



Hyper Historian

Performance Calculations and Functions



APPLICATION NOTE

September 2013

Description: A quick guide to start using Hyper Historian Performance Calculations and Functions

OS Requirement: Windows 8 x64 (Professional or Enterprise Edition), Windows 7 x64 (Professional or Enterprise Edition), Windows Server 2012 x64, Windows Server 2008 R2 x64, Windows Vista x64 SP2 (Business, Ultimate, or Enterprise Edition), Windows Server 2008 x64

General Requirement: Installation of GENESIS64 v10.80 and basic knowledge of Hyper Historian and GENESIS64

Introduction

ICONICS Hyper Historian is a powerful, 64-bit, high performance plant historian. The Hyper Historian provides many features and functions for most mission critical applications demanding the highest level of availability and provides the highest performance possible by taking advantage of 64-bit hardware and software architectures, allowing it to access more CPU power and memory than traditional 32-bit-based historians.

One of the key features of the Hyper Historian is to support extensive data calculation on live and historical data using the new Performance Calculation engine.

This application note will show how to the configure Hyper Historian Performance Calculations for a SPC style $\bar{x}$ and R-chart.

This application note will also cover how to create a Performance Calculation Function which totalizes a Hyper Historian tag and divides it by a custom period.

1. Logging SPC Calculated Tag

The Raw Data Tag

The Performance Calculations are executed against raw data that is logged by the Hyper Historian. So for anything that needs a performance calculation the standard logging tag(s) need to be configured first. To create the calculation for the $\bar{x}$ and R-chart we need a single logging value (the x-value).

The Hyper Historian will automatically make any value that is being logged available for data replay. For this purpose all historical logging tags are also exposed as OPC Data Access tags.

1. In the Workbench open the Hyper Historian provider, add a new Logging Group by right-clicking on **Groups** and choosing “Logging Group”

2. Set the name of the new Logging Group to “PerfCalc” and choose one of the available File Loggers on your system.
3. Under the newly created Logging Group create a new Collector Group by right-clicking on the **PerfCalc** Logging Group and choosing “Collector Group”
4. Set the name of the new Collector Group to “Data” and choose one of the available Collectors on your system.
5. Create new data collections tag by right clicking on the **Data Collections** and choosing Tag
6. Set the name of the new tag to “Source” and the data type to Float32
7. Click on the ellipsis button next to the Signal Name field to open the Data Browser and select: Home > OPC Classic Server > My Server > Simulations > Float > Random, then click in the signal name part of the data browser and change the period to 30, the minimum to 30, the maximum to 50, and the phase to 0 as shown in Figure 1.

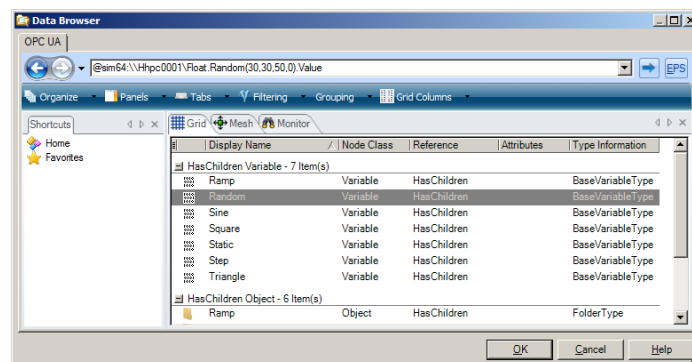


Figure 1 Data Browser

8. Change the **In Group** property of the tag to PerfCalc \ Data.

This completes the configuration of the base (raw data) tag that can then be used in the calculation tags.

The Counters

In SPC calculations it is important to know what samples and measurements you want to include in the calculations.

For that purpose we will create two calculated tags, one to count measurements, i.e. every change of the source value increments the counter and which will reset after a set number of measurements have been received in this case the reset will occur after 5 measurements. The other counter will count the number of samples, which will be incremented every time the 5 measurements have been taken.

As calculation tags are (re-)calculated on a trigger as part of the counter configuration the triggers will also be defined.



Hyper Historian

Performance Calculations and Functions



APPLICATION NOTE

September 2013

Measurement Data Trigger

1. Right click on the **Calculation Triggers** folder within the Hyper Historian Provider and choose “Data Trigger”
2. Change the name to “Measurement” and set the trigger type to “Any Data Point Change”
3. Click on the ellipsis button next to the Data Point field to open the tag browser as shown in Figure 2.
4. Select the Source tag

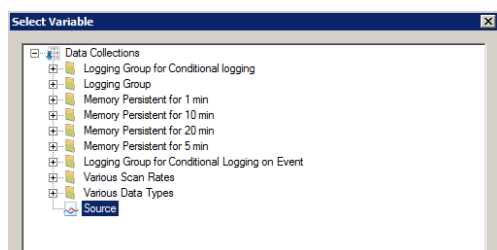


Figure 2 Tag Browser

Measurement Counter

With the trigger created the measurement counter calculation tag can be created

1. Right click on the **Data Collections** tree item and choose Calculated Tag
2. Set the name to “MeasurementCounter”
3. Click in the empty box under the Name column in the Triggers section, and type the letter “D”, this will be the name of the trigger, the name of the trigger is useful when different parts of the expression will have to run depending on which trigger caused the calculated tag to recalculate
4. Click on the button for the Trigger path selection to open the trigger browser as shown in Figure 3.

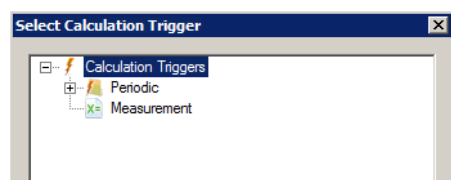


Figure 3 Select the Measurement Trigger

5. Select the previously created Measurement trigger.
6. Change the **In Group** property of the tag to PerfCalc.
7. Click on the Expression tab and enter the following expression (only start a newline after the ↵)

```
IF isgood(trgcur())↵
THEN↵
    IF isgood(tagprev({{calc:MeasurementCounter}}, now()))↵
    THEN (val(tagprev({{calc:MeasurementCounter}}, now())) + 1) % 5↵
    ELSE 1↵
ELSE nologval()↵
```

The first IF statement checks if the received update is of good quality, if not no value will be logged. The second IF statement (only used for good values) will check if the previously logged value was good, and if so, it will add 1 to the running total and by using the modulus function (%) it will reset every 5 values, if no previous value can be found it will start with a value of 1.

The *trgcur()* function returns an data object with the name, value, and quality of the trigger.

The *isgood()* function returns true or false depending on the quality of the specified data object.

The *tagprev()* function takes two parameters, the first is the name of the tag, and the second is a time value, the function will find the previous value before the specified time.

The *nologval()* function is a special instruction to Hyper Historian to ensure it does not log a new value.

Sample Data Trigger

With the measurement counter in place, the trigger for the sample counter can be implemented. The trigger will execute only when new measurements are received

1. Right click on the **Calculation Triggers** folder within the Hyper Historian Provider and choose “Data Trigger”
2. Change the name to “Sample” and set the trigger type to “Any Data Point Change”
3. Click on the ellipsis button next to the Data Point field to open the tag browser as shown in Figure 4.
4. Select the MeasurementCounter tag from the tree

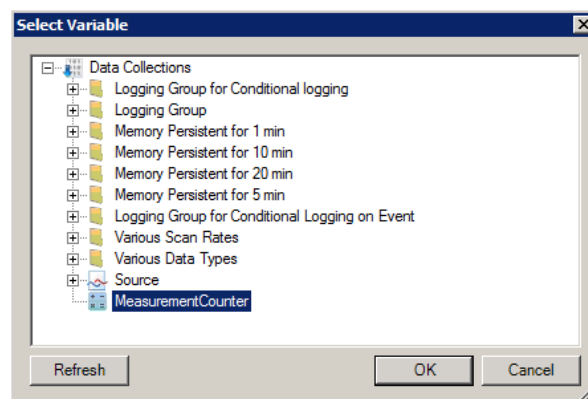


Figure 4 Select the Measurement Counter

Sample Counter

With the trigger created the sample counter calculation tag can be created

1. Right click on the **Data Collections** tree item and choose Calculated Tag
2. Set the name to “SampleCounter”



Hyper Historian

Performance Calculations and Functions



APPLICATION NOTE

September 2013

- Click in the empty box under the Name column in the Triggers section, and type the letter “D”
- Click on the button for the Trigger selection to open the trigger browser as shown in Figure 5.

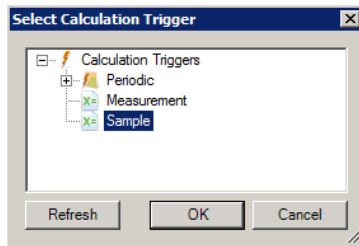


Figure 5 Select the Sample Trigger

- Select the previously created Sample trigger.
- Change the **In Group** property of the tag to PerfCalc.
- Click on the Expression tab and enter the following expression (only start a newline after the ↵)

```
IF isgood(trgcur()) && (val(trgcur()) == 0) ↵
THEN ↵
    IF isgood(tagprev({{calc:SampleCounter}}, now())) ↵
    THEN (val(tagprev({{calc:SampleCounter}}, now())) + 1) ↵
    ELSE 1 ↵
ELSE nologval() ↵
```

The first IF statement checks if the received update is of good quality and the value is zero, which means that 5 measurements have been taken, if not no value will be logged. The second IF statement (only used for good values) will check if the previously logged value was good and if so, it will add 1 to the running total, alternatively if no previous value can be found it will start with a value of 1.

SPC Data Calculation

With the measurement and sample counters in place, the SPC calculation can be implemented.

- Right click on the **Data Collections** tree item and choose Calculated Tag
- Set the name to “xbar”
- Click in the empty box under the Name column in the Triggers section, and type the letter “D”
- Click on the button for the Trigger selection to open the trigger browser as shown in Figure 6.

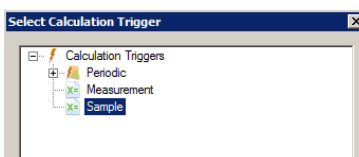


Figure 6 Select the Sample Trigger

- Select the previously created Sample trigger.
- Change the **In Group** property of the tag to PerfCalc.
- Click on the Expression tab and enter the following expression (only start a newline after the ↵)

```
IF isgood(trgcur()) && (val(trgcur()) == 0) ↵
THEN ↵
    tagavg({{data:source}}, time(tagprev({{calc:xbar}}, now()), now(), 80) ↵
ELSE nologval() ↵
```

The IF statement checks if the received update is of good quality and the value is zero, which means that 5 measurements have been taken, if not no value will be logged. If a successful value The *tagavg()* function calculates the average value over a given time range, it requires four parameters, a tag name, the start time, the end time, and a confidence value. In this example the start time is determined by using the time of the last logged sample. The end time is set to the current time, and the confidence value is set to 80, which means that at least 80% of the values in the time period must be good quality for the average value to have a good quality.

The *time()* function returns the actual time of the specified data object.

2. Logging Totalized Calculation Function

The totalizer function in Hyper Historian is return the total value of a logged tag within a set range, and is primarily used when monitoring a flow value rather than a cumulative value.

This section will cover how to create a calculation function to totalize a Hyper Historian tag over a set period, and then divide this value by a custom amount to give a totalized value per time

The Raw Data Tag

The Performance Calculations are executed against raw data that is logged by the Hyper Historian. So for anything that needs a performance calculation the standard logging tag(s) need to be configured first. In this case the Flow Value tag will need to be logged.

The Hyper Historian will automatically make any value that is being logged available for data replay. For this purpose all historical logging tags are also exposed as OPC Data Access tags.

- Create new data collections tag by right clicking on the **Data Collections** and choosing Tag



Hyper Historian

Performance Calculations and Functions



APPLICATION NOTE

September 2013

- Set the name of the new tag to “Flow” and the data type to Float32
- Click on the ellipsis button next to the Signal Name field to open the Data Browser and select: Home > OPC Classic Server > My Server > Simulations > Float > Ramp, then click in the signal name part of the data browser and change the period to 60, the minimum to 1, the maximum to 60, and the phase to 0.
- Change the **In Group** property of the tag to PerfCalc \ Data.

This completes the configuration of the base (raw data) tag that can then be used in the calculation tags. The next section will cover the Calculation Function to manipulate this tags value.

Calculation Function

The Calculation Function exposes all of the properties available in a Calculated Tag and is used to allow parameterization of the calculations so they can be reused across multiple calculated tags.

- In the Hyper Historian provider, add a new Calculation Folder by right-clicking on **Calculation Functions** and choosing “Calculation Folder”
- Set the name of the new Calculation Folder to “TotalizerFunctions”
- Under the newly created Calculation Folder create a new Calculation Function by right-clicking on the **TotalizerFunctions** folder and choosing “Calculation Function”
- Set the name of the new Calculation Function to “TotalizerFunction” and change the Returns type to “DataValue, a set of Timestamp, Value and Quality code”
- The following parameters will need to be added:

Name	Type
Input	variable
StartTime	startTime
EndTime	endTime
PercentGood	percentGood
CalculationPeriod	int

- Select the Expressions tab, and add the Totalizer function by selecting Functions > History Functions > tagtotalize(variable, startTime, endTime, percentGood). This will add the function into the expression editor
- The expression sections in red will need to be replaced for the function to be validated and accepted, first select the text “variable” and open the Variables browser by clicking on the **Variables** button, as shown in Figure 7 and double click the “Input” parameter

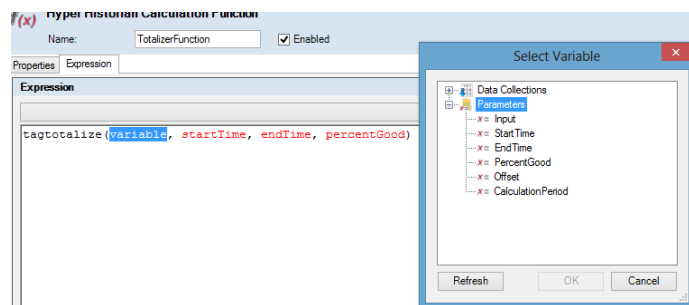


Figure 7 Variable Browser

- Repeat for the remaining parameters so the StartTime, EndTime and PercentGood parameters are filled in, as shown in Figure 8.

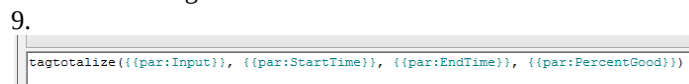


Figure 8 TagTotalize with Parameters

- The CalculationPeriod will be a number of seconds to represent the amount of time in seconds to show the Totalized Value (e.g. Flow Rate per hour would be 3600), the function will need to be divided by this value by appending a “/” and then browsing the parameter to the expression, as shown in Figure 11.

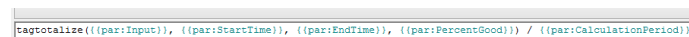


Figure 9 CalculationPeriod

- The total period of the totalized value needs to be calculated, this will be done by subtracting the StartTime parameter from the EndTime parameter this will be added to the end of the expression.
- The above calculation will need to be converted into a TimeSpan, by selecting the Functions > Date and Time Functions > totimespan(value) function.
- As the CalculationPeriod is in seconds, the above TimeSpan will need to be converted into seconds, this is achieved by wrapping it in the totalseconds() function by browsing Functions > Date and Time Functions > totalseconds()
- The entire division function will need to be wrapped in parenthesis to ensure it will calculate correctly.

The completed Expression should appear as follows:

```
tagtotalize({{par:Input}}, {{par:StartTime}}, {{par:EndTime}},
{{par:PercentGood}}) / ({{par:CalculationPeriod}} *
totalseconds(totimespan({{par:EndTime}} - {{par:StartTime}})))
```



Hyper Historian

Performance Calculations and Functions



APPLICATION NOTE

September 2013

This completes the configuration of the Calculation Function, which will be used in a Calculated Tag to be created further on.

Calculation Trigger

The Calculated tag will need to know how frequently it needs to evaluate its expression, so a new Calculation Trigger will need to be created.

1. Add a new Calculation Folder by right-clicking on **Calculation Triggers** and selecting “Trigger Folder”
2. Set the name of the new Trigger folder to “PerfTriggers”
3. Create a new Calculation Trigger by right-clicking on **PerfTriggers** and selecting Time Trigger
4. Set the name of the new Trigger to “Every Minute”
5. Change the properties of the trigger so the starting time is set to a whole minute at least a few hours ago in the past, e.g. Friday, 30 August 2013 14:20:00
6. Set the Recur every property to exactly 1 Mins

This completes the configuration of the Calculation Trigger, which will be used to tell the Calculated Tag how frequently to calculate.

Calculation Tag

Now the Calculation Function, Calculation Trigger and flow tag have been configured, the calculated tag will need to be created and configured to return the Totalized value for the periods defined.

1. In the Hyper Historian Data Collections section a new calculated tag will be created, by right-clicking on **Data Collections** and selecting “Calculated Tag”
2. Give the name of the Calculated Tag “TotalizedTag”
3. Connect the previously configured trigger by selecting the three ellipses button under the Triggers section and browsing to the “Every Minute” trigger inside the **PerfTriggers** folder
4. On the Expressions tab the Calculation Function will need to be browsed by selecting Functions > Calculation Functions > TotalizerFunction, this will add an expression into the editor as shown in Figure 10.

```
$TotalizerFunctions.TotalizerFunction(Input, StartTime, EndTime, PercentGood, CalculationPeriod)
```

Figure 10 TotalizerFunction

5. As with the Calculation Function, the items in red text will need to be replaced for the tag to evaluate correctly. The text named “Input” will need to be replaced with the Raw

Data Tag, by highlighting the text, clicking the **Variables...** button and browsing to the “Flow” tag

6. The “EndTime” text will need to be replaced with the time that the trigger fired, by highlighting the text and selecting Functions > Date and Time Functions > now()
7. The “StartTime” text will need to be replaced with the start time to retrieve the data from, in this example it will be 1 minute before the trigger time, by highlighting the text and selecting Functions > Date and Time Functions > now() and then subtracting Functions > Date and Time Functions > fromminutes(1)
8. The “PercentGood” text will need to be replaced with a number between 1-100 to indicate how many samples out of the section need to be good quality. Replace the text will 80
9. The “CalculationPeriod” text will need to be replaced by the calculation period to be divided by the total period, in this case we will use 15
10. The expression is now complete and should appear as follows:

```
$TotalizerFunctions.TotalizerFunction({{data:Flow}}, now() - fromminutes(1), now(), 80, 15)
```

Now this is complete, it will need to log for a few minutes and then the pen can be browsed to in the TrendWorX64 Viewer to show the TotalizedValue every 15 seconds.