LV-muParser Library
User Guide

Rev. 1.1

Contents

1 Introduction

Many applications require the parsing and efficient evaluation of mathematical expressions. Although LabVIEW provides a built-in formula parsing library, it is not the most efficient and lacks some commonly used functions such as logical and bitwise operations.

The LV-muParser library provides an convenient interface for using the muParser Fast Math Expression Parser (https://beltoforion.de/en/muparser/) in LabVIEW.

A pre-built, modified version of muParser v2.3.5 shared library is installed along side this library.

2 System Requirements

Software:

- National Instruments LabVIEW 2018 (or newer)
- JKI VI Package Manager 2020 (or newer)

3 Installation

Install the latest version of the *muParser Expression Parser API* in VI Package manager from the VIPM Community Repository.

Note that the LAVA Palette package will also be installed if it is not already.

3.1 Shared Library

Four pre-compiled copies of the libmuparser-lv shared library are installed alongside the labview code. They are located in "<LabVIEW>\vi.lib\LAVA\muParser".

1) **libmuparser-x32-lv.dll**: For Windows target running LabVIEW 32-bit.
2) **libmuparser-x64-lv.dll**: For Windows target running LabVIEW 64-bit.
3) **libmuparser-x64-lv.so**: For Linux target running LabVIEW 64-bit.
4) **libmuparser-x64rt-lv.so**: For cRIO RT Linux 64-bit target.

For Windows and Linux targets (not cRIO), the VIPM installer will attempt to copy the appropriate version of the shared library to the filename "libmuparser-lv.dll" in the same directory. This is this file that the LabVIEW code will link to.

It is possible that the copy operation will fail on **Linux targets** if VIPM is not running under the root user. You can manually copy the file "libmuparser-x64-lv.so" to "libmuparser-lv.dll" if the operation had failed during install.

For **cRIO RT Linux targets**, you need to manually copy the “libmuparser-x64rt-lv.so” to the cRIO under “/usr/local/lib/libmuparser-lv.so”. Then run ldconfig on the cRIO. This will create the symbolic link to the library under the name “libmuparser-lv.so.2” which is what the LabVIEW code will link to.

4 Examples

Examples are included in “<LabVIEW>\examples\LAVA\muParser”

- **mupExpr Calc.vi**: A simple expression calculator
- **mupExpr Plot.vi**: A simple single variable plotter
- **mupExpr sglExpr sglVar with constants example.vi:** Demonstrate evaluation of an expression with a single variable
- **mupExpr sglExpr multiVar bulkMode.vi**: Demonstrate bulk mode evaluation of a single expression
- **mupExpr multiExpr multiVar example.vi**: Demonstrate evaluation of multiple expressions in one eval call.

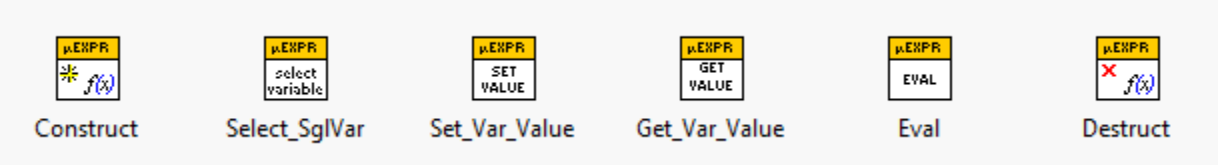
5 Support

If you have any problems with this library or want to suggest changes contact Porter via PM on lavag.org or post your comment on the support forum: https://lavag.org/topic/20262-cr-lv-muparser

The development source code is available on GitHub: https://github.com/rfporter/LV-muParser

6 muParser Palette

The muParser API is located in the LabVIEW functions palette under “Addons -> LAVA -> muParser”.



With these functions, a muExpr Instance can be created, variables and constants can be set and retrieved, and the expression can be evaluated.

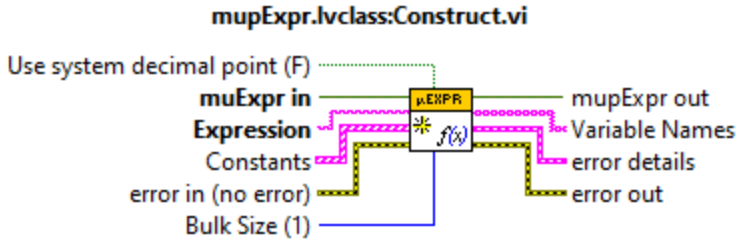
7 muExpr Instance

When a new expression is defined, an new muExpr Instance is created. This instance contains a copy of the expression as well as all parameters required to efficiently perform the expression evaluation.

The muExpr Instance **must** be destroyed when the expression is no longer required in order to release system resources. If the instance wire had been split (copied), destruct is only needed on one of the copies.

7.1 Create muExpr Instance

To create a new muExpr instance, use “Construct.vi” from the muParser Palette.



Inputs:

- **muExpr in**: muExpr instance input. An existing instance will be closed automatically. Normally just wire a constant to this terminal.
- **Expression**: Expression or set of expressions (separated by argument separator). See section Expression Syntax
- **Constants**: Optionally specify a list of constant name-value pairs.
- **Bulk Size**: Set the range for bulk mode evaluations (max size).
- **Use system decimal point**: Set to true for adapting to the system’s decimal point. If the system's decimal point is a comma, the argument separator will be changed to a semi-colon. The default decimal point is a period. The default argument separator is a comma.

Outputs:

- **muExpr out**: Instance wire to use for subsequent operations.
- **Variable Names**: List of variables identified in the expression.
- **Error details** Additional error details including position and token in the expression that caused the error

7.2 Instance Properties

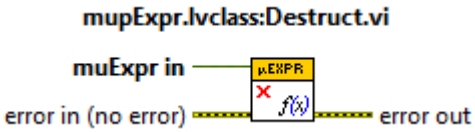
Placing the property node on a muExpr Instance wire will reveal the following properties:

- **Bulk Size**: The number of evaluations to perform when calling bulk mode evaluation.
- **Constants**: 1D array of constant name-value pairs.
- **Expression**: Expression or set of expressions
- **isValid**: Boolean value is true if the expression is valid.
- **SglVar Name**: Name of the variable selected for single variable evaluation.
- **Var Names**: 1D array of all variable names.
- **Var Values**: 1D array of all variable values. Same order as the Var Names array.

All of these properties are read-only.

7.3 Destroying muExpr Instance

To close a muExpr Instance, use “Destruct.vi” from the muParser Palette.



This will release the internal muParser handle and variable pointers.

8 Expression Syntax

8.1 Variable and Constant Naming

Max length: 100 characters

Character set:

0123456789_:abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ

8.2 Data Type

Base data type is set to double precision floating point (64-bit IEEE 754)

Bitwise operations will convert the input values to 32-bit unsigned integer to perform the operation. The return value is in double precision floating point.

Values specified in hexadecimal (starting with 0x) or binary (starting with 0b) are interpreted as 32-bit unsigned integers.

8.3 Built-in Constants

The following table of constants are pre-defined and therefore do not need to be explicitly defined when creating the muExpr instance.

Constant	Value	Description
_pi	3.14159265358979323846264 3	Pi constant
_e	2.71828182845904523536028 7	Euler's constant

8.4 Operators

Operator	Description	Priority	Associativity
=	assignment	0	Right-to-Left
?:	ternary conditional	0	Right-to-Left
||	logical or	1	Left-to-Right
&&	logical and	2	Left-to-Right
|	bitwise or	3	Left-to-Right
|^	bitwise xor	3	Left-to-Right
&	bitwise and	4	Left-to-Right
<=	less or equal	5	Left-to-Right
>=	greater or equal	5	Left-to-Right
!=	not equal	5	Left-to-Right
==	equal	5	Left-to-Right
>	greater than	5	Left-to-Right
<	less than	5	Left-to-Right
>>	bitwise shift right	5	Left-to-Right
<<	bitwise shift left	5	Left-to-Right
+	addition	6	Left-to-Right
-	subtraction	6	Left-to-Right
*	multiplication	7	Left-to-Right
/	division	7	Left-to-Right
%	modulus	7	Left-to-Right
!	logical not	7	Right-to-Left
~	bitwise not	7	Right-to-Left
+	unary plus	7	Right-to-Left
-	unary minus	7	Right-to-Left
^	raise x to the power of y	8	Right-to-Left

8.5 Built-in Functions

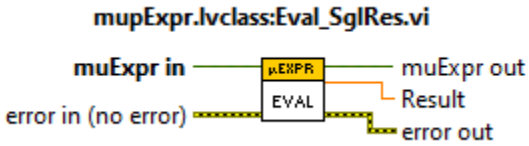
Name	Argc.	Explanation
sin	1	sine function
cos	1	cosine function
tan	1	tangent function
asin	1	arcsine function
acos	1	arccosine function
atan	1	arctangent function
sinh	1	function
cosh	1	hyperbolic cosine
tanh	1	hyperbolic tangent function
asinh	1	hyperbolic arcsine function
acosh	1	hyperbolic arccosine function
atanh	1	hyperbolic arctangent function
log2	1	logarithm to the base 2
log10	1	logarithm to the base 10
log	1	logarithm to base e (2.71828...)
ln	1	logarithm to base e (2.71828...)
exp	1	e raised to the power of x
sqrt	1	square root of a value
sign	1	sign function -1 if x<0; 1 if x>0; 0 if x=0
rint	1	round to nearest integer
abs	1	absolute value
min	var.	min of all arguments
max	var.	max of all arguments
sum	var.	sum of all arguments
avg	var.	mean value of all arguments

9 Evaluating Expressions

Expressions can be evaluated in a few different ways depending on the expression type and evaluation mode.

The "**Eval.vi**" function from the muParser Palette is a polymorphic VI with the following instances:

- **SglRes**: Evaluate the expression, producing a single result.

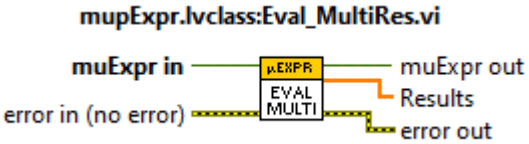


  Note that the variable values must be explicitly set prior to calling this function.

  If multiple expressions have been input, the result of the last one in the list is returned.

  If the variables are defined for bulk mode evaluation, only the first set (index 0) will be evaluated.

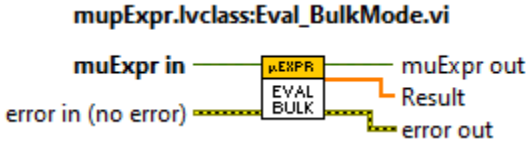
- **MultiRes**: Evaluates all expressions and returns one result per expression.



  Note that the variable values must be explicitly set prior to calling this function.

  If the variables are defined for bulk mode evaluation, only the first set (index 0) will be evaluated.

- **BulkMode**: Evaluates the expression in bulk mode. Returns the 1D array of results.

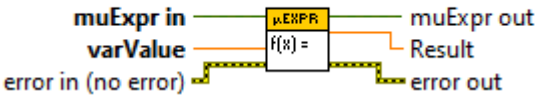


  Note that the variable values must be explicitly set prior to calling this function.

  If multiple expressions have been input, the result of the last one in the list is returned.

- **SglVar_SglRes**: Updates the value of the selected single variable then evaluates the expression. Returns a single result.

mupExpr.lvclass:Eval_SglVar_SglRes.vi



If the expression contains no variables, the input value is ignored.

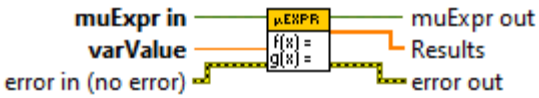
If the expressions contain more than one variable only the "Single Variable" is updated.

If multiple expressions have been input, the result of the last one in the list is returned.

If the variables are defined for bulk mode evaluation, only the first set (index 0) will be evaluated.

- **SglVar_MultiRes**: Updates the value of the selected single variable then evaluates all expressions. Returns one result per expression.

mupExpr.lvclass:Eval_SglVar_MultiRes.vi



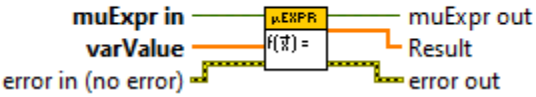
If the expressions contain no variables, the input value is ignored.

If the expressions contain more than one variable only the "Single Variable" is updated.

If the variables are defined for bulk mode evaluation, only the first set (index 0) will be updated and evaluated.

- **SglVar_BulkMode**: Updates the values for the selected single variable then evaluates the expression in bulk mode. Returns the array of results.

mupExpr.lvclass:Eval_SglVar_BulkMode.vi

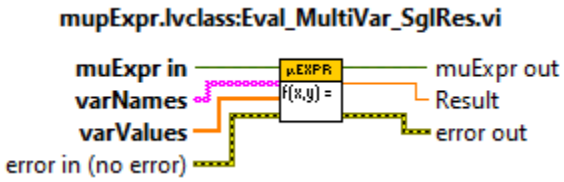


If the expression contains no variables, the input values are ignored.

If the expression contains more than one variable only the "Single Variable" is updated.

If multiple expressions have been input, the result of the last one in the list is returned.

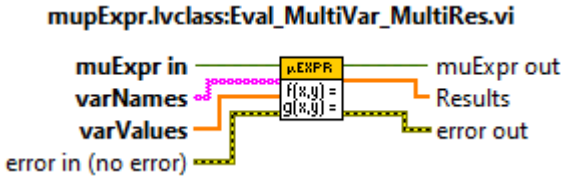
- **MultiVar_SglRes**: Updates the value of multiple variables then evaluates the expression. Returns a single result.



If multiple expressions have been input, the result of the last one in the list is returned.

If the variables are defined for bulk mode evaluation, only the first set (index 0) will be evaluated.

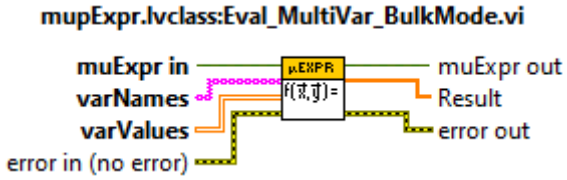
- **MultiVar_MultiRes**: Updates the value of multiple variables then evaluates the expressions. Returns one result per expression.



If any input variable is not found in the expression, a warning is generated.

If the variables are defined for bulk mode evaluation, only the first set (index 0) will be updated and evaluated.

- **MultiVar_BulkMode**: Updates the value of multiple variables then evaluates the expression in bulk mode. Returns the 1D array of results.



Note that the array of values provided for variable values should be of size:

  - Num Rows: Number of Variables
  - Num Cols: Bulk Size

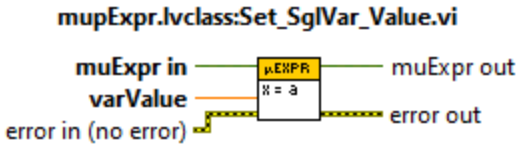
If any input variable is not found in the expression, a warning is generated.

If multiple expressions have been input, the result of the last one in the list is returned.

9.1 Set Variable Value

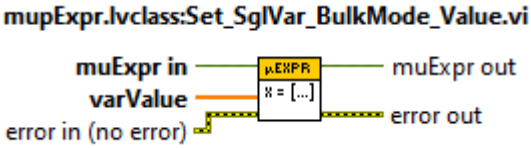
Variable values can be explicitly specified using the “Set_Var_Value.vi” from the muParser Palette. This VI is polymorphic with the following instances:

- **x = a**: Set the value of the expression's selected single variable.



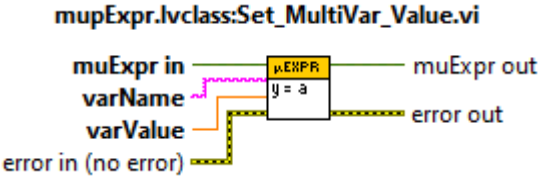
If the expression contains no variables, the input value is ignored.

- **x = […]**: Set the bulk mode values of the expression's selected single variable.



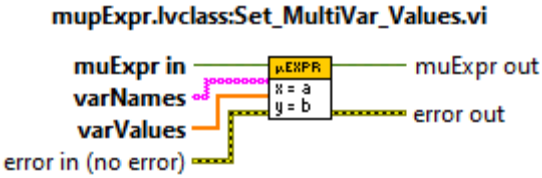
If the expression contains no variables, the input value is ignored.

- **y = a**: Set the value of a specified variable.



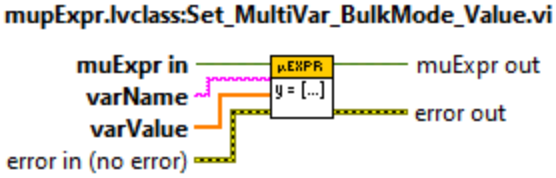
If the variable is not found in the expression, a warning is generated.

- **[x,y] = [a,b]**: Set the values of a list of variables.



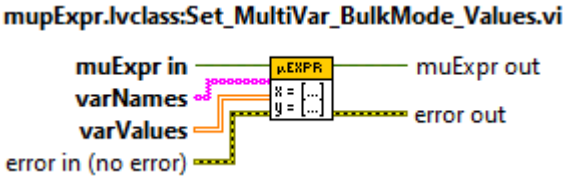
If any variable is not found in the expression, a warning is generated.

- **y = […]**: Set the bulk mode values for a selected variable.



If the variable is not found in the expression, a warning is generated.

- [x,y] = [[…],[…]]: Set the bulk mode values for a list of variables.



Note that the array of values provided for variable values should be of size:

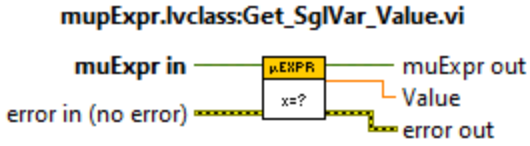
  - Num Rows: Number of Variables
  - Num Cols: Bulk Size

- If any variable is not found in the expression, a warning is generated.

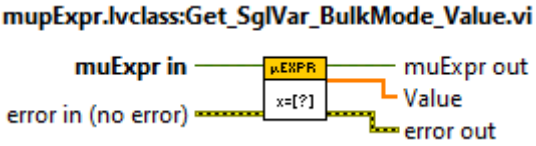
9.2 Get Variable Value

A variable's value can be retrieved using the "Get_Var_Value.vi" from the muParser Palette. This VI is polymorphic with the following instances:

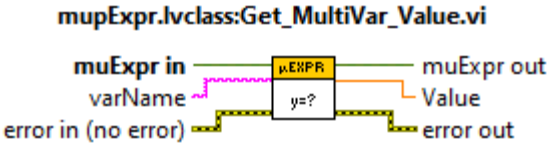
- **x = ?**: Return the expression's single variable value.



- **x = [?]**: Return the expression's single variable's bulk mode values.

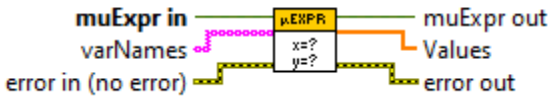


- **y = ?**: Lookup variable value by name.



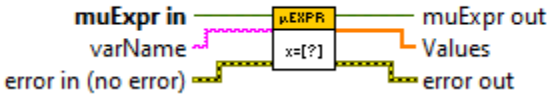
- **[x,y] = [?,?]**: Lookup multiple variable values by name.

mupExpr.lvclass:Get_MultiVar_Values.vi



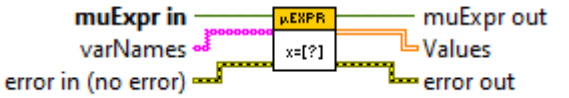
- **y = [?]**: Lookup variable's bulk mode values by name.

mupExpr.lvclass:Get_MultiVar_BulkMode_Value.vi



- **[x,y] = [[?],[?]]**: Lookup bulk mode values of multiple variables.

mupExpr.lvclass:Get_MultiVar_BulkMode_Values.vi



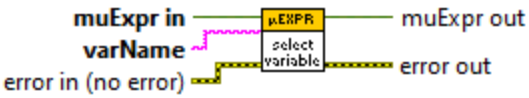
  Returns 2D array with dimensions:

  - Num Rows: Number of Variables
  - Num Cols: Bulk Size

9.3 Specify the Single Variable

By default the selected single variable is the first variable encountered by the parser (index 0 of the Variable names list). To explicitly specify which variable to use as the single variable use "Select_SglVar.vi" from the muParser Palette.

mupExpr.lvclass:Select_SglVar.vi



Select the variable, by name, for use in single variable evaluation.

Returns error if the input variable is not found.

10 Error Codes

Error Code	Description
515000	Unexpected binary operator found
515001	Token cant be identified
515002	Unexpected end of formula. (Example: "2+sin(")
515003	An unexpected argument separator has been found. (Example: "1,23")
515004	An unexpected argument has been found
515005	An unexpected value token has been found
515006	An unexpected variable token has been found
515007	Unexpected parenthesis, opening or closing
515008	A string has been found at an inappropriate position
515009	A string function has been called with a different type of argument
515010	A numerical function has been called with a non value type of argument
515011	Missing parenthesis. (Example: "3*sin(3")
515012	Unexpected function found. (Example: "sin(8)cos(9)")
515013	unterminated string constant. (Example: "3*valueof("hello)")
515014	Too many function parameters
515015	Too few function parameters. (Example: "ite(1<2,2)")
515016	binary operators may only be applied to value items of the same type
515017	result is a string
515018	Invalid function, variable or constant name.
515019	Invalid binary operator identifier.
515020	Invalid infix operator identifier.
515021	Invalid postfix operator identifier.
515022	Trying to overload built-in operator
515023	Invalid callback function pointer
515024	Invalid variable pointer
515025	The expression string is empty
515026	Name conflict
515027	Invalid operator priority
515028	catch division by zero, sqrt(-1), log(0) (currently unused)
515029	Division by zero (currently unused)
515030	Error that does not fit any other code but is not an internal error
515031	Conflict with current locale

515032	Unexpected if then else operator
515033	Missing else clause
515034	Misplaced colon
515035	The vectors submitted to bulk mode computations are too short.
515036	A submitted identifier longer than 100 characters.
515037	The submitted expression is longer than 5000 characters.
515038	Internal error of any kind.
515039	Internal error of any kind.
515104	Variable "%s" not found in expression
515102	**Warning** Variable name "%s" not found in expression "%s"

11 References

Berg, I. (2023, 07 31). Retrieved from muparser - Fast Math Parser Library: https://beltoforion.de/en/muparser/