ADVANCED MOTION CONTROLS

Click & Move®

C++ Function

Two Axis Motion Project

Project Description

In this lesson we will add C++ code in the form of a function block to a Click and Move project.

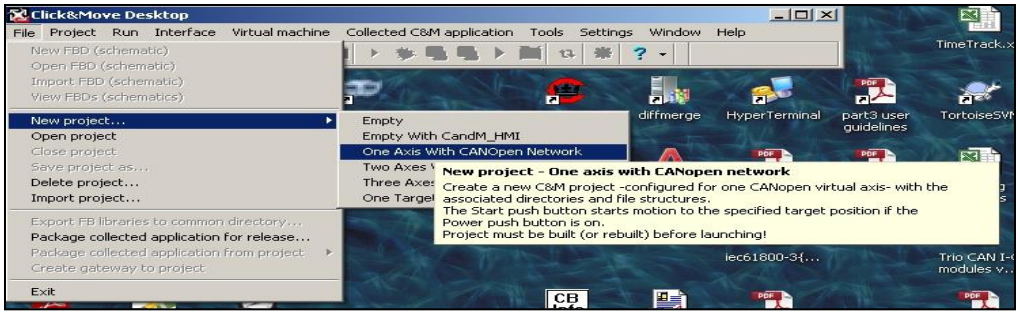
Starting with a new “One Axis with Can Open Network “ project we will add logic to generate and display a running average of the Actual Position of the motor. The complete project named “CPP_EXAMPLE” is available for review.

The Actual Position stream is sampled every 200 ms at which time the NEW_AGERAGE is calculated .

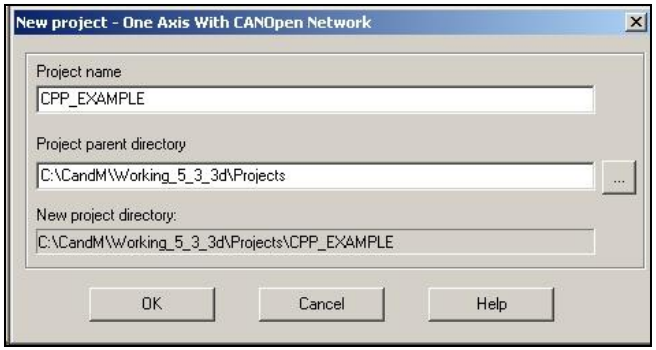
NEW_AVERAGE = (sampled value / WEIGHT) + (LAST_AVERAGE - (LAST_AVERAGE / WEIGHT))

Create New Project

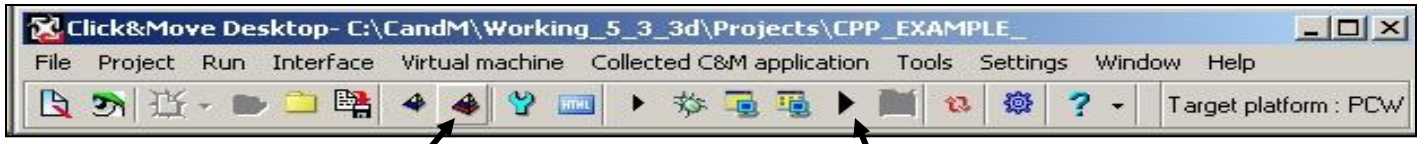
Start C&M and close the open project (if any).
Click “File” “New project” “One Axis With CanOpen Network”.



Save the project to the C&M “Projects” folder under the project name “CPP_EXAMPLE”



Build the new project



Start a full build of the project

Run the Project

(after build completes)

The build should complete without errors as shown in this Message Window.

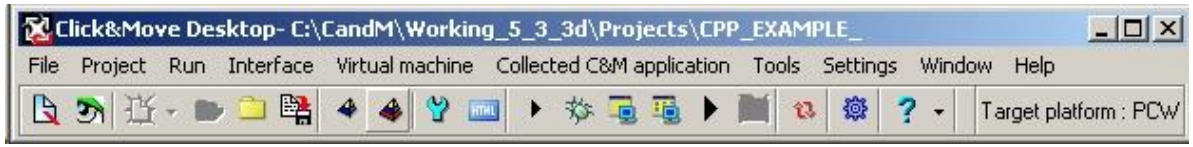
```
make.exe[1]: Nothing to be done for `Build'.
make.exe[1]: Leaving directory `C:/CandM/WOFDE5~1/Projects/CPP_EX~2'
make: Leaving directory `C:/PROGRA~1/A-M-C/CA8154~1/System/MAKEFI~1'
Project is successfully built.
```

We have just created a single axis CAN interface project with a virtual axis. No hardware is required to run this project making it perfect for our C++ project.

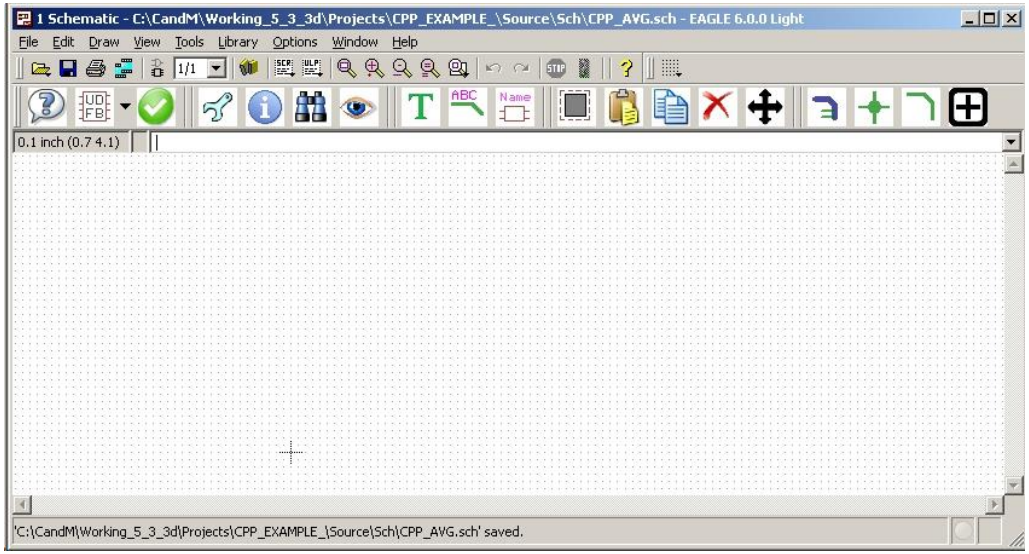
Stop the project and close the HMI and Virtual Axis before going on to the next step

Add Empty Function Block to the Project

On the Desktop click “File” “New FDB (schematic) to start a new schematic.

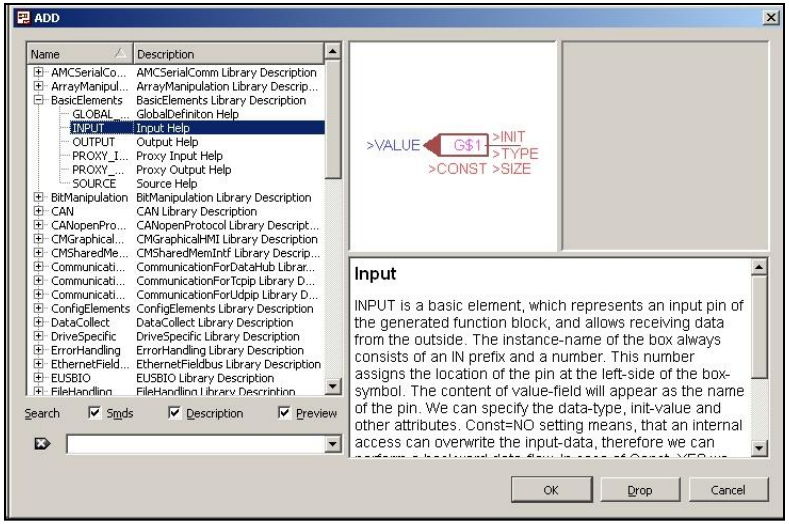


When the new schematic opens up click “File” “Save As” and save the file as “CPP_AVG”.



Add Inputs and Outputs to the Schematic

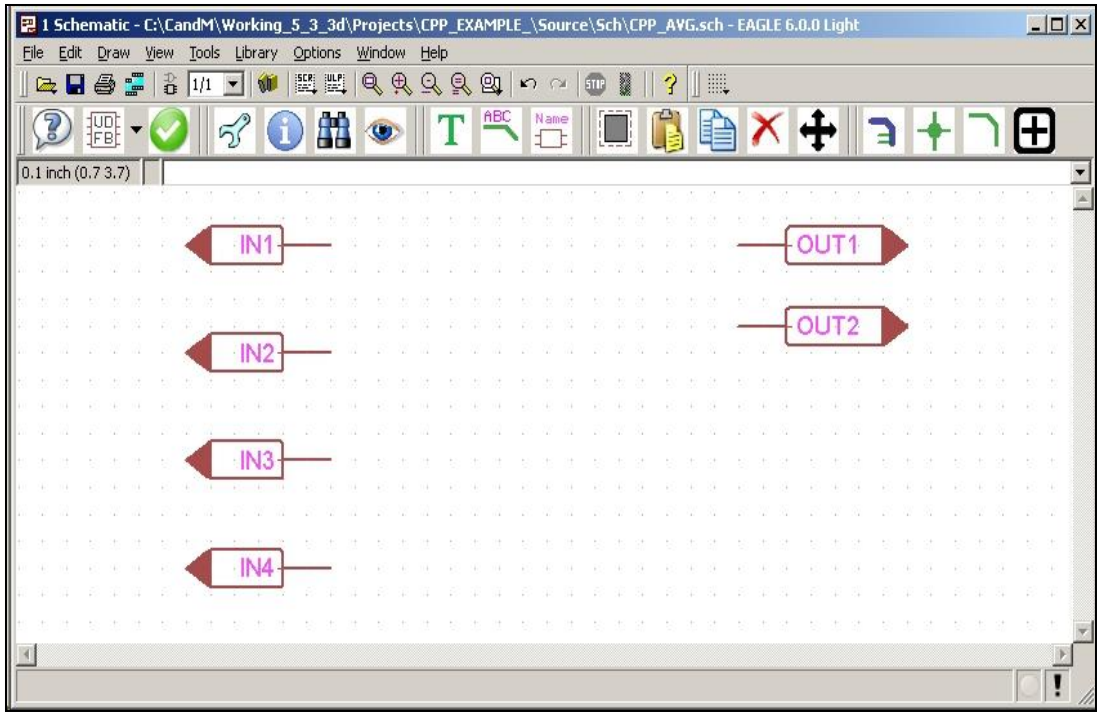
Click “Edit” “Add” or click the add icon  to open the library browser.



Click “BasicElements” “INPUT” “OK” and drop 4 inputs onto the schematic.
Press “Esc” then click “OUTPUT” and add 2 outputs to the schematic.
Press Esc twice to exit.

Add Inputs and Outputs to the Schematic

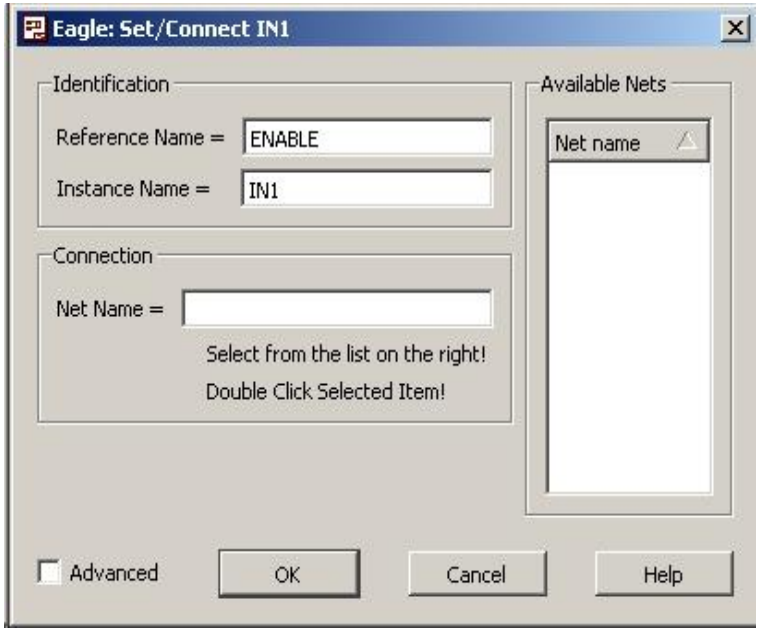
The schematic should look something like this:

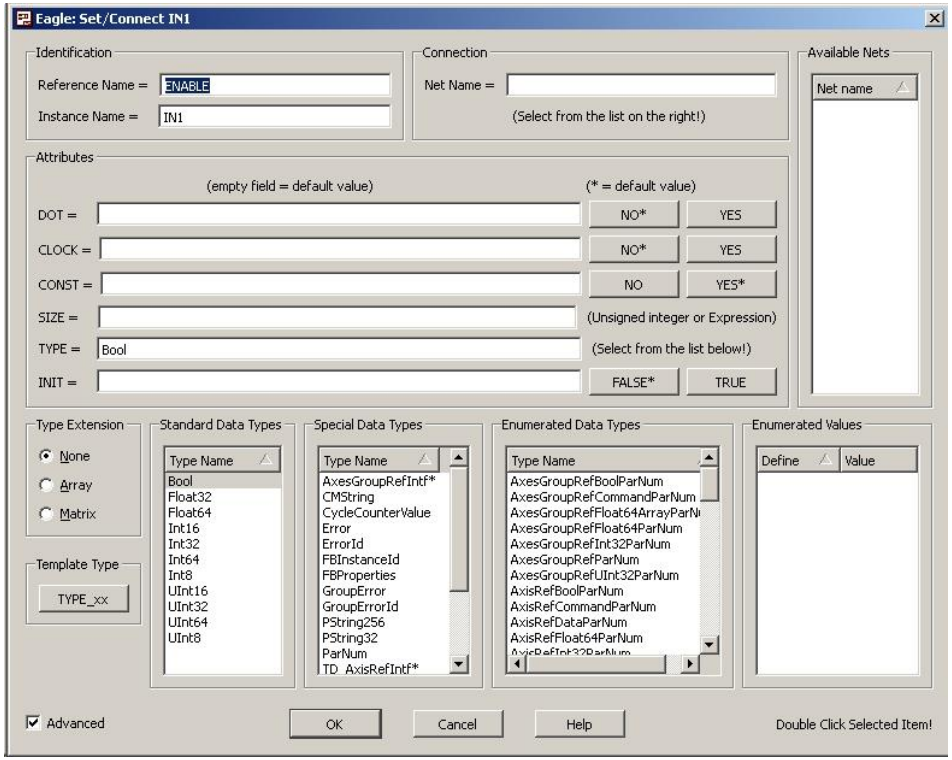


Now we assign a name and a type to each IO object.

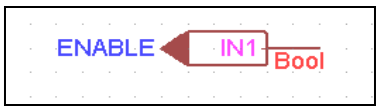
Add Inputs and Outputs to the Schematic

Right click the IN1 input, and choose CM Set/Connect from the pop up list. Enter “ENABLE” in the Reference Name box and then check the Advanced check box.





Double click “Bool” in the Standard Data Types Window. Click the “OK” button when done.



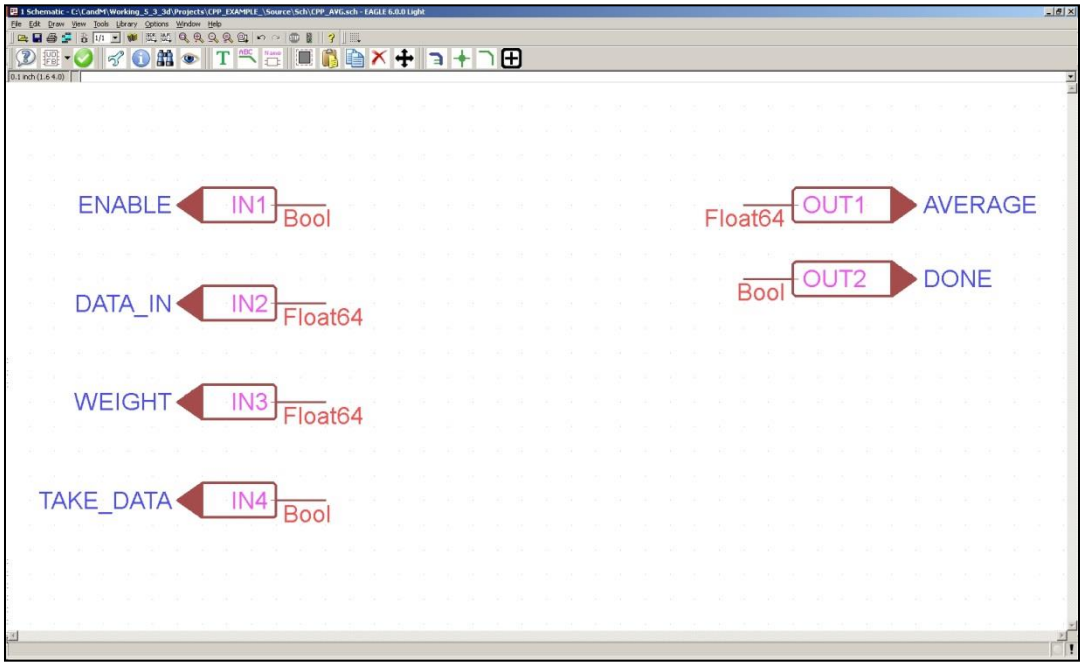
← Now Input 1 looks like this.

Add Inputs and Outputs to the Schematic

Now repeat the process to assign names and types to the other IO's.

IO	TYPE	NAME
IN1	Bool	ENABLE (done)
IN2	Float64	DATA_IN
IN3	Float64	WEIGHT
IN4	Bool	TAKE_DATA
OUT1	Float64	AVERAGE
OUT2	Bool	DONE

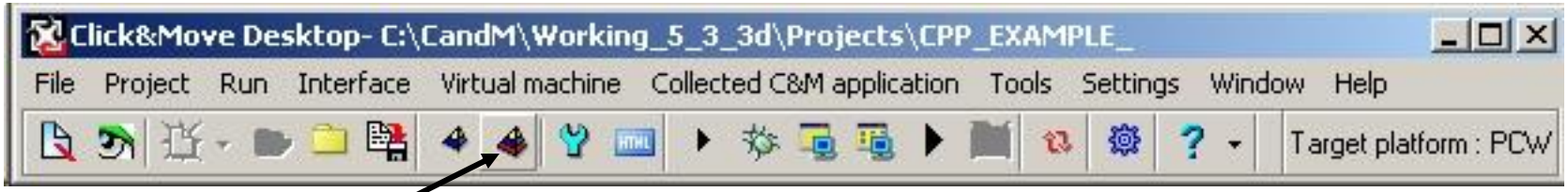
When finished, the schematic will look like this:



Add the new Schematic to the Library

Click the UDFB button on the tool bar. Choose “Create” to add our schematic to the library.

NOTE: Should we some day make changes to an existing function block that affect the inputs and
. outputs, we use the UDFB button and choose “Re-Create” to modify the existing information.



Now start a full build and the compiler will create skeleton c++ and dot h files for our new schematic.

Move the C++ files to the Source Folders

The compiler placed our project source files under the “Generated folder”. We must move them out from under the “Generated” folder.

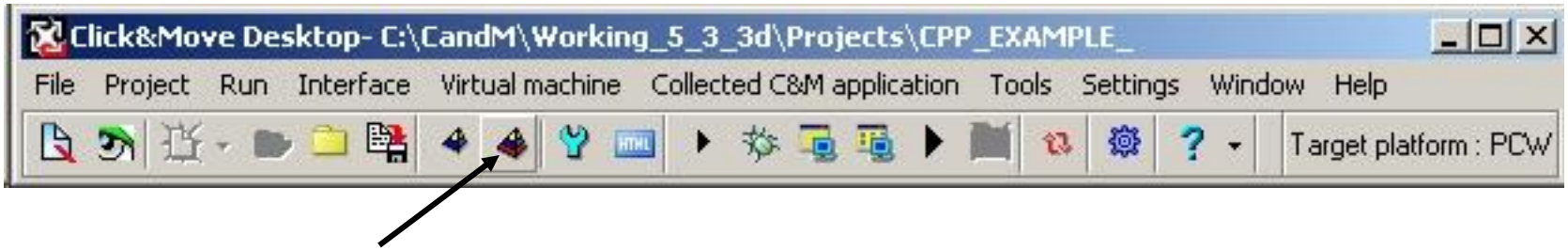
Use Windows File Explorer to move these two files “CppavgImpl.cpp” and “CppavgImpl.h”

From the folder “C:\CandM\Working_5_3_3d\Projects\CPP_EXAMPLE\Generated\Source\Cpp”

To the folder “C:\CandM\Working_5_3_3d\Projects\CPP_EXAMPLE\Source\Cpp”

In the particular case of this example, you will need to create the “\Cpp” folder before moving the files.

(The “\Working_n_n_n(m)\” part of the path will vary with the C&M version)



Now start another full build. Again it should complete with success.

We are now ready to begin adding C++ code to the source files.

Add C++ code to the dot H Source File

Use your favorite text editor to edit the “CppavgImpl.cpp” and “CppavgImpl.h” source files.
(I use Notepad++ and it colors the text)

Add the following lines of code to the CppavgImpl.h file after the line containing “private:”

Bool takeDataWas;
Bool enableWas;
Float64 Average;

When finished the text will look similar to this:

```
    void body(void);
  protected:
  private:
  Bool takeDataWas;
  Bool enableWas;
  Float64 Average;
    // Object Copy Prevention
    CppAvg(const CppAvg&);
    CppAvg& operator=(const CppAvg&);
};
```

Add C++ code to the dot Cpp Source File

Add the following lines of code to the CppavgImpl.cpp file.

Just before the first line containing an Open Brace “{“

,takeDataWas(0)
,enableWas(0)
,Average(0)

And add these lines of code between the Open and Close Brace “{ }” just after the line with “body()”:

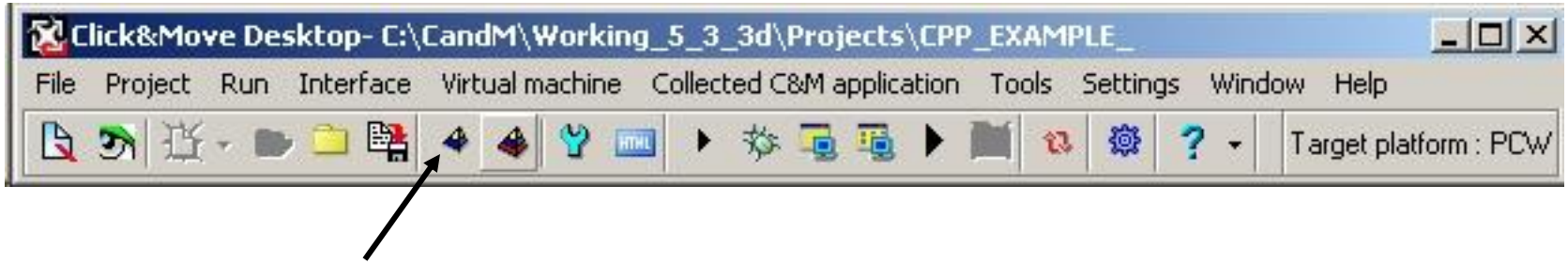
```
{
  if(*pin_Enable)
  {
    if(( *pin_TakeData) && (!takeDataWas))
    {
      src_Average=src_Average-(src_Average / *pin_Weight);
      src_Average=src_Average+(*pin_DataIn / *pin_Weight);
      takeDataWas=1;
    }
    else
    {
      takeDataWas=*pin_TakeData;
    }

  }
  else
  {
    src_Average=0;
    takeDataWas=0;
  }
}
```

Add C++ code to the dot Cpp Source File

After making the changes your source file will look something like this:

```
#ifndef       CppAvgImpl_h
    #include "CppAvgImpl.h"
#endif

using namespace CM;

ImplFBsCM::CppAvg::
CppAvg(Char const * instanceName, const IntfFBsCM::IName *pa
    CppAvgBase(instanceName, parent, nextSibling, systemData,
    ,takeDataWas(0)
    ,enableWas(0)
    ,Average(0)
{
}

ImplFBsCM::CppAvg::
~CppAvg()
{
}

void ImplFBsCM::CppAvg::
body()
{
   if(*pin_Enable)
   {
     if(( *pin_TakeData) && (!takeDataWas))
     {
       src_Average=src_Average-(src_Average / *pin_Weight);
       src_Average=src_Average+(*pin_DataIn / *pin_Weight);
       takeDataWas=1;
     }
     else
     {
       takeDataWas=*pin_TakeData;
     }

   }
   else
   {
     src_Average=0;
     takeDataWas=0;
   }
}
```

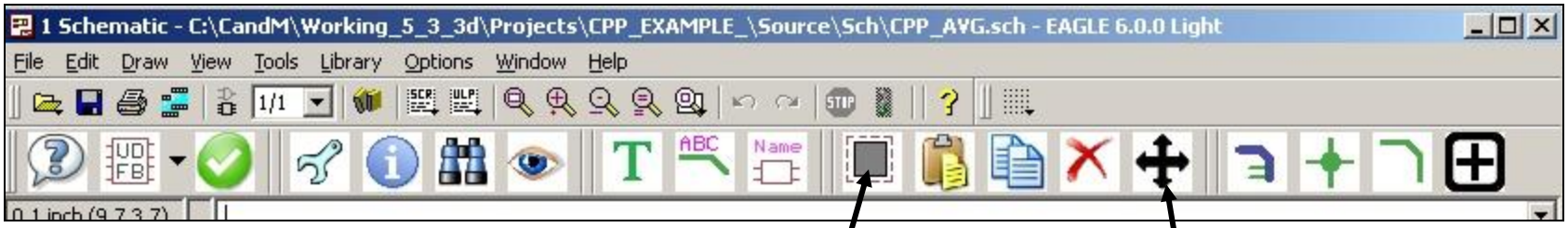
Build the project to verify the work



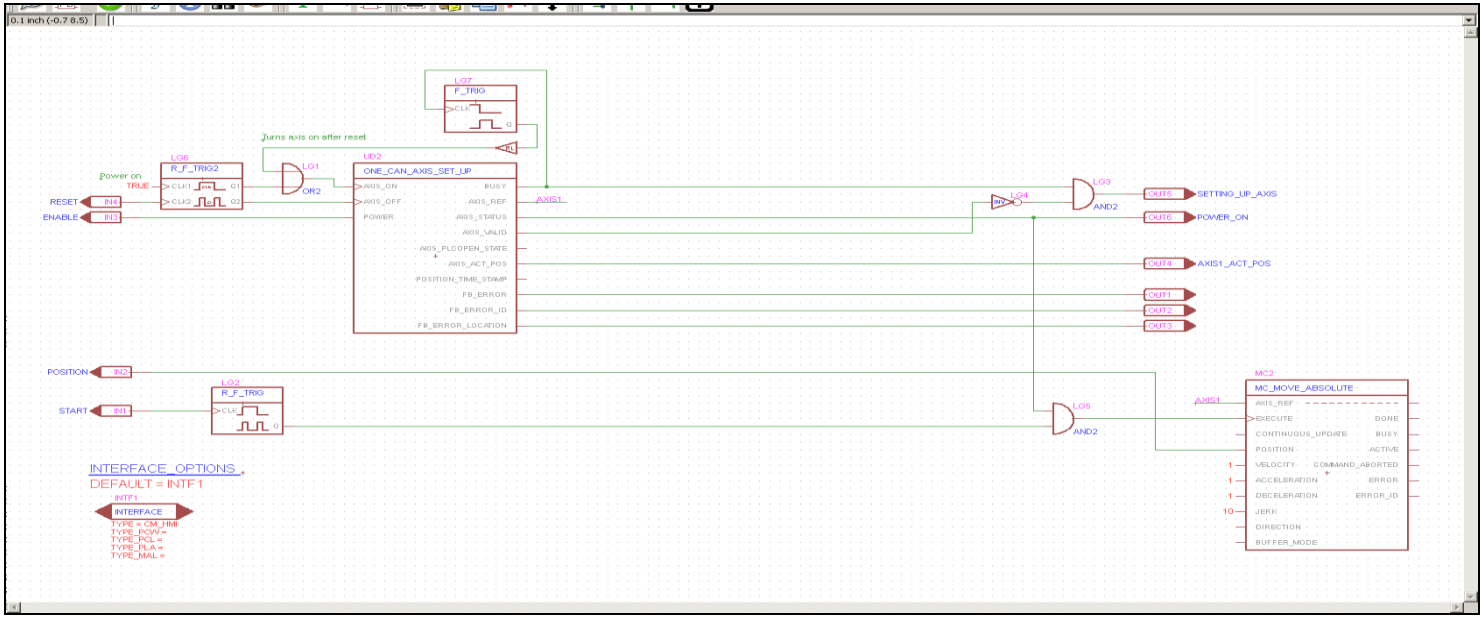
Now start a build for our changes. Again it should complete with success. If not, check your work.

We are now ready to add our CPP function block to the project.

Add the new function block to the project

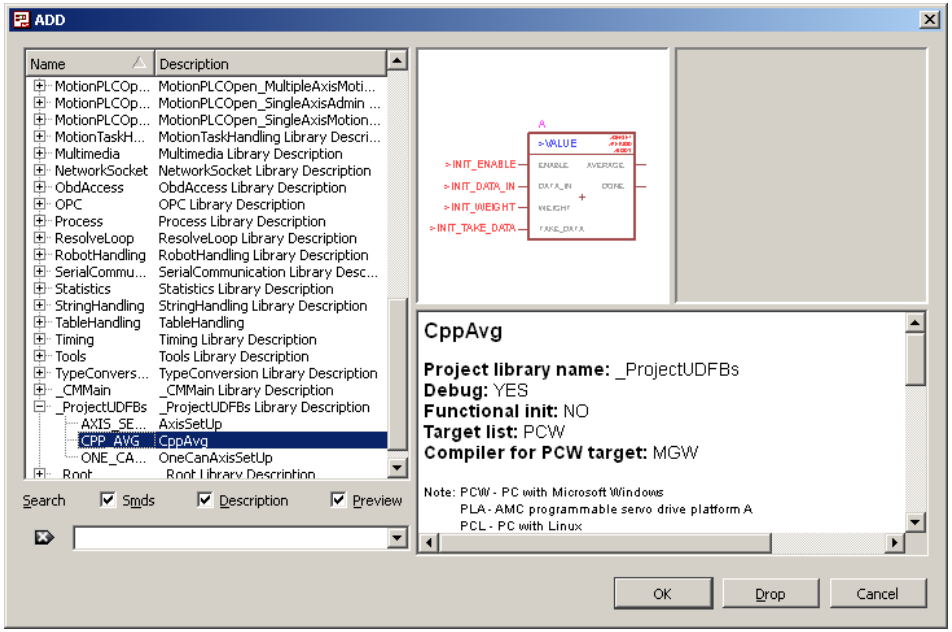


Open the “C_M_MAIN.sch” main schematic. Use the Block Select tool and Move Tool to make some room for our new logic.



Add the new function block to the project

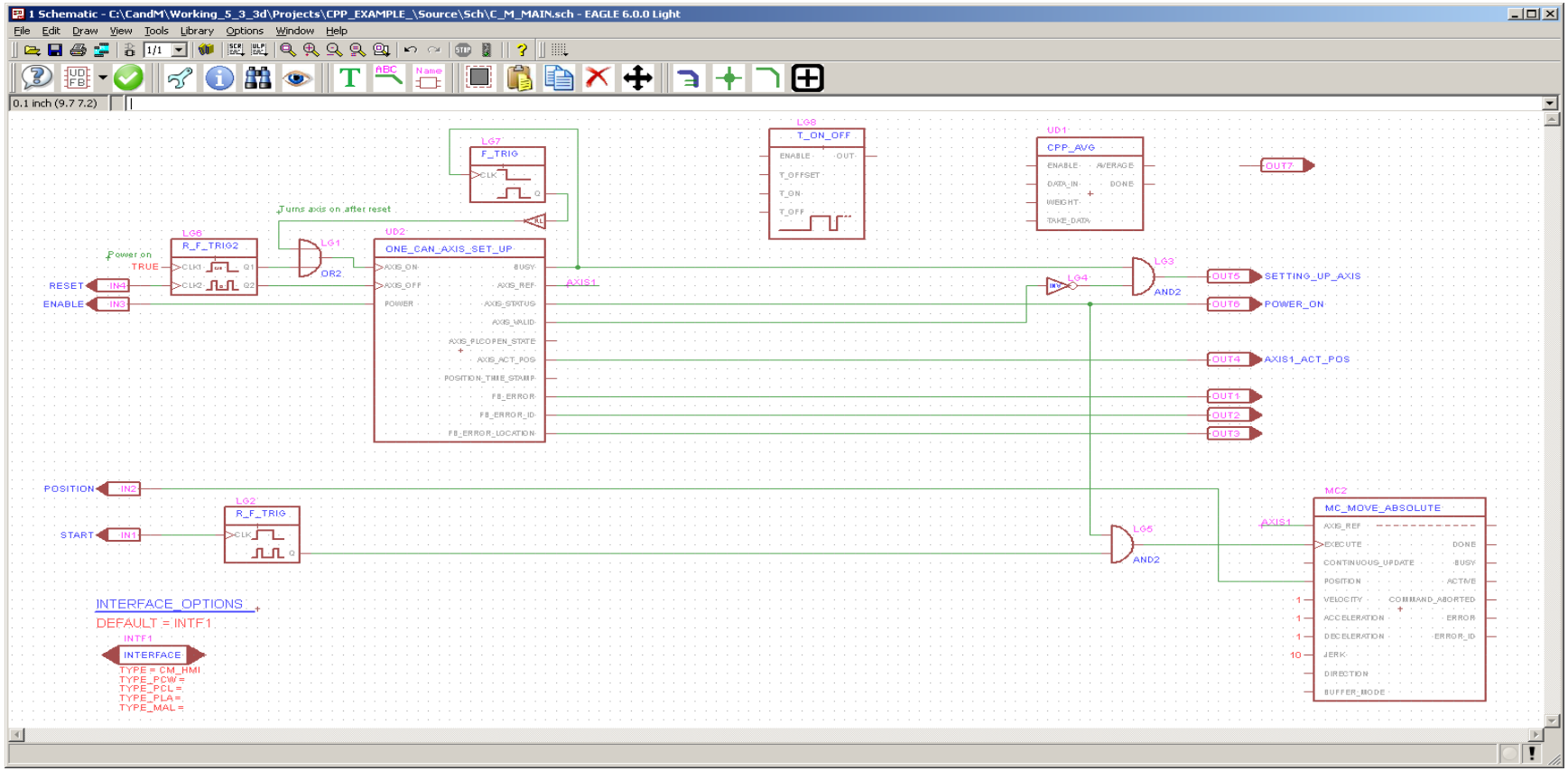
Click “Edit” “Add” or click the add icon  to open the library browser.



Scroll down to the ProjectUDFBs section and then click on CPP_AVG. Drop an instance of the block on the main schematic and press Esc. Scroll up to the Logic section and drop an instance of T_ON_OFF Clock Signal Generator onto the main schematic. Press Esc once. Scroll up again to the Basic Elements section and select OUTPUT. Drop an output onto the main schematic and press Esc twice to quit adding.

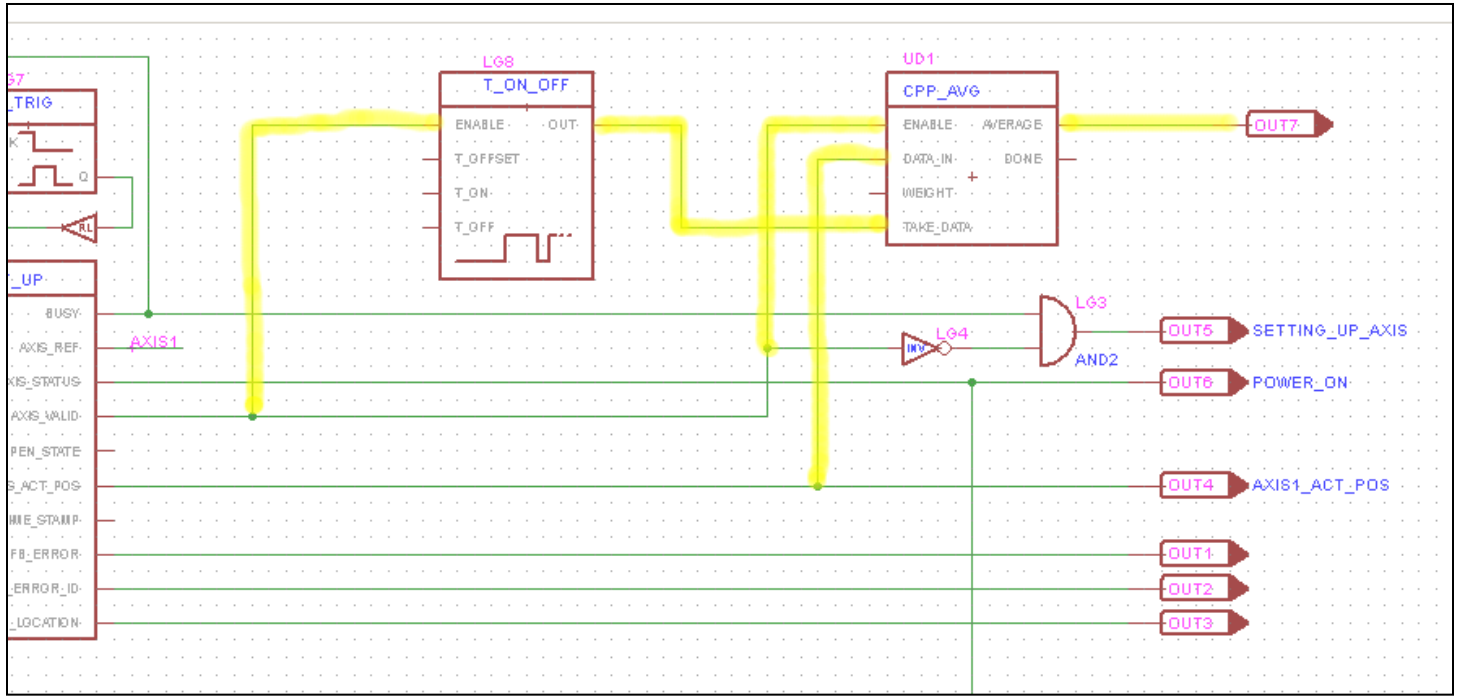
Connect the new blocks to the project

The Main schematic should look something like this:

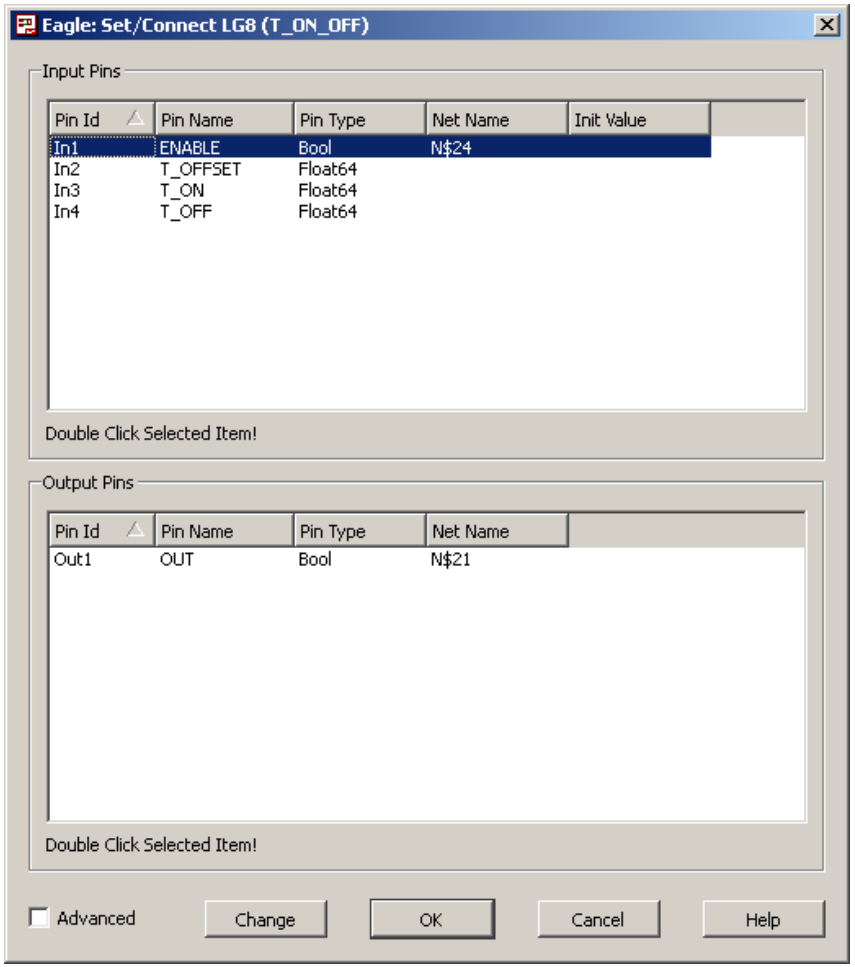


Connect the new blocks to the project

Add connections highlighted in YELLOW to the Main schematic.



Add Default values to control the averaging process.



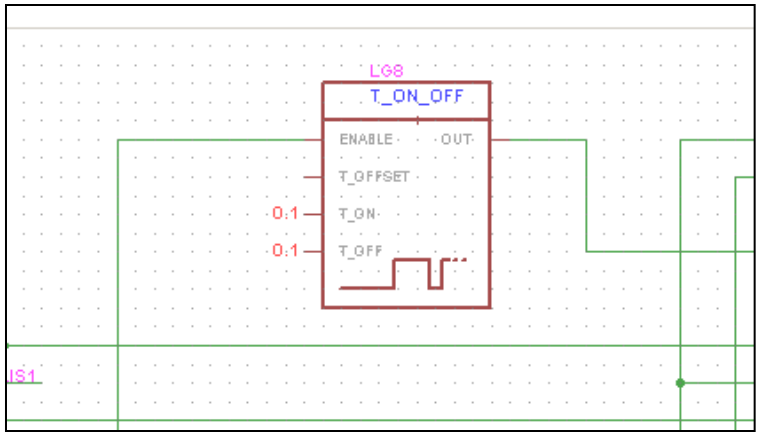
Right click the T_ON_OFF function block and choose “C&M Set/Connect”

Double click the line with “In3” “T_ON” and enter “0.1” into the Init Value window. Click “OK”

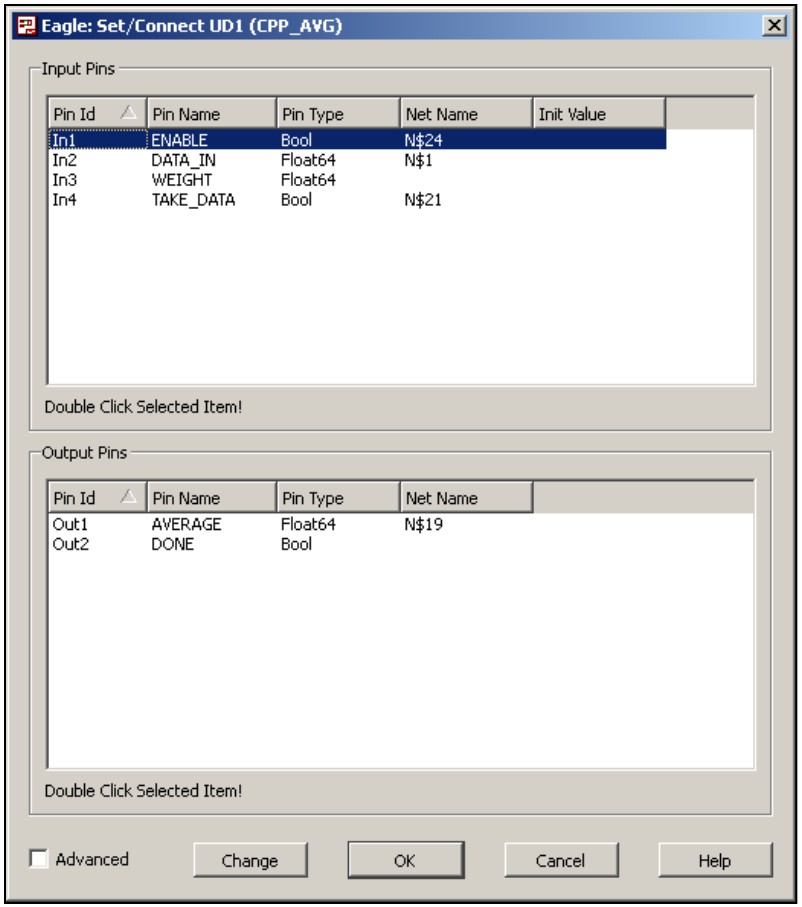
Do the same for the line with “In4” “T_OFF”

Click “OK” to close the window.

The schematic will look like this:



Add Default Values to Control the Averaging Process

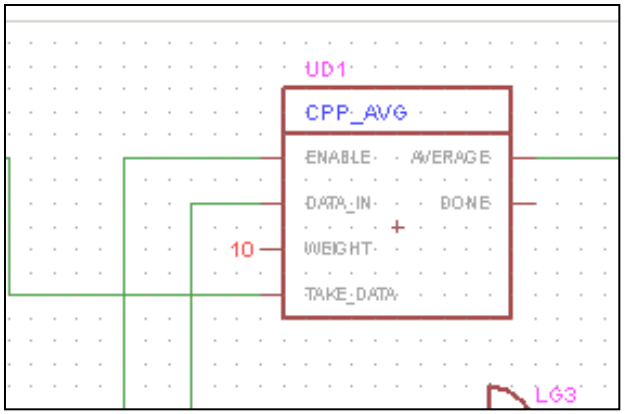


Right click the CPP_AVG function block and choose “C&M Set/Connect”

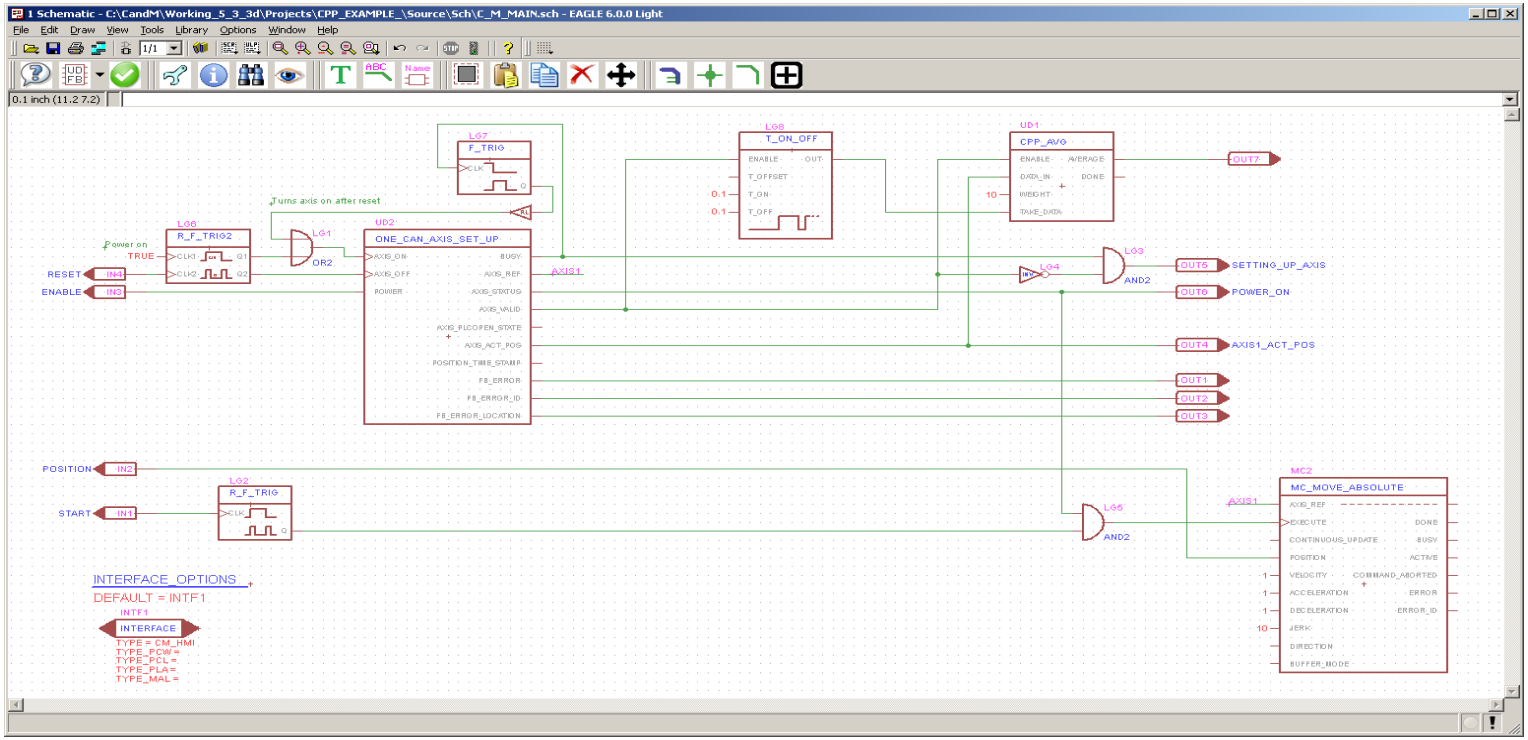
Double click the line with In3 WEIGHT and enter “10” into the Init Value window. Click “OK”

Click “OK” to close the window.

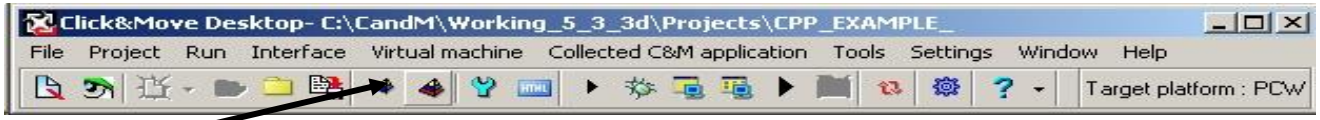
The schematic will look like this:



Verify Main Schematic Page



The Main schematic should look as shown. Save your project, all motion changes are complete.

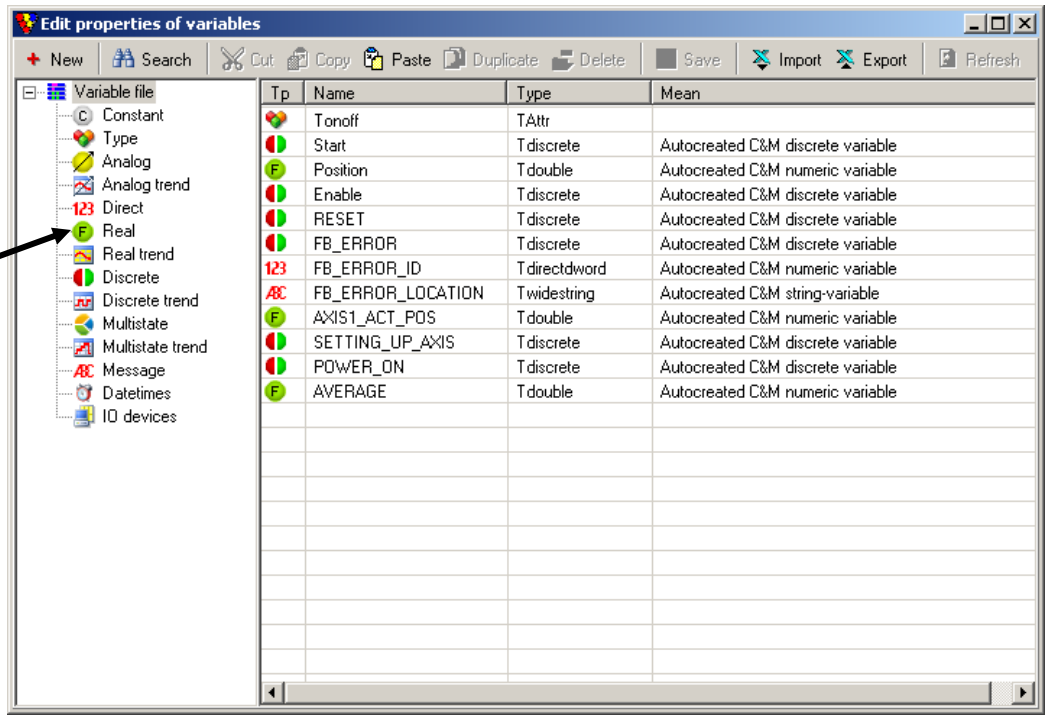


Build the project again, if the build fails, correct any mistakes and try again.

Change Units of the Average

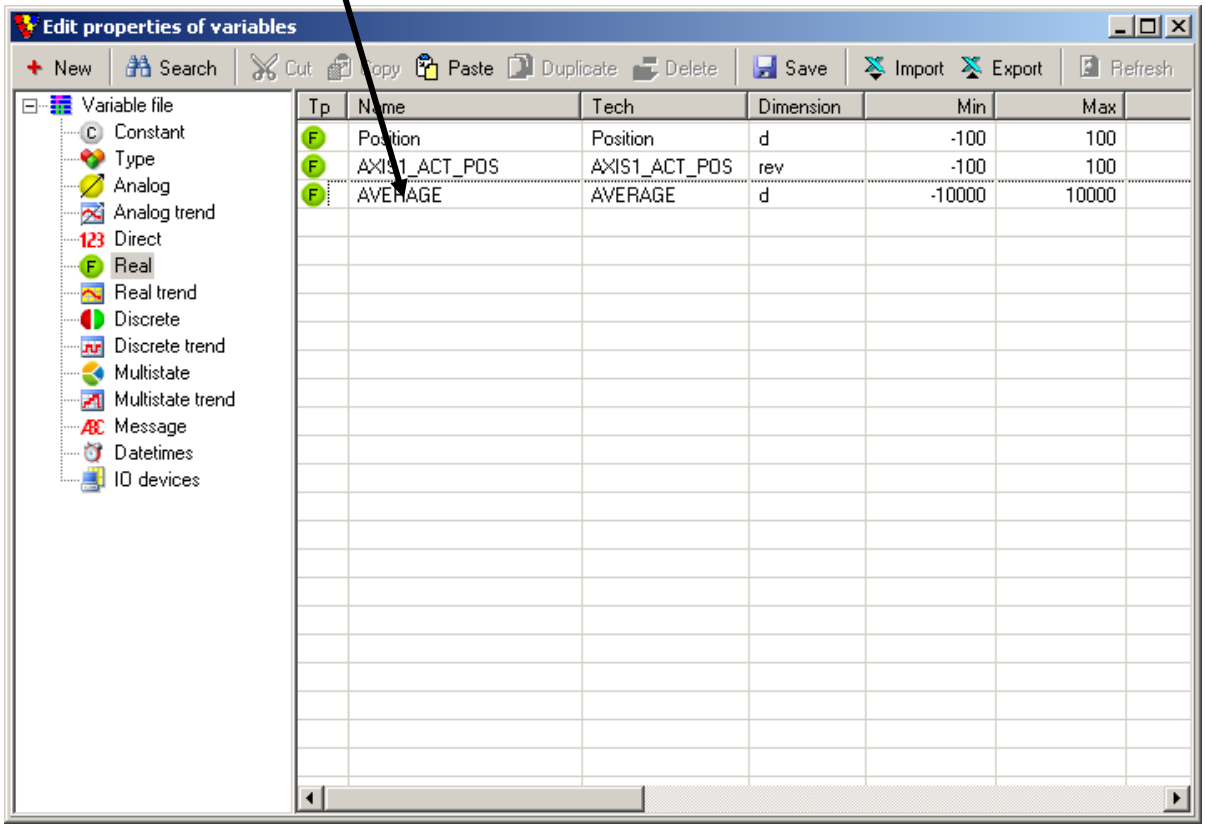
Click “Interface” “C&M-HMI” “Edit C&M-HMI Interface (Main) Variables” to display the variable properties.

Click on “F Real” to filter the list



Change Units of the Average

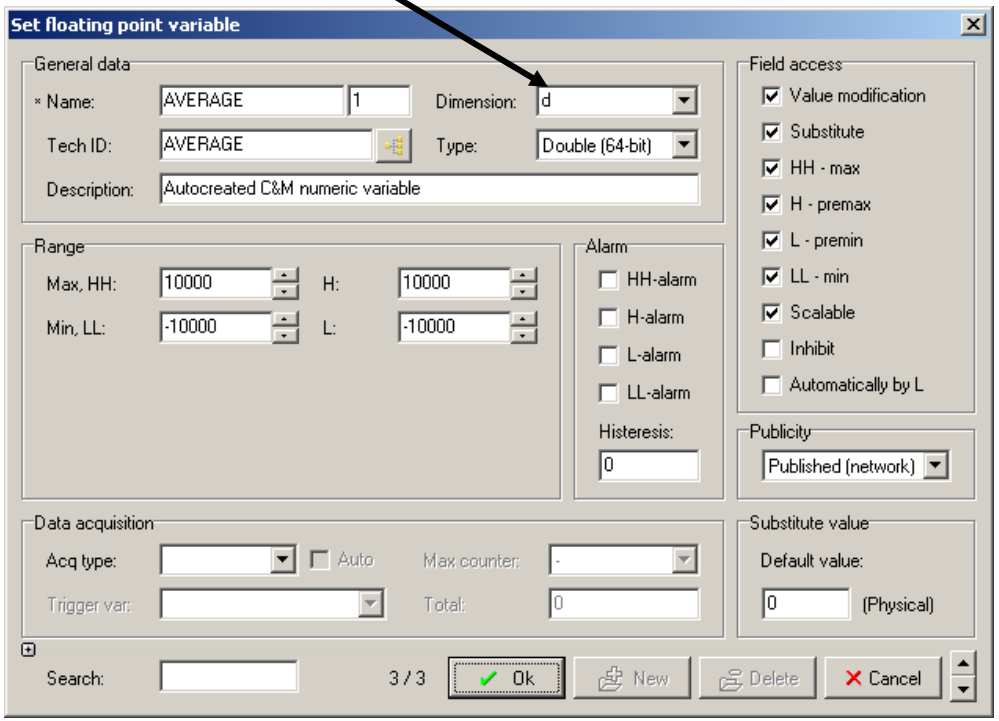
Double click the line containing the AVERAGE variable



Change Units of the Average

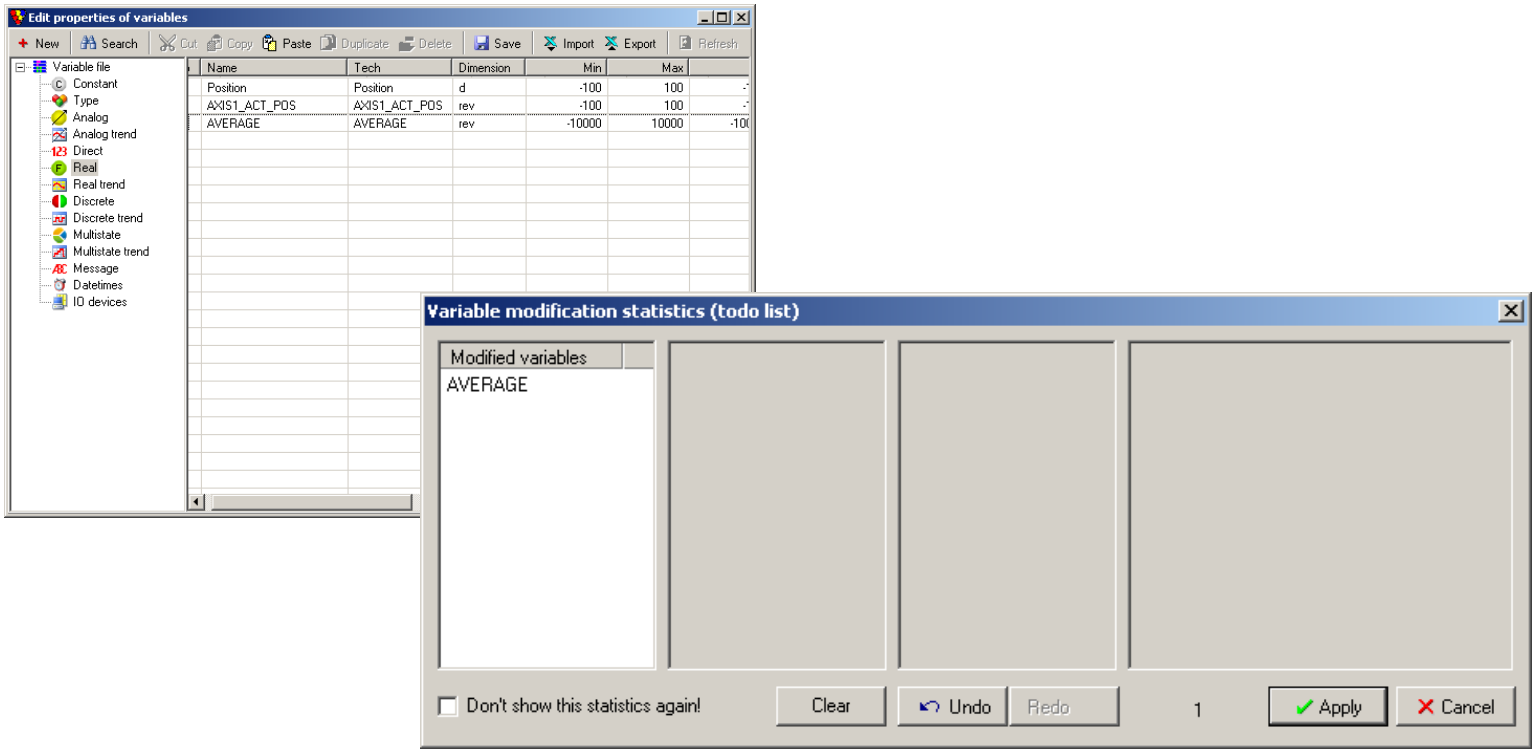
Change the text in the Dimension window to rev (don't use pull down)

Then click OK.

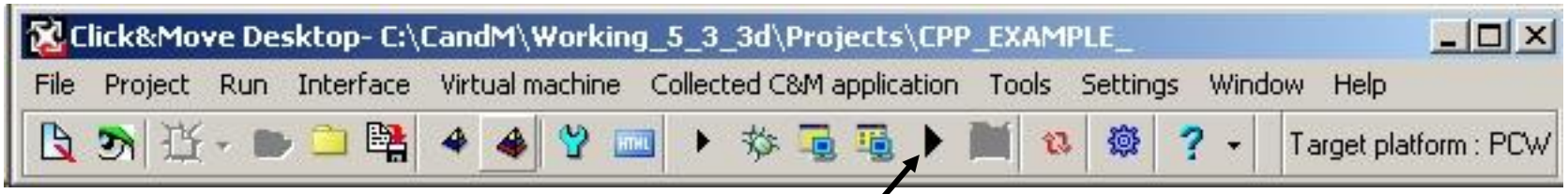


Change Units of the Average

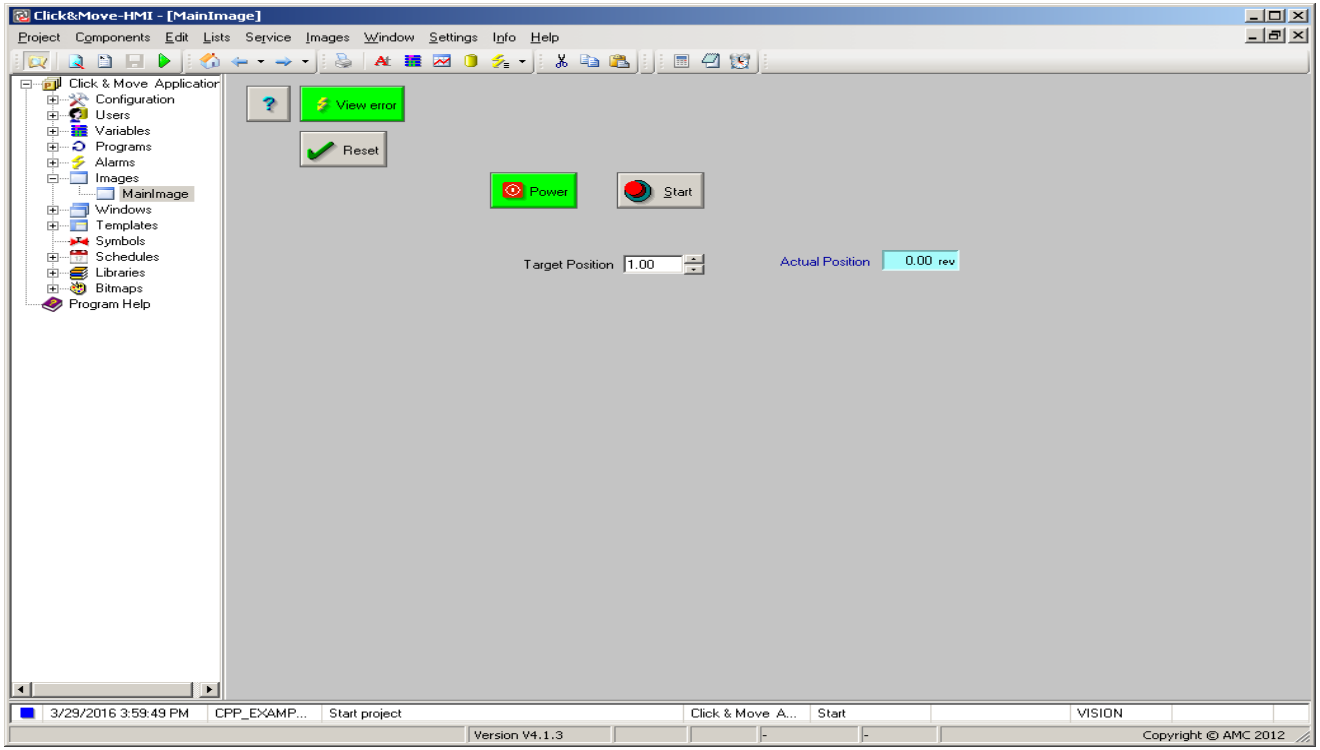
Click the “Save” button and then click “Apply” to close the Todo List window and accept the changes.



Run the Project and update the HMI



Launch the Motion, Virtual Axis, and the HMI with a click of the RUN ALL button.



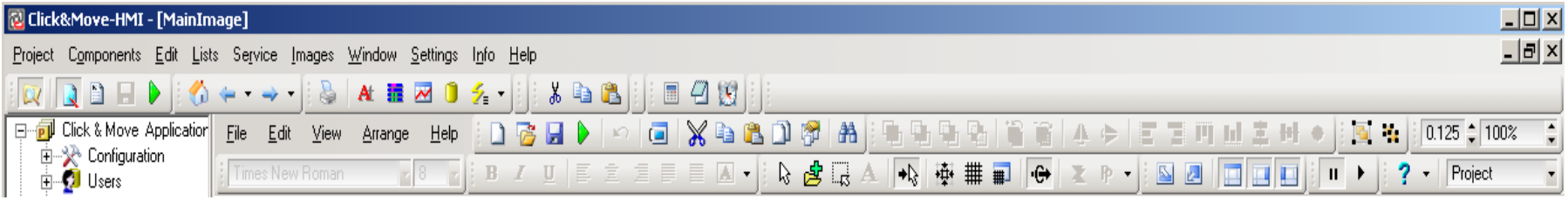
The HMI is launched at the same time and it looks like this:

Add Average Position display to the HMI

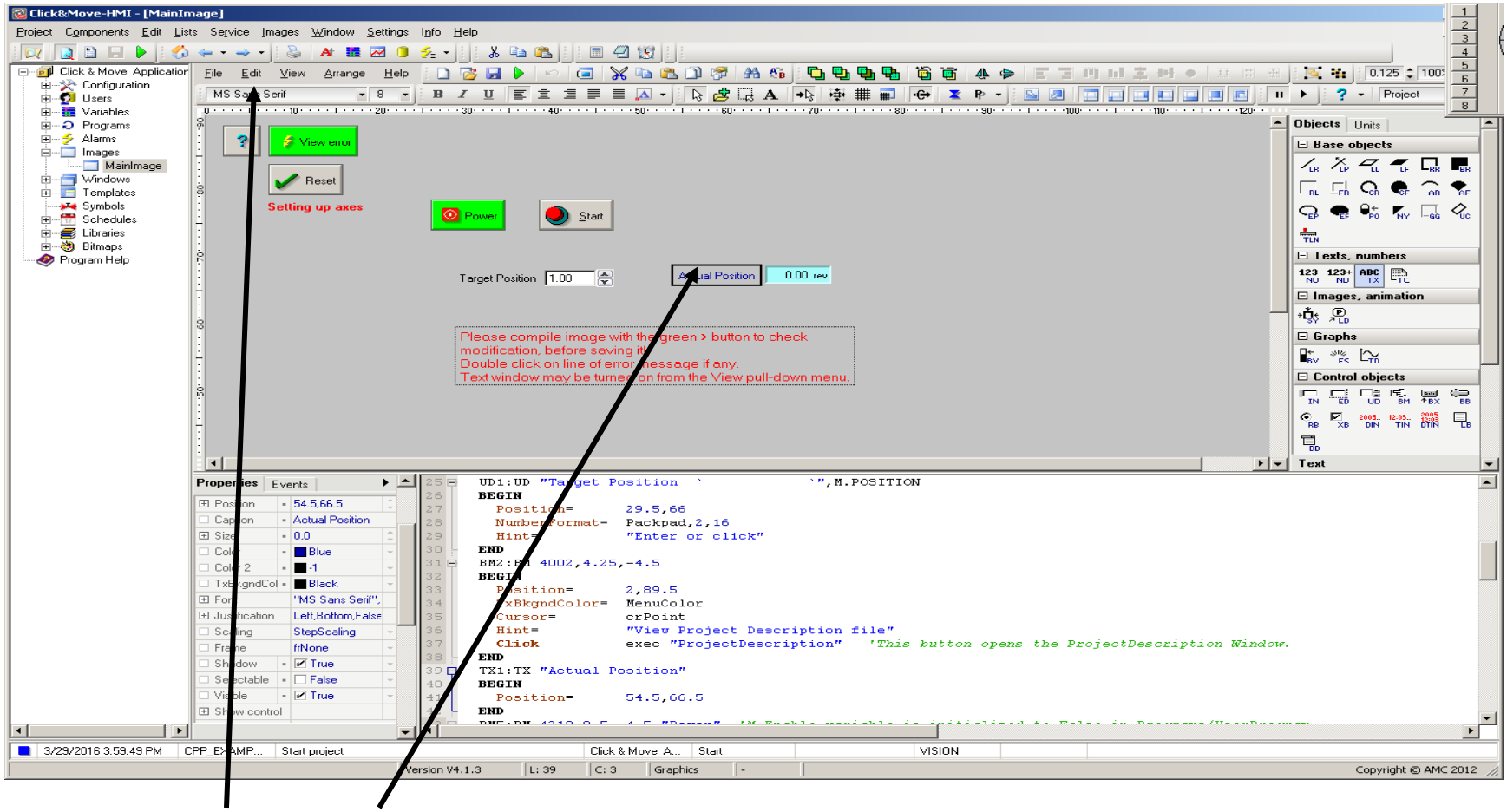
This is the HMI run time tool bar.

Toggle between Run mode and Edit mode with a click of the button.

While Edit mode is active, an enhanced tool bar supports edit functions.



Add Average Position display to the HMI

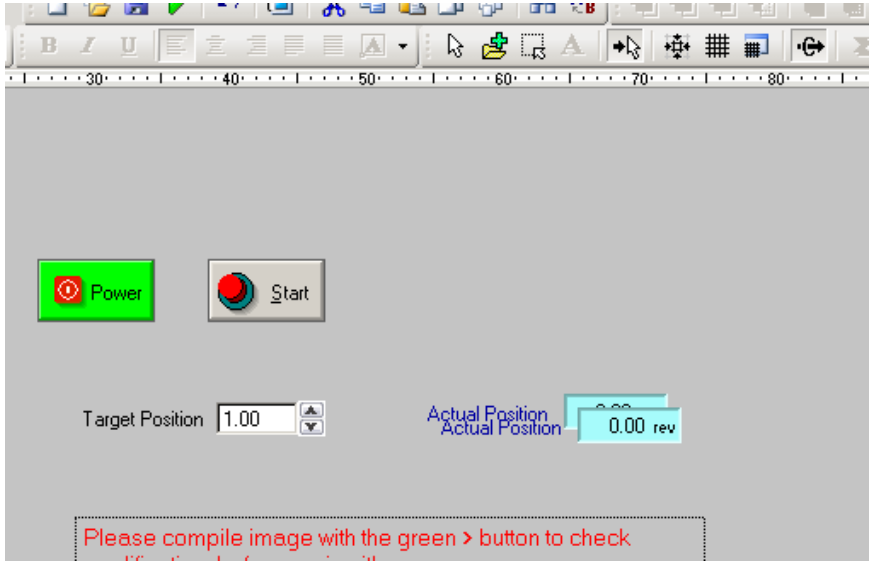


Click on the “Actual Position” text near the center of the screen. Then click “Edit” “Duplicate”

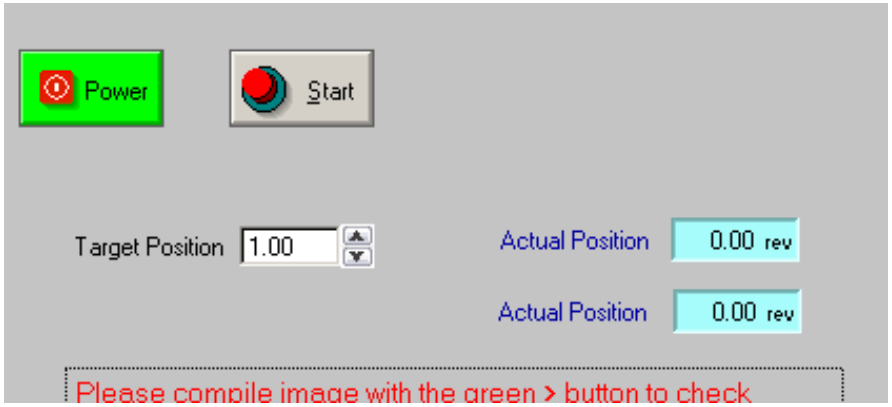
Do the same thing for the numeric display next to the text.

Add Average Position display to the HMI

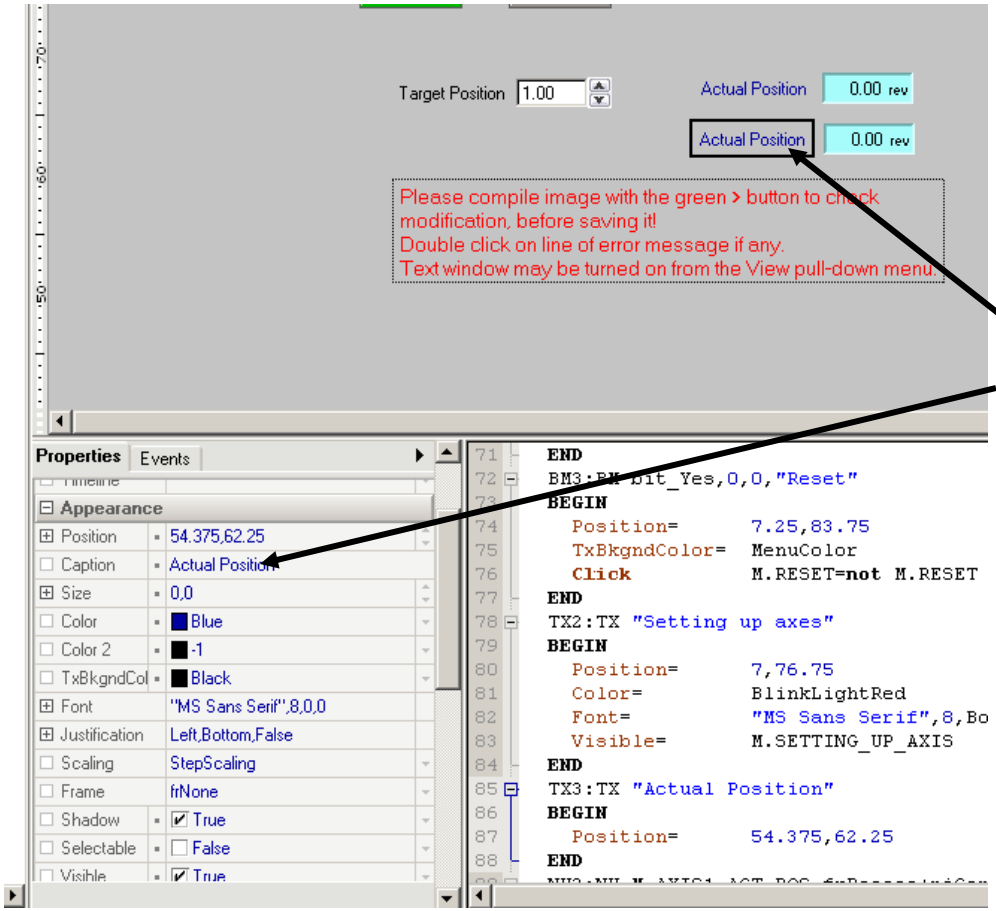
You should have two copies of the Actual Position text and numeric display.



Move them so they do not overlap.

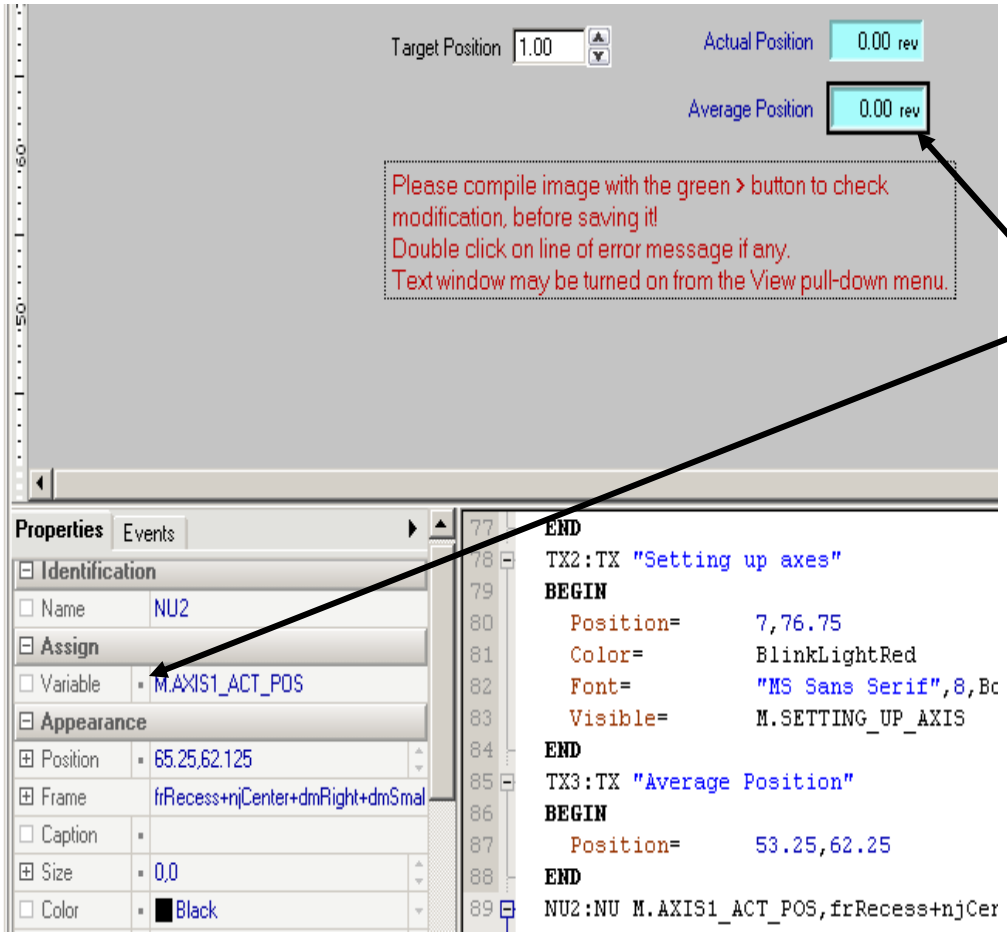


Add Average Position display to the HMI



Click on the new “Actual Position” text we added in the last step. In the properties window find the Caption setting and change it to :Average Position”

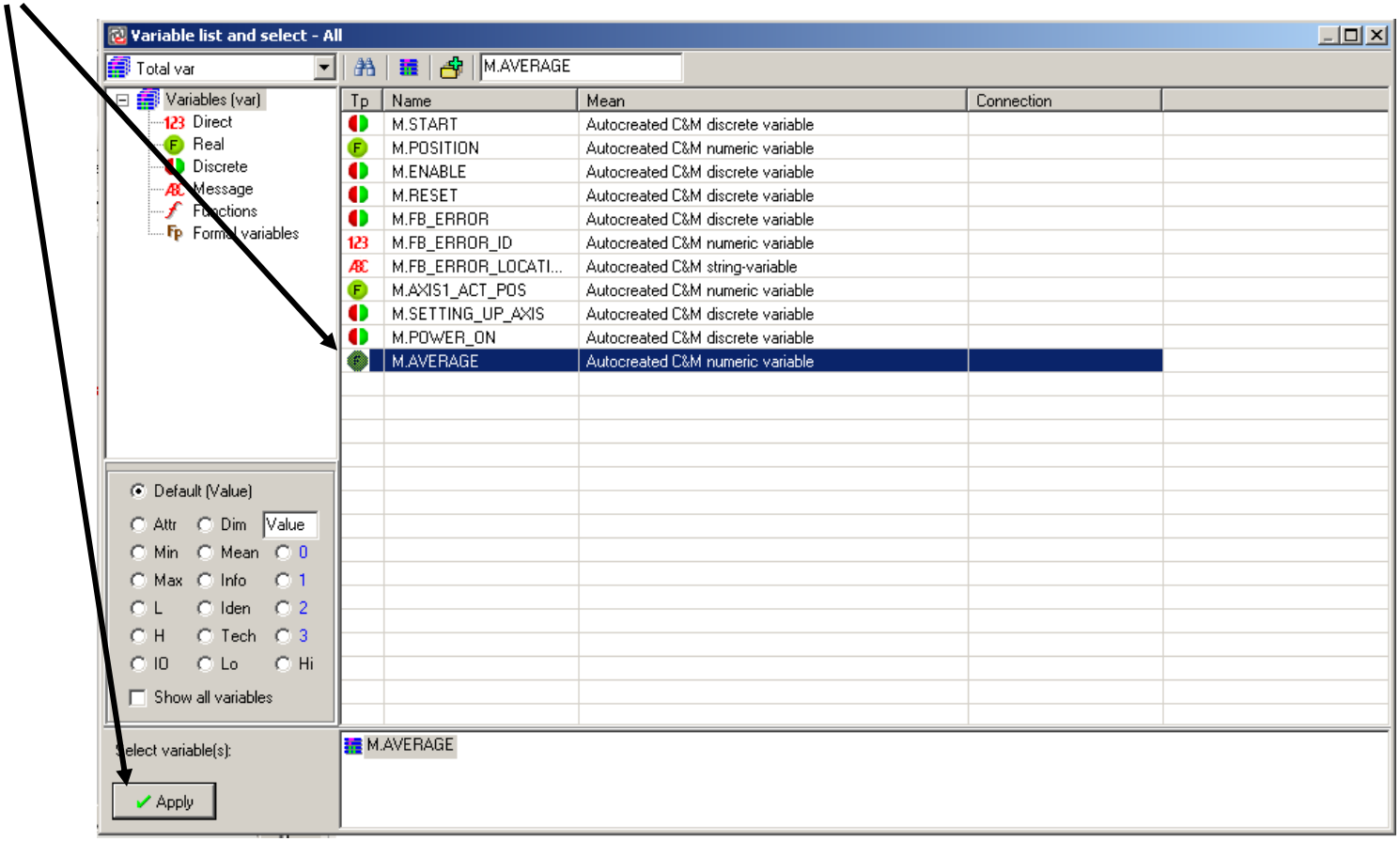
Add Average Position display to the HMI



Click on the new numeric window we added. In the properties window find the “Variable” setting and click the button with the single dot in the middle.

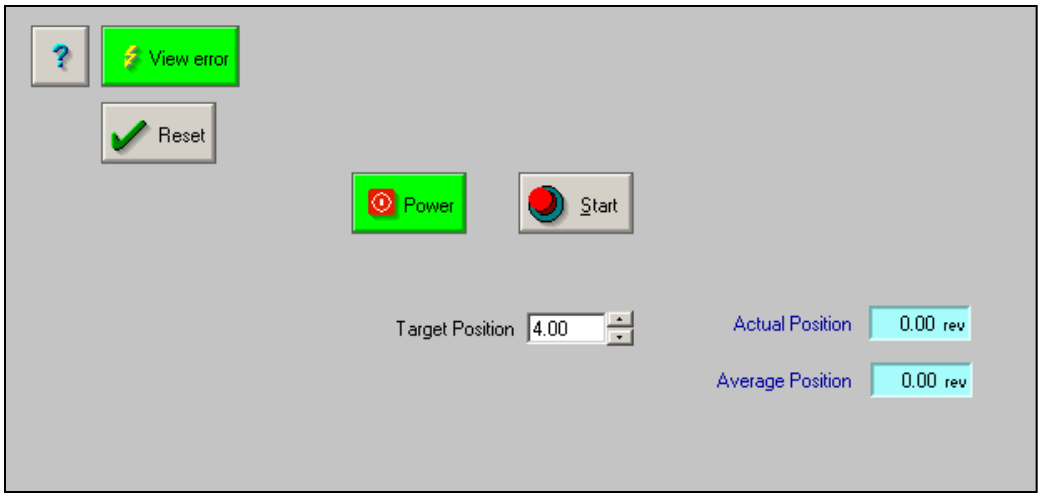
Add Average Position display to the HMI

Click “M.AVERAGE” in the variable list and then click “Apply”



Switch the HMI back to Run Mode

Toggle between Run mode and Edit mode with a click of the button.



To run the project click the power button to turn on the amplifier. Enter a target position and click Start.

The Average Position laggs the actual position by an amount set by the filter Weight and the TakeData cycle time.

END OF PRESENTATION