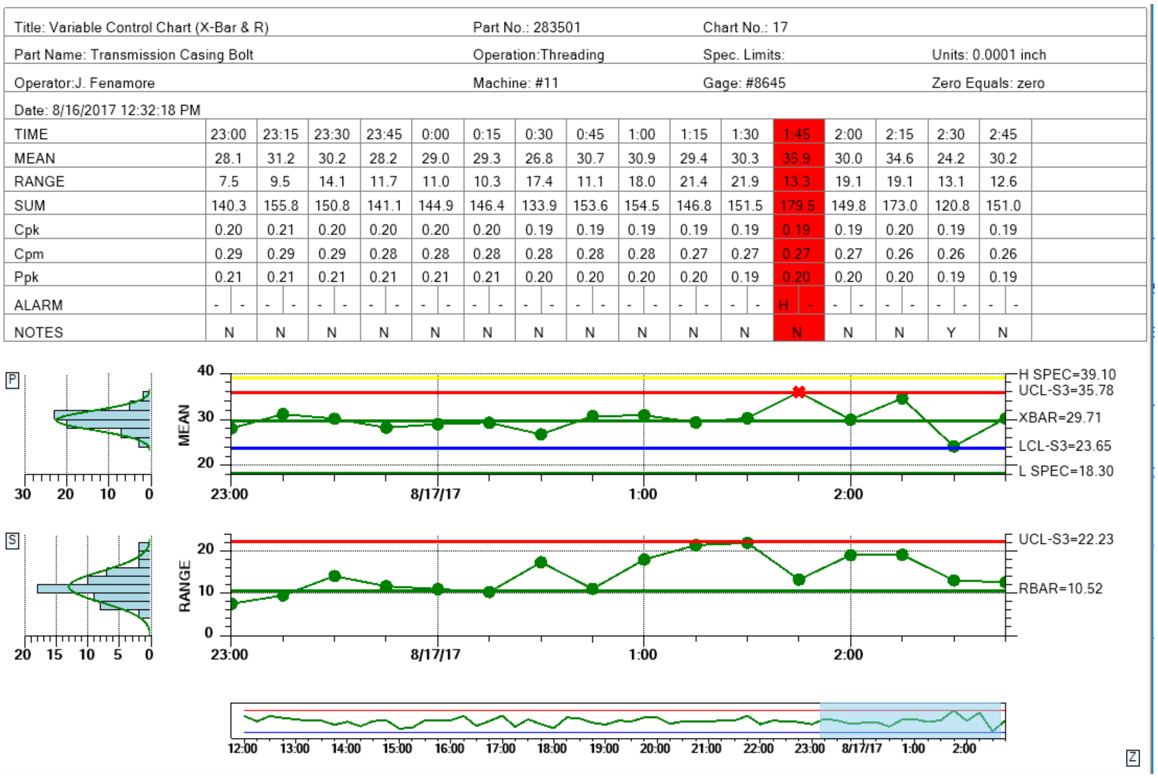
QCSPCChart SPC Control Chart Tools for .Net



Title: Variable Control Chart (X-Bar & R) Part No.: 283501 Chart No.: 17

Part Name: Transmission Casing Bolt Operation:Threading Spec. Limits: Units: 0.0001 inch

Operator:J. Fenamore Machine: #11 Gage: #8645 Zero Equals: zero

Date: 8/16/2017 12:32:18 PM

TIME	23:00	23:15	23:30	23:45	0:00	0:15	0:30	0:45	1:00	1:15	1:30	1:45	2:00	2:15	2:30	2:45
MEAN	28.1	31.2	30.2	28.2	29.0	29.3	26.8	30.7	30.9	29.4	30.3	35.9	30.0	34.6	24.2	30.2
RANGE	7.5	9.5	14.1	11.7	11.0	10.3	17.4	11.1	18.0	21.4	21.9	13.3	19.1	19.1	13.1	12.6
SUM	140.3	155.8	150.8	141.1	144.9	146.4	133.9	153.6	154.5	146.8	151.5	179.5	149.8	173.0	120.8	151.0
Cpk	0.20	0.21	0.20	0.20	0.20	0.20	0.19	0.19	0.19	0.19	0.19	0.19	0.19	0.20	0.19	0.19
Cpm	0.29	0.29	0.29	0.28	0.28	0.28	0.28	0.28	0.28	0.27	0.27	0.27	0.27	0.26	0.26	0.26
Ppk	0.21	0.21	0.21	0.21	0.21	0.21	0.20	0.20	0.20	0.20	0.19	0.20	0.20	0.20	0.19	0.19
ALARM	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	H -	- -	- -	- -	- -
NOTES	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Y	N

Contact Information

Company Web Site: http://www.quinn-curtis.com

Electronic mail

General Information: info@quinn-curtis.com
Sales: sales@quinn-curtis.com

Technical Support Forum

http://www.quinn-curtis.com/ForumFrame.htm

Revision Date 3/14/2023 Rev.3.1

Quinn-Curtis, Inc. Tools for *.Net* END-USER LICENSE AGREEMENT

IMPORTANT-READ CAREFULLY: This Software End-User License Agreement ("EULA") is a legal agreement between you (either an individual or a single entity) and Quinn-Curtis, Inc. for the Quinn-Curtis, Inc. SOFTWARE identified above, which includes all Quinn-Curtis, Inc. ***.Net*** software (on any media) and related documentation (on any media). By installing, copying, or otherwise using the SOFTWARE, you agree to be bound by the terms of this EULA. If you do not agree to the terms of this EULA, do not install or use the SOFTWARE. If the SOFTWARE was mailed to you, return the media envelope, UNOPENED, along with the rest of the package to the location where you obtained it within 30 days from purchase.

1. The SOFTWARE is licensed, not sold.

2. GRANT OF LICENSE.

(A) Developer License. After you have purchased the license for SOFTWARE, and have received the file containing the licensed copy, you are licensed to copy the SOFTWARE only into the memory of the number of computers corresponding to the number of licenses purchased. The primary user of the computer on which each licensed copy of the SOFTWARE is installed may make a second copy for his or her exclusive use on a portable computer. Under no other circumstances may the SOFTWARE be operated at the same time on more than the number of computers for which you have paid a separate license fee. You may not duplicate the SOFTWARE in whole or in part, except that you may make one copy of the SOFTWARE for backup or archival purposes. You may terminate this license at any time by destroying the original and all copies of the SOFTWARE in whatever form.

(B) 30-Day Trial License. You may download and use the SOFTWARE without charge on an evaluation basis for thirty (30) days from the day that you DOWNLOAD the trial version of the SOFTWARE. The termination date of the trial SOFTWARE is embedded in the downloaded SOFTWARE and cannot be changed. You must pay the license fee for a Developer License of the SOFTWARE to continue to use the SOFTWARE after the thirty (30) days. If you continue to use the SOFTWARE after the thirty (30) days without paying the license fee you will be using the SOFTWARE on an unlicensed basis.

Redistribution of 30-Day Trial Copy. Bear in mind that the 30-Day Trial version of the SOFTWARE becomes invalid 30-days after downloaded from our web site, or one of our sponsor's web sites. If you wish to redistribute the 30-day trial version of the SOFTWARE you should arrange to have it redistributed directly from our web site If you are using SOFTWARE on an evaluation basis you may make copies of the evaluation SOFTWARE as you wish; give exact copies of the original evaluation SOFTWARE to anyone; and distribute the evaluation SOFTWARE in its unmodified form via electronic means (Internet, BBS's, Shareware distribution libraries, CD-ROMs, etc.). You may not charge any fee for the copy or use of the evaluation SOFTWARE itself. You must not represent in any way that you are selling the SOFTWARE itself. You must distribute a copy of this EULA with any copy of the SOFTWARE and anyone to whom you distribute the SOFTWARE is subject to this EULA.

(C) Redistributable License. The standard Developer License permits the programmer to deploy and/or distribute applications that use the Quinn-Curtis SOFTWARE, royalty free. We cannot allow developers to use this SOFTWARE to create a graphics toolkit (a library or any type of graphics component that will be used in combination with a program development environment) for resale to other developers.

If you utilize the SOFTWARE in an application program, or in a web site deployment, should we ask, you must supply Quinn-Curtis, Inc. with the name of the application program and/or the URL where the SOFTWARE is installed and being used.

3. RESTRICTIONS. You may not reverse engineer, de-compile, or disassemble the SOFTWARE, except and only to the extent that such activity is expressly permitted by applicable law notwithstanding this

limitation. You may not rent, lease, or lend the SOFTWARE. You may not use the SOFTWARE to perform any illegal purpose.

4. SUPPORT SERVICES. Quinn-Curtis, Inc. may provide you with support services related to the SOFTWARE. Use of Support Services is governed by the Quinn-Curtis, Inc. polices and programs described in the user manual, in online documentation, and/or other Quinn-Curtis, Inc.-provided materials, as they may be modified from time to time. Any supplemental SOFTWARE code provided to you as part of the Support Services shall be considered part of the SOFTWARE and subject to the terms and conditions of this EULA. With respect to technical information you provide to Quinn-Curtis, Inc. as part of the Support Services, Quinn-Curtis, Inc. may use such information for its business purposes, including for product support and development. Quinn-Curtis, Inc. will not utilize such technical information in a form that personally identifies you.

5. TERMINATION. Without prejudice to any other rights, Quinn-Curtis, Inc. may terminate this EULA if you fail to comply with the terms and conditions of this EULA. In such event, you must destroy all copies of the SOFTWARE.

6. COPYRIGHT. The SOFTWARE is protected by United States copyright law and international treaty provisions. You acknowledge that no title to the intellectual property in the SOFTWARE is transferred to you. You further acknowledge that title and full ownership rights to the SOFTWARE will remain the exclusive property of Quinn-Curtis, Inc. and you will not acquire any rights to the SOFTWARE except as expressly set forth in this license. You agree that any copies of the SOFTWARE will contain the same proprietary notices which appear on and in the SOFTWARE.

7. EXPORT RESTRICTIONS. You agree that you will not export or re-export the SOFTWARE to any country, person, entity, or end user subject to U.S.A. export restrictions. Restricted countries currently include, but are not necessarily limited to Cuba, Iran, Iraq, Libya, North Korea, Sudan, and Syria. You warrant and represent that neither the U.S.A. Bureau of Export Administration nor any other federal agency has suspended, revoked or denied your export privileges.

8. NO WARRANTIES. Quinn-Curtis, Inc. expressly disclaims any warranty for the SOFTWARE. THE SOFTWARE AND ANY RELATED DOCUMENTATION IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OR MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NONINFRINGEMENT. THE ENTIRE RISK ARISING OUT OF USE OR PERFORMANCE OF THE SOFTWARE REMAINS WITH YOU.

9. LIMITATION OF LIABILITY. IN NO EVENT SHALL QUINN-CURTIS, INC. OR ITS SUPPLIERS BE LIABLE TO YOU FOR ANY CONSEQUENTIAL, SPECIAL, INCIDENTAL, OR INDIRECT DAMAGES OF ANY KIND ARISING OUT OF THE DELIVERY, PERFORMANCE, OR USE OF THE SUCH DAMAGES. IN ANY EVENT, QUINN-CURTIS'S LIABILITY FOR ANY CLAIM, WHETHER IN CONTRACT, TORT, OR ANY OTHER THEORY OF LIABILITY WILL NOT EXCEED THE GREATER OF U.S. $1.00 OR LICENSE FEE PAID BY YOU.

10. U.S. GOVERNMENT RESTRICTED RIGHTS. The SOFTWARE is provided with RESTRICTED RIGHTS. Use, duplication, or disclosure by the Government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of The Rights in Technical Data and Computer SOFTWARE clause of DFARS 252.227-7013 or subparagraphs (c)(i) and (2) of the Commercial Computer SOFTWARE- Restricted Rights at 48 CFR 52.227-19, as applicable. Manufacturer is: Quinn-Curtis, Inc., 18 Hearthstone Dr., Medfield MA 02052 USA.

11. MISCELLANEOUS. If you acquired the SOFTWARE in the United States, this EULA is governed by the laws of the state of Massachusetts. If you acquired the SOFTWARE outside of the United States, then local laws may apply.

Should you have any questions concerning this EULA, or if you desire to contact Quinn-Curtis, Inc. for any reason, please contact Quinn-Curtis, Inc. by mail at: Quinn-Curtis, Inc., 18 Hearthstone Dr., Medfield MA 02052 USA, or by telephone at: (508)359-6639, or by electronic mail at: support@Quinn-Curtis.com.

Table of Contents

1. Introduction

What's New in Rev. 3.1.

This version adds no new SPC Chart features to the software. Instead updates the software to use the Microsoft Windows .Net 6 Framework. The .Net 6 version of the Framework is the current version which has long term support from Microsoft. There is a .Net 7 version of the Framework, but that version does not promise long term support from Microsoft. There is a .Net 8 version, but that is in early stages of testing.

Unlike earlier .Net version upgrades, going from .Net 4 to .Net 6 is a major upgrade. You are not able to just change the supported version of .Net in the projects properties. If you try, .Net 6 will not be listed. Instead, you have to create a new project from scratch, and the Visual Studio project wizard will include .Net 6 support in the new project by default. Then you will have to add back in all of your source files, correcting any anatomies that pop-up. Once you do that, you will end up with a project that uses the .Net 6 Framework.

We did not have any success using the so-called Upgrade Assistant for converting existing projects to .Net 6 projects, so we are not going to spend any time on that. Instead, assume that you will always need to recreate your project from scratch. There are only a couple of changes you need to make in a ,Net 6 project in order to use the .Net 6 version of our software. First, the DLL names have changed (they now end in “net6”):

qcchart2dnet.dll → qcchart2dnet6.dll
qcspcchartnet.dll → qcspcchartnet6.dll

Second, the namespace for the libraries have changed (they now end in a “net6”):

com.quinncurtis.chart2dnet → com.quinncurtis.chart2dnet6.
com.quinncurtis.spcchartnet → com.quinncurtis.spcchartnet6.

So when you look to add the DLLs as a project reference, look for the qcchart2dnet6.dll and qcspcchartnet6.dll files. And when you reference the namespaces at the top of any files that references the QCChart2D and QCSPCChart classes, use com.quinncurtis.chart2dnet6 and com.quinncurtis.spcchartnet6. In this case the include section of your code might look like:

```
using System;
using System.Windows;
using System.Windows.Input;
using System.Windows.Media;
using com.quinncurtis.chart2dnet6;
using com.quinncurtis.spcchartnet6;
```

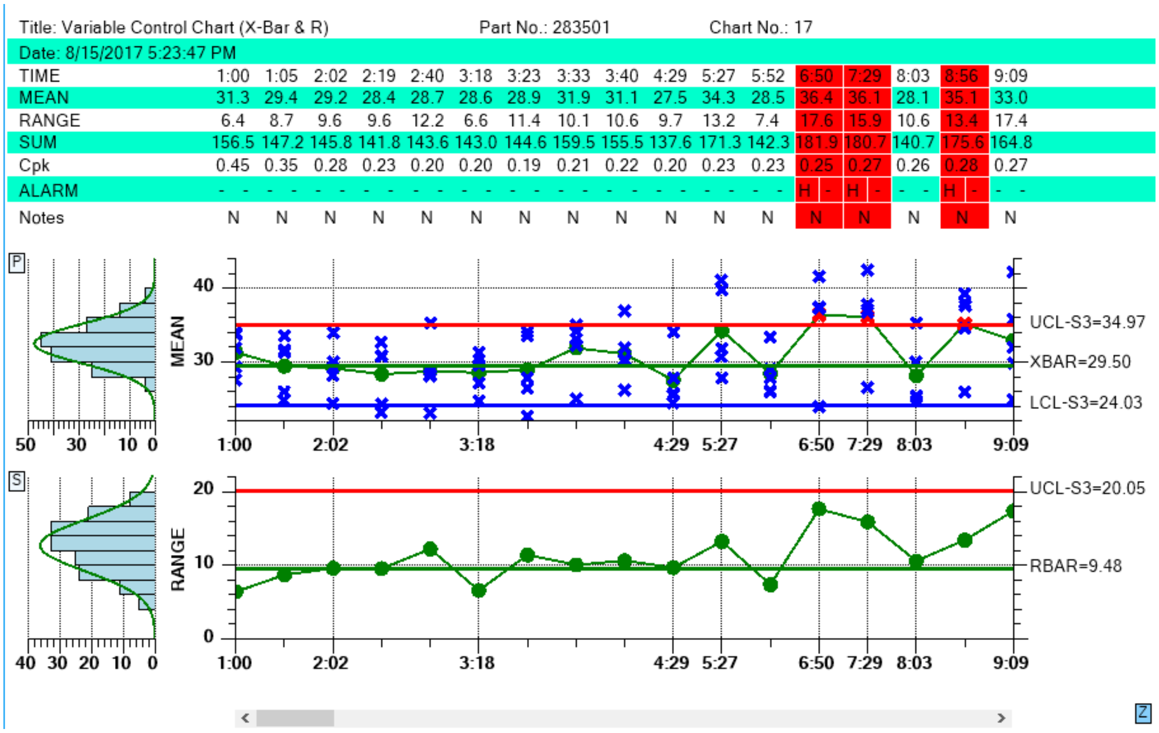
We have eliminated the ASP.Net examples. Microsoft no longer supports Asp.Net webforms in .Net 6. If you want to add SPC charts to a web page, use the JavaScript/TypeScript version of this software. That will enable you to add fully interactive charts to any sort of web framework that supports standard HTML and JavaScript.

Finally, we have removed the Visual Basic programming example projects from the software. You will find references to Visual Basic in the manual, but we aren't going to recreate the Visual Basic example projects, because it is more trouble than it is worth. Even Microsoft has declared Visual Basic a dead language and they will not be evolving it to match new features added to C#.

What's New in Rev. 3.0.

- **1. Charts have been converted to always use Event coordinates**
- **2. Zooming as an option for the scrollbar (UI option)**
- **3. Collapsible Items (UI option)**
- **4. Enhanced Annotations (Programming option)**
- **5. Enhanced Out of Limit Symbol (Programming option)**
- **6. Variable Spec and Control Limits (Programming option)**
- **7. Marking a sample internal as bad (Programming option)**
- **8. Reset N of M counters for control moves (Programming option)**
- **9. Remove Negative LCL control limits for Variable Control Secondary charts, or Attribute Control Primary charts (Programming option)**
- **10. N of M testing when the most recent point entering the test is within limits (Programming option)**
- **11. Sample Interval Updates with an invalid number of samples (Programming option)**
- **12. Alarm Forcing (Programming Option)**
- **13. A Levey-Jennings Variable Control Chart (Programming Option)**
- **14. Batch update with auto-calculated control limits (Programming Option).**

Charts have been converted to always use Event coordinates – Originally, the time-based SPC charts used a Time/Date x-scale in the coordinate system, and the batch-based SPC charts used a linear x-scale in the coordinate system. Both of these have been replaced with a coordinate system which uses an Event-based x-scale. In order to maintain backward compatibility, we also keep the old SPCTime... and SPCBatch... control chart classes, but derive them from the new Event-based SPC chart classes. You may see a few differences between the old and new time-based axis implementations. In the new version, no matter what the time stamp is on a SPCTime... SPC chart, adjacent points will always be equally spaced. So if your sample interval is irregular, or you even skip days or weeks in your sampling, the resulting chart will still display equally spaced adjacent sample records. Also, in the new version tick marks are placed on sample intervals. In the old version, the time axis tick marks were independent of the sample interval time stamps. In the new version, this can cause time axis labels to display the irregular time stamps of the sample interval time stamps. In the old version the tick mark time axis labels would be even values, because they did not use the sample interval time stamp values. You can read more about Event coordinates in the QCChart2D manual which is found in the same folder as QCSPCChart manual.



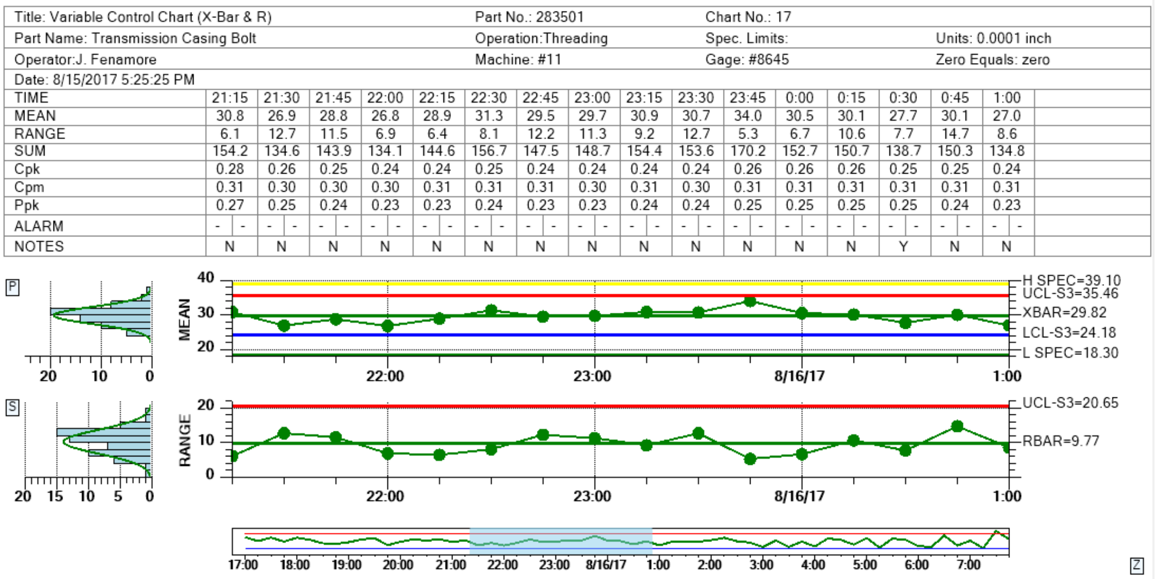
Title: Variable Control Chart (X-Bar & R) Part No.: 283501 Chart No.: 17

Date: 8/15/2017 5:23:47 PM

TIME	1:00	1:05	2:02	2:19	2:40	3:18	3:23	3:33	3:40	4:29	5:27	5:52	6:50	7:29	8:03	8:56	9:09
MEAN	31.3	29.4	29.2	28.4	28.7	28.6	28.9	31.9	31.1	27.5	34.3	28.5	36.4	36.1	28.1	35.1	33.0
RANGE	6.4	8.7	9.6	9.6	12.2	6.6	11.4	10.1	10.6	9.7	13.2	7.4	17.6	15.9	10.6	13.4	17.4
SUM	156.5	147.2	145.8	141.8	143.6	143.0	144.6	159.5	155.5	137.6	171.3	142.3	181.9	180.7	140.7	175.6	164.8
Cpk	0.45	0.35	0.28	0.23	0.20	0.20	0.19	0.21	0.22	0.20	0.23	0.23	0.25	0.27	0.26	0.28	0.27
ALARM	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	H -	H -	- -	H -	- -
Notes	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N

Event-based SPC charts will work with uneven sample intervals.

Zooming as an option for the scrollbar (UI option) –The horizontal scrollbar can be replaced (toggled on/off actually) using the UI, with a zoom window, by clicking on the z-button in the lower right corner. The user will be able to scroll the charts by dragging the transparent blue zoom window left or right, or expand the x-axis scale by dragging the left or right edge of the zoom window.



Title: Variable Control Chart (X-Bar & R) Part No.: 283501 Chart No.: 17

Part Name: Transmission Casing Bolt Operation:Threading Spec. Limits: Units: 0.0001 inch

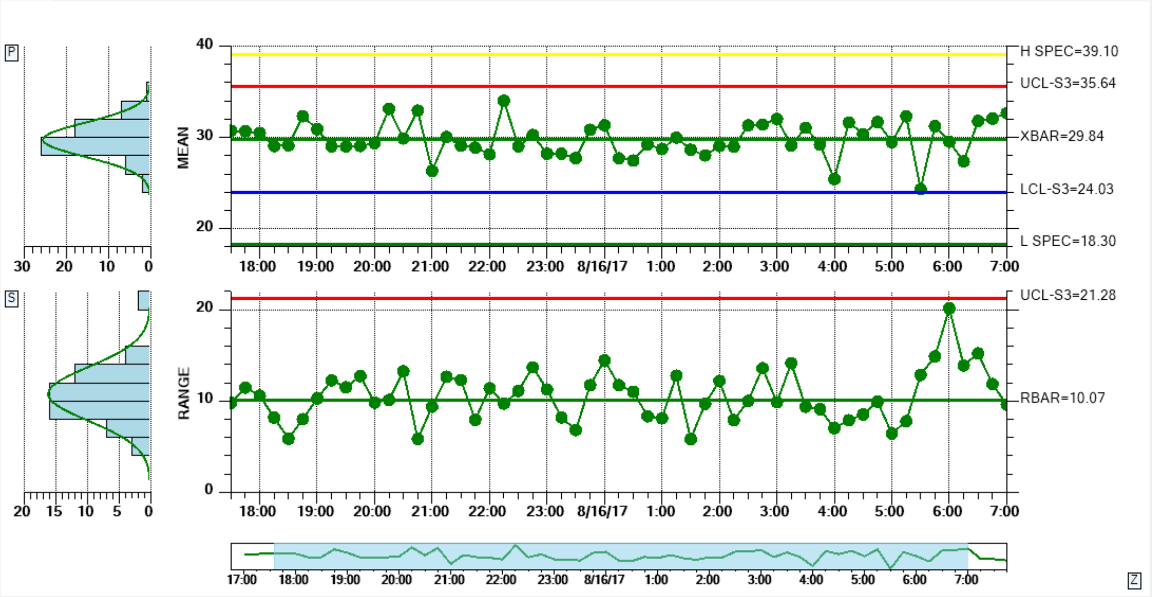
Operator:J. Fenamore Machine: #11 Gage: #8645 Zero Equals: zero

Date: 8/15/2017 5:25:25 PM

TIME	21:15	21:30	21:45	22:00	22:15	22:30	22:45	23:00	23:15	23:30	23:45	0:00	0:15	0:30	0:45	1:00
MEAN	30.8	26.9	28.8	26.8	28.9	31.3	29.5	29.7	30.9	30.7	34.0	30.5	30.1	27.7	30.1	27.0
RANGE	6.1	12.7	11.5	6.9	6.4	8.1	12.2	11.3	9.2	12.7	5.3	6.7	10.6	7.7	14.7	8.6
SUM	154.2	134.6	143.9	134.1	144.6	156.7	147.5	148.7	154.4	153.6	170.2	152.7	150.7	138.7	150.3	134.8
Cpk	0.28	0.26	0.25	0.24	0.24	0.25	0.24	0.24	0.24	0.24	0.26	0.26	0.26	0.25	0.25	0.24
Cpm	0.31	0.30	0.30	0.30	0.31	0.31	0.31	0.30	0.31	0.30	0.31	0.31	0.31	0.31	0.31	0.31
Ppk	0.27	0.25	0.24	0.23	0.23	0.24	0.23	0.23	0.24	0.24	0.25	0.25	0.25	0.25	0.24	0.23
ALARM	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -
NOTES	N	N	N	N	N	N	N	N	N	N	N	N	N	Y	N	N

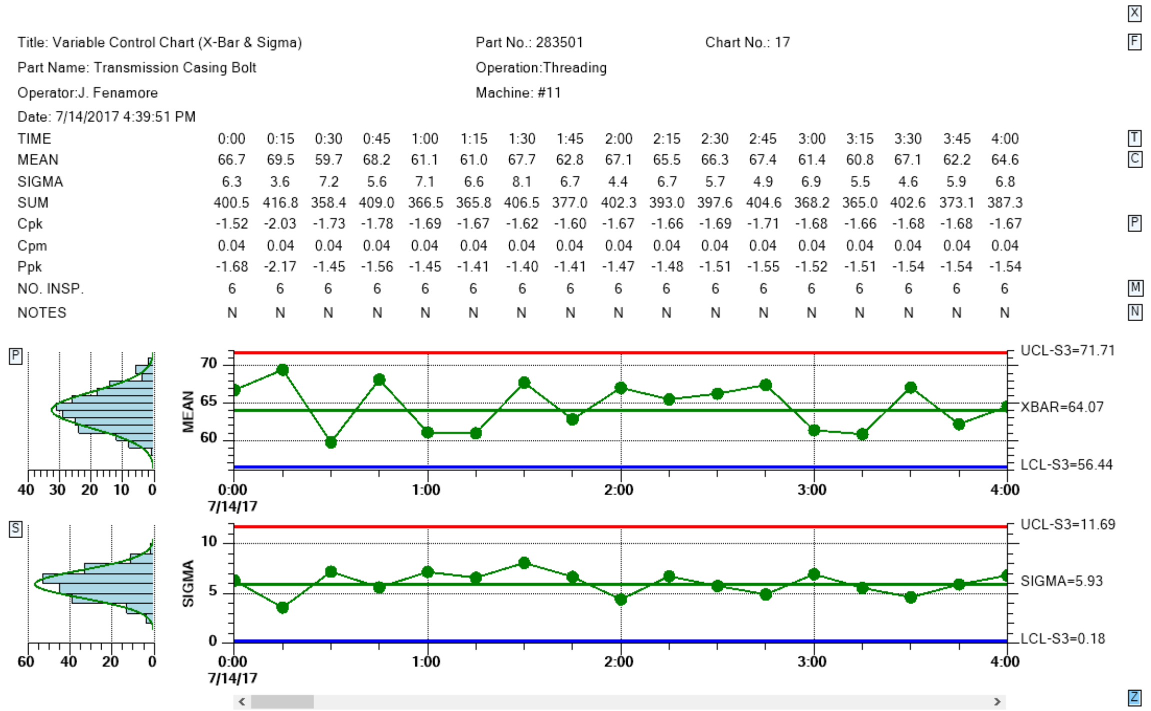
Click on the Z button in the lower right corner and a zoom window will replace the scrollbar. Use the mouse to scroll the charts by dragging the transparent blue zoom window left or right, or expand the x-axis scale by dragging the left or right edge of the zoom window.

If the number of points in the display exceeds the initial setup value for the number of data points, the table is temporarily removed to prevent overlap of the table columns. If the number of data points is reduced to <= the initial number of points, the table will reappear.



If you use the zoom window to display a lot of points, the table at the top will disappear to prevent the table columns from overlapping.

Collapsible Items (UI option) – Buttons on the top and right of the table permit the user to selectively collapse/restore rows of the table, freeing up more space charts. Buttons to the left and bottom of the Primary and Secondary charts also permit the user to collapse/restore either of those charts, allowing the remaining chart to display in all of the remaining space.



Title: Variable Control Chart (X-Bar & Sigma)   Part No.: 283501   Chart No.: 17

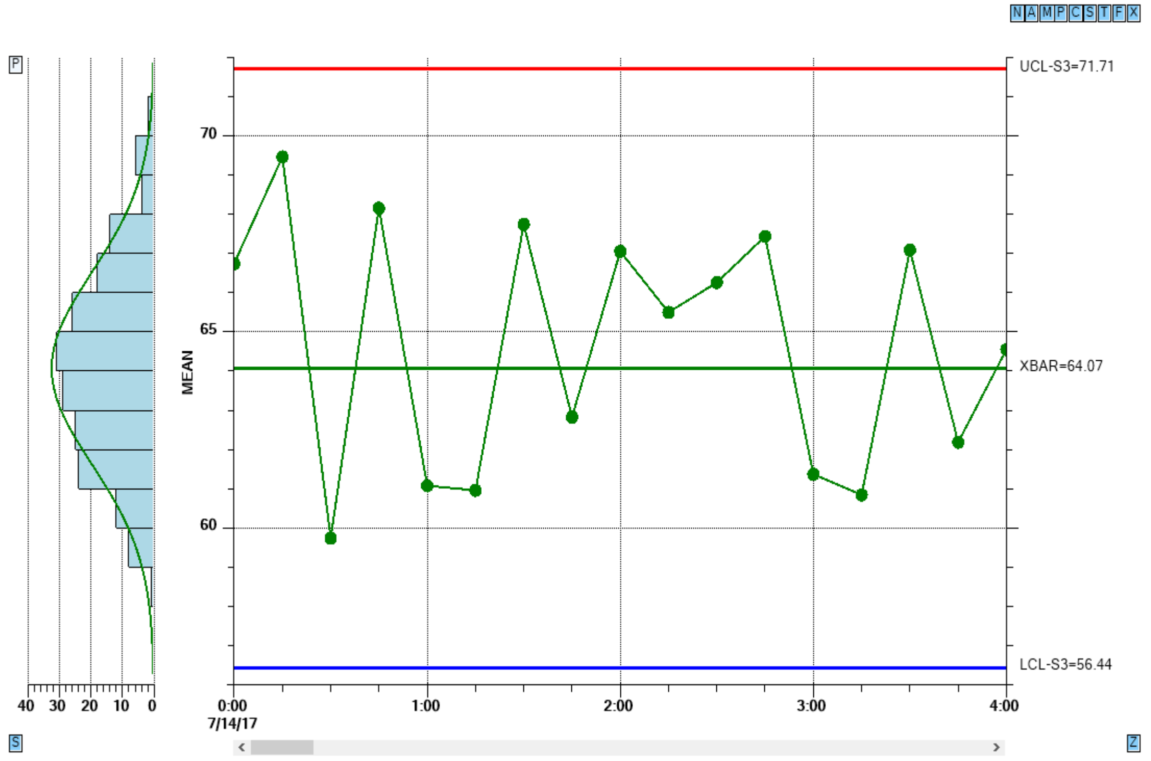
Part Name: Transmission Casing Bolt   Operation:Threading

Operator:J. Fenamore   Machine: #11

Date: 7/14/2017 4:39:51 PM

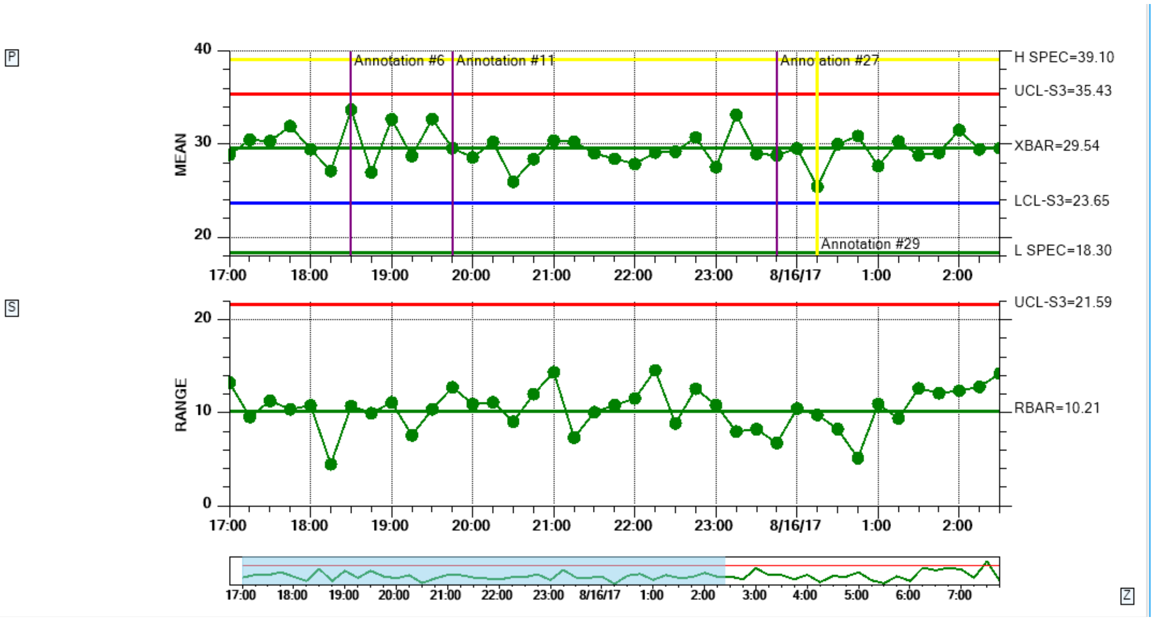
TIME	0:00	0:15	0:30	0:45	1:00	1:15	1:30	1:45	2:00	2:15	2:30	2:45	3:00	3:15	3:30	3:45	4:00
MEAN	66.7	69.5	59.7	68.2	61.1	61.0	67.7	62.8	67.1	65.5	66.3	67.4	61.4	60.8	67.1	62.2	64.6
SIGMA	6.3	3.6	7.2	5.6	7.1	6.6	8.1	6.7	4.4	6.7	5.7	4.9	6.9	5.5	4.6	5.9	6.8
SUM	400.5	416.8	358.4	409.0	366.5	365.8	406.5	377.0	402.3	393.0	397.6	404.6	368.2	365.0	402.6	373.1	387.3
Cpk	-1.52	-2.03	-1.73	-1.78	-1.69	-1.67	-1.62	-1.60	-1.67	-1.66	-1.69	-1.71	-1.68	-1.66	-1.68	-1.68	-1.67
Cpm	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04
Ppk	-1.68	-2.17	-1.45	-1.56	-1.45	-1.41	-1.40	-1.41	-1.47	-1.48	-1.51	-1.55	-1.52	-1.51	-1.54	-1.54	-1.54
NO. INSP.	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
NOTES	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N

Buttons on the top, right and left of the display permit the user to selectively collapse portions of the chart.



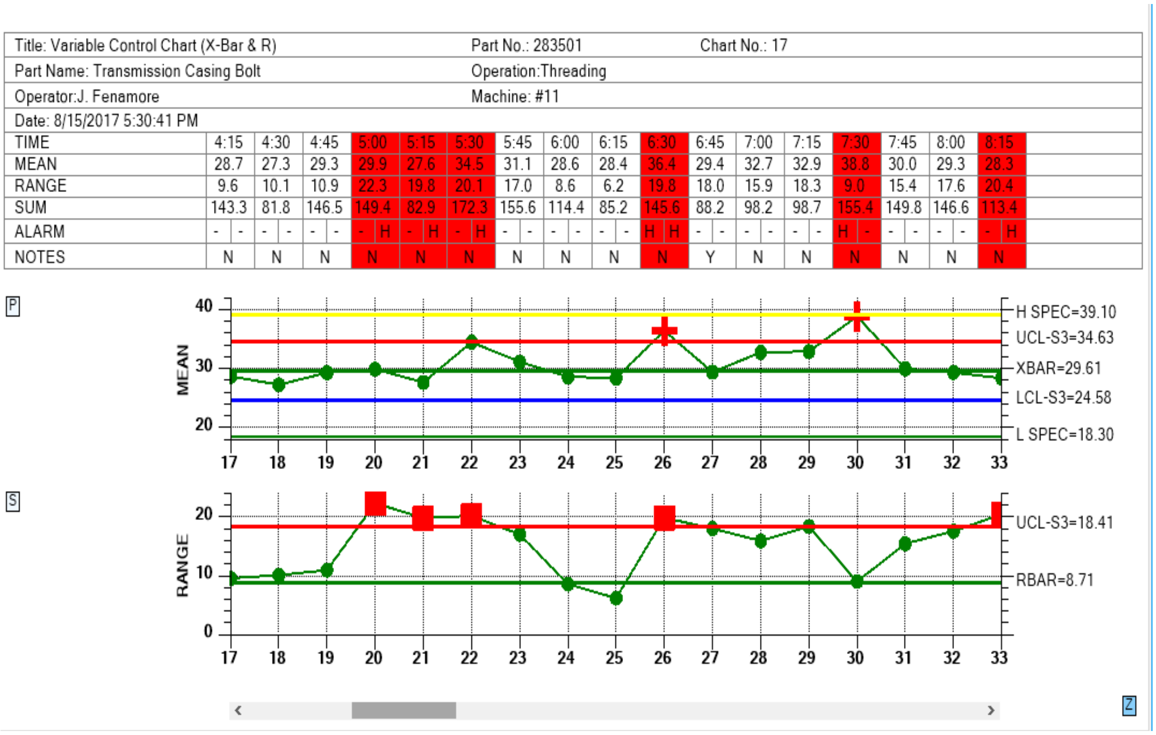
In the example above, all of the chart options except for the Primary chart, have been toggled off using the buttons.

Enhanced Annotations (Programming option) – The chart annotations have been enhanced with a vertical line with accompanying text using programmer specified justification.



The enhanced chart annotations include a vertical line to mark the data point, and many justification options (top, middle, bottom, left, right and center).

Enhanced Out of Limit Symbol (Programming option) – One of the SPC chart options is to mark a data point which is outside of control limits by changing the color of the symbol. It is now possible to also also change the size and symbol type for out of limit symbols.



Title: Variable Control Chart (X-Bar & R) Part No.: 283501 Chart No.: 17

Part Name: Transmission Casing Bolt Operation:Threading

Operator:J. Fenamore Machine: #11

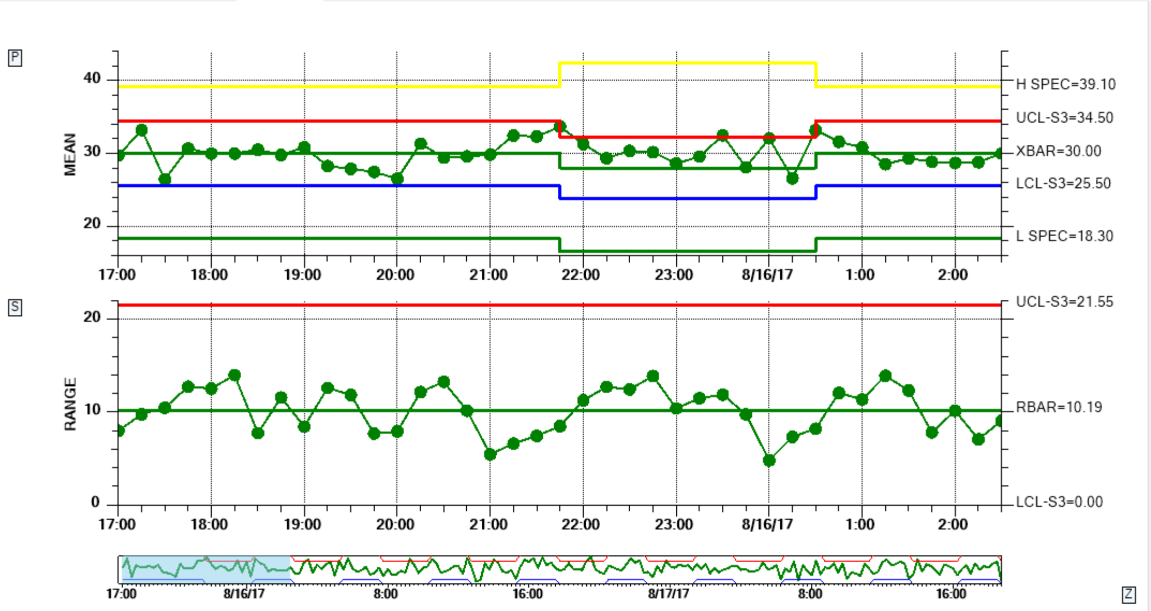
Date: 8/15/2017 5:30:41 PM

TIME	4:15	4:30	4:45	5:00	5:15	5:30	5:45	6:00	6:15	6:30	6:45	7:00	7:15	7:30	7:45	8:00	8:15
MEAN	28.7	27.3	29.3	29.9	27.6	34.5	31.1	28.6	28.4	36.4	29.4	32.7	32.9	38.8	30.0	29.3	28.3
RANGE	9.6	10.1	10.9	22.3	19.8	20.1	17.0	8.6	6.2	19.8	18.0	15.9	18.3	9.0	15.4	17.6	20.4
SUM	143.3	81.8	146.5	149.4	82.9	172.3	155.6	114.4	85.2	145.6	88.2	98.2	98.7	155.4	149.8	146.6	113.4
ALARM	- -	- -	- -	- H	- H	- H	- -	- -	- -	H H	- -	- -	- -	H -	- -	- -	- H
NOTES	N	N	N	N	N	N	N	N	N	N	Y	N	N	N	N	N	N

A data point which is outside of control limits can be automatically marked using color, symbol type, and color.

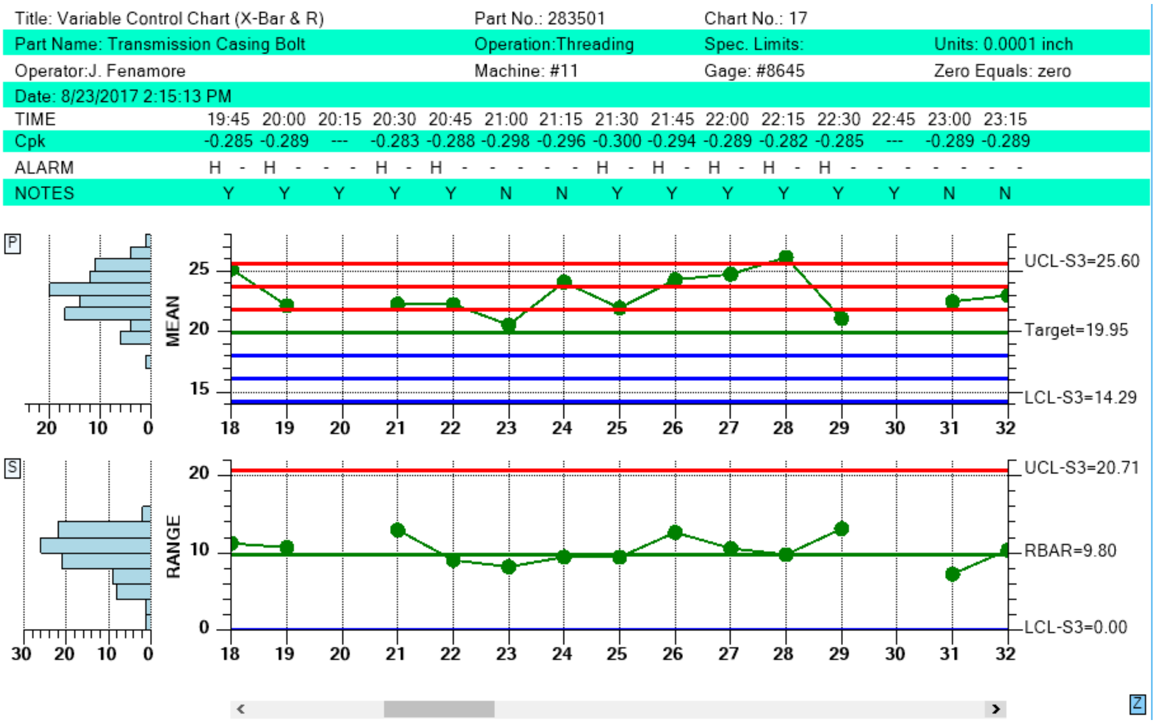
In the example above, the out of control symbol for the Primary chart is an extra large plus sign, while for the the Secondary chart it is an extra large square, both contrasting with the default circle symbol.

Variable Spec and Control Limits (Programming option) – In addition to variable control limits, charts can now have variable specification limits.



Control limits and specification limits can be adjusted on a sample interval by sample interval basis.

Marking a sample internal as bad (Programming option) – A sample interval can be marked as bad. This will disable plotting of the sample interval in the Primary and Secondary charts. It will also prevent the sample interval values from being evaluated in control limit testing, and in auto- calculations for control limits and y-axis range.



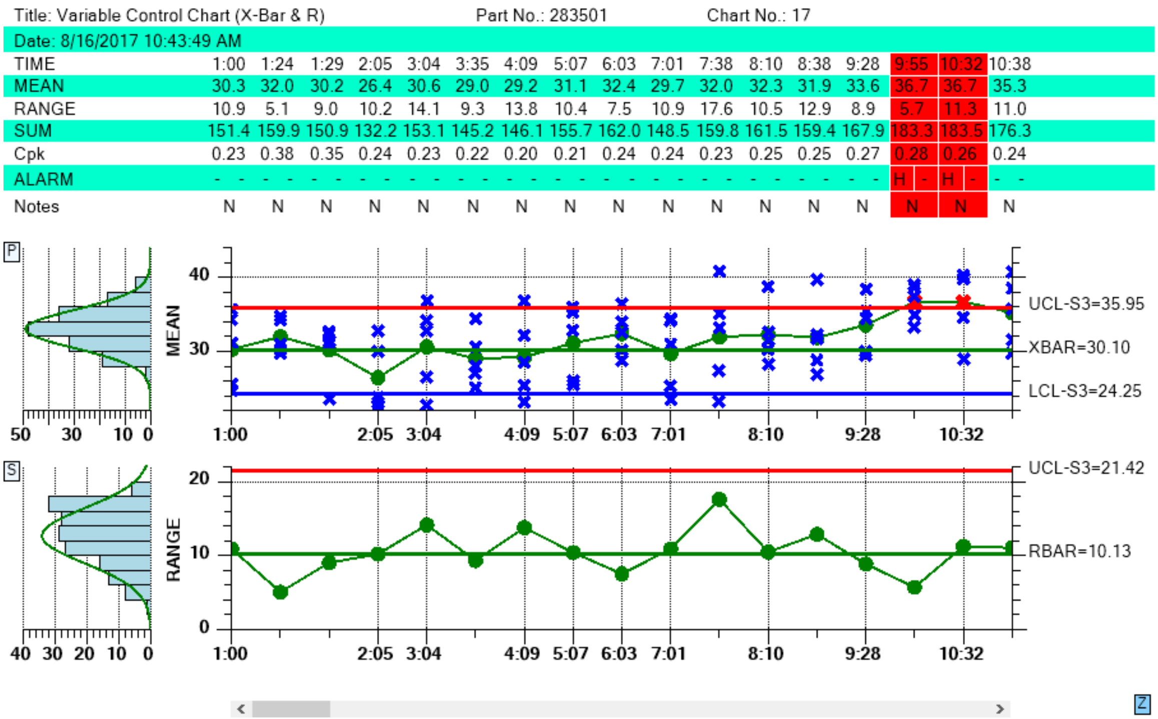
Title: Variable Control Chart (X-Bar & R)	Part No.: 283501	Chart No.: 17	
Part Name: Transmission Casing Bolt	Operation:Threading	Spec. Limits:	Units: 0.0001 inch
Operator:J. Fenamore	Machine: #11	Gage: #8645	Zero Equals: zero
Date: 8/23/2017 2:15:13 PM			

TIME	19:45	20:00	20:15	20:30	20:45	21:00	21:15	21:30	21:45	22:00	22:15	22:30	22:45	23:00	23:15
Cpk	-0.285	-0.289	---	-0.283	-0.288	-0.298	-0.296	-0.300	-0.294	-0.289	-0.282	-0.285	---	-0.289	-0.289
ALARM	H -	H -	- -	H -	H -	- -	- -	H -	H -	H -	H -	H -	- -	- -	- -
NOTES	Y	Y	Y	Y	Y	N	N	Y	Y	Y	Y	Y	Y	N	N

A hole (sample intervals 20 and 30 in the picture above) will appear in the main plot of the Primary and Secondary charts when a sample interval is marked bad.

Reset N of M counters for control moves (Programming option) – There are cases when a user wants to keeps the same chart going, adding new sample intervals with new data, after the issue which caused the process to go out of control has been remedied. But many of the control limit tests found in the named control rules (WECO, Nelson, etc.) use N of M tests, where N out of M sample intervals must violate a specific rule. For example, a WECO rule says that if 4 out of 5 samples are outside of 1-sigma, it is an alarm condition. But if the process is now in control, the user may want to reset all N or M counts back to 0, exactly as if the plotting of the chart had started at the first sample interval. So we have added a reset mechanism for the N or M rules to a zero count for all rules.

Remove Negative LCL control limits for Variable Control Secondary charts, or Attribute Control Primary charts (Programming option) - We added a routine which when called will disable (both from display and alarm checking) any chart LCL control limit if the limit value is <= a specified value, 0.0 in most cases.



Title: Variable Control Chart (X-Bar & R) Part No.: 283501 Chart No.: 17

Date: 8/16/2017 10:43:49 AM

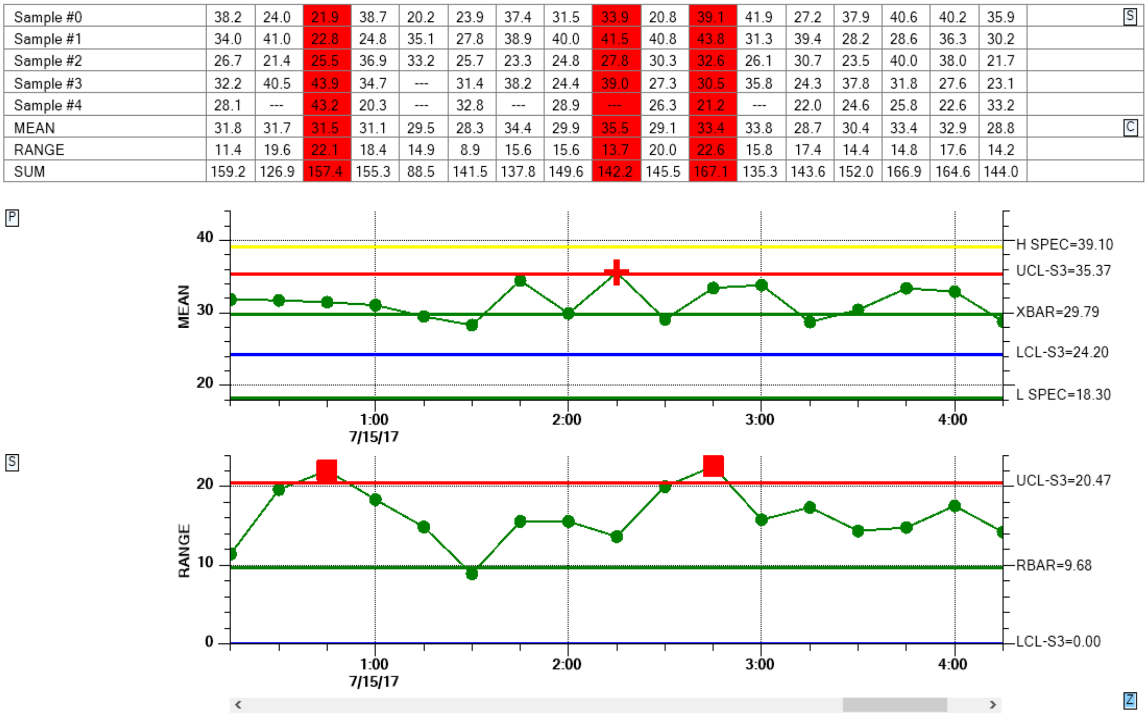
TIME	1:00	1:24	1:29	2:05	3:04	3:35	4:09	5:07	6:03	7:01	7:38	8:10	8:38	9:28	9:55	10:32	10:38
MEAN	30.3	32.0	30.2	26.4	30.6	29.0	29.2	31.1	32.4	29.7	32.0	32.3	31.9	33.6	36.7	36.7	35.3
RANGE	10.9	5.1	9.0	10.2	14.1	9.3	13.8	10.4	7.5	10.9	17.6	10.5	12.9	8.9	5.7	11.3	11.0
SUM	151.4	159.9	150.9	132.2	153.1	145.2	146.1	155.7	162.0	148.5	159.8	161.5	159.4	167.9	183.3	183.5	176.3
Cpk	0.23	0.38	0.35	0.24	0.23	0.22	0.20	0.21	0.24	0.24	0.23	0.25	0.25	0.27	0.28	0.26	0.24
ALARM	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	H -	H -	- -
Notes	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N

Note that the lower control limit for the Secondary (Range) chart is not present. It was calculated be 0.0, and was removed by the software.

N of M testing when the most recent point entering the test is within limits (Programming option) - Our default mode takes a strict approach to N of M testing.

Regardless of the value of the most recent point to enter the calculation (even if it is within the test limits), if N of M values are outside of limits we consider the sample interval to be in alarm. But some customers challenged this interpretation and presented us with published examples which show that if the most recent point is within limits and the previous N points out of limits, the sample interval should NOT be considered in alarm. We researched the issue and found no agreement. It is implemented in the published literature both ways. So a simple global flag has been added you can use to choose one evaluation method or the other, with the default being our original method.

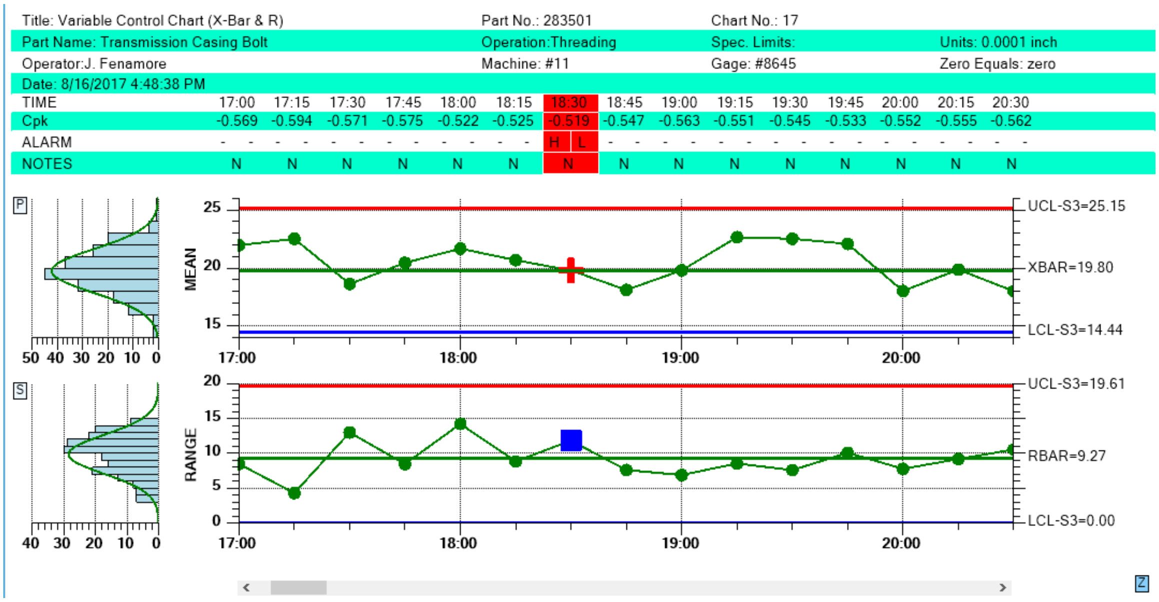
Sample Interval Updates with an invalid number of samples   (Programming option) - When you have specified one of the fixed sample size charts, a more lenient evaluation method for sample interval updates has been implemented. You can now enter less than or greater than the specified number of samples for an interval update without the software complaining. It will process the samples as if the proper number of values was entered, without introducing zero values into the sample interval. So if the specified number of samples per sample interval is 5, and 3 are entered, it will still work. It will calculate the mean and range of the sample interval using only 3 values. However, it will not adjust the control limits to take into account the incorrect number of samples, unlike what is done in the dedicated variable sample size (_VSS) charts. Also, you can not enter in a single value for any of the control charts which require more than one value. Because then you cannot calculate the range, or sigma value of the sample interval, something the software uses internally.



Sample #0	38.2	24.0	21.9	38.7	20.2	23.9	37.4	31.5	33.9	20.8	39.1	41.9	27.2	37.9	40.6	40.2	35.9	S
Sample #1	34.0	41.0	22.8	24.8	35.1	27.8	38.9	40.0	41.5	40.8	43.8	31.3	39.4	28.2	28.6	36.3	30.2	
Sample #2	26.7	21.4	25.5	36.9	33.2	25.7	23.3	24.8	27.8	30.3	32.6	26.1	30.7	23.5	40.0	38.0	21.7	
Sample #3	32.2	40.5	43.9	34.7	---	31.4	38.2	24.4	39.0	27.3	30.5	35.8	24.3	37.8	31.8	27.6	23.1	
Sample #4	28.1	---	43.2	20.3	---	32.8	---	28.9	---	26.3	21.2	---	22.0	24.6	25.8	22.6	33.2	
MEAN	31.8	31.7	31.5	31.1	29.5	28.3	34.4	29.9	35.5	29.1	33.4	33.8	28.7	30.4	33.4	32.9	28.8	C
RANGE	11.4	19.6	22.1	18.4	14.9	8.9	15.6	15.6	13.7	20.0	22.6	15.8	17.4	14.4	14.8	17.6	14.2	
SUM	159.2	126.9	157.4	155.3	88.5	141.5	137.8	149.6	142.2	145.5	167.1	135.3	143.6	152.0	166.9	164.6	144.0	

Note that the sample data in the table has many columns where the number of samples is less than the specified 5 samples per sample interval.

Alarm Forcing – High and low control lines can be set to explicit values, or you can have the software auto-calculated the values based on historical data. As new sample intervals are added to the chart, the new values are compared to existing control limits and the alarm conditions noted. But, you can override the alarm limit display, so that a normal condition shows as an alarm, or an alarm condition shows as normal.



Title: Variable Control Chart (X-Bar & R) Part No.: 283501 Chart No.: 17

Part Name: Transmission Casing Bolt Operation:Threading Spec. Limits: Units: 0.0001 inch

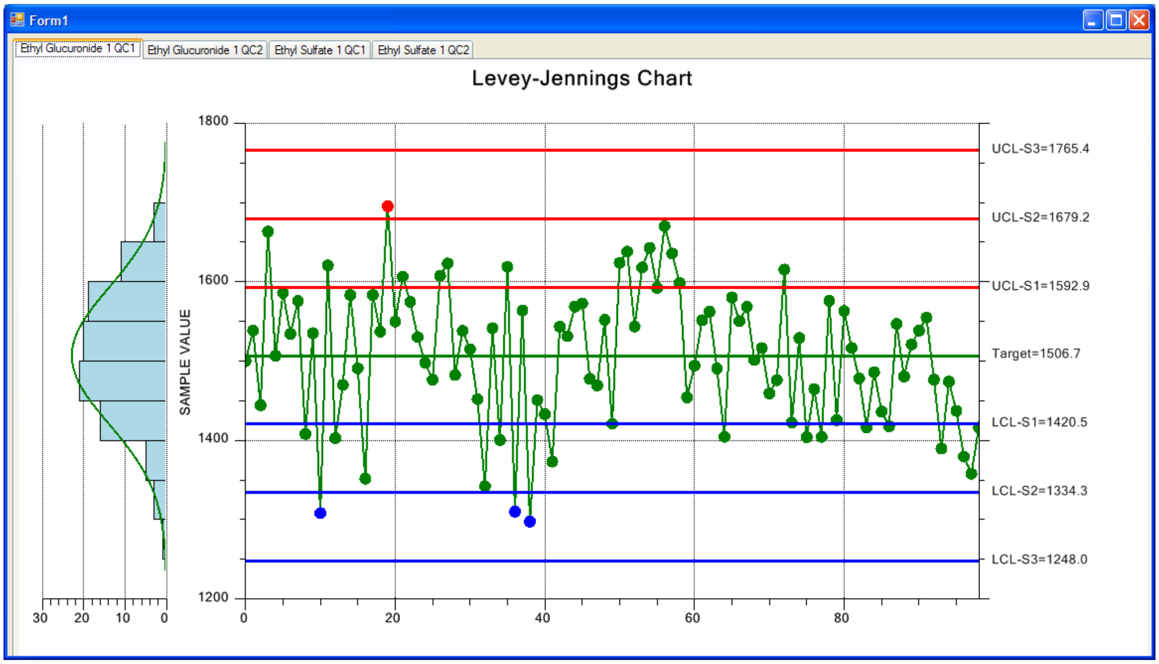
Operator:J. Fenamore Machine: #11 Gage: #8645 Zero Equals: zero

Date: 8/16/2017 4:48:38 PM

TIME	17:00	17:15	17:30	17:45	18:00	18:15	18:30	18:45	19:00	19:15	19:30	19:45	20:00	20:15	20:30
Cpk	-0.569	-0.594	-0.571	-0.575	-0.522	-0.525	-0.519	-0.547	-0.563	-0.551	-0.545	-0.533	-0.552	-0.555	-0.562
ALARM	- -	- -	- -	- -	- -	- -	H L	- -	- -	- -	- -	- -	- -	- -	- -
NOTES	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N

Sample intervals are forced to into an alarm state, regardless of the calculated value.

Levey-Jennings Chart - We added a new SPC chart type, the Levey-Jennings chart, using the Westgard rules.

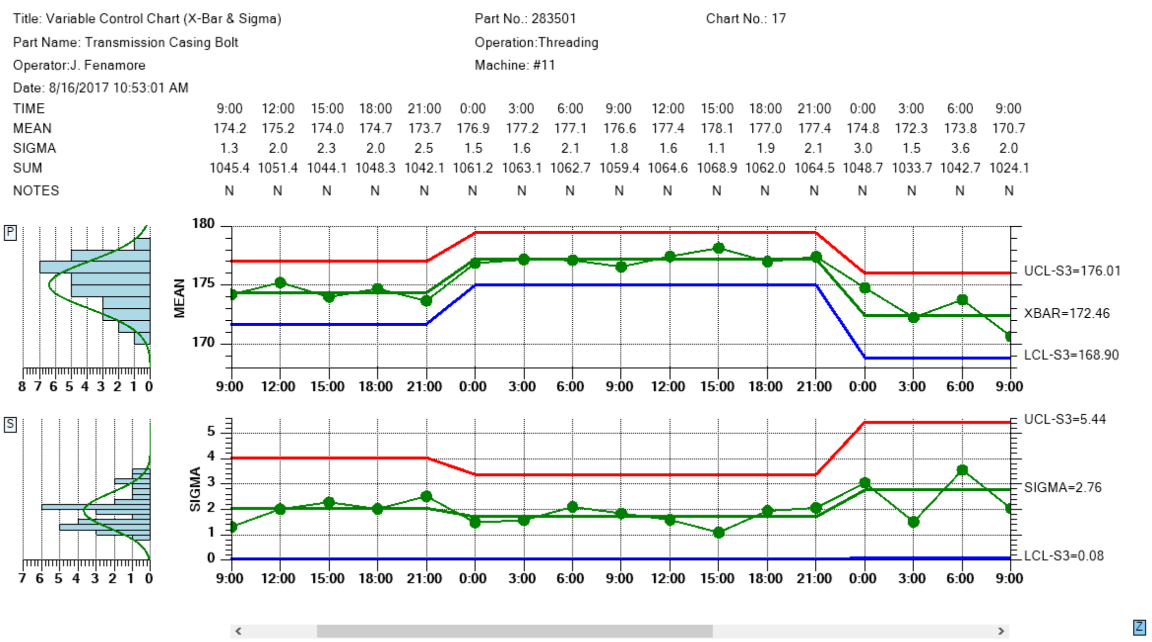


Typical Levey-Jennings Chart using Batch Sampling

The Levey-Jennings chart is used almost exclusively in laboratory settings. It uses a chart very similar to the Individual Range chart above, the major difference being that it only uses the Primary individual data point graph of the chart and does not include the Secondary range graph. Also, the Levey-Jennings chart uses the Westgard rules which utilizes tests involving 1-, 2- and 3- sigma control limits.

Batch update with auto-calculated control limits

We added a routine (CalculateControlLimitsFromBatch) in the main SPCChartBase class, which will calculate a set of control limits based on a batch of N sample intervals. So if your first batch has 20 sample intervals, you would call that routine and you would get back the control limits specific to that set of data. Then you would set the control limits to those values, then update the chart with your sample interval data. The data would be evaluated against the control limits you just set. Then, you can take another batch of N sample intervals, calculate the limits, set the limits, and update the graph again. The new process data, and control limits will be appended to the line plots of the existing chart. The new limits will only apply to the new data values. Repeat ad infinitum.



Title: Variable Control Chart (X-Bar & Sigma) Part No.: 283501 Chart No.: 17

Part Name: Transmission Casing Bolt Operation:Threading

Operator:J. Fenamore Machine: #11

Date: 8/16/2017 10:53:01 AM

TIME	9:00	12:00	15:00	18:00	21:00	0:00	3:00	6:00	9:00	12:00	15:00	18:00	21:00	0:00	3:00	6:00	9:00
MEAN	174.2	175.2	174.0	174.7	173.7	176.9	177.2	177.1	176.6	177.4	178.1	177.0	177.4	174.8	172.3	173.8	170.7
SIGMA	1.3	2.0	2.3	2.0	2.5	1.5	1.6	2.1	1.8	1.6	1.1	1.9	2.1	3.0	1.5	3.6	2.0
SUM	1045.4	1051.4	1044.1	1048.3	1042.1	1061.2	1063.1	1062.7	1059.4	1064.6	1068.9	1062.0	1064.5	1048.7	1033.7	1042.7	1024.1
NOTES	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N

You can auto-calculate control limits for batches of process values, and concatenate the resulting graphs together.

To make it even easier to use, we added flags to the CalculateControlLimitsFromBatch routine which can optionally set the calculated control limits, and update the chart with the supplied data, eliminating the need for the programmer to that externally. So the way this works is that each batch will display all at once, appended to the existing data. There is no incremental update of one sample interval at a time.

What's New in Rev. 2.3

All of the new features found in the 2.3 version of QCSPCChart were added to QCChart2D - primarily a new collection of event based charting classes. None of these routines have been used in the QCSPCChart software.

Event-Based Charting

A new set of classes have been added in support of new, event-based plotting system. In event-based plotting, the coordinate system is scaled to the number of event objects. Each event object represents an x-value, and one or more y-values. The x-value can be time based, or numeric based, while the y-values are numeric based. Since an event object can represent one or more y-values for a single x-value, it can be used as the source for simple plot types (simple line plot, simple bar plot, simple scatter plot, simple line marker plot) and group plot types (open-high-low-close plots, candlestick plots, group bars, stacked bars, etc.). Event objects can also store custom data tooltips, and x-axis strings. The most common use for event-based plotting will be for displaying time-based data which is discontinuous: financial markets data for example. In financial markets, the number trading hours in a day may change, and the actual trading days. Weekends, holidays, and unused portions of the day can be excluded from the plot scale, producing continuous plots of discontinuous data. The following classes have been added to the software in support of event-based charting.

- **ChartEvent** - A ChartEvent object stores the position value, the time stamp, y-values, and custom strings associated with the event.
- **EventArray** - A utility array class used to store ChartEvent objects
- **EventAutoScale** – An auto-scale class used by the EventCoordinates class.
- **EventAxis** - Displays an axis based on an EventCoordinates scale
- **EventAxisLabels** – Displays the string labels labeling the tick marks of an EventAxis
- **EventCoordinates** – Event coordinates define a coordinate system based on the the attached Event datasets
- **EventGroupDataset** – A group dataset which uses ChartEvent objects as the source of the data. It is used to feed data into the group plotting routines.
- **EventScale** – An event scale class used to convert between event coordinates and device coordinates.
- **EventSimpleDataset** - A simple dataset which uses ChartEvent objects as the source of the data. It is used to feed data into the simple plotting routines.

What's New in Rev. 2.2

- **Three new control charts** – Moving Average/Moving Range (MAMR), Moving Average/Moving Sigma (MAMS) and the Number Defects per Million (DPMO). See Chapter 6 (SPC Variable Control Charts) and Chapter 9 (*SPC Attribute Control Charts*).
- **Direct support for additional control rule sets** – Nelson, AIAG, Juran, Hughes, Gitlow, Westgard, and Duncan, in addition to the WE (Western Electric) and Supplemental Rules. See Chapter 8 (*Named and Custom Control Rule Sets*).
- **Create custom sets of control rules**, using any rule from any of the standard named control rule sets. See Chapter 8 (*Named and Custom Control Rule Sets*).
- **Define custom control rules:** based any of our predefined control rule templates See Chapter 8 (*Named and Custom Control Rule Sets*).
- **Regionalization** – All strings used in the software have been moved to a static class which can be initialized at runtime with country specific strings. See Chapter 11 (*Regionalization for non-USA English Markets*).

New Features found in the 2.1 version of QCSPCChart

Revision 2.1 adds the following new features:

- **Specification Limits** – Unique from control limits. See page 212.

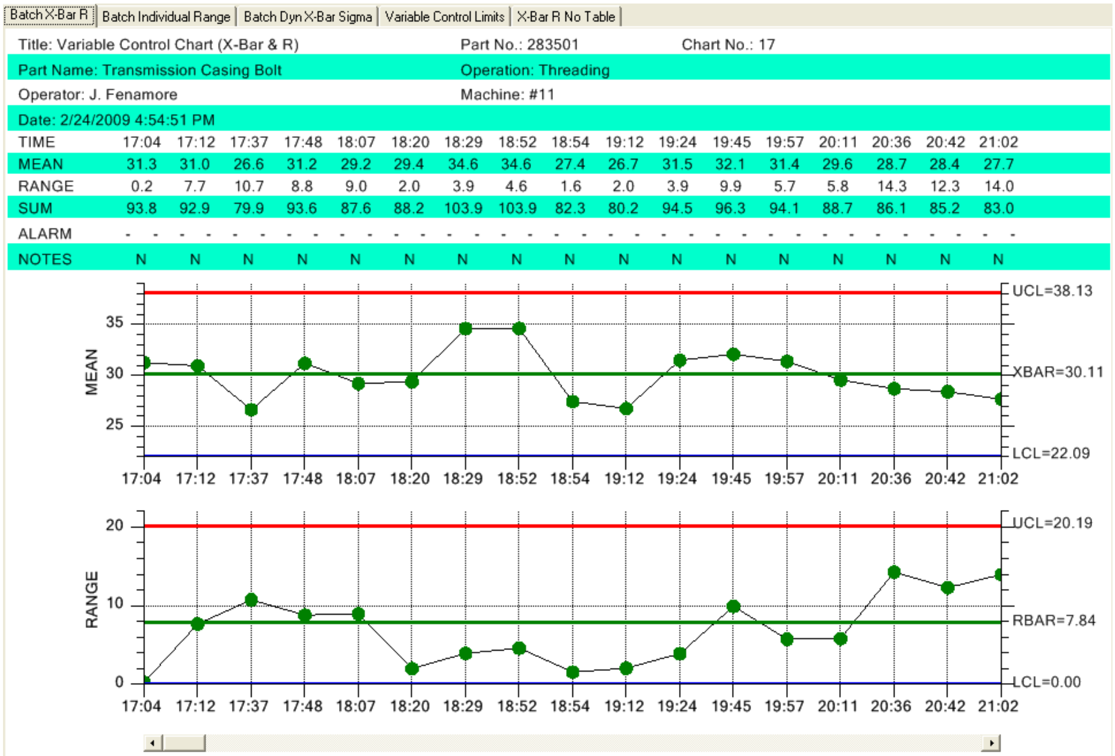
- **Western Electric Trending** (or Supplemental) Rules (Rules 5 – 8). See page 207.
- **Simpler way to set control limits for +-1, 2 and 3 sigma - Add3SigmaControlLimits** – See page 201.
- **Solid area fill between limit lines - ControlLimitLineFillMode** – See page 206.

New Features found in the 2.0 version of QCSPCChart

Revision 2.0 follows Revision 1.7. Most new features associated with revision 2.0 are part of the QCChart2D software, on top of which the QCSPCChart software is built. As far this software goes, only a few featues specific to Revision 2.0 of QCSPCChart have been added. These include:

- The batch control chart templates (SPCBatchVariableControlChart, SPCBatchAttributeControlChart) have new x-axis labeling modes. Label the x-axis tick marks using a batch number (the original and default mode), a time stamp, or a user-defined string.

- The x-axis labels can be rotated 360 degrees.



Title: Variable Control Chart (X-Bar & R)　Part No.: 283501　Chart No.: 17

Part Name: Transmission Casing Bolt　Operation: Threading

Operator: J. Fenamore　Machine: #11

Date: 2/24/2009 4:54:51 PM

TIME	17:04	17:12	17:37	17:48	18:07	18:20	18:29	18:52	18:54	19:12	19:24	19:45	19:57	20:11	20:36	20:42	21:02
MEAN	31.3	31.0	26.6	31.2	29.2	29.4	34.6	34.6	27.4	26.7	31.5	32.1	31.4	29.6	28.7	28.4	27.7
RANGE	0.2	7.7	10.7	8.8	9.0	2.0	3.9	4.6	1.6	2.0	3.9	9.9	5.7	5.8	14.3	12.3	14.0
SUM	93.8	92.9	79.9	93.6	87.6	88.2	103.9	103.9	82.3	80.2	94.5	96.3	94.1	88.7	86.1	85.2	83.0
ALARM	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -
NOTES	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N

The time stamp of batch control chart does not have to be does not have to have an equal time spacing between adjacent sample groups.

Revision 2.0 has added many new to QCChart2D. New features include:

- Five new plot types: **BoxWiskerPlot**, **FloatingStackedBarPlot**, **RingChart**, **SimpleVersaPlot** and **GroupVersaPlot**
- Elapsed time scaling to compliment the time/date scaling. Includes a set of classes specifically for elapsed time charts: **ElapsedTimeLabel**, **ElapsedTimeAutoScale**, **ElapsedTimeAxis**, **ElapsedTimeAxisLabels**, **ElapsedTimeCoordinates**, **ElapsedTimeScale**, **ElapsedTimeSimpleDataset** and **ElapsedTimeGroupDataset**.
- Vertical axis scaling for time/date and elapsed time
- A **DatasetViewer** class for the grid-like display of dataset information in a table.
- A **MagniView** class: a new way to zoom data
- A **CoordinateMove** class – used to pan the coordinate system, left, right, up, down.
- Zoom stack processing is now internal to the ChartZoom class

Refer to the QCChart2D manual for information specific to these new features.

Visual Studio 2005 Projects

All of the example program projects have been converted to the Visual Studio 2005 project format. The VS 2005 project format is the oldest project format we expect to support in years moving forward. While it may be possible to use the QCChart2D/QCSPCChart software with VS 2003, or VS 2002, it is not something we support any more. You should assume that as we continue to enhance the software, we will use features not supported under VS 2003/2002. We can still sell you a Revision 1.7 version of the software, with all of the original VS 2003/2002 projects; however it will not include the features described above.

One difference betwee of VS 2005 and VS 2003/2002 is the VS 2005 creation of *partial* classes when using the **Add User Control** wizard. The VS 2005 **Add User Control** wizard now creates two classes, UserControlName.cs and UserControlName.Designer.cs (or UserControlName.vb and UserControlName.Designer.vb), by default. The Designer specific code is now placed in the UserControlName.Designer.vb file. In VS 2003/2002, where a separate UserControlName.Designer.cs file is NOT created, the Designer code was placed in the main UserControlName.cs file, most of which was hidden using the

```
 #region Windows Form Designer generated code
```

compiler directive. Many of example programs still use this older style, with the single UserControlName file. The single file structure is forward compatible with the VS 2005 compiler. All of the example programs demonstrating new features in the software use the split UserControlName.cs/UserControlName.Designer.cs file structure. This split structure is NOT backward compaticle with VS 2003/2002.

New Features added to the 1.7 version of QCSPCChart

A large number of new features were added to QCSPCChart in the change from Revision 1.6 to 1.7. These are summarized below:

Eliminated the QCLicense License Fle

We have eliminated the QCLicense license file from the software and with it the need to purchase additional Redistributable Licenses. Once you purchase the software, you the developer, can create application programs that use this software and redistribute the programs and our libraries royalty free. As a development tool, i.e. using this software in conjunction with a compiler, the software is still governed by a single user license and cannot be shared by multiple individuals unless additional copies, or a site license, have been purchased.

New SPC Chart Types and Features

- Three new charts have been added to the software: **EWMA** (Exponentially Weighted Moving Average), **MA** (Moving Average), and **CuSum** (Cumulative Sum) charts.

- Three variants of existing charts have been adapted to support variable subgroup sample sizes: X-Bar Sigma, p-Chart (Fraction or Percent of Defective Parts) and u-Chart (Number of Defects per Unit).

- The FrequencyHistogramChart now includes optional limit lines, and an optional fit and overlay of a normal curve to the frequency data.

New Alarm Features

- There is a new status line in the table section that gives a direct indication of whether or not the corresponding process variable is in, or out of, alarm. The status line has a tooltip, so if you click on an alarm, a pop-up box will show you details.

- Optionally, the entire column associated with a sample interval can be color highlighted to indicate an alarm condition.

- Alarm details can be automatically logged to the notes log.

- The symbol used to plot a process variable point in the primary and secondary charts can be made to change color in the event of an alarm.

DesignerSerializationVisibility

We have long found the VS IDE habit of including long initialization lists for the **ChartView** and **SPCBaseChart** properties, when a **ChartView** derived **UserControl** is added to a Form, annoying. We found it was very easy to initialize properties in a **ChartView** or **SPCBaseChart** subclass, only to have them over-ridden in hidden InitializeComponent code. So we added the:

```
 DesignerSerializationVisibility(DesignerSerializationVisibility.Hidden)]
```

compiler flag in front of all **ChartView** and **SPCBaseChart** properties to force the VS IDE to ignore them.

Tutorials

Chapter 12 is a tutorial that describes how to get started with the **SPC Control Chart Tools for .Net** charting software. Chapter 13 is a tutorial that describes how to use the software to create charts for web pages.

Customer Support

Use our forums at http://www.quinn-curtis.com/ForumFrame.htm for customer support. Please, do not post questions on the forum unless you are familiar with this manual and have run the examples programs provided. We try to answer most questions by referring to the manual, or to existing example programs. We will always attempt to answer any question that you may post, but be prepared that we may ask you to create, and send to us, a simple example program. The program should reproduce the problem with no, or minimal interaction, from the user. You should strip out of any code not directly associated with reproducing the problem. You can either your own example or a modified version of one of our own examples.

SPC Control Chart Tools for *.Net* Background

In a competitive world environment, where there are many vendors selling products and services that *appear* to be the same, **quality**, both real and perceived, is often the critical factor determining which product wins in the marketplace. Products that have a reputation for higher quality command a premium, resulting in greater market share and profit margins for the manufacturer. Low quality products not only take a big margin hit at the time of sale, but also taint the manufacturer with a reputation that will hurt future sales, regardless of the quality of future products. Users have a short memory. A company's quality reputation is only as good as the quality of its most recent product.

The measurement, control and gradual improvement of quality is the goal of all quality systems, no matter what the name. Some of the more common systems are known as **SCC** (Statistical Quality Control) **Quality Engineering**, **Six-Sigma**, **TQM** (Total Quality Management), **TQC** (Total Quality Control), **TQA** (Total Quality Assurance) and **CWQC** (Company- Wide Quality Control). These systems work on the principle that management must integrate quality into the basic structure of the company, and not relegate it to a Quality Control group within the company. Historically, most of the innovations in quality systems took place in the 20th century, with pioneering work carried out by Frederick W. Taylor (Principles of Scientific Management), Henry Ford (Ford Motor), W. A. Shewhart (Bell Labs), W. E. Deming (Department of Agriculture, War department, Census Bureau), Dr. Joseph M. Juran (Bell Labs), and Dr. Armand V. Feigenbaum among others. Most quality control engineers are familiar with the story of how the statistical quality control pioneer, W. E. Deming, frustrated that US manufactures only gave lip service to quality, took the quality system he developed to Japan, where it was embraced with almost religious zeal. Japanese industry considers Deming a national hero, where his quality system played a major role in the postwar expansion of the Japanese economy. Twenty to thirty years after Japan embraced his methods, Deming found a new audience for his ideas at US companies that wanted to learn *Japanese* methods of quality control.

All quality systems use Statistical Process Control (**SPC**) to one degree or another. SPC is a family of statistical techniques used to track and adjust the manufacturing process in order to produce gradual improvements in quality. While it is based on sophisticated mathematical analysis involving sampling theory, probability distributions, and statistical inferences, SPC results can usually be summarized using simple charts that even management can understand. SPC charts can show how product quality varies with respect to critical factors that include things like batch number, time of day, work shift personal, production machine, and input materials. These charts have odd names like X-Bar R, Median Range, Individual Range, Fraction Number Non-Conforming, and NP. The charts plot some critical process variable that is a measurement of product quality and compares it to predetermined limits that signify whether or not the process is working properly.

Initially, quality control engineers create all SPC charts by hand. Data points were painstakingly gathered, massaged, summed, averaged and plotted by hand on graph paper. It is still done this way in many cases. Often times it is done by the same factory floor personal who control the process being measured, allowing them to “close the loop” as quickly as possible, correcting potential problems in the process before it goes out of control. Just as important, SPC charts tell the operator when to leave the process alone. Trying to micro-adjust a process, when the process is just exhibiting normal random fluctuations in quality, will often drive the process out of control faster than leaving it alone.

The modern tendency is to automate as much of the SPC chart creation process as possible. Electronic measuring devices can often measure quality in real-time, as items are coming off the line. Usually some form of sampling will be used, where one of every N items is measured. The sampled values form the raw the data used in the SPC chart making process. The values can be entered by hand into a SPC chart making program, or they can be entered directly from a file or database connection, removing the potential for transcription errors. The program displays the sampled data in a SPC chart and/table where the operator or quality engineer can make a judgment about whether or not the process is operating in or out of control.

Usually the SPC engineer tasked with automating an existing SPC charting application has to make a decision about the amount of programming he wants to do. Does he purchase an application package that implements standard SPC charts and then go about defining the charts using some sort of menu driven interface or wizard. This is probably the most expensive in terms of up front costs, and the least flexible, but the cheapest in development costs since a programmer does not have to get involved creating the displays. Another choice is to use a general purpose spreadsheet package with charting capability to record, calculate, and display the charts. This is probably a good choice if your charting needs are simple, and you are prepared to write complicated formulas as spreadsheet entries, and your data input is not automated. Another choice is writing the software from scratch, using a charting toolkit like our ***QCChart2D*** software as the base, and creating custom SPC charts using the primitives in the toolkit. This is cheaper up front, but may be expensive in terms of development costs. Often times the third option is the only one available because the end-user has some unique requirement that the pre-packaged software can't handle, hence everything needs to programmed from scratch.

Quinn-Curtis SPC (Statistical Process Control) Software

We have created a library of SPC routines that represents an intermediate solution. Our SPC software still requires an intermediate level programmer, but it does not require advanced knowledge of SPC or of charting. Built on top our ***QCChart2D,*** it implements templates and support classes for the following SPC charts and control limit calculations.

Variable Control Charts Templates

- Fixed sample size subgroup control charts
  - X-Bar R – (Mean and Range Chart)
  - X-Bar Sigma (Mean and Sigma Chart)
  - Median and Range (Median and Range Chart)
  - X-R (Individual Range Chart)
  - EWMA (Exponentially Weighted Moving Average Chart)
  - MA (Moving Average Chart)
  - MAMR (Moving Average / Moving Range Chart)
  - MAMS (Moving Average / Moving Sigma Chart)
  - CuSum (Tabular Cumulative Sum Chart)
  - Levey-Jennings with Westgard rules
- Variable sample size subgroup control charts
  - X-Bar Sigma (Mean and Sigma Chart)

Attribute Control Charts Templates

- Fixed sample size subgroup control charts
  - p Chart (Fraction or Percent of Defective Parts)
  - np Chart (Number of Defective Parts)
  - c-Chart (Number of Defects)
  - u-Chart (Number of Defects per Unit)

Number Defects per Million (DPMO)
Variable sample size subgroup control charts
p Chart (Fraction or Percent of Defective Parts)
u-Chart (Number of Defects per Unit)

Analysis Chart Templates
Frequency Histograms
Probability Charts
Pareto Charts
SPC Support Calculations
Array statistics (sum, mean, median, range, standard deviation, variance, sorting)
SPC Control Limit Calculations
High and low limit control calculations for X-Bar R, X-Bar Sigma, Median and Range, X-R, p, np, c and u charts
SPC Process Capability Calculations
Variable Control Charts include Cp, Cpl, Cpu, Cpk, Cpm, Pp, Ppl, Ppu, and Ppk process capability statistics
SPC Control Named Rule Sets
Western Electric (WECO) Runtime and Supplemental Rules
Nelson
AIAG
Juran
Hughes
Gitlow
Westgard
Duncan

The **SPC Control Chart Tools for *.Net*** is a family of templates that integrate the **QCChart2D** charting software with tables, data structures and specialized rendering routines used for the static and dynamic display of SPC charts. The SPC chart templates are pre-programmed classes that create, manage and display the graphs and tables corresponding to major SPC control chart types. Each template can be further customized using method and properties. The programmers can customize the plot objects created in the template, allowing tremendous flexibility in the look of the SPC charts.

Like the **QCChart2D** software, the **SPC Control Chart Tools for** .Net uses the graphics features found in the Microsoft .Net API. These include:

- Arbitrary line thickness and line styles for all lines.
- Printer and image output support
- Font support for a large number of fonts, using a variety of font styles, size and rotation attributes.
- Imaging support for a large number of image formats

SPC Control Chart Tools for *.Net* Dependencies

The **SPC Control Chart Tools for *.Net*** class library builds on the **QCChart2D** software package. It uses the classes found in that software and standard classes that ship with the Microsoft .Net API. No other software is required.

Directory Structure of QCSPCChart for *.Net*

The **SPC Control Chart Tools for** .Net class library uses the standard directory structure also used by the **QCChart2D** and **QCRTGraphics** software. It adds the **QCSPCChart** directory structure under the Quinn-Curtis\DotNet folder. For a list of the folders specific to **QCChart2D**, see the manual for **QCChart2D**, QCChart2DNetManual.pdf.

Drive:

Quinn-Curtis\ - Root directory

DotNet\ - Quinn-Curtis .Net based products directory

Docs\ - Quinn-Curtis .Net related documentation directory

Lib\ - Quinn-Curtis .Net related compiled libraries and components directory

QCChart2D\ - QCChart2D examples for C# and VB – This directory contains many example programs for C# and VB specific to the QCChart2D charting software, but not specific to the QCSPCChart software

QCSPCChart\ - QCSPCChart examples for C# and VB

Visual CSharp\ - C# specific directory

Examples net6\ - C# examples directory

FrequencyHistogram – a simple frequency histogram example using the **FrequencyHistogramChart** class

BatchAttributeControlCharts - a collection of batch attribute control charts, including n, np, c, and u charts using the **SPCBatchAttributeControlChart** class.

BatchVariableControlCharts - a collection of batch variable control charts, including X-Bar R, X-Bar Sigma, Median Range, and X-R charts using the **SPCBatchVariableControlChart** class.

TimeAttributeControlCharts - a collection of time attribute control charts, including n, np, c, and u charts using the **SPCTimeAttributeControlChart** class.

TimeVariableControlCharts BatchVariableControlCharts - a collection of time variable control charts, including X-Bar R, X-Bar Sigma, Median Range, and X-R charts using the **SPCTimeVariableControlChart** class.

MiscTimeBasedControlCharts - a collection of time variable control charts, including EWMA, MA and CuSum charts using the **SPCTimeVariableControlChart** class.

MiscBatchBasedControlCharts - a collection of batch variable control charts, including EWMA, MA and CuSum charts using the **SPCTimeVariableControlChart** class.

RulesRulesRules - a collection of control charts demonstrating the the use of named (WECO, Nelson, AIAG, Juran, Hughes, Duncan, Westgard, and Gitlow) and custom rule sets.

ProbabilityPlot - a probability chart using the **ProbabilityChart** class.

ParetoDiagram - a Pareto diagram chart using the **ParetoChart** class.

SPCApplication1 – A simple X-Bar R example program, using **SPCTimeVariableControlChart**, used in the tutorial.

LeveyJennings – A group of four Levey-Jennings charts using Westgard rules and a variety of data and formats.

WERulesVariableControlCharts - a collection of using the WE rules with **SPCTimeVariableControlChart** charts, including X-Bar R, X-Bar Sigma, Median Range, and X-R

VariableSampleSizeControlCharts - a collection of the variable control (X-Bar Sigma), and attribute control (p- and u-charts) that support variable sample subgroup sizes.

NewRev3Features – a collection of example which demonstrate the new features found in Rev. 3.0.

(*** Critical Note ***) Running the Example Programs

The example programs for **SPC Control Chart Tools for .Net** software are supplied in complete source. In order to save space, they have not been pre-compiled which means that many of the intermediate object files needed to view the main form are not present. This means that **ChartView** derived control will not be visible on the main Form if you attempt to view the main form before the project has been compiled. The default state for all of the example projects should be the Start Page. Before you do view any other file or form, do a build of the project. This will cause the intermediate files to be built. If you attempt to view the main Form before building the project, Visual Studio sometimes decides that the **ChartView** control placed on the main form does not exist and deletes it from the project.

There are two versions of the for **SPC Control Chart Tools for *.Net*** class library: the 30-day trial versions, and the developer version. Each version has different characteristics that are summarized below:

30-Day Trial Version

The trial version of **SPC Control Chart Tools for *.Net*** is downloaded in a file named Trial_QCSPCChartR20x. The 30-day trial version stops working 30 days after the initial download. The trial version includes a version message in the upper left corner of the graph window that cannot be removed.

Developer Version

The developer version of **SPC Control Chart Tools for *.Net*** is downloaded in a file with a name similar to NETSPCDEV1UR2x2x561x1.zip The developer version does not time out and you can use it to create application programs that you can distribute royalty free. You can download free updates for a period of 2-years. When you placed your order, you were e-mailed download link(s) that will download the software. Those download links will remain active for at least 2 years and should be used to download current versions of the software. After 2 years you may have to purchase an upgrade to continue to download current versions of the software

Chapter Summary

The remaining chapters of this book discuss the **SPC Control Chart Tools for *.Net*** package designed to run on any hardware that has a .Net runtime installed on it.

Chapter 2 presents a summary of the standard SPC control charts that can be created using the software.

Chapter 3 presents the overall class architecture of the **SPC Control Chart Tools for *.Net*** and summarizes all of the classes found in the software.

Chapter 4 covers how to use the new features found in Rev. 3.0 of the software. Previously, Chapter 4 was a brief summary of QCChart2D, which you can still read about in the QCChart2DNet manual.

Chapter 5 describes the classes that hold SPC control chart data and control limit alarms.

Chapter 6 describes how the **SPCTimeVariableControlChart** and **SPCBatchVariableControlChart** classes create common variable control charts: X-Bar R, Median and Range, X-Bar Sigma, X-R, EWMA, MA and CuSum charts**.**

Chapter 7 describes how the **SPCTimeAttributeControlChart** and **SPCBatchAttributeControlChart** classes create common attribute control charts: p-, np-, c- and u-charts.

Chapter 8 describes how to implement the named control rules (WECO, Nelson, AIAG, Juran, Hughes, Gitlow, Westgard, and Duncan) control rules. It also describes how to implement custom rules sets, and how to define your own rules using our standardized templates.

Chapter 9 describes how the **FrequencyHistogramChart**, **ParetoChart** and **ProbabilityChart** classes create ancillary SPC charts.

Chapter 10 describes how to print the SPC charts, and save them to image files.

Chapter 11 describes how to regionalize the software for non-USA English markets.

Chapter 12 is a tutorial that describes how to use **SPC Control Chart Tools for .Net** to create Windows applications using Visual Studio .Net and Visual C#.

2. Standard SPC Control Charts

There are many different types SPC control charts. Normally they fall into one of two major classifications: *Variable Control Charts*, and *Attribute Control Charts*. Within each classification, there are many sub variants. Often times the same SPC chart type has two or even three different names, depending on the software package and/or the industry the chart is used in. We have provided templates for the following SPC control charts:

Variable Control Charts

- Fixed sample size subgroup control charts
  - X-Bar R – (Mean and Range Chart)
  - X-Bar Sigma (Mean and Sigma Chart)
  - Median and Range (Median and Range Chart)
  - X-R (Individual Range Chart)
  - EWMA (Exponentially Weighted Moving Average Chart)
  - MA (Moving Average Chart)
  - MAMR (Moving Average / Moving Range Chart)
  - MAMS (Moving Average / Moving Sigma Chart)
  - CuSum (Tabular Cumulative Sum Chart)
  - Levey-Jennings with Westgard Rules
- Variable sample size subgroup control charts
  - X-Bar Sigma (Mean and Sigma Chart)

Attribute Control Charts

- Fixed sample size subgroup control charts
  - p Chart (Fraction or Percent of Defective Parts)
  - np Chart (Number of Defective Parts)
  - c-Chart (Number of Defects)
  - u-Chart (Number of Defects per Unit)
  - Number Defects per Million (DPMO)
- Variable sample size subgroup control charts
  - p Chart (Fraction or Percent of Defective Parts)
  - u-Chart (Number of Defects per Unit)

Event-Based, Time-Based and Batch-Based SPC Charts

In Rev. 3.0, we have converted the underlying coordinate system for both Time-based and Batch-based to event coordinates. This simplifies the underlying code because the software no longer has to use two different types of coordinate systems (Time and Linear) to handle the Time-based and Batch-based SPC charts, respectively. Originally we designed the Event coordinate system for use in plotting stock market data. Stock market data is characterized by irregular time stamps on the underlying data. Stocks can stop trading at 4:00 in the afternoon, and start trading again the next day at 9:30. Also, stocks don't trade on NSYE on weekends, holidays, and other special days. And the time stamps on stock trades do not have to fall on equally spaced intervals. But most users of stock market charts want do not want to see long gaps in the data corresponding to time periods when the stock is not trading (nighttime, weekends, holidays, etc.). The Event coordinate system resolves this issue by using a discrete linear scale for the x-axis which is scaled from 0 to the number of events minus one. Each specific event, for example a stock ticker event, is captured and plotted as a new item on the event scale. Adjacent event are always plotted with equal spacing, regardless of the time stamp of the event. So an event captured at 4:00 PM one day can end up adjacent to an event captured at 9:30 the following day. It turns out, this type of scale is perfect for capturing SPC Chart data. It combines the best features of both time-coordinates and batch-coordinates. It can be used for either one. So we creating to new classes, SPCEventVariableControlChart, and SPCEventAttributeControlChart, both of which derive from the SPCChartBase. Both are used pretty much identical to the SPCBatchVariableControlChart and SPCBatchAttributeControlChart. The change in the class derivation looks like:

Rev. 2.3 and earlier

```
ChartView
    SPCChartBase
        SPCTimeVariableControlChart
        SPCBatchVariableControlChart
        SPCTimeAttributeControlChart
        SPCBatchAttributeControlChart
```

Revision 3.0

```
ChartView
    SPCChartBase
        SPCEventVariableControlChart
            SPCTimeVariableControlChart
            SPCBatchVariableControlChart
        SPCEventAttributeControlChart
            SPCTimeAttributeControlChart
            SPCBatchAttributeControlChart
```

So, while Revision 3.0 includes the SPCTimeVariableControlChart, SPCBatchVariableControlChart, SPCTimeAttributeControlChart, SPCBatchAttributeControlChart classes found in earlier versions, they are derived from the SPCEventVariableControlChart and SPCEventAttributeControlChart classes and use the new Event coordinate system for all drawing.

Changes when using the SPCTimeVariableControlChart and SPCTimeAttributeControlChart.

The major change is in the time-coordinate charts, where even if your sample interval is not consistent, the resulting charts will place the data points of the chart at equally spaced intervals. In the original time-base SPC charts, the x-axis tick marks are auto-scaled to rounded time (hh:mm) values, such as 18:00, 19:00, 20:00. The axis tick marks did not necessarily line up with the sample interval spacing because their time stamps may not fall on the exact same spot as the axis tick marks (18:03, 18:57, 19:55...). In the new version of the SPCTime charts, the tick marks will always correspond exactly to the sample interval, and the time label for the tick mark will always display the time stamp of that tick mark. So, now the tick marks will be labeled with 18:03, 18:57, 1955, and also will space the tick marks evenly, even though the sample interval is not even. So if you want nice even values for your axis tick marks, you need to time stamp your sample interval data with nice even values.

Other Quality Control Charts

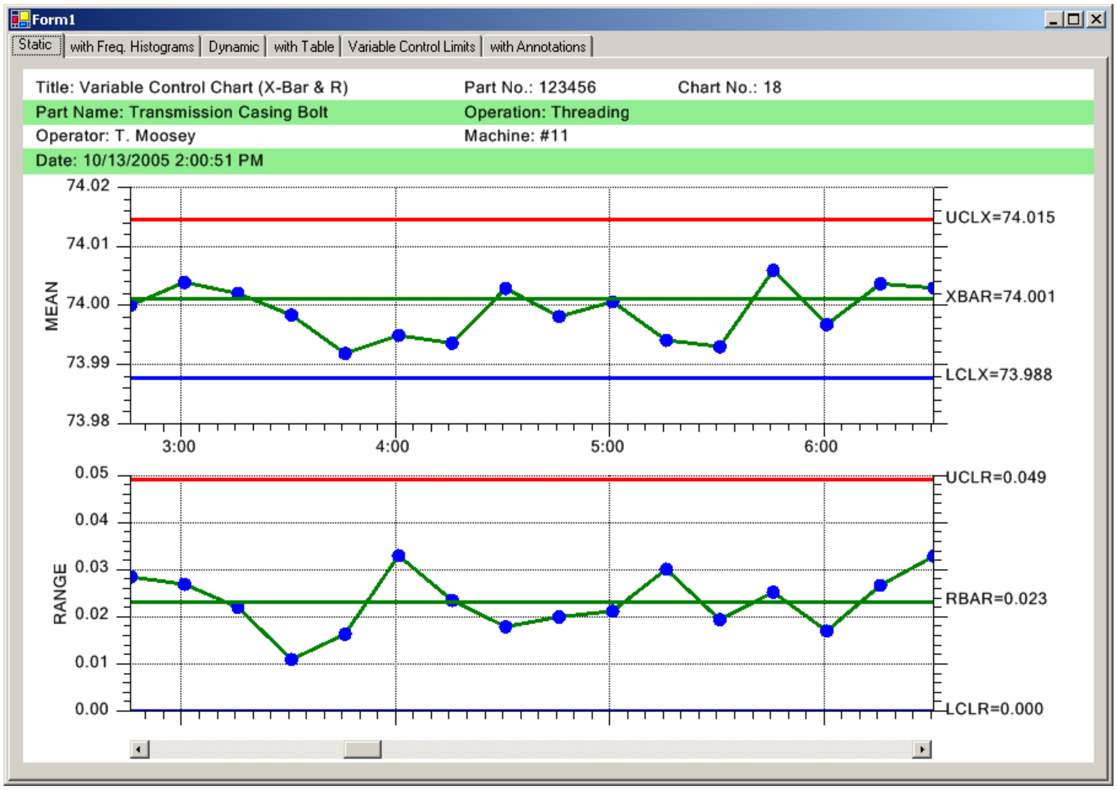
Quality engineers use other, specialized, charts in the analysis of SPC data. We have added chart classes that implement the following SPC analysis charts:

- Frequency Histograms
- Probability Charts
- Pareto Charts

Variable Control Charts

Variable Control Charts are for use with sampled quality data that can be assigned a specific numeric value, other than just 0 or 1. This might include, but is not limited to, the measurement of a critical dimension (height, length, width, radius, etc.), the weight a specific component, or the measurement of an important voltage. Common types of *Variable Control Charts* include X-Bar R (Mean and Range), X-Bar Sigma, Median and Range, X-R (Individual Range), EWMA, MA, MAMR (Moving Average/Moving Range), MAMS (Moving Average/Moving Sigma), Levey-Jennings and CuSum charts.

Typical Time-Base Variable Control Chart (X-Bar R) with header information



Title: Variable Control Chart (X-Bar & R) Part No.: 123456 Chart No.: 18

Part Name: Transmission Casing Bolt Operation: Threading

Operator: T. Moosey Machine: #11

Date: 10/13/2005 2:00:51 PM

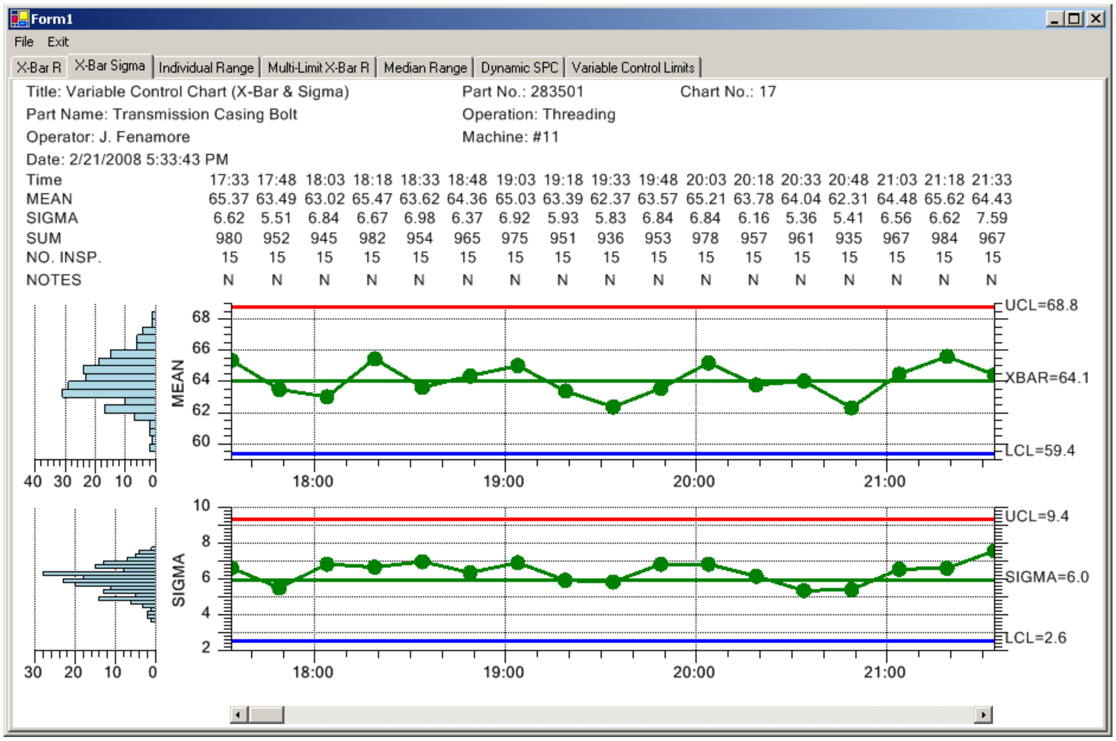
X-Bar R Chart – Also known as the Mean (or Average) and Range Chart

The X-Bar R chart monitors the trend of a critical process variable over time using a statistical sampling method that results in a subgroup of values at each subgroup interval. The X-Bar part of the chart plots the mean of each sample subgroup and the Range part of the chart monitors the difference between the minimum and maximum value in the subgroup.

X-Bar Sigma Chart

Very similar to the X-Bar R Chart, the X-Bar Sigma chart replaces the Range plot with a Sigma plot based on the standard deviation of the measured values within each subgroup. This is a more accurate way of establishing control limits if the sample size of the subgroup is moderately large (> 10). Though computationally more complicated, the use of a computer makes this a non-issue.

Fixed sample size X-Bar Sigma Control chart with header information

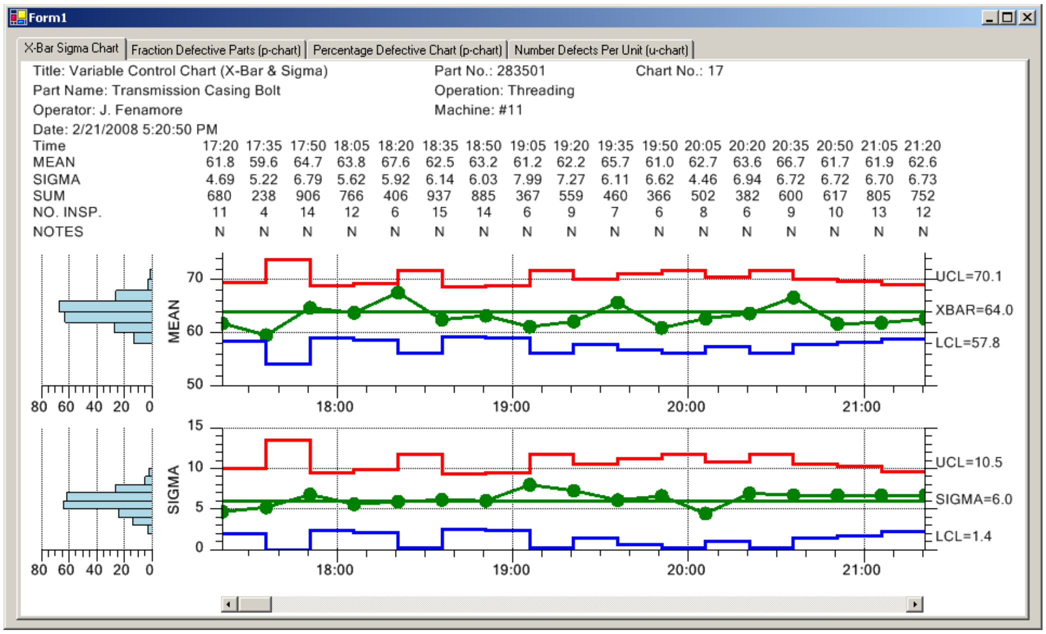


Title: Variable Control Chart (X-Bar & Sigma) Part No.: 283501 Chart No.: 17
Part Name: Transmission Casing Bolt Operation: Threading
Operator: J. Fenamore Machine: #11
Date: 2/21/2008 5:33:43 PM

Time	17:33	17:48	18:03	18:18	18:33	18:48	19:03	19:18	19:33	19:48	20:03	20:18	20:33	20:48	21:03	21:18	21:33
MEAN	65.37	63.49	63.02	65.47	63.62	64.36	65.03	63.39	62.37	63.57	65.21	63.78	64.04	62.31	64.48	65.62	64.43
SIGMA	6.62	5.51	6.84	6.67	6.98	6.37	6.92	5.93	5.83	6.84	6.84	6.16	5.36	5.41	6.56	6.62	7.59
SUM	980	952	945	982	954	965	975	951	936	953	978	957	961	935	967	984	967
NO. INSP.	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15
NOTES	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N

The X-Bar Sigma chart can also be used if the sample subgroup size varies from sampling interval to sampling interval. In this case, the control chart high and low limits vary from sample interval to sample interval, depending on the number of samples in the associated sample subgroup. A low number of samples in the sample subgroup make the band between the high and low limits wider than if a higher number of samples are available. The X-Bar Sigma chart is the only variable control chart that can be used with a variable sample size.

X-Bar Sigma Chart with variable sample size



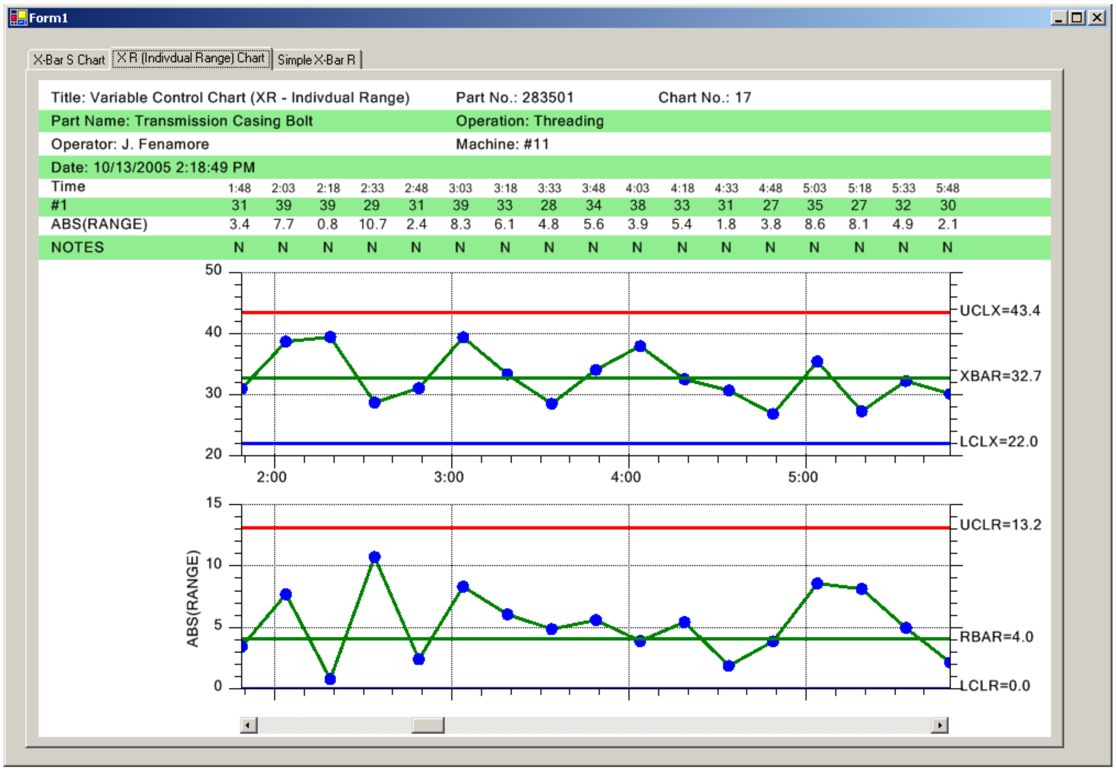
Title: Variable Control Chart (X-Bar & Sigma) Part No.: 283501 Chart No.: 17
Part Name: Transmission Casing Bolt Operation: Threading
Operator: J. Fenamore Machine: #11
Date: 2/21/2008 5:20:50 PM

Time	17:20	17:35	17:50	18:05	18:20	18:35	18:50	19:05	19:20	19:35	19:50	20:05	20:20	20:35	20:50	21:05	21:20
MEAN	61.8	59.6	64.7	63.8	67.6	62.5	63.2	61.2	62.2	65.7	61.0	62.7	63.6	66.7	61.7	61.9	62.6
SIGMA	4.69	5.22	6.79	5.62	5.92	6.14	6.03	7.99	7.27	6.11	6.62	4.46	6.94	6.72	6.72	6.70	6.73
SUM	680	238	906	766	406	937	885	367	559	460	366	502	382	600	617	805	752
NO. INSP.	11	4	14	12	6	15	14	6	9	7	6	8	6	9	10	13	12
NOTES	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N

Median Range – Also known as the Median and Range Chart

Very similar to the X-Bar R Chart, Median Range chart replaces the Mean plot with a Median plot representing the median of the measured values within each subgroup. The Median Range chart requires that the process be well behaved, where the variation in measured variables are (1) known to be distributed normally, (2) are not very often disturbed by assignable causes, and (3) are easily adjusted.

Typical Time-Based Individual Range Chart (X-R) with data table



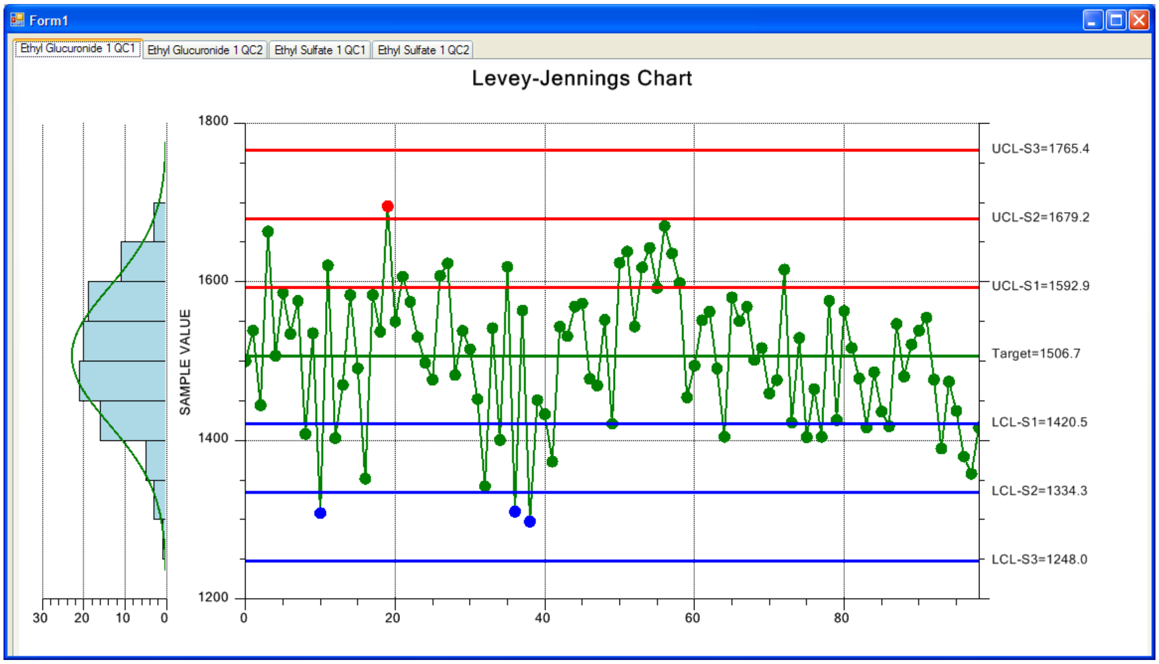
Title: Variable Control Chart (XR - Indivdual Range) Part No.: 283501 Chart No.: 17
Part Name: Transmission Casing Bolt Operation: Threading
Operator: J. Fenamore Machine: #11
Date: 10/13/2005 2:18:49 PM

Time	1:48	2:03	2:18	2:33	2:48	3:03	3:18	3:33	3:48	4:03	4:18	4:33	4:48	5:03	5:18	5:33	5:48
#1	31	39	39	29	31	39	33	28	34	38	33	31	27	35	27	32	30
ABS(RANGE)	3.4	7.7	0.8	10.7	2.4	8.3	6.1	4.8	5.6	3.9	5.4	1.8	3.8	8.6	8.1	4.9	2.1
NOTES	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N

Individual Range Chart – Also known as the X-R Chart

The Individual Range Chart is used when the sample size for a subgroup is 1. This happens frequently when the inspection and collection of data for quality control purposes is automated and 100% of the units manufactured are analyzed. It also happens when the production rate is low and it is inconvenient to have sample sizes other than 1. The X part of the control chart plots the actual sampled value (not a mean or median) for each unit and the R part of the control chart plots a moving range, calculated using the current value of sampled value minus the previous value.

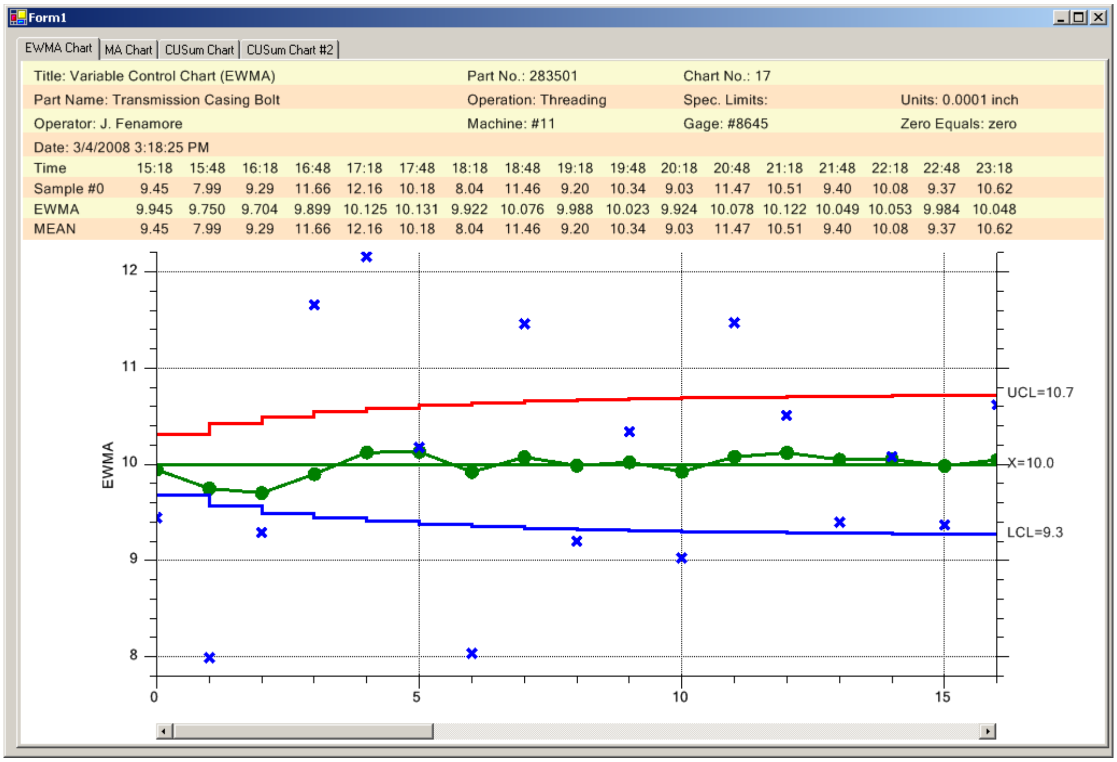
Typical Levey-Jennings Chart using Batch Sampling



Levey-Jennings Chart

The Levey-Jennings chart is used almost exclusively in laboratory settings. It uses a chart very similar to the Individual Range chart above, the major difference being that it only uses the Primary individual data point graph of the chart and does not include the Secondary range graph. Also, the Levey-Jennings chart uses the Westgard rules which utilizes tests involving 1-, 2- and 3- sigma control limits. The control limit calculations depart from all of the other SPC Chart types in that the target value (mean) and control limit (sigma) calculations use the overall mean and standard deviation values from the entire, charted, sample population. See the links https://en.wikipedia.org/wiki/Laboratory_quality_control and https://www.westgard.com/lesson12.htm for more information about the underlying principles of the Levey-Jennings chart.

Typical EWMA Chart using Batch Sampling



Title: Variable Control Chart (EWMA) Part No.: 283501 Chart No.: 17

Part Name: Transmission Casing Bolt Operation: Threading Spec. Limits: Units: 0.0001 inch

Operator: J. Fenamore Machine: #11 Gage: #8645 Zero Equals: zero

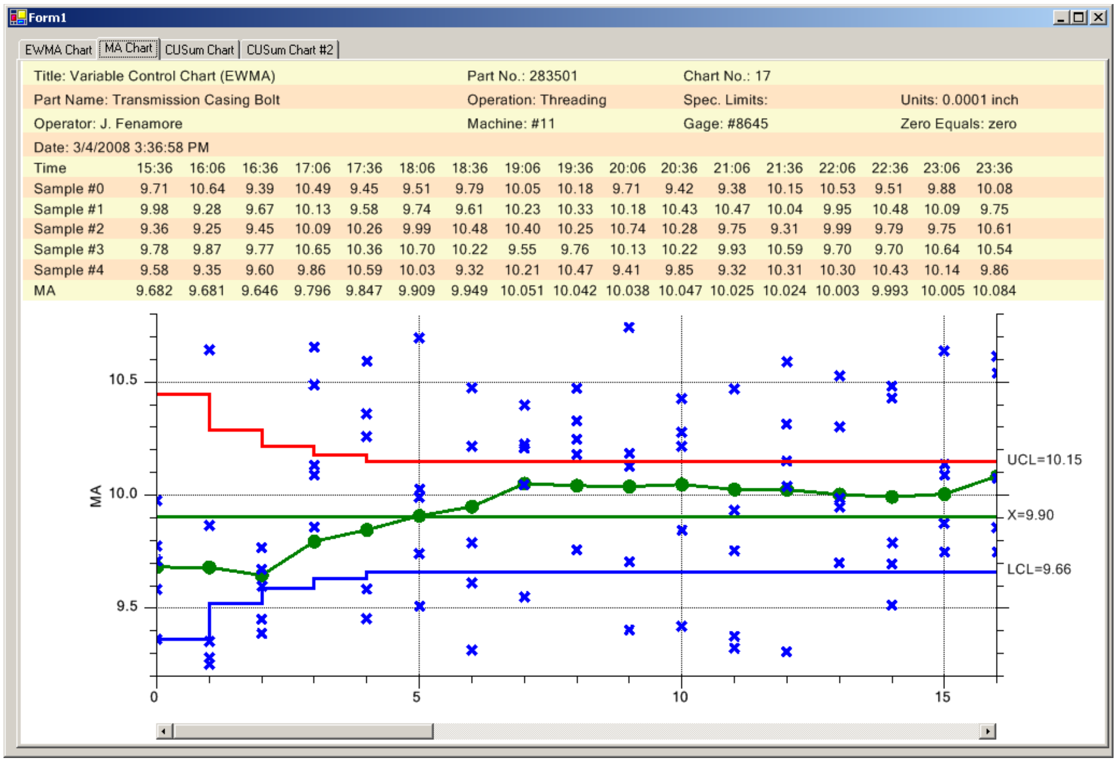
Date: 3/4/2008 3:18:25 PM

Time	15:18	15:48	16:18	16:48	17:18	17:48	18:18	18:48	19:18	19:48	20:18	20:48	21:18	21:48	22:18	22:48	23:18
Sample #0	9.45	7.99	9.29	11.66	12.16	10.18	8.04	11.46	9.20	10.34	9.03	11.47	10.51	9.40	10.08	9.37	10.62
EWMA	9.945	9.750	9.704	9.899	10.125	10.131	9.922	10.076	9.988	10.023	9.924	10.078	10.122	10.049	10.053	9.984	10.048
MEAN	9.45	7.99	9.29	11.66	12.16	10.18	8.04	11.46	9.20	10.34	9.03	11.47	10.51	9.40	10.08	9.37	10.62

EWMA Chart – Exponentially Weighted Moving Average

The EWMA chart is an alternative to the preceding Shewhart type control charts (X-Bar R and I-R charts in particular) and is most useful for detecting small shifts in the process mean. It uses a weighted moving average of previous values to "smooth" the incoming data, minimizing the affect of random noise on the process. It weights the current and most recent values more heavily than older values, allowing the control line to react faster than a simple MA (Moving Average) plot to changes in the process. Like the Shewhart charts, if the EWMA value exceeds the calculated control limits, the process is considered out of control. While it is usually used where the process uses 100% inspection and the sample subgroup size is 1 (same is the I-R chart), it can also be used when sample subgroup sizes are greater than one.

MA (Moving Average) Chart with Sample Values Plotted



Title: Variable Control Chart (EWMA) Part No.: 283501 Chart No.: 17
Part Name: Transmission Casing Bolt Operation: Threading Spec. Limits: Units: 0.0001 inch
Operator: J. Fenamore Machine: #11 Gage: #8645 Zero Equals: zero
Date: 3/4/2008 3:36:58 PM

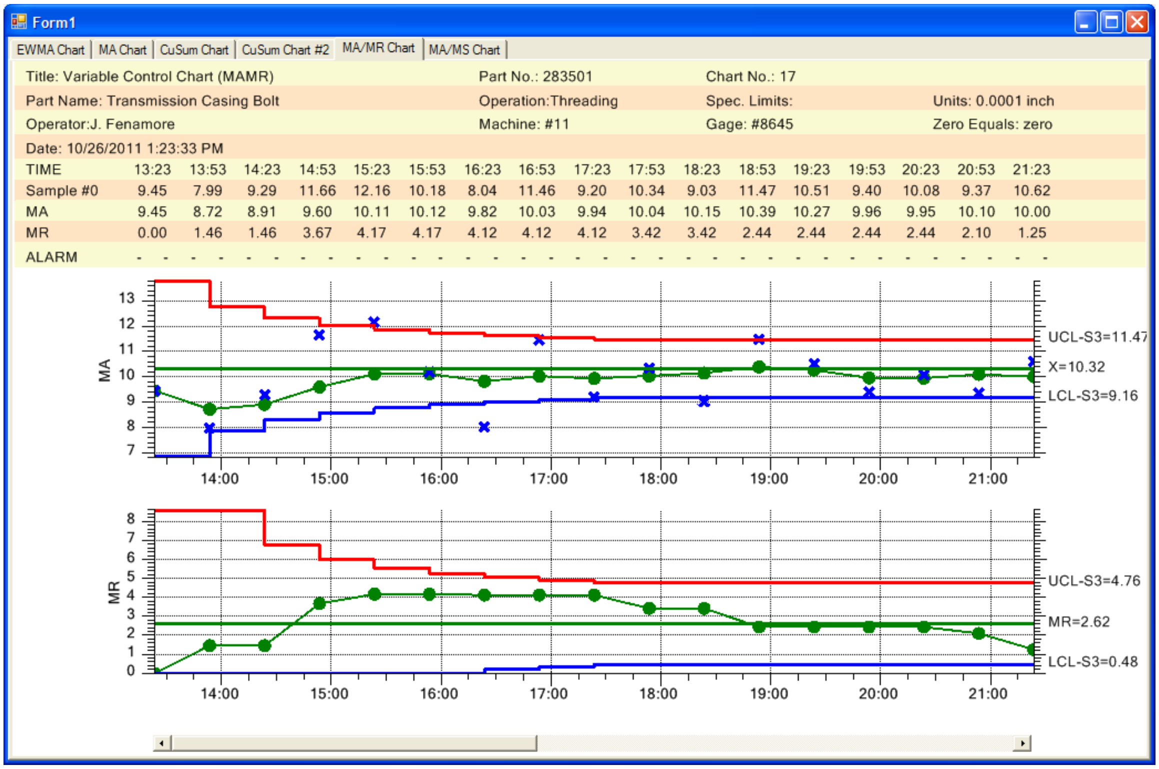
Time	15:36	16:06	16:36	17:06	17:36	18:06	18:36	19:06	19:36	20:06	20:36	21:06	21:36	22:06	22:36	23:06	23:36
Sample #0	9.71	10.64	9.39	10.49	9.45	9.51	9.79	10.05	10.18	9.71	9.42	9.38	10.15	10.53	9.51	9.88	10.08
Sample #1	9.98	9.28	9.67	10.13	9.58	9.74	9.61	10.23	10.33	10.18	10.43	10.47	10.04	9.95	10.48	10.09	9.75
Sample #2	9.36	9.25	9.45	10.09	10.26	9.99	10.48	10.40	10.25	10.74	10.28	9.75	9.31	9.99	9.79	9.75	10.61
Sample #3	9.78	9.87	9.77	10.65	10.36	10.70	10.22	9.55	9.76	10.13	10.22	9.93	10.59	9.70	9.70	10.64	10.54
Sample #4	9.58	9.35	9.60	9.86	10.59	10.03	9.32	10.21	10.47	9.41	9.85	9.32	10.31	10.30	10.43	10.14	9.86
MA	9.682	9.681	9.646	9.796	9.847	9.909	9.949	10.051	10.042	10.038	10.047	10.025	10.024	10.003	9.993	10.005	10.084

MA Chart – Moving Average

The MA chart is another alternative to the preceding Shewhart type control charts (X-Bar R and I-R charts in particular) and is most useful for detecting small shifts in the process mean. It uses a moving average, where the previous (N-1) sample values of the process variable are averaged together along with the process value to produce the current chart value. This helps to "smooth" the incoming data, minimizing the affect of random noise on the process. Unlike the EWMA chart, the MA chart weights the current and previous (N-1) values equally in the average. While the MA chart can often detect small changes in the process mean faster than the Shewhart chart types, it is generally considered inferior to the EWMA chart. Like the Shewhart charts, if the MA value exceeds the calculated control limits, the process is considered out of control.

MAMR Chart – Moving Average / Moving Range

MAMR (Moving Average/Moving Range) Chart with Sample Values Plotted



Title: Variable Control Chart (MAMR) Part No.: 283501 Chart No.: 17

Part Name: Transmission Casing Bolt Operation:Threading Spec. Limits: Units: 0.0001 inch

Operator:J. Fenamore Machine: #11 Gage: #8645 Zero Equals: zero

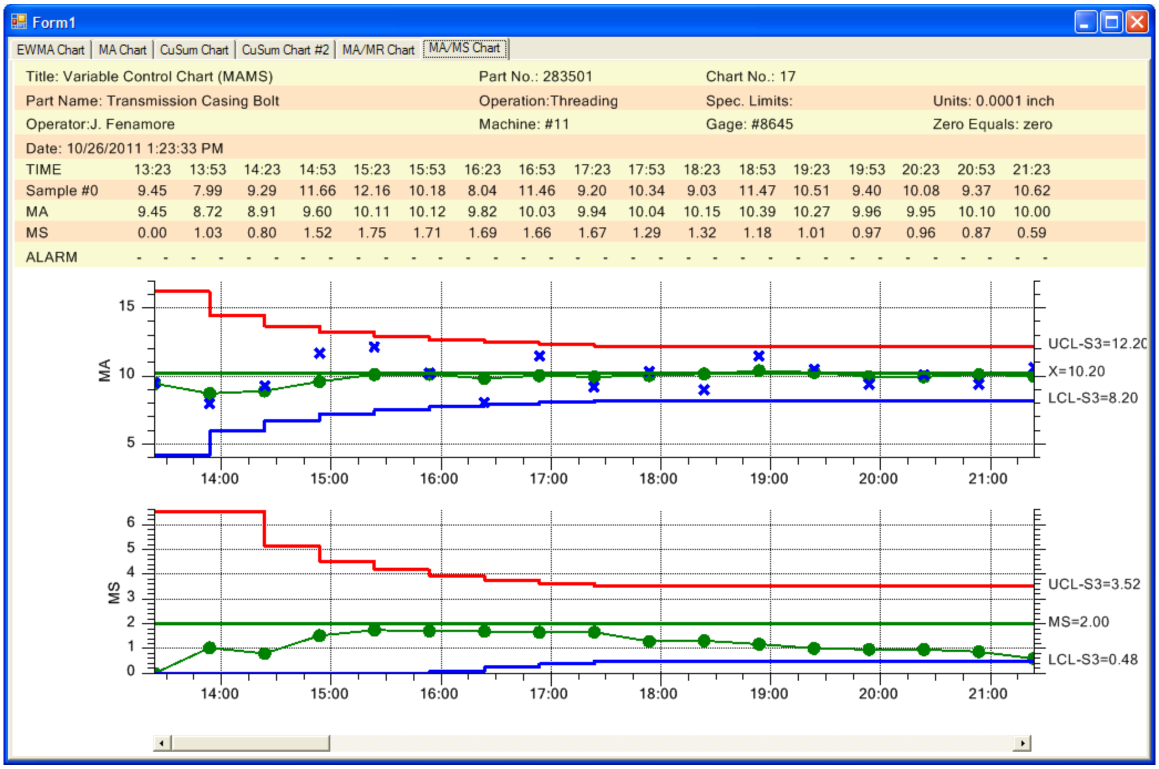
Date: 10/26/2011 1:23:33 PM

TIME	13:23	13:53	14:23	14:53	15:23	15:53	16:23	16:53	17:23	17:53	18:23	18:53	19:23	19:53	20:23	20:53	21:23
Sample #0	9.45	7.99	9.29	11.66	12.16	10.18	8.04	11.46	9.20	10.34	9.03	11.47	10.51	9.40	10.08	9.37	10.62
MA	9.45	8.72	8.91	9.60	10.11	10.12	9.82	10.03	9.94	10.04	10.15	10.39	10.27	9.96	9.95	10.10	10.00
MR	0.00	1.46	1.46	3.67	4.17	4.17	4.12	4.12	4.12	3.42	3.42	2.44	2.44	2.44	2.44	2.10	1.25
ALARM	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -

The MAMR chart combines our Moving Average chart with a Moving Range chart. The Moving Average chart is primary (topmost) chart, and the Moving Range chart is the secondary (bottom) chart. It uses a single sample/subgroup, same as our standard [Individual-Range], [Moving Average], [EWMA], and [Moving Average] charts. When calculating the Moving Range, it windows the same data values used in the Moving Average calculation. Note that the limits are variable (wider) at the beginning, taking into account the fewer samples in the start up of any type of SPC chart which uses a sliding window in the calculation of moving average and moving range statistics.

MAMS Chart – Moving Average / Moving Sigma

MAMS (Moving Average/Moving Sigma) Chart with Sample Values Plotted



Title: Variable Control Chart (MAMS) Part No.: 283501 Chart No.: 17

Part Name: Transmission Casing Bolt Operation:Threading Spec. Limits: Units: 0.0001 inch

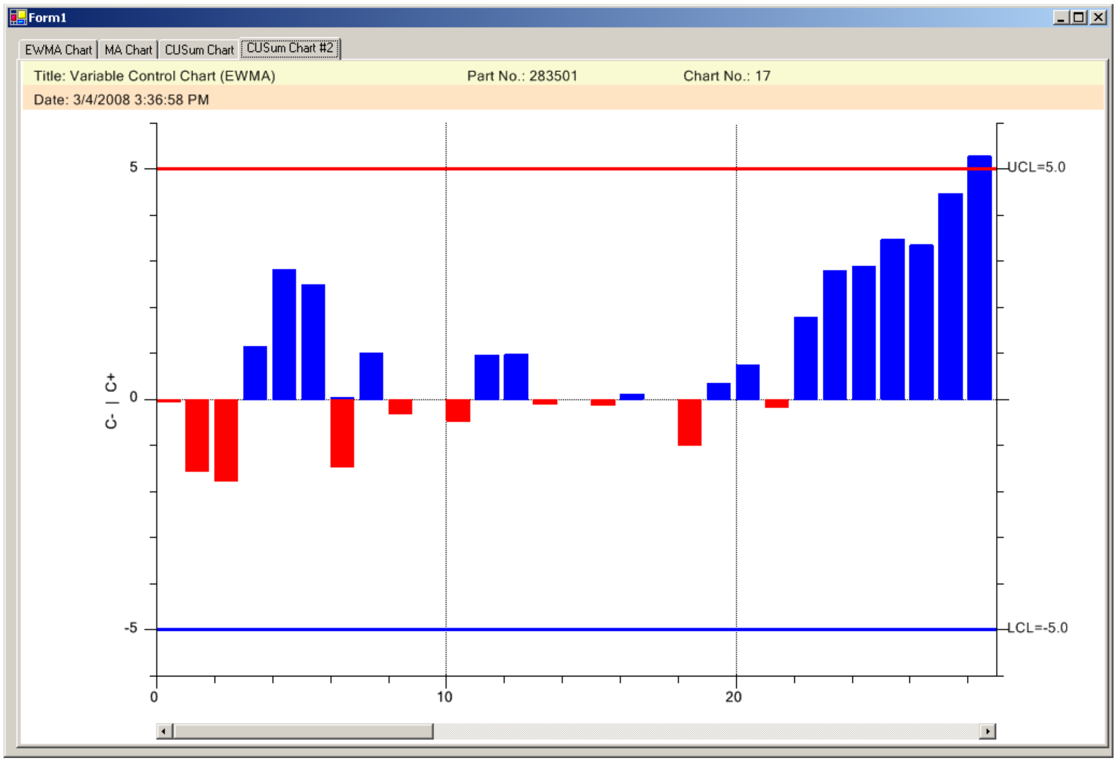
Operator:J. Fenamore Machine: #11 Gage: #8645 Zero Equals: zero

Date: 10/26/2011 1:23:33 PM

TIME	13:23	13:53	14:23	14:53	15:23	15:53	16:23	16:53	17:23	17:53	18:23	18:53	19:23	19:53	20:23	20:53	21:23
Sample #0	9.45	7.99	9.29	11.66	12.16	10.18	8.04	11.46	9.20	10.34	9.03	11.47	10.51	9.40	10.08	9.37	10.62
MA	9.45	8.72	8.91	9.60	10.11	10.12	9.82	10.03	9.94	10.04	10.15	10.39	10.27	9.96	9.95	10.10	10.00
MS	0.00	1.03	0.80	1.52	1.75	1.71	1.69	1.66	1.67	1.29	1.32	1.18	1.01	0.97	0.96	0.87	0.59
ALARM	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -

The MAMS chart combines our Moving Average chart with a Moving Sigma chart. The Moving Average chart is primary (topmost) chart, and the Moving Sigma chart is the secondary (bottom) chart. It uses a single sample/subgroup, same as our standard [Individual-Range], [Moving Average], [EWMA], and [Moving Average] charts. When calculating the Moving Sigma, it windows the same data values used in the Moving Average calculation. Note that the limits are variable (wider) at the beginning, taking into account the fewer samples in the start up of any type of SPC chart which uses a sliding window in the calculation of moving average and moving sigma statistics.

Tabular CuSum Chart



Title: Variable Control Chart (EWMA) Part No.: 283501 Chart No.: 17

Date: 3/4/2008 3:36:58 PM

CuSum Chart – Tabular, one-sided, upper and lower cumulative sum

The CuSum chart is a specialized control chart, which like the EWMA and MA charts, is considered to be more efficient that the Shewhart charts at detecting small shifts in the process mean, particularly if the mean shift is less than 2 sigma. There are several types of CuSum charts, but the easiest to use and the most accurate is considered the tabular CuSum chart and that is the one implemented in this software. The tabular cusum works by accumulating deviations that are above the process mean in one statistic (C+) and accumulating deviations below the process mean in a second statistic (C-). If either statistic (C+ or C-) falls outside of the calculated limits, the process is considered out of control.

Measured Data and Calculated Value Tables

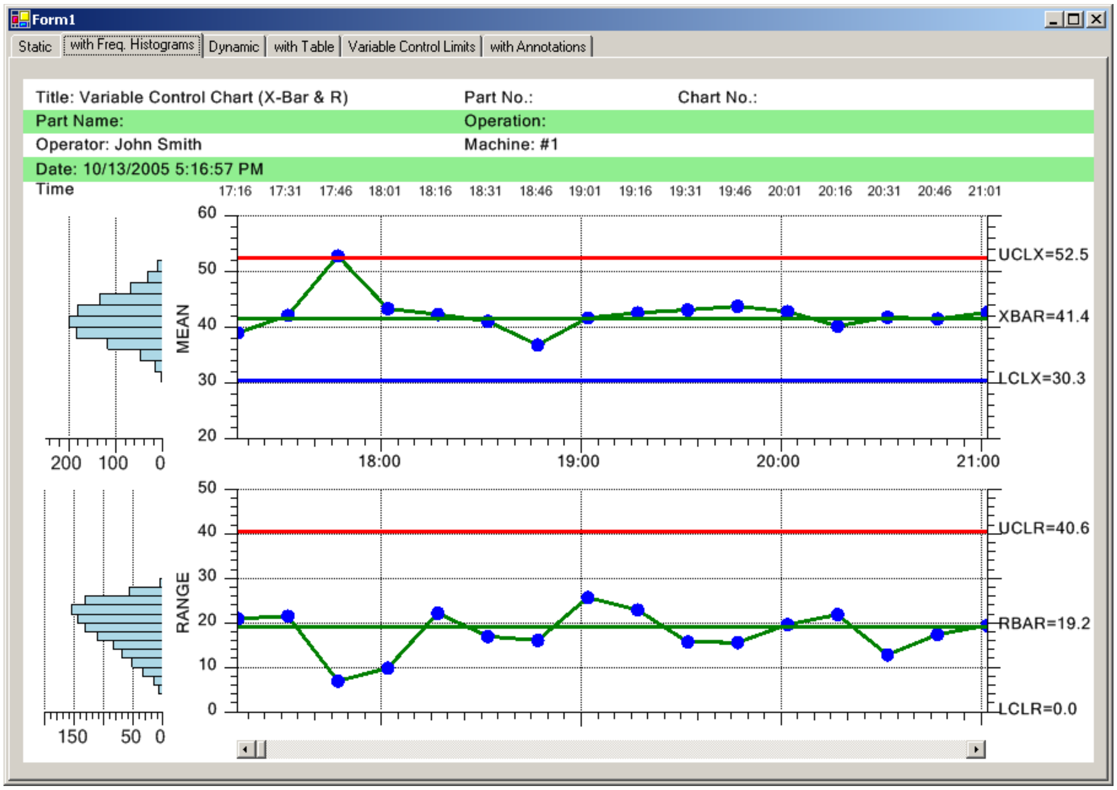
Standard worksheets used to gather and plot SPC data consist of three main parts.

- The first part is the header section, identifying the title of the chart, the monitored process, the machine operator, part number and other important information specific to the chart.

- The second part is the measurement data recording and calculation section, organized as a table, recording the sampled and calculated data in a neat, readable fashion.
- The third part, the actual SPC chart, plots the calculated SPC values for the sample group

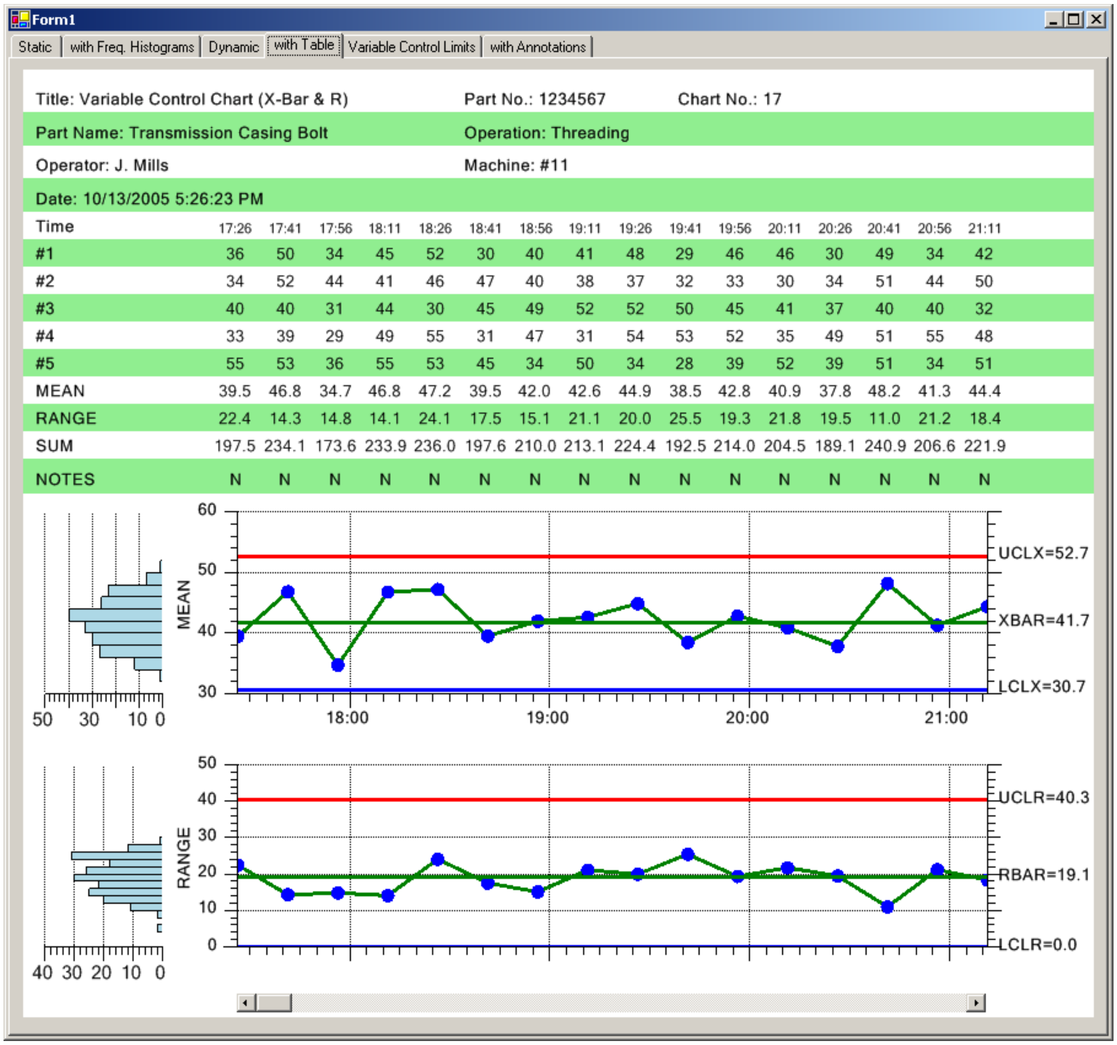
The *Variable Control Chart* templates that we have created have options that enable the programmer to customize and automatically include header information along with a table of the measurement and calculated data, in the SPC chart. Enable the scrollbar option and you can display the tabular measurement data and SPC plots for a window of 8-20 subgroups, from a much larger collection of measurement data represented hundreds or even thousands of subgroups, and use the scrollbar to move through the data, similar to paging through a spreadsheet.

Scrollable Time-Based XBar-R Chart with frequency histograms and basic header information



Title: Variable Control Chart (X-Bar & R) Part No.: Chart No.:
Part Name: Operation:
Operator: John Smith Machine: #1
Date: 10/13/2005 5:16:57 PM

Scrollable Time-Based XBar-R Chart with frequency histograms, header, measurement and calculated value information



Title: Variable Control Chart (X-Bar & R) Part No.: 1234567 Chart No.: 17

Part Name: Transmission Casing Bolt Operation: Threading

Operator: J. Mills Machine: #11

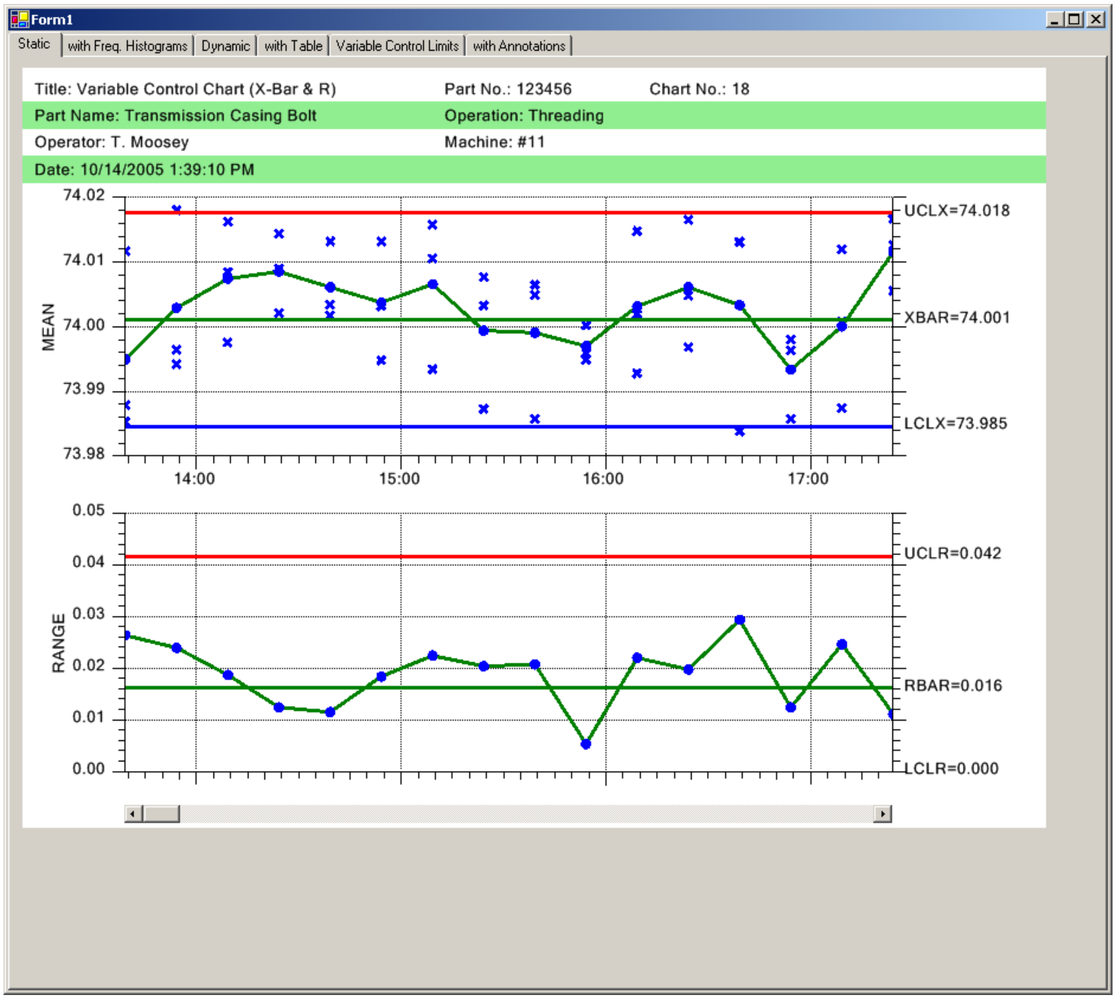
Date: 10/13/2005 5:26:23 PM

Time	17:26	17:41	17:56	18:11	18:26	18:41	18:56	19:11	19:26	19:41	19:56	20:11	20:26	20:41	20:56	21:11
#1	36	50	34	45	52	30	40	41	48	29	46	46	30	49	34	42
#2	34	52	44	41	46	47	40	38	37	32	33	30	34	51	44	50
#3	40	40	31	44	30	45	49	52	52	50	45	41	37	40	40	32
#4	33	39	29	49	55	31	47	31	54	53	52	35	49	51	55	48
#5	55	53	36	55	53	45	34	50	34	28	39	52	39	51	34	51
MEAN	39.5	46.8	34.7	46.8	47.2	39.5	42.0	42.6	44.9	38.5	42.8	40.9	37.8	48.2	41.3	44.4
RANGE	22.4	14.3	14.8	14.1	24.1	17.5	15.1	21.1	20.0	25.5	19.3	21.8	19.5	11.0	21.2	18.4
SUM	197.5	234.1	173.6	233.9	236.0	197.6	210.0	213.1	224.4	192.5	214.0	204.5	189.1	240.9	206.6	221.9
NOTES	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N

Scatter Plots of the Actual Sampled Data

In some cases it useful to plot the actual values of a sample subgroup along with the sample subgroup mean or median. Plot these samples in the SPC chart using additional scatter plots.

Scrollable Time-Based XBar-R Chart with Scatter Plot of Actual Sampled Data

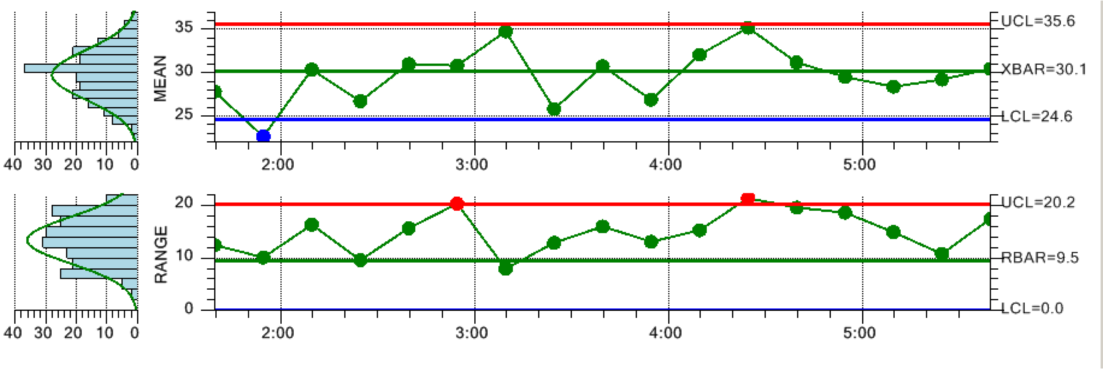


Title: Variable Control Chart (X-Bar & R) Part No.: 123456 Chart No.: 18
Part Name: Transmission Casing Bolt Operation: Threading
Operator: T. Moosey Machine: #11
Date: 10/14/2005 1:39:10 PM

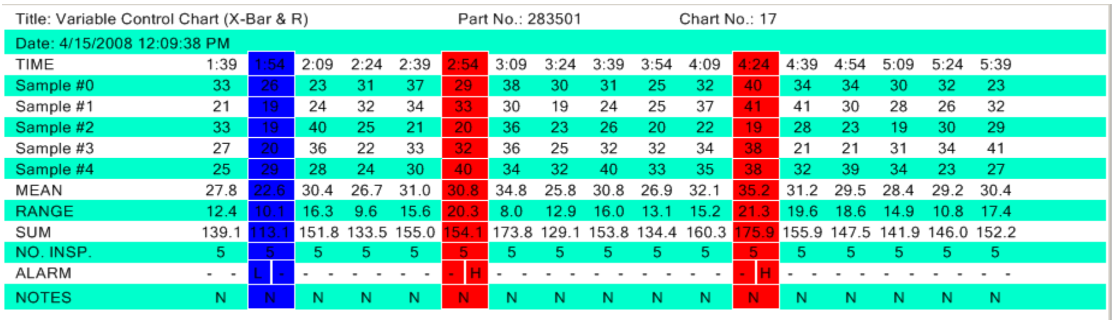
Alarm Notification

Typically, when a process value exceeds a control limit, an alarm condition exists. In order to make sure that the program user identifies an alarm you can emphasize the alarm in several different ways. You can trap the alarm condition using an event delegate, log the alarm to the notes log, highlight the data point symbol in the chart where the alarm occurs, display an alarm status line in the data table, or highlight the entire column of the sample interval where the alarm occurs.

Change the color of a data point that falls outside of alarm limits.



Highlight the column of the sample interval where the alarm occurs

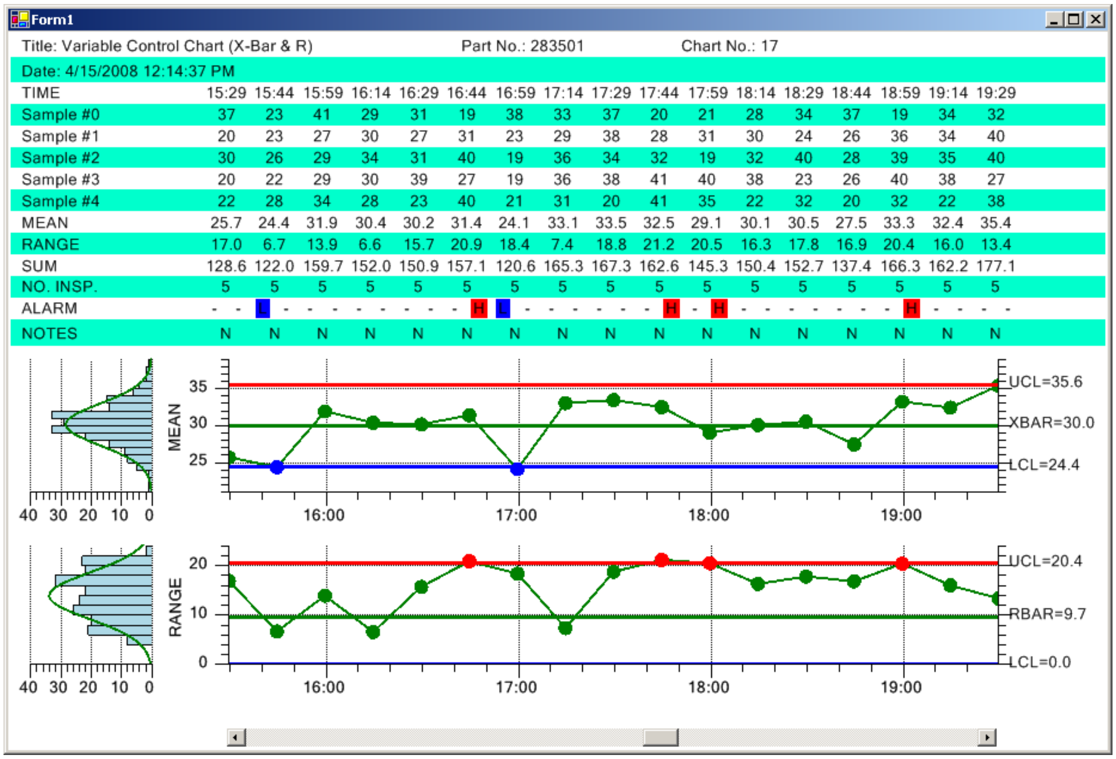


Title: Variable Control Chart (X-Bar & R) Part No.: 283501 Chart No.: 17

Date: 4/15/2008 12:09:38 PM

TIME	1:39	1:54	2:09	2:24	2:39	2:54	3:09	3:24	3:39	3:54	4:09	4:24	4:39	4:54	5:09	5:24	5:39
Sample #0	33	26	23	31	37	29	38	30	31	25	32	40	34	34	30	32	23
Sample #1	21	19	24	32	34	33	30	19	24	25	37	41	41	30	28	26	32
Sample #2	33	19	40	25	21	20	36	23	26	20	22	19	28	23	19	30	29
Sample #3	27	20	36	22	33	32	36	25	32	32	34	38	21	21	31	34	41
Sample #4	25	29	28	24	30	40	34	32	40	33	35	38	32	39	34	23	27
MEAN	27.8	22.6	30.4	26.7	31.0	30.8	34.8	25.8	30.8	26.9	32.1	35.2	31.2	29.5	28.4	29.2	30.4
RANGE	12.4	10.1	16.3	9.6	15.6	20.3	8.0	12.9	16.0	13.1	15.2	21.3	19.6	18.6	14.9	10.8	17.4
SUM	139.1	113.1	151.8	133.5	155.0	154.1	173.8	129.1	153.8	134.4	160.3	175.9	155.9	147.5	141.9	146.0	152.2
NO. INSP.	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
ALARM	- -	L -	- -	- -	- -	- H	- -	- -	- -	- -	- -	- H	- -	- -	- -	- -	- -
NOTES	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N

An alarm status line highlights an alarm condition, and lets you know when chart the (primary or secondary) the alarm occurs in.



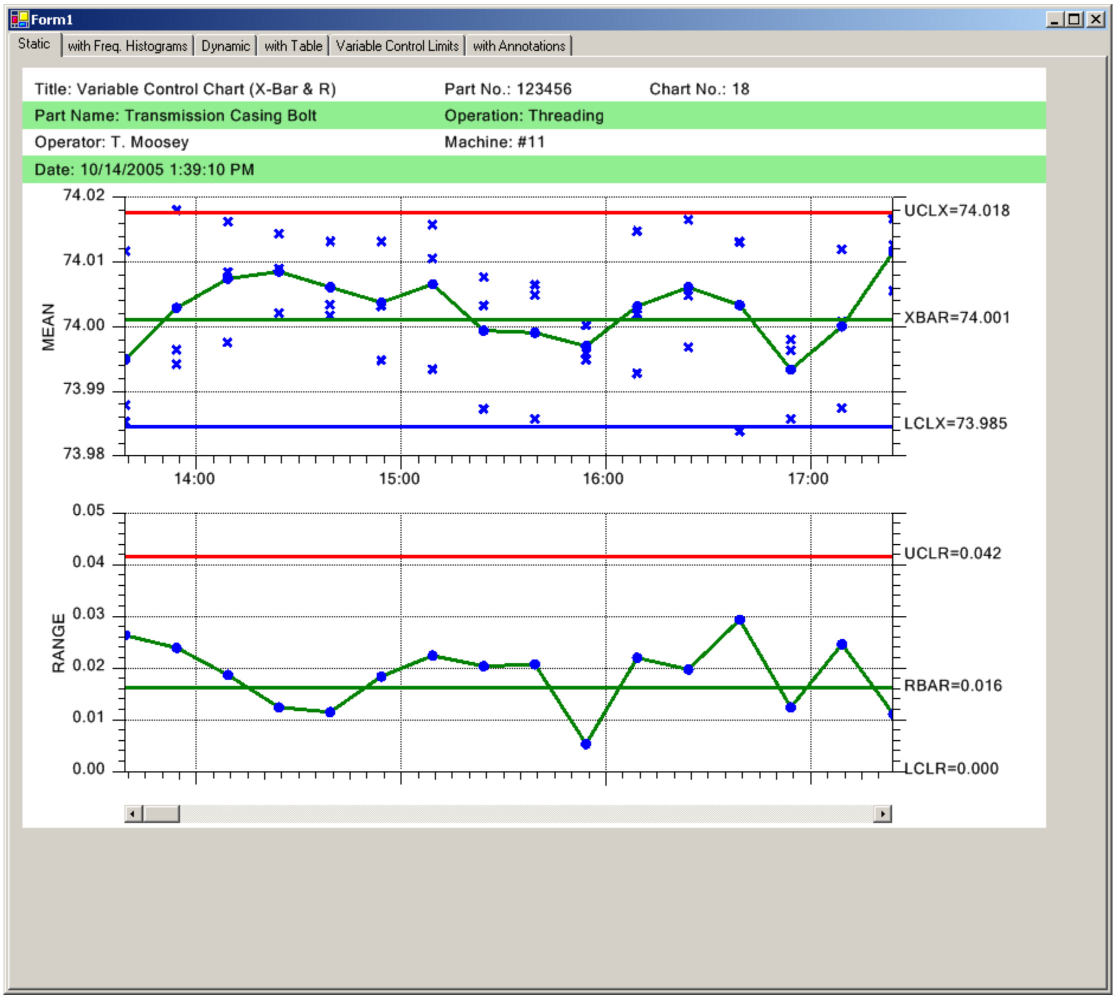
Title: Variable Control Chart (X-Bar & R) Part No.: 283501 Chart No.: 17

Date: 4/15/2008 12:14:37 PM

TIME	15:29	15:44	15:59	16:14	16:29	16:44	16:59	17:14	17:29	17:44	17:59	18:14	18:29	18:44	18:59	19:14	19:29
Sample #0	37	23	41	29	31	19	38	33	37	20	21	28	34	37	19	34	32
Sample #1	20	23	27	30	27	31	23	29	38	28	31	30	24	26	36	34	40
Sample #2	30	26	29	34	31	40	19	36	34	32	19	32	40	28	39	35	40
Sample #3	20	22	29	30	39	27	19	36	38	41	40	38	23	26	40	38	27
Sample #4	22	28	34	28	23	40	21	31	20	41	35	22	32	20	32	22	38
MEAN	25.7	24.4	31.9	30.4	30.2	31.4	24.1	33.1	33.5	32.5	29.1	30.1	30.5	27.5	33.3	32.4	35.4
RANGE	17.0	6.7	13.9	6.6	15.7	20.9	18.4	7.4	18.8	21.2	20.5	16.3	17.8	16.9	20.4	16.0	13.4
SUM	128.6	122.0	159.7	152.0	150.9	157.1	120.6	165.3	167.3	162.6	145.3	150.4	152.7	137.4	166.3	162.2	177.1
NO. INSP.	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
ALARM	- -	L -	- -	- -	- -	- H	L -	- -	- -	- H	- H	- -	- -	- -	- H	- -	- -
NOTES	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N

These alarm highlight features apply to both variable control and attribute control charts.

Scrollable Time-Based XBar-R Chart with Scatter Plot of Actual Sampled Data



Title: Variable Control Chart (X-Bar & R) Part No.: 123456 Chart No.: 18

Part Name: Transmission Casing Bolt Operation: Threading

Operator: T. Moosey Machine: #11

Date: 10/14/2005 1:39:10 PM

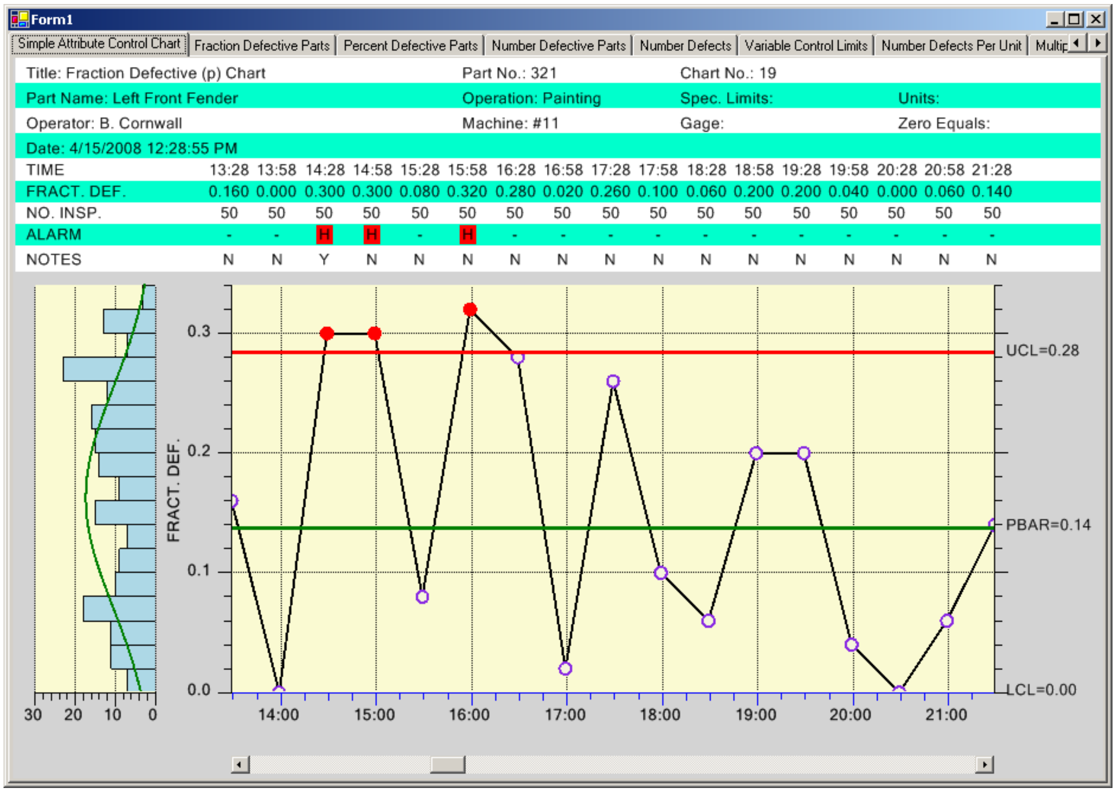
Attribute Control Charts

Attribute Control Charts are a set of control charts specifically designed for tracking defects (also called non-conformities). These types of defects are binary in nature (yes/no), where a part has one or more defects, or it doesn't. Examples of defects are paint scratches, discolorations, breaks in the weave of a textile, dents, cuts, etc. Think of the last car that you bought. The defects in each sample group are counted and run through some statistical calculations. Depending on the type of *Attribute Control Chart*, the number of defective parts are tracked (p-chart and np-chart), or alternatively, the number of defects are tracked (u-chart, c-chart). The difference in terminology "number of defective parts" and "number of defects" is highly significant, since a single part not only can have multiple defect categories (scratch, color, dent, etc), it can also have multiple defects per category. A single part may have 0 – N defects. So keeping track of the number of defective parts is statistically different from keeping track of the number of defects. This affects the way the control limits for each chart are calculated.

Attribute Control Charts

Attribute Control Charts are a set of control charts specifically designed for tracking defects (also called non-conformities). These types of defects are binary in nature (yes/no), where a part has one or more defects, or it doesn't. Examples of defects are paint scratches, discolorations, breaks in the weave of a textile, dents, cuts, etc. Think of the last car that you bought. The defects in each sample group are counted and run through some statistical calculations. Depending on the type of *Attribute Control Chart*, the number of defective parts are tracked (p-chart and np-chart), or alternatively, the number of defects are tracked (u-chart, c-chart). The difference in terminology "number of defective parts" and "number of defects" is highly significant, since a single part not only can have multiple defect categories (scratch, color, dent, etc), it can also have multiple defects per category. A single part may have 0 – N defects. So keeping track of the number of defective parts is statistically different from keeping track of the number of defects. This affects the way the control limits for each chart are calculated.

Typical Time-Based Attribute Control Chart (p-Chart)



Title: Fraction Defective (p) Chart   Part No.: 321   Chart No.: 19
Part Name: Left Front Fender   Operation: Painting   Spec. Limits: Units:
Operator: B. Cornwall   Machine: #11   Gage: Zero Equals:
Date: 4/15/2008 12:28:55 PM

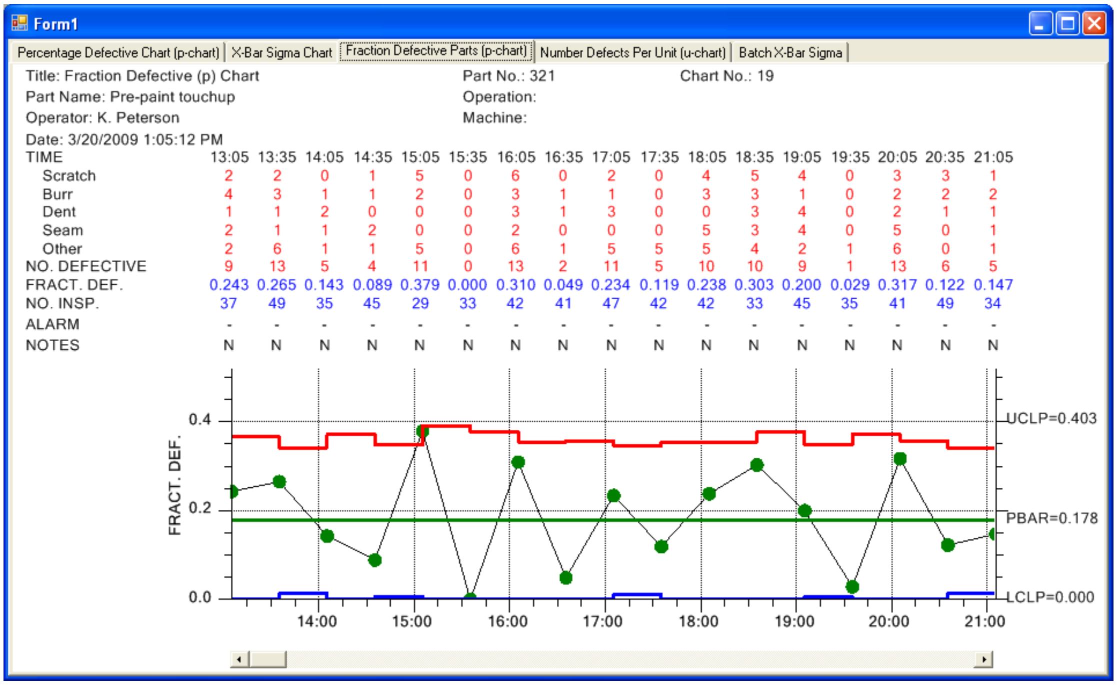
TIME	13:28	13:58	14:28	14:58	15:28	15:58	16:28	16:58	17:28	17:58	18:28	18:58	19:28	19:58	20:28	20:58	21:28
FRACT. DEF.	0.160	0.000	0.300	0.300	0.080	0.320	0.280	0.020	0.260	0.100	0.060	0.200	0.200	0.040	0.000	0.060	0.140
NO. INSP.	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
ALARM	-	-	H	H	-	H	-	-	-	-	-	-	-	-	-	-	-
NOTES	N	N	Y	N	N	N	N	N	N	N	N	N	N	N	N	N	N

p-Chart - Also known as the Percent or Fraction Defective Parts Chart

For a sample subgroup, the number of defective parts is measured and plotted as either a percentage of the total subgroup sample size, or a fraction of the total subgroup sample size. Since the plotted value is a fraction or percent of the sample subgroup size, the size of the sample group can vary without rendering the chart useless.

The p-Chart chart can also be used if the sample subgroup size varies from sampling interval to sampling interval. In this case, the control chart high and low limits vary from sample interval to sample interval, depending on the number of samples in the associated sample subgroup. A low number of samples in the sample subgroup make the band between the high and low limits wider than if a higher number of samples are available. Both the *Fraction Defective Parts and Percent Defective Parts* control charts come in versions that support variable sample sized for a subgroup.

Fraction Defective Parts (p-Chart) with variable sample size



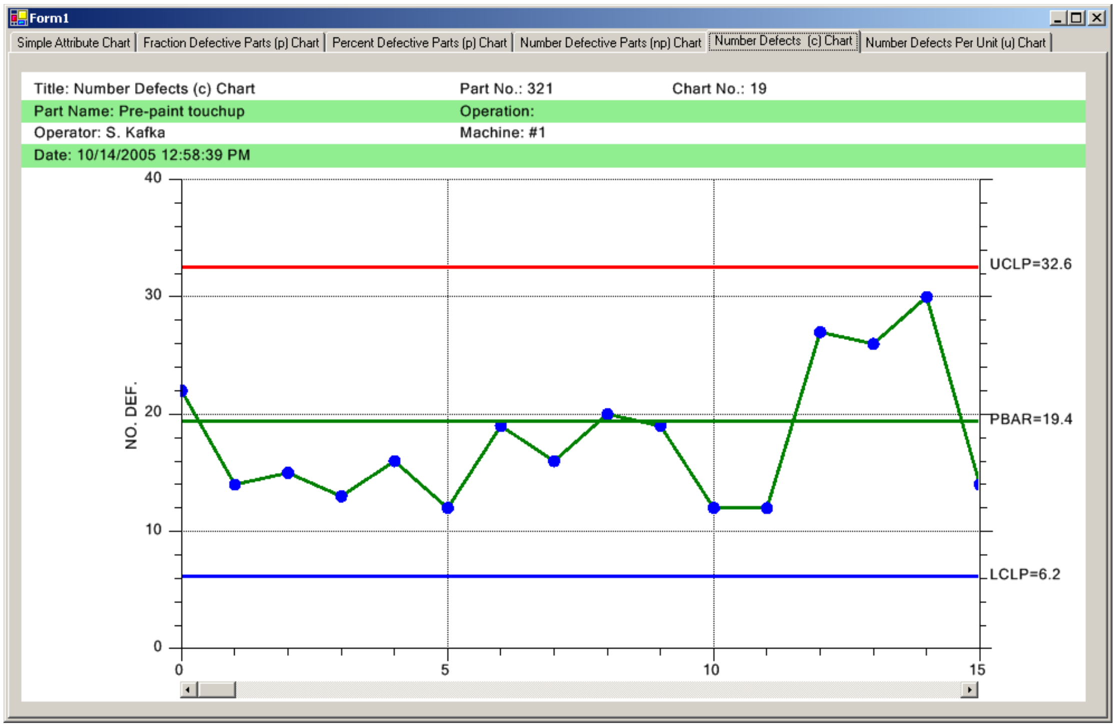
Title: Fraction Defective (p) Chart
Part No.: 321
Chart No.: 19
Part Name: Pre-paint touchup
Operation:
Operator: K. Peterson
Machine:
Date: 3/20/2009 1:05:12 PM

TIME	13:05	13:35	14:05	14:35	15:05	15:35	16:05	16:35	17:05	17:35	18:05	18:35	19:05	19:35	20:05	20:35	21:05
Scratch	2	2	0	1	5	0	6	0	2	0	4	5	4	0	3	3	1
Burr	4	3	1	1	2	0	3	1	1	0	3	3	1	0	2	2	2
Dent	1	1	2	0	0	0	3	1	3	0	0	3	4	0	2	1	1
Seam	2	1	1	2	0	0	2	0	0	0	5	3	4	0	5	0	1
Other	2	6	1	1	5	0	6	1	5	5	5	4	2	1	6	0	1
NO. DEFECTIVE	9	13	5	4	11	0	13	2	11	5	10	10	9	1	13	6	5
FRACT. DEF.	0.243	0.265	0.143	0.089	0.379	0.000	0.310	0.049	0.234	0.119	0.238	0.303	0.200	0.029	0.317	0.122	0.147
NO. INSP.	37	49	35	45	29	33	42	41	47	42	42	33	45	35	41	49	34
ALARM	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
NOTES	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N

np-Chart – Also known as the Number Defective Parts Chart

For a sample subgroup, the number of defective parts is measured and plotted as a simple count. Statistically, in order to compare number of defective parts for one subgroup with the other subgroups, this type of chart requires that the subgroup sample size is fixed across all subgroups.

Typical Number of Defects (c)



Title: Number Defects (c) Chart | Part No.: 321 | Chart No.: 19

Part Name: Pre-paint touchup | Operation:

Operator: S. Kafka | Machine: #1

Date: 10/14/2005 12:58:39 PM

Form1
Simple Attribute Chart
Fraction Defective Parts (p) Chart
Percent Defective Parts (p) Chart
Number Defective Parts (np) Chart
Number Defects (c) Chart
Number Defects Per Unit (u) Chart
NO. DEF.
40
30
20
10
0
0
5
10
15
UCLP=32.6
PBAR=19.4
LCLP=6.2

c-Chart - Also known as the Number of Defects or Number of Non-Conformities Chart

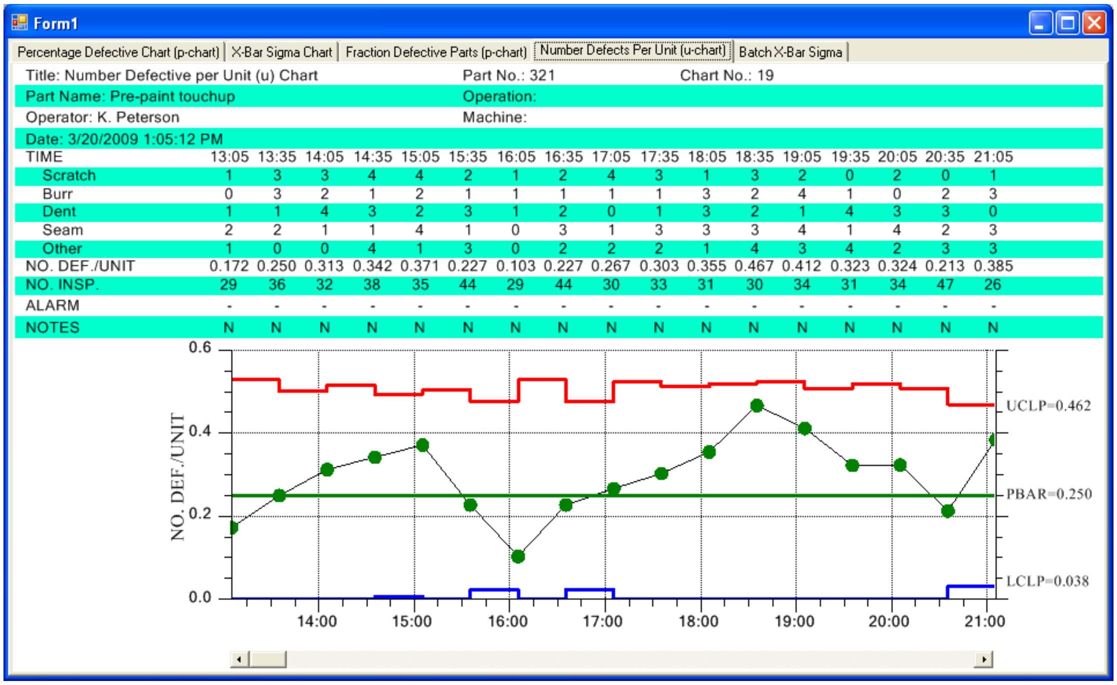
For a sample subgroup, the number of times a defect occurs is measured and plotted as a simple count. Statistically, in order to compare number of defects for one subgroup with the other subgroups, this type of chart requires that the subgroup sample size is fixed across all subgroups.

u-Chart – Also known as the Number of Defects per Unit or Number of Non-Conformities per Unit Chart

For a sample subgroup, the number of times a defect occurs is measured and plotted as either a percentage of the total subgroup sample size, or a fraction of the total subgroup sample size. Since the plotted value is a fraction or percent of the sample subgroup size, the size of the sample group can vary without rendering the chart useless.

The u-Chart chart can also be used if the sample subgroup size varies from sampling interval to sampling interval. In this case, the control chart high and low limits vary from sample interval to sample interval, depending on the number of samples in the associated sample subgroup. A low number of samples in the sample subgroup make the band between the high and low limits wider than if a higher number of samples are available.

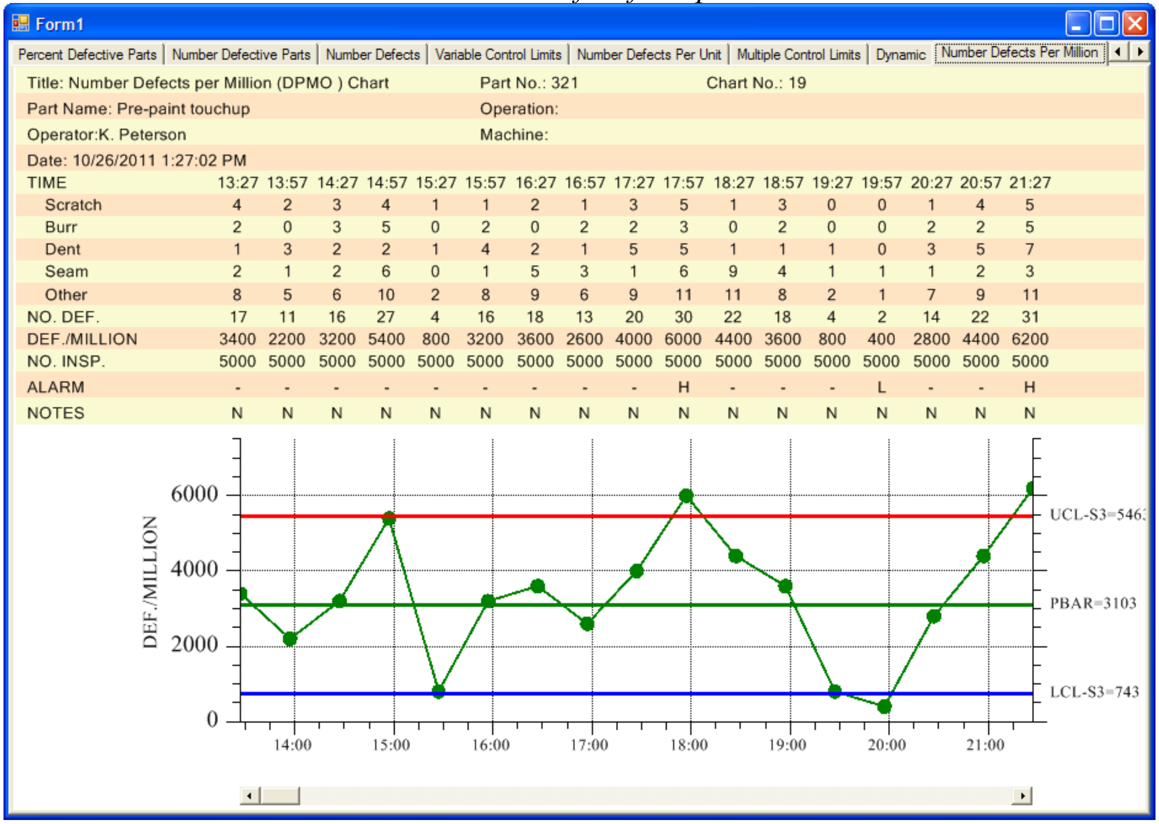
Number of Defects per Unit Chart with variable sample size (u-chart)



Title: Number Defective per Unit (u) Chart   Part No.: 321   Chart No.: 19
Part Name: Pre-paint touchup   Operation:
Operator: K. Peterson   Machine:
Date: 3/20/2009 1:05:12 PM

TIME	13:05	13:35	14:05	14:35	15:05	15:35	16:05	16:35	17:05	17:35	18:05	18:35	19:05	19:35	20:05	20:35	21:05
Scratch	1	3	3	4	4	2	1	2	4	3	1	3	2	0	2	0	1
Burr	0	3	2	1	2	1	1	1	1	1	3	2	4	1	0	2	3
Dent	1	1	4	3	2	3	1	2	0	1	3	2	1	4	3	3	0
Seam	2	2	1	1	4	1	0	3	1	3	3	3	4	1	4	2	3
Other	1	0	0	4	1	3	0	2	2	2	1	4	3	4	2	3	3
NO. DEF./UNIT	0.172	0.250	0.313	0.342	0.371	0.227	0.103	0.227	0.267	0.303	0.355	0.467	0.412	0.323	0.324	0.213	0.385
NO. INSP.	29	36	32	38	35	44	29	44	30	33	31	30	34	31	34	47	26
ALARM	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
NOTES	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N

DPMO Chart – Also known as the Number of Defects per Million Chart



Title: Number Defects per Million (DPMO) Chart   Part No.: 321   Chart No.: 19
Part Name: Pre-paint touchup   Operation:
Operator:K. Peterson   Machine:
Date: 10/26/2011 1:27:02 PM

TIME	13:27	13:57	14:27	14:57	15:27	15:57	16:27	16:57	17:27	17:57	18:27	18:57	19:27	19:57	20:27	20:57	21:27
Scratch	4	2	3	4	1	1	2	1	3	5	1	3	0	0	1	4	5
Burr	2	0	3	5	0	2	0	2	2	3	0	2	0	0	2	2	5
Dent	1	3	2	2	1	4	2	1	5	5	1	1	1	0	3	5	7
Seam	2	1	2	6	0	1	5	3	1	6	9	4	1	1	1	2	3
Other	8	5	6	10	2	8	9	6	9	11	11	8	2	1	7	9	11
NO. DEF.	17	11	16	27	4	16	18	13	20	30	22	18	4	2	14	22	31
DEF./MILLION	3400	2200	3200	5400	800	3200	3600	2600	4000	6000	4400	3600	800	400	2800	4400	6200
NO. INSP.	5000	5000	5000	5000	5000	5000	5000	5000	5000	5000	5000	5000	5000	5000	5000	5000	5000
ALARM	-	-	-	-	-	-	-	-	-	H	-	-	-	L	-	-	H
NOTES	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N

This Attribute Control chart is a combination of the u-chart and the c-chart. The chart normalizes the defect rate, expressing it as defects per million. The chart displays the defect rate as defects per million. The table above gives the defect count in both absolute terms, and in the normalized defects per million used by the chart.

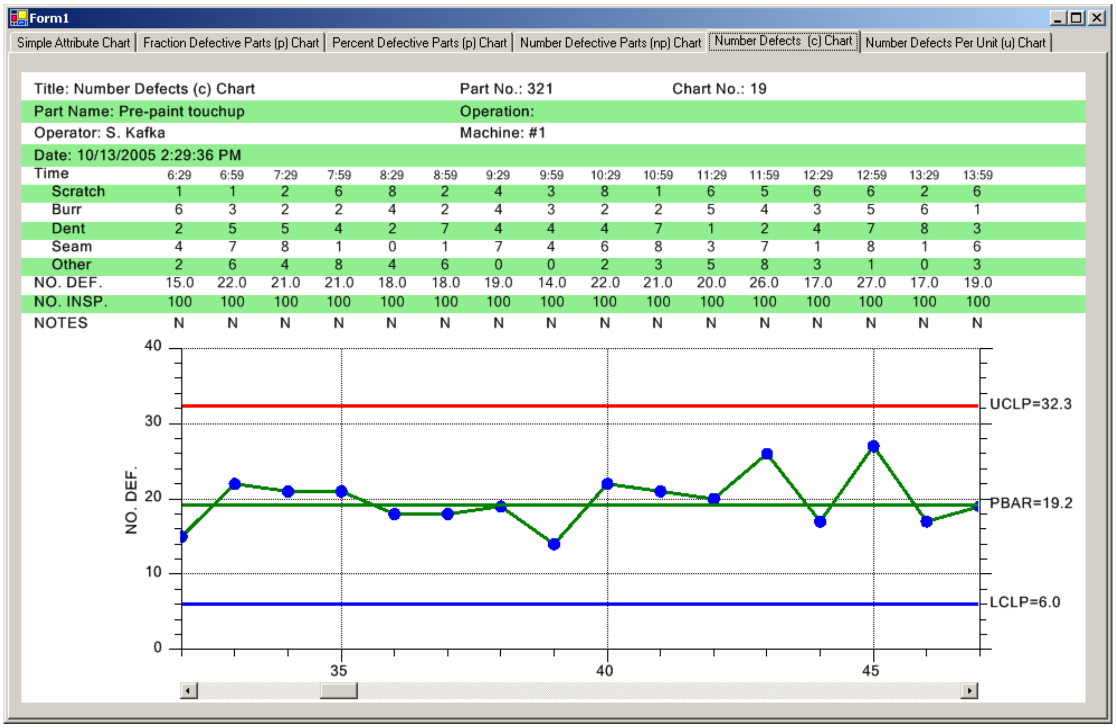
Defect and Defect Category Data Tables

As discussed under the *Variable Control Chart* section, standard worksheets used to gather and plot SPC data consist of three main parts.

- The first part is the header section, identifying the title of the chart, the monitored process, the machine operator, part number and other important information specific to the chart.
- The second part records the defect data, organized as a table recording the defect data and SPC calculations in a neat, readable fashion.
- The third part plots the calculated SPC values in the actual SPC chart.

The *Attribute Control Chart* templates that we have created have options that enable the programmer to customize and automatically include header information along with a table of the defect data, organized by defect category, number of defective parts, or total number of defects. Enable the scrollbar and you can display the tabular defect data and SPC plots for a window of 8-20 subgroups, from a much larger collection of measurement data representing hundreds or even thousands of subgroups, and use the scrollbar to move through the data, similar to paging through a spreadsheet.

Typical Number of Defects (c) Chart with data table


Form1
Simple Attribute Chart
Fraction Defective Parts (p) Chart
Percent Defective Parts (p) Chart
Number Defective Parts (np) Chart
Number Defects (c) Chart
Number Defects Per Unit (u) Chart
40
30
20
10
0
NO. DEF.
UCLP=32.3
PBAR=19.2
LCLP=6.0
35
40
45

Title: Number Defects (c) Chart Part No.: 321 Chart No.: 19
Part Name: Pre-paint touchup Operation:
Operator: S. Kafka Machine: #1
Date: 10/13/2005 2:29:36 PM

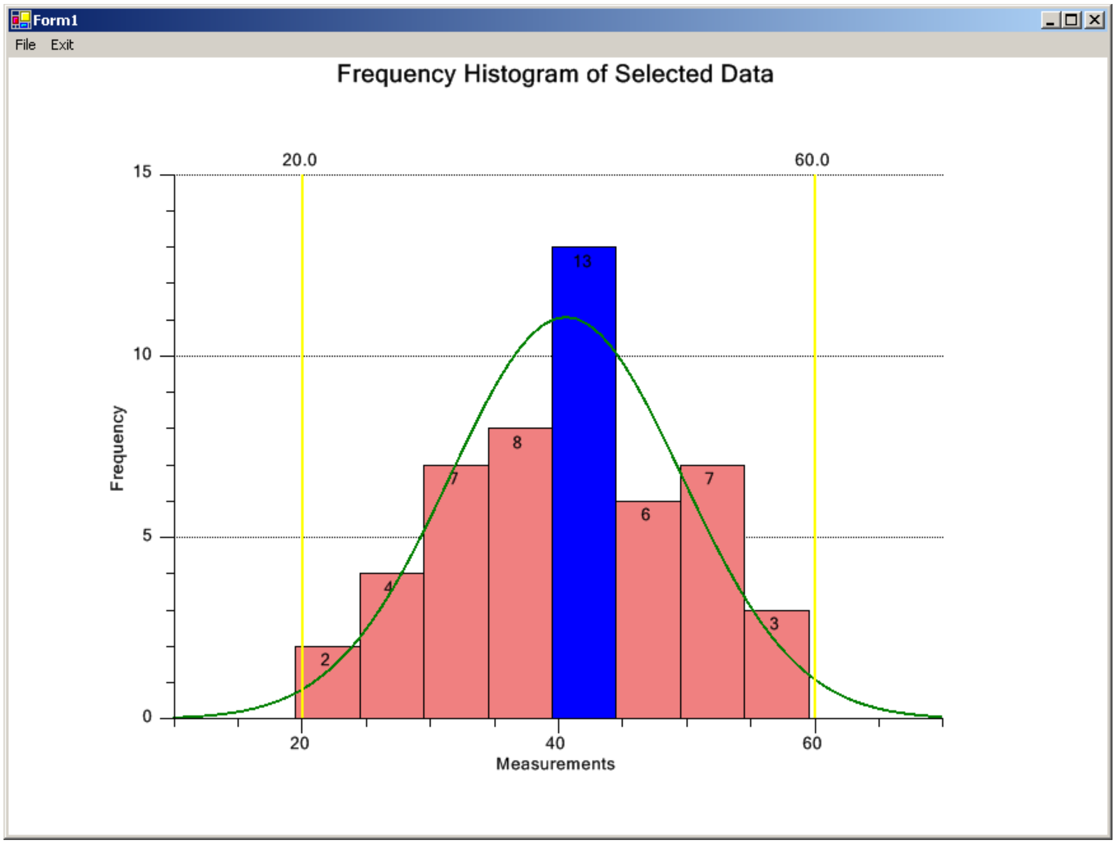
Time	6:29	6:59	7:29	7:59	8:29	8:59	9:29	9:59	10:29	10:59	11:29	11:59	12:29	12:59	13:29	13:59
Scratch	1	1	2	6	8	2	4	3	8	1	6	5	6	6	2	6
Burr	6	3	2	2	4	2	4	3	2	2	5	4	3	5	6	1
Dent	2	5	5	4	2	7	4	4	4	7	1	2	4	7	8	3
Seam	4	7	8	1	0	1	7	4	6	8	3	7	1	8	1	6
Other	2	6	4	8	4	6	0	0	2	3	5	8	3	1	0	3
NO. DEF.	15.0	22.0	21.0	21.0	18.0	18.0	19.0	14.0	22.0	21.0	20.0	26.0	17.0	27.0	17.0	19.0
NO. INSP.	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100
NOTES	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N

Other Important SPC Charts

Frequency Histogram Chart

An SPC control chart tacks the trend of critical variables in a production environment. It is important that the production engineer understand whether or not changes or variation in the critical variables are natural variations due to the tolerances inherent to the production machinery, or whether or not the variations are due to some systemic, assignable cause that needs to be addressed. If the changes in critical variables are the result of natural variations, a frequency histogram of the variations will usually follow one of the common continuous (normal, exponential, gamma, Weibull) or discrete (binomial, Poisson, hypergeometric) distributions. It is the job of the SPC engineer to know what distribution best models his process. Periodically plotting of the variation of critical variables will give SPC engineer important information about the current state of the process. A typical frequency histogram looks like:

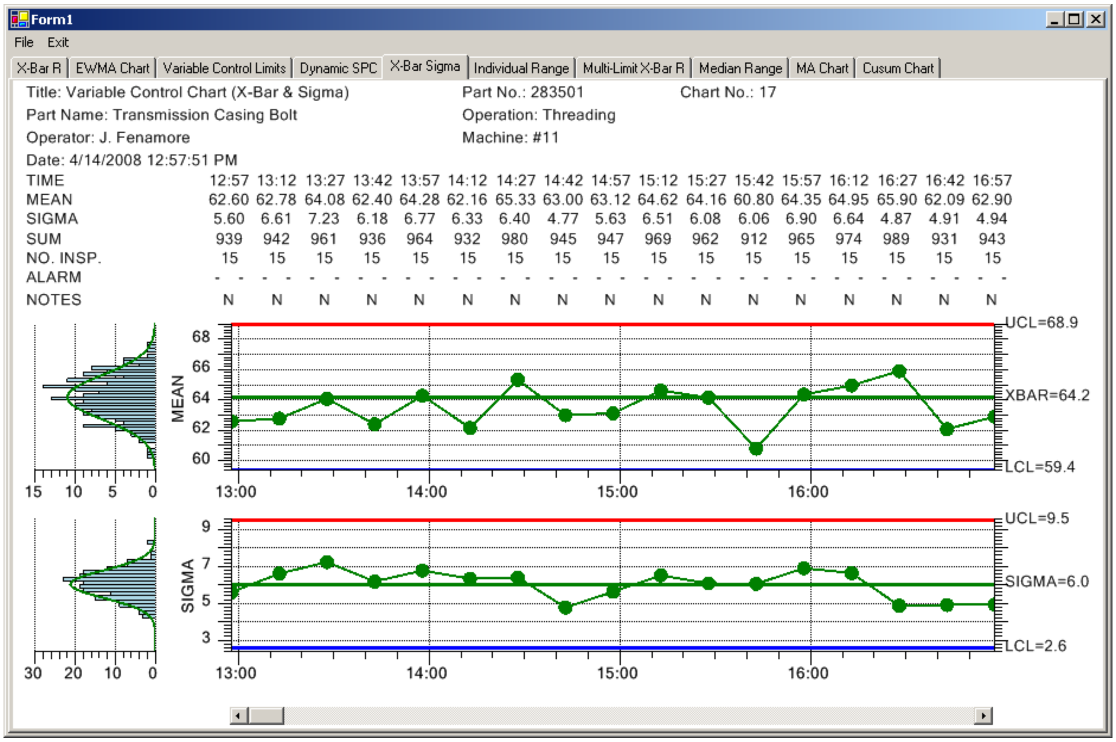
Frequency Histogram Chart



Viewing frequency histograms of both the variation in the primary variable (Mean, Median, count, or actual value), and the secondary variable (Range, Sigma or Moving

Range) side-by-side with the SPC control chart makes it even easier to find out whether the variations are the result of natural variations or the result of some systemic change in the process.

XBar-Sigma Chart with Integral Frequency Histograms



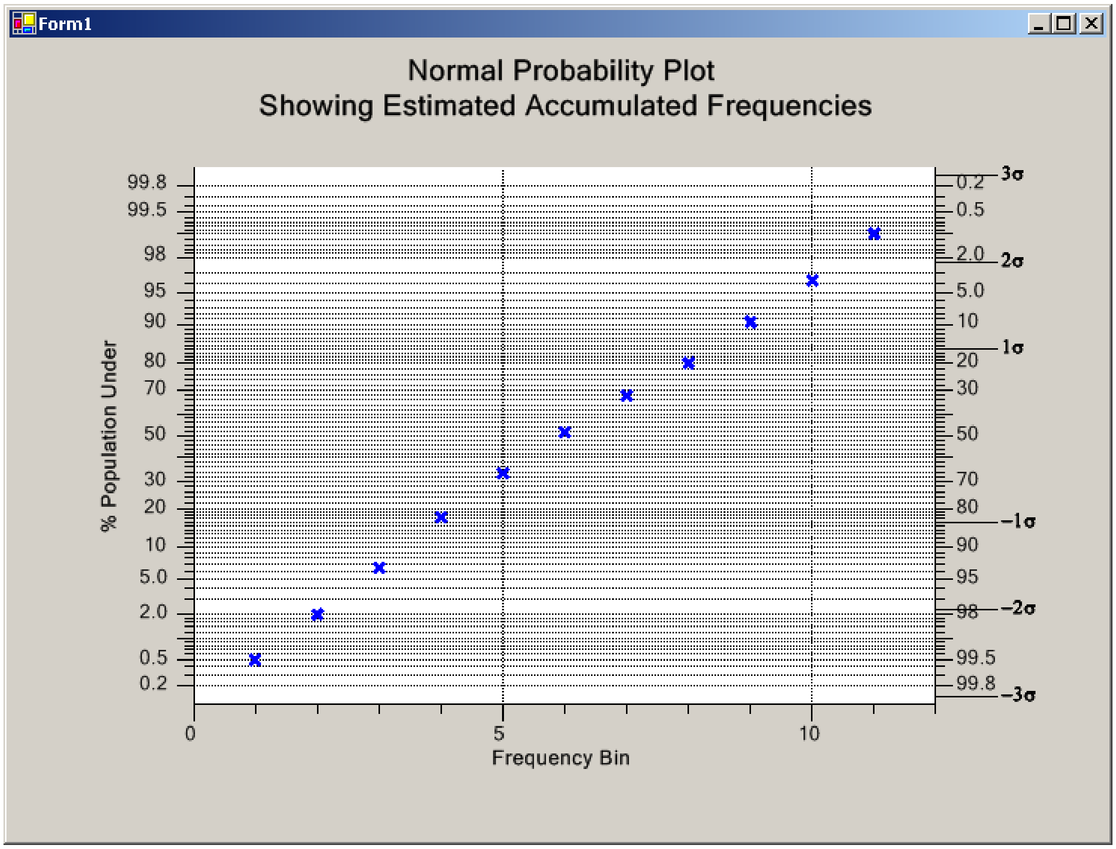
Title: Variable Control Chart (X-Bar & Sigma) Part No.: 283501 Chart No.: 17
Part Name: Transmission Casing Bolt Operation: Threading
Operator: J. Fenamore Machine: #11
Date: 4/14/2008 12:57:51 PM

TIME	12:57	13:12	13:27	13:42	13:57	14:12	14:27	14:42	14:57	15:12	15:27	15:42	15:57	16:12	16:27	16:42	16:57
MEAN	62.60	62.78	64.08	62.40	64.28	62.16	65.33	63.00	63.12	64.62	64.16	60.80	64.35	64.95	65.90	62.09	62.90
SIGMA	5.60	6.61	7.23	6.18	6.77	6.33	6.40	4.77	5.63	6.51	6.08	6.06	6.90	6.64	4.87	4.91	4.94
SUM	939	942	961	936	964	932	980	945	947	969	962	912	965	974	989	931	943
NO. INSP.	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15
ALARM	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -
NOTES	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N

Probability Plots

Another important tool the SPC engineer uses to model the process variation is the probability plot. The probability plot tests whether control chart measurements fit a normal distribution. Usually, the SPC engineer plots probability plot graphs by hand using special probability plot graph paper. We added probability scale and axis classes to the **QCSPCChart** software that plots probability plots directly on the computer. Control chart measurements that follow a normal distribution curve plot as a straight line when plotted in a normal probability plot.

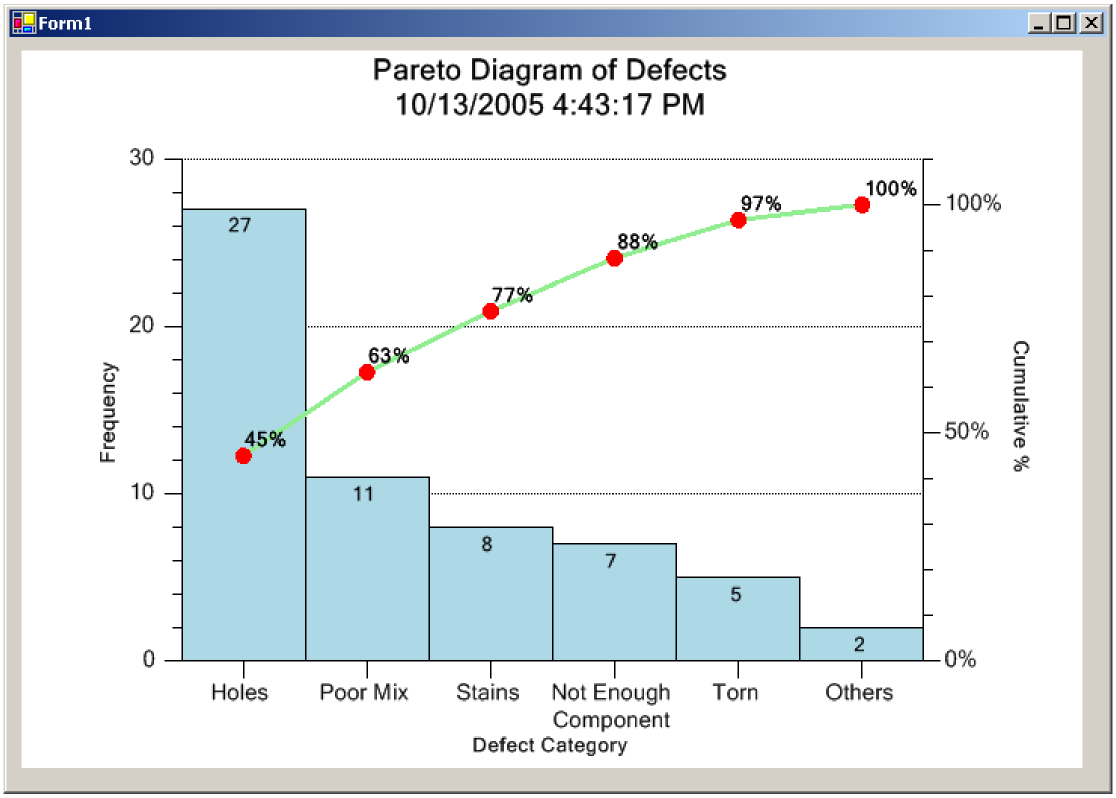
Cumulative Normal Probability Chart



Pareto Diagrams

The Pareto diagram is a special type of bar graph combined with a line plot, plotted in a chart that uses two different y-axis scales. The bar graph scale is a frequency scale that measures the number of times a specific problem has occurred. The line plot scale is a cumulative percentage scale.

Pareto Chart



The chart is easy to interpret. The tallest bar, the left-most one in a Pareto diagram, is the problem that has the most frequent occurrence. The shortest bar, the right-most one, is the problem that has the least frequent occurrence. Time spend on fixing the biggest problem will have the greatest affect on the overall problem rate. This is a simplistic view of actual Pareto analysis, which would usually take into account the cost effectiveness of fixing a specific problem. Never less, it is powerful communication tool that the SPC engineer can use in trying to identify and solve production problems.

3. Class Architecture of the SPC Control Chart Tools for *.Net* Class Library

Major Design Considerations

This chapter presents an overview of the **SPC Control Chart Tools for *.Net*** class architecture. It discusses the major design considerations of the architecture:

Major design consideration specific to **SPC Control Chart Tools for *.Net*** are:

- Direct support for the following SPC chart types:

  Variable Control Charts

  - Fixed sample size subgroup control charts
    - X-Bar R – (Mean and Range) chart
    - X-Bar Sigma (Mean and Sigma) chart
    - Median and Range (Median and Range) chart
    - X-R (Individual Range Chart) chart
    - EWMA (Exponentially Weighted Moving Average Chart)
    - MA (Moving Average Chart)
    - MAMR (Moving Average/Moving Range)
    - MAMS (Moving Average/Moving Sigma)
    - CuSum (Tabular Cumulative Sum Chart)
    - Levey-Jennings
  - Variable sample size subgroup control charts
    - X-Bar Sigma (Mean and Sigma Chart)

  Attribute Control Charts

  - Fixed sample size subgroup control charts
    - p-Chart (Fraction or Percent of Defective Parts, Fraction or Percent Non-Conforming)
    - np Chart (Number of Defective Parts, Number of Non-Conforming)
    - c-Chart (Number of Defects, Number of Non-Conformities)
    - u-Chart (Number of Defects per Unit, Number of Non-Conformities Per Unit)
    - DPMO (Number of Defects per Million)
  - Variable sample size subgroup control charts
    - p Chart (Fraction or Percent of Defective Parts)
    - u-Chart (Number of Defects per Unit)

SPC Analysis Charts

Frequency Histograms
Probability Charts
Pareto Charts

- Minimal programming required – create SPC charts with a few lines of code using our SPC chart templates.

- Integrated frequency histograms support – Display frequency histograms of sampled data, displayed side-by-side, sharing the same y-axis, with the SPC chart.

- Charts Header Information – Customize the chart display with job specific information, for example: Title, Operator, Part Number, Specification Limits, Machine, ect.

- Table display of SPC data – Display the sampled and calculated values for a SPC chart in a table, directly above the associated point in the SPC chart, similar to standardized SPC worksheets.

- Automatic calculation of SPC control limits – Automatically calculate SPC control limits using sampled data, using industry standard SPC control limit algorithms unique to each chart type.

- Automatic y-Axis scaling – Automatically calculated the y-axis scale for SPC charts, taking into account sampled and calculated data points, and any control limit lines added to the graph.

- Alarms – When monitored value exceeds a SPC control limit it can trigger an event that vectors to a user-written alarm processing delegate.

- SPC Process Capability Calculations -Variable Control Charts include Cp, Cpl, Cpu, Cpk, Cpm, Pp, Ppl, Ppu, and Ppk process capability statistics

- Notes – The operator can view or enter notes specific to a specific sample subgroup using a special notes tooltip.

- Data tooltips – The operator can view chart data values using a simple drill-down data tooltip display. The Data tooltips can optionally display sample subgroup data values and statistics, including process capability calculations (Cp, Cpl, Cpu, Cpk, Cpm, Pp, Ppl, Ppu, and Ppk) and customized using notes that have been entered for the sample subgroup.

- Data logging – SPC data (time stamp and/or batch number, sample values, calculated values, control limit values, and notes can be logged to disk in a CSV (commas separated value) file format.

- Scrollable view – Enable the scroll bar option and scroll through the chart and table view of the SPC data for an unlimited number of sample subgroups.

- Other, optional features – There are many optional features that SPC charts often use, including:

  Multiple SPC control limits, corresponding to +-1, 2 and 3 sigma limits.

  Scatter plots of all sampled data values on top of calculated means and medians.

  Data point annotations

The chapter also summarizes the classes in the **SPC Control Chart Tools for .*Net*** library.

SPC Control Chart Tools for .*Net* Class Summary

The **SPC Control Chart Tools for .*Net*** library is a super set of the **QCChart2D** library. The classes of the **QCChart2D** library are an integral part of the software. A summary of the **QCChart2D** classes appears below.

QCChart2D Class Summary

Chart view class The chart view class is a UserControl subclass that manages the graph objects placed in the graph

Data classes There are data classes for simple xy and group data types. There are also data classes that handle System.DateTime date/time data and contour data.

Scale transform classes The scale transform classes handle the conversion of physical coordinate values to working coordinate values for a single dimension.

Coordinate transform classes

The coordinate transform classes handle the conversion of physical coordinate values to working coordinate values for a parametric (2D) coordinate system.

Attribute class The attribute class encapsulates the most common attributes (line color, fill color, line style, line thickness, etc.) for a chart object.

Auto-Scale classes The coordinate transform classes use the auto-scale classes to establish the minimum and maximum values used to scale a 2D coordinate system. The axis classes also use the auto-scale classes to establish proper tick mark spacing values.

Charting object classes The chart object classes includes all objects placeable in a chart. That includes axes, axes labels, plot objects (line plots, bar graphs, scatter plots, etc.), grids, titles, backgrounds, images and arbitrary shapes.

Mouse interaction classes These classes, directly and indirectly System.EventHandler delegates that trap mouse events and permit the user to create and move data cursors, move plot objects, display tooltips and select data points in all types of graphs.

File and printer rendering These classes render the chart image to a printer, to a variety of file formats including JPEG, and BMP, or to a .Net Image object.

Miscellaneous utility classes Other classes use these for data storage, file I/O, and data processing.

For each of these categories see the associated description in the **QCChart2D** manual (QCChart2DNetManual.pdf). The **SPC Control Chart Tools for *.Net*** classes are in addition to the ones above. They are summarized below.

SPC Control Chart Tools for .Net Class Hierarchy

The **QCSPCChart** classes are a super set of the **QCChart2D** charting software. No attempt should be made to utilize the **QCSPCChart** classes without a good understanding of the **QCChart2D** classes. See the **QCChart2DNetManual** PDF file for detailed information about the **QCChart2D** classes. The diagram below depicts the class hierarchy of the **SPC Control Chart Tools for *.Net*** library without the additional **QCChart2D** classes

Namespace com.quinn-curtis.spcchartnet6.

```
com.quinn-curtis.chart2dnet6.ChartView
        FrequencyHistogramChart
        ParetoChart
        ProbabilityChart
        SPCChartBase
                SPCEventAttributeControlChar
                        SPCBatchAttributeControlChart
                        SPCTimeAttributeControlChart
                SPCTimeVariableControlChart
                        SPCBatchAttributeControlChart
                        SPCTimeAttributeControlChart
```

com.quinn-curtis.chart2dnet6.AutoScale
    ProbabilityAutoScale
com.quinn-curtis.chart2dnet6.Axis
    ProbabilityAxis
com.quinn-curtis.chart2dnet6.LinearAxis
    ProbabilitySigmaAxis
com.quinn-curtis.chart2dnet6.PhysicalCoordinates
    ProbabilityCoordinates
com.quinn-curtis.chart2dnet6.Scale
    ProbabilityScale
com.quinn-curtis.chart2dnet6.StringLabel
    NotesLabel
com.quinn-curtis.chart2dnet6.MouseListener
    NotesToolTip
com.quinn-curtis.chart2dnet6.DataToolTip
    SPCDataToolTip

QCSPCChart Classes

SPCControlChartData
SPCControlLimitAlarmArgs
SPCControlLimitRecord
SPCCalculatedValueRecrod
SPCProcessCapabilityRecord
SPCSampledValueRecord
SPCControlParameters
SPCGeneralizedTableDisplay
SPCControlPlotObjects
SPCChartObjects

SPC Control Chart Data

SPCControlChartData

SPC control chart data is stored in the **SPCControlChartData** class. It holds the header information used to customize the chart table, the raw sample data used to prepare the chart, the calculated chart values used in the chart, and the SPC control limits. It contains array lists of **SPCSampledValueRecord**,

SPCControlLimitRecord and **SPCCalculatedValueRecord** objects.

SPCSampledValueRecord

This class encapsulates a sample data value. It includes a description for the item, the current value of the sampled value, and a history of previous values.

SPCControlLimitRecord

This class holds information specific to a SPC control limit: including the current value of the control limit, a history of control limit values, description, and the hysteresis value for alarm checking.

SPCCalculatedValueRecord

The record class for a calculated SPC statistic. It holds the calculated value type (mean, median, sum, variance, standard deviation, etc.), value, description and historical data.

SPCProcessCapabilityRecord

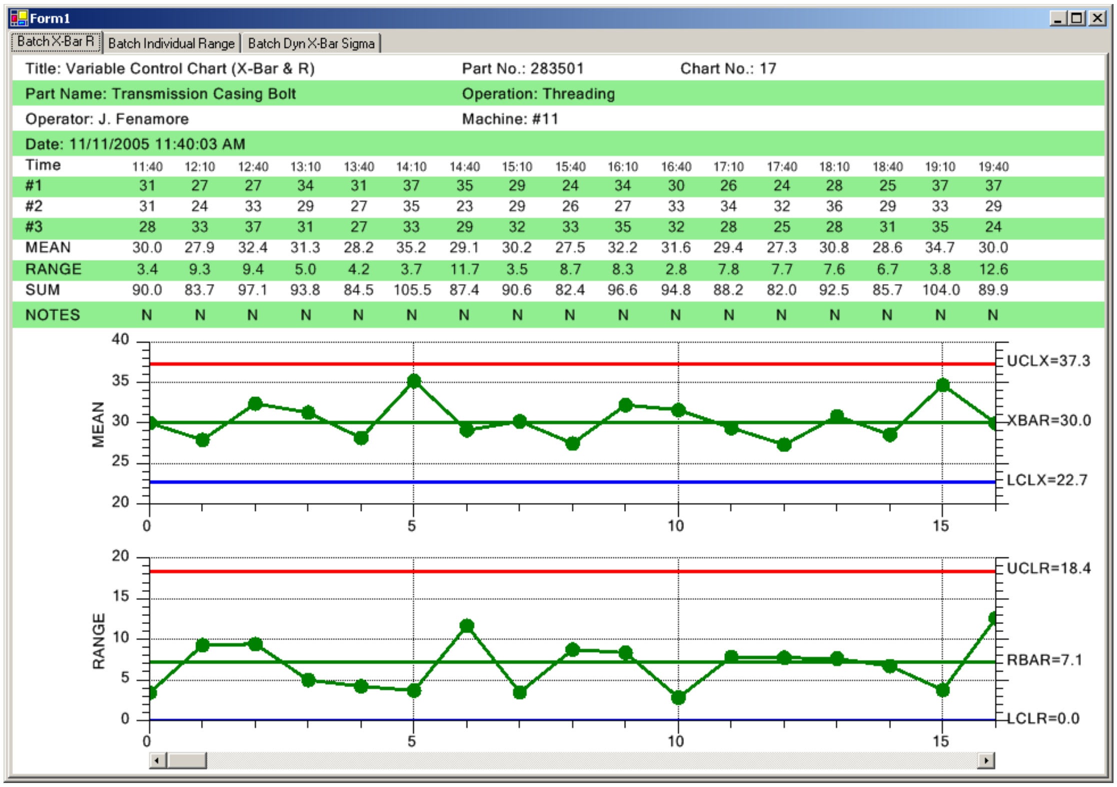
The record class for storing and calculating process capability statistics: Cp, Cpl, Cpu, Cpk, Cpm, Pp, Ppl, Ppu and Ppk.

SPC Charts and Related Chart Objects

SPCChartBase

The **SPCChartBase** forms the base object for all SPC control charts. The variable control chart templates (**SPCBatchVariableControlChart, and SPCTimeVariableControlChart**) are derived from the **SPCEventVariableControlChart** class, which in turn is derived from **SPCChartBase**. The attribute control charts (**SPCBatchAttributeControlChart** and **SPCTimeAttributeControlChart**) are derived from the **SPCEventAttributeControlChart** class, which in turn is derived from SPCChartBase.

Typical Batch Variable Control Chart (Mean and Range or X-Bar R)



Title: Variable Control Chart (X-Bar & R) Part No.: 283501 Chart No.: 17

Part Name: Transmission Casing Bolt Operation: Threading

Operator: J. Fenamore Machine: #11

Date: 11/11/2005 11:40:03 AM

Time	11:40	12:10	12:40	13:10	13:40	14:10	14:40	15:10	15:40	16:10	16:40	17:10	17:40	18:10	18:40	19:10	19:40
#1	31	27	27	34	31	37	35	29	24	34	30	26	24	28	25	37	37
#2	31	24	33	29	27	35	23	29	26	27	33	34	32	36	29	33	29
#3	28	33	37	31	27	33	29	32	33	35	32	28	25	28	31	35	24
MEAN	30.0	27.9	32.4	31.3	28.2	35.2	29.1	30.2	27.5	32.2	31.6	29.4	27.3	30.8	28.6	34.7	30.0
RANGE	3.4	9.3	9.4	5.0	4.2	3.7	11.7	3.5	8.7	8.3	2.8	7.8	7.7	7.6	6.7	3.8	12.6
SUM	90.0	83.7	97.1	93.8	84.5	105.5	87.4	90.6	82.4	96.6	94.8	88.2	82.0	92.5	85.7	104.0	89.9
NOTES	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N

SPCEventVariableControlChart

A Variable Control Chart class that uses an EventCoordinate system with an event based X-Axis. This class creates MEAN_RANGE_CHART, MEDIAN_RANGE_CHART, INDIVIDUAL_RANGE_CHART, MEAN_SIGMA_CHART, MEAN_SIGMA_CHART_VSS, EWMA_CHART, TABCUSUM, MA_CHART, MAMR_CHART and MAMS_CHART chart types.

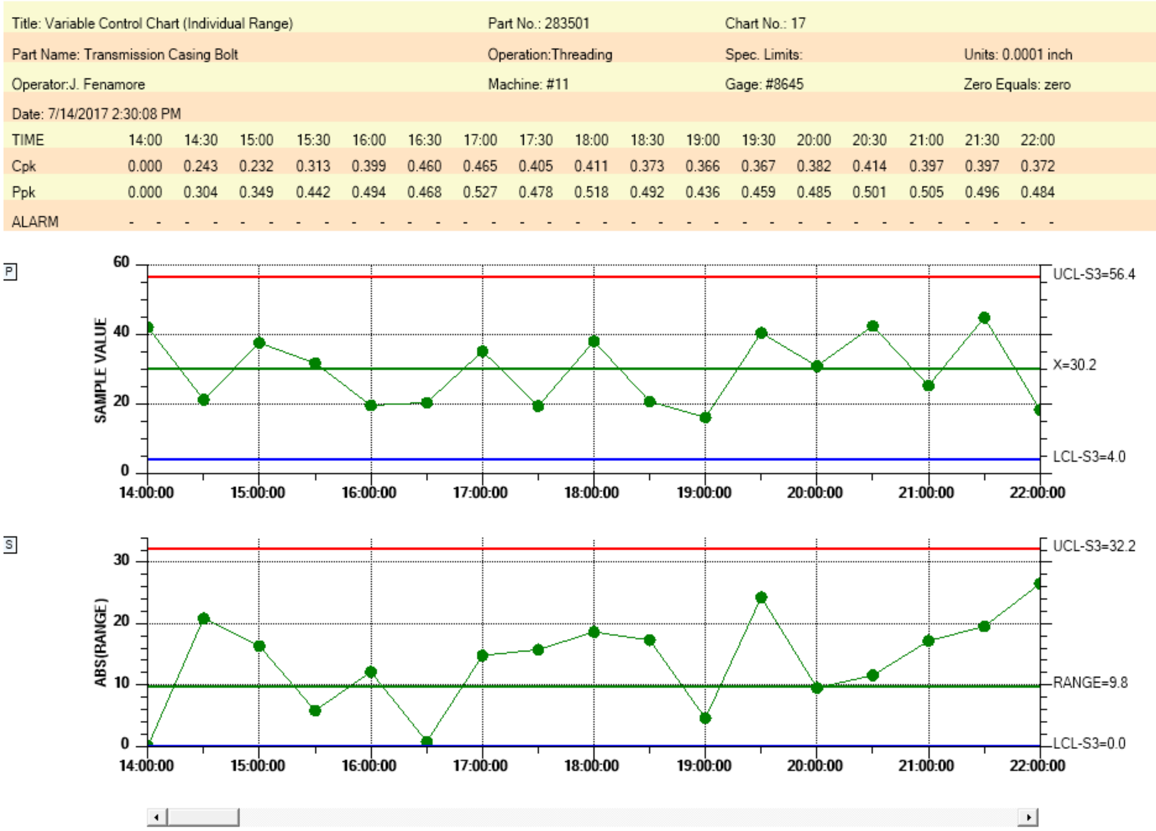
SPCBatchVariableControlChart

A Batch Variable Control Chart class that uses a CartesianCoordinate system with a numeric based X-Axis. This class creates MEAN_RANGE_CHART, MEDIAN_RANGE_CHART, INDIVIDUAL_RANGE_CHART, MEAN_SIGMA_CHART, MEAN_SIGMA_CHART_VSS, EWMA_CHART, TABCUSUM, MA_CHART, MAMR_CHART, LEVEY_JENNINGS_CHART and MAMS_CHART chart types.

SPCTimeVariableControlChart

A Variable Control Chart class that uses a TimeCoordinate system with a time based X-Axis. This class creates MEAN_RANGE_CHART, MEDIAN_RANGE_CHART, INDIVIDUAL_RANGE_CHART, MEAN_SIGMA_CHART, MEAN_SIGMA_CHART_VSS, EWMA_CHART, TABCUSUM, MA_CHART, MAMR_CHART and MAMS_CHART chart types.

Typical SPCBatchCusumControlChart with alarm limits



Title: Variable Control Chart (Individual Range) Part No.: 283501 Chart No.: 17

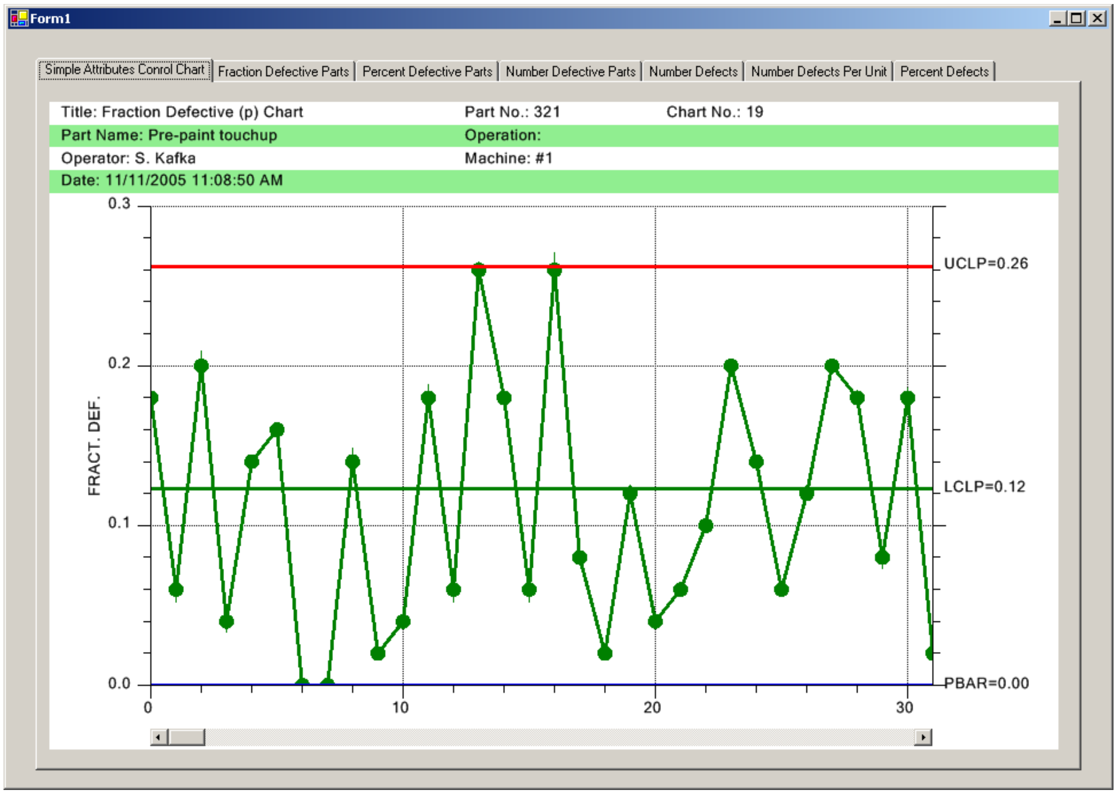
Part Name: Transmission Casing Bolt Operation:Threading Spec. Limits: Units: 0.0001 inch

Operator:J. Fenamore Machine: #11 Gage: #8645 Zero Equals: zero

Date: 7/14/2017 2:30:08 PM

TIME	14:00	14:30	15:00	15:30	16:00	16:30	17:00	17:30	18:00	18:30	19:00	19:30	20:00	20:30	21:00	21:30	22:00
Cpk	0.000	0.243	0.232	0.313	0.399	0.460	0.465	0.405	0.411	0.373	0.366	0.367	0.382	0.414	0.397	0.397	0.372
Ppk	0.000	0.304	0.349	0.442	0.494	0.468	0.527	0.478	0.518	0.492	0.436	0.459	0.485	0.501	0.505	0.496	0.484
ALARM	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -

Typical Batch Attribute Control Chart (Fraction Defective or p-Chart)



Title: Fraction Defective (p) Chart
Part No.: 321
Chart No.: 19
Part Name: Pre-paint touchup
Operation:
Operator: S. Kafka
Machine: #1
Date: 11/11/2005 11:08:50 AM

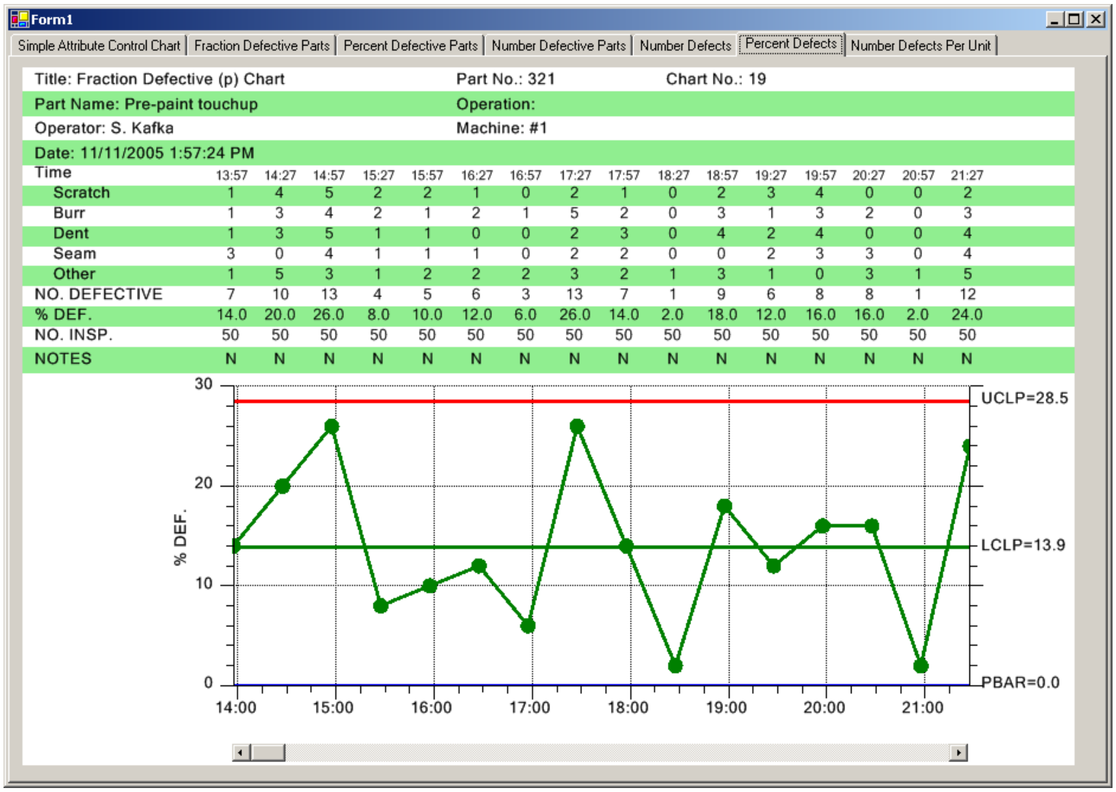
SPCEventAttributeControlChart

An Attribute Control Chart class that uses an EventCoordinates system with an Event X-Axis. This class creates PERCENT_DEFECTIVE_PARTS_CHART, FRACTION_DEFECTIVE_PARTS_CHART, NUMBER_DEFECTIVE_PARTS_CHART, NUMBER_DEFECTS_PERUNIT_CHART, NUMBER_DEFECTS_CHART SPC, NUMBER_DEFECTS_PER_MILLION_CHART, PERCENT_DEFECTIVE_PARTS_CHART_VSS, FRACTION_DEFECTIVE_PARTS_CHART_VSS, NUMBER_DEFECTS_PERUNIT_CHART_VSS chart types

SPCBatchAttributeControlChart

A Batch Attribute Control Chart class that uses a CartesianCoordinate system with a numeric X-Axis. This class creates PERCENT_DEFECTIVE_PARTS_CHART, FRACTION_DEFECTIVE_PARTS_CHART, NUMBER_DEFECTIVE_PARTS_CHART, NUMBER_DEFECTS_PERUNIT_CHART, NUMBER_DEFECTS_CHART SPC, NUMBER_DEFECTS_PER_MILLION_CHART, PERCENT_DEFECTIVE_PARTS_CHART_VSS, FRACTION_DEFECTIVE_PARTS_CHART_VSS, NUMBER_DEFECTS_PERUNIT_CHART_VSS chart types.

Typical Time Attribute Control Chart (Fraction Defective or p-Chart)



Title: Fraction Defective (p) Chart　Part No.: 321　Chart No.: 19

Part Name: Pre-paint touchup　Operation:

Operator: S. Kafka　Machine: #1

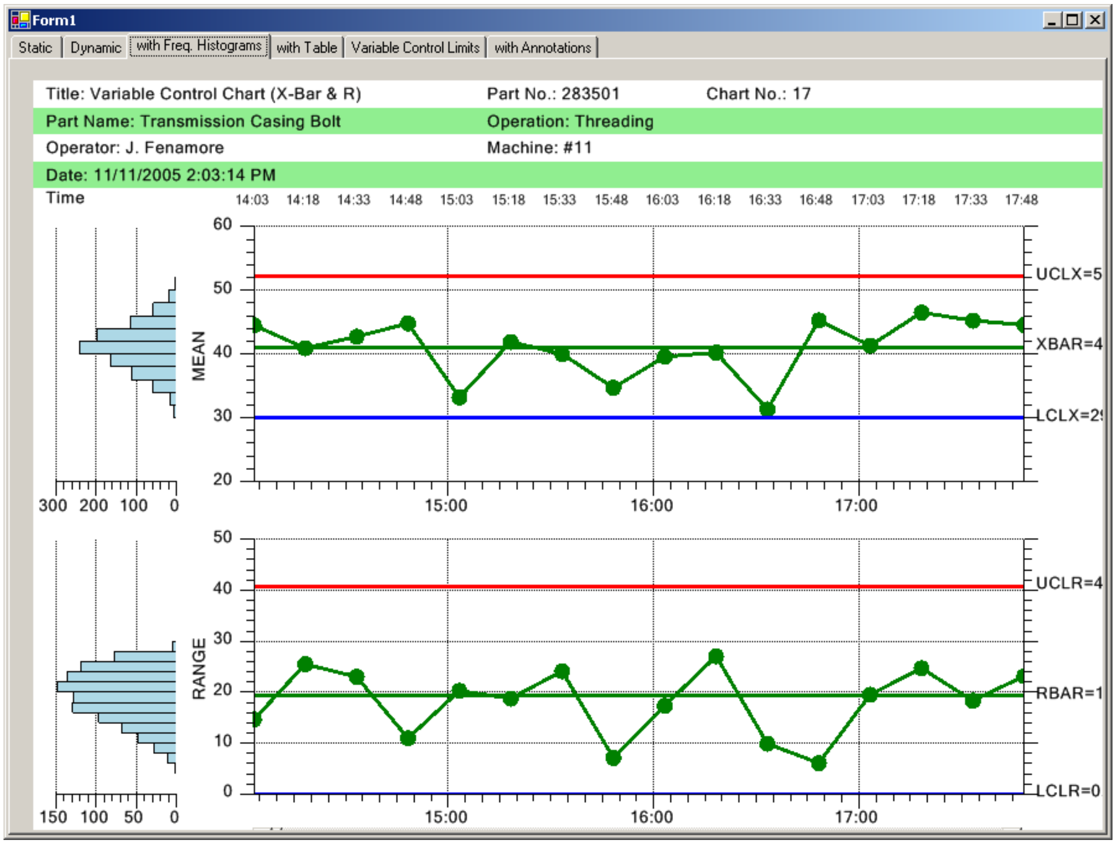
Date: 11/11/2005 1:57:24 PM

Time	13:57	14:27	14:57	15:27	15:57	16:27	16:57	17:27	17:57	18:27	18:57	19:27	19:57	20:27	20:57	21:27
Scratch	1	4	5	2	2	1	0	2	1	0	2	3	4	0	0	2
Burr	1	3	4	2	1	2	1	5	2	0	3	1	3	2	0	3
Dent	1	3	5	1	1	0	0	2	3	0	4	2	4	0	0	4
Seam	3	0	4	1	1	1	0	2	2	0	0	2	3	3	0	4
Other	1	5	3	1	2	2	2	3	2	1	3	1	0	3	1	5
NO. DEFECTIVE	7	10	13	4	5	6	3	13	7	1	9	6	8	8	1	12
% DEF.	14.0	20.0	26.0	8.0	10.0	12.0	6.0	26.0	14.0	2.0	18.0	12.0	16.0	16.0	2.0	24.0
NO. INSP.	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
NOTES	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N

SPCTimeAttributeControlChart

An Attribute Control Chart class that uses a TimeCoordinate system with a time based X-Axis This class creates PERCENT_DEFECTIVE_PARTS_CHART, FRACTION_DEFECTIVE_PARTS_CHART, NUMBER_DEFECTIVE_PARTS_CHART, NUMBER_DEFECTS_PERUNIT_CHART, NUMBER_DEFECTS_CHART, NUMBER_DEFECTS_PER_MILLION_CHART, PERCENT_DEFECTIVE_PARTS_CHART_VSS, FRACTION_DEFECTIVE_PARTS_CHART_VSS, NUMBER_DEFECTS_PERUNIT_CHART_VSS chart types.

Frequency Histograms used in Combination with a SPC Control Chart



Title: Variable Control Chart (X-Bar & R)   Part No.: 283501   Chart No.: 17

Part Name: Transmission Casing Bolt   Operation: Threading

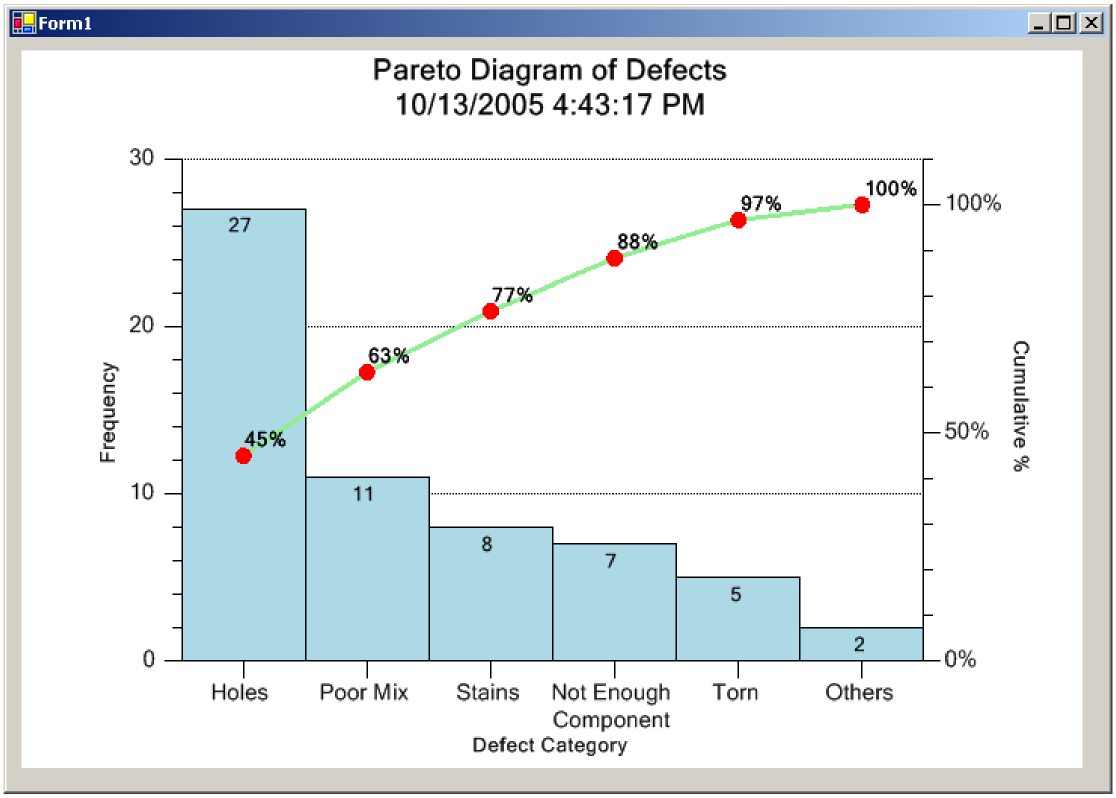
Operator: J. Fenamore   Machine: #11

Date: 11/11/2005 2:03:14 PM

FrequencyHistogramChart

A *Frequency Histogram* checks that the variation in a process variable follows the predicted distribution function (normal, Poisson, chi-squared, etc). The class includes all of the objects needed to draw a complete frequency histogram chart. These objects include objects for data, a coordinate system, titles, axes, axes labels, grids and a bar plot.

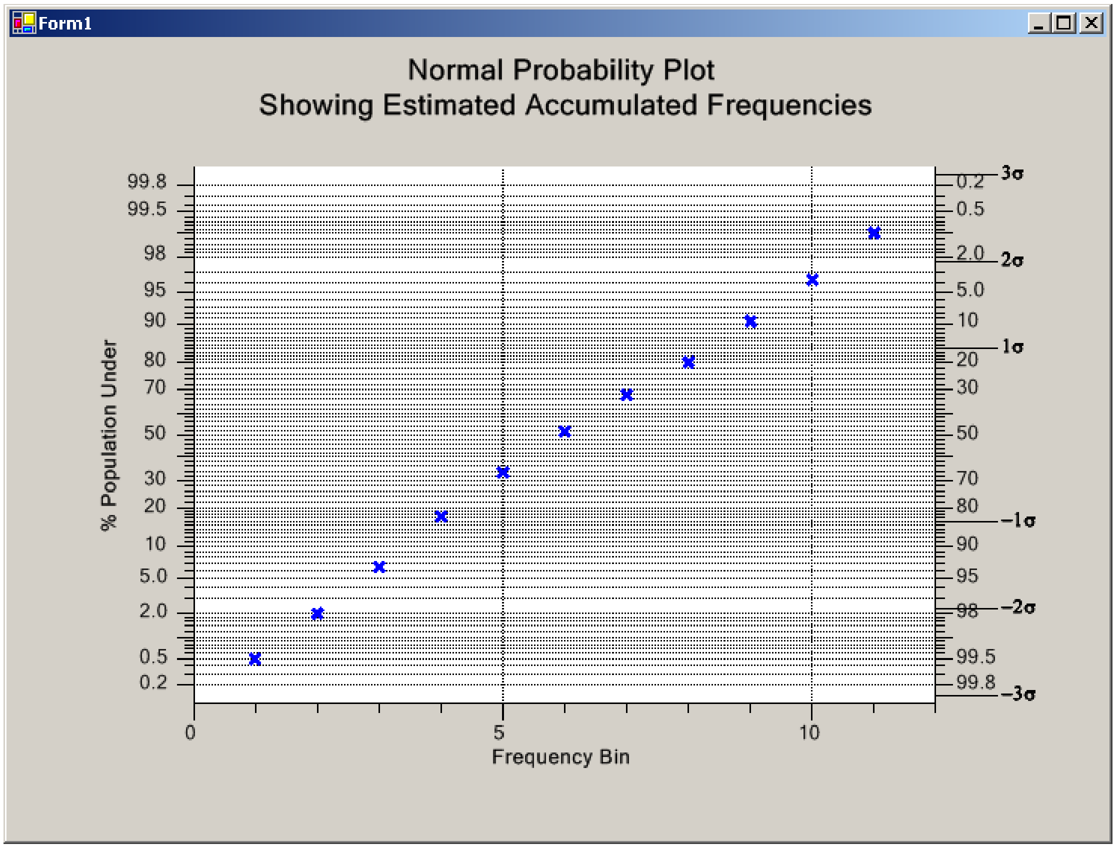
Pareto Chart



ParetoChart

The *Pareto Diagram* is special type of bar graph combined with a line plot, plotted in a chart that uses two different y-axis scales. The bar graph scale is a frequency scale that measures the number of times a specific problem has occurred. The line plot scale is a cumulative percentage scale. The class includes all of the objects needed to draw a complete Pareto chart. These objects include objects for data, coordinate systems, titles, axes, axes labels, grids, numeric labels, and a line plot and bar plot.

Cumulative Normal Probability Chart



Probability Plots The **ProbabilityChart** class is a highly specialized chart template used to plot cumulative frequency data using a coordinate system that has a cumulative probability y-scale. The class includes all of the objects needed to draw a complete Probability chart. These objects include objects for data, coordinate systems, titles, axes, axes labels, grids, numeric labels, and scatter plot. New classes were developed for the **QCChart2D** charting software capable of rendering of probability chart coordinate systems (**ProbabilityScale**, **ProbabilityAutoScale**, **ProbabilityCoordinates**) and probability axes (**ProbabilityAxis**, **ProbabilitySigmaAxis**).

ProbabilityScale The **ProbabilityScale** class implements a cumulative normal probability coordinate system for a single coordinate, x or y. Two such scales provide the scaling routines for x and y in an **PhysicalCoordindates** derived class, **CartesianCoordinates**, for example. This allows for different x and y scale types (linear, cumulative normal probability, time) to be installed independently for x- and y-coordinates.

ProbabilityAutoScale

The **ProbabilityAutoScale** class is used with cumulative normal probability coordinates and auto-scales the plotting area of graphs and to set the minimum and maximum values of the axes displayed in the graphs.

ProbabilityCoordinates

The **ProbabilityCoordinates** class extends the **PhysicalCoordinates** class to support a y-axis probability scale in an xy coordinate plane.

ProbabilityAxis The **ProbabilityAxis** class implements a probability axis where the major tick marks are placed at intervals appropriate to a cumulative probability scale.

ProbabilitySigmaAxis

The **ProbabilitySigmaAxis** class implements a linear axis where the tick marks are placed at linear intervals on the sigma levels of the associated probability scale.

NotesLabel The **NotesLabel** class displays the Notes items in the SPC table.

NotesToolTip The **NotesToolTip** displays the Notes tooltip for the notes items in the SPC table.

SPCDataToolTip The **SPCDataTooTip** displays the data tooltip for SPC Charts..

SPC Calculations

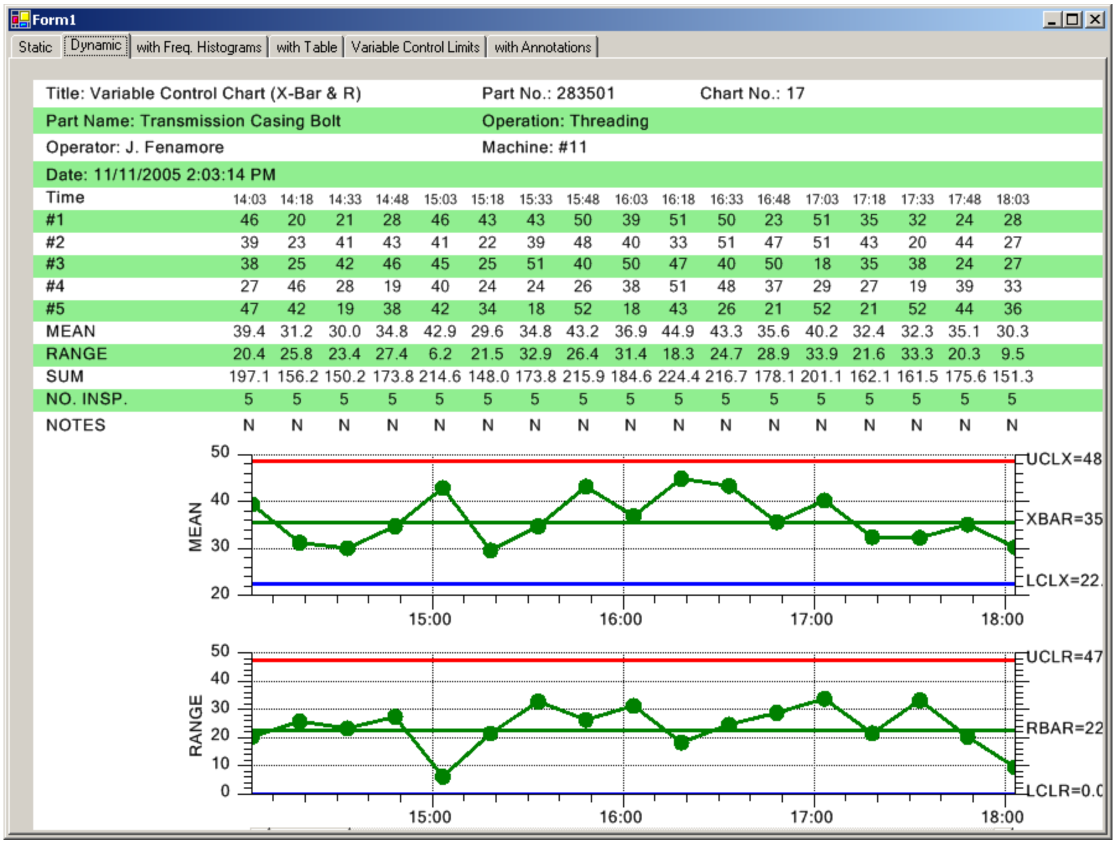
SPCArrayStatistics SPC involves many statistical calculations. The **SPCArrayStatistics** class includes routines for the calculation of sums, means, medians, ranges, minimum values, maximum values, variances, and standard deviations. It also includes routines for array sorting and calculating frequency bins for frequency histograms. It also includes functions that compute cumulative probability values for normal, Poisson, and chi-squared distributions.

SPCControlParameters

The **SPCControlParameters** class contains the factors and formulas for calculating SPC control chart limits for *Variable* and *Attribute Control Charts*. It includes calculations for the most common SPC charts: X-Bar R, Median and Range, X-Bar Sigma, X-R, u-chart, p-chart, np-chart, and c-chart.

Tabular Display

Table Display of Sampled and Calculated SPC Control Chart Value



Title: Variable Control Chart (X-Bar & R) Part No.: 283501 Chart No.: 17

Part Name: Transmission Casing Bolt Operation: Threading

Operator: J. Fenamore Machine: #11

Date: 11/11/2005 2:03:14 PM

Time	14:03	14:18	14:33	14:48	15:03	15:18	15:33	15:48	16:03	16:18	16:33	16:48	17:03	17:18	17:33	17:48	18:03
#1	46	20	21	28	46	43	43	50	39	51	50	23	51	35	32	24	28
#2	39	23	41	43	41	22	39	48	40	33	51	47	51	43	20	44	27
#3	38	25	42	46	45	25	51	40	50	47	40	50	18	35	38	24	27
#4	27	46	28	19	40	24	24	26	38	51	48	37	29	27	19	39	33
#5	47	42	19	38	42	34	18	52	18	43	26	21	52	21	52	44	36
MEAN	39.4	31.2	30.0	34.8	42.9	29.6	34.8	43.2	36.9	44.9	43.3	35.6	40.2	32.4	32.3	35.1	30.3
RANGE	20.4	25.8	23.4	27.4	6.2	21.5	32.9	26.4	31.4	18.3	24.7	28.9	33.9	21.6	33.3	20.3	9.5
SUM	197.1	156.2	150.2	173.8	214.6	148.0	173.8	215.9	184.6	224.4	216.7	178.1	201.1	162.1	161.5	175.6	151.3
NO. INSP.	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
NOTES	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N

SPCGeneralizedTableDisplay

The **SPCGeneralizedTableDisplay** manages a list of **ChartText** objects (**NumericLabel**, **StringLabel** and **TimeLabel** objects), that encapsulate each unique table entry in the SPC chart table. This class also manages the spacing between the rows and columns of the table, and the alternating stripe used as a background for the table.

SPC Control Alarms

SPCControlLimitAlarmArgs

This class passes event information to a **SPCControlLimitAlarmEventDelegate** alarm processing delegate.

SPCControlLimitAlarmEventDelegate

A delegate type for hooking up control limit alarm notifications

4. Programming Rev. 3.0 Features

Chapter 4 was originally a summary of the information in the **QCChart2DNetManual** PDF file. Refer to that manual for information about the QCChart2D classes that underlie the QCSPCChart software..

This chapter assumes that you are already familiar with QCSPCChart, and would like to take advantage of the new features added to Rev. 3.0. If you are new to QCSPCChart, jump to Chapter 5 and come back to this chapter later.

- **1. Charts have been converted to always use Event coordinates**
- **2. Zooming as an option for the scrollbar (UI option)**
- **3. Collapsible Items (UI option)**
- **4. Enhanced Annotations (Programming option)**
- **5. Enhanced Out of Limit Symbol (Programming option)**
- **6. Variable Spec and Control Limits (Programming option)**
- **7. Marking a sample internal as bad (Programming option)**
- **8. Reset N of M counters for control moves (Programming option)**
- **9. Remove Negative LCL control limits for Variable Control Secondary charts, or Attribute Control Primary charts (Programming option)**
- **10. N of M testing when the most recent point entering the test is within limits (Programming option)**
- **11. Sample Interval Updates with an invalid number of samples (Programming option)**
- **12. Alarm Forcing (Programming Option)**
- **13. A Levey-Jennings Variable Control Chart was added shortly after Rev. 2.3 was released**
- **14. Batch update with auto-calculated control limits.**

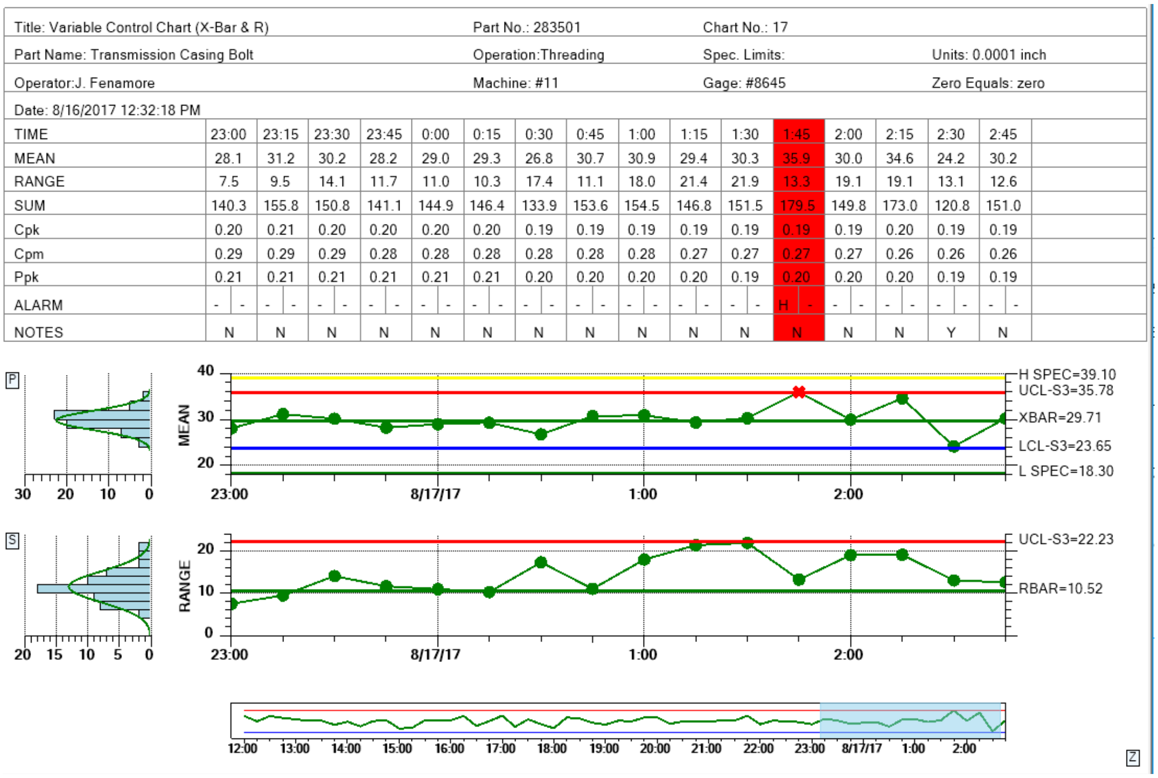
Charts have been converted to always use Event coordinates

Originally, the time-based SPC charts used a Time/Date x-scale in the coordinate system, and the batch-based SPC charts used a linear x-scale in the coordinate system. Both of these have been replaced with a coordinate system which uses an Event-based x-scale. In order to maintain backward compatibility, we also keep the old SPCTime... and SPCBatch... control chart classes, but derive them from the new Event-based SPC chart classes. You may see a few differences between the old and new time-based axis implementations. In the new version, no matter what the time stamp is on a SPCTime... SPC chart, adjacent points will always be equally spaced. So if your sample interval is irregular, or you even skip days or weeks in your sampling, the resulting chart will still display equally spaced adjacent sample records. Also, in the new version tick marks are placed on sample intervals. In the old version, the time axis tick marks were independent of the sample interval time stamps. In the new version, this can cause time axis labels to display the irregular time stamps of the sample interval time stamps. In the old version the tick mark time axis labels would be even values, because they did not use the sample interval time stamp values. You can read more about Event coordinates in the QCChart2D manual which is found in the same folder as QCSPCChart manual.

The new class heirarchy looks like:

```
ChartView
        SPCChartBase
                SPCEventVariableControlChart
                        SPCTimeVariableControlChart
                        SPCBatchVariableControlChart
                SPCEventAttributeControlChart
                        SPCTimeAttributeControlChart
                        SPCBatchAttributeControlChart
```

SPCEventVariableControlChart, SPCBatchVariableControlChart and SPCTimeVariableControlChart



Title: Variable Control Chart (X-Bar & R)	Part No.: 283501	Chart No.: 17	
Part Name: Transmission Casing Bolt	Operation:Threading	Spec. Limits:	Units: 0.0001 inch
Operator:J. Fenamore	Machine: #11	Gage: #8645	Zero Equals: zero
Date: 8/16/2017 12:32:18 PM			

TIME	23:00	23:15	23:30	23:45	0:00	0:15	0:30	0:45	1:00	1:15	1:30	1:45	2:00	2:15	2:30	2:45
MEAN	28.1	31.2	30.2	28.2	29.0	29.3	26.8	30.7	30.9	29.4	30.3	35.9	30.0	34.6	24.2	30.2
RANGE	7.5	9.5	14.1	11.7	11.0	10.3	17.4	11.1	18.0	21.4	21.9	13.3	19.1	19.1	13.1	12.6
SUM	140.3	155.8	150.8	141.1	144.9	146.4	133.9	153.6	154.5	146.8	151.5	179.5	149.8	173.0	120.8	151.0
Cpk	0.20	0.21	0.20	0.20	0.20	0.20	0.19	0.19	0.19	0.19	0.19	0.19	0.19	0.20	0.19	0.19
Cpm	0.29	0.29	0.29	0.28	0.28	0.28	0.28	0.28	0.28	0.27	0.27	0.27	0.27	0.26	0.26	0.26
Ppk	0.21	0.21	0.21	0.21	0.21	0.21	0.20	0.20	0.20	0.20	0.19	0.20	0.20	0.20	0.19	0.19
ALARM	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	H -	- -	- -	- -	- -
NOTES	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Y	N

The SPCTimeVariableControlChart now derives from SPCEventVariableControlChart.

Programs which originally used SPCTimeVariableControlChart can continue to do so. The beginning of your SPC chart class would look like:

```
/// <summary>
/// Summary description for XBarRChart.
/// </summary>
public class XBarRChart : SPCTimeVariableControlChart
{
    ChartCalendar startTime = new ChartCalendar();

    //  SPC variable control chart type
    int charttype = SPCControlChartData.MEAN_RANGE_CHART;
    // Number of samples per sub group
    int numsamplespersubgroup = 5;
    // Number of datapoints in the view
    int numdatapointsinview = 17;
    // The time increment between adjacent subgroups
    int timeincrementminutes = 15;

    public XBarRChart()
    {
        // This call is required by the Windows.Forms Form Designer.
        InitializeComponent();
        // Have the chart fill parent client area
        this.Dock = DockStyle.Fill;
        // Define and draw chart
        InitializeChart();
    }

    public void InitializeChart()
    {

        // Initialize the SPCTimeVariableControlChart
        this.InitSPCTimeVariableControlChart(charttype,
            numsamplespersubgroup, numdatapointsinview, timeincrementminutes);
```

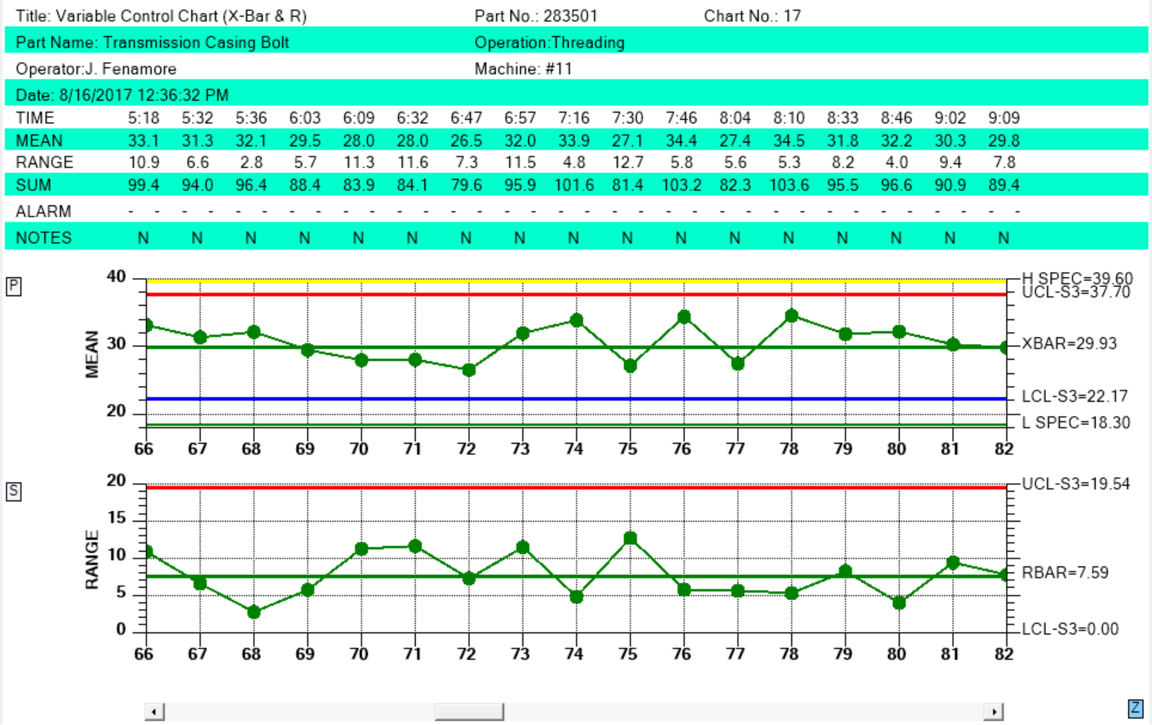
The *timeincrementminutes* parameter in the InitSPCTimeVariableControlChart is now ignored. It is unused in the underlying SPCEventVariableControlChart class because the software no longer considers the exact, or approximate time between sample intervals. Each sample interval update is considered a unique event and the time stamps between samples do not need to be equally spaced. What the SPCTimeVariableControl class does is setup the underlying SPCEventVariableControl to use default time/date values for the x-axis labeling.

Special note about tick mark placement in SPCTimeVariableControlChart

Note the way the x-axis is drawn for the class. It places a tick mark at every sample interval. Major tick marks are placed at sample intervals where the time is considered to have rolled over a major time interval - day changes, hour changes, minute changes, or logical multiples of those. Minor tick marks are place at all other sample intervals. So you do not see a time stamp value at every tick mark. This makes the placement of major tick mark time stamps more efficient when you have many points to plot. However, if you display the table above the chart, and display the time stamp row, (TIME in the picture above), you will see the time stamp for every sample interval.

You may want to display a time stamp at every tick mark, for each and every sample interval. In that case you should use the SPCBatchVariableControlChart, described below, and set the XAxisStringLabelMode propety to SPCChartObjects.AXIS_LABEL_MODE_TIME.

Programs which originally used SPCBatchVariableControlChart will also work as-is.



Title: Variable Control Chart (X-Bar & R) Part No.: 283501 Chart No.: 17

Part Name: Transmission Casing Bolt Operation:Threading

Operator:J. Fenamore Machine: #11

Date: 8/16/2017 12:36:32 PM

TIME	5:18	5:32	5:36	6:03	6:09	6:32	6:47	6:57	7:16	7:30	7:46	8:04	8:10	8:33	8:46	9:02	9:09
MEAN	33.1	31.3	32.1	29.5	28.0	28.0	26.5	32.0	33.9	27.1	34.4	27.4	34.5	31.8	32.2	30.3	29.8
RANGE	10.9	6.6	2.8	5.7	11.3	11.6	7.3	11.5	4.8	12.7	5.8	5.6	5.3	8.2	4.0	9.4	7.8
SUM	99.4	94.0	96.4	88.4	83.9	84.1	79.6	95.9	101.6	81.4	103.2	82.3	103.6	95.5	96.6	90.9	89.4
ALARM	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -
NOTES	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N

SPCBatchVariableControl charts still look the same.

The beginning of your SPC chart class would look like:

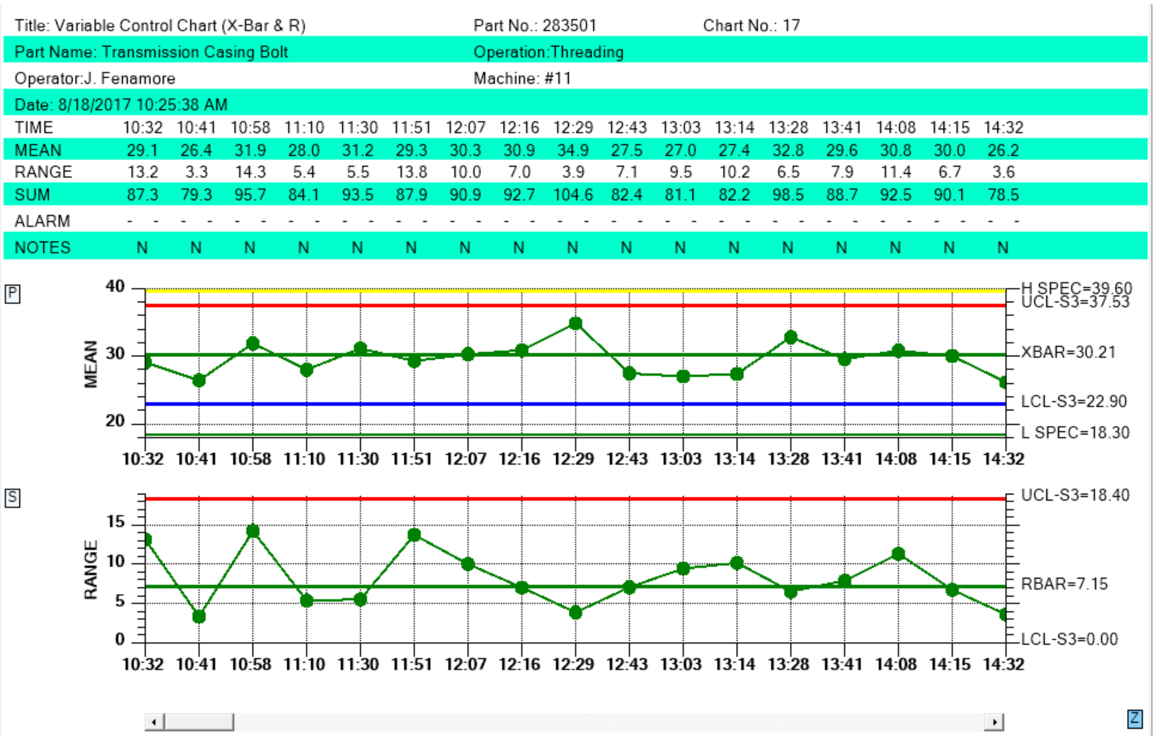
```
public class XBarRChart : SPCBatchVariableControlChart
 {
     ChartCalendar startTime = new ChartCalendar();

     //  SPC variable control chart type
     int charttype = SPCControlChartData.MEAN_RANGE_CHART;
     // Number of samples per sub group
     int numsamplespersubgroup = 5;
     // Number of datapoints in the view
     int numdatapointsinview = 17;

     public XBarRChart()
     {
         // This call is required by the Windows.Forms Form Designer.
         InitializeComponent();
         // Have the chart fill parent client area
         this.Dock = DockStyle.Fill;
         // Define and draw chart
         InitializeChart();
     }


     public void InitializeChart()
     {

         // Initialize the SPCTimeVariableControlChart
         this.InitSPCBatchVariableControlChart(charttype,
             numsamplespersubgroup, numdatapointsinview);
```

What the SPCBatchVariableControl class does is setup the underlying SPCEventVariableControl to use numeric values for the x-axis labeling. You can also update the x-axis with user-defined strings, or time axis labels if you want, exactly as was supported in earlier versions of SPCBatchVariableControl.

Special note about tick mark placement in SPCBatchVariableControlChart

Note the way the x-axis is drawn for the class. It places a major tick mark at every sample interval. So you see a tick mark value for every tick mark. If you want to display the sample interval time stamps at the tick marks, set the set the XAxisStringLabelMode propety to SPCChartObjects.AXIS_LABEL_MODE_TIME. The resulting graph will now look like.

```
XAxisStringLabelMode = SPCChartObjects.AXIS_LABEL_MODE_TIME
```



Title: Variable Control Chart (X-Bar & R) Part No.: 283501 Chart No.: 17

Part Name: Transmission Casing Bolt Operation:Threading

Operator:J. Fenamore Machine: #11

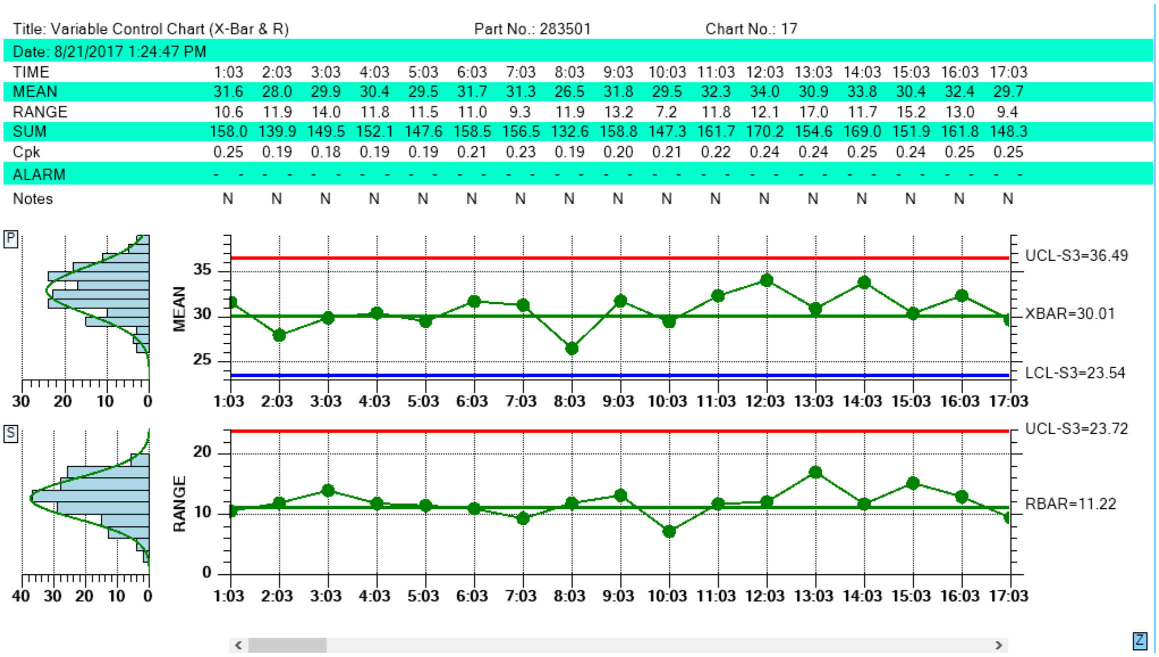
Date: 8/18/2017 10:25:38 AM

TIME	10:32	10:41	10:58	11:10	11:30	11:51	12:07	12:16	12:29	12:43	13:03	13:14	13:28	13:41	14:08	14:15	14:32
MEAN	29.1	26.4	31.9	28.0	31.2	29.3	30.3	30.9	34.9	27.5	27.0	27.4	32.8	29.6	30.8	30.0	26.2
RANGE	13.2	3.3	14.3	5.4	5.5	13.8	10.0	7.0	3.9	7.1	9.5	10.2	6.5	7.9	11.4	6.7	3.6
SUM	87.3	79.3	95.7	84.1	93.5	87.9	90.9	92.7	104.6	82.4	81.1	82.2	98.5	88.7	92.5	90.1	78.5
ALARM	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -
NOTES	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N

Set the XAxisStringLabelMode Property to ***SPCChartObjects.AXIS_LABEL_MODE_TIME*** *for a time axis tick mark label for every sample interval.*

Rather than use SPCTimeVariableControlChart, or SPCBatchVariableControlChart, you can go directly to the underlying base class, SPCEventVariableControlChart. That would look something like:

```
/// <summary>
/// Summary description for XBarRChart.
/// </summary>
public class XBarRChart : SPCEventVariableControlChart
{
    ChartCalendar startTime = new ChartCalendar();

    //  SPC variable control chart type
    int charttype = SPCControlChartData.MEAN_RANGE_CHART;
    // Number of samples per sub group
    int numsamplespersubgroup = 5;
    // Number of datapoints in the view
    int numdatapointsinview = 17;
    // The time increment between adjacent subgroups
    int timeincrementminutes = 15;

    public XBarRChart()
    {
        // This call is required by the Windows.Forms Form Designer.
        InitializeComponent();
        // Have the chart fill parent client area
        this.Dock = DockStyle.Fill;
        // Define and draw chart
        InitializeChart();
    }


    public void InitializeChart()
    {

        // Initialize the SPCTimeVariableControlChart
        this.InitSPCEventVariableControlChart(charttype,
            numsamplespersubgroup, numdatapointsinview);
```



Title: Variable Control Chart (X-Bar & R) Part No.: 283501 Chart No.: 17

Date: 8/21/2017 1:24:47 PM

TIME	1:03	2:03	3:03	4:03	5:03	6:03	7:03	8:03	9:03	10:03	11:03	12:03	13:03	14:03	15:03	16:03	17:03
MEAN	31.6	28.0	29.9	30.4	29.5	31.7	31.3	26.5	31.8	29.5	32.3	34.0	30.9	33.8	30.4	32.4	29.7
RANGE	10.6	11.9	14.0	11.8	11.5	11.0	9.3	11.9	13.2	7.2	11.8	12.1	17.0	11.7	15.2	13.0	9.4
SUM	158.0	139.9	149.5	152.1	147.6	158.5	156.5	132.6	158.8	147.3	161.7	170.2	154.6	169.0	151.9	161.8	148.3
Cpk	0.25	0.19	0.18	0.19	0.19	0.21	0.23	0.19	0.20	0.21	0.22	0.24	0.24	0.25	0.24	0.25	0.25
ALARM	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -
Notes	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N

Derive your chart class directly from SPCEventVariableControlChart

Note that the default x-axis labels for an SPCEventVariableControlChart is time, using a major tick mark, and time axis label for every sample interval. If you want the SPCEventVariableControlChart to use numeric x-axis labels, like a batch chart, set the property XAxisStringLabelMode to one of the following constants:

this.XAxisStringLabelMode =
SPCChartObjects.AXIS_LABEL_MODE_NUMERIC;

Valid constant values for XAxisStringLabelMode are listed below.

SPCChartObjects.AXIS_LABEL_MODE_STRING

Use a user-defined string for the label

SPCChartObjects.AXIS_LABEL_MODE_TIME

Use the event time stamp for the label

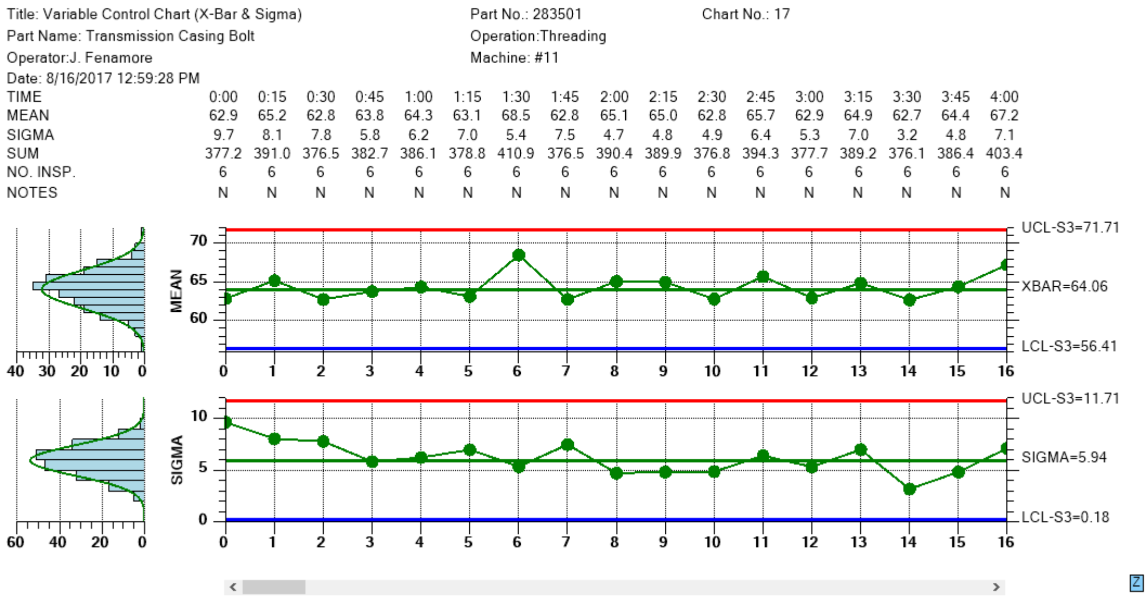
SPCChartObjects.AXIS_LABEL_MODE_NUMERIC

Use the event batch numeric value as the label

SPCChartObjects.AXIS_LABEL_MODE_DEFAULT

Use whatever the default is.

If you set XAxisStringLabelMode property to **SPCChartObjects.AXIS_LABEL_MODE_NUMERIC,** the graph above becomes:



Title: Variable Control Chart (X-Bar & Sigma)
Part Name: Transmission Casing Bolt
Operator:J. Fenamore
Date: 8/16/2017 12:59:28 PM

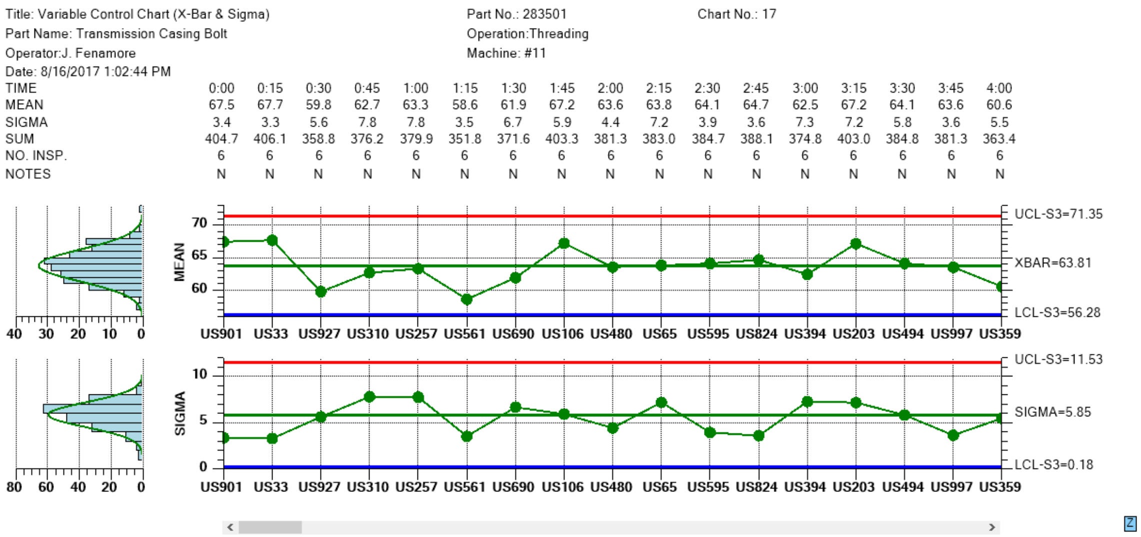
Part No.: 283501
Operation:Threading
Machine: #11

Chart No.: 17

TIME	0:00	0:15	0:30	0:45	1:00	1:15	1:30	1:45	2:00	2:15	2:30	2:45	3:00	3:15	3:30	3:45	4:00
MEAN	62.9	65.2	62.8	63.8	64.3	63.1	68.5	62.8	65.1	65.0	62.8	65.7	62.9	64.9	62.7	64.4	67.2
SIGMA	9.7	8.1	7.8	5.8	6.2	7.0	5.4	7.5	4.7	4.8	4.9	6.4	5.3	7.0	3.2	4.8	7.1
SUM	377.2	391.0	376.5	382.7	386.1	378.8	410.9	376.5	390.4	389.9	376.8	394.3	377.7	389.2	376.1	386.4	403.4
NO. INSP.	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
NOTES	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N

Set XAxisStringLabelMode property to SPCChartObjects.AXIS_LABEL_MODE_NUMERIC, to show the batch numeric value on the x-axis

If you set XAxisStringLabelMode to **SPCChartObjects.AXIS_LABEL_MODE_STRING,** and (very important) assign a user defined string to every sample interval, the graph above becomes:



Title: Variable Control Chart (X-Bar & Sigma) Part No.: 283501 Chart No.: 17
Part Name: Transmission Casing Bolt Operation:Threading
Operator:J. Fenamore Machine: #11
Date: 8/16/2017 1:02:44 PM

TIME	0:00	0:15	0:30	0:45	1:00	1:15	1:30	1:45	2:00	2:15	2:30	2:45	3:00	3:15	3:30	3:45	4:00
MEAN	67.5	67.7	59.8	62.7	63.3	58.6	61.9	67.2	63.6	63.8	64.1	64.7	62.5	67.2	64.1	63.6	60.6
SIGMA	3.4	3.3	5.6	7.8	7.8	3.5	6.7	5.9	4.4	7.2	3.9	3.6	7.3	7.2	5.8	3.6	5.5
SUM	404.7	406.1	358.8	376.2	379.9	351.8	371.6	403.3	381.3	383.0	384.7	388.1	374.8	403.0	384.8	381.3	363.4
NO. INSP.	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
NOTES	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N

40 30 20 10 0
MEAN
70
65
60
UCL-S3=71.35
XBAR=63.81
LCL-S3=56.28
US901 US33 US927 US310 US257 US561 US690 US106 US480 US65 US595 US824 US394 US203 US494 US997 US359
80 60 40 20 0
SIGMA
10
5
0
UCL-S3=11.53
SIGMA=5.85
LCL-S3=0.18
US901 US33 US927 US310 US257 US561 US690 US106 US480 US65 US595 US824 US394 US203 US494 US997 US359

Set XAxisStringLabelMode property to SPCChartObjects.AXIS_LABEL_MODE_STRING, to show the batch user defined string value on the x-axis

The user-defined string is assigned after each sample interval update, something like this:

```
this.ChartData.AddNewSampleRecord(timestamp, samples, notesstring);
                // Make an arbitrary user string.
String userstring = "US" + this.ChartData.CurrentNumberRecords;
this.ChartData.AddAxisUserDefinedString(userstring);
```

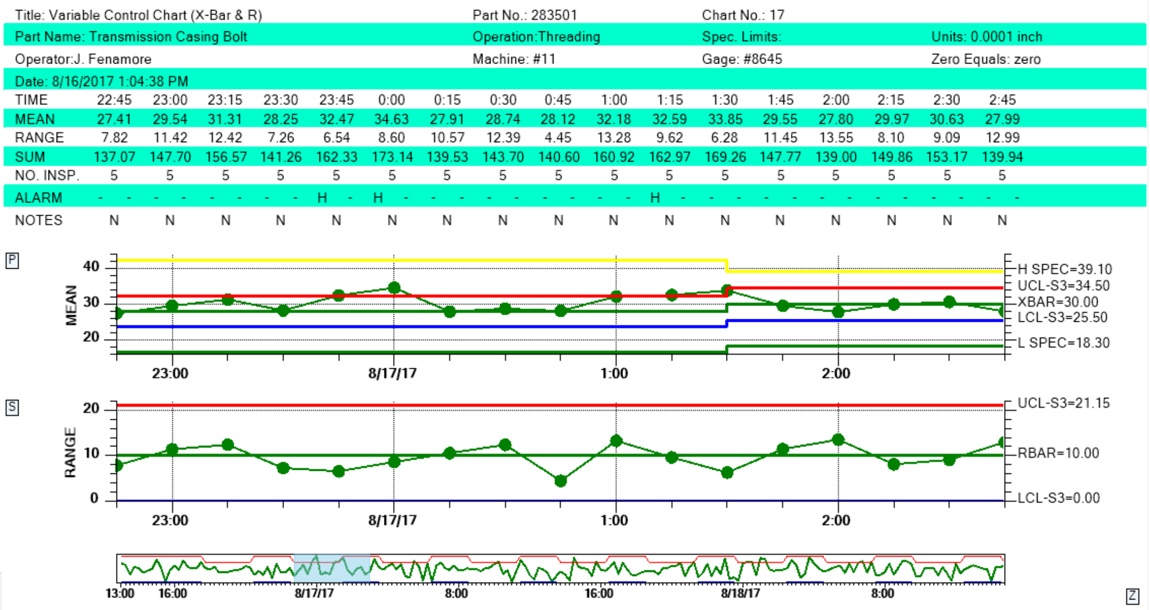
where in this example we just makeup a unique user string based on the current number of records, but in your case it will probably be some string unique to that exact sample interval.

SPCEventAttributeControlChart, SPCBatchAttributeControlChart and SPCTimeAttributeControlChart

The implementatin of SPCTimeAttributeControlChart and SPCBatchAttributeControlChart classes are analogous to the implementations of SPCTimeVariableControlChart and SPCVariableAttributeControlChart classes. They both derive from SPCEventAttributeControlChart. You can either reuse your original code exactly as is, or convert to used the SPCEventAttributeControlChart class directly. The software contains examples of all three.

Zooming as an option for the scrollbar

The horizontal scrollbar can be replaced (toggled on/off actually) using the UI, with a zoom window, by clicking on the z-button in the lower right corner. The user will be able to scroll the charts by dragging the transparent blue zoom window left or right, or expand the x-axis scale by dragging the left or right edge of the zoom window.



Title: Variable Control Chart (X-Bar & R) Part No.: 283501 Chart No.: 17

Part Name: Transmission Casing Bolt Operation:Threading Spec. Limits: Units: 0.0001 inch

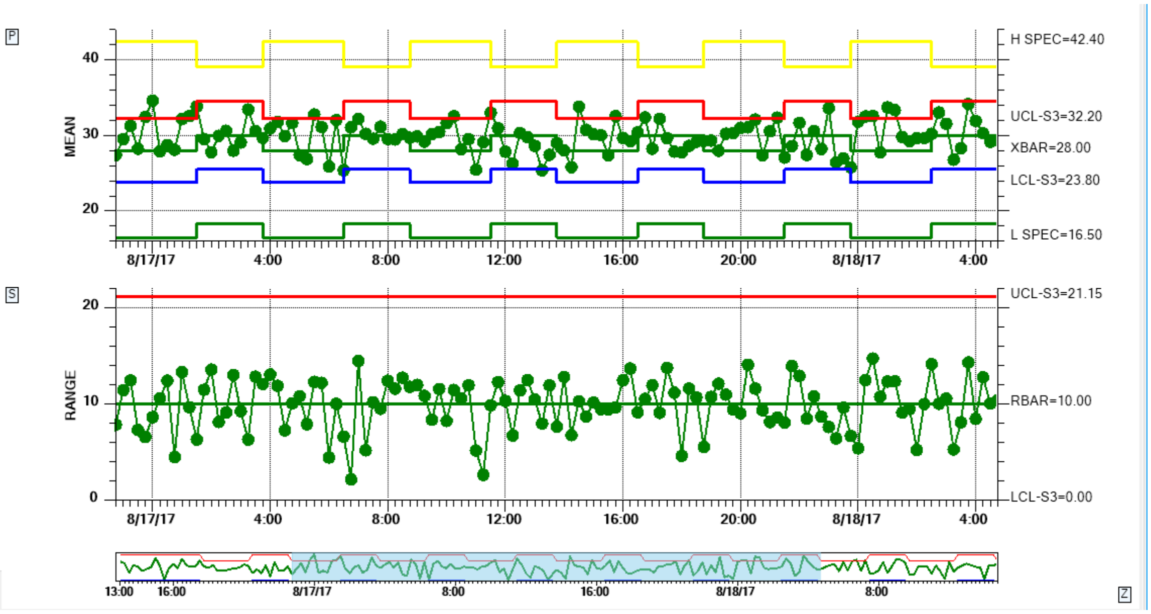
Operator:J. Fenamore Machine: #11 Gage: #8645 Zero Equals: zero

Date: 8/16/2017 1:04:38 PM

TIME	22:45	23:00	23:15	23:30	23:45	0:00	0:15	0:30	0:45	1:00	1:15	1:30	1:45	2:00	2:15	2:30	2:45
MEAN	27.41	29.54	31.31	28.25	32.47	34.63	27.91	28.74	28.12	32.18	32.59	33.85	29.55	27.80	29.97	30.63	27.99
RANGE	7.82	11.42	12.42	7.26	6.54	8.60	10.57	12.39	4.45	13.28	9.62	6.28	11.45	13.55	8.10	9.09	12.99
SUM	137.07	147.70	156.57	141.26	162.33	173.14	139.53	143.70	140.60	160.92	162.97	169.26	147.77	139.00	149.86	153.17	139.94
NO. INSP.	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
ALARM	- -	- -	- -	- -	H -	H -	- -	- -	- -	- -	H -	- -	- -	- -	- -	- -	- -
NOTES	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N

Click on the Z button in the lower right corner and a zoom window will replace the scrollbar. Use the mouse to scroll the charts by dragging the transparent blue zoom window left or right, or expand the x-axis scale by dragging the left or right edge of the zoom window.

If the number of points in the display exceeds the initial setup value for the number of data points, the table is temporarily removed to prevent overlap of the table columns. If the number of data points is reduced to <= the initial number of points, the table will reappear.



If you use the zoom window to display a lot of points, the table at the top will disappear to prevent the table columns from overlapping.

The default display display for all chart types uses the scrollbar. The zoom option is seen as a small button with the character 'Z' in the lower right corner of the display. You enter/exit the zoom mode by clicking on that button. If you do not want the button to show at all, effectively making the zoom option inaccessible to the end user, set EnableZoomToggles property false.

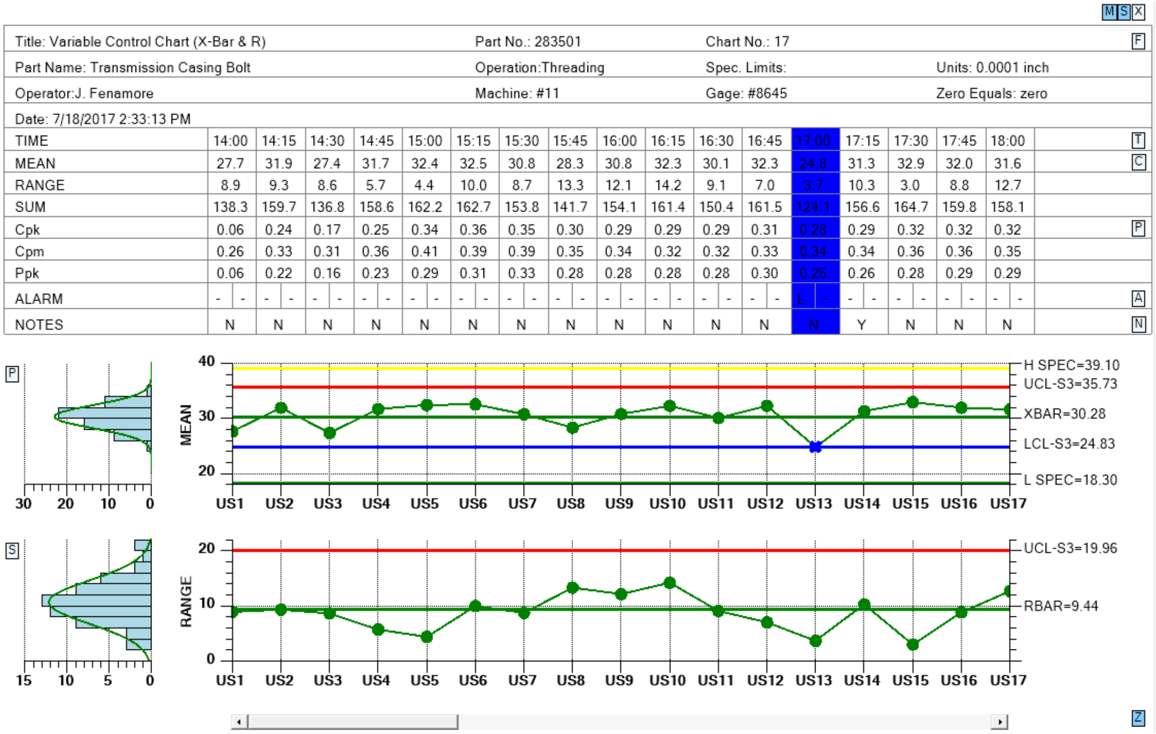
.

```
            this.EnableZoomToggles = false;
```

All of the examples in the NewRev3Features example show this feature.

Collapsible Items

Like the Zoom option described in the previous section, there are also UI buttons which can toggle on and off rows in the table, and the Primary and Secondary charts. Like the Zoom button, these UI buttons can be hidden from the end user if you don't want them to show. See the end of this section for examples. On the top and right of the table buttons will selectively collapse/restore rows of the table, freeing up more space for charts. Buttons to the left and bottom of the Primary and Secondary charts also permit the user to collapse/restore either of those charts, allowing the remaining chart to display in all of the remaining space.



M S X

Title: Variable Control Chart (X-Bar & R) | Part No.: 283501 | Chart No.: 17 | F

Part Name: Transmission Casing Bolt | Operation:Threading | Spec. Limits: | Units: 0.0001 inch

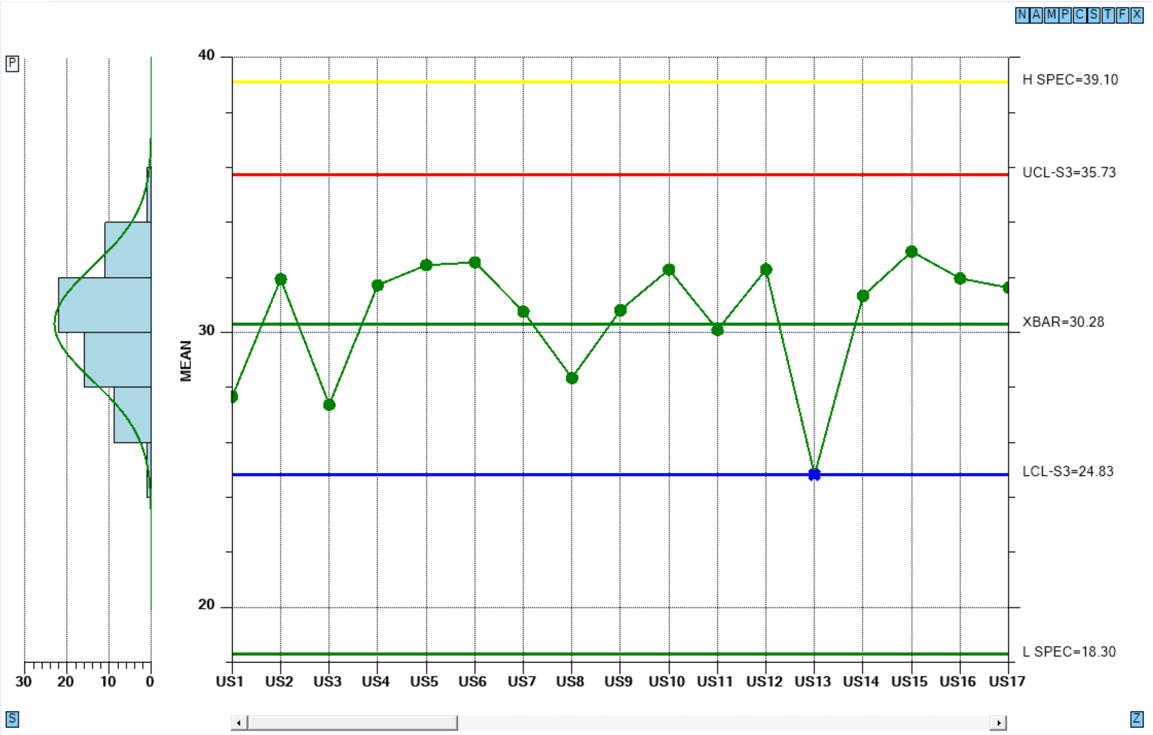
Operator:J. Fenamore | Machine: #11 | Gage: #8645 | Zero Equals: zero

Date: 7/18/2017 2:33:13 PM

TIME	14:00	14:15	14:30	14:45	15:00	15:15	15:30	15:45	16:00	16:15	16:30	16:45	17:00	17:15	17:30	17:45	18:00	T
MEAN	27.7	31.9	27.4	31.7	32.4	32.5	30.8	28.3	30.8	32.3	30.1	32.3	24.8	31.3	32.9	32.0	31.6	C
RANGE	8.9	9.3	8.6	5.7	4.4	10.0	8.7	13.3	12.1	14.2	9.1	7.0	3.7	10.3	3.0	8.8	12.7	
SUM	138.3	159.7	136.8	158.6	162.2	162.7	153.8	141.7	154.1	161.4	150.4	161.5	124.1	156.6	164.7	159.8	158.1	
Cpk	0.06	0.24	0.17	0.25	0.34	0.36	0.35	0.30	0.29	0.29	0.29	0.31	0.28	0.29	0.32	0.32	0.32	P
Cpm	0.26	0.33	0.31	0.36	0.41	0.39	0.39	0.35	0.34	0.32	0.32	0.33	0.34	0.34	0.36	0.36	0.35	
Ppk	0.06	0.22	0.16	0.23	0.29	0.31	0.33	0.28	0.28	0.28	0.28	0.30	0.25	0.26	0.28	0.29	0.29	
ALARM	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	L -	- -	- -	- -	- -	A
NOTES	N	N	N	N	N	N	N	N	N	N	N	N	N	Y	N	N	N	N

Buttons on the top, right and left of the display permit the user to selectively collapse portions of the chart.

Click on the N, A, M, P C and S buttons and shrink the table to following size.



In the example above, all of the chart options except for the Primary chart, have been toggled off using the buttons.

The buttons at the right of the table (and at the top right if toggled off) use the following ID's.

X Turn on/off all table items at once
F Turn on/off form data
T Turn on/off sample interval time stamp data
S Turn on/off sample value data
C Turn on/off calculated value (mean, range, sum, etc.) data
P Turn on/off process capability data
M Turn on/off number of samples data
A Turn on/off alarm data
N Turn on/off notes data
Z Bottom right - Turn on/off zoom control – the only button not affected by the X button above

The buttons at the left of the primary and secondary charts use the following ID's.

P Turn on/off the Primary chart
S Turn on/off the Secondary chart

If you do not want the buttons to show at all, effectively making these UI options inaccessible to the end user, set these properties false.

.

Hide the chart buttons on the left.

```
            this.EnableChartToggles = false;
```

Hide the table buttons on the right.

```
            this.EnableTableRowToggles = false;
```

You can hide ALL of the UI buttons (zoom, chart, and table) using the property EnableDisplayOptionToggles. This is what you use if you do not want the end user to have any control over the display of the table and chart.

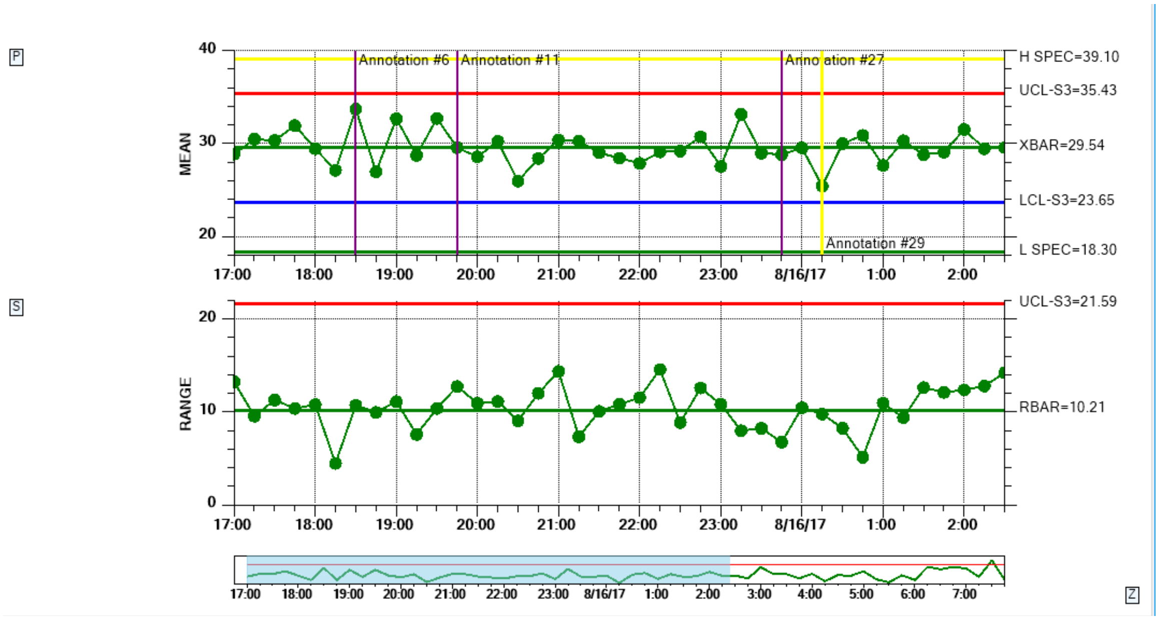
```
           this. EnableDisplayOptionToggles = false;
```

If you want to selectively enable options, first set EnableDisplayOptionToggles true. The other button display enables won't work unless this option is true. Then disable the options you do not want to show. In the example below, only the zoom option is left visible.

```
           this.EnableDisplayOptionToggles = true;
           this.EnableTableRowToggles = false;
           this.EnableChartToggles = false;
           this.EnableZoomToggle = true;
```

Enhanced Annotations

The chart annotations have been enhanced with a vertical line with accompanying text using programmer specified justification.



The enhanced chart annotations include a vertical line to mark the data point, and many justification options (top, middle, bottom, left, right and center).

The new version of AddAnnotation is just an override of the original, with added parameters to specify the vertical line attributes and the text justification.

AddAnnotation

Add an annotation to a data point in the specified SPC chart.

```
public int AddAnnotation(int chart, int datapointindex, String text, int just,
ChartAttribute attrib)
```

where:

chart Specifies whether the annotation is added to the primary, or secondary chart. Use one of the SPChartObjects constants: SPCChartObjects.PRIMARY_CHART or SPCChartObjects.SECONDARY_CHART.

datapointindex The index of the data point the annotation is for.

text A string string representing the annotation.

just The justification for the text to the x-position of the annotation. Use one of the annotation justification constants: ANNOTATION_UPPER_RIGHT, ANNOTATION_UPPER_LEFT, ANNOTATION_LOWER_RIGHT, ANNOTATION_LOWER_LEFT, ANNOTATION_UPPER_CENTER, ANNOTATION_LOWER_CENTER,

ANNOTATION_DATAPOINT_RIGHT,
ANNOTATION_DATAPOINT_LEFT

attrib A the attribute of the vertical line.

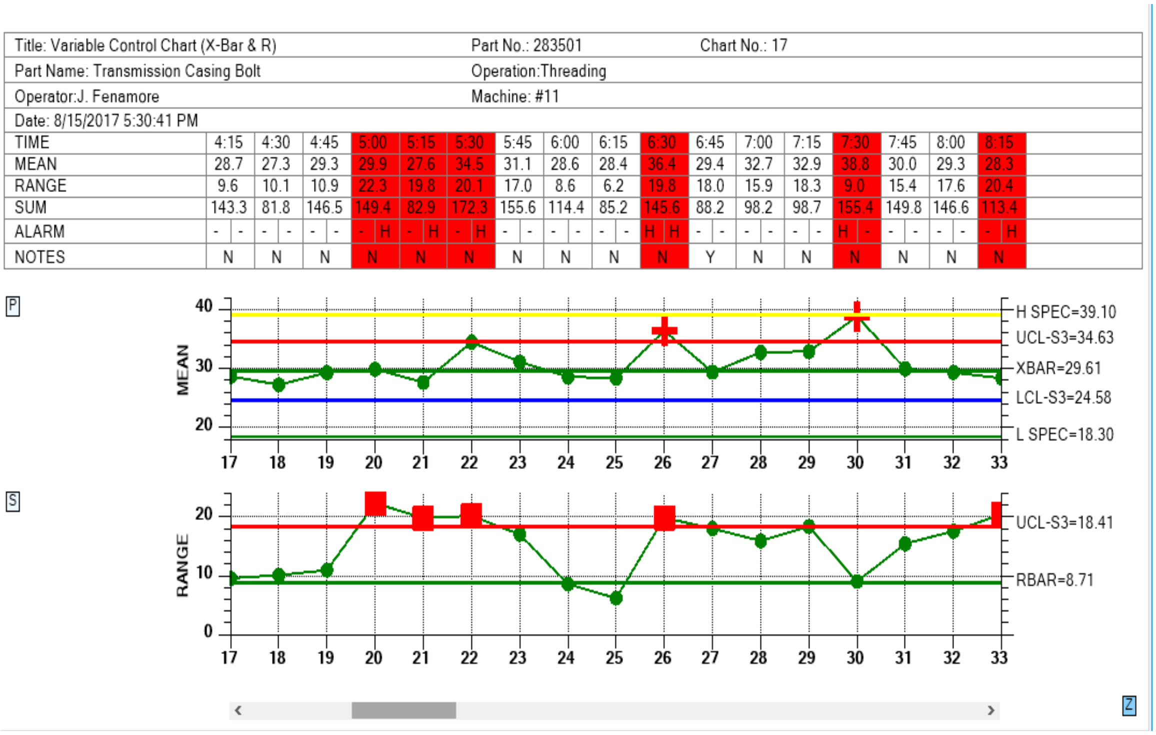
You call the AddAnnotation method immediately after the AddNewSampleRecord method call. For Example:

```
          // Add the new sample subgroup to the chart
this.ChartData.AddNewSampleRecord(timestamp, samples, notesstring);

int index = this.ChartData.CurrentNumberRecords - 1;
string annotstring = "Annotation #" + index.ToString();
ChartAttribute lineattrib = new ChartAttribute(Color.Purple, 2,
         DashStyle.Solid);
// Adjust justification to minimize overlap of adjacent annotations
int annotjust = SPCAnnotation.ANNOTATION_UPPER_LEFT;

this.AddAnnotation(SPCChartObjects.PRIMARY_CHART, index,
        annotstring, annotjust, lineattrib);
```

See the NewRev3Features.Annotations demo for an example.

Enhanced Out of Limit Symbol

One of the SPC chart options is to mark a data point which is outside of control limits by changing the color of the symbol. It is now possible to also also change the size and symbol type for out of limit symbols.



Title: Variable Control Chart (X-Bar & R) Part No.: 283501 Chart No.: 17

Part Name: Transmission Casing Bolt Operation:Threading

Operator:J. Fenamore Machine: #11

Date: 8/15/2017 5:30:41 PM

TIME	4:15	4:30	4:45	5:00	5:15	5:30	5:45	6:00	6:15	6:30	6:45	7:00	7:15	7:30	7:45	8:00	8:15
MEAN	28.7	27.3	29.3	29.9	27.6	34.5	31.1	28.6	28.4	36.4	29.4	32.7	32.9	38.8	30.0	29.3	28.3
RANGE	9.6	10.1	10.9	22.3	19.8	20.1	17.0	8.6	6.2	19.8	18.0	15.9	18.3	9.0	15.4	17.6	20.4
SUM	143.3	81.8	146.5	149.4	82.9	172.3	155.6	114.4	85.2	145.6	88.2	98.2	98.7	155.4	149.8	146.6	113.4
ALARM	- -	- -	- -	- H	- H	- H	- -	- -	- -	H H	- -	- -	- -	H -	- -	- -	- H
NOTES	N	N	N	N	N	N	N	N	N	N	Y	N	N	N	N	N	N

A data point which is outside of control limits can be automatically marked using color, symbol type, and color.

In the example above, the out of control symbol for the Primary chart is an extra large plus sign, while for the the Secondary chart it is an extra large square, both contrasting with the default circle symbol.

The default symbol for the out of control indication is the same as the in control indication, a filled circle. Only a color change signifies out of control. To change the notification symbol, and size, use the OutOfLimitSymbolNumber and OutOfLimitSymbolSize properties, which apply separately to the Primary and Secondary charts.

OutOfLimitSymbolNumber

Set the symbol type for data points found to be in alarm. Use one of the symbol type constants found in the ChartObj class: SQUARE, TRIANGLE, DIAMOND, CROSS, PLUS, STAR, LINE, HBAR, VBAR, CIRCLE. :

OutOfLimitSymbolSize

Set the size in pixels of the out of limit symbol:

For example:

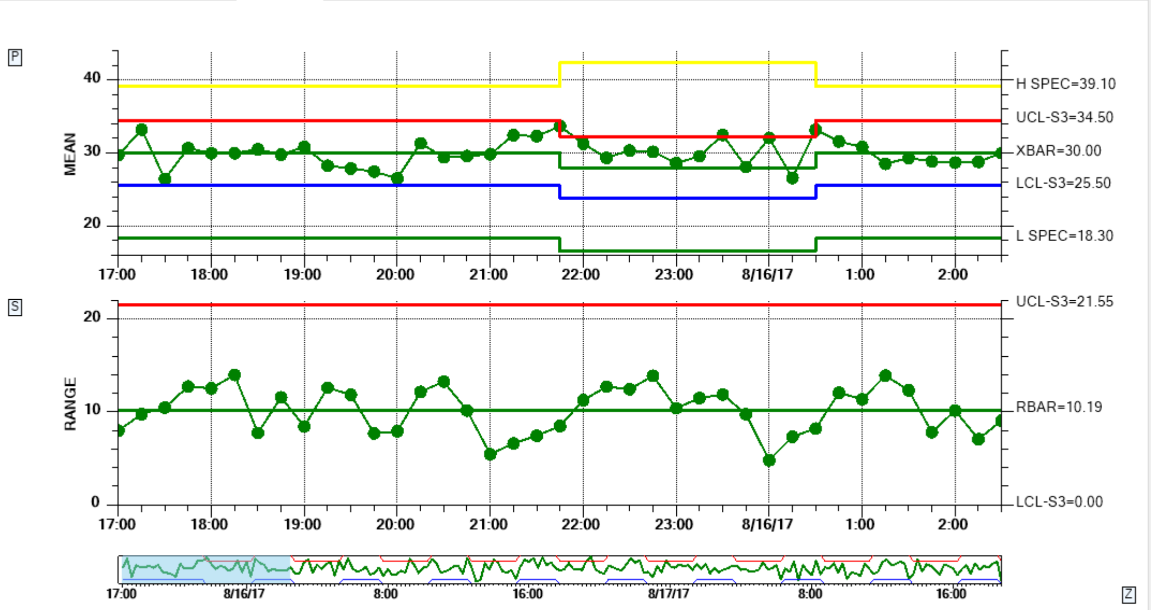
```
// Need to have this on for symbol emphasis
```

```
            this.ChartAlarmEmphasisMode = SPCChartBase.ALARM_HIGHLIGHT_SYMBOL;


// Set symbol emphasis type, and size, for primary chart
            this.PrimaryChart.OutOfLimitSymbolNumber = ChartObj.PLUS;
            this.PrimaryChart.OutOfLimitSymbolSize = 22;

// Set symbol emphasis type, and size, for secondary chart
            this.SecondaryChart.OutOfLimitSymbolNumber = ChartObj.SQUARE;
            this.SecondaryChart.OutOfLimitSymbolSize = 18;
```

See the NewRev3Features.EnhanceLimitSymbols demo for an example.

Variable Spec and Control Limits

In addition to variable control limits, charts can now have variable specification limits.



Control limits and specification limits can be adjusted on a sample interval by sample interval basis.

You can set/change the current control limits immediately before the AddNewSampleRecord call. There are a couple of ways to do this. The preferred way to call the ChartData UpdateControlLimitsUsingMeanAndSigma. Because this method runs through every control limit for a given chart (Primary and Secondary charts are treated separately) and updates them all. If you have plotted the +-1, and +-2 sigma limits, in addition to the +-3 sigma limits, these all need to be updated. If you are using any of the Named Control Rules (WECO, Nelson, etc.) then you have can have up to 20 active limits, and every limit needs to be updated. It is much, much, easier to do that with one call, specifying the new process mean and sigma value, than to try and do it limit by limit. If you plan to set the Secondary chart values this way as well, do not use the

Primary chart mean and sigma values. You must use the center line value you want for the Secondary chart as the mean, and the sigma value of the Secondary chart as the sigma. For the Secondary chart, the center line value is approximately equal to the Primary chart sigma value and the Secondary chart sigma value is 1/3 * (Secondary chart +3 sigma limit minus the center line value).

UpdateControlLimitsUsingMeanAndSigma

Update the control limits using a process mean and sigma value

```
public void UpdateControlLimitsUsingMeanAndSigma(int chartpos, double mean, double
sigma)
```

chartpos	specifiy the chart using one of the SPCChartObjects chart position constants: PRIMARY_CHART or SECONDARY_CHART
mean	specify the mean of the given chart.
sigma	specify the sigma of the given chart

For example:

```
this.ChartData.UpdateControlLimitsUsingMeanAndSigma(SPCChartObjects.PRIMARY_CHART,
        newMeanP, newSigmaP);
 this.ChartData.AddNewSampleRecord(timestamp, samples, notesstring);
```

If your chart only has +-3 sigma limits, you can also use the original SetControlLimitValues method, setting the value for each limit explicitly.

SetControlLimitValues

Set the SPC control limit values for an SPC control chart.

```
public void SetControlLimitValues(double [] limitvalues )
```

limitvalues	An array of double values, one for each control limit in the SPC chart, sorted in the following order [SPC_PRIMARY_CONTROL_TARGET, SPC_PRIMARY_LOWER_CONTROL_LIMIT, SPC_PRIMARY_UPPER_CONTROL_LIMIT, SPC_SECONDARY_CONTROL_TARGET, SPC_SECONDARY_LOWER_CONTROL_LIMIT, SPC_SECONDARY_UPPER_CONTROL_LIMIT].

For example:

```
changeControlLimits = { 28, 23, 33, 9, 0, 18 };
```

```
.
.
.
this.ChartData.SetControlLimitValues(changeControlLimits);
this.ChartData.AddNewSampleRecord(timestamp, samples, notesstring);
```

You can also change the high and low specification limits using the ChartData.SetSpecLimits method.

SetSpecLimits

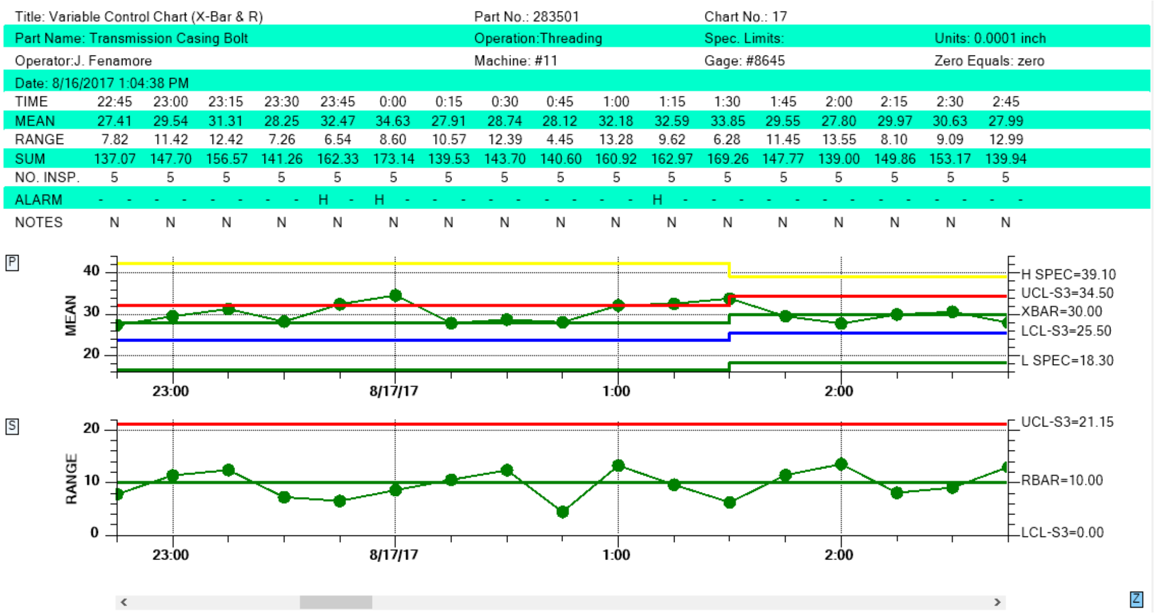
```
public void SetSpecLimits(double lowspeclimit, double highspeclimit)
```

Set high and low specification limits

lowspeclimit	The low spec limit.
highspeclimit	The high spec limit.

The example below updates the control limits and the spec limits.

```
// Create the spec limits in the chart setup
this.PrimaryChart.AddSpecLimit(SPCChartObjects.SPC_LOWER_SPEC_LIMIT, initialLSL,
                     "L SPEC", new ChartAttribute(Colors.Green, 3.0));
this.PrimaryChart.AddSpecLimit(SPCChartObjects.SPC_UPPER_SPEC_LIMIT, initialHSL,
"H SPEC", new ChartAttribute(Colors.Yellow, 3.0));
.
.
.

this.ChartData.UpdateControlLimitsUsingMeanAndSigma(SPCChartObjects.PRIMARY_CHART,
                     originalMeanP, originalSigmaP);
this.ChartData.SetSpecLimits(initialLSL, initialHSL);
this.ChartData.AddNewSampleRecord(timestamp, samples, notesstring);
```



Title: Variable Control Chart (X-Bar & R) Part No.: 283501 Chart No.: 17

Part Name: Transmission Casing Bolt Operation:Threading Spec. Limits: Units: 0.0001 inch

Operator:J. Fenamore Machine: #11 Gage: #8645 Zero Equals: zero

Date: 8/16/2017 1:04:38 PM

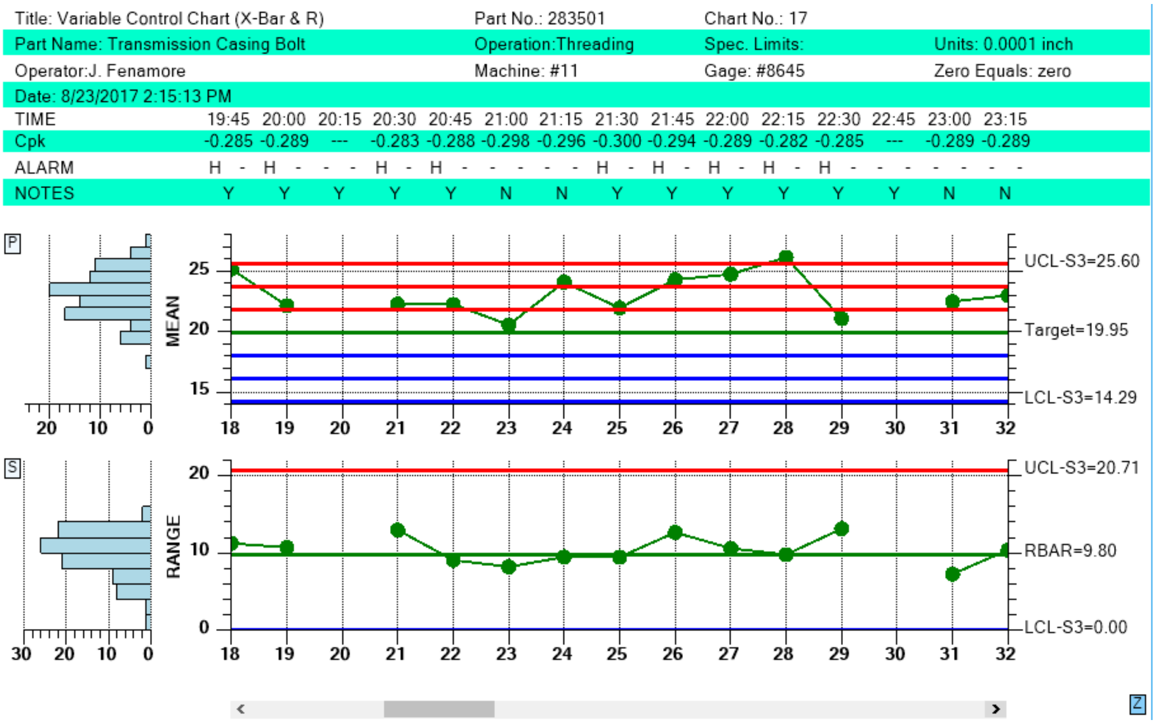
TIME	22:45	23:00	23:15	23:30	23:45	0:00	0:15	0:30	0:45	1:00	1:15	1:30	1:45	2:00	2:15	2:30	2:45
MEAN	27.41	29.54	31.31	28.25	32.47	34.63	27.91	28.74	28.12	32.18	32.59	33.85	29.55	27.80	29.97	30.63	27.99
RANGE	7.82	11.42	12.42	7.26	6.54	8.60	10.57	12.39	4.45	13.28	9.62	6.28	11.45	13.55	8.10	9.09	12.99
SUM	137.07	147.70	156.57	141.26	162.33	173.14	139.53	143.70	140.60	160.92	162.97	169.26	147.77	139.00	149.86	153.17	139.94
NO. INSP.	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
ALARM	- -	- -	- -	- -	H -	H -	- -	- -	- -	- -	H -	- -	- -	- -	- -	- -	- -
NOTES	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N

In this example, the control limits and the specification limits are changed in mid course.

For a complete example, see the demo NewRev3Features.VariableControlAndSpecLimits.

Marking a sample internal as bad

A sample interval can be marked as bad. This will disable plotting of the sample interval in the Primary and Secondary charts. It will also prevent the sample interval values from being evaluated in control limit testing, and in auto- calculations for control limits and y-axis range.



Title: Variable Control Chart (X-Bar & R)	Part No.: 283501	Chart No.: 17	
Part Name: Transmission Casing Bolt	Operation:Threading	Spec. Limits:	Units: 0.0001 inch
Operator:J. Fenamore	Machine: #11	Gage: #8645	Zero Equals: zero
Date: 8/23/2017 2:15:13 PM			

TIME	19:45	20:00	20:15	20:30	20:45	21:00	21:15	21:30	21:45	22:00	22:15	22:30	22:45	23:00	23:15
Cpk	-0.285	-0.289	---	-0.283	-0.288	-0.298	-0.296	-0.300	-0.294	-0.289	-0.282	-0.285	---	-0.289	-0.289
ALARM	H -	H -	- -	H -	H -	- -	- -	H -	H -	H -	H -	H -	- -	- -	- -
NOTES	Y	Y	Y	Y	Y	N	N	Y	Y	Y	Y	Y	Y	N	N

A hole (sample intervals 20 and 30 in the picture above) will appear in the main plot of the Primary and Secondary charts when a sample interval is marked bad.

ExcludeRecordFromControlLimitCalculations

Exclude the specified record from the SOC control limit calculations.

```
public void ExcludeRecordFromControlLimitCalculations(int item, bool exclude)
```

item The index of the item to exclude.

exclude Set true and the item is excluded. Set false and it is included.

For example:

```
int currentRecordNum = this.ChartData.CurrentNumberRecords - 1;

this.ChartData.ExcludeRecordFromControlLimitCalculations(currentRecordNum, true);
```

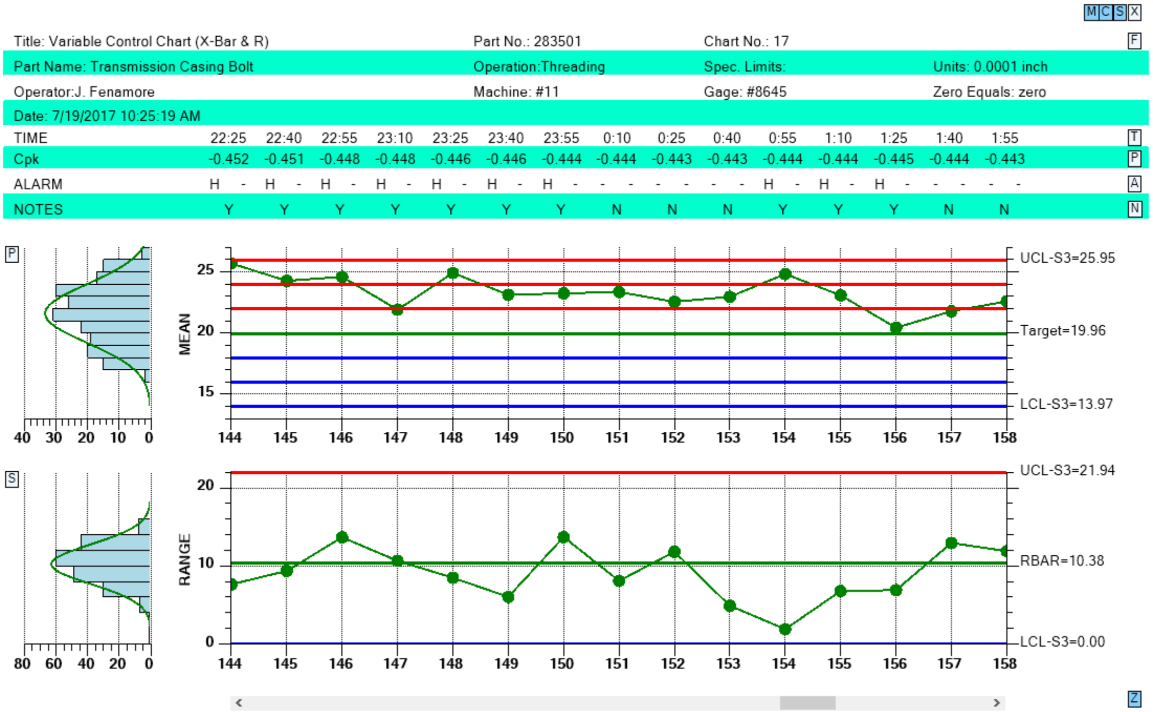
For a complete example, see the module NewRev3Features.IgnoreSampleInterval.

Reset N of M counters for control moves

There are cases when a user wants to keeps the same chart going, adding new sample intervals with new data, after the issue which caused the process to go out of control has been remedied. But many of the control limit tests found in the named control rules (WECO, Nelson, etc.) use N of M tests, where N out of M sample intervals must violate a specific rule. For example, a WECO rule says that if 4 out of 5 samples are outside of 1-sigma, it is an alarm condition. But if the process is now in control, the user may want to reset all N or M counts back to 0, exactly as if the plotting of the chart had started at the first sample interval. So we have added a reset mechanism for the N or M rules to a zero count for all rules.

Use the ChartData method ReCenterControlLimits to reset the N of M counters back to 0.

```
this.ChartData.ReCenterControlLimits();
```



Title: Variable Control Chart (X-Bar & R) Part No.: 283501 Chart No.: 17
Part Name: Transmission Casing Bolt Operation:Threading Spec. Limits: Units: 0.0001 inch
Operator:J. Fenamore Machine: #11 Gage: #8645 Zero Equals: zero
Date: 7/19/2017 10:25:19 AM

TIME	22:25	22:40	22:55	23:10	23:25	23:40	23:55	0:10	0:25	0:40	0:55	1:10	1:25	1:40	1:55
Cpk	-0.452	-0.451	-0.448	-0.448	-0.446	-0.446	-0.444	-0.444	-0.443	-0.443	-0.444	-0.444	-0.445	-0.444	-0.443
ALARM	H -	H -	H -	H -	H -	H -	H -	- -	- -	- -	H -	H -	H -	- -	- -
NOTES	Y	Y	Y	Y	Y	Y	Y	N	N	N	Y	Y	Y	N	N

The N of M counters are reset after sample interval 150 update

The example above uses the Nelson Rules. The samples intervals from 144 to 150 are shown to be in alarm, either 2 of 3 greater than 2-sigma, or 4 of 5 greater than 1-sigma. But sample interval 151 is greater than 1-sigma and should show a 4 of 5 greater than 1-sigma alarm. Yet it is not shown to be in alarm. This is because after the update for sample interval 150, the ReCenterControlLimits method was called, resetting the counters for all N or M test back to zero. So it takes four additional 1-sigma violations to re-trigger the 4 of 5 greater than 1-sigma alarm.

Remove Negative LCL control limits for Variable Control Secondary charts, or Attribute Control Primary charts

We added a routine which when called will disable (both from display and alarm checking) any chart LCL control limit if the limit value is <= a specified value, 0.0 in most cases. Use the ChartData SetAutoDeleteControlLimits method for this. We made the method more general than a simple routine to just delete negative control limits. It will remove any control limit that is (<, <=, >, >=) the specified value.

If a control limit meets the test criteria, which compares the control limit value, to the specified value, using the specified criteria, it is removed.

SetAutoDeleteControlLimits

```
public void SetAutoDeleteControlLimits(int chartpos, int compop, double value)
```

chartpos Restrict the operation to Primary or Secondary chart. Use the constant SPCChartObjects.PRIMARY_CHART or SPCChartObjects.SECONDARY_CHART

compop Use this comparison operator. Use one of the comparison operator constants: SPCControlChartData.COMPARISON_OP_LESSTHAN, SPCControlChartData.COMPARISON_OP_LESSTHAN_OR_EQ, SPCControlChartData.COMPARISON_OP_GREATERTHAN, SPCControlChartData.COMPARISON_OP_GREATERTHAN_OR_EQ,

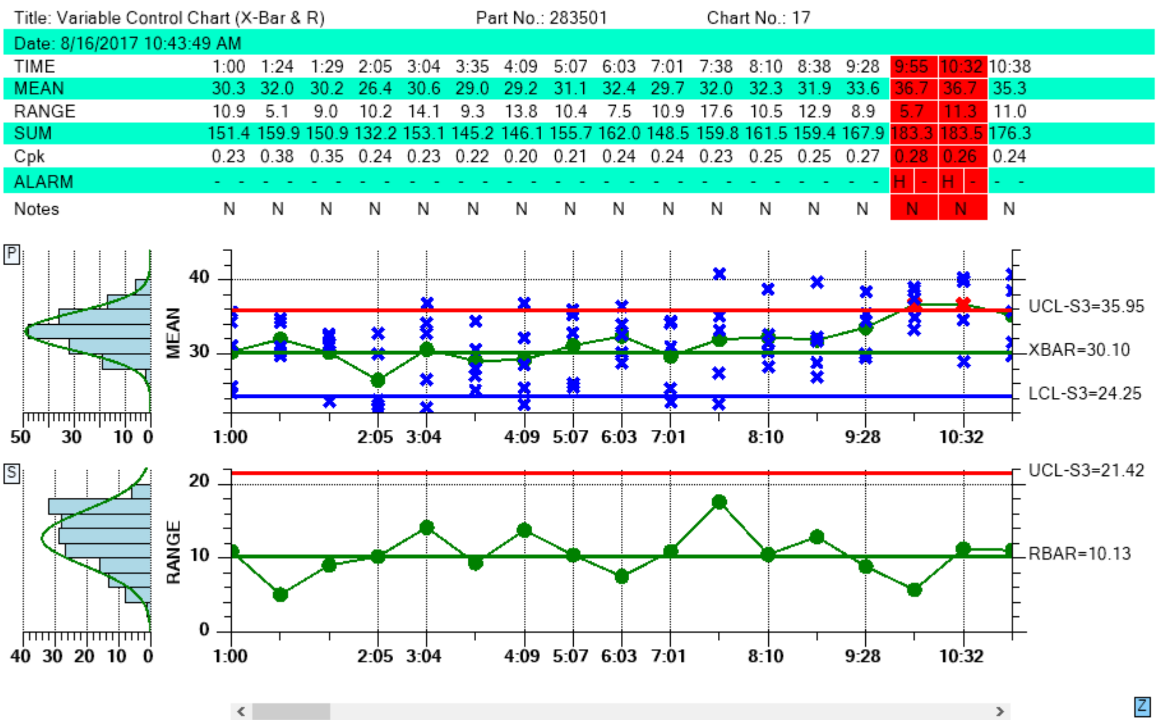
value Value used in comparison.

For a Variable Control chart, If you want to remove the LCL limit, equal to 0.0, from the Range (Secondary chart), call SetAutoDeleteControlLimits with the following parameters.

```
this.ChartData.SetAutoDeleteControlLimits(SPCChartObjects.SECONDARY_CHART,
SPCControlChartData.COMPARISON_OP_LESSTHAN_OR_EQ, 0.0);
```

For an Attributes Control chart, If you want to remove the LCL limit, equal to 0.0, from the Primary Chart, call SetAutoDeleteControlLimits with the following parameters.

```
this.ChartData.SetAutoDeleteControlLimits(SPCChartObjects.PRIMARY_CHART,
SPCControlChartData.COMPARISON_OP_LESSTHAN_OR_EQ, 0.0);
```



Title: Variable Control Chart (X-Bar & R) Part No.: 283501 Chart No.: 17

Date: 8/16/2017 10:43:49 AM

TIME	1:00	1:24	1:29	2:05	3:04	3:35	4:09	5:07	6:03	7:01	7:38	8:10	8:38	9:28	9:55	10:32	10:38
MEAN	30.3	32.0	30.2	26.4	30.6	29.0	29.2	31.1	32.4	29.7	32.0	32.3	31.9	33.6	36.7	36.7	35.3
RANGE	10.9	5.1	9.0	10.2	14.1	9.3	13.8	10.4	7.5	10.9	17.6	10.5	12.9	8.9	5.7	11.3	11.0
SUM	151.4	159.9	150.9	132.2	153.1	145.2	146.1	155.7	162.0	148.5	159.8	161.5	159.4	167.9	183.3	183.5	176.3
Cpk	0.23	0.38	0.35	0.24	0.23	0.22	0.20	0.21	0.24	0.24	0.23	0.25	0.25	0.27	0.28	0.26	0.24
ALARM	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	H -	H -	- -
Notes	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N

Note that the lower control limit for the Secondary (Range) chart is not present. It was calculated be 0.0, and was removed by the software.

If you plan to fill the area between the control limit lines and the center line (zone colors) you must leave the 0.0 lower control limit in place so that the software can fill between the limit and the center line.

N of M testing when the most recent point entering the test is within limits

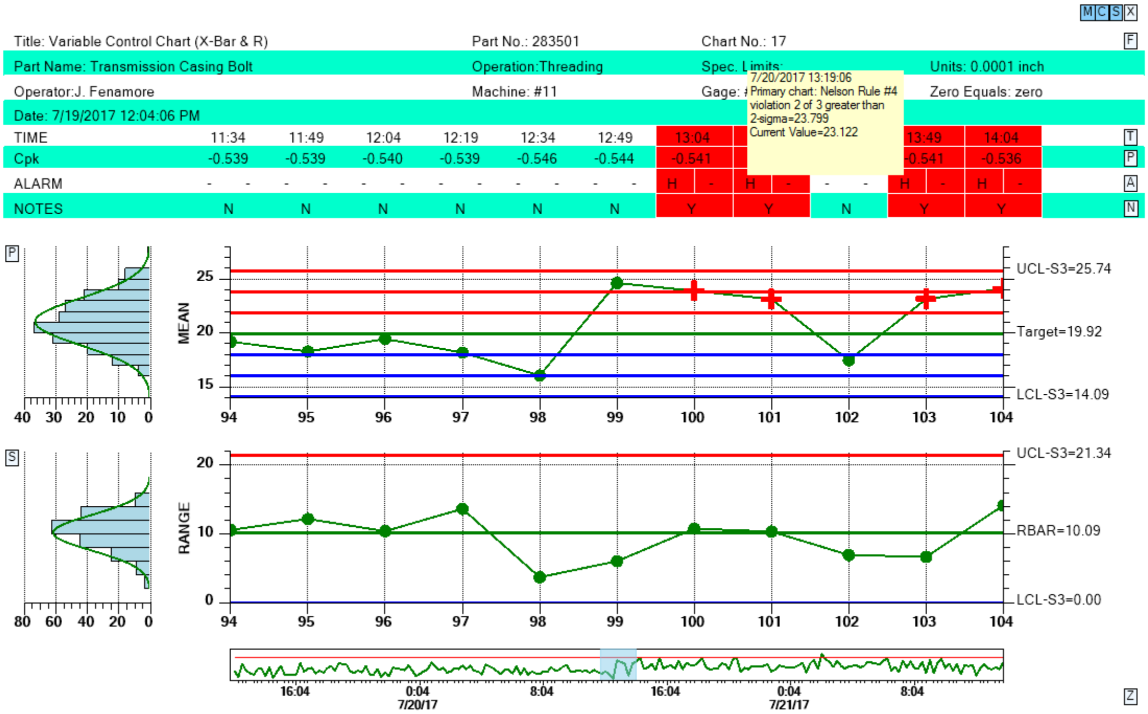
Our default mode takes a strict approach to N of M testing. Regardless of the value of the most recent point to enter the calculation (even if it is within the test limits), if N of M values are outside of limits we consider the sample interval to be in alarm. But some customers challenged this interpretation and presented us with published examples which show that if the most recent point is within limits and the previous N points out of limits, the sample interval should NOT be considered in alarm. The logic for those using the alternative evaluation scheme is that after the the first N values were found to fail the N of M test, a correction was made to the process. And therefore the next sample interval is within limits and should not be in alarm, even though it fails the strict N or M test, since it still picks up the N out of limit values before the process was corrected. We researched the issue and found no agreement. It is implemented in the published literature both ways, going back 50 years. So a simple global flag has been added you can use to choose one evaluation method or the other, with the default being our original method.

If you want to change the N of M evaluation scheme from the default (strict N of M testing), to the alternative method, use the SPCControlLimitRecord

```
SPCControlLimitRecord.DefaultAltNofMRule = true;
```

This is a static property and once set, will affect all charts that are created for the duration of the program. So you can set it at the start of your program and not have to worry about setting in subsequent chart setups.

In the picture below, the default method of N of M valuation produces an alarm at sample interval 101. While sample 101 is within 2-sigma, the previous two samples were greater than 2-sigma, so the 2 out of 3 > 2-sigma test fails for sample interval 101.



Title: Variable Control Chart (X-Bar & R) Part No.: 283501 Chart No.: 17

Part Name: Transmission Casing Bolt Operation:Threading Spec. Limits: Units: 0.0001 inch

Operator:J. Fenamore Machine: #11 Gage: Zero Equals: zero

Date: 7/19/2017 12:04:06 PM

TIME	11:34	11:49	12:04	12:19	12:34	12:49	13:04			13:49	14:04
Cpk	-0.539	-0.539	-0.540	-0.539	-0.546	-0.544	-0.541			-0.541	-0.536
ALARM	- -	- -	- -	- -	- -	- -	H -	H -	- -	H -	H -
NOTES	N	N	N	N	N	N	Y	Y	N	Y	Y

7/20/2017 13:19:06
Primary chart: Nelson Rule #4 violation 2 of 3 greater than 2-sigma=23.799
Current Value=23.122

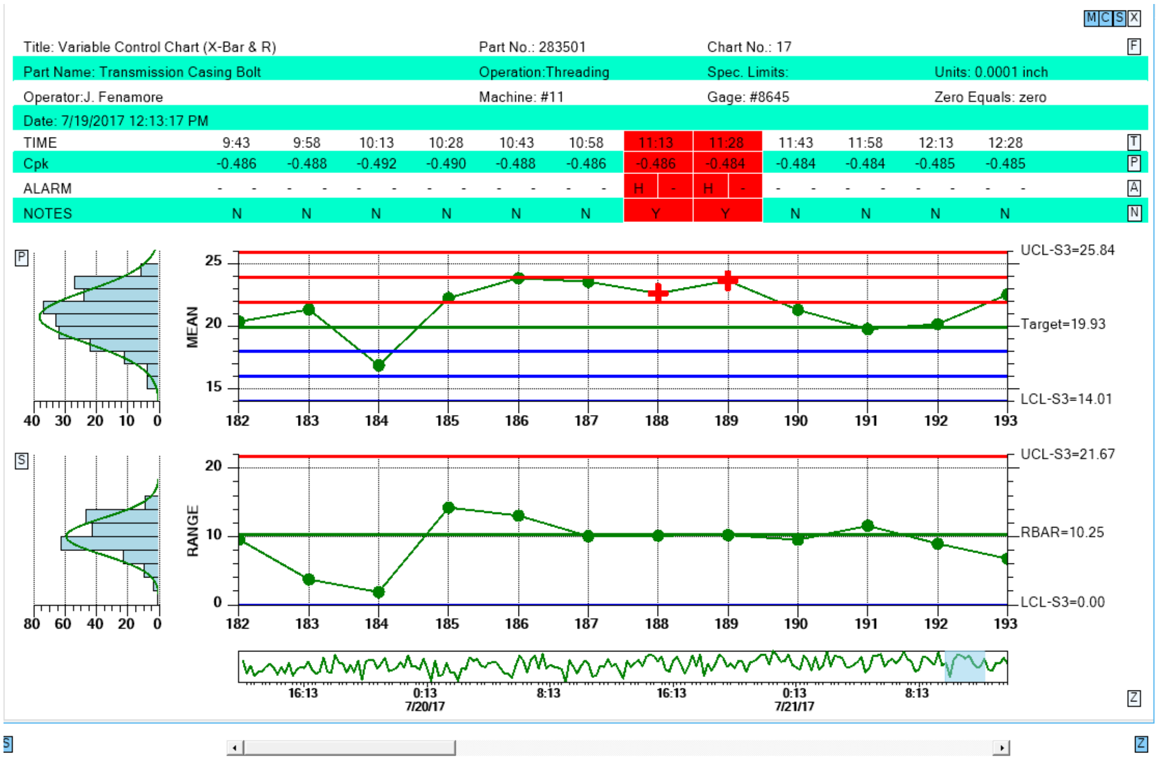
Using the default N of M testing, the sample interval 101 fails the 2 out of 3 > 2-sigma test of the WECO and Nelson rules.

If the following code was added to the chart setup

```
SPCControlLimitRecord.DefaultAltNofMRule = true;
```

sample 101 would NOT be in alarm, even though the previous two samples were > 2-sigma.

In the picture below, DefaultAltNofMRule property has been set true. Using the regular, default rules, the sample interval at 190 would be considered in alarm because 4 out of 5 samples intervals were greater than 1-sigma. But since the alternative evaluation method for N of M rules is enabled, it is shown as NOT being in alarm, because it is within 1-sigma.



Title: Variable Control Chart (X-Bar & R) Part No.: 283501 Chart No.: 17

Part Name: Transmission Casing Bolt Operation:Threading Spec. Limits: Units: 0.0001 inch

Operator:J. Fenamore Machine: #11 Gage: #8645 Zero Equals: zero

Date: 7/19/2017 12:13:17 PM

TIME	9:43	9:58	10:13	10:28	10:43	10:58	11:13	11:28	11:43	11:58	12:13	12:28
Cpk	-0.486	-0.488	-0.492	-0.490	-0.488	-0.486	-0.486	-0.484	-0.484	-0.484	-0.485	-0.485
ALARM	- -	- -	- -	- -	- -	- -	H -	H -	- -	- -	- -	- -
NOTES	N	N	N	N	N	N	Y	Y	N	N	N	N

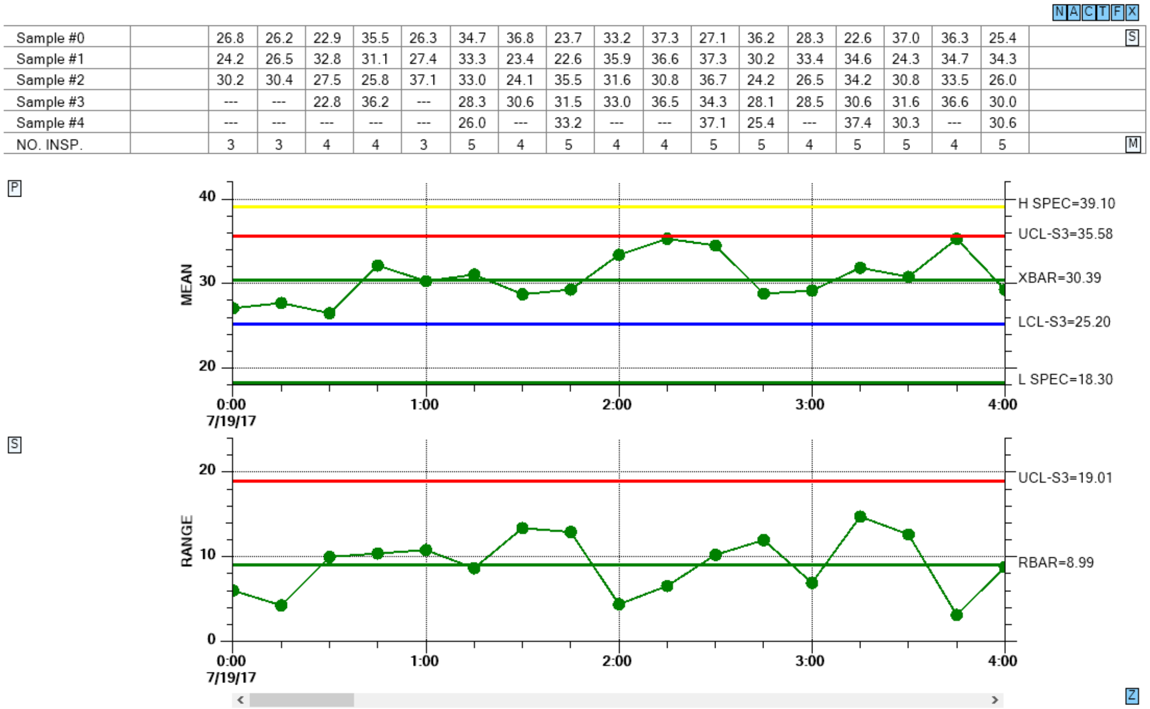
Using the Alternative N of M evaluation method, sample interval 190 does not show an alarm for the 4 out of 5 > 1-sigma test, because it is within 1-sigma.

See the NewRev3Features.ResetNofM demo for an example.

Sample Interval Update with an invalid number of samples

When you have specified one of the fixed sample size charts, a more lenient evaluation method for sample interval updates has been implemented. You can now enter less than or greater than the specified number of samples for an interval update without the software complaining. It will process the samples as if the proper number of values was entered, without introducing zero values into the sample interval. So if the specified number of samples per sample interval is 5, and 3 are entered, it will still work. It will calculate the mean and range of the sample interval using only 3 values. However, it will not adjust the control limits to take into account the incorrect number of samples, unlike what is done in the dedicated variable sample size (_VSS) charts. Also, you cannot enter in a single value for any of the control charts which require more than one value. Because then the software cannot calculate the range or sigma value, a required calculation. The programmer and user are cautioned that any alarms (or lack thereof) triggered when using this option may not be valid.

In the example below, the number of samples per sample interval varies from three to five samples. Note that the table displays as many rows of sample data as the fixed number of samples per sample interval in the chart definition, five in this case. Missing data values are designated using “...”.



Sample #0		26.8	26.2	22.9	35.5	26.3	34.7	36.8	23.7	33.2	37.3	27.1	36.2	28.3	22.6	37.0	36.3	25.4	S
Sample #1		24.2	26.5	32.8	31.1	27.4	33.3	23.4	22.6	35.9	36.6	37.3	30.2	33.4	34.6	24.3	34.7	34.3	
Sample #2		30.2	30.4	27.5	25.8	37.1	33.0	24.1	35.5	31.6	30.8	36.7	24.2	26.5	34.2	30.8	33.5	26.0	
Sample #3		---	---	22.8	36.2	---	28.3	30.6	31.5	33.0	36.5	34.3	28.1	28.5	30.6	31.6	36.6	30.0	
Sample #4		---	---	---	---	---	26.0	---	33.2	---	---	37.1	25.4	---	37.4	30.3	---	30.6	
NO. INSP.		3	3	4	4	3	5	4	5	4	4	5	5	4	5	5	4	5	M

Note that the sample data in the table has many columns where the number of samples is less than the specified 5 samples per sample interval.

If you update the sample interval with more than the specified number of samples, the software will calculate the mean, sigma, and range of the sample interval (and any other calculations in needs to) using all of the values values. It will NOT store the additional values though. The table will not show these additional values. The sample interval data is only displayed up to the original, fixed, number of samples per sample interval.

If you acquire your sample data, and find that you do not have valid, or sufficient samples for the current chart (> 1 for all but the Individual Range charts), you should skip the sample interval update for that sample interval.

See the NewRev3Features.EnhanceLimitSymbols demo for an example.

Alarm Forcing

High and low control lines can be set to explicit values, or you can have the software auto-calculate the values based on historical data. As new sample intervals are added to the chart, the new values are compared to existing control limits and the alarm conditions noted. But, you can override the alarm limit display, so that a normal condition shows as an alarm, or an alarm condition shows as normal. Obviously you can distort the meaning of the chart if you hide alarms or show alarms that the sample data does not support. But that issue is left to the programmer and end user.

SetForcedControlLimitValues

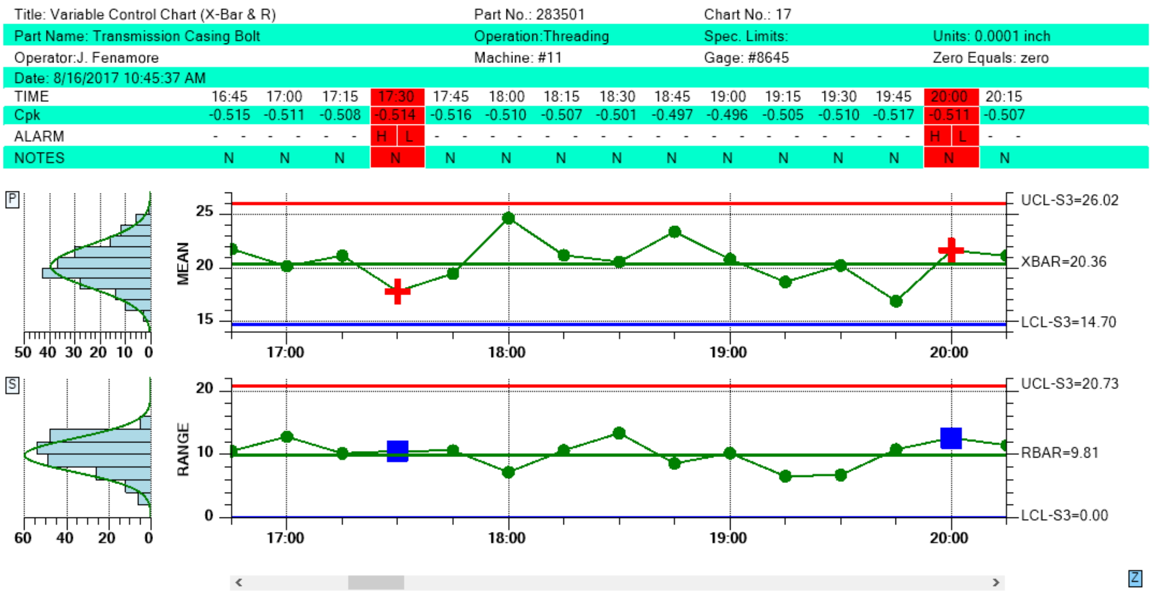
Force the current sample interval of the chart to the specified forcing value.

```
public void SetForcedControlLimitValues(int chartnum, int forcingvalue)
```

chartnum	chart number. Use SPCChartObjects.PRIMARY_CHART or SPCChartObjects.SECONDARY_CHART
forcingvalue	forcing value. Use one of forcing value constants: SPCChartForceAlarm.FORCE_LOW, SPCChartForceAlarm.FORCE_HIGH, SPCChartForceAlarm.FORCE_NORMAL

For example:

```
this.ChartData.SetForcedControlLimitValues(SPCChartObjects.PRIMARY_CHART,
SPCChartForceAlarm.FORCE_HIGH);

this.ChartData.SetForcedControlLimitValues(SPCChartObjects.SECONDARY_CHART,
SPCChartForceAlarm.FORCE_LOW);
```



Title: Variable Control Chart (X-Bar & R) Part No.: 283501 Chart No.: 17
Part Name: Transmission Casing Bolt Operation:Threading Spec. Limits: Units: 0.0001 inch
Operator:J. Fenamore Machine: #11 Gage: #8645 Zero Equals: zero
Date: 8/16/2017 10:45:37 AM

TIME	16:45	17:00	17:15	17:30	17:45	18:00	18:15	18:30	18:45	19:00	19:15	19:30	19:45	20:00	20:15
Cpk	-0.515	-0.511	-0.508	-0.514	-0.516	-0.510	-0.507	-0.501	-0.497	-0.496	-0.505	-0.510	-0.517	-0.511	-0.507
ALARM	- -	- -	- -	H L	- -	- -	- -	- -	- -	- -	- -	- -	- -	H L	- -
NOTES	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N

Sample intervals can be forced to into an alarm state, regardless of the calculated value. All of the alarms in the example above are forced and not representative of the sample data.

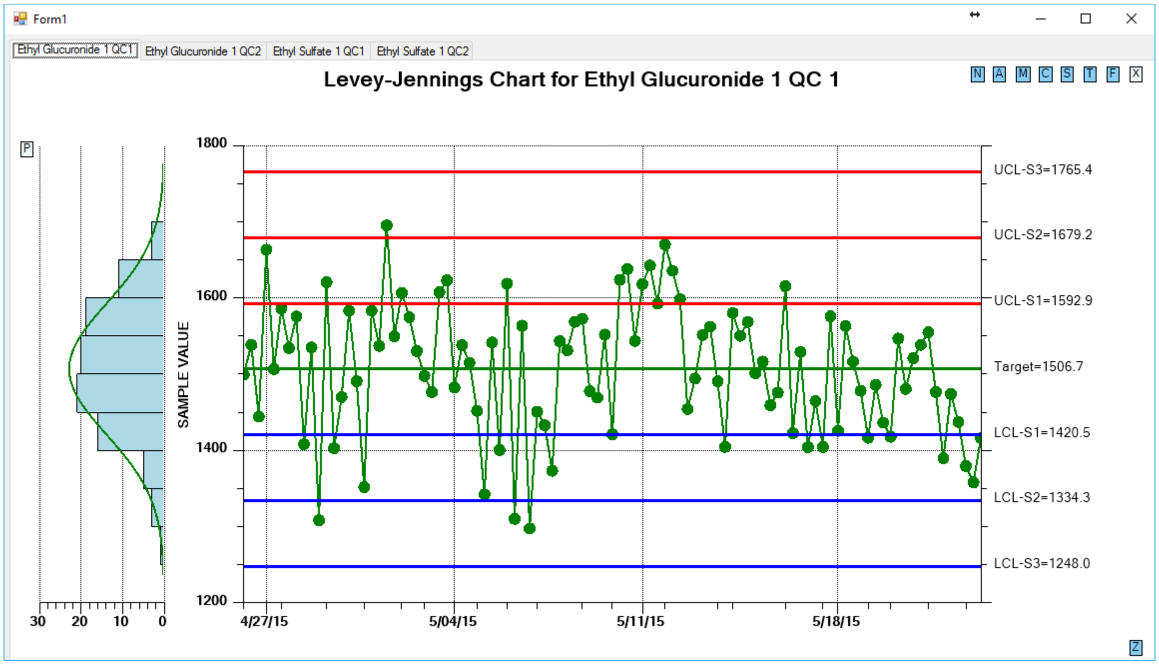
See the NewRev3Features.ForceAlarms demo for an example.

Levey-Jennings Variable Control Chart

The Levey-Jennings chart is used almost exclusively in laboratory settings. It uses a chart very similar to the Individual Range chart above, the major difference being that it only uses the Primary individual data point graph of the chart and does not include the Secondary range graph. Also, the Levey-Jennings chart uses the Westgard rules which utilizes tests involving 1-, 2- and 3- sigma control limits. The control limit calculations depart from all of the other SPC Chart types in that the target value (mean) and control limit (sigma) calculations use the overall mean and standard deviation values from the entire, charted, sample population. See the links https://en.wikipedia.org/wiki/Laboratory_quality_control and https://www.westgard.com/lesson12.htm for more information about the underlying principles of the Levey-Jennings chart.

See the example LeveyJennings for and example of how to setup this type of graph. It comes in SPCBatchVariableControlChart, SPCTimeVariableControlChart and SPCEventVariableControlChart versions. Below is an example of how to setup an event-based Levey-Jennings chart.

```
public partial class LeveyJennings1 : SPCEventVariableControlChart
  {
         ChartCalendar startTime = new ChartCalendar();
            //  SPC variable control chart type
            int charttype = SPCControlChartData.LEVEY_JENNINGS_CHART;
            // Number of samples per sub group
            int numsamplespersubgroup = 1;
            // Number of datapoints in the view
            int numdatapointsinview = 99;


      public LeveyJennings1()
            {
                   // This call is required by the Windows.Forms Form Designer.
                   InitializeComponent();
                   // Have the chart fill parent client area
                   this.Dock = DockStyle.Fill;
                   // Define and draw chart
          if (!IsDesignMode)
                     InitializeChart();

            }



      public void InitializeChart()
            {

                   // Initialize the SPCTimeVariableControlChart
                   this.InitSPCEventVariableControlChart(charttype,
                        numsamplespersubgroup, numdatapointsinview);
```

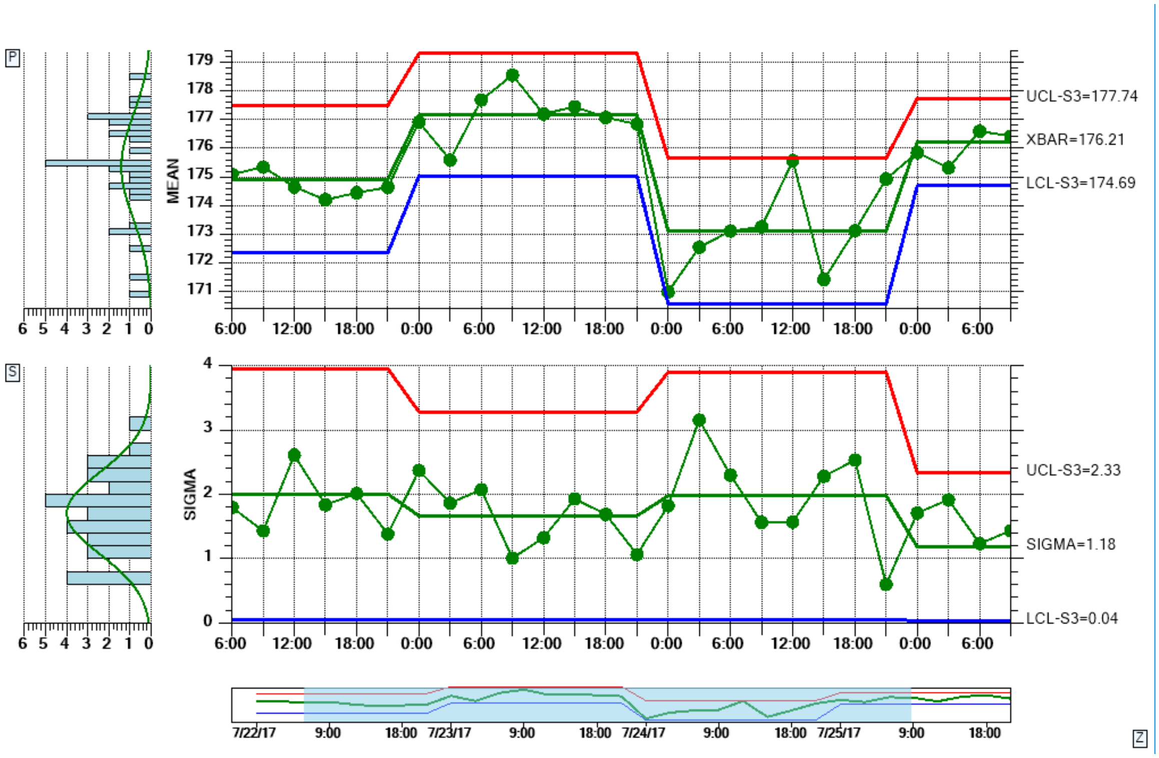


The Levey-Jennings chart only uses the Primary chart.

See the LeveyJennings example demo for an example.

Batch update with auto-calculated control limits

We added a routine (CalculateControlLimitsFromBatch) in the main SPCChartBase class, which will calculate a set of control limits based on a batch of N sample intervals. So if your first batch has 20 sample intervals, you would call that routine and you would get back the control limits specific to that set of data. Then you would set the control limits to those values, then update the chart with your sample interval data. The data would be evaluated against the control limits you just set. Then, you can take another batch of N sample intervals, calculate the limits, set the limits, and update the graph again. The new process data, and control limits will be appended to the line plots of the existing chart. The new limits will only apply to the new data values. Repeat ad infinitum.



You can auto-calculate control limits for batches of process values, and concatenate the resulting graphs together.

To make it even easier to use, we added flags to the CalculateControlLimitsFromBatch routine which can optionally set the calculated control limits, and update the chart with the supplied data, eliminating the need for the programmer to that externally. So the way this works is that each batch will display all at once, appended to the existing data. There is no incremental update of one sample interval at a time.

CalculateControlLimitsFromBatch

Calculate a set of control limits from a batch of sample interval data.

```
public void CalculateControlLimitsFromBatch(TimeArray timestamps,
List<DoubleArray> data, DoubleArray limits, bool setlimits, bool addnewsamples)
```

timestamps	The timestamps.
data	The data.
limits	The limits.
setlimits	if set to true set the control limits of the chart to the calculated limits.
addnewsamples	if set to true update the chart with the sample data. This also requires that the *setlimits* parameter is set true.

In the example code below, we use a data simulator to create the two required arrays: sampleIntervalData and timeStampData. Note that the timeStampData array is one of our TimeArray classes, while the sampleIntervalData class is a .Net generic List class of our DoubleArray class (ie. An array of DoubleArray)

```
List<DoubleArray> sampleIntervalData = new List<DoubleArray>();
TimeArray  timeStampData = new TimeArray();
DoubleArray controlLimits = new DoubleArray();
```

We then call a simple data simulator which fills out the two arrays: with values based on the calling parameters.

```
SimulateData(new ChartCalendar(2017, 7, 22), 8, 175, 5, timeStampData,
sampleIntervalData);
```

So each sample interval of the soon to be graph is represented by a time stamp value (a ChartCalendar value, which is easily derived from the .Net TimeDate class if that is what you use) in the timeStampData array, and an element in the sampleIntervalData array, where each element represents a DoubleArray of sample interval values.

```
this.CalculateControlLimitsFromBatch (timeStampData, sampleIntervalData,
controlLimits, true, true);
```

If you set the *setlimits* parameter to true, the routine automatically sets the charts control limits to the ones calculated for this batch of data. If you set the *addnewsamplesparameter* true, it automatically updates the chart with the sample interval data, using the values of *timestamps*, and *data* array parameters. It calls the ChartData.AddNewSampleRecord for each element of the *timestamps* and *data* array. So setting the last two parameters true replaces the following code in your own program:

```
if (setlimits)
{
    ChartData.SetControlLimitValues(limits.GetElements());
    if (addnewsamples)
    {
       for (int i = 0; i < timestamps.Length; i++)
       {
           DoubleArray sampleinterval = data[i];
           ChartData.AddNewSampleRecord(timestamps[i], sampleinterval);
       }
    }
}
```

5. SPC Control Data and Alarm Classes

SPCControlChartData
SPCControlLimitAlarmArgs
SPCControlLimitRecord
SPCCalculatedValueRecrod
SPCSampledValueRecord
SPCGeneralizedTableDisplay

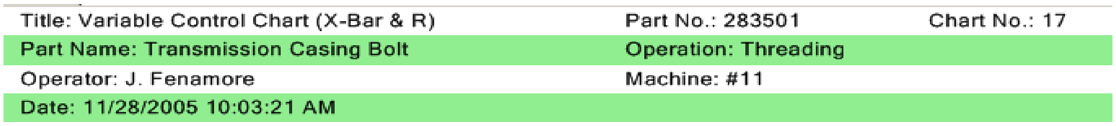
The *Variable* and *Attribute Control Chart* classes share common data and alarm classes. SPC control chart data is stored in the **SPCControlChartData** class. It holds the header information used to customize the chart table, the raw sample data used to prepare the chart, the calculated chart values used in the chart, and the SPC control limits. It contains array lists of **SPCSampledValueRecord**, **SPCControlLimitRecord** and **SPCCalculatedValueRecord** objects. The **SPCGeneralizedTableDisplay** class manages **ChartText** objects used to display data in the table portion of the SPC chart.

Class SPCControlChartData

```
ChartObj
   |
   +-- SPCControlChartData
```

The **SPCControlChartData** class is the core data storage object for all of SPC Control Chart classes. It holds all of the data plotted in the SPC chart. That includes the header information used to customize the chart table,

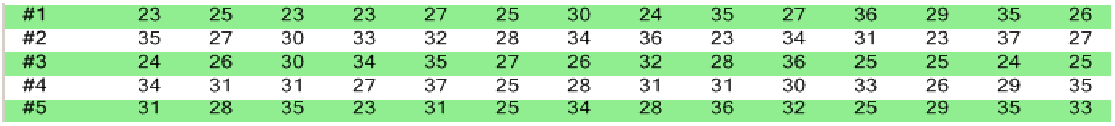
Header Information



Title: Variable Control Chart (X-Bar & R)	Part No.: 283501	Chart No.: 17
Part Name: Transmission Casing Bolt	Operation: Threading	
Operator: J. Fenamore	Machine: #11	
Date: 11/28/2005 10:03:21 AM		

the raw sample data used in the SPC calculations,

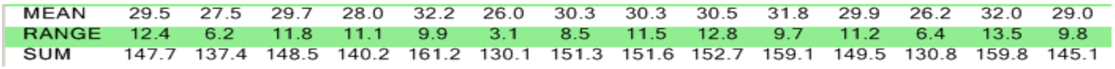
Raw Sample Data



#1	23	25	23	23	27	25	30	24	35	27	36	29	35	26
#2	35	27	30	33	32	28	34	36	23	34	31	23	37	27
#3	24	26	30	34	35	27	26	32	28	36	25	25	24	25
#4	34	31	31	27	37	25	28	31	31	30	33	26	29	35
#5	31	28	35	23	31	25	34	28	36	32	25	29	35	33

the calculated chart values used in the chart, and the SPC control limits,

Calculated Values



MEAN	29.5	27.5	29.7	28.0	32.2	26.0	30.3	30.3	30.5	31.8	29.9	26.2	32.0	29.0
RANGE	12.4	6.2	11.8	11.1	9.9	3.1	8.5	11.5	12.8	9.7	11.2	6.4	13.5	9.8
SUM	147.7	137.4	148.5	140.2	161.2	130.1	151.3	151.6	152.7	159.1	149.5	130.8	159.8	145.1

and any notes you might want to place in the record.

Notes

NOTES	N	N	N	N	N	N	N	N	N	N	N	N	N	N

There is an instance of **SPCControlChartData** in the **SPCChartBase** class. Since the **SPCChartBase** class is the base class for all of the SPC Control Charts (**SPCEventAttributeControlChart**, **SPCEventVariableControlChart**, **SPCBatchAttributeConrolChart**, **SPCBatchVariableControlChart**, **SPCTimeAttributeConrolChart**, **SPCTimeVariableControlChart**), it is accessible from those classes. The data elements of the **SPCControlChartData** class are accessible to the programmer.

SPCControlChartData Methods

The **SPCControlChartData** object is automatically created when the parent **SPCChartBase** object is created. The programmer does not need to instantiate it.

Public Static (Shared) Fields

CUSTOM_ATTRIBUTE_CONTROL_CHART	Chart type constant: Custom SPC Attribute Control Chart (unused)
CUSTOM_VARIABLE_CONTROL_CHART	Chart type constant: Custom SPC Variable Control Chart (not used)
DATALOG_FILE_ALL	Datalog flag specifying that all available items should be logged to the file.
DATALOG_FILE_BATCH_NUMBER	Datalog flag specifying that the batch number should be logged to the file.
DATALOG_FILE_CALCULATED_VALUES	Datalog flag specifying that the calculated values should be logged to the file.

DATALOG_FILE_COLUMN_HEADS	Datalog flag specifying that the column heads should be logged to the file.
DATALOG_FILE_CONTROL_LIMIT_VALUES	Datalog flag specifying that the control limit values should be logged to the file.
DATALOG_FILE_NOTES	Datalog flag specifying that the notes should be logged to the file.
DATALOG_FILE_SAMPLED_VALUES	Datalog flag specifying that the sampled values should be logged to the file.
DATALOG_FILE_TIME_STAMP	Datalog flag specifying that the time stamp should be logged to the file.
DATALOG_USER_STRING	Datalog flag specifying that the file prefix row ist NOT to be included. Using this option will make the file incompatible with the SPCControlChartData routines that read data files.
FRACTION_DEFECTIVE_PARTS_CHART	Chart type constant: Fraction Defective Parts (p-chart) Control Chart
HEADER_STRINGS_LEVEL0	SPC Chart header level constant: display no header strings.
HEADER_STRINGS_LEVEL1	SPC Chart header level constant: display minimal header strings, title, partNumber, chartNumber, dateString
HEADER_STRINGS_LEVEL2	SPC Chart header level constant: display most header strings, title, partNumber, chartNumber, partName, operation, operator, machine, dateString
HEADER_STRINGS_LEVEL3	SPC Chart header level constant: display all header strings, title, partNumber, chartNumber, partName, operation, operator, machine, specification limits, gage, unitofMeasure, zeroEqulas and dateString
INDIVIDUAL_RANGE_CHART	Chart type constant: Individual Range (Individual X) SPC Variable Control Chart
MEAN_RANGE_CHART	Chart type constant: Mean and Range (X-Bar R) SPC Variable

	Control Chart (
MEAN_SIGMA_CHART	Chart type constant: Mean and Sigma (X-Bar Sigma) SPC Variable Control Chart
MEAN_SIGMA_CHART_VSS	Chart type constant: Mean and Sigma (X-Bar Sigma) SPC Variable Control Chart with variable sample size
MEAN_VARIANCE_CHART	Chart type constant: Mean and Variance (X-Bar Variance) SPC Variable Control Chart
MEDIAN_RANGE_CHART	Chart type constant: Median and Range (Median-Range) SPC Variable Control Chart
NO_DATALOG_FILE_PREFIX	Datalog flag specifying that the file prefix row ist NOT to be included. Using this option will make the file incompatible with the SPCControlChartData routines that read data files.
NUMBER_DEFECTIVE_PARTS_CHART	Chart type constant: Number Defective Parts (np-chart) Control Chart
NUMBER_DEFECTS_CHART	Chart type constant: Number Defects (c-chart) Control Chart
NUMBER_DEFECTS_PERUNIT_CHART	Chart type constant: Number Defects per Unit (u-chart) Control Chart
PERCENT_DEFECTIVE_PARTS_CHART	Chart type constant: Percent Defective Parts (p-chart) Control Chart
SPC_PRIMARY_CONTROL_TARGET	Index of primary chart target control limit in controlLimitData array.
SPC_PRIMARY_LOWER_CONTROL_LIMIT	Index of primary chart lower control limit in controlLimitData array.
SPC_PRIMARY_UPPER_CONTROL_LIMIT	Index of primary chart upper control limit in controlLimitData array.
SPC_SECONDARY_CONTROL_TARGET	Index of secondary chart target control limit in controlLimitData array.
SPC_SECONDARY_LOWER_CONTROL_LIMIT	Index of secondary chart lower control limit in controlLimitData array.
SPC_SECONDARY_UPPER_CONTROL_LIMIT	Index of secondary chart upper control limit in controlLimitData array.

Public Static (Shared) Properties

BatchColumnHeadString	Default string used as the batch number column head in the log file. The default value is "Batch #"
DefaultAbsRangeString	Default string used to the y-axis of secondary chart of I-R charts.
DefaultDataLogFilenameRoot	Default string used as the default file name for data logging. Only used if data logging turned on and the DataLogFileOpenForWrite has not been initialized with an explicit filename. The current time is appended to the root to make it a unique filename.
DefaultHighAlarmMessageString	Default string used as the high alarm message for a low control limit.
DefaultLowAlarmMessageString	Default string used to the label the target lower control limit line of the chart.
DefaultLowControlLimitString	Default string used to the label the target low control limit line of the chart.
DefaultMeanString	Default string used to title the y-axis of mean graphs.
DefaultMedianString	Default string used to title the y-axis of median graphs.
DefaultRangeString	Default string used to title the y-axis of range graphs.
DefaultSampleValueString	Default string used to the y-axis of primary chart of I-R charts.
DefaultSigmaString	Default string used to title the y-axis of sigma graphs.
DefaultSumString	Default string used to title the sum table row.
DefaultTargetString	Default string used to the label the target control limit line of the chart.
DefaultUpperControlLimitString	Default string used to the label the target upper control limit line of the chart.
DefaultVarianceString	Default string used to title the y-axis of variance graphs.
DefaultXBarString	Default string used to title primary chart.
DefaultXString	Default string used to the primary chart of I-R charts.
NotesColumString	Default string used as the notes column head in the log file. The default value is "Notes"
SampleValueColumnString	Default string used as the sample value column head in the log file. The default value is "Sample #"

TimeStampColumnString	Default string used as the time stamp column head in the log file. The default value is "Time Stamp"

Public Static (Shared) Methods

CalcRangeBasedDecimalPos	Calculate the decimal precision used to display calculated values in the data table.

Public Instance Constructors

SPCControlChartData	Overloaded. Initializes a new instance of the SPCControlChartData class.

Public Instance Properties

AlarmStateEventEnable	Set to True to signify that any alarm should invoke the AlarmStateEventHandler.
AlarmTransitionEventEnable	Set to True to signify that any change in an alarm state should invoke the AlarmTransitionEventHandler.
ChartNumber	Set/Get data table chart number string.
ChartNumberHeader	Set/Get the header for the chartNumber field.
CurrentNumberRecords	Get the current number of records for the chart.
DatalLogEnable	Set to true to enable data logging. If a data log file has not been previously opened with DataLogFileOpenForWrite, a new data log file is created using the default name, combined with a time stamp.
DataLogCSV	The CSV (Comma Separated Value) specifyier for the logging data SPC data to a file.
DataLogFilename	The string used as the file name for data logging. Set when the DataLogFileOpenForWrite is called.
DataLogFlags	Set/Get the flags that control what items are logged to the data log file. The default has all of the optional items logged to the file. "OR" together individual data log file flags to specify the items you want logger to the file. For example: DatalogFlags = DATALOG_FILE_TIME_STAMP | DATALOG_FILE_SAMPLED_VALUES | DATALOG_FILE_CALCULATED_VALUES | DATALOG_FILE_COLUMN_HEADS. Use one of the SPCControlChartData datalog: DATALOG_FILE_BATCH_NUMBER, DATALOG_FILE_TIME_STAMP, DATALOG_FILE_SAMPLED_VALUES, DATALOG_FILE_CALCULATED_VALUES, DATALOG_FILE_CONTROL_LIMIT_VALUES

	, DATALOG_FILE_NOTES, DATALOG_FILE_COLUMN_HEADS.
DataLogUserString	The dataLogUserString is output as the second line in a datalog file, if the DATALOG_USER_STRING flag is set in dataLogFlags.
DateHeader	Set/Get the header for the dateString field.
DateString	Set/Get data table date string.
DefaultDefectRowHeaderPrefix	Set/Get the default symbol used for the row headers of the sample data items.
DefaultSampleRowHeaderPrefix	Set/Get the default symbol used for the row headers of the sample data items.
DefectiveDecimalPrecision	Set/Get the default value to use for the decimal precision used to display defective item counts, -1 = auto.
Gage	Set/Get data table gage string.
GageHeader	Set/Get the header for the gage field.
Machine	Set/Get data table machine string.
MachineHeader	Set/Get the header for the machine field.
NotesHeader	Set/Get the data table notes header string.
NotesMessage	Set/Get data table notes message string.
NotesToolTips	Set/Get the notes tool tip.
NumberSamplesValueRowHeader	Get/Set the data table number of samples row header.
NumCalculatedValues	Set/Get number of calculated values for each record in the chart.
NumRecordsPerChart	Set/Get the maximum number of records displayable at one time in the chart.
NumSampleCategories	Set/Get the number of categories in an Attribute Control chart. numSampleCategories == sampleSubgroupSize for Variable Control Charts
Operation	Set/Get data table operation string.
OperationHeader	Set/Get the header for the operation field.
OperatorHeader	Set/Get the header for the theOperator field.
PartName	Set/Get data table part name string.
PartNameHeader	Set/Get the header for the partName field.
PartNumber	Set/Get data table part number string.
PartNumberHeader	Set/Get the header for the partNumber field.
PrimaryCalculatedVariableIndex	Set/Get index in the calculatedValues array for the primary calculated value data.
SampleSubgroupSize	Set/Get the number of samples in a sample sub group for a Variable Control chart.
SecondaryCalculatedVariableIndex	Set/Get index in the calculatedValues array for the

	secondary calculated value data.
SPCChartType	Set/Get the control chart type: use one of the SPCControlChartData chart type constants: MEAN_RANGE_CHART, MEDIAN_RANGE_CHART, MEAN_SIGMA_CHART, MEAN_SIGMA_CHART_VSS, MEAN_VARIANCE_CHART, INDIVIDUAL_RANGE_CHART, EWMA_CHART, MA_CHART, LEVEY_JENNINGS_CHART, MAMR_CHART, MAMS_CHART TABCUSUM_CHART, CUSTOM_ATTRIBUTE_CONTROL_CHART, PERCENT_DEFECTIVE_PARTS_CHART, FRACTION_DEFECTIVE_PARTS_CHART, NUMBER_DEFECTIVE_PARTS_CHART, NUMBER_DEFECTS_CHART, NUMBER_DEFECTS_PERUNIT_CHART, NUMBER_DEFECTS_PER_MILLION_CHART.
SpecificationLimits	Set/Get data table specification limits string.
SpecificationLimitsHeader	Set/Get the header for the specificationLimits field.
TheOperator	Set/Get data table operator string.
TimeStamp	Set/Get the time stamp for the most recent sample data added to the class.
TimeValueRowHeader	The data table time value row header.
Title	Set/Get data table title string.
TitleHeader	Set/Get the header for the title field.
UnitOfMeasure	Set/Get data table unit of measure string.
UnitOfMeasureHeader	Set/Get the header for the unit of measure field.
ZeroEquals	Set/Get data table zero equals string.
ZeroEqualsHeader	Set/Get the header for the zeroEqulas field.

Public Instance Methods

AddNewSampleRecord	Overloaded. Add a new sample record with notes to a time-based SPC chart that plots variable control limits.
AppendCurrentRecordValuesToDataLog	This methods will create a text file and append the current SPC data record to that file in a CSV (Comma Separated Value) format. A CSV file can be read by popular spreadsheet and word processing programs. Some localization for different operating systems and locales can be handled by the modifying the default csv (CSV) object.
Clone	Returns an object that is a clone of this

	object.
ControlLimitInitialized	Returns true if the control limit record at the index is initiated.
Copy	Overloaded. Copies the source object.
Copy (inherited from **ChartObj**)	Overloaded. Copies the source object.
DataLogFileOpenForWrite	Overloaded. This methods will create a text file and output the SPC chart data to that file in a CSV (Comma Separated Value) format. A CSV file can be read by popular spreadsheet and word processing programs. Some localization for different operating systems and locales can be handled by the modifying the default csv (CSV) object. Uses the dataLogFlags property as the guide to reading the values of the file, so that property must be properly initialized for the data file being read.
Equals (inherited from **Object**)	Determines whether the specified Object is equal to the current **Object**.
ErrorCheck (inherited from **ChartObj**)	Throws an exception if an error exists in the error chain.
ExcludeRecordFromControlLimitCalculations	Exclude the specified record from the SOC control limit calculations.
GetBatchNumberValue	Get the group number value at the specified index.
GetCalculatedValue	Get a calcualted value at a specific row (index) and column (time).
GetCalculatedValueRecord	Get the calculated value record at the specified index.
GetChartObjIDCntr (inherited from **ChartObj**)	Returns the current value of the chartObjIDCntr static counter.
GetChartObjType (inherited from **ChartObj**)	Returns the chart object type.
GetControlLimit	Get the value of a specific SPC chart limit.
GetControlLimitRecord	Get the control limit record at the specified index.
GetControlLimitString	Get the text for a specific SPC chart limit.
GetControlLimitText	Get the control limit text at the specified index.
GetControlLimitValue	Get a control limit value (for charts with variable control limits) at a specific row (index) and column (time).
GetHashCode (inherited from **Object**)	Serves as a hash function for a particular type, suitable for use in hashing algorithms and data structures like a hash table.

GetNotesString	Get the notes string at the specified index.
GetNumberOfSamplesPerSubgroup	Get the number of samples per subgroup value at the specified index.
GetPrimaryControlLimits	Overloaded. Using the current sampled data values, high, target and low control limits are calculated for the primary chart using an algorithm appropriate to the SPC chart type.
GetSampledValue	Get a sampled value at a specific row (index) and column (time).
GetSampleRowHeaderString	Get data table row header for the sampled (or category) item.
GetSecondaryControlLimits	Overloaded. Using the current sampled data values, high, target and low control limits are calculated for the primary chart using an algorithm appropriate to the SPC chart type.
GetThisChartObjID (inherited from **ChartObj**)	Returns the chartObjID value for this specific object.
GetTimeValue	Get the time stamp value at the specified index.
GetType (inherited from **Object**)	Gets the Type of the current instance.
GetYAxisTitle	Get the y-axis title or a specific index, based description of the item in the SPCCalculatedValueRecord or SPCSampledValueRecord record.
IsControlLimit	Returns true if the control limit record at the index is initiated.
OutputAllValuesToDataLog	Overloaded. This methods will create a text file and output all of the current SPC data records to that file in a CSV (Comma Separated Value) format. A CSV file can be read by popular spreadsheet and word processing programs. Some localization for different operating systems and locales can be handled by the modifying the default csv (CSV) object.
ReadAllValuesFromFile	Overloaded. This methods will read a text file of SPC data records organized in a CSV (Comma Separated Value) format. A CSV file can be read by popular spreadsheet and word processing programs. Some localization for different operating systems and locales can be handled by the modifying the default csv (CSV) object.
ResetSPCChartData	Reset the history buffers of all of the SPC data objects.

Save	Overloaded. This methods will create a text file and output the SPC chart data to that file in a CSV (Comma Separated Value) format. A CSV file can be read by popular spreadsheet and word processing programs. Some localization for different operating systems and locales can be handled by the modifying the default csv (CSV) object.
SetControlLimitString	Set the SPC text for a specific SPC chart limit.
SetControlLimitStrings	Set the SPC control limit text for an SPC control chart.
SetControlLimitValue	Set the SPC value of a specific SPC chart limit.
SetControlLimitValues	Set the SPC control limit values for an SPC control chart.
SetSampleRowHeaderString	Set data table row header for the sampled (or category) item.
SimulateDefectRecord	Overloaded. Simulates a defect measurement for a SPC Attribute Control chart with a specified mean. Used with NUMBER_DEFECTS_CHART and NUMBER_DEFECTS_PERUNIT_CHART charts.
SimulateMeasurementRecord	Overloaded. Simulates a sample measurement for a SPC Variable Control chart with a specified mean value.
SortAlarmObjectsByValue	This method sorts the objects in the controlLimitValues array in the ascending value of their alarm value.
ToString (inherited from **Object**)	Returns a String that represents the current Object.
TransitionEventCondition	Returns true if an alarm transition has taken place.

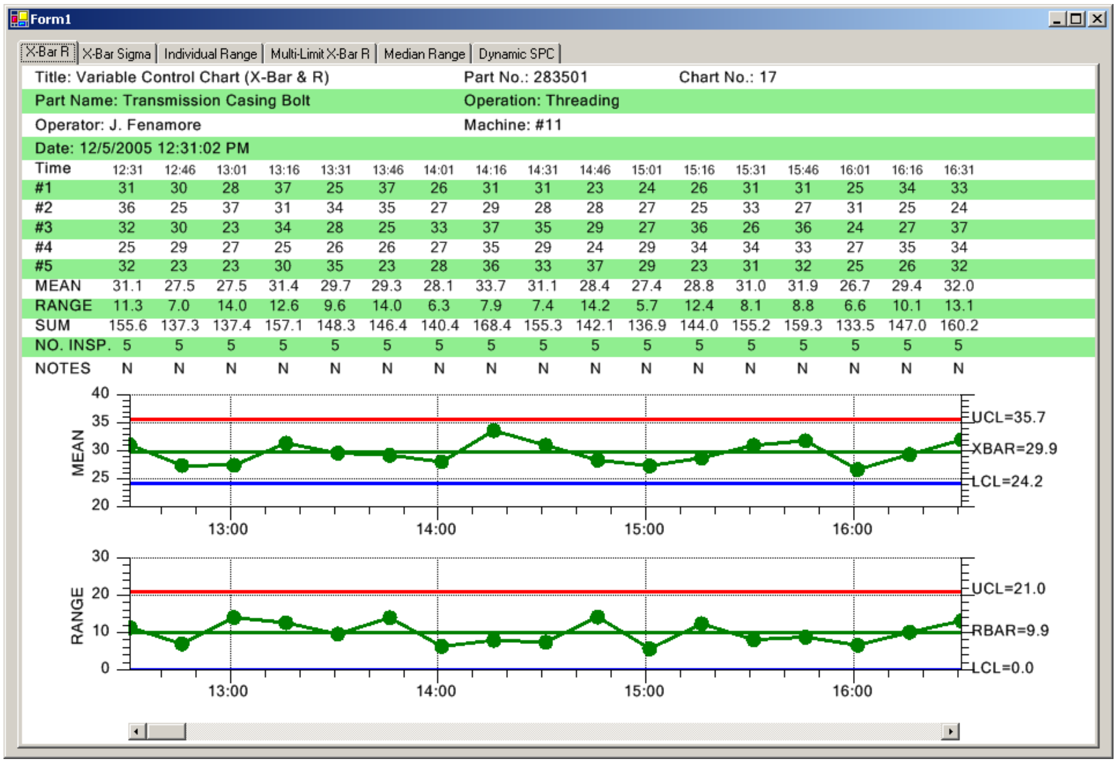
Public Instance Events

AlarmStateEventHandler	Delegate for nodification each time the check of an process variable produces an alarm state condition.
AlarmTransitionEventHandler	Delegate for notification each time the check of an process variable produces a change of state in alarm state condition.

Initializing the SPCControlChartData Class

The control charts **InitSPC**… method call initializes the **SPCControlChartData** object. This establishes the SPC chart type, how many samples per subgroup there are, and how many **SPCSampledValueRecord** objects are stored internal to the **SPCControlChartData** to handle the sampled data.

The table strings used to customize the first section of the chart should be set after the chart **InitSPC**… call, but before the **RebuildChartUsingCurrentData** call. The example below is from the **TimeVariableControlCharts.XBarRChart** example program.



Title: Variable Control Chart (X-Bar & R) Part No.: 283501 Chart No.: 17

Part Name: Transmission Casing Bolt Operation: Threading

Operator: J. Fenamore Machine: #11

Date: 12/5/2005 12:31:02 PM

Time	12:31	12:46	13:01	13:16	13:31	13:46	14:01	14:16	14:31	14:46	15:01	15:16	15:31	15:46	16:01	16:16	16:31
#1	31	30	28	37	25	37	26	31	31	23	24	26	31	31	25	34	33
#2	36	25	37	31	34	35	27	29	28	28	27	25	33	27	31	25	24
#3	32	30	23	34	28	25	33	37	35	29	27	36	26	36	24	27	37
#4	25	29	27	25	26	26	27	35	29	24	29	34	34	33	27	35	34
#5	32	23	23	30	35	23	28	36	33	37	29	23	31	32	25	26	32
MEAN	31.1	27.5	27.5	31.4	29.7	29.3	28.1	33.7	31.1	28.4	27.4	28.8	31.0	31.9	26.7	29.4	32.0
RANGE	11.3	7.0	14.0	12.6	9.6	14.0	6.3	7.9	7.4	14.2	5.7	12.4	8.1	8.8	6.6	10.1	13.1
SUM	155.6	137.3	137.4	157.1	148.3	146.4	140.4	168.4	155.3	142.1	136.9	144.0	155.2	159.3	133.5	147.0	160.2
NO. INSP.	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
NOTES	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N

[C#]

```
//  SPC variable control chart type
int charttype = SPCControlChartData.INDIVIDUAL_RANGE_CHART;
// Number of samples per sub group
int numsamplespersubgroup = 1;
// Number of data points  in the view
int numdatapointsinview = 17;
// The time increment between adjacent subgroups
int sampleincrement = 30;

public IndividualRangeChart()
{
    // This call is required by the Windows.Forms Form Designer.
    InitializeComponent();
    // Have the chart fill parent client area
    this.Dock = DockStyle.Fill;
    // Define and draw chart
    InitializeChart();
}

public  void InitializeChart()
{
//  if (this.IsDesignMode) return;
    // This call is required by the Windows.Forms Form Designer.
    InitializeComponent();
    // Initialize the SPCTimeVariableControlChart
    this.InitSPCTimeVariableControlChart(charttype,
        numsamplespersubgroup, numdatapointsinview, sampleincrement);

    // Set the strings used in the header section of the table
    this.ChartData.Title = "Variable Control Chart (X-Bar & R)";
    this.ChartData.PartNumber = "283501";
    this.ChartData.ChartNumber="17";
    this.ChartData.PartName= "Transmission Casing Bolt";
    this.ChartData.Operation = "Threading";
    this.ChartData.SpecificationLimits="";
    this.ChartData.TheOperator="J. Fenamore";
    this.ChartData.Machine="#11";
    this.ChartData.Gage="#8645";
    this.ChartData.UnitOfMeasure = "0.0001 inch";
    this.ChartData.ZeroEquals="zero";
    this.ChartData.DateString = DateTime.Now.ToString();
    this.ChartData.NotesMessage = "Control limits prepared May 10";
    this.ChartData.NotesHeader = "NOTES"; // row header
    this.HeaderStringsLevel = SPCControlChartData.HEADER_STRINGS_LEVEL1;
    .
    .
    .
    // Rebuild the chart using the current data and settings
    this.RebuildChartUsingCurrentData();

}
```

[VB]

```
'  SPC variable control chart type
Private charttype As Integer = SPCControlChartData.MEAN_RANGE_CHART
' Number of samples per sub group
Private numsamplespersubgroup As Integer = 5
' Number of data points  in the view
Private numdatapointsinview As Integer = 17
' The time increment between adjacent subgroups
Private timeincrementminutes As Integer = 15

#Region " Windows Form Designer generated code "

Public Sub New()
    .
    .
    .
  InitializeChart()
End Sub

#End Region


Public Sub InitializeChart()
    ' Initialize the SPCTimeVariableControlChart
    Me.InitSPCTimeVariableControlChart(charttype, _
           numsamplespersubgroup, numdatapointsinview, timeincrementminutes)


    ' Set the strings used in the header section of the table
    Me.ChartData.Title = "Variable Control Chart (Median Range)"
    Me.ChartData.PartNumber = "283501"
    Me.ChartData.ChartNumber = "17"
    Me.ChartData.PartName = "Transmission Casing Bolt"
    Me.ChartData.Operation = "Threading"
    Me.ChartData.SpecificationLimits = ""
    Me.ChartData.TheOperator = "J. Fenamore"
    Me.ChartData.Machine = "#11"
    Me.ChartData.Gage = "#8645"
    Me.ChartData.UnitOfMeasure = "0.0001 inch"
    Me.ChartData.ZeroEquals = "zero"
    Me.ChartData.DateString = DateTime.Now.ToString()
    Me.ChartData.NotesMessage = "Control limits prepared May 10"
    Me.ChartData.NotesHeader = "NOTES" ' row header
    Me.HeaderStringsLevel = SPCControlChartData.HEADER_STRINGS_LEVEL1
    .
    .
    .
    Me.RebuildChartUsingCurrentData()
End Sub 'InitializeChart
```

Update the sampled data with your measured values using one of the **AddNewSampleRecord** methods.

Method AddNewSampleRecord

This method adds a new sample record to the SPC chart. While *both variable control charts* and *attribute control charts* share the same **ChartData AddNewSampleRecord** methods, the meaning of the data in the *samples* array varies, depending on the chart type. See the sections below: *Adding New Sample Records for Variable Control Charts*, and *Adding New Sample Records for Attribute Control Charts*.

[VB]

```
Overloads Public Sub AddNewSampleRecord( _
   ByVal timestamp As ChartCalendar, _
   ByVal samples As DoubleArray, _
   ByVal notes As String _
)
```

[C#]

```
public void AddNewSampleRecord(
   ChartCalendar timestamp,
   DoubleArray samples,
   string notes
);
```

Parameters

timestamp

Time stamp for the current sample record.

samples

Array of new sample values.

notes

A string specifying any notes associated with this sample subgroup *controllimits*

Array of control limits, one for each control limits (low, target, and high)

There are many other overloaded versions of **AddNewSampleRecord**. Use the one most appropriate to your application.

Adding New Sample Records for Variable Control Charts (Fixed Subgroup Sample Size).

Applies to variable control charts of type: MEAN_RANGE_CHART, MEDIAN_RANGE_CHART, INDIVIDUAL_RANGE_CHART, MEAN_SIGMA_CHART, INDIVIDUAL_RANGE_CHART, EWMA_CHART, LEVEY_JENNINGS_CHART, MA_CHART, MAMR_CHART, MAMS_CHART, TABCUSUM_CHART.

In variable control charts, each data value in the *samples* array represents a specific sample in the sample subgroup. In X-Bar R, X-Bar Sigma, and Median-Range charts, where the sample subgroup size is some fraction of the total production level, there is one value in the *samples* array for each measurement sample in the sample subgroup interval. If the production level is sixty items per hour, and the sample size is five items per hour, then the graph would be updated once an hour with five items in the *samples* array.

[C#]

```
DoubleArray samples = new DoubleArray(5);
//  ChartCalendar initialized with current  time by default
ChartCalendar timestamp = new ChartCalendar();
// Place sample values in array
samples[0] = 0.121;     // First of five samples
samples[1] = 0.212;     // Second of five samples
samples[2] = 0.322;     // Third of five samples
samples[3] = 0.021;     // Fourth of five samples
samples[4] = 0.133;     // Fifth of five samples

// Add the new sample subgroup to the chart
this.ChartData.AddNewSampleRecord(timestamp, samples);
```

[VB]

```
Dim samples As DoubleArray = New DoubleArray(5)
'  ChartCalendar initialized with current  time by default
Dim timestamp As ChartCalendar = New ChartCalendar()
' Place sample values in array
samples(0) = 0.121    ' First of five samples
samples(1) = 0.212    ' Second of five samples
samples(2) = 0.322    ' Third of five samples
samples(3) = 0.021    ' Fourth of five samples
samples(4) = 0.133    ' Fifth of five samples

' Add the new sample subgroup to the chart
 Me.ChartData.AddNewSampleRecord(timestamp, samples)
```

In an Individual-Range chart, which by definition samples 100% of the production level, the *samples* array would only have one value for each update. If the production level is sixty items per hour, with 100% sampling, the graph would be updated once a minute, with a single value in the *samples* array.

Adding New Sample Records to a X-Bar Sigma Chart (Variable Subgroup Sample Size)

Applies to variable control charts of type: MEAN_SIGMA_CHART_VSS

The X-Bar Sigma chart also comes in a version where variable sample sizes are permitted, As in the standard variable control charts, there is one value in the *samples* array for each measurement sample in the sample subgroup interval. The difference is that the length of the *samples* array can change from update to update. It is critically import that the size of the samples array exactly matches the number of samples in the current subgroup

[C#]

```
// GetCurrentSampleSubgroupSize is a fictional method that gets the
// current number of samples in the  sample subgroup. The value of N
// can vary from sample interval to sample interval. You must have a
// valid sample value for each element.

N = GetCurrentSampleSubgroupSize();

// Size array exactly to a length of N
DoubleArray samples = new DoubleArray(N);
//  ChartCalendar initialized with current  time by default
ChartCalendar timestamp = new ChartCalendar();

// Place sample values in array
// You must have a valid sample value for each element of the array size 0..N-1
samples[0] = 0.121;     // First of five samples
samples[1] = 0.212;     // Second of five samples
.
.
.
samples[N-1] = 0.133;       // Last of the samples in the sample subgroup

// Add the new sample subgroup to the chart
this.ChartData.AddNewSampleRecord(timestamp, samples);
```

[VB]

```
‘ GetCurrentSampleSubgroupSize is a fictional method that gets the
‘ current number of samples in the  sample subgroup. The value of N
‘ can vary from sample interval to sample interval. You must have a
‘ valid sample value for each element.

N = GetCurrentSampleSubgroupSize()

‘ Size array exactly to a length of N
Dim samples As DoubleArray = New DoubleArray(N)
'  ChartCalendar initialized with current  time by default
Dim timestamp As ChartCalendar = New ChartCalendar()
' Place sample values in array
samples(0) = 0.121    ' First of five samples
samples(1) = 0.212    ' Second of five samples
```

```
.
.
.
samples(N-1) = 0.133    ' Last of the samples in the sample subgroup

' Add the new sample subgroup to the chart
 Me.ChartData.AddNewSampleRecord(timestamp, samples)
```

Adding New Sample Records for Attribute Control

Updating p- and np-charts (Fixed Sample Subgroup Size)

p-chart = FRACTION_DEFECTIVE_PARTS_CHART
or
PERCENT_DEFECTIVE_PARTS_CHART

np-chart = NUMBER_DEFECTIVE_PARTS_CHART

In attribute control charts, the meaning of the data in the *samples* array varies, depending on whether the attribute control chart measures the number of defective parts (p-, and np-charts), or the total number of defects (u- and c-charts). The major anomaly is that while the p- and np-charts plot the fraction or number of defective parts, the table portion of the chart can display defect counts for any number of defect categories (i.e. paint scratches, dents, burrs, etc.). It is critical to understand that total number of defects, i.e. the sum of the items in the defect categories for a give sample subgroup, do NOT have to add up to the number of defective parts for the sample subgroup. Every defective part not only can have one or more defects, it can have multiple defects of the same defect category. The total number of defects for a sample subgroup will always be equal to or greater than the number of defective parts. When using p- and np-charts that display defect category counts as part of the table, where N is the *numcategories* parameter in the **InitSPCEventAttributeControlChart**, **InitSPCTimeAttributeControlChart** or **InitSPCBatchAttributeControlChart** initialization call, the first N (0.. N-1) elements of the *samples* array holds the defect count for each category. The (N+1)th (or element N in the array) element of the *samples* array holds the total defective parts count. For example, if you initialized the chart with a *numcategories* parameter to five, signifying that you had five defect categories, you would use a *samples* array sized to six, as in the code below:

[C#]

```
DoubleArray samples = new DoubleArray(6);
//  ChartCalendar initialized with current  time by default
ChartCalendar timestamp = new ChartCalendar();
// Place sample values in array
samples[0] = 3;     // Number of defects for defect category #1
samples[1] = 0;     // Number of defects for defect category #2
samples[2] = 4;     // Number of defects for defect category #3
samples[3] = 2;     // Number of defects for defect category #4
samples[4] = 3;     // Number of defects for defect category #5

samples[5] = 4;    // TOTAL number of defective parts in the sample
```

```
// Add the new sample subgroup to the chart
this.ChartData.AddNewSampleRecord(timestamp, samples);
```

[VB]

```
Dim samples As DoubleArray = New DoubleArray(6)
'  ChartCalendar initialized with current  time by default
Dim timestamp As ChartCalendar = New ChartCalendar()
' Place sample values in array
samples(0) = 3      ‘ Number of defects for defect category #1
samples(1) = 0      ‘ Number of defects for defect category #2
samples(2) = 4      ‘ Number of defects for defect category #3
samples(3) = 2      ‘ Number of defects for defect category #4
samples(4) = 3      ‘ Number of defects for defect category #5

samples(5) = 4     ‘ TOTAL number of defective parts in the sample

' Add the new sample subgroup to the chart
 Me.ChartData.AddNewSampleRecord(timestamp, samples)
```

Our example programs obscure this a bit, because we use a special method to simulate defect data for n- and np-charts. The code below is extracted from our TimeAttributeControlCharts.NumberDefectivePartsControlChart example program.

[C#]

```
DoubleArray samples = this.ChartData.SimulateDefectRecord(50 * 0.134,
        SPCControlChartData.NUMBER_DEFECTIVE_PARTS_CHART);
// Add new sample record
this.ChartData.AddNewSampleRecord( timestamp, samples);
```

[VB]

```
Dim samples As DoubleArray = Me.ChartData.SimulateDefectRecord(50 * 0.134, _
    SPCControlChartData.NUMBER_DEFECTIVE_PARTS_CHART)
' Add new sample record
Me.ChartData.AddNewSampleRecord(timestamp, samples)
```

This particular overload for **ChartData.SimulateDefectRecord** knows that since it is a NUMBER_DEFECTIVE_PARTS_CHART chart (np-chart), and since the **ChartData** object was setup with five categories in the **InitSPCTimeAttributeControlChart** call, that is should return a **DoubleArray** with (5 + 1 = 6) elements. The first five elements representing simulated defect counts for the five defect categories, and the sixth element the simulated defective parts count. The defect category count data of the *samples* array is only used in the table part of the display; the defect category counts play NO role in the actual SPC chart. The only value plotted in the SPC chart is the last element in the *samples* array, the defective parts count for the sample subgroup.

Updating p-charts (Variable Sample Subgroup Size)

p-chart = FRACTION_DEFECTIVE_PARTS_CHART_VSS

or
PERCENT_DEFECTIVE_PARTS_CHART_VSS

First, you must read the previous section (**Updating p-charts (Fixed Sample Subgroup Size**) and understand it**.** Because in the case of the p-chart variable sample subgroup case, filling out that array is EXACTLY the same as the fixed sample subgroup case. The number of defects in each defect category go into the first N elements (element 0..N-1) of the samples array. The total number of defective parts go into last (element N) of the samples array. Specify the size of the sample subgroup associated with a given update using the **ChartData.SampleSubgroupSize_VSS** property.

[C#]

```
DoubleArray samples = this.ChartData.SimulateDefectRecord(50 * 0.134,
       SPCControlChartData.NUMBER_DEFECTIVE_PARTS_CHART);

// Randomize the sample subgroup size to some value less than the maximum
// value entered in the call to InitSPCTimeAttributeControlChart,
// and set the charts ChartData.SampleSubgroupSize_VSS property with
//  this value immediately prior to the AddNewSampleRecord call.
this.ChartData.SampleSubgroupSize_VSS =
    numsamplespersubgroup - (int)(25 * ChartSupport.GetRandomDouble());

// Add new sample record
this.ChartData.AddNewSampleRecord( timestamp, samples);
```

[VB]

```
Dim samples As DoubleArray = Me.ChartData.SimulateDefectRecord(50 * 0.134, _
    SPCControlChartData.NUMBER_DEFECTIVE_PARTS_CHART)

‘ Randomize the sample subgroup size to some value less than the maximum
‘ value entered in the call to InitSPCTimeAttributeControlChart,
‘ and set the charts ChartData.SampleSubgroupSize_VSS property with
‘  this value immediately prior to the AddNewSampleRecord call.
Me.ChartData.SampleSubgroupSize_VSS = _
    numsamplespersubgroup - (25 * ChartSupport.GetRandomDouble())


' Add new sample record
Me.ChartData.AddNewSampleRecord(timestamp, samples)
```

Updating c- and u-charts (Fixed Sample Subgroup Size)

c-chart = NUMBER_DEFECTS_CHART

u-chart = NUMBER_DEFECTS_PERUNIT_CHART

In c- and u-charts the number of defective parts is of no consequence. The only thing tracked is the number of defects. Therefore, there is no extra array element tacked onto the end of the *samples* array. Each element of the *samples* array represents the total number of defects for a given defect category. If the *numcategories* parameter in the **InitSPCEventAttributeControlChart, InitSPCTimeAttributeControlChart** or

InitSPCBatchAttributeControlChart is initialized to five, the total number of elements in the *samples* array should be five. For example:

[C#]

```
DoubleArray samples = new DoubleArray(5);
//  ChartCalendar initialized with current  time by default
ChartCalendar timestamp = new ChartCalendar();
// Place sample values in array
samples[0] = 3;     // Number of defects for defect category #1
samples[1] = 0;     // Number of defects for defect category #2
samples[2] = 4;     // Number of defects for defect category #3
samples[3] = 2;     // Number of defects for defect category #4
samples[4] = 3;     // Number of defects for defect category #5

// Add the new sample subgroup to the chart
this.ChartData.AddNewSampleRecord(timestamp, samples);
```

[VB]

```
Dim samples As DoubleArray = New DoubleArray(5)
'  ChartCalendar initialized with current  time by default
Dim timestamp As ChartCalendar = New ChartCalendar()
' Place sample values in array
samples(0) = 3      ` Number of defects for defect category #1
samples(1) = 0      ` Number of defects for defect category #2
samples(2) = 4      ` Number of defects for defect category #3
samples(3) = 2      ` Number of defects for defect category #4
samples(4) = 3      ` Number of defects for defect category #5

' Add the new sample subgroup to the chart
 Me.ChartData.AddNewSampleRecord(timestamp, samples)
```

Updating u-charts (Variable Sample Subgroup Size)

u-chart =   NUMBER_DEFECTS_PERUNIT_CHART_VSS

First, you must read the previous section (**Updating u-charts Fixed Sample Subgroup Size**) and understand it**.** Because in the case of the u-chart variable sample subgroup case, filling out that array is EXACTLY the same as the fixed sample subgroup case. The number of defects in each defect category go into the first N elements (element 0..N-1) of the samples array. Specify the size of the sample subgroup associated with a given update using the **ChartData.SampleSubgroupSize_VSS** property.

[C#]

```
DoubleArray samples = new DoubleArray(5);
//  ChartCalendar initialized with current  time by default
ChartCalendar timestamp = new ChartCalendar();
// Place sample values in array
samples[0] = 3;     // Number of defects for defect category #1
samples[1] = 0;     // Number of defects for defect category #2
samples[2] = 4;     // Number of defects for defect category #3
samples[3] = 2;     // Number of defects for defect category #4
samples[4] = 3;     // Number of defects for defect category #5

// Randomize the sample subgroup size to some value less than the maximum
// value entered in the call to InitSPCTimeAttributeControlChart,
// and set the charts ChartData.SampleSubgroupSize_VSS property with
//  this value immediately prior to the AddNewSampleRecord call.
this.ChartData.SampleSubgroupSize_VSS =
    numsamplespersubgroup - (int)(25 * ChartSupport.GetRandomDouble());

// Add the new sample subgroup to the chart
this.ChartData.AddNewSampleRecord(timestamp, samples);
```

[VB]

```
Dim samples As DoubleArray = New DoubleArray(5)
'  ChartCalendar initialized with current  time by default
Dim timestamp As ChartCalendar = New ChartCalendar()
' Place sample values in array
samples(0) = 3      ‘ Number of defects for defect category #1
samples(1) = 0      ‘ Number of defects for defect category #2
samples(2) = 4      ‘ Number of defects for defect category #3
samples(3) = 2      ‘ Number of defects for defect category #4
samples(4) = 3      ‘ Number of defects for defect category #5

‘ Randomize the sample subgroup size to some value less than the maximum
‘ value entered in the call to InitSPCTimeAttributeControlChart,
‘ and set the charts ChartData.SampleSubgroupSize_VSS property with
‘  this value immediately prior to the AddNewSampleRecord call.
Me.ChartData.SampleSubgroupSize_VSS = _
    numsamplespersubgroup - (25 * ChartSupport.GetRandomDouble())

' Add the new sample subgroup to the chart
 Me.ChartData.AddNewSampleRecord(timestamp, samples)
```

While the table portion of the display can display defect data broken down into categories, only the sum of the defects for a given sample subgroup is used in creating the actual SPC chart. Note that the code below, extracted from the TimeAttributeControlCharts.NumberDefectsControlChart example, uses a different **ChartData.SimulateDefectRecord** method to simulate the defect data.

[C#]

```
// Simulate sample record
DoubleArray samples = this.ChartData.SimulateDefectRecord(19.85/5);
// Add a sample record
this.ChartData.AddNewSampleRecord( timestamp, samples);
```

[VB]

```
' Simulate sample record
  Dim samples As DoubleArray = Me.ChartData.SimulateDefectRecord((19.85 / 5))
   ' Add a sample record
 Me.ChartData.AddNewSampleRecord(timestamp, samples)
```

Other AddNewSampleRecord Methods

Add a new sample record to a time-based SPC chart.

public void AddNewSampleRecord(ChartCalendar,DoubleArray);

Add a new sample record with notes to a time-based SPC chart.

public void AddNewSampleRecord(ChartCalendar,DoubleArray,string);

Add a new sample record to a batch-based SPC chart.

public void AddNewSampleRecord(DoubleArray);

Add a new sample record, with notes, to a batch-based SPC chart.

public void AddNewSampleRecord(DoubleArray,string);

Add a new sample record to a numeric-based SPC chart.

public void AddNewSampleRecord(double,ChartCalendar,DoubleArray,DoubleArray,string);

Add a new sample record, with notes, to a numeric-based SPC chart .

public void AddNewSampleRecord(double,ChartCalendar,DoubleArray,string);

Add a new sample record, with notes, to a batch-based SPC chart.

public void AddNewSampleRecord(double,DoubleArray);

Add a new sample record, with notes, to a batch-based SPC chart.

public void AddNewSampleRecord(double,DoubleArray,string);
In addition to these, there are versions that pass in an additional **DoubleArray** that pass in the current value of variable control limits, if used. See the QCSPCChartNet6.chm compiled help file, under com.quinn-curtis.spcchartnet6 | **SPCControlChartData.AddNewSampleRecord.**

If the **AddNewSampleRecord** overload does not have an explicit **ChartCalendar** time stamp parameter, as in the case several of the overloaded methods, the current time as stored in the system clock is used as the time stamp.

Question - How do you initialize the ChartCalendar object with your own time. Answer - Just use one of the many ChartCalendar constructors. See the QCChart2DNet6.chm compiled help file.

> This constructor creates a new **ChartCalendar** object using the specified **DateTime** value.
>
> public ChartCalendar(DateTime);
>
> This constructor creates a new **ChartCalendar** object using the specified year, month and day.
>
> public ChartCalendar(int,int,int);

This constructor creates a new **ChartCalendar** object using the specified year, month, day, hour, minute and second.

public ChartCalendar(int,int,int,int,int,int);

This constructor creates a new **ChartCalendar** object using the specified year, month, day, hour, minute, second and milliseconds.

public ChartCalendar(int,int,int,int,int,int,int);

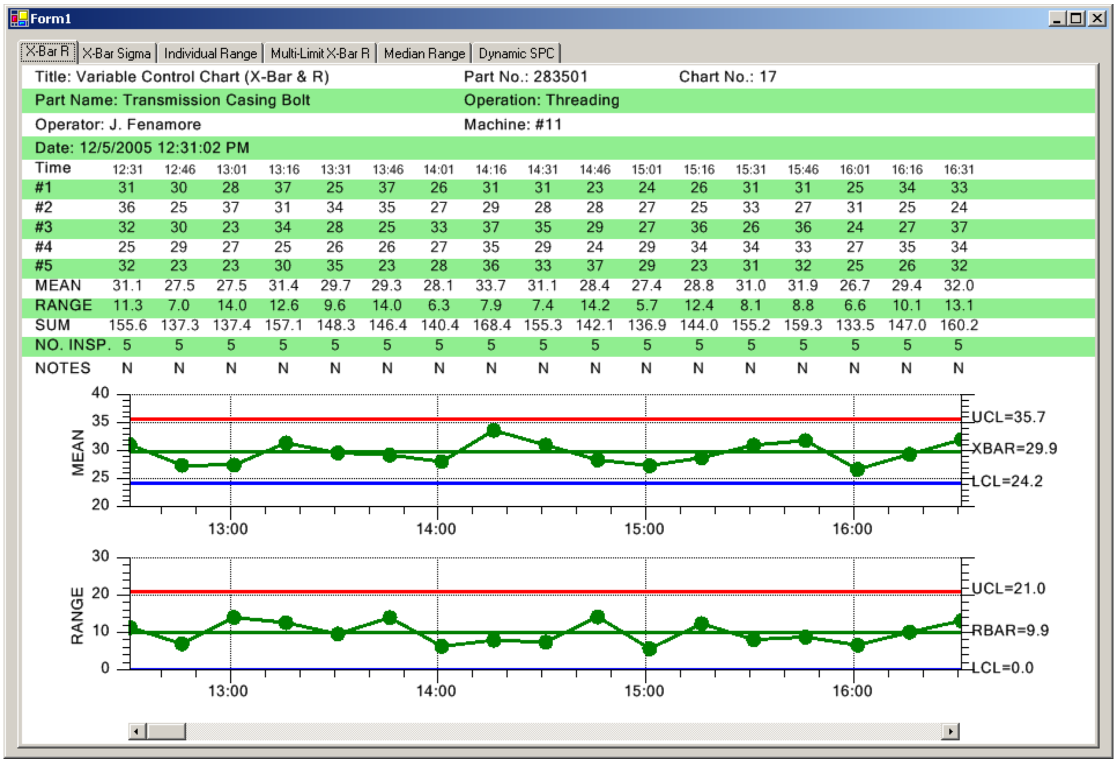
This constructor creates a new **ChartCalendar** object using the designated number of ticks.

public ChartCalendar(long);

This constructor creates a new **ChartCalendar** object using the designated number of milliseconds, or seconds.

public ChartCalendar(long,bool);

The sampled values initialize the chart after the **InitSPC**… call, but before the **RebuildChartUsingCurrentData** call. The example below is from the **TimeVariableControlCharts.XBarRChart** example program. The **AddNewSampleRecord** routine is called in the **SimulateData** method.



Title: Variable Control Chart (X-Bar & R) Part No.: 283501 Chart No.: 17

Part Name: Transmission Casing Bolt Operation: Threading

Operator: J. Fenamore Machine: #11

Date: 12/5/2005 12:31:02 PM

Time	12:31	12:46	13:01	13:16	13:31	13:46	14:01	14:16	14:31	14:46	15:01	15:16	15:31	15:46	16:01	16:16	16:31
#1	31	30	28	37	25	37	26	31	31	23	24	26	31	31	25	34	33
#2	36	25	37	31	34	35	27	29	28	28	27	25	33	27	31	25	24
#3	32	30	23	34	28	25	33	37	35	29	27	36	26	36	24	27	37
#4	25	29	27	25	26	26	27	35	29	24	29	34	34	33	27	35	34
#5	32	23	23	30	35	23	28	36	33	37	29	23	31	32	25	26	32
MEAN	31.1	27.5	27.5	31.4	29.7	29.3	28.1	33.7	31.1	28.4	27.4	28.8	31.0	31.9	26.7	29.4	32.0
RANGE	11.3	7.0	14.0	12.6	9.6	14.0	6.3	7.9	7.4	14.2	5.7	12.4	8.1	8.8	6.6	10.1	13.1
SUM	155.6	137.3	137.4	157.1	148.3	146.4	140.4	168.4	155.3	142.1	136.9	144.0	155.2	159.3	133.5	147.0	160.2
NO. INSP.	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
NOTES	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N

[C#]

```
public void InitializeChart()
{
    this.InitSPCTimeVariableControlChart(charttype, numcategories,
        numsamplespersubgroup, numdatapointsinview, timeincrementminutes);

    .
    .
    .
    SimulateData();

    this.AutoCalculateControlLimits();
    this.AutoScalePrimaryChartYRange();
    this.AutoScaleSecondaryChartYRange();
    this.RebuildChartUsingCurrentData();

}

private void SimulateData()
{   String notesstring = "";
    for (int i=0; i < 200; i++)
    {
        ChartCalendar timestamp = (ChartCalendar) startTime.Clone();
        // Use the ChartData sample simulator to make an array of sample data
        DoubleArray samples = this.ChartData.SimulateMeasurementRecord(30, 10);
        .
        .
        .
        // Add the new sample subgroup to the chart
        this.ChartData.AddNewSampleRecord(timestamp, samples, notesstring);
        // increment simulated time by timeincrementminutes minutes
        startTime.Add(ChartObj.MINUTE, timeincrementminutes);
    }
}
```

[VB]

```
Public Sub InitializeChart()
 ' Initialize the SPCTimeVariableControlChart
     Me.InitSPCTimeVariableControlChart(charttype, numsamplespersubgroup, _
        numdatapointsinview, timeincrementminutes)
    .
    .
    .
    ' Must have data loaded before any of the Auto.. methods are called
    SimulateData()

    ' Calculate the SPC control limits for both graphs of the current SPC chart
    Me.AutoCalculateControlLimits()
    ' Scale the y-axis of the X-Bar chart to display all data and control limits
    Me.AutoScalePrimaryChartYRange()
    ' Scale the y-axis of the Range chart to display all data and control limits
    Me.AutoScaleSecondaryChartYRange()
    ' Rebuild the chart using the current data and settings
    Me.RebuildChartUsingCurrentData()
End Sub 'InitializeChart


Private Sub SimulateData()
    Dim notesstring As [String] = ""
    Dim i As Integer
    For i = 0 To 199
        Dim timestamp As ChartCalendar = CType(startTime.Clone(), ChartCalendar)
        ' Use the ChartData sample simulator to make an array of sample data
        Dim samples As DoubleArray =
           Me.ChartData.SimulateMeasurementRecord(30, 10)
        .
        .
        .
       Me.ChartData.AddNewSampleRecord(timestamp, samples, notesstring)
       ' increment simulated time by timeincrementminutes minutes
       startTime.Add(ChartObj.MINUTE, timeincrementminutes)
    Next i
End Sub 'SimulateData
```

Logging SPC Data to a File

The **SPCControlChartData** method contains routines that log SPC data to a file in a CSV (comma separated value) format. The first row of the file is a prefix of data that defines options and the number of columns associated with sample data, calculated data and control limit data. The second row of data are the column heads for each item in the data log. Starting with the third row, SPC data is output, record by record. If the data logging feature is turned on, every call to the **AddNewSampleRecord** method will result in the output of that record, and calculated values, to the data log.

A typical datalog file (both the width and length of the data file are truncated) appears below.

```
File prefix:   62,5,3,6
Column Heads:  Time Stamp,Sample #0,Sample #1,Sample #2,…
Record #1:     1/24/2006 12:03:40,22.946081345643,30.6379105980219,…
Record #2:     1/24/2006 12:18:40,23.8902424375481,33.7523682840412,…
Record #3:     1/24/2006 12:33:40,33.1602680593078,28.2172109399537,…
```

The values in the file prefix have the following meaning

63 Data log options = SPCControlChartData.DATALOG_FILE_TIME_STAMP |

SPCControlChartData.DATALOG_FILE_SAMPLED_VALUES |

SPCControlChartData.DATALOG_FILE_CALCULATED_VALUES |

SPCControlChartData.DATALOG_FILE_COLUMN_HEADS |

SPCControlChartData.DATALOG_FILE_NOTES

5 There are five sampled values per record
3 There are three calculated values per record (MEAN, RANGE, SUM for example)
6 There are six control limit values per record (`XBAR,LCL,UCL,RBAR,LCL,UCL) for example`

If you want to view a complete datalog file, run the TimeVariableControlCharts example program and after you terminate the program, view the Datalogfile1.text file in the [C#] TimeVariableControlCharts\Bin\Debug folder, or [VB] TimeVariableControlCharts\Bin folder. The following steps, extracted from the TimeVariableControlChart.VariableControlLimitsCharts example program, turn on data logging:

[C#]

```
int datalogflags = SPCControlChartData.DATALOG_FILE_TIME_STAMP |
    SPCControlChartData.DATALOG_FILE_SAMPLED_VALUES |
    SPCControlChartData.DATALOG_FILE_CALCULATED_VALUES |
    SPCControlChartData.DATALOG_FILE_COLUMN_HEADS |
    SPCControlChartData.DATALOG_FILE_NOTES;

    this.ChartData.DataLogFileOpenForWrite("DatalogFile1.txt", datalogflags);
    this.ChartData.DatalLogEnable = true;
.
.
.
    this.ChartData.AddNewSampleRecord(timestamp, samples, notesstring);
```

[VB]

```
Dim datalogflags As Integer = SPCControlChartData.DATALOG_FILE_TIME_STAMP Or _
    SPCControlChartData.DATALOG_FILE_SAMPLED_VALUES Or _
    SPCControlChartData.DATALOG_FILE_CALCULATED_VALUES Or _
    SPCControlChartData.DATALOG_FILE_COLUMN_HEADS Or _
    SPCControlChartData.DATALOG_FILE_NOTES

    Me.ChartData.DataLogFileOpenForWrite("DatalogFile1.txt", datalogflags)
    Me.ChartData.DatalLogEnable = True
.
.
.
```

```
    Me.ChartData.AddNewSampleRecord(timestamp, samples, notesstring)
```

Every call to the **AddNewSampleRecord** method will append a new SPC record to the file specified in the **DataLogFileOpenForWrite** call.

Specify what items are logged to the datalog file using the **DataLogFlag** property. OR the datalog flags constants together to form the final **DataLogFlags** value.

DATALOG_FILE_ALL Datalog flag specifying that all available items should be logged to the file.

DATALOG_FILE_BATCH_NUMBER Datalog flag specifying that the batch number should be logged to the file.

DATALOG_FILE_CALCULATED_VALUES Datalog flag specifying that the calculated values should be logged to the file.

DATALOG_FILE_COLUMN_HEADS Datalog flag specifying that the column heads should be logged to the file.

DATALOG_FILE_CONTROL_LIMIT_VALUES Datalog flag specifying that the control limit values should be logged to the file.

DATALOG_FILE_NOTES Datalog flag specifying that the notes should be logged to the file.

DATALOG_FILE_SAMPLED_VALUES Datalog flag specifying that the sampled values should be logged to the file.

DATALOG_FILE_TIME_STAMP Datalog flag specifying that the time stamp should be logged to the file.

[C#]

```
this.ChartData.DataLogFlags = SPCControlChartData.DATALOG_FILE_TIME_STAMP |
    SPCControlChartData.DATALOG_FILE_SAMPLED_VALUES |
    SPCControlChartData.DATALOG_FILE_CALCULATED_VALUES |
    SPCControlChartData.DATALOG_FILE_COLUMN_HEADS |
    SPCControlChartData.DATALOG_FILE_NOTES;
```

[VB]

```
Me.ChartData.DataLogFlags = SPCControlChartData.DATALOG_FILE_TIME_STAMP Or _
    SPCControlChartData.DATALOG_FILE_SAMPLED_VALUES Or _
     SPCControlChartData.DATALOG_FILE_CALCULATED_VALUES Or _
     SPCControlChartData.DATALOG_FILE_COLUMN_HEADS Or _
    SPCControlChartData.DATALOG_FILE_NOTES
```

It is also possible to read a previously saved datalog file and initialize the **ChartData** object with previously collected data. While the data can be initialized, it is still important that the originating **SPCChartBase** object is initialized properly for the data it is to receive. Use the **ChartData.ReadAllValuesFromFile** method to read previously saved values. The example below is extracted from the TimeVariableControlChart.VariableControlLimitsCharts example program.

[VB]

```
Me.InitSPCTimeVariableControlChart(charttype, numsamplespersubgroup, _
       numdatapointsinview, timeincrementminutes)

.
.
.
If (File.Exists("DatalogFile1.txt")) Then
   Me.ChartData.ReadAllValuesFromFile("DatalogFile1.txt")
```

[C#]

```
this.InitSPCTimeVariableControlChart(charttype,
       numsamplespersubgroup, numdatapointsinview, timeincrementminutes);
.
.
.
if (File.Exists("DatalogFile1.txt"))
{
    this.ChartData.ReadAllValuesFromFile("DatalogFile1.txt");
}
```

It is important that the *charttype* parameter matches the chart type used to save the original data, and that the *numberofsamplepersubgroup* value matches the number of samples in the original data.

Control Limit Alarms

Class SPCControlLimitRecord

ChartObj
        |
        +-- SPCControlLimitRecord

The SPCControlLimitRecord stores control limit alarm information for the SPCControlChartData class. The SPCControlLimitRecord class specifies the type of the alarm, the alarm limit value, alarm text messages and alarm hysteresis value. The SPCControlChartData classes store the SPCControlLimitRecord objects in the SPCControlChartData.ControlLimitValues array list

SPCControlLimitRecord constructors

This constructor creates a new instance of a **SPCControlLimitRecord** object, using the specified spc data object, calculated value object, alarm type, alarm limit value and alarm message.

[VB]

```
Overloads Public Sub New( _
   ByVal processvar As SPCControlChartData, _
   ByVal clr As SPCCalculatedValueRecord, _
   ByVal parametertype As Integer, _
   ByVal alarmlimitvalue As Double, _
   ByVal normalmessage As String, _
   ByVal alarmmessage As String _
)
```

[C#]

```
public SPCControlLimitRecord(
   SPCControlChartData processvar,
   SPCCalculatedValueRecord clr,
   int parametertype,
   double alarmlimitvalue,
   string normalmessage,
   string alarmmessage
);
```

Parameters

processvar
Specifies the process variable that the alarm is attached to.

clr
Specifies the calculated value record the alarm is attached to.

parametertype
Specifies the alarm type: SPC_NOTA_LIMIT, SPC_LOWERTHAN_LIMIT, or SPC_GREATERTHAN_LIMIT.

alarmlimitvalue
Specifies the alarm limit value.

normalmessage
Specifies display message when no alarm present.

alarmmessage
Specifies the alarm message.

The most commonly used **SPCControlLimitRecord** properties are:

Public Static (Shared) Fields

SPC_GREATERTHAN_LIMIT	Specifies the alarm is a greater than alarm.
SPC_LOWERTHAN_LIMIT	Specifies the alarm is a lower than alarm.
SPC_NOTA_LIMIT	Specifies the limit is not an alarm, just a value.

Public Instance Constructors

SPCControlLimitRecord	Overloaded. Initializes a new instance of the **SPCControlLimitRecord** class.

Public Instance Properties

AlarmDisplay	Get/Set the alarm display flag.
AlarmEnable	Get/Set the alarm enable flag.
AlarmMessage	Get/Set the current alarm message.
AlarmState	Get/Set the alarm state, true if the last call to CheckAlarm show that the process variable currently in alarm.
CalculatedValueSrc	Set/Get a reference to the SPCCalculatedValueRecord object associated with the control limit.
ControlLimitText	Get/Set the Normal alarm message;
ControlLimitType	Get/Set the alarm type: SPC_NOTA_LIMIT, SPC_LOWERTHAN_LIMIT, or SPC_GREATERTHAN_LIMIT.
ControlLimitValue	Get/Set the alarm limit value.
ControlLimitValues	Get/Set the controlLimitValues array.
HysteresisValue	Get/Set the alarm hysteresis value.
PrevAlarmState	Get/Set the previous alarm state.
SPCProcessVar	Get/Set the spcDataVar object.
SymbolColor	Get/Set the alarm symbol color.
TextColor	Get/Set the alarm text color.

Public Instance Methods

CheckAlarm	Overloaded. Check the current value against the parameterValue.
Clone	Returns an object that is a Clone of this **SPCControlLimitRecord** object.
Copy	Overloaded. Copies the source **SPCControlLimitRecord** object.
ErrorCheck	Checks the **SPCControlLimitRecord** object for common errors. Current error state. Returns an error code.
GetAlarm	Returns the current alarm state based on the passed in value.
GetControlLimitHistoryValue	Get a values for the controlLimitsValues historical buffer.
SetControlLimitValue	Set current value of the control limit and adds that value to the controlLimitValues historical array.

The SPCControlLimitRecord properties are documented in the QCSPCChartNet6.chm documentation file, located in the \doc subdirectory.

Example of trapping SPCControlLimitRecord alarm using an event delegate

The example below specifies an alarm event delegate for the control limit alarms. The example was extracted from the TimeVariableControlCharts.DynamicXBarRChart example program.

[C#]

```
public void InitializeChart()
{

    // TODO: Add any initialization after the InitForm call

    // Initialize the SPCTimeVariableControlChart
    this.InitSPCTimeVariableControlChart(charttype,
        numsamplespersubgroup, numdatapointsinview, timeincrementminutes);
    // Set the strings used in the header section of the table
    this.ChartData.Title = "Variable Control Chart (X-Bar & R)";
    .
    .
    .

    this.ChartData.AlarmStateEventHandler +=
        new SPCControlLimitAlarmEventDelegate(this.SPCControlLimitAlarm);
    // don't generate alarms in initial data simulation
    this.ChartData.AlarmStateEventEnable = false;

    SimulateData();
    .
    .
    .
    //  generate alarms starting now
    this.ChartData.AlarmStateEventEnable = true;
}

private void SPCControlLimitAlarm(object sender, SPCControlLimitAlarmArgs e)
    {
        SPCControlLimitRecord alarm = e.EventAlarm;
        double alarmlimitvalue = alarm.ControlLimitValue;
        String alarmlimitvaluestring = alarmlimitvalue.ToString();

        SPCControlChartData spcData = alarm.SPCProcessVar;
        SPCCalculatedValueRecord spcSource = e.SPCSource;
        String calculatedvaluestring = spcSource.CalculatedValue.ToString();

        String message = alarm.AlarmMessage;
        ChartCalendar timestamp = spcData.TimeStamp;
        String timestampstring = timestamp.ToString();

        if (alarm.AlarmState)
            Console.Out.WriteLine(timestampstring + " " + message + "=" +
                    alarmlimitvaluestring + " Current Value" + "=" +
                    calculatedvaluestring);
    }
}
```

[VB]

```
Public Sub InitializeChart()
    .
    .
    .
    ' Initialize the SPCTimeVariableControlChart
    Me.InitSPCTimeVariableControlChart(charttype, numsamplespersubgroup, _
       numdatapointsinview, timeincrementminutes)
    .
    .
    .
    AddHandler Me.ChartData.AlarmStateEventHandler, _
           AddressOf Me.SPCControlLimitAlarm
   ' don't generate alarms in initial data simulation
    Me.ChartData.AlarmStateEventEnable = False

    SimulateData()
    .
    .
    .

    Me.RebuildChartUsingCurrentData()

End Sub 'InitializeChart



Private Sub SPCControlLimitAlarm(ByVal sender As Object, _
    ByVal e As SPCControlLimitAlarmArgs)
   Dim alarm As SPCControlLimitRecord = e.EventAlarm
   Dim alarmlimitvalue As Double = alarm.ControlLimitValue
   Dim alarmlimitvaluestring As [String] = alarmlimitvalue.ToString()
   Dim spcData As SPCControlChartData = alarm.SPCProcessVar
   Dim spcSource As SPCCalculatedValueRecord = e.SPCSource
   Dim calculatedvaluestring As [String] = spcSource.CalculatedValue.ToString()
   Dim message As [String] = alarm.AlarmMessage
   Dim timestamp As ChartCalendar = spcData.TimeStamp
   Dim timestampstring As [String] = timestamp.ToString()
   If alarm.AlarmState Then
       Console.Out.WriteLine((timestampstring + " " + message + "=" +
       alarmlimitvaluestring + " Current Value" + "=" +
       calculatedvaluestring))
   End If
End Sub 'SPCControlLimitAlarm
```

Control Limit Alarm Event Handling

Class SPCControlLimitAlarmArgs

ChartObj

|

+-- SPCControlLimitAlarmArgs

The **SPCControlChartData** class can throw an alarm event based on either the current alarm state, or an alarm transition from one alarm state to another. The **SPCControlLimitAlarmArgs** passes alarm data to the event handler. If you want the alarm event triggered only on the initial transition from the no-alarm state to the alarm state, set the **SPCControlChartData.AlarmTransitionEventEnable** to true and the

SPCControlChartData.AlarmStateEventEnable to false**.** In this case, you will get one event when the process variable goes into alarm, and one when it comes out of alarm. If you want a continuous stream of alarm events, as long as the **SPCControlLimitRecord** object is in alarm, set the **SPCControlChartData.AlarmTransitionEventEnable** to false and the **SPCControlChartData.AlarmStateEventEnable** to true. The alarm events will be generated at the same rate as the **SPCControlChartData.AddNewSampleRecord()** method is called.

SPCControlLimitAlarmArgs constructors

You don't really need the constructors since **SPCControlLimitAlarmArgs** objects are created inside the **SPCControlChartData** class when an alarm event needs to be generated.

The most commonly used **SPCControlLimitAlarmArgs** properties are:

Selected Public Instance Properties

Public Instance Properties

AlarmChannel	Get/Set the alarm channel associated with the alarm.
EventAlarm	Get/Set the **SPCControlLimitRecord** object.
SPCSource	Get/Set the SPCCalculatedValueRecord object associated with the alarm.

A complete listing of **SPCControlLimitAlarmArgs** properties are documented in the QCSPCChartNet6.chm documentation file, located in the \doc subdirectory.

Example

Setup and enable an alarm transition event handler in the following manner:
[C#]

```
this.ChartData.AlarmTransitionEventHandler+=
    new SPCControlLimitAlarmEventDelegate(this.SPCControlLimitAlarm);
this.ChartData.AlarmTransitionEventEnable = true;
```

[VB]

```
AddHandler Me.ChartData.AlarmTransitionEventHandler, _
       AddressOf Me.SPCControlLimitAlarm
Me.ChartData.AlarmTransitionEventEnable = true
```

where the handler method is **this.SPCControlLimitAlarm**

[C#]

```
private void SPCControlLimitAlarm(object sender, SPCControlLimitAlarmArgs e)
{
    .
    .
    .
}
```

[VB]

```
Private Sub SPCControlLimitAlarm(ByVal sender As Object, _
    ByVal e As SPCControlLimitAlarmArgs)
.
.
.
.
End Sub 'SPCControlLimitAlarm
```

Setup and enable an alarm state event handler in an identical manner:

[C#]

```
this.ChartData.AlarmStateEventHandler +=
    new SPCControlLimitAlarmEventDelegate(this.SPCControlLimitAlarm);
this.ChartData.AlarmStateEventEnable = true;
```

[VB]

```
 AddHandler Me.ChartData.AlarmStateEventHandler, _
           AddressOf Me.SPCControlLimitAlarm
  Me.ChartData.AlarmStateEventEnable = True
```

where the handler method is this **SPCControlLimitAlarm**.

[C#]

```
private void SPCControlLimitAlarm(object sender, SPCControlLimitAlarmArgs e)
{
    .
    .
    .
}
```

[VB]

```
Private Sub SPCControlLimitAlarm(ByVal sender As Object, _
    ByVal e As SPCControlLimitAlarmArgs)
.
.
.
.
End Sub 'SPCControlLimitAlarm
```

SPCSampledValueRecord

This class encapsulates a sample data value. It includes a description for the item, the current value of the sampled value, and a history of previous values.

An array list of **SPCSampledValueRecord** objects, one for each sample category, is automatically created when the parent **SPCChartBase** object is created. The programmer does not need to instantiate it.

Public Instance Constructors

SPCSampledValueRecord	Overloaded. Initializes a new instance of the SPCSampledValueRecord class.

Public Instance Properties

SampledValue	Get/Set the current value for this record.
SampledValues	Get/Set the historical array of the sampled value record.
ValueDescription	Get/Set the description of sampled value record.

Public Instance Methods

Copy	Copies the source object.
GetCalculatedValueStatistic	Calculate a statistic for the historical data associated with the sample item.
SetSampledValue	Set the current value of the record, and adds the value to the historical array of the sampled value record.

SPCControlLimitRecord

This class holds information specific to a SPC control limit: including the current value of the control limit, a history of control limit values, description, and the hysteresis value for alarm checking.

Public Static (Shared) Fields

SPC_GREATERTHAN_LIMIT	Specifies the alarm is a greater than alarm.
SPC_LOWERTHAN_LIMIT	Specifies the alarm is a lower than alarm.
SPC_NOTA_LIMIT	Specifies the limit is not an alarm, just a value.

Public Instance Constructors

SPCControlLimitRecord	Overloaded. Initializes a new instance of the **SPCControlLimitRecord** class.

Public Instance Fields

controlLimitValues	A historical record of the control limit values.

Public Instance Properties

AlarmDisplay	Get/Set the alarm display flag.
AlarmEnable	Get/Set the alarm enable flag.
AlarmMessage	Get/Set the current alarm message.
AlarmState	Get/Set the alarm state, true if the last call to CheckAlarm show that the process variable currently in alarm.
ControlLimitText	Get/Set the Normal alarm message;
ControlLimitType	Get/Set the alarm type: SPC_NOTA_LIMIT, SPC_LOWERTHAN_LIMIT, or SPC_GREATERTHAN_LIMIT.
ControlLimitValue	Get/Set the alarm limit value.
ControlLimitValues	Get/Set the controlLimitValues array.
HysteresisValue	Get/Set the alarm hysteresis value.
PrevAlarmState	Get/Set the previous alarm state.
SPCProcessVar	Get/Set the spcDataVar object.
SymbolColor	Get/Set the alarm symbol color.
TextColor	Get/Set the alarm text color.

Public Instance Methods

CheckAlarm	Check the current value against the parameterValue.
Clone	Returns an object that is a Clone of this **SPCControlLimitRecord** object.
Copy	Overloaded. Copies the source **SPCControlLimitRecord** object.
Copy (inherited from **ChartObj**)	Overloaded. Copies the source object.
ErrorCheck	Checks the **SPCControlLimitRecord** object for common errors. Current error state. Returns an error code.
GetAlarm	Returns the current alarm state based on the passed in value.
GetControlLimitHistoryValue	Get a values for the controlLimitsValues historical buffer.
SetControlLimitValue	Set current value of the control limit and adds that value to the controlLimitValues historical array.

SPCCalculatedValueRecord

This is the record class for a calculated SPC statistic. It holds the calculated value type (mean, median, sum, variance, standard deviation, etc.), value, description and historical data.

Public Static (Shared) Fields

SPC_CUSTOM_CALC	Constant value for a custom SPC calculation (unused).
SPC_FRACTION_DEFECTIVE_PARTS_CALC	Constant value for a percent defective parts SPC calculation.
SPC_FRACTION_DEFECTS_CALC	Constant value for a fraction defects SPC calculation.
SPC_INDIVIDUAL_ABS_RANGE_CALC	Constant value for a ABS individual range SPC calculation.
SPC_INDIVIDUAL_COPY_VALUE	Constant value for INDIVIDUAL RANGE .
SPC_INDIVIDUAL_RANGE_CALC	Constant value for a individual range SPC calculation.
SPC_MAX_CALC	Constant value for a maximum SPC calculation.
SPC_MEAN_CALC	Constant value for a mean SPC calculation.
SPC_MEAN_N_MINUS_1_CALC	Constant value for a mean SPC calculation using N-1, rather than N.
SPC_MEDIAN_CALC	Constant value for a median SPC calculation.
SPC_MIN_CALC	Constant value for a minimum SPC calculation.
SPC_PERCENT_DEFECTIVE_PARTS_CALC	Constant value for a percent defective parts calculation.
SPC_PERCENT_DEFECTS_CALC	Constant value for a percent defects SPC calculation.
SPC_RANGE_CALC	Constant value for a range SPC calculation.
SPC_STD_DEVIATION_CALC	Constant value for a standar deviation SPC calculation.
SPC_SUM_CALC	Constant value for a sum SPC calculation.
SPC_TOTAL_DEFECTIVE_PARTS_CALC	Constant value for a total defective parts SPC calculation.
SPC_TOTAL_DEFECTS_CALC	Constant value for a total defects SPC calculation.
SPC_VARIANCE_CALC	Constant value for a variance SPC

	calculation.

Public Static (Shared) Methods

CalculateHistoryStatistic	Calculate the calculated value value based on the data in the source array and the specified calculation type.

Public Instance Constructors

SPCCalculatedValueRecord	Overloaded. Initializes a new instance of the SPCCalculatedValueRecord class.

Public Instance Properties

CalculatedValue	Get the current calculation value for this record.
CalculatedValues	Get the reference to the calculatedValue array.
CalculationType	Set/Get the calculation type for this calculation value record. Use one of the SPCCalculatedValueRecord calculation type constants.
MostRecentSampledValues	Get/Set an array holding the values of the most recent sampled, or measured values used in calculating the records calculateValue value.
ValidValueFlags	Get the reference to the validValueFlags array.
ValueDescription	Get/Set the description of calculation value record.

Public Instance Methods

Copy	Copies the source object.
GetCalculatedValueStatistic	Returns the calculated value value based on the data in the calculated historical data array, calculatedValues. Excludes values that are marked invalid in the validValueFlags array.
IsValueValid	Checks to the validValueFlags to see if a value in the calculated historical data array, calculatedValues, is valid.
SetCalculatedValue	Overloaded. Calculate the calculated value value based on the data in the source array. Sets the calculatedValue property of the class to the result.

SPCProcessCapabilityRecord

This is the record class for calculating and storing SPC process capability statistics. It supports calculating the Cp, Cpl, Cpu, Cpk, Cpm, Pp, Ppl, Ppu, and Ppk statistics.

Public Static (Shared) Fields

SPC_CP_CALC	Constant value CP calculation.
SPC_CPK_CALC	Constant value CPK calculation.
SPC_CPL_CALC	Constant value CPL calculation.
SPC_CPM_CALC	Constant value CPM calculation.
SPC_CPU_CALC	Constant value CPU calculation.
SPC_CUSTOM_PC_CALC	Constant value for a custom SPC calculation (unused).
SPC_PP_CALC	Constant value for a sum SPC calculation.
SPC_PPK_CALC	Constant value PPK calculation.
SPC_PPL_CALC	Constant value PPL calculation.
SPC_PPU_CALC	Constant value PPU calculation.

Public Static (Shared) Properties

DefaultProcessCapabilityStrings	Default descriptors for the process capability strings: "", "Cp", "Cpl", "Cpu","Cpk","Pp", "Pl","Pu","Ppk".

Public Instance Constructors

SPCProcessCapabilityRecord	Overloaded. Initializes a new instance of the SPCProcessCapabilityRecord class.

Public Instance Properties

CalculationType	Set/Get the calculation type for this calculation value record. Use one of the SPCProcessCapabilityRecord calculation type constants.
CurrentValue	Get the current calculation value for this record.
CurrentValues	Get the reference to the currentValue array.
LSLValue	Get the LSL value for this record.
USLValue	Get the USL value for this record.
ValidValueFlags	Get the reference to the validValueFlags array.
ValueDescription	Get/Set the description of calculation value record.

Public Instance Methods

CalculateProcessCapabilityValue	Calculate the process capability value..
CalculateProcessCapabilityValues	Calculate the process capability value..
Clone	Returns an object that is a clone of this object.
Copy	Overloaded. Copies the source object.
IsValueValid	Checks to the validValueFlags to see if a value in the calculated historical data array, currentValues, is valid.
Reset	Reset the history buffer of the SPCProcessCapabilityRecord class.
SetProcessCapabilityValue	Calculate the process capability value. Sets the currentValue property of the class to the result.

SPCGeneralizedTableDisplay

This class manages a list of **ChartText** objects (**NumericLabel**, **StringLabel** and **TimeLabel** objects), that encapsulate each unique table entry in the SPC chart table. This class also manages the spacing between the rows and columns of the table, and the alternating stripe used as a background for the table.

Public Static (Shared) Fields

NUMERIC_ROW_SPACING	Constant specifies that the next row to the table should user numeric row spacing.
TABLE_NO_COLOR_BACKGROUND	Constant specifies that the table does not use a background color.
TABLE_SINGLE_COLOR_BACKGROUND	Constant specifies that the table uses a single color for the background (backgroundColor1).
TABLE_SINGLE_COLOR_BACKGROUND_GRID	Constant specifies that the table uses horizontal stripes of color for the background (backgroundColor1 and backgroundColor2).
TABLE_STRIPED_COLOR_BACKGROUND	Constant specifies that the table uses horizontal stripes of color for the background (backgroundColor1 and backgroundColor2).
TEXT_ROW_SPACING	Constant specifies that the next row to the table should user text row spacing.

Public Static (Shared) Properties

DefaultTableFont	Set/Get the default font used in the table display.

Public Instance Constructors

SPCGeneralizedTableDisplay	Overloaded. Initializes a new instance of the SPCGeneralizedTableDisplay class.

Public Instance Properties

BackgroundBarXOffset	Set/Get the background bar left offset, in normalized coordinates.
BackgroundColor1	Set/Get the first of two colors used in the alternating background colors used to delineate the table rows.
BackgroundColor2	Set/Get the second of two colors used in the alternating background colors used to delineate the table rows.
CalculatedItemTemplate	Get/Set the CalculatedItemTemplate object used as a template for displaying calculated numeric values in the table.
CalculatedLabelFont	Get/Set the font used in the display of calculated numeric values in the table.
CurrentColumnPosition	Get/Set the current column position.
CurrentRowPosition	Get/Set the current column position.
NotesItemTemplate	Get/Set the StringItemTemplate object used as a template for displaying string values in the table.
NotesLabelFont	Get/Set the font used in the display of string values in the table.
NumericColumnSpacing	Get/Set the numeric column spacing.
NumericRowSpacing	Get/Set the numeric row spacing.
SampleItemTemplate	Get/Set the SampleItemTemplate object used as a template for displaying numeric values in the table.
SampleLabelFont	Get/Set the font used in the display of sample numeric values in the table.
StartColumnPosition	Get/Set the starting x-position, in normalized coordinates, of the left-most column of the table.
StartRowPosition	Get/Set the starting y-position, in normalized coordinates, of the first row of the table.
StringItemTemplate	Get/Set the StringItemTemplate object used as a template for displaying string values in

	the table.
StringLabelFont	Get/Set the font used in the display of string values in the table.
TableBackgroundMode	Set/Get the first of two colors used in the alternating background colors used to delineate the table rows.
TextColumnSpacing	Get/Set the text column spacing.
TextRowOffset	Set/Get the offset between the start of the row and the top of the text, in normalized coordinates.
TextRowSpacing	Get/Set the text row spacing.
TimeColumnSpacing	Get/Set the time column spacing.
TimeItemTemplate	Get/Set the TimeLabel object used as a template for displaying time values in the table.
TimeLabelFont	Get/Set the font used in the display of time values in the table.
TimeRowSpacing	Get/Set the time row spacing.

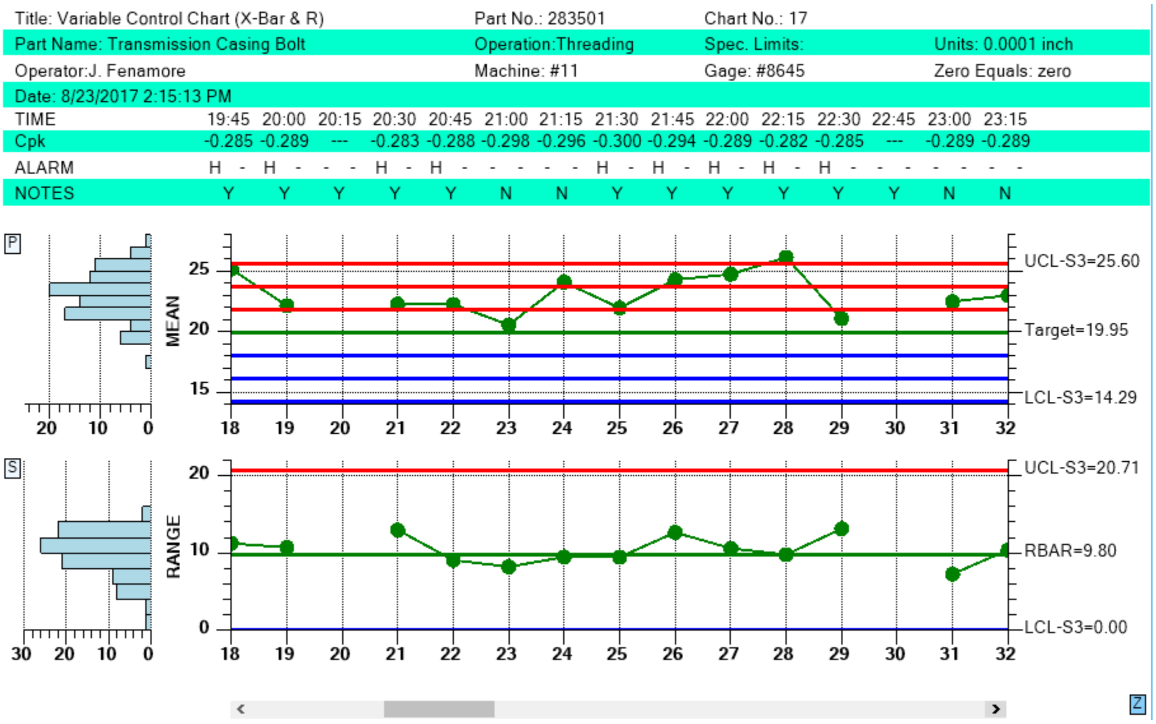
Public Instance Methods

AddCalculatedItem	Overloaded. Add a calculated numeric item to the table, using the specified column spacing increment.
AddHorizontalBar	Add a horizontal bar as a row background for the table.
AddNotesItem	Overloaded. Add a string item to the table, using the specified column spacing increment.
AddNumericItem	Overloaded. Add a numeric item to the table, using the specified column spacing increment.
AddStringItem	Overloaded. Add a string item to the table, using the specified column spacing increment.
AddTimeItem	Overloaded. Add a time item to the table, using the specified column spacing increment.
Clone	Returns an object that is a clone of this object.
Copy	Overloaded. Copies the source object.
GetChartLabel	Get a specific ChartLabel object in the chartLabelArray array list.
IncrementRow	Overloaded. Add another row to the table, using the specified row spacing increment.

InitDefaults	Initialize default values for the class.

Marking a sample internal as bad

A sample interval can be marked as bad. This will disable plotting of the sample interval in the Primary and Secondary charts. It will also prevent the sample interval values from being evaluated in control limit testing, and in auto- calculations for control limits and y-axis range.



Title: Variable Control Chart (X-Bar & R)   Part No.: 283501   Chart No.: 17
Part Name: Transmission Casing Bolt   Operation:Threading   Spec. Limits: Units: 0.0001 inch
Operator:J. Fenamore   Machine: #11   Gage: #8645   Zero Equals: zero
Date: 8/23/2017 2:15:13 PM

TIME	19:45	20:00	20:15	20:30	20:45	21:00	21:15	21:30	21:45	22:00	22:15	22:30	22:45	23:00	23:15
Cpk	-0.285	-0.289	---	-0.283	-0.288	-0.298	-0.296	-0.300	-0.294	-0.289	-0.282	-0.285	---	-0.289	-0.289
ALARM	H -	H -	- -	H -	H -	- -	- -	H -	H -	H -	H -	H -	- -	- -	- -
NOTES	Y	Y	Y	Y	Y	N	N	Y	Y	Y	Y	Y	Y	N	N

A hole (sample intervals 20 and 30 in the picture above) will appear in the main plot of the Primary and Secondary charts when a sample interval is marked bad.

ExcludeRecordFromControlLimitCalculations

Exclude the specified record from the SOC control limit calculations.

```
public void ExcludeRecordFromControlLimitCalculations(int item, bool exclude)
```

item	The index of the item to exclude.
exclude	Set true and the item is excluded. Set false and it is included.

For example:

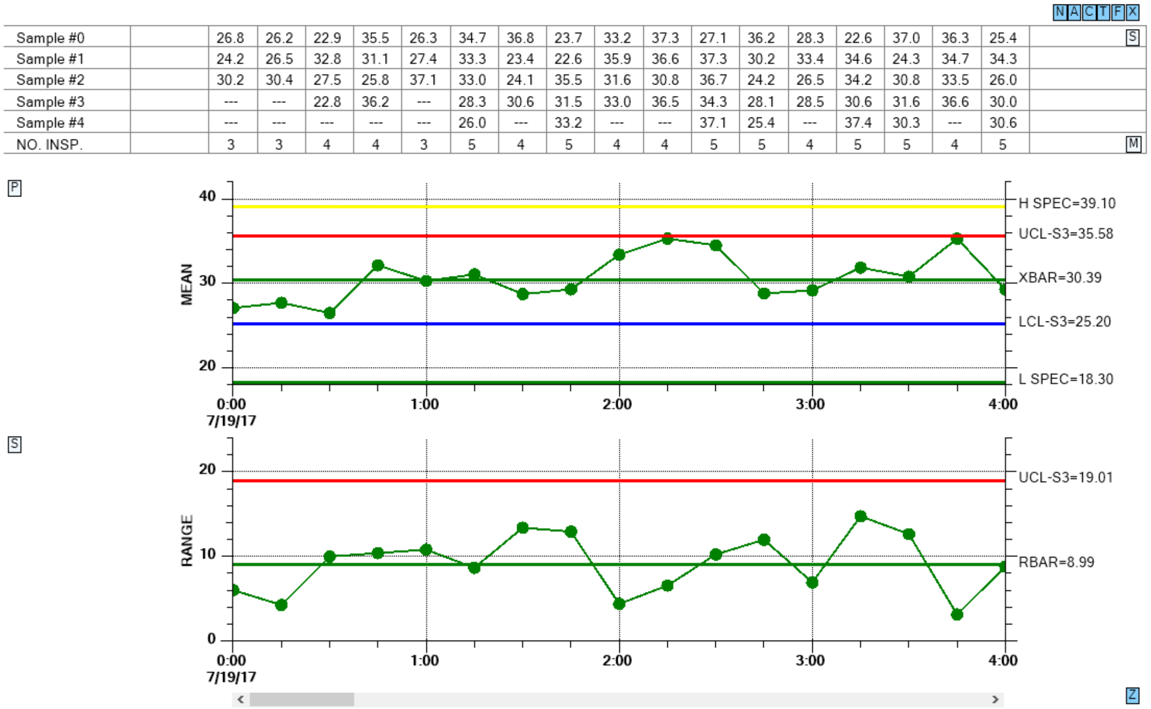
```
int currentRecordNum = this.ChartData.CurrentNumberRecords - 1;

this.ChartData.ExcludeRecordFromControlLimitCalculations(currentRecordNum, true);
```

For a complete example, see the module NewRev3Features.IgnoreSampleInterval.

Sample Interval Update with an invalid number of samples

When you have specified one of the fixed sample size charts, a more lenient evaluation method for sample interval updates has been implemented. You can now enter less than or greater than the specified number of samples for an interval update without the software complaining. It will process the samples as if the proper number of values was entered, without introducing zero values into the sample interval. So if the specified number of samples per sample interval is 5, and 3 are entered, it will still work. It will calculate the mean and range of the sample interval using only 3 values. However, it will not adjust the control limits to take into account the incorrect number of samples, unlike what is done in the dedicated variable sample size (_VSS) charts. Also, you cannot enter in a single value for any of the control charts which require more than one value. Because then the software cannot calculate the range or sigma value, a required calculation. The programmer and user are cautioned that any alarms (or lack thereof) triggered when using this option may not be valid.

In the example below, the number of samples per sample interval varies from three to five samples. Note that the table displays as many rows of sample data as the fixed number of samples per sample interval in the chart definition, five in this case. Missing data values are designated using "...".



Sample #0		26.8	26.2	22.9	35.5	26.3	34.7	36.8	23.7	33.2	37.3	27.1	36.2	28.3	22.6	37.0	36.3	25.4
Sample #1		24.2	26.5	32.8	31.1	27.4	33.3	23.4	22.6	35.9	36.6	37.3	30.2	33.4	34.6	24.3	34.7	34.3
Sample #2		30.2	30.4	27.5	25.8	37.1	33.0	24.1	35.5	31.6	30.8	36.7	24.2	26.5	34.2	30.8	33.5	26.0
Sample #3		---	---	22.8	36.2	---	28.3	30.6	31.5	33.0	36.5	34.3	28.1	28.5	30.6	31.6	36.6	30.0
Sample #4		---	---	---	---	---	26.0	---	33.2	---	---	37.1	25.4	---	37.4	30.3	---	30.6
NO. INSP.		3	3	4	4	3	5	4	5	4	4	5	5	4	5	5	4	5

Note that the sample data in the table has many columns where the number of samples is less than the specified 5 samples per sample interval.

If you update the sample interval with more than the specified number of samples, the software will calculate the mean, sigma, and range of the sample interval (and any other calculations in needs to) using all of the values values. It will NOT store the additional values though. The table will not show these additional values. The sample interval data is only displayed up to the original, fixed, number of samples per sample interval.

If you acquire your sample data, and find that you do not have valid, or sufficient samples for the current chart (> 1 for all but the Individual Range charts), you should skip the sample interval update for that sample interval.

See the NewRev3Features.EnhanceLimitSymbols demo for an example.

Alarm Forcing

High and low control lines can be set to explicit values, or you can have the software auto-calculate the values based on historical data. As new sample intervals are added to the chart, the new values are compared to existing control limits and the alarm conditions noted. But, you can override the alarm limit display, so that a normal condition shows as an alarm, or an alarm condition shows as normal. Obviously you can distort the meaning of the chart if you hide alarms or show alarms that the sample data does not support. But that issue is left to the programmer and end user.

SetForcedControlLimitValues

Force the current sample interval of the chart to the specified forcing value.

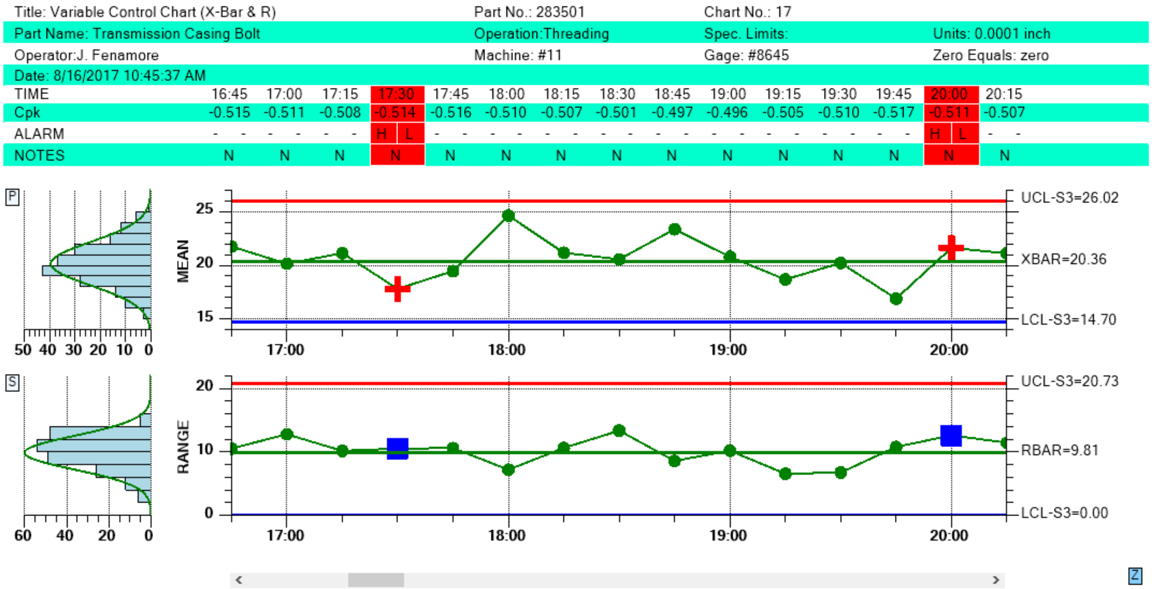
```
public void SetForcedControlLimitValues(int chartnum, int forcingvalue)
```

chartnum chart number. Use SPCChartObjects.PRIMARY_CHART or SPCChartObjects.SECONDARY_CHART

forcingvalue forcing value. Use one of forcing value constants: SPCChartForceAlarm.FORCE_LOW, SPCChartForceAlarm.FORCE_HIGH, SPCChartForceAlarm.FORCE_NORMAL

For example:

```
this.ChartData.SetForcedControlLimitValues(SPCChartObjects.PRIMARY_CHART,
SPCChartForceAlarm.FORCE_HIGH);

this.ChartData.SetForcedControlLimitValues(SPCChartObjects.SECONDARY_CHART,
SPCChartForceAlarm.FORCE_LOW);
```



Title: Variable Control Chart (X-Bar & R) Part No.: 283501 Chart No.: 17

Part Name: Transmission Casing Bolt Operation:Threading Spec. Limits: Units: 0.0001 inch

Operator:J. Fenamore Machine: #11 Gage: #8645 Zero Equals: zero

Date: 8/16/2017 10:45:37 AM

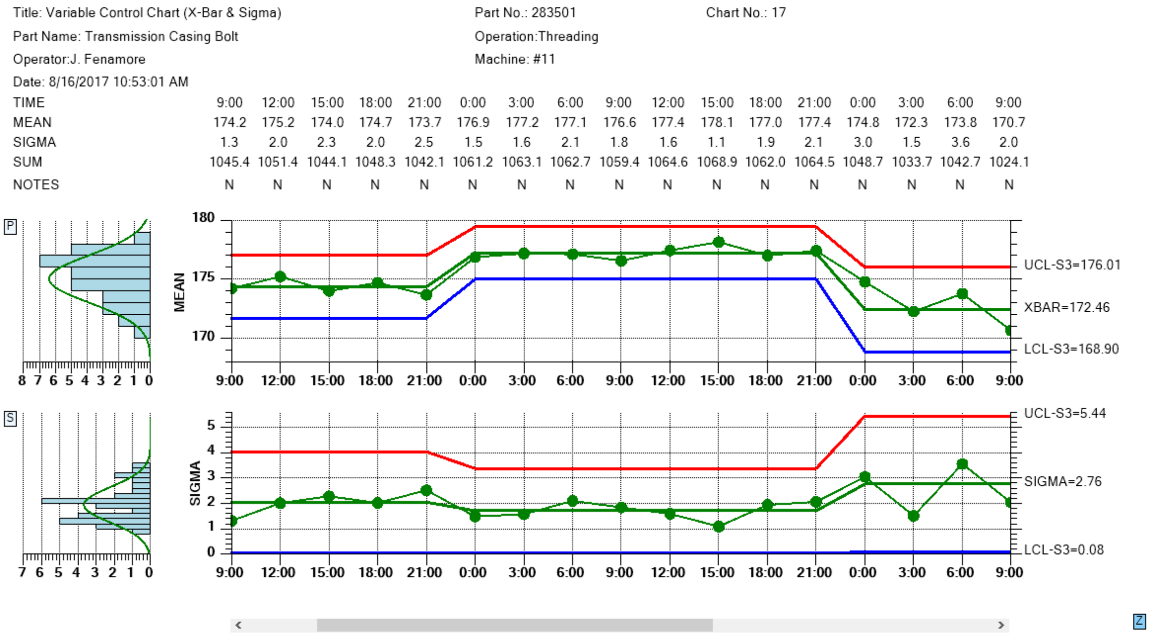
TIME	16:45	17:00	17:15	17:30	17:45	18:00	18:15	18:30	18:45	19:00	19:15	19:30	19:45	20:00	20:15
Cpk	-0.515	-0.511	-0.508	-0.514	-0.516	-0.510	-0.507	-0.501	-0.497	-0.496	-0.505	-0.510	-0.517	-0.511	-0.507
ALARM	- -	- -	- -	H L	- -	- -	- -	- -	- -	- -	- -	- -	- -	H L	- -
NOTES	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N

Sample intervals can be forced to into an alarm state, regardless of the calculated value. All of the alarms in the example above are forced and not representative of the sample data.

See the NewRev3Features.ForceAlarms demo for an example.

Batch update with auto-calculated control limits

We added a routine (CalculateControlLimitsFromBatch) in the main SPCChartBase class, which will calculate a set of control limits based on a batch of N sample intervals. So if your first batch has 20 sample intervals, you would call that routine and you would get back the control limits specific to that set of data. Then you would set the control limits to those values, then update the chart with your sample interval data. The data would be evaluated against the control limits you just set. Then, you can take another batch of N sample intervals, calculate the limits, set the limits, and update the graph again. The new process data, and control limits will be appended to the line plots of the existing chart. The new limits will only apply to the new data values. Repeat ad infinitum.



Title: Variable Control Chart (X-Bar & Sigma)
Part No.: 283501
Chart No.: 17
Part Name: Transmission Casing Bolt
Operation:Threading
Operator:J. Fenamore
Machine: #11
Date: 8/16/2017 10:53:01 AM

TIME	9:00	12:00	15:00	18:00	21:00	0:00	3:00	6:00	9:00	12:00	15:00	18:00	21:00	0:00	3:00	6:00	9:00
MEAN	174.2	175.2	174.0	174.7	173.7	176.9	177.2	177.1	176.6	177.4	178.1	177.0	177.4	174.8	172.3	173.8	170.7
SIGMA	1.3	2.0	2.3	2.0	2.5	1.5	1.6	2.1	1.8	1.6	1.1	1.9	2.1	3.0	1.5	3.6	2.0
SUM	1045.4	1051.4	1044.1	1048.3	1042.1	1061.2	1063.1	1062.7	1059.4	1064.6	1068.9	1062.0	1064.5	1048.7	1033.7	1042.7	1024.1
NOTES	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N

You can auto-calculate control limits for batches of process values, and concatenate the resulting graphs together.

To make it even easier to use, we added flags to the CalculateControlLimitsFromBatch routine which can optionally set the calculated control limits, and update the chart with the supplied data, eliminating the need for the programmer to that externally. So the way this works is that each batch will display all at once, appended to the existing data. There is no incremental update of one sample interval at a time.

CalculateControlLimitsFromBatch

Force the current sample interval of the chart to the specified forcing value.

```
public void CalculateControlLimitsFromBatch(TimeArray timestamps,
List<DoubleArray> data, DoubleArray limits, bool setlimits, bool addnewsamples)
```

timestamps	The timestamps.
data	The data.
limits	The limits.
setlimits	if set to true set the control limits of the chart to the calculated limits.
addnewsamples	if set to true update the chart with the sample data. This also requires that the *setlimits* parameter is set true.

In the example code below, we use a data simulator to create the two required arrays: sampleIntervalData and timeStampData. Note that the timeStampData array is one of our TimeArray classes, while the sampleIntervalData class is a .Net generic List class of our DoubleArray class (ie. An array of DoubleArray)

```
List<DoubleArray> sampleIntervalData = new List<DoubleArray>();
TimeArray  timeStampData = new TimeArray();
DoubleArray controlLimits = new DoubleArray();
```

We then call a simple data simulator which fills out the two arrays: with values based on the calling parameters.

```
SimulateData(new ChartCalendar(2017, 7, 22), 8, 175, 5, timeStampData,
sampleIntervalData);
```

So each sample interval of the soon to be graph is represented by a time stamp value (a ChartCalendar value, which is easily derived from the .Net TimeDate class if that is what you use) in the timeStampData array, and an element in the sampleIntervalData array, where each element represents a DoubleArray of sample interval values.

```
this.CalculateControlLimitsFromBatch (timeStampData, sampleIntervalData,
controlLimits, true, true);
```

If you set the *setlimits* parameter to true, the routine automatically sets the charts control limits to the ones calculated for this batch of data. If you set the *addnewsamplesparameter* true, it automatically updates the chart with the sample interval data, using the values of *timestamps*, and *data* array parameters. It calls the ChartData.AddNewSampleRecord for each element of the *timestamps* and *data* array. So setting the last two parameters true replaces the following code in your own program:

```
if (setlimits)
{
    ChartData.SetControlLimitValues(limits.GetElements());
    if (addnewsamples)
    {
       for (int i = 0; i < timestamps.Length; i++)
       {
           DoubleArray sampleinterval = data[i];
           ChartData.AddNewSampleRecord(timestamps[i], sampleinterval);
       }
    }
}
```

6. SPC Variable Control Charts

SPCEventVariableControlChart
SPCTimeVariableControlChart
SPCBatchVariableControlChart

Variable Control Charts are used with sampled quality data that can be assigned a specific numeric value, other than just 0 or 1. This includes, but is not limited to, the measurement of a critical dimension (height, length, width, radius, etc.), the weight a specific component, or the measurement of an important voltage. The variable control charts supported by this software include X-Bar R (Mean and Range), X-Bar Sigma, Median and Range, X-R (Individual Range), MA (Move Average), MAMR (Moving Average / Moving Range), MAMS (Moving Average / Moving Sigma), EWMA (Exponentially Weighted Moving Average) and CUSum charts.

X-Bar R Chart – Also known as the Mean (or Average) and Range Chart

The X-Bar R chart monitors the trend of a critical process variable over time using a statistical sampling method that results in a subgroup of values at each sample interval. The X-Bar part of the chart plots the mean of each sample subgroup and the Range part of the chart monitors the difference between the minimum and maximum value in the subgroup. X-Bar R charts are created using the **SPCEventVariableControlChart**, **SPCTimeVariableControlChart** and **SPCBatchVariableControlChart** classes.

X-Bar Sigma – Also known as the X-Bar S Chart

Very similar to the X-Bar R chart, the X-Bar Sigma chart replaces the Range plot with a Sigma plot based on the standard deviation of the measured values within each subgroup. This is a more accurate way of establishing control limits if the sample size of the subgroup is moderately large (> 10). Though computationally more complicated, the use of a computer makes this a non-issue. The X-Bar Sigma chart comes in fixed sample subgroup size, and variable sample subgroup size, versions. X-Bar Sigma charts are created using the **SPCEventVariableControlChart, SPCTimeVariableControlChart** and **SPCBatchVariableControlChart** classes.

Median Range – Also known as the Median and Range Chart

Very similar to the X-Bar R Chart, Median Range chart replaces the Mean plot with a Median plot representing the median of the measured values within each subgroup. In order to use a Median Range chart the process needs to be well behaved, where the variation in measured variables are (1) known to be distributed normally, (2) are not very often disturbed by assignable causes, and (3) are easily adjusted. . Median Range charts are created using the

SPCEventVariableControlChart, SPCTimeVariableControlChart and **SPCBatchVariableControlChart** classes.

Individual Range Chart – Also known as the X-R Chart

The Individual Range Chart is used when the sample size for a subgroup is 1. This happens frequently when the inspection and collection of data for quality control purposes is automated and 100% of the units manufactured are analyzed. It also happens when the production rate is low and it is inconvenient to have sample sizes other than 1. The X part of the control chart plots the actual sampled value (not a mean or median) for each unit and the R part of the control chart plots a moving range that is calculated using the current value of sampled value minus the previous value. . Individual Range charts are created using the **SPCEventVariableControlChart, SPCTimeVariableControlChart** and **SPCBatchVariableControlChart** classes.

Levey-Jennings Chart

The Levey-Jennings chart is used almost exclusively in laboratory settings. It uses a chart very similar to the Individual Range chart above, the major difference being that it only uses the Primary individual data point graph of the chart and does not include the Secondary range graph. Also, the Levey-Jennings chart uses the Westgard rules which utilizes tests involving 1-, 2- and 3- sigma control limits. The control limit calculations depart from all of the other SPC Chart types in that the target value (mean) and control limit (sigma) calculations use the overall mean and standard deviation values from the entire, charted, sample population. See the links https://en.wikipedia.org/wiki/Laboratory_quality_control and https://www.westgard.com/lesson12.htm for more information about the underlying principles of the Levey-Jennings chart.

EWMA Chart – Exponentially Weighted Moving Average

The EWMA chart is an alternative to the preceding Shewhart type control charts (X-Bar R and I-R charts in particular) and is most useful for detecting small shifts in the process mean. It uses a weighted moving average of previous values to “smooth” the incoming data, minimizing the affect of random noise on the process. It weights the current and most recent values more heavily than older values, allowing the control line to react faster than a simple MA (Moving Average) plot to changes in the process. Like the Shewhart charts, if the EWMA value exceeds the calculated control limits, the process is considered out of control. While it is usually used where the process uses 100% inspection and the sample subgroup size is 1 (same is the X-R chart), it can also be used when sample subgroup sizes are greater than one. EWMA charts are created using the **SPCEventVariableControlChart, SPCTimeVariableControlChart** and **SPCBatchVariableControlChart** classes.

MA Chart – Moving Average

The MA chartis another alternative to the preceding Shewhart type control charts (X-Bar Rand I-R charts in particular) and is most useful for detecting small shifts in the process mean. It uses a moving average, where the previous (N-1) sample values of the process variable are averaged together along with the current value to produce the current chart value. This helps to "smooth" the incoming data, minimizing the affect of random noise on the process. Unlike the EWMA chart the MA chart weights the current and previous (N-1) values equally in the average. While the MA chart can often detect small changes in the process mean faster than the Shewhart chart types, it is generally less effective than either the EWMA chart, or the CuSum chart MA charts are created using the **SPCEventVariableControlChart, SPCTimeVariableControlChart**and **SPCBatchVariableControlChart**classes.

MAMR Chart – Moving Average/Moving Range

The MAMR chart combines our Moving Average chart with a Moving Range chart. The Moving Average chart is primary (topmost) chart, and the Moving Range chart is the secondary (bottom) chart. It uses a single sample/subgroup, same as our standard [Individual-Range], [Moving Average], [EWMA], and [Moving Average] charts. When calculating the Moving Range, it windows the same data values used in the Moving Average calculation. Note that the limits are variable (wider) at the beginning, taking into account the fewer samples in the start up of any type of SPC chart which uses a sliding window in the calculation of moving average and moving range statistics.

MAMS Chart – Moving Average / Moving Sigma

The MAMS chart combines our Moving Average chart with a Moving Sigma chart. The Moving Average chart is primary (topmost) chart, and the Moving Sigma chart is the secondary (bottom) chart. It uses a single sample/subgroup, same as our standard [Individual-Range], [Moving Average], [EWMA], and [Moving Average] charts. When calculating the Moving Sigma, it windows the same data values used in the Moving Average calculation. Note that the limits are variable (wider) at the beginning, taking into account the fewer samples in the start up of any type of SPC chart which uses a sliding window in the calculation of moving average and moving sigma statistics.

CuSum Chart – Tabular, one-sided, upper and lower cumulative sum

The CuSum chart is a specialized control chart, which like the EWMA and MA charts, is considered to be more efficient that the Shewhart charts at detecting small shifts in the process mean, particularly if the mean shift is less than 2 sigma. There are several types of CuSum charts, but the easiest to use and the most accurate is considered the tabular CuSum chart and that is the one implemented in this software. The tabular cusum works by accumulating deviations that are above the process mean in one statistic (C+) and accumulating deviations below the process mean in a second statistic (C-). If either statistic (C+ or C-) falls outside of the calculated limits, the process is considered out of control.

Time-Based and Batch-Based SPC Charts

The **QCSPCChart** software further categorizes *Variable Control* as either time- or batch- based. Time-based SPC charts are used when data is collected using a subgroup interval corresponding to a specific time interval. Batch-based SPC charts are used when the data subgroup interval is a sequential batch number that does not correspond to a uniform time interval. The major difference in these two types of SPC charts is the display of the x-axis. Variable control charts that sample using a uniform time interval will generally use a time-based x-axis, with time/date axis labels. Variable control charts that sample based on batches will generally use a numeric-based x-axis, with numeric axis labels.

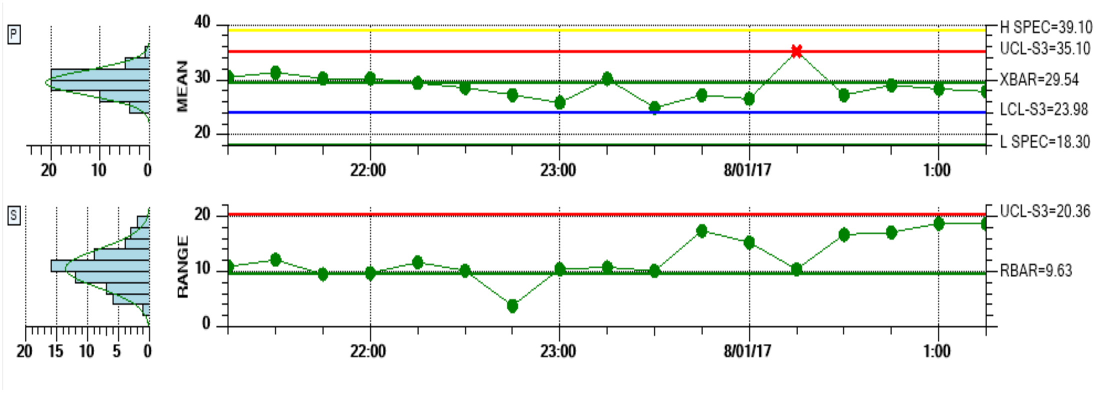
SPECIAL NOTE: The time-based and batch-based SPC charts have been deprecated and replaced by the new event-based charts. In order to maintain backward compatibility, we also keep the old SPCTime... and SPCBatch... control chart classes, but derive them from the new Event-based SPC chart classes. The only difference you might see is an actual benefit. No matter what the time stamp is on a SPCTime... chart, adjacent points will always be equally spaced. So if your sample interval is irregular, or you even skip days or weeks in your sampling, the resulting chart will still display equally spaced adjacent sample records. You can read more about Event coordinates in the QCChart2D manual which is found in the same folder as QCSPCChart manual. **Furthermore**, if you are using the SPCTimeVariableControlChart class, we recommend that you instead use either the SPCBatchVariableControl (or SPCEventVariableControlChart class. If you want to see time/date values on the x-axis, set the XAxisStringLabelMode of the chart to AXIS_LABEL_MODE_TIME.

```
this.XAxisStringLabelMode = SPCChartObjects.AXIS_LABEL_MODE_TIME
```

The new class heirarchy looks like:

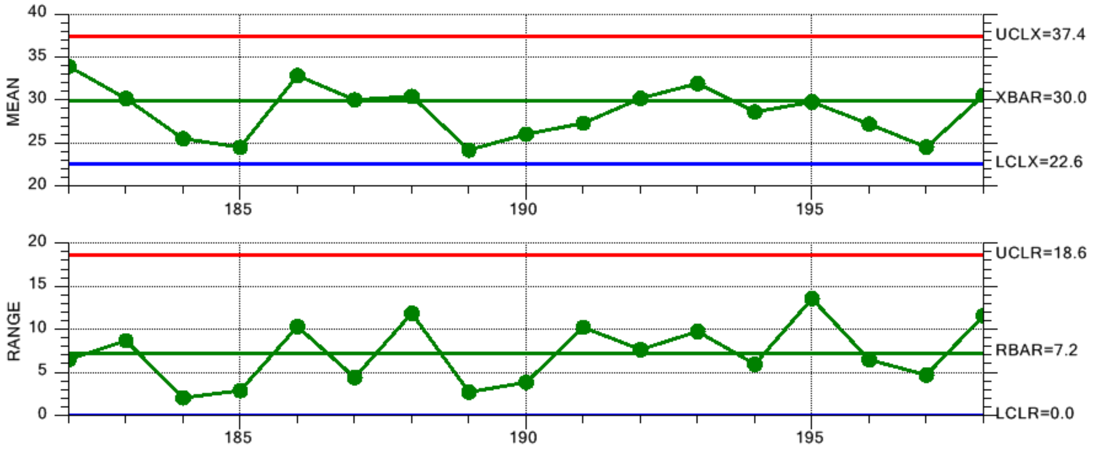
```
ChartView
        SPCChartBase
                SPCEventVariableControlChart
                        SPCTimeVariableControlChart
                        SPCBatchVariableControlChart
                SPCEventAttributeControlChart
                        SPCTimeAttributeControlChart
                        SPCBatchAttributeControlChart
```

Time-Based Variable Control Chart



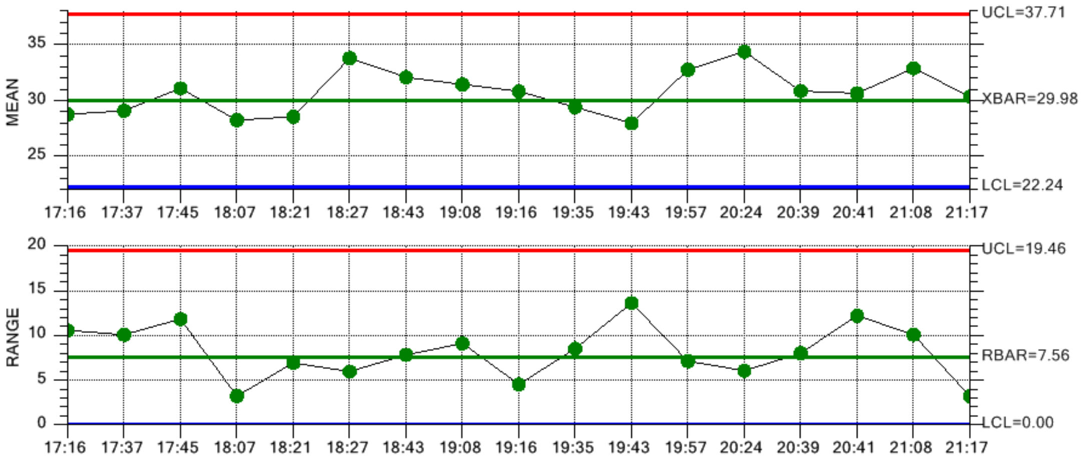
Note the time-based x-axis for both charts.

Batch-Based Variable Control Chart with numeric x-axis



Note the numeric based x-axis for both graphs

Batch-Based Variable Control Chart with time stamp x-axis



Note that even though the time stamp values do not have consistent time interval, the data points are spaced evenly by batch number.

Creating a Variable Control Chart

First, select whether you want to use a time-based variable control chart (use **SPCTimeVariableControlChart**), a batch-based variable control chart (use **SPCBatchVariableControlChart**), or an event-based chart (**SPCEventVariableControlChart**). Use that class as the base class for your chart. Since the classes are so similar and share 95% of all properties in common, only the **SPCEventVariableControlChart** is discussed in detail, with the differences between the two classes discussed at the end of the chapter. The example below is extracted from the **TimeVariableControlCharts.XBarRChart** example program.

[C#]

```
public class XBarRChart : SPCEventVariableControlChart
{
    ChartCalendar startTime = new ChartCalendar();

    //  SPC variable control chart type
    int charttype = SPCControlChartData.MEAN_RANGE_CHART;
    // Number of samples per sub group
    int numsamplespersubgroup = 5;
    // Number of data points  in the view
    int numdatapointsinview = 17;
    // The time increment between adjacent subgroups
    int timeincrementminutes = 15;

    public XBarRChart()
    {
        // This call is required by the Windows.Forms Form Designer.
        InitializeComponent();
        // Have the chart fill parent client area
        this.Dock = DockStyle.Fill;
        // Define and draw chart
        InitializeChart();
    }


    public void InitializeChart()
    {
        // Initialize the SPCEventVariableControlChart
        this.InitSPCEventVariableControlChart(charttype,
            numsamplespersubgroup, numdatapointsinview);
        .
        .
        .
        this.RebuildChartUsingCurrentData();
    }
}
```

[VB]

```
Public Class XBarRChart
    Inherits com.quinn-curtis.spcchartnet6.SPCEventVariableControlChart
    Private startTime As ChartCalendar = New ChartCalendar()
    '  SPC variable control chart type
    Private charttype As Integer = SPCControlChartData.MEAN_RANGE_CHART
    ' Number of samples per sub group
    Private numsamplespersubgroup As Integer = 5
    ' Number of data points  in the view
    Private numdatapointsinview As Integer = 17
    ' The time increment between adjacent subgroups
    Private timeincrementminutes As Integer = 15

#Region " Windows Form Designer generated code "

    Public Sub New()
        MyBase.New()

        'This call is required by the Windows Form Designer.
        InitializeComponent()

        'Add any initialization after the InitializeComponent() call
        Me.Dock = DockStyle.Fill
        ' Define and draw chart
        InitializeChart()
    End Sub

    .
    .
    .
#End Region


Public Sub InitializeChart()
  ' Initialize the SPCEventVariableControlChart
  Me.InitSPCTimeVariableControlChart(charttype, numsamplespersubgroup, _
   numdatapointsinview)

  ' Rebuild the chart using the current data and settings
  Me.RebuildChartUsingCurrentData()
End Sub 'InitializeChart
```

SPCEventVariableControlChart Members

Public Instance Constructors

SPCEventVariableControlChart	Overloaded. Initializes a new instance of the SPCEventVariableControlChart class.

Public Instance Methods

InitSPCEventVariableControlChart	Overloaded. Initialize the class for a specific SPC chart type.
InitSPCEventCusumControlChart	Overloaded. Initialize the class a cusum chart type.

The control chart type (X-Bar R, Median Range, X-Bar Sigma, Individual Range, EWMA, MA) establishes the variable control charts **InitSPCEventVariableControlChart** initialization routine. Note that the X-Bar Sigma chart, with a variable subgroup sample size, is initialized using **InitSPCEventVariableControlChart** with a *charttype* value of MEAN_SIGMA_CHART_VSS. X-Bar Sigma charts with sub groups that use a variable sample size must be updated properly. See the section “Adding New Sample Records to a X-Bar Sigma Chart (Variable Subgroup Sample Size)” in the “SPC Control Data and Alarm Classes” chapter.

SPCEventVariableControlChart.InitSPCEventVariableControlChart Method

This initialization method initializes the most important values in the creation of a SPC chart. If you are using the creating a cusum chart (type TABCUSUM_CHART), your can use the similar **InitSPCEventCusumControlChart** method instead. That version of the Init routine has added parameters for the H and K value of the tabular cusum chart.

[VB]

```
Overloads Public Sub InitSPCEventVariableControlChart( _
   ByVal charttype As Integer, _
   ByVal numsamplespersubgroup As Integer, _
   ByVal numdatapointsinview As Integer
)
```

```
[C#]
public void InitSPCEventVariableControlChart(
   int charttype,
   int numsamplespersubgroup,
   int numdatapointsinview
);
```

Parameters

charttype

The SPC chart type parameter. Use one of the SPCControlChartData SPC chart types: MEAN_RANGE_CHART, MEDIAN_RANGE_CHART, MEAN_SIGMA_CHART, MEAN_SIGMA_CHART_VSS, INDIVIDUAL_RANGE_CHART, EWMA_CHART, LEVEY_JENNINGS_CHART, MA_CHART, MAMR_CHART, MAMS_CHART and TABCUSUM_CHART.

numsamplespersubgroup

Specifies the number of samples that make up a sample subgroup.

numdatapointsinview

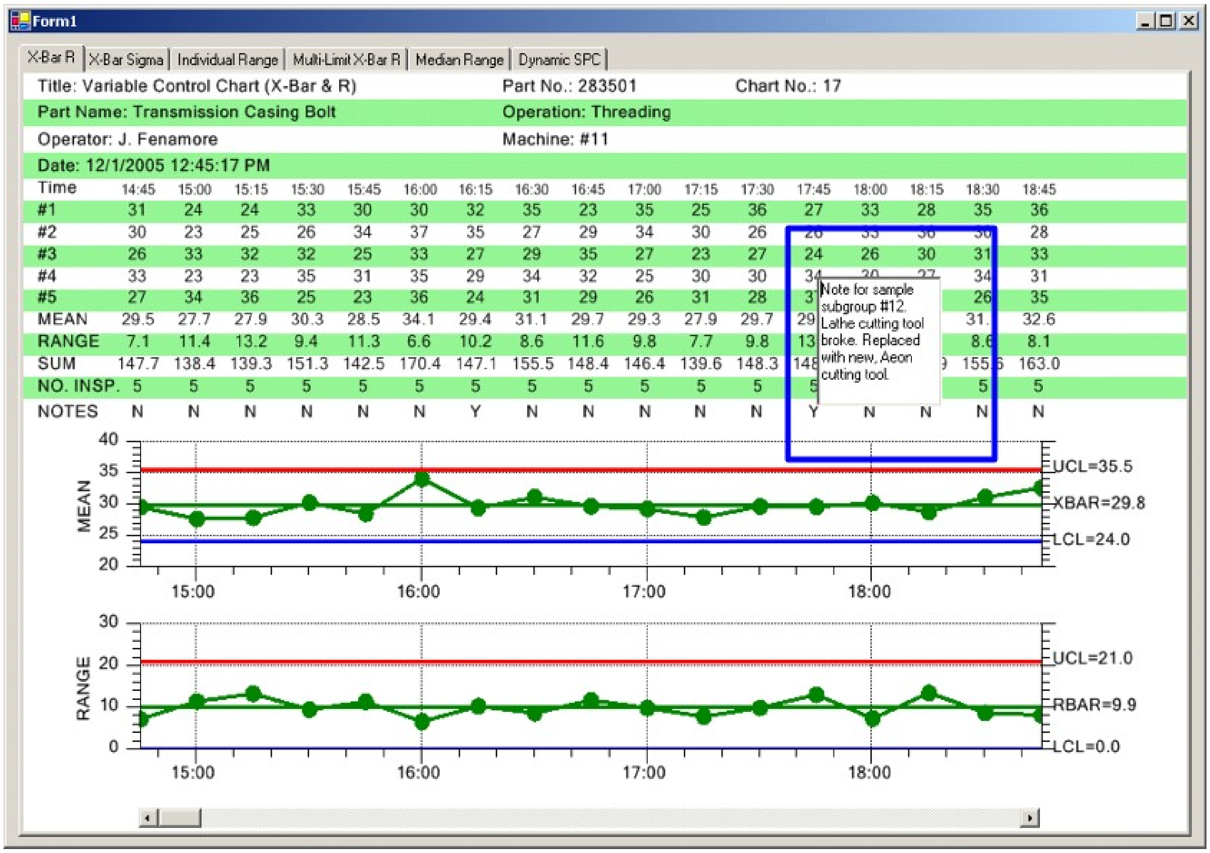
Specifies the number of sample subgroups displayed in the graph at one time.

The *timeincrementminutes* was found in the original intSPCTimeVariableControlChart method, but is not present in the initSPCEventVariableControlChart method

timeincrementminutes

Specifies the normal time increment between adjacent subgroup samples.

The image below further clarifies how these parameters affect the variable control chart.



Title: Variable Control Chart (X-Bar & R) Part No.: 283501 Chart No.: 17

Part Name: Transmission Casing Bolt Operation: Threading

Operator: J. Fenamore Machine: #11

Date: 12/1/2005 12:45:17 PM

Time	14:45	15:00	15:15	15:30	15:45	16:00	16:15	16:30	16:45	17:00	17:15	17:30	17:45	18:00	18:15	18:30	18:45
#1	31	24	24	33	30	30	32	35	23	35	25	36	27	33	28	35	36
#2	30	23	25	26	34	37	35	27	29	34	30	26	26	33	30	30	28
#3	26	33	32	32	25	33	27	29	35	27	23	27	24	26	30	31	33
#4	33	23	23	35	31	35	29	34	32	25	30	30	34	20	27	34	31
#5	27	34	36	25	23	36	24	31	29	26	31	28	3	[illegible]	[illegible]	26	35
MEAN	29.5	27.7	27.9	30.3	28.5	34.1	29.4	31.1	29.7	29.3	27.9	29.7	29	[illegible]	[illegible]	31.	32.6
RANGE	7.1	11.4	13.2	9.4	11.3	6.6	10.2	8.6	11.6	9.8	7.7	9.8	13	[illegible]	[illegible]	8.6	8.1
SUM	147.7	138.4	139.3	151.3	142.5	170.4	147.1	155.5	148.4	146.4	139.6	148.3	148	[illegible]	9	155.6	163.0
NO. INSP.	5	5	5	5	5	5	5	5	5	5	5	5	5	[illegible]	[illegible]	5	5
NOTES	N	N	N	N	N	N	Y	N	N	N	N	N	Y	N	N	N	N

Note for sample subgroup #12. Lathe cutting tool broke. Replaced with new, Aeon cutting tool.

Once the Init routine is called, the chart can be further customized using properties inherited from **SPCBaseChart**, described below.

Public Static (Shared) Properties

DefaultChartFontString	Set/Get the default font used in the table display.

Public Instance Constructors

SPCChartBase	Overloaded. Initializes a new instance of the SPCChartBase class.

Public Instance Properties

AutoLogAlarmsAsNotes	Set to true to automatically log alarms in the notes log.
BottomLabelMargin	Get/Set an additional margin, in normalized coordinates, if only the primary graphs is displayed, allowing for the x-axis labels
ChartAlarmEmphasisMode	Set to SPCChartBaseALARM_HIGHLIGHT_SYMBOL to highlight the process variable symbol if an alarm condition exists. Set to Set to

	SPCChartBase.ALARM_NO_HIGHLIGHT_SYMBOL to turn off alarm highlighting.
ChartData	Get the object that holds the descriptive text, sampled and calculated values associated with the control chart.
ChartInitialized	Returns true if the control chart has been initialized at least once.
ChartTable	Get the object that holds the data table information needed to display the data table along with the chart
DefaultControlLimitSigma	Set/Get that SPC control limits are to be calculated using the 3 sigma level standard.
EnableAlarmStatusValues	If set true enables the alarm status row of the chart table.
EnableCategoryValues	If set true enables the category or sample values rows of the data table
EnableDataToolTip	If set true enables data tooltips
EnableInputStringsDisplay	If set true enables the input string rows of the data table
EnableNotes	If set true enables the notes row of the data table
EnableNotesToolTip	If set true enables data tooltips
EnableScrollBar	If set true the scroll bar is added to the bottom of the chart.
EnableTimeValues	If set true enables the time row of the data table
EnableTotalSamplesValues	If set true enables the total of sampled values row of the data table
GraphBottomPos	Get/Set the bottom edge, using normalized coordinates, of the plotting area for the secondary chart
GraphStartPosX	Get/Set the left edge, using normalized coordinates, of the plotting area for both primary and secondary charts
GraphStartPosY1	Get the top edge, using normalized coordinates, of the plotting area for the primary chart
GraphStartPosY2	Get the top edge, using normalized coordinates, of the plotting area for the secondary chart
GraphStopPosX	Get/Set the right edge, using normalized coordinates, of the plotting area for both primary and secondary charts
GraphStopPosY1	Get the bottom edge, using normalized coordinates, of the plotting area for the primary chart
GraphStopPosY2	Get the bottom edge, using normalized coordinates, of the plotting area for the secondary chart
GraphTopTableOffset	Get/Set the offset of the top of the primary chart from the bottom of the data table, using normalized coordinates
HeaderStringsLevel	Set/Get the level of header strings to include in the chart. Use one of the SPCControlChartData header strings constants: HEADER_STRINGS_LEVEL0, HEADER_STRINGS_LEVEL1, HEADER_STRINGS_LEVEL2, or HEADER_STRINGS_LEVEL3

InterGraphMargin	Get/Set the margin, in normalized coordinates, between the primary and secondary charts
MultipleMouseListener	Set/Get the MultiMouseListener.
PrimaryChart	Get the object that holds he the chart objects needed to display the primary chart
ScrollBarBottomPosition	Get/Set the bottom edge, using normalized coordinates, of the optional scroll bar
ScrollBarPixelHeight	Get/Set the height of the scrollbar in pixels
SecondaryChart	Get the object that holds he the chart objects needed to display the secondary chart
SPCChartType	Specifies the control chart type: use one of the SPCControlChartData chart type constants: MEAN_RANGE_CHART, MEDIAN_RANGE_CHART, MEAN_SIGMA_CHART, MEAN_SIGMA_CHART_VSS, INDIVIDUAL_RANGE_CHART, EWMA_CHART, LEVEY_JENNINGS_CHART, MA_CHART, MAMR_CHART, MAMS_CHART, TABCUSUM_CHART, CUSTOM_ATTRIBUTE_CONTROL_CHART, PERCENT_DEFECTIVE_PARTS_CHART, FRACTION_DEFECTIVE_PARTS_CHART, NUMBER_DEFECTIVE_PARTS_CHART, NUMBER_DEFECTS_CHART, NUMBER_DEFECTS_PERUNIT_CHART.
TableAlarmEmphasisMode	Set the table alarm highlighting to one of the SPCChartBase table highlight constants: ALARM_HIGHLIGHT_NONE, ALARM_HIGHLIGHT_TEXT, ALARM_HIGHLIGHT_OUTLINE, ALARM_HIGHLIGHT_BAR
TableStartPosY	Get the top edge, using normalized coordinates, of the SPC chart table
XScaleMode	Set/Get whether the x-axis is time based, or numeric based.

Public Instance Methods

AddAnnotation	Overloaded. Add a simple annotation to a data point in the specified SPC chart.
AutoCalculateControlLimits	Using the current sampled data values, high, target and low control limits are calculated for both primary and secondary charts using an algorithm appropriate to the SPC chart type.
AutoCalculatePrimaryControlLimits	Using the current sampled data values, high, target and low control limits are calculated for the primary chart using an algorithm appropriate to the SPC chart type.

AutoCalculateSecondaryControlLimits	Using the current sampled data values, high, target and low control limits are calculated for the primary chart using an algorithm appropriate to the SPC chart type.
AutoScaleChartYRange	Auto-scale the y-range of the SPC chart so that all of the sampled data and chart control limits are within the bounds of the y-axis.
AutoScalePrimaryChartYRange	Auto-scale the y-range of the primary SPC chart so that all of the sampled data and chart control limits are within the bounds of the y-axis.
AutoScaleSecondaryChartYRange	Auto-scale the y-range of the SPC chart so that all of the sampled data and chart control limits are within the bounds of the y-axis.
Copy	Overloaded. Copies the source object.
Draw	Overrides the Draw method of the underlying ChartView class, so that the scroll bar can be properly repositioned if the size of the window changes. The graphics context the chart is drawn to.
InitSPCChartBase	This initialization method initializes the most important values in the creation of a SPC chart.
IsTimeScale	Returns true if the coordinate system has a time based x-axis. The coordinate system of the chart.
MakeControlLinePlot	Draw a control line, either a simple straight line, or a variable control line, for the specified chart.
RebuildChartUsingCurrentData	Rebuild the graph taking into account the most recent data values.
RescaleGraphsToScrollbar	Rescale primary and secondary charts based on the position of the value of the scroll bar. The thumb position of the scroll bar.
ResetSPCChartData	Reset the history buffers of all of the SPC data objects.
UpdateControlLimitLabel	Creates a numeric label of the control limit, and adds the numeric label to the spc chart.
UseNoTable	Specifies to create the primary and secondary charts without a table. Just the charts, chart title and optional histograms.

Adding New Sample Records for Variable Control Charts.

In variable control charts, each data value in the *samples* array represents a specific sample in the sample subgroup. In X-Bar R, X-Bar Sigma, and Median-Range charts, where the sample subgroup size is some fraction of the total production level, there is one value in the *samples* array for each measurement sample in the sample subgroup interval. If the production level is sixty items per hour, and the sample size is five items per hour, then the graph would be updated once an hour with five items in the *samples* array.

[C#]

```
DoubleArray samples = new DoubleArray(5);
//  ChartCalendar initialized with current  time by default
ChartCalendar timestamp = new ChartCalendar();
// Place sample values in array
samples[0] = 0.121;     // First of five samples
samples[1] = 0.212;     // Second of five samples
samples[2] = 0.322;     // Third of five samples
samples[3] = 0.021;     // Fourth of five samples
samples[4] = 0.133;     // Fifth of five samples

// Add the new sample subgroup to the chart
this.ChartData.AddNewSampleRecord(timestamp, samples);
```

[VB]

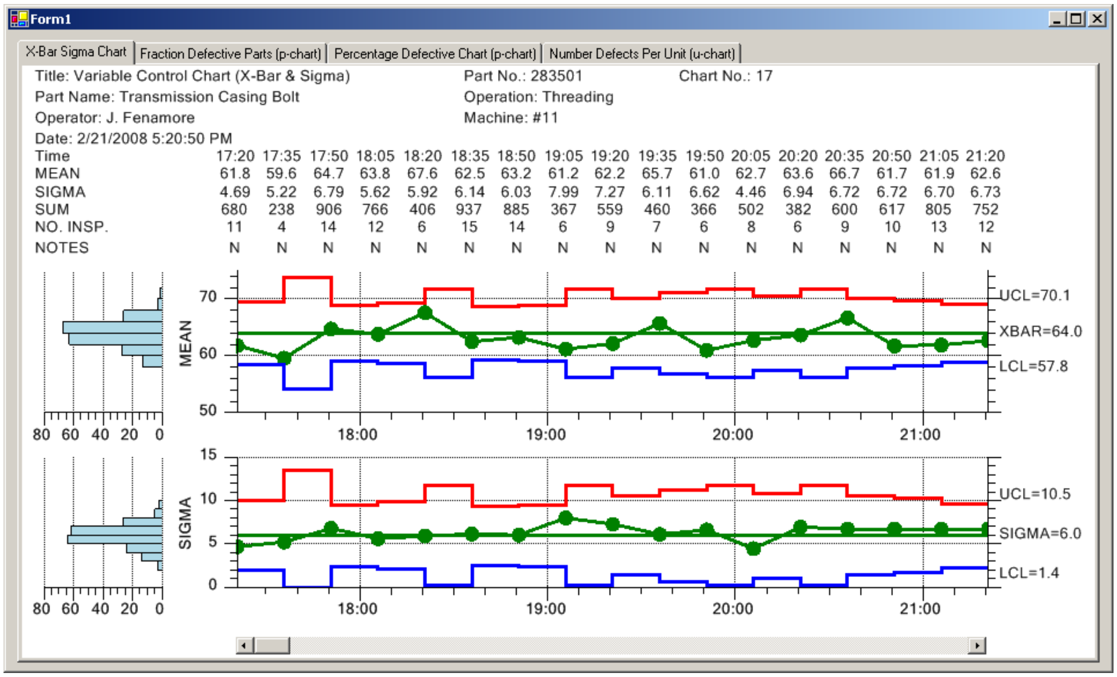
```
Dim samples As DoubleArray = New DoubleArray(5)
'  ChartCalendar initialized with current  time by default
Dim timestamp As ChartCalendar = New ChartCalendar()
' Place sample values in array
samples(0) = 0.121    ' First of five samples
samples(1) = 0.212    ' Second of five samples
samples(2) = 0.322    ' Third of five samples
samples(3) = 0.021    ' Fourth of five samples
samples(4) = 0.133    ' Fifth of five samples

' Add the new sample subgroup to the chart
 Me.ChartData.AddNewSampleRecord(timestamp, samples)
```

In an Individual-Range chart, and EWMA and MA charts that uses rational subgroup sizes of 1, the *samples* array would only have one value for each update. If the production level is sixty items per hour, with 100% sampling, the graph would be updated once a minute, with a single value in the *samples* array.

Updating MEAN_SIGMA_CHART_VSS with a variable number of samples per subgroup

The X-Bar Sigma chart also comes in a version where variable sample sizes are permitted, As in the standard variable control charts, there is one value in the *samples* array for each measurement sample in the sample subgroup interval. The difference is that the length of the *samples* array can change from update to update. It is critically import that the size of the samples array exactly matches the number of samples in the current subgroup

X-Bar Sigma Chart with variable sample size



Form1

X-Bar Sigma Chart | Fraction Defective Parts (p-chart) | Percentage Defective Chart (p-chart) | Number Defects Per Unit (u-chart)

Title: Variable Control Chart (X-Bar & Sigma) Part No.: 283501 Chart No.: 17
Part Name: Transmission Casing Bolt Operation: Threading
Operator: J. Fenamore Machine: #11
Date: 2/21/2008 5:20:50 PM

Time	17:20	17:35	17:50	18:05	18:20	18:35	18:50	19:05	19:20	19:35	19:50	20:05	20:20	20:35	20:50	21:05	21:20
MEAN	61.8	59.6	64.7	63.8	67.6	62.5	63.2	61.2	62.2	65.7	61.0	62.7	63.6	66.7	61.7	61.9	62.6
SIGMA	4.69	5.22	6.79	5.62	5.92	6.14	6.03	7.99	7.27	6.11	6.62	4.46	6.94	6.72	6.72	6.70	6.73
SUM	680	238	906	766	406	937	885	367	559	460	366	502	382	600	617	805	752
NO. INSP.	11	4	14	12	6	15	14	6	9	7	6	8	6	9	10	13	12
NOTES	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N

In this case, the control chart high and low limits vary from sample interval to sample interval, depending on the number of samples in the associated sample subgroup. You can read the sample sizes along the NO.INSP row in the data table above the chart. A low number of samples in the sample subgroup make the band between the high and low limits wider than if a higher number of samples are available. The X-Bar Sigma chart is the only variable control chart that can be used with a variable sample size.

[C#]

```
// GetCurrentSampleSubgroupSize is a fictional method that gets the
// current number of samples in the  sample subgroup. The value of N
// can vary from sample interval to sample interval. You must have a
// valid sample value for each element.

N = GetCurrentSampleSubgroupSize();

// Size array exactly to a length of N
DoubleArray samples = new DoubleArray(N);
//  ChartCalendar initialized with current  time by default
ChartCalendar timestamp = new ChartCalendar();

// Place sample values in array
// You must have a valid sample value for each element of the array size 0..N-1
samples[0] = 0.121;     // First of five samples
samples[1] = 0.212;     // Second of five samples
.
.
.
samples[N-1] = 0.133;       // Last of the samples in the sample subgroup

// Add the new sample subgroup to the chart
this.ChartData.AddNewSampleRecord(timestamp, samples);
```

[VB]

```
‘ GetCurrentSampleSubgroupSize is a fictional method that gets the
‘ current number of samples in the  sample subgroup. The value of N
‘ can vary from sample interval to sample interval. You must have a
‘ valid sample value for each element.

N = GetCurrentSampleSubgroupSize()

‘ Size array exactly to a length of N
Dim samples As DoubleArray = New DoubleArray(N)
'  ChartCalendar initialized with current  time by default
Dim timestamp As ChartCalendar = New ChartCalendar()
' Place sample values in array
samples(0) = 0.121    ' First of five samples
samples(1) = 0.212    ' Second of five samples
.
.
.
samples(N-1) = 0.133    ' Last of the samples in the sample subgroup

' Add the new sample subgroup to the chart
 Me.ChartData.AddNewSampleRecord(timestamp, samples)
```

Measured Data and Calculated Value Tables

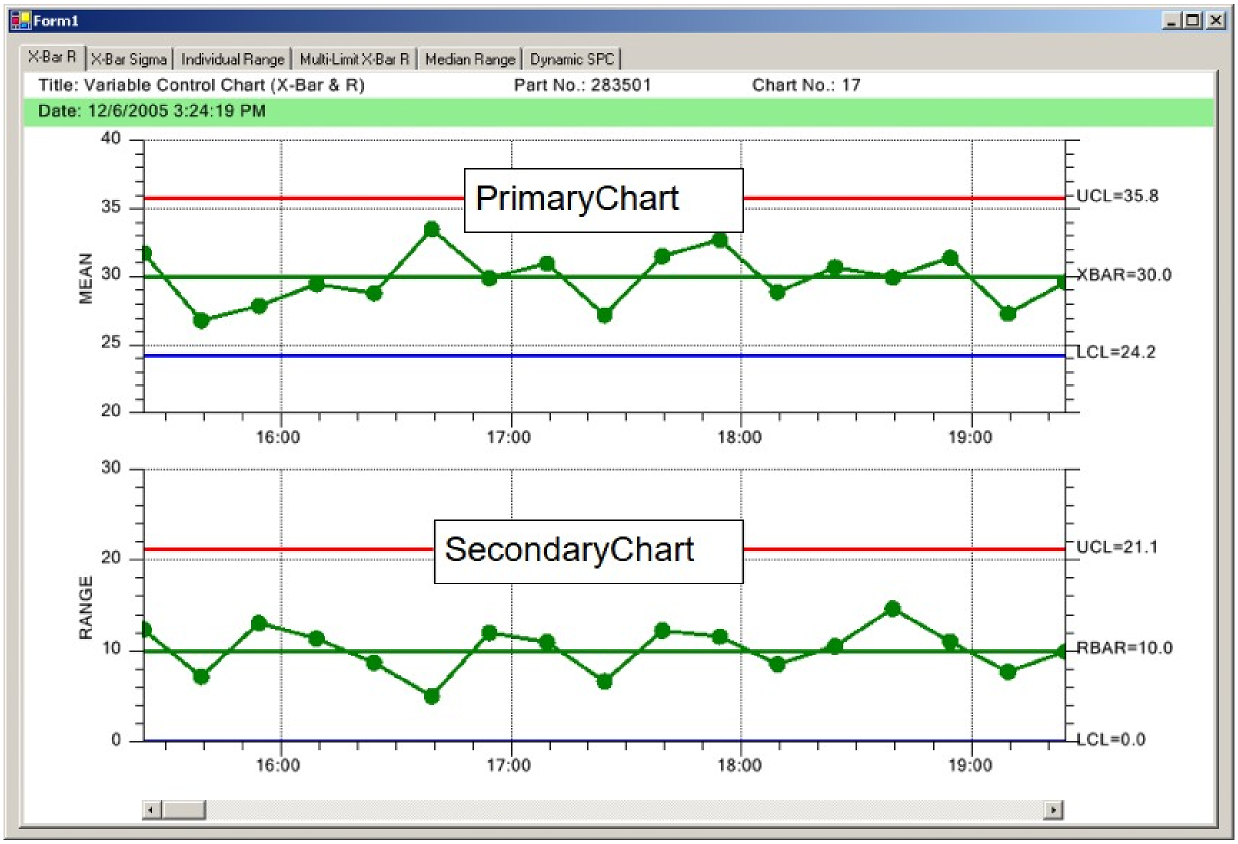
Standard worksheets used to gather and plot SPC data consist of three main parts.

- The first part is the header section, identifying the title of the chart, the monitored process, the machine operator, part number and other important information specific to the chart.
- The second part is the measurement data recording and calculation section, organized as a table where the sample data and calculated values are recorded in a neat, readable fashion.
- The third part plots the calculated SPC values for the sample group variables as a SPC chart.

The chart includes options that enable the programmer to customize and automatically include header information along with a table of the measurement and calculated data, in the SPC chart.

The following properties enable sections of the chart header and table:
:

EnableInputStringsDisplay
EnableCategoryValues
EnableCalculatedValues
EnableTotalSamplesValues
EnableNotes
EnableTotalSamplesValues
EnableTimeValues
EnableProcessCapabilityValues



Title: Variable Control Chart (X-Bar & R) Part No.: 283501 Chart No.: 17

Date: 12/6/2005 3:24:19 PM

In the program the code looks like the following code extracted from the TimeVariableControlCharts.XBarRChart example program

[C#]

```
// Change the default horizontal position and width of the chart
this.GraphStartPosX = 0.1; // start here
this.GraphStopPosX = 0.875;  // end here

// Set the strings used in the header section of the table
this.ChartData.Title = "Variable Control Chart (X-Bar & R)";
this.ChartData.PartNumber = "283501";
this.ChartData.ChartNumber="17";
this.ChartData.PartName= "Transmission Casing Bolt";
this.ChartData.Operation = "Threading";
this.ChartData.SpecificationLimits="";
this.ChartData.TheOperator="J. Fenamore";
this.ChartData.Machine="#11";
this.ChartData.Gage="#8645";
this.ChartData.UnitOfMeasure = "0.0001 inch";
this.ChartData.ZeroEquals="zero";
this.ChartData.DateString = DateTime.Now.ToString();
this.ChartData.NotesMessage = "Control limits prepared May 10";
this.ChartData.NotesHeader = "NOTES"; // row header
    // Set initial scale of the y-axis of the mean chart
// If you are calling AutoScalePrimaryChartYRange this isn't really needed
this.PrimaryChart.MinY = 0;
this.PrimaryChart.MaxY = 40;

// Set initial scale of the y-axis of the range chart
// If you are calling AutoScaleSecondaryChartYRange this isn't really needed
```

```
this.SecondaryChart.MinY = 0;
this.SecondaryChart.MaxY = 40;

// Display the Sampled value rows of the table
this.EnableInputStringsDisplay= true;
// Display the Sampled value rows of the table
this.EnableCategoryValues= true;
// Display the Calculated value rows of the table
this.EnableCalculatedValues= true;
// Display the total samples per subgroup value row
this.EnableTotalSamplesValues= true;
// Display the Notes row of the table
this.EnableNotes= true;
// Display the time stamp row of the table
this.EnableTimeValues = true;
```

[VB]

```
' Change the default horizontal position and width of the chart
Me.GraphStartPosX = 0.1 ' start here
Me.GraphStopPosX = 0.875 ' end here
' Set the strings used in the header section of the table
Me.ChartData.Title = "Variable Control Chart (X-Bar & R)"
Me.ChartData.PartNumber = "283501"
Me.ChartData.ChartNumber = "17"
Me.ChartData.PartName = "Transmission Casing Bolt"
Me.ChartData.Operation = "Threading"
Me.ChartData.SpecificationLimits = ""
Me.ChartData.TheOperator = "J. Fenamore"
Me.ChartData.Machine = "#11"
Me.ChartData.Gage = "#8645"
Me.ChartData.UnitOfMeasure = "0.0001 inch"
Me.ChartData.ZeroEquals = "zero"
Me.ChartData.DateString = DateTime.Now.ToString()
Me.ChartData.NotesMessage = "Control limits prepared May 10"
Me.ChartData.NotesHeader = "NOTES" ' row header
' Set initial scale of the y-axis of the mean chart
' If you are calling AutoScalePrimaryChartYRange this isn't really needed
Me.PrimaryChart.MinY = 0
Me.PrimaryChart.MaxY = 40

' Set initial scale of the y-axis of the range chart
' If you are calling AutoScaleSecondaryChartYRange this isn't really needed
Me.SecondaryChart.MinY = 0
Me.SecondaryChart.MaxY = 40

' Display the Sampled value rows of the table
Me.EnableInputStringsDisplay = True
' Display the Sampled value rows of the table
Me.EnableCategoryValues = True
' Display the Calculated value rows of the table
Me.EnableCalculatedValues = True
' Display the total samples per subgroup value row
Me.EnableTotalSamplesValues = True
‘ Display the Notes row of the table
Me.EnableNotes = True
' Display the time stamp row of the table
Me.EnableTimeValues = True
```

Process Capability Ratios and Process Performance Indices

The data table also displays any process capability statistics that you want to see. The software supports the calculation and display of the Cp, Cpl, Cpu, Cpk, Cpm, Pp, Ppu, Ppl, and Ppk process capability statistics.

In order to display process capability statistics you must first specify the process specification limits that you want the calculations based on. These are not the high and low SPC control limits calculate by this software; rather they externally calculated limits based on the acceptable tolerances allowed for the process under measure. Set the lower specification limit (LSL) and upper specification limit (USL) using the **ChartData.ProcessCapabilityLSLValue** and **ChartData.ProcessCapabilityUSLValue** properties of the chart. The code below is from the TimeVariableControlCharts.XBarRChart example.

```
this.ChartData.ProcessCapabilityLSLValue = 27;
this.ChartData.ProcessCapabilityUSLValue = 35;
```

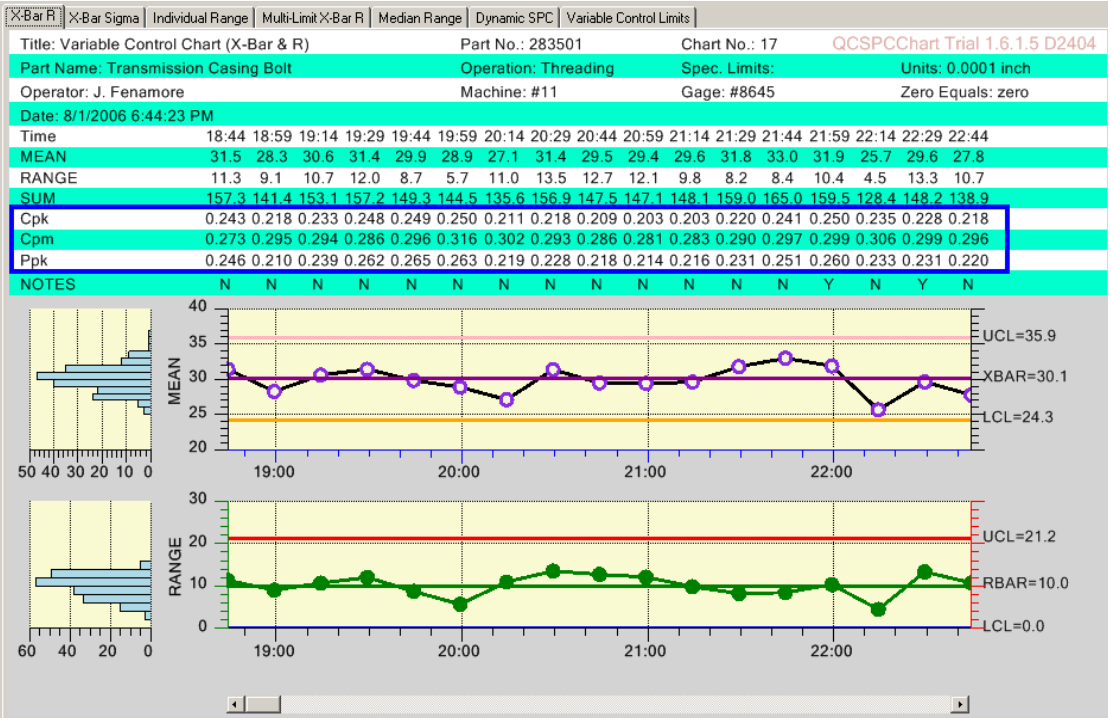
Use the **ChartData.addProcessCapabilityValue** method to specify exactly which process capability statistics you want to see in the table. Use one of the **SPCProcessCapabilityRecord** constants below to specify the statistics that you want displayed.

SPC_CP_CALC	Constant value Cp calculation
SPC_CPL_CALC	Constant value Cpl calculation.
SPC_CPU_CALC	Constant value Cpu calculation.
SPC_CPK_CALC	Constant value Cpk calculation.
SPC_CPM_CALC	Constant value Cpm calculation.
SPC_PP_CALC	Constant value Pp calculation.
SPC_PPL_CALC	Constant value Ppl calculation.
SPC_PPU_CALC	Constant value Ppu calculation.
SPC_PPK_CALC	Constant value PPK calculation.

The code below is from the TimeVariableControlCharts.XBarRChart example.

```
this.ChartData.AddProcessCapabilityValue(SPCProcessCapabilityRecord.SPC_CPK_CALC);
this.ChartData.AddProcessCapabilityValue(SPCProcessCapabilityRecord.SPC_CPM_CALC);
this.ChartData.AddProcessCapabilityValue(SPCProcessCapabilityRecord.SPC_PPK_CALC);
```

This selection will add three rows to the data table, one row each for the Cpk, Cpm and Ppk process capability statistics. Once these steps are carried out, the calculation and display of the statistics is automatic.



Title: Variable Control Chart (X-Bar & R) Part No.: 283501 Chart No.: 17

Part Name: Transmission Casing Bolt Operation: Threading Spec. Limits: Units: 0.0001 inch

Operator: J. Fenamore Machine: #11 Gage: #8645 Zero Equals: zero

Date: 8/1/2006 6:44:23 PM

Time	18:44	18:59	19:14	19:29	19:44	19:59	20:14	20:29	20:44	20:59	21:14	21:29	21:44	21:59	22:14	22:29	22:44
MEAN	31.5	28.3	30.6	31.4	29.9	28.9	27.1	31.4	29.5	29.4	29.6	31.8	33.0	31.9	25.7	29.6	27.8
RANGE	11.3	9.1	10.7	12.0	8.7	5.7	11.0	13.5	12.7	12.1	9.8	8.2	8.4	10.4	4.5	13.3	10.7
SUM	157.3	141.4	153.1	157.2	149.3	144.5	135.6	156.9	147.5	147.1	148.1	159.0	165.0	159.5	128.4	148.2	138.9
Cpk	0.243	0.218	0.233	0.248	0.249	0.250	0.211	0.218	0.209	0.203	0.203	0.220	0.241	0.250	0.235	0.228	0.218
Cpm	0.273	0.295	0.294	0.286	0.296	0.316	0.302	0.293	0.286	0.281	0.283	0.290	0.297	0.299	0.306	0.299	0.296
Ppk	0.246	0.210	0.239	0.262	0.265	0.263	0.219	0.228	0.218	0.214	0.216	0.231	0.251	0.260	0.233	0.231	0.220
NOTES	N	N	N	N	N	N	N	N	N	N	N	N	N	Y	N	Y	N

Formulas Used in Calculating the Process Capability Ratios

The formulas used in calculating the process capability statistics vary. We use the formulas found in the textbook. “Introduction to Statistical Quality Control” by Douglas C. Montgomery, John Wiley and Sons, Inc. 2001.

SPC Control Chart Nomenclature

USL = Upper Specification Limit

LSL = Lower Specification Limit

Tau = Midpoint between USL and LSL = ½ * (LSL + USL)

$\bar{\bar{X}}$ = XDoubleBar - Mean of sample subgroup means (also called the grand average)

$\bar{R}$ = RBar – Mean of sample subgroup ranges

S = Sigma – sample standard deviation – all samples from all subgroups are used to calculate the standard deviation S.

$\overline{S}$ = SigmaBar – Average of sample subgroup sigma's. Each sample subgroup has a calculated standard deviation and the SigmaBar value is the mean of those subgroup standard deviations.

d2 = a constant tabulated in every SPC textbook for various sample sizes.

By convention, the quantity RBar/d2 is used to estimate the process sigma for the Cp, Cpl and Cpu calculations

MINIMUM – a function that returns the lesser of two arguments

SQRT – a function returning the square root of the argument.

Process Capability Ratios (Cp, Cpl, Cpu, Cpk and Cpm)

Cp = (USL – LSL) / (6 * RBar/d2)

Cpl = (XDoubleBar – LSL) / (3 * RBar/d2)

Cpu = (USL - XDoubleBar) / (3 * RBar/d2)

Cpk = MINIMUM (Cpl, Cpu)

Cpm = Cp / (SQRT(1 + V^2)

where

V = (XDoubleBar – Tau) / S

Process Performance Indices (Pp, Ppl, Ppu, Ppk)

Pp = (USL – LSL) / (6 * S)

Ppl = (XDoubleBar – LSL) / (3 * S)

Ppu = (USL - XDoubleBar) / (3 *S)

Ppk = MINIMUM (Ppl, Ppu)

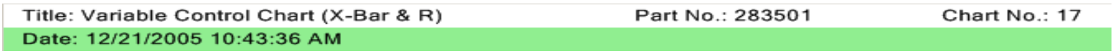
The major difference between the Process Capability Ratios (Cp, Cpl, Cpu, Cpk) and the Process Performance Indices (Pp, Ppl, Ppu, Ppk) is the estimate used for the process sigma. The Process Capability Ratios use the estimate (RBar/d2) and the Process Performance Indices uses the sample standard deviation S. If the process is in control, then Cp vs Pp and Cpk vs Ppk should returns approximately the same values, since both (RBar/d2) and the sample sigma S will be good estimates of the overall process sigma. If the process is NOT in control, then ANSI (American National Standards Institute) recommends that the Process Performance Indices (Pp, Ppl, Ppu, Ppk) be used.

Table Strings

The input header strings display has four sub-levels that display increasing levels of information. The input header strings display level is set using the charts HeaderStringsLevel property. Strings that can be displayed are: Title, PartNumber, ChartNumber, DateString, PartName, Operation, Machine, SpecificationLimits, Gage, UnitOfMeasure, ZeroEquals and DateString. The four levels and the information displayed is listed below:

HEADER_STRINGS_LEVEL0	Display no header information
HEADER_STRINGS_LEVEL1	Display minimal header information: Title, PartNumber, ChartNumber, DateString
HEADER_STRINGS_LEVEL2	Display most header strings: Title, PartNumber, ChartNumber, PartName, Operation, Operator, Machine, DateString
HEADER_STRINGS_LEVEL3	Display all header strings: Title, PartNumber, ChartNumber, DateString, PartName, Operation, Machine, SpecificationLimits, Gage, UnitOfMeasure, ZeroEquals and DateString

The example program TimeVariableControlCharts.XBarRChart demonstrates the use of the **HeaderStringsLevel** property. The example below displays a minimum set of header strings (HeaderStringsLevel = SPCControlChartData.HEADER_STRINGS_LEVEL1).



Title: Variable Control Chart (X-Bar & R) Part No.: 283501 Chart No.: 17
Date: 12/21/2005 10:43:36 AM

[C#]

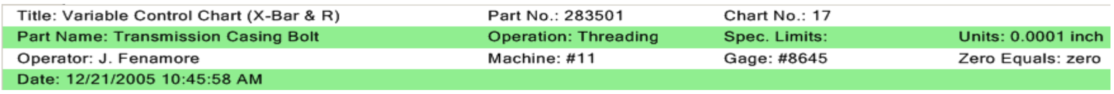
```
// Set the strings used in the header section of the table
```

```
this.ChartData.Title = "Variable Control Chart (X-Bar & R)";
this.ChartData.PartNumber = "283501";
this.ChartData.ChartNumber="17";
this.ChartData.DateString = DateTime.Now.ToString();

this.HeaderStringsLevel = SPCControlChartData.HEADER_STRINGS_LEVEL1;
```

[VB]

```
' Set the strings used in the header section of the table
Me.ChartData.Title = "Variable Control Chart (X-Bar & R)"
Me.ChartData.PartNumber = "283501"
Me.ChartData.ChartNumber = "17"
Me.ChartData.DateString = DateTime.Now.ToString()

Me.HeaderStringsLevel = SPCControlChartData.HEADER_STRINGS_LEVEL1
```

The example below displays a maximum set of header strings (HeaderStringsLevel = SPCControlChartData.HEADER_STRINGS_LEVEL3).



Title: Variable Control Chart (X-Bar & R)	Part No.: 283501	Chart No.: 17	
Part Name: Transmission Casing Bolt	Operation: Threading	Spec. Limits:	Units: 0.0001 inch
Operator: J. Fenamore	Machine: #11	Gage: #8645	Zero Equals: zero
Date: 12/21/2005 10:45:58 AM			

[C#]

```
// Set the strings used in the header section of the table
this.ChartData.Title = "Variable Control Chart (X-Bar & R)";
this.ChartData.PartNumber = "283501";
this.ChartData.ChartNumber="17";
this.ChartData.PartName= "Transmission Casing Bolt";
this.ChartData.Operation = "Threading";
this.ChartData.SpecificationLimits="";
this.ChartData.TheOperator="J. Fenamore";
this.ChartData.Machine="#11";
this.ChartData.Gage="#8645";
this.ChartData.UnitOfMeasure = "0.0001 inch";
this.ChartData.ZeroEquals="zero";
this.ChartData.DateString = DateTime.Now.ToString();
this.HeaderStringsLevel = SPCControlChartData.HEADER_STRINGS_LEVEL3;
```

[VB]

```
' Set the strings used in the header section of the table
Me.ChartData.Title = "Variable Control Chart (X-Bar & R)"
Me.ChartData.PartNumber = "283501"
Me.ChartData.ChartNumber = "17"
Me.ChartData.PartName = "Transmission Casing Bolt"
Me.ChartData.Operation = "Threading"
Me.ChartData.SpecificationLimits = ""
Me.ChartData.TheOperator = "J. Fenamore"
Me.ChartData.Machine = "#11"
Me.ChartData.Gage = "#8645"
Me.ChartData.UnitOfMeasure = "0.0001 inch"
Me.ChartData.ZeroEquals = "zero"
Me.ChartData.DateString = DateTime.Now.ToString()
Me.HeaderStringsLevel = SPCControlChartData.HEADER_STRINGS_LEVEL3
```

The identifying string displayed in front of the input header string can be any string that you want, including non-English language strings. For example, if you want the input header string for the Title to represent a project name:

Project Name: Project XKYZ for PerQuet

Set the properties:

[C#]

```
this.ChartData.Title = "Project XKYZ for PerQuet";
this.ChartData.TitleHeader = "Project Name:";
```

[VB]

```
Me.ChartData.Title = "Project XKYZ for PerQuet"
Me.ChartData.TitleHeader = "Project Name:"
```

Change other header strings using the **ChartData** properties listed below.

- TitleHeader
- PartNumberHeader
- ChartNumberHeader
- PartNameHeader
- OperationHeader
- OperatorHeader
- MachineHeader
- DateHeader
- SpecificationLimitsHeader
- GageHeader
- UnitOfMeasureHeader
- ZeroEqualsHeader
- NotesHeader

Even though the input header string properties have names like Title, PartNumber, ChartNumber, etc., those names are arbitrary. They are really just placeholders for the strings that are placed at the respective position in the table. You can display any combination of strings that you want, rather than the ones we have selected by default, based on commonly used standardized SPC Control Charts.

Table Background Colors

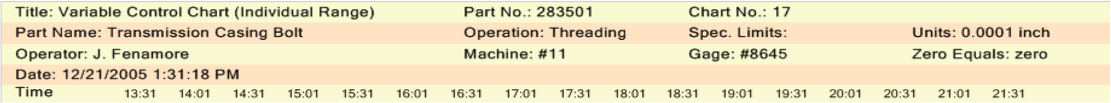
The **ChartTable** property of the chart has some properties that can further customize the chart. The default table background uses the accounting style green-bar striped background. You can change this using the **ChartTable.TableBackgroundMode** property**.** Set the value to one of the TableBackgroundMode constants in the class **SPCGeneralizedTableDisplay**:

TABLE_NO_COLOR_BACKGROUND	Constant specifies that the table does not use a background color.
TABLE_SINGLE_COLOR_BACKGROUND	Constant specifies that the table uses a single color for the background (backgroundColor1)

TABLE_STRIPED_COLOR_BACKGROUND Constant specifies that the table uses horizontal stripes of color for the background (backgroundColor1 and backgroundColor2)

TABLE_SINGLE_COLOR_BACKGROUND_GRIDCELL
Constant specifies that the table uses a grid background, with backgroundColor1 the overall background color and backgroundColor2 the color of the grid lines.

Extracted from the TimeVariableControlCharts.IndividualRangeChart example program



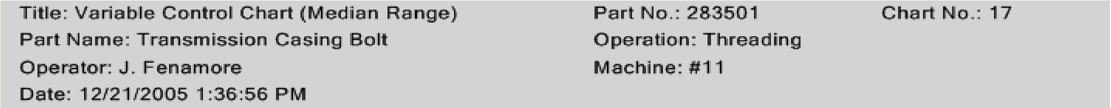
Title: Variable Control Chart (Individual Range)	Part No.: 283501	Chart No.: 17	
Part Name: Transmission Casing Bolt	Operation: Threading	Spec. Limits:	Units: 0.0001 inch
Operator: J. Fenamore	Machine: #11	Gage: #8645	Zero Equals: zero
Date: 12/21/2005 1:31:18 PM			

Time 13:31 14:01 14:31 15:01 15:31 16:01 16:31 17:01 17:31 18:01 18:31 19:01 19:31 20:01 20:31 21:01 21:31

[C#]
```
this.ChartTable.TableBackgroundMode =
    SPCGeneralizedTableDisplay.TABLE_STRIPED_COLOR_BACKGROUND;
this.ChartTable.BackgroundColor1 = Color.Bisque;
this.ChartTable.BackgroundColor2 = Color.LightGoldenrodYellow;
```

[VB]
```
Me.ChartTable.TableBackgroundMode = _
    SPCGeneralizedTableDisplay.TABLE_STRIPED_COLOR_BACKGROUND
Me.ChartTable.BackgroundColor1 = Color.Bisque
Me.ChartTable.BackgroundColor2 = Color.LightGoldenrodYellow
```

Extracted from the TimeVariableControlCharts.MedianRangeChart example program

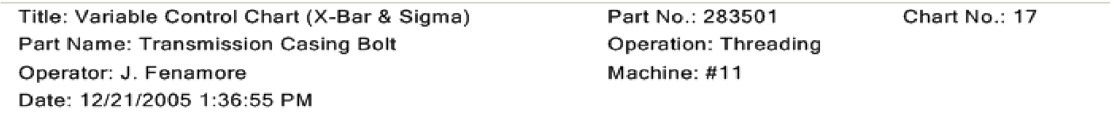


Title: Variable Control Chart (Median Range)	Part No.: 283501	Chart No.: 17
Part Name: Transmission Casing Bolt	Operation: Threading	
Operator: J. Fenamore	Machine: #11	
Date: 12/21/2005 1:36:56 PM		

[C#]
```
this.ChartTable.TableBackgroundMode =
    SPCGeneralizedTableDisplay.TABLE_SINGLE_COLOR_BACKGROUND;
this.ChartTable.BackgroundColor1 = Color.LightGray;
```

[VB]
```
Me.ChartTable.TableBackgroundMode = _
    SPCGeneralizedTableDisplay.TABLE_SINGLE_COLOR_BACKGROUND
Me.ChartTable.BackgroundColor1 = Color.LightGray
```

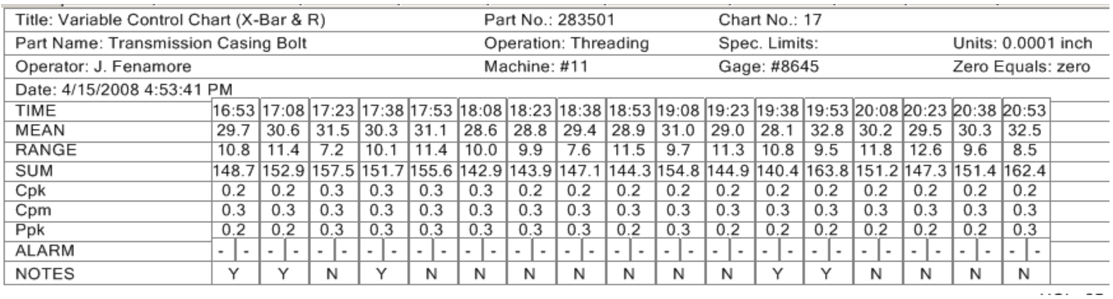
Extracted from the TimeVariableControlCharts.XBarSigma example program



Title: Variable Control Chart (X-Bar & Sigma)	Part No.: 283501	Chart No.: 17
Part Name: Transmission Casing Bolt	Operation: Threading	
Operator: J. Fenamore	Machine: #11	
Date: 12/21/2005 1:36:55 PM		

[C#]
```
this.ChartTable.TableBackgroundMode =
       SPCGeneralizedTableDisplay.TABLE_NO_COLOR_BACKGROUND;
```

[VB]
```
Me.ChartTable.TableBackgroundMode = _
       SPCGeneralizedTableDisplay.TABLE_NO_COLOR_BACKGROUND
```



Title: Variable Control Chart (X-Bar & R) Part No.: 283501 Chart No.: 17
Part Name: Transmission Casing Bolt Operation: Threading Spec. Limits: Units: 0.0001 inch
Operator: J. Fenamore Machine: #11 Gage: #8645 Zero Equals: zero
Date: 4/15/2008 4:53:41 PM

TIME	16:53	17:08	17:23	17:38	17:53	18:08	18:23	18:38	18:53	19:08	19:23	19:38	19:53	20:08	20:23	20:38	20:53
MEAN	29.7	30.6	31.5	30.3	31.1	28.6	28.8	29.4	28.9	31.0	29.0	28.1	32.8	30.2	29.5	30.3	32.5
RANGE	10.8	11.4	7.2	10.1	11.4	10.0	9.9	7.6	11.5	9.7	11.3	10.8	9.5	11.8	12.6	9.6	8.5
SUM	148.7	152.9	157.5	151.7	155.6	142.9	143.9	147.1	144.3	154.8	144.9	140.4	163.8	151.2	147.3	151.4	162.4
Cpk	0.2	0.2	0.3	0.3	0.3	0.3	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2
Cpm	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3
Ppk	0.2	0.2	0.3	0.3	0.3	0.3	0.3	0.3	0.2	0.3	0.2	0.2	0.2	0.2	0.2	0.2	0.3
ALARM	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -
NOTES	Y	Y	N	Y	N	N	N	N	N	N	N	Y	Y	N	N	N	N

[C#]
```
this.ChartTable.TableBackgroundMode =
    SPCGeneralizedTableDisplay.TABLE_SINGLE_COLOR_BACKGROUND_GRIDCELL;
this.ChartTable.BackgroundColor1 = Color.White;
this.ChartTable.BackgroundColor2 = Color.Gray;
```

[VB]
```
Me.ChartTable.TableBackgroundMode =
SPCGeneralizedTableDisplay.TABLE_SINGLE_COLOR_BACKGROUND_GRIDCELL
Me.ChartTable.BackgroundColor1 = Color.White
Me.ChartTable.BackgroundColor2 = Color.Gray
```

Table and Chart Fonts

There are a large number of fonts that you have control over, both the fonts in the table and the fonts in the chart. The programmer can select a default font (as in the case of non-US character set), or select individual fonts for different elements of the table and charts.

Table Fonts

The table fonts are accessed through the charts **ChartTable** property. Below is a list of accessible table fonts:

Property Name	**Description**
TimeLabelFont	The font used in the display of time values in the table.
SampleLabelFont	The font used in the display of sample numeric values in the table.
CalculatedLabelFont	The font used in the display of calculated values in the table.
StringLabelFont	The font used in the display of header string values in the table.
NotesLabelFont	The font used in the display of notes values in the table.

Extracted from the example BatchVariableControlCharts.BatchIndividualRangeChart

[C#]
```
this.ChartTable.SampleLabelFont = new Font("Times", 12, FontStyle.Regular);
```

[VB]

```
Me.ChartTable.SampleLabelFont = new Font("Times", 12, FontStyle.Regular)
```

The **ChartTable** class has a static property, **SPCGeneralizedTableDisplay.DefaultTableFont**, that sets the default Font. Use this if you want to establish a default font for all of the text in a table. This static property must be set BEFORE the charts **Init** routine.

Extracted from the example BatchVariableControlCharts.BatchIndividualRangeChart

[C#]

```
SPCGeneralizedTableDisplay.DefaultTableFont =
    new Font("Microsoft Sans Serif", 10, FontStyle.Regular);
// Initialize the SPCBatchVariableControlChart
this.InitSPCBatchVariableControlChart(charttype, numsamplespersubgroup,
       numdatapointsinview);
.
.
.
```

[VB]

```
SPCGeneralizedTableDisplay.DefaultTableFont = _
    new Font("Microsoft Sans Serif", 10, FontStyle.Regular)
' Initialize the SPCBatchVariableControlChart
Me.InitSPCBatchVariableControlChart(charttype, numsamplespersubgroup, _
    numdatapointsinview)
.
.
.
```

Chart Fonts

There are default chart fonts that are static objects in the **SPCChartObjects** class. They establish the default fonts for related chart objects and if you change them they need to be set before the first charts Init.. call. Since these properties are static, any changes to them will apply to the program as a whole, not just the immediate class.

AxisLabelFont	The font used to label the x- and y- axes.
AxisTitleFont	The font used for the axes titles.
HeaderFont	The font used for the chart title.
SubheadFont	The font used for the chart subhead.
ToolTipFont	The tool tip font.
AnnotationFont	The annotation font.
ControlLimitLabelFont	The font used to label the control limits

Extracted from the example TimeVariableControlCharts.DynamicXBarRChart

[C#]

```
SPCChartObjects.AxisTitleFont = new Font("Times", 12, FontStyle.Regular);
SPCChartObjects.ControlLimitLabelFont = new Font("Times", 10, FontStyle.Regular);

this.InitSPCTimeVariableControlChart(charttype,
       numsamplespersubgroup, numdatapointsinview, timeincrementminutes);
```

[VB]

```
SPCChartObjects.AxisTitleFont = new Font("Times", 12, FontStyle.Regular)
SPCChartObjects.ControlLimitLabelFont = new Font("Times", 10, FontStyle.Regular)

Me.InitSPCTimeVariableControlChart(charttype,  _
       numsamplespersubgroup, numdatapointsinview, timeincrementminutes)
```

The chart class has a static property, **DefaultTableFont**, that sets the default Font string. Since the chart fonts all default to different sizes, the default font is defined using a string specifying the name of the font. This static property must be set BEFORE the charts **Init** routine.

Extracted from the example Extracted from the example
TimeVariableControlCharts.DynamicXBarRChart

[C#]

```
SPCTimeVariableControlChart.DefaultChartFontString = "Times";
this.InitSPCTimeVariableControlChart(charttype,
       numsamplespersubgroup, numdatapointsinview, timeincrementminutes);.
.
.
```

[VB]

```
SPCTimeVariableControlChart.DefaultChartFontString = "Times"
Me.InitSPCTimeVariableControlChart(charttype,  _
       numsamplespersubgroup, numdatapointsinview, timeincrementminutes).
```

Table and Chart Templates

All of the strings displayed in the table and charts use a template unique to the string type. Numeric strings use a **NumericLabel** template, time/date strings use a time **TimeLabel** template, and so on. These templates permit the programmer to customize the display of the strings. Listed below are the various templates.

SPCChartObjects (Accessed in the charts PrimaryChart and SecondaryChart properties)

Property	**Type**	**Description**
XValueTemplate	NumericLabel	The x-value template for the data tooltip.
YValueTemplate	NumericLabel	The y-value template for the data tooltip.
XTimeValueTemplate	TimeLabel	x-value template for the data tooltip.
TextTemplate	ChartText	The text template for the data tooltip.

SPCGeneralizedTableDisplay (Accessed in the charts ChartTable property)

Property	**Type**	**Description**
TimeItemTemplate	TimeLabel	The TimeLabel object used as a template for displaying time values in the table.
SampleItemTemplate	NumericLabel	The NumericLabel object used as a template for displaying the sample values in the table.
CalculatedItemTemplate	NumericLabel	The NumericLabel object used as a template for displaying calculated values in the table.
StringItemTemplate	StringLabel	The StringLabel object used as a template for displaying string values in the table.

NotesItemTemplate	NotesLabel	The NotesLabel object used as a template for displaying string values in the table.

The most common use for these templates is to set the color attributes of a class of objects, or decimal precision of a numeric string.

```
this.ChartTable.SampleItemTemplate.LineColor = Color.Red;
```

SPC Charts without a Table

If you don’t want any of the items we have designated table itmes, just call the **UseNoTable** method. That method removes all of the table items, and displays the primary and/or secondary charts with a simple title and optional histograms.

This initialization method initializes the most important values in the creation of a SPC chart.

[VB]
```
Overloads Public Sub UseNoTable ( _
   ByVal primarychart As Boolean, _
   ByVal secondarychart As Boolean, _
   ByVal histograms As Boolean, _
   ByVal title As String, _
)
```

[C#]
```
public void UseNoTable (
   bool primarychart,
   bool secondarychart,
   bool histograms,
   String title
);
```

Parameters

primarychart

Set to true to display primary chart.

secondarychart

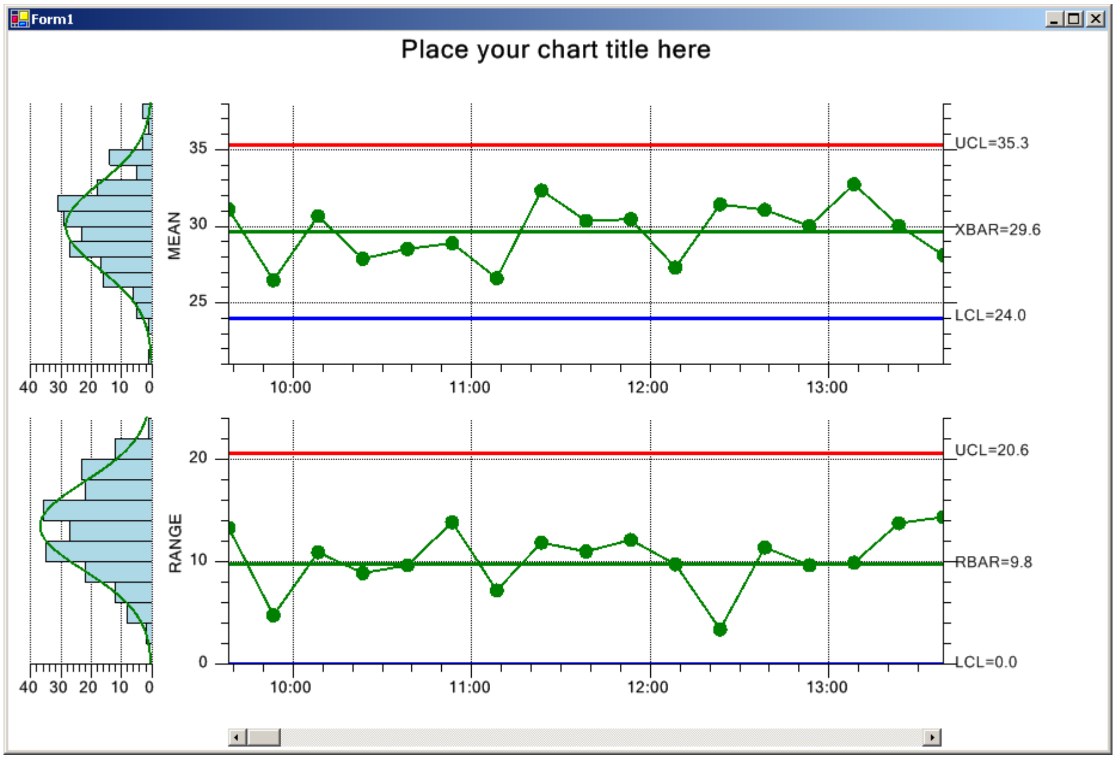
Set to true to display secondary chart.

histograms

Set to true to display chart histograms

title

Specifies the title for the charts



Extracted from the example program SPCApplication1.

[C#]

```
public void InitializeChart()
{
    //  SPC variable control chart type
    int charttype = SPCControlChartData.MEAN_RANGE_CHART;
// Number of datapoints in the view
    int numdatapointsinview = 17;


    // Initialize the SPCEventVariableControlChart
    this.InitSPCEventVariableControlChart(charttype,  numsamplespersubgroup,
                numdatapointsinview);

     this.UseNoTable(true,true,true, "Place your chart title here");

    this.EnableScrollBar = true;
    this.ChartAlarmEmphasisMode= SPCChartBase.ALARM_HIGHLIGHT_SYMBOL;

    // training data
    SimulateData(50, 30, 10);

// Calculate the SPC control limits for both graphs of the current SPC chart (X-Bar R)
    this.AutoCalculateControlLimits();

    // New data added after limits calculated
    SimulateData(150, 30, 15);

    // Scale the y-axis of the X-Bar chart to display all data and control limits
    this.AutoScalePrimaryChartYRange();
    // Scale the y-axis of the Range chart to display all data and control limits
    this.AutoScaleSecondaryChartYRange();
    // Rebuild the chart using the current data and settings
    this.RebuildChartUsingCurrentData();
}
```

[VB]

```
Public Sub InitializeChart()
    '  SPC variable control chart type
    Dim charttype As Integer = SPCControlChartData.MEAN_RANGE_CHART

    ' Number of datapoints in the view
    Dim numdatapointsinview As Integer = 17


     ' Initialize the SPCEventVariableControlChart
     Me.InitSPCEventVariableControlChart (charttype, numsamplespersubgroup,
numdatapointsinview, timeincrementminutes)

     Me.UseNoTable(True, True, True, "Place your chart title here")

     Me.EnableScrollBar = True

     Me.ChartAlarmEmphasisMode = SPCChartBase.ALARM_HIGHLIGHT_SYMBOL

     ' Training data to establish limits
     SimulateData(50, 30, 10)
     Me.AutoCalculateControlLimits()

     ' New data
     SimulateData(50, 30, 15)

     ' Scale the y-axis of the X-Bar chart to display all data and control limits
     Me.AutoScalePrimaryChartYRange()
     ' Scale the y-axis of the Range chart to display all data and control limits
     Me.AutoScaleSecondaryChartYRange()
     ' Rebuild the chart using the current data and settings
     Me.RebuildChartUsingCurrentData()
End Sub 'InitializeChart
```

Chart Position

If the SPC chart does not include frequency histograms on the left (they take up about 20% of the available chart width), you may want to adjust the left and right edges of the chart using the **GraphStartPosX** and **GraphStopPlotX** properties to allow for more room in the display of the data. This also affects the table layout, because the table columns must line up with the chart data points.

[C#]

```
this.GraphStartPosX = 0.1; // start here
this.GraphStopPosX = 0.875;  // end here
```

[VB]

```
Me.GraphStartPosX = 0.1 ' start here
Me.GraphStopPosX = 0.875 ' end here
```

There is not much flexibility positioning the top and bottom of the chart. Depending on the table items enabled, the table starts at the position defined the **TableStartPosY** property, and continues until all of the table items are displayed. It then offsets the top of the primary chart with respect to the bottom of the table by the value of the property **GraphTopTableOffset**. The top of the secondary chart offsets from the bottom of the primary chart by the amount of the property **InterGraphMargin**. The value of the property **GraphBottomPos** defines the bottom of the graph. The default values for these properties are:

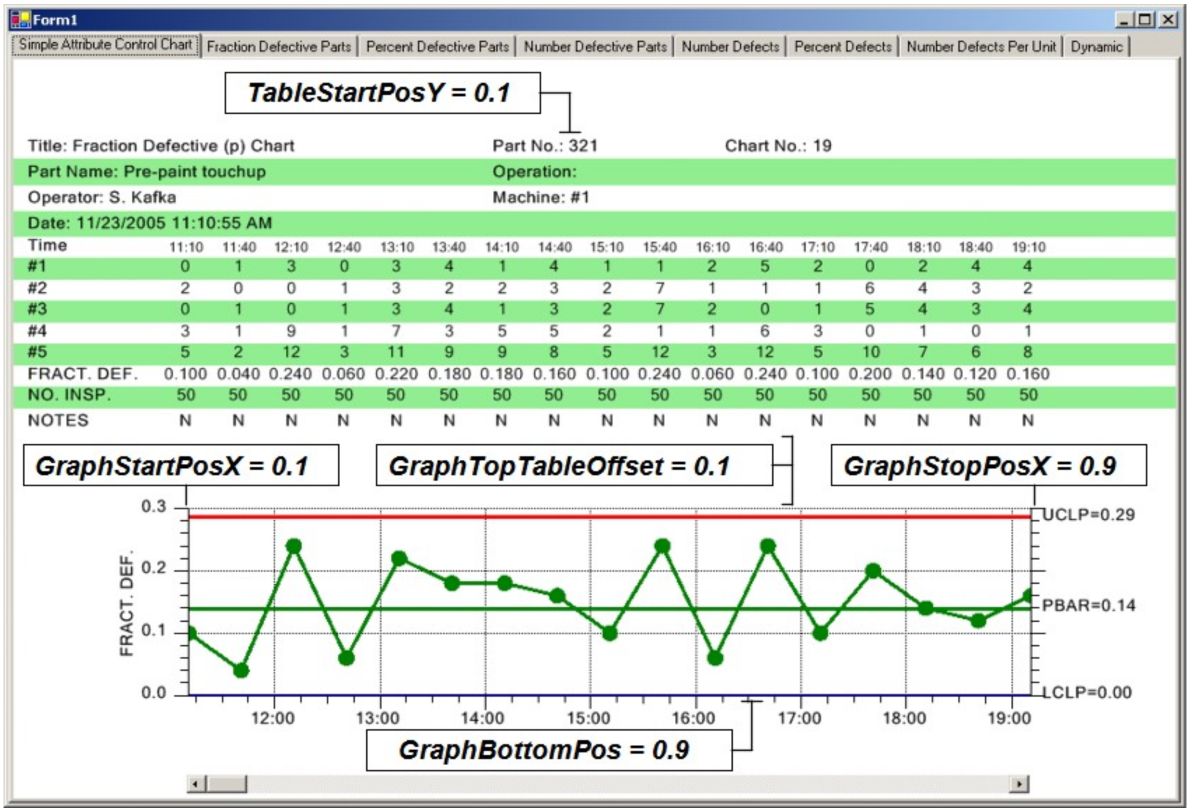
[C#]

```
this.TableStartPosY = 0.00;
this.GraphTopTableOffset = 0.02;
this.InterGraphMargin = 0.075;
this.GraphBottomPos = 0.925;
```

[VB]

```
Me.TableStartPosY = 0.00
Me.GraphTopTableOffset = 0.02
Me.InterGraphMargin = 0.075
Me.GraphBottomPos = 0.925
```

The picture below uses different values for these properties in order to emphasize the affect that these properties have on the resulting chart.



Title: Fraction Defective (p) Chart | Part No.: 321 | Chart No.: 19

Part Name: Pre-paint touchup | Operation:

Operator: S. Kafka | Machine: #1

Date: 11/23/2005 11:10:55 AM

Time	11:10	11:40	12:10	12:40	13:10	13:40	14:10	14:40	15:10	15:40	16:10	16:40	17:10	17:40	18:10	18:40	19:10
#1	0	1	3	0	3	4	1	4	1	1	2	5	2	0	2	4	4
#2	2	0	0	1	3	2	2	3	2	7	1	1	1	6	4	3	2
#3	0	1	0	1	3	4	1	3	2	7	2	0	1	5	4	3	4
#4	3	1	9	1	7	3	5	5	2	1	1	6	3	0	1	0	1
#5	5	2	12	3	11	9	9	8	5	12	3	12	5	10	7	6	8
FRACT. DEF.	0.100	0.040	0.240	0.060	0.220	0.180	0.180	0.160	0.100	0.240	0.060	0.240	0.100	0.200	0.140	0.120	0.160
NO. INSP.	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
NOTES	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N

SPC Control Limits

There are two ways to set the SPC control limit for a chart. The first explicitly sets the limits to values that you calculate on your own, because of some analysis that a quality engineer does on previously collected data. The second auto-calculates the limits using the algorithms supplied in this software.

The quick way to set the limit values and limit strings is to use the charts **ChartData.SetControlLimitValues** and **ChartData.SetControlLimitStrings** methods. This method only works for the default +-3-sigma level control limits, and not any others you may have added using the charts **AddAdditionalControlLimit** method discussed in the *Multiple Control Limits* section. The data values in the *controllimitvalues* and *controllimitstrings* arrays used to pass the control limit information must be sorted in the following order:

[SPC_PRIMARY_CONTROL_TARGET,
SPC_PRIMARY_LOWER_CONTROL_LIMIT,
SPC_PRIMARY_UPPER_CONTROL_LIMIT,
SPC_SECONDARY_CONTROL_TARGET,
SPC_SECONDARY_LOWER_CONTROL_LIMIT,
SPC_SECONDARY_UPPER_CONTROL_LIMIT]

Example code extracted from **the TimeVariableControlsCharts.MedianRangeChart** example program.

[C#]

```
double [] controllimitvalues = {42, 30, 53, 10, 0, 22};
this.ChartData.SetControlLimitValues(controllimitvalues);

string [] controllimitstrings = {"XBAR","LCL", "UCL","RBAR","LCL","UCL"};
this.ChartData.SetControlLimitStrings(controllimitstrings);
```

[VB]

```
Dim controllimitvalues() As Double = {30, 24, 36, 10, 0, 22}
Me.ChartData.SetControlLimitValues(controllimitvalues)

Dim controllimitstrings() As String = {"XBAR", "LCL", "UCL",
"RBAR", "LCL", "UCL"}
Me.ChartData.SetControlLimitStrings(controllimitstrings)
```

You can also set the control limit values and control limit text one value at a time using the **ChartData.SetControlLimitValue** and **ChartData.SetControlLimitString** methods.

A more complicated way to set the control limits explicitly is to first grab a reference to the **SPCControlLimitRecord** for a given control limit, and then change the value of that control limit, and the control limit text, if desired. The example below sets the control limit values and text for the three control limits (target value, upper control limit, and lower control limit) of the primary chart, and the three control limit values for the secondary chart.

[C#]

```
//target control limit primary chart
SPCControlLimitRecord primarytarget =
    ChartData.GetControlLimitRecord(SPCChartObjects.SPC_PRIMARY_CONTROL_TARGET);
primarytarget.ControlLimitValue = 30;
primarytarget.ControlLimitText = "XBAR";

//lower control limit primary chart
SPCControlLimitRecord primarylowercontrollimit =
ChartData.GetControlLimitRecord(SPCControlChartData.SPC_PRIMARY_LOWER_CONTROL_LIMIT);
primarylowercontrollimit.ControlLimitValue = 24;
primarylowercontrollimit.ControlLimitText = "LCL";
```

```
//upper control limit primary chart
SPCControlLimitRecord primaryuppercontrollimit =
ChartData.GetControlLimitRecord(SPCControlChartData.SPC_PRIMARY_UPPER_CONTROL_LIMIT);
primaryuppercontrollimit.ControlLimitValue = 36;
primaryuppercontrollimit.ControlLimitText = "UCL";

// Set control limits for secondary chart

//target control limit secondary chart
SPCControlLimitRecord secondarytarget =
ChartData.GetControlLimitRecord(SPCControlChartData.SPC_SECONDARY_CONTROL_TARGET);
secondarytarget.ControlLimitValue = 10;
secondarytarget.ControlLimitText = "RBAR";

//lower control limit secondary chart
SPCControlLimitRecord secondarylowercontrollimit =
ChartData.GetControlLimitRecord(SPCControlChartData.SPC_SECONDARY_LOWER_CONTROL_LIMIT);
secondarylowercontrollimit.ControlLimitValue = 0;
secondarylowercontrollimit.ControlLimitText = "LCL";

//upper control limit secondary chart
SPCControlLimitRecord secondaryuppercontrollimit =
ChartData.GetControlLimitRecord(SPCControlChartData.SPC_SECONDARY_UPPER_CONTROL_LIMIT);
secondaryuppercontrollimit.ControlLimitValue = 22;
secondaryuppercontrollimit.ControlLimitText = "UCL";
```

[VB]

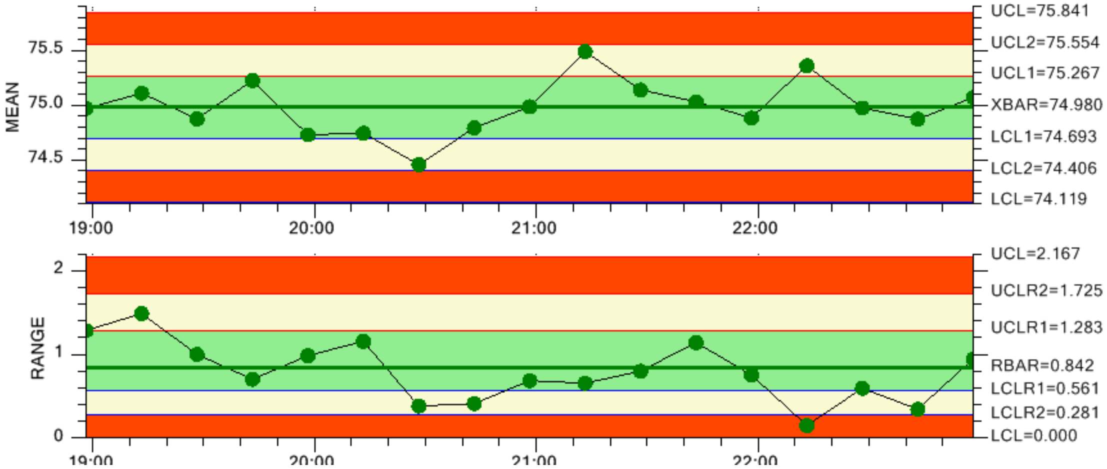
```
'target control limit primary chart
Dim primarytarget As SPCControlLimitRecord = _
  ChartData.GetControlLimitRecord(SPCControlChartData.SPC_PRIMARY_CONTROL_TARGET)
primarytarget.ControlLimitValue = 30
primarytarget.ControlLimitText = "XBAR"

'lower control limit primary chart
Dim primarylowercontrollimit As SPCControlLimitRecord =  _
  ChartData.GetControlLimitRecord(SPCControlChartData.SPC_PRIMARY_LOWER_CONTROL_LIMIT)
primarylowercontrollimit.ControlLimitValue = 24
primarylowercontrollimit.ControlLimitText = "LCL"

'upper control limit primary chart
Dim primaryuppercontrollimit As SPCControlLimitRecord = _
  ChartData.GetControlLimitRecord(SPCControlChartData.SPC_PRIMARY_UPPER_CONTROL_LIMIT)
primaryuppercontrollimit.ControlLimitValue = 36
primaryuppercontrollimit.ControlLimitText = "UCL"

' Set control limits for secondary chart
'target control limit secondary chart
Dim secondarytarget As SPCControlLimitRecord = _
  ChartData.GetControlLimitRecord(SPCControlChartData.SPC_SECONDARY_CONTROL_TARGET)
secondarytarget.ControlLimitValue = 10
secondarytarget.ControlLimitText = "RBAR"

‘lower control limit secondary chart
Dim secondarylowercontrollimit As SPCControlLimitRecord = _
ChartData.GetControlLimitRecord(SPCControlChartData.SPC_SECONDARY_LOWER_CONTROL_LIMIT)
secondarylowercontrollimit.ControlLimitValue = 0
secondarylowercontrollimit.ControlLimitText = "LCL"

'upper control limit secondary chart
Dim secondaryuppercontrollimit As SPCControlLimitRecord = _
ChartData.GetControlLimitRecord(SPCControlChartData.SPC_SECONDARY_UPPER_CONTROL_LIMIT)
secondaryuppercontrollimit.ControlLimitValue = 22
secondaryuppercontrollimit.ControlLimitText = "UCL"
```

We also added a method (Add3SigmaControlLimits) which will generate multiple control limits, for +-1, 2, and 3- sigma levels, based on an initial specification of the target value, and the +-3 sigma control limits. This is most useful if you want to generate +-1, 2 and 3-sigma control limits in order to fill in between them with a zone fill color. See the TimeVariableControlCharts.MultiLimitXBarRChart example. If you call the AutoCalculateControlLimits method, the initial +-1,2 and 3-sigma control limit values will be altered to the new, calculated values, but the control limit lines remain, with their new values. Since you do not normally want to be generating alarm messages for excursions into the +-1 and 2-sigma limit areas, the Add3SigmaControl limits has the option of disabling alarm notification in the case of +-1 and +-2 alarm conditions.

[C#]
```
double target = 75, lowlim = 74, highlim = 76;
bool limitcheck = false;
this.PrimaryChart.Add3SigmaControlLimits(target, lowlim, highlim, limitcheck);
this.PrimaryChart.ControlLimitLineFillMode = true;

target = 1; lowlim = 0; highlim = 2;
this.SecondaryChart.Add3SigmaControlLimits(target, lowlim, highlim, limitcheck);
this.SecondaryChart.ControlLimitLineFillMode = true;
```

[VB]

```
Dim target As Double = 75, lowlim As Double = 74, highlim As Double = 76
Dim limitcheck As Boolean = False
Me.PrimaryChart.Add3SigmaControlLimits(target, lowlim, highlim, limitcheck)
Me.PrimaryChart.ControlLimitLineFillMode = True

target = 1
lowlim = 0
highlim = 2
Me.SecondaryChart.Add3SigmaControlLimits(target, lowlim, highlim, limitcheck)
Me.SecondaryChart.ControlLimitLineFillMode = True
```



Control Limit Fill Option used with +-1, 2 and 3-sigma control limits

The second way to set the control limits is to call the **AutoCalculateControlLimits** method. You must have already added a collection of sampled data values to the charts **ChartData** SPC data object before you can call this method, since the method uses the internal **ChartData** object to provide the historical values needed in the calculation.

[C#]
```
// Must have data loaded before any of the Auto.. methods are called
```

```
SimulateData();

// Calculate the SPC control limits for both graphs of the current SPC
this.AutoCalculateControlLimits();
```

[VB]

```
' Must have data loaded before any of the Auto.. methods are called
SimulateData()

' Calculate the SPC control limits for both graphs of the current SPC
Me.AutoCalculateControlLimits()
```

You can add data to the **ChartData** object, auto-calculate the control limits to establish the SPC control limits, and continue to add new data values. Alternatively, you can set the SPC control limits explicitly, as the result of previous runs, using the previously described **ChartData.SetControlLimitValues** method, add new sampled data values to the **ChartData** object, and after a certain number of updates call the **AutoCalculateControlLimits** method to establish new control limits.

[C#]

```
updateCount++;
this.ChartData.AddNewSampleRecord(timestamp, samples);
if (updateCount > 50) // After 50 sample groups and calculate limits on the fly
{
// Calculate the SPC control limits for the X-Bar part of the current SPC chart
    this.AutoCalculateControlLimits();
    // Scale the y-axis of the X-Bar chart to display all data and control limits
    this.AutoScalePrimaryChartYRange();
    // Scale the y-axis of the Range chart to display all data and control limits
    this.AutoScaleSecondaryChartYRange();
}
```

[VB]

```
updateCount += 1
Me.ChartData.AddNewSampleRecord(timestamp, samples)
If updateCount > 50 Then ' After 50 sample groups and calculate limits on the fly
   ' Calculate the SPC control limits for the X-Bar part of the current SPC chart
   Me.AutoCalculateControlLimits()
   ' Scale the y-axis of the X-Bar chart to display all data and control limits
   Me.AutoScalePrimaryChartYRange()
   ' Scale the y-axis of the Range chart to display all data and control limits
   Me.AutoScaleSecondaryChartYRange()
End If
```

The **AutoCalculateControlLimits** method calculates the control limits for both the primary and secondary charts. If you want to auto-calculate the control limits for just one of the charts, call the **AutoCalculatePrimaryControlLimits** or **AutoCalculateSecondaryControlLimits** method.

Need to exclude records from the control limit calculation? Call the **ChartData.ExcludeRecordFromControlLimitCalculations** method, passing in true to exclude the record.

[C#]

```
for (int i=0; i < 10; i++)
      this.ChartData.ExcludeRecordFromControlLimitCalculations(i,true);
```

[VB]

```
Dim i As Integer
For i = 0 To 9
   Me.ChartData.ExcludeRecordFromControlLimitCalculations(i, True)
Next i
```

Formulas Used in Calculating Control Limits for Variable Control Charts

The SPC control limit formulas used by **AutoCalculateControlLimits** in the software derive from the following sources:

X-Bar R, X-Bar Sigma, EWMA, MA and CuSum - “Introduction to Statistical Quality Control” by Douglas C. Montgomery, John Wiley and Sons, Inc. 2001.

Median-Range, Individual-Range - “SPC Simplified – Practical Steps to Quality” by Robert T. Amsden, Productivity Inc., 1998.

SPC Control Chart Nomenclature

UCL = Upper Control Limit

LCL = Lower Control Limit

Center line = The target value for the process

$\overline{\overline{X}}$ = X double-bar - Mean of sample subgroup means (also called the grand average)

$\overline{R}$ = R-bar – Mean of sample subgroup ranges

$\tilde{R}$ = R-Median – Median of sample subgroup ranges

S = Sigma – sample standard deviation

$\overline{S}$ = Sigma-bar – Average of sample subgroup sigma’s

M = sample Median

$\tilde{M}$ = Median of sample subgroup medians

X-Bar R Chart – Also known as the Mean (or Average) and Range Chart

Control Limits for the X-Bar Chart

UCL = $\bar{\bar{X}} + A_2 * \bar{R}$

Center line = $\bar{\bar{X}}$

LCL = $\bar{\bar{X}} - A_2 * \bar{R}$

Control Limits for the R-Chart

UCL = $\bar{R} + D_4 * \bar{R}$

Center line = $\bar{R}$

LCL = $\bar{R} - D_3 * \bar{R}$

Where the constants A_2, D_3 and D_4 are tabulated in every SPC textbook for various sample sizes.

X-Bar Sigma – Also known as the X-Bar S Chart

Control Limits for the X-Bar Chart

UCL = $\bar{\bar{X}} + A_3 * \bar{S}$

Center line = $\bar{\bar{X}}$

LCL = $\bar{\bar{X}} - A_3 * \bar{S}$

Control Limits for the Sigma-Chart

UCL = $\bar{B}_4 * \bar{S}$

Center line = $\bar{S}$

$$\text{LCL} = \bar{B}_3 * \bar{S}$$

Where the constants A_3, B_3 and B_4 are tabulated in every SPC textbook for various sample sizes.

Median Range – Also known as the Median and Range Chart

Control Limits for the Median Chart

$$\text{UCL} = \tilde{M} + \tilde{A}_2 * \tilde{R}$$

$$\text{Center line} = \tilde{M}$$

$$\text{LCL} = \tilde{M} - \tilde{A}_2 * \tilde{R}$$

Control Limits for the R-Chart

$$\text{UCL} = \tilde{R} + \tilde{D}_4 * \tilde{R}$$

$$\text{Center line} = \tilde{R}$$

$$\text{LCL} = \tilde{R} - \tilde{D}_3 * \tilde{R}$$

The constants A_2, D_3 and D_4 for median-range charts are different than those for mean-range charts. A brief tabulation of the median-range chart specific values appears below

Size	**A2**	**D3**	**D4**
2	2.22	0.0	3.87
3	1.26	0.0	2.75
4	0.83	0.0	2.38
5	0.71	0.0	2.18

Individual Range Chart – Also known as the X-R Chart

Control Limits for the X-Bar Chart

$$\text{UCL} = \bar{X} + E_2 * \bar{R}$$

=

Center line = X

LCL = $\bar{X} - E_2 * \bar{R}$

Control Limits for the R-Chart

UCL = $\bar{R} + D_4 * \bar{R}$

Center line = $\bar{R}$

LCL = 0

$\bar{R}$ in this case is the average of the moving ranges.

$\bar{X}$ in this case is the mean of the samples

Where the constants E_2 and D_4 are tabulated in every SPC textbook for various sample sizes.

Levey-Jennings Chart

Control Limits

+Sigma 1 = Mean + SD
+Sigma 2 = Mean + 2 * SD
+Sigma 3 = Mean + 3 *SD

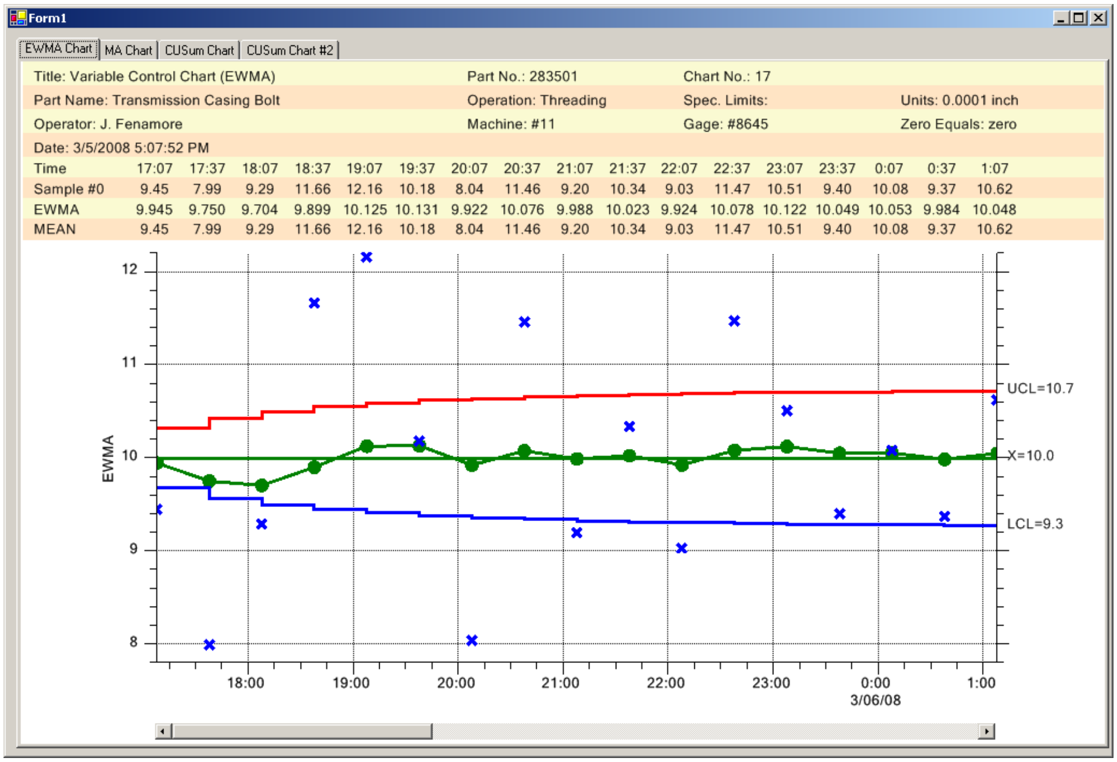
Center line = Mean

-Sigma 1 = Mean - SD
-Sigma 2 = Mean - 2 * SD
-Sigma 3 = Mean - 3 *SD

SD = the standard deviation of the sample population

Mean = the mean of the sample population

EWMA Chart – Exponentially Weighted Moving Average



Title: Variable Control Chart (EWMA) Part No.: 283501 Chart No.: 17

Part Name: Transmission Casing Bolt Operation: Threading Spec. Limits: Units: 0.0001 inch

Operator: J. Fenamore Machine: #11 Gage: #8645 Zero Equals: zero

Date: 3/5/2008 5:07:52 PM

Time	17:07	17:37	18:07	18:37	19:07	19:37	20:07	20:37	21:07	21:37	22:07	22:37	23:07	23:37	0:07	0:37	1:07
Sample #0	9.45	7.99	9.29	11.66	12.16	10.18	8.04	11.46	9.20	10.34	9.03	11.47	10.51	9.40	10.08	9.37	10.62
EWMA	9.945	9.750	9.704	9.899	10.125	10.131	9.922	10.076	9.988	10.023	9.924	10.078	10.122	10.049	10.053	9.984	10.048
MEAN	9.45	7.99	9.29	11.66	12.16	10.18	8.04	11.46	9.20	10.34	9.03	11.47	10.51	9.40	10.08	9.37	10.62

A EWMA chart showing the variable control limits, actual values and EWMA values

The current value (z) for an EWMA chart is calculated as an exponentially weighted moving average of all previous samples.

$$z_i = \lambda * x_i + (1 - \lambda)z_{i-1}$$

where x_i is the sample value for time interval i, the smoothing value λ has the permissible range of $0 < \lambda <= 1$ and the starting value (required with the first sample at i = 0) is the process target value, μ_0 .

Control Limits for the EWMA Chart

$$\text{UCL} = \mu_0 + L * \sigma * \text{Sqrt}(((\lambda /(2- \lambda)) * (1- (1- \lambda)^{2i}))$$

$$\text{Center line} = \mu_0$$

$$\text{LCL} = \mu_0 - L * \sigma * \text{Sqrt}(((\lambda /(2- \lambda)) * (1- (1- \lambda)^{2i}))$$

μ_0 is the process mean

σ is the process standard deviation, also known as sigma

λ is the user specified smoothing value. A typical value for λ is 0.05, 0.1 or 0.2

L is the width of the control limits. The typical value for L is in the range of 2.7 to 3.0 (corresponding to the usual three-sigma control limits).

The software does not calculate optimal λ and L values; that is up to you, the programmer to supply, based on your experience with EWMA charts.

Note that the term $(1- (1- \lambda)^{2i})$ approaches unity as i increases. The implies that the control limits of an EWMA chart will reach approximate steady state values defined by:

$$\text{UCL} \quad = \mu_0 + L \; * \sigma * \; \text{Sqrt}(\lambda /(2-\lambda))$$

$$\text{LCL} \quad = \mu_0 - L \; * \sigma * \; \text{Sqrt}(\lambda /(2-\lambda))$$

It is best if you use the exact equations that take into account the sample period, so that an out of control process can be detected using the tighter control limits that are calculated for small i.

If the EWMA chart is used with subgroup sample sizes greater than 1, the value of x_i is replace by the mean of the corresponding sample subgroup, and the value of σ is replaced by the value σ/sqrt(n), where in is the sample subgroup size.

You specify λ and L immediately after the initialization call **InitSPCTimeVaraibleControlChart, InitSPCBatchVariableControlChart,** or **InitSPCEventVaraibleControlChart** . See the examples MiscTimeBasedControlCharts.EWMAChart, and MiscBatchBasedControlCharts.EWMAChart. Specify L using the **DefaultControlLimitSigma** property, and λ using the **EWMA_Lambda** property. You can optionally set the EWMA starting value (**EWMA**_StartingValue), normally set to the process mean value, and whether or not to use the steady-state EWMA control limits (**UseSSLimits**).

Extracted from the MiscTimeBasedControlCharts.EWMAChart example.

[C#]

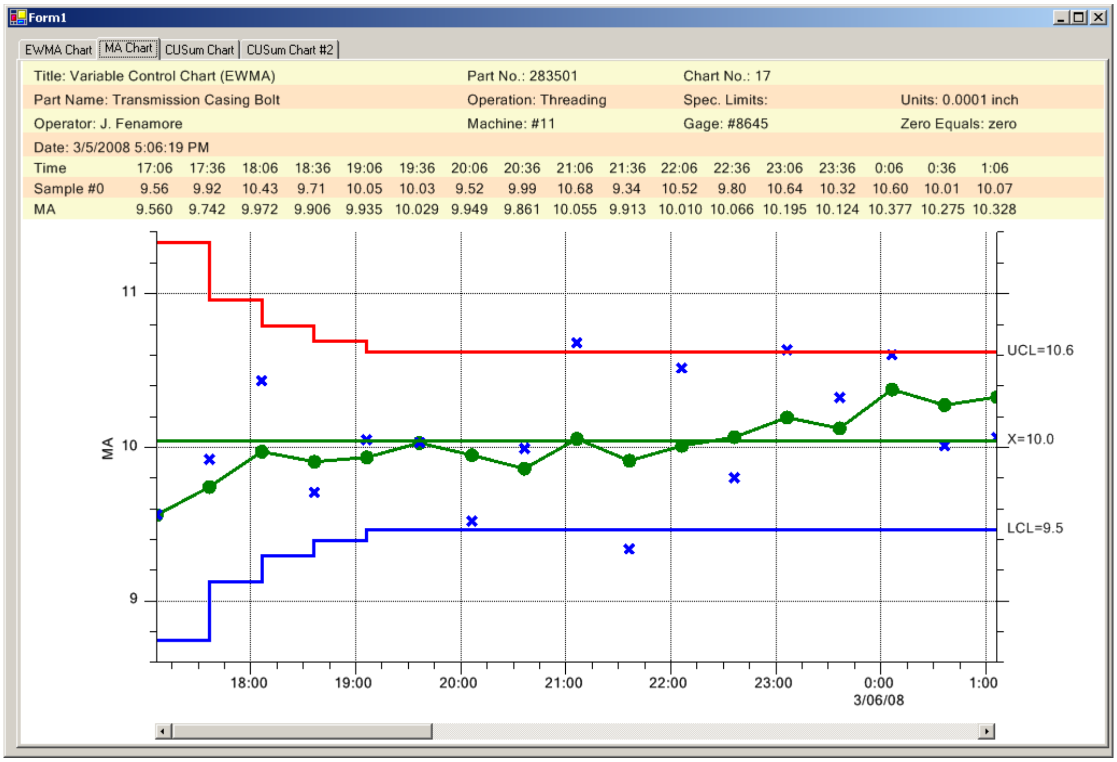
```
//  SPC variable control chart type
int charttype = SPCControlChartData.EWMA_CHART;
.
.
.
```

```
// Initialize the SPCTimeVariableControlChart
this.InitSPCTimeVariableControlChart(charttype,
    numsamplespersubgroup, numdatapointsinview, sampleincrement);

this.ChartData.EWMA_StartingValue = 10; // Set to estimate of mean of process variable.
this.ChartData.EWMA_Lambda = 0.1;
this.DefaultControlLimitSigma = 2.7; // Specifies L value
this.ChartData.EWMA_UseSSLimits = false;
```

[VB]

```
'  SPC variable control chart type
Private charttype As Integer = SPCControlChartData.EWMA_CHART
.
.
.
' Initialize the SPCTimeVariableControlChart
Me.InitSPCTimeVariableControlChart(charttype, numsamplespersubgroup, _
   numdatapointsinview, timeincrementminutes)
Me.ChartData.EWMA_StartingValue = 10 ' Set to estimate of mean of process variable.
Me.ChartData.EWMA_Lambda = 0.1
Me.DefaultControlLimitSigma = 2.7
Me.ChartData.EWMA_UseSSLimits = False
```

MA Chart – Moving Average



Title: Variable Control Chart (EWMA) Part No.: 283501 Chart No.: 17

Part Name: Transmission Casing Bolt Operation: Threading Spec. Limits: Units: 0.0001 inch

Operator: J. Fenamore Machine: #11 Gage: #8645 Zero Equals: zero

Date: 3/5/2008 5:06:19 PM

Time	17:06	17:36	18:06	18:36	19:06	19:36	20:06	20:36	21:06	21:36	22:06	22:36	23:06	23:36	0:06	0:36	1:06
Sample #0	9.56	9.92	10.43	9.71	10.05	10.03	9.52	9.99	10.68	9.34	10.52	9.80	10.64	10.32	10.60	10.01	10.07
MA	9.560	9.742	9.972	9.906	9.935	10.029	9.949	9.861	10.055	9.913	10.010	10.066	10.195	10.124	10.377	10.275	10.328

A MA chart showing the variable control limits, actual values and moving average values

The current value (z) for a MA chart is calculated as a weighted moving average of the N most recent samples.

$$z_i = (x_i + x_{i-1} + x_{i-2} + \ldots \; x_{i-N+1})/N$$

where x_i is the sample value for time interval i, and N is the length of the moving average.

Control Limits for the MA Chart

UCL $= \mu_0 + 3 * \sigma / \text{sqrt}(N)$

Center line $= \mu_0$

LCL $= \mu_0 - 3 * \sigma / \text{sqrt}(N)$

μ_0 is the process mean

σ is the process standard deviation, also known as sigma

N is the length of the moving average used to calculate the current chart value

Control Limits for the MR part of the MAMR (Moving Average/Moving Range Chart

Control Limits for the R-Chart

UCL $= \overline{R} + D_4 * \overline{R}$

Center line $= \overline{R}$

LCL $= 0$

$\overline{R}$ in this case is the average of the moving ranges.

Where the constant D_4 is tabulated in every SPC textbook for various sample sizes.

Control Limits for the MS part of the MAMS (Moving Average/Moving Sigma Chart

$$\text{UCL} = \overline{B}_4 * \overline{S}$$

$$\text{Center line} = \overline{S}$$

$$\text{LCL} = \overline{B}_3 * \overline{S}$$

$\overline{S}$ in this case is the average of the moving sigmas.

Where the constant B_4 is tabulated in every SPC textbook for various sample sizes.

The software does not calculate an optimal N value; that is up to you, the programmer to supply, based on your past experience with MA charts.

For the values of z_i where i < N-1, the weighted average and control limits are calculated using the actual number of samples used in the average, rather than N. This results in expanded values for the control limits for small i < N-1.

You specify N, the length of the moving average, immediately after the initialization call **InitSPCTimeVaraibleControlChart, InitSPCBatchVariableControlChart,** or **InitSPCEventVaraibleControlChart**. Set the process mean and process sigma used in the control limit calculations using the **ProcessMean** and **ProcessSigma** properties.
See the examples MiscTimeBasedControlCharts.MAChart, and MiscBatchBasedControlCharts.MAChart. Specify N using the **MA_w** property.

Extracted from the MiscTimeBasedControlCharts.MAChart example.

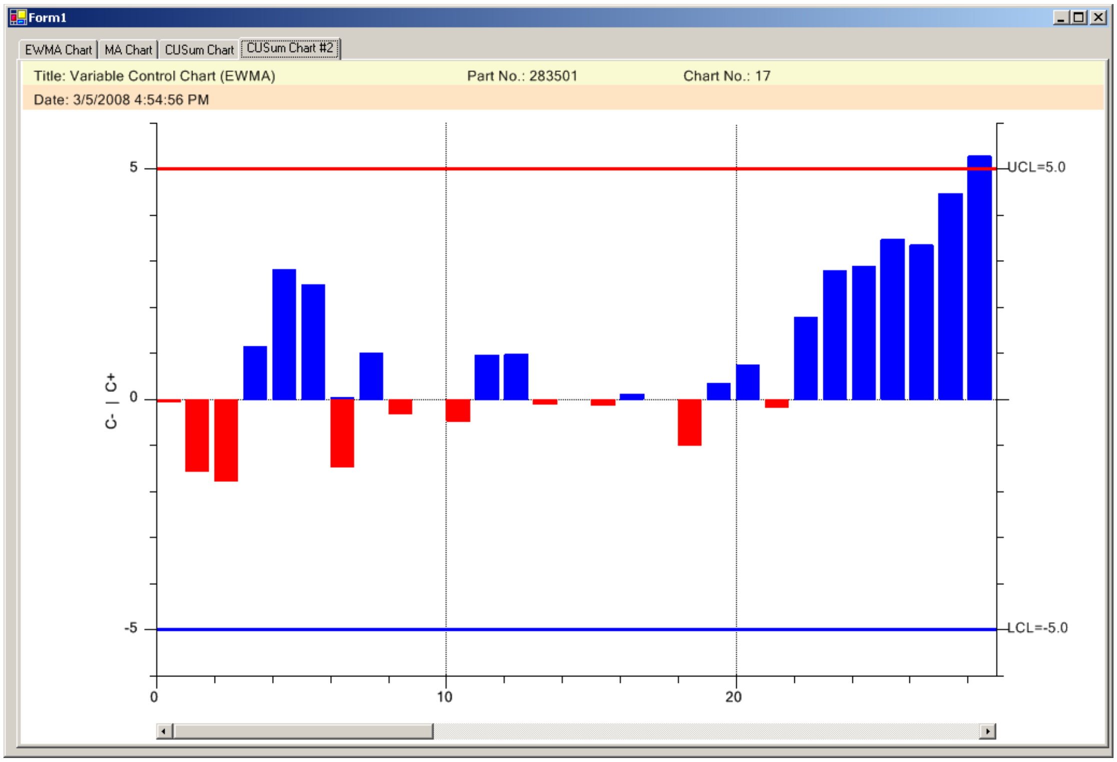
[C#]

```
//  SPC variable control chart type
int charttype = SPCControlChartData.MA_CHART;
.
.
.
// Initialize the SPCTimeVariableControlChart
this.InitSPCTimeVariableControlChart(charttype,
numsamplespersubgroup, numdatapointsinview, sampleincrement);
// Number of time periods in moving average
this.ChartData.MA_w = 9;
```

[VB]

```
'  SPC variable control chart type
Private charttype As Integer = SPCControlChartData.MA_CHART
.
.
.
' Initialize the SPCTimeVariableControlChart
Me.InitSPCTimeVariableControlChart(charttype, numsamplespersubgroup, _
   numdatapointsinview, timeincrementminutes)
' Number of time periods in moving average
Me.ChartData.MA_w = 7
Me.ChartData.ProcessMean = 10.0
Me.ChartData.ProcessSigma = 1.0
```

CuSum Chart – Tabular, one-sided, upper and lower cumulative sum



Title: Variable Control Chart (EWMA) Part No.: 283501 Chart No.: 17

Date: 3/5/2008 4:54:56 PM

A batch CuSum chart exceeding the H value

The tabular cusum works by accumulating deviations from the process mean, μ_0. Positive deviations are accumulated in the one sided upper cusum statistic, C^+, and negative deviations are accumulated in the one sided lower cusum statistic, C^-. The statistics are calculated using the following equations:

$$C^+_i = \max[0, x_i - (\mu_0 + K) + C^+_{i-1}]$$

$$C^-_i = \max[0, (\mu_0 - K) - x_i + C^+_{i-1}]$$

where the starting values are $C^+_0 = C^-_0 = 0$

μ_0 is the process mean

K is the reference (or slack value) that is usually selected to be one-half the magnitude of the difference between the target value, μ_0 , and the out of control process mean value, μ_1, that you are trying to detect.

K = ABS($\mu_{1} - \mu_{0}$)/2

The control limits used by the chart are +-H. If the value of either C^{+} or C^{-} exceed +- H, the process is considered out of control.

The software does not calculate an optimal H or K value; that is up to you, the programmer to supply, based on your past experience with CuSum charts. Typically H is set equal to 5 times the process standard deviation, σ. Typically K is selected to be one-half the magnitude of the difference between the target value, μ_0 , and the out of control process mean value, $\mu_{1,}$ that you are trying to detect. You specify H and K in the initialization call **InitSPCTimeCusumControlChart**, **InitSPCBatchCusumControlChart, or InitSPCEventCusumControlChar**. See the examples MiscTimeBasedControlCharts.CUSumChart, MiscTimeBasedControlCharts.CUSumChart2, MiscBatchBasedControlCharts.CUSumChart, and MiscBatchBasedControlCharts.CUSumChart2.

Extracted from MiscTimeBasedControlCharts.CUSumChart

[C#]

```
int charttype = SPCControlChartData.TABCUSUM_CHART;
double processMean = 10;
double kValue = 0.5;
double hValue = 5;
.
.
// Initialize the SPCTimeVariableControlChart
this.InitSPCTimeCusumControlChart(charttype,
    numsamplespersubgroup, numdatapointsinview, sampleincrement,
    processMean, kValue, hValue);
```

[VB]

```
Private charttype As Integer = SPCControlChartData.TABCUSUM_CHART
Private processMean As Double = 10
Private kValue As Double = 0.5
Private hValue As Double = 5
.
.
' Initialize the SPCTimeVariableControlChart
Me.InitSPCTimeCusumControlChart(charttype, numsamplespersubgroup, _
    numdatapointsinview, timeincrementminutes, processMean, kValue, hValue)
```

Or, you can call the **InitSPCTimeCusumControlChart** method and specify H and K using immediately afterwards using simple property calls.

Extracted from MiscTimeBasedControlCharts.CUSumChart2

[C#]

```
int charttype = SPCControlChartData.TABCUSUM_CHART;
double processMean = 10;
double kValue = 0.5;
double hValue = 5;
.
.
.
```

```
// Initialize the SPCTimeVariableControlChart
this.InitSPCTimeVariableControlChart(charttype,
    numsamplespersubgroup, numdatapointsinview, sampleincrement);
this.ChartData.CusumHValue = hValue;
this.ChartData.CusumKValue = kValue;
this.ChartData.ProcessMean = processMean;
```

[VB]

```
Private charttype As Integer = SPCControlChartData.TABCUSUM_CHART
Private processMean As Double = 10
Private kValue As Double = 0.5
Private hValue As Double = 5
.
.
.
' Initialize the SPCTimeVariableControlChart
Me.InitSPCTimeVariableControlChart(charttype, numsamplespersubgroup, _
    numdatapointsinview, timeincrementminutes)
Me.ChartData.CusumHValue = hValue
Me.ChartData.CusumKValue = kValue
Me.ChartData.ProcessMean = processMean
```

Variable SPC Control Limits

There can be situations where SPC control limits change in a chart. If the control limits change, you need to set the following **ControlLineMode** property to SPCChartObjects.CONTROL_LINE_VARIABLE, as in the example below. The default value is SPCChartObjects.CONTROL_LINE_FIXED.

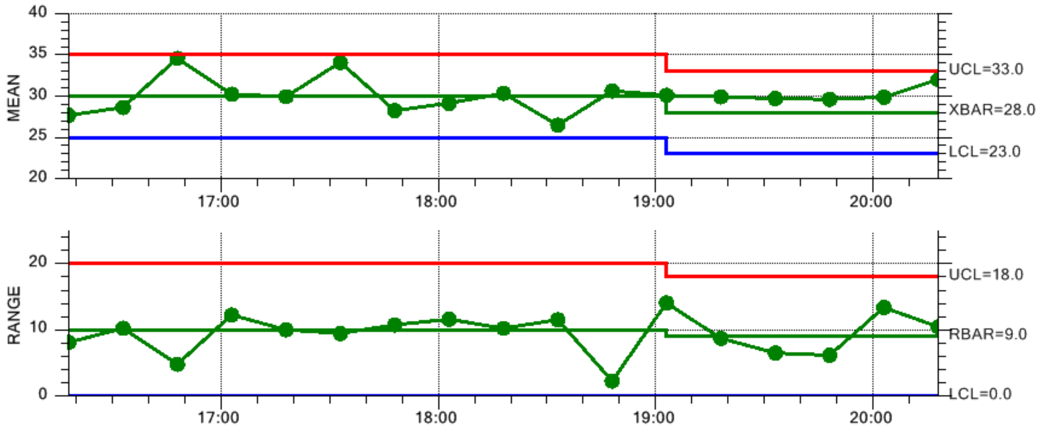
[C#]

```
this.PrimaryChart.ControlLineMode = SPCChartObjects.CONTROL_LINE_VARIABLE;
this.SecondaryChart.ControlLineMode = SPCChartObjects.CONTROL_LINE_VARIABLE;
```

[VB]

```
Me.PrimaryChart.ControlLineMode = SPCChartObjects.CONTROL_LINE_VARIABLE
Me.SecondaryChart.ControlLineMode = SPCChartObjects.CONTROL_LINE_VARIABLE
```

In the SPCChartObjects.CONTROL_LINE_FIXED case, the current SPC control limit plots as a horizontal straight line for the entire width of the chart, regardless if the control limit changes, either explicitly, or using the **AutoCalculateControlLimits** method. If the **ControlLineMode** property is SPCChartObjects.CONTROL_LINE_VARIABLE, the SPC limit value plots at the value it had when the sample subgroup values updated. If you change a control limit value, the control limit line will no longer be a straight horizontal line, instead it will be jagged, or stepped, depending on the changes made.



There are three ways to enter new SPC limit values. See the example program TimeVaraibleControlCharts.VariableControlLimits for an example of all three methods. First, you can use the method **ChartData.SetControlLimitValues** method.

[C#]

```
double [] initialControlLimits = {30, 25, 35, 10, 0, 20};
double [] changeControlLimits = {28, 23, 33, 9, 0, 18};
.
.
this.PrimaryChart.ControlLineMode = SPCChartObjects.CONTROL_LINE_VARIABLE;
.
.
// Change limits at sample subgroup 10
if (i== 10)
{
    this.ChartData.SetControlLimitValues(changeControlLimits);
}
this.ChartData.AddNewSampleRecord(timestamp, samples, notesstring);
```

[VB]

```
Dim initialControlLimits() As Double = {30, 25, 35, 10, 0, 20}
Dim changeControlLimits() As Double = {28, 23, 33, 9, 0, 18}
.
.
Me.PrimaryChart.ControlLineMode = SPCChartObjects.CONTROL_LINE_VARIABLE
.
.
' Change limits at sample subgroup 10
If i = 10 Then
   Me.ChartData.SetControlLimitValues(changeControlLimits)
End If
Me.ChartData.AddNewSampleRecord(timestamp, samples, notesstring)
```

Second, you can use the **AutoCalculateControlLimits** method. You must have already added a collection of sampled data values to the charts **ChartData** SPC data object before you can call this method, since the method uses the internal **ChartData** object to provide the historical values needed in the calculation.

[C#]

```
.
.
this.PrimaryChart.ControlLineMode = SPCChartObjects.CONTROL_LINE_VARIABLE;
.
.

//  Variable Control Limits re-calculated every update after 10 using
//  AutoCalculateControlLimits
    if (i > 10)
        this.AutoCalculateControlLimits();
    this.ChartData.AddNewSampleRecord(timestamp, samples, notesstring);
```

[VB]

```
.
.
Me.PrimaryChart.ControlLineMode = SPCChartObjects.CONTROL_LINE_VARIABLE
.
.

'  Variable Control Limits re-calculated every update after 10 using
`  AutoCalculateControlLimits
If i > 10 Then
  Me.AutoCalculateControlLimits()
End If
Me.ChartData.AddNewSampleRecord(timestamp, samples, notesstring)
```

Last, you can enter the SPC control limits with every new sample subgroup record, using one of the methods that include a control limits array parameter.

[C#]

```
double [] initialControlLimits = {30, 25, 35, 10, 0, 20};
double [] changeControlLimits = {28, 23, 33, 9, 0, 18};
DoubleArray variableControlLimits;
.
.
this.PrimaryChart.ControlLineMode = SPCChartObjects.CONTROL_LINE_VARIABLE;
.
.
//      Variable Control Limits updated using AddNewSampleRecord
    if (i== 10) // need to convert changeControlLimits to a DoubleArray
        variableControlLimits = new DoubleArray(changeControlLimits);
    this.ChartData.AddNewSampleRecord(timestamp, samples,
        variableControlLimits, notesstring);
```

[VB]

```
Dim initialControlLimits() As Double = {30, 25, 35, 10, 0, 20}
Dim changeControlLimits() As Double = {28, 23, 33, 9, 0, 18}
Dim variableControlLimits As DoubleArray
.
.
Me.PrimaryChart.ControlLineMode = SPCChartObjects.CONTROL_LINE_VARIABLE
.
.
`  Variable Control Limits updated using AddNewSampleRecord
 If i = 10 Then ' need to convert changeControlLimits to a DoubleArray
   variableControlLimits = New DoubleArray(changeControlLimits)
 End If
 Me.ChartData.AddNewSampleRecord(timestamp, samples, variableControlLimits, _
        notesstring)
```

SetSpecLimits

```
public void SetSpecLimits(double lowspeclimit, double highspeclimit)
```

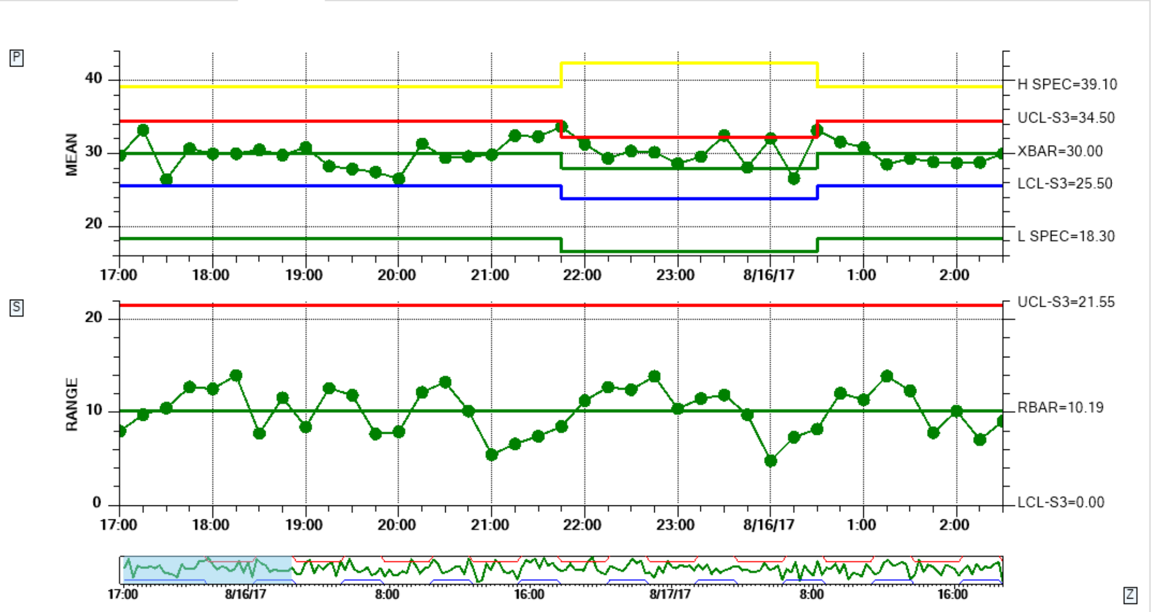
Set high and low specification limits

lowspeclimit	The low spec limit.
highspeclimit	The high spec limit.

The example below updates the control limits and the spec limits.

// Create the spec limits in the chart setup

```
this.PrimaryChart.AddSpecLimit(SPCChartObjects.SPC_LOWER_SPEC_LIMIT, initialLSL, "L
                          SPEC", new ChartAttribute(Colors.Green, 3.0));
this.PrimaryChart.AddSpecLimit(SPCChartObjects.SPC_UPPER_SPEC_LIMIT, initialHSL, "H
SPEC", new ChartAttribute(Colors.Yellow, 3.0));
.
.
.

this.ChartData.UpdateControlLimitsUsingMeanAndSigma(SPCChartObjects.PRIMARY_CHART,
                          originalMeanP, originalSigmaP);
this.ChartData.SetSpecLimits(initialLSL, initialHSL);
this.ChartData.AddNewSampleRecord(timestamp, samples, notesstring);
```



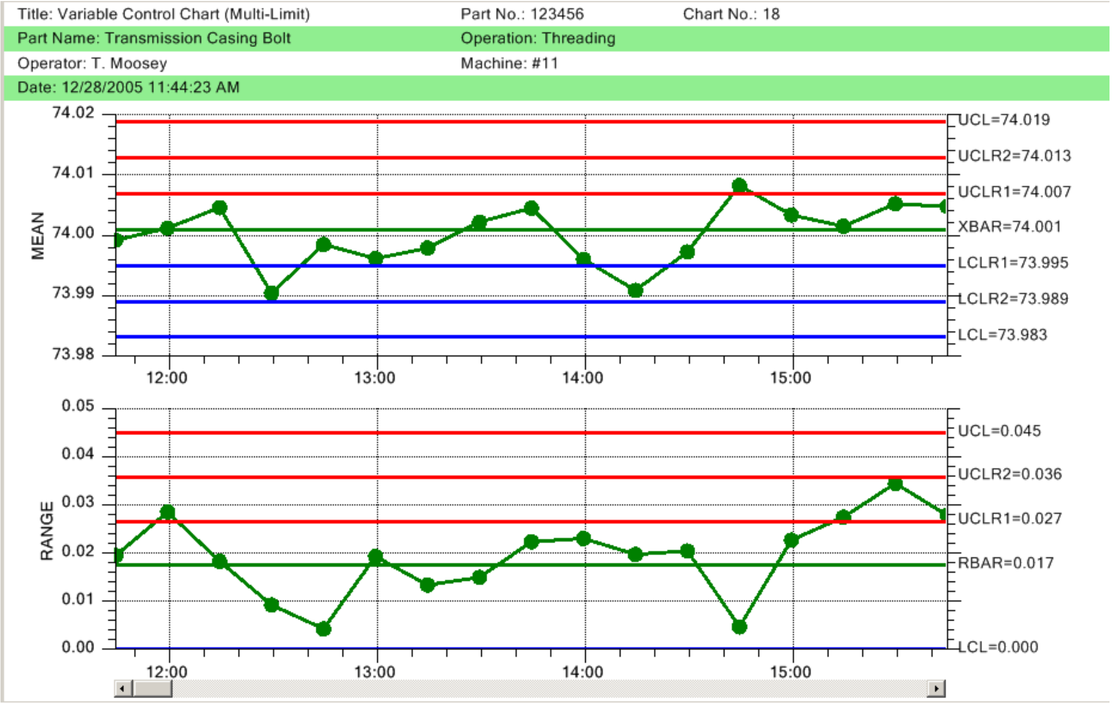
In this example, the control limits and the specification limits are changed in mid course.

For a complete example, see the demo NewRev3Features.VariableControlAndSpecLimits.

Multiple SPC Control Limits

The normal SPC control limit displays at the 3-sigma level, both high and low. A common standard is that if the process variable under observation falls outside of the +-3-sigma limits the process is out of control. The default setup of our variable control charts have a high limit at the +3-sigma level, a low limit at the -3-sigma level, and a target value. There are situations where the quality engineer also wants to display control limits at the 1-sigma and 2-sigma level. The operator might receive some sort of preliminary warning if the process variable exceeds a 2-sigma limit.

You are able to add additional control limit lines to a variable control chart, as in the example program TimeVariableControlCharts.MultiLimitXBarRChart.



Title: Variable Control Chart (Multi-Limit) Part No.: 123456 Chart No.: 18
Part Name: Transmission Casing Bolt Operation: Threading
Operator: T. Moosey Machine: #11
Date: 12/28/2005 11:44:23 AM

We also added a method (Add3SigmaControlLimits) which will generate multiple control limits, for +-1, 2, and 3- sigma levels, based on an initial specification of the target value, and the +-3 sigma control limits. This is most useful if you want to generate +-1, 2 and 3-sigma control limits in order to fill in between them with a zone fill color. See the TimeVariableControlCharts.MultiLimitXBarRChart example. If you call the AutoCalculateControlLimits method, the initial +-1,2 and 3-sigma control limit values will be altered to the new, calculated values, but the control limit lines remain, with their new values. Since you do not normally want to be generating alarm messages for excursions into the +-1 and

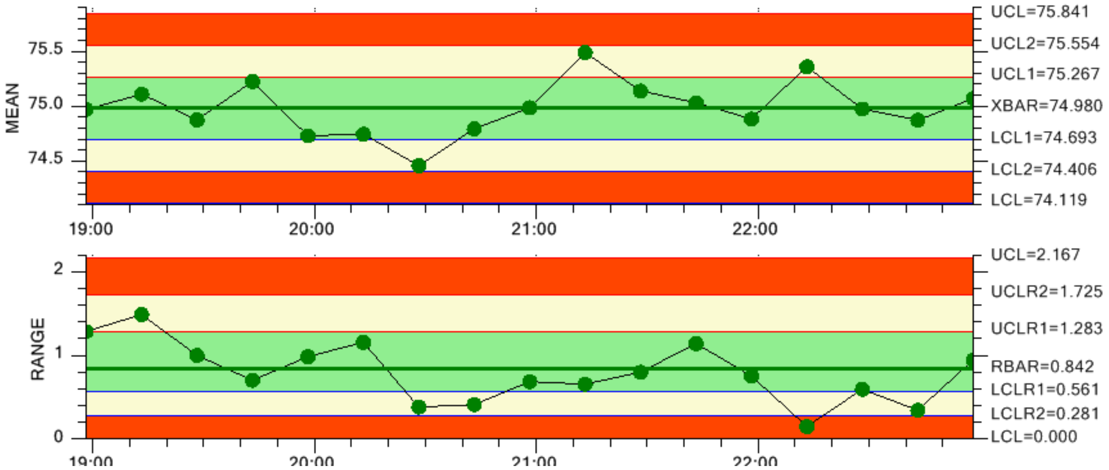
2-sigma limit areas, the Add3SigmaControl limits has the option of disabling alarm notification in the case of +-1 and +-2 alarm conditions.

[C#]

```
double target = 75, lowlim = 74, highlim = 76;
bool limitcheck = false;
this.PrimaryChart.Add3SigmaControlLimits(target, lowlim, highlim, limitcheck);
this.PrimaryChart.ControlLimitLineFillMode = true;

target = 1; lowlim = 0; highlim = 2;
this.SecondaryChart.Add3SigmaControlLimits(target, lowlim, highlim, limitcheck);
this.SecondaryChart.ControlLimitLineFillMode = true;
```

[VB]

```
Dim target As Double = 75, lowlim As Double = 74, highlim As Double = 76
Dim limitcheck As Boolean = False
Me.PrimaryChart.Add3SigmaControlLimits(target, lowlim, highlim, limitcheck)
Me.PrimaryChart.ControlLimitLineFillMode = True

target = 1
lowlim = 0
highlim = 2
Me.SecondaryChart.Add3SigmaControlLimits(target, lowlim, highlim, limitcheck)
Me.SecondaryChart.ControlLimitLineFillMode = True
```



Control Limit Fill Option used with +-1, 2 and 3-sigma control limits

You can also add additional control limits one at a time. By default you get the +-3-sigma control limits. So additional control limits should be considered +-2-sigma and +-1-sigma control limits. Do not confuse control limits with specification limits, which must be added using the **AddSpecLimit** method. There are two steps to adding additional control limits: creating a SPCControlLimitRecord object for the new control limit, and adding
the control limit to the chart using the charts AddAdditionalControlLimit method. It is critical that you add them in a specific order, that order being:

Primary Chart	SPC_LOWER_CONTROL_LIMIT_2	(2-sigma lower limit)
Primary Chart	SPC_UPPER_CONTROL_LIMIT_2	(2-sigma lower limit)
Primary Chart	SPC_LOWER_CONTROL_LIMIT_1	(1-sigma lower limit)

Primary Chart	SPC_UPPER_CONTROL_LIMIT_1	(1-sigma lower limit)
Secondary Chart	SPC_LOWER_CONTROL_LIMIT_2	(2-sigma lower limit)
Secondary Chart	SPC_UPPER_CONTROL_LIMIT_2	(2-sigma lower limit)
Secondary Chart	SPC_LOWER_CONTROL_LIMIT_1	(1-sigma lower limit)
Secondary Chart	SPC_UPPER_CONTROL_LIMIT_1	(1-sigma lower limit)

[C#]

```
double sigma2 = 2.0;
double sigma1 = 1.0;


this.PrimaryChart.GetControlLimitData(SPCChartObjects.SPC_LOWER_CONTROL_LIMIT).LimitValue
= 74;

this.PrimaryChart.GetControlLimitData(SPCChartObjects.SPC_UPPER_CONTROL_LIMIT).LimitValue
= 76;

this.PrimaryChart.GetControlLimitData(SPCChartObjects.SPC_CONTROL_TARGET).LimitValue = 75;

// Create multiple limits
// For PrimaryChart
SPCControlLimitRecord  lcl2 = new SPCControlLimitRecord(this.ChartData,
SPCControlLimitRecord.SPC_LOWERTHAN_LIMIT, 74.33,"LCL2", "LCL2");
SPCControlLimitRecord  ucl2 = new SPCControlLimitRecord(this.ChartData,
SPCControlLimitRecord.SPC_GREATERTHAN_LIMIT, 75.66,"UCL2", "UCL2");

this.PrimaryChart.AddAdditionalControlLimit(lcl2,
SPCChartObjects.SPC_LOWER_CONTROL_LIMIT_2, sigma2);
this.PrimaryChart.AddAdditionalControlLimit(ucl2,
SPCChartObjects.SPC_UPPER_CONTROL_LIMIT_2, sigma2);

SPCControlLimitRecord  lcl3 = new SPCControlLimitRecord(this.ChartData,
SPCControlLimitRecord.SPC_LOWERTHAN_LIMIT, 74.66,"LCL1", "LCL1");
SPCControlLimitRecord  ucl3 = new SPCControlLimitRecord(this.ChartData,
SPCControlLimitRecord.SPC_GREATERTHAN_LIMIT, 75.33,"UCL1", "UCL1");

this.PrimaryChart.AddAdditionalControlLimit(lcl3,
SPCChartObjects.SPC_LOWER_CONTROL_LIMIT_1, sigma1);
this.PrimaryChart.AddAdditionalControlLimit(ucl3,
SPCChartObjects.SPC_UPPER_CONTROL_LIMIT_1, sigma1);

           // For Secondary Chart

this.SecondaryChart.GetControlLimitData(SPCChartObjects.SPC_LOWER_CONTROL_LIMIT).LimitValu
e = 2;

this.SecondaryChart.GetControlLimitData(SPCChartObjects.SPC_UPPER_CONTROL_LIMIT).LimitValu
e = 0;

this.SecondaryChart.GetControlLimitData(SPCChartObjects.SPC_CONTROL_TARGET).LimitValue =
1;

SPCControlLimitRecord  lcl4 = new SPCControlLimitRecord(this.ChartData,
SPCControlLimitRecord.SPC_LOWERTHAN_LIMIT, 0.33,"LCL2", "LCL2");
SPCControlLimitRecord  ucl4 = new SPCControlLimitRecord(this.ChartData,
SPCControlLimitRecord.SPC_GREATERTHAN_LIMIT, 1.66,"UCL2", "UCL2");

this.SecondaryChart.AddAdditionalControlLimit(lcl4,
SPCChartObjects.SPC_LOWER_CONTROL_LIMIT_2, sigma2);
this.SecondaryChart.AddAdditionalControlLimit(ucl4,
SPCChartObjects.SPC_UPPER_CONTROL_LIMIT_2, sigma2);

SPCControlLimitRecord  lcl5 = new SPCControlLimitRecord(this.ChartData,
SPCControlLimitRecord.SPC_LOWERTHAN_LIMIT, 0.66,"LCL1", "LCL1");
```

```
SPCControlLimitRecord  ucl5 = new SPCControlLimitRecord(this.ChartData,
SPCControlLimitRecord.SPC_GREATERTHAN_LIMIT, 1.33,"UCL1", "UCL1");

this.SecondaryChart.AddAdditionalControlLimit(lcl5,
SPCChartObjects.SPC_LOWER_CONTROL_LIMIT_1, sigma1);
this.SecondaryChart.AddAdditionalControlLimit(ucl5,
SPCChartObjects.SPC_UPPER_CONTROL_LIMIT_1, sigma1);

this.PrimaryChart.ControlLimitLineFillMode = true;

this.SecondaryChart.ControlLimitLineFillMode = true;
```

[VB]

```
Dim sigma2 As Double = 2.0
Dim sigma1 As Double = 1.0
' Create multiple limits
For PrimaryChart

Me.PrimaryChart.GetControlLimitData(SPCChartObjects.SPC_LOWER_CONTROL_LIMIT).LimitValue =
74

Me.PrimaryChart.GetControlLimitData(SPCChartObjects.SPC_UPPER_CONTROL_LIMIT).LimitValue =
76
        Me.PrimaryChart.GetControlLimitData(SPCChartObjects.SPC_CONTROL_TARGET).LimitValue
= 75

Dim lcl2 As New SPCControlLimitRecord(Me.ChartData, _
   SPCControlLimitRecord.SPC_LOWERTHAN_LIMIT, 74.33, "LCL2", "LCL2") '
Dim ucl2 As New SPCControlLimitRecord(Me.ChartData, _
     SPCControlLimitRecord.SPC_GREATERTHAN_LIMIT, 75.66, "UCL2", "UCL2")

Me.PrimaryChart.AddAdditionalControlLimit(lcl2, _
    SPCChartObjects.SPC_LOWER_CONTROL_LIMIT_2, sigma2)
Me.PrimaryChart.AddAdditionalControlLimit(ucl2, _
    SPCChartObjects.SPC_UPPER_CONTROL_LIMIT_2, sigma2)

Dim lcl3 As New SPCControlLimitRecord(Me.ChartData, _
      SPCControlLimitRecord.SPC_LOWERTHAN_LIMIT, 74.66, "LCL1", "LCL1")
Dim ucl3 As New SPCControlLimitRecord(Me.ChartData, _
      SPCControlLimitRecord.SPC_GREATERTHAN_LIMIT, 75.33, "UCL1", "UCL1")

Me.PrimaryChart.AddAdditionalControlLimit(lcl3, _
      SPCChartObjects.SPC_LOWER_CONTROL_LIMIT_1, sigma1)
Me.PrimaryChart.AddAdditionalControlLimit(ucl3, _
     SPCChartObjects.SPC_UPPER_CONTROL_LIMIT_1, sigma1)

        ' For Secondary Chart

Me.SecondaryChart.GetControlLimitData(SPCChartObjects.SPC_LOWER_CONTROL_LIMIT).LimitValue
= 0

Me.SecondaryChart.GetControlLimitData(SPCChartObjects.SPC_UPPER_CONTROL_LIMIT).LimitValue
= 2

Me.SecondaryChart.GetControlLimitData(SPCChartObjects.SPC_CONTROL_TARGET).LimitValue = 1

Dim lcl4 As SPCControlLimitRecord = New SPCControlLimitRecord(Me.ChartData,
SPCControlLimitRecord.SPC_LOWERTHAN_LIMIT, 0.33, "LCL2", "LCL2")
Dim ucl4 As SPCControlLimitRecord = New SPCControlLimitRecord(Me.ChartData,
SPCControlLimitRecord.SPC_GREATERTHAN_LIMIT, 1.66, "UCL2", "UCL2")

Me.SecondaryChart.AddAdditionalControlLimit(lcl4,
SPCChartObjects.SPC_LOWER_CONTROL_LIMIT_2, sigma2)
Me.SecondaryChart.AddAdditionalControlLimit(ucl4,
SPCChartObjects.SPC_UPPER_CONTROL_LIMIT_2, sigma2)

Dim lcl5 As SPCControlLimitRecord = New SPCControlLimitRecord(Me.ChartData,
SPCControlLimitRecord.SPC_LOWERTHAN_LIMIT, 0.66, "LCL1", "LCL1")
Dim ucl5 As SPCControlLimitRecord = New SPCControlLimitRecord(Me.ChartData,
SPCControlLimitRecord.SPC_GREATERTHAN_LIMIT, 1.33, "UCL1", "UCL1")
```

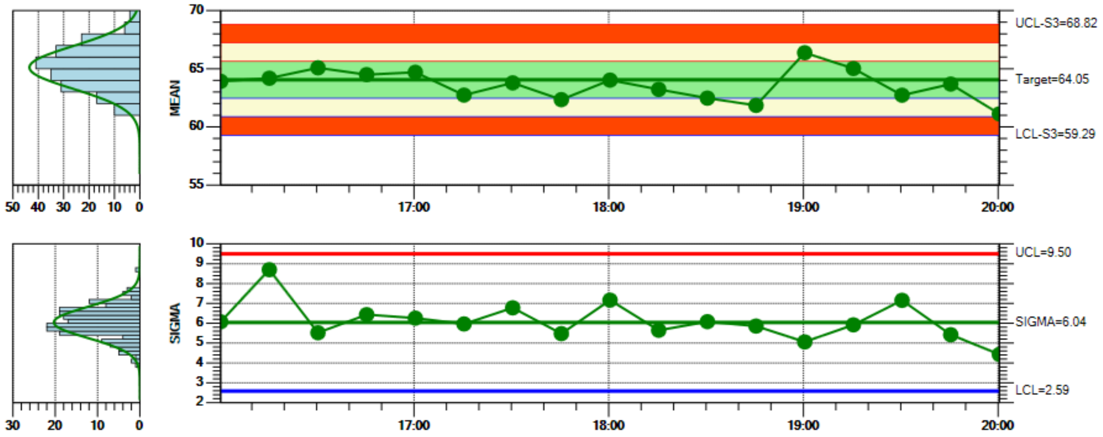
```
Me.SecondaryChart.AddAdditionalControlLimit(lcl5,
SPCChartObjects.SPC_LOWER_CONTROL_LIMIT_1, sigma1)
Me.SecondaryChart.AddAdditionalControlLimit(ucl5,
SPCChartObjects.SPC_UPPER_CONTROL_LIMIT_1, sigma1)

Me.PrimaryChart.ControlLimitLineFillMode = True
Me.SecondaryChart.ControlLimitLineFillMode = True
```

Special Note – When you create a **SPCControlLimitRecord** object, you can specify an actual limit level. If you do not call the charts **AutoCalculateControlLimits** method, the control limit will be displayed at that value. If you do call **AutoCalculateControlLimits** method, the auto-calculated value overrides the initial value (0.0 in the examples above). When you call the charts **AddAdditionalControlLimits** method, you specify the sigma level that is used by the **AutoCalculateControlLimits** to calculate the control limit level.

If you want the control limits displayed as filled areas, set the charts **ControlLimitLineFillMode** property True.

```
[C#]
this.PrimaryChart.ControlLimitLineFillMode = true;
```

This will fill each control limit line from the limit line to the target value of the chart. In order for the fill to work properly, you must set this property after you define all additional control limits. Also, you must add the outer most control limits (SPC_UPPER_CONTROL_LIMIT_3 and SPC_LOWER_CONTROL_LIMIT_3) first, followed by the next outer most limits (SPC_UPPER_CONTROL_LIMIT_2 and SPC_LOWER_CONTROL_LIMIT_2), followed by the inner most control limits (SPC_UPPER_CONTROL_LIMIT_1 and SPC_LOWER_CONTROL_LIMIT_1). This way the fill of the inner limits will partially cover the fill of the outer limits, creating the familiar striped look you want to see.



Western Electric (WE) Runtime Rules

The normal SPC control limit rules display at the 3-sigma level, both high and low. In this case, a simple threshold test determines if a process is in, or out of control. Other, more complex tests rely on more complicated decision-making criteria. The most popular of these are the Western

Electric Rules, also know as the WE Rules, or WE Runtime Rules. These rules utilize historical data for the eight most recent sample intervals and look for a non-random pattern that can signify that the process is out of control, before reaching the normal +-3 sigma limits. A processed is considered out of control if any of the following criteria are met:

*Starting with Revision 3.0, we have added additional, industry standard rules to the software: Nelson, AAIG, Juran, Hughes, Gitlow, Westgard, and Duncan. In addition, you can mix and match rules from the different rule sets, and you can create your own custom rules using our standardize rule templates. See Chapter 8, for more information.

1. **The most recent point plots outside one of the 3-sigma control limits.** If a point lies outside either of these limits, there is only a 0.3% chance that this was caused by the normal process.

2. **Two of the three most recent points plot outside and on the same side as one of the 2-sigma control limits.** The probability that any point will fall outside the warning limit is only 5%. The chances that two out of three points in a row fall outside the warning limit is only about 1%.

3. **Four of the five most recent points plot outside and on the same side as one of the 1-sigma control limits.** In normal processing, 68% of points fall within one sigma of the mean, and 32% fall outside it. The probability that 4 of 5 points fall outside of one sigma is only about 3%.

4. **Eight out of the last eight points plot on the same side of the center line, or target value.** Sometimes you see this as 9 out of 9, or 7 out of 7. There is an equal chance that any given point will fall above or below the mean. The chances that a point falls on the same side of the mean as the one before it is one in two. The odds that the next point will also fall on the same side of the mean is one in four. The probability of getting eight points on the same side of the mean is only around 1%.

These rules apply to both sides of the center line at a time. Therefore, there are eight actual alarm conditions: four for the above center line sigma levels and four for the below center line sigma levels.

There are also additional WE Rules for trending. These are often referred to as WE Supplemental Rules. Don't rely on the rule number, often these are listed in a different order.

5. **Six points in a row increasing or decreasing.** The same logic is used here as for rule 4 above. Sometimes this rule is changed to seven points rising or falling.

6. **Fifteen points in a row within one sigma.** In normal operation, 68% of points will fall within one sigma of the mean. The probability that 15 points in a row will do so, is less than 1%.

7. **Fourteen points in a row alternating direction.** The chances that the second point is always higher than (or always lower than) the preceding point, for all seven pairs is only about 1%.

8. **Eight points in a row outside one sigma.** Since 68% of points lie within one sigma of the mean, the probability that eight points in a row fall outside of the one-sigma line is less than 1%.

While the techniques in the previous section can be used to draw multiple SPC control limit lines on the graph, at the +-1, 2, 3 sigma levels for example, they do not provide for the (x out of y) control criteria used in evaluating the WE rules. The software can be explicitly flagged to evaluate out of control alarm conditions according to the WE Rules, instead of the default +-3 sigma control criteria. It will create alarm lines at the +-1, 2, and 3-sigma control limits and the center line. It will also automatically establish the eight alarm conditions associated with the WE rules. Set the WE rules flag using the PrimaryChart (or SecondaryChart) **UseWERuntimeRules** method. When the variable control charts **AutoCalculatedControlLimits** method is called, the software automatically calculates all of the appropriated control limits, based on the current data. The example below is extracted from the WERulesVariableControlChart.XBarRCharts example program.

If you want to include the WE Trending (Supplemental) rules, in addition to the regular WE Runtime rules, call UseWERuntimeAndSupplementalRules in place of UseWERuntimeRules.

[C#]

```
this.ChartData.AlarmStateEventHandler +=
       new SPCControlLimitAlarmEventDelegate(this.SPCControlLimitAlarm);
// don't generate alarms in initial data simulation
this.ChartData.AlarmStateEventEnable = false;

this.PrimaryChart.UseWERuntimeRules();
//this.PrimaryChart.UseWERuntimeAndSupplementalRules();

// Must have data loaded before any of the Auto.. methods are called
 SimulateData(100, 20);

// Calculate the SPC control limits for both graphs of the current SPC chart (X-Bar R)
this.AutoCalculateControlLimits();

// throw out values used to calculate limits
this.ChartData.ResetSPCChartData();

//  generate alarms now
this.ChartData.AlarmStateEventEnable = true;
// Must have data loaded before any of the Auto.. methods are called
// Check this data against rules
SimulateData(100, 23);
```

[VB]

```
AddHandler Me.ChartData.AlarmStateEventHandler, AddressOf Me.SPCControlLimitAlarm
' don't generate alarms in initial data simulation
Me.ChartData.AlarmStateEventEnable = False

Me.PrimaryChart.UseWERuntimeRules()
‘ Me.PrimaryChart.UseWERuntimeAndSupplementalRules()
```

```
SimulateData()

' Calculate the SPC control limits for the X-Bar part of the current SPC chart (X-Bar R)
Me.AutoCalculateControlLimits()
' Scale the y-axis of the X-Bar chart to display all data and control limits
Me.AutoScalePrimaryChartYRange()
' Scale the y-axis of the Range chart to display all data and control limits
Me.AutoScaleSecondaryChartYRange()
' Rebuild the chart using the current data and settings
Me.RebuildChartUsingCurrentData()

'  generate alarms starting now
Me.ChartData.AlarmStateEventEnable = True
```

If you have setup a method for processing alarm events, the software will call the classes alarm processing method, where you can take appropriate action. If a time interval has multiple alarms, i.e. more than one of the four WR Runtime rules are broken, only the one with the lowest WE rule number is vectored to the alarm event processing routine.

[C#]

```
    this.ChartData.AlarmStateEventHandler +=
        new SPCControlLimitAlarmEventDelegate(this.SPCControlLimitAlarm);
    // don't generate alarms in initial data simulation
    this.ChartData.AlarmStateEventEnable = false;

    this.PrimaryChart.UseWERuntimeRules();
    // Must have data loaded before any of the Auto.. methods are called
     SimulateData(100, 20);

    // Calculate the SPC control limits
    this.AutoCalculateControlLimits();

    // throw out values used to calculate limits
    this.ChartData.ResetSPCChartData();

    // generate alarms now
    this.ChartData.AlarmStateEventEnable = true;
    // Must have data loaded before any of the Auto.. methods are called
    // Check this data against rules
    SimulateData(100, 23);
}


private void SPCControlLimitAlarm(object sender, SPCControlLimitAlarmArgs e)
{
    SPCControlLimitRecord alarm = e.EventAlarm;
    double alarmlimitvalue = alarm.ControlLimitValue;
    String alarmlimitvaluestring = alarmlimitvalue.ToString();

    SPCControlChartData spcData = alarm.SPCProcessVar;
    SPCCalculatedValueRecord spcSource = e.SPCSource;
    String calculatedvaluestring = spcSource.CalculatedValue.ToString();

    String message = alarm.AlarmMessage;
    ChartCalendar timestamp = spcData.TimeStamp;
    String timestampstring = timestamp.ToString();

    String notesstring = "\n" + timestampstring + " " + message + "=" + "\n" +
    alarmlimitvaluestring + " Current Value" + "=" + calculatedvaluestring;
    if (alarm.AlarmState)
    //             Console.Out.WriteLine(notesstring);
        this.ChartData.AppendNotesString(notesstring, true);
}
```

[VB]

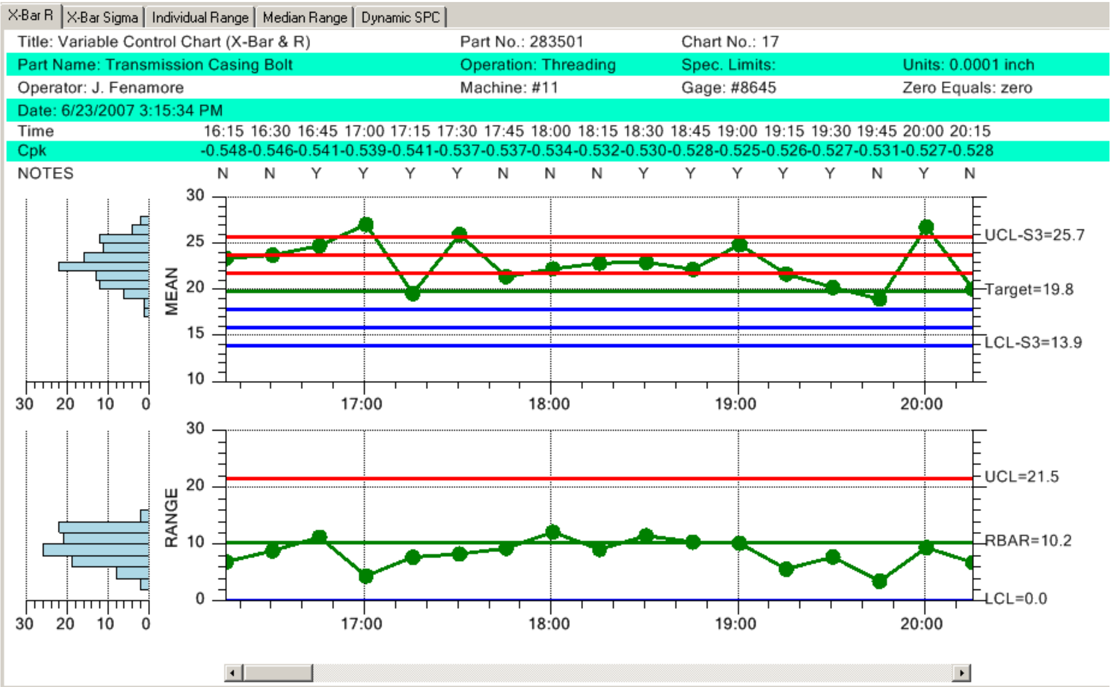
```
 AddHandler Me.ChartData.AlarmStateEventHandler, AddressOf Me.SPCControlLimitAlarm
' don't generate alarms in initial data simulation
  Me.ChartData.AlarmStateEventEnable = False

  Me.PrimaryChart.UseWERuntimeRules()

  SimulateData()

  ' Calculate the SPC control limits for the X-Bar part of the current SPC chart (X-Bar R)
  Me.AutoCalculateControlLimits()
  ' Scale the y-axis of the X-Bar chart to display all data and control limits
  Me.AutoScalePrimaryChartYRange()
  ' Scale the y-axis of the Range chart to display all data and control limits
  Me.AutoScaleSecondaryChartYRange()
  ' Rebuild the chart using the current data and settings
  Me.RebuildChartUsingCurrentData()

  '  generate alarms starting now
  Me.ChartData.AlarmStateEventEnable = True
  End Sub 'SPCControlLimitAlarm
 .
 .
 .

Private Sub SPCControlLimitAlarm(ByVal sender As Object, ByVal e As
SPCControlLimitAlarmArgs)
  Dim alarm As SPCControlLimitRecord = e.EventAlarm
  Dim alarmlimitvalue As Double = alarm.ControlLimitValue
  Dim alarmlimitvaluestring As [String] = alarmlimitvalue.ToString()
  Dim spcData As SPCControlChartData = alarm.SPCProcessVar
  Dim spcSource As SPCCalculatedValueRecord = e.SPCSource
  Dim calculatedvaluestring As [String] = spcSource.CalculatedValue.ToString()

  Dim message As [String] = alarm.AlarmMessage
  Dim timestamp As ChartCalendar = spcData.TimeStamp
  Dim timestampstring As [String] = timestamp.ToString()
  Dim notesstring As String =
       "\n" + timestampstring + " " + message + "=" + "\n" + _
       alarmlimitvaluestring + " Current Value" + "=" + calculatedvaluestring

  If alarm.AlarmState Then
     ' Console.Out.WriteLine((timestampstring + " " + message + "=" +
alarmlimitvaluestring + " Current Value" + "=" + calculatedvaluestring))
     Me.ChartData.AppendNotesString(notesstring, True)
  End If
End Sub 'SPCControlLimitAlarm
```

If you want multiple alarms for a time interval vectored to the alarm processing routine (i.e. it is possible that a time period has WE1, WE2, WE3 and WE4 alarms), set the SPCControlChartData property to SPCControlChartData.REPORT_ALL_ALARMS.

```
this.ChartData.AlarmReportMode = SPCControlChartData.REPORT_ALL_ALARMS
```

The resulting X-Bar R SPC Chart with WE Runtime Rules looks something like this. In this example, the WR Rules violations are processed by the **SPCControlLimitAlarm** method, where the alarm condition is added to the Notes record for the appropriate sample interval. The Y in the

Notes line indicates that an alarm record has been saved for that time interval, and you can click on the Y to see the note describing the alarm condition.



Title: Variable Control Chart (X-Bar & R) Part No.: 283501 Chart No.: 17

Part Name: Transmission Casing Bolt Operation: Threading Spec. Limits: Units: 0.0001 inch

Operator: J. Fenamore Machine: #11 Gage: #8645 Zero Equals: zero

Date: 6/23/2007 3:15:34 PM

```
Time   16:15 16:30 16:45 17:00 17:15 17:30 17:45 18:00 18:15 18:30 18:45 19:00 19:15 19:30 19:45 20:00 20:15
Cpk    -0.548-0.546-0.541-0.539-0.541-0.537-0.537-0.534-0.532-0.530-0.528-0.525-0.526-0.527-0.531-0.527-0.528
NOTES    N     N     Y     Y     Y     Y     N     N     N     Y     Y     Y     Y     Y     N     Y     N
```

Specification Limits

Specification limits are not to be confused with the SPC Control Limits discussed in the previous sections. Specification limits are imposed externally and are not calculated based on the manufacturing process under control. They represent the maximum deviation allowable for the process variable being measured. They are calculated based on input from customers and/or engineering. Usually specification limits are going to be wider than the SPC 3-sigma limits, because you want the SPC control limits to trip before you get to the specification limits. The SPC control limits give you advance notice that the process is going south before you start rejecting parts based on specification limits. You can display specification limits in the same chart as SPC control limits. Use the
AddSpecLimit method of the PrimaryChart or SecondaryChart.

[C#]

```
this.PrimaryChart.AddSpecLimit(SPCChartObjects.SPC_LOWER_SPEC_LIMIT, 18.3, "L
SPEC", new ChartAttribute(Color.Green, 3.0));
this.PrimaryChart.AddSpecLimit(SPCChartObjects.SPC_UPPER_SPEC_LIMIT, 39.1, "H
SPEC", new ChartAttribute(Color.Yellow, 3.0));
```

[VB]

```
Me.PrimaryChart.AddSpecLimit(SPCChartObjects.SPC_LOWER_SPEC_LIMIT, 18.3, "L
SPEC", New ChartAttribute(Color.Green, 3.0))
Me.PrimaryChart.AddSpecLimit(SPCChartObjects.SPC_UPPER_SPEC_LIMIT, 39.1, "H
SPEC", New ChartAttribute(Color.Yellow, 3.0))
```

Chart Y-Scale

You can set the minimum and maximum values of the two charts y-scales manually using the **PrimaryChart.MinY**, **PrimaryChart.MaxY**, **SecondaryChartMinY** and **SecondaryChartMaxY** properties.

[C#]

```
// Set initial scale of the y-axis of the mean chart
// If you are calling AutoScalePrimaryChartYRange this isn't really needed
    this.PrimaryChart.MinY = 0;
    this.PrimaryChart.MaxY = 40;

// Set initial scale of the y-axis of the range chart
// If you are calling AutoScaleSecondaryChartYRange this isn't really needed
    this.SecondaryChart.MinY = 0;
    this.SecondaryChart.MaxY = 40;
```

[VB]

```
' Set initial scale of the y-axis of the mean chart
' If you are calling AutoScalePrimaryChartYRange this isn't really needed
Me.PrimaryChart.MinY = 0
Me.PrimaryChart.MaxY = 40

' Set initial scale of the y-axis of the range chart
' If you are calling AutoScaleSecondaryChartYRange this isn't really needed
Me.SecondaryChart.MinY = 0
Me.SecondaryChart.MaxY = 40
```

It is easiest to just call the auto-scale routines after the chart has been initialized with data, and any control limits calculated.

[C#]

```
// Must have data loaded before any of the Auto.. methods are called
    SimulateData();

// Calculate the SPC control limits for both graphs of the current SPC chart
    this.AutoCalculateControlLimits();

// Scale the y-axis of the X-Bar chart to display all data and control limits
    this.AutoScalePrimaryChartYRange();
// Scale the y-axis of the Range chart to display all data and control limits
    this.AutoScaleSecondaryChartYRange();
```

[VB]

```
' Must have data loaded before any of the Auto.. methods are called
SimulateData()

' Calculate the SPC control limits for both graphs of the current SPC chart
Me.AutoCalculateControlLimits()

' Scale the y-axis of the X-Bar chart to display all data and control limits
Me.AutoScalePrimaryChartYRange()
' Scale the y-axis of the Range chart to display all data and control limits
Me.AutoScaleSecondaryChartYRange()
```

Once all of the graph parameters are set, call the method **RebuildChartUsingCurrentData**.

[C#]

```
// Rebuild the chart using the current data and settings
this.RebuildChartUsingCurrentData();
```

[VB]

```
// Rebuild the chart using the current data and settings
Me.RebuildChartUsingCurrentData
```

If, at any future time you change any of the chart properties, you will need to call **RebuildChartUsingCurrentData** to force a rebuild of the chart, taking into account the current properties. **RebuildChartUsingCurrentData** also invalidates the chart and forces a redraw. Our examples that update dynamically demonstrate this technique. The chart is setup with some initial settings and data values. As data is added in real-time to the graph, the chart SPC limits, and y-scales are constantly recalculated to take into account new data values. The following code is extracted from the **TimeVariableControlCharts.DynamicXBarRChart** example.

[C#]

```
private void timer1_Tick(object sender, System.EventArgs e)
{
    if (this.IsDesignMode) return;
    ChartCalendar timestamp = (ChartCalendar) startTime.Clone();

    // Use the ChartData sample simulator to make an array of sample data
    DoubleArray samples = this.ChartData.SimulateMeasurementRecord(30);
    // Add the new sample subgroup to the chart
    this.ChartData.AddNewSampleRecord(timestamp, samples);

// Calculate the SPC control limits for the X-Bar part of the current SPC chart
    this.AutoCalculateControlLimits();
// Scale the y-axis of the X-Bar chart to display all data and control limits
    this.AutoScalePrimaryChartYRange();
    // Scale the y-axis of the Range chart to display all data and control limits
    this.AutoScaleSecondaryChartYRange();
    // Rebuild and draw the chart using the current data and settings
    this.RebuildChartUsingCurrentData();
    // Simulate timeincrementminutes  minute passing
    startTime.Add(ChartObj.MINUTE, timeincrementminutes);
}
```

[VB]

```
Private Sub Timer1_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Timer1.Tick
    If Me.IsDesignMode Then
        Return
    End If
    Dim timestamp As ChartCalendar = CType(startTime.Clone(), ChartCalendar)

    ' Use the ChartData sample simulator to make an array of sample data
    Dim samples As DoubleArray = Me.ChartData.SimulateMeasurementRecord(30)
    ' Add the new sample subgroup to the chart
    Me.ChartData.AddNewSampleRecord(timestamp, samples)

' Calculate the SPC control limits for the X-Bar part of the current SPC chart
    Me.AutoCalculateControlLimits()
   ' Scale the y-axis of the X-Bar chart to display all data and control limits
    Me.AutoScalePrimaryChartYRange()
    ' Scale the y-axis of the Range chart to display all data and control limits
    Me.AutoScaleSecondaryChartYRange()
    ' Rebuild and draw the chart using the current data and settings
    Me.RebuildChartUsingCurrentData()
    ' Simulate timeincrementminutes  minute passing
    startTime.Add(ChartObj.MINUTE, timeincrementminutes)
End Sub
```

Updating Chart Data

The real-time example above demonstrates how the SPC chart data is updated, using the **ChartData.AddNewSampleRecord** method. In this case, the chart data updates with each timer tick event, though it could just as easily be any other type of event. If you have already collected all of your data and just want to plot it all at once, use a simple loop like most of our examples do to update the data.

[C#]

```
private void SimulateData()
{
    for (int i=0; i < 200; i++)
    {
        ChartCalendar timestamp =
        (ChartCalendar) startTime.Clone(); // use this for time row, not graphs
        // Use the ChartData sample simulator to make an array of sample data
        DoubleArray samples = this.ChartData.SimulateMeasurementRecord(33, 9);
        // Add the new sample subgroup to the chart
        this.ChartData.AddNewSampleRecord(timestamp, samples);
        // increment simulated time by timeincrementminutes minutes
        startTime.Add(ChartObj.MINUTE, sampleincrement);
    }
}
```

[VB]

```
Private Sub SimulateData()
   Dim i As Integer
   For i = 0 To 199
      Dim timestamp As ChartCalendar = CType(startTime.Clone(), ChartCalendar)
      ' use this for time row, not graphs
      ' Use the ChartData sample simulator to make an array of sample data
      Dim samples As DoubleArray = Me.ChartData.SimulateMeasurementRecord(33, 9)
      ' Add the new sample subgroup to the chart
      Me.ChartData.AddNewSampleRecord(timestamp, samples)
      ' increment simulated time by timeincrementminutes minutes
      startTime.Add(ChartObj.MINUTE, sampleincrement)
   Next i
End Sub 'SimulateData
```

In this example the sample data and the time stamp for each sample record is simulated. In your application, you will probably be reading the sample record values from some sort of database or file, along with the actual time stamp for that data.

If you want to include a text note in the sample record, use one of the **ChartData.AddNewSampleRecord** overrides that have a *notes* parameter.

[C#]

```
private void SimulateData()
{  String notesstring = "";
    for (int i=0; i < 200; i++)
    {
        ChartCalendar timestamp = (ChartCalendar) startTime.Clone();
        // Use the ChartData sample simulator to make an array of sample data
        DoubleArray samples = this.ChartData.SimulateMeasurementRecord(30, 10);
        double r = ChartSupport.GetRandomDouble();
        if (r < 0.1) // make a note on every tenth item, on average
        notesstring = "Note for sample subgroup #" +
                      i.ToString() +
                      ". This sample is flagged as having some sort of problem";
               else
                   notesstring = "";
               // Add the new sample subgroup to the chart
        this.ChartData.AddNewSampleRecord(timestamp, samples, notesstring);
        // increment simulated time by timeincrementminutes minutes
        startTime.Add(ChartObj.MINUTE, timeincrementminutes);
    }
}
```

[VB]

```
Private Sub SimulateData()
   Dim notesstring As [String] = ""
   Dim i As Integer
   For i = 0 To 199
      Dim timestamp As ChartCalendar = CType(startTime.Clone(), ChartCalendar)
      ' Use the ChartData sample simulator to make an array of sample data
      Dim samples As DoubleArray = Me.ChartData.SimulateMeasurementRecord(30, 10)
      Dim r As Double = ChartSupport.GetRandomDouble()
      If r < 0.1 Then ' make a note on every tenth item, on average
         notesstring = "Note for sample subgroup #" + i.ToString() + _
                      ". This sample is flagged as having some sort of problem"
      Else
         notesstring = ""
      End If ' Add the new sample subgroup to the chart
      Me.ChartData.AddNewSampleRecord(timestamp, samples, notesstring)
      ' increment simulated time by timeincrementminutes minutes
      startTime.Add(ChartObj.MINUTE, timeincrementminutes)
   Next i
End Sub 'SimulateData
```

There are situations where you might want to add, change, modify, or append a note for a sample subgroup after the **AddNewSampleRecord** method has already been called for the sample subgroup. This can happen if the **AddNewSampleRecord** method call generates an alarm event. In the alarm event processing routine, you can add code that adds a special note to the sample subgroup that generated the alarm. Use the **ChartData.SetNotesString** or **ChartData.AppendNotesString** methods to add notes to the current record, separate from the **AddNewSampleRecord** method.

Extracted from the VariableControlCharts.DynamicXBarRChart example program.

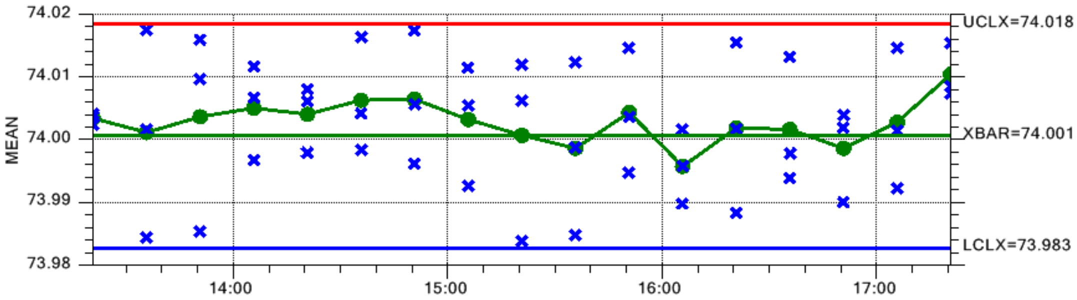
[C#]

```
private void SPCControlLimitAlarm(object sender, SPCControlLimitAlarmArgs e)
{
    SPCControlLimitRecord alarm = e.EventAlarm;
    double alarmlimitvalue = alarm.ControlLimitValue;
    String alarmlimitvaluestring = alarmlimitvalue.ToString();

    SPCControlChartData spcData = alarm.SPCProcessVar;
    SPCCalculatedValueRecord spcSource = e.SPCSource;
    String calculatedvaluestring = spcSource.CalculatedValue.ToString();

    String message = alarm.AlarmMessage;
    ChartCalendar timestamp = spcData.TimeStamp;
    String timestampstring = timestamp.ToString();
    String notesstring = "\n" + timestampstring + " " + message + "=" +
        "\n" +  alarmlimitvaluestring + " Current Value" + "=" +
        calculatedvaluestring;

// Append a notes string to the current record.
    if (alarm.AlarmState)
      this.ChartData.AppnedNotesString(notesstring, true);

}
```

[VB]

```
Private Sub SPCControlLimitAlarm(ByVal sender As Object, ByVal e As
SPCControlLimitAlarmArgs)
   Dim alarm As SPCControlLimitRecord = e.EventAlarm
   Dim alarmlimitvalue As Double = alarm.ControlLimitValue
   Dim alarmlimitvaluestring As [String] = alarmlimitvalue.ToString()

   Dim spcData As SPCControlChartData = alarm.SPCProcessVar
   Dim spcSource As SPCCalculatedValueRecord = e.SPCSource
   Dim calculatedvaluestring As [String] = spcSource.CalculatedValue.ToString()

   Dim message As [String] = alarm.AlarmMessage
   Dim timestamp As ChartCalendar = spcData.TimeStamp
   Dim timestampstring As [String] = timestamp.ToString()
   Dim notesstring As String = "\n" + timestampstring + " " + message _
    + "=" + "\n" +   alarmlimitvaluestring + " Current Value" + "=" +  _
    calculatedvaluestring

   If alarm.AlarmState Then
      Me.ChartData.AppendNotesString(notesstring, True)
   End If
End Sub 'SPCControlLimitAlarm
```

Scatter Plots of the Actual Sampled Data



If you want the actual sample data plotted along with the mean or median of the sample data, set the **PrimaryChart.PlotMeasurementValues** to true.

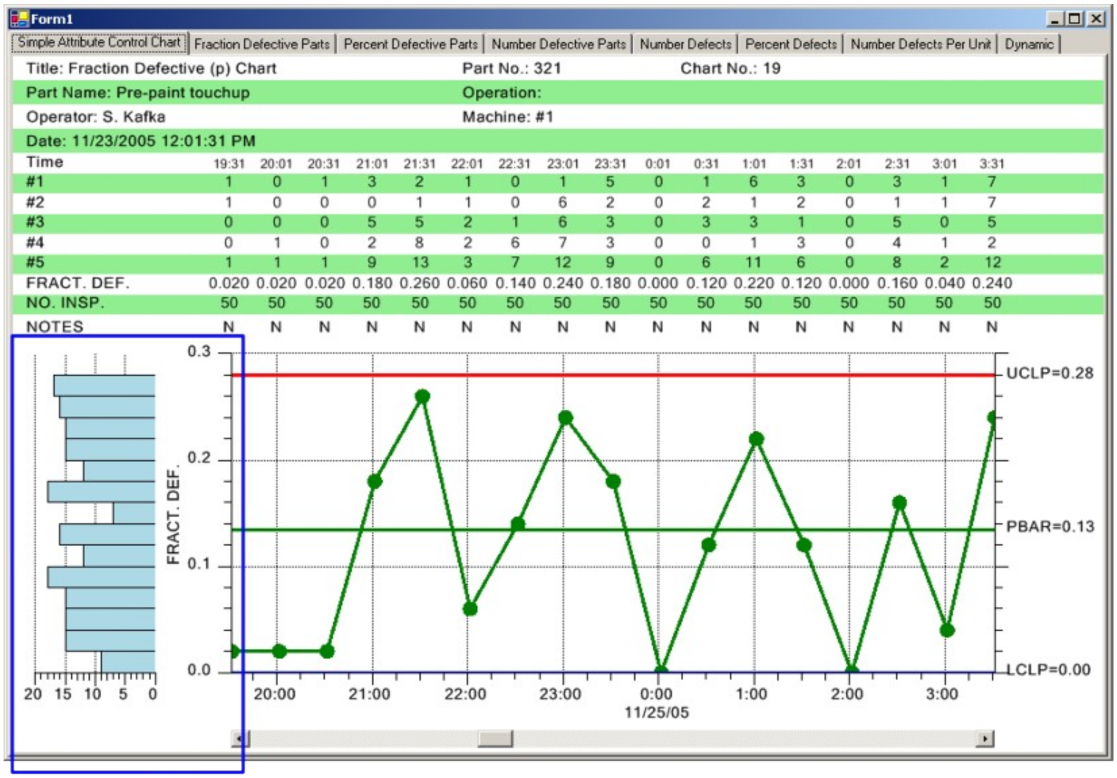
[C#]

```
// Plot individual sampled values as a scatter plot
this.PrimaryChart.PlotMeasurementValues = true;
```

[VB]

```
` Plot individual sampled values as a scatter plot
Me.PrimaryChart.PlotMeasurementValues = True
```

Enable the Chart ScrollBar



Title: Fraction Defective (p) Chart Part No.: 321 Chart No.: 19

Part Name: Pre-paint touchup Operation:

Operator: S. Kafka Machine: #1

Date: 11/23/2005 12:01:31 PM

Time	19:31	20:01	20:31	21:01	21:31	22:01	22:31	23:01	23:31	0:01	0:31	1:01	1:31	2:01	2:31	3:01	3:31
#1	1	0	1	3	2	1	0	1	5	0	1	6	3	0	3	1	7
#2	1	0	0	0	1	1	0	6	2	0	2	1	2	0	1	1	7
#3	0	0	0	5	5	2	1	6	3	0	3	3	1	0	5	0	5
#4	0	1	0	2	8	2	6	7	3	0	0	1	3	0	4	1	2
#5	1	1	1	9	13	3	7	12	9	0	6	11	6	0	8	2	12
FRACT. DEF.	0.020	0.020	0.020	0.180	0.260	0.060	0.140	0.240	0.180	0.000	0.120	0.220	0.120	0.000	0.160	0.040	0.240
NO. INSP.	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
NOTES	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N

Frequency Histogram

Set the **EnableScrollBar** property true to enable the chart scrollbar. You will then be able to window in on 8-20 sample subgroups at a time, from a much larger collection of measurement data representing hundreds or even thousands of subgroups, and use the scrollbar to move through the data, similar to paging through a spreadsheet.

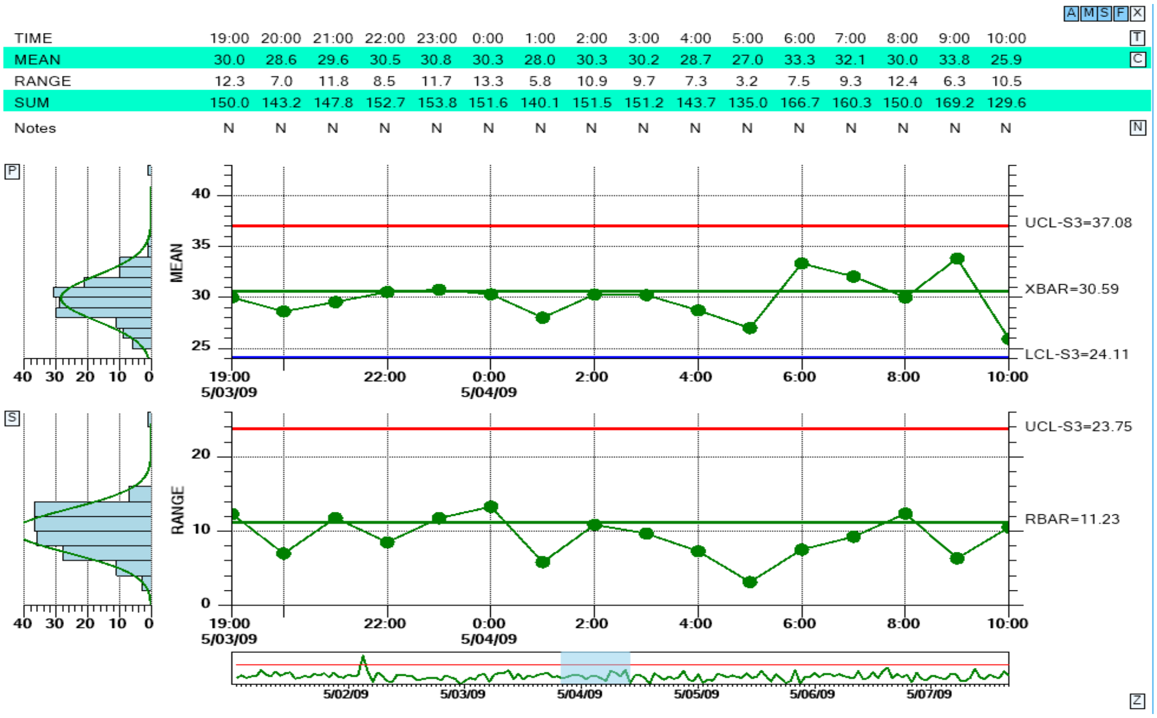
[C#]
```
// enable scroll bar
this.EnableScrollBar = true;
```

[VB]
```
' enable scroll bar
Me.EnableScrollBar = True
```

Once you have initialized the chart with data, and the scrollbar has a range associated with it, you can access the scrollbar using the charts **`HScrollBar1`** `property.`

Zooming as an option for the scrollbar

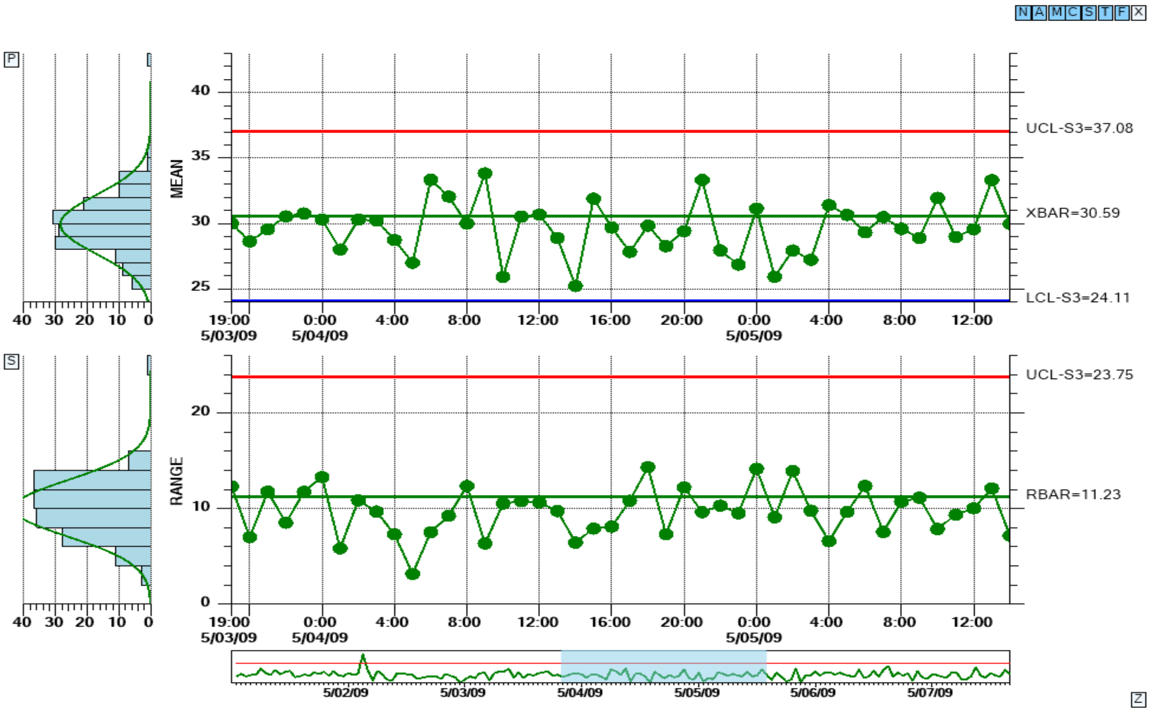
The horizontal scrollbar can be replaced (toggled on/off actually) using the UI, with a zoom window, by clicking on the z-button in the lower right corner. The user will be able to scroll the charts by dragging the transparent blue zoom window left or right, or expand the x-axis scale by dragging the left or right edge of the zoom window.



TIME	19:00	20:00	21:00	22:00	23:00	0:00	1:00	2:00	3:00	4:00	5:00	6:00	7:00	8:00	9:00	10:00
MEAN	30.0	28.6	29.6	30.5	30.8	30.3	28.0	30.3	30.2	28.7	27.0	33.3	32.1	30.0	33.8	25.9
RANGE	12.3	7.0	11.8	8.5	11.7	13.3	5.8	10.9	9.7	7.3	3.2	7.5	9.3	12.4	6.3	10.5
SUM	150.0	143.2	147.8	152.7	153.8	151.6	140.1	151.5	151.2	143.7	135.0	166.7	160.3	150.0	169.2	129.6
Notes	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N

Click on the Z button in the lower right corner and a zoom window will replace the scrollbar. Use the mouse to scroll the charts by dragging the transparent blue zoom window left or right, or expand the x-axis scale by dragging the left or right edge of the zoom window.

If the number of points in the display exceeds the initial setup value for the number of data points, the table is temporarily removed to prevent overlap of the table columns. If the number of data points is reduced to <= the initial number of points, the table will reappear.



If you use the zoom window to display a lot of points, the table at the top will disappear to prevent the table columns from overlapping.

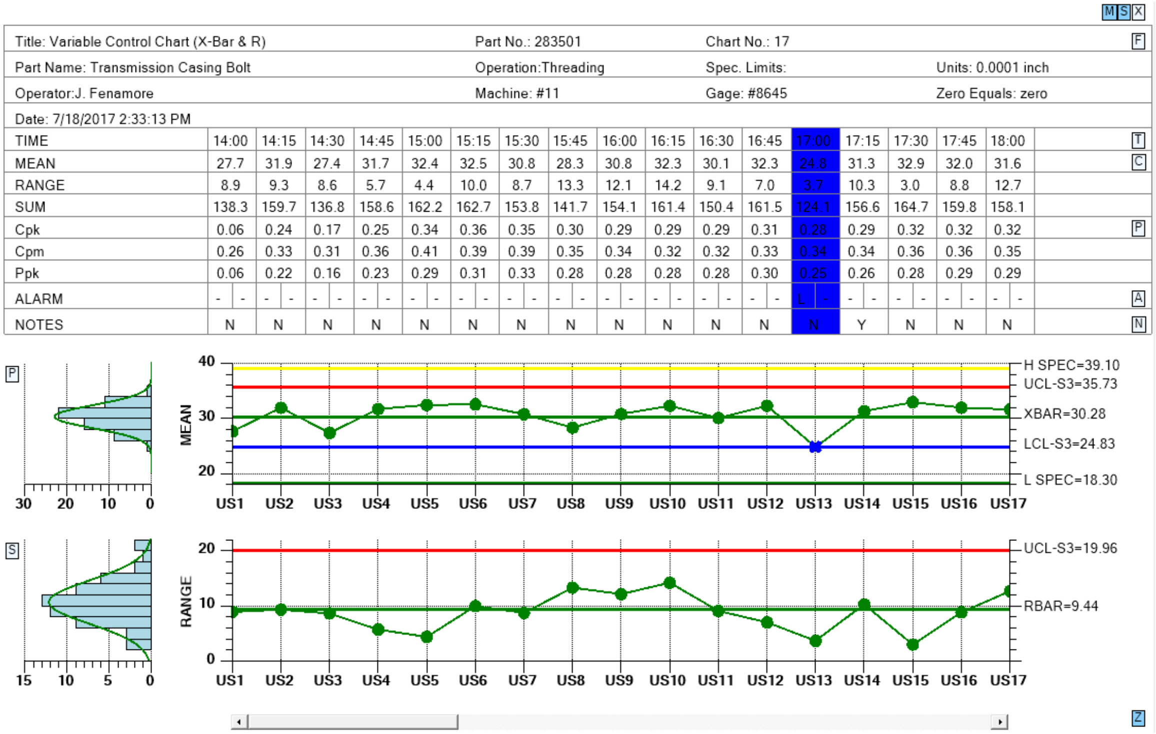
The default display display for all chart types uses the scrollbar. The zoom option is seen as a small button with the character 'Z' in the lower right corner of the display. You enter/exit the zoom mode by clicking on that button. If you do not want the button to show at all, effectively making the zoom option inaccessible to the end user, set EnableZoomToggles property false.
.

```
            this.EnableZoomToggles = false;
```

All of the examples in the NewRev3Features example show this feature.

Collapsible Items

Like the Zoom option described in the previous section, there are also UI buttons which can toggle on and off rows in the table, and the Primary and Secondary charts. Like the Zoom button, these UI buttons can be hidden from the end user if you don't want them to show. See the end of this section for examples. On the top and right of the table buttons will selectively collapse/restore rows of the table, freeing up more space for charts. Buttons to the left and bottom of the Primary and Secondary charts also permit the user to collapse/restore either of those charts, allowing the remaining chart to display in all of the remaining space.



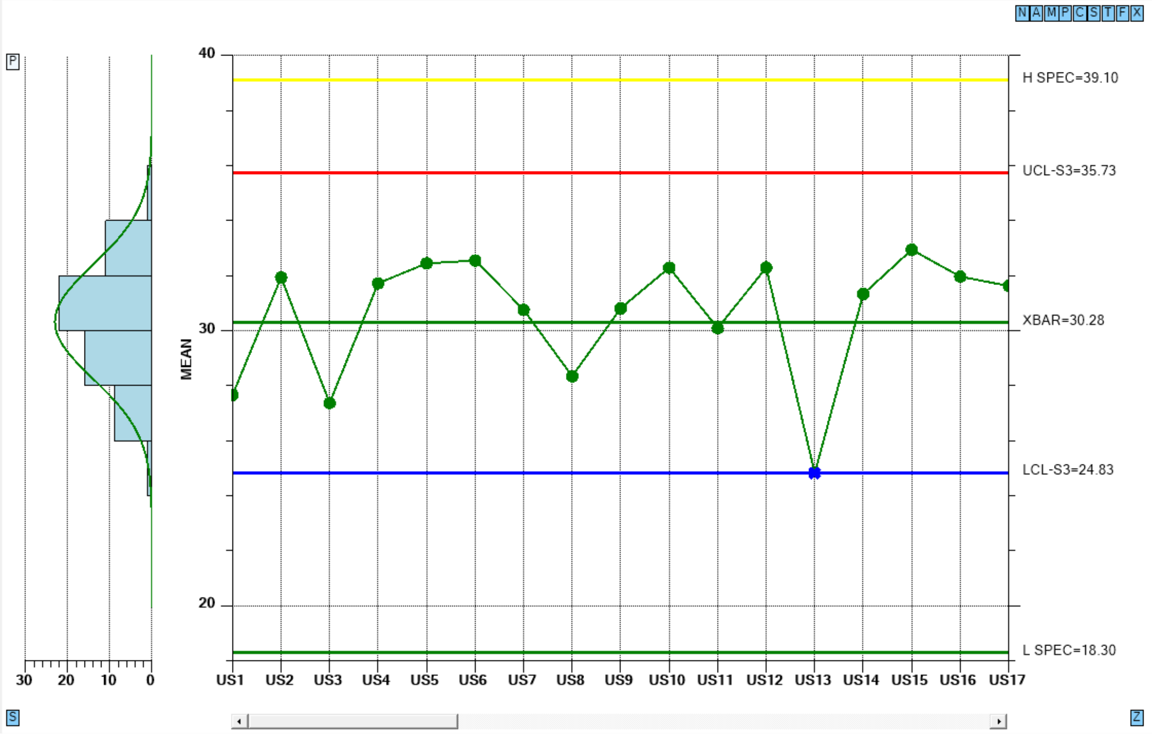
Title: Variable Control Chart (X-Bar & R)	Part No.: 283501	Chart No.: 17	
Part Name: Transmission Casing Bolt	Operation:Threading	Spec. Limits:	Units: 0.0001 inch
Operator:J. Fenamore	Machine: #11	Gage: #8645	Zero Equals: zero
Date: 7/18/2017 2:33:13 PM			

TIME	14:00	14:15	14:30	14:45	15:00	15:15	15:30	15:45	16:00	16:15	16:30	16:45	17:00	17:15	17:30	17:45	18:00
MEAN	27.7	31.9	27.4	31.7	32.4	32.5	30.8	28.3	30.8	32.3	30.1	32.3	24.8	31.3	32.9	32.0	31.6
RANGE	8.9	9.3	8.6	5.7	4.4	10.0	8.7	13.3	12.1	14.2	9.1	7.0	3.7	10.3	3.0	8.8	12.7
SUM	138.3	159.7	136.8	158.6	162.2	162.7	153.8	141.7	154.1	161.4	150.4	161.5	124.1	156.6	164.7	159.8	158.1
Cpk	0.06	0.24	0.17	0.25	0.34	0.36	0.35	0.30	0.29	0.29	0.29	0.31	0.28	0.29	0.32	0.32	0.32
Cpm	0.26	0.33	0.31	0.36	0.41	0.39	0.39	0.35	0.34	0.32	0.32	0.33	0.34	0.34	0.36	0.36	0.35
Ppk	0.06	0.22	0.16	0.23	0.29	0.31	0.33	0.28	0.28	0.28	0.28	0.30	0.25	0.26	0.28	0.29	0.29
ALARM	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	L -	- -	- -	- -	- -
NOTES	N	N	N	N	N	N	N	N	N	N	N	N	N	Y	N	N	N

M
S
X
F
T
C
P
A
N
P
40
30
20
MEAN
30 20 10 0
H SPEC=39.10
UCL-S3=35.73
XBAR=30.28
LCL-S3=24.83
L SPEC=18.30
US1 US2 US3 US4 US5 US6 US7 US8 US9 US10 US11 US12 US13 US14 US15 US16 US17
S
20
10
0
RANGE
15 10 5 0
UCL-S3=19.96
RBAR=9.44
US1 US2 US3 US4 US5 US6 US7 US8 US9 US10 US11 US12 US13 US14 US15 US16 US17
Z

Buttons on the top, right and left of the display permit the user to selectively collapse portions of the chart.

Click on the N, A, M, P C and S buttons and shrink the table to following size.



In the example above, all of the chart options except for the Primary chart, have been toggled off using the buttons.

The buttons at the right of the table (and at the top right if toggled off) use the following ID's.

X	Turn on/off all table items at once
F	Turn on/off form data
T	Turn on/off sample interval time stamp data
S	Turn on/off sample value data
C	Turn on/off calculated value (mean, range, sum, etc.) data
P	Turn on/off process capability data
M	Turn on/off number of samples data
A	Turn on/off alarm data
N	Turn on/off notes data
Z	Bottom right - Turn on/off zoom control – the only button not affected by the X button above

The buttons at the left of the primary and secondary charts use the following ID's.

P	Turn on/off the Primary chart
S	Turn on/off the Secondary chart

If you do not want the buttons to show at all, effectively making these UI options inaccessible to the end user, set these properties false.

.

Hide the chart buttons on the left.

```
this.EnableChartToggles = false;
```

Hide the table buttons on the right.

```
this.EnableTableRowToggles = false;
```

You can hide ALL of the UI buttons (zoom, chart, and table) using the property EnableDisplayOptionToggles. This is what you use if you do not want the end user to have any control over the display of the table and chart.

```
this. EnableDisplayOptionToggles = false;
```

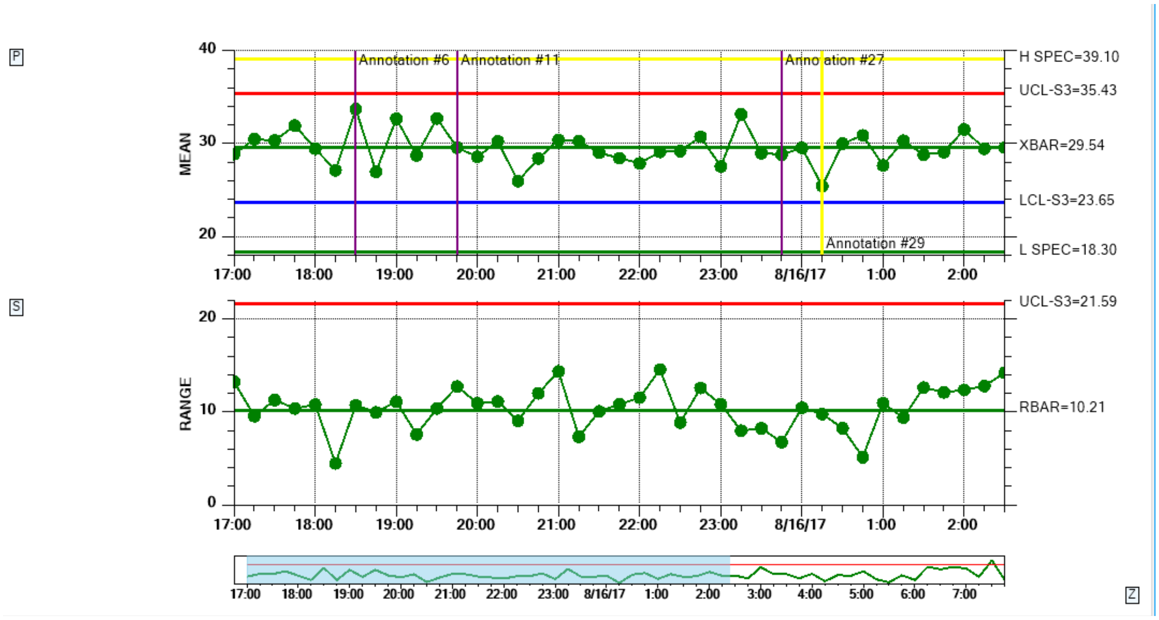
If you want to selectively enable options, first set EnableDisplayOptionToggles true. The other button display enables won't work unless this option is true. Then disable the options you do not want to show. In the example below, only the zoom option is left visible.

```
this.EnableDisplayOptionToggles = true;
```

```
            this.EnableTableRowToggles = false;
            this.EnableChartToggles = false;
            this.EnableZoomToggle = true;
```

Enhanced Annotations

The chart annotations have been enhanced with a vertical line with accompanying text using programmer specified justification.



The enhanced chart annotations include a vertical line to mark the data point, and many justification options (top, middle, bottom, left, right and center).

The new version of AddAnnotation is just an override of the original, with added parameters to specify the vertical line attributes and the text justification.

AddAnnotation

Add an annotation to a data point in the specified SPC chart.

```
public int AddAnnotation(int chart, int datapointindex, String text, int just,
ChartAttribute attrib)
```

where:

chart Specifies whether the annotation is added to the primary, or secondary chart. Use one of the SPCChartObjects constants: SPCChartObjects.PRIMARY_CHART or SPCChartObjects.SECONDARY_CHART.

datapointindex The index of the data point the annotation is for.

text A string string representing the annotation.

just The justification for the text to the x-position of the annotation. Use one of the annotation justification constants: ANNOTATION_UPPER_RIGHT, ANNOTATION_UPPER_LEFT, ANNOTATION_LOWER_RIGHT, ANNOTATION_LOWER_LEFT, ANNOTATION_UPPER_CENTER, ANNOTATION_LOWER_CENTER, ANNOTATION_DATAPOINT_RIGHT, ANNOTATION_DATAPOINT_LEFT

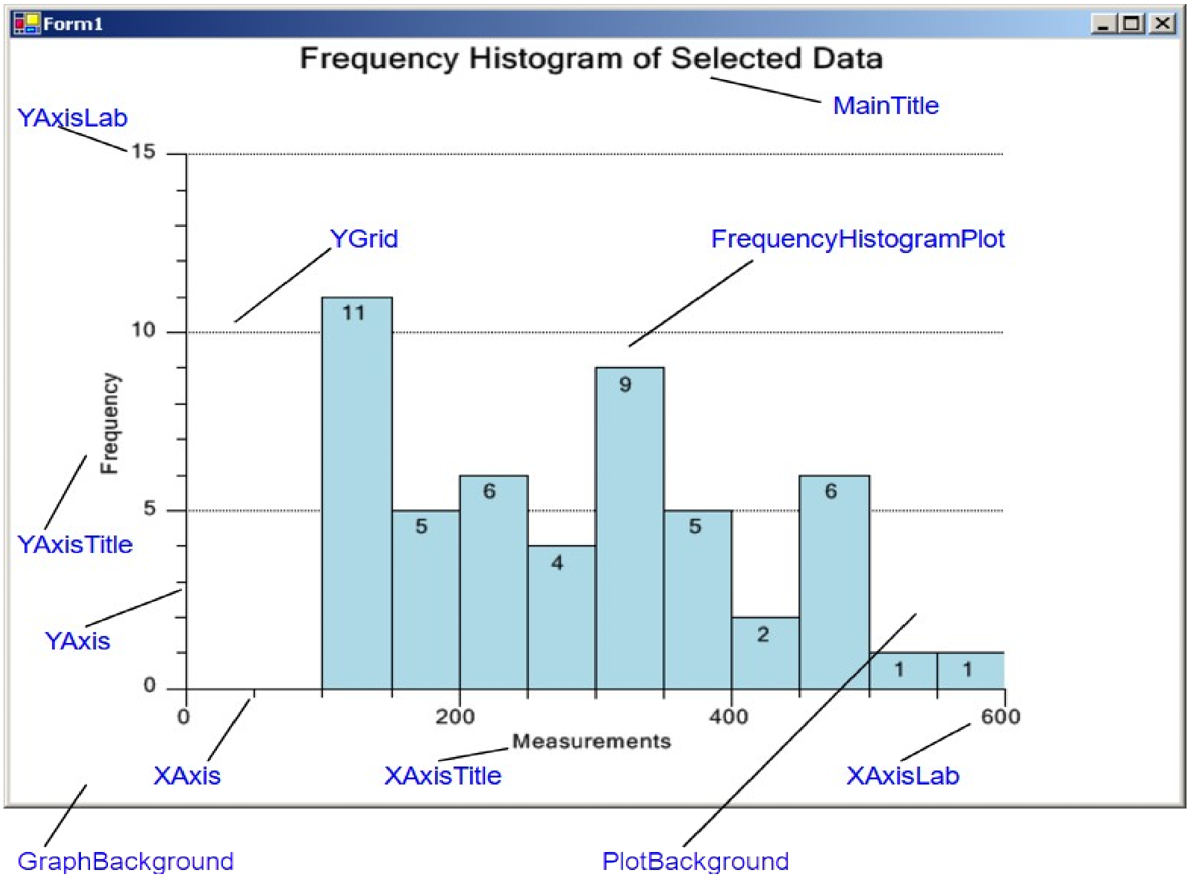
attrib A the attribute of the vertical line.

You call the AddAnnotation method immediately after the AddNewSampleRecord method call. For Example:

```
          // Add the new sample subgroup to the chart
this.ChartData.AddNewSampleRecord(timestamp, samples, notesstring);

int index = this.ChartData.CurrentNumberRecords - 1;
string annotstring = "Annotation #" + index.ToString();
ChartAttribute lineattrib = new ChartAttribute(Color.Purple, 2,
         DashStyle.Solid);
// Adjust justification to minimize overlap of adjacent annotations
int annotjust = SPCAnnotation.ANNOTATION_UPPER_LEFT;

this.AddAnnotation(SPCChartObjects.PRIMARY_CHART, index,
        annotstring, annotjust, lineattrib);
```

See the NewRev3Features.Annotations demo for an example.

SPC Chart Histograms



Viewing frequency histograms of both the variation in the primary variable (Mean, Median, count, or actual value), and the secondary variable (Range, Sigma or Moving Range) side-by-side with the SPC control chart makes it even easier to find out whether the variations are the result of natural variations or the result of some systemic change in the process. You can turn on integrated frequency histograms for either chart using the **PrimaryChart.DisplayFrequencyHistogram** and **SecondaryChart.DisplayFrequencyHistogram** properties of the chart.

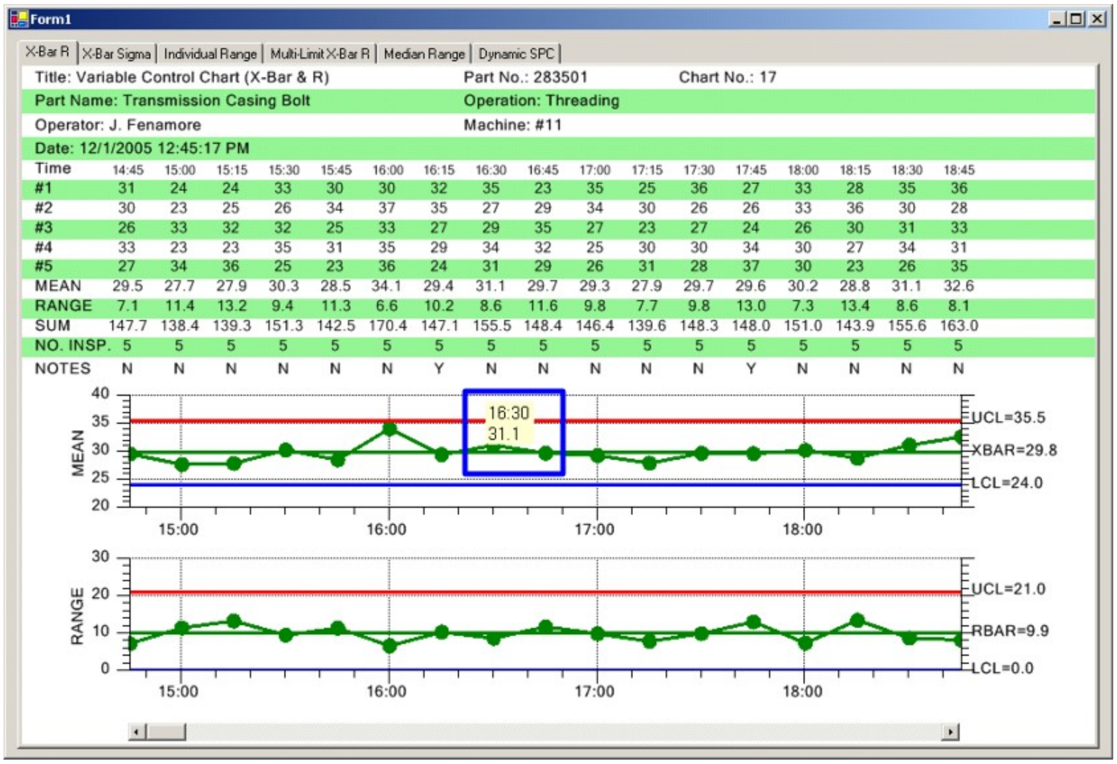
[C#]
```
//  frequency histogram for both charts
this.PrimaryChart.DisplayFrequencyHistogram = true;
this.SecondaryChart.DisplayFrequencyHistogram = true;
```

[VB]
```
`  frequency histogram for both charts
Me.PrimaryChart.DisplayFrequencyHistogram = True
Me.SecondaryChart.DisplayFrequencyHistogram = True
```

SPC Chart Data and Notes Tooltips

You can invoke two types of tooltips using the mouse. The first is a data tooltip. When you hold the mouse button down over one of the data points, in the primary or secondary chart, the x and y values for that data point display in a popup tooltip.

Data Tooltip



Title: Variable Control Chart (X-Bar & R) Part No.: 283501 Chart No.: 17
Part Name: Transmission Casing Bolt Operation: Threading
Operator: J. Fenamore Machine: #11
Date: 12/1/2005 12:45:17 PM

Time	14:45	15:00	15:15	15:30	15:45	16:00	16:15	16:30	16:45	17:00	17:15	17:30	17:45	18:00	18:15	18:30	18:45
#1	31	24	24	33	30	30	32	35	23	35	25	36	27	33	28	35	36
#2	30	23	25	26	34	37	35	27	29	34	30	26	26	33	36	30	28
#3	26	33	32	32	25	33	27	29	35	27	23	27	24	26	30	31	33
#4	33	23	23	35	31	35	29	34	32	25	30	30	34	30	27	34	31
#5	27	34	36	25	23	36	24	31	29	26	31	28	37	30	23	26	35
MEAN	29.5	27.7	27.9	30.3	28.5	34.1	29.4	31.1	29.7	29.3	27.9	29.7	29.6	30.2	28.8	31.1	32.6
RANGE	7.1	11.4	13.2	9.4	11.3	6.6	10.2	8.6	11.6	9.8	7.7	9.8	13.0	7.3	13.4	8.6	8.1
SUM	147.7	138.4	139.3	151.3	142.5	170.4	147.1	155.5	148.4	146.4	139.6	148.3	148.0	151.0	143.9	155.6	163.0
NO. INSP.	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
NOTES	N	N	N	N	N	N	Y	N	N	N	N	N	Y	N	N	N	N

In the default mode, the data tooltip displays the x,y value of the data point nearest the mouse click. If the x-axis is a time axis then the x-value is displayed as a time stamp; otherwise, it is displayed as a simple numeric value, as is the y-value. You can optionally display subgroup information (sample values, calculated values, process capability values and notes) in the data tooltip window, under the x,y value, using enable flags in the primary charts tooltip property.

Extracted from the TimeVariableControlCharts.XBarRChart example.

[C#]

```
this.PrimaryChart.Datatooltip.EnableCategoryValues = true;
this.PrimaryChart.Datatooltip.EnableProcessCapabilityValues = true;
this.PrimaryChart.Datatooltip.EnableCalculatedValues = true;
this.PrimaryChart.Datatooltip.EnableNotesString = true;
```

[VB]

```
Me.PrimaryChart.Datatooltip.EnableCategoryValues = True
Me.PrimaryChart.Datatooltip.EnableProcessCapabilityValues = True
Me.PrimaryChart.Datatooltip.EnableCalculatedValues = True
Me.PrimaryChart.Datatooltip.EnableNotesString = True
```

where

The following properties enable sections of the chart header and table:
PrimaryChart.Datatooltip.EnableCategoryValues
PrimaryChart.Datatooltip.EnableProcessCapabilityValues
PrimaryChart.Datatooltip.EnableCalculatedValues
PrimaryChart.Datatooltip.EnableNotesStrings

Display the category (subgroup sample values) in the data tooltip.

```
PrimaryChart.Datatooltip.EnableCategoryValues = true
```

Display the calculated values used in the chart (Mean, range and sum for an Mean-Range chart).

```
PrimaryChart.Datatooltip.EnableCalculatedValues = true
```

Display the process capability (Cp, Cpl, Cpu, Cpk, Cpm, Pp, Ppl, Ppu and Ppk) statistics currently being calculated for the chart.

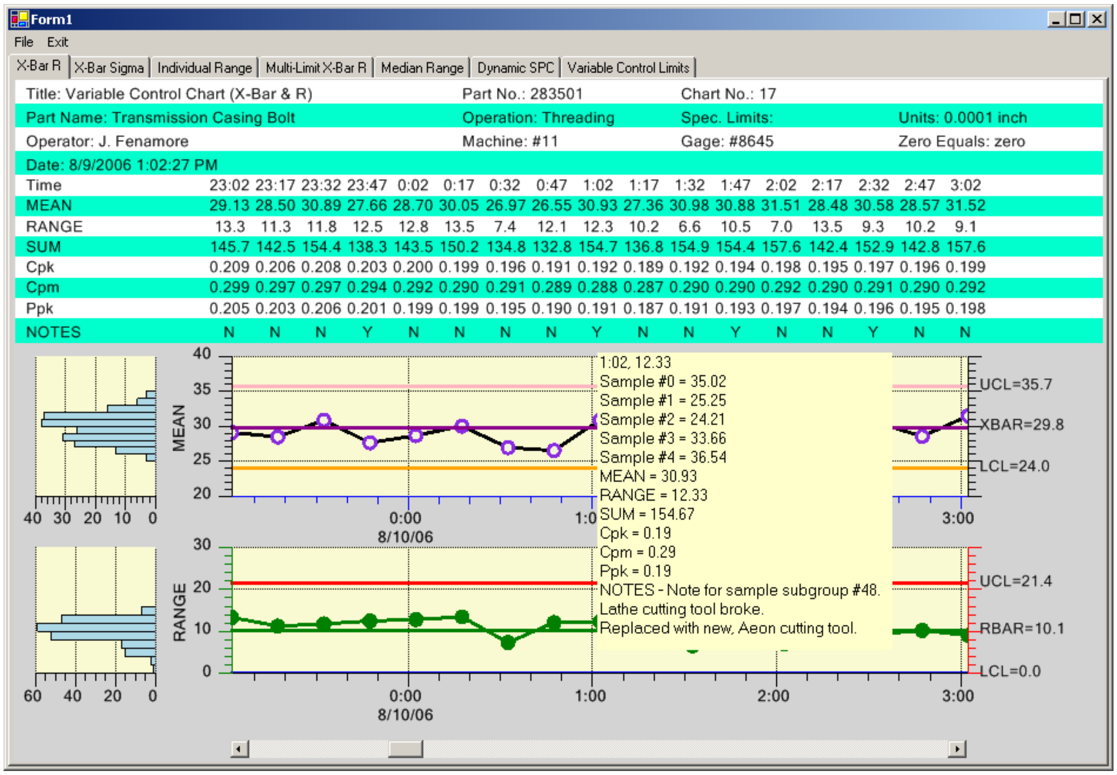
```
PrimaryChart.Datatooltip.EnableProcessCapabilityValues = true
```

Display the current notes string for the sample subgroup.

```
PrimaryChart.Datatooltip.EnableNotesString = true
```

The variable control chart below displays a tooltip with all of the enable options above set true.

Data Tooltip with optional display items



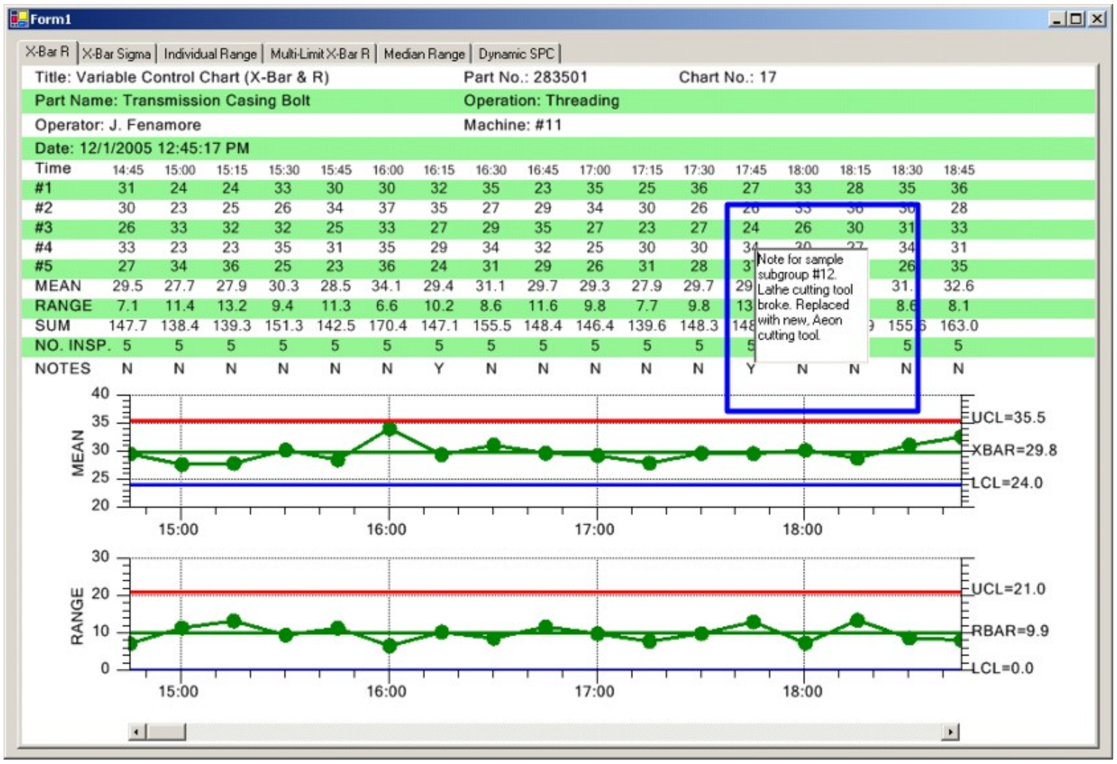
Title: Variable Control Chart (X-Bar & R)	Part No.: 283501	Chart No.: 17	
Part Name: Transmission Casing Bolt	Operation: Threading	Spec. Limits:	Units: 0.0001 inch
Operator: J. Fenamore	Machine: #11	Gage: #8645	Zero Equals: zero
Date: 8/9/2006 1:02:27 PM			

Time	23:02	23:17	23:32	23:47	0:02	0:17	0:32	0:47	1:02	1:17	1:32	1:47	2:02	2:17	2:32	2:47	3:02
MEAN	29.13	28.50	30.89	27.66	28.70	30.05	26.97	26.55	30.93	27.36	30.98	30.88	31.51	28.48	30.58	28.57	31.52
RANGE	13.3	11.3	11.8	12.5	12.8	13.5	7.4	12.1	12.3	10.2	6.6	10.5	7.0	13.5	9.3	10.2	9.1
SUM	145.7	142.5	154.4	138.3	143.5	150.2	134.8	132.8	154.7	136.8	154.9	154.4	157.6	142.4	152.9	142.8	157.6
Cpk	0.209	0.206	0.208	0.203	0.200	0.199	0.196	0.191	0.192	0.189	0.192	0.194	0.198	0.195	0.197	0.196	0.199
Cpm	0.299	0.297	0.297	0.294	0.292	0.290	0.291	0.289	0.288	0.287	0.290	0.290	0.292	0.290	0.291	0.290	0.292
Ppk	0.205	0.203	0.206	0.201	0.199	0.199	0.195	0.190	0.191	0.187	0.191	0.193	0.197	0.194	0.196	0.195	0.198
NOTES	N	N	N	Y	N	N	N	N	Y	N	N	Y	N	N	Y	N	N

1:02, 12.33
Sample #0 = 35.02
Sample #1 = 25.25
Sample #2 = 24.21
Sample #3 = 33.66
Sample #4 = 36.54
MEAN = 30.93
RANGE = 12.33
SUM = 154.67
Cpk = 0.19
Cpm = 0.29
Ppk = 0.19
NOTES - Note for sample subgroup #48.
Lathe cutting tool broke.
Replaced with new, Aeon cutting tool.

If you are displaying the Notes line in the table portion of the chart, the Notes entry for a sample subgroup will display "Y" if a note was recorded for that sample subgroup, or "N" if no note was recorded. Notes are recorded using one of the **ChartData.AddNewSampleRecord** overrides that include a notes parameter, or by using the **ChartData.SetNotes**, or **ChartData.AppendNotes** methods. See the section *Updating Chart Data*. If you click on a "Y" in the Notes row for a sample subgroup, the complete text of the note for that sample subgroup will display in a RichTextBox, immediately above the "Y". You can actually edit the notes in the RichTextBox.

Notes Tooltip



Title: Variable Control Chart (X-Bar & R) Part No.: 283501 Chart No.: 17

Part Name: Transmission Casing Bolt Operation: Threading

Operator: J. Fenamore Machine: #11

Date: 12/1/2005 12:45:17 PM

Time	14:45	15:00	15:15	15:30	15:45	16:00	16:15	16:30	16:45	17:00	17:15	17:30	17:45	18:00	18:15	18:30	18:45
#1	31	24	24	33	30	30	32	35	23	35	25	36	27	33	28	35	36
#2	30	23	25	26	34	37	35	27	29	34	30	26	26	33	36	30	28
#3	26	33	32	32	25	33	27	29	35	27	23	27	24	26	30	31	33
#4	33	23	23	35	31	35	29	34	32	25	30	30	34	30	27	34	31
#5	27	34	36	25	23	36	24	31	29	26	31	28	31	[illegible]	[illegible]	26	35
MEAN	29.5	27.7	27.9	30.3	28.5	34.1	29.4	31.1	29.7	29.3	27.9	29.7	29	[illegible]	[illegible]	31.	32.6
RANGE	7.1	11.4	13.2	9.4	11.3	6.6	10.2	8.6	11.6	9.8	7.7	9.8	13	[illegible]	[illegible]	8.6	8.1
SUM	147.7	138.4	139.3	151.3	142.5	170.4	147.1	155.5	148.4	146.4	139.6	148.3	148	[illegible]	9	155.6	163.0
NO. INSP.	5	5	5	5	5	5	5	5	5	5	5	5	5	[illegible]	[illegible]	5	5
NOTES	N	N	N	N	N	N	Y	N	N	N	N	N	Y	N	N	N	N

Note for sample subgroup #12. Lathe cutting tool broke. Replaced with new, Aeon cutting tool.

[C#]

```
private void SimulateData()
{  String notesstring = "";
    for (int i=0; i < 200; i++)
    {
        .
        .
        .
        this.ChartData.AddNewSampleRecord(timestamp, samples, notesstring);
        // increment simulated time by timeincrementminutes minutes
        startTime.Add(ChartObj.MINUTE, timeincrementminutes);
    }
}
```

[VB]

```
Private Sub SimulateData()
   Dim notesstring As [String] = ""
   Dim i As Integer
   For i = 0 To 199
        .
        .
        .
      Me.ChartData.AddNewSampleRecord(timestamp, samples, notesstring)
      ' increment simulated time by timeincrementminutes minutes
      startTime.Add(ChartObj.MINUTE, timeincrementminutes)
   Next i
End Sub 'SimulateData
```

Both kinds of tooltips are on by default. Turn the tooltips on or off in the program using the **EnableDataToolTip** and **EnableNotesToolTip** flags.

[C#]

```
// Enable data and notes tooltips
this.EnableDataToolTip = true;
this.EnableNotesToolTip = true;
```

[VB]

```
' Enable data and notes tooltips
Me.EnableDataToolTip = True
Me.EnableNotesToolTip = True
```

The notes tooltip has an additional option. In order to make the notes tooltip “editable”, the tooltip, which is .Net RichEditBox, displays on the first click, and goes away on the second click. You can click inside the RichTextBox and not worry the tooltip suddenly disappearing. The notes tooltip works this way by default. If you wish to explicitly set it, or change it so that the tooltip only displays while the mouse button is held down, as the data tooltips do, set the **ChartData.NotesToolTips.ToolTipMode** property to **NotesToolTip.MOUSEDOWN_TOOLTIP**, as in the example below.

[C#]

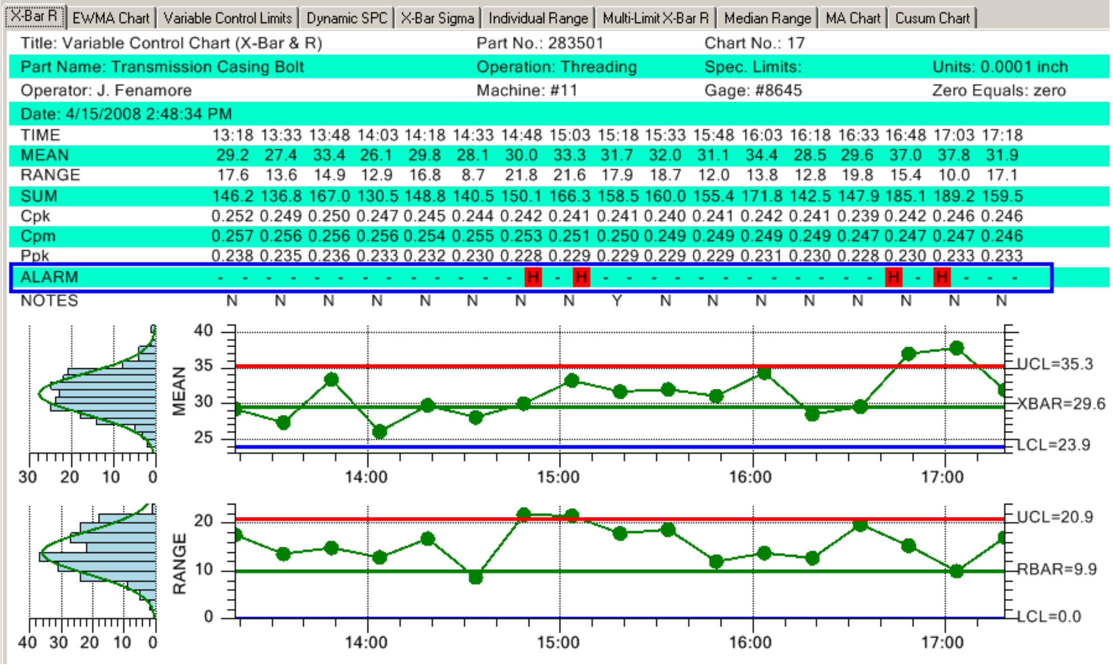
```
// Enable data and notes tooltips
this.EnableDataToolTip = true;
this.EnableNotesToolTip = true;

this.ChartData.NotesToolTips.ButtonMask = MouseButtons.Right;
// default is MOUSETOGGLE_TOOLTIP
this.ChartData.NotesToolTips.ToolTipMode= NotesToolTip.MOUSEDOWN_TOOLTIP;
```

[VB]

```
' Enable data and notes tooltips
Me.EnableDataToolTip = True
Me.EnableNotesToolTip = True

Me.ChartData.NotesToolTips.ButtonMask = MouseButtons.Right
' default is MOUSETOGGLE_TOOLTIP
Me.ChartData.NotesToolTips.ToolTipMode = NotesToolTip.MOUSEDOWN_TOOLTIP
```

Enable Alarm Highlighting

EnableAlarmStatusValues



Title: Variable Control Chart (X-Bar & R) Part No.: 283501 Chart No.: 17

Part Name: Transmission Casing Bolt Operation: Threading Spec. Limits: Units: 0.0001 inch

Operator: J. Fenamore Machine: #11 Gage: #8645 Zero Equals: zero

Date: 4/15/2008 2:48:34 PM

TIME	13:18	13:33	13:48	14:03	14:18	14:33	14:48	15:03	15:18	15:33	15:48	16:03	16:18	16:33	16:48	17:03	17:18
MEAN	29.2	27.4	33.4	26.1	29.8	28.1	30.0	33.3	31.7	32.0	31.1	34.4	28.5	29.6	37.0	37.8	31.9
RANGE	17.6	13.6	14.9	12.9	16.8	8.7	21.8	21.6	17.9	18.7	12.0	13.8	12.8	19.8	15.4	10.0	17.1
SUM	146.2	136.8	167.0	130.5	148.8	140.5	150.1	166.3	158.5	160.0	155.4	171.8	142.5	147.9	185.1	189.2	159.5
Cpk	0.252	0.249	0.250	0.247	0.245	0.244	0.242	0.241	0.241	0.240	0.241	0.242	0.241	0.239	0.242	0.246	0.246
Cpm	0.257	0.256	0.256	0.256	0.254	0.255	0.253	0.251	0.250	0.249	0.249	0.249	0.249	0.247	0.247	0.247	0.246
Ppk	0.238	0.235	0.236	0.233	0.232	0.230	0.228	0.229	0.229	0.229	0.229	0.231	0.230	0.228	0.230	0.233	0.233
ALARM	- -	- -	- -	- -	- -	- -	- H	- H	- -	- -	- -	- -	- -	- -	H -	H -	- -
NOTES	N	N	N	N	N	N	N	N	Y	N	N	N	N	N	N	N	N

There are several alarm highlighting options you can turn on and off. The alarm status line above is turned on/off using the EnableAlarmStatusValues property. We have set it on by default, so you will have to turn it off if you don't want it. Each sample interval has two small boxes that are labeled using one of three different characters. An "H" signifies a high alarm, a "L" signifies a low alarm, and a "-" signifies that there is no alarm.

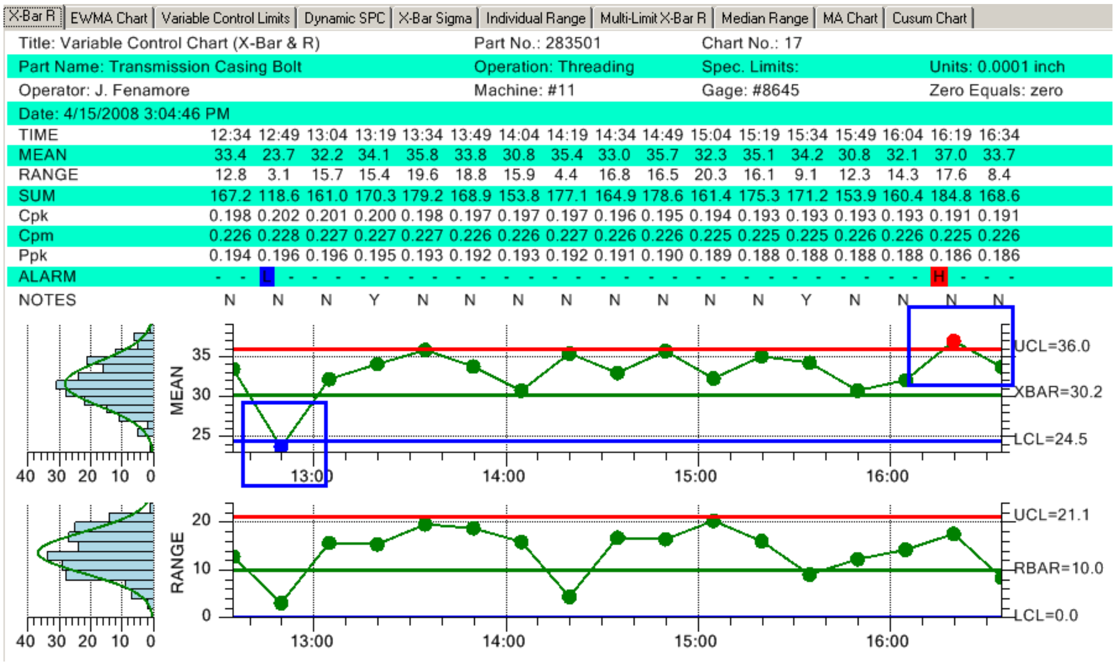
[C#]

```
// Alarm status line
this.EnableAlarmStatusValues = false;
```

[VB]

```
'Alarm status line
Me.EnableAlarmStatusValues = False
```

ChartAlarmEmphasisMode



Title: Variable Control Chart (X-Bar & R) Part No.: 283501 Chart No.: 17

Part Name: Transmission Casing Bolt Operation: Threading Spec. Limits: Units: 0.0001 inch

Operator: J. Fenamore Machine: #11 Gage: #8645 Zero Equals: zero

Date: 4/15/2008 3:04:46 PM

TIME	12:34	12:49	13:04	13:19	13:34	13:49	14:04	14:19	14:34	14:49	15:04	15:19	15:34	15:49	16:04	16:19	16:34
MEAN	33.4	23.7	32.2	34.1	35.8	33.8	30.8	35.4	33.0	35.7	32.3	35.1	34.2	30.8	32.1	37.0	33.7
RANGE	12.8	3.1	15.7	15.4	19.6	18.8	15.9	4.4	16.8	16.5	20.3	16.1	9.1	12.3	14.3	17.6	8.4
SUM	167.2	118.6	161.0	170.3	179.2	168.9	153.8	177.1	164.9	178.6	161.4	175.3	171.2	153.9	160.4	184.8	168.6
Cpk	0.198	0.202	0.201	0.200	0.198	0.197	0.197	0.197	0.196	0.195	0.194	0.193	0.193	0.193	0.193	0.191	0.191
Cpm	0.226	0.228	0.227	0.227	0.227	0.226	0.226	0.227	0.226	0.226	0.225	0.225	0.225	0.226	0.226	0.225	0.226
Ppk	0.194	0.196	0.196	0.195	0.193	0.192	0.193	0.192	0.191	0.190	0.189	0.188	0.188	0.188	0.188	0.186	0.186
ALARM	- -	L	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	H	- -
NOTES	N	N	N	Y	N	N	N	N	N	N	N	N	Y	N	N	N	N

[C#]

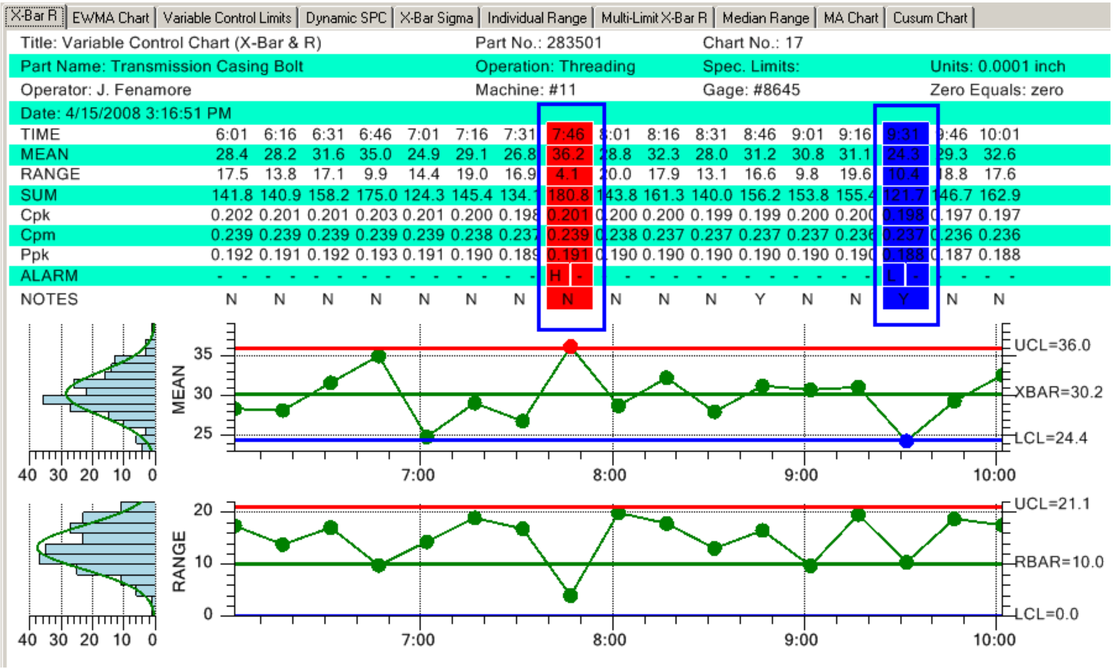
```
// Chart alarm emphasis mode
this.ChartAlarmEmphasisMode = SPCChartBase.ALARM_HIGHLIGHT_SYMBOL;
```

[VB]

```
' Chart alarm emphasis mode
Me.ChartAlarmEmphasisMode = SPCChartBase.ALARM_HIGHLIGHT_SYMBOL
```

The scatter plot symbol used to plot a data point in the primary and secondary charts is normally a fixed color circle. If you turn on the alarm highlighting for chart symbols the symbol color for a sample interval that is in an alarm condition will change to reflect the color of the associated alarm line. In the example above, a low alarm (blue circle) occurs at the beginning of the chart and a high alarm (red circle) occurs at the end of the chart. Alarm symbol highlighting is turned on by default. To turn it off use the SPCChartBase.ALARM_NO_HIGHLIGHT_SYMBOL constants.

TableAlarmEmphasisMode -



Title: Variable Control Chart (X-Bar & R) Part No.: 283501 Chart No.: 17
Part Name: Transmission Casing Bolt Operation: Threading Spec. Limits: Units: 0.0001 inch
Operator: J. Fenamore Machine: #11 Gage: #8645 Zero Equals: zero
Date: 4/15/2008 3:16:51 PM

TIME	6:01	6:16	6:31	6:46	7:01	7:16	7:31	7:46	8:01	8:16	8:31	8:46	9:01	9:16	9:31	9:46	10:01
MEAN	28.4	28.2	31.6	35.0	24.9	29.1	26.8	36.2	28.8	32.3	28.0	31.2	30.8	31.1	24.3	29.3	32.6
RANGE	17.5	13.8	17.1	9.9	14.4	19.0	16.9	4.1	20.0	17.9	13.1	16.6	9.8	19.6	10.4	18.8	17.6
SUM	141.8	140.9	158.2	175.0	124.3	145.4	134.1	180.8	143.8	161.3	140.0	156.2	153.8	155.4	121.7	146.7	162.9
Cpk	0.202	0.201	0.201	0.203	0.201	0.200	0.198	0.201	0.200	0.200	0.199	0.199	0.200	0.200	0.198	0.197	0.197
Cpm	0.239	0.239	0.239	0.239	0.239	0.238	0.237	0.239	0.238	0.237	0.237	0.237	0.237	0.236	0.237	0.236	0.236
Ppk	0.192	0.191	0.192	0.193	0.191	0.190	0.189	0.191	0.190	0.190	0.190	0.190	0.190	0.190	0.188	0.187	0.188
ALARM	- -	- -	- -	- -	- -	- -	- -	H -	- -	- -	- -	- -	- -	- -	L -	- -	- -
NOTES	N	N	N	N	N	N	N	N	N	N	N	Y	N	N	Y	N	N

C#]

```
// Table alarm emphasis mode
this.TableAlarmEmphasisMode = SPCChartBase.ALARM_HIGHLIGHT_BAR;
```

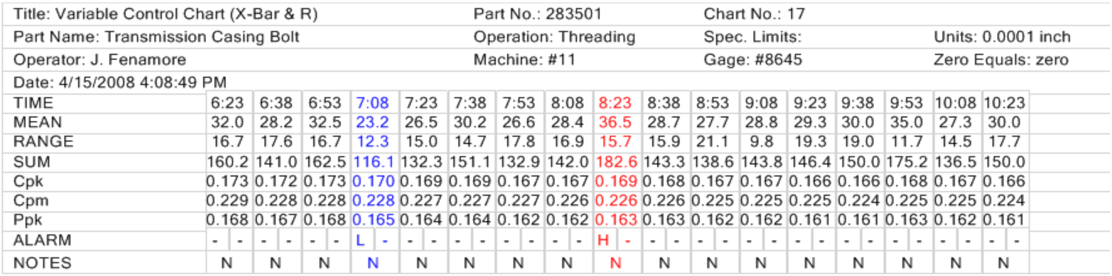
[VB]

```
' Table alarm emphasis mode
Me.TableAlarmEmphasisMode = SPCChartBase.ALARM_HIGHLIGHT_BAR
```

The entire column of the data table can be highlighted when an alarm occurs. There are four modes associated with this property:

ALARM_HIGHLIGHT_NONE	No alarm highlight
ALARM_HIGHLIGHT_TEXT	Text alarm highlight
ALARM_HIGHLIGHT_OUTLINE	Outline alarm highlight
ALARM_HIGHLIGHT_BAR	Bar alarm highlight

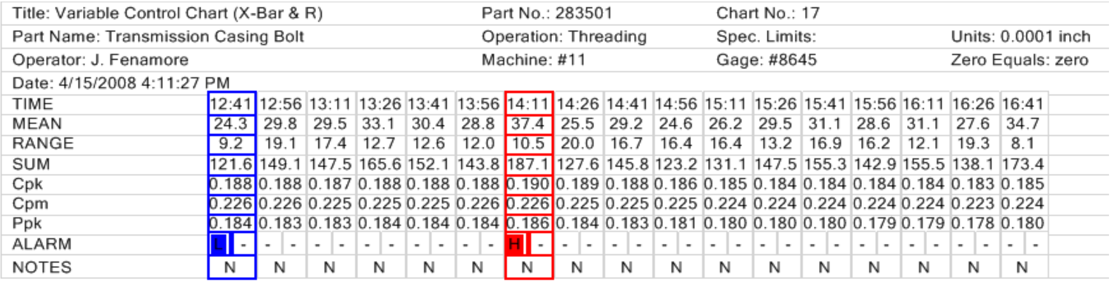
The example above uses the ALARM_HIGHLIGHT_BAR mode.



Title: Variable Control Chart (X-Bar & R) Part No.: 283501 Chart No.: 17
Part Name: Transmission Casing Bolt Operation: Threading Spec. Limits: Units: 0.0001 inch
Operator: J. Fenamore Machine: #11 Gage: #8645 Zero Equals: zero
Date: 4/15/2008 4:08:49 PM

TIME	6:23	6:38	6:53	7:08	7:23	7:38	7:53	8:08	8:23	8:38	8:53	9:08	9:23	9:38	9:53	10:08	10:23
MEAN	32.0	28.2	32.5	23.2	26.5	30.2	26.6	28.4	36.5	28.7	27.7	28.8	29.3	30.0	35.0	27.3	30.0
RANGE	16.7	17.6	16.7	12.3	15.0	14.7	17.8	16.9	15.7	15.9	21.1	9.8	19.3	19.0	11.7	14.5	17.7
SUM	160.2	141.0	162.5	116.1	132.3	151.1	132.9	142.0	182.6	143.3	138.6	143.8	146.4	150.0	175.2	136.5	150.0
Cpk	0.173	0.172	0.173	0.170	0.169	0.169	0.167	0.167	0.169	0.168	0.167	0.167	0.166	0.166	0.168	0.167	0.166
Cpm	0.229	0.228	0.228	0.228	0.227	0.227	0.227	0.226	0.226	0.226	0.225	0.225	0.225	0.224	0.225	0.225	0.224
Ppk	0.168	0.167	0.168	0.165	0.164	0.164	0.162	0.162	0.163	0.163	0.162	0.162	0.161	0.161	0.163	0.162	0.161
ALARM	- -	- -	- -	L -	- -	- -	- -	- -	H -	- -	- -	- -	- -	- -	- -	- -	- -
NOTES	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N

The example above uses the ALARM_HIGHLIGHT_TEXT mode



Title: Variable Control Chart (X-Bar & R) Part No.: 283501 Chart No.: 17

Part Name: Transmission Casing Bolt Operation: Threading Spec. Limits: Units: 0.0001 inch

Operator: J. Fenamore Machine: #11 Gage: #8645 Zero Equals: zero

Date: 4/15/2008 4:11:27 PM

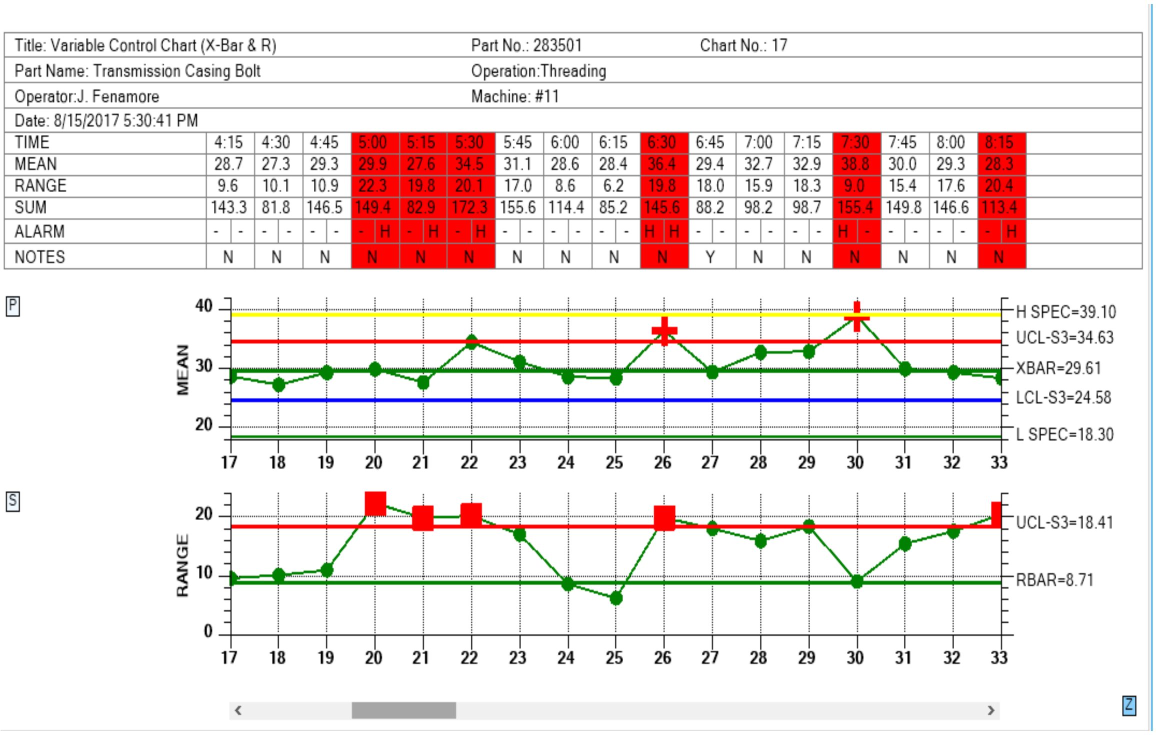
TIME	12:41	12:56	13:11	13:26	13:41	13:56	14:11	14:26	14:41	14:56	15:11	15:26	15:41	15:56	16:11	16:26	16:41
MEAN	24.3	29.8	29.5	33.1	30.4	28.8	37.4	25.5	29.2	24.6	26.2	29.5	31.1	28.6	31.1	27.6	34.7
RANGE	9.2	19.1	17.4	12.7	12.6	12.0	10.5	20.0	16.7	16.4	16.4	13.2	16.9	16.2	12.1	19.3	8.1
SUM	121.6	149.1	147.5	165.6	152.1	143.8	187.1	127.6	145.8	123.2	131.1	147.5	155.3	142.9	155.5	138.1	173.4
Cpk	0.188	0.188	0.187	0.188	0.188	0.188	0.190	0.189	0.188	0.186	0.185	0.184	0.184	0.184	0.184	0.183	0.185
Cpm	0.226	0.226	0.225	0.225	0.225	0.226	0.226	0.225	0.225	0.225	0.224	0.224	0.224	0.224	0.224	0.223	0.224
Ppk	0.184	0.183	0.183	0.184	0.184	0.184	0.186	0.184	0.183	0.181	0.180	0.180	0.180	0.179	0.179	0.178	0.180
ALARM	L -	- -	- -	- -	- -	- -	H -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -
NOTES	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N

The example above uses the ALARM_HIGHLIGHT_OUTLINE mode. In the table above, the column outlines in blue and red reflect what is actually displayed in the chart, whereas in the other TableAlarmEmphasisMode examples the outline just shows where the alarm highlighting occurs.

The default mode is ALARM_HIGHLIGHT_NONE mode.

Enhanced Out of Limit Symbol

One of the SPC chart options is to mark a data point which is outside of control limits by changing the color of the symbol. It is now possible to also also change the size and symbol type for out of limit symbols.



Title: Variable Control Chart (X-Bar & R) Part No.: 283501 Chart No.: 17

Part Name: Transmission Casing Bolt Operation:Threading

Operator:J. Fenamore Machine: #11

Date: 8/15/2017 5:30:41 PM

TIME	4:15	4:30	4:45	5:00	5:15	5:30	5:45	6:00	6:15	6:30	6:45	7:00	7:15	7:30	7:45	8:00	8:15
MEAN	28.7	27.3	29.3	29.9	27.6	34.5	31.1	28.6	28.4	36.4	29.4	32.7	32.9	38.8	30.0	29.3	28.3
RANGE	9.6	10.1	10.9	22.3	19.8	20.1	17.0	8.6	6.2	19.8	18.0	15.9	18.3	9.0	15.4	17.6	20.4
SUM	143.3	81.8	146.5	149.4	82.9	172.3	155.6	114.4	85.2	145.6	88.2	98.2	98.7	155.4	149.8	146.6	113.4
ALARM	- -	- -	- -	- H	- H	- H	- -	- -	- -	H H	- -	- -	- -	H -	- -	- -	- H
NOTES	N	N	N	N	N	N	N	N	N	N	Y	N	N	N	N	N	N

A data point which is outside of control limits can be automatically marked using color, symbol type, and color.

In the example above, the out of control symbol for the Primary chart is an extra large plus sign, while for the the Secondary chart it is an extra large square, both contrasting with the default circle symbol.

The default symbol for the out of control indication is the same as the in control indication, a filled circle. Only a color change signifies out of control. To change the notification symbol, and size, use the OutOfLimitSymbolNumber and OutOfLimitSymbolSize properties, which apply separately to the Primary and Secondary charts.

OutOfLimitSymbolNumber

Set the symbol type for data points found to be in alarm. Use one of the symbol type constants found in the ChartObj class: SQUARE, TRIANGLE, DIAMOND, CROSS, PLUS, STAR, LINE, HBAR, VBAR, CIRCLE. :

OutOfLimitSymbolSize

Set the size in pixels of the out of limit symbol:

For example:

```
// Need to have this on for symbol emphasis
```

```
          this.ChartAlarmEmphasisMode = SPCChartBase.ALARM_HIGHLIGHT_SYMBOL;


// Set symbol emphasis type, and size, for primary chart
            this.PrimaryChart.OutOfLimitSymbolNumber = ChartObj.PLUS;
            this.PrimaryChart.OutOfLimitSymbolSize = 22;

// Set symbol emphasis type, and size, for secondary chart
            this.SecondaryChart.OutOfLimitSymbolNumber = ChartObj.SQUARE;
            this.SecondaryChart.OutOfLimitSymbolSize = 18;
```

See the NewRev3Features.EnhanceLimitSymbols demo for an example.

AutoLogAlarmsAsNotes

When an alarm occurs, details of the alarm can be automatically logged as a Notes record. Just set the AutoLogAlarmsAsNotes property to true.

[C#]

```
this.AutoLogAlarmsAsNotes = true;
```

[VB]

```
Me.AutoLogAlarmsAsNotes = True
```

Creating a Batch-Based or Event-Based Variable Control Chart

The **SPCEventVariableControlChart**, the **SPCTimeVariableContolChart** and **SPCBatchVariableControlChart** derive from the **SPCChartBase** and as a result, the classes are very similar and share 95% of the same properties. Creating and initializing a batch-based SPC chart is much the same as that of a time-based SPC chart. See the example program BatchVariableControlCharts for a variety of batch SPC charts. Derive your base class from either the **SPCEventVariableControlChart** or the **SPCBatchVariableControlChart** class.

[C#]

```
public class BatchXBarRChart : SPCEventVariableControlChart
{
    ChartCalendar startTime = new ChartCalendar();
    //  SPC variable control chart type
    int charttype = SPCControlChartData.MEAN_RANGE_CHART;
    // Number of samples per sub group
    int numsamplespersubgroup = 3;
    // Number of data points  in the view
    int numdatapointsinview = 17;
    // The time increment between adjacent subgroups
    int timeincrementminutes = 15;

    public BatchXBarRChart()
    {
        // This call is required by the Windows.Forms Form Designer.
        InitializeComponent();
        this.Dock = DockStyle.Fill;
        DrawChart();
    }

    public  void DrawChart()
    {
        // Initialize the SPCEventVariableControlChart
        this.InitSPCEventVariableControlChart(charttype,
            numsamplespersubgroup, numdatapointsinview);
        // Change the default horizontal position and width of the chart
        .
        .
        .

        this.RebuildChartUsingCurrentData();
    }
}
```

[VB]

```
Public Class BatchXBarRChart
    Inherits com.quinn-curtis.spcchartnet6.SPCEventVariableControlChart

#Region " Windows Form Designer generated code "
    Public Sub New()
        MyBase.New()

        'This call is required by the Windows Form Designer.
        InitializeComponent()

        'Add any initialization after the InitializeComponent() call
        Me.Dock = DockStyle.Fill
        DrawChart()

    End Sub
    .
    .
    .
#End Region

Dim startTime As New ChartCalendar()
'  SPC variable control chart type
Dim charttype As Integer = SPCControlChartData.MEAN_RANGE_CHART
' Number of samples per sub group
Dim numsamplespersubgroup As Integer = 3
' Number of data points  in the view
Dim numdatapointsinview As Integer = 17
' The time increment between adjacent subgroups
Dim timeincrementminutes As Integer = 15

Public Sub DrawChart()
 ' Initialize the SPCEventVariableControlChart
     Me.InitSPCEventVariableControlChart(charttype, _
       numsamplespersubgroup, numdatapointsinview)
    .
    .
    .

    Me.RebuildChartUsingCurrentData()
End Sub 'DrawChart
```

Establish the control chart type (Mean Range (X-Bar R), Median Range, X-Bar Sigma (Mean Sigma), Individual Range, EWMA, MA, or CuSum) using the variable control charts **InitSPCEventVariableControlChart** (or **InitSPCEventCusumControlChart** if you are creating a cusum chart) initialization routine. Note that the X-Bar Sigma chart, with a variable subgroup sample size, is initialized using **InitSPCEventVariableControlChart** with a *charttype* value of MEAN_SIGMA_CHART_VSS. X-Bar Sigma charts with sub groups that use a variable sample size must be updated properly.

SPCEventVariableControlChart.InitSPCEventVariableControlChart Method

This initialization method initializes the most important values in the creation of a SPC chart.

[VB]

```
Overloads Public Sub InitSPCEventVariableControlChart( _
   ByVal charttype As Integer, _
   ByVal numsamplespersubgroup As Integer, _
   ByVal numdatapointsinview As Integer, _
)
```

[C#]

```
public void InitSPCEventVariableControlChart(
   int charttype,
   int numsamplespersubgroup,
   int numdatapointsinview,
);
```

Parameters

charttype

The SPC chart type parameter. Use one of the SPCControlChartData SPC chart types: MEAN_RANGE_CHART, MEDIAN_RANGE_CHART, MEAN_SIGMA_CHART, MEAN_SIGMA_CHART_VSS, INDIVIDUAL_RANGE_CHART, EWMA_CHART, LEVEY_JENNINGS_CHART, MA_CHART, MAMR_CHART, MAMS_CHART or TABCUSUM_CHART.

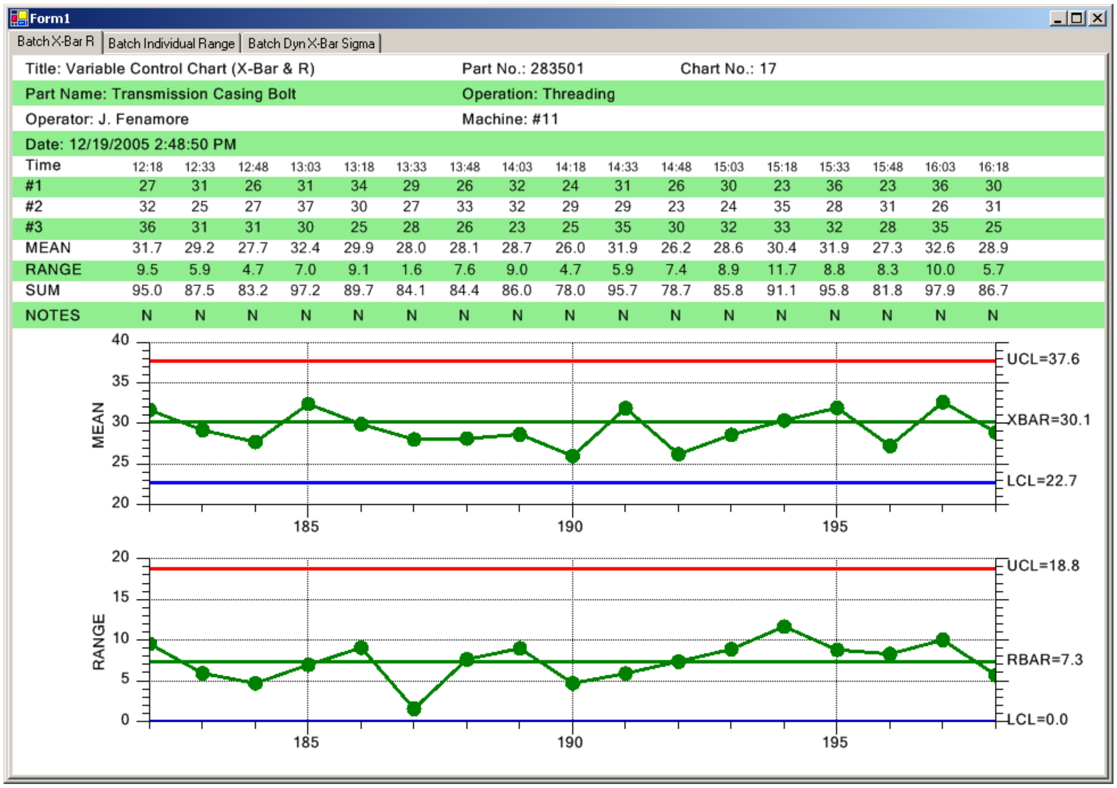
numsamplespersubgroup

Specifies the number of samples that make up a sample subgroup.

numdatapointsinview

Specifies the number of sample subgroups displayed in the graph at one time.

Update the chart data using a **ChartData.AddNewSampleRecord** override that has the batch number (*batchCounter* below) as the first parameter. Even though a time stamp value is also used in the **AddNewSampleRecord** method, it is not used in the actual graph. Instead, it is used as the time stamp for the batch in the table portion of the chart. The following code is extracted from the **BatchVariableControlChart. BatchDynXBarSigmaChart** example program**.**



Title: Variable Control Chart (X-Bar & R) Part No.: 283501 Chart No.: 17

Part Name: Transmission Casing Bolt Operation: Threading

Operator: J. Fenamore Machine: #11

Date: 12/19/2005 2:48:50 PM

Time	12:18	12:33	12:48	13:03	13:18	13:33	13:48	14:03	14:18	14:33	14:48	15:03	15:18	15:33	15:48	16:03	16:18
#1	27	31	26	31	34	29	26	32	24	31	26	30	23	36	23	36	30
#2	32	25	27	37	30	27	33	32	29	29	23	24	35	28	31	26	31
#3	36	31	31	30	25	28	26	23	25	35	30	32	33	32	28	35	25
MEAN	31.7	29.2	27.7	32.4	29.9	28.0	28.1	28.7	26.0	31.9	26.2	28.6	30.4	31.9	27.3	32.6	28.9
RANGE	9.5	5.9	4.7	7.0	9.1	1.6	7.6	9.0	4.7	5.9	7.4	8.9	11.7	8.8	8.3	10.0	5.7
SUM	95.0	87.5	83.2	97.2	89.7	84.1	84.4	86.0	78.0	95.7	78.7	85.8	91.1	95.8	81.8	97.9	86.7
NOTES	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N

[C#]

```
private void SimulateData()
{
    // batch number for a given sample subgroup
    int batchCounter = 0;
    for (int i=0; i < 200; i++)
    {
        // Important to make a new ChartCalendar object each time
       ChartCalendar timestamp = (ChartCalendar) startTime.Clone();
        // Simulate a sample subgroup record
        DoubleArray samples = this.ChartData.SimulateMeasurementRecord(30, 10);
        // Update chart data using i as the batch number
        batchCounter = i;
        // Add a new sample record to the chart data
        this.ChartData.AddNewSampleRecord(batchCounter, timestamp, samples);
        // Simulate passage of timeincrementminutes minutes
        startTime.Add(ChartObj.MINUTE, timeincrementminutes);
    }
}
```

[VB]

```
Private Sub SimulateData()
   ' batch number for a given sample subgroup
   Dim batchCounter As Integer = 0
   Dim i As Integer
   For i = 0 To 199
      ' Important to make a new ChartCalendar object each time
      Dim timestamp As ChartCalendar = CType(startTime.Clone(), ChartCalendar)
      ' Simulate a sample subgroup record
      Dim samples As DoubleArray = Me.ChartData.SimulateMeasurementRecord(30, 10)
      ' Update chart data using i as the batch number
      batchCounter = i
      ' Add a new sample record to the chart data
      Me.ChartData.AddNewSampleRecord(batchCounter, timestamp, samples)
      ' Simulate passage of timeincrementminutes minutes
      startTime.Add(ChartObj.MINUTE, timeincrementminutes)
   Next i
End Sub 'SimulateData
```

Changing the Batch and Event Control Chart X-Axis Labeling Mode

In revisions prior to 2.0, the x-axis tick marks of a batch control chart could only be labeled with the numeric batch number of the sample subgroup. While batch number labeling is still the default mode, it is now possible to label the sample subgroup tick marks using the time stamp of the sample subgroup, or a user-defined string unique to each sample subgroup.

You may find that labeling every subgroup tick mark with a time stamp, or a user-defined string, causes the axis labels to stagger because there is not enough room to display the tick mark label without overlapping its neighbor. In these cases you may wish to reduce the number of sample subgroups you show on the page using the *numdatapointsinview* variable found in all of the example programs.

```
// Number of datapoints in the view
int numdatapointsinview = 13;
```

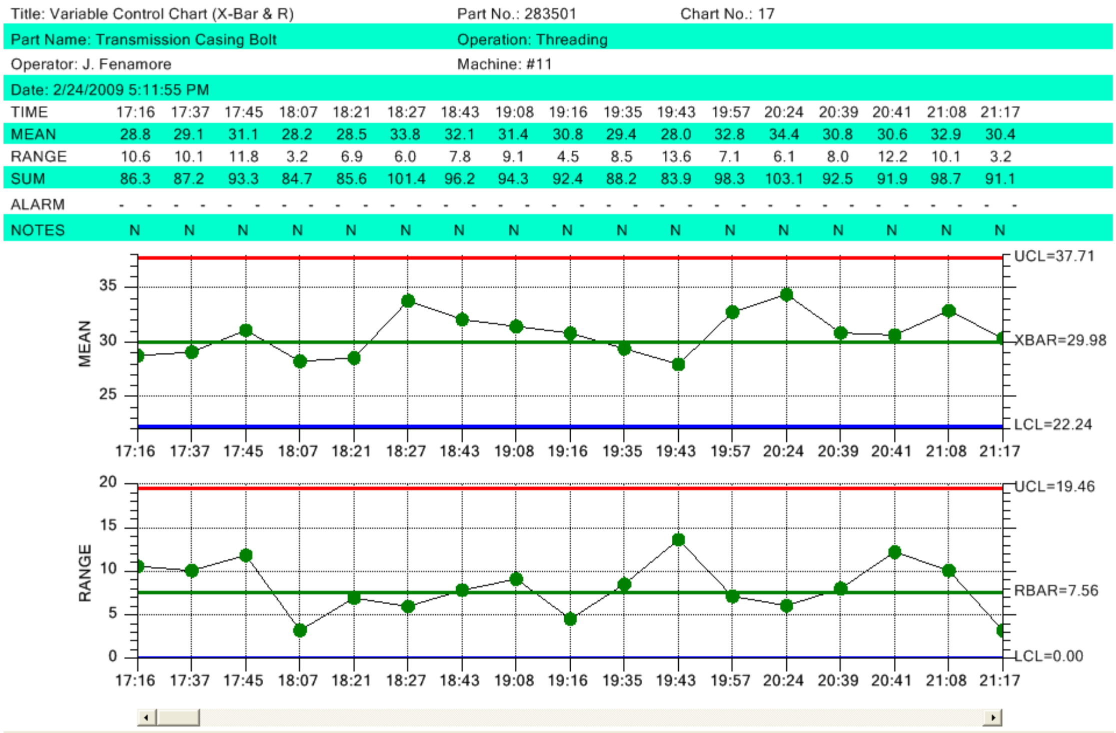
You can rotate the x-axis labels using the charts XAxisLabelRotation property.

```
C#
this.XAxisLabelRotation = 90;

VB
Me.XAxisLabelRotation = 90
```

If you rotate the x-axis labels you may need to leave more room between the primary and secondary graphs, and at the bottom, to allow for the increased height of the labels.

```
C#
this.XAxisLabelRotation = 90;
this.InterGraphMargin = 0.1;
this.GraphBottomPos = 0.85;

VB
Me.XAxisLabelRotation = 90
Me.InterGraphMargin = 0.1
Me.GraphBottomPos = 0.85
```

Batch Control Chart X-Axis Time Stamp Labeling



Title: Variable Control Chart (X-Bar & R) Part No.: 283501 Chart No.: 17
Part Name: Transmission Casing Bolt Operation: Threading
Operator: J. Fenamore Machine: #11
Date: 2/24/2009 5:11:55 PM

TIME	17:16	17:37	17:45	18:07	18:21	18:27	18:43	19:08	19:16	19:35	19:43	19:57	20:24	20:39	20:41	21:08	21:17
MEAN	28.8	29.1	31.1	28.2	28.5	33.8	32.1	31.4	30.8	29.4	28.0	32.8	34.4	30.8	30.6	32.9	30.4
RANGE	10.6	10.1	11.8	3.2	6.9	6.0	7.8	9.1	4.5	8.5	13.6	7.1	6.1	8.0	12.2	10.1	3.2
SUM	86.3	87.2	93.3	84.7	85.6	101.4	96.2	94.3	92.4	88.2	83.9	98.3	103.1	92.5	91.9	98.7	91.1
ALARM	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -
NOTES	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N

Batch X-Bar R Chart using time stamp labeling of the x-axis

Set the x-axis labeling mode using the overall charts XAxisStringLabelMode property, setting it `SPCChartObjects.AXIS_LABEL_MODE_TIME`.

[C#]
```
// enable scroll bar
this.EnableScrollBar = true;
this.EnableCategoryValues = false;

// Label the tick mark with time stamp of sample group
this.XAxisStringLabelMode = SPCChartObjects.AXIS_LABEL_MODE_TIME;
```

[VB]
```
' enable scroll bar
Me.EnableScrollBar = True
Me.EnableCategoryValues = False

' Label the tick mark with time stamp of sample group
Me.XAxisStringLabelMode = SPCChartObjects.AXIS_LABEL_MODE_TIME
```

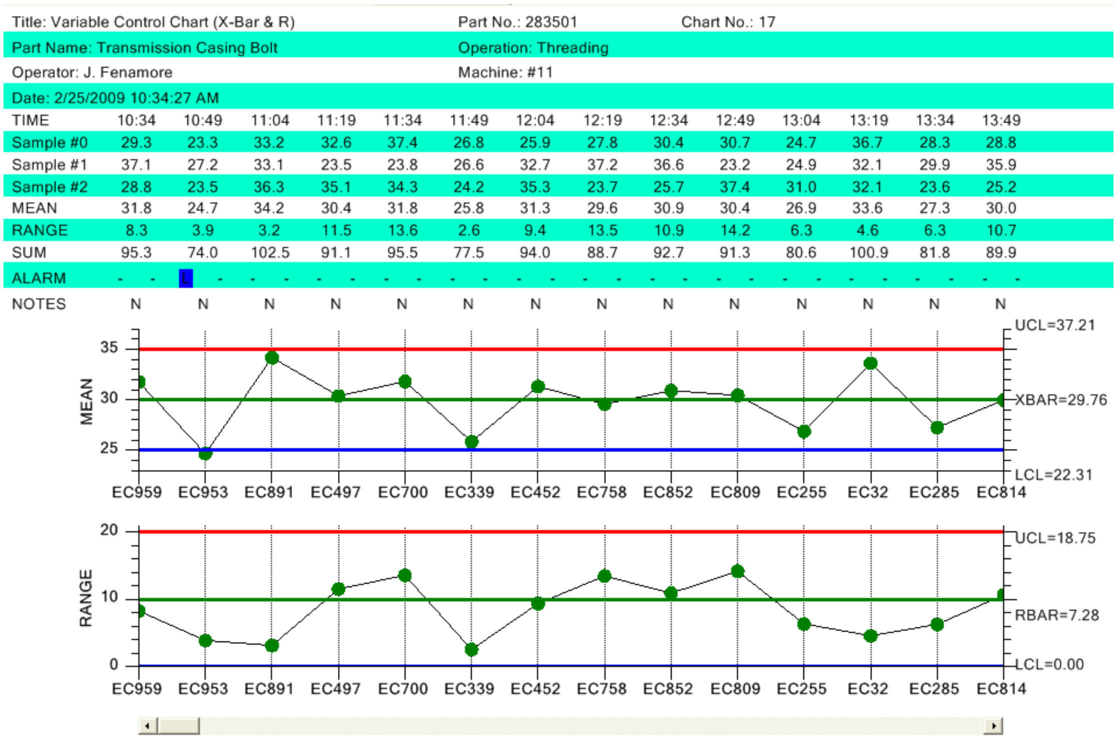
When updating the chart with sample data, use AddNewSampleRecord overload that has batch number and a time stamp parameters.

[C#]
```
this.ChartData.AddNewSampleRecord(batchCounter, timestamp, samples);
```

[VB]
```
Me.ChartData.AddNewSampleRecord(batchCounter, timestamp, samples)
```

See the example program BatchVariableControlCharts.BatchXBarRChart for a complete example. Reset the axis labeling mode back to batch number labeling by assigning the XAxisStringLabelMode property to SPCChartObjects.AXIS_LABEL_MODE_DEFAULT.

Batch Control Chart X-Axis User-Defined String Labeling



Title: Variable Control Chart (X-Bar & R) Part No.: 283501 Chart No.: 17

Part Name: Transmission Casing Bolt Operation: Threading

Operator: J. Fenamore Machine: #11

Date: 2/25/2009 10:34:27 AM

TIME	10:34	10:49	11:04	11:19	11:34	11:49	12:04	12:19	12:34	12:49	13:04	13:19	13:34	13:49
Sample #0	29.3	23.3	33.2	32.6	37.4	26.8	25.9	27.8	30.4	30.7	24.7	36.7	28.3	28.8
Sample #1	37.1	27.2	33.1	23.5	23.8	26.6	32.7	37.2	36.6	23.2	24.9	32.1	29.9	35.9
Sample #2	28.8	23.5	36.3	35.1	34.3	24.2	35.3	23.7	25.7	37.4	31.0	32.1	23.6	25.2
MEAN	31.8	24.7	34.2	30.4	31.8	25.8	31.3	29.6	30.9	30.4	26.9	33.6	27.3	30.0
RANGE	8.3	3.9	3.2	11.5	13.6	2.6	9.4	13.5	10.9	14.2	6.3	4.6	6.3	10.7
SUM	95.3	74.0	102.5	91.1	95.5	77.5	94.0	88.7	92.7	91.3	80.6	100.9	81.8	89.9
ALARM	- -	L -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -
NOTES	N	N	N	N	N	N	N	N	N	N	N	N	N	N

Batch X-Bar R Chart user-defined string labeling of the x-axis

Set the x-axis labeling mode using the overall charts XAxisStringLabelMode property, setting it `SPCChartObjects.AXIS_LABEL_MODE_STRING.`

[C#]

```
// enable scroll bar
this.EnableScrollBar = true;
this.EnableCategoryValues = false;

// Label the tick mark with user-defined strings
this.XAxisStringLabelMode = SPCChartObjects.AXIS_LABEL_MODE_STRING;
```

[VB]

```
' enable scroll bar
Me.EnableScrollBar = True
Me.EnableCategoryValues = False

' Label the tick mark with user-defined strings
Me.XAxisStringLabelMode = SPCChartObjects. AXIS_LABEL_MODE_STRING
```

Use the AddAxisUserDefinedString method to supply a new string for every new sample subgroup. It must be called every time the AddNewSampleRecord method is called, or the user-defined strings will get out of sync with their respective sample subgroup. Reset the axis labeling mode back to batch number labeling by assigning the XAxisStringLabelMode property to SPCChartObjects.AXIS_LABEL_MODE_DEFAULT.

[C#]

```
this.ChartData.AddNewSampleRecord(batchCounter, timestamp, samples,
variableControlLimits);

// Make a random string to simulate some sort of batch sample group ID
int randomnum= (int) (1000 * ChartSupport.GetRandomDouble());
String batchidstring = "EC" + randomnum.ToString();
this.ChartData.AddAxisUserDefinedString(batchidstring);
```

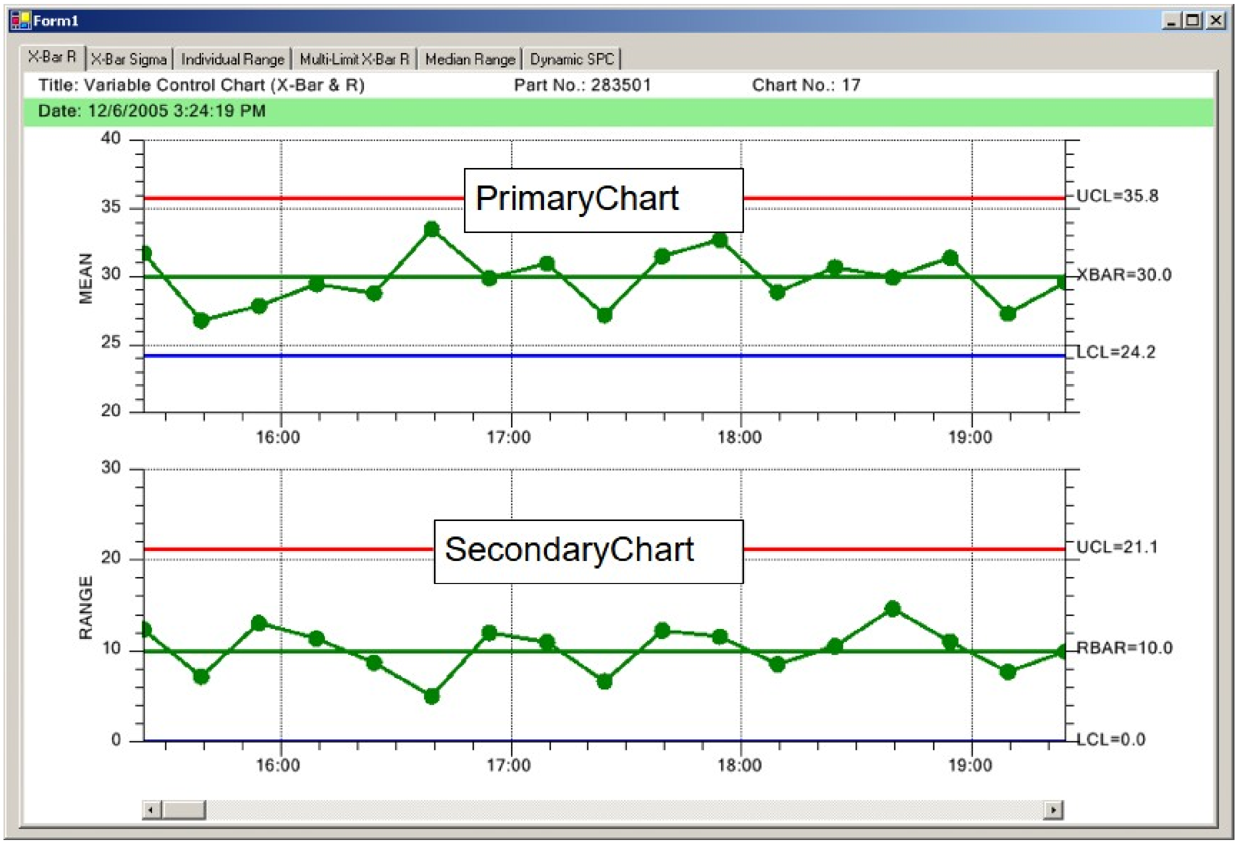
[VB]

```
Me.ChartData.AddNewSampleRecord(batchCounter, timestamp, samples, variableControlLimits)
Dim randomnum As Integer = CInt((1000 * ChartSupport.GetRandomDouble()))
Dim batchidstring As String = "EC" & randomnum.ToString()
Me.ChartData.AddAxisUserDefinedString(batchidstring)
```

See the example program BatchVariableControlCharts.VariableControlLimits for a complete example.

Changing Default Characteristics of the Chart

All *Variable Control Charts* have two distinct graphs, each with its own set of properties. The top graph is the Primary Chart, and the bottom graph is the Secondary Chart.



Title: Variable Control Chart (X-Bar & R) Part No.: 283501 Chart No.: 17

Date: 12/6/2005 3:24:19 PM

Logically enough, the properties of the objects that make up each of these graphs are stored in properties named **PrimaryChart** and **SecondaryChart**. Once the graph is initialized (using the **InitSPCEventVariableControlChart**, **InitSPCTimeVariableControlChart**, or **InitSPCBatchVariableControlChart** method), you can modify the default characteristics of each graph using these properties.

[C#]

```
// Initialize the SPCTimeVariableControlChart
this.InitSPCTimeVariableControlChart(charttype,
        numsamplespersubgroup, numdatapointsinview, timeincrementminutes);
        .
        .
        .

this.PrimaryChart.XAxis.LineColor = Color.Blue;
this.PrimaryChart.XAxis.LineWidth = 3;

this.SecondaryChart.YAxis1.LineColor = Color.Green;
this.SecondaryChart.YAxis2.LineColor = Color.Red;
this.SecondaryChart.YAxis1.LineWidth = 3;
this.SecondaryChart.YAxis2.LineWidth = 3;

this.PrimaryChart.ProcessVariableData.LineMarkerPlot.LineAttributes.PrimaryColor =
Color.Black;
this.PrimaryChart.ProcessVariableData.LineMarkerPlot.SymbolAttributes.PrimaryColor =
Color.BlueViolet;
```

```
this.PrimaryChart.ProcessVariableData.LineMarkerPlot.SymbolAttributes.FillColor =
Color.Beige;
this.PrimaryChart.GraphBackground.FillColor = Color.LightGray;
this.PrimaryChart.PlotBackground.FillColor = Color.LightGoldenrodYellow;
```

[VB]

```
' Initialize the SPCTimeVariableControlChart
Me.InitSPCTimeVariableControlChart(charttype, numsamplespersubgroup, numdatapointsinview,
timeincrementminutes)
.
.
.
Me.PrimaryChart.XAxis.LineColor = Color.Blue
Me.PrimaryChart.XAxis.LineWidth = 3

Me.SecondaryChart.YAxis1.LineColor = Color.Green
Me.SecondaryChart.YAxis2.LineColor = Color.Red
Me.SecondaryChart.YAxis1.LineWidth = 3
Me.SecondaryChart.YAxis2.LineWidth = 3

Me.PrimaryChart.ProcessVariableData.LineMarkerPlot.LineAttributes.PrimaryColor =
Color.Black
Me.PrimaryChart.ProcessVariableData.LineMarkerPlot.SymbolAttributes.PrimaryColor =
Color.BlueViolet
Me.PrimaryChart.ProcessVariableData.LineMarkerPlot.SymbolAttributes.FillColor =
Color.Beige
Me.PrimaryChart.GraphBackground.FillColor = Color.LightGray
```

The **PrimaryChart** and **SecondaryChart** objects are both instances of the **SPCChartObjects** class. The **SPCChartObjects** class contains the objects needed to display a single graph. Below you will find a summary of the class properties.

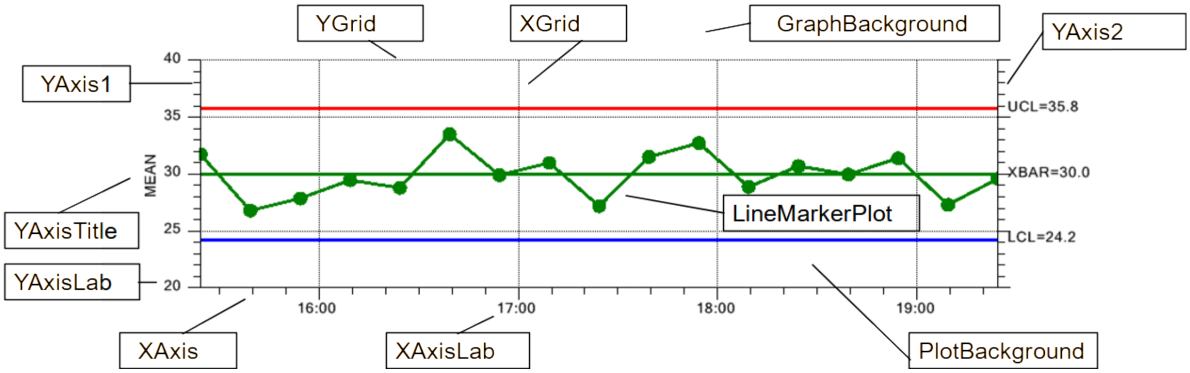
Public Instance Properties

AnnotationArray	Get the array of TextObject objects, representing the annotations of the chart.
AnnotationFont	Set/Get annotation font.
AnnotationNudge	Set/Get the x and y-values use to offset a data points annotation with respect to the actual data point.
AxisLabelFont	Set/Get the font used to label the x- and y- axes.
AxisTitleFont	Set/Get the font used for the axes titles.
ControlLabelPosition	Set/Get that numeric label for a control limit is placed inside, or outside the plot area INSIDE_PLOTAREA.
ControlLimitData	Get the array of the plot objects associated with control limits.
Datatooltip	Get a reference to the charts tooltip.
DefaultChartBackgroundColor	Get/Set the default background color for the graph area.
DefaultNumberControlLimits	Set/Get the number of control limits in the chart.
DefaultPlotBackgroundColor	Get/Set the default background color for the

	plot area.
DisplayChart	Set to true to enable the drawing of this chart.
DisplayFrequencyHistogram	Set to true to enable the drawing of the frequency histogram attached to the chart.
FrequencyHistogramChart	Get a reference to the optional frequency histogram attached to the chart.
GraphBackground	Get a reference to the charts graph background object.
BatchIncrement	Set/Get increment between adjacent samples of Batch type charts that use a numeric x-scale.
BatchStartValue	Set/Get the starting numeric value of the x-scale for Batch type charts that use a numeric x-scale.
BatchStopValue	Set/Get the ending numeric value of the x-scale for Batch type charts that use a numeric x-scale.
Header	Get a reference to the charts header.
HeaderFont	Set/Get the font used for the chart title.
HistogramStartPos	Set/Get the left edge, using normalized coordinates, of the frequency histogram plotting area.
HistogramOffset	Set/Get the offset with respect to the GraphStartPosX value, using normalized coordinates, of the frequency histogram plotting area.
MaxY	Set/Get the maximum value used to scale the y-axis of the chart.
MinY	Set/Get the minimum value used to scale the y-axis of the chart.
ParentSPCChartBase	Set/Get that parent SPCChartBase object.
PlotBackground	Get a reference to the charts plot background object.
PlotMeasurementValues	Set to true to enable the plotting of all sampled values, as a scatter plot, in addition to the mean or median values.
PPhysTransform1	Gets a reference to the charts physical coordinate system.
ProcessVariableData	Holds a reference to an object encapsulating the plot object data associated with the main variable of the chart.
SampledDataData	Get the array of the sample data.
SubHead	Get a reference to the charts subhead.
SubheadFont	Set/Get the font used for the chart subhead.
TableFont	Set/Get the font used for the data table.
TextTemplate	Get/Set the text template for the data tooltip.
TimeIncrementMinutes	Get/Set the increment between adjacent samples

	of charts that use a numeric x-scale.
ToolTipFont	Set/Get tooltip font.
ToolTipSymbol	Get a reference to the charts tooltip symbol.
XAxis	Get a reference to the charts x-axis.
XAxisLab	Get a reference to the charts x-axis labels.
XGrid	Get a reference to the charts x-axis grid.
XValueTemplate	Get/Set the x-value template for the data tooltip.
YAxis1	Get a reference to the charts left y-axis.
YAxis2	Get a reference to the charts right y-axis.
YAxisLab	Get a reference to the charts left y-axis labels.
YAxisTitle	Get a reference to the charts left y-axis title.
YGrid	Get a reference to the charts y-axis grid.
YValueTemplate	Get/Set the y-value template for the data tooltip.

The main objects of the graph are labeled in the graph below.



7. SPC Attribute Control Charts

SPCEventAttributeControlChart
SPCTimeAttributeControlChart
SPCBatchAttributeControlChart

Attribute Control Charts are a set of control charts specifically designed for tracking product defects (also called non-conformities). These types of defects are binary in nature (yes/no), where a part has one or more defects, or it doesn't. Examples of defects are paint scratches, discolorations, breaks in the weave of a textile, dents, cuts, etc. Think of the last car that you bought. The defects in each sample group are counted and run through some statistical calculations. Depending on the type of *Attribute Control Chart*, the number of defective parts are tracked (p-chart and np-chart), or alternatively, the number of defects are tracked (u-chart, c-chart). The difference in terminology "number of defective parts" and "number of defects" is highly significant, since a single part not only can have multiple defect categories (scratch, color, dent, etc), it can also have multiple defects per category. A single part may have 0 – N defects. So keeping track of the number of defective parts is statistically different from keeping track of the number of defects. This affects the way the control limits for each chart are calculated.

p-Chart - Also known as the Percent or Fraction Defective Parts Chart
For a sample subgroup, the number of defective parts is measured and plotted as either a percentage of the total subgroup sample size, or a fraction of the total subgroup sample size. Since the plotted value is a fraction or percent of the sample subgroup size, the size of the sample group can vary without rendering the chart useless.

np-Chart – Also known as the Number Defective Parts Chart
For a sample subgroup, the number of defective parts is measured and plotted as a simple count. Statistically, in order to compare number of defective parts for one subgroup with the other subgroups, this type of chart requires that the subgroup sample size is fixed across all subgroups.

c-Chart - Also known as the Number of Defects or Number of Non-Conformities Chart
For a sample subgroup, the number of times a defect occurs is measured and plotted as a simple count. Statistically, in order to compare number of defects for one subgroup with

the other subgroups, this type of chart requires that the subgroup sample size is fixed across all subgroups.

u-Chart – Also known as the Number of Defects per Unit or Number of Non-Conformities per Unit Chart
For a sample subgroup, the number of times a defect occurs is measured and plotted as either a percentage of the total subgroup sample size, or a fraction of the total subgroup sample size. Since the plotted value is a fraction or percent of the sample subgroup size, the size of the sample group can vary without rendering the chart useless.

DPMO Chart – Also known as the Number of Defects per Million Chart
For a sample subgroup, the number of times a defect occurs is measured and plotted as a value normalized to defects per million. Since the plotted value is normalized to a fixed sample subgroup size, the size of the sample group can vary without rendering the chart useless.

Time-Based and Batch-Based SPC Charts

Attribute Control Charts are further categorized as either time- or batch- based. Use time-based SPC charts when data is collected using a subgroup interval corresponding to a specific time interval. Use batch-based SPC charts when the data subgroup interval is a sequential batch number that does not correspond to a uniform time interval. The major difference in these two types of SPC charts is the display of the x-axis. Control charts that sample using a uniform time interval will generally use a time-based x-axis, with time/date axis labels. Control charts that sample based on batches will generally use a numeric-based x-axis, with numeric axis labels.

SPECIAL NOTE: The time-based and batch-based SPC charts have been deprecated and replaced by the new event-based charts. In order to maintain backward compatibility, we also keep the old SPCTime... and SPCBatch... control chart classes, but derive them from the new Event-based SPC chart classes. The only difference you might see is an actual benefit. No matter what the time stamp is on a SPCTime... chart, adjacent points will always be equally spaced. So if your sample interval is irregular, or you even skip days or weeks in your sampling, the resulting chart will still display equally spaced adjacent sample records. You can read more about Event coordinates in the QCChart2D manual which is found in the same folder as QCSPCChart manual. **Furthermore**, if you are using the SPCTimeAttributeControlChart class, we recommend that you instead use either the SPCBatchAttributeControl (or SPCEventAttributeControlChart class. If you want to see time/date values on the x-axis, set the XAxisStringLabelMode of the chart to AXIS_LABEL_MODE_TIME.

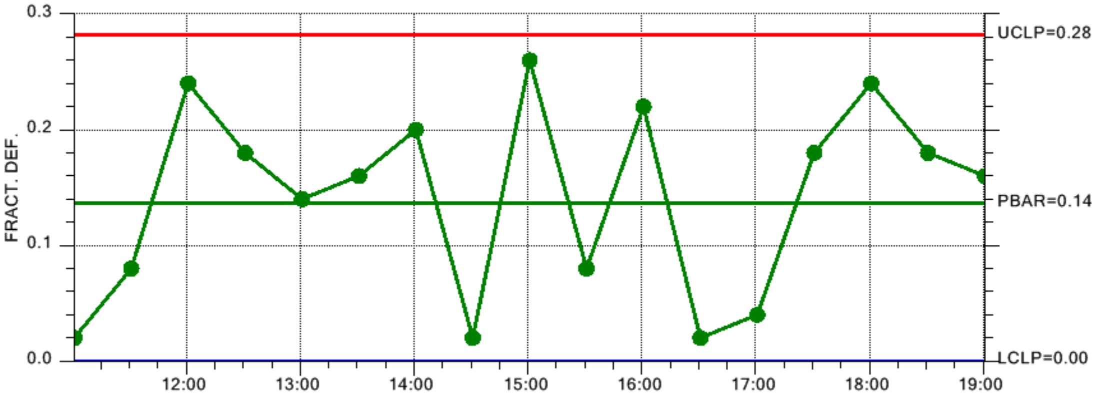
```
this.XAxisStringLabelMode = SPCChartObjects.AXIS_LABEL_MODE_TIME
```

The new class heirarchy looks like:

ChartView

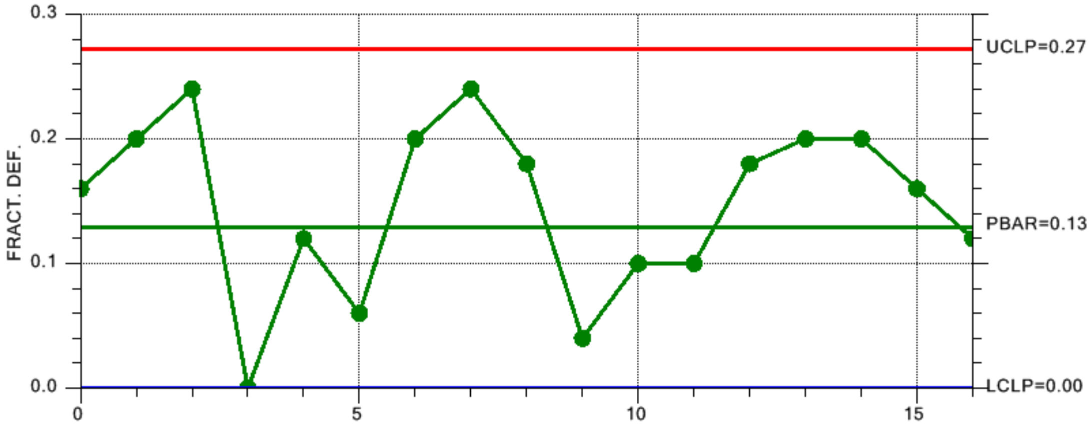
```
ChartView
        SPCChartBase
                SPCEventVariableControlChart
                        SPCTimeVariableControlChart
                        SPCBatchVariableControlChart
                SPCEventAttributeControlChart
                        SPCTimeAttributeControlChart
                        SPCBatchAttributeControlChart
```

Time-Based Attribute Control Chart



Note the time-based x-axis.

Batch-Based Attribute Control Chart



Note the numeric based x-axis.

Attribute Control Charts Consist of Only One Graph

Whereas the *Variable Control Charts* contain two different graphs, which we refer to generically as the primary and secondary graphs of the chart, *Attribute Control Charts* only have a single graph, which we refer to generically as the primary graph of the chart.

EWM<A

Creating an Attribute Control Chart

First, select whether you want to use a time-based variable control chart (use **SPCTimeAttributeControlChart**), a batch-based variable control chart (use **SPCBatchAttributeControlChart**), or an event-based chart (**SPCEventAttributeControlChart**). Use that class as the base class for your chart. Since the two classes are very similar and share 95% of all properties in common, only the **SPCEventAttributeControlChart** is discussed in detail, with the differences between the two classes discussed at the end of the chapter.

[C#]

```
public class FractionDefectivePartsControlChart :
    com.quinn-curtis.spcchartnet6.SPCEventAttributeControlChart
{
    private System.ComponentModel.IContainer components;
    ChartCalendar startTime = new ChartCalendar();
    //  SPC attribute control chart type
    int charttype = SPCControlChartData.FRACTION_DEFECTIVE_PARTS_CHART;
    // Number of samples per sub group
    int numsamplespersubgroup = 50;
    // Number of defect categories
    int numcategories = 6;
    // Number of data points in the view
    int numdatapointsinview = 17;
    // The time increment between adjacent subgroups
    int timeincrementminutes = 30;

    public FractionDefectivePartsControlChart()
    {
        // This call is required by the Windows.Forms Form Designer.
        InitializeComponent();
        // Have the chart fill parent client area
        this.Dock = DockStyle.Fill;
        // Define and draw chart
        InitializeChart();
    }

    void InitializeChart ()
    {
        // Initialize the SPCEventAttributeControlChart
        this.InitSPCEventAttributeControlChart(charttype, numcategories,
            numsamplespersubgroup, numdatapointsinview);
        .
        .
        .

        this.RebuildChartUsingCurrentData();

    }
}
```

[VB]

```
Public Class SimpleAttributeControlChart
    Inherits com.quinn-curtis.spcchartnet6.SPCEventAttributeControlChart

#Region " Windows Form Designer generated code "

    Public Sub New()
        MyBase.New()
        'This call is required by the Windows Form Designer.
        InitializeComponent()
        'Add any initialization after the InitializeComponent() call
        ' Have the chart fill parent client area
        Me.Dock = DockStyle.Fill
        ' Define and draw chart
        InitializeChart ()
    End Sub
.
.
.

#End Region

Dim startTime As New ChartCalendar()
'  SPC attribute control chart type
Dim charttype As Integer = SPCControlChartData.FRACTION_DEFECTIVE_PARTS_CHART
' Number of samples per sub group
Dim numsamplespersubgroup As Integer = 50
' Number of defect categories
Dim numcategories As Integer = 5
' Number of data points in the view
 Dim numdatapointsinview As Integer = 17
 ' The time increment between adjacent subgroups
 Dim timeincrementminutes As Integer = 30

 Sub InitializeChart ()
        ' Initialize the SPCEventAttributeControlChart
    Me.InitSPCEventAttributeControlChart(charttype, numcategories, _
       numsamplespersubgroup, numdatapointsinview)
    .
    .
    .
    Me.RebuildChartUsingCurrentData()
  End Sub 'DrawChart
    .
    .
    .
End Class
```

SPCEventAttributeControlChart Members

SPCEventAttributeControlChart overview

Public Instance Constructors

SPCEventAttributeControlChart	Overloaded. Initializes a new instance of the SPCEventAttributeControlChart class.

Public Instance Methods

InitSPCEventAttributeControlChart	Overloaded. Initialize the class for a specific SPC chart type.

The control chart type (p-, np-, c- and u-charts) is established in the attribute control charts **InitSPCEventAttributeControlChart** initialization routine.

SPCEventAttributeControlChart.InitSPCEventAttributeControlChart Method

This initialization method initializes the most important values in the creation of a SPC chart.

EWM<A

[VB]

```
Overloads Public Sub InitSPCEventAttributeControlChart( _
   ByVal charttype As Integer, _
   ByVal numcategories As Integer, _
   ByVal numsamplespersubgroup As Integer, _
   ByVal numdatapointsinview As Integer, _
   ByVal timeincremenminutest As Integer _
)
```

[C#]

```
public void InitSPCEventAttributeControlChart(
   int charttype,
   int numcategories,
   int numsamplespersubgroup,
   int numdatapointsinview,
   int timeincrementminutes
);
```

Parameters

charttype

Specifies the chart type. Use one of the SPC Attribute Control chart types: PERCENT_DEFECTIVE_PARTS_CHART, FRACTION_DEFECTIVE_PARTS_CHART, NUMBER_DEFECTIVE_PARTS_CHART, NUMBER_DEFECTS_PERUNIT_CHART, NUMBER_DEFECTS_CHART, NUMBER_DEFECTS_PER_MILLION_CHART.

numcategories

In Attribute Control Charts this value represents the number of defect categories used to determine defect counts.

numsamplespersubgroup

In an Attribute Control chart it represents the total sample size per sample subgroup from which the defect data is counted.

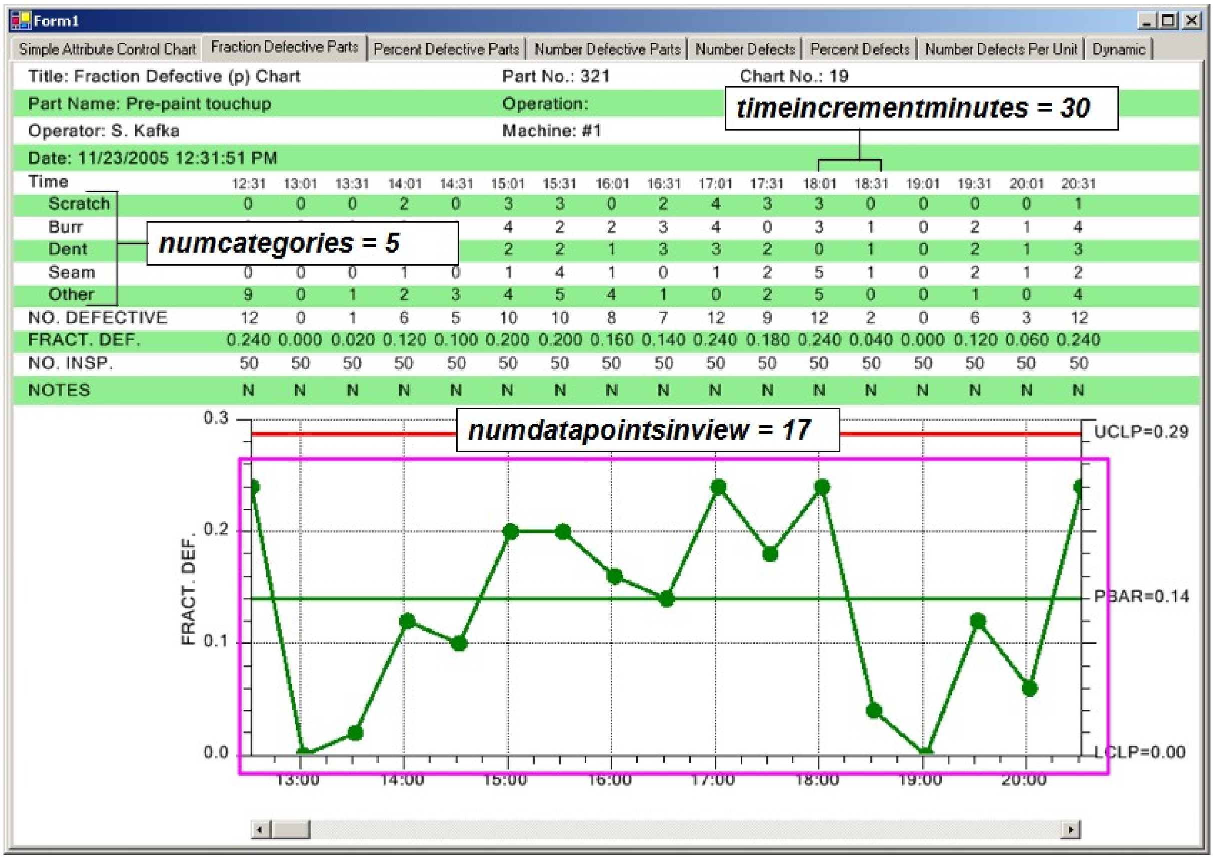
numdatapointsinview

Specifies the number of sample subgroups displayed in the graph at one time.

timeincremenminutes

Specifies the normal time increment between adjacent subgroup samples.

The image below further clarifies how these parameters affect the attribute control chart.



Title: Fraction Defective (p) Chart　Part No.: 321　Chart No.: 19

Part Name: Pre-paint touchup　Operation:

Operator: S. Kafka　Machine: #1

Date: 11/23/2005 12:31:51 PM

Time	12:31	13:01	13:31	14:01	14:31	15:01	15:31	16:01	16:31	17:01	17:31	18:01	18:31	19:01	19:31	20:01	20:31
Scratch	0	0	0	2	0	3	3	0	2	4	3	3	0	0	0	0	1
Burr						4	2	2	3	4	0	3	1	0	2	1	4
Dent						2	2	1	3	3	2	0	1	0	2	1	3
Seam	0	0	0	1	0	1	4	1	0	1	2	5	1	0	2	1	2
Other	9	0	1	2	3	4	5	4	1	0	2	5	0	0	1	0	4
NO. DEFECTIVE	12	0	1	6	5	10	10	8	7	12	9	12	2	0	6	3	12
FRACT. DEF.	0.240	0.000	0.020	0.120	0.100	0.200	0.200	0.160	0.140	0.240	0.180	0.240	0.040	0.000	0.120	0.060	0.240
NO. INSP.	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
NOTES	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N

Once the Init routine is called, the chart can be further customized using properties inherited from **SPCBaseChart**, described below.

Public Static (Shared) Properties

DefaultChartFontString	Set/Get the default font used in the table display.

Public Instance Constructors

SPCChartBase	Overloaded. Initializes a new instance of the SPCChartBase class.

Public Instance Properties

AutoLogAlarmsAsNotes	Set to true to automatically log alarm details in the sample interval Notes record.
BottomLabelMargin	Get/Set an additional margin, in normalized coordinates, if only the primary graphs is displayed, allowing for the x-axis labels
ChartData	Get the object that holds the descriptive text, sampled

	and calculated values associated with the control chart.
ChartAlarmEmphasisMode	Set to SPCChartBaseALARM_HIGHLIGHT_SYMBOL to highlight the process variable symbol if an alarm condition exists. Set to Set to SPCChartBase.ALARM_NO_HIGHLIGHT_SYMBOL to turn off alarm highlighting.
ChartTable	Get the object that holds the data table information needed to display the data table along with the chart
DefaultControlLimitSigma	Set/Get that SPC control limits are to be calculated using the 3 sigma level standard.
EnableAlarmStatusValues	If set true enables the alarm status row of the chart table.
EnableCalculatedValues	If set true enables the calculated values rows of the data table
EnableCategoryValues	If set true enables the category or sample values rows of the data table
EnableDataToolTip	If set true enables data tooltips
EnableInputStringsDisplay	If set true enables the input string rows of the data table
EnableNotes	If set true enables the notes row of the data table
EnableNotesToolTip	If set true enables data tooltips
EnableScrollBar	If set true the scroll bar is added to the bottom of the chart.
EnableTimeValues	If set true enables the time row of the data table
EnableTotalSamplesValues	If set true enables the total of sampled values row of the data table
GraphBottomPos	Get/Set the bottom edge, using normalized coordinates, of the plotting area for the secondary chart
GraphStartPosX	Get/Set the left edge, using normalized coordinates, of the plotting area for both primary and secondary charts
GraphStartPosY1	Get the top edge, using normalized coordinates, of the plotting area for the primary chart
GraphStartPosY2	Get the top edge, using normalized coordinates, of the plotting area for the secondary chart
GraphStopPosX	Get/Set the right edge, using normalized coordinates, of the plotting area for both primary and secondary charts
GraphStopPosY1	Get the bottom edge, using normalized coordinates, of the plotting area for the primary chart
GraphStopPosY2	Get the bottom edge, using normalized coordinates, of the plotting area for the secondary chart
GraphTopTableOffset	Get/Set the offset of the top of the primary chart from

	the bottom of the data table, using normalized coordinates
HeaderStringsLevel	Set/Get the level of header strings to include in the chart. Use one of the SPCControlChartData header strings constants: HEADER_STRINGS_LEVEL0, HEADER_STRINGS_LEVEL1, HEADER_STRINGS_LEVEL2, or HEADER_STRINGS_LEVEL3
InterGraphMargin	Get/Set the margin, in normalized coordinates, between the primary and secondary charts
MultipleMouseListener	Set/Get the MultiMouseListener.
PrimaryChart	Get the object that holds he the chart objects needed to display the primary chart
ScrollBarBottomPosition	Get/Set the bottom edge, using normalized coordinates, of the optional scroll bar
ScrollBarPixelHeight	Get/Set the height of the scrollbar in pixels
SecondaryChart	Get the object that holds he the chart objects needed to display the secondary chart
SPCChartType	Specifies the control chart type: use one of the SPCControlChartData chart type constants: MEAN_RANGE_CHART, MEDIAN_RANGE_CHART, MEAN_SIGMA_CHART, MEAN_SIGMA_CHART_VSS, INDIVIDUAL_RANGE_CHART, CUSTOM_ATTRIBUTE_CONTROL_CHART, PERCENT_DEFECTIVE_PARTS_CHART, FRACTION_DEFECTIVE_PARTS_CHART, NUMBER_DEFECTIVE_PARTS_CHART, NUMBER_DEFECTS_CHART, NUMBER_DEFECTS_PERUNIT_CHART, NUMBER_DEFECTS_PER_MILLION_CHART.
TableAlarmEmphasisMode	Set the table alarm highlighting to one of the SPCChartBase table highlight constants: ALARM_HIGHLIGHT_NONE, ALARM_HIGHLIGHT_TEXT, ALARM_HIGHLIGHT_OUTLINE, ALARM_HIGHLIGHT_BAR
XScaleMode	Set/Get whether the x-axis is time based, or numeric based.

Public Instance Methods

AddAnnotation	Overloaded. Add a simple annotation to a

	data point in the specified SPC chart.
AutoCalculateControlLimits	Using the current sampled data values, high, target and low control limits are calculated for both primary and secondary charts using an algorithm appropriate to the SPC chart type.
AutoCalculatePrimaryControlLimits	Using the current sampled data values, high, target and low control limits are calculated for the primary chart using an algorithm appropriate to the SPC chart type.
AutoCalculateSecondaryControlLimits	Using the current sampled data values, high, target and low control limits are calculated for the primary chart using an algorithm appropriate to the SPC chart type.
AutoScaleChartYRange	Auto-scale the y-range of the SPC chart so that all of the sampled data and chart control limits are within the bounds of the y-axis.
AutoScalePrimaryChartYRange	Auto-scale the y-range of the primary SPC chart so that all of the sampled data and chart control limits are within the bounds of the y-axis.
AutoScaleSecondaryChartYRange	Auto-scale the y-range of the SPC chart so that all of the sampled data and chart control limits are within the bounds of the y-axis.
Copy	Overloaded. Copies the source object.
Draw	Overrides the Draw method of the underlying ChartView class, so that the scroll bar can be properly repositioned if the size of the window changes. The graphics context the chart is drawn to.
InitSPCChartBase	This initialization method initializes the most important values in the creation of a SPC chart.
IsTimeScale	Returns true if the coordinate system has a time based x-axis. The coordinate system of the chart.
MakeControlLinePlot	Draw a control line, either a simple straight line, or a variable control line, for the specified chart.
RebuildChartUsingCurrentData	Rebuild the graph taking into account the most recent data values.
RescaleGraphsToScrollbar	Rescale primary and secondary charts based on the position of the value of the scroll bar. The thumb position of the scroll bar.
ResetSPCChartData	Reset the history buffers of all of the SPC

	data objects.
UpdateControlLimitLabel	Creates a numeric label of the control limit, and adds the numeric label to the spc chart.

Special Note for DPMO Charts

The NUMBER_DEFECTS_PER_MILLION_CHART has an important parameter you may need to set. DPMO charts use an important parameter known is the *defect opportunites per unit*. The default value for the parameter is 1. So if you are using 1 as the the value of *defect opportunites per unit* in your chart, you don't need to do anything. If your value is greater than 1, you need to specify that using code similar to below.

```
C#            this.ChartData.DefectOpportunitiesPerUnit = 5;

VB             Me.ChartData.DefectOpportunitiesPerUnit = 5
```

Adding New Sample Records for Attribute Control Charts.

Attribute Control Chart Cross Reference

p-chart =	FRACTION_DEFECTIVE_PARTS_CHART or PERCENT_DEFECTIVE_PARTS_CHART
np-chart =	NUMBER_DEFECTIVE_PARTS_CHART
c-chart =	NUMBER_DEFECTS_CHART
u-chart =	NUMBER_DEFECTS_PERUNIT_CHART
DPMO =	NUMBER_DEFECTS_PER_MILLION_CHART

Updating p- and np-charts

In attribute control charts, the meaning of the data in the *samples* array varies, depending on whether the attribute control chart measures the number of defective parts (p-, and np-charts), or the total number of defects (u- and c-charts). The major anomaly is that while the p- and np-charts plot the fraction or number of defective parts, the table portion of the chart can display defect counts for any number of defect categories (i.e. paint scratches, dents, burrs, etc.). It is critical to understand that total number of defects, i.e. the sum of the items in the defect categories for a give sample subgroup, do NOT have to add up to the number of defective parts for the sample subgroup. Every defective part not only can have one or more defects, it can have multiple defects of the same defect category. The total number of defects for a sample subgroup will always be equal to or greater than the number of defective parts. When using p- and np-charts that display defect category counts as part of the table, where N is the *numcategories* parameter in the **InitSPCEventAttributeControlChart** or **InitSPCBatchAttributeControlChart** initialization call, the first N elements of the *samples* array holds the defect count for each category. The N+1 element of the *samples* array holds the total defective parts count. For example, if you initialized the chart with a *numcategories* parameter to five, signifying that you had five defect categories, you would use a *samples* array sized to six, as in the code below:

[C#]

```
DoubleArray samples = new DoubleArray(6);
//  ChartCalendar initialized with current  time by default
ChartCalendar timestamp = new ChartCalendar();
// Place sample values in array
samples[0] = 3;     // Number of defects for defect category #1
samples[1] = 0;     // Number of defects for defect category #2
samples[2] = 4;     // Number of defects for defect category #3
samples[3] = 2;     // Number of defects for defect category #4
samples[4] = 3;     // Number of defects for defect category #5

samples[5] = 4;    // TOTAL number of defective parts in the sample

// Add the new sample subgroup to the chart
this.ChartData.AddNewSampleRecord(timestamp, samples);
```

[VB]

```
Dim samples As DoubleArray = New DoubleArray(6)
'  ChartCalendar initialized with current  time by default
Dim timestamp As ChartCalendar = New ChartCalendar()
' Place sample values in array
samples(0) = 3      ` Number of defects for defect category #1
samples(1) = 0      ` Number of defects for defect category #2
samples(2) = 4      ` Number of defects for defect category #3
samples(3) = 2      ` Number of defects for defect category #4
samples(4) = 3      ` Number of defects for defect category #5

samples(5) = 4     ` TOTAL number of defective parts in the sample

' Add the new sample subgroup to the chart
 Me.ChartData.AddNewSampleRecord(timestamp, samples)
```

This is obscured in our example programs a bit because we use a special method to simulate defect data for n- and np-charts. The code below is extracted from our TimeAttributeControlCharts.NumberDefectivePartsControlChart example program.

[C#]

```
DoubleArray samples = this.ChartData.SimulateDefectRecord(50 * 0.134,
        SPCControlChartData.NUMBER_DEFECTIVE_PARTS_CHART);
// Add new sample record
this.ChartData.AddNewSampleRecord( timestamp, samples);
```

[VB]

```
Dim samples As DoubleArray = Me.ChartData.SimulateDefectRecord(50 * 0.134, _
    SPCControlChartData.NUMBER_DEFECTIVE_PARTS_CHART)
' Add new sample record
Me.ChartData.AddNewSampleRecord(timestamp, samples)
```

This particular overload for **ChartData.SimulateDefectRecord** knows that since it is a NUMBER_DEFECTIVE_PARTS_CHART chart (np-chart), and that since the **ChartData** object was setup with five categories in the **InitSPCTimeAttributeControlChart** call, that is should return a **DoubleArray** with (5 + 1 = 6) elements, the first five elements representing simulated defect counts for the five defect categories, and the sixth element the simulated defective parts count. The defect category count data of the *samples* array is only used in the table part of the display; the defect category counts play NO role in the actual SPC chart. The only value that is used in plotting the SPC chart is the last element in the *samples* array, the defective parts count for the sample subgroup.

Updating c- and u-charts

In c- and u-charts the number of defective parts is of no consequence. The only thing that is tracked is the number of defects. Therefore, there is no extra array element tacked onto the end of the *samples* array. Each element of the *samples* array corresponds to the total number of defects for a given defect category. If the *numcategories* parameter in the **InitSPCEventAttributeControlChart**, **InitSPCTimeAttributeControlChart** or **InitSPCBatchAttributeControlChart** is initialized to five, the total number of elements in the *samples* array should be five. For example:

[C#]

```
DoubleArray samples = new DoubleArray(5);
//  ChartCalendar initialized with current  time by default
ChartCalendar timestamp = new ChartCalendar();
// Place sample values in array
samples[0] = 3;     // Number of defects for defect category #1
samples[1] = 0;     // Number of defects for defect category #2
samples[2] = 4;     // Number of defects for defect category #3
samples[3] = 2;     // Number of defects for defect category #4
samples[4] = 3;     // Number of defects for defect category #5

// Add the new sample subgroup to the chart
this.ChartData.AddNewSampleRecord(timestamp, samples);
```

[VB]

```
Dim samples As DoubleArray = New DoubleArray(5)
'  ChartCalendar initialized with current  time by default
Dim timestamp As ChartCalendar = New ChartCalendar()
' Place sample values in array
samples(0) = 3      ‘ Number of defects for defect category #1
samples(1) = 0      ‘ Number of defects for defect category #2
samples(2) = 4      ‘ Number of defects for defect category #3
samples(3) = 2      ‘ Number of defects for defect category #4
samples(4) = 3      ‘ Number of defects for defect category #5

' Add the new sample subgroup to the chart
 Me.ChartData.AddNewSampleRecord(timestamp, samples)
```

While the table portion of the display can display defect data broken down into categories, only the sum of the defects for a given sample subgroup is used in creating the actual SPC chart. Note that the code below, extracted from the TimeAttributeControlCharts.NumberDefectsControlChart example, uses a different **ChartData.SimulateDefectRecord** method to simulate the defect data.

Chart Header Information, Measured Data and Calculated Value Table

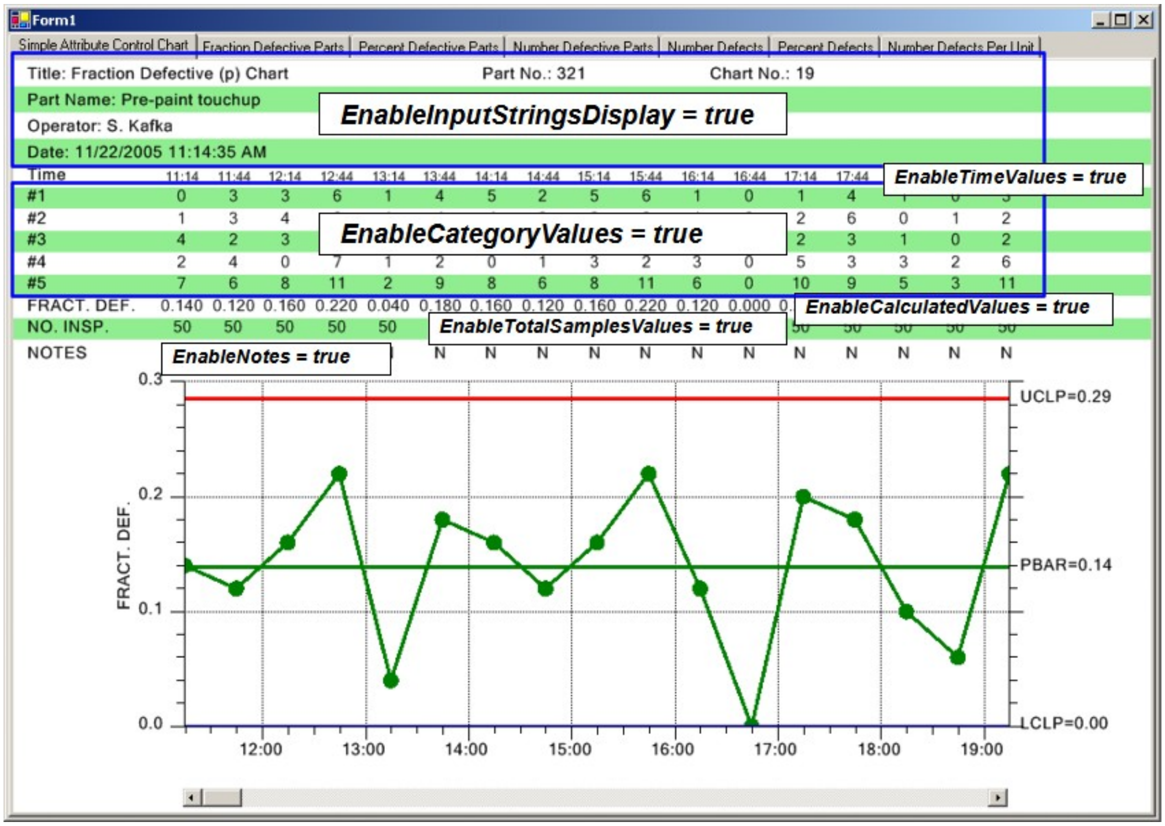
Standard worksheets used to gather and plot SPC data consist of three main parts.

- The first part is the header section, identifying the title of the chart, the monitored process, the machine operator, part number and other important information specific to the chart.
- The second part is the measurement data recording and calculation section, organized as a table recording the sample data and calculated values in a neat, readable fashion.
- The third part plots the calculated SPC values as a SPC chart.

The chart includes options that enable the programmer to customize and automatically include header information along with a table of the measurement and calculated data, in the SPC chart.

The following properties enable sections of the chart header and table:

EnableInputStringsDisplay
EnableCategoryValues
EnableCalculatedValues
EnableTotalSamplesValues
EnableNotes
EnableTimeValues



Title: Fraction Defective (p) Chart   Part No.: 321   Chart No.: 19

Part Name: Pre-paint touchup

Operator: S. Kafka

Date: 11/22/2005 11:14:35 AM

Time	11:14	11:44	12:14	12:44	13:14	13:44	14:14	14:44	15:14	15:44	16:14	16:44	17:14	17:44	[illegible]	[illegible]	[illegible]
#1	0	3	3	6	1	4	5	2	5	6	1	0	1	4	[illegible]	[illegible]	[illegible]
#2	1	3	4	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	2	6	0	1	2
#3	4	2	3	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	2	3	1	0	2
#4	2	4	0	7	1	2	0	1	3	2	3	0	5	3	3	2	6
#5	7	6	8	11	2	9	8	6	8	11	6	0	10	9	5	3	11
FRACT. DEF.	0.140	0.120	0.160	0.220	0.040	0.180	0.160	0.120	0.160	0.220	0.120	0.000	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
NO. INSP.	50	50	50	50	50	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	50	50	50	50	50
NOTES	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	N	N	N	N	N	N	N	N	N	N	N	N

The example code below is extracted from the **TimeAttributeControlCharts.SimpleAttributeControlChart** example program.

[C#]

```
void InitializeChart()
{
...

    // Set the strings used in the header section of the table
    this.ChartData.Title = "Fraction Defective (p) Chart";
    this.ChartData.PartNumber = "321";
    this.ChartData.ChartNumber="19";
    this.ChartData.PartName= "Pre-paint touchup";
    this.ChartData.TheOperator="S. Kafka";

    // Display the Sampled value rows of the table
    this.EnableInputStringsDisplay= true;
    // Display the Sampled value rows of the table
    this.EnableCategoryValues= true;
    // Display the Calculated value rows of the table
    this.EnableCalculatedValues= true;
    // Display the total samples per subgroup value row
    this.EnableTotalSamplesValues= true;
    // Display the Notes row of the table
    this.EnableNotes= true;
    // Display the time stamp row of the table
    this.EnableTimeValues = true;
    .
    .
```

EWM<A

```
    .
    this.RebuildChartUsingCurrentData();
}
```

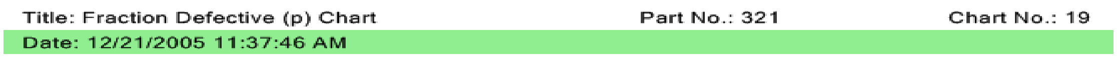
[VB]

```
Sub InitializeChart()
...

   ' Set the strings used in the header section of the table
   Me.ChartData.Title = "Fraction Defective (p) Chart"
   Me.ChartData.PartNumber = "321"
   Me.ChartData.ChartNumber = "19"
   Me.ChartData.PartName = "Pre-paint touchup"
   Me.ChartData.TheOperator = "S. Kafka"

   ' Display the Sampled value rows of the table
   Me.EnableInputStringsDisplay = True
   ' Display the Sampled value rows of the table
   Me.EnableCategoryValues = True
   ' Display the Calculated value rows of the table
   Me.EnableCalculatedValues = True
   ' Display the total samples per subgroup value row
   Me.EnableTotalSamplesValues = True
   ' Display the Notes row of the table
   Me.EnableNotes = True
   ' Display the time stamp row of the table
   Me.EnableTimeValues = True

   Me.RebuildChartUsingCurrentData()
End Sub 'InitializeChart
```

The input header strings display has four sub-levels that display increasing levels of information. The input header strings display level is set using the charts HeaderStringsLevel property. Strings that can be displayed are: Title, PartNumber, ChartNumber, DateString, PartName, Operation, Machine, SpecificationLimits, Gage, UnitOfMeasure, ZeroEquals and DateString. The four levels and the information displayed is listed below:

HEADER_STRINGS_LEVEL0	Display no header information
HEADER_STRINGS_LEVEL1	Display minimal header information: Title, PartNumber, ChartNumber, DateString
HEADER_STRINGS_LEVEL2	Display most header strings: Title, PartNumber, ChartNumber, PartName, Operation, Operator, Machine, DateString
HEADER_STRINGS_LEVEL3	Display all header strings: Title, PartNumber, ChartNumber, DateString, PartName, Operation, Machine, SpecificationLimits, Gage, UnitOfMeasure, ZeroEquals and DateString

The example program TimeAttributeControlCharts.SimpleAttributeControlChart demonstrates the use of the **HeaderStringsLevel** property. The example below displays a minimum set of header strings (HeaderStringsLevel = SPCControlChartData.HEADER_STRINGS_LEVEL1).



Title: Fraction Defective (p) Chart	Part No.: 321	Chart No.: 19
Date: 12/21/2005 11:37:46 AM		

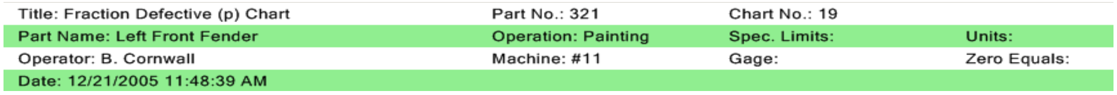
[C#]

```
// Set the strings used in the header section of the table
this.ChartData.Title = "Fraction Defective (p) Chart";
this.ChartData.PartNumber = "321";
this.ChartData.ChartNumber="19";
this.ChartData.DateString = DateTime.Now.ToString();
this.HeaderStringsLevel = SPCControlChartData.HEADER_STRINGS_LEVEL1;
```

[VB]

```
' Set the strings used in the header section of the table
Me.ChartData.Title = "Fraction Defective (p) Chart"
Me.ChartData.PartNumber = "321"
Me.ChartData.ChartNumber = "19"
Me.ChartData.DateString = DateTime.Now.ToString()
Me.HeaderStringsLevel = SPCControlChartData.HEADER_STRINGS_LEVEL1
```

The example below displays a maximum set of header strings (HeaderStringsLevel = SPCControlChartData.HEADER_STRINGS_LEVEL3).



Title: Fraction Defective (p) Chart	Part No.: 321	Chart No.: 19	
Part Name: Left Front Fender	Operation: Painting	Spec. Limits:	Units:
Operator: B. Cornwall	Machine: #11	Gage:	Zero Equals:
Date: 12/21/2005 11:48:39 AM			

[C#]

```
// Set the strings used in the header section of the table
this.ChartData.Title = "Fraction Defective (p) Chart";
this.ChartData.PartNumber = "283501";
this.ChartData.ChartNumber="17";
this.ChartData.TheOperator="B. Cornwall";
this.ChartData.PartName= "Left Front Fender";
this.ChartData.Operation = "Painting";
this.ChartData.SpecificationLimits="";
this.ChartData.Machine="#11";
this.ChartData.Gage="";
this.ChartData.UnitOfMeasure = "";
this.ChartData.ZeroEquals="";
this.ChartData.DateString = DateTime.Now.ToString();
this.HeaderStringsLevel = SPCControlChartData.HEADER_STRINGS_LEVEL3;
```

[VB]

```
' Set the strings used in the header section of the table
Me.ChartData.Title = "Fraction Defective (p) Chart"
Me.ChartData.PartNumber = "283501"
Me.ChartData.ChartNumber = "17"
Me.ChartData.TheOperator = "B. Cornwall"
Me.ChartData.PartName = "Left Front Fender"
Me.ChartData.Operation = "Painting"
Me.ChartData.SpecificationLimits = ""
Me.ChartData.Machine = "#11"
Me.ChartData.Gage = ""
Me.ChartData.UnitOfMeasure = ""
Me.ChartData.ZeroEquals = ""
Me.ChartData.DateString = DateTime.Now.ToString()
Me.HeaderStringsLevel = SPCControlChartData.HEADER_STRINGS_LEVEL3
```

The identifying string displayed in front of the input header string can be any string that you want, including non-English language string. For example, if you want the input header string for the Title to represent a project name:

Project Name: Project XKYZ for PerQuet

Set the properties:

[C#]

```
this.ChartData.Title = "Project XKYZ for PerQuet";
this.ChartData.TitleHeader = "Project Name:";
```

[VB]

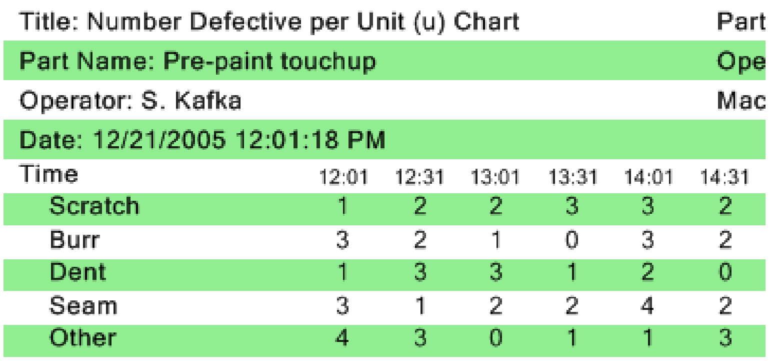
```
Me.ChartData.Title = "Project XKYZ for PerQuet"
Me.ChartData.TitleHeader = "Project Name:"
```

Change other headers using the ChartData properties listed below.

- TitleHeader
- PartNumberHeader
- ChartNumberHeader
- PartNameHeader
- OperationHeader
- OperatorHeader
- MachineHeader
- DateHeader
- SpecificationLimitsHeader
- GageHeader
- UnitOfMeasureHeader
- ZeroEqualsHeader
- NotesHeader

Even though the input header string properties have names like Title, PartNumber, ChartNumber, etc., those names are arbitrary. They are really just placeholders for the strings that are placed at the respective position in the table. You can display any combination of strings that you want, rather than the ones we have selected by default, based on commonly used standardized SPC Control Charts.

Depending on the control chart type, you may want to customize the category header strings. In most of our examples, we use the category header strings: Scratch, Burr, Dent, Seam, and Other, to represent common defect categories. You can change these strings to anything that you want using the **ChartData.SetSampleRowHeaderString** method. See the example program TimeAttributeControlCharts.NumberDefectsControlChart.



Title: Number Defective per Unit (u) Chart Part

Part Name: Pre-paint touchup Ope

Operator: S. Kafka Mac

Date: 12/21/2005 12:01:18 PM

Time	12:01	12:31	13:01	13:31	14:01	14:31
Scratch	1	2	2	3	3	2
Burr	3	2	1	0	3	2
Dent	1	3	3	1	2	0
Seam	3	1	2	2	4	2
Other	4	3	0	1	1	3

[C#]

```
// Set the table row headers strings for defect categories
this.ChartData.SetSampleRowHeaderString(0, "    Scratch");
this.ChartData.SetSampleRowHeaderString(1, "    Burr");
this.ChartData.SetSampleRowHeaderString(2, "    Dent");
this.ChartData.SetSampleRowHeaderString(3, "    Seam");
this.ChartData.SetSampleRowHeaderString(4, "    Other");
```

[VB]

```
' Set the table row headers strings for defect categories
Me.ChartData.SetSampleRowHeaderString(0, "    Scratch")
Me.ChartData.SetSampleRowHeaderString(1, "    Burr")
Me.ChartData.SetSampleRowHeaderString(2, "    Dent")
Me.ChartData.SetSampleRowHeaderString(3, "    Seam")
Me.ChartData.SetSampleRowHeaderString(4, "    Other")
```

The **ChartTable** property of the chart has properties that further customize the chart. The default table background uses the accounting style green-bar striped background. You can change this using the **ChartTable.TableBackgroundMode** property**.** Set the value to one of the TableBackgroundMode constants:

TABLE_NO_COLOR_BACKGROUND Constant specifies that the table does not use a background color.

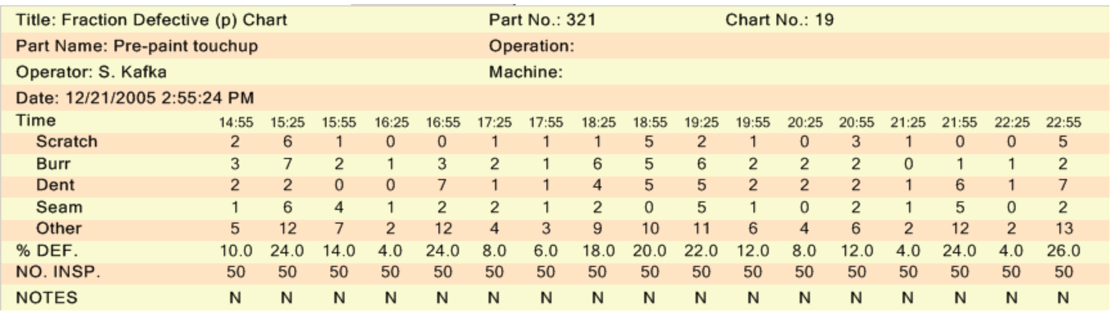
TABLE_SINGLE_COLOR_BACKGROUND Constant specifies that the table uses a single color for the background (backgroundColor1)

TABLE_STRIPED_COLOR_BACKGROUND Constant specifies that the table uses horizontal stripes of color for the background (backgroundColor1 and backgroundColor2)

TABLE_SINGLE_COLOR_BACKGROUND_GRIDCELL
Constant specifies that the table uses a grid background, with backgroundColor1 the overall

background color and backgroundColor2 the color of the grid lines.

Extracted from the TimeAttributeControlCharts.PercentDefectivePartsControlChart example program



Title: Fraction Defective (p) Chart   Part No.: 321   Chart No.: 19
Part Name: Pre-paint touchup   Operation:
Operator: S. Kafka   Machine:
Date: 12/21/2005 2:55:24 PM

Time	14:55	15:25	15:55	16:25	16:55	17:25	17:55	18:25	18:55	19:25	19:55	20:25	20:55	21:25	21:55	22:25	22:55
Scratch	2	6	1	0	0	1	1	1	5	2	1	0	3	1	0	0	5
Burr	3	7	2	1	3	2	1	6	5	6	2	2	2	0	1	1	2
Dent	2	2	0	0	7	1	1	4	5	5	2	2	2	1	6	1	7
Seam	1	6	4	1	2	2	1	2	0	5	1	0	2	1	5	0	2
Other	5	12	7	2	12	4	3	9	10	11	6	4	6	2	12	2	13
% DEF.	10.0	24.0	14.0	4.0	24.0	8.0	6.0	18.0	20.0	22.0	12.0	8.0	12.0	4.0	24.0	4.0	26.0
NO. INSP.	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
NOTES	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N

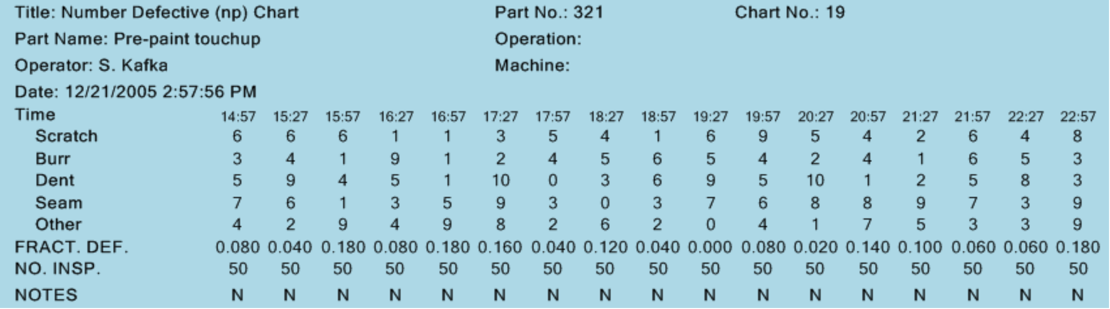
[C#]

```
this.ChartTable.TableBackgroundMode =
    SPCGeneralizedTableDisplay.TABLE_STRIPED_COLOR_BACKGROUND;
this.ChartTable.BackgroundColor1 = Color.Bisque;
this.ChartTable.BackgroundColor2 = Color.LightGoldenrodYellow;
```

[VB]

```
Me.ChartTable.TableBackgroundMode = _
    SPCGeneralizedTableDisplay.TABLE_STRIPED_COLOR_BACKGROUND
Me.ChartTable.BackgroundColor1 = Color.Bisque
Me.ChartTable.BackgroundColor2 = Color.LightGoldenrodYellow
```

Extracted from the TimeAttributeControlCharts.NumberDefectivePartsControlChart example program



Title: Number Defective (np) Chart   Part No.: 321   Chart No.: 19
Part Name: Pre-paint touchup   Operation:
Operator: S. Kafka   Machine:
Date: 12/21/2005 2:57:56 PM

Time	14:57	15:27	15:57	16:27	16:57	17:27	17:57	18:27	18:57	19:27	19:57	20:27	20:57	21:27	21:57	22:27	22:57
Scratch	6	6	6	1	1	3	5	4	1	6	9	5	4	2	6	4	8
Burr	3	4	1	9	1	2	4	5	6	5	4	2	4	1	6	5	3
Dent	5	9	4	5	1	10	0	3	6	9	5	10	1	2	5	8	3
Seam	7	6	1	3	5	9	3	0	3	7	6	8	8	9	7	3	9
Other	4	2	9	4	9	8	2	6	2	0	4	1	7	5	3	3	9
FRACT. DEF.	0.080	0.040	0.180	0.080	0.180	0.160	0.040	0.120	0.040	0.000	0.080	0.020	0.140	0.100	0.060	0.060	0.180
NO. INSP.	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
NOTES	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N

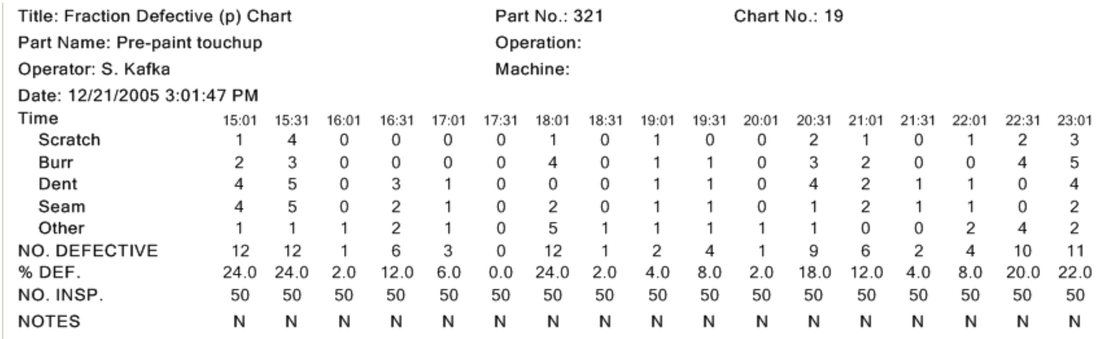
[C#]

```
this.ChartTable.TableBackgroundMode =
    SPCGeneralizedTableDisplay.TABLE_SINGLE_COLOR_BACKGROUND;
this.ChartTable.BackgroundColor1 = Color.LightBlue;
```

[VB]

```
Me.ChartTable.TableBackgroundMode = _
    SPCGeneralizedTableDisplay.TABLE_SINGLE_COLOR_BACKGROUND
Me.ChartTable.BackgroundColor1 = Color.LightBlue
```

Extracted from the TimeAttributeControlCharts.NumberDefectivePartsControlChart example program

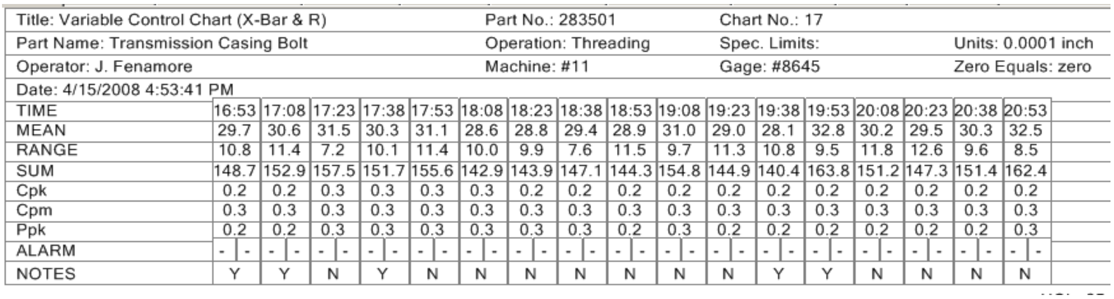


Title: Fraction Defective (p) Chart   Part No.: 321   Chart No.: 19
Part Name: Pre-paint touchup   Operation:
Operator: S. Kafka   Machine:
Date: 12/21/2005 3:01:47 PM

Time	15:01	15:31	16:01	16:31	17:01	17:31	18:01	18:31	19:01	19:31	20:01	20:31	21:01	21:31	22:01	22:31	23:01
Scratch	1	4	0	0	0	0	1	0	1	0	0	2	1	0	1	2	3
Burr	2	3	0	0	0	0	4	0	1	1	0	3	2	0	0	4	5
Dent	4	5	0	3	1	0	0	0	1	1	0	4	2	1	1	0	4
Seam	4	5	0	2	1	0	2	0	1	1	0	1	2	1	1	0	2
Other	1	1	1	2	1	0	5	1	1	1	1	1	0	0	2	4	2
NO. DEFECTIVE	12	12	1	6	3	0	12	1	2	4	1	9	6	2	4	10	11
% DEF.	24.0	24.0	2.0	12.0	6.0	0.0	24.0	2.0	4.0	8.0	2.0	18.0	12.0	4.0	8.0	20.0	22.0
NO. INSP.	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
NOTES	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N

[C#]
```
this.ChartTable.TableBackgroundMode =
        SPCGeneralizedTableDisplay.TABLE_NO_COLOR_BACKGROUND;
```

[VB]
```
Me.ChartTable.TableBackgroundMode = _
    SPCGeneralizedTableDisplay.TABLE_NO_COLOR_BACKGROUND
```



Title: Variable Control Chart (X-Bar & R)   Part No.: 283501   Chart No.: 17
Part Name: Transmission Casing Bolt   Operation: Threading   Spec. Limits: Units: 0.0001 inch
Operator: J. Fenamore   Machine: #11   Gage: #8645   Zero Equals: zero
Date: 4/15/2008 4:53:41 PM

TIME	16:53	17:08	17:23	17:38	17:53	18:08	18:23	18:38	18:53	19:08	19:23	19:38	19:53	20:08	20:23	20:38	20:53
MEAN	29.7	30.6	31.5	30.3	31.1	28.6	28.8	29.4	28.9	31.0	29.0	28.1	32.8	30.2	29.5	30.3	32.5
RANGE	10.8	11.4	7.2	10.1	11.4	10.0	9.9	7.6	11.5	9.7	11.3	10.8	9.5	11.8	12.6	9.6	8.5
SUM	148.7	152.9	157.5	151.7	155.6	142.9	143.9	147.1	144.3	154.8	144.9	140.4	163.8	151.2	147.3	151.4	162.4
Cpk	0.2	0.2	0.3	0.3	0.3	0.3	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2
Cpm	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3
Ppk	0.2	0.2	0.3	0.3	0.3	0.3	0.3	0.3	0.2	0.3	0.2	0.2	0.2	0.2	0.2	0.2	0.3
ALARM	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -
NOTES	Y	Y	N	Y	N	N	N	N	N	N	N	Y	Y	N	N	N	N

[C#]
```
this.ChartTable.TableBackgroundMode =
    SPCGeneralizedTableDisplay.TABLE_SINGLE_COLOR_BACKGROUND_GRIDCELL;
this.ChartTable.BackgroundColor1 = Color.White;
this.ChartTable.BackgroundColor2 = Color.Gray;
```

[VB]
```
Me.ChartTable.TableBackgroundMode =
SPCGeneralizedTableDisplay.TABLE_SINGLE_COLOR_BACKGROUND_GRIDCELL
Me.ChartTable.BackgroundColor1 = Color.White
Me.ChartTable.BackgroundColor2 = Color.Gray
```

Table and Chart Fonts

There are a large number of fonts that you have control over, both the fonts in the table and the fonts in the chart. The programmer can select a default font (as in the case of non-US character set), or select individual fonts for different elements of the table and charts.

Table Fonts

The table fonts are accessed through the charts **ChartTable** property. Below is a list of accessible table fonts:

TimeLabelFont The font used in the display of time values in the table.
SampleLabelFont The font used in the display of sample numeric values in the table.
CalculatedLabelFont The font used in the display of calculated values in the table.
StringLabelFont The font used in the display of header string values in the table.
NotesLabelFont The font used in the display of notes values in the table.

Extracted from the example
BatchAttributeControlCharts.PercentDefectivePartsControlChart

[C#]
```
this.ChartTable.SampleLabelFont = new Font("Times", 12, FontStyle.Regular);
```

[VB]
```
Me.ChartTable.SampleLabelFont = new Font("Times", 12, FontStyle.Regular)
```

The **ChartTable** class has a static property, **SPCGeneralizedTableDisplay.DefaultTableFont**, that sets the default Font. Use this if you want to establish a default font for all of the text in a table. This static property must be set BEFORE the charts **Init** routine.

Extracted from the example
BatchAttributeControlCharts.PercentDefectivePartsControlChart

[C#]
```
SPCGeneralizedTableDisplay.DefaultTableFont =
    new Font("Microsoft Sans Serif", 11, FontStyle.Regular);
// Initialize the SPCBatchVariableControlChart
this.InitSPCBatchAttributeControlChart(charttype, numcategories,
               numsamplespersubgroup, numdatapointsinview);
.
.
.
```

[VB]
```
SPCGeneralizedTableDisplay.DefaultTableFont = _
    new Font("Microsoft Sans Serif", 11, FontStyle.Regular)
‘ Initialize the SPCBatchAttrbiuteControlChart
Me.InitSPCBatchAttributeControlChart(charttype, numcategories, _
               numsamplespersubgroup, numdatapointsinview);
.
.
.
```

Chart Fonts

There are default chart fonts that are static objects in the **SPCChartObjects** class. They establish the default fonts for related chart objects and if you change them they need to be set before the first charts Init.. call. Since these properties are static, any changes to them will apply to the program as a whole, not just the immediate class.

AxisLabelFont The font used to label the x- and y- axes.
AxisTitleFont The font used for the axes titles.
HeaderFont The font used for the chart title.
SubheadFont The font used for the chart subhead.
ToolTipFont The tool tip font.

AnnotationFont	The annotation font.
ControlLimitLabelFont	The font used to label the control limits

Extracted from the example TimeAttributeControlCharts.PercentDefectiveChart

[C#]
```
SPCChartObjects.AxisTitleFont = new Font("Times", 12, FontStyle.Regular);
SPCChartObjects.ControlLimitLabelFont = new Font("Times", 10, FontStyle.Regular);

this.InitSPCTimeAttributeControlChart(charttype, numcategories,
               numsamplespersubgroup, numdatapointsinview, timeincrementminutes);
```

[VB]
```
SPCChartObjects.AxisTitleFont = new Font("Times", 12, FontStyle.Regular)
SPCChartObjects.ControlLimitLabelFont = new Font("Times", 10, FontStyle.Regular)

Me.InitSPCTimeAttributeControlChart(charttype, numcategories, _
               numsamplespersubgroup, numdatapointsinview, timeincrementminutes)
```

The chart class static property, **DefaultTableFont**, sets the default Font string. Since the chart fonts all default to different sizes, the default font is defined using a string specifying the name of the font. This static property must be set BEFORE the charts **Init** routine.

Extracted from the example Extracted from the example TimeAttributeControlCharts.PercentDefectiveChart

[C#]
```
PercentDefectivePartsControlChart.DefaultChartFontString = "Times";

this.InitSPCTimeAttributeControlChart(charttype, numcategories,
               numsamplespersubgroup, numdatapointsinview, timeincrementminutes);
.
.
.
```

[VB]
```
PercentDefectivePartsControlChart.DefaultChartFontString = "Times"
Me.InitSPCTimeAttributeControlChart(charttype, numcategories, _
               numsamplespersubgroup, numdatapointsinview, timeincrementminutes)
.
.
```

These static properties establish the default fonts for a group of objects as a whole. For example, all charts will have the same x- and y-axis label fonts. You can still change the individual fonts for an individual object in a specific chart. For example, if in the Primary Chart you want the x-axis label font to be size 10, and the y-axis label font to be size 14, you can set them individually after the charts Init.. method has been called.

[C#]
```
this.InitSPCTimeAttributeControlChart(charttype, numcategories,
               numsamplespersubgroup, numdatapointsinview, timeincrementminutes);
.
.
this.PrimaryChart.XAxisLab.TextFont = new Font("Times", 10, FontStyle.Regular);
this.PrimaryChart.YAxisLab.TextFont = new Font("Times", 14, FontStyle.Regular);
```

EWM<

[VB]

```
Me.InitSPCTimeAttributeControlChart(charttype, numcategories,
               numsamplespersubgroup, numdatapointsinview, timeincrementminutes)
.
.
Me.PrimaryChart.XAxisLab.TextFont = new Font("Times", 10, FontStyle.Regular)
Me.PrimaryChart.YAxisLab.TextFont = new Font("Times", 14, FontStyle.Regular)
```

Table and Chart Templates

All of the strings displayed in the table and charts use a template unique to the string type. Numeric strings use a **NumericLabel** template, time/date strings use a time **TimeLabel** template, and so on. These templates permit the programmer to customize the display of the strings. The various templates are listed below:

SPCChartObjects (Accessed in the charts PrimaryChart and SecondaryChart properties)

Property	**Type**	**Description**
XValueTemplate	NumericLabel	The x-value template for the data tooltip.
YValueTemplate	NumericLabel	The y-value template for the data tooltip.
XTimeValueTemplate	TimeLabel	x-value template for the data tooltip.
TextTemplate	ChartText	The text template for the data tooltip.

SPCGeneralizedTableDisplay (Accessed in the charts ChartTable property)

Property	**Type**	**Description**
TimeItemTemplate	TimeLabel	The TimeLabel object used as a template for displaying time values in the table.
SampleItemTemplate	NumericLabel	The NumericLabel object used as a template for displaying the sample values in the table.
CalculatedItemTemplate	NumericLabel	The NumericLabel object used as a template for displaying calculated values in the table.
StringItemTemplate	StringLabel	The StringLabel object used as a template for displaying string values in the table.
NotesItemTemplate	NotesLabel	The NotesLabel object used as a template for displaying string values in the table.

The most common use for these templates is to set the color attributes of a class of objects, or the decimal precision of a numeric string.

```
this.ChartTable.SampleItemTemplate.LineColor = Color.Red;
```

Chart Position

If the SPC chart does not include frequency histograms on the left (they take up about 20% of the available chart width), you can adjust the left and right edges of the chart using the **GraphStartPosX** and **GraphStopPlotX** properties to allow for more room in the display of the data. This also affects the table layout, because the table columns must line up with the chart data points.

[C#]

```
this.GraphStartPosX = 0.1; // start here
this.GraphStopPosX = 0.875;  // end here
```

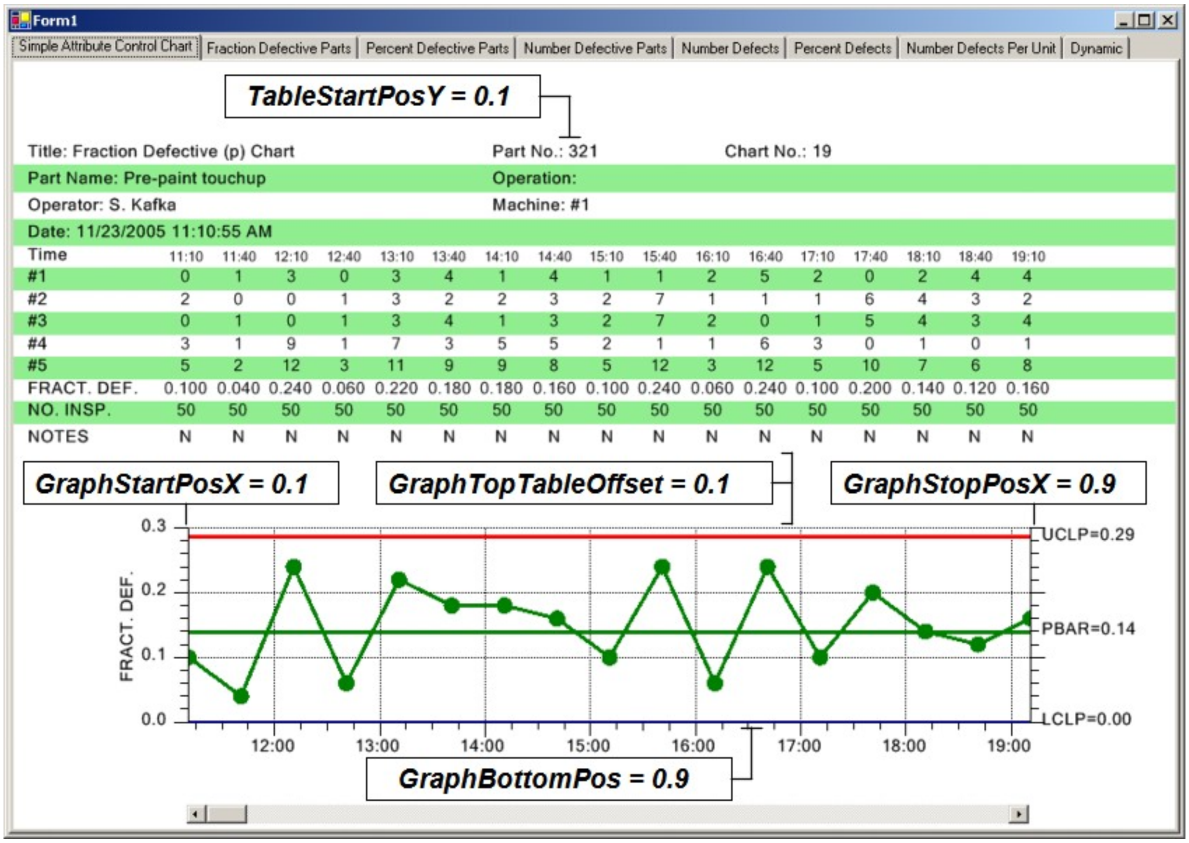
[VB]

```
Me.GraphStartPosX = 0.1 ' start here
Me.GraphStopPosX = 0.875  ' end here
```

There is not much flexibility positioning the top and bottom of the chart. Depending on the table items enabled, the table starts at the position defined by the **TableStartPosY** property, and continues until all of the table items are displayed. It then offsets the top of the primary chart with respect to the bottom of the table by the value of the property **GraphTopTableOffset**. The value of the property **GraphBottomPos** defines the bottom of the graph. The default values for these properties are:

```
TableStartPosY = 0.00
GraphTopTableOffset = 0.02
GraphBottomPos = 0.925
```

The picture below uses different values for these properties in order to emphasize the affect that these properties have on the resulting chart.



Title: Fraction Defective (p) Chart   Part No.: 321   Chart No.: 19
Part Name: Pre-paint touchup   Operation:
Operator: S. Kafka   Machine: #1
Date: 11/23/2005 11:10:55 AM

Time	11:10	11:40	12:10	12:40	13:10	13:40	14:10	14:40	15:10	15:40	16:10	16:40	17:10	17:40	18:10	18:40	19:10
#1	0	1	3	0	3	4	1	4	1	1	2	5	2	0	2	4	4
#2	2	0	0	1	3	2	2	3	2	7	1	1	1	6	4	3	2
#3	0	1	0	1	3	4	1	3	2	7	2	0	1	5	4	3	4
#4	3	1	9	1	7	3	5	5	2	1	1	6	3	0	1	0	1
#5	5	2	12	3	11	9	9	8	5	12	3	12	5	10	7	6	8
FRACT. DEF.	0.100	0.040	0.240	0.060	0.220	0.180	0.180	0.160	0.100	0.240	0.060	0.240	0.100	0.200	0.140	0.120	0.160
NO. INSP.	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
NOTES	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N

SPC Control Limits

There are two methods you can use to set the SPC control limit for a chart. The first method explicitly sets the limits to values that you calculate on your own, because of some analysis that a quality engineer does on previously collected data. The second method auto-calculates the limits using the algorithms supplied in this software.

The quick way to set the limit values and limit strings is to use the charts **ChartData.SetControlLimitValues** and **ChartData.SetControlLimitStrings** methods. This method only works for the default +-3-sigma level control limits, and not any others you may have added using the charts **AddAdditionalControlLimit** method discussed in the *Multiple Control Limits* section. The data values in the *controllimitvalues* and *controllimitstrings* arrays used to pass the control limit information must be sorted in the following order:

[SPC_PRIMARY_CONTROL_TARGET,
SPC_PRIMARY_LOWER_CONTROL_LIMIT,
SPC_PRIMARY_UPPER_CONTROL_LIMIT]

[C#]

```
double [] controllimitvalues = {0.13, 0.0, 0.25};
this.ChartData.SetControlLimitValues(controllimitvalues);

string [] controllimitstrings = {"PBAR","LCL", "UCL”};
```

```
this.ChartData.SetControlLimitStrings(controllimitstrings);
```

[VB]

```
Dim controllimitvalues() As Double = {0.13, 0.0, 0.25}
Me.ChartData.SetControlLimitValues(controllimitvalues)

Dim controllimitstrings() As String = {"PBAR", "LCL", "UCL"}
Me.ChartData.SetControlLimitStrings(controllimitstrings)
```

You can also set the control limit values and control limit text one value at a time using the **ChartData.SetControlLimitValue** and **ChartData.SetControlLimitString** methods.

A more complicated way to set the control limits explicitly is to first grab a reference to the **SPCControlLimitRecord** for a given control limit, and then change the value of that control limit, and the control limit text, if desired. The example below sets the control limit values and text for the three control limits (target value, upper control limit, and lower control limit) of the primary chart, and the three control limit values for the secondary chart.

[C#]

```
// Set control limits for primary chart

//target control limit primary chart
SPCControlLimitRecord primarytarget =
    ChartData.GetControlLimitRecord(SPCControlChartData.SPC_PRIMARY_CONTROL_TARGET);
primarytarget.ControlLimitValue = 0.13;
primarytarget.ControlLimitText = "PBAR";

//lower control limit primary chart
SPCControlLimitRecord primarylowercontrollimit =
ChartData.GetControlLimitRecord(SPCControlChartData.SPC_PRIMARY_LOWER_CONTROL_LIMIT);
primarylowercontrollimit.ControlLimitValue = 0.0;
primarylowercontrollimit.ControlLimitText = "LCL";

//upper control limit primary chart
SPCControlLimitRecord primaryuppercontrollimit =
    ChartData.GetControlLimitRecord(SPCControlChartData.SPC_PRIMARY_UPPER_CONTROL_LIMIT);
primaryuppercontrollimit.ControlLimitValue = 0.25;
primaryuppercontrollimit.ControlLimitText = "UCL";
```

[VB]

```
' Set control limits for primary chart
'target control limit primary chart
Dim primarytarget As SPCControlLimitRecord = _
  ChartData.GetControlLimitRecord(SPCControlChartData.SPC_PRIMARY_CONTROL_TARGET)
primarytarget.ControlLimitValue = 0.13
primarytarget.ControlLimitText = "PBAR"

'lower control limit primary chart
Dim primarylowercontrollimit As SPCControlLimitRecord = _

ChartData.GetControlLimitRecord(SPCControlChartData.SPC_PRIMARY_LOWER_CONTROL_LIMI
T)
primarylowercontrollimit.ControlLimitValue = 0.0
primarylowercontrollimit.ControlLimitText = "LCL"

'upper control limit primary chart
Dim primaryuppercontrollimit As SPCControlLimitRecord = _

ChartData.GetControlLimitRecord(SPCControlChartData.SPC_PRIMARY_UPPER_CONTROL_LIMI
T)
primaryuppercontrollimit.ControlLimitValue = 0.25
primaryuppercontrollimit.ControlLimitText = "UCL"
```

The second way to set the control limits is to call the **AutoCalculateControlLimits** method. You must have already added a collection of sampled data values to the charts **ChartData** SPC data object before you can call this method, since the method uses the internal **ChartData** object to provide the historical values needed in the calculation.

[C#]
```
// Must have data loaded before any of the Auto.. methods are called
SimulateData();

// Calculate the SPC control limits for both graphs of the current SPC

this.AutoCalculateControlLimits();
```

[VB]
```
‘ Must have data loaded before any of the Auto.. methods are called
SimulateData()

‘ Calculate the SPC control limits for both graphs of the current SPC

Me.AutoCalculateControlLimits()
```

You can add data to the **ChartData** object, auto-calculate the control limits to establish the SPC control limits, and then continue to add new data values. Alternatively, you can set the SPC control limits explicitly as the result of previous runs, using the previously described **ChartData.SetControlLimitValues** method, and add new sampled data values to the **ChartData** object, and after a certain number of updates call the **AutoCalculateControlLimits** method to establish new control limits.

[C#]
```
updateCount++;
this.ChartData.AddNewSampleRecord(timestamp, samples);
if (updateCount > 50) // After 50 sample groups and calculate limits on the fly
{
// Calculate the SPC control limits for the X-Bar part of the current SPC chart
    this.AutoCalculateControlLimits();
    // Scale the y-axis of the X-Bar chart to display all data and control limits
    this.AutoScalePrimaryChartYRange();
}
```

[VB]
```
updateCount += 1
Me.ChartData.AddNewSampleRecord(timestamp, samples)
If updateCount > 50 Then ' After 50 sample groups and calculate limits on the fly
   ' Calculate the SPC control limits for the X-Bar part of the current SPC chart
   Me.AutoCalculateControlLimits()
   ' Scale the y-axis of the X-Bar chart to display all data and control limits
   Me.AutoScalePrimaryChartYRange()
End If
```

Need to exclude records from the control limit calculation? Call the **ChartData.ExcludeRecordFromControlLimitCalculations** method, passing in true to exclude the record.

[C#]

```
for (int i=0; i < 10; i++)
      this.ChartData.ExcludeRecordFromControlLimitCalculations(i,true);
```

[VB]

```
Dim i As Integer
For i = 0 To 9
   Me.ChartData.ExcludeRecordFromControlLimitCalculations(i, True)
Next i
```

Formulas Used in Calculating Control Limits for Attribute Control Charts

The SPC control limit formulas used in the software derive from the following source:

Fraction Defective Parts, Number Defective Parts, Number Defects, Number Defects Per Unit - "Introduction to Statistical Quality Control" by Douglas C. Montgomery, John Wiley and Sons, Inc. 2001.

Percent Defective Parts - "SPC Simplified – Practical Steps to Quality" by Robert T. Amsden, Productivity Inc., 1998.

SPC Control Chart Nomenclature

UCL = Upper Control Limit

LCL = Lower Control Limit

Center line = The target value for the process

p = estimate (or average) of the fraction defective (or non-conforming) parts

P = estimate (or average) of the percent defective (or non-conforming) parts

c = estimate (or average) of the number of defects (or nonconformities)

u = estimate (or average) of the number of defects (or nonconformities) per unit

n = number of samples per subgroup

dopu = defect opportunities per unit (applies only the DPMO chart)

dpmo = defects per million opportunities (applies only the DPMO chart)
calculated as: dpmo = (1,000,000 * numberOfDefects) / (sampleSize * dopu)

up = estimate (or average) of the dpmo values

EWM<

Fraction Defective Parts – Also known as Fraction Non-Conforming or p-chart

UCL = p + 3 * Sqrt (p * (1- p) / n)

Center line = p

LCL = p - 3 * Sqrt (p * (1- p) / n)

Percent Defective Parts – Also known as Percent Non-Conforming or p-chart

UCL = p + 3 * Sqrt (p * (100% - p) / n)

Center line = p

LCL = p - 3 * Sqrt (p * (100% - p) / n)

Number of Defective Parts – Also known as the Number Nonconforming or np-chart

UCL = (n * p) + 3 * Sqrt ((n * p) * (1- p) / n)

Center line = (n * p)

LCL = (n * p) - 3 * Sqrt ((n * p) * (1- p) / n)

In this case the value (n * p) represents the average number of defective parts per sample subgroup. Since p is the estimate (or average) of the fraction defective per sample subgroup, n * p is the average number of defective per sample subgroup. Or you can add up all the number defective parts in all subgroups and divide by the number of subgroups, that to will reduce to the average number of defective per sample subgroup

Number Defects Per Million – Also known as DPMO

UCL = up + 3000 * Sqrt (up /(dopu * n))

Center line = up

LCL = up - 3000 * Sqrt (up/(dopu * n))

Number of Defects Control Chart – Also known as Number Nonconformities or c-chart

UCL = c + 3 * Sqrt (c)

Center line = c

LCL = c - 3 * Sqrt (c)

Number of Defects per Unit Control Chart – Also known as Number Nonconformities per Unit or u-chart

UCL = u + 3 * Sqrt (u / n)

Center line = u

LCL = u - 3 * Sqrt (u / n)

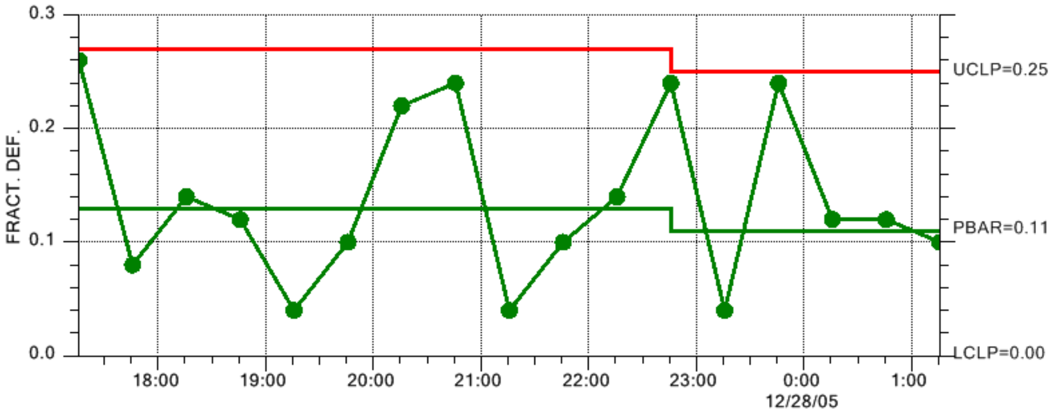
Variable SPC Control Limits

There can be situations where the SPC control limit changes in a chart. If your control limits change, you need to set the following **ControlLineMode** property to SPCChartObjects.CONTROL_LINE_VARIABLE, as in the example below. The default value is SPCChartObjects.CONTROL_LINE_FIXED.

```
this.PrimaryChart.ControlLineMode = SPCChartObjects.CONTROL_LINE_VARIABLE;
```

EWM<A

In the SPCChartObjects.CONTROL_LINE_FIXED case, the current SPC control limit plots as a horizontal straight line for the entire width of the chart, regardless if the control limit changes, either explicitly, or using the **AutoCalculateControlLimits** method. If the **ControlLineMode** property is SPCChartObjects.CONTROL_LINE_VARIABLE, the SPC limit value plots at the value it had when the sample subgroup values updated. If you change a control limit value, the control limit line will no longer be a straight horizontal line, instead it will be jagged, or stepped, depending on the changes made.



There are three ways to enter new SPC limit values. See the example program TimeAttributeControlCharts.VariableControlLimits for an example of all three methods. First, you can use the method **ChartData.SetControlLimitValues** method.

[C#]

```
double [] initialControlLimits = {0.13, 0.0, 0.27};
double [] changeControlLimits = {0.11, 0.0, 0.25};
.
.
this.PrimaryChart.ControlLineMode = SPCChartObjects.CONTROL_LINE_VARIABLE;
.
.
// Change limits at sample subgroup 10
if (i== 10)
{
    this.ChartData.SetControlLimitValues(changeControlLimits);
}
this.ChartData.AddNewSampleRecord(timestamp, samples);
```

[VB]

```
Dim initialControlLimits As Double() = {0.13, 0.0, 0.27}
Dim changeControlLimits As Double() = {0.11, 0.0, 0.25}
.
.
Me.PrimaryChart.ControlLineMode = SPCChartObjects.CONTROL_LINE_VARIABLE
.
.
' Change limits at sample subgroup 10
If i = 10 Then
    Me.ChartData.SetControlLimitValues(changeControlLimits)
End If
```

```
Me.ChartData.AddNewSampleRecord(timestamp, samples)
```

Second, you can use the **AutoCalculateControlLimits** method. You must have already added a collection of sampled data values to the charts **ChartData** SPC data object before you can call this method, since the method uses the internal **ChartData** object to provide the historical values needed in the calculation.

[C#]

```
.
.
this.PrimaryChart.ControlLineMode = SPCChartObjects.CONTROL_LINE_VARIABLE;
.
.

// Variable Control Limits re-calculated every update after 10 using
//  AutoCalculateControlLimits
if (i > 10)
    this.AutoCalculateControlLimits();
this.ChartData.AddNewSampleRecord(timestamp, samples);
```

[VB]

```
.
.
Me.PrimaryChart.ControlLineMode = SPCChartObjects.CONTROL_LINE_VARIABLE
.
.

'  Variable Control Limits re-calculated every update after 10 using
`  AutoCalculateControlLimits
If i > 10 Then
  Me.AutoCalculateControlLimits()
End If
Me.ChartData.AddNewSampleRecord(timestamp, samples)
```

Last, you can enter the SPC control limits with every new sample subgroup record, using one of the methods that include a control limits array parameter.

[C#]

```
double [] initialControlLimits = {0.13, 0.0, 0.27};
double [] changeControlLimits = {0.11, 0.0, 0.25};
DoubleArray variableControlLimits;.
.
.
this.PrimaryChart.ControlLineMode = SPCChartObjects.CONTROL_LINE_VARIABLE;
.
.
//     Variable Control Limits updated using AddNewSampleRecord
    if (i== 10) // need to convert changeControlLimits to a DoubleArray
        variableControlLimits = new DoubleArray(changeControlLimits);
    this.ChartData.AddNewSampleRecord(timestamp, samples,
        variableControlLimits, notesstring);
```

[VB]

```
Dim initialControlLimits As Double() = {0.13, 0.0, 0.27}
```

EWM<

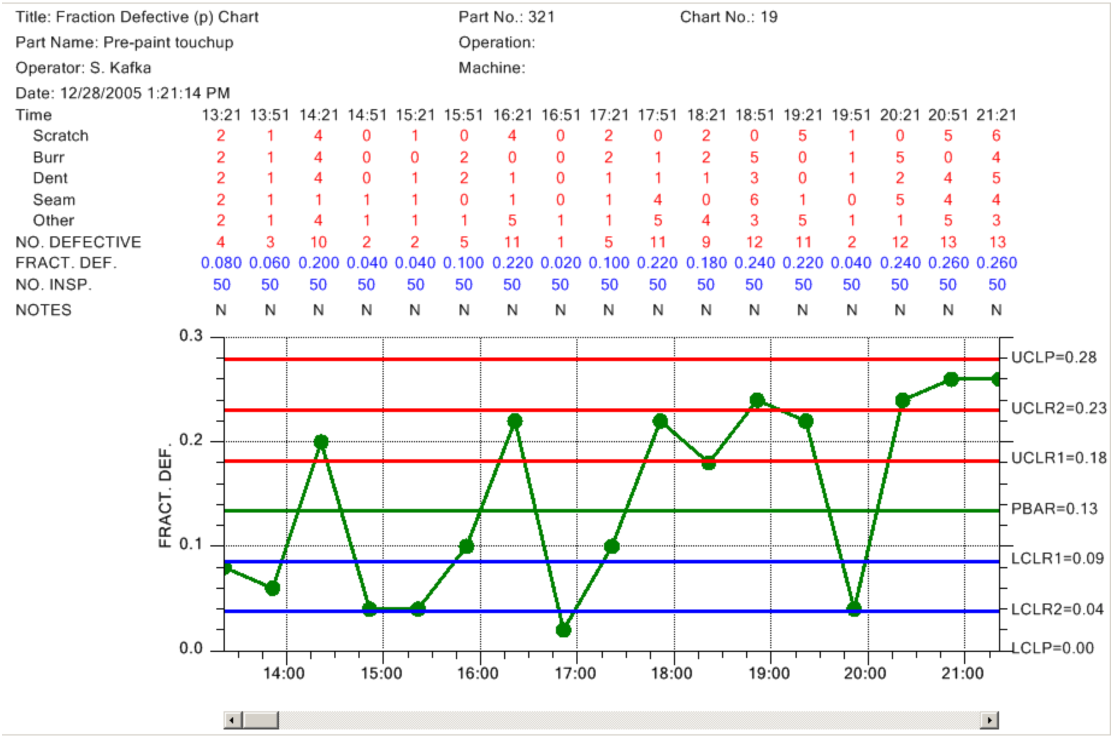
```
Dim changeControlLimits As Double() = {0.11, 0.0, 0.25}
Dim variableControlLimits As DoubleArray
.
.
this.PrimaryChart.ControlLineMode = SPCChartObjects.CONTROL_LINE_VARIABLE;
.
.
'   Variable Control Limits updated using AddNewSampleRecord
If i = 10 Then ' need to convert changeControlLimits to a DoubleArray
  variableControlLimits = New DoubleArray(changeControlLimits)
End If
Me.ChartData.AddNewSampleRecord(timestamp, samples, variableControlLimits)
```

Multiple SPC Control Limits

The normal SPC control limit displays at the 3-sigma level, both high and low.A common standard is that if the process variable under observation falls outside of the +-3-sigma limits the process is out of control. The default setup of our variable control charts have a high limit at the +3-sigma level, a low limit at the -3-sigma level, and a target value. There are situations where the quality engineer also wants to display control limits at the 1-sigma and 2-sigma level. The operator might receive some sort of preliminary warning if the process variable exceeds a 2-sigma limit.

.

You are able to add additional control limit lines to an attribute control chart, as in the example program TimeAttributeControlCharts.MultipleControlLimitsChart.



Title: Fraction Defective (p) Chart　　Part No.: 321　　Chart No.: 19

Part Name: Pre-paint touchup　　Operation:

Operator: S. Kafka　　Machine:

Date: 12/28/2005 1:21:14 PM

Time	13:21	13:51	14:21	14:51	15:21	15:51	16:21	16:51	17:21	17:51	18:21	18:51	19:21	19:51	20:21	20:51	21:21
Scratch	2	1	4	0	1	0	4	0	2	0	2	0	5	1	0	5	6
Burr	2	1	4	0	0	2	0	0	2	1	2	5	0	1	5	0	4
Dent	2	1	4	0	1	2	1	0	1	1	1	3	0	1	2	4	5
Seam	2	1	1	1	1	0	1	0	1	4	0	6	1	0	5	4	4
Other	2	1	4	1	1	1	5	1	1	5	4	3	5	1	1	5	3
NO. DEFECTIVE	4	3	10	2	2	5	11	1	5	11	9	12	11	2	12	13	13
FRACT. DEF.	0.080	0.060	0.200	0.040	0.040	0.100	0.220	0.020	0.100	0.220	0.180	0.240	0.220	0.040	0.240	0.260	0.260
NO. INSP.	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
NOTES	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N

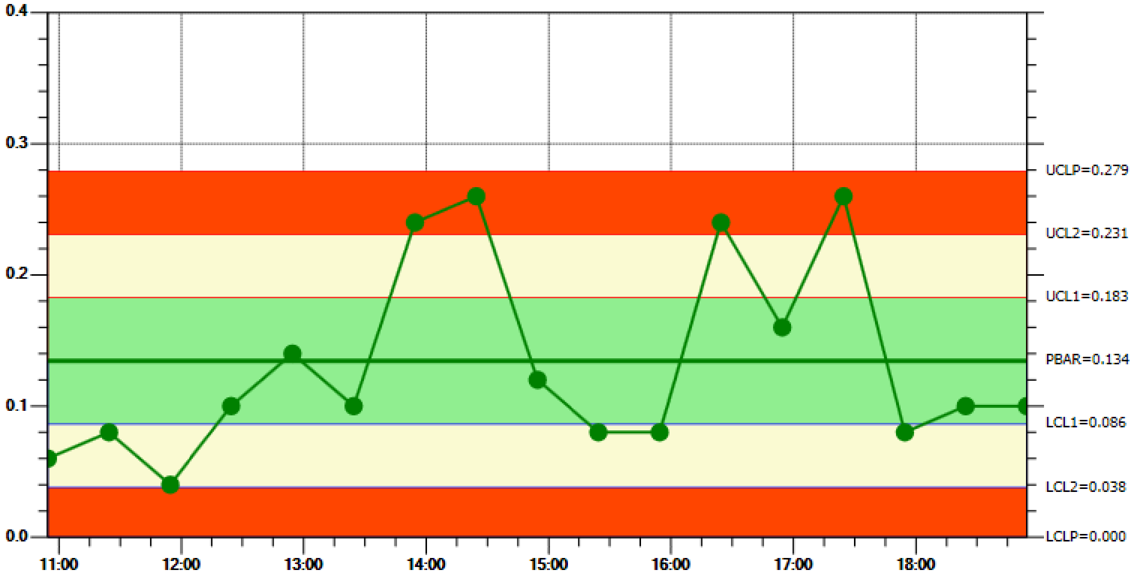
We added a method (**Add3SigmaControlLimits**) which will generate multiple control limits, for +-1, 2, and 3- sigma levels, based on an initial specification of the target value, and the +-3 sigma control limits. This is most useful if you want to generate +-1, 2 and 3-sigma control limits in order to fill in between them with a zone fill color. See the TimeAttributeControlCharts.MultiControlLimitsChart example. If you call the **AutoCalculateControlLimits** method, the initial +-1,2 and 3-sigma control limit values will be altered to the new, calculated values, but the control limit lines remain, with their new values. Since you do not normally want to be generating alarm messages for excursions into the +-1 and 2-sigma limit areas, the **Add3SigmaControl** limits has the option of disabling alarm notification in the case of +-1 and +-2 alarm conditions.

[C#]

```
double target = 0.14, lowlim = 0, highlim = 0.28;
bool limitcheck = false;
this.PrimaryChart.Add3SigmaControlLimits(target, lowlim, highlim, limitcheck);
this.PrimaryChart.ControlLimitLineFillMode = true;
```

[VB]

```
Dim target As Double = 0.14, lowlim As Double = 0, highlim As Double = 0.28
Dim limitcheck As Boolean = False
Me.PrimaryChart.Add3SigmaControlLimits(target, lowlim, highlim, limitcheck)
Me.PrimaryChart.ControlLimitLineFillMode = True
```



Control Limit Fill Option used with +-1, 2 and 3-sigma control limits

You can also add additional control limits one at a time. By default you get the +-3-sigma control limits. So additional control limits should be considered +-2-sigma and +-1-sigma control limits. Do not confuse control limits with specification limits, which must be added using the **AddSpecLimit** method. There are two steps to adding additional control limits: creating a **SPCControlLimitRecord** object for the new control limit, and adding the control limit to the chart using the charts **AddAdditionalControlLimit** method**.** It is critical that you add them in a specific order, that order being:

Primary Chart	SPC_LOWER_CONTROL_LIMIT_2	(2-sigma lower limit)
Primary Chart	SPC_UPPER_CONTROL_LIMIT_2	(2-sigma upper limit)
Primary Chart	SPC_LOWER_CONTROL_LIMIT_1	(1-sigma lower limit)
Primary Chart	SPC_UPPER_CONTROL_LIMIT_1	(1-sigma upper limit)

[C#]

```
double sigma2 = 2.0;
double sigma1 = 1.0;
// Create multiple limits
// For PrimaryChart
SPCControlLimitRecord  lcl2 = new SPCControlLimitRecord(this.ChartData,
SPCControlLimitRecord.SPC_LOWERTHAN_LIMIT, 0,"LCL2", "LCL2");
SPCControlLimitRecord  ucl2 = new SPCControlLimitRecord(this.ChartData,
SPCControlLimitRecord.SPC_GREATERTHAN_LIMIT, 0.28,"UCL2", "UCL2");

this.PrimaryChart.AddAdditionalControlLimit(lcl2,
SPCChartObjects.SPC_LOWER_CONTROL_LIMIT_2, sigma2);
this.PrimaryChart.AddAdditionalControlLimit(ucl2,
SPCChartObjects.SPC_UPPER_CONTROL_LIMIT_2, sigma2);

SPCControlLimitRecord  lcl3 = new SPCControlLimitRecord(this.ChartData,
SPCControlLimitRecord.SPC_LOWERTHAN_LIMIT, 0.07,"LCL1", "LCL1");
SPCControlLimitRecord  ucl3 = new SPCControlLimitRecord(this.ChartData,
SPCControlLimitRecord.SPC_GREATERTHAN_LIMIT, 0.21,"UCL1", "UCL1");

this.PrimaryChart.AddAdditionalControlLimit(lcl3,
SPCChartObjects.SPC_LOWER_CONTROL_LIMIT_1, sigma1);
this.PrimaryChart.AddAdditionalControlLimit(ucl3,
SPCChartObjects.SPC_UPPER_CONTROL_LIMIT_1, sigma1);
```

[VB]

```
Dim sigma2 As Double = 2.0
Dim sigma1 As Double = 1.0
' Create multiple limits
' For PrimaryChart
Dim lcl2 As SPCControlLimitRecord = New SPCControlLimitRecord(Me.ChartData,
SPCControlLimitRecord.SPC_LOWERTHAN_LIMIT, 0, "LCL2", "LCL2")
Dim ucl2 As SPCControlLimitRecord = New SPCControlLimitRecord(Me.ChartData,
SPCControlLimitRecord.SPC_GREATERTHAN_LIMIT, 0.28, "UCL2", "UCL2")

Me.PrimaryChart.AddAdditionalControlLimit(lcl2,
SPCChartObjects.SPC_LOWER_CONTROL_LIMIT_2, sigma2)
Me.PrimaryChart.AddAdditionalControlLimit(ucl2,
SPCChartObjects.SPC_UPPER_CONTROL_LIMIT_2, sigma2)

Dim lcl3 As SPCControlLimitRecord = New SPCControlLimitRecord(Me.ChartData,
SPCControlLimitRecord.SPC_LOWERTHAN_LIMIT, 0.07, "LCL1", "LCL1")
Dim ucl3 As SPCControlLimitRecord = New SPCControlLimitRecord(Me.ChartData,
SPCControlLimitRecord.SPC_GREATERTHAN_LIMIT, 0.21, "UCL1", "UCL1")

Me.PrimaryChart.AddAdditionalControlLimit(lcl3,
SPCChartObjects.SPC_LOWER_CONTROL_LIMIT_1, sigma1)
Me.PrimaryChart.AddAdditionalControlLimit(ucl3,
SPCChartObjects.SPC_UPPER_CONTROL_LIMIT_1, sigma1)
```

Special Note – When you create a **SPCControlLimitRecord** object, you can specify an actual limit level. If you do not call the charts **AutoCalculateControlLimits** method, the control limit will be displayed at that value. If you do call **AutoCalculateControlLimits** method, the auto-calculated value overrides the initial value (0.0 in the examples above). When you call the charts **AddAdditionalControlLimits** method, you specify the sigma level that is used by the **AutoCalculateControlLimits** to calculate the control limit level.

If you want the control limits displayed as filled areas, set the charts **ControlLimitLineFillMode** property True.

```
[C#]
spcChart.PrimaryChart.ControlLimitLineFillMode = true;
```

This will fill each control limit line from the limit line to the target value of the chart. In order for the fill to work properly, you must set this property after you define all additional control limits. Also, you must add the outer most control limits (SPC_UPPER_CONTROL_LIMIT_3 and SPC_LOWER_CONTROL_LIMIT_3) first, followed by the next outer most limits (SPC_UPPER_CONTROL_LIMIT_2 and SPC_LOWER_CONTROL_LIMIT_2), followed by the inner most control limits (SPC_UPPER_CONTROL_LIMIT_1 and SPC_LOWER_CONTROL_LIMIT_1). This way the fill of the inner limits will partially cover the fill of the outer limits, creating the familiar striped look you want to see.

Chart Y-Scale

You can set the minimum and maximum values of the two charts y-scales manually using the **PrimaryChart.MinY**, **PrimaryChart.MaxY**, **SecondaryChartMinY** and **SecondaryChartMaxY** properties.

[C#]

```
// Set initial scale of the y-axis of the mean chart
// If you are calling AutoScalePrimaryChartYRange this isn't really needed
this.PrimaryChart.MinY = 0;
this.PrimaryChart.MaxY = 40;
```

[VB]

```
' Set initial scale of the y-axis of the mean chart
' If you are calling AutoScalePrimaryChartYRange this isn't really needed
Me.PrimaryChart.MinY = 0
Me.PrimaryChart.MaxY = 40
```

It is easiest to just call the auto-scale routines after the chart has been initialized with data, and any control limits calculated.

[C#]

```
// Must have data loaded before any of the Auto.. methods are called
    SimulateData();

// Calculate the SPC control limits for both graphs of the current SPC chart
```

```
    this.AutoCalculateControlLimits();

// Scale the y-axis of the X-Bar chart to display all data and control limits
    this.AutoScalePrimaryChartYRange();
```

[VB]

```
' Must have data loaded before any of the Auto.. methods are called
SimulateData()

' Calculate the SPC control limits for both graphs of the current SPC chart
Me.AutoCalculateControlLimits()

' Scale the y-axis of the X-Bar chart to display all data and control limits
Me.AutoScalePrimaryChartYRange()
```

Once all of the graph parameters are set, call the method **RebuildChartUsingCurrentData**.

[C#]

```
// Rebuild the chart using the current data and settings
this.RebuildChartUsingCurrentData();
```

[VB]

```
' Rebuild the chart using the current data and settings
Me.RebuildChartUsingCurrentData()
```

If, at any future time you change any of the chart properties, you will need to call **RebuildChartUsingCurrentData** to force a rebuild of the chart, taking into account the current properties. **RebuildChartUsingCurrentData** invalidates the chart and forces a redraw**.** Our examples that update dynamically demonstrate this technique. The chart is setup with some initial settings and data values. As data is added in real-time to the graph, the chart SPC limits, and y-scales are constantly recalculated to take into account new data values. The code below is extracted from the **TimeAttributeControlCharts.DynamicAttributeControlChart** example program.

[C#]

```
private void timer1_Tick(object sender, System.EventArgs e)
{
    ChartCalendar timestamp = (ChartCalendar) startTime.Clone();
    // This simulates an assignable defect for each category, the last category
    //  is assigned the total number of defective parts, not defects.
    DoubleArray samples = this.ChartData.SimulateDefectRecord(50 * 0.134,
        SPCControlChartData.PERCENT_DEFECTIVE_PARTS_CHART);
    // Add new sample record
    this.ChartData.AddNewSampleRecord(timestamp, samples);
    // Simulate 30  minute passing
    startTime.Add(ChartObj.MINUTE, 30);

    // Calculate the SPC control limits
    this.AutoCalculatePrimaryControlLimits();
    // Scale the y-axis of the SPC chart to display all data and control limits
    this.AutoScalePrimaryChartYRange();
    // Rebuild the chart using the current data and settings
    this.RebuildChartUsingCurrentData();
}
```

[VB]

```
Private Sub Timer1_Tick(ByVal sender As System.Object,
    ByVal e As System.EventArgs) Handles Timer1.Tick
    Dim timestamp As ChartCalendar = CType(startTime.Clone(), ChartCalendar)
    ' This simulates an assignable defect for each category, the last category
    '  is assigned the total number of defective parts, not defects.
    Dim samples As DoubleArray = _
        Me.ChartData.SimulateDefectRecord(50 * 0.134, _
        SPCControlChartData.PERCENT_DEFECTIVE_PARTS_CHART)
    ' Add new sample record
     Me.ChartData.AddNewSampleRecord(timestamp, samples)
    ' Simulate 30  minute passing
    startTime.Add(ChartObj.MINUTE, 30)

    ' Calculate the SPC control limits
    Me.AutoCalculatePrimaryControlLimits()
    ' Scale the y-axis of the SPC chart to display all data and control limits
    Me.AutoScalePrimaryChartYRange()
    ' Rebuild the chart using the current data and settings
    Me.RebuildChartUsingCurrentData()
    Me.UpdateDraw()
End Sub 'timer1_Tick
```

Updating Chart Data

The real-time example above demonstrates how the SPC chart data is updated, using the **ChartData.AddNewSampleRecord** method. In this case, the chart data updates with each timer tick event, though it could just as easily be any other type of event. If you have already collected all of your data and just want to plot it all at once, use a simple loop like most of our examples do to update the data.

[C#]

```
private void SimulateData()
{
    if (this.IsDesignMode) return;

    for (int i=0; i < 200; i++)
    {
        ChartCalendar timestamp = (ChartCalendar) startTime.Clone();
    // This simulates an assignable defect for each category, the last category
    //  is assigned the total number of defective parts, not defects.
        DoubleArray samples = this.ChartData.SimulateDefectRecord(50 * 0.134,
            SPCControlChartData.PERCENT_DEFECTIVE_PARTS_CHART);
    // Add new sample record
        this.ChartData.AddNewSampleRecord(timestamp, samples);
        // Simulate 30  minute passing
        startTime.Add(ChartObj.MINUTE, 30);
    }
}
```

[VB]

```
Private Sub SimulateData()
   If Me.IsDesignMode Then
      Return
   End If
   Dim i As Integer
   For i = 0 To 199
      Dim timestamp As ChartCalendar = CType(startTime.Clone(), ChartCalendar)
      ' This simulates an assignable defect for each category, the last category
      '  is assigned the total number of defective parts, not defects.
      Dim samples As DoubleArray = _
           Me.ChartData.SimulateDefectRecord(50 * 0.134, _
            SPCControlChartData.PERCENT_DEFECTIVE_PARTS_CHART)
      ' Add new sample record
      Me.ChartData.AddNewSampleRecord(timestamp, samples)
      ' Simulate 30  minute passing
      startTime.Add(ChartObj.MINUTE, 30)
   Next i
End Sub 'SimulateData
```

In this example the sample data and the time stamp for each sample record is simulated. In your application, you will probably be reading the sample record values from some sort of database or file, along with the actual time stamp for that data.

If you want to append a text note to a sample record, use one of the **ChartData.AddNewSampleRecord** overrides that have a *notes* parameter. The code below is extracted from **the TimeAttributeControlCharts.SimpleAttributeControlChart** example.

[C#]

```
private void SimulateData()
{   String notesstring = "";
    if (this.IsDesignMode) return;
    for (int i=0; i < 200; i++)
    {
        ChartCalendar timestamp = (ChartCalendar) startTime.Clone();
    // This simulates an assignable defect for each category, the last category
    //  is assigned the total number of defective parts, not defects.
        DoubleArray samples = this.ChartData.SimulateDefectRecord(50 * 0.134,
            SPCControlChartData.PERCENT_DEFECTIVE_PARTS_CHART);
        double r = ChartSupport.GetRandomDouble();
        if (r < 0.1) // make a note on every tenth item, on average
            notesstring = "Note for sample subgroup #" + i.ToString() +
                ". Spray paint nozzel clogged. Replaced with new, Enois nozzle.";
        else
            notesstring = "";
        // Add new sample record
        this.ChartData.AddNewSampleRecord(timestamp, samples, notesstring);
        // Simulate 30  minute passing
        startTime.Add(ChartObj.MINUTE, 30);
    }
}
```

[VB]

```
Private Sub SimulateData() '
   Dim notesstring As [String] = ""
   If Me.IsDesignMode Then
     Return
   End If
   Dim i As Integer
   For i = 0 To 199
    Dim timestamp As ChartCalendar = CType(startTime.Clone(), ChartCalendar)
    ' This simulates an assignable defect for each category, the last category
    '  is assigned the total number of defective parts, not defects.
    Dim samples As DoubleArray = Me.ChartData.SimulateDefectRecord(50 * 0.134, _
        SPCControlChartData.PERCENT_DEFECTIVE_PARTS_CHART)

     Dim r As Double = ChartSupport.GetRandomDouble()
     If r < 0.1 Then ' make a note on every tenth item, on average
         notesstring = "Note for sample subgroup #" + i.ToString() + _
```

```
           ". Spray paint nozzel clogged. Replaced with new, Enois nozzle."
     Else
         notesstring = ""
     End If
     ' Add new sample record
      Me.ChartData.AddNewSampleRecord(timestamp, samples, notesstring)
     ' Simulate 30  minute passing
      startTime.Add(ChartObj.MINUTE, 30)
   Next i
End Sub 'SimulateData
```

Scatter Plots of the Actual Sampled Data

- This option is not applicable for attribute control charts.

Enable Chart ScrollBar

Set the **EnableScrollBar** property true to enable the chart scrollbar. You will then be able to window in on 8-20 sample subgroups at a time, from a much larger collection of measurement data representing hundreds or even thousands of subgroups, and use the scrollbar to move through the data, similar to paging through a spreadsheet.

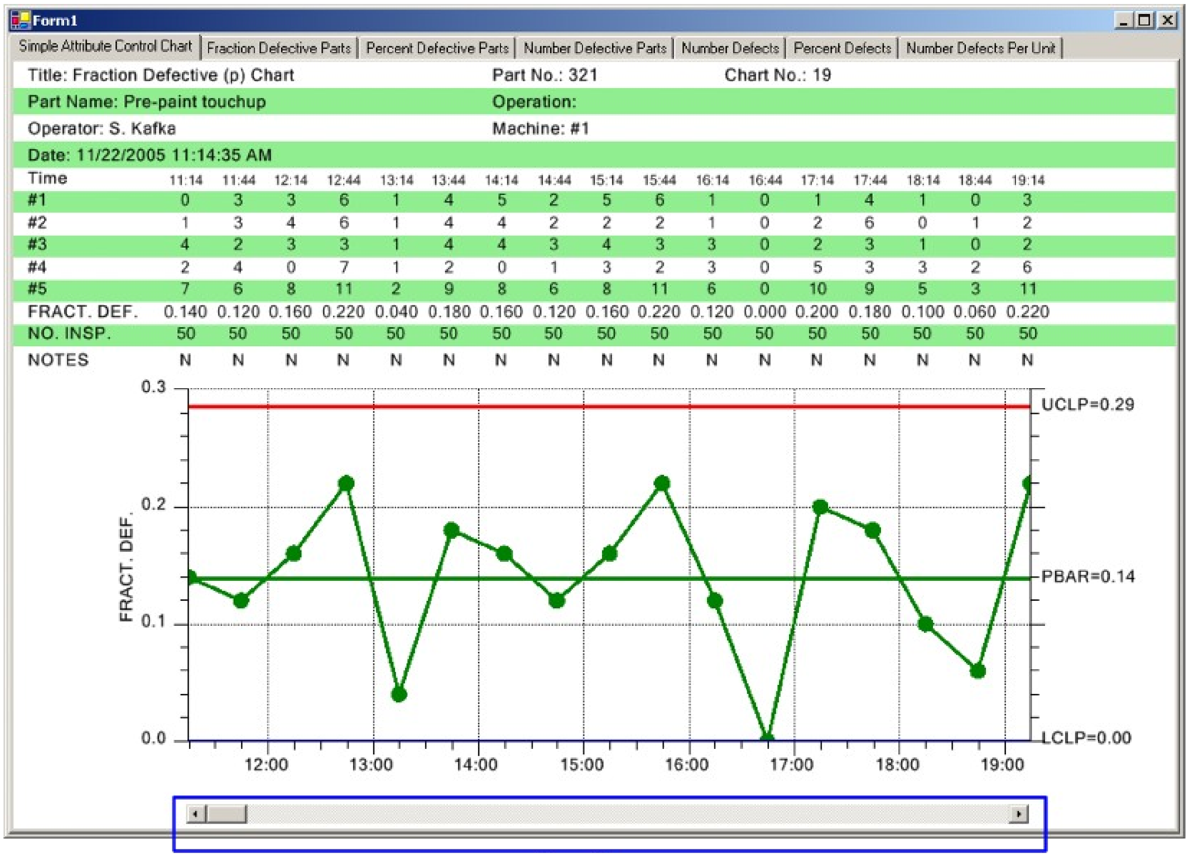
[C#]

```
// enable scroll bar
this.EnableScrollBar = true;
```

[VB]

```
' Enable scroll bar
Me.EnableScrollBar = True
```

EWM<A



Title: Fraction Defective (p) Chart Part No.: 321 Chart No.: 19

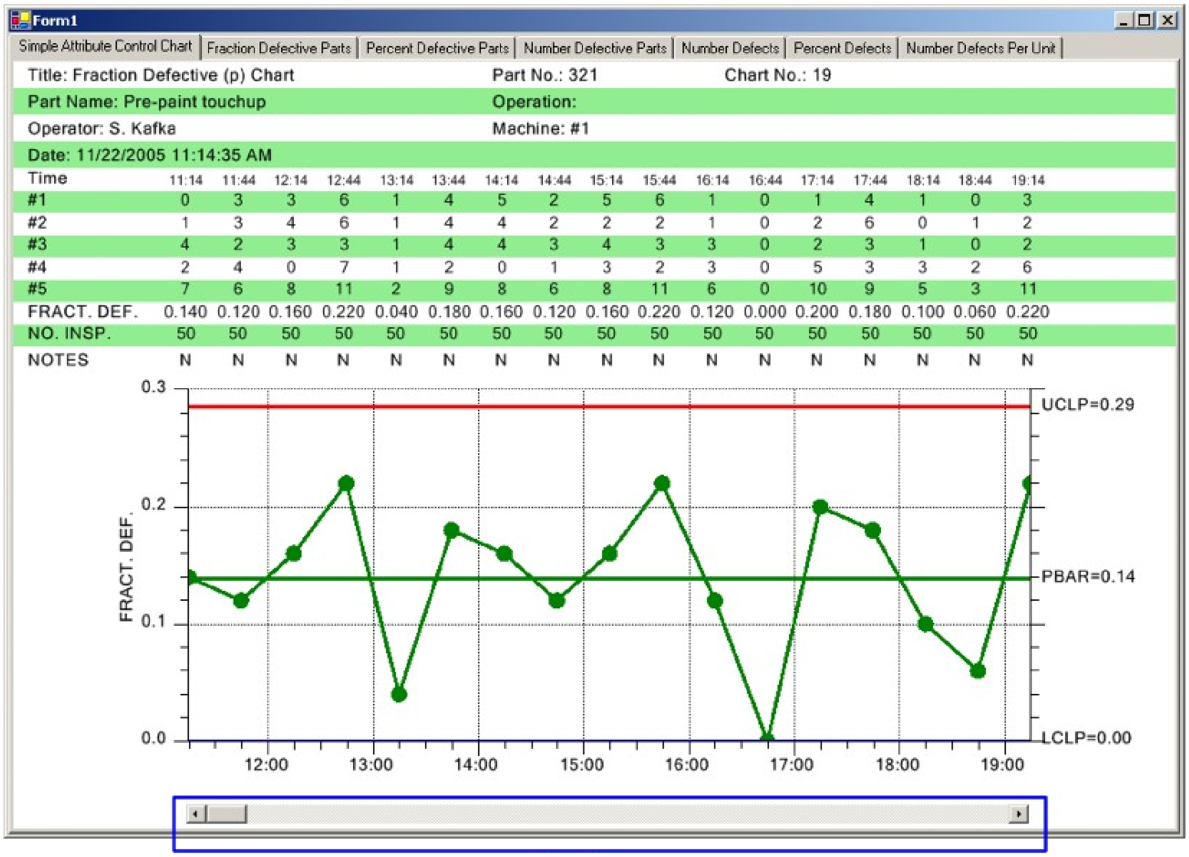
Part Name: Pre-paint touchup Operation:

Operator: S. Kafka Machine: #1

Date: 11/22/2005 11:14:35 AM

Time	11:14	11:44	12:14	12:44	13:14	13:44	14:14	14:44	15:14	15:44	16:14	16:44	17:14	17:44	18:14	18:44	19:14
#1	0	3	3	6	1	4	5	2	5	6	1	0	1	4	1	0	3
#2	1	3	4	6	1	4	4	2	2	2	1	0	2	6	0	1	2
#3	4	2	3	3	1	4	4	3	4	3	3	0	2	3	1	0	2
#4	2	4	0	7	1	2	0	1	3	2	3	0	5	3	3	2	6
#5	7	6	8	11	2	9	8	6	8	11	6	0	10	9	5	3	11
FRACT. DEF.	0.140	0.120	0.160	0.220	0.040	0.180	0.160	0.120	0.160	0.220	0.120	0.000	0.200	0.180	0.100	0.060	0.220
NO. INSP.	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
NOTES	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N

Scrollbar



Title: Fraction Defective (p) Chart Part No.: 321 Chart No.: 19
Part Name: Pre-paint touchup Operation:
Operator: S. Kafka Machine: #1
Date: 11/22/2005 11:14:35 AM

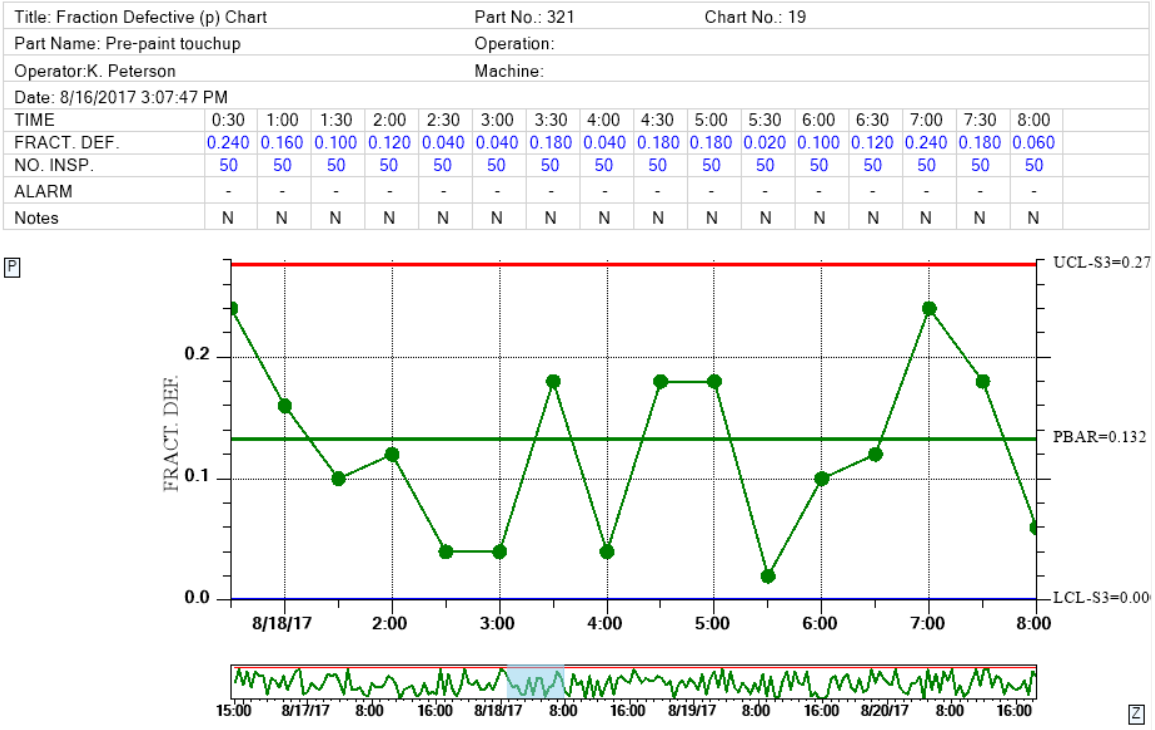
Time	11:14	11:44	12:14	12:44	13:14	13:44	14:14	14:44	15:14	15:44	16:14	16:44	17:14	17:44	18:14	18:44	19:14
#1	0	3	3	6	1	4	5	2	5	6	1	0	1	4	1	0	3
#2	1	3	4	6	1	4	4	2	2	2	1	0	2	6	0	1	2
#3	4	2	3	3	1	4	4	3	4	3	3	0	2	3	1	0	2
#4	2	4	0	7	1	2	0	1	3	2	3	0	5	3	3	2	6
#5	7	6	8	11	2	9	8	6	8	11	6	0	10	9	5	3	11
FRACT. DEF.	0.140	0.120	0.160	0.220	0.040	0.180	0.160	0.120	0.160	0.220	0.120	0.000	0.200	0.180	0.100	0.060	0.220
NO. INSP.	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
NOTES	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N

Scrollbar

Zooming as an option for the scrollbar

The horizontal scrollbar can be replaced (toggled on/off actually) using the UI, with a zoom window, by clicking on the z-button in the lower right corner. The user will be able to scroll the charts by dragging the transparent blue zoom window left or right, or expand the x-axis scale by dragging the left or right edge of the zoom window.

EWM<A



Title: Fraction Defective (p) Chart | Part No.: 321 | Chart No.: 19

Part Name: Pre-paint touchup | Operation:

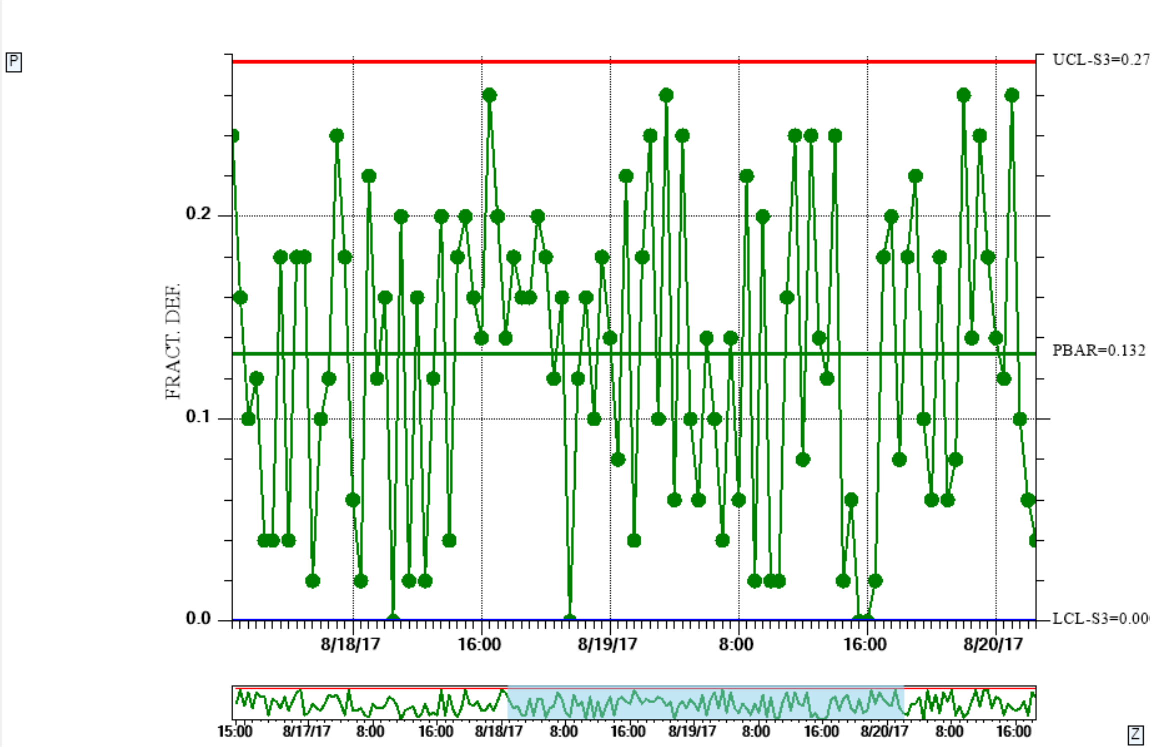
Operator:K. Peterson | Machine:

Date: 8/16/2017 3:07:47 PM

TIME	0:30	1:00	1:30	2:00	2:30	3:00	3:30	4:00	4:30	5:00	5:30	6:00	6:30	7:00	7:30	8:00
FRACT. DEF.	0.240	0.160	0.100	0.120	0.040	0.040	0.180	0.040	0.180	0.180	0.020	0.100	0.120	0.240	0.180	0.060
NO. INSP.	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
ALARM	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Notes	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N

Click on the Z button in the lower right corner and a zoom window will replace the scrollbar. Use the mouse to scroll the charts by dragging the transparent blue zoom window left or right, or expand the x-axis scale by dragging the left or right edge of the zoom window.

If the number of points in the display exceeds the initial setup value for the number of data points, the table is temporarily removed to prevent overlap of the table columns. If the number of data points is reduced to <= the initial number of points, the table will reappear.



If you use the zoom window to display a lot of points, the table at the top will disappear to prevent the table columns from overlapping.

The default display display for all chart types uses the scrollbar. The zoom option is seen as a small button with the character 'Z' in the lower right corner of the display. You enter/exit the zoom mode by clicking on that button. If you do not want the button to show at all, effectively making the zoom option inaccessible to the end user, set EnableZoomToggles property false.

.

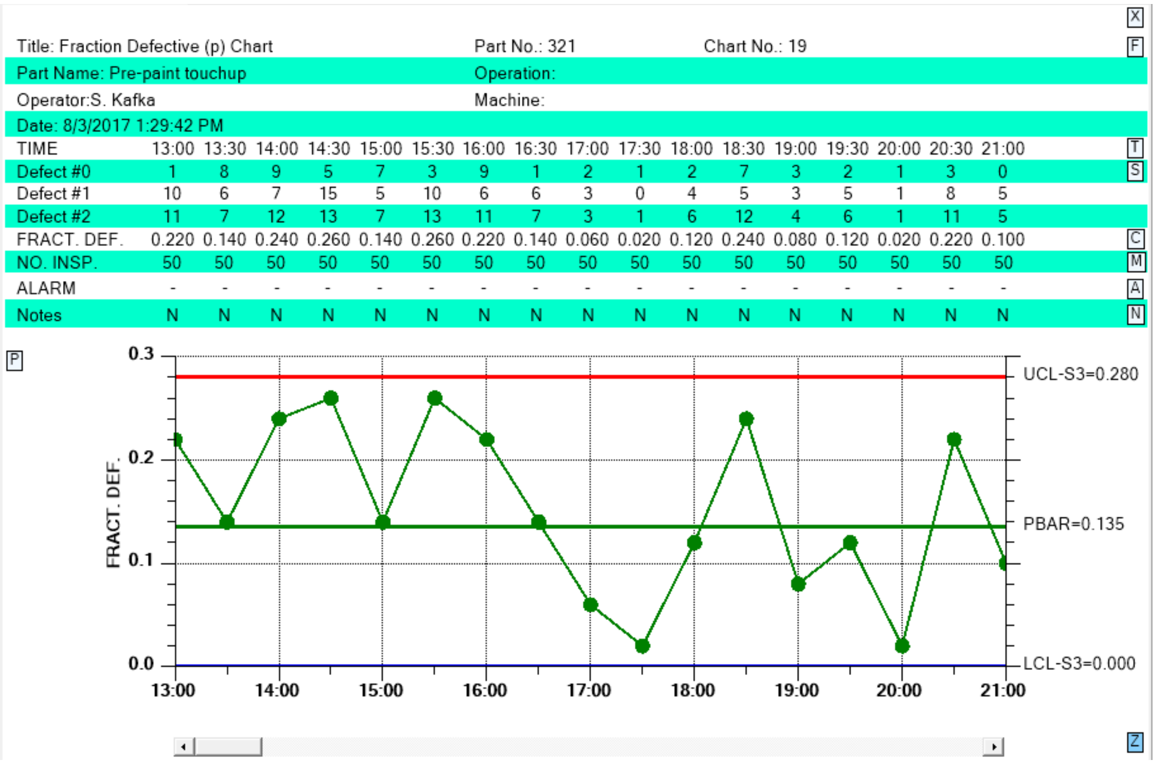
```
            this.EnableZoomToggles = false;
```

All of the examples in the NewRev3Features example show this feature.

Collapsible Items

Like the Zoom option described in the previous section, there are also UI buttons which can toggle on and off rows in the table, and the Primary and Secondary charts. Like the Zoom button, these UI buttons can be hidden from the end user if you don't want them to show. See the end of this section for examples. On the top and right of the table buttons will selectively collapse/restore rows of the table, freeing up more space for charts. Buttons to the left and bottom of the Primary and Secondary charts also permit the user to collapse/restore either of those charts, allowing the remaining chart to display in all of the remaining space.

EWM<A



Title: Fraction Defective (p) Chart Part No.: 321 Chart No.: 19

Part Name: Pre-paint touchup Operation:

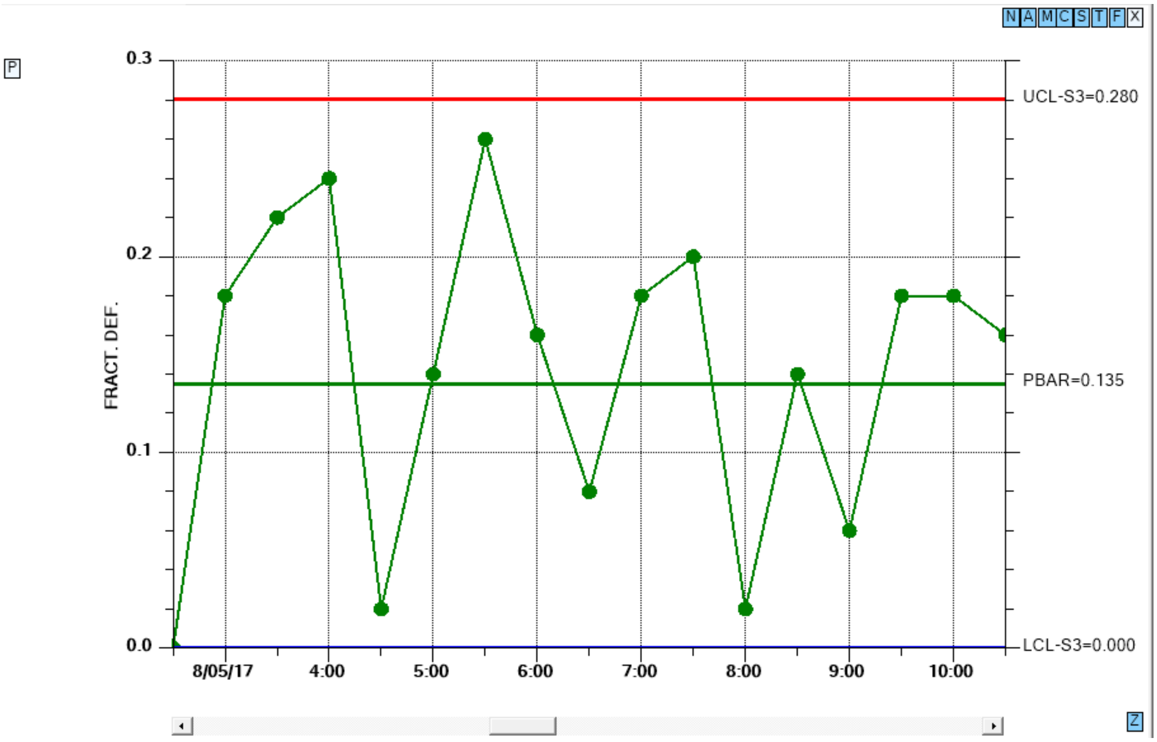
Operator:S. Kafka Machine:

Date: 8/3/2017 1:29:42 PM

TIME	13:00	13:30	14:00	14:30	15:00	15:30	16:00	16:30	17:00	17:30	18:00	18:30	19:00	19:30	20:00	20:30	21:00
Defect #0	1	8	9	5	7	3	9	1	2	1	2	7	3	2	1	3	0
Defect #1	10	6	7	15	5	10	6	6	3	0	4	5	3	5	1	8	5
Defect #2	11	7	12	13	7	13	11	7	3	1	6	12	4	6	1	11	5
FRACT. DEF.	0.220	0.140	0.240	0.260	0.140	0.260	0.220	0.140	0.060	0.020	0.120	0.240	0.080	0.120	0.020	0.220	0.100
NO. INSP.	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
ALARM	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Notes	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N

Buttons on the top, right and left of the display permit the user to selectively collapse portions of the chart.

Click on the N, A, M, P C and S buttons and shrink the table to following size.



In the example above, all of the chart options except for the Primary chart, have been toggled off using the buttons.

The buttons at the right of the table (and at the top right if toggled off) use the following ID's.

X	Turn on/off all table items at once
F	Turn on/off form data
T	Turn on/off sample interval time stamp data
S	Turn on/off sample value data
C	Turn on/off calculated value (mean, range, sum, etc.) data
P	Turn on/off process capability data
M	Turn on/off number of samples data
A	Turn on/off alarm data
N	Turn on/off notes data
Z	Bottom right - Turn on/off zoom control – the only button not affected by the X button above

The buttons at the left of the primary and secondary charts use the following ID's.

P	Turn on/off the Primary chart
S	Turn on/off the Secondary chart

If you do not want the buttons to show at all, effectively making these UI options inaccessible to the end user, set these properties false.

.

Hide the chart buttons on the left.

```
this.EnableChartToggles = false;
```

Hide the table buttons on the right.

```
this.EnableTableRowToggles = false;
```

You can hide ALL of the UI buttons (zoom, chart, and table) using the property EnableDisplayOptionToggles. This is what you use if you do not want the end user to have any control over the display of the table and chart.

```
this. EnableDisplayOptionToggles = false;
```

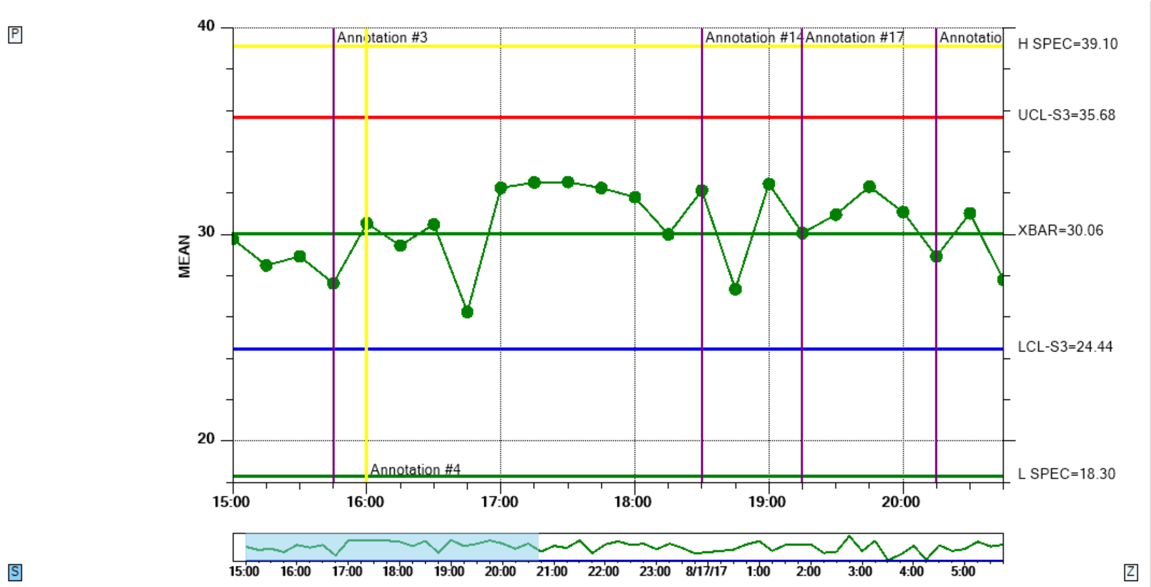
If you want to selectively enable options, first set EnableDisplayOptionToggles true. The other button display enables won't work unless this option is true. Then disable the options you do not want to show. In the example below, only the zoom option is left visible.

EWM<

```
            this.EnableDisplayOptionToggles = true;
            this.EnableTableRowToggles = false;
            this.EnableChartToggles = false;
            this.EnableZoomToggle = true;
```

Enhanced Annotations

The chart annotations have been enhanced with a vertical line with accompanying text using programmer specified justification.



The enhanced chart annotations include a vertical line to mark the data point, and many justification options (top, middle, bottom, left, right and center).

The new version of AddAnnotation is just an override of the original, with added parameters to specify the vertical line attributes and the text justification.

AddAnnotation

Add an annotation to a data point in the specified SPC chart.

```
public int AddAnnotation(int chart, int datapointindex, String text, int just,
ChartAttribute attrib)
```

where:

chart Specifies whether the annotation is added to the primary, or secondary chart. Use one of the SPChartObjects constants:

SPCChartObjects.PRIMARY_CHART or SPCChartObjects.SECONDARY_CHART.

datapointindex The index of the data point the annotation is for.

text A string string representing the annotation.

just The justification for the text to the x-position of the annotation. Use one of the annotation justification constants: ANNOTATION_UPPER_RIGHT, ANNOTATION_UPPER_LEFT, ANNOTATION_LOWER_RIGHT, ANNOTATION_LOWER_LEFT, ANNOTATION_UPPER_CENTER, ANNOTATION_LOWER_CENTER, ANNOTATION_DATAPOINT_RIGHT, ANNOTATION_DATAPOINT_LEFT

attrib A the attribute of the vertical line.

You call the AddAnnotation method immediately after the AddNewSampleRecord method call. For Example:

```
          // Add the new sample subgroup to the chart
this.ChartData.AddNewSampleRecord(timestamp, samples, notesstring);

int index = this.ChartData.CurrentNumberRecords - 1;
string annotstring = "Annotation #" + index.ToString();
ChartAttribute lineattrib = new ChartAttribute(Color.Purple, 2,
         DashStyle.Solid);
// Adjust justification to minimize overlap of adjacent annotations
int annotjust = SPCAnnotation.ANNOTATION_UPPER_LEFT;

this.AddAnnotation(SPCChartObjects.PRIMARY_CHART, index,
        annotstring, annotjust, lineattrib);
```

See the NewRev3Features.Annotations demo for an example.

SPC Chart Histograms

Viewing frequency histograms of the variation in the primary variable side-by-side with the SPC control chart makes it even easier to find out whether the variations are the result of natural variations or the result of some systemic change in the process. You can turn on integrated frequency histograms for either chart using the **PrimaryChart.DisplayFrequencyHistogram** property of the chart.

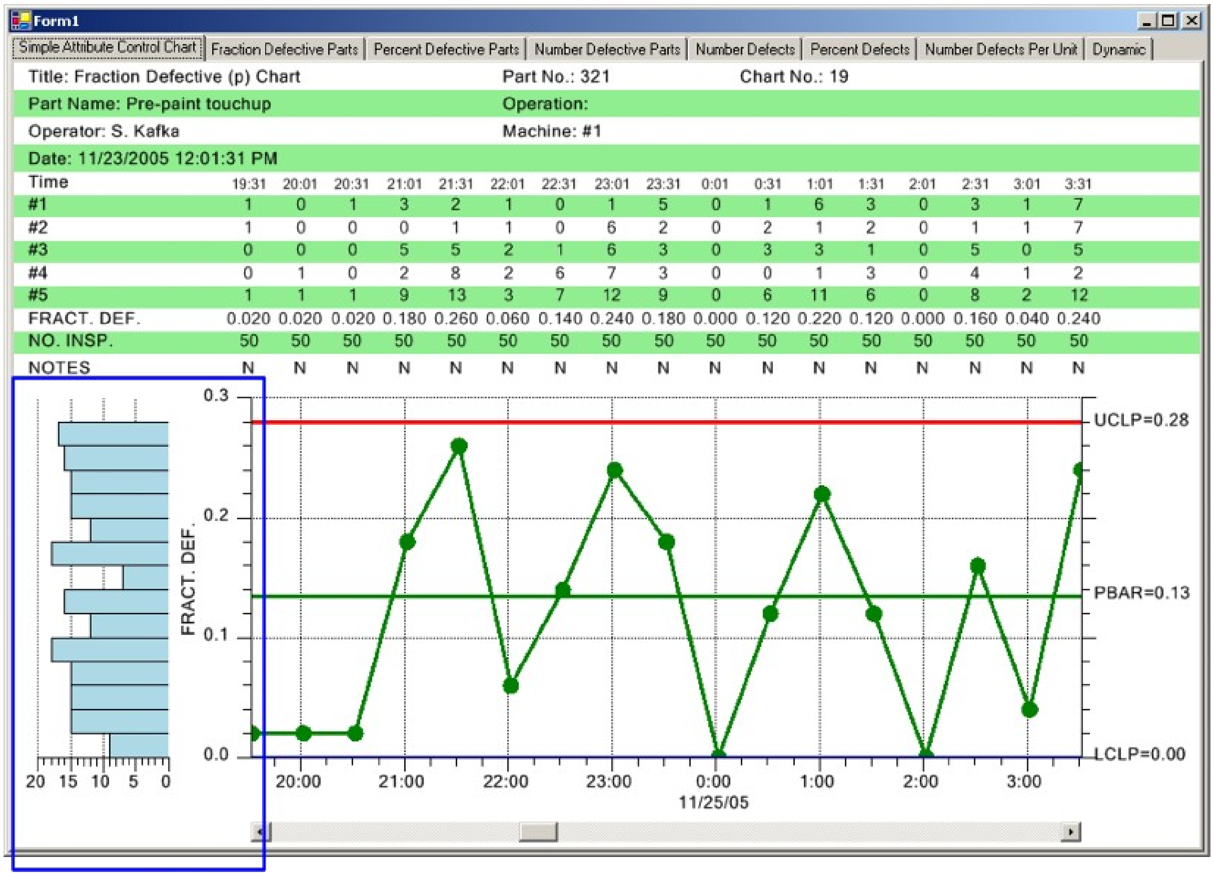
[C#]

```
//  frequency histogram for both charts
this.PrimaryChart.DisplayFrequencyHistogram = true;
```

EWM<

[VB]

```
'  frequency histogram for both charts
Me.PrimaryChart.DisplayFrequencyHistogram = True
```



Title: Fraction Defective (p) Chart	Part No.: 321	Chart No.: 19

Part Name: Pre-paint touchup	Operation:

Operator: S. Kafka	Machine: #1

Date: 11/23/2005 12:01:31 PM

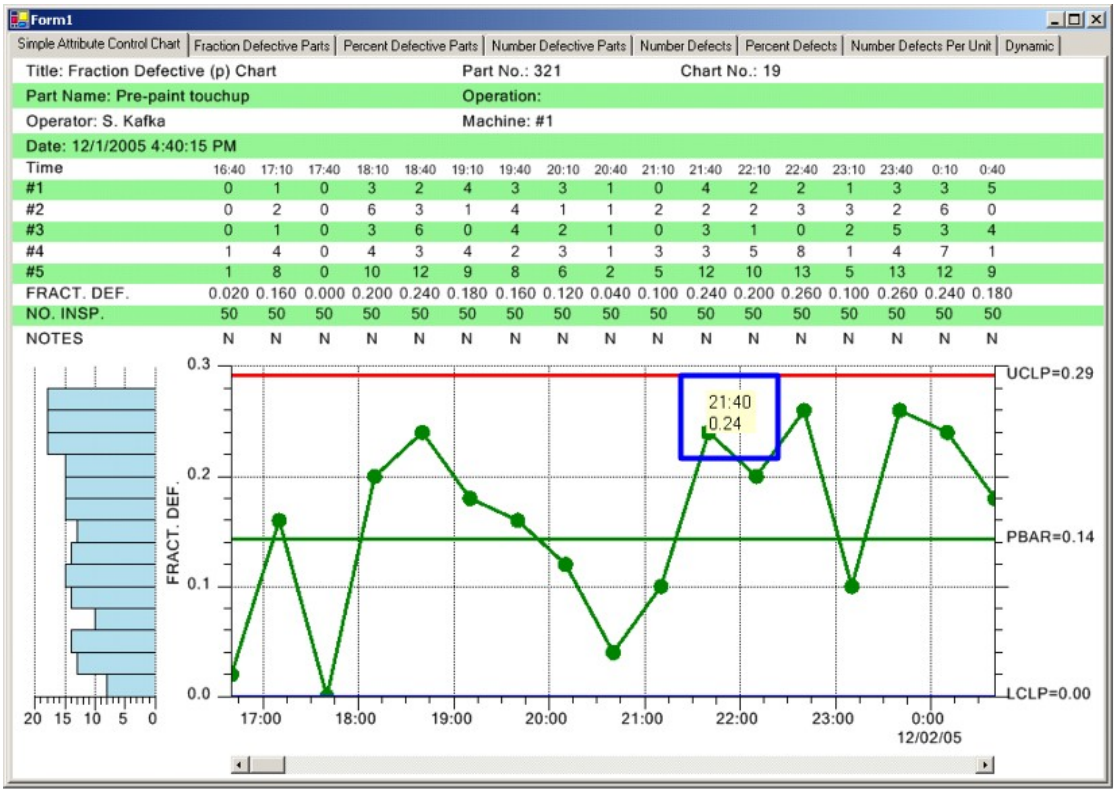
Time	19:31	20:01	20:31	21:01	21:31	22:01	22:31	23:01	23:31	0:01	0:31	1:01	1:31	2:01	2:31	3:01	3:31
#1	1	0	1	3	2	1	0	1	5	0	1	6	3	0	3	1	7
#2	1	0	0	0	1	1	0	6	2	0	2	1	2	0	1	1	7
#3	0	0	0	5	5	2	1	6	3	0	3	3	1	0	5	0	5
#4	0	1	0	2	8	2	6	7	3	0	0	1	3	0	4	1	2
#5	1	1	1	9	13	3	7	12	9	0	6	11	6	0	8	2	12
FRACT. DEF.	0.020	0.020	0.020	0.180	0.260	0.060	0.140	0.240	0.180	0.000	0.120	0.220	0.120	0.000	0.160	0.040	0.240
NO. INSP.	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
NOTES	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N

Frequency Histogram

SPC Chart Data and Notes Tooltips

You can invoke two types of tooltips using the mouse. The first is a data tooltip. When you hold the mouse button down over one of the data points in the primary chart, the x and y values for that data point display in a popup tooltip..

Data Tooltip



Title: Fraction Defective (p) Chart Part No.: 321 Chart No.: 19

Part Name: Pre-paint touchup Operation:

Operator: S. Kafka Machine: #1

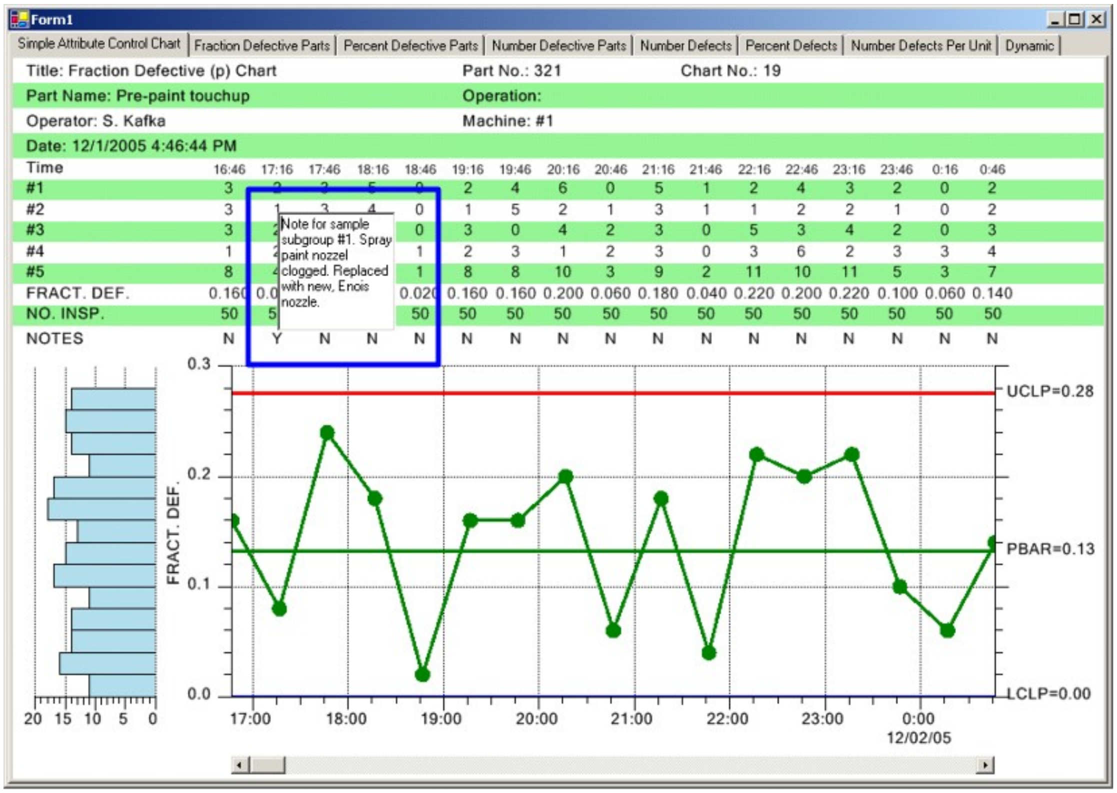
Date: 12/1/2005 4:40:15 PM

Time	16:40	17:10	17:40	18:10	18:40	19:10	19:40	20:10	20:40	21:10	21:40	22:10	22:40	23:10	23:40	0:10	0:40
#1	0	1	0	3	2	4	3	3	1	0	4	2	2	1	3	3	5
#2	0	2	0	6	3	1	4	1	1	2	2	2	3	3	2	6	0
#3	0	1	0	3	6	0	4	2	1	0	3	1	0	2	5	3	4
#4	1	4	0	4	3	4	2	3	1	3	3	5	8	1	4	7	1
#5	1	8	0	10	12	9	8	6	2	5	12	10	13	5	13	12	9
FRACT. DEF.	0.020	0.160	0.000	0.200	0.240	0.180	0.160	0.120	0.040	0.100	0.240	0.200	0.260	0.100	0.260	0.240	0.180
NO. INSP.	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
NOTES	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N

If you are displaying the Notes line in the table portion of the chart, the Notes entry for a sample subgroup displays “Y” if a note was recorded for that sample subgroup, or “N” if no note was recorded. Notes are recorded using one of the **ChartData.AddNewSampleRecord** overrides that include a notes parameter. See the section Updating Chart Data. If you click on a “Y” in the Notes row for a sample subgroup, the complete text of the note for that sample subgroup will display in a RichTextBox, immediately above the “Y”. You can actually edit the notes in the RichTextBox.

Notes Tooltip

EWM<A



Title: Fraction Defective (p) Chart　Part No.: 321　Chart No.: 19

Part Name: Pre-paint touchup　Operation:

Operator: S. Kafka　Machine: #1

Date: 12/1/2005 4:46:44 PM

Time	16:46	17:16	17:46	18:16	18:46	19:16	19:46	20:16	20:46	21:16	21:46	22:16	22:46	23:16	23:46	0:16	0:46
#1	3	2	3	5	0	2	4	6	0	5	1	2	4	3	2	0	2
#2	3	1	3	4	0	1	5	2	1	3	1	1	2	2	1	0	2
#3	3	2			0	3	0	4	2	3	0	5	3	4	2	0	3
#4	1	2			1	2	3	1	2	3	0	3	6	2	3	3	4
#5	8	4			1	8	8	10	3	9	2	11	10	11	5	3	7
FRACT. DEF.	0.160	0.0			0.020	0.160	0.160	0.200	0.060	0.180	0.040	0.220	0.200	0.220	0.100	0.060	0.140
NO. INSP.	50	5			50	50	50	50	50	50	50	50	50	50	50	50	50
NOTES	N	Y	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N

Note for sample subgroup #1. Spray paint nozzel clogged. Replaced with new, Enois nozzle.

[C#]

```
private void SimulateData()
{   String notesstring = "";
    if (this.IsDesignMode) return;
    for (int i=0; i < 200; i++)
    {
        .
        .
        .
        // Add new sample record
        this.ChartData.AddNewSampleRecord(timestamp, samples, notesstring);
        // Simulate 30  minute passing
        startTime.Add(ChartObj.MINUTE, 30);
    }
}
```

[VB]

```
Private Sub SimulateData()
   Dim notesstring As [String] = ""
   If Me.IsDesignMode Then
      Return
   End If
   Dim i As Integer
   For i = 0 To 199
        .
        .
        .

      ' Add new sample record
      Me.ChartData.AddNewSampleRecord(timestamp, samples, notesstring)
      ' Simulate 30  minute passing
      startTime.Add(ChartObj.MINUTE, 30)
   Next i
End Sub 'SimulateData
```

Both kinds of tooltips are on by default. Turn the tooltips on or off in the program using the **EnableDataToolTip** and **EnableNotesToolTip** flags.

[C#]

```
// Enable data and notes tooltips
this.EnableDataToolTip = true;
this.EnableNotesToolTip = true;
```

[VB]

```
' Enable data and notes tooltips
Me.EnableDataToolTip = True
Me.EnableNotesToolTip = True
```

The notes tooltip has an additional option. In order to make the notes tooltip “editable”, the tooltip, which is .Net RichEditBox, displays on the first click, and goes away on the second click. You can click inside the RichTextBox and not worry the tooltip suddenly disappearing. The notes tooltip works this way by default. If you wish to explicitly set it, or change it so that the tooltip only displays while the mouse button is held down, as the data tooltips do, set the **ChartData.NotesToolTips.ToolTipMode** property to **NotesToolTip.MOUSEDOWN_TOOLTIP**, as in the example below.

[C#]

```
// Enable data and notes tooltips
this.EnableDataToolTip = true;
this.EnableNotesToolTip = true;

this.ChartData.NotesToolTips.ButtonMask = MouseButtons.Right;
// default is MOUSETOGGLE_TOOLTIP
this.ChartData.NotesToolTips.ToolTipMode= NotesToolTip.MOUSEDOWN_TOOLTIP;
```

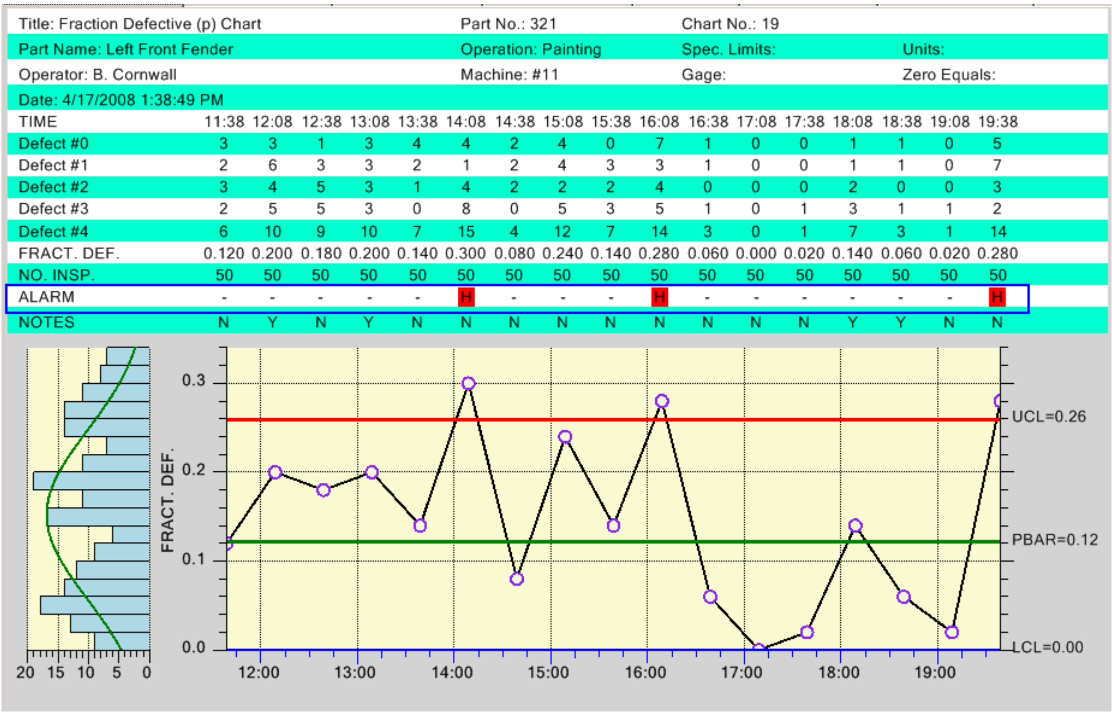
[VB]

```
' Enable data and notes tooltips
Me.EnableDataToolTip = True
Me.EnableNotesToolTip = True

Me.ChartData.NotesToolTips.ButtonMask = MouseButtons.Right
' default is MOUSETOGGLE_TOOLTIP
Me.ChartData.NotesToolTips.ToolTipMode = NotesToolTip.MOUSEDOWN_TOOLTIP
```

Enable Alarm Highlighting

EnableAlarmStatusValues

EWM<



Title: Fraction Defective (p) Chart	Part No.: 321	Chart No.: 19	
Part Name: Left Front Fender	Operation: Painting	Spec. Limits:	Units:
Operator: B. Cornwall	Machine: #11	Gage:	Zero Equals:
Date: 4/17/2008 1:38:49 PM			

TIME	11:38	12:08	12:38	13:08	13:38	14:08	14:38	15:08	15:38	16:08	16:38	17:08	17:38	18:08	18:38	19:08	19:38
Defect #0	3	3	1	3	4	4	2	4	0	7	1	0	0	1	1	0	5
Defect #1	2	6	3	3	2	1	2	4	3	3	1	0	0	1	1	0	7
Defect #2	3	4	5	3	1	4	2	2	2	4	0	0	0	2	0	0	3
Defect #3	2	5	5	3	0	8	0	5	3	5	1	0	1	3	1	1	2
Defect #4	6	10	9	10	7	15	4	12	7	14	3	0	1	7	3	1	14
FRACT. DEF.	0.120	0.200	0.180	0.200	0.140	0.300	0.080	0.240	0.140	0.280	0.060	0.000	0.020	0.140	0.060	0.020	0.280
NO. INSP.	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
ALARM	-	-	-	-	-	H	-	-	-	H	-	-	-	-	-	-	H
NOTES	N	Y	N	Y	N	N	N	N	N	N	N	N	N	Y	Y	N	N

There are several alarm highlighting options you can turn on and off. The alarm status line above is turned on/off using the EnableAlarmStatusValues property. We have set it on by default, so you will have to turn it off if you don't want it. Each sample interval has two small boxes that are labeled using one of several different characters, listed below. The most common are an "H" signifying a high alarm, a "L" signifying a low alarm, and a "-" signifying that there is no alarm. When specialized control rules are implemented, either using the named rules discussed in Chapter 8, or custom rules involving trending, oscillation, or stratification, a "T", "O" or "S" may also appear.

"-"	No alarm condition
"H"	High - Measured value is above a high limit
"L"	Low - Measured value falls below a low limit
"T"	Trending - Measured value is trending up (or down).
"O"	Oscillation - Measured value is oscillating (alternating) up and down.
"S"	Stratification - Measured value is stuck in a narrow band.

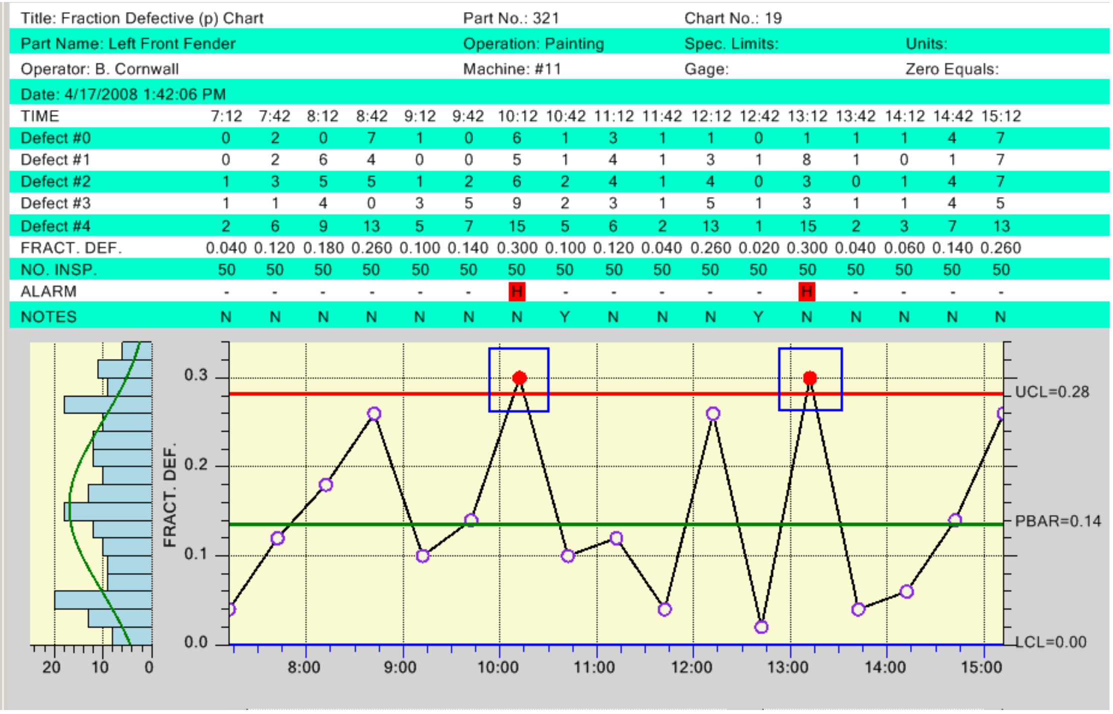
[C#]

```
// Alarm status line
this.EnableAlarmStatusValues = false;
```

[VB]

```
'Alarm status line
Me.EnableAlarmStatusValues = False
```

ChartAlarmEmphasisMode



Title: Fraction Defective (p) Chart | Part No.: 321 | Chart No.: 19

Part Name: Left Front Fender | Operation: Painting | Spec. Limits: | Units:

Operator: B. Cornwall | Machine: #11 | Gage: | Zero Equals:

Date: 4/17/2008 1:42:06 PM

TIME	7:12	7:42	8:12	8:42	9:12	9:42	10:12	10:42	11:12	11:42	12:12	12:42	13:12	13:42	14:12	14:42	15:12
Defect #0	0	2	0	7	1	0	6	1	3	1	1	0	1	1	1	4	7
Defect #1	0	2	6	4	0	0	5	1	4	1	3	1	8	1	0	1	7
Defect #2	1	3	5	5	1	2	6	2	4	1	4	0	3	0	1	4	7
Defect #3	1	1	4	0	3	5	9	2	3	1	5	1	3	1	1	4	5
Defect #4	2	6	9	13	5	7	15	5	6	2	13	1	15	2	3	7	13
FRACT. DEF.	0.040	0.120	0.180	0.260	0.100	0.140	0.300	0.100	0.120	0.040	0.260	0.020	0.300	0.040	0.060	0.140	0.260
NO. INSP.	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
ALARM	-	-	-	-	-	-	H	-	-	-	-	-	H	-	-	-	-
NOTES	N	N	N	N	N	N	N	Y	N	N	N	Y	N	N	N	N	N

[C#]

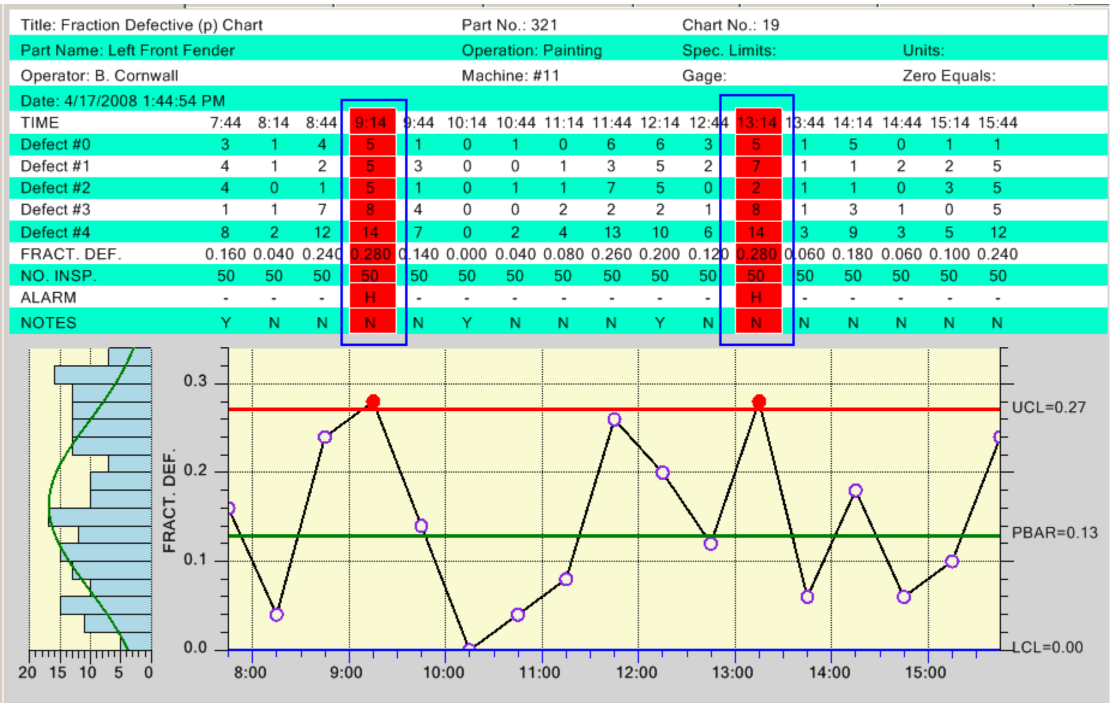
```
// Chart alarm emphasis mode
this.ChartAlarmEmphasisMode = SPCChartBase.ALARM_HIGHLIGHT_SYMBOL;
```

[VB]

```
' Chart alarm emphasis mode
Me.ChartAlarmEmphasisMode = SPCChartBase.ALARM_HIGHLIGHT_SYMBOL
```

The scatter plot symbol used to plot a data point in the chart is normally a fixed color circle. If you turn on the alarm highlighting for chart symbols the symbol color for a sample interval that is in an alarm condition will change to reflect the color of the associated alarm line. In the example above, a low alarm (blue circle) occurs at the beginning of the chart and a high alarm (red circle) occurs at the end of the chart. Alarm symbol highlighting is turned on by default. To turn it off use the SPCChartBase.ALARM_NO_HIGHLIGHT_SYMBOL constants.

TableAlarmEmphasisMode -



Title: Fraction Defective (p) Chart Part No.: 321 Chart No.: 19
Part Name: Left Front Fender Operation: Painting Spec. Limits: Units:
Operator: B. Cornwall Machine: #11 Gage: Zero Equals:
Date: 4/17/2008 1:44:54 PM

TIME	7:44	8:14	8:44	9:14	9:44	10:14	10:44	11:14	11:44	12:14	12:44	13:14	13:44	14:14	14:44	15:14	15:44
Defect #0	3	1	4	5	1	0	1	0	6	6	3	5	1	5	0	1	1
Defect #1	4	1	2	5	3	0	0	1	3	5	2	7	1	1	2	2	5
Defect #2	4	0	1	5	1	0	1	1	7	5	0	2	1	1	0	3	5
Defect #3	1	1	7	8	4	0	0	2	2	2	1	8	1	3	1	0	5
Defect #4	8	2	12	14	7	0	2	4	13	10	6	14	3	9	3	5	12
FRACT. DEF.	0.160	0.040	0.240	0.280	0.140	0.000	0.040	0.080	0.260	0.200	0.120	0.280	0.060	0.180	0.060	0.100	0.240
NO. INSP.	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
ALARM	-	-	-	H	-	-	-	-	-	-	-	H	-	-	-	-	-
NOTES	Y	N	N	N	N	Y	N	N	N	Y	N	N	N	N	N	N	N

C#]

```
// Table alarm emphasis mode
this.TableAlarmEmphasisMode = SPCChartBase.ALARM_HIGHLIGHT_BAR;
```

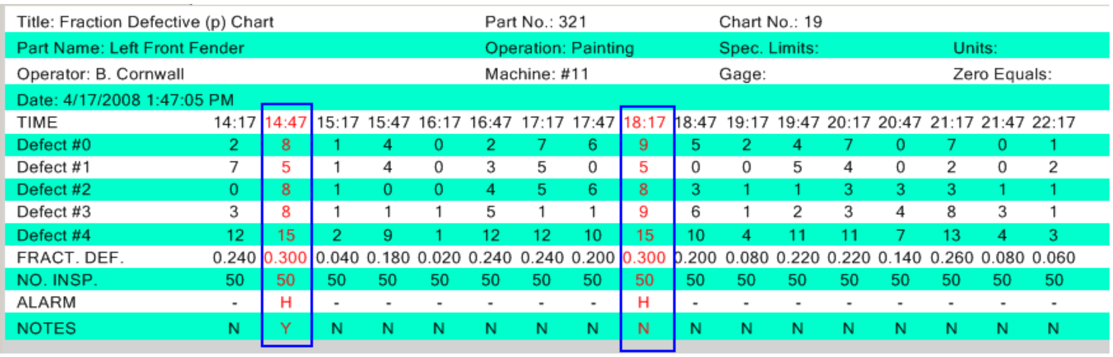
[VB]

```
' Table alarm emphasis mode
Me.TableAlarmEmphasisMode = SPCChartBase.ALARM_HIGHLIGHT_BAR
```

The entire column of the data table can be highlighted when an alarm occurs. There are four modes associated with this property:

ALARM_HIGHLIGHT_NONE	No alarm highlight
ALARM_HIGHLIGHT_TEXT	Text alarm highlight
ALARM_HIGHLIGHT_OUTLINE	Outline alarm highlight
ALARM_HIGHLIGHT_BAR	Bar alarm highlight

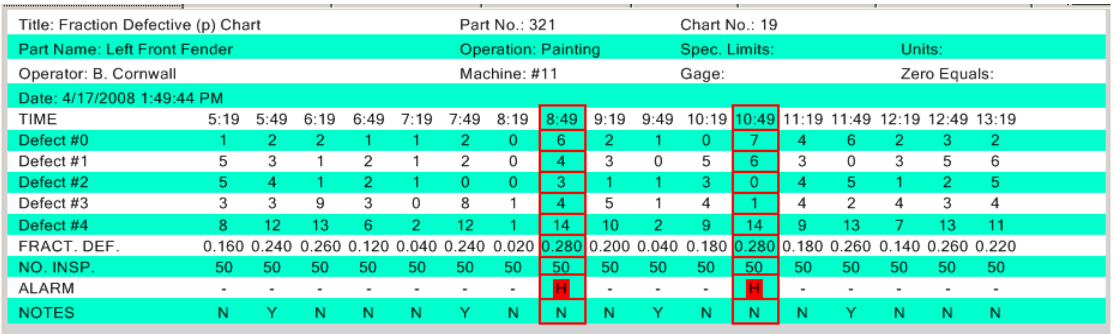
The example above uses the ALARM_HIGHLIGHT_BAR mode.



Title: Fraction Defective (p) Chart	Part No.: 321	Chart No.: 19	
Part Name: Left Front Fender	Operation: Painting	Spec. Limits:	Units:
Operator: B. Cornwall	Machine: #11	Gage:	Zero Equals:
Date: 4/17/2008 1:47:05 PM			

TIME	14:17	14:47	15:17	15:47	16:17	16:47	17:17	17:47	18:17	18:47	19:17	19:47	20:17	20:47	21:17	21:47	22:17
Defect #0	2	8	1	4	0	2	7	6	9	5	2	4	7	0	7	0	1
Defect #1	7	5	1	4	0	3	5	0	5	0	0	5	4	0	2	0	2
Defect #2	0	8	1	0	0	4	5	6	8	3	1	1	3	3	3	1	1
Defect #3	3	8	1	1	1	5	1	1	9	6	1	2	3	4	8	3	1
Defect #4	12	15	2	9	1	12	12	10	15	10	4	11	11	7	13	4	3
FRACT. DEF.	0.240	0.300	0.040	0.180	0.020	0.240	0.240	0.200	0.300	0.200	0.080	0.220	0.220	0.140	0.260	0.080	0.060
NO. INSP.	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
ALARM	-	H	-	-	-	-	-	-	H	-	-	-	-	-	-	-	-
NOTES	N	Y	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N

The example above uses the ALARM_HIGHLIGHT_TEXT mode



Title: Fraction Defective (p) Chart	Part No.: 321	Chart No.: 19	
Part Name: Left Front Fender	Operation: Painting	Spec. Limits:	Units:
Operator: B. Cornwall	Machine: #11	Gage:	Zero Equals:
Date: 4/17/2008 1:49:44 PM			

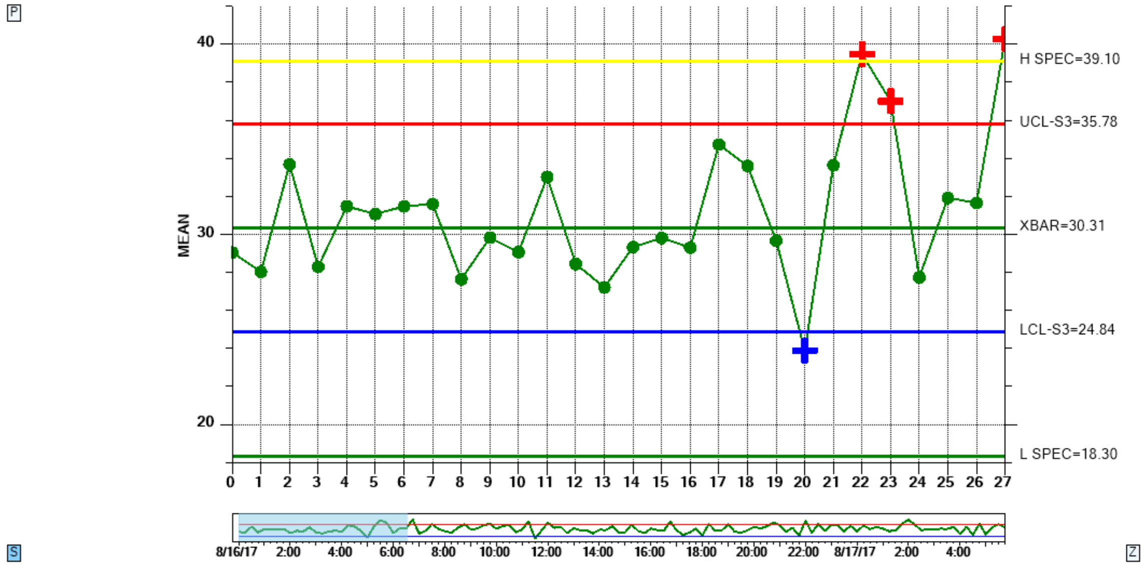
TIME	5:19	5:49	6:19	6:49	7:19	7:49	8:19	8:49	9:19	9:49	10:19	10:49	11:19	11:49	12:19	12:49	13:19
Defect #0	1	2	2	1	1	2	0	6	2	1	0	7	4	6	2	3	2
Defect #1	5	3	1	2	1	2	0	4	3	0	5	6	3	0	3	5	6
Defect #2	5	4	1	2	1	0	0	3	1	1	3	0	4	5	1	2	5
Defect #3	3	3	9	3	0	8	1	4	5	1	4	1	4	2	4	3	4
Defect #4	8	12	13	6	2	12	1	14	10	2	9	14	9	13	7	13	11
FRACT. DEF.	0.160	0.240	0.260	0.120	0.040	0.240	0.020	0.280	0.200	0.040	0.180	0.280	0.180	0.260	0.140	0.260	0.220
NO. INSP.	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
ALARM	-	-	-	-	-	-	-	H	-	-	-	H	-	-	-	-	-
NOTES	N	Y	N	N	N	Y	N	N	N	Y	N	N	N	Y	N	N	N

The example above uses the ALARM_HIGHLIGHT_OUTLINE mode. In the table above, the column outlines in blue and red reflect what is actually displayed in the chart, whereas in the other TableAlarmEmphasisMode examples the outline just shows where the alarm highlighting occurs.

The default mode is ALARM_HIGHLIGHT_NONE mode.

Enhanced Out of Limit Symbol

One of the SPC chart options is to mark a data point which is outside of control limits by changing the color of the symbol. It is now possible to also also change the size and symbol type for out of limit symbols.



A data point which is outside of control limits can be automatically marked using color, symbol type, and color.

In the example above, the out of control symbol for the Primary chart is an extra large plus sign, while for the the Secondary chart it is an extra large square, both contrasting with the default circle symbol.

The default symbol for the out of control indication is the same as the in control indication, a filled circle. Only a color change signifies out of control. To change the notification symbol, and size, use the OutOfLimitSymbolNumber and OutOfLimitSymbolSize properties, which apply separately to the Primary and Secondary charts.

OutOfLimitSymbolNumber

Set the symbol type for data points found to be in alarm. Use one of the symbol type constants found in the ChartObj class: SQUARE, TRIANGLE, DIAMOND, CROSS, PLUS, STAR, LINE, HBAR, VBAR, CIRCLE. :

OutOfLimitSymbolSize

Set the size in pixels of the out of limit symbol:

For example:

```
// Need to have this on for symbol emphasis
            this.ChartAlarmEmphasisMode = SPCChartBase.ALARM_HIGHLIGHT_SYMBOL;


// Set symbol emphasis type, and size, for primary chart
```

```
            this.PrimaryChart.OutOfLimitSymbolNumber = ChartObj.PLUS;
            this.PrimaryChart.OutOfLimitSymbolSize = 22;

// Set symbol emphasis type, and size, for secondary chart
            this.SecondaryChart.OutOfLimitSymbolNumber = ChartObj.SQUARE;
            this.SecondaryChart.OutOfLimitSymbolSize = 18;
```

See the NewRev3Features.EnhanceLimitSymbols demo for an example.

AutoLogAlarmsAsNotes

When an alarm occurs, details of the alarm can be automatically logged as a Notes record. Just set the AutoLogAlarmsAsNotes property to true.

[C#]

```
this.AutoLogAlarmsAsNotes = true;
```

[VB]

```
Me.AutoLogAlarmsAsNotes = True
```

Creating a Batch-Based Attribute Control Chart

Both the **SPCEventAttributeContolChart, SPCTimeAttributeContolChart** and **SPCBatchAttributeControlChart** derive from the **SPCChartBase** and as a result the two classes are very similar and share 95% of all properties in common. Creating and initializing a batch-based SPC chart is much the same as that of a time-based SPC chart. Derive your base class from **SPCBatchAttributeControlChart** class as seen in the **BatchAttributeControlChart.SimpleAttributeControlChart** example program.

[C#]

```
public class SimpleAttributeControlChart:
    com.quinn-curtis.spcchartnet6.SPCBatchAttributeControlChart
    {
        private System.ComponentModel.IContainer components;
        ChartCalendar startTime = new ChartCalendar();
        int batchCounter = 0;
        //  SPC attribute control chart type
        int charttype = SPCControlChartData.FRACTION_DEFECTIVE_PARTS_CHART;
        // Number of samples per sub group
        int numsamplespersubgroup = 50;
        // Number of defect categories
        int numcategories = 5;
        // Number of data points in the view
        int numdatapointsinview = 17;
        // The time increment between adjacent subgroups
        int timeincrementminutes = 30;

        public SimpleAttributeControlChart()
        {
            // This call is required by the Windows.Forms Form Designer.
            InitializeComponent();
            // Have the chart fill parent client area
            this.Dock = DockStyle.Fill;
            // Define and draw chart
            InitializeChart();
        }
```

EWM<

```
        void InitializeChart()
        {
            // Initialize the SPCBatchAttributeControlChart
            this.InitSPCBatchAttributeControlChart(charttype, numcategories,
                numsamplespersubgroup, numdatapointsinview);
            .
            .
            .
        }
}
```

[VB]

```
Public Class SimpleAttributeControlChart
    Inherits com.quinn-curtis.spcchartnet6.SPCBatchAttributeControlChart

#Region " Windows Form Designer generated code "
    Public Sub New()
        MyBase.New()

        'This call is required by the Windows Form Designer.
        InitializeComponent()

        'Add any initialization after the InitializeComponent() call
        ' Have the chart fill parent client area
        Me.Dock = DockStyle.Fill
        ' Define and draw chart
        InitializeChart ()
    End Sub

    .
    .
    .

#End Region


    Dim startTime As New ChartCalendar()
    Dim batchCounter As Integer = 0
    '  SPC attribute control chart type
    Dim charttype As Integer = SPCControlChartData.FRACTION_DEFECTIVE_PARTS_CHART
    ' Number of samples per sub group
    Dim numsamplespersubgroup As Integer = 50
    ' Number of defect categories
    Dim numcategories As Integer = 5
    ' Number of data points in the view
    Dim numdatapointsinview As Integer = 17
    ' The time increment between adjacent subgroups
    Dim timeincrementminutes As Integer = 30


    Sub InitializeChart ()
        ' Initialize the SPCBatchAttributeControlChart
        Me.InitSPCBatchAttributeControlChart(charttype, numcategories, _
           numsamplespersubgroup, numdatapointsinview)
        .
        .
        .
    End Sub 'DrawChart
```

Establish the control chart type (p-, np-, c- or u-chart) using the attribute control charts **InitSPCBatchAttributeControlChart** initialization routine.

SPCBatchAttributeControlChart.InitSPCBatchAttributeControlChart Method

This initialization method initializes the most important values in the creation of a SPC chart.

[VB]

```
Overloads Public Sub InitSPCBatchAttributeControlChart( _
   ByVal charttype As Integer, _
   ByVal numcategories As Integer, _
   ByVal numsamplespersubgroup As Integer, _
   ByVal numdatapointsinview As Integer, _
)
```

[C#]

```
public void InitSPCBatchAttributeControlChart(
   int charttype,
   int numcategories,
   int numsamplespersubgroup,
   int numdatapointsinview,
);
```

Parameters

charttype

Specifies the chart type. Use one of the SPC Attribute Control chart types: PERCENT_DEFECTIVE_PARTS_CHART, FRACTION_DEFECTIVE_PARTS_CHART, NUMBER_DEFECTIVE_PARTS_CHART, NUMBER_DEFECTS_PERUNIT_CHART, NUMBER_DEFECTS_CHART, NUMBER_DEFECTS_PER_MILLION_CHART.

numcategories

In Attribute Control Charts this value represents the number of defect categories used to determine defect counts.

numsamplespersubgroup

In an Attribute Control chart it represents the total sample size per sample subgroup from which the defect data is counted.

numdatapointsinview

Specifies the number of sample subgroups displayed in the graph at one time.

Update the chart data using the **ChartData.AddNewSampleRecord** method, using an override that has the batch number (*batchCounter* below) as the first parameter. Even though a time stamp value is used in the **AddNewSampleRecord** method, it is not used in the actual graph. Instead, it is used as the time stamp for the batch in the table portion of the chart.

[C#]

```
private void SimulateData()
{
    if (this.IsDesignMode) return;
    for (int i=0; i < 200; i++)
    {
        double batchnumber = batchCounter;
        ChartCalendar timestamp = (ChartCalendar) startTime.Clone();
        // Simulate a new sample record
        DoubleArray samples = this.ChartData.SimulateDefectRecord(50 * 0.134,
            SPCControlChartData.PERCENT_DEFECTIVE_PARTS_CHART);
        //  add a new sample record
        this.ChartData.AddNewSampleRecord(batchnumber, timestamp, samples);
        // Simulate timeincrementminutes  minute passing
        startTime.Add(ChartObj.MINUTE, timeincrementminutes);
        batchCounter++;
    }
}
```

[VB]

```
      Private Sub SimulateData() '
        If Me.IsDesignMode Then
            Return
        End If
        Dim i As Integer
        For i = 0 To 199
            Dim group As Double = batchCounter
            Dim timestamp As ChartCalendar = _
                CType(startTime.Clone(), ChartCalendar)
            ' Simulate a new sample record
            Dim samples As DoubleArray = _
                Me.ChartData.SimulateDefectRecord(50 * 0.134, _
                SPCControlChartData.PERCENT_DEFECTIVE_PARTS_CHART)
            '  add a new sample record
            Me.ChartData.AddNewSampleRecord(group, timestamp, samples)
            ' Simulate timeincrementminutes  minute passing
            startTime.Add(ChartObj.MINUTE, timeincrementminutes)
            batchCounter += 1
        Next i
    End Sub 'SimulateData
```

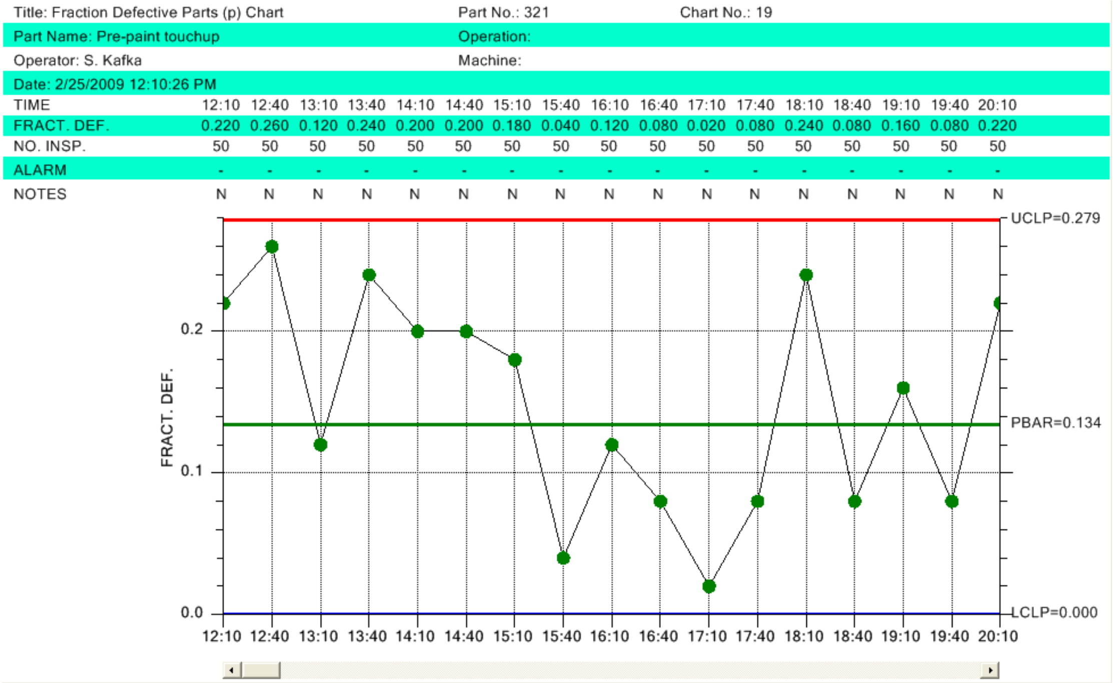
Changing the Batch and Event Control Chart X-Axis Labeling Mode

In revisions prior to 2.0, the x-axis tick marks of a batch control chart could only be labeled with the numeric batch number of the sample subgroup. While batch number labeling is still the default mode, it is now possible to label the sample subgroup tick marks using the time stamp of the sample subgroup, or a user-defined string unique to each sample subgroup.

You may find that labeling every subgroup tick mark with a time stamp, or a user-defined string, causes the axis labels to stagger because there is not enough room to display the tick mark label without overlapping its neighbor. In these cases you may wish to reduce the number of sample subgroups you show on the page using the *numdatapointsinview* variable found in all of the example programs.

```
// Number of datapoints in the view
int numdatapointsinview = 13;
```

Batch Control Chart X-Axis Time Stamp Labeling



Title: Fraction Defective Parts (p) Chart Part No.: 321 Chart No.: 19
Part Name: Pre-paint touchup Operation:
Operator: S. Kafka Machine:
Date: 2/25/2009 12:10:26 PM

TIME	12:10	12:40	13:10	13:40	14:10	14:40	15:10	15:40	16:10	16:40	17:10	17:40	18:10	18:40	19:10	19:40	20:10
FRACT. DEF.	0.220	0.260	0.120	0.240	0.200	0.200	0.180	0.040	0.120	0.080	0.020	0.080	0.240	0.080	0.160	0.080	0.220
NO. INSP.	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
ALARM	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
NOTES	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N

Fraction Defective Parts Chart using time stamp labeling of the x-axis

Set the x-axis labeling mode using the overall charts XAxisStringLabelMode property, setting it `SPCChartObjects.AXIS_LABEL_MODE_TIME.`

[C#]

```
// Label the tick mark with time stamp of sample group
this.XAxisStringLabelMode = SPCChartObjects.AXIS_LABEL_MODE_TIME;
```

[VB]

```
' Label the tick mark with time stamp of sample group
Me.XAxisStringLabelMode = SPCChartObjects.AXIS_LABEL_MODE_TIME
```

When updating the chart with sample data, use AddNewSampleRecord overload that has batch number and a time stamp parameters.

[C#]

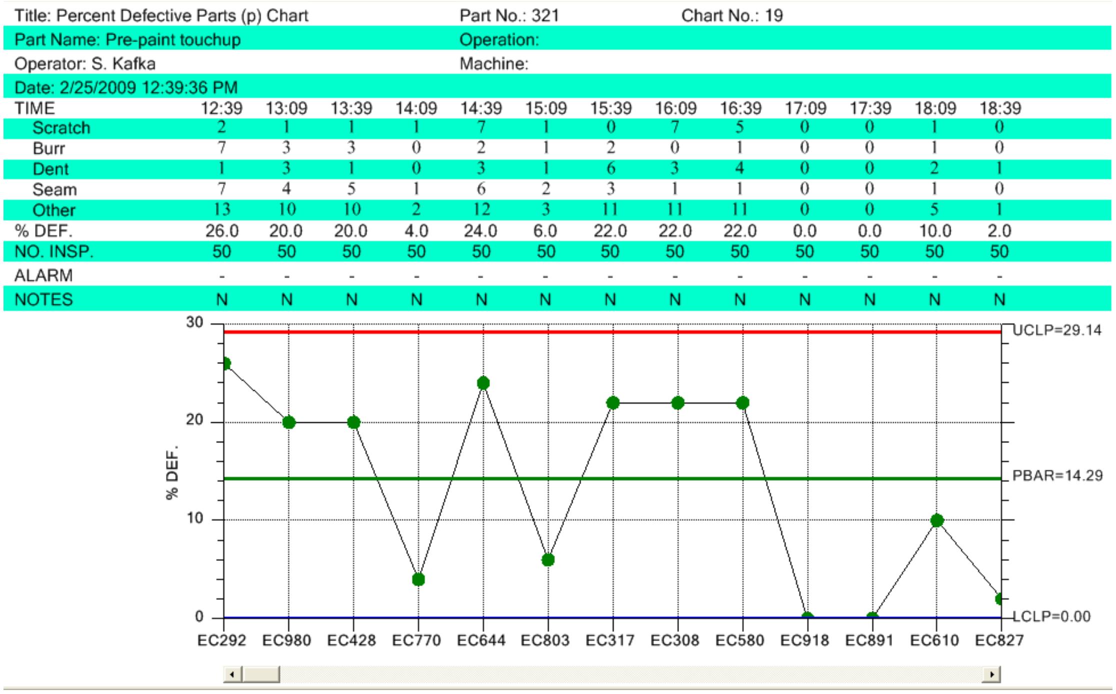
```
this.ChartData.AddNewSampleRecord(batchCounter, timestamp, samples);
```

[VB]

```
Me.ChartData.AddNewSampleRecord(batchCounter, timestamp, samples)
```

See the example program BatchAttributeControlCharts.FractionDefectivePartsControlChart for a complete example. Reset the axis labeling mode back to batch number labeling by assigning the XAxisStringLabelMode property to SPCChartObjects.AXIS_LABEL_MODE_DEFAULT.

Batch Control Chart X-Axis User-Defined String Labeling



Title: Percent Defective Parts (p) Chart   Part No.: 321   Chart No.: 19

Part Name: Pre-paint touchup   Operation:

Operator: S. Kafka   Machine:

Date: 2/25/2009 12:39:36 PM

TIME	12:39	13:09	13:39	14:09	14:39	15:09	15:39	16:09	16:39	17:09	17:39	18:09	18:39
Scratch	2	1	1	1	7	1	0	7	5	0	0	1	0
Burr	7	3	3	0	2	1	2	0	1	0	0	1	0
Dent	1	3	1	0	3	1	6	3	4	0	0	2	1
Seam	7	4	5	1	6	2	3	1	1	0	0	1	0
Other	13	10	10	2	12	3	11	11	11	0	0	5	1
% DEF.	26.0	20.0	20.0	4.0	24.0	6.0	22.0	22.0	22.0	0.0	0.0	10.0	2.0
NO. INSP.	50	50	50	50	50	50	50	50	50	50	50	50	50
ALARM	-	-	-	-	-	-	-	-	-	-	-	-	-
NOTES	N	N	N	N	N	N	N	N	N	N	N	N	N

Percent Defective Parts Chart using user-defined string labeling of the x-axis

Set the x-axis labeling mode using the overall charts XAxisStringLabelMode property, setting it `SPCChartObjects.AXIS_LABEL_MODE_STRING`.

[C#]

```
// enable scroll bar
this.EnableScrollBar = true;
this.EnableCategoryValues = false;

// Label the tick mark with user-defined strings
this.XAxisStringLabelMode = SPCChartObjects.AXIS_LABEL_MODE_STRING;
```

[VB]

```
' enable scroll bar
Me.EnableScrollBar = True
Me.EnableCategoryValues = False

' Label the tick mark with user-defined strings
Me.XAxisStringLabelMode = SPCChartObjects. AXIS_LABEL_MODE_STRING
```

Use the AddAxisUserDefinedString method to supply a new string for every new sample subgroup. It must be called every time the AddNewSampleRecord method is called, or the user-defined strings will get out of sync with their respective sample subgroup. Reset the axis labeling mode back to batch number labeling by assigning the XAxisStringLabelMode property to SPCChartObjects.AXIS_LABEL_MODE_DEFAULT.

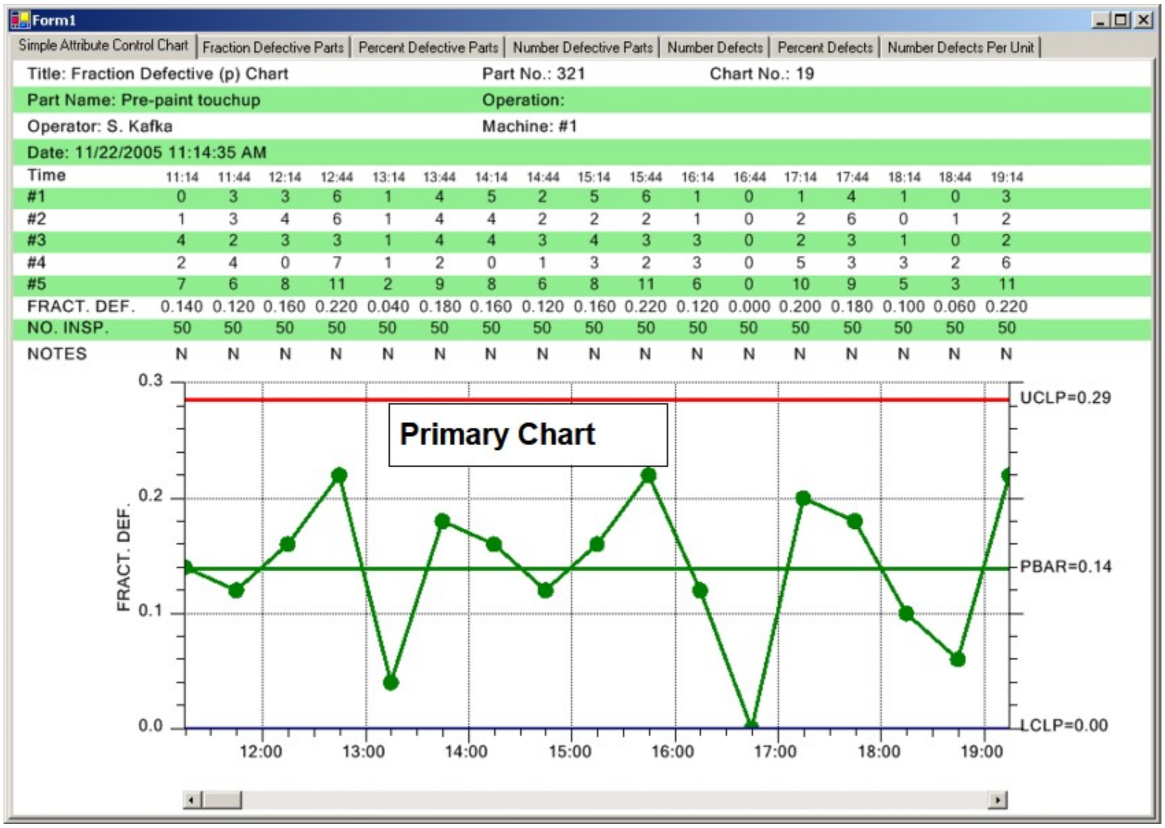
[C#]
```
this.ChartData.AddNewSampleRecord(batchCounter, timestamp, samples,
variableControlLimits);

// Make a random string to simulate some sort of batch sample group ID
int randomnum= (int) (1000 * ChartSupport.GetRandomDouble());
String batchidstring = "EC" + randomnum.ToString();
this.ChartData.AddAxisUserDefinedString(batchidstring);
```

[VB]
```
' Add a new sample record
Me.ChartData.AddNewSampleRecord(batchnumber, timestamp, samples)
Dim randomnum As Integer = CInt((1000 * ChartSupport.GetRandomDouble()))
Dim batchidstring As String = "EC" & randomnum.ToString()
Me.ChartData.AddAxisUserDefinedString(batchidstring)
```

See the example program BatchAttributeControlCharts.PercentDefectivePartsControlChart for a complete example.

Changing Default Characteristics of the Chart

All *Attribute Control Charts* have one distinct graph with its own set of properties. This graph is the Primary Chart.



Title: Fraction Defective (p) Chart Part No.: 321 Chart No.: 19
Part Name: Pre-paint touchup Operation:
Operator: S. Kafka Machine: #1
Date: 11/22/2005 11:14:35 AM

Time	11:14	11:44	12:14	12:44	13:14	13:44	14:14	14:44	15:14	15:44	16:14	16:44	17:14	17:44	18:14	18:44	19:14
#1	0	3	3	6	1	4	5	2	5	6	1	0	1	4	1	0	3
#2	1	3	4	6	1	4	4	2	2	2	1	0	2	6	0	1	2
#3	4	2	3	3	1	4	4	3	4	3	3	0	2	3	1	0	2
#4	2	4	0	7	1	2	0	1	3	2	3	0	5	3	3	2	6
#5	7	6	8	11	2	9	8	6	8	11	6	0	10	9	5	3	11
FRACT. DEF.	0.140	0.120	0.160	0.220	0.040	0.180	0.160	0.120	0.160	0.220	0.120	0.000	0.200	0.180	0.100	0.060	0.220
NO. INSP.	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
NOTES	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N

Logically enough, the properties of the objects that make up each of these graphs are stored in a property named PrimaryChart. Once the graph is initialized (using the **InitSPCEventAttributeControlChart, InitSPCTimeAttributeControlChart**, or **InitSPCBatchAttributeControlChart** method), you can modify the default characteristics of each graph using these properties.

```
// Initialize the SPCTimeAttributeControlChart
this.InitSPCTimeAttributeControlChart(charttype, numcategories,
    numsamplespersubgroup, numdatapointsinview, timeincrementminutes);

.
.
.
.

this.GraphStartPosX = 0.2;
// Enable scroll bar
this.EnableScrollBar = true;
this.PrimaryChart.DisplayFrequencyHistogram = true;
this.PrimaryChart.XAxis.LineColor = Color.Blue;
this.PrimaryChart.XAxis.LineWidth = 3;
this.PrimaryChart.ProcessVariableData.LineMarkerPlot.LineAttributes.PrimaryColor =
Color.Black;this.PrimaryChart.ProcessVariableData.LineMarkerPlot.SymbolAttributes.
PrimaryColor = Color.BlueViolet;
this.PrimaryChart.ProcessVariableData.LineMarkerPlot.SymbolAttributes.FillColor =
Color.Beige;

this.PrimaryChart.GraphBackground.FillColor = Color.LightGray;
this.PrimaryChart.PlotBackground.FillColor = Color.LightGoldenrodYellow;
this.PrimaryChart.FrequencyHistogramChart.PlotBackground.FillColor =
Color.LightGoldenrodYellow;
```

The **PrimaryChart** object is an instance of the **SPCChartObjects** class. The **SPCChartObjects** class contains the objects needed to display a single graph. Below you will find a summary of the class properties.

Public Instance Properties

AnnotationArray
Get the array of TextObject objects, representing the annotations of the chart.

AnnotationFont
Set/Get annotation font.

AnnotationNudge
Set/Get the x and y-values use to offset a data points annotation with respect to the actual data point.

AxisLabelFont
Set/Get the font used to label the x- and y-axes.

AxisTitleFont
Set/Get the font used for the axes titles.

ControlLabelPosition
Set/Get that numeric label for a control limit is placed inside, or outside the plot area INSIDE_PLOTAREA.

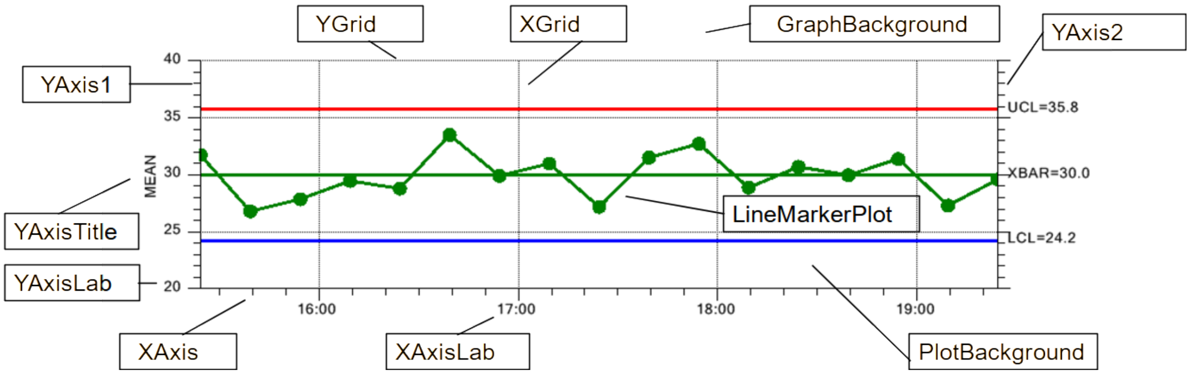
ControlLimitData
Get the array of the plot objects associated with control limits.

Datatooltip	Get a reference to the charts tooltip.
DefaultChartBackgroundColor	Get/Set the default background color for the graph area.
DefaultNumberControlLimits	Set/Get the number of control limits in the chart.
DefaultPlotBackgroundColor	Get/Set the default background color for the plot area.
DisplayChart	Set to true to enable the drawing of this chart.
DisplayFrequencyHistogram	Set to true to enable the drawing of the frequency histogram attached to the chart.
FrequencyHistogramChart	Get a reference to the optional frequency histogram attached to the chart.
GraphBackground	Get a reference to the charts graph background object.
BatchIncrement	Set/Get increment between adjacent samples of Batch type charts that use a numeric x-scale.
BatchStartValue	Set/Get the starting numeric value of the x-scale for Batch type charts that use a numeric x-scale.
BatchStopValue	Set/Get the ending numeric value of the x-scale for Batch type charts that use a numeric x-scale.
Header	Get a reference to the charts header.
HeaderFont	Set/Get the font used for the chart title.
HistogramStartPos	Set/Get the left edge, using normalized coordinates, of the frequency histogram plotting area.
HistogramOffset	Set/Get the offset of the histogram with respect to the GraphStartPosX position, using normalized coordinates, of the frequency histogram plotting area.
MaxY	Set/Get the maximum value used to scale the y-axis of the chart.
MinY	Set/Get the minimum value used to scale the y-axis of the chart.
ParentSPCChartBase	Set/Get that parent SPCChartBase object.
PlotBackground	Get a reference to the charts plot background object.
PlotMeasurementValues	Set to true to enable the plotting of all sampled values, as a scatter plot, in addition to the mean or median values.
PPhysTransform1	Gets a reference to the charts physical

EWM<A

	coordinate system.
ProcessVariableData	Holds a reference to an object encapsulating the plot object data associated with the main variable of the chart.
SampledDataData	Get the array of the sample data.
SubHead	Get a reference to the charts subhead.
SubheadFont	Set/Get the font used for the chart subhead.
TableFont	Set/Get the font used for the data table.
TextTemplate	Get/Set the text template for the data tooltip.
TimeIncrementMinutes	Get/Set the increment between adjacent samples of charts that use a numeric x-scale.
ToolTipFont	Set/Get tooltip font.
ToolTipSymbol	Get a reference to the charts tooltip symbol.
XAxis	Get a reference to the charts x-axis.
XAxisLab	Get a reference to the charts x-axis labels.
XGrid	Get a reference to the charts x-axis grid.
XValueTemplate	Get/Set the x-value template for the data tooltip.
YAxis1	Get a reference to the charts left y-axis.
YAxis2	Get a reference to the charts right y-axis.
YAxisLab	Get a reference to the charts left y-axis labels.
YAxisTitle	Get a reference to the charts left y-axis title.
YGrid	Get a reference to the charts y-axis grid.
YValueTemplate	Get/Set the y-value template for the data tooltip.

The main objects of the graph are labeled in the graph below.



8. Named and Custom Control Rule Sets

Western Electric (WECO) Rules

The normal SPC control limit rules display at the 3-sigma level, both high and low. In this case, a simple threshold test determines if a process is in, or out of control. Once a process is brought under control using the simple 3-sigma level tests, quality engineers often want to increase the sensitivity of the control chart, detecting and correcting problems before the 3-sigma control limits are reached. Other, more complex tests rely on more complicated decision-making criteria. These rules utilize historical data and look for a non-random pattern that can signify that the process is out of control, before reaching the normal +-3 sigma limits. The most popular of these are the Western Electric Rules, also know as the WECO Rules, or WE Runtime Rules. First implemented by the Western Electric Co. in the 1920's, these quality control guidelines were codified in the 1950's and form the basis for all of the other rule sets. Different industries across the globe have have developed their own variants on the WECO Rules. Other sets of rules, common enough to have an identifying name, i.e. *named rules*, are listed below.

WECO Runtime and Supplemental Rules – Western Electric Co. - Western Electric Company (1956), *Statistical Quality Control handbook.* (1 ed.), Indianapolis, Indiana: Western Electric Co., p. v, OCLC 33858387. Sometimes the Supplemental Rules are referred to as the Montgomery Rules, after the statistical quality control expert Douglas Mongtomery. *Introduction to Statistical Quality Control* (5 ed.), Hoboken, New Jersey: John Wiley & Sons, ISBN 9780471656319

Nelson Rules – The Nelson rules were first published in the October 1984 issue of the Journal of Quality Technology in an article by Lloyd S Nelson.

AIAG Rules– The (AIAG) Automotive Industry Action Group control rules are published in the their industry group “Statistical Process Control Handbook”.

Juran Rules - Joseph M. Juran was an international expert in quality control and defined these rules in his “Juran's Quality Handbook”, McGraw-Hill Professional; 6 edition (May 19, 2010), **ISBN-10:** 0071629734

Hughes Rules – The only sources we could find for the Hughes rules were all second hand. If anyone can direct is to an original source for the Hughes Rules, please send an e-mail to support@quinn-curtis.com.

Duncan Rules – Acheson Johnston Duncan was an international expert in quality control and published his rules in the text book “Quality control and industrial statistics” (fifth edition). Irwin, 1986.

Gitlow Rules - Dr. Howard S. Gitlow is an international expert in Sigma Six, TQM and SPC. His rules are found in his book “Tools and Methods for the Improvement of Quality”, 1989, **ISBN-10: 0256056803** .

Westgard Rules – The Westgard rules are based on the work of James Westgard, a leading expert in laboratory quality management . They are considered “Laboratory quality control rules”. You can find more information about the Westgard Rules, and James Westgard at the web site: http://www.westgard.com

The rules sets have many individual rules in common. In particular, the WECO rules and the Nelson rules, have 7 out of 8 rules in common, and only differ in the fourth rule.

Western Electric (WECO) Rules

In the Western Electric Rules A process is considered out of control if any of the following criteria are met:

1. **The most recent point plots outside one of the 3-sigma control limits.** If a point lies outside either of these limits, there is only a 0.3% chance that this was caused by the normal process.

2. **Two of the three most recent points plot outside and on the same side as one of the 2-sigma control limits.** The probability that any point will fall outside the warning limit is only 5%. The chances that two out of three points in a row fall outside the warning limit is only about 1%.

3. **Four of the five most recent points plot outside and on the same side as one of the 1-sigma control limits.** In normal processing, 68% of points fall within one sigma of the mean, and 32% fall outside it. The probability that 4 of 5 points fall outside of one sigma is only about 3%.

4. **Eight out of the last eight points plot on the same side of the center line, or target value.** Sometimes you see this as 9 out of 9, or 7 out of 7. There is an equal chance that any given point will fall above or below the mean. The chances that a point falls on the same side of the mean as the one before it is one in two. The odds that the next point will also fall on the same side of the mean is one in four. The probability of getting eight points on the same side of the mean is only around 1%.

These rules apply to both sides of the center line at a time. Therefore, there are eight actual alarm conditions: four for the above center line sigma levels and four for the below center line sigma levels.

There are also additional WE Rules for trending. These are often referred to as WE Supplemental Rules. Don't rely on the rule number, often these are listed in a different order.

5. **Six points in a row increasing or decreasing.** The same logic is used here as for rule 4 above. Sometimes this rule is changed to seven points rising or falling.

6. **Fifteen points in a row within one sigma.** In normal operation, 68% of points will fall within one sigma of the mean. The probability that 15 points in a row will do so, is less than 1%.

7. **Fourteen points in a row alternating direction.** The chances that the second point is always higher than (or always lower than) the preceding point, for all seven pairs is only about 1%.

8. **Eight points in a row outside one sigma.** Since 68% of points lie within one sigma of the mean, the probability that eight points in a row fall outside of the one-sigma line is less than 1%.

The rules are described as they appear in the literature. In many cases, a given rule actually specifies two test conditions; the first being a value N out of M above a plus sigma control limit, and the second being a value N out of M below a minus sigma control limit. Examples of this are rules #1, #2 and #3 for WECO and Nelson rules. In other cases, similar rules only contain one test case; N out of M above (or below) a given sigma control limit. Example of this are the Juran rules #2..#5, Hughes Rules #2..#9, Gitlow Rules #2..#5, and Duncan Rules #2..#5.

While the list of named rules below follow what is presented in the literature, the actual rule numbering should be ignored. That is because in the software we implement all rules as simple single condition rules. The first rule in all of the named rule sets is implemented as two rules; a single point greater than 3-sigma; and a single point less than -3-sigma. And WECO and Nelson rules #2 and #3 are implemented as four rules; two N out of M greater than x-sigma condition limits, and two N out of M less than x-sigma condition limits. A complete cross reference to the named rules listed below, and our own rule number system is found in Table 1. This is important, because when you try to access a particular named rule within the software, you must use our rule number system.

Basic Rules

The Basic Rules are the default rules for all of the SPC charts. They correspond to the +-3-sigma rules used by almost every industry standard SPC chart implementation.

1. One of one point is outside of +-3-sigma control limits

Nelson Rules

The Nelson rules are almost identical to the combination of the WECO Runtime and Supplemental Rules. The only difference is in Rule #4.

4. Nine out of the last nine points plot on the same side of the center line, or target value.

AIAG Rules

1. One of one point is outside of +-3-sigma control limits
2. Seven out of seven are above or below center line
3. Seven points in a row increasing
4. Seven points in a row decreasing

Juran Rules

1. One of one point is outside of +- 3-sigma control limits
2. Two of three points above 2-sigma control limits
3. Two of three points below -2-sigma control limits
4. Four of five points is above 1-sigma control limits
5. Four of five points is below -1-sigma control limit s
6. Six points in a row increasing
7. Six points in a row decreasing
8. Nine out of nine are above or below center line
9. Eight points in a row on both sides of center line, none in zone C

Hughes Rules

1. One of one point is outside of +- 3-sigma control limits
2. Two of three points above 2-sigma control limits
3. Two of three points below -2-sigma control limit s
4. Three of seven points above 2-sigma control limits
5. Three of seven points below -2-sigma control limit s
6. Four of ten points above 2-sigma control limits
7. Four of ten points below -2-sigma control limits
8. Four of five points is above 1-sigma control limits
9. Four of five points is below -1-sigma control limits
10. Seven points in a row increasing
11. Seven points in a row decreasing
12. Ten of eleven are above center line
13. Ten of eleven are below center line
14. Twelve of fourteen are above center line
15. Twelve of fourteen are below center line

Gitlow Rules

1. One of one point is outside of +- 3-sigma control limits
2. Two of three points above 2-sigma control limits
3. Two of three points below -2-sigma control limits
4. Four of five points is above 1-sigma control limits
5. Four of five points is below -1-sigma control limits
6. Eight points in a row increasing
7. Eight points in a row decreasing
8. Eight out of Eight are above center line

9. Eight out of Eight are below center line

Duncan Rules

1. One of one point is outside of +- 3-sigma control limits
2. Two of three points above 2-sigma control limits
3. Two of three points below -2-sigma control limits
4. Four of five points is above 1-sigma control limits
5. Four of five points is below -1-sigma control limits
6. Seven points in a row increasing
7. Seven points in a row decreasing

Westgard Rules

1. One of one point is outside of +- 3-sigma control limits - 13s
2. Two of two points outside +-2-sigma control limits - 22s
3. Four of four points outside +-1-sigma control limits - 41s
4. Ten of ten points on one side of center line - 10x
5. Two adjacent points on opposite sides of +-2-sigma - R4s
6. Seven of seven points in a trend increasing or decreasing - 7T
7. One of one point is outside of +- 2-sigma control limits – 12s
8. Two of three points outside +-2-sigma control limits - 2of32s
9. Three of three points outside +-1-sigma control limits - 31s
10. Six of six points on one side of center line - 6x
11. Eight of eight points on one side of center line - 8x
12. Nine of nine points on one side of center line - 9x
13. Twelve of twelve points on one side of center line – 12x

By default, only the first six Westgard rules described above are enabled. The others can be turned on using the **UseNamedRuleSet** method and setting *ruleflags* array elements true for the additional rules. Make sure you use our rule numbers and not the rule numbering above.

The Levey-Jennings chart uses its own subset of these rules:

- One of one point is outside of +- 3-sigma control limits – 13s
- Two of two points outside +-2-sigma control limits – 22s
- Four of four points outside +-1-sigma control limits – 41s
- Ten of ten points on one side of center line - 10x
- Two adjacent points on opposite sides of +-2-sigma – R4s

- One of one point is outside of +- 2-sigma control limits – 12s

with the others being optional.

- Seven of seven points in a trend increasing or decreasing - 7T
- Two of three points outside +-2-sigma control limits - 2of32s
- Three of three points outside +-1-sigma control limits - 31s
- Six of six points on one side of center line - 6x
- Eight of eight points on one side of center line - 8x
- Nine of nine points on one side of center line - 9x
- Twelve of twelve points on one side of center line – 12x

Control Rule Templates

All of the named rules fall into one of our standard rule categories. Each rule category is a flexible template which can be used to evaluate a test condition across a wide range of parameters. A list of the template categories appears below.

Standardized Templates for Control Rule Evaluation

Template #	
	Standard Control Limit tests
1	N of M above X sigma (from center line), used for UCL tests
2	N of M below X sigma (from center line), used for LCL tests
3	Reserved
4	N of M beyond X sigma (from center line, either side) or control limits – points beyond the +- limit values – don't have to all be on one side
	Trending
5	N of M trending up (increasing)
6	N of M trending down (decreasing)
7	N of M trending up (increasing) or down (decreasing)
	Hugging (lack of variance)
8	N of M within X sigma (from center line, either side)
9	N ofr M within X sigma of each other (no reference to center line)
	Oscillation
10	N of M alternating about X sigma (from center line)
11	N of M alternating (no reference to center line)

For example, rule #1 for all of the named rules (a single point plots outside of +- 3 sigma) is implemented as one instance of template #1 (N of M above X sigma, where N=1, M=1 and X = 3) and one instance of template #2 (N of M below X sigma) where N=1, M=1 and X = -3).

Rule #2 for WECO and Nelson (two of three point plots outside of +- 2 sigma) is implemented as one instance of template #1 (N of M above X sigma, where N=2, M=3 and X = 2) and one instance of template #2 (N of M below X sigma) where N=2, M=3 and X = -2).

Rule #4 and #5 for Hughes (three of seven points above/below 2-sigma control limit) is implemented as one instance of template #1 (N of M above X sigma, where N=3, M=7 and X = 2) and one instance of template #2 (N of M below X sigma) where N=3, M=7 and X = -2).

Rule #6 for Gitlow (eight points in a row increasing) is implemented as one instance of template #5 (N of M trending up) where N=8 and M=8.

The templates are important because using them you can modify any existing named rule, changing the M, N or X parameter. Or, you can create completely new rules.

Taking these factors into account, we have redefined and renumbered the rules, identifying each with the template and parameters used by each rule, .

Standardized Template Parameters and Rule # Cross Reference for Named Rules

Basic Rules			**New**			
Rule #	**Description**	**Rule #**	**Template #**	**N of**	**M**	**X**
#1	1 of 1 < 3 sigma	1	1	1	1	-3
	1 of 1 > 3 sigma	2	2	1	1	3

WECO		**New**				
Rule #	**Description**	**Rule #**	**Template #**	**N of**	**M**	**X**
#1	1 of 1 < 3 sigma	1	1	1	1	-3
	1 of 1 > 3 sigma	2	2	1	1	3
#2	2 of 3 < 2 sigma	3	1	2	3	-2
	2 of 3 > 2 sigma	4	2	2	3	2
#3	4 of 5 < sigma	5	1	4	5	-1
	4 of 5 > sigma	6	2	4	5	1
#4	8 of 8 < center line	7	1	8	8	0
	8 of 8 > center line	8	2	8	8	0

WECO+Supplemental **New**

Rule #	Description	Rule #	Template #	N of	M	X
#1	1 of 1 < 3 sigma	1	1	1	1	-3
	1 of 1 > 3 sigma	2	2	1	1	3
#2	2 of 3 < 2 sigma	3	1	2	3	-2
	2 of 3 > 2 sigma	4	2	2	3	2
#3	4 of 5 < sigma	5	1	4	5	-1
	4 of 5 > sigma	6	2	4	5	1
#4	8 of 8 < center line	7	1	8	8	0
	8 of 8 > center line	8	2	8	8	0
#5	6 of 6 incr. or dec.	9	7	6	6	0
#6	15 of 15 within 1 sigma	10	8	15	15	1
#7	14 of 14 alternating	11	11	14	14	0
#8	8 of 8 outside zone C	12	4	8	8	1

Nelson		New				
Rule #	**Description**	**Rule #**	**Template #**	**N of**	**M**	**X**
#1	1 of 1 < 3 sigma	1	1	1	1	-3
	1 of 1 > 3 sigma	2	2	1	1	3
#2	2 of 3 < 2 sigma	3	1	2	3	-2
	2 of 3 > 2 sigma	4	2	2	3	2
#3	4 of 5 < sigma	5	1	4	5	-1
	4 of 5 > sigma	6	2	4	5	1
#4	9 of 9 < center line	7	1	9	9	0
	9 of 9 > center line	8	2	9	9	0
#5	6 of 6 incr. or dec.	9	7	6	6	0
#6	15 of 15 within 1 sigma	10	8	15	15	1
#7	14 of 14 alternating	11	11	14	14	0
#8	8 points outside zone C	12	4	8	8	1

AIAG		New				
Rule #	**Description**	**Rule #**	**Template #**	**N of**	**M**	**X**
#1	1 of 1 < 3 sigma	1	1	1	1	-3
	1 of 1 > 3 sigma	2	2	1	1	3
#2	7 of 7 < center line	3	1	7	7	0
#3	7 of 7 > center line	4	2	7	7	0
#4	7 of 7 increasing	5	5	7	7	0
#5	7 of 7 decreasing	6	6	7	7	0

Juran		New				
Rule #	**Description**	**Rule #**	**Template #**	**N of**	**M**	**X**
#1	1 of 1 < 3 sigma	1	1	1	1	-3
	1 of 1 > 3 sigma	2	2	1	1	3
#2	2 of 3 < 2 sigma	3	1	2	3	-2

#3	2 of 3 > 2 sigma	4	2	2	3	2
#4	4 of 5 < sigma	5	1	4	5	-1
#5	4 of 5 > sigma	6	2	4	5	1
#6	6 of 6 increasing	7	5	6	6	0
#7	6 of 6 decreasing	8	6	6	6	0
#8	9 of 9 > center line	9	1	9	9	0
#9	9 of 9 < center line	10	2	9	9	0
#10	8 of 8 outside zone C	11	4	8	8	1

Hughes		**New**				
Rule #	**Description**	**Rule #**	**Template #**	**N of**	**M**	**X**
#1	1 of 1 < 3 sigma	1	1	1	1	-3
	1 of 1 > 3 sigma	2	2	1	1	3
#2	2 of 3 < 2 sigma	3	1	2	3	-2
#3	2 of 3 > 2 sigma	4	2	2	3	2
#4	3 of 7 < 2 sigma	5	1	3	7	-2
#5	3 of 7 > 2 sigma	6	2	3	7	2
#6	4 of 10 < 2 sigma	7	1	4	10	-2
#7	4 of 10 > 2 sigma	8	2	4	10	2
#8	4 of 5 < sigma	9	1	4	5	-1
#9	4 of 5 > sigma	10	2	4	5	1
#10	7 of 7 increasing	11	5	7	7	0
#11	7 of 7 decreasing	12	6	7	7	0
#12	10 of 11 < center line	13	1	10	11	0
#13	10 of 11 > center line	14	2	10	11	0
#14	12 of 14 < center line	15	1	12	14	0
#15	12 of 14 > center line	16	2	12	14	0

Gitlow		**New**				
Rule #	**Description**	**Rule #**	**Template #**	**N of**	**M**	**X**
#1	1 of 1 < 3 sigma	1	1	1	1	-3
	1 of 1 > 3 sigma	2	2	1	1	3
#2	2 of 3 < 2 sigma	3	1	2	3	-2
#3	2 of 3 > 2 sigma	4	2	2	3	2
#4	4 of 5 < sigma	5	1	4	5	-1
#5	4 of 5 > sigma	6	2	4	5	1
#6	8 of 8 increasing	7	5	8	8	0
#7	8 of 8 decreasing	8	6	8	8	0
#8	8 of 8 < center line	9	1	8	8	0
#9	8 of 8 > center line	10	2	8	8	0

Duncan		**New**				
Rule #	**Description**	**Rule #**	**Template #**	**N of**	**M**	**X**

#1	1 of 1 < 3 sigma	1	1	1	1	-3
	1 of 1 > 3 sigma	2	2	1	1	3
#2	2 of 3 < 2 sigma	3	1	2	3	-2
#3	2 of 3 > 2 sigma	4	2	2	3	2
#4	4 of 5 < sigma	5	1	4	5	-1
#5	4 of 5 > sigma	6	2	4	5	1
#6	7 of 7 increasing	7	5	7	7	0
#7	7 of 7 decreasing	8	6	7	7	0

Westgard		**New**				
Rule #	**Description**	**Rule #**	**Template #**	**N of**	**M**	**X**
1	1 of 1 < 3 sigma	1	1	1	1	-3
	1 of 1 > 3 sigma	2	2	1	1	3
2	2 of 2 < 2 sigma	3	1	2	2	-2
	2 of 2 > 2 sigma	4	2	2	2	2
3	4 of 4 < 1 sigma	5	1	4	4	-1
	4 of 4 > 1 sigma	6	2	4	4	1
4	10 of 10 < centerline	7	1	10	10	0
	10 of 10 > centerline	8	2	10	10	0
5	R2s – 2-sigma limits	9	10	1	1	2
6	7 of 7 trending	10	7	7	7	0
7	1 of 1 > 2 sigma	11	1	1	1	-2
	1 of 1 < 2 sigma	12	2	1	1	2
8	2 of 3 > 3 sigma	13	1	2	3	-2
	2 of 3 < 3 sigma	14	2	2	3	2
9	3 of 3 > 1 sigma	15	1	3	3	-1
	3 of 3 < 1 sigma	16	2	3	3	1
10	6 of 6 < centerline	17	1	6	6	0
	6 of 6 > centerline	18	2	6	6	0
11	8 of 8 < centerline	19	1	8	8	0
	8 of 8 > centerline	20	2	8	8	0
12	9 of 9 < centerline	21	1	9	9	0
	9 of 9 > centerline	22	2	9	9	0
13	12 of 12 < centerline	23	1	12	12	0
	12 of 12 > centerline	24	2	12	12	0

For the Levey-Jennings chart, the following (the ones not crossed out) Westgard rules are in effect:

Westgard		**New**				
Rule #	**Description**	**Rule #**	**Template #**	**N of**	**M**	**X**
1	1 of 1 < 3 sigma	1	1	1	1	-3
	1 of 1 > 3 sigma	2	2	1	1	3
2	2 of 2 < 2 sigma	3	1	2	2	-2
	2 of 2 > 2 sigma	4	2	2	2	2

3	4 of 4 < 1 sigma	5	1	4	4	-1
	4 of 4 > 1 sigma	6	2	4	4	1
4	10 of 10 < centerline	7	1	10	10	0
	10 of 10 > centerline	8	2	10	10	0
5	R2s – 2-sigma limits	9	10	1	1	2
6	7 of 7 trending	10	7	7	0	
7	1 of 1 > 2 sigma	11	1	1	-2	
	1 of 1 < 2 sigma	12	2	1	2	
8	2 of 3 > 3 sigma	13	1	2	3	-2
	2 of 3 < 3 sigma	14	2	2	3	2
9	3 of 3 > 1 sigma	15	1	3	3	-1
	3 of 3 < 1 sigma	16	2	3	3	1
10	6 of 6 < centerline	17	1	6	6	0
	6 of 6 > centerline	18	2	6	6	0
11	8 of 8 < centerline	19	1	8	8	0
	8 of 8 > centerline	20	2	8	8	0
12	9 of 9 < centerline	21	1	9	9	0
	9 of 9 > centerline	22	2	9	9	0
13	12 of 12 < centerline	23	1	12	12	0
	12 of 12 > centerline	24	2	12	12	0

Implementing a Named Rule Set

You are able to add a named rule set to an SPC application using a single call. Call the **UseNamedRuleSet** method, passing in the appropriate rule ID.

SPCChartObjects.UseNamedRuleSet Method

Visual Basic (Declaration)

```
Public Sub UseNamedRuleSet ( _
        ruleset As Integer _
)
```

```
Public Sub UseNamedRuleSet ( _
        ruleset As Integer, _
        ruleflags As BoolArray

_
)
```

C#

```
public void UseNamedRuleSet(
        int ruleset
)
```

```
public void UseNamedRuleSet(
        int ruleset,
        BoolArray ruleflags
)
```

Parameters

ruleset

Type: System..::.Int32

One of the SPCControlLimitRecord named rule indentifiers: BASIC_RULES, WECO_RULES,WECOANDSUPP_RULES, NELSON_RULES,AIAG_RULES, JURAN_RULES, HUGHES_RULES,GITLOW_RULES, WESTGARD_RULES, and DUNCAN_RULES.

ruleflags

Type: com.quinn-curtis.chart2dnet6..::.BoolArray

An array of bool, one for each named rule in the rule set. ///

Special Note for Levey-Jennings chart users:

You do not need to specify Westgard Rules for the Levey-Jennings chart. They are added automatically when the chart is created.

[C#]

```
// Use the standard +-3-sigma control limits. Equivalent to WECO control rules #1 and #2.
this.PrimaryChart.UseNamedRuleSet(SPCControlLimitRecord.BASIC_RULES);

// Use the WECO rules.  This corresponds WECO rules #1... #4, which in our organization of the
rules is #1 to #8
this.PrimaryChart.UseNamedRuleSet(SPCControlLimitRecord.WECO_RULES);

// Use the WECO rules.  This corresponds WECO rules #1... #8 , which in our organization of the
rules is #1 to #12
this.PrimaryChart.UseNamedRuleSet(SPCControlLimitRecord.WECOANDSUPP_RULES);

// Use the complete set of Nelson rules.
this.PrimaryChart.UseNamedRuleSet(SPCControlLimitRecord.NELSON_RULES);

// Use the complete set of AIAG rules.
this.PrimaryChart.UseNamedRuleSet(SPCControlLimitRecord.AIAG_RULES);

// Use the complete set of Juran rules.
this.PrimaryChart.UseNamedRuleSet(SPCControlLimitRecord.JURAN_RULES);

// Use the complete set of Hughes rules.
this.PrimaryChart.UseNamedRuleSet(SPCControlLimitRecord.HUGHES_RULES);
```

```
// Use the complete set of Gitlow rules.
this.PrimaryChart.UseNamedRuleSet(SPCControlLimitRecord.GITLOW_RULES);

// Use the complete set of Westgard rules.
this.PrimaryChart.UseNamedRuleSet(SPCControlLimitRecord.WESTGARD_RULES);

// Use the complete set of Duncan rules.
this.PrimaryChart.UseNamedRuleSet(SPCControlLimitRecord.DUNCAN_RULES);
```

[VB]

```
' Use the standard +-3-sigma control limits. Equivalent to WECO control rules #1 and #2.
Me.PrimaryChart.UseNamedRuleSet(SPCControlLimitRecord.BASIC_RULES)

' Use the WECO rules.  This corresponds WECO rules #1... #4, which in our organization of the
rules is #1 to #8
Me.PrimaryChart.UseNamedRuleSet(SPCControlLimitRecord.WECO_RULES)
.
.
.
' Use the complete set of Duncan rules.
Me.PrimaryChart.UseNamedRuleSet(SPCControlLimitRecord.DUNCAN_RULES)
```

The named rule setup methods also come in variants which allow you to selectively enable/disable one or more rules from the named rule set. You do that by building a BoolArray object, containing true or false for each named rule in the rule set, and calling **UseNamedRuleSet** method. The **GetInitializedRuleBoolArray** is just a simple utility method which creates and fills out an appropriately sized BoolArray array with a default true, or false value.

SPCChartObjects.GetInitializedRuleBoolArray Method

Visual Basic (Declaration)

```
Public Function GetInitializedRuleBoolArray ( _
        ruleset As Integer, _
        initialvalue As Boolean _
) As BoolArray
```

C#

```
public BoolArray GetInitializedRuleBoolArray(
        int ruleset,
        bool initialvalue
)
```

Parameters

ruleset

Type: System..::.Int32
One of the SPCControlLimitRecord named rule identifiers: BASIC_RULES, WECO_RULES,WECOANDSUPP_RULES, NELSON_RULES,AIAG_RULES, JURAN_RULES, HUGHES_RULES,GITLOW_RULES, WESTGARD_RULES, and DUNCAN_RULES.

initialvalue

Type: System..::.Boolean
True or False, all values of the returned BoolArray are initialized to this value.

*All rule numbering is based on our rule numbering, which separates greater than and less than tests into separate rules, as detailed in the previous tables.

[C#]

```
BoolArray ruleflags = this.PrimaryChart.GetInitializedRuleBoolArray(
    SPCControlLimitRecord.NELSON_RULES, true);
ruleflags[9] = false;
ruleflags[10] = false;
this.PrimaryChart.UseNamedRuleSet(SPCControlLimitRecord.NELSON_RULES , ruleflags);
```

[VB]

```
Dim ruleflags As BoolArray =
    Me.PrimaryChart.GetInitializedRuleBoolArray(SPCControlLimitRecord.NELSON_RULES, True)
  ruleflags(9) = False
  ruleflags(10) = False

 Me.PrimaryChart.UseNamedRuleSet(SPCControlLimitRecord.NELSON_RULES)
```

The example above disables rule #9 (6 of 6 increasing or decreasing) and rule #10 (15 of 15 within 1 sigma) from the Nelson Rules. You use our rule numbering system for specifying which rule.

See the example program RulesRulesRules for examples of how to setup an SPC chart for a given set of control rules. Once you add a set of named control rules to your SPC chart, the next thing you will want to do is set the control limits. You can either set the limits using known values, or you can have our software calculate the limits using previously acquired sample data.

If you want to explicitly set the limits you must know the historical process mean (also called the center line) and the historical process sigma. You may already know your process sigma, or you may need to calculate it as 1/3 * (UCL – process mean), where UCL is your historical +3-sigma upper control limit. Once you have those two values, everything else is automatic. Just call the **UpdateControlLimitUsingMeanAndSigma** method. It will run through all of the control limit records and fill out the appropriate limit values and other critical parameters.

SPCControlChartData.UpdateControlLimitsUsingMeanAndSigma Method

Visual Basic (Declaration)

```
Public Sub UpdateControlLimitsUsingMeanAndSigma ( _
        chartpos As Integer, _
        mean As Double, _
        sigma As Double _
)
```

C#

```
public void UpdateControlLimitsUsingMeanAndSigma(
        int chartpos,
        double mean,
        double sigma
)
```

Parameters

chartpos

Type: System..::.Int32
specifiy either SPC_PRIMARY_CONTROL_TARGET or
SPC_SECONDARY_CONTROL_TARGET

mean

Type: System..::.Double
specify the process mean.

sigma

Type: System..::.Double
specify the process sigma.

```
double processMean = your process mean
double processSigma = your process sigma
this.ChartData.UpdateControlLimitsUsingMeanAndSigma(SPCChartObjects.PRIMARY_CHART,
    processMean, processSigma);
```

The center line value and sigma have different meanings for the Primary and Secondary charts. So the **UpdateControlLimitsUsingMean** and Sigma applies to only one at a time. If you use it for the secondary chart control limits, use your historical center line value for the secondary chart type you are using. Calculate the sigma value as 1/3 * (UCL – center line), where UCL is your historical +3-sigma upper control limit for your secondary chart.

You can also auto-calculate the control limits by adding test data to your application (fed into the chart using the **AddNewSampleRecord** method), and calling **AutoCalculatedControlLimits.** This fills out the **SPCControlLimit** record for each control rule of the named rule set and makes control limit checking possible. You will find the AutoCalculateControlLimits method used in all of SPC charts of the RulesRulesRules example program.

[C#]

```
this.PrimaryChart.UseNamedRuleSet(SPCControlLimitRecord.WECO_RULES);

// Must have data loaded before any of the Auto.. methods are called
 SimulateData(100, 20);

// Calculate the SPC control limits for both graphs of the current SPC chart
this.AutoCalculateControlLimits();
```

[VB]

```
Me.PrimaryChart.UseNamedRuleSet(SPCControlLimitRecord.WECO_RULES)

' Must have data loaded before any of the Auto.. methods are called
 SimulateData(100, 20)

' Calculate the SPC control limits for both graphs of the current SPC chart
Me.AutoCalculateControlLimits()
```

Modifying Existing Named Rules

Perhaps you like everything about a named rule set, except for one or more rules. For example, you want to use the Hughes rules, but want to change the N of M parameters of rules #15 and #16. You do that using the **SPCControlLimitRecord** method **SetCustomRuleParameters**.

SPCControlLimitRecord.SetCustomRuleParameters Method

Visual Basic (Declaration)

```
Public Sub SetCustomRuleParameters ( _
        valuesincalc As Integer, _
        numvaluesforruleviolation As Integer _
)
```

C#

```
public void SetCustomRuleParameters(
        int valuesincalc,
        int numvaluesforruleviolation
)
```

C#

```
public void SetCustomRuleParameters(
        int valuesincalc,
        int numvaluesforruleviolation
)
```

Parameters

valuesincalc

Type: System..::.Int32

Specifies the number of values to use in calculation.

numvaluesforruleviolation

Type: System..::.Int32

Specifies the number of values that must be outside alarm limit for rule violation.

The example below changes the N of M parameters of Hughes rules #15 and #16 from their default N of M value (12 of 14), to the values (15 of 18).

[C#]

```
this.PrimaryChart.UseNamedRuleSet(SPCControlLimitRecord.HUGHES_RULES);

int M = 18;
int N = 15;
SPCControlLimitRecord clr =
     this.ChartData.GetNamedControlRuleRecord(SPCControlLimitRecord.HUGHES_RULES, 15);
clr.SetCustomRuleParameters(M, N);
clr = this.ChartData.GetNamedControlRuleRecord(SPCControlLimitRecord.HUGHES_RULES, 16);
clr.SetCustomRuleParameters(M, N);
```

[VB]

```
Me.PrimaryChart.UseNamedRuleSet(SPCControlLimitRecord.HUGHES_RULES)

Dim M As Integer = 18
Dim N As Integer = 15
Dim clr As SPCControlLimitRecord =
   Me.ChartData.GetNamedControlRuleRecord(SPCControlLimitRecord.HUGHES_RULES, 15)
clr.SetCustomRuleParameters(M, N)
clr = Me.ChartData.GetNamedControlRuleRecord(SPCControlLimitRecord.HUGHES_RULES, 16)
clr.SetCustomRuleParameters(M, N)
```

When trying to access the **SPCControlLimitRecord** for a given control rule, it is very important that you use the **GetNamedControlRuleRecord** method, not the **GetControlLimitRecord** method. This is because the array indexing of the **ControlLimitRecords** does not match the named control rule numbering, i.e., **GetNamedControlRuleRecord**(15) and **GetControlLimitRecord**(15) return different SPCControlLimitRecords.

If you want to enable/disable a specific rule, after they have all been created,

[C#]

```
SPCControlLimitRecord clr =
        this.ChartData.GetNamedControlRuleRecord(SPCControlLimitRecord.AIAG_RULES, 3);
clr.AlarmEnable = false;
```

[VB]

```
SPCControlLimitRecord clr =
        Me.ChartData.GetNamedControlRuleRecord(SPCControlLimitRecord.AIAG_RULES, 3)
clr.AlarmEnable = false
```

Creating Custom Rules Sets Based on Named Rules

You can create your own custom set of rules, mixing and matching rules from the standard, named rule sets, using the the **AddControlRule** method. Also, you can invent your own rules, based on one of our standard templates, and add those rules to your custom rule set.

SPCChartObjects.AddControlRule Method

Visual Basic (Declaration)

```
Public Function AddControlRule ( _
        ruleset As Integer, _
        rulenum As Integer _
) As Integer
```

C#

```
public int AddControlRule(
        int ruleset,
        int rulenum
)
```

Parameters

ruleset

Type: System..::.Int32
One of the SPCControlLimitRecord named rule indentifiers: BASIC_RULES, WECO_RULES,WECOANDSUPP_RULES, NELSON_RULES,AIAG_RULES, JURAN_RULES, HUGHES_RULES,GITLOW_RULES, WESTGARD_RULES, and DUNCAN_RULES.

rulenum

Type: System..::.Int32
The rule number (our rule number) ///

Even if you do not call one of the **Use..Rules** methods, you still end up with four control limits. These correspond to the +-3-sigma control limits, for both the Primary and Secondary (were applicable) chart. So, you do not need to add those to your custom set of rules. Start with the rules you want to add after the standard +-3-sigma rules. If, for some reason you cannot live with the default +-3-sigma rules, you can disable them with a call to **EnableDefaultLimits**(false, false).

Say you want to create custom rule set for the Primary chart, combining rules from Nelson (#3, #4), Juran (#5, #5), AIAG (#3,#4), Hughes (#12) and Duncan (#8). Make the following calls in the setup portion of your program.

[C#]

```
this.PrimaryChart.AddControlRule(SPCControlLimitRecord.NELSON_RULES, 3);
this.PrimaryChart.AddControlRule(SPCControlLimitRecord.NELSON_RULES, 4);
this.PrimaryChart.AddControlRule(SPCControlLimitRecord.JURAN_RULES, 5);
this.PrimaryChart.AddControlRule(SPCControlLimitRecord.JURAN_RULES, 6);
this.PrimaryChart.AddControlRule(SPCControlLimitRecord.AIAG_RULES, 3);
this.PrimaryChart.AddControlRule(SPCControlLimitRecord.AIAG_RULES, 4);
this.PrimaryChart.AddControlRule(SPCControlLimitRecord.HUGHES_RULES, 12);
this.PrimaryChart.AddControlRule(SPCControlLimitRecord.DUNCAN_RULES, 8);
```

[VB]

```
 Me.PrimaryChart.EnableDefaultLimits(False, False)
 Me.PrimaryChart.AddControlRule(SPCControlLimitRecord.WECO_RULES, 1)
 Me.PrimaryChart.AddControlRule(SPCControlLimitRecord.WECO_RULES, 2)
 Me.PrimaryChart.AddControlRule(SPCControlLimitRecord.NELSON_RULES, 3)
 Me.PrimaryChart.AddControlRule(SPCControlLimitRecord.NELSON_RULES, 4)
 Me.PrimaryChart.AddControlRule(SPCControlLimitRecord.JURAN_RULES, 5)
 Me.PrimaryChart.AddControlRule(SPCControlLimitRecord.JURAN_RULES, 6)
 Me.PrimaryChart.AddControlRule(SPCControlLimitRecord.AIAG_RULES, 3)
 Me.PrimaryChart.AddControlRule(SPCControlLimitRecord.AIAG_RULES, 4)
 Me.PrimaryChart.AddControlRule(SPCControlLimitRecord.HUGHES_RULES, 12)
 Me.PrimaryChart.AddControlRule(SPCControlLimitRecord.DUNCAN_RULES, 8)
```

Normally there will be no reason to set custom rules for the secondary chart, since all of the named rules apply to the Primary chart. Nothing stops you from doing it though. We can't say whether or not it makes sense statistically.

Creating Custom Rules Sets Based on a Template

Add your own custom rule to the rule set using another version of the **AddControlRuleRecord** method. This one specifies a template, N out of M values, a sigma level to attach the control rule to, and a flag on whether or not to display a limit line for the control rule. If you have multiple control rules attached to a given sigma level, you should only display a control line for one of them.

SPCChartObjects.AddControlRule Method

Visual Basic (Declaration)

```
Public Function AddControlRule ( _
        template As Integer, _
        n As Integer, _
        m As Integer, _
        sigmavalue As Double, _
        displaylimitline As Boolean _
) As Integer
```

C#

```
public int AddControlRule(
        int template,
        int n,
        int m,
        double sigmavalue,
        bool displaylimitline
)
```

Parameters

template

Type: System..::.Int32

Specifies the standardized template number.

n

Type: System..::.Int32

The n value for the n of m condition ///

m

Type: System..::.Int32

The m value for the n of m condition ///

sigmavalue

Type: System..::.Double

The sigma value associated with the limit ///

displaylimitline

Type: System..::.Boolean

True to display a limit line for the control limit ///

In your code it would something like:

[C#]

```
int template = 2; // N of M  Greater than limit value
int N = 10;
int M = 13;
double sigmavalue = 2; // control limit value is the +2-sigma value
bool displaylimitline = false; // no limit line.
this.PrimaryChart.AddControlRule(template, N, M,
    sigmavalue, displaylimitline);
```

[VB]

```
Dim template As Integer = 2
 ' N of M  Greater than limit value
 Dim N As Integer = 10
 Dim M As Integer = 13
 Dim sigmavalue As Double = 2
 ' trending is not attached to a sigma value
 Dim displaylimitline As Boolean = False
 ' no limit line.
 Me.PrimaryChart.AddControlRule(template, N, M, sigmavalue, displaylimitline)
```

Creating Custom Rules Not Associated With Sigma Levels

Most of the preceding control rules are based on the mean and sigma of the current control chart. The trending rules (N of M increasing/ decreasing) are an exception, because they don't use the mean or sigma value anywhere in their evaluation. Regardless, since many of the named rules include trending rules, they are included with the previous section. There are a couple of other control rules not directly related to the mean and sigma value of the chart. The first is a simple numeric limit. The limit is meant to be independent of the mean and sigma level of the chart. It can also be considered a specification limit. We have three routines you can use to add a numeric control limit to a chart. The first, **AddSpecLimit**, creates a specification limit which monitors the main variable of the chart, and compares its value to the specified numeric threshold. It will display in the chart as a line with a limit string to the right of the plot area.

SPCChartObjects.AddSpecLimit Method

Visual Basic (Declaration)

```
Public Function AddSpecLimit ( _
        speclimittype As Integer, _
        value As Double, _
        displaystring As String, _
        attrib As ChartAttribute _
) As SPCControlPlotObjectData
```

C#

```
public SPCControlPlotObjectData AddSpecLimit(
        int speclimittype,
        double value,
        string displaystring,
        ChartAttribute attrib
```

)

Parameters

speclimittype

Type: System..::.Int32
Specifiy either SPCChartObjects.SPC_LOWER_SPEC_LIMIT or SPCChartObjects.SPC_UPPER_SPEC_LIMIT

value

Type: System..::.Double
Specifies the value of the specification limit.

displaystring

Type: System..::.String
The optional display string displayed to the right of the spec limit line.

attrib

Type: com.quinn-curtis.chart2dnet6..::.ChartAttribute
The line attributes of the spec limit line.

Return Value

The SPCControlPlotObjectData object of the specification limit.

[C#]

```
this.PrimaryChart.AddSpecLimit(SPCChartObjects.SPC_LOWER_SPEC_LIMIT,
    18.3, "L SPEC", new ChartAttribute(Color.Green, 3.0));
this.PrimaryChart.AddSpecLimit(SPCChartObjects.SPC_UPPER_SPEC_LIMIT,
    39.1, "H SPEC", new ChartAttribute(Color.Yellow, 3.0));
```

[VB]

```
Me.PrimaryChart.AddSpecLimit(SPCChartObjects.SPC_LOWER_SPEC_LIMIT,
    18.3, "L SPEC", new ChartAttribute(Color.Green, 3.0))
Me.PrimaryChart.AddSpecLimit(SPCChartObjects.SPC_UPPER_SPEC_LIMIT,
    39.1, "H SPEC", new ChartAttribute(Color.Yellow, 3.0))
```

The second, **AddNumericControlLimit**, creates a limit which monitors the value of one of the charts SPCCalculatedValueRecord object values. This is what you would use if you wanted to monitor a subgroup mean, range, sigma against some arbitrary control limit, since in the appropriate charts, the subgroup mean, range and sigma are SPCCalculatedValueRecord objects.

The type, and ordering of the SPCCalculatedValueRecord records is unique to each chart type. Below is a list of the chart types and the calculated values for each.

MEAN_RANGE_CHART

0. SPCCalculatedValueRecord.SPC_MEAN_CALC,
1. SPCCalculatedValueRecord.SPC_RANGE_CALC
2. SPCCalculatedValueRecord.SPC_SUM_CALC

```
MEDIAN_RANGE_CHART
        0. SPCCalculatedValueRecord.SPC_MEDIAN_CALC
        1. SPCCalculatedValueRecord.SPC_RANGE_CALC;
MEAN_SIGMA_CHART
        0. SPCCalculatedValueRecord.SPC_MEAN_CALC
        1. SPCCalculatedValueRecord.SPC_STD_DEVIATION_CALC
        2. SPCCalculatedValueRecord.SPC_SUM_CALC
MEAN_SIGMA_CHART_VSS
        0. SPCCalculatedValueRecord.SPC_MEAN_VSS_CALC
        1. SPCCalculatedValueRecord.SPC_STD_DEVIATION_VSS_CALC
        2. SPCCalculatedValueRecord.SPC_SUM_CALC
MEAN_VARIANCE_CHART
        0. SPCCalculatedValueRecord.SPC_MEAN_CALC
        1. SPCCalculatedValueRecord.SPC_VARIANCE_CALC
        2. SPCCalculatedValueRecord.SPC_SUM_CALC
INDIVIDUAL_RANGE_CHART
        0. SPCCalculatedValueRecord.SPC_INDIVIDUAL_COPY_VALUE
        1. SPCCalculatedValueRecord.SPC_INDIVIDUAL_ABS_RANGE_CAL
MAMR_CHART
        0. SPCCalculatedValueRecord.SPC_MA_CALC
        1. SPCCalculatedValueRecord.SPC_MR_CALC
MAMS_CHART
        0. SPCCalculatedValueRecord.SPC_MA_CALC
        1. SPCCalculatedValueRecord.SPC_MS_CALC
EWMA_CHART
        0. SPCCalculatedValueRecord.SPC_EWMA_CALC
        1. SPCCalculatedValueRecord.SPC_MEAN_CALC
LEVEY_JENNINGS_CHART
        0. SPCCalculatedValueRecord.SPC_INDIVIDUAL_COPY_VALUE
MA_CHART
        0. SPCCalculatedValueRecord.SPC_MA_CALC
TABCUSUM_CHAR
        0. SPCCalculatedValueRecord.SPC_CUSUM_CPLUS_CALC
        1. SPCCalculatedValueRecord.SPC_CUSUM_CMINUS_CALC
        2. SPCCalculatedValueRecord.SPC_MEAN_CALC, "MEAN"


 PERCENT_DEFECTIVE_PARTS_CHART)
        0. SPCCalculatedValueRecord.SPC_PERCENT_DEFECTIVE_PARTS_CALC
FRACTION_DEFECTIVE_PARTS_CHART
        0. SPCCalculatedValueRecord.SPC_FRACTION_DEFECTIVE_PARTS_CALC
FRACTION_DEFECTIVE_PARTS_CHART_VSS
        0. SPCCalculatedValueRecord.SPC_FRACTION_DEFECTIVE_PARTS_VSS_CALC
NUMBER_DEFECTIVE_PARTS_CHART
        0. SPCCalculatedValueRecord.SPC_TOTAL_DEFECTIVE_PARTS_CALC
NUMBER_DEFECTS_CHART
        0. SPCCalculatedValueRecord.SPC_TOTAL_DEFECTS_CALC
```

NUMBER_DEFECTS_PERUNIT_CHART
    0. SPCCalculatedValueRecord.SPC_FRACTION_DEFECTS_CALC
NUMBER_DEFECTS_PERUNIT_CHART_VSS
    0. SPCCalculatedValueRecord.SPC_FRACTION_DEFECTS_VSS_CALC
PERCENT_DEFECTIVE_PARTS_CHART_VSS
    0. SPCCalculatedValueRecord.SPC_PERCENT_DEFECTIVE_PARTS_VSS_CALC
NUMBER_DEFECTS_PER_MILLION_CHART
    0. SPCCalculatedValueRecord.SPC_TOTAL_DEFECTS_CALC
    1. SPCCalculatedValueRecord.SPC_NUMBER_DEFECTS_PER_MILLION_CALC

SPCChartObjects.AddNumericControlLimit Method

Visual Basic (Declaration)

```
Public Function AddNumericControlLimit ( _
        sourcevar As SPCCalculatedValueRecord, _
        limtype As Integer, _
        value As Double, _
        displaystring As String, _
        addline As Boolean, _
        attrib As ChartAttribute _
) As SPCControlPlotObjectData
```

C#

```
public SPCControlPlotObjectData AddNumericControlLimit(
        SPCCalculatedValueRecord sourcevar,
        int limtype,
        double value,
        string displaystring,
        bool addline,
        ChartAttribute attrib
)
```

Parameters

sourcevar

Type: com.quinncurtis.spcchartspcchartnet..::.SPCCalculatedValueRecord
The SPCCalculatedValue source of the item to test.

limtype

Type: System..::.Int32
Specifiy either SPCChartObjects.SPC_LOWERTHAN_LIMIT or SPCChartObjects.SPC_GREATERTHAN_LIMIT

value

Type: System..::.Double
Specifies the value of the control limit.

displaystring

Type: System..::.String

The optional display string displayed to the right of the limit line.

addline

Type: System..::.Boolean

True and the limit is displayed as a line in the chart.

attrib

Type: com.quinn-curtis.chart2dnet6..::.ChartAttribute

The line attributes of the spec limit line.

Return Value

The SPCControlPlotObjectData object of the numeric limit.

[C#]

```
// this.ChartData.GetCalculatedValueRecord(0) returns the calcuated value which calculates the
subgroup mean

SPCCalculatedValueRecord cvr = this.ChartData.GetCalculatedValueRecord(0);
this.PrimaryChart.AddNumericControlLimit(cvr, SPCChartObjects.SPC_LOWER_SPEC_LIMIT,
        22.3, "L CV(0)", true, new ChartAttribute(Color.Green, 3.0));
this.PrimaryChart.AddNumericControlLimit(cvr, SPCChartObjects.SPC_UPPER_SPEC_LIMIT,
        32.1, "H  CV(0)", true, new ChartAttribute(Color.Yellow, 3.0));
```

[VB]

```
' Me.ChartData.GetCalculatedValueRecord(0) returns the calcuated value which calculates the
subgroup mean
Dim cvr As SPCCalculatedValueRecord = Me.ChartData.GetCalculatedValueRecord(0)

Me.PrimaryChart.AddNumericControlLimit(cvr, SPCChartObjects.SPC_LOWER_SPEC_LIMIT,
    22.3, "L  CV(0)", True, New ChartAttribute(Color.Green, 3.0))

Me.PrimaryChart.AddNumericControlLimit(cvr, SPCChartObjects.SPC_UPPER_SPEC_LIMIT,
    32.1, "H  CV(0)", True, New ChartAttribute(Color.Yellow, 3.0))
```

The third, **AddProcessCapabilityLimit**, creates a limit which monitors the value of one of the charts SPCProcessCapabilityRecord object values.

SPCChartObjects.AddProcessCapabilityControlLimit Method

Visual Basic (Declaration)

```
Public Function AddProcessCapabilityControlLimit ( _
        pcindex As Integer, _
        limtype As Integer, _
        value As Double, _
        displaystring As String, _
        attrib As ChartAttribute _
) As SPCControlPlotObjectData
```

C#

```
public SPCControlPlotObjectData AddProcessCapabilityControlLimit(
        int pcindex,
        int limtype,
        double value,
        string displaystring,
        ChartAttribute attrib
)
```

Parameters

pcindex

Type: System..::.Int32
Specify the process capability limit index (based on the order the process capabilities were added to the chart, starting at 0.

limtype

Type: System..::.Int32
Specifiy either SPCChartObjects.SPC_LOWER_PC_LIMIT or SPCChartObjects.SPC_UPPER_PC_LIMIT

value

Type: System..::.Double
Specifies the value of the limit.

displaystring

Type: System..::.String
Text included with the alarm if it is triggered.

attrib

Type: com.quinn-curtis.chart2dnet6..::.ChartAttribute
The line attributes of the spec limit line.

Return Value

The SPCControlPlotObjectData object of the specification limit.

[C#]

```
// Attached  control limits to the first Process Capability added to the chart
this.PrimaryChart.AddProcessCapabilityControlLimit(0, SPCChartObjects.SPC_LOWER_PC_LIMIT,
    0.195, "L Cpk", new ChartAttribute(Color.Green, 3.0));
this.PrimaryChart.AddProcessCapabilityControlLimit(0, SPCChartObjects.SPC_UPPER_PC_LIMIT,
    0.25, "H Cpk", new ChartAttribute(Color.Yellow, 3.0));

// Attached  control limits to the second Process Capability added to the chart
this.PrimaryChart.AddProcessCapabilityControlLimit(1, SPCChartObjects.SPC_LOWER_PC_LIMIT,
    0.195, "L Cpm", new ChartAttribute(Color.Green, 3.0));
this.PrimaryChart.AddProcessCapabilityControlLimit(1, SPCChartObjects.SPC_UPPER_PC_LIMIT,
    0.25, "H Cpm", new ChartAttribute(Color.Yellow, 3.0));
```

[VB]

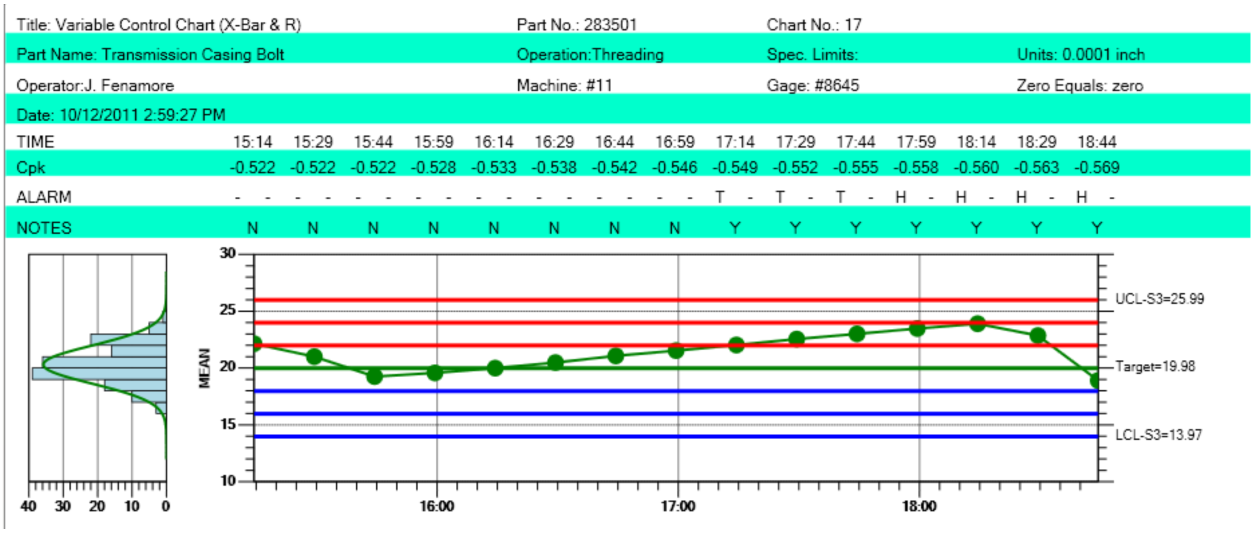
```
Me.PrimaryChart.AddProcessCapabilityControlLimit(0, SPCChartObjects.SPC_LOWER_PC_LIMIT,
    0.195, "L Cpk", New ChartAttribute(Color.Green, 3.0))
Me.PrimaryChart.AddProcessCapabilityControlLimit(0, SPCChartObjects.SPC_UPPER_PC_LIMIT,
    0.25, "H Cpk", New ChartAttribute(Color.Yellow, 3.0))

Me.PrimaryChart.AddProcessCapabilityControlLimit(1, SPCChartObjects.SPC_LOWER_PC_LIMIT,
    0.195, "L Cpm", New ChartAttribute(Color.Green, 3.0))
Me.PrimaryChart.AddProcessCapabilityControlLimit(1, SPCChartObjects.SPC_UPPER_PC_LIMIT,
    0.25, "H Cpm", New ChartAttribute(Color.Yellow, 3.0))
```

Enable Alarm Highlighting

The alarm status line above is turned on/off using the **EnableAlarmStatusValues** property. We have set it on by default, so you will have to turn it off if you don’t want it. Each sample interval has two small boxes that are labeled using one of several different characters, listed below. The most common are an “H” signifying a high alarm, a “L” signifying a low alarm, and a “-“ signifying that there is no alarm. When specialized control rules are implemented, using the named rules, or custom rules involving trending, oscillation, or stratification, a “T”, “O” or “S” may also appear.

“-”	No alarm condition
“H”	High - Measured value is above a high limit
“L”	Low - Measured value falls below a low limit
“T”	Trending - Measured value is trending up (or down).
“O”	Oscillation - Measured value is oscillating (alternating) up and down.
“S”	Stratification - Measured value is stuck in a narrow band.

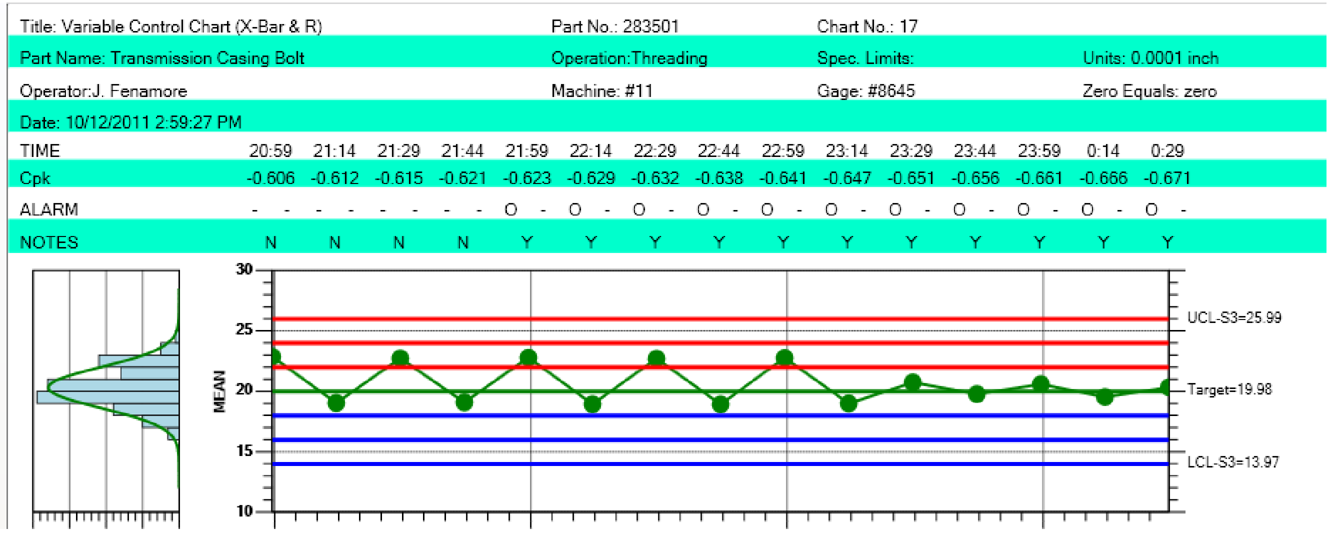
Trending



Title: Variable Control Chart (X-Bar & R)	Part No.: 283501	Chart No.: 17	
Part Name: Transmission Casing Bolt	Operation:Threading	Spec. Limits:	Units: 0.0001 inch
Operator:J. Fenamore	Machine: #11	Gage: #8645	Zero Equals: zero
Date: 10/12/2011 2:59:27 PM			

TIME	15:14	15:29	15:44	15:59	16:14	16:29	16:44	16:59	17:14	17:29	17:44	17:59	18:14	18:29	18:44
Cpk	-0.522	-0.522	-0.522	-0.528	-0.533	-0.538	-0.542	-0.546	-0.549	-0.552	-0.555	-0.558	-0.560	-0.563	-0.569
ALARM	- -	- -	- -	- -	- -	- -	- -	- -	T -	T -	T -	H -	H -	H -	H -
NOTES	N	N	N	N	N	N	N	N	Y	Y	Y	Y	Y	Y	Y

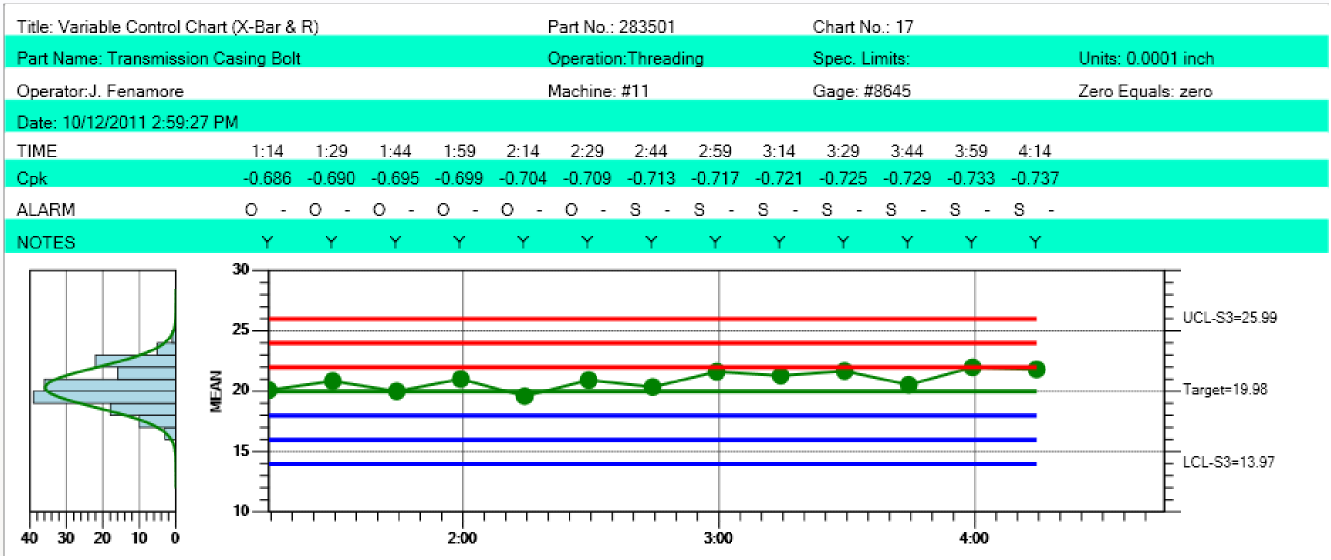
Oscillation



Title: Variable Control Chart (X-Bar & R)	Part No.: 283501	Chart No.: 17	
Part Name: Transmission Casing Bolt	Operation:Threading	Spec. Limits:	Units: 0.0001 inch
Operator:J. Fenamore	Machine: #11	Gage: #8645	Zero Equals: zero
Date: 10/12/2011 2:59:27 PM			

TIME	20:59	21:14	21:29	21:44	21:59	22:14	22:29	22:44	22:59	23:14	23:29	23:44	23:59	0:14	0:29
Cpk	-0.606	-0.612	-0.615	-0.621	-0.623	-0.629	-0.632	-0.638	-0.641	-0.647	-0.651	-0.656	-0.661	-0.666	-0.671
ALARM	- -	- -	- -	- -	O -	O -	O -	O -	O -	O -	O -	O -	O -	O -	O -
NOTES	N	N	N	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y

Stratification



Title: Variable Control Chart (X-Bar & R) Part No.: 283501 Chart No.: 17

Part Name: Transmission Casing Bolt Operation:Threading Spec. Limits: Units: 0.0001 inch

Operator:J. Fenamore Machine: #11 Gage: #8645 Zero Equals: zero

Date: 10/12/2011 2:59:27 PM

TIME	1:14	1:29	1:44	1:59	2:14	2:29	2:44	2:59	3:14	3:29	3:44	3:59	4:14
Cpk	-0.686	-0.690	-0.695	-0.699	-0.704	-0.709	-0.713	-0.717	-0.721	-0.725	-0.729	-0.733	-0.737
ALARM	O -	O -	O -	O -	O -	O -	S -	S -	S -	S -	S -	S -	S -
NOTES	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y

[C#]

```
// Alarm status line
this.EnableAlarmStatusValues = false;
```

[VB]

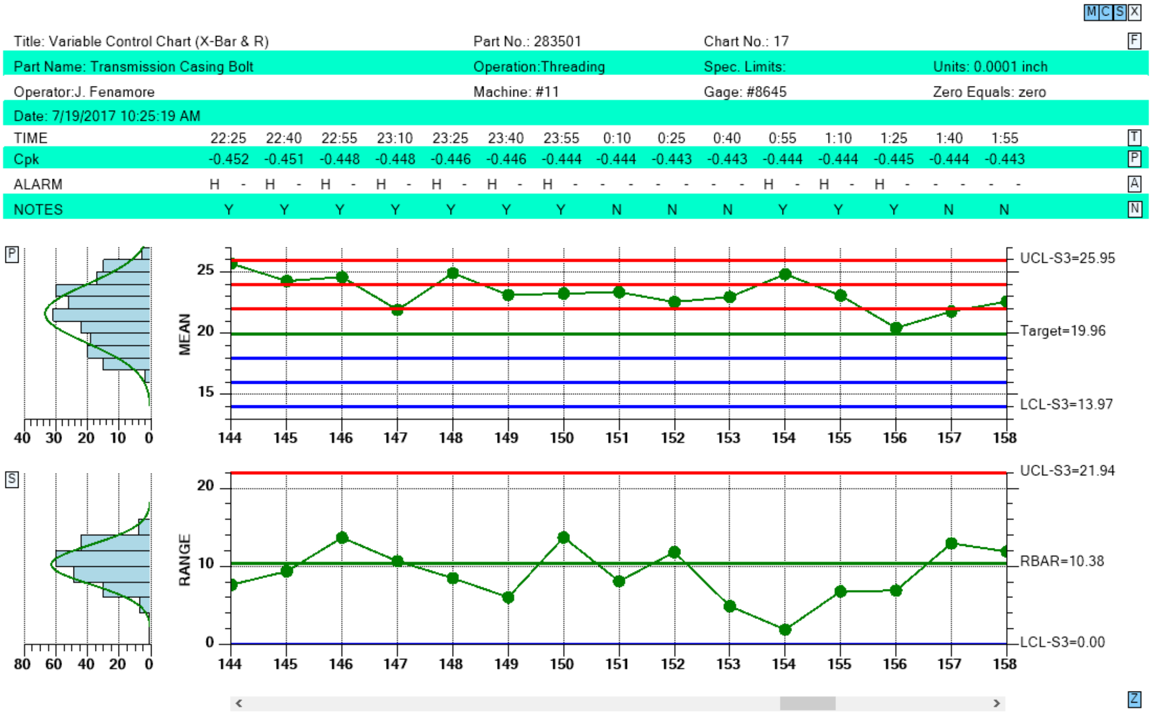
```
'Alarm status line
Me.EnableAlarmStatusValues = False
```

Reset N of M counters for control moves

There are cases when a user wants to keeps the same chart going, adding new sample intervals with new data, after the issue which caused the process to go out of control has been remedied. But many of the control limit tests found in the named control rules (WECO, Nelson, etc.) use N of M tests, where N out of M sample intervals must violate a specific rule. For example, a WECO rule says that if 4 out of 5 samples are outside of 1-sigma, it is an alarm condition. But if the process is now in control, the user may want to reset all N or M counts back to 0, exactly as if the plotting of the chart had started at the first sample interval. So we have added a reset mechanism for the N or M rules to a zero count for all rules.

Use the ChartData method ReCenterControlLimits to reset the N of M counters back to 0.

```
this.ChartData.ReCenterControlLimits();
```



Title: Variable Control Chart (X-Bar & R) Part No.: 283501 Chart No.: 17

Part Name: Transmission Casing Bolt Operation:Threading Spec. Limits: Units: 0.0001 inch

Operator:J. Fenamore Machine: #11 Gage: #8645 Zero Equals: zero

Date: 7/19/2017 10:25:19 AM

TIME	22:25	22:40	22:55	23:10	23:25	23:40	23:55	0:10	0:25	0:40	0:55	1:10	1:25	1:40	1:55
Cpk	-0.452	-0.451	-0.448	-0.448	-0.446	-0.446	-0.444	-0.444	-0.443	-0.443	-0.444	-0.444	-0.445	-0.444	-0.443
ALARM	H -	H -	H -	H -	H -	H -	H -	- -	- -	- -	H -	H -	H -	- -	- -
NOTES	Y	Y	Y	Y	Y	Y	Y	N	N	N	Y	Y	Y	N	N

The N of M counters are reset after sample interval 150 update

The example above uses the Nelson Rules. The samples intervals from 144 to 150 are shown to be in alarm, either 2 of 3 greater than 2-sigma, or 4 of 5 greater than 1-sigma. But sample interval 151 is greater than 1-sigma and should show a 4 of 5 greater than 1-sigma alarm. Yet it is not shown to be in alarm. This is because after the update for sample interval 150, the ReCenterControlLimits method was called, resetting the counters for all N or M test back to zero. So it takes four additional 1-sigma violations to re-trigger the 4 of 5 greater than 1-sigma alarm.

Remove Negative LCL control limits for Variable Control Secondary charts, or Attribute Control Primary charts

We added a routine which when called will disable (both from display and alarm checking) any chart LCL control limit if the limit value is <= a specified value, 0.0 in most cases. Use the ChartData SetAutoDeleteControlLimits method for this. We made the method more general than a simple routine to just delete negative control limits. It will remove any control limit that is (<, <=, >, >=) the specified value.

If a control limit meets the test criteria, which compares the control limit value, to the specified value, using the specified criteria, it is removed.

SetAutoDeleteControlLimits

```
public void SetAutoDeleteControlLimits(int chartpos, int compop, double value)
```

chartpos Restrict the operation to Primary or Secondary chart. Use the constant SPCChartObjects.PRIMARY_CHART or SPCChartObjects.SECONDARY_CHART

compop Use this comparison operator. Use one of the comparison operator constants:
SPCControlChartData.COMPARISON_OP_LESSTHAN,
SPCControlChartData.COMPARISON_OP_LESSTHAN_OR_EQ,
SPCControlChartData.COMPARISON_OP_GREATERTHAN,
SPCControlChartData.COMPARISON_OP_GREATERTHAN_OR_EQ,

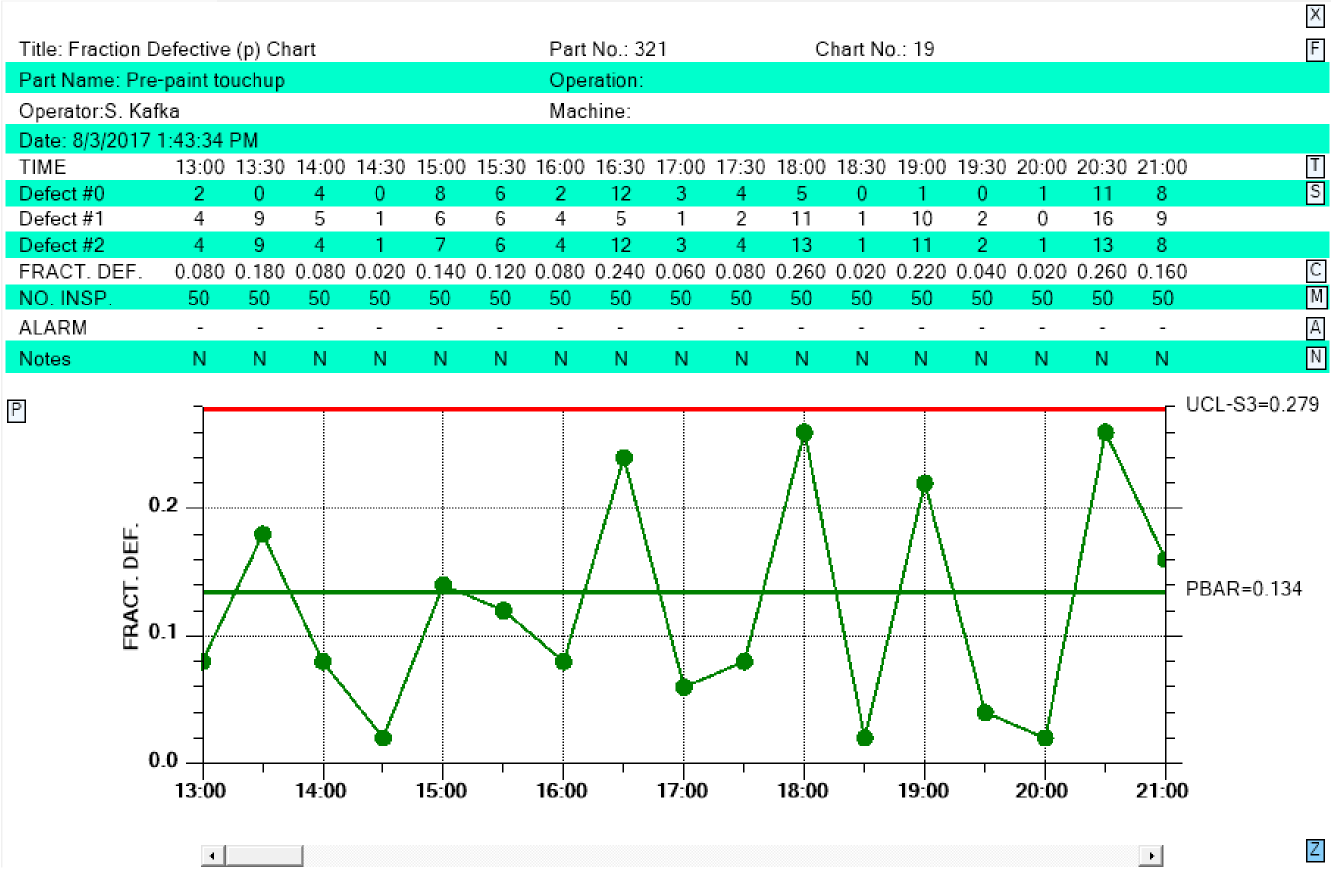
value Value used in comparison.

For a Variable Control chart, If you want to remove the LCL limit, equal to 0.0, from the Range (Secondary chart), call SetAutoDeleteControlLimits with the following parameters.

```
this.ChartData.SetAutoDeleteControlLimits(SPCChartObjects.SECONDARY_CHART,
SPCControlChartData.COMPARISON_OP_LESSTHAN_OR_EQ, 0.0);
```

For an Attributes Control chart, If you want to remove the LCL limit, equal to 0.0, from the Primary Chart, call SetAutoDeleteControlLimits with the following parameters.

```
this.ChartData.SetAutoDeleteControlLimits(SPCChartObjects.PRIMARY_CHART,
SPCControlChartData.COMPARISON_OP_LESSTHAN_OR_EQ, 0.0);
```



Title: Fraction Defective (p) Chart Part No.: 321 Chart No.: 19

Part Name: Pre-paint touchup Operation:

Operator:S. Kafka Machine:

Date: 8/3/2017 1:43:34 PM

TIME	13:00	13:30	14:00	14:30	15:00	15:30	16:00	16:30	17:00	17:30	18:00	18:30	19:00	19:30	20:00	20:30	21:00
Defect #0	2	0	4	0	8	6	2	12	3	4	5	0	1	0	1	11	8
Defect #1	4	9	5	1	6	6	4	5	1	2	11	1	10	2	0	16	9
Defect #2	4	9	4	1	7	6	4	12	3	4	13	1	11	2	1	13	8
FRACT. DEF.	0.080	0.180	0.080	0.020	0.140	0.120	0.080	0.240	0.060	0.080	0.260	0.020	0.220	0.040	0.020	0.260	0.160
NO. INSP.	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50	50
ALARM	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Notes	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N

Note that the lower control limit for the Primary chart is not present. It was calculated be 0.0, and was removed by the software.

If you plan to fill the area between the control limit lines and the center line (zone colors) you must leave the 0.0 lower control limit in place so that the software can fill between the limit and the center line.

N of M testing when the most recent point entering the test is within limits

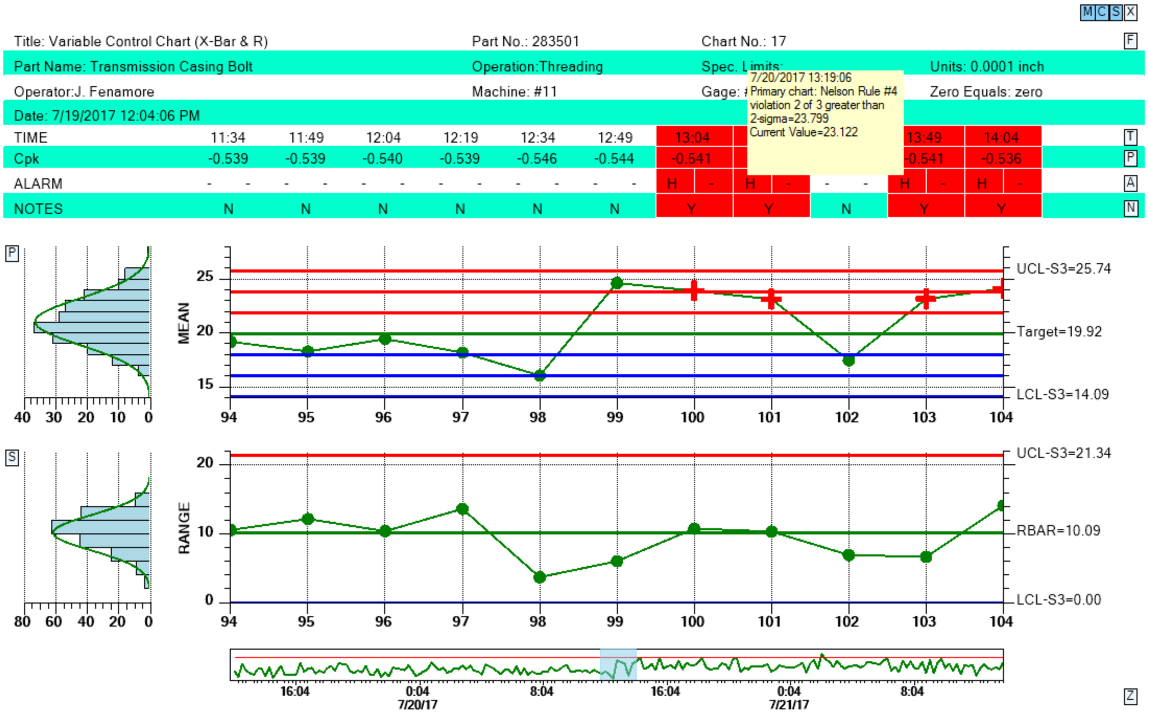
Our default mode takes a strict approach to N of M testing. Regardless of the value of the most recent point to enter the calculation (even if it is within the test limits), if N of M values are outside of limits we consider the sample interval to be in alarm. But some customers challenged this interpretation and presented us with published examples which show that if the most recent point is within limits and the previous N points out of limits, the sample interval should NOT be considered in alarm. The logic for those using the alternative evaluation scheme is that after the the first N values were found to fail the N of M test, a correction was made to the process. And therefore the next sample interval is within limits and should not be in alarm, even though it fails the strict N or M test, since it still picks up the N out of limit values before the process was corrected. We researched the issue and found no agreement. It is

implemented in the published literature both ways, going back 50 years. So a simple global flag has been added you can use to choose one evaluation method or the other, with the default being our original method.
If you want to change the N of M evaluation scheme from the default (strict N of M testing), to the alternative method, use the SPCControlLimitRecord

```
SPCControlLimitRecord.DefaultAltNofMRule = true;
```

This is a static property and once set, will affect all charts that are created for the duration of the program. So you can set it at the start of your program and not have to worry about setting in subsequent chart setups.

In the picture below, the default method of N of M valuation produces an alarm at sample interval 101. While sample 101 is within 2-sigma, the previous two samples were greater than 2-sigma, so the 2 out of 3 > 2-sigma test fails for sample interval 101.


Title: Variable Control Chart (X-Bar & R)
Part No.: 283501
Chart No.: 17
Part Name: Transmission Casing Bolt
Operation:Threading
Spec. Limits:
Units: 0.0001 inch
Operator:J. Fenamore
Machine: #11
Gage:
Zero Equals: zero
Date: 7/19/2017 12:04:06 PM

7/20/2017 13:19:06
Primary chart: Nelson Rule #4 violation 2 of 3 greater than 2-sigma=23.799
Current Value=23.122

TIME	11:34	11:49	12:04	12:19	12:34	12:49	13:04			13:49	14:04
Cpk	-0.539	-0.539	-0.540	-0.539	-0.546	-0.544	-0.541			-0.541	-0.536
ALARM	- -	- -	- -	- -	- -	- -	H -	H -	- -	H -	H -
NOTES	N	N	N	N	N	N	Y	Y	N	Y	Y

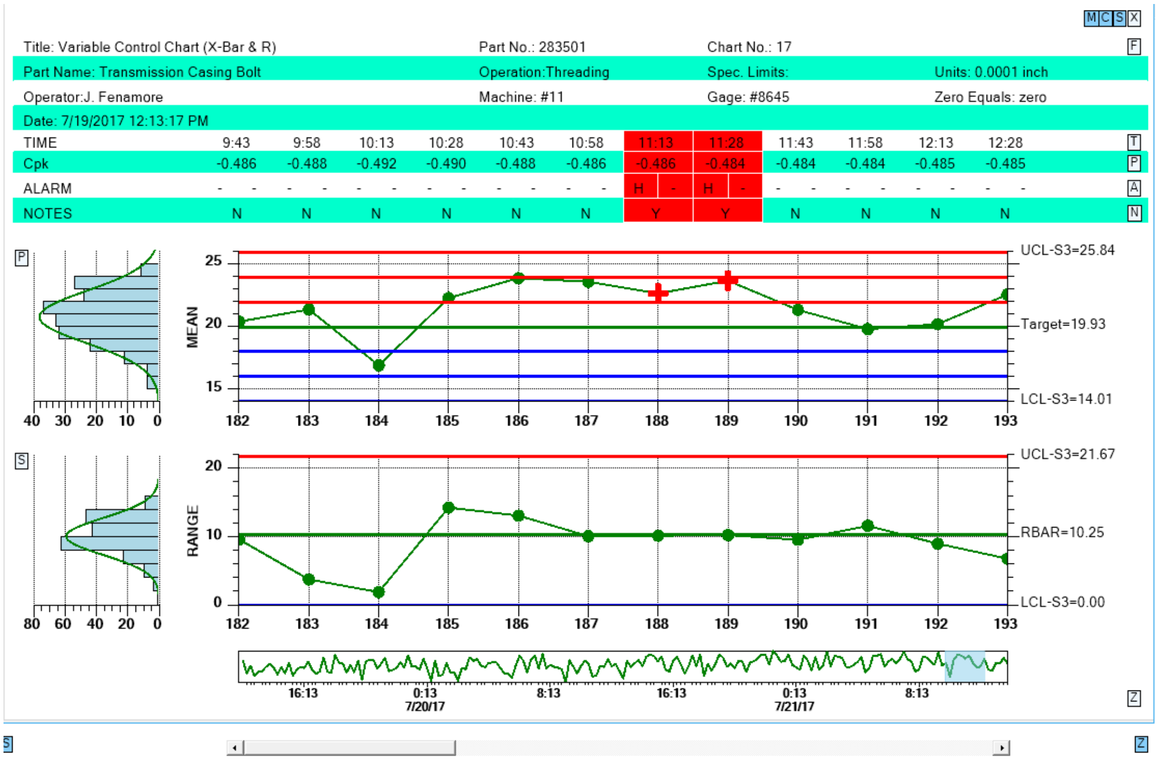
Using the default N of M testing, the sample interval 101 fails the 2 out of 3 > 2-sigma test of the WECO and Nelson rules.

If the following code was added to the chart setup

```
SPCControlLimitRecord.DefaultAltNofMRule = true;
```

sample 101 would NOT be in alarm, even though the previous two samples were > 2-sigma.

In the picture below, DefaultAltNofMRule property has been set true. Using the regular, default rules, the sample interval at 190 would be considered in alarm because 4 out of 5 samples intervals were greater than 1-sigma. But since the alternative evaluation method for N of M rules is enabled, it is shown as NOT being in alarm, because it is within 1-sigma.



Title: Variable Control Chart (X-Bar & R) Part No.: 283501 Chart No.: 17

Part Name: Transmission Casing Bolt Operation:Threading Spec. Limits: Units: 0.0001 inch

Operator:J. Fenamore Machine: #11 Gage: #8645 Zero Equals: zero

Date: 7/19/2017 12:13:17 PM

TIME	9:43	9:58	10:13	10:28	10:43	10:58	11:13	11:28	11:43	11:58	12:13	12:28
Cpk	-0.486	-0.488	-0.492	-0.490	-0.488	-0.486	-0.486	-0.484	-0.484	-0.484	-0.485	-0.485
ALARM	- -	- -	- -	- -	- -	- -	H -	H -	- -	- -	- -	- -
NOTES	N	N	N	N	N	N	Y	Y	N	N	N	N

Using the Alternative N of M evaluation method, sample interval 190 does not show an alarm for the 4 out of 5 > 1-sigma test, because it is within 1-sigma.

See the NewRev3Features.ResetNofM demo for an example.

Control Limit Alarm Event Handling

The **SPCControlChartData** class can throw an alarm event based on either the current alarm state, or an alarm transition from one alarm state to another. The **SPCControlLimitAlarmArgs** passes alarm data to the event handler. See *Chapter 5 - SPC Control Data and Alarm Classes,* for examples of using an alarm event handler.

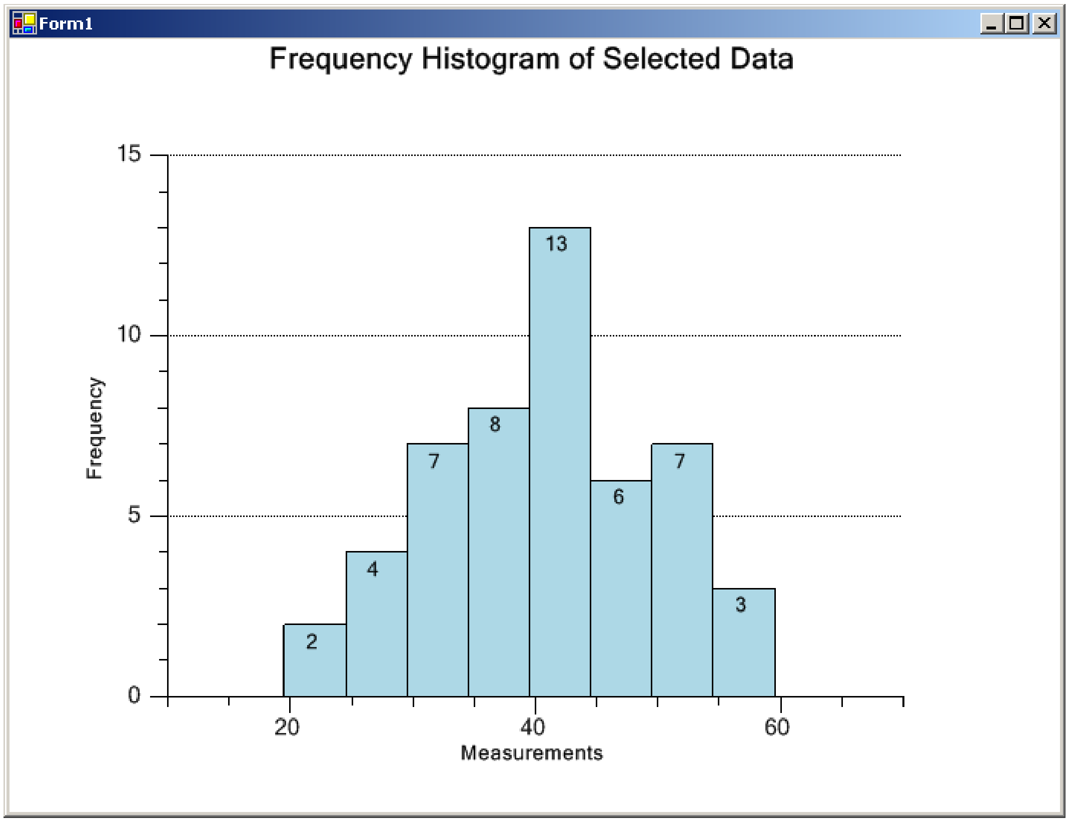
9. Frequency Histogram, Pareto Diagram and Normal-Probability Charts.

FrequencyHistogramChart
ParetoChart
ProbabilityChart

Frequency Histogram Chart

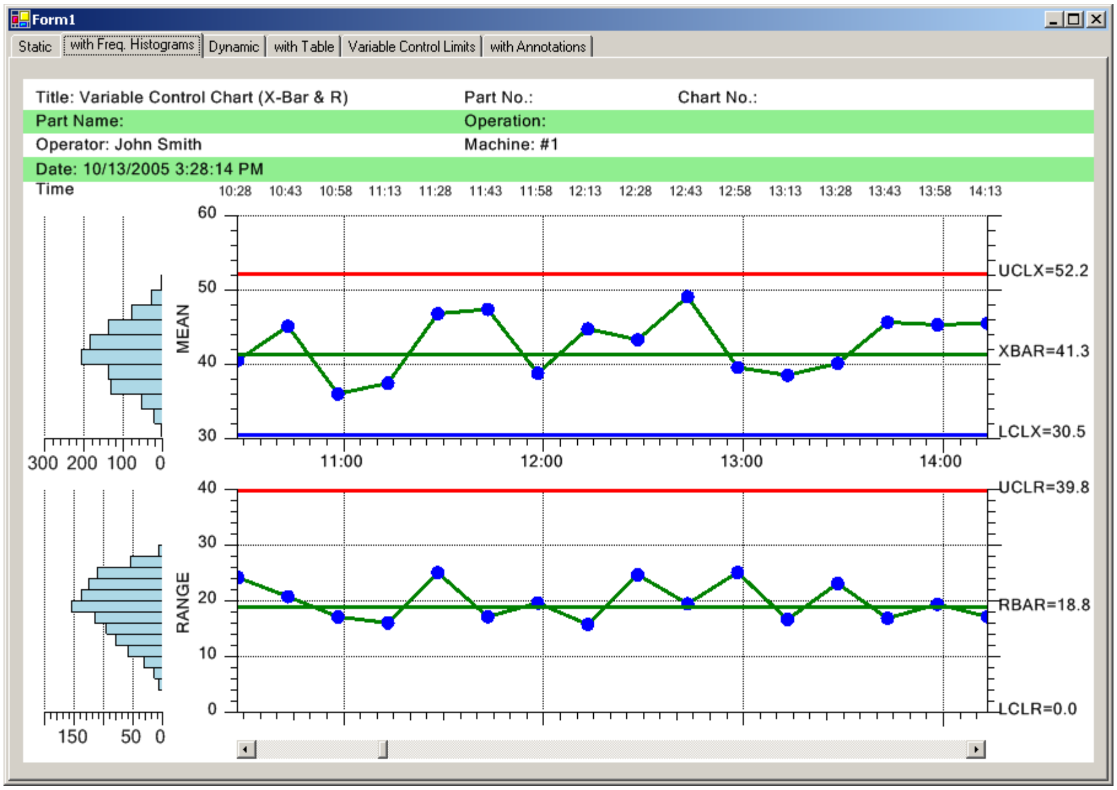
An SPC control chart will allow you to track the trend of critical variables in a production environment. It is important that the production engineer understand whether or not changes or variation in the critical variables are natural variations due to the tolerances inherent to the production machinery, or whether or not the variations are due to some systemic, assignable cause that needs to be addressed. If the changes in critical variables are due to the natural variations, then a frequency histogram of the variations will usually follow one of the common continuous (normal, exponential, gamma, Weibull) or discrete (binomial, Poisson, hypergeometric) distributions. It is the job of the SPC engineer to know what distribution best models his process. Periodically plotting of the variation of critical variables will give SPC engineer important information about the current state of the process. A typical frequency histogram looks like:

Frequency Histogram Chart



Viewing frequency histograms of both the variation in the primary variable (Mean, Median, count, or actual value), and the secondary variable (Range, Sigma or Moving Range) side-by-side with the SPC control chart makes it even easier to find out whether the variations are the result of natural variations or the result of some systemic change in the process.

XBar-R Chart with Integral Frequency Histograms



Title: Variable Control Chart (X-Bar & R)   Part No.:   Chart No.:
Part Name: Operation:
Operator: John Smith   Machine: #1
Date: 10/13/2005 3:28:14 PM

Creating an Independent (not part of a SPC chart) Frequency Histogram

The **FrequencyHistogramChart** class creates a standalone frequency histogram. It is a simple template where you need only supply data and set a few properties to create a proper frequency histogram. The example below extracted from the **FrequencyHistogram.FrequencyHistogramUserControl1** example program.

[C#]

```
public class FrequencyHistogramUserControl1:
    com.quinn-curtis.spcchartnet6.FrequencyHistogramChart
{
    public FrequencyHistogramUserControl1()
    {
        // This call is required by the Windows.Forms Form Designer.
        InitializeComponent();
        // Have the chart fill parent client area
        this.Dock = DockStyle.Fill;
        // Define and draw chart
        InitializeChart();
    }

    void InitializeChart()
    {
        // Frequency bins
    double [] freqLimits = {19.5, 24.5, 29.5, 34.5, 39.5, 44.5, 49.5, 54.5, 59.5};
        // data to be sorted into frequency bins
        double [] freqValues = {32,44,44,42,57,
                             26,51,23,33,27,
                             42,46,43,45,44,
                             53,37,25,38,44,
                             36,40,36,48,56,
                             47,40,58,45,38,
                             32,39,43,31,45,
                             41,37,31,39,33,
                             20,50,33,50,51,
                             28,51,40,52,43};
        // Initialize histogram
        this.InitFrequencyHistogram(freqLimits, freqValues);
        // Set bar orientation
        this.chartOrientation = ChartObj.VERT_DIR;
        // Build chart
        this.BuildChart();
    }
    .
    .
    .
}
```

[VB]

```
Public Class FrequencyHistogramUserControl1
    Inherits com.quinn-curtis.spcchartnet6.FrequencyHistogramChart


#Region " Windows Form Designer generated code "

    Public Sub New()
        MyBase.New()
        'This call is required by the Windows Form Designer.
        InitializeComponent()
        'Add any initialization after the InitializeComponent() call
        Me.Dock = DockStyle.Fill
        InitializeChart()
    End Sub
    .
    .
    .
#End Region



    Sub InitializeChart()
       ' Frequency bins
        Dim freqLimits() As Double = _
                {19.5, 24.5, 29.5, 34.5, 39.5, 44.5, 49.5, 54.5, 59.5}
        'data to be sorted into frequency bins
        Dim freqValues() As Double = _
                {32, 44, 44, 42, 57, 26, 51, 23, 33, _
                27, 42, 46, 43, 45, 44, 53, 37, 25, _
                38, 44, 36, 40, 36, 48, 56, 47, 40, _
                58, 45, 38, 32, 39, 43, 31, 45, 41, _
                37, 31, 39, 33, 20, 50, 33, 50, 51, _
                28, 51, 40, 52, 43}
        Me.InitFrequencyHistogram(freqLimits, freqValues) '
        Me.BuildChart()
    End Sub 'InitializeChart
End Class
```

All you have to do is supply the raw data, and the values of the frequency bins for which you want to accumulate values. The **FrequencyHistogramChart** class auto-scale a coordinate system, creates the proper x- and y-axes, and draws the resulting frequency histogram as a bar plot.

FrequencyHistogramChart.InitFrequencyHistogram Method

Initializes the histogram frequency bin limits, and the data values for the histogram.

[VB]

```
Public Sub InitFrequencyHistogram( _
   ByVal frequencylimits As Double(), _
   ByVal frequencyvalues As Double() _
)
```

[C#]

```
public void InitFrequencyHistogram(
   double[] frequencylimits,
   double[] frequencyvalues
);
```

Parameters

frequencylimits

The frequency limits of the histogram bins.

frequencyvalues

An array the values that are counted with respect to the frequency bins.

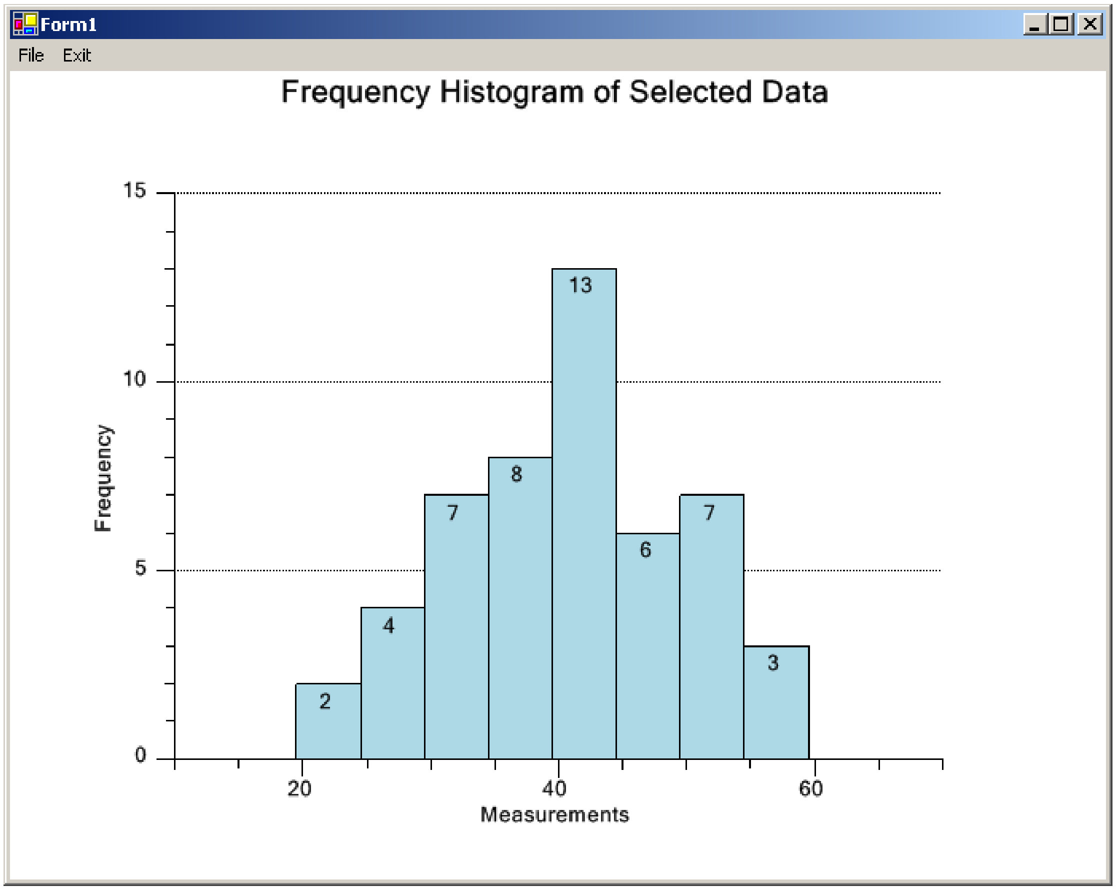
The image below uses the following data:

[C#]

```
// Frequency bins
double [] freqLimits = {100, 150, 200, 250, 300, 350, 400, 450, 500, 550, 600};
// data to be sorted into frequency bins
double [] freqValues =    {121, 349, 345, 322, 277,
                          162, 218, 134, 133, 476,
                          323, 367, 133, 354, 245,
                          434, 476, 352, 185, 144,
                          165, 105, 461, 386, 263,
                          476, 304, 180, 557, 482,
                          327, 293, 539, 318, 251,
                          218, 472, 218, 199, 330,
                          109, 101, 137, 300, 119,
                          380, 410, 206, 122, 238};
```

[VB]

```
' Frequency bins
 Dim freqLimits() As Double = _
    {100, 150, 200, 250, 300, 350, 400, 450, 500, 550, 600}  '
 ' data to be sorted into frequency bins
Dim freqValues() As Double = _
          {121, 349, 345, 322, 277, 162, 218, 134, 133, _
          476, 323, 367, 133, 354, 245, 434, 476, 352, _
          185, 144, 165, 105, 461, 386, 263, 476, 304, _
          180, 557, 482, 327, 293, 539, 318, 251, 218, _
          472, 218, 199, 330, 109, 101, 137, 300, 119, _
          380, 410, 206, 122, 238}
```



Once the Init routine is called, the chart can be further customized using the properties and methods below.

Public Static (Shared) Properties

DefaultAxisLabelsFont Get/Set the default font used for the axes labels and axes titles.

DefaultChartFontString Set/Get the default font used in the chart. This is a string specifying the name of the font.

DefaultDataValueFont Get/Set the default font used for the numeric values labeling the bars.

DefaultFooterFont Get/Set the font used for the chart footer.

DefaultMainTitleFont Get/Set the font used for the main title.

DefaultSubHeadFont Get/Set the font used for the main title.

DefaultToolTipFont Set/Get the default font object used for the tooltip.

Public Instance Constructors

FrequencyHistogramChart	Overloaded. Initializes a new instance of the FrequencyHistogramChart class.

Public Instance Properties

AutoNormalCurve	Set to true and a normal curve with the same area as the histogram is plotted in the chart
BarAttrib	Get the primary bar attribute object for the bars of the histogram.
BarDataValue	Get the numeric label template object used to place numeric values on the bars.
BarFillColor	Sets the fill color for the chart object.
BarLineColor	Sets the line color for the chart object.
BarLineWidth	Sets the line width for the chart object.
ChartOrientation	Get/Set the orientation of the histogram bars in the chart.
CoordinateSystem	Get the coordinate system object for the histogram.
Datatooltip	Get the data tooltip object for the chart.
Footer	Get the footer object for the chart.
FrequencyHistogramPlot	Get the histogram plot object.
FrequencyLimits	Get the DoubleArray object that holds the limit values for the frequency bins of the histogram.
FrequencyValues	Get the DoubleArray object that holds the values that are counted with respect to the frequency bins.
GraphBackground	Get the graph background object.
GraphBorder	Get the default graph border for the chart.
HistogramDataset	Get the GroupDatset object that holds the data used to plot the histogram.
MainTitle	Get the main title object for the chart.
PlotBackground	Get the plot background object.
ResetOnDraw	Set/Get True the ChartView object list is cleared with each redraw
SubHead	Get the subhead title object for the chart.
XAxis	Get the x-axis object.
XAxisLab	Get the x-axis labels object.
XAxisTitle	Get the x-axis title object.
XGrid	Get the x-axis grid object.
YAxis	Get the y-axis object.
YAxisLab	Get the y-axis labels object. Accessible only

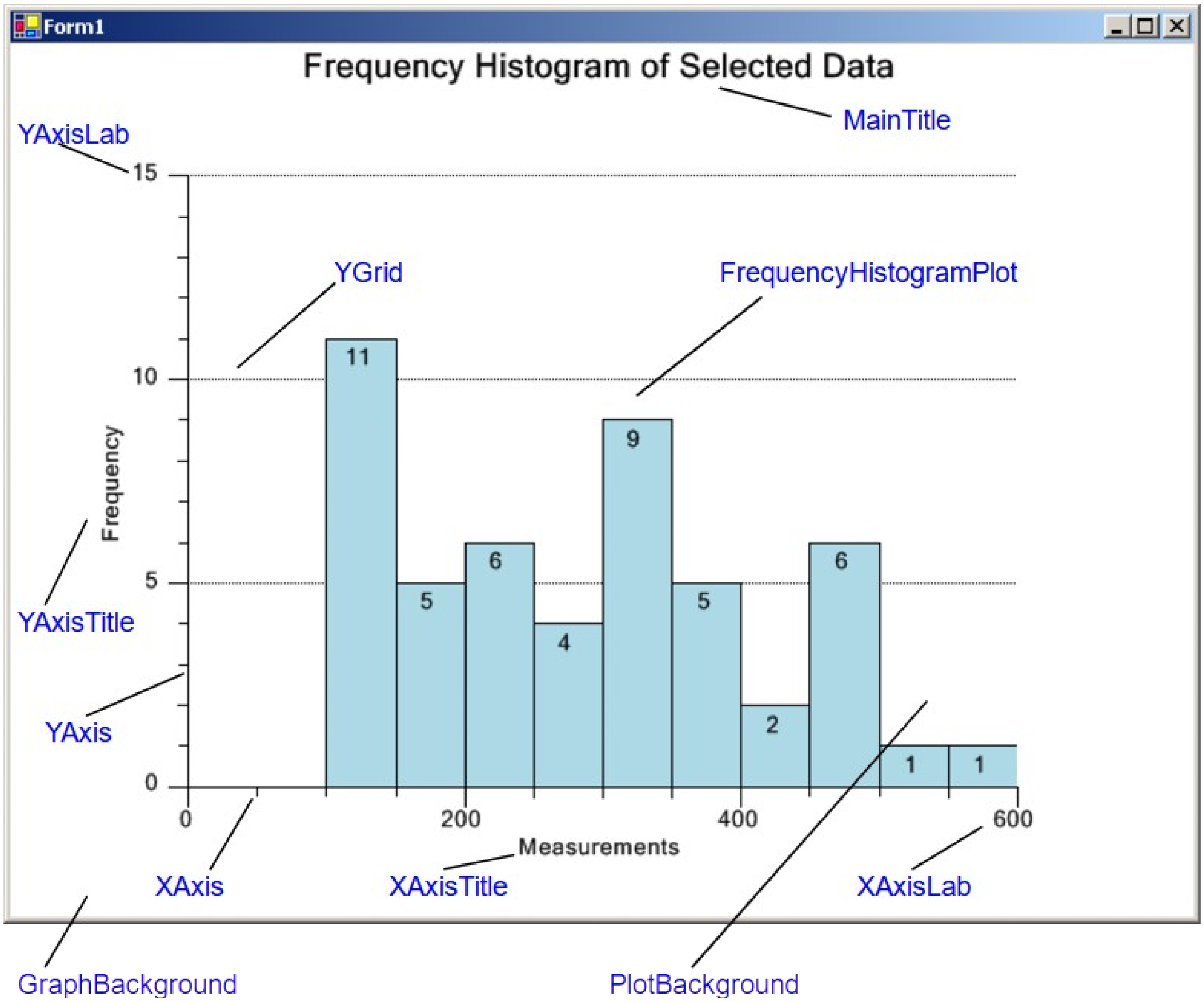
	after BuildGraph
YAxisTitle	Get the y-axis title object.
YGrid	Get the y-axis grid object.

Public Instance Methods

AddFrequencyHistogramControlLine	Add a control limit line to the frequency histogram
Copy	Overloaded. Copies the source FrequencyHistogramChart object.
InitFrequencyHistogram	Initializes the histogram frequency bin limits, and the data values to be analyzed for the histogram.
InitFrequencyHistogramDataset	Builds the histogram dataset, histogramDataset, using the values in frequencyValues and frequencyLimits.

Changing Default Characteristics of the Chart

A **FrequencyHistogramChart** object has one distinct graph with its own set of properties. Once the graph is initialized (using the **InitFrequencyHistogram**, or one of the **FrequencyHistogramChart** constructors**)**, you can modify the default characteristics of each graph using these properties. For example, you can change the color of y-axis, and the y-axis labels using the LineColor property of those objects.



[C#]

```
void InitializeChart()
{
    // Frequency bins
double [] freqLimits = {100, 150, 200, 250, 300, 350, 400, 450, 500, 550, 600};
  // data to be sorted into frequency bins
    double [] freqValues =    {121, 349, 345, 322, 277,
                             162, 218, 134, 133, 476,
                             323, 367, 133, 354, 245,
                             434, 476, 352, 185, 144,
                             165, 105, 461, 386, 263,
                             476, 304, 180, 557, 482,
                             327, 293, 539, 318, 251,
                             218, 472, 218, 199, 330,
                             109, 101, 137, 300, 119,
// Initialize histogram
    this.InitFrequencyHistogram(freqLimits, freqValues);
    this.YAxis.LineColor = Color.Green;
    this.YAxis.LineWidth = 3;
    this.YAxisLab.LineColor = Color.DarkMagenta;
    this.BuildChart();
}
```

[VB]
```
    Sub InitializeChart()
        ' Frequency bins
        Dim freqLimits() As Double = _
          {100, 150, 200, 250, 300, 350, 400, 450, 500, 550, 600}  '
        ' data to be sorted into frequency bins
        Dim freqValues() As Double = _
          {121, 349, 345, 322, 277, 162, 218, 134, 133, _
          476, 323, 367, 133, 354, 245, 434, 476, 352, _
          185, 144, 165, 105, 461, 386, 263, 476, 304, _
          180, 557, 482, 327, 293, 539, 318, 251, 218, _
          472, 218, 199, 330, 109, 101, 137, 300, 119, _
          380, 410, 206, 122, 238}

        Me.InitFrequencyHistogram(freqLimits, freqValues)

        Me.ChartOrientation = ChartObj.VERT_DIR
        Me.BarFillColor = Color.LightCoral
        Me.FrequencyHistogramPlot.SetSegmentFillColor(4, Color.Blue)

        Me.BuildChart()
    End Sub 'InitializeChart
```

Special Considerations

1. The **FrequencyHistogramChart** class uses the **QCChart2D HistogramPlot** plot object class to draw the histogram bars. That class uses individually assignable colors for each bar of the bar plot. The standard **ChartObj.LineColor** and **ChartObj.FillColor** properties do not work to change the color of the histogram bars in this case. Instead, you can change the histogram bar colors by calling **SetSegmentLineColor** and **SetSegmentFillColor**.

[C#]
```
For (int i=0; i < 9; i++)
{
  this.FrequencyHistogramPlot.SetSegmentFillColor(i,Color.Blue);
  this.FrequencyHistogramPlot.SetSegmentLineColor(i,Color.Black);
}
```

[VB]
```
Dim i As Integer
For i = 0 To 8
   Me.FrequencyHistogramPlot.SetSegmentFillColor(i, Color.Blue)
   Me.FrequencyHistogramPlot.SetSegmentLineColor(i, Color.Black)
Next i}
```

You can also use the utility properties, **BarFillColor**, **BarLineColor** and **BarLineWidth**, we added to the **FrequencyHistogramPlot** that will set all of the bars of the histogram plot at once.

[C#]

```
  this.BarFillColor = Color.Blue;
  this.BarLineColor = Color.Black;
```

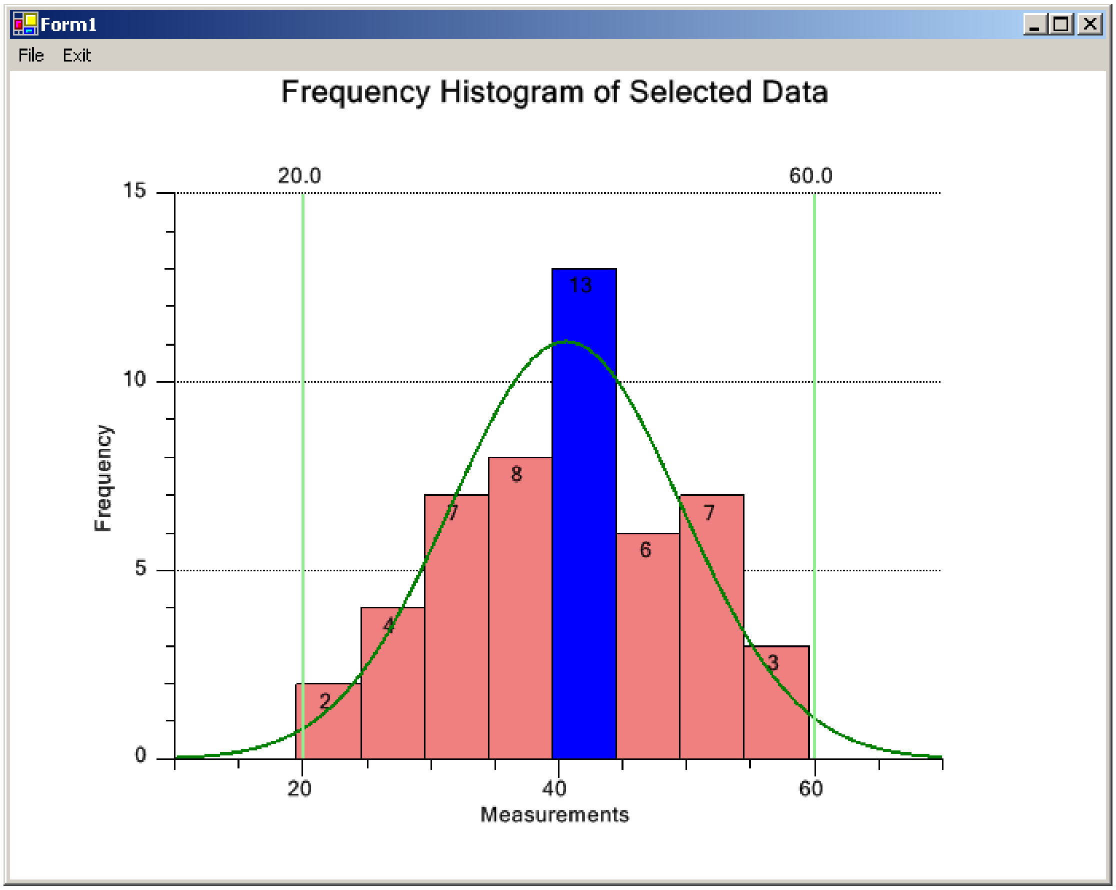
[VB]

```
  Me.BarFillColor = Color.Blue
  Me.BarLineColor = Color.Black
```

Adding Control Lines and Normal Curve to Histogram Plot

Revision 1.7 adds a couple of new features to the histogram plot. First, you can add control limit alarm lines to the histogram plot. The control limit lines will be parallel to the frequency axis. Second, a normal distribution curve can be overlaid on top of the histogram data. The parameters are selected to give the normal distribution curve the same mean, standard deviation and area as the underlying histogram data. If the underlying data is normal, then there should be a relatively close fit between the normal curve and the underlying frequency data.

Histogram Control Limit Lines and Normal Curve fit



[C#]

```
this.AddFrequencyHistogramControlLine(20.0,new ChartAttribute(Color.LightGreen,
2));
this.AddFrequencyHistogramControlLine(60.0,new ChartAttribute(Color.LightGreen,
2));
this.AutoNormalCurve = true;
```

[VB]

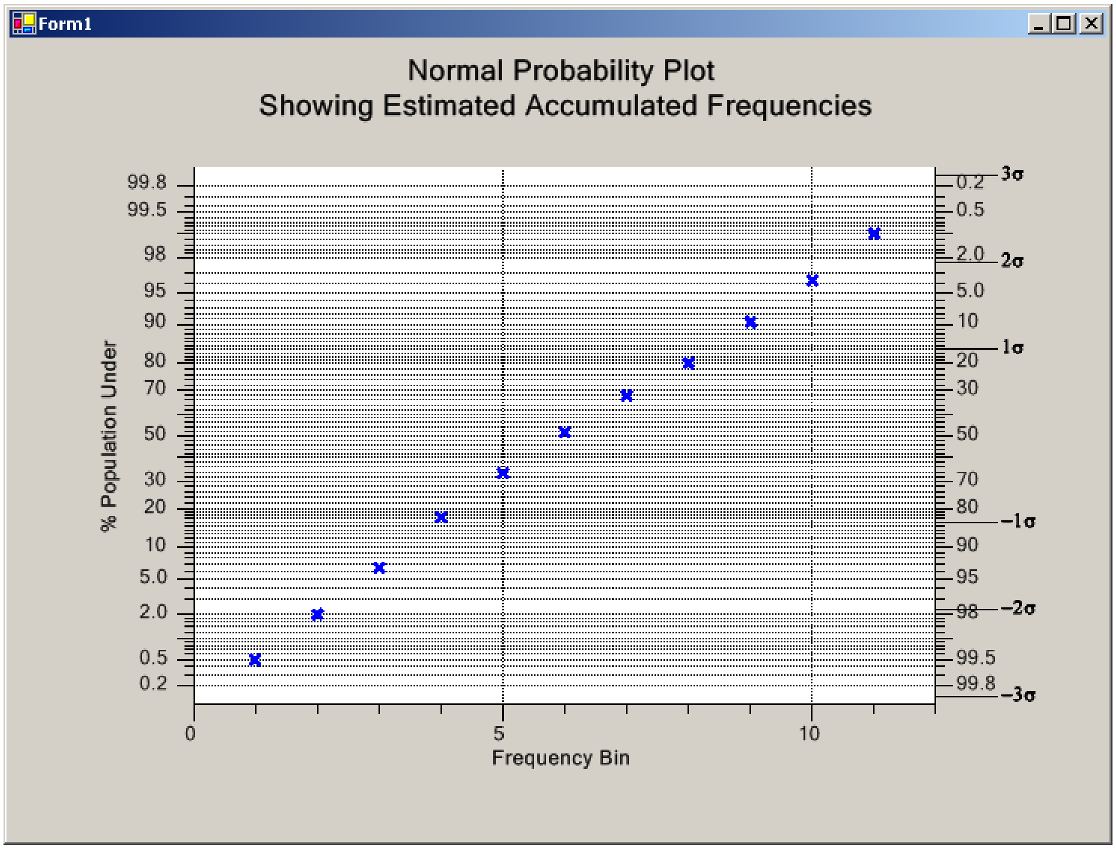
```
Me.AddFrequencyHistogramControlLine(20.0, new ChartAttribute(Color.LightGreen, 2))
Me.AddFrequencyHistogramControlLine(60.0, new ChartAttribute(Color.LightGreen, 2))
Me.AutoNormalCurve = True
```

Probability Plots

Another important tool the SPC engineer uses to model the process variation is the probability plot. The probability plot is a simple way to test whether control chart measurements fit a normal distribution. Usually probability plot graphs are plotted by hand using special probability plot graph paper. We have added probability scale and axis classes that enable you to plot probability plots directly on the computer. Control chart measurements that follow a normal distribution curve will plot as a straight line when plotted in a normal probability plot.

Creating a Probability Plot

Cumulative Normal Probability Chart



The **ProbabilityChart** class creates a standalone probability plot. It is a simple template where you need only supply data and set a few properties to create a proper frequency histogram. The example below is extracted from the **ProbabilityPlot.ProbabilityPlotUserControl1** example program.

[C#]

```
public class ProbabilityPlotUserControl1:
    com.quinn-curtis.spcchartnet6.ProbabilityChart
{
    public ProbabilityPlotUserControl1()
    {
        // This call is required by the Windows.Forms Form Designer.
        InitializeComponent();
        // Have the chart fill parent client area
        this.Dock = DockStyle.Fill;
        // Define and draw chart
        InitializeChart();
    }


    void InitializeChart()
    {
        // TODO: Add any initialization after the InitForm call
        double []x1 = {1,2,3,4,5,6,7,8,9};
        double []y1 = {2, 6, 13, 26, 58, 82, 93, 97,100};
        this.InitProbabilityChart(x1, y1);
        this.BuildChart();
    }
    .
    .
    .
}
```

[VB]

```
Public Class ProbabilityPlotUserControl1
    Inherits com.quinn-curtis.spcchartnet6.ProbabilityChart

#Region " Windows Form Designer generated code "

    Public Sub New()
        MyBase.New()

        'This call is required by the Windows Form Designer.
        InitializeComponent()
        'Add any initialization after the InitializeComponent() call
         Me.Dock = DockStyle.Fill
        InitializeChart()
    End Sub
    .
    .
    .
#End Region

    Sub InitializeChart()
        ' TODO: Add any initialization after the InitForm call
        Dim x1() As Double = {1.0, 2, 3, 4, 5, 6, 7, 8, 9}
        Dim y1() As Double = {2, 6, 13, 26, 58, 82, 93, 97, 100}

        Me.InitProbabilityChart(x1, y1)
        Me.BuildChart() '
    End Sub 'InitializeChart '
    .
    .
    .
End Class
```

All you have to do is supply the raw data, and the values of the frequency bins for which you want to accumulate values. The **ProbabilityChart** class auto-scale a coordinate system, creates the proper x- and y-axes, and draws the resulting probability plot as a scatter plot.

ProbabilityChart.InitProbabilityChart Method

Initializes the x- and y-values of the data points plotted in the probability plot.

[VB]

```
Public Sub InitProbabilityChart( _
   ByVal xvalues As Double(), _
   ByVal yvalues As Double() _
)
```

[C#]

```
public void InitProbabilityChart(
   double[] xvalues,
   double[] yvalues
);
```

Parameters

xvalues

The x-values of the data points plotted in the probability plot.

yvalues

The y-values of the data points plotted in the probability plot.

Public Static (Shared) Properties

DefaultAxisLabelsFont	Set/Get default font object used for the axes labels.
DefaultAxisTitleFont	Set/Get the default font object used for the axes titles. Set this properties BuildGraph.
DefaultChartFontString	Set/Get the default font used in the chart. This is a string specifying the name of the font.
DefaultFooterFont	Set/Get the default font object used for the chart footer.
DefaultMainTitleFont	Set/Get the default font object used for the main chart title.
DefaultSigmaFont	Set/Get the default font object used for the axis labels sigma character.
DefaultSubHeadFont	Set/Get the default font object used for the subhead title.
DefaultToolTipFont	Set/Get the default font object used for the

tooltip.

Public Instance Constructors

ProbabilityChart	Overloaded. Initializes a new instance of the ProbabilityChart class.

Public Instance Properties

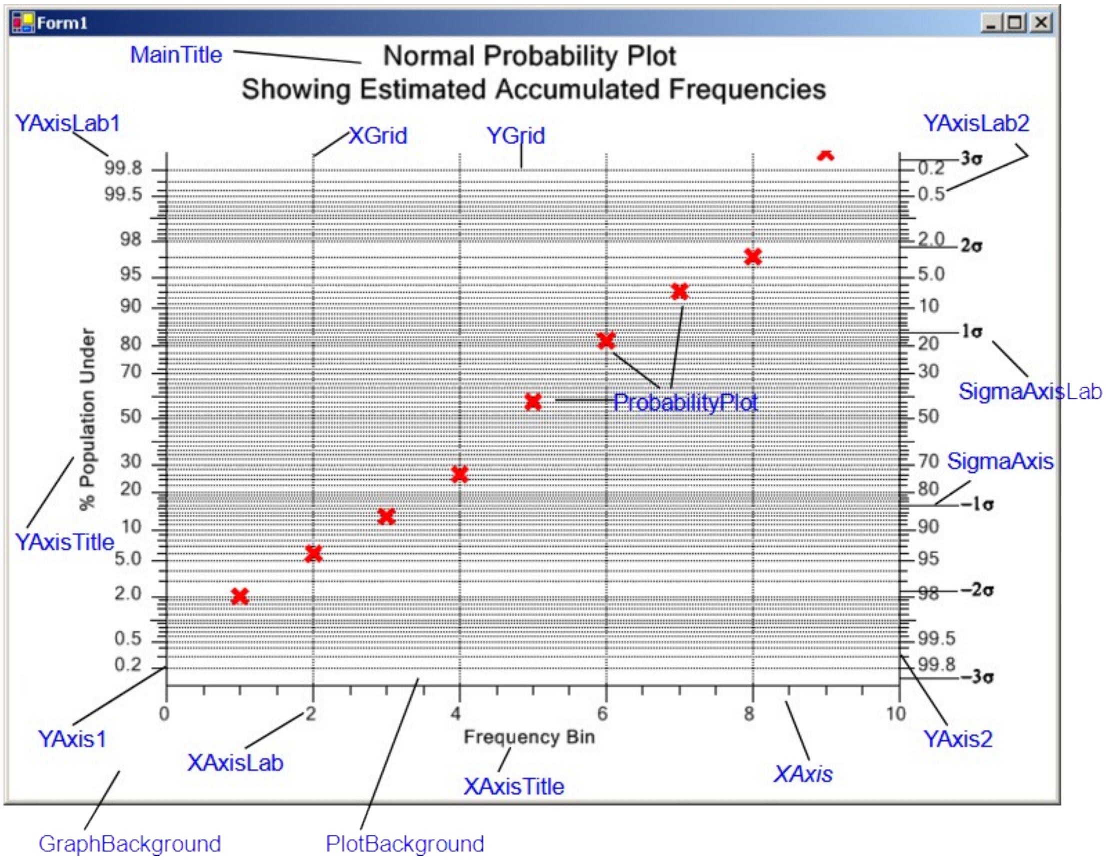
CoordinateSystem	Get the probability coordinate system for the chart.
Datatooltip	Get the chart data tooltip.
DefaultGraphBorder	Get/Set the default graph border object for the chart.
Footer	Get the chart footer object.
GraphBackground	Get the graph background object.
MainTitle	Get the chart title object.
PlotAttrib	Get the default primary plot attribute object for the plot of the chart. Set attributes before BuildChart.
PlotBackground	Get the plot background object.
ProbabilityDataset	Get the dataset holding the data values of the plot.
ProbabilityPlot	Get probability plot scatter plot object.
ResetOnDraw	Set/Get True the ChartView object list is cleared with each redraw
SigmaAxis	Get the sigma y-axis object of the chart.
SigmaAxisLab	Get the sigma y-axis labels object of the chart.
SubHead	Get the chart subhead object.
SymbolSize	Get/Set the default symbol size. Set attributes before BuildChart.
TextTemplate	Get the default text object template for the data tooltip.
ToolTipSymbol	Get the tooltip symbol object for the data tooltip.
XAxis	Get the x-axis object of the chart.
XAxisLab	Get the x-axis labels object of the chart.
XAxisTitle	Get the x-axis title object of the of the chart.
XGrid	Get the x-axis grid object of the of the chart.
XValues	Get the DoubleArray of the x-values of the data points plotted in the probability plot.
XValueTemplate	Get the default x-value object template for the data tooltip.
YAxis1	Get the left probability y-axis object of the

	chart.
YAxis2	Get the right probability y-axis object of the chart.
YAxisLab1	Get the left probability y-axis labels object of the chart.
YAxisLab2	Get the right probability y-axis labels object of the chart.
YAxisTitle	Get the y-axis title object of the of the chart.
YGrid	Get the y-axis title object of the of the chart.
YValues	Get the DoubleArray of the y-values of the data points plotted in the probability plot.
YValueTemplate	Get the default y-value object template for the data tooltip.

Public Instance Methods

BuildChart	Overloaded. Builds the probability chart using the base objects ChartView.
Copy	Overloaded. Copies the source ProbabilityChart object.
InitProbabilityChart	Initializes the x- and y-values of the data points plotted in the probability plot.
InitProbabilityDatasets	Builds the histogram dataset, histogramDataset, using the values in frequencyValues and frequencyLimits.

Changing Default Characteristics of the Chart



Once the graph is initialized (using the **InitProbabilityChart** , or one of the **ProbabilityChart** constructors**)**, you can modify the default characteristics of each graph using these properties. For example, you can change the color of y-axis, and the y-axis labels using the **LineColor** property of those objects.

[C#]

```
void InitializeChart()
{
    // TODO: Add any initialization after the InitForm call
    double []x1 = {1,2,3,4,5,6,7,8,9};
    double []y1 = {2, 6, 13, 26, 58, 82, 93, 97,100};
    this.InitProbabilityChart(x1, y1);

    this.ProbabilityPlot.LineColor = Color.Red;
    this.ProbabilityPlot.ChartObjAttributes.SymbolSize = 12;
    this.YAxis1.LineColor = Color.Green;
    this.YAxis1.LineWidth = 3;
    this.YAxis2.LineColor = Color.Blue;
    this.YAxis2.LineWidth = 3;
    this.YAxisLab1.LineColor = Color.DarkMagenta;
    this.BuildChart();
}
```

[VB]

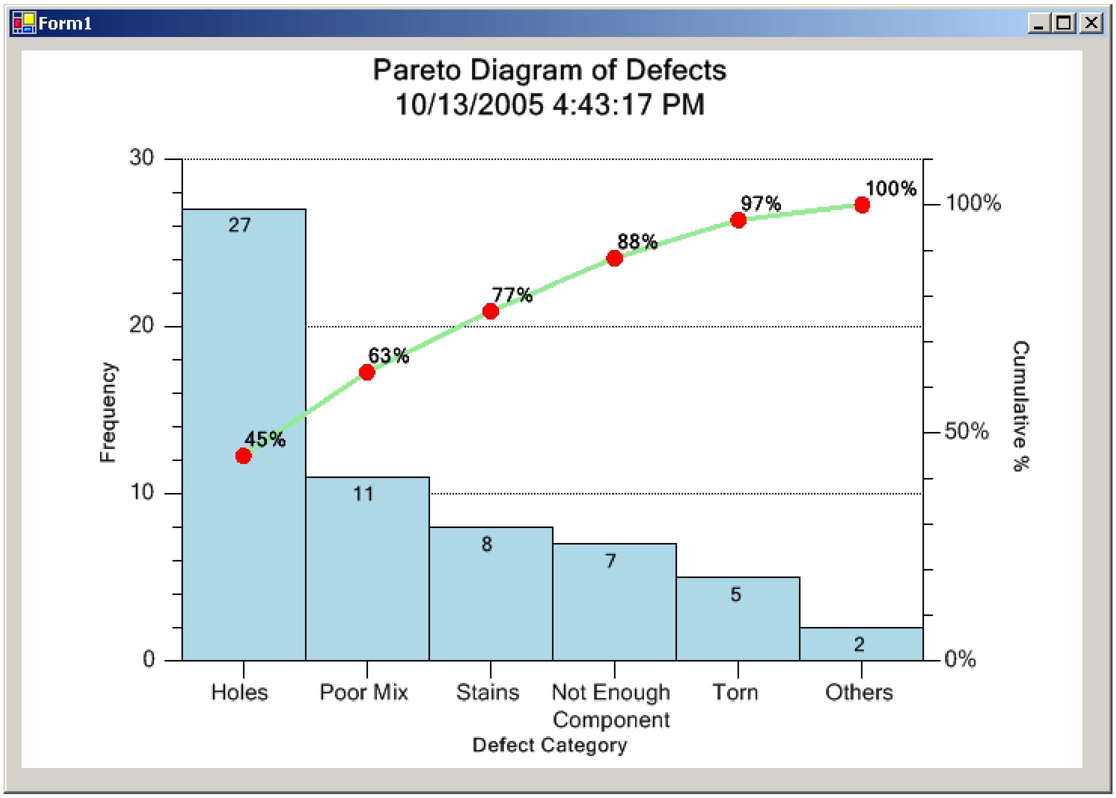
```
    Sub InitializeChart()
```

```
        ' TODO: Add any initialization after the InitForm call
        Dim x1() As Double = {1.0, 2, 3, 4, 5, 6, 7, 8, 9}
        Dim y1() As Double = {2, 6, 13, 26, 58, 82, 93, 97, 100}

        Me.InitProbabilityChart(x1, y1)

        Me.ProbabilityPlot.SetColor(Color.Red) '

        ' sets both line and fill color
        Me.ProbabilityPlot.ChartObjAttributes.SymbolSize = 12
        Me.YAxis1.LineColor = Color.Green
        Me.YAxis1.LineWidth = 3
        Me.ProbabilityPlot.LineColor = Color.Red
        Me.ProbabilityPlot.ChartObjAttributes.SymbolSize = 12

        Me.YAxis2.LineColor = Color.Blue
        Me.YAxis2.LineWidth = 3
        Me.YAxisLab1.LineColor = Color.DarkMagenta

        Me.BuildChart() '

    End Sub 'InitializeChart '
```

Pareto Diagrams

The Pareto diagram is special type of bar graph combined with a line plot, plotted in a chart that uses two different y-axis scales. The bar graph scale is a frequency scale that measures the number of times a specific problem has occurred. The line plot scale is a cumulative percentage scale.

Pareto Chart



The chart is easy to interpret. The tallest bar, the left-most one in a Pareto diagram, is the problem that has the most frequent occurrence. The shortest bar, the right-most one, is the problem that has the least frequent occurrence. Time spend on fixing the biggest problem will have the greatest affect on the overall problem rate. This is a simplistic view of actual Pareto analysis, which would usually take into account the cost effectiveness of fixing a specific problem. Never less, it is powerful communication tool that the SPC engineer can use in trying to identify and solve production problems.

Creating a Pareto Diagram

The **ParetoChart** class creates a standalone Pareto Diagram chart. It is a simple template where you need only supply data and set a few properties to create a proper frequency histogram. The example below is from the ParetoPlotUserControl1 file of the ParetoDiagram example program.

[C#]

```
public class ParetoPlotUserControl1 : com.quinn-curtis.spcchartnet6.ParetoChart
{
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.Container components = null;

        public ParetoPlotUserControl1()
        {
            // This call is required by the Windows.Forms Form Designer.
            InitializeComponent();

            // Have the chart fill parent client area
            this.Dock = DockStyle.Fill;
            // Define and draw chart
            InitializeChart();
        }

        void InitializeChart()
        {
            // add Pareto chart categories, values and strings
            this.AddCategoryItem(5, "Torn");
            this.AddCategoryItem(7, "Not Enough\nComponent");
            this.AddCategoryItem(2, "Others");
            this.AddCategoryItem(11, "Poor Mix");
            this.AddCategoryItem(27, "Holes");
            this.AddCategoryItem(8, "Stains");
            // Build chart
            this.BuildChart();

        }
    .
    .
    .
}
```

[VB]

```
Public Class ParetoPlotUserControl1
    Inherits com.quinn-curtis.spcchartnet6.ParetoChart
#Region " Windows Form Designer generated code "

    Public Sub New()
        MyBase.New()
        'This call is required by the Windows Form Designer.
        InitializeComponent()
        'Add any initialization after the InitializeComponent() call
        'Add any initialization after the InitializeComponent() call
        Me.Dock = DockStyle.Fill
        InitializeChart()
    End Sub

    .
    .
    .
#End Region

    Sub InitializeChart()
        ' add Pareto chart categories, values and strings
        Me.AddCategoryItem(5, "Torn")
        Me.AddCategoryItem(7, "Not Enough" + ControlChars.Lf + "Component")
        Me.AddCategoryItem(2, "Others")
        Me.AddCategoryItem(11, "Poor Mix")
        Me.AddCategoryItem(27, "Holes")
        Me.AddCategoryItem(8, "Stains")
        ' Build chart
        Me.BuildChart()
    End Sub 'InitializeChart

End Class
```

All you have to do is supply the raw data, and the values of the frequency bins for which you want to accumulate values. The **ParetoChart** class auto-scale a coordinate system, creates the proper x- and y-axes, and draws the resulting probability plot as a scatter plot.

ParetoChart.InitParetoChart Method

Initializes the x- and y-values of the data points plotted in the probability plot.

[VB]

```
Public Sub InitParetoChart( _
   ByVal categoryitems As Double(), _
   ByVal stringitems As String() _
)
```

[C#]

```
public void InitParetoChart (
   double[] categoryitems,
   string[] stringitems
);
```

Parameters

categoryitems

The values for each category in the Pareto chart.

stringitems

The strings identifying each category in the Pareto chart.

You can add the category item values and string item strings one by one using the **AddCategoryItem** method.

ParetoChart.AddCategoryItem Method

Add an item to the categoryValues and categoryStrings arrays.

[VB]

```
Public Sub AddCategoryItem( _
   ByVal itemfreq As Double, _
   ByVal itemstring As String _
)
```

[C#]

```
public void AddCategoryItem(
   double itemfreq,
   string itemstring
);
```

Parameters

itemfreq

The count of how many times this category has occurred.

itemstring

The string identifying the category item.

Public Static (Shared) Properties

DefaultAxisLabelsFont
Set/Get default font object used for the axes labels and axes titles. Set attributes before BuildChart.

DefaultChartFontString
Set/Get the default font used in the chart. This is a string specifying the name of the font.

DefaultFooterFont
Set/Get the default footer font. Set attributes before BuildChart.

DefaultMainTitleFont
Get/Set the default chart title font. Set attributes before BuildChart.

DefaultSubHeadFont
Get/Set the default chart title font. Set attributes before BuildChart.

DefaultToolTipFont
Set/Get the default font object used for the tooltip.

Public Instance Constructors

ParetoChart
Overloaded. Initializes a new instance of the ParetoChart class.

Public Instance Properties

BarAttrib
Get the default primary bar attribute object for the bars of the chart. Set attributes before

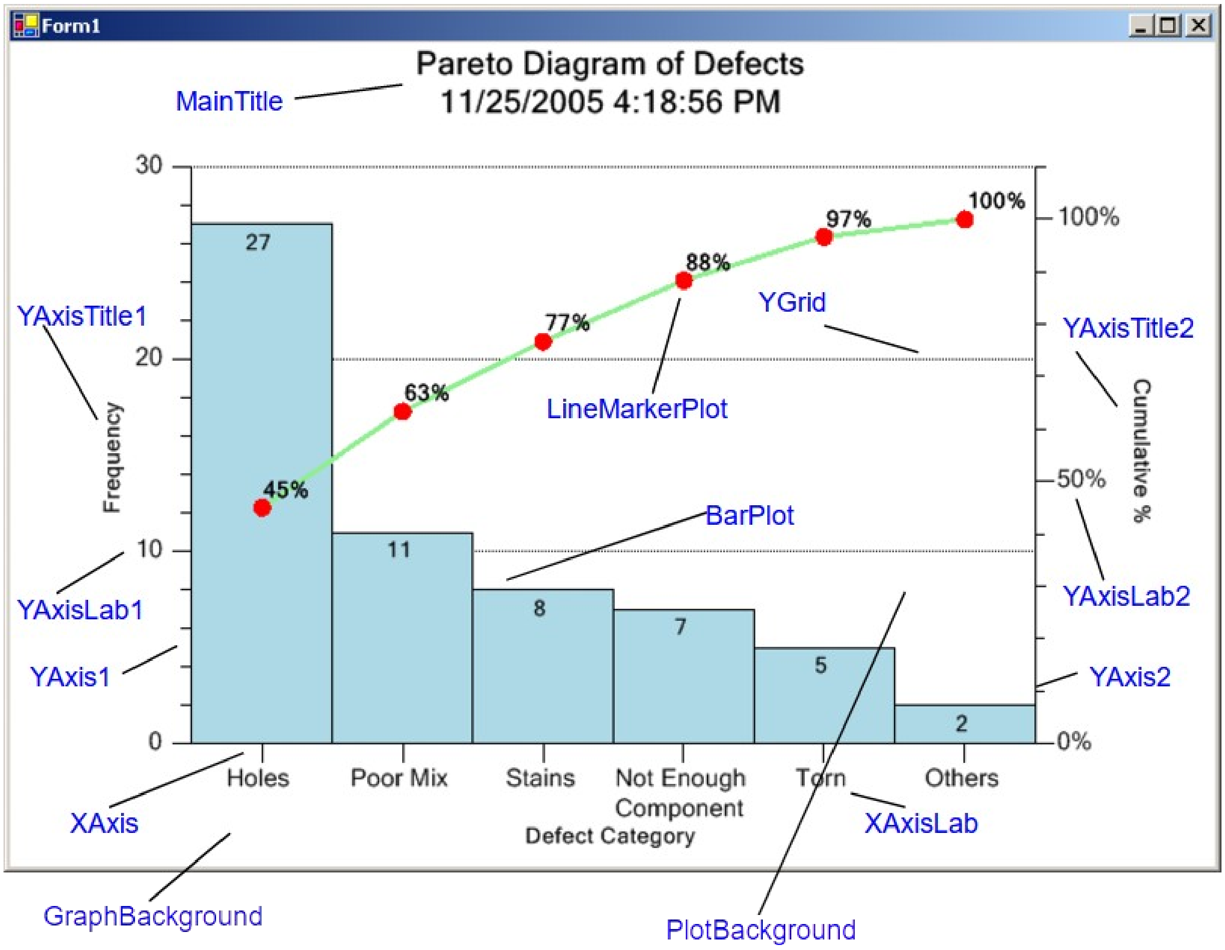
	BuildChart.
BarDataValue	Get the default numeric label template used to label the values of bar plot of the frequency histogram part of the chart. Set attributes before BuildChart.
BarPlot	Get the histogram bar plot object of the frequency histogram part of the chart.
BarWidth	Get the default width value of the frequency histogram bars. Set attributes before BuildChart.
CategoryStrings	Get the StringArray object holding the strings used to label the categories of the Pareto plot. Set attributes before BuildChart.
CategoryValues	Get the DoubleArray object holding the category values used in building the Pareto plot. Set attributes before BuildChart.
CoordinateSystem1	Get the coordinate system object of the frequency histogram part of the chart.
CoordinateSystem2	Get the coordinate system object of the cumulative frequency part of the chart.
CumulativeFreqDataset	Get the dataset object used to hold the cumulative frequency values of the data plot.
Datatooltip	Get the data tooltip object for the chart.
DefaultGraphBorder	Get/Set the default graph border object for the chart. Set attributes before BuildChart.
Footer	Get the footer of the chart.
FrequencyDataset	Get the dataset object used to hold the frequency histogram values of the bar plot.
GraphBackground	Get the graph background object.
LineAttrib	Get the default primary line attribute object for the line plot of the chart. Set attributes before BuildChart.
LineMarkerPlot	Get the line marker plot object displaying the cumulative frequency part of the chart.
LineMarkerPlotDataValue	Get the default numeric template object used to label the line marker plot of the cumulative frequency part of the chart. Set attributes before BuildChart.
MainTitle	Get main title object of the chart.
PlotBackground	Get the plot background object.
ResetOnDraw	Set/Get True the ChartView object list is cleared with each redraw
Scale2StartY	Set/Get the starting y-value for the

	cumulative frequency scale.
Scale2StopY	Get/Set the ending y-value for the cumulative frequency scale.
SymbolAttrib	Get the default symbolAttrib attribute object for the symbols of the chart. Set attributes before BuildChart.
XAxis	Get the x-axis of the chart object.
XAxisLab	Get the x-axis string labels object of the chart.
XAxisTitle	Get the x-axis title object of the of the chart.
YAxis1	Get the y-axis object of the frequency histogram part of the chart.
YAxis2	Get the y-axis object of the cumulative frequency part of the chart.
YAxisLab1	Get the y-axis labels object of the frequency histogram part of the chart.
YAxisLab2	Get the y-axis numeric labels object of the cumulative frequency part of the chart.
YAxisTitle1	Get the y-axis title object of the frequency histogram part of the chart.
YAxisTitle2	Get the y-axis title object of the cumulative frequency part of the chart.
YGrid	Get the y-axis grid object of the chart.

Public Instance Methods

AddCategoryItem	Add an item to the categoryValues and categoryStrings arrays.
BuildChart	Overloaded. Builds the Pareto chart using the base objects ChartView.
Copy	Overloaded. Copies the source ParetoChart object.
InitParetoChart	Initializes the category values, and the category strings for the Pareto chart.
InitParetoChartsDatasets	Builds the histogram dataset, histogramDataset, using the values in frequencyValues and frequencyLimits.
SortCategoryItems	Sort the category values and category strings.

Changing Default Characteristics of the Chart



Once the graph is initialized (using the **InitParetoChart** , or one of the **ParetoChart** constructors**)**, you can modify the default characteristics of each graph using these properties. For example, you can change the color of y-axis, and the y-axis labels using the **LineColor** property of those objects.

[C#]

```
void InitializeChart()
{
    this.AddCategoryItem(5, "Torn");
    this.AddCategoryItem(7, "Not Enough\nComponent");
    this.AddCategoryItem(2, "Others");
    this.AddCategoryItem(11, "Poor Mix");
    this.AddCategoryItem(27, "Holes");
    this.AddCategoryItem(8, "Stains");
    this.LineMarkerPlot.LineAttributes.PrimaryColor = Color.Blue;
    this.LineMarkerPlot.SymbolAttributes.PrimaryColor = Color.Black;
    this.YAxis1.LineColor = Color.Green;
    this.YAxis1.LineWidth = 3;
    this.YAxis2.LineColor = Color.Blue;
    this.YAxis2.LineWidth = 3;
    this.YAxisLab1.LineColor = Color.DarkMagenta;

    this.BuildChart();
}
```

[VB]

```
    Sub InitializeChart()
        ' add Pareto chart categories, values and strings
        Me.AddCategoryItem(5, "Torn")
        Me.AddCategoryItem(7, "Not Enough" + ControlChars.Lf + "Component")
        Me.AddCategoryItem(2, "Others")
        Me.AddCategoryItem(11, "Poor Mix")
        Me.AddCategoryItem(27, "Holes")
        Me.AddCategoryItem(8, "Stains")
        ' Build chart
        Me.LineMarkerPlot.LineAttributes.PrimaryColor = Color.Blue
        Me.LineMarkerPlot.SymbolAttributes.PrimaryColor = Color.Black
        Me.YAxis1.LineColor = Color.Green
        Me.YAxis1.LineWidth = 3
        Me.YAxis2.LineColor = Color.Blue
        Me.YAxis2.LineWidth = 3
        Me.YAxisLab1.LineColor = Color.DarkMagenta

        Me.BuildChart()
    End Sub 'InitializeChart
```

10. File and Printer Rendering Classes

ChartPrint
BufferedImage

All of the chart classes described in this manual:

com.quinn-curtis.chart2dnet6.ChartView
- FrequencyHistogramChart
- ParetoChart
- ProbabilityChart
- SPCChartBase
  - SPCBatchAttributeControlChart
  - SPCBatchVariableControlChart
  - SPCTimeAttributeControlChart
  - SPCTimeVariableControlChart

are derived from the **com.quinn-curtis.chart2dnet6.ChartView** class. The chart and table information displayed in these charts can be printed and saved to image files using the techniques described in the **QCChart2D** manual, QCChart2DNetManual.pdf. This chapter repeats that information, substituting examples extracted from the for **SPC Control Chart Tools for .Net** examples.

High quality B&W and color printing is an important feature of the charting library. The resulting graph renders on the printer using the resolution of the output device, for both text and graphical elements of the chart, and does not transfer a grainy image from the computer to the printer. The **QCChart2D for .***Net* software uses the Microsoft .Net **PrintDocument** component to implement printing. Since the aspect ratio of the printed page is different from the aspect ratio of common displays, options are included that allow different modes for positioning and sizing the chart on the printed page.

The **BufferedImage** class converts a chart into a .Net **Bitmap** object, or saves the chart to a file in any of the graphics formats supported by the **System.Drawing.Imaging.ImageFormat** class. The image file is placeable in a web page or an application program.

Printing a Chart

Class ChartPrint

ChartObj
|
+--ChartPrint

The **ChartPrint** class uses the Microsoft .Net **PrintDocument** component to implement printing. The class selects, setups, and outputs a chart to a printer.

ChartPrint constructor

```
[Visual Basic]
Overloads Public Sub New( _
   ByVal component As ChartView, _
   ByVal nsizemode As Integer _
)
[C#]
public ChartPrint(
   ChartView component,
   int nsizemode
);
```

component Specifies the **ChartView** object to be printed.

nsizemode Specifies the printer mapping mode. Use one of the mapping mode constants:

PRT_MAX Print the view so that paper is used maximally. Text prints proportional to other objects, aspect ratio is NOT maintained

PRT_EXACT Print the view at the same size as the screen, at least as far as .Net maintains a one to one correspondence in the printing engine. The aspect ration of the view is maintained.

PRT_RECT Print the view to the specified rectangle, specified using the SetPrintRect method and normalized coordinates.

Call the **ChartPrint.DoPrintDialog** method after creating the **ChartPrint** object. Then call the **ChartPrint.DoPrintPage** method, rendering the chart to the printer. If the **DoPrintDialog** method is not called prior to **DoPrintPage**, the **DoPrintPage** method automatically invokes the **DoPrintDialog** method. Subsequent calls to **DoPrintPage** will not invoke the **DoPrintDialog** method.

ChartPrint example **(extracted from the example program FrequencyHistogram.Form1)**

```
namespace FrequencyHistogram
{
    /// <summary>
    /// Summary description for Form1.
    /// </summary>
    public class Form1 : System.Windows.Forms.Form
    {
        private FrequencyHistogramUserControl1 frequencyHistogramUserControl1;
        .
        .
        .
        ChartPrint printobj = null;

        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.Container components = null;

        public Form1()
        {
            //
            // Required for Windows Form Designer support
            //
            InitializeComponent();

            //
            // TODO: Add any constructor code after InitializeComponent call
            //
            // Create histogram user control
            frequencyHistogramUserControl1 = new FrequencyHistogramUserControl1();
            // Add control to form
            this.Controls.Add(frequencyHistogramUserControl1);
        }

        private void menuItem2_Click(object sender, System.EventArgs e)
        {
            if (frequencyHistogramUserControl1 != null)
                this.PageSetup(frequencyHistogramUserControl1, sender, e);
        }

        private void menuItem3_Click(object sender, System.EventArgs e)
        {
            if (frequencyHistogramUserControl1 != null)
                this.PrinterSetup(frequencyHistogramUserControl1, sender, e);

        }

        private void menuItem4_Click(object sender, System.EventArgs e)
        {
            if (frequencyHistogramUserControl1 != null)
                this.PrintPreview(frequencyHistogramUserControl1, sender, e);

        }

        private void menuItem5_Click(object sender, System.EventArgs e)
        {
            if (frequencyHistogramUserControl1 != null)
                this.PrintPage(frequencyHistogramUserControl1, sender, e);
        }


        private void menuItem8_Click(object sender, System.EventArgs e)
        {
            Application.Exit();
        }


        // This routine invokes the chart objects PageSetupItem method
    public void PageSetup(ChartView charview, object sender, System.EventArgs e)
        {
            if (charview != null)
            {
                if (printobj == null)
                {
                    printobj = new ChartPrint(charview);
                }
                else
                    printobj.PrintChartView = charview;
                printobj.PageSetupItem(sender, e);
            }
        }

        // This routine invokes the chart objects printer setup dialog method
public void PrinterSetup(ChartView charview, object sender, System.EventArgs e)
        {
            if (charview != null)
            {
                if (printobj == null)
                {
                    printobj = new ChartPrint(charview);
                }
                else
                    printobj.PrintChartView = charview;
                printobj.DoPrintDialog();
            }
        }

        // This routine invokes the chart objects PrintPreviewItem method
        public void PrintPreview(ChartView charview, object sender,
System.EventArgs e)
        {
            if (charview != null)
            {
                if (printobj == null)
                {
                    printobj = new ChartPrint(charview);
                }
                else
                    printobj.PrintChartView = charview;
                printobj.PrintPreviewItem(sender, e);
            }
        }

// This routine prints a chart by invoking the chart objects DocPrintPage method
    public void PrintPage(ChartView charview, object sender, System.EventArgs e)
        {
            if (charview != null)
            {
                if (printobj == null)
                {
                    printobj = new ChartPrint(charview);
                    printobj.DoPrintDialog();
                }
                else
                    printobj.PrintChartView = charview;

                printobj.DocPrintPage( sender,  e);
            }
        }
}
```

[Visual Basic]

```
Imports com.quinn-curtis.chart2dnet6
Imports com.quinn-curtis.spcchartnet6
Imports System.IO
Imports System.Text
Imports System.Drawing.Imaging

Public Class Form1
    Inherits System.Windows.Forms.Form

    Dim histogramplot As FrequencyHistogramUserControl1
    Dim printobj As ChartPrint

#Region " Windows Form Designer generated code "

    Public Sub New()
        MyBase.New()

        'This call is required by the Windows Form Designer.
        InitializeComponent()

        'Add any initialization after the InitializeComponent() call
        histogramplot = New FrequencyHistogramUserControl1()
        Me.Controls.Add(histogramplot)
    End Sub
    .
    .
    .

    Private Sub menuItem2_Click(ByVal sender As Object, _
       ByVal e As System.EventArgs) Handles MenuItem2.Click
        If Not (histogramplot Is Nothing) Then
            Me.PageSetup(histogramplot, sender, e)
        End If
    End Sub 'menuItem2_Click

    Private Sub menuItem3_Click(ByVal sender As Object, _
       ByVal e As System.EventArgs) Handles MenuItem3.Click
        If Not (histogramplot Is Nothing) Then
            Me.PrinterSetup(histogramplot, sender, e)
        End If
    End Sub 'menuItem3_Click

    Private Sub menuItem4_Click(ByVal sender As Object, _
       ByVal e As System.EventArgs) Handles MenuItem4.Click
        If Not (histogramplot Is Nothing) Then
            Me.PrintPreview(histogramplot, sender, e)
        End If
    End Sub 'menuItem4_Click

    Private Sub menuItem5_Click(ByVal sender As Object, _
       ByVal e As System.EventArgs) Handles MenuItem5.Click
        If Not (histogramplot Is Nothing) Then
            Me.PrintPage(histogramplot, sender, e)
        End If
    End Sub 'menuItem5_Click

    Private Sub menuItem8_Click(ByVal sender As Object, _
       ByVal e As System.EventArgs) Handles MenuItem8.Click
        Application.Exit()
    End Sub 'menuItem9_Click


    ' This routine invokes the chart objects PageSetupItem method
    Public Sub PageSetup(ByVal charview As ChartView, _
       ByVal sender As Object, ByVal e As System.EventArgs)
        If Not (charview Is Nothing) Then
            If printobj Is Nothing Then
                printobj = New ChartPrint(charview)
            Else
                printobj.PrintChartView = charview
            End If
            printobj.PageSetupItem(sender, e)
        End If
    End Sub 'PageSetup


    ' This routine invokes the chart objects printer setup dialog method
    Public Sub PrinterSetup(ByVal charview As ChartView, _
       ByVal sender As Object, ByVal e As System.EventArgs)
        If Not (charview Is Nothing) Then
            If printobj Is Nothing Then
                printobj = New ChartPrint(charview)
            Else
                printobj.PrintChartView = charview
            End If
            printobj.DoPrintDialog()
        End If
    End Sub 'PrinterSetup


    ' This routine invokes the chart objects PrintPreviewItem method
    Public Sub PrintPreview(ByVal charview As ChartView, _
       ByVal sender As Object, ByVal e As System.EventArgs)
        If Not (charview Is Nothing) Then
            If printobj Is Nothing Then
                printobj = New ChartPrint(charview)
            Else
                printobj.PrintChartView = charview
            End If
            printobj.PrintPreviewItem(sender, e)
        End If
    End Sub 'PrintPreview


    ' This routine prints a chart by invoking the chart objects DocPrintPage
method
    Public Sub PrintPage(ByVal charview As ChartView, _
       ByVal sender As Object, ByVal e As System.EventArgs)
        If Not (charview Is Nothing) Then
            If printobj Is Nothing Then
                printobj = New ChartPrint(charview)
                printobj.DoPrintDialog()
            Else
                printobj.PrintChartView = charview
            End If
            printobj.DocPrintPage(sender, e)
        End If
    End Sub 'PrintPage
End Class
```

The example TimeVariableControlCharts demonstrates how to one chart among many displayed in a Tab control.

Capturing the Chart as a Buffered Image

Class BufferedImage

ChartObj
|
+-- BufferedImage

The **BufferedImage** class creates a **Bitmap** object that renders a **ChartView** object into an image buffer. The rendering takes place when the **BufferedImage**.Render method or **BufferedImage.SaveImage** method is called.

BufferedImage constructor

```
[VB]
Overloads Public Sub New( _
   ByVal component As ChartView, _
   ByVal imgformat As ImageFormat _
)

[Visual Basic]
Overloads Public Sub New( _
   ByVal component As ChartView _
)

[C#]
public BufferedImage(
   ChartView component,
   ImageFormat imgformat
);


public BufferedImage(
   ChartView component
);
```

component The **ChartView** object that is the source for the chart image.

imageformat An image format object specifying the format of the rendered image.

The **BufferedImage.GetBufferedImage** method converts the chart to the .Net **Bitmap** object specified by the imageformat object and returns a reference the resulting bitmap.

BufferedImage example (extracted from the example program FrequencyHistogram.Form1)

[C#]

```
// This routine displays a dialog box, in response to an event, that prompts
// the user for the name and
// file type of the image to be saved.
//  The file type is derived from the file extension.
// The chart represented by "this" object is saved to the file
// using the specified format.
public void SaveAsFile(ChartView chartview, object sender, System.EventArgs e)
{
    String filename = this.Name;
    SaveFileDialog imagefilechooser = new SaveFileDialog();
    imagefilechooser.Filter =
"Image Files(*.BMP;*.JPG;*.GIF;*.TIFF;*.PNG)|*.BMP;*.JPG;*.GIF;*.TIFF;*.PNG|All
files (*.*)|*.*";
    imagefilechooser.FileName = filename;
    if (imagefilechooser.ShowDialog() == DialogResult.OK)
    {
        filename = imagefilechooser.FileName;
        FileInfo fileinformation = new FileInfo(filename);
        String fileext = fileinformation.Extension;
        fileext = fileext.ToUpper();
        ImageFormat fileimageformat;
        if (fileext == ".BMP" )
            fileimageformat = ImageFormat.Bmp;
        else if ((fileext == ".JPG") || (fileext == ".JPEG"))
            fileimageformat = ImageFormat.Jpeg;
        else if ((fileext == ".GIF"))
            fileimageformat = ImageFormat.Gif;
        else if ((fileext == ".TIF") || (fileext == ".TIFF"))
            fileimageformat = ImageFormat.Tiff;
        else if ((fileext == ".PNG"))
            fileimageformat = ImageFormat.Png;
        else
            fileimageformat = ImageFormat.Bmp;

        BufferedImage savegraph = new BufferedImage(chartview, fileimageformat);
        savegraph.Render();
        savegraph.SaveImage(filename);
    }
}
```

[VB]

```
' This routine displays a dialog box, in response to an event, that prompts
' the user for the name and
' file type of the image to be saved. The file type is derived from the file
`  extension.
' The chart represented by "this" object is saved to the file using the specified
` format.

Public Sub SaveAsFile(ByVal chartview As ChartView, _
    ByVal sender As Object, ByVal e As System.EventArgs)
    Dim filename As [String] = Me.Name
    Dim imagefilechooser As New SaveFileDialog()
    imagefilechooser.Filter = _
"Image Files(*.BMP;*.JPG;*.GIF;*.TIFF;*.PNG)|*.BMP;*.JPG;*.GIF;*.TIFF;*.PNG|All
files (*.*)|*.*"
     imagefilechooser.FileName = filename
     If imagefilechooser.ShowDialog() = DialogResult.OK Then
         filename = imagefilechooser.FileName
         Dim fileinformation As New FileInfo(filename)
         Dim fileext As [String] = fileinformation.Extension
         fileext = fileext.ToUpper()
         Dim fileimageformat As ImageFormat
         If fileext = ".BMP" Then
             fileimageformat = ImageFormat.Bmp
         Else
             If fileext = ".JPG" Or fileext = ".JPEG" Then
                 fileimageformat = ImageFormat.Jpeg
             Else
                 If fileext = ".GIF" Then
                     fileimageformat = ImageFormat.Gif
                 Else
                     If fileext = ".TIF" Or fileext = ".TIFF" Then
                         fileimageformat = ImageFormat.Tiff
                     Else
                         If fileext = ".PNG" Then
                             fileimageformat = ImageFormat.Png
                         Else
                            fileimageformat = ImageFormat.Bmp
                         End If
                     End If
                End If
```

```
            End If
        End If
        Dim savegraph As New BufferedImage(chartview, fileimageformat)
        savegraph.Render()
        savegraph.SaveImage(filename)
    End If
End Sub 'SaveAsFile
```

11. Regionalization for non-USA English Markets

SPCChartStrings

Starting with Revision 3.0, we provide a much improved structure for adjusting the software for different cultures and languages. All of the pre-defined strings in the software have been moved to the static class SPCChartStrings, which can be modified at run-time, or, if you have the QCSPCChartNet6.dll source code, at compile time. You can create multiple sets of strings, one for each unique region you sell to, and initialize the software to that set at run-time. While something similar is often done using resource files in the final .Net application program, it was not possible to add a user-customizable resource file to a pre-compiled library, like our QCSPCChartNet6 dll.

Apart from the strings, there are a couple other areas which benefit from regionalization. First is the use of the “,” (comma) in some locales as the decimal separator, in place of the “.” (period). Our software uses the standard numeric conversion routines found in .Net, for converting numeric values to strings, and these automatically take into account the proper format for the region recognized by the computer. So, you shouldn't have to do anything there.

Also, date/time values are subject to regional differences; specifically the order of the month-day-year in short form strings of the form 10/1/2011 (“M/dd/yyyy” US English format), compared to 1/10/2011 ((“dd/M/yyyy” European format). There a couple of date/time formats in the list below, located at the enumerated values **defaultDateFormat** and **defaultTimeStampFormat.** You should change those to whatever time format you want. The format specification is that used by the .Net DateTime.ToString method.

The **SPChartStrings** class looks like:

```
public static class SPCChartStrings
{
       public  enum   SPCStringEnum: int
        {
            empty=0,
            chartFont,
            highAlarmStatus,
            lowAlarmStatus,
            shortStringNo,
            shortStringYes,
             .
```

```
    .
    .
        };

        // US-en
        static String[] usEnglishStrings = {
            "",
            "Microsoft Sans Serif",
            "H",
            "L",
            "N",
            "Y",
               .
               .
               .

      };

        static ArrayList globalStringList = new ArrayList();

        public static String[] currentDefaultStrings = usEnglishStrings;
        .
        .
        .

}
```

The **SPCStringEnum** enumeration contains an item for every default string used in the software. Paired with that is the currentDefaultStringsList, which has one string element for every enumerated value in the **SPCStringEnum** enumeration. The software pulls the strings it uses from the currentDefaultStrings array. It is initialize by default with the usEnglishStrings array; the string values of which are listed in the tables below.

SPCStringEnum	**Default Strings**	**Description**
start=0	"start"	Marks the start of the enumeration
chartFont	"Microsoft Sans Serif"	default font string
highAlarmStatus	"H"	alarm status line - High short string
lowAlarmStatus	"L"	alarm status line - Low short string
shortStringNo	"N"	No short string
shortStringYes	"Y"	Yes short string
dataLogUserString	""	default data log user string

sPCControlChartDataTitle	"Variable Control Chart (X-Bar & R)"	Default chart title
zeroEqualsZero	"zero"	table zero string
timeValueRowHeader	"TIME"	TIME row header
alarmStatusValueRowHeader	"ALARM"	ALARM row header
numberSamplesValueRowHeader	"NO. INSP."	NO. INSP. row header
titleHeader	"Title: "	Title field caption
partNumberHeader	"Part No.: "	Part number field caption
chartNumberHeader	"Chart No.: "	Chart number field caption
partNameHeader	"Part Name: "	Part name field caption
operationHeader	"Operation:"	Operation field caption
operatorHeader	"Operator:"	Operator field caption
machineHeader	"Machine: "	Machine field caption
dateHeader	"Date: "	Date field caption
specificationLimitsHeader	"Spec. Limits: "	Spec limits field caption
gageHeader	"Gage: "	Chart number field caption
unitOfMeasureHeader	"Units: "	Chart number field caption
zeroEqualsHeader	"Zero Equals: "	Chart number field caption
defaultMean	"MEAN"	MEAN Calculated value row header
defaultMedian	"MEDIAN"	MEDIAN Calculated value row header
defaultRange	"RANGE"	RANGE Calculated value row header
defaultVariance	"VARIANCE"	VARIANCE Calculated value row header

defaultSigma	"SIGMA"	SIGMA Calculated value row header
defaultSum	"SUM"	SUM Calculated value row header
defaultSampleValue	"SAMPLE VALUE"	SAMPLE VALUE alculated value row header
defaultAbsRange	"ABS(RANGE)"	ABS(RANGE) Calculated value row header
defaultMovingAverage	"“MA”	
defaultCusumCPlus	""C+"	
defaultCusumCMinus	""C-"	
defaultEWMA	""EWMA"	
defaultPercentDefective	" "% DEF."	
defaultFractionDefective	""FRACT. DEF."	
defaultNumberDefective	""NO. DEF."	
defaultNumberDefects	""NO. DEF."	
defaultNumberDefectsPerUnit	""NO. DEF./UNIT"	
defaultNumberDefectsPerMillion	""DEF./MILLION"	
defaultPBar	""PBAR"	
defaultAttributeLCL	""LCLP"	
defaultAttributeUCL	""UCLP"	
defaultAbsMovingRange	"MR"	Moving Range Calculated value row header
defaultAbsMovingSigma	"MS"	Moving Sigam Calculated value row header
defaultX	"X"	Default string used to label centerline value of I-R chart.
defaultXBar	"XBAR"	Default string used to label centerline value for

		XBar chart
defaultRBar	"RBAR"	Default string used to label centerline value for Range chart
defaultTarget	"Target"	Default string used for target
defaultLowControlLimit	"LCL"	Default string used to label low control limit line
defaultLowAlarmMessage	"Low Alarm"	Default string used for low alarm limit message
defaultUpperControlLimit	"UCL"	Default string used to label high control limit line
defaultHighAlarmMessage	"High Alarm"	Default string used for high alarm limit message
defaultSampleRowHeaderPrefix	"Sample #"	Row header for Sample # rows
defaultDefectRowHeaderPrefix	"Defect #"	Row header for Defect # rows
batchColumnHead	"Batch #"	Default string used as the batch number column head in the log file.
timeStampColumn	"Time Stamp"	Default string used as the time stamp column head in the log file.
sampleValueColumn	"Sample #"	Default string used as the sample value column head in the log file.
notesColumn	"Notes"	Default string used as the notes value column head in the log file.
defaultDateFormat	"M/dd/yyyy"	Default date format used by the software.
defaultTimeStampFormat	"M/dd/yyyy HH:mm:ss"	Default full date/time format used by the software.
defaultDataLogFilenameRoot	"SPCDataLog"	Root string used for auto-naming of log data file.
dataLogFilename	"SPCDataLog"	Datalog default file name
frequencyHistogramXAxisTitle	"Measurements"	Frequency Histogram default x-axis title.
frequencyHistogramYAxisTitle	"Frequency"	Frequency Histogram default y-axis title.

frequencyHistogramMainTitle	"Frequency Histogram"	Frequency Histogram default main title.
paretoChartXAxisTitle	"Defect Category"	Pareto chart x-axis title
paretoChartYAxis1Title	"Frequency"	Pareto chart left y-axis title
paretoChartYAxis2Title	"Cumulative %"	Pareto chart right y-axis title
paretoChartMainTitle	"Pareto Diagram"	Pareto chart main title
probabilityChartXAxisTitle	"Frequency Bin"	Probability chart x-axis title
probabilityChartYAxisTitle	"% Population Under"	Probability chart y-axis title
probabilityChartMainTitle	"Normal Probability Plot"	Probability chart main title
basic	"Basic"	Basic rules string
weco	"WECO"	WECO rules string
wecowsupp	"WECO+SUPPLEMENTAL"	WECO and Supplemental Rules string
nelson	"Nelson"	Nelson rules string
aiag	"AIAG"	AIAG rules string
juran	"Juran"	Juran rules string
hughes	"Hughes"	Hughes rules string
gitlow	"Gitlow"	Gitlow rules string
duncan	"Duncan"	Duncan rules string
westgard	"Westgard"	Westgard rules string
primarychart	"Primary chart"	Used in alarm messages to specify the Primary Chart variable chart is in alarm
secondarychart	"Secondary chart"	Used in alarm messages to specify the Secondary

		Chart variable chart is in alarm
greaterthan	"greater than"	Used in alarm messages to specify that a greater than alarm limit has been violated
lessthan	"less than"	Used in alarm messages to specify that a less than alarm limit has been violated
above	"above"	Used in alarm messages to specify that values above a limit
below	"below"	Used in alarm messages to specify that values below a limit
increasing	"increasing"	Used in alarm messages to specify that values are increasing
decreasing	"decreasing"	Used in alarm messages to specify that values are decreasing
trending	"trending"	Used in alarm messages to specify that values are trending
within	"within"	Used in alarm messages to specify that values are within certain limits
outside	"outside"	Used in alarm messages to specify that values are outside certain limits
alternating	"alternating"	Used in alarm messages to specify that values are alternating about a limit value
centerline	"center line"	Used in alarm messages to specify the center line of the chart
r2s	"R2s"	Used in alarm messages to specify the R2s test
sigmaShort	"S"	alarm status line - sigma short string
beyondAlarmStatus	"B"	alarm status line - beyond short string
trendingAlarmStatus	"T"	alarm status line - trending short string
stratificationAlarmStatus	"S"	alarm status line - stratification short string
oscillationAlarmStatus	"O"	alarm status line - oscillation short string

r2sAlarmStatus	"R"	alarm status line - R2s short string
rule	"Rule"	used in alarm messages for word "Rule"
violation	"violation"	used in alarm messages for word "violation"
sigma	"sigma"	used in alarm messages for word "sigma"
target	"Target"	used in alarm messages for word "Target"
ucl	"UCL"	used in alarm messages for to designate Upper Control Limit
lcl	"LCL"	used in alarm messages for to designate Lower Control Limit
defaultCp	"Cp"	used to label Cp row of table
defaultCpl	"Cpl"	used to label Cpl row of table
defaultCpu	"Cpu"	used to label Cpu row of table
defaultCpk	"Cpk"	used to label Cpk row of table
defaultCpm	"Cpm"	used to label Cpm row of table
defaultPp	"PPp"	used to label Pp row of table
defaultPl	"Ppl"	used to label Pl row of table
defaultPu	"Ppu"	used to label Pu row of table
defaultPpk	"Ppk"	used to label Ppk row of table
end	"end"	Marks the end of the enumeration

The enumerated values can be used to access and modify any specific string.

```
String oldstring = SPCChartStrings.SetString(
     SPCChartStrings.SPCStringEnum.defaultMean,"AVERAGE");

oldstring = SPCChartStrings.SetString(
     SPCChartStrings.SPCStringEnum.defaultDateFormat,“dd/M/yyyy”);
```

Where the static **SetString** method returns the previous value for the string.

The **SPCChartStrings** module is used to define default, and static strings, for the various **QCSPCChart** classes, when those classes are initialized. Trying to set the **SPCChartStrings** strings using **SetString** after any of the SPC charts have been instantiated will not have the desired effect. Since the SPC charts classes are normally instantiated in the main Window.xaml file, you must change any strings before that intialization takes place. The best way to do that is to initialize the string in a static method in the main Form file. You will find an example in the TimeVariableControlCharts.Form1.cs file. Call the static method using an initialization of a static variable in the global variables section of the class. This will guarantee that the strings get initialized first. Since the SPCChartStrings class is static, you can call it anytime. It does not need instantiation.

```
 public partial class Window1 : Window
    {
        DynamicXBarRChart dxbrc;
        .
        .
        .

        static bool initStringsComplete = InitStrings();

        public Window1()
        {


            InitializeComponent();

            InitializeCharts();
        }

        static bool InitStrings()
        {
           String oldstring =
SPCChartStrings.SetString(  SPCChartStrings.SPCStringEnum.defaultMean,"MEAN");
            // Euro standard
           // oldstring = SPCChartStrings.SetString(
           //      SPCChartStrings.SPCStringEnum.defaultDateFormat, "dd/M/yyyy");
            // us-Eng standard
            oldstring = SPCChartStrings.SetString(
                   SPCChartStrings.SPCStringEnum.defaultDateFormat, "M/dd/yyyy");
               return true;
           }

.
.
}
```

You can change every string used in the software, line by line, using the technique above. Another way is to create a properly sized and initialized string array and change every string in the software with a single call to the **SPCChartStrings.SetCurrentDefaultStrings** method. You will find an example in the MiscBatchBasedControlCharts.Form1 example.

[C#]

```
 static bool initStringsFlag = InitStrings();

 static bool InitStrings()
 {
         // US-en
    String[] usEnglishStrings = {
    "start", // used to mark the beginning of the array
    "Microsoft Sans Serif",  // default font string
    "H",          // alarm status line - High short string
    "L",          // alarm status line - Low short string
    "N",          // No short string
     "Y",          // Yes short string
     "",            // default data log user string
     "Variable Control Chart (X-Bar & R)", // Default chart title
     "zero",  // table zero string
     "TIME",  // TIME row header
     "ALARM",  // ALARM row header
     "NO. INSP.", // NO. INSP. row header
     "Title: ",   // Title field caption
     "Part No.: ", // Part number field caption
     "Chart No.: ", // Chart number field caption
    .
    .
    .

    "end"  // used to mark the end of the array
          };

         SPCChartStrings.SetCurrentDefaultStrings ( usEnglishStrings);
         return true;
        }
```

[VB]

```
  Shared initStringsFlag As Boolean = InitStrings()

  Private Shared Function InitStrings() As Boolean
    ' US-en

    Dim usEnglishStrings As [String]() = {"start", "Microsoft Sans Serif", "H", "L", "N", "Y", _
```

```
        "", "Variable Control Chart (X-Bar & R)", "zero", "TIME", "ALARM", "NO. INSP.", _
        "Title: ", "Part No.: ", "Chart No.: ", "Part Name: ", "Operation:", "Operator:", _
        "Machine: ", "Date: ", "Spec. Limits: ", "Gage: ", "Units: ", "Zero Equals: ", _
        "MEAN", "MEDIAN", "RANGE", "VARIANCE", "SIGMA", "SUM", _
        "SAMPLE VALUE", "ABS(RANGE)", "MA", "C+", "C-", "EWMA", _
        "% DEF.", "FRACT. DEF.", "NO. DEF.", "NO. DEF.", "NO. DEF./UNIT", "DEF./MILLION", _
        "PBAR", "LCLP", "UCLP", "MR", "MS", "X", _
        "XBAR", "RBAR", "Target", "LCL", "Low Alarm", "UCL", _
        "High Alarm", "Sample #", "Defect #", "Batch #", "Time Stamp", "Sample #", _
        "Notes", "M/dd/yyyy", "M/dd/yyyy HH:mm:ss", "SPCDataLog", "SPCDataLog", "Measurements", _
        "Frequency", "Frequency Histogram", "Defect Category", "Frequency", "Cumulative %", "Pareto
Diagram", _
        "Frequency Bin", "% Population Under", "Normal Probability Plot", "Basic", "WECO",
"WECO+SUPPLEMENTAL", _
        "Nelson", "AIAG", "Juran", "Hughes", "Gitlow", "Duncan", "Westgard", _
        "Primary chart", "Secondary chart", "greater than", "less than", "above", "below", _
        "increasing", "decreasing", "trending", "within", "outside", "alternating", _
        "center line", "R2s", "S", "B", "T", "S", "O", "R", _
        "Rule", "violation", "sigma", "Target", "UCL", "LCL", _
        "Cp", "Cpl", "Cpu", "Cpk", "Cpm", "Pp", _
        "Pl", "Pu", "Ppk", "end"}

        SPCChartStrings.SetCurrentDefaultStrings(usEnglishStrings)
        Return True
    End Function
```

12. Using SPC Control Chart Tools for .Net to Create Windows Applications

(*** Critical Note ***) Running the Example Programs

The example programs for **SPC Control Chart Tools for .Net** software are supplied in complete source. In order to save space, they have not been pre-compiled which means that many of the intermediate object files needed to view the main form are not present. This means that **ChartView** derived control will not be visible on the main Form if you attempt to view the main form before the project has been compiled. The default state for all of the example projects should be the Start Page. Before you do view any other file or form, do a build of the project. This will cause the intermediate files to be built. If you attempt to view the main Form before building the project, Visual Studio sometimes decides that the **ChartView** control placed on the main form does not exist and deletes it from the project.

The primary view class of the QCSPCChart library is the ChartView class. The ChartView class derives from the .Net System.Windows.Forms.UserControl class. It has the properties and methods of the underlying UserControl class.

Follow the following steps in order to incorporate the QCSPCChart classes into your program. This is not the only way to add charts to an application. In general, any technique that works with UserControl derived classes will work. We found the technique described below to be the most flexible.

.Net Framework 6.0

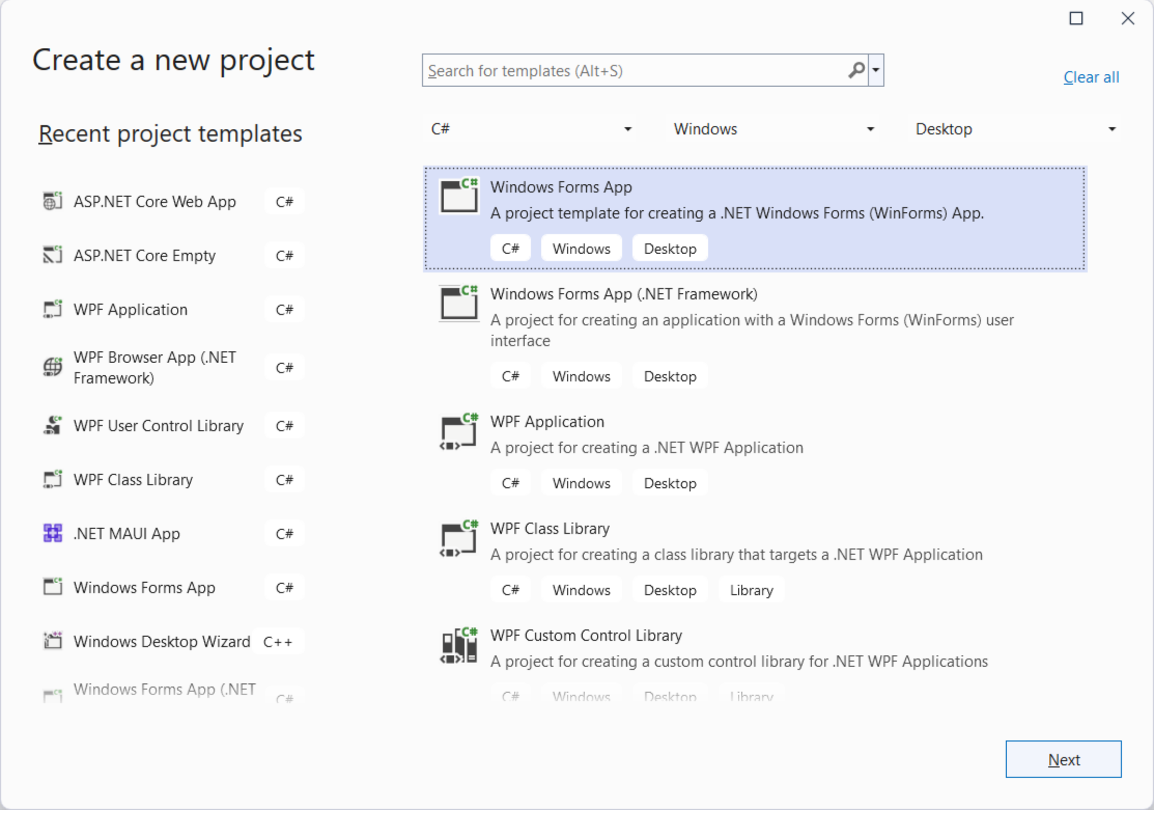
Starting with Rev. 3.1, the QCChart2DNet6 and QCSPCChartNet6 DLLs have been recompiled to target the .Net 6.0 Framework. Also, all of the example program projects have been converted to target the .Net 6 Framework.

Visual Studio 2022

All of the projects in this software were recreated for .Net 6 using Visual Studio 2022. Since Visual Studio 2022 Community Edition can be downloaded for free from Microsoft, we assume that everyone is using this version or greater. You can create projects from scratch which utilize this software using earlier versions of Visual Studio (2019, 2017), it just that projects specifically created using Visual Studio 2022 may not be backward compatible with earlier versions of Visual Studio.

Visual C# for .*Net*

If you do not already have an application program project, create one using the Visual Studio project wizard (File | New | Project. Select C# | Windows | Desktop across the top. | Visual C# Projects | Windows Application). Select the project type Windows Forms App.



Create a new project

Search for templates (Alt+S)

Clear all

C# | Windows | Desktop

Recent project templates

ASP.NET Core Web App	C#
ASP.NET Core Empty	C#
WPF Application	C#
WPF Browser App (.NET Framework)	C#
WPF User Control Library	C#
WPF Class Library	C#
.NET MAUI App	C#
Windows Forms App	C#
Windows Desktop Wizard	C++
Windows Forms App (.NET	C#

Windows Forms App
A project template for creating a .NET Windows Forms (WinForms) App.
C# Windows Desktop

Windows Forms App (.NET Framework)
A project for creating an application with a Windows Forms (WinForms) user interface
C# Windows Desktop

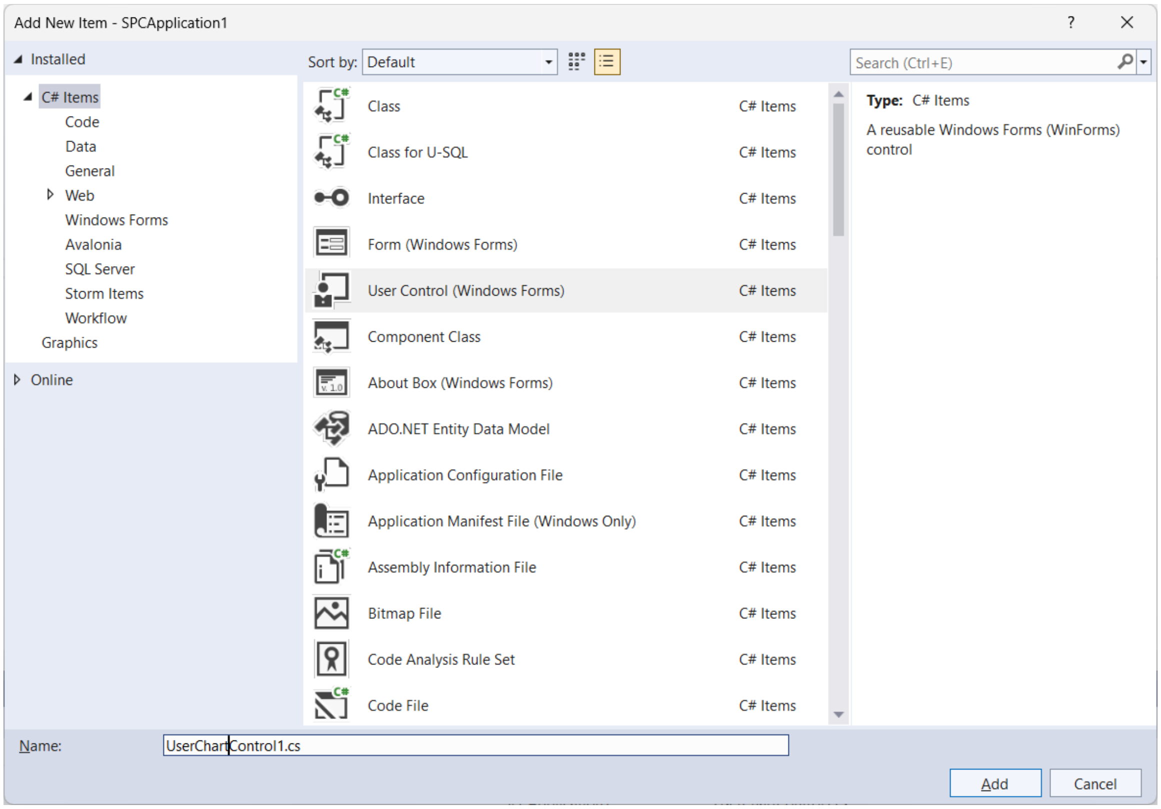
WPF Application
A project for creating a .NET WPF Application
C# Windows Desktop

WPF Class Library
A project for creating a class library that targets a .NET WPF Application
C# Windows Desktop Library

WPF Custom Control Library
A project for creating a custom control library for .NET WPF Applications
C# Windows Desktop Library

Give the project a unique name (our version of this example is SPCApplication1). You will end with a basic Form based application. For purposes of this example, the chart is placed in the initial, default form.

- Add a User Control class to the project (**Project | Add User Control**). Enter a class name of **UserChartControl1**.



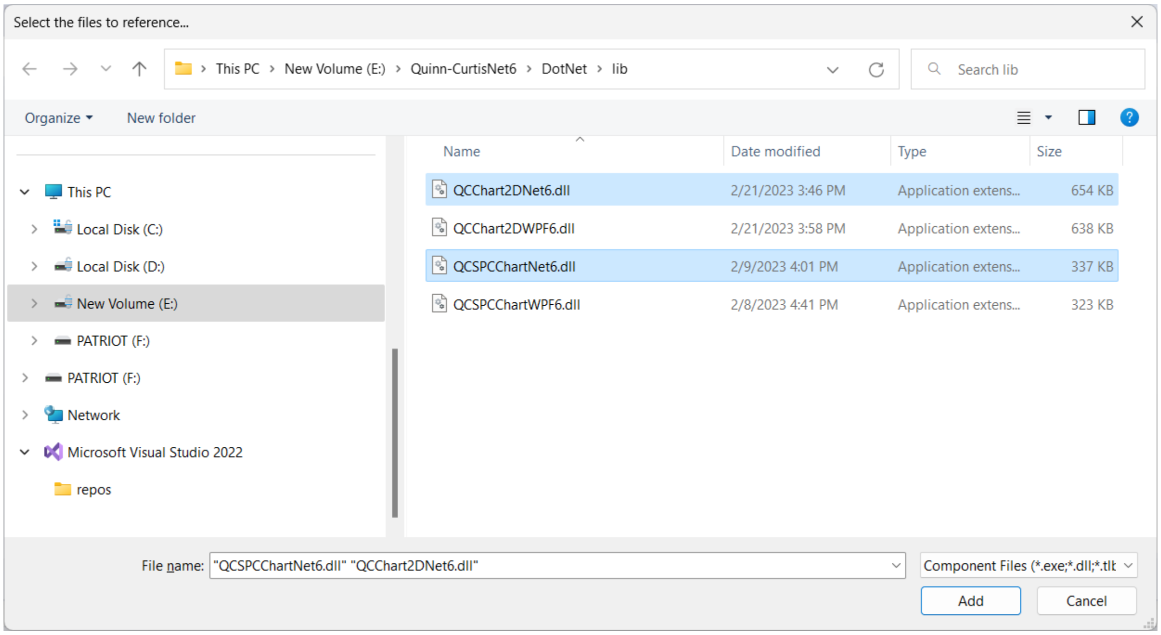
Class	C# Items
Class for U-SQL	C# Items
Interface	C# Items
Form (Windows Forms)	C# Items
User Control (Windows Forms)	C# Items
Component Class	C# Items
About Box (Windows Forms)	C# Items
ADO.NET Entity Data Model	C# Items
Application Configuration File	C# Items
Application Manifest File (Windows Only)	C# Items
Assembly Information File	C# Items
Bitmap File	C# Items
Code Analysis Rule Set	C# Items
Code File	C# Items

Type: C# Items

A reusable Windows Forms (WinForms) control

Name: UserChartControl1.cs

- Right click on Add | Project Reference in the Solution Explorer window and Browse to the Quinn-Curtis/DotNet/lib subdirectory and select and add ***BOTH*** the **QCSPCChartNet6.DLL** and **QCChart2DNet6.DLL**.



Name	Date modified	Type	Size
QCChart2DNet6.dll	2/21/2023 3:46 PM	Application extens...	654 KB
QCChart2DWPF6.dll	2/21/2023 3:58 PM	Application extens...	638 KB
QCSPCChartNet6.dll	2/9/2023 4:01 PM	Application extens...	337 KB
QCSPCChartWPF6.dll	2/8/2023 4:41 PM	Application extens...	323 KB

File name: "QCSPCChartNet6.dll" "QCChart2DNet6.dll"

Component Files (*.exe;*.dll;*.tlk

View the UserChartControl1.cs code. Change the base class of UserChartControl1 to one of the SPC Chart classes derived from com.quinn-curtis.chart2dnet6.ChartView. These are:

SPCEventVariableControlChart
SPCEventAttributeControlChart
SPCTimeVariableControlChart
SPCTimeAttributeControlChart
SPCBatchVariableControlChart
SPCBatchAttributeControlChart
FrequencyHistogramChart
ProbabilityChart
ParetoChart

This adds a local version of the control to the project. The C# form code should now look like:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Drawing;
using System.Data;
using System.Text;
using System.Windows.Forms;
using System.Drawing.Drawing2D;
using com.quinn-curtis.chart2dnet6;
using com.quinn-curtis.spcchartnet6;

namespace SPCApplication1
{
    public partial class
UserChartControl1 : .quinncurtis.spcchartnet.SPCEventVariableControlChart
    {
        public UserChartControl1()
        {
            InitializeComponent();
        }


    }
}
```

Note that the file uses a class modifier of *partial*. This means that there is more code associated with the class in the related UserControlChart1.Designer.cs file. Normally you will not need to edit that file.

- ***Critical Step:*** Make sure you add the following lines to the top of the **UserChartControl1.cs** code to resolve the **QCSPCChart** and other graphics classes used in the example.

  ```
  using System.Drawing;
  using System.Drawing.Drawing2D;
  using com.quinn-curtis.chart2dnet6;
  using com.quinn-curtis.spcchartnet6;
  ```

- Build the Solution (**Build | Build Solution**). This will compile the **UserChartControl1**class and make it accessible as a component on the Toolbox. If the project fails to compile, go back and check the previous steps.

  You can create as many custom chart controls as your application requires. Each custom chart control will inherit from one of the SPC chart classes listed above.

- Right click on the **UserChartControl1** class form and view the underlying C# code. We placed all of the chart customization code in the **InitializeGraph** method. Until this method is called, the **UserChartControl1**class appears as an empty shell. Your can call this method from the **UserChartControl1**class constructor;

```
public UserChartControl1()
{
    // This call is required by the Windows.Forms Form Designer.
    InitializeComponent();

    // TODO: Add any initialization after the InitForm call
    // Have the chart fill parent client area
    this.Dock = DockStyle.Fill;
    // Define and draw chart
    InitializeChart();
}
```

or from somewhere outside of the class to avoid problems associated debugging errors in user controls at design time.

- Go to the main form, **Form1**. You can use either of the two following method to place the **UserChartControl1**class on the form.

  1. In the Form1 source file, add a variable of type **UserChartControl1** in the declaration section, and in the Form1 constructor, instantiate the class and add it to the Form1 controls list:

  or

  2. Go to the toolbox and select the **UserChartControl1** from the Windows Forms list. Drop it onto the main form and size it.

- Define the chart by customizing the **UserChartControl1.InitializeChart** method. See the SPCApplication1.UserChartControl1.cs file for the complete code listing.

```
ChartCalendar startTime = new ChartCalendar();
// The time increment between adjacent subgroups
int timeincrementminutes = 15;
// Number of samples per sub group
int numsamplespersubgroup = 5;

.
.
.

public void InitializeChart()
{
    //  SPC variable control chart type
    int charttype = SPCControlChartData.MEAN_RANGE_CHART;

    // Number of datapoints in the view
    int numdatapointsinview = 17;


    // Initialize the SPCTimeVariableControlChart
    this.InitSPCEventVariableControlChart(charttype,
        numsamplespersubgroup, numdatapointsinview);
    this.HeaderStringsLevel = SPCControlChartData.HEADER_STRINGS_LEVEL1;

   // Set the strings used in the header section of the table
    this.ChartData.Title = "Variable Control Chart (X-Bar & R)";
    this.ChartData.PartNumber = "283501";
    this.ChartData.ChartNumber="17";
    this.ChartData.PartName= "Transmission Casing Bolt";
    this.ChartData.Operation = "Threading";

    // Display the Sampled value rows of the table
    this.EnableInputStringsDisplay= true;
    // Display the Sampled value rows of the table
    this.EnableCategoryValues= true;
    // Display the Calculated value rows of the table
    this.EnableCalculatedValues= true;
    // Display the total samples per subgroup value row
    this.EnableTotalSamplesValues= true;
    // Display the Notes row of the table
    this.EnableNotes= true;
    // Display the time stamp row of the table
    this.EnableTimeValues = true;
        .
        .
        .
}
```

- You should be able to compile the project without error. No chart will be visible yet.

- You should now be able to compile, run and view the entire project. Any changes you make in the **UserChartControl1** form will be reflected in the application. If you still have problems go back and study the many example programs we have provided.

Index