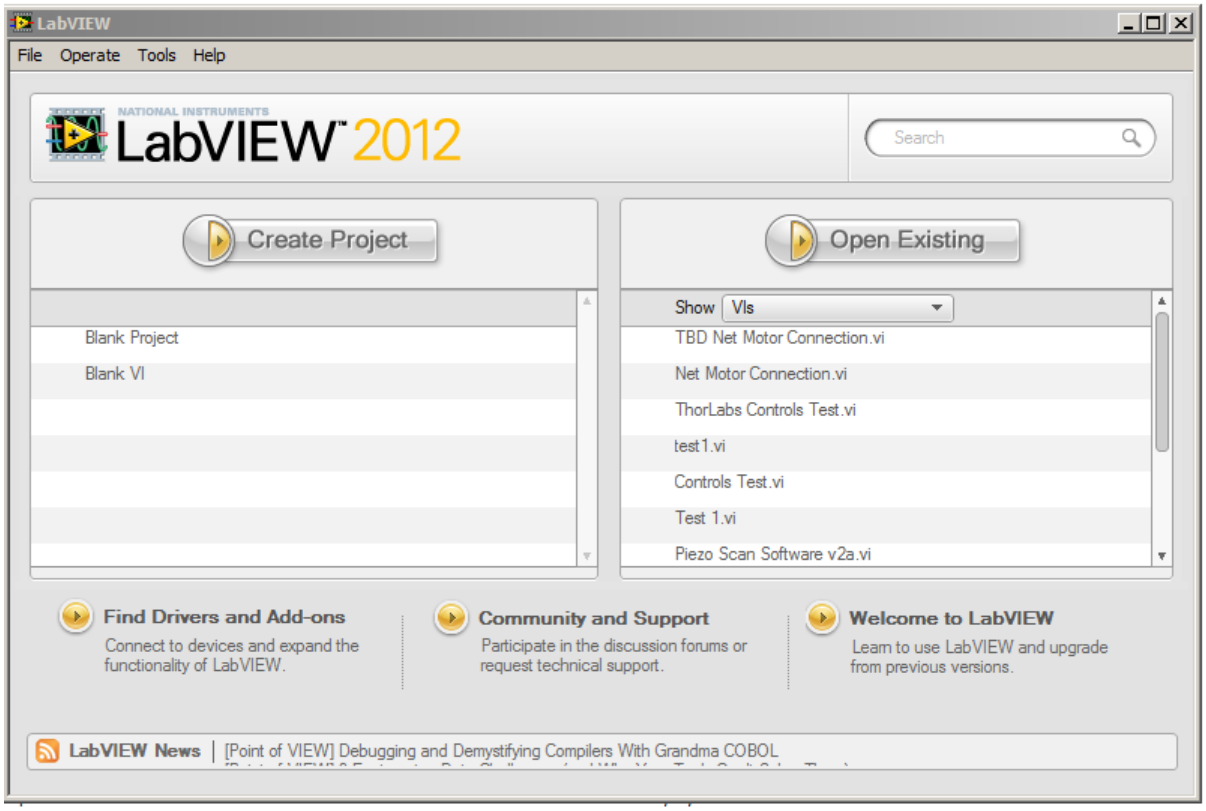
Kinesis LabView Guide

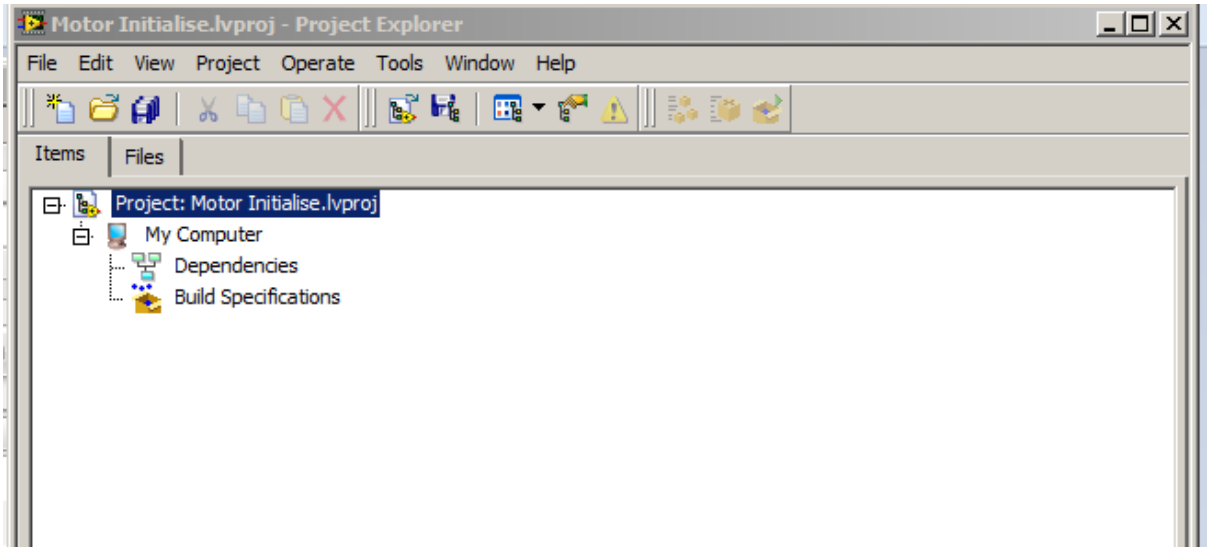
Table of Contents

Creating the Kinesis LabView Project File and Folder

1) On the LabView introduction screen select “Blank Project”

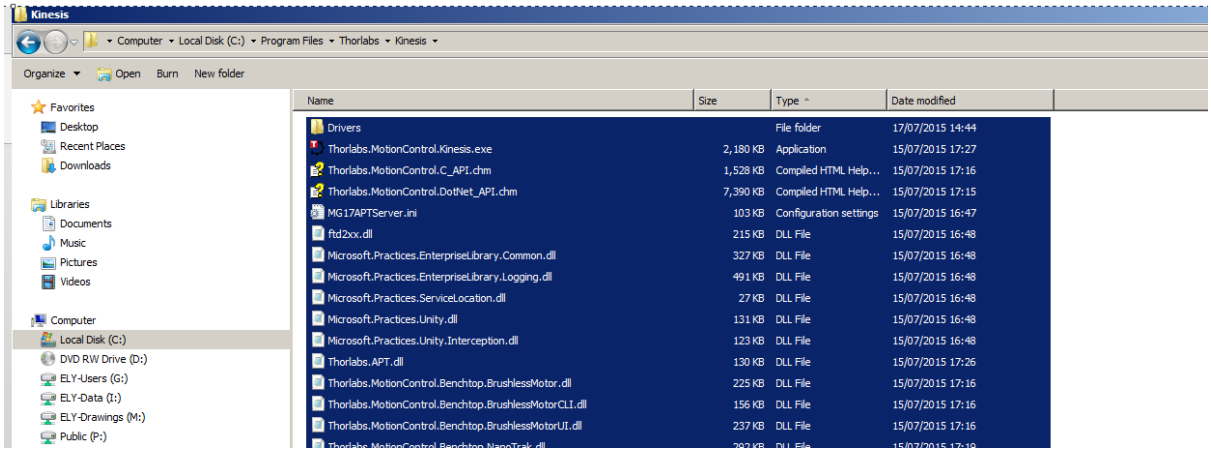


2) Save the Project in a New Folder, rename the folder to a suitable name. A new folder is needed as the Kinesis .dll files need to be copied into this folder. Once the folder is created save the LabView project in this folder, again with a suitable project name.

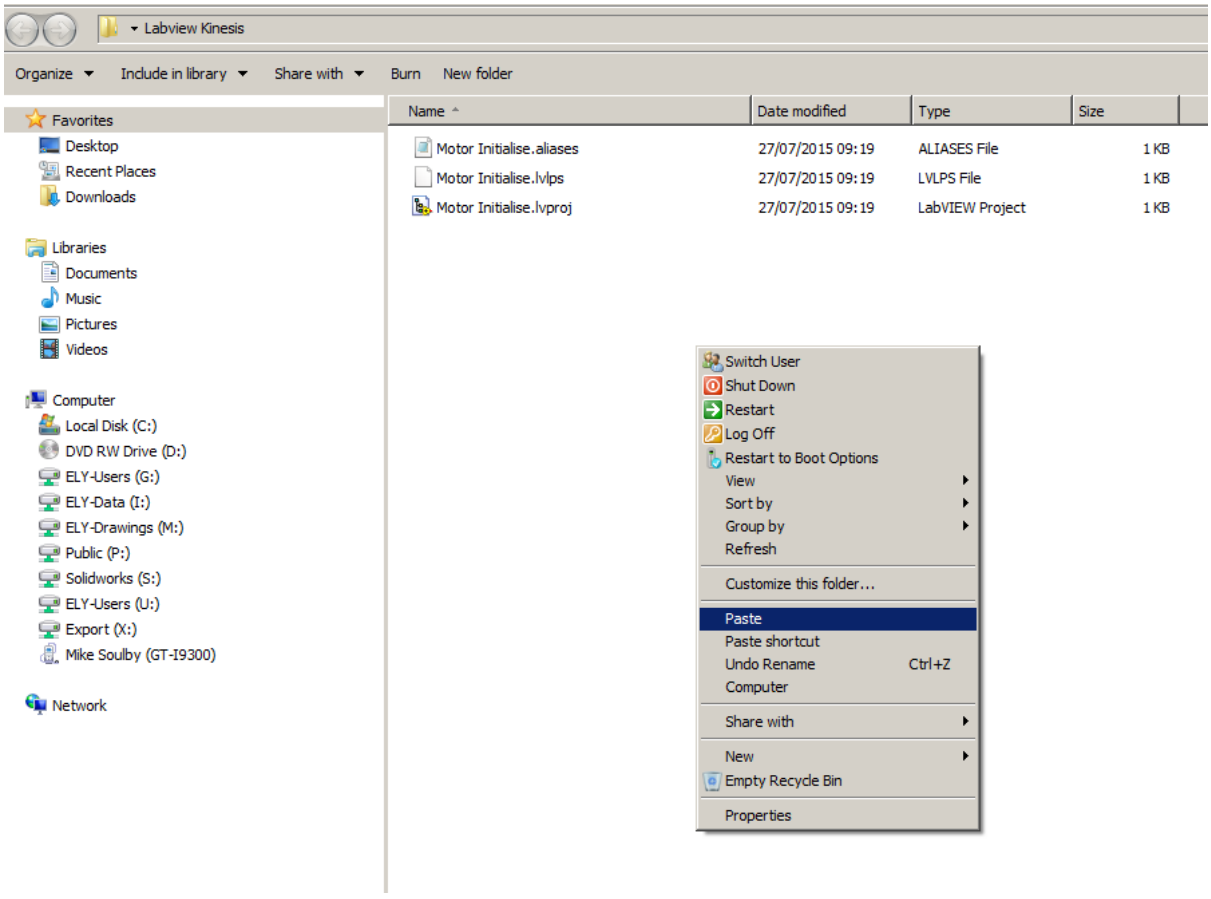


3) The Kinesis .dll files now need to be copied to this new project folder. Navigate to the Kinesis installation folder using windows explorer. The default directory for the Thorlabs Kinesis installation is C:\Program Files\Thorlabs\Kinesis

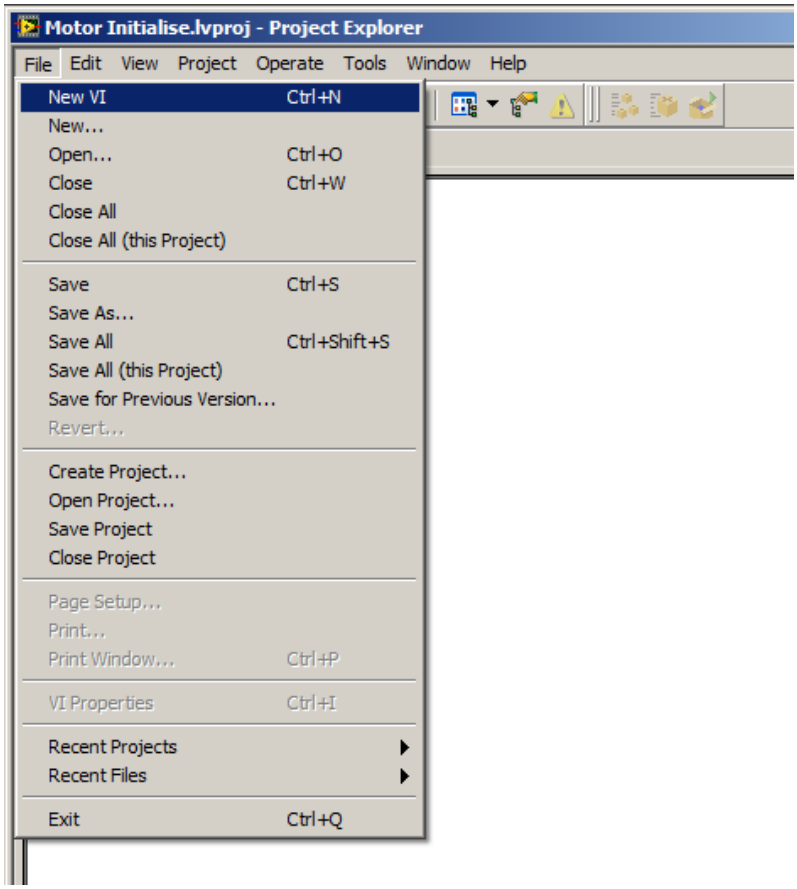
4) In this folder the easiest way to ensure all the required files are copied to the project folder is to simply select all files using Ctrl + A, then copy the files using Ctrl + C.



5) Now navigate back to the LabView project folder created in step (2) and paste the Kinesis files into here, using Ctrl + V.



6) Once the files have been copied, back in the LabView project window select File>New VI. This will create a new Virtual Instrument that will allow you to begin writing your own application.



7) The new VI will open up a new Front Panel and corresponding Block Diagram window.

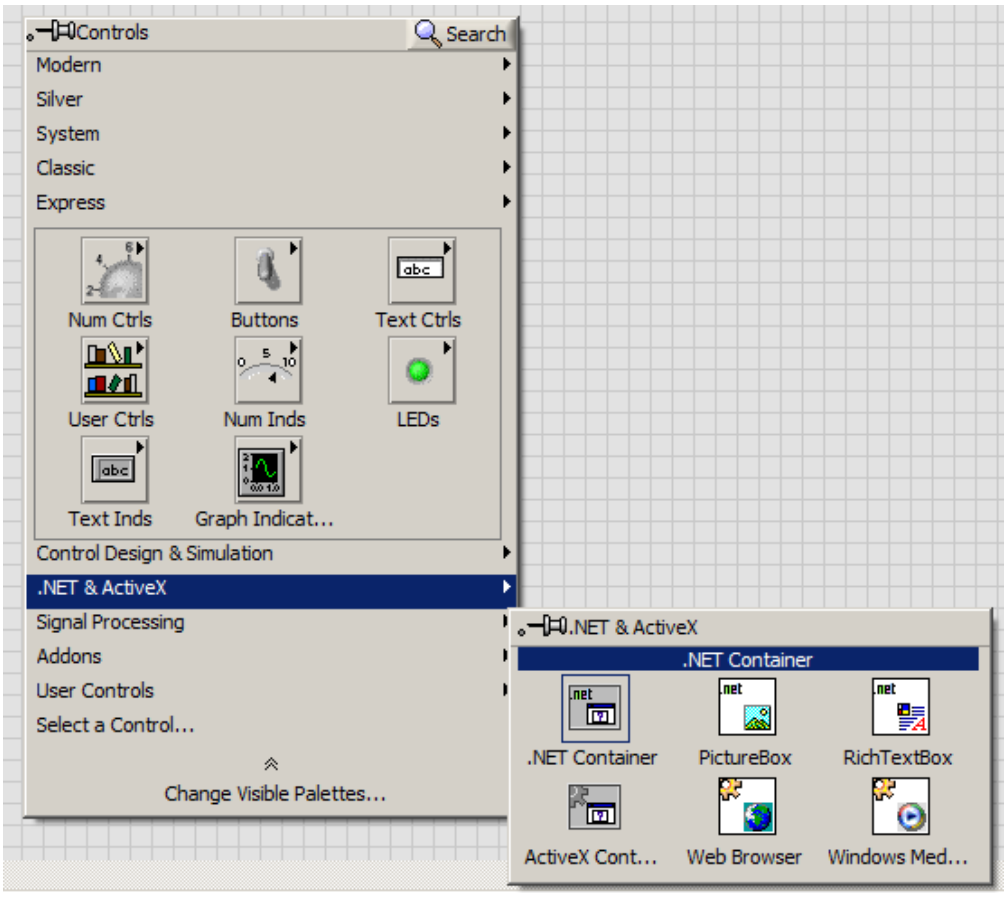
Adding an .NET control to the front panel

LabView provides many standard controls and also has the ability to host third party controls through mechanisms such as .NET.

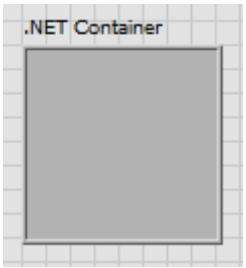
Kinesis software is exposed through .NET to allow users to incorporate hardware control through their own custom applications.

Complete the following steps to add a Kinesis Motor Control to the front panel.

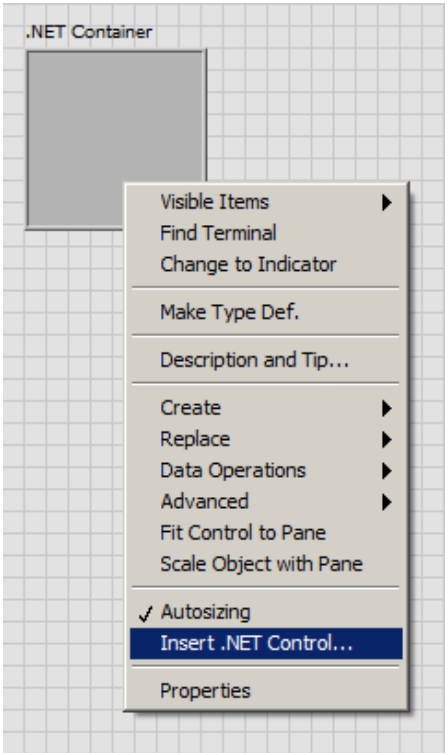
1) Expand the Controls palette, then select the .NET & ActiveX palette. If the Controls palette is not visible, select View/Controls Palette.



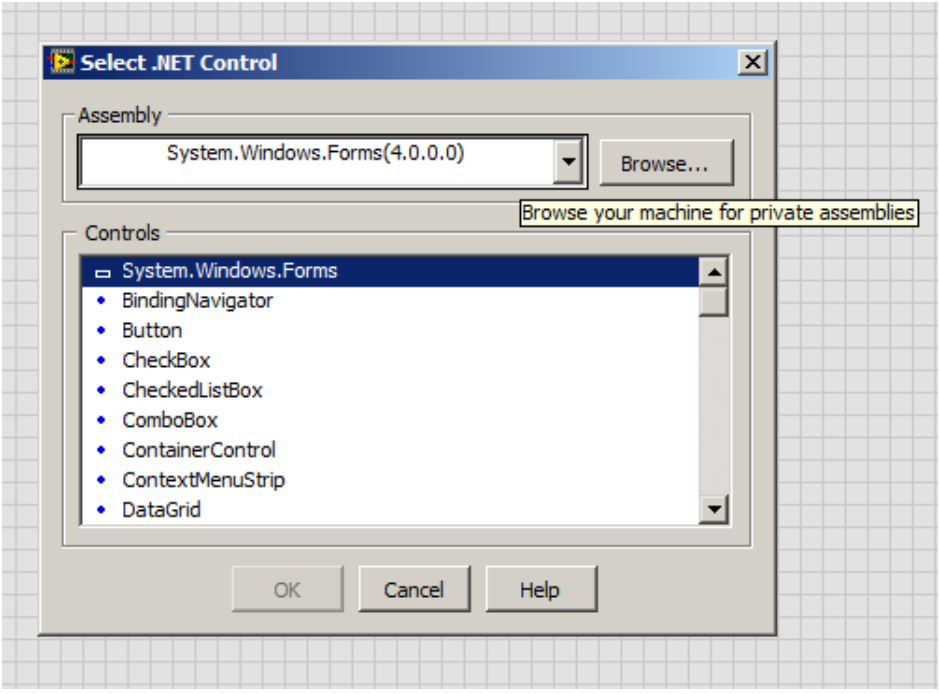
2) Select the .NET Container to attach it to the cursor, and then place the control on the front panel. Notice at this stage the container is empty.



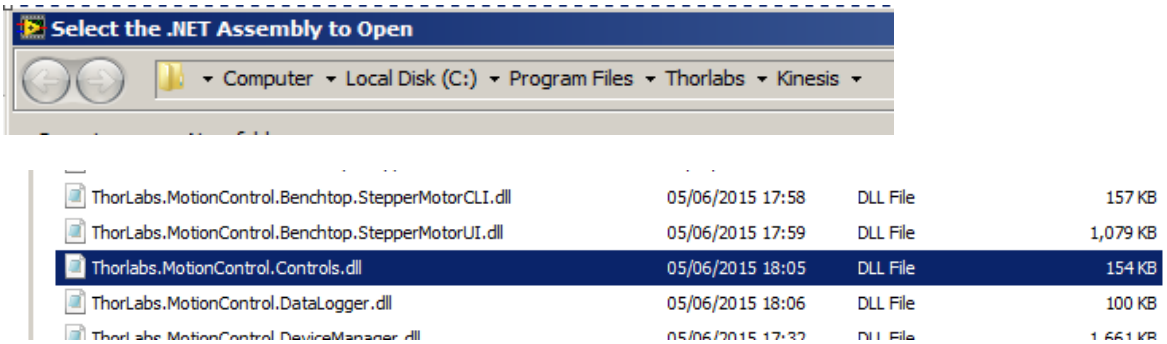
3) Right click on the centre of the .NET Container and select from the shortcut menu “Insert .NET Control”.



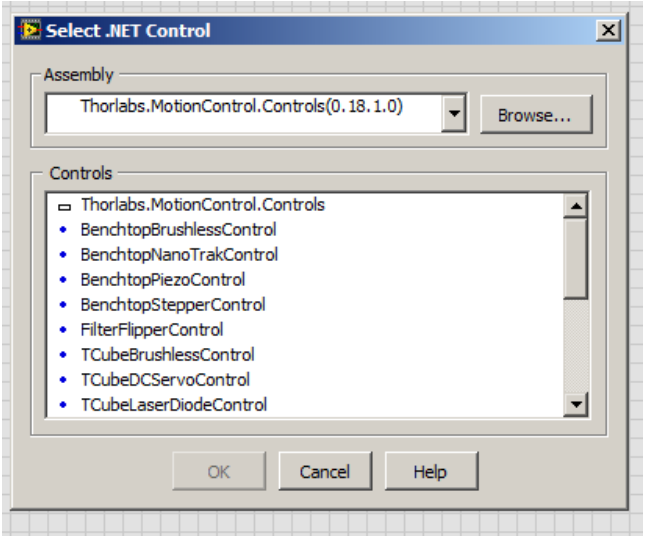
4) From the dialog window that opens, select Browse… to manually search for the Kinesis .NET control assembly.



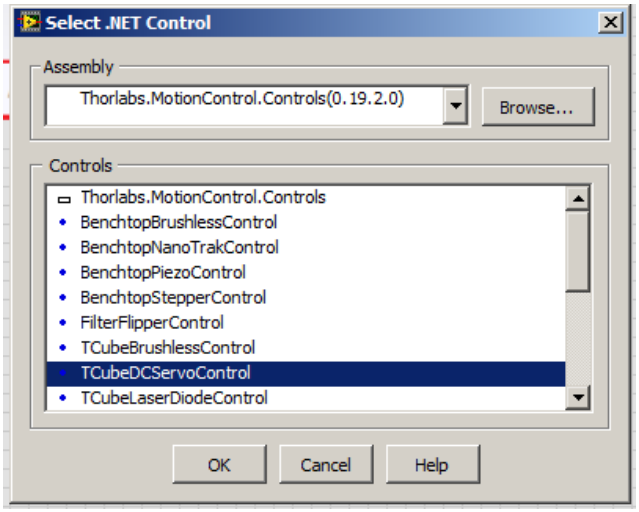
5) From the LabView Project folder where the controls and DLLs are copied select the file **Thorlabs.MotionControl.Controls.dll**



6) This will load all of the compatible Kinesis controls

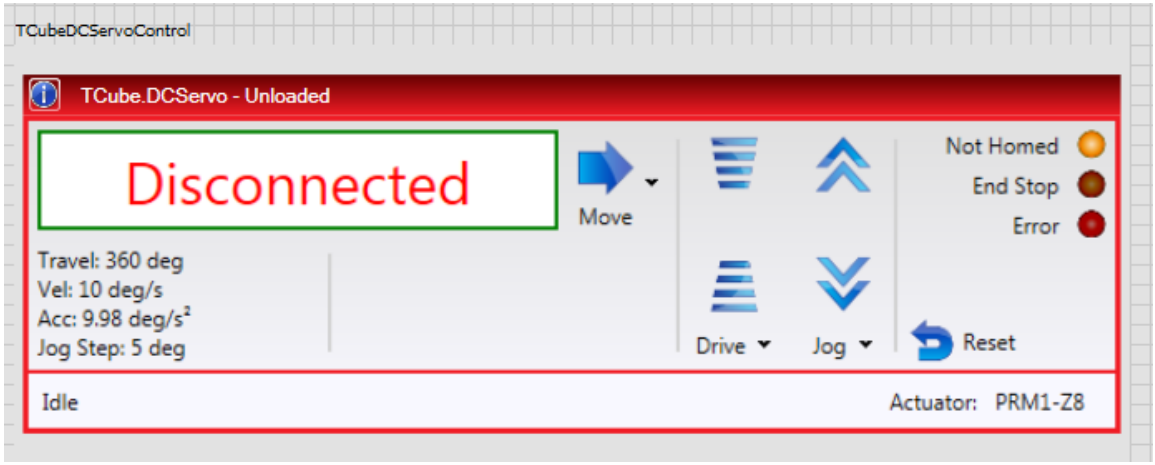


7) From the list displayed, select the control type applicable to the hardware unit you want to load. For example, to insert a T-Cube DC Motor Control object select **TCubeDCServoControl.** Similarly, to insert a Benchtop Brushless DC Motor Control Object, select **BenchtopBrushlessControl.**



Note: In LabView the **TCubeDCServoControl** control represents the .NET control used to interface with DC servo motor controller hardware. Other hardware types have their own .NET controls as shown in the list above.

The .NET container should now contain the Kinesis motor control. Resize and position the control as required as shown.



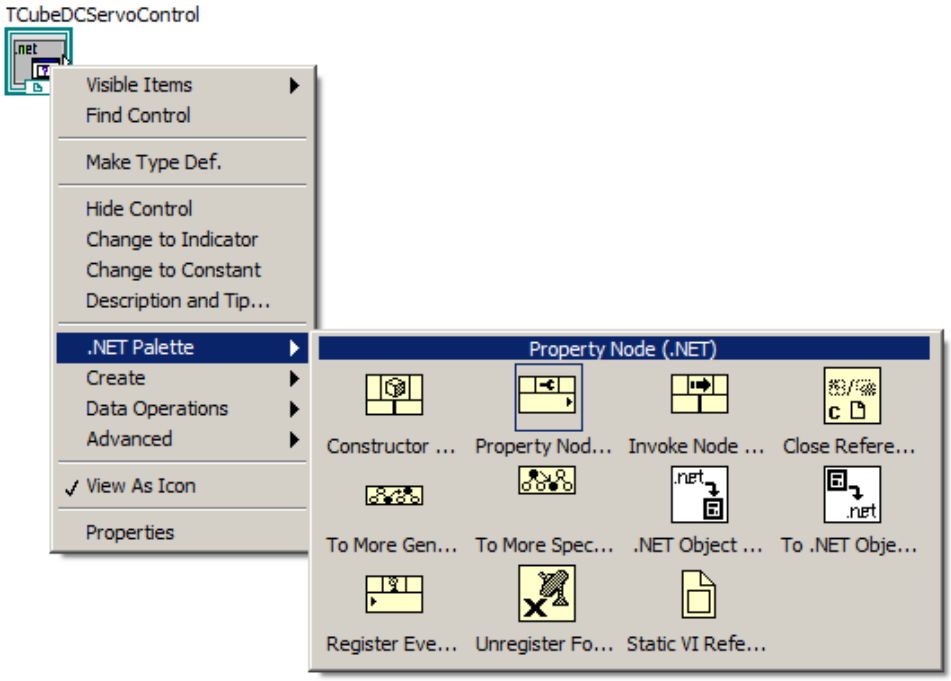
Calling a .NET Property

Before the control can run the correct serial number of the controller needs to be set to the >NET control object. Complete the following steps to set the serial number property of the Kinesis motor control object.

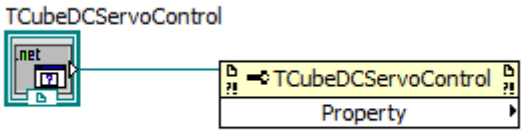
1) Select the Block Diagram window. If not visible, select Window/Show Block Diagram.

2) Display the .NET palette as shown by right clicking the TCubeDCServoControl object and selecting .NET Palette from the drop down list.

3) Select the Property Node icon to attach it to the cursor. Then drop the node onto the block diagram.

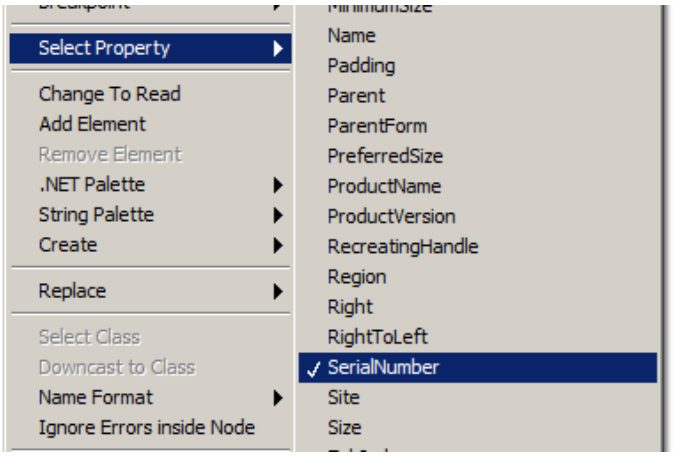


4) Wire the Kinesis control object terminal in the block diagram to the input reference connector of the property node by first clicking on the Kinesis control output connection to begin wiring. Click on the input ref connection of the property node to complete the wiring.



5) In wiring the property node to the Kinesis control object terminal the property node now has information available as to what properties are available.

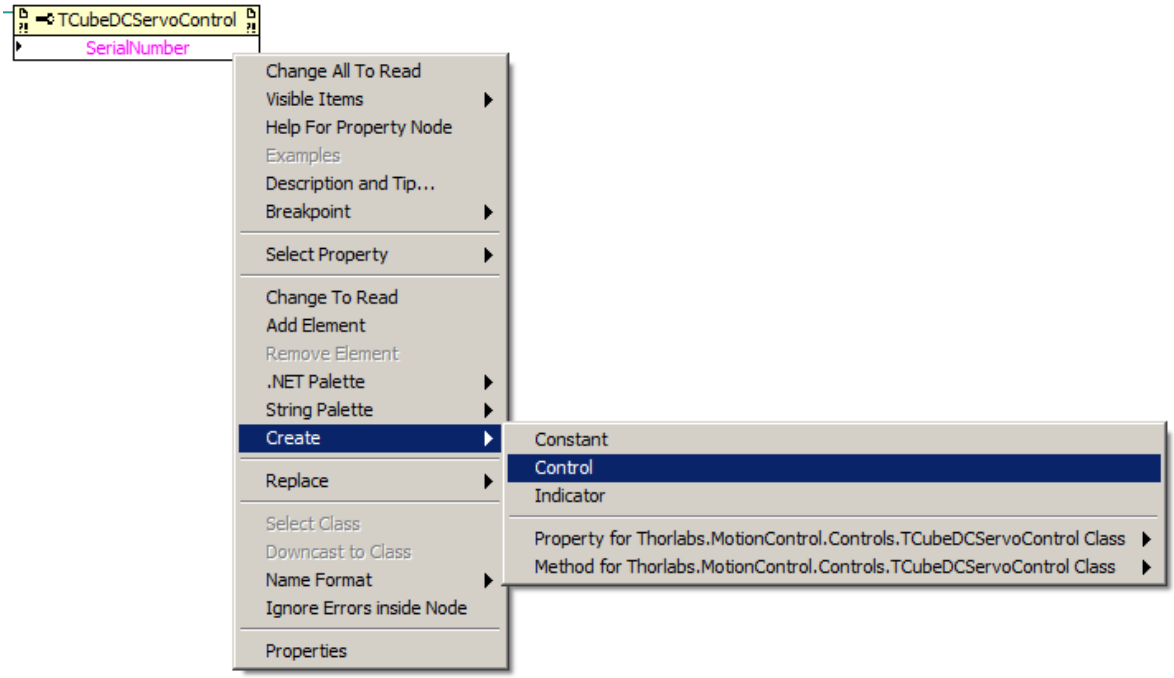
6) Right click the white area of the property node which by default displays the word property. In the shortcut menu choose **Select Property**, this will lists all of the available properties for the Kinesis Motor Control. Select the **SerialNumber** property from the list.

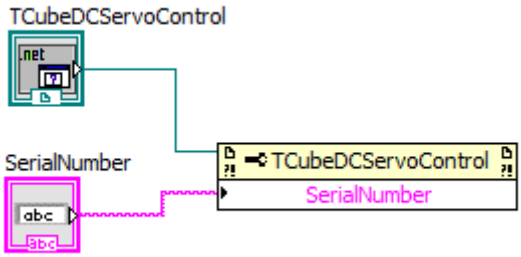


7) The property node now automatically displays the correct property node name and connectors.

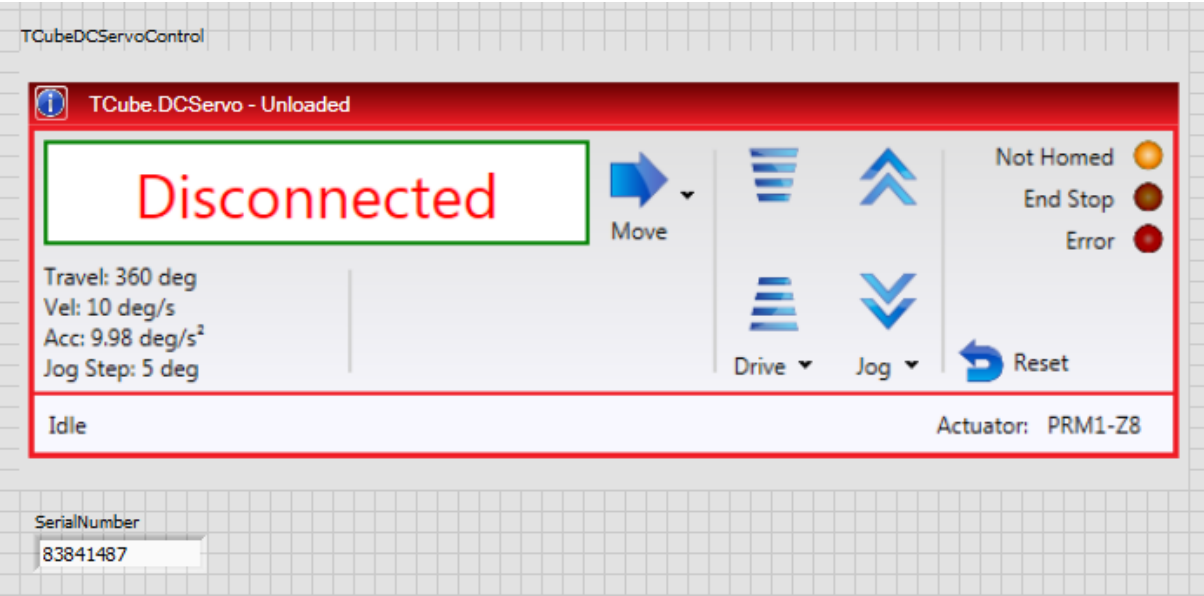
8) By default the property node parameter is set to read the current value. In this instance we would like to set the serial number. Right click the white area of the property node again and select **Change All To Write** from the shortcut menu.

9) To set the property, data needs to be connected to the input of the property node. Right click the input of the property node and select from the shortcut menu displayed Create / Control. A control, of the correct data type, is created and wired automatically.

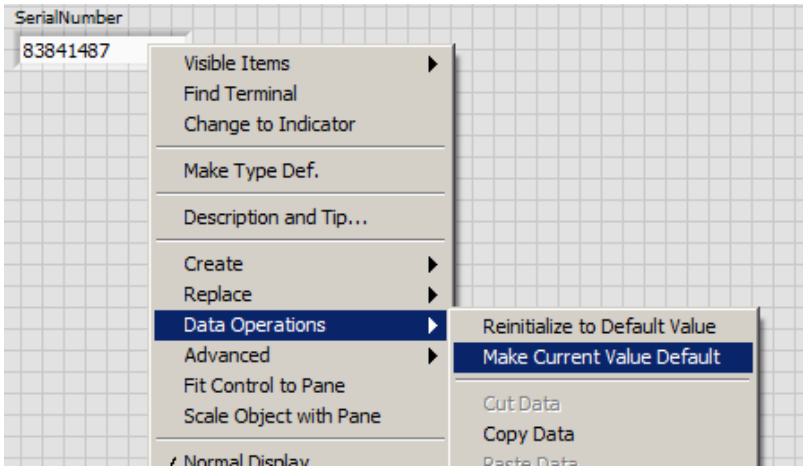




10) A control where the serial number of the controller can be entered will now be visible on the front panel. To go to the front panel select Window / Show Front Panel. The control can be repositioned to a suitable position. The serial number of the controller should now be entered.



11) To make this serial number the default value of this control, right click the control and select Data Operations / Make Current Value Default.

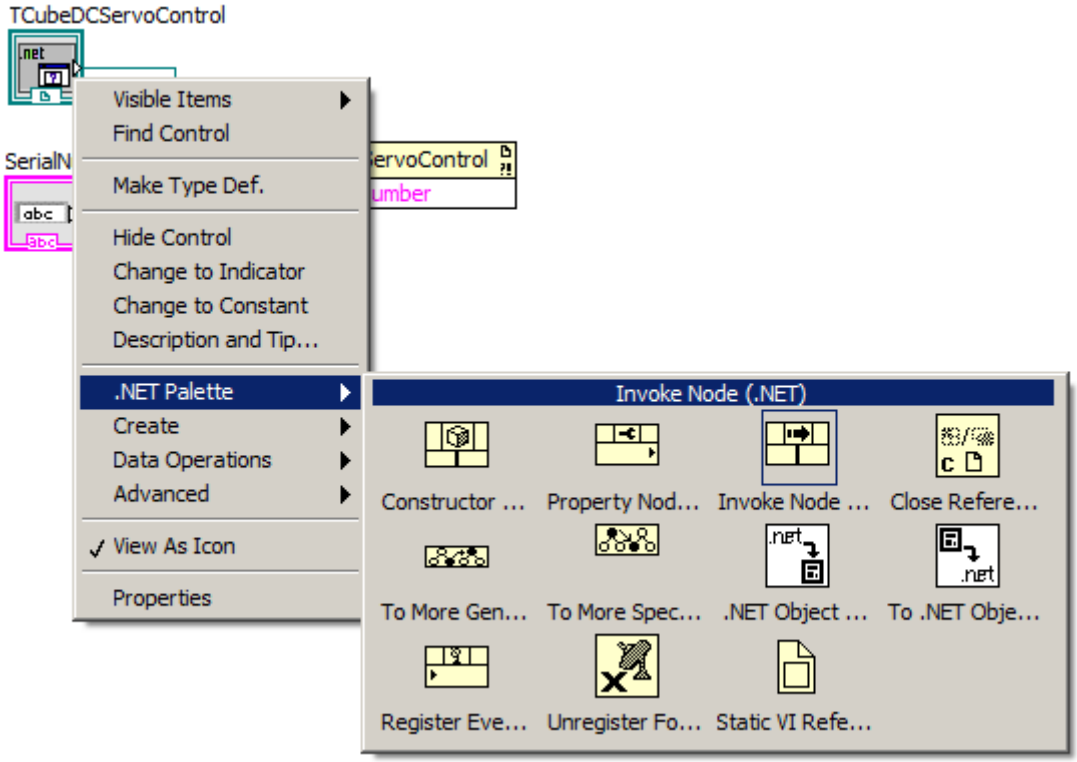


Calling a .NET Method

Methods of the Kinesis control object can be called within the block diagram by using a .NET Invoke Node. In this example we will call the **CreateDevice** method which starts communication with connected Kinesis hardware. Complete the following steps to call the **CreateDevice** method of the Kinesis control object.

1) Select the Block Diagram window. If not visible, select Window/Show Block Diagram.

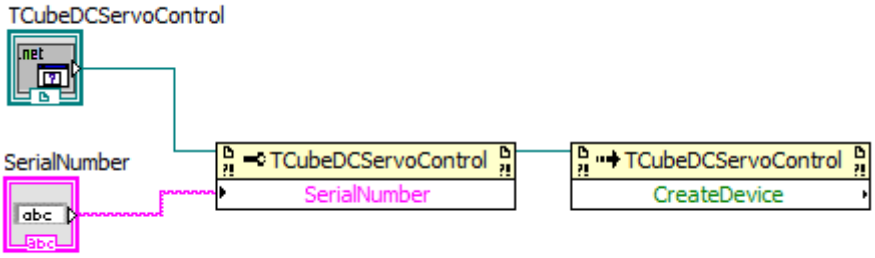
2) Display the .NET palette as shown by right clicking the TCubeDCServoControl object and selecting .NET Palette from the drop down list.



3) Select the Invoke Node icon to attach it to the cursor. Then drop the node onto the block diagram. To ensure that the serial number is set prior to calling the start control method wire the output reference of the previous SerialNumber property node to the input reference of the CreateDevice invoke node. This reference is the same reference as the original Kinesis control object, however using the output side of the property node ensures that the serial number will be set before calling the CreateDevice method.

4) Click the reference output of the property node to begin wiring. Click the reference input of the invoke node to complete the wiring. In wiring the invoke node to the Kinesis control object reference the invoke node now has information available as to what methods are available.

5) Right click the white area of the invoke node which by default displays the word method. In the shortcut menu displayed select **Methods**. The available methods are presented in a further shortcut menu. Select the CreateDevice method from the list. The invoke node now automatically displays the correct method name. This particular method requires no further parameters.



The program would be able to run in its current state however there would be no way to stop the control. Therefore we need to add a similar method and some execution control to prevent the control stopping prematurely and also to terminate communication to the controller prior to the program finishing.

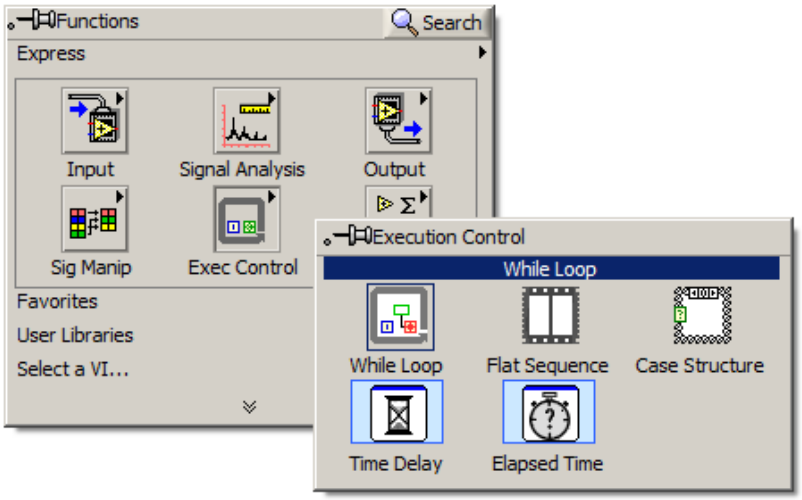
To control the execution control of the VI programming execution structures will need to be added to the block diagram. These help control the order in which LabView function, methods and properties are called, and they can also help prevent the program stopping prematurely.

In this case a **While Loop** should be added to the block diagram. A while loop will repeat any section of code enclosed in the While Loop box until the stop condition is satisfied, at which point the While Loop will complete and the program proceed to the next step.

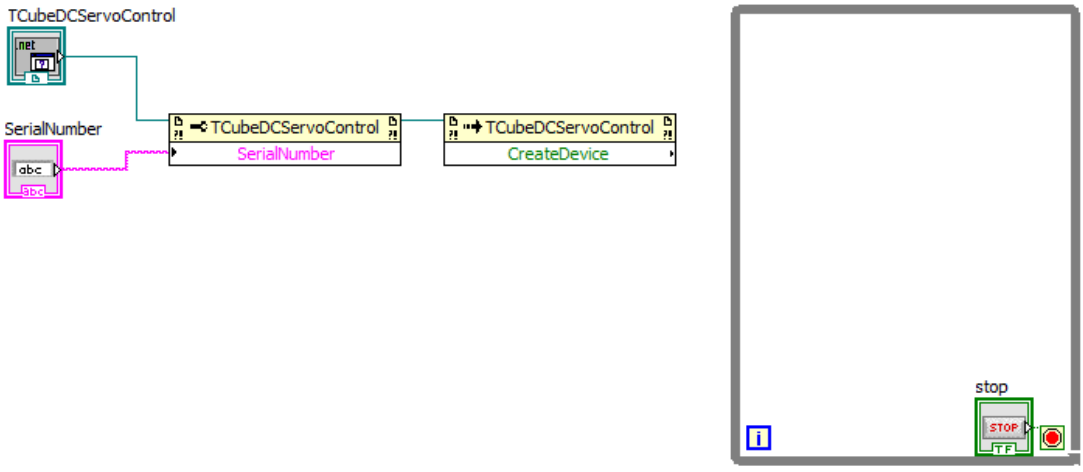
To Insert a While Loop to the block diagram complete the following steps.

1) If the block diagram is not visible select from the front panel menu bar Window / Show Block Diagram.

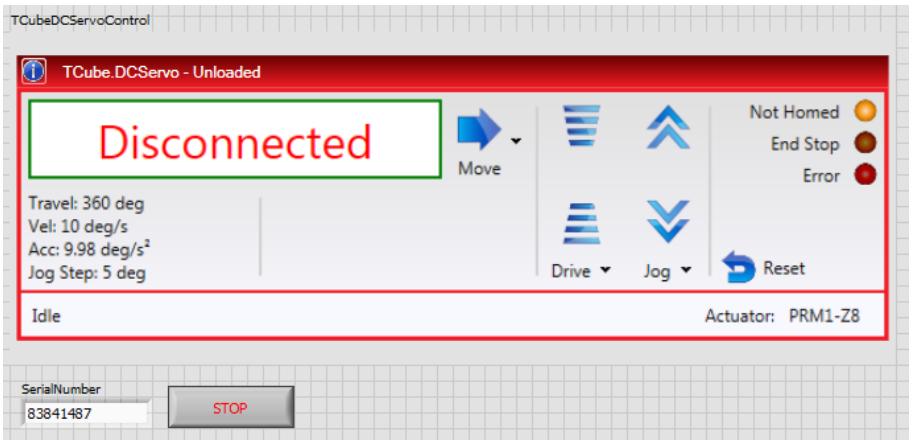
2) Right click the block diagram workspace to view the Functions Palette. To select a While Loop select the Execution Control menu, from the subsequent menu select While Loop.



3) Draw a rectangle onto the block diagram to the right of the CreateDevice method icon, this will create a while loop in the block diagram as shown.



4) A corresponding STOP button to stop the While Loop Will now be displayed on the front panel, which can be repositioned if necessary.



To stop communication with the hardware after the While Loop has finished the VI needs to tell the hardware to stop. To do this complete the following step to at the **CloseDevice** method to the VI.

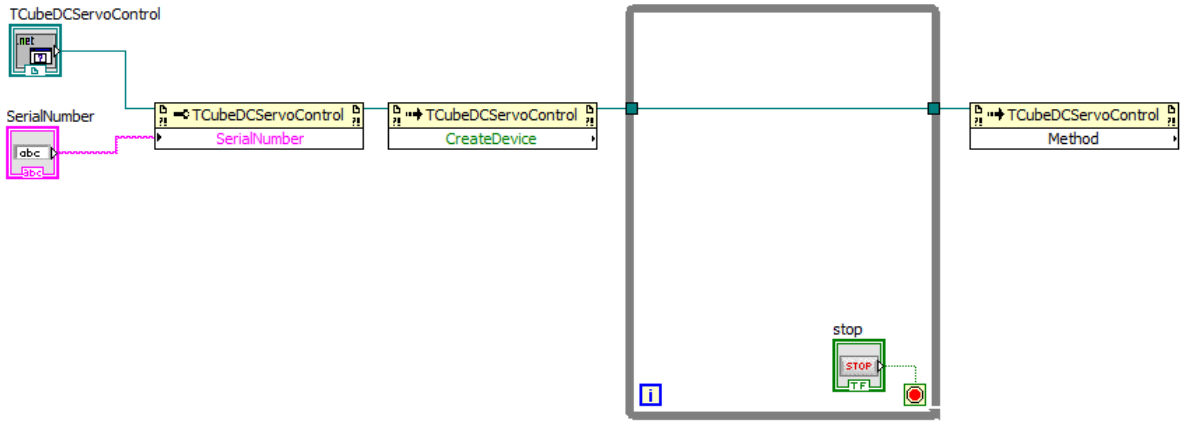
1) Display the .NET palette as shown by right clicking the TCubeDCServoControl object and selecting .NET Palette from the drop down list.

2) Select the Invoke Node icon to attach it to the cursor. Then drop the node onto the block diagram.

3) Click the reference output connector of the StartCtrl invoke node to begin wiring. Click the left hand edge of the while loop to complete the wiring segment.

4) Click the connector block just created on the left hand edge of the while loop to begin wiring again. Click the right hand edge of the while loop to complete the wiring segment.

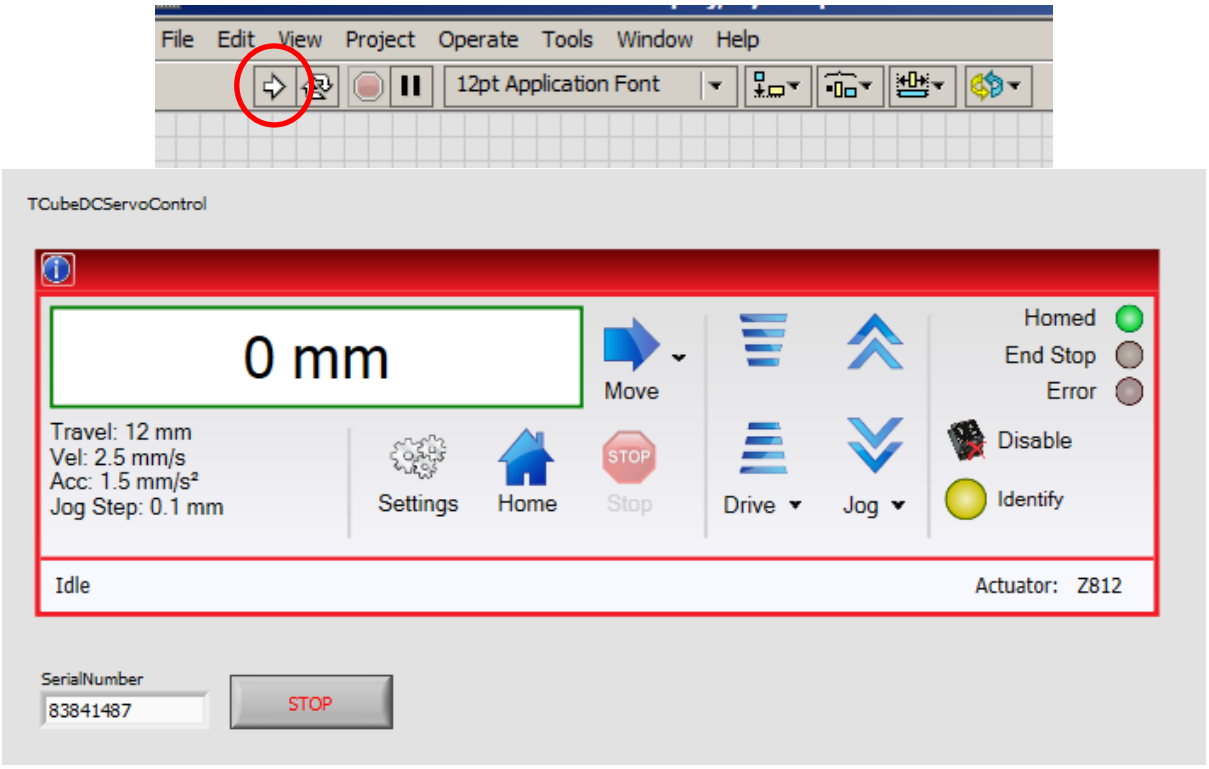
5) Click the connector block just created on the right hand edge of the while loop to begin wiring again. Click the reference input connector on the second invoke node. The wiring is shown below.



6) Right click the white area of the second invoke node which by default displays the word method. In the shortcut menu displayed select Methods. The available methods are presented in a further shortcut menu. Select the **CloseDevice** method from the list.

Note: The reference of the second invoke node enters and exits the while loop. Because the while loop does not terminate until the stop button is pressed, the second invoke node inputs cannot be satisfied and therefore will not execute until the loop is exited.

7) Click File/Save to save the changes.



Locating the .NET Method List

The .Net API guide is Thorlabs.MotionControl.DotNet_API.chm which can be found within the install folder (e.g C:\Program Files\Thorlabs\Kinesis for 64-bit install)

The list of methods available for each device can be found within the specific dependencies list (required DLLs) utilised by a specific device. As the Kinesis MotionControl libraries is a hierarchical structure, there is not a single list containing all functions however each dependency lists only the available methods which it supports. For example, for BBD201 these functions are collected under the following..

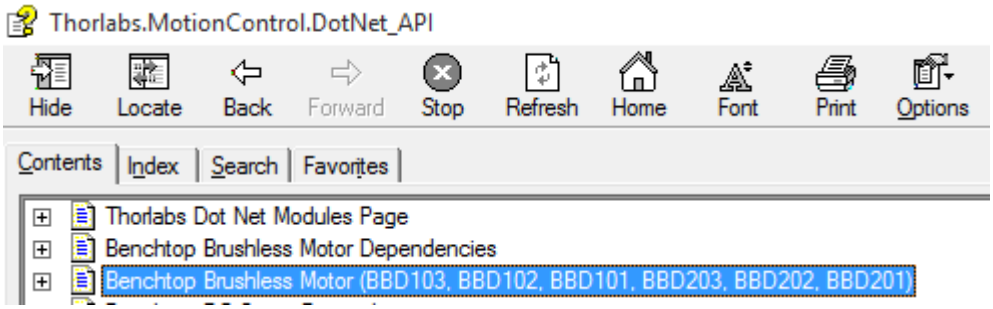
Thorlabs.MotionControl.Benchtop.BrushlessMotorCLI

Thorlabs.MotionControl.Benchtop.BrushlessMotorUI

Thorlabs.MotionControl.GenericMotorCLI

Thorlabs.MotionControl.DeviceManagerCLI

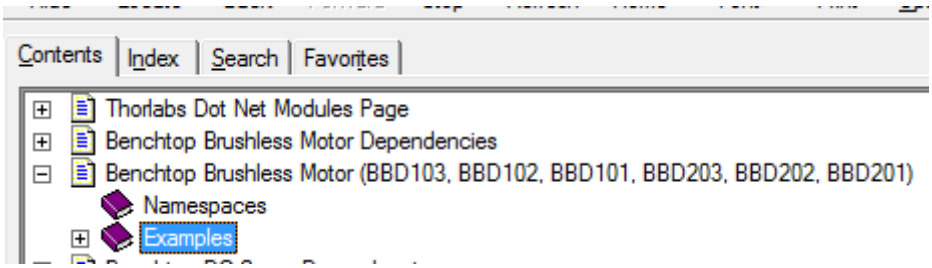
This list can be found in the Contents area of the help file as shown below.



Clicking through Thorlabs.MotionControl.Benchtop.BrushlessMotorCLI module, this holds the method GetChannel which returns a BrushlessMotorChannel. The BrushlessMotorChannel has a list of methods which can be applied to the channel.

An overview of how to use these methods can be found in the examples area.

Please note that these are C# examples, however the same principle can be used to program using LabView.

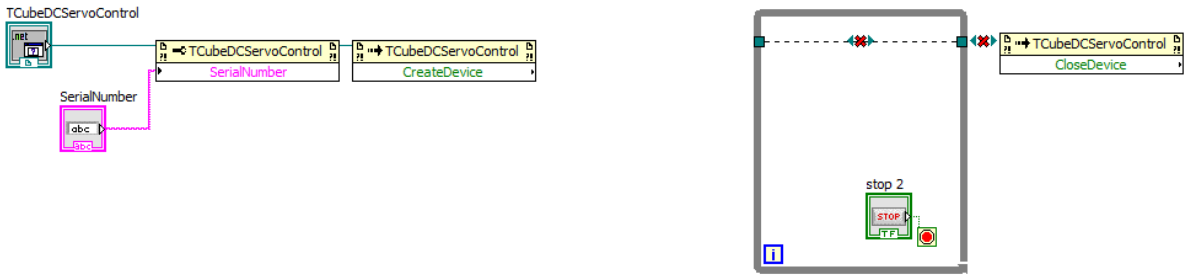


Refer to the dependency list for your specific device to find out which DLLs need to be copied to their output folder.

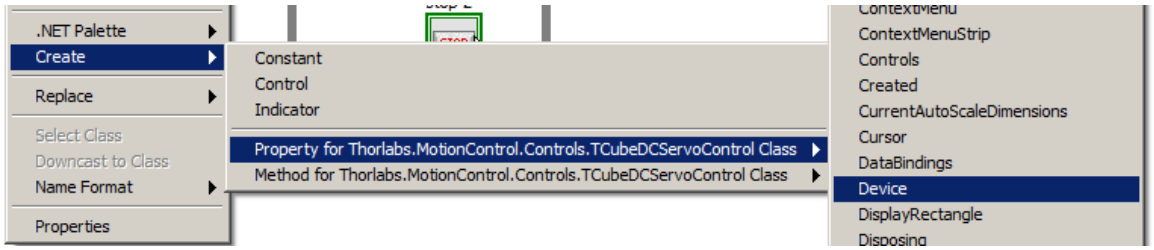
Querying the Device Position

A common application is to query to position of the motor. In order to use the motor methods the .NET property **Device** needs to be included into the block diagram. The **Device** property provides access to functions specific to the motor, i.e. Home, Move, GetStatus, etc. The following steps will allow you to insert this property.

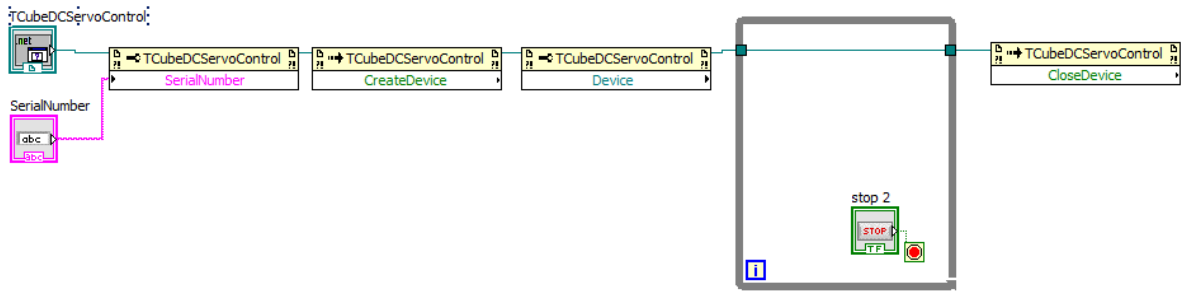
1) Using the VI created in the previous sections of this document, delete the wire between the CreateDevice property and the while loop, this is done by simply selecting the wire and pressing delete.



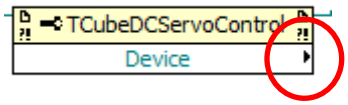
2) The property **Device** will now need adding. Right click on the CreateDevice property node and select **Create** / **Property for Thorlabs.MotionControl.Controls.TCubeDCServoControl Class** from the subsequent drop down menu select **Device** and insert this onto the block diagram between the CreateControl property and the While Loop.



3) Rewire the output terminal of the CreateDevice property node to the Device input property node. Then wire the output of the Device property node to the existing connection on the While Loop as shown.

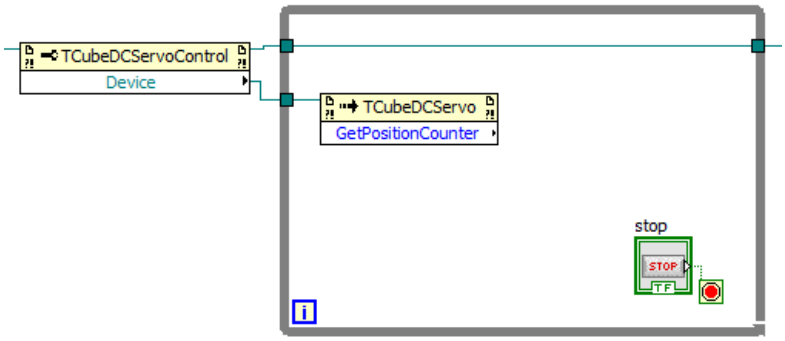


4) To create a method to call a motor function right click on the output arrow of the Device property node (circled) and select
Create / Method for Thorlabs.MotionControl.TCube.DCServoCLI.TCubeDCServo Class



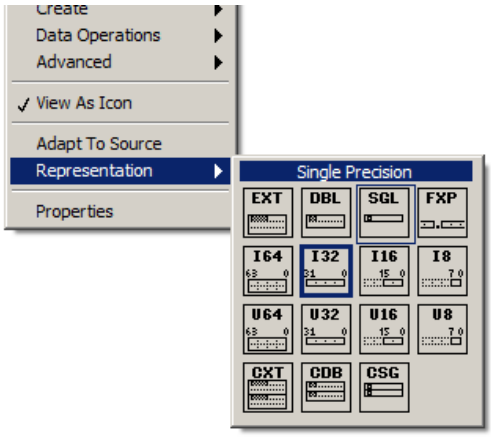
5) From the method drop down menu select the method **GetPositionCounter**. This will create an Invoke node with the selected method. Insert this GetPositionCounter method into the While Loop.

6) Wire the output arrow of the Device property node into the While Loop and onto the input reference connector of the GetPostionCounter method as shown.



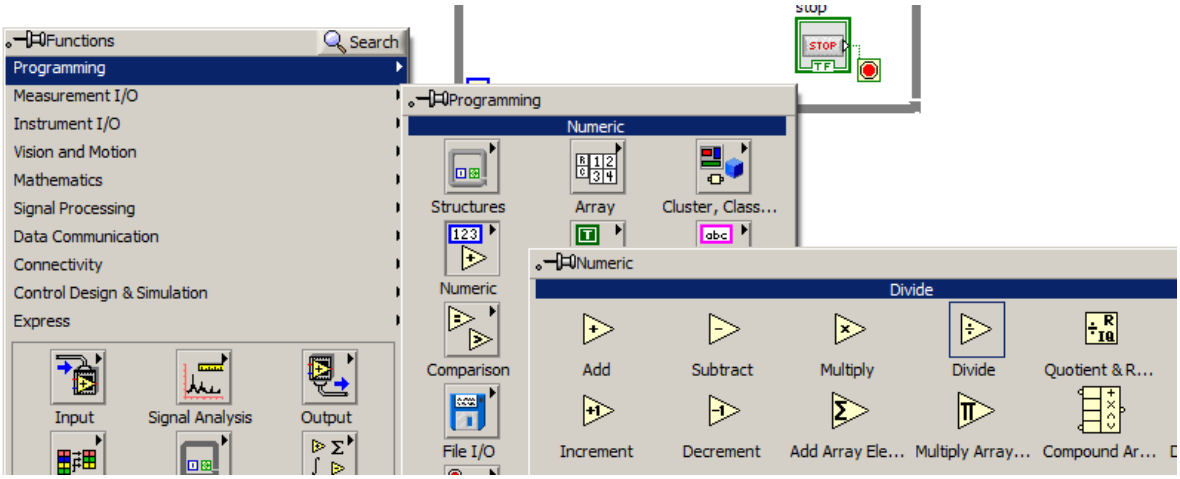
7) To display the position on the front panel Right click the output connector of the method GetPositionCounter and select from the shortcut menu displayed Create / Indictor. An indicator will be created on the front panel and wired automatically.

8) The numerical representation of this indicator by default will currently only show integer values. To change this to display decimal places right click on the GetPositionCounter icon in the block diagram and select Representation and then from the menu select **Single Precision**, the colour of the icon will change to reflect this change.



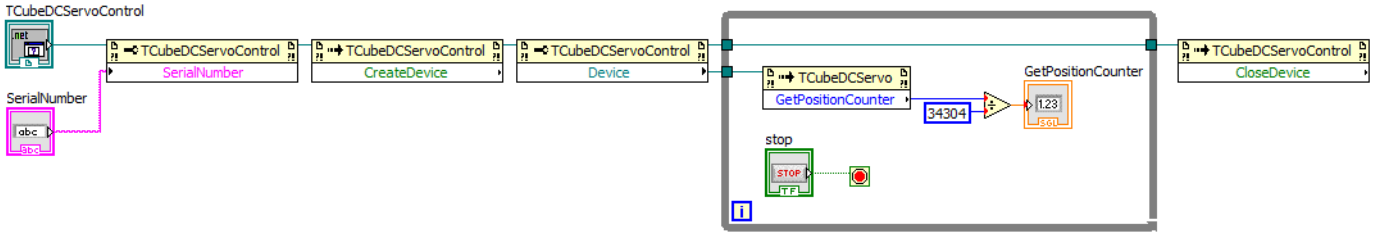
9) The GetPositionCounter only displays the encoder count and not the position in real units. To display the correct position the encoder count will need to be divided by the correct conversion factor. For Z8 linear motors the conversion factor is 34304 counts per millimetre. To display this on the front panel the output from the GetPositionCounter method will need to be divided by 34304.

10) To insert a division operator right click on the block diagram to access the function palette select Programming / Numeric / Divide and insert a division operator into the While Loop.

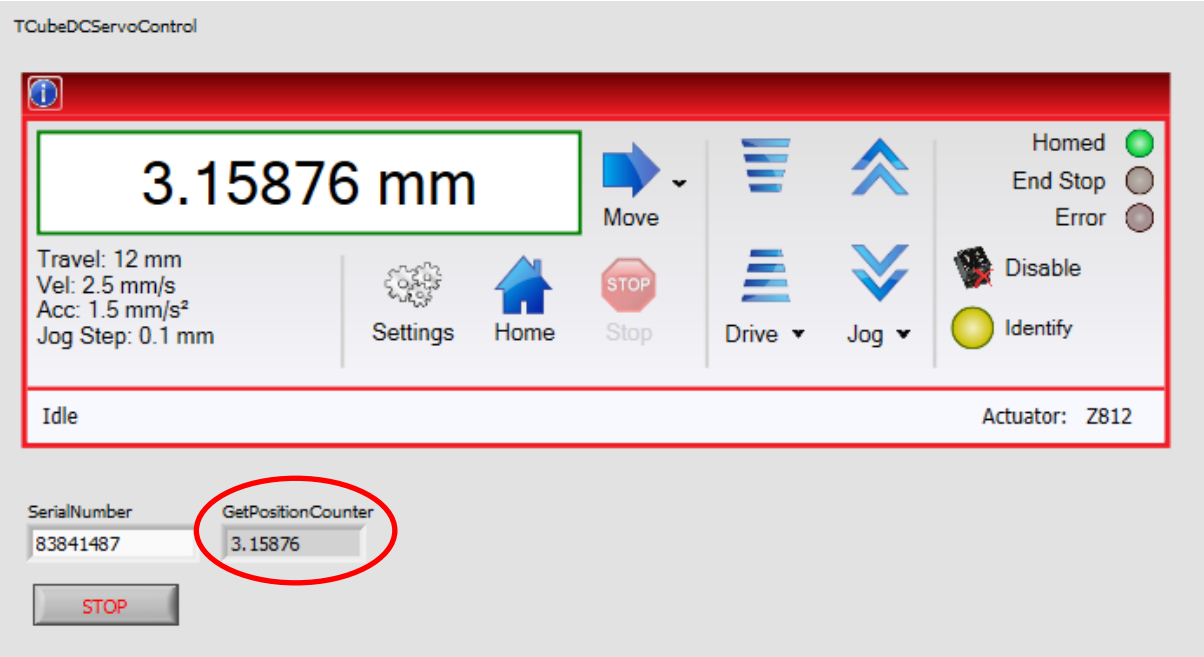


11) Right click the input connector of the lower input of the division operator and select from the shortcut menu displayed Create / Constant. A suitable data type constant is created and wired automatically. The conversion factor should be entered into the entry box, for Z8 linear motors the value should be 34304 to convert to millimetres.

12) The block diagram wiring now needs completing and should now be wired as shown in the following diagram



13) The position indicator can now be viewed on the front panel, circled in the figure below. When the position is changed this dialog will now show the current position.



14) Click File/Save to save the changes.

Controlling Speed

Although not completely necessary to ensure correct VI execution, it is good programming practice in LabView to limit code execution speed where possible. If we take the example that has been built in the previous sections the code will attempt to read the motor controller's position as fast as possible and display for the user.

This is a good example of where program execution speed can be reduced without affecting the program functionality. For a user displayed variable there is little to gain by updating at speeds of more than 5Hz, as the user would not notice the extra speed and may even be unreadable.

Within the while loop a short delay to hold execution by 50ms would ensure that the CPU and APT system are not over burdened with unnecessary processing. A time delay VI can be used to achieve this.

Complete the following steps to insert a wait call within the while loop to reduce the execution speed of the loop.

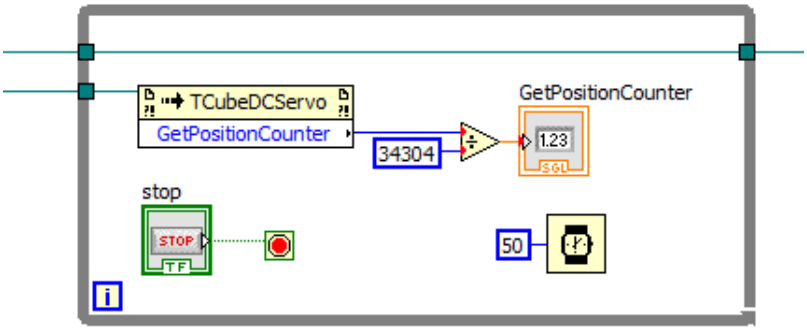
1) Select the Block Diagram window. If not visible, select Window/Show Block Diagram.

2) Display the Timing palette by right clicking the block diagram to display the functions palette and then navigate to Functions Palette/Programming/Timing.

4) Select the Wait (ms) icon to attach it to the cursor. Then drop the VI onto the block diagram within the while loop.

5) Right click the input connector of the Wait(ms) VI and select Create/Constant from the shortcut menu displayed. A constant, of suitable data type, is created and wired automatically. Edit the default value by double clicking the constant and enter 50. The units are milliseconds.

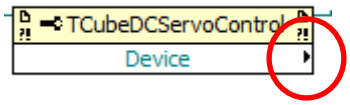
6) Select File/Save to save the changes.



Homing the Motor

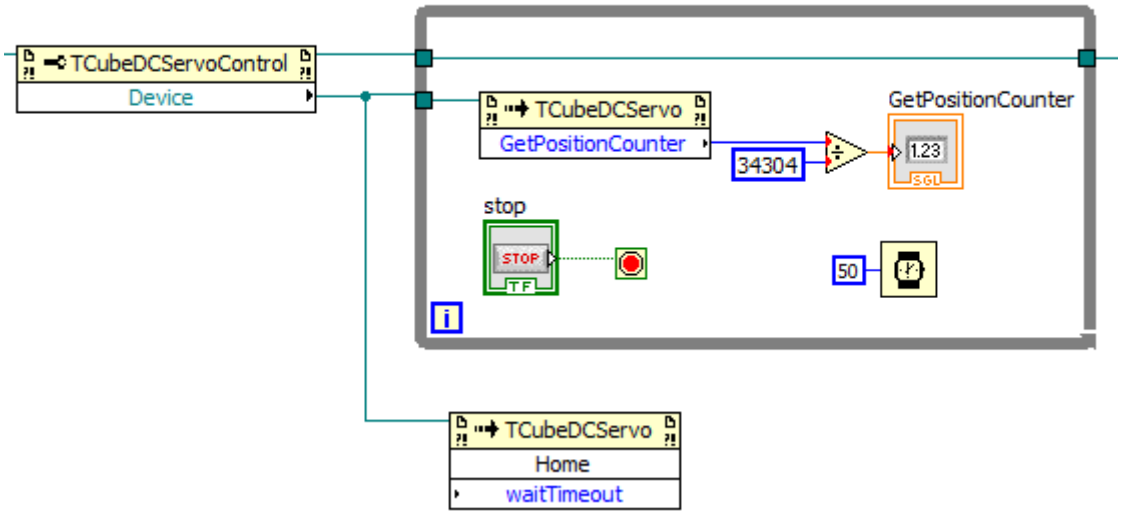
Many motor devices are required to be homed before they can be operated for the first time after power up. This helps to define the Zero Datum position so the absolute position of the stage can be determined. To add a homing method to the VI created in the previous sections please complete the following steps.

1) To create a method to call a motor function right click on the output arrow of the Device property node (circled) and select
Create / Method for Thorlabs.MotionControl.TCube.DCServoCLI.TCubeDCServo Class



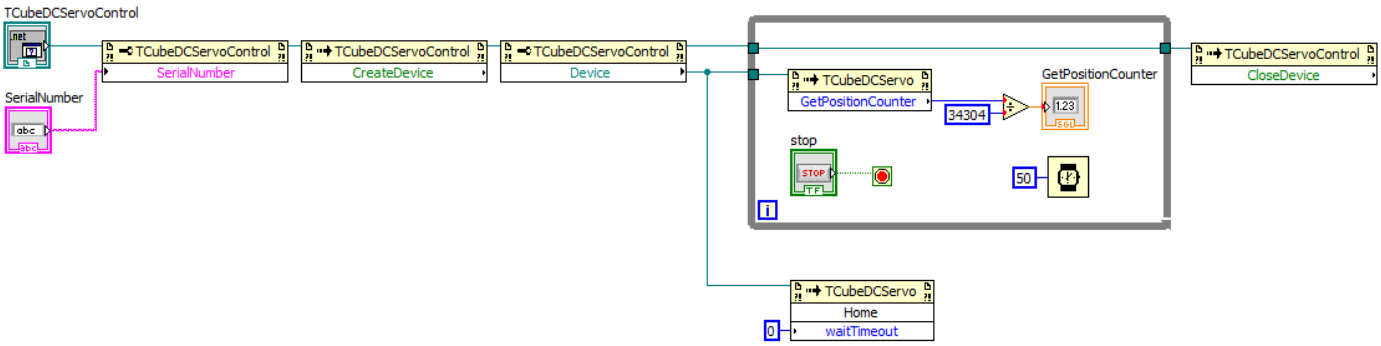
2) From the method drop down menu select the method Home(Int32 waitTimeout). This will create an Invoke node with the selected method. Place this method onto the block diagram.

3) Wire the input reference of the Home method to the wire connecting the Device Property Node to the While loop as shown in the figure below.



4) The method requires a timeout to be entered. If this is value is 0 then the function will return immediately. If this value is non zero, then the function will wait until the move completes or the timeout elapses, whichever comes first. To add a wait timeout right click the input connector of the waitTimeout parameter and select from the shortcut menu displayed Create / Constant. A suitable data type constant is created and wired automatically. Enter a suitable timeout.

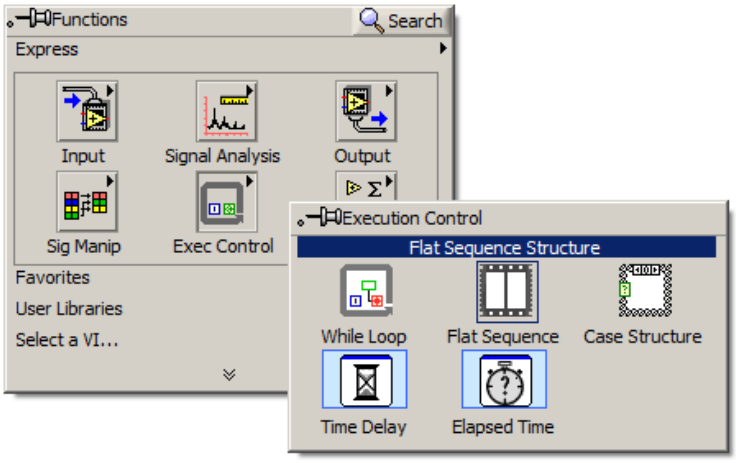
5) The completed block diagram should now look similar to the one shown in the figure below.



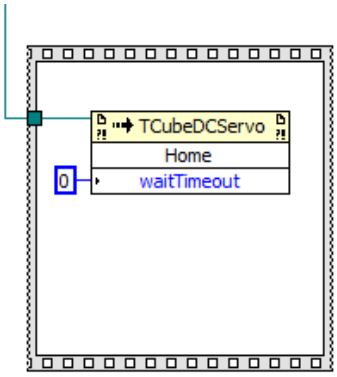
Moving the Motor

1) Before we call the move commands we need to make sure the motor homes first. To control the execution control a **Flat Structure** will need to be added to the block diagram around the Home method. These help control the sequence in which tasks are completed in labview. The appearance looks like a film strip and each frame is executed in turn, from left to right.

2) To add a flat structure to the block diagram Right click the block diagram workspace to view the Functions Palette. To select a Flat Sequence select the Execution Control menu, from the subsequent menu select Flat Sequence.

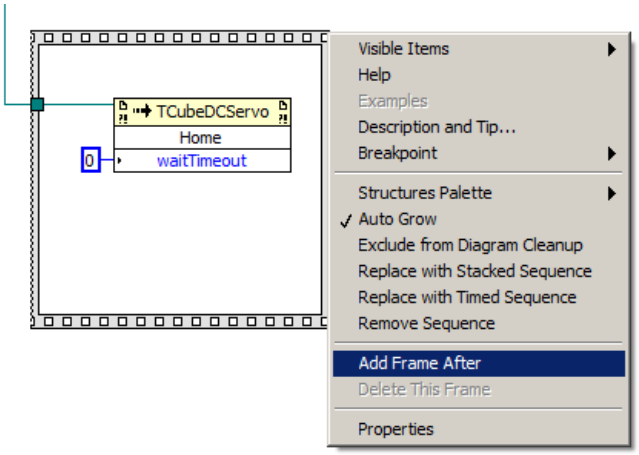


3) Draw a rectangle onto the block diagram around the Home method, this will create a single frame around the Home method as shown below.



4) The move will be executed in the next frame to add a second frame to the flat sequence right click on the edge of the flat structure. From the menu select **Add Frame After**. Enlarge the frame by dragging the edge of the rectangle.

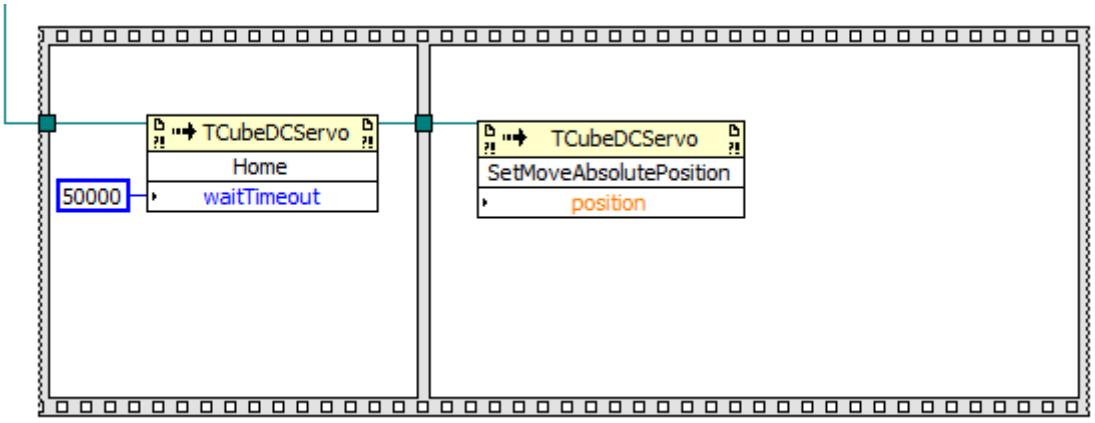
5) To give the homing move enough time to complete increase the waitTimeout on the home method, setting this to ~50000 is sufficient large enough so that the home is completed, however this value will depend on how far away from the home position the stage is and also what type of stage you are using.



1) To move the motor to an absolute position to absolute position firstly needs to be set before the move can execute. To set the absolute move position right click on the Home method invoke node and select **Create / Method for Thorlabs.MotionControl.TCube.DCServoCLI.TCubeDCServo Class**

2) From the method drop down menu select the method **SetMoveAbsolutePosition(Decimal Position)** This will create an Invoke node with the selected method. Place this method onto the second frame of the flat structure.

3) Wire the output reference of the Home method to the input reference of the SetMoveAbsolutePosition method.



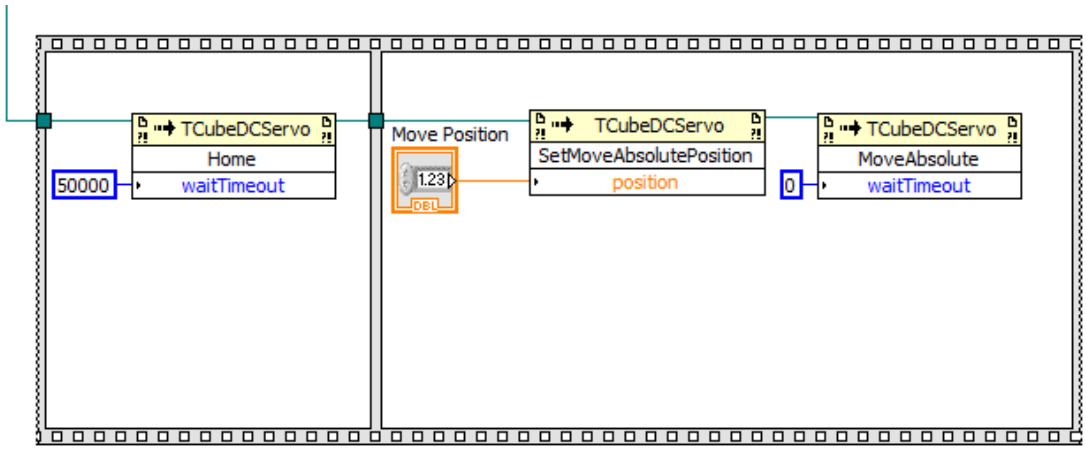
4) To create a control on the front panel for entering the position distance right click the input connector of the method SetMoveAbsolutePosition and select from the shortcut menu displayed Create / Control. A control will be created on the front panel and wired automatically.

5) To execute the move the MoveAbsolute method needs to be added after the SetMoveAbsolutePOsition method. To add this method right click on the SetMoveAbsolutePosition method invoke node and select Create / Method for Thorlabs.MotionControl.TCube.DCServoCLI.TCubeDCServo Class

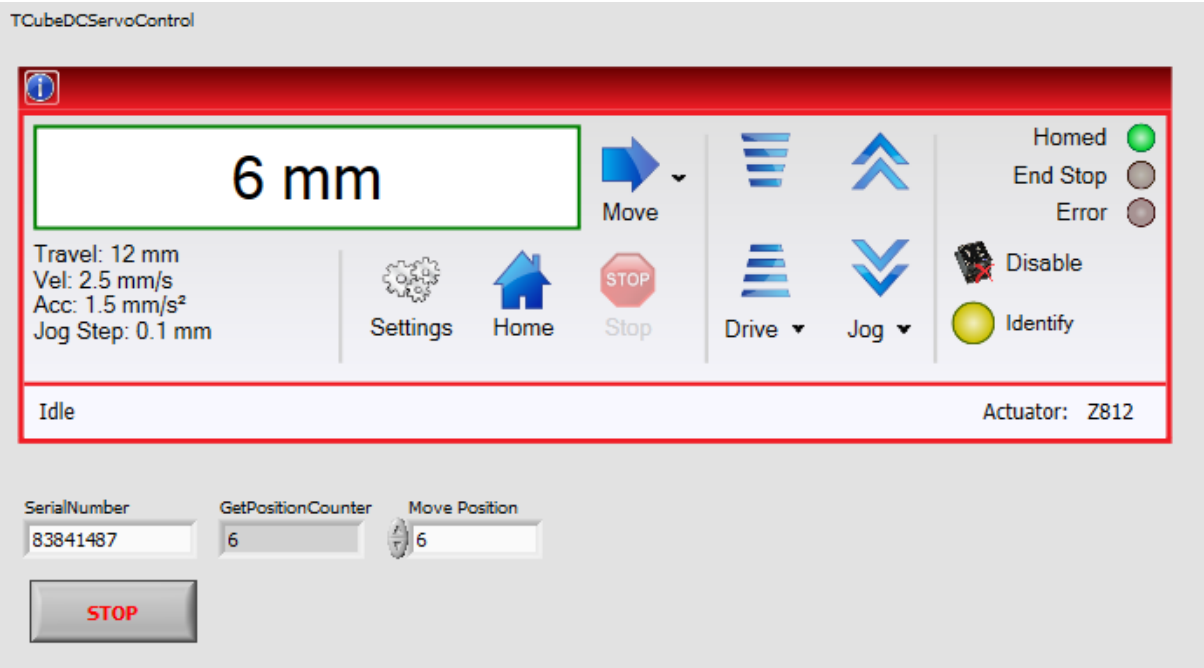
6) From the method drop down menu select the method **MoveAbsolute(Int32 waitTimeout)** this will create an Invoke node with the selected method. Place this method onto the second frame of the flat structure.

7) Wire the output reference of the SetMoveAbsolutePosition method to the input reference of the MoveAbsolute method.

8) To add a wait timeout right click the input connector of the waitTimeout parameter and select from the shortcut menu displayed Create / Constant. A suitable data type constant is created and wired automatically. Zero can be used for the time out in this instance.



9) On the front panel there will now be a control to allow you to enter a position, which should be entered prior to running the VI.



Creating an Iterative Movement Loop

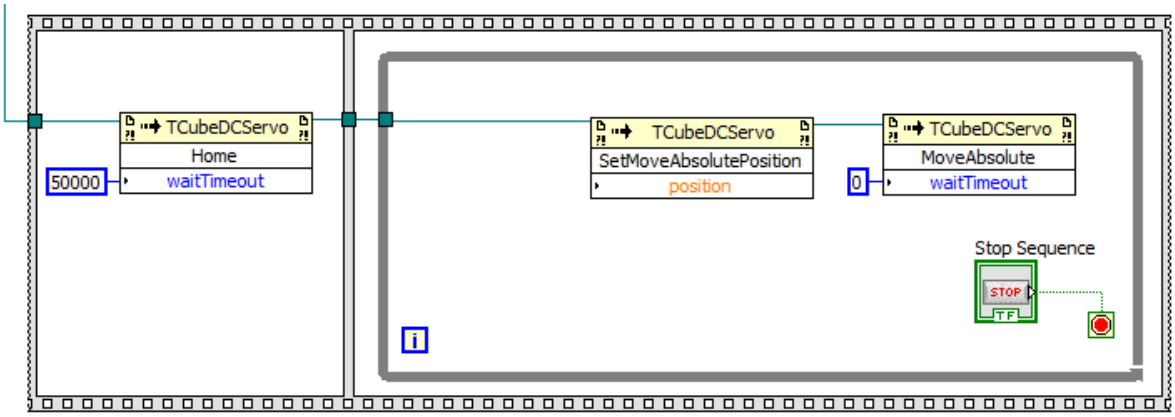
In its current state the VI would not be very useful: it will simply move to the position set before running the program and there would be no further automation of the motor. A common application is to step the motor by a fixed distance as part of a measurement series i.e. the motor will move a small step after which another instrument would take a measurement at this position before the motor moves to the next position where the measurement is repeated.

To create an iterative loop using a While Loop complete the following steps.

1) Right click the block diagram workspace to view the Functions Palette. To select a While Loop select the Execution Control menu, from the subsequent menu select While Loop.

2) Draw a rectangle in the second frame of the flat structure enclosing both the SetMoveAbsolutePosition and MoveAbsolute methods. A new while loop will now be created around these.

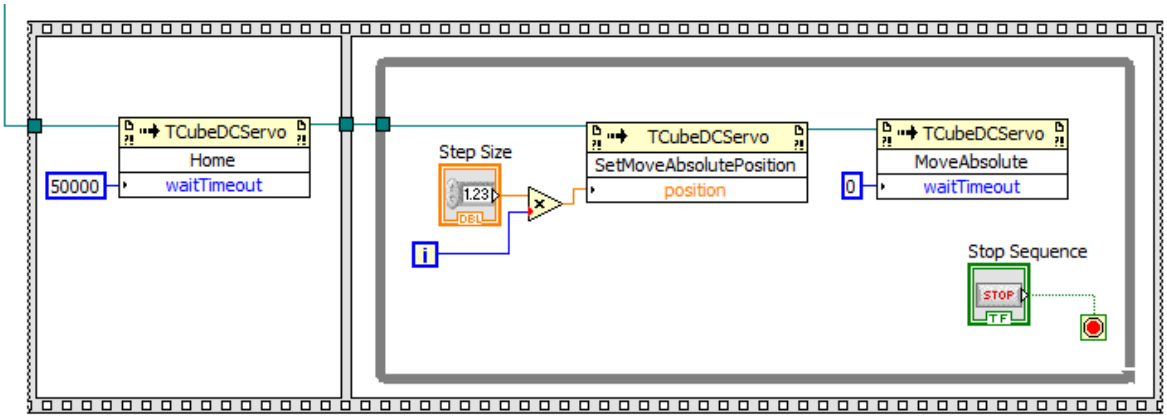
3) Delete the existing position control and connecting wire, this will be replaced in the following steps.



3) The small  icon inside the while loop outputs the iteration count of the loop, i.e. how many times the loop has been executed, starting at zero. To move the motor in discrete steps this iteration count should be multiplied by a fixed step size. To insert a multiplication operator right click on the block diagram to access the function palette select Programming / Numeric / Multiply and insert a multiplication operator into the While Loop.

4) Wire the output of the multiplication operator to the input parameter Position on the SetMoveAbsolutePosition method invoke node. Wire the output of the iteration counter to the bottom input of the multiplication operator.

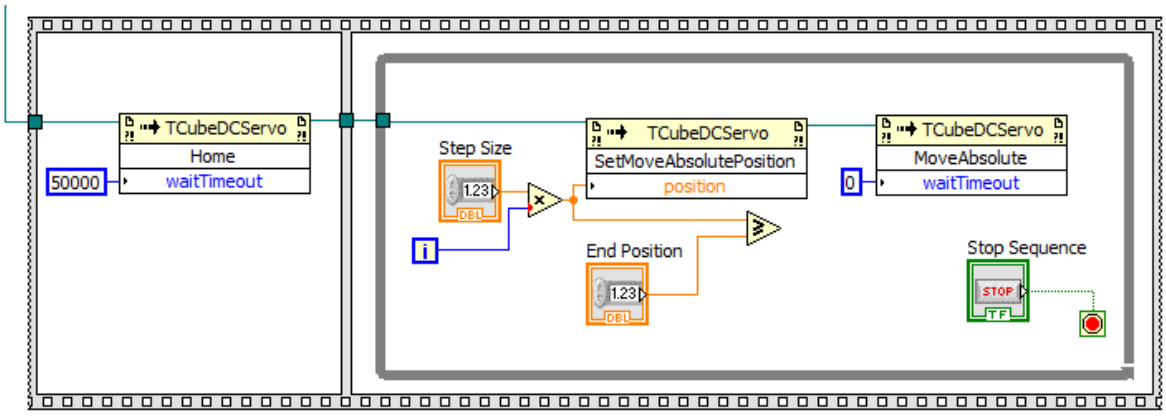
5) To create a control on the front panel for entering the move step distance right click the top input connector of the multiply operator and select from the shortcut menu displayed Create / Control. A control will be created on the front panel and wired automatically.



6) To ensure the stage stops moving at the end of its sequence it is useful to enter a final position so that when the stage reaches this position the movement will stop. To do this the commanded position can be compared against a pre-set final position using the greater than or equal operator; if the input position is equal to or greater than the final position then this means the sequence has reached the end position and the motor should stop.

7) To insert a Greater or Equal operator right click on the block diagram to access the function palette select Programming / Comparison / Greater or Equal? and insert into the While Loop. Wire the top input of the greater or equal operator to the output wire of the multiplication operator as shown in the block diagram below.

8) To create a control on the front panel for entering the final position for the sequence right click the bottom input connector of the greater or equal operator and select from the shortcut menu displayed Create / Control. A control will be created on the front panel and wired automatically.

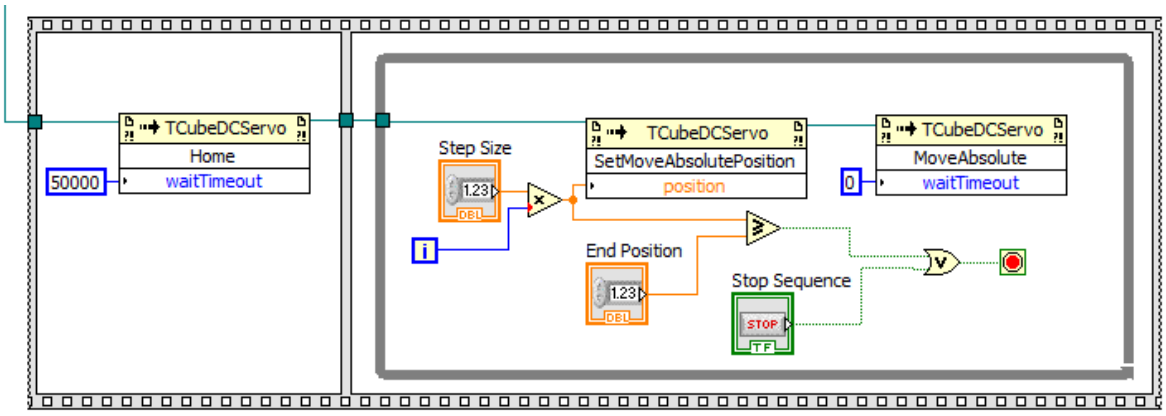


9) The ability to stop the while loop on demand, or at the end of the sequence some OR logic needs to be added. To do this the wiring between the stop button terminal and the while loop stop node needs to be removed and replaced with some Boolean logic.

10) The Greater or Equal operator produces a Boolean output, this can be used to stop the while loop when the sequence reaches the final position as the output will be TRUE at the final position.

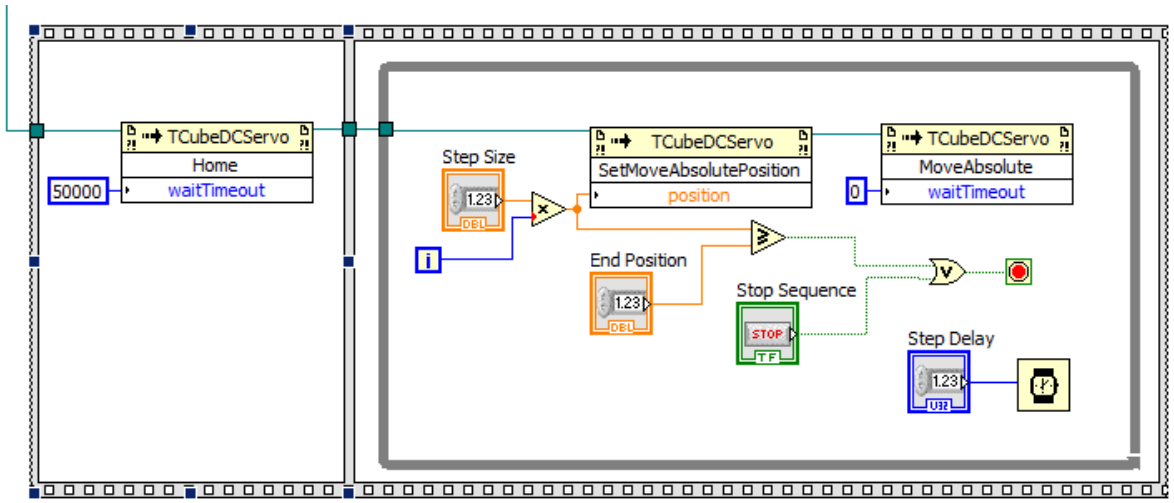
11) Display the Boolean palette by right clicking the white workspace area of the block diagram and select from the shortcut menu Programming/Boolean. Select the OR icon to attach it to the cursor. Click in the block diagram and place inside the while loop structure.

12) Wire the output of the greater or equal and stop button icon to the inputs of the OR logic operator, wire the output to the while loop stop node.



13) In order to allow time to carry out a measurement between move steps it is necessary to add a delay between each loop iteration. Display the Timing palette by right clicking the block diagram to display the functions palette and then navigate to Functions Palette/Programming/Timing. Select the Wait (ms) icon to attach it to the cursor. Then drop the VI onto the block diagram within the while loop.

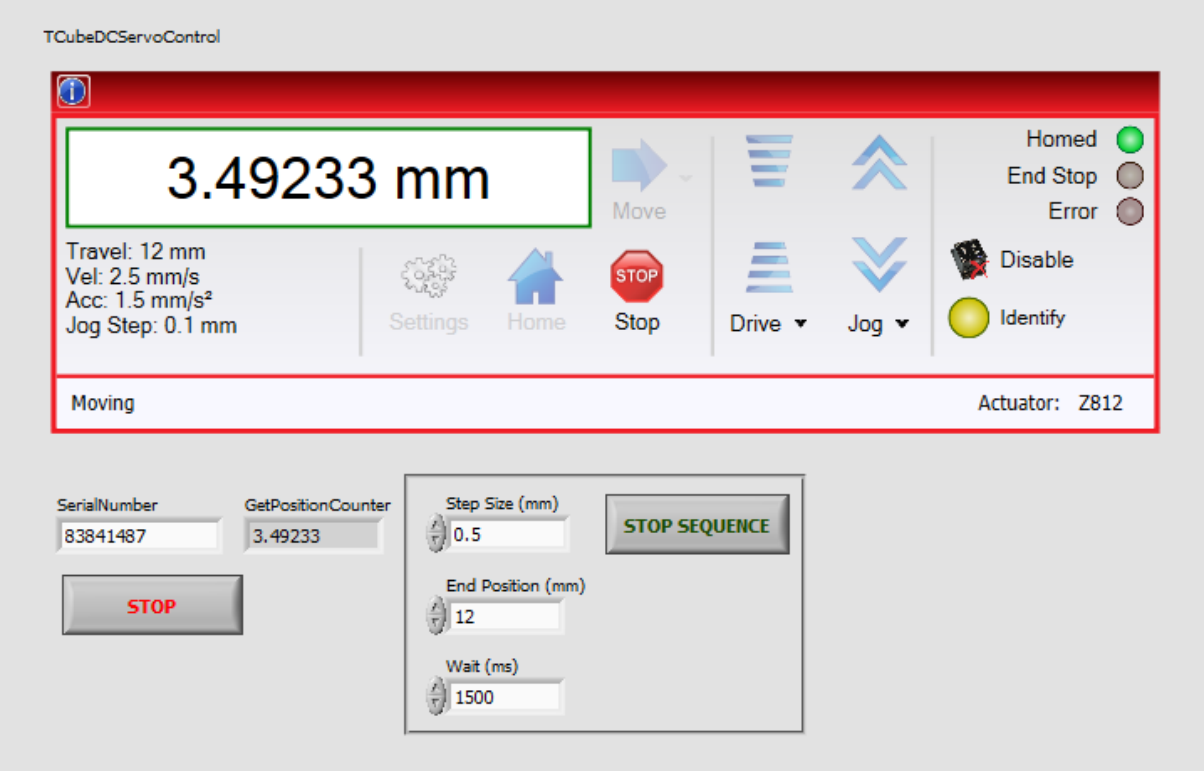
14) Right click the input connector of the Wait(ms) VI and select Create/Control from the shortcut menu displayed. A control that allow you to set the delay between move steps will now be added to the front panel, where a suitable delay time in milliseconds can be entered.



15) The Front panel will now have a number of additional controls, if the front panel is not visible select from the block diagram menu bar Window / Show Front Panel.

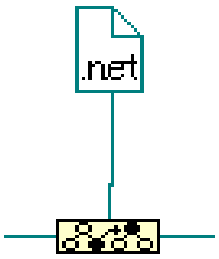
16) The new controls can now be renamed and moved to a more suitable position as shown in figure below.

17) Select File/Save to save the changes.

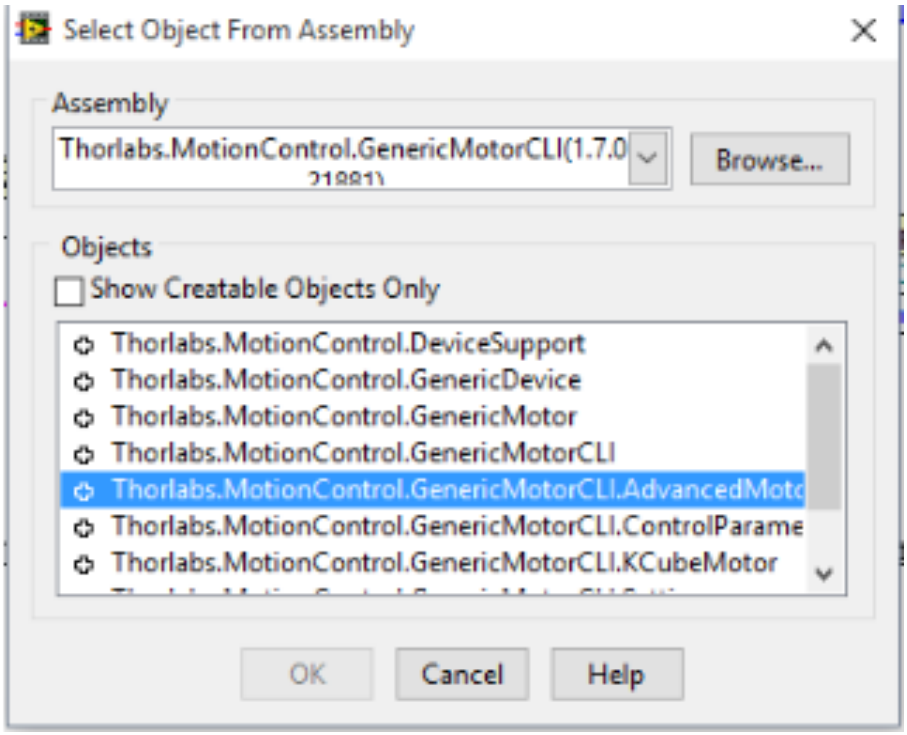

TCubeDCServoControl
3.49233 mm
Move
Homed
End Stop
Error
Travel: 12 mm
Vel: 2.5 mm/s
Acc: 1.5 mm/s²
Jog Step: 0.1 mm
Settings
Home
STOP
Stop
Drive
Jog
Disable
Identify
Moving
Actuator: Z812
SerialNumber
83841487
GetPositionCounter
3.49233
STOP
Step Size (mm)
0.5
STOP SEQUENCE
End Position (mm)
12
Wait (ms)
1500

Setting Up a Multi-Axis Program

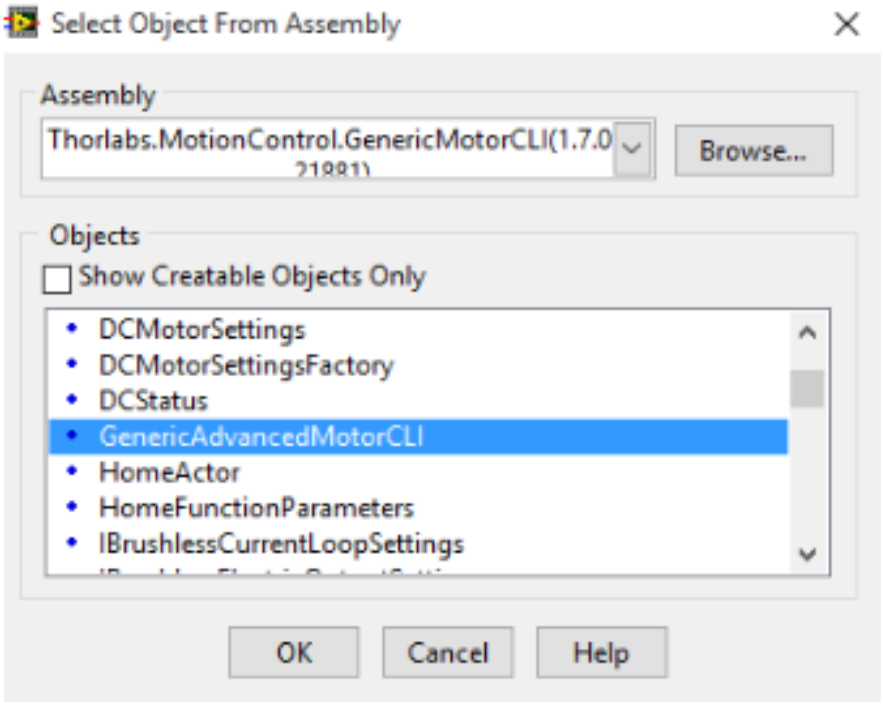
1) On the target class node, right click and select Create Constant



2) Right click on the new .NET contant icon, go down to "Select .NET Class" then click browse
3) Browse to your project folder (which should contain all the copied dlls and select "Thorlabs.MotionControl.GenericMotorCLI.dll"
4) In the dialog box that now appears, expand the Thorlabs.MotionControl.GenericMotorCLI.ADvancedMotor



5) In the expanded list scoll down to GenericAdvanceMotorCLI and then click OK



6) For a 2-channel benchtop controller you will require two .NET objects on your front panel for each channel. To select the channel, add the 'ChannelNumber' element to the BenchtopStepperControl property node & add the channel number