



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.

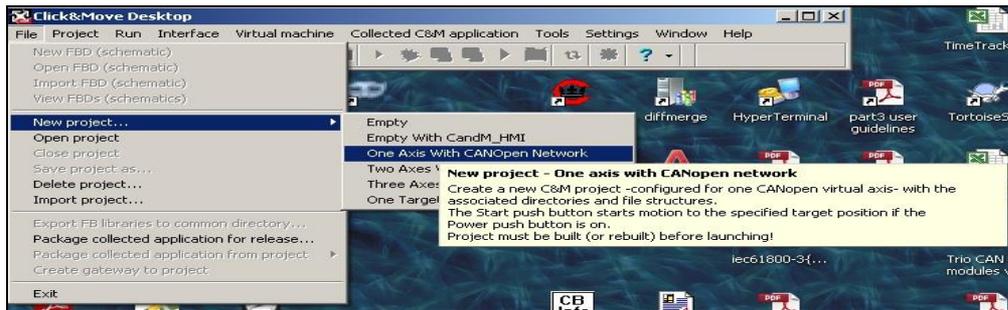
The Actual Position stream is sampled every 200 ms at which time the NEW_AVERAGE is calculated .

$$\text{NEW_AVERAGE} = (\text{sampled value} / \text{WEIGHT}) + (\text{LAST_AVERAGE} - (\text{LAST_AVERAGE} / \text{WEIGHT}))$$

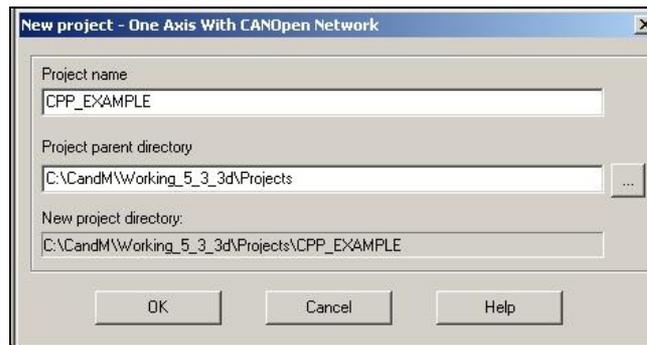
Create New Project

Start C&M and close the open project (if any).

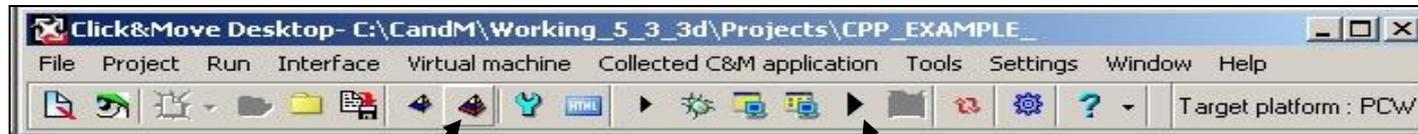
Click “File” “New project” “One Axis With CanOpen Network”.



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.

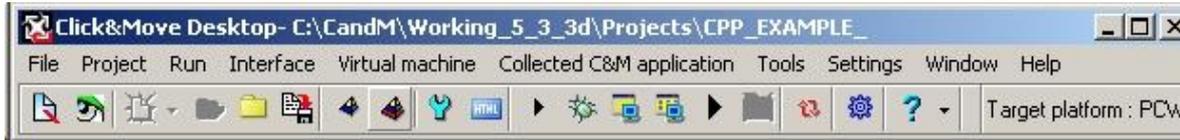
```
make.exe[1]: Nothing to be done for 'build'.  
make.exe[1]: Leaving directory `C:/CandM/WOFDE5~1/Projects/ CPP_EX~2'  
make: Leaving directory `C:/PROGRA~1/A-M-C/CA8154~1/System/MAKEFI~1'  
Project is successfully built.
```

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.

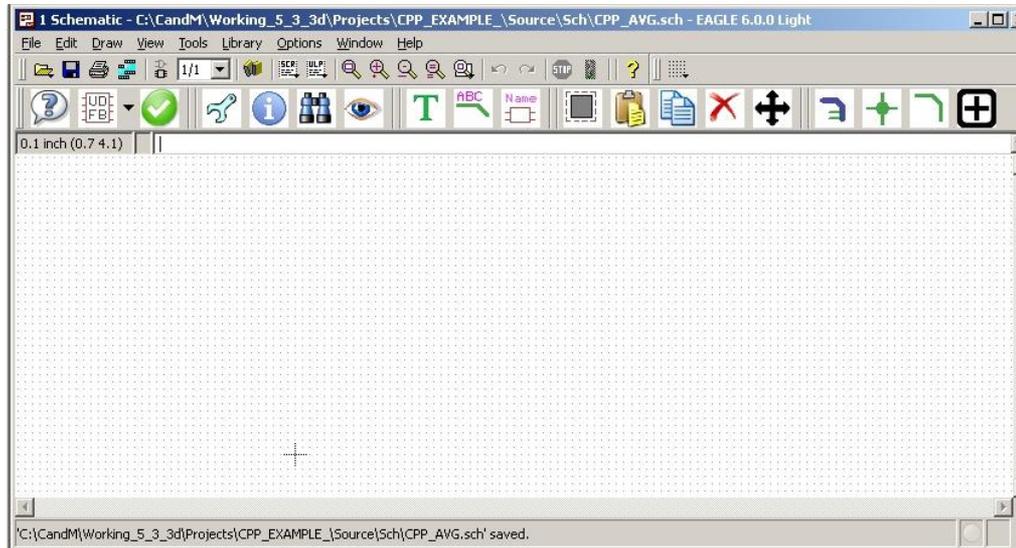
Stop the project and close the HMI and Virtual Axis before going on to the next step

Add Empty Function Block to the Project

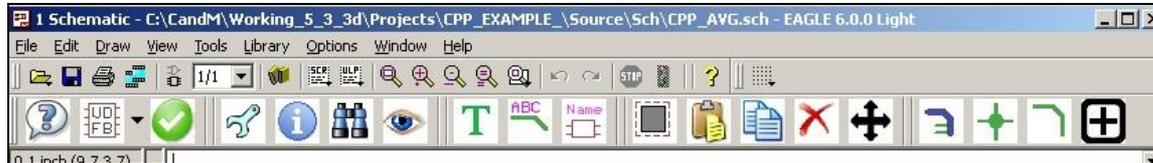
On the Desktop click “File” “New FDB (schematic)” to start a new schematic.



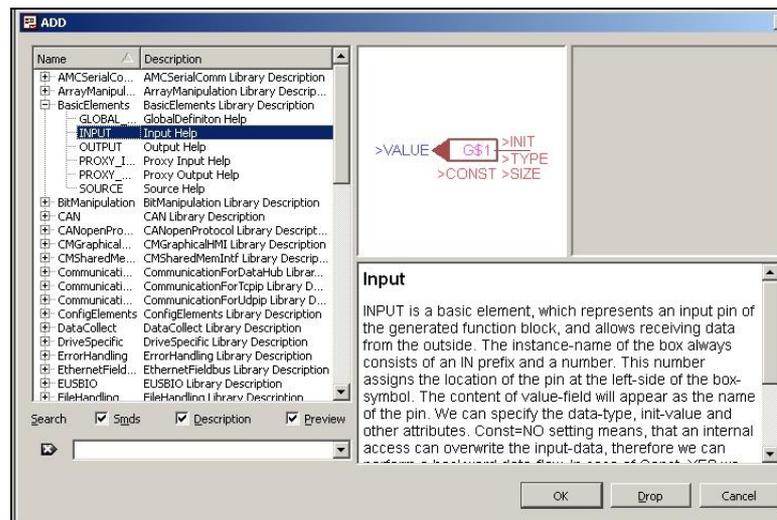
When the new schematic opens up click “File” “Save As” and save the file as “CPP_AVG”.



Add Inputs and Outputs to the Schematic



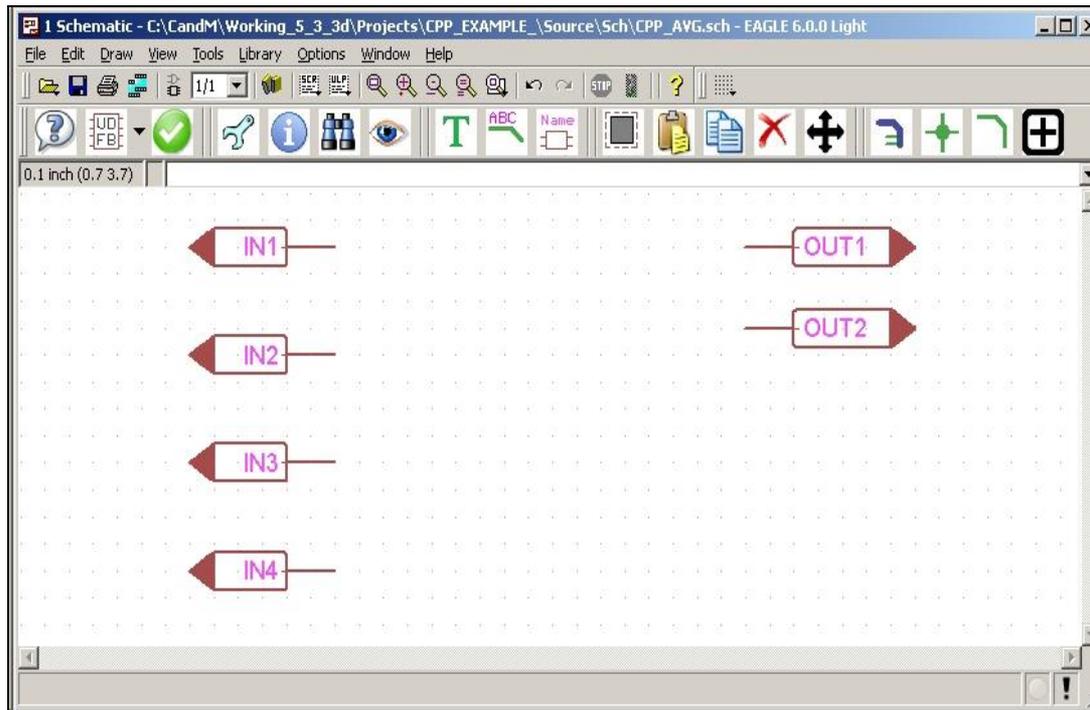
Click “Edit” “Add” or click the add icon  to open the library browser.



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

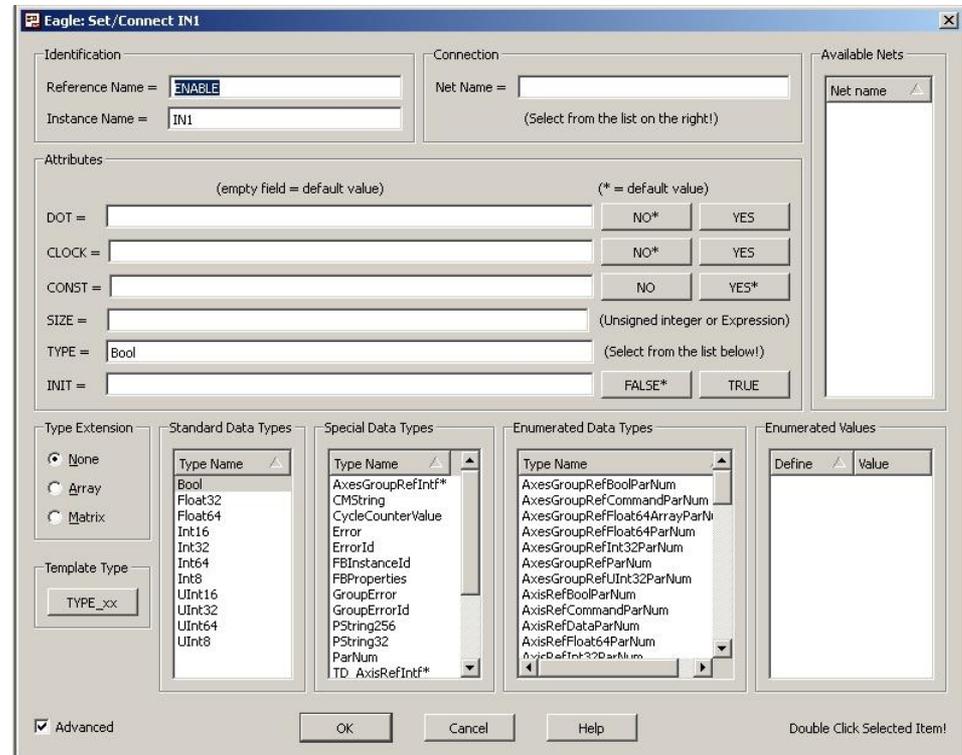
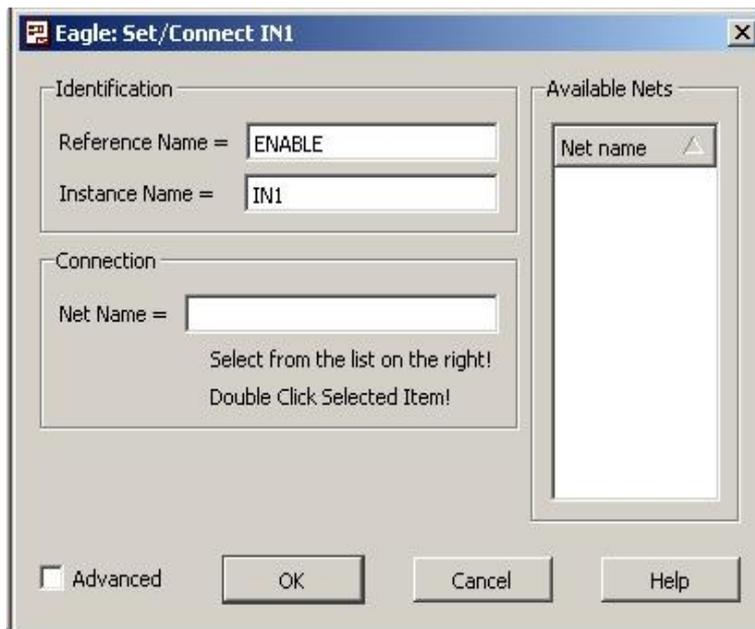
The schematic should look something like this:



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

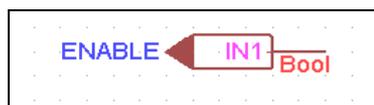
Add Inputs and Outputs to the Schematic

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.

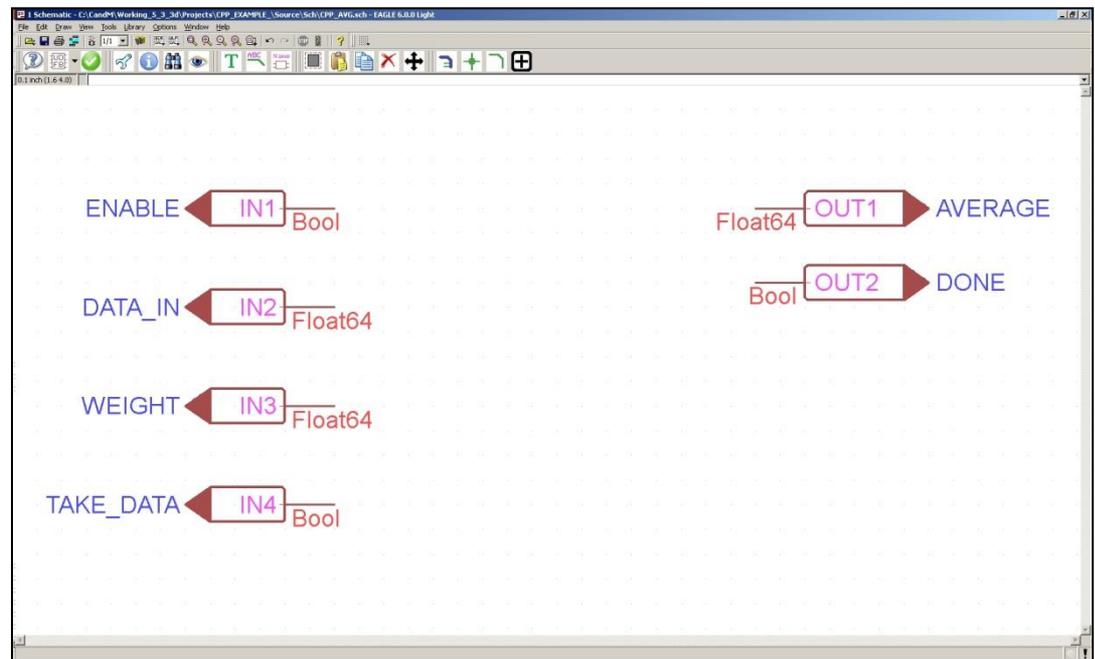


Add Inputs and Outputs to the Schematic

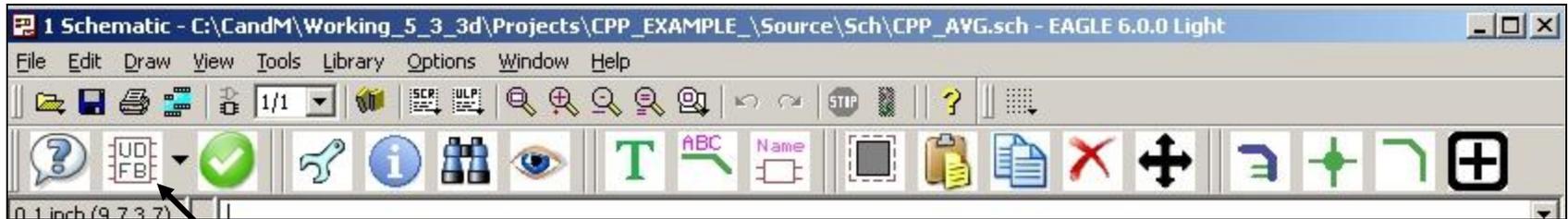
Now repeat the process to assign names and types to the other IO's.

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:

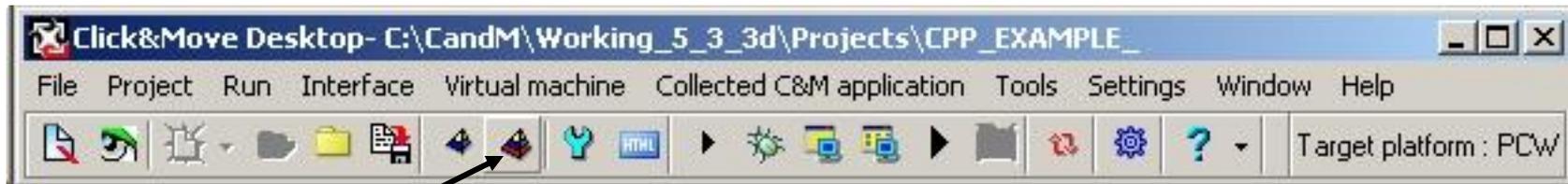


Add the new Schematic to the Library



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.

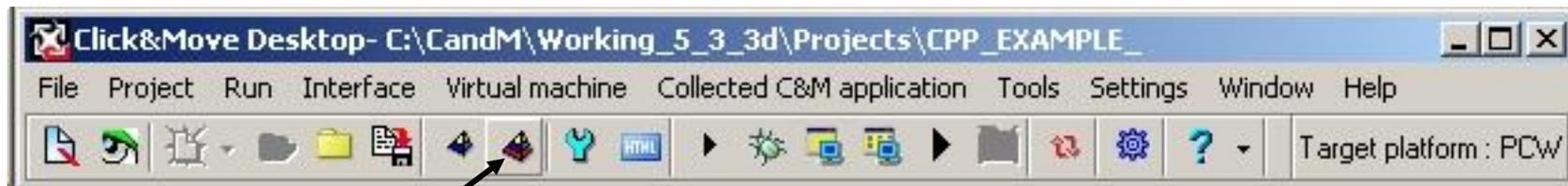
Use Windows File Explorer to move these two files “CppavgImpl.cpp” and “CppavgImpl.h”

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;
    }
}
```

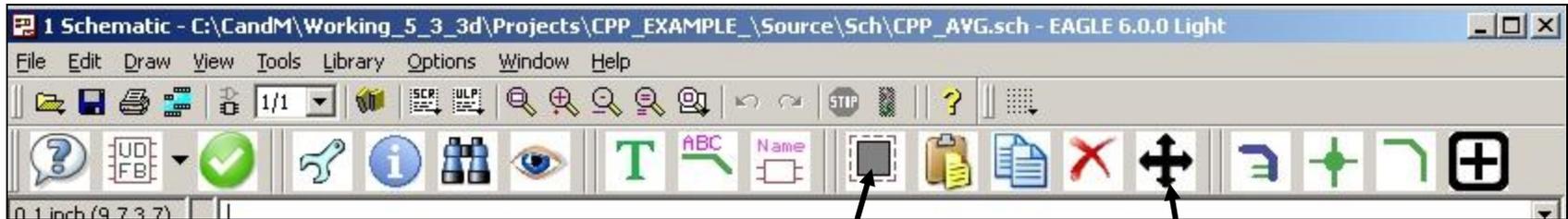
Build the project to verify the work



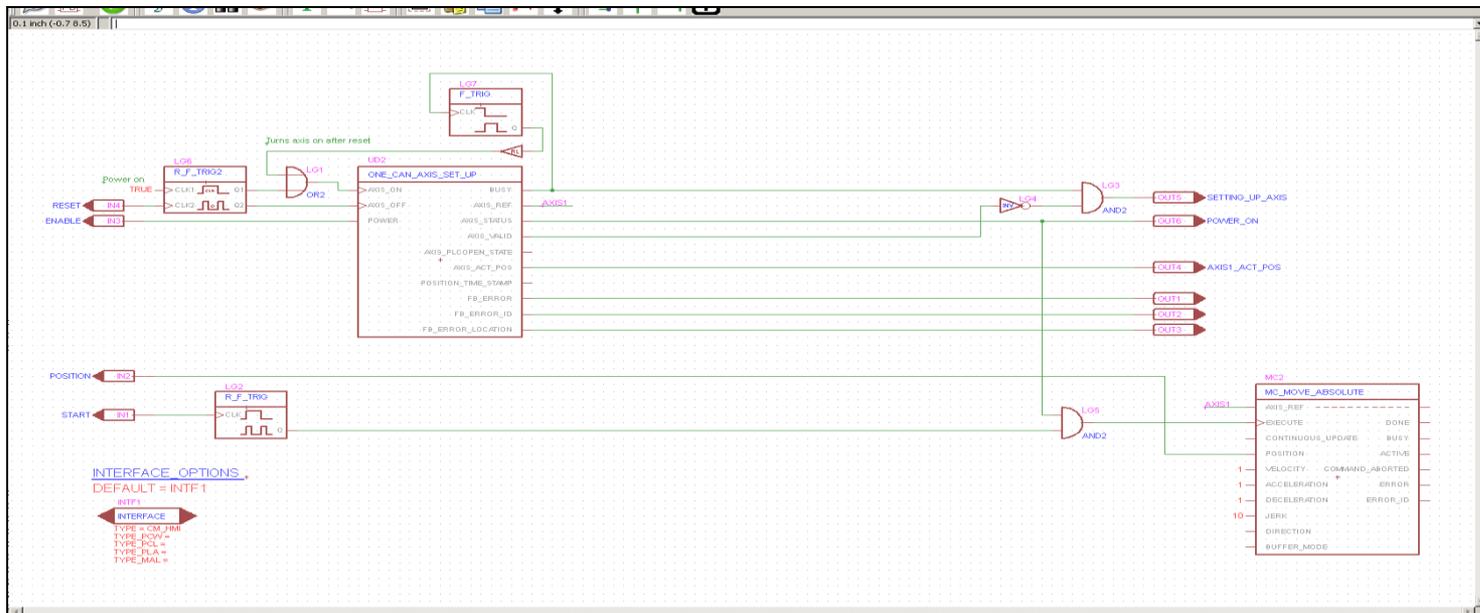
Now start a build for our changes. Again it should complete with success. If not, check your work.

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

Add the new function block to the project

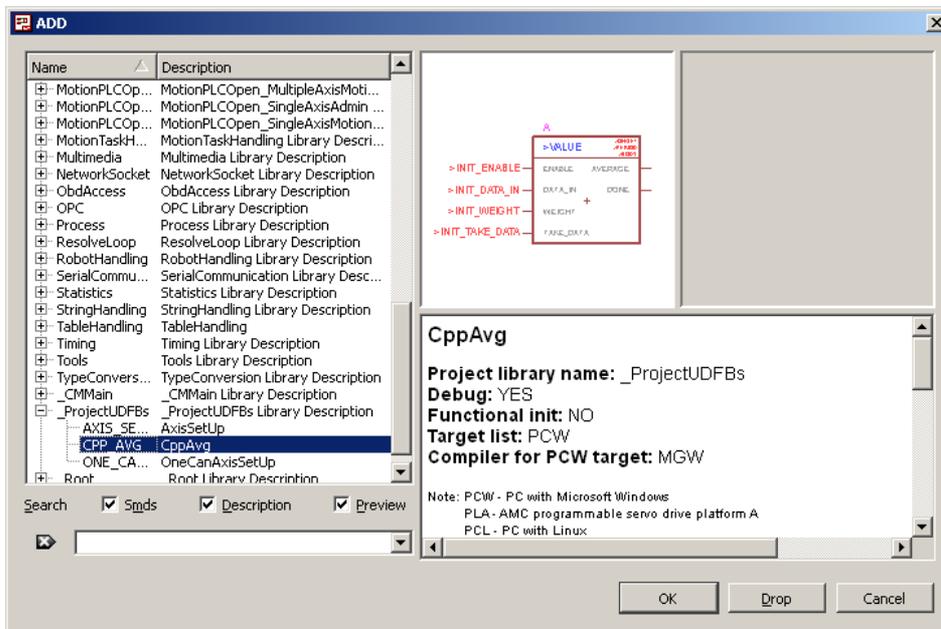


Open the "C_M_MAIN.sch" main schematic. Use the Block Select tool and Move Tool to make some room for our new logic.



Add the new function block to the project

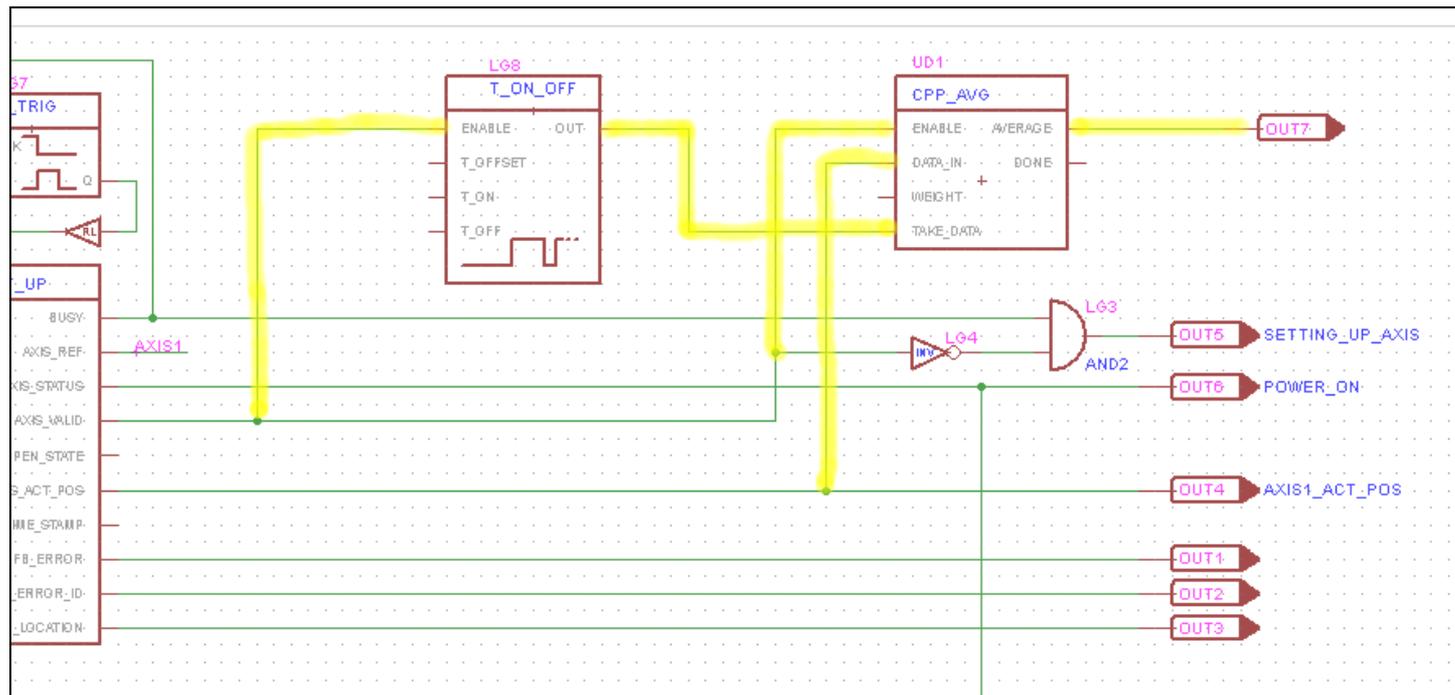
Click “Edit” “Add” or click the add icon  to open the library browser.



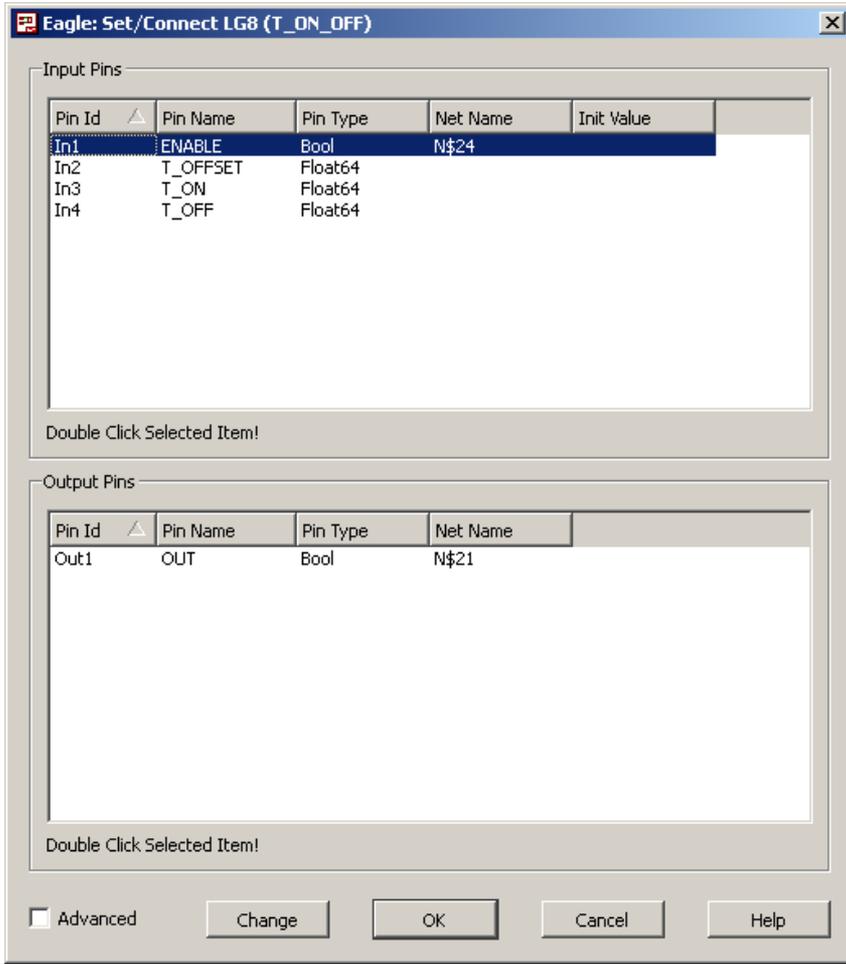
Scroll down to the ProjectUDFBs section and then click on CPP_AVG. Drop an instance of the block on the main schematic and press Esc. Scroll up to the Logic section and drop an instance of T_ON_OFF Clock Signal Generator onto the main schematic. Press Esc once. Scroll up again to the Basic Elements section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

Connect the new blocks to the project

Add connections highlighted in YELLOW to the Main schematic.



Add Default values to control the averaging process.



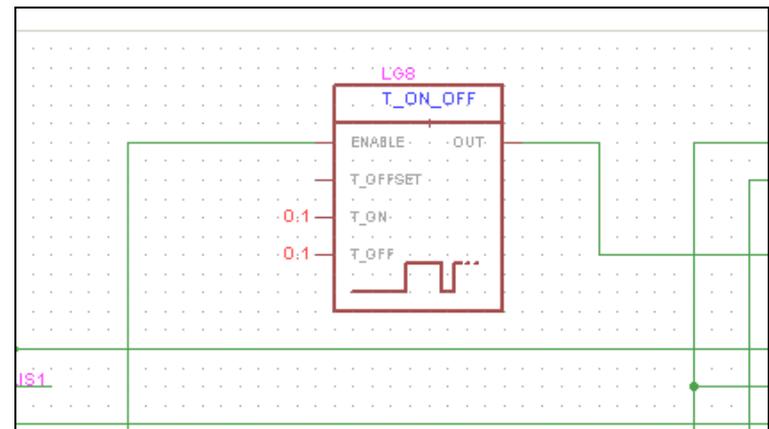
Right click the T_ON_OFF function block and choose “C&M Set/Connect”

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

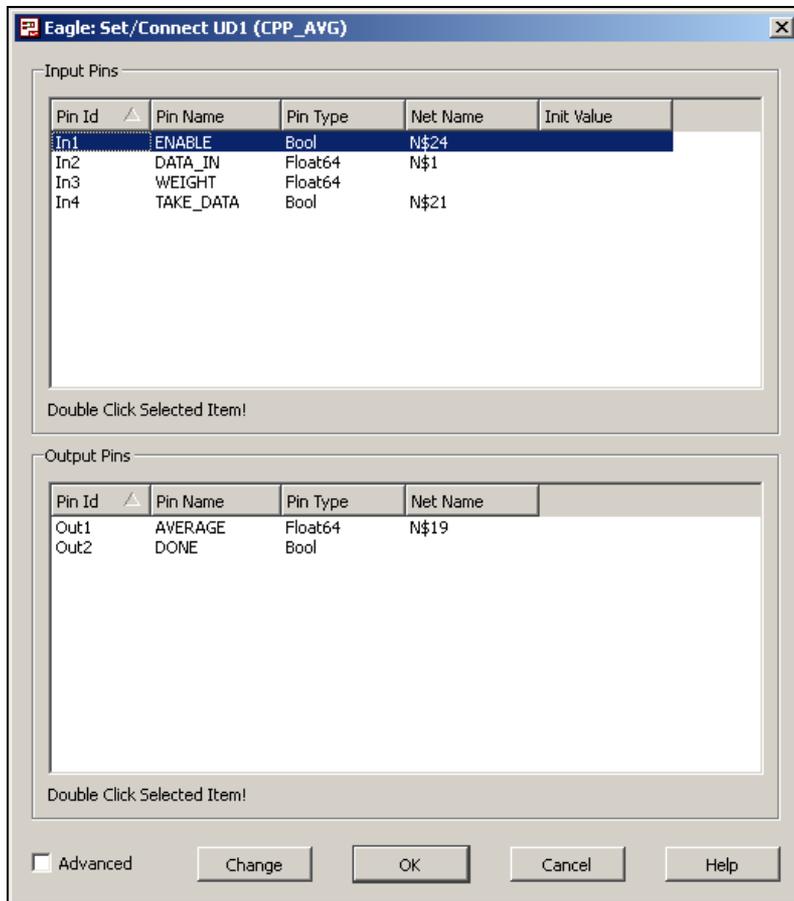
Do the same for the line with “In4” “T_OFF”

Click “OK” to close the window.

The schematic will look like this:



Add Default Values to Control the Averaging Process

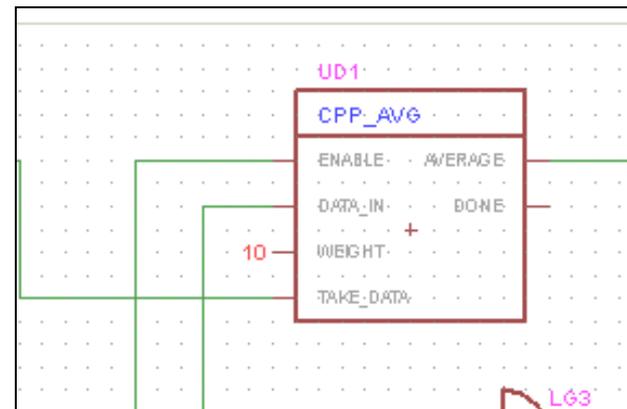


Right click the CPP_AVG function block and choose “C&M Set/Connect”

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

Click “OK” to close the window.

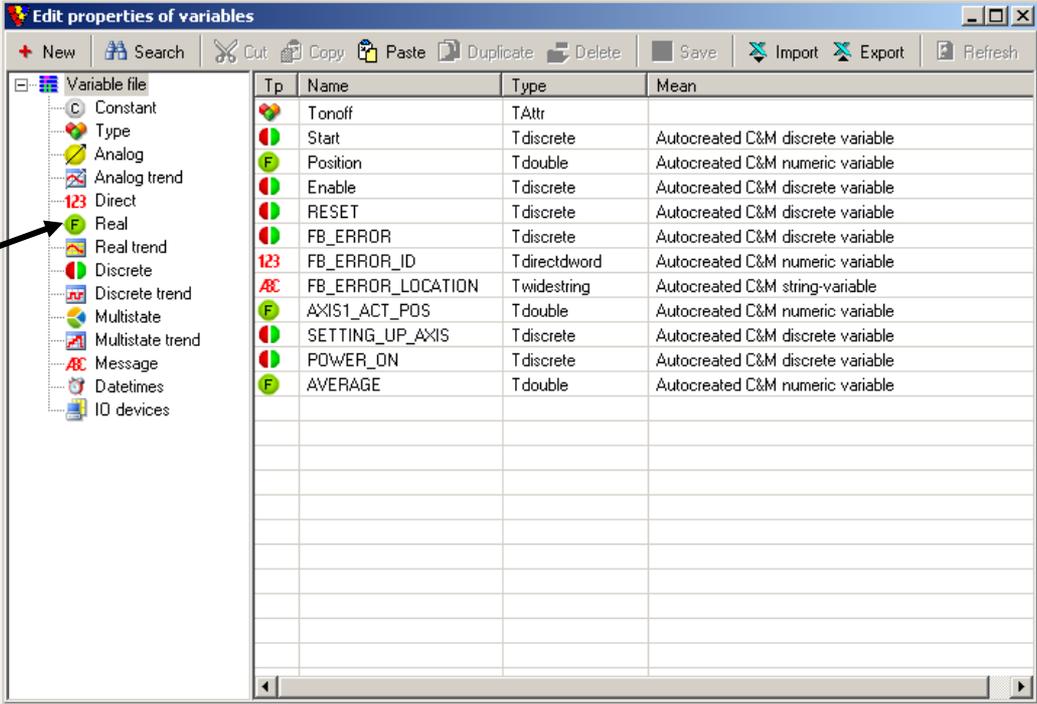
The schematic will look like this:



Change Units of the Average

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

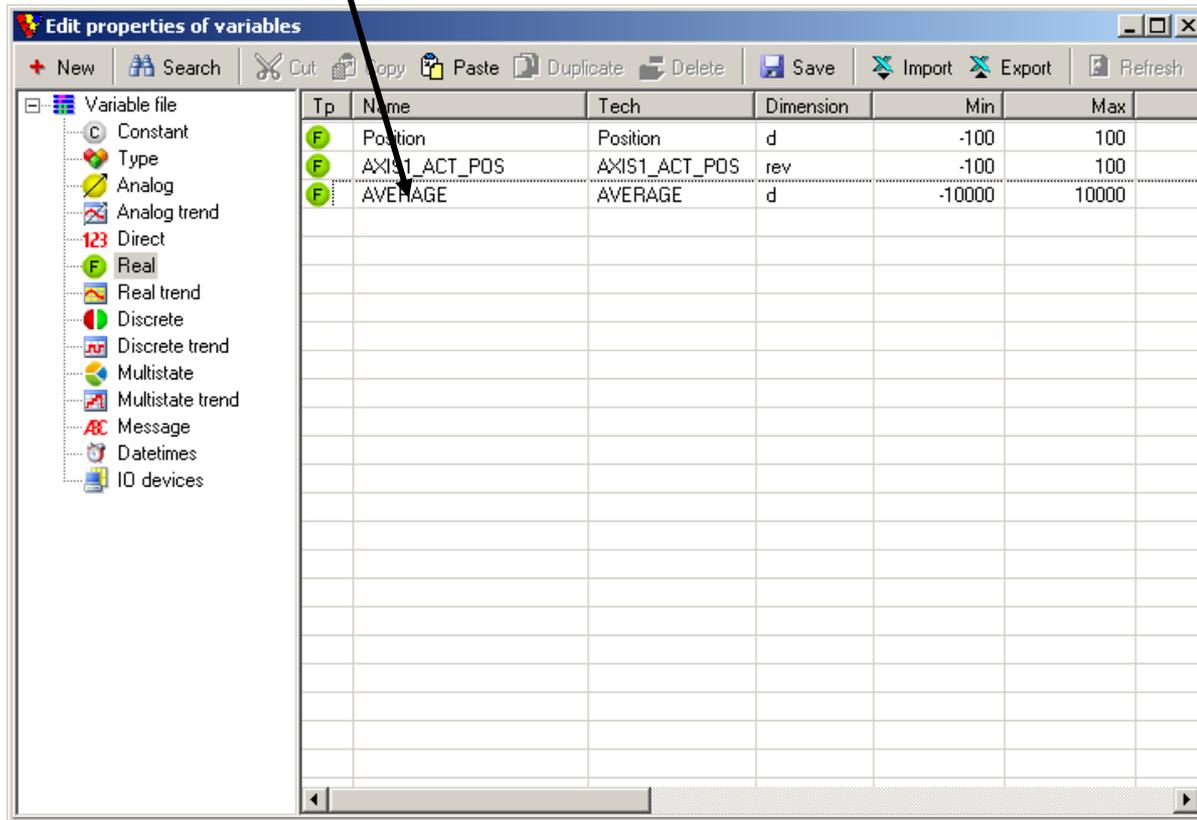
Click on “F Real”
to filter the list



Tp	Name	Type	Mean
	Tonoff	TAttr	
	Start	Tdiscrete	Autocreated C&M discrete variable
F	Position	Tdouble	Autocreated C&M numeric variable
	Enable	Tdiscrete	Autocreated C&M discrete variable
	RESET	Tdiscrete	Autocreated C&M discrete variable
	FB_ERROR	Tdiscrete	Autocreated C&M discrete variable
123	FB_ERROR_ID	Tdirectdword	Autocreated C&M numeric variable
	FB_ERROR_LOCATION	Twstring	Autocreated C&M string-variable
F	AXIS1_ACT_POS	Tdouble	Autocreated C&M numeric variable
	SETTING_UP_AXIS	Tdiscrete	Autocreated C&M discrete variable
	POWER_ON	Tdiscrete	Autocreated C&M discrete variable
F	AVERAGE	Tdouble	Autocreated C&M numeric variable

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.

The screenshot shows the 'Set floating point variable' dialog box. The 'Dimension' dropdown menu is highlighted with an arrow. The 'Name' field contains 'AVERAGE' and the 'Tech ID' field also contains 'AVERAGE'. The 'Type' is set to 'Double (64-bit)'. The 'Description' is 'Autogenerated C&M numeric variable'. The 'Range' section shows 'Max, HH: 10000' and 'Min, LL: -10000'. The 'Alarm' section has 'HH-alarm', 'H-alarm', 'L-alarm', and 'LL-alarm' all unchecked. The 'Field access' section has 'Value modification', 'Substitute', 'HH - max', 'H - premax', 'L - premin', 'LL - min', and 'Scalable' all checked. The 'Publicity' dropdown is set to 'Published (network)'. The 'Substitute value' section has a 'Default value' of '0 (Physical)'. The 'Data acquisition' section has 'Acq type' set to a dropdown, 'Auto' unchecked, 'Max, counter' set to a dropdown, 'Trigger var' set to a dropdown, and 'Total' set to '0'. The 'Search' field is empty, and the '3 / 3' indicator is visible. The 'OK' button is highlighted with a green checkmark.

Change Units of the Average

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

The image shows two overlapping windows from a software application. The top window, titled "Edit properties of variables", contains a table with the following data:

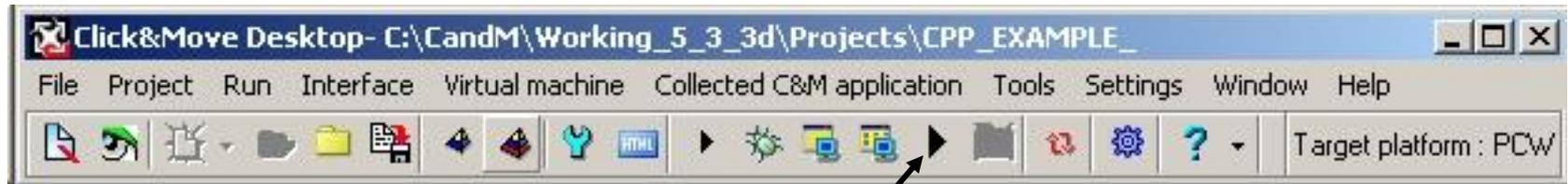
Name	Tech	Dimension	Min	Max
Position	Position	d	-100	100
AXIS1_ACT_POS	AXIS1_ACT_POS	rev	-100	100
AVERAGE	AVERAGE	rev	-10000	10000

The bottom window, titled "Variable modification statistics (todo list)", shows a list of modified variables:

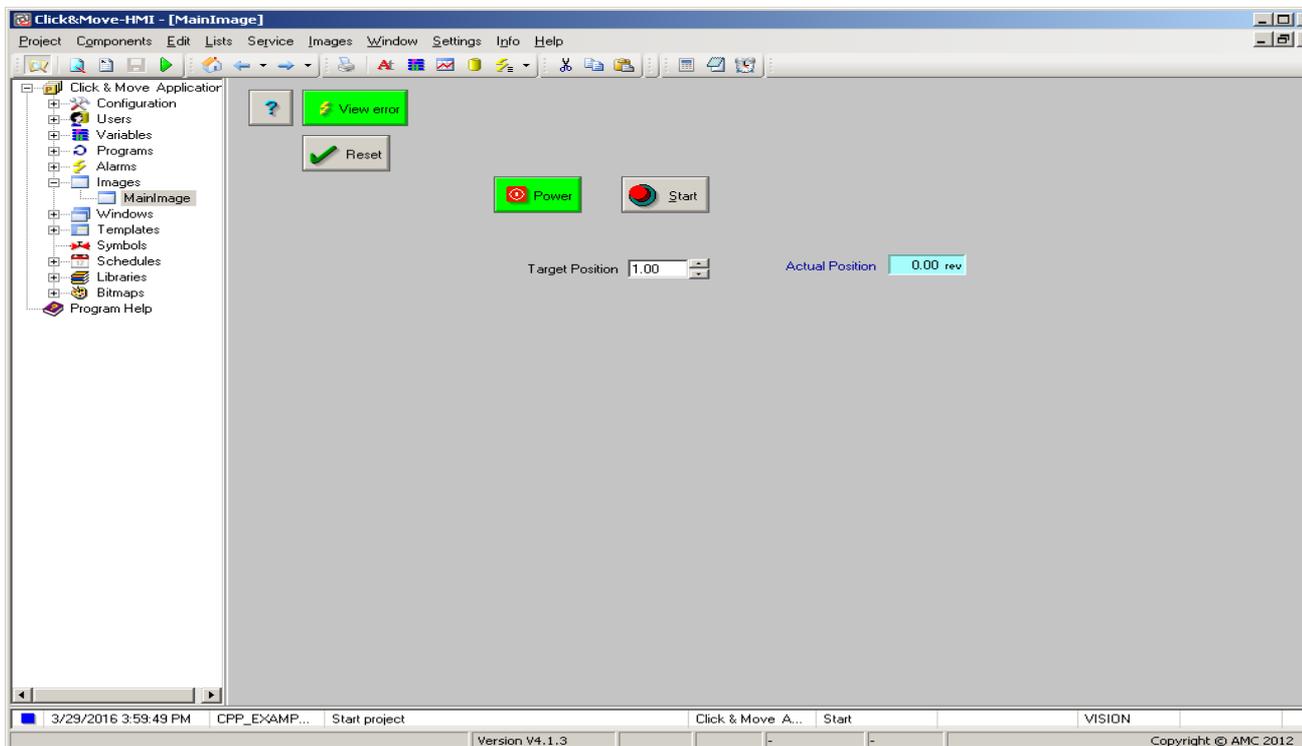
Modified variables
AVERAGE

At the bottom of this window, there is a checkbox labeled "Don't show this statistics again!", a "Clear" button, "Undo" and "Redo" buttons, a counter showing "1", and "Apply" and "Cancel" buttons.

Run the Project and update the HMI

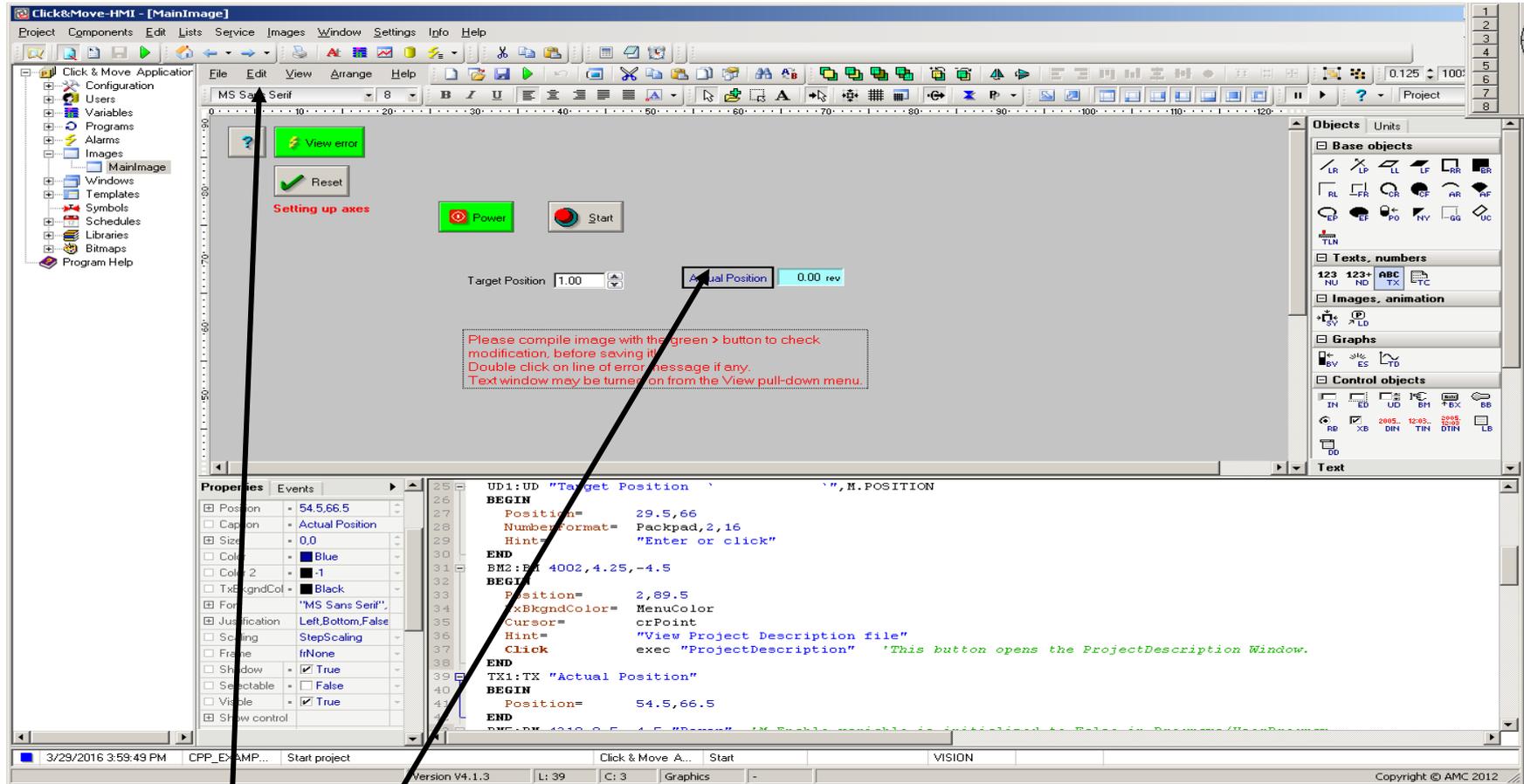


Launch the Motion, Virtual Axis, and the HMI with a click of the RUN ALL button.



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

Add Average Position display to the HMI

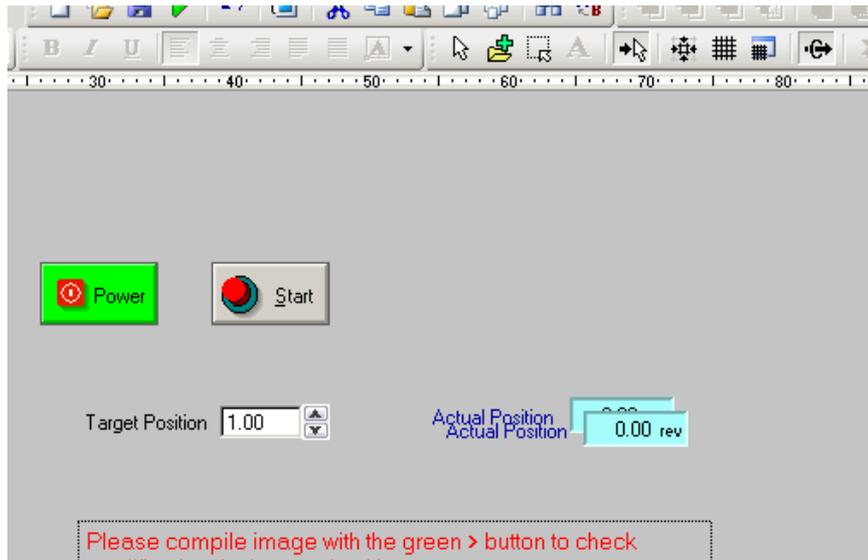


Click on the “Actual Position” text near the center of the screen. Then click “Edit” “Duplicate”

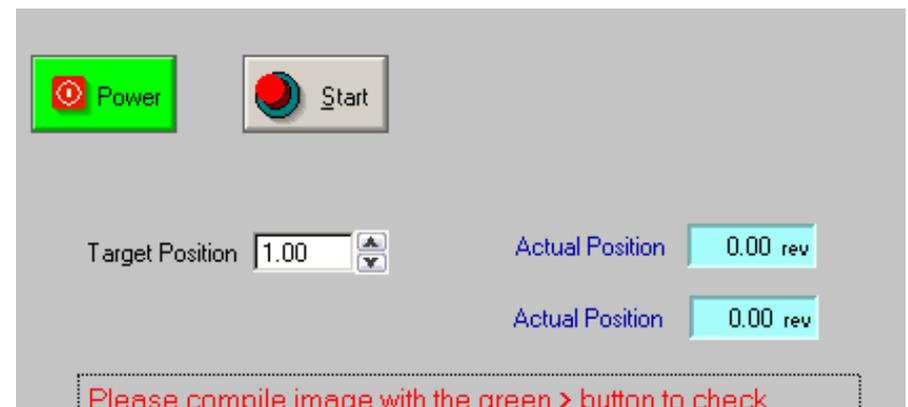
Do the same thing for the numeric display next to the text.

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

The screenshot displays the HMI development environment. At the top, there is a control panel with a 'Target Position' input field set to 1.00 and two 'Actual Position' display fields, both showing 0.00 rev. A red text box with a dashed border contains the following instructions: 'Please compile image with the green > button to check modification, before saving it! Double click on line of error message if any. Text window may be turned on from the View pull-down menu.' Below the control panel is the 'Properties' window, which is currently showing the 'Appearance' tab. The 'Caption' property is set to 'Actual Position'. To the right of the Properties window is the 'Events' window, which shows a list of ladder logic rungs. Rung 85 is selected, showing the following code:

```
85 TX3:TX "Actual Position"  
86 BEGIN  
87 Position= 54.375,62.25  
88 END
```

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

The screenshot shows an HMI design tool interface. At the top, there are three numeric displays: 'Target Position' with a value of 1.00, 'Actual Position' with a value of 0.00 rev, and 'Average Position' with a value of 0.00 rev. A red text box in the center contains the following instructions: 'Please compile image with the green > button to check modification, before saving it! Double click on line of error message if any. Text window may be turned on from the View pull-down menu.' Below the main design area, there is a 'Properties' window for an object named 'NU2'. The 'Assign' section shows the variable 'M.AXIS1_ACT_POS'. The 'Appearance' section shows 'Position' as 65.25,62.125, 'Frame' as frRecess+njCenter+dmRight+dmSmall, 'Caption' as empty, 'Size' as 0,0, and 'Color' as Black. To the right of the properties window is a code editor showing ladder logic for two text boxes: TX2:TX 'Setting up axes' and TX3:TX 'Average Position'. The code for TX2 includes settings for Position, Color (BlinkLightRed), Font ('MS Sans Serif', 8, Bc), and Visible (M.SETTING_UP_AXIS). The code for TX3 includes a Position setting of 53.25,62.25. At the bottom of the code editor, there is a line: NU2:NU M.AXIS1_ACT_POS,frRecess+njCer.

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”

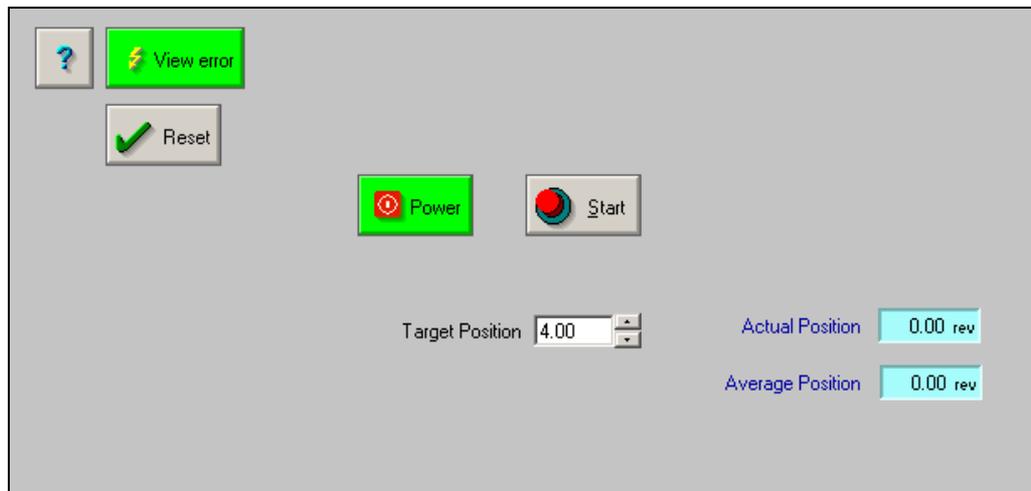
The screenshot shows the 'Variable list and select - All' dialog box. The 'Variables (var)' tree on the left is expanded to 'Formal variables'. The main table lists variables with the following data:

Tp	Name	Mean	Connection
123	M.START	Autocreated C&M discrete variable	
F	M.POSITION	Autocreated C&M numeric variable	
123	M.ENABLE	Autocreated C&M discrete variable	
123	M.RESET	Autocreated C&M discrete variable	
123	M.FB_ERROR	Autocreated C&M discrete variable	
123	M.FB_ERROR_ID	Autocreated C&M numeric variable	
ABC	M.FB_ERROR_LOCATI...	Autocreated C&M string-variable	
F	M.AXIS1_ACT_POS	Autocreated C&M numeric variable	
123	M.SETTING_UP_AXIS	Autocreated C&M discrete variable	
123	M.POWER_ON	Autocreated C&M discrete variable	
F	M.AVERAGE	Autocreated C&M numeric variable	

The 'M.AVERAGE' row is highlighted in blue. Below the table, there are radio buttons for 'Default (Value)', 'Attr', 'Dim', 'Value', 'Min', 'Mean', '0', 'Max', 'Info', '1', 'L', 'Iden', '2', 'H', 'Tech', '3', 'ID', 'Lo', 'Hi', and a checkbox for 'Show all variables'. At the bottom, there is a 'Select variable(s):' field containing 'M.AVERAGE' and an 'Apply' button with a green checkmark.

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 lags the actual position by an amount set by the filter Weight and the TakeData cycle time.

END OF PRESENTATION

C++ Function Block Project