# $i^3$ Mathematical Function Tutorial

## Introduction

The purpose of this tutorial is to demonstrate the i3 mathematical functions and graphical representation capabilities with bar graphs, trends and meters.

The i3 has basic mathematical functions such as add, multiply, subtract and divide but it has can perform floating-point maths. Therefore, the i3 has sine, cosine and tan functions amongst others in special Advanced Maths operations.

## Programming Maths Functions

Design a program where the user can enter the year they were born and the i3 will calculate how many days they have been alive for (approximately). The program will then have another screen where the user will enter the radius of a circle and the i3 will calculate the area, circumference and diameter. Finally, we will set up the trend function to display the result of a sine calculation.

## Memory requirements of mathematical functions

The i3 can handle floating point maths as well as standard integer maths. These calculations require different data types and the different data types require different sizes of memory to store them.

### Data Types

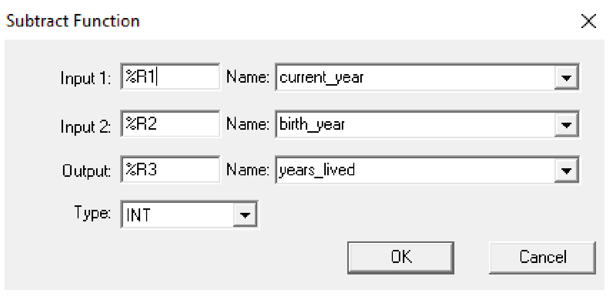
| | |
|---|---|
| BOOL - | Boolean; A single bit. It can contain only the values '0' or '1', a.k.a 'FALSE' or 'TRUE' |
| BYTE - | Byte; A string of 8 consecutive bits. Byte format is used more where the value of the data is not as important as the bit patterns (shifts and rotates). |
| WORD - | Word; A string of 16 consecutive bits. Word format is used more where the value of the data is not as important as the bit patterns (shifts and rotates). |
| DWORD - | Double Word; A string of 32 consecutive bits. DWORD format is used where the value of the data is not as important as the bit patterns (shifts and rotates). |
| INT - | Integer; A 16-bit signed value. Integers are used where the value of the data is expected to be in the range of -32,768 to +32,767 |
| SINT - | Short Integer; An 8-bit signed value. Short Integers are used where the value of the data is expected to be in the range of -128 to +127. |
| DINT - | Double Integer; A 32-bit signed value. Double Integers are used where the value of the data is expected to be in the range of -2,147,483,648 to +2,147,483,647. |
| UINT - | Unsigned Integer; A 16-bit unsigned value. Unsigned Integers are used where the value of the data is expected to be in the range of -0 (zero) to 65,535. |
| USINT - | Unsigned Short Integer; An 8-bit unsigned value. Unsigned Short Integers are used where the value of the data is expected to be in the range of 0 (zero) to 255 |
| UDINT - | Unsigned Double Integer; A 32-bit unsigned value. Unsigned Double Integers are used where the value of the data is expected to be in the range of 0 (zero) to 4,294,967,296. |
| REAL - | Floating Point; A 32-bit value. Values are stored and operated on in IEEE single precision (six digit) format. Values range from -3.40282E+38 to +3.40282E+38. |
| STRING - | String; A variable-length succession of characters. Each character is represented by one byte. |

## Programming the Ladder Logic

We will begin by writing the code for the days lived calculation. First, we will subtract the year of birth from the current year to find the years lived. Then multiply the years lived by 365 (days in a year) to find the approximate days lived.

Insert a N/O contact at A1 and assign to %S07, Always ON (ALW_ON). This is so that the function blocks on this rung are always enabled.

Select the Subtract function from the maths operations menu and insert it into the rung and enter the following detail.


Subtract Function

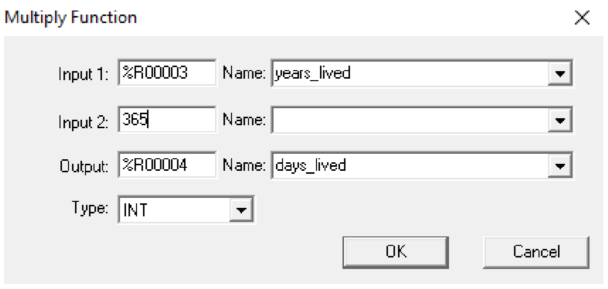
Input 1: %R1 Name: current_year

Input 2: %R2 Name: birth_year

Output: %R3 Name: years_lived

Type: INT

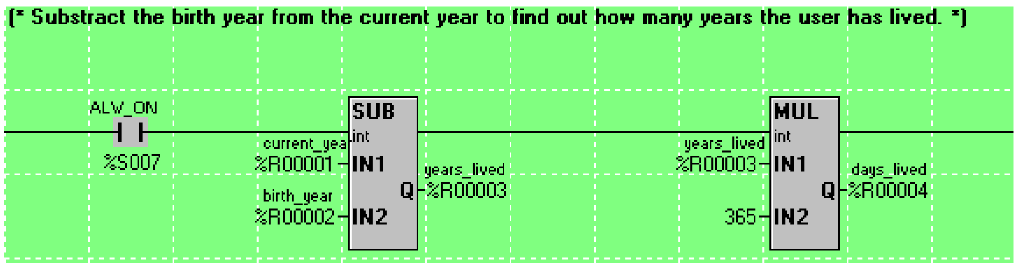
Next insert a Multiply function to the right of the Subtract function on the same rung.
Enter the details as shown below.



Multiply Function

| | | | |
|---|---|---|---|
| Input 1: | %R00003 | Name: | years_lived |
| Input 2: | 365 | Name: | |
| Output: | %R00004 | Name: | days_lived |
| Type: | INT | | |

The line of ladder logic should now look like the logic below.

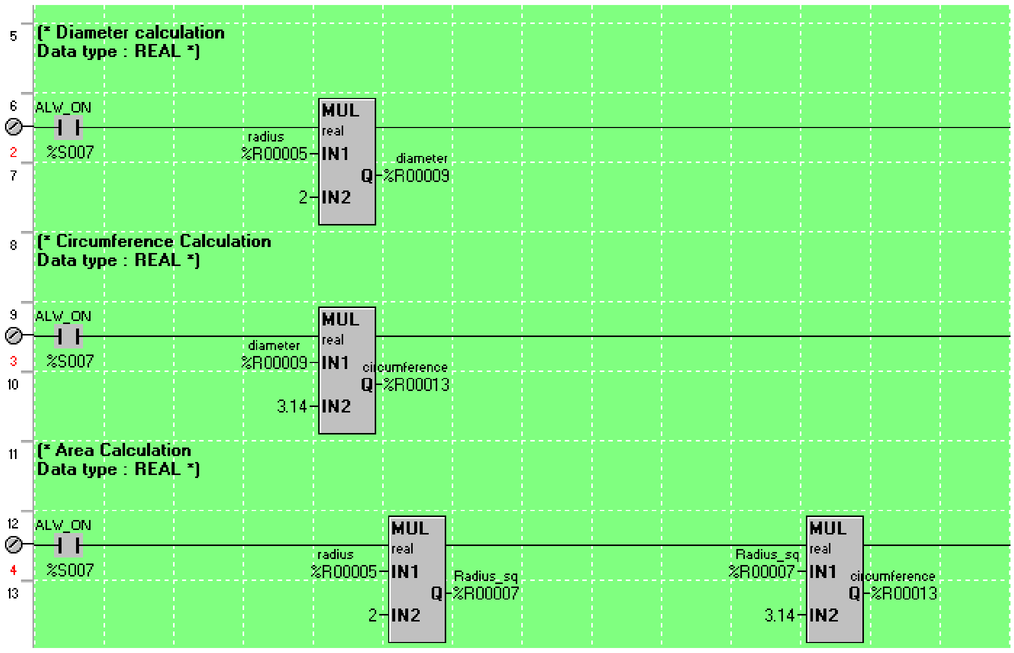


(* Substract the birth year from the current year to find out how many years the user has lived. *)

The formulas to calculate the diameter, circumference and area of a circle from the radius are as follows:

| | |
|---|---|
| $d = 2r$ | $d$ = diameter |
| $circ = 2\pi r$ | $r$ = radius |
| $area = 2\pi r^2$ | $circ$ = circumference |
| | $\pi = 3.14$ |

We are going to perform each calculation on a separate line, this is just so that it is easy to view. Each line will begin with an Always On (ALW_ON) contact, so that the functions are always enabled.



(* Diameter calculation
Data type : REAL *)

(* Circumference Calculation
Data type : REAL *)

(* Area Calculation
Data type : REAL *)

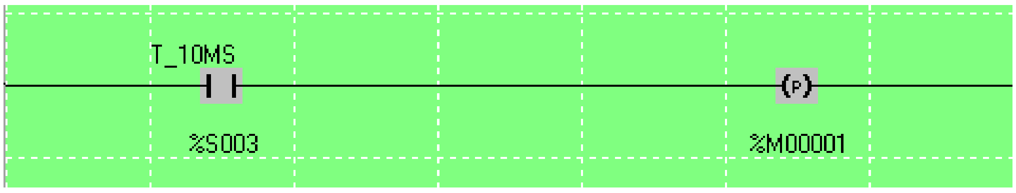
All of the calculations performed are using the data format Real. These require two registers, please note that the register addresses are at least 2 consecutive numbers apart. i.e. the variable “radius” is stored in %R05 and %R06 but it is only addressed to %R05.

Lastly, we are going to write the logic for the trend.

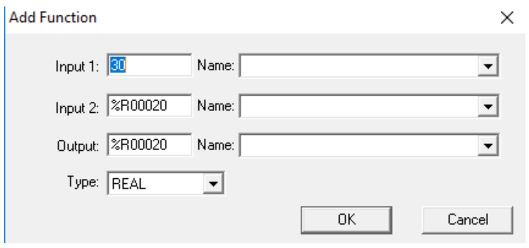
The sine function is going to operate on a set of values; 0-360 in steps of 30. To achieve this, we are going to use a ten-millisecond pulse to add 30 to a register until it equals a set value.

# $i^3$ Mathematical Function Tutorial

First insert a N/O contact and address it to %S03 10mS system pulse. This will act as the trigger for the addition function, then on the same rung add a N/O coils that operates on a Pulse, addressing it to %M01. Then on the Rung below insert a N/O contact in the first column addressing it to %M01. This gives us an action that operates only on the rising edge of a transition.



Next add the addition function from the maths operations and insert it in the same rung as the N/O contact. Next insert an Equal to function from the Compare operations to the right of the addition block. Finally insert a Move function (MOV) to move 0 into the register when it's equal to 360. Insert the details into the functions as shown below.



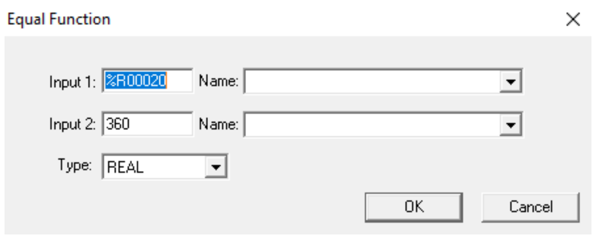
Add Function

Input 1: 30 Name:

Input 2: %R00020 Name:

Output: %R00020 Name:

Type: REAL

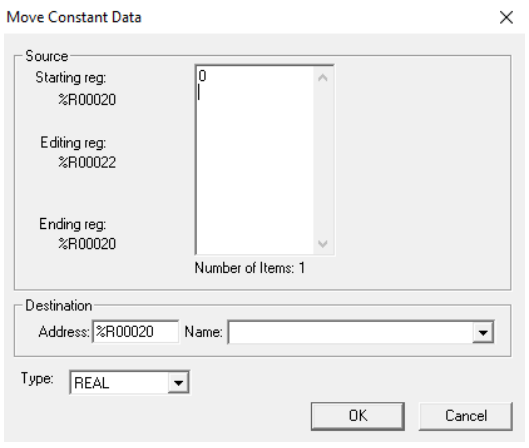


Equal Function

Input 1: %R00020 Name:

Input 2: 360 Name:

Type: REAL



Move Constant Data

Source

Starting reg: %R00020

Editing reg: %R00022

Ending reg: %R00020

0

Number of Items: 1

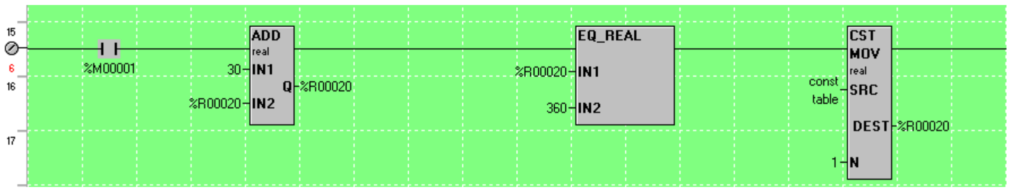
Destination

Address: %R00020 Name:

Type: REAL

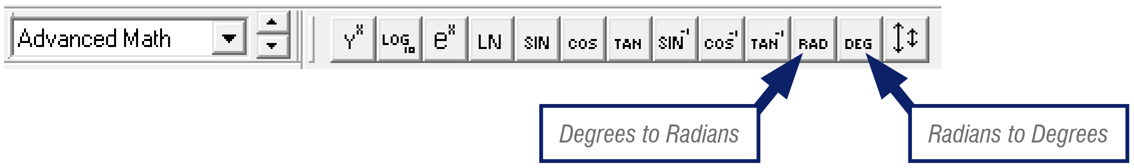
A constant move function was use as it allows for the movement of real values and keeps the calculation constant in Real data.

The rung of code should now like the ladder logic below.



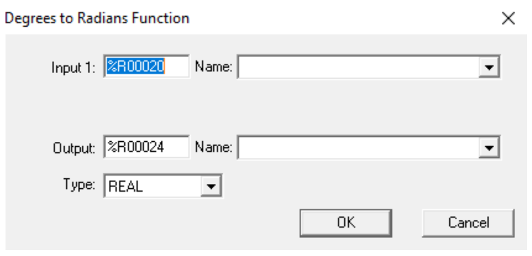
Now that the data to be operated on has been configured, insert a N/O contact on the Run below, in the first rung and assign to the Always On (ALW_ON) system bit. On this rung we are going to insert two conversion of data type functions and the sine function.

The sin, cos and tan functions all operate in radians. Thus, the input to the function must be in radian and so the first function to insert on this rung will be a degree to radian function. This function is in the advanced math operations.



# $i^3$ Mathematical Function Tutorial

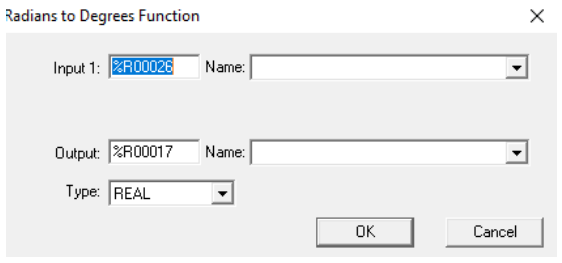
To the right of the degree to radian function we are going to insert the sine function. Select the sine function from the advanced maths operations. Lastly, we are going to insert a conversion function to convert the function back from radian to degrees. Which will be more understandable to us.


Degrees to Radians Function

Input 1: %R00020 Name:

Output: %R00024 Name:

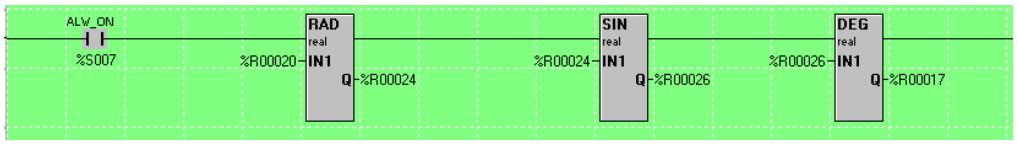
Type: REAL

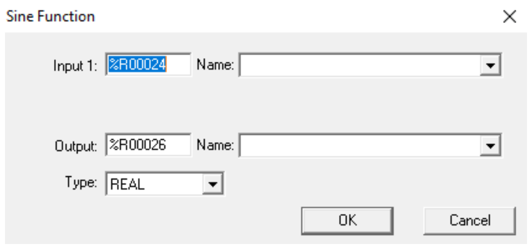

Radians to Degrees Function

Input 1: %R00026 Name:

Output: %R00017 Name:

Type: REAL




Sine Function

Input 1: %R00024 Name:

Output: %R00026 Name:

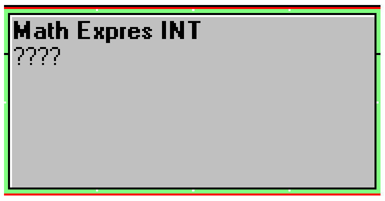
Type: REAL

NOTE: Real values require 2 consecutive registers.

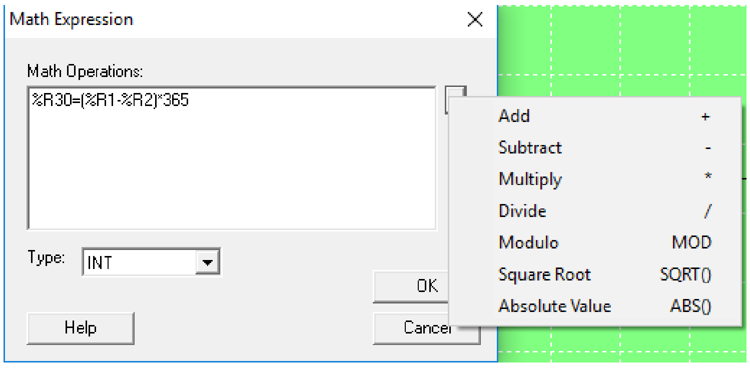
## Equation Function

Calculations could also be performed in the equation function using only one function block and saving on memory.

The equation function allows the user to enter an equation into a single function block. To select the function block, click on the icon (x+y) in the maths operations menu.



Double click on the function block to open up the editor.


Math Expression

Math Operations:

```
%R30=(%R1-%R2)*365
```

Type: INT

| | |
|---|---|
| Add | + |
| Subtract | - |
| Multiply | * |
| Divide | / |
| Modulo | MOD |
| Square Root | SQRT() |
| Absolute Value | ABS() |

By clicking on the arrow, a menu pops up to display what mathematical operations are valid in this function block

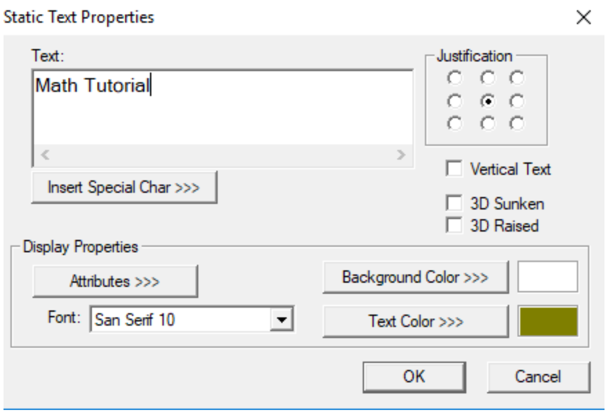
Enter the expression to calculate the days lived equation into the function as shown, starting with the result register equal to the expression. If you want to use the value in a register, then enter the register address as shown.

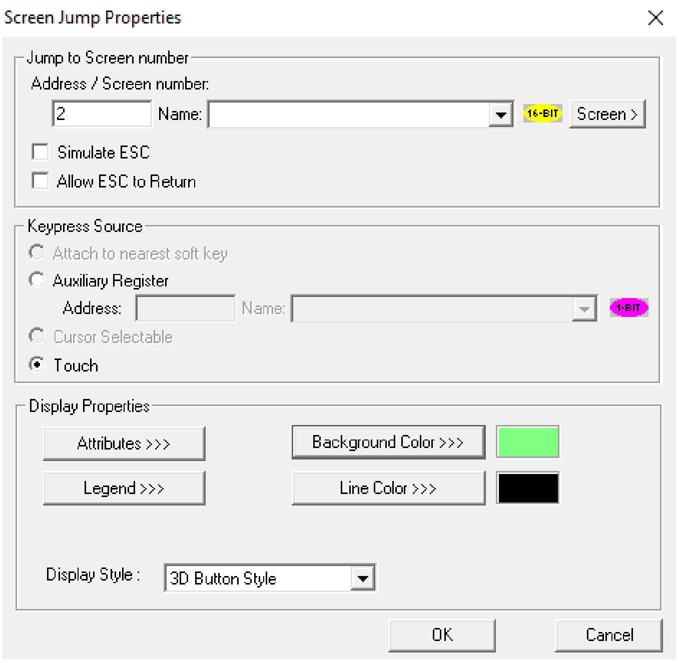
# $i^3$ Mathematical Function Tutorial

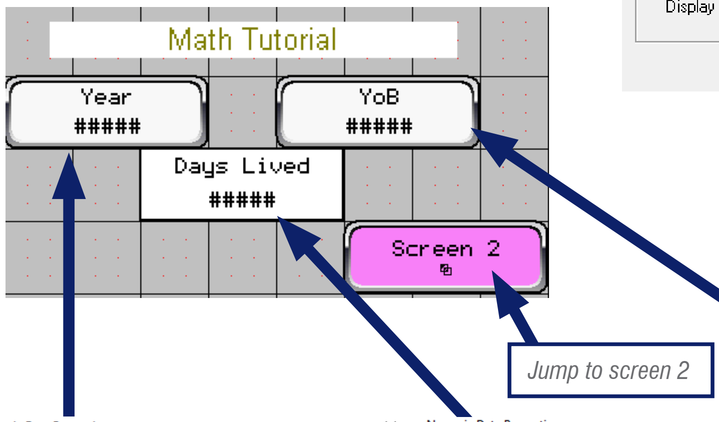
## Screen Editor Programming

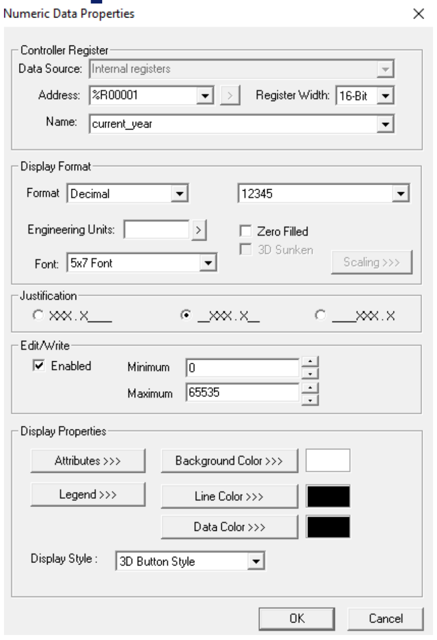
With all the ladder logic configured, we now need to edit the screens to display the information appropriately.

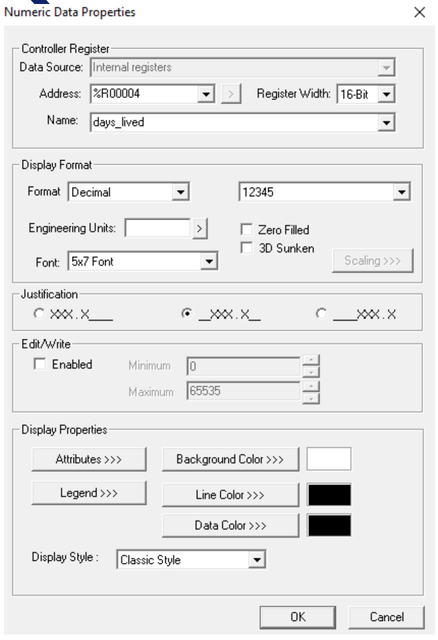
On the first screen we are going to have the user able to enter the current year and their year of birth. The screen will then display the days lived calculation result. There will also be a screen jump button so that the user can scroll through the screen.

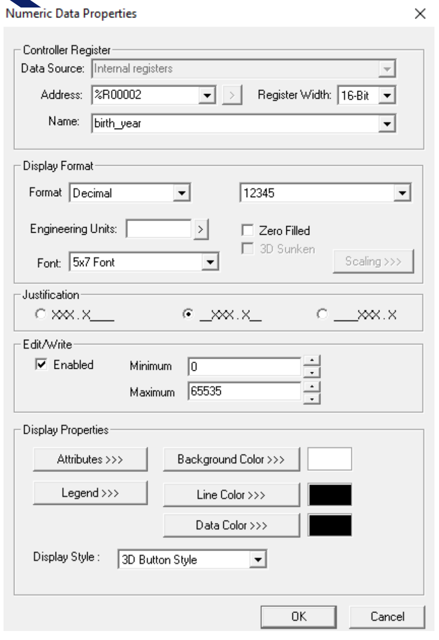

Static Text Properties
Text: Math Tutorial
Justification: [✓ centre]
Vertical Text: ☐
3D Sunken: ☐
3D Raised: ☐
Display Properties
Font: San Serif 10
Background Color >>> ⬜
Text Color >>> 🟫


Screen Jump Properties
Jump to Screen number
Address / Screen number: 2
Name:
Simulate ESC: ☐
Allow ESC to Return: ☐
Keypress Source
Attach to nearest soft key
Auxiliary Register
Address:
Name:
Cursor Selectable
[✓ Touch]
Display Properties
Background Color >>> 🟩
Line Color >>> ⬛
Display Style : 3D Button Style



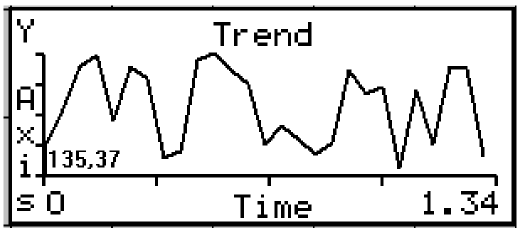

Numeric Data Properties
Controller Register
Data Source: Internal registers
Address: %R00001
Register Width: 16-Bit
Name: current_year
Display Format
Format: Decimal
12345
Engineering Units:
Zero Filled: ☐
3D Sunken: ☐
Font: 5x7 Font
Justification: ○ XXX.X___ [✓ _XXX.X_] ○ ___XXX.X
Edit/Write
Enabled: ☑
Minimum: 0
Maximum: 65535
Display Properties
Background Color >>> ⬜
Line Color >>> ⬛
Data Color >>> ⬛
Display Style : 3D Button Style


Numeric Data Properties
Controller Register
Data Source: Internal registers
Address: %R00004
Register Width: 16-Bit
Name: days_lived
Display Format
Format: Decimal
12345
Engineering Units:
Zero Filled: ☐
3D Sunken: ☐
Font: 5x7 Font
Justification: ○ XXX.X___ [✓ _XXX.X_] ○ ___XXX.X
Edit/Write
Enabled: ☐
Minimum: 0
Maximum: 65535
Display Properties
Background Color >>> ⬜
Line Color >>> ⬛
Data Color >>> ⬛
Display Style : Classic Style


Numeric Data Properties
Controller Register
Data Source: Internal registers
Address: %R00002
Register Width: 16-Bit
Name: birth_year
Display Format
Format: Decimal
12345
Engineering Units:
Zero Filled: ☐
3D Sunken: ☐
Font: 5x7 Font
Justification: ○ XXX.X___ [✓ _XXX.X_] ○ ___XXX.X
Edit/Write
Enabled: ☑
Minimum: 0
Maximum: 65535
Display Properties
Background Color >>> ⬜
Line Color >>> ⬛
Data Color >>> ⬛
Display Style : 3D Button Style

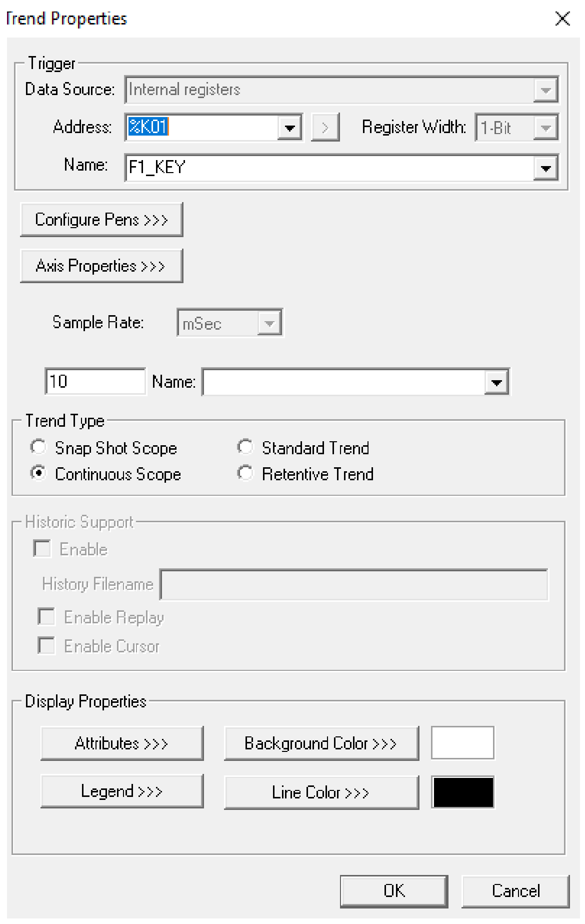
# $i^3$ Mathematical Function Tutorial

On the second screen we are going to display the trend of the sine function and have a screen jump to the next screen.



Click on the trend icon and insert it into the middle of the screen. Resize the box to get a bigger trend, covering almost the full screen.

Setting the trigger to key 1 and having the trend type as a continuous scope, means that when the key is pressed there will be a trend, when it is releasing the trend stops.



Trend Properties

Trigger
Data Source: Internal registers
Address: %K01
Register Width: 1-Bit
Name: F1_KEY

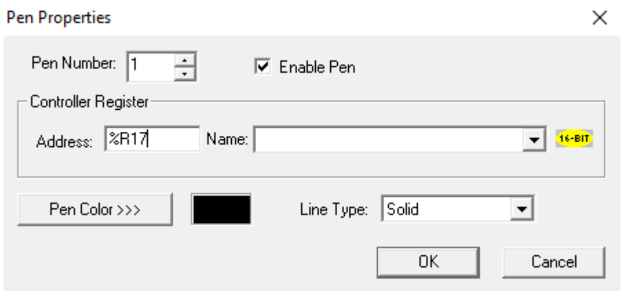
Sample Rate: mSec
10
Name:

Trend Type
Snap Shot Scope
Standard Trend
[✓ Continuous Scope]
Retentive Trend

Historic Support
☐ Enable
History Filename
☐ Enable Replay
☐ Enable Cursor

Display Properties

Set the pen to match the output register of the sin function

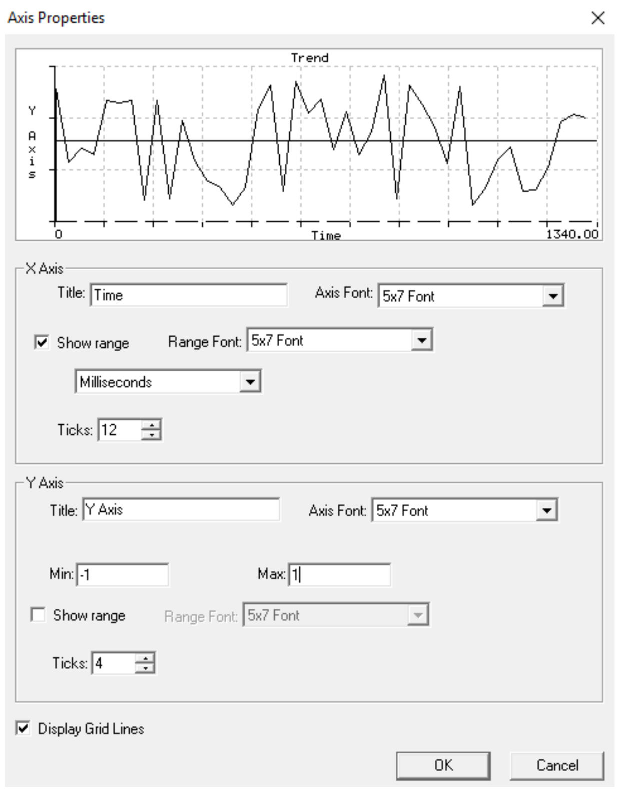


Pen Properties

Pen Number: 1
☑ Enable Pen

Controller Register
Address: %R17
Name:
16-BIT

Line Type: Solid



Axis Properties

X Axis
Title: Time
Axis Font: 5x7 Font
☑ Show range
Range Font: 5x7 Font
Milliseconds
Ticks: 12

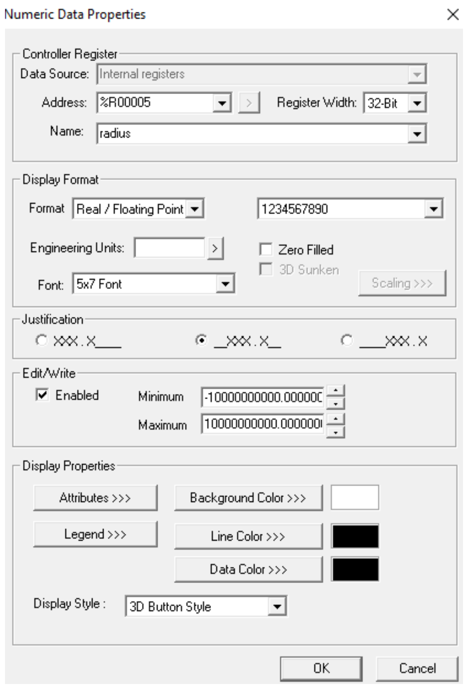
Y Axis
Title: Y Axis
Axis Font: 5x7 Font
Min: -1
Max: 1
☐ Show range
Range Font: 5x7 Font
Ticks: 4

☑ Display Grid Lines

Set up the axis properties as shown opposite.

# $i^3$ Mathematical Function Tutorial

On the last screen we are going to have a screen jump back to the first screen, so the user can cycle through. We are also going to display the circle calculations. The user will only be able to enter into one numeric box.



Numeric Data Properties

Controller Register
Data Source: Internal registers
Address: %R00005 Register Width: 32-Bit
Name: radius

Display Format
Format Real / Floating Point 1234567890
Engineering Units:
[ ] Zero Filled
[ ] 3D Sunken
Font: 5x7 Font

Justification
○ XXX . X___ [✓ ● __XXX . X__] ○ ___XXX . X

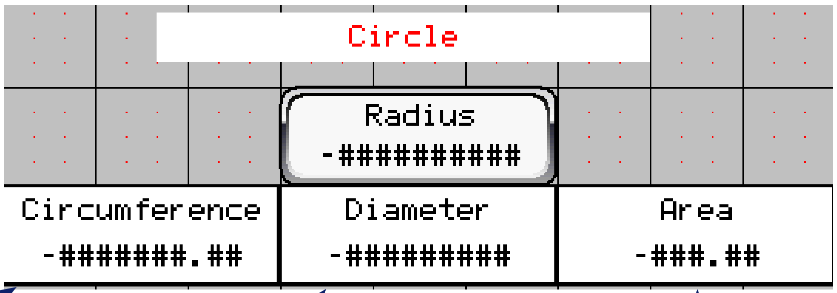
Edit/Write
[✓ Enabled] Minimum -10000000000.00000C Maximum 10000000000.000000

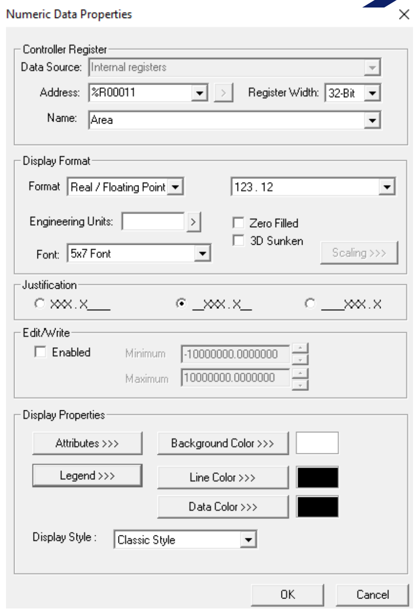
Display Properties
Display Style : 3D Button Style

Set the screen up with the 4 numeric data functions. Set only the Radius to be editable and the others are non-editable.

They should all be set to Real / Floating point to match the data type used in the ladder logic.

Enter suitable legends and if you can enter the Engineering units as you want.





Numeric Data Properties

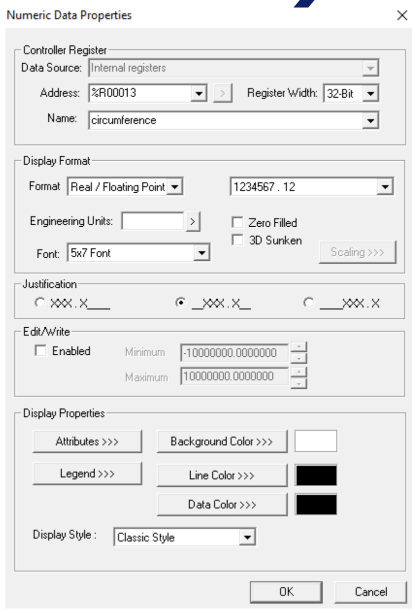
Controller Register
Data Source: Internal registers
Address: %R00011 Register Width: 32-Bit
Name: Area

Display Format
Format Real / Floating Point 123 . 12
Engineering Units:
[ ] Zero Filled
[ ] 3D Sunken
Font: 5x7 Font

Justification
○ XXX . X___ [✓ ● __XXX . X__] ○ ___XXX . X

Edit/Write
[ ] Enabled Minimum -10000000.0000000 Maximum 10000000.0000000

Display Properties
Display Style : Classic Style



Numeric Data Properties

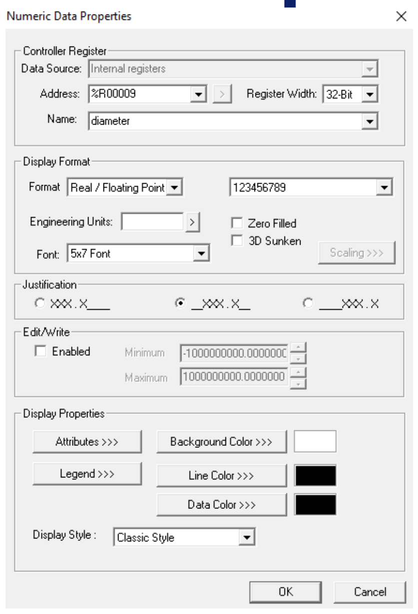
Controller Register
Data Source: Internal registers
Address: %R00013 Register Width: 32-Bit
Name: circumference

Display Format
Format Real / Floating Point 1234567 . 12
Engineering Units:
[ ] Zero Filled
[ ] 3D Sunken
Font: 5x7 Font

Justification
○ XXX . X___ [✓ ● __XXX . X__] ○ ___XXX . X

Edit/Write
[ ] Enabled Minimum -10000000.0000000 Maximum 10000000.0000000

Display Properties
Display Style : Classic Style



Numeric Data Properties

Controller Register
Data Source: Internal registers
Address: %R00009 Register Width: 32-Bit
Name: diameter

Display Format
Format Real / Floating Point 123456789
Engineering Units:
[ ] Zero Filled
[ ] 3D Sunken
Font: 5x7 Font

Justification
○ XXX . X___ [✓ ● __XXX . X__] ○ ___XXX . X

Edit/Write
[ ] Enabled Minimum -1000000000.000000C Maximum 1000000000.0000000

Display Properties
Display Style : Classic Style

**IMO Precision Controls Ltd**
Unit 3, The Interchange, Frobisher Way
Hatfield, Hertfordshire AL10 9TG UK

Tel: +44 (0)1707 414 444 Email: sales@imopc.com
Fax: +44 (0)1707 414 445 Web: www.imopc.com

*Bespoke technical training courses available at our Hatfield training facility. Call 01707 414 444 for more information*