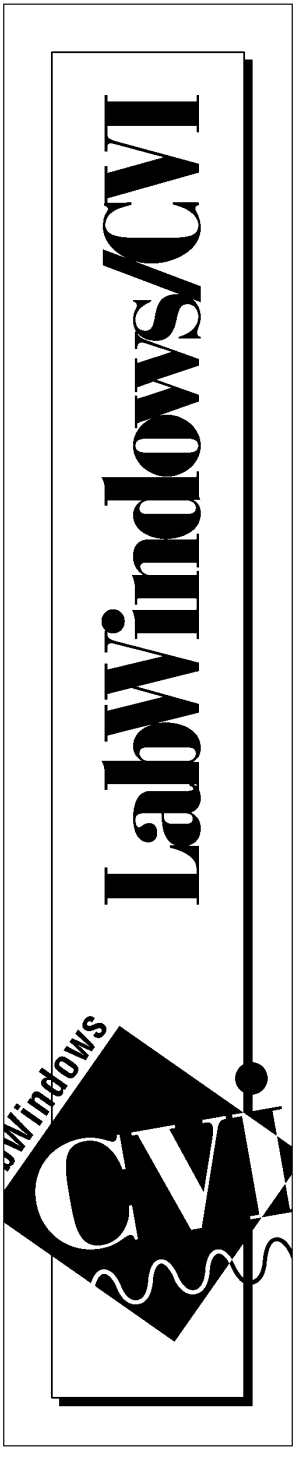


LabWindows®/CVI SPC Toolkit Reference Manual

January 1996 Edition

Part Number 321062A-01

Internet Support

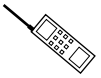
GPIB: gpib.support@natinst.com
DAQ: daq.support@natinst.com
VXI: vxi.support@natinst.com
LabVIEW: lv.support@natinst.com
LabWindows: lw.support@natinst.com
HiQ: hiq.support@natinst.com
VISA: visa.support@natinst.com
FTP Site: ftp.natinst.com
Web Address: www.natinst.com

Bulletin Board Support

BBS United States: (512) 794-5422 or (800) 327-3077
BBS United Kingdom: 01635 551422
BBS France: 1 48 65 15 59

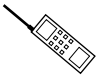
FaxBack Support

(512) 418-1111 or (800) 329-7177



Telephone Support (U.S.)

Tel: (512) 795-8248
Fax: (512) 794-5678 or (800) 328-2203



International Offices

Australia 03 9 879 9422, Austria 0662 45 79 90 0, Belgium 02 757 00 20,
Canada (Ontario) 519 622 9310, Canada (Québec) 514 694 8521, Denmark 45 76 26 00,
Finland 90 527 2321, France 1 48 14 24 24, Germany 089 741 31 30, Hong Kong 2645 3186,
Italy 02 48301892, Japan 03 5472 2970, Korea 02 596 7456, Mexico 5 202 2544,
Netherlands 03480 33466, Norway 32 84 84 00, Singapore 2265886, Spain 91 640 0085,
Sweden 08 730 49 70, Switzerland 056 20 51 51, Taiwan 02 377 1200, U.K. 01635 523545

National Instruments Corporate Headquarters

6504 Bridge Point Parkway Austin, TX 78730-5039 Tel: (512) 794-0100

Important Information

Warranty

The media on which you receive National Instruments software are warranted not to fail to execute programming instructions, due to defects in materials and workmanship, for a period of 90 days from date of shipment, as evidenced by receipts or other documentation. National Instruments will, at its option, repair or replace software media that do not execute programming instructions if National Instruments receives notice of such defects during the warranty period. National Instruments does not warrant that the operation of the software shall be uninterrupted or error free.

A Return Material Authorization (RMA) number must be obtained from the factory and clearly marked on the outside of the package before any equipment will be accepted for warranty work. National Instruments will pay the shipping costs of returning to the owner parts which are covered by warranty.

National Instruments believes that the information in this manual is accurate. The document has been carefully reviewed for technical accuracy. In the event that technical or typographical errors exist, National Instruments reserves the right to make changes to subsequent editions of this document without prior notice to holders of this edition. The reader should consult National Instruments if errors are suspected. In no event shall National Instruments be liable for any damages arising out of or related to this document or the information contained in it.

Except as specified herein, National Instruments makes no warranties, express or implied, and specifically disclaims any warranty of merchantability or fitness for a particular purpose. Customer's right to recover damages caused by fault or negligence on the part of National Instruments shall be limited to the amount theretofore paid by the customer. National Instruments will not be liable for damages resulting from loss of data, profits, use of products, or incidental or consequential damages, even if advised of the possibility thereof. This limitation of the liability of National Instruments will apply regardless of the form of action, whether in contract or tort, including negligence. Any action against National Instruments must be brought within one year after the cause of action accrues. National Instruments shall not be liable for any delay in performance due to causes beyond its reasonable control. The warranty provided herein does not cover damages, defects, malfunctions, or service failures caused by owner's failure to follow the National Instruments installation, operation, or maintenance instructions; owner's modification of the product; owner's abuse, misuse, or negligent acts; and power failure or surges, fire, flood, accident, actions of third parties, or other events outside reasonable control.

Copyright

Trademarks

WARNING REGARDING MEDICAL AND CLINICAL USE OF NATIONAL INSTRUMENTS PRODUCTS

National Instruments products are not designed with components and testing intended to ensure a level of reliability suitable for use in treatment and diagnosis of humans. Applications of National Instruments products involving medical or clinical treatment can create a potential for accidental injury caused by product failure, or by errors on the part of the user or application designer. Any use or application of National Instruments products for or involving medical or clinical treatment must be performed by properly trained and qualified medical personnel, and all traditional medical safeguards, equipment, and procedures that are appropriate in the particular situation to prevent serious injury or death should always continue to be used when National Instruments products are being used. National Instruments products are NOT intended to be a substitute for any form of established process, procedure, or equipment used to monitor or safeguard human health and safety in medical or clinical treatment.

Table of Contents

Contents

About This Manual ix
- Organization of This Manual ix
- Conventions Used in This Manual x
- Related Documentation x
- Customer Communication xi

Chapter 1
Installation 1-1
- Installation 1-1
  - Windows 1-1
  - SPARCstation 1-2
- Requirements for Using the SPC Toolkit 1-3

Chapter 2
SPC Overview 2-1
- What is SPC? 2-1
  - SPC Symbols 2-1
  - Purpose of SPC 2-2
  - The Tools of SPC 2-2
    - Control Charts 2-2
    - Process Capability 2-6
    - Pareto Analysis 2-7

Chapter 3
Control Chart Functions 3-1
- Calculating Control Chart Limits and Points 3-1
  - Variables Chart Functions 3-2
  - Attributes Chart Functions 3-4
  - Functions for Drawing Charts 3-4
    - Functions for Plotting Control Chart Points and Limits 3-5
    - Functions for Creating Graphs of Raw Process Data 3-5
  - Rule Checker Functions 3-6

Chapter 4
Process Statistics Functions 4-1
Process Statistics Calculation Functions 4-1
Process Statistics Drawing Functions 4-2

Chapter 5
Pareto Analysis Functions 5-1
Pareto Counter 5-2
Draw Pareto Chart 5-2

Chapter 6
SPC Function Reference 6-1
SPCCheckControlLimits 6-2
SPCComputeProcessCapability 6-4
SPCDetectProcessShift 6-6
SPCDrawChartWithVarLimits 6-8
SPCDrawChartWithZones 6-10
SPCDrawControlChart 6-12
SPCDrawHistogram 6-15
SPCDrawHistogramWithLimits 6-16
SPCDrawNormalPDFWithLimits 6-18
SPCDrawParetoChart 6-21
SPCDrawRunChart 6-23
SPCDrawSinglePoint 6-26
SPCDrawSinglePointWithZones 6-27
SPCDrawTierChart 6-29
SPCFitNrmlPDFToHistogram 6-32
SPCGeneralHistogram 6-33
SPCGetChartPlotAttributes 6-36
SPCGetErrorText 6-38
SPCParetoCounter 6-38
SPCPlotNormalPDF 6-40
SPCProcessMeanAndSigma 6-42
SPCRuleCheckATTWE 6-44
SPCRuleCheckNelson 6-46
SPCSampleStatistics 6-50
SPCSequenceChecker 6-51
SPCSetChartPlotAttributes 6-53
SPCSinglePtXBarRs 6-55
SPCSinglePtmXBarAndmR 6-56
SPCXAndmR 6-57

SPCXBarAndR 6-61
SPCXBarAnds 6-65
SPCc 6-69
SPCmXBarAndmR 6-71
SPCnp 6-75
SPCp 6-77
SPCu 6-80

Appendix A
Legend Control Function Reference A-1
LGClearLegendCtrl A-2
LGConvertGraphToLegend A-2
LGCreateLegendControl A-4
LGDeleteLegendItem A-5
LGDisplayLegendItems A-6
LGGetErrorText A-7
LGGetLegendCtrlAttribute A-7
LGGetLegendItemAttribute A-9
LGInsertLegendItem A-11
LGInsertLegendItemForPlot A-16
LGNumberOfLegendItems A-17
LGSetLegendCtrlAttribute A-17
LGSetLegendItemAttribute A-19

Appendix B
Customer Communication B-1

Glossary Glossary-1

About This Manual

The *LabWindows/CVI SPC Toolkit Reference Manual* describes the LabWindows/CVI add-on package you can use for implementing statistical process control (SPC) functions.

Organization of This Manual

This manual is organized as follows:

- Chapter 1, *Installation*, contains installation instructions, and discusses the LabWindows/CVI SPC Toolkit functions and examples.
- Chapter 2, *SPC Overview*, gives a short overview of Statistical Process Control (SPC).
- Chapter 3, *Control Chart Functions*, describes the control chart functions, which include the variables charts, attributes charts, chart drawing, and rule checking functions. The control chart functions compute control limits for control charts, create control chart graphs, and apply rules to control chart data that detect out-of-control conditions.
- Chapter 4, *Process Statistics Functions*, describes the process statistics functions, which are useful for process capability analysis and for viewing and measuring process distribution.
- Chapter 5, *Pareto Analysis Functions*, describes the Pareto analysis functions, which include the Pareto Counter function, and the Pareto Chart function.
- Chapter 6, *SPC Function Reference*, describes the toolkit functions in detail.
- Appendix A, *Legend Control Function Reference*, describes the legend control functions used by the SPC toolkit.
- Appendix B, *Customer Communication*, contains forms you can use to request help from National Instruments or to comment on our products and manuals.

- The *Glossary* contains an alphabetical list and description of terms used in this manual, including abbreviations, acronyms, metric prefixes, mnemonics, and symbols.

Conventions Used in This Manual

The following conventions are used in this manual:

bold — Bold text denotes menus, menu items, and VI input and output parameters.

italic — Italic text denotes emphasis, a cross reference, or an introduction to a key concept. Italic text also denotes a variable such as *filename* or *N* when it appears in a text passage.

bold italic — Bold italic text denotes a note, caution, or warning.

`monospace` — Monospace font denotes text or characters that you enter using the keyboard. File names, directory names, drive names, sections of code, programming examples, syntax examples, and messages and responses that the computer automatically prints to the screen also appear in this font.

This icon to the left of bold italicized text denotes a warning, which alerts you to the possibility of damage to you or your equipment.

This icon to the left of bold italicized text denotes a caution, which alerts you to the possibility of data loss or a system crash.

This icon to the left of bold italicized text denotes a note, which alerts you to important information.

Abbreviations, acronyms, metric prefixes, mnemonics, symbols, and terms are listed in the *Glossary*.

Related Documentation

The following documents contain information that you may find helpful as you read this manual:

- Your LabWindows/CVI *Getting Started* manual.
- Your LabWindows/CVI user manual.
- American Society for Quality Control. *American National Standard. Definitions, Symbols, Formulas, and Tables for Control Charts*, 1987. Publication number: ANSI/ASQC A1-1987.

- Breyfogle, Forest W., *Statistical Methods for Testing, Development, and Manufacturing*, John Wiley and Sons, 1992.
- Montgomery, Douglas C., *Introduction to Statistical Quality Control*, John Wiley and Sons, 2nd edition, 1991.
- Wheeler, Donald J. and Chambers, David S., *Understanding Statistical Process Control*, SPC Press, 2nd edition, 1992.

Customer Communication

National Instruments wants to receive your comments on our products and manuals. We are interested in the applications you develop with our products, and we want to help if you have problems with them. To make it easy for you to contact us, this manual contains comment and technical support forms for you to complete. These forms are in the *Customer Communication* appendix at the end of this manual.

Chapter 1
Installation

This chapter contains the installation procedure, gives an overview of Statistical Process Control (SPC), and discusses the LabWindows/CVI SPC Toolkit functions and examples.

Installation

The following sections contain instructions for installing the SPC Toolkit on Windows and Sun SPARCstation. The SPC Toolkit comes in compressed form on floppy disks. Installing the SPC Toolkit requires approximately 4 MB, and LabWindows/CVI must be installed on your computer.

Windows

Installing for use with LabWindows/CVI for Windows 3.1

You can install the SPC Toolkit from the Windows File Manager, or with the **Run...** command from the **File** menu of the Program Manager.

1. Insert the first SPC Toolkit disk into the 3.5-in. disk drive and run the SETUP.EXE program using one of the following methods.
   - From Windows, select **Run...** from the **File** menu of the Program Manager. A dialog box appears. Type X:\SETUP (where X is the proper drive designation). Press <Enter> or select **OK**.
   - From Windows, launch the File Manager. Click on the drive icon that contains the installation disk. Find SETUP.EXE in the list of files on that disk and double-click on it.
2. After you choose an installation option, follow the instructions that appear on the screen. The installer prompts you to name the directory that contains LabWindows/CVI and its associated files.

After you install the LabWindows/CVI SPC Toolkit, your LabWindows/CVI directory contains a new SPC directory.

Installing for use with LabWindows/CVI for Windows 95 or Windows NT

You can install the SPC Toolkit from the Windows Explorer, the Windows File Manager, or with the **Run...** command from the **Start** menu of the Windows 95 Taskbar or the **File** menu of the Windows NT Program Manager.

1. Insert the first SPC Toolkit disk into the 3.5-in. disk drive and run the SETUP32.EXE program using one of the following methods.
   - From Windows, select **Run...** from the **File** menu of the Program Manager. A dialog box appears. Type X:\SETUP32 (where X is the proper drive designation). Press <Enter> or select **OK**.
   - From Windows, launch the Windows Explorer or the File Manager. Click on the drive icon that contains the installation disk. Find SETUP32.EXE in the list of files on that disk and double-click on it.
2. After you choose an installation option, follow the instructions that appear on the screen. The installer prompts you to name the directory that contains LabWindows/CVI and its associated files.

After you install the LabWindows/CVI SPC Toolkit, your LabWindows/CVI directory contains a new SPC directory.

SPARCstation

You can install the SPC Toolkit as described in the following steps. You do not need root privileges to install the SPC Toolkit, but you must be able to write to the LabWindows/CVI directory where the SPC Toolkit will be installed.

If your system runs Solaris 2.2 or later, enter the following command to determine whether your system is running the volume manager:

```
ps -a | fgrep vold
```

The following message usually appears to tell you that the volume manager is running:

```
14818 pts/9 S 0:01 /usr/sbin/vold
```

Installing under Volume Manager

If volume manager is running, install the SPC toolkit as follows:

1. Use the `cd` command to change to a directory where you have write permission, such as `/var/tmp` or your home directory.
2. Insert the first SPC Toolkit disk into the 3.5 in. disk drive.
3. Enter the `volcheck` command.
4. To extract the installation script enter the following command:

```
tar xf /vol/dev/aliases/floppy0 INSTALL
```

2. To run the installation script, enter the command `./INSTALL`. Follow the instructions on the screen. The installer prompts you to name the directory that contains LabWindows/CVI and its associated files.

Installing without Volume Manger running or under Solaris1

If volume manager is not running or if your system runs Solaris 1, install the SPC toolkit as follows:

1. Use the `cd` command to change to a directory where you have write permission, such as `/var/tmp` or your home directory.
2. Insert the first SPC Toolkit disk into the 3.5 in. disk drive.
3. Enter the command `tar xf /dev/rfd0c INSTALL` to extract the installation script.
4. To run the installation script, enter the command `./INSTALL`. Follow the instructions on the screen. The installer prompts you to name the directory that contains LabWindows/CVI and its associated files.

After you install the LabWindows/CVI SPC Toolkit, your LabWindows/CVI directory contains a new SPC directory.

Requirements for Using the SPC Toolkit

To build an SPC application, you use parts of the standard LabWindows/CVI programming libraries. The SPC Toolkit package adds the missing pieces you need to complete your SPC application. The SPC Toolkit consists of a function library that implements key SPC functions such as control charts, process statistics, and Pareto analysis. The SPC Toolkit also contains several functions that generate typical SPC graphs.

To use Statistical Process Control effectively, you must be trained in SPC methods. SPC training gives you essential judgment ability and experience. Using the default settings of pre-existing templates is no substitute for applying training and experience to your project. The SPC Toolkit package gives you a basic framework to use with LabWindows/CVI to create SPC applications. However, this framework can serve you best when you receive training in SPC methods or have access to someone who has SPC expertise.

Two especially good sources of information on statistical process control methods are listed in the *Related Documentation* section of *About This Manual*. One of them, *Understanding Statistical Process Control* by Wheeler and Chambers, can help you understand how to apply SPC methods. The other, *Introduction to Statistical /Quality Control* by Montgomery, explains the theoretical and mathematical basis for SPC.

You must have LabWindows/CVI programming experience to use the SPC toolkit. You can explore the simple examples included in the SPC samples directory after going through Chapter 1 in both the LabWindows/CVI user and tutorial manuals which cover basic LabWindows/CVI principles. To modify the more advanced SPC application examples successfully, however, you must be an advanced LabWindows/CVI user.

In the next chapter, you will take a brief look at the organization of the SPC functions. The following section guides you through some of the LabWindows/CVI programming techniques that you can apply to statistical process control.

Chapter 2
SPC Overview

This chapter provides an overview of basic Statistical Process Control (SPC) concepts.

What is SPC?

SPC is an approach to analysis that helps users make continuous improvement in quality and productivity. SPC consists of an orientation toward quality standards combined with a set of statistical methods which help users achieve those quality standards.

SPC methods have been used in industry since the 1940s to monitor the quality of manufacturing and other processes. SPC users detect and correct process problems early and efficiently, using statistical analysis to extract information from data that otherwise appears random or is easily misinterpreted.

Statistical Process Control techniques can be used wherever a product or service is produced, for example, discrete manufacturing, ATE, continuous production, and tracking paperwork problems.

Although the basic SPC techniques are straightforward, proper application of SPC requires that you be trained in SPC methods. Other experts, such as statisticians, can make crucial contributions to SPC projects. It is easy to misapply SPC. The art of SPC includes deciding what properties of the process to measure, when to take measurements and what conclusions to draw from the analysis. Be sure to apply sufficient expertise to your SPC project.

SPC Symbols

You may see some or all of the following SPC symbols in the SPC sections of this manual:

- X, a single measurement.
- X-bar or $\overline{X}$, the average of a set of data.

- mX-bar or $m\overline{X}$, the moving average, the average of *n* successive individual observations.
- R, the range of a set of data.
- R-bar or $\overline{R}$, the average of a set of ranges.
- mR, moving range, the range of *n* successive individual observations.
- mR-bar or $\overline{mR}$, the average moving range.
- s, the standard deviation for a set of data.

Purpose of SPC

All processes have a natural variability or randomness associated with them. For example, you can produce sheet metal that is within thickness tolerances, but always varies slightly in thickness. A process which exhibits only this natural variation is said to be in statistical control or simply in control. Conversely, a process is out of control if there is more variability in the process than can be attributed to natural randomness. The purpose of SPC is to bring a process into control by identifying what keeps the process out of control. These out-of-control factors are called assignable causes. After the process is in-control, the process quality can be continuously measured and improved.

The Tools of SPC

You can use three basic Statistical Process Control methods:

- Control Charts track variation within the process and detect out-of-control conditions so that you can correct them.
- Process Capability Analysis predicts variation in the process output for an in-control process.
- Pareto Analysis helps prioritize correction of assignable causes.

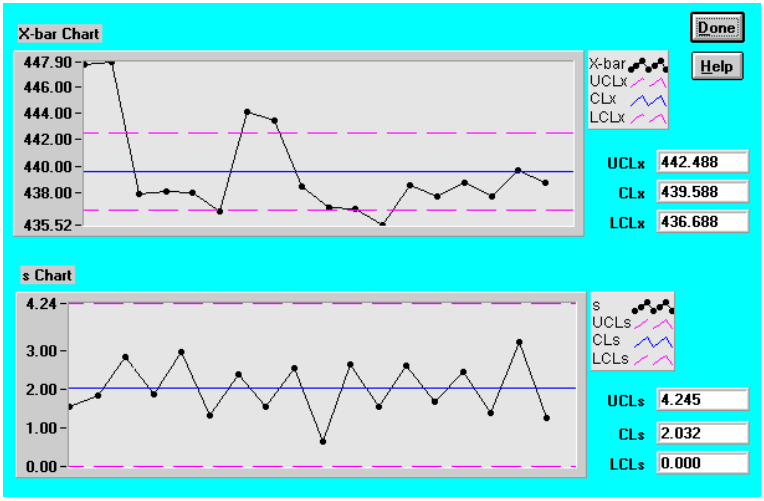
Control Charts

You use control charts to track and characterize process variability and to detect non-random behavior in a process. Control charts plot statistics taken from the process against upper and lower control limits. These upper and lower control limits are also determined from the measured or counted process data. If a plotted point exceeds the three standard-error upper or lower control limits on the chart, there is over a 99% probability that the point is caused by a non-random event.

If non-random behavior exists, then you will be able to find the assignable cause. In other words, something happened to the process that can be identified and corrected because it is not the result of natural variability. Control charts also helps you detect process shifts, such as when the process mean or variation increases or decreases over long periods of time.

There are two categories of control charts: variables charts and attributes charts.

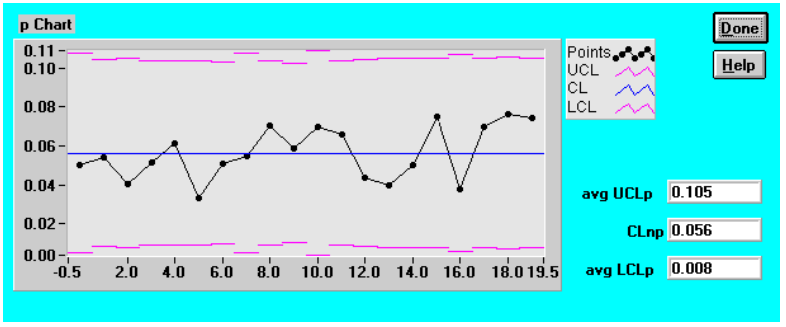
Variables charts track the values associated with specific measurements on samples such as thickness, voltage, or viscosity. Variables charts also help you characterize both continuous and discrete processes. X-bar charts (X-bar is the average of the individual measurements x in a sample) typically display data from discrete manufacturing processes, for example, where a sample consists of measurements from multiple pieces manufactured over a given period of time. The mean and sigma of the sample then reflect the variation in the process over that time period. In practice, an X-bar chart is often paired with an R chart or s chart. These charts track the dispersion (range or standard deviation of the sample) of each sample. The X-bar and s (average of X and standard deviation of X) chart shown below is a variables chart that tracks changes in the mean and the variation in a measured variable from a process.



If your process data can be grouped into rational subgroups (samples) use the X-bar and R or X-bar and s control charts. If only a single value

is available for a given period of time, use an x and mR or mX-bar and mR chart.

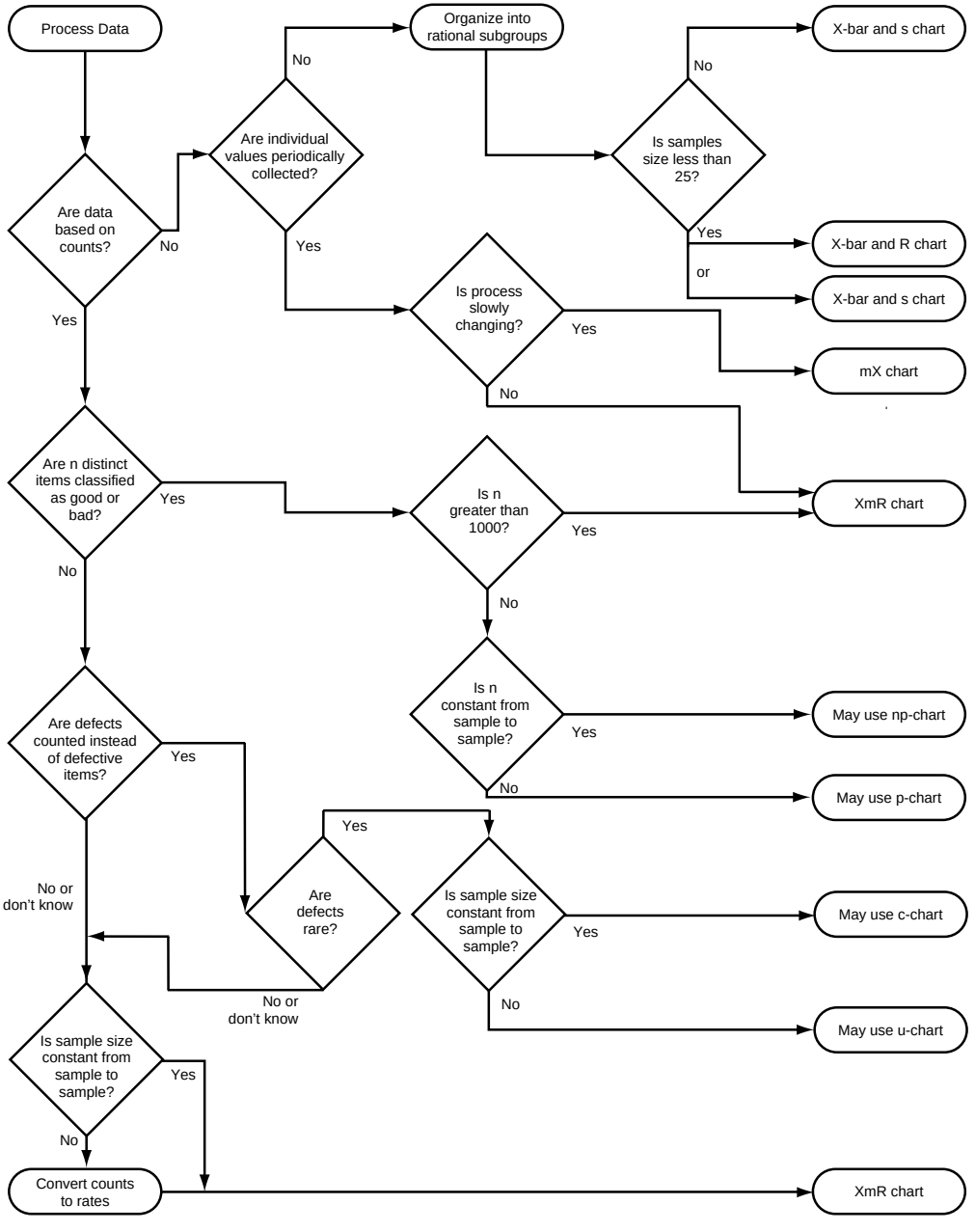
Attributes charts track counted results such as the number of units failed per production run or the number of defects per unit. There are four types of attributes charts, np-chart, p-chart, c-chart and u-chart. If you are counting rejected units use an np-chart or p-chart. If you are counting blemishes or defects per sample instead of defective items, use a c-chart or u-chart.



Whether you use variables or attributes charts, you need to take samples over a period of time that encompasses all natural variations in the process. For example, the temperature in the manufacturing facility may vary predictably over a 24-hour period and show up as an out-of-control variation in a control chart with a smaller sample interval. On the other hand, you can use control charts to detect and then eliminate selected natural variations over time.

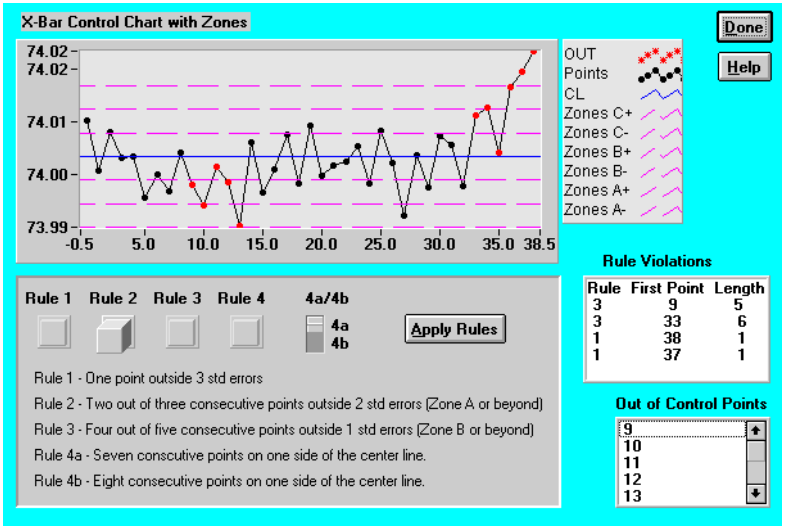
It is vital that your observations of a sample represent the natural variation inherent in the process. Otherwise the results can falsely indicate an out-of-control process.

The following flowchart can help you choose the type of control chart to use.



Flowchart on the Use of Control Charts

Points exceeding the control chart limits give you the first indication that a process is not in-control. Other patterns can show you that the variation in the data is non-random. For example, a Zone chart can help detect significant variation. In the figure below, patterns such as one point outside 3 standard errors, four out of five points in a row outside 1 standard error, and seven consecutive points above or below the center line are detected.

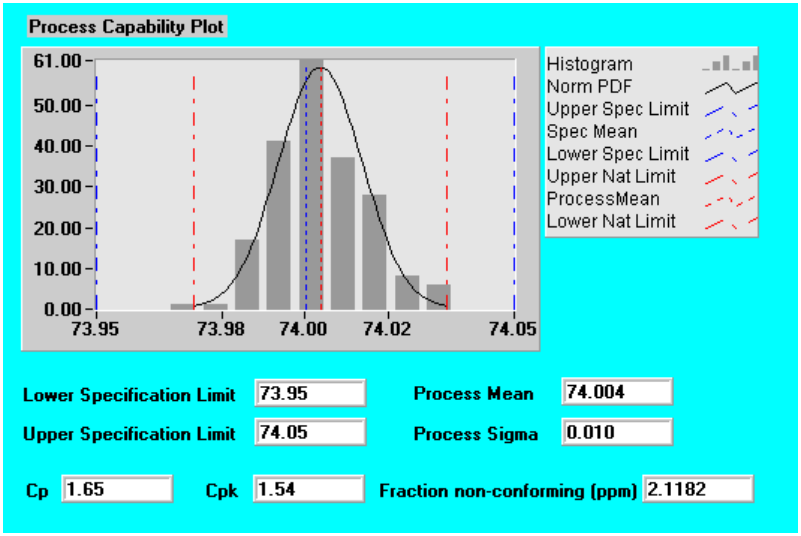


The rules shown above are known as the Western Electric or AT&T rules. Each detected pattern indicates a high probability that the data is not random and the operator or process engineer can investigate and find an assignable cause.

Rules can also help you detect a shift in the process.

Process Capability

If a process is in control, you study process capability to learn what the process is capable of producing when its natural variations are taken into account.

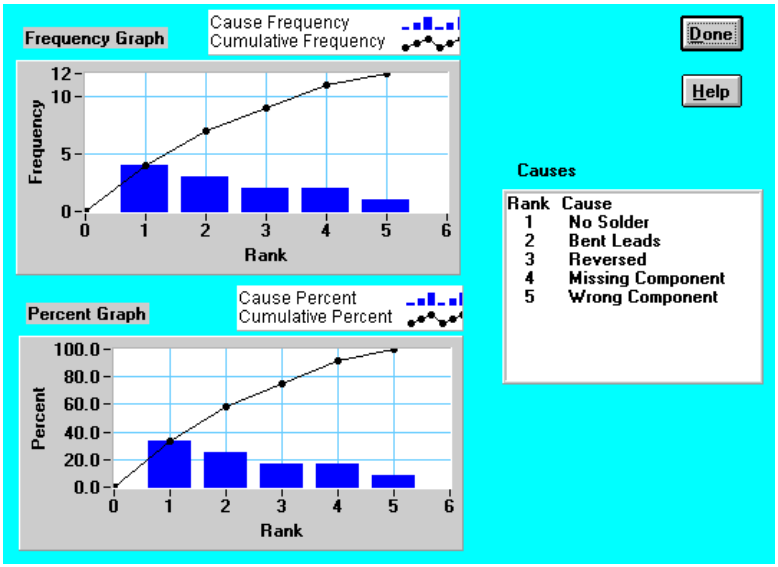


You can plot the distribution of individual observations from your process against the upper and lower specification limits that the manufacturer wishes to meet. Process capability calculations such as Process Capability Index and Cpk indicate how the process is distributed versus the target specification.

If the process data is normally distributed, the fraction non-conforming can also be estimated as part of the process capability measurement. Fraction non-conforming is an estimate of the fraction which is above or below the specification limits in parts per million of the process output. A histogram of individual process measurements overlaid with a bell curve helps you judge whether a given process variable has a normal distribution. It is important to note both the variation in the process (3 * process sigma) and how centered the process is (process mean). Fraction non-conforming will be an invalid estimate unless the variable is truly normally distributed.

Pareto Analysis

Pareto analysis can help you prioritize detection and correction of assignable causes in the process. These assignable causes may display clearly in your control charts or you may have to identify problems through product inspection.



Your Pareto analysis totals the number of times each assignable cause occurred and plots from largest to smallest. The Pareto chart shows which problems to tackle first, and the benefit you can expect from solving each problem.

Chapter 3
Control Chart Functions

This chapter describes the control chart functions. The control chart functions compute control limits for control charts, create control chart graphs, and apply rules to control chart data that detect out-of-control conditions.

Calculating Control Chart Limits and Points

The variables and attributes chart functions compute the points to be plotted on the control charts, as well as the center line and control limits for the control chart. The process data input to the chart functions is a one-dimensional (1D) or two-dimensional (2D) array of samples. The control chart functions return output arrays and chart limits structures which can be passed to one of the chart drawing functions to create the desired control chart graph.

The following list describes other key parameters:

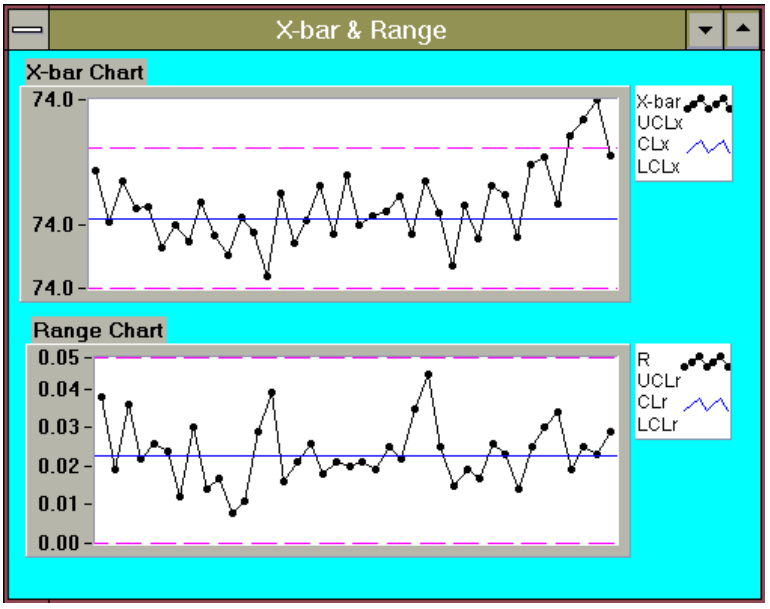
- To use a control chart function to compute the control limits from the input sample data, you select a subset of the input array for the Control Chart with the **Index Spec** parameter. The index specifier parameter designates the start and end index of the samples that the control chart limit calculations use.
- You can also exclude specific samples from the control limit calculation by passing an array of the sample indices as the **Indices to Ignore** parameter of the function. This parameter lets you exclude samples that your rule checking functions have previously shown to be out of control.
- The **Chart Limits** structure contains members for the upper control limit (UCL), center line (CL), lower control limit (LCL), and the standard error that was used to calculate the upper and lower control limits. By default, the limits are equal to center line +/- 3 standard errors.
- The **Samples in Limit Calc** output parameter returns the actual number of samples the function used to calculate the control limits. If both input parameters Limit Source and Index Spec are NULL,

the function calculates the control limits from the entire input array.

- Normally, the control limits are calculated from the input sample data. However, when you supply a Limit Source parameter, the control chart functions will calculate control limits based on the parameter's values. The structures used for the chart limit source are not the same for all the functions.
- The **Standard Error Multiplier** input specifies the multiplier for the function to use when calculating the upper and lower control limits. The default multiplier is 3.0. You do not need to change this input unless you are using upper and lower control limits other than +/- 3.0 standard errors.

Variables Chart Functions

You use variables charts to detect out-of-control conditions in measured process values. The functions for creating variables charts generate output parameters for two control charts—sample mean and variation. The chart for sample mean tracks change in the mean of each sample against control limits. The chart for sample variation tracks the variation in the distribution of each sample against control limits. A typical variables control chart, X-bar and R Chart, is shown in the following illustration.



The variables chart functions named in the following list are described in detail in Chapter 6, *SPC Function Reference*.

- X-bar and s Chart
- X-bar and R Chart
- x and mR Chart
- mX-bar and mR Chart
- Single Point X-bar and R/S
- Single Point x/mX-bar and mR

The X-bar and s Chart and X-bar and R Chart functions take a two-dimensional input array of samples, where each column contains an individual observation within a sample, and each row is a sample. **Sample Count** is the number of rows in the array and **Sample Size** is the number of columns. The X-bar and R Chart function handles sample sizes of 25 or less (25 columns). The X-bar and s Chart function places no limit on the sample size.

The x and mR Chart function and mX-bar and mR Chart function take a one-dimensional input array of individual observations. The functions calculate the moving average range from *n* consecutive observations, where *n* is sample size input. By default, *n* is set to two.

The Single Point X-bar and R/S function calculates points for sample mean and variation control charts, one sample at a time, and uses both the range and sample standard deviation calculations. You can use this function to calculate individual points for a control chart when generating control charts in real time. You also need to use the X-bar and s or X-bar and R function if you want to calculate the control limits.

The Single Point x/mX-bar and mR function calculates the following characteristics for a single measurement (X):

- Individual points
- Moving average
- Moving range

You can also use this function to calculate individual points for a control chart when you want to generate control charts in real time. If you want to calculate the control limits, you need to use the X and mR or mX-bar and mR function.

The variables control chart functions are each described in detail later in Chapter 6, *SPC Function Reference*.

Attributes Chart Functions

You use attributes charts to detect out-of-control conditions on process data that is counted, such as the number of parts defective in a sample of *n* units inspected or number of defects per unit. The SPC Toolkit includes the following attribute charts.

- np chart
- p chart
- c chart
- u chart

The attributes chart functions take one or more one-dimensional arrays as the input data. The p chart and u chart handle either a fixed sample size or variable sample sizes. When the sample sizes are variable, the function calculates the variable control limits.

The attributes chart functions generate outputs for a single control chart. Inputs are one or more 1D arrays that contain values counted from the process. The output includes an array of points to plot on the control chart and the chart limits. In a p or u chart, the sample size inspected may vary for each sample or the sample size may be constant. You can choose one of the following two input parameters: **Sample Size V,** an array input for a variable number inspected, *n*, or **Sample Size C,** a scalar input for a constant number inspected, *n*.

The output arrays **UCL** and **LCL** are the variable control limits (p and u charts only). The **Chart Limits** structure contains the average upper control limit (UCL), center line (CL), average lower control limit (LCL), and the standard error from which the function calculates the upper and lower control limits.

The attributes chart functions are described in detail in Chapter 6, *SPC Function Reference*.

Functions for Drawing Charts

The SPC Toolkit contains several functions that graph the result of calculation by the control chart functions. You can also use the standard LabWindows/CVI graphing functions to present SPC data. The functions in this toolkit use a graph control to draw limits against control chart points, a format that is typical of SPC graph presentations.

Functions for Plotting Control Chart Points and Limits

The control chart functions calculate control chart limits and points. The functions in the following list generate a graph of and the computed points for the control chart, the center line and the upper and lower limit lines (or zone lines for Draw Chart with Zones).

- Draw Control Chart used with constant control limits, is the basic control chart graph.
- Draw Chart with Zones draws zones or warning limits (from constant control limits) and helps you run rules.
- Draw Chart with Var Limits helps you graph variable control limits (p and u charts).

You can also use strip charts to plot your control charts. However, strip charts are more difficult to manage. The strip chart must have the proper number of traces and the calling program must set all trace attributes and the y-axis range. The SPC Toolkit implements strip chart presentation for control charts and run charts only.

These functions are described in detail in Chapter 6, *SPC Function Reference*.

Functions for Creating Graphs of Raw Process Data

The Draw Run Chart and Draw Tier Chart functions create graphs that are independent of the type of control chart you use, and are convenient for displaying the individual observations that make up your samples. This class of graphs can also plot your data against specification limits and/or natural process limits.

Specification limits are user-defined tolerances for the process output. Natural process limits are computed from the samples and represent the process mean and (± 3.0 * process sigma). The natural process limits are a statistic of the variability in your raw data; they are not control limits. The graphing functions for raw process data can be described as follows.

- Draw Run Chart plots a run chart of the individuals within each sample in order of occurrence. This function optionally displays

specification limits and/or natural process limits (process mean and 3 sigma) against the data.

- Draw Tier Chart (variables charts only) plots all observations (individuals) within each sample. This function optionally displays specification limits and/or natural process limits (process mean and ± 3.0 * process sigma) against the data.

These functions have a **Display Mode** specifier that you can use to control whether the specification limits or the natural process limits are drawn. The display mode specifier also designates the sigma multiplier for the function to use for the natural process limits (default 3.0). When you pass NULL for the display mode input parameter, the graphing function uses the defaults. The defaults are not the same for all the functions.

The functions for drawing charts are described in detail in Chapter 6, *SPC Function Reference*.

Rule Checker Functions

The SPC Toolkit contains the following rule checker functions to test whether points exceed the control limits, whether any of the run rules are violated, and to detect process shift.

- Check Control Limits identifies samples that exceed the upper and lower control limits.
- Rule Checker (AT&T/WE) identifies samples that violate one or more of the selected AT&T/Western Electric run rules.
- Rule Checker (Nelson) identifies samples that violate one or more of the selected Nelson run rules.
- Process Shift Detector detects process shift with respect to center line.
- Check Sequence identifies samples violating a generic *n* out of *m* sequence.

When a function identifies out-of-control points, you can use the **Out of Control Points** array from the rule checker function as the **Indices to Ignore** input parameter of the control chart functions and thereby prevent these samples from distorting calculation of the control limit.

The rule checker functions are described in detail in Chapter 6, *SPC Function Reference*.

Chapter 4
Process Statistics Functions

This chapter describes the process statistics functions, which help you analyze process capability and view and measure process distribution. The process statistics functions perform the following operations.

- Compute process mean and sigma
- Compute process capability ratios and reject rates
- Create and graph histograms
- Plot normal probability distribution functions against histograms and process specification limits

Process Statistics Calculation Functions

The SPC Toolkit includes six functions to help you calculate process statistics.

- **Process Mean and Sigma**—Computes **Process Mean**, **Process Sigma** and the natural process limits, **upper NPL** and **lower NPL**, from process samples. You can estimate the process sigma in several ways. If the sample size is greater than one, you can use either sample standard deviation s or range R to estimate the process sigma. You select s or R by the **type** parameter. If the sample size is one, the function automatically uses the moving range mR to estimate the process sigma.
- **Compute Process Capability**—Given the specification limits and the **Process Mean** and **Process Sigma,** computes the process capability ratios **Cp**, **Cpk**, and **Cpkm** as well as **Fraction Non Conf**, the estimated fraction non-conforming in parts per million (ppm). Notice that the fraction non-conforming is valid only if the process is normally distributed. Computes one-sided upper and one-sided lower in addition to two-sided process capability ratios and fraction non-conforming.
- **Sample Statistics Function**—Computes **median**, **mean**, **sample std dev**, **skewness** and **kurtosis** statistics from the input array **sample X**
- **General Histogram**—Finds the discrete histogram of the input sequence **X**. You can allow the function to create the bins automatically by Sturge's Rule or select the **Max** and **Min** values

to include in the histogram, the **Number of Bins** and **Inclusion** (the way bin boundaries are treated). You can also provide **Custom Bins**. The output parameter **Axis** specifies the center values for each bin and **Histogram** specifies the histogram itself.

- **Fit Nrml PDF to Histogram**—Given the bin centers from a histogram (**Axis** values output from the General Histogram function), the estimated **Sigma** of the observations for the histogram, and the total **Number of Observations** in the histogram, calculates the **Normal PDF Height**, which is the height for a normal probability distribution function (PDF) that fits the histogram.
- **Plot Normal PDF** — Given the **Process Mean** and **Process Sigma**, and the **PDF Height** and **PDF Width** returns an arrays of the **X** and **Y** values for the normal distribution. You can graph these values to view the estimated distribution of a sample or group of samples or to plot a normal PDF against a histogram.

Process Statistics Drawing Functions

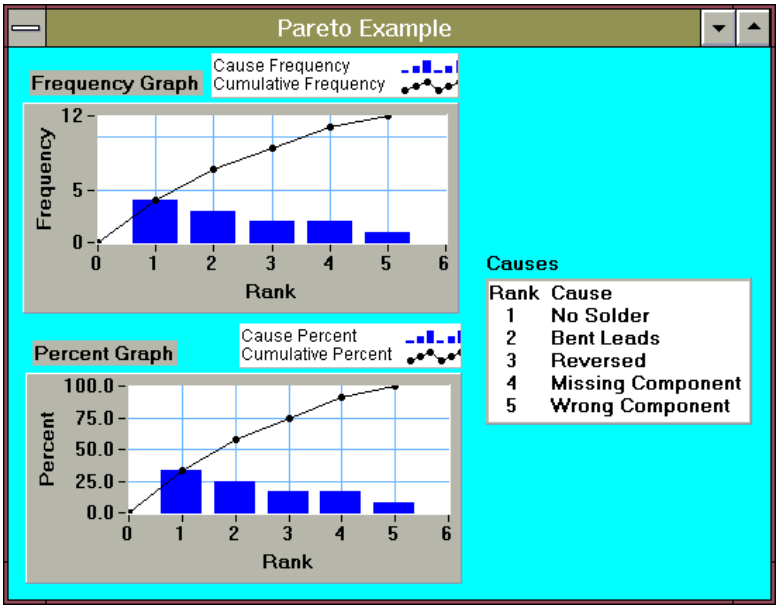
The SPC Toolkit includes three functions to help you plot process statistics.

- **Draw Histogram**—Given the **Bin Centers X** and **Histogram Y** (The **Axis** and **Histogram** outputs of the General Histogram function) draws the graph of the histogram.
- **Draw Histogram with Limits**—Given the **Bin Centers X** and **Histogram Y** (The **Axis** and **Histogram** outputs of the General Histogram function) as well as the **Upper Spec Limit**, **Lower Spec Limit**, **Process Mean**, and **Process Sigma**, draws the graph of the histogram with Specification Limits and Natural Limits.
- **Draw Normal PDF with Limits**—Given the **Upper Spec Limit**, **Lower Spec Limit**, **Process Mean**, **Process Sigma**, **Number of PDF Points**, **PDF Height**, and **PDF Width**, draws the graph of a normal probability distribution function (PDF) of the process against the specification limits and process mean and sigma. With the **Display Mode** parameter, you can control drawing of the specification limits and the natural process limits. This function is often used with Fit Nrml PDF to Histogram and Draw Histogram to overlay the normal PDF on a histogram.

Chapter 5
Pareto Analysis Functions

This chapter describes the two Pareto analysis functions, Pareto Counter and Draw Pareto Chart.

- **Pareto Counter**—given either an unsorted list of causes or a list of causes and corresponding count, produces a set of Pareto totals for each cause.
- **Draw Pareto Chart**—given a set of Pareto values, creates a Pareto chart of the frequency of occurrence of each cause, and a Pareto chart of the percentage contribution of each cause.



The Pareto Counter function accepts two alternative inputs:

- An unsorted list of causes (an array of strings or integers) or
- An array of causes and an array of the corresponding total number of occurrences of each cause.

You can use either integer cause codes or string cause descriptions.

Pareto Counter

Given either an unsorted list of causes or an unsorted list of causes and the number of occurrences for each cause (**Causes Array Integer** or **Causes Array String** and **Cause Counts)**, sorts the list from the cause with the largest number of occurrences to the smallest. Also computes Pareto statistics for each cause, including frequency, cumulative frequency, percentage of total, and cumulative percentage of total for each cause.

Draw Pareto Chart

Given a set of **Pareto Values** (output of the Pareto Counter function), creates two Pareto charts and the associated legend. One is a bar chart of the frequency of occurrence of each cause. The other is a bar chart of the percentage contribution of each cause. The legend is a list of cause codes with their rank displayed in a text box control.

Chapter 6
SPC Function Reference

This chapter is a reference tool for programmers. It describes each function in the Statistical Process Control library. The functions are listed in alphabetical order with a description of the function, C syntax of the function, a description of each parameter, and possible error codes.

The following function tree lists all SPC functions. (Refer to Appendix A, *Legend Control Function Reference*, for descriptions of functions you can use when you want to customize the legends in your SPC graphs and charts.)

Statistical Process Control	**Function Name**
Variables Charts	
X-Bar and s Chart	***SPCXBarAnds***
X-Bar and R Chart	***SPCXBarAndR***
X and mR Chart	***SPCXAndmR***
mX-Bar and mR Chart	***SPCmXBarAndmR***
Single Point X-Bar and R or s	***SPCSinglePtXBarRs***
Single Point x Or mX-Bar and mR	***SPCSinglePtmXBarAndmR***
Attributes Charts	
np Chart	***SPCnp***
p Chart	***SPCp***
c Chart	***SPCc***
u Chart	***SPCu***
Rule Checking	
Check Control Limits	***SPCCheckControlLimits***
Rule Checker - AT&T/WE	***SPCRuleCheckATTWE***
Rule Checker - Nelson	***SPCRuleCheckNelson***
Process Shift Detector	***SPCDetectProcessShift***
Sequence Checker	***SPCSequenceChecker***
Pareto Analysis	
Pareto Counter	***SPCParetoCounter***
Process Statistics	
Process Mean And Sigma	***SPCProcessMeanAndSigma***
Compute Process Capability	***SPCComputeProcessCapability***
Sample Statistics	***SPCSampleStatistics***
General Histogram	***SPCGeneralHistogram***
Fit Normal PDF to Histogram	***SPCFitNrmlPDFToHistogram***
Plot Normal PDF	***SPCPlotNormalPDFChart***

(continues)

Function Tree, Continued

Chart Drawing	
Draw Control Chart	***SPCDrawControlChart***
Draw Single Point	***SPCDrawSinglePoint***
Draw Chart with Zones	***SPCDrawChartWithZones***
Draw Single Point with Zones	***SPCDrawSinglePointWithZones***
Draw Chart with Var Limits	***SPCDrawChartWithVarLimits***
Draw Run Chart	***SPCDrawRunChart***
Draw Tier Chart	***SPCDrawTierChart***
Draw Histogram	***SPCDrawHistogram***
Draw Histogram with Limits	***SPCDrawHistogramWithLimits***
Draw Normal PDF with Limits	***SPCDrawNormalPDFWithLimits***
Draw Pareto Chart	***SPCDrawParetoChart***
Get SPC Plot Attribute	***SPCGetChartPlotAttributes***
Set SPC Plot Attribute	***SPCSetChartPlotAttributes***
Errors	
Get Error Text	***SPCGetErrorText***

The following functions descriptions are in alphabetical order.

SPCCheckControlLimits

`int` **SPCCheckControlLimits** **(**`double` **points**`[]`**,** `int` **numberofPoints,** `double` **UCL,** `double` **LCL,** `double **` **outofControlPoints,** `int *` **outofControlCount);**

Purpose

Given a set of control chart points and the upper and lower control limits from the control chart functions, this function checks for points that exceed the control limits.

Example

```
int *OutOfControlPts;/* for out of control points */
int pointCount;
SPCXAndmR (data, INDIV_COUNT, SAMPLE_SIZE, NULL, NULL, 0, 3.00,
       NULL, &indivInCalc, &xbar, &limitsX, mR, &limitsR,
       &processSigma);
SPCCheckControlLimits(data, INDIV_COUNT, limitsX.UCL,
       limitsX.LCL, &OutOfControlPts, &pointCount);
/* process out of control points */
...
free(OutOfControlPts);
```

Parameters

Input	**points**	double-precision array	The points plotted on the control chart, normally the output of one of the control chart functions such as an X-Bar array.
	numberofPoints	integer	The number of points in the points array.
	UCL	double-precision	The upper control limit line. Normally, this is the UCL element of the limits structure returned by the one of the control chart functions.
	LCL	double-precision	The lower control limit line. Normally, this is the LCL element of the limits structure returned by the one of the control chart functions.
Output	**outofControlPoints**	double-precision * (passed by reference)	An array listing the indices of out of control points found in the points input array. The out of control points are listed in order of how much each point exceeds the control limits. The most extremely out-of-control point is listed first, followed by the next most out-of-control point, and so forth. **Note:** ***The array is allocated by this function. The caller must free the array when it is no longer needed.***
	outofControlCount	integer (passed by reference)	The number of points in the Out of Control Points array.

Return Value

The result code for the function. Zero indicates success, negative values indicate errors.

SPCComputeProcessCapability

```
int SPCComputeProcessCapability (int mode, double upperSpecLimit,
                                 double lowerSpecLimit, double processMean,
                                 double processSigma, double sigmaTolerance,
                                 double *cpU, double *cpL, double *cp,
                                 double *cpk, double *cpkm,
                                 double *fractionNonConf,
                                 double *upperFractionNonConf,
                                 double *lowerFractionNonConf);
```

Purpose

Given the specification limits, the process mean and the process sigma, compute the process capability ratios Cp, Cpk, and Cpkm as well as the estimated process fraction non-conforming in parts per million (ppm).

Note: ***The fraction non-conforming is valid only if the process is normally distributed.***

Parameters

Input	**mode**	integer	Selects whether to calculate two-sided, one-sided upper only or one-sided lower only process capability and fraction non-conforming. The possible values are listed in the *Parameter Discussion*.
	upperSpecLimit	double-precision	The upper specification limit of the process.
	lowerSpecLimit	double-precision	The lower specification limit of the process.
	processMean	double-precision	The estimated process mean. See Also: `SPCProcessMeanAndSigma`
	processSigma	double-precision	The estimated process sigma. See Also: `SPCProcessMeanAndSigma`
	sigmaTolerance	double-precision	The sigma multiplier to use in the process capability calculations. The default value is 6.0, the tolerance spread most commonly used for process capability calculations.

(continues)

Parameters (Continued)

Output	**cpU**	double-precision (passed by reference)	Pointer to the one-sided upper process-capability ratio: $CpU = \frac{\text{upper spec limit - process mean}}{\text{0.5 sigma tolerance} * \text{process sigma}}$
	cpL	double-precision (passed by reference)	Pointer to the one-sided lower process-capability ratio: $CpL = \frac{\text{process mean - lower spec limit}}{\text{0.5 sigma tolerance} * \text{process sigma}}$
	cp	double-precision (passed by reference)	The process capability index (also known as PCR or PCI): $Cp = \frac{\text{upper spec limit - lower spec limit}}{\text{sigma tolerance} * \text{process sigma}}$ Cp is always calculated with respect to both the upper and lower specification limits.
	cpk	double-precision (passed by reference)	The centered capability ratio, also known as the Cpk index. Cpk = MIN(CpU, CpL)
	cpkm	double-precision (passed by reference)	A centered process capability ratio: $Cpkm = \frac{\text{upper spec limit - lower spec limit}}{\text{sigma tolerance * tau}}$ The value *tau* is described in the *Parameter Discussion*.
	fractionNonConf	double-precision (passed by reference)	The estimated fraction non-conforming in parts per million based on a normal process distribution.
	upperFractionNonConf	double-precision (passed by reference)	The estimated fraction non-conforming in parts per million above the upper spec limit based on a normal process distribution.
	lowerFractionNonConf	double-precision (passed by reference)	The estimated fraction non-conforming in parts per million below the lower spec limit based on a normal process distribution.

Return Value

The result code for the function. Zero indicates success, negative values indicate errors.

Parameter Discussion

Constant	Value	Description
SPC_TWO_SIDED	0	Cpk and fraction non-conforming are calculated with respect to both the upper and lower specification limits. This is the default value.
SPC_UPPER_ONLY	1	Cpk and fraction non-conforming are calculated with respect to the upper specification limit only. In this case, Cpk = CpU and fraction non-conforming = fraction > USL.
SPC_LOWER_ONLY	2	Cpk and fraction non-conforming are calculated with respect to the lower specification limit only.In this case Cpk = CpL and fraction non-conforming = fraction < LSL.

Note: ***The values* Cp *and* Cpkm *are always calculated from both the upper and lower specification limits, regardless of the value of* mode.**

For the **cpkm** parameter, *tau* has the following value in a centered process capability ratio:

$$tau = \sqrt{\text{process sigma} + \frac{(\text{process mean} - (\text{up spec} - \text{low spec}))^2}{2}}$$

SPCDetectProcessShift

```
int SPCDetectProcessShift (double points[], int numberofPoints, double CL,
                           int *processShiftDetected, int *firstPointofShift,
                           int *patternDetected);
```

Purpose

Uses run rules to detect a process shift in a control chart with respect to the center line. This function can be used to determine when to recalculate control chart limits.

This function searches for the following patterns that signal a process shift.

Pattern 1: At least 10 out of 11 consecutive data points are on the same side of the center line.

Pattern 2: At least 12 out of 14 consecutive data points are on the same side of the center line.
Pattern 3: At least 14 out of 17 consecutive data points are on the same side of the center line.
Pattern 4: At least 16 out of 20 consecutive data points are on the same side of the center line.

Example

```
SPCDrawControlChart(pan, PANEL_GRAPH1, xLegendCtrl, Xbar,
       NUM_SAMPLES, 0, &limitsX);
SPCDetectProcessShift(Xbar,NUM_SAMPLES, limitsX.CL, &shift,
       &firstPoint, &pattern);
if (shift) printf("Process Shift, pattern %d firstPoint %d\n",
       pattern, firstPoint);
```

Parameters

Input	**points**	double-precision array	The points plotted on the control chart, normally the output of one of the control chart functions such as an X-Bar array.
	numberofPoints	integer	The number of points in the points array.
	CL	double-precision	The center line. Normally, this is the CL element of the limits structure returned by one of the control chart functions.
Output	**processShiftDetected**	integer (passed by reference)	Set to TRUE (1) if the function detected one or more of the four patterns which indicate process shift.
	firstPointofShift	integer (passed by reference)	The index of the first point in the detected process shift or -1 if no process shift was detected. This point can be used as the start index for recalculating control limits.
	patternDetected	integer (passed by reference)	The number of the first pattern detected (1 to 4) or 0 if no pattern was detected.

Return Value

The result code for the function. Zero indicates success, negative values indicate errors.

SPCDrawChartWithVarLimits

int **SPCDrawChartWithVarLimits** (int **panel,** int **graphControl,** int **legendControl,**
double **points**[], int **numberofPoints,**
int **startIndex,** int **Xoffset,**
tSPCChartLimits ***chartLimits,** double **UCL**[],
double **LCL**[]**);**

Purpose

Given an array of control chart points, arrays for the upper and lower control limits and the control chart limits structures as created by one control chart functions, draws a variable limits control chart on the supplied graph control. Use this function with the u chart or p chart if you have a variable number of units inspected per sample.

Example

```
double p[SAMP_COUNT], UCL[SAMP_COUNT], LCL[SAMP_COUNT];

SPCp(rejects, SAMP_COUNT, inspected, 0, NULL, NULL, 0, 3.0, NULL,
       p, &samplesInCalc, &limits, UCL, LCL);

SPCDrawChartWithVarLimits (pan, PANEL_GRAPH, legend, p,
       SAMP_COUNT, 0, 0, &limits, UCL, LCL);
```

Parameters

Input	**panel**	integer	The panel containing the destination graph control.
	graphControl	integer	The control id of the destination graph control.
	legendControl	integer	The control id of the destination graph control for the chart legend. If negative, the legend is not drawn.
	points	double-precision array	Array of the points to be plotted on the chart. Normally the output of the p Chart or c Chart function.
	numberofPoints	integer	The number of points in the Points array.
	startIndex	integer	The index of the first point in the Points array to draw.

(continues)

Parameters (Continued)

	Xoffset	integer	The number of units to offset the plot along the X axis. This parameter is useful when appending new data to an existing control chart. **Note:** ***This parameter has no effect on strip charts***.
	chartLimits	tSPCChartLimits *	Points to a structure containing the limits for the control chart. Normally, the limit output of the p Chart or c Chart function. The elements of the structure are, `double` **UCL**—The average upper control limit for the chart. `double` **CL**—The center line for the chart. `double` **LCL**—The average lower control limit for the chart. `double` **stdErr**—The standard error associated with CL
	UCL	double-precision array	Array containing the variable upper control limit to be drawn on the control chart. The array must contain at least Number of Points elements.
	LCL	double-precision array	Array containing the variable lower control limit to be drawn on the control chart. The array must contain at least Number of Points elements.

Return Value

The result code for the function. Zero indicates success, negative values indicate errors.

Parameter Discussion

Note: ***When using graph controls, it is recommended that you use the default attribute settings,*** `ATTR_DATA_MODE` ***set to*** `VAL_RETAIN` ***and axis scaling set to*** `VAL_AUTOSCALE`***. Otherwise, if a graph discards and auto-scales its data, portions of the SPC drawn data may not appear.***

SPCDrawChartWithZones

int **SPCDrawChartWithZones** (int **panel,** int **graphControl,** int **legendControl,**
double **points**[], int **numberofPoints,** int **startIndex,**
int **Xoffset,** tSPCChartLimits ***chartLimits,**
int **redrawLimits,** int **numberofZones);**

Purpose

Draws a control chart with points plotted against zones. This kind of chart is useful when visually applying run rules to a control chart.

Example

```
SPCXBarAnds (data, SAMP_COUNT, SAMP_SIZE, NULL, NULL, 0, 3.00,
       NULL, &xbarbar, &samplesInCalc, Xbar, &limitsX, s,
       &limitss, &processSigma);

SPCDrawChartWithZones(pan, PANEL_GRAPH, legendCtrlX, Xbar,
       SAMP_COUNT, 0, 0, &limitsX, 1, 3);
```

Parameters

Input	**panel**	integer	The panel containing the destination graph control.
	graphControl	integer	The control id of the destination graph (or strip chart) control. For Details, see *Parameter Discussion*.
	legendControl	integer	The control ID of the destination graph control for the chart legend. If negative, the legend is not drawn.
	points	double-precision array	Array of the points to be plotted on the chart. Normally the output of one of the control chart functions, such as an X-Bar array.
	numberofPoints	integer	The number of points in the Points array.
	startIndex	integer	The index of the first point in the Points array to draw.

(continues)

Parameters (Continued)

	Xoffset	integer	The number of units to offset the plot along the X axis. This parameter is useful when appending new data to an existing control chart. **Note:** ***This parameter has no effect on strip charts***.
	chartLimits	tSPCChartLimits *	`double` **UCL**—The upper control limit for the chart. `double` **CL**—The center line for the chart. `double` **LCL**—The lower control limit for the chart. `double` **stdErr**—The standard error associated with CL.
	redrawLimits	integer	Specifies whether the control limits and center line are redrawn from the start of the graph or appended to the existing limits. Redrawing dashed or dotted limit lines yields a more uniform appearance when plotting a small number of points at a time. Redrawing is also used to fill in the limit lines for points plotted before enough data has been collected to calculate the limit lines. Append is most useful when new limits are calculated. **Note:** ***This parameter has no effect on strip charts***.
	numberofZones	integer	The number of zones to mark on the control chart. The default is three, zones A, B, and C. Zone C is one standard error on either side of the center line, zone B is two standard error and zone A is three standard error.

Return Value

The result code for the function. Zero indicates success, negative values indicate errors.

Parameter Discussion

Note: ***When using graph controls, it is recommended that you use the default attribute settings,*** `ATTR_DATA_MODE` ***set to*** `VAL_RETAIN` ***and axis scaling set to*** `VAL_AUTOSCALE`***. Otherwise, if a graph discards and auto-scales its data, portions of the SPC drawn data may not appear.***

When you use a strip chart, the chart must have one, two, four, six, or eight traces. When drawing only the control chart points, use one trace. Use two traces when drawing the control chart points and the center line. To draw one, two or three zones use four, six or eight traces.

The following table describes the data plotted for each trace.

Constant	Value	Description
SPC_POINTS_TRACE_NUM	1	The control chart points
SPC_CL_TRACE_NUM	2	Control chart center line
SPC_ZONE_C_POS_TRACE_NUM	3	Zone C + (one std error)
SPC_ZONE_C_NEG_TRACE_NUM	4	Zone C -
SPC_ZONE_B_POS_TRACE_NUM	5	Zone B + (two std error)
SPC_ZONE_B_NEG_TRACE_NUM	6	Zone B -
SPC_ZONE_A_POS_TRACE_NUM	7	Zone A + (three std error)
SPC_ZONE_A_NEG_TRACE_NUM	8	Zone A -

SPCDrawControlChart

`int` **SPCDrawControlChart (**`int` **panel,** `int` **graphControl,** `int` **legendControl,**
`double` **points[],** `int` **numberofPoints,** `int` **startIndex,**
`int` **Xoffset,** `tSPCChartLimits *`**chartLimits,**
`int` **redrawLimits);**

Purpose

Draws a control chart with points plotted against upper and lower control limits and center line.

Example

```
SPCXBarAndR (data, SAMP_COUNT, SAMP_SIZE, NULL, NULL, 0, 3.00,
       NULL, &xbarbar, &samplesInCalc, Xbar, &limitsX, R,
```

```
        &limitsR, &processSigma);
SPCDrawControlChart(pan, PANEL_GRAPH, LegendCtrl, Xbar,
        SAMP_COUNT, 0, 0, &limitsX, 1);
```

Parameters

Input	**panel**	integer	The panel containing the destination graph control.
	graphControl	integer	The control id of the destination graph (or strip chart) control.
	legendControl	integer	The control id of the legend control for the chart legend. If negative, the legend is not shown.
	points	double-precision array	Array of the points to be plotted on the chart. Normally the output of one of the control chart functions, such as an X-Bar array.
	numberofPoints	integer	The number of points in the Points array.
	startIndex	integer	The index of the first point in the Points array to draw.
	Xoffset	integer	The number of units to offset the plot along the X axis. This parameter is useful when appending new data to an existing control chart. **Note:** ***This parameter has no effect on strip charts***.
	chartLimits	tSPCChartLimits *	Points to a structure containing the limits for the control chart. Normally the output limits of one of the control chart functions, such as X-Bar limits. If NULL, the chart limits are not drawn. The elements of the structure are, `double` **UCL**—The upper control limit for the chart. `double` **CL**—The center line for the chart. `double` **LCL**—The lower control limit for the chart. `double` **stdErr**—The standard error associated with CL.

(continues)

Parameters (Continued)

	redrawLimits	integer	Specifies whether the control limits and center line are redrawn from the start of the graph or appended to the existing limits. Redrawing dashed or dotted limit lines yields a more uniform appearance when plotting a small number of points at a time. Redrawing is also used to fill in the limit lines for points plotted before enough data has been collected to calculate the limit lines. Append is most useful when new limits are calculated. **Note:** ***This parameter has no effect on strip charts.***

Return Value

The result code for the function. Zero indicates success, a negative value indicate an error.

Parameter Discussion

Note: ***When using graph controls, it is recommended that you use the default attribute settings,*** `ATTR_DATA_MODE` ***set to*** `VAL_RETAIN` ***and axis scaling set to*** `VAL_AUTOSCALE`***. Otherwise, if a graph discards and auto-scales its data, portions of the SPC drawn data may not appear.***

When you use a strip chart, the chart must have either four traces or one trace. Use four traces when drawing the chart limits or one trace when drawing only the control chart points.

The following table describes the data plotted for each trace.

Constant	Value	Description
`SPC_POINTS_TRACE_NUM`	`1`	Control chart points
`SPC_CL_TRACE_NUM`	`2`	Center line
`SPC_UCL_TRACE_NUM`	`3`	Upper control limit
`SPC_LCL_TRACE_NUM`	`4`	Lower control limit

SPCDrawHistogram

`int` **SPCDrawHistogram** **(**`int` **panel,** `int` **graphControl,** `int` **legendControl,** `double` **binCentersX**`[]`**,** `int` **histogramY**`[]`**,** `int` **pointCount);**

Purpose

Draws a vertical bar graph on an existing graph control. The bars are centered at each of the Bin Centers.

Example

```
int *histogram, *axis; /* pointers to arrays */
int binCount;
tSPCNumOutside numOutside;

SPCGeneralHistogram(data, SAMP_COUNT * SAMP_SIZE, NULL, NULL,
       NULL, 0, NULL, 0, &histogram, &axis, &binCount,
       &numOutside);

SPCDrawHistogram(pan, PANEL_GRAPH, axis, histogram, binCount);

free(histogram);
free(axis);
```

Parameters

Input	**panel**	integer	The panel containing the destination graph control.
	graphControl	integer	The control id of the destination graph control.
	legendControl	integer	The control id of the destination graph control for the histogram legend. If negative, the legend is not drawn.
	binCentersX	double-precision array	The x-axis values for the center of each histogram bin. Normally, this is the Axis output array from the General Histogram function.
	histogramY	integer array	The bin counts for each bin in the histogram ordered from left to right. Normally, this is the Histogram output array from the General Histogram function.
	pointCount	integer	The number of points represented by Histogram Y and Bin Centers X.

Return Value

The result code for the function. Zero indicates success, negative values indicate errors.

Parameter Discussion

Note: ***When using graph controls, it is recommended that you use the default attribute settings,*** ATTR_DATA_MODE ***set to*** VAL_RETAIN ***and axis scaling set to*** VAL_AUTOSCALE***. Otherwise, if a graph discards and auto-scales its data, portions of the SPC drawn data may not appear.***

SPCDrawHistogramWithLimits

```
int SPCDrawHistogramWithLimits (int panel, int graphControl, int legendControl,
                                double binCentersX[], int histogramY[],
                                int numberofPoints, double upperSpecLimit,
                                double lowerSpecLimit, double processMean,
                                double processSigma,
                                tSPCDisplayMode *displayMode);
```

Purpose

Draws a vertical bar graph on an existing graph control. The bar graph is centered at each of the Bin Centers. The specification limits and natural process limits are also drawn as vertical lines against the bar graph.

Example

```
int *histogram, *axis; /* pointers to arrays */
int binCount;
tSPCNumOutside numOutside;

SPCProcessMeanAndSigma(data, SAMP_COUNT, SAMP_SIZE, sigmaFromS,
       3.0, 0, &procMean, &procSigma, &uNPL, &lNPL);

SPCGeneralHistogram(data, SAMP_COUNT * SAMP_SIZE, NULL,
       NULL,  NULL, 0, NULL, 0, &histogram, &axis, &binCount,
       &numOutside);

SPCDrawHistogramWithLimits(pan, PANEL_GRAPH, legendCtrl,
       axis,  histogram, binCount, 74.05, 73.95, procMean,
       procSigma, NULL);

free(histogram);
free(axis);
```

Parameters

Input	**panel**	integer	The panel containing the destination graph control.
	graphControl	integer	The control id of the destination graph control.
	legendControl	integer	The control id of the legend control for the chart legend. If negative, the legend is not drawn.
	binCentersX	double-precision array	The x-axis values for the center of each histogram bin. Normally, this is the Axis output array from the General Histogram function.
	histogramY	integer array	The bin counts for each bin in the histogram ordered from left to right. Normally, this is the Histogram output array from the General Histogram function.
	numberofPoints	integer	The number of points in the Histogram Y and Bin Centers X arrays.
	upperSpecLimit	double-precision	The upper specification limit of the process.
	lowerSpecLimit	double-precision	The lower specification limit of the process.
	processMean	double-precision	The estimated process mean. See Also: `SPCProcessMeanAndSigma`
	processSigma	double-precision	The estimated process sigma. See Also: `SPCProcessMeanAndSigma`
	displayMode	tSPCDisplayMode *	Points to a structure for specifying display mode information. The elements of the structure are, `int` **showSpecLimits**—if non-zero, the specification limits will be drawn. `int` **showNPLimits**—if non-zero, the natural process limits will be drawn. `double` **NPLimitSigma**—the sigma multiplier for the natural process limits, typically 3.0 is used. If NULL, the specification limits and natural process limits will be drawn and 3 is used as the sigma multiplier for the natural process limits. The default value is NULL.

Return Value

The result code for the function. Zero indicates success, negative values indicate errors.

Parameter Discussion

Note: ***When using graph controls, it is recommended that you use the default attribute settings,*** ATTR_DATA_MODE ***set to*** VAL_RETAIN ***and axis scaling set to*** VAL_AUTOSCALE***. Otherwise, if a graph discards and auto-scales its data, portions of the SPC drawn data may not appear.***

SPCDrawNormalPDFWithLimits

```
int SPCDrawNormalPDFWithLimits (int panel, int graphControl, int legendControl,
                                double upperSpecLimit, double lowerSpecLimit,
                                double processMean, double processSigma,
                                int numberPDFpoints, double PDFHeight,
                                double PDFWidth,
                                tSPCDisplayMode *displayMode);
```

Purpose

Given the specification limits and the process mean and process sigma, creates a graph of a normal probability distribution function (PDF) of the process against the specification limits and process mean and sigma.

Example

```
SPCProcessMeanAndSigma(data, SAMP_COUNT, SAMP_SIZE, sigmaFromR,
       3.0, 2, &procMean, &procSigma, &uNPL, &lNPL);
SPCGeneralHistogram(data, SAMP_COUNT * SAMP_SIZE, NULL,
       NULL, NULL, 0, NULL, 0, &histogram, &axis, &binCount,
       &numOutside);
SPCFitNrmlPDFToHistogram (axis, binCount, procSigma,
       SAMP_COUNT * SAMP_SIZE, &normPDFHeight);
SPCDrawNormalPDFWithLimits(pan, PANEL_GRAPH, legendCtrl,
       uSpecLim, lSpecLim, procMean, procSigma, 50,
       normPDFHeight, 3.0, NULL);
```

Parameters

Input	**panel**	integer	The panel containing the destination graph control.
	graphControl	integer	The control id of the destination graph control.
	legendControl	integer	The control id of the legend control for the chart legend. If negative, the graph is not drawn.
	upperSpecLimit	double-precision	The upper specification limit of the process.
	lowerSpecLimit	double-precision	The lower specification limit of the process.
	processMean	double-precision	The estimated process mean. See Also: `SPCProcessMeanAndSigma`
	processSigma	double-precision	The estimated process sigma. See Also: `SPCProcessMeanAndSigma`
	numberPDFpoints	integer	The number of points to create for the normal PDF plot. The default value is 50.
	PDFHeight	double-precision	The height to draw the PDF. If zero, the function will draw the PDF with a height proportional to 1/(process sigma), corresponding to an area of one under the PDF curve. When fitting a normal PDF to a histogram, use the PDF Height output from the Fit Normal PDF to Histogram function.
	PDFWidth	double-precision	The width to draw the PDF in terms of +/- sigma. The default value is 3.0.

(continues)

Parameters (Continued)

	displayMode	tSPCDisplayMode *	Points to a structure for specifying display mode information. The elements of the structure are, `int` **showSpecLimits**—if non-zero, the specification limits will be drawn. `int` **showNPLimits**—if non-zero, the natural process limits will be drawn. `double` **NPLimitSigma**—the sigma multiplier for the natural process limits, typically 3.0 is used. If NULL, the specification limits and natural process limits will be drawn and 3 is used as the sigma multiplier for the natural process limits. The default value is NULL.

Return Value

The result code for the function. Zero indicates success, negative values indicate errors.

Parameter Discussion

Note: ***When using graph controls, it is recommended that you use the default attribute settings,*** `ATTR_DATA_MODE` ***set to*** `VAL_RETAIN` ***and axis scaling set to*** `VAL_AUTOSCALE`***. Otherwise, if a graph discards and auto-scales its data, portions of the SPC drawn data may not appear.***

SPCDrawParetoChart

```
int SPCDrawParetoChart (int panel, int frequencyGraphControl,
                        int frequencyLegendGraphControl,
                        int percentGraphControl, int percentLegendGraphControl,
                        int causeListTextBoxControl,
                        tSPCParetoValuePtr *paretoValues,
                        int paretoValuesCount);
```

Purpose

Given a set of Pareto values (output of the Pareto Counter function), draws two Pareto charts. One is a bar chart of the frequency of occurrence of each cause. The other is a bar chart of the percentage contribution of each cause.

Example

```
SPCParetoCounter(NULL, causes, NULL, CAUSE_COUNT, &paretoValues,
       &paretoValCount, &totalCount);

SPCDrawParetoChart(pan, PANEL_GRAPH1, freqLegend, PANEL_GRAPH2,
       percentLegend, PANEL_CAUSELIST, paretoValues,
       paretoValCount);

/* must free memory allocated by SPCParetoCounter */
for (i = 0; i < paretoValCount; i++) {
       free(paretoValues[i]->causeStr;
       free(paretoValues[i]);
}
free(paretoValues);
```

Parameters

Input	**panel**	integer	The panel containing the destination graph control(s).
	frequencyGraphControl	integer	The control id of the destination graph control for bar graph of the frequency of each cause. If negative, the graph is not drawn.

(continues)

Parameters (Continued)

	frequencyLegendGraphControl	integer	The control id of the destination graph control for the frequency Pareto chart legend. If negative, the legend is not drawn.
	percentGraphControl	integer	The control id of the destination graph control for the bar graph of percentage contribution of occurrence of each cause. If negative, the graph is not drawn.
	percentLegendGraphControl	integer	The control id of the destination graph control for the percent Pareto chart legend. If negative, the legend is not drawn.
	causeListTextBoxControl	integer	The control id of a text box for displaying the cause list. If negative, the cause list is not shown.
	paretoValues	tSPCParetoValuePtr *	An array of pointers to structures containing the Pareto statistics for each cause. Normally, this is the output of the Pareto Counter function. See *Parameter Discussion*.
	paretoValuesCount	integer	The number of elements in the Pareto Values array.

Return Value

The result code for the function. Zero indicates success, negative values indicate errors.

Parameter Discussion

paretoValues—The elements of the structure are,

- `char *`**causeStr**—The cause string. NULL if using integer cause codes.
- `int` **causeInt**—The cause code. NULL if using cause strings.

- `int` **frequency**—The number of occurrences of the cause.
- `int` **cumFrequency**—The total of occurrences of this cause and all previous causes.
- `int` **percentOfTotal**—The percent contribution of cause.
- `int` **cumPercent**—The total of percentages of this cause all previous causes.

Note: ***When using graph controls, it is recommended that you use the default attribute settings,*** `ATTR_DATA_MODE` ***set to*** `VAL_RETAIN` ***and axis scaling set to*** `VAL_AUTOSCALE`***. Otherwise, if a graph discards and auto-scales its data, portions of the SPC drawn data may not appear.***

SPCDrawRunChart

`int` **SPCDrawRunChart(**`int` **panel,** `int` **graphControl,** `int` **legendControl,**
`void *` **samplesX,** `int` **sampleCount,** `int` **sampleSize,**
`int` **startSample,** `int` **Xoffset,** `double` **upperSpecLimit,**
`double` **lowerSpecLimit,** `double` **processMean,**
`double` **processSigma,** `tSPCDisplayMode *` **displayMode,**
`int` **redrawLimits);**

Purpose

Draws a run chart of the individuals making up the samples X array in time order. The function interprets the individual observations in the samples array to be in time order by row, and the individuals within each row by column.

Example

```
tSPCDisplayMode displayMode;

SPCProcessMeanAndSigma(data, SAMP_COUNT, SAMP_SIZE, sigmaFromR,
       3.0, 2, &procMean, &procSigma, &uNPL, &lNPL);

displayMode.showSpecLimits = 1;
displayMode.showNPLimits   = 0;   /* don't show nat limits */
displayMode.NPLimitSigma   = 3.0;

SPCDrawRunChart(pan, PANEL_GRAPH1, legendCtrl, data, SAMP_COUNT,
       SAMPLE_SIZE, 0, 0, usl, lsl, procMean, procSigma,
       &displayMode, 1);
```

Parameters

Input	**panel**	integer	The panel containing the destination graph control.
	graphControl	integer	The control id of the destination graph control.
	legendControl	integer	The control id of the destination graph control for the chart legend. If negative, the legend is not drawn.
	samplesX	void *	The 2D array of points to be plotted on the run chart. Each row is a sample with n observations where n is the number of columns in the 2D array. n is also known as the sample or subgroup size. A 1D array can also be used if Sample Size is set to one.
	sampleCount	integer	The number of samples in the Samples X array.
	sampleSize	integer	The number of individual observations in each sample in the Samples X array.
	startSample	integer	The index of the first sample in the samples X array to draw.
	Xoffset	integer	The number of units to offset the plot along the X axis. This parameter is useful when appending new data to an existing control chart. **Note:** ***This parameter has no effect on strip charts***.
	upperSpecLimit	double-precision	The upper specification limit of the process.
	lowerSpecLimit	double-precision	The lower specification limit of the process.
	processMean	double-precision	The estimated process mean. See also: `SPCProcessMeanAndSigma`

(continues)

Parameters (Continued)

	processSigma	double-precision	The estimated process sigma. See also: `SPCProcessMeanAndSigma`
	displayMode	tSPCDisplayMode *	Points to a structure for specifying display mode information. The elements of the structure are, `int` **showSpecLimits**—if non-zero, the specification limits will be drawn. `int` **showNPLimits**—if non-zero, the natural process limits will be drawn. `double` **NPLimitSigma**—the sigma multiplier for the natural process limits, typically 3.0 is used. If NULL, the specification limits and natural process limits will not be drawn and 3 is used as the sigma multiplier for the natural process limits. The default value is NULL.
	redrawLimits	integer	Specifies whether the specification and natural limits are redrawn from the start of the graph or appended to the existing limits. Redrawing dashed or dotted limit lines yields a more uniform appearance when plotting a small number of points at a time. Redrawing is also used to fill in the limit lines for points plotted before enough data has been collected to calculate the limit lines. Append is most useful when new limits are calculated.

Return Value

The result code for the function. Zero indicates success, negative values indicate errors.

Parameter Discussion

Note: ***When using graph controls, it is recommended that you use the default attribute settings,*** `ATTR_DATA_MODE` ***set to*** `VAL_RETAIN` ***and axis scaling set to*** `VAL_AUTOSCALE`***. Otherwise, if a graph discards and auto-scales its data, portions of the SPC drawn data may not appear.***

SPCDrawSinglePoint

`int` **SPCDrawSinglePoint** **(**`int` **panel,** `int` **graphControl,** `double` **point,**
`tSPCChartLimits *`**chartLimits,** `int` **redrawLimits);**

Purpose

Draws a single point on a control chart with points plotted against upper and lower control limits and center line. This function is useful for updating control charts as new samples are collected.

Example

```
double history[SAMP_SIZE];
int historyCount = 0;
SPCSinglePtmXBarAndmR(newPoint, SAMP_SIZE, &mXbar, &mR,
       history, &historyCount);
SPCDrawSinglePoint(panel, PANEL_GRAPH2, mR, &sLimitsR, 1);
```

Parameters

Input	**panel**	integer	The panel containing the destination graph control.
	graphControl	integer	The control id of the destination graph (or strip chart) control. For details, see *Parameter Discussion*.
	point	double-precision	The new point to plot on the control chart.
	chartLimits	tSPCChartLimits *	Points to a structure containing the limits for the control chart. Normally the output limits of one of the control chart functions, such as X-Bar limits. If NULL, the chart limits are not drawn. The elements of the structure are, `double` **UCL**—The upper control limit for the chart. `double` **CL**—The center line for the chart. `double` **LCL**—The lower control limit for the chart. `double` **stdErr**—The standard error associated with CL.
	redrawLimits	integer	Specifies whether the control limits and center line are redrawn or appended. When dashed lines are extended one point at a time, the dashes begin to merge together creating the appearance of a solid line. Redrawing the lines eliminates this effect. **Note:** ***This parameter has no effect on strip chart controls***.

Return Value

The result code for the function. Zero indicates success, negative values indicate errors.

Parameter Discussion

Note: ***When using graph controls, it is recommended that you use the default attribute settings,*** `ATTR_DATA_MODE` ***set to*** `VAL_RETAIN` ***and axis scaling set to*** `VAL_AUTOSCALE`***. Otherwise, if a graph discards and auto-scales its data, portions of the SPC drawn data may not appear.***

When you use a strip chart, the chart must have either four traces or one trace. Use four traces when drawing the chart limits or one trace when drawing only the control chart points.

The following table describes the data plotted for each trace.

Constant	Value	Description
SPC_POINTS_TRACE_NUM	1	Control chart points
SPC_CL_TRACE_NUM	2	Center line
SPC_UCL_TRACE_NUM	3	Upper control limit
SPC_LCL_TRACE_NUM	4	Lower control limit

SPCDrawSinglePointWithZones

```
int SPCDrawSinglePointWithZones (int panel, int graphControl, double point,
                                 tSPCChartLimits *chartLimits,
                                 int redrawLimits, int numberofZones);
```

Purpose

Draws a single point on a control chart with points plotted against zones. This function is useful for updating control charts as new samples are collected.

Example

```
/* assumes limitsX calculated by SPCmXBarAndmR */
SPCSinglePtmXBarAndmR(moreData, SAMP_SIZE, 0, &mXbar, &mr,
       history, &historyCount);
SPCDrawSinglePointWithZones(panel, PANEL_GRAPH1, mXbar,
       limitsX, 1, 3);
```

Parameters

Input	**panel**	integer	The panel containing the destination graph control.
	graphControl	integer	The control id of the destination graph (or strip chart) control. For details, see *Parameter Discussion*.
	point	double-precision	The new point to plot on the control chart.
	chartLimits	tSPCChartLimits *	Points to a structure containing the limits for the control chart. Normally, this is the limit output of one of the control chart functions. The elements of the structure are, `double` **UCL**—The upper control limit for the chart. `double` **CL**—The center line for the chart. `double` **LCL**—The lower control limit for the chart. `double` **stdErr**—The standard error associated with CL.
	redrawLimits	integer	Specifies whether the zone lines and center line are redrawn or appended. When dashed lines are extended one point at a time, the dashes begin to merge together creating the appearance of a solid line. Redrawing the lines eliminates this effect. **Note:** ***This parameter has no effect on strip chart controls.***
	numberofZones	integer	The number of zones to mark on the control chart. The default is three, zones A, B, and C. Zone C is one standard error on either side of the center line, zone B is two standard error and zone A is three standard error.

Return Value

The result code for the function. Zero indicates success, negative values indicate errors.

Parameter Discussion

Note: ***When using graph controls, it is recommended that you use the default attribute settings,*** `ATTR_DATA_MODE` ***set to*** `VAL_RETAIN` ***and axis scaling***

set to VAL_AUTOSCALE***. Otherwise, if a graph discards and auto-scales its data, portions of the SPC drawn data may not appear.***

When you use a strip chart, the chart must have one, two, four, six or eight traces. When drawing only the control chart points, use one trace. Use two traces when drawing the control chart points and the center line. To draw one, two or three zones use four, six or eight traces.

The following table describes the data plotted for each trace.

Constant	Value	Description
SPC_POINTS_TRACE_NUM	1	The control chart points
SPC_CL_TRACE_NUM	2	Control chart center line
SPC_ZONE_C_POS_TRACE_NUM	3	Zone C + (one std error)
SPC_ZONE_C_NEG_TRACE_NUM	4	Zone C -
SPC_ZONE_B_POS_TRACE_NUM	5	Zone B + (two std error)
SPC_ZONE_B_NEG_TRACE_NUM	6	Zone B -
SPC_ZONE_A_POS_TRACE_NUM	7	Zone A + (three std error)
SPC_ZONE_A_NEG_TRACE_NUM	8	Zone A -

SPCDrawTierChart

```
int SPCDrawTierChart(int panel, int graphControl, int legendControl,
                     void *samplesX, int sampleCount, int sampleSize,
                     int startSample, int Xoffset, double upperSpecLimit,
                     double lowerSpecLimit, double processMean,
                     double processSigma, tSPCDisplayMode *displayMode,
                     int redrawLimits);
```

Purpose

Draws a tier chart (also known as a tolerance diagram). Plots the spread of the observations in each sample.

Example

```
SPCProcessMeanAndSigma(data, SAMP_COUNT, SAMP_SIZE,
       sigmafromR, 3.0, 2, &procMean, &procSigma, &uNPL, &lNPL);
SPCDrawTierChart(pan, PANEL_GRAPH, legendCtrl, data, SAMP_COUNT,
       SAMP_SIZE, 0, 0, usl, lsl, procMean, procSigma, NULL, 1);
```

Parameters

Input	**panel**	integer	The panel containing the destination graph control.
	graphControl	integer	The control id of the destination graph control.
	legendControl	integer	The control id of the destination graph control for the chart legend. If negative, the legend is not drawn.
	samplesX	void *	The 2D array of points to be plotted on the tolerance chart. Each row is a sample with n observations where n is the number of columns in the 2D array. n is also known as the sample or subgroup size.
	sampleCount	integer	The number of samples in the samples X array.
	sampleSize	integer	The number of individual observations in each sample in the Samples X array.
	startSample	integer	The index of the first sample in the samples X array to draw.
	Xoffset	integer	The number of units to offset the plot along the X axis. This parameter is useful when appending new data to an existing control chart. **Note:** ***This parameter has no effect on strip charts.***
	upperSpecLimit	double-precision	The upper specification limit of the process.
	lowerSpecLimit	double-precision	The lower specification limit of the process.
	processMean	double-precision	The estimated process mean. **Note:** ***The process mean is needed only when drawing the natural limits is enabled.*** ***See also*:** `SPCProcessMeanAndSigma`
	processSigma	double-precision	The estimated process sigma. **Note:** ***The process sigma is needed only when drawing the natural limits is enabled.*** ***See also*:** `SPCProcessMeanAndSigma`

(continues)

Parameters (Continued)

	displayMode	tSPCDisplayMode *	Points to a structure for specifying display mode information. See *Parameter Discussion.*
	redrawLimits	integer	Specifies whether the specification and natural limits are redrawn from the start of the graph or appended to the existing limits. Redrawing dashed or dotted limit lines yields a more uniform appearance when plotting a small number of points at a time. Redrawing is also used to fill in the limit lines for points plotted before enough data has been collected to calculate the limit lines. Append is most useful when new limits are calculated.

Return Value

The result code for the function. Zero indicates success, negative values indicate errors.

Parameter Discussion

displayMode—The elements of the structure are,

- `int` **showSpecLimits**—if non-zero, the specification limits will be drawn.
- `int` **showNPLimits**—if non-zero, the natural process limits will be drawn.
- `double` **NPLimitSigma**—the sigma multiplier for the natural process limits, typically 3.0 is used.
- If NULL, the specification limits and natural process limits will be drawn and 3 is used as the sigma multiplier for the natural process limits. The default value is NULL.

Note: ***When using graph controls, it is recommended that you use the default attribute settings,*** `ATTR_DATA_MODE` ***set to*** `VAL_RETAIN` ***and axis scaling set to*** `VAL_AUTOSCALE`***. Otherwise, if a graph discards and auto-scales its data, portions of the SPC drawn data may not appear.***

SPCFitNrmlPDFToHistogram

`int` **SPCFitNrmlPDFToHistogram** **(**`double` **binCenters**`[]`**,** `int` **binCenterCount,**
`int` **sigma,** `int` **numberofObservations,**
`double *`**normalPDFHeight);**

Purpose

Given the bin centers from a histogram (axis values output from the General Histogram function), the estimated sigma of the observations for the histogram, and the total number of observations in the histogram, calculates the height for a normal probability distribution function (PDF) that fits the histogram.

Example

```
SPCProcessMeanAndSigma(data, SAMP_COUNT, SAMP_SIZE, sigmaFromR,
   3.0, 2, &procMean, &procSigma, &uNPL, &lNPL);
SPCGeneralHistogram(data, SAMP_COUNT * SAMP_SIZE, NULL, NULL,
   NULL, 0, NULL, 0, &histogram, &axis, &binCount, &numOutside);
SPCFitNrmlPDFToHistogram (axis, binCount, procSigma, SAMP_COUNT *
   SAMP_SIZE, &normPDFHeight);
SPCDrawNormalPDFWithLimits(pan, PANEL_GRAPH, legendCtrl, uSpecLim,
   lSpecLim, procMean, procSigma, 50, normPDFHeight, 3.0, NULL);
```

Parameters

Input	**binCenters**	double-precision array	The x-axis values for the center of each histogram bin. Most often, this will be the Axis array from the General Histogram function.
	binCenterCount	integer	The number of items in the Bin Centers array.
	sigma	integer	The estimated sigma of the observations from which the histogram was created. See Also: `SPCProcessMeanAndSigma`
	numberofObservations	integer	The number of observations from which the histogram was created.
Output	**normalPDFHeight**	double-precision (passed by reference)	The height of the normal PDF that will fit the histogram. Use this value as an input to the Plot Normal PDF or Draw Normal PDF with Limits Functions.

Return Value

The result code for the function. Zero indicates success, negative values indicate errors.

SPCGeneralHistogram

```
int SPCGeneralHistogram (double x, int numberofElements, double *max,
                         double *min, int *numberofBins, int inclusion,
                         tSPCBinSpec *customBins, int customBinCount,
                         double **axis, int **histogram, int *outputBinCount,
                         tSPCNumOutside *numberOutside);
```

Purpose

Finds the discrete histogram of the input sequence X based on the given bin specifications.

Example

```
int *histogram, *axis; /* pointers to arrays */
int binCount;
tSPCNumOutside numOutside;

SPCGeneralHistogram(data, SAMP_COUNT * SAMP_SIZE, NULL,
       NULL, NULL, 0, NULL, 0, &histogram, &axis, &binCount,
       &numOutside);

SPCDrawHistogram(pan, PANEL_GRAPH, histogram, axis, binCount);

free(histogram);
free(axis);
```

Parameters

Input	**x**	double-precision	The input data.
	numberofElements	integer	The number of elements in the input data X.
	max	double-precision *	Points to the maximum value to include in the histogram. If NULL, the function will use the maximum value from the input data X.
	min	double-precision *	Points to the minimum value to include in the histogram. If NULL, the function will use the minimum value from the input data X.
	numberofBins	integer *	Points to the number of bins in the histogram. If NULL, the number of bins will be computed according to Sturges Rule: Number of bins = 1 + 3.3 log(sizeof(X))
	inclusion	integer	Specifies whether the boundaries of the bin are included in the bin. Valid values are: 0 - include the lower boundary. 1 - include the upper boundary. 2 - include both boundaries.
	customBins	tSPCBinSpec *	Points to an array of structures which specifies the boundaries of each bin of the histogram. If NULL, the Max, Min, Number of Bins and Inclusion are used instead to specify a set of uniformly spaced bins. Valid values are: 0 - include the lower boundary. 1 - include the upper boundary. 2 - include both boundaries. See *Parameter Discussion.*
	customBinCount	integer	The number of bin specifications in the Custom Bins array.

(continues)

Parameters (Continued)

Output	**axis**	double-precision * (passed by reference)	Contains the mid-point values of the intervals used to determine the histogram. The true plot of the histogram of the input array is obtained by plotting Axis vs. Histogram. **Note:** ***The array is allocated by this function. The caller must free the array when it is no longer needed***.
	histogram	pointer to integer (passed by reference)	Contains the histogram of the input array. The true plot of the histogram of the input array is obtained by plotting Axis vs. Histogram. **Note:** ***The array is allocated by this function. The caller must free the array when it is no longer needed.***
	outputBinCount	integer (passed by reference)	The number of bins represented in the Histogram and Axis arrays. This value is useful when the Number of Bins is NULL forcing the function to use Struges' Rule to calculate the number of bins.
	numberOutside	tSPCNumOutside (passed by reference)	Points to a structure specifying the number of elements of the input data X which do not fall in any bin. If NULL, this data is not returned. See *Parameter Discussion.*

Return Value

The result code for the function. Zero indicates success, negative values indicate errors.

Parameter Discussion

customBins—The elements of the structure are,

- `double` **lower**—lower boundary of the bin.
- `double` **upper**—upper boundary of the bin.
- `int` **inclusion**—specifies whether the boundaries of the bin are included in the bin.

numberOutside—The elements of the structure are,

- `int` **total**—The total number of points in X not falling in any bin.
- `int` **above**—The number of values in X above the upper boundary of the last bin.

- `int` **below**—The number of values in X below the lower boundary of the first bin.

Note: ***The values of above and below have meaning only if bins are specified such that* Bin[0].upper <= Bin[1].lower < Bin[1].upper, *and so on*.**

SPCGetChartPlotAttributes

```
int SPCGetChartPlotAttributes (int panel, int graphControl, int plotClass,
                               int *plotStyle, int *pointStyle, int *lineStyle,
                               int *color);
```

Purpose

Gets the plot attributes for a class of plot lines (for example Center Line or Upper Specification Limit). If the plot attributes of this graph have not been modified via `SetSPCPlotAttribute`, the default values for the class are returned.

Example

```
SPCGetChartPlotAttributes(pan, PANEL_GRAPH, SPC_PLOT_ATTR_UCL,
      &plotStyle, &pointStyle, &lineStyle, &color);
/* ensure that LCL limit matches UCL */
SPCSetChartPlotAttributes(pan, PANEL_GRAPH, SPC_PLOT_ATTR_LCL,
      plotStyle, pointStyle, lineStyle, color);
```

Parameters

Input	**panel**	integer	The panel containing the destination graph control.
	graphControl	integer	The control id of the destination graph control.
	plotClass	integer	The class of the plot line. The *Parameter Discussion* lists the plots classes that this toolkit works with.
Output	**plotStyle**	integer (passed by reference)	The plot style of the plot.
	pointStyle	integer (passed by reference)	The point style of the plot.
	lineStyle	integer (passed by reference)	The line style of the plot.
	color	integer (passed by reference)	The color of the plot.

Return Value

The result code for the function. Zero indicates success, negative values indicate errors.

Parameter Discussion

Plot Class	Description
SPC_PLOT_ATTR_CENTER	Center line of control chart
SPC_PLOT_ATTR_UCL	Upper Control Limit
SPC_PLOT_ATTR_LCL	Lower Control Limit
SPC_PLOT_ATTR_POINTS	The data of interest
SPC_PLOT_ATTR_ZONE_C_POS	One Standard Error Zone
SPC_PLOT_ATTR_ZONE_C_NEG	
SPC_PLOT_ATTR_ZONE_B_POS	Two Standard Error Zone
SPC_PLOT_ATTR_ZONE_B_NEG	
SPC_PLOT_ATTR_ZONE_A_POS	Three Standard Error Zone
SPC_PLOT_ATTR_ZONE_A_NEG	
SPC_PLOT_ATTR_USL	Upper Specification Limit
SPC_PLOT_ATTR_LSL	Lower Specification Limit
SPC_PLOT_ATTR_SPEC_MEAN	Specification Mean
SPC_PLOT_ATTR_UNPL	Upper Natural Process Limit
SPC_PLOT_ATTR_LNPL	Lower Natural Process Limit
SPC_PLOT_ATTR_PROC_MEAN	Natural Process Mean
SPC_PLOT_ATTR_PROC_MEAN_NOSPEC	Natural Process Mean with no spec limits
SPC_PLOT_ATTR_UNPL_NOSPEC	Upper Natural Limit with no spec limits
SPC_PLOT_ATTR_LNPL_NOSPEC	Lower Natural Limit with no spec limits
SPC_PLOT_ATTR_TIER	Tier Chart Points
SPC_PLOT_ATTR_HISTOGRAM	Histogram Bars
SPC_PLOT_ATTR_PARETO_CUMULATIVE	Pareto Chart Cumulative Line
SPC_PLOT_ATTR_PARETO_BARS	Pareto Chart Bars
SPC_PLOT_ATTR_PDF	Probability Distribution Function

SPCGetErrorText

```
char *SPCGetErrorText (int errorCode);
```

Purpose

Returns the text associated with an error code.

Parameters

Input	**errorCode**	int	The error code returned by one of the SPC functions.

Return Value

The text associated with the error code.

SPCParetoCounter

```
int SPCParetoCounter (int causesArrayInteger[], char *causesArrayString[],
                      int causeCounts[], int arraySizes,
                      tSPCParetoValues **paretoValues,
                      int *paretoValuesCount, int *totalOccurrenceCount);
```

Purpose

Given an unsorted list of individual cause observations, or an unsorted list of cause types and the number of occurrences of each cause, sort the list from the cause with the largest number of occurrences to the smallest and compute Pareto statistics for each cause.

Example

```
SPCParetoCounter(NULL, causes, NULL, causeArraySize,
       &paretoValues, &paretoValCount, &totalCount);

SPCDrawParetoChart(pan, PANEL_GRAPH1, freqLegend, PANEL_GRAPH2,
       percentLegend, PANEL_CAUSELIST, paretoValues,
       paretoValCount);

/* must free memory allocated by SPCParetoCounter */
for (i = 0; i < paretoValCount; i++) {
       free(paretoValues[i]->causeStr;
       free(paretoValues[i]);
}
free(paretoValues);
```

Parameters

Input	**causesArrayInteger**	integer array	Array of integer cause codes. The array contains either an unsorted list of individual cause observations, or an unsorted list of cause types with the count for each cause in the Cause Counts array. If NULL, the function uses the Causes Array String array instead.
	causesArrayString	string	Array of cause strings. The array contains either an unsorted list of individual cause observations, or an unsorted list of cause types with the count for each cause in the Cause Counts array. If NULL, the function uses the Causes Array Integer array instead.
	causeCounts	integer array	The array of cause counts for the causes in Cause Array Integer or Cause Array String array. If NULL, the causes are assumed to be uncounted.
	arraySizes	integer	The number of elements in Causes Array Integer or Causes Array String. Also the number of elements in Cause Counts if used.
Output	**paretoValues**	tSPCParetoValues * (passed by reference)	An array of pointers to structures containing the Pareto statistics for each cause. See *Parameter Discussion.* **Note:** ***The array is allocated by this function. The caller must free the elements of the array, then free the array when it is no longer needed***.
	paretoValuesCount	integer (passed by reference)	The number of elements in the Pareto Values array.
	totalOccurrenceCount	integer (passed by reference)	The total number of occurrences of all causes.

Return Value

The result code for the function. Zero indicates success, negative values indicate errors.

Parameter Discussion

paretoValues—The elements of the structure are,

- `char *` **causeStr**—The cause string. NULL if using integer cause codes.
- `int` **causeInt**—The cause code. NULL if using cause strings.
- `int` **frequency**—The number of occurrences of the cause.
- `int` **cumFrequency**—The total of occurrences of this cause and all previous causes.
- `int` **percentOfTotal**—The percent contribution of cause.
- `int` **cumPercent**—The total of percentages of this cause and all previous causes.

SPCPlotNormalPDF

`int` **SPCPlotNormalPDF** **(**`double` **processMean,** `double` **processSigma,**
`int` **numberPDFpoints,** `double` **PDFHeight,**
`double` **PDFWidth,** `double` **x[], double y[]);**

Purpose

Given the process mean and process sigma, outputs arrays containing the XY values of the normal probability distribution function (PDF).

Note: ***You can plot and draw the Normal PDF in one step with*** `SPCDrawNormalPDFWithLimits`**.**

Example

```
SPCGeneralHistogram(data, SAMP_COUNT * SAMP_SIZE, &max, &min,
       NULL, 0, NULL, 0, &histogram, &axis, &binCount,
       &numOutside);
SPCFitNrmlPDFToHistogram (axis, binCount, procSigma,
       SAMP_COUNT * SAMP_SIZE, &normPDFHeight);
SPCPlotNormalPDF (procMean, procSigma, 50, normPDFHeight,
       3.0, &x, &y);
```

```
PlotXY (pan, PANEL_GRAPH1, x, y, 50, VAL_DOUBLE, VAL_DOUBLE,
       VAL_THIN_LINE, VAL_EMPTY_SQUARE, VAL_SOLID, 1, VAL_BLACK);
```

Parameters

Input	**processMean**	double-precision	The estimated process mean. See Also: `SPCProcessMeanAndSigma`
	processSigma	double-precision	The estimated process sigma.
	numberPDFpoints	integer	The number of points to create for the normal PDF plot. The default value is 50.
	PDFHeight	double-precision	The height to draw the PDF. If zero, the function will draw the PDF with a height proportional to 1/(process sigma), corresponding to an area of one under the PDF curve. When fitting a normal PDF to a histogram, use the PDF Height output from the Fit Normal PDF to Histogram function.
	PDFWidth	double-precision	The width to draw the PDF in terms of +/- sigma. The default value is 3.0.
	x	double-precision array	Array containing the X value of each normal PDF plot point. The array must contain at least Number of PDF Points elements. **Note:** ***The array must be supplied by the caller***.
	y	double-precision array	Array containing the Y value of each normal PDF plot point. The array must contain at least Number of PDF Points elements. **Note:** ***The array must be supplied by the caller***.

Return Value

The result code for the function. Zero indicates success, negative values indicate errors.

SPCProcessMeanAndSigma

```
int SPCProcessMeanAndSigma (void *samplesX, int sampleCount, int sampleSize,
                            int type, double sigmaMultiplier,
                            int nformovingRange, double *processMean,
                            double *processSigma, double *upperNPL,
                            double *lowerNPL);
```

Purpose

Computes process mean and sigma as well as upper and lower natural process limits from process samples. You can estimate the process sigma in several ways. If the sample size is greater than one you can use either sample standard deviation s or range R to estimate the process sigma. If Sample Size is one, the function automatically uses the moving range to estimate the process sigma.

Example

```
SPCProcessMeanAndSigma(data, SAMP_COUNT, SAMP_SIZE,
      SPC_PROCESS_SIGMA_FROM_S, 3.0, 2, &procMean,
      &procSigma, &uNPL, &lNPL);
```

Parameters

Input	**samplesX**	void *	Samples on which to compute Process Mean and Process Sigma. Each row is a sample with n observations where n is the number of columns in the 2d array. If Sample Size is one, a 1D array will be correctly interpreted.
	sampleCount	integer	The number of samples in samples X.
	sampleSize	integer	The number of observations in each sample in samples X.
	type	integer	Selects the type of process sigma computation to perform. If 1, the process sigma estimate is based on mean sample standard deviation. If 0, the process sigma estimate is based on mean sample range. The default value is 1. **Note:** ***If Sample Size is 1, the process sigma estimate is based on mean moving range.***

(continues)

Parameters (Continued)

	sigmaMultiplier	double-precision	The sigma multiplier for calculating upper NPL and lower NPL. The default value is 3.0.
	nformovingRange	integer	The number of samples to include in the moving Range when Sample Size is 1. This value is ignored if the Sample Size is not 1. This value may range from 2 to 25. The default value is 2.
Output	**processMean**	double-precision (passed by reference)	Mean of the process, estimated from X-bar-bar (or X-bar if Sample Size is 1).
	processSigma	double-precision (passed by reference)	The sigma of the process, estimated from, $\frac{\text{s - bar}}{\text{c4}}$, if type is s $\frac{\text{R - bar}}{\text{d2}}$, if type is R $\frac{\text{mR - bar}}{\text{d2}}$, if Sample Size is 1.
	upperNPL	double-precision (passed by reference)	The upper natural process limit of the process computed as, process mean + sigma multiplier * process sigma.
	lowerNPL	double-precision (passed by reference)	The lower natural process limit of the process computed as, process mean - sigma multiplier * process sigma

Return Value

The result code for the function. Zero indicates success, negative values indicate errors.

SPCRuleCheckATTWE

```
int SPCRuleCheckATTWE (double points[], int numberofPoints, double CL,
                       double stderror, int ruleSelection, int rule4aor4b,
                       int **outofControlPoints, int *outofControlCount,
                       tSPCRuleViolation **ruleViolations,
                       int *violationCount);
```

Purpose

Applies the AT&T/Western Electric run rules to the input points array. Given a set of control chart points and the center line and std error from a control chart function, this function checks whether points are out of control (or non-randomly distributed) according to the rules enabled. See the Rule Selection parameter for more details on the AT&T/Western Electric run rules.

Example

```
int *OutOfControlPts;/* for out of control points */
int pointCount, ruleViolations;
tSPCRuleViolation *ruleViolations;

SPCXBarAndR (data, SAMP_COUNT, SAMP_SIZE, NULL, NULL,
       0, 3.00, NULL, &xbarbar, &samplesInCalc, Xbar,
       &limitsX, R, &limitsR, &processSigma);

SPCRuleCheckATTWE(Xbar, SAMP_COUNT, limitsX.CL,
       limitsX.stdErr, SPC_ATT_RULE1 + SPC_ATT_RULE3, 1,
       &OutOfControlPts, &pointCount, &ruleViolations,
       &violationCount);
/* process rule violations and out of control points */
...

/* free memory allocated by SPCRuleCheckATTWE */
free(OutOfControlPts)
for (i = 0; i < violationCount; i++)
       free(ruleViolations[i]);
free(ruleViolations);
```

Parameters

Input	**points**	double-precision array	The points plotted on the control chart, normally the output of one of the control chart functions such as an X-bar array.
	numberofPoints	integer	The number of points in the points array.
	CL	double-precision	The center line. Normally, this is the CL element of the limits structure returned by one of the control chart functions.
	stderror	double-precision	The standard error associated with CL. Normally, this is the stdErr element of the limits structure returned by one of the control chart functions.
	ruleSelection	integer	Specifies the run rule to apply to the points array. The *Parameter Discussion* lists the values you can use.
	rule4aor4b	integer	Selects rule 4a or 4b when rule 4 is selected. Rule 4a tests for seven consecutive points on one side of the center line. Rule 4b tests for eight consecutive points on one side of the center line.
Output	**outofControlPoints**	integer * (passed by reference)	An array listing the indices of the out of control points in the points input array. **Note:** ***The array is allocated by this function. The caller must free the allocated memory when the array is no longer needed***.
	outofControlCount	integer (passed by reference)	The number of elements in the Out of Control array.
	ruleViolations	tSPCRuleViolation * (passed by reference)	Points to an array of structures which contains information about all the rule violations detected. See *Parameter Discussion.* **Note:** ***The array is allocated by this function. The caller must free the elements of the array, then free the array when it is no longer needed***.
	violationCount	integer (passed by reference)	The number of rule violations found.

Return Value

The result code for the function. Zero indicates success, negative values indicate errors.

Parameter Discussion

Constant	Value	Description
SPC_ATT_RULE1	1	One point outside 3 std errors.
SPC_ATT_RULE2	2	Two out of 3 consecutive points outside 2 std errors (Zone A or beyond)
SPC_ATT_RULE3	4	Four out of five consecutive points outside 1 std errors (Zone B or beyond)
SPC_ATT_RULE4	8	Seven or eight consecutive points on one side of the center line. The Rule 4a or 4b control selects the interpretation.
SPC_ATT_ALL	15	Enable all AT&T/Western Electric Rules

You can combine these constants by adding them together, for example, SPC_ATT_RULE1 + SPC_ATT_RULE4. The default value is SPC_ATT_ALL.

ruleViolations—The elements of the structure are,

- `char` **rule**`[3]`—the rule violated (1, 2, 3, 4a, 4b)
- `int` **firstPoint**—The index of the first point of the sequence of points which violate the given rule.
- `int` **runLength**—the length of the sequence of points which violate the given rule.

SPCRuleCheckNelson

`int` **SPCRuleCheckNelson** **(**`double` **points**`[]`**,** `int` **numberofPoints,** `double` **CL,**
`double` **stderror,** `int` **testSelection,**
`int **`**outofControlPoints,** `int *`**outofControlCount,**
`tSPCRuleViolation **`**ruleViolations,**
`int *`**violationCount);**

Purpose

Applies the Nelson tests to the input points array. Given a set of control chart points and the center line and std error from a control chart function, this function checks whether

points are out of control (or non-randomly distributed) according to the Nelson tests enabled. See the Rule Selection parameter for more details on the Nelson tests.

Example

```
int *OutOfControlPts;/* for out of control points */
int pointCount, ruleViolations;
tSPCRuleViolation *ruleViolations;

SPCXBarAndR (data, SAMP_COUNT, SAMP_SIZE, NULL, NULL,
       0, 3.00, NULL, &xbarbar, &samplesInCalc, Xbar,
       &limitsX, R, &limitsR, &processSigma);

SPCRuleCheckNelson(Xbar, SAMP_COUNT, limitsX.CL,
       limitsX.stdErr, SPC_NELSON_RULE5 + SPC_NELSON_RULE6,
       &OutOfControlPts, &pointCount, &ruleViolations,
       &violationCount);

/* process rule violations and out of control points */
...

/* free memory allocated by SPCRuleCheckNelson */
free(OutOfControlPts);
for (i = 0; i < violationCount; i++)
       free(ruleViolations[i]);
free(ruleViolations);
```

Parameters

Input	**points**	double-precision array	The points plotted on the control chart, normally the output of one of the control chart functions such as an X-Bar array.
	numberofPoints	integer	The number of points in the points array.
	CL	double-precision	The center line. Normally, this is the CL element of the limits structure returned by one of the control chart functions.

(continues)

Parameters (Continued)

	stderror	double-precision	The standard error associated with CL. Normally, this is the stdErr element of the limits structure returned by one of the control chart functions.
	testSelection	integer	Specifies the test(s) to apply to the points array. The *Parameter Discussion* lists the values you can use.
Output	**outofControlPoints**	integer *(passed by reference)	An array listing the indices of the out of control points found in the points input array. **Note:** ***The array is allocated by this function. The caller must free the allocated memory when the array is no longer needed***.
	outofControlCount	integer (passed by reference)	The number of out of control points found.
	ruleViolations	tSPCRuleViolation * (passed by reference)	Points to an array of structures which contains information about all the rule violations detected. See *Parameter Discussion.* **Note:** ***The array is allocated by this function. The caller must free the elements of the array, then free the array when it is no longer needed.***
	violationCount	integer (passed by reference)	The number of rule violations found.

Return Value

The result code for the function. Zero indicates success, negative values indicate errors.

Parameter Discussion

Constant	Value	Description
SPC_NELSON_RULE1	1	One point beyond Zone A (3 std errors)
SPC_NELSON_RULE2	2	Nine points in a row in Zone C (center line) or beyond
SPC_NELSON_RULE3	4	Six points in a row steadily increasing or decreasing
SPC_NELSON_RULE4	8	Fourteen points in a row alternating up and down
SPC_NELSON_RULE5	16	Two out of three points in a row in Zone A or beyond (outside 2 std errors)
SPC_NELSON_RULE6	32	Four out of five points in a row in Zone B or beyond (outside 1 std errors)
SPC_NELSON_RULE7	64	Fifteen points in a row in Zone C (within 1 std errors) above and below center line
SPC_NELSON_RULE8	128	Eight points in a row on both sides of the center line with none in Zone C (within 1 std errors)
SPC_NELSON_ALL	255	Enable all Nelson tests.

You can combine these constants by adding them together. The default value is SPC_NELSON_ALL.

ruleViolations—The elements of the structure are,

- `char` **rule**`[3]`—the test violated (1, 2, 3, 4, 5, 6, 7, or 8)
- `int` **firstPoint**—The index of the first point of the sequence of points which violate the given test.
- `int` **runLength**—the length of the sequence of points which violate the given test.

SPCSampleStatistics

```
int SPCSampleStatistics (double sampleX[], int sampleSize, double *median,
                         double *mean, double *samplestddev,
                         double *skewness, double *kurtosis);
```

Purpose

Computes statistics for a single sample of Sample Size individual observations.

Example

```
SPCSampleStatistics(sample, SAMP_SIZE, &sampMedian, &sampMean,
       &sampStdDev, &sampSkewness, &sampKurtosis);
```

Parameters

Input	**sampleX**	double-precision array	One sample consisting of Sample Size individual observations. Sample statistics are computed from this sample.
	sampleSize	integer	The number of individual observations in sample X.
Output	**median**	double-precision (passed by reference)	The center value of the points in sample X when sorted in ascending order. If the number of points in sample X is even, the median is the average of the center pair of points.
	mean	double-precision (passed by reference)	The average of the individual observations in sample X.
	samplestddev	double-precision (passed by reference)	The sample standard deviation computed as, $\sqrt{\frac{\text{sum } (x[i] - \bar{x})^2}{(n-1)^2}}$ Where *n* is the sample size.

(continues)

Parameters (Continued)

	skewness	double-precision (passed by reference)	The degree of asymmetry in the distribution of the points in sample X around the mean. A normal distribution has a skewness of 0. Skewness is computed as, $\frac{\text{sum}((x[i] - x)^3)}{n * (\text{sample std dev})^3}$ Where n is the sample size.
	kurtosis	double-precision (passed by reference)	The relative peakedness or flatness of the distribution of the points in sample X. A normal distribution has a kurtosis of 3. Kurtosis is computed as, $\frac{\text{sum}((x[i] - x)^4)}{n * (\text{sample std dev})^4}$ Where n is the sample size.

Return Value

The result code for the function. Zero indicates success, negative values indicate errors.

SPCSequenceChecker

`int` **SPCSequenceChecker** **(**`int` **pointExceedsBound**`[]`**,** `int` **numberofpoints,**
`int` **violationsPerRun,** `int` **runLength,**
`int` **violationStartPoint**`[]`**,** `int` **violationRunLength**`[]`**,**
`int` ***arraySizes);**

Purpose

Searches the Boolean Point Exceeds Bounds input array for at least Violations Per Run TRUE values within Run Length values. You can use this function as a building block for creating rule checker functions.

Example

```
int *startPoint, *runLengths int arraySizes;
/* check Nelson rule 7 ( 15 out of 15 points in Zone C ) */
upperLimit = CL + stdErr;
```

```
lowerLimit = CL - stdErr;
for (i = 0; i < pointCount; i++) {
      violations[i] = ((points[i] < upperLimit) &&
            (points[i] > lowerLimit));
}
SPCSequenceChecker(violations, pointCount, 15, 15,
      &startPoint, &runLengths, &arraySizes);
/* Process violation runs */
....
/* free memory allocated by SPCSequenceChecker */
free(startPoints);
free(runLengths);
```

Parameters

Input	**pointExceedsBound**	integer array	An array of Boolean values indicating which points in some other array exceeded some test bound.
	numberofpoints	integer	The number of items in the Point Exceeds Bound array.
	violationsPerRun	integer	The number of violations per Run Length that indicate that a rule has been violated.
	runLength	integer	The run length for the rule.
Output	**violationStartPoint**	integer array	Start point of each sequence violating the rule. **Note:** ***The array is allocated by this function. The caller must free the array when it is no longer needed***.
	violationRunLength	integer array	The length of each sequence violating the rule. **Note:** ***The array is allocated by this function. The caller must free the array when it is no longer needed***.
	arraySizes	integer (passed by reference)	The number of elements in the Violation Start Point and the Violation Run Length arrays.

Return Value

The result code for the function. Zero indicates success, negative values indicate errors.

SPCSetChartPlotAttributes

```
int SPCSetChartPlotAttributes (int panel, int graphControl, int plotClass,
                               int plotStyle, int pointStyle, int lineStyle, int color);
```

Purpose

Sets the plot attributes for a class of plot lines (for example Center Line or Upper Specification Limits). The new attributes apply to all subsequent plots of the given plot class on the specified graph control.

Example

```
/* Change upper and lower control limits to solid green */
/* lines. Don't change the plot style or point style */
SPCSetChartPlotAttributes(pan, FIRST_RANGE,
       SPC_PLOT_ATTR_UCL, -1, -1, VAL_SOLID, VAL_GREEN);
SPCSetChartPlotAttributes(pan, FIRST_XBAR,
       SPC_PLOT_ATTR_UCL, -1, -1, VAL_SOLID, VAL_GREEN);
```

Parameters

Input	**panel**	integer	The panel containing the destination graph control.
	graphControl	integer	The control id of the destination graph control.
	plotClass	integer	The class of the plot line. The *Parameter Discussion* lists the plot classes that this toolkit works with.
	plotStyle	integer	The new plot style. If negative, the attribute is not changed.
	pointStyle	integer	The new point style. If negative, the attribute is not changed.
	lineStyle	integer	The new line style. If negative, the attribute is not changed.
	color	integer	The new color. If negative, the attribute is not changed.

Return Value

The result code for the function. Zero indicates success, negative values indicate errors.

Parameter Discussion

Plot Class	Description
SPC_PLOT_ATTR_CENTER	Center line of control chart
SPC_PLOT_ATTR_UCL	Upper Control Limit
SPC_PLOT_ATTR_LCL	Lower Control Limit
SPC_PLOT_ATTR_POINTS	The data of interest
SPC_PLOT_ATTR_ZONE_C_POS	One Standard Error Zone
SPC_PLOT_ATTR_ZONE_C_NEG	
SPC_PLOT_ATTR_ZONE_B_POS	Two Standard Error Zone
SPC_PLOT_ATTR_ZONE_B_NEG	
SPC_PLOT_ATTR_ZONE_A_POS	Three Standard Error Zone
SPC_PLOT_ATTR_ZONE_A_NEG	
SPC_PLOT_ATTR_USL	Upper Specification Limit
SPC_PLOT_ATTR_LSL	Lower Specification Limit
SPC_PLOT_ATTR_SPEC_MEAN	Specification Mean
SPC_PLOT_ATTR_UNPL	Upper Natural Process Limit
SPC_PLOT_ATTR_LNPL	Lower Natural Process Limit
SPC_PLOT_ATTR_PROC_MEAN	Natural Process Mean
SPC_PLOT_ATTR_PROC_MEAN_NOSPEC	Natural Process Mean with no spec limits
SPC_PLOT_ATTR_UNPL_NOSPEC	Upper Natural Limit with no spec limits
SPC_PLOT_ATTR_LNPL_NOSPEC	Lower Natural Limit with no spec limits
SPC_PLOT_ATTR_TIER	Tier Chart Points
SPC_PLOT_ATTR_HISTOGRAM	Histogram Bars
SPC_PLOT_ATTR_PARETO_CUMULATIVE	Pareto Chart Cumulative Line
SPC_PLOT_ATTR_PARETO_BARS	Pareto Chart Bars
SPC_PLOT_ATTR_PDF	Probability Distribution Function

SPCSinglePtXBarRs

`int` **SPCSinglePtXBarRs** **(**`double` **sampleX**`[]`**,** `int` **sampleSize,** `double *`**xbar,** `double *`**r,** `double *`**s);**

Purpose

Computes single points for both X-bar and R or X-bar and s control charts. Computing single points is useful for plotting charts one point at a time while you are collecting samples. You must continue to use the X-bar and R or X-bar and s functions to calculate the control limits for the chart.

Example

```
/* calculate a single point for an X-bar and s chart */
SPCSinglePtXBarRs(moreData[more_i], SAMPLE_SIZE,
       &xbar, NULL, &s);

/* assumes control limits have been calculated elsewhere */
SPCDrawSinglePoint(panel, PANEL_GRAPH1, xbar, &limitsX, 1);
SPCDrawSinglePoint(panel, PANEL_GRAPH2, s, &limitss, 1);
```

Parameters

Input	**sampleX**	double-precision array	The sample on which to compute the control chart points. An array of 2 or more individual observations.
	sampleSize	integer	The sample/subgroup size. In other words, the number of observations Sample X.
Output	**xbar**	double-precision (passed by reference)	The mean of the input sample.
	r	double-precision (passed by reference)	The range of the input sample.
	s	double-precision (passed by reference)	The sample standard deviation of the input sample.

Return Value

The result code for the function. Zero indicates success, negative values indicate an error.

SPCSinglePtmXBarAndmR

```
int SPCSinglePtmXBarAndmR (double individualx, int sampleSize, double history[],
                           int *historyCount, double *mXbar, double *mR);
```

Purpose

Computes points and limits for individuals x and moving Range control charts or moving Average and moving Range control charts. Computing single points is useful for plotting control charts one point at a time while you are collecting samples. You must still use the x and mR or mX-Bar and mR functions to calculate the control limits for the charts. The function maintains the history of previous individuals for calculating the moving Average and moving Range.

Example

```
/* calculate a single point for and mX-bar and mR chart */
SPCSinglePtmXBarAndmR(anotherPt, SAMP_SIZE, history,
      &historyCount, &mXbar, &mR);

/* assumes control limits have been calculated elsewhere */
SPCDrawSinglePoint(panel, PANEL_GRAPH1, mXbar, &limitsX, 1);
SPCDrawSinglePoint(panel, PANEL_GRAPH2, mR, &limitsR, 1);
```

Parameters

Input	**individualx**	double-precision	Individual on which to compute the control chart points.
	sampleSize	integer	The number of consecutive individual values to use in the moving average and moving Range calculation. The value may range from 2 to 25. The default value is 2.
	history	double-precision array	Array of the previous Sample Size - 1 individuals. Previous values are required for moving Average and moving Range calculations. The function updates the History array (and History Count) automatically. The history array must contain at least Sample Size elements.

(continues)

Parameters (Continued)

Output	**historyCount**	integer (passed by reference)	The number of previous individuals in the History array. The function maintains the history of previous individuals for calculating the moving Average and moving Range. When calling this function for the first time, this variable normally points to the value zero.
	mXbar	double-precision (passed by reference)	The average of the current individual x and the previous Sample Size - 1 individual x values.
	mR	double-precision (passed by reference)	The range of the current individual x and the previous Sample Size - 1 individual x values.

Return Value

The result code for the function. Zero indicates success, positive values indicate special conditions, negative values indicate errors.

Note: ***Until sufficient (Sample Size) individuals collect in the History array, the Result Code of this function is*** `SPC_NEED_MORE_INDIVIDUALS` ***(1) rather than*** `SPC_SUCCESS` ***(0).***

SPCXAndmR

```
int SPCXAndmR (void *individualsx, int individualsCount, int sampleSize,
               tSPCIndexSpec *indexSpec, int indicestoIgnore[],
               int numbertoIgnore, double stdErrorMultiplier,
               tSPCXAndmRLimitSrc *limitSource, int *indivsinLimitCalc,
               double *xBar, tSPCChartLimits *xLimits,
               double mRArray[], tSPCChartLimits *mRLimits,
               double *processSigma);
```

Purpose

Computes points and limits for individuals x and moving Range control charts. These are control charts for showing mean and moving Range.

Example

```
SPCXAndmR (data, INDIV_COUNT, SAMP_SIZE, NULL, NULL, 0,
       3.00, NULL, &indivInCalc, &xbar, &limitsX, mR,
       &limitsR, &processSigma);
SPCDrawControlChart(pan, PANEL_GRAPH1, xLegendCtrl, data,
       INDIV_COUNT, 0, 0, &limitsX, 1);
SPCDrawControlChart(pan, PANEL_GRAPH2, mRLegendCtrl, mR,
       INDIV_COUNT, 0, 0, &limitsR, 1);
```

Parameter

Input	**individualsx**	void *	Individual observations (or samples of subgroup size 1) on which to compute control limits.
	individualsCount	integer	The number of samples in Individuals x.
	sampleSize	integer	The number of consecutive individual values to use in the moving Range calculation. The value may range from 2 to 25. The default value is 2.
	indexSpec	tSPCIndexSpec *	Points to a structure which specifies the range of individuals to use for the control limits calculation. If NULL, the function uses all the individuals in the Individuals x input array in the control limits calculation. The default is NULL. The elements of the structure are, `int` **startIndex**—The index of the first individual to include in the control limit calculation. `int` **endIndex**—The index of the last individual to include in the control limit calculation. If zero, selects the last individual.
	indicestoIgnore	integer array	An array of the indices of individuals to exclude from the control limit calculation. This is useful for eliminating out-of-control points from the control limit calculation. If NULL, no individuals are excluded. The default value is NULL. **Note:** ***The samples are excluded from the limit calculation only. The function does calculate moving Range values for the excluded samples***.

(continues)

Parameters (Continued)

	numbertoIgnore	integer	The number of indices in the Indices to Ignore array. The default value is zero.
	stdErrorMultiplier	double-precision	The multiplier to use for the upper and lower control limits. The default value is 3.0, other values should only be used in special cases.
	limitSource	tSPCXAndmR-LimitSrc *	Points to a structure which specifies whether or not to use standard values for the chart limit calculations. If NULL, this function calculates the chart limits from the data in the individuals x array. The default value is NULL. See *Parameter Discussion*.
	indivsinLimitCalc	integer (passed by reference)	The number of individuals used to calculate the control chart limits.
	xBar	double-precision (passed by reference)	An estimate of the process mean based on the grand average of the individuals included in the control limit calculation. If the control limits are calculated from standard values, this is set to the standard mean.
	xLimits	tSPCChartLimits (passed by reference)	Points to a structure containing the limits for the X chart. If NULL, the X control limits are not calculated. See *Parameter Discussion*.
	mRArray	double-precision array	The moving range of individuals, {x[i],...,x[i-n+1]} where n is the sample size. This is the array plotted on the mR control chart. The caller must allocate this array. The array must contain at least SampleCount—SampleSize + 1 elements.
	mRLimits	tSPCChartLimits (passed by reference)	Points to a structure containing the control limits for the mR chart. If NULL, the control limits are not calculated. See *Parameter Discussion.*
	processSigma	double-precision (passed by reference)	An estimate (mR-bar/d2) of the process sigma (standard deviation) based on the average of the moving range between individuals included in the control limit calculation. If the control limits are calculated from standard values, this is set to standard R0/d2 or standard sigma.

Return Value

The result code for the function. Zero indicates success, a negative value indicates an error.

Parameter Discussion

limitSource—When using standard values the center line for the x control chart, x-bar, is set to std mean, and the center line for the mR control chart, mR-bar, is set to std R0 or std sigma * d2. The elements of the structure are,

- `int` **source**—Selects one of three sources for chart limits calculations:
  - 0 - From data. The chart limits are calculated from the data in the Individuals x array.
  - 1 - Use std mean, R0. The chart limits are calculated from standard values for mean and range.
  - 2 - Use std mean, sigma. The chart limits are calculated from standard values for mean and sigma.
- `double` **stdMean**—The standard mean value to use when calculating chart limits from standard values.
- `double` **sigmaOrR0**—The standard R0 or sigma value to use in calculating chart limits from standard values. Set this value to R0 if source is 1 or sigma if source is 2.

xLimits—The elements of the structure are,

- `double` **UCL**—The upper control limit for the chart.
- `double` **CL**—The center line for the chart. **CL** = x-bar or standard mean and is also the estimated process mean, if calculated from the input individuals.
- `double` **LCL**—The lower control limit for the chart.
- `double` **stdErr**—The standard error associated with CL

mRLimits—The elements of the structure are,

- `double` **UCL**—The upper control limit for the chart.
- `double` **CL**—The center line for the chart. CL = mR-bar if calculated from the input data individuals, or is standard R0 or standard sigma * d2.
- `double` **LCL**—The lower control limit for the chart.
- `double` **stdErr**—The standard error associated with CL

SPCXBarAndR

```
int SPCXBarAndR (void *samplesX, int sampleCount, int sampleSize,
                 tSPCIndexSpec *indexSpec, int indicestoIgnore[],
                 int numbertoIgnore, double stdErrorMultiplier,
                 tSPCXBarAndRLimitSrc *limitSource, double *xbarbar,
                 int *sampsinLimitCalc, double xbarArray[],
                 tSPCChartLimits *xbarLimits, double Rarray[],
                 tSPCChartLimits *RLimits, double *processSigma);
```

Purpose

Computes points and limits for X-bar and Range control charts. These are control charts for showing mean and Range.

Note: ***The Range Chart is limited to samples or subgroups of 25 observations or less, because for sample size of ten or more, the sample range loses efficiency of sample variance. For sample sizes of ten or larger, the X-bar and s chart is recommended over the X-bar and Range chart.***

Example

```
SPCXBarAndR (data, SAMP_COUNT, SAMP_SIZE, NULL, NULL, 0,
       3.00, NULL, &xbarbar, &samplesInCalc, Xbar, &limitsX,
       R, &limitsR, &processSigma);

SPCDrawControlChart(pan, PANEL_GRAPH1, LegendCtrl, Xbar,
       SAMP_COUNT, 0, 0, &limitsX, 1);

SPCDrawControlChart(pan, PANEL_GRAPH2, LegendCtrl, R,
       SAMP_COUNT, 0, 0, &limitsR, 1);
```

Parameters

Input	**samplesX**	void *	A 2D array of samples on which to compute control limits and points for the X-Bar and Range charts. Each row is a sample with n observations where n is the number of columns in the 2D array. N is also known as the subgroup size. The minimum sample/subgroup size is 2, the maximum is 25.
	sampleCount	integer	The number of samples in Samples X.
	sampleSize	integer	The sample subgroup size. In other words, the number of observations in each sample in Samples X.
	indexSpec	tSPCIndexSpec *	Points to a structure which specifies the range of samples to use for the control limits calculation. If NULL, the function uses all the samples in the Samples X input array in the control limit calculation. The default is NULL. The elements of the structure are, `int` **startIndex**—The index of the first sample to include in the control limit calculation. `int` **endIndex**—The index of the last sample to include in the control limit calculation. If zero, selects the last sample.
	indicestoIgnore	integer array	An array of the indices of samples to exclude from the control limit calculation. This is useful for eliminating out-of-control points from the control limit calculation. If NULL, no samples are excluded. The default value is NULL. **Note:** ***The samples are excluded from the limit calculation only. The function does calculate X-Bar and R values for the excluded samples.***
	numbertoIgnore	integer	The number of indices in the Indices to Ignore array. The default value is zero.

(continues)

Parameters (Continued)

	stdErrorMultiplier	double-precision	The multiplier to use for the upper and lower control limits. The default value is 3.0, other values should only be used in special cases.
	limitSource	tSPCXBarAndR-LimitSrc *	Points to a structure which specifies whether or not to use standard values for the chart limit calculations. If NULL, this function calculates the chart limits from the data in the samples array X. The default value is NULL. See *Parameter Discussion*.
Output	**xbarbar**	double-precision (passed by reference)	An estimate of the process mean based on the grand average of the samples included in the control limit calculation. If the control limits are calculated from standard values, this is set to the standard mean.
	sampsinLimitCalc	integer (passed by reference)	The number of samples used to calculate the control chart limits.
	xbarArray	double-precision array	The mean of each input sample. This is the array plotted on the X-Bar control chart. The caller must allocate this array. The array must contain at least Sample Count elements.
	xbarLimits	tSPCChartLimits (passed by reference)	Points to a structure containing the control limits for the X-Bar chart. If NULL, the control limits are not calculated. See *Parameter Discussion*.
	Rarray	double-precision array	The range of each input sample. This is the array plotted on the R control chart. The caller must allocate this array. The array must contain at least Sample Count elements.
	Rlimits	tSPCChartLimits (passed by reference)	Points to a structure containing the control limits for the R chart. If NULL, the control limits are not calculated. See *Parameter Discussion*.
	processSigma	double-precision (passed by reference)	An estimate (R-bar/d4) of the process sigma (standard deviation) based on the average standard deviation of the samples included in the control limit calculation. If the control limits are calculated from standard values, this is set to standard R0/d2 or standard sigma.

Return Value

The result code for the function. Zero indicates success, negative values indicate errors.

Parameter Discussion

limitSource—When using standard values the center line for the X-Bar control chart, X-bar-bar, is set to std mean, and the center line for the R control chart, R-bar, is set to std R0 or std sigma * d2. The elements of the structure are,

- `int` **source**—Selects one of three sources for chart limits calculations:
  - 0 - From data. The chart limits are calculated from the data in the samples X array.
  - 1 - Use std mean, R0. The chart limits are calculated from standard values for mean and range.
  - 2 - Use std mean, sigma. The chart limits are calculated from standard values for mean and sigma.
- `double` **stdMean**—The standard mean value to use when calculating chart limits from standard values.
- `double` **sigmaOrR0**—The standard R0 or sigma value to use in calculating chart limits from standard values. Set this value to R0 if source is 1 or sigma if source is 2.

xbarLimits—The elements of the structure are,

- `double` **UCL**—The upper control limit for the chart.
- `double` **CL**—The center line for the chart. CL = X-bar-bar or standard mean and is also the estimated process mean, if calculated from the input samples
- `double` **LCL**—The lower control limit for the chart.
- `double` **stdErr**—The standard error associated with CL

Rlimits—The elements of the structure are,

- `double` **UCL**—The upper control limit for the chart.
- `double` **CL**—The center line for the chart. CL = R-bar if calculated from the input samples, or is standard R0 or standard sigma * d2.
- `double` **LCL**—The lower control limit for the chart.
- `double` **stdErr**—The standard error associated with CL

SPCXBarAnds

```
int SPCXBarAnds (void *samplesX, int sampleCount, int sampleSize,
                 tSPCIndexSpec *indexSpec, int indicestoIgnore[],
                 int numbertoIgnore, double stdErrorMultiplier,
                 tSPCXBarAndsLimitSrc *limitSource, double *xbarbar,
                 int *sampsinLimitCalc, double xbarArray[],
                 tSPCChartLimits **xbarLimits, double sArray[],
                 tSPCChartLimits **sLimits, double *processSigma);
```

Purpose

Computes points and limits for X-bar and s control charts. These are control charts for showing mean and sample standard deviation.

Example

```
SPCXBarAnds (data, SAMP_COUNT, SAMP_SIZE, NULL,
       NULL, 0, 3.00, NULL, &xbarbar, &samplesInCalc,
       Xbar, &limitsX, s, &limitss, &processSigma);

SPCDrawChartWithZones(pan, PANEL_GRAPH, LegendCtrlX, Xbar,
       SAMP_COUNT, 0, 0, &limitsX, 1, 3);

SPCDrawControlChart(pan, PANEL_GRAPH2, LegendCtrls, s,
       SAMP_COUNT, 0, 0, &limitss, 1);
```

Parameters

Input	**samplesX**	void *	A 2D array of samples on which to compute control limits and points for the X-Bar and s charts. Each row is a sample with n observations where n is the number of columns in the 2D array. N is also known as the subgroup size. The minimum sample/subgroup size is 2.
	sampleCount	integer	The number of samples in Samples X.
	sampleSize	integer	The sample subgroup size. In other words, the number of observations in each sample in Samples X.
	indexSpec	tSPCIndexSpec *	Points to a structure which specifies the range of samples to use for the control limits calculation. If NULL, the function uses all the samples in the Samples X input array in the control limit calculation. The default is NULL. The elements of the structure are, `int` **startIndex**—The index of the first sample to include in the control limit calculation. `int` **endIndex**—The index of the last sample to include in the control limit calculation. If zero, selects the last sample.
	indicestoIgnore	integer array	An array of the indices of samples to exclude from the control limit calculation. This is useful for eliminating out-of-control points from the control limit calculation. If NULL, no samples are excluded. The default value is NULL. **Note:** ***The samples are excluded from the limit calculation only. The function does calculate X-Bar and s values for the excluded samples.***
	numbertoIgnore	integer	The number of indices in the Indices to Ignore array. The default value is zero.

(continues)

Parameters (Continued)

	stdErrorMultiplier	double-precision	The multiplier to use for the upper and lower control limits. The default value is 3.0, other values should only be used in special cases.
	limitSource	tSPCXBarAnds-LimitSrc *	Points to a structure which specifies whether or not to use standard values for the chart limit calculations. If NULL, this function calculates the chart limits from the data in the samples X array. The default value is NULL. See *Parameter Discussion*.
Output	**xbarbar**	double-precision (passed by reference)	An estimate of the process mean based on the grand average of the samples included in the control limit calculation. If the control limits are calculated from standard values, this is set to the standard mean.
	sampsinLimitCalc	integer (passed by reference)	The number of samples used to calculate the control chart limits.
	xbarArray	double-precision array	The mean of each input sample. This is the array plotted on the X-Bar control chart. The caller must allocate this array. The array must contain at least Sample Count elements.
	xbarLimits	tSPCChartLimits * (passed by reference)	Points to a structure containing the control limits for the X-Bar chart. If NULL, the X-Bar control limits are not calculated. See *Parameter Discussion.*
	sArray	double-precision array	The sample standard deviation of each input sample. This is the array plotted on the s control chart. The caller must allocate this array. The array must contain at least Sample Count elements.
	sLimits	tSPCChartLimits * (passed by reference)	Points to a structure containing control limits for the s chart. If NULL, the control limits are not calculated. See *Parameter Discussion*.
	processSigma	double-precision (passed by reference)	An estimate (s-bar/c4) of the process sigma (standard deviation) based on the average standard deviation of the samples included in the control limit calculation. If the control limits are calculated from standard values, this is set to standard s0/c4 or standard sigma.

Return Value

The result code for the function. Zero indicates success, negative values indicate errors.

Parameter Discussion

limitSource—When using standard values the center line for the X-Bar control chart, X-bar-bar, is set to std mean, and the center line for the s control chart, s-bar, is set to std s0 or std sigma * c4. The elements of the structure are,

- `int` **source**—Selects one of three sources for chart limits calculations:
  - 0 - From data. The chart limits are calculated from the data in the samples X array.
  - 1 - Use std mean, s0. The chart limits are calculated from standard values for mean and sample standard variation.
  - 2 - Use std mean, sigma. The chart limits are calculated from standard values for mean and sigma.
- `double` **stdMean**—The standard mean value to use when calculating chart limits from standard values.
- `double` **sigmaOrS0**—The standard s0 or sigma value to use in calculating chart limits from standard values. Set this value to s0 if source is 1 or sigma if source is 2

xbarLimits—The elements of the structure are,

- `double` **UCL**—The upper control limit for the chart.
- `double` **CL**—The center line for the chart. CL = X-bar-bar or standard mean and is also the estimated process mean, if calculated from the input samples
- `double` **LCL**—The lower control limit for the chart.
- `double` **stdErr**—The standard error associated with CL

sLimits—The elements of the structure are,

- `double` **UCL**—The upper control limit for the chart.
- `double` **CL**—The center line for the chart. CL = s-bar if calculated from the input samples, or is standard s0 or standard sigma * c4.
- `double` **LCL**—The lower control limit for the chart.
- `double` **stdErr**—The standard error associated with CL

SPCc

```
int SPCc (double c[], int sampleCount, tSPCIndexSpec *indexSpec,
          int indicestoIgnore[], int numbertoIgnore, double stdErrorMultiplier,
          tSPCXBarAndsLimitSrc *limitSource, int *samplesinCalc,
          tSPCChartLimits *cLimits);
```

Purpose

Computes points and limits for c chart, a control chart for the number of non-conformities or defects per sample inspected.

Example

```
SPCc(blemishes, SAMP_COUNT, NULL, NULL, 0, 3.0, NULL,
       &samplesInCalc, &limits);

SPCDrawControlChart (pan, PANEL_GRAPH, legend, blemishes,
       SAMP_COUNT, 0, 0, &limits, 1);
```

Parameters

Input	**c**	double-precision array	Array containing the number of non-conformities or defects in each sample inspected. This is also the array plotted on the c control chart.
	sampleCount	integer	The number of samples.
	indexSpec	tSPCIndexSpec *	Points to a structure which specifies the range of samples to use for the controls limits calculation. If NULL, the function uses all the samples in the c input array in the control limit calculation. The default is NULL. The elements of the structure are, `int` **startIndex**—The index of the first sample to include in the control limit calculation. `int` **endIndex**—The index of the last sample to include in the control limit calculation. If zero, selects the last sample.

(continues)

Parameters (Continued)

	indicestoIgnore	integer array	An array of the indices of samples to exclude from the control limit calculation. This is useful for eliminating out-of-control points from the control limit calculation. If NULL, no samples are excluded. The default value is NULL.
	numbertoIgnore	integer	The number of indices in the Indices to Ignore array. The default value is zero.
	stdErrorMultiplier	double-precision	The multiplier to use for the upper and lower control limits. The default value is 3.0, other values should only be used in special cases.
	limitSource	tSPCXBarAnds-LimitSrc *	Points to the value for std c0 when calculating chart limits from standard values. If NULL, this function calculates the chart limits from the data in the input array(s). The default value is NULL. See *Parameter Discussion.*
Output	**samplesinCalc**	integer (passed by reference)	The number of samples used to calculate the control chart limits.
	cLimits	tSPCChartLimits (passed by reference)	Points to a structure containing the limits for the c chart. If NULL, the limits are not calculated. See *Parameter Discussion*.

Return Value

The result code for the function. Zero indicates success, negative values indicate an error.

Parameter Discussion

limitSource—When using standard values, the center line for the control chart, c-bar is set to c0. The elements of the structure are,

- `int` **source**—Selects one of two sources for chart limits calculations:
  - 0 - From data. The chart limits are calculated from the data in the input arrays as qualified by the index spec and indices to ignore inputs. (default selection)
  - 1 - Use std c0. The standard c0 value to use when calculating chart limits from standard values.

- double **stdc0**—The standard c0 value to use when calculating chart limits from standard values. The center line for the control chart c-bar will then be set to c0.

cLimits—The elements of the structure are,

- double **UCL**—The upper control limit for the chart.
- double **CL**—The center line for the chart. CL = c-bar or standard c0. c-bar is the estimated number of non-conformities per sample for the process if calculated from the input data.
- double **LCL**—The lower control limit for the chart.
- double **stdErr**—The standard error associated with CL.

SPCmXBarAndmR

int **SPCmXBarAndmR (**void ***individualsx,** int **individualsCount,** int **sampleSize,**
tSPCIndexSpec ***indexSpec,** int **indicestoIgnore**[],
int **numbertoIgnore,** double **stdErrorMultiplier,**
tSPCXBarAndRLimitSrc ***limitSource,** double ***mXbarbar,**
int ***indivsinLimitCalc,** double **mXbarArray**[],
tSPCChartLimits ***mXbarLimits,** double **mRArray**[],
tSPCChartLimits ***mRLimits,** double ***processSigma);**

Purpose

Computes points and limits for mX-bar and moving Range control charts. These are control charts for showing moving average and moving Range.

Example

```
SPCmXBarAndmR (data, SAMP_COUNT, SAMP_SIZE, NULL, NULL, 0,
       3.00, NULL, &xbarbar, &samplesInCalc, mXbar, &limitsX,
       mR, &limitsR, &processSigma);

SPCDrawChartWithZones(pan, PANEL_GRAPH1, mXbarLegendCtrl,
       mXbar, SAMP_COUNT-SAMP_SIZE, 0, 0, &sLimitsX, 1, 3);

SPCDrawControlChart(pan, PANEL_GRAPH2, mRLegendCtrl, mR,
       SAMP_COUNT-SAMP_SIZE, 0, 0, &sLimitsR, 1);
```

Parameters

Input	**individualsx**	void *	Individual observations (or samples of subgroup size 1) on which to compute control limits.
	individualsCount	integer	The number of individuals in Individuals x.
	sampleSize	integer	The number of consecutive individual values to use in the moving average and moving Range calculation. The value may range from 2 to 25. The default value is 2.
	indexSpec	tSPCIndexSpec *	Points to a structure which specifies the range of individuals to use for the control limits calculation. If NULL, the function uses all the individuals in the Individuals x input array in the control limits calculation. The default is NULL. The elements of the structure are, `int` **startIndex**—The index of the first individual to include in the control limit calculation. `int` **endIndex**—The index of the last individual to include in the control limit calculation. If zero, selects the last individual.
	indicestoIgnore	integer array	An array of the indices of individuals to exclude from the control limit calculation. This is useful for eliminating out-of-control points from the control limit calculation. If NULL, no individuals are excluded. The default value is NULL. **Note:** ***The samples are excluded from the limit calculation only. The function does calculate mX-Bar and mR values for the excluded samples.***
	numbertoIgnore	integer	The number of indices in the Indices to Ignore array. The default value is zero.
	stdErrorMultiplier	double-precision	The multiplier to use for the upper and lower control limits. The default value is 3.0, other values should only be used in special cases.

(continues)

Parameters (Continued)

	limitSource	tSPCXBarAndR-LimitSrc *	Points to a structure which specifies whether or not to use standard values for the chart limit calculations. If NULL, this function calculates the chart limits from the data in the individuals x array. The default value is NULL. See *Parameter Discussion*.
Output	**mXbarbar**	double-precision (passed by reference)	An estimate of the process mean based on the average of the moving average of the individuals included in the control limit calculation. If the control limits are calculated from standard values, this is set to the standard mean.
	indivsinLimitCalc	integer (passed by reference)	The number of samples used to calculate the control chart limits.
	mXbarArray	double-precision array	The moving average of individuals {x[i],...,x[i-n+1)]}, where n is the sample size. This is the array plotted on the mX-Bar control chart. The caller must allocate this array. The array must contain at least (SampleCount - SampleSize + 1) elements.
	mXbarLimits	tSPCChartLimits (passed by reference)	Points to a structure containing the control limits for the mX-Bar chart. If NULL, the control limits are not calculated. See *Parameter Discussion*.
	mRArray	double-precision array	The moving range of individuals, {x[i],...,x[i-n+1]} where n is the sample size. This is the array plotted on the mR control chart. The caller must allocate this array. The array must contain at least SampleCount - SampleSize + 1 elements.
	mRLimits	tSPCChartLimits (passed by reference)	Points to a structure containing the control limits for the mR chart. If NULL, the control limits are not calculated. See *Parameter Discussion*.
	processSigma	double-precision (passed by reference)	An estimate (mR-bar/d2) of the process sigma (standard deviation) based on the average of the moving ranges included in the control limit calculation. If the control limits are calculated from standard values, this is set to standard R0/d2 or standard sigma.

Return Value

The result code for the function. Zero indicates success, negative values indicate errors.

Parameter Discussion

limitSource—When using standard values the center line for the mX-Bar control chart, mX-bar-bar, is set to std mean, and the center line for the mR control chart, mR-bar, is set to std R0 or std sigma * d2. The elements of the structure are,

- `int` **source**—Selects one of three sources for chart limits calculations:
  - 0 - From data. The chart limits are calculated from the data in the Individuals x array.
  - 1 - Use std mean, R0. The chart limits are calculated from standard values for mean and range.
  - 2 - Use std mean, sigma. The chart limits are calculated from standard values for mean and sigma.
- `double` **stdMean**—The standard mean value to use when calculating chart limits from standard values.
- `double` **sigmaOrR0**—The standard R0 or sigma value to use in calculating chart limits from standard values. Set this value to R0 if source is 1 or sigma if source is 2.

mXbarLimits—The elements of the structure are,

- `double` **UCL**—The upper control limit for the chart.
- `double` **CL**—The center line for the chart. CL = mX-bar-bar or standard mean and is also the estimated process mean, if calculated from the input individuals.
- `double` **LCL**—The lower control limit for the chart.
- `double` **stdErr**—The standard error associated with CL

mRLimits—The elements of the structure are,

- `double` **UCL**—The upper control limit for the chart.
- `double` **CL**—The center line for the chart. CL = mR-bar if calculated from the input data individuals, or is standard R0 or standard sigma * d2.
- `double` **LCL**—The lower control limit for the chart.
- `double` **stdErr**—The standard error associated with CL

SPCnp

```
int SPCnp (double r[], int sampleCount, int sampleSize,
        tSPCIndexSpec *indexSpec, int indicestoIgnore[], int numbertoIgnore,
        double stdErrorMultiplier, tSPCXBarAndsLimitSrc *limitSource,
        int *samplesinCalc, tSPCChartLimits *npLimits);
```

Purpose

Computes points and limits for an np chart, a control chart for the number of non-conforming units per sample inspected.

Example

```
SPCnp(rejects, SAMP_COUNT, SAMP_SIZE, NULL, NULL, 0, 3.0,
      NULL, &samplesInCalc, &limits);
SPCDrawControlChart (pan, PANEL_GRAPH, legend, rejects,
      SAMP_COUNT, 0, 0, &limits, 1);
```

Parameters

Input	**r**	double-precision array	Array containing the number of units non-conforming in each sample inspected. This is also the array plotted on the np control chart.
	sampleCount	integer	The number of samples.
	sampleSize	integer	The number of units inspected per sample.
	indexSpec	tSPCIndexSpec *	Points to a structure which specifies the range of samples to use for the controls limits calculation. If NULL, the function uses all the samples in the r input array in the control limit calculation. The default is NULL. The elements of the structure are, `int` **startIndex**—The index of the first sample to include in the control limit calculation. `int` **endIndex**—The index of the last sample to include in the control limit calculation. If zero, selects the last sample.

(continues)

Parameters (Continued)

	indicestoIgnore	integer array	An array of the indices of samples to exclude from the control limit calculation. This is useful for eliminating out-of-control points from the control limit calculation. If NULL, no samples are excluded. The default value is NULL.
	numbertoIgnore	integer	The number of indices in the Indices to Ignore array. The default value is zero.
	stdErrorMultiplier	double-precision	The multiplier to use for the upper and lower control limits. The default value is 3.0, other values should only be used in special cases.
	limitSource	tSPCXBarAnds-LimitSrc *	Points to the value for std p0 when calculating chart limits from standard values. If NULL, this function calculates the chart limits from the data in the input array(s). The default value is NULL. See *Parameter Discussion.*
Output	**samplesinCalc**	integer (passed by reference)	The number of samples used to calculate the control chart limits.
	npLimits	tSPCChartLimits (passed by reference)	Points to a structure containing the limits for the np chart. If NULL, the limits are not calculated. See *Parameter Discussion*.

Return Value

The result code for the function. Zero indicates success, negative values indicate errors.

Parameter Discussion

limitSource—When using standard values, the center line for the control chart, n*p-bar is set to n*p0. The elements of the structure are,

- `int` **source**—Selects one of two sources for chart limits calculations:
  - 0 - From data. The chart limits are calculated from the data in the input arrays as qualified by the index spec and indices to ignore inputs. (default selection)
  - 1 - Use std p0. The standard p0 value to use when calculating chart limits from the standard value p0.

- `double` **stdp0**—The standard p0 value to use when calculating chart limits from standard values. The center line for the control chart n*p-bar will then be set to n*p0.

npLimits—The elements of the structure are,

- `double` **UCL**—The upper control limit for the chart.
- `double` **CL**—The center line for the chart. CL = np-bar or standard p0*n. np-bar is the estimated number non-conforming units for the process if calculated from the input data.
- `double` **LCL**—The lower control limit for the chart.
- `double` **stdErr**—The standard error associated with CL.

SPCp

`int` **SPCp** **(**`double` **r[], ** `int` **sampleCount, ** `int` **sampleSizeV[], ** `int` **sampleSizeC, ** `tSPCIndexSpec *` **indexSpec, ** `int` **indicestoIgnore[], ** `int` **numbertoIgnore, ** `double` **stdErrorMultiplier, ** `double *` **limitSource, ** `double` **p[], ** `int *` **samplesinCalc, ** `tSPCChartLimits *` **pChartLimits, ** `double` **UCL[], ** `double` **LCL[]);**

Purpose

Computes points and limits for a p chart, a control chart for the fraction of non-conforming units per sample inspected.

Example

```
SPCp(rejects, SAMP_COUNT, inspected, 0, NULL, NULL, 0, 3.0,
       NULL, p, &samplesInCalc, &limits, UCL, LCL);
SPCDrawChartWithVarLimits (pan, PANEL_GRAPH, legend, p,
       SAMP_COUNT, 0, 0, &limits, UCL, LCL);
```

Parameters

Input	**r**	double-precision array	Array containing the number of units non-conforming in each sample inspected.
	sampleCount	integer	The number of samples.
	sampleSizeV	integer array	An array containing the number of units inspected per sample for each value of r. Use this array if a variable number of units were inspected for each entry in r. The array length must be the same as r. If NULL, the value in Sample Size C is used as the sample size for all samples.
	sampleSizeC	integer	The constant number of units inspected per sample. Set this value if the same number of units were inspected for all samples If Sample Size V is not NULL, this value is ignored.
	indexSpec	tSPCIndexSpec *	Points to a structure which specifies the range of samples to use for the controls limits calculation. If NULL, the function uses all the samples in the r input array in the control limit calculation. The default is NULL. The elements of the structure are, `int` **startIndex**—The index of the first sample to include in the control limit calculation. `int` **endIndex**—The index of the last sample to include in the control limit calculation. If zero, selects the last sample.
	indicestoIgnore	integer array	An array of the indices of samples to exclude from the control limit calculation. This is useful for eliminating out-of-control points from the control limit calculation. If NULL, no samples are excluded. The default value is NULL. **Note:** ***The samples are excluded from the limit calculation only. The function does calculate p values for the excluded samples.***

(continues)

Parameters (Continued)

	numbertoIgnore	integer	The number of indices in the Indices to Ignore array. The default value is zero.
	stdErrorMultiplier	double-precision	The multiplier to use for the upper and lower control limits. The default value is 3.0, other values should only be used in special cases.
	limitSource	double-precision *	Points to the value for std p0 when calculating chart limits from standard values. If NULL, this function calculates the chart limits from the data in the input array(s). The default value is NULL. See *Parameter Discussion*.
Output	**p**	double-precision array	The number of units non-conforming divided by the number inspected. This is the fraction non-conforming in each input sample. This is the array plotted on the p control chart.
	samplesinCalc	integer (passed by reference)	The number of samples used to calculate the control chart limits.
	pChartLimits	tSPCChartLimits (passed by reference)	Points to a structure containing the average control limits for the p chart. If NULL, the average control limits are not calculated. See *Parameter Discussion*.
	UCL	double-precision array	Array for the variable upper control limits. If NULL, the variable upper control limits are not returned. If Std Error Multiplier is three, this will be p-bar + 3 standard errors. The standard error calculation varies with n. The array must contain at least Sample Count elements.
	LCL	double-precision array	Array for the variable lower control limits. If NULL, the variable lower control limits are not returned. If Std Error Multiplier is three, this will be p-bar - 3 standard errors. The standard error calculation varies with n. The array must contain at least Sample Count elements.

Return Value

The result code for the function. Zero indicates success, negative values indicate errors.

Parameter Discussion

limitSource—When using standard values, the center line for the control chart, p-bar is set to p0. The elements of the structure are,

- `int` **source**—Selects one of two sources for chart limits calculations:
  - 0 - From data. The chart limits are calculated from the data in the input arrays as qualified by the index spec and indices to ignore inputs. (default selection)
  - 1 - Use std p0. The standard p0 value to use when calculating chart limits from the standard value p0.
- `double` **stdp0**—The standard p0 value to use when calculating chart limits from standard values. The center line for the control chart p-bar will then be set to p0.

pChartLimits—The elements of the structure are,

- `double` **UCL**—The average of the variable upper control limit for the chart. If the number of units inspected per sample is constant, use this value for the upper control limit.
- `double` **CL**—The center line for the chart. CL = p-bar or standard p0*n. np-bar is the estimated number non-conforming units for the process if calculated from the input data.
- `double` **LCL**—The average of the variable lower control limit for the chart. If the number of units inspected per sample is constant, use this value for the lower control limit.
- `double` **stdErr**—The standard error associated with CL.

SPCu

`int` **SPCu** **(**`double` **c**`[]`**,** `int` **sampleCount,** `int` **sampleSizeV**`[]`**,** `int` **sampleSizeC,** `tSPCIndexSpec *`**indexSpec,** `int` **indicestoIgnore**`[]`**,** `int` **numbertoIgnore,** `double` **stdErrorMultiplier,** `tSPCXBarAndsLimitSrc *`**limitSource,** `double` **u**`[]`**,** `int *`**samplesinCalc,** `tSPCChartLimits *`**uChartLimits,** `double` **UCL**`[]`**,** `double` **LCL**`[]`**);**

Purpose

Computes points and limits for a u chart, a control chart for the fraction of non-conformities or defects per sample inspected.

Example

```
SPCu(blemishes, SAMP_COUNT, inspected, 0, NULL, NULL, 3.0,
       NULL, u, &samplesInCalc, &limits, UCL, LCL) ;
```

```
SPCDrawChartWithVarLimits (pan, PANEL_GRAPH, legend, u,
       SAMP_COUNT, 0, 0, &limits, UCL, LCL);
```

Parameters

Input	**c**	double-precision array	Array containing the number of non-conformities or defects in each sample inspected.
	sampleCount	integer	The number of samples.
	sampleSizeV	integer array	An array containing the number of units inspected per sample for each value of c. Use this array if a variable number of units were inspected for each entry in c. The array length must be the same as c. If NULL, the value in Sample Size C is used as the sample size for all samples.
	sampleSizeC	integer	The constant number of units inspected per sample. Set this value if the same number of units were inspected for all samples. If Sample Size V is not NULL, this value is ignored.
	indexSpec	tSPCIndexSpec *	Points to a structure which specifies the range of samples to use for the controls limits calculation. If NULL, the function uses all the samples in the c input array in the control limit calculation. The default is NULL. The elements of the structure are, `int` **startIndex**—The index of the first sample to include in the control limit calculation. `int` **endIndex**—The index of the last sample to include in the control limit calculation. If zero, selects the last sample.
	indicestoIgnore	integer array	An array of the indices of samples to exclude from the control limit calculation. This is useful for eliminating out-of-control points from the control limit calculation. If NULL, no samples are excluded. The default value is NULL. **Note:** ***The samples are excluded from the limit calculation only. The function does calculate u values for the excluded samples.***

(continues)

Parameters (Continued)

	numbertoIgnore	integer	The number of indices in the Indices to Ignore array. The default value is zero.
	stdErrorMultiplier	double-precision	The multiplier to use for the upper and lower control limits. The default value is 3.0, other values should only be used in special cases.
	limitSource	tSPCXBarAnds-LimitSrc *	Points to the value for u0 when calculating chart limits from standard values. If NULL, this function calculates the chart limits from the data in the input array(s). The default value is NULL. See *Parameter Discussion*.
Output	**u**	double-precision array	The number of defects divided by the number inspected. This is the average number of non-conformities or defects per unit inspected. This array is plotted on the u control chart. The array must contain at least Sample Count elements.
	samplesinCalc	integer (passed by reference)	The number of samples used to calculate the control chart limits.
	uChartLimits	tSPCChartLimits (passed by reference)	Points to a structure containing the average control limits for the u chart. If NULL, the control limits are not calculated. See *Parameter Discussion*.
	UCL	double-precision array	Array for the variable upper control limits. If NULL, the variable upper control limits are not returned. If Std Error Multiplier is three, this will be u-bar + 3 standard errors. The standard error calculation varies with n. The array must contain at least Sample Count elements.
	LCL	double-precision array	Array for the variable lower control limits. If NULL, the variable upper control limits are not returned. If Std Error Multiplier is three, this will be u-bar - 3 standard errors. The standard error calculation varies with n. The array must contain at least Sample Count elements.

Return Value

The result code for the function. Zero indicates success, a negative values indicates an error.

Parameter Discussion

limitSource—When using standard values, the center line for the control chart, u-bar is set to u0. The elements of the structure are,

- `int` **source**—Selects one of two sources for chart limits calculations:
  - 0 - From data. The chart limits are calculated from the data in the input arrays as qualified by the index spec and indices to ignore inputs. (default selection)
  - 1 - Use std u0. The standard u0 value to use when calculating chart limits from standard values.
- `double` **stdu0**—The standard p0 value to use when calculating chart limits from standard values. The center line for the control chart u-bar will then be set to u0.

uChartLimits—The elements of the structure are,

- `double` **UCL**—The average of the variable upper control limit for the chart. If the number of units inspected per sample is constant, use this value for the upper control limit.
- `double` **CL**—The center line for the chart. CL = u-bar or standard u0. u-bar is the estimated number non-conformities per sample for the process if calculated from the input data.
- `double` **LCL**—The average of the variable lower control limit for the chart. If the number of units inspected per sample is constant, use this value for the lower control limit.
- `double` **stdErr**—The standard error associated with CL.

Appendix A
Legend Control Function Reference

The SPC Toolkit lets you use legends with drawing functions. You can use the legend functions to create legend controls for the SPC functions and to modify the legends that appear by default. This appendix describes each function in the Legend Control library. The functions are listed in alphabetical order with a description of the function, C syntax of the function, a description of each parameter, and possible error codes.

The following function tree lists all functions that help you customize the legends in your SPC graphs and charts.

Legend Control	**Function Name**
Create Legend Control	***LGCreateLegendControl***
Convert Graph to Legend	***LGConvertGraphToLegend***
Insert Legend Item	***LGInsertLegendItem***
Insert Legend Item for Plot	***LGInsertLegendItemForPlot***
Delete Legend Item	***LGDeleteLegendItem***
Delete All Legend Items	***LGClearLegendCtrl***
Number of Legend Items	***LGNumberOfLegendItems***
Display Legend Items	***LGDisplayLegendItems***
Get Legend Control Attribute	***LGGetLegendCtrlAttribute***
Set Legend Control Attribute	***LGSetLegendCtrlAttribute***
Get Legend Item Attribute	***LGGetLegendItemAttribute***
Set Legend Item Attribute	***LGSetLegendItemAttribute***
Get Error Text	***LGGetErrorText***

The class provides functions for creating and manipulating legend controls. A legend control is a graph control which displays legend information.

The following functions descriptions are in alphabetical order.

LGClearLegendCtrl

int LGClearLegendCtrl (int panelHandle, int legendControl);

Purpose

Delete all legend items from the legend control.

Parameters

Input	**panelHandle**	integer	The panel id of the panel which contains the legend control.
	legendControl	integer	The control id of the legend control.

Return Value

The result code of the function. Zero (LG_SUCCESS) indicates success. A negative value indicates an error.

LGConvertGraphToLegend

int LGConvertGraphToLegend (int panelHandle, int graphtoConvert, int anchorControl, int relativePosition, int autoSize, int showSamples);

Purpose

Converts an existing graph control to a legend control.

Parameters

Input	**panelHandle**	integer	The panel ID of the panel which contains the legend control.
	graphtoConvert	integer	The control ID of the graph control to convert to a legend control. **Note:** ***When you convert a graph control into a legend, the graph control attribute,*** ATTR_DATA_MODE***, must be set to*** VAL_RETAIN**.**

(continues)

Parameters

	anchorControl	integer	The control ID of the anchor control. The legend control's position is set relative to the anchor control. If the anchor control is a graph or strip chart, the legend's background and border colors are taken from the graph or strip chart. **Note:** ***If the anchor control is not a valid control, the relative position parameter is ignored. You must set the position of the legend control with*** `SetCtrlAttribute`**.**
	relativePosition	integer	The position of the legend control with respect to the anchor control. Valid values appear in the *Parameter Discussion.*
	autoSize	integer	Specifies whether the legend control is automatically resized when legends are added or deleted.
	showSamples	integer	Specifies whether a sample plot is included with each legend item.

Return Value

The result code of the function. Zero (`LG_SUCCESS`) indicates success. A negative value indicates an error.

Parameter Discussion

`LG_POS_DONT_MOVE`	`0`
`LG_POS_ABOVE_LEFT`	`1`
`LG_POS_ABOVE_CENTER`	`2`
`LG_POS_ABOVE_RIGHT`	`3`
`LG_POS_RIGHT_TOP`	`4`
`LG_POS_RIGHT_CENTER`	`5`
`LG_POS_RIGHT_BOTTOM`	`6`
`LG_POS_BELOW_RIGHT`	`7`
`LG_POS_BELOW_CENTER`	`8`
`LG_POS_BELOW_LEFT`	`9`
`LG_POS_LEFT_BOTTOM`	`10`
`LG_POS_LEFT_CENTER`	`11`
`LG_POS_LEFT_TOP`	`12`

LGCreateLegendControl

`int` **LGCreateLegendControl** **(**`int` **panelHandle,** `int` **anchorControl,**
`int` **relativePosition,** `int` **autoSize,** `int` **showSamples,**
`int *`**newLegendControl);**

Purpose

Creates a legend control.

Parameters

Input	**panelHandle**	integer	The panel ID of the panel which contains the legend control.
	anchorControl	integer	The control ID of the anchor control. The legend control's position is set relative to the anchor control. If the anchor control is a graph or strip chart, the legend's background color and border color are taken from the graph or strip chart. **Note:** ***If the anchor control is not a valid control, the relative position parameter is ignored. You must set the position of the legend control with*** `SetCtrlAttribute`**.**
	relativePosition	integer	The position of the legend control with respect to the anchor control. Valid values appear in the *Parameter Discussion*.
	autoSize	integer	Controls whether the legend control is automatically resized when legends are added or deleted.
	showSamples	integer	Specifies whether a sample plot is included with each legend item.
Output	**newLegendControl**	integer (passed by reference)	The new legend control.

Return Value

The result code of the function. Zero (`LG_SUCCESS`) indicates success. A negative value indicates an error.

Parameter Discussion

LG_POS_DONT_MOVE	0
LG_POS_ABOVE_LEFT	1
LG_POS_ABOVE_CENTER	2
LG_POS_ABOVE_RIGHT	3
LG_POS_RIGHT_TOP	4
LG_POS_RIGHT_CENTER	5
LG_POS_RIGHT_BOTTOM	6
LG_POS_BELOW_RIGHT	7
LG_POS_BELOW_CENTER	8
LG_POS_BELOW_LEFT	9
LG_POS_LEFT_BOTTOM	10
LG_POS_LEFT_CENTER	11
LG_POS_LEFT_TOP	12

Note: ***If the anchor control is not a valid control, the relative position parameter is ignored. You must set the position of the legend control with*** `SetCtrlAttribute`.

LGDeleteLegendItem

`int` **LGDeleteLegendItem (**`int` **panelHandle,** `int` **legendControl,** `int` **deletePosition);**

Purpose

Delete the legend at the specified position from the legend control.

Parameters

Input	**panelHandle**	integer	The panel ID of the panel which contains the legend control.
	legendControl	integer	The control ID of the legend control.
	deletePosition	integer	The position of the legend item to delete. The position may be a number from 1 to the number of legend items in the legend control. To delete at the top of the legend control, the constant FRONT_OF_LIST may be used. To delete at the bottom of the legend control, the constant END_OF_LIST may be used.

Return Value

The result code of the function. Zero (LG_SUCCESS) indicates success. A negative value indicates an error.

LGDisplayLegendItems

`int` **LGDisplayLegendItems (**`int` **panelHandle,** `int` **legendControl);**

Purpose

Draws the currently specified legends on the legend control. This function is most useful when delayed update (LG_ATTR_DELAY_UPDATE) is active. Normally, the legend control is updated immediately whenever a legend item changes.

Note: ***The legends are also redrawn when delayed update is deactivated.***

Parameters

Input	**panelHandle**	integer	The panel ID of the panel which contains the legend control.
	legendControl	integer	The control ID of the legend control.

Return Value

The result code of the function. Zero (LG_SUCCESS) indicates success. A negative value indicates an error.

LGGetErrorText

```
char *LGGetErrorText    (int errorCode);
```

Purpose

Returns the text associated with an error code.

Parameters

Input	**errorCode**	integer	The error code returned by one of the Legend Control functions.

Return Value

The text associated with the error code.

LGGetLegendCtrlAttribute

```
int LGGetLegendCtrlAttribute (int panelHandle, int legendControl, int attribute, ...);
```

Purpose

Gets the value of an attribute of the legend control.

Parameters

Input	**panelHandle**	integer	The panel ID of the panel which contains the legend control.
	legendControl	integer	The control ID of the legend control.
	attribute	integer	Specifies the attribute whose value you want to get.
Output	**attributeValue**	any type (passed by reference)	The value of the legend control attribute. A table in the *Parameter Discussion* summarizes the type and value range for each attribute.

Return Value

The result code of the function. Zero (LG_SUCCESS) indicates success. A negative value indicates an error.

Parameter Discussion

Attribute and Corresponding Values

Attribute	Type	Value	Description
LG_ATTR_META_FONT	string	Meta font name	The meta font used when legends are drawn.
LG_ATTR_SHOW_SAMPLES	integer	Boolean	Specifies whether to include a sample plot with the legend.
LG_ATTR_REL_POS	integer	position constant	The position of the legend control with respect to the anchor control. The valid positions are: LG_POS_DONT_MOVE 0 LG_POS_ABOVE_LEFT 1 LG_POS_ABOVE_CENTER 2 LG_POS_ABOVE_RIGHT 3 LG_POS_RIGHT_TOP 4 LG_POS_RIGHT_CENTER 5 LG_POS_RIGHT_BOTTOM 6 LG_POS_BELOW_RIGHT 7 LG_POS_BELOW_CENTER 8 LG_POS_BELOW_LEFT 9 LG_POS_LEFT_BOTTOM 10 LG_POS_LEFT_CENTER 11 LG_POS_LEFT_TOP 12
LG_ATTR_AUTO_SIZE	integer	Boolean	Specifies whether to automatically resize the legend control when legend items are added or deleted.
LG_ATTR_OFFSET_FROM_ANCHOR	integer	pixel offset	Specifies the offset in pixels from the anchor control to the legend control.
LG_ATTR_ANCHOR_CTRL	integer	control id	Specifies the anchor control for the legend.

(continues)

Attribute and Corresponding Values (Continued)

LG_ATTR_GRAPH_BG_COLOR	integer	color	Specifies the graph background color. You can also set this attribute with SetCtrlAttribute.
LG_ATTR_PLOT_BG_COLOR	integer	color	Specifies the plot background color. You can also set this attribute with SetCtrlAttribute.
LG_ATTR_DELAY_UPDATE	integer	Boolean	Specifies whether to update the legend control automatically after each attribute change. If delayed update is active, you can force an update by calling LGDisplayLegends or deactivating delayed update.

LGGetLegendItemAttribute

int **LGGetLegendItemAttribute** (int **panelHandle,** int **legendControl,**
int **legendItemNumber,** int **attribute,...);**

Purpose

Gets the value of an attribute for an individual legend item in the legend control.

Parameters

Input	**panelHandle**	integer	The panel ID of the panel which contains the legend control.
	legendControl	integer	The control ID of the legend control.
	legendItemNumber	integer	The number of the legend item in the legend control. You can also use FRONT_OF_LIST and END_OF_LIST. **Note:** ***The first legend item appears at the top of the legend control.***

(continues)

Parameters (Continued)

	attribute	integer	Specifies the legend item attribute whose value you want to get. You can perform a get on the attributes listed in the *Parameter Discussion*.
Output	**attributeValue**	any type (passed by reference)	The value for the legend item attribute. The following table summarizes the type and value range for each attribute.

Return Value

The result code of the function. Zero (LG_SUCCESS) indicates success. A negative value indicates an error.

Parameter Discussion

Attribute and Corresponding Values

LG_ATTR_LEGEND_TEXT	string	legend text	The text of the legend item.
LG_ATTR_TEXT_COLOR	integer	color	The color of the legend item text.
LG_ATTR_PLOT_STYLE	integer	plot style	The plot style of the legend item sample plot.
LG_ATTR_POINT_STYLE	integer	point style	The point style of the legend item sample plot.
LG_ATTR_LINE_STYLE	integer	line style	The line style of the sample legend item plot.
LG_ATTR_PLOT_COLOR	integer	color	The color of the sample legend item plot.

LGInsertLegendItem

```
int LGInsertLegendItem (int panelHandle, int legendControl, int insertPosition,
                        char legendText[], int textColor, int plotStyle,
                        int pointStyle, int lineStyle, int plotColor);
```

Purpose

Inserts a legend in a legend control at the specified position.

Parameters

Input	**panelHandle**	integer	The panel ID of the panel which contains the legend control.
	legendControl	integer	The control ID of the legend control.
	insertPosition	integer	The position at which to insert the legend item. The first legend item appears at the top of the legend control. The position may be a number from 1 to the number of legend items in the legend control. To insert at the top of the legend control, the constant FRONT_OF_LIST may be used. To insert at the bottom of the legend control, the constant END_OF_LIST may be used.
	legendText	string	Specifies the text of the legend item.
	textColor	integer	Specifies the color of the legend item text.
	plotStyle	integer	The curve style to be used in the sample plot of the legend item. Valid styles are listed in the *Parameter Discussion.*
	pointStyle	integer	The point style to be used in the sample plot of the legend item. Valid styles are listed in the *Parameter Discussion.*
	lineStyle	integer	The line style to be used in the sample plot of the legend item. Valid styles are listed in the *Parameter Discussion.*
	plotColor	integer	Specifies the plot color of the sample plot of the legend item. An RGB value is a 4-byte integer with the hexadecimal format 0x00RRGGBB. RR, GG, and BB are the respective red, green, and blue components of the color value. The first sixteen colors listed are the sixteen standard colors. Predefined RGB Values are listed in the *Parameter Discussion*.

Return Value

The result code of the function. Zero (LG_SUCCESS) indicates success. A negative value indicates an error.

Parameter Discussion

plotStyle

thin line	VAL_THIN_LINE
connected points	VAL_CONNECTED_POINTS
scatter	VAL_SCATTER
vertical bar	VAL_VERTICAL_BAR
horizontal bar	VAL_HORIZONTAL_BAR
fat line	VAL_FAT_LINE
thin step	VAL_THIN_STEP
fat step	VAL_FAT_STEP
base zero vertical bar	VAL_BASE_ZERO_VERTICAL_BAR
base zero horizontal bar	VAL_BASE_ZERO_HORIZONTAL_BAR

Notes on the plotStyle Parameter

Thin Line—The data points are plotted as a sequence of connected line segments, drawn from point to point. The line segments are 1 pixel wide. No marker symbols are drawn at any of the data points.

Connected Points—Similar to "line" except that marker symbols are drawn at some or all of the data points, depending on the selected point frequency.

Scatter—Similar to "connected points" except that only points are drawn with no line segments between them.

Vertical Bar—The data points are plotted as a sequence of vertical bars anchored at the bottom of the plot area, where the height of the bar represents the magnitude of the data point.

Horizontal Bar—The data points are plotted as a sequence of horizontal bars anchored on the left edge of the plot area, where the length of the bar represents the magnitude of the data point.

Fat Line—The data points are plotted as a sequence of connected line segments, drawn from point to point. The line segments are 3 pixels wide. No marker symbols are drawn at any of the data points. On the Windows platform, Fat Line forces the Line Style to solid.

Thin Step—The data points are connected by two 1-pixel wide lines at right angles to each other (the vertical drawn first, then the horizontal.)

Fat Step—The data points are connected by two 3-pixel wide lines at right angles to each other (the vertical drawn first, then the horizontal.)

Base Zero Vertical Bar—Similar to (4) with the exception that the base of the bar is the Y-axis origin. Bars can then extend upwards or downwards depending on whether the data value is positive or negative.

Base Zero Horizontal Bar—Similar to (5) with the exception that the base of the bar is the X-axis origin. Bars can then extend to the right or to the left depending on whether the data value is positive or negative.

pointStyle

empty square	VAL_EMPTY_SQUARE
solid square	VAL_SOLID_SQUARE
asterisk	VAL_ASTERISK
dotted empty square	VAL_DOTTED_EMPTY_SQUARE
dotted solid square	VAL_DOTTED_SOLID_SQUARE
solid diamond	VAL_SOLID_DIAMOND
empty square with X	VAL_EMPTY_SQUARE_WITH_X
empty square with cross	VAL_EMPTY_SQUARE_WITH_CROSS
bold	XVAL_BOLD_X
small solid square	VAL_SMALL_SOLID_SQUARE
simple dot	VAL_SIMPLE_DOT
empty circle	VAL_EMPTY_CIRCLE
solid circle	VAL_SOLID_CIRCLE
dotted solid circle	VAL_DOTTED_SOLID_CIRCLE
dotted empty circle	VAL_DOTTED_EMPTY_CIRCLE
bold cross	VAL_BOLD_CROSS
cross	VAL_CROSS
small cross	VAL_SMALL_CROSS
X	VAL_X
small	X VAL_SMALL_X
dotted solid diamond	VAL_DOTTED_SOLID_DIAMOND
empty diamond	VAL_EMPTY_DIAMOND
dotted empty diamond	VAL_DOTTED_EMPTY_DIAMOND
small empty square	VAL_SMALL_EMPTY_SQUARE
no point	VAL_NO_POINT

lineStyle

solid	VAL_SOLID
das	VAL_DASH
dot	VAL_DOT
dash dot	VAL_DASH_DOT
dash dot dot	VAL_DASH_DOT_DOT

plotColor

```
          VAL_RED = 0xFF0000L
         VAL_BLUE = 0x0000FFL
        VAL_GREEN = 0x00FF00L
         VAL_CYAN = 0x00FFFFL
      VAL_MAGENTA = 0xFF00FFL
       VAL_YELLOW = 0xFFFF00L
       VAL_DK_RED = 0x800000L
      VAL_DK_BLUE = 0x000080L
     VAL_DK_GREEN = 0x008000L
      VAL_DK_CYAN = 0x008080L
   VAL_DK_MAGENTA = 0x800080L
    VAL_DK_YELLOW = 0x808000L
      VAL_LT_GRAY = 0xCCCCCCL
      VAL_DK_GRAY = 0x808080L
        VAL_BLACK = 0x000000L
        VAL_WHITE = 0xFFFFFFL
  VAL_PANEL_GRAY = VAL_LT_GRAY
         VAL_GRAY = 0xA0A0A0L
    VAL_OFFWHITE = 0xE5E5E5L
VAL_TRANSPARENT = 0x1000000L
```

Note: ***In addition to the predefined color palette values, You can also use the User Interface Library function,*** `MakeColor`***, to create an RGB value from red, green, and blue color components. To enter user-defined color values, select*** **Toggle Control Style** ***from the*** **Option** ***menu, then manually enter the color value.***

LGInsertLegendItemForPlot

```
int LGInsertLegendItemForPlot (int panelHandle, int legendControl,
                               int insertPosition, char legendText[], int textColor,
                               int plotHandle);
```

Purpose

Inserts a legend in a legend control at the specified position. The plot style, point style, line style and color of the sample plot are set to the same values as the specified plot handle.

Parameters

Input	**panelHandle**	integer	The panel ID of the panel which contains the legend control.
	legendControl	integer	The control ID of the legend control.
	insertPosition	integer	The position at which to insert the legend item. The first legend item appears at the top of the legend control. The position may be a number from 1 to the number of legends in the legend control. To insert at the top of the legend control, the constant FRONT_OF_LIST may be used. To insert at the bottom of the legend control, the constant END_OF_LIST may be used.
	legendText	string	The text of the legend item.
	textColor	integer	Specifies the color of the legend item text.
	plotHandle	integer	The handle of an existing plot on the anchor graph control.

Return Value

The result code of the function. Zero (LG_SUCCESS) indicates success. A negative value indicates an error.

LGNumberOfLegendItems

`int` **LGNumberOfLegendItems (**`int` **panelHandle,** `int` **legendControl,**
`int *` **numberofItems);**

Purpose

Get the number of legend items in the legend control.

Parameters

Input	**panelHandle**	integer	The panel id of the panel which contains the legend control.
	legendControl	integer	The control id of the legend control.
Output	**numberofItems**	integer (passed by reference)	The number of items in the legend control.

Return Value

The result code of the function. Zero (`LG_SUCCESS`) indicates success. A negative value indicates an error.

LGSetLegendCtrlAttribute

`int` **LGSetLegendCtrlAttribute (**`int` **panelHandle,** `int` **legendControl,** `int` **attribute, ...);**

Purpose

Sets an attribute of the legend control.

Parameters

Input	**panelHandle**	integer	The panel ID of the panel which contains the legend control.
	legendControl	integer	The control ID of the legend control.
	attribute	integer	Specifies the attribute whose value you want to set. Attributes that you can set are listed in the *Parameter Discussion.*
	attributeValue	any type (passed by value)	The new value for the legend control attribute. A table in the *Parameter Discussion* summarizes the type and value range for each attribute.

Return Value

The result code of the function. Zero (LG_SUCCESS) indicates success. A negative value indicates an error.

Parameter Discussion

Attribute and Corresponding Values

Attribute	Type	Value	Description
LG_ATTR_META_FONT	string	Meta font name	The meta font used when legends are drawn.
LG_ATTR_SHOW_SAMPLES	integer	Boolean	Specifies whether to include a sample plot with the legend.
LG_ATTR_REL_POS	integer	position constant	The position of the legend control with respect to the anchor control. The valid positions are: LG_POS_DONT_MOVE 0 LG_POS_ABOVE_LEFT 1 LG_POS_ABOVE_CENTER 2 LG_POS_ABOVE_RIGHT 3 LG_POS_RIGHT_TOP 4 LG_POS_RIGHT_CENTER 5 LG_POS_RIGHT_BOTTOM 6 LG_POS_BELOW_RIGHT 7 LG_POS_BELOW_CENTER 8 LG_POS_BELOW_LEFT 9 LG_POS_LEFT_BOTTOM 10 LG_POS_LEFT_CENTER 11 LG_POS_LEFT_TOP 12 **Note:** ***If the anchor control is not a valid control, the relative position attribute is ignored. You must set the position of the legend control with*** SetCtrlAttribute**.**

(continues)

Attribute and Corresponding Values (Continued)

LG_ATTR_AUTO_SIZE	integer	Boolean	Specifies whether to automatically resize the legend control when legends are added or deleted.
LG_ATTR_OFFSET_FROM_ANCHOR	integer	pixel offset	Specifies the offset in pixels from the anchor control to the legend control. **Note:** ***If the anchor control is not a valid control, the offset from anchor attribute is ignored. You must set the position of the legend control with*** SetCtrlAttribute**.**
LG_ATTR_ANCHOR_CTRL	integer	control id	Specifies the anchor control for the legend.
LG_ATTR_GRAPH_BG_COLOR	integer	color	Specifies the graph background color. You can also set this attribute with SetCtrlAttribute.
LG_ATTR_PLOT_BG_COLOR	integer	color	Specifies the plot background color. You can also set this attribute with SetCtrlAttribute.
LG_ATTR_DELAY_UPDATE	integer	Boolean	Specifies whether to update the legend control automatically after each attribute change. If delayed update is active, you can force an update by calling LGDisplayLegends or deactivating delayed update.

LGSetLegendItemAttribute

int **LGSetLegendItemAttribute (**int **panelHandle,** int **legendControl,**
int **legendItemNumber,** int **attribute, ...);**

Purpose

Sets an attribute for an individual legend in the legend control.

Parameters

Input	**panelHandle**	integer	The panel ID of the panel which contains the legend control.
	legendControl	integer	The control ID of the legend control.
	legendItemNumber	integer	The number of the individual legend in the legend control. You can also use FRONT_OF_LIST and END_OF_LIST. **Note:** ***The first legend appears at the top of the legend control***.
	attribute	integer	Specifies the attribute whose value you want to set. The attributes that can by set are listed in the *Parameter Discussion*.
	attributeValue	any type (passed by value)	The new value for the legend item attribute. A table in the *Parameter Discussion* summarizes the type and value range for each attribute.

Return Value

The result code of the function. Zero (LG_SUCCESS) indicates success. A negative value indicates an error.

Parameter Discussion

attributeValue

Attribute	Type	Value	Description
LG_ATTR_LEGEND_TEXT	string	legend text	The text of the legend item.
LG_ATTR_TEXT_COLOR	integer	color	The color of the legend item text.
LG_ATTR_PLOT_STYLE	integer	plot style	The plot style of the legend item sample plot.
LG_ATTR_POINT_STYLE	integer	point style	The point style of the legend item sample plot.
LG_ATTR_LINE_STYLE	integer	line style	The line style of the sample legend item plot.
LG_ATTR_PLOT_COLOR	integer	color	The color of the sample legend item plot.

Appendix B
Customer Communication

For your convenience, this appendix contains forms to help you gather the information necessary to help us solve technical problems you might have as well as a form you can use to comment on the product documentation. Filling out a copy of the *Technical Support Form* before contacting National Instruments helps us help you better and faster.

National Instruments provides comprehensive technical assistance around the world. In the U.S. and Canada, applications engineers are available Monday through Friday from 8:00 a.m. to 6:00 p.m. (central time). In other countries, contact the nearest branch office. You may fax questions to us at any time.

Electronic Services

Bulletin Board Support

National Instruments has BBS and FTP sites dedicated for 24-hour support with a collection of files and documents to answer most common customer questions. From these sites, you can also download the latest instrument drivers, updates, and example programs. For recorded instructions on how to use the bulletin board and FTP services and for BBS automated information, call (512) 795-6990. You can access these services at:

United States: (512) 794-5422 or (800) 327-3077
Up to 14,400 baud, 8 data bits, 1 stop bit, no parity

United Kingdom: 01635 551422
Up to 9,600 baud, 8 data bits, 1 stop bit, no parity

France: 1 48 65 15 59
Up to 9,600 baud, 8 data bits, 1 stop bit, no parity

FTP Support

To access our FTP site, log on to our Internet host, `ftp.natinst.com`, as anonymous and use your Internet address, such as `joesmith@anywhere.com`, as your password. The support files and documents are located in the `/support` directories.

FaxBack Support

FaxBack is a 24-hour information retrieval system containing a library of documents on a wide range of technical information. You can access FaxBack from a touch-tone telephone at the following numbers:

(512) 418-1111 or (800) 329-7177

E-Mail Support (currently U.S. only)

You can submit technical support questions to the appropriate applications engineering team through e-mail at the Internet addresses listed below. Remember to include your name, address, and phone number so we can contact you with solutions and suggestions.

GPIB:	`gpib.support@natinst.com`
DAQ:	`daq.support@natinst.com`
VXI:	`vxi.support@natinst.com`
LabVIEW:	`lv.support@natinst.com`
LabWindows:	`lw.support@natinst.com`
HiQ:	`hiq.support@natinst.com`
VISA:	`visa.support@natinst.com`

Fax and Telephone Support

National Instruments has branch offices all over the world. Use the list below to find the technical support number for your country. If there is no National Instruments office in your country, contact the source from which you purchased your software to obtain support.

	 Telephone	 Fax
Australia	03 9 879 9422	03 9 879 9179
Austria	0662 45 79 90 0	0662 45 79 90 19
Belgium	02 757 00 20	02 757 03 11
Canada (Ontario)	519 622 9310	519 622 9311
Canada (Quebec)	514 694 8521	514 694 4399
Denmark	45 76 26 00	45 76 71 11
Finland	90 527 2321	90 502 2930
France	1 48 14 24 24	1 48 14 24 14
Germany	089 741 31 30	089 714 60 35
Hong Kong	2645 3186	2686 8505
Italy	02 48301892	02 48301915
Japan	03 5472 2970	03 5472 2977
Korea	02 596 7456	02 596 7455
Mexico	5 202 2544	5 520 3282
Netherlands	03480 33466	03480 30673
Norway	32 84 84 00	32 84 86 00
Singapore	2265886	2265887
Spain	91 640 0085	91 640 0533
Sweden	08 730 49 70	08 730 43 70
Switzerland	056 20 51 51	056 20 51 55
Taiwan	02 377 1200	02 737 4644
U.K.	01635 523545	01635 523154

Technical Support Form

Photocopy this form and update it each time you make changes to your software or hardware, and use the completed copy of this form as a reference for your current configuration. Completing this form accurately before contacting National Instruments for technical support helps our applications engineers answer your questions more efficiently.

If you are using any National Instruments hardware or software products related to this problem, include the configuration forms from their user manuals. Include additional pages if necessary.

Name _____

Company _____

Address _____

_____

Fax (___)_____ Phone (___)_____

Computer brand _____ Model _____ Processor _____

Operating system (include version number _____

Clock Speed _____ MHz _____ RAM _____ MB _____ Display adapter _____

Mouse _____yes _____no Other adapters installed _____

Hard disk capacity _____MB Brand _____

Instruments used _____

National Instruments hardware product model _____ Revision _____

Configuration _____

National Instruments software product _____ Version _____

Configuration _____

The problem is _____

_____

_____

_____

List any error messages _____

_____

_____

The following steps will reproduce the problem _____

_____

_____

Hardware and Software Configuration Form

Record the settings and revisions of your hardware and software on the line to the right of each item. Complete a new copy of this form each time you revise your software or hardware configuration, and use this form as a reference for your current configuration. Completing this form accurately before contacting National Instruments for technical support helps our applications engineers answer your questions more efficiently.

[The information below is product-specific. Actual contents vary according to product. Check with your content expert and product manager.]

National Instruments Products

Data Acquisition Hardware Revision _____

Interrupt Level of Hardware _____

DMA Channels of Hardware _____

Base I/O Address of Hardware _____

NI-DAQ, LabVIEW, or
LabWindows Version _____

Other Products

Computer Make and Model _____

Microprocessor _____

Clock Frequency _____

Type of Video Board Installed _____

Operating System _____

Operating System Version _____

Operating System Mode _____

Programming Language _____

Programming Language Version _____

Other Boards in System _____

Base I/O Address of Other Boards _____

DMA Channels of Other Boards _____

Interrupt Level of Other Boards _____

Documentation Comment Form

National Instruments encourages you to comment on the documentation supplied with our products. This information helps us provide quality products to meet your needs.

Title: LabWindows®/CVI SPC Toolkit Reference Manual

Edition Date: January 1996

Part Number: 321062A-01

Please comment on the completeness, clarity, and organization of the manual.

_____

_____

_____

_____

_____

_____

_____

If you find errors in the manual, please record the page numbers and describe the errors.

_____

_____

_____

_____

_____

_____

_____

Thank you for your help.

Name _____

Title _____

Company _____

Address _____

_____

Phone (_____) _____

Mail to: Technical Publications
National Instruments Corporation
6504 Bridge Point Parkway, MS 53-02
Austin, TX 78730-5039

Fax to: Technical Publications
National Instruments Corporation
MS 53-02
(512) 794-5678

Glossary

Prefix	Meaning	Value
n-	nano-	10^{-9}
µ-	micro-	10^{-6}
m-	milli-	10^{-3}

A

Assignable Cause — A cause that can be detected and identified as contributing to process variation.

Attribute Data — Data from counting process results such as number of units non-conforming per inspected sample, or number of defects per inspected sample containing *n* units. As opposed to measured data, attribute data has a discrete number of possible values. p, np, u and c control charts are used to track attribute data.

Attributes Chart — A control chart that tracks whether a process is in control by tracking attribute data—data counted from the process. p, np, u, c charts are attributes charts.

B

Blemish — See Defect.

C

c Chart — Control chart that uses number of defects per sample to monitor the stability of the process. Each sample contains *n* units, and contains c defects per sample (zero or more defects per unit). The value of *n* must be constant from sample to sample.

center line — A line on a control chart representing average long-term value of the statistic plotted on the control chart.

control chart	A charting method for determining the stability of a process; that is, whether or not a process is in a state of statistical control. Shewart control charts monitor process stability by plotting sample statistics against a center line and control limits.
control limits	Upper and lower limits on a control chart represent the amount of variation about a center line that can be attributed to chance causes for a given process characteristic. Control chart points that fall outside the control limits signal that the process is not in control and that some action should be taken. Control limits may be calculated from process data or from standard values.
Cp process capability index	A measure in sigma units of the process capability that is a ratio of the spread between the specification limits over the *m* sigma spread of the process variation, where *m* is normally 6.
Cpk centered capability ratio	A measure in sigma units of the process capability with respect to how well the process is centered relative to the specification limits, also known as distance to nearest specification.

D

Defect	A measured characteristic of a specific unit of product or service that prevents the unit from meeting a specification requirement or is otherwise undesirable. In a unit, one or more defects are possible, and many different types of defects are possible. Also known as a non-conformity, if the characteristic prevents the unit from meeting a specification requirement.

F

fraction non-conforming	The number of units in a sample not conforming to specification divided by the total number of units in the sample.

H

Histogram	A graphical summary of data in which the individual values are sorted by the range of values into which they fall (also known as bins), and in which the number of individuals falling within a each bin is counted. The data is plotted by showing the number of values (frequency) in each bin.

I

in-control process	A stable process whose variation is due only to chance causes; that is, no assignable causes of variation are present.
individual observation	A single measurement of a process characteristic.

M

matrix	Two-dimensional array.
measurement data	Data that is a result of observations or measurements of a characteristic that has a continuous range. As opposed to attribute data, measurement data is not discrete; that is, any discreteness in measurement data is due to the resolution of the measuring device, not the characteristic.
moving Average chart	Control chart that uses the average of *n* successive individual observations from a process to track the stability of the process mean. This type of control chart is typically used when the sample size is one.
moving Range Chart	Control chart that uses the range between *n* successive individual observations from a process to track the stability of the process variation. This type of control chart is typically used when sample size is one.

N

natural process limits	The limits which contain a stated fraction of the individual observations in a population. For a normally distributed population, the stated fraction is typically the process mean +/- 3.0 sigma.
non-conforming unit	A unit of product or service that does not meet a specification requirement.
normally distributed	If a process is normally distributed, expected values for individuals within the process population fall on a normal or bell-shaped curve.
np Chart	Control chart that uses the number of non-conforming units in a sample to monitor the stability of the process. The sample contains *n* units, and zero or more units may be non-conforming. The value *n* be constant from sample to sample.

O

observation	A measurement of a process characteristic.

P

p chart	Control chart that uses the fraction of non-conforming units in a sample to monitor the stability of the process. The sample contains *n* units, and zero or more units may be non-conforming. The value *n* may vary from sample to sample.
Pareto chart	A chart of a set of counted or totalized characteristics in which the characteristics are ranked in order of their frequency of occurrence. In SPC, Pareto charts are normally used to evaluate relative contribution of assignable causes and prioritize corrective action.
plot	A graphical representation of an array of data shown either on a graph or a chart.
process capability	A measure of the ability of a process to produce product within specification, assuming the process is stable. Specifically the variability that can be expected for a given process characteristic with respect to the specification limits for the characteristic.

R

R (Range) chart	Control chart that uses the sample range R to track the stability of the variation in the process. Sample range is max sample value minus min sample value. Sample size must be two or more.
run chart	A chart of the individual observations in a set of samples plotted in time order of occurrence.
run rules	Rules applied to a consecutive set of points on a control chart, that are used to detect changes in the process such as out-of-control conditions, or process shift.

S

s (std dev) chart	Control chart that uses the sample standard deviation s to determine the stability of variation in the process. Sample size must be two or more.

sample	A set of measurements (observations) or units (from which counted data is taken) used as a basis for evaluating the process.
sample (or subgroup) size	For measurement data, this is the number of observations (or individual measurements) making up the sample. For attribute data, this is the number of units n inspected for the counted characteristic.
specification limits	Limits that define the range within which a product or characteristic conforms to specification or user requirements (also known as tolerance limits).
standard values	Known standard values for process range, sample standard deviation, mean or sigma from which control limits can be calculated.
subgroup	A set of measurements (observations) taken from a larger set.

T

tier chart	A chart in which the individual observations in each sample are plotted vertically. It is a useful means of visualizing the variation or spread in each sample.

U

u chart	Control chart that uses the average number of defects per sample to monitor the stability of the process. The sample contains *n* units, and contains c defects per sample (zero or more defects per unit). The value *n* may vary from sample to sample.

V

variables chart	A pair of control charts that track whether a process is in control by tracking mean and variability of samples of measured data. Variables charts include X-bar and s, X-bar and R, x and mR charts, and mX-bar and mR charts.

X

x (individual) chart	Control chart that uses the individual observations from a process to track the stability of the process mean. This type of control chart is typically used when sample size is one.
X-bar chart	Control chart that uses the sample mean, X-bar, to track the stability of the process mean. Sample size must be two or more.