or a string literal, column, or variable. When the program displays a date variable or column without an edit mask, the date is displayed according to the following rules:

- For DATETIME columns and SQR DATE variables, SQR uses the format specified by the SQR_DB_DATE_FORMAT setting.
If this has not been set, SQR uses the first database-dependent format as listed in the Default Database Formats table.
- For DATE columns, SQR uses the format specified by the SQR_DB_DATE_ONLY_FORMAT setting.
If this has not been set, SQR uses the format listed in the Default Database Formats table.
- For TIME columns, SQR uses the format specified by the SQR_DB_TIME_ONLY_FORMAT setting.
If this has not been set, SQR uses the format as listed in the TIME Column Formats table.

When displaying a date in a string literal, column, or variable using EDIT or DATE, the string must be in the format specified by the SQR_DB_DATE_FORMAT setting, one of the database-dependent formats as listed in the Default Database Formats table, or the database-independent format 'YYYYMMDD[HH24[MI[SS[NNNNN]]]]].

Parameters

<i>Parameter</i>	<i>Description</i>
cursor_position	Specifies the position on the screen to begin the display.
{ CLEAR-SCREEN CS }	Clears the screen and sets the cursor position to (1,1).
{ CLEAR-LINE CL }	Clears a line from the current cursor position to the end of the line.
{ any_lit _var _col }	Specifies the information to be displayed.
EDIT	Shows variables under an edit mask. If the mask contains spaces, enclose it in single quotes. For additional information regarding edit masks, see the PRINT command.
NUMBER	Indicates that any_lit _var _col is to be formatted using the NUMBER-EDIT-MASK from the current locale. (See the ALTER-LOCALE command.) This option is not valid for date variables.
MONEY	Indicates that any_lit _var _col is to be formatted using the MONEY-EDIT-MASK from the current locale. (See the ALTER-LOCALE command.) This option is not valid for date variables.
DATE	Indicates that any_lit _var _col is to be formatted using the DATE-EDIT-MASK from the current locale. (See the ALTER-LOCALE command.) This option is not valid for numeric variables. If DATE-EDIT-MASK has not been specified, the date is displayed using the default format for that database (see the Default Database Formats table).
BOLD, BLINK, UNDERLINE, and REVERSE	Changes the display of characters on terminals that support those characteristics. Some terminals support two or more characteristics at the same time for the same text. To turn all special display characteristics off, use NORMAL.
NORMAL	Turns off all special display characteristics set with BOLD, BLINK, UNDERLINE, and REVERSE.
BEEP	Causes the terminal to beep.
NOLINE	Inhibits a line advance.

Example

The following program segments illustrate the various features of the SHOW command:

```

!
! Show a string using an edit mask
!
let $ssn = '123456789'
show $ssn edit xxx-xx-xxxx

```

Produces the following output:

```
123-45-6789
```

```

!
! Show a number using an edit mask
!
show 1234567.89 edit 999,999,999.99

```

Produces the following output:

```
1,234,567.89
```

```

!
! Show a number using the default edit mask
!
show 123.78

```

Produces the following output:

```
123.780000
```

```

!
! Show a number using the locale default numeric edit mask
!
alter-locale number-edit-mask = '99,999,999.99'
show 123456.78 number

```

Produces the following output:

```
123,456.78
```

```

!
! Show a number using the locale default money edit mask
!
alter-locale money-edit-mask = '$$,$$$,$$8.99'
show 123456.78 money

```

Produces the following output:

```
$123,456.78
```

```

!
! Show a date column using the locale default date edit mask
!
begin-select
dcol
  from tables
end-select
alter-locale date-edit-mask = 'DD-Mon-YYYY'
show &dcol date

```

Produces the following output:

```
01-Jan-2004
```

```
!
! Show two values on the same line
!
show 'Hello' ' World'
```

Produces the following output:

```
Hello World
```

```
!
! Show two values on the same line with editing of the values
!
let #taxes = 123456.78
show 'You owe ' #taxes money ' in back taxes.'
```

Produces the following output:

```
You owe $123,456.78 in back taxes.
```

The following program illustrates the usage of additional options of the SHOW command. Only terminals that support the ANSI escape characters can use the cursor control, screen blanking, line blanking, and display attributes.

```
begin-program
!
! Produces a menu for the user to select from
!
show clear-screen
      (3,30) bold 'Accounting Reports for XYZ Company' normal
      (+2,10) '1. Monthly Details of Accounts'
      (+1,10) '2. Monthly Summary'
      (+1,10) '3. Quarterly Details of Accounts'
      (+1,10) '4. Quarterly Summary'

!
! Show a line of text and numerics combined
!
show (+2,1)
      'The price is ' #price edit 999.99

      '      Total = ' #total edit 99999.99

!
! Put an error message on a particular line
!
show (24,1) clear-line 'Error in SQL. Please try again.' beep
end-program
```

See Also

The LET command for information about copying, editing, or converting fields.

The EDIT parameter of the PRINT command for a description of the edit masks.

The ALTER-LOCALE command for a description of the arguments NUMBER-EDIT-MASK, MONEY-EDIT-MASK, and DATE-EDIT-MASK.

DISPLAY

STOP

Syntax

```
STOP [QUIET]
```

Description

The STOP command halts SQR and executes a ROLLBACK command (not in Sybase, Microsoft SQL Server, or Informix). All report page buffers are flushed, if they contain data; however, no headers or footers are printed and the AFTER-PAGE and AFTER-REPORT procedures are not executed.

STOP is useful in testing.

Parameters

<i>Parameter</i>	<i>Description</i>
QUIET	Causes the report to complete with the "SQR: End Of Run" message, instead of ending with an error message.

Example

The following example shows the STOP command:

```
if #error-status = 1
    rollback
    stop
else
    commit
    stop quiet
end-if
```

STRING

Syntax

```
STRING {src_any_lit|_var|_col}...BY {delim_txt_lit|_var|_col}
INTO dst_txt_var
```

Description

Concatenates a list of variables, columns, or literals into a single text variable. Each member of the list is separated by the specified delimiter string.

The destination string must not be included in the list of source strings.

Parameters

<i>Parameter</i>	<i>Description</i>
{src_any_lit _var _col}	<p>Specifies one or more fields to be concatenated, separated by the <i>delim_txt_lit _var _col</i> character or characters, and placed into the <i>dst_txt_var</i> variable.</p> <p>If the source is a date variable or column, it is converted to a string according to the following rules:</p> <p>For DATETIME columns and SQR DATE variables, SQR uses the format specified by the SQR_DB_DATE_FORMAT setting.</p> <p>If this has not been set, SQR uses the first database-dependent format as listed in the Default Database Formats table.</p> <p>For DATE columns, SQR uses the format specified by the SQR_DB_DATE_ONLY_FORMAT setting.</p> <p>If this has not been set, SQR uses the format listed in the Default Database Formats table.</p> <p>For TIME columns, SQR uses the format specified by the SQR_DB_TIME_ONLY_FORMAT setting.</p> <p>If this has not been set, SQR uses the format as listed in the TIME Column Formats table.</p>
{delim_txt_lit _var _col}	Specifies one or more characters to be used as separator characters between the source fields.
dst_txt_var	Specifies the destination field for the concatenated result.

Example

The following example shows the STRING command:

```
string  &name  &city  &state  &zip  by  ' - '  into  $show-info
      ! Result:  Sam Mann - New York - NY - 11287
string  &cust_num  &entry-date  &total  by  ','  into  $cust-data
      ! Result:  100014,12-MAR-04,127
      ! Use null delimiter.
string  &code1  &code2  &code3  by  ''  into  $codes123
      ! Result:  AGL
```

See Also

The UNSTRING command for additional information.

The "||" concatenation operator in the Operators table under the LET command.

SUBTRACT

Syntax

```
SUBTRACT {src_num_lit|_var|_col} FROM dst_num_var[ROUND=nn]
```

Description

Subtracts the first value from the second and moves the result into the second field.

When dealing with money-related values (dollars and cents), use decimal variables rather than float variables. Float variables are stored as double-precision floating-point numbers, and small inaccuracies can appear when you are subtracting many numbers in succession. These inaccuracies can appear due to the way floating point numbers are represented by different hardware and software implementations.

Parameters

<i>Parameter</i>	<i>Description</i>
{src_num_lit _var _col}	Is subtracted from the contents of <i>dst_num_var</i> .
dst_num_var	Contains the result after execution.
ROUND	Rounds the result to the specified number of digits to the right of the decimal point. For float variables this value can be from 0 (zero) to 15. For decimal variables, this value can be from 0 to the precision of the variable. For integer variables, this argument is not appropriate.

Example

The following example shows the SUBTRACT command:

```
subtract 1 from #total      ! #total - 1
subtract &discount from #price ! #price - &discount
```

See Also

The ADD command for more information.

The LET command for information about complex arithmetic expressions.

TOC-ENTRY

Syntax

```
TOC-ENTRY
TEXT={src_txt_lit|_var|_col}
[LEVEL={level_num_lit|_var|_col}]
```

Description

Enter the text in the table of contents at the desired level.

Parameters

<i>Parameter</i>	<i>Description</i>
TEXT	Specifies the text to be placed in the table of contents.
LEVEL	Specifies the level at which to place the text. If this argument is not specified, the value of the previous level is used.

Example

The following example shows the TOC-ENTRY command:

```
toc-entry text = &heading
toc-entry text = &caption level=2
```

See Also

The DECLARE-TOC command.

UNSTRING

Syntax

```
UNSTRING {{src_txt_lit|_var|_col}}|{{src_date_var|_col}}
BY {delim_txt_lit|_var|_col}
INTO dst_txt_var...
```

Description

Copies portions of a string into one or more text variables.

Each substring is located using the specified delimiter. The source string must not be included in the list of destination strings.

If more destination strings than substrings are found in the source strings, the extra destination strings are each set to an empty string.

If more substrings are found in the source string than in the destination strings, the extra substrings are not processed. The programmer is responsible for ensuring that enough destination strings are specified.

If the source is a date variable or column, it is converted to a string according to the following rules:

- For DATETIME columns and SQR DATE variables, SQR uses the format specified by the SQR_DB_DATE_FORMAT setting.

If this has not been set, SQR uses the first database-dependent format as listed in the Default Database Formats table.

- For DATE columns, SQR uses the format specified by the SQR_DB_DATE_ONLY_FORMAT setting.

If this has not been set, SQR uses the format listed in the Default Database Formats table.

- For TIME columns, SQR uses the format specified by the SQR_DB_TIME_ONLY_FORMAT setting.

If this has not been set, SQR uses the format as listed in the TIME Column Formats table.

Parameters

<i>Parameter</i>	<i>Description</i>
{src_txt_lit _var _col} {src_date_var _col}	Specifies the source field to be parsed.
delim_txt_lit _var _col	Specifies one or more characters to be used to delimit the fields within {src_txt_lit _var _col} {src_date_var _col}
dst_txt_var	Specifies one or more destination fields to receive the results.

Example

The following example shows the UNSTRING command:

```
unstring $show-info by ' - ' into $name $city $state $zip
unstring $cust-data by ',' into $cust_num $entry-date $total
```

See Also

STRING, EXTRACT

The substr and instr functions in the Miscellaneous Functions table under the LET command.

UPPERCASE

Syntax

```
UPPERCASE txt_var
```

Description

Converts a string variable to uppercase.

Parameters

<i>Parameter</i>	<i>Description</i>
<code>txt_var</code>	Specifies a text variable to be converted to uppercase.

Example

The following example shows the UPPERCASE command:

```
input  $state  'Enter state abbreviation'  
uppercase $state      ! Force uppercase.
```

See Also

The upper function in the Miscellaneous Functions table under the LET command.

USE

Syntax

```
USE database
```

Description

Uses the named database rather than the default database associated with your username. (Sybase and Microsoft SQL Server only.)

Use USE in the SETUP section only. When used, it must appear at the top of your report, before any queries are defined.

To reference more than one database in a program, specify secondary databases explicitly. For example:

```
from sqdb.sqr.customers
```

You cannot issue the Sybase or Microsoft SQL Server USE command from within an SQL paragraph.

Parameters

<i>Parameter</i>	<i>Description</i>
database	Specifies the name of the database to use.

Example

The following example shows the USE command:

```
begin-setup
  use pubs
end-setup
```

See Also

[Chapter 4, "Invoking SQR Execute," page 269](#)

USE-COLUMN

Syntax

```
USE-COLUMN {column_number_int_lit|_var|_col}
```

Description

Sets the current column.

The column must be previously defined with the COLUMNS command.

To stop printing within columns, use a column number of 0 (zero). Printing returns to normal; however, the columns remain defined for subsequent NEXT-COLUMN or USE-COLUMN commands.

Parameters

<i>Parameter</i>	<i>Description</i>
{ <i>column_number_int_lit</i> _var _col} }	Specifies the number of the defined column (not the location on the page). For example, if five columns are defined, the <i>column_number_int_lit</i> _var _col can be 1 to 5.

Example

The following example shows the USE-COLUMN command:

```
use-column 3      ! Print total in 3rd column.
print #total    () 999,999
use-column 0      ! End of column printing.
```

USE-PRINTER-TYPE

Syntax

USE-PRINTER-TYPE *printer-type*

Description

Sets the printer type to be used for the current report.

The USE-PRINTER-TYPE command sets or alters the printer type to be used for the current report. The USE-PRINTER-TYPE command must appear before the first output is written to that report. If output has already been written to the report file, the USE-PRINTER-TYPE command is ignored.

Parameters

<i>Parameter</i>	<i>Description</i>
printer-type	Specifies the printer type to be used for the current report. See DECLARE-PRINTER for valid types.

Example

The following example shows the USE-PRINTER-TYPE command:

```
use-report customer_orders
use-printer-type PostScript
print (1, 1) 'Customer Name: '
print () $customer_name
```

See Also

DECLARE-PRINTER, DECLARE-REPORT, USE-REPORT

USE-PROCEDURE

Syntax

```
USE-PROCEDURE
[FOR-REPORTS=(report_name1[,report_namei]...)]
[BEFORE-REPORT=procedure_name[(arg1[,argi]...)]]
[AFTER-REPORT=procedure_name[(arg1[,argi]...)]]
[BEFORE-PAGE=procedure_name[(arg1[,argi]...)]]
[AFTER-PAGE=procedure_name[(arg1[,argi]...)]]
```

Description

Changes the procedure usage.

The USE-PROCEDURE command must be issued in the PROGRAM or PROCEDURE sections of an SQR program. USE-PROCEDURE is a runtime command; its compile-time equivalent is DECLARE-PROCEDURE. You can use the command as often as required to change to the necessary procedures required by the reports. If you issue multiple USE-PROCEDURE commands, each remains in effect for that report until altered by another USE-PROCEDURE command for that report. In this way, you can use one to change common procedures for ALL reports and others to change unique procedures for individual reports. The referenced procedures can accept arguments.

If no FOR-REPORTS is specified, ALL is assumed. Initially, the default for each of the four procedure types is NONE. If a procedure is defined in one DECLARE-PROCEDURE for a report, that procedure is used unless NONE is specified.

You can change the BEFORE-REPORT only before the first output is written to that report, because that causes the BEFORE-REPORT procedure to be executed.

Parameters

<i>Parameter</i>	<i>Description</i>
FOR-REPORTS	Specifies the reports that are to use these procedures. This argument is required only for a program with multiple reports. If you are writing a program that produces a single report, you can ignore this argument.
BEFORE-REPORT	Specifies a procedure to execute at the time of execution of the first command, which causes output to be generated. You can use the command, for example, to create a report heading.
AFTER-REPORT	Specifies a procedure to execute just before the report file is closed at the end of the report. This argument can be used to print totals or other closing summary information. If no report was generated, the procedure does not execute.
BEFORE-PAGE	Specifies a procedure to execute at the beginning of every page, just before the first output command for the page. It can be used, for example, to set up page totals.

Parameter	Description
AFTER-PAGE	Specifies a procedure to execute just before each page is written to the file. This argument can be used, for example, to display page totals. You can also specify arguments to be passed to the procedure. Arguments can be any variable, column, or literal.

Example

The following example shows the USE-PROCEDURE command:

```

use-procedure           ! These procedures will
for-reports=(all)       ! be used by all reports
before-report=report_heading
after-report=report_footing
use-procedure           ! These procedures will
for-reports=(customer)  ! be used by the customer
before-page=page_setup  ! report
after-page=page_totals
use-procedure           ! The after-report
for-reports=(summary)   ! procedure will be
after-report=none        ! disabled for the
                        ! summary report

```

See Also

DECLARE-PROCEDURE

USE-REPORT

Syntax

```
USE-REPORT {report_name_lit/_var/_col}
```

Description

For programs with multiple reports, enables the user to switch between reports.

The USE-REPORT command specifies to which report files the subsequent report output is to be written. An application can contain several USE-REPORT statements to control several reports.

You must specify the report name and report characteristics in a DECLARE-REPORT paragraph and in the associated DECLARE-LAYOUT and DECLARE- PRINTER paragraphs.

Parameters

<i>Parameter</i>	<i>Description</i>
{report_name_lit _var _col}	Specifies the report to become the current report. All subsequent PRINT and PRINT-DIRECT statements are written to this report until the next USE-REPORT is encountered.

Example

The following example shows the USE-REPORT command:

```
use-report customer_orders
use-printer-type PostScript
print (1, 1) 'Customer Name: '
print () $customer_name
```

See Also

DECLARE-REPORT, DECLARE-LAYOUT, DECLARE-PRINTER, USE-PRINTER-TYPE

WHILE

Syntax

```
WHILE logical_expression
```

The general format of a WHILE command is:

```
WHILE logical_expression
SQR_commands...
[BREAK]
SQR_commands...
END-WHILE
```

Description

Begins a **WHILE ... END-WHILE** loop.

The WHILE loop continues until the condition being tested is FALSE.

An expression returning 0 (zero) is considered FALSE; an expression returning nonzero is TRUE.

BREAK causes an immediate exit of the WHILE loop; SQR continues with the command immediately following END-WHILE.

WHILE commands can be nested to any level and can include or be included within IF and EVALUATE commands.

Parameters

<i>Parameter</i>	<i>Description</i>
logical_expression	A valid logical expression. See the LET command for a description of logical expressions.

Example

This example shows an IF nested within a WHILE:

```
while #count < 50
  do get_statistics
    if #stat_count = 100
      break      ! Exit WHILE loop.
    end-if
    add 1 to #count
end-while
```

You can use single numeric variables in your expression to make your program more readable, for example, when using flags:

```
move 1 to #have_data
...
while #have_data
  ...processing...
end-while
```

This example sets up an infinite loop:

```
while 1
  ...processing...
  if ...
    break      ! Exit loop
  end-if
end-while
```

Any complex expression can be used in the WHILE command, as shown in this example:

```
while #count < 100 and (not #end-file or isnull(&state))
  ...
end-while
```

See Also

The LET command for a description of expressions.

WRITE

Syntax

```
WRITE {filenum_lit|_var|_col} FROM
{{{txt_lit|_var|_col}|{date_var|_col}|num_col}
[:len_int_lit]}|{num_lit|_var:len_int_lit}}...
[STATUS=status_num_var]
```

Description

Writes a record to a file from data stored in variables, columns, or literals.

The file must already be opened for writing.

If length is specified, the variable is either truncated at that length or padded with spaces to that length. If length is not specified (for string variables or database columns), the current length of the variable is used.

When you are writing numeric variables, the length argument is required. Only 1-byte, 2-byte, or 4-byte binary integers are written. Floating point values are not supported directly in the WRITE command. However, you can first convert floating point numbers to strings and then write the string.

When you are writing binary data, you must open the file using the FIXED or FIXED-NOLF qualifiers. The file is not portable across platforms because binary numbers are represented differently.

When writing a date variable or column, the date is converted to a string according to the following rules:

- For DATETIME columns and SQR DATE variables, SQR uses the format specified by the SQR_DB_DATE_FORMAT setting.
If this has not been set, SQR uses the first database-dependent format as listed in the Default Database Formats table.
- For DATE columns, SQR uses the format specified by the SQR_DB_DATE_ONLY_FORMAT setting.
If this has not been set, SQR uses the format listed in the DATE Column Formats table.
- For TIME columns, SQR uses the format specified by the SQR_DB_TIME_ONLY_FORMAT setting.
If this has not been set, SQR uses the format as listed in the TIME Column Formats table.

Text literals take the length of the literal.

Files opened for writing are treated as having variable-length records. If you need a fixed-length record, specify a length for each variable written to the file.

The total length of the variables and literals being written must not be greater (but can be less) than the record length specified when the file was opened. Records are not padded, but are written with the total length of all variables in the WRITE command.

If STATUS is specified, SQR returns 0 if the write is successful; otherwise, it returns the value of *errno*, which is system-dependent.

Parameters

Parameter	Description
filenum_lit _var _col	Specifies the number assigned in the OPEN command to the file to be written.
{ {txt_lit _var _col} {date_var _col} num_col} [:len_int_lit]} {num_lit _var:len_int_lit}	Specifies one or more variables to be written. <i>len_int_lit</i> specifies the length of each field of data.
STATUS	Specifies an optional variable into which a write status is returned.

Example

The following example shows the WRITE command:

```
write 5 from $name:20 $city:15 $state:2
write 17 from $company ' - ' $city ' - ' $state ' ' $zip
write #j2 from #rate:2 #amount:4 #quantity:1
move #total to $tot 99999.99           ! Convert floating point to string.
write 1 from $tot
let $date1 = datenow()                 ! Put the current date and time
                                       ! into DATE variable
write 3 from $date1:20
```

See Also

OPEN, CLOSE, READ

Chapter 3

Generating HTML Output

This chapter describes the procedures that enable SQR for PeopleSoft to generate HTML output. You can publish the output on an internet, intranet, or extranet website. An SQR program without HTML procedures has limited HTML capabilities; therefore, adding HTML procedures to the SQR program enhances the appearance of the HTML output.

This chapter discusses:

- HTML general-purpose procedures.
- HTML heading procedures.
- HTML highlighting procedures.
- HTML hypertext link procedures.
- HTML list procedures.
- HTML table procedures.

See Also

Enterprise PeopleTools 8.50 PeopleBook: SQR for PeopleSoft Developers, "Generating and Publishing HTML from an SQR Program"

HTML General Purpose Procedures

The following table lists the HTML general-purpose procedures:

<i>Procedure</i>	<i>Description</i>
html_br	<p>Produces the specified number of line breaks in a paragraph by using the HTML
 tag. The paragraph continues onto the next line.</p> <p>Syntax:</p> <pre>html_br(number count, string attributes)</pre> <p>count: The number of HTML
 tags that are inserted.</p> <p>attributes: The HTML attributes that are incorporated inside the HTML
 tag.</p> <p>Example:</p> <pre>print 'Here is some text' () do html_br(3, '') print 'Here is some text three lines down' ()</pre>
html_center	<p>Marks the start of text to be centered on the HTML document by using the HTML <CENTER> tag. You can also accomplish this by using the SQR print statement with CENTER specified in the code.</p> <p>Syntax:</p> <pre>html_center()</pre> <p>attributes: The HTML attributes that are incorporated inside the HTML <CENTER> tag.</p> <p>Example:</p> <pre>do html_center('') print 'Here is some centered text' () do html_center_end</pre>
html_center_end	<p>Marks the end of text that was previously specified as centered.</p> <p>Syntax:</p> <pre>html_center_end</pre>
html_hr	<p>Produces a horizontal divider between sections of text by using the HTML <HR> tag.</p> <p>Syntax:</p> <pre>html_hr(string attributes)</pre> <p>attributes: The HTML attributes that are incorporated inside the HTML <HR> tag.</p> <p>Example:</p> <pre>print 'Here is some text' () do html_hr('') print 'Text after a horizontal divider' ()</pre>

<i>Procedure</i>	<i>Description</i>
html_img	<p>Inserts an image by using the HTML tag. You can also do this by using the PRINT-IMAGE command; however, the html_img procedure enables you to specify the full set of available HTML attributes.</p> <p>Syntax:</p> <pre>html_img(string attributes)</pre> <p>attributes: The HTML attributes that are incorporated inside the HTML tag. Some common attributes include:</p> <ul style="list-style-type: none"> src: The URL of the image to be inserted. Example: src=/images/abc.gif height: The height of the image in pixels. Example: height=200 width: The width of the image in pixels. Example: width=400 <p>Example:</p> <pre>do html_img('src="/images/stop.gif"')</pre>
html_nobr	<p>Marks the start of text that cannot be wrapped by using the HTML <NOBR> tag.</p> <p>Syntax:</p> <pre>html_nobr</pre> <p>Example:</p> <pre>do html_nobr('') print 'Long line of text that shouldn't wrap' => () do html_nobr_end</pre>
html_nobr_end	<p>Marks the end of text that cannot be wrapped by using the HTML </NOBR> tag.</p> <p>Syntax:</p> <pre>html_nobr_end</pre>
html_on	<p>Activates the HTML procedures. Call this procedure at the start of an SQR program; otherwise, the HTML procedures are not activated. After the HTML procedures are activated, format the appearance of the web page by using the various HTML procedures.</p> <p>Syntax:</p> <pre>html_on</pre> <p>Example:</p> <pre>do html_on</pre>

Procedure	Description
html_p	<p>Marks the start of a new paragraph by using the HTML <P> tag.</p> <p>Syntax:</p> <pre>html_p(string attributes)</pre> <p>attributes: Defines the HTML attributes that are incorporated inside the HTML <P> tag. A common attribute is align = left right center, which specifies the alignment of the paragraph.</p> <p>Example:</p> <pre>do html_p('ALIGN=RIGHT') print 'Right aligned text' (1,1) do html_p_end print 'Normally aligned text' (+1,1)</pre>
html_p_end	<p>Marks the end of a paragraph by using the HTML </P> tag. The end of a paragraph is typically implied and the procedure is technically not needed; however, it's good practice to specify it for completeness.</p> <p>Syntax:</p> <pre>html_p_end</pre>
html_set_body_attributes	<p>Specifies the attributes that are incorporated into the HTML <BODY> tag. Call this procedure at the start of the SQL program.</p> <p>Syntax:</p> <pre>html_set_body_attributes(string attributes)</pre> <p>attributes: Defines the HTML attributes that are incorporated inside the HTML <BODY> tag. Some common attributes include:</p> <ul style="list-style-type: none"> background: Specifies the image to display in the background of the web page. Example: background=/images/logo.gif bgcolor=#rrggbb: Specifies the background color of the web page. Example: bgcolor=#80FFF <p>Example:</p> <pre>do html_set_body_attributes('BACKGROUND="> /images/x.gif"')</pre>

<i>Procedure</i>	<i>Description</i>
html_set_head_tags	<p>Specifies the tags that are incorporated between the HTML <HEAD> and </HEAD> tags. These are empty by default. One common tag to set is the HTML <TITLE> tag, which specifies the title to display for the web page. Call this procedure at the start of the SQL program.</p> <p>Syntax:</p> <pre>html_set_head_tags(string attributes)</pre> <p>attributes: Defines the HTML attributes that are incorporated between the HTML <HEAD> and </HEAD> tags.</p> <p>Example:</p> <pre>do html_set_head_tags('<TITLE>My Report<=> /TITLE>')</pre>

HTML Heading Procedures

The following table lists the HTML heading procedures:

<i>Procedure</i>	<i>Description</i>
html_h1	<p>Marks the start of text for heading level one by using the HTML <H1> tag. This heading text appears more prominently than heading level two.</p> <p>Syntax:</p> <pre>html_h1(string attributes)</pre> <p>attributes: Defines the HTML attributes that are incorporated inside the HTML <H1> tag.</p> <p>Example:</p> <pre>do html_h1('') print 'This is a heading' () do html_h1_end</pre>
html_h1_end	<p>Marks the end of text for heading level one by using the HTML </H1> tag.</p> <p>Syntax:</p> <pre>html_h1_end</pre>

<i>Procedure</i>	<i>Description</i>
html_h2	<p>Marks the start of text for heading level two by using the HTML <H2> tag. This heading text appears less prominently than heading level one and more prominently than heading level three.</p> <p>Syntax:</p> <pre>html_h2(string attributes)</pre> <p>attributes: Defines the HTML attributes that are incorporated inside the HTML <H2> tag.</p> <p>Example:</p> <pre>do html_h2('') print 'This is a heading' () do html_h2_end</pre>
html_h2_end	<p>Marks the end of text for heading level two by using the HTML </H2> tag.</p> <p>Syntax:</p> <pre>html_h2_end</pre>
html_h3	<p>Marks the start of text for heading level three by using the HTML <H3> tag. This heading text appears less prominently than heading level two and more prominently than heading level four.</p> <p>Note. This heading level is the default value.</p> <p>Syntax:</p> <pre>html_h3(string attributes)</pre> <p>attributes: Defines the HTML attributes that are incorporated inside the HTML <H3> tag.</p>
html_h3_end	<p>Marks the end of text for heading level three by using the HTML </H3> tag.</p> <p>Syntax:</p> <pre>html_h3_end</pre>
html_h4	<p>Marks the start of text for heading level four by using the HTML <H4> tag. This heading text appears less prominently than heading level three and more prominently than heading level five.</p> <p>Syntax:</p> <pre>html_h4(string attributes)</pre> <p>attributes: Defines the HTML attributes that are incorporated inside the HTML <H4> tag.</p>

<i>Procedure</i>	<i>Description</i>
html_h4_end	Marks the end of text for heading level four by using the HTML <code></H4></code> tag. Syntax: <code>html_h4_end</code>
html_h5	Marks the start of text for heading level five by using the HTML <code><H5></code> tag. This heading text appears less prominently than heading level four and more prominently than heading level six. Syntax: <code>html_h5(string attributes)</code> attributes: Defines the HTML attributes that are incorporated inside the HTML <code><H5></code> tag.
html_h5_end	Marks the end of text for heading level five by using the HTML <code></H5></code> tag. Syntax: <code>html_h5_end</code>
html_h6	Marks the start of text for heading level six by using the HTML <code><H6></code> tag. This heading text appears less prominently than heading level five. Syntax: <code>html_h6(string attributes)</code> attributes: Defines the HTML attributes that are incorporated inside the HTML <code><H6></code> tag.
html_h6_end	Marks the end of text for heading level six by using the HTML <code></H6></code> tag. Syntax: <code>html_h6_end</code>

HTML Highlighting Procedures

The following table lists the HTML highlighting procedures:

<i>Procedure</i>	<i>Description</i>
html_blink	<p>Marks the start of blinking style text by using the HTML <BLINK> tag.</p> <p>Syntax:</p> <pre>html_blink(string attributes)</pre> <p>attributes: Defines the HTML attributes that are incorporated inside the HTML <BLINK> tag.</p> <p>Example:</p> <pre>do html_blink('') print 'This is blinking text' () do html_blink_end</pre>
html_blink_end	<p>Marks the end of blinking style by using the HTML </BLINK> tag.</p> <p>Syntax:</p> <pre>html_blink_end</pre>
html_cite	<p>Marks the start of text in citation style by using the HTML <CITE> tag.</p> <p>Syntax:</p> <pre>html_cite(string attributes)</pre> <p>attributes: Defines the HTML attributes that are incorporated inside the HTML <CITE> tag.</p> <p>Example:</p> <pre>do html_cite('') print 'This is a citation' () do html_cite_end</pre>
html_cite_end	<p>Marks the end of text in citation style by using the HTML </CITE> tag.</p> <p>Syntax:</p> <pre>html_cite_end</pre>
html_code	<p>Marks the start of text in code style by using the HTML <CODE> tag.</p> <p>Syntax:</p> <pre>html_code(string attributes)</pre> <p>attributes: Defines the HTML attributes that are incorporated inside the HTML <CODE> tag.</p> <p>Example:</p> <pre>do html_code('') print 'Here is code style text' () do html_code_end</pre>

<i>Procedure</i>	<i>Description</i>
html_code_end	Marks the end of text in code style by using the HTML <code></CODE></code> tag. Syntax: <code>html_code_end</code>
html_kbd	Marks the start of text in keyboard input style by using the HTML <code><KBD></code> tag. Syntax: <code>html_kbd(string attributes)</code> attributes: Defines the HTML attributes that are incorporated inside the HTML <code><KBD></code> tag. Example: <pre>do html_kbd('') print 'Here is keyboard style text' () do html_kbd_end</pre>
html_kbd_end	Marks the end of text in keyboard style by using the HTML <code></KBD></code> tag. Syntax: <code>html_kbd_end</code>
html_samp	Marks the start of text in sample style by using the HTML <code><SAMP></code> tag. Syntax: <code>html_samp(string attributes)</code> attributes: Defines the HTML attributes that are incorporated inside the HTML <code><SAMP></code> tag. Example: <pre>do html_samp('') print 'Here is sample style text' () do html_samp_end</pre>
html_samp_end	Marks the end of text in sample style by using the HTML <code></SAMP></code> tag. Syntax: <code>html_samp_end</code>

<i>Procedure</i>	<i>Description</i>
html_strike	<p>Marks the start of text in strikethrough style by using the HTML <STRIKE> tag.</p> <p>Syntax:</p> <pre>html_strike(string attributes)</pre> <p>attributes: Defines the HTML attributes that are incorporated inside the HTML <STRIKE> tag.</p> <p>Example:</p> <pre>do html_strike('') print 'Here is strikethrough text' () do html_strike_end</pre>
html_strike_end	<p>Marks the end of text in strikethrough style by using the HTML </STRIKE> tag.</p> <p>Syntax:</p> <pre>html_strike_end</pre>
html_sub	<p>Marks the start of text in subscript style by using the HTML <SUB> tag.</p> <p>Syntax:</p> <pre>html_sub(string attributes)</pre> <p>attributes: Defines the HTML attributes that are incorporated inside the HTML <SUB> tag.</p> <p>Example:</p> <pre>print 'Here is' () do html_sub('') print 'subscript text' () do html_sub_end</pre>
html_sub_end	<p>Marks the end of text in subscript style by using the HTML </SUB> tag.</p> <p>Syntax:</p> <pre>html_sub_end</pre>
html_sup	<p>Marks the start of text in superscript style by using the HTML <SUP> tag.</p> <p>Syntax:</p> <pre>html_sup(string attributes)</pre> <p>attributes: Defines the HTML attributes that are incorporated inside the HTML <SUP> tag.</p> <p>Example:</p> <pre>print 'Here is' () do html_sup('') print 'superscript text' () do html_sup_end</pre>

<i>Procedure</i>	<i>Description</i>
html_sup_end	<p>Marks the end of text in superscript style by using the HTML </SUP> tag.</p> <p>Syntax:</p> <pre>html_sup_end</pre>

HTML Hypertext Link Procedures

The following table lists the HTML hypertext link procedures:

<i>Procedure</i>	<i>Description</i>
html_a	<p>Marks the start of a hypertext link by using the HTML <A> tag. When the user clicks the area with the hypertext link, the web browser switches to the specified HTML document.</p> <p>Syntax:</p> <pre>html_a(string attributes)</pre> <p>attributes: Defines the HTML attributes that are incorporated inside the HTML <A> tag. At a minimum, you should define the href attribute, which specifies the URL of an HTML document.</p> <p>Some common attributes include:</p> <ul style="list-style-type: none"> • href: Indicates where the hypertext link points. Example: href=home.html • name: Indicates an anchor to which a hypertext link can point. Example: name=marker1 <p>Example:</p> <p>The anchor is positioned at the top of the document. The first hypertext link points to the HTML document named otherdoc.html. The second hypertext link points to the anchor named TOP.</p> <pre>do html_a('NAME=TOP') do html_a_end print 'At the top of document' () do html_br(20, '') do html_a('HREF=otherdoc.html') print 'Go to other document' () do html_a_end do html_p('') do html_a('HREF=#TOP') print 'Go to top of document' () do html_a_end</pre>

<i>Procedure</i>	<i>Description</i>
html_a_end	Marks the end of a hypertext link by using the HTML <code></code> tag. Syntax: <code>html_a_end</code>

HTML List Procedures

The following table lists the HTML list procedures:

<i>Procedure</i>	<i>Description</i>
html_dd	Marks the start of a definition in a definition list by using the HTML <code><DD></code> tag. Syntax: <code>html_dd(string attributes)</code> attributes: Defines the HTML attributes that are incorporated inside the HTML <code><DD></code> tag.
html_dd_end	Marks the end of a definition in a definition list by using the HTML <code></DD></code> tag. The end of a definition in a definition list is typically implied and not needed; however, it's good form to specify it for completeness. Syntax: <code>html_dd_end</code>
html_dir	Marks the start of a directory list by using the HTML <code><DIR></code> tag. Syntax: <code>html_dir(string attributes)</code> attributes: Defines the HTML attributes that are incorporated inside the HTML <code><DIR></code> tag. Example: <pre>do html_dir('') do html_li('') print 'First item' () do html_li('') print 'Second item' () do html_li('') print 'Last item' () do html_dir_end</pre>

Procedure	Description
html_dir_end	Marks the end of a directory list by using the HTML </DIR> tag. Syntax: html_dir_end
html_dl	Marks the start of a definition list by using the HTML <DL> tag. A definition list displays a list of terms and definitions. The term displays above and to the left of the definition. Use the html_dt procedure to display a term. Use the html_dd procedure to display a definition. Syntax: html_dl(string attributes) attributes: Defines the HTML attributes that are incorporated inside the HTML <DL> tag. Example: <pre>do html_dl('') do html_dt('') print 'A Daisy' () do html_dd('') print 'A sweet and innocent flower.' () do html_dt('') print 'A Rose' () do html_dd('') print 'A very passionate flower.' () do html_dl_end</pre>
html_dl_end	Marks the end of a definition list by using the HTML </DL> tag. Syntax: html_dl_end
html_dt	Marks the start of a term in a definition list by using the HTML <DT> tag. Syntax: html_dt(string attributes) attributes: Defines the HTML attributes that are incorporated inside the HTML <DT> tag.
html_dt_end	Marks the end of a term in a definition list by using the HTML </DT> tag. Syntax: html_dt_end

<i>Procedure</i>	<i>Description</i>
html_li	<p>Marks the start of a list item by using the HTML tag.</p> <p>Syntax:</p> <pre>html_li(string attributes)</pre> <p>attributes: Defines the HTML attributes that are incorporated inside the HTML tag.</p>
html_li_end	<p>Marks the end of a list item by using the HTML tag. The end of a list item is typically implied and not needed; however, you should specify it for completeness.</p> <p>Syntax:</p> <pre>html_li_end</pre>
html_menu	<p>Marks the start of a menu by using the HTML <MENU> tag. Use the html_li procedure to identify each item in the list.</p> <p>Syntax:</p> <pre>html_menu(string attributes)</pre> <p>attributes: Defines the HTML attributes that are incorporated inside the HTML <MENU> tag.</p> <p>Example:</p> <pre>do html_menu('') do html_li('') print 'First item' () do html_li('') print 'Second item' () do html_li('') print 'Last item' () do html_menu_end</pre>
html_menu_end	<p>Marks the end of a menu by using the HTML </MENU> tag.</p> <p>Syntax:</p> <pre>html_menu_end</pre>

<i>Procedure</i>	<i>Description</i>
html_ol	<p>Marks the start of an ordered list by using the HTML tag. Each item in the list typically appears indented to the right with a number to the left. Use the html_li procedure to identify each item in the list.</p> <p>Syntax:</p> <pre>html_ol(string attributes)</pre> <p>attributes: Defines the HTML attributes that are incorporated inside the HTML tag.</p> <p>Example:</p> <pre>do html_ol('') do html_li('') print 'First item' () do html_li('') print 'Second item' () do html_li('') print 'Last item' () do html_ol_end</pre>
html_ol_end	<p>Marks the end of an ordered list by using the HTML tag.</p> <p>Syntax:</p> <pre>html_ol_end</pre>
html_ul	<p>Marks the start of an unordered list by using the HTML tag. Each item in the list typically appears indented to the right with a bullet to the left. Use the html_li procedure to identify each item in the list.</p> <p>Syntax:</p> <pre>html_ul(string attributes)</pre> <p>attributes: defines the HTML attributes that are incorporated inside the HTML tag.</p> <p>Example:</p> <pre>do html_ul('') do html_li('') print 'First item' () do html_li('') print 'Second item' () do html_li('') print 'Last item' () do html_ul_end</pre>
html_ul_end	<p>Marks the end of an unordered list by using the HTML tag.</p> <p>Syntax:</p> <pre>html_ul_end</pre>

HTML Table Procedures

The following table lists the HTML table procedures:

<i>Procedure</i>	<i>Description</i>
html_caption	<p>Marks the start of a table caption by using the HTML <CAPTION> tag.</p> <p>Syntax:</p> <pre>html_caption(string attributes)</pre> <p>attributes: Defines the HTML attributes that are incorporated inside the HTML <CAPTION> tag.</p>
html_caption_end	<p>Marks the end of a table caption by using the HTML </CAPTION> tag. The end of a table caption is typically implied and not needed; however, you should specify it for completeness.</p> <p>Syntax:</p> <pre>html_caption_end</pre>

<i>Procedure</i>	<i>Description</i>
html_table	<p>Marks the start of a table by using the HTML <TABLE> tag.</p> <p>Syntax:</p> <pre>html_table(string attributes)</pre> <p>attributes: Defines the HTML attributes that are incorporated inside the HTML <TABLE> tag.</p> <p>Some common attributes include:</p> <ul style="list-style-type: none"> • border: Specifies that a border appears around each cell of the table. • width: Specifies the width of the entire table in pixels. • cols: Specifies the number of columns in the table. <p>Example: COLS=4</p> <p>Example: Displaying database records in a tabular format. The html_caption_end, html_tr_end, html_td_end, and html_th_end procedures are used for completeness; however, they are typically implied and not needed.</p> <pre>! start the table & display the column headings do html_table('border') do html_caption('') print 'Customer Records' (1,1) do html_caption_end do html_tr('') do html_th('') print 'Cust No' (+1,1) do html_th_end do html_th('') print 'Name" (,10) do html_th_end do html_tr_end ! display each record begin-select do html_tr('') do html_td('') cust_num (1,1,6) edit 099999 do html_td_end do html_td('') name (1,10,25) do html_td_end do html_tr_end next-listing skiplines=1 need=1 from customers end-select ! end the table do html_table_end</pre>
html_table_end	<p>Marks the end of a table by using the HTML </TABLE> tag.</p> <p>Syntax:</p> <pre>html_table_end</pre>

<i>Procedure</i>	<i>Description</i>
html_td	<p>Marks the start of a new column in a table row by using the HTML <code><TD></code> tag. This specifies that the text that follows is displayed in the column.</p> <p>Syntax:</p> <pre>html_td(string attributes)</pre> <p>attributes: Defines the HTML attributes that are incorporated inside the HTML <code><TD></code> tag.</p>
html_td_end	<p>Marks the end of a column in a table by using the HTML <code></TD></code> tag. The end of a column is typically implied and not needed; however, you should specify it for completeness.</p> <p>Syntax:</p> <pre>html_td_end</pre>
html_th	<p>Marks the start of a new column header in a table row by using the HTML <code><TH></code> tag. This specifies that the text that follows is displayed as the header of the column.</p> <p>Syntax:</p> <pre>html_th(string attributes)</pre> <p>attributes: Defines the HTML attributes that are incorporated inside the HTML <code><TH></code> tag.</p>
html_th_end	<p>Marks the end of a column header in a table by using the HTML <code></TH></code> tag. The end of a column header is typically implied and not needed; however, you should specify it for completeness.</p> <p>Syntax:</p> <pre>html_th_end</pre>
html_tr	<p>Marks the start of a new row in a table by using the HTML <code><TR></code> tag.</p> <p>Syntax:</p> <pre>html_tr(string attributes)</pre> <p>attributes: Defines the HTML attributes that are incorporated inside the HTML <code><TR></code> tag.</p>
html_tr_end	<p>Marks the end of a row by using the HTML <code></TR></code> tag. The end of a row in a table is typically implied and not needed; however, you should specify it for completeness.</p> <p>Syntax:</p> <pre>html_tr_end</pre>

Chapter 4

Invoking SQR Execute

SQR Execute is a runtime program that enables you to run a previously compiled SQR program.

This chapter discusses how to:

- Run SQR Execute
- Use SQR Execute flags

Running SQR Execute

To run SQR Execute, enter the following command. (If you are using Microsoft Windows, invoke SQRWT rather than SQR.)

```
SQR [program][connectivity][flags...][args...][@file...]
```

Using SQR Execute Flags

The following table describes the SQR Execute command-line flags. See the SQR command-line arguments section for information about program, connectivity, args, and @file.

See [Chapter 1, "Understanding SQR for PeopleSoft," SQR Command-Line Arguments, page 5.](#)

<i>Flag</i>	<i>Description</i>
-A	Appends the output to an existing output file of the same name. If the file does not exist, it creates a new one. This flag is useful when you want to run the same report more than once, but you want to create only one output file.
-C	(Microsoft Windows) Specifies that the Cancel dialog box appears while the program is running so that you can easily terminate the program.

Flag	Description
-BURST:{xx}	<p>Specifies the type of bursting to be performed.</p> <p>-BURST:T generates the table of contents file only.</p> <p>-BURST:S generates the report output according to the symbolic table of contents entries that are set in the program with the TOC- ENTRY command's LEVEL argument. In -BURST:S[{l}], <i>l</i> is the level at which to burst. The -BURST:S setting is equivalent to -BURST:S1.</p> <p>-BURST:P generates the report output by report page numbers. In -BURST:P[{l} , {s} [, {s}] ...] , <i>l</i> is the number of logical report pages that each .htm file contains and <i>s</i> is the page selection: <i>n</i>, <i>n-</i>, <i>m</i>, <i>-m</i>, or <i>n-</i>. The -BURST:P setting is equivalent to -BURST:P0,1- when using -PRINTER:HT or -BURST:P1 when using -PRINTER:EH.</p> <p>Note. -BURST:P and -BURST:S require -PRINTER:EH or -PRINTER:HT. The page range selection feature of -BURST:P requires -PRINTER:HT. -BURST:T requires -PRINTER:HT.</p>
-CB	(Microsoft Windows) Forces the communication box.
-Dnn	(Non-Microsoft Windows) Causes SQR to display the report output on the terminal at the same time that it is being written to the output file. The <i>nn</i> is the maximum number of lines to display before pausing. If you don't enter a number after -D, the display scrolls continuously. The printer type must be LP; otherwise SQR does not display any output. If the program is producing more than one report, SQR displays only the first report.
-DBdatabase	(Sybase) Causes the SQR program to use the specified database, overriding any USE command in the SQR program.
-E[file]	Directs error messages to the named file or to the default file, program.err. If no errors occur, no file is created.

<i>Flag</i>	<i>Description</i>
-EH_APPLETS:dir	Specifies the directory location of the enhanced HTML applets. The default directory for these applets is IMAGES. Note. The -EH flags in this table are applicable only when either the -PRINTER:EH or -PRINTER:EP flag is specified.
-EH_BQD	Generates a {report}.bqd file from the report data. Also associates a BQD icon with {report}.bqd in the navigation bar.
-EH_BQD:file	Associates the BQD icon with the specified file.
-EH_BROWSER:xx	Specifies the target browser. When set to ALL, SQR automatically determines which browser is being used, invokes a browser-specific file, and generates HTML that is designed for that browser. When set to BASIC, SQR generates HTML that is suitable for all browsers. When set to IE, SQR generates HTML that is designed for Microsoft Internet Explorer. When set to NETSCAPE, SQR generates HTML that is designed for Netscape.
-EH_CSV	Generates a {report}.csv file from the report data.
-EH_CSV:file	Associates the CSV icon with the specified file.
-EH_CSVONLY	Creates a CSV file, but does not create an HTML file.
-EH_FULLHTML:xx	Switches between HTML 3.0 and HTML 3.2. When set to TRUE, SQR generates HTML 3.2. When set to FALSE, SQR generates HTML 3.0
-EH_Icons:dir	Specifies the directory in for the referenced icons.
-EH_LANGUAGE:xx	Sets the language that is used for the HTML navigation bar. You can specify English, French, German, Portuguese, Spanish, Japanese, Simplified Chinese, and Korean.

Flag	Description
-EH_PDF	Associates a PDF icon with {report}.pdf in the navigation bar.
-EH_Scale:{nn}	Sets the scaling factor from 50 to 200.
-F[file directory]	<p>Overrides the default output file name, program.lis. The default action places the program.lis file in the same directory as the program.sqr file.</p> <p>To use the current directory, specify -F without an argument.</p> <p>To change the name of the output file, specify -F with the new name. If the new name does not specify a directory, the file is created in the current directory.</p> <p>The output file is not created until data is actually printed on the page. If no data is printed, no output file is created.</p> <p>Specify the file name and directory for different operating systems:</p> <ul style="list-style-type: none"> • UNIX/Linux • MVS
-GPRINT=YES NO	(MVS) -GPRINT=YES causes ANSI control characters written to the first column of each record of the SQR output file.
-ID	Displays the copyright banner on the console.
-KEEP	In addition to .lis files, creates an .spf file for each report that the program generates.
-NOLIS	Prevents the creation of .lis files. Instead, .spf files are created.
-O[file]	Directs log messages to the specified file or to programt.log, if no file is specified. By default, the sqr.log file is used in the current working directory.
-P	(MVS) Suppresses printer control characters from column 1.
-PB	(Informix) Causes column data to retain trailing blanks.

Flag	Description
-PRINTER:xx	<p>Causes printer type <i>xx</i> to be used when creating output files:</p> <ul style="list-style-type: none"> • EH • EP • HP • HT • LP • PD • PS • WP <p>Types LP, HP, and PS produce files with the .lis extension.</p> <p>Types EH and HT produce .htm file output.</p> <p>Type HT produces files in HTML version 2.0 with the report content inside the <PRE></PRE> tags.</p> <p>Type EH produces reports in which content is fully formatted with HTML version 3.0 or 3.2 tags.</p> <p>On Microsoft Windows systems, the WP extension sends the output to the default Microsoft Windows printer. To specify a nondefault Microsoft Windows printer, enter -PRINTER:WP:{printer name}. The {printer name} is the name that is assigned to the printer. For example, to send output to a Microsoft Windows printer named NewPrinter, use -PRINTER:WP:NewPrinter. If the printer name has spaces, enclose the entire argument in quotes. If you also want to create an .spf file, use -KEEP.</p>
-S	<p>Requests that the status of all cursors be displayed at the end of the report run. Status includes the text of each SQL statement, the number of times that each was compiled and run, and the total number of rows selected.</p>
-Tnn	<p>Specifies that you want to test the report for <i>nn</i> pages. SQR ignores all Order By clauses in Select statements to save time during testing. If the program is producing more than one report, SQR stops after the specified number of pages that are defined for the first report have been printed.</p>

Flag	Description
-T{B Z BZ ZB}	<p>(IBM MVS and DB2):</p> <ul style="list-style-type: none"> -TB prevents SQR from removing trailing blanks from database character columns. -TZ prevents SQR from removing trailing zeros from the decimal portion of numeric columns. -TBZ or -TZB prevents both. <p>(Windows/DB2, Sybase CT-Lib, and ODBC): -TB trims trailing blanks from database character columns.</p> <p>Note. The -TB flag only has an effect if SQR is connecting to either a DB2, Sybase CT LIB, or ODBC (MSS) database. Confusingly, the behavior of the -TB command-line flag varies depending on your platform. If you are using one of the above mentioned databases and are running SQR on z/OS, the -TB flag will behave as follows:</p> <p>If you do not use the -TB flag, trailing blanks are trimmed.</p> <p>If you do use the -TB flag, trailing blanks are not trimmed.</p> <p>If you are running SQR on any other platform, the behavior of -TB is the opposite:</p> <p>If you do not use the -TB flag, trailing blanks are not trimmed.</p> <p>If you do use the -TB flag, trailing blanks are trimmed.</p>
-Vserver	(Sybase) Uses the named server.
-XB	(Non-Microsoft Windows) Suppresses the SQR banner and the <i>SQR.... End of Run</i> message.
-XCB	(Microsoft Windows) Does not use the communication box.

<i>Flag</i>	<i>Description</i>
-XL	<p>Prevents SQR from signing in to the database. Programs that are run in this mode cannot contain any SQL statements.</p> <p>-XL enables you to run SQR without accessing the database. You still must supply at least an empty slash (/) on the command line as a placeholder for the connectivity information. For example: <code>sqr myprog / -xl</code></p> <p>Some database files must be available for SQR to run whether SQR signs in to the database or not.</p> <p>See information about your particular operating system and database to determine which files you need.</p>
-XMB	(Microsoft Windows) Disables the error message display so that a program can be run without interruption by error message boxes. Error messages are sent only to an .err file. See the -E flag for more information.
-XNAV	Prevents SQR from creating the navigation bar in .htm files that are generated with -PRINTER:HT. This occurs when only a single .htm file is produced. Multiple .htm files that are generated from a single report always contain the navigation bar.
-XP	(Sybase DBLib) Prevents SQR from creating temporary stored procedures. See BEGIN-SELECT for more information.
-XTB	Preserves the trailing blanks in an .lis file at the end of a line.
-XTOC	Prevents SQR from generating the table of contents for the report. SQR ignores this flag when -PRINTER:EH or -PRINTER:HT is also specified.
-ZIF{file}	Sets the full path and name of the SQR initialization file, <code>sqr.ini</code> .
-ZIV	Invokes the SPF Viewer after generating the <code>program.spf</code> file. This flag implicitly invokes the -KEEP flag to create <code>program.spf</code> . In the case of multiple output files, only the first report file is passed to the viewer.

<i>Flag</i>	<i>Description</i>
-ZMF{ file }	Specifies the full path and name of the SQR error message file, sqrerr.dat.

See Also

Chapter 1, "Understanding SQR for PeopleSoft," SQR Command-Line Arguments, page 5

Chapter 5

Using SQR Print

This chapter provides an overview of SQR Print and discusses how to:

- Generate output from the command line.
- Use SQR Print command-line flags.
- Generate output in Microsoft Windows.

Understanding SQR Print

SQR Print enables you to create printer-specific reports for any of the file types that are supported by SQR. SQR Print converts portable printer-independent files (spfs) into printer-specific files. SQR and SQRT (or SQRWT, for Windows) create .spf files when you use the -KEEP and -NOLIS command-line flags.

Generating Output from the Command Line

To begin running SQR Print, enter the following command. (If you are in Microsoft Windows, invoke SQRWP rather than SQRP.)

```
SQRP [spf-file] [flags...]
```

The following table describes *spf-file* and *flags*.

SQR Print writes an .lis file with the same name as the .spf file, but with the lis extension. You can override this name with the -F command-line flag.

The -PRINTER command-line flag specifies the printer type. SQR offers several options:

- Line printer
- HP LaserJet
- PostScript
- HTML
- Enhanced HTML
- Adobe PDF
- Enhanced HTML and Adobe PDF

If the report contains graphics and you select a line printer, SQR Print ignores graphic elements (such as lines, boxes, and charts) and prints only text.

Using SQR Print Command-Line Flags

The following table describes the SQR Print command-line flags:

<i>Flag</i>	<i>Description</i>
-A	Appends the output to an existing output file of the same name. If the file does not exist, it creates a new one. This flag is useful when you want to run the same report more than once, but you want to create only one output file.
-BURST:{xx}	<p>Specifies the type of bursting to be performed.</p> <p>-BURST:T generates the table of contents file only.</p> <p>-BURST:S generates the report output according to the symbolic table of contents entries that are set in the program with the TOC-ENTRY command's LEVEL argument. In -BURST:S[{l}], <i>l</i> is the level at which to burst. The -BURST:S setting is equivalent to -BURST:S1.</p> <p>-BURST:P generates the report output by report page numbers. In -BURST:P[{l} , {s} [, {s}] ...] , <i>l</i> is the number of logical report pages that each .htm file contains and <i>s</i> is the page selection: {<i>n</i>}, {<i>n</i>}-{<i>m</i>}, -{<i>m</i>}, or <i>n</i>-. The -BURST:P setting is equivalent to -BURST:P0,1- when using -PRINTER:HT or -BURST:P1 when using -PRINTER:EH.</p> <p>Note. -BURST:P and -BURST:S require -PRINTER:EH or -PRINTER:HT. The page range selection feature of -BURST:P requires -PRINTER:HT.</p> <p>BURST:T requires -PRINTER:HT.</p>
-Dnn	(Non-Microsoft Windows) Displays the report output on the terminal at the same time that it is being written to the output file. The <i>nn</i> is the maximum number of lines is displayed before pausing. If you don't enter a number after -D, the displayed output scrolls continuously. The printer type must be LP or the no output is displayed. If the program is producing more than one report, only the first report is displayed.

<i>Flag</i>	<i>Description</i>
-E[file]	<p>Directs error messages to the named file or to the default file, program.err. If no errors occur, no file is created.</p> <p>Note. The following -EH flags are applicable only when either the -PRINTER:EH or -PRINTER:EP flag is specified.</p>
-EH_APPLETS:dir	Specifies the directory location of the enhanced HTML applets. The default directory for these applets is IMAGES.
-EH_BQD	Generates a {report}.bqd file from the report data. Also associates a BQD icon with {report}.bqd in the navigation bar.
-EH_BQD:file	Associates the BQD icon with the specified file.
-EH_BROWSER:xx	<p>Specifies the target browser. When set to ALL, SQR automatically determines which browser is being used, invokes a browser-specific file, and generates HTML that is designed for that browser.</p> <p>When set to BASIC, SQR generates HTML that is suitable for all browsers.</p> <p>When set to IE, SQR generates HTML that is designed for Microsoft Internet Explorer.</p> <p>When set to NETSCAPE, SQR generates HTML that is designed for Netscape.</p>
-EH_CSV	Generates a {report}.csv file from the report data.
-EH_CSV:file	Associates the CSV icon with the specified file.
-EH_CSVONLY	Creates a CSV file, but does not create an HTML file.
-EH_FULLHTML:xx	<p>Switches between HTML 3.0 and HTML 3.2. When set to TRUE, SQR generates HTML 3.2.</p> <p>When set to FALSE, SQR generates HTML 3.0</p>
-EH_Icons:dir	Specifies the directory in which the HTML should find the referenced icons.

Flag	Description
-EH_LANGUAGE:xx	Sets the language that is used for the HTML navigation bar. You can specify English, French, German, Portuguese, Spanish, Japanese, Simplified Chinese, or Korean.
-EH_PDF	Associates a PDF icon with {report}.pdf in the navigation bar.
-EH_Scale:{nn}	Sets the scaling factor from 50 to 200.
-F[file directory]	<p>Overrides the default output file name, program.lis. The default action places program.lis in the same directory as the program.sqr file.</p> <p>To use the current directory, specify -F without an argument.</p> <p>To change the name of the output file, specify -F with the new name. If the new name does not specify a directory, the file is created in the current directory. The output file is not created until data is actually printed on the page. If no data is printed, no output file is created.</p> <p>Specify these file names and directories for different operating systems:</p> <ul style="list-style-type: none"> • UNIX/Linux • MVS
-ID	Displays the copyright banner on the console.
-O[file]	Directs log messages to the specified file or to programt.log if no file is specified. By default, the sqr.log file is used in the current working directory.
-P	(IBM MVS) Suppresses printer control characters from column 1.

Flag	Description
-PRINTER:xx	<p>Causes SQR to use printer type <i>xx</i> when creating output files:</p> <ul style="list-style-type: none"> • EH • EP • HP • HT • LP • PD <p>Types LP, HP, and PS produce files with the .lis extension.</p> <p>Types EH and HT produce .htm file output.</p> <p>On Microsoft Windows systems, the WP extension sends the output to the default Microsoft Windows printer. To specify a nondefault Microsoft Windows printer, enter -PRINTER:WP:{printer name}. The {printer name} is the name that is assigned to the printer. For example, to send output to a Microsoft Windows printer named NewPrinter, use -PRINTER:WP:NewPrinter. If your printer name has spaces, enclose the entire argument in quotes. If you also want to create an .spf file, use -KEEP.</p>
-XB	(Non-Microsoft Windows) Suppresses the SQR banner and the <i>SQR.... End of Run</i> message.
-XNAV	Prevents SQR from creating the navigation bar in .htm files that are generated with -PRINTER:HT. This occurs when only a single .htm file is produced. Multiple .htm files that are generated from a single report always contain the navigation bar.
-XTB	Preserves the trailing blanks in a .lis file at the end of a line.
-XTOC	Prevents SQR from generating the table of contents for the report. This flag is ignored when -PRINTER:EH or -PRINTER:HT is also specified.
-ZIF{file}	Sets the full path and name of the SQR initialization file, sqr.ini.

<i>Flag</i>	<i>Description</i>
-ZMF{ file }	Specifies the full path and name of the SQR error message file, sqr-err.dat.

Generating Output in Microsoft Windows

In Microsoft Windows, SQR Print's graphical user interface enables you to generate output from the Print dialog box. In addition to the previously mentioned SQR Print output options, you can also select a Microsoft Windows printer. This spools the SQR output to your default Microsoft Windows printer or print server.

To generate output in Microsoft Windows:

1. Select File, Print.

The Print dialog box appears.

2. Under Generate output for, select the option next to the type of output that you want.
3. Specify a file path.
4. Select the Print to file check box.
5. Click OK.

Chapter 6

Avoiding Older SQR Commands

This chapter provides an overview of older SQR commands and discusses how to use them.

Understanding Older SQR Commands

Avoid incorporating the commands covered in this chapter in your SQR code. Even though they are technically supported in this release, they do not interact well with the current SQR lexicon and may cause unpredictable results. SQR may not support these commands in future releases, so you should remove these commands from your code as soon as it is feasible.

If your code still contains older SQR commands, refer to this table as you replace them with their updated alternatives:

<i>Old Commands</i>	<i>Alternative to Use Instead</i>
BEGIN-REPORT (END-REPORT)	BEGIN-PROGRAM (END-PROGRAM)
DATE-TIME	datenow function
DECLARE PRINTER	DECLARE-PRINTER
DECLARE PROCEDURE	DECLARE-PROCEDURE
DOLLAR-SYMBOL	ALTER-LOCALE
GRAPHIC FONT	ALTER-PRINTER
MONEY-SYMBOL	ALTER-LOCALE
NO-FORMFEED	DECLARE-LAYOUT
PAGE-SIZE	DECLARE-LAYOUT

<i>Old Commands</i>	<i>Alternative to Use Instead</i>
PRINTER-DEINIT	DECLARE-PRINTER
PRINTER-INIT	DECLARE-PRINTER
PRINT ... CODE	PRINT ... CODE-PRINTER

Note. Two older commands, DECLARE PRINTER and DECLARE PROCEDURE, do not contain hyphens. The new commands, DECLARE-PRINTER and DECLARE-PROCEDURE, contain hyphens.

Using Older SQR Commands

This section discusses each of the older SQR commands.

BEGIN-REPORT

You should no longer use this command because it may be discontinued in a future release. To use the newer SQR functionality, use BEGIN-PROGRAM.

Syntax

Use this syntax:

```
BEGIN-REPORT
```

Description

Begins a report.

After processing the commands in the SETUP section, SQR starts running the program at the BEGIN-REPORT section. The PROGRAM section typically contains a list of DO commands, though you can also use other commands. This is the only required section in an SQR program.

Example

For example:

```
begin-report
  do startup
  do main
  do finish
end-report
```

DATE-TIME

You should no longer use this command because it may be discontinued in a future release. To use the newer SQR functionality, use the `datetime` function in the `LET` command.

Syntax

Use this syntax:

```
DATE-TIME position [date_format[col_var]]
```

Description

Retrieves the current date and time from the local machine (or from the database for Oracle and some IBM DB2 platforms) and places it in the output file at the specified position or into a column variable.

If *col_var* is specified, *date_format* must be supplied and the current date and time is retrieved each time this command is run. Otherwise, the date is retrieved only at the program start, and the same date and time is printed each time.

If *date_format* is not specified, the date is returned in the default format for that database. The following table provides the default date-time formats for SQR-supported databases:

Database	Default Date-Time Format
Oracle	DD-Mon-YYYY HH:MI PM
Informix	YYYY-MM-DD HH:MI YYYY-MM-DD HH:MI:SS.NNN
IBM DB2	YYYY-MM-DD-HH:MI YYYY-MM-DD- HH:MI:SS.NNNNNN
Sybase	DD-MON-YYYY HH:MI

For some databases, there are two default formats. The first format prints the date-time, as in the following example:

```
date-time (+1,1)
```

The second format retrieves the date-time into a column variable:

```
date-time () ' ' &date1
```

For those databases with only one default format, that format is always used in either of these cases.

See the table showing miscellaneous functions under the `LET` command for information about the valid edit mask format codes.

Parameters

position	Specifies the position for printing the date.
date_format	Represents a string literal containing the date format mask.
col_var	Places the retrieved date-time into a column variable rather than in the output file.

Example

For example:

```
date-time (1,50) MM/DD/YY
date-time (1,1) 'Day Mon DD, YYYY'
date-time () HH:MI &time
date-time (+1,70) 'MON DD YYYY HH24:MI' &datetime
date-time (#i, #j) 'YYYY-MM-DD' &date1
```

See the \$current-date reserved and datenow functions that are described in the table showing miscellaneous functions under the LET command.

See ALTER-LOCALE

DECLARE PRINTER

You should no longer use this command because it may be discontinued in a future release. To use the newer SQR functionality, use DECLARE-LAYOUT and DECLARE-PRINTER.

Syntax

Use this syntax:

```
DECLARE PRINTER
[TYPE=printer_type_lit]
[ORIENTATION=orientation_lit]
[LEFT-MARGIN=left_margin_num_lit]
[TOP-MARGIN=top_margin_num_lit]
[LINE-SIZE=line_size_num_lit]
[CHAR-SIZE=char_size_num_lit]
[LINES-INCH=lines_inch_int_lit]
[CHARS-INCH=chars_inch_num_lit]
[POINT-SIZE=point_size_num_lit]
[FONT-TYPE=font_type_txt_lit]
[SYMBOL-SET=symbol_set_id_lit]
[STARTUP-FILE=file_name_txt_lit]
[FONT=font_int_lit]
[BEFORE-BOLD=before_bold_string_txt_lit]
[AFTER-BOLD=after_bold_string_txt_lit]
```

Description

Specifies the printer type and sets printer characteristics.

Use the DECLARE PRINTER command either in the SETUP section or in the body of the report. Generally, you should use it in the SETUP section. However, if you do not know what type of printer you will be using until the report is run, or if you need to change some of the arguments depending on user selection, you can put several DECLARE PRINTER commands in the body of the report and run the one that you need.

The following arguments take effect only once, upon execution of the first PRINT command, and thereafter have no effect even if changed:

```

LINE-SIZE
CHAR-SIZE
LINES-INCH
CHARS-INCH
ORIENTATION

```

SQR maps its line and column positions on the page by using a grid that is determined by the LINE-SIZE and CHAR-SIZE (or LINES-INCH and CHARS-INCH) arguments. Each printed piece of text is placed on the page by means of this grid. Because the characters in proportional fonts vary in width, a word or string may be wider than the horizontal space that you have allotted, especially in words containing uppercase letters. To account for this behavior, you can either move the column position in the PRINT statement or indicate a larger CHAR-SIZE value in the DECLARE PRINTER command.

Arguments

The following table describes the arguments for the DECLARE PRINTER command:

Argument	Choice or Measure	Default Value	Description
TYPE	LINE-PRINTER, POSTSCRIPT, HPLASERJET	LINE-PRINTER	<p>SQR creates output that is specific to each printer.</p> <p>Line printer files generally contain ASCII characters and can be viewed by a text editor.</p> <p>PostScript files contain ASCII characters, but you need to know PostScript to understand what will appear on the printer.</p> <p>HP LaserJet files are binary files and cannot be edited or viewed.</p>
ORIENTATION	PORTRAIT, LANDSCAPE	PORTRAIT	<p>Portrait pages are printed vertically.</p> <p>Landscape pages are printed horizontally.</p> <p>Printing in landscape mode on HP LaserJet printers requires landscape fonts.</p>

<i>Argument</i>	<i>Choice or Measure</i>	<i>Default Value</i>	<i>Description</i>
LEFT-MARGIN	inches	0.5	<p>This argument does not apply to line printers.</p> <p>This is the amount of blank space to leave at the left side of the page.</p>
TOP-MARGIN	inches	0.5	<p>This argument does not apply to line printers.</p> <p>This is the amount of blank space to leave at the top of the page.</p>
LINE-SIZE	points	12	<p>This argument does not apply to line printers.</p> <p>This is the size of each SQR line on the page. There are 72 points per inch. If LINE-SIZE is not specified, it follows the value for POINT-SIZE, if specified. The default value of 12 points yields 6 lines per inch.</p>
CHAR-SIZE	points	7.2	<p>This argument does not apply to line printers.</p> <p>This is the size of each SQR horizontal character column on the page (for example, the distance between the locations [1,12] and [1,13]). If CHAR-SIZE is not specified and the point size is less than 8.6, CHAR-SIZE is set to 4.32, which yields 16.6 characters per inch. The default value of 7.2 yields 10 characters per inch.</p>
LINES-INCH	lines	6	<p>This argument does not apply to line printers.</p> <p>This is an alternate way of indicating the, in lines per inch rather than in points (as in LINE-SIZE).</p>

Argument	Choice or Measure	Default Value	Description
CHARS-INCH	characters	10	<p>This argument does not apply to line printers.</p> <p>This is an alternate way of indicating the width of each SQR character column, in characters per inch rather than in points (as in CHAR-SIZE).</p>
POINT-SIZE	points	12	<p>This argument does not apply to line printers.</p> <p>This is the beginning size of the selected font.</p>
FONT-TYPE	PROPORTIONAL, FIXED	Depends on the font	<p>This argument applies only to HP LaserJet printers and must be specified only for font types that are not listed as being available for HP LaserJet printers in SQR in the previous DECLARE-PRINTER section.</p>
SYMBOL-SET	HP defined sets	0U	<p>This argument applies only to HP LaserJet printers. The default value, 0U, is for the ASCII symbol set. Additional symbol sets exist.</p> <p><i>See HP LaserJet Technical Reference Manual.</i></p>
STARTUP-FILE	filename	POSTSCRI.STR	<p>This argument applies only to PostScript printers. Use this to specify an alternate startup file. Unless otherwise specified, the default startup file is located in the directory that is specified by the environment variable SQDIR.</p>

<i>Argument</i>	<i>Choice or Measure</i>	<i>Default Value</i>	<i>Description</i>
FONT	font_number	3	<p>This is the font number of the typeface to use. For HP LaserJet printers, this is the typeface value as defined by Hewlett-Packard.</p> <p><i>See HP LaserJet Technical Reference Manual.</i></p> <p>For PostScript printers, SQR supplies a list of fonts and arbitrary font number assignments in the postscri.str file. The font numbers are the same as those for HP LaserJet printers, wherever possible, so that you can use the same font number for reports to be printed on both types of printers. You can modify the font list in postscri.str to add or delete fonts. Read the postscri.str file for instructions. See the fonts that are listed as being available for HP LaserJet printers in SQR in the previous DECLARE-PRINTER section. See also the table that lists the fonts that are available in the SQR postscri.str file, also in the DECLARE-PRINTER section.</p>

<i>Argument</i>	<i>Choice or Measure</i>	<i>Default Value</i>	<i>Description</i>
BEFORE-BOLD	any string	None	<p>The BEFORE-BOLD and AFTER- BOLD arguments are for line printers only. They specify the character string to turn bold on and off. If the string contains blank characters, enclose it in single quotation marks. To specify nonprintable characters, such as ESC, enclose the decimal value within angle brackets as follows: BEFORE-BOLD=<27>[r ! Turn on bold AFTER-BOLD=<27>[u ! Turn it off</p> <p>These arguments work with the BOLD argument of the PRINT command.</p>
AFTER-BOLD	any string	None	See BEFORE-BOLD.

The font that you choose—in orientation, typeface, and point size—must be an internal font (available in a font cartridge) or downloaded to the printer.

For fonts that are not listed as being available for HP LaserJet printers in SQR in the "DECLARE-PRINTER" section in the "SQR Command Reference" chapter, you must indicate the font style by using the FONT-TYPE argument; otherwise, the correct typeface cannot be selected by the printer.

DECLARE PROCEDURE

You should no longer use this command because it may be discontinued in a future release. To use the newer SQR functionality, use DECLARE-PROCEDURE.

Syntax

Use this syntax:

```
DECLARE PROCEDURE
[ BEFORE-REPORT=procedure_name ]
[ AFTER-REPORT=procedure_name ]
[ BEFORE-PAGE=procedure_name ]
[ AFTER-PAGE=procedure_name ]
```

Description

Defines specific event procedures.

Use the **DECLARE PROCEDURE** command either in the **SETUP** section or in the body of the report. You can use the command as often as you like.

If you issue multiple **DECLARE PROCEDURE** commands, the last one takes precedence. In this way, you can turn procedures on and off while the report is running. The referenced procedures do not take any arguments; however, they may be local. In addition, they can print only into the body of the report; that is, they cannot print into the header and footer areas.

Parameters

BEFORE-REPORT	Specifies a procedure to run at the time of the first PRINT command. For example, you use this to create a report heading.
AFTER-REPORT	Specifies a procedure to run just before the report file is closed at the end of the report. Use this to print totals or other closing summary information. If no report was generated, the procedure does not run.
BEFORE-PAGE	Specifies a procedure to run at the beginning of every page, just before the first PRINT command for the page. For example, you use this to set up page totals.
AFTER-PAGE	Specifies a procedure to run just before each page is written to the file. For example, you use this to display page totals.

Example

For example:

```
declare procedure
  before-page=page_setup
  after-page=page_totals
```

DOLLAR-SYMBOL

You should no longer use this command because it may be discontinued in a future release. To use the newer SQR functionality, use **ALTER-LOCALE**.

Syntax

Use this syntax:

```
DOLLAR-SYMBOL  new_symbol
```

Description

Redefines the currency symbol within numeric edit masks.

The dollar sign (\$) is the default currency symbol for coding edit masks in the program that prints on report listings. **DOLLAR-SYMBOL** provides a way to change that symbol for both the edit mask and for printing.

To change the symbol that prints on the report, use MONEY-SYMBOL in the PROCEDURE section. Use DOLLAR-SYMBOL and MONEY-SYMBOL together to configure SQR programs and the reports that they produce.

Use this command only in the SETUP section.

Note. The MONEY-SYMBOL command has the same effect as these options of the ALTER-LOCALE command: MONEY-SIGN and MONEY-SIGN-LOCATION=LEFT.

The following table lists the characters that DOLLAR-SYMBOL cannot take:

Type	Character
Numbers	0 8 9
Alphabetical	b e n r v B E N R V
Symbols	., - + ! * _ ` < > ()

Parameters

new_symbol Specifies a new, single character to be used in edit masks instead of the dollar sign (\$).

Example

For example:

```
begin-setup
  dollar-symbol £      ! Define £ as the currency symbol
end-setup
begin-procedure
...
print #amount ( ) edit £££,999.99
...
end-procedure
```

In the previous example, if you used the dollar sign in the edit mask after defining the dollar symbol as £, the following error message appears:

```
Bad numeric 'edit' format: $$$,999.99
```

See the ALTER-LOCALE command for a description of other locale-specific parameters.

GRAPHIC FONT

You should no longer use this command because it may be discontinued in a future release. To use the newer SQR functionality, use ALTER-PRINTER and DECLARE-PRINTER to set the FONT, FONT-TYPE, POINT-SIZE, and PITCH.

Syntax

Use this syntax:

```
GRAPHIC  ( )
FONT    { font_number_int_lit | _var }
[point_size_int_lit | _var] {1|0}
[pitch_int_lit | _var]]
```

Description

Changes a font.

Parameters

font_number	For HP LaserJet printers, the specified font must be installed in the printer. For PostScript printers, the font must be defined in the postscri.str file.
point_size	If <i>point_size</i> is omitted, the size from the most recent DECLARE-PRINTER or GRAPHIC FONT command is used.
{1 0}	This argument is for HP LaserJet printers only. It is needed only if you are using a font that SQR does recognize. (See the fonts that are listed as being available for HP LaserJet printers in SQR in the DECLARE-PRINTER section in the chapter "SQR Command Reference.") A 1 indicates a proportional font, and a 0 indicates a fixed-pitch font. The default is proportional.
pitch	If the specified font is fixed pitch, also indicate the pitch in characters per inch.

Example

For example:

```
graphic ( ) font 23 8.5      ! Century Schoolbook, 8.5 points
graphic ( ) font 6 12 0 10 ! Letter Gothic, 12 points,
                        ! fixed, 10 characters per inch
graphic ( ) font :#font_number :#point_size
```

See ALTER-PRINTER and DECLARE-PRINTER for information about setting and changing the FONT, FONT-TYPE, POINT-SIZE, and PITCH.

MONEY-SYMBOL

You should no longer use this command because it may be discontinued in a future release. To use the newer SQR functionality, use ALTER-LOCALE.

Syntax

Use this syntax:

```
MONEY-SYMBOL new_symbol
```

Description

Redefines the currency symbol to be printed.

To change the symbol that prints on the report, use the MONEY-SYMBOL in the PROCEDURE section. When the MONEY-SYMBOL is set, that value is used until the next MONEY-SYMBOL command runs.

Use DOLLAR-SYMBOL and MONEY-SYMBOL together to configure SQR programs and the reports that they produce.

To indicate a nonedit character, surround its decimal value with angle brackets (<>). See the table under the DOLLAR-SYMBOL command for characters that cannot be used with MONEY-SYMBOL.

Note. The MONEY-SYMBOL command has the same effect as the MONEY-SIGN and MONEY-SIGN-LOCATION=LEFT options of the ALTER- LOCALE command.

Parameters

new_symbol	Specifies a new, single character to replace the dollar sign (\$) or DOLLAR-SYMBOL character on the printed report.
-------------------	---

Example

For example:

```
begin-setup
  dollar-symbol £! Define £ as the
    ! currency symbol
end-setup
begin-procedure! If #Amount=1234.56
...
money-symbol £
print #Amount ( ) Edit £££,999.99    ! Prints as: £1,234.56
...
money-symbol $
print #Amount ( ) Edit £££,999.99    ! Prints as: $1,234.56
...
money-symbol
print #Amount ( ) Edit £££,999.99    ! Prints as: 1,234.56
...
end-procedure
```

See the DOLLAR-SYMBOL and ALTER-LOCALE commands.

NO-FORMFEED

You should no longer use this command because it may be discontinued in a future release. To use the newer SQR functionality, use the FORMFEED parameter of the DECLARE-LAYOUT command.

Syntax

Use this syntax:

```
NO-FORMFEED
```

Description

Prevents form-feed characters from being written to the output file.

NO-FORMFEED is useful for certain types of reports; for example, flat file output. It is used only in the SETUP section.

Do not write form-feed control characters directly into the output file between pages.

Example

For example:

```
begin-setup  
  no-formfeed  
end-setup
```

See Also

See the FORMFEED qualifier in DECLARE-LAYOUT.

PAGE-SIZE

You should no longer use this command because it may be discontinued in a future release. To use the newer SQR functionality, use the MAX-LINES and MAX-COLUMNS parameters of the DECLARE-LAYOUT command.

Syntax

Use this syntax:

```
PAGE-SIZE  page_depth_num_lit  page_width_num_lit
```

Description

Sets the page size.

If you are printing multiple reports, you must use the PAPER-SIZE parameter of the DECLARE-LAYOUT command.

This command is used in the SETUP section only.

Specify the page depth in lines and the page width in columns. An average report that is printed on 8 1/2 by 11 inch paper might have a page size of 60 lines by 80 columns. A 3-inch by 5-inch sales lead card might have a size of 18 by 50.

If the page size is not specified, the default of 62 lines by 132 columns is used.

For line printers, SQR stores one complete page in a buffer before writing the page to the output file when you issue a NEW-PAGE command or when a page overflow occurs.

You can define a page to be 1 line deep and 4,000 characters wide, which you can use for writing large flat files, perhaps for copying to magnetic tape. Each time a NEW-PAGE occurs, one record is written. Use the NO-FORMFEED command in the SETUP section to suppress form-feed characters between pages.

Use a page width that is at least one character larger than the rightmost position that will be written. This prevents unwanted wrapping when printing. When the last column position on a line is printed, the current position becomes the first position of the next line. This can cause confusion when using relative line positioning with the NEXT-LISTING command. Having a wider page than necessary does not waste any file space because SQR trims trailing blanks on each line before writing the report file.

Determine the size of the internal page buffer that stores a complete page in memory by multiplying the page depth by the width in the PAGE-SIZE command. For personal computers, the page buffer is limited to 64K bytes. On other computers, the page buffer is limited only by the amount of memory that is available.

Example

For example:

```
begin-setup
  page-size 57 132! 57 lines long by 132 columns wide
end-setup
```

See Also

See the PAPER-SIZE parameter of the DECLARE-LAYOUT command.

PRINT ... CODE

You should no longer use this command because it may be discontinued in a future release. To use the newer SQR functionality, use PRINT ... CODE-PRINTER.

If you use CODE, the sequence is assumed to be for the printer type that is specified in the DECLARE-REPORT or for the default printer, if none is specified.

Syntax

Use this syntax:

```
PRINT . . . CODE
```

Parameters

CODE CODE is a qualifier that may be discontinued in a future release. Use CODE-PRINTER instead.

PRINTER-DEINIT

You should no longer use this command because it may be discontinued in a future release. To use the newer SQR functionality, use the RESET-STRING parameter of the DECLARE-PRINTER command.

Syntax

Use this syntax:

```
PRINTER-DEINIT initialization_string
```

Description

Sends control or other characters to the printer at the end of a report.

Specify nondisplay characters by placing their decimal values inside angle brackets. For example, <27> is the ESC or escape character.

The PRINTER-DEINIT command is used only in the SETUP section and is designed for use with line printers. It has limited functionality with HP LaserJet and PostScript printers.

Example

For example:

```
begin-setup
  printer-deinit<27>[7J ! Reset the printer
end-setup
```

See Also

See the ENCODE command for another method of printing nondisplay characters. See the chr function in the table listing miscellaneous functions under the LET command in the chapter "SQR Command Reference."

PRINTER-INIT

You should no longer use this command because it may be discontinued in a future release. To use the newer SQR functionality, use the INIT-STRING parameter of the DECLARE-PRINTER command.

Syntax

Use this syntax:

```
PRINTER-INIT initialization_string
```

Description

Sends control or other characters to the printer at the beginning of a report.

Specify nondisplay characters by placing their decimal values inside angle brackets. For example, <27> is the ESC or escape character.

The PRINTER-INIT command is used only in the SETUP section and is designed for use with line printers. It has limited functionality with HP LaserJet and PostScript printers.

Example

For example:

```
begin-setup
  printer-init<27>[7J ! Set the printer
end-setup
```

See Also

See the ENCODE command for another method of printing nondisplay characters. See the chr function in the table listing miscellaneous functions under the LET command in the chapter "SQR Command Reference."

Chapter 7

Using the PSSQR.INI File and the PSSQR Command Line

This chapter discusses:

- pssqr.ini installation process
- Default settings section
- Processing-limits section
- Environment sections
- Locale section
- Fonts section
- HTML-Images section
- PDF Fonts section
- PDF Fonts: Exclusion Ranges Section
- TrueType Fonts Section
- Enhanced-HTML section
- Colors section
- pssqr.exe command-line options

Installing PSSQR.INI

The pssqr.ini file is the initialization file for SQR for PeopleSoft. SQR uses the settings and parameters in this file during the compile and execution phases.

The installation process installs a default initialization file called pssqr.ini, which is located in the SQR directory under the <PS_CFG_HOME> directory. The installation process also installs files named pssqr<language_cd>.ini, which are used to create language-specific configurations.

Microsoft Windows Platforms

On Microsoft Windows, SQR looks for the initialization file in the following locations, in order:

1. The file name specified by the `-ZIF{file}` command-line flag.
2. The directory in which the executable image resides.
3. The Microsoft Windows system directory.

z/OS

On z/OS, SQRINI is required during initialization. The dataset member &PSHLQ..SQRSRC(PSSQRINI) must be specified in the JCL to start the SQR process.

All Other Platforms

On all other platforms, SQR looks for the initialization file in the following locations, in order:

1. The file name specified by the `-ZIF{file}` command-line flag.
2. The current working directory.
3. The directory specified using the SQRDIR environment variable.

SQR automatically sets up SQRDIR.

You can make changes or additions to the pssqr.ini file.

This example shows the format of the pssqr.ini file:

```
; Comments are lines that start with a semicolon. The semicolon
; must be the first character of the line and therefore cannot be
; part of another line.
;
; Leading and trailing space characters are ignored. To preserve
; the space characters you must surround the value with either
; single (') or double (") quote characters. SQR will remove
; them when the entry is processed.
;
[Section_Name]
Entry = Value
.
.

[Another_Section_Name]
Entry = Value
.
.
```

Default Settings Section

This table describes the SQR default settings:

<i>Entry</i>	<i>Value</i>	<i>Description</i>
<code>ForceSpaceAfterComma=> {TRUE FALSE}</code>	TRUE FALSE	The default setting is FALSE. DB2 only: Forces a space after every comma not in a literal value. This is to support DECIMAL=COMMA setting on the OS390.
<code>ShowDBWarnings=> {TRUE FALSE}</code>	TRUE FALSE	The default setting is FALSE. DB2 only: If set to TRUE, SQR displays database warnings in the SQR output file.

Entry	Value	Description
AllowDateAsChar=> {TRUE FALSE}	TRUE FALSE	<p>The default setting is FALSE.</p> <p>By default, SQR produces an error when a dynamic column specification does not match the database definition of the column. That is, character equals character, date equals date, and numeric equals numeric.</p> <p>When this value is set to TRUE, SQR allows character to be equal to either character or date columns.</p> <p>When a date column is type cast to be a character, SQR creates the string according to the following rules:</p> <p>For DATETIME columns, SQR uses the format specified by the SQR_DB_DATE_FORMAT setting. If this has not been set, SQR uses the first database-dependent format as listed in the table showing default database formats in the Edit section under the PRINT command.</p> <p>For DATE columns, SQR uses the format specified by the SQR_DB_DATE_ONLY_FORMAT setting. If this has not been set, the format listed in the table showing default database formats in the Edit section under the PRINT command.</p> <p>For TIME columns, SQR uses the format specified by the SQR_DB_TIME_ONLY_FORMAT setting. If this has not been set, SQR uses the format as listed the table showing the TIME column formats in the Edit section under the PRINT command.</p> <p>In the following example, AllowDateAsChar=True. This allows \$Col1 to be either date or text.</p> <pre> Begin-Select [\$Col1] &col1=Text [\$Col2] &col2=Date [\$Col3] &col3=Number from MyTable End-Select </pre>

Entry	Value	Description
OUTPUT-FILE-MODE	LONG SHORT	<p>Specifies the file name convention used for HTML output. SHORT specifies DOS style (8.3) and LONG specifies UNIX/Linux style (non 8.3). The default is LONG. This setting is ignored on 16-bit platforms. The DECLARE-TOC and -BURST commands force Output-File-Mode = LONG.</p> <p>The following represents the file formats for UNIX/Linux, DOS, and Windows.</p> <p><i>SQR and SQRT</i>: {Program} is the name of the SQR/SQRT file without the extension.</p>
OUTPUT-FILE-MODE (continued)	LONG SHORT	<p>(Continued)</p> <p>For Output-File-Mode = SHORT, SQR-generated file names are limited to a DOS 8.3 format.</p> <ul style="list-style-type: none"> • Output file = {program}.lis for first, and {program}.lnn for multi-reports • SPF file = {program}.spf for first, and {program}.snn for multi-reports • PDF file = {program}.pdf for first; and {program}.pnn for multi-reports • HTM file = {program}.htm for "frame, and {program}.hbb for report bodies • GIF file={program}.gxx for all reports <p>bb ranges from 00 to 99 and represents the report number.</p> <p>nn ranges from 01 to 99 and represents the report number.</p> <p>xx ranges from 00 to ZZ and represents the graphic number.</p>

Entry	Value	Description
OUTPUT-FILE-MODE (continued)	LONG SHORT	<p>(Continued)</p> <p>For Output-File-Mode = LONG, SQR-generated file names are not constrained to a DOS 8.3 format. {output}={program} of first report and {program}_nn for multi-reports.</p> <ul style="list-style-type: none"> • Output file = {output}..lis • SPF file = {output}..spf • PDF file = {output}..pdf • GIF file = {output}_zz..spf • HTM files = {output}..htm, {output}_bb..htm, {output}_frm..htm, {output}_toc./htm, {output}_nav.htm <p>bb ranges from 01 to ZHJOZI and represents the bursted page group number.</p> <p>nn ranges from 01 to 99 and represents the report number.</p> <p>zz ranges from 00 to ZHJOZI and represent the graphic number.</p>
OUTPUT-FILE-MODE (continued)	LONG SHORT	<p>(Continued)</p> <p>SQRP: {file name} is the name of the SPF file without the extension</p> <p>For Output-File-Mode = SHORT, SQR-generated file names are limited to a DOS 8.3 format.</p> <ul style="list-style-type: none"> • Output file = {file name}.lis • GIF file = {file name}.gxx • PDF file = {file name}.pdf • HTM file = .htm and {file name}.h00 <p>xx ranges from 00 to ZZ and represents the graphic number.</p>

Entry	Value	Description
OUTPUT-FILE-MODE (continued)	LONG SHORT	<p>(Continued)</p> <p>For Output-File-Mode = LONG, SQR-generated file names are not limited to a DOS 8.3 format.</p> <ul style="list-style-type: none"> • Output file = {file name}.lis • PDF file = {file name}.pdf • GIF file = {file name}_zz.spf • HTM files = {file name}.htm, {file name}_bb.htm, {file name}_frm.HTM, {file name}_toc.htm {file name}_nav.htm <p>bb ranges from 01 to ZHJOZI and represents the bursted page group number.</p> <p>zz ranges from 00 to ZHJOZI and represents the graphic number.</p>
LOCALE	Name of a locale defined in the sqr.ini file or the name SYSTEM.	<p>Specifies the initial locale that SQR loads when the program starts. The value of SYSTEM is used to reference the default locale.</p> <p>See Chapter 2, "SQR Command Reference," ALTER-LOCALE, page 28.</p>
DEFAULT-NUMERIC	INTEGER FLOAT DECIMAL[(p)] V30	<p>Specifies the default numeric type for variables. The command line flag -DNT overrides this setting. Similarly, the DECLARE-VARIABLE command overrides this setting.</p> <p>See Chapter 2, "SQR Command Reference," DECLARE-VARIABLE, page 108.</p>
OutputFormFeedWithDashD=> {TRUE FALSE}	TRUE FALSE	<p>The default value is FALSE.</p> <p>When set to TRUE, the -Dnn command line flag outputs the Form-Feed character that denotes a page break.</p>
OutputTwoDigitYearWarningMsg=> {TRUE FALSE}	TRUE FALSE	<p>The default value is TRUE.</p> <p>When set to TRUE, SQR generates a warning message (sent to the warning file) when a YY or RR date edit mask is encountered during a program run. This affects only SQR code that is processed.</p>

<i>Entry</i>	<i>Value</i>	<i>Description</i>
UseY2kCenturyAlgorithm= {TRUE FALSE}	TRUE FALSE	The default value is FALSE. When set to TRUE, SQR treats the YY date edit mask as though it is an RR edit mask. See the RR edit mask.

Note. Use the setting V30 to process numbers in the same manner as in prior releases (before V4.0). Specifically, all numeric variables and literals are declared as FLOAT, including integer literals.

Processing-Limits Section

The Processing-Limits section is used to define the sizes and limitations of some of the internal structures used by SQR, which has a direct impact on memory requirements. The entries are the same as those used in the file specified with the -MFILE command-line flag. If the -MFILE command-line flag is used, the Processing-Limits section of the file is not processed.

<i>Entry</i>	<i>Default Value</i>	<i>Maximum Value</i>	<i>Entry Size</i>	<i>Description</i>
BREAKS	100	64K-1	4	Number of BREAK arguments allowed for each EVALUATE or IF command.
DYNAMICARGS	70	32K-1	14	Maximum number of dynamic SQL arguments.
EXPRESSIONSPACE	8192	64K-1	1	Maximum length, in bytes, of temporary string storage used during LET operations.
FORWARDREFS	200	32K-1	8	Maximum number of column forward references.
LONGSPACE	32K-2	32K-2	1	Maximum buffer size to transfer text and image data in bytes.

Entry	Default Value	Maximum Value	Entry Size	Description
ONBREAKS	30	64K-1	8	Maximum number of ON-BREAK LEVEL=values per SET.
POSITIONS	1800	64K-1	14	Maximum number of placement parameters, for example, (10,5,30).
PROGLINEPARS	18000	64K-1	2	Maximum number of arguments for all program lines. This value is generally 3 or 4 times the value set for PROGLINES.
PROGLINES	5000	32K-1	8	Maximum number of program lines (SQR commands).
QUERIES	60	32K-1	60	Maximum number of BEGIN-SQL and BEGIN-SELECT paragraphs. This is database dependent and can vary. This size is used as a close approximation.
QUERYARGS	240	64K-1	6	Maximum number of arguments (bind variables) for all SQL or SELECT statements. The number of arguments required is one more than the number used in your report file.
SQLSIZE	4000	64K-1	1	Maximum length of a SQL statement in characters.

Entry	Default Value	Maximum Value	Entry Size	Description
STRINGSPACE	15000	64K-1	1	Maximum size of string space for program line arguments, in bytes.
SUBVARS	100	32K-1	8	Maximum number of runtime substitution variables.
VARIABLES	1500	64K-1	18	Maximum number of variables (<i>string</i> , <i>float</i> , <i>integer</i> , <i>decimal</i>), literal values, and database columns. Add 4 to the entry size for Informix.
WHENS	70	64K-1	4	Maximum number of WHEN arguments allowed for each EVALUATE command.

The maximum value refers to the number of entries allowed as shown in the previous table; however, limits are lower for 32-bit machines. In any case, SQR indicates the limit if you exceed it.

In addition to increasing the sizes, you may also lower them to decrease the amount of memory used. This might be advantageous, for example, for certain applications running in the PC environment, where memory is limited.

Environment Sections

The Environment sections { Common | Oracle | Informix | ODBC | DB2 | RDB | Sybase } define the environment variables used by SQR. An environment variable can be defined in multiple environment sections; however, a definition in a database-specific environment section takes precedence over an assignment in the Environment:Common section.

The environment variables that can be set are *SQRDIR*, *SQRFLAGS*, and *DSQUERY*. On Microsoft Windows systems, *SQRDIR* is required and is automatically defined in the appropriate database-specific environment section during the SQR installation. The other environment variables are optional. *SQRFLAGS* specifies the default command-line flags for all invocations of SQR. *DSQUERY* (Sybase only) identifies the default Sybase server to use.

On Microsoft Windows systems only, the SQR Extension section defines DLLs containing new user functions (UFUNC) and user calls (UCALL). UFUNC and UCALL reside inside SQREXT.DLL and/or other DLLs.

When SQRW.DLL and SQRWT.DLL are being loaded, they look for SQREXT.DLL in the same directory, and for any DLLs specified in the SQR Extension section in sqr.ini such as:

```
[SQR Extension]
c:\sqrexts\sqrext1.dll=
c:\sqrexts\sqrext2.dll=
c:\sqrexts\sqrext3.dll=
```

Any new extension DLLs containing new user functions must be listed in the SQR Extension section in sqr.ini.

See *Enterprise PeopleTools 8.50 PeopleBook: SQR for PeopleSoft Developers*, "Using Interoperability Features."

For Oracle on Windows, SQR uses dynamic binding of Oracle routines. When SQR attempts to access an Oracle database, it searches for the Oracle DLL as follows:

- The file described by the value of ORACLE_DLL entry in the Environment:Oracle section of the sqr.ini file.
- OCIW32.DLL (Oracle supplied).
- ORANT71.DLL (Oracle supplied).

Entry	Value	Description
SQRParseSQLEnable={ TRUE FALSE }	TRUE FALSE	<p>The default setting is FALSE. If SQRParseSQLEnable=TRUE is added to the PSSQR.INI file (in the Common or Oracle stanza). SQR performs SQL parsing.</p> <p>Note. The function required to perform this parsing is not a supported function on Oracle 8.X but does appear to function. If a customer wishes to assume the risk of using a feature that might not be fixed if a problem is found, then they can turn on this parameter. Oracle 9.X customers can use this feature without any problem, although there is a performance hit of between 2-10%.</p>

Locale Section

This section specifies the default settings for the locale identified by locale-name (which can consist of A-Z, 0-9, hyphen, and underscore). A number of locales are predefined in the pssqr.ini file. Depending on your application, the settings for these locales may have to be altered or new locales may have to be added. A locale can be referenced or altered at runtime using the ALTER-LOCALE command. The entries for a locale section are described in the following table.

Note. The SYSTEM locale is provided for your reference, but is commented out. The settings for the SYSTEM locale, if set, are ignored. Use the ALTER-LOCALE command to change the SYSTEM locale settings at runtime.

See [Chapter 6, "Avoiding Older SQR Commands," page 283](#).

<i>Entry</i>	<i>Description</i>
NUMBER-EDIT-MASK	The default numeric edit mask format when the keyword NUMBER accompanies the DISPLAY, MOVE, PRINT, or SHOW command.
MONEY-EDIT-MASK	The default numeric edit mask format when the keyword MONEY accompanies the DISPLAY, MOVE, PRINT, or SHOW command.
DATE-EDIT-MASK	The default date edit mask format when the keyword DATE accompanies the DISPLAY, MOVE, PRINT, or SHOW command, or the LET datetostr() or strtodate() functions.
INPUT-DATE-EDIT-MASK	The default date format to use with the INPUT command when TYPE=DATE is specified with the command or the input variable is a DATE variable.
MONEY-SIGN	Specifies the characters to replace the (\$) edit character.
MONEY-SIGN-LOCATION	Specifies the location of the MONEY-SIGN character. Valid values are LEFT and RIGHT.
THOUSAND-SEPARATOR	Specifies the character to replace the comma (,) edit character.
DECIMAL-SEPARATOR	Specifies the character to replace the period (.) edit character.

Entry	Description
DATE-SEPARATOR	Specifies the character to replace the slash (/) character.
TIME-SEPARATOR	Specifies the character to replace the colon (:) character.
EDIT-OPTION-NA	Specifies the characters to replace the na option.
EDIT-OPTION-AM	Specifies the characters to replace AM
EDIT-OPTION-PM	Specifies the characters to replace PM.
EDIT-OPTION-AD	Specifies the characters to replace AD.
EDIT-OPTION-BC	Specifies the characters to replace BC.
DAY-OF-WEEK-CASE	<p>Specifies how the case for the DAY-OF-WEEK-FULL or DAY-OF-WEEK-SHORT entries are affected when used with the format codes <i>DAY</i> or <i>DY</i>. Valid values are UPPER, LOWER, EDIT, and NO-CHANGE.</p> <p>Use UPPER and LOWER to force the output to either all uppercase or lowercase, ignoring the case of the format code in the edit mask.</p> <p>Use EDIT to follow the case as specified with the format code in the edit mask.</p> <p>Use NO-CHANGE to ignore the case of the format code and output the day of week as explicitly listed in the DAY-OF-WEEK-FULL or DAY-OF-WEEK-SHORT entries.</p>
DAY-OF-WEEK-FULL	Specifies the full names for the days of the week. SQR considers the first day of the week to be Sunday. All seven days must be specified.
DAY-OF-WEEK-SHORT	Specifies the abbreviated names for the days of the week. SQR considers the first day of the week to be Sunday. All seven abbreviations must be specified.

<i>Entry</i>	<i>Description</i>
MONTHS-CASE	<p>Specifies how the case for the MONTHS-FULL or MONTHS-SHORT entries is affected when used with the format codes <i>MONTH</i> or <i>MON</i>. Valid values are UPPER, LOWER, EDIT, and NO-CHANGE.</p> <p>Use UPPER and LOWER to force the output to either all uppercase or lowercase, ignoring the case of the format code in the edit mask.</p> <p>Use EDIT to follow the case as specified with the format code in the edit mask.</p> <p>Use NO-CHANGE to ignore the case of the format code and output the month as explicitly listed in the MONTHS-FULL or MONTHS-SHORT entries.</p>
MONTHS-FULL	Specifies the full names for the months of the year. SQR considers the first month of the year to be January. All 12 months must be specified.
MONTHS-SHORT	Specifies the abbreviated names for the months of the year. SQR considers the first month of the year to be January. All 12 abbreviations must be specified.

Fonts Section

The Fonts section lists the fonts available to SQR when printing on Microsoft Windows printer devices (using the -PRINTER:WP command-line flag), viewing SPF output using SPF Viewer on Microsoft Windows (SQRWV), and creating style sheet for Enhanced HTML output. This section does not apply to PostScript or HP LaserJet printer types.

See [Chapter 2, "SQR Command Reference," DECLARE-PRINTER, page 93](#) and [Chapter 6, "Avoiding Older SQR Commands," page 283](#).

Adding Fonts Entries

Within the Fonts section, there are a number of predefined font entries. You can add entries by using the font numbers 900 through 999. Each entry consists of a font name, a font style (fixed or proportional), and a bold indicator, all of which are associated with a font number, for example:

```
4=Arial,proportional
or
300=Courier New,fixed,bold
```

Note. Proportional is assumed if the second parameter starts with a *P*. Also, bold is assumed if a third parameter is supplied.

Using the font number, commands such as ALTER-PRINTER and DECLARE-PRINTER can reference a particular font style.

Specifying Character Sets in Windows

On Microsoft Windows, you can use the CharacterSet entry either to determine the Microsoft Windows default character set or to specify a character set. This enables you to print any standard character set to a Windows printer (-PRINTER:WP) or to view an SPF file that displays the appropriate character set.

The syntax is:

```
CharacterSet=DEFAULT|AUTO|character_set
```

The arguments are:

- DEFAULT reflects current SQR functionality.
- AUTO automatically determines the default character set of the Microsoft Windows installation and uses the default set when generating reports.

CharacterSet specifies one of these keywords: ANSI, ARABIC, BALTIC, CHINESEBIG5, EASTEUROPE, GB2312, GREEK, HANGUL, HEBREW, JOHAB, MAC, OEM, RUSSIAN, SHIFTJIS, SYMBOL, THAI, TURKISH, VIETNAMESE.

HTML-Images Section

The HTML-Images section defines the parameters that SQR uses when generating HTML report output files.

<i>Entry</i>	<i>Value</i>	<i>DefaultValue</i>	<i>Description</i>
FIRST-PAGE	HEIGHT, WIDTH, NAME	60,60,firstpg.gif	Specifies the NAME of the graphic image file that accesses the first page of the report. The HEIGHT and WIDTH are values specified in pixels.
PREV-PAGE	HEIGHT, WIDTH, NAME	60,60,prevpg.gif	Specifies the NAME of the graphic image file that accesses the previous page of the report. The HEIGHT and WIDTH are values specified in pixels.

<i>Entry</i>	<i>Value</i>	<i>DefaultValue</i>	<i>Description</i>
NEXT-PAGE	HEIGHT, WIDTH, NAME	60,60,nextpg.gif	Specifies the NAME of the graphic image file that accesses the next page of the report. The HEIGHT and WIDTH are values specified in pixels
LAST-PAGE	HEIGHT, WIDTH, NAME	60,60,lastpg.gif	Specifies the NAME of the graphic image file that accesses the last page of the report. The HEIGHT and WIDTH are values specified in pixels.
WALLPAPER	NAME		Specifies the NAME of the graphic image file used as the background image for the report.
Navbar Background	NAME		Background image of navigation bar.

Note. SQR does not perform any validation of the graphic image file names provided. The user is responsible for ensuring that the graphic image files are in a location that the browser can access.

PDF Fonts Section

The PDF Fonts section lists the available fonts for SQR when printing using the -PRINTER:PD command-line flag. Fonts specified are case sensitive.

Unlike the Fonts section, the PDF Fonts section takes a list of fonts mapped to a single font number. You can specify up to 10 fonts for a single font number. Depending on the character to print, SQR determines the font to use from the list. The following is the syntax of this font list:

```
Font Number=Font1, Font2, Font3 ...
```

The list is ordered by priority from the left. If Font1 has a glyph (an image of a character) for the character you want to print, it is used, but if Font1 does not have a glyph, Font2 is checked and used if it has a glyph for that character.

Font Number is a decimal number that is used to specify fonts in the SQR program using statements like DEFINE-PRINTER or ALTER-PRINTER.

Font numbers that are multiples of 100 (300, 3200, and so on) are recognized as a bold version of the base font. For example, font 300 is the bold version of Font3.

Font numbers 30 to 39 are recognized as italic fonts, and multiples of 30 to 39 (3000, 3100, and so on) are bold italic versions of the font. If you assign nonbold fonts for a font number recognized as bold, (such as 600), the text is printed in bold.

Font1, Font2, and Font3 represent the font name. You can use Adobe Reader core fonts, Adobe Reader Asian Font Pack fonts, or TrueType fonts.

Adobe Reader core fonts are the fonts that Adobe Reader natively supports. The following is a list of the core fonts:

- Courier
- Courier-Bold
- Courier-Oblique
- Courier-BoldOblique
- Helvetica
- Helvetica-Bold
- Helvetica-Oblique
- Helvetica-BoldOblique
- Times-Roman
- Times-Bold
- Times-Italic
- Times-BoldItalic
- Symbol
- ZapfDingbats

Adobe Reader Asian Font Pack fonts are the font packages Adobe provides as add-on to Adobe Reader for the purpose of viewing Asian text. You can download these fonts from the Adobe website, or if you use Adobe Reader 6, the Asian Font Pack automatically downloads when you open a PDF file that contains a font from the Asian Font Pack. Alternatively, you can use the Adobe Reader localized to any of the Japanese, Chinese, or Korean languages. These localized versions come with the same fonts that are included in the Adobe Asian Font Pack. The following is a list of the Asian Font Pack fonts used with SQR:

- HeiseiKakuGo-W5 (Japanese)
- HeiseiMin-W3 (Japanese)
- MHei-Medium (Traditional Chinese)
- MSung-Light (Traditional Chinese)
- STSong-Light (Simplified Chinese)
- HYGoThic-Medium (Korean)
- HYSMyeongJo-Medium (Korean)

In addition, you can use TrueType/OpenType fonts that exist on the machine where you run SQR. If you use TrueType fonts for PDF output, they will be always embedded as subset of the original font. To use TrueType fonts for PDF output, you need to specify the mapping between the font name and the font file path in the [TrueType Fonts] section, and then include the font name in the font list under the [PDF Fonts] section. See below more information about the [TrueType Fonts] section.

See [Chapter 7, "Using the PSSQR.INI File and the PSSQR Command Line," TrueType Font Section, page 318.](#)

PDF Fonts: Exclusion Ranges Section

The PDF Fonts: Exclusion Ranges section defines character ranges you want to exclude from the range a font covers. Without an exclusion range, SQR uses all of the character ranges a font covers, and then checks the font listed in the next priority only when it does not cover a character. By specifying an exclusion range, you can use a font in a lower priority for a specific character or a range of characters. This mechanism allows users to control which font to use down to a single character level.

The syntax in this section is as follows:

```
Font Number=Font1 Exclusion Range, Font2 Exclusion Range, ...
```

Exclusion ranges apply to the list of fonts defined in the PDF Fonts section. For example, if you have a PDF Fonts section like the following:

```
3=Courier,Cumberland,HeiseiKakuGo-W5,...
```

In addition, if you define the PDF Fonts: Exclusion Ranges section like this

```
3=0x20AC,0x3070-0x30FF,,
```

then you are defining that Euro currency symbol (0x20AC in Unicode) is rendered using the Cumberland font, even though Courier has it. In addition, Greek characters (from 0x3070 to 0x30FF in Unicode) are rendered using HeiseiKakuGo-W5 font, even though Cumberland has them.

Each Exclusion range is specified using Unicode (UCS-2) codepoint, in decimal or in hexadecimal, in the following ways:

0x20AC (single character)

160-255 (range)

0x20AC|0x00A0-0x00FF (multiple characters and ranges)

Hexadecimal numbers need to be preceded by 0x; otherwise, they are recognized as decimal.

TrueType Font Section

The TrueType Font section defines the mapping of font name, which is an alias SQR uses internally to look up the font, and physical font file name. You need to specify the font in the full path, unless you have placed fonts in a directory specified in the Font Path parameter, or in a Windows font folder if you are running SQR on Windows.

The syntax is:

```
font name=fontfile path
```

For example, if you have the font `courier.ttf` in the `c:\user\fonts` directory, you set the following in this section:

```
CourierNew=c:\user\fonts\courier.ttf
```

Font name can be any string that is convenient for you to identify the font, and does not need correspond to the internal name of the font. You should not use the same font name that is used for Adobe core fonts or for Asian Font Pack fonts. If you set the same name for any of Adobe core fonts or Asian Font Pack fonts, the TrueType font is used.

Note. TrueType font embedding is now supported by SQR running on z/OS. However, the Font Path parameter under the TrueType Font section is not supported for z/OS. If you're on z/OS, you need to specify the full path to each entry.

If you have a TrueType Collection (TTC) file, you will also need to specify the font number to access the specific font included in the collection. For example, if MS P Mincho is included in the TrueType Collection file `msmincho.ttc`, then you will need to specify:

```
MSPMincho=c:\winnt\fonts\msmincho.ttc,1
```

The font number within the TrueType collection starts at 0. `msmincho.ttc` contains MS Mincho and MS P Mincho in this order; thus the number 0 represents MS Mincho and 1 represents MS P Mincho. If you do not specify font number, SQR uses font with font number 0 from the TrueType Collection file.

Note. You can use only Microsoft type of Unicode based TrueType/OpenType fonts with SQR. SQR requires TrueType/OpenType font to have CMAP table with Platform ID 3 (Microsoft), Encoding ID 0 (Symbol), 4 (UCS-2), or 10 (UCS-4) and table format 4 or 12. SQR does not support OpenType fonts with CFF (Postscript) outline. You can use OpenType fonts with TrueType outlines, but SQR does not make use of advanced layout features provided with OpenType font.

Enhanced-HTML Section

The Enhanced-HTML section is used to define various default actions that SQR takes when generating HTML output using the `-EH` command-line flag.

<i>Entry</i>	<i>Value</i>	<i>Description</i>
Browser={ ALL IE NETS CAPE }	ALL IE NETSCAPE	<p>When set to ALL, the generated HTML automatically determines which browser is being used and invokes the proper browser-specific file.</p> <p>When set to IE, the generated HTML is designed for Internet Explorer.</p> <p>When set to NETSCAPE, the generated HTML is designed for Netscape.</p> <p>The default action is to generate HTML suitable for all browsers.</p>
Language={ English French German Portuguese Spanish Japanese Simplified Chinese Korean }	English French German Portuguese Spanish Japanese Simplified Chinese Korean	<p>Sets the language used for the HTML navigation bar.</p> <p>The default setting is English.</p>
FullHTML={ TRUE FALSE }	TRUE FALSE	<p>When set to TRUE, HTML 3.2 is generated. When set to FALSE, HTML 3.0 is generated. The default setting is FALSE.</p>

Colors Section

The Colors section defines the default colors that you can use in your SQRs. Enter the default colors in this format:

```
[Colors]
color_name = ({rgb})
color_name = ({rgb})
...
color_name = ({rgb})
```

The default colors implicitly installed are:

- black=(0,0,0)
- white=(255,255,255)
- gray=(128,128,128)
- silver=(192,192,192)
- red=(255,0,0)
- green=(0,255,0)

- blue=(0,0,255)
- yellow=(255,255,0)
- purple=(128,0,128)
- olive=(128,128,0)
- navy=(0,0,128)
- aqua=(0,255,255)
- lime=(0,128,0)
- maroon=(128,0,0)
- teal=(0,128,128)
- fuchsia=(255,0,255)

Using PSSQR.EXE Command-Line Options

The PeopleSoft system provides a shell called pssqr.exe that extends SQR to handle the submission of SQR programs under Microsoft Windows and UNIX/Linux operating systems. Pssqr.exe implements a process that ensures that output is sent to the appropriate destination in a way that is consistent across platforms.

Note. PeopleSoft does not support running pssqr.exe directly from a DOS or Unix/Linux command line. Pssqr.exe is a wrapper program used by PeopleSoft Process Scheduler to run SQR reports; it is not designed to run manually outside of Process Scheduler.

Pssqr.exe does not run on OS/390, however, all output formats are supported on all platforms.

Pssqr.exe provides the following features:

- Expanded output formats: SQR Viewer, HTML, PDF, CSV (Spreadsheet Standard), HP, Postscript, Line Printer.
- Expanded printer format: Microsoft Windows Default Printer (Win32 only), HP, Postscript, Line Printer.
- Enhanced delivery of reports to printers: PSSQR sends reports to the printer instead of SQR, which resolves issues that can be encountered on non-Microsoft networks.
- File output and logs have unique names. If the process instance is sent, the file names will be <SQR Program>_<Instance>.xxx.
If not, then the log names will be <SQR Program>_<timestamp>.xxx.
- Common command line interface for both Microsoft Windows and UNIX/Linux.
- Reads Configuration Manager settings to determine the flags (SQRFLAGS) to use.
- Support for multiple report output.

Default values come from Configuration Manager if run from the command line.

Command-Line Parameter	Description
-CT	Database type. Valid values are ORACLE, DB2, DB2390, INFORMIX, MICROSOFT, SYBASE.
-CS	Server name.
-COWN	Database owner. This applies to OS/390 only.
-CD	Database name.
-CA	Access ID.
-CAP	Access password.
-RP	Program name.
-I	Process instance.
-R	Run control ID.
-CO	User ID.
-OT	Output type: <ul style="list-style-type: none"> • 2 (File) • 3 (Printer) • 4 (Window)

Command-Line Parameter	Description
-OF	Output format. Column headings refer to the -OT parameter. <ul style="list-style-type: none"> • 2 (Adobe Acrobat) • 3 (Comma delimited) • 4 (HP format) • 5 (HTML documents) • 6 (Line printer) • 10 (Postscript) • 13 (SQriBE portable) • 14 (Text files) • 15 (Windows default printer)
-OP	Output destination.
-MR	Multiple report output. Values: # (maximum 99)
-AF	Additional flags.
-AP	Additional information.
-TR	Enable trace.
-DL	Display log file.
-FILE	Enter the name of a parameter file containing the parameters you want to pass to PSSQR. The system deletes the file immediately after use.
-LG	Language Code. Takes a three-letter language code used across PeopleSoft systems. This flag is used to determine which language version of the PSSQR.INI file to use. If you specify LG JPN, then pssqr.exe picks up the Japanese configuration file (PSSQRJPN.INI) instead of PSSQR.INI. This parameter takes effect only when the %LANGUAGE_CD% variable is used for specifying the PSSQR.INI file in PSSQRFLAGS entry of the Process Scheduler configuration or Configuration Manager.

<i>Command-Line Parameter</i>	<i>Description</i>
-LPFLAGS "<flags>"	UNIX/Linux only: Overrides the flags passed to the lp command.
-DEBUGLP ON	Displays command used to print the SQR report.

Note. Note: Avoid using a hyphen (-) or @ sign in the PSSQR.EXE command line, for instance as part of the run control ID. The hyphen (-) and @ sign characters are SQR reserved characters.

See Also

Enterprise PeopleTools 8.50 PeopleBook: SQR for PeopleSoft Developers, "Using the SQR Command Line," Using Reserved Characters

Appendix A

SQR Messages

This chapter describes all the messages produced by SQR for the PeopleSoft application. It discusses:

- Unnumbered messages
- Numbered messages

Note. Two digits (`nn) appear as replacement markers in the messages. Descriptions of these replacement markers are listed in the message itself. The messages contain the proper value when they appear on the screen.

Unnumbered Messages

This table lists all unnumbered SQR messages:

<i>Message</i>	<i>Description</i>
Out of memory.	This error occurs when a call to the C routine 'malloc()' fails. (32-bit machines) Use the -Mfile command-line flag to reduce some of the different memory requirements. Remove unneeded TSRs. (UNIX/Linux) Increase the size of the system swap file.
No cursors defined.	This message is issued from the -S command-line flag. The SQR program did not contain any commands that required a database cursor.
Not processed due to report errors.	This message is issued from the -S command-line flag. SQR cannot provide information about the cursor due to errors in the program.
Enter `01`02	This message prompts the user to type the value to be assigned to the specified variable. `01 = First character of the variable name. `02 = Rest of the variable name.

Message	Description
NOPROMPT used - Enter value below	(Microsoft Windows) This message appears when an INPUT command is defined with the NOPROMPT argument.
Enter `01	This message prompts the user to enter the value to be assigned to the specified substitution variable. `01 = Name of the substitution variable
Enter this run's parameters:	This message prompts the user to enter the values for the parameters defined in the program.
Error on line `01: `02	SQR detected an error while processing the report file. Correct the error and rerun. `01 = Source line number. `02 = Source line.
Error in include file "`01" on line `02: `03	SQR detected an error while processing the report file. Correct the error and rerun. `01 = Name of the include file. `02 = Source line number. `03 = Source line.
Warning on line `01: `02	SQR detected a nonfatal error while processing the report file. `01 = Source line number. `02 = Source line.
Warning in include file "`01" on line `02: `03	SQR detected a nonfatal error while processing the report file. `01 = Name of the include file. `02 = Source line number. `03 = Source line.
Type RETURN for more, C to continue w/o display, X to exit run:	This informational message is used in conjunction with the -D command-line flag.
Error at: `01 Loading Oracle DLL Failed!!!	(Oracle) This title for the dialog box informs the user that SQR could not load the Oracle DLL.

Message	Description
Errors were found in the program file.	This error message tells the user that there were errors in the program. The user can correct the errors and rerun.
Errors were found during the program run.	This error message tells the user to there were errors running the program. The user can correct the errors and rerun.
`01: End of Run.	This is an informational message. `01 = Image name (for example, SQR)
Enter report name:	This message prompts the user to enter the name of the report (.sqr or .sqt) to run.
Enter database name:	This message prompts the user to enter the name of the database.
Enter database[/username]: Enter Username:	This message prompts the user to enter the user name to log onto the database.
Enter Password:	This message prompts the user to enter the password. For security reasons, the password is not echoed.
Customer ID:	This is a text message.
Press Enter to close...	This is a text message.
`01: Program Aborting.	This is an informational message. `01 = Image name (for example, SQR)
*** Internal Coding Error ***	This is an informational message.
SQL DataServer Message	(Microsoft Windows) This message is a title for the error message dialog box.
Operating-System error	(Microsoft Windows) This message is a title for the error message dialog box.
DB-Library error	(Microsoft Windows) This message is a title for the error message dialog box.

Message	Description
`01 is running. Click the Cancel button to interrupt it.	(Microsoft Windows) This message is the body of the - C cancel dialog box. The user can quit the program run by clicking the Cancel button.
Table of Contents	This message is the text for HTML driver.
Previous	This message is the text for HTML driver.
Next	This message is the text for HTML driver.
First Page	This message is the text for HTML driver.
Last Page	This message is the text for HTML driver.
PAGE	This message is the text for HTML driver.

Numbered Messages

This table lists numbered messages:

Error Number	Error Message	Suggestion/Interpretation
000001	Error while opening the message file: `01' (`02): `03	Try reloading the sqrrrr.dat file from the release media. If the error persists, contact technical support. `01 = Name of the error message file. `02 = System error code. `03 = System error message.
000002	Error while reading the message file. (`01): `02	Try reloading the sqrrrr.dat file from the release media. If the error persists, contact technical support. `01 = Name of the error message file. `02 = System error code. `03 = System error message.

Error Number	Error Message	Suggestion/Interpretation
000003	Error while closing the message file. (^01): ^02	Try reloading the sqrrrr.dat file from the release media. If the error persists, contact technical support. ^01 = Name of the error message file. ^02 = System error code. ^03 = System error message.
000004	Error while seeking the message file. (^01): ^02	Try reloading the sqrrrr.dat file from the release media. If the error persists, contact technical support. ^01 = Name of the error message file. ^02 = System error code. ^03 = System error message.
000005	Corrupt message file: Invalid header information.	Try reloading the sqrrrr.dat file from the release media. If the error persists, contact technical support.
000006	Corrupt message file: Invalid count (Got ^01, Should be ^02).	The header contains an invalid entry count. <ol style="list-style-type: none">1. Make sure SQRDIR points to the correct directory.2. Try reloading the sqrrrr.dat file from the release media. If the error persists, contact technical support. ^01 = The value read from the header. ^02 = The correct value.

Error Number	Error Message	Suggestion/Interpretation
000007	Cannot handle message file version `01.	<p>This message occurs when the release of SQR does not support the header version.</p> <ol style="list-style-type: none"> 1. Make sure SQDIR points to the correct directory. 2. Try reloading the sqrrr.dat file from the release media. If the error persists, contact technical support. <p>`01 = Unsupported version read from the header.</p>
000010	Invalid SEMCode encountered: `01.	<p>An invalid code was passed to the error message handler. Try reloading the files from the release media. If the error persists, contact technical support.</p> <p>`01 = Invalid code.</p>
000011	Unknown conversion type (`01) for code `02.	<p>Try reloading the sqrrr.dat file from the release media. If the error persists, contact technical support.</p> <p>`01 = Invalid type. `02 = Internal error code.</p>
000012	Message `01 must be either Preload or BuiltIn.	<p>The type error code is not correct. Try reloading the sqrrr.dat file from the release media. If the error persists, contact technical support.</p> <p>`01 = Error code.</p>
000013	Cannot point to message `01.	<p>The error handler cannot position to the desired error code. Try reloading the sqrrr.dat file from the release media. If the error persists, contact technical support.</p> <p>`01 = Error code.</p>
000014	The required environment variable `01 has not been defined.	<p>Define the named environment variable and restart SQR.</p> <p>`01 = Environment variable name.</p>

Error Number	Error Message	Suggestion/Interpretation
000015	The Meta ESC characters do not match (Got `01', Should be `02').	The meta escape character defined in the header does not match what the error message handler expects. Try reloading the sqrrr.dat file from the release media. If the error persists, contact technical support. `01 = What was found in the header. `02 = What was expected to be found
000016	`01() called to process (`02) and the message file is not open.	The specified error routine was called but the error message file was not open. Try reloading the files from the release media. If the error persists, contact technical support. `01 = Name of the routine `02 = Error code
000017	Message `01 must be ReportParameters or CopyrightNotice.	Try reloading the sqrrr.dat file from the release media. If the error persists, contact technical support. `01 = Error code
000018	Allocation header does not point to a valid heap.	(Microsoft Windows) This message is the result of a memory overwrite. Record the steps leading up to the error and contact technical support.
000019	Allocation header has an invalid size.	(Microsoft Windows) This message is the result of a memory overwrite. Record the steps leading up to the error and contact technical support.
000020	GLOBAL header has an invalid size.	(Microsoft Windows) This message is the result of a memory overwrite. Record the steps leading up to the error, and contact technical support.
000021	Cannot free GLOBAL allocation.	(Microsoft Windows) This message is the result of a memory overwrite. Record the steps leading up to the error and contact technical support.

Error Number	Error Message	Suggestion/Interpretation
000028	Cannot access the initialization file: `01 (`02): `03	The initialization file specified by the -ZIF command line flag cannot be accessed. `01 = Name of the file. `02 = System error code. `03 = System error message.
000202	DPUT: Bad field number.	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
000203	DARRAY: Unknown command number.	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
000204	`01: Cannot find `02 command.	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. `01 = Name of the routine. `02 = Name of the command.
000205	DDO: DO arguments do not match procedure's.	<obsolete>
000206	SDO: Bad params for DO command.	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
000207	SDO: Bad params for BEGIN-PROCEDURE command.	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.

Error Number	Error Message	Suggestion/Interpretation
000208	SGOTO: Bad command numbers.	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
000209	SGOTO: Bad goto function parameters.	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
000210	SGOTO: Could not find beginning of section or paragraph.	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
000211	SGOTO: Bad label: from parameters.	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
000212	COMPAR: Unknown relational (numeric) operator.	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
000213	COMPAR: Unknown relational (string) operator.	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
000214	DONBRK: Unknown case for putlin.	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.

Error Number	Error Message	Suggestion/Interpretation
000215	`01: Bad length case for numeric `02.	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. `01 = Name of the routine. `02 = name of the variable.
000216	GARRAY: Unknown command number.	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
000217	GCMDS: No Gfunc found.	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
000218	GDOC: Unknown document type.	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
000219	GLET: Bad operator.	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
000220	GLET: Stack incorrect for expression - arg `01.	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. `01 = Number of the argument.
000221	GLET: Unknown operator type.	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.

Error Number	Error Message	Suggestion/Interpretation
000222	GLET: Unknown operator in expression.	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
000223	GPARS: Column not SCOL, TCOL or NCOL type.	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
000224	GPARS: Bad parameter format: `01 =`02=	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. `01 = Internal command format string. `02 = Bad format field found.
000225	GPARS: No end of required word in parfmt: `01	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. `01 = Internal command format string.
000226	GPARS: Bad parfmt entry: `01	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. `01 = Internal command format string.
000227	GPARS: Bad parameter string.	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.

Error Number	Error Message	Suggestion/Interpretation
000228	GPARS: Repeat count bad: `01	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. `01 = Internal command format string.
000229	GPARS: Only a,b,8,9 allowed for repeats: `01	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. `01 = Internal command format string.
000230	GPARS: Missing required x: `01	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. `01 = Internal command format string.
000231	GPARS: Bad type in 'ckvrpr()'. 	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
000232	GPROC: No Gfunc found.	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
000233	GRDWRT: Unknown command number.	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
000234	GSHOW: Unknown SHOW option.	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.

Error Number	Error Message	Suggestion/Interpretation
000235	PGMPARS: 'addvar()' passed maxlen but not column.	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
000238	PGMPARS: `01' passed invalid parameter number: `02.	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. `01 = Routine name. `02 = Invalid parameter number.
000239	PGMPARS: 'fxclrf()' encountered bad column reference type: `01.	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. `01 = Internal variable type code.
000240	PLCMNT: 'getplc()' passed invalid element number: `01.	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. `01 = Invalid element number.
000241	RDPGM: Command array size exceeded (change COMDMAX to at least `01).	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. `01 = Maximum internal command number supported.
000242	RDPGM: Bad match adding internal variable: `01	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. `01 = Internal variable name.

Error Number	Error Message	Suggestion/Interpretation
000243	RDPGM: No cmdget function found for BEGIN_S.	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
000244	Function `01 not included in run-time package.	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. `01 = Name of the SQR routine.
000245	SETSQL: Could not find variable `01', in Run Time.	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. `01 = Variable name.
000248	SIFWHL: Command number incorrect.	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
000249	SPINIT: Bad parameters.	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
000251	DBFFIX: DBDATLEN returned out of range status.	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
000252	DPRPST: Error converting Sybase type for EXECUTE.	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.

Error Number	Error Message	Suggestion/Interpretation
000254	SETSQL: Could not find variable entry in list.	(Oracle) This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
000255	DBDESC: SQLD not = number of select columns.	(DB2, Informix) This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
000256	DBFETCH: Unknown variable dbtype encountered: `01 (`02)	(DB2, Informix) This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. `01 = Variable name. `02 = Unknown database type.
000257	WRITE_SPF: Unknown code encountered: `01	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. `01 = Unknown SPF code.
000258	`01: Cannot find LOAD-LOOKUP table: `02	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. `01 = Name of the routine. `02 = Name of the table.
000259	PGMPARS: `01' called with wrong variable `02'	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. `01 = Name of the routine. `02 = Name of the variable.

Error Number	Error Message	Suggestion/Interpretation
000260	SQTMGT: Could not find 'vars' entry with 'nvars' index of `01'.	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. `01 = Index into nvars table.
000261	MODIFYVAR: Attempt to change variable which is not xVAR (^01).	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. `01 = name of the variable.
000262	MODIFYVAR: Incompatible variable types (^01) and (^02).	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. `01 = Variable type (from). `02 = Variable type (to).
001100	Out of query arguments; use -Mfile to increase QUERYARGS.	This is the total number of variable references (\$Var, #Var, &Col) allowed in the context of a BEGIN-SQL or BEGIN- SELECT command. Use the -Mfile flag on the command line to specify a file containing an entry that increases the currently defined value.
001201	Cannot open the argument file: `01'. (^02): `03	Depends on the system error message. `01 = Name of the file. `02 = System error code. `03 = System error message.
001202	Cannot close the argument file. (^01): `02	Depends on the system error message. `01 = System error code. `02 = System error message.

Error Number	Error Message	Suggestion/Interpretation
001203	Cannot open the - Mfile: ^01'. (^02): ^03	Depends on the system error message. ^01 = Name of the file. ^02 = System error code. ^03 = System error message.
001204	Minimum value for ^01' in the - Mfile is ^02.	Correct the -Mfile entry. ^01 = Keyword in question. ^02 = Minimum value allowed.
001205	Maximum value for ^01' in the - Mfile is ^02.	Correct the -Mfile entry. ^01 = Keyword in question. ^02 = Maximum value allowed.
001206	Invalid -Mfile entry: ^01'.	Correct the -Mfile entry. ^01 = The line from the -Mfile.
001207	Cannot close the - Mfile. (^01): ^02	Depends on the system error message. ^01 = System error code. ^02 = System error message.
001209	The minimum value for ^01' (^02) is ^03.	Value out of range. ^01 = Entry name. ^02 = Specified value. ^03 = Minimum value.
001210	The maximum value for ^01' (^02) is ^03.	Value out of range. ^01 = Entry name. ^02 = Specified value. ^03 = Maximum value.
001211	The value for ^01' (^02) is not an integer number.	Value must be a integer value. ^01 = Entry name. ^02 = Specified value.

Error Number	Error Message	Suggestion/Interpretation
001300	Bind list does not match query (do not use '@__p' string).	SQR reserves the variable names that start with "@__p" for internal use. Edit the source code and use different variable names.
001301	Forward references not permitted in select list bind variables.	Within the body of BEGIN-SQL paragraphs, forward references to &column names are not permitted. Move the BEGIN-SQL paragraph after the &column definition.
001302	SQL buffer too small; use -Mfile to increase SQLSIZE.	The SQL statement exceeds the size of the internal SQL buffer. Use the -Mfile flag on the command line to specify a file containing an entry that increases the currently defined value.
001303	Error in SQL (perhaps missing &name after expression):	The database server has determined that the SQL statement is in error. The actual error text from the server follows this message. Correct the SQL statement.
001304	Check SELECT columns, expressions and 'where' clause for syntax.	The database server has determined that the SQL statement is in error. The actual error text from the server follows this message. Correct the SQL statement.
001305	CMPSQL: Unknown data type in database: `01.	Contact technical support with the version of the database you are connected to. `01 = Datatype in question.
001306	Bind value too large (IMAGE, TEXT not allowed).	IMAGE and TEXT data types cannot be used as bind variables. Modify your SQL statement to use other columns to perform the same selection logic.
001307	CMPSQL: DBDEFN failed.	(Oracle, ODBC, Informix) This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.

Error Number	Error Message	Suggestion/Interpretation
001308	`01: Could not bind column `02.	(Oracle, ODBC, Informix) This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. `01 = Name of the SQR routine. `02 = Name of the column.
001309	The type for '&`01' (`02) does not match the type from the database (`03).	Correct the source code. `01 = Name of the column/expression pseudonym. `02 = User specified type. `03 = Database type.
001400	Only numerics allowed for arithmetic.	Only #numeric variables, &columns, and literals are permitted in the arithmetic commands. Correct the source code.
001401	Optional qualifier is ROUND=n (0-`01).	Correct the syntax. `01 = Maximum value for ROUND.
001402	Optional qualifiers for DIVIDE are ON- ERROR={HIGH ZER O} and ROUND=n.	Correct the syntax.
001403	Attempting division by zero.	Use the ON-ERROR = HIGH ZERO option to prevent this error from halting the program.
001404	Bad number of digits to ROUND or TRUNC (0-15).	Correct the syntax.
001405	WARNING: The ROUND or TRUNC qualifier is greater than the number's precision.	Correct the syntax.
001500	Array element out of range (`01) for array `02' on line `03.	Correct the source logic. `01 = Element number passed. `02 = Name of the array. `03 = Program line number.

Error Number	Error Message	Suggestion/Interpretation
001501	Field element out of range (^01) for array ^02', field ^03', on line ^04.	Correct the source logic. ^01 = Element number passed. ^02 = Name of the array. ^03 = Name of the field. ^04 = Program line number.
001502	WARNING: Attempting division by zero on line ^01. Array field ^02' unchanged. Run continuing...	The ARRAY-DIVIDE command has attempted division by zero. The division has been ignored; the result field is unchanged. Add logic to account for this possibility. ^01 = Program line number. ^02 = Name of field.
001601	'FILL' not appropriate for numeric data.	The FILL argument to the PRINT command may be used only for text fields. Move the #numeric variable to a \$string variable, and then print the string variable.
001700	Report ^01': Columns must be between 1 and the page width (^02).	The specified value is wider than the width of the page. Correct the source line. ^01 = Name of the current report. ^02 = Page width.
001702	Report ^01': GOTO- TOP=^02 must be between 0 and the page depth (^03).	The value specified on the GOTO-TOP argument of the NEXT-COLUMN command was either less than 1 or greater than the page depth. Correct the source line. ^01 = Name of the current report. ^02 = Goto-Top value. ^03 = Page width.
001703	Report ^01': ERASE- PAGE=^02 must be between 0 and the page depth (^03).	The line number specified on the ERASE- PAGE argument of the NEXT-COLUMN command is greater than the page depth. Correct the source line. ^01 = Name of the current report. ^02 = Erase-Page value. ^03 = Page width.

Error Number	Error Message	Suggestion/Interpretation
001704	Report `01': The NEXT-COLUMN command is not legal in the `02 section with the qualifier AT-END=NEWPAGE.	Correct the source line. `01 = Name of the current report `02 = Name of the section
001705	Report `01': Column number `02 is not defined.	The column number specified with the USE- COLUMN command is greater than the highest column defined in the COLUMNS command. Correct the source line. `01 = Name of the current report `02 = Column number
001800	Format for CONNECT: username/password [ON- ERROR=procedure[(arg1[,argi]...)]]	Correct the syntax.
001801	Cannot use CONNECT while SQL statements are active.	Correct the program logic to ensure that all BEGIN-SELECT paragraphs have completed before executing the CONNECT command.
001802	Logoff failed prior to CONNECT.	The database server returned an error while trying to log off from the database. SQR ends the program run since it cannot continue.
001803	CONNECT failed. Perhaps user name/password incorrect.	The specified connectivity information is incorrect or there might have been a network failure. Use the ON-ERROR flag to trap any errors during the program run; otherwise SQR ends the program run.
001804	Sybase extensions SET and SETUSER not permitted in SQR.	Remove SET and SETUSER from the source.
001805	USE allowed once in SETUP section only, not in BEGIN-SQL. Elsewhere, specify db.[user].table...	Correct the source.

Error Number	Error Message	Suggestion/Interpretation
001806	Out of query space. Use -Mfile to increase QUERIES.	The number of SQL statements has been exceeded. Use the -Mfile flag on the command line specify a file that contains an entry that increases a greater value than is currently defined.
001807	The requested database connection (^01) is already active.	The -Cnn value specified is being used by another BEGIN-SELECT paragraph that is currently selecting data. Use another connection number. ^01 = Connection number
001808	Cannot find inactive database cursor. Program too large.	Too many BEGIN-SELECT and BEGIN-SQL paragraphs are active at the same time. Reduce the complexity of the program.
001809	Database commit failed.	(Oracle, DB2, ODBC) This is an error that should never occur during normal operations. Record the steps leading up to the error and contact your system administrator.
001810	Database rollback failed.	(Oracle, DB2, ODBC) This is an error that should never occur during normal operations. Record the steps leading up to the error and contact your system administrator.
001811	Cannot open database cursor.	(Oracle, ODBC) This is an error that should never occur during normal operations. Record the steps leading up to the error and contact your system administrator.
001901	Variable for date-time must begin with '&'.	Correct the syntax.
001913	Format code must be SYYYY when specifying signed year.	Correct the edit mask.
001914	Bad input data (^01) for edit mask: ^02'.	Correct the input. ^01 = Data being converted ^02 = Edit mask

Error Number	Error Message	Suggestion/Interpretation
001915	Year cannot be zero.	Correct the date.
001916	Year must be between -4713 and 9999 inclusive.	Correct the date.
001917	Ambiguous date-time.	Correct the date.
001918	`01' is not a valid date part.	Correct the date part. `01 = Date part.
001919	Invalid day of week.	Correct the date.
001920	Format code cannot appear in date input format: `01'.	Correct the edit mask. `01 = Improper format characters.
001921	Bad date mask starting at: `01'.	Correct the edit mask. `01 = Improper format characters.
001922	Seconds past midnight must be between 0 and 86399.	Correct the date.
001923	Seconds must be between 0 and 59.	Correct the date.
001924	Minutes must be between 0 and 59.	Correct the date.
001925	Month must be between 1 and 12.	Correct the date.
001926	Day must be between 1 and `01.	Correct the date.
001927	Hour must be between 1 and 12.	Correct the date.
001928	Hour must be between 0 to 23.	Correct the date.
001929	HH24 precludes the use of meridian indicator.	Correct the edit mask.
001930	HH12 requires meridian indicator.	Correct the edit mask.

Error Number	Error Message	Suggestion/Interpretation
001931	Day of year must be between 1 and 365 (366 for leap year).	Correct the date.
001932	Date string too long.	Correct the date.
001933	The month (`01) is not valid for the current locale or database.	Correct the date. `01 = Name of the month.
001934	The format mask must be a literal when the date-time is not loaded into a variable.	Correct the format mask. The format mask must be a literal when the date-time is not loaded into a variable.
001935	Date-time format too long.	Correct the format mask.
001936	Bad date-time format.	Correct the format mask.
001937	Bad SQL for default date-time. (Table DUAL required for syntax.)	(Oracle) The format mask needs to be corrected or there is a problem with the database server.
001937	Bad SQL for default date-time. (Table DUAL required for syntax.)	(DB2) The format mask needs to be corrected or there is a problem with the database server.
001938	Cannot recompile sql.	A fatal error relating to the SQL statement used to retrieve the date-time was encountered. Record the steps leading up to the error and contact your system administrator.
001939	Problem executing cursor.	A fatal error relating to the SQL statement used to retrieve the date-time was encountered. Record the steps leading up to the error and contact your system administrator.
001940	Error fetching row.	A fatal error relating to the SQL statement used to retrieve the date-time was encountered. Record the steps leading up to the error and contact your system administrator.

Error Number	Error Message	Suggestion/Interpretation
001941	Cannot redefine variable addresses.	A fatal error relating to the SQL statement used to retrieve the date-time was encountered. Record the steps leading up to the error and contact your system administrator.
001942	The date `01' is not in the format SYYYYMMDD[HH24[MI[SS[N NNNNN]]]].	When specifying an SQR date, at a minimum, the date must be specified. The time is optional. `01 = The invalid date.
001943	The date `01' is not in one of the accepted formats listed below: MM/DD/YYYY [BC AD] [HH:MI[:SS[.NNNNN N]] [AM PM]] MM-DD-YYYY [BC AD] [HH:MI[:SS[.NNNNN N]] [AM PM]] MM.DD.YYYY [BC AD] [HH:MI[:SS[.NNNNN N]] [AM PM]] SYYYYMMDD[HH24[MI[SS[N NNNNN]]]]	The date specified with the INPUT command was not in one the default formats. Reenter the date in a valid format. `01 = The invalid date.
001944	The date `01' is not in the format specified by SQR_DB_DATE_FORMAT or in one of the accepted formats listed below: DD-MON-YY SYYYYMMDD[HH24[MI[SS[N NNNNN]]]]	(Oracle) The date was not in one of the expected formats for this database. `01 = The invalid date.

Error Number	Error Message	Suggestion/Interpretation
001944	<p>The date `01' is not in the format specified by SQR_DB_DATE_FORMAT or in one of the accepted formats listed below:</p> <p>Mon DD YYYY [HH:MI[:SS[.NNN]] [AM PM]]</p> <p>Mon DD YYYY [HH:MI[:SS[:NNN]] [AM PM]]</p> <p>YYYYMMDD [HH:MI[:SS[.NNN]] [AM PM]]</p> <p>YYYYMMDD [HH:MI[:SS[:NNN]] [AM PM]]</p> <p>SYYYYMMDD[HH24[MI[SS[N NNNN]]]]</p>	<p>(Sybase) The date was not in one of the expected formats for this database.</p> <p>`01 = The invalid date.</p>
001944	<p>The date `01' is not in the format specified by SQR_DB_DATE_FORMAT or in one of the accepted formats listed below:</p> <p>Mon DD YYYY [HH:MI[:SS[.NNN]] [AM PM]]</p> <p>Mon DD YYYY [HH:MI[:SS[:NNN]] [AM PM]]</p> <p>YYYYMMDD [HH:MI[:SS[.NNN]] [AM PM]]</p> <p>YYYYMMDD [HH:MI[:SS[:NNN]] [AM PM]]</p> <p>SYYYYMMDD[HH24[MI[SS[N NNNN]]]]</p>	<p>(ODBC) The date was not in one of the expected formats for this database.</p> <p>`01 = The invalid date.</p>
001944	<p>The date `01' is not in the format specified by SQR_DB_DATE_FORMAT or in one of the accepted formats listed below:</p> <p>YYYY-MM-DD HH:MI:SS.NNN SYYYYMMDD[HH24[MI[SS[N NNNN]]]]</p>	<p>(Informix) The date was not in one of the expected formats for this database.</p> <p>`01 = The invalid date.</p>

Error Number	Error Message	Suggestion/Interpretation
001944	The date `01' is not in the format specified by SQR_DB_DATE_FORMAT or in one of the accepted formats listed below: YYYY-MM-DD[-HH.MI.SS[.NNNNNN]] MM/DD/YYYY DD.MM.YYYY SYYYMMDD[HH24[MI[SS[N NNNNN]]]]	(DB2) The date was not in one of the expected formats for this database. `01 = The invalid date.
001945	SQR does not support dates before `01'.	SQR does not support dates before the one specified in the message. `01 = Smallest date
001946	The date variables are incompatible with each other.	The SQR function references two date variables that cannot logically be used together, for example, DateDiff of 'date- only' and 'time-only' dates.
002000	Procedure name used more than once: `01'.	Specify a unique name for the procedure. `01 = Procedure name
002001	Could not find procedure: `01'.	Verify the spelling of the procedure name. `01 = Procedure name
002002	DO arguments do not match procedure's.	The argument lists for the DO and BEGIN- PROCEDURE commands must match in both type and count. Correct the source line.
002003	DO argument must be \$string or #number to accept returned value.	Correct the syntax.
002100	Edit string too long.	The edit mask must be less than 255 characters. Reduce the length of the edit mask.

Error Number	Error Message	Suggestion/Interpretation
002101	Bad numeric 'edit' format: `01	The numeric edit mask contains an invalid character. See the PRINT command for the valid numeric edit mask characters. `01 = Invalid character
002103	DOLLAR-SYMBOL must be a single alphanumeric character or its decimal value enclosed in brackets: <nnn>.	Correct the syntax.
002104	DOLLAR-SYMBOL cannot be any of the following characters: `01	Correct the syntax. `01 = List of invalid characters
002106	MONEY-SYMBOL must be a single alphanumeric character or its decimal value enclosed in brackets: <nnn>.	Correct the syntax.
002107	MONEY-SYMBOL cannot be any of the following characters: `01	Correct the syntax.
002200	ENCODE string too large; maximum is `01.	Break up the ENCODE command. `01 = Maximum length of an ENCODE string supported by this version of SQR for PeopleSoft.
002300	EXIT-SELECT failed.	The database command to cancel the query returned an error. Try running the SQR program again. The error could be related to a network or server problem. If the error persists, contact your system administrator.
002301	EXIT-SELECT valid only within SELECT paragraph.	Remove the EXIT-SELECT command.
002400	Duplicate label's - do not know which one to GOTO.	Labels must be unique within the section or paragraph where they are defined. Give each label a unique name.

Error Number	Error Message	Suggestion/Interpretation
002401	(Labels must be in same section or paragraph as GOTO.) Cannot find a matching label for GOTO command.	Check the source code.
002500	Error getting INPUT.	The C routine "fgets()" returned an error and SQR ends the program run.
002501	Unknown INPUT datatype: type={ char number integer date }	Correct the syntax.
002502	INPUT STATUS= must reference #variable.	Correct the syntax.
002503	Unknown qualifier for INPUT.	Correct the syntax.
002506	Too long. Maximum `01 characters.	The response to the INPUT statement was too long. Re-enter the data. `01 = Maximum characters allowed
002507	Incorrect. Format for floating point number: [+/-]99.99[E99]	Invalid number was entered for an INPUT request. Re-enter the data.
002508	Incorrect. Format for integer: [+/-]999999	Invalid integer was entered for an INPUT request. Re-enter the data.
002510	A format mask can only be specified when TYPE=DATE is used.	Correct the syntax.
002511	The format mask cannot be stored in a date variable.	Correct the syntax.
002512	The input variable type does not match the TYPE qualifier.	Correct the syntax.
002513	Number too large for INTEGER. Valid range is -2147483648 to 2147483647.	The number was too large to be stored as an integer. Values are from -2147483648 to 2147483647. Re-enter the data.

Error Number	Error Message	Suggestion/Interpretation
002514	<p>Enter a date in one of the following formats:</p> <p>MM/DD/YYYY [HH:MI[:SS[.NNNNN N]] [AM PM]]</p> <p>MM-DD-YYYY [HH:MI[:SS[.NNNNN N]] [AM PM]]</p> <p>MM.DD.YYYY [HH:MI[:SS[.NNNNN N]] [AM PM]]</p> <p>SYYYMMDD[HH24[MI[SS[N NNNN]]]]</p>	The date cannot be blank. Enter a date in one of the specified formats.
002515	`01 required user interaction but user interaction was disabled by the -XI command line flag.	The specified command required user interaction, but user interaction was disabled by the -XI command line flag. `01 = Name of the command
002600	LOAD-LOOKUP table `01' has not been defined.	Add a LOAD-LOOKUP command. `01 = Load lookup table name
002601	Missing value for `01= in LOAD-LOOKUP.	Correct the syntax. `01 = Name of missing required parameter
002602	Bad value for `01= in LOAD-LOOKUP.	Correct the syntax. `01 = Name of the parameter
002603	LOAD-LOOKUP `01= cannot reference a variable in the Setup section.	Either move the LOAD-LOOKUP command from the Setup section or remove the variable reference. `01 = Name of the parameter
002604	LOAD-LOOKUP names must be unique.	Give each LOAD-LOOKUP array a unique name.

Error Number	Error Message	Suggestion/Interpretation
002605	Cannot compile SQL for LOAD-LOOKUP table ``01'.	The database server returned an error while trying to compile the SQL statement needed to process the LOAD-LOOKUP command. Check the column and table names. Also check the WHERE= clause for errors. `01 = Load lookup table name
002606	Could not set up cursor for LOAD-LOOKUP table ``01'.	The database server returned an error while trying to compile the SQL statement needed to set up the LOAD-LOOKUP command. Verify the column and table names. Review the WHERE= clause for errors. `01 = Load lookup table name
002607	Problem executing the cursor for LOAD- LOOKUP table ``01'.	The database server returned an error while trying to execute the SQL statement needed to process the LOAD-LOOKUP command. `01 = Load lookup table name
002609	Integers only allowed in numeric lookup keys.	Correct the source line.
002610	Numeric lookup keys must be <= ``01 digits.	Correct the source line. `01 = maximum length supported
002611	Bad return fetching row from database in LOAD-LOOKUP table ``01'.	The database server returned an error while fetching the data. `01 = Load lookup table name
002613	Loading ``01' lookup table ...	This message can be inhibited by using the QUIET argument on the LOAD-LOOKUP command. `01 = Name of the load lookup table `02 = Number of rows loaded

Error Number	Error Message	Suggestion/Interpretation
002615	Warning: `01 duplicate keys found in `02' lookup table.	This message can be inhibited by using the QUIET argument on the LOAD-LOOKUP command. `01 = Number of duplicate keys `02 = Name of the load lookup table
002616	LOAD-LOOKUP `01= must reference a numeric variable or literal.	Correct the source line. `01 = Name of the parameter
002617	LOAD-LOOKUP `01= must reference a string variable or literal.	Correct the source line. `01 = Name of the parameter
002618	LOAD-LOOKUP `01= variable `02' has not been defined.	Correct the source line. `01 = Name of the parameter `02 = Name of the undefined variable
002619	LOAD-LOOKUP cannot support `01 rows; maximum is `02.	Reduce the ROWS= value. `01 = ROWS= value `02 = Maximum value allowed
002620	`01 command not allowed with -XL option in effect.	Either use the #IF command to conditionally compile the program when -XL is being used or do not execute this SQR report with the -XL option. `01 = SQR command
002700	Line to stop erasing for 'NEW-PAGE' is larger than the page depth.	Correct the source line.
002800	'ON-BREAK' not appropriate for numeric data.	The ON-BREAK argument to the PRINT command may be used only for text fields. Move the #numeric variable to a \$string variable, and then print the \$string variable.
002801	SET= and LEVEL= must be >= zero when indicated.	Correct the source line.

Error Number	Error Message	Suggestion/Interpretation
002802	Cannot use old style PROCEDURE= with BEFORE= or AFTER=.	Correct the syntax.
002803	Out of ON-BREAKS; use -Mfile to increase ONBREAKS.	Use the -Mfile flag on the command line to specify a file containing an entry that increases the currently defined value.
002804	SET= must be same for all ON- BREAKs in Select.	All the ON-BREAKS in a query must belong to the same SET. Use SET= to differentiate between ON- BREAKs in different queries. Correct the source line.
002805	ON-BREAK with BEFORE or AFTER must be inside Select.	Correct the source line.
002806	SAVE= must be a \$string variable.	Correct the syntax.
002900	Record :types are FIXED, VARY or FIXED_NOLF (default is VARY).	Correct the syntax.
002901	STATUS variable for `01 must be #Numeric.	Correct the syntax. `01 = SQR command affected
002902	OPEN missing required qualifiers: RECORD={rec_len} FOR- READING FOR- WRITING FOR- APPEND	Correct the syntax.
002903	Too many external files opened; maximum is `01.	Reduce the number of open external files needed by the program. `01 = Maximum number of open external files supported by this version of SQR
002904	File number already opened.	Verify the program logic.

Error Number	Error Message	Suggestion/Interpretation
002905	Cannot open file `01' AS `02.(`03): `04	SQR stops. `01 = Filename `02 = File number `03 = System error code `04 = System error message
002906	Cannot close file `01. (`02): `03	SQR stops. `01 = File number `02 = System error code `03 = System error message
002907	Problem closing user file(s) at the end of run.	This message may indicate system problems.
002908	Warning: Cannot CLOSE file `01 -- file not opened.	While not an error, this message indicates a problem with your SQR code. `01 = File number
003000	PAGE-NUMBER strings too long.	The pre-and post-PAGE-NUMBER strings must be less than 74 characters. Correct the source line.
003100	Cannot find document marker referenced in POSITION command.	Defines the specified @ marker in a BEGIN- DOCUMENT paragraph. Verify that the @ marker names are spelled correctly.
003101	Only 'COLUMNS nn...' allowed after document marker in POSITION command.	Correct the syntax.
003200	Specified file number not opened for reading.	Files must be opened for reading in order to use the READ command with them. Correct the program logic.
003201	Line `01: Error reading the file. (`02): `03	`01 = Program line number `02 = System error code `03 = System error message

Error Number	Error Message	Suggestion/Interpretation
003202	Specified file number not opened for writing.	Files must be opened for writing in order to use the WRITE command with them. Correct the program logic.
003203	Line `01: Error writing the file. (`02): `03	`01 = Program line number `02 = System error code `03 = System error message
003204	Length of variables exceeds record length.	The total of the lengths indicated in the command must be less than the RECORD= argument used on the OPEN command. Search for a typographical error or recalculate the RECORD= value.
003205	Numeric binary transfer allowed with FIXED or FIXED_NOLF records only.	By default, all files are opened in VARY (variable length) mode, thus prohibiting the transfer of numeric binary data. Add the:FIXED or FIXED_NOLF option to the RECORD= argument on the appropriate OPEN command.
003206	Command not complete.	Correct the syntax.
003207	File number must be a numeric literal, variable, or column.	Correct the syntax.
003208	Missing required length in READ command.	Correct the syntax.
003209	Bad length for READ or WRITE command.	Correct the syntax.
003210	\$String or #numeric variables required for READ.	Correct the syntax.
003211	#Numeric variables and literals must have :length of 1, 2 or 4 bytes.	Correct the syntax.
003212	#Numeric variables and literals on CDC may only have :length of 1 or 3 bytes.	Correct the syntax.

Error Number	Error Message	Suggestion/Interpretation
003213	:Length not allowed for \$date variables, length of 18 is assumed.	Correct the syntax.
003300	Unknown qualifier for STOP.	Correct the syntax.
003301	Program stopped by user request.	This is an informational message.
003400	Wrap not appropriate for numeric data.	The WRAP argument to the PRINT command may be used only for text fields. Move the #numeric variable to a \$string variable first, and then print the \$string variable.
003401	Max `01 chars/line for reverse WRAP.	Reduce the number of characters specified. `01 = Maximum number of characters supported by this version of SQR.
003402	Max `01 chars/line for WRAP with ON= or STRIP=	Reduce the number of characters specified. `01 = Maximum number of characters supported by this version of SQR
003403	Bad <number> in WRAP qualifier.	The number inside the angled brackets must be a valid number (1 - 255). Correct the source line.
003404	Missing '>' in WRAP qualifier.	A leading "<" in the ON= or STRIP= qualifier indicates that a numeric value is following, which must be ended by a closing ">". Correct the source line.
003405	The value for `01' (`02) must be `03 0.	The value specified for the specified qualifier is invalid. Correct the program logic. `01 = Qualifier name `02 = Value encountered `03 = Relation to zero (<,<=,=,>=,>)

Error Number	Error Message	Suggestion/Interpretation
003500	PUT, GET or ARRAY- xxxx command incomplete. Required word missing.	Correct the syntax.
003501	Did not find end of literal.	The ending quote character (') was not found at the end of the literal. Add the ending quote character.
003502	Literal too long.	Literal strings can be up to 256 characters long. Break up the literal into smaller pieces and combine using the LET command.
003503	Unknown variable type.	Variable names must begin with \$, #, or &. Correct the source line.
003504	Cannot find 'array_name (#element)'.	The element number was not specified. Correct the source line.
003505	'(#Element)' variable not found for array.	Each GET or PUT command must indicate the element or row number to access in the array. Correct the source line.
003506	Array specified not defined with CREATE-ARRAY.	Use the CREATE-ARRAY command to define each array before referencing that array in other commands. Verify that the array names are spelled correctly.
003507	Bad element reference for array (#variable 123).	The element number is larger than the number of rows defined in the CREATE-ARRAY command. Review the program logic to make sure that the element number was not inadvertently changed.
003508	Did not find ending ')' for field.	The "occurs" number for an array field is missing a right parenthesis. Correct the source line.
003509	Field not defined in array: `01	Verify that there are no misspelled field names in the CREATE-ARRAY command. `01 = Undefined field name

Error Number	Error Message	Suggestion/Interpretation
003510	More variables than fields specified in array command.	The ARRAY command must not have more variables listed to the left of the array name than there are matching fields defined for the array. Check against the CREATE-ARRAY command.
003511	More variables in command than fields in array.	The ARRAY command must not have more variables listed to the left of the array name than there are matching fields defined for the array. Check against the CREATE-ARRAY command.
003512	Only numeric variables and fields allowed with array arithmetic commands.	The ARRAY-ADD, ARRAY-SUBTRACT, ARRAY-MULTIPLY, and ARRAY-DIVIDE commands may have only numeric variables or literals as the source fields. Move the string data into a #numeric variable and then reference the #numeric variable.
003513	GET can only be used with \$string or #numeric variables.	You can move array fields only into \$string variables or #numeric variables. Correct the source line.
003514	PUT and GET variables must match array field types.	When moving data into or out of arrays, the source or destination variables must match the array fields in type. CHAR fields can be stored into/from strings, NUMBER fields into/from numeric variables. Check the CREATE-ARRAY command.
003515	More fields than variables found in array command.	The ARRAY command must not have more variables listed to the left of the array name than there are matching fields defined for the array. Check against the CREATE-ARRAY command.
003516	Too many arrays defined; maximum is `01.	Reduce the number of arrays needed by the program. `01 = Maximum number of arrays supported by this version of SQR

Error Number	Error Message	Suggestion/Interpretation
003517	Missing '=specifier' in qualifier: `01	Correct the syntax. 01 = Name of missing required parameter
003518	Duplicate array name: `01	Change the name of the array. `01 = Array name in question
003519	Too many fields defined; maximum is `01.	Reduce the number of fields. `01 = Maximum number of fields allowed per array
003520	Missing 'type' in CREATE-ARRAY FIELD= `01	Correct the syntax. `01 = The name of the field
003521	Duplicate FIELD name: `01	Change the name of one of the fields. `01 = The name of the field
003522	Optional :nn for FIELD must be between 1 and 64K.	Correct the source line.
003523	CREATE-ARRAY FIELDS :type must be one of the following: `01	Correct the syntax.
003525	Missing NAME= in CREATE-ARRAY.	Correct the syntax.
003526	Missing or incorrect SIZE= in CREATE-ARRAY.	Correct the syntax.
003527	Missing FIELD= statements in CREATE-ARRAY.	Correct the syntax.
003528	Array dimensioned too large for PC in CREATE-ARRAY.	On 32-bit systems, the maximum allocation that can be made is 65520 characters. The array as specified would exceed this limit. Reduce the number of entries.

Error Number	Error Message	Suggestion/Interpretation
003529	Missing or invalid initialization value for field `01.	Correct the syntax. 01 = Name of the field
003600	Missing 'ask' variable name.	Correct the syntax.
003601	Out of substitution or #DEFINE variables; use -Mfile to increase SUBVARS.	Use the -Mfile flag on the command line to specify a file containing an entry that increases the currently defined value.
003603	WARNING: Substitution variables do not vary when saved with run-time.	This is an informational message.
003605	No substitution variable entered.	The C routine "fgets()" returned an error and SQR ends the program run.
003700	Did not find end of paragraph: `01	The END-paragraph command to match the specified paragraph is missing. Correct the source file. `01 = BEGIN-paragraph in question
003701	Invalid command.	Verify the spelling of the command.
003702	Command not allowed in this section: `01	Correct the syntax. `01 = Offending command name
003703	Paragraph not allowed inside procedure.	The BEGIN-paragraph command is not allowed here. Review your SQR code for a misplaced paragraph.
003704	Missing procedure name.	Correct the syntax.
003705	Extra argument found.	Correct the syntax.
003706	Missing Comma.	Correct the syntax.

Error Number	Error Message	Suggestion/Interpretation
003707	Bad Argument List.	The DO or BEGIN-PROCEDURE command has an error in its argument list, possibly extra characters after the final right parentheses. Correct the source line.
003708	Empty Argument.	The DO or BEGIN-PROCEDURE command has an error in its argument list, possibly two commas in a row inside the parentheses. Correct the source line.
003709	Only \$string and #number variables allowed for BEGIN-PROCEDURE parameters.	Correct the syntax.
003710	Unknown argument type.	An argument in a DO or BEGIN-PROCEDURE command is incorrect. Verify the spelling of the variable types.
003711	Indicate :\$string or :#number returned values in BEGIN-PROCEDURE only.	Correct the syntax.
003712	Missing).	Correct the syntax.
003713	`01 paragraph not allowed with -XL option in effect.	Either use the #IF command to conditionally compile the program when -XL is being used or do not execute this SQR report with the -XL option. `01 = Name of the BEGIN-paragraph
003714	Bad database connection number.	The -Cnn value must be a non-zero value. Correct the source line.
003715	Did not find end of paragraph: `01 (No 'from...' clause found.)	Correct the source code. `01 = BEGIN-command in question
003716	Error in SQL statement.	The database server has determined that the SQL statement is in error. The actual error text from the server follows this message. Correct the SQL statement.

Error Number	Error Message	Suggestion/Interpretation
003717	Extra characters after expression continuation.	Remove the extra characters after the dash.
003718	Did not find end of expression.	An expression in a SELECT list must end with either a &column variable or a position parameter "(Row,Col,Len)". Correct the source line.
003719	Columns names and expressions must be unique or be given unique pseudonyms (&name).	You are trying to select the same &column name more than once. Change the assigned &column name by using an alias after the name. Columns retrieved from the database are assigned names by prepending an "&" to the beginning of the name.
003720	Bad number specified for 'LOOPS=' on 'BEGIN-SELECT; Maximum is 32767'.	If your program logic requires that you stop processing after more than 32767 rows have been retrieved, you could count the rows manually and use the EXIT-SELECT command to break out of the SELECT loop.
003721	Bad param found on 'BEGIN-SELECT' line; Format is: BEGIN-SELECT [DISTINCT] [-Cnn] [-Bnn] [LOOPS=nn] [ON-ERROR=procedure[(arg1[,argi]...)]]	(DB2) Correct the syntax.
003721	Bad param found on 'BEGIN-SELECT' line; Format is: BEGIN-SELECT [DISTINCT] [-Cnn] [LOOPS=nn] [ON-ERROR=procedure[(arg1[,argi]...)]]	(Informix) Correct the syntax.
003721	Bad param found on 'BEGIN-SELECT' line; Format is: BEGIN-SELECT [DISTINCT] [-Cnn] [LOOPS=nn] [ON-ERROR=procedure[(arg1[,argi]...)]] [-DB=database]	(ODBC) Correct the syntax.

Error Number	Error Message	Suggestion/Interpretation
003721	Bad param found on 'BEGIN-SELECT' line; Format is: BEGIN-SELECT [DISTINCT] [-Cnn] [-Bnn] [LOOPS=nn] [ON-ERROR=procedure[(arg1[,argi]...)]]	(Oracle) Correct the syntax.
003721	Bad param found on 'BEGIN-SELECT' line; Format is: BEGIN-SELECT [DISTINCT] [-Cnn] [-XP] [LOOPS=nnn] [ON-ERROR=procedure[(arg1[,argi]...)]]	(Sybase) Correct the syntax.
003722	Could not set up cursor.	An error occurred while trying to compile the SQL statement. Review any \$string variable references. Correct the SQL statement or use the ON-ERROR= option to trap the error during the program run.
003723	Problem executing cursor.	An error occurred while trying to execute the SQL statement. Review the \$string variable references. Correct the SQL statement or use the ON-ERROR= option to trap the error during the program run.
003724	Could not exit query loop.	The database command to cancel the query returned an error. Try running the SQR program again. The error could be related to a network or server problem. If the error persists, contact your system administrator.
003725	Bad return fetching row from database.	The database returned an error status for the last row that was fetched. This is commonly due to the buffer not being large enough. If selecting expressions, make sure that the length of the first expression is adequate for all rows selected.
003726	Literal in SQL expression missing closing quote.	Literals must be surrounded by single quotes ('). To embed a quote within a literal use two single quotes in sequence ("). Correct the source line.

Error Number	Error Message	Suggestion/Interpretation
003727	SQL expression not ended, perhaps parentheses not balanced.	An expression in a SELECT list must end with either a &column variable or a position parameter "(Row,Col,Len)". Correct the source line.
003728	SQL expression not ended, perhaps missing &name.	An expression in a SELECT list must end with either a &column variable or a position parameter "(Row,Col,Len)". Correct the source line.
003729	SQL expression is missing &name or has unbalanced parentheses.	An expression in a SELECT list must end with either a &column variable or a position parameter "(Row,Col,Len)". Correct the source line.
003730	Incorrect arguments for BEGIN-SQL: [-Cnn] [ON-ERROR=procedure[(arg1[,argi]...)]]	(DB2) Correct the syntax.
003730	Incorrect arguments for BEGIN-SQL: [-Cnn] [ON-ERROR=procedure[(arg1[,argi]...)]]	(Informix) Correct the syntax.
003730	Incorrect arguments for BEGIN-SQL: [-Cnn] [-NR] [ON-ERROR=procedure(arg1[,argi]...)] [-DB=]	(ODBC) Correct the syntax.
003730	Incorrect arguments for BEGIN-SQL: [-Cnn] [ON-ERROR=procedure[(arg1[,argi]...)]]	(Oracle) Correct the syntax.
003730	Incorrect arguments for BEGIN-SQL: [-Cnn] [-XP] [ON-ERROR=procedure[(arg1[,argi]...)]]	(Sybase) Correct the syntax.
003731	Did not find 'END- SQL' after 'BEGIN- SQL'.	Correct the source file.
003732	ON-ERROR= for 'BEGIN-SQL' in SETUP section must be STOP, WARN or SKIP.	Correct the syntax.

Error Number	Error Message	Suggestion/Interpretation
003733	Could not create procedure for SQL.	(Sybase) SQR could not create a stored procedure for the SQL statement. The most likely cause for failure is that the user name you are using to run the report under does not have the proper privileges. Either grant the user CREATE PROCEDURE privilege or use the -XP command line option to inhibit SQR from creating temporary stored procedures for SQL statements.
003734	Could not compile SQL.	Correct the SQL statement or use the ON-ERROR= option to trap the error during the program run.
003735	Could not execute SQL.	An error occurred while trying to compile the SQL statement. Correct the SQL statement or use the ON-ERROR= option to trap the error during the program run.
003736	Please use BEGIN-SELECT - END-SELECT section for SELECT statements.	(Oracle, Informix, ODBC) Correct the source code.
003737	Bad fetch buffer count.	(Oracle, Sybase) The -B flag specifies an illegal value. Correct the source code.
003738	Report interrupted by request.	This is an informational message.
003741	Dynamic column must be \$string variable.	Correct the syntax.
003742	Dynamic column missing `01'.	Correct the syntax. `01 = Missing character
003743	Dynamic columns must have a &pseudonym.	Correct the syntax.
003744	&pseudonym =type must be 'char', 'number', or 'date'.	Correct the syntax.

Error Number	Error Message	Suggestion/Interpretation
003745	Only a variable name may be between the `01' and `02' characters.	Correct the syntax. `01 = Leading character `02 = Trailing character
003746	When dynamic columns are used all non-dynamic columns and expressions must be defined with &name=type.	Add &name=type to all expressions and non-dynamic columns.
003747	When the table name is dynamic each column and expression must be defined with &name=type.	Add &name=type to all expressions and non-dynamic columns.
003800	Too many document paragraphs; maximum is `01.	There are too many BEGIN-DOCUMENT paragraphs. Reduce the number of DOCUMENT paragraphs needed by the program. `01 = Maximum number supported by this version of SQR
003801	Too many document markers; maximum is `01.	There are too many BEGIN-DOCUMENT paragraphs. Reduce the number of DOCUMENT paragraphs needed by the program. `01 = Maximum number supported by this version of SQR
003802	Duplicate document marker.	Specify a unique name for the the document marker.
003803	Did not find 'END- DOCUMENT' after 'BEGIN-DOCUMENT'.	The BEGIN-DOCUMENT paragraph must end with END-DOCUMENT. Correct the source code.
003900	EXECUTE command is incomplete.	Correct the syntax.
003901	Bad -Cnn connection number for EXECUTE.	The -Cnn value must be a nonzero value. Correct the source line.
003902	@#Return_status must be #numeric (missing #).	Correct the source line.

Error Number	Error Message	Suggestion/Interpretation
003903	Missing '=' after `01.	Correct the source line. `01 = The parameter in question
003904	Unknown variable type.	Variable names must begin with \$, #, or &. Correct the source line.
003905	OUT[PUT] variables for EXECUTE may only be \$variable or #variable.	Correct the syntax.
003906	The only EXECUTE option is WITH RECOMPILE.	Correct the syntax.
003907	You must EXECUTE ... INTO &columns.	Correct the syntax.
003908	Unknown datatype for EXECUTE...INTO &columns.	Verify the spelling of the data type. If the data type is correct, then contact customer technical support so SQR can be updated.
003909	EXECUTE...INTO &columns must be unique.	The &column name assigned to the column must be unique throughout the report. Specify a unique name for the column.
003910	Missing (length) for datatype in EXECUTE.	Correct the source line.
003911	Datatype should not have (length) in EXECUTE.	Correct the source line.
003912	DO= in EXECUTE requires INTO... variables.	Correct the syntax.
003913	Could not EXECUTE stored procedure.	Record the database error message displayed with this message. If needed, contact your system administrator.

Error Number	Error Message	Suggestion/Interpretation
003914	Bad return fetching row from database.	Record the database error message displayed with this message. If necessary, contact your system administrator.
003915	Could not set up EXECUTE cursor.	The database server returned an error while trying to compile the SQL statement needed to set up the EXECUTE command.
004000	Result #variable or \$variable or '=' missing in expression.	The LET command is not properly formatted. Correct the source line.
004001	Expression too complex.	The expression is either too long or is too deeply nested. Break the expression into smaller expressions.
004002	Parentheses unbalanced in expression.	A left or right parenthesis is missing. Correct the source line.
004003	Too many variables; maximum is `01.	Break the expression into smaller expressions. `01 = Maximum number supported by this version of SQR
004004	Empty expression.	The expression is invalid. Correct the source line.
004005	Extra comma in expression.	An argument is missing after a comma in the expression. Correct the source line.
004006	Unknown operator `01'. Do you mean `02' ?	The concatenation operator is . Correct the source line.
004007	Too many &column forward references in expression; maximum is `01.	The expression contains too many forward references. Break the expression into smaller expressions. `01 = Maximum number supported by this version of SQR

Error Number	Error Message	Suggestion/Interpretation
004008	Unknown function or variable in expression: `01	The specified function is not an SQR built-in function nor does it exist in the user-modifiable file UFUNC.C. Verify that the function names are spelled correctly. `01 = Function name
004009	Function `01' missing parentheses.	All functions in an expression must be followed by their arguments enclosed in parentheses. Correct the source line.
004010	Empty parentheses or expression.	A pair of parentheses were found with nothing inside them. Remove the () in question from the source line.
004011	User function `01' has incorrect number of arguments.	Look at the file UFUNC.C to determine the correct number and type of arguments required for the specified function. `01 = User function name
004012	Function `01' has incorrect number of arguments.	Correct the syntax of the function. Functions are described under the LET command. `01 = SQR function name
004013	Missing operator in expression.	Correct the source line.
004014	Operator `01' missing argument.	Correct the syntax of the function. Functions are described under the LET command. `01 = Operator
004015	Function `01' missing argument.	Correct the syntax of the function. Functions are described under the LET command. `01 = SQR function name
004016	Function or operator `01' missing arguments.	Correct the syntax of the function. Functions are described under the LET command. `01 = SQR function name

Error Number	Error Message	Suggestion/Interpretation
004017	User function `01' requires character argument.	Look at the file UFUNC.C to determine the correct number and type of arguments required for the specified function. `01 = User function name
004018	User function `01' requires numeric argument.	Look at the file UFUNC.C to determine the correct number and type of arguments required for the specified function. `01 = User function name
004019	User function `01' requires \$string variable.	Look at the file UFUNC.C to determine the correct number and type of arguments required for the specified function. `01 = User function name
004020	User function `01' requires #numeric variable.	Look at the file UFUNC.C to determine the correct number and type of arguments required for the specified function. `01 = User function name
004021	User function `01' has incorrect argument type list. Must be of: c,n,C,N	The UFUNC.C file has a bad definition for the specified function. Correct the UFUNC.C program file, recompile UFUNC.C, and recreate the SQR executable. `01 = User function name
004022	User function `01' missing arguments.	Look at the file UFUNC.C to determine the correct number and type of arguments required for the specified function. `01 = User function name
004023	User function `01' has incorrect return type. Must be c or n.	The UFUNC.C file has a bad definition for the specified function. Correct the UFUNC.C program file, recompile UFUNC.C, and recreate the SQR executable. `01 = User function name

Error Number	Error Message	Suggestion/Interpretation
004024	'isnull' requires a &column, \$string or \$date argument.	#numeric variables cannot be NULL. Correct the source line.
004025	'nvl' requires a &column, \$string or \$date as its first argument.	#numeric variables cannot be NULL. Correct the source line.
004026	Function or operator `01' requires character argument.	Correct the source line. `01 = Function or operator
004027	Function or operator `01' requires numeric argument.	Correct the source line. `01 = Function or operator
004028	IF or WHILE expression must return logical result.	The expression used must evaluate a statement that will be TRUE or FALSE. Correct the source line.
004029	Attempting division by zero in expression.	The expression tried to divide a number by zero. Use the COND() function to determine whether the divisor is zero; then divide by something else (for example, 1).
004030	Attempting division by zero with '%'. %'	An attempt was made to divide a number using the " %" operator. Use the COND() function to determine whether the divisor is zero; then divide by something else (for example, 1).
004031	The number used with '%' (`01) is out of range.	The "%" operator works only with integers. Correct the program logic. `01 = Maximum value allowed
004032	User function has unknown return type -- expecting n or c -- need to recompile Run-Time file?	SQR detected an error while processing a user defined function. If you are running an .sqt file, it probably needs to be recompiled because the user function has changed its definition. If you are running an .sqr file, you need to correct the UFUNC.C program file, recompile UFUNC.C, and recreate the SQR executable.

Error Number	Error Message	Suggestion/Interpretation
004033	In user function use C type with allocated string to change \$variable.	SQR detected an error while processing a user defined function. Correct the UFUNC.C program file, recompile UFUNC.C, and recreate the SQR executable.
004034	Could not find array `01' in ARRAY function.	Verify the spelling of the array name. `01 = Array name
004035	Could not find array field `01' in ARRAY function.	Verify the spelling of the array field name. `01 = Array field name
004036	Math error in expression (usually over- or under-flow).	Most of the SQR mathematical built-in functions have a corresponding C library routine. One returned an error. Break the expression into discrete expressions in order to identify the function that caused the error.
004037	Error executing expression.	Record the steps leading up to the error and contact technical support.
004038	Out of space while processing expression; Use -Mfile to increase EXPRESSIONSPACE.	The expression requires more temporary string storage than is currently allocated. Use the -Mfile flag on the command line to specify a file that contains an entry that increases by a greater value than is currently defined.
004039	`01' assumed to be a variable name, not an expression.	Warning message. `01 = Expression in question
004040	The array `01' has not been defined.	Define the array using the CREATE-ARRAY command. `01 = Array name
004041	The field `01' is not valid for array `02'.	Correct the source code. `01 = Field name `02 = Array name

Error Number	Error Message	Suggestion/Interpretation
004042	The array reference ``01' has an incorrect number of parameters specified.	Correct the source code. `01 = Array name
004043	The array reference ``01' requires numeric parameters for the element and occurs arguments.	Correct the source code. `01 = Array name.
004045	Function or operator ``01' requires date argument.	Correct the source code. `01 = Array name
004046	Incompatible types between expression and variable.	Correct the source code.
004047	The field ``01' is must be 'char' or 'float'.	Correct the source code. `01 = Field name
004048	Function or operator ``01' must be a string or date argument.	Correct the source line. `01 = Function or operator
004100	Use 'print' command to format data outside SELECT query.	You must precede PRINT command arguments (WRAP, ON-BREAK.) with an explicit PRINT command when outside of a BEGIN-SELECT paragraph. Correct the source line.
004101	Cannot find required parameter.	Correct the syntax.
004102	Bad number found.	A command expecting a numeric literal or :#numeric variable reference found an illegal number definition or a reference to a string variable or column. Correct the source line.
004103	Cannot find required numeric parameter.	Correct the syntax.
004104	Cannot find placement parameters.	The position qualifier "(Row,Col,Len)" was not found. Search for a missing parentheses.

Error Number	Error Message	Suggestion/Interpretation
004105	Placement parameter incorrect.	The "Row", "Column" or "Length" fields are invalid or ill-formed. Correct the source line.
004106	Invalid second function on line.	An SQR command used as a qualifier for a primary command (for example, PRINT) is incorrect. Correct the source line.
004107	Second function must be FORMAT type.	The PRINT command may have format command qualifiers such as WRAP, CENTER, or FILL. Other qualifier commands are not permitted.
004108	Missing operator =, <, >, ...	Correct the source line.
004109	Invalid operator.	Correct the source line.
004110	Missing variable.	Correct the syntax.
004111	Please give this expression a &pseudonym.	Expressions selected in a BEGIN-SELECT paragraph should be given an &Name or be followed by a print position "(Row,Col,Len)". Correct the source line.
004112	Wrong variable type.	Correct the syntax.
004113	Command incomplete, expected `01'.	Correct the syntax. `01 = What was expected
004114	Expecting `01', found `02'.	Correct the syntax. `01 = What was expected `02 = What was encountered
004115	Unknown command or extra parameters found (missing quotes?).	Correct the syntax.
004116	Duplicate references to parameter `01'.	Correct the syntax. `01 = Duplicated parameter

Error Number	Error Message	Suggestion/Interpretation
004117	Unexpected equal sign found with `01'.	Correct the syntax. `01 = Parameter name
004118	Qualifier `01' cannot be used with the following qualifiers:	Correct the syntax. `01 = Qualifier name
004119	Expecting numeric column, found string column.	Correct the syntax.
004120	Date variables (`01) cannot be used with this command.	Correct the syntax. `01 = Parameter name
004200	Page width and depth must be > 0 and < 32767.	The values specified with the PAGE-SIZE command are out of specified range. Specify legal values.
004201	Page buffer must be < 65536 on PC SQR.	The maximum page buffer allocation on a PC is 65536. The Page-Depth * Page-Width cannot exceed this value. Reduce the Page-Depth or Page-Width.
004202	Cannot generate line printer output for this report because position qualifier(s) may be out of range. If you are running this report, specify PRINTER: {HP,EH,HT,PS,WP} for graphical printer output.	The report output cannot be generated for a Line Printer. If your report was designed for a graphics printer, specify - PRINTER: {HP,EH,HT,PS,WP} for graphical printer output.
004300	Missing end of placement (...) in SHOW.	The placement parameter is ill-formed. Correct the source line.
004301	Bad (...) location in SHOW.	Screen positions must be valid numbers. Correct the source line.
004302	Missing literal or variable name to EDIT in SHOW.	A literal or variable name must immediately precede the EDIT, NUMBER, MONEY, or DATE keywords.

Error Number	Error Message	Suggestion/Interpretation
004303	Missing edit mask in SHOW.	The word EDIT must be followed by a valid edit mask. Correct the source line.
004304	Only string variable allowed for dynamic edit mask.	Dynamic edit masks may only be stored in \$variables. Correct the line.
004305	Unknown option for SHOW.	Correct the syntax.
004400	Program too large; use -Mfile to increase PROGLINES.	The SQR program contains too many SQR command lines. Use the -Mfile flag on the command line to specify a file containing an entry that increases the currently defined value.
004401	Out of param storage; use -Mfile to increase PROGLINEPARS.	The SQR program contains too many SQR command line parameters. Use the -Mfile flag on the command line to specify a file containing an entry that increases the currently defined value.
004402	Out of string storage; use -Mfile to increase STRINGSPACE.	The space allocated to hold the static string variables ('...') has been used. Use the - Mfile flag on the command line to specify a file containing an entry that increases the currently defined value.
004403	Out of variables; use - Mfile to increase VARIABLES.	There are too many variables (string, numeric), literals and database columns. Use the -Mfile flag on the command line to specify a file containing an entry that increases the currently defined value.
004405	Out of forward &column or \$variable references; use -Mfile to increase FORWARDREFS.	A forward referenced variable is a variable that is referenced before it is defined. Use the -Mfile flag on the command line to specify a file containing an entry that increases the currently defined value.

Error Number	Error Message	Suggestion/Interpretation
004406	Number `01 not allowed.	Use a different value. `01 = Internal number
004407	Referenced variables not defined:	References were made to column variables (&var) that are not defined in the program. The list of variable names follows this message.
004500	Out of Print positions; use -Mfile to increase POSITIONS.	A print position is the "(Row,Col,Len)" parameter. Use the -Mfile flag on the command line to specify a file containing an entry that increases the currently defined value.
004501	Use '+' and negate variable for reverse relative placement.	The use of "-#variable" is not legal here. Negate the #variable value and use "+#variable".
004503	Fixed line placement #variable must be > 0. Use relative positioning, (+#line,10,0).	Correct the source line as indicated.
004504	Fixed column placement #variable must be > 0. Use relative positioning, (5,+#col,0).	Correct the source line as indicated.
004505	Length placement #variable must be >= 0.	The length field cannot be a negative value. Correct the source line.
004600	CODE not appropriate for numeric data.	The CODE qualifier to the PRINT command may only be used for text fields. Move the "#Variable" to a "\$Variable" first and then print the "\$Variable".
004601	Unknown option for GRAPHIC command: BOX, HORZ-LINE, VERT-LINE or FONT	Correct the syntax.

Error Number	Error Message	Suggestion/Interpretation
004602	GRAPHIC BOX out of bounds. Row: `01, Column: `02, Width: `03, Depth: `04	SQR ends the program run. `01 = Row `02 = Column `03 = Width `04 = Depth
004603	GRAPHIC VERT- LINE out of bounds. Row: `01, Column: `02, Length: `03	SQR ends the program run. `01 = Row `02 = Column `03 = Length
004604	GRAPHIC HORZ- LINE out of bounds. Row: `01, Column: `02, Length: `03	SQR ends the program run. `01 = Row `02 = Column `03 = Length
004700	Cannot open the program file: `01' (`02): `03	Depends on the system error message. `01 = Name of the program file `02 = System error code `03 = System error message
004701	Cannot logon to the database.	The connectivity information is either incorrect or the database server is unavailable. Verify the connectivity information and the server availability.
004702	Line found outside paragraph.	All commands must be within BEGIN-... END-statements. Correct the source code.
004703	Cannot close the program file. (`01): `02	Depends on the system error message. `01 = System error code `02 = System error message
004704	#ENDIF not found for #IF.	Missing an #ENDIF to complete conditional compilation. Correct the source code.

Error Number	Error Message	Suggestion/Interpretation
004705	Program line too long; maximum is `01.	Break the program line into smaller lines. `01 = Maximum line length supported by this version of SQR
004706	Substitution variable {`01} would cause this line to exceed the maximum line length of `02 characters.	The substitution variable value would cause this line to exceed the maximum line size. Break the program line into smaller lines. `01 = Name of the substitution variable `02 = Maximum line length supported by this version of SQR
004707	No value found for substitution variable: {`01}	An empty value was found for the substitution variable. Verify the spelling of the substitution variable. `01 = Name of the substitution variable
004708	#ELSE without preceding #IF.	Missing an #IF or #IFDEF or #IFNDEF to begin conditional compilation. Correct the source code.
004709	#ENDIF without preceding #IF.	Missing an #IF or #IFDEF or #IFNDEF to begin conditional compilation. Correct the source code.
004710	#IF's nested too deeply; maximum is `01.	Reduce the number of nested #IF directives. `01 = The maximum depth supported by this version of SQR
004711	#INCLUDE files nested too deeply; maximum is `01.	Reduce the number of nested #INCLUDE directives. `01 = The maximum depth supported by this version of SQR

Error Number	Error Message	Suggestion/Interpretation
004712	Include file name too long; Modify -I flag.	The combined -I directory name with the #INCLUDE file name exceeds the maximum length permitted for a complete path name. Verify the spelling of both the -I command flag and the #INCLUDE filename.
004713	Cannot open the #INCLUDE file: `01' (^02): `03	`01 = Include file name `02 = System error code `03 = System error message
004714	Cannot close the #INCLUDE file: `01' (^02): `03	`01 = Include file name `02 = System error code `03 = System error message
004716	'BEGIN-REPORT' command not found in program.	This section is required for all reports. Correct the source code.
004717	Cannot open the report output file: `01' (^02): `03	`01 = Output file name `02 = System error code `03 = System error message
004719	Cannot logoff the database.	The database server returned an error while trying to log off from the database. SQR ends the program run.
004720	Cannot open the run-time file: `01'. (^02): `03	SQR ends the program run. `01 = Run-Time file name `02 = System error code `03 = System error message
004721	Cannot close the run-time file. (^01): `02	SQR ends the program run. `01 = System error code `02 = System error message
004722	Error reading the run-time file. (^01): `02	SQR ends the program run. `01 = System error code `02 = System error message

Error Number	Error Message	Suggestion/Interpretation
004723	Run time file must be recreated for this version of SQR.	The runtime file was created by a earlier version of SQR and is incompatible with the current version. Recreate the .sqt (runtime) file.
004724	The -XL option cannot be specified with this run-time file because access to the database is required.	Do not use the -XL option.
004725	Cannot open cursor.	The database server returned an error indicating that a new database cursor or logon could not be completed. See the error message from the database server.
004726	Cannot create procedure for SQL statement.	(Sybase) SQR could not create a stored procedure for the SQL statement. The most likely cause for failure is that the user name you are running the report under does not have the proper privileges. Either grant the user CREATE PROCEDURE privilege or use the -XP command line option to inhibit SQR from creating temporary stored procedures for SQL statements.
004727	Error writing the runtime file. (^01): ^02	^01 = System error code ^02 = System error message
004728	You must specify a Partitioned Data Set name and member to build a .sqt(member) run-time file. Could not create the run-time file.	(MVS) Use the proper format to specify the name of the .sqt file.
004729	Cannot find inactive database cursor. Program too large.	(Oracle, DB2) The program has too many concurrent database cursors. Reduce the complexity of the program.
004730	Run-time saved in file: ^01	This is an informational message. ^01 = Name of the .sqt file created

Error Number	Error Message	Suggestion/Interpretation
004735	Unknown variable type encountered in run-time file: `01	SQR ends loading the runtime file. `01 = Variable type
004736	Unexpected End-Of-File while processing the run-time file.	SQR ends loading the runtime file.
004737	Cannot load the run- time file because it was built for the `01database and `02 is built for the `03 database.	SQR ends loading the runtime file. `01 = Database name from runtime file `02 = SQR image name `03 = Database that SQR is built for
004738	'END-REPORT' not paired with 'BEGIN- REPORT'.	Correct the source code.
004739	'END-PROGRAM' not paired with 'BEGIN- PROGRAM'.	Correct the source code.
004743	#INCLUDE filename must be enclosed in quotation marks.	Correct the syntax.
004744	#INCLUDE command format is: #Include 'filename'.	Correct the syntax.
004745	Array field (`01.`02) specification exceeds the PC 64K limit.	Reduce the size of the field requirements. `01 = Array name `02 = Field name
004746	Layout `01' specifications exceeds the PC 64K limit.	The layout is too large for the 32-bit version of SQR to handle. `01 = Layout name
004747	The SQT file is corrupted and cannot be processed.	SQR ends loading the runtime file.

Error Number	Error Message	Suggestion/Interpretation
004748	The user function '^01' needs to be defined as entry ^02 in the user function table. It requires a definition of: Return Type = ^03' Arg Count = ^04 Arg Types = ""^05"	The SQT file requires that the specified user function be defined. ^01 = User function name ^02 = Entry in the user function table ^03 = Return type ^04 = Argument count ^05 = Argument types
004749	An attempt was made to move ^01 characters into ^02'. The maximum allowed is ^03 characters.	An attempt was made to move too much data into an SQR string variable. ^01 = Number of characters to be moved ^02 = Variable name ^03 = Maximum characters allowed
004802	PRINTER TYPE must be HTML, HPLASER-JET, POSTSCRIPT, or LINEPRINTER.	Correct the syntax.
004805	Both BEFORE-BOLD and AFTER-BOLD must be specified.	Correct the syntax.
004807	Unknown DECLARE qualifier.	Correct the syntax.
004900	Out of dynamic SQL arguments [\$...]; use -Mfile to increase DYNAMICARGS.	Use the -Mfile flag on the command line to specify a file containing an entry that increases the currently defined value.
004901	Date variables (^01) cannot be used in BEGIN-SQL or BEGIN-SELECT paragraphs.	Correct the source code. ^01 = Variable name
005000	Report ^01' heading section size exceeds the page depth.	Reduce the size of the heading or increase the page depth.
005001	Report ^01' footing location must be less than the page depth.	Reduce the size of the footing or increase the page depth.

Error Number	Error Message	Suggestion/Interpretation
005002	Check 'BEGIN- HEADING' commands: Discovered 2nd page-initialization while heading in progress.	The BEGIN-HEADING procedure either caused an overflow of the current page or it issued a command that caused a page eject to occur. Review any procedure invoked by the BEGIN-HEADING section to ensure that the commands do not overflow the page or cause a page eject.
005003	Check 'BEGIN-FOOTING' commands; perhaps number of footing lines is too small. Discovered 2nd page- write while footing in progress.	The BEGIN-FOOTING procedure either caused an overflow of the current page or it issued a command that caused a page eject to occur. Review any procedure invoked by the BEGIN-FOOTING section to ensure that the commands do not overflow the page or cause a page eject.
005004	Attempt to execute the `01 command while processing the `02 section.	Change the SQR program logic to prevent the command from executing while the specified section is active. `01 = Command name `02 = Section name
005005	Report `01' already has been assigned a `02 section.	Correct the source code. `01 = Report name `02 = Duplicated section name
005006	You cannot define more than one default `01' section.	Correct the source code. `01 = Duplicated section name
005007	Report `01' has overlapping heading and footing sections.	Correct the source code. `01 = Report name
005008	TOC `01' already has been assigned a `02 section.	Correct the source code. `01 = Table of Contents name `02 = Duplicated section name

Error Number	Error Message	Suggestion/Interpretation
005100	'IF', 'WHILE', 'EVALUATE' commands nested too deeply; maximum is `01.	Reduce the nested commands. `01 = Maximum depth allowed by this version of SQR
005101	'BREAK' found outside 'WHILE' or 'EVALUATE' statement.	The BREAK command is valid only in the context of a WHILE or EVALUATE statement. Correct the source code.
005102	Out of Break commands; Use -Mfile to increase BREAKS.	This is the number of BREAK commands allowed per EVALUATE command. Use the -Mfile flag on the command line to specify a file containing an entry that increases the currently defined value.
005103	END-WHILE found without matching 'WHILE'.	Correct the source code.
005104	'IF' or 'EVALUATE' command not completed before 'END-WHILE'.	Correct the syntax.
005105	'ELSE' found without matching 'IF'.	ELSE can be used only within the context of an IF command. Correct the source code.
005106	Single 'ELSE' found inside 'WHILE' or 'EVALUATE' statement.	ELSE can be used only within the context of an IF command. Correct the source code.
005107	Only one 'ELSE' allowed per 'IF'.	Rewrite the source code to use nested IF statements.
005108	Found 'END-IF' without matching 'IF'.	Each IF command must have a matching END-IF command. Correct the source code.

Error Number	Error Message	Suggestion/Interpretation
005109	'WHILE' or 'EVALUATE' command not completed before 'END-IF'.	You are missing a closing END-WHILE or END-EVALUATE command before END- IF. IF, WHILE, and EVALUATE statements can be nested, but they cannot cross each other's boundaries. Each inner statement must be complete before a closing statement is ended. Correct the source code.
005110	EVALUATE statements nested too deep; maximum is `01.	Reduce the number of nested statements. `01 = Maximum depth supported by this version of SQR
005111	'WHEN' found outside 'EVALUATE' clause.	WHEN may be used only in the context of an EVALUATE clause. Correct the source code.
005112	'IF' or 'WHILE' not completed before 'WHEN' statement.	Correct the syntax.
005113	Out of When commands; Use -Mfile to increase WHENS.	Use the -Mfile flag on the command line to specify a file containing an entry that increases the currently defined value.
005114	Incorrect types for comparison. Both must be of the same type (string, numeric or date).	Correct the source line.
005115	'When-other' found outside 'Evaluate' statement.	WHEN can be used only in the context of an EVALUATE statement. Correct the source code.
005116	'IF' or 'WHILE' not ended before 'WHEN- OTHER' command.	Correct the syntax.
005117	Only one 'WHEN- OTHER' allowed per 'EVALUATE'.	Correct the syntax.
005118	Found 'END-EVALUATE' without matching 'EVALUATE'.	Each EVALUATE command must have a matching END-EVALUATE command. Correct the source code.

Error Number	Error Message	Suggestion/Interpretation
005119	'IF' or 'WHILE' command not completed before 'END-EVALUATE'.	Correct the syntax.
005120	'WHEN-OTHER' must be after all 'WHEN's.	Correct the syntax.
005121	No 'WHEN's found inside 'EVALUATE' statement.	Correct the syntax.
005122	'IF', 'EVALUATE' and 'WHILE' statements cannot cross sections or paragraphs.	These commands must be contained within a single section or paragraph. Correct the source code.
005200	Did not find '>' after <....	A leading left angled bracket (<) indicates that you are beginning an decimal value representing a character, which must be ended by a right angled bracket (>). Correct the source line.
005201	Bad character in <...>.	Numbers in angled brackets (<>) must be between 1 and 255. Correct the source line.
005202	Bad number in <...>.	Numbers in angled brackets (<>) must be between 1 and 255. Correct the source line.
005203	<...> string is too long; maximum is `01 characters.	Reduce the length of the string. If this is not possible, use a PRINT-DIRECT command in a BEGIN-REPORT or END-REPORT procedure. `01 = Maximum number of characters supported by this version of SQR
005300	Did not find '=' after qualifier: `01	Correct the syntax. `01 = Qualifier name
005301	Qualifier `01' requires a numeric value.	Correct the syntax. `01 = Qualifier name

Error Number	Error Message	Suggestion/Interpretation
005302	Incorrect value for qualifier `01'. Valid values are:	Correct the source line. `01 = Qualifier name
005303	Invalid qualifier `01'. Valid qualifiers are:	Correct the source line. `01 = Qualifier name
005304	Qualifier `01' requires a numeric literal, variable, or column.	Correct the source line. `01 = Qualifier name
005305	Qualifier `01' references a numeric variable that has not been defined.	Correct the source line. `01 = Qualifier name
005306	Qualifier `01' requires a string literal, variable, or column.	Correct the source line. `01 = Qualifier name
005307	List not terminated.	Correct the syntax.
005308	Missing comma in list.	Correct the syntax.
005309	Required argument `01' was not specified.	Correct the source line. `01 = Qualifier name
005310	Qualifier `01' has already been specified.	Correct the source line. `01 = Qualifier name
005311	Qualifier `01' requires a string literal.	Correct the source line. `01 = Qualifier name
005312	Qualifier `01' requires a list of values: (val [,val]...).	Correct the source line. `01 = Qualifier name
005313	Qualifier `01' requires a integer value.	Correct the source line. `01 = Qualifier name
005314	Invalid character in variable name `01'.	Correct the source line. `01 = Invalid character

Error Number	Error Message	Suggestion/Interpretation
005315	Qualifier `01' references a string variable that has not been defined.	Correct the source line. `01 = Qualifier name
005316	Qualifier `01' uses an invalid Unit-Of-Measure suffix. Valid suffixes are: dp pt mm cm in	Correct the source line. `01 = Qualifier name
005400	Second page write attempted while writing current page. Check BEFORE- PAGE, AFTER-PAGE procedures.	Review any procedure invoked by the BEFORE-PAGE or AFTER-PAGE procedures to ensure that the commands do not overflow the page or cause a page eject.
005402	String cannot be placed on page: `01 -- placement specified is out of range. (^02,^03,^04)	Ensure the values are within the page limits. `01 = Text value `02 = Row `03 = Column `04 = Length
005403	Error writing the output file. (^01): `02	`01 = System error code `02 = System error message
005404	Cannot open the Postscript startup file: `01 (^02): `03	`01 = Name of the file `02 = System error code `03 = System error message
005405	SQR trial copy exiting after `01 pages.	`01 = Number of pages.
005406	Exiting after requested number of test pages (^01).	`01 = Number of pages.
005408	Program stopped by user request.	This is an informational message.
005500	Cannot set parse_only option.	(Sybase) The DB-Library routine dbsetopt() returned an error. This error should never occur. Contact technical support.

Error Number	Error Message	Suggestion/Interpretation
005501	Cannot reset parse_only option.	(Sybase) The DB-Library routine dbclropt() returned an error. This error should never occur. Contact technical support.
005502	Cannot drop SQR generated stored procedure: `01.	(Sybase) See the database server error message that was also output. This error should never occur. Contact technical support. `01 = Stored procedure name
005503	Cannot use `01 datatype as bind variable.	(Sybase) Use another database column. `01 = The database datatype.
005504	Unknown datatype for bind variable: `01 Cannot create stored procedure.	(Sybase) Contact technical support. `01 = Unknown database datatype
005505	SQL too large to create stored procedure.	(Sybase) The size of the SQL text needed to create the stored procedure is too large for SQR to process. Add the -XP option to the BEGIN-SQL or BEGIN-SELECT command.
005506	SQR's EXECUTE command not available for this version of Sybase.	(Sybase) Some early versions of Sybase SQL Server or Microsoft SQL Server do not support Remote Procedure Calls (RPCs). Update your database server.
005507	Could not add param to remote procedure call.	(Sybase) A DB-Library routine returned an unexpected error. See the error message from the database.
005508	The number of EXECUTE...INTO &columns does not match the procedure.	(Sybase) Verify the definition for the stored procedure you are referencing.
005509	Incorrect number of INTO &columns defined in EXECUTE.	(Sybase) Verify the definition for the stored procedure you are referencing.

Error Number	Error Message	Suggestion/Interpretation
005510	Error converting OUTPUT Sybase type for EXECUTE.	(Sybase) The DB-Library routine dbconvert() failed to convert the data from the stored procedure. Contact technical support.
005511	Number of OUTPUT parameters from EXECUTE is incorrect.	(Sybase) Verify the definition for the stored procedure you are referencing.
005512	Missing default database name for USE.	(Sybase) Correct the syntax.
005512	Missing default database name for USE.	(ODBC) Could not connect to the specified datasource.
005513	You may only specify 'USE db' once, before any SQL statements are executed.	(Sybase) Only one USE command is allowed in a report. Place the SETUP section at the beginning of the SQR report.
005515	Undefined variable referenced in - DB flag: `01	(ODBC) Verify that there are no misspellings. `01 = Variable name
005523	Database commit failed.	The database command to perform a commit returned an error. Try running the SQR program again. The error could be related to a network or server problem. If the error persists, contact your system administrator.
005524	Cannot close database cursor.	The database command to close the database cursor returned an error. Try running the SQR program again. The error could be related to a network or server problem. If the error persists, contact your system administrator.
005528	DB2 SQL `01 error `02 in cursor `03:	(DB2) `01 = Routine name `02 = Error code `03 = SQR cursor number

Error Number	Error Message	Suggestion/Interpretation
005528	INFORMIX SQL `01 error `02 (ISAM: `03) in cursor `04: `05	(Informix) `01 = Routine name `02 = Error code `03 = ISAM code `04 = SQR cursor number `05 = Error message from database
005528	ODBC SQL `01 error `02 in cursor `03: `04	(ODBC) `01 = Routine name `02 = Error code `03 = SQR cursor number `04 = Error message from database
005528	ORACLE `01 error `02 in cursor `03: `04	(Oracle) `01 = Routine name `02 = Error code `03 = SQR cursor number `04 = Error message from database
005528	Sybase `01 error in cursor `02: `03	(Sybase) `01 = Routine name `02 = SQR cursor number `03 = Error message from database
005532	System 10 files are missing.	(Sybase) Contact your system administrator.
005533	Not a System 10 SQL Server.	(Sybase) The CT-Library version of SQR can only connect to a System 10 server. Use the DB-Library version of SQR to connect to a version earlier than System 10 server.
005534	SQL too long for PREPARE/DECLARE; maximum `01 characters.	(DB2) The SQL statement is too large. `01 = Maximum number of characters supported by this version of SQR

Error Number	Error Message	Suggestion/Interpretation
005536	Unknown error message number: `01.	(DB2) `01 = Error message number
005537	Empty error message returned from system for error number: `01.	(DB2) `01 = Error message number
005538	Invalid SELECT statement; COMPUTE clauses are not supported.	(Sybase) The select statement contains a COMPUTE clause that is not supported.
005539	Could not connect to datasource specified in -db variable: `01'.	(ODBC) Could not connect to the specified datasource.
005540	Not connected to a database, database access is not allowed.	The SQR program is no longer connected to a database. Commands that access the database can no longer be used. This situation can occur if the CONNECT fails and the ON-ERROR option was used.
005543	Specify the Oracle DLL name in the sqr.ini file in [Environment:Oracle] section for ORACLE_DLL entry, such as ORACLE_DLL=orant71.dll	(Oracle) SQR was unable to load the Oracle DLL. By default, SQR looks first for ociw32.dll or the DLL specified by the ORACLE_DLL entry in the [Environment:Oracle] section of the sqr.ini file. If that DLL could not be loaded, then SQR attempts to load orant71.dll.
005600	GETWRD: Word too long; maximum is `01.	Reduce the length of the "word". `01 = Maximum size of a "word" supported by this version of SQR
005700	Cannot call SQR recursively.	SQR cannot be called recursively. This error can only occur if a User Function from either UFUNC.C or UCALL.C calls the sqr() routine. Do not call sqr() from a UFUNC.C or UCALL.C routine.

Error Number	Error Message	Suggestion/Interpretation
005701	Too many SQR command line arguments; maximum is `01	To pass more than this number of arguments, use a @file argument file containing one argument per line. `01 = Maximum number supported by this version of SQR.
005702	Log file name specified is too long.	Reduce the length of the log file name.
005703	Error opening the SQR log file: `01' (`02): `03	`01 = Name of the file `02 = System error code `03 = System error message
005704	Missing program name.	The name of the program file was not found on the command line. The program name must be the first parameter on the command line.
005705	Program file name specified is too long.	Reduce the length of the program file name.
005707	Error opening the -E error file: `01' (`02): `03	`01 = Name of the file `02 = System error code `03 = System error message
005708	Cannot find `01 in SQRDIR, PATH or \SQR.	The specified file cannot be located in any of the directories pointed to by the mentioned environment variables or default directories. Make sure the "file" is present in one of the locations searched. `01 = File name
005709	`01 environment variable is not defined.	As of version 2.5, the environment variable SQRDIR must be defined. `01 = Name of the environment variable

Error Number	Error Message	Suggestion/Interpretation
005710	`01 path too long.	The length of the directory path plus the length of the file name to be opened is too long for SQR to handle. Reduce the length of the directory path. `01 = Environment variable name
005711	Bad number in -T test flag.	The number specified must be > zero. Correct the value.
005716	Unknown flag on command line: `01	Correct the syntax. `01 = Unknown command line flag
005720	Error opening tty. (^01): `02	(Tru64, UNIX/Linux) This is an error that should never occur during normal operations. Record the steps leading up to the error and contact your system administrator. `01 = System error code `02 = System error message
005721	Error with 'ioctl()'. (^01): `02	(Tru64, UNIX/Linux) This is an error that should never occur during normal operations. Record the steps leading up to the error and contact your system administrator. `01 = System error code `02 = System error message
005722	Error reading tty. (^01): `02	(Tru64, UNIX/Linux) This is an error that should never occur during normal operations. Record the steps leading up to the error and contact your system administrator. `01 = System error code `02 = System error message
005723	Error closing tty. (^01): `02	(Tru64, UNIX/Linux) This is an error that should never occur during normal operations. Record the steps leading up to the error and contact your system administrator. `01 = System error code `02 = System error message

Error Number	Error Message	Suggestion/Interpretation
005724	Bad number in -B flag.	(Oracle, Sybase) The number specified must be greater than zero. Correct the value.
005734	No program name given.	The report name must be the first command line argument.
005737	Unknown printer type specified with - PRINTER: switch.	The printer type can be EH, HT, LP, HP, PS, or WP. WP is valid only with PC/Windows.
005738	Database name needs to be included with - DB switch.	(ODBC) Could not connect to the specified datasource.
005738	Database name needs to be included with - DB switch.	(Sybase) Supply the database name.
005739	Too many -F switches; maximum is `01.	Reduce the number of -F switches. `01 = Maximum number allowed
005740	-F and outfile name are required with DDN or DD style SQR {program} name.	(MVS) Correct the JCL stream.
005741	Attempting to use SQR {program} file for outfile.	(MVS) Correct the JCL stream.
005742	Attempt to invoke viewer (using WinExec) failed; error code = `01.	(Windows) `01 = System error code
005743	Unknown numeric type specified with - DNT: switch.	Correct the command line.
005744	-DNT:Decimal precision (`01) is out of range (`02 - `03).	Correct the command line. `01 = Specified precision `02 = Minimum allowed `03 = Maximum allowed

Error Number	Error Message	Suggestion/Interpretation
005745	The specified default numeric type `01 = `02' is invalid.	Correct the sqr.ini file entry. `01 = Entry `02 = Value
005746	The decimal precision `01 = `02' is out of range (`03 - `04).	Correct the sqr.ini file entry. `01 = Entry `02 = Value `03 = Minimum allowed `04 = Maximum allowed
005747	The following error(s) occurred while processing the [01] section from the sqr.ini file.	See the error message(s) that follow. `01 = Name of the section
005750	The -Burst switch is not properly formatted.	The "Burst" command line flag is not properly formatted.
005751	The -Burst switch cannot be used with the -NOLIS switch.	The "Burst" command line flag cannot be specified when the -NOLIS command line flag is also specified.
005752	The -Burst switch requires either the - Printer:HT or - Printer:EH switch to be specified.	The "Burst" command line flag is applicable only when HTML code is produced. You must specify either the -PRINTER:HT or -PRINTER:EH switch.
005753	The -Burst:S and - Burst:T switches can only be used against an SPF file which was generated with SQR v4.1 and above.	The "Burst" command line flag can only be specified when processing a SPF file that was generated by SQR v4.1 and above. Older SPF files do not contain the proper information that permits bursting.
005754	The -Burst switch caused no output to be generated.	The "Burst" command line flag was specified with a set of page ranges that prevented any output to be created. Change the page ranges.
005900	Bad number in -`01	(Windows) Specify a valid number. `01 = Command line option

Error Number	Error Message	Suggestion/Interpretation
005901	Bad filename in ^01	(Windows) Specify a valid file name. ^01 = Command line option
005902	Bad directory in ^01	(Windows) Specify a valid directory path. ^01 = Command line option
005903	Cannot access the @ parameter file (^01): ^02	(Windows) Depends on the system error message. ^01 = System error code ^02 = System error message
005904	The argument list is too long; maximum is ^01.	(Windows) To pass more than this number of arguments, use a @file argument file containing one argument per line. ^01 = Maximum number supported by this version of SQR.
005905	Cannot open the report file (^01): ^02	(Windows) Depends on the system error message. ^01 = System error code ^02 = System error message
005906	Invalid filename entered.	(Windows) Re-enter with a valid file name.
006000	Error writing the printer file. (^01): ^02	This is an error that can occur during normal operations due to the system environment (for example, file locking and permissions). Record the steps leading up to the error and contact your system administrator. ^01 = System error code ^02 = System error message

Error Number	Error Message	Suggestion/Interpretation
006001	Error reading the printer file. (^01): `02	This is an error that can occur during normal operations due to the system environment (for example, file locking and permissions). Record the steps leading up to the error and contact your system administrator. `01 = System error code `02 = System error message
006002	Cannot open the printer file: ^01 (^02): ^03	This is an error that can occur during normal operations due to the system environment (for example, file locking and permissions). Record the steps leading up to the error and contact your system administrator. `01 = Name of the file `02 = System error code `03 = System error message
006003	Unexpected End-Of- File while processing the printer file.	The file might be corrupted. Try to recreate the .spf file. If the error persists, contact technical support.
006004	Encountered unknown SPF code (^01) while reading the printer file.	The file might be corrupted. Try to recreate the .spf file. If the error persists, contact technical support. `01 = Unknown SPF code
006100	Duplicate chart (^01).	Each chart must be given a unique name. `01 = Chart name
006101	Unknown chart (^01).	Chart could not be found. `01 = Chart name
006102	Number of chart data- array columns specified (^01) exceeds the number of array columns (^02).	Correct the source code. `01 = Number of data-array columns `02 = Number of array columns

Error Number	Error Message	Suggestion/Interpretation
006103	Number of chart data- array rows specified (^01) exceeds the number of array rows (^02).	Correct the source code. ^01 = Number of data-array rows ^02 = Number of array rows
006104	Too many pie segments (^01). Max is ^02.	Correct the source code. ^01 = Number of segments ^02 = Maximum allowed segments
006105	Chart module is not initialized.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006106	XY charts may have only numeric columns.	Correct the syntax.
006107	The 3rd column in the data array must be a character column to specify USE-3RD- DATA-COLUMN.	Correct the syntax.
006108	Invalid chart size or placement.	Correct the source code.
006120	INTERNAL: Bad chart index from stack (^01).	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. ^01 = Chart index
006121	INTERNAL: Unknown SQR BG Interface message (^01).	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. ^01 = Message code
006122	INTERNAL: Unsupported Graftsman chart type (^01).	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. ^01 = Chart type

Error Number	Error Message	Suggestion/Interpretation
006123	INTERNAL: Unsupported pie-explode setting (^01).	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. ^01 = Setting value
006124	INTERNAL: Unsupported tick-mark placement (^01).	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. ^01 = Placement value
006125	Grafsmann interface message (^01) not supported.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. ^01 = Message code
006126	Unrecognized return code (^01) from Grafsmann command message (^02).	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. ^01 = Return code ^02 = Message code
006127	Cannot fit Chart/Image into the current page. Position: (^01, ^02) Size: (^03, ^04)	Correct the source code. SQR ends the program run. ^01 = Row ^02 = Column ^03 = Width ^04 = Depth
006128	Check coordinate values.	Correct the syntax.
006140	Duplicate image (^01).	Images must be given unique names. ^01 = Image name
006141	Unknown image (^01).	Image name could not be found. ^01 = Image name

Error Number	Error Message	Suggestion/Interpretation
006142	Cannot open image file (^01). (^02): ^03	`01 = Name of the file `02 = System error code `03 = System error message
006150	INTERNAL: Bad image index from stack (^01).	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. `01 = Image name
006200	This report has already been defined.	Each report must be given a unique name.
006201	This layout has already been defined.	Each layout must be given a unique name.
006202	This printer has already been defined.	Each printer must be given a unique name.
006203	The values for '01' must be > 0.	Correct the syntax. `01 = Qualifier name
006204	Qualifiers '01' and '02' are mutually exclusive.	Correct the syntax. `01 = Qualifier name `02 = Qualifier name
006205	Qualifier '01' is not applicable with a 'default' printer.	Correct the syntax. `01 = Qualifier name
006206	The list must contain report names or ALL.	Correct the syntax.
006207	'ALL' must be specified by itself.	Correct the syntax.
006208	No report name was specified.	Correct the syntax.
006209	No layout name was specified.	Correct the syntax.

Error Number	Error Message	Suggestion/Interpretation
006210	No printer name was specified.	Correct the syntax.
006211	The name cannot be 'ALL'.	Correct the syntax.
006212	The name can only contain characters [0-9 A-Z _ -].	Correct the syntax.
006213	Report '01' is referenced by multiple '02' printers.	Correct the syntax. `01 = Report name `02 = Printer type
006214	Qualifier '01' is not allowed with a '02' printer.	Correct the syntax. `01 = Qualifier name `02 = Printer type
006215	The value for '01' must be '02 0.	Correct the syntax. `01 = Qualifier name `02 = Relation to zero (<,<=,=,>=,>)
006216	Report '01' does not exist.	Correct the syntax. `01 = Report name
006217	The report name can be a string literal, variable, or column.	Correct the syntax. `01 = Report name
006218	Referenced layouts not defined:	A list of undefined layouts follows this message.
006219	Referenced reports not defined:	A list of undefined reports follows this message.
006220	Referenced printers not defined:	A list of undefined printers follows this message.

Error Number	Error Message	Suggestion/Interpretation
006221	The following SQR commands (listed below) cannot be used when any of the following NEW SQR commands are also used in the same report:	Correct the syntax.
006224	No printer type was specified.	Correct the syntax.
006225	Incorrect value for printer type. Valid values are:	Correct the syntax. A list of valid printer types follows this message.
006226	Attempt to execute the `01 command while processing the `02 procedure.	SQR ends the program run. `01 = SQR command `02 = Procedure name
006227	Incorrect value for 'paper-size'. Specify the actual dimensions or one of the following names:	Correct the syntax. A list of valid predefined paper-size names follows this message.
006228	Referenced TOC (Table Of Contents) not defined:	A list of undefined Table of Contents follows this message.
006229	This TOC (Table Of Contents) has already been defined.	Each Table of Contents must be given a unique name.
006230	The list must contain TOC (Table of Contents) names or ALL.	Correct the syntax.
006231	The TOC (Table Of Contents) entry cannot be positioned given the LEVEL (`01) and INDENTATION (`02) values.	The Table of Contents entry will not fit given the specified level and current indentation values. `01 = Specified LEVEL= value `02 = Current INDENTATION= value
006232	`01 command not allowed while generating the Table of Contents.	The specified command cannot be used while the Table of Contents is being generated. `01 = SQR command

Error Number	Error Message	Suggestion/Interpretation
006233	<p>The TOC (Table of Contents) entry "A" cannot be processed because the existing entry "B" is positioned below it.</p> <p>A: Line = `01, Level = `02, Text = `03'</p> <p>B: Line = `04, Level = `05, Text = `06'</p>	<p>Correct the program logic to eliminate the conflict between the two TOC (Table of Contents) entries.</p> <p>`01 = A: Line number</p> <p>`02 = A: Level value</p> <p>`03 = A: Text value</p> <p>`04 = B: Line number</p> <p>`05 = B: Level value</p> <p>`06 = B: Text value</p>
006300	Unknown parameter (`01).	<p>Correct the syntax.</p> <p>`01 = Parameter name</p>
006301	Value not valid for parameter (`01).	<p>Correct the syntax.</p> <p>`01 = Parameter name</p>
006302	Invalid option (`02) for parameter (`01).	<p>Correct the syntax.</p> <p>`01 = Parameter name</p> <p>`02 = Option</p>
006303	Parameter (`01) is required, but has not been specified.	<p>Correct the syntax.</p> <p>`01 = Parameter name</p>
006304	Parameter (`01) already specified.	<p>Correct the syntax.</p> <p>`01 = Parameter name</p>
006305	Parameter (`01) does not support &columns.	<p>Correct the syntax.</p> <p>`01 = Parameter name</p>
006306	Parameter (`01) requires equal sign.	<p>Correct the syntax.</p> <p>`01 = Parameter name</p>
006307	Parameter (`01) has an unquoted string.	<p>Correct the syntax.</p> <p>`01 = Parameter name</p>

Error Number	Error Message	Suggestion/Interpretation
006308	Missing part of specification for parameter (^01).	Correct the syntax. ^01 = Parameter name
006309	Parameter (^01) requires literal.	Correct the syntax. ^01 = Parameter name
006310	Parameter (^01) requires valid numeric value.	Correct the syntax. ^01 = Parameter name
006311	Parameter (^01) requires integer value.	Correct the syntax. ^01 = Parameter name
006312	Parameter (^01) does not support type supplied.	Correct the syntax. ^01 = Parameter name
006313	Parameter (^01) requires valid string. Perhaps quote or \$ is missing.	Correct the syntax. ^01 = Parameter name
006314	Parameter (^01) does not accept 'NONE' in this context.	Correct the syntax. ^01 = Parameter name
006315	Parameter (^01) requires proper object name.	Correct the syntax. ^01 = Parameter name
006316	Parameter (^01) requires array name.	Correct the syntax. ^01 = Parameter name
006317	Parameter (^01) does not accept 'AUTOSCALE' in this context.	Correct the syntax. ^01 = Parameter name
006318	Parameter (^01) has improper value list.	Correct the syntax. ^01 = Parameter name
006320	Parameter (^01) does not support relative values.	Correct the syntax. ^01 = Parameter name

Error Number	Error Message	Suggestion/Interpretation
006350	Conversion [(^01) to (^02)] is not supported.	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. ^01 = From type ^02 = To type
006352	INTERNAL: Unsupported option/request (^01) in (^02).	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. ^01 = Option/request code ^02 = Function name
006354	INTERNAL: Unknown data type, (^01).	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. ^01 = Data type
006355	INTERNAL: Unable to retrieve parameter value, (^01).	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. ^01 = Parameter name
006356	INTERNAL: Data type (^02) not valid for parameter (^01).	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. ^01 = Parameter name ^02 = Data type
006357	INTERNAL: Data location (^02) not valid for data type (^01).	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. ^01 = Data location ^02 = Data type

Error Number	Error Message	Suggestion/Interpretation
006358	INTERNAL: Cannot decode string (^01) to index.	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. ^01 = String to decode
006359	INTERNAL: Cannot set bit value (^02) for parameter (^01).	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. ^01 = Parameter name ^02 = Value
006360	INTERNAL: Unknown program state (^01).	This is an internal error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. ^01 = State
006400	Unsupported background color.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006401	Unsupported border color.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006402	Border width out of range.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006403	X position out of range.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006404	Y position out of range.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.

Error Number	Error Message	Suggestion/Interpretation
006405	X size out of range.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006406	Y size out of range.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006407	Unsupported font.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006408	Unsupported font style.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006409	Unsupported font color.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006410	Unsupported horizontal text justification value.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006411	Unsupported vertical text justification value.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006412	Unsupported font path.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006413	Unsupported font rotation.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.

Error Number	Error Message	Suggestion/Interpretation
006414	Font size out of range.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006415	Text line id# out of range.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006416	Unsupported chart type.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006417	Unsupported chart sub-type.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006418	Unsupported chart orientation (not H or V).	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006419	Unsupported perspective (not 2D or 3D).	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006420	Unsupported axis (not X or Y).	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006421	Unsupported axis label data type.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006422	Dataset id# out of range.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.

Error Number	Error Message	Suggestion/Interpretation
006423	Unsupported dataset type.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006424	Unsupported dataset color.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006425	Unsupported dataset line style.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006426	Unsupported dataset fill pattern.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006427	Unsupported dataset marker.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006428	Chart type does not support Y-axis datasets.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006429	Pie-chart segment id# is out of range.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006430	Unsupported pie-segment color.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006431	Unsupported pie-segment border color.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.

Error Number	Error Message	Suggestion/Interpretation
006432	Unsupported pie-segment pattern.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006433	Unsupported pie- segment explode setting.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006434	Command only valid for charts of type 'pie'.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006435	Pie-chart radius out of range.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006436	Pie-chart starting angle out of range.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006437	Unsupported pie-chart fill direction. Must be clockwise or counter-clockwise.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006438	Unsupported pie- segment label position.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006439	Unsupported pie- segment quantity display position.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006440	Unsupported pie- segment per-cent display position.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.

Error Number	Error Message	Suggestion/Interpretation
006441	Unsupported legend style.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006442	Unsupported legend horizontal position.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006443	Unsupported legend vertical position.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006444	Text charts do not support legend.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006445	Number of datasets specified does not match data.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006446	Unsupported axis label position.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006447	Unsupported axis type (not LINEAR or LOG).	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006448	Pie and text charts do not support axis control.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006449, 006450	Unsupported axis min scaling.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.

Error Number	Error Message	Suggestion/Interpretation
006451	Unsupported axis max scaling.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006452	Beginning of tickmarks is after end.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006453	Unsupported tickmark type.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006454	Unsupported grid type.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006455	Unsupported grid color.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006456	Grid line width out of range.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006457	Unable to open grafcap file.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006458	Unsupported grafcap device.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006459	Error in grafcap entry specification.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.

Error Number	Error Message	Suggestion/Interpretation
006460	Unable to open chart output destination.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006461	Internal error during ggDraw.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006462	Improper parameters passed to gscale.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006463	The shared library specified in the grafcap file could not be found.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006464	A function called from the shared library specified in the grafcap file could not be found.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006500	The bar code could not be positioned on the page. Row: `01, Column: `02, Height: `03	Correct the source code. `01 = Row `02 = Column `03 = Height
006501	Unknown BCL error (`01) encountered.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support. `01 = BCL error code
006502	Invalid bar code type (`01): Valid values are from 1 to 15.	Correct the source code. `01 = Bar code type.
006503	The length of the bar code text '01' must be between 1 and 30 characters.	Correct the source code. `01 = Bar code text

Error Number	Error Message	Suggestion/Interpretation
006504	The length of the caption text '01' must be between 1 and 30 characters.	Correct the source code. `01 = Caption text
006505	Invalid printer type (`01): Valid values are from 0 to 13.	Correct the source code. `01 = Printer type
006506	Invalid offset: Valid values are from 0 to 250.	Correct the source code.
006507	Invalid height (`01): Valid values are from 0.1 to 2.0 inches.	Correct the source code. `01 = Height
006508	Invalid checksum: Valid values are from 0 to 2.	Correct the source code.
006509	Invalid pass: Valid values are from 1 to 6.	Correct the source code.
006510	The bar code text '01' is not valid for the type of bar code (`02) selected.	Correct the source code. `01 = Bar code text `02 = Bar code type
006511	Internal error: Could not generate the bar code.	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006512	Internal error: Bar code buffer required too large (>32K).	This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.
006601	Cannot allocate the device context for the default printer.	(Windows) This is an error that can occur due to lack of system resources. Record the steps leading up to the error and contact your system administrator.

Error Number	Error Message	Suggestion/Interpretation
006602	Failed to start printing the document.	(Windows) This is an error that can occur due to lack of system resources or a problem with the printer. Record the steps leading up to the error and contact your system administrator.
006603	New-page (start) failed on page `01.	(Windows) This is an error that can occur due to lack of system resources or a problem with the printer. Record the steps leading up to the error and contact your system administrator. `01 = Page number
006604	New-page (end) failed on page `01.	(Windows) This is an error that can occur due to lack of system resources or a problem with the printer. Record the steps leading up to the error and contact your system administrator. `01 = Page number
006605	End document failed.	(Windows) This is an error that can occur due to lack of system resources or a problem with the printer. Record the steps leading up to the error and contact your system administrator.
006606	Error reading font information from the [Fonts] section in sqr.ini. Using the default font.	(Windows) Correct the [Fonts] section in the sqr.ini file.
006607	Failed to create a brush for shading.	(Windows) This is an error that can occur due to lack of system resources. Record the steps leading up to the error and contact your system administrator.
006608	Failed to select font `01.	(Windows) This is an error that can occur due to lack of system resources. Record the steps leading up to the error and contact your system administrator. `01 = Font name

Error Number	Error Message	Suggestion/Interpretation
006609	Failed to modify font `01.	(Windows) This is an error that can occur due to lack of system resources. Record the steps leading up to the error and contact your system administrator. `01 = Font name
006610	Failed to create a pen that was required to draw a box.	(Windows) This is an error that can occur due to lack of system resources. Record the steps leading up to the error and contact your system administrator.
006611	Failed to create a pen that was required to draw a horizontal line.	(Windows) This is an error that can occur due to lack of system resources. Record the steps leading up to the error and contact your system administrator.
006612	Failed to create a pen that was required to draw a vertical line.	(Windows) This is an error that can occur due to lack of system resources. Record the steps leading up to the error and contact your system administrator.
006613	Failed to open the image bitmap file (`01). (`02): `03	(Windows) This is an error that can occur during normal operations due to the system environment (file locking, permissions). Record the steps leading up to the error and contact your system administrator. `01 = Name of the file `02 = System error code `03 = System error message
006614	The file (`01) does not contain a valid bitmap.	(Windows) Specify a valid bitmap file. `01 = Name of the file
006615	Failed to create the palette for image (`01).	(Windows) This is an error that can occur due to lack of system resources or an invalid bitmap. Record the steps leading up to the error and contact your system administrator. `01 = Name of the file

Error Number	Error Message	Suggestion/Interpretation
006616	Failed to load RLE into memory for image (^01).	(Windows) This is an error that can occur due to lack of system resources. Record the steps leading up to the error and contact your system administrator. ^01 = Name of the file
006617	Failed to convert DIB to DDB for image (^01).	(Windows) This is an error that can occur due to lack of system resources. Record the steps leading up to the error and contact your system administrator. ^01 = Name of the file
006618	Failed to draw the bitmap image (^01).	(Windows) This is an error that can occur due to lack of system resources. Record the steps leading up to the error and contact your system administrator. ^01 = Name of the file
006619	Cannot access the default printer's driver.	(Windows) This is an error that can occur due to lack of system resources or a problem with the printer. Record the steps leading up to the error and contact your system administrator.
006620	Cannot select the charting clip area onto the printers DC.	(Windows) This is an error that can occur due to lack of system resources or a problem with the printer. Record the steps leading up to the error and contact your system administrator.
006621	Cannot select create a metafile required for business graphics.	(Windows) This is an error that can occur due to lack of system resources. Record the steps leading up to the error and contact your system administrator.
006622	Cannot create a region required for business graphics.	(Windows) This is an error that can occur due to lack of system resources. Record the steps leading up to the error and contact your system administrator.

Error Number	Error Message	Suggestion/Interpretation
006623	Cannot create a DC required for business graphics.	(Windows) This is an error that can occur due to lack of system resources. Record the steps leading up to the error and contact your system administrator.
006624	Cannot create a bitmap required for business graphics.	(Windows) This is an error that can occur due to lack of system resources. Record the steps leading up to the error and contact your system administrator.
006625	Business graphics failed while setting up the device (ggWinDevice).	(Windows) This is an error that can occur due to lack of system resources. Record the steps leading up to the error and contact your system administrator.
006626	Cannot draw business graphics.	(Windows) This is an error that can occur due to lack of system resources or due to a damaged LIBSTI.INI file. The LIBSTI.INI file resides in the Windows main directory. Make sure that the GPATH= and IPT= entries point to a valid SQR BINW directory. Record the steps leading up to the error and contact your system administrator.
006700	SQRDIR is not defined.	(Windows) The variable SQRDIR must be defined in the sqr.ini file.
006701	Could not allocate memory while attempting to register the .spf filename extension.	(Windows) This is an error that can occur due to lack of system resources. Record the steps leading up to the error and contact your system administrator.
006702	Could not allocate memory for the page cache.	(Windows) This is an error that should never occur during normal operations. Record the steps leading up to the error and contact technical support.

Error Number	Error Message	Suggestion/Interpretation
006704	Cannot open or read file (^01) (^02): `03	(Windows) This is an error that can occur during normal operations due to the system environment (for example, file locking and permissions). Record the steps leading up to the error and contact your system administrator. `01 = Name of the file `02 = System error code `03 = System error message
006705	File (^01) is not in SPF packet format.	(Windows) The file was not produced by SQR or has been corrupted. `01 = Name of the file
006706	Failed to identify the start of the report (^01).	(Windows) The file was not produced by SQR or has been corrupted. `01 = Name of the file
006707	An invalid seek was made for page `01.	(Windows) This is an internal error which should not occur under normal operations. Contact technical support. `01 = Page number
006708	Too many errors were encountered while processing the file. Processing has been stopped.	(Windows) This is an error that can occur due to lack of system resources. Record the steps leading up to the error and contact your system administrator.
006709	Failed to open the image bitmap file (^01). (^02): `03 This message is displayed only once per SPF file.	(Windows) This is an error that can occur during normal operations due to the system environment (for example, file locking and permissions). Record the steps leading up to the error and contact your system administrator. `01 = Name of the file `02 = System error code `03 = System error message

Error Number	Error Message	Suggestion/Interpretation
006800	`01: Detected internal program error.	This is an internal error that should never occur during normal operation. Record the steps leading up to the error and contact technical support. `01 = Name of the routine
006801	`01: Null Operand Passed as input.	This is an internal error that should never occur during normal operation. Record the steps leading up to the error and contact technical support. `01 = Name of the routine
006802	`01: Decimal Exponent Under/Overflow.	Exponent Under/Overflow: Exponent of decimal number has exceeded the valid boundaries established for the decimal type. Review the documentation for the current upper and lower bounds of a decimal object. `01 = Name of the routine
006803	`01: Decimal to Integer Conversion Under/Overflow.	Integer Under/Overflow: Cannot convert input decimal object into a valid integer number. Decimal object exceeds the established integer boundaries for this machine architecture. Review the magnitude and sign of the decimal object to ensure that it falls within the upper and lower bounds of an integer number. `01 = Name of the routine
006804	`01: Decimal to Float Conversion Under/Overflow.	Floating Point Under/Overflow: Cannot convert input decimal object into a valid floating point number. The decimal object exceeds the established floating point boundaries for this machine architecture. Review the magnitude and sign of the decimal object to ensure that it falls within the upper and lower bounds of a floating point number. `01 = Name of the routine

Error Number	Error Message	Suggestion/Interpretation
006805	`01: Decimal Precision Under/Overflow.	Decimal Precision Under/Overflow: Attempt made to initialize decimal object with an invalid precision. Verify the input precision value against the documented upper and lower boundaries for a decimal object. `01 = Name of the routine
006806	`01: String to Decimal Object Conversion Error.	String To Decimal Conversion Error: The length of input string is greater than the precision of underlying decimal object. Either increase the precision of the decimal object or reduce the size of the input mantissa to match the decimal object precision. `01 = Name of the routine
006807	`01: Truncation/Rounding Error - Outside Valid Range for Decimal Object.	Truncation/Rounding Error: Input truncation or round value is outside the valid range for this decimal object. Ensure that the truncation/round value is greater than or equal to zero and less than the precision of the underlying decimal object. `01 = Name of the routine
006808	`01: Decimal Error: Cannot Divide by Zero.	Decimal Math Divide by Zero Error: Attempt made to divide a decimal object by zero. Ensure that the divisor does not equal zero before attempting to divide. `01 = Name of the routine
006900	There is no default printer set up on your system. Use the Control Panel "Printers" applet to define it.	(Windows) SQR Print requires that a default printer be defined. Use the "Printers" applet in the Control Panel to define one.
007000	The locale `01' is not defined in the sqr.ini file.	Verify the spelling of the locale name or the sqr.ini file. `01 = Locale name
007001	At least one qualifier must be specified.	Correct the source code.

Error Number	Error Message	Suggestion/Interpretation
007002	The value for '01' must be a list of 02 string literals, variables or columns.	Correct the source code. `01 = Qualifier `02 = Number of entities in list
007003	The values for '01' and '02' cannot be the same.	Correct the source code. `01 = Qualifier `02 = Qualifier
007004	The value for '01' (^02) must be a single character which is not in the list: "03".	Correct the source code. `01 = Qualifier `02 = Value `03 = List of invalid characters
007005	The value for `01' (^02) is invalid. Valid values are:	Correct the source code. `01 = Qualifier `02 = Value
007006	The last character of the `01' value (^02) cannot be a digit or the minus sign or the same as either of the separators.	Correct the source code. `01 = Qualifier `02 = Invalid character
007007	The first character of the `01' value (^02) cannot be a digit or the minus sign or the same as either of the separators.	Correct the source code. `01 = Qualifier `02 = Invalid character
00700	The following errors occurred while processing the (^01) locale from the sqr.ini file.	This message precedes error messages encountered while processing the sqr.ini file. `01 = Locale name
007009	The value for `01' cannot be 'DEFAULT' or 'SYSTEM'.	Correct the syntax. `01 = Qualifier
007010	The value for `01' (^02) is not properly formatted: Did not find the '>' for the '<nnn>' construct.	Correct the syntax. `01 = Qualifier `02 = Value

Error Number	Error Message	Suggestion/Interpretation
007011	The value for `01' (`02) is not properly formatted: The value of an '<nnn>' construct must be from 1 to 255.	Correct the syntax. `01 = Qualifier `02 = Value
007012	The default locale (`01) specified in the [`02] section of the sqr.ini file has not been defined.	Correct the syntax. `01 = Locale name `02 = Section name
007013	The value for `01' (`02) must be a list of `03 quoted string literals.	Correct the syntax. `01 = Qualifier `02 = Value `03 = Number of entities in list
007014	The entry (`01 = `02) is not valid.	Correct the sqr.ini entry. `01 = Qualifier from the sqr.ini file `02 = Qualifier's value
007100	The use of an edit mask or the keywords NUMBER, MONEY or DATE is not legal when storing numeric variables.	Correct the source code.
007101	The last keyword is not `01'.	Correct the source code. `01 = Keyword
007102	Incompatible source and destination variable types.	Correct the source code.
007103	The keyword (`01) is not compatible with the variable (`02).	Correct the source code. `01 = Keyword `02 = Variable name
007104	The use of an edit mask or the keyword DATE is not legal if both variables are date variables.	Correct the source code.

Error Number	Error Message	Suggestion/Interpretation
007200	The specified precision (^01) is out of range (^02 - ^03).	Correct the source code. ^01 = Specified precision ^02 = Minimum precision ^03 = Maximum precision
007201	The precision is specified by a value from ^01 to ^02 surrounded by parentheses.	Correct the source code. ^01 = Minimum precision ^02 = Maximum precision
007202	Variable (^01) is not a decimal variable and cannot have a precision associated with it.	Correct the source code. ^01 = Variable name
007203	A string variable name is required here.	Correct the source code.
007204	A numeric variable name is required here.	Correct the source code.
007205	The variable (^01) has already been defined as ^02' and may not be redefined.	Correct the source code. ^01 = Variable name ^02 = Variable type
007206	The variable type has not been specified.	Correct the source code.
007207	This command is only allowed within local procedures.	Correct the source code.
007208	This command must be before all other commands in the procedure.	Correct the source code.
007209	Only string (\$) and numeric (#) variables may be declared.	Correct the source code.
007210	Invalid variable name specified.	Correct the source code.
007211	You cannot declare a global variable from within a procedure.	Correct the source code.

Error Number	Error Message	Suggestion/Interpretation
007400	The specified character is invalid in the current character set.	Correct the program logic.
007401	`01' is not a valid value for the ENCODING environment variable.	The specified encoding scheme is not known by SQR. `01 = ENCODING environment variable setting.
007402	The Double-Byte LET function `01' is not supported in this version of SQR.	The SQT file contains a reference to a LET function, which is not supported by this version of SQR. `01 = LET function name
007403	The Double-Byte SQR command `01' is not supported in this version of SQR.	The SQT file contains a reference to an SQR command, which is not supported by this version of SQR. `01 = SQR command name
007404	Double-Byte .sqt files are not supported by this version of SQR.	The runtime file was created by the double-byte version of SQR and is incompatible with the current version.
007405	The barcode text `01' cannot contain double- byte characters.	Correct the source code. `01 = Bar code text
007501	Using `01 edit mask from (`02) against (`03)	A date edit mask element, which could cause date data to be incorrectly interpreted, was detected. This warning message can be turned off by setting the "OutputTwoDigitYearWarningMsg" entry in the [Default- Settings] section of the sqr.ini file to FALSE. `01 = Edit mask element `02 = Edit mask being used `03 = Value being applied to the edit mask
007601	Cannot access the Java file (`01) (`02): `03	SQR cannot access the required file. `01 = Name of the file `02 = System error code `03 = System error message

Error Number	Error Message	Suggestion/Interpretation
007602	-EH_Scale: value (^01) is out of range (^02 - ^03).	Correct the command line. ^01 = Specified scale ^02 = Minimum allowed ^03 = Maximum allowed
007603	-Printer:EH functionality is not available on this platform.	Enhanced HTML functionality is not available on this platform.
007604	-Printer:PD functionality is not available on this platform.	PDF functionality is not available on this platform.
007701	Did not find end of paragraph: ^01 (No 'end-execute' clause found.)	Correct the source code. ^01 = BEGIN-command in question.
007702	Invalid entry for keyword, ^01=^02'	Correct the source code.
007703	May only specify either PROCEDURE=, or COMMAND=, or GETDATA=, exclusive.	Correct the source code.
007704	Must specify a SCHEMA.	Correct the source code.
007705	Must specify either a PROCEDURE, COMMAND, or GETDATA.	Correct the source code.
007706	CONNECTION ^01' not found. No such connection.	Correct the source code.
007707	The returned set of Procedure parameters (INOUT and OUT) (length = ^01 items) did not include one or more of the specified items.	Stored procedure error.
007708	Encountered a parameter of type ^01'. Valid types are either IN, OUT, or INOUT. If no type is entered, the type defaults to IN.	Stored procedure error.

Error Number	Error Message	Suggestion/Interpretation
007709	The datasource failed to provide the expected return status value. Verify the query metadata.	Datasource error.
007710	The datasource failed to provide the expected number of elements in the return status list.	Datasource error.
007711	Failed to login to the requested datasource (Connection=`01', username=`02'). DETAILS: `03	Logon failed.
007712	The requested rowset (`01) was not available. Verify the query metadata.	Not enough row sets.
007713	Missing or invalid Registry.properties file. Verify that the CLASSPATH includes SQDIR, that SQDIR contains the folder with the Registry.properties file, and that the Registry.properties file is valid.	Incorrect environment setup.
007714	The datasource (`01') does not support the requested capability (`02'). Check the capabilities list for the datasource, located in the Properties folder.	Invalid query for datasource.
007715	Failed to start the Java Virtual Machine (JVM). Possible causes are: missing or invalid jdk files, incorrect CLASSPATH, or insufficient resources.	Incorrect environment setup.
007716	The current rowset (`01) contained no rows. Check the return status and/or metadata for the requested service to determine the cause.	No data.
007717	The query failed. DETAILS: `01	Query failed.
007718	Failure setting property `01'. DETAILS: `02	Property-set failed.

Error Number	Error Message	Suggestion/Interpretation
007719	The value for keyword `01' exceeds the maximum length of `02 characters.	Keyword value too long.
007720	A fatal error occurred while fetching against the current rowset: DETAILS: `01	A failure occurred during row fetch.
007721	Parameter `01 (`02) was passed to the PROCEDURE as data type `03; expected (`04) type `05. Verify the query metadata.	A failure occurred during row fetch.
007722	Invalid query parameter: Reason: `01	Bad procedure parameter.
007723	Too many parameters (= `01) were supplied to the query. Verify the query metadata.	Bad procedure parameter.
007724	Parameter `01 (`02) was passed to the PROCEDURE as type `03; expected type `04. Verify the query metadata.	Bad procedure parameter.
007725	Parameter `01 (`02', JDO-type `03), specified 'NULL', is a required-parameter. Specify a value or variable name.	Bad procedure parameter.
007726	The list-variable parameter to the query is too long. Maximum number of elements is 30.	List too long.
007727	Unable to retrieve metadata for Procedure=`01, Schema=`02. DETAILS: `03	Metadata check failed.
007728	Parameter list type mismatch (#`01, SQR type = `02). The datasource expected a parameter of type `03. Verify the query metadata.	Parameter list mismatch.

Error Number	Error Message	Suggestion/Interpretation
007729	List size mismatch detected while fetching data of type ROW, `01 items, into SQR list- variable, `02 items. Fetching will proceed to the smaller size.	List size mismatch.
007730	Incorrect syntax for BEGIN-SELECT ... FROM. Options are: FROM ROWSETS=... FROM PARAMETER= \$strvar strlit	Bad BEGIN-SELECT syntax.
007731	Attempted to pass as INOUT or OUT a parameter which was of type ROWSET (`01). Use of such parameters is supported as IN only, after which they may be used in a BEGIN- SELECT construct.	Bad parameter keyword.
007732	Attempt to use a scalar SQR variable (`01') to reference a ROWSET procedure parameter (`02'). Use either the keyword 'NULL', or an SQR LIST variable (%var). Verify the query metadata.	Bad proc parameter.
007733	The list of keywords entered to the PARAMETERS keyword must be terminated with a semi-colon.	Bad proc parameter. Correct the source code.
007734	Datasource '01' not found. The Connection being used by this query specifies a data-source which is not listed in the DDO Registry ('02'). DETAILS: `03.	<obsolete>
007735	Missing one or more DDO {fname} .jar files. Verify the location of the original- installation files, and that they are accessible. Error code: `01. Classpath: `02.	<obsolete>
007736	Unable to open Connection ('01') to data-source ('02'). Possible causes: (a) the Declare-Connection specification is invalid, or (b) the datasource is no longer available. DETAILS: `03.	Bad environment.

Error Number	Error Message	Suggestion/Interpretation
007737	Unable to locate one or more entry points in an SQR {fname}.jar file. Verify that the original-installation files have not become corrupted.	Bad environment.
007738	At least one JNI method pointer was lost. This should never occur: record the steps leading up to this failure, and contact Technical Support. DETAILS: Schema='01', Proc='02'.	Bad environment.
007739	Unable to locate query object '01' in the specified schema ('02'). DETAILS: '03.	Bad environment.
007740	Invalid &pseudonym or 'TYPE=' data-type specified for a begin-select column-variable. Valid types are: CHAR, TEXT, DATE, NUMBER.	Correct the syntax.
007741	Illegal attempt to fetch a non-scalar field into a column variable. Correct the query.	Correct the syntax.
007742	The output parameter specified in 'Begin- Select ... From Parameter = '01' is not available. Available parameters: '02.	Bad command.
007743	The output parameter specified in 'Begin- Select ... From Parameter = '01' is not of type ROWSET. Verify the query metadata.	Bad command.
007744	Illegal attempt to assign an SQR variable ('01') of type '02' the value from a DDO object ('03') of type '04'. Verify the query metadata.	<obsolete>
007745	Illegal attempt to assign an SQR column variable ('01') of type '02' the value from a DDO object of type '03'. Verify the query metadata.	<obsolete>

Error Number	Error Message	Suggestion/Interpretation
007746	Failed to locate the requested Rowset (^01) while processing the query. The last available Rowset number is ^02. Verify the query metadata.	Not enough row sets.
007747	The query raised a DDO exception. DETAILS: ^01.	<obsolete>
007748	A BEGIN-SELECT paragraph was coded, but the query returned no Rows.	No data warning.
007749	Invalid syntax for PARAMETERS=(...) statement. Use: PARAMETERS=(%v \$v #v &v NULL SKIP numlit datelit textlit [IN INOUT], ...) All parameters must be specified. Optional parameters which are to be ignored may be specified by the keyword 'NULL' or 'SKIP'. Correct the syntax.	Incorrect syntax.
007750	FATAL: Failure creating Java object.	General failure.
007751	Attempt to create a List variable of size greater than the maximum size of ^01 items.	General failure.
007752	Parameter-list item ^01' is not a member of the parameter list for this Query. Verify the query metadata.	No such input/inout parameter.
007753	Attempt to access List- row (^01) beyond the List size (^02 rows).	Bad list assignment/setup.
007754	Attempt to assign/modify a List row is not compatible with the List definition.	Bad list assignment/setup.

Error Number	Error Message	Suggestion/Interpretation
007755	Attempt to assign a row to a non-existent List variable. Define the List first, using the syntax: let %lname[size] = list(NUMBER DATE TEXT #var \$var [, ...])	Bad list assignment/setup.
007756	Incorrect syntax for List-variable reference. Use: let [\$ #]var = %listname[nlit #var].colname	Bad list assignment/setup.
007757	Alter-connection statement missing 'DSN=...'.	Improper alter-conn.
007758	List-definition size specifier must be literal.	Improper alter-conn.
007759	Attempt to access a non-existent List- column ('01').	No such list column name.
007760	Must specify one of the keywords, FROM- ROWSETS or FROM_PARAMETER .	Incorrect syntax for LOAD_LOOKUP.
007761	Incorrect syntax to Load-lookup 'PARAMETERS=' keyword. Use: PARAMETERS=(slit nlit \$var #var % var &var, ...) No line wrapping is allowed for this usage.	Incorrect syntax for LOAD_LOOKUP.
007762	Too many parameters (`02) entered to Load- Lookup command. Max parameters is `01.	Incorrect syntax for LOAD_LOOKUP.
007763	Problem executing the cursor for LOAD- LOOKUP table `01'. DETAILS: `02.	The database server returned an error while trying to execute the SQL statement needed to process the LOAD-LOOKUP command. `01 = Load lookup table name
007764	Bad return fetching row from database in LOAD-LOOKUP table `01'. DETAILS: `02	The database server returned an error while fetching the data. `01 = Load lookup table name

Error Number	Error Message	Suggestion/Interpretation
007765	DC, DI sort options not supported with this SQR version. To sort, use SORT=SC or SORT=SI.	Database sort not supported for LOAD_LOOKUP with DDO. <obsolete>
007766	Must specify a query keyword; PROCEDURE=, COMMAND= or GETDATA=.	Incorrect syntax for Load-lookup. Specify a keyword representing the query.
007767	Unknown column variable type.	Unknown data type returned by the server.
007768	The property `01` was not found in the property sheet for the specified datasource (`02). Available property names are: `03. The datasource property sheet does not include the named property.	Verify the metadata and correct the syntax.
007769	The specified CONNECTION (`01') references a datasource whose property sheet does not show support for the Get-Data query method. The datasource property sheet does not show support for Get-Data.	Verify the metadata and property sheet and correct the syntax.
007770	Attempt to create a Selector (or MDSelector) object failed. This event should not occur. Contact your system administrator.	Failed to create the requested object. Contact your system administrator.
009000	Error reading the font information from the [`01] section in sqr.ini. Font name `02 is too long. The maximum length allowed for a font name is `03.	Use a shorter font name in sqr.ini. `01 = Printing device specific font configuration section. `02 = Font name. `03 = Internal limit for font name.
009001	Error reading the font information from the [`01] section in sqr.ini. Font file path `02 is too long. The maximum length allowed for a font file path is `03.	Use a shorter font file path in sqr.ini. Relocate font file if necessary. `01 = Font configuration section name. `02 = Font file path. `03 = Internal limit for file path.

Error Number	Error Message	Suggestion/Interpretation
009002	Font `01 is not valid for `02 output. Please correct the font configuration in the [`02 Fonts] section of the sqr.ini file.	Use a valid font. Refer to the documentation for supported fonts. If you are using disk based fonts like TrueType, also ensure that the font type specific configuration section (like [TrueType Fonts]) is correctly configured. `01 = Font name. `02 = Selected output type.
009003	File name for `01 font `02 is not specified correctly. Please correct the configuration in the [`01 Fonts] section of sqr.ini file.	The font file path is not specified correctly. Correct the sqr.ini file. `01 = Font type. `02 = Font name.
009004	`01 font file `02 cannot be opened. Please check if the file exists, or correct the font configuration in the [`01 Fonts] section of the sqr.ini file.	Either the font file does not exist in the location specified in sqr.ini file, or the path specified in sqr.ini file is not correct. Correct the error and try again. `01 = Font type. `02 = Font file path.
009005	The directory ID specified for `01 in the [TrueType Fonts] section of the sqr.ini file exceeds the actual number of fonts included in the TrueType collection font file. `02 includes `03 fonts. Please specify a valid directory ID.	Correct the configuration in sqr.ini so that the directory ID is set correctly for the font. Note that the directory ID starts from 0. `01 = Font name. `02 = Font file path. `03 = Actual number of fonts included in TrueType collection.
009006	Font `01 is not a supported type of TrueType/OpenType font.	Currently, SQR does not support CFF based OpenType font. Remove the font from the configuration and try again.
009007	Font `01 is not a supported type of TrueType/OpenType font. The `02' table is missing from the font file.	This TrueType/OpenType font does not have the table SQR needs for processing. Remove the font from the configuration and try again. `01 = Font name. `02 = Table name.

Error Number	Error Message	Suggestion/Interpretation
009008	Font `01 is not a supported type of TrueType/OpenType font. This font does not allow embedding in a document.	SQR requires TrueType/OpenType font with embedding feature allowed to include the subset of the font in the print output. This TrueType/OpenType font does not allow embedding. Remove the font from the configuration and try again.
009009	Font `01 is not a supported type of TrueType/OpenType font. This font does not have a supported type of character to glyph mapping (CMAP) table.	SQR requires a TrueType/OpenType font with CMAP table with Platform ID 3 (Microsoft), Encoding ID 0 (Symbol), 4 (UCS-2), or 10 (UCS-4) and table format 4 or 12. Other CMAP table types and encodings are currently not supported. Remove the font from the configuration and try again. `01 = Font name
009010	Error in the [`01:Exclusion Ranges] section in the INI file. Start character code must be greater than the end character code.	In the exclusion range section, the start character code must be smaller than the end character code. Review the section in INI file and correct error. You can specify the range in a decimal or hexadecimal number. If you use a hexadecimal number, you must prefix the number with '0x' or the number is recognized as a decimal.

Appendix B

Using SQR Sample Programs

This appendix discusses SQR samples.

Understanding SQR Samples

SQR Samples is a library of SQR programs that you can use to adapt and experiment with programs. These programs are stored in the <PS_HOME>\bin\sqr\<database_platform>\SAMPLE (SAMPLEW for Microsoft Windows) directory. You can modify these programs to create configured SQRs.

SQR Sample Programs

Each program comprises a report specification and a sample of the output. This table describes all of the sample SQR programs.

<i>Name</i>	<i>Description</i>
_____.DAT	Data files that are used by the loadall.sqr programs.
_____.MEM	SQR startup files to run small, medium, and large SQR programs.
APPEND.SQR	Demonstrates the APPEND and FIXED-NOLF (no line feed) commands.
APTDIARY.SQR	Demonstrates columns and text wrapping.
AREA100.SQR	Demonstrates a 100% area chart.
BAR100.SQR	Demonstrates a 100% bar chart.
BARCODE.SQR	Demonstrates printing a bar code.
CALENDAR.SQR	Demonstrates nondatabase formatting.

<i>Name</i>	<i>Description</i>
COMP_FOR.SQR	Prints a graph of the forecasted and actual sales for a given employee.
COMP_F_G.SQR	Prints a graph of the forecasted and actual sales for a month or quarter.
COMP_PLN.SQR	Prints a graph of the planned and actual sales for a given employee.
COMP_P_G.SQR	Prints a graph of the planned and actual sales for month or quarter.
COVLET02.SQR	Uses SQR to input data from a user, enter the data in the database, and write a form letter by using a DOCUMENT paragraph.
CUST.SQR	Prints a list of all of the customers, bursted by page.
CUSTLBLS.SQR	Demonstrates printing mailing labels within columns.
CUSTOMER.SQR	Demonstrates multiple detail lines and the NEXT-LISTING command.
CUSTOMR2.SQR	Demonstrates the use of the ON-BREAK argument to the PRINT command.
CUSTOMR3.SQR	Demonstrates the use of the INPUT command to change the report output.
CUSTOMR4.SQR	Demonstrates the use of arrays.
CUSTOMR5.SQR	Demonstrates dynamic queries to enable the user to qualify a report as it runs.
CUST_SUM.SQR	Prints a group of information about each customer in the customer table.
CUSTTAPE.SQR	Demonstrates the flat file output for magnetic tape or other postprocessing.
DATAA.DAT	Needed for append.sqr.

<i>Name</i>	<i>Description</i>
DATAB.DAT	Needed for append.sqr.
DROPALL.SQR	Drops all of the SQR sample tables that are created by the LOADALL program.
DROPPROC.SQR	(Sybase) Deletes leftover, temporary stored procedures belonging to the user.
DYNAMCOL.SQR	Demonstrates the use of dynamic columns, dynamic tables, and variables that are passed to the ON-ERROR procedure.
EMP.SQR	Prints a list of all of the employees, bursted by page.
EMP_COMM.SQR	Calculates each employee's commission, based on sales.
EMP_M_Q.SQR	Lists all employee quotas for a given month or quarter.
ENVELOPE.SQR	Demonstrates the use of printing envelope with proper bar codes.
EXPORT.SQR	Creates two SQR reports: one to export a database table and one to import that table. Data from the table is stored in an external operating system file in compressed format, with trailing blanks removed.
FLATFILE.SQR	Creates an SQR report to extract a database table and place it in a flat file.
FLOATBAR.SQR	Demonstrates a floating bar chart.
FOR_CUST.SQR	Creates a sales forecast for a given customer, grouped by month or quarter.
FOR_EMP.SQR	Creates a sales forecast for a given employee, grouped by month or quarter.
FOR_PROD.SQR	Creates a sales forecast for a given product, grouped by month or quarter.
FOR_REG.SQR	Creates a sales forecast for a given region, grouped by month or quarter.

<i>Name</i>	<i>Description</i>
FOR_SUM.SQR	Creates a table of projected product sales with links to more information.
FORMLETR.SQR	Demonstrates form letters by using a document paragraph.
HILO.SQR	Demonstrates a high-low-close chart.
HISTGRAM.SQR	Demonstrates a histogram chart.
INQUIRY.SQR	Creates an SQR program to display rows at your terminal that are selected from a database table that you specify. The resulting SQR program prompts you to qualify the rows to be selected, display those rows, then repeat.
INVOICE.SQR	Demonstrates creating multiple reports, printing invoices, and printing envelopes.
LOADALL.SQR	Creates and loads the sample tables that are used in the preceding SQR programs.
MAKEDATA.SQR	Creates a data file with a fixed length and NOLF attributes.
MAKEREPT.SQR	Helps you create SQR reports more quickly.
MULTIPLE.SQR	Demonstrates creating multiple reports.
NESTREPT.SQR	Demonstrates nesting procedures.
ORDERS.SQR	Lists all of the orders and the order lines that are associated with them.
ORD_MONG.SQR	Lists all orders for a given month and groups them by employee number.
ORD_M_Q.SQR	Lists all orders for a given month or quarter.
ORD_PROD.SQR	Lists all orders for a given product.
ORD_REGG.SQR	Creates a report of all orders from a given region, grouped by month or quarter.

<i>Name</i>	<i>Description</i>
ORD_SUM.SQR	Displays an summary of orders, grouped by month.
ORD_S_Q.SQR	Prints a graph of the percent of orders for each region (in a year) and four graphs of the percent of orders for each region (one for each quarter of that year).
OVERBAR.SQR	Demonstrates an overlapped bar chart.
PHONELST.SQR	Demonstrates printing within columns, page headings, and page footings.
PLN_EMP.SQR	Creates a sales plan for a given employee, grouped by month or quarter.
PLN_GEN.SQR	Creates a sales plan, grouped by month or quarter.
PLN_REG.SQR	Creates a sales plan for given region, grouped by month or quarter.
PRODUCT.SQR	Lists products and their prices and graphs orders of products.
SALELEAD.SQR	Demonstrates DOCUMENT paragraphs.
SALES.SQR	Demonstrates charting from stored data and printing several charts on one page.
SCATTER.SQR	Demonstrates a scatter chart.
SHOWPROC.SQR	(Sybase) Displays leftover, temporary stored procedures belonging to the user.
STCKAREA.SQR	Demonstrates a stacked area chart.
SQR3DBAR.SQR	Demonstrates a three-dimensional bar chart.
SQRLASER.SQR	Demonstrates graphic and file input/output commands.
SQRLINE.SQR	Demonstrates a line chart.

<i>Name</i>	<i>Description</i>
SQRLOGO.SQR	Demonstrates printing images.
SQRPIE.SQR	Demonstrates a pie chart.
TABREP.SQR	Creates a tabular SQR report for a table that you select.
UPDATE.SQR	Generates an SQR program that enables you to query and update database tables. The created program uses the <i>SHOW</i> command to simulate a menu interface.

Index

Numerics/Symbols

_ character, in variable names 18
: character, in variable names 18
' character, in text literals 22

{report}.bqd file 8
{report}.csv file 9
{sqr-database} reserved variable 20
{sqr-dbcs} reserved variable 20
{sqr-encoding} reserved variable 20
{sqr-encoding- console} reserved variable 19
{sqr-encoding- database} reserved variable 19
{sqr-encoding-file-input} reserved variable 19
{sqr-encoding-file-output} reserved variable 19
{sqr-encoding-report-output} reserved variable 19
{sqr-encoding-source} reserved variable 19
{sqr-hostname} reserved variable 20
{sqr-platform} reserved variable 20

@ character 17
@file... command-line argument 6
\$ character 17
\$current-date reserved variable 18
\$sql-error reserved variable 19
\$sqr-database reserved variable 20
\$sqr-dbcs reserved variable 20
\$sqr-encoding-console reserved variable 19
\$sqr-encoding- database reserved variable 19
\$sqr-encoding- file-input reserved variable 19
\$sqr-encoding-file-output reserved variable 19
\$sqr-encoding-report-output reserved variable 19
\$sqr-encoding-source reserved variable 19
\$sqr-locale reserved variable 20
\$sqr-platform reserved variable 20
\$sqr-program reserved variable 20
\$sqr-report reserved variable 21
\$sqr-ver reserved variable 20
\$username reserved variable 20, 67
& character 17
character 17
#current-column reserved variable 18
#current-line reserved variable 18, 185
#DEBUG command 71
 example 72
#DEFINE command 110, 139, 140
 example 111
#ELSE compiler directive 118, 136
#end-file reserved variable 18, 224
#ENDIF compiler directive 120, 136
#END-IF compiler directive 120
 example 120
#IF compiler directive 120, 135, 136

 example 137
#IFDEF compiler directive 120, 136, 139
#IFNDEF compiler directive 120, 136, 140
#INCLUDE command 11, 140
 example 141
#page-count reserved variable 19
#return-status reserved variable 19
#sql-count reserved variable 19
#sql-status reserved variable 19
#sqr-max-columns reserved variable 20
#sqr-max-lines reserved variable 20
#sqr-pid reserved variable 20
% character 17
3D-EFFECTS argument, DECLARE-CHART
 command 76

A

absolute value (abs) function 147
acos function 148
ADD command 24
 example 25

-AF command-line parameter 323
-A flag 6, 269, 278

AFTER argument, BEGIN-EXECUTE command
 43
AFTER-BOLD argument
 DECLARE PRINTER command 291
 DECLARE-PRINTER command 98
 PRINT command 196
AFTER-PAGE argument
 DECLARE PROCEDURE command 292
 DECLARE-PROCEDURE command 103
 DECLARE-TOC command 107
 USE-PROCEDURE command 245
AFTER-REPORT argument
 DECLARE PROCEDURE command 292
 DECLARE-PROCEDURE command 103
 USE-PROCEDURE command 244
AFTER-TOC argument, DECLARE-TOC
 command 106
aggregate functions 17
ALTER-COLOR-MAP command 25
 example 27
ALTER-LOCALE command
 28, 283, 292, 294, 312
 example 32
ALTER-PRINTER command
 35, 93, 101, 283, 293
 example 36

- ALTER-REPORT command 36
 - example 37
- any_col abbreviation 24
- any_lit abbreviation 23
- any_var abbreviation 23

- AP command-line parameter 323

- APPEND argument, SECURITY command 227
- applets, HTML 8
- arccosine (acos) function 148
- arcsine (asin) function 148
- arctangent (atan) function 148
- args... command-line argument 6
- arguments
 - separating from commands 4
 - using with procedures 116
- ARRAY-ADD command 38
 - example 39
- ARRAY-DIVIDE command 38
 - example 39
- array field operands 145
- array function 157
- ARRAY-MULTIPLY command 38
 - example 39
- arrays
 - arithmetic 38
 - ARRAY-ADD command 38
 - ARRAY-DIVIDE command 38
 - ARRAY-MULTIPLY command 38
 - ARRAY-SUBTRACT command 38
 - CLEAR-ARRAY command 63
 - CREATE-ARRAY command 68
 - elements 69
 - maximums 69
 - moving data into 222
 - resetting fields 63
 - retrieving data from 130
 - searching 178
- ARRAY-SUBTRACT command 38
 - example 39
- ascii function 158
- ascii function 157
- asin function 148
- ASK command 6, 54, 110, 139, 140
 - example 40
 - preventing user input 16
 - understanding 40
- atan function 148
- AT-END argument, NEXT-COLUMN command 187

- B**

- bar codes
 - printing 213
 - types 213
 - UPC-A 215
- BATCH-MODE argument, INPUT command 143
- BEEP argument, SHOW command 233
- BEFORE argument, BEGIN-EXECUTE command 43
- BEFORE-BOLD argument
 - DECLARE PRINTER command 291
 - DECLARE-PRINTER command 98
 - PRINT command 196
- BEFORE-PAGE argument
 - DECLARE PROCEDURE command 292
 - DECLARE-PROCEDURE command 103
 - DECLARE-TOC command 107
 - USE-PROCEDURE command 244
- BEFORE-REPORT argument
 - DECLARE PROCEDURE command 292
 - DECLARE-PROCEDURE command 103
 - USE-PROCEDURE command 244
- BEFORE-TOC argument, DECLARE-TOC command 106
- BEGIN-DOCUMENT command 41, 120
 - example 42
- BEGIN-DOCUMENT paragraph
 - marker locations 17
- BEGIN-EXECUTE command 42
 - example 44
- BEGIN-FOOTING command 45, 120
 - example 46, 120
- BEGIN-HEADING command 46, 120
 - example 47
- BEGIN-PROCEDURE command 48, 121
 - example 49
- BEGIN-PROGRAM command 50, 121, 283, 284
 - example 51
- BEGIN-REPORT command 283, 284
- BEGIN-SELECT command 42, 51, 121
 - example 53
- BEGIN-SETUP command 2, 53, 121
 - example 54
- BEGIN-SQL command 54, 55, 121
 - example 57
- BEGIN TRANSACTION command, Sybase and Microsoft SQL Server 226
- bind variables 52
- blanks, trailing 12, 15, 16
- BLINK argument, SHOW command 233

- Bnn argument, BEGIN-SELECT command 52
- Bnn flag 6

- BOLD argument
 - PRINT command 196
 - SHOW command 233
- BORDER argument, DECLARE-CHART command 77
- BOTTOM-MARGIN argument, DECLARE-LAYOUT command 89
- BOX argument
 - GRAPHIC command 134
 - PRINT command 197
- boxes, drawing 134

braces, use of in syntax 3
 branching, EVALUATE command 122
 BREAK argument, EVALUATE command 123
 BREAK command 57
 browser, determining 8
 buffer 6

-BURST:{xx} flag 7, 270, 278

buttons, HTML 315

C

C, writing functions in 172

-CA command-line parameter 322

CALL command 58
 example 60
 CALL SYSTEM command 58
 Cancel dialog box 7

-CAP command-line parameter 322

CAPTION argument, PRINT-BAR-CODE
 command 215
 case/switch 122

-CB flag 7, 270
 -CD command-line parameter 322

ceil function 149
 CENTER argument, PRINT command 197

-C flag 7, 269

CHAR, TEXT data type 142
 character columns 17
 characters
 control 11, 298, 299
 converting single-byte to multibyte 226
 display 118
 multibyte 226
 nondisplay 118, 298, 299
 printer control 12
 single-byte 226
 character sequences, finding 128
 character sets, specifying in Windows 315
 CHARS-INCH argument, DECLARE PRINTER
 command 289
 CHAR-SIZE argument, DECLARE PRINTER
 command 288
 CHART.FILL option 74
 ChartAreaBackground 75
 ChartAreaForeground 75
 ChartBackground 75
 ChartForeground 75
 charts 216
 business 75
 colors 75
 defining attributes 74
 printing 217
 types 217
 CHART-SIZE argument, DECLARE-CHART
 command 75
 CHAR-WIDTH argument
 DECLARE-LAYOUT command 86, 90
 CHECKSUM argument, PRINT-BAR-CODE
 command 215
 chr function 158
 CLEAR-ARRAY command 63, 69
 example 63
 CLEAR-LINE argument, SHOW command 233
 CLEAR-SCREEN argument, SHOW command
 233
 CLOSE command 63, 189
 example 64

-Cnn argument
 BEGIN-SELECT command 52
 BEGIN-SQL command 56
 -CO command-line parameter 322

CODE-PRINTER argument, PRINT command
 197
 COLOR_1 argument, CREATE-COLOR-

- PALETTE command 70
- COLOR_2 argument, CREATE-COLOR-PALETTE command 70
- COLOR_n argument, CREATE-COLOR-PALETTE command 70
- COLOR argument, DECLARE-PRINTER command 95
- COLOR-PALETTE argument, DECLARE-CHART command 76
- color palettes 74
- colors
 - altering defined 25
 - aqua 26
 - black 26
 - blue 26
 - defaults 26, 83, 132, 228, 229
 - defining in SQR reports 82
 - defining new 25
 - fuchsia 26
 - gray 26
 - green 26
 - lime 26
 - maroon 26
 - naming 26, 131
 - navy 26
 - olive 26
 - palettes 70, 74
 - purple 26
 - red 26
 - references 71
 - retrieving 131
 - RGB values 26
 - silver 26
 - teal 26
 - white 26
 - yellow 26
- Colors section, pssqr.ini 320
- columns 17
 - character 17
 - date 17
 - determining value 122
 - displaying 112
 - dynamic 17
 - global 18
 - local 18
 - not supported in DECLARE-CHART or PRINT-CHART command 74
 - number 17
 - reserved variable for current 18
 - reserved variable for maximum 20
 - setting current 242
- COLUMNS argument, POSITION command 192
- COLUMNS command 64, 187, 242
- COMMAND argument, BEGIN-EXECUTE command 43
- command line, SQR 4
- command-line arguments 5
 - @file... 6
 - args... 6
 - connectivity 5
 - flags 5
 - program 5
- command-line flags 6
 - SQR Print 278
- commands
 - abbreviations in 23
 - avoiding older 283
 - breaking 4
 - case insensitivity 4
 - hyphens in 4
 - length of 18
 - listed alphabetically 23
 - overview 23
 - rules for entering 4
 - separating from arguments 4
- comments 4
- COMMIT command
 - example 65
 - suppressing 15
 - understanding 65
- COMMIT TRANSACTION command 66
- comparison operators 137, 145
- compiler directives 118, 120, 139, 140
 - nesting 136
 - table of 136
- CONCAT command
 - edit masks 205
 - example 67
 - understanding 66
- concatenation, with STRING command 236
- cond function 158
- conditional processing 122, 138, 246
- Configuration Manager 321
- CONNECT command 65, 67
 - example 68
- connection_name argument 83
- CONNECTION argument, BEGIN-EXECUTE command 43
- connectivity command-line argument 5
- constants
 - specifying 111
 - understanding 22
- control characters 11, 298, 299
- copyright banner 11
- cos function 149
- cosh function 149
- cosine (cos) function 149
- COWN command-line parameter 322
- CREATE-ARRAY command 38, 54, 68, 222
 - example 69
- CREATE-COLOR-PALETTE command 70
 - example 71
- CS command-line parameter 322
- CT command-line parameter 322
- currency symbols

- redefining 295
- specifying 30, 292
- cursors
 - controlling 232
 - database 67
 - displaying 14

D

- data
 - elements 17
 - input 141
 - retrieving from array 130
 - types supported by INPUT command 141
- DATA-ARRAY argument, PRINT-CHART command 216, 219
- DATA-ARRAY-COLUMN-COUNT argument, PRINT-CHART command 216, 219
- DATA-ARRAY-COLUMN-LABELS argument, PRINT-CHART command 216, 219
- DATA-ARRAY-ROW-COUNT argument, PRINT-CHART command 216, 219
- database
 - commit 65
 - cursors 67
 - joins 175
 - preventing signin 16
 - suppressing commit 15
 - types 20
 - username 20
 - using designated 241
- database functions 17
- data elements
 - columns 17
 - defining 17
 - literals 22
 - variables 17
- DATA-LABELS argument, DECLARE-CHART command 76
- datasource, logon parameters 84
- date
 - columns 17
 - default formats 30, 206
 - displaying 112
 - edit format characters 202
 - edit masks 30
 - reserved variable for current 18
 - variables 17
- date_col abbreviation 23
- date_var abbreviation 23
- dateadd function 159
- DATE argument
 - DECLARE-VARIABLE command 109
 - DISPLAY command 113
 - MOVE command 181
 - PRINT command 198
 - SHOW command 233
- DATE column type 198
- DATE data type 142
- datediff function 159
- DATE-EDIT-MASK keyword, ALTER-LOCALE command 29, 30
- date functions
 - dateadd 159
 - datediff 159
 - datenow 159

- datetostr 160
- strtodate 167
- datenow function 159, 283, 285
- DATE-SEPARATOR keyword, ALTER-LOCALE command 29, 31
- DATETIME column type 198
- DATE-TIME command 283, 285
- datetostr function 160
- DAY-OF-WEEK-CASE keyword, ALTER-LOCALE command 29, 31
- DAY-OF-WEEK-FULL keyword, ALTER-LOCALE command 29, 31
- DAY-OF-WEEK-SHORT keyword, ALTER-LOCALE command 29, 31
- DB2 226
 - COMMIT command 65
 - ROLLBACK command 226
- DBconnectionstring argument, BEGIN-SELECT command 52, 56
- DBdatabase flag 7, 270
- DEBUG flag 7, 71, 136, 139, 140
- DEBUGLP ON command-line parameter 324
- DECIMAL argument, DECLARE-VARIABLE command 109
- DECIMAL-SEPARATOR keyword, ALTER-LOCALE command 29, 31
- decimal variables 18, 25
- DECLARE-CHART command 54, 73, 119, 216
 - example 80
- DECLARE-COLOR-MAP command 82
 - example 83
- DECLARE-CONNECTION command 83
 - example 84
- DECLARE-IMAGE command 54, 85, 119
 - example 85
- DECLARE LAYOUT command 286
- DECLARE-LAYOUT command
 - 54, 86, 119, 185, 283
 - arguments of 88
 - example 90
 - FORMFEED argument 296
 - MAX-COLUMNS argument 296
 - MAX-LINES argument 296
 - PAPER-SIZE argument 297
- DECLARE PRINTER command 283, 286
- DECLARE-PRINTER command
 - 54, 93, 119, 196, 283, 284, 286, 293
 - example 102
 - FILL specification 74
 - INIT-STRING argument 298
 - RESET-STRING argument 298
- DECLARE PROCEDURE command 283, 291
- DECLARE-PROCEDURE command
 - 54, 102, 119, 220, 244, 283, 291
 - example 103
- DECLARE-REPORT command 54, 86, 104, 119
 - example 105
- DECLARE-TOC command 54, 106

- example 107
- DECLARE-VARIABLE command 54, 108, 119
 - example 109
- default colors 228
- DEFAULT layout 86
- DEFAULT-NUMERIC argument, DECLARE-VARIABLE command 109
- default settings section
 - pssqr.ini 302
- DEFAULT value 30
- deg function 150
- degrees 150
- delete function 156
- destination field 180
- dimension hierarchy
 - extending 231
 - specifying 230
- display characters 118
- DISPLAY command
 - edit masks 205
 - example 113
 - understanding 112
- DISTINCT argument, BEGIN-SELECT command 52
- DIVIDE command 115
 - example 116

- DL command-line parameter 323
- Dnn flag 7, 270, 278
- DNT flag 7, 108

- DO argument, EXECUTE command 125
- DO command 49, 50, 116
 - example 117
- document marker 41, 192
- document paragraphs 4, 41, 192, 446
- DOLLAR-SYMBOL command 283, 292, 295
- DOT-LEADER argument, DECLARE-TOC command 106
- double-byte character strings 20
- double-byte strings 180
- double precision floating point numbers 25
- dst_any_var argument, MOVE command 181
- dynamic columns 17
- dynamic query variables 52

E

- E[file] flag 8, 270, 279

- e10 function 150
- EDIT argument
 - PRINT command 198

- SHOW command 233
- edit function 160
- edit masks 67
 - case sensitivity 204
 - changing dynamically 205
 - currency symbols 30, 292
 - date 30
 - date samples 204
 - numeric 30
 - numeric samples 200
 - uses 201
- EDIT-OPTION-AD keyword, ALTER-LOCALE command 29, 31
- EDIT-OPTION-AM keyword, ALTER-LOCALE command 29, 31
- EDIT-OPTION-BC keyword, ALTER-LOCALE command 29, 31
- EDIT-OPTION-NA keyword, ALTER-LOCALE command 29, 31
- EDIT-OPTION-PM keyword, ALTER-LOCALE command 29, 31
- edit types 198

- EH_APPLETS:dir flag 8, 271, 279
- EH_BQD:file flag 8, 271, 279
- EH_BQD flag 8, 271, 279
- EH_BROWSER:xx flag 8, 271, 279
- EH_CSV:file flag 9, 271, 279
- EH_CSV flag 9, 271, 279
- EH_CSVONLY flag 9, 271, 279
- EH_FULLHTML:xx flag 9, 271, 279
- EH_Icons:dir flag 9, 271, 279, 280
- EH_IMAGES:dir flag 9
- EH_KEEP flag 9
- EH_LANGUAGE:xx flag 10, 271
- EH_PDF flag 10, 272, 280
- EH_Scale:{nn} flag 10, 272, 280
- EH_XML:file flag 10
- EH_ZIP[:file] flag 10

- ellipses, use of in syntax 3
- ELSE command 118
- ENCODE command 118, 298, 299
 - example 119
- ENCODING environment variable 20
- END-DECLARE command 119
- END-DOCUMENT command 41, 119
- END-EVALUATE command 119
- END-EXECUTE command 42
- END-FOOTING command 45, 119
 - example 120
- END-HEADING command 46, 119
- END-IF command 121, 138
- ending a query 126
- END-PROCEDURE command 48, 121
- END-PROGRAM command 50, 121, 283
 - example 121
- END-REPORT command 283
- END-SELECT command 51, 121
- END-SETUP command 2, 53, 121

- END-SQL command 55, 121
- END-WHILE command 121, 246
- Enhanced-HTML section, pssqr.ini 319
- Environment sections, pssqr.ini 310
- ERASE-PAGE argument, NEXT-COLUMN command 187
- error messages *See* messages
- errors
 - testing conditions 226
 - trapping 52
- escape sequences 119
- EVALUATE command 120, 122, 247
 - branching 122
 - example 123
 - exiting from with BREAK 58
 - nesting 122
- exclamation points
 - displaying 4
 - for comment lines 4
 - use of in syntax 3, 4
- exclusion ranges 318
- execute
 - begin 42
 - end 42
- EXECUTE command
 - example 125
 - Sybase and Microsoft SQL Server 124
- exists function 156
- EXIT-SELECT command 126
 - example 126
- exp function 150
- exponents 150
- expressions 145
 - operands 145
 - operators 146
- EXTENT argument, LOAD-LOOKUP command 176
- EXTRACT command 127
 - example 128

F

- F[file|directory] flag 11, 272, 280

- FIELD argument, CREATE-ARRAY command 69

- FILE command-line parameter 323

- file number 190
- file-related functions 156
 - delete 156
 - exists 156

- rename 156
- files
 - closing 63
 - include 140
 - including external source 140
 - reading 224
- file types
 - FIXED 190
 - FIXED_NOLF 190
 - VARY 190
- FILL argument
 - DECLARE-CHART command 76
 - PRINT command 207
- FIND command 128
 - example 129
- FIXED_NOLF argument, OPEN command 190
- FIXED_NOLF file types 190
- FIXED argument, OPEN command 190
- FIXED file type 190
- flagging SQR 6
- flags command-line argument 5
- FLOAT argument, DECLARE-VARIABLE command 109
- floating point numbers 25
- float variables 18, 25
- floor function 151
- FONT argument
 - ALTER-PRINTER command 36
 - DECLARE PRINTER command 290
 - DECLARE-PRINTER command 97
- fonts
 - adding entries 314
 - changing 294
 - HP LaserJet 98
 - PostScript 99
 - Windows 101
- Fonts section, pssqr.ini 101, 314
 - specifying character sets 315
- FONT-TYPE argument
 - ALTER-PRINTER command 36
 - DECLARE PRINTER command 289
 - DECLARE-PRINTER command 95
- FooterBackground 75
- FooterForeground 75
- footers 2
- FOOTING argument, ALTER-REPORT command 36, 37
- footings 36
- FOOTING section 2, 45
- FOOTING-SIZE argument, ALTER-REPORT command 37
- FOR-APPEND argument, OPEN command 190
- FOREGROUND/BACKGROUND argument, PRINT command 207
- format_mask argument, MOVE command 181
- FORMAT argument, INPUT command 142, 143
- FORMFEED argument, DECLARE-LAYOUT command 88, 296
- form feeds, trailing 16
- form-letter reports 4
- form letters 4
- FOR-READING argument, OPEN command 190
- FOR-REPORTS argument
 - BEGIN-FOOTING command 45
 - BEGIN-HEADING command 47
 - DECLARE-PRINTER command 94
 - DECLARE-PROCEDURE command 103
 - DECLARE-TOC command 106

- USE-PROCEDURE command 244
- FOR-TOCS argument
 - BEGIN-FOOTING command 45
 - BEGIN-HEADING command 47
- FOR-WRITING argument, OPEN command 190
- frame.html 8
- FROM PARAMETER, BEGIN-EXECUTE command 43
- FROM ROWSET argument, BEGIN-EXECUTE command 43
- functions
 - custom 172
 - file-related 147, 156
 - miscellaneous 147, 156
 - numeric 147, 173
 - random 173
 - string 172
 - transcendental 155
- functions, miscellaneous
 - array 157
 - ascii 157
 - asciic 158
 - chr (character) 158
 - cond 158
 - edit 160
 - getenv (get environment variable) 160
 - isblank 161
 - isnull 162
 - to_char 169
 - to_multi_byte 170
 - to_number 170
 - to_single_byte 170
 - wrapdepth 172

G

- GET-COLOR command 131
 - example 132
- GET command 130
 - example 130
- getenv function 160
- GOTO command 133
 - example 133
- GOTO-TOP argument, NEXT-COLUMN command 187

-GPRINT=YES | NO flag 11, 272

- GRAPHIC BOX command 133
 - example 135
- GRAPHIC command 133
- GRAPHIC FONT command 283, 293
- GRAPHIC HORZ-LINE command 133
 - example 135
- GRAPHIC VERT-LINE command 133
 - example 135
- grid, SQR 86

H

- HeaderBackground 75
- HeaderForeground 75
- headers 2
- HEADING argument, ALTER-REPORT command 36, 37
- headings 36, 46
- HEADING section 2, 46
- HEADING-SIZE argument, ALTER-REPORT command 37
- HEIGHT argument, PRINT-BAR CODE command 214
- HORZ-LINE argument, GRAPHIC command 135
- hostname, reserved variable for current 20
- HPLASERJET 94
- HTML
 - <BODY> tags 254
 - <HEAD> tags 255
 - activating procedures 253
 - adding horizontal divider 252
 - applets 8
 - blinking text 258
 - centering 252
 - citations 258
 - code listings 258
 - column headers 268
 - columns in tables 268
 - definition lists 263
 - definitions in definition lists 262
 - directory lists 262
 - displaying 8
 - ending paragraphs 254
 - enhanced code 9
 - general-purpose procedures 251
 - generating 8
 - generating output 251
 - heading procedures 255
 - highlighting procedures 257
 - hypertext link procedures 261
 - hypertext links 261
 - inserting images 253
 - keyboard input 259
 - line breaks 252
 - list items 264
 - list procedures 262
 - marking text that cannot wrap 253
 - menus 264
 - ordered lists 265
 - preventing wrapping 253
 - publishing 251
 - rows in tables 268
 - sample style text 259
 - starting new paragraph 254
 - strikethrough text 260
 - subscript text 260
 - superscript text 260
 - table captions 266
 - table procedures 266
 - tables 267
 - terms in definition lists 263
 - unordered lists 265
 - html_a_end procedure 262
 - html_a procedure 261
 - html_blink_end procedure 258
 - html_blink procedure 258
 - html_br procedure 252

- html_caption_end procedure 266
- html_caption procedure 266
- html_center_end procedure 252
- html_center procedure 252
- html_cite_end procedure 258
- html_cite procedure 258
- html_code_end procedure 259
- html_code procedure 258
- html_dd_end procedure 262
- html_dd procedure 262
- html_dir_end procedure 263
- html_dir procedure 262
- html_dl_end procedure 263
- html_dl procedure 263
- html_dt_end procedure 263
- html_dt procedure 263
- html_h1_end procedure 255
- html_h1 procedure 255
- html_h2_end procedure 256
- html_h2 procedure 256
- html_h3_end procedure 256
- html_h3 procedure 256
- html_h4_end procedure 257
- html_h4 procedure 256
- html_h5_end procedure 257
- html_h5 procedure 257
- html_h6_end procedure 257
- html_h6 procedure 257
- html_hr procedure 252
- html_img procedure 253
- html_kbd_end procedure 259
- html_kbd procedure 259
- html_li_end procedure 264
- html_li procedure 264
- html_menu_end procedure 264
- html_menu procedure 264
- html_nobr_end procedure 253
- html_nobr procedure 253
- html_ol_end procedure 265
- html_ol procedure 265
- html_on procedure 253
- html_p_end procedure 254
- html_p procedure 254
- html_samp_end procedure 259
- html_samp procedure 259
- html_set_body_attributes procedure 254
- html_set_head_tags procedure 255
- html_strike_end procedure 260
- html_strike procedure 260
- html_sub_end procedure 260
- html_sub procedure 260
- html_sup_end procedure 261
- html_sup procedure 260
- html_table_end procedure 267
- html_table procedure 267
- html_td_end procedure 268
- html_td procedure 268
- html_th_end procedure 268
- html_th procedure 268
- html_tr_end procedure 268
- html_tr procedure 268
- html_ul_end procedure 265
- html_ul procedure 265
- HTML buttons 315
- HTML-Images section, pssqr.ini 315
- hyperbolic cosine (cosh) function 149
- hyperbolic sine 154
- hyphens

- in command-line flags 6
- in command names 4
- in commands 4

I

- I command-line parameter 322
- ID flag 11, 272, 280
- Idir_list flag 11
- IF command 121, 138
 - and WHILE commands 247
 - example 138
 - nested within a WHILE loop 247
 - nesting 138
- images 85
 - declaring 85
 - printing 221
- IMAGE-SIZE argument
 - DECLARE-IMAGE command 85
 - PRINT-IMAGE command 221
- include files
 - nesting 140
 - understanding 140
- INDENTATION argument, DECLARE-TOC
 - command 106
- indenting text 41
- Informix
 - COMMIT command 65
 - ROLLBACK command 226
 - STOP command 236
- initcap function 61
- initialization file 301
- INIT-STRING argument, DECLARE-PRINTER
 - command 94, 298
- INPUT command
 - example 143
 - preventing user input 16
 - prompts 141
 - understanding 141
- INPUT commands 6
- INPUT-DATE-EDIT-MASK keyword, ALTER-
 - LOCALE command 29, 30
- INSERT command 141
- instrb function 161
- instr function 161
- int_lit abbreviation 24
- INTEGER argument, DECLARE-VARIABLE
 - command 109
- INTEGER data type 142
- integer variables 18
- INTO argument, EXECUTE command 125
- isblank function 161
- isnull function 162
- ITEM-COLOR argument, DECLARE-CHART
 - command 75, 76

J

- joins 175

K

-KEEP flag 11, 13, 104, 272

KEY argument, LOAD-LOOKUP command 176

L

labels 133

LAST-PAGE command 144
example 144

layout

DEFAULT 86
reports 86

LAYOUT argument, DECLARE-REPORT
command 105

LEFT-MARGIN argument
DECLARE-LAYOUT command 89
DECLARE PRINTER command 288

LEGEND argument, DECLARE-CHART
command 77

LegendBackground 75

LegendForeground 75

LEGEND-PLACEMENT argument, DECLARE-
CHART command 77

LEGEND-PRESENTATION argument,
DECLARE-CHART command 77

LEGEND-TITLE argument, DECLARE-CHART
command 77

lengthb function 162

length function 162

lengthp function 162

lengtht function 163

LET command 145
datenow function 285
edit masks 205
example 174

LEVEL argument, TOC-ENTRY command 239

-LG command-line parameter 323

LINE-HEIGHT argument, DECLARE-LAYOUT
command 86, 90

LINEPRINTER 94

lines

breaking in HTML 252
drawing 134
horizontal 135
reserved variable for current 18
reserved variable for maximum 20
use of in syntax 4
vertical 135

LINES-INCH argument, DECLARE PRINTER
command 288

LINE-SIZE argument, DECLARE PRINTER
command 288

LINE-WIDTH argument, DECLARE-LAYOUT
command 89

lis files 12

list variables 17, 21

accessing 21
arguments 21
assigning 21
creating 21
defining 21

literals 22

determining value 122
displaying 112, 232
numeric 22
text 22

-LL{s | d}{c | i} flag 12

loading an internal table 175

LOAD-LOOKUP command 12, 54, 174, 178
example 177

LOCAL argument, BEGIN-PROCEDURE
command 49

LOCALE keyword, ALTER-LOCALE command
30

locales

changing parameters 28
defined 28
reserved variable for current 20
selecting 28
SYSTEM 28

Locale section, pssqr.ini 312

local procedures 49

log10 function 151

log base 10 151

log base e 151

log function 151

logical comparison operators 145

logical expressions 138

log messages 12

logon, defining parameters 84

LOOKUP command 175, 178
example 178

lookup tables, searching 178

LOOPS argument, BEGIN-SELECT command 52

lowercase, use of in syntax 3, 4

LOWERCASE command 179
example 179

lower function 163

LOWER value, DAY-OF-WEEK-CASE keyword
31

lpad function 163

-LPFLAGS "<flags>" command-line parameter
324

ltrim function 164

M

marker locations 17
MATCH argument, PRINT command 207
mathematical operators 145
MAX-COLUMNS argument, DECLARE-LAYOUT command 89
MAXLEN argument, INPUT command 141, 143
MAX-LINES argument, DECLARE-LAYOUT command 90
MBTOSBS command 179
members, returning set 231
messages
 database errors 19
 directing to a file 8
 disabling display 16
 numbered 328
 setting path and name of error message file 17
 unnumbered 325

-Mfile flag 12

Microsoft SQL Server
 BEGIN TRANSACTION command 226
 ROLLBACK TRANSACTION command 226
 STOP command 236
miscellaneous functions
 ltrim 164
 range 165
 replace 166
 roman 166
 rpad 166
 rtrim 167
MODE argument, SECURITY command 228
mod function 152
MONEY argument
 DISPLAY command 113
 MOVE command 181
 PRINT command 208
 SHOW command 233
MONEY-EDIT-MASK keyword, ALTER-LOCALE command 29, 30
MONEY-SIGN keyword, ALTER-LOCALE command 29, 30
MONEY-SIGN-LOCATION keyword, ALTER-LOCALE command 29, 30

MONEY-SYMBOL command 283, 293, 294
MONTHS-CASE keyword, ALTER-LOCALE command 30, 31
MONTHS-FULL keyword, ALTER-LOCALE command 30, 31
MONTHS-SHORT keyword, ALTER-LOCALE command 30, 31
MOVE command
 edit masks 205
 example 181
 understanding 180

-MR command-line parameter 323

multibyte characters 226
multiple reports 35, 45
MULTIPLY command 184
 example 184
multirow lists 21

N

NAME argument
 ALTER-COLOR-MAP command 26
 BEGIN-FOOTING command 45
 BEGIN-HEADING command 47
 CLEAR-ARRAY command 63
 CREATE-ARRAY command 69
 CREATE-COLOR-PALETTE command 70
 LOAD-LOOKUP command 176
natural log base e raised to x power 150
navigation bar 16
NEED argument, NEXT-LISTING command 189
nesting
 arguments 147
 IF command 138
NEW-PAGE command 185
 example 185
NEW-REPORT command 186
 example 186
NEXT-COLUMN command 64, 187
 example 187
NEXT-LISTING command 188
 example 189
nn abbreviation 24
NO-ADVANCE argument, NEXT-LISTING command 188
NO-CHANGE value, DAY-OF-WEEK-CASE keyword 31
NO-DUPLICATE argument, DECLARE-CONNECTION command 83
NO-FORMFEED command 283, 296
NOLF argument, PRINT-DIRECT command 220
NOLINE argument
 DISPLAY command 112, 113
 SHOW command 233

-NOLIS flag 12, 104, 272

nondisplay characters 118, 298, 299
 nonrecursive variables 21
 NOP argument, PRINT command 208
 NOPROMPT argument, INPUT command 142, 143
 NORMAL argument, SHOW command 233
 NOWAIT argument, CALL command 60
 num_col abbreviation 23
 num_lit abbreviation 24
 num_var abbreviation 23
 NUMBER argument
 DISPLAY command 113
 MOVE command 181
 PRINT command 208
 SHOW command 233
 number columns 17
 NUMBER data type 142
 NUMBER-EDIT-MASK keyword, ALTER-
 LOCALE command 28, 30
 numerals
 numeric literals 22
 numeric edit format characters 199
 numeric edit masks 30
 numeric functions 147, 173
 absolute value 147
 arccosine (acos) 148
 arcsine (asin) 148
 arctangent (atan) 148
 ceil 149
 cosine (cos) 149
 degrees (deg) 150
 e10 (10 raised to x power) 150
 exp (natural log base e raised to x⇒ power)
 150
 floor (largest integer) 151
 hyperbolic cosine (cosh) 149
 hyperbolic sine (sinh) 154
 hyperbolic tangent (tanh) 155
 log10 (log base 10) 151
 log base e 151
 mod 152
 power 152
 radians (rad) 153
 round 153
 sign 153
 sin 154
 sine 154
 square root (sqrt) 154
 tangent (tan) 155
 truncate (trunc) 155
 numeric literals 22
 numeric operands 145
 numeric variables 17, 18
 nvl function 164

O

ODBC
 COMMIT command 65
 ROLLBACK command 226

-O[file] flag 12, 272, 280

-OF command-line parameter 323
 -olim flag 12

ON-BREAK argument, PRINT command 208
 ON-ERROR argument
 BEGIN-EXECUTE command 43
 BEGIN-SELECT command 52
 BEGIN-SQL command 57
 CONNECT command 68
 DIVIDE command 116
 EXECUTE command 125

-OP command-line parameter 323

OPEN command 189
 example 190
 operands
 converting 146
 understanding 145
 operating system
 issuing commands 58
 opening files 189
 reserved variable for returning 20
 operators, precedence 146
 Oracle 52
 BEGIN-SQL command, -Cnn 56
 COMMIT command 65
 PL/SQL 55
 ROLLBACK command 226
 ORIENTATION argument
 DECLARE-LAYOUT command 87, 89
 DECLARE PRINTER command 287

-OT command-line parameter 322

output

- available types 2
- choosing printer type 13
- HTML 251
- layout of files 86
- overriding default file 11
- reserved variable for returning file name 21
- SQR Print 277, 282
- using the command line 277
- writing directly to files 220

OUTPUT argument, EXECUTE command 125

P

page buffer 220

PAGE-DEPTH argument, DECLARE-LAYOUT command 86, 89

page footers 2

page headers 2

PAGE-NUMBER command 191

- example 191

page numbers

- last 144
- reserved variable for current 19

page overflow 185

pages

- adding numbers 191
- new 185
- setting position 192
- writing 185
- writing data at specified position 193

PAGE-SIZE command 283, 296

palettes 74

PAPER-SIZE argument, DECLARE-LAYOUT command 87, 88, 297

paragraphs

- completing 119, 121
- document 4, 41, 192, 446
- SELECT 51, 126
- SQL 55

PARAMETER_LIST argument, BEGIN-EXECUTE command 43

PARAMETERS argument, DECLARE-CONNECTION command 83

parentheses, use of in syntax 3

PASSWORD argument, DECLARE-CONNECTION command 83

-PB flag 12, 272

PDF Fonts: Exclusion Ranges section, pssqr.ini 318

PDF Fonts section, pssqr.ini 316

-P flag 12, 272, 280

pie chart, printing 219

PIE-SEGMENT-EXPLODE argument, DECLARE-CHART command 78

PIE-SEGMENT-PERCENT-DISPLAY argument, DECLARE-CHART command 77

PIE-SEGMENT-QUANTITY-DISPLAY argument, DECLARE-CHART command 77

PITCH argument

- ALTER-PRINTER command 36
- DECLARE-PRINTER command 96

PL/SQL 55

PlotAreaBackground 75

PlotAreaForeground 75

POINT-MARKERS argument, DECLARE-CHART command 77

POINT-SIZE argument

- ALTER-PRINTER command 35
- DECLARE PRINTER command 289
- DECLARE-PRINTER command 95

position abbreviation 24

position argument, PRINT-IMAGE command 221

POSITION command 192

- example 192

postscri.str file 196

POSTSCRIPT 94

power function 152

precedence, operators 146

precision 145

predefined variables *See* reserved variables

PRINT ... CODE command 284, 297

PRINT ... CODE-PRINTER command 284, 297

PRINT-BAR-CODE command 213

- example 215

PRINT-CHART command 74, 216

- example 219

PRINT command 131, 193

- encode characters 119
- format commands 194

Print dialog box, Windows 282

PRINT-DIRECT command

- encode characters 119
- example 220
- understanding 220

-PRINTER:EH flag 13

-PRINTER:EP flag 13

-PRINTER:HP flag 13

-PRINTER:HT flag 13

-PRINTER:LP flag 13

-PRINTER:PD flag 13

-PRINTER:PS flag 13

-PRINTER:WP flag 13, 101

-PRINTER:xx flag 13, 273, 281

- PRINTER.COLOR option 74
- PRINTER argument, PRINT-DIRECT command 220
- printer control characters 12
- PRINTER-DEINIT command 284, 298
- PRINTER-INIT command 284, 298
- printers
 - altering parameters at runtime 35
 - overriding defaults 93
- printer type
 - choosing 13
 - setting 243
- PRINTER-TYPE argument, DECLARE-REPORT command 105
- PRINT-IMAGE command
 - and DECLARE-IMAGE 58
 - example 221
 - understanding 221
- printing
 - Hebrew 198
 - on Windows 101
- PRINT-TEXT-BACKGROUND argument
 - GET-COLOR command 131
 - SET-COLOR command 229
- PRINT-TEXT-FOREGROUND argument
 - GET-COLOR command 131
 - SET-COLOR command 229
- PROCEDURE argument, BEGIN-EXECUTE command 43
- procedures
 - beginning 48
 - calling recursively 48
 - declaring 103
 - global 48
 - invoking 116
 - local 48
 - passing date arguments to 49
 - passing numeric variables to 49
 - passing string arguments to 49
 - running 2, 116
 - stored 55
 - USE-PROCEDURE command 244
- PROCEDURE section 3
- process file, reserved variable for returning 20
- process ID, reserved variable for returning 20
- Processing-Limits section, pssqr.ini 308
- program.lis file 11
- program.log file 12
- program.spf file 16
- program.sqr file 11
- program.sqt file 13
- program command-line argument 5
- programs, samples 443
- PROGRAM section 2, 50
- program structure 2
- prompts, INPUT command 141
- PROPERTIES argument, BEGIN-EXECUTE command 43
- pssqr.exe, command-line options 321
- pssqr.ini
 - for other platforms 301
 - for Windows 301
- pssqr.ini sections
 - Colors 320
 - default settings 302

- Enhanced-HTML 319
- Environment 310
- Fonts 314
- HTML-Images 315
- Locale 312
- PDF Fonts 316
- PDF Fonts: Exclusion Ranges 318
- Processing-Limits 308
- TrueType font 318
- publishing HTML 251
- PUT command
 - example 223
 - rules 222
 - understanding 222

Q

- queries, ending 126
- QUIET argument
 - LOAD-LOOKUP command 177
 - STOP command 236
- quotes
 - use of in commands 23

R

- radians (rad) function 153
- random function 173
- range function 165

- R command-line parameter 322

- READ command 223
 - example 225
- RECORD argument, OPEN command 190
- records
 - reading 224
 - writing to a file 248
- relinking SQR 172
- REMOVE argument, SECURITY command 227
- rename function 156
- replace function 166
- report footer 2
- report headers 2
- report procedures
 - running 2
- reports
 - BEGIN-HEADING command 46
 - closing 186
 - defining 104
 - designing 2
 - displaying output on screen 7
 - layout 2, 86
 - marking sections for security 227
 - multiple 35, 45, 186

- opening 186
- output types 2
- referencing external source files 140
- reserved variable for returning name of
 - output file 21
- samples 22
- switching between 245
- testing 14
- reserved variables 18
 - \$username 67
 - global versus local 49
- RESET-STRING argument, DECLARE-PRINTER command 95
- RETURN_VALUE argument, LOAD-LOOKUP command 176
- REVERSE argument, SHOW command 233
- rgb argument
 - CREATE-COLOR-PALETTE command 71
 - DECLARE-COLOR-MAP command 83
- RGB values 26
- RIGHT-MARGIN argument, DECLARE-LAYOUT command 89
- ROLLBACK command 226, 236
 - and COMMIT command 65
 - example 226
- ROLLBACK TRANSACTION command 66
- ROLLBACK TRANSACTION command, Sybase and Microsoft SQL Server 226
- roman function 166
- ROUND argument
 - ADD command 25, 116
 - MULTIPLY command 184
 - SUBTRACT command 238
- round function 153
- ROWCOUNT command, Oracle and Sybase 19
- rows
 - affected by SELECT paragraph 19
 - buffering 6
 - eliminating duplicates from query 52
 - reserved variable for counting 19
 - specifying number to retrieve 52
 - specifying number to retrieve simultaneously 52
- ROWS argument, LOAD-LOOKUP command 176
- rpadd function 166
- RP command-line parameter 322
- RS flag 13, 14
- RSV argument, BEGIN-EXECUTE command 43
- rtrim function 167
- runtime file
 - saving 13
 - using 14

S

- sample programs 443
- sample reports 22
- SBTOMBS command 226
- scaling factor 10
- SCHEMA argument, BEGIN-EXECUTE command 43
- screen I/O 232
- searches, using LOAD-LOOKUP 175
- sections, completing 119, 121
- SECURITY command 227
 - example 228
- security IDs 227
- SELECT paragraphs
 - beginning 51
 - exiting 126
- sequential processing, FOR-READING 190
- SET argument, SECURITY command 227
- SET-COLOR command 228
 - example 229
- SET-GENERATIONS command 230
 - example 230
- SET-LEVELS command 231
 - example 231
- SET-MEMBERS command 231
 - example 231
- SETUP section 2, 54
- S flag 14, 273
- SHADE argument, PRINT command 197, 211
- SHOW command
 - edit masks 205
 - example 233
 - understanding 232
 - versus DISPLAY command 113
- sid argument, SECURITY command 227
- sign function 153
- sine function 154
- sin function 154
- single-byte characters 226
- single-byte strings 180
- single quotes, displaying 4
- single quotes, use of in syntax 3
- sinh function 154
- SIZE argument, CREATE-ARRAY command 69
- SKIPLINES argument, NEXT-LISTING command 189
- SORT argument, LOAD-LOOKUP command 176
- SOURCE argument
 - DECLARE-IMAGE command 85
 - PRINT-IMAGE command 221
- source code 2
- source field 180
- source files, including external 140
- spaces, use of in syntax 4
- spf files 12, 13
- SPF Viewer, invoking 16
- SQL, syntax errors 52
- SQL paragraphs 55
- SQR

- avoiding older commands 283
- banner 15
- benefits of 2
- command-line arguments 5
- command-line interface 4
- compiling 136
- data elements 17
- defined 1
- executables 4
- flags 6
- grid 86
- high-level programming capabilities 2
- initialization file 16, 301
- language 1, 2
- language overview 1
- object-oriented program logic 2
- page buffer 220
- preventing database sign-in 16
- procedures 2
- process file 20
- process ID 20
- program structure 1, 2
- relinking 172
- relinking to CALL 59
- reserved variable for returning process file 20
- reserved variable for returning process ID 20
- reserved variables 18
- rules for entering commands 1, 4
- sample programs 443
- stopping 236
- syntax conventions 1, 3
- tools 1
- understanding 1, 2
- writing applications with 2
- SQR_DB_DATE_FORMAT setting 49, 66
- SQR_DB_DATE_ONLY_FORMAT setting 49, 66
- SQR_DB_TIME_ONLY_FORMAT setting 49, 66
- sqr.ini file 16
 - Default-Settings section of 30
- sqr.log file 12
- SQR commands
 - rules for entering 1
- sqr-encoding reserved variable 20
- sqrerr.dat file 17
- SQR executable 4
- SQR Execute 1, 4
 - flags 269
 - running 269
- SQREXT.MAK 59
- sqr-hostname, reserved variable 20
- SQRMKE 59
- SQRP executable 4, 277
- SQR Print 1, 4, 277
 - command-line flags 278
 - generating output 282
 - Windows Print dialog box 282
- SQR reports 2
- SQR Samples 1
- SQRT executable 4, 269
- sqrt function 154
- SQRW executable 4
- SQRWP executable 4, 277
- SQRWT executable 4, 269
- square brackets, use of in syntax 3
- square root 154
- STARTUP-FILE argument
 - DECLARE PRINTER command 289
 - DECLARE-PRINTER command 96
- STATUS argument
 - BEGIN-EXECUTE command 43
 - INPUT command 142, 143
 - INPUT command, values for 142
 - OPEN command 190
 - READ command 224
 - WRITE command 249
- STOP command 236
 - example 236
- stored procedures
 - executing 124
 - preventing creation 16, 52, 56
 - understanding 55
- STRING command 236
 - example 237
- string functions 172
 - extract 127
 - instr 161
 - instrb 161
 - length 162
 - lengthb 162
 - lengthp 162
 - lengtht 163
 - lower 163
 - lowercase 179
 - lpad 163
 - ltrim 164
 - substr 168
 - substrb 168
 - substrp 169
 - substrt 169
 - transform 171
 - translate 171
 - unicode 171
 - upper 171
- string operands 145
- strings
 - converting double-byte to single-byte 180
 - copying 127
 - copying into text variables 239
 - double-byte 180
 - finding character sequences within 128
 - single-byte 180
- string values 156
- string variables, converting to uppercase 241
- strtodate function 167, 204
- Structured Query Report *See* SQR
- structuring a program 2
- subroutines
 - calling 58, 59
 - writing 58
- substitution variables
 - #DEFINE command 110
 - #IF compiler directive 136
 - #IFDEF compiler directive 139
 - #IFNDEF compiler directive 140
 - ASK command 40
- substrb function 168
- substr function 168
- substrings 127, 240
- substrp function 169
- substrt function 169
- SUB-TITLE argument, DECLARE-CHART
 - command 76
- SUBTRACT command 238
 - example 238
- Sybase
 - BEGIN TRANSACTION command 226

- ROLLBACK TRANSACTION command
 - 226
- STOP command 236
- USE command 241
- SYBASE CT-Lib 52
- SYMBOL-SET argument
 - ALTER-PRINTER command 36
 - DECLARE PRINTER command 289
 - DECLARE-PRINTER command 95
- syntax conventions 3
- SYSTEM argument, CALL command 60, 61
- SYSTEM locale 28
- SYSTEM value 30

T

- T{B} flag 15
- T{B|Z|BZ|ZB} flag 274
- T{Z} flag 15
- TABLE argument, LOAD-LOOKUP command 176
- table of contents
 - See Also* DECLARE-TOC command
 - creating entries 239
 - defining 106
 - footing for 45
 - heading for 47
 - sharing 106
 - suppressing 16
- tables
 - loading internal 175
- tabs
 - use of in syntax 4
- tabular reports, NEXT-LISTING command 188
- tan (tangent) function 155
- tanh function 155
- temporary stored procedures 16
- testing
 - error conditions 226
 - STOP command 236
- TEXT argument
 - DECLARE-VARIABLE command 109
 - PRINT-BAR-CODE command 215
 - TOC-ENTRY command 239
- text edit format characters 198
- text literals 22
- text variables 17
 - concatenating 236
 - copying strings into 239
- THOUSAND-SEPARATOR keyword, ALTER-LOCALE command 29, 31
- TIME column type 198
- times
 - displaying 113
 - reserved variable for current 18
- TIME-SEPARATOR keyword, ALTER-LOCALE command 29, 31
- TITLE argument, DECLARE-CHART command 76

- Tnn flag 14, 273

- to_char function 169
- to_multi_byte function 170
- to_number function 170
- to_single_byte function 170
- TOC argument, DECLARE-REPORT command 105
- TOC-ENTRY command 239
 - example 239
- TOP-MARGIN argument
 - DECLARE-LAYOUT command 89
 - DECLARE PRINTER command 288
- trailing blanks
 - preserving 12, 16
 - trimming 15
- trailing form feeds 16
- trailing zeros 15
- transcendental functions 155
- transform function 171
- translate function 171

- TR command-line parameter 323

- TrueType Font section, pssqr.ini 318
- trunc (truncate) function 155
- txt_col abbreviation 23
- txt_lit abbreviation 24
- txt_var abbreviation 23
- TYPE argument
 - DECLARE-CHART command 77
 - DECLARE-IMAGE command 85
 - DECLARE PRINTER command 287
 - DECLARE-PRINTER command 94
 - INPUT command 141, 143
 - PRINT-BAR-CODE command 214
 - PRINT-IMAGE command 221

U

- UCALL.C source file 58
- UCALL function
 - arguments 58
 - avoiding database calls 58
- UFUNC.C 172
- UNDERLINE argument
 - PRINT command 211
 - SHOW command 233
- underscores
 - in procedure names 4

- in variable names 4
- unicode function 171
- UNSTRING command 239
 - example 240
- UPC-A bar code 215
- UPDATE command 141
- uppercase, use of in syntax 3, 4
- UPPERCASE command 241
 - example 241
- upper function 171
- UPPER value, DAY-OF-WEEK-CASE keyword 31
- USE-COLUMN command 64, 242
 - example 243
- USE command 54, 241
 - example 242
- USE-PRINTER-TYPE command 243
 - example 243
- USE-PROCEDURE command 244
 - example 245
- USER argument, DECLARE-CONNECTION
 - command 83
- USE-REPORT command 245
 - example 246
- user input
 - INPUT command 141
 - preventing 16
- username, database 20

V

- VALUE argument, ALTER-COLOR-MAP
 - command 26
- value determination 122
- variables 17
 - assigning 22
 - assigning values of expressions to 145
 - bind 52
 - case sensitivity 18
 - converting to uppercase 241
 - date 17
 - decimal 18, 25
 - declaring 108
 - determining value 122
 - displaying 112, 232
 - dynamic query 52
 - float 18, 25
 - global 18
 - initialization 18
 - in syntax 3
 - integer 18
 - length of 17
 - list 17, 21
 - list variable arguments 21
 - local 18
 - names 17
 - naming 17
 - nonrecursive 21
 - numeric 17, 18
 - reference 40
 - reserved 18
 - rules 17
 - string 127
 - substitution 40, 111, 139
 - text 17
- VARY argument, OPEN command 190
- VARY file type 190

- vertical bars, use of in syntax 3
- VERT-LINE argument, GRAPHIC command 135

- Vserver flag 15, 274

W

- WAIT argument, CALL command 60
- WHEN argument, EVALUATE command 123
- WHEN-OTHER argument, EVALUATE command 123
- WHERE argument, LOAD-LOOKUP command 176
- WHILE ... END-WHILE loop 58
- WHILE command 121
 - example 247
 - exiting from with BREAK 58
 - understanding 246
- WHILE loop 246
- wildcard characters, on UNIX/Linux-based systems 5
- Windows
 - generating output 282
 - selecting a printer 282
- WITH RECOMPILE argument, EXECUTE
 - command 125
- WRAP argument, PRINT command 211
- wrapdepth function 172
- WRITE command 248
 - example 249

X

- X-AXIS-GRID or Y-AXIS-GRID argument,
 - DECLARE-CHART command 80
- X-AXIS-LABEL or Y-AXIS-LABEL argument,
 - DECLARE-CHART command 78
- X-AXIS-MAJOR INCREMENT or Y-AXIS-MAJOR INCREMENT argument, DECLARE-CHART
 - command 79
- X-AXIS-MAJOR-TICK- MARKS argument,
 - DECLARE-CHART command 79
- X-AXIS-MAX-VALUE argument, DECLARE-CHART
 - command 78
- X-AXIS-MINOR INCREMENT or Y-AXIS-MINOR INCREMENT argument, DECLARE-CHART
 - command 79
- X-AXIS-MINOR-TICK-MARKS argument,
 - DECLARE-CHART command 79
- X-AXIS-MIN-VALUE argument, DECLARE-CHART
 - command 78
- X-AXIS-SCALE or Y-AXIS-SCALE argument,
 - DECLARE-CHART command 78
- X-AXIS-TICK-MARK-PLACEMENT or Y-AXIS-TICK-MARK-PLACEMENT argument,
 - DECLARE-CHART command 79

- XB flag 15, 274, 281
- XC argument, EXECUTE command 124
- XCB flag 15, 274
- XC flag 15
- XI flag 16
- XLFF flag 16
- XL flag 16, 275
- XMB flag 16, 275
- XNAV flag 16, 275, 281
- XP argument
 - BEGIN-SELECT command 52
 - BEGIN-SQL command 56
- XP flag 16, 275
- XTB flag 16, 275, 281
- XTOC flag 16, 275, 281

Y

- Y-AXIS-MAJOR-TICK- MARKS argument, DECLARE-CHART command 79
- Y-AXIS-MAX-VALUE argument, DECLARE-CHART command 78
- Y-AXIS-MINOR-TICK-MARKS argument, DECLARE-CHART command 79
- Y-AXIS-MIN-VALUE argument, DECLARE-CHART command 78

Z

- zeros, trailing 15

- ZIF{file} flag 16, 275, 281

- ZIP+4 Postnet 215

- ZIV flag 16, 275
- ZMF{file} flag 17, 276, 282

