API for LabView

Application Note

agito

www.agito-akribis.com

Member of Akribis Systems group

Revision History

Version	Description	Date
1.0	Initial release	21 November 2022

Contact Information

Manufacturer	Agito Akribis Systems Ltd., Member of Akribis Systems Group
Address	6 Yad-Harutsim St., P.O.Box 7172, Kfar-Saba 4464103
Telephone	+972-9-8909797
Website	www.agito-akribis.com

Copyright Notice

Products Rights

AGDx, AGCx, AGMx, AGAx, AGIx, and AGLx are products designed by Agito Akribis Systems Ltd. in Israel. Sales of the products are licensed to Akribis Systems Pte Ltd. under intercompany license agreement.

Agito Akribis Systems Ltd. has full rights to distribute above products worldwide.

Disclaimer

This product documentation was accurate and reliable at the time of its release.

Agito Akribis Systems Ltd. reserves the right to change the specifications of the product described in this manual without notice at any time.

Trademarks

Agito PCSuite is a trademark of Agito Akribis Systems Ltd.

Contents

1 About this Application Note

This application note describes the use of the AAMotion API in the LabView software to interact with Agito Motion controllers.

It includes the instructions to download the .NET library for AAMotion which will be used to call methods and objects to interact with the controller.

It also explains how the .NET objects can be implemented in a LabView project to perform basic functionality such as connecting to, sending commands to and reading information from the controller.

This application note assumes that the user has knowledge in using LabView. As such, it will not include details about using LabView. The user may visit https://learn.ni.com/learn/article/labview-tutorial to learn more about LabView if necessary.

1.1 Introduction to LabView

LabView is a software from National Instrument that provide a graphical data flow programming interface for the user to create virtual instruments (applications). It is widely used for testing and data acquisition applications.

1.2 Introduction to AAMotion

AAComms is a .NET library that provides for the AAComm server which the user may use to establish communication with Agito motion controllers. It is a low-level API that targets the communication layer.

AAMotion is a .NET library that is built upon AAComms. It is a high-level library that provides for higher level methods and objects such as moving a motor, or doing homing.

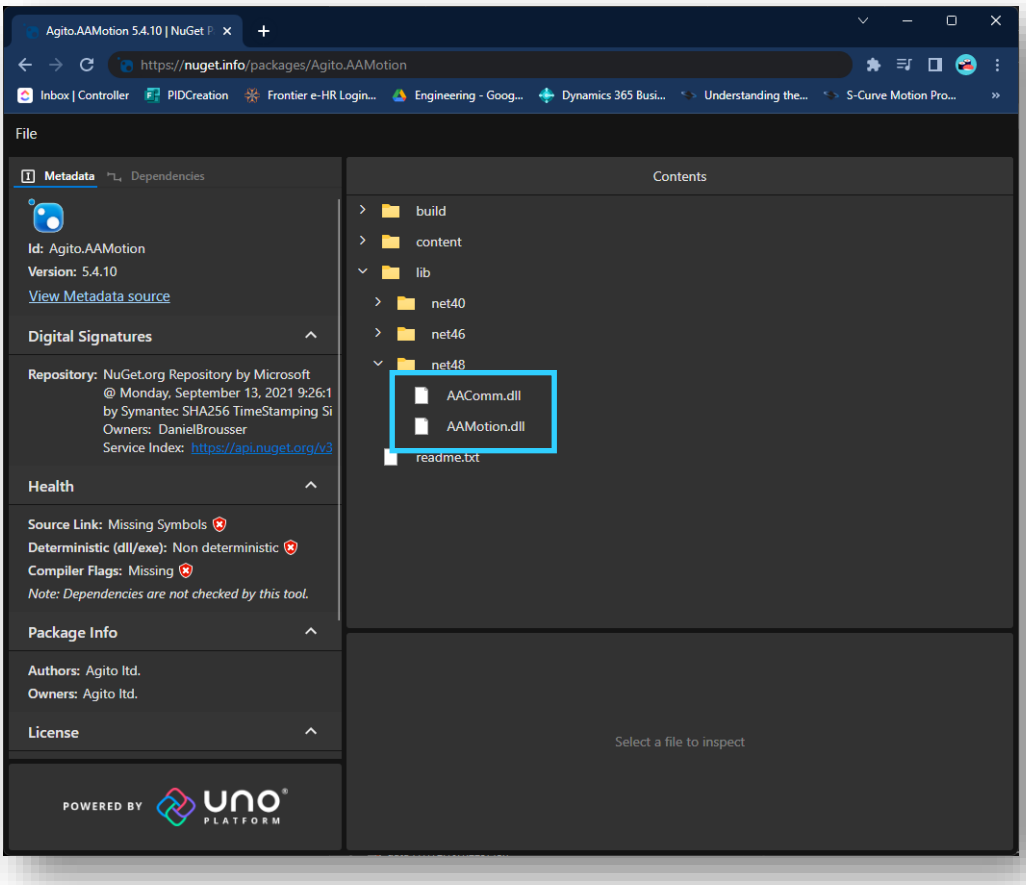
2 Prerequisites

2.1 AAMotion API Libraries

AAMotion and AAComm libraries are updated and uploaded to the Nuget cloud.

Go to https://nuget.info/packages/Agito.AAMotion, download the following two files, named AAMotion.dll and AAComm.dll.

Paste these files in a suitable location within your project. These files will be referenced later.

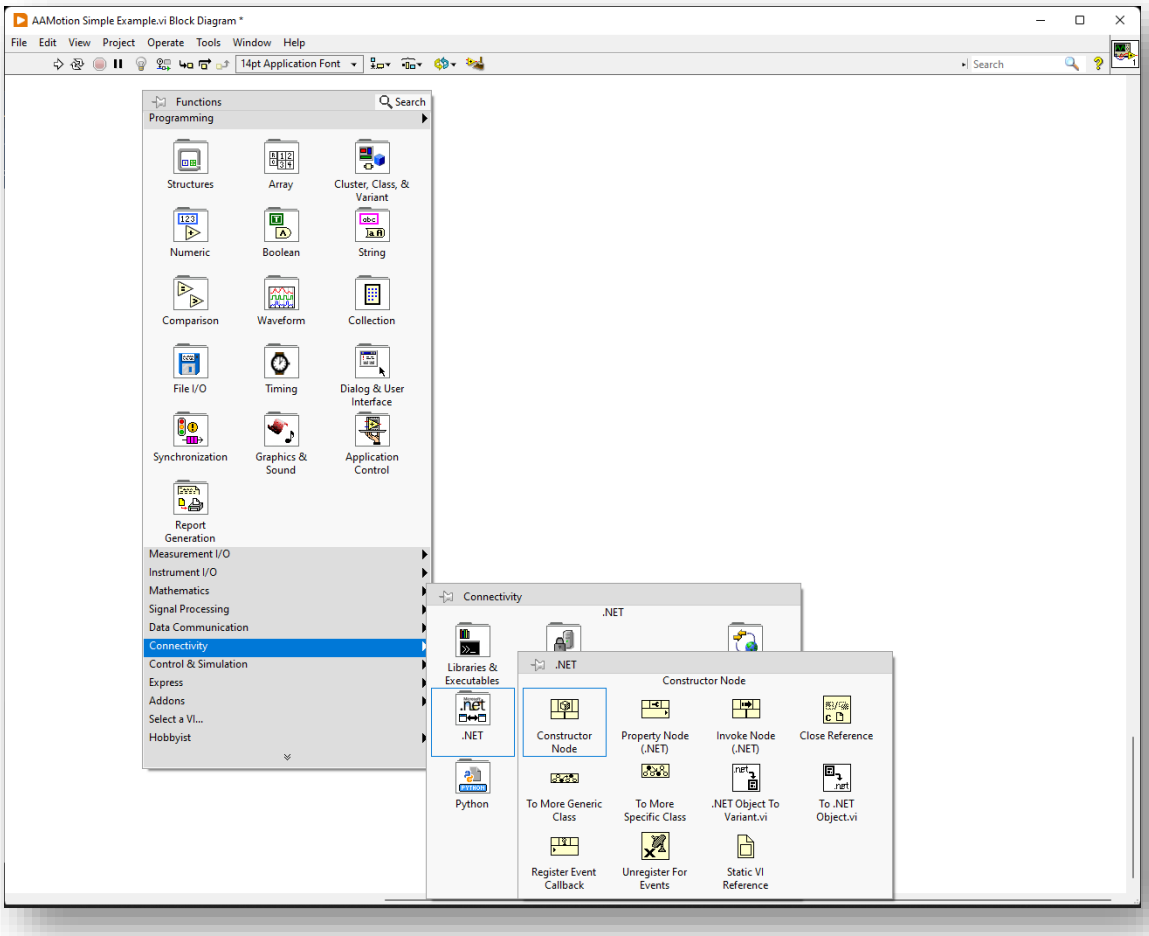


3 Basics

3.1 . NET Blocks

LabView provides for interfaces to use the .NET framework. To access these interfaces, navigate to Functions -> Connectivity -> .NET.

In this note, only the Constructor Node, Property Node and Invoke node will be used.



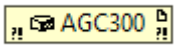
3.2 Constructor Node Block

The Constructor Node allows for the instantiation of classes within the AAMotion library.

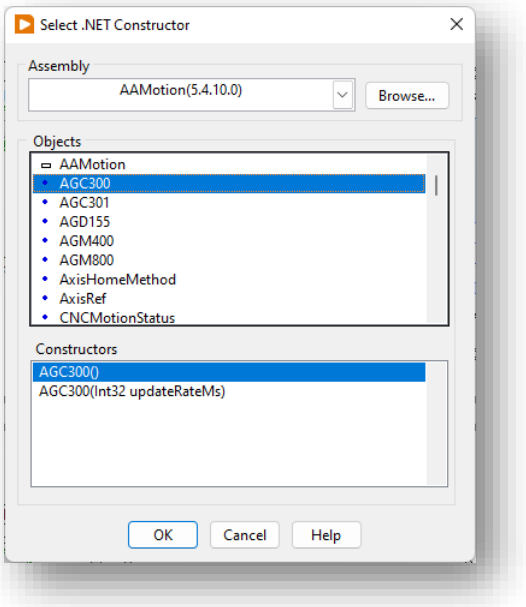
1.1.1 Controller Class

AAMotion implements a motion controller class for each of the products. AGC300 class can be used for the AGC300 and AGD200 while AGC301 can be used for AGC301 and AGD301 since they share the same control board.

The controller class will be the main object used to access the controller, its axes and parameters.



The default constructor instantiates a controller object that polls for status updates every 50ms. Should a different update rate be required, an override exists to define a custom update rate. In general, communication over TCP IP is not real time, and it is not recommended to set the update rate lower than 5ms as it may cause traffic congestion.



3.3 Property Node Block

The Property Node allows for accessing properties of the object linked to it.

1.1.2 Axis Class

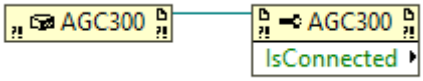
An axis is a 'property' of a motion controller object; a motion controller object can have one-many axes. AGC300, for example, is a motion controller that has 3 axes (A, B, C).

The block diagram shows an axis 'property' of an instance of AGC300.

1.1.3 General Properties

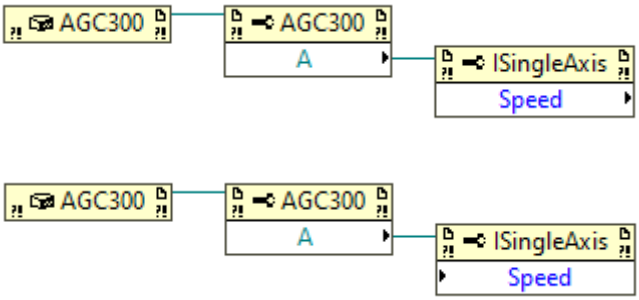
Motion Controllers have properties that depict the state it is in. Properties can also be configured to manipulate the behavior of the controller.

IsConnnected is an example of a read-on property that returns a true if the connector is connected and false if it is not.



'Properties' can also have properties. Speed is an example of a property of an axis (which is a property itself). Properties can be read-only, write-only, or both. Speed is an example or a property that is both readable and can be written to.

To write to a property, right-click on the node and select the option to Change All To Write. This option is only available to properties that have setters implemented.



3.4 Invoke Node Block

The Invoke Node Block allows for calling of methods and functions of the object linked to it.

1.1.4 Methods

AAMotion caters for methods to execute certain functionality of the controller.

Connect is a method that takes in an IP address as a string and searches for such a controller. If it successfully finds and connects to the controller, it returns a True value, otherwise it returns False.

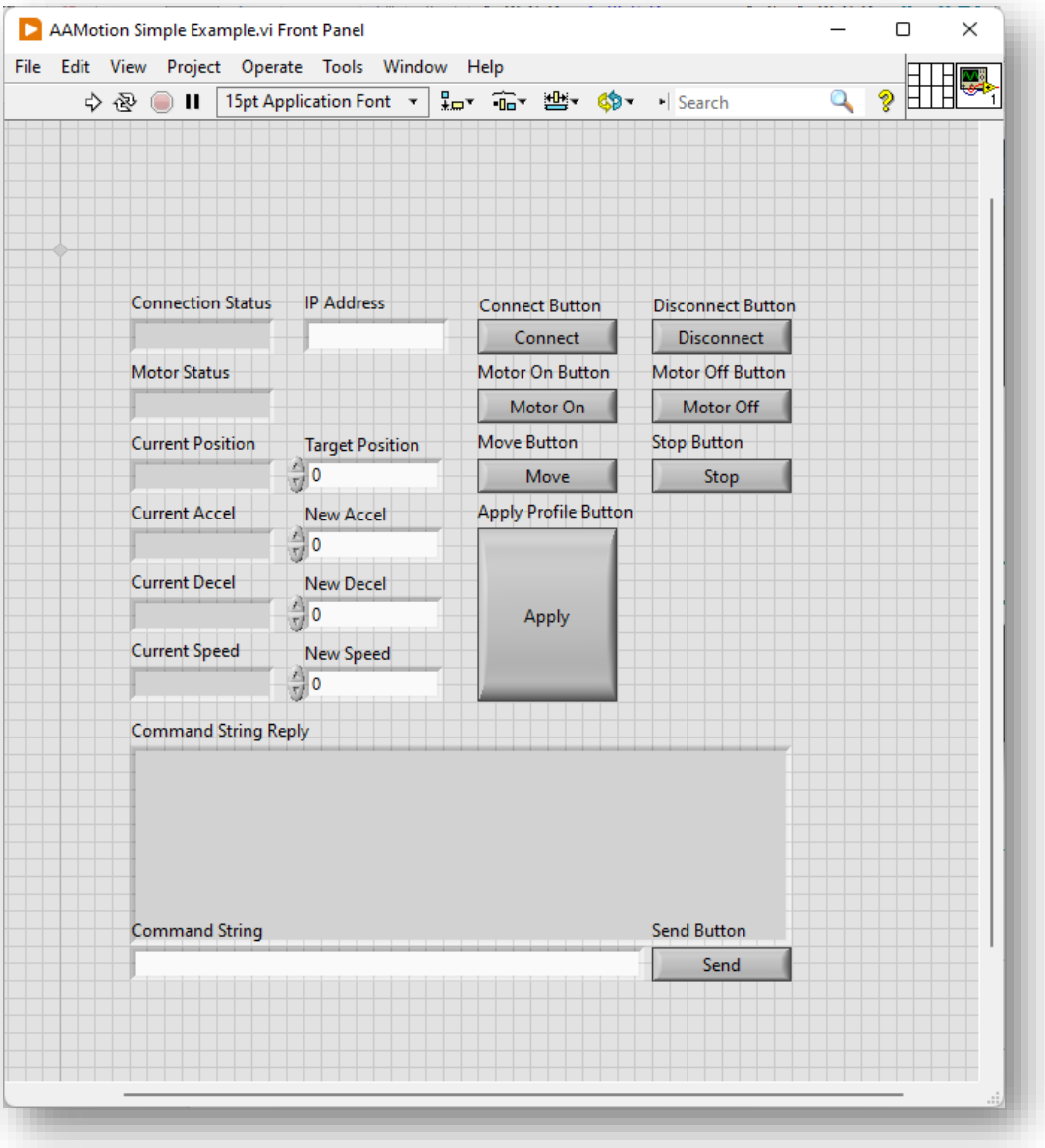
4 Simple Example

The following steps explain how to setup a simple Virtual Instrument on Labview to,

- Connect to a controller
- Read the position of the axis A
- Move the motor
- Send generic commands

4.1 Create a LabView Project

Create a LabView project and populate the project with the following controls. For purposes of showing the API, these controls will be tied directly to the API.

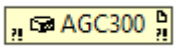


4.2 Block Diagrams

1.1.5 Create a motion controller

Instantiate a single instance of a controller; this object will be the base object for accessing the controller.

Right click on the canvas and go to, Functions -> .NET -> Constructor Node.

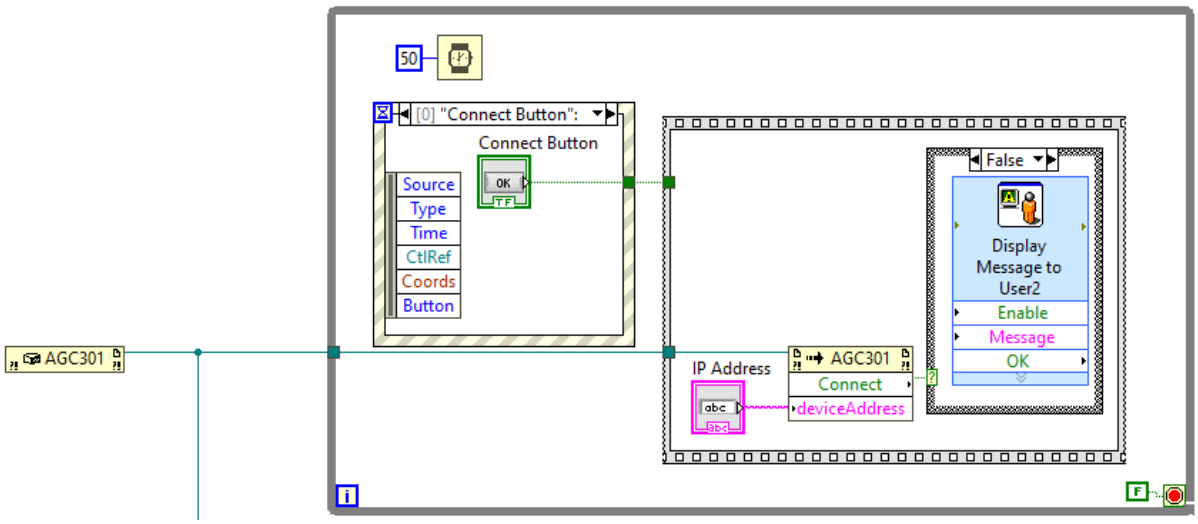


1.1.6 Create a button to connect to the controller

Create an Invoke Node and wire it to the controller object. This provides access to call the methods available to the controller object.

Connect() method will be used to create a connection between the PC and the controller.

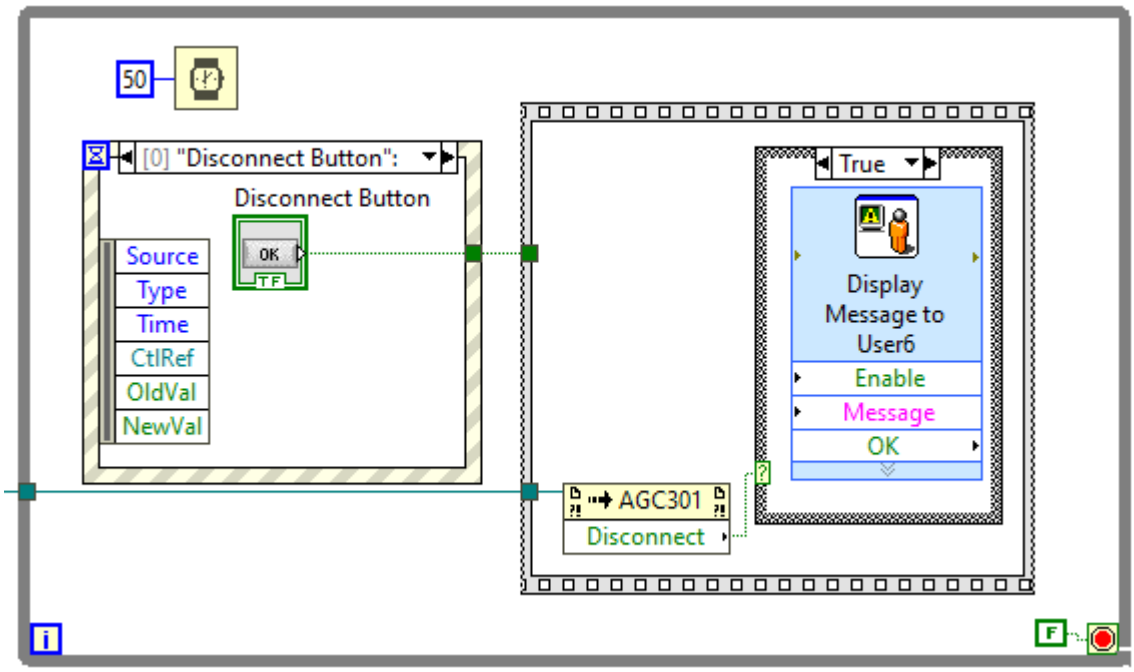
1. Right click on the canvas and go to, Functions -> .NET -> Invoke Node.
2. Select the method to use as Connect(String deviceAddress)
3. Add in the necessary events and inputs to trigger the method upon button click.



1.1.7 Create a button to disconnect from the controller

Disconnect() method will be used to disconnect the controller.

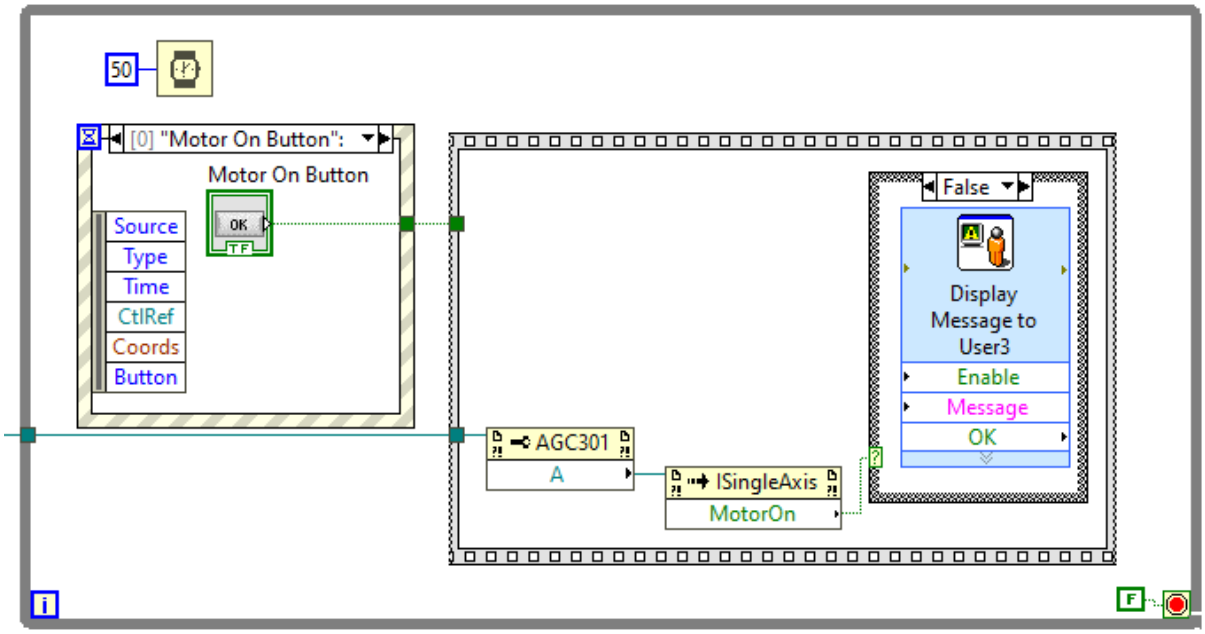
1. Right click on the canvas and go to, Functions -> .NET -> Invoke Node.
2. Select the method to use as Disconnect().
3. Add in the necessary events and inputs to trigger the method upon button click.



1.1.8 Create a button to motor on axis A

MotorOn() method will be used to motor on the axis.

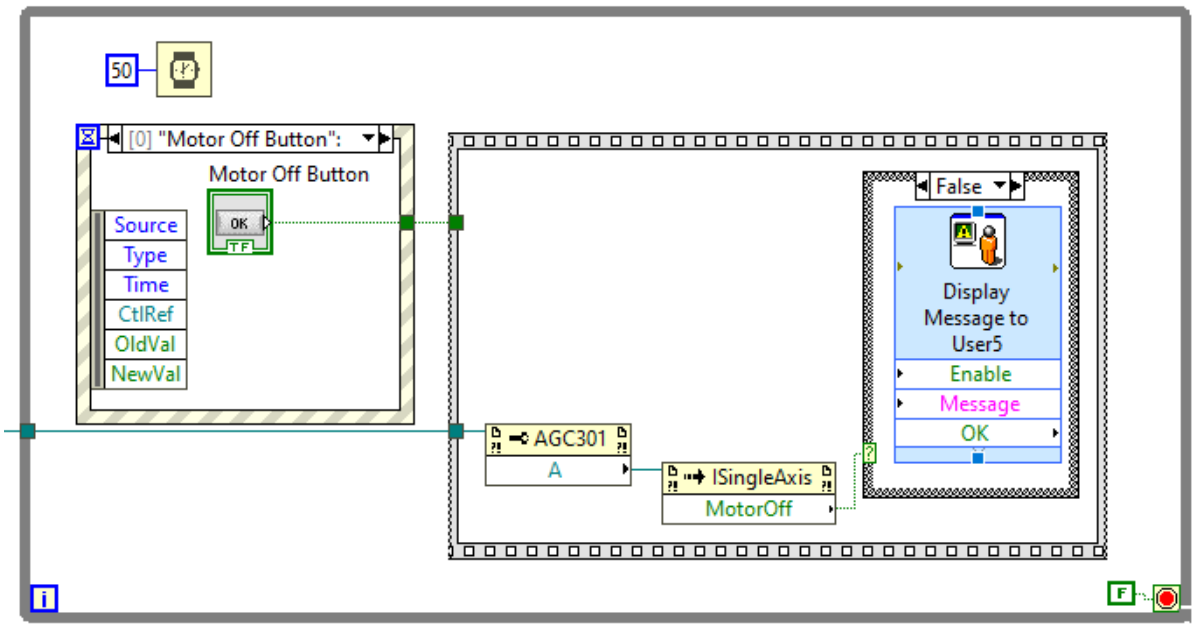
1. Right click on the canvas and go to, Functions -> .NET -> Property Node.
2. Select the property to be axis “A”.
3. Right click on the canvas and go to, Functions -> .NET -> Invoke Node.
4. Select the method to use as MotorOn().
5. Add in the necessary events and inputs to trigger the method upon button click.



1.1.9 Create a button to motor off axis A

MotorOff() method will be used to motor on the axis.

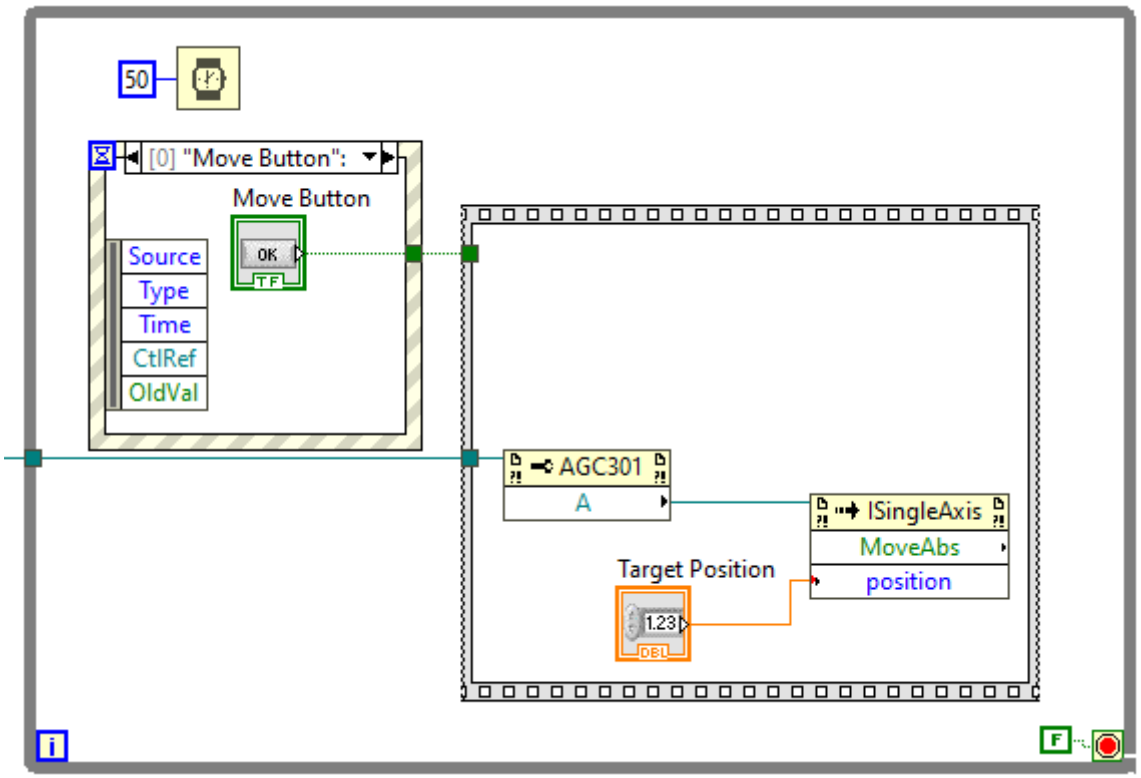
1. Right click on the canvas and go to, Functions -> .NET -> Property Node.
2. Select the property to be axis “A”.
3. Right click on the canvas and go to, Functions -> .NET -> Invoke Node.
4. Select the method to use as MotorOff().
5. Add in the necessary events and inputs to trigger the method upon button click.



1.1.10 Create a button to move axis A

MoveAbs() method will be used to move the motor to an absolute position. It takes in an integer parameter as the target position.

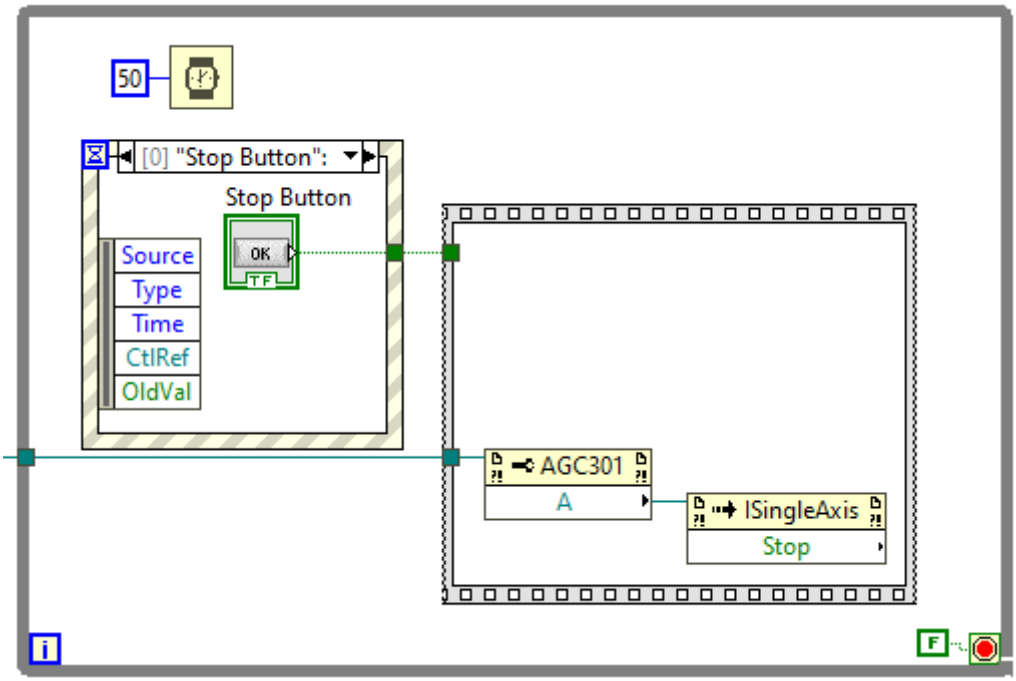
1. Right click on the canvas and go to, Functions -> .NET -> Property Node.
2. Select the property to be axis “A”.
3. Right click on the canvas and go to, Functions -> .NET -> Invoke Node.
4. Select the method to use as MoveAbsolute().
5. Add in the necessary events and inputs to trigger the method upon button click.



1.1.11 Create a button to stop axis A

Stop() method will be used to decelerate the motor to a stop.

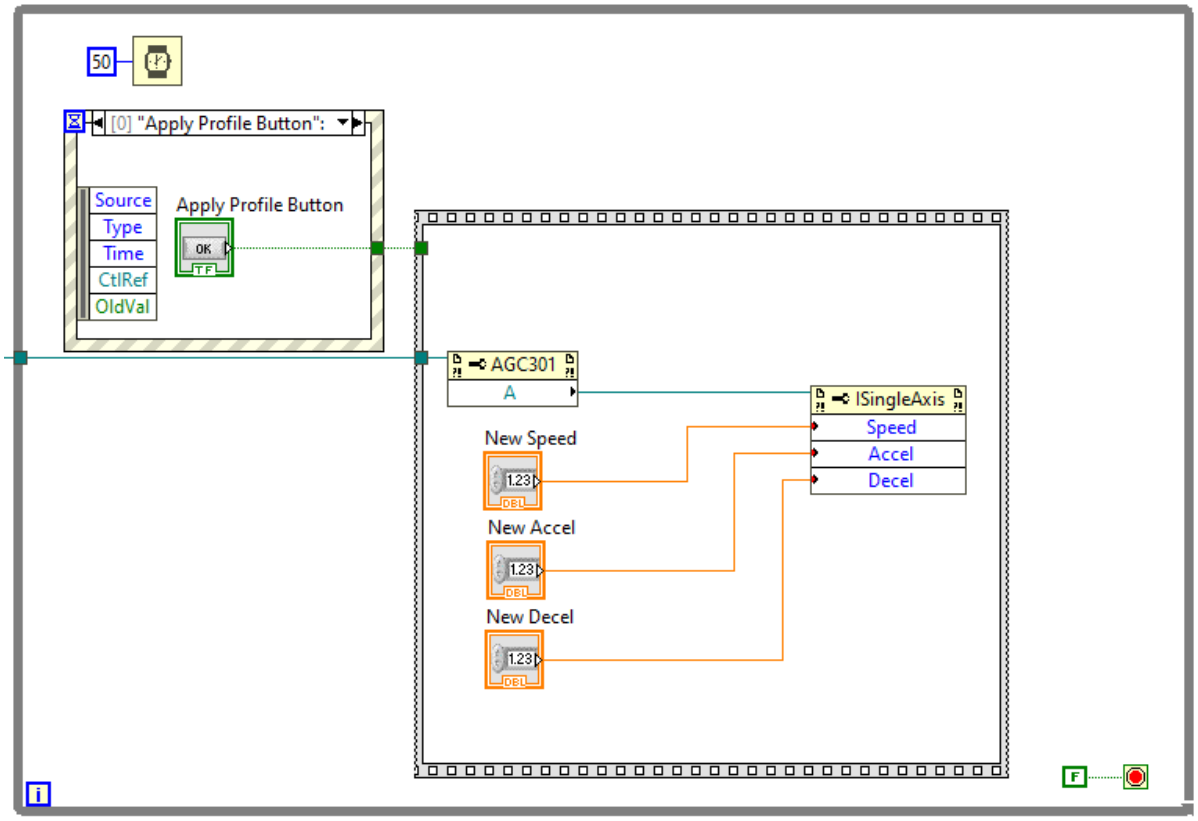
1. Right click on the canvas and go to, Functions -> .NET -> Property Node.
2. Select the property to be axis “A”.
3. Right click on the canvas and go to, Functions -> .NET -> Invoke Node.
4. Select the method to use as Stop().
5. Add in the necessary events and inputs to trigger the method upon button click.



1.1.12 Create a button to set parameters

Values may be assigned to properties that have setters. Motion profile parameters such as speed, acceleration, and deceleration are examples of such properties belonging to an axis.

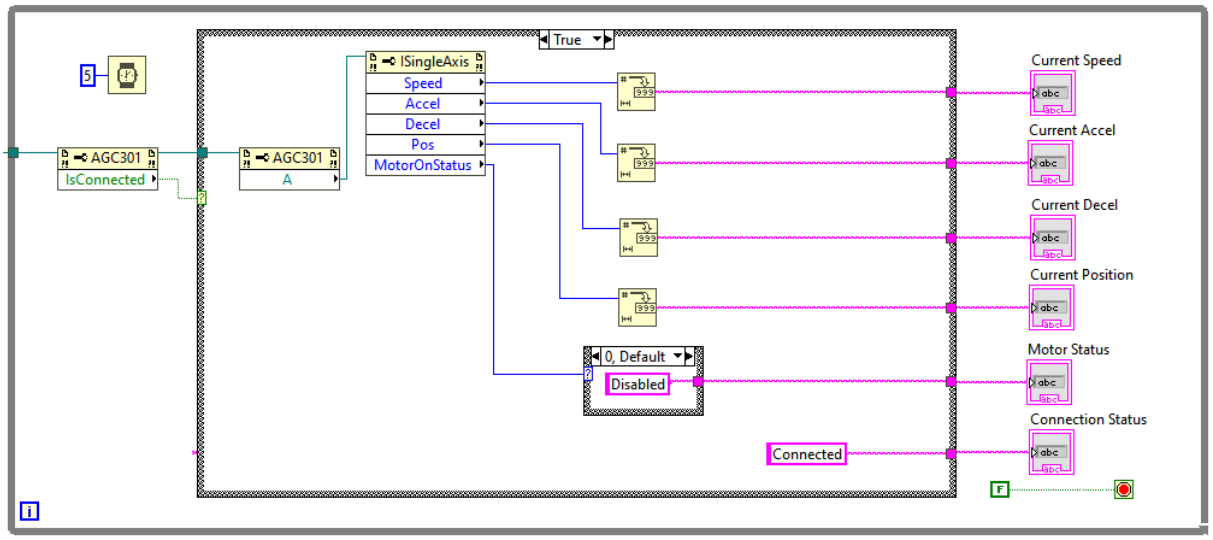
1. Right click on the canvas and go to, Functions -> .NET -> Property Node.
2. Select the property to be axis “A”.
3. Right click on the canvas and go to, Functions -> .NET -> Property Node.
4. Mouse over box and extend the number of elements.
5. Select the properties to be “Speed”, “Accel” and “Decel”.
6. Add in the necessary events and inputs to trigger the property upon button click.

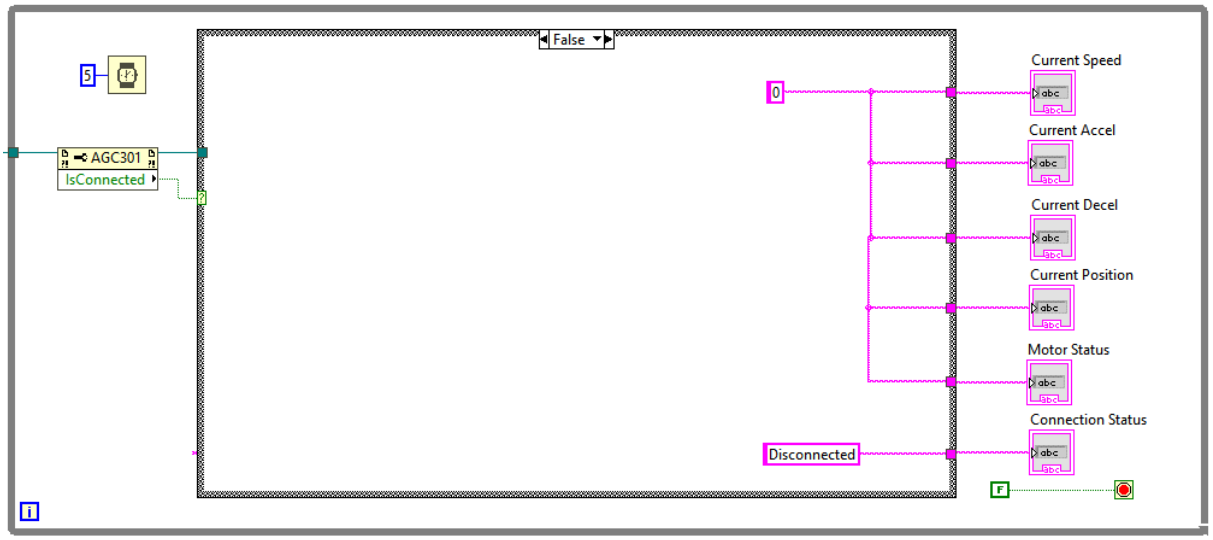


1.1.13 Create a loop to update parameters

Values may be assigned to properties that have setters. Motion profile parameters such as speed, acceleration, and deceleration are examples of such properties belonging to an axis.

1. Right click on the canvas and go to, Functions -> .NET -> Property Node.
2. Select the property to be axis “A”.
3. Right click on the canvas and go to, Functions -> .NET -> Property Node.
4. Mouse over box and extend the number of elements.
5. Select the properties to be “Speed”, “Accel” and “Decel”.
6. Add in the necessary events and inputs to trigger the property upon button click.





1.1.14 Create a terminal to send generic commands

SendCommandString() is a method that takes in a generic string command and returns a string response. Refer to the keyword manual and communication manual for the syntax and response.

1. Right click on the canvas and go to, Functions -> .NET -> Method Node.
2. Select the method to be SendCommandString().
3. Add in the necessary events and inputs to trigger the property upon button click.