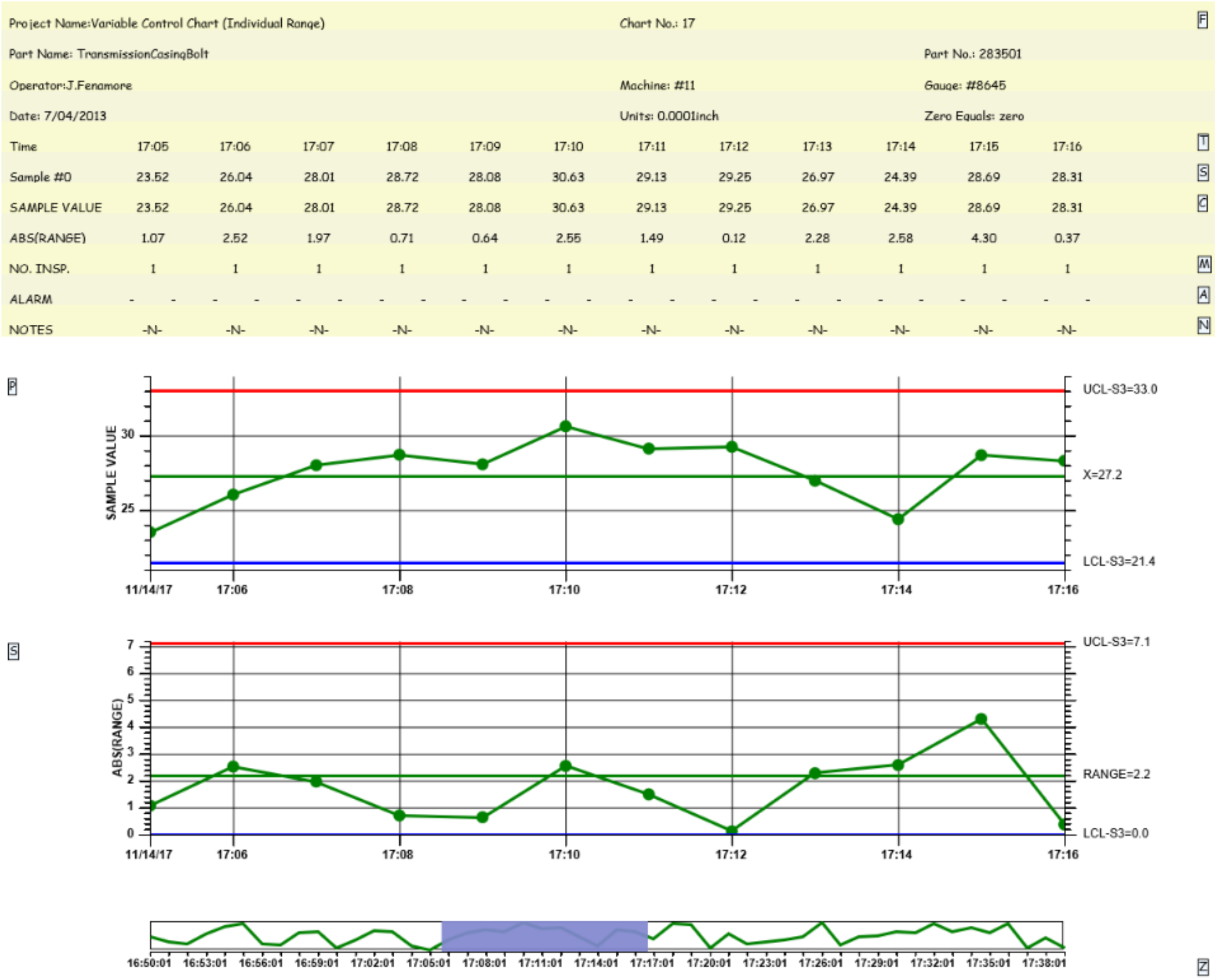
QCSPCChart+ SPC Control Chart Tools for Javascript (Rev. 3.6)



Project Name:Variable Control Chart (Individual Range) Chart No.: 17 F

Part Name: TransmissionCasingBolt Part No.: 283501

Operator:J.Fenamore Machine: #11 Gauge: #8645

Date: 7/04/2013 Units: 0.0001inch Zero Equals: zero

Time	17:05	17:06	17:07	17:08	17:09	17:10	17:11	17:12	17:13	17:14	17:15	17:16	T
Sample #0	23.52	26.04	28.01	28.72	28.08	30.63	29.13	29.25	26.97	24.39	28.69	28.31	S
SAMPLE VALUE	23.52	26.04	28.01	28.72	28.08	30.63	29.13	29.25	26.97	24.39	28.69	28.31	C
ABS(RANGE)	1.07	2.52	1.97	0.71	0.64	2.55	1.49	0.12	2.28	2.58	4.30	0.37	
NO. INSP.	1	1	1	1	1	1	1	1	1	1	1	1	M
ALARM	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	A
NOTES	-N-	-N-	-N-	-N-	-N-	-N-	-N-	-N-	-N-	-N-	-N-	-N-	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 10/11/2017 Rev. 3.6

Quinn-Curtis, Inc. Tools for Javascript 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.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 do not allow developers to use this SOFTWARE to create a graphics or charting 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. Should you need to do this, you need to have a licensing agreement with Quinn-Curtis which permits redistribution of our libraries as part of a development system.

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. QCSPCChart+ SPC Charting Tools for Javascript (Enhanced Version)

Introduction

The **QCSPCChart+ for Javascript** software represents an adaptation of the **QCSPCChart** library to the **Javascript** and the HTML5 user interface framework. It was created utilizing the Google GWT (Google Web Toolkit) development tool. All of the system dependent graphics (.Net GDI+, Java graphics, Android graphics, etc) have been replaced with HTML5, Canvas-based, Javascript equivalents. The result is an easy to use, interactive, SPC Charting package which will run on any computer which supports a modern browser. A modern browser is considered any browser which supports HTML5, and most importantly, supports the HTML5 Canvas element. The browsers listed below meet this requirement.

IE (9+)
Firefox (1.5+)
Safari (1.3+)
Chrome
Opera (9+)

While IE 8 (Internet Explorer) supports a limited subset of HTML5 features, it does not support the Canvas element to the degree required by this software, and therefore is considered incompatible.

Enhancements added to Revision 3.6 of QCSPCChart+

QCSPCChart+ 3.6 is a new revision of the QCSPCChart+ for Javascript software. It includes enhancements that make it symetrical with the 3.0 versions of QCSPCChart for .Net and QCSPCChart for WPF. These enhancements include:

1. Charts have been converted to always use Event coordinates. Previously one of two different coordinate systems were used: Time (for time-based control charts) and Cartesian (for batch-based control charts). The Event coordinate system replaces both because it has aspects of both Time- and Event-based coordinate systems.
2. Zooming as an option for the scrollbar (UI option)

3. Collapsible Items (UI option)
4. Reset N of M counters for control moves (Programming option)
5. Remove Negative LCL control limits for Variable Control Secondary
6. charts, or Attribute Control Primary charts (Programming option)
7. N of M testing when the most recent point entering the test is within
8. limits (Programming option)
9. A Levey-Jennings Variable Control Chart (Programming Option)

Enhancements added to QCSPCChart+

QCSPCChart+ is an enhanced version of our regular QCSPCChart for Javascript software. The enhanced features were added for a customer who gave us permission to make the resulting software commercially available. These enhancements include:

1. The symbol highlighting for out of limit data points will include a change of scatter plot symbol, in addition to the current change of color. This feature applies to all chart types.

2. Enhanced annotations with vertical line and a variety of text justification options. This feature applies to all chart types.

3. Variable control limits and variable specification limits, for each sample interval, for all chart types (including the new ones listed under items 5 and 6).

4. Enhanced limit checking for all chart types: a variable can be marked out of range for any sample interval using a flag (rather than the charts internal control limit test).

5. New single variable SPC chart (Xbar-R-MR), which has three sub plots: Xbar, Range, and MR (Moving Range derived from the Xbar chart). The added MR subplot has its own +-3 sigma set of control limits.

6. Five new Multi-Variable SPC Charts for IX-MR, Xbar-R, Xbar-Sigma, Xbar-R-MR, and a Mixed Chart. The Mixed Chart can combine Xbar-R, Xbar-Sigma, and I-R charts in the same graph. Other Features of the Multi-Variable SPC Charts include:

7. Dual y-scales and axes – a plot will be assigned to one or the other. The dual y-axis scale applies to all sub charts.
   Auto-scaling for all sub plots each with dual y-axis scales
   A legend will identify which plot is associated with which y-axis

8. A variable number of samples per sample interval are permitted for the Primary Variable and Multi-Variable data update. If zero sample are entered for a sample interval, then a dummy record is entered at that point in the data, in order to maintain synchronization with the other records, but the data is considered invalid for plotting purposes in the charts.

Tutorials

Chapter 20 is a tutorial that describes how to define a simple chart and deploy it to a web page. Read the first couple of chapters, and then the tutorial.

HTML

Originally, web pages were static. The purpose of a web browser was to display a static HTML document. The static limitation quickly ran its course and users wanted more and more direct interaction with a web page, analogous to interacting with an application program installed on on a desktop computer. So Javascript was invented to control elements of the web page, dynamically serve up documents, and asynchronously communicating with servers in support user-driven content. Originally meant as a client-side programming language, to be run in a web browser, its role has been expanded to include include desktop, cloud, and server applications.

HTML is a text-based standard format for the display of text and data by a web browser. Javascript can automate elements within HTML to render content to the browser. And HTML elements can call Javascript code in order to process content requests.

HTML5

HTML underwent a major revision upgrade with the introduction of a preliminary HTML5 specification around 2008. HTML5 represents an attempt to integrate many of the features required for the cross platform support of current, and next generation desktop, mobile and cloud applications. It specifically adds elements for the support of audio, video, and vector-based device independent graphics applications. It is now in the final stages of the standardization process, with an final specification expected by the end of 2014. Because HTML5 is expected to play an critical role in the future of the Internet, the major web browsers have already adopted most all of the features set forth in the working drafts of the specification.

According to the late Steve Jobs, the future of web programming is HTML5. This comment was prompted in an interview with Jobs about his ongoing feud with Adobe and their ubiquitous Flash player plug-in. Even though the Flash player had the dominant market share as means of displaying audio and video in a web browser, Jobs refused to permit support for Flash in the Apple iOS mobile operating systems. His stated reason was that HTML5, with its extensive support for audio and video, renders Flash technology obsolete. Why install a separate plug-in (Flash), when a modern browser with integrated HTML5 support offers vastly superior routines for rendering audio and video. The way to get at all of the features of HTML5 is to use Javascript. Javascript and HTML5 are now supported on all major browsers, running on all major operating systems, for both desktop and mobile platforms. Sounds ideal. You might think that all you have to do is use HTML5 and Javascript in your web page, and it will work flawlessly in every case. But you would be wrong. HTML5 implementations are left up to the web browser companies, rather than a central controlling authority, and because of this, they all differ. Developers programming directly in Javascript have to become familiar with the differences in the HTML5 support across browsers. In their Javascript code they must detect which browser is executing the Javascript code, and branch to code specific to that browser. For an advanced application, this can be very tedious and time consuming.

Javascript

Javascript is an interpreted programming language created to make web browsers more dynamic and interactive. It was originally developed by Netscape in 1995 as a feature of their web browser. While it includes "Java" as the root of its name, that was a unfortunate marketing gimmick. Netscape picked the name solely to ride the coat-tails of the hype being written about the Java programming language introduced at approximately the same time. The only resemblance Javascript has to the Java programming language is a small degree of syntactical similarity, mostly the result of both languages being strongly influenced by C++ (for Java) and C (for Javascript). Since that time, all of the major browsers have added support for Javascript.

GWT (Google Web Toolkit) as a Productivity Tool

The conversion of our QCSPCChart software to Javascript and HTML5 posed significant challenges. While Javascript has some object-oriented features, it is not a true object-oriented language. So adapting software written in an OOP language (C#, Java, and C++, among others) to Javascript, is a giant step backwards. We ruled out manually translating the software into Javascript as impractical based on time and cost constraints.

We have versions of our current products (QCChart2D, QCRTGraph, QCSPCChart, QCMatPack, and QCChart3D) in both Java and C#. We investigated what tools were available to translate Java and C# to Javascript, and quickly arrived at the conclusion that the Google GWT, used in conjunction with the Eclipse development environment, was the only workable option. So that locked us into to using a Java version of QCSPCChart as the code base. We have two different variants of QCSPCChart written in Java. The first is a pure Java version suitable for Java desktop, applet and servlet applications. That version will run on Windows and Linux machines. The second is a version written for the Android platform which runs predominantly on phones and tablets. The main difference between the two versions is that they utilize different graphics libraries. The pure Java versions makes calls to the standard Java **java.awt.graphics** libraries. These libraries are not available under Android, and in the Android version of QCSPCChart we handle screen output by drawing to a Android Canvas object using the standard Android **android.graphics** library.

The GWT programming environment supports neither **java.awt.graphics**, nor **android.graphics** graphics libraries. Instead they have a third library, **com.google.gwt.canvas**, optimized for Javascript and HTML5. So we created a new version of the QCSPCChart software around the GWT **com.google.gwt.canvas** graphics library. It is highly optimized for rendering graphics and text in an HTML5 Canvas object, and because of this it can't be used in browsers which don't have a full HTML5 Canvas implementation. The only commonly used browser this rules out is IE8, because the HTML5 Canvas object is only partially supported in that version. The software works fine with IE9 and IE10. In fact, we find the hardware acceleration of HTML5 Canvas rendering is fastest with IE9 and IE10, compared to Firefox, Chrome, and Safari.

GWT supports a subset of the standard Java libraries, in order to minimize the complexity of the Java to Javascript cross compiler. While somewhat limiting, it is a useable subset. Whenever we ran into library calls in our original code which were not supported by GWT, we just substituted other, similar library calls which were supported, or in some cases we just recreated the function of the original library call and added it into our QCSPCChart library.

So, once a Java code base is written in GWT compatible code, it can be compiled into a Javascript version of the same code base. A majority of GWT programmers write an entire application using GWT. For example, a typical application would be a browser based e-mail app, such as Google **gmail**. The e-mail system is written in Java, compiled using GWT, and the compiler output is a set of Javascript files. You copy the Javascript files to the server and invoke the application from an HTML page which references the GWT files. Direct Javascript calls, from handwritten Javascript code calling internal library functions of the application is not really supported. This is because the Javascript code generated by the GWT compiler, is highly optimized, compressed, and obfuscated, and it also undergoes a major structural change as the OOP source code (Java) is translated into non-OOP code (Javascript). There are ways around this using a feature of GWT call JSNI (JavaScript Native Interface) and we utilize that feature in a few critical areas. But in general, the programmer (i.e. you), will not be calling our Javascript library functions directly.

The GWT compiler produces up to six variants of your web program, one for each of the major browsers it supports. GWT automatically processes the differences in HTML5 support across the major browsers, and generates browser specific Javascript in support of your original Java source program. When the browser starts executing the GWT generated program, the first thing it does is detect which browser the code is executing in, and then loads the Javascript module appropriate to the browser, using a technique known as deferred binding. This is very efficient, because only the highly optimized code for the specific browser is downloaded from the server to the client, all of the other versions are left behind.

GWT (Google Web Toolkit) is extensively documented by Google on their web site:
http://www.gwtproject.org/
Here is a summary of GWT on Wikipedia: http://en.wikipedia.org/wiki/Google_Web_Toolkit

JSON as the Scripting Language for an SPC Chart

As we said earlier, most users of GWT write their application program in Java and compile it into Javascript, then deploy the resulting Javascript files and folders to a website. Unfortunately, this does not work if you are a third party vender who wants to create a library for use by programmers writing their application using HTML and Javascript on a web page. First, you don't have the underlying source code (written in Java using GWT libraries) to our QCSPCChart library. Second, you probably don't want to get involved with Java and GWT – it's outside of your comfort zone. Third, you can't call into our Javascript libraries, even if you are using Javascript, because the libraries are compressed, un-objectived, and obfuscated, a by-product of the GWT compile.

So, a programmer cannot customize SPC charts using Javascript calls, at least not in the same fashion as you do with the .Net and Java versions of our software. Instead, you will customize SPC charts using a scripting language we have developed, utilizing JSON (JavaScript Object Notation). JSON is a widely used, text-based open standard designed for human-readable data interchange. It can be used with virtually any language, though I expect that you will imbed the JSON in Javascript found in the host HTML page. A typical SPC chart script looks like:

```
{
    "SPCChart": {
        "InitChartProperties": {
            "SPCChartType": "MEAN_RANGE_CHART",
            "ChartMode": "Batch",
            "NumSamplesPerSubgroup": 5,
            "NumDatapointsInView": 12,
            "TimeIncrementMinutes": 15
        },
        "Scrollbar": {
            "EnableScrollBar": true
        },
        "TableSetup": {
            "HeaderStringsLevel": "HEADER_STRINGS_LEVEL2",
            "EnableInputStringsDisplay": true,
            "EnableCategoryValues": false,
            "EnableCalculatedValues": true,
            "EnableTotalSamplesValues": true,
            "EnableNotes": true,
            "EnableTimeValues": true,
            "EnableNotesToolTip": true,
            "TableBackgroundMode": "TABLE_NO_COLOR_BACKGROUND",
            "TableAlarmEmphasisMode": "ALARM_HIGHLIGHT_BAR",
            "ChartAlarmEmphasisMode": "ALARM_HIGHLIGHT_SYMBOL",
            "ChartData": {
                "Title": "Variable Control Chart (X-Bar R)",
                "PartNumber": "283501",
                "ChartNumber": "17",
                "PartName": "Transmission Casing Bolt",
                "Operation": "Threading",
                "SpecificationLimits": "27.0 to 35.0",
                "Operator": "J. Fenamore",
                "Machine": "#11",
                "Gauge": "#8645",
                "UnitOfMeasure": "0.0001 inch",
                "ZeroEquals": "zero",
                "DateString": "7/04/2013",
                "NotesMessage": "Control limits prepared May 10",
                "NotesHeader": "NOTES"
            }
        },
        "Events": {
            "EnableDataToolTip": true,
            "EnableJSONDataToolTip": false,
            "AlarmStateEventEnable": false
        },
        "PrimaryChartSetup": {
            "FrequencyHistogram": {
                "EnableDisplayFrequencyHistogram": true
            }
        },
        "SecondaryChartSetup": {
            "FrequencyHistogram": {
                "EnableDisplayFrequencyHistogram": true
            }
        }
    }
}
```

This example (BatchXBarR) is extracted from an example script (chartDefExampleScripts.js) where you will find many of the example listed in this software. The are many more options than the ones seen in the example above.

JSON data structures can be defined using Javascript, as we do in all of our example web pages, and then converted to a JSON string using a Javascript utility function named, appropriately enough, JSON. The JSON.stringify function will take a Javascript data structure and covert it to a JSON string. All of our examples convert a Javascript data structure into JSON using JSON.stringify. You can also go the other direction. Some event processing routines in our library (alarms, tooltips, and data retrieval) will return data embedded in a JSON string. You can convert a JSON string into a Javascript data object using the JSON.parse function. You want the values as a Javascript data object because then you can access the data using standard Javascript dot notation:

```
      var s = pushGetJSONSampleIntervalData(32);

      var jsonobj = JSON.parse(s);

      var s2 = jsonobj.SPCSampleIntervalData.PrimaryChartAlarmMessage;
```

JSON is widely documented on the web, so try and read the following links:

http://www.json.org/ - Introducing JSON

http://www.json.org/js.html – JSON in Javascript

SPC Control Chart Tools 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.

The current SPC trend is for data to be centralized on a server in a database, and the display to be localized on the client computer. The local display on the client can be a desktop application, or a web-based application. We have several versions of QCSPCChart for the display of SPC data in a desktop application: specifically for .Net, WPF (Windows Presentation Foundation), and Java. For mobile applications, we have QCSPCChart for Android. For web based applications we have a Silverlight version. It is also possible to use the .Net, WPF, and Java versions in a web application, though each has their drawbacks.

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)
Xbar-R-MR (Mean-Range-Moving Range Chart)
Multi-Variable SPC Charts for :
IX-MR, Xbar-R, Xbar-Sigma, Xbar-R-MR, and a Mixed Chart.

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

Design Considerations

- Minimal programming required – create SPC charts with a few lines of Javascript code, and JSON, 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.

- 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.

- Support for named control rule sets: WE, Nelson, AIAG, Juran, Hughes, Duncan, Westgard, and Gilow

- Support for custom control rule sets based on our pre-defined templates.

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

- Data point annotations

- Alarm forcing – An sample interval can be forced into alarm

- Alarm highlighting – Symbols can change color or shape when a sample interval is in an alarm condition.

- Variable control limits and specificationlimits for all chart types
-

The **SPC Control Chart Tools for Javascript** 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 a JSON script. 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 Javascript** uses HTML 5 features including:

- Resolution independence. HTML 5's emphasis on vector graphics means that programs can be more easily designed to be independent of the resolution of the output device.
- Arbitrary line thickness and line styles for all lines.
- Gradients, fill patterns and color transparency for solid objects.
- Improved font support for a large number of fonts, using a variety of font styles, size and rotation attributes.

Web-Based Versions of QCSPCChart

Not counting this, the Javascript version of QCSPCChart, we have four different variants of QCSPCChart which can be used to display charts in web browsers. Each variant has weaknesses which prevent it from being a true cross-platform, cross-browser method used to implement a truly interactive chart.

Web Browser Application Frameworks which are not Javascript

.Net – Using the regular .Net version, you can write ASP.Net applications which run on a server in *headless* mode. *Headless* means that custom chart images are created on demand and converted to jpg or png format without being specifically rendered to a server workstation monitor. The converted images are transferred to the client-side workstation and displayed in an HTML (or Asp .Net equivalent) image element. The drawback of this techniques is that the chart is not interactive. Data tool tips, editing, scrolling, and real-time updates are all either limited, or non-existent. Also, the host server is limited to ones supporting Asp .Net program development. This requires Administrator privileges on the host server not granted to many developers using shared servers on third party hosts.

Java – Java has a long history of being used to write desktop, server and client side applications. Using Java, it is possible to write cross platform, client-side web browser applications which render graphics directly on the local workstation. When a Java application is run in a browser, and the browser host does not have a Java run-time installed, a workstation specific subset of the Java run-time environment is downloaded into the client memory, and that run-time is used to execute (interpret) the Java application program. Unfortunately, it contains many security holes which render it unusable in high security applications. Even though Sun (the controlling authority behind Java) seems to issue new revisions of Java weekly to fix old security flaws, new holes seem to appear just as fast. Java is simply to easy for malicious hackers to subvert. Over the last several years, web browsers have carefully reduced the number of Java features they support, making it a moving target to write against, since programs which work under IE 6 may not work under IE8, IE9 and IE10. Modern browsers are turning Java support off as part of their recommended security settings.

WPF – WPF and Silverlight are very similar. While Silverlight is strictly a framework for writing rich internet applications, WPF can be used to write applications for both the desktop and the internet. It has much in common with Silverlight, and both share an XAML-based method of creating the user interface for an application, and both are normally programmed using a .Net language (C# or VB). They do not share a common run-time though. So WPF requires a different run-time plug-in than Silverlight. Also, WPF browser applications will only run on workstations which have a recent version of the .Net run-time installed. This means it has more limited browser support than Silverlight, with no support of Apple workstations, no support for Linux, and no support for mobile applications (IOS and Android). There are also rumors that Microsoft is placing WPF in the same *no further development* category as Silverlight.

Silverlight – Silverlight was intended to be Microsoft's answer to Java as a client-side programming language for web browser applications. First introduced in 2007, its most recent version is Version 5.0, released in 2011. Silverlight run-time plug-ins are available for browsers running on Windows and OSX (Apple) based workstations. Silverlight was a significant innovation for Microsoft. It does provide a very powerful application framework for writing rich internet applications. Unfortunate, early attempts (Moonlight) to port the run-time to Linux-based browsers have been abandoned. Also, no plug-ins were ever created for browsers running on mobile devices running IOS (Apple) and Android (Google). Even Microsoft's own Mobile phone does not support Silverlight. The general consensus is that Microsoft has stopped development work on Silverlight. They will probably support it for years to come, because of the large installed base, but it is a dead end development-wise.

QCSPCChart for Javascript

So, how do you combine these elements: the QCSPCChart software, GWT, and your web site. First, we have compiled the QCSPCChart software, using GWT, into the various browser specific components. All of the compiler output is found in a folder named QCSPCChartGWTWar. Also in that folder is one or more HTML files. You would place a copy that folder on your web site, probably in the root of the web site, though that does not have to be the case.

All of the examples, SPCSimple.html for example, have a block of standardized code in the header section used to invoke the GWT generated Javascript files making up QCSPCChart. This block of code looks like:

```
   <script type="text/Javascript" language="Javascript"
src="qcspcchartgwt/qcspcchartgwt.nocache.js"></script>
```

When the web page is loaded, this line of code is executed and everything else cascades from there.

You will need to define a chart. This is done using a JSON construct as described earlier. We use a simple data format, compatible with JSON, and Javascript data structures. That structure sets the properties of the chart you want to create. Most all of the hundreds of properties are optional and if you don't set a property it is assigned a default value. In our main example, SPCExampleScripts, we have created Javascript objects for each of the main chart types supported by the software. Some include data initialization using actual data values, others use simulated data. Some use the integrated table above the charts, others do not. Below is a chart script for a minimal IR chart.

```
 var TimeIR =
{
     "SPCChart": {
          "InitChartProperties": {
             "SPCChartType": "INDIVIDUAL_RANGE_CHART",
             "ChartMode": "Time",
             "NumSamplesPerSubgroup": 1,
             "NumDatapointsInView": 12,
             "TimeIncrementMinutes": 15
          },

          "UseNoTable": {
             "PrimaryChart": true,
             "SecondaryChart": true,
             "Histograms": true,
             "Title": "Individual Range Chart Part XKY"
          },
          "SampleData": {
             "DataSimulation": {
             "StartCount": 0,
             "Count": 50,
             "Mean": 27.5,
             "Range": 5
          }
      },
      "Methods": {
          "AutoCalculateControlLimits": true,
          "AutoScaleYAxes": true,
          "RebuildUsingCurrentData": true
      }
    }
 };
```

Simple JSON Formatting Rules in a Nutshell

The left-hand side, or property name, of of each Property/Value pair is always a string, represented as quoted text: "SPCChart", "InitChartProperties", "SPCChartType", "ChartMode", etc. The right-hand side can have several different formats, depending on the context. If the property value is a JSON numeric value, it is represented without quotes, for both real and integer (50, 27.5, 5). If the property value is a JSON boolean value, it is represented as either true or false, without quotes. If the property value is a string, it is represented using a quoted text ("Individual Range Chart Part XKY"). If the property value is a QCSPCChart constant ("INDIVIDUAL_RANGE_CHART") it is also represented using quoted text. A property can also be the parent of a subgroup of Property/Values pairs , "InitChartProperties" for example. In that the subgroup is bracket in {..}. Also, a property can represent an array of values, in which cast the values are braketed by [..]. You will see examples of that later. So in summary:

Property Name (left hand side)	Always in quotes
Numeric values (right hand side)	A number with no quotes
Boolean values (right hand side)	true or false, no quotes
String values (right hand side)	Text in quotes
An array (right hand side)	Comma separated values surrounded by brackets [..]
Another child block	Surrounded in curly brackets {.. }

The "InitChartProperties" block defines the SPC chart type (Individual Range), the mode (Time or Batch), the number of samples per sub group (always one for an IR chart) and the number of data points in the view, and the time increment between adjacent samples. If you wanted to initialize a Batch version of the same chart, the only thing which would change would be :

```
 "InitChartProperties": {
     "SPCChartType": "INDIVIDUAL_RANGE_CHART",
     "ChartMode": "Batch",
     "NumSamplesPerSubgroup": 1,
     "NumDatapointsInView": 12,
  },
```

The **ChartMode** has been changed to "Batch" and the **TimeIncrementMinutes** has been removed, since it doesn't apply to a batch chart.

The "UseNoTable" block is a utility to remove the entire table, and to just set the most common default properties for the chart in general. You can enable/disable the display of the primary chart, secondary chart, the histograms and the title.

```
"UseNoTable": {
      "PrimaryChart": true,
      "SecondaryChart": true,
      "Histograms": true,
      "Title": "Individual Range Chart Part XKY"
},
```

Data is simulated. The parameters for the simulation simulated in the "DataSimulation" block, under "SampleData".

```
"SampleData": {
      "DataSimulation": {
          "StartCount": 0,
          "Count": 50,
          "Mean": 27,
          "Range": 5
       }
}
```

Fifty sample values are simulated, with a mean of 27 and a range of 5. If you want so substitute real values, then the block would look something like this:

```
               "SampleData": {
                       "SampleIntervalRecords": [
                       {
                           "SampleValues": [
                               27.53131515148628
                           ],
                           "BatchCount": 0,
                           "TimeStamp": 1371830829074,
                           "Note": ""
                       },
                       {
                           "SampleValues": [
                               27.444285005240214

                           ],
                           "BatchCount": 1,
                           "TimeStamp": 1371831729074,
                           "Note": ""
                       },
                       {
                           "SampleValues": [
                               35.21321620109259,
                           ],
                           "BatchCount": 2,
                           "TimeStamp": 1371832629074,
                           "Note": ""
                       },
                       {
                           "SampleValues": [
                               27.898302097237174
                           ],
                           "BatchCount": 3,
                           "TimeStamp": 1371833529074,
                           "Note": ""
                       },
                       {
                           "SampleValues": [
                               22.94549873989527,
                           ],
                           "BatchCount": 4,
                           "TimeStamp": 1371834429074,
                           "Note": ""
                       },
                 .
                 .
                 .
                       {
                           "SampleValues": [
                               22.94549873989527,
                           ],
                           "BatchCount": 49,
                           "TimeStamp": 1371834429074,
                           "Note": ""
                       }
                ]
}
```

Note that the SampleIntervalRecords block is an array of child blocks, and within each element of that array, there is another array, SampleValues, which is an array of numeric values, representing sample data.

The last block, "Methods", is a list methods which can be executed after the "SampleData" block is processed. These items represent methods for auto-calculating control limits, auto-scaling the y-axes of the charts, and the rebuilding of the chart taking into account new values.

There are many more options you can set in the defining JSON script. You can create the chart using one JSON script, and update it using another. You can reset the sample values back to empty, and run a new set of data through the chart. Subsequent chapters in the manual go into more detail about the available options.

Dynamic Creation of JSON

Typically, the chart creation JSON script will be static, or nearly static. That means you can hand-code a template for a specific application. You can customize specific properties of the template using standard Javascript programming. For example, you start with the TimeXBarR example from the chartdefSimple.js file. All of the property values of the TimeXBarR record variable are filled-out with default values. But now you want to customize it a bit. You can use standard Javascript to do that, referencing the fields of the TimeXBarR record variable.

```
function defineChartUsingJSON( )
{
   TimeXBarR.SPCChart.TableSetup.ChartData.Title = "QC Mean Range Chart";
   TimeXBarR.SPCChart.TableSetup.ChartData.PartNumber = "122";
   TimeXBarR.SPCChart.TableSetup.ChartData.ChartNumber = "3";
   TimeXBarR.SPCChart.TableSetup.ChartData.PartName = "Widget X23";
   TimeXBarR.SPCChart.TableSetup.ChartData.Operation = "Flange Drilling";
   TimeXBarR.SPCChart.TableSetup.ChartData.Operator = "Mike Holtzman";

   var s = JSON.stringify(TimeXBarR);
  return s;
}
```

Data updates, using the SampleData.SampleIntervalRecords property involves appending new elements to an already existing array. Use the Javascript push function to do that. In the example below, all of the sample data is stored as an array of records within TimeXBarR.SPCChart.SampleData.SampleIntervalRecords. Each element of SampleIntervalRecords contains a structure which contains a SampleValues array, which is an array of values, one for each sample of a sample interval. So an array of SampleValues is created, and populated with sample data for a sample interval. That array is combined with BatchCount, TimeStamp, and Note data values in a SampleIntervalRecord, and that records is appended at the end of TimeXBarR.SPCChart.SampleData.SampleIntervalRecords using **push**.

```
function defineChartUsingJSON( )
   {
   var SampleIntervalRecord = {
                   "SampleValues": [],
                   "BatchCount": 0,
                   "TimeStamp": 1371830829074 + 20 * 900000,
                   "Note": "" };
   var SampleValues = new Array();

   SampleValues.push(27.53131515148628);
   SampleValues.push(33.95771604022404);
   SampleValues.push(24.310097827061817);
   SampleValues.push(28.282642847792765);
   SampleValues.push(30.2908518818265);

   SampleIntervalRecord.SampleValues = SampleValues;
   TimeXBarR.SPCChart.SampleData.SampleIntervalRecords.push(SampleIntervalRecord);


   var s = JSON.stringify(TimeXBarR);
   return s;
 }
```

JSON and Web Services

JSON is also widely used in what are called web services, or communication between a client and a server. JSON is used as the message system, encapsulating data structures in simple text which can be easily converted to and from Javascript, without having to write dedicated parsers. It replaces XML data formats in many instances. While JSON is not as flexible or as all encompassing as XML, it does have many advantages. JSON is generally faster, less verbose, and easier to read in raw form. You probably use a JSON web service everyday, because Google uses it to transfer data to and from their ubiquitous web search edit box, offering up full-word selections even as you are still typing in letters.

All of the examples we use in the software assume that the chart defining JSON scripts are either present on the server in the form of Javascript include files, or, they are generated dynamically by our library, as in the case of simulated data, or they are generated dynamically in the HTML web page using Javascript. Another possibility is that the application program on the server generate JSON scripts and send them directly to the client HTML page using web services, most likely using some variant of Ajax technologies. Server-side implementations of this are going to be unique to the server. Linux-based servers are probably going to use Java Servlets to serve up JSON scripts, while Microsoft-based servers are probably going to use one of their .Net-based web service libraries, of which they have a confusing assortment. On the server end, there are libraries for every server programming language which will aid you in creating dynamic JSON scripts. On the client end, you can stick with Javascript, and the jQuery utility library.

This subject is not directly related to QCSPCChart library though. The QCSPCChart library assumes you can obtain JSON formatted scripts, in string format, readable by our library. How you do that is up to you. If you want to experiment with sending and receiving JSON scripts using JSON Web services, that can be done without our library. Once you can successfully send and receive the scripts, then you can introduce our library and start displaying the charts the scripts define in the web page.

You will find a simple example which uses the jQuery.Ajax function ($.Ajax) to communicate with a PHP script running on a server in Chapter 20. It assumes that you PHP running on your server.

Important Javascript vs JSON Considerations

In the example above, a Javascript structure, which will be converted to a JSON string using JSON.stringify, is manipulated as a standard Javascript object. If you have long programmed Javascript, you may expect to see the property (or key) names on the left side of the colon to be defined without quotes. The SampleIntervalRecord declaration becomes:

```
var SampleIntervalRecord = {
                   SampleValues: [],
                   BatchCount: 0,
                   TimeStamp: 1371830829074 + 20 * 900000,
                   Note: "" };
```

where the property names on the left side of the colon are not quoted. This will work in many cases. The actual JSON script is the output of the JSON.stringify function. That function will take the unquoted property names and surround them in quotes as part of the conversion from a Javascript object into a formatted JSON string. If the property names are surrounded by quotes, it just leaves them alone.

Regardless of how you define the property names (with or without quotes), you can still access the property values using standard Javascript dot notation:

var samplevalue =
        TimeXBarR.SPCChart.SampleData.SampleIntervalRecords.SampleValues[0];

You will find that 100% of our examples declare the property names using quotes. Because if you do not use quotes, the Javascript/JSON code cannot be checked using a JSON validator, such as http://jsonlint.com/ . The standard definition of JSON uses quoted property names.

How to Pass the JSON Script into the QCSPCChart Library

Once the chart defining JSON script is created, you need a means of passing the script into the QCSPCChart library. There are several ways of doing this. The first method is to place the Javascript function **defineChartUsingJSON**, in a copy of the skeleton HTML file, QCSPCSkeleton.html. Start with your own copy of the QCSPCSkeleton.html (rename it anything you want) so that you have the important code needed to initialize the parts dependent on GWT. This function should take the Javascript JSON object, convert it to a string using the JSON stringify function, and return the resulting string value. The QCSPCSkeleton.html already includes the defineChartUsingJSON function, so just pretend you copy it into that file.

```
    <!--                                                          -->
    <!--Specify the chart defining javascript file containing JSON script -->
    <!--                                                          -->

  <script src="chartdefUserDefined.js"></script>
```

```
<script>

 function defineChartUsingJSON( )
{
  var s = JSON.stringify(TimeXBarR);
  return s;
}

 </script>
```

When the QCSPCChart library starts, it looks to see if the function **defineChartUsingJSON** is in the parent HTML file (QCSPCSkeleton .html in this case). If it is, it executes the function, and processes the JSON script which is returned. If it isn't found, nothing bad happens, the library just waits until some other chart defining event occurs. This is the best technique to use if you want a default chart to load when the web page is loaded.

A second technique is to push a JSON chart definition into the QCSPCChart library. This requires some HTML element to trigger an event, which then calls a Javascript function, which then pushes a JSON script into the QCSPCChart library by calling the function **pushJSONChartCreate**. The **pushJSONChartCreate** function is an un-obfuscated Javascript function exported from the QCSPCChartGWT libray, so that you can call it from within the main HTML page. In the SPCExampleScripts.html example, we set the default chart on the web page using **defineChartUsingJSON**, and then we let you change the default by selecting a new chart from a drop down list, implemented using an HTML select item. Selecting a new chart using the drop down select element triggers an event, which calls the **displayChart** function. Using the passed in value of the *chartid* variable, the defining chart JSON script is retrieved using **getChartItem**, and then that script is converted to a string and passed into the **pushJSONChartCreate** function.

```
function displayChart(chartid) {
 var chartitem = getChartItem(chartid);
 pushJSONChartCreate(JSON.stringify(chartitem));
}
```

Chapter 15 and the tutorial in Chapter 20 discuss these techniques in more detail.

Directory Structure of QCSPCChart for Javascript

The **SPC Control Chart Tools for Javascript** 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**, QCChart2DJavascriptManual.pdf.

Drive:

Quinn-Curtis\ - Root directory

GWTJavascript\ - Quinn-Curtis GWT / Javascript folder

Docs\ - documentation directory

QCSPCChartGWTDoc.pdf – This document, the User guide

QCSPCChartGWTWar\ - The redistributable folder for deployment

qcspcchartgwt\ – this folder contains the compiled, chached, Javascript libraries for QCSPCChartGWT. There are at least six different version of the libraries optimized for the HTML5 support in the major browsers (IE, Firefox, Chrome, and Safari).

SPCSimple.html – a simple example HTML page representing a Javascript program displying a single SPC Chart (X-Bar R).

ChartdefSimple.js – contains the Javascript/JSON definition of the chart referenced in the SPCSimple.html web page.

SPCMediumSimple.html – a medium-simple example HTML page representing a Javascript program displaying a single SPC Chart (X-Bar R) with real-time updates from JSON objects, alarm processing, and retrieving overall SPC statistics from the chart.

ChartdefMediumSimple.js – contains the Javascript/JSON definition of the chart referenced in the SPCMediumSimple.html web page.

MediumSimpleDataUpdate.js – contains the Javascript/JSON definition of the data update used in the SPCMediumSimple.html example.

SPCComplex.html – a complicated example HTML page representing a Javascript program displaying a single SPC Chart (X-Bar R), WECO rules, simulated data updates, alarm processing, and retrieving overall SPC statistics from the chart, retrieving point specific SPC statistics, and processing mouse clicks and adding an annotation to the chart.

ChartdefComplex.js – contains the Javascript/JSON definition of the chart referenced in the SPCComplex.html web page.

chartdefEnhExamples.js – contains the Javascript/JSON definition of the chart referenced in the EnhQCSPCChartExample.html web page.

EnhQCSPCChartExample.html – an example which demostrates all of the new features added to to Plus version (QCSPCChart+).

SPCExampleScripts.html – a complicated example program which lets you select from a list of over 40 different SPC charts.

ChartdefExampleScripts.js – contains the Javascript/JSON definitions of the charts referenced in the SPCExampleScripts.html web page.

QCSPCSkeleton.html – the shell of a simple example you can use to build your own application. It's the same code as the SPCSimple.html page.

ChartdefUserDefined.js – an example X-Bar R script. It's the same as the ChartdefSimple.js script.

QCSPCChartGWT.css – a css style sheet controlling some of the default characteristics of the chart

There are two versions of the for **SPC Control Chart Tools for Javascript** 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 Javascript** is downloaded in a file named Trial_QCSPCChartPJSR30x. 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 Javascript** is downloaded in a file with a name similar to JSSPCDEV2UR3x0x561x1.zip The developer version does not time out and you can use it to create web pages for web sites. 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.

Tutorials

Chapter 20 is a tutorial that describes how to define a simple chart and deploy it to a web page.

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 use either your own example or a modified version of one of our own examples.

Chapter Summary

The remaining chapters of this book discuss the **SPC Control Chart Tools for Javascript** package.

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

Chapter 3 – Overview of JSON Scripting of SPC Charts – summarizes how SPC charts are defined using a JSON scripting lauguage.

Chapter 4 – Static Property Initialization - describes static properties which can be initialized using JSON scripting.

Chapter 5 - SPC Inital Chart Setup - describes the initial setup of an SPC chart.

Chapter 6 – SPC Data Table Setup – the setup of the optional SPC chart data table**.**

Chapter 7 – SPC Chart Setup - describes setup options for the SPC charts.

Chapter 8 – Adding Data to an SPC Chart – describes techniques for adding data to an SPC chart.

Chapter 9 – Calculate and Update Methods – describes methods for the auto-calculation of control parameters, and the update of the display.

Chapter 10 – Variable Control Charts - describes options associated with variable control SPC Charts. This includes the new Multi-Variable SPC Charts added to QCSPCChart+.

Chapter 11 – Attribute Control Charts - describes options associated with attribute control SPC Charts.

Chapter 12 - Process Capability Ratios and Process Performance Indices - describes how to enable the calculation of - Process Capability Ratios and Process Performance Indices .

Chapter 13 - Named and Custom Control Rule Sets – describes how to setup and modify the control rules of popular SPC control chart rule sets.

Chapter 14 - Event Handling for Alarms and Tooltips – explains how to process control limit alarms, and create custom data tooltips and annotations using Javascript.

Chapter 15 - JSNI Calls into the QCSPCChart Library – how to call utility functions in the library from Javascript.

Chapter 16 – Cascading Style Sheets (CSS) – setting the default font and background color of the charts using cascading style sheet parameters.

Chapter 17 – Frequency Histograms – how to create a standalone frequency histogram.

Chapter 18 – Pareto Chart – how to create a standalone Pareto chart.

Chapter 19 - Regionalization for non-USA English Markets – how to customize the software for non-USA markets.

Chapter 20 - Using SPC Control Chart Tools for Javascript to Create Web Applications - implementing a web site.

2. Standard SPC Control Charts

Variable Control Charts
Attribute Control Charts
Other Important SPC 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 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
  - Xbar-R-MR (Mean-Range-Moving Range Chart)
  - Multi-Variable SPC Charts for :
    - IX-MR, Xbar-R, Xbar-Sigma, Xbar-R-MR, and a Mixed Chart.
- 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)

Time-Based and Batch-Based SPC Charts

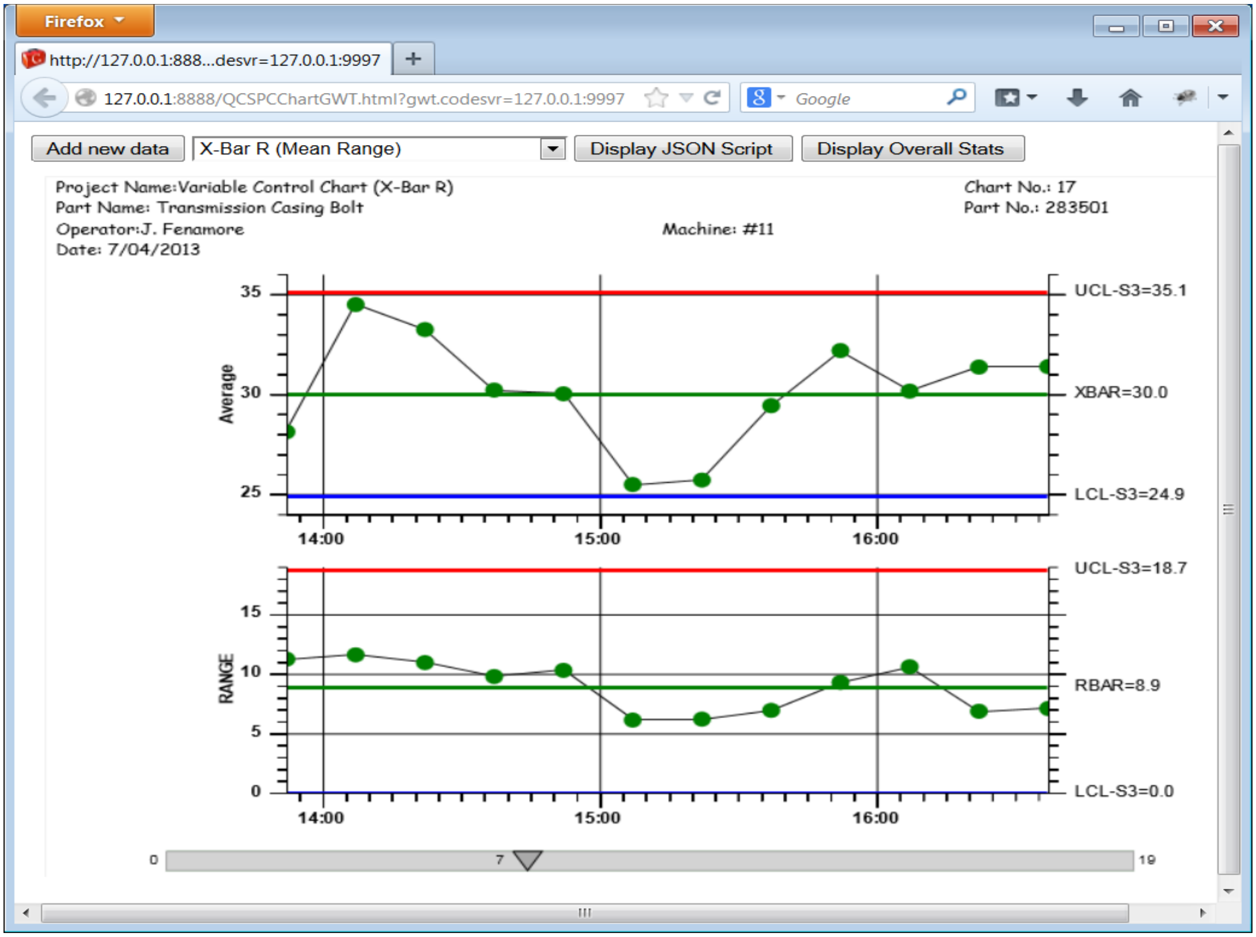
We have further categorized *Variable Control charts* and *Attribute Control Charts* as either time- or batch- based. While you may not find this distinction in SPC textbooks (we didn't), it makes sense to us as charting experts. Quality engineers use time-based SPC charts when data is collected using a subgroup interval corresponding to a specific time interval. They 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. 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.

Note: Starting with Revision 2.0, batch control charts can label the x-axis using one of three options: numeric labeling (the original and default mode), time stamp labeling, and user defined string labeling. Since this affects batch control charts, time stamps to not have to be equally spaced, or even sequential.

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



Project Name:Variable Control Chart (X-Bar R)
Part Name: Transmission Casing Bolt
Operator:J. Fenamore
Date: 7/04/2013
Machine: #11
Chart No.: 17
Part No.: 283501

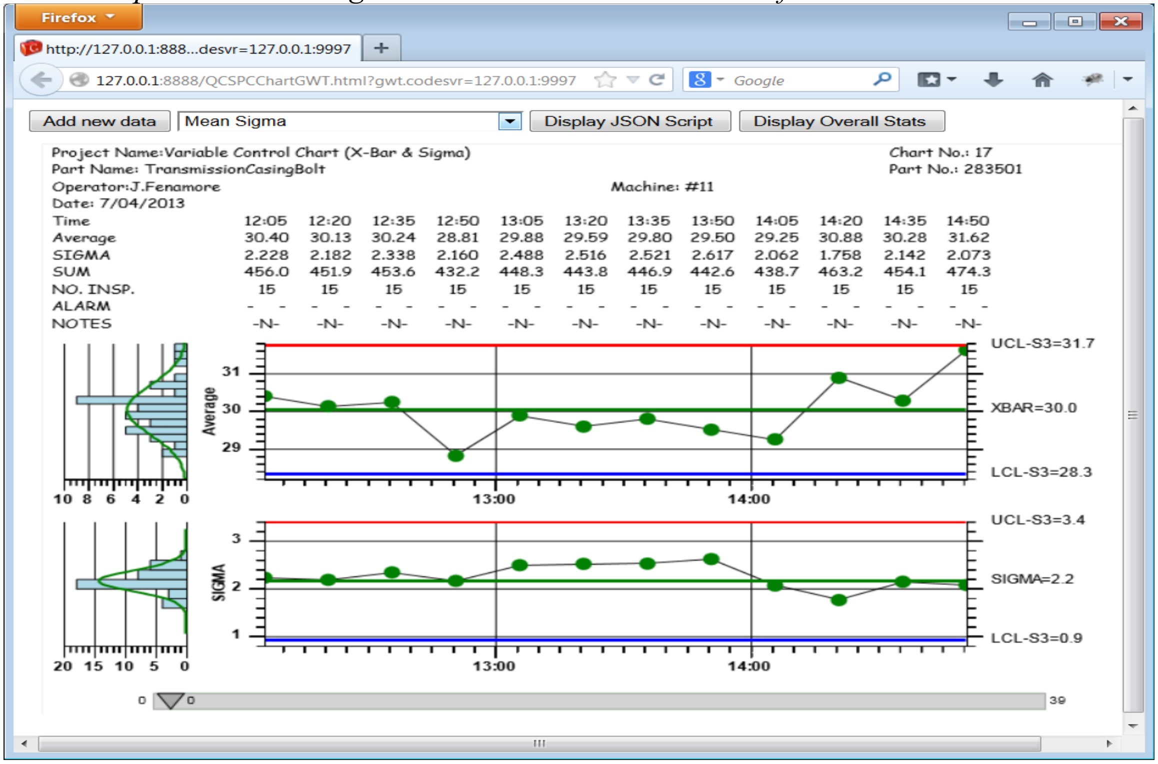
X-Bar R Chart – Also known as the Mean (or Average) and Range Chart

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



Firefox
http://127.0.0.1:888...desvr=127.0.0.1:9997
127.0.0.1:8888/QCSPCChartGWT.html?gwt.codesvr=127.0.0.1:9997
Google
Add new data
Mean Sigma
Display JSON Script
Display Overall Stats

Project Name:Variable Control Chart (X-Bar & Sigma)
Part Name: TransmissionCasingBolt
Operator:J.Fenamore
Date: 7/04/2013

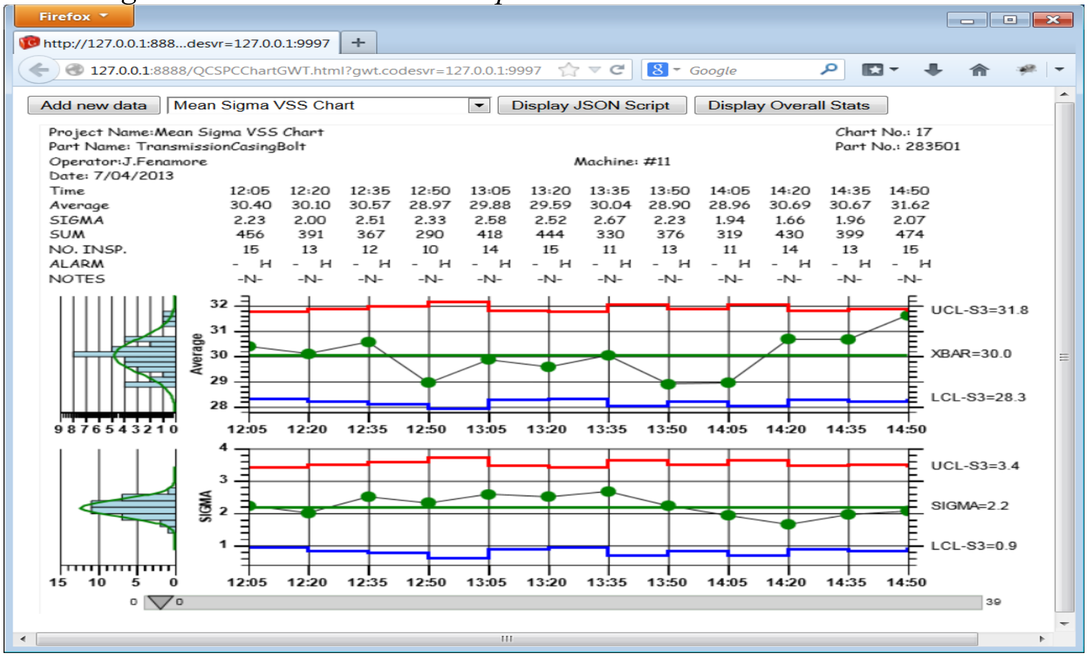
Chart No.: 17
Part No.: 283501

Machine: #11

Time	12:05	12:20	12:35	12:50	13:05	13:20	13:35	13:50	14:05	14:20	14:35	14:50
Average	30.40	30.13	30.24	28.81	29.88	29.59	29.80	29.50	29.25	30.88	30.28	31.62
SIGMA	2.228	2.182	2.338	2.160	2.488	2.516	2.521	2.617	2.062	1.758	2.142	2.073
SUM	456.0	451.9	453.6	432.2	448.3	443.8	446.9	442.6	438.7	463.2	454.1	474.3
NO. INSP.	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-

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



Firefox
http://127.0.0.1:888...desvr=127.0.0.1:9997
127.0.0.1:8888/QCSPCChartGWT.html?gwt.codesvr=127.0.0.1:9997
Google
Add new data
Mean Sigma VSS Chart
Display JSON Script
Display Overall Stats

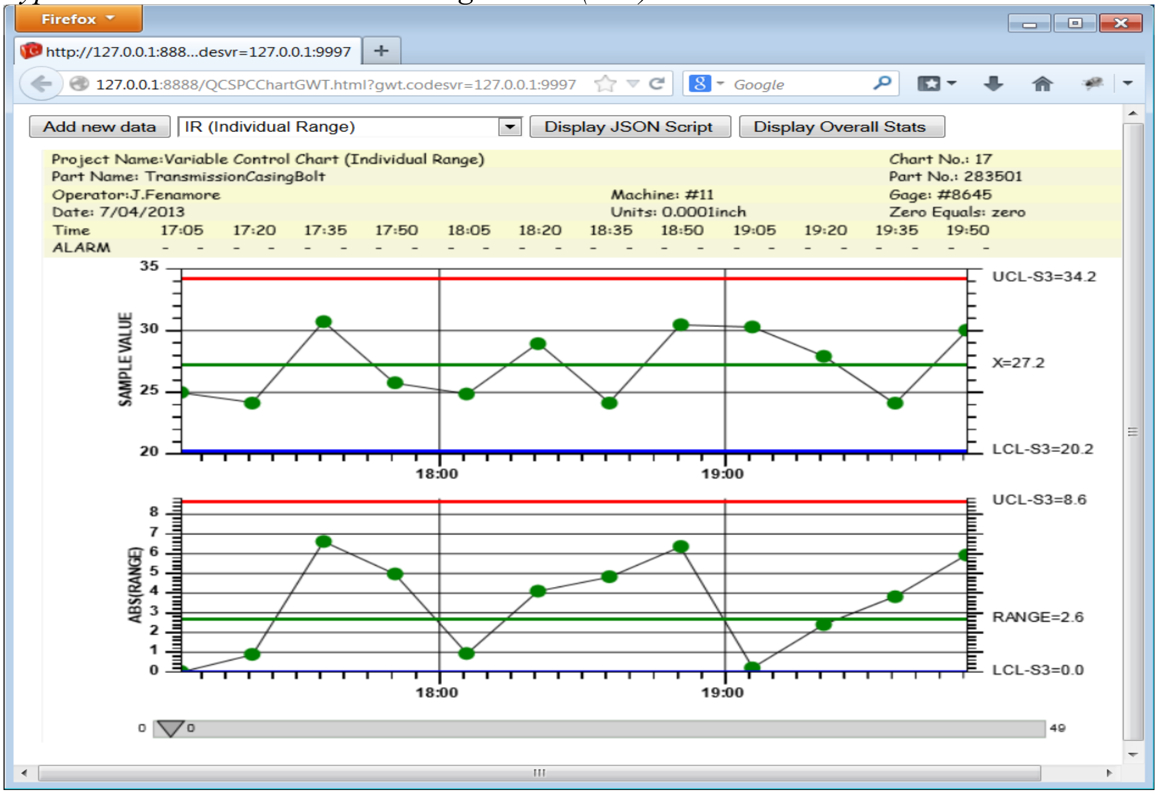
Project Name:Mean Sigma VSS Chart
Part Name: TransmissionCasingBolt
Operator:J.Fenamore
Date: 7/04/2013
Machine: #11
Chart No.: 17
Part No.: 283501

Time	12:05	12:20	12:35	12:50	13:05	13:20	13:35	13:50	14:05	14:20	14:35	14:50
Average	30.40	30.10	30.57	28.97	29.88	29.59	30.04	28.90	28.96	30.69	30.67	31.62
SIGMA	2.23	2.00	2.51	2.33	2.58	2.52	2.67	2.23	1.94	1.66	1.96	2.07
SUM	456	391	367	290	418	444	330	376	319	430	399	474
NO. INSP.	15	13	12	10	14	15	11	13	11	14	13	15
ALARM	- H	- H	- H	- H	- H	- H	- H	- H	- H	- H	- H	- H
NOTES	-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)



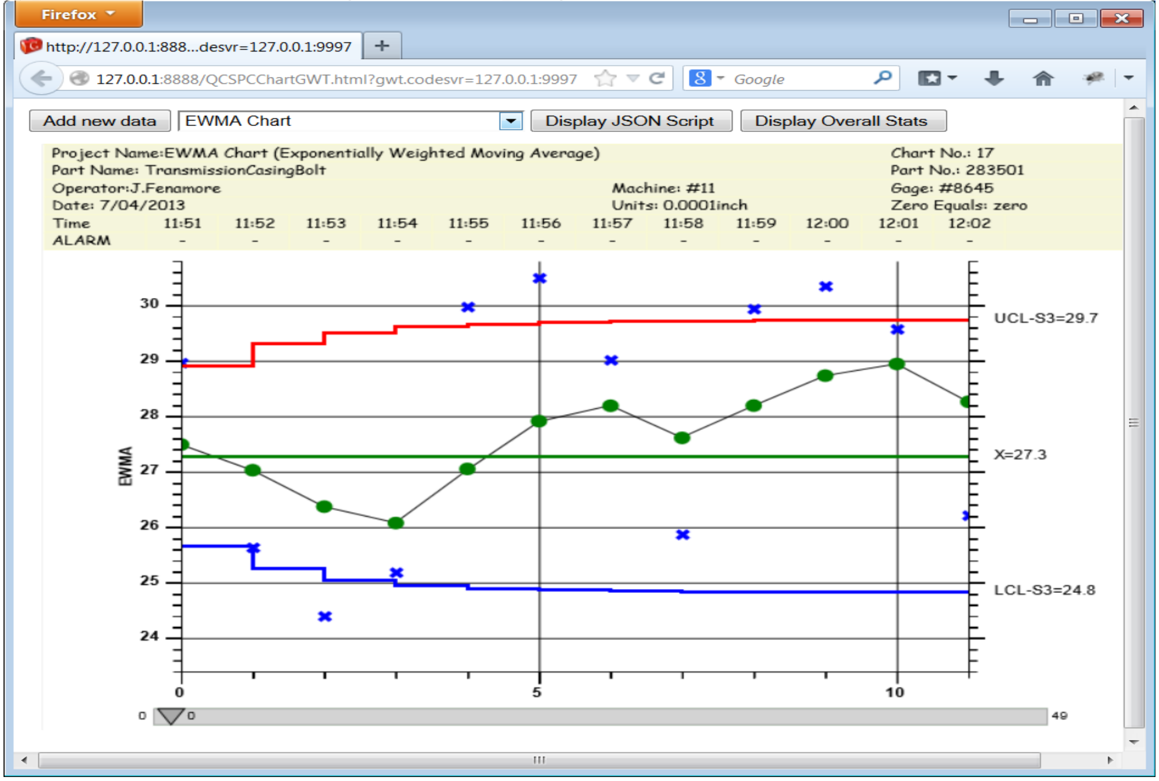
Firefox
http://127.0.0.1:888...desvr=127.0.0.1:9997
127.0.0.1:8888/QCSPCChartGWT.html?gwt.codesvr=127.0.0.1:9997
Google
Add new data
IR (Individual Range)
Display JSON Script
Display Overall Stats

Project Name:Variable Control Chart (Individual Range) Chart No.: 17
Part Name: TransmissionCasingBolt Part No.: 283501
Operator:J.Fenamore Machine: #11 Gage: #8645
Date: 7/04/2013 Units: 0.0001inch Zero Equals: zero
Time 17:05 17:20 17:35 17:50 18:05 18:20 18:35 18:50 19:05 19:20 19:35 19:50
ALARM -

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 EWMA Chart using Batch Sampling



Project Name:EWMA Chart (Exponentially Weighted Moving Average)
Part Name: TransmissionCasingBolt
Operator:J.Fenamore
Date: 7/04/2013
Machine: #11
Units: 0.0001inch
Chart No.: 17
Part No.: 283501
Gage: #8645
Zero Equals: zero

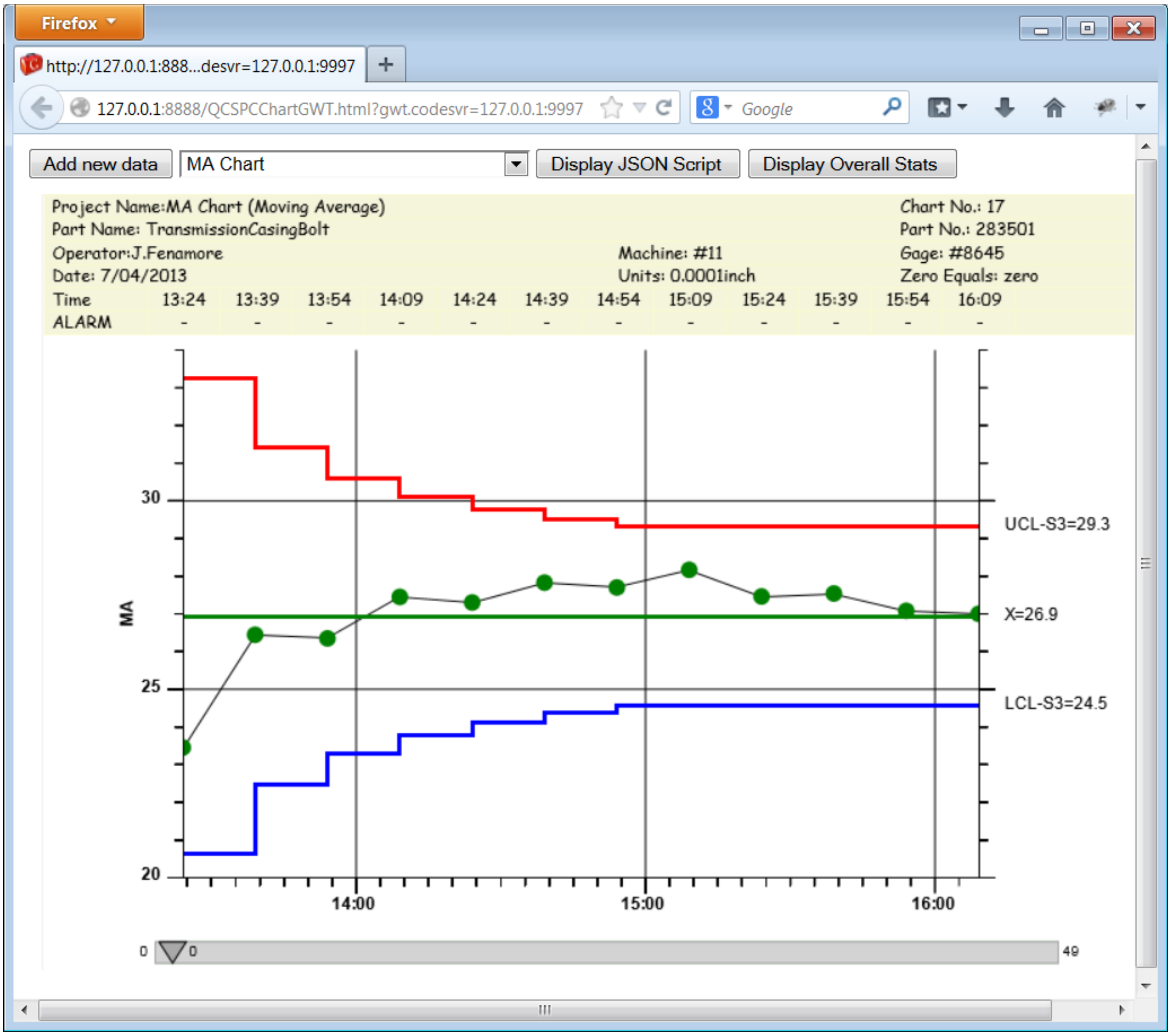
Time	11:51	11:52	11:53	11:54	11:55	11:56	11:57	11:58	11:59	12:00	12:01	12:02
ALARM	-	-	-	-	-	-	-	-	-	-	-	-

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 Chart – Moving Average

MA (Moving Average) Chart with Sample Values Plotted



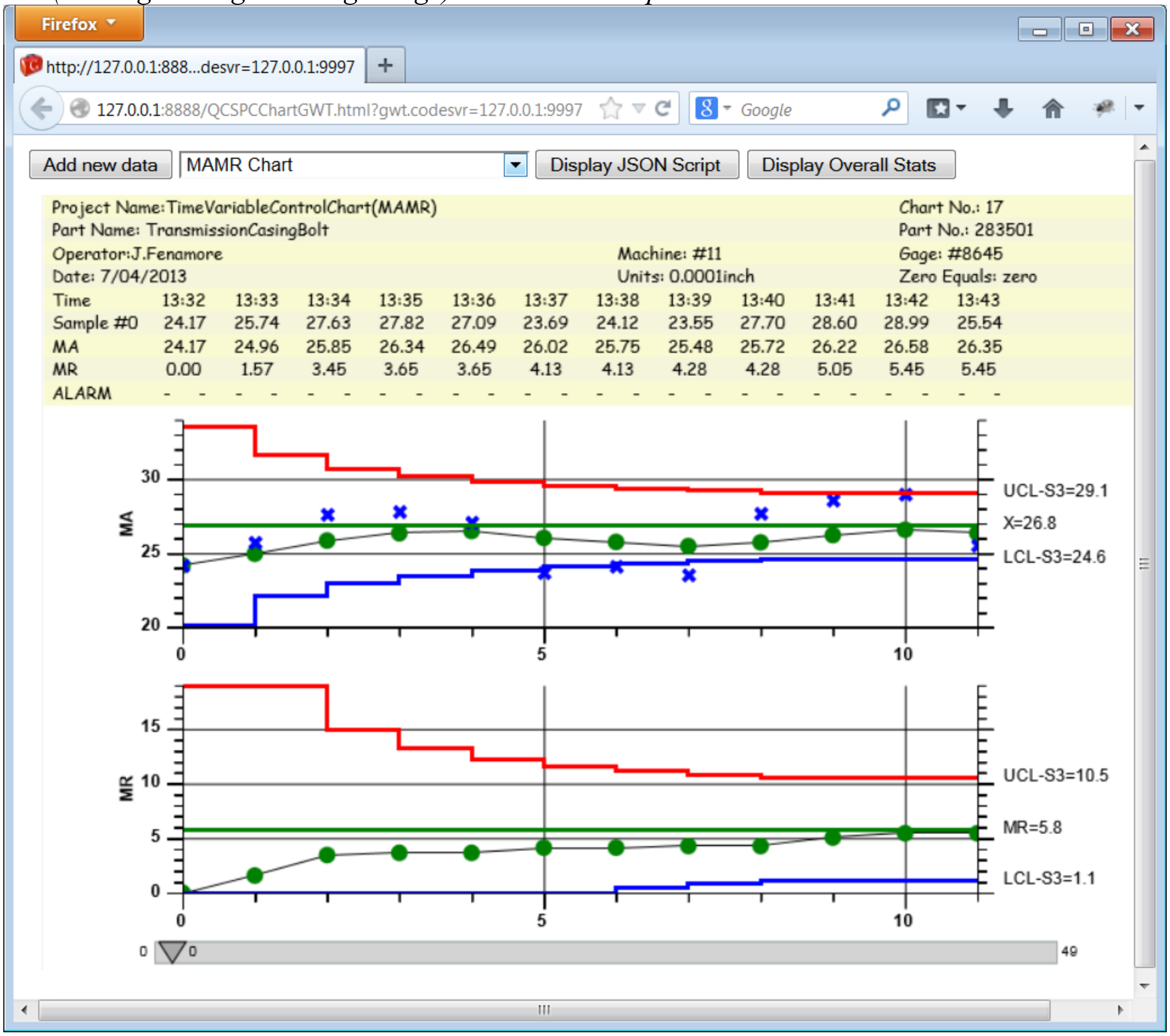
Project Name:MA Chart (Moving Average)
Part Name: TransmissionCasingBolt
Operator:J.Fenamore
Date: 7/04/2013
Machine: #11
Units: 0.0001inch
Chart No.: 17
Part No.: 283501
Gage: #8645
Zero Equals: zero

Time	13:24	13:39	13:54	14:09	14:24	14:39	14:54	15:09	15:24	15:39	15:54	16:09
ALARM	-	-	-	-	-	-	-	-	-	-	-	-

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



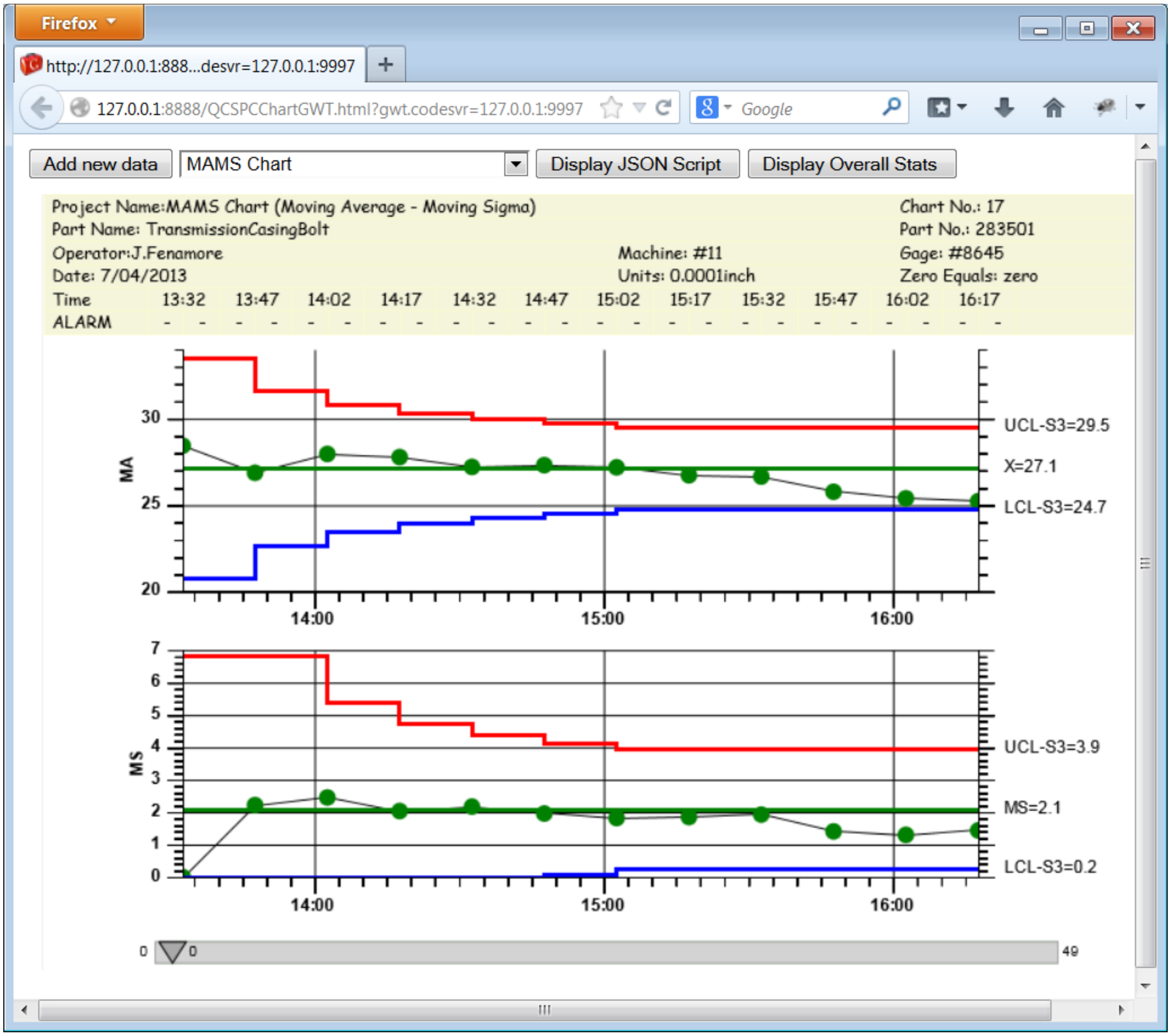
Project Name:TimeVariableControlChart(MAMR) Chart No.: 17
Part Name: TransmissionCasingBolt Part No.: 283501
Operator:J.Fenamore Machine: #11 Gage: #8645
Date: 7/04/2013 Units: 0.0001inch Zero Equals: zero

Time	13:32	13:33	13:34	13:35	13:36	13:37	13:38	13:39	13:40	13:41	13:42	13:43
Sample #0	24.17	25.74	27.63	27.82	27.09	23.69	24.12	23.55	27.70	28.60	28.99	25.54
MA	24.17	24.96	25.85	26.34	26.49	26.02	25.75	25.48	25.72	26.22	26.58	26.35
MR	0.00	1.57	3.45	3.65	3.65	4.13	4.13	4.28	4.28	5.05	5.45	5.45
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



Project Name:MAMS Chart (Moving Average - Moving Sigma) Chart No.: 17
Part Name: TransmissionCasingBolt Part No.: 283501
Operator:J.Fenamore Machine: #11 Gage: #8645
Date: 7/04/2013 Units: 0.0001inch Zero Equals: zero

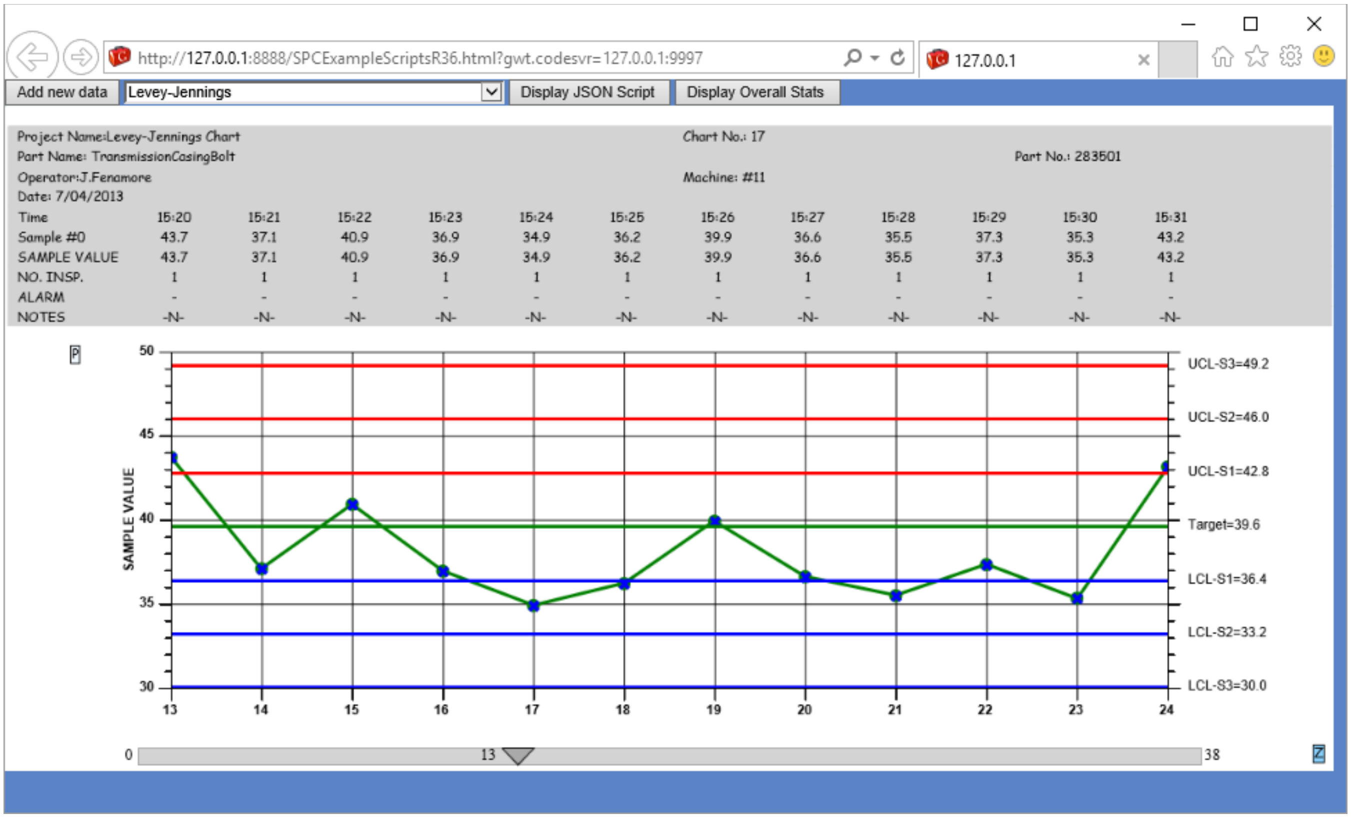
Time	13:32	13:47	14:02	14:17	14:32	14:47	15:02	15:17	15:32	15:47	16:02	16:17
ALARM	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -

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.

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

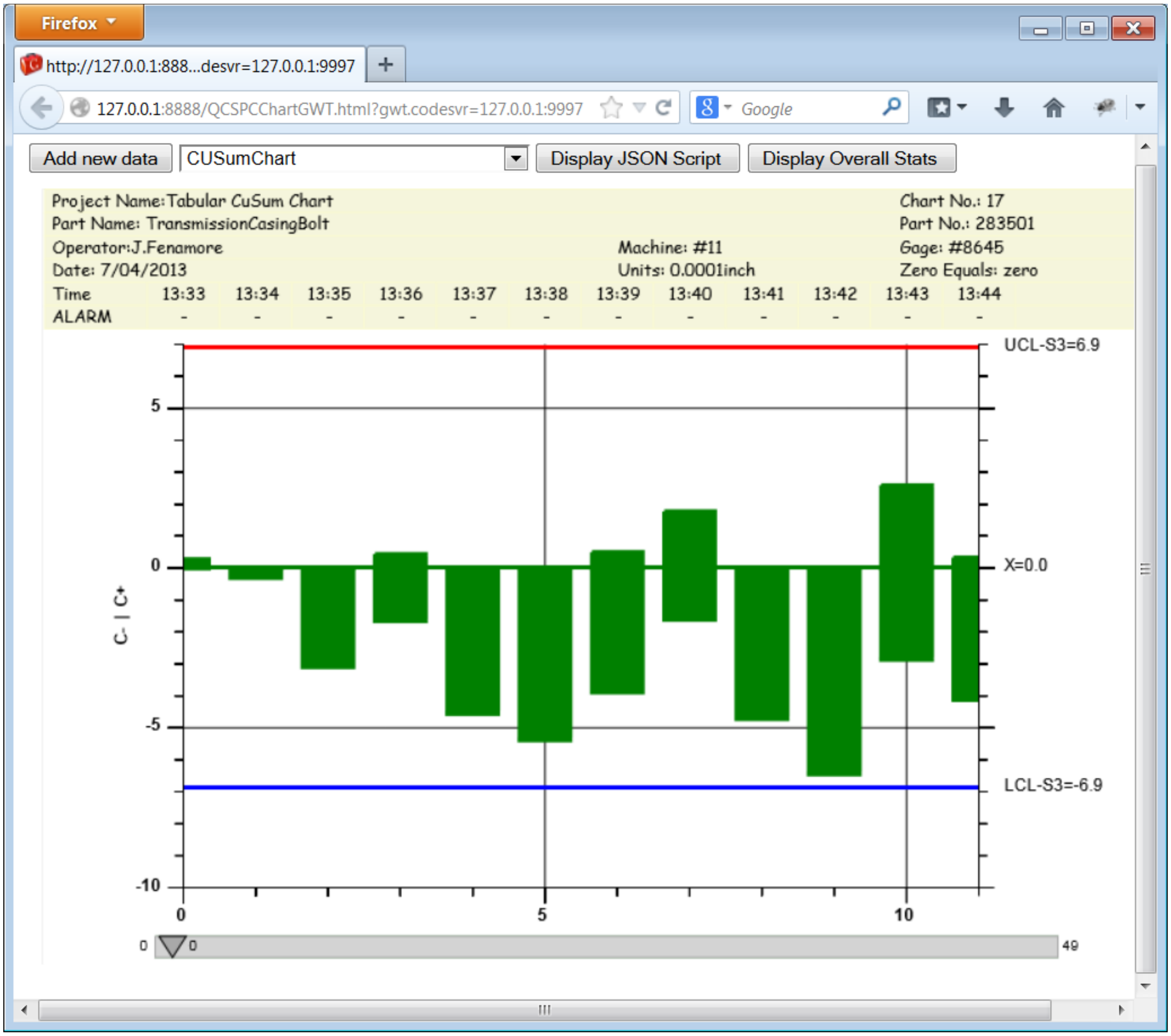


Project Name:Levey-Jennings Chart
Chart No.: 17
Part Name: TransmissionCasingBolt
Part No.: 283501
Operator:J.Fenamore
Machine: #11
Date: 7/04/2013

Time	15:20	15:21	15:22	15:23	15:24	15:25	15:26	15:27	15:28	15:29	15:30	15:31
Sample #0	43.7	37.1	40.9	36.9	34.9	36.2	39.9	36.6	35.5	37.3	35.3	43.2
SAMPLE VALUE	43.7	37.1	40.9	36.9	34.9	36.2	39.9	36.6	35.5	37.3	35.3	43.2
NO. INSP.	1	1	1	1	1	1	1	1	1	1	1	1
ALARM	-	-	-	-	-	-	-	-	-	-	-	-
NOTES	-N-	-N-	-N-	-N-	-N-	-N-	-N-	-N-	-N-	-N-	-N-	-N-

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.

Tabular CuSum Chart



Project Name:Tabular CuSum Chart
Part Name: TransmissionCasingBolt
Operator:J.Fenamore
Date: 7/04/2013
Machine: #11
Units: 0.0001inch
Chart No.: 17
Part No.: 283501
Gage: #8645
Zero Equals: zero

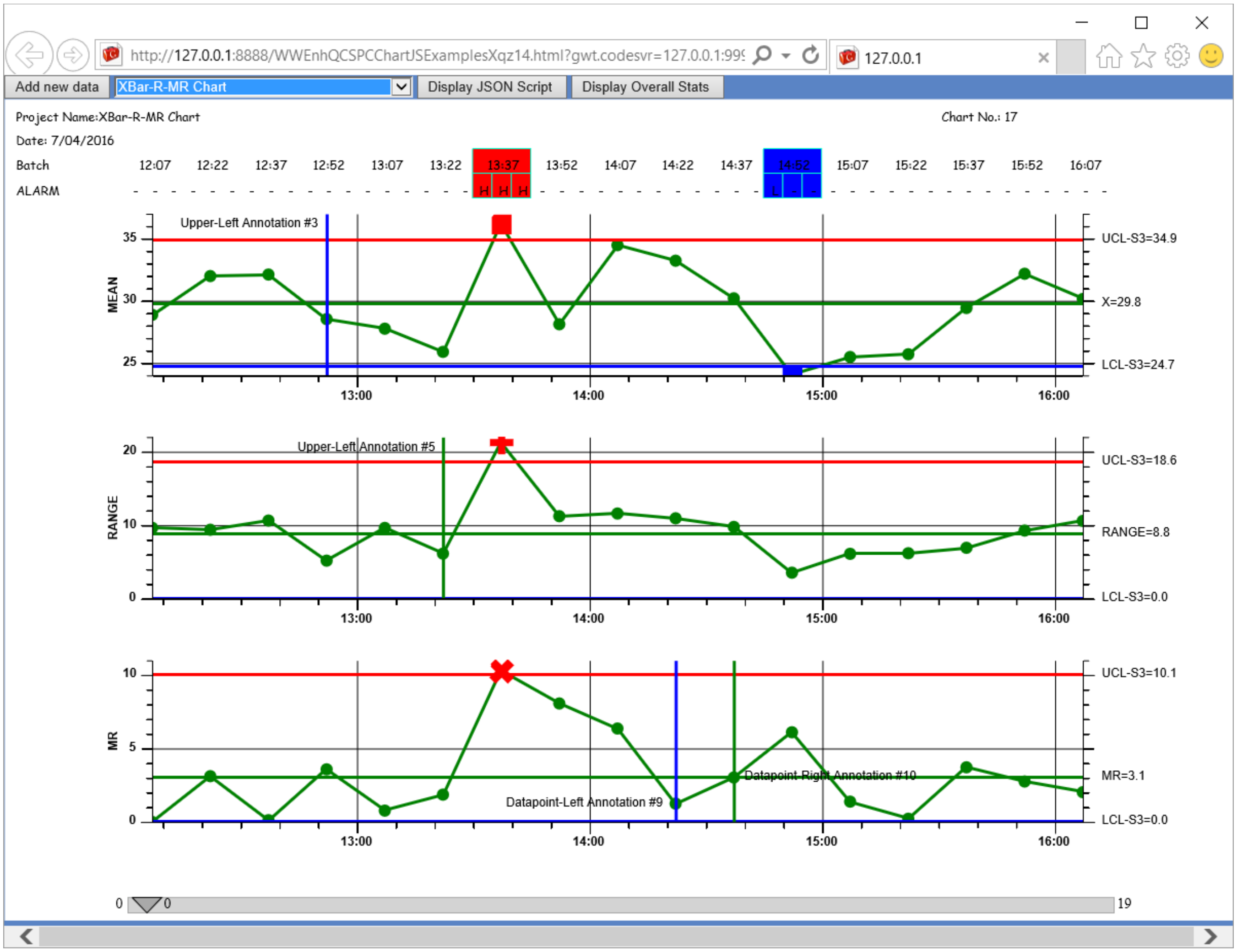
Time	13:33	13:34	13:35	13:36	13:37	13:38	13:39	13:40	13:41	13:42	13:43	13:44
ALARM	-	-	-	-	-	-	-	-	-	-	-	-

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.

XBar-R-MR Chart – Mean-Range-Moving Average Chart

XBar-R-MR Chart



Project Name:XBar-R-MR Chart

Date: 7/04/2016

Chart No.: 17

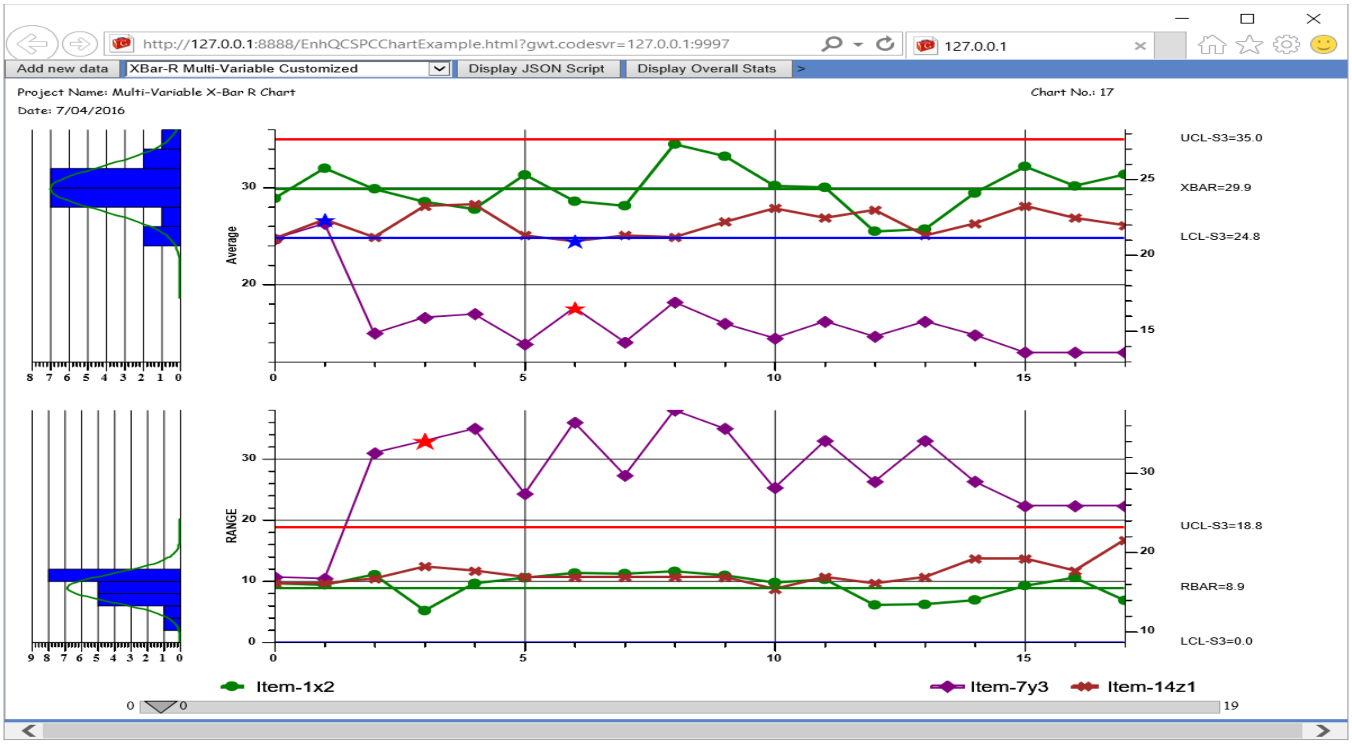
The Xbar-R-MR (Mean-Range-Moving Range) chart is a specialized control chart added to the QCSPCChart+ version of the software. It adds a third chart to the usual two of the Xbar-R chart. The third chart displays the moving range of the sample interval means, i.e. MR(N) = Mean(N) – Mean(N-1).

Multi-Variable Control Chart

Multi-Variable Variable Control Charts are new class of charts added to the QCSPCChart+ version of the software. A Multi-Variable chart, has one main SPC chart plot, plotted as the main plot in both the Primary and Secondary chart windows. The main SPCChart plot will be processed exactly like the single channel variants (Xbar-R, Xbar-Sigma, I-R, others), with control limit checking performed against establish limits. But, there can also be up to 5 additional sub-plots, overlaid on top of the main

SPC Chart Plot. Since these additional sub-plots probably do not have exactly the same control limits as the main plot, no limit checking is performed on the sub-plots. If the programmer wants to indicate that a sub-plot sample interval is out of control, they can do that using alarm limit forcing, described in Chapter 8.

Multi-Variable Xbar-R chart



Project Name: Multi-Variable X-Bar R Chart

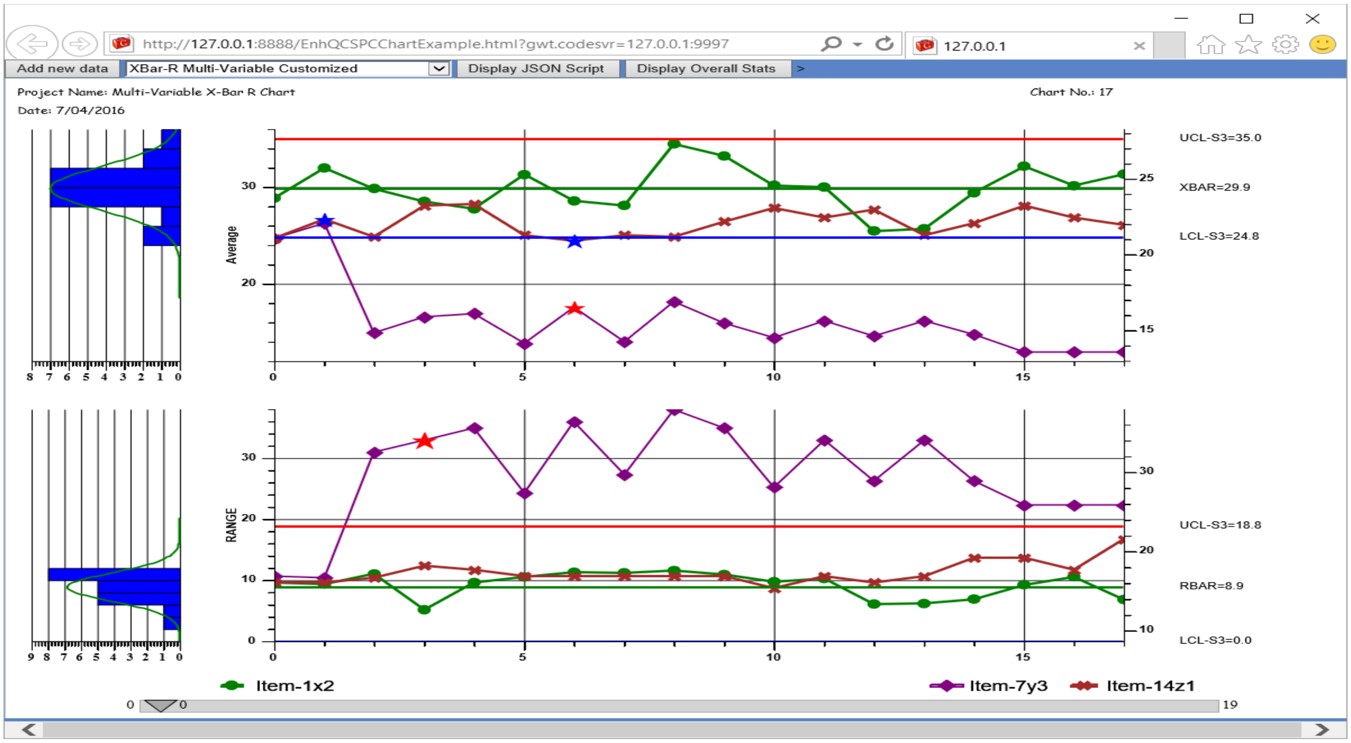
Date: 7/04/2016

Chart No.: 17

The example above is a Multi-Variable Xbar-R chart, with the main plot in green, and two sub plots. Note, not only can the color of a data point changed, but also the symbol (to a STAR), as the result of alarm forcing.

Multi-Variable X-Bar R Chart

Multi-Variable Xbar-R



Project Name: Multi-Variable X-Bar R Chart

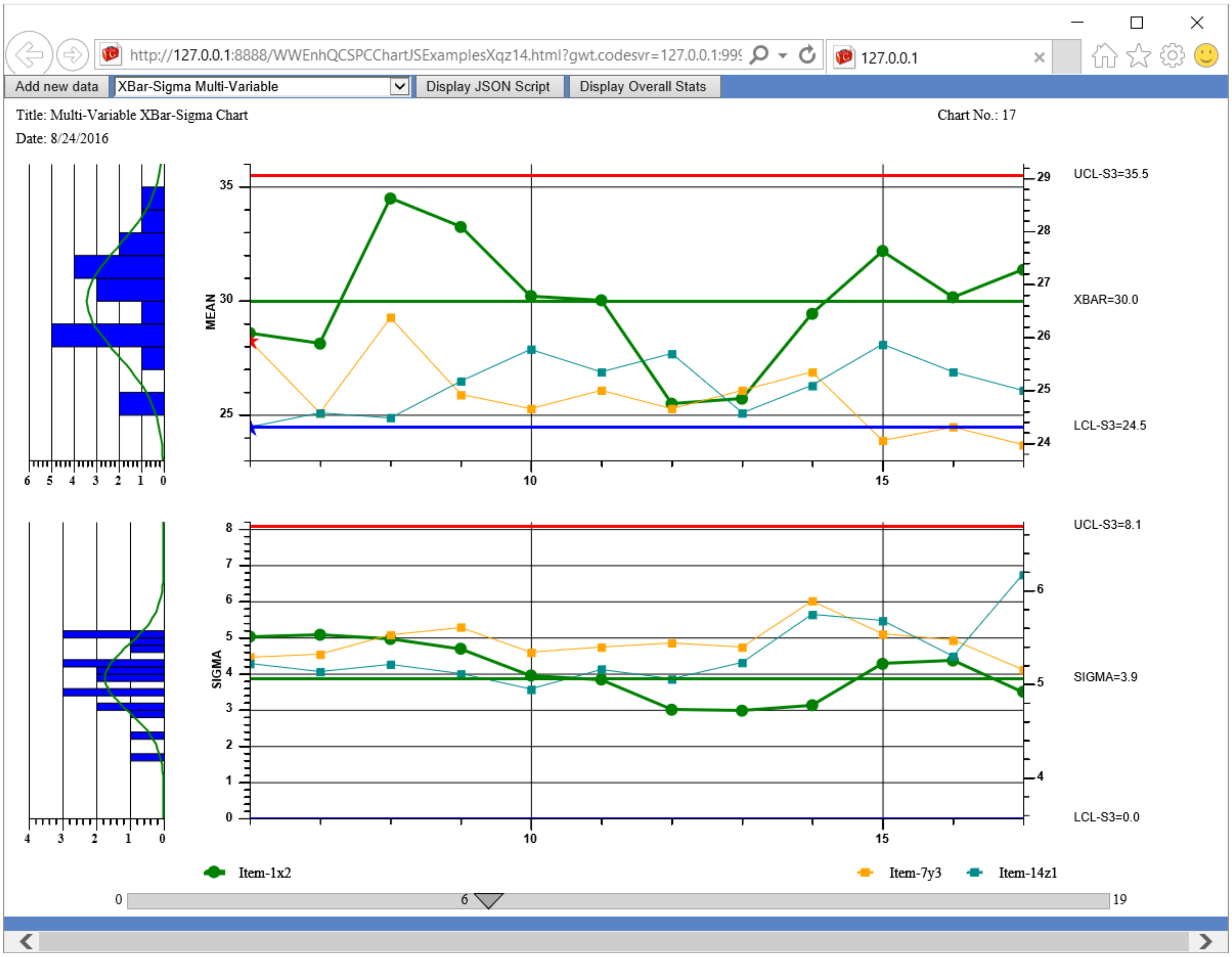
Date: 7/04/2016

Chart No.: 17

The Multi-Variable Xbar-R chart can combine the data from several processes, into a single SPC Chart. The Xbar data from multiple process can be overlaid in the Primary Chart, and the Range data can be overlaid in the Secondary Chart. The chart control limits are applied against the main plot in the chart, but not against the sub-plots, of which there can be up to five. Since the processes under observation may not have identical dynamic ranges, two coordinate systems are supported, using independent left and right y-axis. The main plot always references the left y-axis, while the sub-plots can reference either the left or the right y-axis. Both left and right-axes auto-scale to the data plotted in their respective coordinate system. A legend under each y-axis identifies which plots are assigned to each axis. The histograms on the left only apply to the main plot of each chart.

Multi-Variable Xbar-Sigma Chart

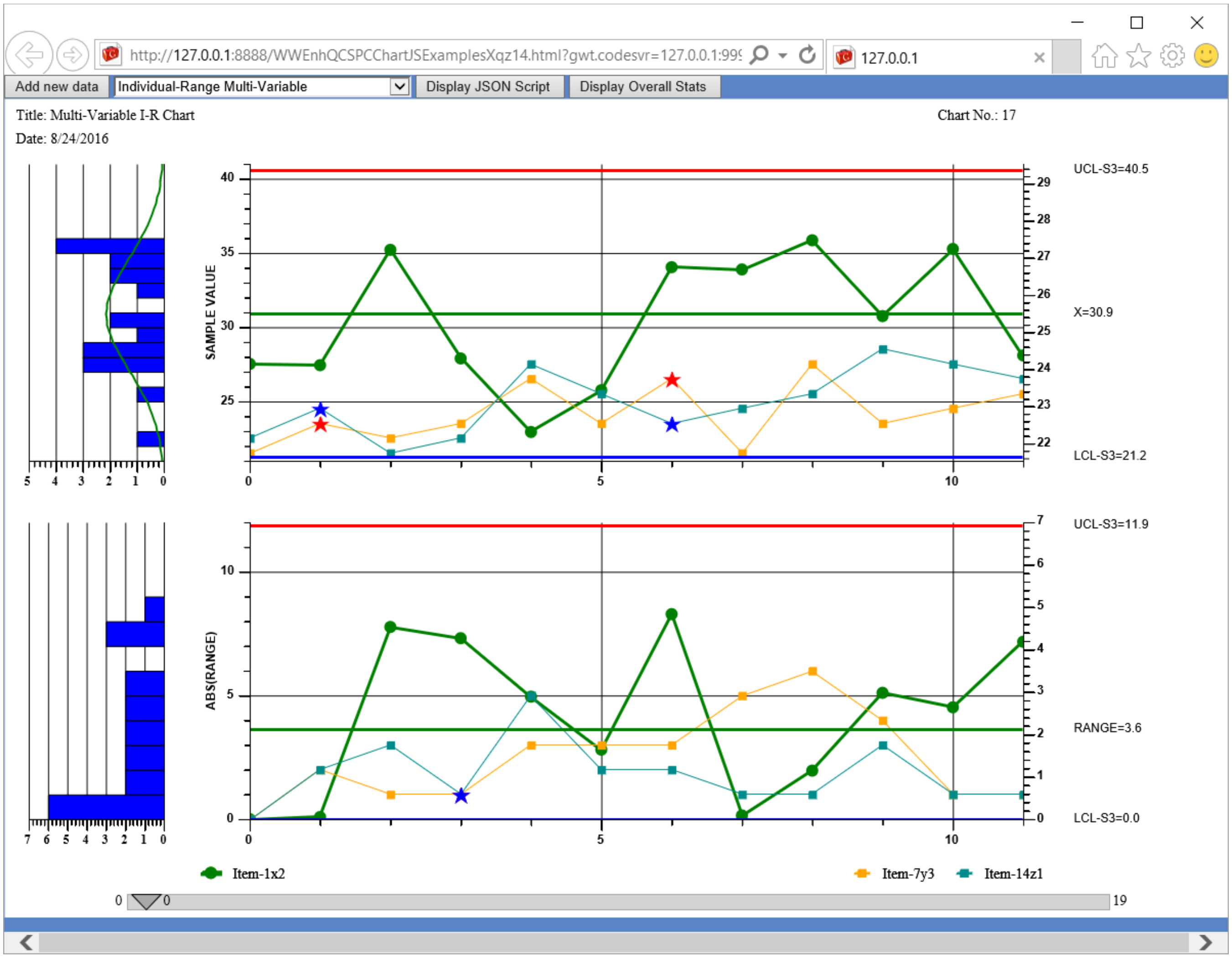
Multi-Variable Xbar-Sigma chart



The Multi-Variable Xbar-Sigma chart is the same as the Multi-Variable Xbar-R plot, only the Secondary chart plots the sigma values of the each variables sample interval, rather than the Range values.

Multi-Variable I-R (X-R, Individual-Range) Chart

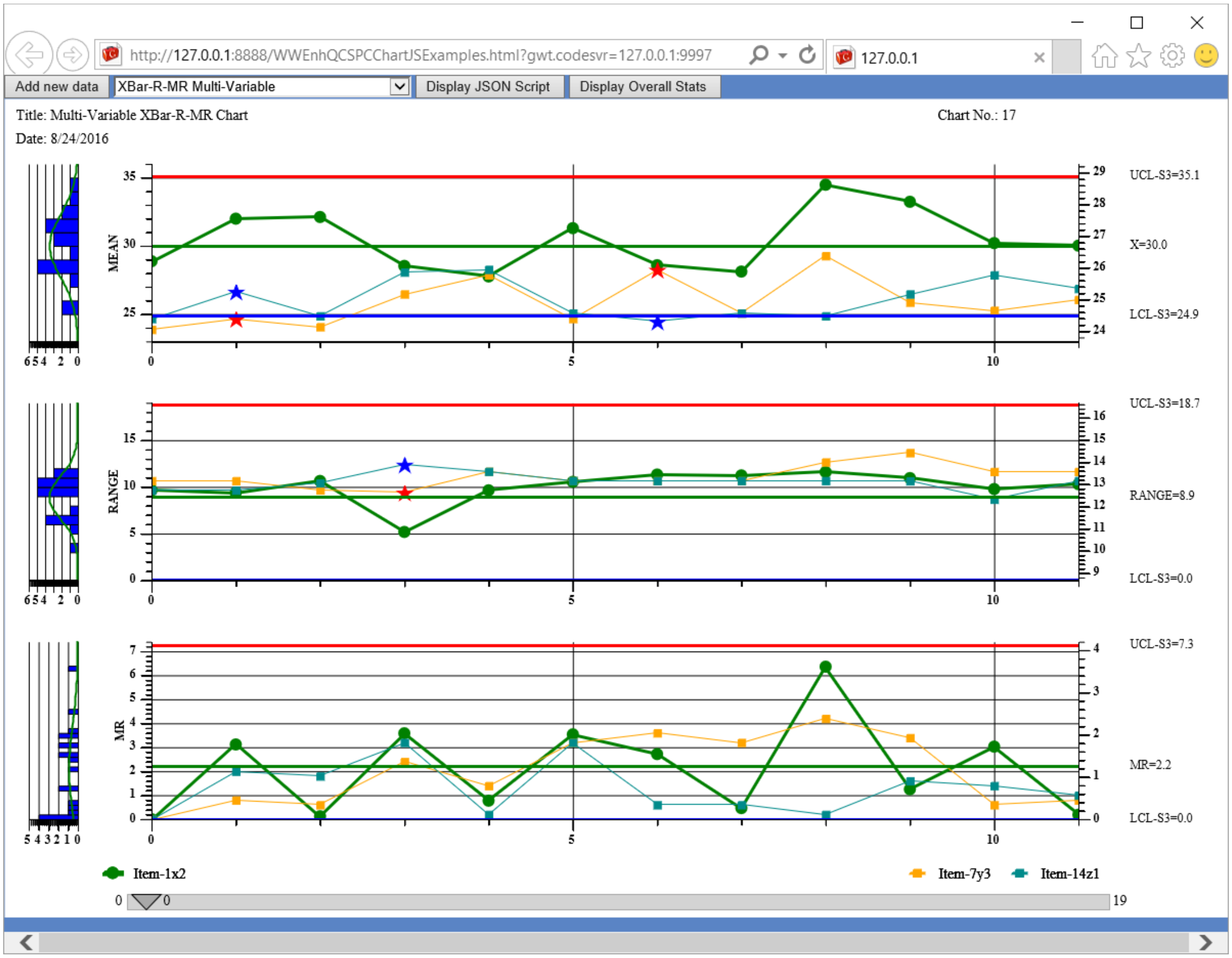
Multi-Variable I-R chart



The Multi-Variable I-R chart is the same as the Multi-Variable Xbar-R plot, except that each variables sample data only contains one sample. The single value for each variable is plotted as a trace in the Primary Chart. The Secondary Chart plots the moving range of the single sample values plotted in the Primary Chart, i.e. Range(N) = Sample(N) – Sample (N-1).

Multi-Variable Xbar-R-MR (Mean-Range-Moving Range) Chart

Multi-Variable XBar-R-MR chart



http://127.0.0.1:8888/WWEnhQCSPCChartJSExamples.html?gwt.codesvr=127.0.0.1:9997

127.0.0.1

Add new data | XBar-R-MR Multi-Variable | Display JSON Script | Display Overall Stats

Title: Multi-Variable XBar-R-MR Chart

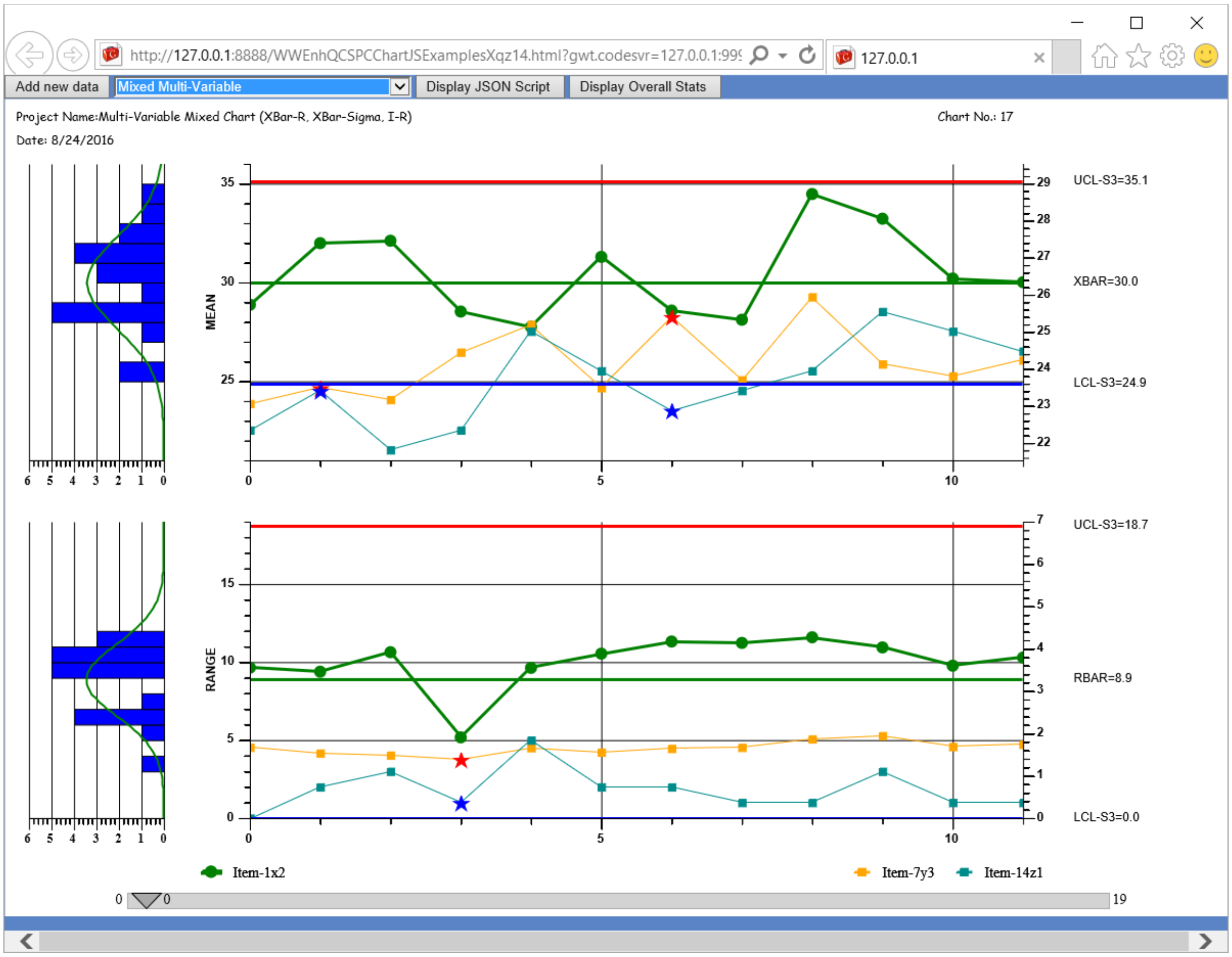
Date: 8/24/2016

Chart No.: 17

The Multi-Variable Xbar-R-MR chart is based on the new Xbar-R-MR chart type which has three charts, rather than the usual two. The third chart displays the moving range of the Primary Chart sample interval means, i.e. MR(N) = Mean(N) – Mean(N-1).

Multi-Variable Mixed Chart

Multi-Variable Mixed chart



http://127.0.0.1:8888/WWEnhQCSPCChartJSExamplesXqz14.html?gwt.codesvr=127.0.0.1:99

Project Name:Multi-Variable Mixed Chart (XBar-R, XBar-Sigma, I-R)

Chart No.: 17

Date: 8/24/2016

The Multi-VariableMixed chart is will combine charts of different types. You can combine Xbar-R, Xbar-Sigma and I-R data into the same chart.

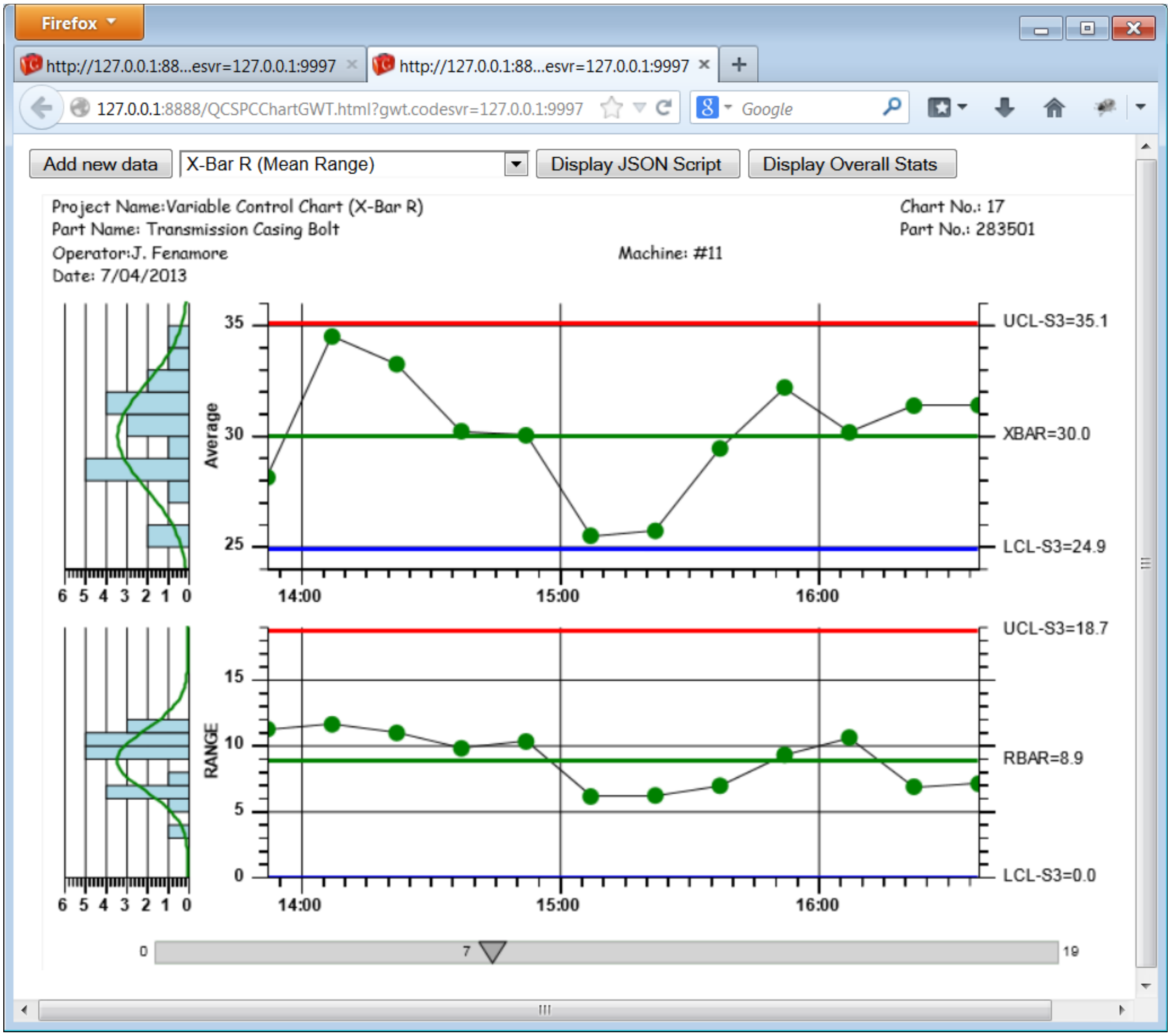
Measured Data and Calculated Value Tables

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

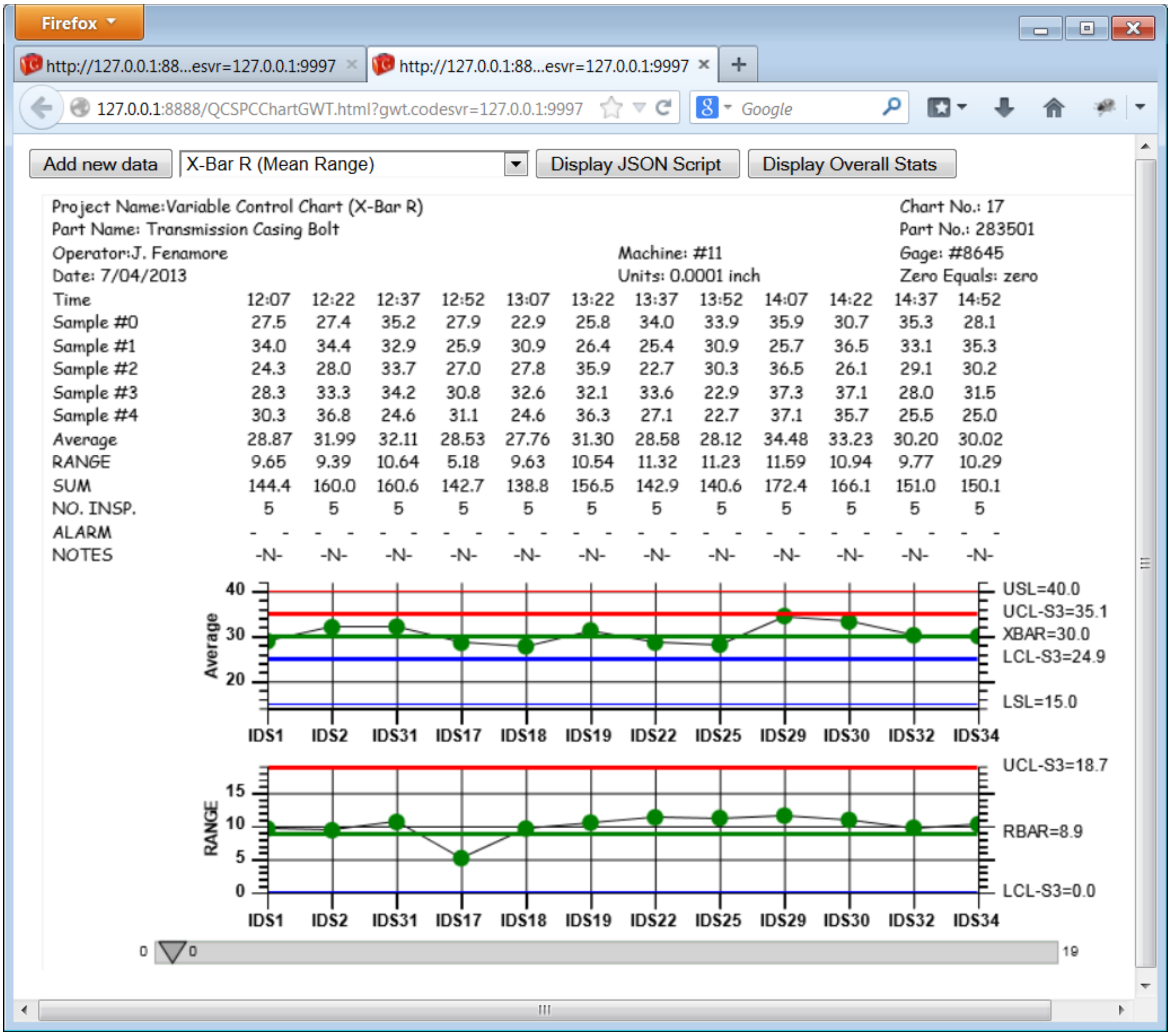


Project Name:Variable Control Chart (X-Bar R)
Part Name: Transmission Casing Bolt
Operator:J. Fenamore
Date: 7/04/2013

Machine: #11

Chart No.: 17
Part No.: 283501

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



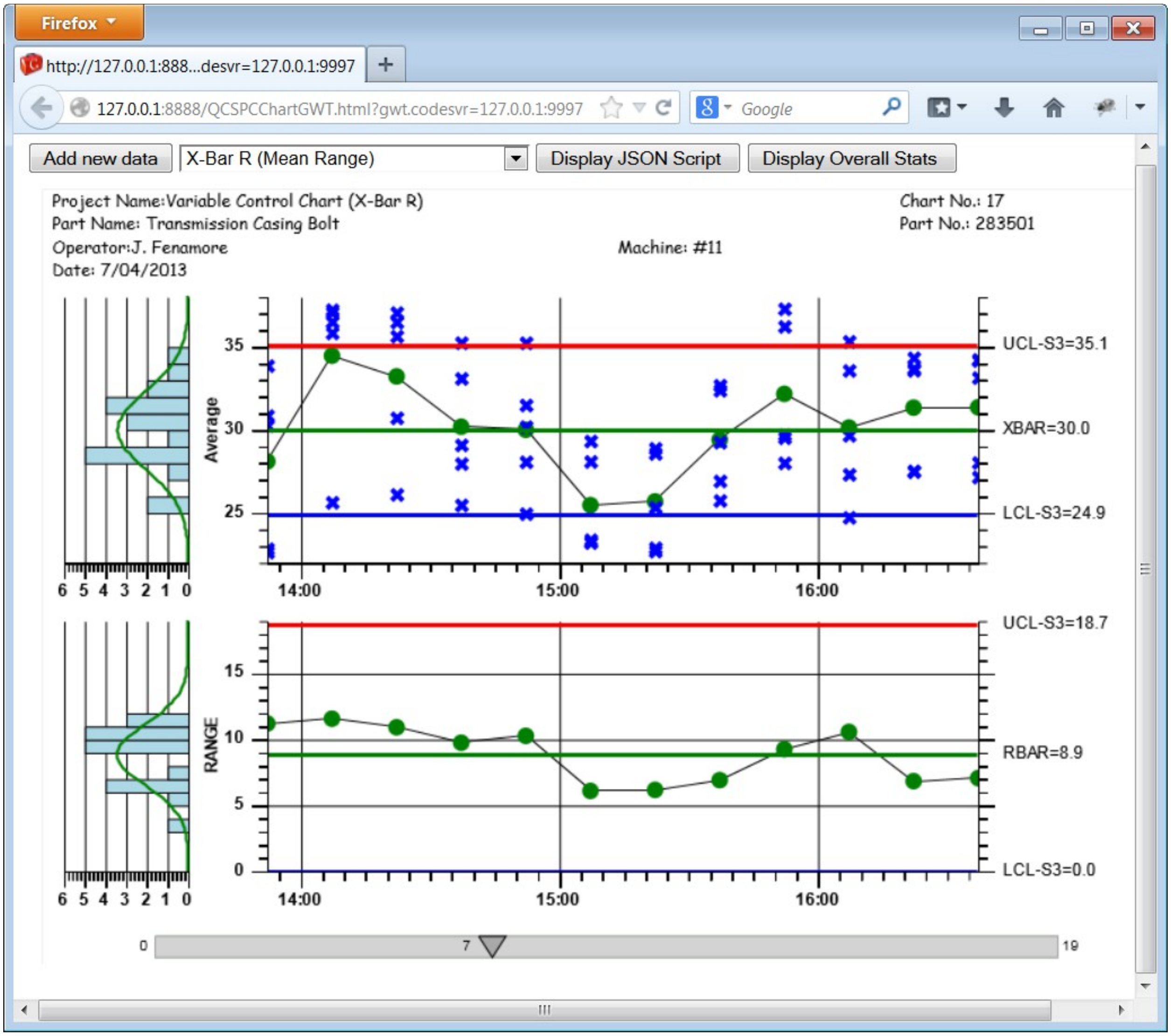
Project Name:Variable Control Chart (X-Bar R) Chart No.: 17
Part Name: Transmission Casing Bolt Part No.: 283501
Operator:J. Fenamore Machine: #11 Gage: #8645
Date: 7/04/2013 Units: 0.0001 inch Zero Equals: zero

Time	12:07	12:22	12:37	12:52	13:07	13:22	13:37	13:52	14:07	14:22	14:37	14:52
Sample #0	27.5	27.4	35.2	27.9	22.9	25.8	34.0	33.9	35.9	30.7	35.3	28.1
Sample #1	34.0	34.4	32.9	25.9	30.9	26.4	25.4	30.9	25.7	36.5	33.1	35.3
Sample #2	24.3	28.0	33.7	27.0	27.8	35.9	22.7	30.3	36.5	26.1	29.1	30.2
Sample #3	28.3	33.3	34.2	30.8	32.6	32.1	33.6	22.9	37.3	37.1	28.0	31.5
Sample #4	30.3	36.8	24.6	31.1	24.6	36.3	27.1	22.7	37.1	35.7	25.5	25.0
Average	28.87	31.99	32.11	28.53	27.76	31.30	28.58	28.12	34.48	33.23	30.20	30.02
RANGE	9.65	9.39	10.64	5.18	9.63	10.54	11.32	11.23	11.59	10.94	9.77	10.29
SUM	144.4	160.0	160.6	142.7	138.8	156.5	142.9	140.6	172.4	166.1	151.0	150.1
NO. INSP.	5	5	5	5	5	5	5	5	5	5	5	5
ALARM	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -
NOTES	-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



Project Name:Variable Control Chart (X-Bar R)
Part Name: Transmission Casing Bolt
Operator:J. Fenamore
Date: 7/04/2013

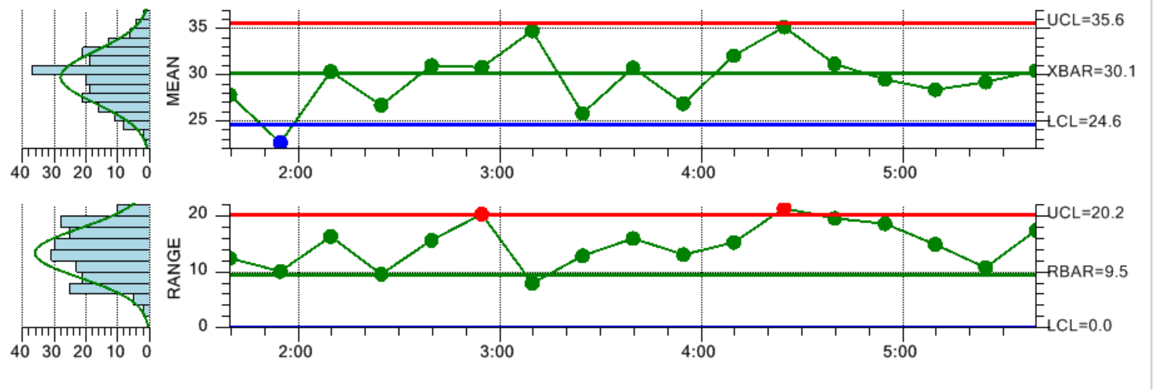
Machine: #11

Chart No.: 17
Part No.: 283501

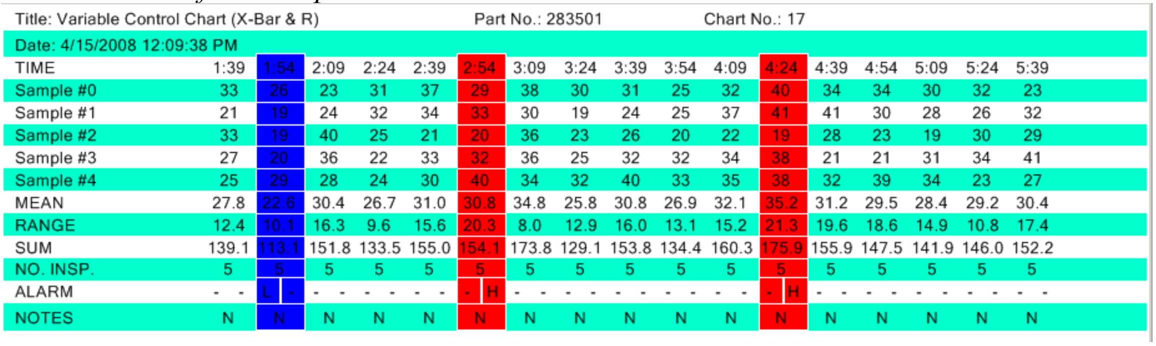
Alarm Notification

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

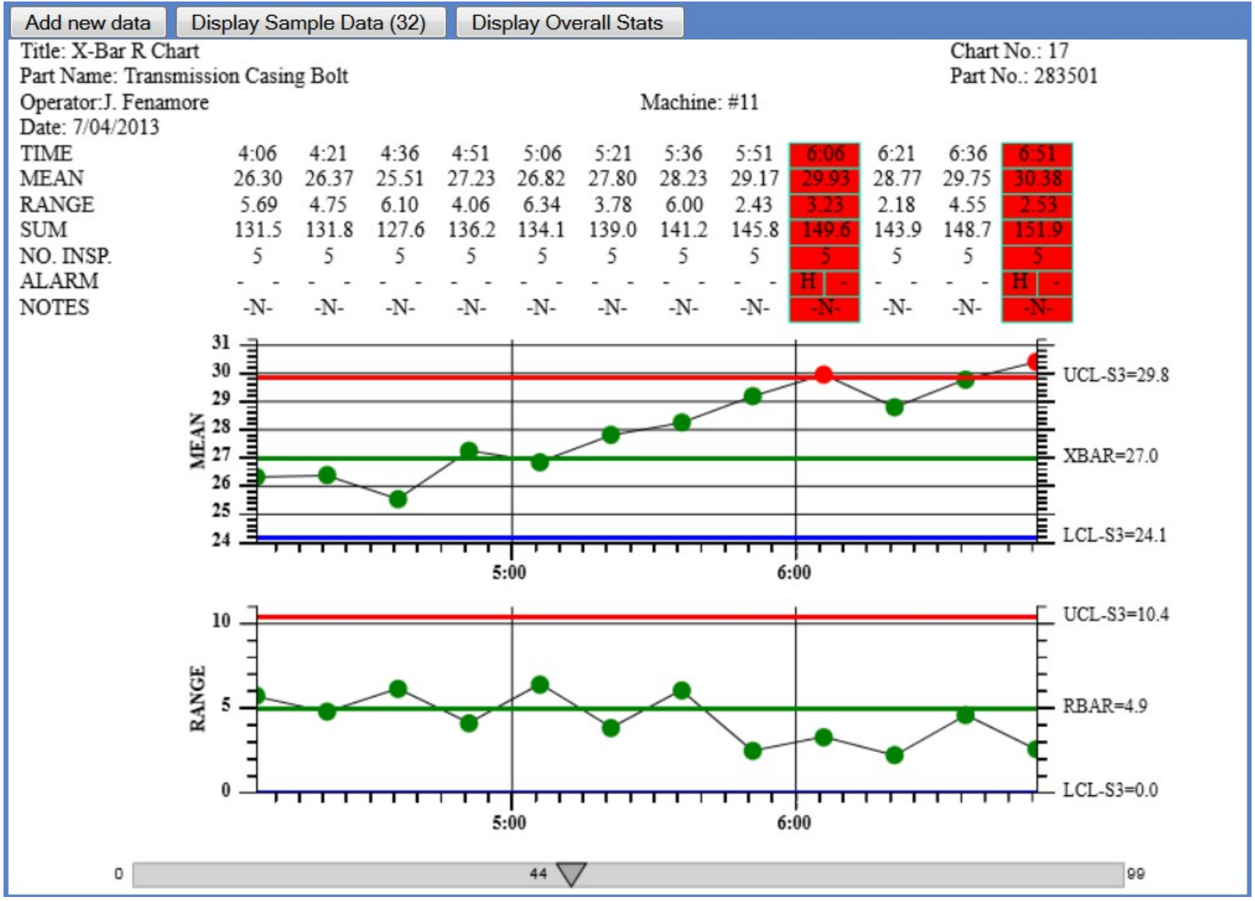


Title: Variable Control Chart (X-Bar & R) Part No.: 283501 Chart No.: 17

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: X-Bar R Chart
Chart No.: 17
Part Name: Transmission Casing Bolt
Part No.: 283501
Operator:J. Fenamore
Machine: #11
Date: 7/04/2013

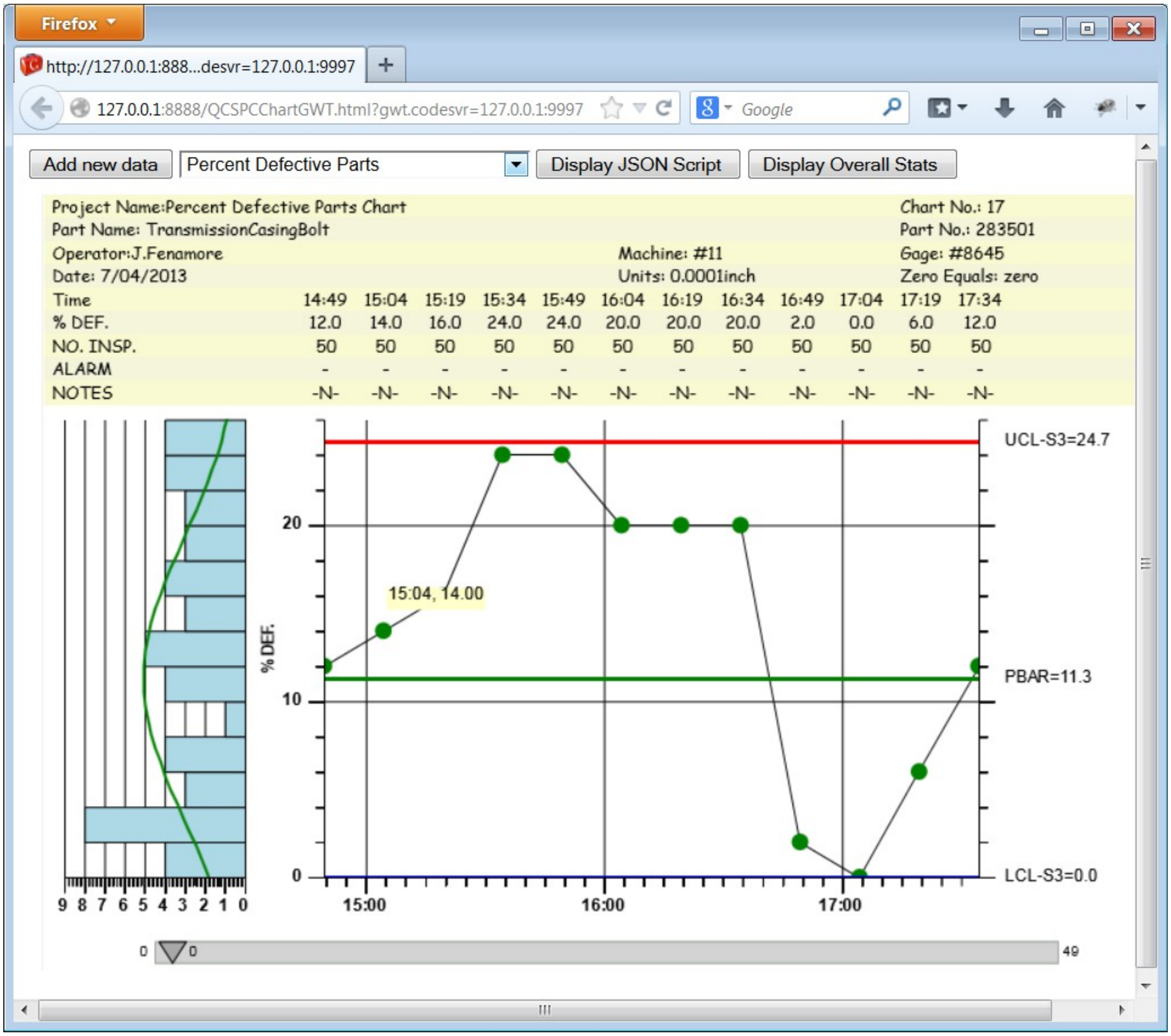
TIME	4:06	4:21	4:36	4:51	5:06	5:21	5:36	5:51	6:06	6:21	6:36	6:51
MEAN	26.30	26.37	25.51	27.23	26.82	27.80	28.23	29.17	29.93	28.77	29.75	30.38
RANGE	5.69	4.75	6.10	4.06	6.34	3.78	6.00	2.43	3.23	2.18	4.55	2.53
SUM	131.5	131.8	127.6	136.2	134.1	139.0	141.2	145.8	149.6	143.9	148.7	151.9
NO. INSP.	5	5	5	5	5	5	5	5	5	5	5	5
ALARM	- -	- -	- -	- -	- -	- -	- -	- -	H -	- -	- -	H -
NOTES	-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.

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)



Project Name:Percent Defective Parts Chart
Part Name: TransmissionCasingBolt
Operator:J.Fenamore
Date: 7/04/2013
Machine: #11
Units: 0.0001inch
Chart No.: 17
Part No.: 283501
Gage: #8645
Zero Equals: zero

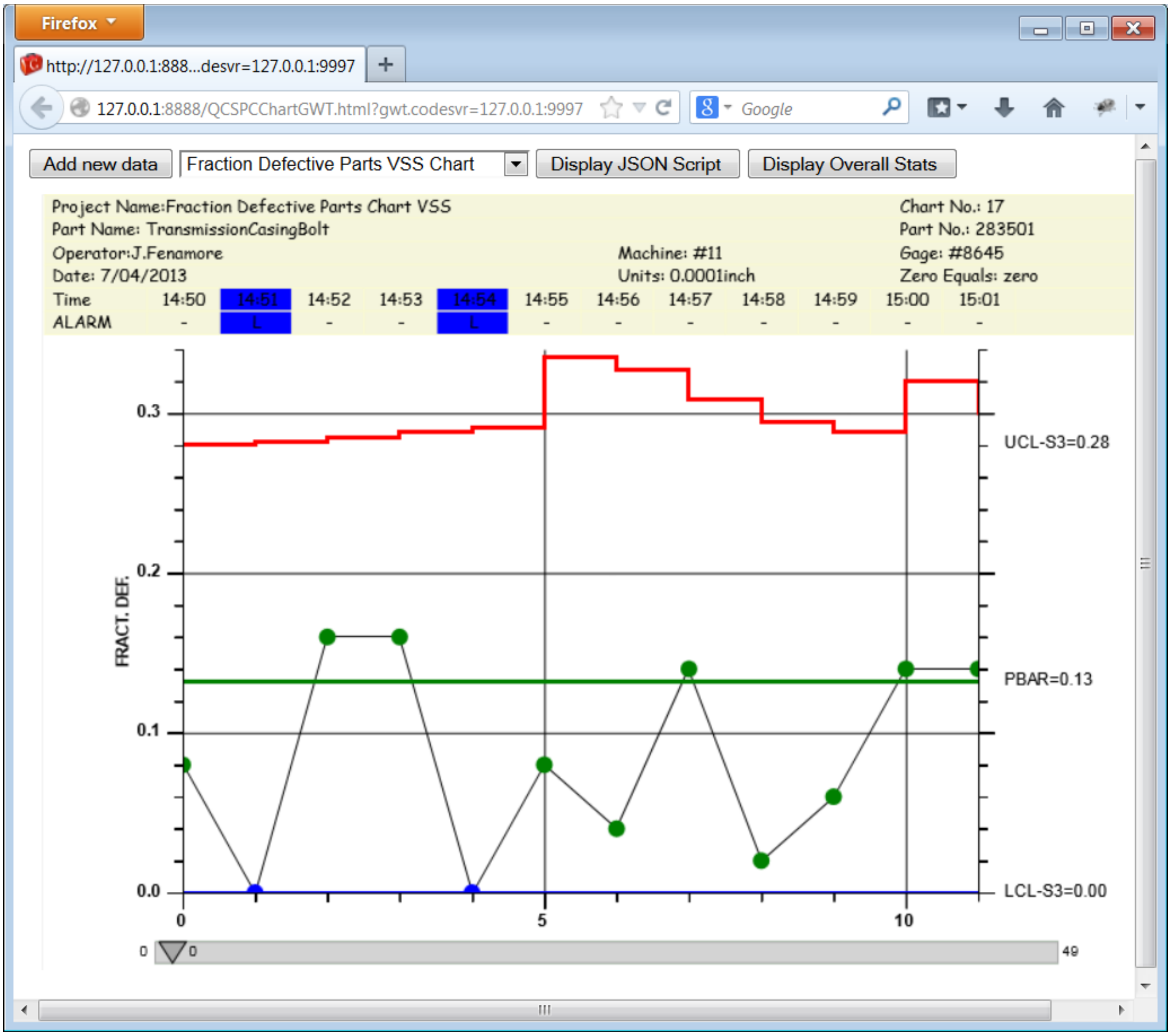
Time	14:49	15:04	15:19	15:34	15:49	16:04	16:19	16:34	16:49	17:04	17:19	17:34
% DEF.	12.0	14.0	16.0	24.0	24.0	20.0	20.0	20.0	2.0	0.0	6.0	12.0
NO. INSP.	50	50	50	50	50	50	50	50	50	50	50	50
ALARM	-	-	-	-	-	-	-	-	-	-	-	-
NOTES	-N-	-N-	-N-	-N-	-N-	-N-	-N-	-N-	-N-	-N-	-N-	-N-

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


Firefox
http://127.0.0.1:888...desvr=127.0.0.1:9997
127.0.0.1:8888/QCSPCChartGWT.html?gwt.codesvr=127.0.0.1:9997
Google
Add new data
Fraction Defective Parts VSS Chart
Display JSON Script
Display Overall Stats
0.3
0.2
0.1
0.0
FRACT. DEF.
UCL-S3=0.28
PBAR=0.13
LCL-S3=0.00
0
5
10
0
0
49

Project Name:Fraction Defective Parts Chart VSS
Part Name: TransmissionCasingBolt
Operator:J.Fenamore
Date: 7/04/2013

Machine: #11
Units: 0.0001inch

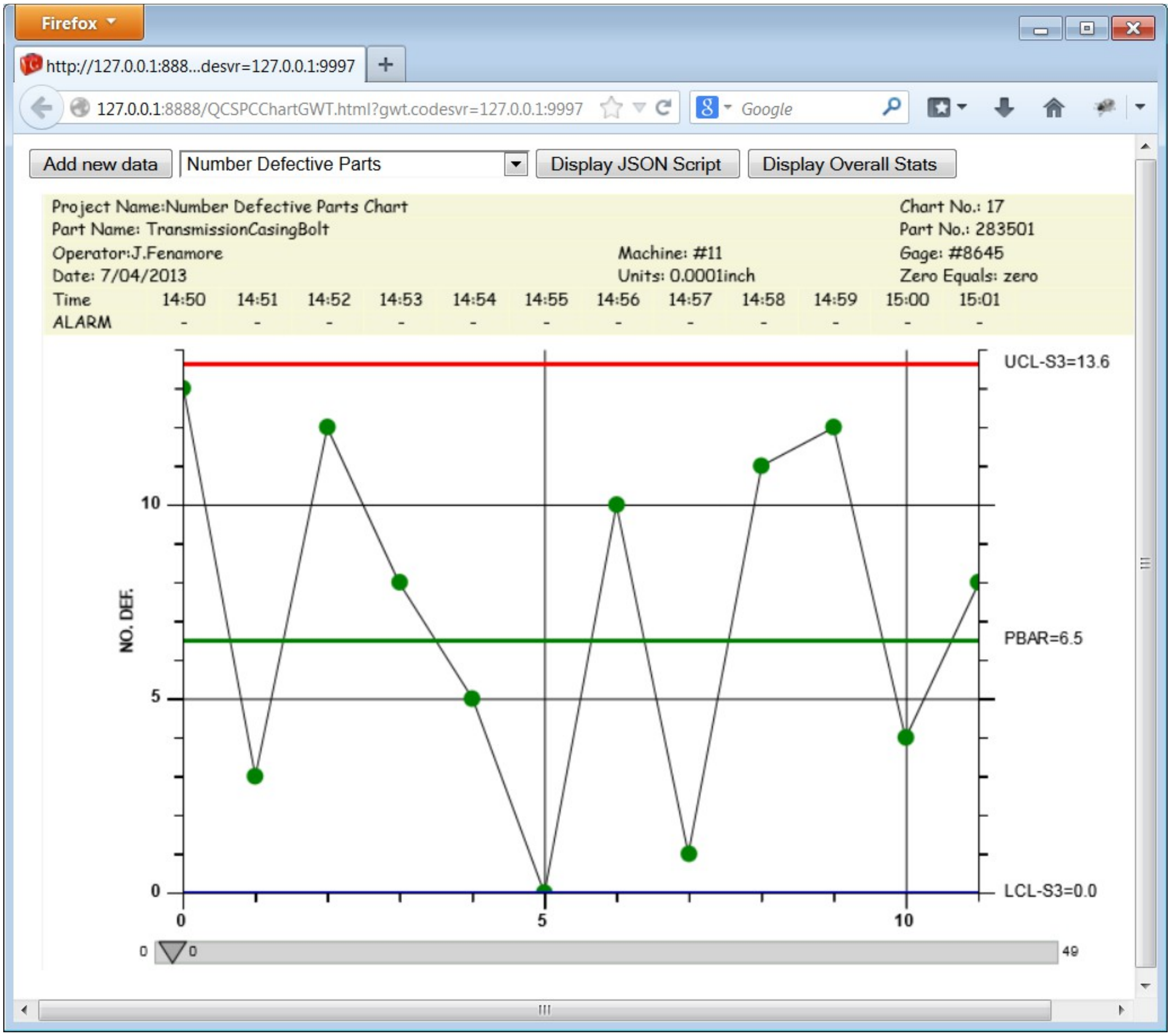
Chart No.: 17
Part No.: 283501
Gage: #8645
Zero Equals: zero

Time	14:50	14:51	14:52	14:53	14:54	14:55	14:56	14:57	14:58	14:59	15:00	15:01
ALARM	-	L	-	-	L	-	-	-	-	-	-	-

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)


Firefox
http://127.0.0.1:888...desvr=127.0.0.1:9997
127.0.0.1:8888/QCSPCChartGWT.html?gwt.codesvr=127.0.0.1:9997
Google
Add new data
Number Defective Parts
Display JSON Script
Display Overall Stats

Project Name:Number Defective Parts Chart   Chart No.: 17
Part Name: TransmissionCasingBolt   Part No.: 283501
Operator:J.Fenamore   Machine: #11   Gage: #8645
Date: 7/04/2013   Units: 0.0001inch   Zero Equals: zero

Time	14:50	14:51	14:52	14:53	14:54	14:55	14:56	14:57	14:58	14:59	15:00	15:01
ALARM	-	-	-	-	-	-	-	-	-	-	-	-

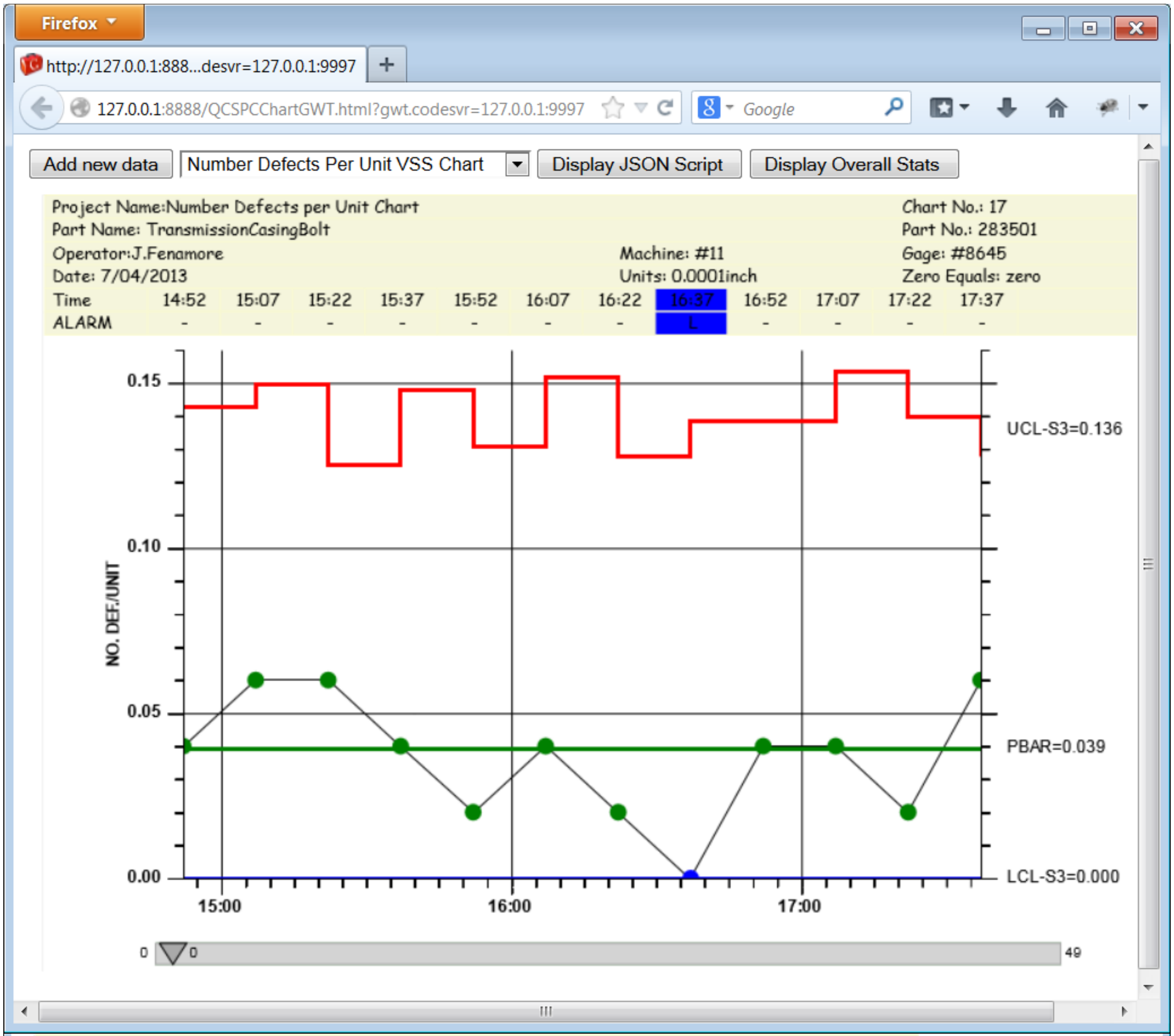
c-Chart - Also known as the Number of Defects or Number of Non-Conformities Chart

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.

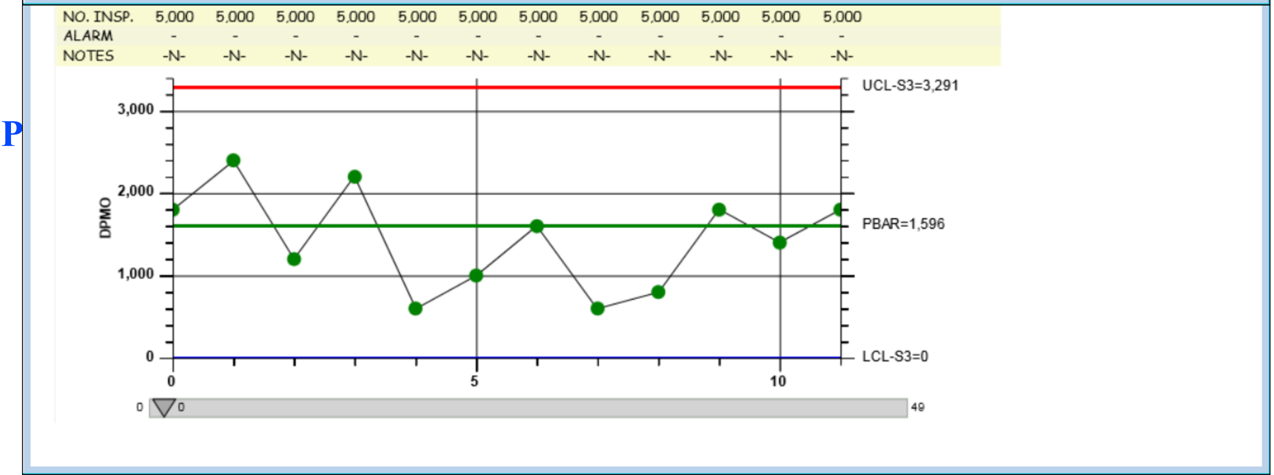
The u-Chart chart can also be used if the sample subgroup size varies from sampling interval to sampling interval. In this case, the control chart high and low limits vary from sample interval to sample interval, depending on the number of samples in the associated sample subgroup. A low number of samples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


Firefox
http://127.0.0.1:888...desvr=127.0.0.1:9997
127.0.0.1:8888/QCSPCChartGWT.html?gwt.codesvr=127.0.0.1:9997
Google
Add new data
Number Defects Per Unit VSS Chart
Display JSON Script
Display Overall Stats

Project Name:Number Defects per Unit Chart
Part Name: TransmissionCasingBolt
Operator:J.Fenamore
Date: 7/04/2013
Machine: #11
Units: 0.0001inch
Chart No.: 17
Part No.: 283501
Gage: #8645
Zero Equals: zero

Time	14:52	15:07	15:22	15:37	15:52	16:07	16:22	16:37	16:52	17:07	17:22	17:37
ALARM	-	-	-	-	-	-	-	L	-	-	-	-

DP



NO. INSP.	5,000	5,000	5,000	5,000	5,000	5,000	5,000	5,000	5,000	5,000	5,000	5,000
ALARM	-	-	-	-	-	-	-	-	-	-	-	-
NOTES	-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.

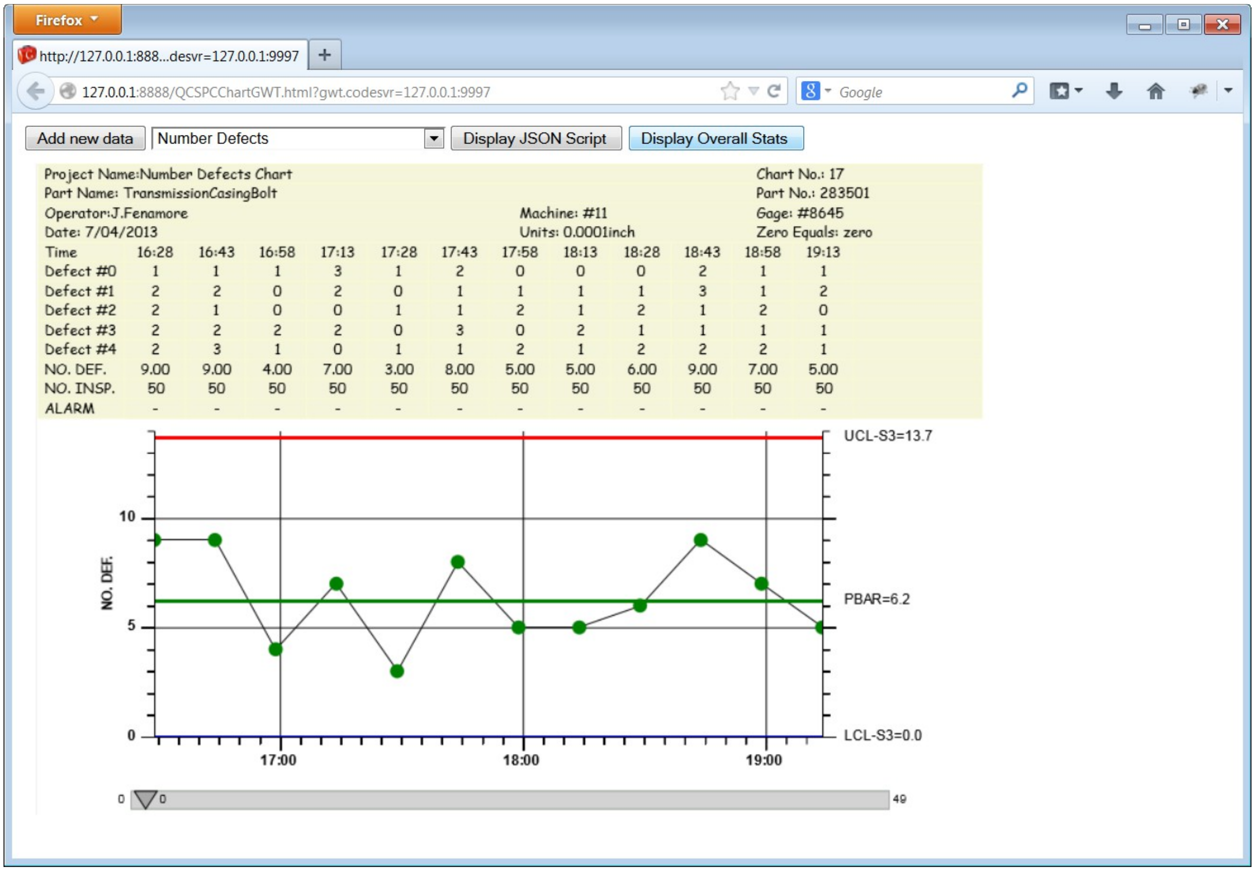
Defect and Defect Category Data Tables

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



Project Name:Number Defects Chart
Part Name: TransmissionCasingBolt
Operator:J.Fenamore
Date: 7/04/2013
Machine: #11
Units: 0.0001inch
Chart No.: 17
Part No.: 283501
Gage: #8645
Zero Equals: zero

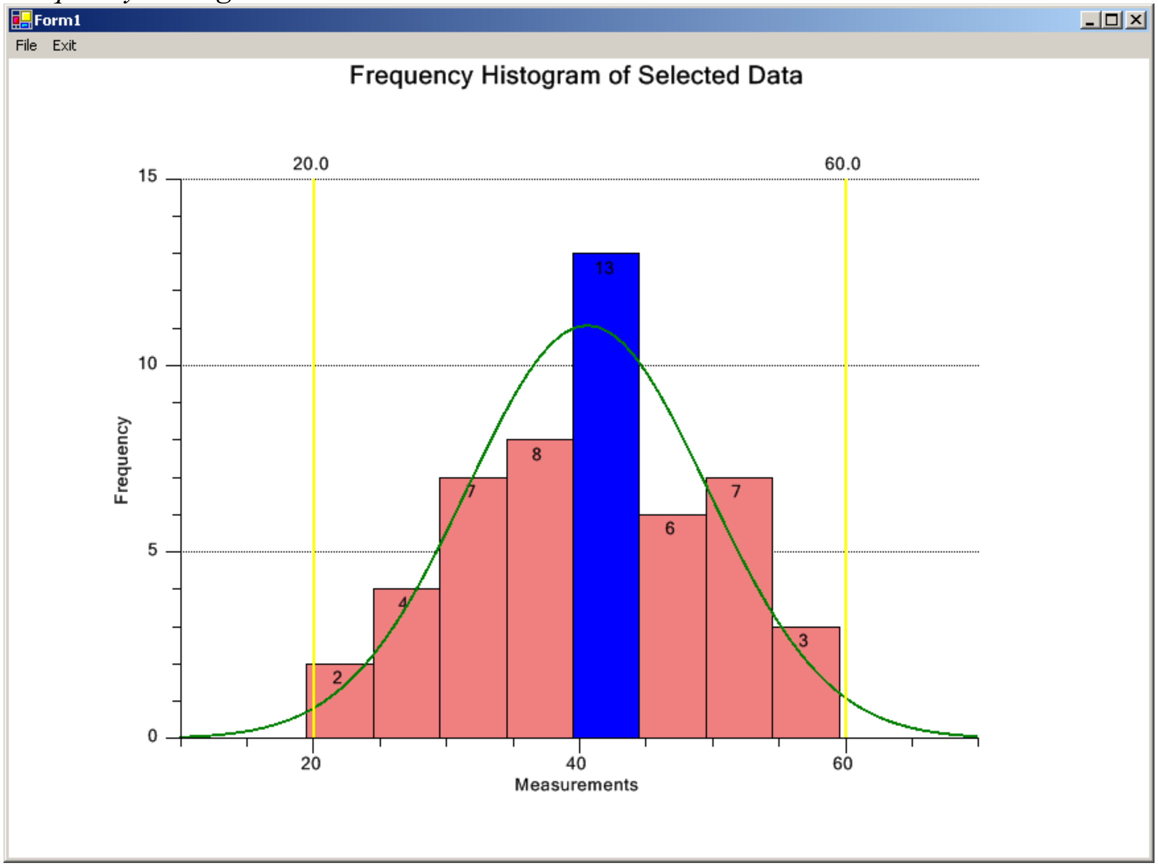
Time	16:28	16:43	16:58	17:13	17:28	17:43	17:58	18:13	18:28	18:43	18:58	19:13
Defect #0	1	1	1	3	1	2	0	0	0	2	1	1
Defect #1	2	2	0	2	0	1	1	1	1	3	1	2
Defect #2	2	1	0	0	1	1	2	1	2	1	2	0
Defect #3	2	2	2	2	0	3	0	2	1	1	1	1
Defect #4	2	3	1	0	1	1	2	1	2	2	2	1
NO. DEF.	9.00	9.00	4.00	7.00	3.00	8.00	5.00	5.00	6.00	9.00	7.00	5.00
NO. INSP.	50	50	50	50	50	50	50	50	50	50	50	50
ALARM	-	-	-	-	-	-	-	-	-	-	-	-

Other Important SPC Charts

Frequency Histogram Chart

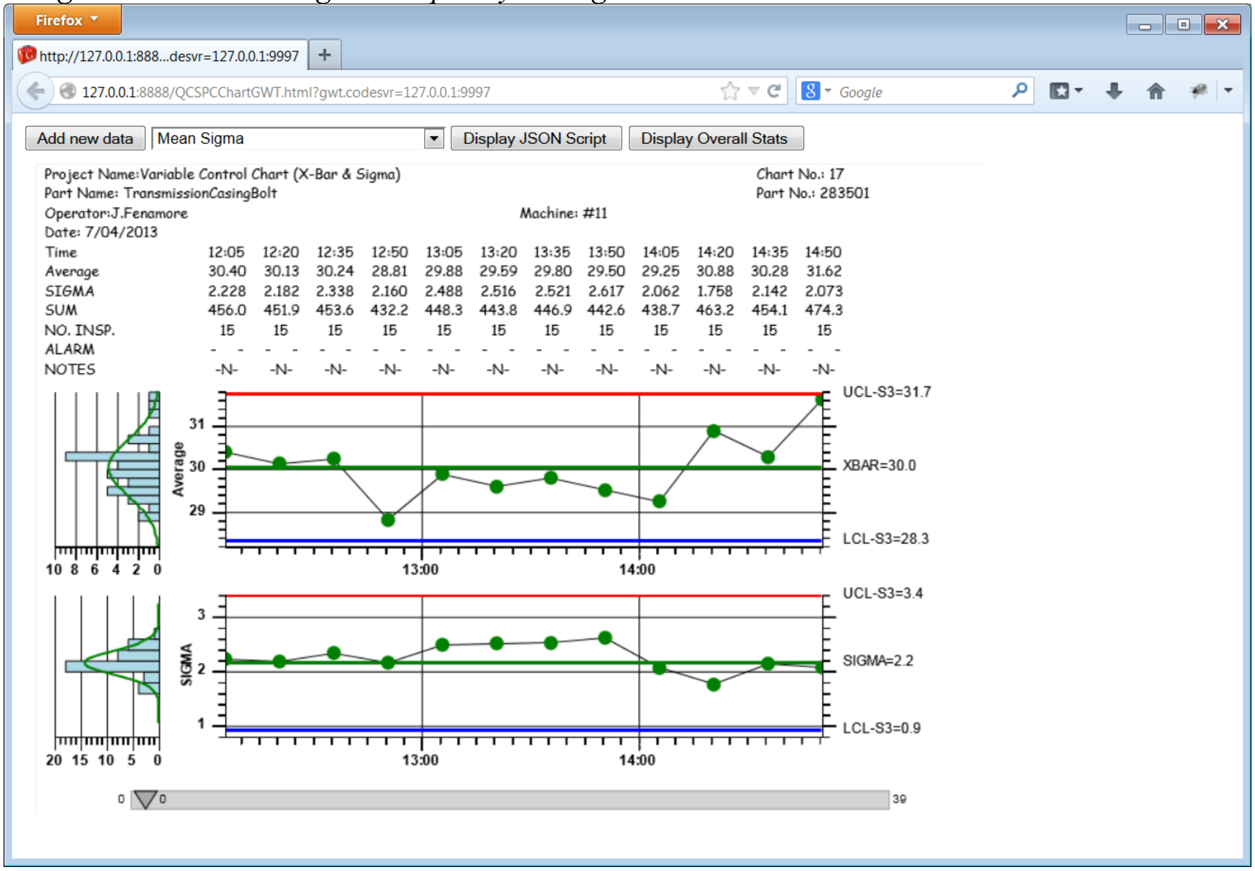
An SPC control chart tacks the trend of critical variables in a production environment. It is important that the production engineer understand whether or not changes or variation in the critical variables are natural variations due to the tolerances inherent to the production machinery, or whether or 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


Firefox
http://127.0.0.1:888...desvr=127.0.0.1:9997
127.0.0.1:8888/QCSPCChartGWT.html?gwt.codesvr=127.0.0.1:9997
Google
Add new data
Mean Sigma
Display JSON Script
Display Overall Stats

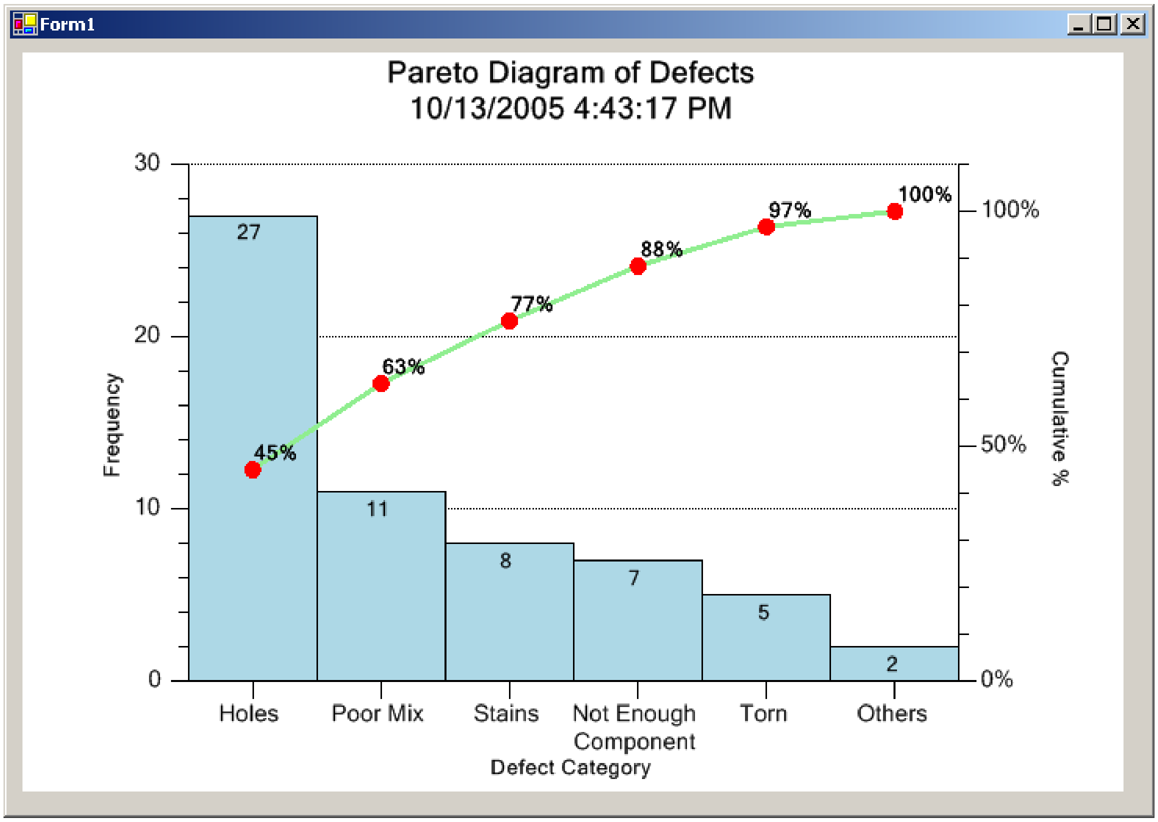
Project Name:Variable Control Chart (X-Bar & Sigma) Chart No.: 17
Part Name: TransmissionCasingBolt Part No.: 283501
Operator:J.Fenamore Machine: #11
Date: 7/04/2013

Time	12:05	12:20	12:35	12:50	13:05	13:20	13:35	13:50	14:05	14:20	14:35	14:50
Average	30.40	30.13	30.24	28.81	29.88	29.59	29.80	29.50	29.25	30.88	30.28	31.62
SIGMA	2.228	2.182	2.338	2.160	2.488	2.516	2.521	2.617	2.062	1.758	2.142	2.073
SUM	456.0	451.9	453.6	432.2	448.3	443.8	446.9	442.6	438.7	463.2	454.1	474.3
NO. INSP.	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-

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. Overview of JSON Scripting of SPC Charts

Top Level

StaticProperties
SPCChart
FrequencyHistogram
ParetoChart

Secondary Level

```
StaticProperties
       Canvas
          Width
          Height
       DefaultFontName
       DefaultTableFont
       DefaultAlarmColors
       SPCChartStrings
       DefaultChartFonts
       ControlLimitLabelFont;
       DefaultMultiVariableColors
       DefaultMultiVariableSymbolSize
       DefaultMultiVariableLineWidth
       DefaultMultiVariablePlotSymbol
       AltNofMRule
       SPCChartStrings
          .
          .
          .
          TimeValueRowHeader: String: "TIME"
          AlarmStatusValueRowHeader: String: "ALARM"
          NumberSamplesValueRowHeader: String: "NO. INSP."
          TitleHeader: String: "Title: "
          PartNumberHeader: String: "Part No.: "
          ChartNumberHeader: String: "Chart No.: "
          PartNameHeader: String: "Part Name: "
          OperationHeader: String: "Operation: "
          OperatorHeader: String: "Operator: "
          .
          .
          .
```

```
        There are 125 static strings constants you can set, listed in detail in the
         next section


        InitChartProperties
        ChartPositioning
        Scrollbar
        UseNoTable
        TableSetup
        MiscChartDataProperties
        PrimaryChartSetup
        SecondaryChartSetup
        Events
        SampleData
        SecondaryChartSetup
        Methods

FrequencyHistogram
        ChartSetup
        FrequencyHistogramData
        Methods

ParetoChart
        ChartSetup
        ParetoChartData
        Methods
```

Third Level

At this level you will see all of the properties, object types, and default values if appropriate.

Our indented, hierarchical description of the JSON scripting language for the SPCChart software is slightly different that they way the JSON will look in your actual program. The following conventions are followed.

Our description of a JSON property includes the property name, the property type, and a default value if appropriate. The `SPCChartType` definition looks like:

```
        SPCChartType: SPC String constant: "MEAN_RANGE_CHART"
```

where 'SPCChartType' is the property name, 'SPC String constant' is the property type, and "MEAN_RANGE_CHART" is the default value.

In actual JSON code, all object names (objects on the left side of the colon ":", are strings and are surrounded by quotes. The description block for InitChartProperties:

```
InitChartProperties
        SPCChartType: SPC String constant: "MEAN_RANGE_CHART"
        ChartMode: SPC String constant: "Batch"
        NumSamplesPerSubgroup: integer: 5
        NumCategoriesPerSubgroup: integer: 5
```

```
      NumDatapointsInView: integer: 15
      CuSumKValue: double: 0.5
      CuSumHValue: double: 5
      CuSumMeanValue:double: 10
```

is coded in JSON as:

```
"InitChartProperties": {
      "SPCChartType": "MEAN_RANGE_CHART",
      "ChartMode": "Time",
      "NumSamplesPerSubgroup": 5,
      "NumDatapointsInView": 12,
      "TimeIncrementMinutes": 15
 },
```

By convention, property names have the starting letter of each word capitalized. In actuality, property names ignore case.

Property values (objects on the right side of the colon ":", can be primitive objects (integer, double, boolean and String), arrays of primitive objects, or arrays of property/value pairs. Arrays are comma delimited lists, surrounded by brackets "[]". The integer, double and boolean types DO NOT use quotes. The string primitive does use quotes.

The properties which start with CuSum apply only to the CuSum chart type and do not have to be included in any other chart type.

Curly brackets and commas were mostly eliminated from the outline below in order to increase readability. They remain where array objects use them to delineate an array of objects of the same type. The block:

```
NamedRuleSet
   RuleSet: SPC string constant
   RuleEnable: [boolean, boolean, …]
```

is actually coded in JSON as:

```
"NamedRuleSet"
{
   "RuleSet": "WECO_RULES",
   "RuleEnable": [true, true, true, true, true, true, false, false]
}
```

An object of type integer should be entered as an integer value, with no quotes around it. The block:

```
Canvas
   Width: integer: 800
   Height: integer: 600
```

```
is actually coded as
```

```
"Canvas": {
        "Width": 800,
        "Height": 550
},
```

An object of type double should be entered as an double value, with no quotes around it. The block:

```
ChartPositioning
   GraphStartPosX: double: 0.15
   GraphStopPosX: double:  0.8
```

is actually coded in JSON as

```
"ChartPositioning": {
      "GraphStartPosX": 0.125,
      "GraphStopPosX": 0.8
},
```

A double can be specified using a decimal format, 123.455, or an integer format, 123.

There are many string constants used in the software. A string constant is a predefined string which represents a specific value to the software. One type of string constant is a color constant. The software supports a long list of predefined names for colors. **A complete list of the supported Color constants appears in the Appendix.** String constants are always surrounded by quotes. The block:

```
DefaultAlarmColors
   Target: Color String constant: "GREEN"
   LowAlarm: Color String constant: "BLUE"
   HighAlarm: Color String constant: "RED"
```

is coded in JSON as:

```
"DefaultAlarmColors": {
   "Target": "GREEN",
   "LowAlarm": "BLUE",
   "HighAlarm": "RED"
},
```

Another type of string constant are SPCChart constants. They are used throughout the software. Like the color constants, they must be surrounded by quotes. The block:

```
TableSetup
   HeaderStringsLevel: String
   EnableCategoryValues: boolean
   EnableCalculatedValues: boolean
   EnableTotalSamplesValues: boolean
   EnableNotes: boolean
   TableBackgroundMode: SPC String constant
   BackgroundColor1: Color String constant
   BackgroundColor2: Color String constant
```

is coded in JSON as:

```
"TableSetup": {
      "HeaderStringsLevel": "HEADER_STRINGS_LEVEL3",
      "EnableCategoryValues": true,
      "EnableCalculatedValues": true,
      "EnableTotalSamplesValues": false,
      "EnableNotes": false,
      "TableBackgroundMode": "TABLE_STRIPED_COLOR_BACKGROUND",
      "BackgroundColor1": "BEIGE",
      "BackgroundColor2": "LIGHTGOLDENRODYELLOW"

}
```

By convention, color and other constants are shown in all caps. In actuality, constants ignore case.

Below is the hierarchical structure of the QCSPCChart JSON scripting language.

```
StaticProperties
   Canvas
      Width: integer: 800
      Height: integer: 600

   SPCChartStrings
      .
      .
      .
      TimeValueRowHeader: String: "TIME"
      AlarmStatusValueRowHeader: String: "ALARM"
      NumberSamplesValueRowHeader: String: "NO. INSP."
      TitleHeader: String: "Title: "
      PartNumberHeader: String: "Part No.: "
      ChartNumberHeader: String: "Chart No.: "
      PartNameHeader: String: "Part Name: "
      OperationHeader: String: "Operation: "
      OperatorHeader: String: "Operator: "
      .
      .
      .
      There are 125 static strings constants you can set, listed in detail in the
       next section

   DefaultMultiVariableColors
   DefaultMultiVariableSymbolSize
   DefaultMultiVariableLineWidth
   DefaultMultiVariablePlotSymbol
   AltNofMRule
   DefaultFontName: String: "sans-serif"
   DefaultTableFont:
      Name: String: "sans-serif"
      Size: double: 14
      Style: String: "PLAIN"
   DefaultAlarmColors
      Target: Color String constant: "GREEN"
      LowAlarm: Color String constant: "BLUE"
      HighAlarm: Color String constant: "RED"
      Trending: Color String constant:  "TAN"
      Stratification: Color String constant  :  "SALMON"
      Alternating: Color String constant "ORANGE"
      HighSpecLimit: Color String constant: "RED"
      LowSpecLimit: Color String constant: "BLUE"

   DefaultChartFonts
      AxisLabelFont
             Name: String: "sans-serif"
             Size: double: 12
          Style: String: "BOLD"
      AxisTitleFont: standard Name, Size:12, Style: BOLD font properties
      MainTitleFont: standard Name, Size:18, Style: BOLD font properties
      SubHeadFont: standard Name, Size:14, Style: BOLD font properties
      ToolTipFont: standard Name, Size:12, Style: PLAIN font properties
      AnnotationFont: standard Name, Size:12, Style: PLAIN font properties
      ControlLimitLabelFont: standard Name, Size:12, Style: PLAIN font properties

   AltNofMRule: boolean: false



SPCChart
   InitChartProperties
      SPCChartType: SPC String constant: "MEAN_RANGE_CHART"
      ChartMode: SPC String constant: "Batch"
      NumSamplesPerSubgroup: integer: 5
      NumCategoriesPerSubgroup: integer: 5
      NumDatapointsInView: integer: 15
      TimeIncrementMinutes: double: 15
      CuSumKValue: double: 0.5
      CuSumHValue: double: 5
      CuSumMeanValue:double: 10
```

The next three properties below are used only with Multi-Variable Chart types

```
   NumberAddedVariables: integer: 2
   MultiVariableMixedPrimaryChartType:SPC String constant: "MEAN_RANGE_CHART”
   MultiVariableMixedChartSubTypes: [SPC String constant: "MEAN_RANGE_CHART”,..]



   ChartPositioning
      GraphStartPosX: double: 0.15
      GraphStopPosX: double:  0.8
      TableStartPosY: double: 0.0
      GraphTopTableOffset : double: 0.02
      InterGraphMargin: double: 0.075
      GraphBottomPos: double: 0.90
      BottomLabelMargin: double: 0.0
   Scrollbar
      EnableScrollBar: boolean: true
      ScrollbarPosition:SPC String constants: "SCROLLBAR_POSITION_MIN"
```

```
    ScrollbarValue: double: 0
    EnableZoomToggle: boolean: true
UseNoTable
    PrimaryChart: boolean: true
    SecondaryChart: boolean: true
    Histograms: boolean: true
    Title: String: ""
TableSetup
    HeaderStringsLevel: SPC String constant: "HEADER_STRINGS_LEVEL2"
    EnableInputStringsDisplay: boolean: true
    EnableCategoryValues: boolean: true
    EnableSampleValues: boolean: true
    EnableCalculatedValues: boolean: true
    EnableProcessCapabilityValues: boolean: true
    EnableTotalSamplesValues: boolean: true
    EnableNotes: boolean: true
    EnableTimeValues: boolean: true
    EnableDataToolTip: boolean: true
    EnableNotesToolTip: boolean: true
    EnableTableRowToggles: boolean: false
    NotesToolTip
        ButtonMask: SPC String constant: "BUTTON1_MASK"
        ToolTipMode: SPC String constant: "MOUSETOGGLE_TOOLTIP"
        NotesReadOnly: boolean: false
    TableBackgroundMode:SPC String Constant: "TABLE_SINGLE_COLOR_BACKGROUND"
    TableAlarmEmphasisMode: SPC String Constant: "ALARM_HIGHLIGHT_NONE"
    ChartAlarmEmphasisMode: SPC String Constant: "ALARM_HIGHLIGHT_SYMBOL"
    BackgroundColor1: Color String constant: "WHITE"
    ChartData
        Title: String: ""
        PartNumber: String: ""
        ChartNumber: String: ""
        PartName: String: ""
        Operation: String: ""
        SpecificationLimits: String: ""
        Operator: String: ""
        Machine: String: ""
        Gauge: String: ""
        UnitOfMeasure: String: ""
        ZeroEquals: String: ""
        DateString: String: ""
        NotesMessage: String: ""
        ProcessCapabilitySetup
            LSLValue: double: 0
            USLValue: double: 1
            EnableCPK: boolean: false
            EnableCPM: boolean: false
            EnablePPK: boolean: false
            EnableCPL: boolean: false
            EnableCPU: boolean: false
            EnablePPL: boolean: false
            EnablePPU: boolean: false
        SampleItemDecimals: integer: 2
        CalculatedItemDecimals: integer: 2
        ProcessCapabilityDecimals: integer: 2
        CustomTimeFormatString: String: ""
        SampleRowHeaderStrings: [String: "", String: "",..]
        TimeFormat: SPC String constant: "TIMEDATEFORMAT_24HM"
        TitleHeader: String: "Title:"
        PartNumberHeader : String: "Part No:"
        ChartNumberHeader: String: "Chart No:"
        PartNameHeader: String: "Part Name:"
        OperationHeader: String: "Operation:"
        SpecificationLimitsHeader: String: "Spec. Limits:"
        OperatorHeader: String: "Operator:"
        MachineHeader: String: "Machine:"
        GaugeHeader: String: "Date:"
        UnitOfMeasureHeader: String: "Units:"
        ZeroEqualsHeader: String: "Zero Equals:"
        DateHeader: String: "Date:"
        NotesHeader: String: "Notes:"
        TimeValueRowHeader: String: "TIME"

  MiscChartDataProperties
        MA_W: integer: 5
        EWMA_Lambda: double:  0.2
        EWMA_UseSSLimits: boolean: false
        EWMA_StartingValue: double: 1.0
        DefaultControlLimitSigma: double: 3
        AutoLogAlarmsAsNotes: boolean: false
        AlarmTimeFormatString: String: "EEE, d MMM yyyy HH:mm:ss Z"
        DefectOpportunitiesPerUnit: double: 1
        NotesReadOnly: boolean: false
        AlarmReportMode: SPC String constant: "REPORT_ALL_ALARMS"
        EnableDisplayOptionsToggles: boolean: true

        AddNote
           Index: integer: 0
           Note: string: ""
           Append: boolean: false

        Legend chart features found only in QCSPCChart+
        Legend
           EnableLegend: boolean: true
           LegendStrings: [string, string, ...] :
                            ["Item-1", "Item-2", "Item-3"] ,...]
           LegendFont:
              Name: String: "sans-serif"
              Size: integer: 18
              Style: SPC String constant: "Plain"

        Multi-Varaible chart features found only in QCSPCChart+
         MultiVariableItems
           YAxisMapping:  [SPC String constant, SPC String constant, ...]:
                          ["LEFT_AXIS","LEFT_AXIS"]
           MultiVariableSymbolSize: [ integer, integer,...]: [16, 12]
           MultiVariableLineWidth:: [ integer, integer,...]: [2, 3]
          MultiVariablePlotSymbol:[SPC String constant,SPC String constant, ...]:
                          ["SQUARE", "SQUARE"],
           MultiVariableColors:[SPC Color constant, SPC Color constant, ...]:
                         ["PURPLE", "BROWN"]
```

```
PrimaryChartSetup
      EnableChart: boolean: true
      EnableChartDisplayToggles: boolean: true
      XAxis
         LineColor: Color String constant: "BLACK"
         LineWidth: double: 1
         Enable: boolean: true
      XAxisLabels
         Font
            Name: String: "sans-serif"
            Size: double: 12
            Style: SPC String Constant: "PLAIN"
         TextColor: Color String constant: "BLACK"
         Rotation: double: 0
         Format: SPC String constant: "TIMEDATEFORMAT_24HM"
         CustomFormatString: String: ""
         OverlapLabelMode: SPC String constant: "OVERLAP_LABEL_STAGGER"
         AxisLabelMode: SPC String constant: "AXIS_LABEL_MODE_DEFAULT"
         Enable: boolean: true
      YAxisLeft
         LineColor: Color String constant: "BLACK"
         LineWidth: double: 1
         MinValue: double: 0
         MaxValue: double: 1

      YAxisLeftLabels
         Font
            Name: String: "sans-serif"
            Size: double: 12
            Style: SPC String Constant: "PLAIN"
         TextColor: Color String constant: "BLACK"
         Rotation: double: 0
         Format:  constant(String)
         OverlapLabelMode:  SPC String constant: "OVERLAP_LABEL_STAGGER"
         Decimal: integer: 1
         AxisLabelMode: SPC String constant: "AXIS_LABEL_MODE_DEFAULT"
         Enable: boolean: true

      YAxisRight
         LineColor: Color String constant: "BLACK"
         LineWidth: double: 1
         Enable: boolean: true
```

Multi-Varaible chart features found only in QCSPCChart+

```
         MinValue: double: 0
         MaxValue: double: 1

      FrequencyHistogram
         EnableDisplayFrequencyHistogram: boolean: true
         PlotBackgroundColor : Color String constant: "WHITE"
         BarColor: Color String constant: "LIGHTBLUE"

      PlotMeasurementValues: boolean: false
      LineMarkerPlot
         LineColor: Color String constant: "BLUE"
         LineWidth: double: 1
         SymbolColor: Color String constant: "BLUE"
```

```
   SymbolFillColor: Color String constant: "BLUE"
   SymbolType: SPC String constant: "CIRCLE"
   Enable: boolean: true

GraphBackground
   FillColor: Color String constant: "WHITE"
   BackgroundMode: SPC String constant: "SIMPLECOLORMODE"
   GradientStartColor: Color String constant: "WHITE"
   GradientStopColor: Color String constant: "LIGHTGRAY"
   Enable: boolean: true

PlotBackground
   FillColor: Color String constant: "WHITE"
   BackgroundMode: SPC String constant: "SIMPLECOLORMODE"
   GradientStartColor: Color String constant: "WHITE"
   GradientStopColor: Color String constant: "LIGHTGRAY"
   Enable: boolean: true

OutOfLimitSymbolSize: integer: 18
OutOfLimitSymbolType: SPC String constant:: "SQUARE"


ControlLimits
   Font
      Name: String: "sans-serif"
      Size: double: 12
      Style: SPC String Constant: "PLAIN"
   DefaultLimits [ boolean: true, boolean: true]
   SetLimits: [double: 0, double: 0, double 0]
   Decimal: integer: 1
   ZoneFill: boolean: false
   ZoneColors:  [
                   Color String constant: "ORANGERED",
                   Color String constant: "LIGHTGOLDENRODYELLOW",
                   Color String constant: "LIGHTGREEN"
                ]
   Target
      LineColor: Color String constant: "GREEN"
      TextColor: Color String constant: "BLACK"
      LineWidth: double: 1
      LimitValue: double: 0
      DisplayString: String: "XBAR"
      EnableAlarmLine: boolean: true
      EnableAlarmChecking : boolean: true
   LCL3
      LineColor: Color String constant
      TextColor: Color String constant
      LineWidth: double: 1
      LimitValue: double: 0
      DisplayString: String: "LCL"
      EnableAlarmLine: boolean: true
      EnableAlarmChecking : boolean: true
      EnableAlarmLineText: String: true
   UCL3
      LineColor: Color String constant
      TextColor: Color String constant
   LineWidth: double: 1
   LimitValue: double: 0
   DisplayString: String: UCL
   EnableAlarmLine: boolean: true
   EnableAlarmChecking : boolean: true
   EnableAlarmLineText: String: true
LCL2
   LineColor: Color String constant
   TextColor: Color String constant
   LineWidth: double: 1
   LimitValue: double: 0
   DisplayString: String: "LCL"
   EnableAlarmLine: boolean: true
   EnableAlarmChecking : boolean: true
   EnableAlarmLineText: String: true
UCL2
   LineColor: Color String constant
   TextColor: Color String constant
   LineWidth: double: 1
   LimitValue: double: 0
   DisplayString: String: UCL
   EnableAlarmLine: boolean: true
   EnableAlarmChecking : boolean: true
   EnableAlarmLineText: String: true
LCL1
   LineColor: Color String constant
   TextColor: Color String constant
   LineWidth: double: 1
   LimitValue: double: 0
   DisplayString: String: "LCL"
   EnableAlarmLine: boolean: true
   EnableAlarmChecking : boolean: true
   EnableAlarmLineText: String: true
UCL1
   LineColor: Color String constant
   TextColor: Color String constant
   LineWidth: double: 1
   LimitValue: double: 0
   DisplayString: String: UCL
   EnableAlarmLine: boolean: true
   EnableAlarmChecking : boolean: true
   EnableAlarmLineText: String: true
123SigmaControlLimits
   Target: double: 0
   LCL3Value: double: 0
   UCL3Value: double: 0
   AlarmTest12: boolean: true
   EnableAlarmLine: boolean: true
   EnableAlarmChecking: boolean: true
   EnableAlarmLineText: boolean: true
NamedRuleSet
      RuleSet: SPC string constant
      RuleEnable [boolean, boolean …]
      CustomizeRules: [   {
                                 "RuleNumber": 15,
                                 "M": 18,
                                       "N": 15
                                 },

                                 {     "RuleNumber": 15,
                                       "M": 18,
                                       "N": 15
                                 },
                                 ...
                              ]
   AddControlRules
   [   {
             RuleSet: : SPC String constant: "BASIC_RULES"
             RuleNumber: integer: 2
             EnableAlarmLine: boolean: true
             EnableAlarmChecking: boolean: true
             EnableAlarmLineText: String: true
             LimitValue: double: 0
             N: integer: 1
             M: integer: 1
             TemplateNumber: integer: 2
             SigmaLevel: double: 3
       },
       {
             RuleSet: : SPC String constant: "BASIC_RULES"
             RuleNumber: integer: 2
             EnableAlarmLine: boolean
             EnableAlarmChecking: boolean
             EnableAlarmLineText: String
             LimitValue: double: 0
             N: integer: 1
             M: integer: 1
       }, ...

   ]
   SpecifyControlLimitsUsingMeanAndSigma
       Mean: double: 1
       Sigma: double: 1

   AutoDeleteControlLimits
       LimitValue: double: 0
       ComparisonOperator: double: 0



SpecificationLimits
   Font
       Name: String: "sans-serif"
       Size: double: 12
       Style: SPC String Constant: "PLAIN"
   Decimal: integer: 1
   LowSpecificationLimit
       LineColor: Color String constant: "BLUE"
       TextColor: Color String constant: "BLACK"
       LineWidth: double: 1
```

```
          LimitValue: double: 0
          DisplayString: String: "LSL"
          EnableAlarmLine: boolean: true
          nableAlarmChecking  : boolean: true
          EnableAlarmLineText: String: true
       HighSpecificationLimit
          LineColor: Color String constant: "RED"
          TextColor: Color String constant: "BLACK"
          LineWidth: double: 1
          LimitValue: double: 0
          DisplayString: String: "USL"
          EnableAlarmLine: boolean: true
          EnableAlarmChecking : boolean: true



SecondaryChartSetup

       The SecondaryChartSetup is same as PrimaryChartSetup, except
   that there is no NamedRuleSet block

ThirdChartSetup

       The ThirdChartSetup is same as PrimaryChartSetup, except
       that there is no NamedRuleSet block. The  ThirdChartSetup
       only applies to the Xbar-R-MR chart type and its multi-variable variant

Annotations
       [ {
          ChartPosition: SPC String Constant: "PRIMARY_CHART":
          DatapointIndex: integer: 14
          Font
             Name: String: "sans-serif"
             Size: double: 12
             Style: SPC String Constant: "PLAIN"
          Text: string: "Text"
          TextColor: Color String constant: "RED"
          Justify: SPC String Constant: "ANNOTATION_UPPER_RIGHT"
          LineColor: Color String constant: "BLACK"
          LineWidth: integer: 1
         },
          {
          ChartPosition: SPC String Constant: "PRIMARY_CHART":
          DatapointIndex: integer: 14
          Font
             Name: String: "sans-serif"
             Size: double: 12
             Style: SPC String Constant: "PLAIN"
          Text: string: "Text"
          TextColor: Color String constant: "RED"
          Justify: SPC String Constant: "ANNOTATION_UPPER_RIGHT"
          LineColor: Color String constant: "BLACK"
          LineWidth: integer: 1

         },
          .
```

```
              .
              .
          ]



    Events
       AlarmStateEventEnable: boolean: true
       AlarmTransitionEventEnable: boolean: false
       EnableDataToolTip: boolean: true
       EnableJSONDataToolTip: boolean: false
       DataToolTip
           EnableCategoryValues: boolean: false
           EnableProcessCapabilityValues: boolean: false
           EnableCalculatedValues: boolean: false
           EnableNotesString: boolean: false
       EnableNotesToolTip: boolean: true
       NotesToolTip;
          ButtonMask: SPC String constant: "BUTTON1_MASK"
          ToolTipMode: SPC String constant: "MOUSETOGGLE_TOOLTIP"
          NotesReadOnly: boolean: false
       EnableAlarmStatusValues: boolean: true
       ChartAlarmEmphasisMode: SPC string constant: "ALARM_HIGHLIGHT_SYMBOL"

    SampleData
       SampleIntervalRecords
       [  {
             TimeStamp: double:
             BatchCount: integer: 1
             Note: String: ""
             BatchIDString: String: ""
             VariableControlLimits: [double:1,double:1, ...]
             SampleSubgroupSize_VSS: integer: -1
             SampleValues [double, double,... ]

Variable Specification Limits features found only in QCSPCChart+
             VariableSpecificationLimits: [double:1,double:1, ...]


Main Variable Alarm Forcing features found only in QCSPCChart+
             PrimaryForceAlarm: SPC string constant: "NO_FORCING"
             SecondaryForceAlarm: SPC string constant: "NO_FORCING"
             ThirdForceAlarm: SPC string constant: "NO_FORCING"

Multi-Variable chart features found only in QCSPCChart+
             MultiVariableSampleValues1: [double, double,... ]
             MultiVariableSampleValues2: [double, double,... ]
             MultiVariableSampleValues3: [double, double,... ]
             MultiVariableSampleValues4: [double, double,... ]
             MultiVariableSampleValues5: [double, double,... ]

Multi-Variable Alarm Forcing features found only in QCSPCChart+
             MultiVariablePrimaryForceAlarm: [SPC string constant: "NO_FORCING",
                                              SPC string constant: "NO_FORCING",
                                              …]
```

```
            MultiVariableSecondaryForceAlarm: [SPC string constant: "NO_FORCING",
                                              SPC string constant: "NO_FORCING",
                                              …]
            MultiVariableThirdForceAlarm: [SPC string constant: "NO_FORCING",
                                              SPC string constant: "NO_FORCING",
                                              …]


         },
         {
            TimeStamp: double:
            BatchCount: integer: 2
            Note: String: ""
            BatchIDString: String: ""
            VariableControlLimits: [double:1,double:1, ...]
            VariableSpecificationLimits: [double:1,double:1, ...]
            SampleSubgroupSize_VSS: integer: -1
            SampleValues [double, double,... ]

            Variable Specification Limits features found only in QCSPCChart+
            Main Variable Alarm Forcing features found only in QCSPCChart+
            Multi-Variable chart features found only in QCSPCChart+
            Multi-Variable Alarm Forcing features found only in QCSPCChart+

         },
      ]

      DataSimulation
         StartCount: integer: 0
         Count: integer: 20
         Mean: double: 1
         Range: range: 1
      ExcludeRecords: [integer, integer, ..]
      IncludeRecords: [integer, integer, ..]
      ResetSPCChartData
   Methods
      AutoCalculateControlLimits [boolean, boolean]
      AutoScaleYAxes [boolean, boolean]
      RebuildUsingCurrentData
      UpdateDisplay

      ResetControlLimitNofMCounts
      AutoDeleteControlLimits


FrequencyHistogram
   ChartSetup
         ChartPositioning
            X1: double
            Y1: double
            X2: double
            Y2: double
         MainTitle
            Font
               Name: String
               Size: double
               Style: String
            TextColor: Color String constant
            Text: String
         CoordinateSystem
            MinXScale: double
            MinYScale: double
            MaxXScale: double
            MaxYScale: double
         XAxis
            LineColor: Color String constant
            LineWidth: double
         XAxisLabels
            Font
               Name: String
               Size: double
               Style: String
            TextColor: Color String constant
            Rotation: double
            Format: SPC String constant
            OverlapLabelMode: SPC String constant
            Decimal: integer
            AxisLabelsStrings: [String, String, ...]
         XAxisTitle
            Font
               Name: String
               Size: double
               Style: String
            TextColor: Color String constant
            Text: String
         YAxis
            LineColor: Color String constant
            LineWidth: double
         YAxisLabels
            Font
               Name: String
               Size: double
               Style: String
            TextColor: Color String constant
            Rotation: double
            Format: SPC String constant
            OverlapLabelMode: SPC String constant
            Decimal: integer
            AxisLabelsStrings: [String, String, ...]
         YAxisTitle
            Font
               Name: String
               Size: double
               Style: String
            TextColor: Color String constant
            Text: String
         HistogramPlot
            LineColor: Color String constant
            LineWidth: double
            BarColor: Color String constant
         GraphBackground
```

```
            FillColor: Color String constant
            BackgroundMode: SPC String constant
            GradientStartColor: Color String constant
            GradientStopColor: Color String constant
         PlotBackground
            FillColor: Color String constant
            BackgroundMode: SPC String constant
            GradientStartColor: Color String constant
            GradientStopColor: Color String constant
      LimitValueDecs: integer
      ControlLines
      [  {
            LimitValue: double
            LineColor: Color String constant
            LineWidth: double
         },
         {
            LimitValue: double
            LineColor: Color String constant
            LineWidth: double
         }, ...
      ]
      NormalCurveLine
      {
         Enable: boolean
         LineColor: Color String constant
         LineWidth: double
      }
   FrequencyHistogramData
      SampleValues [double, double, ...]
      FrequencyBins [double, double, ...]
   Methods
      RebuildAndDraw


ParetoChart
   ChartSetup
         ChartPositioning
            X1: double
            Y1: double
            X2: double
            Y2: double
         MainTitle
            Font
               Name: String
               Size: double
               Style: String
            TextColor: Color String constant
            Text: String
         CoordinateSystem1
            MinXScale: double
            MinYScale: double
            MaxXScale: double
            MaxYScale: double
         CoordinateSystem2
            MinXScale: double
```

```
        MinYScale: double
        MaxXScale: double
        MaxYScale: double
    XAxis
        LineColor: Color String constant
        LineWidth: double
    XAxisLabels
        Font
            Name: String
            Size: double
            Style: String
        TextColor: Color String constant
        Rotation: double
        Format: SPC String constant
        OverlapLabelMode: SPC String constant
        Decimal: integer
        AxisLabelsStrings: [String, String, ..]
    XAxisTitle
        Font
            Name: String
            Size: double
            Style: String
        TextColor: Color String constant
        Text: String
    YAxisLeft
        LineColor: Color String constant
        LineWidth: double
        MinValue: double
        MaxValue: double

    YAxisLeftLabels
        Font
            Name: String
            Size: double
            Style: String
        TextColor: Color String constant
        Rotation: double
        Format: SPC String constant
        OverlapLabelMode: SPC String constant
        Decimal: integer
        AxisLabelsStrings: [String, String, ..]
    YAxisLeftTitle
        Font
            Name: String
            Size: double
            Style: String
        TextColor: Color String constant
        Text: String
    YAxisRight
        LineColor: Color String constant
        LineWidth: double
        MinValue: double
        MaxValue: double

    YAxisRightLabels
        Font
```

```
             Name: String
             Size: double
             Style: String
          TextColor: Color String constant
          Rotation: double
          Format: SPC String constant
          OverlapLabelModeconstant (String)
          Decimal: integer
          AxisLabelsStrings: [String, String, ...]
       YAxisRightTitle
          Font
             Name: String
             Size: double
             Style: String
          TextColor: Color String constant
          Text: String
       BarPlot
          LineColor: Color String constant
          LineWidth: double
          BarColor: Color String constant
       LineMarkerPlot
          LineColor: Color String constant
          LineWidth: double
          SymbolFillColor: Color String constant
          SymbolLineColor: Color String constant
          SymbolColor: Color String constant
          SymbolSize: double
       GraphBackground
          FillColor: Color String constant
          BackgroundMode: SPC String constant
          GradientStartColor: Color String constant
          GradientStopColor: Color String constant
       PlotBackground
          FillColor: Color String constant
          BackgroundMode: SPC String constant
          GradientStartColor: Color String constant
          GradientStopColor: Color String constant
ParetoChartData
       CategoryItems [double, ...]
       CategoryStrings [String, ...]
Methods
   RebuildAndDraw
```

Full List of the Static SPCChartStrings Objects

This is a list of the default, static, strings used in the software. You can change the value to string you want. In some case, such as font-family names, and time formats, you must supply a valid string.

```
SPCChartStrings
```

start	"start" - used to mark the beginning of the array

ChartFont	"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
GaugeHeader	"Gauge: " - 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" - Moving Average
DefaultCusumCPlus	"C+" - CuSum Plus string
DefaultCusumCMinus	"C-" - CuSum Minus string
DefaultEWMA	"EWMA" - EWMA string

DefaultPercentDefective	"% DEF." - Percent Defective
DefaultFractionDefective	"FRACT. DEF." - Fraction Defective
DefaultNumberDefective	"NO. DEF." - Number Defective
DefaultNumberDefects	"NO. DEF." - Number Defects
DefaultNumberDefectsPerUnit	"NO. DEF./UNIT" - Number Defects per Unit
DefaultNumberDefectsPerMillion	"DPMO" - Number Defects per Million
DefaultPBar	"PBAR" - Target label for Attribute charts
DefaultAttributeLCL	"LCLP" - Low limit label for Attribute charts
DefaultAttributeUCL	"UCLP" - High limit label for Attribute charts
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, usually over-ridden when data log opened.
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",
Weco	"WECO" - WECO rules string
Wecowsupp	"WECO+SUPPLEMENTAL" - WECO 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 Westgard Rule R2s #9
SigmaShort	"S" - Used in alarm messages as sigma short string
BbeyondAlarmStatus	"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
R4sAlarmStatus	"R" - alarm status line - R4s 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"
DefaultCpl	"Cpl"
DefaultCpu	"Cpu"
DefaultCpk	"Cpk"
DefaultCpm	"Cpm"
DefaultPp	"Pp"
DefaultPl	"Pl"
DefaultPu	"Pu"
DefaultPpk	"Ppk"
Canceltext	"Cancel" - used for buttons in dialogs that have cancel button
Alarmstatusdialogtitle	"Alarm Status" - used as the title for the alarm

	status dialog box
end	"end" - used to mark the end of the array

The example below is extracted from the TimeXBarR script found in the chartDefExampleScripts.js file.

```
 "StaticProperties": {
                     "Canvas": {
                         "Width": 800,
                         "Height": 550
                     },
                     "DefaultFontName": "Arial, sans-serif",
                     "DefaultTableFont": {
                         "Name": "'Comic Sans MS', cursive, sans-serif",
                         "Size": 12,
                         "Style": "Plain"
                     },
                     "SPCChartStrings": {
                         "TitleHeader": "Project Name:",
                         "DefaultMean": "Average",
                         "TimeValueRowHeader": "Time"
                     }
                 },
```

Note that the property name (i.e. "DefaultMean", and the new string value (i.e. "Average") are surrounded by quotes.

4. Static Property Initialization

```
StaticProperties
   Canvas
   SPCChartStrings
   DefaultFontName
   DefaultTableFont:
   DefaultAlarmColors
   DefaultTableFonts
   AltNofMRule
```

Multi-variable chart features found only in QCSPCChart+

```
   DefaultMultiVariableColors
   DefaultMultiVariableSymbolSize
   DefaultMultiVariableLineWidth
   DefaultMultiVariablePlotSymbol
```

There are a large number of properties which correspond to static properties in the underlying libraries. They are static because they are mean to be set once, on the loading of the parent HTML page, and stay in effect for the duration of the page lifetime, regardless of the number of charts which are displayed. Examples of this are the Canvas: Height and Width properties, the DefaultFontName used to specify the Font family used within the charts, and default alarm colors used by the various control limit display routines. There is also a long list (SPCChartStrings) of strings which can be change to regionalize the software for a specific region or language. These all fall under the StaticProperties block.

Properties are displayed in the

[Property Name]: [type]: [default value]

format for readability, which is not a JSON format. Use the example code listings for proper JSON formatted scripts.

```
StaticProperties
   Canvas
      Width: integer: 800
      Height: integer: 600
   SPCChartStrings
      .
      .
      .
      TimeValueRowHeader: String: "TIME"
      AlarmStatusValueRowHeader: String: "ALARM"
      NumberSamplesValueRowHeader: String: "NO. INSP."
      TitleHeader: String: "Title: "
      PartNumberHeader: String: "Part No.: "
      ChartNumberHeader: String: "Chart No.: "
      PartNameHeader: String: "Part Name: "
     OperationHeader: String: "Operation: "
     OperatorHeader: String: "Operator: "
     .
     .
     .
     There are 125 static strings constants you can set

  DefaultFontName: String: "sans-serif"
  DefaultTableFont:
     Name: String: "sans-serif"
     Size: double: 14
      Style: String: "PLAIN"

  DefaultAlarmColors
     Target: Color String constant: "GREEN"
     LowAlarm: Color String constant: "BLUE"
     HighAlarm: Color String constant: "RED"
     Trending: Color String constant:  "TAN"
     Stratification: Color String constant   :  "SALMON"
     Alternating: Color String constant "ORANGE"
     HighSpecLimit = Color String constant: "RED"
     LowSpecLimit =Color String constant: "BLUE"
 DefaultChartFonts
     AxisLabelFont
         Name: String: "sans-serif"
         Size: double: 12
        Style: String: "BOLD"
     AxisTitleFont: standard Name, Size:12, Style: BOLD font properties
     MainTitleFont: standard Name, Size:18, Style: BOLD font properties
     SubHeadFont: standard Name, Size:14, Style: BOLD font properties
     ToolTipFont: standard Name, Size:12, Style: PLAIN font properties
     AnnotationFont: standard Name, Size:12, Style: PLAIN font properties
     ControlLimitLabelFont: standard Name, Size:12, Style: PLAIN font properties
```

Canvas

```
   Canvas
      Width: integer: 800
      Height: integer: 600
```

The Canvas object controls the size of the underlying HTML5 Canvas that the charts are place in.

Properties

```
Width        An integer value specifying the Canvas width
Height       An integer value specifying the Canvas height
```

Example

```
    "Canvas": {
        "Width": 800,
        "Height": 550
    },
```

SPCChartStrings

There are 125 string constants (more or less) used in the software. Their values are all static. You can change any, or all of the string constants to match your requirements. See the SPCChartStrings Listing in the previous chapter for a complete list of the strings.

```
SPCChartStrings
    .
    .
    TimeValueRowHeader: String: "TIME"
    AlarmStatusValueRowHeader: String: "ALARM"
    NumberSamplesValueRowHeader: String: "NO. INSP."
    TitleHeader: String: "Title: "
    PartNumberHeader: String: "Part No.: "
    ChartNumberHeader: String: "Chart No.: "
    PartNameHeader: String: "Part Name: "
    OperationHeader: String: "Operation: "
    OperatorHeader: String: "Operator: "
    .
    .
    .
```

A complete list of SPCChartStrings is found at the end of Chapter 3.

Change the strings using the following JSON example:

```
"SPCChartStrings": {
    "DefaultMean": "Average",
    "TimeValueRowHeader": "Time"
}
```

List as many of the 125 strings as you need to change. Make sure to surround both the string property name, "DefaultMean" for example, and the string value, "Average" with quotes.

DefaultFontName

```
DefaultFontName: String: "sans-serif"
```

The DefaultFontName specifies the Font family used by default in the charts and table. So, if you want to change the default font from sans-serif to Comic Sans MS, use the following JSON statement.

```
DefaultFontName: "'Comic Sans MS'",
```

Specifying a very specific font such as "Comic Sans MS" is risky on a web page though, because the device displaying the web page may not have that font installed. The font-name property should hold several font names as a "fallback" system, to ensure maximum compatibility between browsers/operating systems. If the browser does not support the first font, it tries the next font. Start with the font you want, and end with a generic family, to let the browser pick a similar font in the generic family, if no other fonts are available.

Below are some commonly used font combinations, organized by generic family.

Serif Fonts	
Georgia, serif	Example Text
"'Palatino Linotype", "'Book Antiqua", Palatino, serif	Example Text
"Times New Roman", Times, serif	Example Text
Sans-Serif Fonts	
Arial, Helvetica, sans-serif	Example Text
"Arial Black", Gadget, sans-serif	**Example Text**
"Comic Sans MS", cursive, sans-serif	Example Text
Impact, Charcoal, sans-serif	**Example Text**
"Lucida Sans Unicode", "Lucida Grande", sans-serif	Example Text
Tahoma, Geneva, sans-serif	Example Text
"Trebuchet MS", Helvetica, sans-serif	Example Text
Verdana, Geneva, sans-serif	Example Text
Monospace Fonts	
"Courier New", Courier, monospace	`Example Text`
"Lucida Console", Monaco, monospace	`Example Text`

Note that font names which use a space as part of the name, "Time New Roman" for example, are surrounded by quotes, while those that have a single word as a name, Times for example, do not. When specifying strings in the JSON file you must avoid doubled quotes, so substitute an apostrophe, ', for an internal quote symbol. So the DefaultFontName block becomes:

```
DefaultFontName: "'Comic Sans MS', cursive, sans-serif",
```

DefaultTableFont

The same is true of the DefaultTableFont, though in this case you can specify a font size and style, if you want it to vary from the DefaultFontName used in the charts.

```
DefaultTableFont:
   Name: String: "sans-serif"
   Size: double: 14
   Style: String: "PLAIN"
```

where:

Name
Any valid HTML font-family name

Size
Size in points

Style
Use of the style string constants: "Plain", "Normal", "Bold","Italic","Bold Italic"

Example JSON script

```
"DefaultTableFont":
{   "Name": "Times, cursive, sans-serif",
    "Size": 10,
    "Style": "Plain"
},
```

Font names which include embedded spaces, as in 'Comic Sans MS' should be surrounded with apostrophes within the quoted string.

```
"DefaultTableFont":
{   "Name": "'Comic Sans MS', cursive, sans-serif",
    "Size": 10,
    "Style": "Plain"
},
```

DefaultAlarmColors

```
   DefaultAlarmColors
      Target: Color String constant: "GREEN"
      LowAlarm: Color String constant: "BLUE"
      HighAlarm: Color String constant: "RED"
      Trending: Color String constant:  "TAN"
      Stratification: Color String constant   :  "SALMON"
      Alternating: Color String constant "ORANGE"
      HighSpecLimit = Color String constant: "RED"
      LowSpecLimit =Color String constant: "BLUE"
```

The default color choice is completely arbitrary, and everyone seems to like to set their own color combinations. Define your own alarm colors using the JSON construct below.

```
"DefaultAlarmColors"
{
   "Target": "FORESTGREEN",
   "LowAlarm": "ALICEBLUE",
   "HighAlarm": "CRIMSON",
   "Trending": "BROWN",
   "Stratification": "BURLYWOOD",
   "Alternating": "PALEVIOLETRED",
   "HighSpecLimit" : "INDIANRED",
   "LowSpecLimit": "CADETBLUE"
}
```

Include only the items you want to change. The other will remain at their default value. See the appendix for a full list of color constants. A complete list of valid color constants is found in the Appendix.

DefaultChartFonts

```
DefaultChartFonts
   AxisLabelFont
      Name: String: "sans-serif"
      Size: double: 12
      Style: String: "BOLD"
   AxisTitleFont: as above: Name, Size:12, Style: BOLD
   MainTitleFont: as above:  Name, Size:18, Style: BOLD
   SubHeadFont: as above:  Name, Size:14, Style: BOLD
   ToolTipFont: as above:  Name, Size:12, Style: PLAIN
   AnnotationFont: as above:  Name, Size:12, Style: PLAIN
   ControlLimitLabelFont: as above:  Name, Size:12, Style: PLAIN
```

Define your own fonts as below.

```
"DefaultChartFonts": {
    "AxisLabelFont": {
        "Name": "sans-serif",
        "Size": 16,
        "Style": "PLAIN"
    },
    "AxisTitleFont": {
        "Name": "sans-serif",
        "Size": 16,
        "Style": "BOLD"
    }
}
```

AltNofMRule

```
AltNofMRule: boolean: false
```

The AltNofMRule property sets an alternative alarm processing method for those rules which use a N of M control rule, such as most of the named rules sets have for their 2 out of 3 outside of 2-sigma, and 4 out of 5 outside of 1-sigma rules. The default method we use is if that if N of M (2 of 3, or 4 of 5) is out of bounds for the current and previous sample intervals, then the current sample is in alarm, regardless if the current sample is actually within bounds. In the 2 of 3 example, this means that if the previous two sample intervals are out of bounds, but the current sample interval is in-bounds, the current sample fails the 2 of 3 test. The **alternative** is to look at the current sample and if it is not out of bounds, consider the sample interval not in alarm, even if the previous 2 of 2, or 4 of 4 sample intervals are out of bounds.

So, don't set **AltNofMRule** if you want to use the default method. Set it true if you do want to use the alternative alarm processing method.

```
“AltNofMRule”: true,
```

Multi-variable chart features found only in QCSPCChart+

DefaultMultiVariableColors

There are multi-variable charts default properties which you can set, eliminating the need individual colors for each multi-variable plot obects. The DefaultMultiVariableColors property is a Color array, while the others are simple properties.

```
DefaultMultiVariableColors:[SPC Color constant, SPC Color constant, ...]:
                           ["PURPLE", "BROWN"]
```

An array of string color constants, sized to the number of added variables, specifying the color of the added variable plot. The default colors for added variables are: [ORANGE, DARKCYAN, BURLYWOOD, DARKBLUE, YELLOW], but you can change them to anything you want.

DefaultMultiVariableSymbolSize

```
DefaultMultiVariableSymbolSize:  integer: 8
```

An integer specifying the symbol size (1..64) of the added variable plot.

DefaultMultiVariableLineWidth

```
DefaultMultiVariableLineWidth: integer: 1
```

An integer specifying the line width (1..7) of the added variable plot. The default value is 1, in order to differentiate the added variables from the main variable, which has a default line thickness of 3.

DefaultMultiVariablePlotSymbol

```
DefaultMultiVariablePlotSymbol: SPC String constant: "PLUS"
```

A string specifying the symbol shape of the added variable plot. The default value is “SQUARE”, in order to differentiate the added variables from the main variable, which has a default symbol shape of “CIRCLE”. Specify the symbol type using one of the SPC shape string constants: "SQUARE", "UPTRIANGLE", "DOWNTRIANGLE", "DIAMOND", "PLUS", "CROSS", "STAR" and "CIRCLE"

Example

```
"DefaultMultiVariableColors": ["BLACK", "MAGENTA", "BLUE"],
"DefaultMultiVariableSymbolSize": 16,
"DefaultMultiVariableLineWidth": 5,
"DefaultMultiVariablePlotSymbol": "PLUS",
```

5. SPC Initial Chart Setup

- SPCChart
  - InitChartProperties
  - ChartPositioning
  - Scrollbar
  - UseNoTable

This chapter discusses the initial setup of an SPC Chart, which includes all Variable and Attribute control charts supported in the software. This includes the following chart types:

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
  - Xbar-R-MR (Mean-Range-Moving Range Chart)
  - Multi-Variable SPC Charts for :
    - IX-MR, Xbar-R, Xbar-Sigma, Xbar-R-MR, and a Mixed Chart.
- 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)

All of these chart types come in both time-based and batch-based versions.

It does not apply to the setup of frequency histogram charts, or Pareto charts. The setup of those charts are described in other chapters.

InitChartProperties setup

The initial SPC Chart setup should be the first thing your script does, after the initialization of the Canvas, and any static items you set. Using the InitChartProperties JSON object, you specify the SPC chart type, the x-axis mode, the number of samples or categories per sub interval, the number of sample sub intervals per display, and in the case of a time-based control chart, the approximate time between adjacent sample intervals. A typical SPC chart setup for an X-bar R (MEAN_RANGE_CHART) chart looks like this:

```
"StaticProperties":
{
   "Canvas": {
      "Width": 800,
      "Height": 550
   },
},

"SPCChart": {
   "InitChartProperties": {
      "SPCChartType": "MEAN_RANGE_CHART",
      "ChartMode": "Time",
      "NumSamplesPerSubgroup": 5,
      "NumDatapointsInView": 12,
      "TimeIncrementMinutes": 15
   },
```

A typical setup for a fixed type multi-variable chart (QCSPCChart+), will need specify the number of added variables, using the property *NumberAddedVariables*:

```
"InitChartProperties": {
  "SPCChartType": "MULTIVARIABLE_MEAN_RANGE_CHART",
  "ChartMode": "BATCH",
  "NumSamplesPerSubgroup": 5,
  "NumDatapointsInView": 12,
  "TimeIncrementMinutes": 15,
  "NumberAddedVariables": 2
},
```

The multi-variable mixed plot (QCSPCChart+) is the most complicated, and it adds an additional two properties (*MultiVariableMixedPrimaryChartType* and *MultiVariableMixedChartSubTypes*) to the setup above:

```
"InitChartProperties": {
  "SPCChartType": "MULTIVARIABLE_MIXED_CHART",
      "ChartMode": "BATCH",
      "NumSamplesPerSubgroup": 5,
      "NumDatapointsInView": 12,
      "TimeIncrementMinutes": 15,
      "NumberAddedVariables": 2,
      "MultiVariableMixedPrimaryChartType": "MEAN_RANGE_CHART",
      "MultiVariableMixedChartSubTypes": [
        "MEAN_SIGMA_CHART",
        "INDIVIDUAL_RANGE_CHART"
      ]
    },
```

Note that InitChartProperties falls under SPCChart, and the SPCChart is after Canvas (for initial chart sizing), and the SPCChartStrings initialization (an initializer of static string properties). You must maintain this order when initializing any of the charts. The Canvas and SPCChartStrings blocks only need to be processed once, so if you have other JSON chart definitions in the same file, or elsewhere in same web page, and you want to maintain the same Canvas and SPCChartStrings, you do not need to initialize them again. But if you do change them, the change will affect all charts, not just the one they are attached to.

The JSON objects under InitChartProperties can have the following values:

SPCChartType

The SPC chart type parameter. Use one of the string constants strings: MEAN_RANGE_CHART, MEDIAN_RANGE_CHART, MEAN_SIGMA_CHART, MEAN_SIGMA_CHART_VSS, INDIVIDUAL_RANGE_CHART, EWMA_CHART, MA_CHART, MAMR_CHART, MAMS_CHART, MEAN_RANGE_MR_CHART, LEVEY_JENNINGS_CHART and TABCUSUM_CHART,

or for the Multi-Variable chart types (QCSPCChart+), use one of the string constants: MULTIVARIABLE_MEAN_RANGE_CHART, MULTIVARIABLE_MEAN_SIGMA_CHART, MULTIVARIABLE_INDIVIDUAL_RANGE_CHART, MULTIVARIABLE_MEAN_RANGE_MR_CHART, and MULTIVARIABLE_MIXED_CHART

or use one of the Attribute control chart types:

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.

Note: Since the MULTIVARIABLE_MIXED_CHART does not actually specify the chart type used for the main plot, and added multi-variable plots, these are specified in the *MultiVariableMixedPrimaryChartType* and *MultiVariableMixedChartSubTypes* blocks.

ChartMode

Specifies if the x-axis is time-based (Time), or batch-base (Batch). Use the string constant string Time or Batch.

NumCategories

In an Attribute Control Charts this value represents the number of defect categories used to determine defect counts. Specify a numeric value, no quotes. Since the example above is for a Variable Control Chart (MEAN_RANGE_CHART), the NumCategories property does not need to be set.

NumSamplesPerSubgroup

Specifies the number of samples that make up a sample subgroup. If the SPCChartType is one of the variable sample size chart types, this value must be the maximum number of samples per subgroup. Specify a numeric value, no quotes.

NumDatapointsInView

Specifies the number of sample subgroups displayed in the graph at one time. Specify a numeric value, no quotes.

TimeIncrementMinutes

Specifies the approximate time increment (in minutes) between adjacent subgroup samples. This applies only to the Time ChartMode. Specify a numeric value, no quotes. Can be a double (0.5) to specify a fraction of a minute.

The following parameters only apply to CusSum charts.

CuSumKValue

A CuSum charts K value

CuSumHValue

A CuSum charts H value

CuSumMeanValue

A CuSum charts mean value

The following three properties apply to Multi-Variable Chart Types (QCSPCChart+). The Multi-Variable Chart have the SPCChartType set to one of the following five string constants: MULTIVARIABLE_MEAN_RANGE_CHART, MULTIVARIABLE_MEAN_SIGMA_CHART, MULTIVARIABLE_INDIVIDUAL_RANGE_CHART, MULTIVARIABLE_MEAN_RANGE_MR_CHART, and MULTIVARIABLE_MIXED_CHART

NumberAddedVariables

A multi-variable chart type has one main chart variable, and one or more added variables. The value of the *NumberAddedVariables* property specifies the number of added variables. So if you have the main variable, and two added variables, the *NumberAddedVariables* should be set to 2. The total number of variables in the chart will be three (the main, plus two added variables).

MultiVariableMixedPrimaryChartType

The MULTIVARIABLE_MIXED_CHART does not actually specify the chart type used for the main plot, and added multi-variable plots. Specify the chart type of the main plot by setting the MultiVariableMixedPrimaryChartType to one of the string constants: MEAN_RANGE_CHART, MEAN_SIGMA_CHART, or INDIVIDUAL_RANGE_CHART.

MultiVariableMixedChartSubTypes

The MULTIVARIABLE_MIXED_CHART does not actually specify the chart type used for the added multi-variable plots. Specify the added chart types using an array of strings, where the elements are the string constants: MEAN_RANGE_CHART, MEAN_SIGMA_CHART, or INDIVIDUAL_RANGE_CHART. You only specify the added chart types, the main chart type is specified using the MultiVariableMixedPrimaryChartType property.

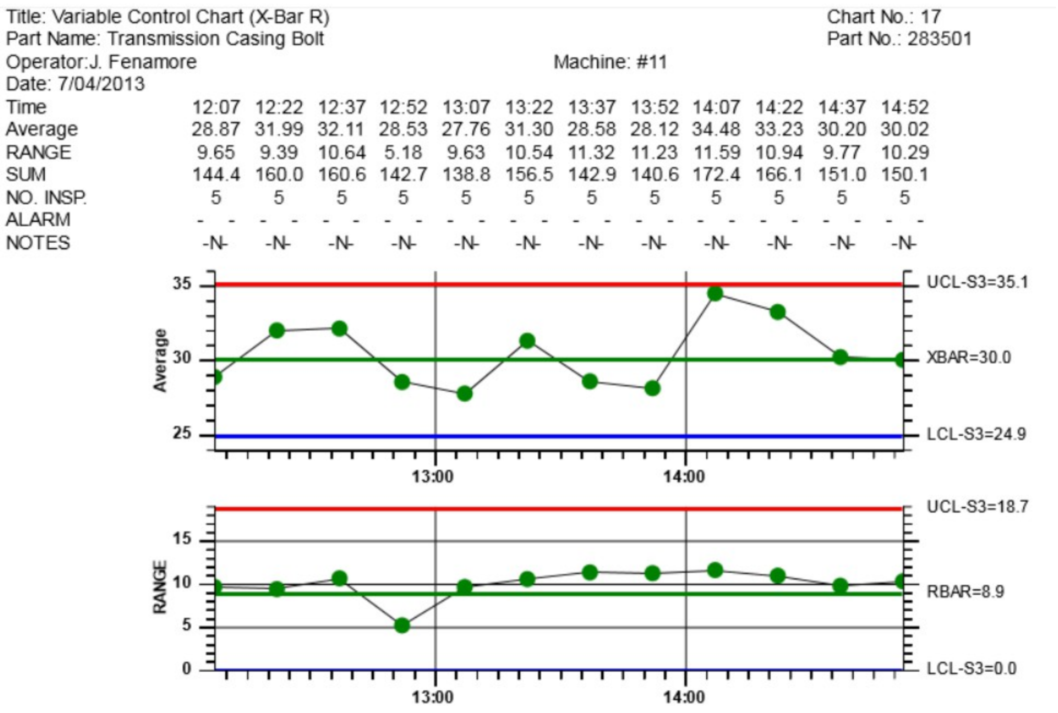
Examples of Typical Chart Setups

Time-based Mean-Range (Xbar-R) Chart

A chart setup with these parameters

```
"SPCChart": {
  "InitChartProperties": {
     "SPCChartType": "MEAN_RANGE_CHART",
     "ChartMode": "Time",
     "NumSamplesPerSubgroup": 5,
     "NumDatapointsInView": 12,
     "TimeIncrementMinutes": 15
  },
```

would look something like this:



Title: Variable Control Chart (X-Bar R) Chart No.: 17
Part Name: Transmission Casing Bolt Part No.: 283501
Operator:J. Fenamore Machine: #11
Date: 7/04/2013

Time	12:07	12:22	12:37	12:52	13:07	13:22	13:37	13:52	14:07	14:22	14:37	14:52
Average	28.87	31.99	32.11	28.53	27.76	31.30	28.58	28.12	34.48	33.23	30.20	30.02
RANGE	9.65	9.39	10.64	5.18	9.63	10.54	11.32	11.23	11.59	10.94	9.77	10.29
SUM	144.4	160.0	160.6	142.7	138.8	156.5	142.9	140.6	172.4	166.1	151.0	150.1
NO. INSP.	5	5	5	5	5	5	5	5	5	5	5	5
ALARM	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -
NOTES	-N-	-N-	-N-	-N-	-N-	-N-	-N-	-N-	-N-	-N-	-N-	-N-

Note the following items:

SPCChartType

The chart type is a MEAN_RANGE_CHART, also known as an X-bar R Chart.

ChartMode

The x-axis uses a time scale.

NumSamplesPerSubgroup

The NO. INSP. row is always 5, since a MEAN_RANGE_CHART requires a fixed sample size per sample interval.

NumDatapointsInView

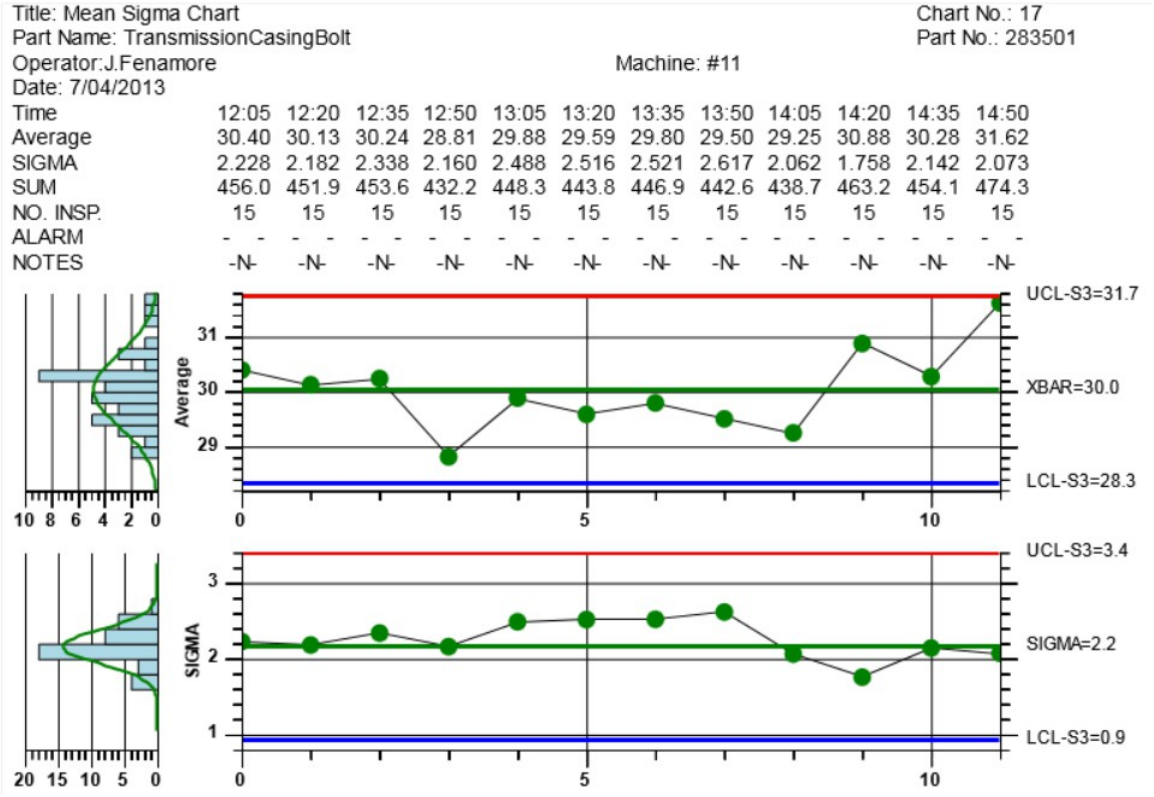
The number of data points in each chart is 12, as is the number of data columns in the table

TimeIncrementMinutes

The time interval between adjacent samples table is 15 minutes.

Example for an Batch-mode Mean Sigma chart (X-bar Sigma)

```
"InitChartProperties": {
  "SPCChartType": "MEAN_SIGMA_CHART",
  "ChartMode": "Batch",
  "NumSamplesPerSubgroup": 15,
  "NumDatapointsInView": 12,
  "TimeIncrementMinutes": 15
},
```



Title: Mean Sigma Chart
Part Name: TransmissionCasingBolt
Operator:J.Fenamore
Date: 7/04/2013

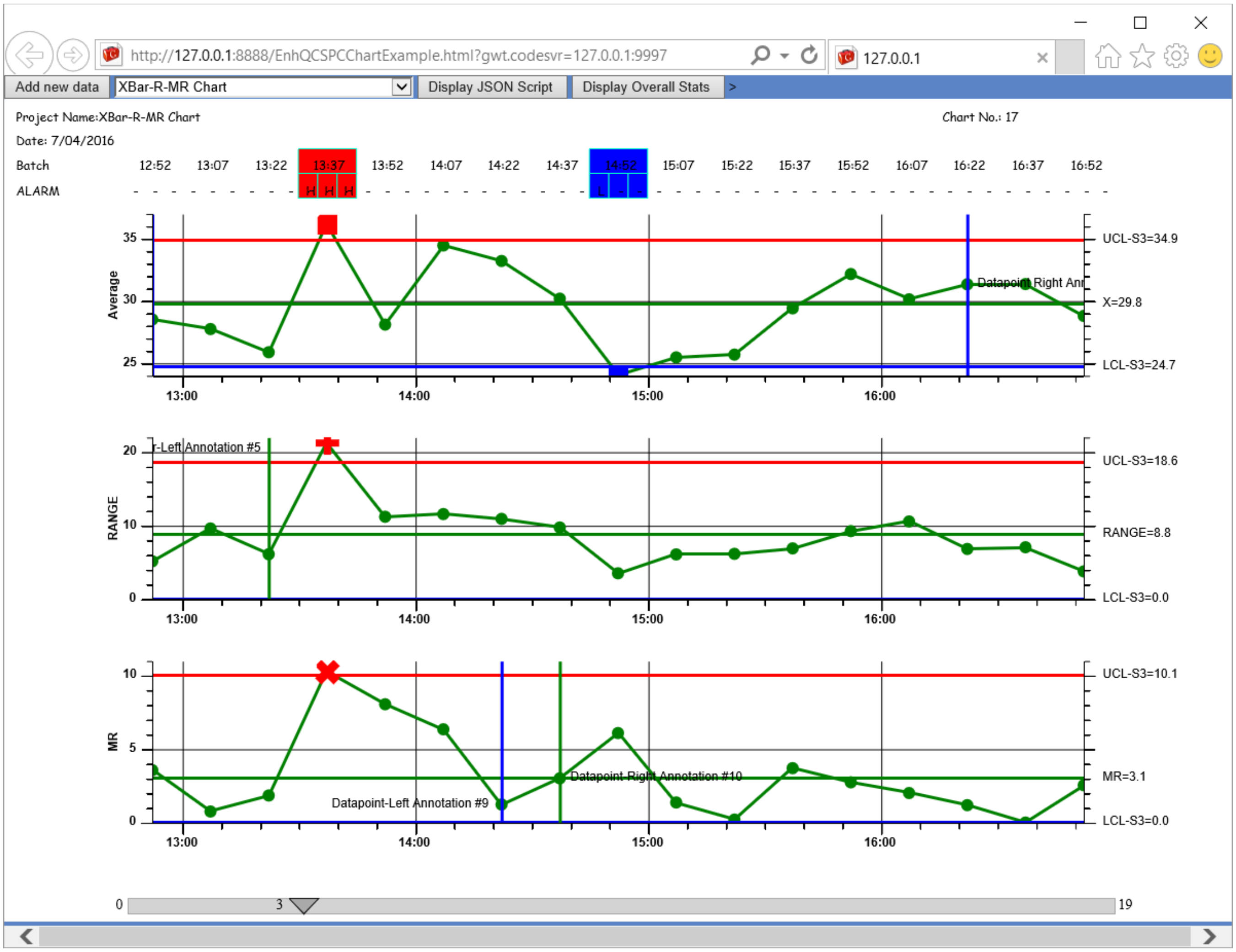
Chart No.: 17
Part No.: 283501

Machine: #11

Time	12:05	12:20	12:35	12:50	13:05	13:20	13:35	13:50	14:05	14:20	14:35	14:50
Average	30.40	30.13	30.24	28.81	29.88	29.59	29.80	29.50	29.25	30.88	30.28	31.62
SIGMA	2.228	2.182	2.338	2.160	2.488	2.516	2.521	2.617	2.062	1.758	2.142	2.073
SUM	456.0	451.9	453.6	432.2	448.3	443.8	446.9	442.6	438.7	463.2	454.1	474.3
NO. INSP.	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-

Example for an Batch-mode Mean-Range-MR chart (X-R-MR)

```
        "InitChartProperties": {
           "SPCChartType": "MEAN_RANGE_MR_CHART",
           "ChartMode": "TIME",
           "NumSamplesPerSubgroup": 5,
           "NumDatapointsInView": 17,
           "TimeIncrementMinutes": 15
        },
```



Project Name:XBar-R-MR Chart

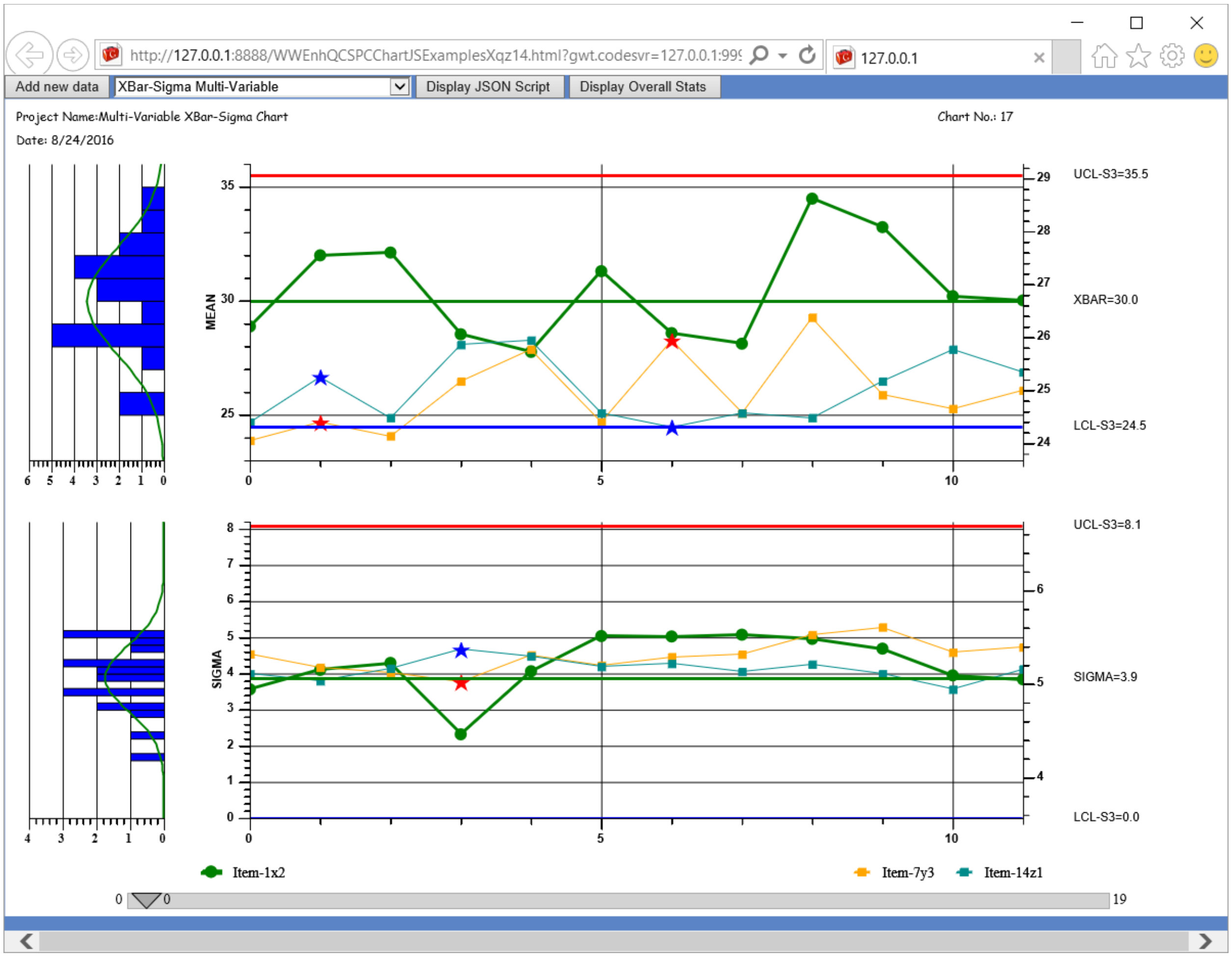
Chart No.: 17

Date: 7/04/2016

Example for an Batch-mode Multi-Variable Mean-Sigma (Xbar-Sigma) chart (QCSPCChart+)

```
"SPCChart": {
  "InitChartProperties": {
    "SPCChartType": "MULTIVARIABLE_MEAN_SIGMA_CHART",
    "ChartMode": "BATCH",
    "NumSamplesPerSubgroup": 5,
    "NumDatapointsInView": 12,
    "TimeIncrementMinutes": 15,
    "NumberAddedVariables": 2
  },
```

This multi-variable chart plots multiple Mean-Sigma plots in the same chart: one main and two added variables, for a total of three variables.



Project Name:Multi-Variable XBar-Sigma Chart

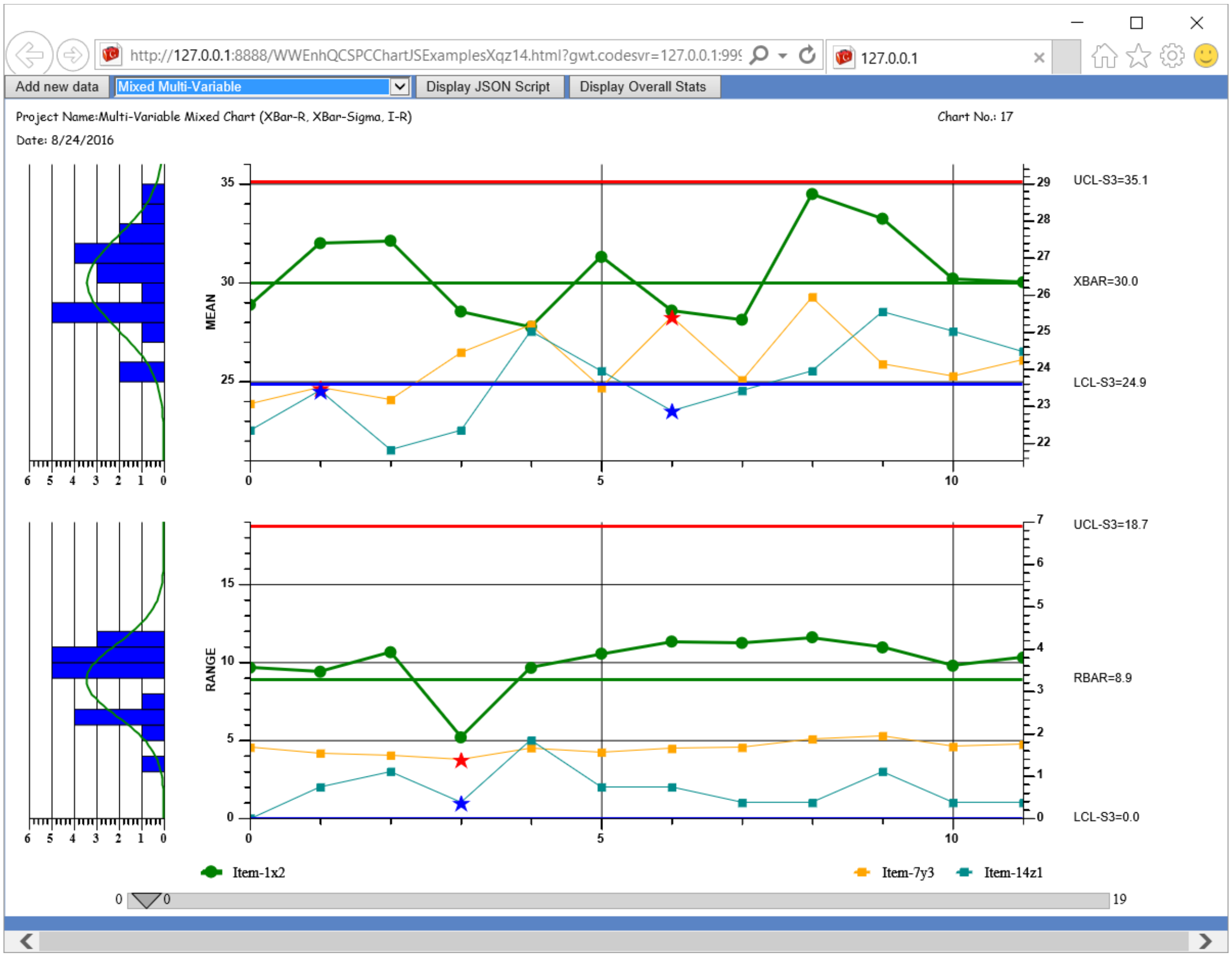
Date: 8/24/2016

Chart No.: 17

Example for an Batch-mode Multi-Variable Mixed Chart (QCSPCChart+)

```
  "SPCChart": {
    "InitChartProperties": {
      "SPCChartType": "MULTIVARIABLE_MIXED_CHART",
      "ChartMode": "BATCH",
      "NumSamplesPerSubgroup": 5,
      "NumDatapointsInView": 12,
      "TimeIncrementMinutes": 15,
      "NumberAddedVariables": 2,
      "MultiVariableMixedPrimaryChartType": "MEAN_RANGE_CHART",
      "MultiVariableMixedChartSubTypes": [
        "MEAN_SIGMA_CHART",
        "INDIVIDUAL_RANGE_CHART"
      ]
    },
```

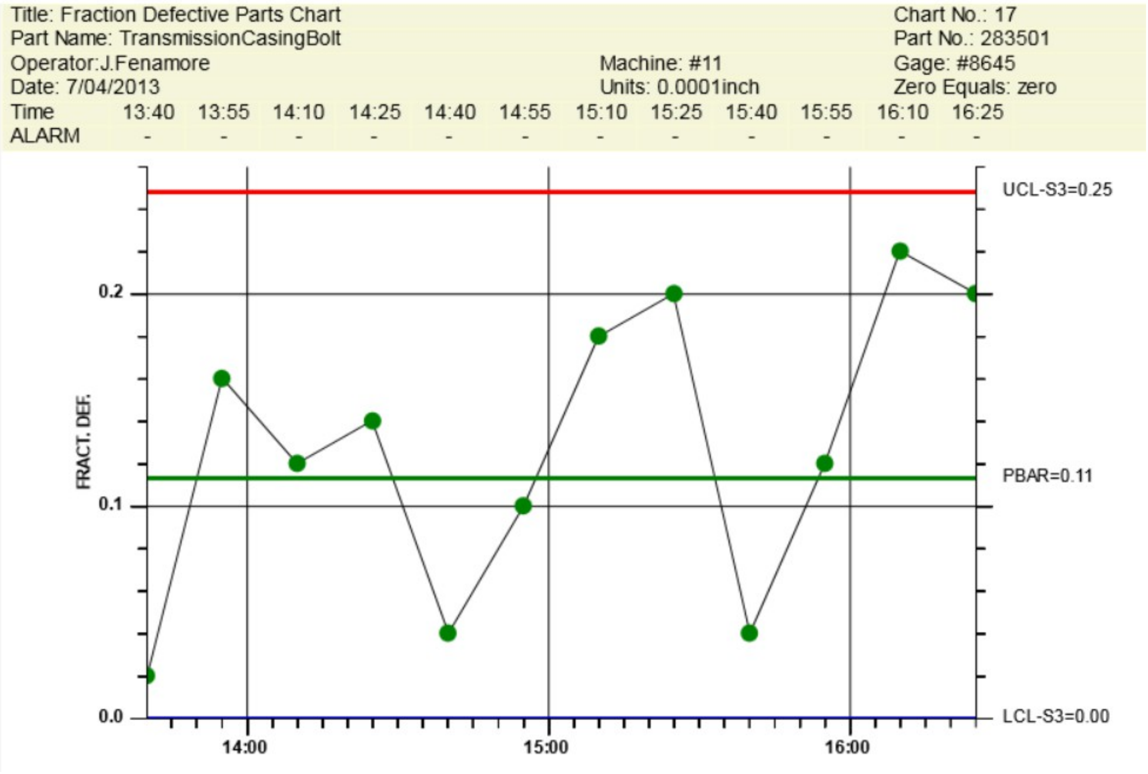
This multi-variable chart plots a Mean-Range chart as the main variable, and a mean-Sigma plot and an Individual-Range plot as the two added variables, for a total of three variables.



Example for an time-based Fraction Defective Parts chart

```
"InitChartProperties": {
  "SPCChartType": "FRACTION_DEFECTIVE_PARTS_CHART",
  "ChartMode": "Time",
  "NumCategories": 5,
  "NumSamplesPerSubgroup": 50,
  "NumDatapointsInView": 12,
  "TimeIncrementMinutes": 15
},
```

There are many minor variants of this basic structure. See the example JSON scripts for the example closest to your application.



Title: Fraction Defective Parts Chart | Chart No.: 17
Part Name: TransmissionCasingBolt | Part No.: 283501
Operator:J.Fenamore | Machine: #11 | Gage: #8645
Date: 7/04/2013 | Units: 0.0001inch | Zero Equals: zero

Time	13:40	13:55	14:10	14:25	14:40	14:55	15:10	15:25	15:40	15:55	16:10	16:25
ALARM	-	-	-	-	-	-	-	-	-	-	-	-

Chart and Table Positioning

The position of the table and the charts are interrelated. The general algorithm is that the table takes precedence. The SPC display will try and utilize the entire area of the parent window. With respect to the y-dimension, the table will take as much vertical room in the window as it needs to display the specified number of rows, given the specified text size. Whatever vertical real-estate left over in the window is used to display the one or two charts which follow. There are also constraints on the width of the table, because most of the table is divided into columns, corresponding to the discrete sample intervals of the chart. Make the table width too small, and it will be hard to fit the desired number of columns, and still have the text readable. There are a handful of properties which control the positioning parameters, within these constraints.

The position of the table, and the SPC charts in the display window is controlled by the ChartPositioning object.

```
SPCChart
   ChartPositioning
      GraphStartPosX: double: 0.15
      GraphStopPosX: double:  0.8
      TableStartPosY: double: 0.0
      GraphTopTableOffset : double: 0.02
      InterGraphMargin: double: 0.075
      GraphBottomPos: double: 0.90
      BottomLabelMargin: double: 0.0
```

GraphStartPosX
Specifies the left edge, using normalized coordinates, of the plotting area for both primary and secondary charts

GraphStopPosX
Specifies the right edge, using normalized coordinates, of the plotting area for both primary and secondary charts

TableStartPosY
Specifies the top edge, using normalized coordinates, of the SPC chart table

GraphTopTableOffset
Specifies the offset of the top of the primary chart from the bottom of the data table, using normalized coordinates

GraphBottomPos
Specifies the bottom edge, using normalized coordinates, of the plotting area for the secondary chart

InterGraphMargin
Specifies the margin, in normalized coordinates, between the primary and seconday charts

BottomLabelMargin
Specifies an additional margin, in normalized coordinates, if only the primary graphs is displayed, allowing for the x-axis labels

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.

```
"ChartPositioning": {
   "GraphStartPosX": 0.1,
   "GraphStopPosX" : 0.875
},
```

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:

```
"ChartPositioning": {
   "TableStartPosY" : 0.00,
   "GraphTopTableOffset" : 0.02,
   "GraphBottomPos" : 0.925
},
```

Scrollbar

```
Scrollbar
   EnableScrollBar: boolean: true
   ScrollbarPosition:SPC String constants: "SCROLLBAR_POSITION_MIN"
   ScrollbarValue: double: 0
   EnableZoomToggle: boolean: true
```

Normally, you have more data than can fit on the screen at once. In order to view the unseen data, a scroll bar is used at the bottom of the chart area to scroll left or right. The scrollbar will appear by default if there are more data points than the NumDatapointsInView property. You can change the default behavior by explicitly turning the scroll bar on and off. You can also set the initial value of the scroll bar so some known value, using the ScrollbarValue property, or you can force the go to the maximum value of the scroll bar after any data updates. That way the most recent data will always be in view. Or, you can specify that after a RebuildUsingCurrentData, which usually increases the scroll bars range of values, that the scrollbar position to show the must recently added data ("SCROLLBAR_POSITION_MAX"), or the oldest data ("SCROLLBAR_POSITION_MIN"). If you want the scrollbar to remain at the same position after an update use the "SCROLLBAR_POSITION_UNCHANGED" constant.

EnableScrollBar

Set to true or false to enable/disable the scroll bar.

ScrollbarPosition

Set to the string constant value "SCROLLBAR_POSITION_MIN", "SCROLLBAR_POSITION_MAX" or "SCROLLBAR_POSITION_UNCHANGED". The value "SCROLLBAR_POSITION_MIN" forces the scroll bar to remain at it's minimum position (oldest data) after any updates. The value "SCROLLBAR_POSITION_MAX" is the opposite of "SCROLLBAR_POSITION_MIN", and it will force the scroll bar to its maximum position (newest data) after any updates. And the "SCROLLBAR_POSITION_UNCHANGED" value will leave the scroll bar positioned at its current index.

ScrollbarValue

Ues this property to set the scroll bar value to a specified index value. The index value reflects the starting index of the data in the chart, corresponding to the left-most point in the current chart view.

Enable scroll bar and set its position to `SCROLLBAR_POSITION_MAX`

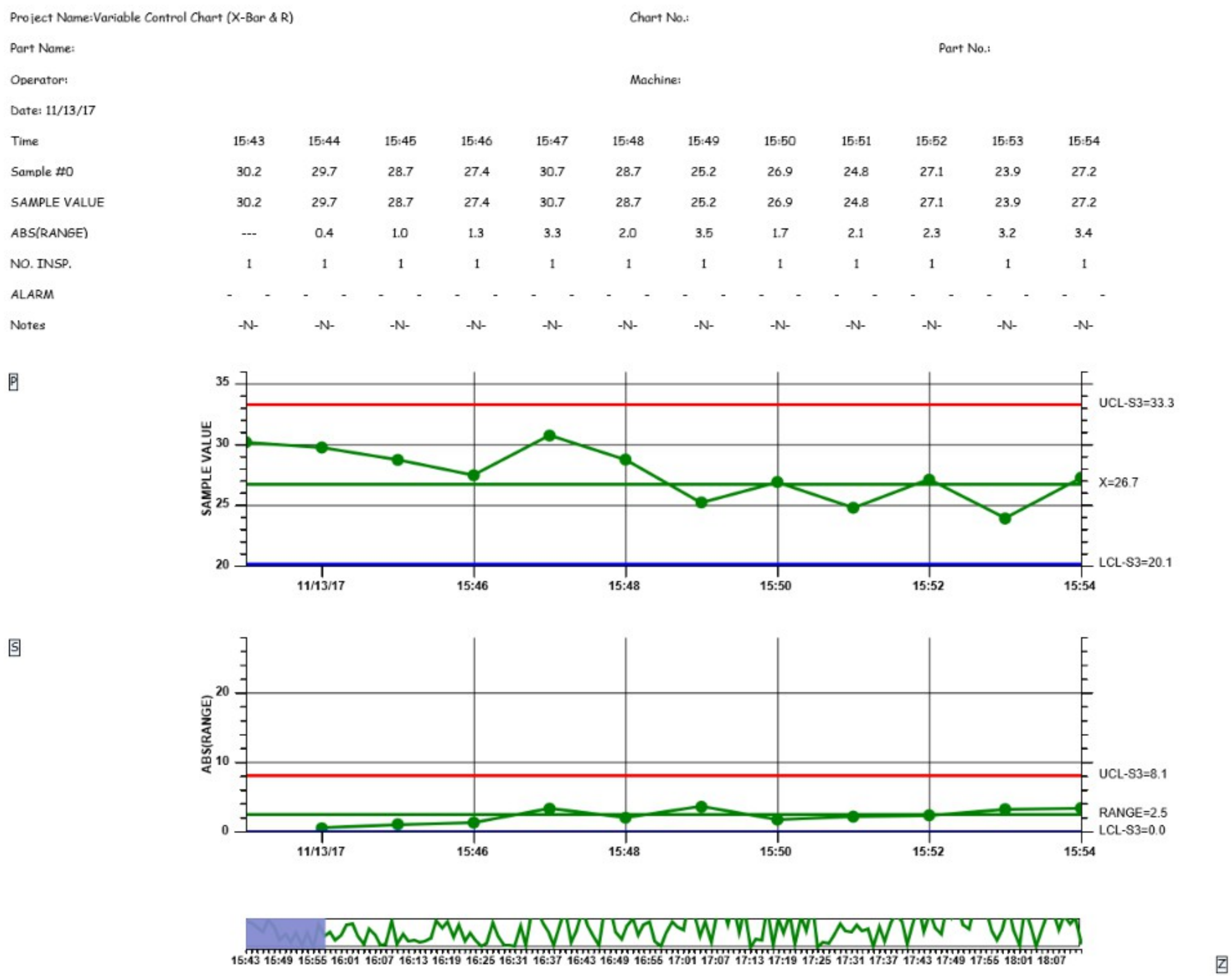
```
"Scrollbar": {
   "EnableScrollBar": true,
   "ScrollbarPosition": "SCROLLBAR_POSITION_MAX"
},
```

Enable scroll bar and set its position to the chart data index 123.

```
"Scrollbar": {
   "EnableScrollBar": true,
   "ScrollbarValue": 123
},
```

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.



Project Name:Variable Control Chart (X-Bar & R) Chart No.:

Part Name: Part No.:

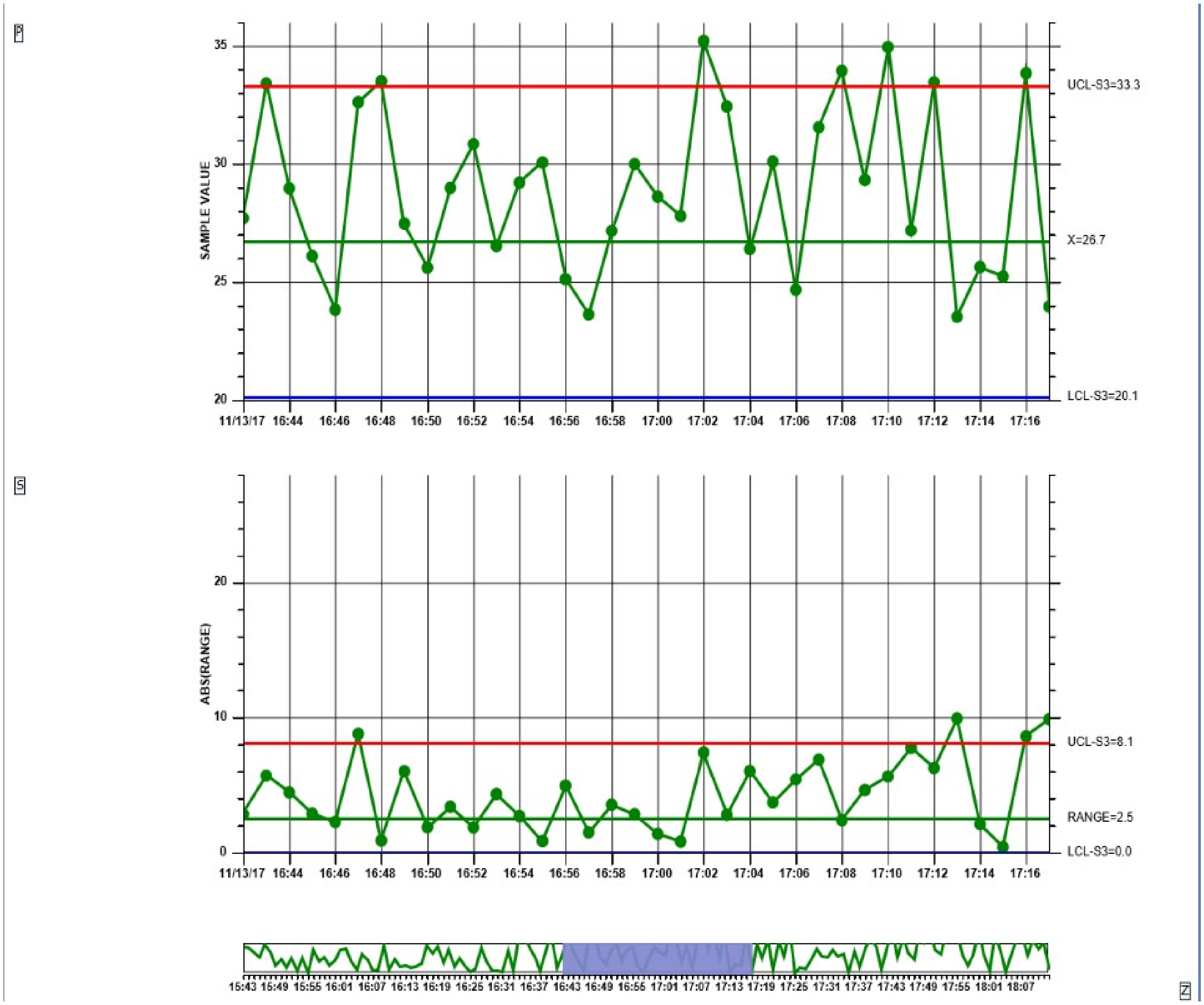
Operator: Machine:

Date: 11/13/17

Time	15:43	15:44	15:45	15:46	15:47	15:48	15:49	15:50	15:51	15:52	15:53	15:54
Sample #0	30.2	29.7	28.7	27.4	30.7	28.7	25.2	26.9	24.8	27.1	23.9	27.2
SAMPLE VALUE	30.2	29.7	28.7	27.4	30.7	28.7	25.2	26.9	24.8	27.1	23.9	27.2
ABS(RANGE)	---	0.4	1.0	1.3	3.3	2.0	3.5	1.7	2.1	2.3	3.2	3.4
NO. INSP.	1	1	1	1	1	1	1	1	1	1	1	1
ALARM	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -
Notes	-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 EnableZoomToggle property false.

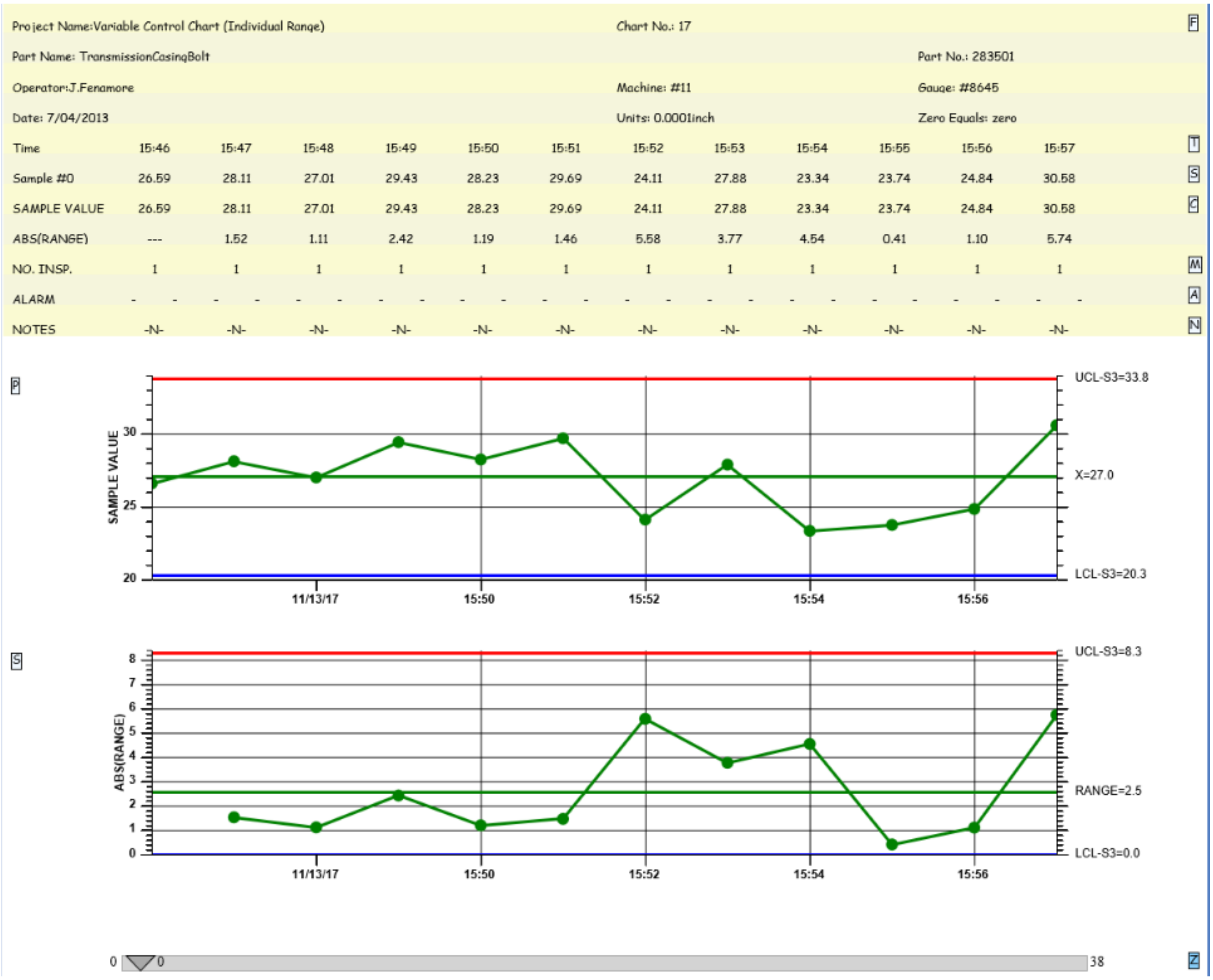
.

```
“EnableZoomToggle”: false
```

Since the zoom toggle is on by default all of the examples 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.



Project Name:Variable Control Chart (Individual Range) Chart No.: 17

Part Name: TransmissionCasingBolt Part No.: 283501

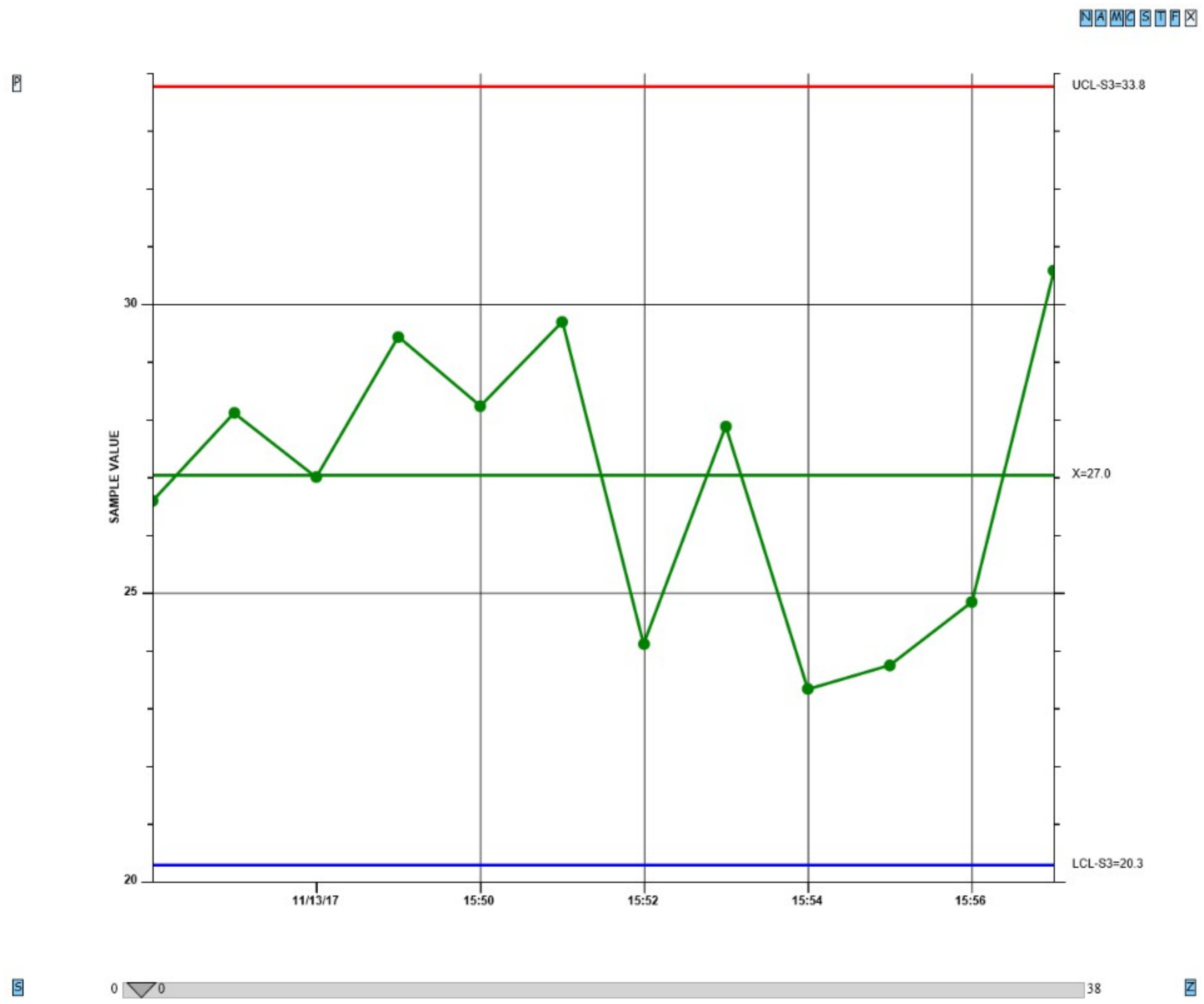
Operator:J.Fenamore Machine: #11 Gauge: #8645

Date: 7/04/2013 Units: 0.0001inch Zero Equals: zero

Time	15:46	15:47	15:48	15:49	15:50	15:51	15:52	15:53	15:54	15:55	15:56	15:57
Sample #0	26.59	28.11	27.01	29.43	28.23	29.69	24.11	27.88	23.34	23.74	24.84	30.58
SAMPLE VALUE	26.59	28.11	27.01	29.43	28.23	29.69	24.11	27.88	23.34	23.74	24.84	30.58
ABS(RANGE)	---	1.52	1.11	2.42	1.19	1.46	5.58	3.77	4.54	0.41	1.10	5.74
NO. INSP.	1	1	1	1	1	1	1	1	1	1	1	1
ALARM	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -
NOTES	-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.

```
“PrimaryChartSetup”: {
   “EnableChartDisplayToggles”: false
}
```

Hide the table buttons on the right.

```
"TableSetup": {
          “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.

```
“MiscChartDataProperties”: {
         “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. Note that each toggle is in a different block.

```
“MiscChartDataProperties”: {
         “EnableDisplayOptionToggles”: false
}

"TableSetup": {
          “EnableTableRowToggles”: false
}

“PrimaryChartSetup”: {
   “EnableChartDisplayToggles”: false
}

“SPCChart”: {
   “Scrollbar”: {
```

```
            “EnableZoomToggle”: true
    }
}
```

,

UseNoTable

A common option is to remove the table from the display; either you don't need the table data, since it is redundant with much of the information displayed in the chart, or you want to plot many more data points than the table permits. Since the table displays the sample interval data in columnar format, the number of data points in the chart need to match the number of columns in the table. This restricts the number of data points you can display in the graph to something in the range of 10-20, i.e. the number of columns which will fit on the screen. Eliminate the table and you can fit 100's of sample interval data points on the screen at once. Only use property if you want the chart to not have a table. Otherwise leave it out of the JSON script entirely.

```
UseNoTable
   PrimaryChart: boolean: true
   SecondaryChart: boolean: true
   Histograms: boolean: true
   Title: String: ""
```

PrimaryChart

Set to true to display the primary chart.

SecondaryChart

Set to true to display the secondary chart.

Histograms

Set to true to display the histograms to the left of each chart.

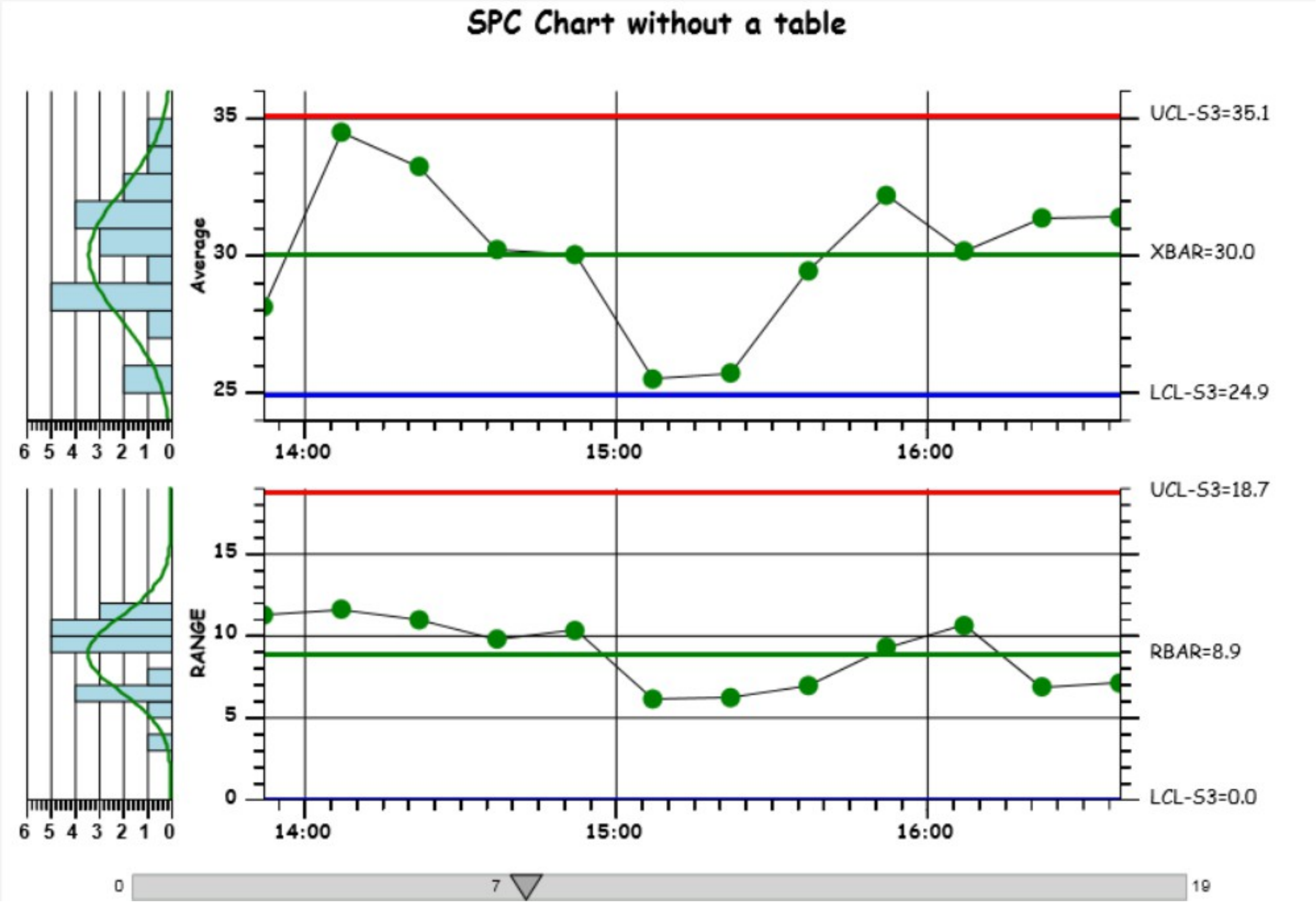
Title

Specify a string title to display above the graphs.

When using UseNoTable, do NOT use a TableSetup block in your JSON script.

Example

```
    "UseNoTable": {
       "PrimaryChart": true,
       "SecondaryChart": true,
       "Histograms": true,
        "Title": "SPC Chart without a table"
    },
```



MiscChartDataProperties

```
MA_W: integer: 5
EWMA_Lambda: double:  0.2
EWMA_UseSSLimits: boolean: false
EWMA_StartingValue: double: 1.0
DefaultControlLimitSigma: double: 3
AutoLogAlarmsAsNotes: boolean: false
AlarmTimeFormatString: String: "EEE, d MMM yyyy HH:mm:ss Z"
DefectOpportunitiesPerUnit: double: 1
NotesReadOnly: boolean: false
AlarmReportMode: SPC String constant: "REPORT_ALL_ALARMS"
EnableDisplayOptionsToggles: boolean: true
AddNote
    Index: integer: 0
    Note: string: ""
    Append: boolean: false
```

Legend chart features found only in QCSPCChart+

```
Legend
   EnableLegend: boolean: true
   LegendStrings: [string, string, ...] : ["Item-1", "Item-2", "Item-3"] ,...]
      LegendFont:
         Name: String: "sans-serif"
         Size: integer: 18
         Style: SPC String constant: "Plain"
```

Multi-Varaible chart features found only in QCSPCChart+

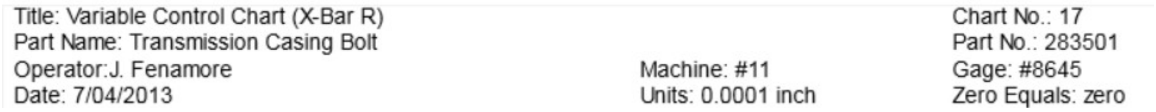
```
MultiVariableItems
   YaxisMapping:  [SPC String constant, SPC String constant, ...]:
                    ["LEFT_AXIS","LEFT_AXIS"]
   MultiVariableSymbolSize: [ integer, integer,...]: [16, 12]
   MultiVariableLineWidth:: [ integer, integer,...]: [2, 3]
   MultiVariablePlotSymbol:[SPC String constant,SPC String constant, ...]:
                    ["SQUARE", "SQUARE"],
   MultiVariableColors:[SPC Color constant, SPC Color constant, ...]:
                          ["PURPLE", "BROWN"]
```

6. SPC Data Table Setup

```
SPCChart

 TableSetup
    HeaderStringsLevel: SPC String constant: "HEADER_STRINGS_LEVEL2"
    EnableInputStringsDisplay: boolean: true
    EnableCategoryValues: boolean: true
    EnableSampleValues: boolean: true
    EnableCalculatedValues: boolean: true
    EnableProcessCapabilityValues: boolean: true
    EnableTotalSamplesValues: boolean: true
    EnableNotes: boolean: true
    EnableTimeValues: boolean: true
    EnableDataToolTip: boolean: true
    EnableNotesToolTip: boolean: true
    EnableTableRowToggles: boolean: false
    NotesToolTip
       ButtonMask: SPC String constant: "BUTTON1_MASK"
       ToolTipMode: SPC String constant: "MOUSETOGGLE_TOOLTIP"
       NotesReadOnly: boolean: false
    TableBackgroundMode:SPC String Constant: "TABLE_SINGLE_COLOR_BACKGROUND"
    TableAlarmEmphasisMode: SPC String Constant: "ALARM_HIGHLIGHT_NONE"
    ChartAlarmEmphasisMode: SPC String Constant: "ALARM_HIGHLIGHT_SYMBOL"
    BackgroundColor1: Color String constant: "WHITE"
    ChartData
    EnableTableRowToggles:  boolean:false
```

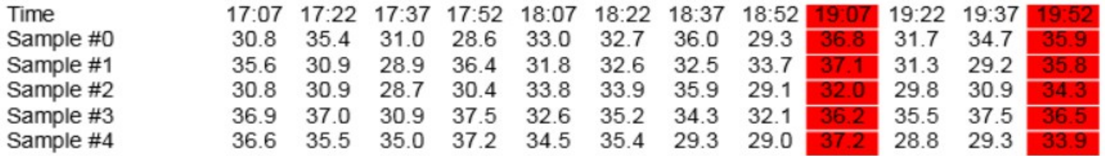
The SPC Data Table is a table which appears above the actual SPC charts. It is designed to be a generic form for the display of run specific information for an SPC Chart. The top most part of the table is the header, where items such as the chart Title, Part Number, Part Name, etc., are displayed. The second part of the table displays numeric data values for each sample interval of the chart, including the raw sample values, and the calculated values of the chart (mean, range, sum, sigma, etc.). The third part of the table is status information, display the current alarm status, and any notes assocated with the sample interval. All parts of the table are optional. You can minimize the table, or skip it entirely, using the various options.

Header Information



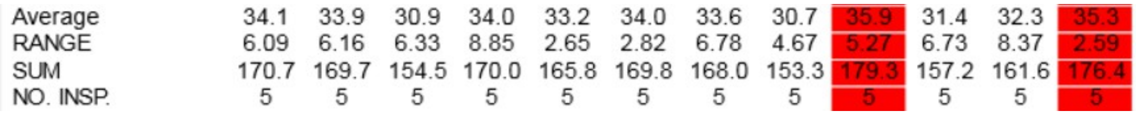
Title: Variable Control Chart (X-Bar R)		Chart No.: 17
Part Name: Transmission Casing Bolt		Part No.: 283501
Operator:J. Fenamore	Machine: #11	Gage: #8645
Date: 7/04/2013	Units: 0.0001 inch	Zero Equals: zero

Raw Sample Data



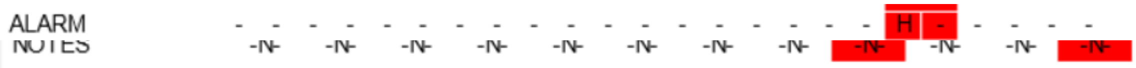
Time	17:07	17:22	17:37	17:52	18:07	18:22	18:37	18:52	19:07	19:22	19:37	19:52
Sample #0	30.8	35.4	31.0	28.6	33.0	32.7	36.0	29.3	36.8	31.7	34.7	35.9
Sample #1	35.6	30.9	28.9	36.4	31.8	32.6	32.5	33.7	37.1	31.3	29.2	35.8
Sample #2	30.8	30.9	28.7	30.4	33.8	33.9	35.9	29.1	32.0	29.8	30.9	34.3
Sample #3	36.9	37.0	30.9	37.5	32.6	35.2	34.3	32.1	36.2	35.5	37.5	36.5
Sample #4	36.6	35.5	35.0	37.2	34.5	35.4	29.3	29.0	37.2	28.8	29.3	33.9

Calculated Data Values



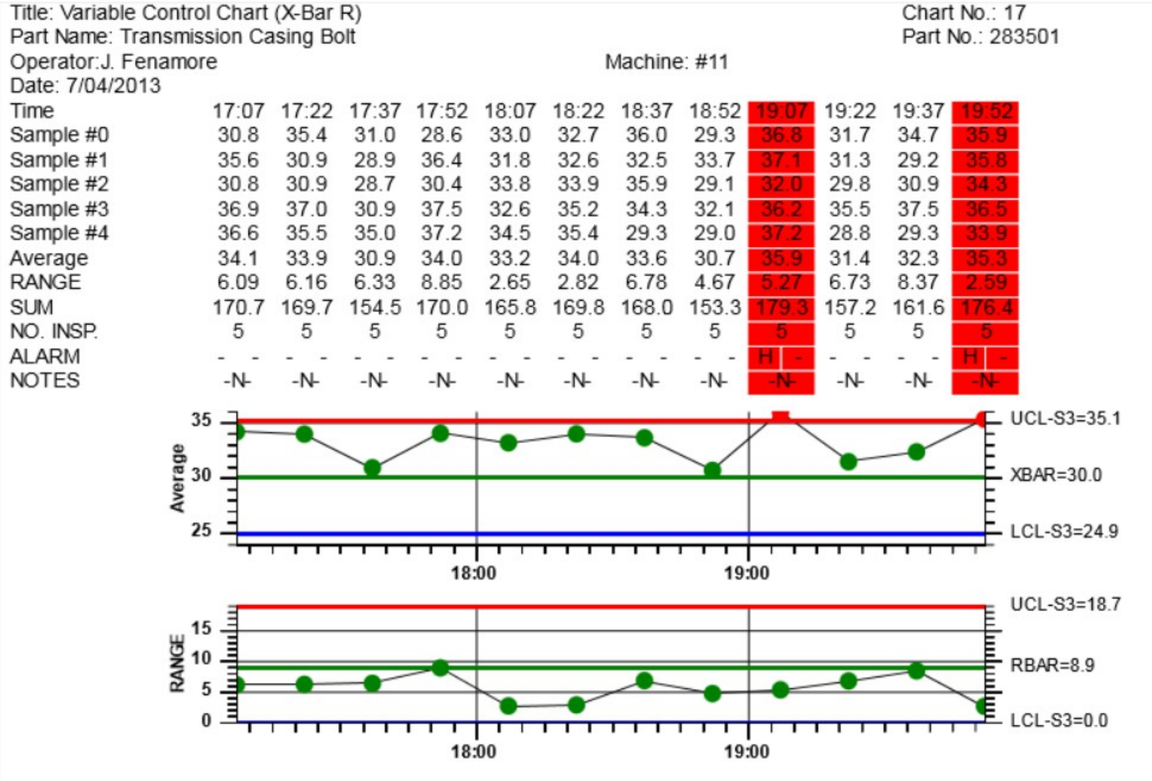
Average	34.1	33.9	30.9	34.0	33.2	34.0	33.6	30.7	35.9	31.4	32.3	35.3
RANGE	6.09	6.16	6.33	8.85	2.65	2.82	6.78	4.67	5.27	6.73	8.37	2.59
SUM	170.7	169.7	154.5	170.0	165.8	169.8	168.0	153.3	179.3	157.2	161.6	176.4
NO. INSP.	5	5	5	5	5	5	5	5	5	5	5	5

Status



ALARM	- -	- -	- -	- -	- -	- -	- -	- -	- -	H -	- -	- -
NOTES	-N-	-N-	-N-	-N-	-N-	-N-	-N-	-N-	-N-	-N-	-N-	-N-

Put it all together and it would look something like:



Title: Variable Control Chart (X-Bar R) Chart No.: 17
Part Name: Transmission Casing Bolt Part No.: 283501
Operator:J. Fenamore Machine: #11
Date: 7/04/2013

Time	17:07	17:22	17:37	17:52	18:07	18:22	18:37	18:52	19:07	19:22	19:37	19:52
Sample #0	30.8	35.4	31.0	28.6	33.0	32.7	36.0	29.3	36.8	31.7	34.7	35.9
Sample #1	35.6	30.9	28.9	36.4	31.8	32.6	32.5	33.7	37.1	31.3	29.2	35.8
Sample #2	30.8	30.9	28.7	30.4	33.8	33.9	35.9	29.1	32.0	29.8	30.9	34.3
Sample #3	36.9	37.0	30.9	37.5	32.6	35.2	34.3	32.1	36.2	35.5	37.5	36.5
Sample #4	36.6	35.5	35.0	37.2	34.5	35.4	29.3	29.0	37.2	28.8	29.3	33.9
Average	34.1	33.9	30.9	34.0	33.2	34.0	33.6	30.7	35.9	31.4	32.3	35.3
RANGE	6.09	6.16	6.33	8.85	2.65	2.82	6.78	4.67	5.27	6.73	8.37	2.59
SUM	170.7	169.7	154.5	170.0	165.8	169.8	168.0	153.3	179.3	157.2	161.6	176.4
NO. INSP.	5	5	5	5	5	5	5	5	5	5	5	5
ALARM	- -	- -	- -	- -	- -	- -	- -	- -	H -	- -	- -	H -
NOTES	-N-	-N-	-N-	-N-	-N-	-N-	-N-	-N-	-N-	-N-	-N-	-N-

When you combine the table with the chart, the number of data points displayed should be limited so that the table columns can line up with the data points in the chart underneath.

Table Setup Items

The TableSetup property is under the SPCChart property. TableSetup contains all of the values needed to customize the table display and features.

```
SPCChart
   TableSetup
      HeaderStringsLevel: SPC String constant: "HEADER_STRINGS_LEVEL2"
      EnableInputStringsDisplay: boolean: true
      EnableCategoryValues: boolean: true
      EnableSampleValues: boolean: true
      EnableCalculatedValues: boolean: true
      EnableProcessCapabilityValues: boolean: true
      EnableTotalSamplesValues: boolean: true
      EnableNotes: boolean: true
      EnableTimeValues: boolean: true
      EnableDataToolTip: boolean: true
      EnableNotesToolTip: boolean: true
      EnableTableRowToggles: boolean: false
      NotesToolTip
         ButtonMask: SPC String constant: "BUTTON1_MASK"
         ToolTipMode: SPC String constant: "MOUSETOGGLE_TOOLTIP"
         NotesReadOnly: boolean: false
      TableBackgroundMode:SPC String Constant: "TABLE_SINGLE_COLOR_BACKGROUND"
      TableAlarmEmphasisMode: SPC String Constant: "ALARM_HIGHLIGHT_NONE"
      ChartAlarmEmphasisMode: SPC String Constant: "ALARM_HIGHLIGHT_SYMBOL"
      BackgroundColor1: Color String constant: "WHITE"
```

HeaderStringsLevel

Example:

```
   "HeaderStringsLevel": "HEADER_STRINGS_LEVEL3",
```

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, Gauge, 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, Gauge, UnitOfMeasure, ZeroEquals and DateString

EnableInputStringsDisplay

Example:

```
      "EnableInputStringsDisplay": true
```

This turns on/off the header part of the table. If false, same as "HeaderStringsLevel": "HEADER_STRINGS_LEVEL0".

EnableCategoryValues (EnableSampleValues also works)

Example:

```
     "EnableCategoryValues": true
```

This turns on/off the display of the sample value data for each sample subinterval.

EnableCalculatedValues

Example:

```
     "EnableCalculatedValues": true
```

This turns on/off the display of the calculated values for each sample subinterval.

EnableProcessCapabilityValues

Example:

```
     "EnableProcessCapabilityValues": true
```

This turns on/off the display of the process capability values for each sample subinterval.

EnableTotalSamplesValues

Example:

```
     "EnableTotalSamplesValues": true
```

This turns on/off the display of the total number of samples for each sample subinterval.

EnableNotes

Example:

```
    "EnableNotes": true
```

This turns on/off the display of the Notes line of the table.

EnableTimeValues

Example:

```
    "EnableTimeValues": true
```

This turns on/off the display of the Time line of the table

EnableDataToolTip

Example:

```
    "EnableDataToolTip": true
```

This turns on/off the of the data tooltip for the charts.

EnableNotesToolTip

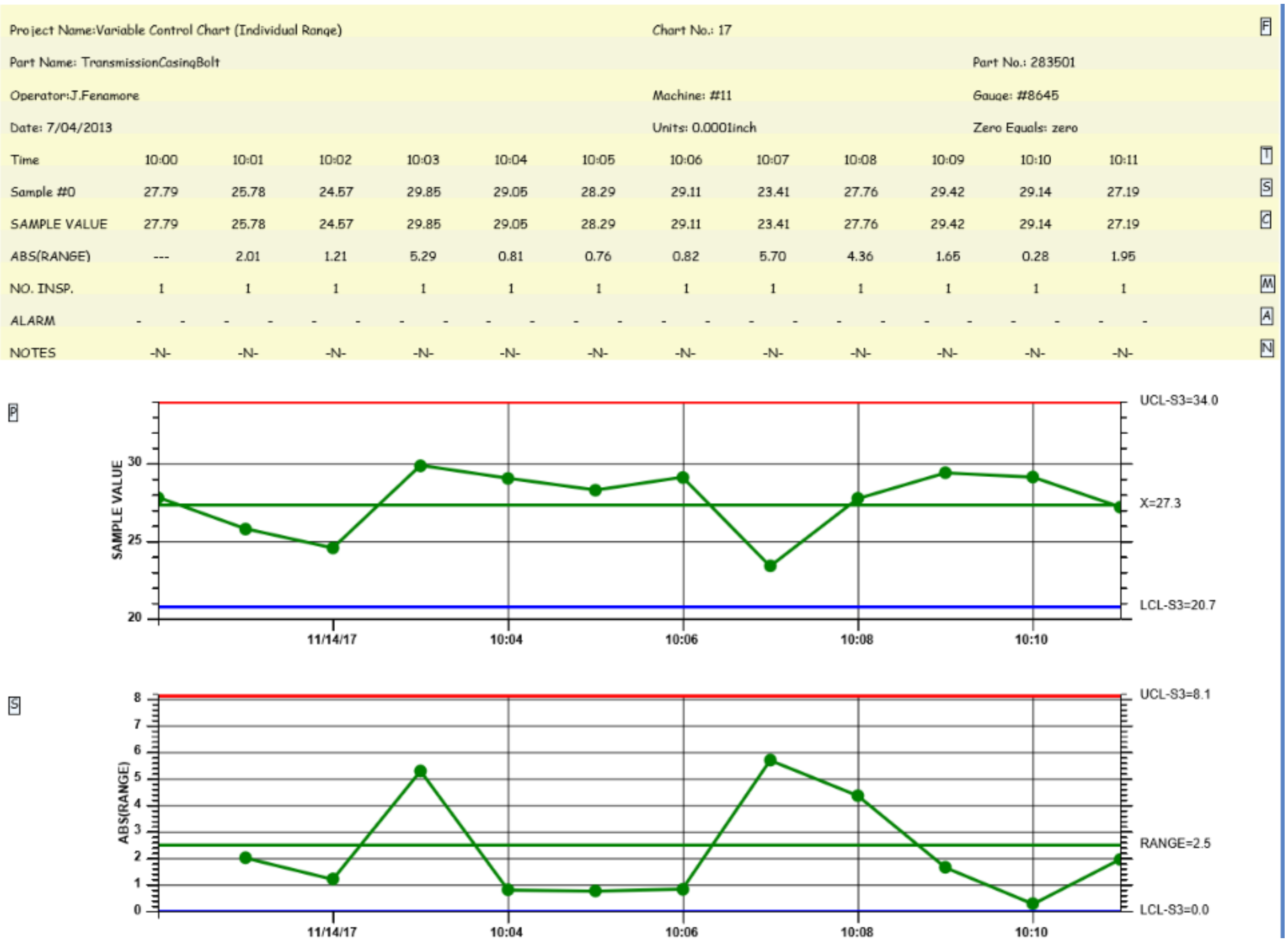
Example:

```
    "EnableNotesToolTip": true
```

This turns on/off the of the Notes tooltip for the charts.

EnableTableRowToggles

Like the zoom and chart display options described elsewhere, there are also UI buttons which can toggle on and off rows in the table Like the Zoom button, these UI buttons can be hidden from the end user if you don't want them to show. On the top and right of the table buttons will selectively collapse/restore rows of the table, freeing up more space for charts.



Project Name:Variable Control Chart (Individual Range) Chart No.: 17

Part Name: TransmissionCasingBolt Part No.: 283501

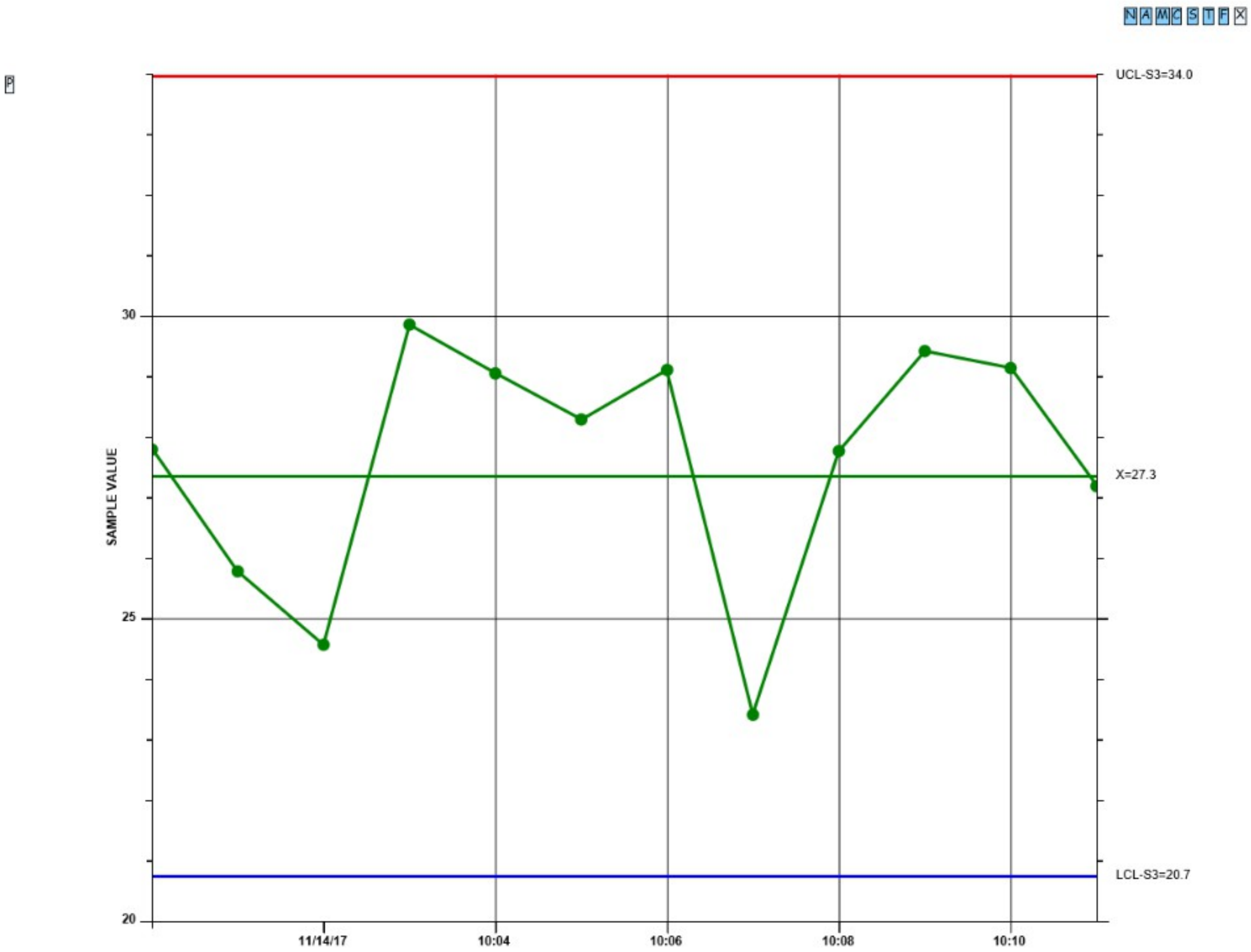
Operator:J.Fenamore Machine: #11 Gauge: #8645

Date: 7/04/2013 Units: 0.0001inch Zero Equals: zero

Time	10:00	10:01	10:02	10:03	10:04	10:05	10:06	10:07	10:08	10:09	10:10	10:11
Sample #0	27.79	25.78	24.57	29.85	29.05	28.29	29.11	23.41	27.76	29.42	29.14	27.19
SAMPLE VALUE	27.79	25.78	24.57	29.85	29.05	28.29	29.11	23.41	27.76	29.42	29.14	27.19
ABS(RANGE)	---	2.01	1.21	5.29	0.81	0.76	0.82	5.70	4.36	1.65	0.28	1.95
NO. INSP.	1	1	1	1	1	1	1	1	1	1	1	1
ALARM	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -
NOTES	-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

Hide the chart buttons on the left.

```
“PrimaryChartSetup”: {
   “EnableChartDisplayToggles”: false
}
```

Hide the table buttons on the right.

```
"TableSetup": {
          “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.

```
“MiscChartDataProperties”: {
         “EnableDisplayOptionToggles”: false
}
```

TableBackgroundMode

Example:

```
     "TableBackgroundMode": "TABLE_NO_COLOR_BACKGROUND"
```

The default table background uses the accounting style green-bar striped background. You can change this using the **TableBackgroundMode** property**.** Set the value to one of the TableBackgroundMode constants**.**

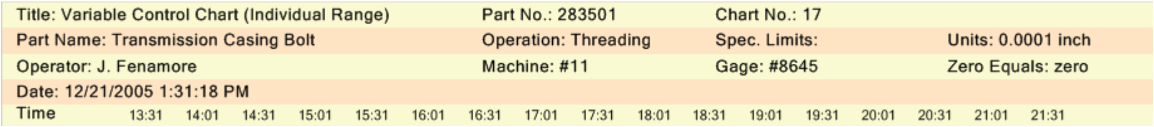
TABLE_NO_COLOR_BACKGROUND
Constant specifies that the table does not use a background color.

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 chartDefExampleScripts.js TimeIR example JSON script.

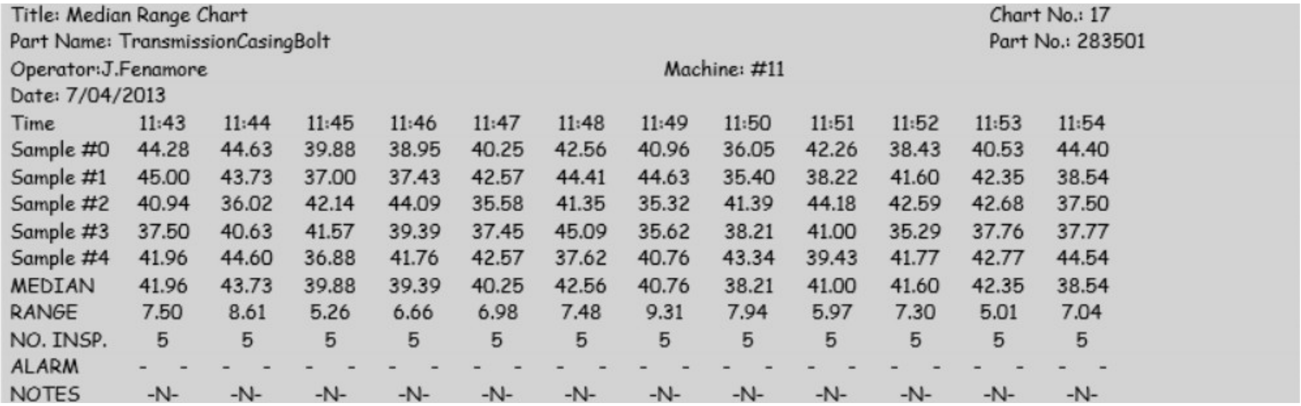


Title: Variable Control Chart (Individual Range)	Part No.: 283501	Chart No.: 17	
Part Name: Transmission Casing Bolt	Operation: Threading	Spec. Limits:	Units: 0.0001 inch
Operator: J. Fenamore	Machine: #11	Gage: #8645	Zero Equals: zero
Date: 12/21/2005 1:31:18 PM			

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

```
"TableBackgroundMode": "TABLE_STRIPED_COLOR_BACKGROUND",
"BackgroundColor1": "BEIGE",
"BackgroundColor2": "LIGHTGOLDENRODYELLOW",
```

Extracted from the chartDefExampleScripts.js BatchMedianRangeChart example JSON script

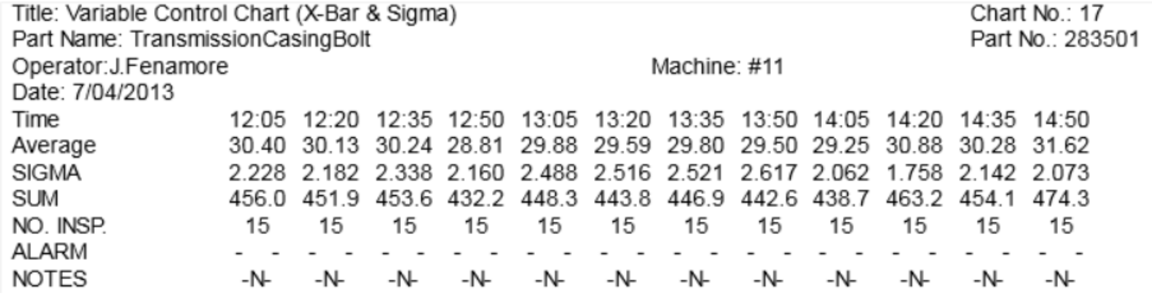


Title: Median Range Chart		Chart No.: 17
Part Name: TransmissionCasingBolt		Part No.: 283501
Operator:J.Fenamore	Machine: #11	
Date: 7/04/2013		

Time	11:43	11:44	11:45	11:46	11:47	11:48	11:49	11:50	11:51	11:52	11:53	11:54
Sample #0	44.28	44.63	39.88	38.95	40.25	42.56	40.96	36.05	42.26	38.43	40.53	44.40
Sample #1	45.00	43.73	37.00	37.43	42.57	44.41	44.63	35.40	38.22	41.60	42.35	38.54
Sample #2	40.94	36.02	42.14	44.09	35.58	41.35	35.32	41.39	44.18	42.59	42.68	37.50
Sample #3	37.50	40.63	41.57	39.39	37.45	45.09	35.62	38.21	41.00	35.29	37.76	37.77
Sample #4	41.96	44.60	36.88	41.76	42.57	37.62	40.76	43.34	39.43	41.77	42.77	44.54
MEDIAN	41.96	43.73	39.88	39.39	40.25	42.56	40.76	38.21	41.00	41.60	42.35	38.54
RANGE	7.50	8.61	5.26	6.66	6.98	7.48	9.31	7.94	5.97	7.30	5.01	7.04
NO. INSP.	5	5	5	5	5	5	5	5	5	5	5	5
ALARM	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -
NOTES	-N-	-N-	-N-	-N-	-N-	-N-	-N-	-N-	-N-	-N-	-N-	-N-

```
"TableBackgroundMode": "TABLE_SINGLE_COLOR_BACKGROUND",
"BackgroundColor1": "LIGHTGREY",
```

Extracted from the chartDefExampleScripts.js TimeMeanSigma example JSON script

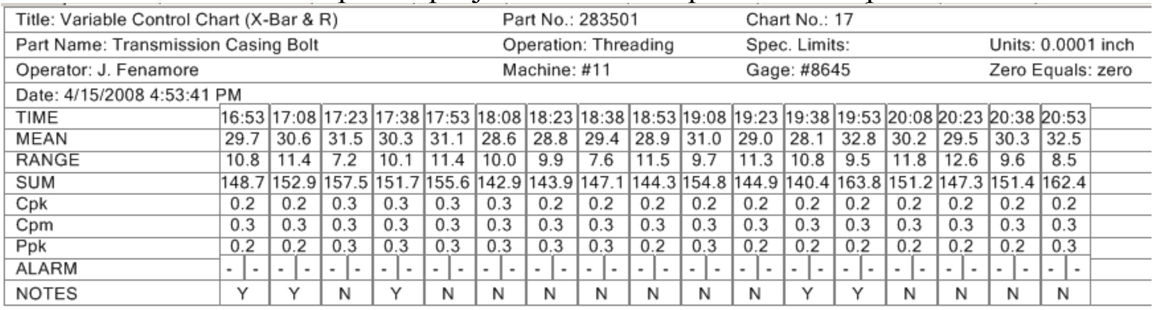


Title: Variable Control Chart (X-Bar & Sigma) Chart No.: 17
Part Name: TransmissionCasingBolt Part No.: 283501
Operator:J.Fenamore Machine: #11
Date: 7/04/2013

Time	12:05	12:20	12:35	12:50	13:05	13:20	13:35	13:50	14:05	14:20	14:35	14:50
Average	30.40	30.13	30.24	28.81	29.88	29.59	29.80	29.50	29.25	30.88	30.28	31.62
SIGMA	2.228	2.182	2.338	2.160	2.488	2.516	2.521	2.617	2.062	1.758	2.142	2.073
SUM	456.0	451.9	453.6	432.2	448.3	443.8	446.9	442.6	438.7	463.2	454.1	474.3
NO. INSP.	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-

```
"TableBackgroundMode": "TABLE_NO_COLOR_BACKGROUND",
```

Extracted from the chartDefExampleScripts.js BatchIR example JSON script

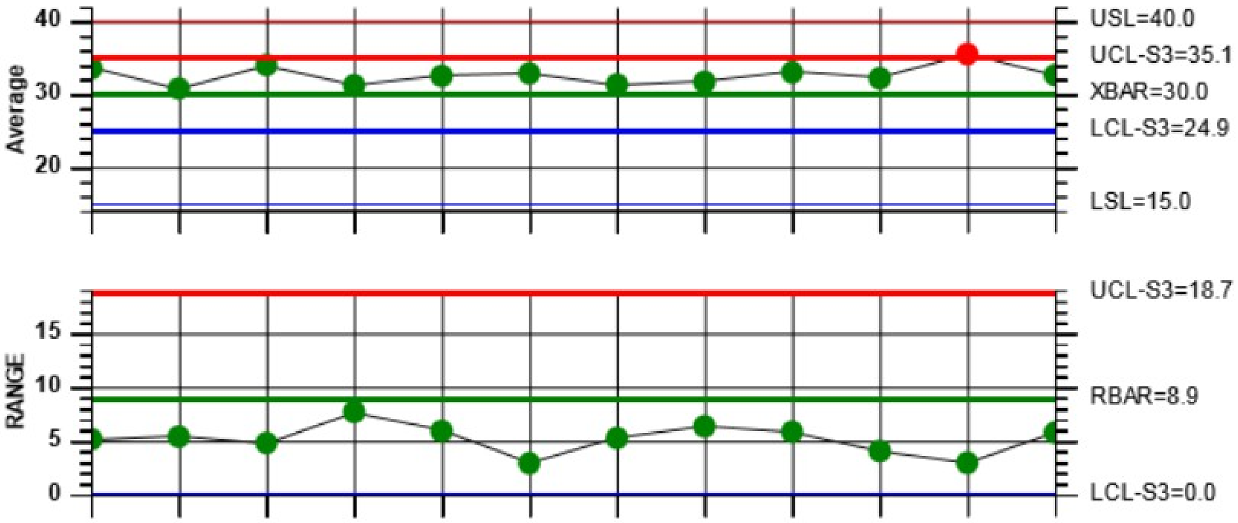


Title: Variable Control Chart (X-Bar & R) Part No.: 283501 Chart No.: 17
Part Name: Transmission Casing Bolt Operation: Threading Spec. Limits: Units: 0.0001 inch
Operator: J. Fenamore Machine: #11 Gage: #8645 Zero Equals: zero
Date: 4/15/2008 4:53:41 PM

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

```
"TableBackgroundMode":"TABLE_SINGLE_COLOR_BACKGROUND_GRIDCELL",
"BackgroundColor1": "WHITE",
 "BackgroundColor2": "GRAY",
```

ChartAlarmEmphasisMode



Example

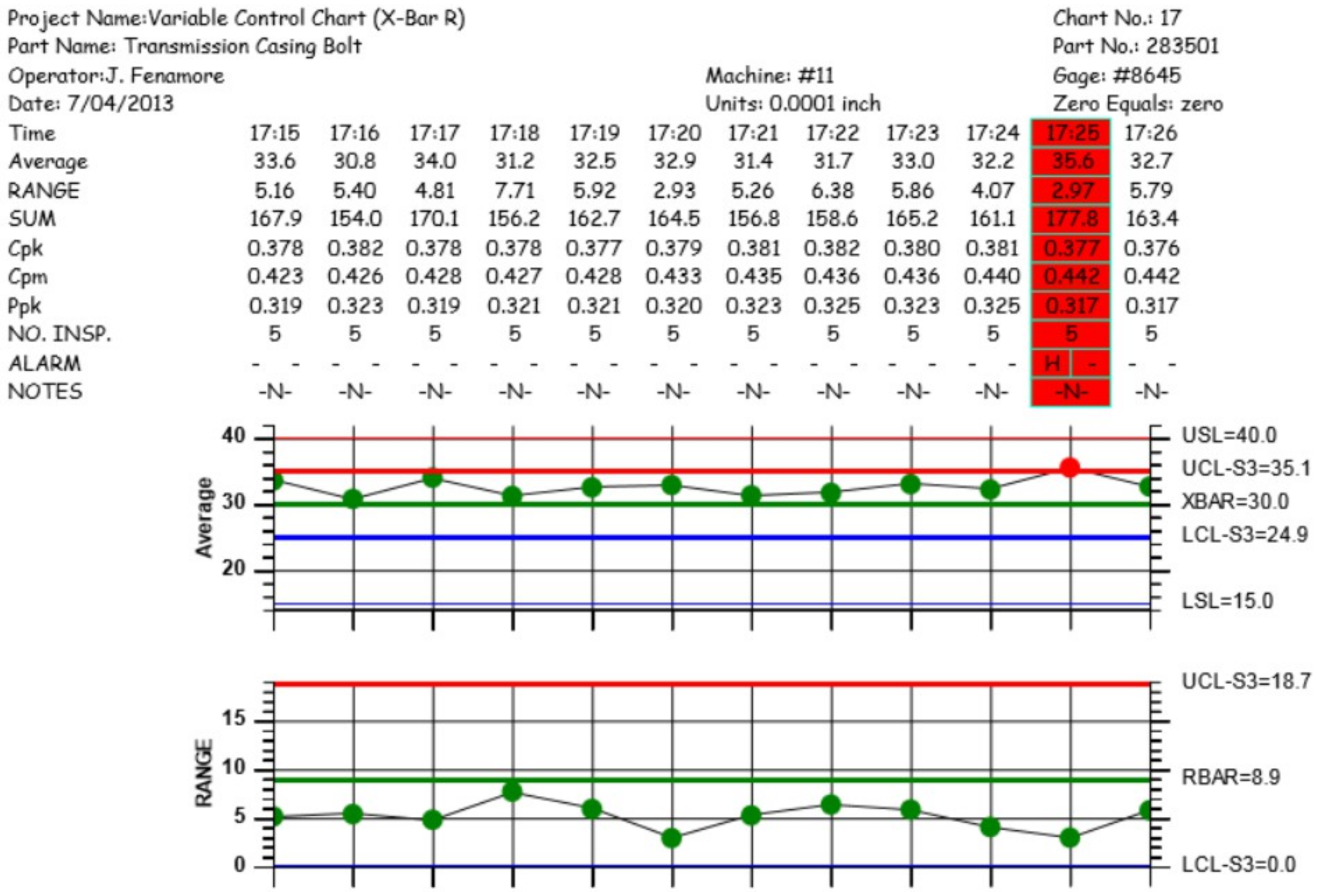
```
"ChartAlarmEmphasisMode":"ALARM_HIGHLIGHT_SYMBOL",
```

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 ALARM_NO_HIGHLIGHT_SYMBOL constants.

Custom Alarm Symbol Shapes and Size

The QCSPCChart+ version permits you to specify a custom symbol shape, and symbol size, for the alarm high symbol. You do this by setting the *OutOfLimitSymbolSize* and *OutOfLimitSymbolType* properties under the *PrimaryChartSetup*, *SecondaryChartSetup* or *ThirdChartSetup* blocks.

TableAlarmEmphasisMode



Project Name:Variable Control Chart (X-Bar R)
Part Name: Transmission Casing Bolt
Operator:J. Fenamore
Date: 7/04/2013

Machine: #11
Units: 0.0001 inch

Chart No.: 17
Part No.: 283501
Gage: #8645
Zero Equals: zero

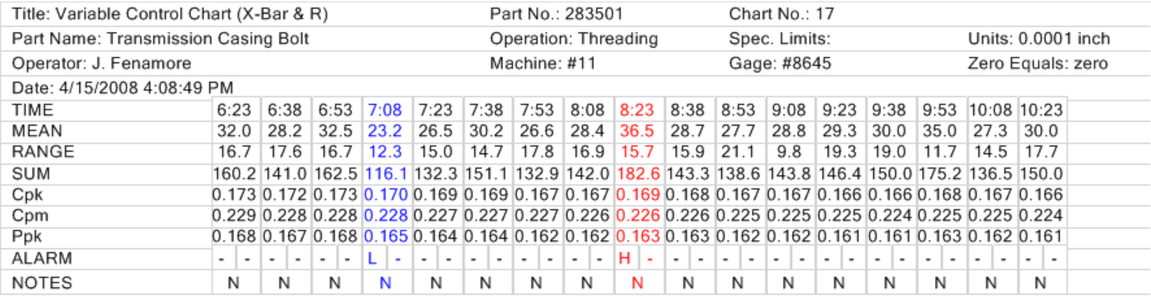
Time	17:15	17:16	17:17	17:18	17:19	17:20	17:21	17:22	17:23	17:24	17:25	17:26
Average	33.6	30.8	34.0	31.2	32.5	32.9	31.4	31.7	33.0	32.2	35.6	32.7
RANGE	5.16	5.40	4.81	7.71	5.92	2.93	5.26	6.38	5.86	4.07	2.97	5.79
SUM	167.9	154.0	170.1	156.2	162.7	164.5	156.8	158.6	165.2	161.1	177.8	163.4
Cpk	0.378	0.382	0.378	0.378	0.377	0.379	0.381	0.382	0.380	0.381	0.377	0.376
Cpm	0.423	0.426	0.428	0.427	0.428	0.433	0.435	0.436	0.436	0.440	0.442	0.442
Ppk	0.319	0.323	0.319	0.321	0.321	0.320	0.323	0.325	0.323	0.325	0.317	0.317
NO. INSP.	5	5	5	5	5	5	5	5	5	5	5	5
ALARM	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	H -	- -
NOTES	-N-	-N-	-N-	-N-	-N-	-N-	-N-	-N-	-N-	-N-	-N-	-N-

Example
`"TableAlarmEmphasisMode":"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.

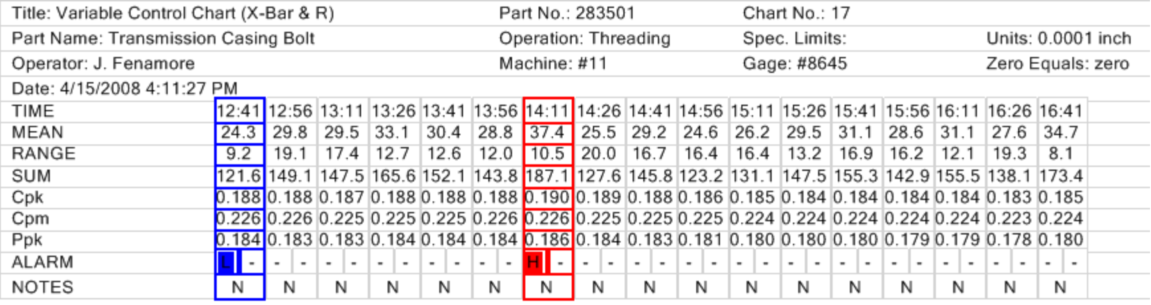


Title: Variable Control Chart (X-Bar & R)	Part No.: 283501	Chart No.: 17	
Part Name: Transmission Casing Bolt	Operation: Threading	Spec. Limits:	Units: 0.0001 inch
Operator: J. Fenamore	Machine: #11	Gage: #8645	Zero Equals: zero

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.

ChartData

```
ChartData
   Title: String: ""
   PartNumber: String: ""
   ChartNumber: String: ""
   PartName: String: ""
   Operation: String: ""
   SpecificationLimits: String: ""
   Operator: String: ""
   Machine: String: ""
   Gauge: String: ""
   UnitOfMeasure: String: ""
   ZeroEquals: String: ""
   DateString: String: ""
   NotesMessage: String: ""
   ProcessCapabilitySetup
      LSLValue: double: 0
      USLValue: double: 1
      EnableCPK: boolean: false
      EnableCPM: boolean: false
      EnablePPK: boolean: false
      EnableCPL: boolean: false
      EnableCPU: boolean: false
      EnablePPL: boolean: false
      EnablePPU: boolean: false
   SampleItemDecimals: integer: 2
   CalculatedItemDecimals: integer: 2
   ProcessCapabilityDecimals: integer: 2
   CustomTimeFormatString: String: ""
```

```
   TimeFormat: SPC String constant: "TIMEDATEFORMAT_24HM"
```

ChartData is underneath the TableSetup object. It has a list of properties which control the strings displayed in the top (header) section of the table.

DateHeader	Set the header for the dateString field.
DateString	Set data table date string.
Gauge	Set data table Gauge string.
Machine	Set data table machine string.
NotesMessage	Set data table notes message string.
Operation	Set data table operation string.
PartName	Set data table part name string.
PartNumber	Set data table part number string.
SpecificationLimits	Set data table specification limits string.
Operator	Set data table operator string.
Title	Set data table title string.
UnitOfMeasure	Set data table unit of measure string.
ZeroEquals	Set data table zero equals string.

Example

```
"ChartData": {
   "Title": "Variable Control Chart (X-Bar R)",
   "PartNumber": "283501",
   "ChartNumber": "17",
   "PartName": "Transmission Casing Bolt",
   "Operation": "Threading",
   "SpecificationLimits": "27.0 to 35.0",
   "Operator": "J. Fenamore",
   "Machine": "#11",
   "Gauge": "#8645",
   "UnitOfMeasure": "0.0001 inch",
   "ZeroEquals": "zero",
   "DateString": "7/04/2013",
   "NotesMessage": "Control limits prepared May 10",
   "NotesHeader": "NOTES"
 }
```

The above properties represent the values of the fields displayed on the screen. There are also static properties and values which control the label displayed in front of the ChartData field values. If you want to customize these values, see the chapter on static initializations.

7. SPC Chart Setup

```
PrimaryChartSetup
      EnableChart: boolean: true
      EnableChartDisplayToggles: boolean: true
      XAxis
      XAxisLabels
      YAxisLeft
      YAxisLeftLabels
      FrequencyHistogram
      PlotMeasurementValues: boolean: false
      LineMarkerPlot
      GraphBackground
      PlotBackground
      OutOfLimitSymbolSize
      OutOfLimitSymbolType
      ControlLimits
         Target
         LCL3
         UCL3
         LCL2
         UCL2
         LCL1
         UCL1
         123SigmaControlLimits
         AddControlRules
         SpecifyControlLimitsUsingMeanAndSigma

   SpecificationLimits
      LowSpecificationLimit
      HighSpecificationLimit

SecondaryChartSetup
```

SPC charts are the one or two charts which appear under the SPC table. They represent the graphical interpretation of the SPC data. Most of the variable control charts use two charts. For example, the X-Bar R chart (also called a Mean Range chart, *`MEAN_RANGE_CHART`*) includes a primary chart which plots the sample interval mean values and a secondary chart which plots the sample interval range values. In the primary chart, sometimes the median stands in for the range (*`MEDIAN_RANGE_CHART)`*, because before everyone stated using computers for SPC, the median was easier to calculate by hand than the mean. Also, in some primary charts (EWMA, MA, MAMS, MAMR), a moving average across sample intervals is used, to reduce spurious out of control signals due to noise. In the secondary chart, sometimes the sample standard deviation, or variance, is used instead of the range. There are variable control charts which also use a single chart, rather than a synchronized pair of charts. These include the some of the moving average charts (EWMA, MA) and the tabular CuSum chart (TABSUB_CHART). All of the attribute control charts use a single chart.

There are are also a couple of auxiliary charts, while used extensively in SPC, are not classified as SPC charts for the purposes of this chapter. They do not share a the common structure as the other SPC charts and cannot be programmed under the SPCChart object of the defining JSON script. These two charts have their own defining structure, and are programmed under their own JSON blocks (FrequencyHistogram and ParetoChart), defined in Chapter 17 and 18.

Once the initial chart type is defined, a long list of default properties, representing common usage for the selected chart type, are set automatically. You can modify the default setup using JSON. Note, you only need to specify a JSON property:value pair if you want to change a property from it's default.

Enable Chart

```
PrimaryChartSetup
      EnableChart: boolean: true
```

Set to true by default, set the EnableChart property to false and the associated chart (Primary or Secondary) will not display. This is used almost 100% of the time to turn off the secondary chart (a range or sigma chart in the case of Variable control charts), leaving just the primary chart occupying the entire chart area.

```
"SecondaryChartSetup": {
            "EnableChart":  false
            }
```

Enable the Chart Display Toggles

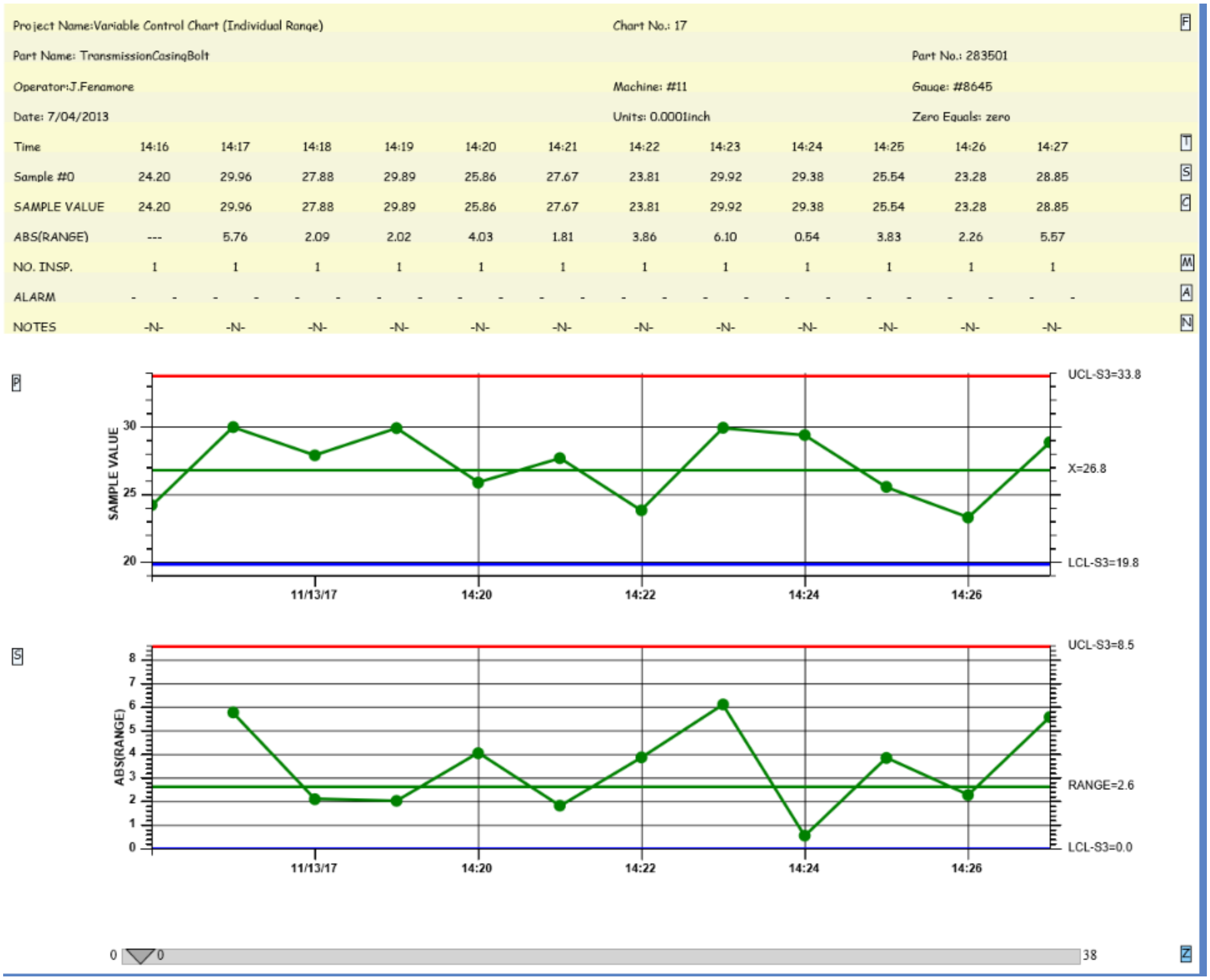
```
PrimaryChartSetup
      EnableChartDisplayToggles: boolean: true
```

Set to true by default, set the EnableChartDisplayToggles property to false and the chart display button toggles to the left of the chart will not display. Setting the property false will turn off the toggles for all charts (Primary, Secondary and Third).

```
"PrimaryChartSetup": {
            "EnableChartDisplayToggles":  false
           }
```

Collapsible Items

Like the Zoom option 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. 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.

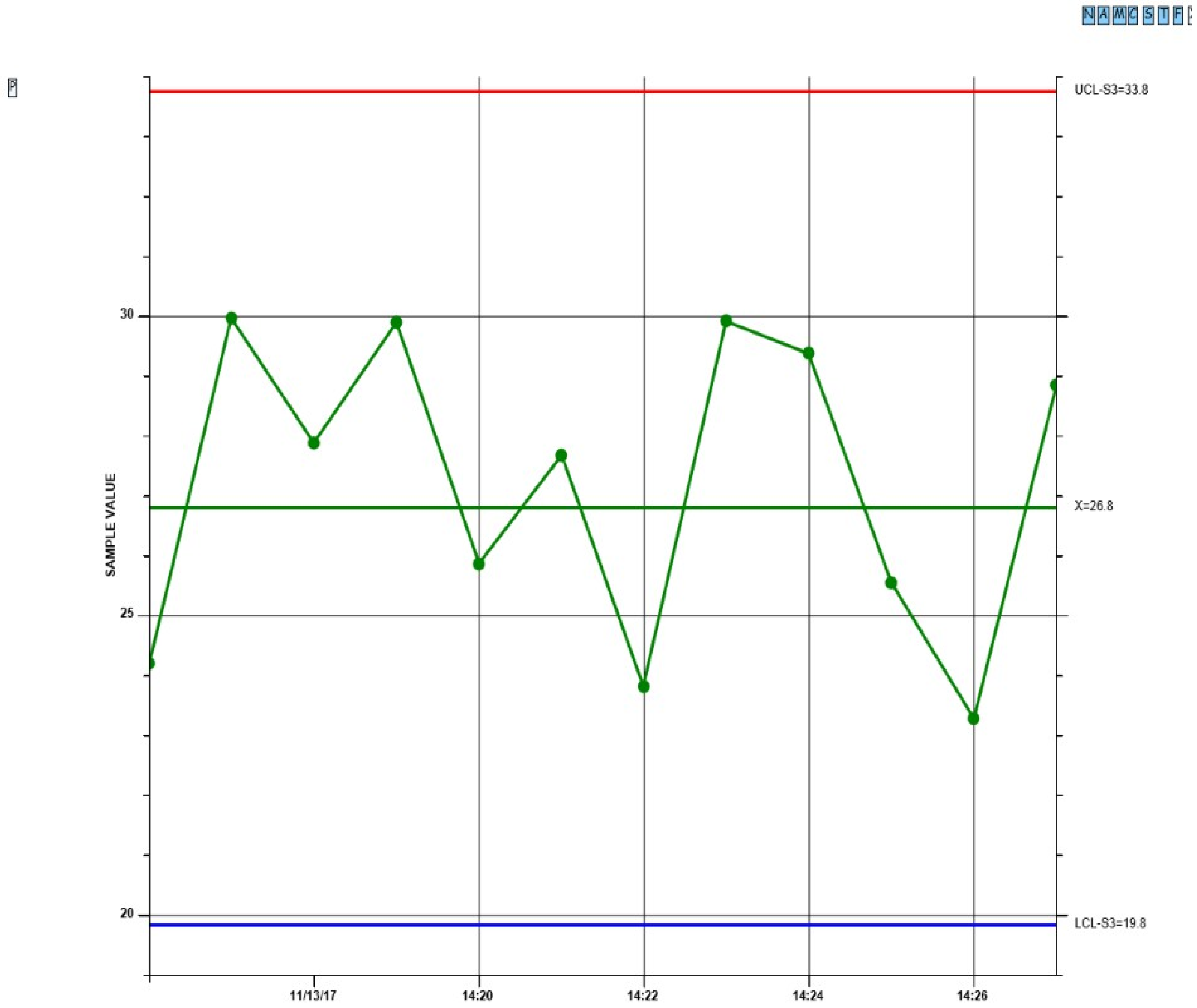


Project Name:Variable Control Chart (Individual Range)   Chart No.: 17   F
Part Name: TransmissionCasingBolt   Part No.: 283501
Operator:J.Fenamore   Machine: #11   Gauge: #8645
Date: 7/04/2013   Units: 0.0001inch   Zero Equals: zero

Time	14:16	14:17	14:18	14:19	14:20	14:21	14:22	14:23	14:24	14:25	14:26	14:27	T
Sample #0	24.20	29.96	27.88	29.89	25.86	27.67	23.81	29.92	29.38	25.54	23.28	28.85	S
SAMPLE VALUE	24.20	29.96	27.88	29.89	25.86	27.67	23.81	29.92	29.38	25.54	23.28	28.85	C
ABS(RANGE)	---	5.76	2.09	2.02	4.03	1.81	3.86	6.10	0.54	3.83	2.26	5.57	
NO. INSP.	1	1	1	1	1	1	1	1	1	1	1	1	M
ALARM	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	A
NOTES	-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, F, T, C and S buttons and shrink the table to following size.



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.

```
“PrimaryChartSetup”: {
   “EnableChartDisplayToggles”: false
}
```

Hide the table buttons on the right.

```
"TableSetup": {
           “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.

```
“MiscChartDataProperties”: {
          “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. Note that each toggle is in a different block.

```
“MiscChartDataProperties”: {
          “EnableDisplayOptionToggles”: false
}

"TableSetup": {
           “EnableTableRowToggles”: false
}

“PrimaryChartSetup”: {
   “EnableChartDisplayToggles”: false
}

“SPCChart”: {
   “Scrollbar”: {
          “EnableZoomToggle”: true
   }
}
```

Axes

Most of the options associated with the x- and y-axes, and axes labels, are set automatically, based on the range and magnitude of the data values being plotted, and the position of the scroll bar. What you are left with are basic attribute settings (color and line width), and some format settings in the case of a time-based x-axis.

XAxis

```
XAxis
```

```
   LineColor: Color String constant: "BLACK"
   LineWidth: double: 1
```

LineColor

Set the line color using one of the color constant strings

LineWidth

Set the line width to something other than the default value of 1

Example

```
"XAxis": {
           "LineColor": "BLUE",
           "LineWidth": 3
         },
```

XAxisLabels

```
XAxisLabels
   Font
      Name: String: "sans-serif"
      Size: double: 12
      Style: SPC String Constant: "PLAIN"
   TextColor: Color String constant: "BLACK"
   Rotation: double: 0
   Format: SPC String constant: "TIMEDATEFORMAT_24HM"
   CustomFormatString: String: ""
   OverlapLabelMode: SPC String constant: "OVERLAP_LABEL_STAGGER"
   AxisLabelMode: SPC String constant: "AXIS_LABEL_MODE_DEFAULT"
```

Font

Name

Set the axis labels font family to something other than the default "sans-serif". Follow the font naming guidelines detailed in Static Properties chapter, under DefaultTableFont

Size

Font size in points.

Style

Use of the style string constants: "Plain", "Normal", "Bold","Italic","Bold Italic".

TextColor

Color of the axis labels. Use one of the string color constants.

Rotation

Rotate the labels about their horizontal postiion. Specify using an integer value of degrees

Format

Select one of our predefined time/date formats. Use one of the SPC string constants.

CustomFormatString

Specify your own time/date string, following

OverlapLabelMode

Specifies a repositioning strategy to use if the axis labels are so close together they start to overlap. Use one of the overlap label constants: OVERLAP_LABEL_STAGGER, OVERLAP_LABEL_DELETE, or OVERLAP_LABEL_DRAW.

AxisLabelMode

Set labeling mode of the x-axis. Use AXIS_LABEL_MODE_DEFAULT for the default time labeling for Time-based controls charts and numeric for Batch-based controls charts. Use AXIS_LABEL_MODE_STRING to add user-defined labels specified using the BatchIDString of the SampleData block. Use AXIS_LABEL_MODE_TIME to table the axis with the time stamp of the associated record.

Example

```
"XAxisLabels": {
               "AxisLabelMode": "AXIS_LABEL_MODE_STRING"
                },
```

YAxisLeft

```
YAxisLeft
   LineColor: Color String constant: "BLACK"
   LineWidth: double: 1
   MinValue: double: 0
   MaxValue: double: 1
```

LineColor

Set the line color using one of the color constant strings

LineWidth

Set the line width to something other than the default value of 1

MinValue

The minimum value of the y-axis

MaxValue

The maximum value of the y-axis

It you set the MinValue or MaxValues for the y-axis, those values will only stick if you do not call the AutoScaleYAxes method. Otherwise, the values for the axis minimum and maximum will be calculated to be inclusive of of the data values and control limits in the graph.

Example

```
"YAxisLeft": {
               "LineColor": "GREEN",
               "LineWidth": 3
             },
```

YAxisLeftLabels

```
YAxisLeftLabels
   Font
      Name: String: "sans-serif"
      Size: double: 12
      Style: SPC String Constant: "PLAIN"
   TextColor: Color String constant: "BLACK"
   Rotation: double: 0
   Format:  constant(String)
   OverlapLabelMode:  SPC String constant: "OVERLAP_LABEL_STAGGER"
   Decimal: integer: 1
```

Font

Name

Set the axis labels font family to something other than the default "sans-serif". Follow the font naming guidelines detailed in Static Properties chapter, under DefaultTableFont

Size

Font size in points.

Style

Use of the style string constants: "Plain", "Normal", "Bold","Italic","Bold Italic".

TextColor

Color of the axis labels. Use one of the string color constants.

Rotation

Rotate the labels about their horizontal postiion. Specify using an integer value of degrees

Format

Select one of our predefined time/date formats. Use one of the SPC string constants.

CustomFormatString

Specify your own time/date string, following

OverlapLabelMode

Specifies a repositioning strategy to use if the axis labels are so close together they start to overlap. Use one of the overlap label constants: OVERLAP_LABEL_STAGGER, OVERLAP_LABEL_DELETE, or OVERLAP_LABEL_DRAW.

Decimal

Specify the decimal precision for a numeric axis

Example

```
"YAxisLeftLabels": {
                      "TextColor": "RED",
                      "Font": {
                         "Size": 14,
                         "Style":  "BOLD"
                      }
                   },
```

YAxisRight

```
YAxisRight
   LineColor: Color String constant: "BLACK"
   LineWidth: double: 1
   MinValue: double: 0
    MaxValue: double: 1
```

LineColor

Set the line color using one of the color constant strings

LineWidth

Set the line width to something other than the default value of 1.

MinValue

The minimum value of the y-axis

MaxValue

The maximum value of the y-axis

In the single variable chart scenario, the minimum and maximum value of the right y-axis tracks the left y-axis. Do not try and set independent values in the case. In the multi-variable scenario, you can set an set of independent minimum and maximum values for the right y-axis.

Example

Single Variable and Multi-Variable which uses only a single y-axis scale

```
"YAxisRight": {
               "LineColor": "GREEN",
               "LineWidth": 3
             },
```

Multi-Variable chart with an independent right y-axis

```
"YAxisLeft": {
              "LineColor": "GREEN",
              "MinValue":10,
              "MaxValue":30
             },
"YAxisRight": {
                "LineColor": "RED",
                "MinValue":20,
                "MaxValue":50
            },
```

FrequencyHistogram

```
FrequencyHistogram
   EnableDisplayFrequencyHistogram: boolean: true
   PlotBackgroundColor : Color String constant: "WHITE"
   BarColor: Color String constant: "LIGHTBLUE"
```

EnableDisplayFrequencyHistogram

Set to false to disable the frequency displayed to the left of the chart area.

PlotBackgroundColor

Specify the color of the frequency histogram plot area. Use on of the string color constants.

BarColor

Specify the color of the frequency histogram bars. Use on of the string color constants.

PlotMeasurementValues

```
PlotMeasurementValues: boolean: false
```

PlotMeasurementValues

Set to true to plot each sample of a sample interval as a scatter plot symbol.

Example

```
"PlotMeasurmentValues": true,
```

LineMarkerPlot

```
LineMarkerPlot
   LineColor: Color String constant: "BLUE"
   LineWidth: double: 1
   SymbolColor: Color String constant: "BLUE"
   SymbolFillColor: Color String constant: "BLUE"
   SymbolType: SPC String constant: "CIRCLE"
```

LineColor

Specify the color of the connecting the symbols of the charts line marker plot. Use one of the string color constants.

LineWidth

Specify the line width of the connecting the symbols of the charts line marker plot.

SymbolColor

Specify the outline color of the line marker plot symbol.

SymbolFillColor

Specify the fill color of the line marker plot symbol.

SymbolType

Specify the symbol type of the line marker plot. Use one of the SPC Chart string constants: NOSYMBOL, SQUARE, UPTRIANGLE, DOWNTRIANGLE ,DIAMOND, CROSS, PLUS, STAR, LINE, HBAR, VBAR, BAR3D, CIRCLE.

Example

```
"LineMarkerPlot": {
                   "LineColor": "GREEN",
                   "LineWidth": 2,
                   "SymbolColor": "SPRINGGREEN",
                   "SymbolFillColor": "SPRINGGREEN",
                   "SymbolType": "CIRCLE"
               },
```

GraphBackground

```
GraphBackground
   FillColor: Color String constant: "WHITE"
   BackgroundMode: SPC String constant: "SIMPLECOLORMODE"
   GradientStartColor: Color String constant: "WHITE"
   GradientStopColor: Color String constant: "LIGHTGRAY"
```

This property defines background properties of the graph background.

FillColor

Specify the background color for solid fills (BackgroundMode = SIMPLECOLORMODE)

BackgroundMode

Specify the background mode of the background. Use one of the SPC chart string constants: SIMPLEGRADIENTMODE or SIMPLECOLORMODE.

GradientStartColor

If SIMPLEGRADIENTMODE is specified as the BackgroundMode, specify the starting gradient color.

GradientStopColor

If SIMPLEGRADIENTMODE is specified as the BackgroundMode, specify the ending gradient color.

```
"GraphBackground": {
                   "FillColor": "BROWN",
                   "BackgroundMode": "SIMPLECOLORMODE"
               },
```

PlotBackground

```
PlotBackground
   FillColor: Color String constant: "WHITE"
   BackgroundMode: SPC String constant: "SIMPLECOLORMODE"
   GradientStartColor: Color String constant: "WHITE"
   GradientStopColor: Color String constant: "LIGHTGRAY"
```

This property defines background properties of the plot area background.

FillColor

Specify the background color for solid fills (BackgroundMode = SIMPLECOLORMODE)

BackgroundMode

Specify the background mode of the background. Use one of the SPC chart string constants: SIMPLEGRADIENTMODE or SIMPLECOLORMODE.

GradientStartColor

If SIMPLEGRADIENTMODE is specified as the BackgroundMode, specify the starting gradient color.

GradientStopColor

If SIMPLEGRADIENTMODE is specified as the BackgroundMode, specify the ending gradient color.

Example

```
"PlotBackground": {
```

```
                "FillColor": "BROWN",
                "BackgroundMode": "SIMPLECOLORMODE"
            },
```

Control Limits

```
ControlLimits
             Font
                Name: String: "sans-serif"
                Size: double: 12
                Style: SPC String Constant: "PLAIN"
             DefaultLimits [ boolean: true, boolean: true]
             SetLimits: [double: 0, double: 0, double 0]
             Decimal: integer: 1
             OutOfLimitSymbolSize: integer: 14
             OutOfLimitSymbolType: SPC String Contants: "CIRCLE"
             ZoneFill: boolean: false
             ZoneColors:  [
                             Color String constant: "ORANGERED",
                             Color String constant: "LIGHTGOLDENRODYELLOW",
                             Color String constant: "LIGHTGREEN"
                          ]
             Target
                LineColor: Color String constant: "GREEN"
                TextColor: Color String constant: "BLACK"
                LineWidth: double: 1
                LimitValue: double: 0
                DisplayString: String: "XBAR"
                EnableAlarmLine: boolean: true
                EnableAlarmChecking : boolean: true
             LCL3
                LineColor: Color String constant
                TextColor: Color String constant
                LineWidth: double: 1
                LimitValue: double: 0
                DisplayString: String: "LCL"
                EnableAlarmLine: boolean: true
                EnableAlarmChecking : boolean: true
                EnableAlarmLineText: String: true
             UCL3
                LineColor: Color String constant
                TextColor: Color String constant
                LineWidth: double: 1
                LimitValue: double: 0
                DisplayString: String: UCL
                EnableAlarmLine: boolean: true
                EnableAlarmChecking : boolean: true
                EnableAlarmLineText: String: true
             LCL2
                LineColor: Color String constant
                TextColor: Color String constant
                LineWidth: double: 1
                LimitValue: double: 0
```

```
   DisplayString: String: "LCL"
   EnableAlarmLine: boolean: true
   EnableAlarmChecking : boolean: true
   EnableAlarmLineText: String: true
UCL2
   LineColor: Color String constant
   TextColor: Color String constant
   LineWidth: double: 1
   LimitValue: double: 0
   DisplayString: String: UCL
   EnableAlarmLine: boolean: true
   EnableAlarmChecking : boolean: true
   EnableAlarmLineText: String: true
LCL1
   LineColor: Color String constant
   TextColor: Color String constant
   LineWidth: double: 1
   LimitValue: double: 0
   DisplayString: String: "LCL"
   EnableAlarmLine: boolean: true
   EnableAlarmChecking : boolean: true
   EnableAlarmLineText: String: true
UCL1
   LineColor: Color String constant
   TextColor: Color String constant
   LineWidth: double: 1
   LimitValue: double: 0
   DisplayString: String: UCL
   EnableAlarmLine: boolean: true
   EnableAlarmChecking : boolean: true
   EnableAlarmLineText: String: true
123SigmaControlLimits
   Target: double: 0
   LCL3Value: double: 0
   UCL3Value: double: 0
   AlarmTest12: boolean: true
   EnableAlarmLine: boolean: true
   EnableAlarmChecking: boolean: true
   EnableAlarmLineText: boolean: true
NamedRuleSet
      RuleSet: SPC string constant
      RuleEnable [boolean, boolean …]
      CustomizeRules: [   {
                                "RuleNumber": 15,
                                "M": 18,
                                "N": 15
                          },

                          {     "RuleNumber": 15,
                                "M": 18,
                                "N": 15
                          },
                          ...
                      ]
```

Font

Specifies the Font used to annotate the control limits on the RHS of the chart.

```
Font
   Name: String: "sans-serif"
   Size: double: 12
   Style: SPC String Constant: "PLAIN"
```

Name

Set the axis labels font family to something other than the default "sans-serif". Follow the font naming guidelines detailed in Static Properties chapter, under DefaultTableFont

Size

Font size in points.

Style

Use of the style string constants: "Plain", "Normal", "Bold","Italic","Bold Italic".

Example

```
"Font": {
      "Size":  10,
      "Style": "PLAIN"
     },
```

DefaultLimits

```
DefaultLimits [ boolean: true, boolean: true]
```

Specifies whether default UCL and LCL limits are enabled.

DefaultLimits is an array of two booleans.

DefaultLimits[0]

Set to false to disable the checking of the default +- 3 Sigma limits, also known as UCL (upper control limits) and LCL (lower control limit).

DefaultLimits[1]

Set to false to disable drawing the +- 3 Sigma limits lines and associated text.

Example

```
"DefaultLimits": [false, false],
```

SetLimits

```
SetLimits: [target: double:0 , lcl: double: 0, hdl: double: 0]
```

A quick way to set the three limits for a chart: Target, LCL (low control limit) and UCL (upper control limit).

Example

```
"SetLimits":[30.0, 25.0, 35.0],
```

Decimal

```
Decimal: integer: 1
```

Decimal
Set to the decimal precision to display the limit values.

Example

```
"Decimal":2,
```

ZoneFill

```
ZoneFill: boolean: false
```

Set to true and the area between the control limit lines are filled with a solid color

ZoneColors

```
ZoneColors:  [
                Color String constant: "ORANGERED",
                Color String constant: "LIGHTGOLDENRODYELLOW",
                Color String constant: "LIGHTGREEN"
             ]
```

Set to true and the area between the control limit lines are filled with a solid color

```
"ZoneColors": ["ORANGERED", "LIGHTGOLDENRODYELLOW","LIGHTGREEN"]
```

Out of Limit Limit Symbol

```
OutOfLimitSymbolSize: integer: 14
OutOfLimitSymbolType: SPC String Contants: "CIRCLE"
```

OutOfLimitSymbolSize

Specify the size of the symbol in points (1..64)

OutOfLimitSymbolType

Specify the symbol type using one of the SPC shape string constants: "SQUARE", "UPTRIANGLE", "DOWNTRIANGLE", "DIAMOND", "PLUS", "CROSS", "STAR" and "CIRCLE"

Note: The OutOfLimitSymbolSize and OutOfLimitSymbolType properties are also found directly under the PrimaryChartSetup, SecondaryChartSetup and ThirdChartSetup blocks, without the need to include the ControlLimit block. All of our examples though place these two properties under a ControlLimits block.

Target

```
Target
   LineColor: Color String constant: "GREEN"
   TextColor: Color String constant: "BLACK"
   LineWidth: double: 1
   LimitValue: double: 0
   DisplayString: String: "XBAR"
   EnableAlarmLine: boolean: true
   EnableAlarmChecking : boolean: true
```

Set the properties associated with the Target line

LineColor

The line color of the limit line.

TextColor

The text color of the limit line label.

LineWidth

The line width of the limit line

LimitValue

Set the limit value.

DisplayString

Set the text string which precedes the numeric value of the limit line label.

EnableAlarmLine

Set to false to disable the alarm line.

EnableAlarmChecking

Set to false to disable the limit testing against the limit value.

Example

```
"Target": {
            "DisplayString": "TargetXX",
            "EnableAlarmLine": true,
            "EnableAlarmChecking": true,
            "LimitValue": 30,
            "EnableAlarmLineText": true
          },
```

LCL3, LCL2, LCL1

```
LCL3
   LineColor: Color String constant
   TextColor: Color String constant
   LineWidth: double: 1
   LimitValue: double: 0
   DisplayString: String: "LCL"
   EnableAlarmLine: boolean: true
   EnableAlarmChecking : boolean: true
   EnableAlarmLineText: String: true


LCL2 and LCL1 are the same.
```

Set the properties associated with the Lower Control Limit (LCL3, LCL2, LCL1) lines. The LCL3 line represents the - 3-Sigma limit for the chart. For charts with 2- and 1-Sigma limits, LCL2 = - 2 Sigma limit and LCL1= - 1-Sigma limit.

LineColor

The line color of the limit line.

TextColor

The text color of the limit line label.

LineWidth
The line width of the limit line

LimitValue
Set the limit value.

DisplayString
Set the text string which precedes the numeric value of the limit line label.

EnableAlarmLine
Set to false to disable the alarm line.

EnableAlarmChecking
Set to false to disable the limit testing against the limit value.

EnableAlarmLineText
Set to false to disable the limit line text on the right.

Example

```
          "LCL3": {
           "DisplayString": "LCLXX",
           "EnableAlarmLine": true,
           "EnableAlarmChecking": true,
           "LimitValue": 25,
           "EnableAlarmLineText": false
        },
```

UCL3, UCL2, UCL1,

```
UCL3
   LineColor: Color String constant
   TextColor: Color String constant
   LineWidth: double: 1
   LimitValue: double: 0
   DisplayString: String: "LCL"
   EnableAlarmLine: boolean: true
   EnableAlarmChecking : boolean: true
   EnableAlarmLineText: String: true

UCL2 and UCL1 are the same.
```

Set the properties associated with the Upper Control Limit (UCL3, UCL2, UCL1) lines. The UCL3 line represents the + 3-Sigma limit for the chart. For charts with 2- and 1-Sigma limits, UCL2 = + 2 Sigma limit and UCL1= + 1-Sigma limit.

LineColor

The line color of the limit line.

TextColor

The text color of the limit line label.

LineWidth

The line width of the limit line

LimitValue

Set the limit value.

DisplayString

Set the text string which precedes the numeric value of the limit line label.

EnableAlarmLine

Set to false to disable the alarm line.

EnableAlarmChecking

Set to false to disable the limit testing against the limit value.

EnableAlarmLineText

Set to false to disable the limit line text on the right.

Example

```
"UCL3": {
   "DisplayString": "UCLXX",
   "EnableAlarmLine": false,
   "EnableAlarmChecking": true,
   "LimitValue": 35,
   "EnableAlarmLineText": true
}
```

123SigmaControlLimits

```
123SigmaControlLimits
   Target: double: 0
   LCL3Value: double: 0
   UCL3Value: double: 0
   AlarmTest12: boolean: true
   EnableAlarmLine: boolean: true
   EnableAlarmChecking: boolean: true
   EnableAlarmLineText: boolean: true
```

This method will display control limits for +-1, +-2, and +-3 Sigma, given the target value and the LCL3 value and the UCL3 value.

Target

Specify the target (centerline) value for chart.

LCL3Value

Specify the LCL3 (- 3-sigma low control limit) value for chart.

UCL3Value

Specify the UCL3 (+ 3-sigma upper control limit) value for chart.

AlarmTest12

Set to true if you want limit testing (1 out of one outside limit) for +-1 and +-2 sigma control limit lines.

EnableAlarmLine

Set to false to disable the limit lines

EnableAlarmChecking

Set to false to disable the limit testing

EnableAlarmLineText

Set to false to disable the limit line text

Example

```
"123SigmaControlLimits": {
      "Target":  30,
      "LCL3Value": 25,
      "UCL3Value": 30,
      "AlarmTest12": true ,
      "EnableAlarmLine": true,
      "EnableAlarmChecking":  true,
      "EnableAlarmLineText":  true
}
```

NamedRuleSet

```
NamedRuleSet
      RuleSet: string constant
      RuleEnable [boolean, boolean …]
      CustomizeRules: [   {
```

```
                                "RuleNumber": 15,
                                "M": 18,
                                "N": 15
                            },

                            {   "RuleNumber": 15,
                                "M": 18,
                                "N": 15
                            },
                                          ...
                      ]
```

The NameRuleSet property will invoke a complete set of control rules based one of following standard rule sets: BASIC_RULES, WECO_RULES,WECOANDSUPP_RULES, NELSON_RULES,AIAG_RULES, JURAN_RULES, HUGHES_RULES,GITLOW_RULES, WESTGARD_RULES,and DUNCAN_RULES. For a complete discussion of named control rules, see chapter xxxx.

RuleSet

One of the named rule identifiers: BASIC_RULES, WECO_RULES,WECOANDSUPP_RULES, NELSON_RULES,AIAG_RULES, JURAN_RULES, HUGHES_RULES,GITLOW_RULES, WESTGARD_RULES,and DUNCAN_RULES.

RuleEnable

An array of boolean, one for each named rule in the rule set. All of the rules are enabled by default. This permits you to disable specific rules.

CustomizeRules

An array, one for each rule you want to modify in the ruleset. Each block in the array contains the rule number, the M-value (N out of M must exceed the limit value for a violation to occur) and an N-value.

RuleNumber

The rule number (our rule number)

M

The M-value (N out of M must exceed the limit value for a violation to occur)

N

The N-value (N out of M must exceed the limit value for a violation to occur)

Example

```
"NamedRuleSet":
```

```
{
  "RuleSet": "WECO_RULES",
  "RuleEnable": [ true, true, false, true, false, true, true, true],
  "CustomizeRules: [   {
                             "RuleNumber": 3,
                             "M": 2,
                             "N": 1
                       }
                   [
}
```

AddControlRules

```
AddControlRules
          [  {
                RuleSet: : SPC String constant: "BASIC_RULES"
                   RuleNumber: integer: 2
                   EnableAlarmLine: boolean: true
                   EnableAlarmChecking: boolean: true
                   EnableAlarmLineText: String: true
                   M: integer: 1
                   N: integer: 1
             },
             {
                RuleSet: : SPC String constant: "BASIC_RULES"
                   RuleNumber: integer: 2
                   EnableAlarmLine: boolean
                   EnableAlarmChecking: boolean
                   EnableAlarmLineText: String
                   M: integer: 1
                   N: integer: 1
             }, ...

          ]
```

The AddControlRules property is an array of control rule specifications. Since it is an array, you can add as many control rules as you want. Each specification block in the array defines one control rule. Note how the control rule array is bracketed by [ ], signifying the start and and of the array. Each block element in the array is bracketed using { }.

A control rule block element has the following parameters:

RuleSet

One of the named rule identifiers: BASIC_RULES, WECO_RULES,WECOANDSUPP_RULES, NELSON_RULES,AIAG_RULES, JURAN_RULES, HUGHES_RULES,GITLOW_RULES, WESTGARD_RULES, DUNCAN_RULES and CUSTOM_TEMPLATE_BASED_RULE.

RuleNumber

The rule number (our rule number). If the RuleSet is of type CUSTOM_TEMPLATE_BASED_RULE, then the RuleNumber specifies the template number of the desired template.

EnableAlarmLine

Enable the drawing of the limit line for the control rule.

EnableAlarmChecking

Enable alarm checking for the the control rule.

EnableAlarmLineText

Enable the drawing of the limit text for the control rule.

M

The M-value (N out of M must exceed the limit value for a violation to occur,)

N

The N-value (N out of M must exceed the limit value for a violation to occur)

If the RuleSet is CUSTOM_TEMPLATE_BASED_RULE, the following parmeters are also valid:

SigmaLevel

The sigma level of the desired control rule template.

A multi-rule example would look something like:

```
"AddControlRules": [
      {
          "RuleSet": "WECO_RULES",
          "RuleNumber":  2
      },
      {
          "RuleSet": "WECO_RULES",
          "RuleNumber":  3
      },
      {
          "RuleSet": "NELSON_RULES",
          "RuleNumber":  12
      },
      {
          "RuleSet": "JURAN_RULES",
          "RuleNumber":  9,
          "EnableAlarmLine": false,
          "EnableAlarmChecking": true,
          "EnableAlarmLineText": false
      }
]
```

SpecifyControlLimitsUsingMeanAndSigma

```
SpecifyControlLimitsUsingMeanAndSigma
   Mean: double: 1
   Sigma: double: 1
```

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 invoke SpecifyControlLimitsUsingMeanAndSigma method. It will run through all of the control limit records and fill out the appropriate limit values and other critical parameters.

Mean

Specify the process mean.

Sigma

Specify the process sigma.

The center line value and sigma have different meanings for the Primary and Secondary charts. So the **SpecifyControlLimitsUsingMeanAndSigma** 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.

```
"SpecifyControlLimitsUsingMeanAndSigma": {
   "Mean":  30,
   "Sigma": 1.666
}
```

SpecificationLimits

```
SpecificationLimits
   Font
      Name: String: "sans-serif"
      Size: double: 12
      Style: SPC String Constant: "PLAIN"
   Decimal: integer: 1
   LowSpecificationLimit
      LineColor: Color String constant: "BLUE"
     TextColor: Color String constant: "BLACK"
      LineWidth: double: 1
      LimitValue: double: 0
      DisplayString: String: "LSL"
      EnableAlarmLine: boolean: true
      EnableAlarmChecking : boolean: true
      EnableAlarmLineText: String: true
   HighSpecificationLimit: double
      LineColor: Color String constant: "RED"
      TextColor: Color String constant: "BLACK"
      LineWidth: double: 1
      LimitValue: double: 0
      DisplayString: String: "USL"
      EnableAlarmLine: boolean: true
      EnableAlarmChecking : boolean: true
```

Font

Specifies the Font used to annotate the control limits on the RHS of the chart.

```
Font
   Name: String: "sans-serif"
   Size: double: 12
   Style: SPC String Constant: "PLAIN"
```

Name

Set the axis labels font family to something other than the default "sans-serif". Follow the font naming guidelines detailed in Static Properties chapter, under DefaultTableFont

Size

Font size in points.

Style

Use of the style string constants: "Plain", "Normal", "Bold","Italic","Bold Italic".

Decimal

```
Decimal: integer: 1
```

Decimal

Set to the decimal precision to display the limit values.

LowSpecificationLimit

```
LowSpecificationLimit
   LineColor: Color String constant: "BLUE"
   TextColor: Color String constant: "BLACK"
   LineWidth: double: 1
   LimitValue: double: 0
   DisplayString: String: "LSL"
   EnableAlarmLine: boolean: true
   EnableAlarmChecking : boolean: true
   EnableAlarmLineText: String: true
```

Set the properties associated with the Low Specification Limit (LSL) line

LineColor
The line color of the limit line.

TextColor
The text color of the limit line label.

LineWidth
The line width of the limit line

LimitValue
Set the limit value.

DisplayString
Set the text string which precedes the numeric value of the limit line label.

EnableAlarmLine
Set to false to disable the alarm line. False also excludes the limit from the auto-scaling calculations.

EnableAlarmChecking
Set to false to disable the limit testing against the limit value.

EnableAlarmLineText
Set to false to disable the limit line text on the right.

Example

```
"SpecificationLimits":
{
    "LowSpecificationLimit":
    {
       "LimitValue": 15,
       "LineColor": "BLUE",
       "DisplayString": "LSLX"
    },
   "HighSpecificationLimit":
    {
       "LimitValue": 40,
       "LineColor": "RED",
       "DisplayString": "HSLX"
    }
}
```

HighSpecificationLimit

```
HighSpecificationLimit
   LineColor: Color String constant: "RED"
   TextColor: Color String constant: "BLACK"
   LineWidth: double: 1
   LimitValue: double: 0
   DisplayString: String: "USL"
   EnableAlarmLine: boolean: true
   EnableAlarmChecking : boolean: true
```

Set the properties associated with the High Specification Limit (HSL) line

LineColor
The line color of the limit line.

TextColor
The text color of the limit line label.

LineWidth
The line width of the limit line

LimitValue
Set the limit value.

DisplayString
Set the text string which precedes the numeric value of the limit line label.

EnableAlarmLine
Set to false to disable the alarm line. False also excludes the limit from the auto-scaling calculations.

EnableAlarmChecking
Set to false to disable the limit testing against the limit value.

EnableAlarmLineText
Set to false to disable the limit line text on the right.

Example

```
"SpecificationLimits":
{
    "LowSpecificationLimit":
    {
      "LimitValue": 15,
      "LineColor": "BLUE",
      "DisplayString": "LSLX"
    },
   "HighSpecificationLimit":
    {
      "LimitValue": 40,
      "LineColor": "RED",
      "DisplayString": "HSLX"

    }
}
```

SecondaryChartSetup

```
SecondaryChartSetup
```

The SecondaryChartSetup is same as PrimaryChartSetup, except that there is no NamedRuleSet block. You should not try and use any of the named rules, either individually, or in a set, in a secondary chart, because they were never intended for use in a secondary chart. All named control rules apply to the tracking of the measured variable in the Primary chart.

ThirdChartSetup

```
ThirdChartSetup
```

The ThirdChartSetup (only used in the Xbar-R-MR chart types) is same as PrimaryChartSetup, except that there is no NamedRuleSet block. You should not try and use any of the named rules, either individually, or in a set, in a third chart, because they were never intended for use in a third chart. All named control rules apply to the tracking of the measured variable in the Primary chart.

8. Adding Data to an SPC Chart

```
SPCChart
   SampleData
      SampleIntervalRecords
      DataSimulation
         StartCount: integer: 0
         Count: integer: 20
         Mean: double: 1
         Range: range: 1
      ExcludeRecords: [integer, integer, ..]
      IncludeRecords: [integer, integer, ..]
      ResetSPCChartData

Alarm Forcing features found only in QCSPCChart+
      PrimaryForceAlarm: SPC string constant: "NO_FORCING"
      SecondaryForceAlarm: SPC string constant: "NO_FORCING"
      ThirdForceAlarm: SPC string constant: "NO_FORCING"

Multi-Variable chart features found only in QCSPCChart+
      MultiVariableSampleValues1: [double, double,... ]
      MultiVariableSampleValues2: [double, double,... ]
      MultiVariableSampleValues3: [double, double,... ]
      MultiVariableSampleValues4: [double, double,... ]
      MultiVariableSampleValues5: [double, double,... ]
      MultiVariablePrimaryForceAlarm: [SPC string constant: "NO_FORCING",
                                       SPC string constant: "NO_FORCING",
                                       …]
      MultiVariableSecondaryForceAlarm: [SPC string constant: "NO_FORCING",
                                       SPC string constant: "NO_FORCING",
                                       …]
      MultiVariableThirdForceAlarm: [SPC string constant: "NO_FORCING",
                                       SPC string constant: "NO_FORCING",
                                       …]
```

A SPC Chart can be defined prior to adding any data to it. In that case, you will have to at least specify reasonable values for the control limits. Otherwise, the chart auto-scaling of the y-axes will not work properly, because the chart will have nothing to auto-scale against. Since you have no data, you can't use the AutoScaleControlLimits, since that requires that data already be entered into the chart. The best way to establish some default values for the control limits is to use the SpecifyControlLimitsUsingMeanAndSigma property, discussed in the previous chapter. That will set values for all of the sigma-based control limits, and establish some initial y-scaling values.

Data is added to the charting using the SampleData property. It supports the following options:

- One or more sample interval records, using an array structure.

- Each sample interval record has the following properties:
- TimeStamp
- BatchCount
- Note
- Batch ID string
- Sample sub group size for variable control limits
- Sample values: an array of numeric values, one for each sample in the sample subgroup. If the variable control chart is a fixed sample size per subgroup charts, you can still enter less than the target samples per subgroup. You can also enter more values than the specified number of samples per subgroup. However, if you enter more values than the specified number of samples per subgroup, the extra data values are not saved, once the summary statistics for the sample interval (mean, range, standard deviation, etc.) are calculated. So chart options which display the individual data values for the sample subgroup, in the table or as scatter plots symbols in the chart, will not display the discarded data. A record without any data values is marked invalid and a hole will appear in the chart at that point.
- For the multi-variable chart types (QCSPCChart+), up to five additional sample values blocks can be added, one block for each of the variables added to the main variable of a multi-variable chart. The added sample blocks for the added multi-variables have the names MultiVariableSampleValues1 .. MultiVariableSampleValues5. You only need to include the blocks you need for a given multi-variable chart type, with the specified number of added variables. Similar to the Sample value rules described above, the multi-variable data for a given sample interval can include data for 1 to 25 sample points. A record without any data values is marked invalid and a hole will appear in the chart at that point.
- The alarm state for an sample interval of the main variable, and any added variables, can be forced to a given state, reflected by the attributes of the symbol color, size and shape used in the display of the data point for the given sample interval.
- Control limits and Specifications can be changed on the fly, and specified for each sample interval
- Data simulation for all chart types
- Exclude one or more sample records from control limit calculations
- Include (or re-include previous excluded) sample records
- Reset SPC Data

SampleIntervalRecords

```
SampleData
   SampleIntervalRecords
   [  {
         TimeStamp: double:
         BatchCount: integer: 1
         Note: String: ""
         BatchIDString: String: ""
         VariableControlLimits: [double:1,double:1, ...]
         VariableSpecificaitonLimits: [double:1,double:1, ...]
         SampleSubgroupSize_VSS: integer: -1
```

```
             SampleValues [double, double,... ]

Variable Specification Limits features found only in QCSPCChart+
             VariableSpecificationLimits: [double:1,double:1, ...]


Main Variable Alarm Forcing features found only in QCSPCChart+
             PrimaryForceAlarm: SPC string constant: "NO_FORCING"
             SecondaryForceAlarm: SPC string constant: "NO_FORCING"
             ThirdForceAlarm: SPC string constant: "NO_FORCING"

Multi-Variable chart features found only in QCSPCChart+
             MultiVariableSampleValues1: [double, double,... ]
             MultiVariableSampleValues2: [double, double,... ]
             MultiVariableSampleValues3: [double, double,... ]
             MultiVariableSampleValues4: [double, double,... ]
             MultiVariableSampleValues5: [double, double,... ]

Multi-Variable Alarm Forcing features found only in QCSPCChart+
             MultiVariablePrimaryForceAlarm: [SPC string constant: "NO_FORCING",
                                              SPC string constant: "NO_FORCING",
                                              ...]
             MultiVariableSecondaryForceAlarm: [SPC string constant: "NO_FORCING",
                                              SPC string constant: "NO_FORCING",
                                              ...]
             MultiVariableThirdForceAlarm: [SPC string constant: "NO_FORCING",
                                              SPC string constant: "NO_FORCING",
                                              ...]


          },
          {
             TimeStamp: double:
             BatchCount: integer: 2
             Note: String: ""
             BatchIDString: String: ""
             VariableControlLimits: [double:1,double:1, ...]
             VariableSpecificationLimits: [double:1,double:1, ...]
             SampleSubgroupSize_VSS: integer: -1
             SampleValues [double, double,... ]

             Variable Specification Limits features found only in QCSPCChart+
             Main Variable Alarm Forcing features found only in QCSPCChart+
             Multi-Variable chart features found only in QCSPCChart+
             Multi-Variable Alarm Forcing features found only in QCSPCChart+
          },
       ]
```

TimeStamp

Since there is no reliable standard across browsers for time/date data, this value is expressed as the Unix standard of elapsed milliseconds since Thursday, 1 January 1970. The TimeStamp positions the sample data on the x-axis for time-based SPC charts. Not so for batch-based SPC charts. There the sample data is positioned on the x-axis using the batch number. In batch-based SPC charts, the time stamp value for each batch (in regular HH:MM:SS format) can optionally be displayed in the table above the charts, and they can also be used as tick mark labels, replacing the batch number labels.

Javascript Date Object

In Javascript, you can process time/date values using the Javascript Date object. The standard Javascript constructors for the Date class are:

```
new Date() // current date and time
new Date(value) //milliseconds since 1970/01/01
new Date(dateString)
new Date(year, month, day, hours, minutes, seconds, milliseconds)
```

where:

value Integer value representing the number of milliseconds since 1 January 1970 00:00:00 UTC (Unix Epoch).

dateString String value representing a date. The string should be in a format recognized by the Date.parse() method (IETF-compliant RFC 2822 timestamps and also a version of ISO8601).

year Integer value representing the year. The year must always be provided in full (e.g. 98 is treated as 98, not 1998).

month Integer value representing the month, beginning with 0 for January to 11 for December.

day Integer value representing the day of the month.

hour Integer value representing the hour of the day.

minute Integer value representing the minute segment of a time.

second Integer value representing the second segment of a time.

millisecond Integer value representing the millisecond segment of a time.

Most parameters above are optional. Not specifying, causes 0 to be passed in. Once a Date object is created, a number of methods allow you to operate on it. Most methods allow you to get and set the year, month, day, hour, minute, second, and milliseconds of the object, using either local time or UTC (universal, or GMT) time. All dates are calculated in milliseconds from 01 January, 1970 00:00:00 Universal Time (UTC) with a day containing 86,400,000 milliseconds. Some examples of initiating a date:

```
var today = new Date();
var d1 = new Date("November 15, 2013 11:13:00");
var d2 = new Date(2013,10,15);
var d3 = new Date(2013,10,15,11,13,0);
```

When converting a Date object into a value for use as a time stamp in one of our JSON scripts, just use the Date objects getTime function.

```
var timestamp = d3.getTime();
```

If your charts JSON definition was in the `MediumSimpleDataUpdateObject object, then you can set the time stamp using code similar to below:`

```
MediumSimpleDataUpdateObject.SPCChart.SampleData.SampleIntervalRecords[i].TimeStamp =
timestamp;
```

In our SPCMediumSimple.html example, we use the following code to calculate a new time stamp which is:

```
var sampleintervalmilliseconds = 900000;

var timestamp = new
MediumSimpleDataUpdateObject.SPCChart.SampleData.SampleIntervalRecords[i].TimeStamp +
sampleintervalmilliseconds);

MediumSimpleDataUpdateObject.SPCChart.SampleData.SampleIntervalRecords[i].TimeStamp =
timestamp.getTime();
```

where the previous time stamp is retrieved from the chart update JSON script, a new Date object created with it by adding in an additional 900000 milliseconds, representing 15 minutes (1000 * 60 * 15 = 90000). The new Date is then written back to the JSON records as the getTime() value, which returns milliseconds.

This is exactly the same as the following code, which does not use the Date object at all:

```
MediumSimpleDataUpdateObject.SPCChart.SampleData.SampleIntervalRecords[i].TimeStamp
+=  sampleintervalmilliseconds;
```

Since Date is a standard Javascript object, you will find countless examples of how to manipulate time/date values on the web. Here are a couple of tutorials:

http://msdn.microsoft.com/en-us/library/ie/ee532932(v=vs.94).aspx
http://www.techrepublic.com/article/manipulate-time-and-date-values-with-javascripts-date-object/

Special Note on the use of Time Stamps
If you are using a **Time-based SPC Chart**, then you MUST specify time stamps which are monotonic and evenly spaced. Monotonic means that the values always increase. You can' t enter data from today, followed by data from yesterday. That is going backward in time and is non-monotonic. The Time-based control charts also require that the time stamps increase at a regular rate, i.e. 15 minutes. In time stamp units (milliseconds), this would be an increase of (15 * 60 * 1000 = 900000) milliseconds per sample interval. While this time stamp increment does not have to be exact, it should be close, or else the data plotted in the SPC chart will not line up with the table. You can't enter data from an 8-hour run yesterday, followed by an 8-hour run today. That would leave large gaps in the chart. If you have irregular time stamp data, you must use the Batch-based SPC Chart type, which ignores the time stamp when positioning data points in a chart. See the discussion of InitChartProperties in Chapter 5, SPC

Initial Chart Setup. The Batch-based carts are more flexible than the Time-based charts, and we recommend everyone use them.

BatchCount

The batch number of the associated sample interval record. The BatchCount value is used if you are using a batch-based control chart. BatchCount values must be monotonic, meaning they always increase. You **cannot** enter in a group of sample intervals from today, with batch numbers 100-200, followed by another group from yesterday with batch numbers 100-200. It is up to you to sort the batch data into the proper order.

If you don't think you can keep track of the BatchCount number, don't specify it. The value will automatically be assigned a value equal to the current number of sample interval records currently in the system. That will result in valid values.

BatchNumber

Same as BatchCount.

Note

A note which can be attached to the sample interval record, and displayed in the notes field of a sample interval record in the data table.

BatchIDString

A batch ID string can be associated with a sample interval. This is used with batch-based control charts. Instead of labeling the x-axis tick marks with the BatchCount, or the TimeStamp, you can lable it with the BatchIDString.

VariableControlLimits: [double, double, …]

The initial control limits are specified in the main setup portion of the charts JSON setup, under the *ControlLimits* block. You can set the limits manually, or use our auto-calculation methods to calculate the control limits based on the existing data. Those control limits will remain in effect until changed. You can change them by specifying new control limits in the VariableControlLimits array. You do not need to specify VariableControlLimits for each SampleInterval array element though. Control limits will remain in effect until changed. The order of the values in the VariableControlLimits array is [Primary target, Primary LCL3, Primary UCL3, Secondary target, Secondary LCL3, Secondary UCL3]. Other limits are ignored. This method only works if you are working with the default +-3 Sigma control limits (+ targets) for the Primary and Secondary charts.

```
"SampleData": {
             "SampleIntervalRecords": [{
                "SampleValues": [
                   27.53131515148628,
                   33.95771604022404,
                   24.310097827061817,
                   28.282642847792765,
                   30.2908518818265
                ],
                "BatchCount": 0,
                "TimeStamp": 1371830829074,
                "Note": ""
             }, {
                "SampleValues": [
                   27.444285005240214,
                   34.38930645615096,
                   28.0203674441636,
                   33.27153359969366,
                   36.8305571558275
                ],
                "BatchCount": 1,
                "TimeStamp": 1371831729074,
                "Note": ""
             }, {
                "SampleValues": [
                   35.21321620109259,
                   32.93940741018088,
                   33.66485557976163,
                   34.17314124609133,
                   24.576683179863725
                ],
                "VariableControlLimits": [31, 26, 36, 5, 1, 11],
                "BatchCount": 2,
                "TimeStamp": 1371832629074,
                "Note": ""
             },
```

SampleSubgroupSize_VSS: integer: -1

SampleValues [double, double,...]

An array of numeric values corresponding to the sample data for a sample interval. The meaning of the data is specific to the SPC chart type (Variable or Attribute, fixed or variable sample size per sample interval), so you must take that into account.

SampleValues for Variable Control Charts with a fixed 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, MA_CHART, LEVEY_JENNINGS_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.

```
   "SampleData": {
```

```
"SampleIntervalRecords": [
{
    "SampleValues": [
        27.53131515148628,
        33.95771604022404,
        24.310097827061817,
        28.282642847792765,
        30.2908518818265
    ],
    "BatchCount": 0,
    "TimeStamp": 1371830829074,
    "Note": ""
},
{
    "SampleValues": [
        27.444285005240214,
        34.38930645615096,
        28.0203674441636,
        33.27153359969366,
        36.8305571558275
    ],
    "BatchCount": 1,
    "TimeStamp": 1371831729074,
    "Note": ""
},
```

In an Individual-Range chart, which by definition samples 100% of the production level, the SampleValues 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 SampleValues array.

SampleValues for Variable Control Charts with a variable 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 SampleValues array for each measurement sample in the sample subgroup interval. The difference is that the length of the SampleValues array can change from update to update.
We often hear from those using the X-Bar Sigma as the most universal of the Variable Control Charts. They feed in variable amounts of data for each sample interval from 1 item, to over 20 items. This is not a valid use of the X-Bar Sigma chart with variable sample size. An X-Bar sigma chart requires that you have at least 10-15 samples per sample interval. So don't use it where you expect a low number of samples in a given sample interval. This is for statistical reasons. For point counts less than 10, using the range of values within a sample sub-interval is a better way to estimate the process standard deviation value, than the actual standard deviation for the same sample interval.

```
"SampleData": {
       "SampleIntervalRecords": [
           {
               "BatchCount": 0,
               "TimeStamp": 1374768344076,
               "Note": "",
               "SampleValues": [
                   27.908233086630105,
                   31.940402816755082,
                   27.55563653827735,
                   30.083357069668647,
                   33.642341315640614,
                   28.72361222739654,
                   32.099133969171135,
                   26.356050567985285,
                   29.31049201044222,
                   28.499216431790145,
                   31.04435971419966,
                   33.2381471474555,
                   31.589757172384306,
                   32.438862725116614,
                   31.53988206474448
               ]
           },
           {
               "BatchCount": 1,
               "TimeStamp": 1374769244076,
               "Note": "",
               "SampleValues": [
                   28.749312830468913,
                   26.404064179124912,
                   33.28922196391206,
                  31.694060174687134,
                   26.90641611483808,
                   31.48391922918815,
                   30.209803672319776,
                   29.86474359585655,
                   32.53515676358969,
                   29.12428709515284,
                   29.75699541053711,
                   31.017198158995903,
                   30.224697293173516
               ]
           },
           {
               "BatchCount": 2,
               "TimeStamp": 1374770144076,
               "Note": "",
               "SampleValues": [
                   26.801870534836045,
                   27.378134477854075,
                   33.08638714532048,
                   31.769229875222915,
                   32.0028978203766,
                   27.226930046247567,
                   28.687810832891774,
                   33.60598037448174,
                   29.262192232690143,
                   32.27140254488991,
                   32.70426215916774,
                   32.02023454576835
               ]
           },
```

SampleValues for Attribute Control Charts (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
DPMO =	NUMBER_DEFECTS_PER_MILLION_CHART

In attribute control charts, the meaning of the data in the *SampleValues* 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 **InitChartProperties** initialization property block, the first N-1 (0.. N-2) elements of the *SampleValues* array holds the defect count for each category. The (N)th (or element N-1 in the array) element of the *SampleValues* 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 *SampleValues* array sized to six, as in the code below:

```
"SampleData": {
       "SampleIntervalRecords": [
           {
               "BatchCount": 0,
               "TimeStamp": 1374768344076,
               "Note": "",
               "SampleValues": [
                   3,
                   0,
                   4,
                   2,
                   4
               ]
           },
           {
               "BatchCount": 1,
               "TimeStamp": 1374769244076,
               "Note": "",
               "SampleValues": [
                   2,
                   2,
                   1,
                   4,
                   6
               ]
           },
           {
               "BatchCount": 2,
               "TimeStamp": 1374770144076,
               "Note": "",
               "SampleValues": [
                   3,
                   1,
                   3,
                   2,
                   5
               ]
           },
```

Updating p-charts (Variable Sample Subgroup Size)

p-chart = FRACTION_DEFECTIVE_PARTS_CHART_VSS,

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.

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

```
"SampleData": {
       "SampleIntervalRecords": [
           {
               "BatchCount": 0,
               "TimeStamp": 1374768344076,
               "Note": "",
               "SampleSubgroupSize_VSS": 5,
               "SampleValues": [
                   3,
                   0,
                   4,
                   2,
                   3
               ]
           },
           {
               "BatchCount": 1,
               "TimeStamp": 1374769244076,
               "Note": "",
               "SampleSubgroupSize_VSS": 4,
               "SampleValues": [
                   2,
                   1,
                   4,
                   1
               ]
           },
           {
               "BatchCount": 2,
               "TimeStamp": 1374770144076,
               "Note": "",
               "SampleSubgroupSize_VSS": 6,
               "SampleValues": [
                   3,
                   1,
                   3,
                   2,
                   2,
                   4
               ]
           },
```

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.

```
"SampleData": {
       "SampleIntervalRecords": [
           {
               "BatchCount": 0,
               "TimeStamp": 1374768344076,
               "Note": "",
                "SampleSubgroupSize_VSS": 5,
               "SampleValues": [
                   3,
                   0,
                   4,
                   2,
                   3
               ]
           },
           {
               "BatchCount": 1,
               "TimeStamp": 1374769244076,
               "Note": "",
                "SampleSubgroupSize_VSS": 4,
               "SampleValues": [
                   2,
                   1,
                   4,
                   1
               ]
           },
           {
               "BatchCount": 2,
               "TimeStamp": 1374770144076,
               "Note": "",
                "SampleSubgroupSize_VSS": 6,
               "SampleValues": [
                   3,
                   1,
                   3,
                   2,
                   2,
                   4
               ]
           },
```

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.

Features Found only the Enhanced QCSPCChart+ Version

Fixed Variable Control Charts, with Extra or Missing data.
Variable Specification Limits features
Main Variable Alarm Forcing features
Multi-Variable chart features found
Multi-Variable Alarm Forcing features

Fixed Variable Control Charts, with Extra or Missing data

Fixed Variable Control Charts, with Extra or Missing data.

A special provision has been added to the data handling of the Primary Variable for fixed sample interval variable control charts. If the variable control chart uses a fixed number of samples per subgroup, you can still enter less than the target samples per subgroup. You can also enter more values than the specified number of samples per subgroup. However, if you enter more values than the specified number of samples per subgroup, the extra data values are not saved, once the summary statistics for the sample interval (mean, range, standard deviation, etc.) are calculated. So chart options which display the individual data values for the sample subgroup, in the table or as scatter plots symbols in the chart, will not display the discarded data. Also, if you use the auto-calculated control limits feature, the control limit calculation will NOT take into account variable sample size, and will instead use the original specified sample size for the chart. Only the Control Charts for Variable Sample Size, described in the next section do that. A record without any data values is marked invalid and a hole will appear in the chart at that point. The example below uses a fixed sample size of 5. But the SampleValues array contains, sample subgroups with a number of elements ranging from 0 to 8.

```
"SampleData": {
                          "SampleIntervalRecords": [
                          {
                              "SampleValues": [
                                  27.53131515148628,
                                  33.95771604022404,
                                  24.310097827061817,
                                  28.282642847792765,
                                  30.2908518818265
                              ],
                              "BatchCount": 0,
                              "TimeStamp": 1371830829074,
                              "Note": ""
                          },
                          {
                              "SampleValues": [
                                  27.444285005240214,
                                  34.38930645615096,
                                  28.0203674441636
                              ],
                              "BatchCount": 1,
                              "TimeStamp": 1371831729074,
                              "Note": ""
                          },
                          {
```

```
                          "SampleValues": [
                              27.444285005240214,
                              314.38930645615096,
                              227.53131515148628,
                              313.95771604022404,
                              254.310097827061817,
                              28.282642847792765,
                              30.2908518818265,
                              28.0203674441636,
                          ],
                          "BatchCount": 2,
                          "TimeStamp": 1371832729074,
                          "Note": ""
                      },
                      {
                          "SampleValues": [],
                          "BatchCount": 3,
                          "TimeStamp": 1371833729074,
                          "Note": ""
                      },
```

VariableSpecificationLimits: [double, double, …]

The initial specification limits are specified in the main setup portion of the charts JSON setup, under the *ControlLimits | SpecificationLimits* block. You must set the limits manually. You can have a high specification limit and a low specification limit. Or you can just have one or the other. But if you include a specification limit, you must assign a valid value to it, preferably one that matches the dynamic range of the data, else the specification limit will distort the charts auto-scaling of the y-axis. Specification limits will remain in effect until changed. You can change them by specifying new specification limits in the VariableSpecificationLimits array. You do not need to specify VariableSpecificationLimits for each SampleInterval array element though. Specification limits will remain in effect until changed. The order of the values in the VariableSpecificationLimits array is [Low Specification Limit, High Specification Limit]. If you don't have a Low Specification Limit in the chart, just enter a value of 0.0 for the first element of the array. See the example script TimeXBarRVariableSpecAndControlLimits in the chartdefEnhExample.js file.

```
"SampleData": {
             "SampleIntervalRecords": [{
                "SampleValues": [
                   27.53131515148628,
                   33.95771604022404,
                   24.310097827061817,
                   28.282642847792765,
                   30.2908518818265
                ],
                "BatchCount": 0,
                "TimeStamp": 1371830829074,
                "Note": ""
             }, {
                "SampleValues": [
                   27.444285005240214,
                   34.38930645615096,
         28.0203674441636,
         33.27153359969366,
         36.8305571558275
      ],
      "BatchCount": 1,
      "TimeStamp": 1371831729074,
      "Note": ""
   }, {
      "SampleValues": [
         35.21321620109259,
         32.93940741018088,
         33.66485557976163,
         34.1731412460913З,
         24.576683179863725
      ],
      "VariableSpecificationLimits": [13, 45],
      "BatchCount": 2,
      "TimeStamp": 1371832629074,
      "Note": ""
   },
```

PrimaryForceAlarm: SPC string constant: "`NO_FORCING`"

SecondaryForceAlarm: SPC string constant: "`NO_FORCING`"

ThirdForceAlarm: SPC string constant: "`NO_FORCING`"

These three properties apply to the main variable of the Primary Chart, Secondary Chart and in the case of Xbar-R-MR charts, the Third Chart of an SPC chart. They will force the alarm state (as indicated by the color, shaped and size of the symbol) to indicate a high alarm, low alarm, or normal state. If left at the default value of "NO_FORCING", no forcing is done, and the variable is left to its normally calculated value. The value values are: “NO_FORCING”, “FORCE_HIGH”, “FORCE_LOW”, and “FORCE_NORMAL”. The forcing values apply only to sample interval they are entered in. They do not carry forward into subsequent sample intervals. See the example script TimeXBarRForceOutOfLimit in the chartdefEnhExample.js file.

```
"SampleData": {
   "SampleIntervalRecords": [{
      "SampleValues": [
         27.53131515148628,
         33.95771604022404,
         24.310097827061817,
         28.282642847792765,
         30.2908518818265
      ],
      "BatchCount": 0,
      "TimeStamp": 1371830829074,
      "PrimaryForceAlarm": "FORCE_LOW",
      "SecondaryForceAlarm": "FORCE_HIGH",
      "Note": ""
   }, {
      "SampleValues": [
         27.444285005240214,
         34.38930645615096,
         28.0203674441636,
         33.27153359969366,
         36.8305571558275
      ],
      "BatchCount": 1,
      "TimeStamp": 1371831729074,
      "Note": ""
   },
```

MultiVariableSampleValues1: [double, double,...]

MultiVariableSampleValues2: [double, double,...]

MultiVariableSampleValues3: [double, double,...]

MultiVariableSampleValues4: [double, double,...]

MultiVariableSampleValues5: [double, double,...]

These five array properties are used to enter data into the added variables of a multi-variable chart (QCSPCChart+). The format is the same as the SampleValues array data block described earlier, which is used to enter the data for the main variable of the chart. If you have a Multi-Variable chart (a MULTIVARIABLE_MEAN_RANGE_CHART for example), and you have two added variables, then you would include MultiVariableSamples1 and MultiVariableSamples2, in your SampleIntervalRecords block. In this case, you must include these two array properties in every sample interval. But you do not need to include the other, unused array properties (MultiVariableSampleValues3, MultiVariableSampleValues4, and MultiVariableSampleValues5. See the example scripts MultiVariableBatchXBarR,MultiVariableBatchXBarRCustomized, MultiVariableBatchXBarRMR, MultiVariableBatchXBarSigma, MultiVariableBatchIndividualRange, and MixedMultiVariableBatch in the chartdefEnhExample.js file.

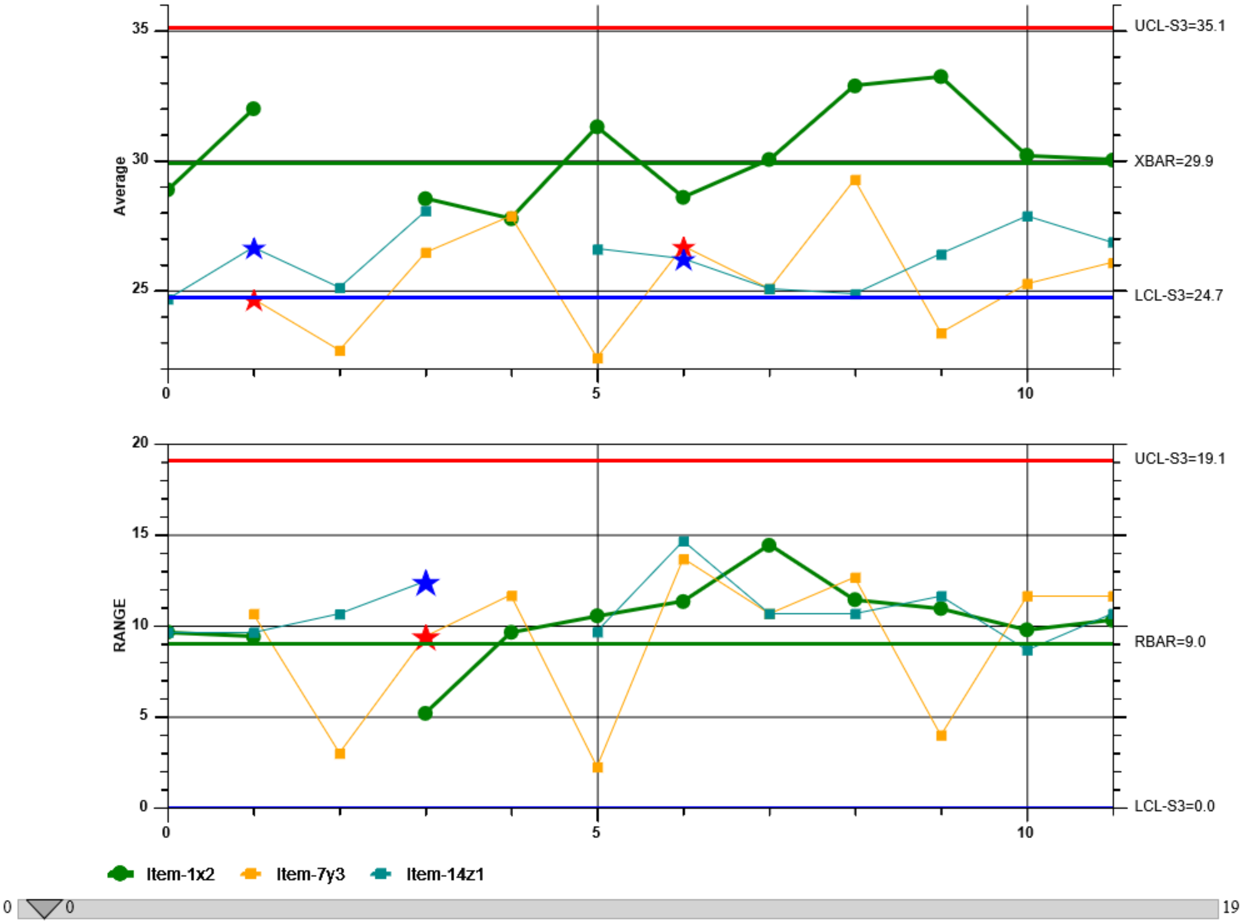
```
"SampleData": {
       "SampleIntervalRecords": [{
          "SampleValues": [
             27.53131515148628,
             33.95771604022404,
             24.310097827061817,
             28.282642847792765,
             30.2908518818265
          ],
           "MultiVariableSampleValues1":
             [
                21.53131515148628,
                31.95771604022404,
                22.310097827061817,
                22.282642847792765,
          21.2908518818265
       ],
    "MultiVariableSampleValues2":
       [
          22.53131515148628,
          30.95771604022404,
          21.310097827061817,
          22.282642847792765,
          26.2908518818265
       ],
   "BatchCount": 0,
   "TimeStamp": 1371830829074,
   "Note": ""
}, {
   "SampleValues": [
      27.444285005240214,
      34.38930645615096,
      28.0203674441636,
      33.27153359969366,
      36.8305571558275
   ],
   "MultiVariableSampleValues1":
      [
         23.53131515148628,
         31.95771604022404,
         23.310097827061817,
         23.282642847792765,
         21.2908518818265
      ],
    "MultiVariableSampleValues2":
      [
         24.53131515148628,
         32.95771604022404,
         23.310097827061817,
         25.282642847792765,
         27.2908518818265
      ],
   "BatchCount": 1,
    "MultiVariablePrimaryForceAlarm":["FORCE_HIGH", "FORCE_LOW"],

   "TimeStamp": 1371831729074,
   "Note": ""
},
```

Multi-Variable Data, with Extra or Missing data.

A special provision has been added to the data handling of the Multi-variable data. Regardless of the chart type, the array used to input the multi-variable data can be of any length, up to 25 elements. You can vary the number of samples per subgroup from sample interval to sample interval. You can enter more values than the specified number of samples per subgroup and you can enter less. A record without any data values is marked invalid and a hole will appear in the chart at that point. The example below uses a fixed sample size of 5. But the MultiVariableSampleValues arrays contain, sample subgroups with a number of elements ranging from 0 to 12. Internal to the software, the individual values of the MultiVariableSampleValues arrays are not stored. Instead, each array is reduced to the summary statistics (mean, range, sigma, etc.) needed for the plots in the one, two or three charts of the current SPC chart.

```
"SampleData": {
   "SampleIntervalRecords": [{
      "SampleValues": [
         27.53131515148628,
         33.95771604022404,
         24.310097827061817,
         28.282642847792765,
         30.2908518818265
      ],
       "MultiVariableSampleValues1":
         [],
       "MultiVariableSampleValues2":
      [
            22.53131515148628,
            30.95771604022404,
            21.310097827061817,
            22.282642847792765,
            26.2908518818265
      ],
      "BatchCount": 0,
      "TimeStamp": 1371830829074,
      "Note": ""
   }, {
      "SampleValues": [
         27.444285005240214,
         34.38930645615096,
         28.0203674441636,
         33.27153359969366,
         36.8305571558275
      ],
      "MultiVariableSampleValues1":
      [
         23.53131515148628,
          31.95771604022404,
         23.310097827061817,
         23.282642847792765,
         21.2908518818265
      ],
       "MultiVariableSampleValues2":
       [
         24.53131515148628,
         32.95771604022404,
         23.310097827061817,
         25.282642847792765,
         27.2908518818265
       ],
      "BatchCount": 1,
       "MultiVariablePrimaryForceAlarm":["FORCE_HIGH", "FORCE_LOW"],

      "TimeStamp": 1371831729074,
      "Note": ""
   }, {
      "SampleValues": [],
      "MultiVariableSampleValues1":
         [
            22.53131515148628,
            24.282642847792765,
            21.2908518818265
         ],
       "MultiVariableSampleValues2":
      [
         21.53131515148628,
         31.95771604022404,
         24.310097827061817,
         22.282642847792765,
         30.95771604022404,
         21.310097827061817,
         24.282642847792765,

         24.2908518818265
      ],
      "BatchCount": 2,
      "TimeStamp": 1371832629074,
      "Note": ""
   }, {
      "SampleValues": [
         27.898302097237174,
         25.906531082892915,
         26.950768095191137,
         30.812058501916457,
         31.085075984847936
      ],
      "MultiVariableSampleValues1":
         [
            23.53131515148628,
            32.95771604022404,
            25.310097827061817,
            24.282642847792765,
            26.2908518818265
         ],
       "MultiVariableSampleValues2":
      [
         22.53131515148628,
         34.95771604022404,
         25.310097827061817,
         29.282642847792765,
         28.2908518818265
      ],
       "MultiVariableSecondaryForceAlarm":["FORCE_HIGH", "FORCE_LOW"],
      "BatchCount": 3,
      "TimeStamp": 1371833529074,
      "Note": ""

   }, {
      "SampleValues": [
         22.94549873989527,
         30.867679835728896,
         27.79692256857592,
         32.57962619388354,
         24.601635858609224
      ],
      "MultiVariableSampleValues1":
      [
         26.53131515148628,
         34.95771604022404,
         25.310097827061817,
         29.282642847792765,
         23.2908518818265
      ],
       "MultiVariableSampleValues2": [ ],

      "BatchCount": 4,
      "TimeStamp": 1371834429074,
      "Note": ""
   },
```

The resulting chart looks like:



Note the holes in the chart at sample interval 0 (First of the multi-variable items), 2 (Primary Variable), and 4 (Second of the multivariable items), where empty records have been inserted with no data.

MultiVariablePrimaryForceAlarm: [SPC string constant: "NO_FORCING", SPC string constant: "NO_FORCING", ...]

MultiVariableSecondaryForceAlarm: [SPC string constant: "NO_FORCING", SPC string constant: "NO_FORCING", ...]

MultiVariableThirdForceAlarm: [SPC string constant: "NO_FORCING",   SPC string constant: "NO_FORCING", ...]

These three array properties force the alarm state (as indicated by the color, shaped and size of the symbol) to indicate a high alarm, low alarm, or normal state. The arrays are sized for number of added variables to your multi-variable chart. If you have a Multi-Variable chart (a MULTIVARIABLE_MEAN_RANGE_CHART for example), and you have two added variables, then the arrays would have two elements, for the first and second added variable. If you want to force the main variable in the chart, use the PrimaryForceAlarm (SecondaryForceAlarm or ThirdForceAlarm) properties described earlier. If left at the default value of "NO_FORCING", no forcing is done. The value values are: “NO_FORCING”, “FORCE_HIGH”, “FORCE_LOW”, and “FORCE_NORMAL”. The forcing values apply only to sample interval they are entered in. They do not carry forward into subsequent sample intervals.

The added variables are not checked against the established control limits, because they come from a different process and most likely have their own set of control limits. The alarm forcing of the added variables is the means by which the user can identify that one of the added variables is outside of its own established control limits. See the example scripts MultiVariableBatchXBarR, MultiVariableBatchXBarRCustomized, MultiVariableBatchXBarRMR, MultiVariableBatchXBarSigma, MultiVariableBatchIndividualRange, and MixedMultiVariableBatch in the chartdefEnhExample.js file.

```
{
             "SampleValues": [
                27.444285005240214,
                34.38930645615096,
                28.0203674441636,
                33.27153359969366,
                36.8305571558275
             ],
             "MultiVariableSampleValues1":
                [
                   23.53131515148628,
                   31.95771604022404,
                   23.310097827061817,
                   23.282642847792765,
                   21.2908518818265
                ],
              "MultiVariableSampleValues2":
                      [
                         24.53131515148628,
                         32.95771604022404,
                         23.310097827061817,
                         25.282642847792765,
                         27.2908518818265
             ],
             "BatchCount": 1,
             "MultiVariablePrimaryForceAlarm":["FORCE_HIGH", "FORCE_LOW"],
             "TimeStamp": 1371831729074,
             "Note": ""
          }, {
             "SampleValues": [
                35.21321620109259,
                32.93940741018088,
                33.66485557976163,
                34.17314124609133,
                24.576683179863725
             ],
             "MultiVariableSampleValues1":
                [
                   22.53131515148628,
                   30.95771604022404,
                   21.310097827061817,
                   24.282642847792765,
          21.2908518818265
       ],
    "MultiVariableSampleValues2":
             [
                16.53131515148628,
                31.95771604022404,
                24.310097827061817,
                22.282642847792765,
                24.2908518818265
   ],
   "BatchCount": 2,
   "TimeStamp": 1371832629074,
   "Note": ""
}, {
   "SampleValues": [
      27.898302097237174,
      25.906531082892915,
      26.950768095191137,
      30.812058501916457,
      31.085075984847936
   ],
   "MultiVariableSampleValues1":
      [
         23.53131515148628,
         32.95771604022404,
         25.310097827061817,
         24.282642847792765,
         26.2908518818265
```

```
      ],
    "MultiVariableSampleValues2":
             [
                22.53131515148628,
                                 34.95771604022404,
                                 25.310097827061817,
                                 29.282642847792765,
                                 28.2908518818265
                    ],
                    "MultiVariableSecondaryForceAlarm":["FORCE_HIGH", "FORCE_LOW"],
                    "BatchCount": 3,
                    "TimeStamp": 1371833529074,
                    "Note": ""

                },
```

DataSimulation

```
   DataSimulation
      StartCount: integer: 0
      Count: integer: 20
      Mean: double: 1
      Range: range: 1
```

The data simulation function is useful for testing your chart design, without the need to update the chart using SampleIntervalRecords. You can simulate data for all of the SPC charts using this method.

StartCount

Specifies the staring BatchCount of the simulation. Simulated data once for a chart, using a StartingCount of 0, and a Count of 100, you will end up with BatchCount values of 0 to 99. If you add new simulated data to the chart, you will need to set the StartCount property to 100, so as to not repeat the BatchCount numbers. The next 100 BatchCount values will then be 100 to 199, without repeating any BatchCount values.

Count

The number of sample intervals to simulate.

Mean

The mean of the sample interval data values.

Range

The range of the sample interval data values.

The DataSimulation method knows what chart type it is, how many samples per sample interval are needed, and all of the other things specific to the charts. It takes all of that into account when generating simulation data. You can see the simulated values in the table section of the chart.

```
"SampleData": {
   "DataSimulation": {
       "StartCount": 0,
       "Count": 50,
       "Mean": 27,
       "Range": 5
```

```
    }
},
```

ExcludeRecords

```
   ExcludeRecords: [integer, integer, ..]
```

It may be that bad values have found there way into the sample data. In that case you can exclude the data from being used in SPC Control limit calculations. Enter an array of integer values, specifying the indicies of the records to exclude.

```
"ExcludeRecords": [2, 7, 17, 31]
```

IncludeRecords

```
   IncludeRecords: [integer, integer, ..]
```

If you exclude records, they remain excluded. Should you decide you need to include them again, use IncludeRecords.

```
IncludeRecords: [2,  31]
```

ResetSPCChartData

```
ResetSPCChartData
```

Use this method to reset all of the SampleIntervalRecords to empty.

Dynamic Creation of JSON

Typically, the chart creation JSON script will be static, or nearly static. That means you can hand-code a template for a specific application. You can customize specific properties of the template using standard Javascript programming. For example, you start with the TimeXBarR example from the chartdefSimple.js file. All of the property values of the TimeXBarR record variable are filled-out with default values. But now you want to customize it a bit. You can use standard Javascript to do that, referencing the fields of the TimeXBarR record variable.

```
function defineChartUsingJSON( )
{
   TimeXBarR.SPCChart.TableSetup.ChartData.Title = "QC Mean Range Chart";
   TimeXBarR.SPCChart.TableSetup.ChartData.PartNumber = "122";
   TimeXBarR.SPCChart.TableSetup.ChartData.ChartNumber = "3";
   TimeXBarR.SPCChart.TableSetup.ChartData.PartName = "Widget X23";
   TimeXBarR.SPCChart.TableSetup.ChartData.Operation = "Flange Drilling";
   TimeXBarR.SPCChart.TableSetup.ChartData.Operator = "Mike Holtzman";

   var s = JSON.stringify(TimeXBarR);
  return s;
}
```

Data updates, using the SampleData.SampleIntervalRecords property involves appending new elements to an already existing array. Use the Javascript push function to do that. In the example below, all of the sample data is stored as an array of records within TimeXBarR.SPCChart.SampleData.SampleIntervalRecords. Each element of SampleIntervalRecords contains a structure which contains a SampleValues array, which is an array of values, one for each sample of a sample interval. So an array of SampleValues is created, and populated with sample data for a sample interval. That array is combined with BatchCount, TimeStamp, and Note data values in a SampleIntervalRecord, and that records is appended at the end of TimeXBarR.SPCChart.SampleData.SampleIntervalRecords using **push**.

```
function defineChartUsingJSON( )
   {
   var SampleIntervalRecord = {
                     "SampleValues": [],
                     "BatchCount": 0,
                     "TimeStamp": 1371830829074 + 20 * 900000,
                     "Note": "" };
   var SampleValues = new Array();

   SampleValues.push(27.53131515148628);
   SampleValues.push(33.95771604022404);
   SampleValues.push(24.310097827061817);
   SampleValues.push(28.282642847792765);
   SampleValues.push(30.2908518818265);

   SampleIntervalRecord.SampleValues = SampleValues;
   TimeXBarR.SPCChart.SampleData.SampleIntervalRecords.push(SampleIntervalRecord);


   var s = JSON.stringify(TimeXBarR);
   return s;
 }
```

9. Calculate and Update Methods

```
    Methods
       AutoCalculateControlLimits
       AutoScaleYAxes
       RebuildUsingCurrentData
       UpdateDisplay
       ResetControlLimitNofMCounts
       AutoDeleteControlLimits
```

The Methods block is a group of routines which should be called after you setup all of the other chart properties, and update the chart with the most recent data.

AutoCalculateControlLimits

```
AutoCalculateControlLimits [boolean, boolean]
```

This method will auto-calculate sigma based control limits, for primary and secondary charts, using formulas appropriate to each chart type. It uses all of the data currently added to the chart.

The format is pretty flexible. While it is shown as using a boolean array of two elements, it works with any of the following formats:

No parameters - Calculates control limits for primary chart, and if present, the secondary chart

```
AutoCalculateControlLimits
```

One, none-array parameter - Calculates control limits for primary, and if present, the secondary chart

```
AutoCalculateControlLimits: true
```

An array parameter of two boolean elements. The first element enables the calculation of control limits for the primary chart, and the second element enables the calculation of control limits for the secondary chart.

```
AutoCalculateControlLimits: [true, false]
```

Example

```
"Methods": {
           "AutoCalculateControlLimits": true,
           "AutoScaleYAxes": true,
           "RebuildUsingCurrentData": true
          }
```

AutoScaleYAxes

```
   AutoScaleYAxes [boolean, boolean]
```

This method will auto-scale the y-axes of the primary and secondary chart, taking into account all of the data added to the chart, all control limits, and all specification limits.

The format is pretty flexible. While it is shown as using a boolean array of two elements, it works with any of the following formats:

No parameters - Auto-scales the y-axis of the primary chart, and if present, the secondary chart)

```
AutoCalculateControlLimits
```

One, none-array parameter - Auto-scales the y-axis of the primary chart, and if present, the secondary chart)

```
AutoCalculateControlLimits: true
```

An array parameter of two boolean elements. The first element enables the auto-scale the y-axis for the primary chart, and the second element enables Auto-scales the y-axis of the primary chart for the secondary chart.

```
AutoCalculateControlLimits: [true, false]
```

RebuildUsingCurrentData

```
RebuildUsingCurrentData: boolean
```

If the chart has been changed in way, it needs to be rebuilt in order to for the changes to be visible on the screen. When you add new data chart using SampleData block (actual or simulated data), you also need to call RebuildUsingCurrentData for the new data so show up in the chart.

If a boolean parameter is not provided, it is considered true.

```
"RebuildUsingCurrentData"
```

is the same as

```
"RebuildUsingCurrentData": true
```

UpdateDisplay

```
UpdateDisplay
```

This routine forces a redraw of the chart. Normally it is not needed, since the RebuildUsingCurrentData method is what you use when you make changes to the chart, and then want them displayed. We include it here just in case it is needed for reasons we cannot predect.

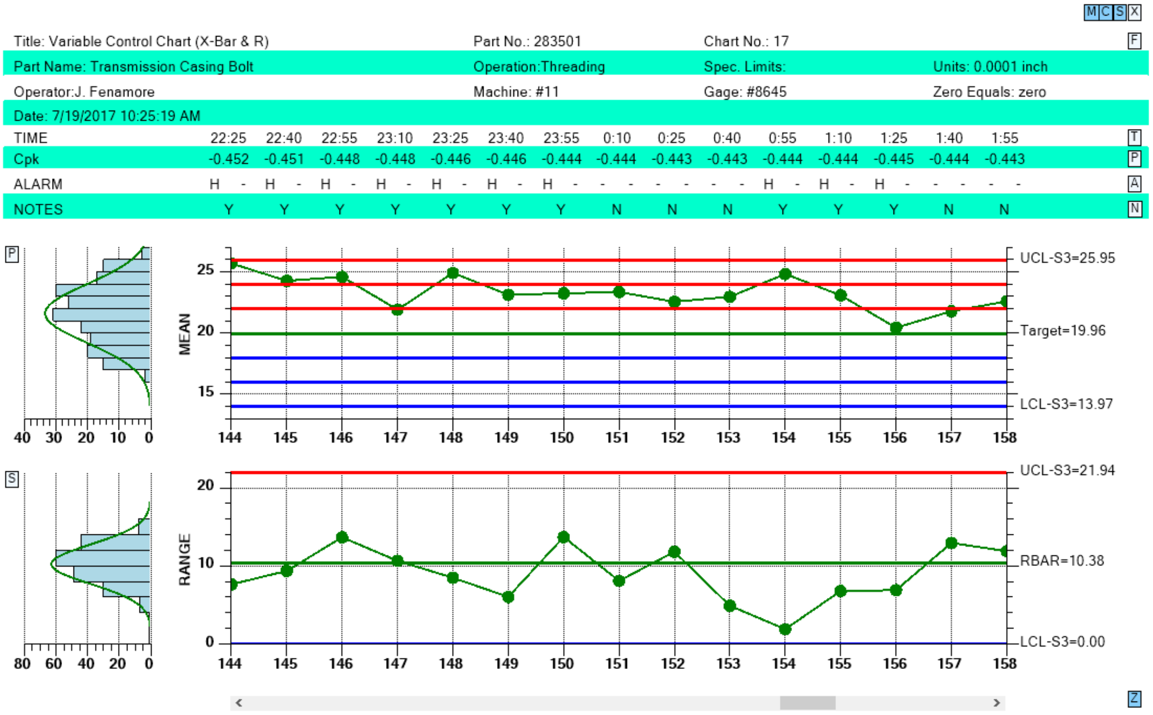
```
UpdateDisplay
```

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 method `ResetControlLimitNofMCounts` to reset the N of M counters back to 0.

```
  "Methods": {
           "ResetControlLimitNofMCounts": true

        },
```



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.

AutoDeleteControlLimits

```
AutoDeleteControlLimits
        LimitValue1
        LimitValue2
        LimitValue3
        ComparisonOperator
```

LimitValue1 Remove limits in the Primary Chart which are (ComparisonOperator) than the LimitValue1 value.

LimitValue2 Remove limits in the Secondary Chart which are (ComparisonOperator) than the LimitValue2 value.

LimitValue3 Remove limits in the Third Chart which are (ComparisonOperator) than the LimitValue3 value.

ComparitonOperator 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,

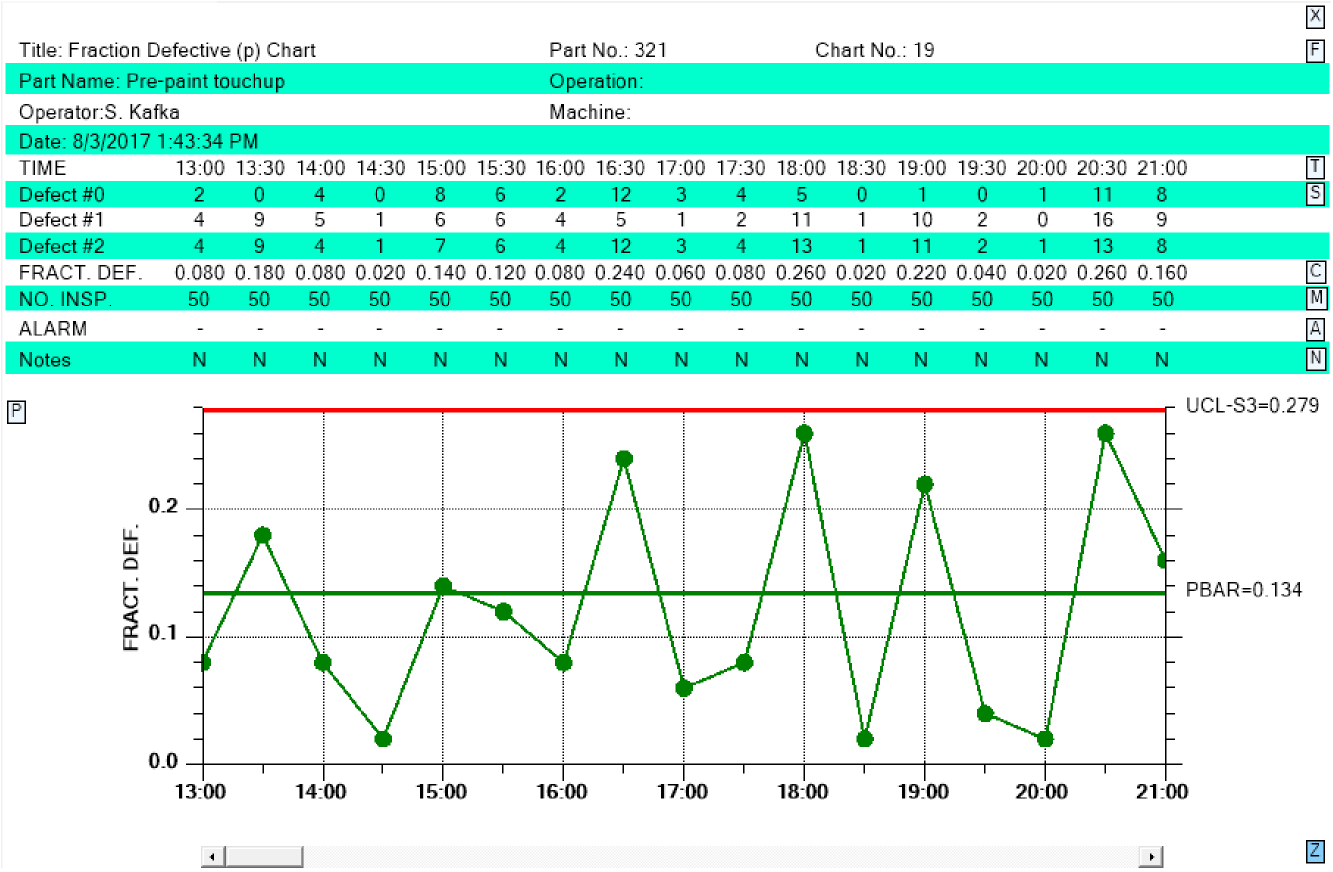
For a Variable Control chart, If you want to remove the LCL limit, equal to 0.0, from the Range (Secondary chart), call AutoDeleteControlLimits with the following parameters.

```
Methods: {
```

```
.
.
.

       "AutoDeleteControlLimits":  {
          "LimitValue2": 0.0,
            "ComparisonOperator":  "COMPARISON_OP_LESSTHAN_OR_EQ"
       }
}
```

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

```
Methods: {
.
.
.

       "AutoDeleteControlLimits":  {
          "LimitValue1": 0.0,
          "ComparisonOperator":  "COMPARISON_OP_LESSTHAN_OR_EQ"
       }
}
```



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.

10. Variable Control Charts

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 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.

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.

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.

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.

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 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.

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.

SPC Chart types found only the QCSPCChart+ version of the software.

Xbar-R-MR (Mean-Range-Moving Range Chart)

New single variable SPC chart (Xbar-R-MR), which has three sub plots: Xbar, Range, and MR (Moving Range derived from the Xbar chart). The added MR subplot has its own +-3 sigma set of control limits.

Multi-Variable SPC Charts (QCSPCChart+) for : IX-MR, Xbar-R, Xbar-Sigma, Xbar-R-MR, and a Mixed Chart.

Five new Multi-Variable SPC Charts for IX-MR, Xbar-R, Xbar-Sigma, Xbar-R-MR, and a Mixed Chart. The Mixed Chart can combine Xbar-R, Xbar-Sigma, and I-R charts in the same graph. Other Features of the Multi-Variable SPC Charts include:

- Dual y-scales and axes – a plot will be assigned to one or the other. The dual y-axis scale applies to all sub charts.
- Auto-scaling for all sub plots each with dual y-axis scales
- A legend will identify which plot is associated with which y-axis

Time-Based and Batch-Based SPC Charts

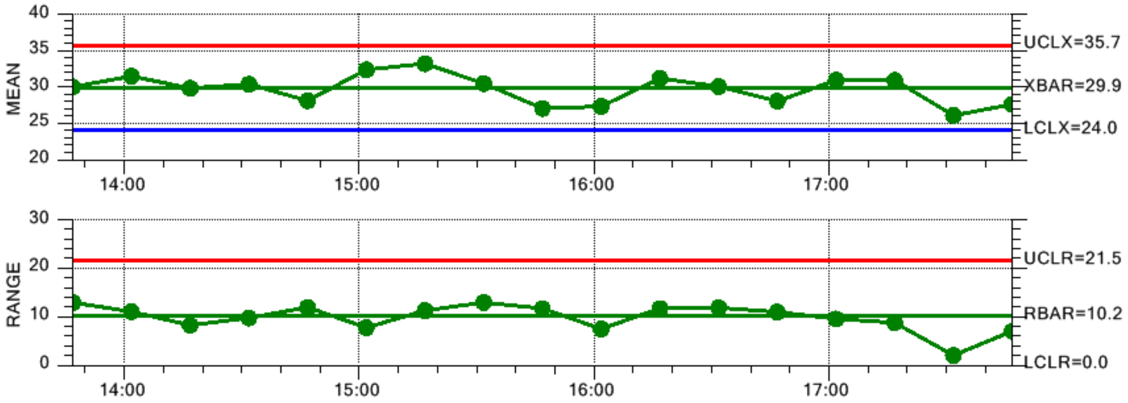
The **QCSPCChart** software further categorizes *Variable Control* as either time- or batch- based. Time-based SPC charts are used when data is collected using a subgroup interval corresponding to a specific time interval. Batch-based SPC charts are used when the data subgroup interval is a sequential 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 on the use of Time-based versus Batch-based Variable Control Charts

If you are using a **Time-based SPC Chart**, then you MUST specify time stamps which are monotonic and evenly spaced. Monotonic means that the values always increase. You can' t enter data from today, followed by data from yesterday. That is going backward in time and is non-monotonic. The Time-based control charts also require that the time stamps increase at a regular rate, i.e. 15 minutes. In time stamp units (milliseconds), this would be an increase of (15 * 60 * 1000 = 900000) milliseconds per sample interval. While this time stamp increment does not have to be exact, it should be close, or else the data plotted in the SPC chart will not line up with the table. You can't enter data from an 8-hour run yesterday, followed by an 8-hour run today. That would leave large gaps in the chart. If you have irregular time stamp data, you must use the Batch-based SPC Chart type, which ignores the time stamp when positioning data points in a chart. See the discussion of InitChartProperties in Chapter 5, SPC Initial Chart Setup.

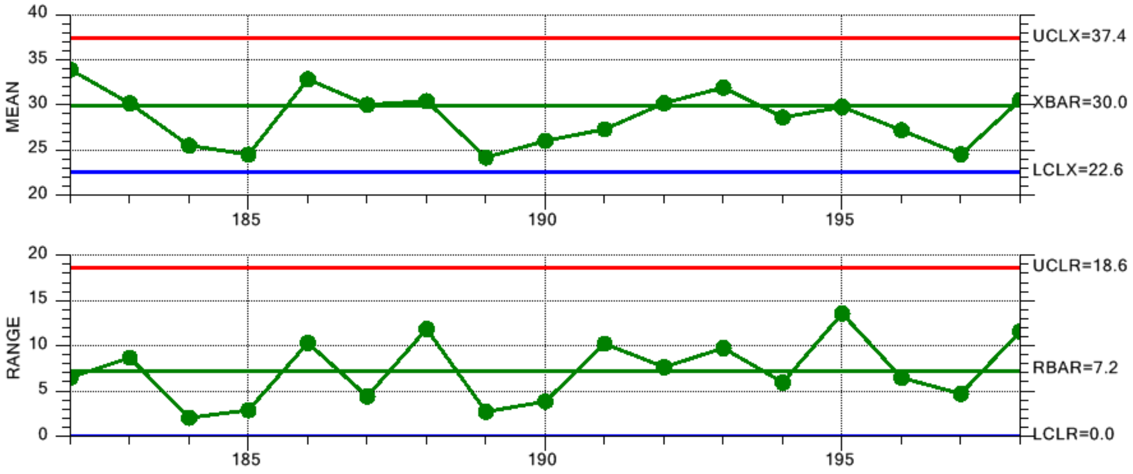
Most users do not have an even time interval between sample subgroups, which is a requirement for the Time-based charts. The batch control charts can label the x-axis using one of three options: numeric labeling (the original and default mode), time stamp labeling, and user defined string labeling. **Because of this change, we recommend that most users use the Batch-based SPC charts.**

Time-Based Variable Control Chart



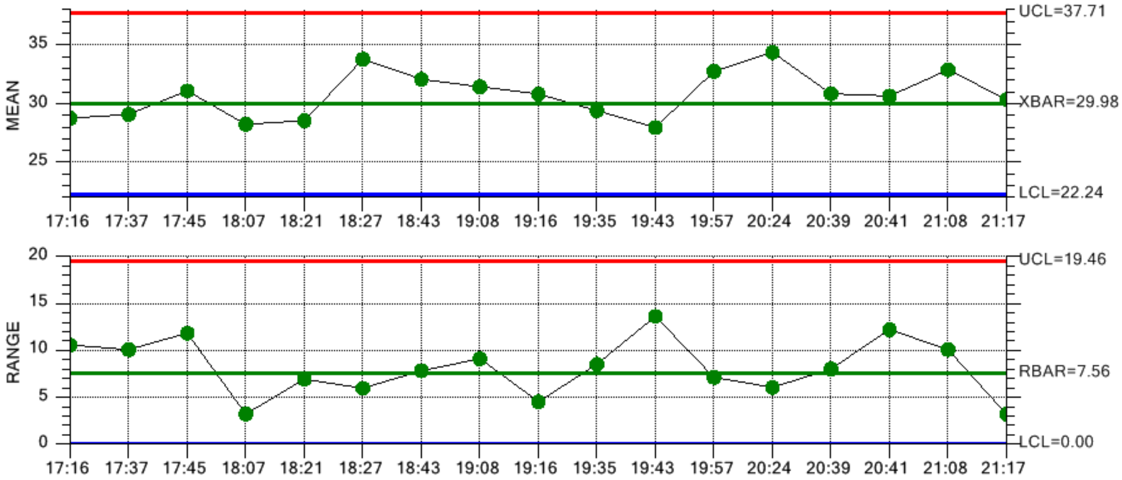
Note the time-based x-axis for both charts.

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

The chart type, and whether or not is is time-based or batch-based, is defined in the SPCChart: **InitChartProperties** block.

The InitChartProperties block has the following properties.

SPCChartType

The SPC chart type parameter. Use one of the string constants strings: MEAN_RANGE_CHART, MEDIAN_RANGE_CHART, MEAN_SIGMA_CHART, MEAN_SIGMA_CHART_VSS, INDIVIDUAL_RANGE_CHART, EWMA_CHART, MA_CHART, MAMR_CHART, MAMS_CHART, MEAN_RANGE_MR_CHART, LEVEY_JENNINGS_CHART and TABCUSUM_CHART. There are also constants for the multi-variable chart type: MULTIVARIABLE_MEAN_RANGE_CHART, MULTIVARIABLE_MEAN_SIGMA_CHART, MULTIVARIABLE_INDIVIDUAL_RANGE_CHART, MULTIVARIABLE_MEAN_RANGE_MR_CHART, and MULTIVARIABLE_MIXED_CHART.

ChartMode

Specifies if the x-axis is time-based (Time), or batch-base (Batch). Use the string constant string Time or Batch.

NumCategories

In an Attribute Control Charts this value represents the number of defect categories used to determine defect counts. Specify a numeric value, no quotes. Since the example above is for a Variable Control Chart (MEAN_RANGE_CHART), the NumCategories property does not need to be set.

NumSamplesPerSubgroup

Specifies the number of samples that make up a sample subgroup. If the SPCChartType is one of the variable sample size chart types, this value must be the maximum number of samples per subgroup. Specify a numeric value, no quotes.

NumDatapointsInView

Specifies the number of sample subgroups displayed in the graph at one time. Specify a numeric value, no quotes.

TimeIncrementMinutes

Specifies the approximate time increment (in minutes) between adjacent subgroup samples. This applies only to the Time ChartMode. Specify a numeric value, no quotes. Can be a double (0.5) to specify a fraction of a minute.

There are also three parameters which are used exclusively the CuSum chart type. You do not need to include them in any other chart.

CuSumKValue

A CuSum charts K value

CuSumHValue

A CuSum charts H value

CuSumMeanValue

A CuSum charts mean value

Example: Mean-Range (X-Bar R) with a Batch-based x-axis

```
"SPCChart": {
   "InitChartProperties": {
      "SPCChartType": "MEAN_RANGE_CHART",
       "ChartMode": "Batch",
       "NumSamplesPerSubgroup": 5,
       "NumDatapointsInView": 12
   },
```

The example above uses 5 samples per subgroup, and has a chart width of 12 points at a time.

Example: Individual-Range (IR) with a Time-based x-axis

```
"SPCChart": {
   "InitChartProperties": {
      "SPCChartType": "INDIVIDUAL_RANGE_CHART",
      "ChartMode": "Time",
      "NumSamplesPerSubgroup": 1,
      "NumDatapointsInView": 13,
      "TimeIncrementMinutes": 15
   },
```

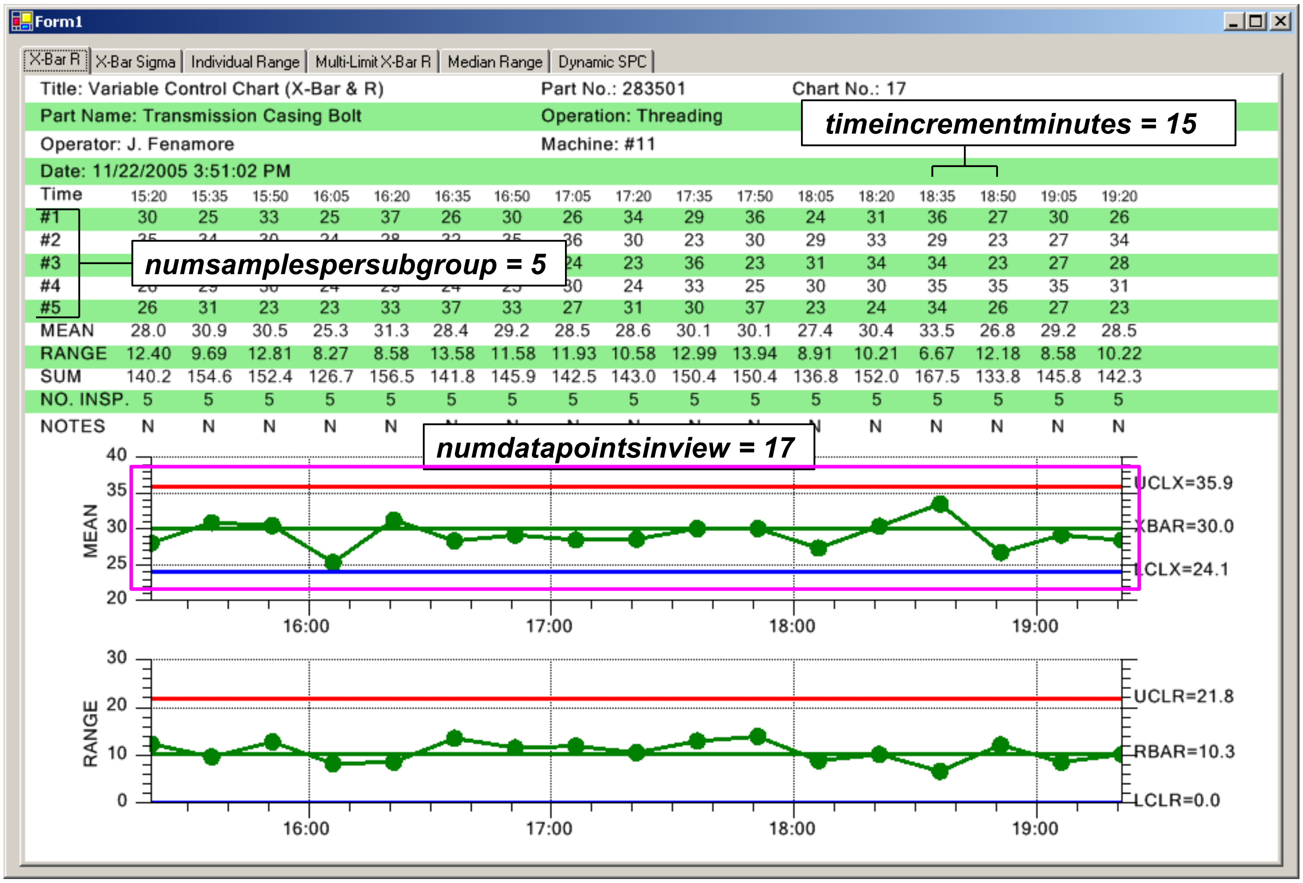
The example above uses 1 sample per subgroup, and has a chart width of 13 points at a time. Since it is time-based control chart, you need to specify a **TimeIncrementMinutes** parameter, 15 in this case.

Note that the X-Bar Sigma chart, with a variable subgroup sample size, is initialized using **InitChartProperties** with a SPCChartType value of MEAN_SIGMA_CHART_VSS.

```
"SPCChart": {
   "InitChartProperties": {
      "SPCChartType": "MEAN_SIGMA_CHART_VSS",
       "ChartMode": "Batch",
       "NumSamplesPerSubgroup": 15,
       "NumDatapointsInView": 12
   },
```

Initialize the **NumSamplesPerSubgroup** with the maximum number of sample per subgroup you expect for the subgroup data. 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.**

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: 11/22/2005 3:51:02 PM

Time	15:20	15:35	15:50	16:05	16:20	16:35	16:50	17:05	17:20	17:35	17:50	18:05	18:20	18:35	18:50	19:05	19:20
#1	30	25	33	25	37	26	30	26	34	29	36	24	31	36	27	30	26
#2	35	24	30	24	28	22	25	36	30	23	30	29	33	29	23	27	34
#3	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	24	23	36	23	31	34	34	23	27	28
#4	[illegible]	29	30	24	29	24	25	30	24	33	25	30	30	35	35	35	31
#5	26	31	23	23	33	37	33	27	31	30	37	23	24	34	26	27	23
MEAN	28.0	30.9	30.5	25.3	31.3	28.4	29.2	28.5	28.6	30.1	30.1	27.4	30.4	33.5	26.8	29.2	28.5
RANGE	12.40	9.69	12.81	8.27	8.58	13.58	11.58	11.93	10.58	12.99	13.94	8.91	10.21	6.67	12.18	8.58	10.22
SUM	140.2	154.6	152.4	126.7	156.5	141.8	145.9	142.5	143.0	150.4	150.4	136.8	152.0	167.5	133.8	145.8	142.3
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

timeincrementminutes = 15

numsamplespersubgroup = 5

numdatapointsinview = 17

Creating a Multi-Variable Control Chart (QCSPCChart+)

The chart type, and whether or not is is time-based or batch-based, is defined in the SPCChart: **InitChartProperties** block.

The InitChartProperties block has the following properties.

SPCChartType

The SPC chart type parameter. Use one of the multi-variable string constants strings: MULTIVARIABLE_MEAN_RANGE_CHART, MULTIVARIABLE_MEAN_SIGMA_CHART, MULTIVARIABLE_INDIVIDUAL_RANGE_CHART, MULTIVARIABLE_MEAN_RANGE_MR_CHART, and MULTIVARIABLE_MIXED_CHART.

ChartMode

Specifies if the x-axis is time-based (Time), or batch-base (Batch). Use the string constant string Time or Batch.

NumSamplesPerSubgroup

Specifies the number of samples that make up a sample subgroup. Specify a numeric value, no quotes.

NumDatapointsInView

Specifies the number of sample subgroups displayed in the graph at one time. Specify a numeric value, no quotes.

TimeIncrementMinutes

Specifies the approximate time increment (in minutes) between adjacent subgroup samples. This applies only to the Time ChartMode. Specify a numeric value, no quotes. Can be a double (0.5) to specify a fraction of a minute.

There are also three parameters which are used exclusively the multi-variable chart types. You do not need to include them if the chart is not of the multi-variable type.

NumberAddedVariables

All multi-variable chart types require this value. It specifies how many additional variables you are adding to the chart, in addition to the main variable. This control how many multi-variables are updated in the *SampleData* | *SampleIntervalRecords* block, and how many traces are plotted in the Primary, Secondary and Third chart windows. So if you have the main variable, and two added variables, the *NumberAddedVariables* should be set to 2. The total number of variables in the chart will be three (the main, plus two added variables).

MultiVariableMixedPrimaryChartType

Only the MULTIVARIABLE_MIXED_CHART multi-variable chart type require this value. It specifies the type of the main variable in the chart. This control how many multi-variables are updated in the SampleData | SampleIntervalRecords block, and how many traces are plotted in the Primary, Secondary and Third chart windows.

MultiVariableMixedChartSubTypes

The MULTIVARIABLE_MIXED_CHART does not actually specify the chart type used for the added multi-variable plots. Specify the added chart types using an array of strings, where the elements are the string constants: MEAN_RANGE_CHART, MEAN_SIGMA_CHART, or INDIVIDUAL_RANGE_CHART. You only specify the added chart types, the main chart type is specified using the MultiVariableMixedPrimaryChartType property.

Example: Multi-variable Mean-Range (X-Bar R) with a Batch-based x-axis

```
"InitChartProperties": {
   "SPCChartType": "MULTIVARIABLE_MEAN_RANGE_CHART",
      "ChartMode": "BATCH",
      "NumSamplesPerSubgroup": 5,
      "NumDatapointsInView": 18,
      "TimeIncrementMinutes": 15,
      "NumberAddedVariables": 2
   },
```

The example above uses 5 samples per subgroup, and has a chart width of 18 points at a time.

Example: Multi-variable Individual-Range (IR) with a Time-based x-axis

```
"SPCChart": {
   "InitChartProperties": {
      "SPCChartType": "MULTIVARIABLE_INDIVIDUAL_RANGE_CHART",
      "ChartMode": "TIME",
      "NumSamplesPerSubgroup": 1,
      "NumDatapointsInView": 12,
      "TimeIncrementMinutes": 15,
      "NumberAddedVariables": 2
   },
```

The example above uses 1 sample per subgroup, and has a chart width of 12 points at a time. Since it is time-based control chart, you need to specify a **TimeIncrementMinutes** parameter, 15 in this case.

Example: Multi-variable Mixed chart with a Batch-based x-axis

```
"SPCChart": {
   "InitChartProperties": {
      "SPCChartType": "MULTIVARIABLE_MIXED_CHART",
      "ChartMode": "BATCH",
      "NumSamplesPerSubgroup": 5,
      "NumDatapointsInView": 12,
      "TimeIncrementMinutes": 15,
      "NumberAddedVariables": 2,
      "MultiVariableMixedPrimaryChartType": "MEAN_RANGE_CHART",
      "MultiVariableMixedChartSubTypes": [ "MEAN_SIGMA_CHART",
                                          "INDIVIDUAL_RANGE_CHART"]
   },
```

The example specifies that the main variable will be plotted as a Mean-Range (Xbar-R) chart. The two added variables will be plotted as a Mean-Sigma (Xbar-Sigma) chart, and an Inividual-Range (I-R) chart.

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.

Below is an example of how to update a typical X-Bar R chart, setup for NumSamplesPerSubgroup = 5, using the SampleData block. You will find many more examples in the Chapter 8, Adding Data to an SPC Chart.

```
"SampleData": {
                    "SampleIntervalRecords": [
                    {
                        "SampleValues": [
                            27.53131515148628,
                            33.95771604022404,
                            24.310097827061817,
                            28.282642847792765,
                            30.2908518818265
                        ],
                        "BatchCount": 0,
                        "TimeStamp": 1371830829074,
                        "Note": ""
                    },
                    {
                        "SampleValues": [
                            27.444285005240214,
                            34.38930645615096,
                            28.0203674441636,
                            33.27153359969366,
                            36.8305571558275
                        ],
                        "BatchCount": 1,
                        "TimeStamp": 1371831729074,
                        "Note": ""
                    },
```

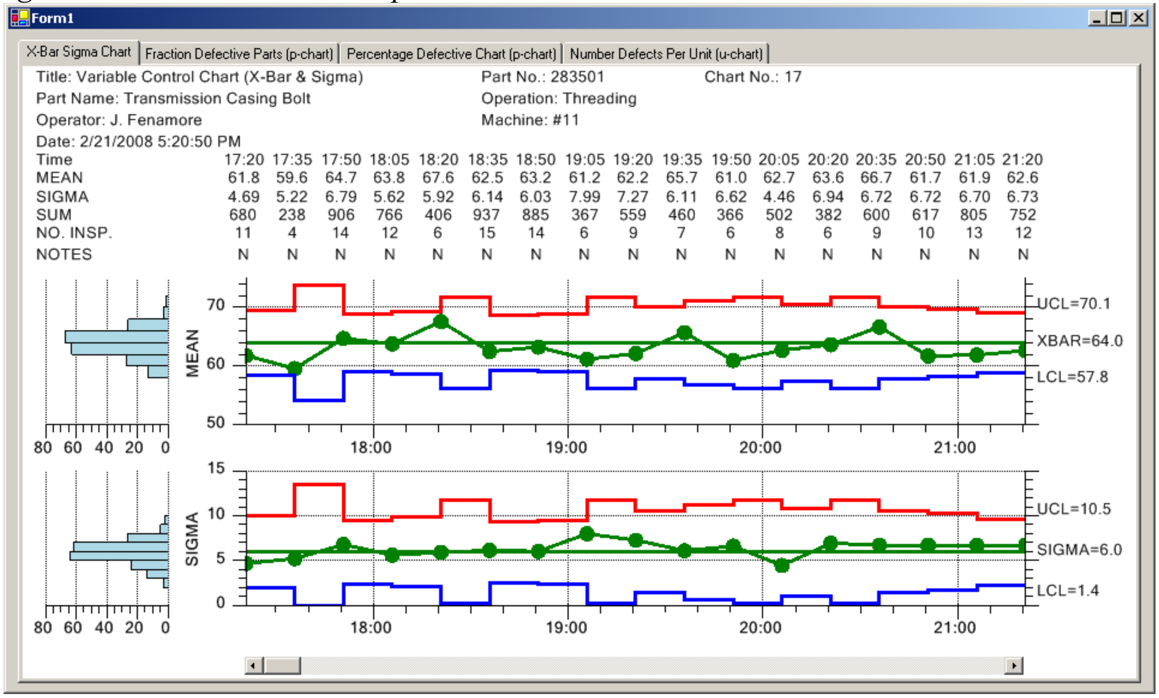
In an Individual-Range chart, and EWMA and MA charts that uses rational subgroup sizes of 1, the *samples* array would only have one value for each update. If the production level is sixty items per hour, with 100% sampling, the graph would be updated once a minute, with a single value in the *samples* array.

```
"SampleData": {
                    "SampleIntervalRecords": [
                    {
                        "SampleValues": [
                            27.53131515148628
                           ],
                        "BatchCount": 0,
                        "TimeStamp": 1371830829074,
                        "Note": ""
                    },
                    {
                        "SampleValues": [
                            27.444285005240214
                        ],
                        "BatchCount": 1,
                        "TimeStamp": 1371831729074,
                        "Note": ""
                    },
```

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.

X-Bar Sigma Chart with variable sample size



Title: Variable Control Chart (X-Bar & Sigma)
Part Name: Transmission Casing Bolt
Operator: J. Fenamore
Date: 2/21/2008 5:20:50 PM

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

Chart No.: 17

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.

```
"SampleData": {
       "SampleIntervalRecords": [
           {
               "BatchCount": 0,
               "TimeStamp": 1374768344076,
               "Note": "",
               "SampleValues": [
                   27.908233086630105,
                   31.940402816755082,
                   27.55563653827735,
                   30.083357069668647,
                   33.642341315640614,
                   28.72361222739654,
                   32.099133969171135,
                   26.356050567985285,
                   29.31049201044222,
                   28.499216431790145,
                   31.04435971419966,
                   33.2381471474555,
                   31.589757172384306,
                   32.438862725116614,
            31.53988206474448
        ]
    },
    {
        "BatchCount": 1,
        "TimeStamp": 1374769244076,
        "Note": "",
        "SampleValues": [
            28.749312830468913,
            26.404064179124912,
            33.28922196391206,
            31.694060174687134,
            26.90641611483808,
            31.48391922918815,
            30.209803672319776,
            29.86474359585655,
            32.53515676358969,
            29.12428709515284,
            29.75699541053711,
            31.017198158995903,
            30.224697293173516
        ]
    },
    {
        "BatchCount": 2,
        "TimeStamp": 1374770144076,
        "Note": "",
        "SampleValues": [
            26.801870534836045,
            27.378134477854075,
            33.08638714532048,
            31.769229875222915,
            32.0028978203766,
            27.226930046247567,
            28.687810832891774,
            33.60598037448174,
            29.262192232690143,
            32.27140254488991,
            32.70426215916774,
            32.02023454576835
        ]
    }
```

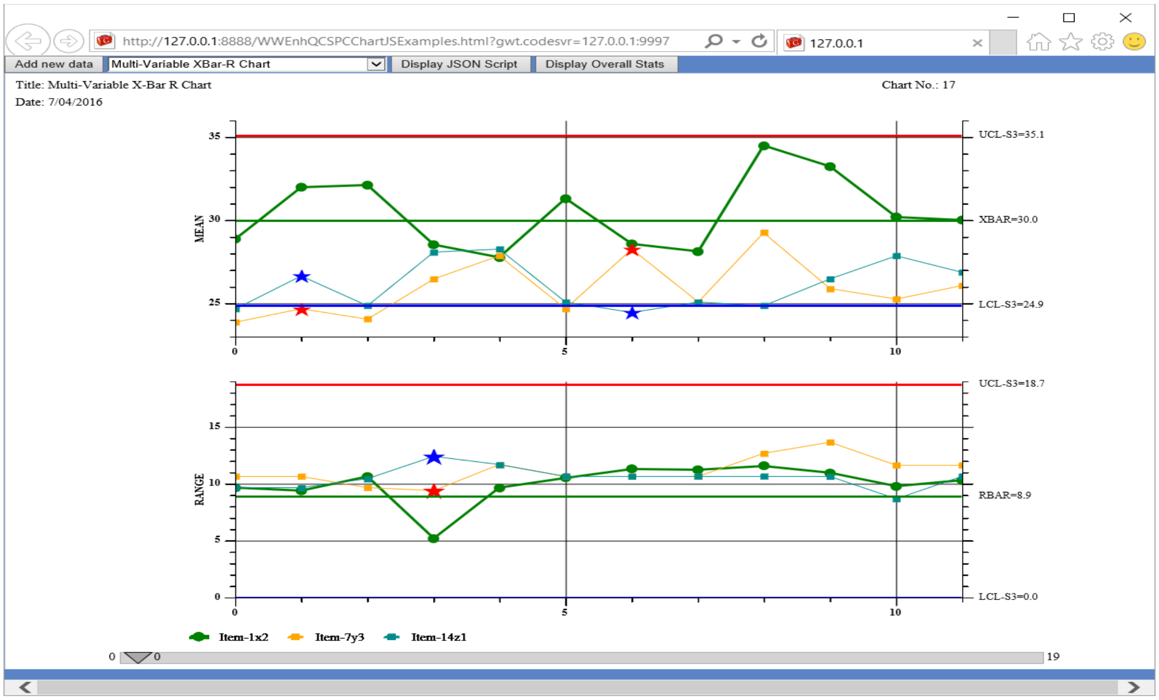
Updating MULTIVARIABLE_MEAN_RANGE_CHART (QCSPCChart+) with two added variables (extracted from the chartdefEhnExamples.js file, script MultiVariableBatchXBarR)

Assuming that your chart has been setup with the following parameters:

```
"InitChartProperties": {
   "SPCChartType": "MULTIVARIABLE_MEAN_RANGE_CHART",
   "ChartMode": "BATCH",
   "NumSamplesPerSubgroup": 5,
   "NumDatapointsInView": 18,
   "TimeIncrementMinutes": 15,
   "NumberAddedVariables": 2
},
```

Since there are two added variables, you, need to provide three data arrays for each sample interval update: SampleValues, MultiVariableSamples1, MultiVariableSamples2.

```
"SampleData": {
               "SampleIntervalRecords": [{
                  "SampleValues": [
                     27.53131515148628,
                     33.95771604022404,
                     24.310097827061817,
                     28.282642847792765,
                     30.2908518818265
                  ],
                   "MultiVariableSampleValues1":
                     [
                        21.53131515148628,
                        31.95771604022404,
                        22.310097827061817,
                        22.282642847792765,
                        21.2908518818265
                     ],
                   "MultiVariableSampleValues2":
                           [
                              22.53131515148628,
                              30.95771604022404,
                              21.310097827061817,
                              22.282642847792765,
                              26.2908518818265
                  ],
                  "BatchCount": 0,
                  "TimeStamp": 1371830829074,
                  "Note": ""
               }, {
                  "SampleValues": [
                     27.444285005240214,
                     34.38930645615096,
                     28.0203674441636,
                     33.27153359969366,
                     36.8305571558275
                  ],
                  "MultiVariableSampleValues1":
                     [
                        23.53131515148628,
                        31.95771604022404,
                        23.310097827061817,
                        23.282642847792765,
                        21.2908518818265
                     ],
                   "MultiVariableSampleValues2":
                           [
                        24.53131515148628,
                        32.95771604022404,
                        23.310097827061817,
                        25.282642847792765,
                        27.2908518818265
           ],
           "BatchCount": 1,
            "MultiVariablePrimaryForceAlarm":["FORCE_HIGH", "FORCE_LOW"],

           "TimeStamp": 1371831729074,
           "Note": ""
        },
```



Title: Multi-Variable X-Bar R Chart

Date: 7/04/2016

Chart No.: 17

Note that in the second sample interval update, the multi-variable plots have been forced into an alarm state (MultiVariablePrimaryForceAlarm).

Measured Data and Calculated Value Tables

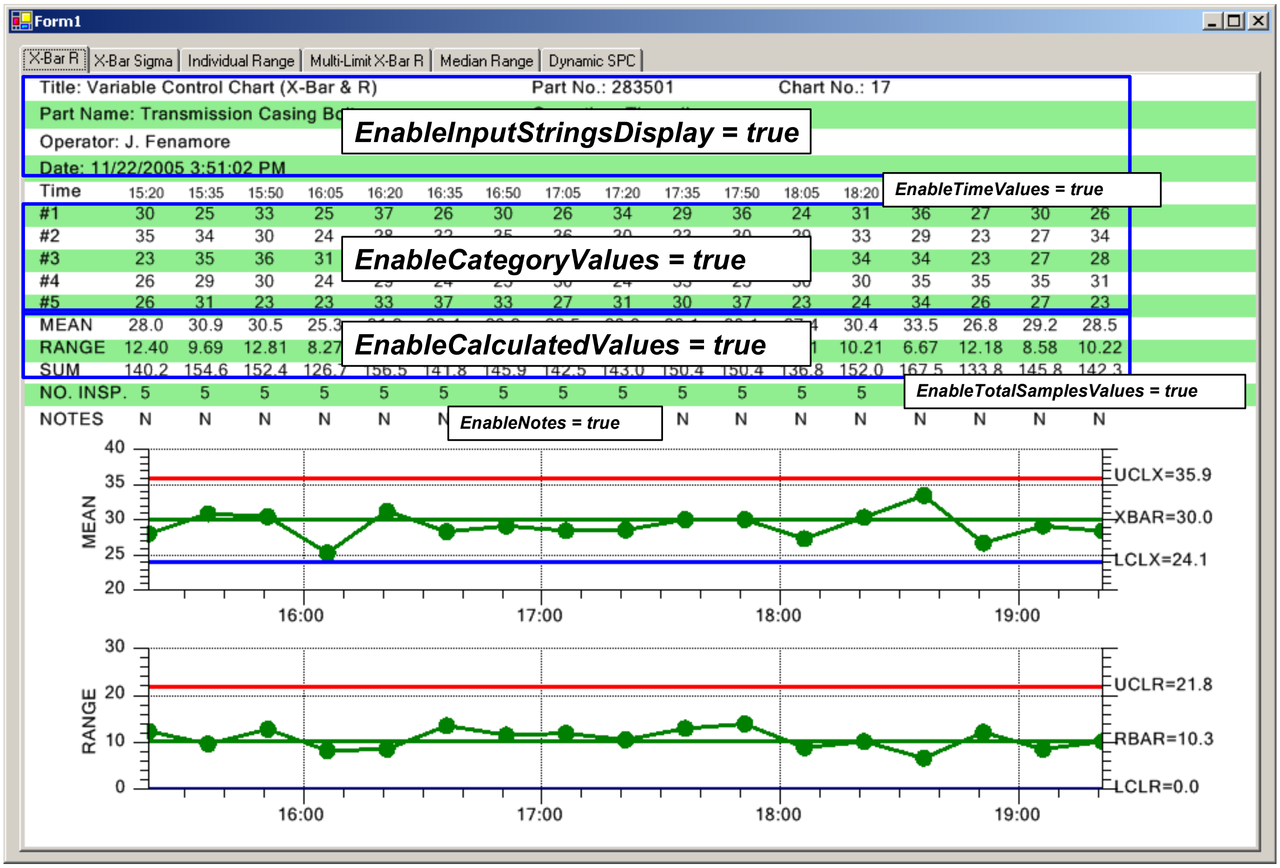
Standard worksheets used to gather and plot SPC data consist of three main parts.

- 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:
:

```
TableSetup
    EnableInputStringsDisplay: boolean: true
    EnableSampleValues: boolean: true
    EnableCalculatedValues: boolean: true
    EnableProcessCapabilityValues: boolean: true
    EnableTotalSamplesValues: boolean: true
    EnableNotes: boolean: true
    EnableTimeValues: boolean: true
```



Title: Variable Control Chart (X-Bar & R) Part No.: 283501 Chart No.: 17
Part Name: Transmission Casing Bo
Operator: J. Fenamore
Date: 11/22/2005 3:51:02 PM

Time	15:20	15:35	15:50	16:05	16:20	16:35	16:50	17:05	17:20	17:35	17:50	18:05	18:20	[illegible]	[illegible]	[illegible]	[illegible]
#1	30	25	33	25	37	26	30	26	34	29	36	24	31	36	27	30	26
#2	35	34	30	24	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	33	29	23	27	34
#3	23	35	36	31	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	34	34	23	27	28
#4	26	29	30	24	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	30	35	35	35	31
#5	26	31	23	23	33	37	33	27	31	30	37	23	24	34	26	27	23
MEAN	28.0	30.9	30.5	25.3	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	30.4	33.5	26.8	29.2	28.5
RANGE	12.40	9.69	12.81	8.27	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	10.21	6.67	12.18	8.58	10.22
SUM	140.2	154.6	152.4	126.7	156.5	141.8	145.9	142.5	143.0	150.4	150.4	136.8	152.0	167.5	133.8	145.8	142.3
NO. INSP.	5	5	5	5	5	5	5	5	5	5	5	5	5	[illegible]	[illegible]	[illegible]	[illegible]
NOTES	N	N	N	N	N	N	[illegible]	[illegible]	[illegible]	N	N	N	N	N	N	N	N

In the program the code looks like the following code extracted from the chartDefExampleScripts.js TimeXBarR example JSON script

```
"SPCChart": {
    "InitChartProperties": {
        "SPCChartType": "MEAN_RANGE_CHART",
        "ChartMode": "Time",
        "NumSamplesPerSubgroup": 5,
        "NumDatapointsInView": 12,
        "TimeIncrementMinutes": 15
    },
    "Scrollbar": {
        "EnableScrollBar": true,
        "ScrollbarPosition": "SCROLLBAR_POSITION_MAX"
    },

    "TableSetup": {
        "HeaderStringsLevel": "HEADER_STRINGS_LEVEL2",
        "EnableInputStringsDisplay": true,
        "EnableCategoryValues": true,
        "EnableCalculatedValues": true,
        "EnableTotalSamplesValues": true,
        "EnableNotes": true,
        "EnableTimeValues": true,
        "EnableNotesToolTip": true,
        "TableBackgroundMode":"TABLE_NO_COLOR_BACKGROUND",
        "TableAlarmEmphasisMode":"ALARM_HIGHLIGHT_BAR",
        "ChartAlarmEmphasisMode":"ALARM_HIGHLIGHT_SYMBOL",
        "ChartData": {
            "Title": "Variable Control Chart (X-Bar R)",
            "PartNumber": "283501",
            "ChartNumber": "17",
            "PartName": "Transmission Casing Bolt",
            "Operation": "Threading",
            "SpecificationLimits": "27.0 to 35.0",
            "Operator": "J. Fenamore",
            "Machine": "#11",
            "Gauge": "#8645",
            "UnitOfMeasure": "0.0001 inch",
            "ZeroEquals": "zero",
            "DateString": "7/04/2013",
            "NotesMessage": "Control limits prepared May 10",
            "NotesHeader": "NOTES"
        }
    },
```

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.ProcessCapabilitySetup.LSLValue** and

ChartData.ProcessCapabilitySetup.USLValue properties of the chart. The code below is from the chartDefExampleScripts.js TimeXBarR example JSON script.

```
"ProcessCapabilitySetup": {
   "LSLValue": 27,
   "USLValue": 35,
   "EnableCPK": true,
   "EnableCPM": true,
   "EnablePPK": true
}
```

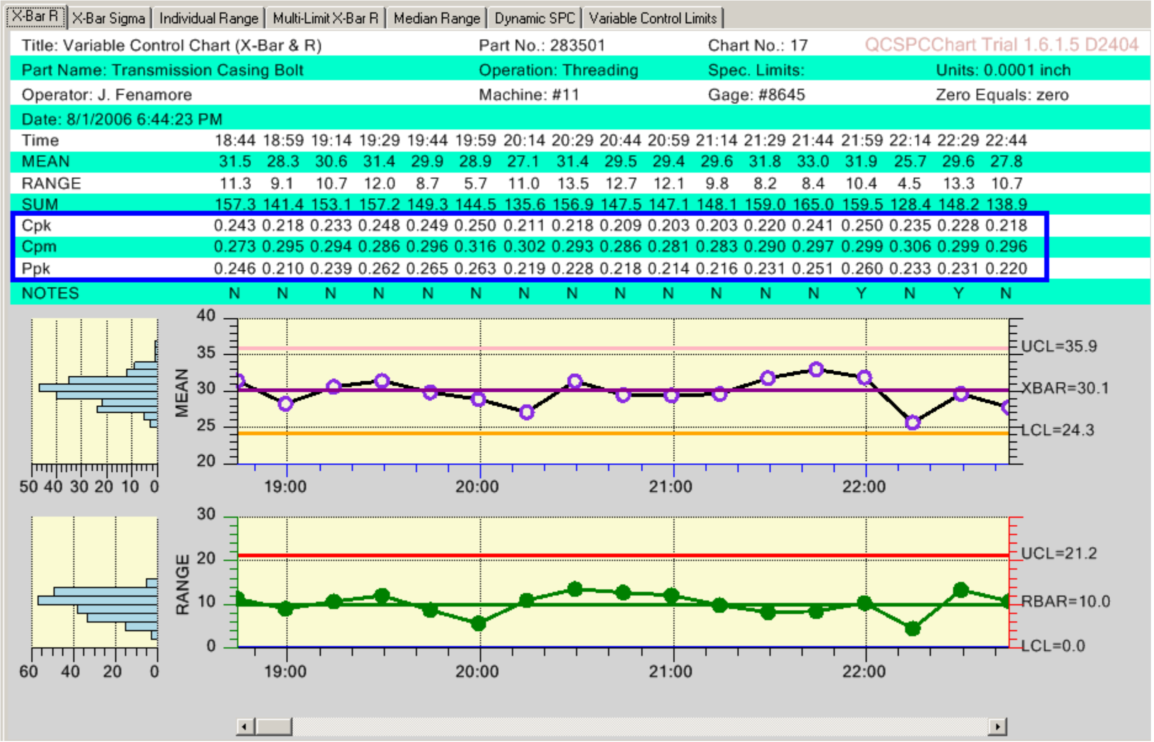
Enable the process capability items you want to include in the chart using one of the boolean properties from the list below.

```
EnableCPK: boolean: false
EnableCPM: boolean: false
EnablePPK: boolean: false
EnableCPL: boolean: false
EnableCPU: boolean: false
EnablePPL: boolean: false
EnablePPU: boolean: false
```

The code below is from the chartDefExampleScripts.js TimeXBarR example JSON script.

```
"ChartData": {
      "Title": "Variable Control Chart (X-Bar R)",
      "PartNumber": "283501",
      .
      .
      .
      "ProcessCapabilitySetup": {
         "LSLValue": 27,
         "USLValue": 35,
         "EnableCPK": true,
         "EnableCPM": true,
         "EnablePPK": true
      }

},
```

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. If you don't want to see the process capability statistics in the display, set the TableSetup.EnableProcessCapabilityValues property to false.



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.

_

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, Gauge, UnitOfMeasure, ZeroEquals and DateString. The four levels and the information displayed is listed below:

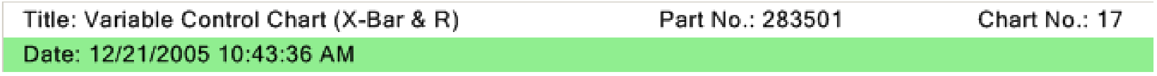
HEADER_STRINGS_LEVEL0 Display no header information

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, Gauge, UnitOfMeasure, ZeroEquals and DateString

The example JSON script chartDefExampleScripts.js TimeXBarR demonstrates the use of the **HeaderStringsLevel** property. The example below displays a minimum set of header strings (HeaderStringsLevel = HEADER_STRINGS_LEVEL1).

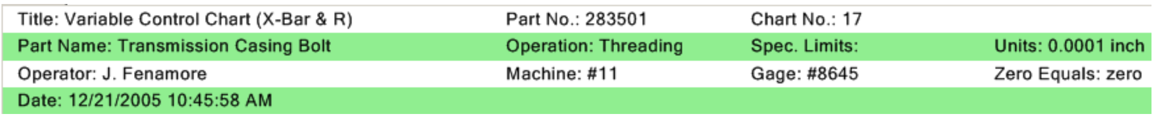


Title: Variable Control Chart (X-Bar & R) Part No.: 283501 Chart No.: 17

Date: 12/21/2005 10:43:36 AM

```
"TableSetup": {
    "HeaderStringsLevel": "HEADER_STRINGS_LEVEL1",
    "EnableInputStringsDisplay": true,
    "ChartData": {
        "Title": "Variable Control Chart (X-Bar R)",
        "PartNumber": "283501",
        "ChartNumber": "17",
```

The example below displays a maximum set of header strings (HeaderStringsLevel = 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			

```
"TableSetup": {
    "HeaderStringsLevel": "HEADER_STRINGS_LEVEL1",
    "EnableInputStringsDisplay": true,
    "ChartData": {
        "Title": "Variable Control Chart (X-Bar R)",
        "PartNumber": "283501",
        "ChartNumber": "17",
        "PartName": "Transmission Casing Bolt",
        "Operation": "Threading",
        "SpecificationLimits": "27.0 to 35.0",
        "Operator": "J. Fenamore",
        "Machine": "#11",
        "Gauge": "#8645",
        "UnitOfMeasure": "0.0001 inch",
        "ZeroEquals": "zero",
```

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:

```
"StaticProperties": {
   "SPCChartStrings": {
      "TitleHeader": "Project Name:",
      "DefaultMean": "Average",
      "TimeValueRowHeader": "Time"
   }
}
```

Change other header strings using the **ChartData** properties listed below.

- TitleHeader
- PartNumberHeader
- ChartNumberHeader
- PartNameHeader
- OperationHeader

- OperatorHeader
- MachineHeader
- DateHeader
- SpecificationLimitsHeader
- GaugeHeader
- 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.

Special Note

Rather than change the header string using

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.

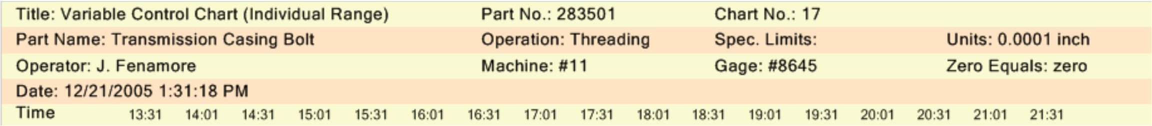
TABLE_NO_COLOR_BACKGROUND Constant specifies that the table does not use a background color.

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.

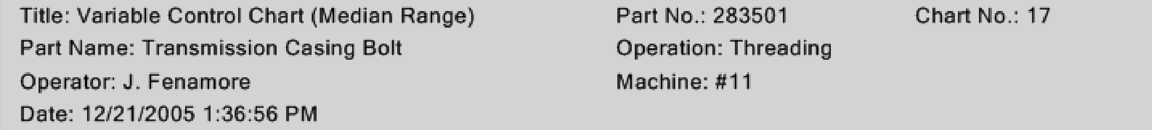
Extracted from the chartDefExampleScripts.js TimeIR example JSON script



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

```
"TableSetup": {
                    .
                    .
                    .
                  "TableBackgroundMode": "TABLE_STRIPED_COLOR_BACKGROUND",
                  "BackgroundColor1": "BEIGE",
                  "BackgroundColor2": "LIGHTGOLDENRODYELLOW",
```

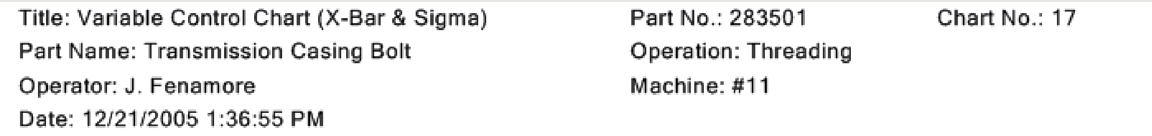
Extracted from the chartDefExampleScripts.js BatchMedianRange example JSON script



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

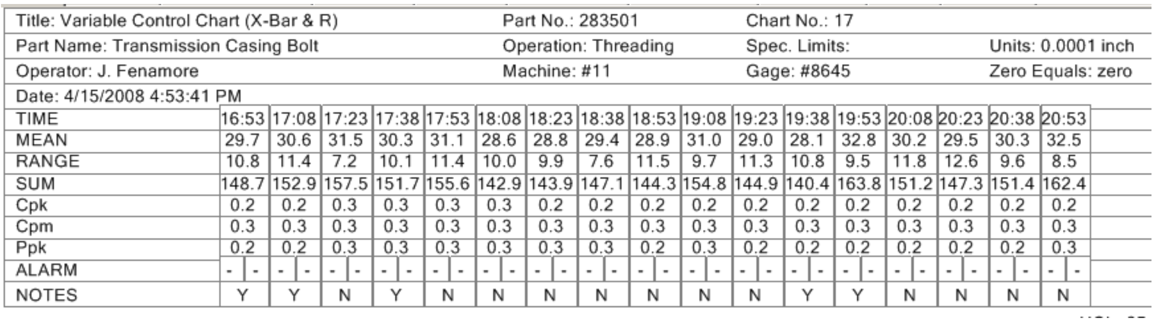
```
"TableSetup": {
          .
          .
          .
     "TableBackgroundMode": "TABLE_SINGLE_COLOR_BACKGROUND",
     "BackgroundColor1": "LIGHTGRAY",
```

Extracted from the chartDefExampleScripts.js TimeXBarSigma JSON script.



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

```
"TableSetup": {
          .
          .
          .
      "TableBackgroundMode": "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

```
"TableSetup": {
          .
          .
          .
     "TableBackgroundMode": "TABLE_SINGLE_COLOR_BACKGROUND_GRIDCELL",
     "BackgroundColor1": "WHITE",
     "BackgroundColor2": "GRAY",
```

Table and Chart Fonts

The StaticProperties block has static property which is used to set the default table font. Use this if you want to override the default font-family used for both tables and charts, established using the DefaultFontName property. Setting the static properties needs to be done first thing in the first JSON chart definition file you process.

Extracted from the chartDefExampleScripts.js BatchIR example JSON script.

```
"StaticProperties":
{
   "DefaultFontName":  "Arial, sans-serif",
   "DefaultTableFont":
   {  "Name": "'Comic Sans MS', cursive, sans-serif",
      "Size": 12,
      "Style": "Plain"
   },
},
```

In the example above, the default font is set to Arial. Normally this would apply to both charts and tables. However, the default table font is over-ridden to Comic Sans MS. So now the charts are in Arial, and the table are is in Comic Sans MS.

Chart Fonts

The default font family is set using the static DefaultFontName property used in the example above. The font sizes though vary from chart object to object in a chart. It is possible to set the font for a class of objects using the StaticProperties block. object by object bases. They establish the default fonts for the chart objects of a given type.

AxisLabelFont	The font used to label the x- and y- axes.
AxisTitleFont	The font used for the axes titles.
MainTitleFont	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

```
StaticProperties
   DefaultChartFonts
      AxisLabelFont
         Name: String: "sans-serif"
         Size: double: 12
         Style: String: "BOLD"
      AxisTitleFont: standard Name, Size:12, Style: BOLD font properties
      MainTitleFont: standard Name, Size:18, Style: BOLD font properties
      SubHeadFont: standard Name, Size:14, Style: BOLD font properties
      ToolTipFont: standard Name, Size:12, Style: PLAIN font properties
      AnnotationFont: standard Name, Size:12, Style: PLAIN font properties
      ControlLimitLabelFont: standard Name, Size:12, Style: PLAIN font properties
```

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 **InitChartProperties** routine.

```
StaticProperties: {
      "DefaultChartFonts": {
          "AxisLabelFont": {
              "Name": "sans-serif",
              "Size": 16,
              "Style": "PLAIN"
           },
           "AxisTitleFont": {
               "Name": "sans-serif",
               "Size": 16,
               "Style": "BOLD"
           }
      }
 }
```

Setting Decimal Precision in the Table

Properties under the ChartData block of the TableSetup block will set the decimal precision of the sample values, the calculated values and the process capability values of the table.

```
"SPCChart": {
   "TableSetup": {
       "HeaderStringsLevel": "HEADER_STRINGS_LEVEL2",

             .
             .
             .
       "ChartData": {
       "Title": "Variable Control Chart (X-Bar R)",
             .
             .
             .
        "CalculatedItemDecimals": 3,
        "ProcessCapabilityDecimals": 1,
        "SampleItemDecimals": 0
   }
}
```

SPC Charts without a Table

If you don’t want any of the items we have designated table itmes, use the **UseNoTable** block. It 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.

```
UseNoTable
   PrimaryChart: boolean: true
   SecondaryChart: boolean: true
   Histograms: boolean: true
   Title: String: ""
```

PrimaryChart

Set to true to display the primary chart.

SecondaryChart

Set to true to display the secondary chart.

Histograms

Set to true to display the histograms to the left of each chartSpecify a string title to display above the graphs.

In the odd case that there is a third chart (Xbar-R-MR), the third chart tracks the value of the SecondaryChart property.

Important Note: When using UseNoTable, do NOT use a TableSetup block in your JSON script.

Example

```
"UseNoTable": {
   "PrimaryChart": true,
   "SecondaryChart": true,
   "Histograms": true,
   "Title": "Place your chart title here"
},
```

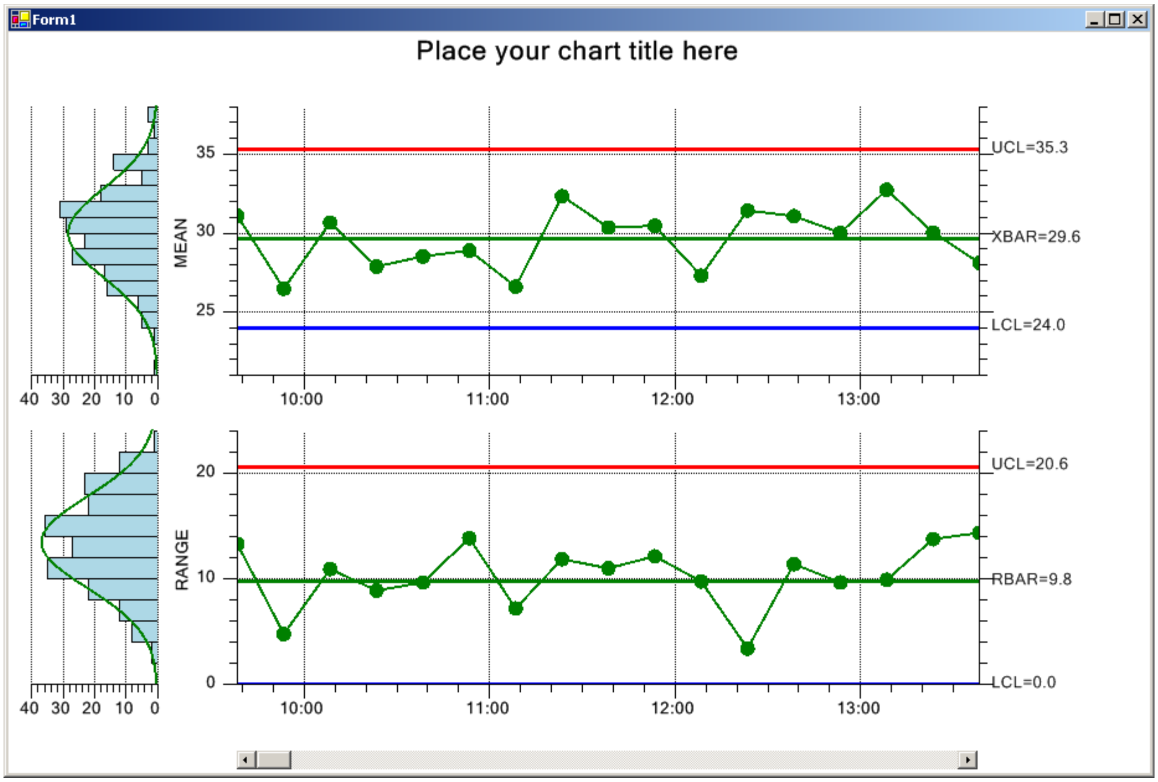


Chart Position

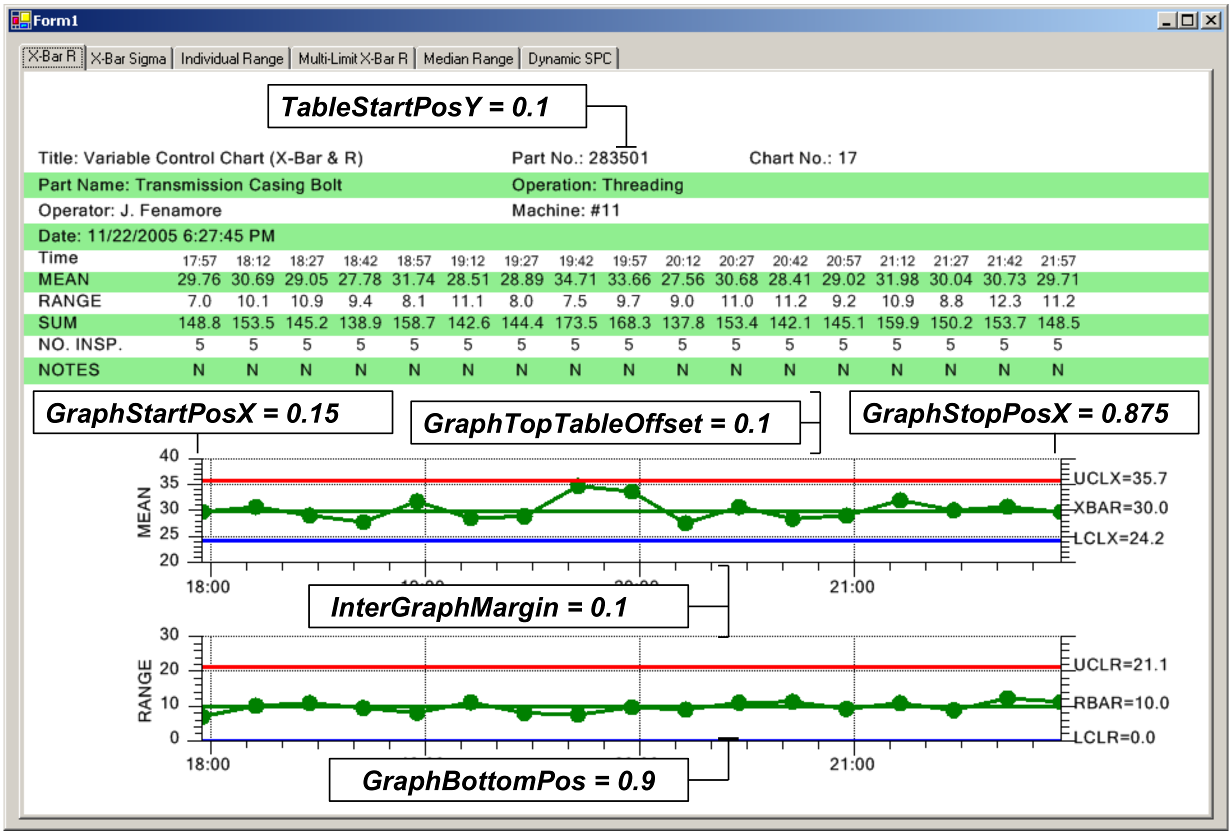
If the SPC chart does not include frequency histograms on the left (they take up about 20% of the available chart width), you may want to adjust the left and right edges of the chart using the **GraphStartPosX** and **GraphStopPlotX** properties to allow for more room in the display of the data. This also affects the table layout, because the table columns must line up with the chart data points.

```
"ChartPositioning": {
       "GraphStartPosX": 0.1,
       "GraphStopPosX": 0.875
},
```

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:

```
"ChartPositioning": {
   "GraphStartPosX": 0.15,
   "GraphStopPosX": 0.8,
   "TableStartPosY": 0.0,
   "GraphTopTableOffset": 0.02,
   "InterGraphMargin": 0.075,
   "GraphBottomPos": 0.90,
   "BottomLabelMargin": 0.0
}
```

The picture below uses different values for these properties in order to emphasize the affect that these properties have on the resulting chart.



TableStartPosY = 0.1

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: 11/22/2005 6:27:45 PM

Time	17:57	18:12	18:27	18:42	18:57	19:12	19:27	19:42	19:57	20:12	20:27	20:42	20:57	21:12	21:27	21:42	21:57
MEAN	29.76	30.69	29.05	27.78	31.74	28.51	28.89	34.71	33.66	27.56	30.68	28.41	29.02	31.98	30.04	30.73	29.71
RANGE	7.0	10.1	10.9	9.4	8.1	11.1	8.0	7.5	9.7	9.0	11.0	11.2	9.2	10.9	8.8	12.3	11.2
SUM	148.8	153.5	145.2	138.9	158.7	142.6	144.4	173.5	168.3	137.8	153.4	142.1	145.1	159.9	150.2	153.7	148.5
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

GraphStartPosX = 0.15

GraphTopTableOffset = 0.1

GraphStopPosX = 0.875

InterGraphMargin = 0.1

GraphBottomPos = 0.9

SPC Control Limits

There are several 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. Another d auto-calculates the limits using the algorithms supplied in this software. If you want to set the Target, LCL3 (-3-sigma limit) and UCL3 (3-sigma limit), to explicit values, you can do in the Target, LCL3 and UCL3 blocks of the PrimaryChartSetup | ControlLimits block. Assign the Value property to the limit value you want.

```
"ControlLimits": {
   "Target": {
      "DisplayString": "TargetXX",
      "EnableAlarmLine": true,
      "EnableAlarmChecking": true,
      "LimitValue": 30,
      "EnableAlarmLineText": true
   },
   "LCL3": {
      "DisplayString": "LCLXX",
      "EnableAlarmLine": true,
      "EnableAlarmChecking": true,
      "LimitValue": 25,
      "EnableAlarmLineText": false
   },
   "UCL3": {
      "DisplayString": "UCLXX",
      "EnableAlarmLine": false,
      "EnableAlarmChecking": true,
      "LimitValue": 35,
      "EnableAlarmLineText": true
    }
  }
```

If you have more than the standard +- Sigma control limits, it is better if you use the SpecifyControlLimitsUsingMeanAndSigma block to set all the limits.

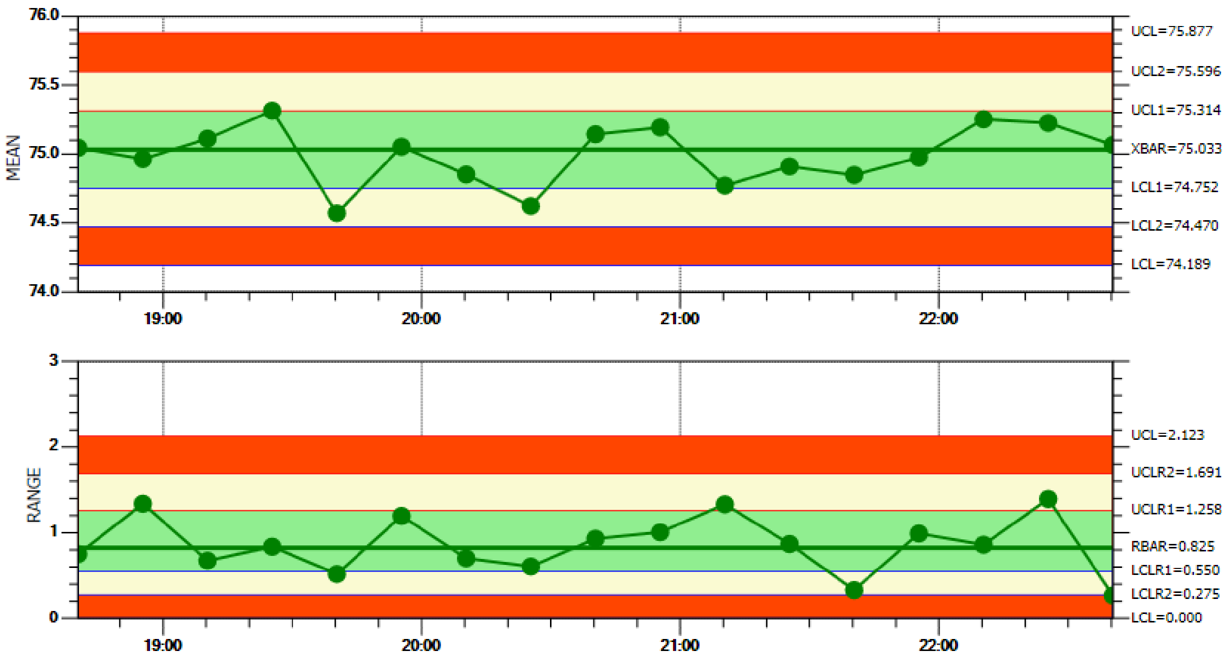
```
SpecifyControlLimitsUsingMeanAndSigma
   Mean: double: 1
   Sigma: double: 1
```

In the example above, where the Target was 30, the LCL3 value 25, and the UCL value 25, the mean an sigma values would be: Mean = Target = 30 and Sigma = (ULC3 – Mean) / 3 = 1.6666.

```
"SpecifyControlLimitsUsingMeanAndSigma": {
   "Mean":  30,
   "Sigma": 1.666
}
```

There is another property block (**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 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.

```
"123SigmaControlLimits": {
      "Target":  30,
      "LCL3Value": 25,
      "UCL3Value": 30,
      "AlarmTest12": true ,
     "EnableAlarmLine": true,
      "EnableAlarmChecking":  false,
      "EnableAlarmLineText":  true
}
```



Control Limit Fill Option used with +-1, 2 and 3-sigma control limits

The second way to set the control limits is to 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.

AutoCalculateControlLimits takes a boolean array parameter: one for the Primary Chart and one for the Secondary Chart. If you leave out the array parameter, it is the same as the values [true, true]

```
"Methods": {
      "AutoCalculateControlLimits": [true,true],
      "AutoScaleYAxes": [true,true],
      "RebuildUsingCurrentData": true
 }
```

Almost always, a call to AutoCalculateControlLimits will be followed by a call to AutoScaleYAxes to rescale the chart to take into account the new control limits, and RebuildUsingCurrentData to rebuild the graph to show the new limits.

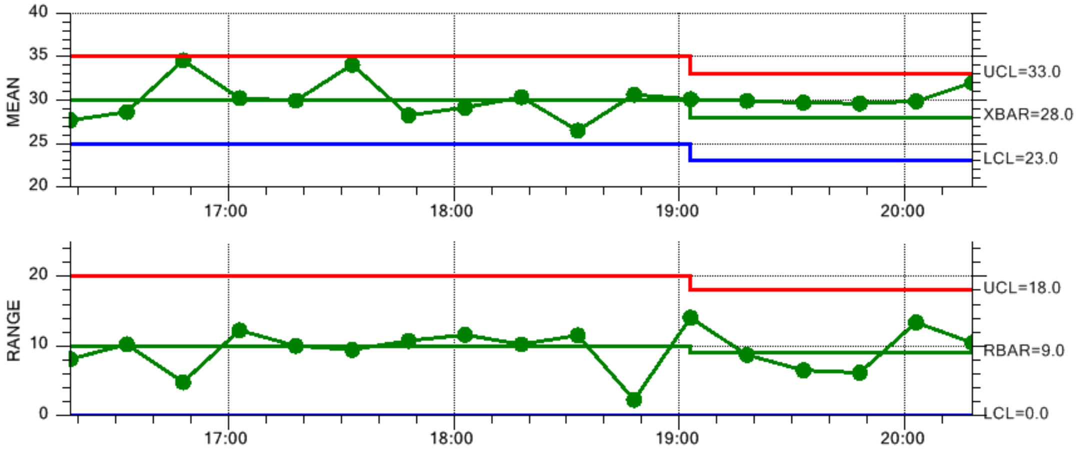
You can add data to the **ChartData** object, auto-calculate the control limits to establish the SPC control limits, and continue to add new data values. Alternatively, you can set the SPC control limits explicitly, as the result of previous runs, using the Target, LCL3, UCL3, 123SigmaControlLimits, or SpecifyControlLimitsUsingMeanAndSigma properties.

Need to exclude records from the control limit calculation? Mark which ones to exclude using the SampleData | ExcludeRecords properties.

```
"ExcludeRecords": [2, 7, 17, 31]
```

Variable SPC Control Limits

There can be situations where SPC control limits change in a chart.



There are four ways to enter new SPC limit values. First, you can use the **PrimaryChartSetup.ControlLimits.SetLimits** array property**.**

```
"PrimaryChartSetup": {
   "ControlLimits": {
          "SetLimits": [28, 23, 33]
   }
}
```

and the **SecondaryChartSetup.ControlLimits.SetLimits** array property**.**

```
"SecondaryChartSetup": {
   "ControlLimits": {
          "SetLimits": [9, 0, 18]
   }
}
```

This method only works if you are working with the default +-3 Sigma control limits (+ targets) for the Primary and Secondary charts.

Second, you can use the **Methods.AutoCalculateControlLimits** method. You must have already added a collection of sampled data values to the chart before you can call this method, since the method needs historical values needed in the calculation.

```
"Methods": {
      "AutoCalculateControlLimits": [true,true]

 }
```

This method works to set the control limits for all sigma-based limits..

Third, you can use the **PrimaryChartSetup.SpecifyControlLimitsUsingMeanAndSigma** property and just set the mean and sigma of the process.

```
"PrimaryChartSetup": {
   "SpecifyControlLimitsUsingMeanAndSigma": {
      "Mean": 28,
      "Sigma": 5
   }
}
```

Also, the **SecondaryChartSetup.SpecifyControlLimitsUsingMeanAndSigma** property.

```
"SecondaryChartSetup": {
   "SpecifyControlLimitsUsingMeanAndSigma": {
      "Mean": 5,
      "Sigma": 2
   }
}
```

For the SecondaryChartSetup, you must call it with the mean (centerline value) and sigma of that chart, not the primary chart. They are different.

This method works to set the control limits for all sigma-based limits..

Last, you can enter the SPC control limits with every new sample subgroup record, using SampleData.SampleIntervalRecords.VariableControlLimits array parameter.

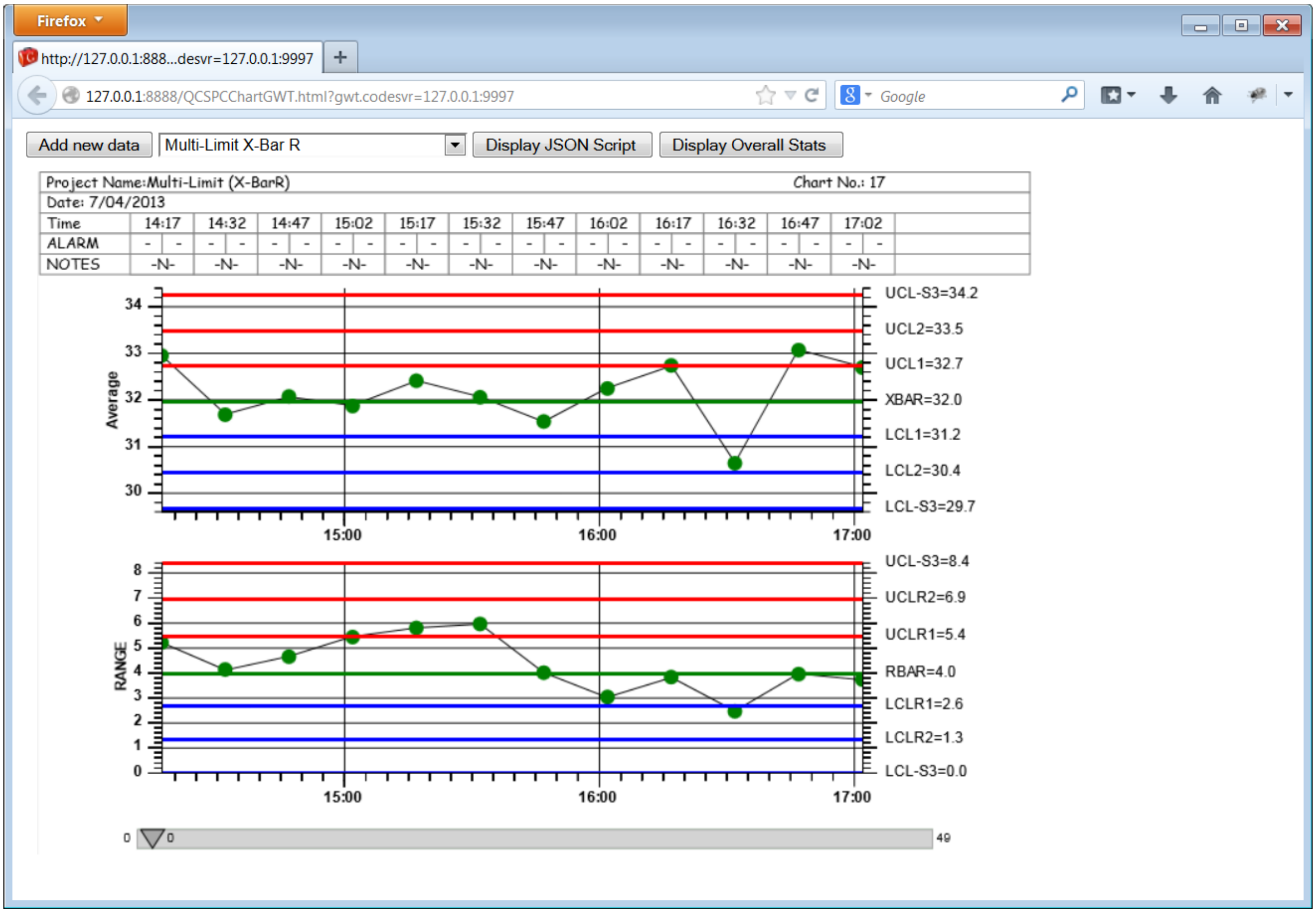
```
"SampleData": {
        "SampleIntervalRecords": [
        {
            "SampleValues": [
                27.53131515148628,
                33.95771604022404,
                24.310097827061817,
                28.282642847792765,
                30.2908518818265
            ],
             "VariableControlLimits": [28, 23, 33, 9, 0, 18],
            "BatchCount": 0,
            "TimeStamp": 1371830829074,
            "Note": ""
        },
```

The order of the values in the VariableControlLimits array is [Primary target, Primary LCL3, Primary UCL3, Secondary target, Secondary LCL3, Secondary UCL3]. Other limits are ignored.
This method only works if you are working with the default +-3 Sigma control limits (+ targets) for the Primary and Secondary charts.

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 JSON script chartDefExampleScripts.js TimeMultiLimitXBarR.

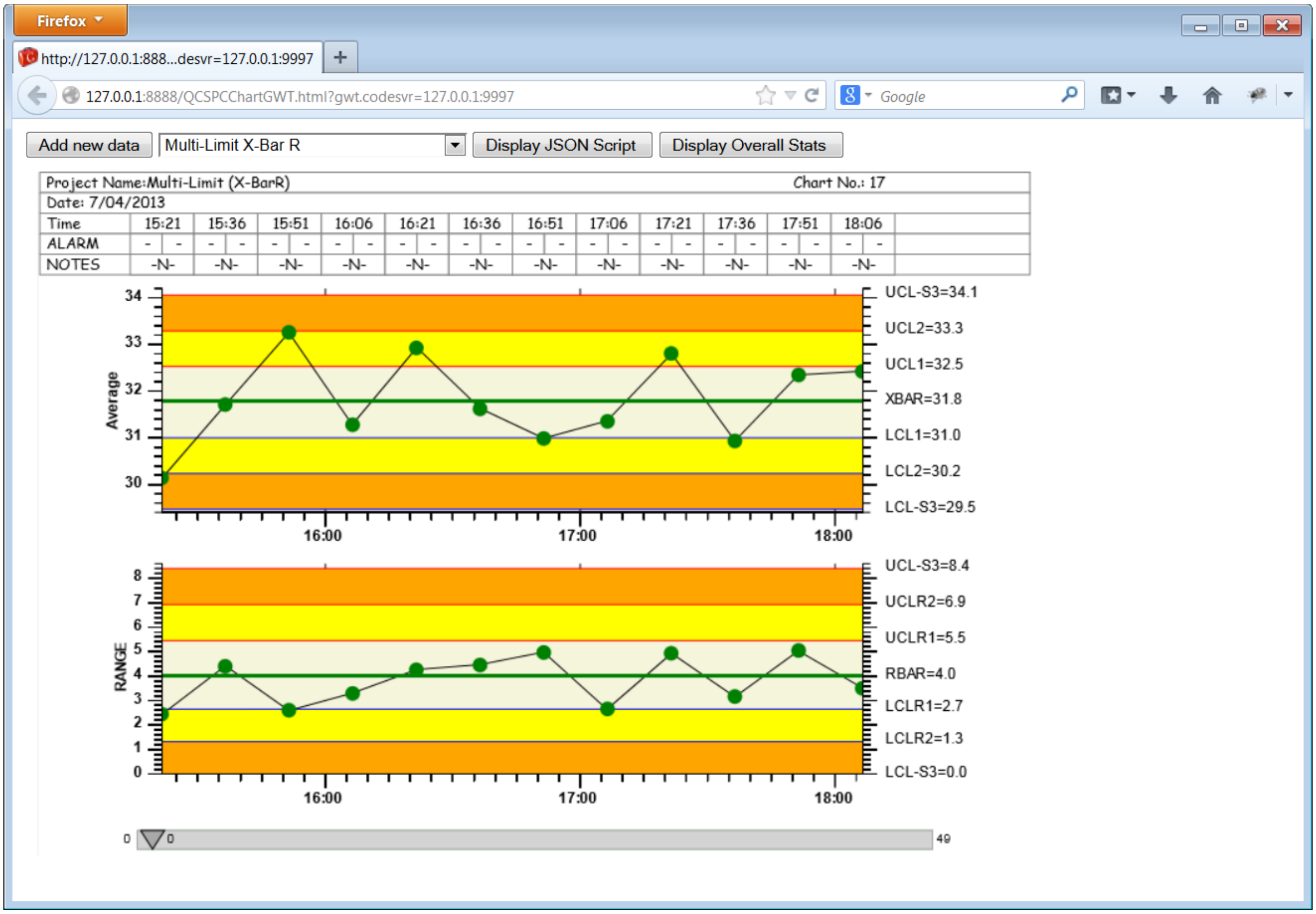


Project Name:Multi-Limit (X-BarR) Chart No.: 17
Date: 7/04/2013

Time	14:17	14:32	14:47	15:02	15:17	15:32	15:47	16:02	16:17	16:32	16:47	17:02
ALARM	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -
NOTES	-N-	-N-	-N-	-N-	-N-	-N-	-N-	-N-	-N-	-N-	-N-	-N-

We also added a method (**123SigmaControlLimits**) 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 chartDefExampleScripts.js TimeMultiLimitXBarRChart JSON script. 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 **123SigmaControlLimits** limits has the option of disabling alarm notification, using AlarmTest12: false, in the case of +-1 and +-2 alarm conditions.

```
"PrimaryChartSetup": {
    "ControlLimits": {
        "123SigmaControlLimits": {
            "Target": 32,
            "LCL3Value": 28,
            "UCL3Value": 36,
            "AlarmTest12": false
        },
        "ZoneFill": true,
        "ZoneColors": [
                         "ORANGE",
                         "YELLOW",
                         "BEIGE"
                      ]
    }
},
"SecondaryChartSetup": {
    "ControlLimits": {
        "123SigmaControlLimits": {
            "Target": 2,
            "LCL3Value": 0,
            "UCL3Value": 5,
            "AlarmTest12": false

        },
        "ZoneFill": true,
        "ZoneColors": [
                         "ORANGE",
                         "YELLOW",
                         "BEIGE"
                     ]
    }
},
```


Firefox
http://127.0.0.1:888...desvr=127.0.0.1:9997
127.0.0.1:8888/QCSPCChartGWT.html?gwt.codesvr=127.0.0.1:9997
Google
Add new data
Multi-Limit X-Bar R
Display JSON Script
Display Overall Stats

Project Name:Multi-Limit (X-BarR)												Chart No.: 17
Date: 7/04/2013												
Time	15:21	15:36	15:51	16:06	16:21	16:36	16:51	17:06	17:21	17:36	17:51	18:06
ALARM	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -
NOTES	-N-	-N-	-N-	-N-	-N-	-N-	-N-	-N-	-N-	-N-	-N-	-N-

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 **SpecificationLimits** block**.** 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)
Secondary Chart	SPC_LOWER_CONTROL_LIMIT_2	(2-sigma upper limit)
Secondary Chart	SPC_UPPER_CONTROL_LIMIT_2	(2-sigma upper limit)
Secondary Chart	SPC_LOWER_CONTROL_LIMIT_1	(1-sigma upper limit)
Secondary Chart	SPC_UPPER_CONTROL_LIMIT_1	(1-sigma upper limit)

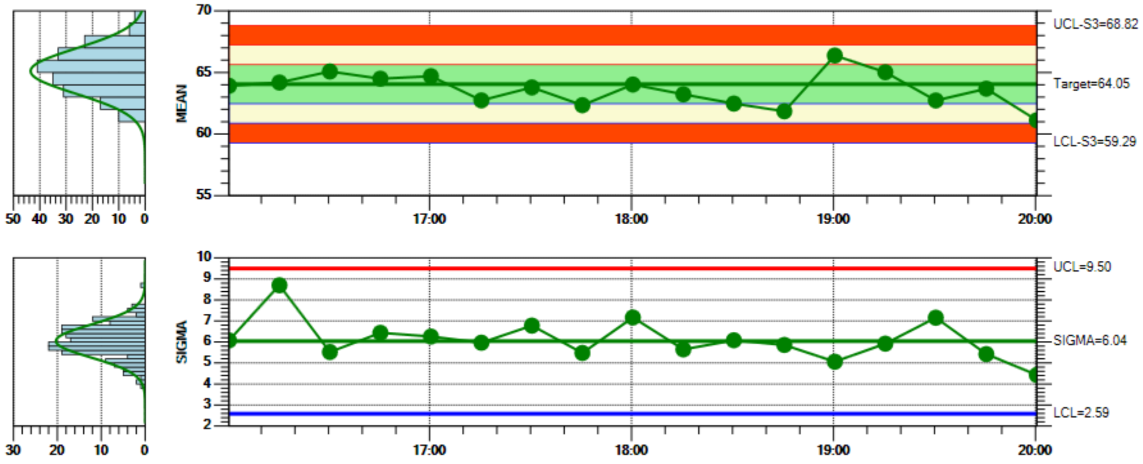
```
"AddControlRules": [
                 {
                    "RuleSet": "BASIC_RULES",
                    "RuleNumber":  3
                },
                {
                    "RuleSet": "BASIC_RULES",
                    "RuleNumber":  4
                },
                {
                    "RuleSet": "BASIC_RULES",
                    "RuleNumber":  5
                },
                {
                    "RuleSet": "BASIC_RULES",
                    "RuleNumber":  6
                }
            ]
```

Special Note – You can specify a specific value using the LimitValue propety. 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). The software know what sigma level is assigned to a given control rule, and 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 ZoneFill flag under ControlLimits.

```
"PrimaryChartSetup": {
    "ControlLimits": {
        "ZoneFill": false,
                "ZoneColors": [
                            "ORANGE",
                            "YELLOW",
                            "BEIGE"
                        ]
            }
        }
```

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. In order for the algorithm to work, 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.



If you are using any of the names rule sets (WECO, NELSON, ects.), call it in the ControlLimits block.

See the example program WERulesVariableControlCharts.XBarSigmaChart.

```
    "ControlLimits":
    {
       "ZoneFill": true,
       "NamedRuleSet":
       {
          "RuleSet": "WECO_RULES",
          "RuleEnable": [ true, true, true, true, true, true, true, true]
       }
  }
```

Excluding Control Limits from Auto-Scaling Calculations

Normally, all control and specification limits, whether or not they have been set to a valid value, are used in the auto-scaling calculation for the y-axis. If the chart is intended to display values in the range 41 to 42, including control limits at the default, unassigned value of 0.0 will force the y-axis to scale from 0 to 42, instead of 41 to 42, resulting in a loss of resolution in the displayed signal. If you want to exclude the control limit from the auto-scaling calculation, disable it from the display, and the auto-scaling, using the **EnableAlarmLine** property. The auto-scaling for that control limit is tied to that property. In the example below, the lower control limit (LCL3) is excluded from the auto-scaling calculation.

```
"ControlLimits": {
   "Target": {
      "DisplayString": "TargetXX",
      "EnableAlarmLine": true,
      "EnableAlarmChecking": true,
      "LimitValue": 30,
      "EnableAlarmLineText": true
   },
   "LCL3": {
      "DisplayString": "LCLXX",
      "EnableAlarmLine": false,
      "EnableAlarmChecking": true,
      "LimitValue": 25,
      "EnableAlarmLineText": false
   },
   "UCL3": {
      "DisplayString": "UCLXX",
      "EnableAlarmLine": false,
      "EnableAlarmChecking": true,
      "LimitValue": 35,
      "EnableAlarmLineText": true
    }
  }
```

The same EnableAlarmLine property is also found in the Specification Limits block. Set it to false to not have it show, or be included in the auto-scaling calculations. In the example below, the LowSpecificationLimit is excluded.

```
"SpecificationLimits":
{
    "LowSpecificationLimit":
    {
       "LimitValue": 15,
        "EnableAlarmLine": false,
       "LineColor": "BLUE",
       "DisplayString": "LSLX"
    },
   "HighSpecificationLimit":
    {
       "LimitValue": 40,
       "LineColor": "RED",
       "DisplayString": "HSLX"

    }
}
```

Named Rule Sets

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. These are described in detail in Chapter 13. 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:

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 RuleSet property to WECO_RULES, using the PrimaryChartSetup **NamedRuleSet** block. When the variable control charts **AutoCalculatedControlLimits** method is called, the software automatically calculates all of the appropriated control limits, based on the current data.

If you want to include the WECO Trending (Supplemental) rules, in addition to the regular WECO Runtime rules, set the RuleSet property to WECOANDSUPP_RULES instead of WECO_RULES.

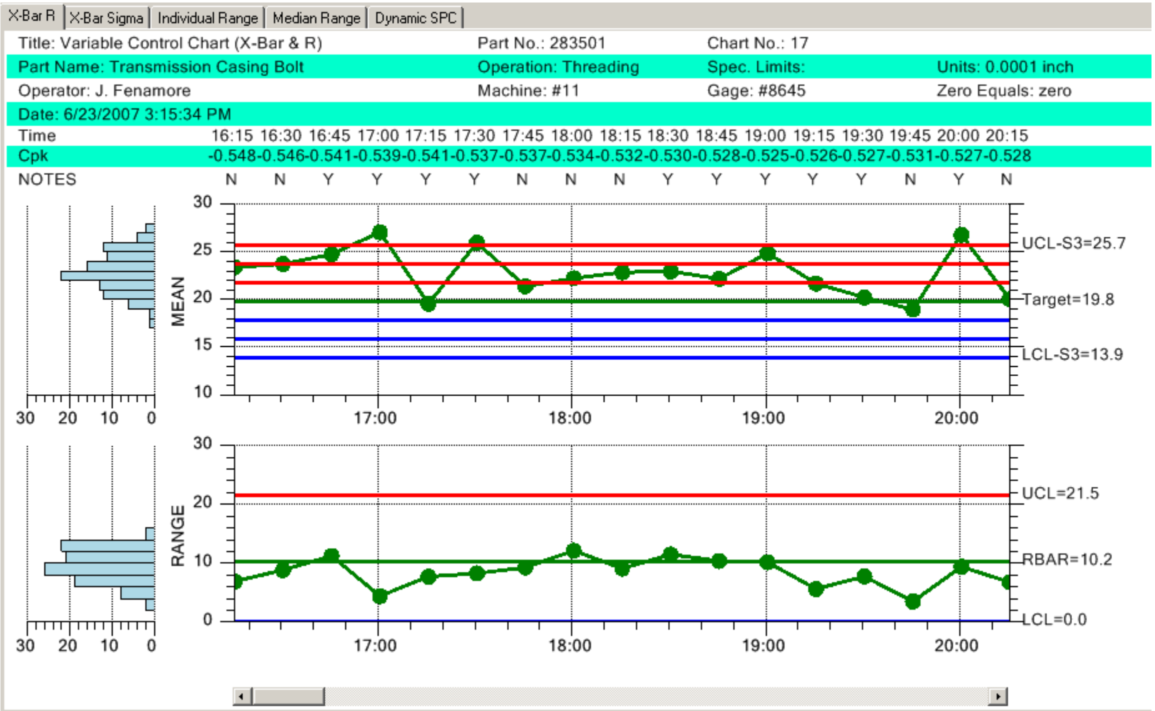
```
    "ControlLimits":
    {
      "ZoneFill": true,
      "NamedRuleSet":
      {
          "RuleSet": "WECOANDSUPP_RULES",
          "RuleEnable": [ true, true, true, true, true]
      }
    }
```

If you have enable alarm event processing, the software will call your Javascript 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. See Chapter 14 – Event Handling for Alarms and Tooltips for details about alarm event handling.

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 MiscChartDataProperties.AlarmReportMode property to REPORT_ALL_ALARMS.

```
"MiscChartDataProperties": {
   "AlarmReportMode": "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 **SpecificationLimits** block of the PrimaryChartSetup or SecondaryChartSetup block.

```
"SpecificationLimits":
{
   "LowSpecificationLimit":
   {
```

```
            "LimitValue": 15
         },
         "HighSpecificationLimit":
         {
            "LimitValue": 40
         }
       }
```

Set the **EnableAlarmLine** property to false to not have it show, or be included in the auto-scaling calculations. In the example below, the LowSpecificationLimit is excluded.

```
"SpecificationLimits":
{
    "LowSpecificationLimit":
    {
       "LimitValue": 15,
        "EnableAlarmLine": false,
       "LineColor": "BLUE",
       "DisplayString": "LSLX"
    },
   "HighSpecificationLimit":
    {
       "LimitValue": 40,
       "LineColor": "RED",
       "DisplayString": "HSLX"

    }
}
```

Chart Y-Scale

You can set the minimum and maximum values of the two charts y-scales manually using the YAxisLeft bloc of properties.

```
"YAxisLeft":
{
       "MinValue": 10,
       "MaxValue": 45
}
```

It is easiest to just call the auto-scale routines after the chart has been initialized with data, and any control limits calculated.

```
      "Methods": {
            "AutoCalculateControlLimits": true,
            "AutoScaleYAxes": true,
            "RebuildUsingCurrentData": true
        }
```

Once all of the graph parameters are set, call the method **RebuildUsingCurrentData**.

If, at any future time you change any of the chart properties, you will need to process **Methods**.**RebuildUsingCurrentData** to force a rebuild of the chart, taking into account the current properties. **RebuildUsingCurrentData** also invalidates the chart and forces a redraw.

Updating Chart Data

The real-time example above demonstrates how the SPC chart data is updated, using the SampleData.SampleIntervalRecords array property.

```
    "SPCChart": {

        "SampleData": {
            "SampleIntervalRecords": [
                {
                    "SampleValues": [
                        27.53131515148628,
                        33.95771604022404,
                        24.310097827061817,
                        28.282642847792765,
                        30.2908518818265
                    ],
                    "BatchCount": 50,
                    "TimeStamp": 1371830829074,
                    "BatchIDString": "IDS50",
                    "Note": ""
                },
                {
                    "SampleValues": [
                        27.444285005240214,
                        34.38930645615096,
                        28.0203674441636,
                        33.27153359969366,
                        36.8305571558275
                    ],
                    "BatchCount": 51,
                    "TimeStamp": 1371831729074,
                    "BatchIDString": "IDS51",
                    "Note": ""
                },
                {
                    "SampleValues": [
                        35.21321620109259,
                        32.93940741018088,
                        33.66485557976163,
                        34.17314124609133,
                        24.576683179863725
                    ],
                    "BatchCount": 52,
                    "TimeStamp": 1371832629074,
                    "BatchIDString": "IDS52",
                    "Note": ""
                },
                {
                    "SampleValues": [
                        27.898302097237174,
                        25.906531082892915,
                        26.950768095191137,
                        30.812058501916457,
                        31.085075984847936
                    ],
                    "BatchCount": 53,
                    "TimeStamp": 1371833529074,
                    "BatchIDString": "IDS53",
                    "Note": ""
                }

            ]
        },
        "Methods": {
            "AutoCalculateControlLimits": true,
            "AutoScaleYAxes": true,
            "RebuildUsingCurrentData": true
        }
    }
}
```

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. Make sure you start the BatchCount from where you left off. Don't start at 0 again. If you are using a Time-based control chart, make sure the time value of the TimeStamp starts where the old time stamp stopped.

Note: Since there is no reliable standard across browsers for time/date data, this value is expressed as the Unix standard of elapsed milliseconds since Thursday, 1 January 1970. The TimeStamp value is used in both time-based SPC Charts, and batch-based SPC Charts, so you must be able to convert from time, to the millisecond equivalent value in order to input the time stamp.

If you want to include a text note in the sample record, just fillout the appropriate Note property of the appropriate SampleIntervalRecords item.

```
        "SPCChart": {

            "SampleData": {
```

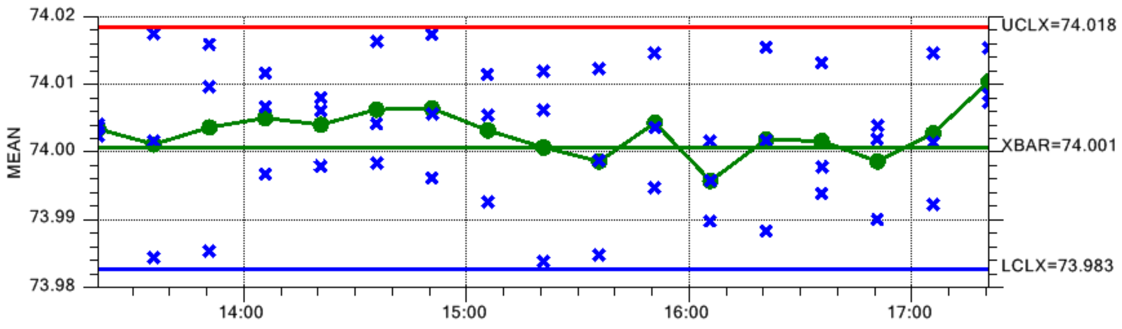
```
"SampleIntervalRecords": [
    {
        "SampleValues": [
            27.53131515148628,
            33.95771604022404,
            24.310097827061817,
            28.282642847792765,
            30.2908518818265
        ],
        "BatchCount": 50,
        "TimeStamp": 1371830829074,
        "BatchIDString": "IDS50",
        "Note": "Important Note #14"
    },
    {
        "SampleValues": [
            27.444285005240214,
            34.38930645615096,
            28.0203674441636,
            33.27153359969366,
            36.8305571558275
        ],
        "BatchCount": 51,
        "TimeStamp": 1371831729074,
        "BatchIDString": "IDS51",
        "Note": ""
    },
    {
        "SampleValues": [
            35.21321620109259,
            32.93940741018088,
            33.66485557976163,
            34.17314124609133,
            24.576683179863725
        ],
        "BatchCount": 52,
        "TimeStamp": 1371832629074,
        "BatchIDString": "IDS52",
        "Note": "Importan Note #15"
    },
    {
        "SampleValues": [
            27.898302097237174,
            25.906531082892915,
            26.950768095191137,
            30.812058501916457,
            31.085075984847936
        ],
        "BatchCount": 53,
        "TimeStamp": 1371833529074,
        "BatchIDString": "IDS53",
        "Note": ""
    }

]
```

```
                },
                "Methods": {
                    "AutoCalculateControlLimits": true,
                    "AutoScaleYAxes": true,
                    "RebuildUsingCurrentData": true
                }
            }
        }
```

There are situations where you might want to add, change, modify, or append a note for a sample subgroup after the sample record has already been added. This can happen if the **adding a new sample subgroup triggers an alarm** 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 AddNote block to add notes to the current record, independent of the `SampleData` block.

```
"MiscChartDataProperties": {
          "AddNote": [
             {
               "Index": 10,
               "Note": "Important Note #1",
               "Append": true
             } ,
             {
               "Index": 17,
               "Note": "Important Note #2",
               "Append": true
             }
          ]
  }
```

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 **PrimaryChartSetup**.**PlotMeasurementValues** property to true.

```
      "PrimaryChartSetup": {
           "PlotMeasurementValues": true
       },
```

Enable the Chart ScrollBar

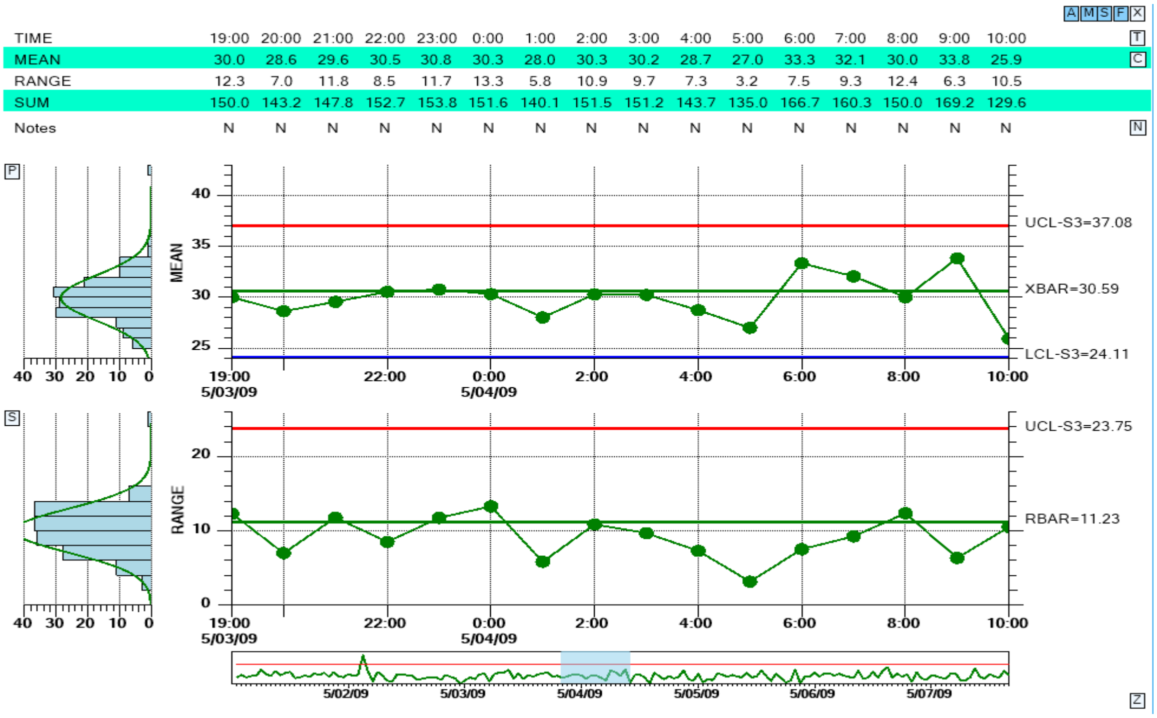
Set the **Scrollbar**.**EnableScrollBar** property true to enable the chart scrollbar. You will then be able to window in on 8-20 sample subgroups at a time, from a much larger collection of measurement data representing hundreds or even thousands of subgroups, and use the scrollbar to move through the data, similar to paging through a spreadsheet.

```
"Scrollbar": {
             "EnableScrollBar": true,
             "ScrollbarPosition": "SCROLLBAR_POSITION_MAX"
},
```

You can also set the initial value of the scroll bar so some know value, using the ScrollbarValue property, or you can force the go to the maximum value of the scroll bar after any data updates. That way the most recent data will always be in view. Or, you can specify that after a RebuildUsingCurrentData, which usually increases the scroll bars range of values, that the scrollbar position to show the must recently added data ("SCROLLBAR_POSITION_MAX"), or the oldest data ("SCROLLBAR_POSITION_MIN").

Zooming as an option for the scrollbar

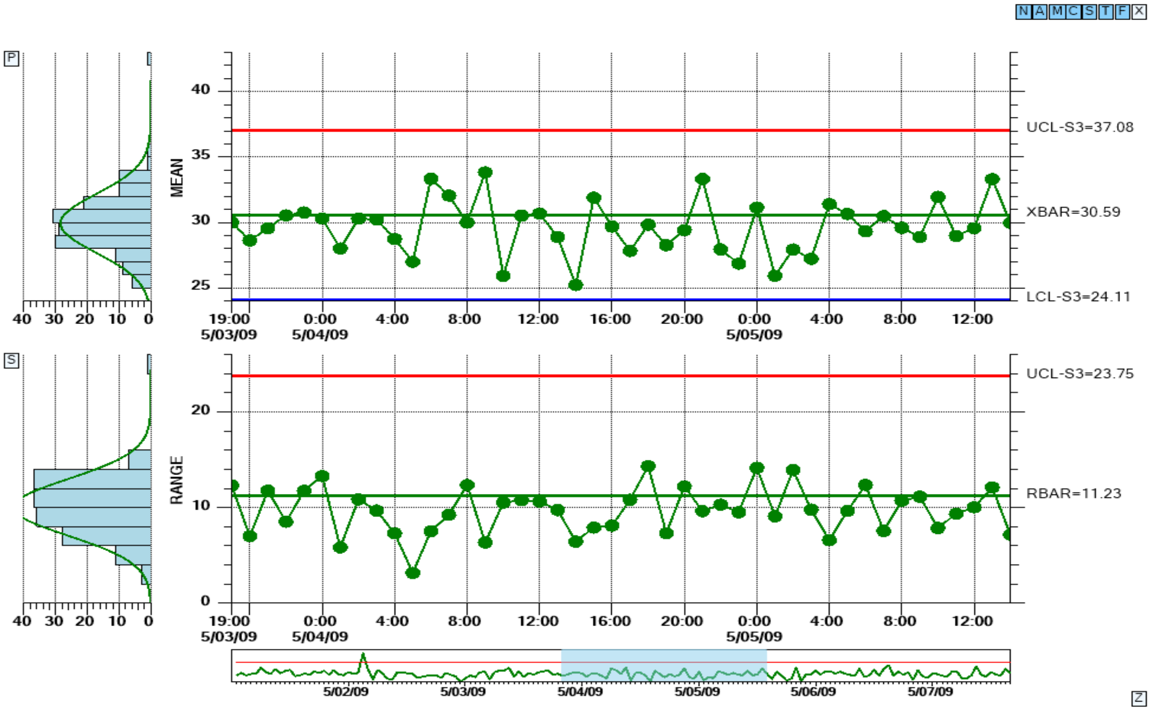
The horizontal scrollbar can be replaced (toggled on/off actually) using the UI, with a zoom window, by clicking on the z-button in the lower right corner. The user will be able to scroll the charts by dragging the transparent blue zoom window left or right, or expand the x-axis scale by dragging the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

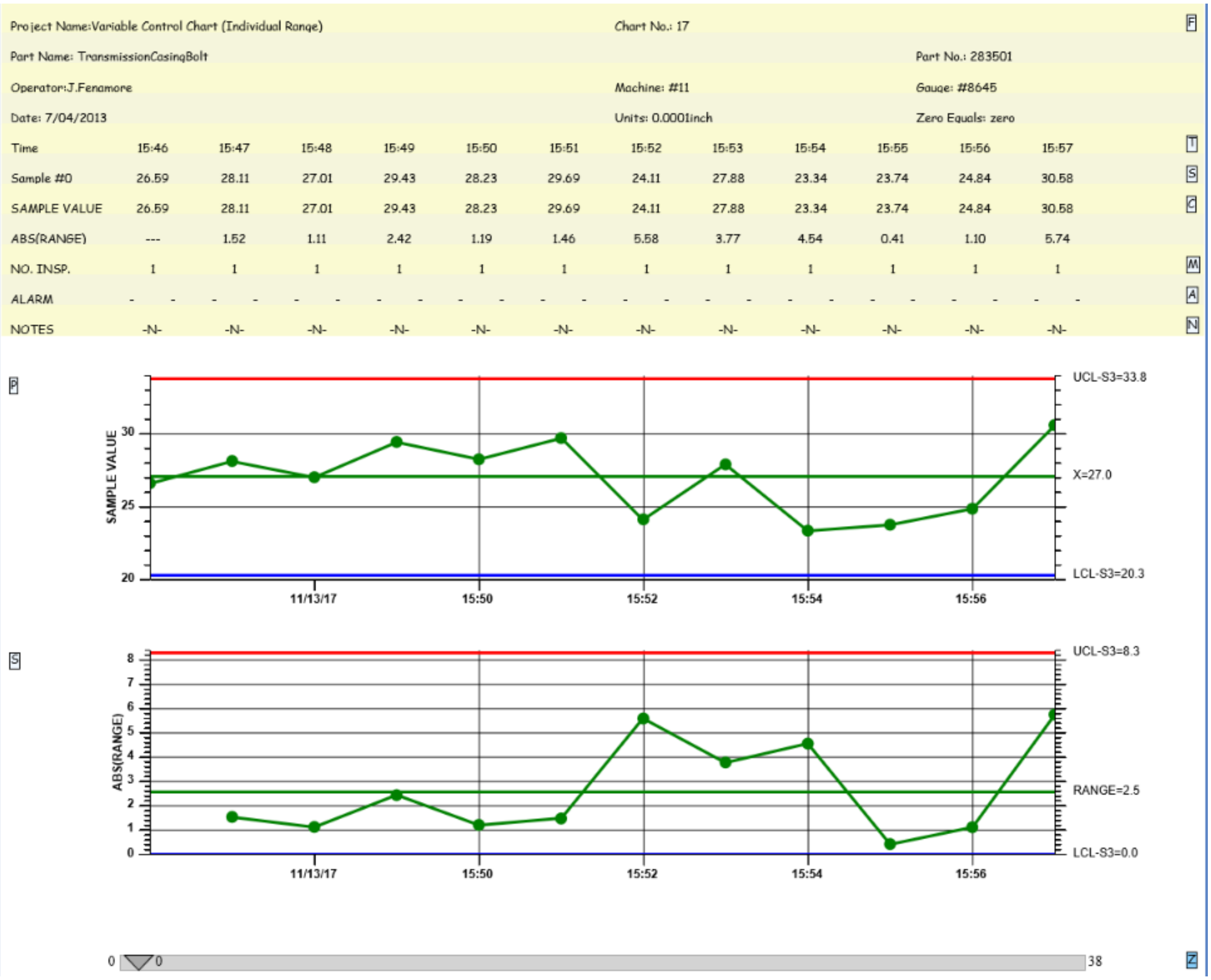
.

```
            "EnableZoomToggle": false
```

Since the zoom toggle is on by default, all of the examples 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.



Project Name:Variable Control Chart (Individual Range) Chart No.: 17

Part Name: TransmissionCasingBolt Part No.: 283501

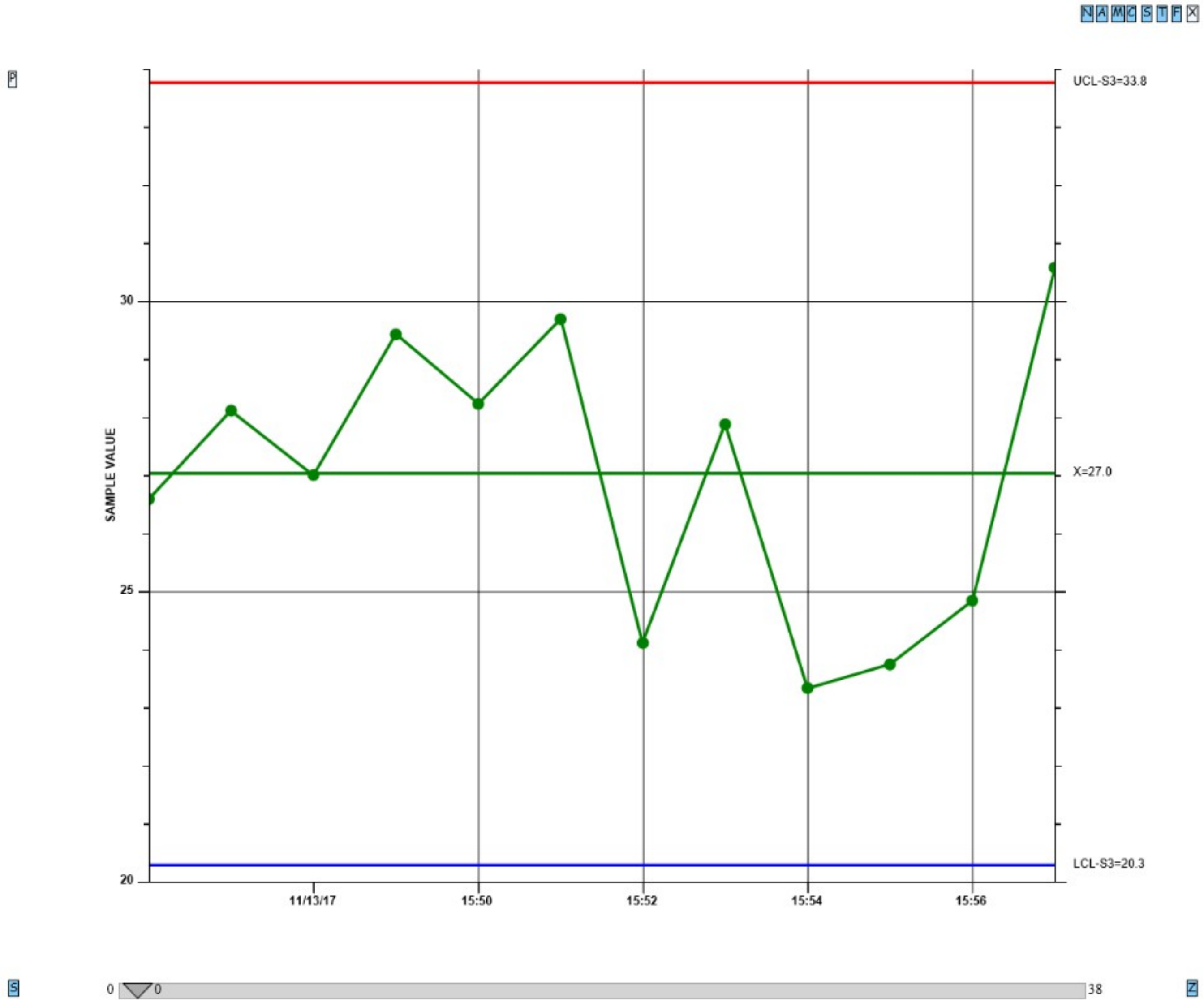
Operator:J.Fenamore Machine: #11 Gauge: #8645

Date: 7/04/2013 Units: 0.0001inch Zero Equals: zero

Time	15:46	15:47	15:48	15:49	15:50	15:51	15:52	15:53	15:54	15:55	15:56	15:57
Sample #0	26.59	28.11	27.01	29.43	28.23	29.69	24.11	27.88	23.34	23.74	24.84	30.58
SAMPLE VALUE	26.59	28.11	27.01	29.43	28.23	29.69	24.11	27.88	23.34	23.74	24.84	30.58
ABS(RANGE)	---	1.52	1.11	2.42	1.19	1.46	5.58	3.77	4.54	0.41	1.10	5.74
NO. INSP.	1	1	1	1	1	1	1	1	1	1	1	1
ALARM	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -
NOTES	-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.

```
“PrimaryChartSetup”: {
   “EnableChartDisplayToggles”: false
}
```

Hide the table buttons on the right.

```
"TableSetup": {
          “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.

```
“MiscChartDataProperties”: {
         “EnableDisplayOptionToggles”: false
}
```

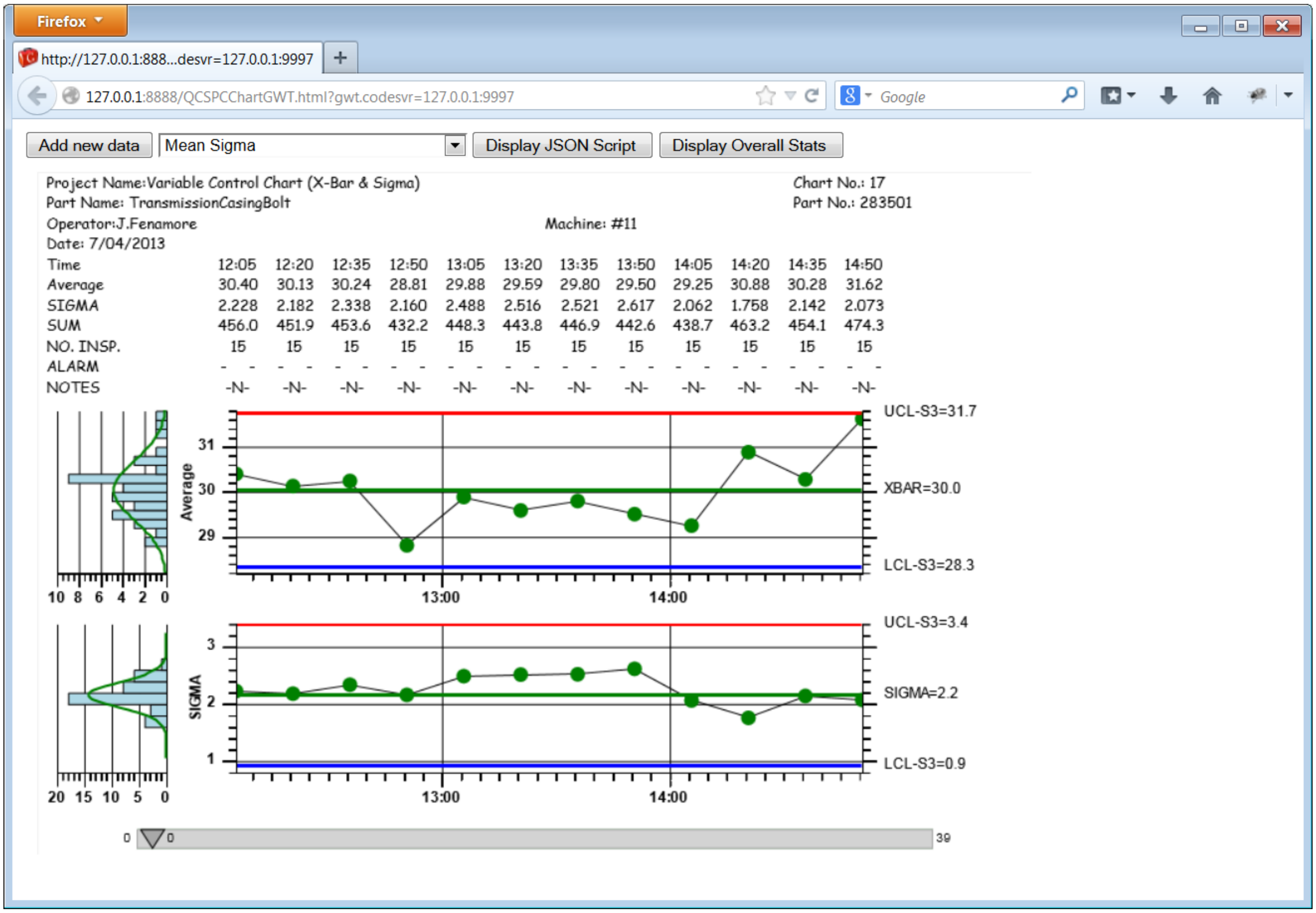
If you want to selectively enable options, first set EnableDisplayOptionToggles true. The other button display enables won't work unless this option is true. Then disable the options you do not want to show. In the example below, only the zoom option is left visible. Note that each toggle is in a different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

```
“MiscChartDataProperties”: {
         “EnableDisplayOptionToggles”: false
}

"TableSetup": {
          “EnableTableRowToggles”: false
}

“PrimaryChartSetup”: {
   “EnableChartDisplayToggles”: false
}

“SPCChart”: {
   “Scrollbar”: {
```

```
            “EnableZoomToggle”: true
    }
}
```

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 **PrimaryChartSetup.FrequencyHistogram.EnableDisplayFrequencyHistogram** and **SecondaryChartSetup.FrequencyHistogram.EnableDisplayFrequencyHistogram** properties of the chart.Enhanced


Firefox
http://127.0.0.1:888...desvr=127.0.0.1:9997
127.0.0.1:8888/QCSPCChartGWT.html?gwt.codesvr=127.0.0.1:9997
Google
Add new data
Mean Sigma
Display JSON Script
Display Overall Stats

Project Name:Variable Control Chart (X-Bar & Sigma)
Part Name: TransmissionCasingBolt
Operator:J.Fenamore
Date: 7/04/2013

Machine: #11

Chart No.: 17
Part No.: 283501

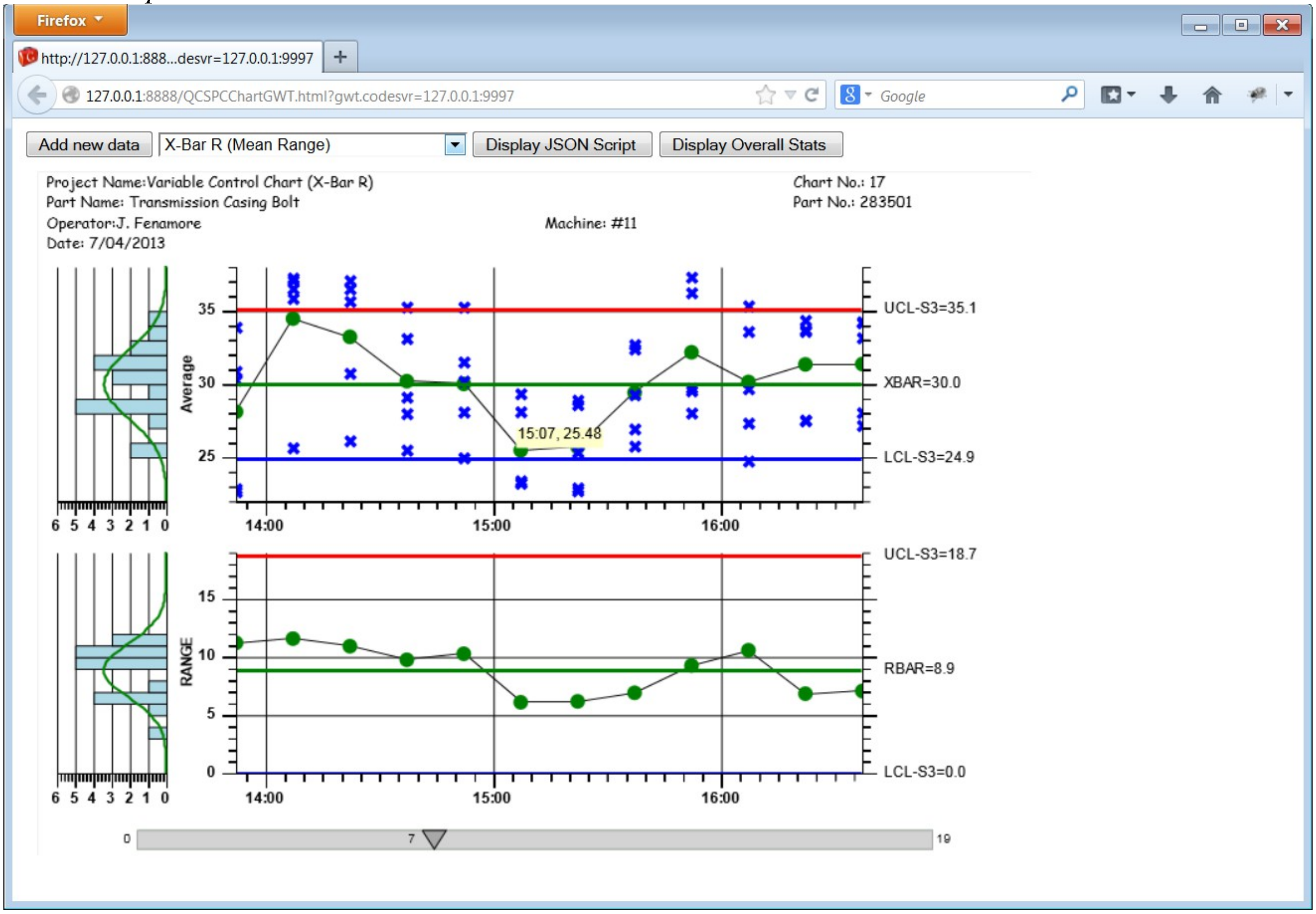
Time	12:05	12:20	12:35	12:50	13:05	13:20	13:35	13:50	14:05	14:20	14:35	14:50
Average	30.40	30.13	30.24	28.81	29.88	29.59	29.80	29.50	29.25	30.88	30.28	31.62
SIGMA	2.228	2.182	2.338	2.160	2.488	2.516	2.521	2.617	2.062	1.758	2.142	2.073
SUM	456.0	451.9	453.6	432.2	448.3	443.8	446.9	442.6	438.7	463.2	454.1	474.3
NO. INSP.	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-

```
 "PrimaryChartSetup": {
   "FrequencyHistogram": {
          "EnableDisplayFrequencyHistogram": true
       }
 },
 "SecondaryChartSetup": {
   "FrequencyHistogram": {
           "EnableDisplayFrequencyHistogram": 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



In the default mode, the data tooltip displays the x,y value if you hover over the datapoint. 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 chartdefSampleScripts.js CustomizeChartAppearance example.

```
"Events": {
    "EnableDataToolTip": true,
    "EnableJSONDataToolTip": false,
    "AlarmStateEventEnable": false,
    "DataToolTip":
      {
         "EnableCategoryValues": true,
         "EnableProcessCapabilityValues": true,
         "EnableCalculatedValues": true,
         "EnableNotesString": true
      }

}
```

where

Events.DataToolTip.EnableCategoryValues
Display the category (subgroup sample values) in the data tooltip.

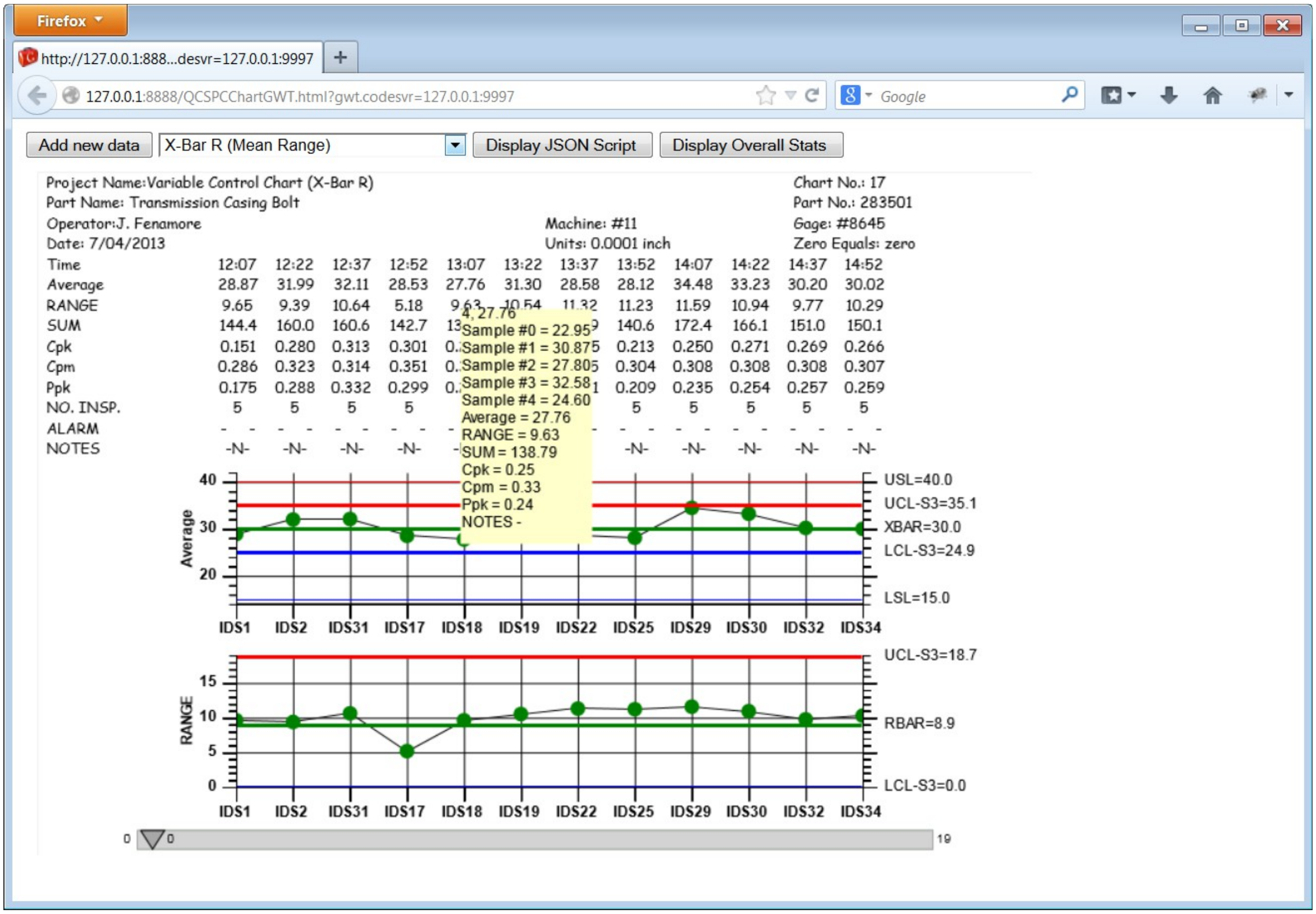
Events.DataToolTip.EnableProcessCapabilityValues
Display the process capability (Cp, Cpl, Cpu, Cpk, Cpm, Pp, Ppl, Ppu and Ppk) statistics currently being calculated for the chart.

Events.DataToolTip.EnableCalculatedValues
Display the calculated values used in the chart (Mean, range and sum for an Mean-Range chart).

Events.DataToolTip.EnableNotesStrings
Display the current notes string for the sample subgroup.

The variable control chart below displays a tooltip with all of the enable options above set true.

Data Tooltip with optional display items



Firefox
http://127.0.0.1:888...desvr=127.0.0.1:9997
127.0.0.1:8888/QCSPCChartGWT.html?gwt.codesvr=127.0.0.1:9997
Google
Add new data
X-Bar R (Mean Range)
Display JSON Script
Display Overall Stats

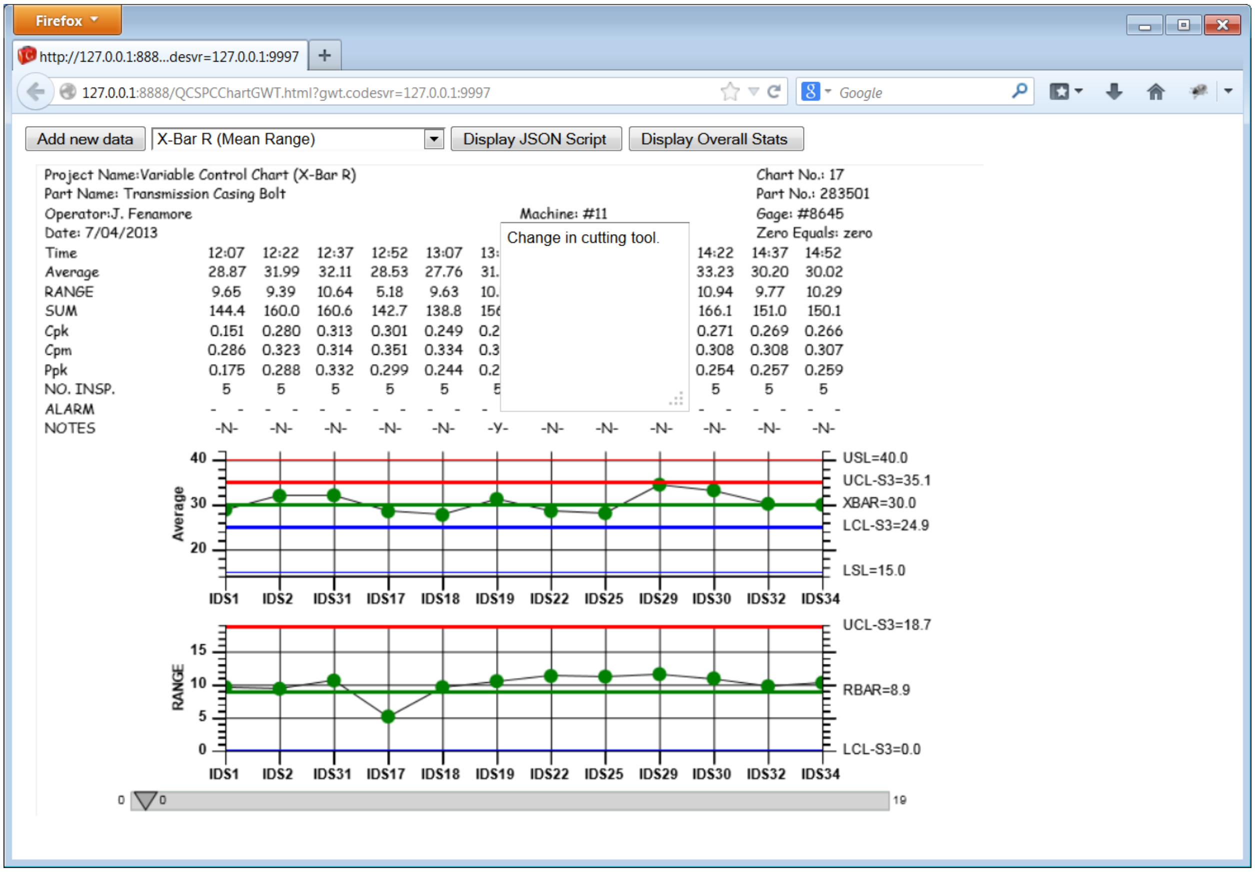
Project Name:Variable Control Chart (X-Bar R) Chart No.: 17
Part Name: Transmission Casing Bolt Part No.: 283501
Operator:J. Fenamore Machine: #11 Gage: #8645
Date: 7/04/2013 Units: 0.0001 inch Zero Equals: zero

Time	12:07	12:22	12:37	12:52	13:07	13:22	13:37	13:52	14:07	14:22	14:37	14:52
Average	28.87	31.99	32.11	28.53	27.76	31.30	28.58	28.12	34.48	33.23	30.20	30.02
RANGE	9.65	9.39	10.64	5.18	9.63	10.54	11.32	11.23	11.59	10.94	9.77	10.29
SUM	144.4	160.0	160.6	142.7	13		9	140.6	172.4	166.1	151.0	150.1
Cpk	0.151	0.280	0.313	0.301	0.		5	0.213	0.250	0.271	0.269	0.266
Cpm	0.286	0.323	0.314	0.351	0.		5	0.304	0.308	0.308	0.308	0.307
Ppk	0.175	0.288	0.332	0.299	0.		1	0.209	0.235	0.254	0.257	0.259
NO. INSP.	5	5	5	5				5	5	5	5	5
ALARM	- -	- -	- -	- -	-		-	- -	- -	- -	- -	- -
NOTES	-N-	-N-	-N-	-N-	-			-N-	-N-	-N-	-N-	-N-

4, 27.76
Sample #0 = 22.95
Sample #1 = 30.875
Sample #2 = 27.805
Sample #3 = 32.58
Sample #4 = 24.60
Average = 27.76
RANGE = 9.63
SUM = 138.79
Cpk = 0.25
Cpm = 0.33
Ppk = 0.24
NOTES -

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. 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 editable dialog box, immediately above the "Y".

Notes Tooltip



Firefox
http://127.0.0.1:888...desvr=127.0.0.1:9997
127.0.0.1:8888/QCSPCChartGWT.html?gwt.codesvr=127.0.0.1:9997
Google
Add new data
X-Bar R (Mean Range)
Display JSON Script
Display Overall Stats

Project Name:Variable Control Chart (X-Bar R)
Part Name: Transmission Casing Bolt
Operator:J. Fenamore
Date: 7/04/2013

Machine: #11

Chart No.: 17
Part No.: 283501
Gage: #8645
Zero Equals: zero

Change in cutting tool.

Time	12:07	12:22	12:37	12:52	13:07	13:				14:22	14:37	14:52
Average	28.87	31.99	32.11	28.53	27.76	31.				33.23	30.20	30.02
RANGE	9.65	9.39	10.64	5.18	9.63	10.				10.94	9.77	10.29
SUM	144.4	160.0	160.6	142.7	138.8	156				166.1	151.0	150.1
Cpk	0.151	0.280	0.313	0.301	0.249	0.2				0.271	0.269	0.266
Cpm	0.286	0.323	0.314	0.351	0.334	0.3				0.308	0.308	0.307
Ppk	0.175	0.288	0.332	0.299	0.244	0.2				0.254	0.257	0.259
NO. INSP.	5	5	5	5	5	5				5	5	5
ALARM	- -	- -	- -	- -	- -	-				- -	- -	- -
NOTES	-N-	-N-	-N-	-N-	-N-	-Y-	-N-	-N-	-N-	-N-	-N-	-N-

Both kinds of tooltips are on by default. Turn the tooltips on or off in the program using the **Events.EnableDataToolTip** and **Events.EnableNotesToolTip** properties.

```
"Events": {
    "EnableDataToolTip": true,
    "EnableNotesToolTip": true,
    "EnableJSONDataToolTip": false,
    "AlarmStateEventEnable": false,
    "DataToolTip":
      {
          "EnableCategoryValues": true,
          "EnableProcessCapabilityValues": true,
          "EnableCalculatedValues": true,
          "EnableNotesString": true
      }

 },
```

The notes tooltip has an additional option. In order to make the notes tooltip "editable", the notes edit box, displays on the first click, and goes away on the second click. You can click inside the notes box 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, set the **Events.NotesToolTip.ToolTipMode** property to **MOUSEDOWN_TOOLTIP**, as in the example below.

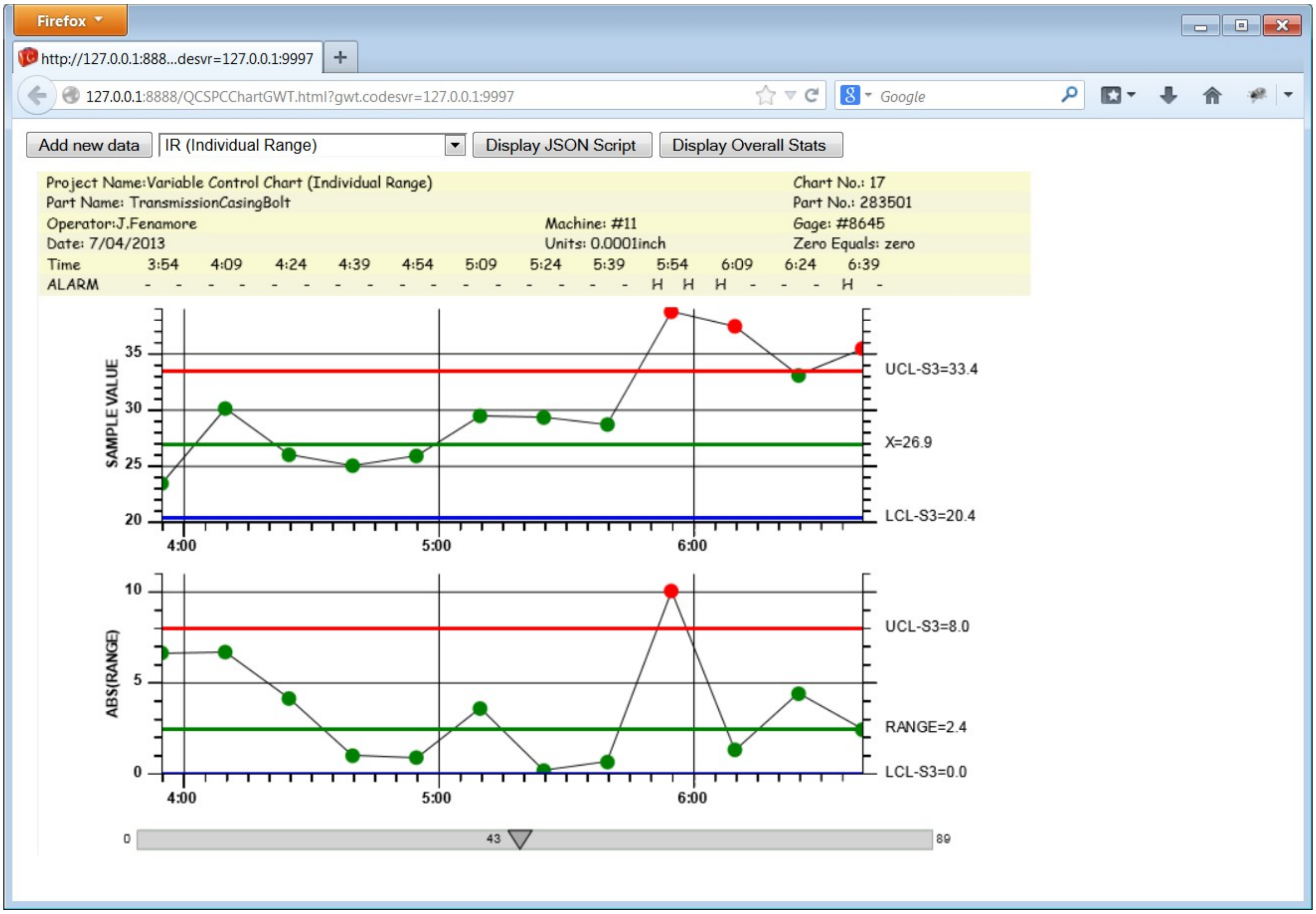
```
"Events": {
    "EnableDataToolTip": true,
    "EnableNotesToolTip": true,
    "EnableJSONDataToolTip": false,
    "AlarmStateEventEnable": false,
    "DataToolTip":
      {
         "EnableCategoryValues": true,
         "EnableProcessCapabilityValues": true,
         "EnableCalculatedValues": true,
         "EnableNotesString": true
      },
      "NotesToolTip": {
         "ToolTipMode": "MOUSEDOWN_TOOLTIP",
         "NotesReadOnly": true
      }

 },
```

Enable Alarm Highlighting

EnableAlarmStatusValues

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.



Project Name:Variable Control Chart (Individual Range) Chart No.: 17
Part Name: TransmissionCasingBolt Part No.: 283501
Operator:J.Fenamore Machine: #11 Gage: #8645
Date: 7/04/2013 Units: 0.0001inch Zero Equals: zero
Time 3:54 4:09 4:24 4:39 4:54 5:09 5:24 5:39 5:54 6:09 6:24 6:39
ALARM - - - - - - - - - - - - - - - - H H H - - - H -

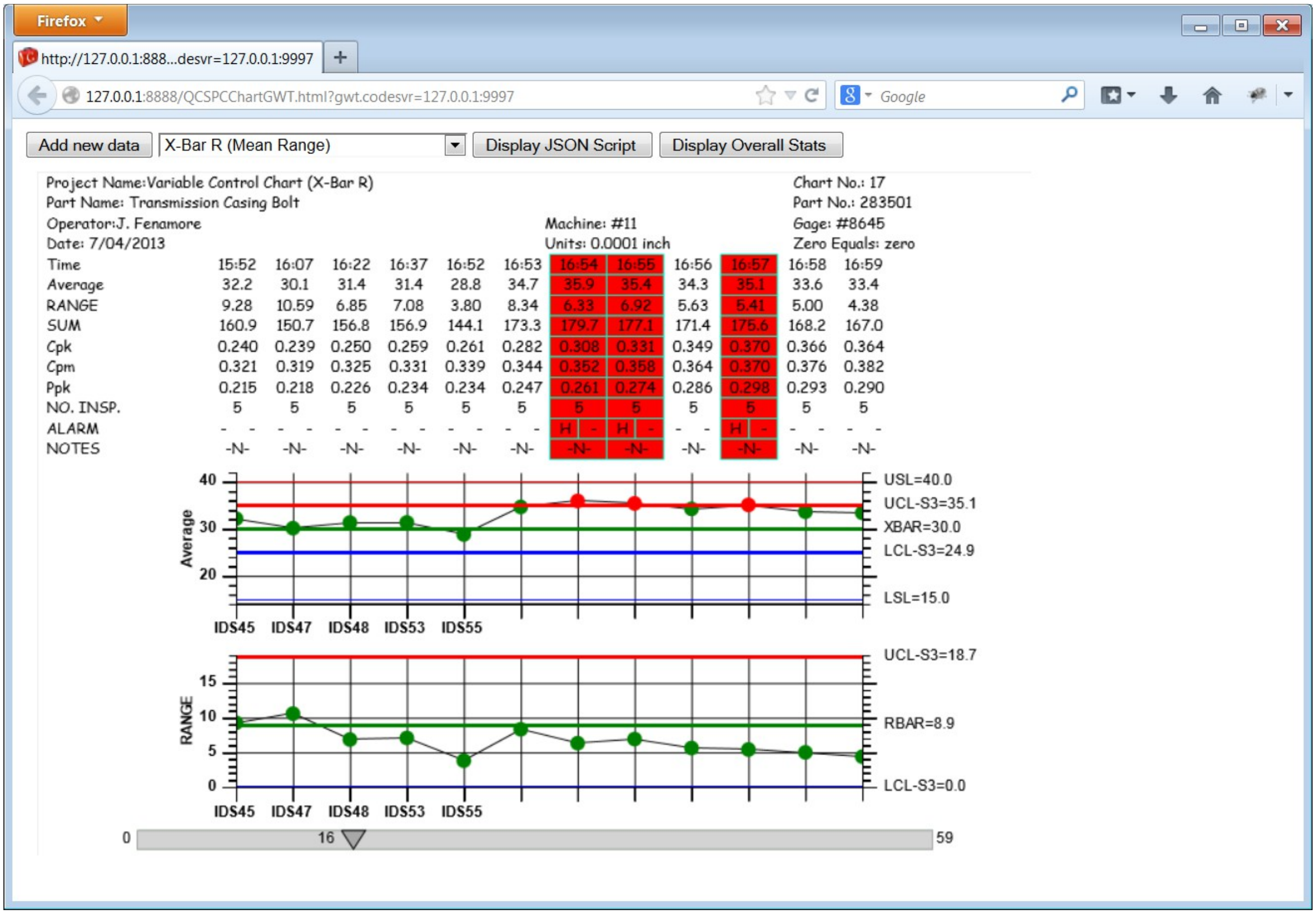
```
"Events": {
    "EnableDataToolTip": true,
    "EnableNotesToolTip": true,
    "EnableAlarmStatusValues": true

 },
```

ChartAlarmEmphasisMode

```
"Events": {
    "EnableDataToolTip": true,
    "EnableNotesToolTip": true,
    "EnableAlarmStatusValues": true,
    "ChartAlarmEmphasisMode":" 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 ALARM_NO_HIGHLIGHT_SYMBOL constant.

TableAlarmEmphasisMode -


Firefox
http://127.0.0.1:888...desvr=127.0.0.1:9997
127.0.0.1:8888/QCSPCChartGWT.html?gwt.codesvr=127.0.0.1:9997
Google
Add new data
X-Bar R (Mean Range)
Display JSON Script
Display Overall Stats

Project Name:Variable Control Chart (X-Bar R) Chart No.: 17
Part Name: Transmission Casing Bolt Part No.: 283501
Operator:J. Fenamore Machine: #11 Gage: #8645
Date: 7/04/2013 Units: 0.0001 inch Zero Equals: zero

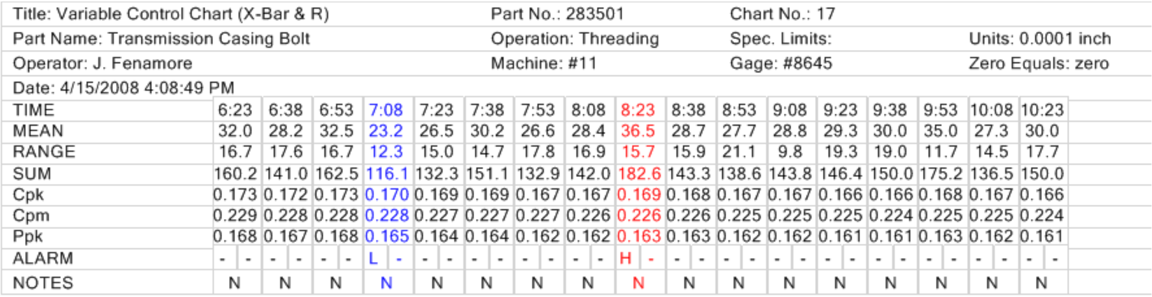
Time	15:52	16:07	16:22	16:37	16:52	16:53	16:54	16:55	16:56	16:57	16:58	16:59
Average	32.2	30.1	31.4	31.4	28.8	34.7	35.9	35.4	34.3	35.1	33.6	33.4
RANGE	9.28	10.59	6.85	7.08	3.80	8.34	6.33	6.92	5.63	5.41	5.00	4.38
SUM	160.9	150.7	156.8	156.9	144.1	173.3	179.7	177.1	171.4	175.6	168.2	167.0
Cpk	0.240	0.239	0.250	0.259	0.261	0.282	0.308	0.331	0.349	0.370	0.366	0.364
Cpm	0.321	0.319	0.325	0.331	0.339	0.344	0.352	0.358	0.364	0.370	0.376	0.382
Ppk	0.215	0.218	0.226	0.234	0.234	0.247	0.261	0.274	0.286	0.298	0.293	0.290
NO. INSP.	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-

```
"TableSetup": {

        "TableAlarmEmphasisMode": "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

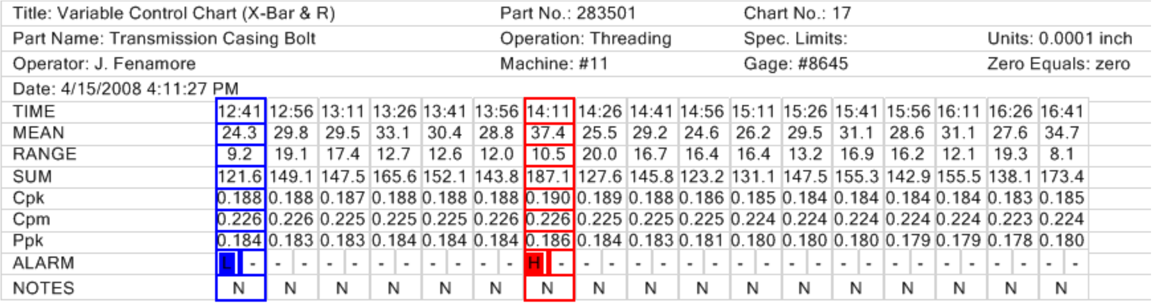
The example above uses the ALARM_HIGHLIGHT_BAR mode.



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.

AutoLogAlarmsAsNotes

When an alarm occurs, details of the alarm can be automatically logged as a Notes record. Just set the `MiscChartDataProperties.`AutoLogAlarmsAsNotes property to true.

```
"MiscChartDataProperties": {
   "AutoLogAlarmsAsNotes": true
}
```

Creating a Batch-Based Variable Control Chart

The batch-based and time-based SPC control charts 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 JSON scripts for a variety of batch-based SPC Charts.

```
"SPCChart": {
       "InitChartProperties": {
           "SPCChartType": "MEAN_RANGE_CHART",
           "ChartMode": "Batch",
           "NumSamplesPerSubgroup": 5,
           "NumDatapointsInView": 12,
           "TimeIncrementMinutes": 15
       },
```

Use the InitChartProperties.SPCChartType property to set the type of the chart: Mean Range (X-Bar R Median Range, X-Bar SigmaMean Sigma), Individual RangeEWMA, MA, or CuSum) . Note that the X-Bar Sigma chart, with a variable subgroup sample size, is initialized using 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.

The InitChartProperties block has the following properties.

SPCChartType

The SPC chart type parameter. Use one of the string constants strings: MEAN_RANGE_CHART, MEDIAN_RANGE_CHART, MEAN_SIGMA_CHART, MEAN_SIGMA_CHART_VSS, INDIVIDUAL_RANGE_CHART, EWMA_CHART, MA_CHART, MAMR_CHART, MAMS_CHART, LEVEY_JENNINGS_CHART and TABCUSUM_CHART,

ChartMode

Specifies if the x-axis is time-based (Time), or batch-base (Batch). Use the string constant string Time or Batch.

NumCategories

In an Attribute Control Charts this value represents the number of defect categories used to determine defect counts. Specify a numeric value, no quotes. Since the example above is for a Variable Control Chart (MEAN_RANGE_CHART), the NumCategories property does not need to be set.

NumSamplesPerSubgroup

Specifies the number of samples that make up a sample subgroup. If the SPCChartType is one of the variable sample size chart types, this value must be the maximum number of samples per subgroup. Specify a numeric value, no quotes.

NumDatapointsInView

Specifies the number of sample subgroups displayed in the graph at one time. Specify a numeric value, no quotes.

TimeIncrementMinutes

Specifies the approximate time increment (in minutes) between adjacent subgroup samples. This applies only to the Time ChartMode. Specify a numeric value, no quotes. Can be a double (0.5) to specify a fraction of a minute.

There are also three parameters which are used exclusively the CuSum chart type. You do not need to include them in any other chart.

CuSumKValue

A CuSum charts K value

CuSumHValue

A CuSum charts H value

CuSumMeanValue

A CuSum charts mean value

The following three properties apply to Multi-Variable Chart Types (QCSPCChart+). The Multi-Variable Chart have the SPCChartType set to one of the following five string constants: MULTIVARIABLE_MEAN_RANGE_CHART, MULTIVARIABLE_MEAN_SIGMA_CHART, MULTIVARIABLE_INDIVIDUAL_RANGE_CHART, MULTIVARIABLE_MEAN_RANGE_MR_CHART, and MULTIVARIABLE_MIXED_CHART

NumberAddedVariables

A multi-variable chart type has one main chart variable, and one or more added variables. The value of the *NumberAddedVariables* property specifies the number of added variables. So if you have the main variable, and two added variables, the *NumberAddedVariables* should be set to 2. The total number of variables in the chart will be three (the main, plus two added variables).

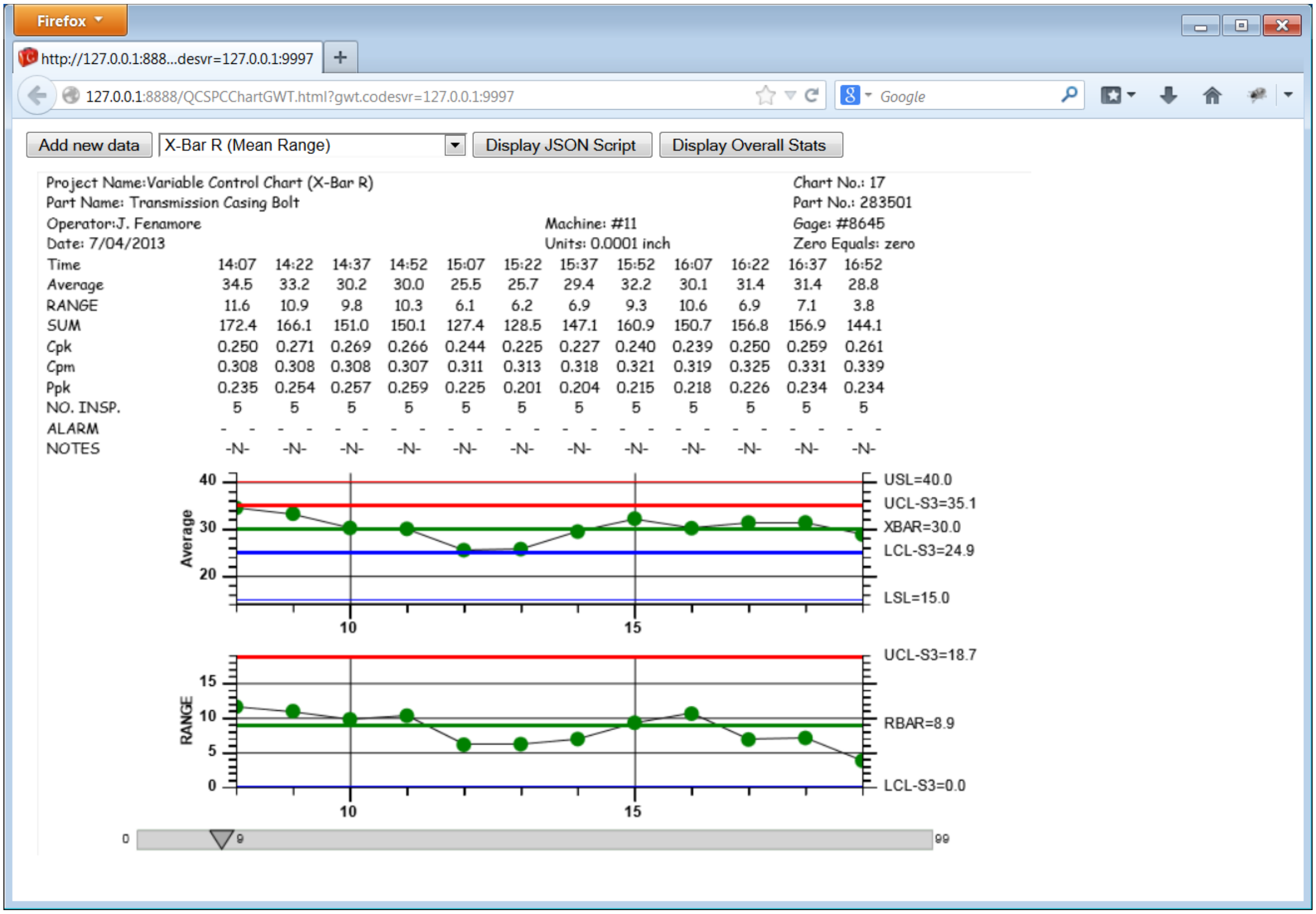
MultiVariableMixedPrimaryChartType

The MULTIVARIABLE_MIXED_CHART does not actually specify the chart type used for the main plot, and added multi-variable plots. Specify the chart type of the main plot by setting the MultiVariableMixedPrimaryChartType to one of the string constants: MEAN_RANGE_CHART, MEAN_SIGMA_CHART, or INDIVIDUAL_RANGE_CHART.

MultiVariableMixedChartSubTypes

The MULTIVARIABLE_MIXED_CHART does not actually specify the chart type used for the added multi-variable plots. Specify the added chart types using an array of strings, where the elements are the string constants: MEAN_RANGE_CHART, MEAN_SIGMA_CHART, or INDIVIDUAL_RANGE_CHART. You only specify the added chart types, the main chart type is specified using the MultiVariableMixedPrimaryChartType property.

Update the chart data using the **ChartData. SampleIntervalRecords** properties, and specify a batch number (*BatchCount* below). Even though a time stamp value is also specified, it is not used for positioning 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 chartDefExampleScripts.js **BatchXBarR** JSON script**.**



Project Name:Variable Control Chart (X-Bar R) Chart No.: 17
Part Name: Transmission Casing Bolt Part No.: 283501
Operator:J. Fenamore Machine: #11 Gage: #8645
Date: 7/04/2013 Units: 0.0001 inch Zero Equals: zero

Time	14:07	14:22	14:37	14:52	15:07	15:22	15:37	15:52	16:07	16:22	16:37	16:52
Average	34.5	33.2	30.2	30.0	25.5	25.7	29.4	32.2	30.1	31.4	31.4	28.8
RANGE	11.6	10.9	9.8	10.3	6.1	6.2	6.9	9.3	10.6	6.9	7.1	3.8
SUM	172.4	166.1	151.0	150.1	127.4	128.5	147.1	160.9	150.7	156.8	156.9	144.1
Cpk	0.250	0.271	0.269	0.266	0.244	0.225	0.227	0.240	0.239	0.250	0.259	0.261
Cpm	0.308	0.308	0.308	0.307	0.311	0.313	0.318	0.321	0.319	0.325	0.331	0.339
Ppk	0.235	0.254	0.257	0.259	0.225	0.201	0.204	0.215	0.218	0.226	0.234	0.234
NO. INSP.	5	5	5	5	5	5	5	5	5	5	5	5
ALARM	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -
NOTES	-N-	-N-	-N-	-N-	-N-	-N-	-N-	-N-	-N-	-N-	-N-	-N-

```
"SampleData": {
       "SampleIntervalRecords": [
           {
               "SampleValues": [
                   27.53131515148628,
                   33.95771604022404,
                   24.310097827061817,
                   28.282642847792765,
                   30.2908518818265
               ],
               "BatchCount": 0,
               "TimeStamp": 1371830829074,
               "BatchIDString": "IDS0",
               "Note": ""
```

```
             },
             {
                 "SampleValues": [
                     27.444285005240214,
                     34.38930645615096,
                     28.0203674441636,
                     33.27153359969366,
                     36.8305571558275
                 ],
                 "BatchCount": 1,
                 "TimeStamp": 1371831729074,
                 "BatchIDString": "IDS1",
                 "Note": ""
             },
             {
                 "SampleValues": [
                     35.21321620109259,
                     32.93940741018088,
                     33.66485557976163,
                     34.17314124609133,
                     24.576683179863725
                 ],
                 "BatchCount": 2,
                 "TimeStamp": 1371832629074,
                 "BatchIDString": "IDS2",
                 "Note": ""
             },
       .
       .
       .

             {
                 "SampleValues": [
                     30.56585901649224,
                     26.764807472584284,
                     30.22766077749437,
                     29.43260723522982,
                     27.080310485264213
                 ],
                 "BatchCount": 19,
                 "TimeStamp": 1371847929074,
                 "BatchIDString": "IDS19",
                 "Note": ""
             }
   ]
}
```

Changing the Batch Control Chart X-Axis Labeling Mode

The default mode of the x-axis tick marks of a batch control chart is to label them with the numeric batch number of the sample subgroup. It is also 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`* property found in all of the example programs.

```
       "SPCChart": {
              "InitChartProperties": {
                  "SPCChartType": "MEAN_RANGE_CHART",
                  "ChartMode": "Batch",
                  "NumSamplesPerSubgroup": 5,
                  "NumDatapointsInView": 9,
                  "TimeIncrementMinutes": 15
              },
```

You can rotate the x-axis labels using the charts PrimaryChartSetup.XAxisLabes.Rotation property .

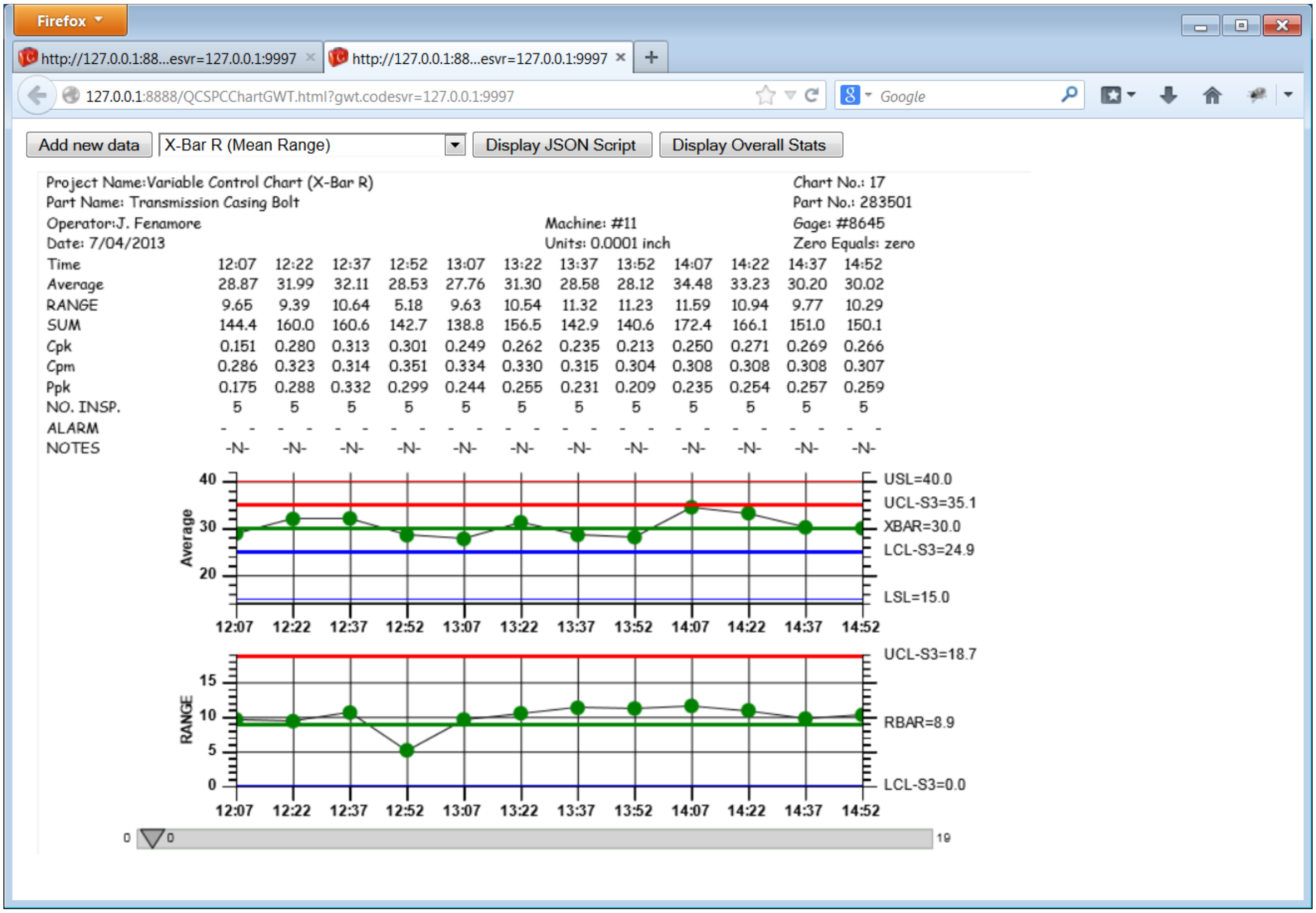
```
        "PrimaryChartSetup": {
            "XAxisLabels": {
                "Rotation": 90
            },
```

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.

```
"ChartPositioning": {
   "InterGraphMargin":0.1,
   "GraphBottomPos":0.85
},
```

Batch Control Chart X-Axis Time Stamp Labeling

Batch X-Bar R Chart using time stamp labeling of the x-axis



Firefox
http://127.0.0.1:88...esvr=127.0.0.1:9997
http://127.0.0.1:88...esvr=127.0.0.1:9997
127.0.0.1:8888/QCSPCChartGWT.html?gwt.codesvr=127.0.0.1:9997
Google
Add new data
X-Bar R (Mean Range)
Display JSON Script
Display Overall Stats

Project Name:Variable Control Chart (X-Bar R)   Chart No.: 17
Part Name: Transmission Casing Bolt   Part No.: 283501
Operator:J. Fenamore   Machine: #11   Gage: #8645
Date: 7/04/2013   Units: 0.0001 inch   Zero Equals: zero

Time	12:07	12:22	12:37	12:52	13:07	13:22	13:37	13:52	14:07	14:22	14:37	14:52
Average	28.87	31.99	32.11	28.53	27.76	31.30	28.58	28.12	34.48	33.23	30.20	30.02
RANGE	9.65	9.39	10.64	5.18	9.63	10.54	11.32	11.23	11.59	10.94	9.77	10.29
SUM	144.4	160.0	160.6	142.7	138.8	156.5	142.9	140.6	172.4	166.1	151.0	150.1
Cpk	0.151	0.280	0.313	0.301	0.249	0.262	0.235	0.213	0.250	0.271	0.269	0.266
Cpm	0.286	0.323	0.314	0.351	0.334	0.330	0.315	0.304	0.308	0.308	0.308	0.307
Ppk	0.175	0.288	0.332	0.299	0.244	0.255	0.231	0.209	0.235	0.254	0.257	0.259
NO. INSP.	5	5	5	5	5	5	5	5	5	5	5	5
ALARM	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -
NOTES	-N-	-N-	-N-	-N-	-N-	-N-	-N-	-N-	-N-	-N-	-N-	-N-

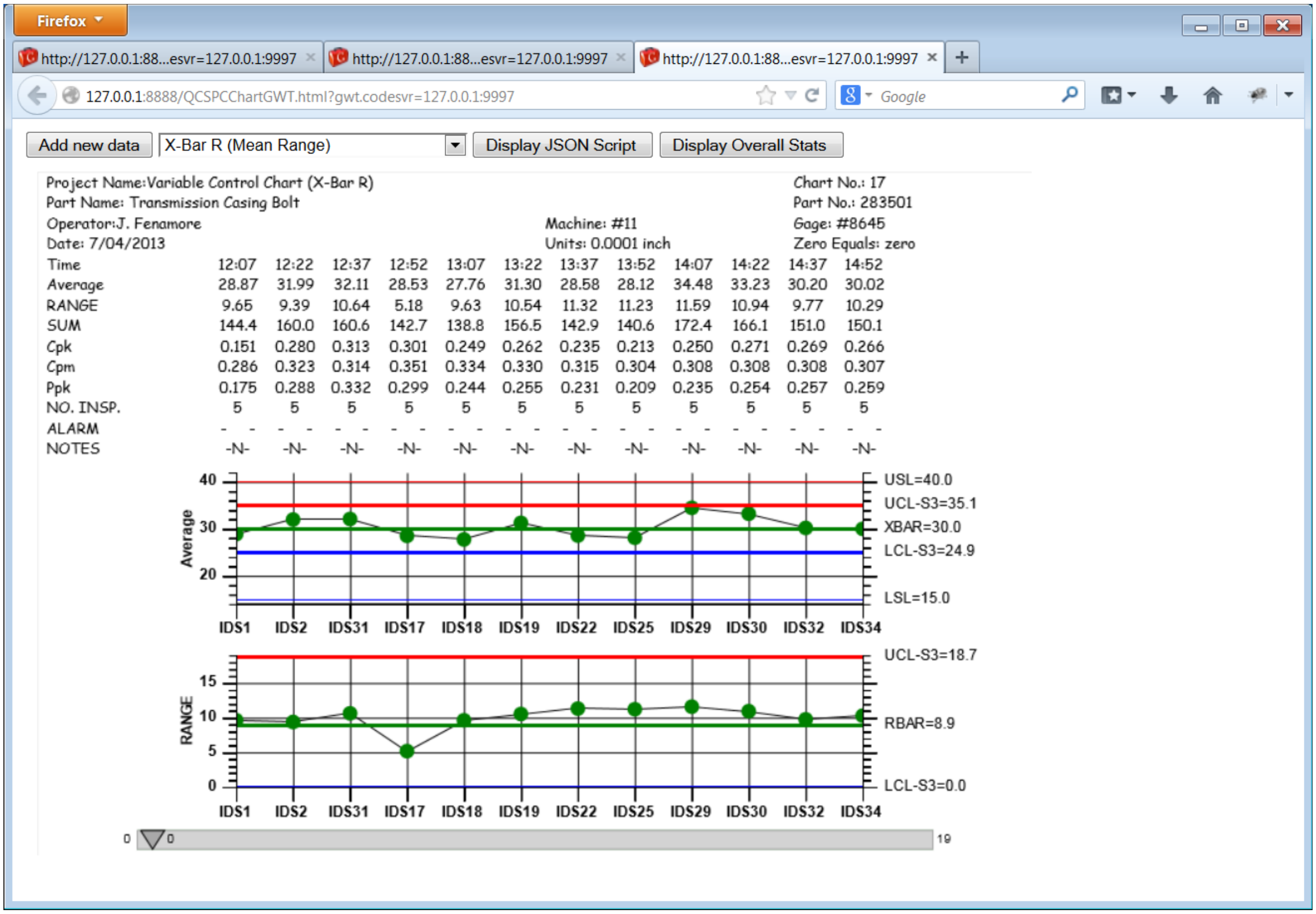
Set the x-axis labeling mode using the PrimaryChartSetup.XAxisLabels.AxisLabelMode property, setting it AXIS_LABEL_MODE_TIME.

```
"PrimaryChartSetup": {
        "XAxisLabels": {
             "AxisLabelMode": "AXIS_LABEL_MODE_TIME"
          },
```

See the JSON script chartDefExampleScripts.js BatchXBarRChart for a complete example. Reset the axis labeling mode back to batch number labeling by assigning the AxisLabelMode property to AXIS_LABEL_MODE_DEFAULT.

Batch Control Chart X-Axis User-Defined String Labeling

Batch X-Bar R Chart user-defined string labeling of the x-axis


Firefox
http://127.0.0.1:88...esvr=127.0.0.1:9997
http://127.0.0.1:88...esvr=127.0.0.1:9997
http://127.0.0.1:88...esvr=127.0.0.1:9997
127.0.0.1:8888/QCSPCChartGWT.html?gwt.codesvr=127.0.0.1:9997
Google
Add new data
X-Bar R (Mean Range)
Display JSON Script
Display Overall Stats
40
30
20
Average
USL=40.0
UCL-S3=35.1
XBAR=30.0
LCL-S3=24.9
LSL=15.0
IDS1 IDS2 IDS31 IDS17 IDS18 IDS19 IDS22 IDS25 IDS29 IDS30 IDS32 IDS34
15
10
5
0
RANGE
UCL-S3=18.7
RBAR=8.9
LCL-S3=0.0
IDS1 IDS2 IDS31 IDS17 IDS18 IDS19 IDS22 IDS25 IDS29 IDS30 IDS32 IDS34
0
0
19

Project Name:Variable Control Chart (X-Bar R) Chart No.: 17
Part Name: Transmission Casing Bolt Part No.: 283501
Operator:J. Fenamore Machine: #11 Gage: #8645
Date: 7/04/2013 Units: 0.0001 inch Zero Equals: zero

Time	12:07	12:22	12:37	12:52	13:07	13:22	13:37	13:52	14:07	14:22	14:37	14:52
Average	28.87	31.99	32.11	28.53	27.76	31.30	28.58	28.12	34.48	33.23	30.20	30.02
RANGE	9.65	9.39	10.64	5.18	9.63	10.54	11.32	11.23	11.59	10.94	9.77	10.29
SUM	144.4	160.0	160.6	142.7	138.8	156.5	142.9	140.6	172.4	166.1	151.0	150.1
Cpk	0.151	0.280	0.313	0.301	0.249	0.262	0.235	0.213	0.250	0.271	0.269	0.266
Cpm	0.286	0.323	0.314	0.351	0.334	0.330	0.315	0.304	0.308	0.308	0.308	0.307
Ppk	0.175	0.288	0.332	0.299	0.244	0.255	0.231	0.209	0.235	0.254	0.257	0.259
NO. INSP.	5	5	5	5	5	5	5	5	5	5	5	5
ALARM	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -	- -
NOTES	-N-	-N-	-N-	-N-	-N-	-N-	-N-	-N-	-N-	-N-	-N-	-N-

Set the x-axis labeling mode using the overall charts AxisLabelMode property, setting it AXIS_LABEL_MODE_STRING.

```
"PrimaryChartSetup": {
       "XAxisLabels": {
            "AxisLabelMode": "AXIS_LABEL_MODE_STRING"
 },
```

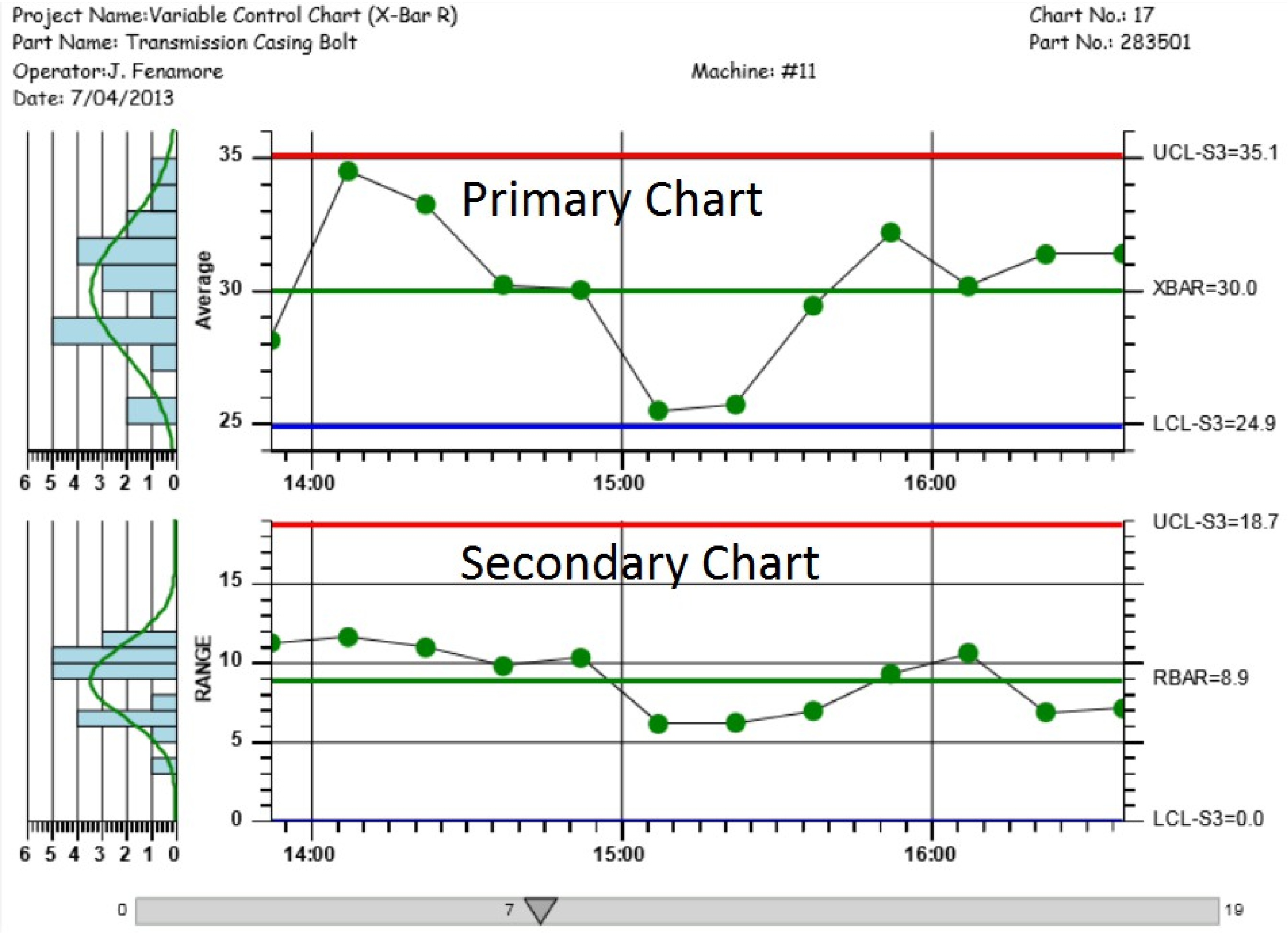
Set the string using the SampleData.SampleIntervalRecords.BatchIDString property.

Reset the axis labeling mode back to batch number labeling by assigning the AxisLabelMode property to AXIS_LABEL_MODE_DEFAULT.

```
"SampleData": {
    "SampleIntervalRecords": [
        {
            "SampleValues": [
                27.53131515148628,
                33.95771604022404,
                24.310097827061817,
                28.282642847792765,
                30.2908518818265
            ],
            "BatchCount": 0,
            "TimeStamp": 1371830829074,
            "BatchIDString": "IDS0",
            "Note": ""
        },
        {
            "SampleValues": [
                27.444285005240214,
                34.38930645615096,
                28.0203674441636,
                33.27153359969366,
                36.8305571558275
            ],
            "BatchCount": 1,
            "TimeStamp": 1371831729074,
            "BatchIDString": "IDS1",
            "Note": ""
        },
        {
            "SampleValues": [
                35.21321620109259,
                32.93940741018088,
                33.66485557976163,
                34.17314124609133,
                24.576683179863725
             ],
   "BatchCount": 2,
       "TimeStamp": 1371832629074,
       "BatchIDString": "IDS2",
       "Note": ""
  },
```

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. Two graph types (MEAN_RANGE_MR_CHART and MULTIVARIABLE_ MEAN_RANGE_MR_CHART) also have a Third Chart to display the Moving Range.



You can modify the default characteristics of each graph using these properties.

```
"PrimaryChartSetup": {
    "FrequencyHistogram": {
        "EnableDisplayFrequencyHistogram": true,
        "PlotBackgroundColor": "WHITE",
        "BarColor": "BLUE"
    },
    "LineMarkerPlot": {
        "LineColor": "GREEN",
        "LineWidth": 2,
        "SymbolColor": "SPRINGGREEN",
        "SymbolFillColor": "SPRINGGREEN",
        "SymbolType": "CIRCLE"
    },
    "PlotBackground": {
        "FillColor": "BROWN",
        "BackgroundMode": "SIMPLECOLORMODE"
    },
    "XAxis": {
        "LineColor": "BLUE",
         "LineWidth": 3
    },
    "YAxisLeft": {
        "LineColor": "GREEN",
        "LineWidth": 3
    },
    "YAxisRight": {
        "LineColor": "RED",
        "LineWidth": 3
    },
```

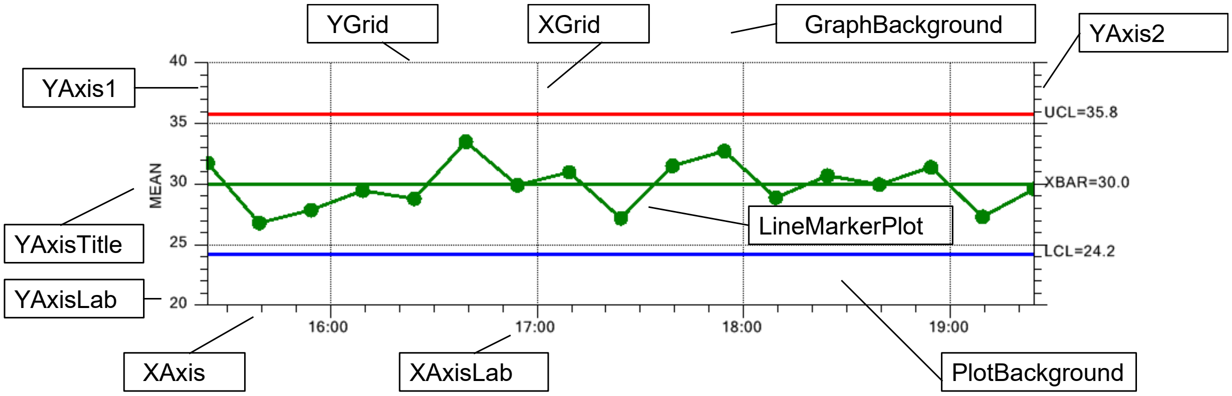
Similarly, for the Secondary chart it would be:

```
"SecondaryChartSetup": {
     "FrequencyHistogram": {
         "EnableDisplayFrequencyHistogram": true,
         "PlotBackgroundColor": "WHITE",
         "BarColor": "BLUE"
     },
     "LineMarkerPlot": {
         "LineColor": "GREEN",
         "LineWidth": 2,
         "SymbolColor": "SPRINGGREEN",
         "SymbolFillColor": "SPRINGGREEN",
         "SymbolType": "CIRCLE"
     },
     "PlotBackground": {
         "FillColor": "BROWN",
         "BackgroundMode": "SIMPLECOLORMODE"
     },
     "XAxis": {
         "LineColor": "BLUE",
         "LineWidth": 3
    },
    "YAxisLeft": {
        "LineColor": "GREEN",
        "LineWidth": 3
    },
    "YAxisRight": {
        "LineColor": "RED",
        "LineWidth": 3
    },
```

The Third chart setup would look similar to the Secondary chart setup.

```
"ThirdChartSetup": {
     "FrequencyHistogram": {
         "EnableDisplayFrequencyHistogram": true,
         "PlotBackgroundColor": "WHITE",
         "BarColor": "BLUE"
     },
     "LineMarkerPlot": {
         "LineColor": "GREEN",
         "LineWidth": 2,
         "SymbolColor": "SPRINGGREEN",
         "SymbolFillColor": "SPRINGGREEN",
         "SymbolType": "CIRCLE"
     },
     "PlotBackground": {
         "FillColor": "BROWN",
         "BackgroundMode": "SIMPLECOLORMODE"
     },
     "XAxis": {
         "LineColor": "BLUE",
         "LineWidth": 3
    },
    "YAxisLeft": {
        "LineColor": "GREEN",
        "LineWidth": 3
    },
    "YAxisRight": {
        "LineColor": "RED",
        "LineWidth": 3
    },
```

The main objects of the graph are labeled in the graph below.



MiscChartDataProperties

The MiscChartDataProperties block contains miscellaneous items global to the chart as a whole.

Multi-Variable Plot Attributes

Plot attributes for multi-variable charts is a feature found only in QCSPCChart+

```
MultiVariableItems
```

```
    "MultiVariableSymbolSize":[14, 14],
     "MultiVariableLineWidth":[1, 1],
     "MultiVariablePlotSymbol":["SQUARE", "SQUARE"],
     "MultiVariableColors":["PURPLE", "BROWN"]
```

The attributes for the main plot are set elsewhere. These items are strictly for plot lines of the added variables. There are all arrays, sized for the number of added variables in your chart.

MultiVariableSymbolSize

An array of integer, sized to the number of added variables, specifying the symbol size (1..64) of the added variable plot.

MultiVariableLineWidth

An array of integer, sized to the number of added variables, specifying the line width (1..7) of the added variable plot. The default value is 1, in order to differentiate the added variables from the main variable, which has a default line thickness of 3.

MultiVariablePlotSymbol

An array of string, sized to the number of added variables, specifying the symbol shape of the added variable plot. The default value is “SQUARE”, in order to differentiate the added variables from the main variable, which has a default symbol shape of “CIRCLE”. Specify the symbol type using one of the SPC shape string constants: "SQUARE", "UPTRIANGLE", "DOWNTRIANGLE", "DIAMOND", "PLUS", "CROSS", "STAR" and "CIRCLE"

MultiVariableColors

An array of string color constants, sized to the number of added variables, specifying the color of the added variable plot. The default colors for added variables are: [ORANGE, DARKCYAN, BURLYWOOD, DARKBLUE, YELLOW], but you can change them to anything you want.

Y-Axes Mapping

Y-axis mapping for multi-variable charts is a feature found only in QCSPCChart+

```
MultiVariableItems
   "YAxisMapping": ["RIGHT_AXIS","RIGHT_AXIS"],
```

Multi-variable charts can aggregate variables of completely different dynamic range. In order to do that, the software uses a second y-axis, with corresponding scale. The right y-axis in a single variable chart is the same as the left y-axis. In the single variable chart the left and right-axis will always be in sync. But if you create one of the multi-variable chart types, the right y-axis can be completely independent of the left y-axis. So you have the option of assigning your added variables (the main variable always references the left y-axis) to either the left y-axis, or the right y-axis. If you assign an added variable to the right y-axis, then the right y-axis becomes independent of the left y-axis. The software will look at all of the variables assigned to the left y-axis and auto-scale it accordingly. It will look at the variables assigned to the right y-axis and auto-scale it independently of the left y-axis.

Legends

Legend chart features found only in QCSPCChart+

```
Legend
   EnableLegend: boolean: true
   LegendStrings: [string, string, ...] :
               ["Item-1", "Item-2", "Item-3"] ,...]
   LegendFont:
      Name: String: "sans-serif"
      Size: integer: 18
      Style: SPC String constant: "Plain"
```

The Multi-variable charts necessitate the use of a legend to differentiate the traces in a chart. Each legend item can be assigned a user-defined string to make identification easier. When the second (right) y-axis is used, the legend items will group under the y-axis the associated trace is referenced against.

EnableLegend: boolean: true

Set to false if you want to hide the legend. You only need to use this parameter if for some reason you want to disable the legend, since if you include a Legend block, the legend is on by default.

LegendStrings: [string, string, ...]

Specify the legend strings. The array elements are [Main variable string, Added Variable #1 string, Added Variable #2 string, …]. You only need to specify as many strings as variables you are displaying in a multi-variable chart. If you have multi-variable chart with two added variables, you will use an array containint three string.

LegendFont

You can modify the font used by the legend using this block.

Name: String: "sans-serif"

Specify a valid font string.

Size: integer: 18

Specify the font size in points.

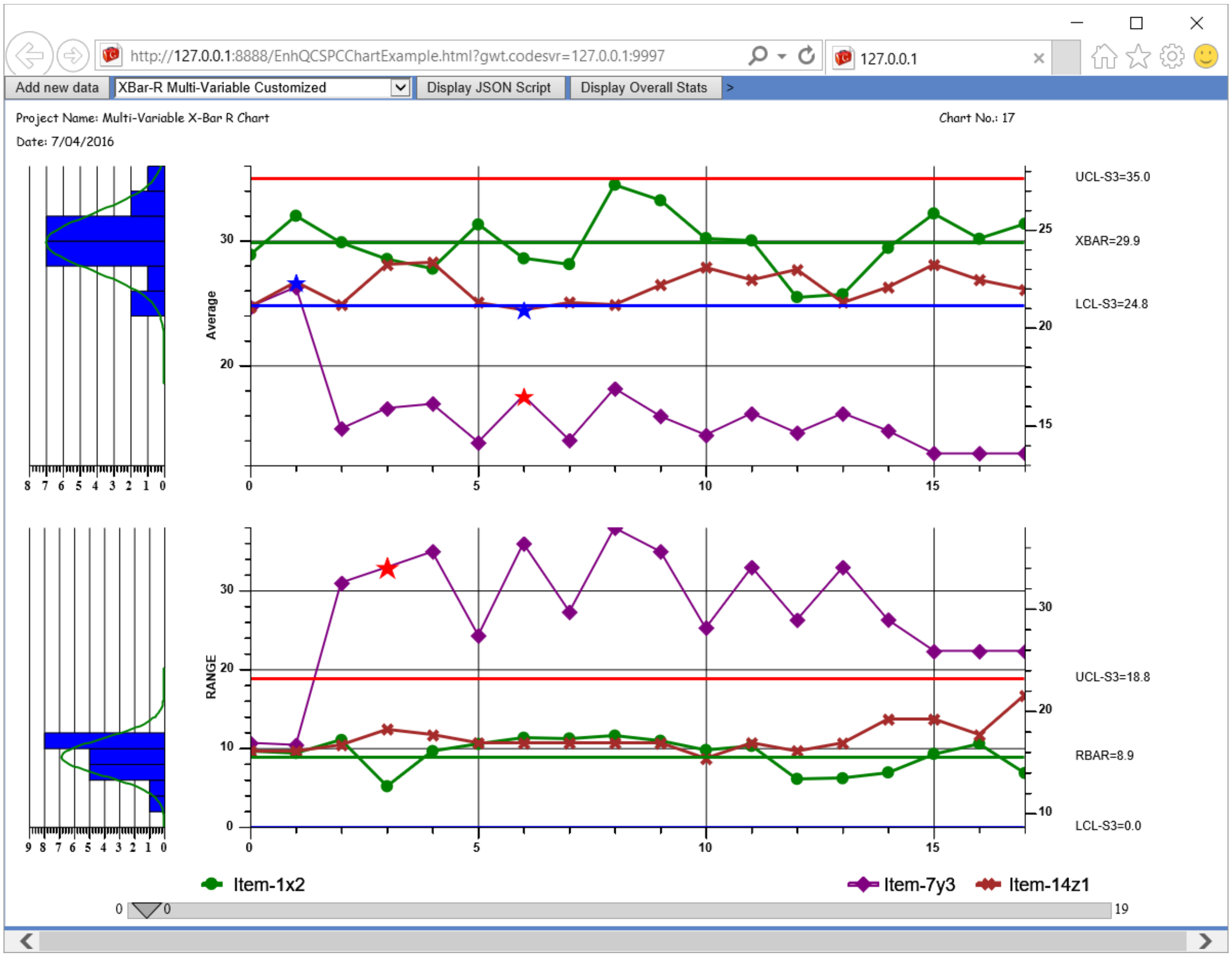
Style: SPC String constant: "Plain”

Use of the style string constants: "Plain", "Normal", "Bold","Italic","Bold Italic".

For example, the following MiscChartDataProperties block is used to setup the legend for a multi-variable Xbar-R chart, with two added variables. The added variables are assigned to the right y-axis. The example below was extracted from the chartdefEnhExamples.js file, script MultiVariableBatchXBarRCustomized.

```
"MiscChartDataProperties":
{
    "MultiVariableItems":
    {
         "YAxisMapping": ["RIGHT_AXIS","RIGHT_AXIS"],
          "MultiVariableSymbolSize":[16, 12],
         "MultiVariableLineWidth":[2, 3],

         "MultiVariablePlotSymbol":["DIAMOND", "CROSS"],
         "MultiVariableColors":["PURPLE", "BROWN"]
    },
     "Legend":
     {
       "LegendStrings": ["Item-1x2", "Item-7y3", "Item-14z1"],
       "LegendFont": {
         "Size": 18,
         "Style": "Plain"
      }
     }
},
```

The resulting chart looks like this.



Project Name: Multi-Variable X-Bar R Chart

Date: 7/04/2016

Chart No.: 17

Note that both of the added variables are mapped to the right y-axis, and the y-axis scale (mapped to the main variable dynamic range) is different than the right y-axis scale, mapped to the dynamic range of the two added variables.

Formulas Used in Calculating +-3 Sigma 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. 2008.

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

$\bar{\bar{X}}$ = X double-bar - Mean of sample subgroup means (also called the grand average)

$\bar{R}$ = R-bar – Mean of sample subgroup ranges

$\tilde{R}$ = R-Median – Median of sample subgroup ranges

S = Sigma – sample standard deviation

$\bar{S}$ = 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 $= \bar{R} + D_4 * \bar{R}$

$$\text{Center line} \quad = \quad \overline{R}$$

$$\text{LCL} \quad = \quad \overline{R} \;-\; D_3 * \overline{R}$$

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

$$\text{UCL} \quad = \quad \overline{\overline{X}} \;+\; A_3 * \overline{S}$$

$$\text{Center line} \quad = \quad \overline{\overline{X}}$$

$$\text{LCL} \quad = \quad \overline{\overline{X}} \;-\; A_3 * \overline{S}$$

Control Limits for the Sigma-Chart

$$\text{UCL} \quad = \quad \overline{B}_4 * \overline{S}$$

$$\text{Center line} \quad = \quad \overline{S}$$

$$\text{LCL} \quad = \quad \overline{B}_3 * \overline{S}$$

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

$$\text{UCL} \quad = \quad \tilde{M} \;+\; \tilde{A}_2 * \tilde{R}$$

$$\text{Center line} \quad = \quad \tilde{M}$$

$$\sim \qquad \sim \qquad \sim$$

$$\text{LCL} = M - A_2 * R$$

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

$$\text{UCL} = \bar{X} + E_2 * \bar{R}$$

$$\text{Center line} = \bar{\bar{X}}$$

$$\text{LCL} = \bar{X} - E_2 * \bar{R}$$

Control Limits for the R-Chart

$$\text{UCL} = \bar{R} + D_4 * \bar{R}$$

$$\text{Center line} = \bar{R}$$

$$\text{LCL} = 0$$

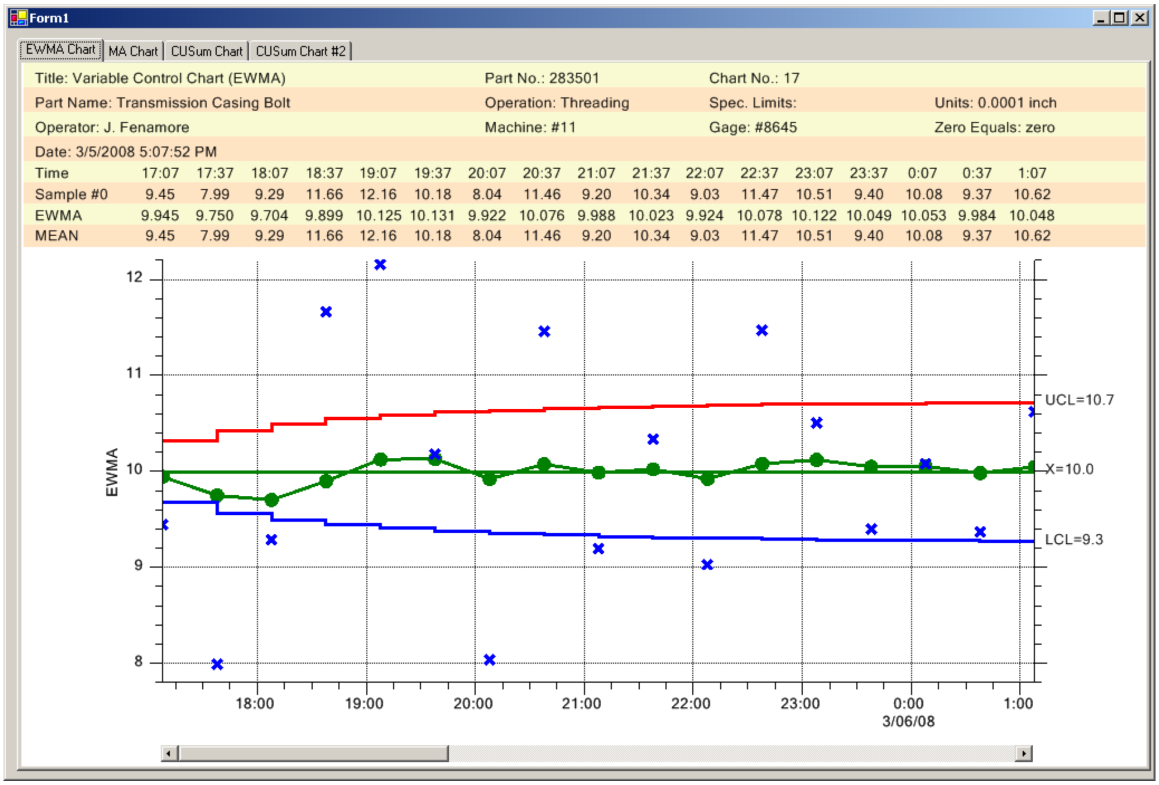
$\overline{R}$ in this case is the average of the moving ranges.

$\overline{X}$ 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.

EWMA Chart – Exponentially Weighted Moving Average

A EWMA chart showing the variable control limits, actual values and EWMA values



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

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

$$UCL \quad = \mu_0 + L \; * \sigma * \; \text{Sqrt}(\; ((\lambda /(2- \lambda)) * (1- (1- \lambda)^{2i}))$$

Center line = μ_0

LCL = μ_0 - L * σ * Sqrt(((λ /(2- λ)) * (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:

UCL = μ_0 + L * σ * Sqrt(λ /(2-λ))

LCL = μ_0 - L * σ * Sqrt(λ /(2-λ))

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.

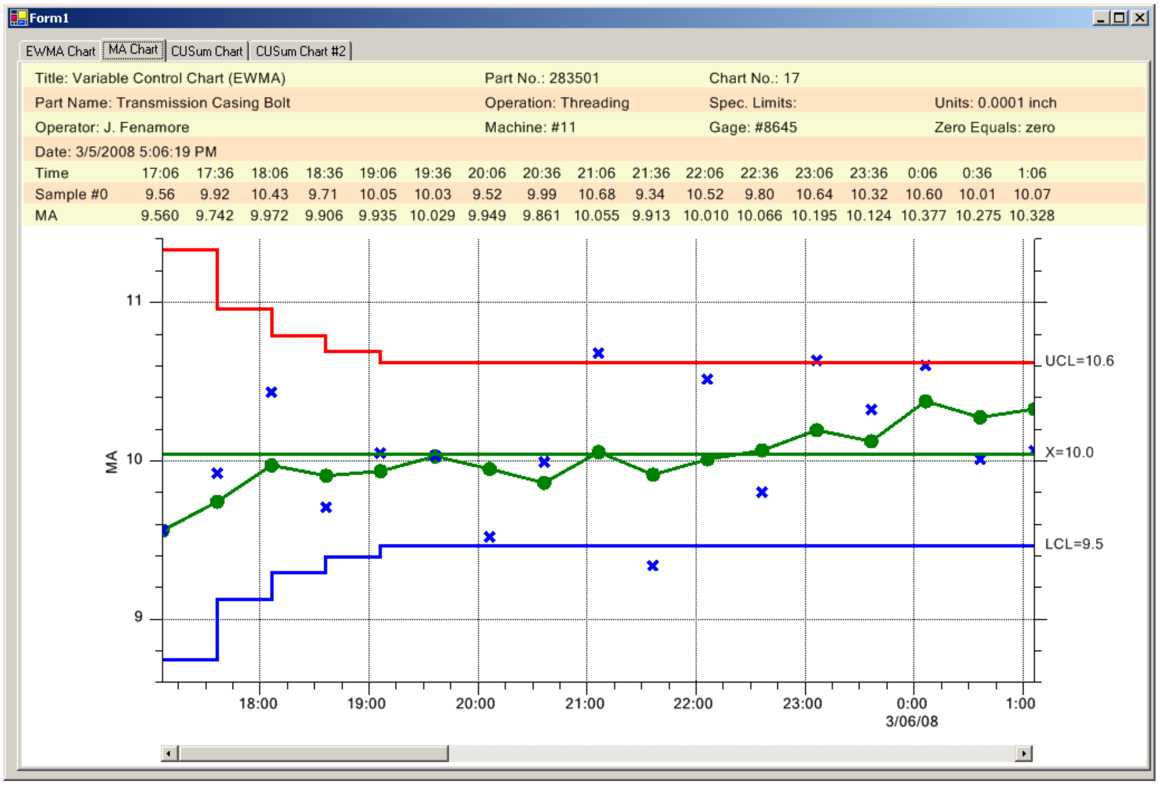
You specify λ and L immediately after the initialization **InitChartProperties** . See the example JSON script chartDefExampleScripts.js BatchEWMA. Specify L using the **MiscChartDataProperties.DefaultControlLimitSigma** property, and λ using the **MiscChartDataProperties.EWMA_Lambda** property. You can optionally set the EWMA starting value (**MiscChartDataProperties.EWMA_StartingValue**), normally set to the process mean value, and whether or not to use the steady-state EWMA control limits (**MiscChartDataProperties.EWMAUseSSLimits**).

Extracted from the BatchEWMA example.

```
"SPCChart": {
    "InitChartProperties": {
        "SPCChartType": "EWMA_CHART",
        "ChartMode": "Batch",
        "NumSamplesPerSubgroup": 1,
        "NumDatapointsInView": 12,
        "TimeIncrementMinutes": 15
    },
    "MiscChartDataProperties": {
        "EWMA_Lambda": 0.25,
        "EWMA_UseSSLimits": false
 },
```

MA Chart – Moving Average

A MA chart showing the variable control limits, actual values and moving average values



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

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

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 $i < N-1$.

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.

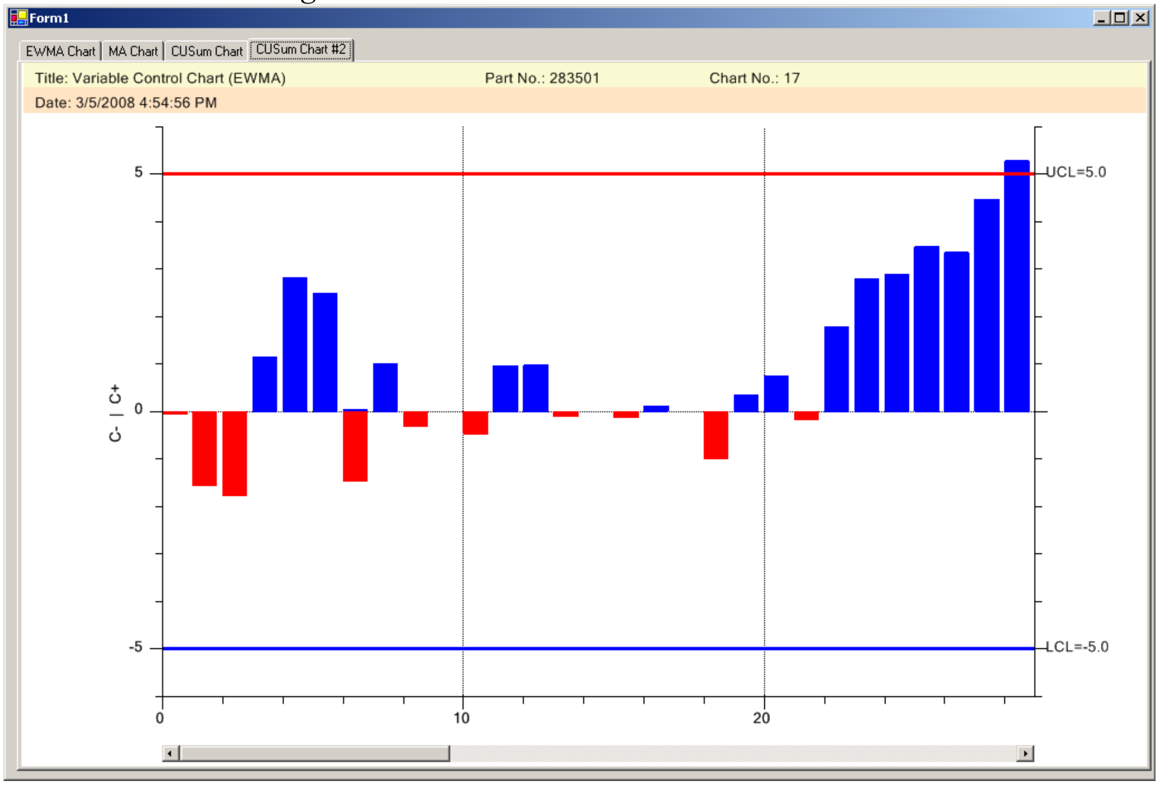
For the values of z_i where $i < N-1$, the weighted average and control limits are calculated using the actual number of samples used in the average, rather than N. This results in expanded values for the control limits for small $i < N-1$.

You specify N, the length of the moving average using the MiscChartDataProperties.MA_w property. Set the process mean and process sigma used in the control limit calculations using the **ProcessMean** and **ProcessSigma** properties.
See the example chartDefExampleScripts.js TimeMA JSON script.

```
 "SPCChart": {
      "InitChartProperties": {
         "SPCChartType": "MA_CHART",
         "ChartMode": "Time",
         "NumSamplesPerSubgroup": 1,
         "NumDatapointsInView": 12,
         "TimeIncrementMinutes": 15
    },

    "MiscChartDataProperties": {
     "MA_W": 7,
     "ProcessMean": 10,
     "ProcessSigma": 1
},
```

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

A batch CuSum chart exceeding the H value



Title: Variable Control Chart (EWMA)

Part No.: 283501

Chart No.: 17

Date: 3/5/2008 4:54:56 PM

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 InitChartProperties block.

Extracted from MiscTimeBasedControlCharts.CUSumChart

```
"SPCChart": {
    "InitChartProperties": {
        "SPCChartType": "TABCUSUM_CHART",
        "ChartMode": "Batch",
        "NumSamplesPerSubgroup": 1,
        "NumDatapointsInView": 12,
        "TimeIncrementMinutes": 15,
        "CuSumKValue": 0.5,
        "CuSumHValue": 5.0,
        "CuSumMeanValue": 10

  },
```

11. SPC Attribute Control Charts

Introduction to SPC Attribute Control Charts

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.

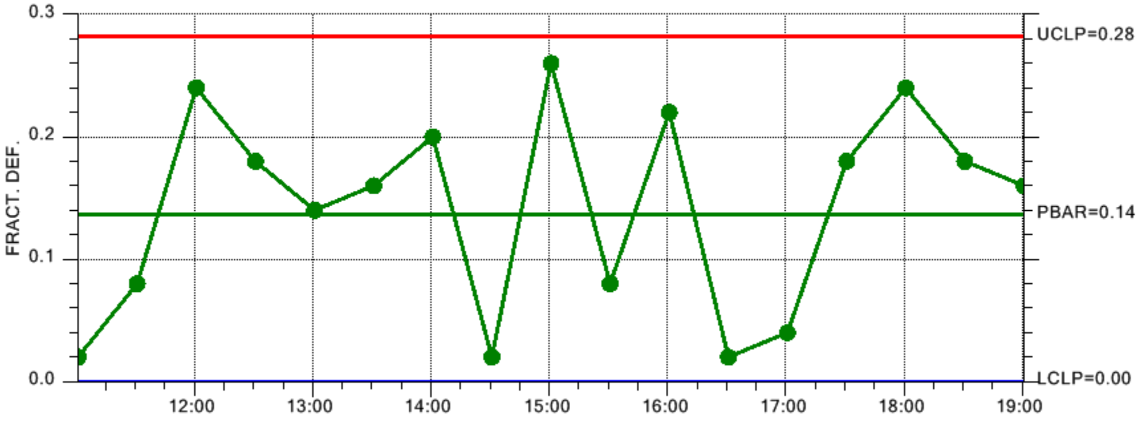
u-Chart – Also known as the Number of Defects per Unit or Number of Non-Conformities per Unit Chart
For a sample subgroup, the number of times a defect occurs is measured and plotted as either a percentage of the total subgroup sample size, or a fraction of the total subgroup sample size. Since the plotted 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

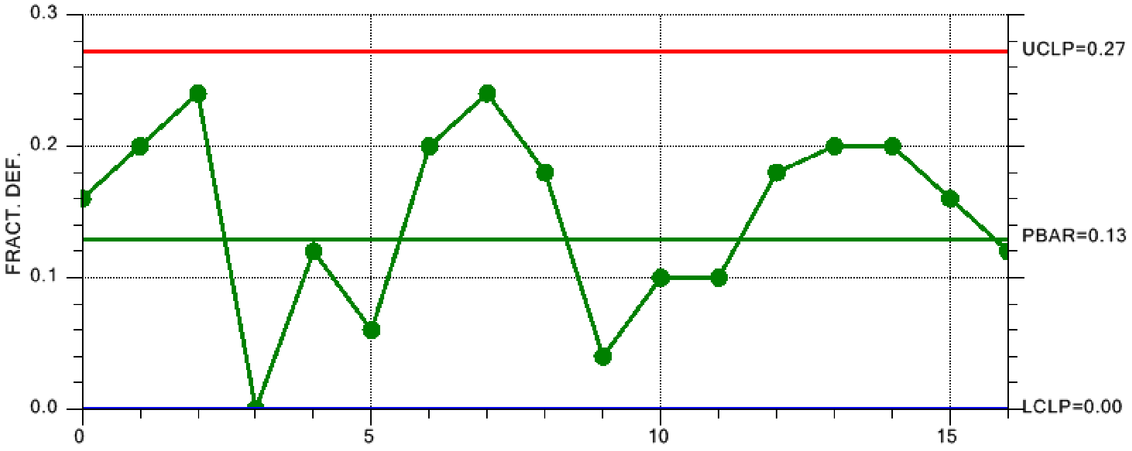
Attribute Control Charts are further categorized as either time- or batch- based. Use time-based SPC charts when data is collected using a subgroup interval corresponding to a specific time interval. Use batch-based SPC charts when the data subgroup interval is a sequential batch number that does not 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.

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.

Creating an Attribute Control Chart

The chart type, and whether or not is is time-based or batch-based, is defined in the SPCChart: InitChartProperties block.

The InitChartProperties block has the following properties.

Parameters

SPCCharType

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.

ChartMode

Specifies if the x-axis is time-based (Time), or batch-base (Batch). Use the string constant string Time or Batch.

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.

TimeIncrementMinutes

Specifies the normal time increment between adjacent subgroup samples. This applies only to the Time ChartMode. Specify a numeric value, no quotes. Can be a double (0.5) to specify a fraction of a minute.

The image below further clarifies how these parameters affect the attribute control chart.

Once the past the initial setup, the chart can be further customized.

Time-based Fraction Defective Parts control chart.

```
"SPCChart": {
       "InitChartProperties": {
          "SPCChartType": "FRACTION_DEFECTIVE_PARTS_CHART",
          "ChartMode": "Time",
          "NumCategories": 5,
          "NumSamplesPerSubgroup": 50,
          "NumDatapointsInView": 12,
          "TimeIncrementMinutes": 15
      },
```

Batch-based Fraction Defective Parts control chart.

```
"SPCChart": {
       "InitChartProperties": {
          "SPCChartType": "FRACTION_DEFECTIVE_PARTS_CHART",
          "ChartMode": "Batch",
          "NumCategories": 5,
          "NumSamplesPerSubgroup": 50,
          "NumDatapointsInView": 12,
          "TimeIncrementMinutes": 15
      },
```

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 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.

```
"MiscChartDataProperties": {
   "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-, np- and DPMO-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 **SPCChart.InitChartProperties** initialization, the first N-1 elements of the *samples* array holds the defect count for each category. The Nth element of the *samples* array holds the total defective parts count.

The comments "//" cannot actually be included in a JSON script.

```
"SampleData": {
 "SampleIntervalRecords": [
 {
    "SampleValues": [
         3,  // Number of defects for defect category #1
         0,  // Number of defects for defect category #2
         4,  // Number of defects for defect category #3
         2,  // Number of defects for defect category #4

         4   // TOTAL number of defective parts in the sample
    ],
    "BatchCount": 0,
    "TimeStamp": 1371830829074,
    "Note": ""
},
{
    "SampleValues": [
         1,
         4,
         0,
         1,

         5   ‘ TOTAL number of defective parts in the sample
     ],
     "BatchCount": 1,
     "TimeStamp": 1371831729074,
     "Note": ""
},
```

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 chartDefExampleScripts.js **TimeNumberDefectiveParts** JSON script.

```
"SampleData": {
     "DataSimulation": {
        "StartCount": 0,
        "Count": 50,
        "Mean": 6.5
 },
```

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

The comments "//" cannot actually be included in a JSON script.

```
"SampleData": {
     "SampleIntervalRecords": [
     {
        "SampleValues": [
             3,  // Number of defects for defect category #1
             0,  // Number of defects for defect category #2
             4,  // Number of defects for defect category #3
             2,  // Number of defects for defect category #4
             3,  // Number of defects for defect category #5
        ],
        "BatchCount": 0,
        "TimeStamp": 1371830829074,
        "Note": ""
    },
    {
        "SampleValues": [
             1,
             4,
             0,
             1,
             2
         ],
         "BatchCount": 1,
         "TimeStamp": 1371831729074,
         "Note": ""
    },
```

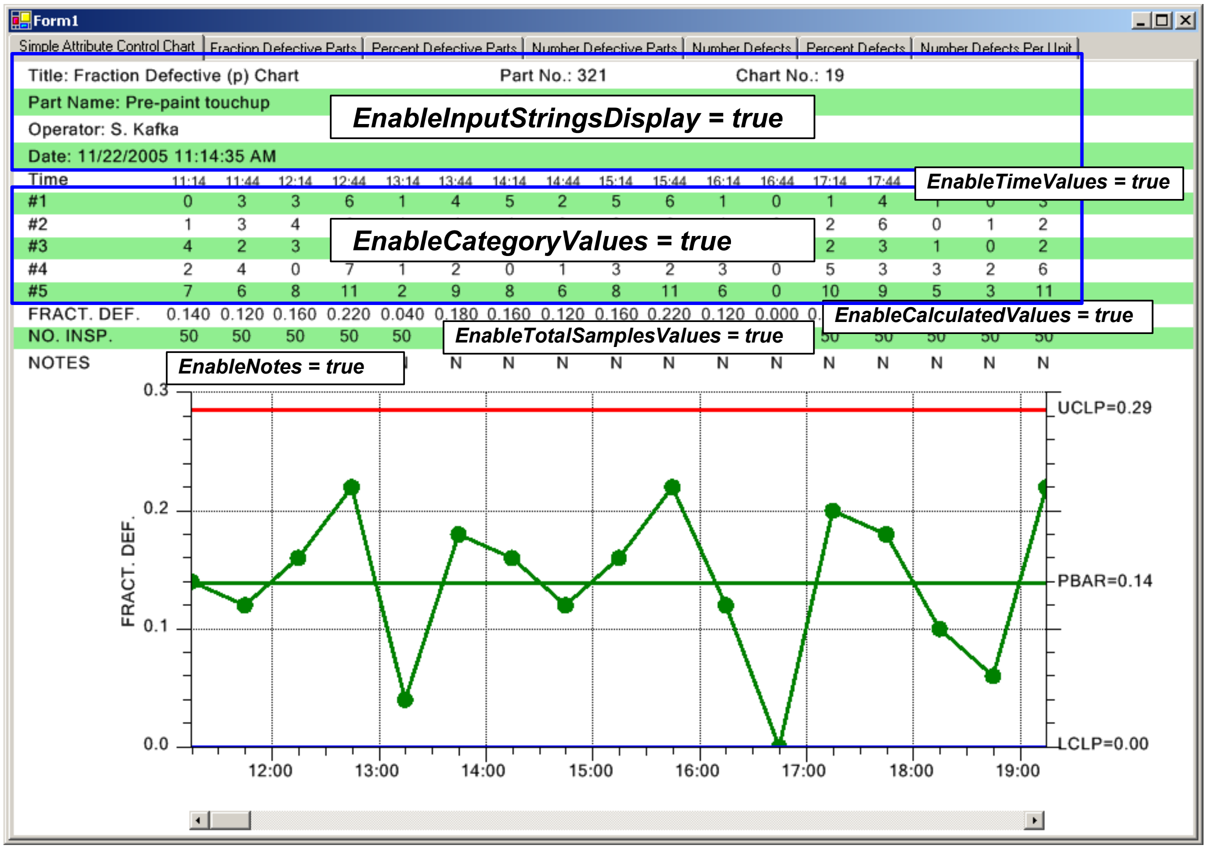
While the table portion of the display can display defect data broken down into categories, only the sum of the defects for a given sample subgroup is used in creating the actual SPC chart.

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.



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 following properties enable sections of the chart header and table:

EnableInputStringsDisplay
EnableCategoryValues
EnableCalculatedValues
EnableTotalSamplesValues
EnableNotes
EnableTimeValues

```
TableSetup
    EnableInputStringsDisplay: boolean: true
    EnableCategoryValues: boolean: true
    EnableCalculatedValues: boolean: true
    EnableTotalSamplesValues: boolean: true
    EnableNotes: boolean: true
    EnableTimeValues: boolean: true
```

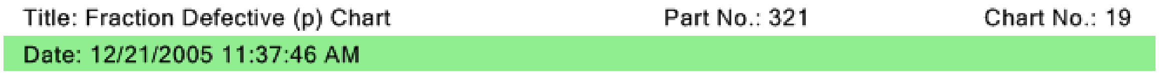
The example code below is extracted from the chartDefExampleScripts.js **TimeFractionDefectiveParts** JSON script.

```
    "SPCChart": {
         "InitChartProperties": {
             "SPCChartType": "FRACTION_DEFECTIVE_PARTS_CHART",
             "ChartMode": "Time",
             "NumCategories": 5,
             "NumSamplesPerSubgroup": 50,
             "NumDatapointsInView": 12,
             "TimeIncrementMinutes": 15
          },
          "ChartPositioning": {
             "GraphStartPosX": 0.125
          },
          "Scrollbar": {
             "EnableScrollBar": true
          },
          "TableSetup": {
             "HeaderStringsLevel": "HEADER_STRINGS_LEVEL3",
             "EnableInputStringsDisplay": true,
             "EnableCategoryValues": false,
             "EnableCalculatedValues": false,
             "EnableTotalSamplesValues": false,
             "EnableNotes": false,
             "EnableTimeValues": true,
             "EnableNotesToolTip": true,
             "TableBackgroundMode": "TABLE_SINGLE_COLOR_BACKGROUND_GRIDCELL",
             "BackgroundColor1": "BEIGE",
             "BackgroundColor2": "LIGHTGOLDENRODYELLOW",
             "TableAlarmEmphasisMode": "ALARM_HIGHLIGHT_BAR",
             "ChartAlarmEmphasisMode": "ALARM_HIGHLIGHT_SYMBOL",
             "ChartData": {
                "Title": "Fraction Defective Parts Chart",
                "PartNumber": "283501",
                "ChartNumber": "17",
                "PartName": "TransmissionCasingBolt",
                "Operation": "Threading",
                "SpecificationLimits": "27.0 to 35.0",
                "Operator": "J.Fenamore",
                "Machine": "#11",
                "Gauge": "#8645",
                "UnitOfMeasure": "0.0001inch",
                "ZeroEquals": "zero",
                "DateString": "7/04/2013",
                "NotesMessage": "ControllimitspreparedMay10",
                "NotesHeader": "NOTES"
             }
        },
```

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, Gauge, 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, Gauge, UnitOfMeasure, ZeroEquals and DateString

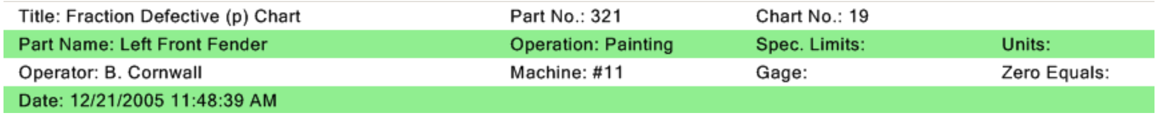
The **TimeFractionDefectiveParts** demonstrates the use of the **HeaderStringsLevel** property. The example below displays a minimum set of header strings (HeaderStringsLevel = HEADER_STRINGS_LEVEL1).



Title: Fraction Defective (p) Chart	Part No.: 321	Chart No.: 19
Date: 12/21/2005 11:37:46 AM		

```
"ChartData": {
   "Title": "Fraction Defective Parts Chart",
   "PartNumber": "283501",
   "ChartNumber": "17",
   "PartName": "TransmissionCasingBolt",
   "Operation": "Threading",
   "SpecificationLimits": "27.0 to 35.0",
    "HeaderStringsLevel":"HEADER_STRINGS_LEVEL1"
```

The example below displays a maximum set of header strings (HeaderStringsLevel = 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			

```
"ChartData": {
   "Title": "Fraction Defective (p) Chart",
   "PartNumber": "283501",
   "ChartNumber":"17",
   "Operator":"B. Cornwall",
   "PartName": "Left Front Fender",
   "Operation": "Painting",
   "SpecificationLimits":"",
   "Machine":"#11",
   "Gauge":"",
   "UnitOfMeasure": "",
   "ZeroEquals":"",
   "HeaderStringsLevel": "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:

```
"StaticProperties": {
    "SPCChartStrings": {
        "TitleHeader": "Project Name:",
        "DefaultMean": "Average",
        "TimeValueRowHeader": "Time"
    }
}
```

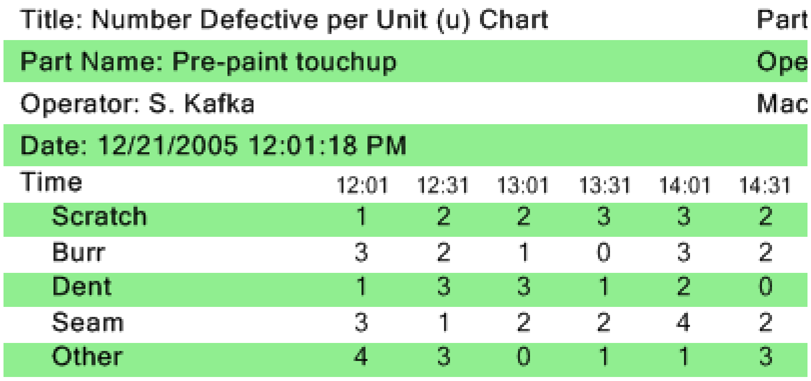
Change other headers using the ChartData properties listed below.

- TitleHeader
- PartNumberHeader
- ChartNumberHeader
- PartNameHeader
- OperationHeader
- OperatorHeader
- MachineHeader
- DateHeader
- SpecificationLimitsHeader
- GaugeHeader
- 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.SampleRowHeaderStrings property. See the chartDefExampleScripts.js TimeNumberDefects example JSON script.



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

```
"ChartData": {
   "Title": "Fraction Defective (p) Chart",
   "PartNumber": "283501",
   "ChartNumber":"17",
   "Operator":"B. Cornwall",
   "PartName": "Left Front Fender",
   "Operation": "Painting",
   "SpecificationLimits":"",
   "Machine":"#11",
   "Gauge":"",
   "UnitOfMeasure": "",
   "ZeroEquals":"",
   "HeaderStringsLevel": "HEADER_STRINGS_LEVEL3",
   "SampleRowHeaderStrings":["    Scratch",
                            "    Burr",
                            "    Dent",
                            "    Seam",
                            "    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:

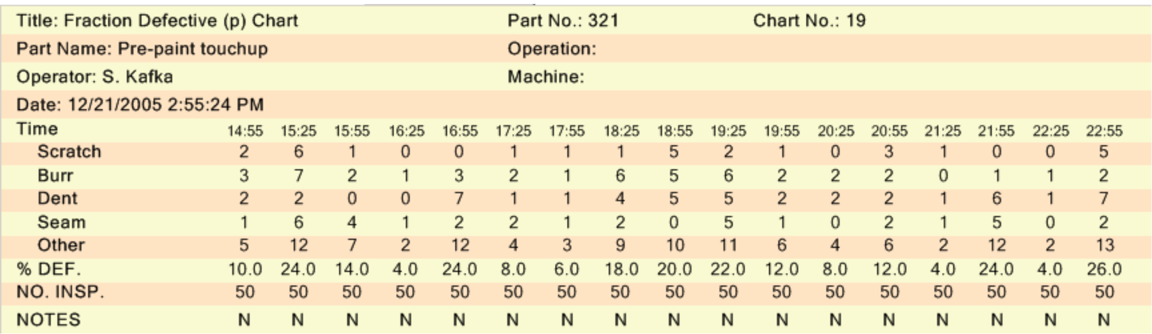
TABLE_NO_COLOR_BACKGROUND Constant specifies that the table does not use a background color.

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 chartDefExampleScripts.js TimePercentDefectiveParts JSON script.



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

Part Name: Pre-paint touchup | Operation:

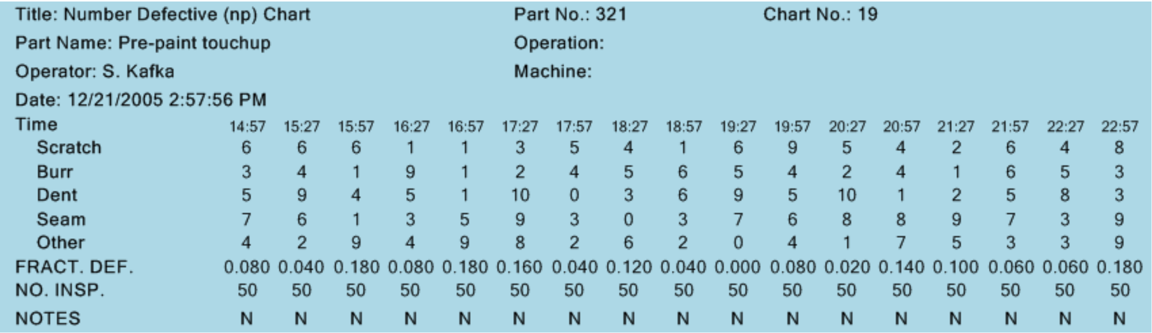
Operator: S. Kafka | Machine:

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

```
"TableSetup": {

        "TableBackgroundMode": "TABLE_STRIPED_COLOR_BACKGROUND",
        "BackgroundColor1": "BEIGE",
        "BackgroundColor2": "LIGHTGOLDENRODYELLOW",
```

Extracted from the chartDefExampleScripts.js TimeNumberDefectiveParts JSON script.



Title: Number Defective (np) Chart | Part No.: 321 | Chart No.: 19

Part Name: Pre-paint touchup | Operation:

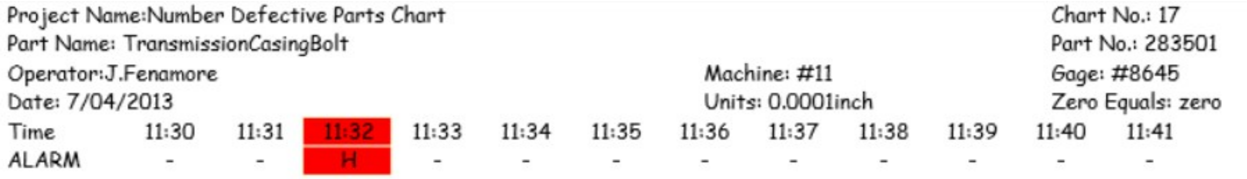
Operator: S. Kafka | Machine:

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

```
"TableSetup": {

        "TableBackgroundMode": "TABLE_SINGLE_COLOR_BACKGROUND",
    "BackgroundColor1": "LIGHTBLUE",
```

Extracted from the chartDefExampleScripts.js BatchNumberDefectiveParts JSON script.



Project Name:Number Defective Parts Chart | Chart No.: 17

Part Name: TransmissionCasingBolt | Part No.: 283501

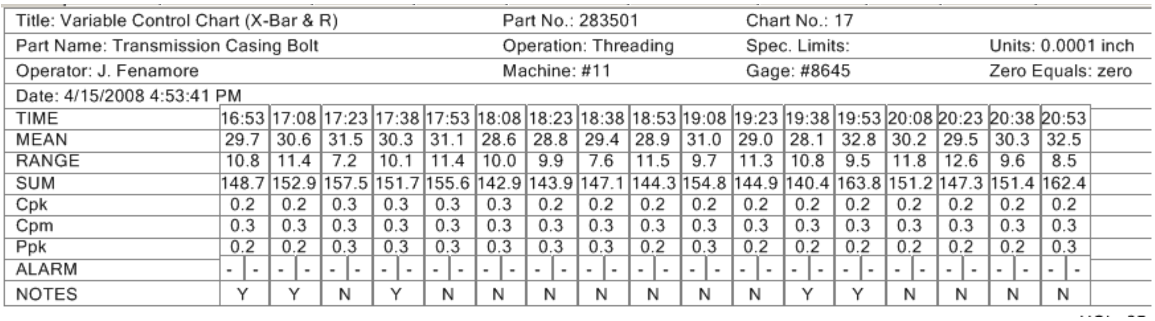
Operator:J.Fenamore | Machine: #11 | Gage: #8645

Date: 7/04/2013 | Units: 0.0001inch | Zero Equals: zero

Time	11:30	11:31	11:32	11:33	11:34	11:35	11:36	11:37	11:38	11:39	11:40	11:41
ALARM	-	-	H	-	-	-	-	-	-	-	-	-

```
"TableSetup": {
                  .
                  .
                  .
                  "TableBackgroundMode": "TABLE_NO_COLOR_BACKGROUND",
```

The TABLE_SINGLE_COLOR_BACKGROUND_GRIDCELL value will give a background color of BackgroundColor1, and a grid outline color of BacgroundColor2.



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

```
"TableSetup": {
                  .
                  .
                  .
                  "TableBackgroundMode": "TABLE_SINGLE_COLOR_BACKGROUND_GRIDCELL",
                  "BackgroundColor1": "WHITE",
                  "BackgroundColor2": "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.

The StaticProperties block has property which is used to set the default table font. Use this if you want to override the default font-family used for both tables and charts, established using the DefaultFontName property. Setting the static properties needs to be done first thing in the first JSON chart definition file you process.

Extracted from the chartDefExampleScripts.js TimeXBarR JSON script.

```
   "StaticProperties":
       {
              "DefaultFontName":  "Arial, sans-serif",
              "DefaultTableFont":
              {      "Name": "'Comic Sans MS', cursive, sans-serif",
                     "Size": 12,
                     "Style": "Plain"
              },
       },
```

The **ChartTable** class has a static property, **StaticProperties.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.

```
DefaultChartFonts
      AxisLabelFont
         Name: String: "sans-serif"
         Size: double: 12
         Style: String: "BOLD"
      AxisTitleFont: standard Name, Size:12, Style: BOLD font properties
      MainTitleFont: standard Name, Size:18, Style: BOLD font properties
```

```
SubHeadFont: standard Name, Size:14, Style: BOLD font properties
ToolTipFont: standard Name, Size:12, Style: PLAIN font properties
AnnotationFont: standard Name, Size:12, Style: PLAIN font properties
ControlLimitLabelFont: standard Name, Size:12, Style: PLAIN font properties
```

Chart Position

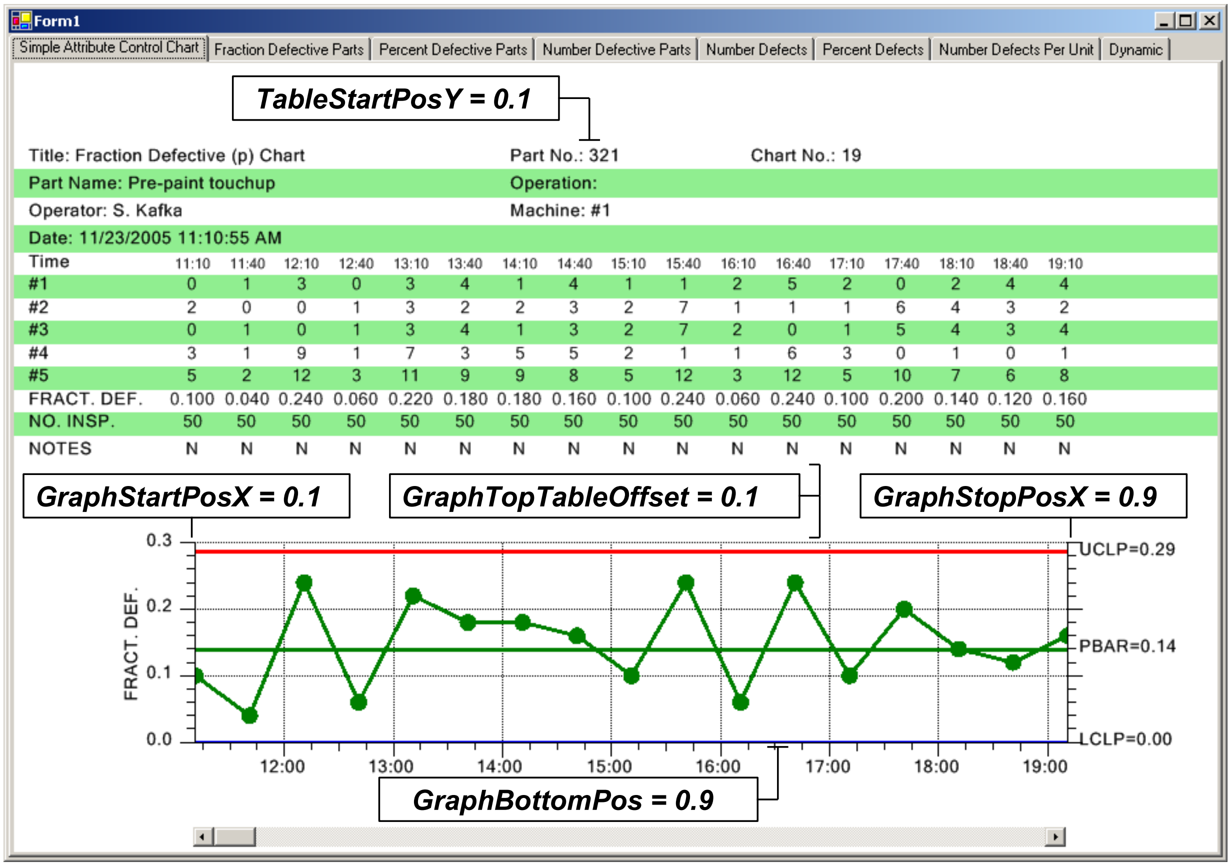
If the SPC chart does not include frequency histograms on the left (they take up about 20% of the available chart width), you may want to adjust the left and right edges of the chart using the **GraphStartPosX** and **GraphStopPlotX** properties to allow for more room in the display of the data. This also affects the table layout, because the table columns must line up with the chart data points.

```
"ChartPositioning": {
       "GraphStartPosX": 0.1,
       "GraphStopPosX": 0.875
},
```

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:

```
"ChartPositioning": {
   "GraphStartPosX": 0.15,
   "GraphStopPosX": 0.8,
   "TableStartPosY": 0.0,
   "GraphTopTableOffset": 0.02,
   "InterGraphMargin": 0.075,
   "GraphBottomPos": 0.90,
   "BottomLabelMargin": 0.0
}
```

The picture below uses different values for these properties in order to emphasize the affect that these properties have on the resulting chart.



TableStartPosY = 0.1

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

GraphStartPosX = 0.1

GraphTopTableOffset = 0.1

GraphStopPosX = 0.9

GraphBottomPos = 0.9

SPC Control Limits

There are several 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. Another d auto-calculates the limits using the algorithms supplied in this software. If you want to set the Target, LCL3 (-3-sigma limit) and UCL3 (3-sigma limit), to explicit values, you can do in the Target, LCL3 and UCL3 blocks of the PrimaryChartSetup | ControlLimits block. Assign the Value property to the limit value you want.

```
"ControlLimits": {
   "Target": {
      "DisplayString": "PBAR",
      "EnableAlarmLine": true,
      "EnableAlarmChecking": true,
      "LimitValue": 0.13,
      "EnableAlarmLineText": true
   },
   "LCL3": {
      "DisplayString": "LCL",
      "EnableAlarmLine": true,
      "EnableAlarmChecking": true,
      "LimitValue": 0,
      "EnableAlarmLineText": false
   },
   "UCL3": {
      "DisplayString": "UCL",
      "EnableAlarmLine": false,
      "EnableAlarmChecking": true,
      "LimitValue": 0.25,
      "EnableAlarmLineText": true
    }
  }
```

If you have more than the standard +- Sigma control limits, it is better if you use the SpecifyControlLimitsUsingMeanAndSigma block to set all the limits.

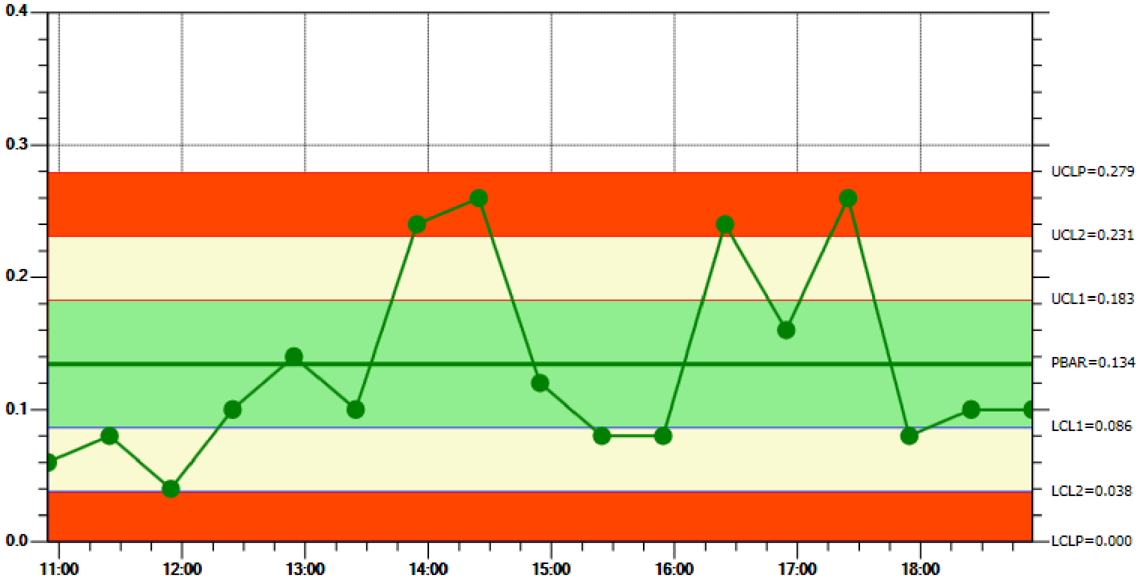
```
SpecifyControlLimitsUsingMeanAndSigma
   Mean: double: 1
   Sigma: double: 1
```

In the example above, where the Target was 0.13, the LCL3 value 0, and the UCL value 0.25, the mean an sigma values would be: Mean = Target = 0.13 and Sigma = (ULC3 – Mean) / 3 =0.08333

```
"SpecifyControlLimitsUsingMeanAndSigma": {
   "Mean":  0.13,
   "Sigma": 0.08333
}
```

There is another property block (**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 MultiLimitAttributeChart 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.

```
"ControlLimits": {
   "ZoneFill": true ,
   "123SigmaControlLimits": {
      "Target":  0.14,
      "LCL3Value": 0,
      "UCL3Value": 0.28,
      "AlarmTest12": true ,
     "EnableAlarmLine": true ,
      "EnableAlarmChecking":  false,
        "EnableAlarmLineText":  true
    },
}
```



Control Limit Fill Option used with +-1, 2 and 3-sigma control limits

The second way to set the control limits is to 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.

AutoCalculateControlLimits takes a boolean array parameter: one for the Primary Chart and one for the Secondary Chart. If you leave out the array parameter, it is the same as the values [true]

```
   "Methods": {
         "AutoCalculateControlLimits": [true,true],
         "AutoScaleYAxes": [true],
         "RebuildUsingCurrentData": true
    }
```

Almost always, a call to AutoCalculateControlLimits will be followed by a call to AutoScaleYAxes to rescale the chart to take into account the new control limits, and RebuildUsingCurrentData to rebuild the graph to show the new limits.

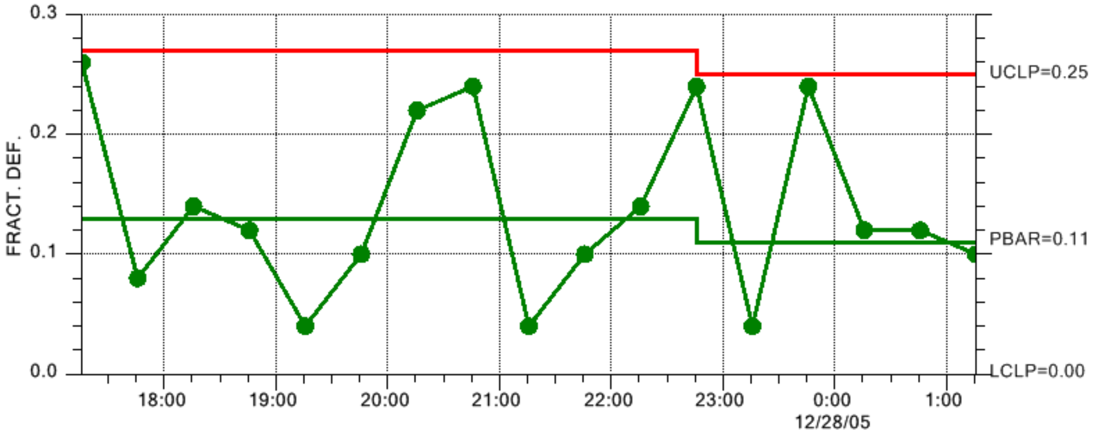
You can add data to the **ChartData** object, auto-calculate the control limits to establish the SPC control limits, and continue to add new data values. Alternatively, you can set the SPC control limits explicitly, as the result of previous runs, using the Target, LCL3, UCL3, 123SigmaControlLimits, or SpecifyControlLimitsUsingMeanAndSigma properties.

Need to exclude records from the control limit calculation? Mark which ones to exclude using the SampleData | ExcludeRecords properties.

```
"ExcludeRecords": [2, 7, 17, 31]
```

Variable SPC Control Limits

There can be situations where the SPC control limit changes in a chart.



There are four ways to enter new SPC limit values. See the example program VariableControlLimits for an example of all three methods. First, you can use the **PrimaryChartSetup.ControlLimits.SetLimits** array property**.**

```
"PrimaryChartSetup": {
   "ControlLimits": {
         "SetLimits": [0.11, 0.0, 0.25]
   }
}
```

This method only works if you are working with the default +-3 Sigma control limits (+ targets) for the Primary chart.

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.

Second, you can use the **Methods.AutoCalculateControlLimits** method. You must have already added a collection of sampled data values to the chart before you can call this method, since the method needs historical values needed in the calculation.

```
   "Methods": {
         "AutoCalculateControlLimits": [true, true]
    }
```

This method works to set the control limits for all sigma-based limits..

Third, you can use the **PrimaryChartSetup.SpecifyControlLimitsUsingMeanAndSigma** property and just set the mean and sigma of the process.

```
"PrimaryChartSetup": {
   "SpecifyControlLimitsUsingMeanAndSigma": {
      "Mean": 0.11,
      "Sigma": 0.0833
   }
}
```

Last, you can enter the SPC control limits with every new sample subgroup record, using SampleData.SampleIntervalRecords.VariableControlLimits array parameter.

```
"SampleData": {
        "SampleIntervalRecords": [
        {
            "SampleValues": [
                4,
                1,
                2,
                3,
                 5
            ],
            "VariableControlLimits": [0.11, 0.0, 0.25],
            "BatchCount": 0,
            "TimeStamp": 1371830829074,
            "Note": ""
        },
```

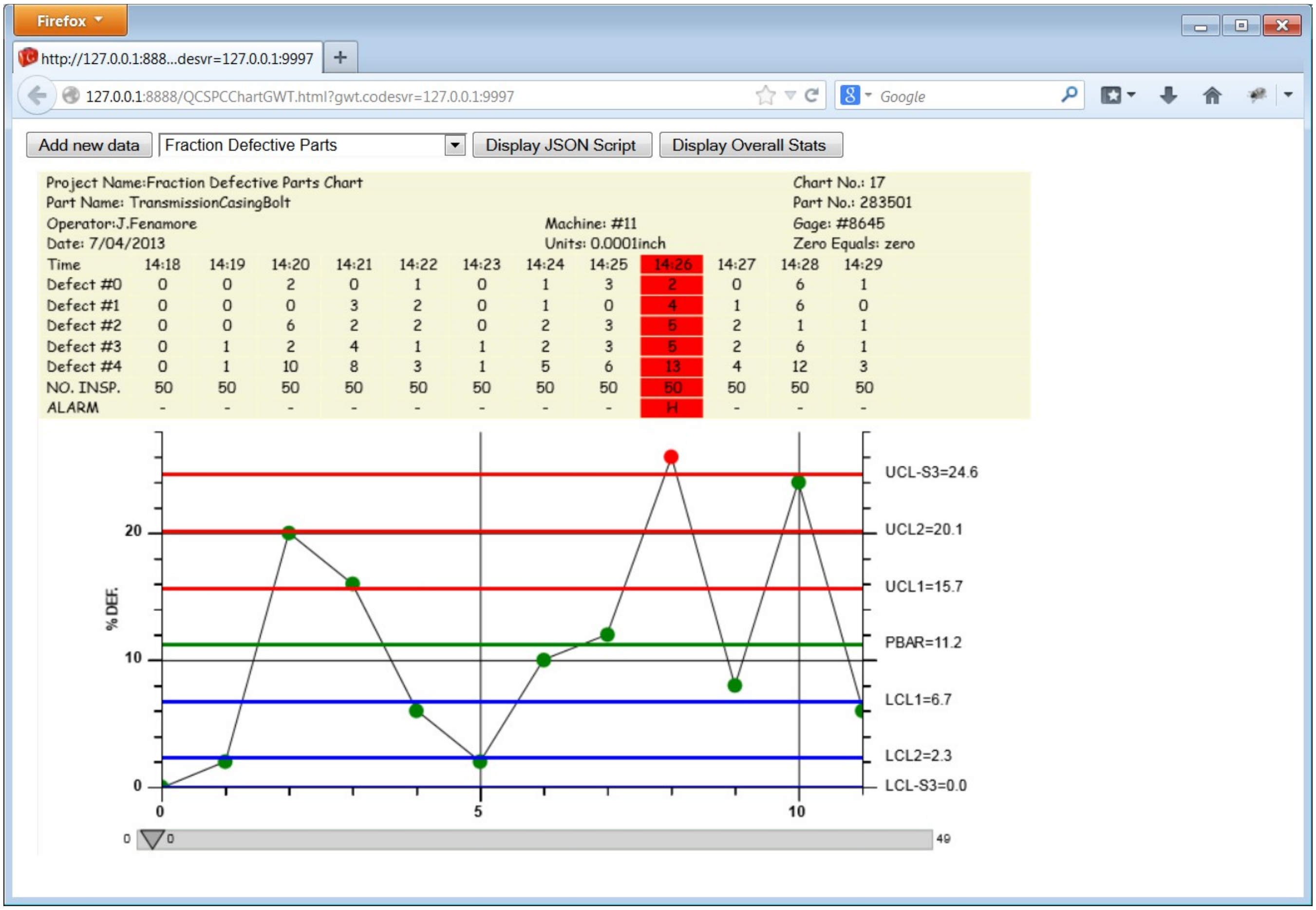
The order of the values in the VariableControlLimits array is [Primary target, Primary LCL3, Primary UCL3]. Other limits are ignored. This method only works if you are working with the default +-3 Sigma control limits (+ targets) for the Primary charts.

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 Batch



Project Name:Fraction Defective Parts Chart
Part Name: TransmissionCasingBolt
Operator:J.Fenamore
Date: 7/04/2013
Machine: #11
Units: 0.0001inch
Chart No.: 17
Part No.: 283501
Gage: #8645
Zero Equals: zero

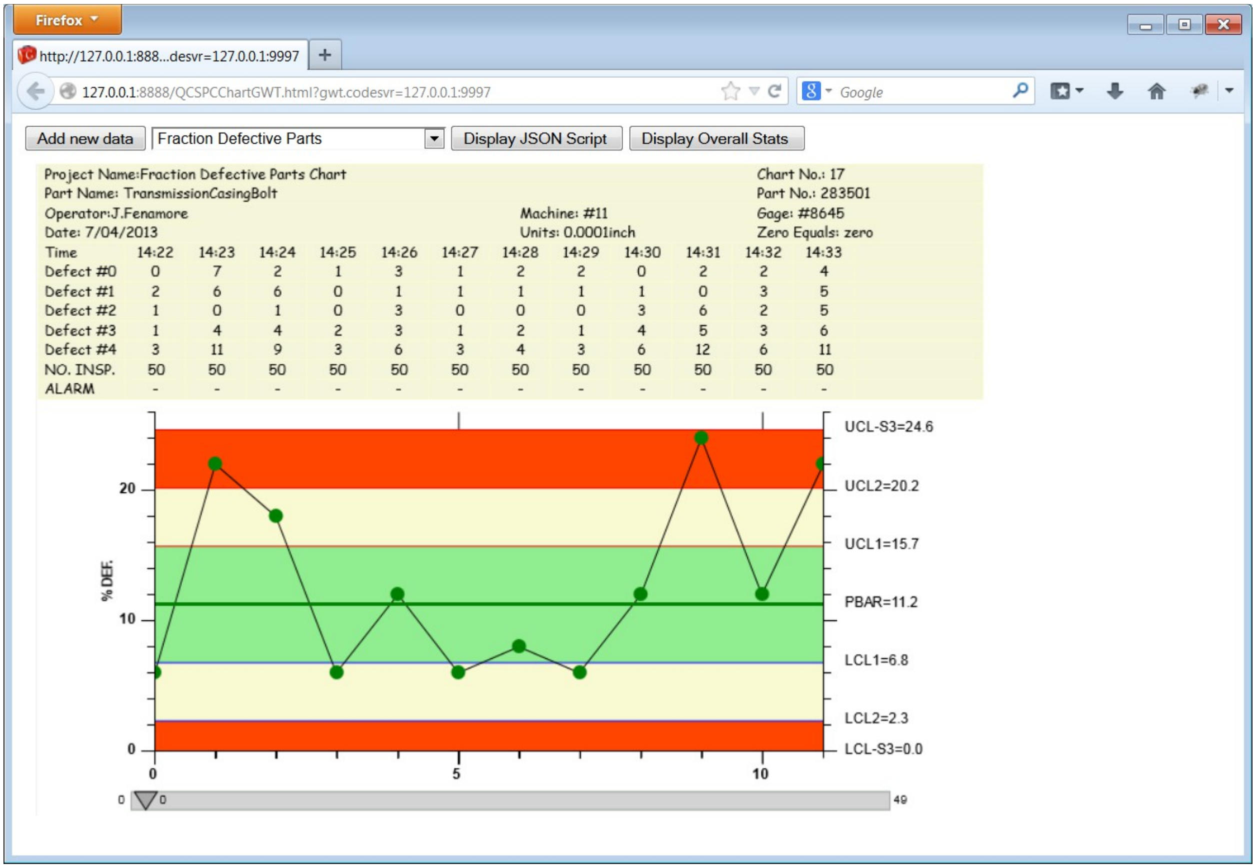
Time	14:18	14:19	14:20	14:21	14:22	14:23	14:24	14:25	14:26	14:27	14:28	14:29
Defect #0	0	0	2	0	1	0	1	3	2	0	6	1
Defect #1	0	0	0	3	2	0	1	0	4	1	6	0
Defect #2	0	0	6	2	2	0	2	3	5	2	1	1
Defect #3	0	1	2	4	1	1	2	3	5	2	6	1
Defect #4	0	1	10	8	3	1	5	6	13	4	12	3
NO. INSP.	50	50	50	50	50	50	50	50	50	50	50	50
ALARM	-	-	-	-	-	-	-	-	H	-	-	-

We added a method (**123SigmaControlLimits**) 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 chartDefExampleScripts.js BatchFractionDefectiveParts JSON script. 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 **123SigmaControlLimits** limits has the option of disabling alarm notification in the case of +-1 and +-2 alarm conditions.

```
"PrimaryChartSetup":   {
                          "ControlLimits": {
                             "ZoneFill": false,
                             "123SigmaControlLimits": {
                                "Target": 0.13,
                                "LCL3Value": 0,
                                "UCL3Value": 0.28,
                                "AlarmTest12": false
                             }
                          }
                       },
```

Control Limit Fill Option used with +-1, 2 and 3-sigma control limits



Project Name:Fraction Defective Parts Chart — Chart No.: 17
Part Name: TransmissionCasingBolt — Part No.: 283501
Operator:J.Fenamore — Machine: #11 — Gage: #8645
Date: 7/04/2013 — Units: 0.0001inch — Zero Equals: zero

Time	14:22	14:23	14:24	14:25	14:26	14:27	14:28	14:29	14:30	14:31	14:32	14:33
Defect #0	0	7	2	1	3	1	2	2	0	2	2	4
Defect #1	2	6	6	0	1	1	1	1	1	0	3	5
Defect #2	1	0	1	0	3	0	0	0	3	6	2	5
Defect #3	1	4	4	2	3	1	2	1	4	5	3	6
Defect #4	3	11	9	3	6	3	4	3	6	12	6	11
NO. INSP.	50	50	50	50	50	50	50	50	50	50	50	50
ALARM	-	-	-	-	-	-	-	-	-	-	-	-

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 **SpecificationLimits** block. 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)

```
"AddControlRules": [
                    {
                       "RuleSet": "BASIC_RULES",
                       "RuleNumber":  3,
                   },
                   {
                       "RuleSet": "BASIC_RULES",
                       "RuleNumber":  4
```

```
                },
                {
                    "RuleSet": "BASIC_RULES",
                    "RuleNumber":  5
                },
                {
                    "RuleSet": "BASIC_RULES",
                    "RuleNumber":  6
                }
            ]
```

Special Note – You can specify a specific value using the LimitValue propety. 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). The software know what sigma level is assigned to a given control rule, and 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 ZoneFill flag under ControlLimits.

```
"PrimaryChartSetup": {
    "ControlLimits": {
        "ZoneFill": true,
        "ZoneColors": ["ORANGERED", "LIGHTGOLDENRODYELLOW", "LIGHTGREEN"]
     }
  }
```

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. In order for the algorithm to work, 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 YAxisLeft bloc of properties.

```
"YAxisLeft":
{
      "MinValue": 0,
      "MaxValue": 2.5
}
```

It is easiest to just call the auto-scale routines after the chart has been initialized with data, and any control limits calculated.

```
"Methods": {
     "AutoCalculateControlLimits": true,
     "AutoScaleYAxes": true,
     "RebuildUsingCurrentData": true
 }
```

Once all of the graph parameters are set, call the method **RebuildUsingCurrentData**.

If, at any future time you change any of the chart properties, you will need to call **RebuildUsingCurrentData** to force a rebuild of the chart, taking into account the current properties. **RebuildUsingCurrentData** 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. example.

Updating Chart Data

The real-time example above demonstrates how the SPC chart data is updated, using the SampleData.SampleIntervalRecords array property.

```
"SPCChart": {

           "SampleData": {
               "SampleIntervalRecords": [
                   {
                       "SampleValues": [
                           5,
                           6,
                           5,
                           6,
                           13
                       ],
                       "BatchCount": 50,
                       "TimeStamp": 1371830829074,
                       "BatchIDString": "IDS50",
                       "Note": ""
                   },
                   {
                       "SampleValues": [
                           2,
                           2,
                           1,
                           2,
                           4
                       ],
                       "BatchCount": 51,
                       "TimeStamp": 1371831729074,
                       "BatchIDString": "IDS51",
                       "Note": ""
                   },
                   {
                       "SampleValues": [
                           0,
                           0,
                           1,
                           1,
                           2
                       ],
                       "BatchCount": 52,
                       "TimeStamp": 1371832629074,
                       "BatchIDString": "IDS52",
                       "Note": ""
                   },
                   {
                       "SampleValues": [
                           2,
                          5,
                           1,
                           2,
                           3
                       ],
                       "BatchCount": 53,
                       "TimeStamp": 1371833529074,
                       "BatchIDString": "IDS53",
                       "Note": ""
                   }

               ]
           },
           "Methods": {
               "AutoCalculateControlLimits": true,
               "AutoScaleYAxes": true,
               "RebuildUsingCurrentData": true
           }
       }
   }
```

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.

Note: Since there is no reliable standard across browsers for time/date data, this value is expressed as the Unix standard of elapsed milliseconds since Thursday, 1 January 1970. The TimeStamp value is used in both time-based SPC Charts, and batch-based SPC Charts, so you must be able to convert from time, to the millisecond equivalent value in order to input the time stamp.

If you want to include a text note in the sample record, just fill out the appropriate Note property of the appropriate SampleIntervalRecords item..

```
"SPCChart": {

            "SampleData": {
                "SampleIntervalRecords": [
                    {
                        "SampleValues": [
                            5,
                            6,
                            5,
                            6,
                            13
                        ],
                        "BatchCount": 50,
                        "TimeStamp": 1371830829074,
                        "BatchIDString": "IDS50",
                        "Note": "Important Note #50"
                    },
                    {
                        "SampleValues": [
                            2,
                            2,
                            1,
                            2,
                            4
                        ],
                        "BatchCount": 51,
                        "TimeStamp": 1371831729074,
                        "BatchIDString": "IDS51",
                        "Note": "Important Note #51"
                    },
                    {
                        "SampleValues": [
                            0,
                            0,
                            1,
                            1,
                            2
                        ],
                        "BatchCount": 52,
                        "TimeStamp": 1371832629074,
                        "BatchIDString": "IDS52",
                        "Note": "Important Note #52"
                    },
                    {
                        "SampleValues": [
                            2,
                            5,
                            1,
                            2,
                            3
                        ],
                        "BatchCount": 53,
                        "TimeStamp": 1371833529074,
                        "BatchIDString": "IDS53",
                        "Note": "Important Note #53"
                    }

                ]
            },
            "Methods": {
                "AutoCalculateControlLimits": true,
                "AutoScaleYAxes": true,
                "RebuildUsingCurrentData": true
            }
        }
    }
```

Scatter Plots of the Actual Sampled Data

- This option is not applicable for attribute control charts.

Enable Chart ScrollBar

Set the **Scrollbar**.**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.

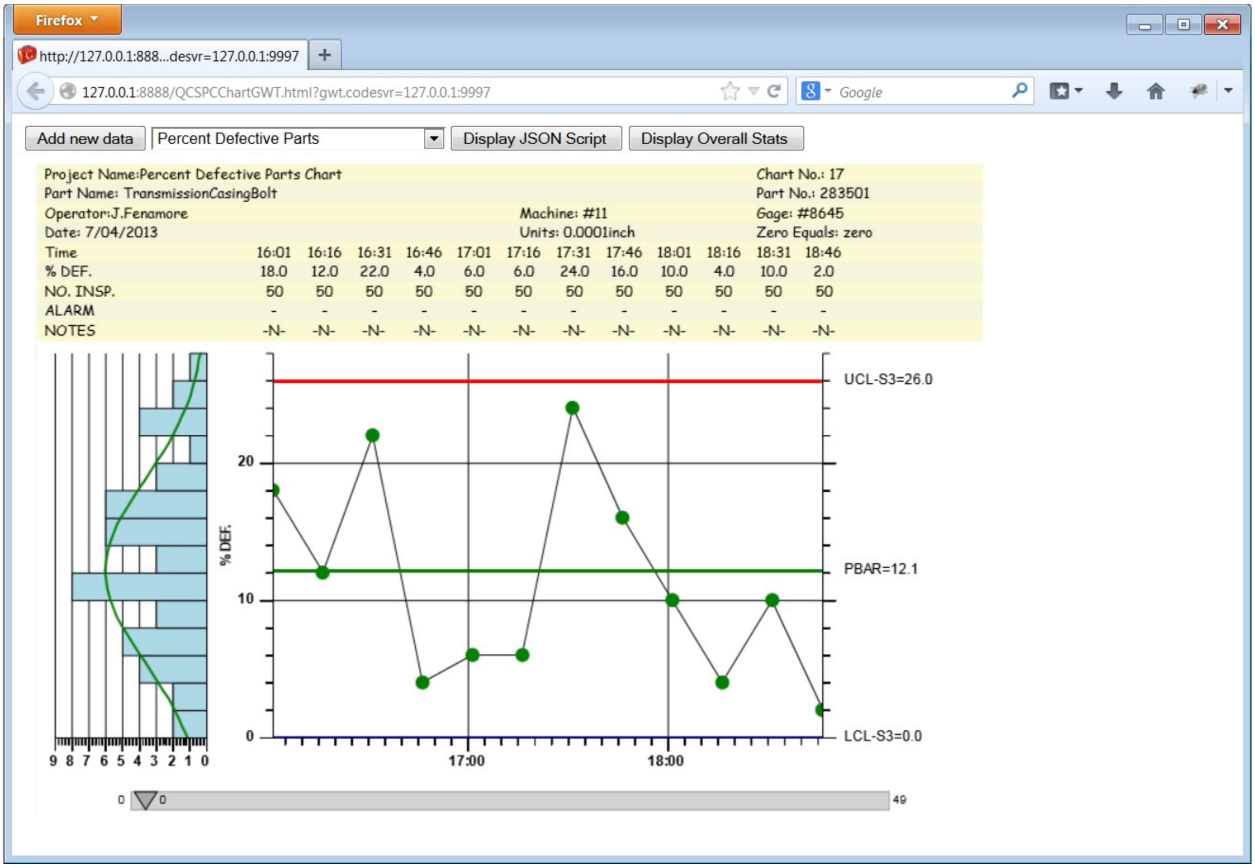
```
"Scrollbar": {
             "EnableScrollBar": true,
             "ScrollbarPosition": "SCROLLBAR_POSITION_MAX"
},
```

You can also set the initial value of the scroll bar so some know value, using the ScrollbarValue property, or you can force the go to the maximum value of the scroll bar after any data updates. That way the most recent data will always be in view. Or, you can specify that after a RebuildUsingCurrentData, which usually increases the scroll bars range of values, that the scrollbar position to show the must recently added data ("SCROLLBAR_POSITION_MAX"), or the oldest data ("SCROLLBAR_POSITION_MIN").

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 **PrimaryChartSetup.FrequencyHistogram. EnableDisplayFrequencyHistogram** property of the chart.

```
"PrimaryChartSetup": {
          "FrequencyHistogram": {
          "EnableDisplayFrequencyHistogram": true
          }
},
```


Firefox
http://127.0.0.1:888...desvr=127.0.0.1:9997
127.0.0.1:8888/QCSPCChartGWT.html?gwt.codesvr=127.0.0.1:9997
Google
Add new data
Percent Defective Parts
Display JSON Script
Display Overall Stats

Project Name:Percent Defective Parts Chart Chart No.: 17
Part Name: TransmissionCasingBolt Part No.: 283501
Operator:J.Fenamore Machine: #11 Gage: #8645
Date: 7/04/2013 Units: 0.0001inch Zero Equals: zero

Time	16:01	16:16	16:31	16:46	17:01	17:16	17:31	17:46	18:01	18:16	18:31	18:46
% DEF.	18.0	12.0	22.0	4.0	6.0	6.0	24.0	16.0	10.0	4.0	10.0	2.0
NO. INSP.	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-

SPC Chart Data and Notes Tooltips

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.

```
Events
   EnableDataToolTip: boolean: true
   EnableJSONDataToolTip: boolean: false
   DataToolTip
       EnableCategoryValues: boolean: false
       EnableProcessCapabilityValues: boolean: false
       EnableCalculatedValues: boolean: false
       EnableNotesString: boolean: false
   EnableNotesToolTip: boolean: true
   NotesToolTip;
      ButtonMask: SPC String constant: "BUTTON1_MASK"
      ToolTipMode: SPC String constant: "MOUSETOGGLE_TOOLTIP"
      NotesReadOnly: boolean: false
```

where

Events.DataToolTip.EnableCategoryValues
Display the category (subgroup sample values) in the data tooltip.

Events.DataToolTip.EnableProcessCapabilityValues
Display the process capability (Cp, Cpl, Cpu, Cpk, Cpm, Pp, Ppl, Ppu and Ppk) statistics currently being calculated for the chart.

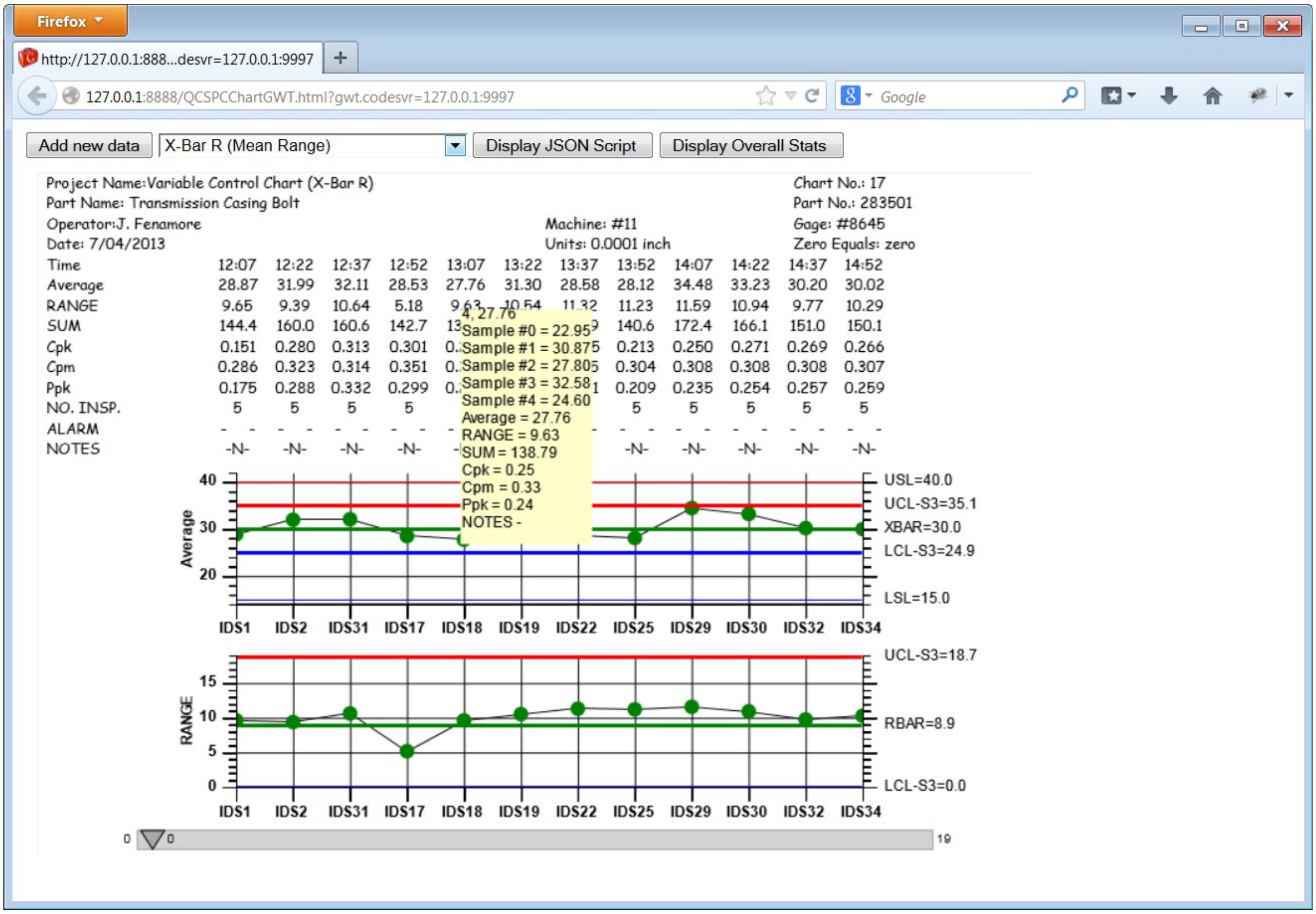
Events.DataToolTip.EnableCalculatedValues
Display the calculated values used in the chart (Mean, range and sum for an Mean-Range chart).

Events.DataToolTip.EnableNotesStrings
Display the current notes string for the sample subgroup.

Extracted from the BatchFractionDefectiveParts example.

```
"Events": {
       "EnableDataToolTip": true,
       "AlarmStateEventEnable": true
 },
```

Data Tooltip


Firefox
http://127.0.0.1:888...desvr=127.0.0.1:9997
127.0.0.1:8888/QCSPCChartGWT.html?gwt.codesvr=127.0.0.1:9997
Google
Add new data
X-Bar R (Mean Range)
Display JSON Script
Display Overall Stats
40
30
20
Average
USL=40.0
UCL-S3=35.1
XBAR=30.0
LCL-S3=24.9
LSL=15.0
IDS1 IDS2 IDS31 IDS17 IDS18 IDS19 IDS22 IDS25 IDS29 IDS30 IDS32 IDS34
15
10
5
0
RANGE
UCL-S3=18.7
RBAR=8.9
LCL-S3=0.0
IDS1 IDS2 IDS31 IDS17 IDS18 IDS19 IDS22 IDS25 IDS29 IDS30 IDS32 IDS34
0
0
19

Project Name:Variable Control Chart (X-Bar R)
Part Name: Transmission Casing Bolt
Operator:J. Fenamore
Date: 7/04/2013

Machine: #11
Units: 0.0001 inch

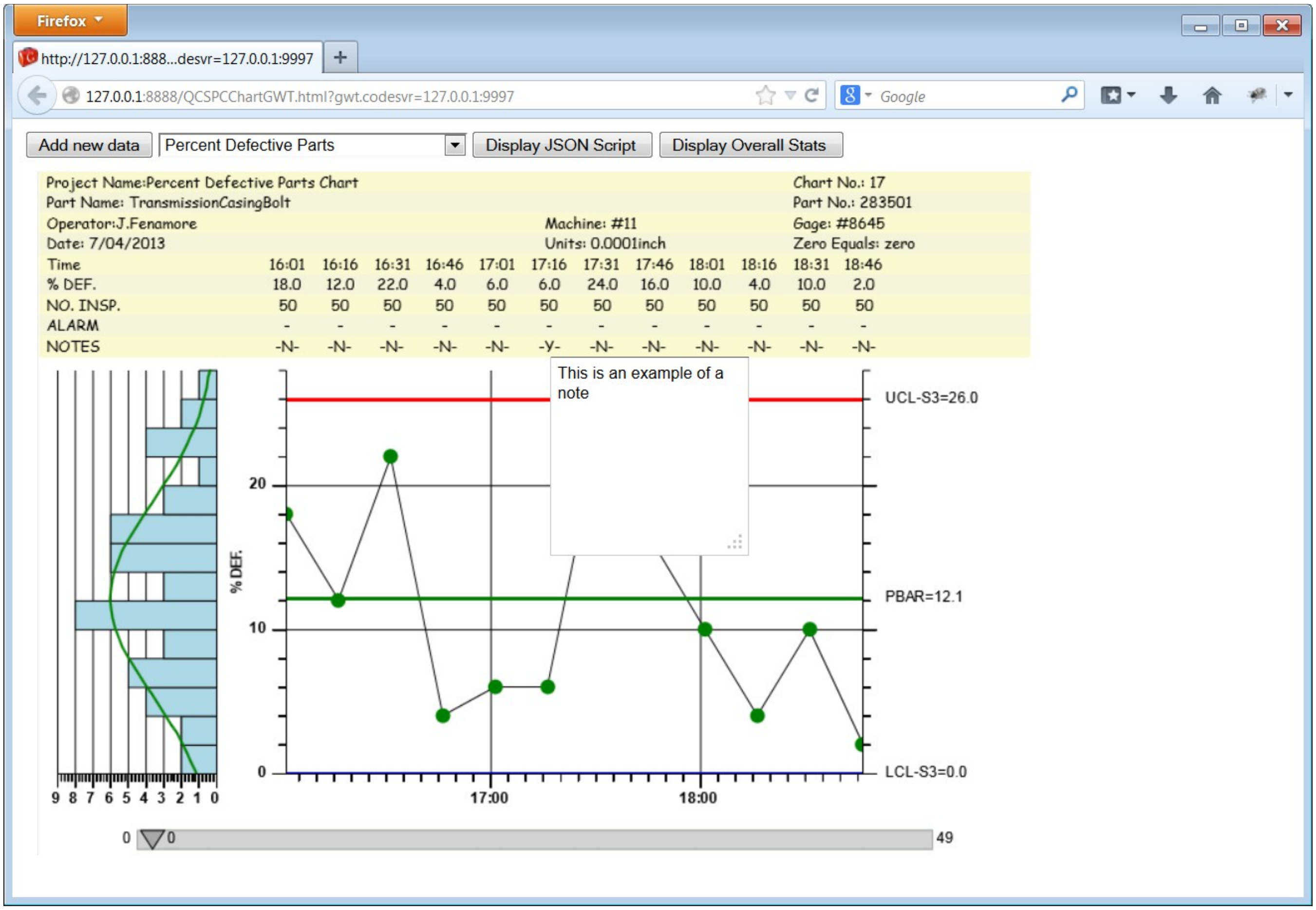
Chart No.: 17
Part No.: 283501
Gage: #8645
Zero Equals: zero

Time	12:07	12:22	12:37	12:52	13:07	13:22	13:37	13:52	14:07	14:22	14:37	14:52
Average	28.87	31.99	32.11	28.53	27.76	31.30	28.58	28.12	34.48	33.23	30.20	30.02
RANGE	9.65	9.39	10.64	5.18	9.63	10.54	11.32	11.23	11.59	10.94	9.77	10.29
SUM	144.4	160.0	160.6	142.7	13		9	140.6	172.4	166.1	151.0	150.1
Cpk	0.151	0.280	0.313	0.301	0.		5	0.213	0.250	0.271	0.269	0.266
Cpm	0.286	0.323	0.314	0.351	0.		5	0.304	0.308	0.308	0.308	0.307
Ppk	0.175	0.288	0.332	0.299	0.		1	0.209	0.235	0.254	0.257	0.259
NO. INSP.	5	5	5	5				5	5	5	5	5
ALARM	- -	- -	- -	- -	-		-	- -	- -	- -	- -	- -
NOTES	-N-	-N-	-N-	-N-	-			-N-	-N-	-N-	-N-	-N-

4, 27.76
Sample #0 = 22.95
Sample #1 = 30.875
Sample #2 = 27.805
Sample #3 = 32.58
Sample #4 = 24.60
Average = 27.76
RANGE = 9.63
SUM = 138.79
Cpk = 0.25
Cpm = 0.33
Ppk = 0.24
NOTES -

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. 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 editable dialog box, immediately above the "Y".

Notes Tooltip



Project Name:Percent Defective Parts Chart　　Chart No.: 17
Part Name: TransmissionCasingBolt　　Part No.: 283501
Operator:J.Fenamore　　Machine: #11　　Gage: #8645
Date: 7/04/2013　　Units: 0.0001inch　　Zero Equals: zero

Time	16:01	16:16	16:31	16:46	17:01	17:16	17:31	17:46	18:01	18:16	18:31	18:46
% DEF.	18.0	12.0	22.0	4.0	6.0	6.0	24.0	16.0	10.0	4.0	10.0	2.0
NO. INSP.	50	50	50	50	50	50	50	50	50	50	50	50
ALARM	-	-	-	-	-	-	-	-	-	-	-	-
NOTES	-N-	-N-	-N-	-N-	-N-	-Y-	-N-	-N-	-N-	-N-	-N-	-N-

This is an example of a note

Both kinds of tooltips are on by default. Turn the tooltips on or off in the program using the **Events.EnableDataToolTip** and **Events.EnableNotesToolTip** properties.

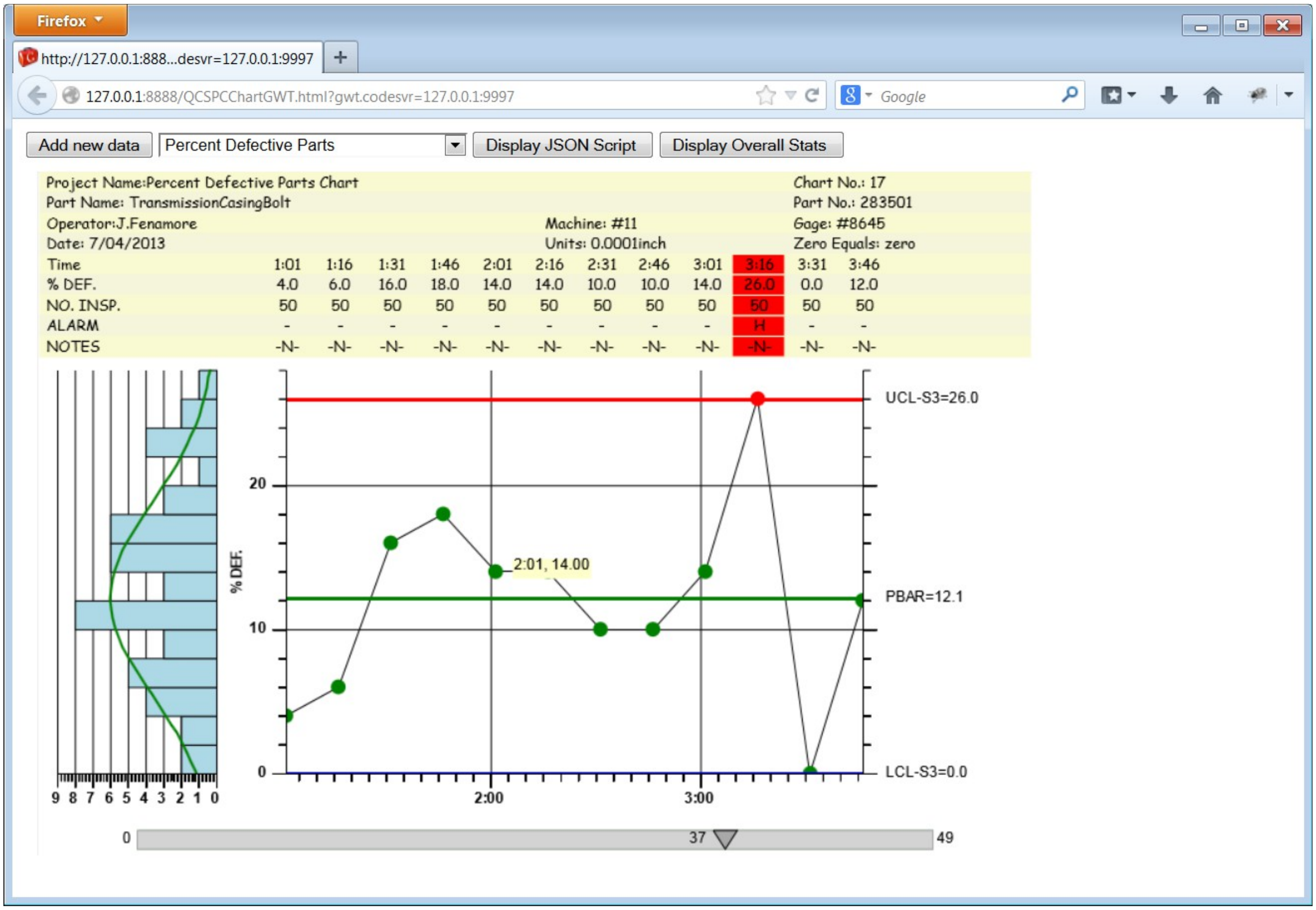
```
"Events": {
    "EnableDataToolTip": true,
    "EnableNotesToolTip": true,
    "EnableJSONDataToolTip": false,
    "AlarmStateEventEnable": false,
    "DataToolTip":
      {
          "EnableCategoryValues": true,
          "EnableProcessCapabilityValues": true,
          "EnableCalculatedValues": true,
          "EnableNotesString": true
      }

 },
```

The notes tooltip has an additional option. In order to make the notes tooltip "editable", the notes edit box, displays on the first click, and goes away on the second click. You can click inside the notes box 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, set the **Events.NotesToolTip.ToolTipMode** property to **MOUSEDOWN_TOOLTIP**, as in the example below.

```
"Events": {
    "EnableDataToolTip": true,
    "EnableNotesToolTip": true,
    "EnableJSONDataToolTip": false,
    "AlarmStateEventEnable": false,
    "DataToolTip":
      {
         "EnableCategoryValues": true,
         "EnableProcessCapabilityValues": true,
         "EnableCalculatedValues": true,
         "EnableNotesString": true
      },
      "NotesToolTip": {
         "ToolTipMode": "MOUSEDOWN_TOOLTIP",
         "NotesReadOnly": true
      }

 },
```

Enable Alarm Highlighting

EnableAlarmStatusValues



Project Name:Percent Defective Parts Chart　　Chart No.: 17
Part Name: TransmissionCasingBolt　　Part No.: 283501
Operator:J.Fenamore　　Machine: #11　　Gage: #8645
Date: 7/04/2013　　Units: 0.0001inch　　Zero Equals: zero

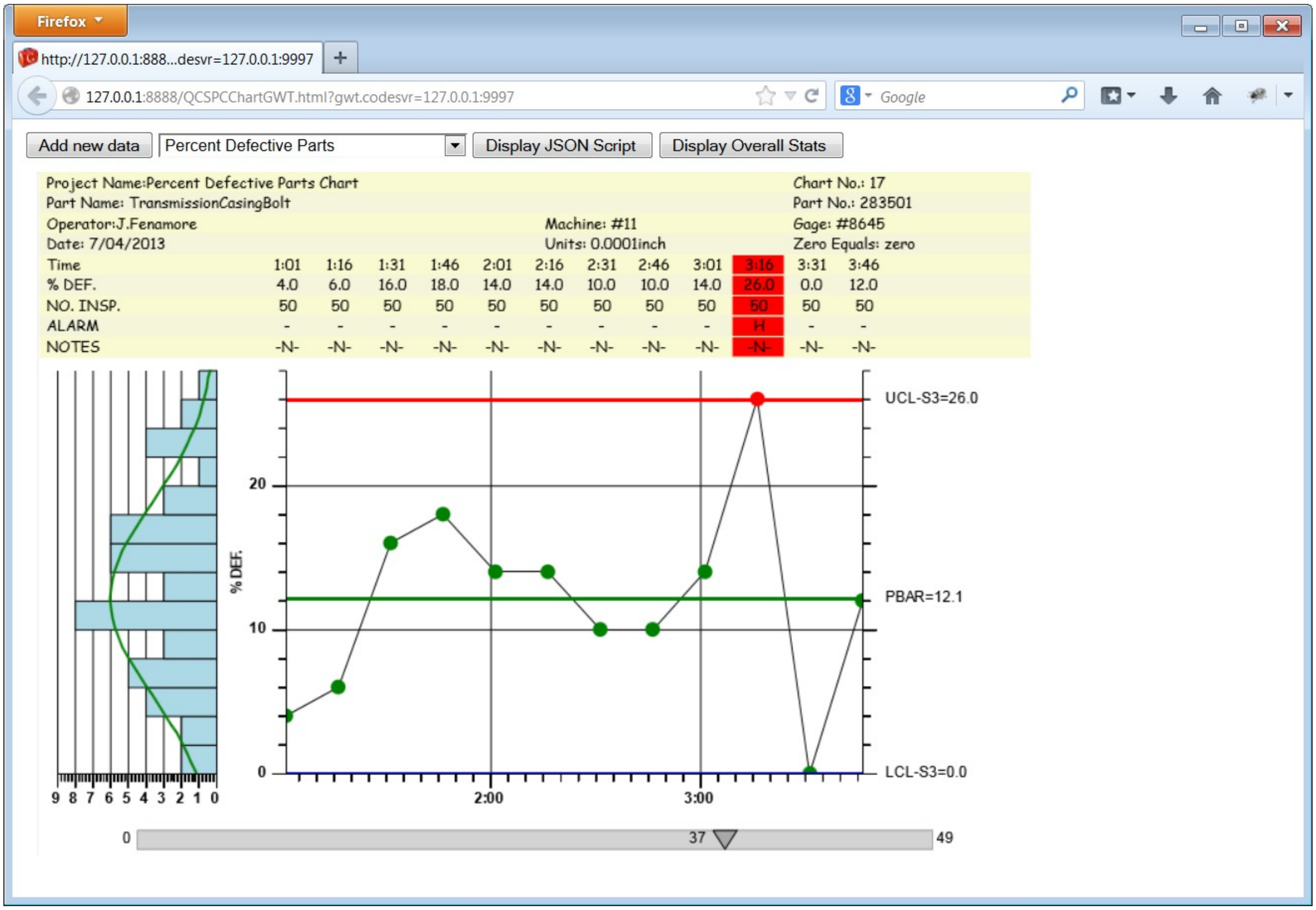
Time	1:01	1:16	1:31	1:46	2:01	2:16	2:31	2:46	3:01	3:16	3:31	3:46
% DEF.	4.0	6.0	16.0	18.0	14.0	14.0	10.0	10.0	14.0	26.0	0.0	12.0
NO. INSP.	50	50	50	50	50	50	50	50	50	50	50	50
ALARM	-	-	-	-	-	-	-	-	-	H	-	-
NOTES	-N-	-N-	-N-	-N-	-N-	-N-	-N-	-N-	-N-	-N-	-N-	-N-

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 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.

```
"Events": {
    "EnableDataToolTip": true,
    "EnableNotesToolTip": true,
    "EnableAlarmStatusValues": true

 },
```

ChartAlarmEmphasisMode



Project Name:Percent Defective Parts Chart Chart No.: 17
Part Name: TransmissionCasingBolt Part No.: 283501
Operator:J.Fenamore Machine: #11 Gage: #8645
Date: 7/04/2013 Units: 0.0001inch Zero Equals: zero

Time	1:01	1:16	1:31	1:46	2:01	2:16	2:31	2:46	3:01	3:16	3:31	3:46
% DEF.	4.0	6.0	16.0	18.0	14.0	14.0	10.0	10.0	14.0	26.0	0.0	12.0
NO. INSP.	50	50	50	50	50	50	50	50	50	50	50	50
ALARM	-	-	-	-	-	-	-	-	-	H	-	-
NOTES	-N-	-N-	-N-	-N-	-N-	-N-	-N-	-N-	-N-	-N-	-N-	-N-

```
"Events": {
    "EnableDataToolTip": true,
    "EnableNotesToolTip": true,
    "EnableAlarmStatusValues": true,
   "ChartAlarmEmphasisMode":" 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 ALARM_NO_HIGHLIGHT_SYMBOL constants.

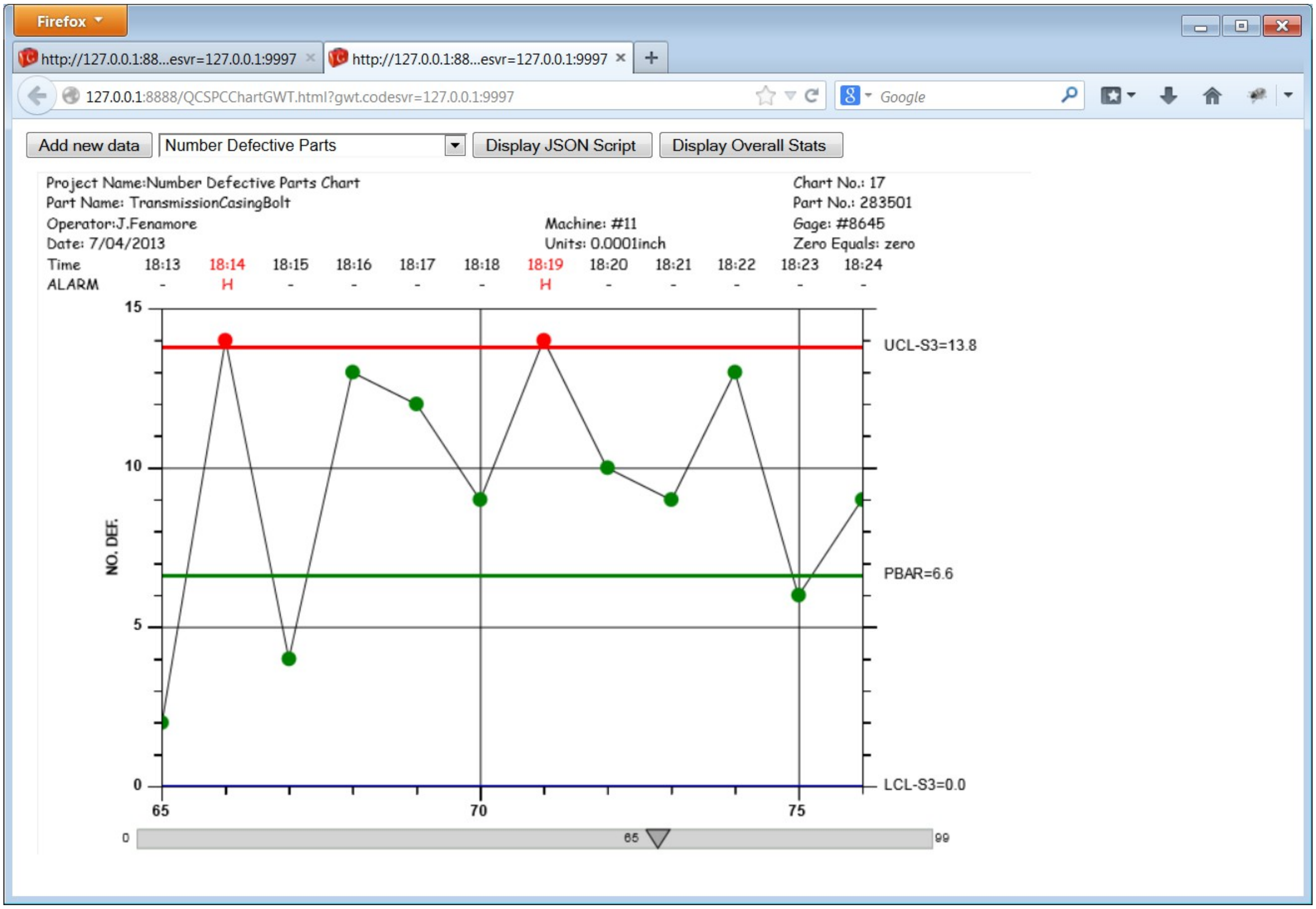
TableAlarmEmphasisMode -

```
      "TableSetup": {

               "TableAlarmEmphasisMode": "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.



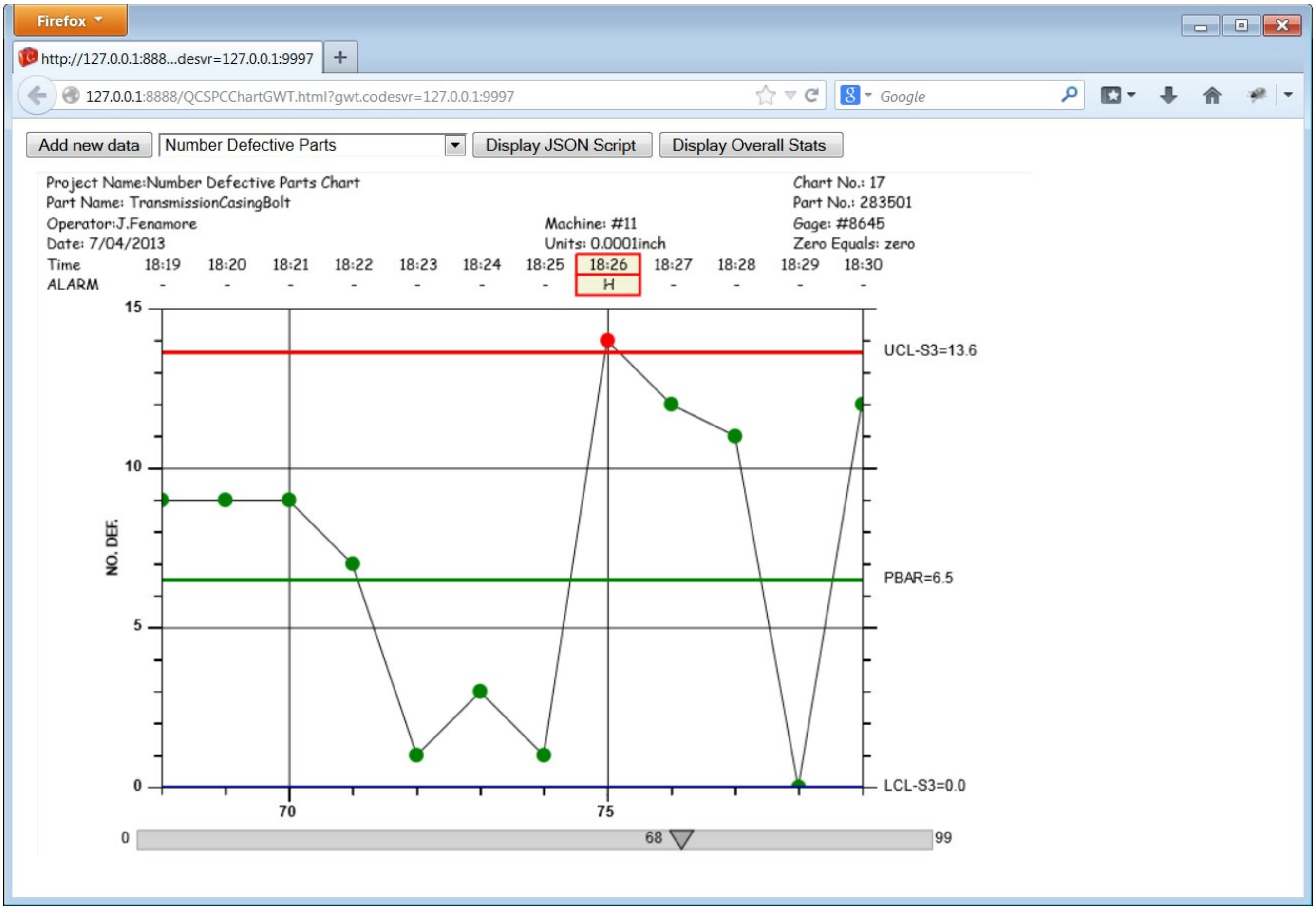
Firefox
http://127.0.0.1:88...esvr=127.0.0.1:9997
http://127.0.0.1:88...esvr=127.0.0.1:9997
127.0.0.1:8888/QCSPCChartGWT.html?gwt.codesvr=127.0.0.1:9997
Google
Add new data
Number Defective Parts
Display JSON Script
Display Overall Stats

Project Name:Number Defective Parts Chart
Part Name: TransmissionCasingBolt
Operator:J.Fenamore
Date: 7/04/2013
Machine: #11
Units: 0.0001inch
Chart No.: 17
Part No.: 283501
Gage: #8645
Zero Equals: zero

Time	18:13	18:14	18:15	18:16	18:17	18:18	18:19	18:20	18:21	18:22	18:23	18:24
ALARM	-	H	-	-	-	-	H	-	-	-	-	-

15
10
5
0
NO. DEF.
UCL-S3=13.8
PBAR=6.6
LCL-S3=0.0
65
70
75
0
65
99

The example above uses the ALARM_HIGHLIGHT_TEXT mode



Project Name:Number Defective Parts Chart
Part Name: TransmissionCasingBolt
Operator:J.Fenamore
Date: 7/04/2013
Machine: #11
Units: 0.0001inch
Chart No.: 17
Part No.: 283501
Gage: #8645
Zero Equals: zero

Time	18:19	18:20	18:21	18:22	18:23	18:24	18:25	18:26	18:27	18:28	18:29	18:30
ALARM	-	-	-	-	-	-	-	H	-	-	-	-

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.

AutoLogAlarmsAsNotes

When an alarm occurs, details of the alarm can be automatically logged as a Notes record. Just set the AutoLogAlarmsAsNotes property to true.

```
"MiscChartDataProperties": {
   "AutoLogAlarmsAsNotes": true
}
```

Creating a Batch-Based Attribute Control Chart

The batch-based and time-based SPC charts 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. The chart type, and whether or not is is time-based or batch-based, is defined in the SPCChart.InitChartProperties block.

The InitChartProperties block has the following properties.

Parameters

SPCCharType

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.

ChartMode

Specifies if the x-axis is time-based (Time), or batch-base (Batch). Use the string constant string Time or Batch.

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.

TimeIncrementMinutes

Specifies the normal time increment between adjacent subgroup samples. This applies only to the Time ChartMode. Specify a numeric value, no quotes. Can be a double (0.5) to specify a fraction of a minute.

```
"SPCChart": {
      "InitChartProperties": {
          "SPCChartType": "FRACTION_DEFECTIVE_PARTS_CHART",
          "ChartMode": "Batch",
          "NumCategories": 5,
          "NumSamplesPerSubgroup": 50,
          "NumDatapointsInView": 12,
          "TimeIncrementMinutes": 15
      },
```

Adding New Sample Records for Batch 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-, np- and DPMO-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 **InitChartProperties** initialization call, the first N-1 elements of the *samples* array holds the defect count for each category. The Nth element of the *samples* array holds the total defective parts count.

The comments "//" cannot actually be included in a JSON script.

```
"SampleData": {
     "SampleIntervalRecords": [
     {
        "SampleValues": [
             3,  // Number of defects for defect category #1
             0,  // Number of defects for defect category #2
             4,  // Number of defects for defect category #3
             2,  // Number of defects for defect category #4

             4   // TOTAL number of defective parts in the sample
        ],
        "BatchCount": 0,
        "TimeStamp": 1371830829074,
        "Note": ""
    },
    {
        "SampleValues": [
             1,
             4,
             0,
             1,

             5   ‘ TOTAL number of defective parts in the sample
         ],
         "BatchCount": 1,
         "TimeStamp": 1371831729074,
         "Note": ""
    },
```

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 chartDefExampleScripts.js BatchNumberDefectiveParts JSON script.

```
"SampleData": {
     "DataSimulation": {
     "StartCount": 0,
     "Count": 50,
     "Mean": 6.5
 },
```

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

The comments “//” cannot actually be included in a JSON script.

```
"SampleData": {
     "SampleIntervalRecords": [
     {
        "SampleValues": [
             3,  // Number of defects for defect category #1
             0,  // Number of defects for defect category #2
             4,  // Number of defects for defect category #3
             2,  // Number of defects for defect category #4
             3,  // Number of defects for defect category #5
        ],
        "BatchCount": 0,
        "TimeStamp": 1371830829074,
        "Note": ""
    },
{
    "SampleValues": [
        1,
        4,
        0,
        1,
        2
     ],
     "BatchCount": 1,
     "TimeStamp": 1371831729074,
     "Note": ""
},
```

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.

Changing the Batch Control Chart X-Axis Labeling Mode

The default mode of the x-axis tick marks of a batch control chart is to label them with the numeric batch number of the sample subgroup. It is also 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** property found in all of the example programs.

```
"SPCChart": {
       "InitChartProperties": {
             "SPCChartType": "FRACTION_DEFECTIVE_PARTS_CHART",
             "ChartMode": "Batch",
             "NumCategories": 5,
             "NumSamplesPerSubgroup": 50,
             "NumDatapointsInView": 12

       },
```

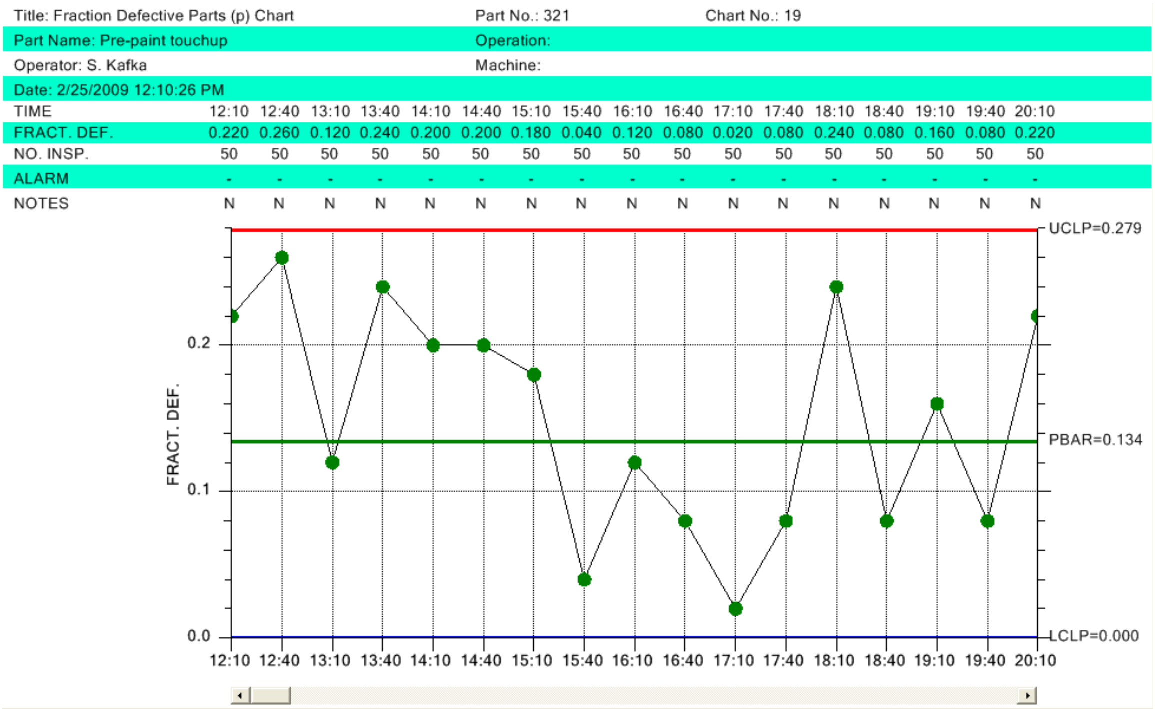
You can rotate the x-axis labels using the charts XAxisLabels.Rotation property.

```
"PrimaryChartSetup": {
    "XAxisLabels": {
        "Rotation": 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.

```
"ChartPositioning": {
   "InterGraphMargin":0.1,
   "GraphBottomPos":0.85
},
```

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

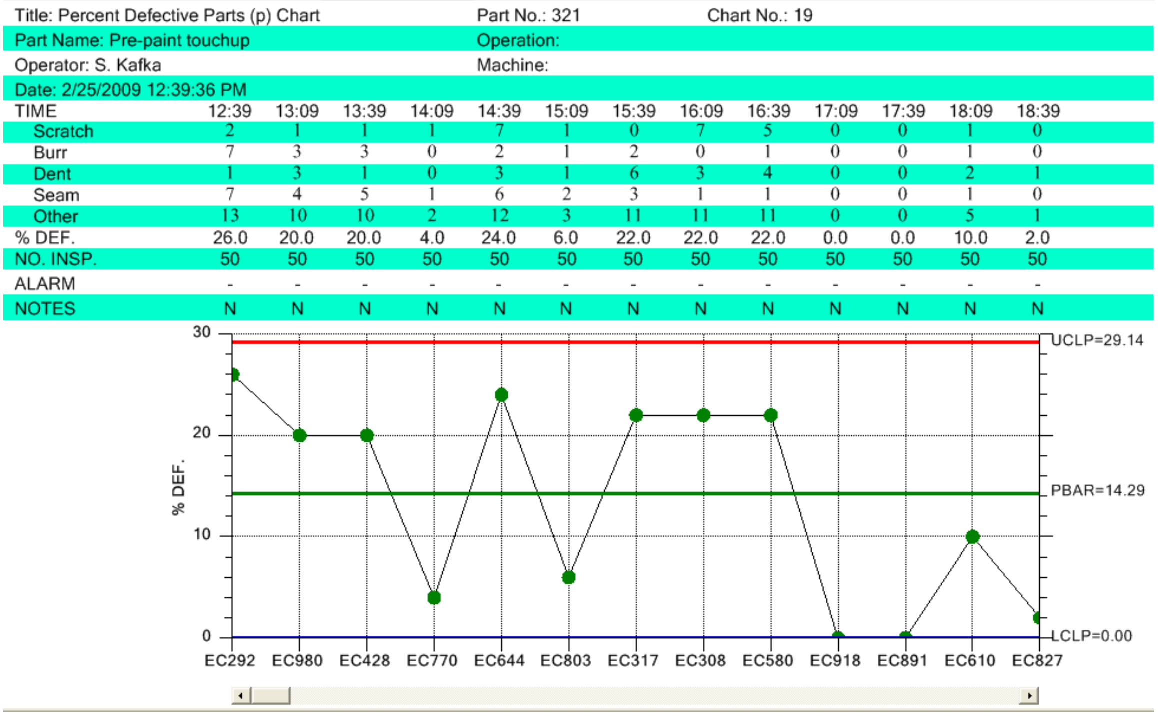
Set the x-axis labeling mode using the PrimaryChartSetup.XAxisLabels.AxisLabelMode property, setting it AXIS_LABEL_MODE_TIME.

```
   "PrimaryChartSetup": {
      "XAxisLabels": {
         "AxisLabelMode": "AXIS_LABEL_MODE_TIME"
      },
```

See the example program BatchXBarRChart for a complete example. Reset the axis labeling mode back to batch number labeling by assigning the AxisLabelMode property to AXIS_LABEL_MODE_DEFAULT.

Batch Control Chart X-Axis User-Defined String Labeling

Defective Parts Chart using user-defined string labeling of the x-axis



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

Set the x-axis labeling mode using the overall charts AxisLabelMode property, setting it AXIS_LABEL_MODE_STRING.

```
"PrimaryChartSetup": {
   "XAxisLabels": {
      "AxisLabelMode": "AXIS_LABEL_MODE_STRING"
    },
```

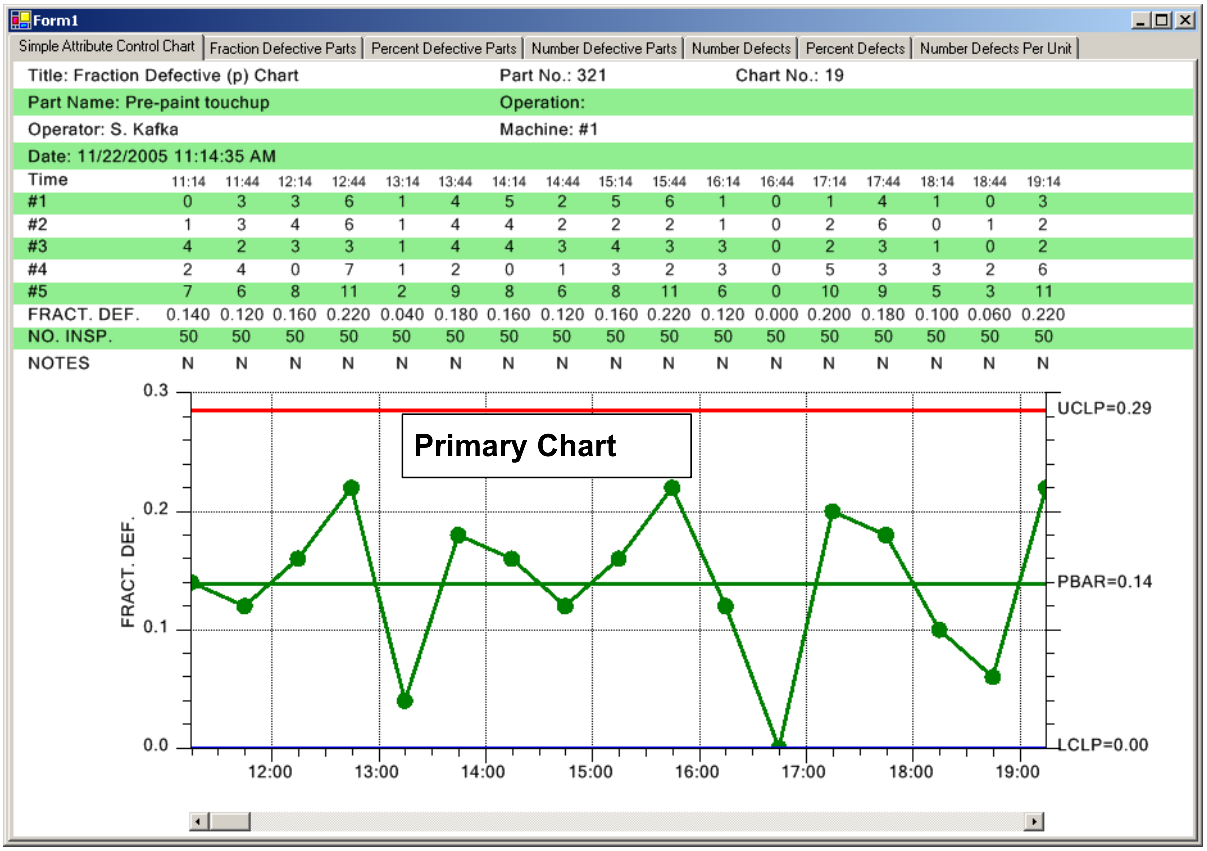
Set the string using the SampleData.SampleIntervalRecords.BatchIDString property.

Reset the axis labeling mode back to batch number labeling by assigning the AxisLabelMode property to AXIS_LABEL_MODE_DEFAULT.

```
          "SampleData": {
             "SampleIntervalRecords": [
                {
                   "SampleValues": [
                      5,
                      6,
                      5,
                      6,
                      13
                   ],
                   "BatchCount": 50,
                    "TimeStamp": 1371830829074,
                    "BatchIDString": "IDS50",
                    "Note": ""
                },
                {
                    "SampleValues": [
                        2,
                        2,
                        1,
                        2,
                        4
                    ],
                    "BatchCount": 51,
                    "TimeStamp": 1371831729074,
                    "BatchIDString": "IDS51",
                    "Note": ""
                },
                {
                    "SampleValues": [
                        0,
                        0,
                        1,
                        1,
                        2
                    ],
                    "BatchCount": 52,
                    "TimeStamp": 1371832629074,
                    "BatchIDString": "IDS52",
                    "Note": ""
                },
                {
                    "SampleValues": [
                        2,
                       5,
                        1,
                        2,
                        3
                    ],
                    "BatchCount": 53,
                    "TimeStamp": 1371833529074,
                    "BatchIDString": "IDS53",
                    "Note": ""
                }

            ]
        },
```

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.



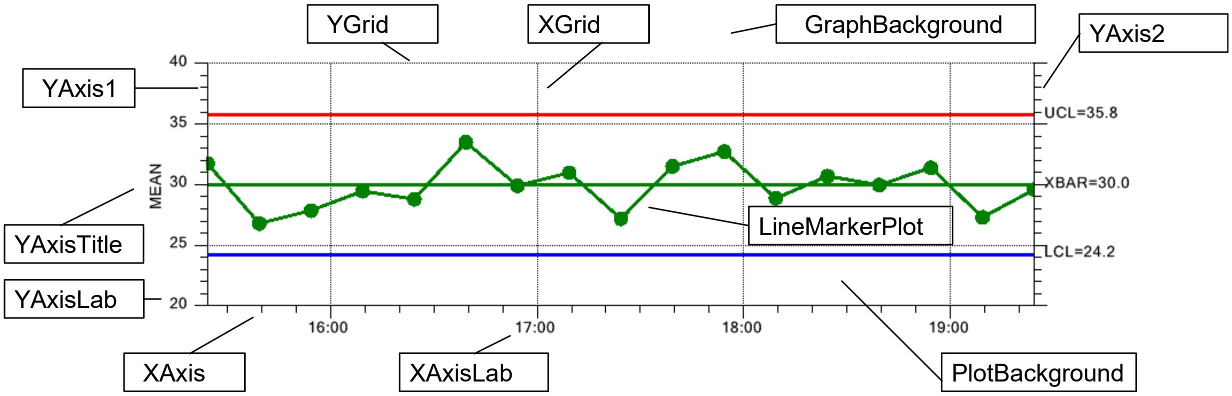
Title: Fraction Defective (p) Chart Part No.: 321 Chart No.: 19
Part Name: Pre-paint touchup Operation:
Operator: S. Kafka Machine: #1
Date: 11/22/2005 11:14:35 AM

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

You can modify the default characteristics of each graph using these properties.

```
"PrimaryChartSetup": {
    "FrequencyHistogram": {
        "EnableDisplayFrequencyHistogram": true,
        "PlotBackgroundColor": "WHITE",
        "BarColor": "BLUE"
    },
    "LineMarkerPlot": {
        "LineColor": "GREEN",
        "LineWidth": 2,
        "SymbolColor": "SPRINGGREEN",
        "SymbolFillColor": "SPRINGGREEN",
        "SymbolType": "CIRCLE"
    },
    "PlotBackground": {
        "FillColor": "BROWN",
        "BackgroundMode": "SIMPLECOLORMODE"
    },
    "XAxis": {
        "LineColor": "BLUE",
            "LineWidth": 3
        },
        "YAxisLeft": {
            "LineColor": "GREEN",
            "LineWidth": 3
        },
        "YAxisRight": {
            "LineColor": "RED",
            "LineWidth": 3
        },
```

The main objects of the graph are labeled in the graph below.



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

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 (pP * (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

$$\text{UCL} = (n * p) + 3 * \text{Sqrt}\ ((n * p) * (1- p) / n)$$

$$\text{Center line} = (n * p)$$

$$\text{LCL} = (n * p) - 3 * \text{Sqrt}\ ((n * p) * (1- p) / n)$$

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

$$\text{UCL} = up + 3000 * \text{Sqrt}\ (up /(dopu * n))$$

$$\text{Center line} = up$$

$$\text{LCL} = up - 3000 * \text{Sqrt}\ (up/(dopu * n))$$

Number of Defects Control Chart – Also known as Number Nonconformities or c-chart

$$\text{UCL} = c + 3 * \text{Sqrt}\ (c)$$

$$\text{Center line} = c$$

$$\text{LCL} = c - 3 * \text{Sqrt}\ (c)$$

Number of Defects per Unit Control Chart – Also known as Number Nonconformities per Unit or u-chart

$$\text{UCL} = u + 3 * \text{Sqrt}\ (u / n)$$

Center line = u

LCL = u - 3 * Sqrt (u / n)

12. Process Capability Ratios and Process Performance Indices

Introduction to Process Capability Ratios and Process Performance

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.ProcessCapabilitySetup.LSLValue** and **ChartData.ProcessCapabilitySetup.USLValue** properties of the chart. The code below is from the chartDefExampleScripts.js TimeXBarR example JSON script.

```
"ChartData": {

   "ProcessCapabilitySetup": {
      "LSLValue": 27,
      "USLValue": 35,
      "EnableCPK": true,
      "EnableCPM": true,
      "EnablePPK": true
   }

 }
```

Enable which process capability statistics you want to see in the table. Use one of the **Enable properties** constants below to specify the statistics that you want displayed.

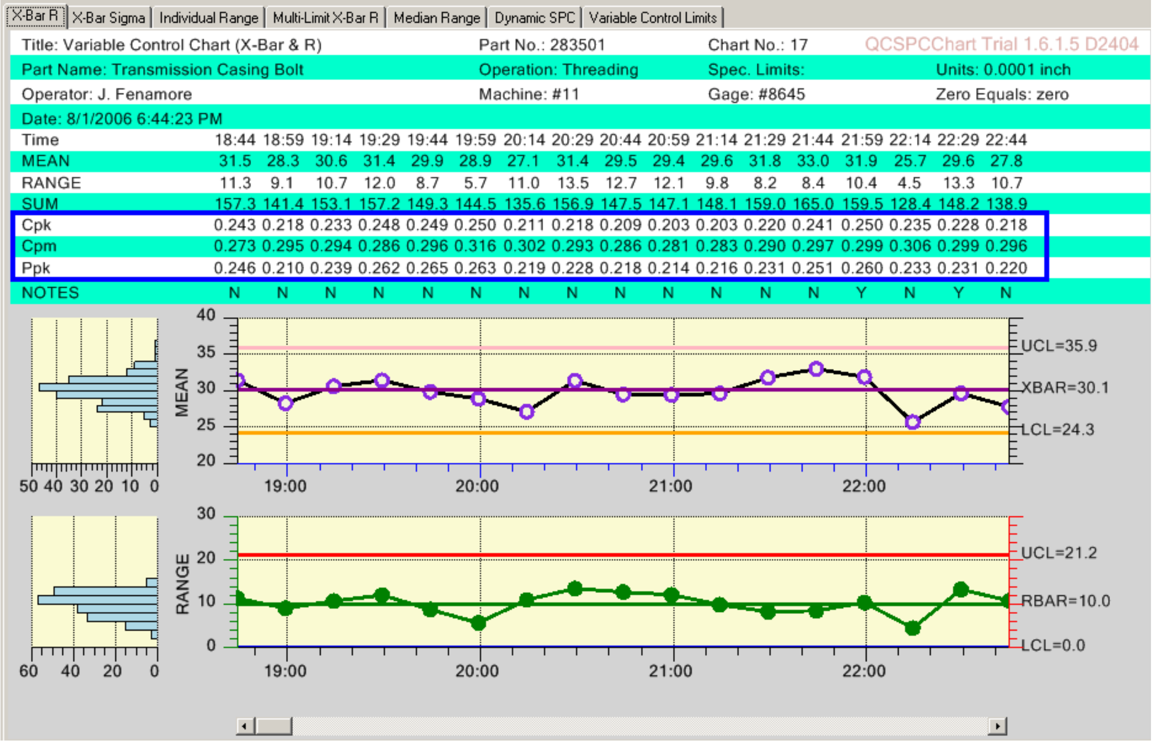
```
EnableCPK: boolean: false
EnableCPM: boolean: false
EnablePPK: boolean: false
EnableCPL: boolean: false
EnableCPU: boolean: false
EnablePPL: boolean: false
EnablePPU: boolean: false
```

The code below is from the chartDefExampleScripts.js TimeXBarR example JSON script.

```
"ChartData": {
```

```
          "ProcessCapabilitySetup": {
             "LSLValue": 27,
             "USLValue": 35,
             "EnableCPK": true,
             "EnableCPM": true,
             "EnablePPK": true
          }
}
```

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)

$\overline{\overline{X}}$ = XDoubleBar - Mean of sample subgroup means (also called the grand average)

$\overline{R}$ = 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.

13. Named and Custom Control Rule Sets

Named Rule Sets

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.

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 limit
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 limit
2. Two of three points above 2-sigma control limit
3. Two of three points below -2-sigma control limit
4. Four of five points is above 1-sigma control limit
5. Four of five points is below -1-sigma control limit
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 limit
2. Two of three points above 2-sigma control limit
3. Two of three points below -2-sigma control limit
4. Three of seven points above 2-sigma control limit
5. Three of seven points below -2-sigma control limit
6. Four of ten points above 2-sigma control limit
7. Four of ten points below -2-sigma control limit
8. Four of five points is above 1-sigma control limit
9. Four of five points is below -1-sigma control limit
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 limit
2. Two of three points above 2-sigma control limit
3. Two of three points below -2-sigma control limit
4. Four of five points is above 1-sigma control limit
5. Four of five points is below -1-sigma control limit
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 limit
2. Two of three points above 2-sigma control limit
3. Two of three points below -2-sigma control limit
4. Four of five points is above 1-sigma control limit
5. Four of five points is below -1-sigma control limit
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.

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 of 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

we have expanded the Basic Rule #'s to include the following additional rules, which might be used in some cases. These rules would only be enabled in a special case.

#2	1 of 1 < 2 sigma	3	1	1	1	-2
	1 of 1 > 2 sigma	4	2	1	1	2
#3	1 of 1 < 1 sigma	5	1	1	1	-1
	1 of 1 > 1 sigma	6	2	1	1	1

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

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.

NamedRuleSet

```
NamedRuleSet
       RuleSet: string constant
       RuleEnable [boolean, boolean …]
```

The NameRuleSet property will invoke a complete set of control rules based one of following standard rule sets: BASIC_RULES, WECO_RULES,WECOANDSUPP_RULES, NELSON_RULES,AIAG_RULES, JURAN_RULES, HUGHES_RULES,GITLOW_RULES, WESTGARD_RULES,and DUNCAN_RULES. For a complete discussion of named control rules, see chapter xxxx.

RuleSet

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.

RuleEnable

An array of boolean, one for each named rule in the rule set. All of the rules are enabled by default. This permits you to disable specific rules.

Example

```
"NamedRuleSet":
{
  "RuleSet": "WECO_RULES",
  "RuleEnable": [ true, true, false, true, false, true, true, true]
}
```

***Important Note:** 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. You use our rule numbering system for specifying which rule.

See one of the following JSON scripts for examples of how to setup an SPC chart for a given set of control rules: WECORules, WECOAndSupplementalRules, NelsonRules, AIAGRules, HughesRule. 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 **SpecifyControlLimitsUsingMeanAndSigma** method. It will run through all of the control limit records and fill out the appropriate limit values and other critical parameters.

SpecifyControlLimitsUsingMeanAndSigma

```
SpecifyControlLimitsUsingMeanAndSigma
   Mean: double: 1
   Sigma: double: 1
```

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 invoke SpecifyControlLimitsUsingMeanAndSigma method. It will run through all of the control limit records and fill out the appropriate limit values and other critical parameters.

Mean

specify the process mean.

Sigma

specify the process sigma.

The center line value and sigma have different meanings for the Primary and Secondary charts. So the **SpecifyControlLimitsUsingMeanAndSigma** 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.

```
"SpecifyControlLimitsUsingMeanAndSigma": {
   "Mean":  30,
   "Sigma": 1.666
};
```

You can also auto-calculate the control limits by adding test data to your application (fed into the chart using the SampleData block), and calling **AutoCalculateControlLimits.** This establishes control limits 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 which establish named rule sets.

```
 "PrimaryChartSetup": {
    "ControlLimits":
    {
      "ZoneFill": true,
      "NamedRuleSet":
      {
         "RuleSet": "WECO_RULES"

      }
   }
 },
"SampleData": {
   "DataSimulation": {
      "StartCount": 0,
      "Count": 50,
      "Mean": 27,
      "Range": 5
   }
},
"Methods": {
   "AutoCalculateControlLimits": true,
   "AutoScaleYAxes": true,
   "RebuildUsingCurrentData": true
}
```

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 **NamedRuleSet.CustomizeRules** array property.

```
NamedRuleSet
   RuleSet: SPC string constant
   RuleEnable [boolean, boolean …]
   CustomizeRules: [   {
                          "RuleNumber": 15,
                          "M": 18,
                          "N": 15
                       },
                       {  "RuleNumber": 15,
                          "M": 18,
                          "N": 15
                       },
                       ...
                 ]
```

CustomizeRules
An array, one for each rule you want to modify in the ruleset. Each block in the array contains the rule number, the M-value (N out of M must exceed the limit value for a violation to occur) and an N-value.

RuleNumber
The rule number (our rule number)

M
The M-value (N out of M must exceed the limit value for a violation to occur). Specifies the number of values to use in calculation

N
The N-value (N out of M must exceed the limit value for a violation to occur)
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). Specifies the number of values that must be outside alarm limit for rule violation.

```
"PrimaryChartSetup": {
      "NamedRuleSet":
      {
         "RuleSet": "HUGHES_RULES",
         "CustomizeRules": [
         {
            "RuleNumber": 15,
            "M": 18,
            "N": 15
         },
         {
            "RuleNumber": 16,
            "M": 18,
            "N": 15
          }
       ]
     }
},
```

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 **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.

```
AddControlRules
[  {
         RuleSet: : SPC String constant: "BASIC_RULES"
         RuleNumber: integer: 2
         EnableAlarmLine: boolean: true
         EnableAlarmChecking: boolean: true
         EnableAlarmLineText: String: true
         M: integer: 1
         N: integer: 1
   },
   {
         RuleSet: : SPC String constant: "BASIC_RULES"
         RuleNumber: integer: 2
         EnableAlarmLine: boolean
         EnableAlarmChecking: boolean
         EnableAlarmLineText: String
         M: integer: 1
         N: integer: 1
   }, ...

]
```

The **AddControlRules** property is an array of control rule specifications. Since it is an array, you can add as many control rules as you want. Each specification block in the array defines one control rule. Note how the control rule array is bracketed by [ ], signifying the start and and of the array. Each block element in the array is bracketed using { }.

A control rule block element has the following parameters:

RuleSet
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.

RuleNumber
The rule number (our rule number)

EnableAlarmLine
Enable the drawing of the limit line for the control rule.

EnableAlarmCheckin
Enable alarm checking for the the control rule.

EnableAlarmLineText
Enable the drawing of the limit text for the control rule.

M
The M-value (N out of M must exceed the limit value for a violation to occur,)

N
The N-value (N out of M must exceed the limit value for a violation to occur)

A multi-rule example would look something like:

```
"AddControlRules": [
      {
         "RuleSet": "WECO_RULES",
         "RuleNumber":  2
      },
      {
         "RuleSet": "WECO_RULES",
         "RuleNumber":  3
      },
      {
         "RuleSet": "NELSON_RULES",
         "RuleNumber":  12
      },
      {
         "RuleSet": "JURAN_RULES",
         "RuleNumber":  9,
         "EnableAlarmLine": false,
         "EnableAlarmChecking": true,
         "EnableAlarmLineText": false
      }
]
```

Even if you do use AddControlRules, you still start 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 ControlLimits.**DefaultLimits[false, false]**.

```
DefaultLimits [ boolean: true, boolean: true]
```

Specifies whether default UCL and LCL limits are enabled.

DefaultLimits is an array of two booleans.

DefaultLimits[0]
Set to false to disable the checking of the default +- 3 Sigma limits, also known as UCL (upper control limits) and LCL (lower control limit).

DefaultLimits[1]
Set to false to disable drawing the +- 3 Sigma limits lines and associated text.

```
"DefaultLimits": [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). The default +-3 -sigma rules are left in place.

```
"ControlLimits": {
      "AddControlRules": [
             {
                    "RuleSet": "NELSON_RULES",
                    "RuleNumber":  3
             },
             {
                    "RuleSet": "NELSON_RULES",
                    "RuleNumber":  4
             },
             {
                    "RuleSet": "JURAN_RULES",
                    "RuleNumber":  5
             },
             {
```

```
                        "RuleSet": "JURAN_RULES",
                        "RuleNumber":  6
                },
                {
                        "RuleSet": "AIAG_RULES",
                        "RuleNumber":  5
                },
                {
                        "RuleSet": "AIAG_RULES",
                        "RuleNumber":  6
                },
                {
                        "RuleSet": "HUGHES_RULES",
                        "RuleNumber":  12
                },
                {
                        "RuleSet": "DUNCAN_RULES",
                        "RuleNumber":  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. Whether it makes any statistical sense is doubtful.

Creating Custom Rules Sets Based on a Template

Add your own custom rule to the rule set using different parameters of the **AddControlRule** 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.

A control rule block element has the following parameters:

RuleSet

One of the named rule identifiers: BASIC_RULES, WECO_RULES,WECOANDSUPP_RULES, NELSON_RULES,AIAG_RULES, JURAN_RULES, HUGHES_RULES,GITLOW_RULES, WESTGARD_RULES, DUNCAN_RULES and CUSTOM_TEMPLATE_BASED_RULE.

RuleNumber

The rule number (our rule number). If the RuleSet is of type CUSTOM_TEMPLATE_BASED_RULE, then the RuleNumber specifies the template number of the desired template.

EnableAlarmLine

Enable the drawing of the limit line for the control rule.

EnableAlarmChecking

Enable alarm checking for the the control rule.

EnableAlarmLineText

Enable the drawing of the limit text for the control rule.

M

The M-value (N out of M must exceed the limit value for a violation to occur,)

N

The N-value (N out of M must exceed the limit value for a violation to occur)

If the RuleSet is CUSTOM_TEMPLATE_BASED_RULE, the following parmeters are also valid:

SigmaLevel

The sigma level of the desired control rule template.

In your code it would something like:

```
"ControlLimits": {
   "AddControlRules": [
      {
         "RuleSet": "CUSTOM_TEMPLATE_BASED_RULE",
         "RuleNumber":  1,
         "N": 10,
         "M":  13,
         "SigmaLevel": -2,
         "EnableAlarmLine": false,
         "EnableAlarmLineText": false

      },
      {
         "RuleSet": "CUSTOM_TEMPLATE_BASED_RULE",
         "RuleNumber":  1,
         "N": 10,
         "M":  13,
         "SigmaLevel": 2,
         "EnableAlarmLine": false,
         "EnableAlarmLineText": false
      }
   ]

}
```

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. Specification limits are control rules not directly related to the mean and sigma value of the chart. Use a SpecificaitonLimits block to add spec limits to a chart.

```
SpecificationLimits
   Font
      Name: String: "sans-serif"
      Size: double: 12
      Style: SPC String Constant: "PLAIN"
   Decimal: integer: 1
   LowSpecificationLimit
      LineColor: Color String constant: "BLUE"
      TextColor: Color String constant: "BLACK"
      LineWidth: double: 1
      LimitValue: double: 0
      DisplayString: String: "LSL"
      EnableAlarmLine: boolean: true
      EnableAlarmChecking : boolean: true
      EnableAlarmLineText: String: true
   HighSpecificationLimit: double
      LineColor: Color String constant: "RED"
      TextColor: Color String constant: "BLACK"
      LineWidth: double: 1
      LimitValue: double: 0
      DisplayString: String: "USL"
      EnableAlarmLine: boolean: true
      EnableAlarmChecking : boolean: true
```

Font

Specifies the Font used to annotate the control limits on the RHS of the chart.

```
Font
```

```
   Name: String: "sans-serif"
   Size: double: 12
   Style: SPC String Constant: "PLAIN"
```

Name

Set the axis labels font family to something other than the default "sans-serif". Follow the font naming guidelines detailed in Static Properties chapter, under DefaultTableFont

Size

Font size in points.

Style

Use of the style string constants: "Plain", "Normal", "Bold","Italic","Bold Italic".

Decimal

```
Decimal: integer: 1
```

Decimal

Set to the decimal precision to display the limit values.

LowSpecificationLimit

```
LowSpecificationLimit
   LineColor: Color String constant: "BLUE"
   TextColor: Color String constant: "BLACK"
   LineWidth: double: 1
   LimitValue: double: 0
   DisplayString: String: "LSL"
   EnableAlarmLine: boolean: true
   EnableAlarmChecking : boolean: true
   EnableAlarmLineText: String: true
```

Set the properties associated with the Low SpecificationLimit (LSL) line

LineColor

The line color of the limit line.

TextColor

The text color of the limit line label.

LineWidth

The line width of the limit line

LimitValue
Set the limit value.

DisplayString
Set the text string which precedes the numeric value of the limit line label.

EnableAlarmLine
Set to false to disable the alarm line.

EnableAlarmChecking
Set to false to disable the limit testing against the limit value.

EnableAlarmLineText
Set to false to disable the limit line text on the right.

HighSpecificationLimit

```
HighSpecificationLimit
LineColor: Color String constant: "RED"
TextColor: Color String constant: "BLACK"
LineWidth: double: 1
LimitValue: double: 0
DisplayString: String: "USL"
EnableAlarmLine: boolean: true
EnableAlarmChecking : boolean: true
```

Set the properties associated with the High SpecificationLimit (HSL) line.
Same property set as LowSpecificationLimi.t

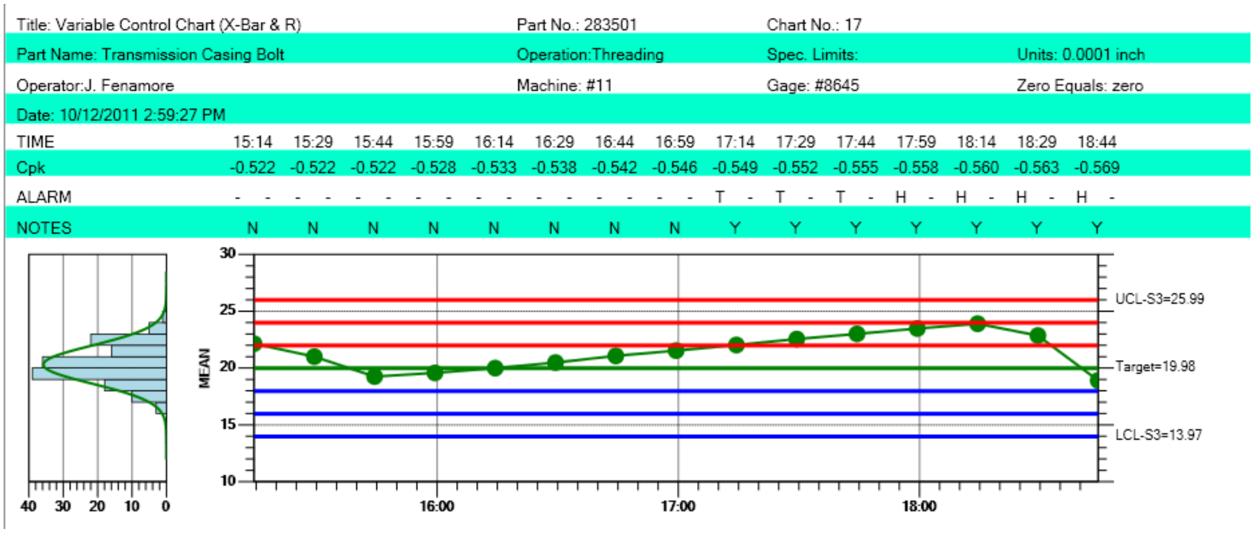
Enable Alarm Highlighting

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.

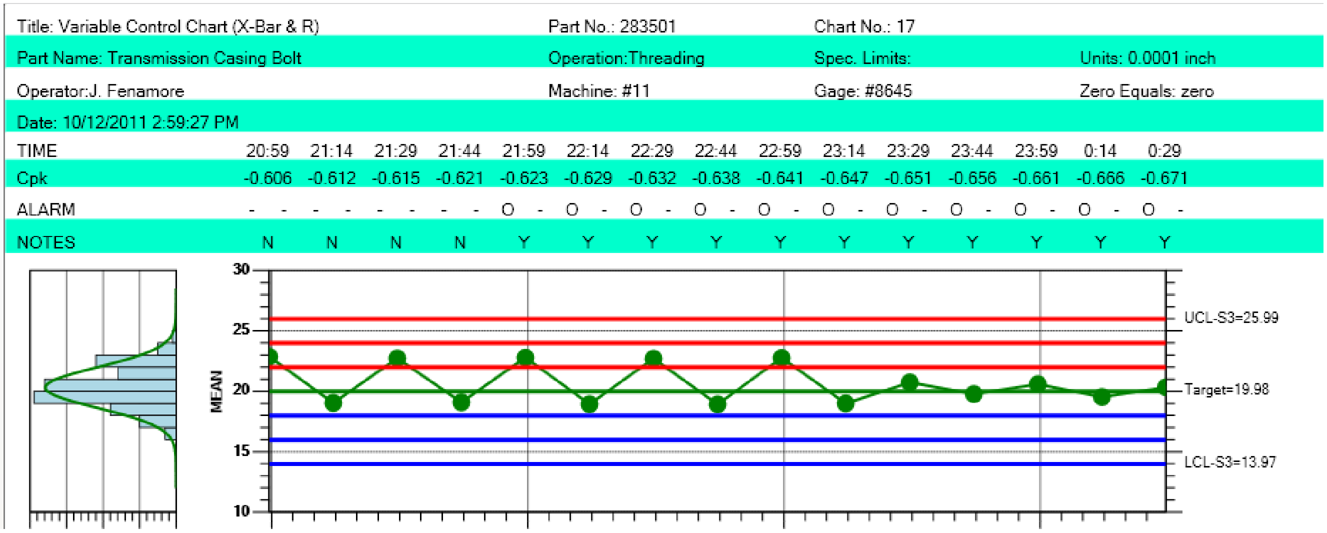
Trending



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

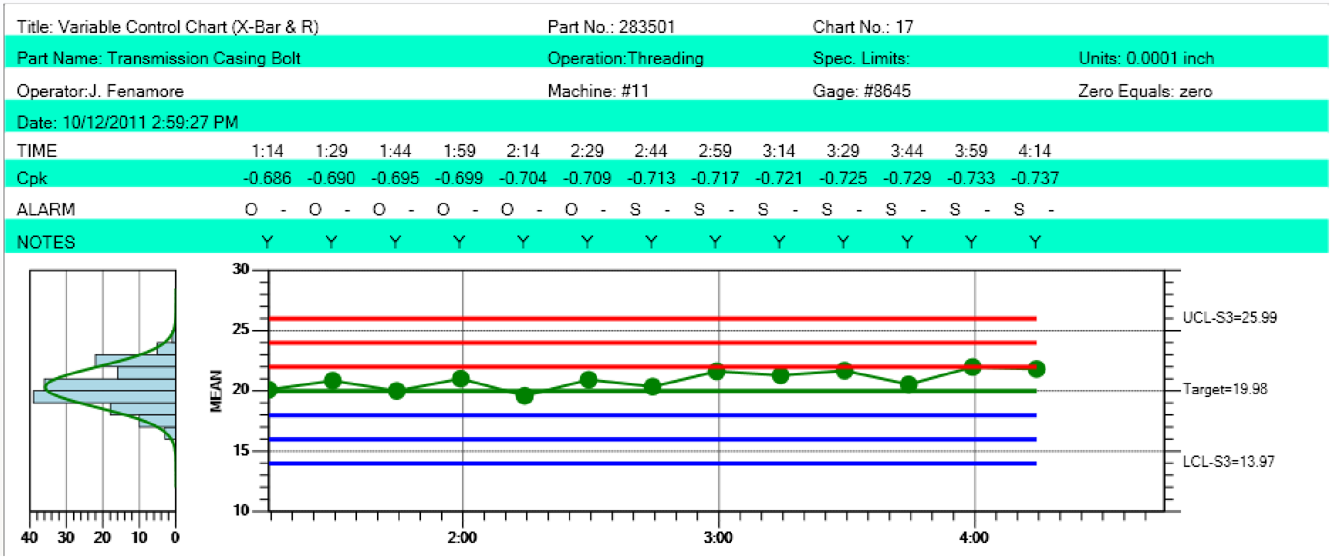
Oscillation



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

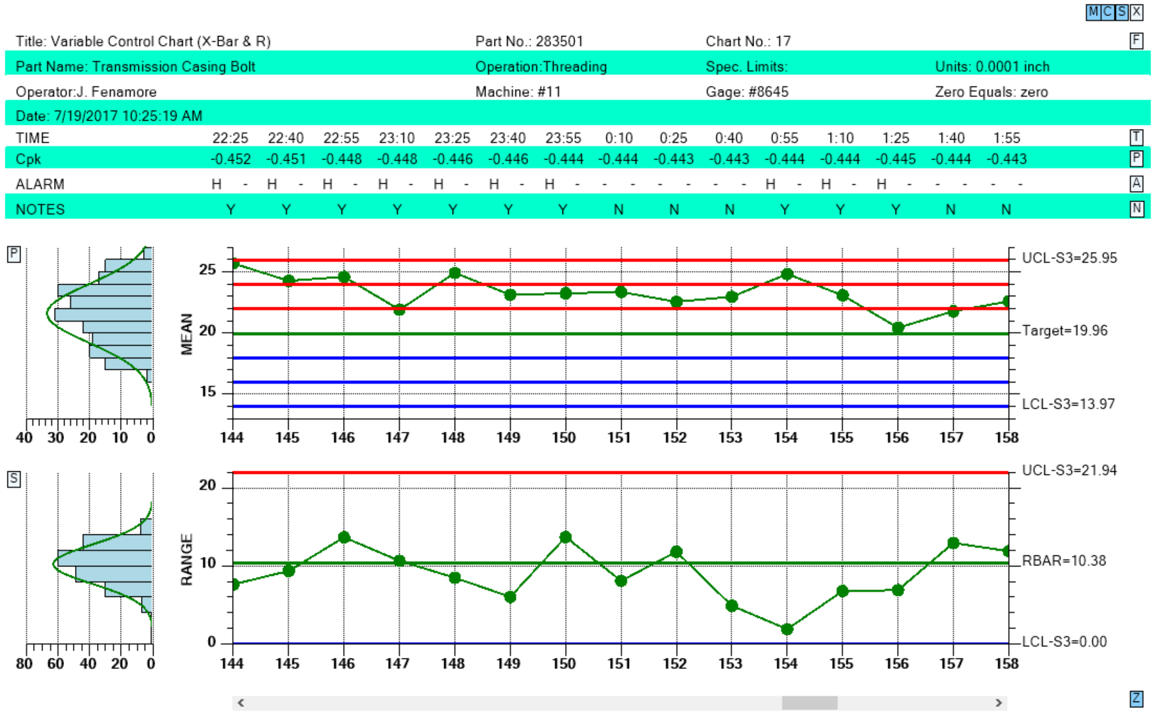
```
  "Events": {
            "EnableAlarmStatusValues": true

        },
```

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 method `ResetControlLimitNofMCounts` to reset the N of M counters back to 0.

```
  "Methods": {
            "ResetControlLimitNofMCounts": true

        },
```



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.

AutoDeleteControlLimits

```
AutoDeleteControlLimits
        LimitValue1
        LimitValue2
        LimitValue3
```

```
ComparisonOperator
```

LimitValue1 Remove limits in the Primary Chart which are (ComparisonOperator) than the LimitValue1 value.

LimitValue2 Remove limits in the Secondary Chart which are (ComparisonOperator) than the LimitValue2 value.

LimitValue3 Remove limits in the Third Chart which are (ComparisonOperator) than the LimitValue3 value.

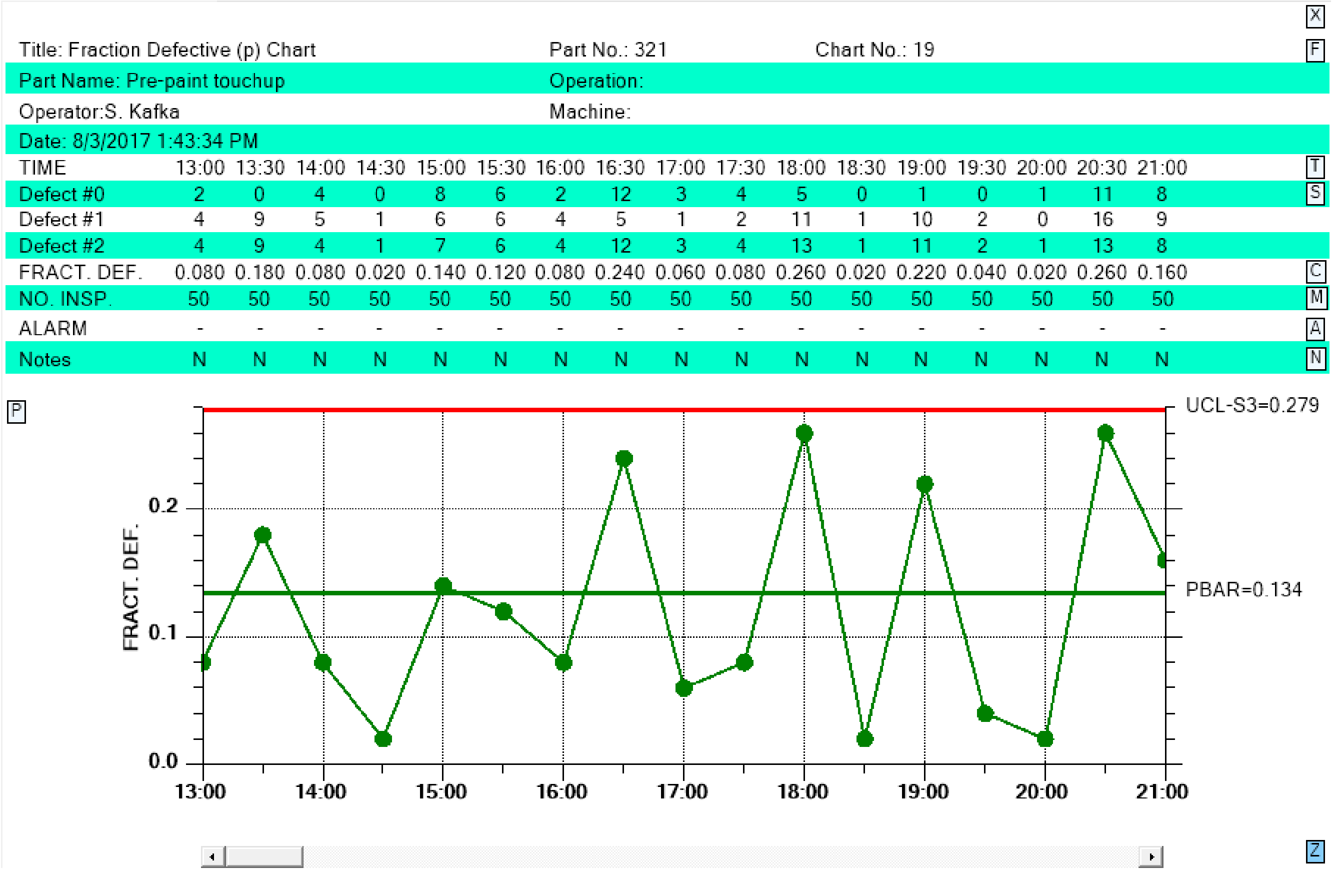
ComparitonOperator Use this comparison operator. Use one of the comparison operator constants: SPCControlChartData.COMPARISON_OP_LESSTHAN, SPCControlChartData.COMPARISON_OP_LESSTHAN_OR_EQ, SPCControlChartData.COMPARISON_OP_GREATERTHAN, SPCControlChartData.COMPARISON_OP_GREATERTHAN_OR_EQ,

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

```
Methods: {
.
.
.

       "AutoDeleteControlLimits":  {
          “LimitValue2”: 0.0,
            “ComparisonOperator”:  “COMPARISON_OP_LESSTHAN_OR_EQ”
       }
}
```

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

```
Methods: {
.
.
.

       "AutoDeleteControlLimits":  {
          “LimitValue1”: 0.0,
          “ComparisonOperator”:  “COMPARISON_OP_LESSTHAN_OR_EQ”
       }
}
```



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

AltNofMRule

```
AltNofMRule: boolean: false
```

The AltNofMRule static property sets an alternative alarm processing method for those rules which use a N of M control rule, such as most of the named rules sets have for their 2 out of 3 outside of 2-sigma, and 4 out of 5 outside of 1-sigma rules. The default method we use is if that if N of M (2 of 3, or 4 of 5) is out of bounds for the current and previous sample intervals, then the current sample is in alarm, regardless if the current sample is actually within bounds. In the 2 of 3 example, this means that if the previous two sample intervals are out of bounds, but the current sample interval is in-bounds, the current sample fails the 2 of 3 test. The **alternative** is to look at the current sample and if it is not out of bounds, consider the sample interval not in alarm, even if the previous 2 of 2, or 4 of 4 sample intervals are out of bounds.

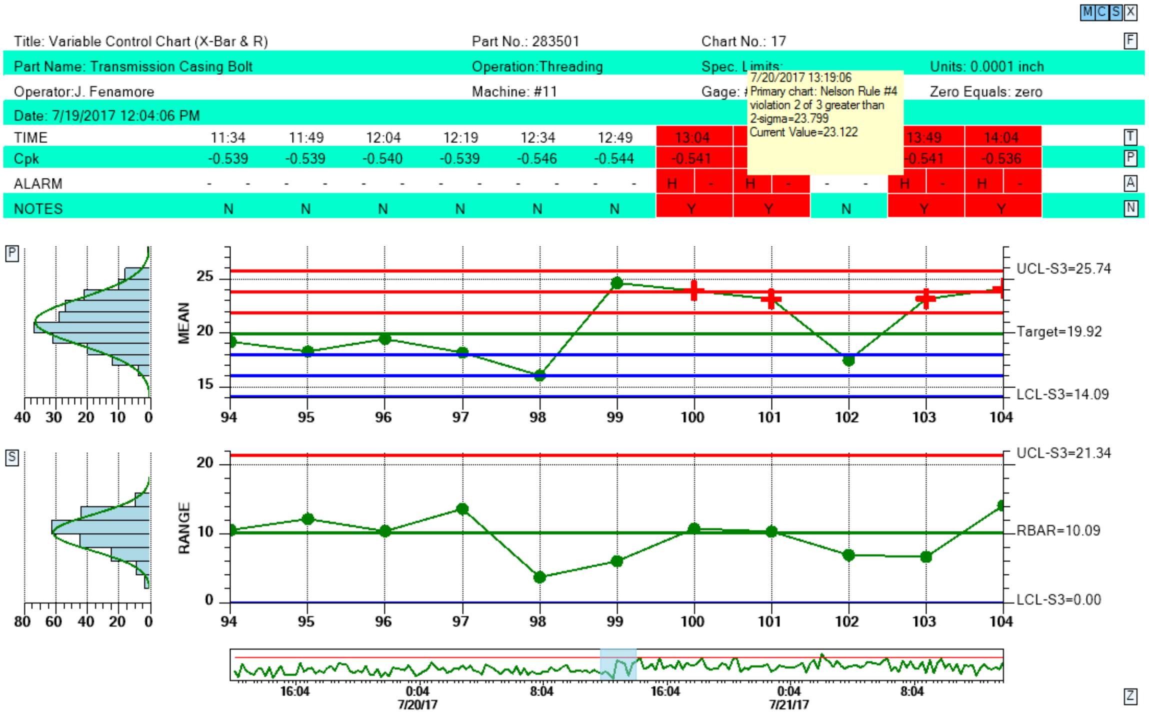
So, don't set **AltNofMRule** if you want to use the default method. Set it true if you do want to use the alternative alarm processing method.

```
“StaticProperties”: {
.
.
.

    “AltNofMRule”: 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.

Usi *the*



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

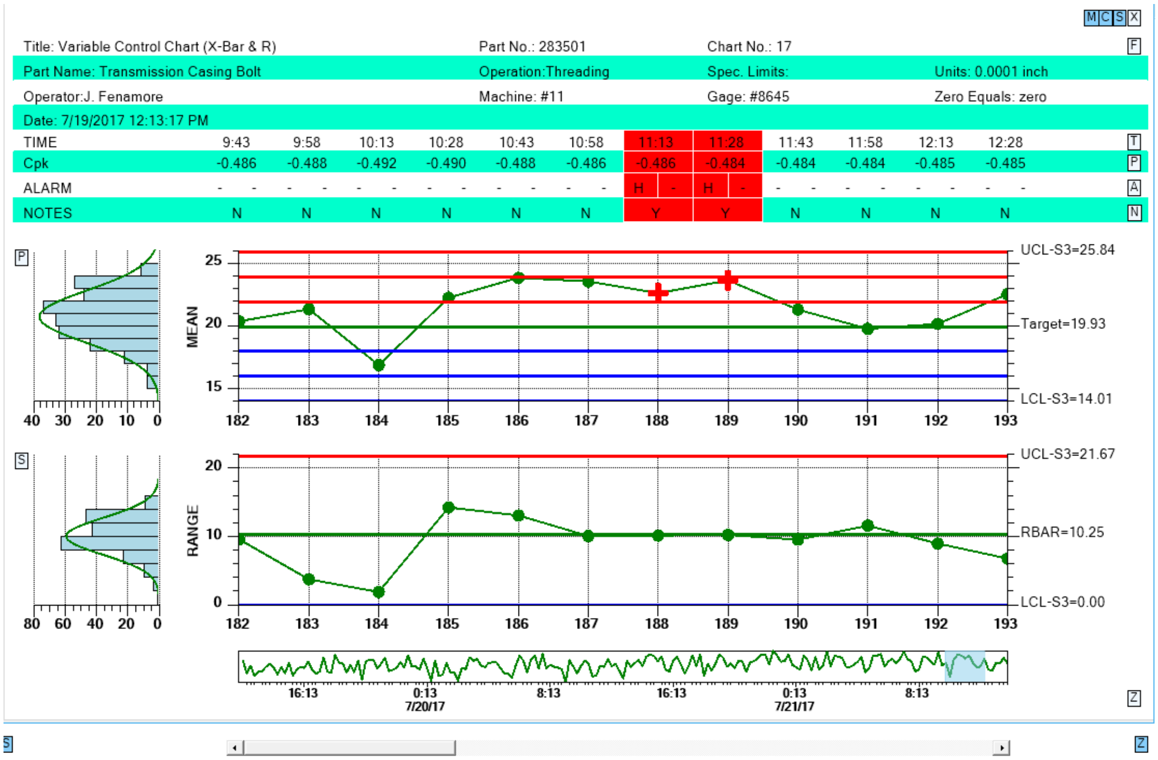
TIME	11:34	11:49	12:04	12:19	12:34	12:49	13:04			13:49	14:04
Cpk	-0.539	-0.539	-0.540	-0.539	-0.546	-0.544	-0.541			-0.541	-0.536
ALARM	- -	- -	- -	- -	- -	- -	H -	H -	- -	H -	H -
NOTES	N	N	N	N	N	N	Y	Y	N	Y	Y

If the following code was added to the StaticProperties of the chart setup

```
“AltNofMRule”: true,
```

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

In the picture below, AltNofMRule 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.

14. Event Handling for Alarms and Tooltips

The alarm processing and notes components of the SPC Charting Tools generate events which can in turn invoke events in the Javascript code of the host HTML page. This is done using GWT JSNI routines which allow a bridge between the SPC Chart library, and the Javacode you write for your HTML page.

Processing Alarms

First, you must have one of the alarm event processing routines turned on. This is done in the Events block using the Events.AlarmStateEventEnable, or Events.AlarmTransitionEventEnable properties.

```
Events
   AlarmStateEventEnable: boolean: true
   AlarmTransitionEventEnable: boolean: false
```

AlarmStateEventEnable

If AlarmStateEventEnable is true, then at every sample interval, if the process variable is in any alarm condition, it triggers an alarm event.

AlarmTransitionEventEnable

If AlarmTransitionEventEnable is true, then only the transition into and out of an alarm condition triggers an alarm event. So if the process variable enters an alarm condition, and stays there, only the sample interval where the alarm condition starts triggers an event. A new event is triggered only if the process variable leaves the alarm condition, or enters a different one.

```
"Events": {

    "AlarmStateEventEnable": true

},
```

If that is done, any control limit violations for the chart will try and vector into the host HTML file. Specifically, it will try and vector to a Javascript function with the name JSONSPCAlarmEvent, passing in a JSON object detailing the control limit violation. The JSONSPCAlarmEvent function in your HTML must use the template below, slightly modified from the SPCMediumSimple example program:

```
function JSONSPCAlarmEvent(s)
{
    var jsonobj = JSON.parse(s);
    var message =  jsonobj.SPCAlarmEvent.AlarmMessage;
    var index = jsonobj.SPCAlarmEvent.DataIndex;
}
```

The event data is passed into the function as a JSON string, which you can parse into a standard Javascript record variable. The data structures contained therein can be accessed as any Javascript record structure. The structure of the data is:

```
SPCAlarmEvent
   ChartNumber: integer
   DataIndex: integer
   AlarmLimitValue: double
   CurrentValue: double
   TimeStampLong: long
   TimeStampString: string
   AlarmMessage: string
   SigmaLevel: double
   Template: integer
   RuleSet: integer
   RuleNumber: integer
   IsSigmaLimit: boolean
   NumberValuesForRuleViolation: integer
   NumberValuesInCalculation: integer
```

The way it actually appears as a JSON string, using actual values, is:

```
          {
          "SPCAlarmEvent": {
                "ChartNumber": 0,
                "DataIndex": 7,
                "AlarmLimitValue": 21,
                "CurrentValue": 22,
                "TimeStampLong": 1371831729074,
                "TimeStampString": "07/21/2013 13:11:01",
                "AlarmMessage": "High Alarm",
                "SigmaLevel": 3,
                "Template":  0,
                "RuleSet":   0,
                "RuleNumber": 1,
                "IsSigmaLimit": true,
                "NumberValuesForRuleViolation": 1,
                "NumberValuesInCalculation":  1
                }
         }
```

The data values of the fields in the SPCAlarmEvent structure are a accessed using the standard Javascript dot notation, as seen in the JSONSPCAlarmEvent example above.

ChartNumber

This value is always 0 in the current version of the software.

DataIndex

The index of the sample interval of the alarm in the current chart.

AlarmLimitValue

The control limit value at the sample interval of the alarm. Since some control charts can have variable control limits, the control limit can vary from sample interval to sample interval.

CurrentValue

The value of the process variable at the sample interval of the alarm.

TimeStampLong

The time stamp of the sample interval as a long number, representing milliseconds.

TimeStampString

A string representation of the type sample of the sample interval.

AlarmMessage

The alarm message associated with the alarm.

SigmaLevel

The sigma level of the alarm. Only applicatible if IsSigmaLimit true.

Template

The template number of the alarm.

RuleSet

Identifies the RuleSet of the alarm.

RuleNumber

Identifies the Rule Number of the alarm.

IsSigmaLimit

True if the control limit is sigma-based. Otherwise it is probably a specification limit, which is not sigma-based

NumberValuesForRuleViolation

The N of an N of M test. Specifies the number of values (N) , out of NumberValuesInCalculation (M) sequential values tested.

NumberValuesInCalculation

The M of an N of M test. Specifies the number of sequential values tested for a given control limit test.

One of the things that the GWT JSNI interface allows the library to do is to check to see if a JSONSPCAlarmEvent is present in the host HTML file. If it finds it it calls it, otherwise it skips the call. What it can't do is determine if the code in the JSONSPCAlarmEvent is valid Javascript. So if you have an error in the JSONSPCAlarmEvent, it won't work correctly.

Example

You will find an example of processing an alarm in the SPCMediumSimple.html file.

Processing Data Tooltips

First, you must have one of the data tooltip processing routines turned on. This is done in the Events block using the Events.EnableDataToolTip , or Events. EnableJSONDataToolTip properties.

```
Events

   EnableDataToolTip: boolean: true
   EnableJSONDataToolTip: boolean: false
   DataToolTip
      EnableCategoryValues: boolean: false
      EnableProcessCapabilityValues: boolean: false
      EnableCalculatedValues: boolean: false
      EnableNotesString: boolean: false
```

The EnableDataToolTip option is self-contained. You can set various options for what is displayed in the tool tip, items such as the sample values, process capabilitiyvalues, calculated values and the notes string. Those items are displayed in a pop-up box if you click on a data point in the chart. The EnableJSONDataToolTip option allows you to display anything you want in the pop-up window. When the mouse is clicked on a data point, that information is vectored into the parent HTML file, where you can define a custom string, and return that string to the tool tip processing routine, where it is displayed in the pop-up window.

Standard Data Tooltips

The properties below work together. If EnableDataToolTip is true, when a data point is clicked on, a tool tip will appear with the display options enabled under the DataToolTip block.

```
Events

   EnableDataToolTip: boolean: true
   DataToolTip
      EnableCategoryValues: boolean: false
      EnableProcessCapabilityValues: boolean: false
      EnableCalculatedValues: boolean: false
      EnableNotesString: boolean: false
```

EnableCategoryValues
Display the category (subgroup sample values) in the data tooltip.

EnableProcessCapabilityValues
Display the process capability (Cp, Cpl, Cpu, Cpk, Cpm, Pp, Ppl, Ppu and Ppk) statistics currently being calculated for the chart.

EnableCalculatedValues
Display the calculated values used in the chart (Mean, range and sum for an Mean-Range chart).

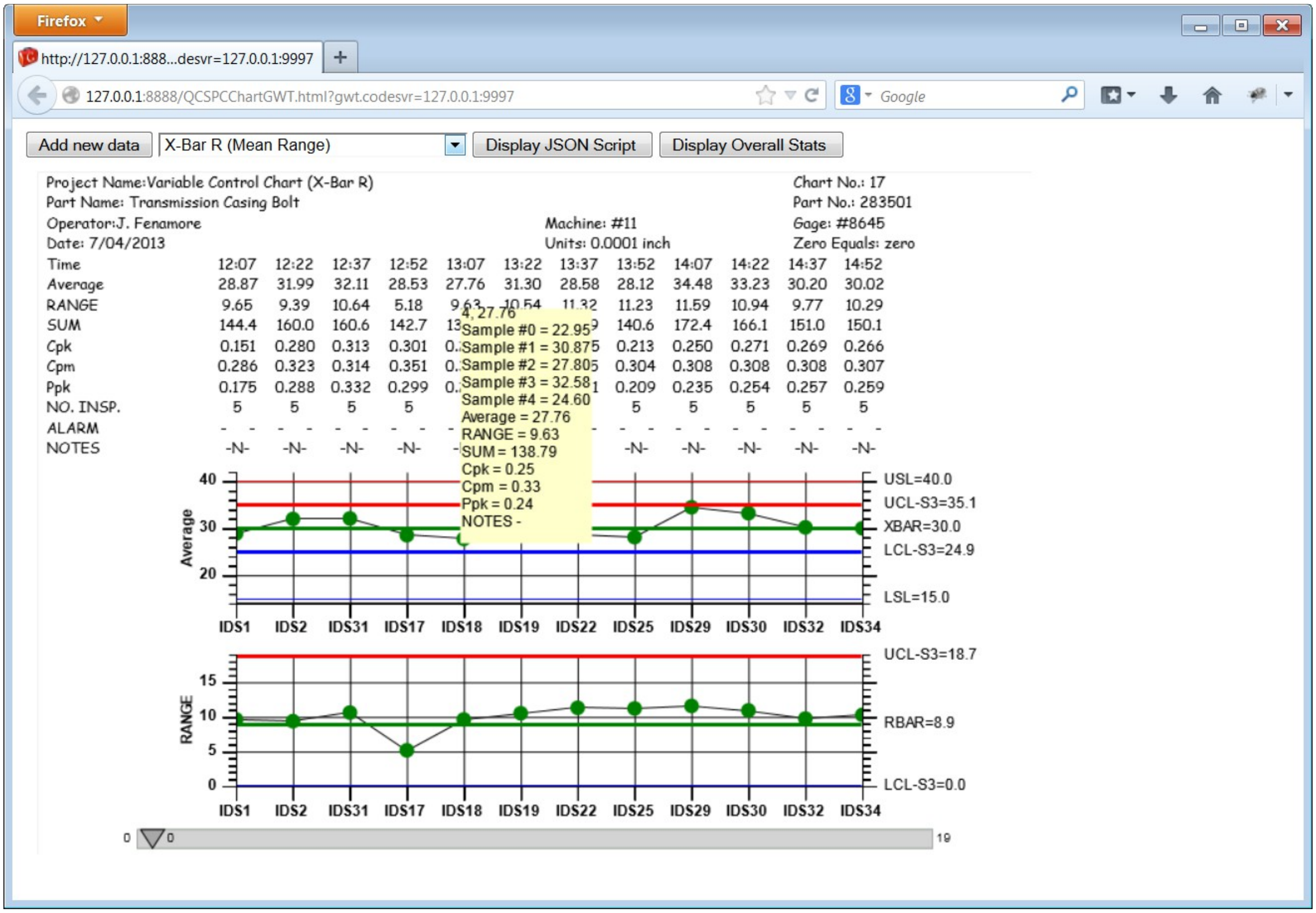
EnableNotesStrings
Display the current notes string for the sample subgroup.

Example

The variable control chart below displays a tooltip with all of the enable options above set true. See an example in the SPCMediumSimple.js JSON script file.

```
"Events": {
    "EnableDataToolTip": true,
    "EnableNotesToolTip": true,
    "EnableJSONDataToolTip": false,
    "AlarmStateEventEnable": true

},
```

Data Tooltip with optional display items


Firefox
http://127.0.0.1:888...desvr=127.0.0.1:9997
127.0.0.1:8888/QCSPCChartGWT.html?gwt.codesvr=127.0.0.1:9997
Google
Add new data
X-Bar R (Mean Range)
Display JSON Script
Display Overall Stats

Project Name:Variable Control Chart (X-Bar R)
Part Name: Transmission Casing Bolt
Operator:J. Fenamore
Date: 7/04/2013

Machine: #11
Units: 0.0001 inch

Chart No.: 17
Part No.: 283501
Gage: #8645
Zero Equals: zero

Time	12:07	12:22	12:37	12:52	13:07	13:22	13:37	13:52	14:07	14:22	14:37	14:52
Average	28.87	31.99	32.11	28.53	27.76	31.30	28.58	28.12	34.48	33.23	30.20	30.02
RANGE	9.65	9.39	10.64	5.18	9.63	10.54	11.32	11.23	11.59	10.94	9.77	10.29
SUM	144.4	160.0	160.6	142.7	13		9	140.6	172.4	166.1	151.0	150.1
Cpk	0.151	0.280	0.313	0.301	0.		5	0.213	0.250	0.271	0.269	0.266
Cpm	0.286	0.323	0.314	0.351	0.		5	0.304	0.308	0.308	0.308	0.307
Ppk	0.175	0.288	0.332	0.299	0.		1	0.209	0.235	0.254	0.257	0.259
NO. INSP.	5	5	5	5				5	5	5	5	5
ALARM	- -	- -	- -	- -	-		-	- -	- -	- -	- -	- -
NOTES	-N-	-N-	-N-	-N-	-			-N-	-N-	-N-	-N-	-N-

4, 27.76
Sample #0 = 22.95
Sample #1 = 30.875
Sample #2 = 27.805
Sample #3 = 32.58
Sample #4 = 24.60
Average = 27.76
RANGE = 9.63
SUM = 138.79
Cpk = 0.25
Cpm = 0.33
Ppk = 0.24
NOTES -

User-Define Data Tooltips and Annotations

```
Events

   EnableJSONDataToolTip: boolean: false
```

Set EnableJSONDataToolTip true to enable user-defined tooltips. When a data point is clicked, the JSON data tooltip tries to vector out into the host HTML file. It looks for the Javascript method JSONChartMouseEvent with the following template:

```
function JSONChartMouseEvent( s )
{
    var jsonobj = JSON.parse(s);
    var jsonresult =
    { SPCChartMouseEventResult:
       {
          "Action": "TOOLTIP_ACTION_ANNOTATE",
          "Message": "Point #" + jsonobj.SPCChartMouseEvent.DataIndex,
          "TextBoxWidthChar": 40,
          "TextBoxHeightChar": 10
       }
    };
```

```
      var result=JSON.stringify(jsonresult);
      return result;
  }
```

It parses the JSON string passed into the function, and returns the JSON structure specifying the resulting tool tip message, action, and dimensions. The structure of the incoming JSON string is:

```
       "SPCChartMouseEvent": {
            "ChartType": integer
            "DataIndex": integer
            "SampleIntervalMean": double
            "SampleIntervalSigma": double
            "SampleValues": [double, double, double...]
            "TimeStampLong": long
            "TimeStampString": string
            "Notes":string
```

The way it actually appears as a JSON string, using actual values, is:

```
          {
          "SPCChartMouseEvent": {
               "ChartType": 0,
               "DataIndex": 7,
               "SampleIntervalMean": 22.0,
               "SampleIntervalSigma": 6,
               "SampleValues": [ 23.1, 14.1, 25.3, 21.3, 22.4],
               "TimeStampLong": 1371831729074,
               "TimeStampString": "07/21/2013 13:11:01",
               "Notes": "This is a note"
               }
         }
```

You can access individual fields of the record using standard java script dot notation, as demonstrated in the JSONChartMouseEvent example above: the sample interval index of the mouse click is found in jsonobj.SPCChartMouseEvent.DataIndex.

Example

An example is found in the SPCComplex.html example.

SPCChartMouseEvent

ChartType

The chart type of the current SPC chart.

DataIndex

The sample interval index of the selected point.

SampleIntervalMean
The sample mean of the current sample interval.

SampleIntervalSigma
The sample standard deviation of the current sample interval.

SampleValues
An array of the sample values of the sample interval

TimeStampLong
The time stamp of the sample interval as a long number, representing milliseconds.

TimeStampString
A string representation of the type sample of the sample interval.

Notes
A string containing any note attached to the current sample interval record.

The JSONChartMouseEvent should return a JSON structure which looks like:

```
SPCChartMouseEventResult
      Action: "TOOLTIP_ACTION_DIALOG",
      Message: string: "",
      TextBoxWidthPx: integer: 150,
      TextBoxHeightPx: integer: 150,
      TextBoxWidthChar: integer: 20,
      TextBoxHeightChar: integer: 10
```

The way it actually appears as a JSON string, using actual values, as created in the JSONChartMouseEvent above, is:

```
      { SPCChartMouseEventResult:
         {
            "Action": "TOOLTIP_ACTION_ANNOTATE",
            "Message": "Point #" + jsonobj.SPCChartMouseEvent.DataIndex,
            "TextBoxWidthChar": 40,
            "TextBoxHeightChar": 10
         }
      }
```

Example

An example is found in the SPCComplex.html example.

SPCChartMouseEventResult

Action

Select the action you want. You can annotate the data point with the message string using "TOOLTIP_ACTION_ANNOTATE", or you can open a pop-up dialog box displaying the message string using TOOLTIP_ACTION_DIALOG.

Message

The string to be displayed as either an annotation, or as a tooltip dialog box.

TextBoxWidthPx

Width of the dialog box in pixels. If you use TextBoxWidthPx and TextBoxHeightPx, do not use TextBoxWidthChart and TextBoxHeightChart.

TextBoxHeightPx

Height of the dialog box in pixels. If you use TextBoxWidthPx and TextBoxHeightPx, do not use TextBoxWidthChart and TextBoxHeightChart.

TextBoxWidthChar

Width of the dialog box in characters (approximate because of proportional character spacing). If you use TextBoxWidthChar and TextBoxHeightChar, do not use TextBoxWidthPx and TextBoxHeightPx.

TextBoxHeightChar

Height of the dialog box in characters. If you use TextBoxWidthChar and TextBoxHeightChar, do not use TextBoxWidthPx and TextBoxHeightPx.

Enhanced Annotations

Enhanced Annotations are a feature of QCSPCChart+

Annotations can be added in two places. The first is in the SampleIntervalRecords block. The SampleIntervalRecords block is an array containing the sample interval data, one array element for each sample interval. Inside any of the array elements you can insert an Annotation block which defines the annotation to be displayed at that sample interval. For example, the sample interval at

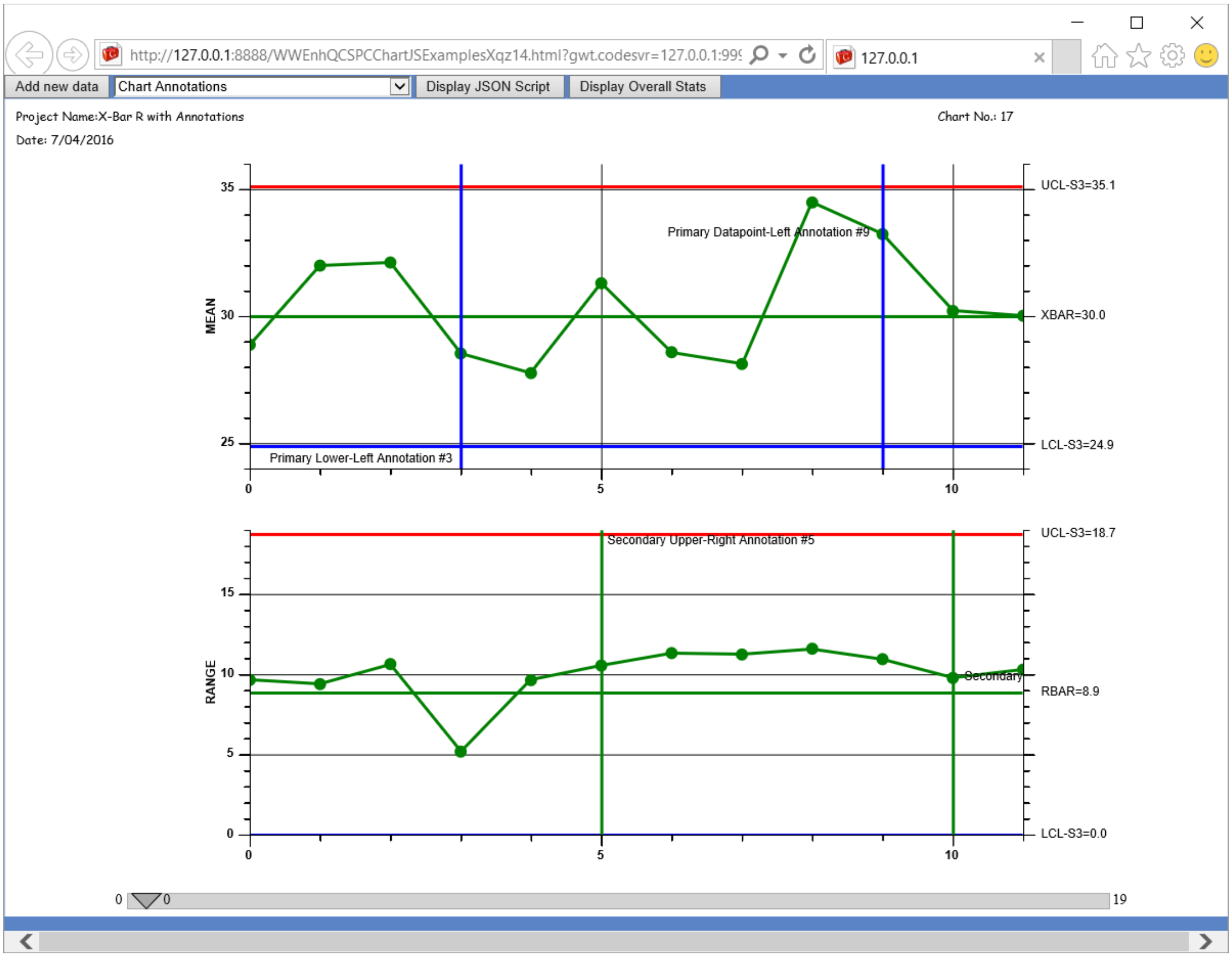
BatchCount 17 includes an annotation block, while those at BatchCount 16 and BatchCount 18 do not. This example was extracted from the chartdefEnhExamples.js file, script TimeXBarRAnnotations.

```
.
.
.
         {
          "SampleValues": [
            35.34918665138229,
            33.60402762568151,
            29.69203231293511,
            24.756849726860697,
            27.34428622213201
          ],
          "BatchCount": 16,
          "TimeStamp": 1371845229074,
          "Note": ""
        },
        {
          "SampleValues": [
            33.727605556103605,
            27.50821643159519,
            33.61161600390977,
            34.36241767438651,
            27.55491000970097
          ],
          "BatchCount": 17,
          "TimeStamp": 1371846129074,
          "Note": "",
          "Annotation": {
            "ChartPosition": "PRIMARY_CHART",
            "Text": "Primary Lower-Left Annotation #17",
            "TextColor": "RED",
            "Justify": "ANNOTATION_DATAPOINT_RIGHT",
            "LineColor": "BLUE",
            "LineWidth": 3
          }
        },
        {
          "SampleValues": [
            34.26565897109985,
            28.056929833331772,
            27.183128112177673,
            33.17121401058127,
            34.191900955504686
          ],
          "BatchCount": 18,
          "TimeStamp": 1371847029074,
          "Note": ""
        },
```

Or you can specify the annotations outside of the SampleIntervalRecords block, after all of the data has been entered, as in the following example. In this case the annotations are grouped as an array under the Annotations block. Since a specific sample interval index is referenced, DatapointIndex, sample interval data needs to have been already entered up to that point, otherwise the annotation is ignored since it is beyond the bounds of the data.

```
  "SampleData": {
      "SampleIntervalRecords": [
        {
          "SampleValues": [
            27.53131515148628,
            33.95771604022404,
            24.310097827061817,
            28.282642847792765,
            30.2908518818265
          ],
          "BatchCount": 0,
          "TimeStamp": 1371830829074,
          "Note": ""
        },
        {
          "SampleValues": [
            27.444285005240214,
            34.38930645615096,
            28.0203674441636,
            33.27153359969366,
            36.8305571558275
          ],
          "BatchCount": 1,
          "TimeStamp": 1371831729074,
          "Note": ""
        },
        {
          "SampleValues": [
            35.21321620109259,
            32.93940741018088,
            33.66485557976163,
            34.17314124609133,
            24.576683179863725
          ],
          "BatchCount": 2,
          "TimeStamp": 1371832629074,
          "Note": ""
        },
.
.
.
        {
          "SampleValues": [
            30.56585901649224,
            26.764807472584284,
            30.22766077749437,
            29.43260723522982,
            27.080310485264213
          ],
          "BatchCount": 19,
          "TimeStamp": 1371847929074,
          "Note": ""
        }
      ]
    },
    "Annotations": [
      {
        "ChartPosition": "PRIMARY_CHART",
        "Text": "Primary Lower-Left Annotation #3",
        "TextColor": "RED",
        "Justify": "ANNOTATION_LOWER_LEFT",
        "LineColor": "BLUE",
        "LineWidth": 3,
        "DatapointIndex": 3
      },
      {
        "ChartPosition": "SECONDARY_CHART",
        "Text": "Secondary Upper-Right Annotation #5",
        "TextColor": "RED",
        "Justify": "ANNOTATION_UPPER_RIGHT",
        "LineColor": "GREEN",
        "LineWidth": 3,
        "DatapointIndex": 5
      },
      {
        "ChartPosition": "PRIMARY_CHART",
        "Text": "Primary Datapoint-Left Annotation #9",
        "TextColor": "RED",
        "Justify": "ANNOTATION_DATAPOINT_LEFT",
        "LineColor": "BLUE",
        "LineWidth": 3,
        "DatapointIndex": 9
      },
      {
        "ChartPosition": "SECONDARY_CHART",
        "Text": "Secondary Datapoint-Right Annotation #10",
        "TextColor": "RED",
        "Justify": "ANNOTATION_DATAPOINT_RIGHT",
        "LineColor": "GREEN",
        "LineWidth": 3,
        "DatapointIndex": 10
      }
    ],
```

Below is a typical chart with annotations.


http://127.0.0.1:8888/WWEnhQCSPCChartJSExamplesXqz14.html?gwt.codesvr=127.0.0.1:999
127.0.0.1
Add new data
Chart Annotations
Display JSON Script
Display Overall Stats
Project Name:X-Bar R with Annotations
Date: 7/04/2016
Chart No.: 17
MEAN
35
30
25
UCL-S3=35.1
XBAR=30.0
LCL-S3=24.9
Primary Datapoint-Left Annotation #9
Primary Lower-Left Annotation #3
0
5
10
RANGE
15
10
5
0
UCL-S3=18.7
RBAR=8.9
LCL-S3=0.0
Secondary Upper-Right Annotation #5
Secondary
0
0
19

The Annotation block has the following options

Annotation

ChartPosition

Specify "PRIMARY_CHART", "SECONDARY_CHART", "THIRD_CHART", or "ALL_CHARTS"

Text

Specify the annotation text - "Shutdown fault on line #24" for example

TextColor

Color of text – Use one of the SPC Chart color constants, "BLACK" for example.

Justify

Text Justification with respect to vertical Line - There are six justification options:

"ANNOTATION_UPPER_RIGHT", "ANNOTATION_UPPER_LEFT",
"ANNOTATION_LOWER_RIGHT", "ANNOTATION_LOWER_LEFT",
"ANNOTATION_DATAPOINT_RIGHT", "ANNOTATION_DATAPOINT_LEFT",

LineColor

The Color of the vertical line - Use one of the SPC Chart color constants, "BLUE" for example

LineWidth

The line width of the vertical line - 3 for example

DatapointIndex

The 0-based index of where the annotation goes in the sample data.

Only needed when the annotations are specified outside of the SampleIntervalRecords block, otherwise the DatapointIndex is implicit.

15. JSNI Calls into the QCSPCChart Library

Direct Javascript calls, from handwritten Javascript code calling internal library functions of the application is not supported, except for a limited number of exported functions. This is because the Javascript code generated by the GWT compiler, is highly optimized, compressed, and obfuscated, and it also undergoes a major structural change as the OOP source code (Java) is translated into non-OOP code (Javascript). There are ways around this using a feature of GWT call JSNI (JavaScript Native Interface) and we utilize that feature in a few critical areas. But in general, the programmer (i.e. you), will not be calling our Javascript library functions directly.

We call these *push* functions because our QCSPCChart library pushes, or adds, these functions automatically into your base HTML file. So you can call them exactly as if they were part of the Javascript code in your file.

The exported functions are divided into four groups: functions which create or modify a chart, data simulation methods, display of JSON script, and functions which return a JSON structure of data to the calling Javascript code in the HTML base file.

These routines require a medium to advanced knowledge of Javascript and JSON. You may need to refer to a text books and on-line sources about Javascript and JSON in order to make full use of these functions.

http://www.w3schools.com/js/ - Javascript Tutorial

http://www.json.org/ - Introducing JSON

http://www.json.org/js.html – JSON in Javascript

Chart Creation/Modification Functions

pushJSONChartCreate

This method can be used to create a new chart. Pass in a chart defining JSON script and if properly formatted, the resulting chart will display on the web page. The initial chart can contain data; just include the data in the SampleData property of the defining JSON script. Or the data can be added later by calling pushJSONChartUpdate.

The Javascript template of pushJSONChartCreate is:

```
pushJSONChartCreate(String jsonstring)
```

where jsonstring is a properly formatted JSON string.

In use, it would look like:

```
   function processCreateChartClick( )
   {
     pushJSONChartCreate( JSON.stringify(TimeXBarR));
   }
```

In this case, TimeXBarR is a Javascript JSON variable. A minimal example is:

```
var TimeXBarR=
 {
        "SPCChart": {   .

               "InitChartProperties": {
                    "SPCChartType": "MEAN_RANGE_CHART",
                    "ChartMode": "Time",
                    "NumSamplesPerSubgroup": 5,
                    "NumDatapointsInView": 12,
                    "TimeIncrementMinutes": 15
           },
       .
       .
       .
}
```

Example

The JSON.stringify call converts the variable into its JSON string equivalent. You will find an example of pushJSONChartCreate in the SPCExampleScripts.html example. It is called in the displayChart function, where it displays the chart selected from the drop down list.

PushJSONChartCreateDiv

This method almost identical to the PushJSONChartCreate method. The difference is that you can specify a DIV in your HTML page to place the chart it.

So, in your calling HTML page you define a place holder for your chart.

```
  <body>

    <!-- OPTIONAL: include this if you want history support -->
    <iframe src="javascript:''" id="__gwt_historyFrame" tabIndex='-1'
style="position:absolute;width:0;height:0;border:0"></iframe>

    <!-- RECOMMENDED if your web app will not function without JavaScript enabled
-->
    <noscript>
      <div style="width: 22em; position: absolute; left: 50%; margin-left: -11em;
color: red; background-color: white; border: 1px solid red; padding: 4px; font-
family: sans-serif">
        Your web browser must have JavaScript enabled
        in order for this application to display correctly.
      </div>
    </noscript>

      <div id="contentContainer">
      </div>

  <INPUT TYPE="button" NAME="addData" VALUE="Add new data"
onClick="addDataClick();">
  <INPUT TYPE="button" NAME="displayJSONScript" VALUE="Display JSON Script "
onClick="processJSONScriptClick();">
  <INPUT TYPE="button" NAME="displayJSONOverallStatistics" VALUE="Display Overall
Stats " onClick="processJSONOverallStatisticsClick();">


  </body>
```

Then, in response to some event, a button click for example, you can create the chart on-the-fly, and place it in the "contentContainer" DIV.

```
    function processJSONScriptClick( )
   {
       var s = JSON.stringify(TimeXBarR,undefined,2);
       pushJSONChartCreateDiv(s,  "contentContainer");

   }
```

pushJSONChartUpdate

This method is very similar to the pushJSONChartCreate method.. The difference is that the pushJSONChartUpdate function requires that a chart already be created. It supplies data, or modified parameters, to the current chart. Using this method, you can add data to a chart one sample interval at a time, or a block of sample intervals at a time. It should NOT include an **InitChartProperties** block, because that is a chart creation block, and this function is just used to update the existing chart.

The Javascript template of pushJSONChartUpdate is:

```
pushJSONChartUpdate(String jsonstring)
```

where jsonstring is a properly formatted JSON string.

In use, it would look like:

```
function processUpdateChartClick( )
{
  pushJSONChartUpdate( JSON.stringify(NewChartData));
}
```

In this case, NewChartData is a Javascript JSON variable.

```
var NewChartData=
 {
       "SPCChart": {   .

             "SampleData": {
                       "SampleIntervalRecords": [
                       {
                           "SampleValues": [
                               27.53131515148628,
                               33.95771604022404,
                               24.310097827061817,
                               28.282642847792765,
                               30.2908518818265
                           ],
                           "BatchCount": 0,
                           "TimeStamp": 1371830829074,
                           "Note": ""
                       },
                       {
                           "SampleValues": [
                               27.444285005240214,
                               34.38930645615096,
                               28.0203674441636,
                               33.27153359969366,
                               36.8305571558275
                           ],
                           "BatchCount": 1,
                           "TimeStamp": 1371831729074,
                           "Note": ""
                       },
                       {
                           "SampleValues": [
                               35.21321620109259,
                               32.93940741018088,
                               33.66485557976163,
                               34.17314124609133,
                               24.576683179863725
                           ],
                           "BatchCount": 2,
                           "TimeStamp": 1371832629074,
                           "Note": ""
                       },
                    .
                    .
                    .
                    ]
                 }
             }
}
```

Example

The JSON.stringify call converts the variable into its JSON string equivalent. The SPCMediumSimple.html page has an example of updating a chart using pushJSONChartUpdate.

Data Simulation Functions

pushGetSimulateDataJSON

The easiest way to debug a new chart is to use simulated data. The pushGetSimulatedDataJSON function returns a properly formatted string of data in JSON format. So you can make a call in your HTML Javascript code, get a JSON string of formatted data back, and then turn around and use the JSON string in a call to our pushJSONChartUpdate method.

The format of the pushGetSimulateDataJSON function is:

```
String  pushGetSimulateDataJSON(int count, double mean, double range)
```

where

count

Number of sample intervals to simulate.

mean

The desired mean of the simulated data.

range

The desired range of the simulated data.

The function already knows how many samples per subgroup are defined for the chart.

Below is an example of calling pushGetSimulateDataJSON, and then using the resulting JSON formatted string in a subsequent call to pushJSONChartUpdate.

```
   function processClick( )
   {
     var jsonstring = pushGetSimulateDataJSON(40, 35, 6);
     pushJSONChartUpdate(0, jsonstring);
   }
```

Note that since pushGetSimulateDataJSON returns a JSON string, it is already a string, and does not need to be converted to a string using JSON.stringify. You can, if you want, convert the JSON string into a Javascript JSON object using JSON.parse. Once you have the Javascript JSON object version of the data, you can read, or modify the data under program control. Since it is a Javascript object, you do need to call JSON.stringify before using it in the call to pushJSONChartUpdate. If you want to format the string, you can use a version of JSON,stringify which formats the resulting JSON string.

```
   function processClick( )
   {
     var jsonstring = pushGetSimulateDataJSON(40, 35, 6);
     var jsonobj = JSON.parse(jsonstring);

     var formattedjsonstring = JSON.stringify(jsonobj,undefined,2);

     pushJSONChartUpdate(formattedjsonstring);
   }
```

pushSimulateDataUpdate

This method directly updates the chart with simulated data. There are no intermediate stops to and from a JSON formatted string, as was demonstrated in the pushGetSimulateDataJSON description.

The format of the pushSimulateDataUpdate function is:

```
void  pushSimulateDataUpdate(int count, double mean, double range)
```

where

count
Number of sample intervals to simulate.

mean
The desired mean of the simulated data.

range
The desired range of the simulated data.

The function already knows how many samples per subgroup are defined for the chart.

Below is a simple example which has the same affect on the chart as the examples for the pushGetSimulateDataJSON function.

```
   function processClick( )
   {
     pushSimulateDataUpdate(40, 35, 6);
   }
```

pushSimulateDataUpdatePercentChange

This method directly updates the chart with simulated data. There are no intermediate stops to and from a JSON formatted string, as was demonstrated in the pushGetSimulateDataJSON description. It calculates the current mean and sigma of the data, and simulates new data which reflects a percentage change from those values.

The format of the pushSimulateDataUpdate function is:

```
void  pushSimulateDataUpdate(int count, double meanpercentchange, double
rangepercentchange)
```

where

count
Number of sample intervals to simulate.

meanpercentchange
The desired change in the mean of the simulated data.

rangepercentchange
The desired change in the range of the simulated data.

The function already knows how many samples per subgroup are defined for the chart.

Below is a simple example which has the same affect on the chart as the examples for the pushGetSimulateDataJSON function.

```
   function processAddDataClick( )
   {
      pushSimulateDataUpdatePercentChange
   }
```

Example
An example is found in the SPCComplex.html example page.

Display of JSON Script

pushDisplayJSONScript

The pushDisplayJSONScript function is a useful utility. It displays a formatted JSON script in a pop-up window in the browser.

The format of the pushDisplayJSONScript function is:

```
void  pushSimulateDataUpdate(String jsonstring)
```

where:

jsonstring
A JSON formatted string to display in the popup window.

```
    function processJSONScriptClick( )
   {
     var s = JSON.stringify(TimeXBarR,undefined,2);

     pushDisplayJSONScript(s);

   }
```

where TimeXBar, the chart defining JSON script, is converted to a JSON string, using JSON.stringify, and then displayed in a pop-up window using pushDisplayJSONScript. The call to JSON.stringify, with three parameters, is a version which formats the JSON string with indentation, making it easier to read. In this case the indentation is set to level 2.

Example
The pushDisplayJSONScript function is used in the SPCMediumSimple and SPCExampleScripts examples.

Data Retrieval Functions

pushGetJSONSampleIntervalData

Programmers always want to retrieve values not part of their original input, but calculated internally by the software. The pushGetJSONSampleIntervalData retrieves data values associated with a given sample interval. It returns the sample interval data in a JSON structure, which can be parsed into a Javascript object. The format of the JSON structure is:

```
{
  "SPCSampleIntervalData": {
    "DataIndex": integer,
    "TimeStampLong": integer
    "TimeStampString": string,
    "NumberOfSamples": integer,
    "SampleData": double [],
    "Mean": double,
    "Minimum": double,
    "Maximum": double,
    "Range": double,
    "StandardDevation": double,
    "Median": double,
    "Variance":double,
    "Note": string
    "PrimaryChartAlarmState": integer,
    "SecondaryChartAlarmState": integer,
    "PrimaryChartAlarmMessage": string,
    "SecondaryChartAlarmMessage": string,,
    "ProcessPerformance": {
      "Cpk": double,
      "Cpm":double,
      "Ppk": double,
       .
       .
       .
    }
  }
}
```

In actual use, a filled-out SPCSampleIntervalData would look like:

```
{
  "SPCSampleIntervalData": {
    "DataIndex": 52,
    "TimeStampLong": 1371877629074,
    "TimeStampString": "6/22/2013 01:07:09",
    "NumberOfSamples": 5,
    "SampleData": [
      37.86739857994523,
      37.38247376256557,
      33.93501030715761,
      31.009898186882122,
      38.61221167576862
    ],
    "Mean": 35.76139850246383,
    "Minimum": 31.009898186882122,
    "Maximum": 38.61221167576862,
    "Range": 7.602313488886498,
    "StandardDevation": 3.2055696125369533,
    "Median": 37.38247376256557,
    "Variance": 10.275676540820314,
    "Note": "",
    "PrimaryChartAlarmState": 2,
    "SecondaryChartAlarmState": 2,
    "PrimaryChartAlarmMessage": "6/22/2013 01:07:09 \nPrimary chart: Basic Rule #2
violation 1 of 1 greater than 3-sigma=35.090\nCurrent Value=35.761",
    "SecondaryChartAlarmMessage": "",
    "ProcessPerformance": {
      "Cpk": 0.2070010871064765,
      "Cpm": 0.36491231082172015,
      "Ppk": 0.14704411677834697
    }
  }
}
```

The process performance indices will include all process performance indices added to the chart in the charts ProcessCapabilitySetup section of the chart SPCChart.TableSetup.ProcessCapabilitySetup block, as seen below.

```
        "ProcessCapabilitySetup": {
          "LSLValue": 27,
          "USLValue": 35,
          "EnableCPK": true,
          "EnableCPM": true,
          "EnablePPK": true
        }
```

Once you retrieve the sample interval data, you can turn it into a Javascript object by calling JSON.parse. From there, you can access individual fields as you would any Javascript record structure. In the example below, the PrimaryChartAlarmMessage is retrieved using the expression jsonobj.SPCSampleIntervalData.PrimaryChartAlarmMessage.

```
   function processJSONSampleIntervalDataClick( )
   {
      var s = pushGetJSONSampleIntervalData(52);

      var jsonobj = JSON.parse(s);

      var s2 = jsonobj.SPCSampleIntervalData.PrimaryChartAlarmMessage;

      alert(s2.toString());

   }
```

Example

The pushGetJSONSampleIntervalData function is used in the SPCMediumComplex example.

pushGetJSONOverallStatistics

The pushGetJSONOverallStatistics returns statistics calculated using all of the data, rather than a single sample interval of data. It returns the overall statistics data in a JSON structure, which can be parsed into a Javascript object. The format of the JSON structure is:

```
{
  "SPCOverallStatistics": {
    "ChartType":string,
    "NumberOfCharts": int,
    "NumberOfSampleIntervals":int,
    "PrimaryChart": {
      "Mean": double,
      "Minimum": double,
      "Maximum": double,
      "Range": double,
      "StandardDevation": double,
      "Median":  double,
      "Variance": double,
      "UCL3": double,
      "Target":  double,
      "LCL3": double
    },
    "SecondaryChart": {
      "Mean":  double,
      "Minimum":  double,
      "Maximum":  double,
      "Range":  double,
      "StandardDevation": double,
      "Median":  double,
      "Variance":  double,
      "UCL3": double,
      "Target":  double,
      "LCL3": double
    },
    "ProcessPerformance": {
      "Cpk": double,
      "Cpm":double,
      "Ppk": double,
       .
       .
       .

    }
  }
}
```

In actual use, a filled-out SPCOverallStatistics would look like:

```
{
  "SPCOverallStatistics": {
    "ChartType": "MEAN_RANGE_CHART",
    "NumberOfCharts": 2,
    "NumberOfSampleIntervals": 60,
    "PrimaryChart": {
      "Mean": 29.983142222229475,
      "Minimum": 25.482478927553114,
      "Maximum": 34.476685371240386,
      "Range": 8.994206443687272,
      "StandardDeviation": 2.336294386647534,
      "Median": 30.082618566756053,
      "Variance": 5.458271461080777,
      "UCL3": 35.08998767593801,
      "Target": 29.983142222229475,
      "LCL3": 24.876296768520937
    },
    "SecondaryChart": {
      "Mean": 8.850685361713235,
      "Minimum": 3.801051543907956,
      "Maximum": 11.593146884947827,
      "Range": 7.792095341039872,
      "StandardDevation": 2.314209860374365,
      "Median": 9.640872833575247,
      "Variance": 5.355567277853939,
      "UCL3": 18.710348854661777,
      "Target": 8.850685361713235,
      "LCL3": 0
    },
    "ProcessPerformance": {
      "Cpk": 0.17125640122162375,
      "Cpm": 0.3385273201944916,
      "Ppk": 0.13191522282471005
    }
  }
}
```

The process performance indices will include all process performance indices added to the chart in the charts ProcessCapabilitySetup section of the chart SPCChart.TableSetup.ProcessCapabilitySetup block, as seen below.

```
        "ProcessCapabilitySetup": {
          "LSLValue": 27,
          "USLValue": 35,
          "EnableCPK": true,
          "EnableCPM": true,
          "EnablePPK": true
        }
```

Example

The pushGetJSONOverallStatistics function is used in the SPCMediumSimple and SPCMediumComplex examples.

16. CSS Style Sheets

A limited number of style sheet properties are supported in the software. These include the background color of the charts, the default font-family, and margin properties.

You will find a QCSPCChartGWT.css file in the root of the war folder. Inside are several style definitions which can be used to define a few styles which are used by the software.

Background Color

Normally, the background colors used for the charts are set in the charts JSON file, using the GraphBackground and PlotBackground properties.

```
"PrimaryChartSetup": {

   "GraphBackground": {
          "FillColor": "GREEN",
         "BackgroundMode": "SIMPLECOLORMODE"
   },

   "PlotBackground": {
          "FillColor": "BROWN",
          "BackgroundMode": "SIMPLECOLORMODE"
   }
}

"SecondaryChartSetup": {
   "PlotBackground": {
          "FillColor": "BROWN",
          "BackgroundMode": "SIMPLECOLORMODE"
   }
}
```

There is no need to process the GraphBackground property in the SecondaryChartSetup, because the GraphBackground of the PrimaryChartSetup also applies to the SecondaryChartSetup.

The GraphBackground and PlotBackground have default values of "WHITE", and that color is used for the charts background if no value is explicitly assigned.

If you prefer, you can use the background color of the parent HTML page. You do this by disabling the GraphBackground and PlotBackground objects using associated Enable property.

```
"PrimaryChartSetup": {
       "GraphBackground": {
          "Enable": false
       },
       "PlotBackground": {
          "Enable": false
       }

},
"SecondaryChartSetup": {
   "PlotBackground": {
      "Enable": false
   }
},
```

If you do this for a chart, and since the HTML5 Canvas object we use for drawing the charts has a transparent background, the HTML5 page will show through. Therefore the background color of the HTML parent page will become the background for the SPC charts.

One way to specify the background color for the HTML page is to define a body css style tag in the QCSPCChartGWT.css file, and set the background-color property. If you want the color to override any other styles properties for the page, add the !important flag as seen below.

```
body {background-color:#5a84c5 !important;}
```

You will find this line commented out in the QCSPCChartGWT.css file.

Default Font Family

The default font family can be set using the font-family css property of the .mainCanvas style, located in the QCSPCChartGWT.css file.

```
.mainCanvas {

   font-family:"Times New Roman", Times, serif;

}
```

This property sets the default font for the charts and table of the SPC charts. If you do not use the JSON properties for setting default properties, then the .mainCanvas.font-family property will be in affect. However, you can override the .mainCanvas.font-family property using JSON properties we also provide.

```
           "StaticProperties": {

               "DefaultFontName": "Arial, sans-serif",
               "DefaultTableFont": {
                   "Name": "'Comic Sans MS', cursive, sans-serif",
                   "Size": 12,
                   "Style": "Plain"
               }

           },
```

In this case, even though the .mainCanvas.font-family property is set to "Times New Roman" in the QCSPCChartGWT.css file, it is overridden by the DefaultFontName, and DefaultTableFont properties of the chart defining JSON script.

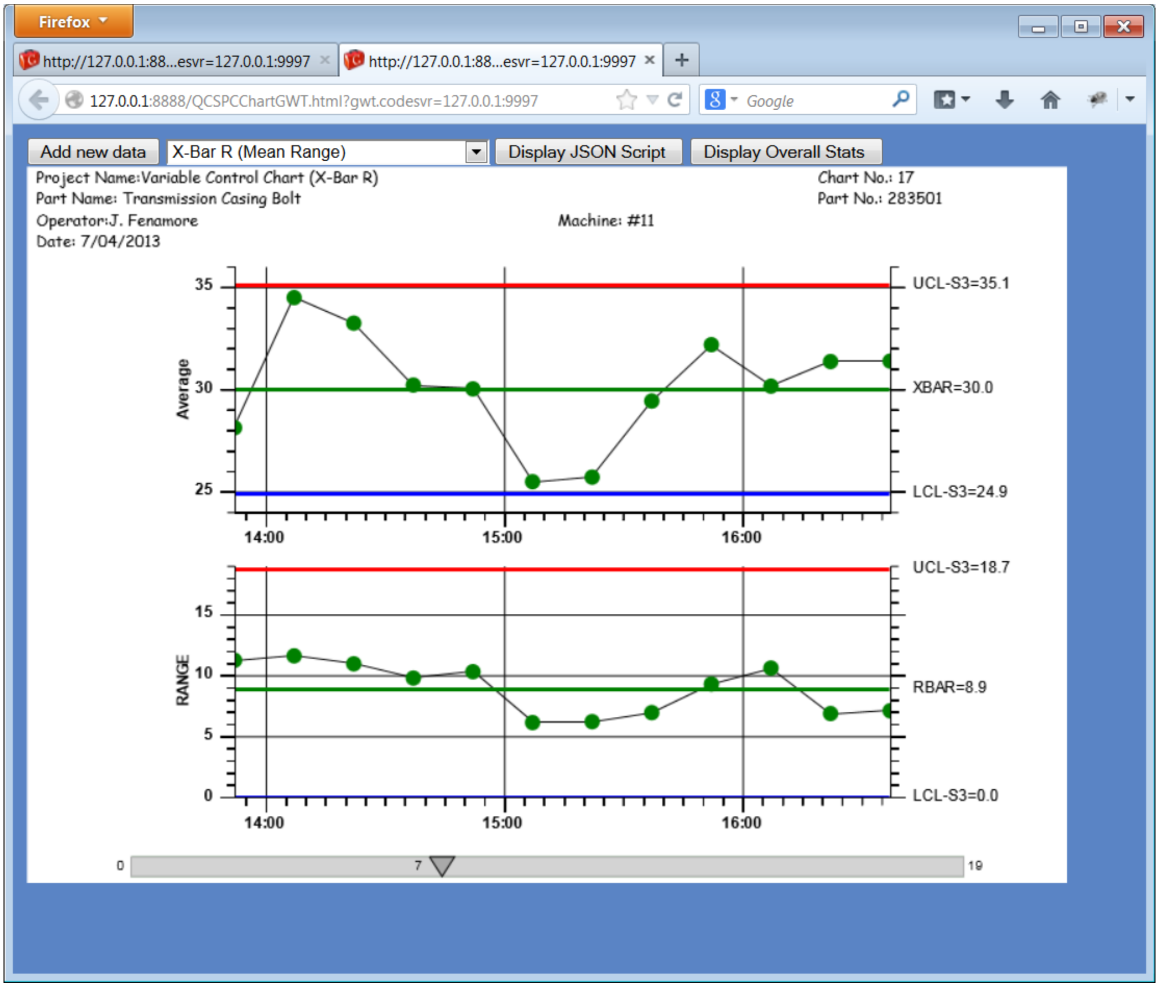
Chart Position

You can offset the postion of the chart within the HTML page using margin propeties in the .verticalPanel style located in the QCSPCChartGWT.css file. The HTML5 canvas we use for drawing is placed in a VerticalPanel object. The VerticalPanel is in turned place in the Iframe of the HTML page used by GWT.

The default margin property value is set to auto.

```
.verticalPanel {
   width: 100%;
   height: 100%;
   margin: auto;
}
```

This produces a chart something like this:



Firefox
http://127.0.0.1:88...esvr=127.0.0.1:9997
http://127.0.0.1:88...esvr=127.0.0.1:9997
127.0.0.1:8888/QCSPCChartGWT.html?gwt.codesvr=127.0.0.1:9997
Google
Add new data
X-Bar R (Mean Range)
Display JSON Script
Display Overall Stats

Project Name:Variable Control Chart (X-Bar R)
Part Name: Transmission Casing Bolt
Operator:J. Fenamore
Date: 7/04/2013
Machine: #11
Chart No.: 17
Part No.: 283501

This sizes the vertical panel to SPC Chart canvas object placed inside. If you want a custom margin around the SPC Chart, specifiy individual margin values.

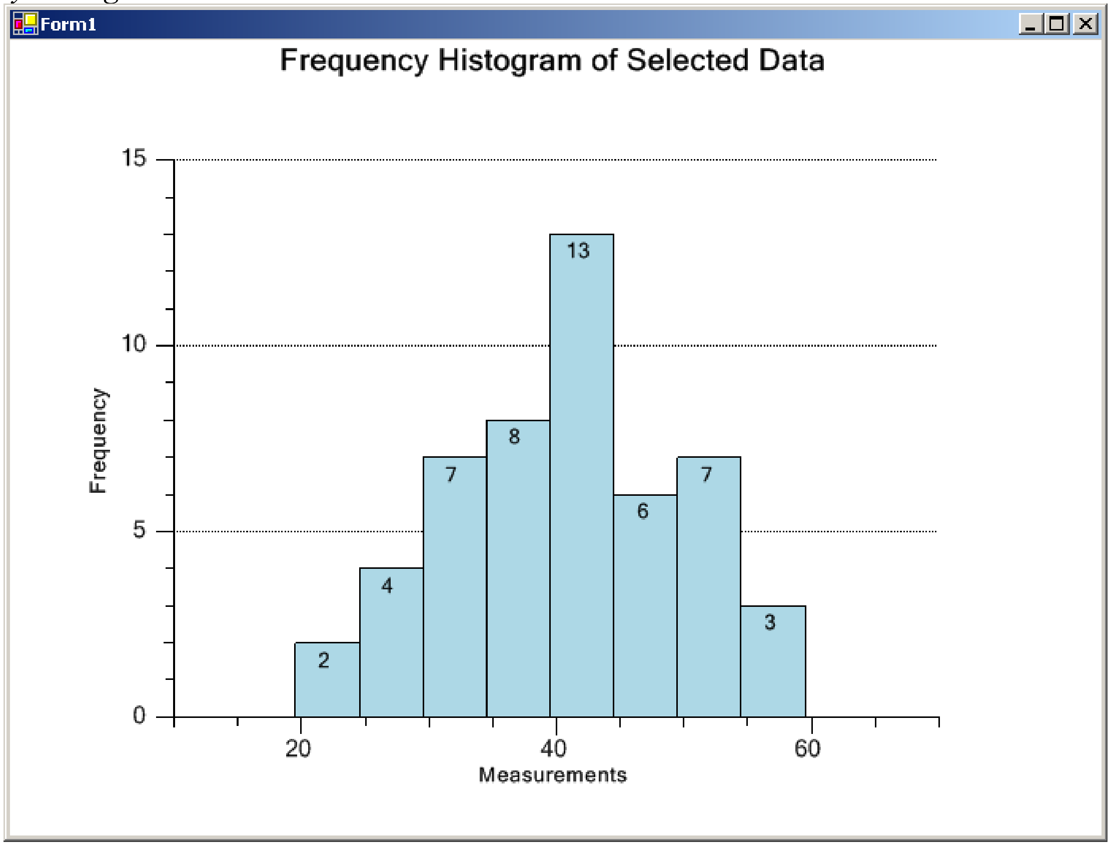
```
.verticalPanel {
   width: 100%;
   height: 100%;

    margin-top:100px;
    margin-left:50px;

}
```

17. Frequency Histogram

Frequency Histogram Chart

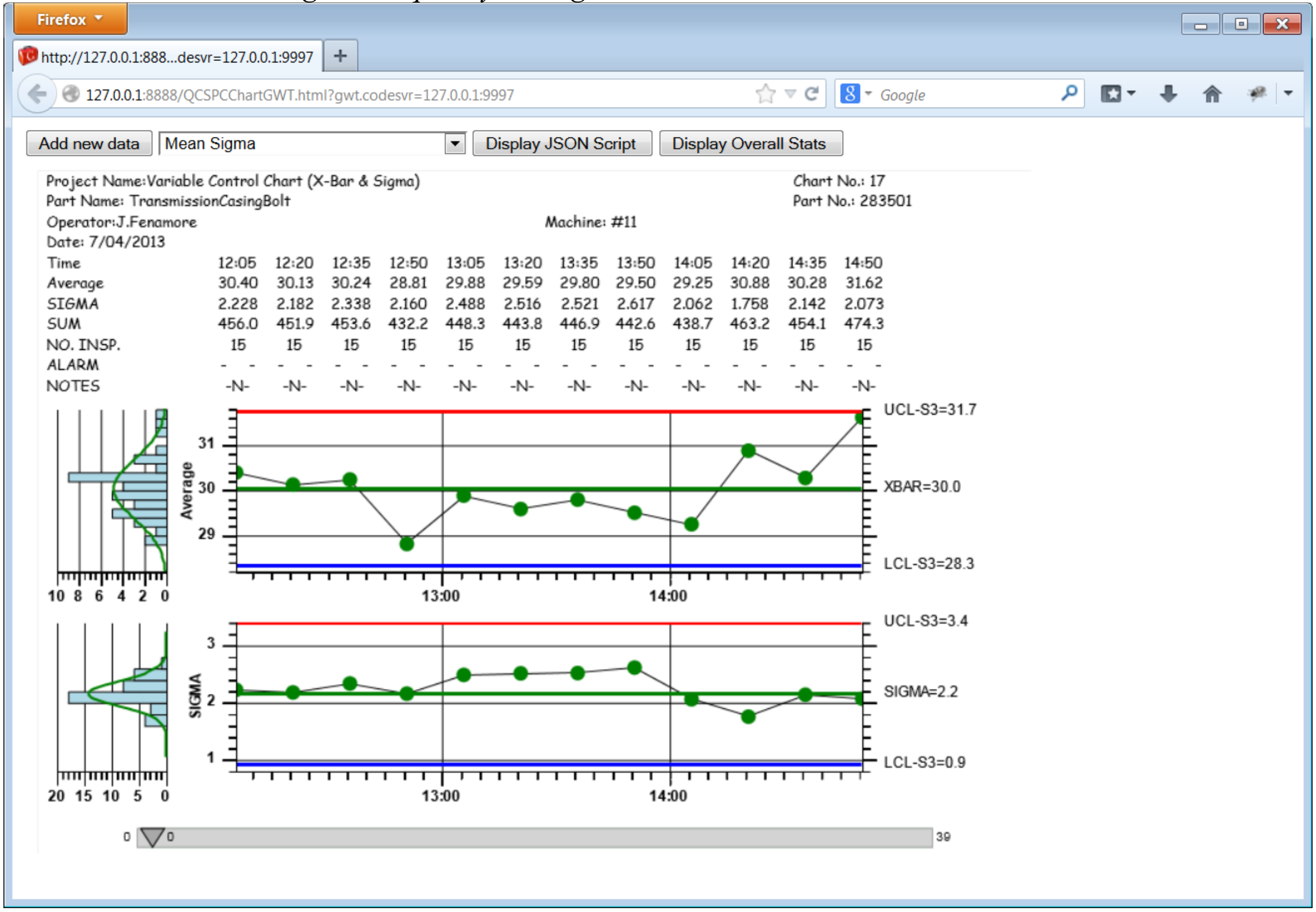
An SPC control chart will allow you to track the trend of critical variables in a production environment. It is important that the production engineer understand whether or not changes or variation in the critical variables are natural variations due to the tolerances inherent to the production machinery, 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



Project Name:Variable Control Chart (X-Bar & Sigma)
Part Name: TransmissionCasingBolt
Operator:J.Fenamore
Date: 7/04/2013

Chart No.: 17
Part No.: 283501
Machine: #11

Time	12:05	12:20	12:35	12:50	13:05	13:20	13:35	13:50	14:05	14:20	14:35	14:50
Average	30.40	30.13	30.24	28.81	29.88	29.59	29.80	29.50	29.25	30.88	30.28	31.62
SIGMA	2.228	2.182	2.338	2.160	2.488	2.516	2.521	2.617	2.062	1.758	2.142	2.073
SUM	456.0	451.9	453.6	432.2	448.3	443.8	446.9	442.6	438.7	463.2	454.1	474.3
NO. INSP.	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-

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.FrequencyHistogramPlot** example program.

```
 "FrequencyHistogram": {
      "ChartSetup": {
         "HistogramPlot": {
            "LineColor": "BLACK",
            "LineWidth": 1,
            "FillColor": "GREEN"
         },
         "ControlLines": [
           {
             "LimitValue": 21,
             "LineColor": "BLUE",
             "LineWidth": 3
           },
            {
             "LimitValue": 58,
             "LineColor": "RED",
             "LineWidth": 3
          }
        ],
      "NormalCurveLine": {
         "Enable": true,
         "LineColor": "YELLOW",
         "LineWidth": 3
      }
      },
      "FrequencyHistogramData": {
      "SampleValues": [
                         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
                      ],
      "FrequencyBins": [
                      19.5,  24.5,  29.5, 34.5, 39.5,
                      44.5, 49.5,   54.5, 59.5
                      ]
      },
"Methods": {
   "RebuildUsingCurrentData": true
   }
  }
```

Supplying Data to A Frequency Histogram Chart

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.

FrequencyHistogramData

```
FrequencyHistogramData
   SampleValues [double, double, ...]
   FrequencyBins [double, double, ...]
```

Initializes the histogram frequency bin limits, and the data values for the histogram.

]

Parameters

FrequencyBins

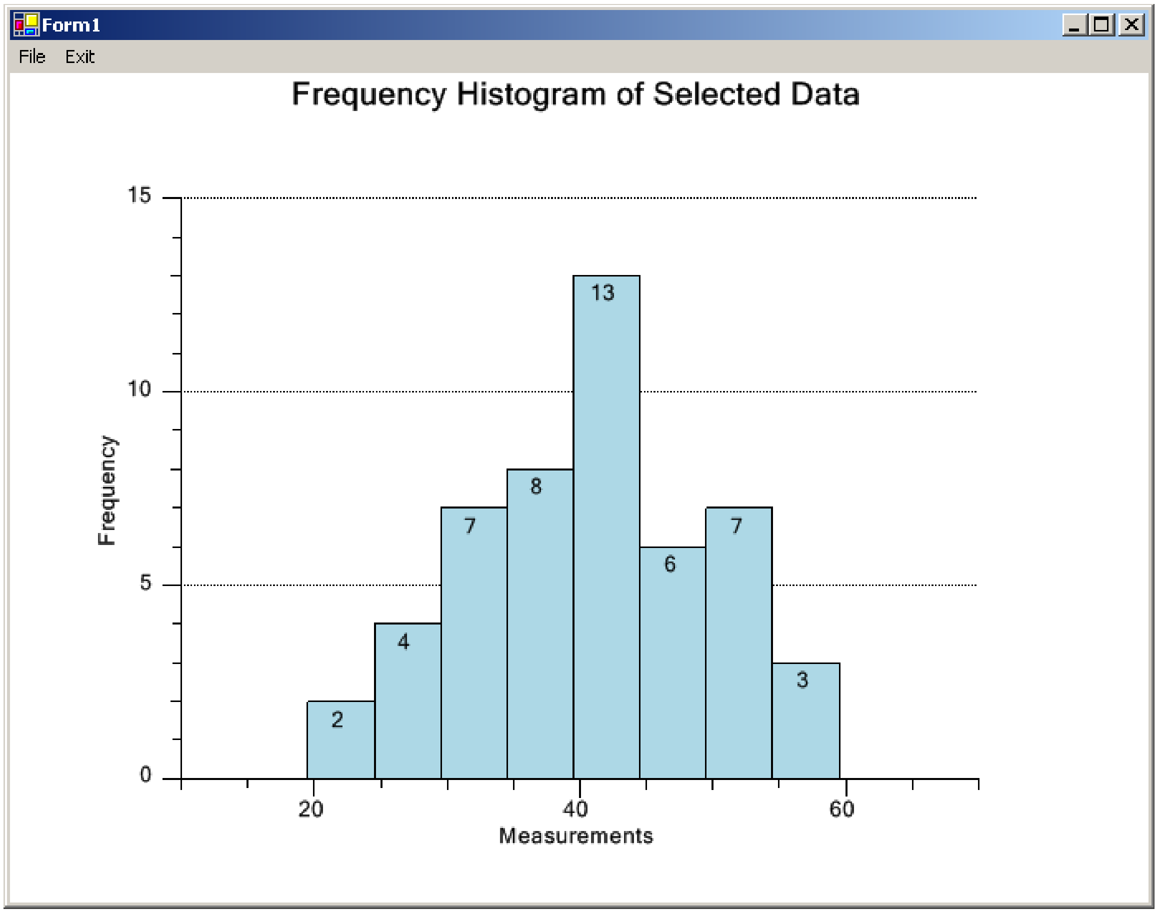
The frequency limits of the histogram bins.

SampleValues

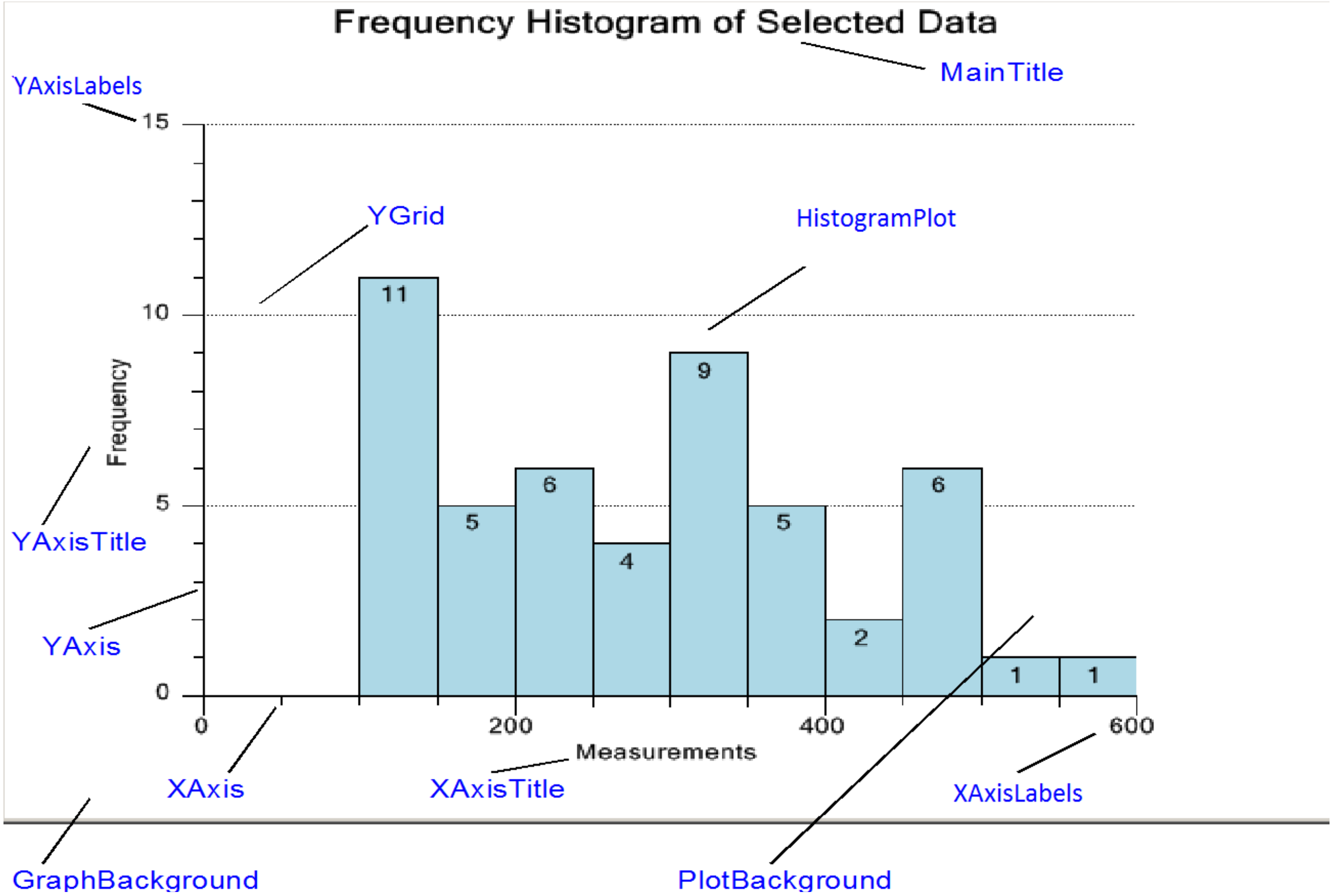
An array the values that are counted with respect to the frequency bins.

The image below uses the following data:

```
"SampleValues": [
                   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
                ],
"FrequencyBins": [
                   19.5,  24.5,  29.5, 34.5, 39.5,
                   44.5, 49.5,   54.5, 59.5
                ]
```


Form1
File
Exit
Frequency Histogram of Selected Data
Frequency
15
10
5
0
2
4
7
8
13
6
7
3
20
40
60
Measurements

Changing Default Characteristics of the Chart



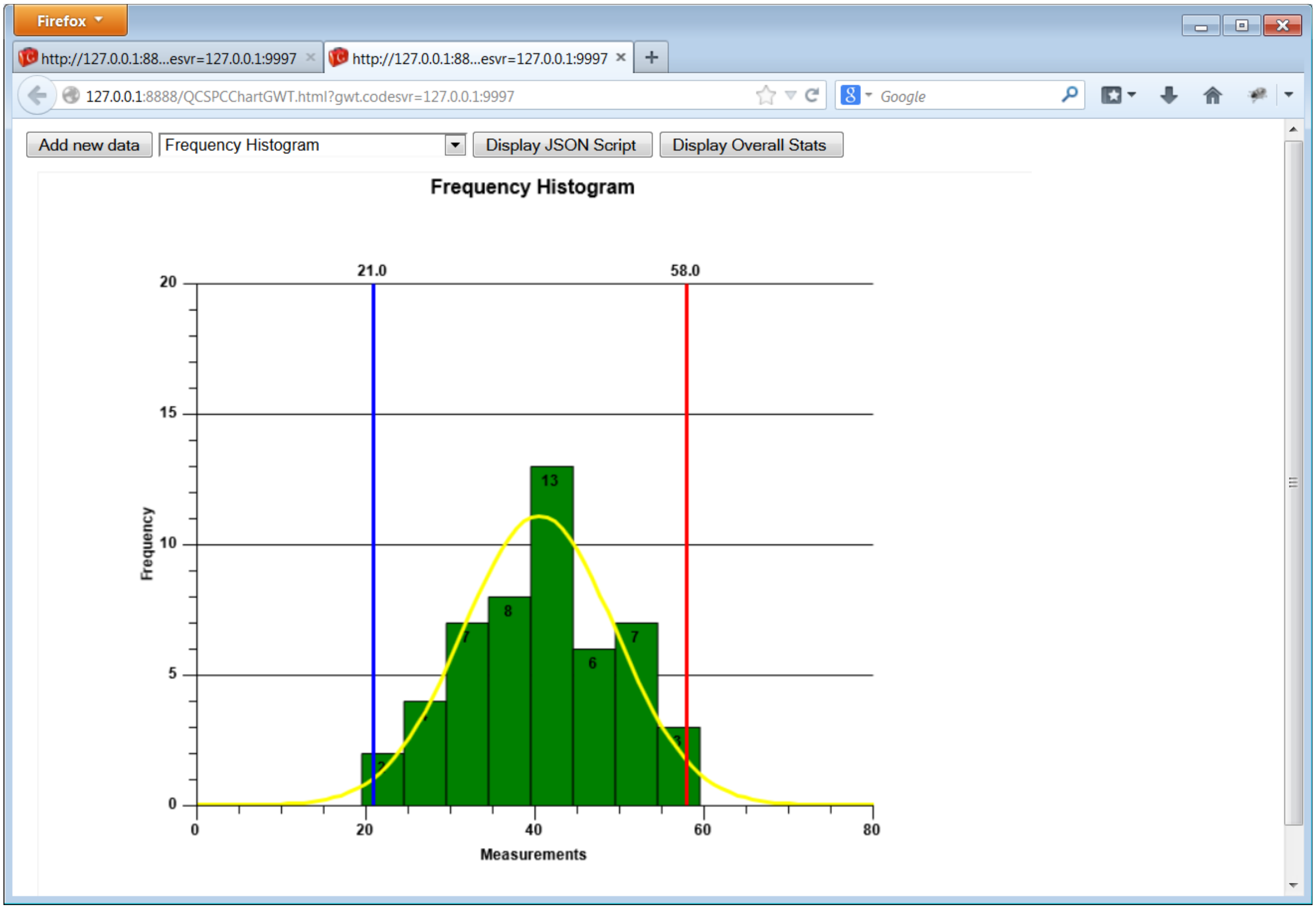
A **FrequencyHistogramChart**object has one distinct graph with its own set of properties. Once the graph is initialized (using the **InitFrequencyHistogram**, or one of the **FrequencyHistogramChart** constructors**)**, you can modify the default characteristics of each graph using these properties. For example, you can change the color of y-axis, and the y-axis labels using the LineColor property of those objects.

```
"ChartSetup": {
          "HistogramPlot": {
             "LineColor": "BLACK",
             "LineWidth": 1,
             "FillColor": "GREEN"
           },
           "YAxis": {
                "LineColor": "GREEN",
                "LineWidth": 3
          }
          "YAxisLabels": {
             "TextColor" = "DARKMAGENTA"
          }
}
```

Adding Control Lines and Normal Curve to Histogram Plot

You can add control limit lines, and a normal distribution curve to the frequency histogram. The control limit lines will be parallel to the frequency axis. 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



The block in red below shows how to add control lines to the Frequency Histogram. The block in green shows how to add a normal curvefit.

```
{
        "FrequencyHistogram": {
            "ChartSetup": {
                "HistogramPlot": {
```

```
                "LineColor": "BLACK",
                "LineWidth": 1,
                "FillColor": "GREEN"
            },
            "ControlLines": [
                {
                    "LimitValue": 21,
                    "LineColor": "BLUE",
                    "LineWidth": 3
                },
                {
                    "LimitValue": 58,
                    "LineColor": "RED",
                    "LineWidth": 3
                }
            ],
            "NormalCurveLine": {
                    "Enable": true,
                    "LineColor": "YELLOW",
                    "LineWidth": 3
            }
        },
        "FrequencyHistogramData": {
            "SampleValues": [
                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
             ],
            "FrequencyBins": [
                19.5, 24.5, 29.5, 34.5, 39.5, 44.5, 49.5, 54.5, 59.5
       ]
        },
        "Methods": {
            "RebuildAndDraw": true
        }
    }
}
```

JSON Structure Summary

```
FrequencyHistogram
    ChartSetup
          ChartPositioning
          X1: double
          Y1: double
          X2: double
```

```
Y2: double
MainTitle
   Font
      Name: String
      Size: double
      Style: String
   TextColor: Color String constant
   Text: String
CoordinateSystem
   MinXScale: double
   MinYScale: double
   MaxXScale: double
   MaxYScale: double
XAxis
   LineColor: Color String constant
   LineWidth: double
XAxisLabels
   Font
      Name: String
      Size: double
      Style: String
   TextColor: Color String constant
   Rotation: double
   Format: SPC String constant
   OverlapLabelMode: SPC String constant
   Decimal: integer
   AxisLabelsStrings: [String, String, ...]
XAxisTitle
   Font
      Name: String
      Size: double
      Style: String
   TextColor: Color String constant
   Text: String
YAxis
   LineColor: Color String constant
   LineWidth: double
YAxisLabels
   Font
      Name: String
      Size: double
      Style: String
   TextColor: Color String constant
   Rotation: double
   Format: SPC String constant
   OverlapLabelMode: SPC String constant
   Decimal: integer
   AxisLabelsStrings: [String, String, ...]
YAxisTitle
   Font
      Name: String
      Size: double
      Style: String
   TextColor: Color String constant
   Text: String
```

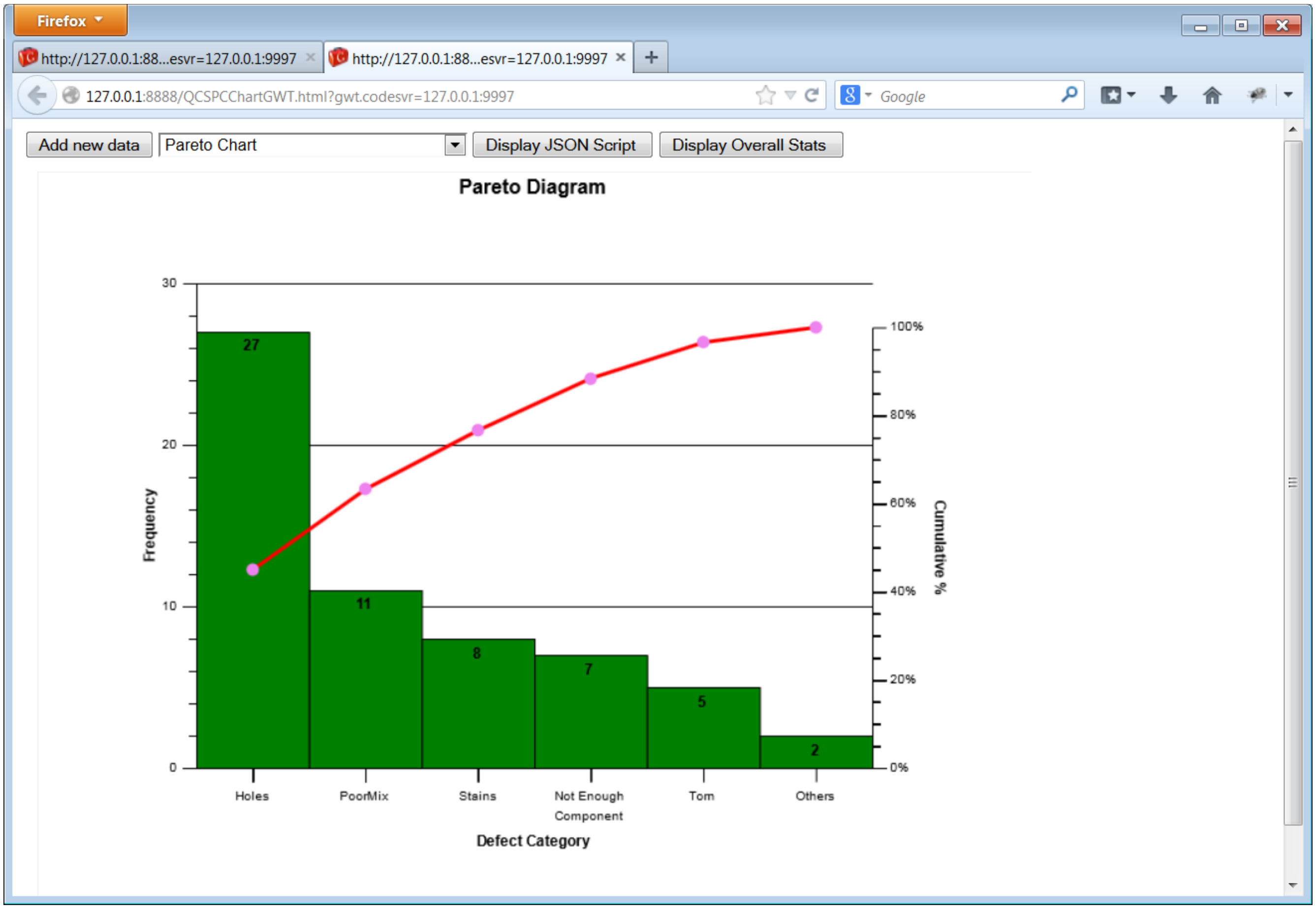
```
          HistogramPlot
             LineColor: Color String constant
             LineWidth: double
             BarColor: Color String constant
          GraphBackground
             FillColor: Color String constant
             BackgroundMode: SPC String constant
             GradientStartColor: Color String constant
             GradientStopColor: Color String constant
          PlotBackground
             FillColor: Color String constant
             BackgroundMode: SPC String constant
             GradientStartColor: Color String constant
             GradientStopColor: Color String constant
       LimitValueDecs: integer
       ControlLines
       [  {
             LimitValue: double
             LineColor: Color String constant
             LineWidth: double
          },
          {
             LimitValue: double
             LineColor: Color String constant
             LineWidth: double
          }, ...
       ]
       NormalCurveLine
       {
          Enable: boolean
          LineColor: Color String constant
          LineWidth: double
       }
    FrequencyHistogramData
       SampleValues [double, double, ...]
       FrequencyBins [double, double, ...]
    Methods
       RebuildAndDraw
```

18. Pareto Diagrams

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 ParetoPlot file of the ParetoDiagram example program.

```
    "ParetoChart": {
        "ChartSetup": {
            "BarPlot": {
                "LineColor": "BLACK",
                "LineWidth": 1,
                "FillColor": "GREEN"
            },
            "LineMarkerPlot": {
                "LineColor": "RED",
                "SymbolColor": "VIOLET"
            }
        },
        "ParetoChartData": {
            "CategoryItems": [
                5,
                7,
                2,
                11,
                27,
                8
            ],
            "CategoryStrings": [
                "Torn",
                "Not Enough\nComponent",
                "Others",
                "PoorMix",
                "Holes",
                "Stains"
            ]
        },
        "Methods": {
            "RebuildUsingCurrentData": true
        }
    }
}
```

Supplying a Pareto Chart with Data

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.

ParetoChartData

```
ParetoChartData
      CategoryItems [double, ...]
      CategoryStrings [String, ...]
```

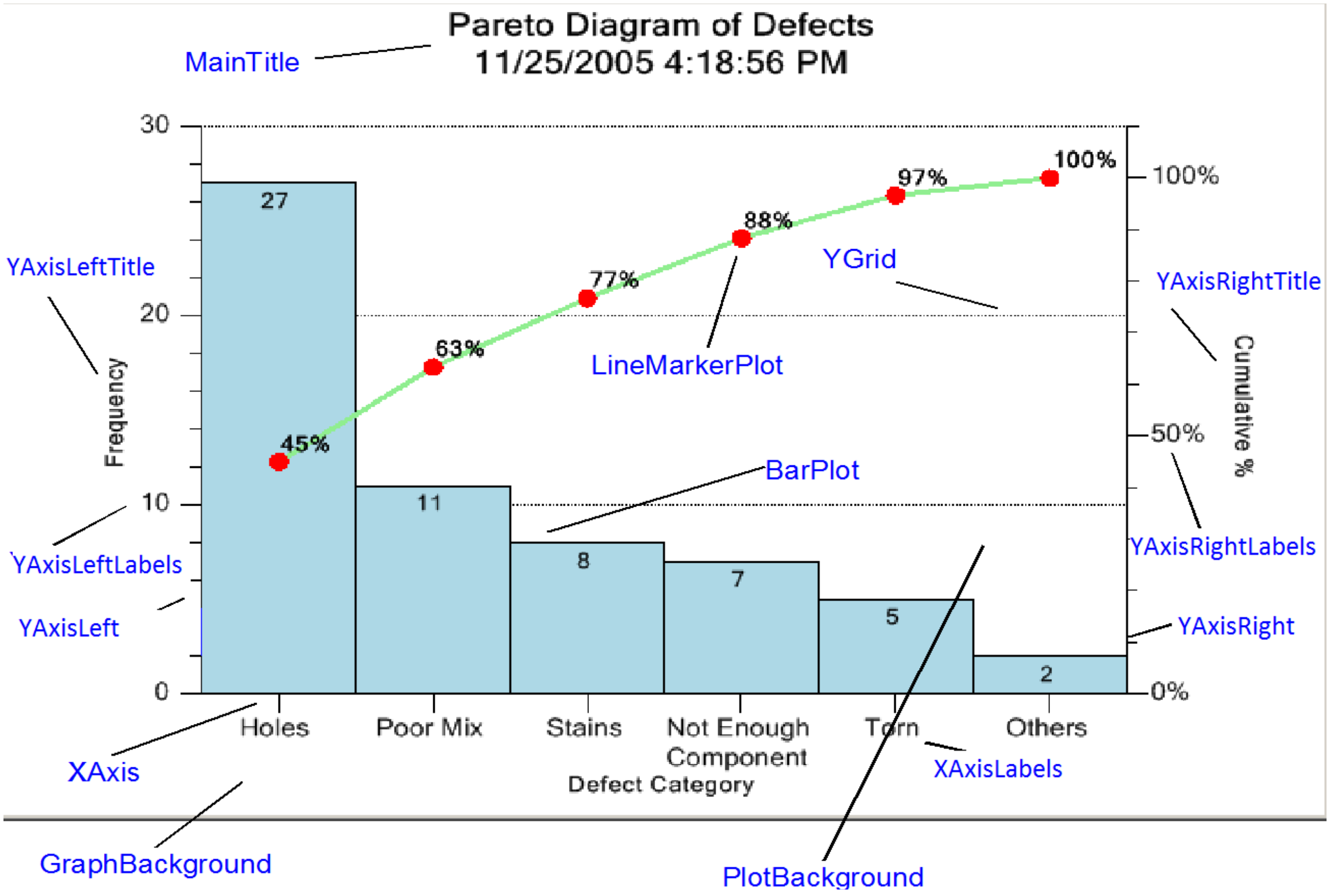
Initializes the x- and y-values of the data points plotted in the probability plot.

CategoryItems
The values for each category in the Pareto chart.
CategoryStrings
The strings identifying each category in the Pareto chart.

Example

```
"ParetoChartData": {
    "CategoryItems": [
        5,
        7,
        2,
        11,
        27,
        8
    ],
    "CategoryStrings": [
        "Torn",
        "Not Enough\nComponent",
        "Others",
        "PoorMix",
        "Holes",
        "Stains"
    ]
},
```

Changing Default Characteristics of the Chart



You can modify the default characteristics of each graph using these properties. For example, you can change the color of bar plot, and the LineMarkerPlot using the **LineColor** property of those objects.

```
"ChartSetup": {
    "BarPlot": {
        "LineColor": "BLACK",
        "LineWidth": 1,
        "FillColor": "GREEN"
    },
    "LineMarkerPlot": {
        "LineColor": "RED",
        "SymbolColor": "VIOLET"
    }
},
```

JSON Structure Summary

```
ParetoChart
   ChartSetup
         ChartPositioning
         X1: double
         Y1: double
         X2: double
```

```
Y2: double
MainTitle
   Font
      Name: String
      Size: double
      Style: String
   TextColor: Color String constant
   Text: String
CoordinateSystem1
   MinXScale: double
   MinYScale: double
   MaxXScale: double
   MaxYScale: double
CoordinateSystem2
   MinXScale: double
   MinYScale: double
   MaxXScale: double
   MaxYScale: double
XAxis
   LineColor: Color String constant
   LineWidth: double
XAxisLabels
   Font
      Name: String
      Size: double
      Style: String
   TextColor: Color String constant
   Rotation: double
   Format: SPC String constant
   OverlapLabelMode: SPC String constant
   Decimal: integer
   AxisLabelsStrings: [String, String, ..]
XAxisTitle
   Font
      Name: String
      Size: double
      Style: String
   TextColor: Color String constant
   Text: String
YAxisLeft
   LineColor: Color String constant
   LineWidth: double
   MinValue: double
   MaxValue: double
YAxisLeftLabels
   Font
      Name: String
      Size: double
      Style: String
   TextColor: Color String constant
   Rotation: double
   Format: SPC String constant
   OverlapLabelMode: SPC String constant
   Decimal: integer
   AxisLabelsStrings: [String, String, ..]
```

```
        YAxisLeftTitle
            Font
                Name: String
                Size: double
                Style: String
            TextColor: Color String constant
            Text: String
        YAxisRight
            LineColor: Color String constant
            LineWidth: double
            MinValue: double
            MaxValue: double
        YAxisRightLabels
            Font
                Name: String
                Size: double
                Style: String
            TextColor: Color String constant
            Rotation: double
            Format: SPC String constant
            OverlapLabelModeconstant (String)
            Decimal: integer
            AxisLabelsStrings: [String, String, ...]
        YAxisRightTitle
            Font
                Name: String
                Size: double
                Style: String
            TextColor: Color String constant
            Text: String
        BarPlot
            LineColor: Color String constant
            LineWidth: double
            BarColor: Color String constant
        LineMarkerPlot
            LineColor: Color String constant
            LineWidth: double
            SymbolFillColor: Color String constant
            SymbolLineColor: Color String constant
            SymbolColor: Color String constant
            SymbolSize: double
        GraphBackground
            FillColor: Color String constant
            BackgroundMode: SPC String constant
            GradientStartColor: Color String constant
            GradientStopColor: Color String constant
        PlotBackground
            FillColor: Color String constant
            BackgroundMode: SPC String constant
            GradientStartColor: Color String constant
            GradientStopColor: Color String constant
ParetoChartData
        CategoryItems [double, ...]
        CategoryStrings [String, ...]
Methods
    RebuildAndDraw
```

19. Regionalization

```
StaticProperties
   SPCChartStrings
      .
      .
      .
      TimeValueRowHeader: String: "TIME"
      AlarmStatusValueRowHeader: String: "ALARM"
      NumberSamplesValueRowHeader: String: "NO. INSP."
      TitleHeader: String: "Title: "
      PartNumberHeader: String: "Part No.: "
      ChartNumberHeader: String: "Chart No.: "
      PartNameHeader: String: "Part Name: "
      OperationHeader: String: "Operation: "
      OperatorHeader: String: "Operator: "
      .
      .
      .
```

Regionalization is done through initialization of static strings within the library. This is done using the StaticProperties.SPCChartStrings property block. There are 125 string constants (more or less) used in the software. Their values are all static. You can change any, or all of the string constants to match your requirements.

A list of the SPCChartStrings appears at the end of this chapter.

Change the strings using the following JSON example:

```
"SPCChartStrings": {
   "DefaultMean": "Average",
   "TimeValueRowHeader": "Time"
}
```

List as many of the 125 strings as you need to change. Make sure to surround both the string property name, "DefaultMean" for example, and the string value, "Average" with quotes.

Example

The SPCExampleScripts.js TimeXBarR example has an example of how to use the QCSPCChartString property block.

Full List of the Static SPCChartStrings Objects

SPCChartStrings

start	"start" - used to mark the beginning of the array
ChartFont	"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
GaugeHeader	"Gauge: " - 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" - Moving Average
DefaultCusumCPlus	"C+" - CuSum Plus string
DefaultCusumCMinus	"C-" - CuSum Minus string
DefaultEWMA	"EWMA" - EWMA string
DefaultPercentDefective	"% DEF." - Percent Defective
DefaultFractionDefective	"FRACT. DEF." - Fraction Defective
DefaultNumberDefective	"NO. DEF." - Number Defective
DefaultNumberDefects	"NO. DEF." - Number Defects
DefaultNumberDefectsPerUnit	"NO. DEF./UNIT" - Number Defects per Unit
DefaultNumberDefectsPerMillion	"DPMO" - Number Defects per Million
DefaultPBar	"PBAR" - Target label for Attribute charts
DefaultAttributeLCL	"LCLP" - Low limit label for Attribute charts
DefaultAttributeUCL	"UCLP" - High limit label for Attribute charts
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, usually over-ridden when data log opened.
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",
Weco	"WECO" - WECO rules string
Wecowsupp	"WECO+SUPPLEMENTAL" - WECO 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 Westgard Rule R2s #9
SigmaShort	"S" - Used in alarm messages as sigma short string
BbeyondAlarmStatus	"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
R4sAlarmStatus	"R" - alarm status line - R4s 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"
DefaultCpl	"Cpl"
DefaultCpu	"Cpu"

DefaultCpk	"Cpk"
DefaultCpm	"Cpm"
DefaultPp	"Pp"
DefaultPl	"Pl"
DefaultPu	"Pu"
DefaultPpk	"Ppk"
Canceltext	"Cancel" - used for buttons in dialogs that have cancel button
Alarmstatusdialogtitle	"Alarm Status" - used as the title for the alarm status dialog box
end	"end" - used to mark the end of the array

20. Using SPC Control Chart Tools for Javascript to Create Web Applications

Deployment to an actual web site

It is very simple to deploy an application to a web site. Unzip the distribution zip file, JSSPCDEV1.zip, to your hard drive. You will end up with the directory structure below.

Drive:

Quinn-Curtis\ - Root directory

GWTJavascript\ - Quinn-Curtis GWT / Javascript folder

Docs\ - documentation directory

QCSPCChartGWTDoc.pdf – User guide

QCSPCChartGWTWar\ - Contains the

qcspcchartgwt\ – this folder contains the compiled, chached, Javascript libraries for QCSPCChartGWT. There are at least six different version of the libraries optimized for the HTML5 support in the major browsers (IE, Firefox, Chrome, and Safari).

QCSPCChartGWT.css – a css style sheet controlling some of the default characteristics of the chart

EXAMPLE PROGRAMS and SCRIPTS

SPCSimple.html
ChartdefSimple.js
SPCMediumSimple.html
ChartdefMediumSimple.js
MediumSimpleDataUpdate.js

SPCComplex.html
ChartdefComplex.js
SPCExampleScripts.html
ChartdefExampleScripts.js
QCSPCSkeleton.html
ChartdefUserDefined.js

QCSPCChart+ examples (contains 9 scripts)

EnhQCSPCChartExample.html

chartdefEnhExamples.js

New Features found in Rev. 3.6

SPCExampleScriptsR36.html

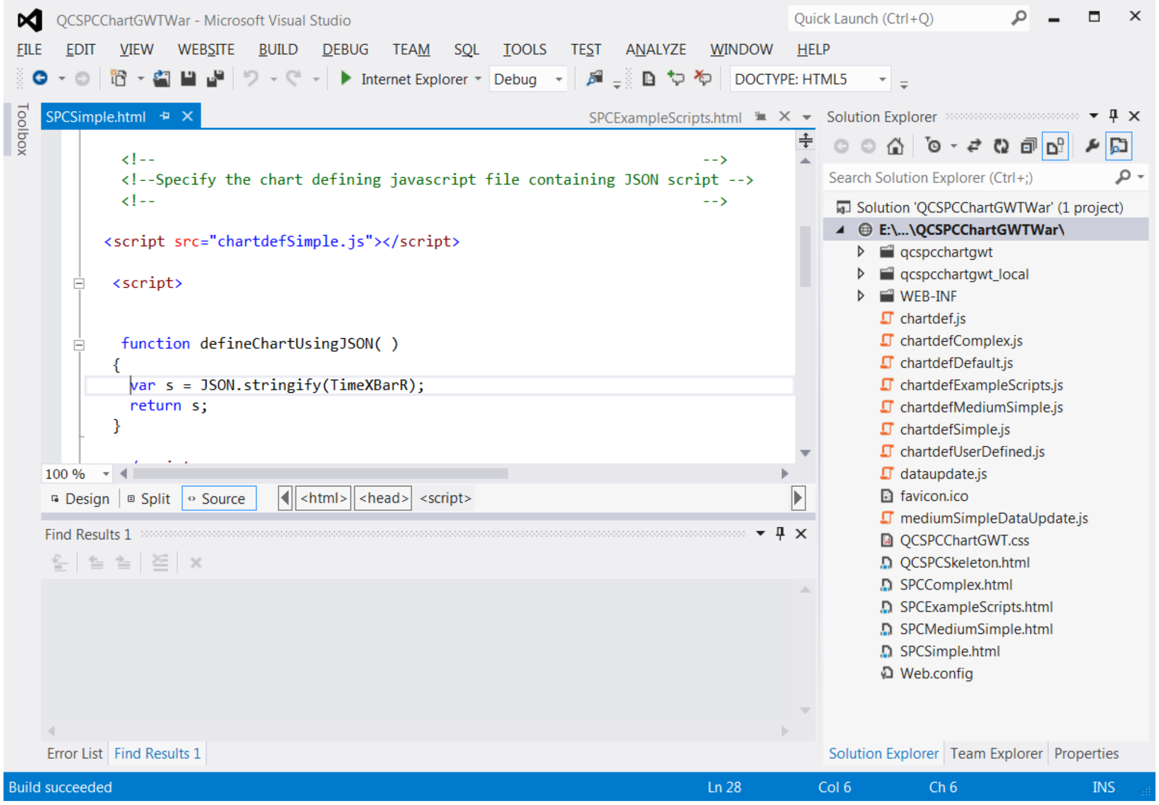
chartdefExampleScriptsR36.js

Copy the folder QCSPCChartGWTWar to the appropriate folder of your web site, either the development web site, or the deployment web site. It contains the QCSPCChartGWT folder, which contains the compiled, chached, Javascript libraries for QCSPCChartGWT. There are at least six different version of the libraries optimized for the HTML5 support in the major browsers (IE, Firefox, Chrome, and Safari).

Deployment to a computer that is not a web server

You are able to deploy the software to a computer which is not setup as a web server. If you have Visual Studio (2010, 2012) running under Windows 7, installed on a desktop machine, you can deploy the software to that environment. Just unzip the distribution zip file to a local drive on your desktop. Load Visual Studio and select File | Open |Website and point to the QCSPCChartGWTWar folder. You will see the list of HTML and JS files in the Solution Explorer.

Visual Studio 2012 editing a HTML file

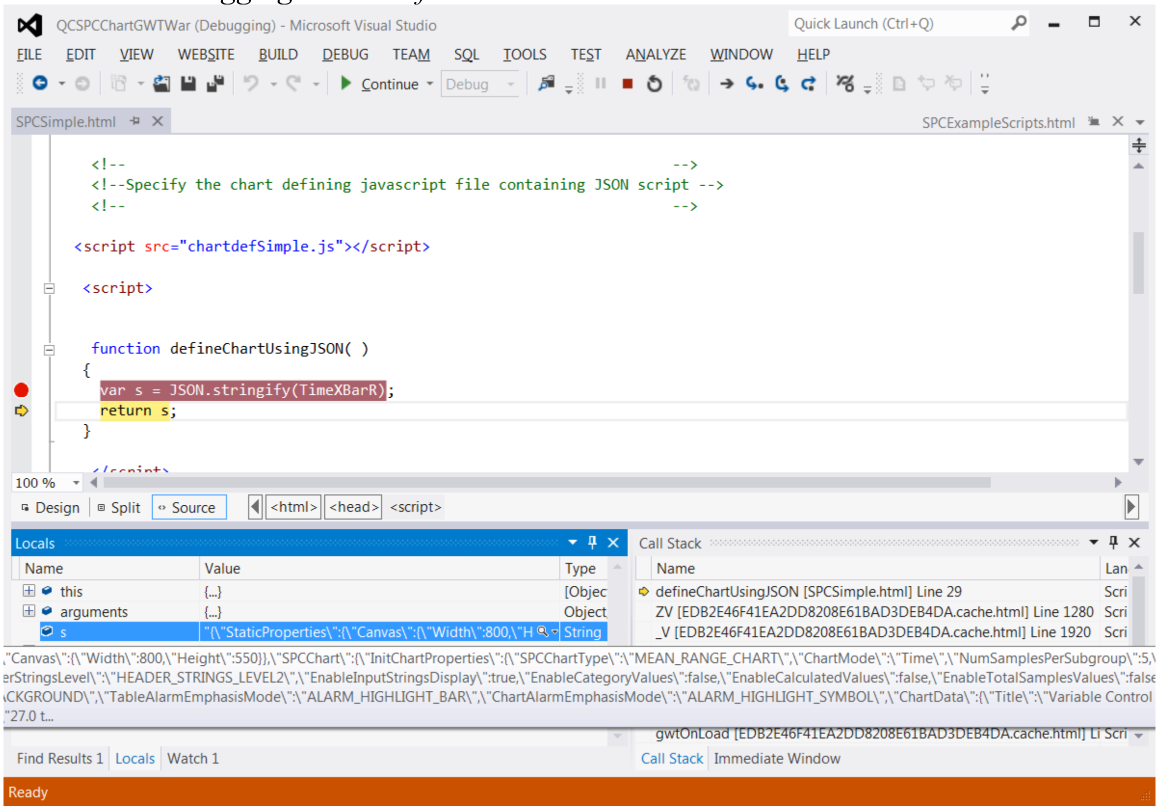


```
  <!--                                                    -->
  <!--Specify the chart defining javascript file containing JSON script -->
  <!--                                                    -->

<script src="chartdefSimple.js"></script>

 <script>


  function defineChartUsingJSON( )
 {
   var s = JSON.stringify(TimeXBarR);
   return s;
 }
```

Editing and Debugging using Visual Studio

You can edit the raw text of the HTML and JS files using the Visual Studio editor. If you want to display the HTML file in the Visual Studio local server, load it into the editor, and select Main Menu | DEBUG | Start without Debugging. User **Internet Explorer** as the browser. Firefox seem to have some Canvas clipping issues when use as the host browser for Visual Studio. You can also debug the HTML file using Visual Studio: Select Main Menu | DEBUG | Start Debugging. It will stop at any Javascript breakpoints you set and allow you to step through the Javascript code.

Visual Studio 2012 debugging a HTML file



```
    <!--                                                    -->
    <!--Specify the chart defining javascript file containing JSON script -->
    <!--                                                    -->

  <script src="chartdefSimple.js"></script>

   <script>

    function defineChartUsingJSON( )
   {
     var s = JSON.stringify(TimeXBarR);
     return s;
   }

    </script>
```

Locals

Name	Value	Type
this	{...}	[Objec
arguments	{...}	Object
s	"{\"StaticProperties\":{\"Canvas\":{\"Width\":800,\"H	String

Call Stack

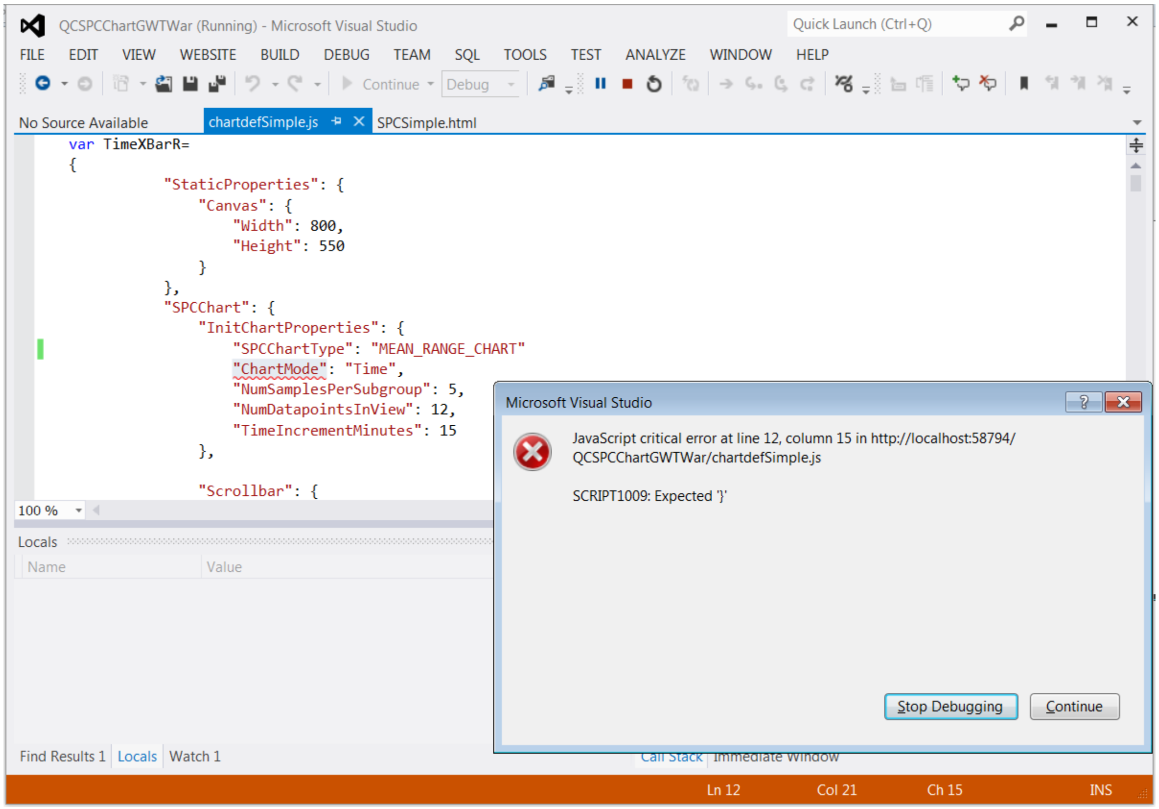
Name	Lan
defineChartUsingJSON [SPCSimple.html] Line 29	Scri
ZV [EDB2E46F41EA2DD8208E61BAD3DEB4DA.cache.html] Line 1280	Scri
_V [EDB2E46F41EA2DD8208E61BAD3DEB4DA.cache.html] Line 1920	Scri
gwtOnLoad [EDB2E46F41EA2DD8208E61BAD3DEB4DA.cache.html] Li	Scri

```
,"Canvas\":{\"Width\":800,\"Height\":550}},\"SPCChart\":{\"InitChartProperties\":{\"SPCChartType\":\"MEAN_RANGE_CHART\",\"ChartMode\":\"Time\",\"NumSamplesPerSubgroup\":5,\
erStringsLevel\":\"HEADER_STRINGS_LEVEL2\",\"EnableInputStringsDisplay\":true,\"EnableCategoryValues\":false,\"EnableCalculatedValues\":false,\"EnableTotalSamplesValues\":false
\CKGROUND\",\"TableAlarmEmphasisMode\":\"ALARM_HIGHLIGHT_BAR\",\"ChartAlarmEmphasisMode\":\"ALARM_HIGHLIGHT_SYMBOL\",\"ChartData\":{\"Title\":\"Variable Control
,"27.0 t...
```

The Visual Studio 2012 debugger will also help you debug the JSON script files, the ones ending in *.js in our examples), by identifying syntax errors in the scripts. In the JSON fragment below, a comma has been dropped from the TimeXBarR definition in the chartDefSimple.js file, and the line highlighted in RED:

```
var TimeXBarR=
 {
          "StaticProperties": {
              "Canvas": {
                  "Width": 800,
                  "Height": 550
              }
          },
          "SPCChart": {
              "InitChartProperties": {
                  "SPCChartType": "MEAN_RANGE_CHART"
                  "ChartMode": "Time",
                  "NumSamplesPerSubgroup": 5,
                  "NumDatapointsInView": 12,
                  "TimeIncrementMinutes": 15
              },
```

Attempts to run the SPCSimple.html file will result in a Visual Studio JavaScript critical error.



```
    var TimeXBarR=
    {
            "StaticProperties": {
                "Canvas": {
                    "Width": 800,
                    "Height": 550
                }
            },
            "SPCChart": {
                "InitChartProperties": {
                    "SPCChartType": "MEAN_RANGE_CHART"
                    "ChartMode": "Time",
                    "NumSamplesPerSubgroup": 5,
                    "NumDatapointsInView": 12,
                    "TimeIncrementMinutes": 15
                },

                "Scrollbar": {
```

JavaScript critical error at line 12, column 15 in http://localhost:58794/QCSPCChartGWTWar/chartdefSimple.js

SCRIPT1009: Expected '}'

We also strongly encourage that you check your JSON scripts using JSONLint, found at http://jsonlint.com/, described in the following Example Applications section.

Because the use of Firefox launched from Visual Studio seems to have some Canvas clipping issues, testing of the different browsers will need to occur with the software installed on an actual web site.

Example Applications

The QCSPCChartGWTWar folder also contains several HTML files, representing the example web pages we created demonstrating the different chart types and options. For each HTML file, there is one or more *.js Javascript include file, which contains one or more chart defining JSON scripts for the associated HTML page. These include:
SPCSimple.html and an include file containing a chart defining JSON script, ChartdefSimple.js.

SPCMediumSimple.html and an include file containing a chart defining JSON script, ChartdefMediumSimple.js, and another include file containing update data, MediumSimpleDataUpdate.js.

SPCComplex.html and an include file containing a chart defining JSON script, ChartdefComplex.js.

SPCExampleScripts.html – and an include file containing a chart defining JSON script, ChartdefExampleScripts.js.

QCSPCChart+ example (contains 9 scripts)
EnhQCSPCChartExample.html and an include file containing 9 chart defining JSON scripts, chartdefEnhExamples.js

If you plan to run the demos, leave these files in place. If you have created your own HTML file, and associated include files, you can delete all of these in your deployment. They are just examples, they have no function in support of the libraries found in the SPC folder.

The QCSPCChartGWT libraries were developed using the Eclipse IDE. The underlying source code of the libraries is Java. The QCSPCChartGWT libraries make use of libraries supplied by Google with their GWT compiler. The end result of the GWT compilation process results in 100% Javascript code, with no Java dependence. You do NOT need Eclipse, a Java Runtime, or any .Net (2.0, 3.0, 4.0 or 4.5) library installed on the target server. You only need the files in the QCSPCChartGWTWar folder. And our example HTML files do not need to be included either. Only the one or more HTML files you create for your own web application.

Since all of the code in the library is client-side Javascript, nothing special needs to be done on the server regarding permissions. The client-side browser will definitely need to be setup to allow execution of Javascript. Below is a description of how to do that for the most common browsers.

Internet Explorer - select the browser Tools | Internet Options | Security | Custom Level | Scripting | Active Scripting | Enable.

Firefox (Mozilla) - Javascript is on by default.

Chrome - menu icon ≡ on the browser toolbar | Settings | Show advanced settings | Content Settings in Privacy section | Allow all sites to run JavaScript in the JavaScript section.

Safari - To enable or disable JavaScript, tap Advanced and turn JavaScript on or off. JavaScript lets web programmers control elements of the page. For example, a page that uses JavaScript might display the current date and time or cause a linked page to appear in a new pop-up page.

Once you have the QCSPCChartGWTWar folder copied to the www directory of your web site, you should be able to display a web page by pointing your browser to one of the HTML files within. If you are running our examples, you should be able to point to the [yourwebsite].QCSPCChartGWTWar/SPCSimple.html page and display a basic X-Bar R chart. On our web site, the web page is found at:

http://quinn-curtis.com/QCSPCChartGWTWar/SPCSimple.html

The other examples are accessed using a similar URL.

http://quinn-curtis.com/QCSPCChartGWTWar/SPCMediumSimple.html

http://quinn-curtis.com/QCSPCChartGWTWar/SPCComplex.html

http://quinn-curtis.com/QCSPCChartGWTWar/SPCExampleScripts.html

Inside the QCSPCChartSimple.html example, you will find the following HTML and Javascript code. The other example HTML pages look pretty much the same, with more Javascript code to control the HTML elements found on the page

```
<!doctype html>
<!-- The DOCTYPE declaration above will set the     -->
<!-- browser's rendering engine into                -->
<!-- "Standards Mode". Replacing this declaration   -->
<!-- with a "Quirks Mode" doctype is not supported. -->

<html>
  <head>

    <meta http-equiv="content-type" content="text/html; charset=UTF-8">

    <!--                                                   -->
    <!-- Consider inlining CSS to reduce the number of requested files -->
    <!--                                                   -->
  <link type="text/css" rel="stylesheet" href="QCSPCChartGWT.css">

    <!--                                                   -->
    <!--Specify the chart defining Javascript file containing JSON script -->
    <!--                                                   -->

  <script src="chartdefSimple.js"></script>

   <script>
    <!--                                                   -->
    <!-- The QCSPCChartGWT library calls this function if present.     -->
    <!-- It returns a string representation of the JSON script.
-->
   function defineChartUsingJSON( )
   {
      var s = JSON.stringify(TimeXBarR);
     return s;
   }
    </script>

    <!--                                          -->
    <!-- This script loads your compiled module.   -->
    <!-- If you add any GWT meta tags, they must   -->
    <!-- be added before this line.                -->
    <!--                                          -->
   <script type="text/Javascript" language="Javascript"
src="qcspcchartgwt/qcspcchartgwt.nocache.js"></script>
  </head>


  <body onload="loadform()">

    <!-- OPTIONAL: include this if you want history support -->
    <iframe src="Javascript:''" id="__gwt_historyFrame" tabIndex='-1'
style="position:absolute;width:0;height:0;border:0"></iframe>

    <!-- RECOMMENDED if your web app will not function without JavaScript enabled -->
    <noscript>
      <div style="width: 22em; position: absolute; left: 50%; margin-left: -11em;
color: red; background-color: white; border: 1px solid red; padding: 4px; font-
family: sans-serif">
        Your web browser must have JavaScript enabled
        in order for this application to display correctly.
      </div>
    </noscript>

  </body>
</html>
```

The line:

```
  <link type="text/css" rel="stylesheet" href="QCSPCChartGWT.css">
```

references the QCSPCChartGWT.css stylesheet. You can customize elements of the chart by modifying the contents of that include file. See Chapter 19 CSS Style Sheets.

The line:

```
  <script src="chartdefSimple.js"></script>
```

references the chartdefSimple.js Javascript include file, which defines a Javascript record structure (TimeXBarR) which can be converted using the JSON.stringify function into a JSON scripting string. The Javascript definition of the TimeXBarR record could just have easily been included in the main HTML file, without using the chartdefSimple.js include file. It's just that the TimeXBarR definition, particularly if it has a lot of data, can be quite long and will make the main HTML file harder to understand.

The Javascript function defineChartUsingJSON

```
   function defineChartUsingJSON( )
   {
      var s = JSON.stringify(TimeXBarR);
     return s;
   }
```

is the critical link link into the QCSPCChartGWT library. The library, when it starts up, looks to see if this function is present in the host HTML page. If the function is present, it calls it, and expects the function to return a valid chart defining JSON string. The JSON string must be a syntactically correct JSON structure, and it must properly define a SPC chart following the rules discussed in this manual.

The Javascript record structure which is converted into a JSON string must be valid JSON. However, it can be difficult to debug a JSON compatible Javascript structure. Typical errors even an expert is going to make include:

Leave an extra comma at the end of the last item in a block

```
"SPCChart": {
    "InitChartProperties": {
        "SPCChartType": "MEAN_RANGE_CHART",
        "ChartMode": "Time",
        "NumSamplesPerSubgroup": 5,
        "NumDatapointsInView": 12,
        "TimeIncrementMinutes": 15,
    },
    "Scrollbar": {
        "EnableScrollBar": true,
        "ScrollbarPosition": "SCROLLBAR_POSITION_MAX"
    },
```

Leave out a comma between adjacent blocks

```
"SPCChart": {
    "InitChartProperties": {
        "SPCChartType": "MEAN_RANGE_CHART",
        "ChartMode": "Time",
        "NumSamplesPerSubgroup": 5,
        "NumDatapointsInView": 12,
        "TimeIncrementMinutes": 15
    }
    "Scrollbar": {
        "EnableScrollBar": true,
        "ScrollbarPosition": "SCROLLBAR_POSITION_MAX"
    },
```

Fail to quote the property name

```
"InitChartProperties": {
         "SPCChartType": "MEAN_RANGE_CHART",
          ChartMode: "Time",
         "NumSamplesPerSubgroup": 5,
         "NumDatapointsInView": 12,
         "TimeIncrementMinutes": 15
     }
```

Include only a single quote in a property name or value

```
          "InitChartProperties": {
                  "SPCChartType": "MEAN_RANGE_CHART",
                   "ChartMode: "Time",
                  "NumSamplesPerSubgroup": 5,
                  "NumDatapointsInView": 12,
                  "TimeIncrementMinutes": 15
              }
```

Fail to put quotes around a string value or string constant

```
          "InitChartProperties": {
                  "SPCChartType": MEAN_RANGE_CHART,
                  "ChartMode": "Time",
                  "NumSamplesPerSubgroup": 5,
                  "NumDatapointsInView": 12,
                  "TimeIncrementMinutes": 15
              }
```

Put quotes around a numeric or boolean value.

```
          "InitChartProperties": {
                  "SPCChartType": "MEAN_RANGE_CHART",
                   "ChartMode": "Time",
                   "NumSamplesPerSubgroup": "5",
                   "NumDatapointsInView": 12,
                   "TimeIncrementMinutes": 15
              }
```

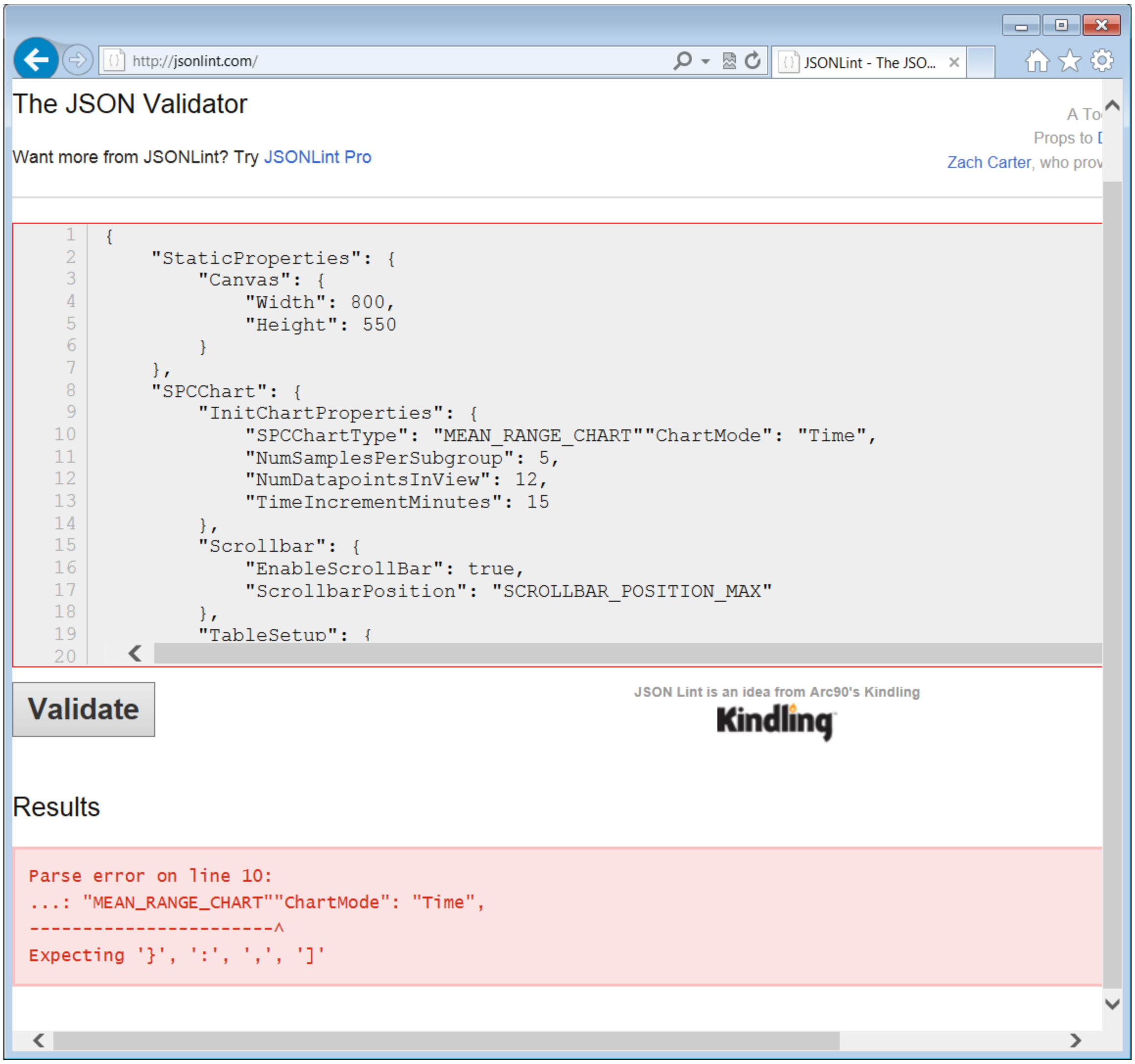
All of the examples above have errors in them.

JSONLint.com

We relied on an external site, http://jsonlint.com/, where you can paste a JSON structure, have it checked and get meaningful debugging information back. It is not a Javascript checker though. So if you copy and paste a JSON structure, such as the TimeXBarR Javascript variable defined in the chartDefSimple.js include file, you leave out "**var TimeXBarR** = " part, and the trailing semicolon "**;**". The part you include is the block between, and *including* the first and last curly bracket, {…}.

```
 {
    "StaticProperties": {
        "Canvas": {
            "Width": 800,
            "Height": 550
        }
    },
    "SPCChart": {
        "InitChartProperties": {
            "SPCChartType": "MEAN_RANGE_CHART",
            "ChartMode": "Time",
            "NumSamplesPerSubgroup": 5,
            "NumDatapointsInView": 12,
            "TimeIncrementMinutes": 15
          },
          .
          .
          .

       }
}
```

An invalid JSON script will produce an error in the JSONLint application which looks something like this:



The JSON Validator

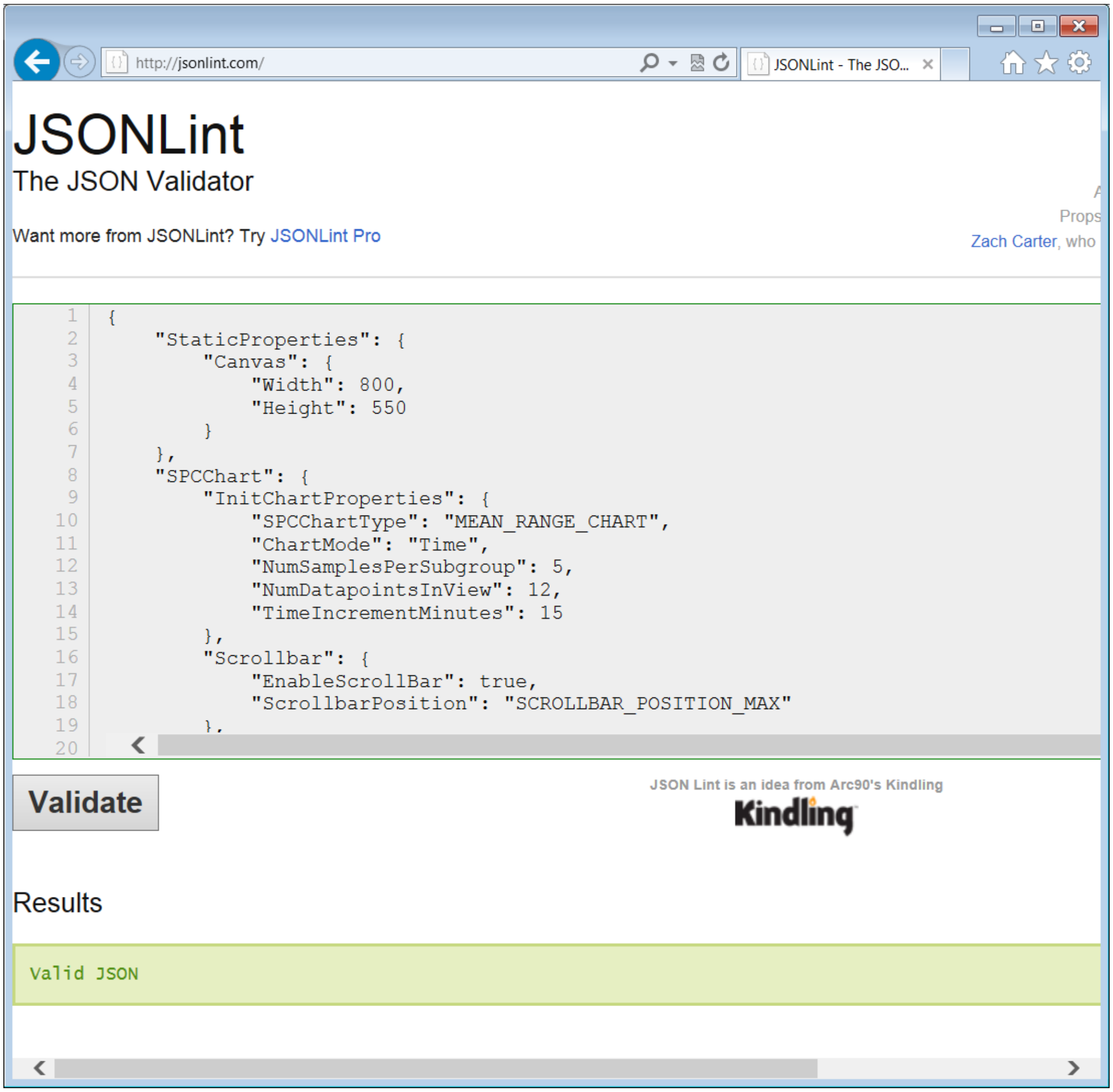
Want more from JSONLint? Try JSONLint Pro

```
{
    "StaticProperties": {
        "Canvas": {
            "Width": 800,
            "Height": 550
        }
    },
    "SPCChart": {
        "InitChartProperties": {
            "SPCChartType": "MEAN_RANGE_CHART""ChartMode": "Time",
            "NumSamplesPerSubgroup": 5,
            "NumDatapointsInView": 12,
            "TimeIncrementMinutes": 15
        },
        "Scrollbar": {
            "EnableScrollBar": true,
            "ScrollbarPosition": "SCROLLBAR_POSITION_MAX"
        },
        "TableSetup": {
```

Results

```
Parse error on line 10:
...: "MEAN_RANGE_CHART""ChartMode": "Time",
-----------------------^
Expecting '}', ':', ',', ']'
```

You can edit the script in place, correcting errors as the arise. Once you have removed all of the syntax errors from the script, you will get the Valid JSON message in the Results window. At that point you can copy the JSON script out of the edit window and back into your source Javascript file.



JSONLint

The JSON Validator

Want more from JSONLint? Try JSONLint Pro

A
Props
Zach Carter, who

```
{
    "StaticProperties": {
        "Canvas": {
            "Width": 800,
            "Height": 550
        }
    },
    "SPCChart": {
        "InitChartProperties": {
            "SPCChartType": "MEAN_RANGE_CHART",
            "ChartMode": "Time",
            "NumSamplesPerSubgroup": 5,
            "NumDatapointsInView": 12,
            "TimeIncrementMinutes": 15
        },
        "Scrollbar": {
            "EnableScrollBar": true,
            "ScrollbarPosition": "SCROLLBAR_POSITION_MAX"
        },
```

JSON Lint is an idea from Arc90's Kindling

Results

```
Valid JSON
```

Just because the Javascript record structure passes a JSON syntax check doesn't mean it is setup correctly for our JSON SPC chart parser. You must get the basic order of elements correct, and the hierarchy of the properties correct. For any given property block, the software does a keyword check for all valid sub-elements allowed within that block. For example, when parsing the **SPCChart** block, the valid keywords for sub-elements are:

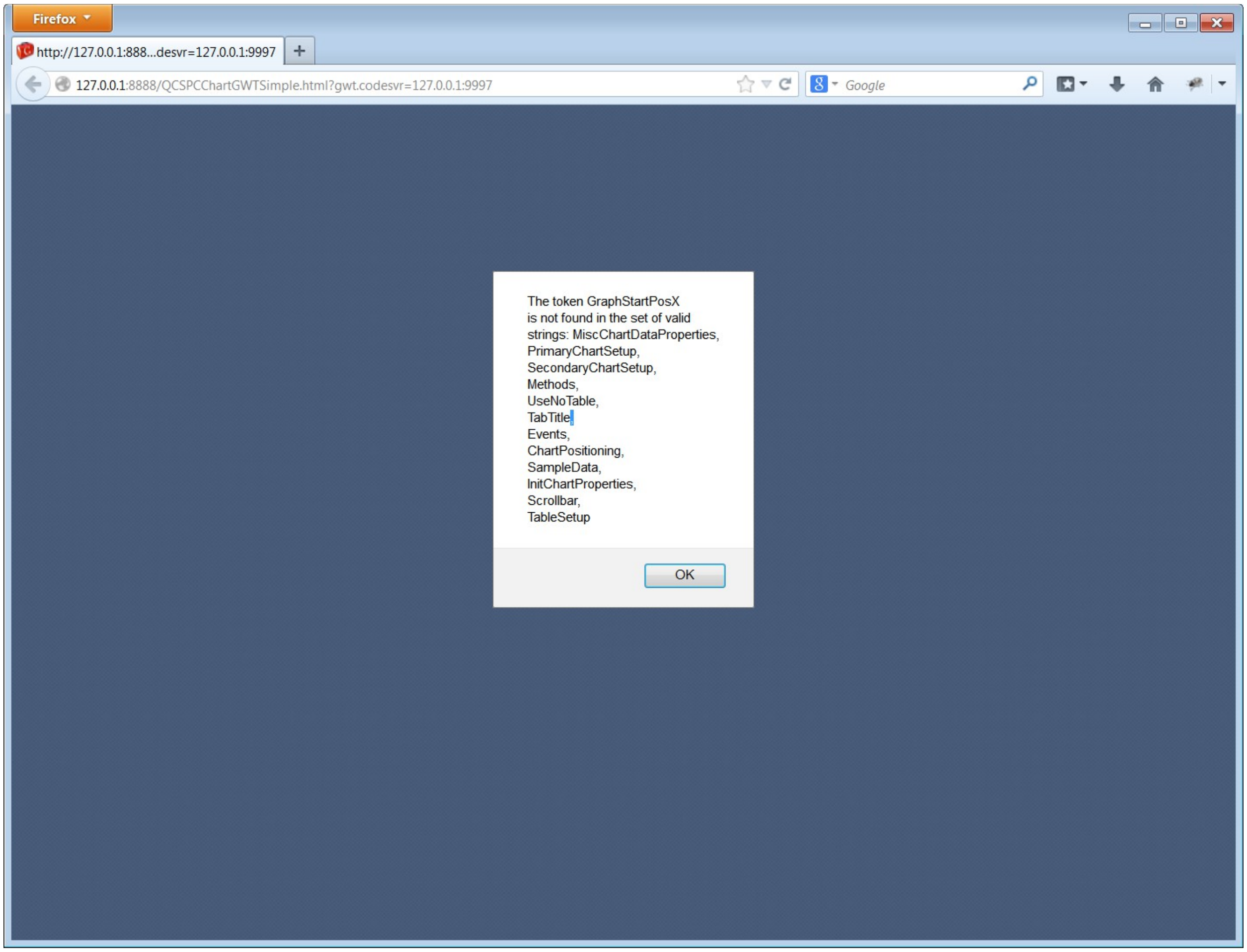
```
InitChartProperties
ChartPositioning
Scrollbar
UseNoTable
TableSetup
MiscChartDataProperties
PrimaryChartSetup
SecondaryChartSetup
Events
```

```
SampleData
Methods
```

If any other keyword is found as an immediate child of the SPCChart block, an error is generated and a message displayed on the web page. For example, the following JSON script passes the http://jsonlint.com/ syntax check. However, if you pass this string to the library, it will generate an error, because the GraphStartPosX and GraphStopPosX properties are out of place.

```
 {
    "StaticProperties": {
        "Canvas": {
            "Width": 800,
            "Height": 550
        }
    },
    "SPCChart": {
        "InitChartProperties": {
            "SPCChartType": "MEAN_RANGE_CHART",
            "ChartMode": "Time",
            "NumSamplesPerSubgroup": 5,
            "NumDatapointsInView": 12,
            "TimeIncrementMinutes": 15
          },
          "GraphStartPosX": 0.15,
          "GraphStopPosX": 0.8
       }
}
```

The error message you would see looks something like this:


Firefox
http://127.0.0.1:888...desvr=127.0.0.1:9997
127.0.0.1:8888/QCSPCChartGWTSimple.html?gwt.codesvr=127.0.0.1:9997
Google
OK

The token GraphStartPosX
is not found in the set of valid
strings: MiscChartDataProperties,
PrimaryChartSetup,
SecondaryChartSetup,
Methods,
UseNoTable,
TabTitle,
Events,
ChartPositioning,
SampleData,
InitChartProperties,
Scrollbar,
TableSetup

When you press the OK button, you will get another error message for the GraphStopPosX property too. This tells you the GraphStartPosX and GraphStopPosX properties are out of place. The corrected code should look like:

```
{
   "StaticProperties": {
       "Canvas": {
           "Width": 800,
           "Height": 550
       }
   },
   "SPCChart": {
       "InitChartProperties": {
           "SPCChartType": "MEAN_RANGE_CHART",
           "ChartMode": "Time",
           "NumSamplesPerSubgroup": 5,
           "NumDatapointsInView": 12,
           "TimeIncrementMinutes": 15
         },
         "ChartPositioning": {
             "GraphStartPosX": 0.15,
                "GraphStopPosX": 0.8
            }
        }
}
```

The **defineChartUsingJSON** function above loads a chart automatically when the parent HTML page is loaded. Instead of that sequence, you may want the chart to only be loaded and displayed in response to an external event. In that case, you would leave the defineChartUsingJSON function out of your HTML file. Instead, make a call to the function **pushJSONChartCreate**, passing in the same JSON string as the **defineChartUsingJSON** example.

```
function displayChart() {
  pushJSONChartCreate(JSON.stringify(TimeXBarR));
}
```

The **pushJSONChartCreate** function is an un-obfuscated Javascript function exported from the QCSPCChartGWT libray, so that you can call it from within the main HTML page. In all of our example programs, we set the default chart on the web page using **defineChartUsingJSON.** In the case of the SPCExampleScripts example, we let you change the default by selecting a new chart from a drop down list, implemented using an HTML select item. Selecting a new chart using the drop down select element triggers an event, which calls the **displayChart** function. Using the passed in value of the *chartid* variable, the defining chart JSON script is retrieved using **getChartItem**, and then that script is converted to a string and passed into the **pushJSONChartCreate** function.

```
function displayChart(chartid) {
 var chartitem = getChartItem(chartid);
 pushJSONChartCreate(JSON.stringify(chartitem));
}
```

All of the examples contain the block of Javascript seen below.

```
   <!--                                        -->
    <!-- This script loads your compiled module.   -->
    <!-- If you add any GWT meta tags, they must   -->
    <!-- be added before this line.                -->
    <!--                                           -->
   <script type="text/Javascript" language="Javascript"
src="qcspcchartgwt/qcspcchartgwt.nocache.js"></script>
  </head>


  <body onload="loadform()">

    <!-- OPTIONAL: include this if you want history support -->
    <iframe src="Javascript:''" id="__gwt_historyFrame" tabIndex='-1'
style="position:absolute;width:0;height:0;border:0"></iframe>

    <!-- RECOMMENDED if your web app will not function without JavaScript enabled -->
    <noscript>
      <div style="width: 22em; position: absolute; left: 50%; margin-left: -11em;
color: red; background-color: white; border: 1px solid red; padding: 4px; font-
family: sans-serif">
        Your web browser must have JavaScript enabled
        in order for this application to display correctly.
      </div>
    </noscript>

  </body>
```

This is the GWT magic which loads the browser specific version of the GWT libraries, and the QCSPCChartGWT libraries, and creates an iframe which is used to display the charts. You should not modify anything in this section.

Javascript, jQuery, Ajax and PHP

The data you feed into the charts will probably come from some sort of web service. You may retrieve it directly from a database using jQuery, or you may communicate with a server side program, which serves up the data as JSON script. Here are some examples of how to do that.

Update Chart Data by Pushing New Array Elements into a charts SampleData Stucture

Files: phpPushDataExample.html, phppushdata.php, chartDefPHPExample.js

In the Javascript located in the HTML page, it uses the jQuery Ajax function ($.Ajax) to communicate with a PHP program located on the same server which serves up the HTML page.

The PHP (phppushdata.php) looks like:

```
<?php
header('Content-type: application/json; charset=utf-8');

srand();

$ret = array(
             "SampleValues"=> array(
                27.53 + rand(0,5),
                33.95 + rand(0,5),
                24.31 + rand(0,5),
                28.28 + rand(0,5),
                30.29 + rand(0,5),
             ),
             "BatchCount"=> 0,
             "TimeStamp"=> 1371830829074,
             "Note"=> ""
          );
      echo json_encode($ret);
```

```
exit();
?>
```

Make sure you include the header block.

This uses the PHP array syntax to duplicate the record structure of a SampleValues block. The PHP function **json_enode** takes variable $ret and converts it into JSON using the PHP function json_encode, and echos it back to the calling program. The resulting JSON code will look something like:

{"SampleValues":
[28.67,36.68,24.82,29.6,31.13],"BatchCount":0,"TimeStamp":1371830829074,"Note":""}

which is a valid JSON object, and which matches the structure of an array element of our SPCChart.SampleData.SampleIntervalRecords block. Since it is a match, it can be "pushed" into an existing charts SampleIntervalRecords block, appending it as a new array element of that array.

We use the $(window).load event to start things off once the page is loaded. The $.ajax function is placed as the function called by the **setInterval** function, when it is triggered. In this case, it is setup to trigger every 5000 milliseconds.

Special Note – Instead of $(window).load, some have tried to use the jQuery $(document).ready event. While this event is triggered, according to GWT documentation, it may be triggered before GWT has finished its setup, and this means that the functions in our library (such as pushJSONChartCreate), may not be available. So, if you want to guarantee that the page has been loaded, and GWT is done with its business, use $(window).load as seen below.

```
var timerOn = true;
var timerID = 0;
var timerCount = 0;

function stopStartTimer() {

    if (timerOn) {
        clearInterval(timerID);
        timerOn = false;
    }
    else {
        timerID = setInterval(timerFunction, 5000);
        timerOn = true;
    }
 }

 function timerFunction()
 {
     $.ajax({
         url: "phppushdata.php",
         type: "GET",
         dataType: "json",
          error: function (jqXHR, textStatus, errorThrown) {
              alert("Error");

          },
          success: function (result) {
              //  alert(JSON.stringify(result));

              result.BatchCount = timerCount;
              // simulate a sample interval every 15 minutes
              result.TimeStamp = result.TimeStamp + 900000 * timerCount;

              TimeXBarR.SPCChart.SampleData.SampleIntervalRecords.push(result);
              pushJSONChartCreate(JSON.stringify(TimeXBarR));
              timerCount++;

          }
      });

  }

$(window).load(function () {
    timerID = setInterval(function () { timerFunction() }, 5000);
    timerOn = true;
});
```

$.ajax Parameters

url: The url parameter points to the php file, in this case it is in the same directory as the HTML file. If you require that the source PHP program is on an entirely different server than the HTML page, that requires a more advanced use of the jQuery, and you will need to study the jQuery.Ajax documnetation for more information about how to do that.

type: This is a GET operation.

dataType: This must be json

error: You can specify an error processing function which is called if an error is generated.

success: This is called if the PHP file successfully returns. The succes function is where the chart is updated. In this case, we modify the resulting JSON object BatchCount, and TimeStamp properties, to better simulate real-time data. In your case, if your values are not simulated, these values should be set to their proper values in the PHP code. The JSON object represented by result is pushed into the SPCChart.SampleData.SampleIntervalRecords as a new array element. Then the entire TimeXBarR JSON script is converted to a string (JSON.stringify) and used to create a new chart, using pushJSONChartCreate, taking into account any new data values which have been added.

```
    TimeXBarR.SPCChart.SampleData.SampleIntervalRecords.push(result);
    pushJSONChartCreate(JSON.stringify(TimeXBarR));
```

The Javascript code for the setInterval event was extracted from the file phpPushDataExample.html, which is the HTML file that would reside on your server, along with the phpjsondata.php file, and the .js file, which contains the initial JSON definition of the TimeXBarR. Note that in TimeXBarR definition, the SampleData block contains a SampleIntervalRecords definition which is defined as an empty array. It is to this array you are adding sample interval records to with the call to TimeXBarR.SPCChart.SampleData.SampleIntervalRecords.push(result);

```
var TimeXBarR=
 {
          "StaticProperties": {
              "Canvas": {
                  "Width": 1024,
                  "Height": 768
              }
          },
          "SPCChart": {
              "InitChartProperties": {
                  "SPCChartType": "MEAN_RANGE_CHART",
                  "ChartMode": "Time",
                  "NumSamplesPerSubgroup": 5,
                  "NumDatapointsInView": 12,
                  "TimeIncrementMinutes": 15
              },
              "Scrollbar": {
                  "EnableScrollBar": true,
                  "ScrollbarPosition": "SCROLLBAR_POSITION_MIN"
              },
              "SampleData": {
                  "SampleIntervalRecords": []
              },

             .
             .
             .

              "Events": {
                  "EnableDataToolTip": true,
                  "EnableNotesToolTip": true

              },
              "Methods": {
                  "AutoCalculateControlLimits": true,
                  "AutoScaleYAxes": true,
                  "RebuildUsingCurrentData": true
              }
          }
      };
```

Special Note – You will probably need to copy an installation of the QCSPCChartGWTWar folder to an acutal server before you will able to get the phpPushDataExample.html example to work properly. While it is supposed to be possible, we weren't able to execute php files using the Eclipse or Visual Studio development environments, using their built-in, local servers.

It is also assumed that you have a working installation of PHP installed on your server. Most Linux systems have it by default, but you may need add it to your Windows Server Installation. See the web site http://php.net/ for details concerning downloads and installation. It's free.

Update Chart Data Using pushJSONChartUpdate with PHP

Files: phpUpdateExample.html, phpupdatedata.php, chartDefPHPExample.js

The previous example demonstrates how retrieve data from a PHP file in JSON format, and append it data to the current Javascript definition of a chart, and then update the web page with the chart using the pushJSONChartCreate function. In the next example, an entire SPCChart.SampleData JSON block will be acquired from a PHP service. The PHP structures used to define the data block become much more complicated. In this case, we took an example JSON block we new was correct, and used a JSON to PHP translation tool found here: http://php.fnlist.com/php/json_decode , to translate it into PHP.

```
{
    "SPCChart": {
        "SampleData": {
            "SampleIntervalRecords": [
                {
                    "SampleValues": [
                        27.53131515148628,
                        33.95771604022404,
                        24.310097827061817,
                        28.282642847792765,
                        30.2908518818265
                    ],
                    "BatchCount": 0,
                    "TimeStamp": 1371830829074,
                    "Note": ""
                },
                {
                    "SampleValues": [
                        27.444285005240214,
                        34.38930645615096,
                        28.0203674441636,
                        33.27153359969366,
                        36.8305571558275
                    ],
                    "BatchCount": 1,
                    "TimeStamp": 1371831729074,
                    "Note": ""
                },
                {
                    "SampleValues": [
                        35.21321620109259,
                        32.93940741018088,
                        33.66485557976163,
                        34.17314124609133,
                        24.576683179863725
                    ],
                    "BatchCount": 2,
                    "TimeStamp": 1371832629074,
                    "Note": ""
                },
                {
                    "SampleValues": [
                        27.898302097237174,
                        25.906531082892915,
                        26.950768095191137,
                        30.812058501916457,
                        31.085075984847936
                    ],
                    "BatchCount": 3,
                    "TimeStamp": 1371833529074,
                    "Note": ""
                }
            ]
        },
        "Methods": {
            "AutoCalculateControlLimits": true,
            "AutoScaleYAxes": true,
            "RebuildUsingCurrentData": true
        }
    }
}
```

The resulting PHP code looks like:

```
stdClass::__set_state(array(
   'SPCChart' =>
  stdClass::__set_state(array(
     'SampleData' =>
    stdClass::__set_state(array(
       'SampleIntervalRecords' =>
      array (
        0 =>
        stdClass::__set_state(array(
           'SampleValues' =>
          array (
            0 => 27.531315151486279,
            1 => 33.957716040224042,
            2 => 24.310097827061817,
            3 => 28.282642847792765,
            4 => 30.290851881826502,
          ),
           'BatchCount' => 0,
           'TimeStamp' => 1371830829074,
           'Note' => '',
        )),
        1 =>
        stdClass::__set_state(array(
           'SampleValues' =>
          array (
            0 => 27.444285005240214,
            1 => 34.389306456150962,
            2 => 28.0203674441636,
            3 => 33.271533599693662,
            4 => 36.830557155827499,
          ),
           'BatchCount' => 1,
           'TimeStamp' => 1371831729074,
           'Note' => '',
        )),
        2 =>
        stdClass::__set_state(array(
           'SampleValues' =>
          array (
            0 => 35.213216201092592,
            1 => 32.939407410180877,
            2 => 33.664855579761628,
            3 => 34.17314124609133,
            4 => 24.576683179863725,
          ),
           'BatchCount' => 2,
           'TimeStamp' => 1371832629074,
           'Note' => '',
        )),
        3 =>
        stdClass::__set_state(array(
           'SampleValues' =>
          array (
            0 => 27.898302097237174,
            1 => 25.906531082892915,
            2 => 26.950768095191137,
            3 => 30.812058501916457,
            4 => 31.085075984847936,
          ),
           'BatchCount' => 3,
           'TimeStamp' => 1371833529074,
           'Note' => '',
        )),
      ),
    )),
     'Methods' =>
    stdClass::__set_state(array(
       'AutoCalculateControlLimits' => true,
       'AutoScaleYAxes' => true,
       'RebuildUsingCurrentData' => true,
    )),
  )),
));
```

The **stdClass::__set_state** macro is useless in this case, so remove all of them (replace **stdClass::__set_state** with nothing) . Also, you can remove the explicit array indexing, 0 =>, 1 =>, 2 =>, etc. The JSON=>PHP translation does leave in trailing commas on the last element of arrays, but since they are removed properly in the **json_encode call**, just leave them it, rather than risk deleting one which is critical. The final file (phpupdatedata.php) looks like:

```
<?php
header('Content-type: application/json; charset=utf-8');
srand();


$ret = (array(
   'SPCChart' =>
  (array(
     'SampleData' =>
    (array(
       'SampleIntervalRecords' =>
      array (
        (array(
           'SampleValues' =>
          array (
             27.531315151486279,
             33.957716040224042,
             24.310097827061817,
             28.282642847792765,
             30.290851881826502,
          ),
           'BatchCount' => 0,
           'TimeStamp' => 1371830829074,
           'Note' => '',
        )),
        (array(
           'SampleValues' =>
          array (
             27.444285005240214,
             34.389306456150962,
             28.0203674441636,
             33.271533599693662,
             36.830557155827499,
          ),
           'BatchCount' => 1,
           'TimeStamp' => 1371831729074,
           'Note' => '',
        )),
        (array(
           'SampleValues' =>
          array (
            35.213216201092592,
            32.939407410180877,
            33.664855579761628,
            34.173141246091333,
            24.576683179863725,
          ),
           'BatchCount' => 2,
           'TimeStamp' => 1371832629074,
           'Note' => '',
        )),
        (array(
           'SampleValues' =>
          array (
            27.898302097237174,
            25.906531082892915,
            26.950768095191137,
            30.812058501916457,
            31.085075984847936,
          ),
           'BatchCount' => 3,
           'TimeStamp' => 1371833529074,
           'Note' => '',
        )),
      ),
    )),
     'Methods' =>
    (array(
       'AutoCalculateControlLimits' => true,
       'AutoScaleYAxes' => true,
       'RebuildUsingCurrentData' => true,
    )),
  )),
));


     echo json_encode($ret);

exit();
?>
```

Setup the setTimeout function using the following code. Note, that since were are just updating the currently defined chart, and not defining a new from scratch, we call the **pushJSONChartUpdate** function. The code segment below is extracted from the phpUpdateExample.html file.

```
$(window).load(function () {
    timerID = setTimeout(function () {
        $.ajax({
            url: "phpupdatedata.php",
            type: "GET",
            dataType: "json",
            error: function (jqXHR, textStatus, errorThrown) {
                alert("Error");

            },
            success: function (result) {
                // alert(JSON.stringify(result));
                pushJSONChartUpdate(JSON.stringify(result));

            }
        })
    }, 1000);
});
```

Create a New Chart from PHP Script

Files: phpCreateExample.html, phpCreateChart.php, chartDefPHPExample.js

Below is an example which uses PHP to create a completely new chart, with data. Since it is a completely new chart, the **pushJSONChartCreate** method is used. In the example below, we change the current chart to an Individual-Range chart. The JSON of the chart we want to create is:

```
{
    "SPCChart": {
        "InitChartProperties": {
            "SPCChartType": "INDIVIDUAL_RANGE_CHART",
            "ChartMode": "Batch",
            "NumSamplesPerSubgroup": 1,
            "NumDatapointsInView": 12,
            "TimeIncrementMinutes": 15
        },
        "ChartPositioning": {
            "GraphStartPosX": 0.125
        },
        "Scrollbar": {
            "EnableScrollBar": true
        },
        "TableSetup": {
            "HeaderStringsLevel": "HEADER_STRINGS_LEVEL3",
            "EnableInputStringsDisplay": true,
            "EnableCategoryValues": false,
            "EnableCalculatedValues": false,
            "EnableTotalSamplesValues": false,
            "EnableNotes": false,
            "EnableTimeValues": true,
            "EnableNotesToolTip": true,
            "TableBackgroundMode": "TABLE_SINGLE_COLOR_BACKGROUND_GRIDCELL",
            "BackgroundColor1": "WHITE",
            "BackgroundColor2": "GRAY",
            "TableAlarmEmphasisMode": "ALARM_HIGHLIGHT_BAR",
            "ChartAlarmEmphasisMode": "ALARM_HIGHLIGHT_SYMBOL",
            "ChartData": {
                "Title": "Individual Range Chart",
                "PartNumber": "283501",
                "ChartNumber": "17",
                "PartName": "TransmissionCasingBolt",
                "Operation": "Threading",
                "SpecificationLimits": "27.0 to 35.0",
                "Operator": "J.Fenamore",
                "Machine": "#11",
                "Gauge": "#8645",
                "UnitOfMeasure": "0.0001inch",
                "ZeroEquals": "zero",
                "DateString": "7/04/2013",
                "NotesMessage": "ControllimitspreparedMay10",
                "NotesHeader": "NOTES"
            }
        },
        "Events": {
            "EnableDataToolTip": true,
            "AlarmStateEventEnable": true
        },
        "SampleData": {
            "SampleIntervalRecords": [
                {
                    "SampleValues": [
                        27.53131515148628
                    ],
                    "BatchCount": 0,
                    "TimeStamp": 1371830829074,
                    "Note": ""
                },
                {
                    "SampleValues": [
                        27.444285005240214
                    ],
                    "BatchCount": 1,
                    "TimeStamp": 1371831729074,
                    "Note": ""
                },
                {
                    "SampleValues": [
                        35.21321620109259
                    ],
                    "BatchCount": 2,
                    "TimeStamp": 1371832629074,
                    "Note": ""
                },
                {
                    "SampleValues": [
                        27.898302097237174
                    ],
                    "BatchCount": 3,
                    "TimeStamp": 1371833529074,
                    "Note": ""
                },
                {
                    "SampleValues": [
                        22.94549873989527
                    ],
                    "BatchCount": 4,
                    "TimeStamp": 1371834429074,
                    "Note": ""
            },
            {
                "SampleValues": [
                    25.757197681039116
                ],
                "BatchCount": 5,
                "TimeStamp": 1371835329074,
                "Note": ""
            },
            {
                "SampleValues": [
                    34.04503169439933
                ],
                "BatchCount": 6,
                "TimeStamp": 1371836229074,
                "Note": ""
            },
            {
                "SampleValues": [
                    33.893820803904475
                ],
                "BatchCount": 7,
                "TimeStamp": 1371837129074,
                "Note": ""
            },
            {
                "SampleValues": [
                    35.85689222409033
                ],
                "BatchCount": 8,
                "TimeStamp": 1371838029074,
                "Note": ""
            },
            {
                "SampleValues": [
                    30.749686153841764
                ],
                "BatchCount": 9,
                "TimeStamp": 1371838929074,
                "Note": ""
            },
            {
                "SampleValues": [
                    35.27238495008025
                ],
                "BatchCount": 10,
                "TimeStamp": 1371839829074,
                "Note": ""
            }
        ]
    },
    "Methods": {
        "AutoCalculateControlLimits": true,
        "AutoScaleYAxes": true,
        "RebuildUsingCurrentData": true
        }
    }
}
```

The JSON → PHP conversion results in:

```
stdClass::__set_state(array(
   'SPCChart' =>
  stdClass::__set_state(array(
     'InitChartProperties' =>
    stdClass::__set_state(array(
       'SPCChartType' => 'INDIVIDUAL_RANGE_CHART',
       'ChartMode' => 'Batch',
       'NumSamplesPerSubgroup' => 1,
       'NumDatapointsInView' => 12,
       'TimeIncrementMinutes' => 15,
    )),
     'ChartPositioning' =>
    stdClass::__set_state(array(
       'GraphStartPosX' => 0.125,
    )),
     'Scrollbar' =>
    stdClass::__set_state(array(
       'EnableScrollBar' => true,
    )),
     'TableSetup' =>
    stdClass::__set_state(array(
       'HeaderStringsLevel' => 'HEADER_STRINGS_LEVEL3',
       'EnableInputStringsDisplay' => true,
       'EnableCategoryValues' => false,
       'EnableCalculatedValues' => false,
       'EnableTotalSamplesValues' => false,
       'EnableNotes' => false,
       'EnableTimeValues' => true,
       'EnableNotesToolTip' => true,
       'TableBackgroundMode' => 'TABLE_SINGLE_COLOR_BACKGROUND_GRIDCELL',
       'BackgroundColor1' => 'WHITE',
       'BackgroundColor2' => 'GRAY',
       'TableAlarmEmphasisMode' => 'ALARM_HIGHLIGHT_BAR',
       'ChartAlarmEmphasisMode' => 'ALARM_HIGHLIGHT_SYMBOL',
       'ChartData' =>
      stdClass::__set_state(array(
         'Title' => 'Individual Range Chart',
         'PartNumber' => '283501',
         'ChartNumber' => '17',
         'PartName' => 'TransmissionCasingBolt',
         'Operation' => 'Threading',
         'SpecificationLimits' => '27.0 to 35.0',
         'Operator' => 'J.Fenamore',
         'Machine' => '#11',
         'Gauge' => '#8645',
         'UnitOfMeasure' => '0.0001inch',
         'ZeroEquals' => 'zero',
     'DateString' => '7/04/2013',
     'NotesMessage' => 'ControllimitspreparedMay10',
     'NotesHeader' => 'NOTES',
  )),
)),
 'Events' =>
stdClass::__set_state(array(
   'EnableDataToolTip' => true,
   'AlarmStateEventEnable' => true,
)),
 'SampleData' =>
stdClass::__set_state(array(
   'SampleIntervalRecords' =>
  array (
    0 =>
    stdClass::__set_state(array(
       'SampleValues' =>
      array (
        0 => 27.531315151486279,
      ),
       'BatchCount' => 0,
       'TimeStamp' => 1371830829074,
       'Note' => '',
    )),
    1 =>
    stdClass::__set_state(array(
       'SampleValues' =>
      array (
        0 => 27.444285005240214,
      ),
       'BatchCount' => 1,
       'TimeStamp' => 1371831729074,
       'Note' => '',
    )),
    2 =>
    stdClass::__set_state(array(
       'SampleValues' =>
      array (
        0 => 35.213216201092592,
      ),
       'BatchCount' => 2,
       'TimeStamp' => 1371832629074,
       'Note' => '',
    )),
    3 =>
    stdClass::__set_state(array(
       'SampleValues' =>
      array (
        0 => 27.898302097237174,
      ),
       'BatchCount' => 3,
       'TimeStamp' => 1371833529074,
       'Note' => '',
    )),
    4 =>
```

```
stdClass::__set_state(array(
   'SampleValues' =>
  array (
    0 => 22.945498739895271,
  ),
   'BatchCount' => 4,
   'TimeStamp' => 1371834429074,
   'Note' => '',
)),
5 =>
stdClass::__set_state(array(
   'SampleValues' =>
  array (
    0 => 25.757197681039116,
  ),
   'BatchCount' => 5,
   'TimeStamp' => 1371835329074,
   'Note' => '',
)),
6 =>
stdClass::__set_state(array(
   'SampleValues' =>
  array (
    0 => 34.045031694399327,
  ),
   'BatchCount' => 6,
   'TimeStamp' => 1371836229074,
   'Note' => '',
)),
7 =>
stdClass::__set_state(array(
   'SampleValues' =>
  array (
    0 => 33.893820803904475,
  ),
   'BatchCount' => 7,
   'TimeStamp' => 1371837129074,
   'Note' => '',
)),
8 =>
stdClass::__set_state(array(
   'SampleValues' =>
  array (
    0 => 35.856892224090331,
  ),
   'BatchCount' => 8,
   'TimeStamp' => 1371838029074,
   'Note' => '',
)),
9 =>
stdClass::__set_state(array(
   'SampleValues' =>
  array (
    0 => 30.749686153841765,
  ),
           'BatchCount' => 9,
           'TimeStamp' => 1371838929074,
           'Note' => '',
        )),
        10 =>
        stdClass::__set_state(array(
           'SampleValues' =>
          array (
            0 => 35.272384950080252,
          ),
           'BatchCount' => 10,
           'TimeStamp' => 1371839829074,
           'Note' => '',
        )),
      ),
    )),
     'Methods' =>
    stdClass::__set_state(array(
       'AutoCalculateControlLimits' => true,
       'AutoScaleYAxes' => true,
       'RebuildUsingCurrentData' => true,
    )),
  )),
));
```

After removing the **stdClass::__set_state** macros, and the explicit array indexing, the resulting PHP file looks like:

```
<?php
header('Content-type: application/json; charset=utf-8');
srand();


$ret =
(array(
   'SPCChart' =>
  (array(
     'InitChartProperties' =>
    (array(
       'SPCChartType' => 'INDIVIDUAL_RANGE_CHART',
       'ChartMode' => 'Batch',
       'NumSamplesPerSubgroup' => 1,
       'NumDatapointsInView' => 12,
       'TimeIncrementMinutes' => 15,
    )),
     'ChartPositioning' =>
    (array(
       'GraphStartPosX' => 0.125,
    )),
     'Scrollbar' =>
    (array(
       'EnableScrollBar' => true,
    )),
     'TableSetup' =>
    (array(
   'HeaderStringsLevel' => 'HEADER_STRINGS_LEVEL3',
   'EnableInputStringsDisplay' => true,
   'EnableCategoryValues' => false,
   'EnableCalculatedValues' => false,
   'EnableTotalSamplesValues' => false,
   'EnableNotes' => false,
   'EnableTimeValues' => true,
   'EnableNotesToolTip' => true,
   'TableBackgroundMode' => 'TABLE_SINGLE_COLOR_BACKGROUND_GRIDCELL',
   'BackgroundColor1' => 'WHITE',
   'BackgroundColor2' => 'GRAY',
   'TableAlarmEmphasisMode' => 'ALARM_HIGHLIGHT_BAR',
   'ChartAlarmEmphasisMode' => 'ALARM_HIGHLIGHT_SYMBOL',
   'ChartData' =>
  (array(
     'Title' => 'Individual Range Chart',
     'PartNumber' => '283501',
     'ChartNumber' => '17',
     'PartName' => 'TransmissionCasingBolt',
     'Operation' => 'Threading',
     'SpecificationLimits' => '27.0 to 35.0',
     'Operator' => 'J.Fenamore',
     'Machine' => '#11',
     'Gauge' => '#8645',
     'UnitOfMeasure' => '0.0001inch',
     'ZeroEquals' => 'zero',
     'DateString' => '7/04/2013',
     'NotesMessage' => 'ControllimitspreparedMay10',
     'NotesHeader' => 'NOTES',
  )),
)),
 'Events' =>
(array(
   'EnableDataToolTip' => true,
   'AlarmStateEventEnable' => true,
)),
 'SampleData' =>
(array(
   'SampleIntervalRecords' =>
  array (
    (array(
       'SampleValues' =>
      array (
        27.531315151486279,
      ),
       'BatchCount' => 0,
       'TimeStamp' => 1371830829074,
       'Note' => '',
    )),
    (array(
       'SampleValues' =>
      array (
        27.444285005240214,
      ),
       'BatchCount' => 1,
       'TimeStamp' => 1371831729074,
       'Note' => '',
    )),
    (array(
       'SampleValues' =>
      array (
        35.213216201092592,
      ),
       'BatchCount' => 2,
       'TimeStamp' => 1371832629074,
       'Note' => '',
    )),
    (array(
       'SampleValues' =>
      array (
        27.898302097237174,
      ),
       'BatchCount' => 3,
       'TimeStamp' => 1371833529074,
       'Note' => '',
    )),
    (array(
       'SampleValues' =>
      array (
        22.945498739895271,
      ),
       'BatchCount' => 4,
       'TimeStamp' => 1371834429074,
       'Note' => '',
    )),
    (array(
       'SampleValues' =>
      array (
        25.757197681039116,
      ),
       'BatchCount' => 5,
       'TimeStamp' => 1371835329074,
       'Note' => '',
    )),
    (array(
       'SampleValues' =>
      array (
        34.045031694399327,
      ),
       'BatchCount' => 6,
       'TimeStamp' => 1371836229074,
       'Note' => '',
    )),
    (array(
       'SampleValues' =>
      array (
        33.893820803904475,
      ),
       'BatchCount' => 7,
       'TimeStamp' => 1371837129074,
               'Note' => '',
            )),
            (array(
               'SampleValues' =>
              array (
                35.856892224090331,
              ),
               'BatchCount' => 8,
               'TimeStamp' => 1371838029074,
               'Note' => '',
            )),
            (array(
               'SampleValues' =>
              array (
                30.749686153841765,
              ),
               'BatchCount' => 9,
               'TimeStamp' => 1371838929074,
               'Note' => '',
            )),
            (array(
               'SampleValues' =>
              array (
                35.272384950080252,
              ),
               'BatchCount' => 10,
               'TimeStamp' => 1371839829074,
               'Note' => '',
            )),
          ),
        )),
         'Methods' =>
        (array(
           'AutoCalculateControlLimits' => true,
           'AutoScaleYAxes' => true,
           'RebuildUsingCurrentData' => true,
        )),
      )),
    ));

         echo json_encode($ret);

    exit();
    ?>
```

Setup the setTimeout function using the following code. Note, that since were are creating a new chart, we call the pushJSONChartCreate function. The code segment below is extracted from the phpCreateExample.js file.

```
   $(window).load(function () {
       timerID = setTimeout(function () {
           $.ajax({
            url: "phpcreatechart.php",
            type: "GET",
            dataType: "json",
            error: function (jqXHR, textStatus, errorThrown) {
                alert("Error");

            },
            success: function (result) {
                alert(JSON.stringify(result));

                pushJSONChartCreate(JSON.stringify(result));
            }
        })
    }, 1000);
});
```

Appendix

List of Color Constants:

	HEX	R	G	B
ALICEBLUE	0xfff0f8ff	0.422	0.059	1.000
ANTIQUEWHITE	0xfffaebd7	0.905	0.140	0.980
AQUA	0xff00ffff	0.500	1.000	1.000
AQUAMARINE	0xff7fffd4	0.556	0.502	1.000
AZURE	0xfff0ffff	0.500	0.059	1.000
BEIGE	0xfff5f5dc	0.833	0.102	0.961
BISQUE	0xffffe4c4	0.910	0.231	1.000
BLACK	0xff000000	0.000	0.000	0.000
BLANCHEDALMOND	0xffffebcd	0.900	0.196	1.000
BLUE	0xff0000ff	0.333	1.000	1.000
BLUEVIOLET	0xff8a2be2	0.247	0.810	0.886
BROWN	0xffa52a2a	0.000	0.745	0.647
BURLYWOOD	0xffdeb887	0.906	0.392	0.871
CADETBLUE	0xff5f9ea0	0.495	0.406	0.627
CHARTREUSE	0xff7fff00	0.750	1.000	1.000
CHOCOLATE	0xffd2691e	0.931	0.857	0.824
CORAL	0xffff7f50	0.955	0.686	1.000
CORNFLOWERBLUE	0xff6495ed	0.393	0.578	0.929
CORNSILK	0xfffff8dc	0.867	0.137	1.000
CRIMSON	0xffdc143c	0.033	0.909	0.863

CYAN	0xff00ffff			
DARKBLUE	0xff00008b	0.333	1.000	0.545
DARKCYAN	0xff008b8b	0.500	1.000	0.545
DARKGOLDENROD	0xffb8860b	0.882	0.940	0.722
DARKGRAY	0xffa9a9a9	0.000	0.000	0.663
DARKGREY	0xffa9a9a9	0.000	0.000	0.663
DARKGREEN	0xff006400	0.667	1.000	0.392
DARKKHAKI	0xffbdb76b	0.846	0.434	0.741
DARKMAGENTA	0xff8b008b	0.167	1.000	0.545
DARKOLIVEGREEN	0xff556b2f	0.772	0.561	0.420
DARKORANGE	0xffff8c00	0.908	1.000	1.000
DARKORCHID	0xff9932cc	0.222	0.755	0.800
DARKRED	0xff8b0000	0.000	1.000	0.545
DARKSALMON	0xffe9967a	0.958	0.476	0.914
DARKSEAGREEN	0xff8fbc8f	0.667	0.239	0.737
DARKSLATEBLUE	0xff483d8b	0.310	0.561	0.545
DARKSLATEGRAY	0xff2f4f4f	0.500	0.405	0.310
DARKSLATEGREY	0xff2f4f4f	0.500	0.405	0.310
DARKTURQUOISE	0xff00ced1	0.498	1.000	0.820
DARKVIOLET	0xff9400d3	0.216	1.000	0.827
DEEPPINK	0xffff1493	0.090	0.922	1.000
DEEPSKYBLUE	0xff00bfff	0.458	1.000	1.000
DIMGRAY	0xff696969	0.000	0.000	0.412
DIMGREY	0xff696969	0.000	0.000	0.412
DODGERBLUE	0xff1e90ff	0.418	0.882	1.000
FIREBRICK	0xffb22222	0.000	0.809	0.698
FLORALWHITE	0xfffffaf0	0.889	0.059	1.000

FORESTGREEN	0xff228b22	0.667	0.755	0.545
FUCHSIA	0xffff00ff	0.167	1.000	1.000
GAINSBORO	0xffdcdcdc	0.000	0.000	0.863
GHOSTWHITE	0xfff8f8ff	0.333	0.027	1.000
GOLDENROD	0xffdaa520	0.881	0.853	0.855
GRAY	0xff808080	0.000	0.000	0.502
GREY	0xff808080	0.000	0.000	0.502
GREEN	0xff008000	0.667	1.000	0.502
GREENYELLOW	0xffadff2f	0.768	0.816	1.000
HONEYDEW	0xfff0fff0	0.667	0.059	1.000
HOTPINK	0xffff69b4	0.083	0.588	1.000
INDIANRED	0xffcd5c5c	0.000	0.551	0.804
INDIGO	0xff4b0082	0.237	1.000	0.510
IVORY	0xfffffff0	0.833	0.059	1.000
KHAKI	0xfff0e68c	0.850	0.417	0.941
LAVENDER	0xffe6e6fa	0.333	0.080	0.980
LAVENDERBLUSH	0xfffff0f5	0.056	0.059	1.000
LAWNGREEN	0xff7cfc00	0.749	1.000	0.988
LEMONCHIFFON	0xfffffacd	0.850	0.196	1.000
LIGHTBLUE	0xffadd8e6	0.459	0.248	0.902
LIGHTCORAL	0xfff08080	0.000	0.467	0.941
LIGHTCYAN	0xffe0ffff	0.500	0.122	1.000
LIGHTGOLDENRODYELLOW	0xfffafad2	0.833	0.160	0.980
LIGHTGREEN	0xff90ee90	0.667	0.395	0.933
LIGHTGREY	0xffd3d3d3	0.000	0.000	0.827
LIGHTGRAY	0xffd3d3d3	0.000	0.000	0.827
LIGHTPINK	0xffffb6c1	0.025	0.286	1.000
LIGHTSALMON	0xffffa07a	0.952	0.522	1.000

LIGHTSEAGREEN	0xff20b2aa	0.509	0.820	0.698
LIGHTSKYBLUE	0xff87cefa	0.436	0.460	0.980
LIGHTSLATEGRAY	0xff778899	0.417	0.222	0.600
LIGHTSLATEGREY	0xff778899	0.417	0.222	0.600
LIGHTSTEELBLUE	0xffb0c4de	0.406	0.207	0.871
LIGHTYELLOW	0xffffffe0	0.833	0.122	1.000
LIME	0xff00ff00	0.667	1.000	1.000
LIMEGREEN	0xff32cd32	0.667	0.756	0.804
LINEN	0xfffaf0e6	0.917	0.080	0.980
MAGENTA	0xffff00ff	0.167	1.000	1.000
MAROON	0xff800000	0.000	1.000	0.502
MEDIUMAQUAMARINE	0xff66cdaa	0.557	0.502	0.804
MEDIUMBLUE	0xff0000cd	0.333	1.000	0.804
MEDIUMORCHID	0xffba55d3	0.200	0.597	0.827
MEDIUMPURPLE	0xff9370db	0.279	0.489	0.859
MEDIUMSEAGREEN	0xff3cb371	0.592	0.665	0.702
MEDIUMSLATEBLUE	0xff7b68ee	0.310	0.563	0.933
MEDIUMSPRINGGREEN	0xff00fa9a	0.564	1.000	0.980
MEDIUMTURQUOISE	0xff48d1cc	0.506	0.656	0.820
MEDIUMVIOLETRED	0xffc71585	0.105	0.894	0.780
MIDNIGHTBLUE	0xff191970	0.333	0.777	0.439
MINTCREAM	0xfff5fffa	0.583	0.039	1.000
MISTYROSE	0xffffe4e1	0.983	0.118	1.000
MOCCASIN	0xffffe4b5	0.894	0.290	1.000
NAVAJOWHITE	0xffffdead	0.900	0.322	1.000
NAVY	0xff000080	0.333	1.000	0.502
OLDLACE	0xfffdf5e6	0.891	0.091	0.992

OLIVE	0xff808000	0.833	1.000	0.502
OLIVEDRAB	0xff6b8e23	0.779	0.754	0.557
ORANGE	0xffffa500	0.892	1.000	1.000
ORANGERED	0xffff4500	0.955	1.000	1.000
ORCHID	0xffda70d6	0.160	0.486	0.855
PALEGOLDENROD	0xffeee8aa	0.848	0.286	0.933
PALEGREEN	0xff98fb98	0.667	0.394	0.984
PALETURQUOISE	0xffafeeee	0.500	0.265	0.933
PALEVIOLETRED	0xffdb7093	0.055	0.489	0.859
PAPAYAWHIP	0xffffefd5	0.897	0.165	1.000
PEACHPUFF	0xffffdab9	0.921	0.275	1.000
PINK	0xffffc0cb			
POWDERBLUE	0xffb0e0e6	0.481	0.235	0.902
PURPLE	0xff800080	0.167	1.000	0.502
RED	0xffff0000	0.000	1.000	1.000
ROSYBROWN	0xffbc8f8f	0.000	0.239	0.737
ROYALBLUE	0xff4169e1	0.375	0.711	0.882
SADDLEBROWN	0xff8b4513	0.931	0.863	0.545
SALMON	0xfffa8072	0.983	0.544	0.980
SANDYBROWN	0xfff4a460	0.923	0.607	0.957
SEAGREEN	0xff2e8b57	0.593	0.669	0.545
SEASHELL	0xfffff5ee	0.931	0.067	1.000
SIENNA	0xffa0522d	0.946	0.719	0.627
SILVER	0xffc0c0c0	0.000	0.000	0.753
SKYBLUE	0xff87ceeb	0.452	0.426	0.922
SLATEBLUE	0xff6a5acd	0.310	0.561	0.804
SLATEGRAY	0xff708090	0.417	0.222	0.565
SLATEGREY	0xff708090	0.417	0.222	0.565

SPRINGGREEN	0xff00ff7f	0.584	1.000	1.000
STEELBLUE	0xff4682b4	0.424	0.611	0.706
TAN	0xffD2B48C			
TEAL	0xff008080	0.500	1.000	0.502
THISTLE	0xffd8bfd8	0.167	0.116	0.847
TOMATO	0xffff6347	0.975	0.722	1.000
TURQUOISE	0xff40e0d0	0.517	0.714	0.878
VIOLET	0xffee82ee			
WHEAT	0xffF5DEB3			
WHITE	0xffffffff			
WHITESMOKE	0xffF5F5F5			
YELLOW	0xffFFFF00			
YELLOWGREEN	0xff9ACD32			
TABLEGREEN	0xff00ffcc			
EMPTYCOLOR	0x00000000			
TRANSPARENT	0x00000000			

Index