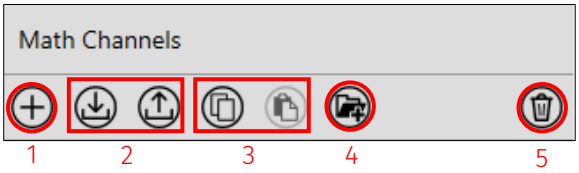
Maths channels overview

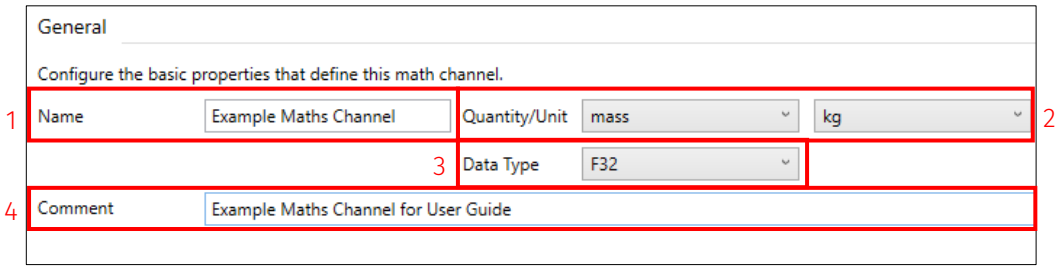
Maths channels allow you to configure mathematical inputs, together with mathematical manipulation of channels using mathematical functions. The following provides a generic overview and some examples of various typical maths channels.

Add a maths channel

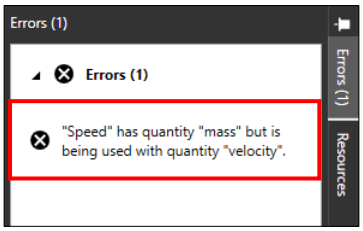
To create a new maths channel, click the **+** button (1). Use the 'import' and 'export' tools to import and export maths channels between existing setups. You can import and export multiple channels as a group (2). Copy and paste maths channels with the copy and paste tools (3.). You can add math channels to groups to aid setup organisation and structuring (4). Use the 'bin' tool to delete maths channels (5).



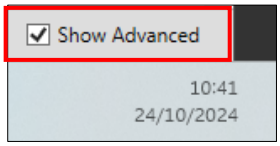
Once a maths channel is added you can configure the name of the channel (1), the units of the channel (2), select the data type (3), and add an optional comment about the channel (4).



Note 1: If maths channels are used for system channels, they may be required to have specific units or a specific data type. See the Errors menu to check for unit and data type errors.

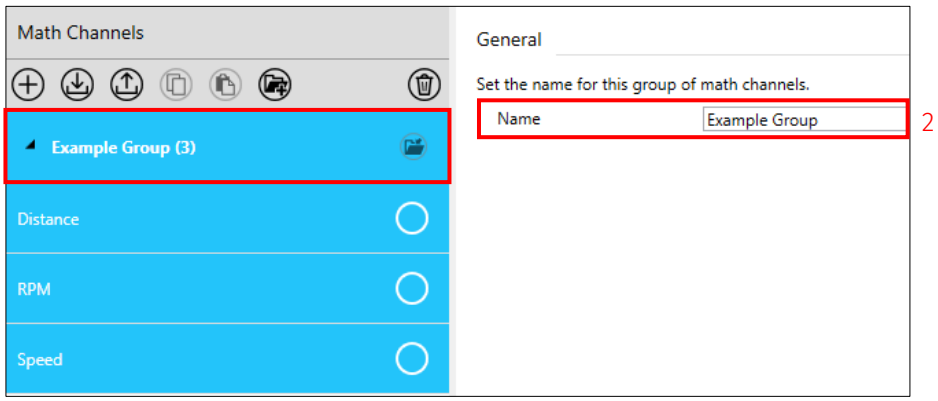


Note 2: If the Data Type menu is not shown, make sure that the Show Advanced check box at the bottom right is selected.

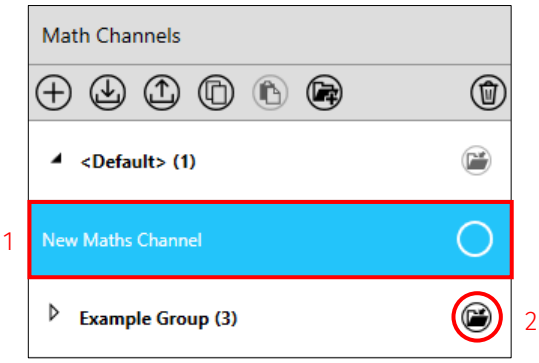


Group maths channels

When a maths channel is created, it is put in the <Default> group. To rename the group, click the group name (1) and enter a new name (2).



Select a maths channels (1) and use the 'Add to group' tool (2) to move that channel.

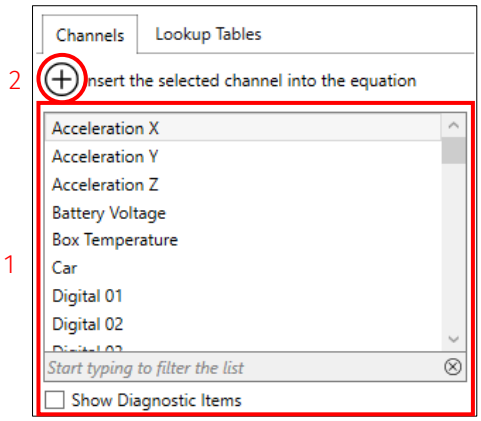


Maths channel equation

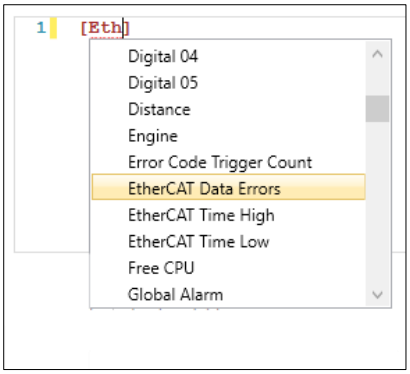
The equation box is where you enter channels equations. All the calculations are performed here. Equations can be as simple as a single value or a channel reference, or as complicated as several choose statements and registers with mathematical functions.

Channels

You can add channels to the equation from the available channels list (1) and search a channel with the standard or magic search functions. Click the **+** tool to add the channel (2) or double-click a channel from the list.

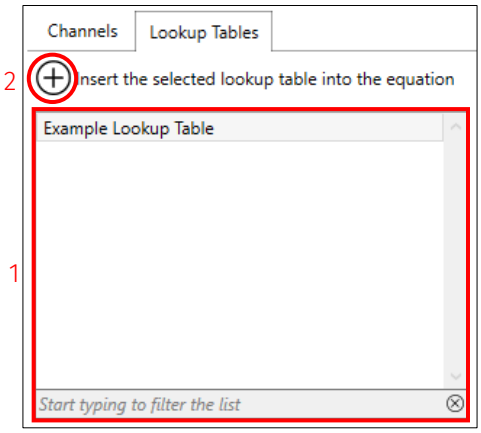


Tip: You can type '[]', and click 'CTRL + Space' and enter an equation within the brackets to add and search for a channel via the channel search menu. Enter a channel name to filter the search.



Lookup tables

Lookup tables (LUTs) can be added to the equation from the available LUTs list (1). You can search for LUTs with the standard or magic search functions. Click the **+** tool to add the LUT (2) or double-click from the LUT list.



A LUT must be written in the following syntax:

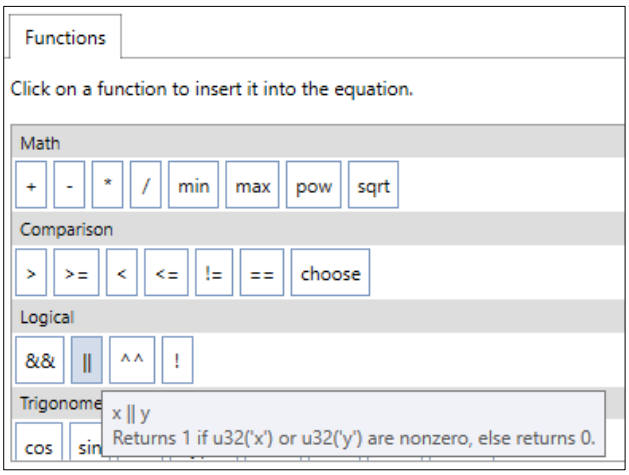
lut('*LUT Name*', '*First dimension reference*', '*Second dimension reference**')

*If using a 2D LUT.

Functions

The **Functions** menu lists all the available mathematical operators to be used in the equation. The list of functions is extensive and includes general mathematical, logical, bitwise operators, and trigonometric operators.

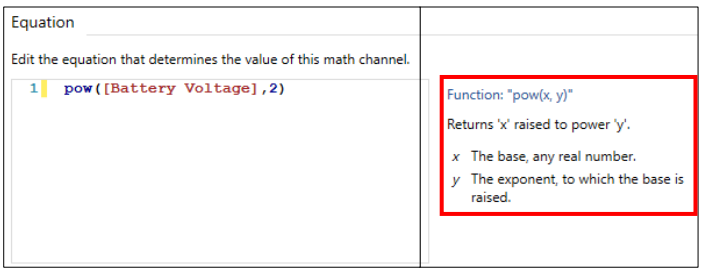
Hover over a function in the menu to display a brief description about the function. Click the function to add it to the equation. You can also type functions into the equation.



Function help

Function help can help you to configure maths channels in the correct format and to help diagnose any syntax errors in the equation. The help displays a small description about the selected function in the equation and provides an example and breakdown of how the function syntax format (1).

Note: The yellow highlight next to the line number indicates a change since the maths channel was last opened.



Rates

The **Rate** section is used to configure the calculation rate of the maths channel. There are two settings:

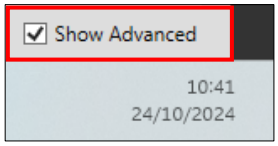
- **Inherit Rate From** – The calculation rate is set by the rate of the selected channel.
- **Fixed Rate At** – You can define the calculation rate from a list of the available rates.

The Consumption Rates section is used to configure the maximum rates at which the input channel(s) are consumed. There are three key options:

- My Rate – Uses the calculation rate of the maths channel set in Rate.
- Auto – Uses the maximum rate of the input channel.
- 1Hz-20khz – You can select the input channel(s) at 1Hz-20kHz.

Note 1: If the Inherit Rate From input channel option is selected for the calculation rate, then the consumption rate for the input channel is automatically set to 'My Rate'.

Note 2: If the Data Type menu is not shown, make sure that the Show Advanced check box at the bottom right is selected.

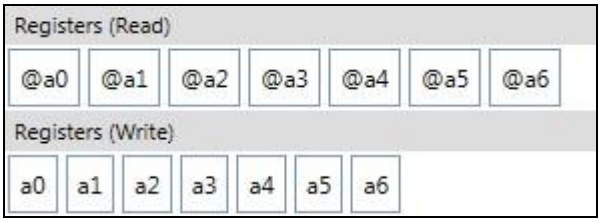


Tip: If for example, you want to log the Maths channel at 20Hz, containing an input channel that is being produced elsewhere on the device at 50Hz, then the calculation rate needs to be set to 100Hz (to satisfy both requirements). Set the calculation rate to be a common (or least common) multiple of all uses of the channel.

Registers

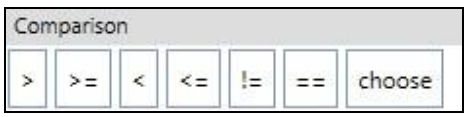
Registers act as short-term memory variables, enabling you to produce channels dependent on historic values.

In the Functions menu there is the option to add in up to seven 'read' and 'write' registers named from a0 to a6. These registers have a default value of zero whenever the device is reset (power cycle, new setup sent, and so on). The registers are evaluated at the calculation rate of the channel. For example, if a channel is set to 100Hz, the registers are updated every 10ms.



Choose statements

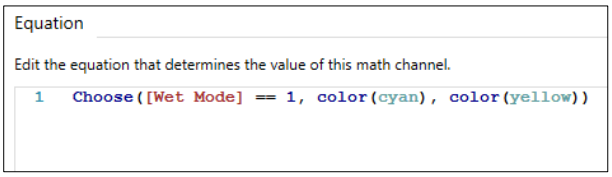
Choose statements are an 'if/else' statement. In other words, 'If' something is true, do X, otherwise do Y. Multiple choose statements can be 'nested' to support more complex comparison functions. The 'Choose' function is in the Comparison section of the Functions menu.



Colour control

You can configure a maths channel can be configured for dynamic colour control of a display item or for shift lights. From the colour function in the equation, a colour can be referenced by its name (for example, cyan).

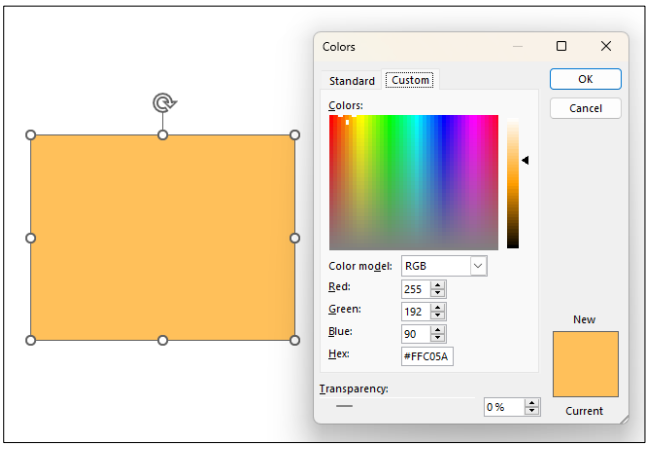
Note: The data type must be U32.



You can also select a colour by its RGB value as a U32 decimal number, where blue is the first byte, green is the second byte, red is the third byte, and brightness is the fourth byte.

For example, pure red at full brightness would equal 4,294,901,760

The following peach colour combining all three R, G and B portions at 50% brightness would equal 268,419,162.



You can also write the colour in bit-wise format where:

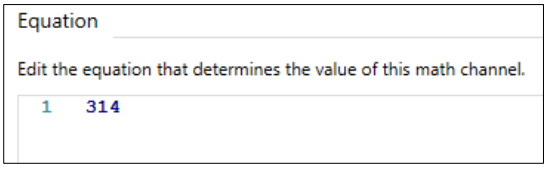
color = (("BRIGHTNESS VALUE"<< 24) | ("RED VALUE" << 16) | ("GREEN VALUE" << 8) | ("BLUE VALUE"))

color = (("127"<< 24) | ("255" << 16) | ("192" << 8) | ("90"))

Example maths channels

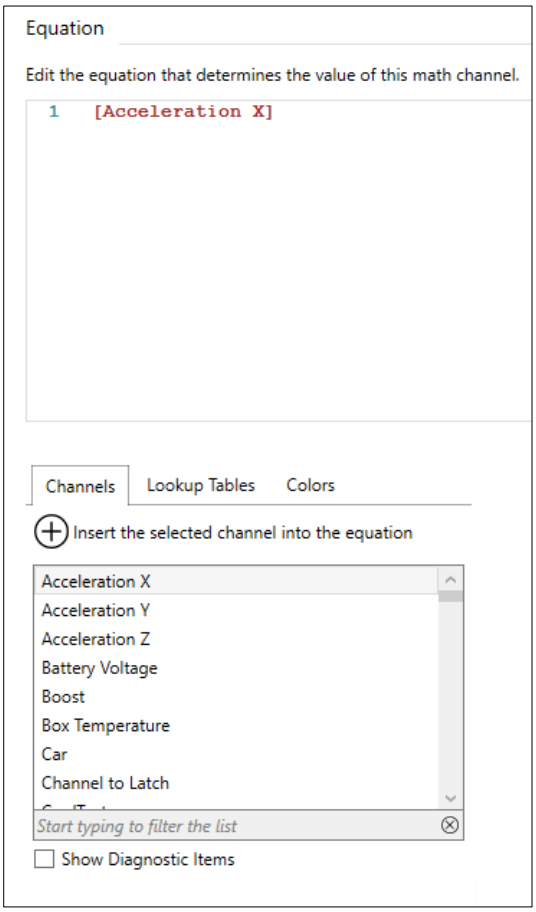
Example 1 – Static value

A maths channel can be a static value. This channel can be referenced and always outputs the configured number. A fixed rate of 1Hz is recommended for this type of channel because there is no computation, thus saving logger CPU processing.



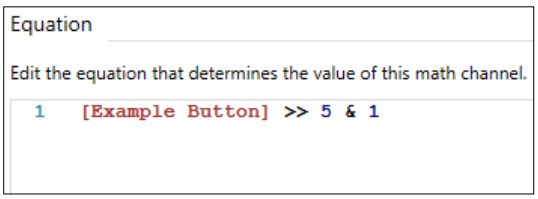
Example 2 – Channel reference

A maths channel can be another channel or event. You can type the channel name or select it from the list below. The syntax for using a channel is as follows: **[Channel Name]**. This type of maths channel can be used to rename native device channels to User defined name. It can also be used to increase the rate of logic channels (Logic Channels are calculated at a maximum of 50Hz).



Example 3 – Bitwise channel

You can use a maths channel to 'Shift' and 'AND' a bit-field channel and return the value. For a button where a 'Click-Latched' option is bit 5 (see Setups – Buttons), it can be 'shifted right' by 5 and 'ANDed' with 1, to return a value of 1 when the 'Click-Latched' button is true.



Example 4 – Choose statement

You can configure a maths channel to complete an 'If' statement decision, like a logic channel. Use the 'Choose' function to test a statement to output a value if true, and then output another if false.

Equation

Edit the equation that determines the value of this math channel.

```
1   Choose([Battery Voltage]<10.5, 1, 0)
```

Example 5 – Registers – simple filter

You can use a maths channel to apply an average filter to a noisy channel by using registers.

Equation

Edit the equation that determines the value of this math channel.

```
1   a4 (@a3) ;
2   a3 (@a2) ;
3   a2 (@a1) ;
4   a1 (@a0) ;
5   a0 ([ExampleChannel]) ;
6   (@a4 + @a3 + @a2 + @a1 + @a0) / 5
```

When this math channel is evaluated, it reads the functions from top to bottom, so register **a4** is evaluated first, then **a3**, and so on. The register name is given first, followed by the equation or value that sets the register value in parentheses. If you need to read the value of a registers value once it has been defined, the register name must be preceded by an **@**. The register name then turns green to show that it has been used. For example, the first line of this channel equation defines register **a4** as the previous value of register **a3**.

The fundamental function within this channel is line 5, **a0 ([ExampleChannel])**, where **[ExampleChannel]** represents the channel that requires filtering. This function sets **a0** as the instantaneous sample of the channel **[ExampleChannel]**. On the next channel sample, **a1** is populated with this first channel value and **a0** is replaced with a new channel value. Each channel value then propagates up the registers from **a0** to a4, where **a0** holds the most recent value and **a4** holds the most historic value (which in this case is four samples before **a0**). If this channel is calculated at 100Hz, then the filter response is like a 20Hz roll off but with a time (phase) delay of approximately 4ms, which, incidentally, may affect dynamic data analysis. Finally, the contents of the registers are summated, and divided by the number of registers used, to produce an average value.

Example 6 – Registers - counter

An alternative approach to a logic channel counter is to create a counter with registers in a maths channel. The following channel counts from 0 to 100 and resets to 0 when it reaches at 100.

Equation

Edit the equation that determines the value of this math channel.

```
1   a0 (choose( @a0 < 100 , @a0 + 1, 0)) ;
2   @a0
```

When this maths channel is first evaluated, a0 is equal to 0. In other words, the 'choose' statement means that, if a0 is less than 100, then increment a0 by 1, otherwise set a0 to 0. If the rate of this channel is set to 1Hz, then the output increments by 1 each second until it reaches 100 and then resets to 0. If the rate of this channel is set to 100Hz, then the output increments by 1 each 10ms second until it reaches 100 and then resets to 0.

Example 7 – Registers – last beacon code seen

This channel captures the last valid code seen by the beacon input. The following equation is used.

Equation

Edit the equation that determines the value of this math channel.

```
1  a3 (F) ;
2  a2 ( (@a2 * (@a3 > 65000) ) + (@a3 * (@a3 < 65000)) ) ;
3  @a2
```

The variable 'F' is the channel Beacon Raw. In this instance. a2 has two possible values in this function. If 'F' is greater than 65000, a2 remains at its original value. If a3 is greater than 65000, a2 is set to output the value of F' which is the new beacon code.

Example 8 – Registers – beacon counter

This is a monotonic count of all the beacons seen since the last logger reset. It can be easily modified to give just the number of end of laps beacons or the number of split beacons. The following equation is used where 'F' is again Beacon Raw.

Equation

Edit the equation that determines the value of this math channel.

```
1  a4( (@a4 + (@a4!=0)) * (@a4 < 100) ) ;
2  a1( @a1 + (@a4==2) ) ;
3  a4( @a4 + ((@a4==0) * (F < 65000)) ) ;
4  @a1
```

This can be thought of as a 'state engine' with 101 states. a4 holds the state (between 0 and 100) and a1 holds the beacon count. This channel creates the following state operation:

Do nothing. If 'F' is a valid code (<65000), then increment a4.
Increment a4. If a4 is greater than 100, set a4 = 0.
Increment a1, increment a4. If a4 is greater than 100, set a4 = 0.
Increment a4, if a4 is greater than 100, set a4 = 0.
Increment a4, if a4 is greater than 100, set a4 = 0 (which in this state is true so we return to state 0, that is a4 is only incremented when a4 is non-zero, or when a4 is zero AND 'F' is a valid code).

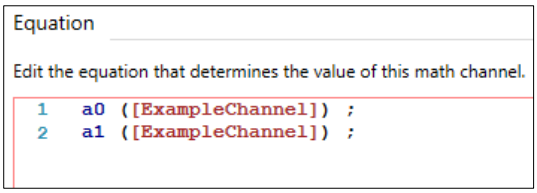
Advice For registers

Synchronization

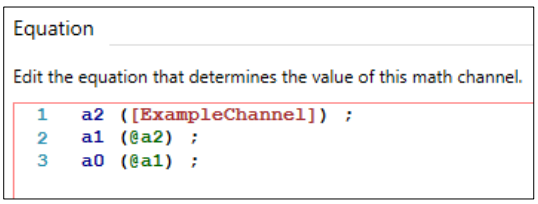
Maths channels are sampled sync executed channels which operate in a standard from top to bottom. Therefore, sampling the same channel at different functions (different lines of the equation) may produce different answers. This is because the functions are calculated as they are read, from top to bottom; they are not all calculated at the same time. As a result, cumulative channels can be influenced by a different start point for each sample.

Input channel sampling multiple time per output channel

Sampling the same channel in more than one function (different lines of the equation) may produce different answers. This is because the functions are calculated as they are read, from top to bottom. In other words, they are not all calculated at the same time. As a result, cumulative channels can be influenced by a different start point for each sample.



Any input that changes its value midway through the calculation of the output channel can cause a problem. Careful consideration of the phase between channels must be taken when the sampling rates of the inputs and outputs are different. For example, a0 and a1 are not necessarily the same, as [ExampleChannel] may have been resampled between the calculation of a0 and a1 if the sampling rates are different. Whereas, the following equation ensures that all the registers hold the same value.



This is because once the value of [ExampleChannel] is assigned to a2, the values of the following registers take their value from a2 rather than the channel itself. As the sampling rate of all the registers are held at the same rate defined by the output channel, this prevents the value of [ExampleChannel] being used throughout an equation from changing.