AD-DA.ca, Math Tools

Plugin Description

Math expression evaluator plugin for Q-SYS.

The plugin lets you wire numeric values into a configurable number of input pins, compute any Lua-style mathematical expression that references those inputs through `{1}`, `{2}` … `{N}` placeholders, and expose the result as a formatted text string.

The number of inputs is set through the `Input Count` property; the plugin dynamically creates that many `Input` text indicator pins. Any numeric source can be routed into these pins as a string and will be parsed transparently (dot or comma decimal).

Formulas are written in a familiar infix syntax (`+ - * / % ^`, parentheses, comparison, `and / or / not`) and have access to all of Lua's `math.*` library plus a curated set of helpers (`round`, `floor`, `ceil`, `abs`, `sqrt`, `pow`, `min`, `max`, `clamp`, `sum`, `avg`, `mean`, `median`, `mod`, `log`, `exp`, the trig family, `rad/deg/pi`, and an `if_(cond, a, b)` ternary). Placeholder `{N}` (uppercase) alone is replaced by the input count, allowing dynamic averaging like `({1}+{2}+{3}) / {N}`.

An optional `Unit` string is appended to the formatted result (`25.0 dB`, `90°`, `42%`). A `Decimals` knob controls display precision. An `Auto` toggle re-evaluates on every input change with built-in debouncing, otherwise the `Evaluate` trigger button runs the formula on demand.

The plugin also supports an arbitrary number of **Conditional Results**: each one is a `Result <op> Value → LED` row where the operator is picked from a combo (`>`, `>=`, `<`, `<=`, `==`, `!=`) and the compared value can be a hard-coded number or any expression with `{N}` placeholders. The matching LED turns on when the condition is true.

Key Features

- **Dynamic Input Count** — Property-driven number of numeric input pins (1 to 32)
- **Expression Engine** — Full Lua infix syntax with standard operator precedence
- **Rich Function Library** — `round, floor, ceil, abs, sqrt, pow, min, max, clamp, sum, avg, mean, median, mod, log, exp, sin/cos/tan, asin/acos/atan, rad, deg, pi, if_(cond,a,b)`, plus all of `math.*`
- **Placeholders** — `{1}..{N}` for input values, `{N}` alone for the input count; also exposed as `x1..xN`, `i1..iN`, `v1..vN`, and `N`
- **Auto / Manual Evaluation** — `AutoEvaluate` toggle (debounced 50 ms) or one-shot `Evaluate` trigger

- **Optional Unit** — String appended to the formatted result with smart spacing (`25 dB`, `90°`, `42%`)
- **Decimals Control** — Display precision 0–10 digits
- **Rounding Mode** — Choose between `Round` (nearest) and `Truncate` (toward zero) when applying the decimals precision
- **Numeric Pin Output** — Separate `ResultValue` float pin for downstream chaining
- **Conditional Results** — N configurable `Result <op> Value → LED` rows; value can be a literal or any expression
- **Status-driven Errors** — Parse / eval / NaN / Infinity reported in the `Status` indicator
- **Help Page** — Built-in formula reference inside the plugin UI
- **Comma / Dot Decimal** — Locale-tolerant numeric parsing
- **Debug Levels** — Adjustable Eval / Function / Verbose logging

---

Plugin version

This document is written for the plugin:

- AD-DA.ca, Math Tools v1.0.0

Release note

2026-06 : V1.0.0.0

- Initial release

QDS compatibility

This plugin has been developed and tested on:

- Q-SYS Designer version `10.2.1`

Properties

License

Property name: `License`
Type: `string`
Description: Primary license used for runtime validation on hardware Cores. Not required in Emulation mode.

License Backup

Property name: `License Backup`
Type: `string`
Description: Secondary license used if the primary license check fails (typically the redundant Core license).

Input Count

Property name: `Input Count`
Type: `integer`
Min: `1`
Max: `32`
Default: 2
Description: Number of numeric input pins to create. Each input becomes an `Input` indicator pin available as `{N}` in the formula.

Conditional Result Count

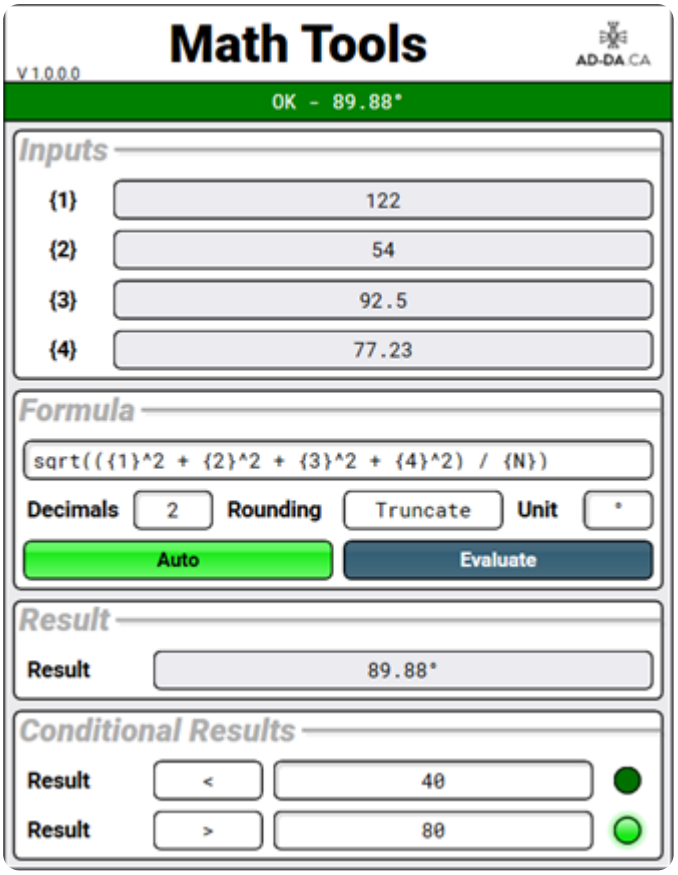
Property name: `Conditional Result Count`
Type: `integer`
Min: `0`
Max: `32`
Default: `2`
Description: Number of conditional result rows (operator + value + LED). Set to `0` to hide the Conditional Results section entirely.

Debug Print

Property name: `Debug Print`
Type: `enum`
Choices: `None` / `Eval` / `Function Calls` / `All` / `Debug`
Description: Controls plugin debug output level. `Eval` traces every substituted expression and its result, `Function Calls` traces internal function entry, `Debug` enables everything.

Controls

Control Page



STATUS

Controls name: `Status`
Type: `Indicator-Status`
Description: Reports plugin health and the latest evaluation outcome.

> OK — `<result>`

> FAULT — Formula failed to parse (`Parse: …`) or failed to evaluate (`Eval: …`)

> COMPROMISED — Result is `NaN` or `Infinity`

INPUTS

The plugin creates one `Input` pin per slot, where `N` ranges from `1` up to `Input Count`.

INPUT N

Controls name: `Input` (indexed when `Input Count > 1`)
Type: `Indicator-Text` (Input pin)
Description: Numeric input value referenced in the formula as `{N}`. Accepts dot or comma

decimal separator. Empty or non-numeric content is treated as `0`. Wire any text or numeric source into this pin — the value will be re-parsed on every change.

FORMULA

EXPRESSION

Controls name: `Formula`
Type: `Text-Text Edit`
Description: The mathematical expression to evaluate. Reference inputs with `{1}`, `{2}` … `{N}` placeholders (or the aliases `x1..xN`, `i1..iN`, `v1..vN`). The literal `{N}` (uppercase) alone is substituted by the input count. Supports all Lua arithmetic, comparison and logical operators plus the helpers listed in the Formula Reference.

DECIMALS

Controls name: `Decimals`
Type: `Knob-Integer`
Range: `0` to `10`
Description: Number of decimal digits shown in the formatted `Result` string. Does not affect the `ResultValue` numeric pin (which always carries the rounded value at the same precision).

Rounding

Controls name: `Rounding`
Type: `Text-ComboBox`
Choices: `Round` / `Truncate`
Description: How the numeric result is reduced to the chosen `Decimals` precision.
`Round` rounds to the nearest value (`17.5` with 0 decimals → `18`).
`Truncate` drops the extra digits toward zero (`17.5` → `17`, `-17.5` → `-17`).

UNIT

Controls name: `Unit`
Type: `Text-Text Edit`
Description: Optional unit of measurement appended to the formatted `Result` string. A space separator is inserted automatically, except when the unit starts with `%`, `°`, `'` or `"` (so `25%`, `90°`, `6'` stay tight). Leave empty to disable.

AUTO

Controls name: `AutoEvaluate`
Type: `Button-Toggle`
Description: When ON (default), the formula is re-evaluated automatically on every input, formula,

decimals, unit or conditional change (debounced at 50 ms). When OFF, evaluation only runs when the `Evaluate` trigger is pressed.

EVALUATE

Controls name: `Evaluate`
Type: `Button-Trigger`
Description: Forces an immediate re-evaluation of the formula, regardless of the `AutoEvaluate` state.

RESULT

RESULT

Controls name: `Result`
Type: `Indicator-Text`
Description: Final formatted result string — rounded to `Decimals` digits and suffixed with `Unit` when set. For non-numeric expression results (rare, e.g. a string returned from a custom expression) the raw value is shown as-is.

CONDITIONAL RESULTS

The plugin creates one block of the three controls below per conditional, where `N` ranges from `1` up to `Conditional Result Count`. Section is hidden when the property is `0`.

OPERATOR

Controls name: `ConditionalOperator` (indexed)
Type: `Text-ComboBox`
Choices: `>` / `>=` / `<` / `<=` / `==` / `!=`
Description: Comparison operator applied between the numeric result and the `Value` field. `!=` is an alias for Lua's `~=`.

VALUE

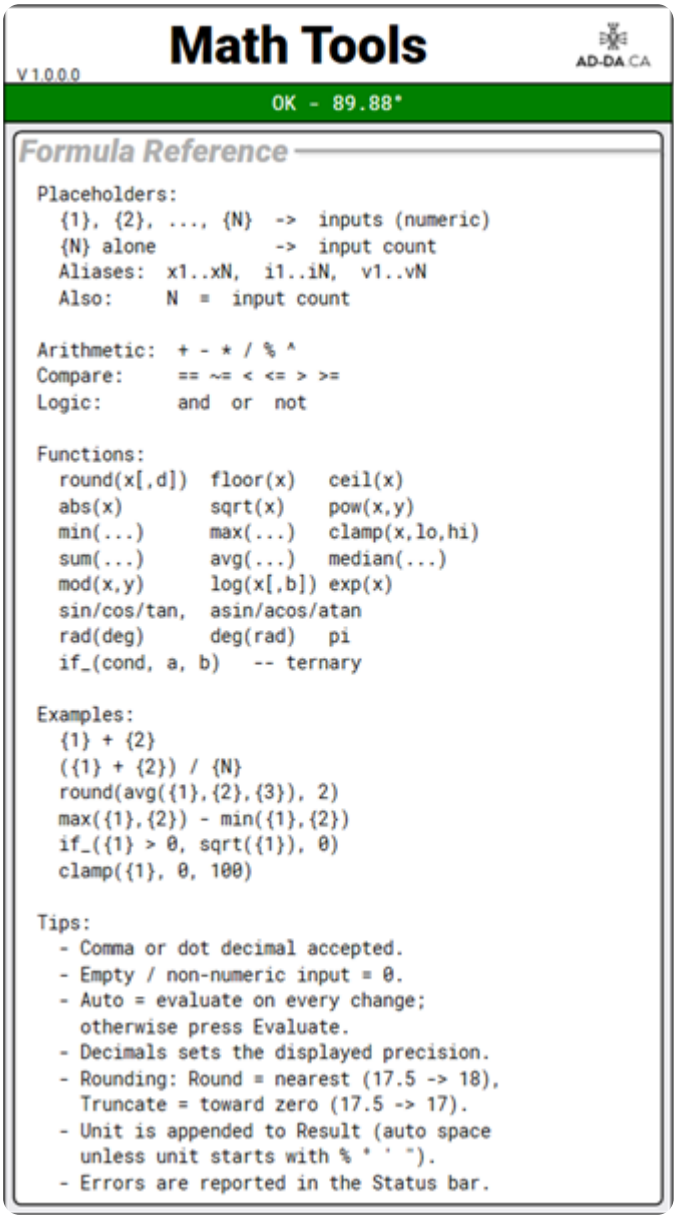
Controls name: `ConditionalValue` (indexed)
Type: `Text-Text Edit`
Description: Right-hand side of the comparison. Can be a literal number (`3`, `1.5`, `-12`) or any expression with placeholders (`{1}`, `avg({1},{2})`, `pi`, `{N}*2`, …). Evaluated with the same engine and the current input values.

LED

Controls name: `ConditionalResult` (indexed)
Type: `Indicator-LED` (Output pin)
Description: Boolean output pin — ON when the comparison `Result <Operator> Value` is true, OFF otherwise. Forced OFF on empty formula, parse error, NaN, Infinity, or non-numeric result.

Help Page

The Help page renders an in-plugin formula reference: placeholders, operator precedence, full function list, copy-paste examples, and usage tips. See Formula Reference below for the same content.



Appendix

Formula Reference

Placeholders

- `{1}`, `{2}` … `{N}` — input pin values (numeric)
- `{N}` alone (uppercase) — input count
- Aliases: `x1..xN`, `i1..iN`, `v1..vN`, and bare `N` / `n`

Operators (standard Lua precedence, highest to lowest)

#	Operators	Notes
1	`^`	Right-associative: `2^3^2` = 512
2	unary `-`, `not`	`-2^2` = -4 (unary minus binds *after* `^`)
3	`*` `/` `//` `%`	`//` = floor division
4	`+` `-`	
5	`<` `<=` `>` `>=` `==` `~=`	
6	`and`	Short-circuit
7	`or`	Short-circuit

Every `{N}` substitution is wrapped in parentheses before parsing, so input values never bleed into surrounding operators.

Functions

- Rounding / sign — `round(x[,d])`, `floor(x)`, `ceil(x)`, `abs(x)`
- Power / log — `sqrt(x)`, `pow(x,y)`, `log(x[,b])`, `exp(x)`
- Min / max / clamp — `min(...)`, `max(...)`, `clamp(x, lo, hi)`
- Aggregates — `sum(...)`, `avg(...)`, `mean(...)`, `median(...)`
- Modulo — `mod(x, y)` (alias of `math.fmod`)
- Trigonometry — `sin`, `cos`, `tan`, `asin`, `acos`, `atan`, `rad(deg)`, `deg(rad)`, `pi`
- Ternary — `if_(cond, a, b)` — returns `a` if `cond` is truthy, else `b`
- All of Lua's `math.*` library is also available unprefixed

Examples

Goal	Formula
Sum	`{1} + {2}`
Average of all inputs (any count)	`({1} + {2} + {3}) / {N}`
Rounded average	`round(avg({1},{2},{3}), 2)`

Spread (max − min)	`max({1},{2}) - min({1},{2})`
Safe square root	`if_({1} > 0, sqrt({1}), 0)`
Clamp to 0..100	`clamp({1}, 0, 100)`
Percent	`({1} * {2}) / 100`
Boolean to 1/0	`if_({1} > 0, 1, 0)`
Degrees → radians sine	`sin(rad({1}))`

Tips

- Inputs accept dot or comma decimal separator.
- Empty or non-numeric input values are treated as `0`.
- If the formula returns a boolean, it is converted to `1` / `0`.
- Errors are reported in the `Status` controls.