

QualityWindow

The continuous improvement software



Calculated Variables

Formulas and Function Reference

October 19, 2026

Table of Contents

Understanding Calculated Variables	4
How calculations work	4
Set up the formula in QW Admin	4
A first example.....	4
Writing formulas	5
Refer to Variables	5
Arithmetic and parentheses	5
Using functions and missing values	6
Function syntax	6
Combine functions.....	6
Missing values.....	6
Practical calculation examples	7
Limit a calculated result	7
Select a result from a code	7
Assign a shift from the record time	7
Accumulate downtime minutes within a shift	7
Counters using input and output files	8
Configure the counter.....	8
Accumulate a production quantity	8
Mathematical functions	9
Trigonometry	9
Functions within one record	10
Choose between a record and a moving calculation	10
Moving and historical functions	11
Retrieve a previous value	12
Conditions and selection	13
Conditional functions	13
Comparison operators.....	13
Select from value ranges.....	13
Date and time functions	14
Convert fraction text	Error! Bookmark not defined.
Time differences	15

Calculated Variables - Formulas and Function Reference

Between consecutive records	15
From the record time to a numeric time	15
Between two Variables.....	15
Between two date and time pairs	15
Date and time Units matter	15
Limits and compliance	16
Retrieve a fixed limit or target	16
Count and summarize conformance	16
Summarize rule violations.....	16
Moving capability functions	17
Cpk and Ppk use different variation estimates	17
Other capability calculations	17
Review a formula before using its results	17

Understanding Calculated Variables

Calculated Variables derive their values from formulas. Use them to calculate results from entered measurements, apply conditions, summarize recent records, or maintain a counter using an input/output file.

This guide explains how to write formulas in Quality Window 6.1, then provides examples and a function reference grouped by purpose. Start with the setup and formula sections if you are creating a calculation for the first time.

How calculations work

Calculated Variables do not appear on the Add, Edit, Insert, or Copy data entry screens. QW determines their values when the record is saved. Their results can be charted, used in other calculations, and assigned rules and limits.

QW calculates Variables in ascending Variable-number order. If one Calculated Variable uses the result of another, the calculation it depends on must have a lower Variable number.

For example, calculate an average in V16 before using that result in V17. Reversing that dependency can produce an incorrect result.

Set up the formula in QW Admin

1. In the Variable configuration, use the Calculated Variable type and enter the expression in the formula setting.
2. Enter only the expression. For a constant value of 14, enter 14. Do not add the old CALC: prefix; it causes an error.
3. Use the Formula button to display available functions. Right-click in the formula area to display the available Variables.
4. Allow enough Variable length for the result, including a decimal point and a negative sign when needed.
5. Save the Variable. QW Admin checks for balanced parentheses and square brackets and checks function names against its valid function list. Correct any reported error before saving.

A first example

To calculate the average of V4, V5, and V6, enter:

```
@AVG[V4..V6]
```

If the values are 10, 12, and 14, the result is 12. If V6 is blank, the result is 11 because this function averages only the two entered values.

Writing formulas

A formula can contain constants, Variable references, arithmetic operators, parentheses, and functions. Do not include a Variable assignment such as V17= when entering the expression for V17.

Refer to Variables

V followed by a number identifies a Variable: V3, V4, and so on. V1 is the record date and V2 is the record time. A reference normally uses the value in the current record; moving functions can also read earlier records.

Numeric, Calculated, Average, Range, and Popup Variables can be used in formulas. A Popup Variable returns the key value selected for that popup.

Notation	Meaning	Example
V4	One Variable	V4 * 1.33
V4,V6,V9	Separate entries in a function list	@SUM[V4,V6,V9]
V4..V8	Consecutive Variables, including both ends	@AVG[V4..V8]
V4..V8,V12	A range plus another Variable	@AVG[V4..V8,V12]

Arithmetic and parentheses

Use parentheses to make the intended calculation clear. Powers are evaluated before multiplication and division, which are evaluated before addition and subtraction.

Operator	Operation	Example
+	Add	V16 + 23
-	Subtract	V16 - 23
*	Multiply	V16 * 1.33
/	Divide	V16 / V12
^	Raise to a power	V12^2

The following examples show how parentheses change the result:

Expression	Result
3+4*6^2	147
((3+4)*6)^2	1764
3+((4*6)^2)	579
(3+4)*(6^2)	252

Using functions and missing values

Function syntax

A function begins with @ and encloses its arguments in square brackets. Commas separate arguments. Square brackets are required even when the function takes no arguments.

```
@FUNCTION[argument1,argument2]
@PI[ ]
```

Each function accepts particular arguments. A value can be a constant, a Variable value, or a supported expression. A Variable selector identifies the Variable itself so a function can read its history, limits, or stored text. Use the forms shown in the reference.

Combine functions

A function can be used inside another function. The inner calculation is evaluated first.

```
@ABS[V18-@AVG[V3..V8,V12]]
```

This calculates the average of V3 through V8 and V12, subtracts that average from V18, then returns the absolute difference.

The following expression calculates an average and returns either that average or 10, whichever is smaller:

```
@MIN[@AVG[V4..V8],10]
```

If all Variables from V4 through V8 are missing, the average returns null. The outer @MIN function ignores that null and returns 10.

Missing values

A null is a missing value; it is not zero. Many aggregate functions skip null values, but their results differ when no usable values remain.

Function	Behaviour with missing values
@AVG, @SUM, @MIN, @MAX	Ignore nulls; return null if all inputs are null.
@COUNT	Counts non-null inputs; returns 0 if all are null.
@RANGE, @SD	Require at least two non-null values.
@MCOUNT	Returns null when no qualifying values are available.

Use @ISNA to test for a missing value and @NA[] to return one. This expression returns 10 when V5 is missing; otherwise it returns V5:

```
@IF[@ISNA[V5],10,V5]
```

Null handling is not identical for every operator. With addition, a non-null value can remain when the other side is null. Use an explicit null test when missing input should change the result.

Practical calculation examples

Limit a calculated result

Return 99 when V5 exceeds 99; otherwise return V5:

```
@IF[V5>99,99,V5]
```

To return a missing value instead of 99 when the limit is exceeded:

```
@IF[V5>99,@NA[],V5]
```

Select a result from a code

```
@CHOOSE[V6,10,20,30]
```

V6=0 returns 10, V6=1 returns 20, and V6=2 returns 30. Selection starts at zero. A null index returns the first option; an invalid index returns null.

Assign a shift from the record time

This example assigns code 10 from 07:30 up to 15:30, code 20 from 15:30 up to 23:30, and code 30 at all other times. It uses minutes since midnight to avoid overlapping decimal-hour ranges.

```
@IF[@HOUR[]*60+@MINUTE[]<450,30,  
@IF[@HOUR[]*60+@MINUTE[]<930,10,  
@IF[@HOUR[]*60+@MINUTE[]<1410,20,30]]]
```

Enter the expression as one formula. The line breaks above are for readability. @MINUTE[] reads the record minute; @MIN[...] finds the smallest value in a list.

Accumulate downtime minutes within a shift

In this example, V3 contains the shift code, V9 contains minutes down, and V11 contains accumulated minutes down. Enter this formula for V11:

```
@IF[V3=@MPREV[V3,5],V9+@MPREV[V11,5],V9]
```

If the current shift code matches the prior non-null shift code found by @MPREV, the formula adds V9 to the prior accumulated total. Otherwise, it starts with V9.

This tests the shift code only; it does not identify a unique dated shift. The two @MPREV calls search independently, so missing values can cause them to retrieve values from different earlier records.

Counters using input and output files

A Calculated Variable can read a saved value from an input file, update it with a formula, and write the result to an output file. Using the same file for input and output carries the saved result into the next record.

Configure the counter

The following example uses V17 as the counter:

Setting	Value
Input filename	SHIFTTOT.V17
Output filename	SHIFTTOT.V17
Formula for a counter increasing by one	V17 + 1

When the Add screen opens, QW reads the saved file value into V17. When the record is saved, the formula calculates the new value and QW writes it to the output file. The next Add operation reads that saved result.

If the input file is not found, QW assumes a starting value of 0. With V17 + 1, the first saved result is therefore 1.

Value read from file	Formula	Saved result
0	V17 + 1	1
1	V17 + 1	2
2	V17 + 1	3

Accumulate a production quantity

To add a quantity from another Variable instead of increasing by one, use:

V17 + V8

Here, V8 contains the quantity to add for the current record. If the saved value is 128 and V8 is 13, the new total is 141. The next record begins with that saved total.

This approach can support a running production count or accumulated lot quantity. The expression determines the increment: use a constant for a fixed step or another Variable for an amount that changes each record.

The filename alone does not define a shift or lot reset. Neither example contains reset logic.

Mathematical functions

In this section, value means a number, a Variable value, or a supported numeric expression.

Function and example	Result or purpose
@ABS[value] @ABS[V5-(V14+V18)]	Absolute value. A null input remains null.
@INT[value] @INT[-1.2]	Rounds down to an integer. The example returns -2, not -1.
@SGN[value] @SGN[V5-12]	Returns -1 for a negative value, 0 for zero, or 1 for a positive value.
@MOD[dividend,divisor] @MOD[V5,4]	Remainder calculated as dividend minus divisor multiplied by the rounded-down quotient. If V5=19, the result is 3. The divisor must not be zero. Both inputs must be non-null.
@SQRT[value] @SQRT[V16]	Square root.
@LOG[value] @LOG[V10/2]	Base-10 logarithm.
@LN[value] @LN[V10/2]	Natural logarithm.
@EXP[value] @EXP[V12*3]	Exponential, e raised to the supplied value.
@PI[]	Returns 3.1415926.

The current calculation implementation evaluates @SQRT, @LOG, @LN, and @EXP only for positive inputs. Zero, negative, or null inputs return null for these functions.

Trigonometry

Function	Purpose
@SIN[V13]	Sine of an angle in radians.
@COS[V13]	Cosine of an angle in radians.
@TAN[V13]	Tangent of an angle in radians.
@ATAN[V13]	Inverse tangent; returns an angle in radians.

Convert degrees to radians with $V13 * @PI[] / 180$. Convert radians to degrees with $V13 * 180 / @PI[]$.

Convert fraction text

Use @FRACTION to convert a measurement entered as fraction text into a numeric value for calculations. This is useful for imperial measurements, such as dimensions recorded in fractional inches.

Calculated Variables - Formulas and Function Reference

The function accepts a fraction, a whole number with a fraction, or a leading negative sign.

`@FRACTION[V8]`

For example, if V8 contains 1 1/2, the result is 1.5. If the measurement is in inches, the result remains in inches; the function converts the notation, not the unit of measure.

Use a Variable reference because this function reads the stored text. Empty or malformed input returns null.

Functions within one record

These functions work with values from the current record. A list can contain individual Variables, consecutive Variable ranges, or numeric values, as appropriate to the function.

Function and example	Result or purpose
@SUM[list] @SUM[V15..V23]	Sum of the non-null values.
@COUNT[list] @COUNT[V12..V18]	Number of non-null values.
@MIN[list] @MIN[V15,V18,V21]	Smallest non-null value.
@MAX[list] @MAX[V15,V18,V21]	Largest non-null value.
@AVG[list] @AVG[V4..V6,V9,V10]	Arithmetic mean of the non-null values.
@RANGE[list] @RANGE[V4..V6,V9]	Maximum minus minimum. Requires at least two non-null values.
@SD[list] @SD[V12,V17..V22]	Sample standard deviation. Requires at least two non-null values and uses count minus one in the denominator.
@POP[list] @POP[V9..V11]	Returns the arithmetic mean of non-null values. When charted, displays the average with a vertical line showing the minimum and maximum values (whisker lines).
@XBAR[list] @XBAR[V9..V11]	Returns the arithmetic mean of the selected Variables for each record. When charted, displays an X-Bar chart with subgroup-based control limits and a Range chart beneath it.

Choose between a record and a moving calculation

@AVG[V9..V11] averages the entered values of three Variables in the current record.

@MAVG[V9..V11,5] averages the available values of those Variables across five records, including the current record.

The final 5 in the moving form is a record count. In a single-record list function, a literal number is a value to include in the calculation. It is not a historical-window setting.

Although @AVG, @POP, and @XBAR calculate the same arithmetic mean, @POP and @XBAR affect how QW charts the result. @POP displays the average with minimum-to-maximum whiskers showing the spread of the included values. @XBAR displays subgroup averages using calculated X-Bar control limits, with a Range chart beneath it.

How X-Bar chart limits are calculated

@XBAR treats the source Variables in each record as a subgroup. The number of source Variables determines the subgroup size.

For example:

```
@XBAR[V5..V7]
```

This calculates the average of V5, V6, and V7 for each record, giving a subgroup size of three.

QW calculates the X-Bar chart limits using the overall average of the subgroup averages, the average subgroup range, and an A2 constant selected from its X/R table.

Chart limit	Calculation
Target	Overall average of the subgroup averages
Upper Control Limit	Target + (A2 × average subgroup range)
Lower Control Limit	Target – (A2 × average subgroup range)
Upper Warning Limit	Halfway between the target and Upper Control Limit
Lower Warning Limit	Halfway between the target and Lower Control Limit

Upper and Lower Specification Limits are not used on the X-Bar control chart, even when fixed specification limits are configured for the Variable.

For a subgroup size of three, A2 is 1.023. The current table contains subgroup sizes through 25; if more than 25 source Variables are included, QW uses the constant for 25.

The X-Bar control chart uses these calculated limits even when fixed limits or a fixed target are configured for the Variable. Changing the limit calculation method does not change the X-Bar control-chart formula.

These limits also govern X-Bar control-chart rule evaluation and data-point colours. Configured fixed limits remain available to other views and calculations that use them, such as histogram views.

Moving and historical functions

Moving summary functions use the current record and earlier records available in QW's record buffer. The final argument specifies the record-window length, and the preceding arguments identify the Variables to include. @MPREV is different: it searches earlier records for a previous non-null value and excludes the current record.

@MAVG[V16,5]

This averages V16 across the current record and four earlier records. When fewer records are available, the window begins at the earliest available record. Null values are skipped.

For functions supporting several Variables, all selected non-null values contribute to one result. @MAVG[V9..V11,5] can therefore average up to 15 values: three Variables across five records.

Function and example	Result or purpose
@MAVG[V9..V11,5]	Moving arithmetic mean.
@MRANGE[V9..V11,5]	Moving maximum minus minimum; requires at least two non-null values.
@MSD[V9..V11,5]	Moving sample standard deviation; requires at least two non-null values.
@MCOUNT[V9..V11,5]	Count of non-null values; returns null if none qualify.
@MMIN[V9..V11,5]	Smallest non-null value in the window.
@MMAX[V9..V11,5]	Largest non-null value in the window.
@MSUM[V9..V11,5]	Sum of non-null values in the window.
@MPREV[V16,5]	Most recent prior non-null value found within the search window.
@MO%OOL[V9..V11,5]	Observed percentage of values outside specification limits over the moving window.

Retrieve a previous value

@MPREV starts with the record immediately before the current record. It searches backward and returns the first qualifying non-null value. If none is found, it returns null.

Unlike the other moving summaries, @MPREV does not include the current record as a candidate value. Use a Variable reference for the value to retrieve, not a literal measurement.

Moving results depend on the records available in memory. Loading a different set of historical records can affect the values available to the calculation.

Conditions and selection

Conditional functions

Function	Purpose
@IF[condition,T,F]	Returns T when condition evaluates to exactly 1; otherwise returns F.
@CHOOSE[index, option0,option1,...]	Selects an option using a zero-based index. A null index selects the first option; an invalid index returns null.
@CASE[value, lower,upper,result,...]	Tests inclusive ranges in order and returns the result for the first matching range. No match returns null.
@ISNA[value]	Returns 1 if the value is null; otherwise returns 0.
@NA[]	Returns a null value.

Comparison operators

Use comparisons in the condition argument of @IF.

Operator	Meaning	Example
=	Equal to	V5=99
<>	Not equal to	V5<>99
<	Less than	V5<99
<=	Less than or equal to	V5<=99
>	Greater than	V5>99
>=	Greater than or equal to	V5>=99

Select from value ranges

```
@CASE[V5,0,9,10,10,19,20,20,29,30]
```

This returns 10 for values from 0 through 9, 20 for values from 10 through 19, and 30 for values from 20 through 29. Values outside those ranges, including a value such as 9.5, do not match and return null.

Each range requires three arguments: lower boundary, upper boundary, and result. Both boundaries are included. If ranges overlap, the first matching range wins.

Date and time functions

Without an argument, date functions use the current record date in V1 and time functions use the current record time in V2. They do not read the computer's current date or time.

Supply a Variable reference to read another date or time Variable. Date/time content must be in the format expected by the function; arbitrary display text is not interchangeable with a date or time value.

Function	Result	Alternate Variable example
@YEAR[]	Four-digit year from the record date, such as 2026.	@YEAR[V3]
@MONTH[]	Month from the record date.	@MONTH[V3]
@DAY[]	Day of the month from the record date.	@DAY[V3]
@DOW[]	Day of week: Sunday=1 through Saturday=7.	@DOW[V3]
@JULIAN[]	Day of year, from 1 through 366.	@JULIAN[V3]
@HOUR[]	Hour from the record time, using a 24-hour clock.	@HOUR[V8]
@MINUTE[]	Minute from the record time.	@MINUTE[V8]
@SECOND[]	Second from the record time.	@SECOND[V8]

For a record time of 14:10:03, @HOUR[] returns 14, @MINUTE[] returns 10, and @SECOND[] returns 3.

@JULIAN[] is a day-of-year number. It restarts at 1 on January 1; it is not a continuous count of days across years.

Time differences

@TIMEDIFF returns seconds. Divide its result by 60 for minutes or by 3600 for hours. Choose the form that matches how the source Variables store their values.

Between consecutive records

```
@TIMEDIFF[ ]
```

Returns the current record timestamp minus the previous record timestamp, in seconds. A previous record must be available.

From the record time to a numeric time

```
@TIMEDIFF[V8]
```

V8 must contain a numeric time in HH.MM form: 9.30 means 09:30 and 14.05 means 14:05. This is not decimal hours. The result is the time from the record's hour and minute to V8, with midnight wrapping. This form uses hours and minutes only; seconds from the record timestamp are ignored, and V8 does not represent seconds.

Between two Variables

```
@TIMEDIFF[V3,V4]
```

The result runs from V3 to V4. If neither Variable has date/time Units, both are interpreted as numeric HH.MM times, with midnight wrapping.

If either Variable has Units containing DATE[or TIME[, QW subtracts the stored numeric date/time values instead. The Variables must contain compatible values; one date Variable and one time Variable are not automatically combined into a timestamp.

Between two date and time pairs

```
@TIMEDIFF[V3,V4,V5,V6]
```

This calculates $((V5 + V6) - (V3 + V4))$ multiplied by 86,400 using the stored numeric values. For separate date and time pairs, those values must already have the correct date/time representation. All four Variables must contain non-null values; otherwise the result is null.

Date and time Units matter

For Text Variables, Units containing DATE[or TIME[cause valid date/time text to be converted to numeric date/time values when QW prepares the calculation data. A display mask alone does not perform that conversion. Parsing depends on the computer's regional date/time interpretation. Invalid or unavailable values return null.

Use actual date/time values when elapsed periods can span several days. A time of day alone does not identify which date it belongs to.

Limits and compliance

Retrieve a fixed limit or target

Function	Value returned
@USL[V16]	Upper Specification Limit
@LSL[V16]	Lower Specification Limit
@UCL[V16]	Upper Control Limit
@LCL[V16]	Lower Control Limit
@TARGET[V16]	Target

These functions return a value only when the corresponding limit or target is fixed. A missing or non-fixed value returns null.

Count and summarize conformance

Function and example	Result or purpose
@OSPEC[V5..V9,V14]	Count of values outside their fixed specification limits.
@OSPEC %[V5..V9,V14]	Percentage of tested Variables outside fixed specification limits.
@OCTRL[V5,V7..V14]	Count of values outside their fixed control limits.
@OCTRL %[V5,V7..V14]	Percentage of tested Variables outside fixed control limits.
@COMPLYSL %[V12..V18]	Percentage meeting applicable fixed specification limits.
@COMPLYCL %[V12..V18]	Percentage meeting applicable fixed control limits.

Out-of-limit checks apply to Numeric, Average, Range, and Calculated Variables. A value is outside a limit when it is greater than the fixed upper limit or less than the fixed lower limit.

Percentage functions exclude Variables without an applicable fixed limit. Compliance numerators count non-null values that are not outside those limits. When no qualifying tested values are available, the percentage functions return 0; this does not establish that measurements passed.

For @COMPLYSL% and @OSPEC%, Variables with applicable fixed specification limits remain in the denominator even when their values are blank. Blank values count as neither compliant nor out of specification, so the two percentages may not add up to 100%.

Summarize rule violations

@RULE[V14..V18] counts selected Variables whose configured rules report a violation. It counts Variables with violations, not the total number of individual rule conditions violated.

@RULE %[V14..V18] returns the percentage of selected Variables with configured rules that report a violation. If none of the selected Variables has a configured rule, the result is 0.

Moving capability functions

These functions take one primary Variable and a record-window length. They use that Variable's applicable limits together with statistics calculated from the moving window.

Function and example	Result or purpose
@MTZ[V16,5]	Moving Target Z metric.
@MCR[V16,5]	Moving Capability Ratio.
@MCPK[V16,5]	Moving Cpk.
@MPPK[V16,5]	Moving Ppk.
@MC%OOL[V16,5]	Calculated out-of-specification percentage based on moving statistics and fixed specification limits.

Cpk and Ppk use different variation estimates

@MCPK estimates standard deviation from the average consecutive absolute difference divided by 1.128. It uses that estimate with the moving average and fixed specification limits.

@MPPK uses the sample standard deviation, moving average, and fixed specification limits. The two functions therefore should not be treated as interchangeable names for the same calculation.

Other capability calculations

@MCR uses sample standard deviation, the mean, and the selected Variable's specification limits.

@MTZ uses the moving mean and sample standard deviation. It uses fixed warning limits where available; otherwise, when the required fixed target and control limits exist, it derives the warning limits halfway between the target and corresponding control limit.

@MC%OOL uses moving statistics and the selected Variable's fixed specification limits, USL and LSL. Fewer than two values produce null. Both @MC%OOL and @MO%OOL use specification limits, not control limits. Older descriptions referring to control limits are incorrect.

Review a formula before using its results

Check the Variable references, parentheses, and square brackets. Confirm that dependencies use lower-numbered Calculated Variables and that date/time inputs use the expected representation.

Check a result with known values, then include missing inputs and relevant boundaries in the review. For historical calculations, confirm that the needed records are available. For limit-based calculations, confirm that the applicable limits are fixed.

For assistance with a calculation, contact support@busitech.com.

Calculated Variables - Formulas and Function Reference