Simple Example To Test AKD BASIC Modbus TCP Communications (Part 2: using KVB) Rev. A 9/25/2019

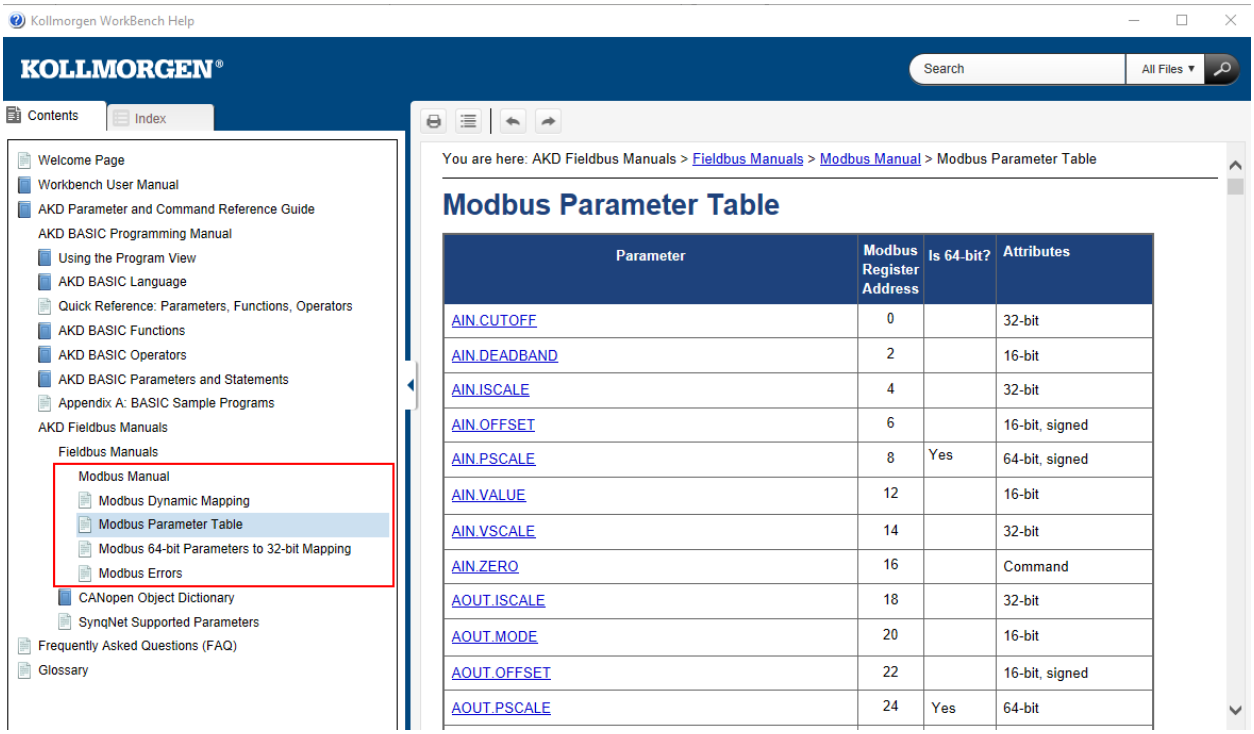
This application note will demonstrate how to prove you can read and write to the AKD BASIC variables over Modbus TCP. In Part1 of this series we demonstrated communications using a software based Modbus TCP Master called Modbus Poll.

In part 2 we will accomplish the same objectives using Kollmorgen Visualization Builder software. If you've already created the program in part 1 it can be used to start part 2 otherwise the development is repeated here.

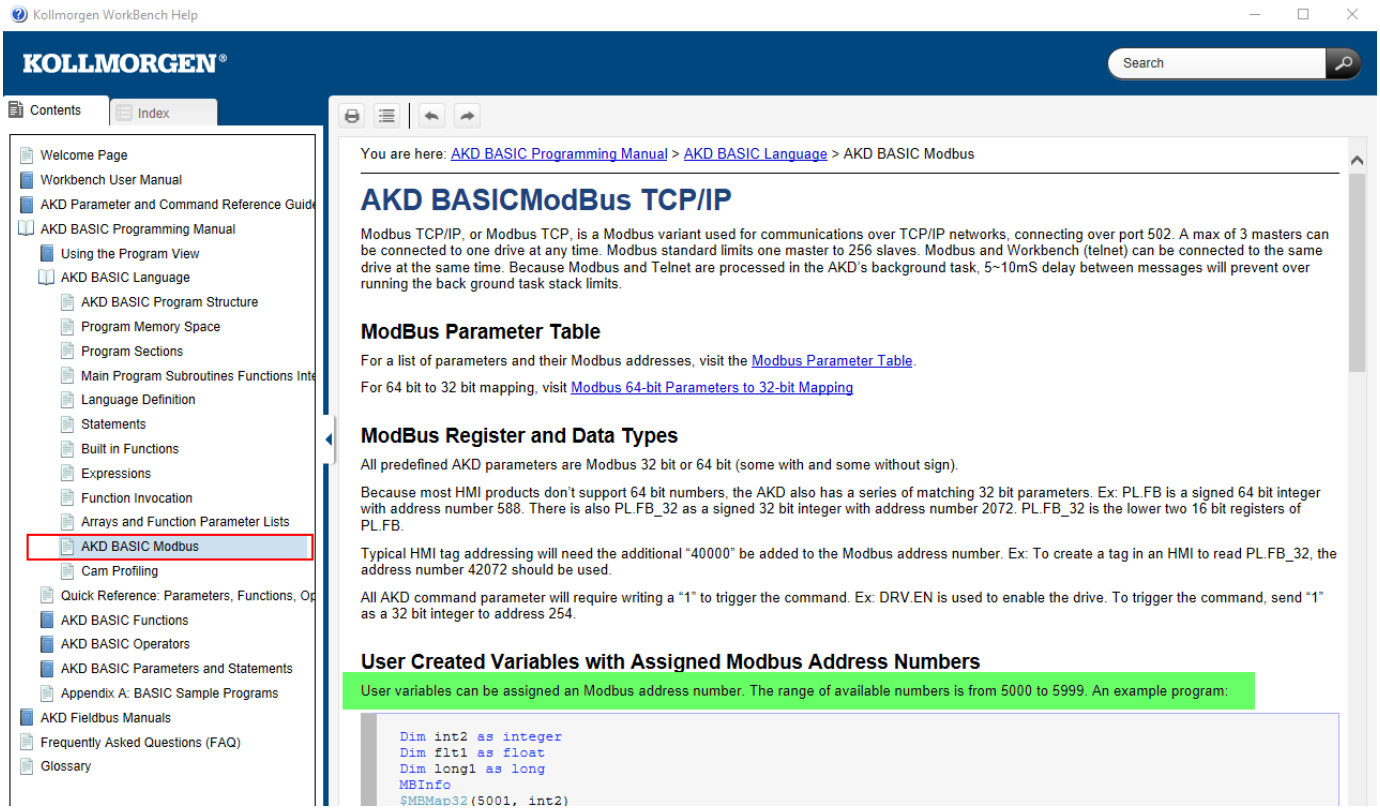
There are several ranges of Modbus TCP Parameters (and addresses) in the AKD-B, -P, and -T drives

(note the AKD-P-NBPN, Profinet drive does not support Modbus TCP).

The Modbus Manual is embedded in Workbench Help. See each individual parameter descriptions to determine if the parameter is supported by the AKD-T (AKD BASIC) or not.



In addition to the standard parameter table and the Modbus 64bit Parameters to 32bit Parameters there are also registers that can be mapped to variables in the AKD BASIC Program so a HMI or PLC can read/write values from/to the AKD BASIC program. In this case there is a specific section of Workbench under AKD BASIC Proramming Manual->AKD BASIC Language->AKD BASIC Modbus. Per below the available address range is from 5000 to 5999 (each 16 bit integer registers).



Note in order for the 5000 range (AKD BASIC variables Modbus) registers to be valid and not produce Modbus errors on attempt to access the AKD BASIC program must run and on execution declare variables (and possibly initialize them to default values) and map them to specific Modbus registers (in the 5000 range).

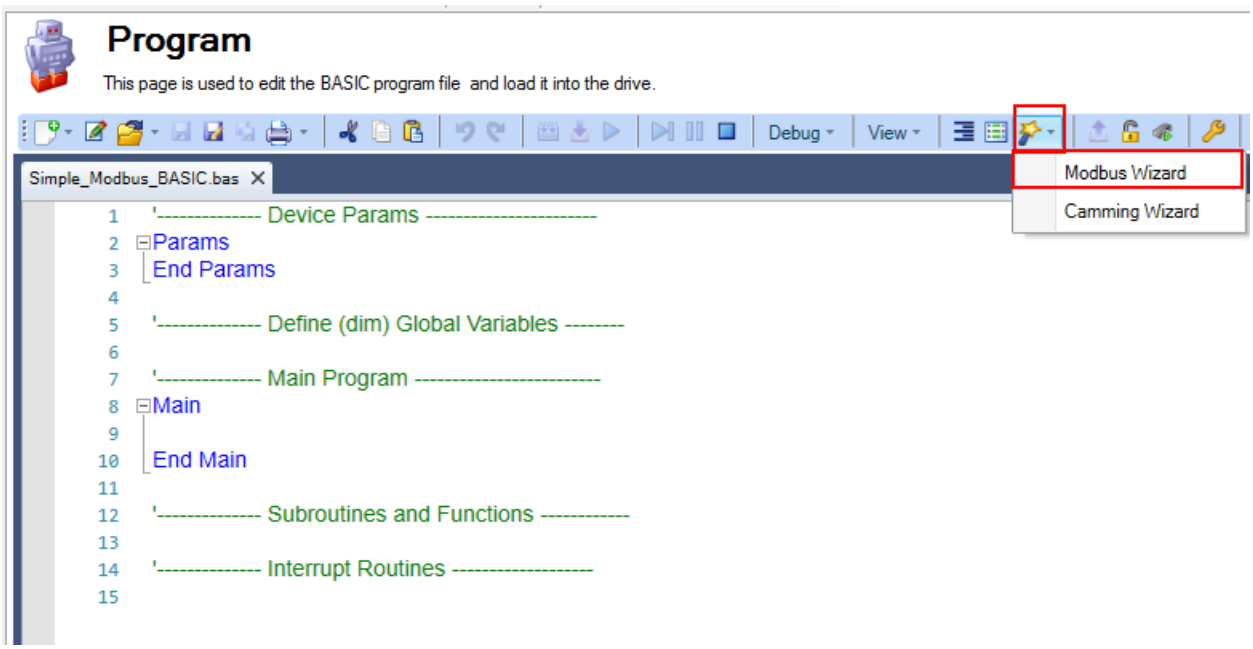
For the first example we will read the value of a variable in the AKD BASIC program to confirm we have Modbus communications.

Create a New Program in the AKD BASIC drive once online with Workbench.

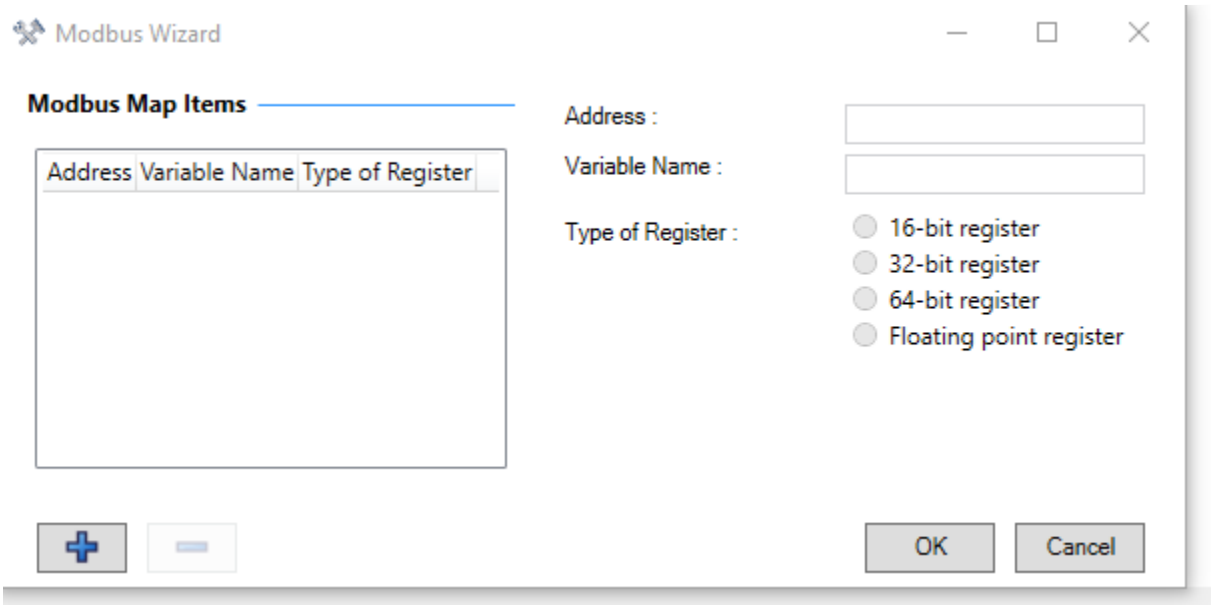
Next create a new variable called “var1”.

```
Simple_Modbus_BASIC.bas*
1   '-------------- Device Params -----------------------
2   Params
3     End Params
4
5   '-------------- Define (dim) Global Variables --------
6   Dim var1 as integer
7   '-------------- Main Program ------------------------
8   Main
9
10    End Main
11
12  '-------------- Subroutines and Functions ------------
13
14  '-------------- Interrupt Routines -------------------
15
```

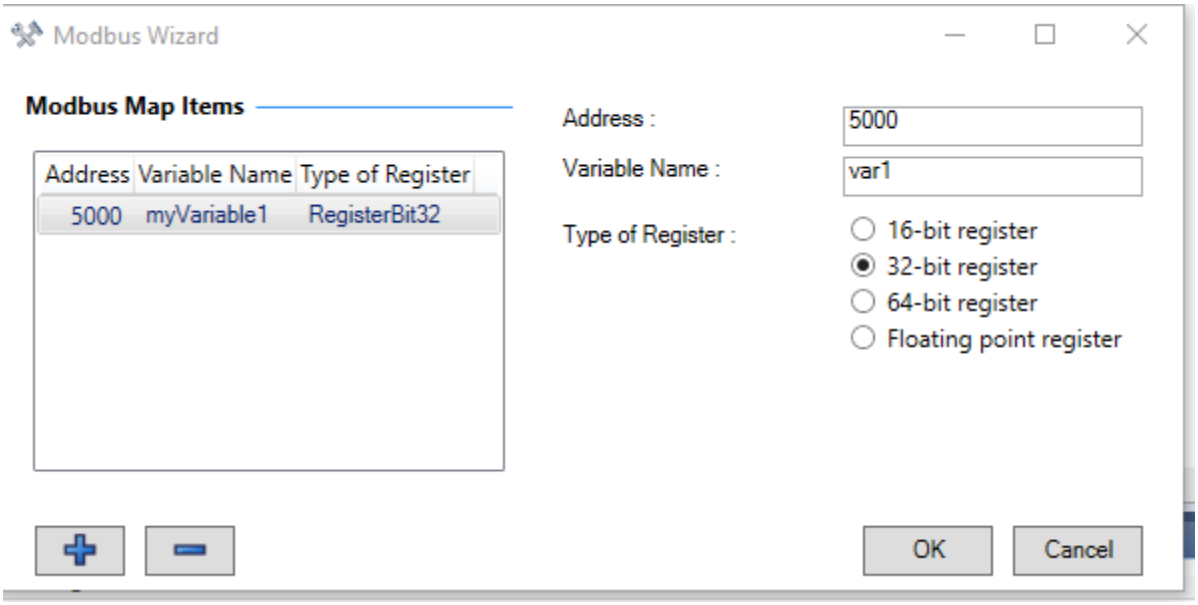
You’ll note there is a wand with a star on it in the toolbar this is the Wizards icon. One of the choices is “Modbus Wizard”.



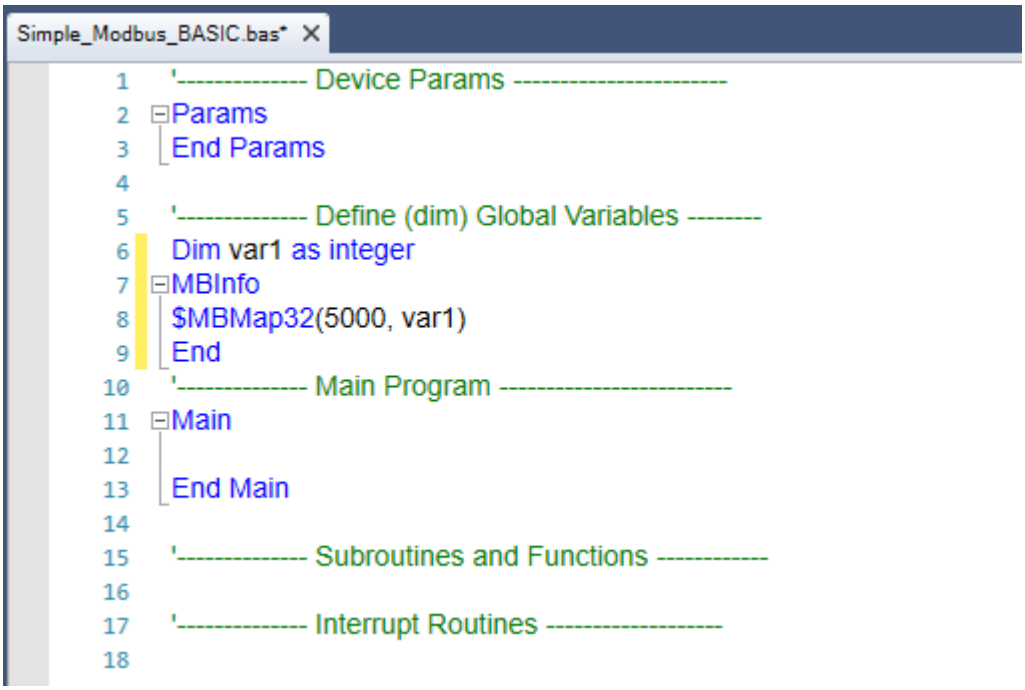
Selecting the Modbus Wizard calls up the following window that allows you to map a Modbus address to the variable we just created.



Click on the + button in the bottom left side of the Modbus Wizard window. It will add a variable starting at 5000. At this point it is possible to make changes to the setup for this first item in the list so I changed the variable name to “var1” and the type of register to “32-bit”. This means over Modbus the high word will be 5000 and the low word is 5001 (2 consecutive 16 bit registers). It is possible to use dynamic mapping to do word swapping but often the Modbus TCP master can handle it if required. This is beyond the scope of this Quick Start. Click ok to accept the changes.



Note between Dim and Main the Modbus mapping is now declared MBINFO:End and inside the declaration is the mapping of the 32 bit variable, var 1, and the starting address 5000.

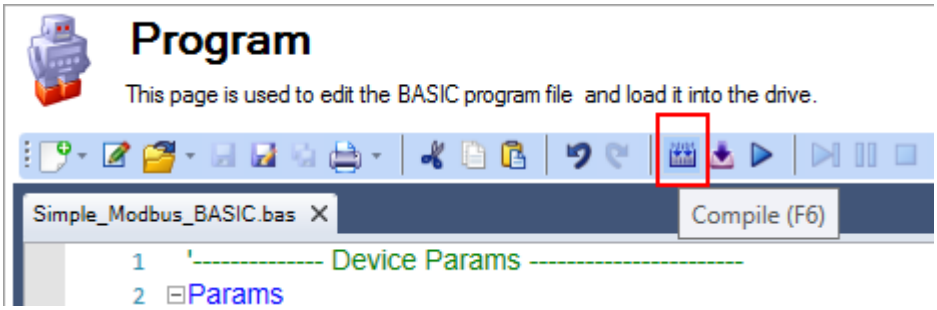


Next so the Modbus TCP Master can read a value (other than zero) we will initialize the var1 variable to a value sufficiently large enough to be contained in the high and low words. I also added a main loop using the While:Wend statement. The program now looks like this:

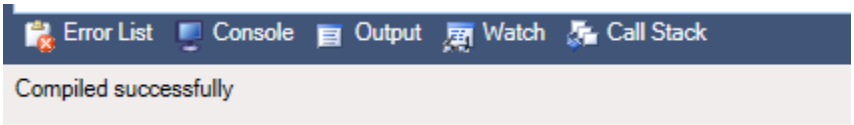
Simple_Modbus_BASIC.bas

```
1   '-------------- Device Params ----------------------
2   Params
3   End Params
4
5   '-------------- Define (dim) Global Variables --------
6   Dim var1 as integer
7   MBInfo
8   $MBMap32(5000, var1)
9   End
10  '-------------- Main Program ------------------------
11  Main
12  var1=1234567
13  While 1=1
14   'main loop
15  Wend
16  End Main
17
18  '-------------- Subroutines and Functions ------------
19
20  '-------------- Interrupt Routines -------------------
21
```

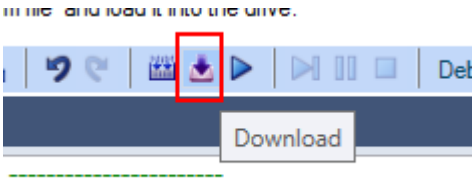
The program is now ready for the first test. In the toolbar click on the Compile Icon.



The bottom toolbar should indicate “Compiled successfully”.

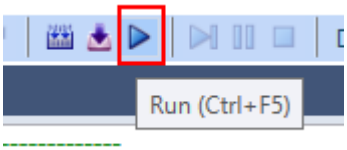


Next download the compiled program to the drive.



The status bar at the bottom of the Program screen in Workbench should indicated “Downloaded successfully” and the name of the program should be at the far right.

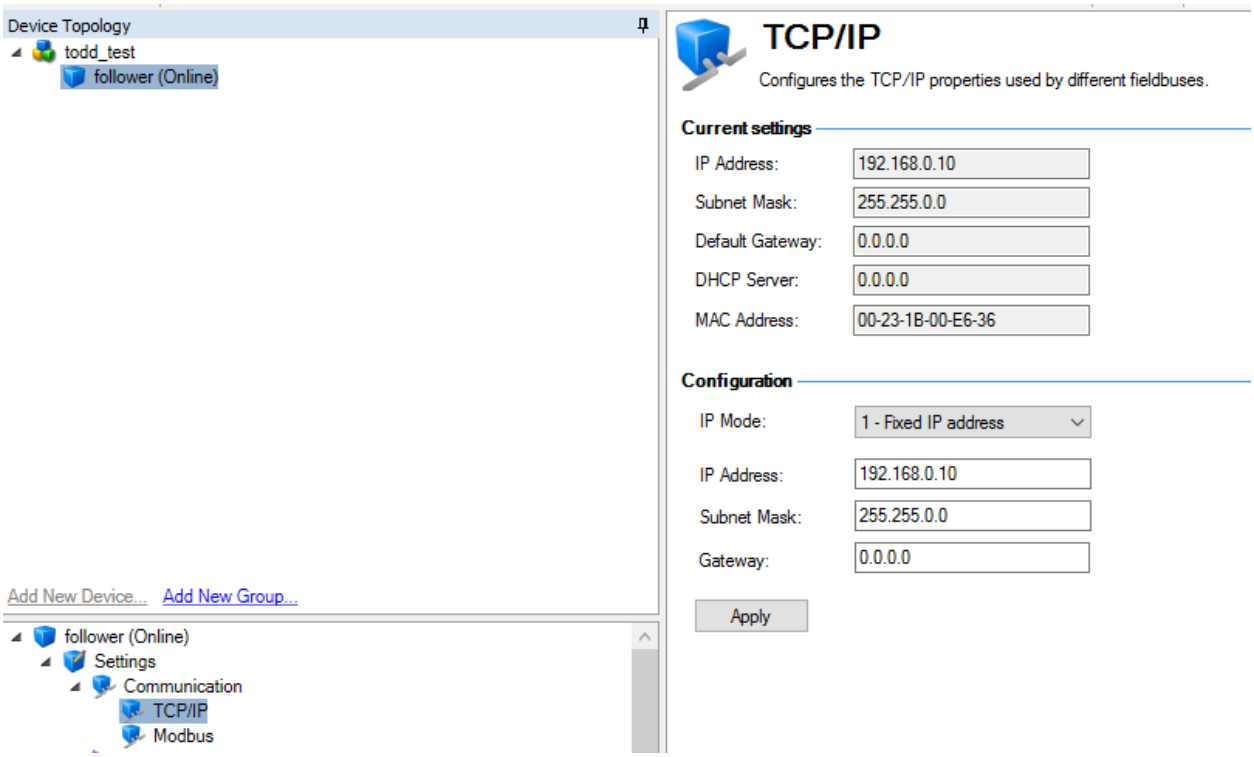
Finally run the program by pressing the Run icon in the toolbar.



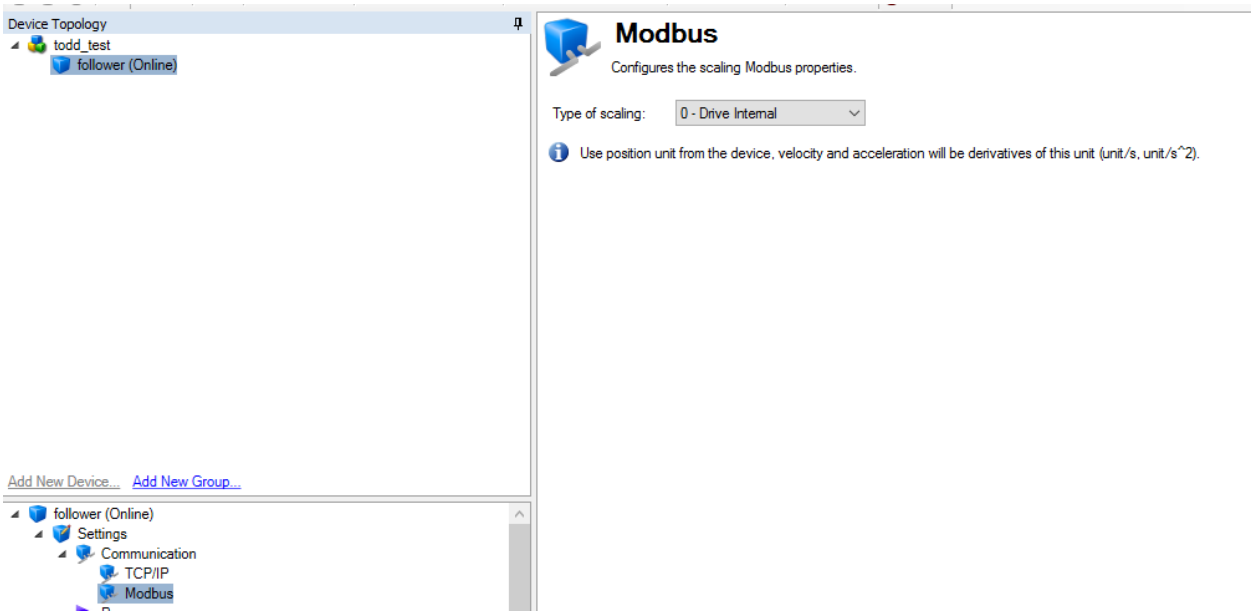
The status bar should show Running and a scanning bar to the left of the Program name should also indicate the program is running.

At this point the only thing the program does is map the var1 parameter to Modbus TCP addresses and initializes it to a value.

In my AKD BASIC drive the following IP Address was set:



I also did not want to use additional Modbus scaling in the AKD drive so under Communications->Modbus I set the Type of Scaling to “0-Drive Internal”.

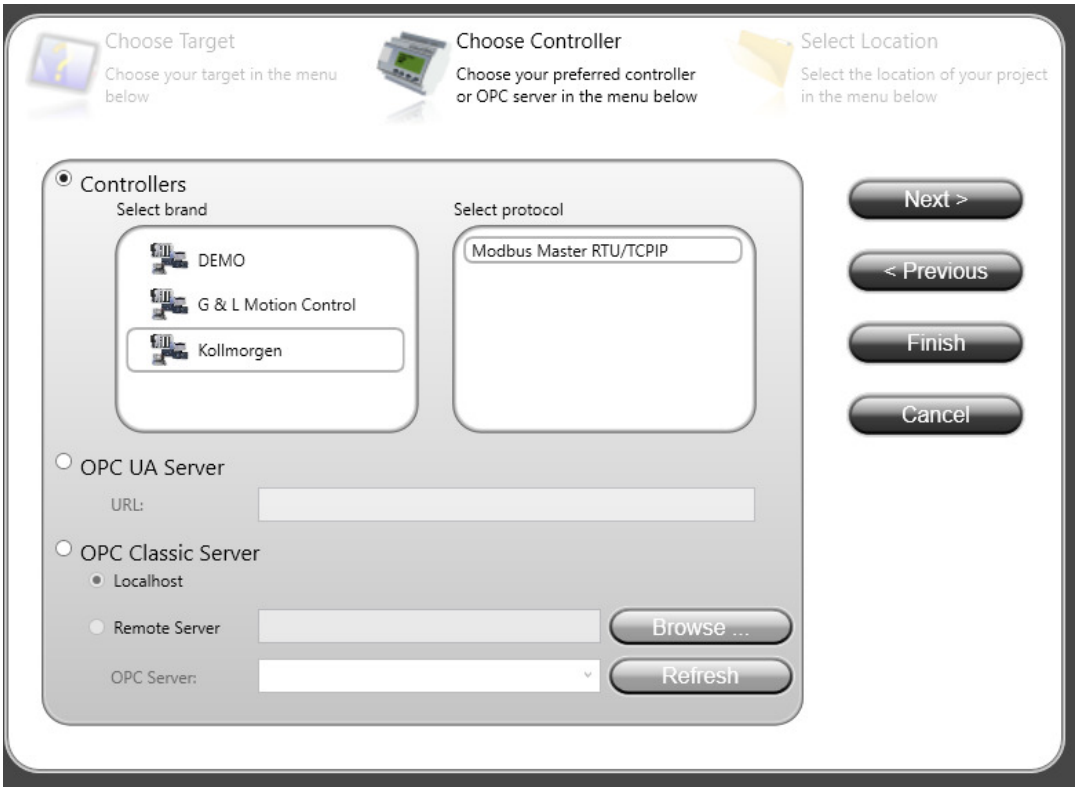


We are now ready to read the value using KVB. I am using v2.4 at the time of this application note. After opening the KVB application on your PC the following screen appears. For this quick start select “Create New Project”.

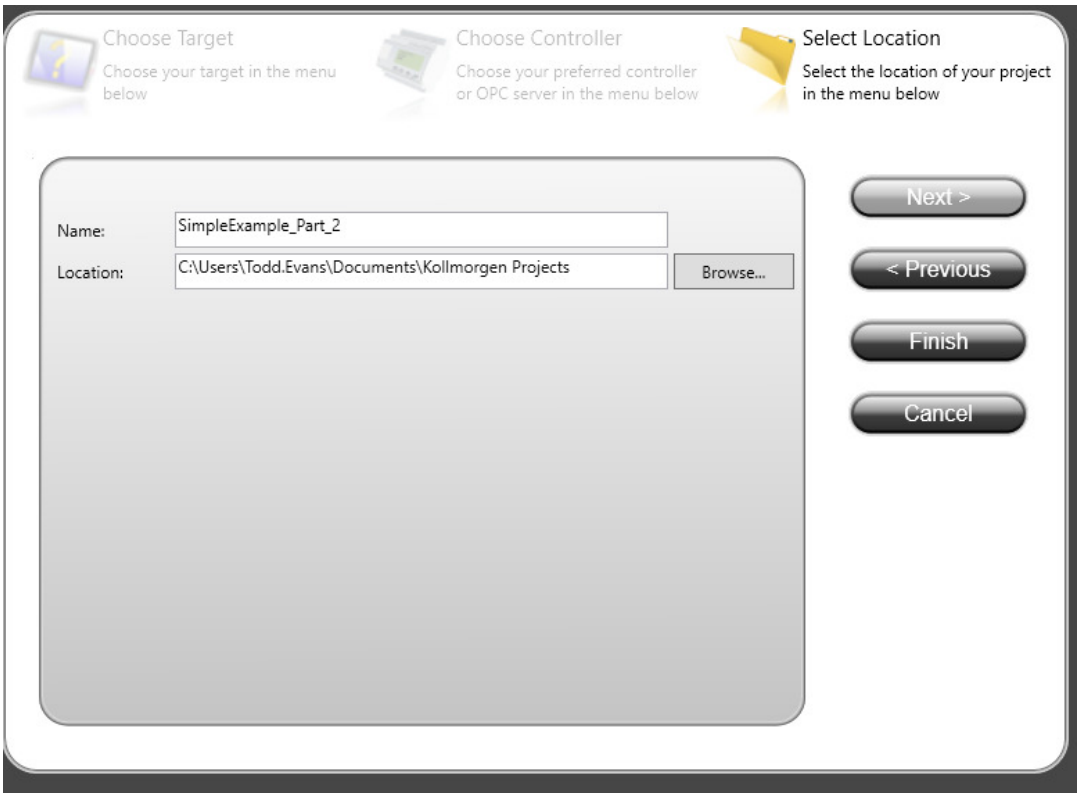
Next choose the appropriate AKI model. For the quick start and demonstration I am not going to use the AKI hardware and simply simulate on the monitor of my PC. See the KVB manual for procedures for

setting the AKI's IP address, etc. I arbitrarily choose at 7 inch operator panel.

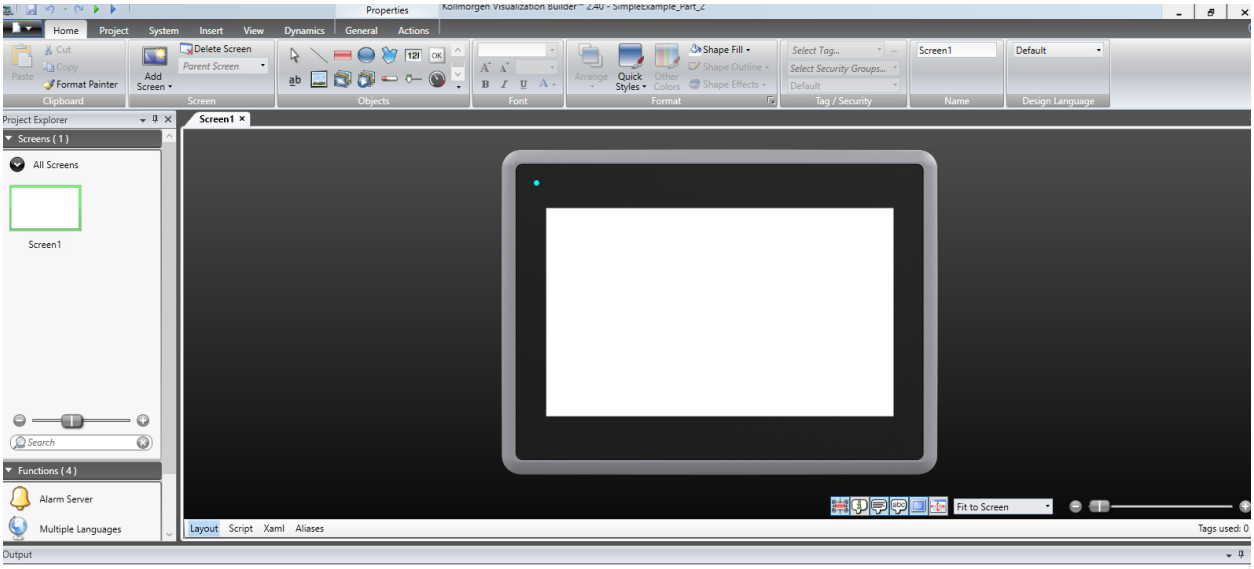
On clicking the Next button you will be given the option to select the Controller. In this case we want Kollmorgen and the Modbus TCP protocol.



Click Next and give the project a name and a path to where you want the file(s) to be placed. Then click Finish

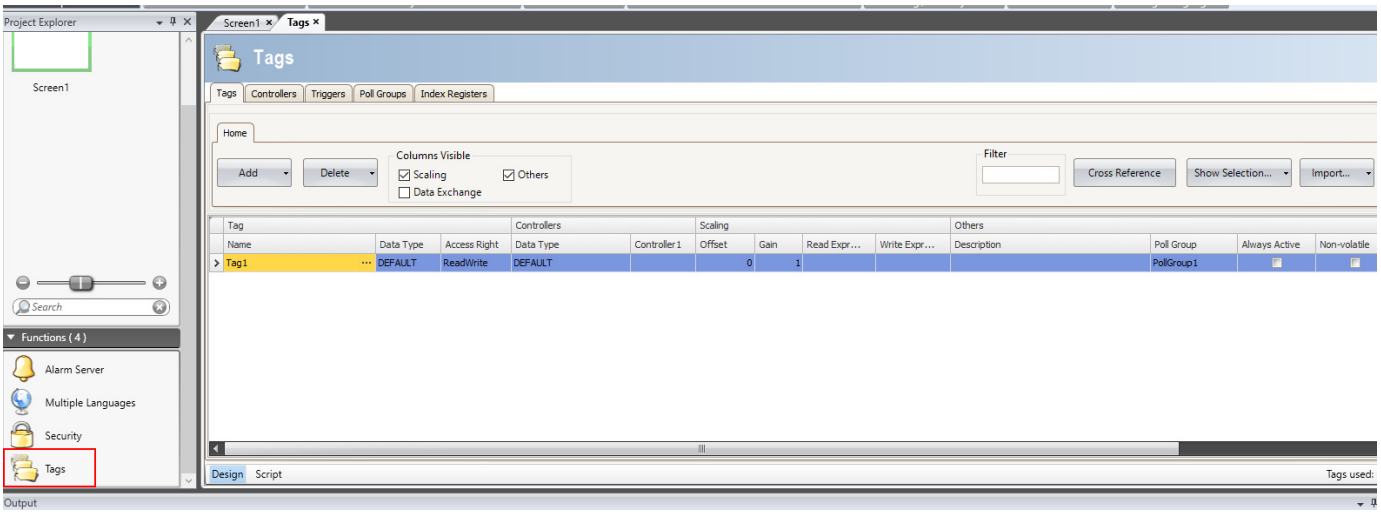


The project will now be created and you will have the development environment opened at this point:

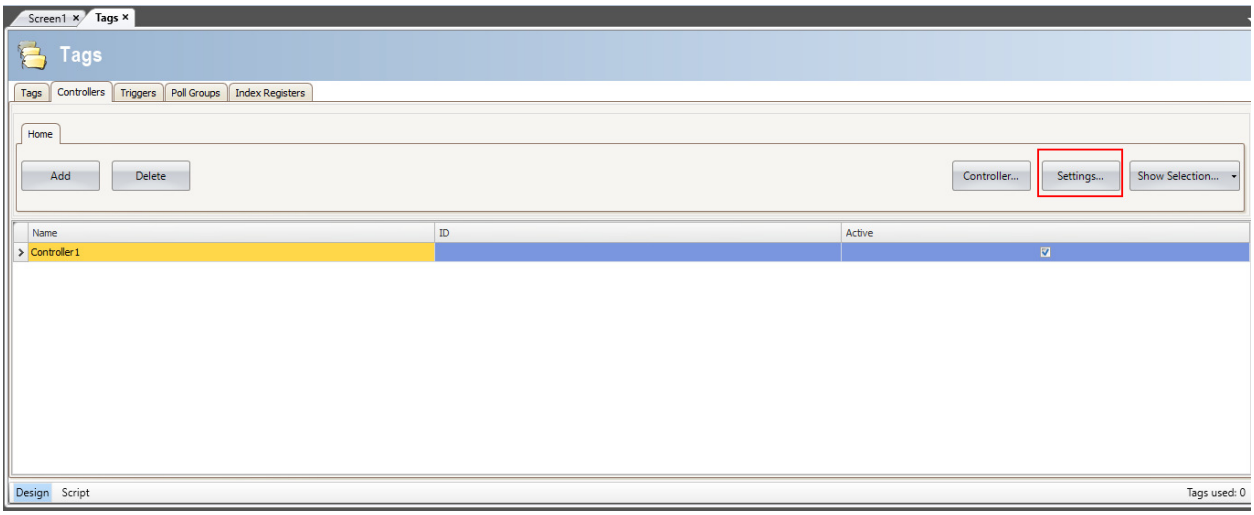


The first thing that must be done is to configure the communications so KVB knows the Modbus TCP slave's IP address.

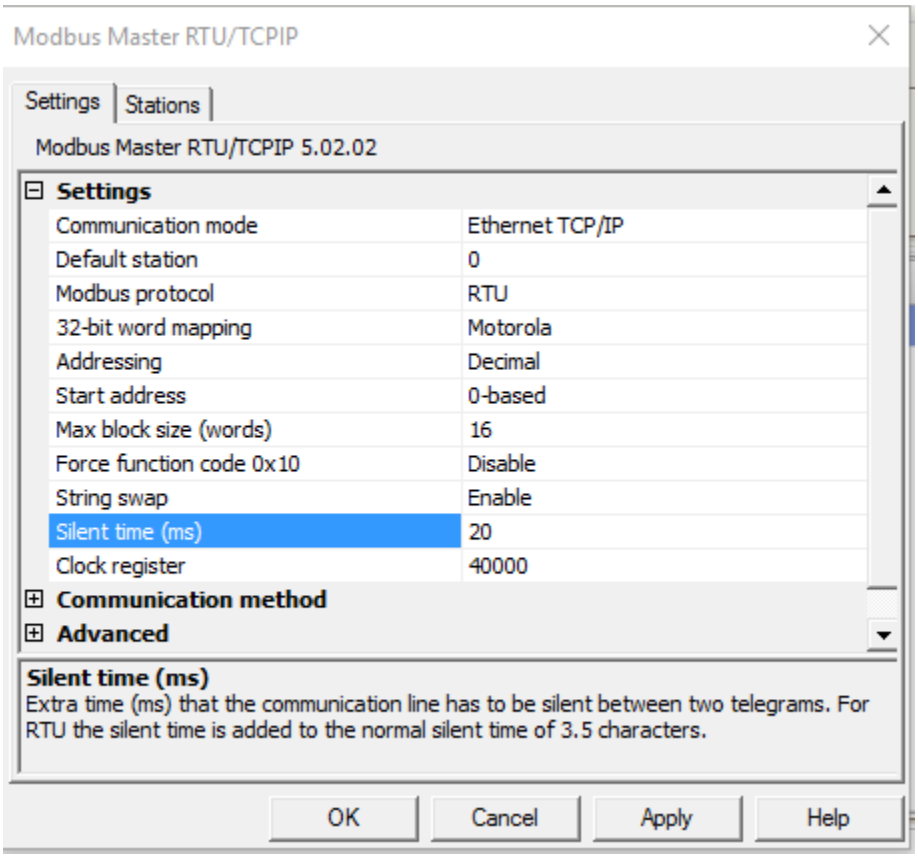
Use the scroll bar on the left in the Project Explorer to scroll down and then click on Tags. The Tags window will appear and there are tabs Tags, Controllers, Triggers, Poll Groups, and Index Registers. Click on the "Controllers" tab.



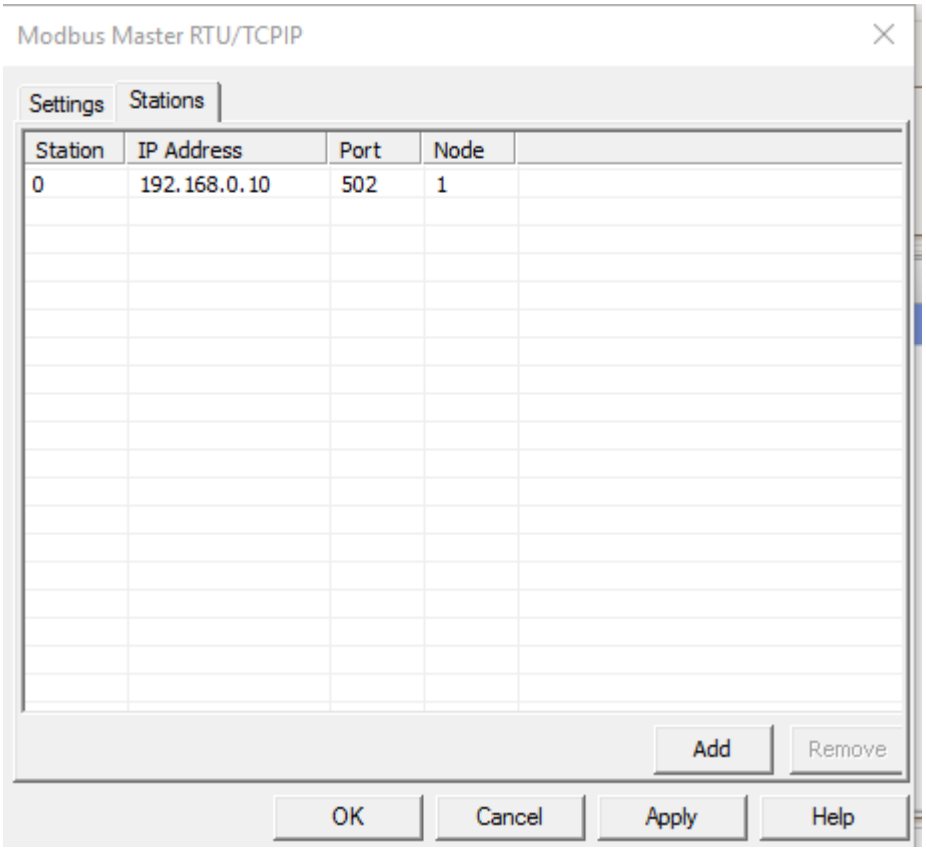
Click on the "Settings" button:



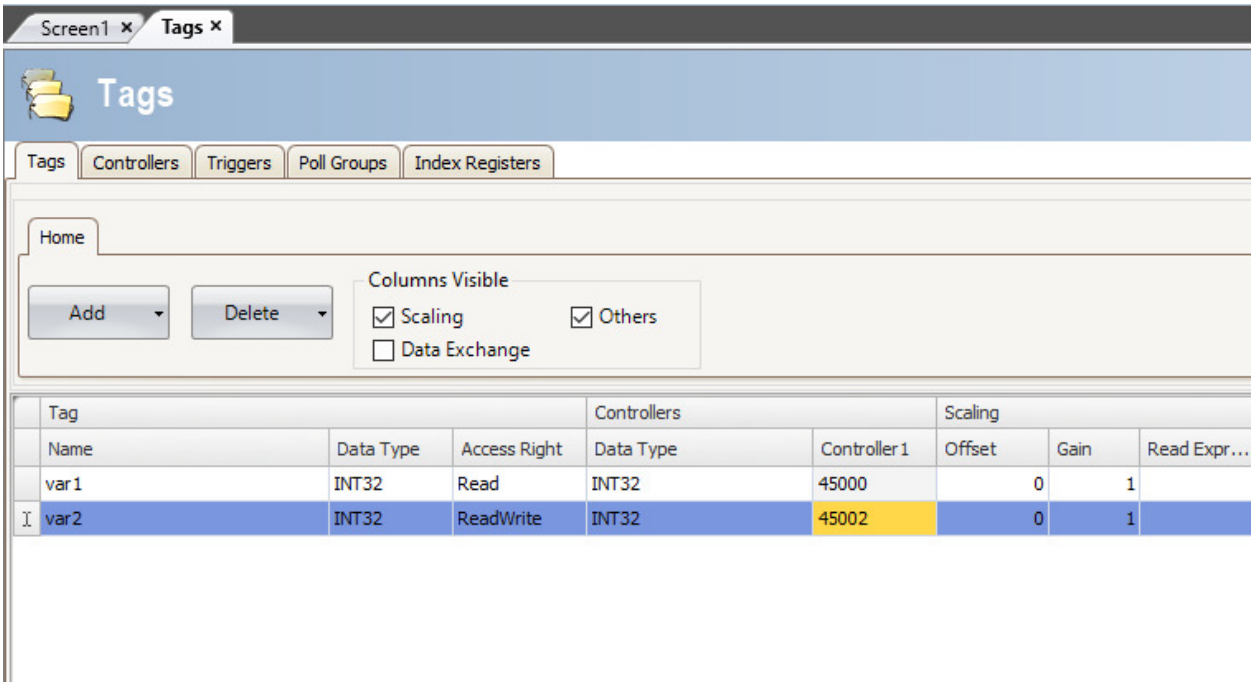
To help with communication traffic I changed “Silent Time(ms)” on the Settings tab to 20 msec. Note the default is Ethernet TCP/IP communication mode. Next click on the Stations tab.



On the Stations tab I changed the IP Address to the same IP Address as the target AKD BASIC drive. Then I clicked “Apply” and then “OK”.

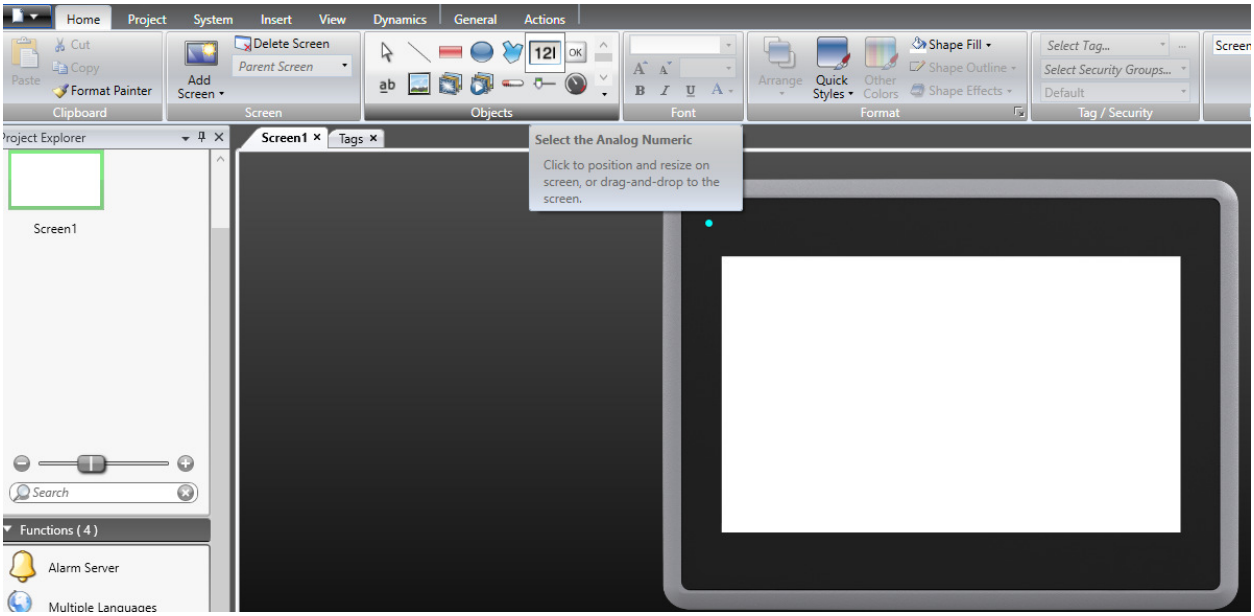


Click on the Tags tab and add the following tags. Note the tag names are the same as the variable names declared in the AKD BASIC program and the Tag Data Type was configured for INT32 as well as the "Controllers" Data Type in the Tag chart below. Under Controller 1 it is necessary to declare the Modbus TCP address for each variable (tag). Also take note that although the starting addresses in the AKD BASIC were 5000 and 5002 respectively KVB requires that you use 40000 + AKD Modbus Address so in this example 40000 + 5000= 45000 and 40000 + 5002=45002.

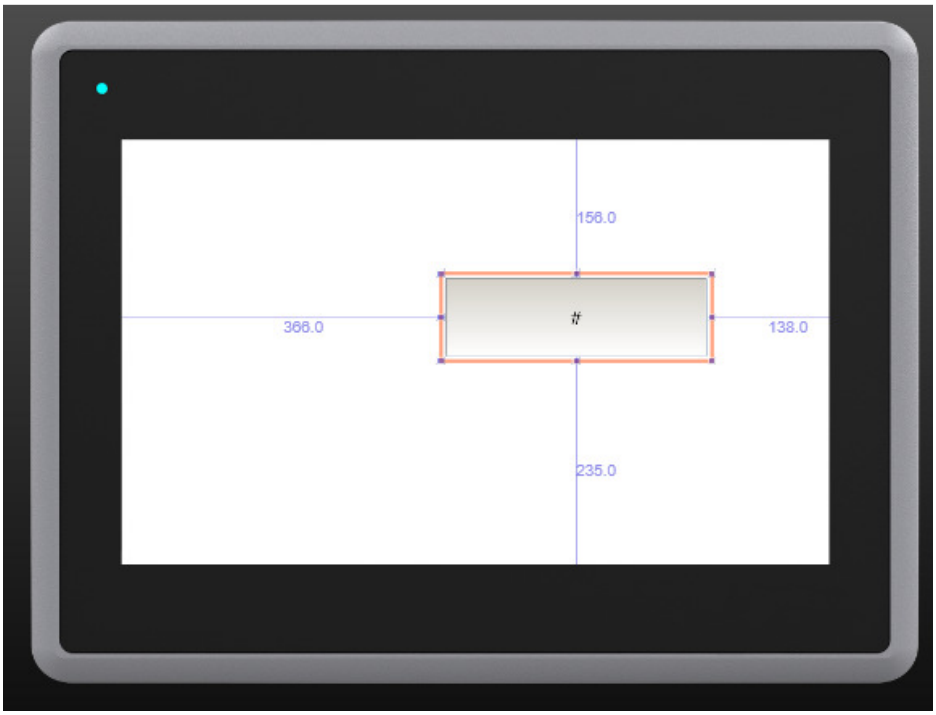


Before concerning ourselves with writing the value we want to prove the project can communicate and simply read var1 from the AKD BASIC program.

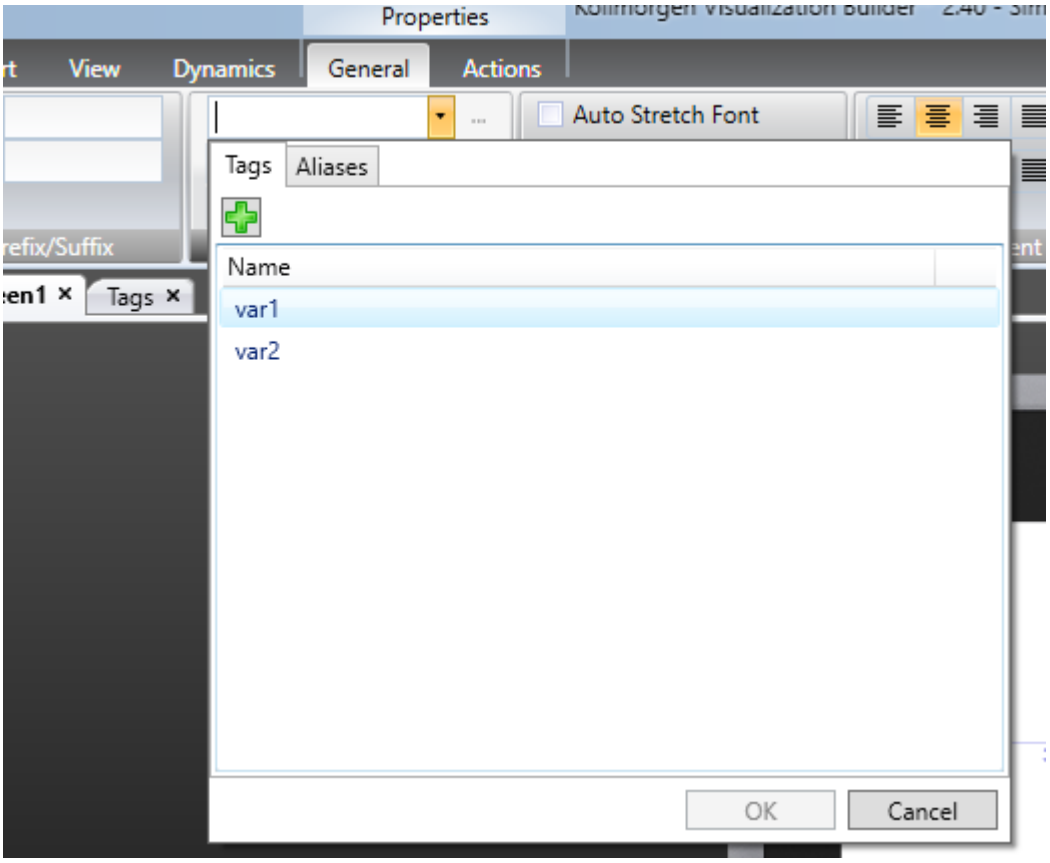
Click on Screen1 and select the Analog Numeric button from the toolbar.

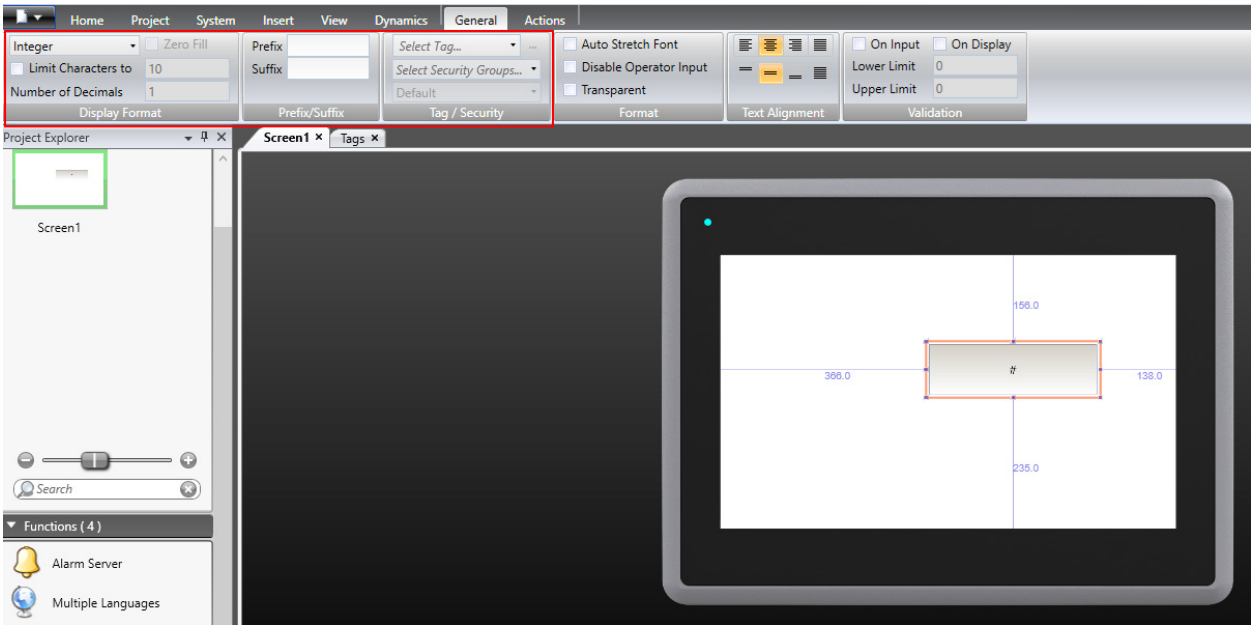


On selection click on the screen's workspace and use the crosshairs and your mouse to add the data field.

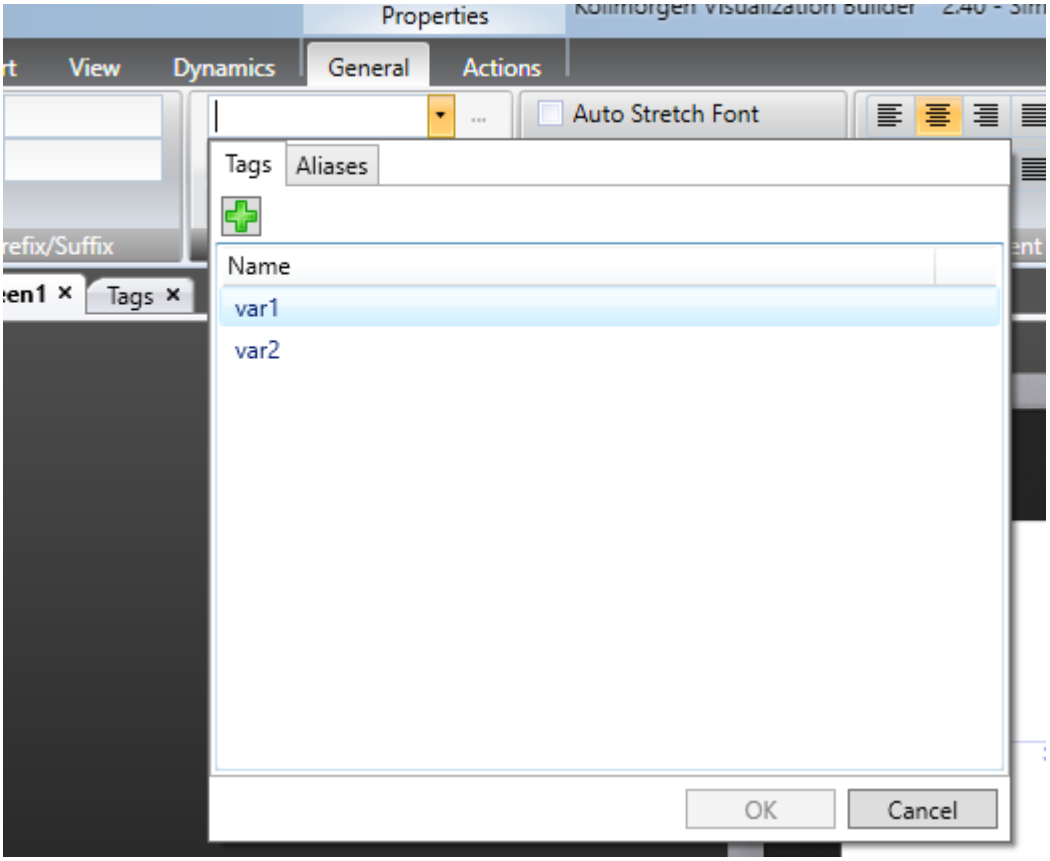


With the data field selected as above click on the General pull-down menu which will show the default format is Integer. There is a “Select Tag” list box you can click on to list the tags declared in the Tags Table.

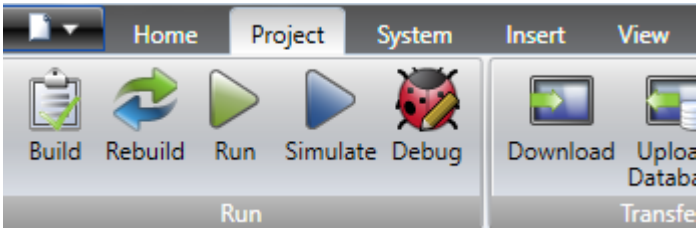




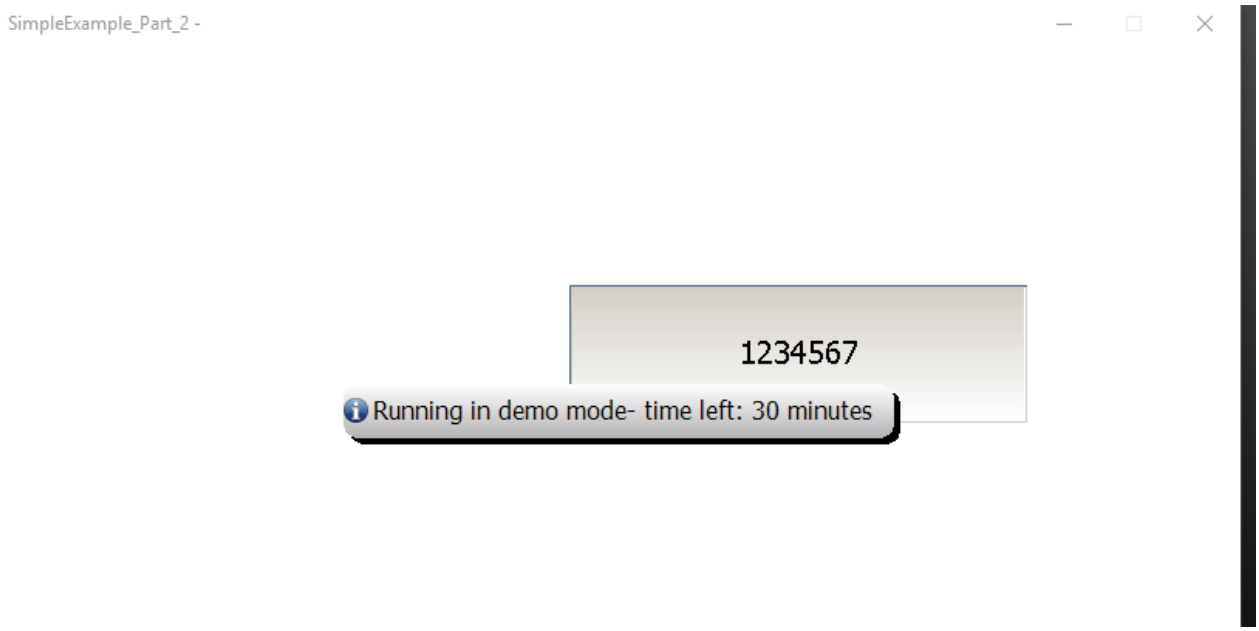
We want to select var1.



We're ready to Build and Run the Project. First selected the Project menu and clicked on Build (there should be no errors after building in the Output window at the bottom of KVB. Next I clicked on Run.



Because I am running the project on my PC instead of downloading it to an actual KVB unit it states “Running in demo mode- time left X minutes”. Note the value of var1 is 1234567 which is what it was initialized to in the AKD BASIC program.

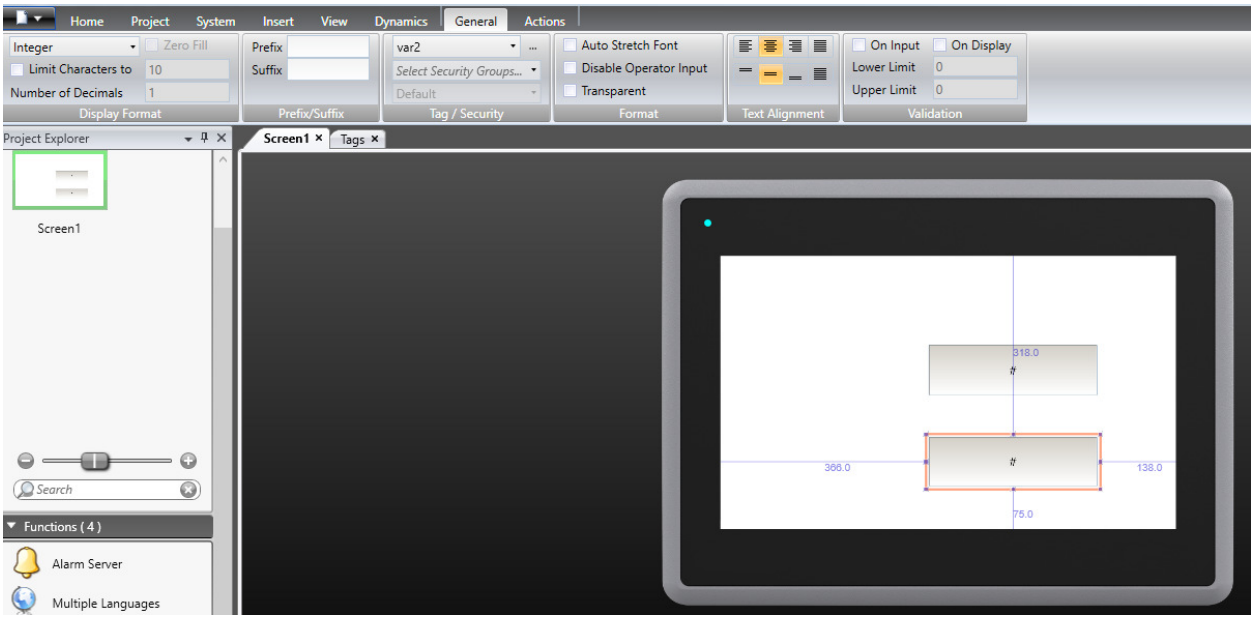


Click the X (close window) in the simulated screen to end the simulation.

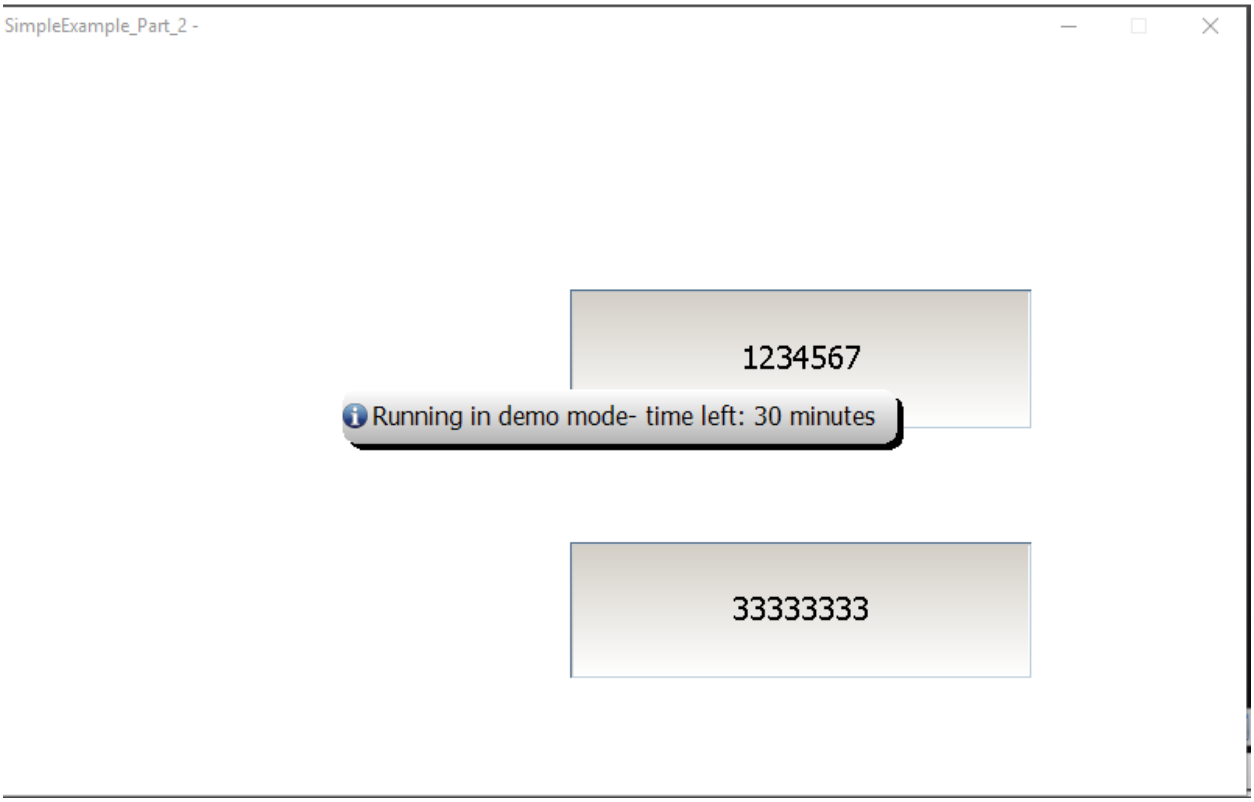
Next we will add the ability to enter a value from the screen and write to var2 (and see it change in the AKD BASIC program).

I simply added another Analog Numeric field on the screen and configured it for the tag var2.

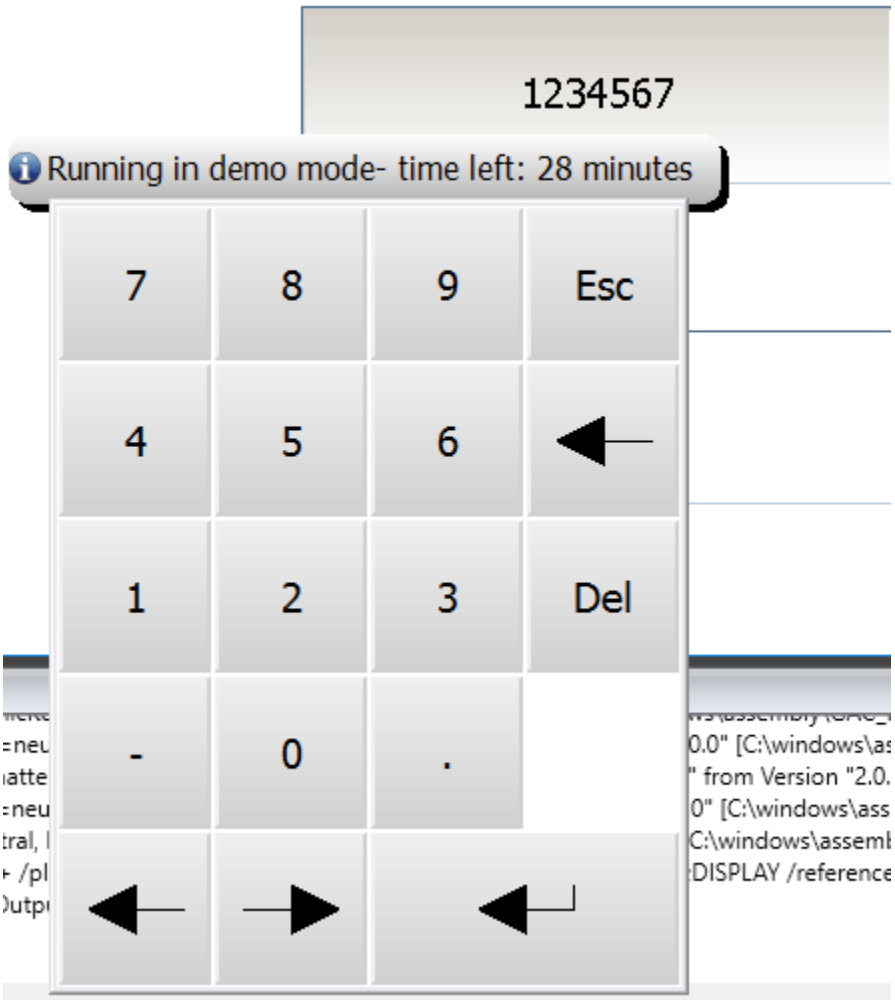
I then selected the Project Menu and then Build and Run as before.

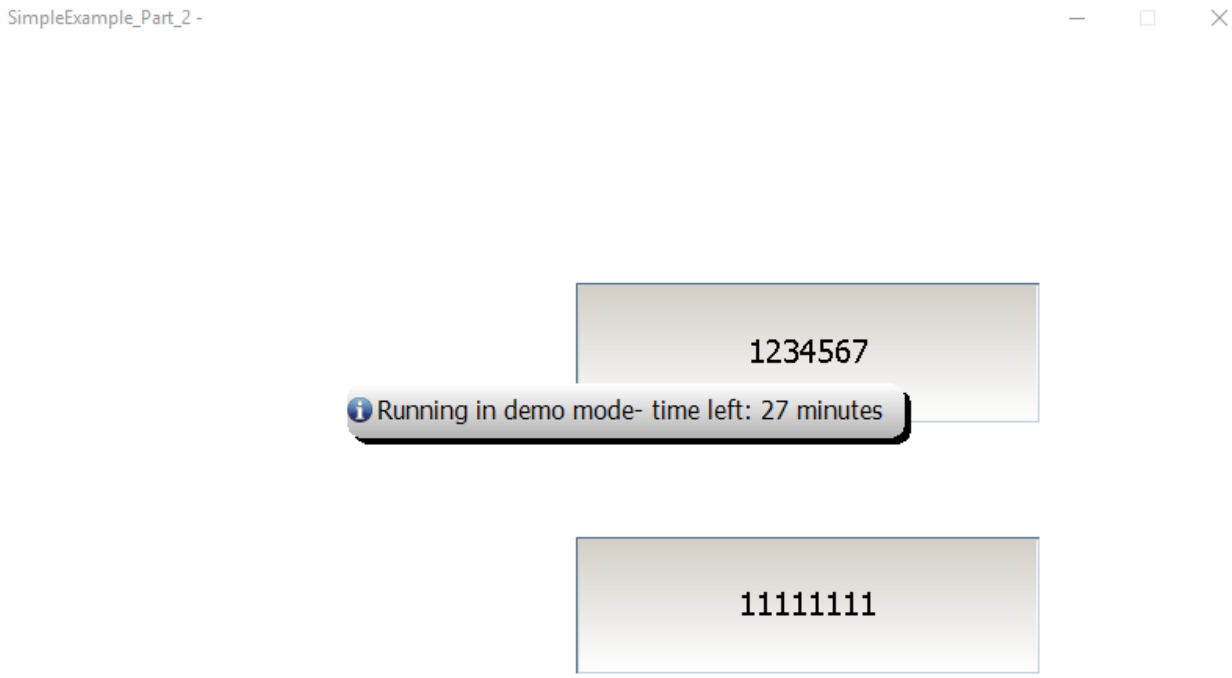


On simulation you'll note the value of var2 is still 33333333 since I left it in the same state as Part1.

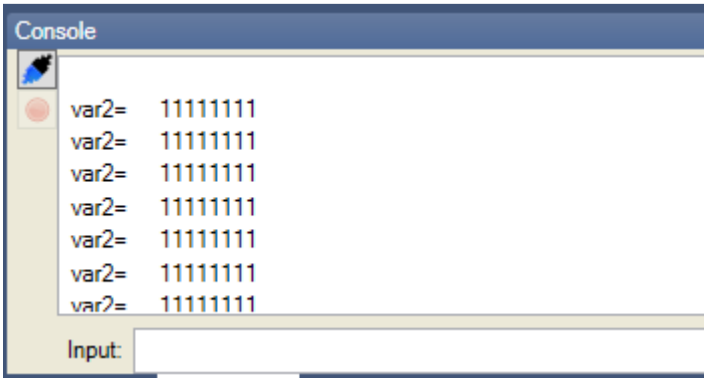


To change the value I click on the field which pops up a Keypad. I changed the value to 11111111.





Checking back at the Console on the Program screen of Workbench the value indeed changed.



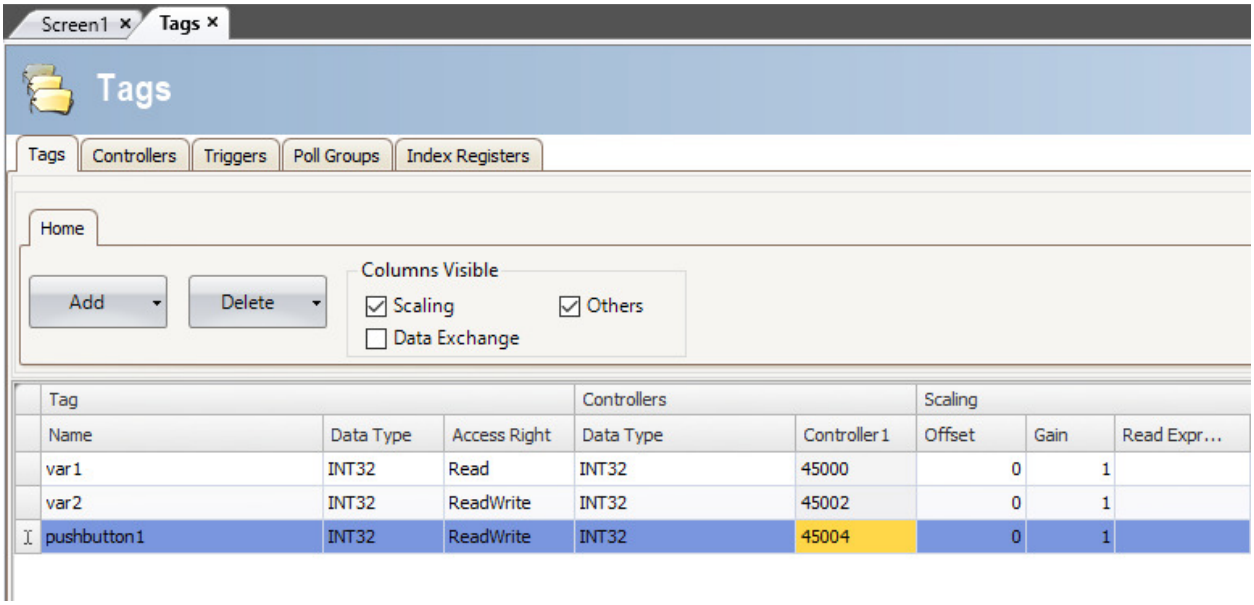
It is possible to use buttons to trigger values as well. To demonstrate I made the following additions to the program. I declared a variable called pushbutton1, mapped it to the next starting Modbus address, commented out the first print statement so we can focus on the button for now and added a new print statement to monitor the variable pushbutton1.

Simple_Modbus_BASIC.bas

```
1   '-------------- Device Params -----------------------
2   Params...
4
5   '-------------- Define (dim) Global Variables --------
6   Dim var1,var2, pushbutton1 as integer
7   MBInfo
8   $MBMap32(5000, var1)
9   $MBMap32(5002, var2)
10  $MBMap32(5004, pushbutton1 )
11  End
12  '-------------- Main Program ------------------------
13  Main
14  var1=1234567
15  While 1=1
16      'Print "var2= ",var2
17      Print "Pushbutton1= ",pushbutton1
18  Wend
19  End Main
20
21  '-------------- Subroutines and Functions ------------
```

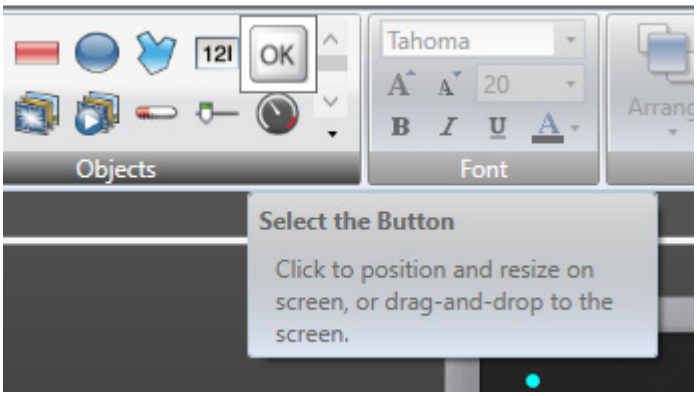
I Saved, Compiled, and Downloaded the program to the AKD BASIC.

Moving back to the KVB software and the Tags table, I added a new tag named it pushbutton1, set the data type to INT32 ReadWrite and set the Controllers Data Type to INT32 and set the address to 45004.



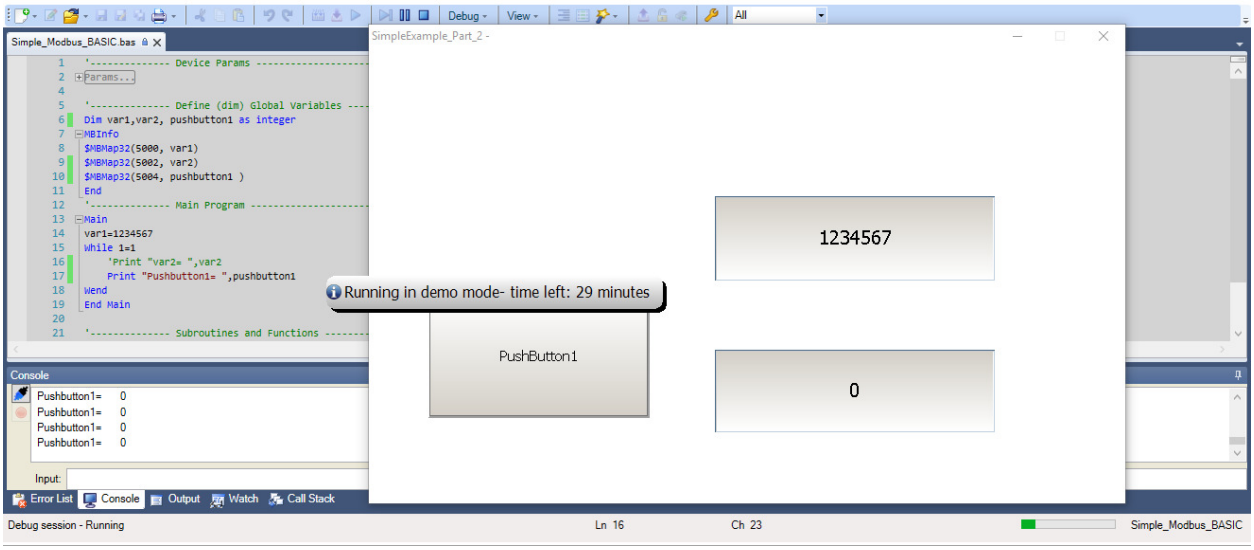
Tag			Controllers		Scaling		
Name	Data Type	Access Right	Data Type	Controller 1	Offset	Gain	Read Expr...
var1	INT32	Read	INT32	45000	0	1	
var2	INT32	ReadWrite	INT32	45002	0	1	
pushbutton1	INT32	ReadWrite	INT32	45004	0	1	

From the toolbar in KVB I selected the button object.



Draw the button on Screen1. Select the Actions menu and in this case I set the Mouse Down and Mouse Up function to “Set Analog”, pointed the operation to the variable “pushbutton1” and set the value to 1 when the button (mouse) is pressed DOWN and a value of 0 when the button (mouse) is pressed UP (i.e. released). Note it is possible to on Click Set Analog to a constant value (i.e. 1). You will find this is the case in the sample AKI to AKD BASIC program. I encourage you to download the sample project from our website and open it up offline to explore the objects and their configurations.

Now we are ready to simulate. Build and Run as before. Note in the AKD BASIC console, the value is 0 without the Pushbutton on the screen being pressed.



While holding pushbutton on the screen down, the value in the program changes to 1.