

11 Expression Block Lab

11.1 Foreword

The Expression Block is one of the most powerful blocks and perhaps the most often used block in MOD 30ML. This lab will provide a series of partial applications where an Expression Block would be the best tool to solve the requirement. Enter the appropriate expression and input names. Be prepared to demonstrate to the class your proposal for each problem.

11.2 Objectives

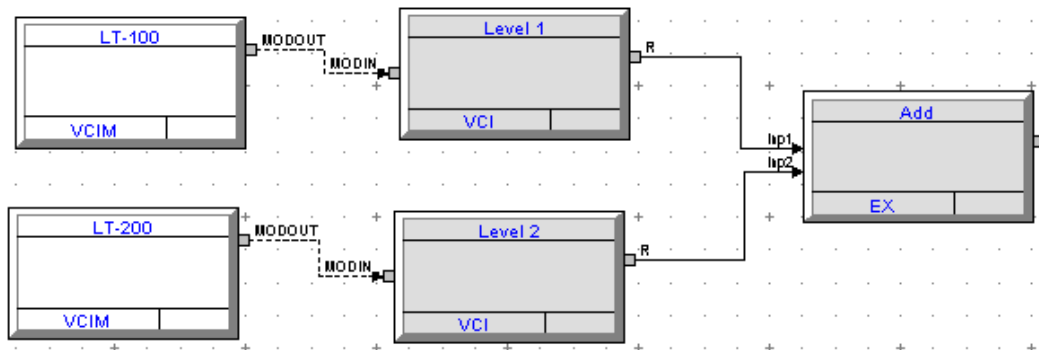
In this lab you will configure 3 process applications using the Expression block. Refer to the example given below and the reference material provided in this chapter to get more information about the EX block.

11.3 Instructions

11.3.1 Example: Add 2 Inputs *(Note: See the end of this section for operators and rules)*

1. Configure 2 analog inputs and their corresponding Signal conditioning (Input Function) blocks as shown in the next figure. See the end of this section for additional information.
2. Configure an EX block:
 - Add 2 inputs under the **Inputs** tab of the EX block by typing names **inp1** and **inp2** and data type as **Floating Point**.

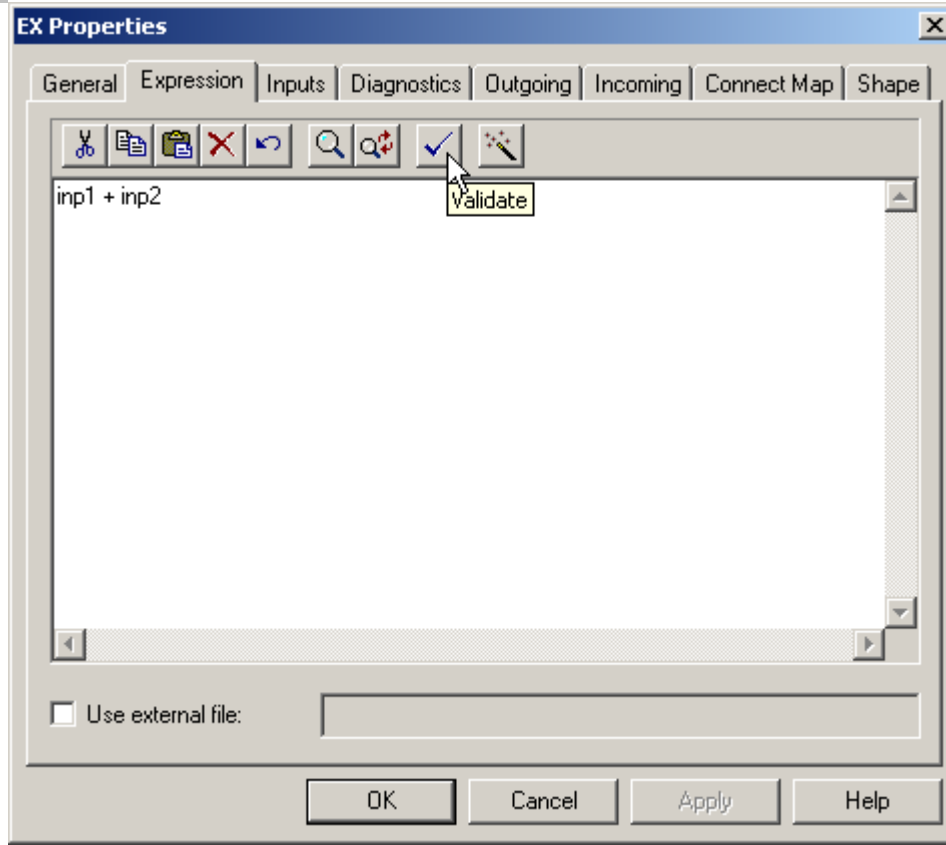
Figure 11 .1.
EX block Example



- Type the expression: **inp1 + inp2** under the **Expression** tab.
- Click on the Validate button to verify that the syntax is correct. See the figure below:

Expression Block Lab

Figure 11 .2.
EX block Example



- Specify the data type for the result as Floating point under the General tab of the EX block.
3. Compile the database.

Use the above techniques to configure the following 3 examples:

11.3.2 Application #1

Find the average pressure between four boiler headers.

Hints: Add 4 analog input (4-20 mA) and corresponding Signal conditioning blocks. Add an EX block and configure 4 inputs. Use these 4 inputs in your expression to find the average.

11.3.3 Application #2

Turn on a fan when the temperature is greater than 82 degrees and provided the room door is not open.

Hints: You need a thermocouple or RTD input and a signal conditioning blocks. Configure a DIM block for the room door and a DOM block for the fan. Use If Then Else statement or simple mathematical expression in the EX block.

11.3.4 Application #3

Select the one input out of 4 that is the lowest value.

Hints: Configure 4 inputs. There are more than one ways of configuring this in the EX block. One of them is to use nested If Then Else statements. Take one input at a time and compare that with the other 3 inputs to see if it is the lowest.

11.4 Expression Operation

Operator precedence in an expression starts with the unary (single operand) operators and continues with the binary (double operand) operators. The order of evaluation can be changed using parentheses or the conditional operators. The order of precedence is:

Precedence	Operator	
1	**	RAISED_TO_THE_POWER
2	SQRT	SQUARE_ROOT
	MOM	MOMENTARY
	!	Logical NOT
	ABS	ABSOLUTE
	EXP	EXPONENTIAL
	NLOG	NATURAL_LOG
	LOG	LOG_10
	INT	INTEGER
3	*	TIMES
	/	DIVIDE
4	+	PLUS
	-	MINUS
5	<	LESS_THAN
	>	GREATER_THAN
	<=	LESS_THAN_OR_EQUAL
	>=	GREATER_THAN_OR_EQUAL
6	==	EQUALS
	!=	UNEQUAL
7	&&	Logical AND
8		Logical OR

11.4.1 General Rules

For Boolean operators, data of any type is considered TRUE if non-zero. For logical operations, all operands are scaled to DISCRETE during evaluation.

For comparison or arithmetic operations, all operands are scaled to floating point during evaluation. DATE operands are expanded to include the century and treated as 4 byte unsigned integers. Prior to applying the operator(s), if the year is in the range 0–89, 100 is added to make comparisons work properly across the calendar year 2000 (valid for 1990 to 2089). Thus 3/18/95 (stored internally as \$5F0312) is treated as \$005F0312, or decimal 6226706, while 7/25/2012, (stored internally as \$0C0719) is treated as \$00700719, or decimal 7341849. HEX and ASCII operands are not allowed.

Evaluation Results

The result of an expression evaluation is DISCRETE if the last operator processed was logical, or FLOATING POINT if it was arithmetic. This result is then scaled to the configured data types for the result and auxiliary result outputs of the expression block.

The MOMENTARY Operator

The MOMENTARY operator allows a user to embed an ‘edge detection’ in an expression. The result of the MOMENTARY operation is TRUE only if its operand was FALSE in the most recent evaluation and is now TRUE. Note that the MOMENTARY operator is distinct from the MOMENTARY DISCRETE input data type.

The INTEGER Operator

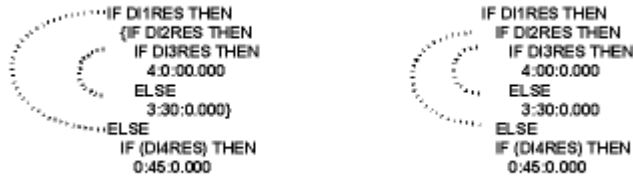
The INTEGER operator removes the decimal portion of a number, leaving only the integer portion. It does not round the number. For example, $\text{INT}(12.678) = 12$ and $\text{INT}(-746.21) = -746$.

The Conditional Operators

The conditional operators (IF, THEN, and ELSE) let you specify when operations are evaluated. The IF expression (between the IF and THEN conditionals) is evaluated first. This expression may be enclosed in parentheses. When the calculated value of the IF expression is TRUE (non-zero), the THEN expression is evaluated. When it is FALSE (zero), the ELSE expression is evaluated. The ELSE expression is optional. When the IF expression evaluates to FALSE and there is no ELSE, the result is not updated. Brackets ({}) should be used for nesting conditionals. If brackets are not present, an ELSE expression is paired with the last unpaired IF. Notice how the brackets change the pairing of the IF and ELSE expressions in the following example.

Figure 11 .3.

If Then Else



Using Expression Blocks as Recipes

Expression block inputs can be used as a recipe. Any number of local or remote inputs to the block can be configured without using them in the expression, and for this purpose only, HEX and ASCII inputs are allowed.

Syntax Errors

When there is a syntax error in the expression or a stack overflow during evaluation (can only happen with a very deeply nested expression), the expression error diagnostic will be reported, the results will not be updated and output qualities will be set bad.

Momentary Discrete Local Input

The value of a MOMENTARY DISCRETE local input is changed back to FALSE whenever a TRUE is found so that it is detected in only one evaluation of an expression. The momentary discrete feature allows a user to embed a 'push-button' in an expression. MOMENTARY DISCRETE is a unique data type which can only be used as a local input to an expression block. Another block pointed at this input will see a DISCRETE data type. A local input of this type is configured by setting the local data to HIGH or LOW.