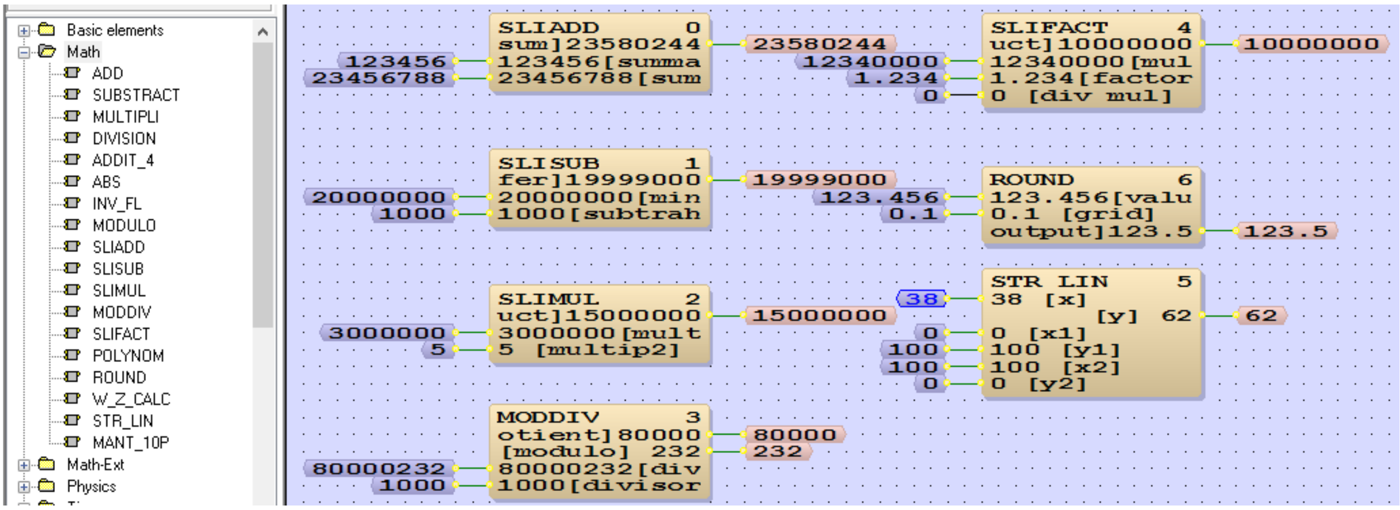
# TT200302 – FUP - SLI and Misc Math Functions

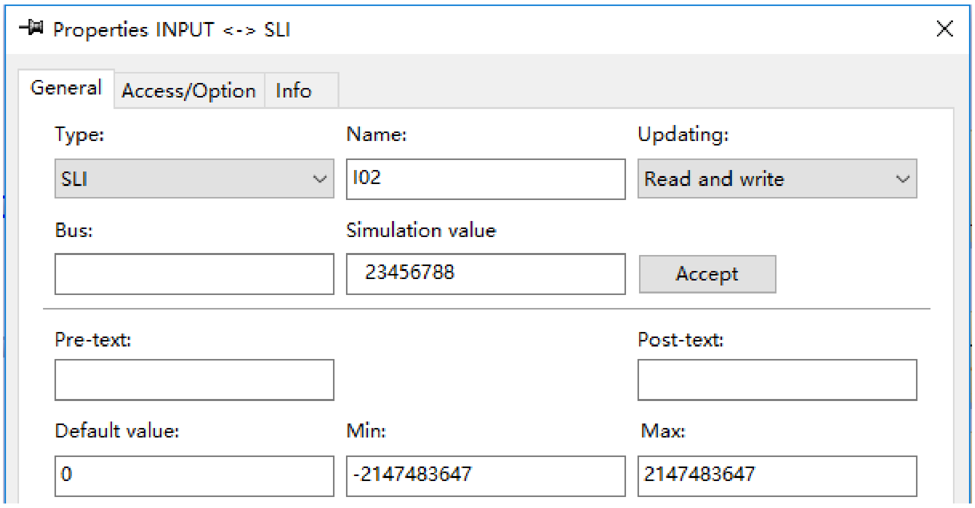
| | |
|---|---|
| **Note** | This Support Knowledge Base article KB is the result of a support request.<br>It is not part of the official documentation of DEOS AG and does not claim to be complete.<br>The article is intended to support the solution of a similar problem.<br>If you have any questions, comments or additions, please contact DEOS AG Support. |
| **Title** | SLI and Misc Math Functions (TT200302) |
| **Object** | FUP |
| **Reference version** | 2 |
| **Date** | 03.2020 |
| **Author** | EK |
| **Goal** | To explain the usage of the SLI and Misc Math Function Blocks |

**Content:**



# TT200302 – FUP - SLI and Misc Math Functions

1. In FUP, most of the time we use floating point for analog value, and use the normal “Math” modules for the calculation. But for some situations, e.g. power meter kWh value, the number is so large that floating point is not accurate enough for them. In that case, we can use type SLI (signed large Integer) that can handle up to a value of 2,147,483,647.



Properties INPUT <-> SLI

General | Access/Option | Info

Type: SLI

Name: I02

Updating: Read and write

Bus:

Simulation value: 23456788

Pre-text:

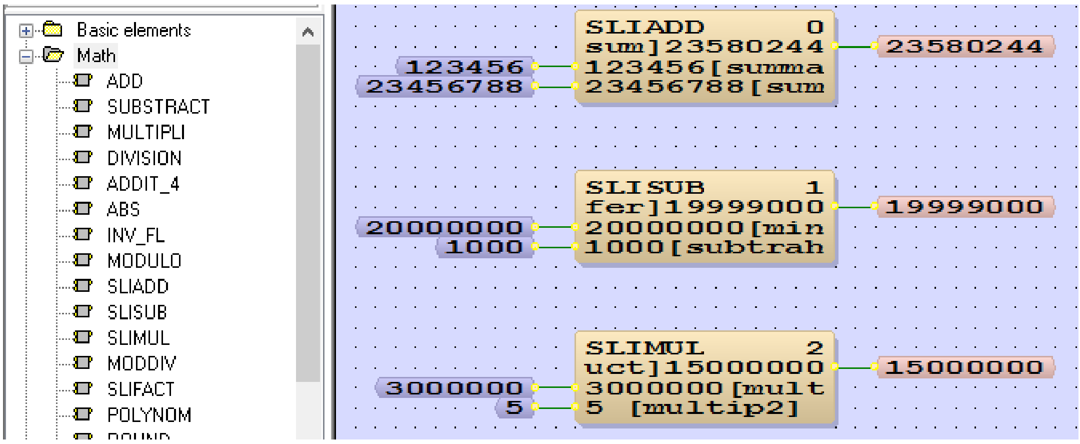
Post-text:

Default value: 0

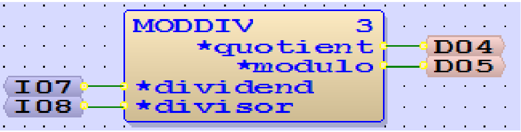
Min: -2147483647

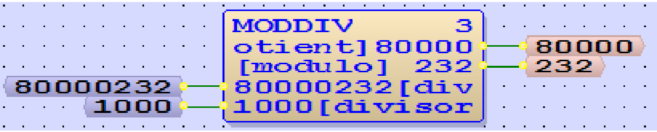
Max: 2147483647

2. Since type SLI is a 32 bits integer, so we have some special modules for the calculation, e.g. SLIADD, SLISUB and SLIMUL. They are under the “Math” group

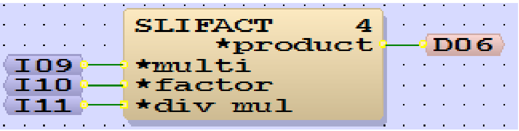


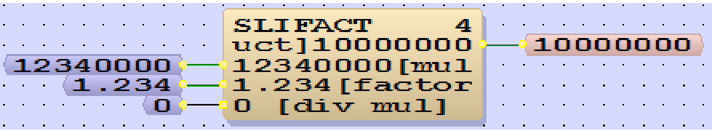
3. In integer division and modulus, the dividend is divided by the divisor into an integer quotient and a remainder. The integer quotient operation is referred to as integer division, and the integer remainder operation is the modulus. In FUP, the module is “MODDIV”





4. In some cases, we want to multiply or divide an SLI number with a floating point number, and return an SLI number. In this case, we can use the SLIFACT module. If we set the input “div_mul” to 0, then it becomes a division module, like below example

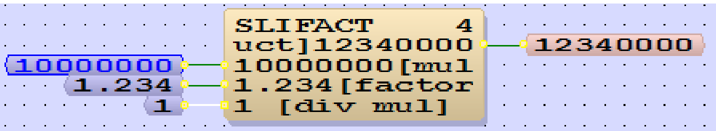




5. Please note that in FUP simulation, all the inputs and outputs are simulated as floating point numbers, so sometimes you may get incorrect results if the SLI number is very large because

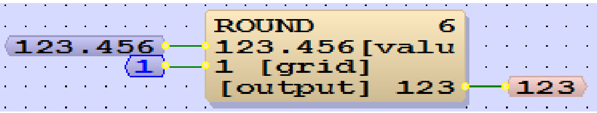
of the accuracy of the floating point number. Once you compile and upload it to the controller, the calculation will become correct.

6. If you set the input “div_mul” to 1, then it becomes a multiply module, like below example

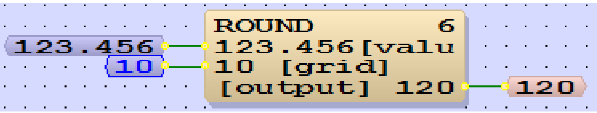


```
            SLIFACT        4
            uct]12340000 ── 12340000
10000000 ── 10000000[mul
   1.234 ── 1.234[factor
       1 ── 1  [div mul]
```

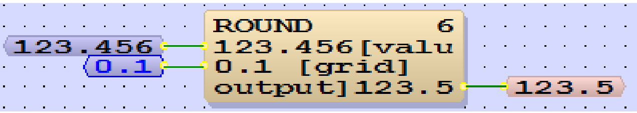
7. The next module is a module for floating point number called “ROUND”. Rounding means making a number simpler but keeping its value close to what is was. In the “ROUND” module, we can set the “approximately” using the input “grid”. Here are some examples



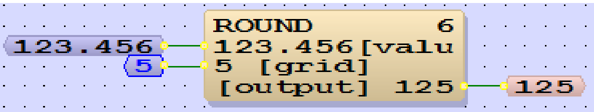
```
         ROUND          6
123.456 ── 123.456[valu
      1 ── 1  [grid]
           [output] 123 ── 123
```



```
         ROUND          6
123.456 ── 123.456[valu
     10 ── 10  [grid]
           [output] 120 ── 120
```

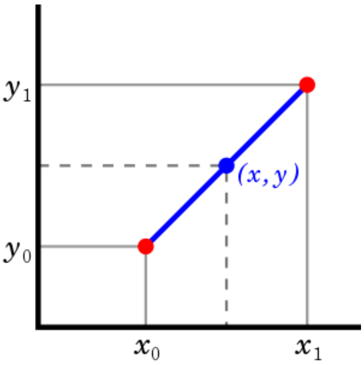


```
         ROUND          6
123.456 ── 123.456[valu
    0.1 ── 0.1  [grid]
           output]123.5 ── 123.5
```

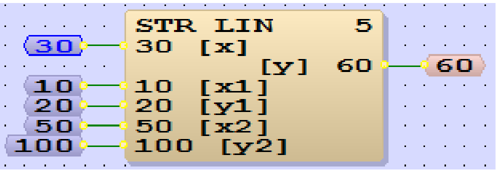


```
         ROUND          6
123.456 ── 123.456[valu
      5 ── 5  [grid]
           [output] 125 ── 125
```

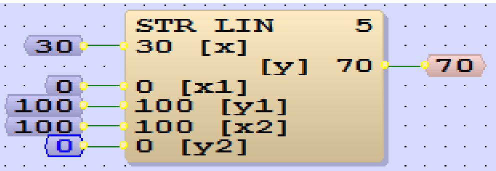
8. The last module is “STR_LIN”. This is the linear interpolation module



9. Below are 2 examples for your reference.

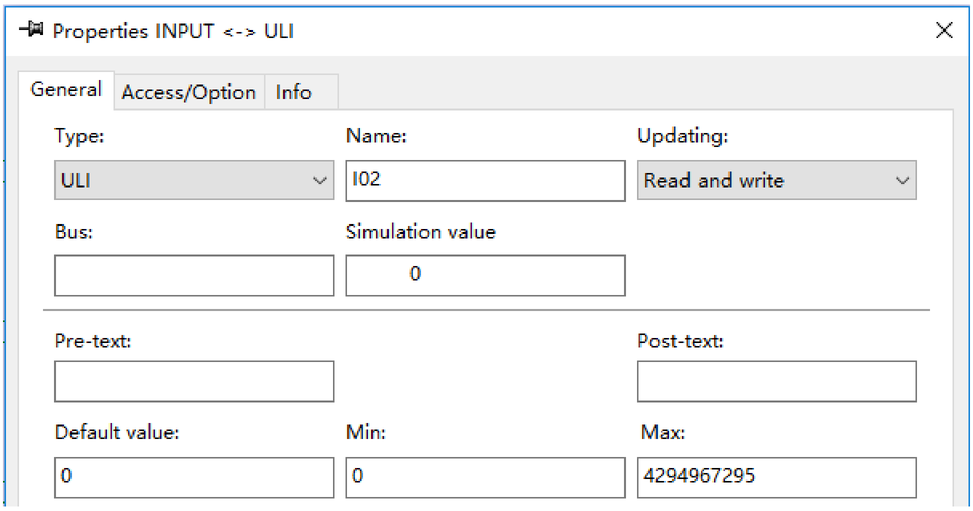


```
      STR LIN       5
 30 ── 30  [x]
           [y]  60 ── 60
 10 ── 10  [x1]
 20 ── 20  [y1]
 50 ── 50  [x2]
100 ── 100  [y2]
```



```
      STR LIN       5
 30 ── 30  [x]
           [y]  70 ── 70
  0 ── 0  [x1]
100 ── 100  [y1]
100 ── 100  [x2]
  0 ── 0  [y2]
```

10. We also have a data type ULI which is unsigned large integer, which can handle value up to 4,294,967,295. Finally, in the FUP “Modifier” group, you can find many modules that converts number between different data types



Properties INPUT <-> ULI

Type: ULI

Name: I02

Updating: Read and write

Bus:

Simulation value: 0

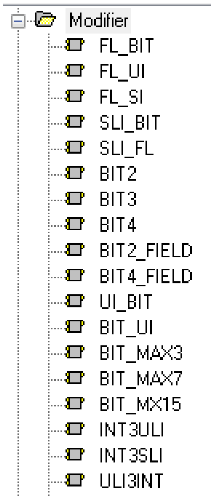
Pre-text:

Post-text:

Default value: 0

Min: 0

Max: 4294967295



Modifier
- FL_BIT
- FL_UI
- FL_SI
- SLI_BIT
- SLI_FL
- BIT2
- BIT3
- BIT4
- BIT2_FIELD
- BIT4_FIELD
- UI_BIT
- BIT_UI
- BIT_MAX3
- BIT_MAX7
- BIT_MX15
- INT3ULI
- INT3SLI
- ULI3INT