



Math Channels – Formula Syntax

Use of identifiers

- Channel names, surround the channel name with quotation marks, example: "RPM", "GPS Speed" ...
- Channels, referenced by functions, CHF(Engine RPM). CHF stays for *CHannel Function*. You need to insert the exact AiM function name, you can be eased into this using the "Choose Function" button (center bottom of the window).
- Log Sheets items, surround the item name with the dollar char, example: \$Dew Point\$, \$Humidity\$.

Surrounding chars, like " and \$ allow the presence of spaces in identifier names.

Units

In case you want an identifier to be used always with the same units of measure, just add the units you want aside the identifier name, example: "GPS Speed"[mph] * 2

If you export this channel, and you give it to a friend that uses km/h for displaying speed, well... his GPS Speed channel will be displayed in km/h, but the math channel you're giving him will be always converting his speed into mph for its calculations.

Math Channels – Functions

Following here a list of, at time of editing of this document, all functions available in the RS3 math channel engine, together with a usage prototype.

Logical

All these functions (except IF) output 0 or 1.

AND, OR, NOT, XOR and IF need conditions. Theorically, a condition is 1 (for true) or 0 (for false). In practice, all values different from 0 are interpreted as 1 (true).



"EQ" - "EQ(Equal,to)"

Outputs 1 if the channel value matches a specific value, 0 otherwise.

Example EQ(Gear, 2) would be telling you when/where your vehicle uses the 2nd gear.

This condition can also be written as "Gear == 2".

"GE" - "GE(GreaterOrEqual,to)"

Outputs 1 if the channel value is greater than, or equal to, a specific value, 0 otherwise.

Example "EQ(WaterT[F], 180)" would be telling you when/where your water temperature is greater than, or equal to, 180 F.

This condition can also be written with ">=" operator, i.e. "WaterT[F] >= 180".

"GT" - "GT(Greater,than)"

Outputs 1 if the channel value is greater than, a specific value, 0 otherwise. It can also be written using the ">" operator.

"LE" - "LE(LowerOrEqual,to)"

Outputs 1 if the channel value is lower than, or equal to, a specific value, 0 otherwise. It can also be written using the "<=" operator.

"LT" - "LT(Lower,than)"

Outputs 1 if the channel value is lower than, a specific value, 0 otherwise. It can also be written using the "<" operator.

"NE" - "NE(Different,from)"

Outputs 1 if the channel value is different from a specific value, 0 otherwise. It can also be written using the "!=" operator.

"AND" - "(Condition1) AND (Condition2)"

Outputs 1 if both conditions are different from 0, 0 otherwise.



"OR" - "(Condition1) OR (Condition2)"

Outputs 1 if any condition is different from 0, 0 otherwise.

"XOR" - "(Condition1) XOR (Condition2)"

Exclusive OR, outputs 1 if either one of the two conditions is different from zero and the other condition is equal to zero, 0 otherwise.

"NOT" - "NOT(Condition)"

Inverts a condition, so it outputs 1 if the condition is zero, 0 otherwise.

"IF" - "IF(condition,ifTrue,ifFalse)"

Given a condition, when this is different from zero, the channel equals what passed as second parameter, otherwise the channel equals what passed as third parameter.

Example IF("GPS LonAcc"[g] < 0, -"GPS LonAcc", 0) would give you the absolute value of the inline acceleration when this is lower than 0, otherwise it would output 0.

Mathematical

"POW" - "POW(ChanOrVal,Exponent)"

Power.

"SQRT" - "SQRT(ChanOrValGreaterOrEqualToZero)"

Square root.

"LN" - "LN(ChanOrValGreaterThanZero)"

Logarithm (Euler base).

"LOG" - "LOG(ChanOrValGreaterThanZero)"

Logarithm (base 10).



"ABS" - "ABS(ChanOrVal)"

Absolute value. The result is the input value if positive, the input value made positive when negative.

"EXP" - "EXP(ChanOrVal)"

Exponential.

"CEIL" - "CEIL(ChanOrVal)"

Returns the closest bigger integer. Examples: 3.1 returns as 4.0, 3.8 returns as 4.0, -3.1 returns as -3.0, -3.8 returns as -3.0.

"FLOOR" - "FLOOR(ChanOrVal)"

Returns the closest smaller integer. Examples: 3.1 returns as 3.0, 3.8 returns as 3.0, -3.1 returns as -4.0, -3.8 returns as -4.0.

"DERIV" - "DERIV(ChanOrVal)"

Derivative.

"INTEG" - "INTEG(ChanOrVal)"

Integral.

"LAP_INTEG" - "LAP_INTEG(ChanOrVal)"

Integral, with its value resetting at start/finish line crossing.

"CR_INTEG" - "CR_INTEG(Chan, Condition)"

Conditional Reset Integral, with its value resetting when given condition is different from 0.

"AVG" - "AVG(ChanOrVal,ChanOrVal)"



Outputs the average of the values passed as input. Pay attention to the fact that you can pass more than two parameters. Example AVG("SpeedFL", "SpeedFR", "SpeedRL", "SpeedRR") would compute the average value of all the four corners speeds.

"MAX" - "MAX(ChanOrVal,ChanOrVal)"

Outputs the maximum of the values passed as input. Pay attention to the fact that you can pass more than two parameters. Example MAX("SpeedFL", "SpeedFR", "SpeedRL", "SpeedRR") would compute the maximum value of all the four corners speeds.

"MIN" - "MIN(ChanOrVal,ChanOrVal)"

Outputs the minimum of the values passed as input. Pay attention to the fact that you can pass more than two parameters, as well as AVG and MAX.

Trims

These function are meant to be used to trim (or limit) the value of a input channel to specific threshold values. HIGH_PASS features a lower limit, LOW_PASS features an upper limit, while BAND_PASS features both.

"HIGH_PASS" - "HIGH_PASS(ChanOrVal,LowThreshold)"

"LOW_PASS" - "LOW_PASS(ChanOrVal,HighThreshold)"

"BAND_PASS" - "BAND_PASS(ChanOrVal,LowThreshold,HighThreshold)"

Filters

"FIR" - "FIR(Chan, Number_From_1_To_3)"

Finite Impulse Response (FIR) are a class of filters, basically every output sample is the result of the weighted sum of N samples before. There's a wide literature upon FIR filters, we implemented a rather strong low pass filter, tailoring the needs of our data acquisition. The strength can be decided passing 1 to 3 as a second parameter (1 is mild, 3 is strong).

"ROLL_AVG" - "ROLL_AVG(Chan,Number_From_3_To_10)"



Rolling average (sometimes, in literature, named moving average). Every output sample is the average value of the closest 3 to 10 (second parameter) samples passed as input. It is a mild filter, useful to compute a channel value upon its trend over the considered interval. All samples in the average feature the same weight.

"EMA" - "EMA(Chan, alpha)"

Exponential moving average. The first parameter needs to be a channel, while the second is any number from 0 to 1.

$$\text{Output}(t) = \alpha * \text{Chan}(t) + (1 - \alpha) * \text{Chan}(t-1)$$

Timing

"LAPTIME" - "LAPTIME()"

Outputs, in seconds, the rolling time within the lap.

"SESSIONTIME" - "SESSIONTIME()"

Outputs, in seconds, the rolling time within the session.

"TIME_SHIFT" - "TIME_SHIFT(Channel,ShiftMilliSeconds)"

The output of this function is the channel itself, just shifted in time, shift time being passed as second parameter.

Specific Purpose

"SLIP" - "SLIP(RPM,SPEED)"

The output is basically the ratio between speed and rpm, just normalized to appear as percentage. Theoretically these two values are strictly bound by the drive line, so, highlighting a deviation from linearity shows where slippage occurs.

"BIKE_ANGLE" - "BIKE_ANGLE(SPEED,GYRO)"



The output is an estimation of bike lean angle. "Estimation" means that it's not the true exact value, but that it can be a very good and quick tool for a quick comparison between different setups.

"BIKE_ACCLAT" - "BIKE_ACCLAT(SPEED,GYRO)"

This channel has been kept for compatibility purpose with RS2A, having been computed as "GPS LatAcc" within GPS channels.

"BIKE_CORNRAD" - "BIKE_CORNRAD(SPEED,GYRO)"

This channel has been kept as is for compatibility purpose with RS2A, having been inserted by default within GPS channels as "GPS Radius".

"GEAR" - "GEAR(RPM,SPEED,minGearN,maxGearN)"

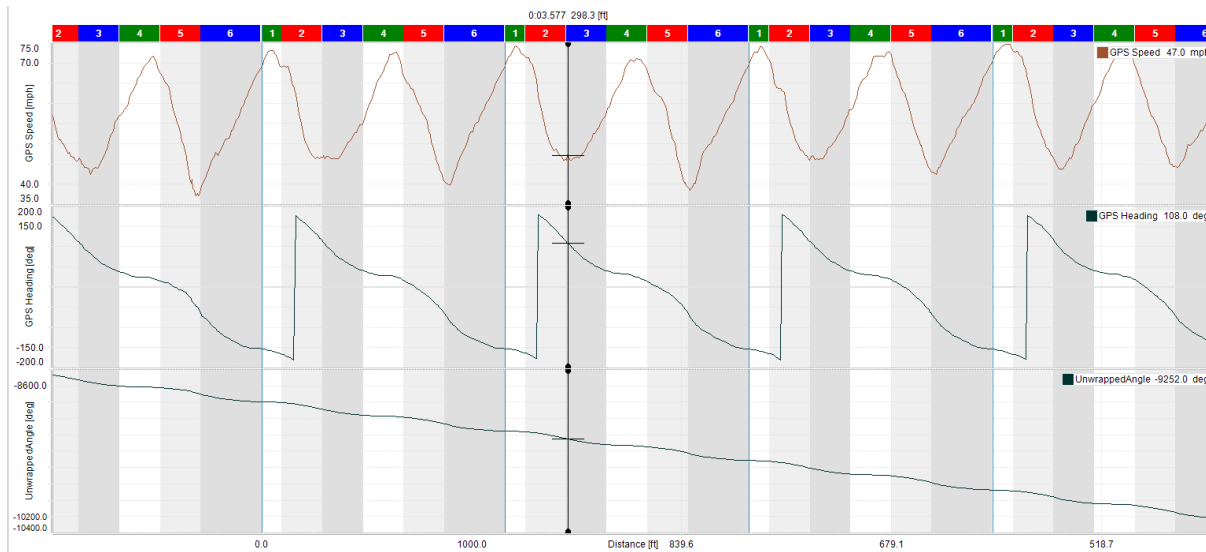
Useful when no gearbox sensor is available on the vehicle. The output of this channel is the gear used, given as input the values of rpm, speed and first/last gear used.

Normally first gear is 1, but there are cases in which the 1st gear is used just for a stand start and never inserted any more... in these cases passing 2 as first gear can help the algorithm to better perform in identifying used gear.

Same for last gear... if your gearbox features 7 gears but the track configuration makes 6th and 7th gears useless, you need to input 5 as last gear.

"UWA" - "UWA(ANGLE)"

Unwrapped angle. An example is worth a thousand words. Three channels taken from an oval session: speed, heading angle (decreases as the track is counterclockwise, jumping from -180 to 180 while crossing the southwise direction), unwrapped angle (keeps decreasing lap after lap).



Advanced Logics

"COND_VALID_AT_LEAST" - "COND_VALID_AT_LEAST(ConditionChannel, TrueOrFalse, AtLeast, InLen)"

Outputs 1 if the given condition is true (you can also choose false) for at least a given number of samples in a given samples number, say 2 samples in 5 total samples.

"COND_VALID_W_HISTER" - "COND_VALID_W_HISTER(ConditionChannel, TrueOrFalse, ForNsamples, ChangeHisteresis)"

Outputs 1 if the given condition is true (you can also choose false) for at least a given number of samples, the change of output being applied after a hysteresis number of samples.

Trigonometry

The purpose of the following lines is not to explain trigonometry, but to detail that you can pass as argument a numeric value or a channel.

SIN, COS, TAN need radians as input, so you need to take care of converting angles from degrees to radians.

ASIN, ACOS, ATAN, ATAN2 output radians, so you need to pay attention to angle conversion.

Please notice that the difference between ATAN and ATAN2 is that ATAN resolves angles between -90 deg to 90 deg, while ATAN2 resolver from -180 to 180.



"**COS**" - "COS(ChanOrValRadians)"

"**ACOS**" - "ACOS(ChanOrValBetweenMinusOneAndPlusOne)"

"**SIN**" - "SIN(ChanOrValRadians)"

"**ASIN**" - "ASIN(ChanOrValBetweenMinusOneAndPlusOne)"

"**TAN**" - "TAN(ChanOrValRadians)"

"**ATAN**" - "ATAN(ChanOrValBetweenMinusOneAndPlusOne)"

"**ATAN2**" - "ATAN2(ChanOrVal_X,ChanOrVal_Y)"

Custom Purpose

"**TCR_2022**" - "TCR_2022(DriverCond, EngRPM, OverBoost)"