

Using LabVIEW™

with

BiPOM Boards

Quick Start Guide

Document Revision: 1.02

Date: 14 February, 2012



BiPOM Electronics, Inc.

16301 Blue Ridge Road, Missouri City , Texas 77489

Telephone: 1-713-283-9970. Fax: 1-281-416-2806

E-mail: info@bipom.com

Web: www.bipom.com

© 2012 BiPOM Electronics, Inc. All Rights Reserved.

All trademarked names in this document are the property of respective owners.

Overview

LabVIEW™ (abbreviation for **L**aboratory **V**irtual **I**nstrumentation **E**ngineering **W**orkbench) is a visual programming language and development environment from National Instruments™. LabVIEW™ is commonly used for data acquisition, instrument control and industrial and lab automation on various operating systems including Microsoft Windows, UNIX, Linux, and Mac OS X.

ClientAPI is a set of library functions that simplifies microcontroller project development. ClientAPI runs on a host Windows PC connected to the server portion of the package (Server51) that runs on our micro-controller boards. ClientAPI and Server51 communicate through RS232 serial connection.

The purpose of this document is to describe how to use LabVIEW™ with BiPOM boards, including the popular MINI-MAX series using ClientAPI as the interface.

Minimum Requirements

1. A Windows PC that is capable of running LabVIEW™
2. One available Serial (COM) port (A USB to COM converter can be used if needed)
3. An 8051 compatible BiPOM board such as MINI-MAX/51-C2, -D, -E, -F

Architecture

ClientAPI allows third-party applications running on a Windows PC to communicate with, control and collect data from BiPOM boards in a client-server fashion. The application on the PC acts as the client and the board acts as a server as shown in Figure 1:

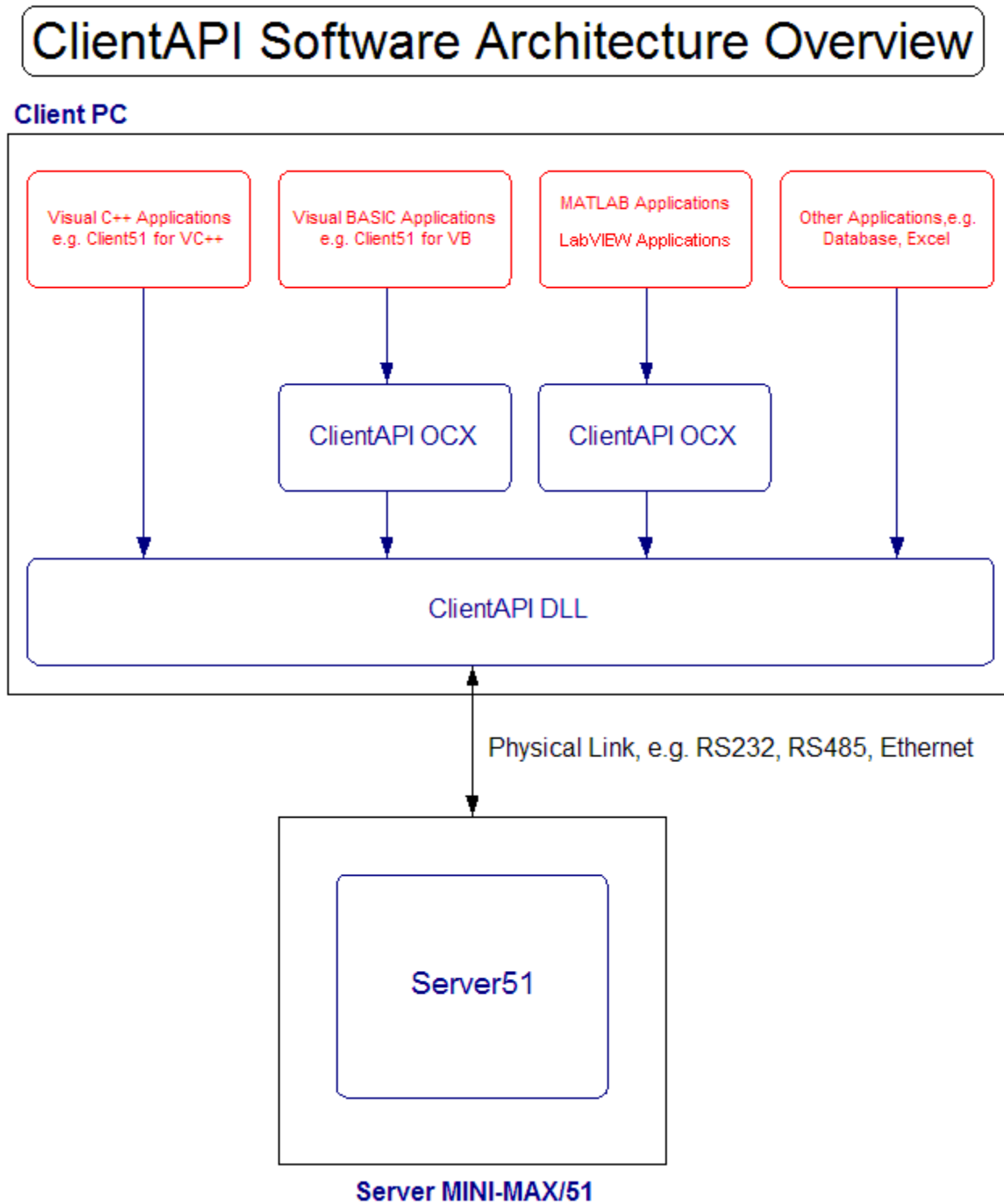


Figure 1

This mechanism allows the Windows PC to access all the resources and peripherals on our microcontroller boards using popular PC-based languages such as Visual BASIC, Visual C++, MATLAB™, LabVIEW™, Excel and others for the purpose of easily building your own data acquisition, instrumentation or control system in a language of your choice.

ClientAPI comes in two flavors:

- As an ActiveX Control (***ClientAPI.ocx***)
- As a Dynamically Linked Library (DLL: ***ClientAPI.lib*** and ***ClientAPI.dll***)

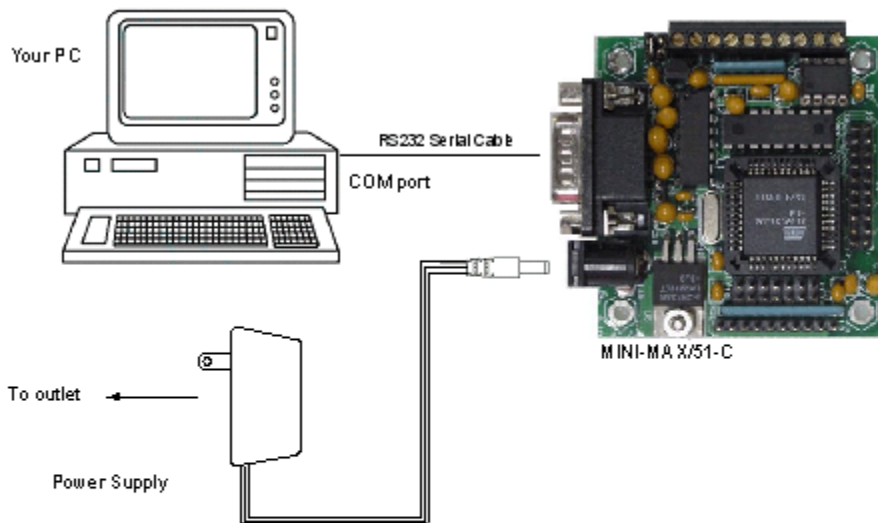
DLL is more suitable for use with Visual C++ applications where as ActiveX Control is the method of choice for Visual BASIC, LabVIEW™ and MATLAB™ developers.

Hardware Setup

1. Place the MINI-MAX/51-C2 Microcontroller board on a clean, non-conductive surface.
2. Connect the provided 6VDC power supply plug to the power jack on the MINI-MAX/51-C2. Do not connect the power supply to the outlet yet.

CAUTION: Do not use a power supply other than one that is supplied or approved by BiPOM Electronics. Use of another power supply voids the warranty and may permanently DAMAGE the board or the computer to which the board is connected !!!

4. Connect the MINI-MAX/51-C2 to an available serial port on the PC using the supplied serial cable as shown below.
5. Connect the 6VDC power supply to a suitable wall outlet.



3. Software Setup

Server (board) side Setup

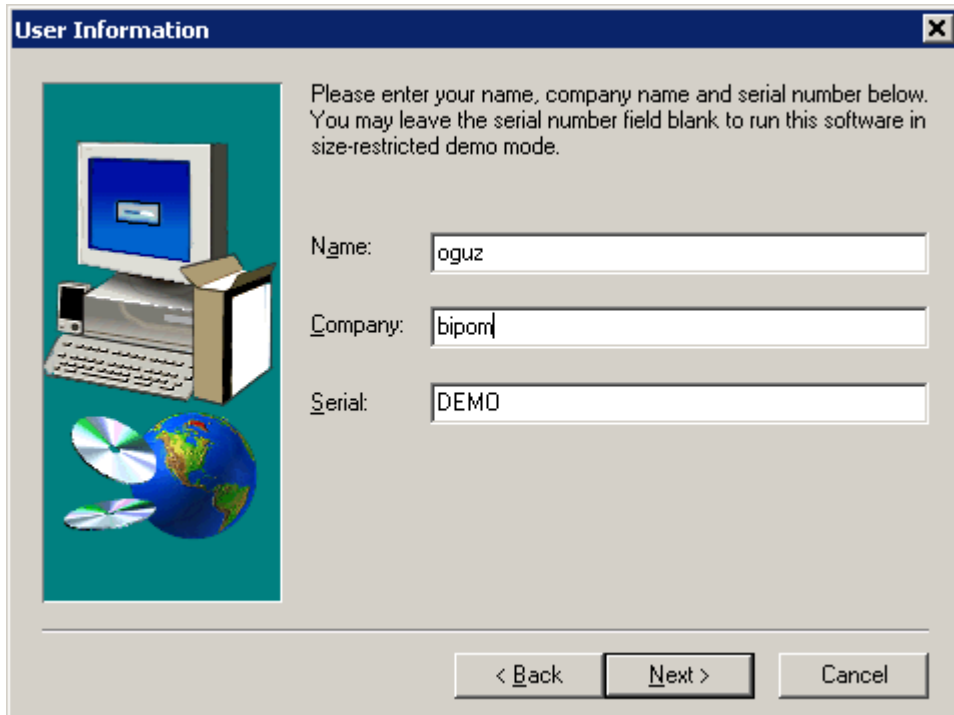
Download and install ClientAPI from BiPOM website:

<http://www.bipom.com/products/us/2899953.html>

Download and install Micro C 8051 Development System from BiPOM website:

http://www.bipom.com/8051dev_down.php

You can install this package as demo when asked for serial number during installation:



User Information

Please enter your name, company name and serial number below.
You may leave the serial number field blank to run this software in size-restricted demo mode.

Name:

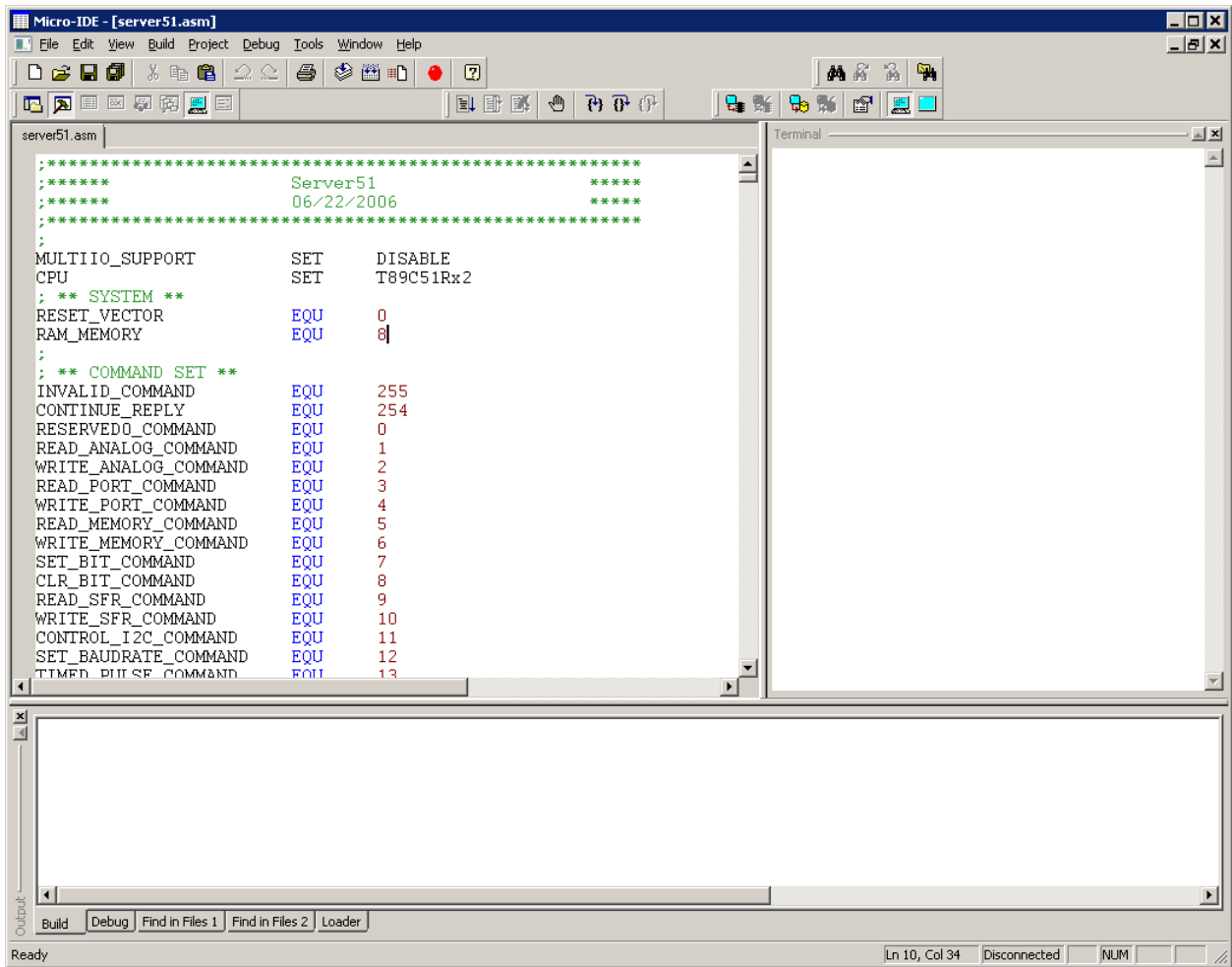
Company:

Serial:

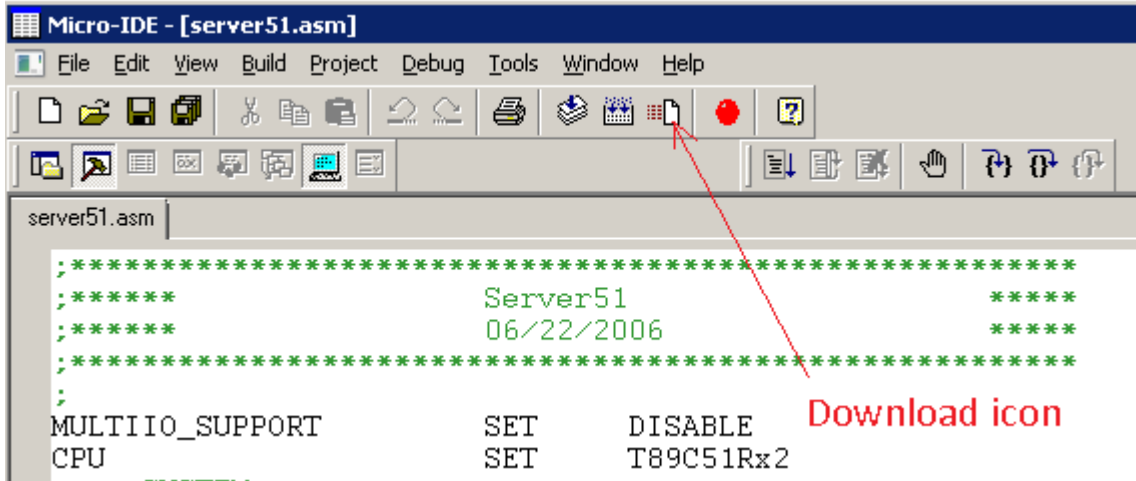
< Back Next > Cancel

After the above installations are complete, double click server51.prj file under C:\bipom\devtools\ClientAPI\Server51 to open the Server51 project.

This will start Micro-IDE and open the Server51 project.



While the MINI-MAX board is connected to your PC and powered, click on the Download icon to download the Server51 executable to the board:



After the download completes, exit Micro-IDE and the board is ready to use with LabVIEW.

Client (PC) side Setup

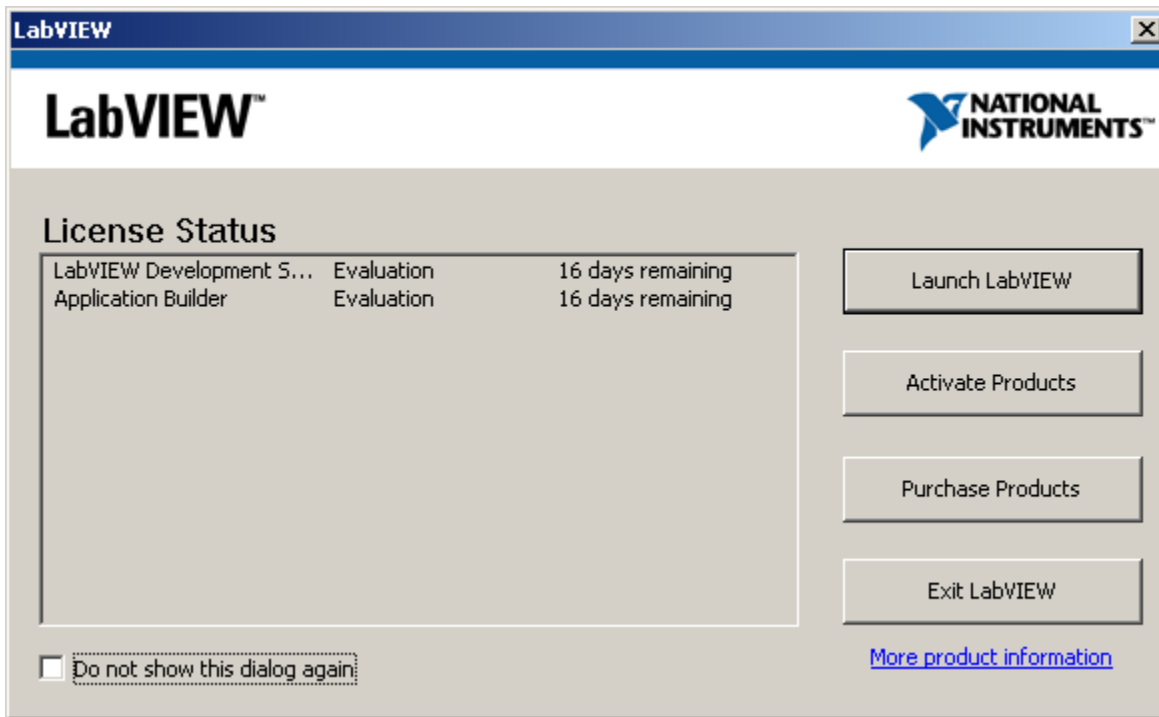
If you have not already done so, please install LabVIEW™ on your PC. If you do not have the full license for LabVIEW™, you can download the trial version from National Instruments™ website (www.ni.com)

The base package of LabVIEW™ is sufficient for use with ClientAPI. No special add-ons are required.

Start LabVIEW™ from the Start menu. In this document, we use LabVIEW™ 2009; other versions should work but have not been tested:



LabVIEW™ License Status window may appear depending on your license.

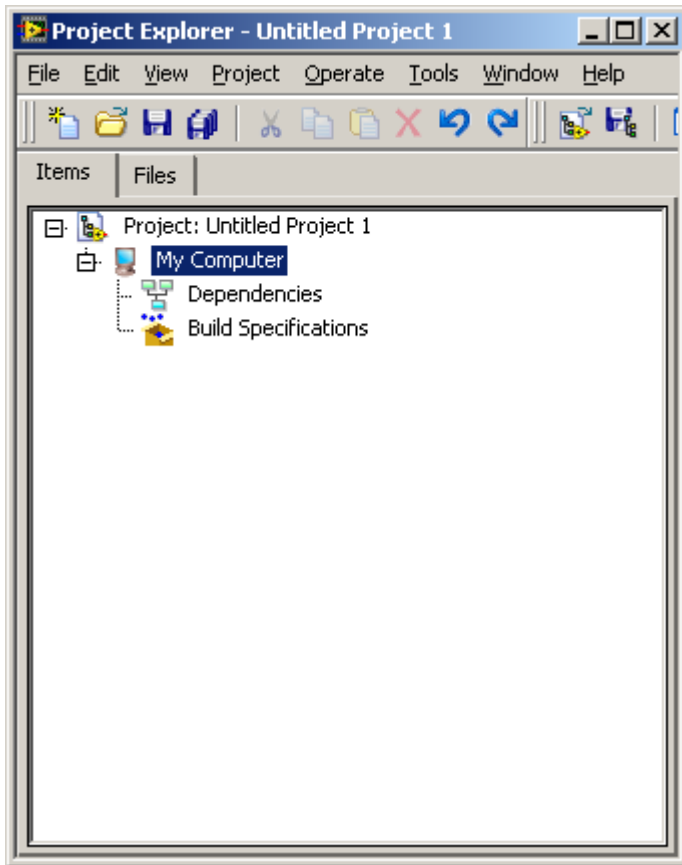


Click on Launch LabVIEW™ button. The startup screen will appear:



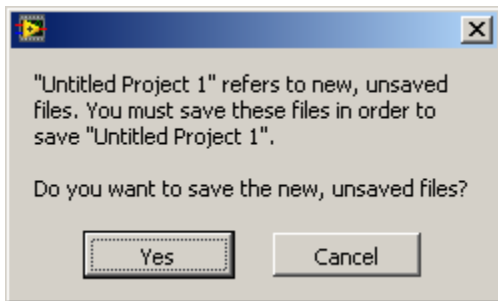
Now that we have started LabVIEW™, we will create a simple LabVIEW™ program to read voltage values from an analog input channel 0 of MINI-MAX/51-C2 board and plot these values on a chart. The program will blink a green LED on the screen each time a voltage value is acquired from the board. The program will also display communications status and other error/status information.

Click on Empty Project under New section to create a new project:

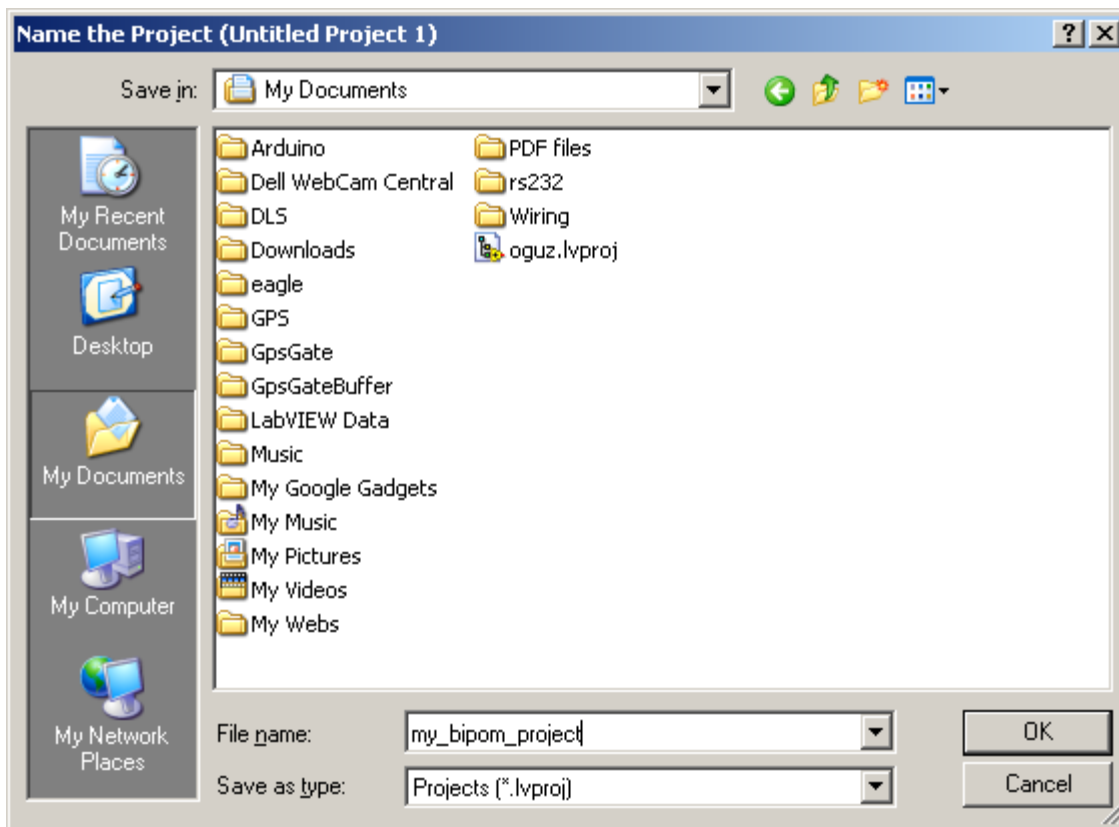


Select File->New VI to create a new VI (Virtual Instrument).

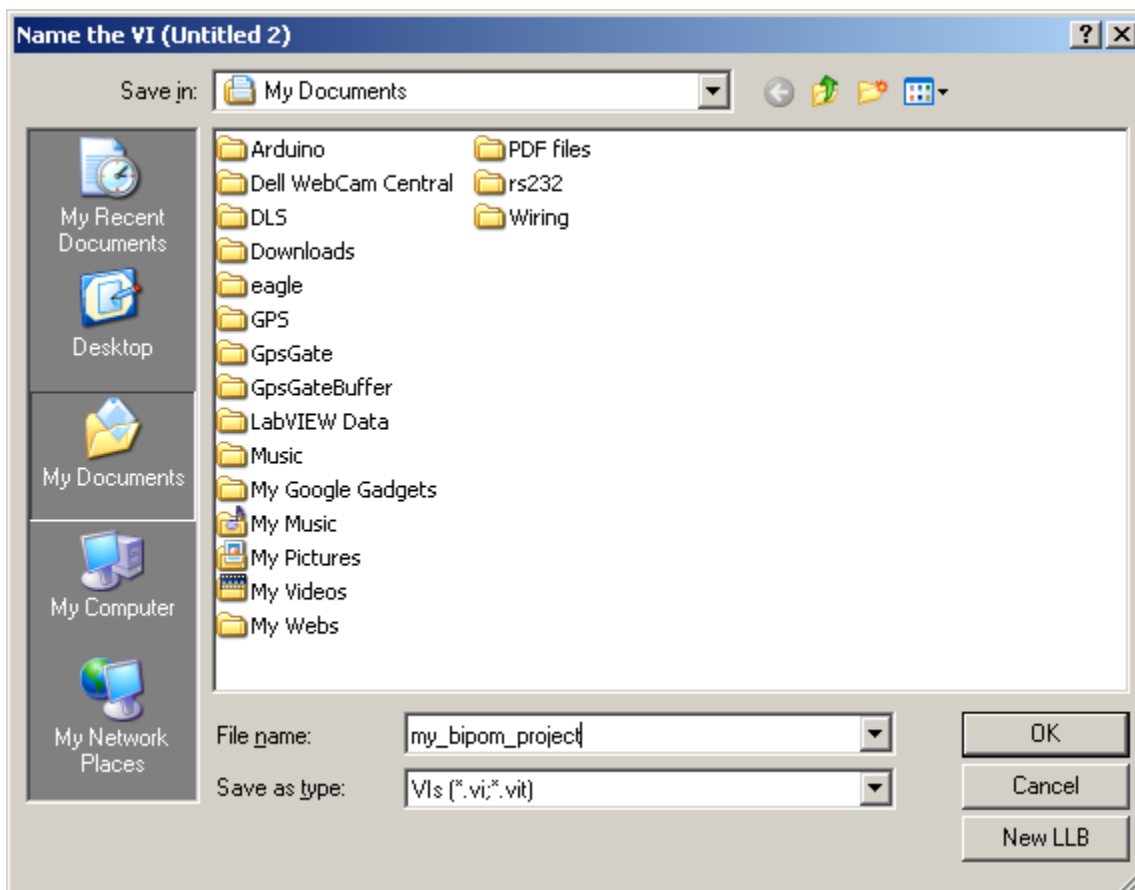
First, save this new project using File-Save As. You will see this message:



Click Yes and give a name to your project:



Click OK. Also, give a name to your VI:

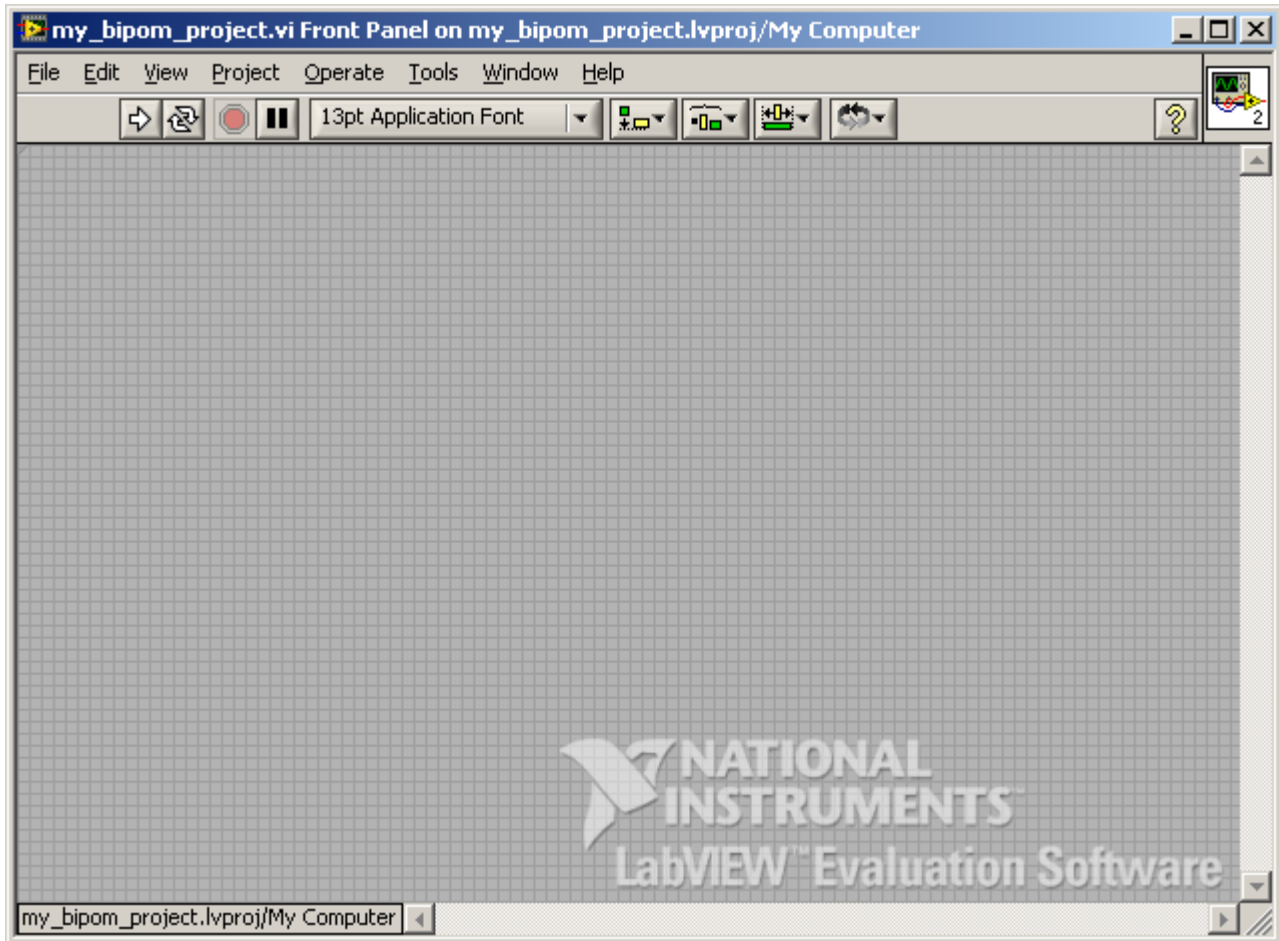


Click OK. This will create the following files in the folder where you saved the project:

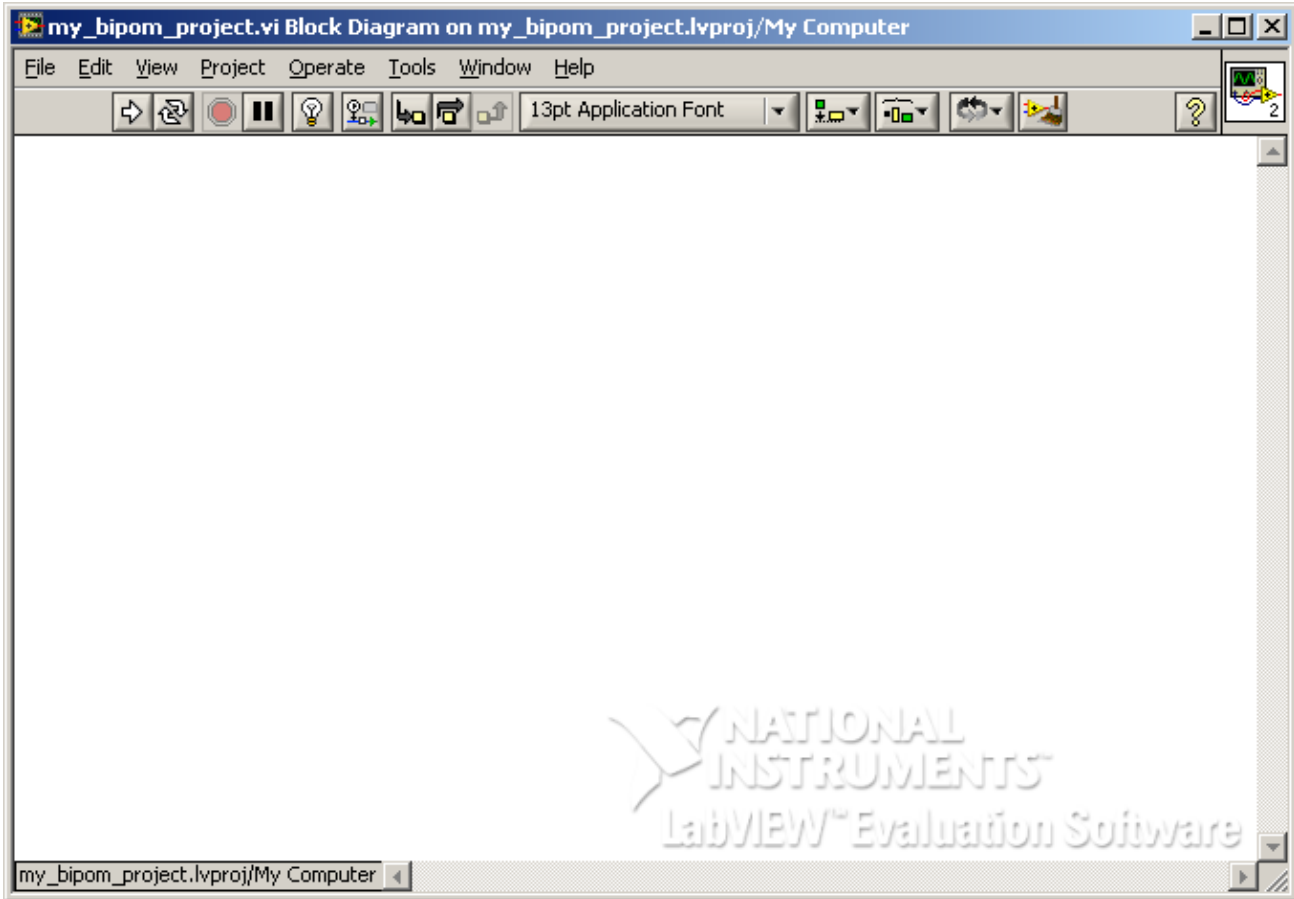
my_bipom_project.aliases
my_bipom_project.lvpls
my_bipom_project.lvproj
my_bipom_project.vi

Your VI consists of 2 windows:

- **Front Panel:** This is the user interface (display) for your LabVIEW™ program:

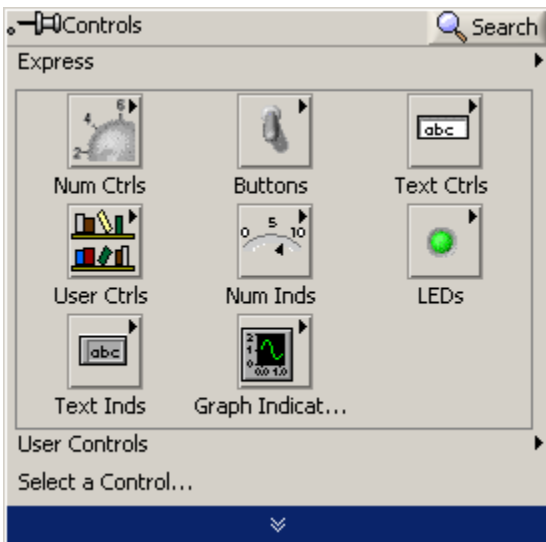


- **Block Diagram:** This is the actual program that consists of blocks and their interconnections.

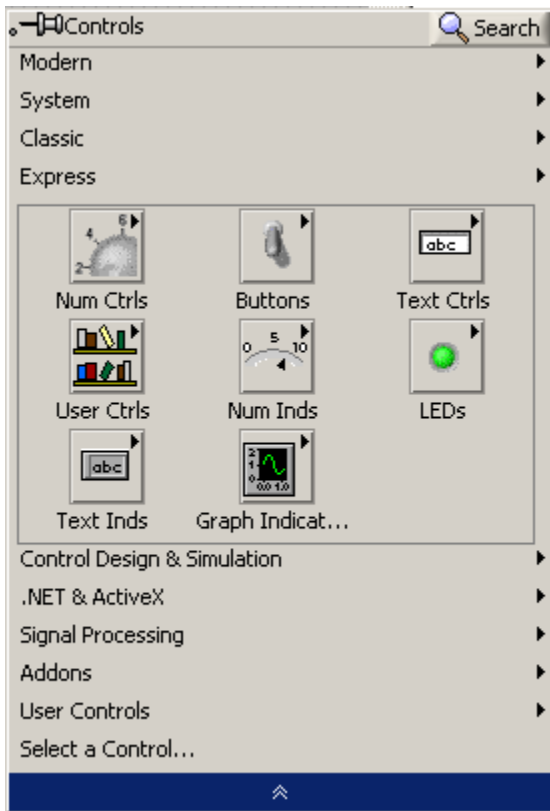


Now, we will start entering our program into the Block Diagram window:

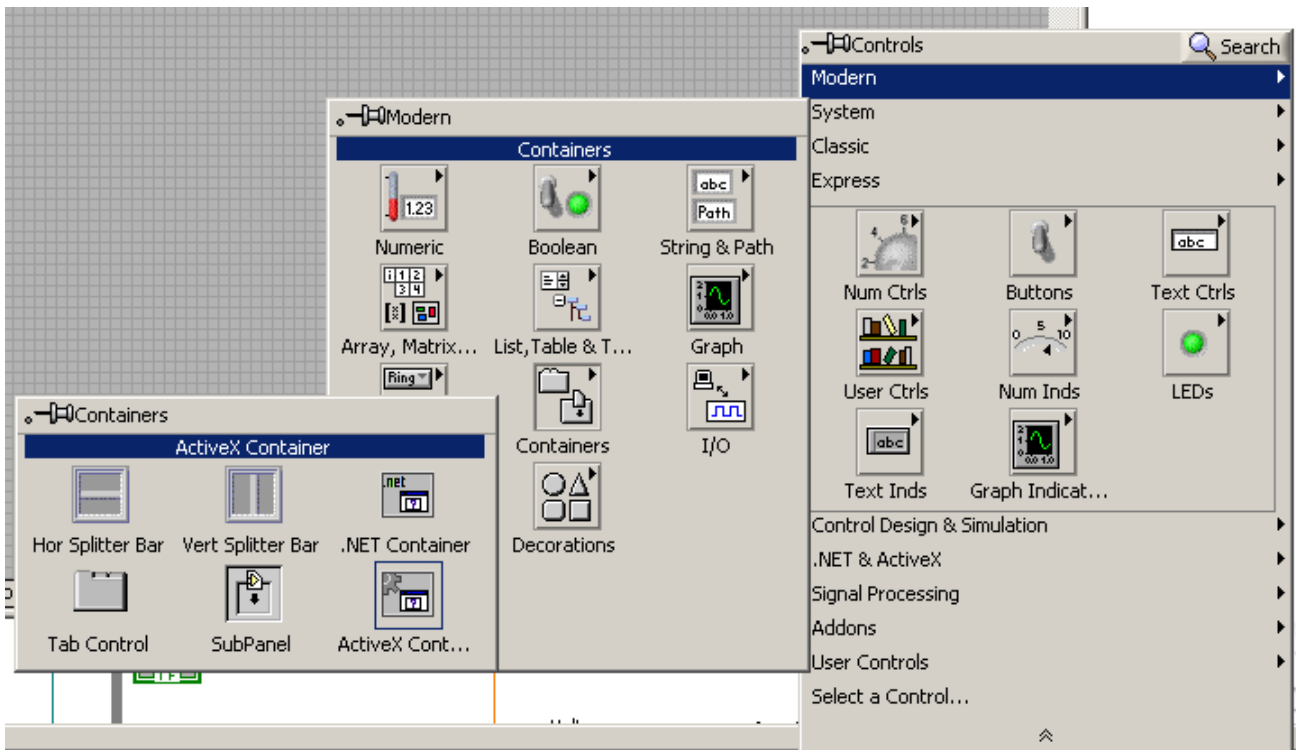
Insert the ActiveX control for ClientAPI into the Front Panel. To do this, right click on Front Panel. This will bring up **Controls** window:



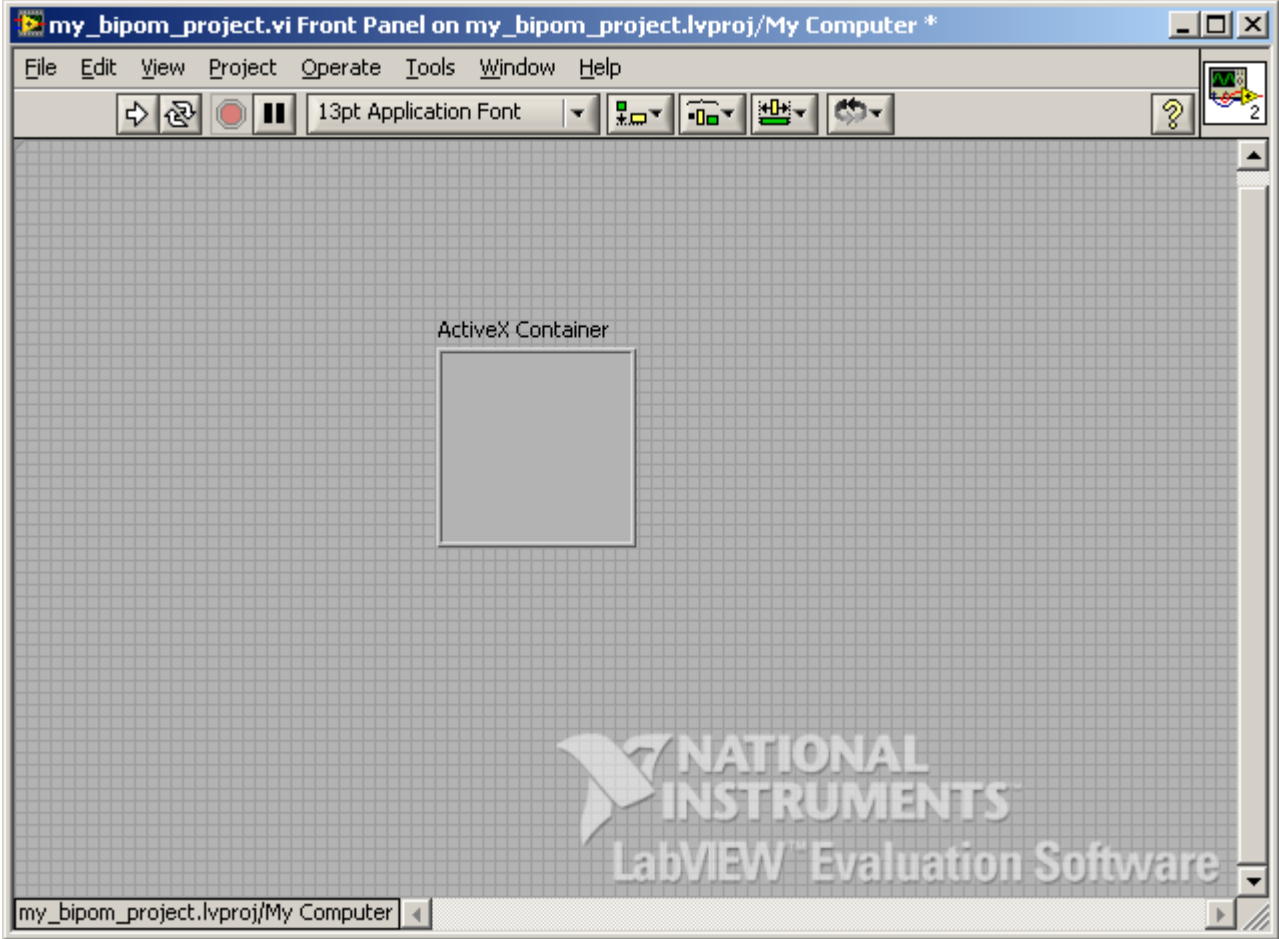
Click on the down arrow at the bottom of this window to expand the selections:



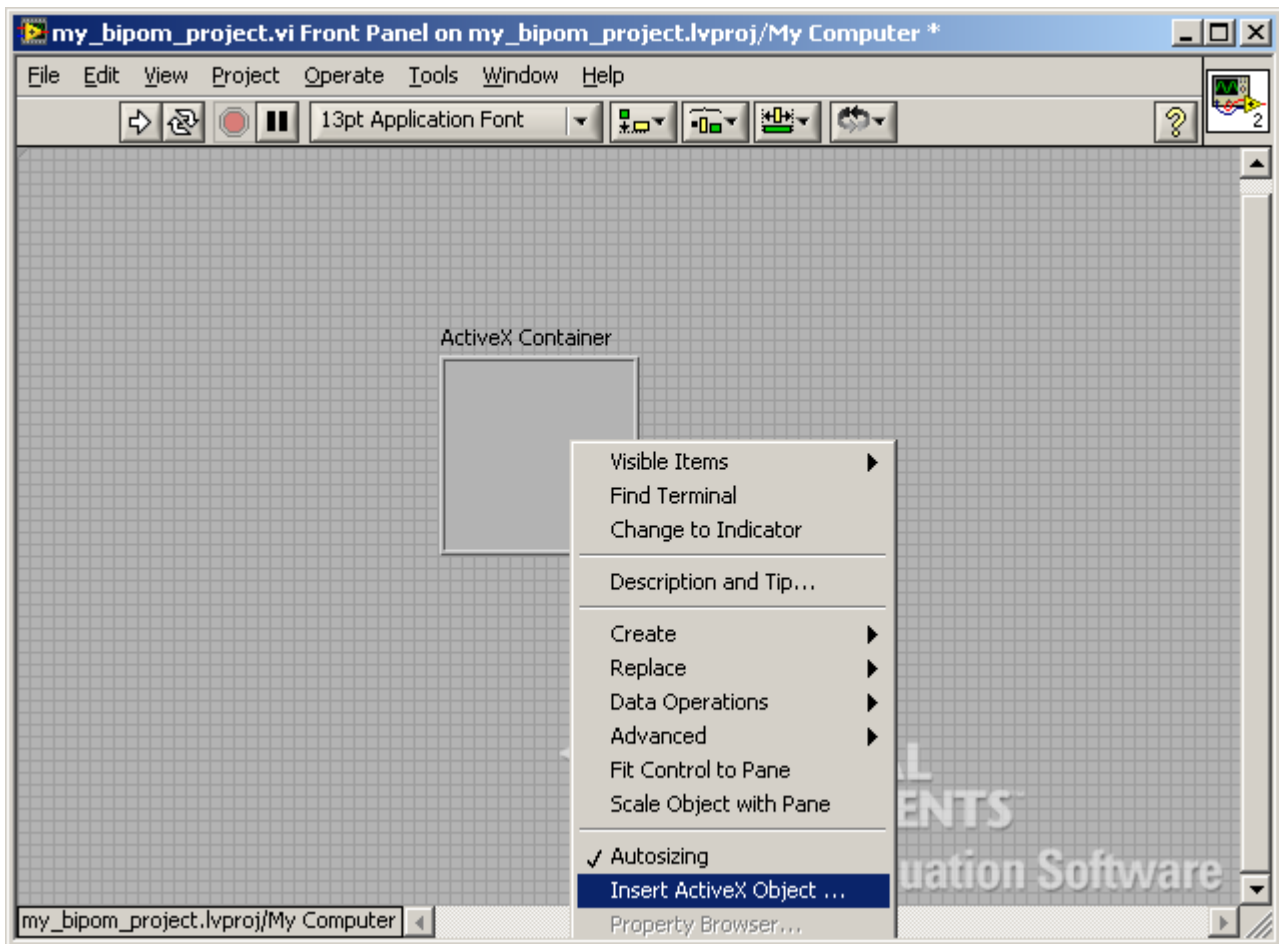
Select **Modern** and then **Containers** and then **ActiveX Controls**:



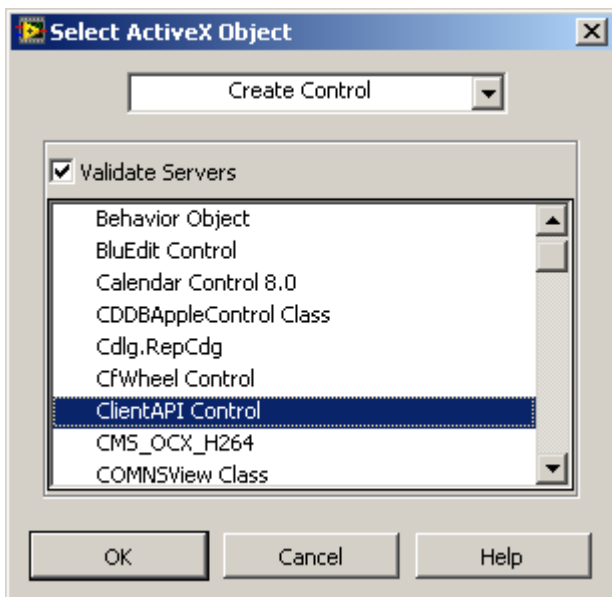
Place the ActiveX container on the Front Panel:



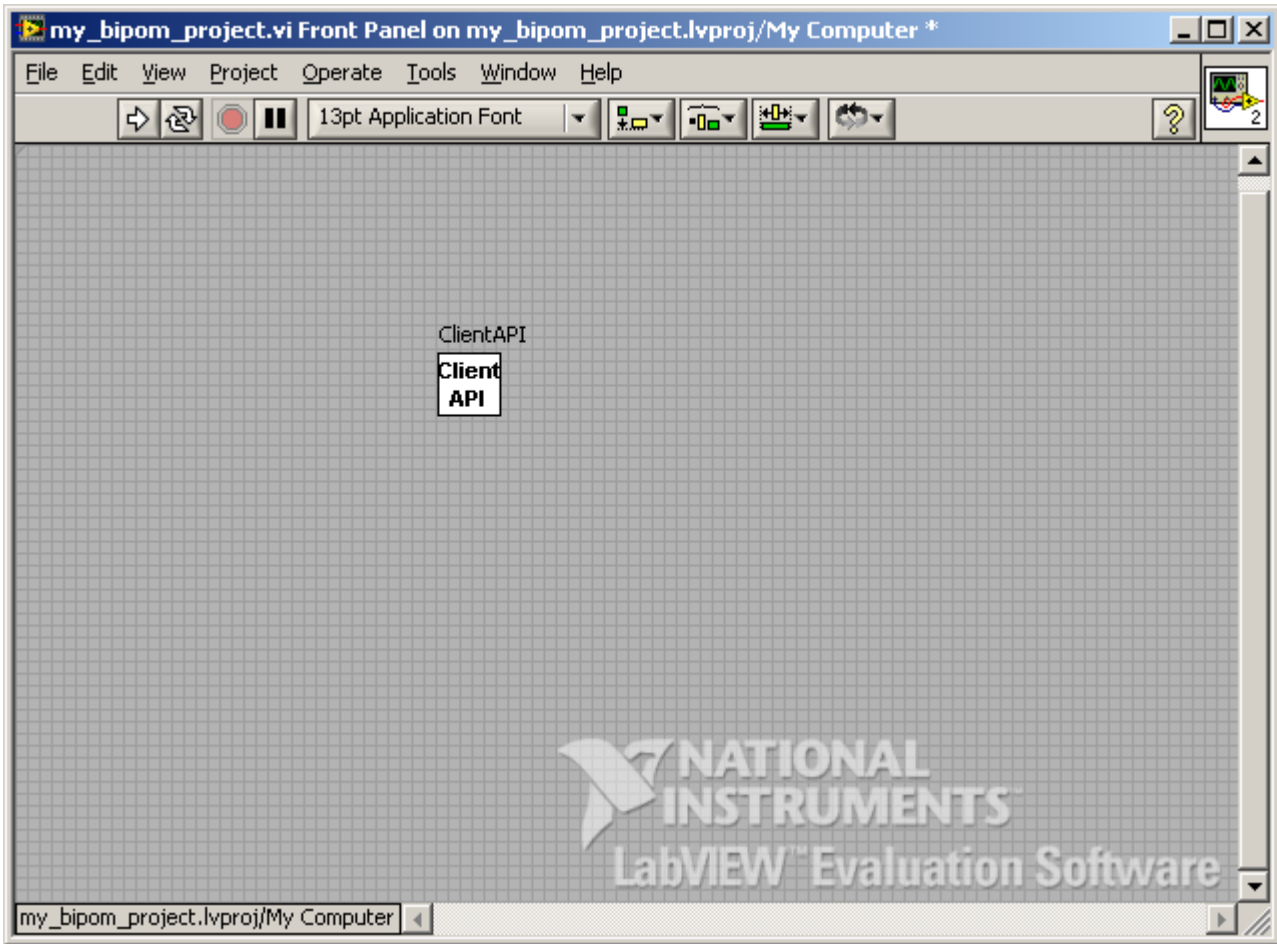
Right click on ActiveX Container and select **Insert ActiveX Object**.



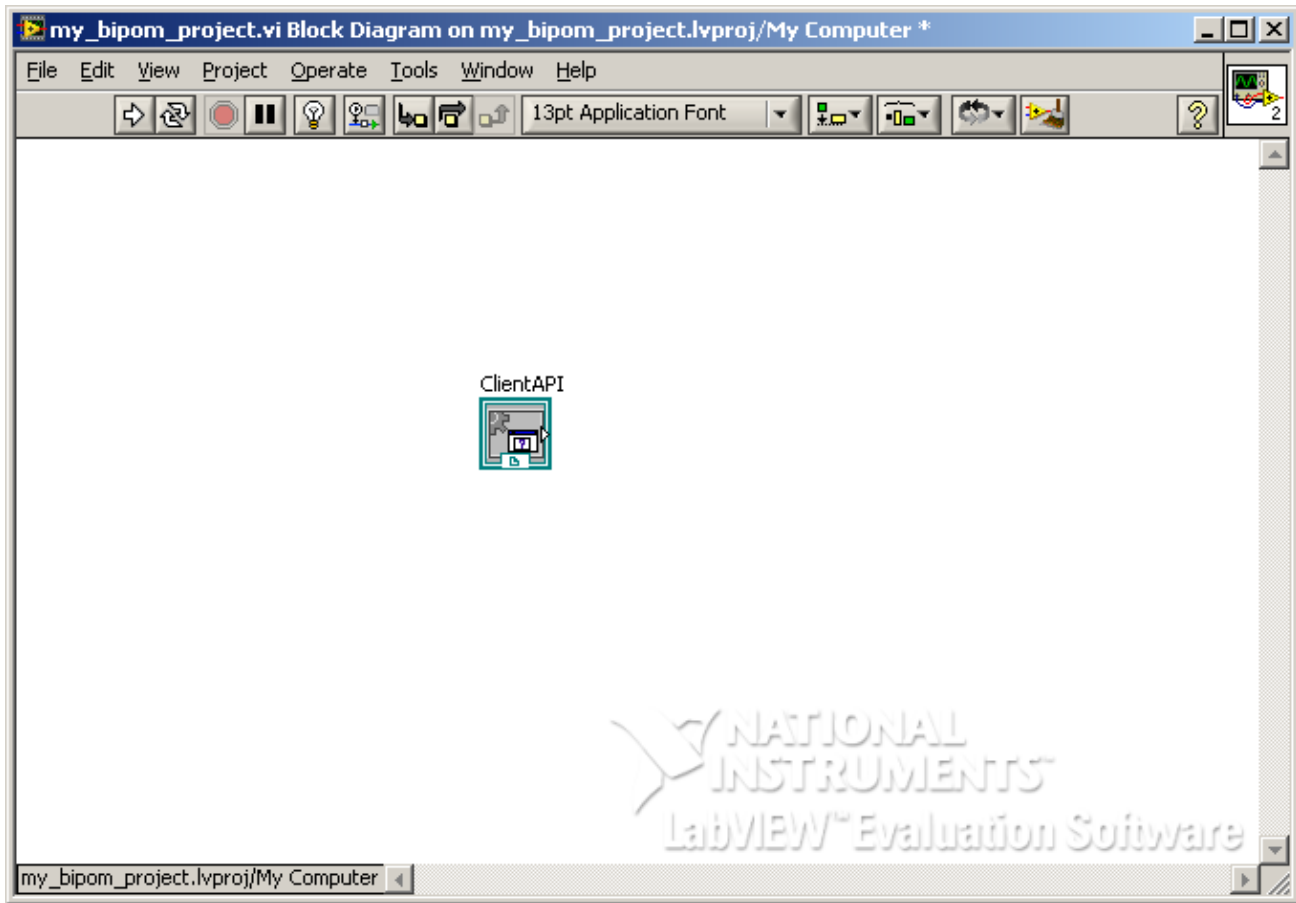
Select **ClientAPI Control** from the list:



Click OK. Resize the ClientAPI object as needed:



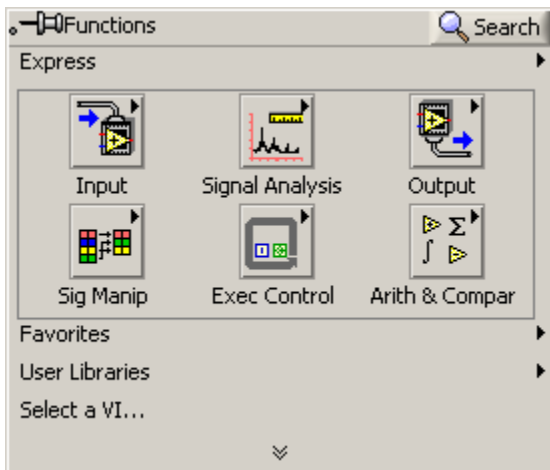
In Front Panel, select **Window->Show Block Diagram**. This will bring up the Block Diagram window. You will notice that the ClientAPI control has also been added to Block Diagram:



Save your work so far by selecting **File->Save All** from either Block Diagram or Front Panel window.

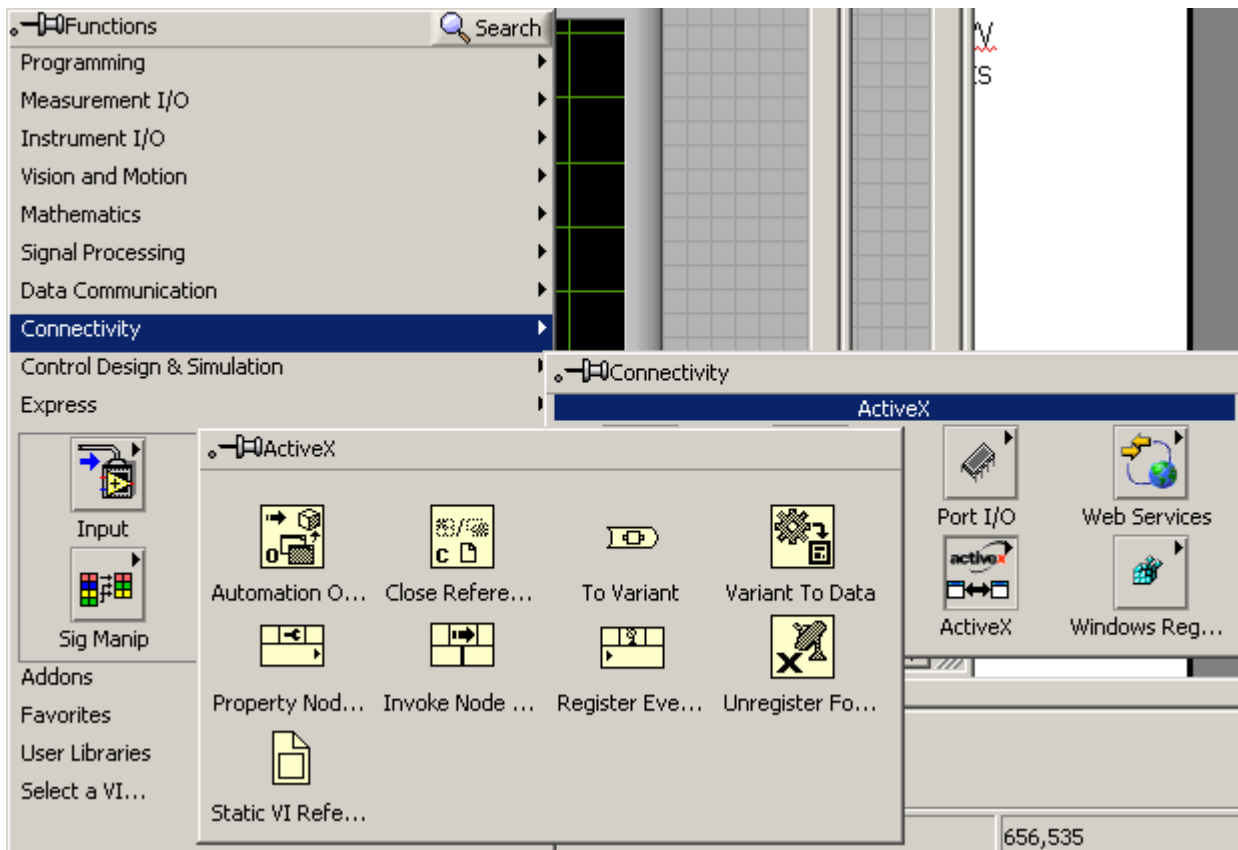
So far, we have successfully inserted the ClientAPI ActiveX Control into our LabVIEW™ program. As next step, we will insert a Property Node that gives us access to the objects inside the ClientAPI ActiveX Control.

In Block Diagram window, right click on Block Diagram area and the **Functions** window will appear:

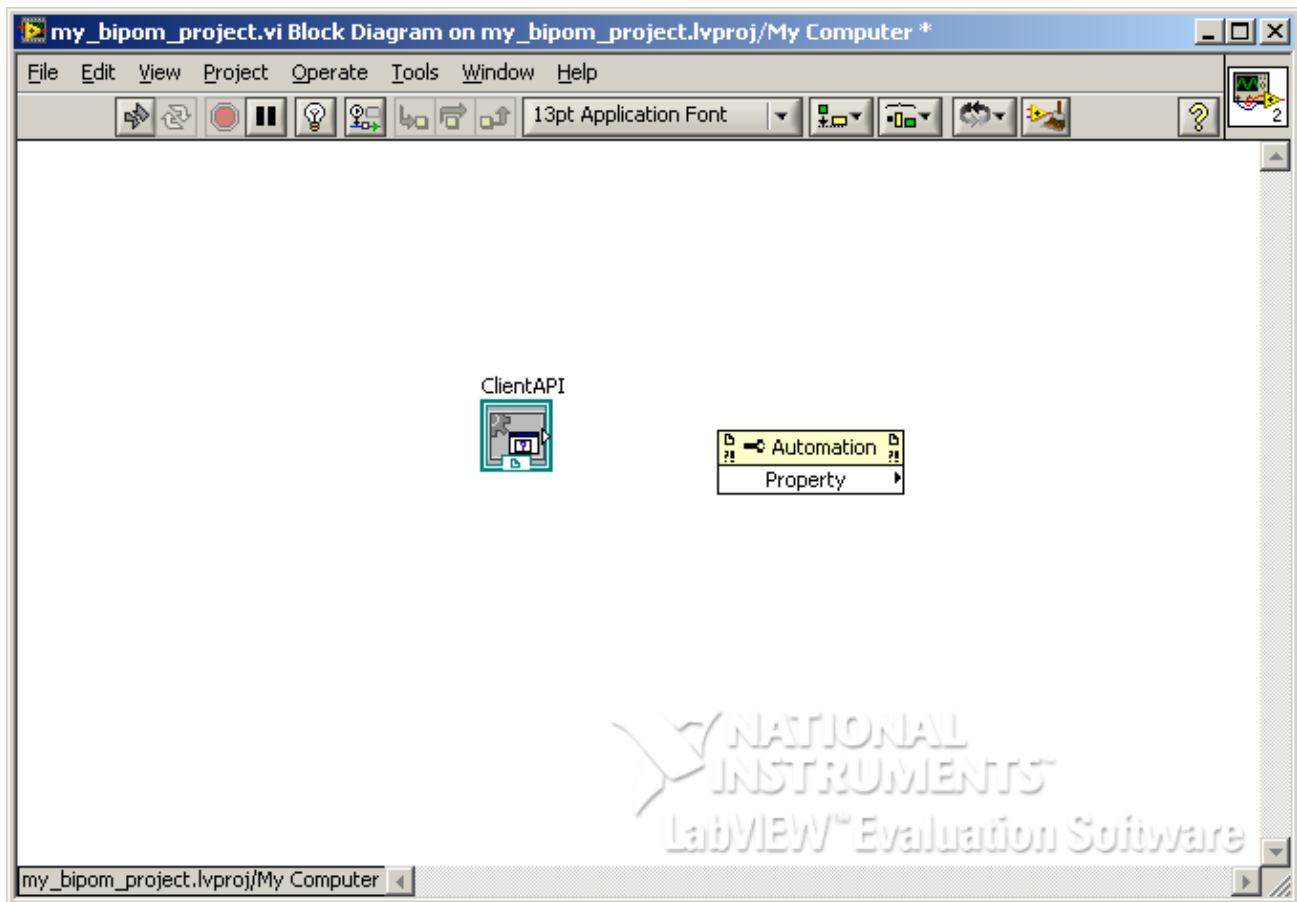


Click on the down arrow at the bottom of this window to expand the selections:

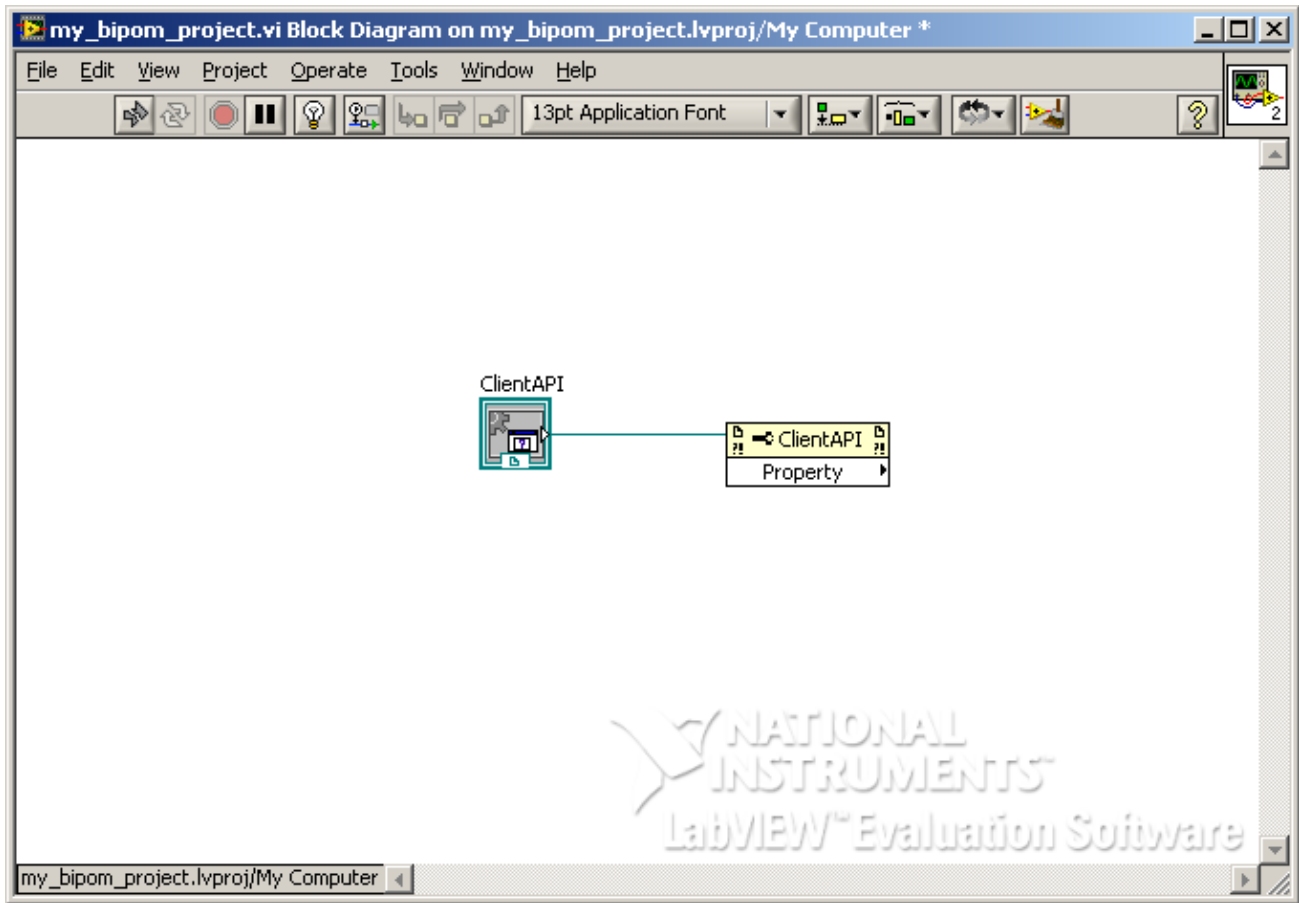
Select **Connectivity->ActiveX->Property Node**:



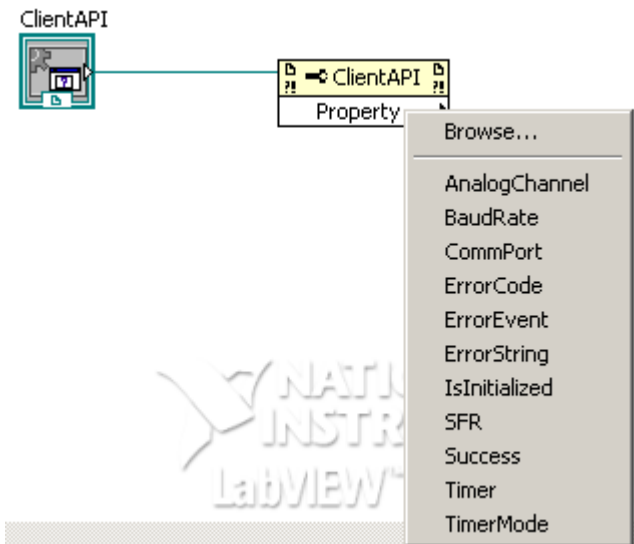
Drop the **Property Node** function onto the Block Diagram area:



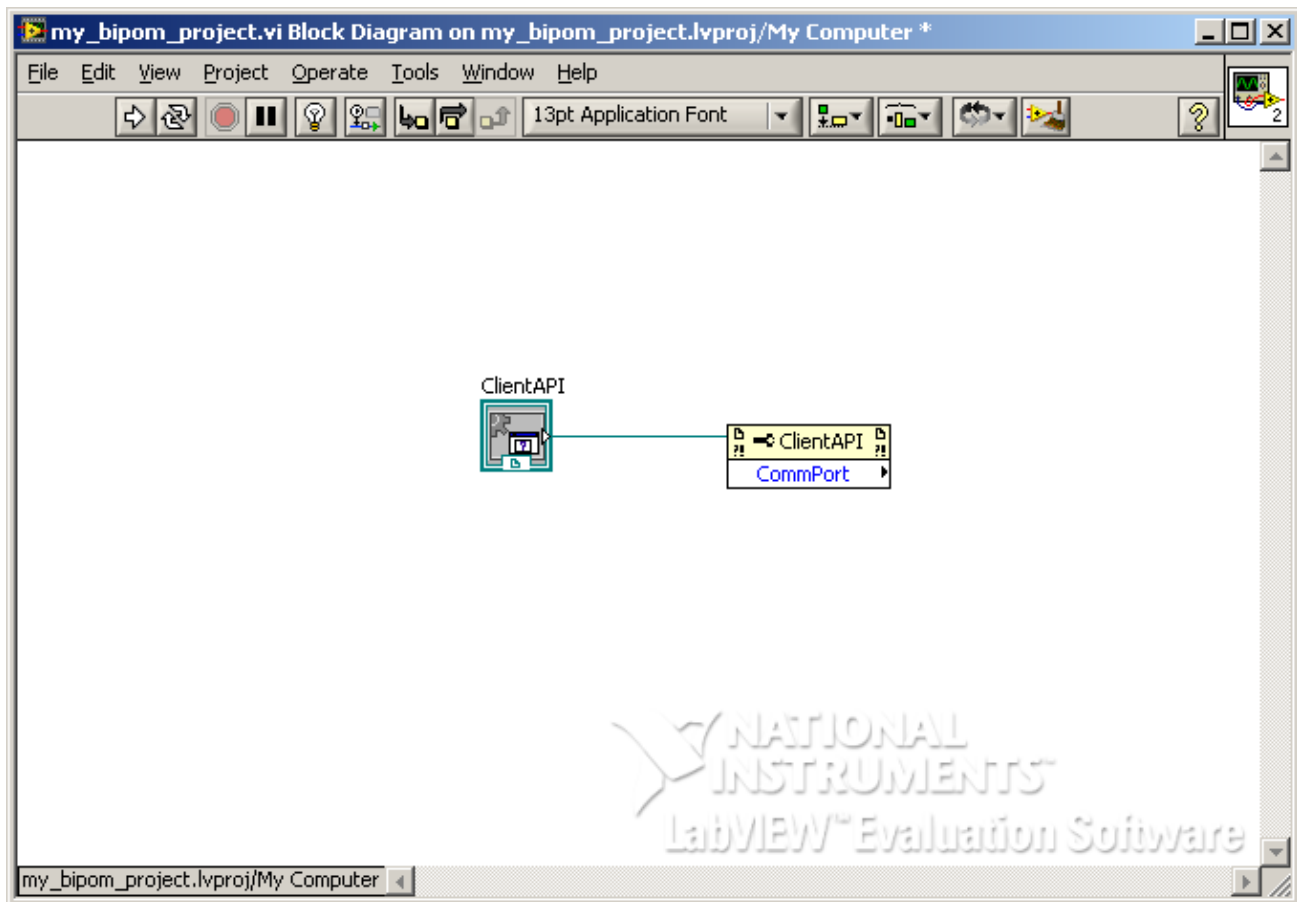
Connect ClientAPI ActiveX Control object to the **Property Node** using a wire. You will notice that the name of the **Property Node** will change to ClientAPI since it is now connected to and associated with ClientAPI ActiveX Control:



Left click on Property field. You will notice that the objects inside ClientAPI ActiveX control are now visible. J

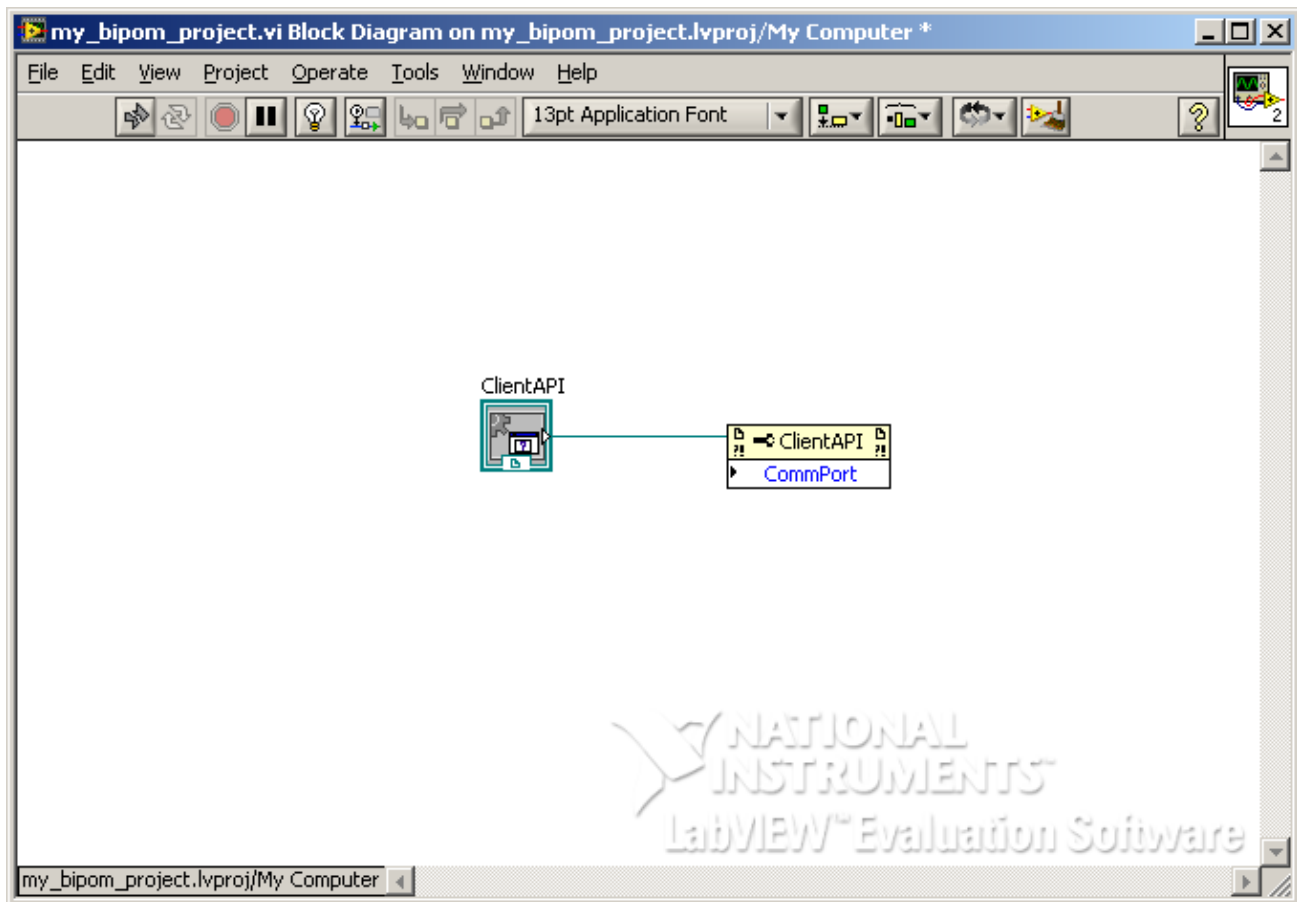


Select **CommPort** from the list. The name of the Property field will change to **CommPort**.



Right click on **CommPort** field and select **Change to Write**. This means that we will write to this field (and not read from it),

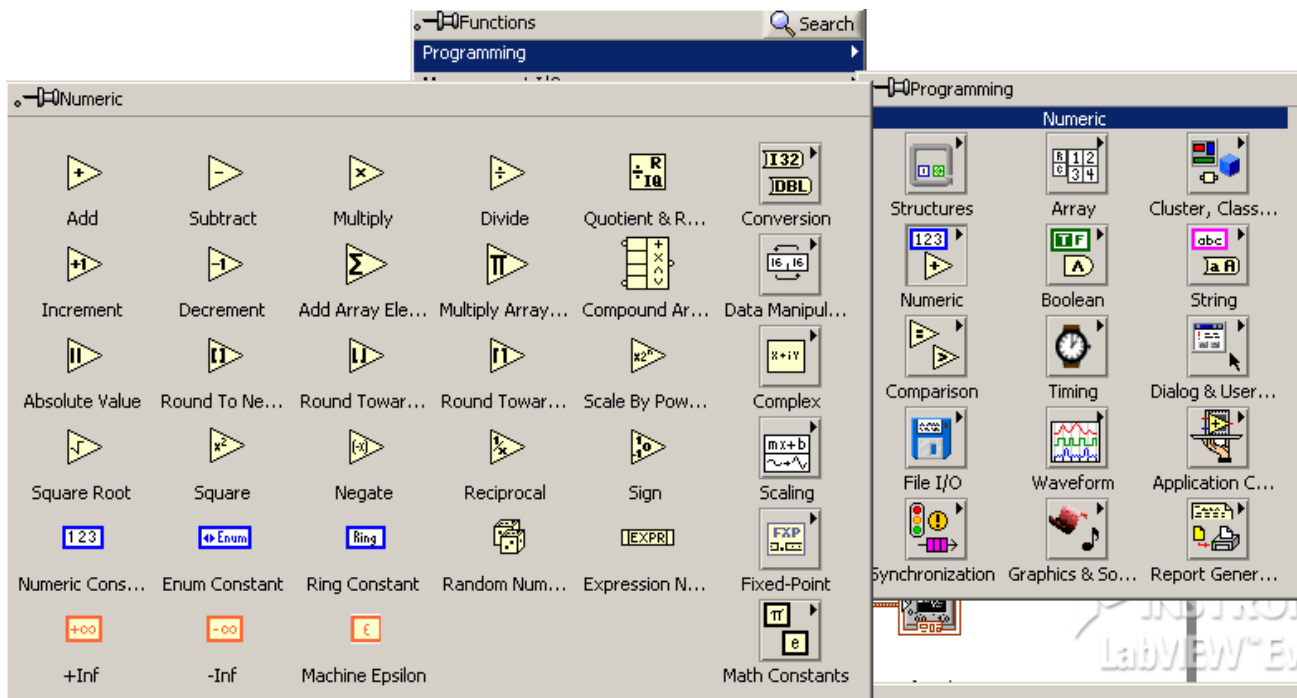
You will notice that the little arrow from the right side of **CommPort** field moves to the left side, indicating that it is a write-only field:



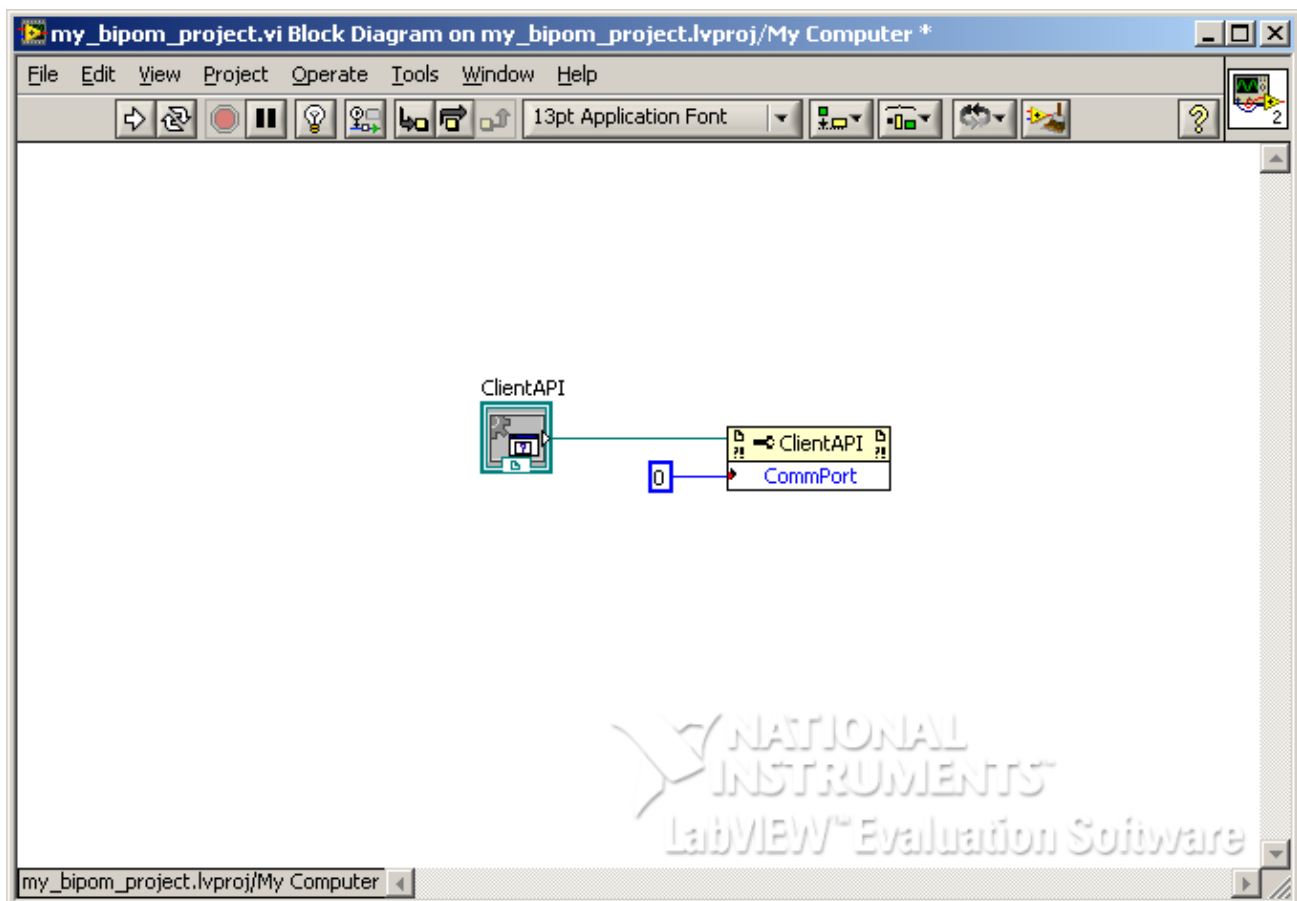
Now we can assign the COM port. This is the COM port that LabVIEW™ will use to communicate with the board. COM port numbering is 0-based as follows:

COM1: 0
COM2: 1
COM3: 2
COM4: 3

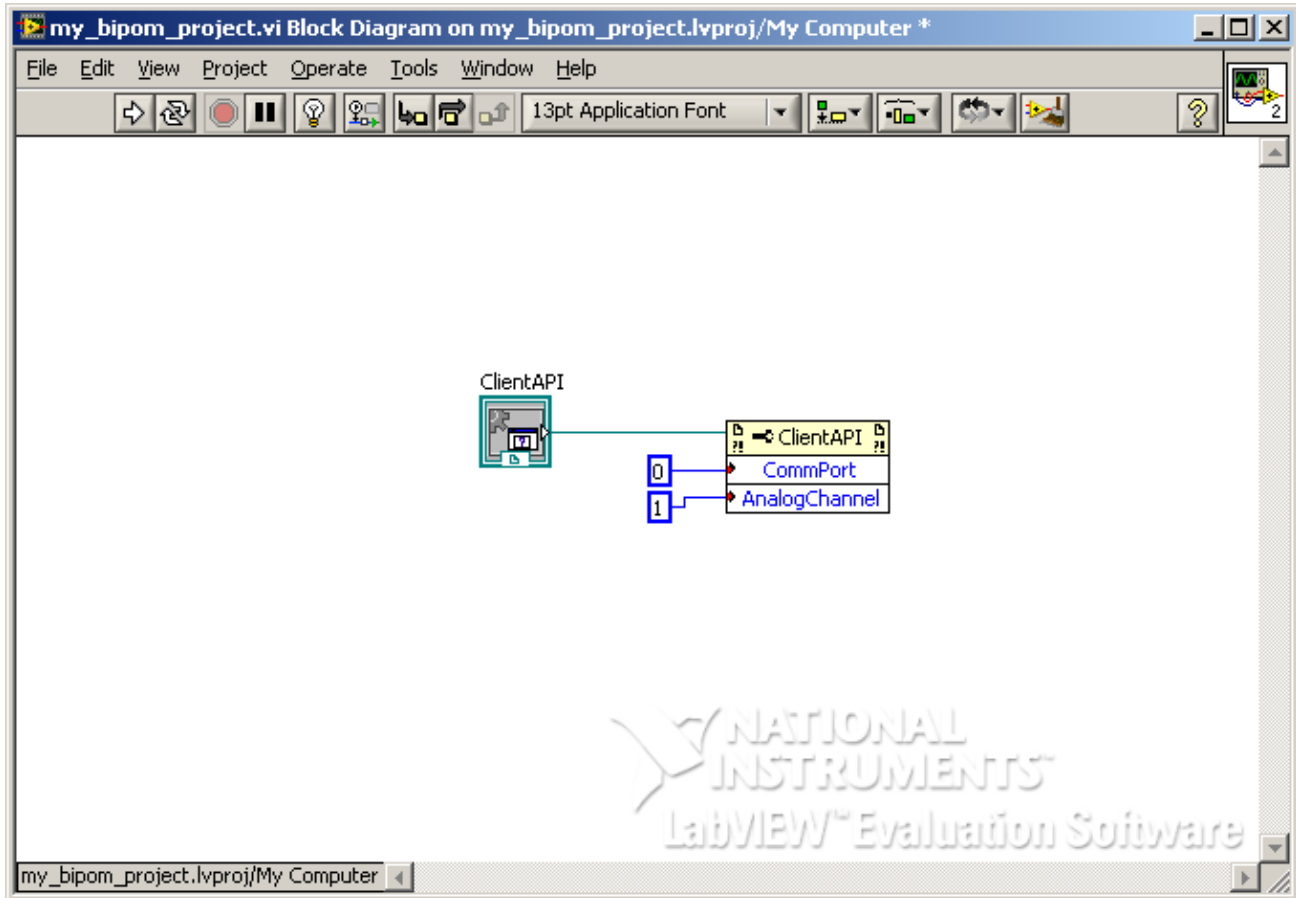
In this example, we will use COM1 so we assign a value of 0 to **CommPort** field. To do this, right click in Block Diagram area and select **Numeric Constant** under **Functions->Programming->Numeric**:



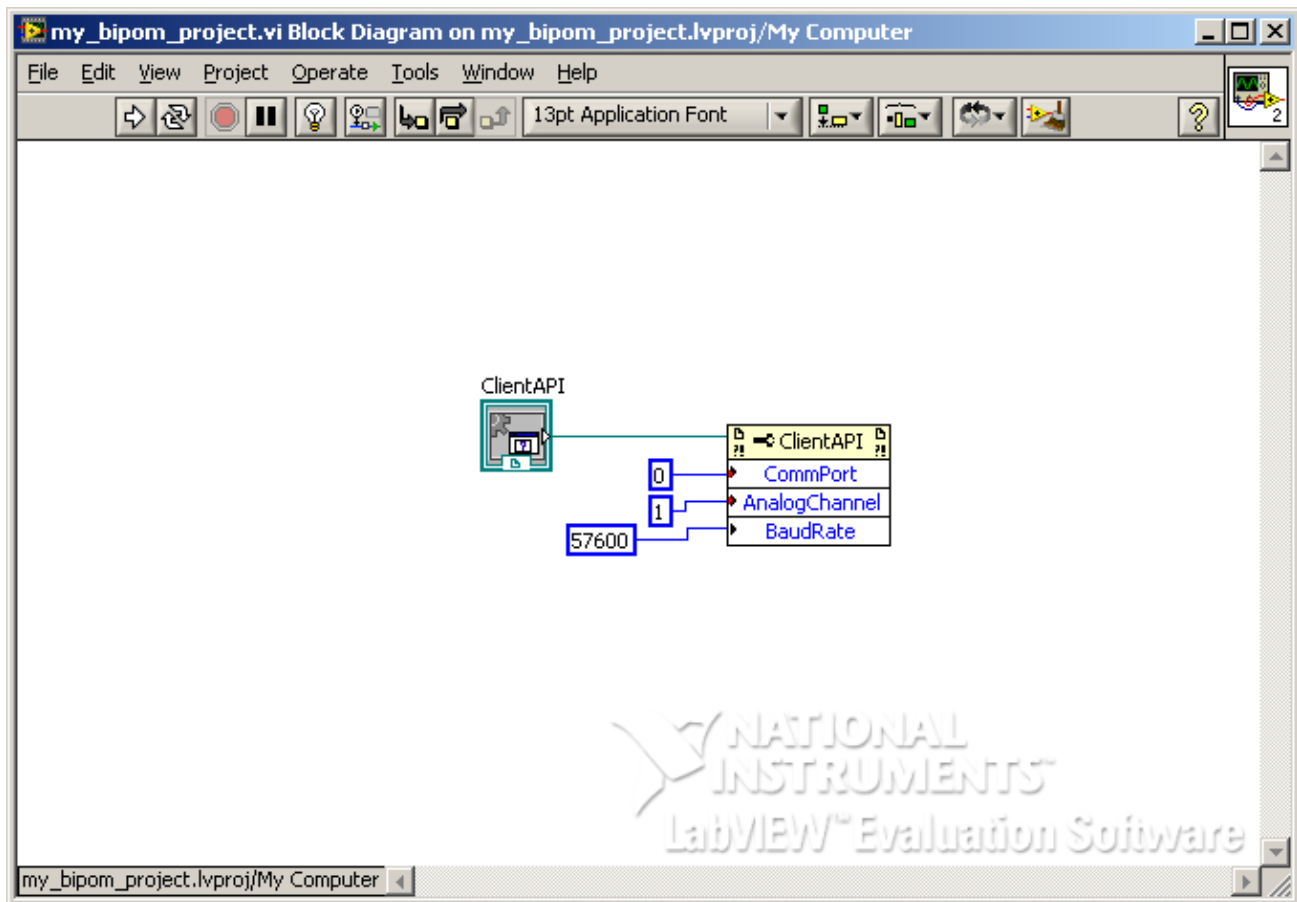
Drag and drop **Numeric Constant** onto the Block Diagram and assign it a value of 0. Also, connect the **Numeric Constant** to the **CommPort** field of **Property Node**:



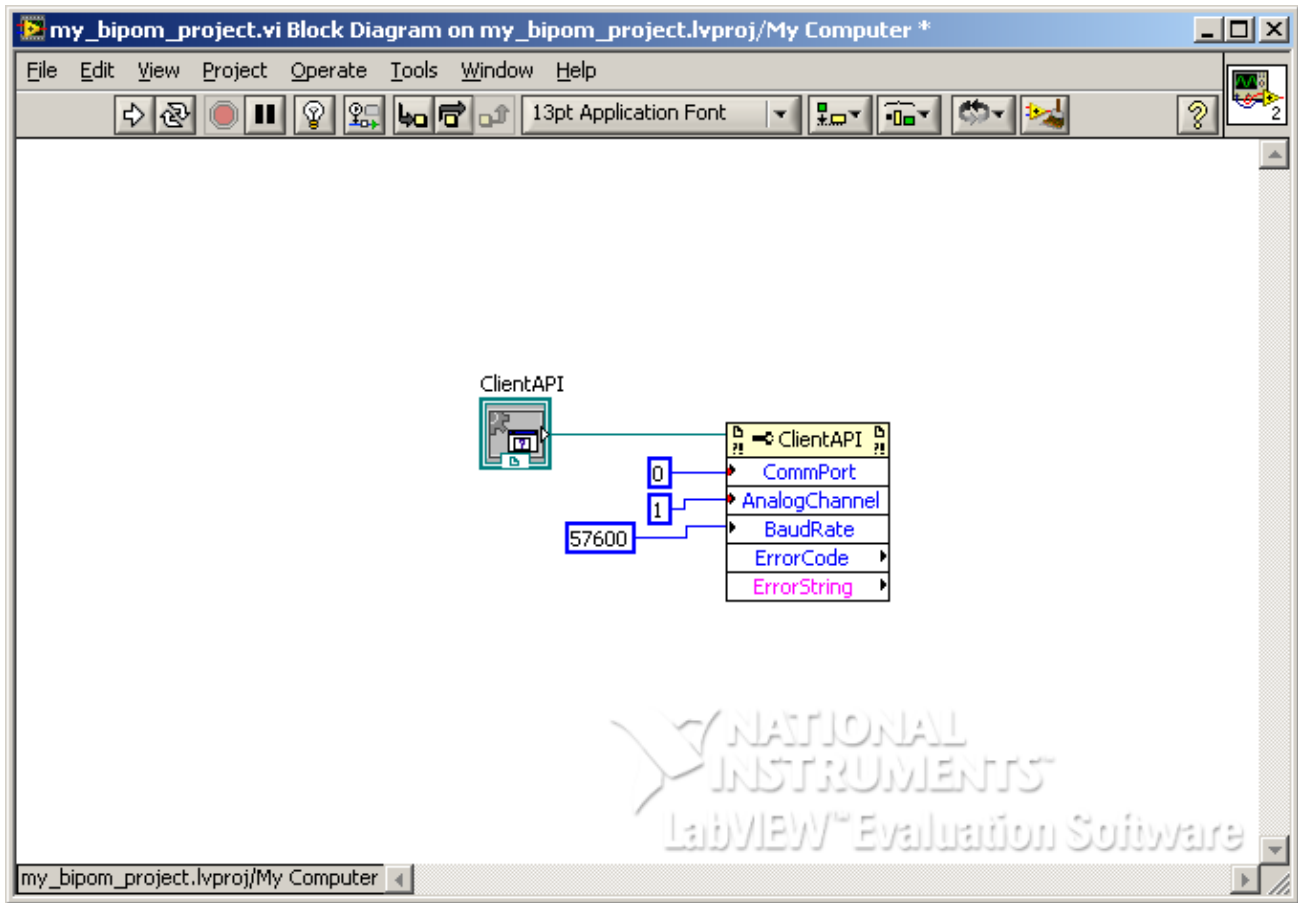
So far, we have successfully assigned a value of 0 to **CommPort** field. Now we want to add more fields to configure in our **Property Node**. Right click on **CommPort** field and select **Add Element** from the menu. This will add a second field, also with the name of **CommPort**. To change this, right click on the second **CommPort** field and choose the **Select Property** and then **AnalogChannel** from the menu. We will now configure the analog channel that we will make the measurement from. In this example, we will use the Analog Channel 1. So similar to how we assigned a numeric constant to **CommPort**, assign a numeric constant to **AnalogChannel** field:



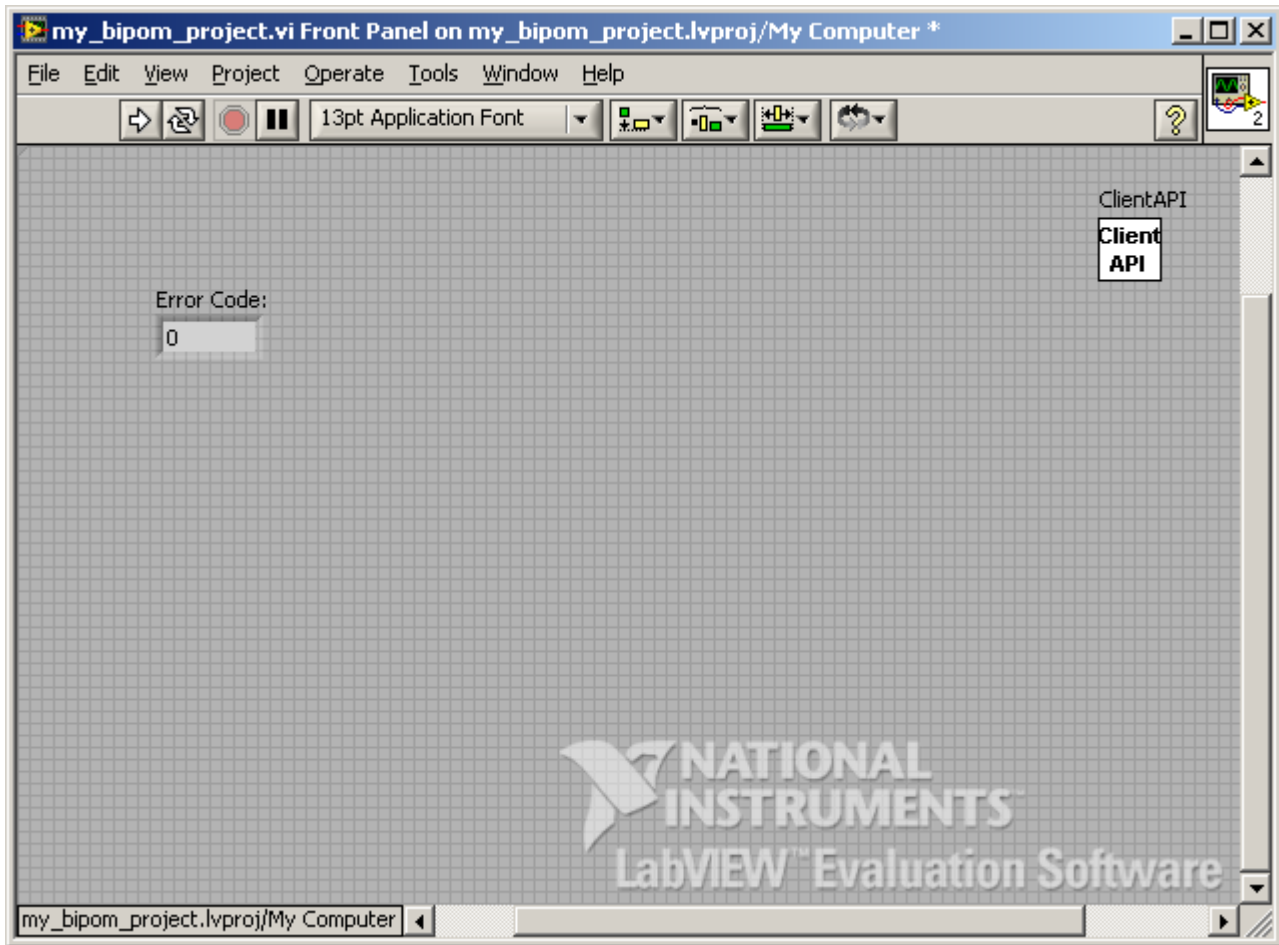
Similarly, add the field **BaudRate** and assign it a value of **57600**. By default, ClientAPI operates at 57600 baud when communicating with the board:



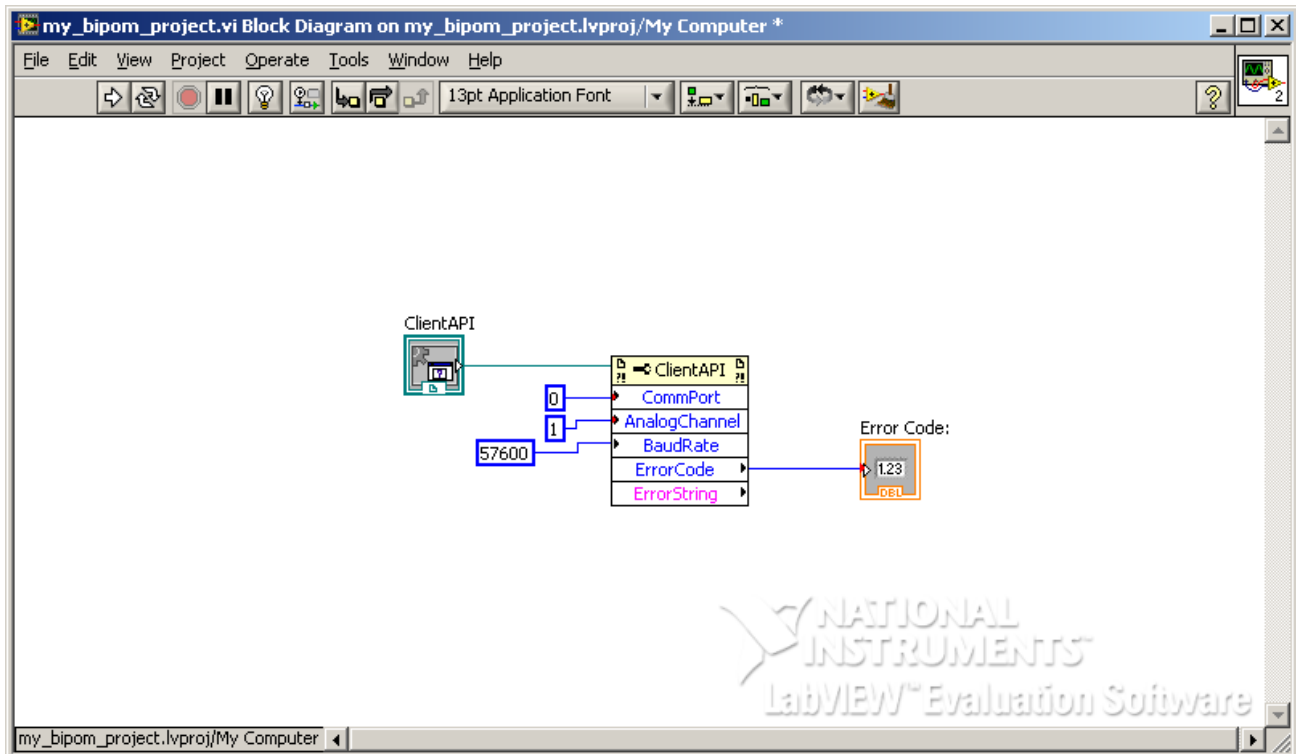
Add two more fields to the **Property Node**: **ErrorString** and **ErrorCode**. **ErrorCode** is the error code returned from ActiveX functions and **ErrorString** is human readable description of the error code. After adding these fields, change them to Read-only using **Change To Read** option from the right click menu. This is because we will not write to these two fields; we will only read from them:



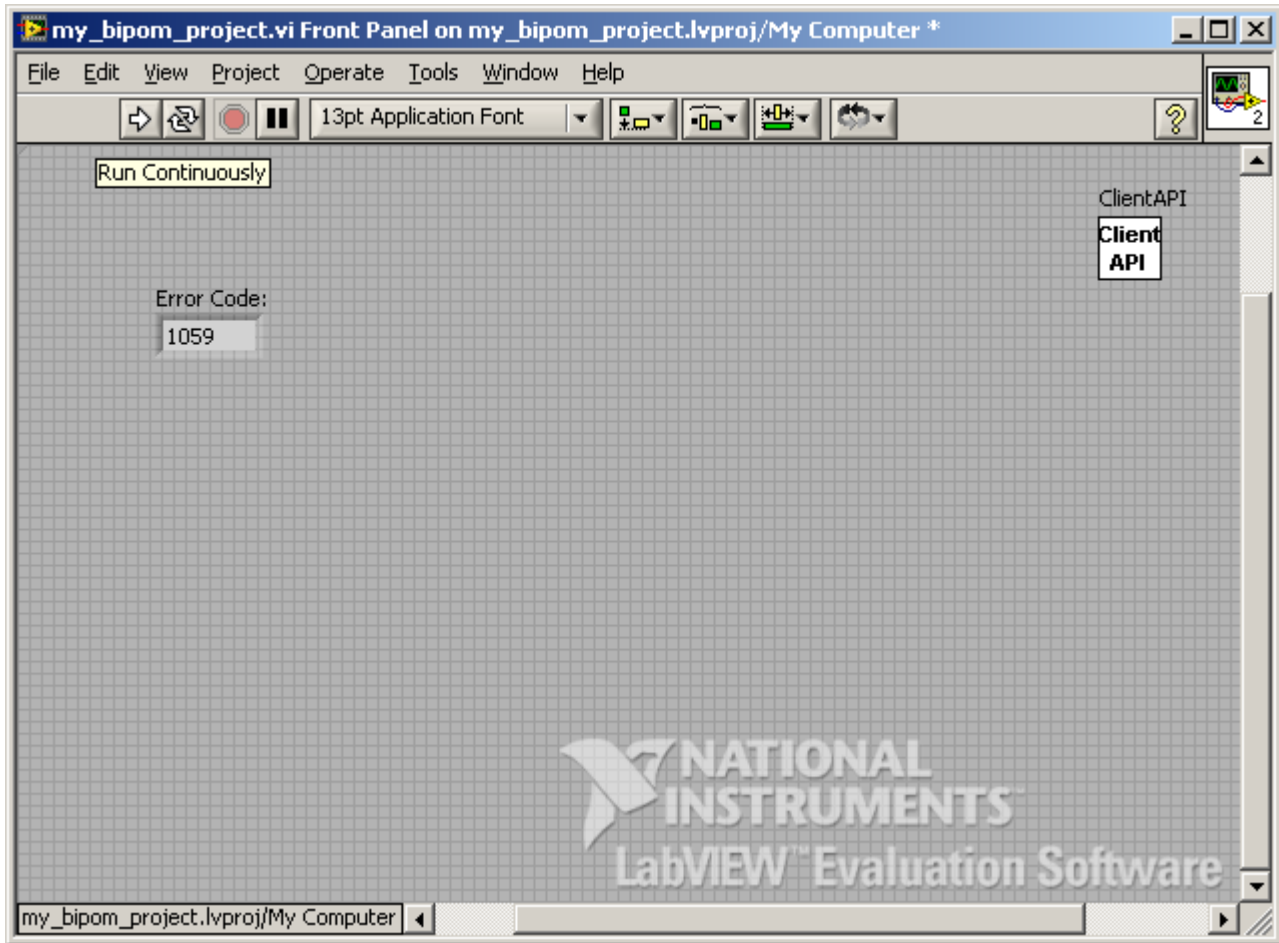
Now switch to the Front Panel window and add a **Numeric Indicator** control. Name the label of this control as **Error Code**:



In Block Diagram, connect the **Numeric Indicator** to the **ErrorCode** field:

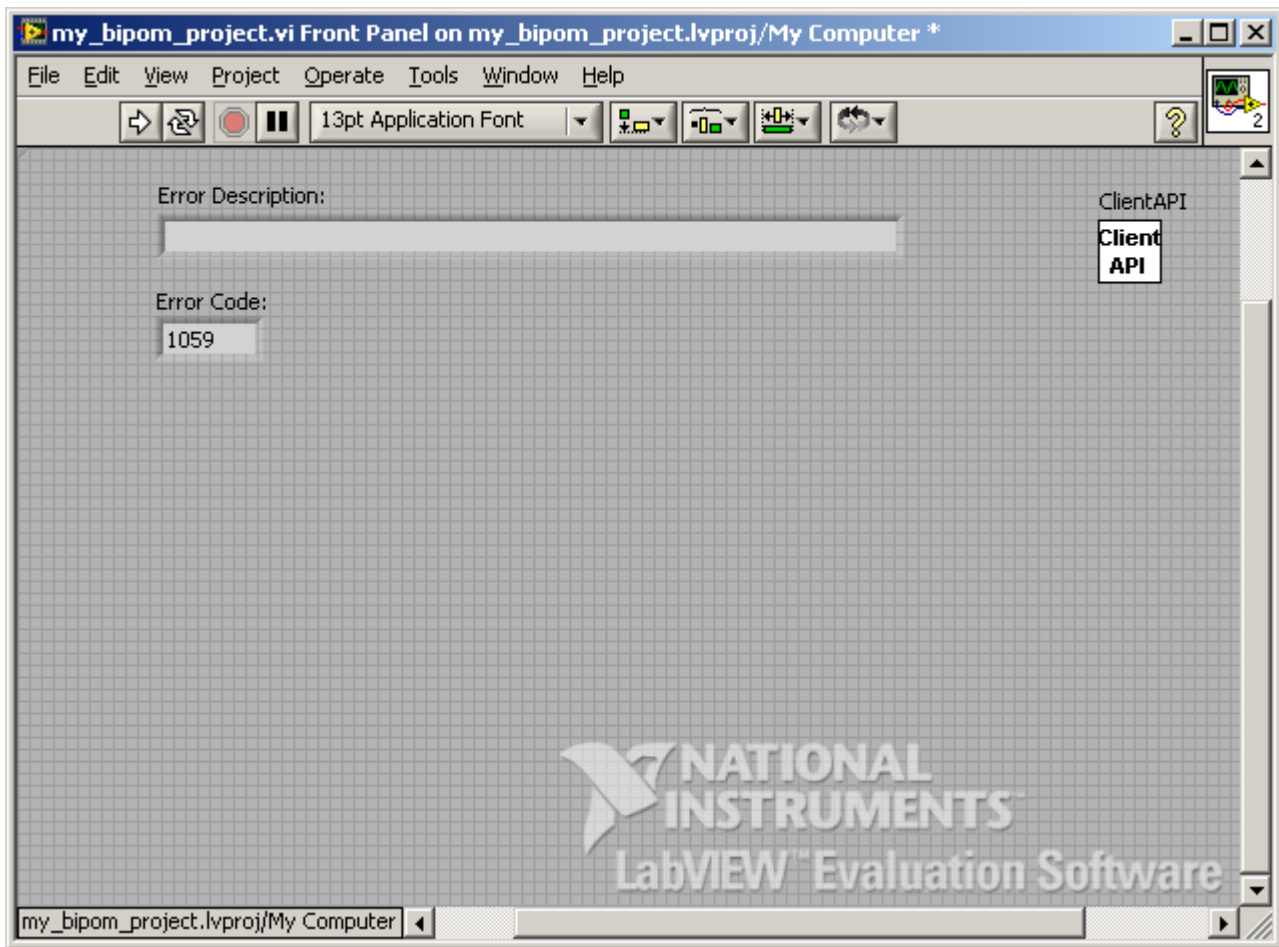


This means that the results of **ErrorCode** field will be displayed on **Error Code: Numeric Indicator**. To test our progress so far, run the program by clicking the **Run Continuously** button in Front Panel:

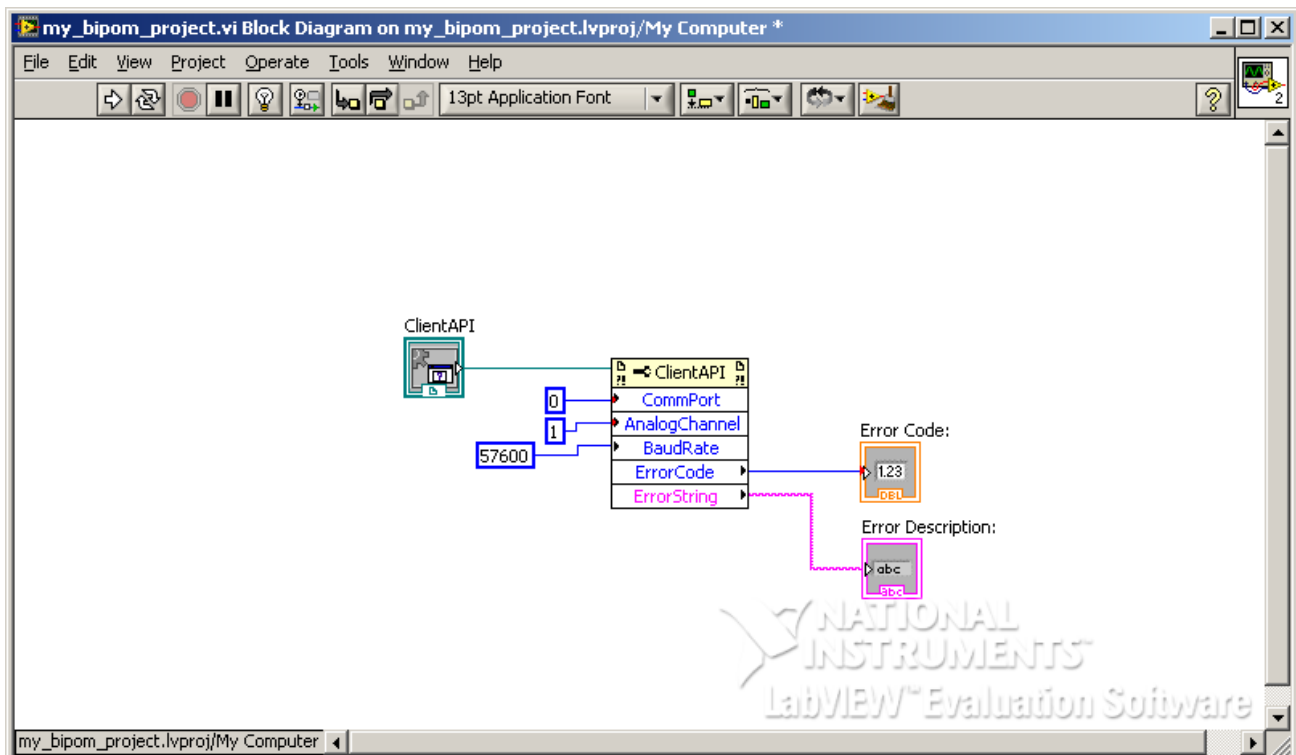


If the board is not connected or powered up, you will see an error code of 1059. To translate this error to human readable description, let's add an error string field to the Front Panel.

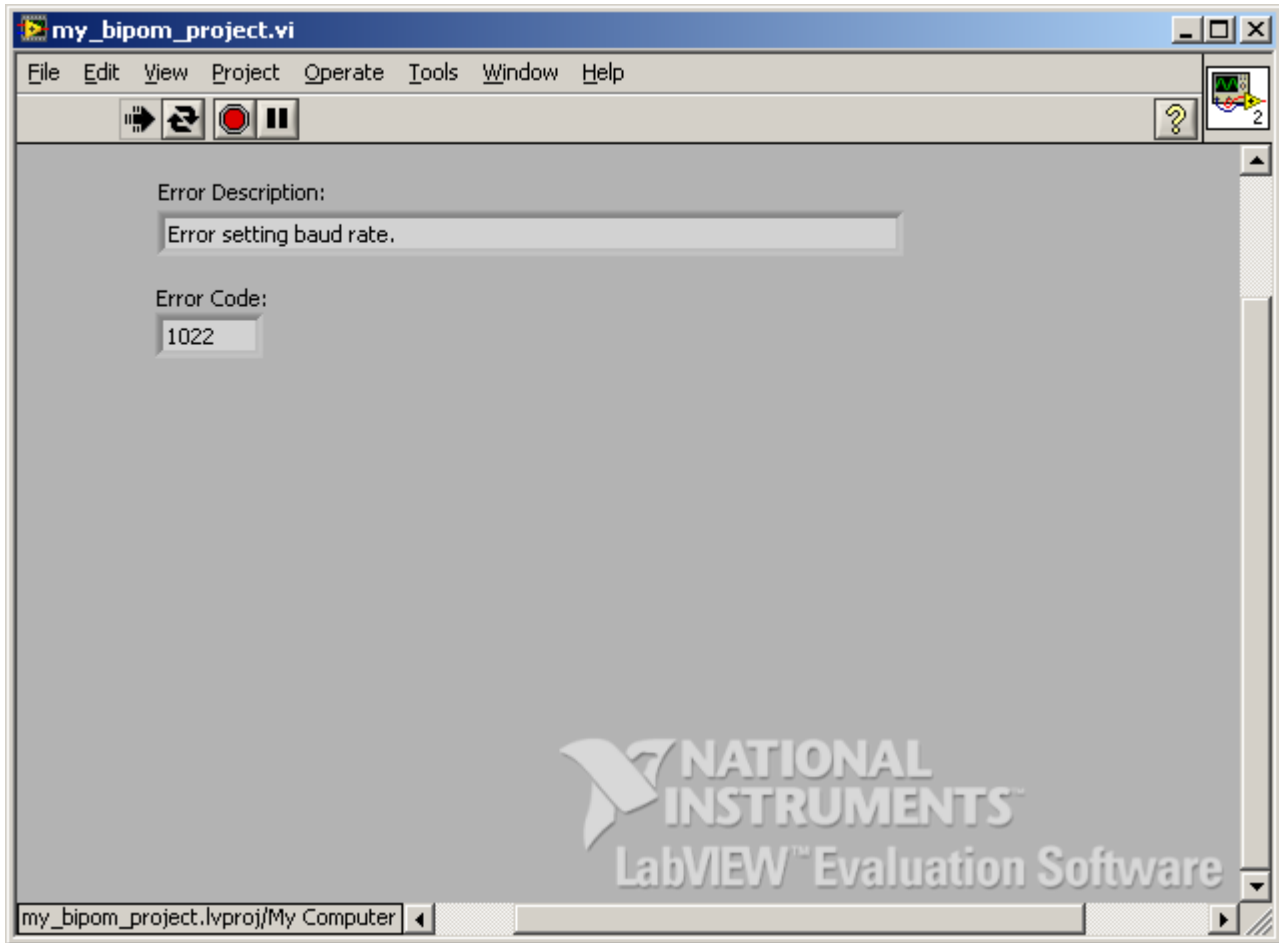
Right click on Front Panel area and select **String Indicator** under **Text Inds** group. Place the **String Indicator** on the Front Panel, size as needed and rename as **Error Description:**



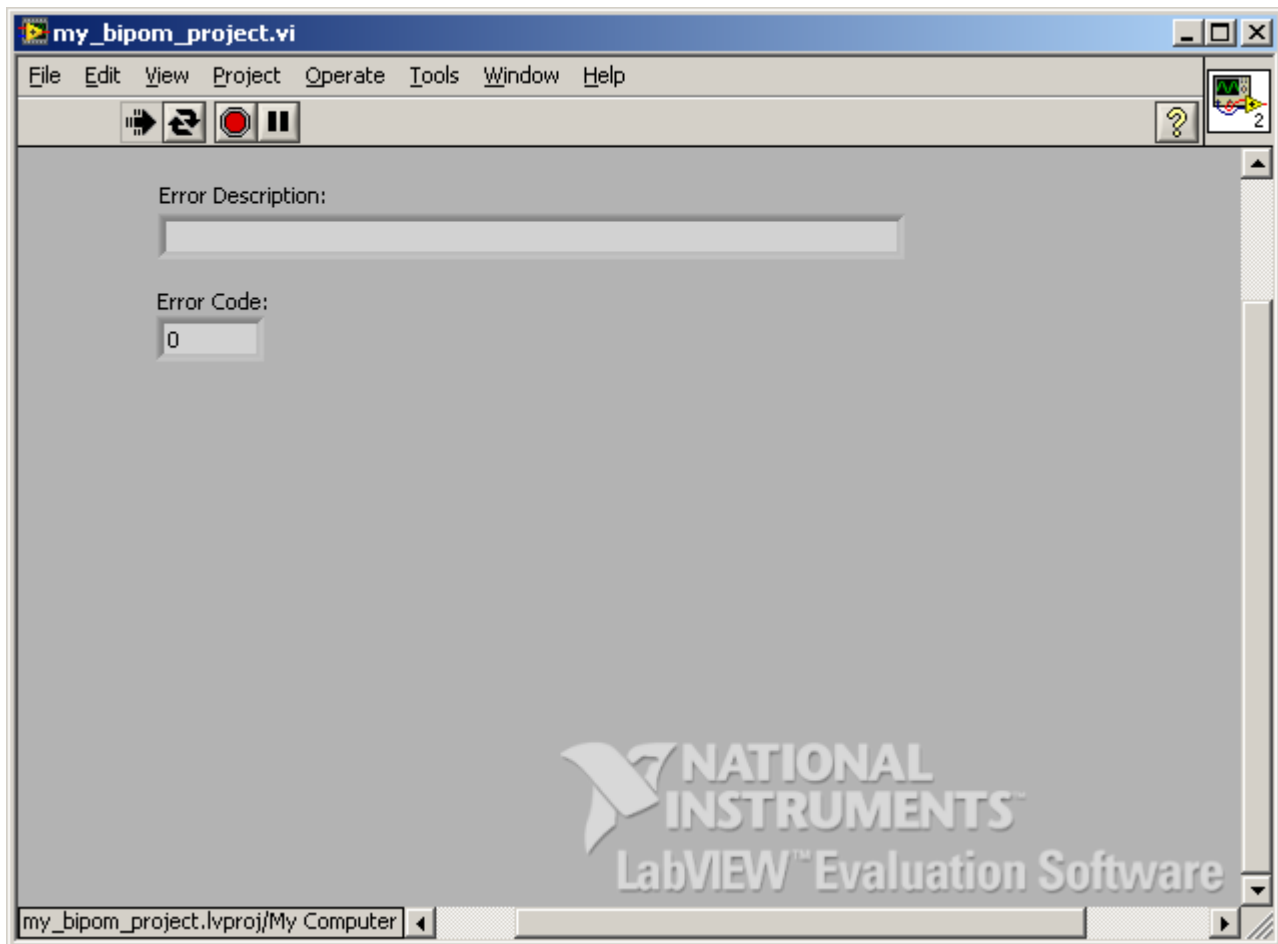
Now, go back to Block Diagram window and connect the **ErrorDescription** field to **Error Description** indicator:



Run the program again and you will see both the error code and error description displayed:

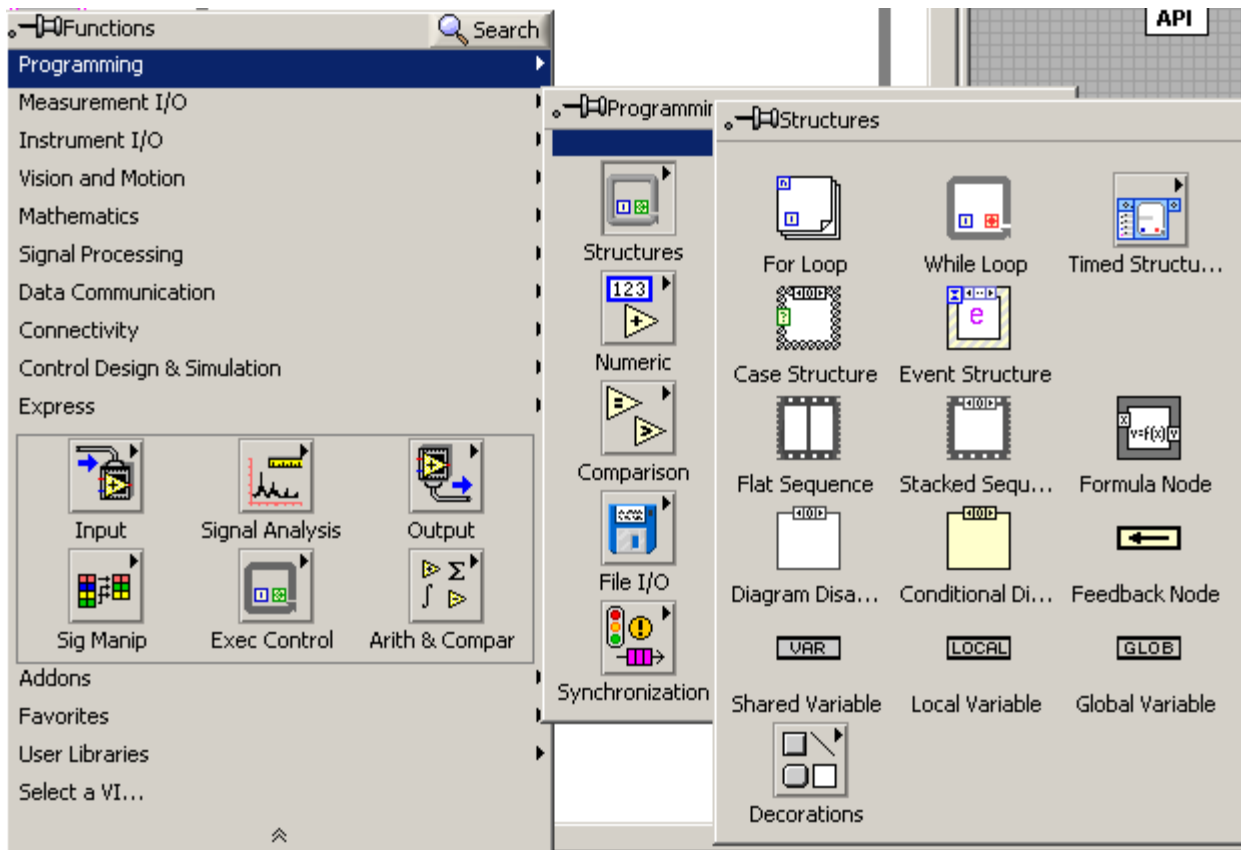


At this point, we can connect a real board to the PC and the error should disappear. Connect the board to COM1 of the PC and power the board. Run the LabVIEW™ program. If everything is working well, the error should disappear. Note the Error Code of 0 which means SUCCESS:

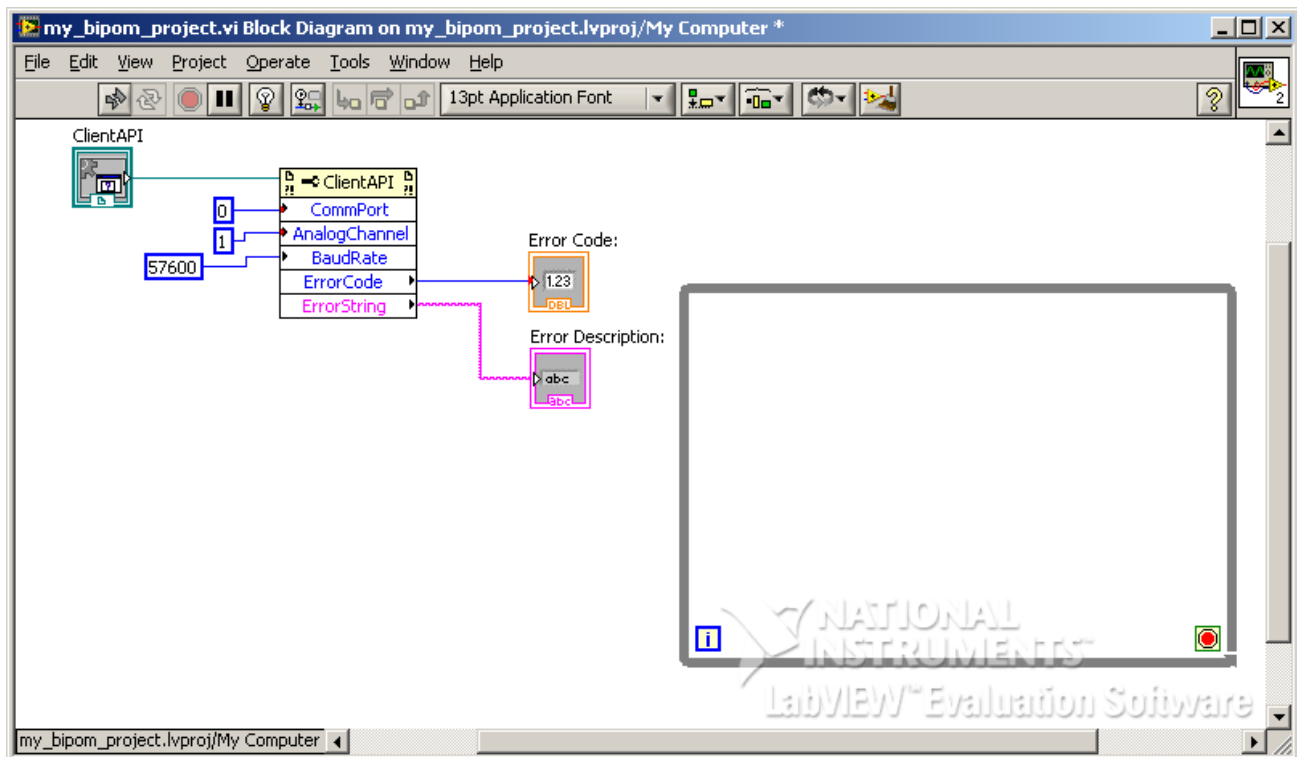


Stop the program. As next step, we will add a section to the Block Diagram to acquire data from the board periodically. For this purpose, we will use a **While Loop** that runs continuously. In our example, the loop will run every 10 milliseconds. In other words, we will collect one data point from the board every 10 milliseconds.

Add a **While Loop** block to the Block Diagram. **While Loop** is under **Programming->Structures** window of Functions menu that you start from the Block Diagram with a right click:

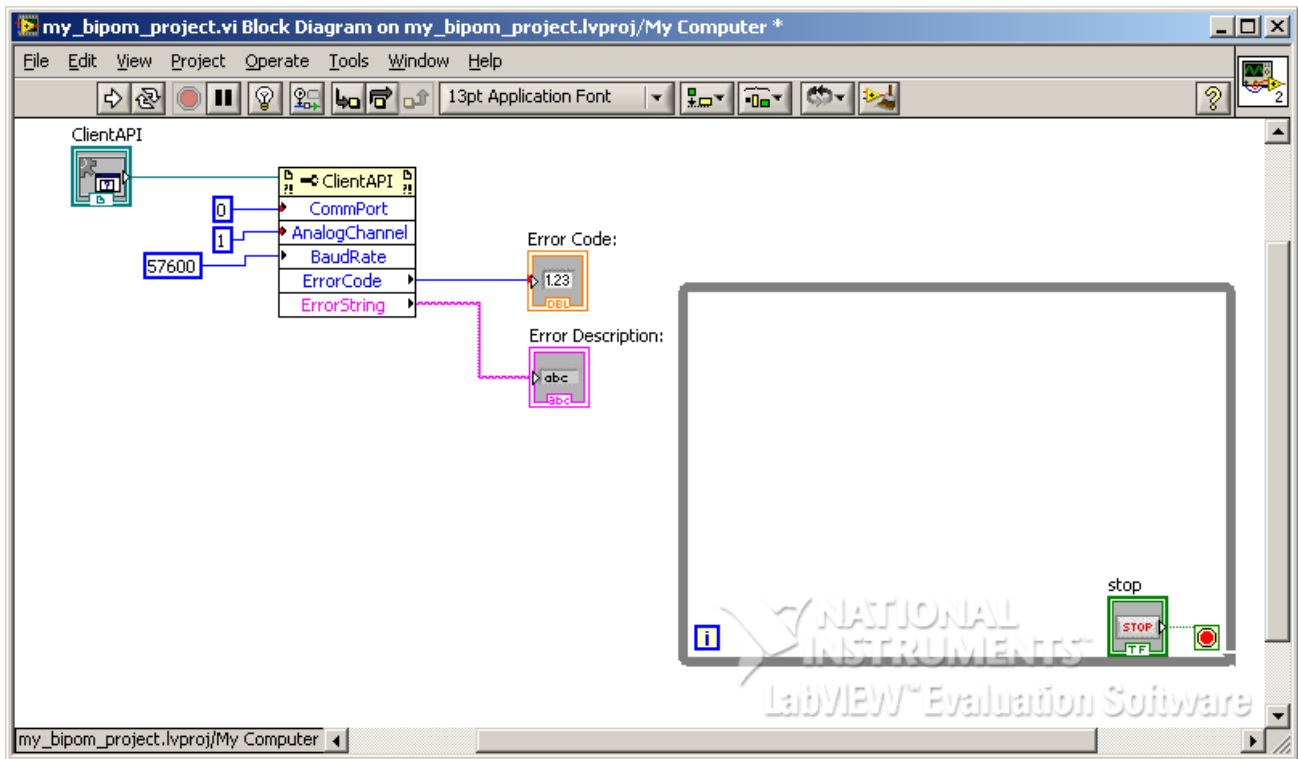


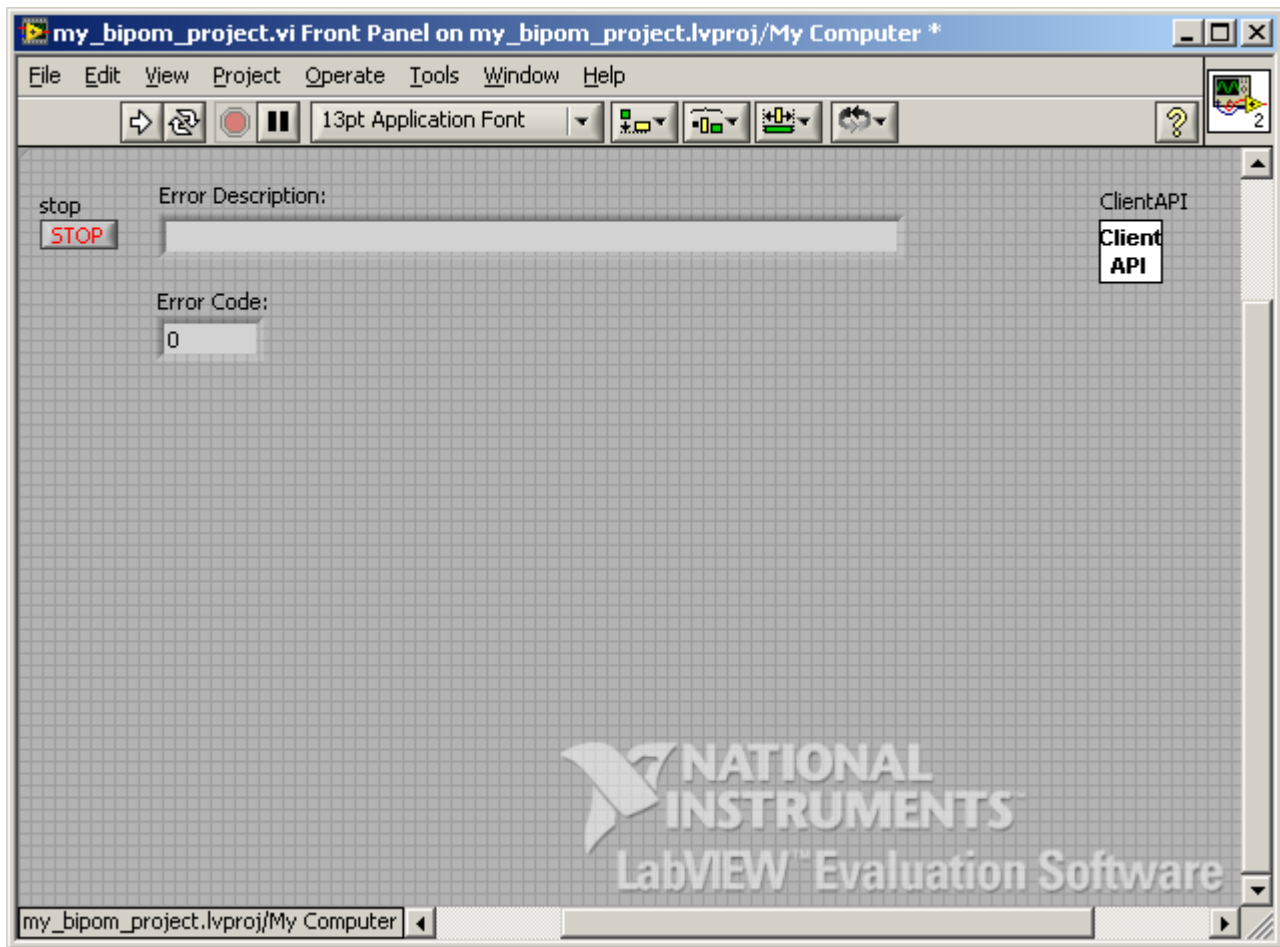
Drag and drop the **While Loop** into the Block Diagram and resize it bigger to allow other blocks to be inserted into the **While Loop**:



You will notice the small **i** logo at the bottom left. This is the loop iteration counter. The small **Stop** sign at the bottom right is the stop condition of the loop.

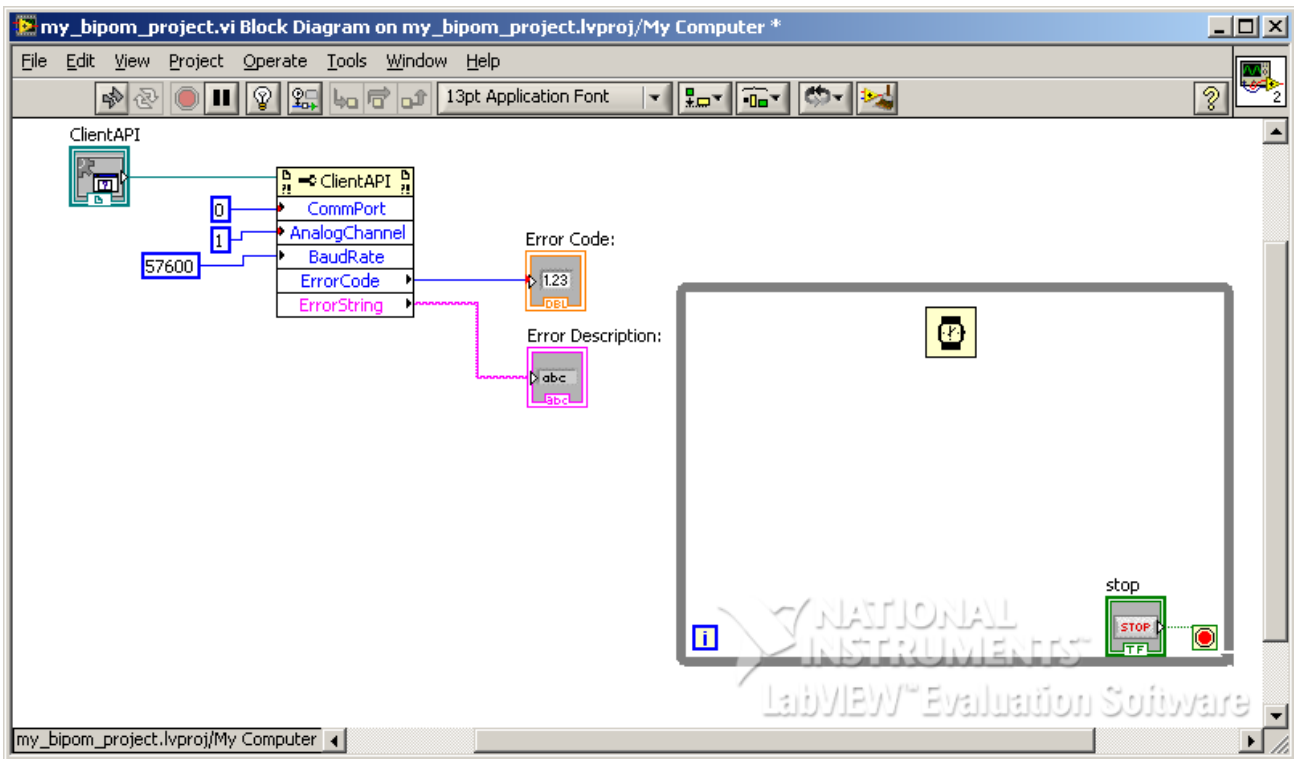
Right click on **Stop** sign and select **Create Control**. A **Stop Control** will be added to Block Diagram and a **STOP** button will be added to Front Panel:



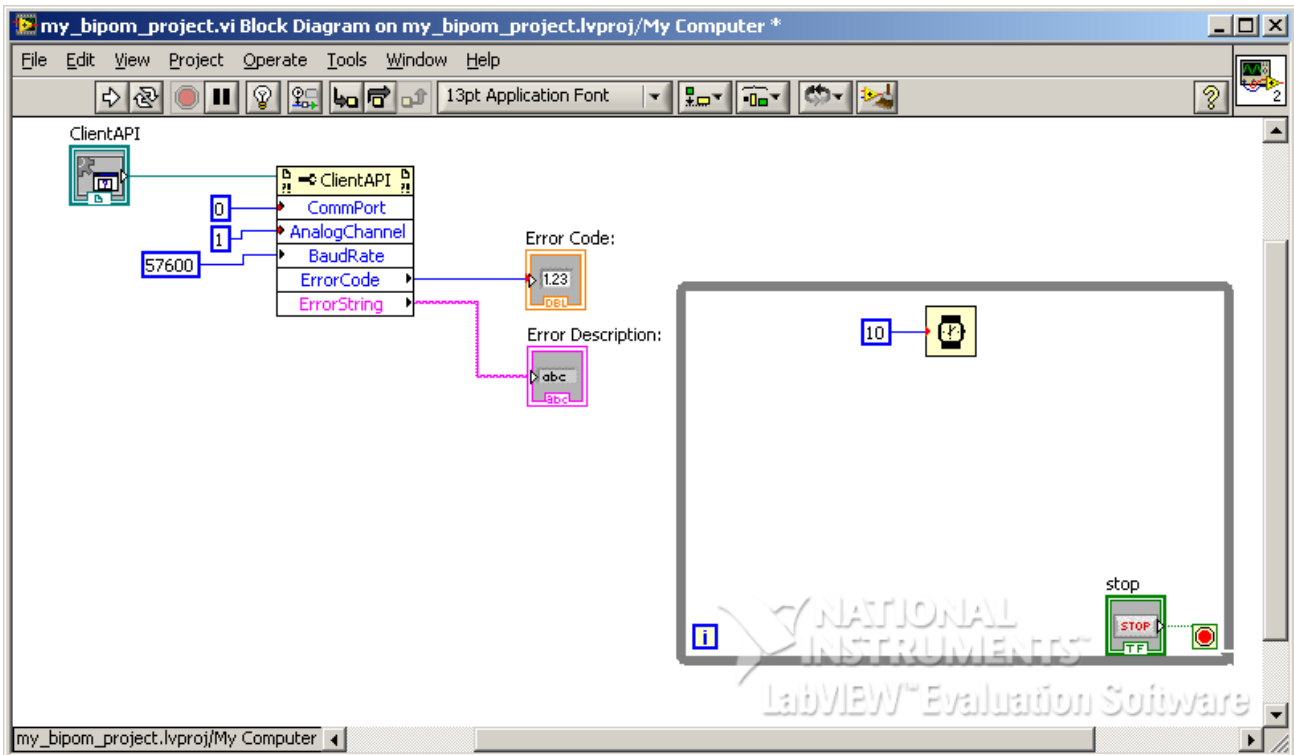


This **STOP** button can be used to stop loop operation anytime.

Add a **Wait** function to the **While Loop**. **Wait** function is under **Programming->Timing** window of Functions menu:

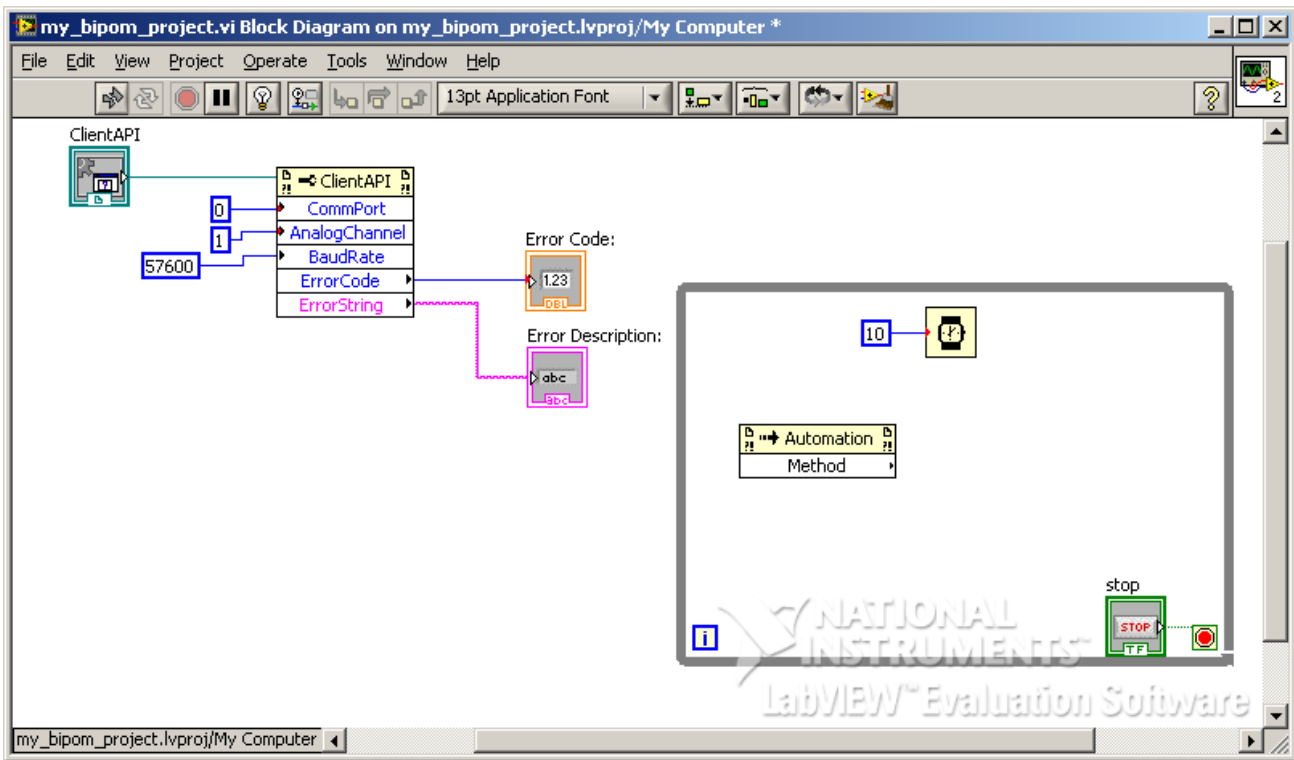


This will control how often the loop will operate. Assign a value of 10 to the **Wait** function:

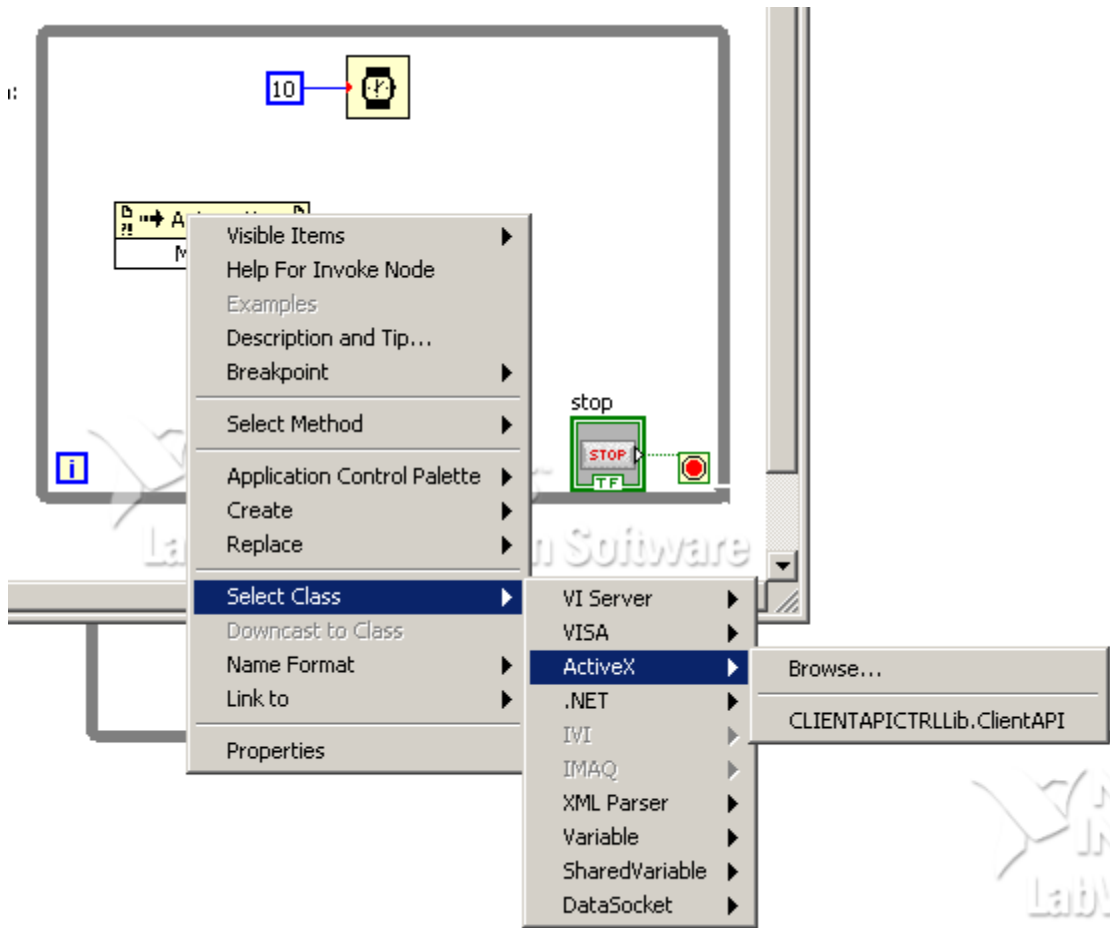


This means that the loop will execute every 10 milliseconds.

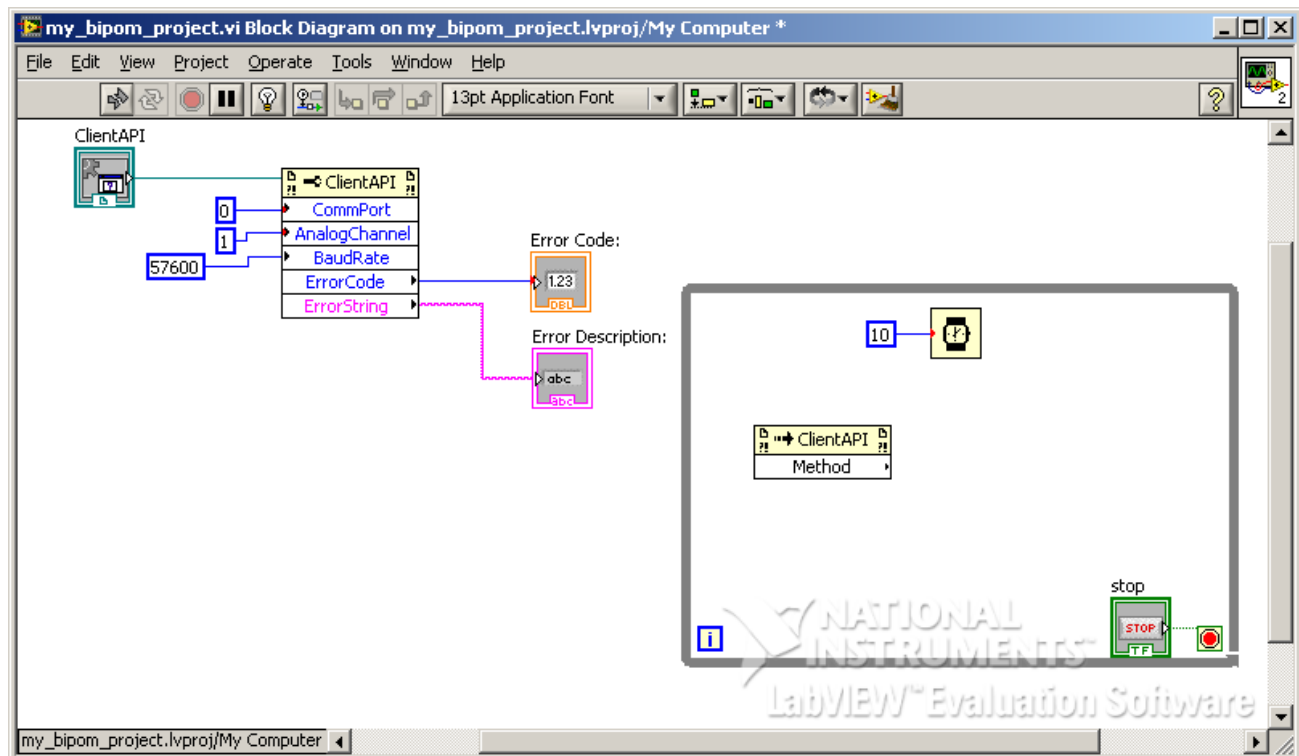
Now that we have a loop that runs every 10 milliseconds, we will add an **Invoke Node** to the **While Loop** to invoke the ClientAPI **ReadAnalog** function. This is the function that communicates with the board and reads one analog value. Right click in Block Diagram to bring the Functions menu and select **Connectivity->ActiveX->Invoke Node** and drop inside the **While Loop** area:



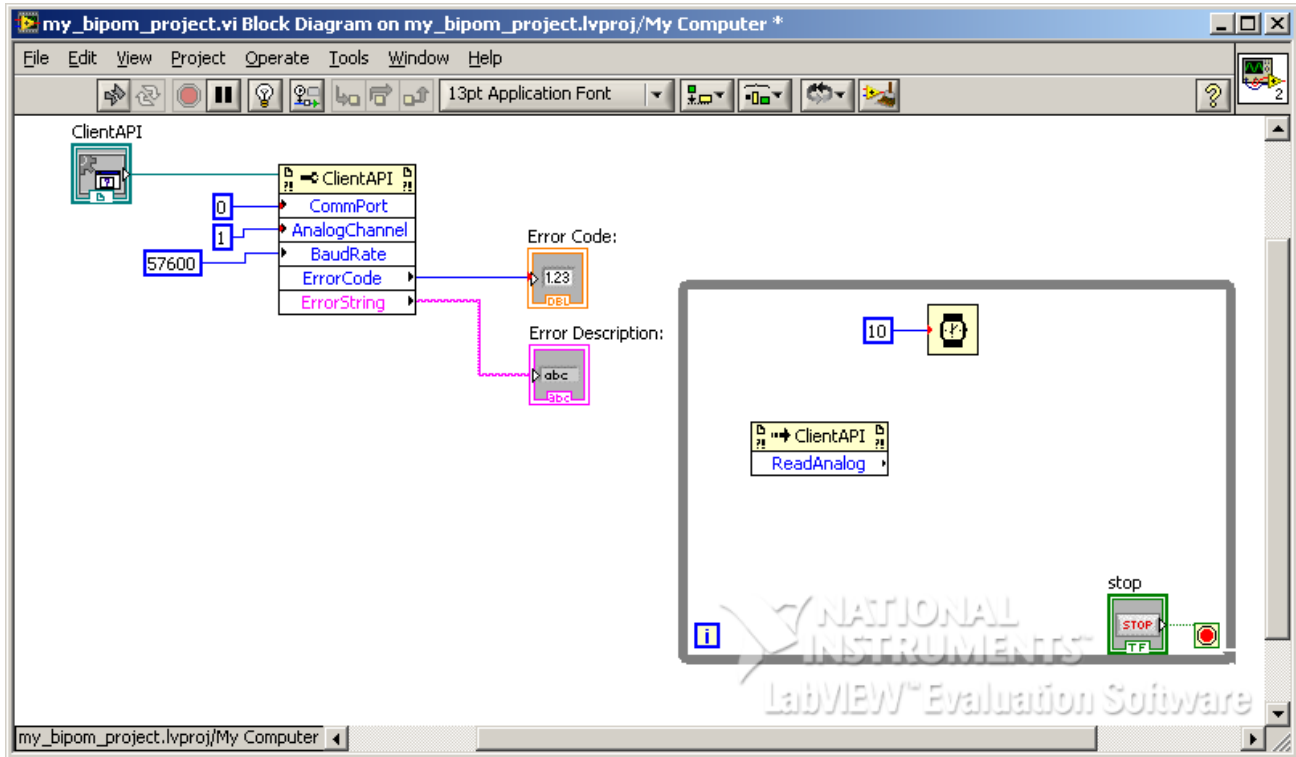
Right click on **Invoke Node** and select **Select Class->ActiveX->CLIENTAPICTRLlib.ClientAPI:**



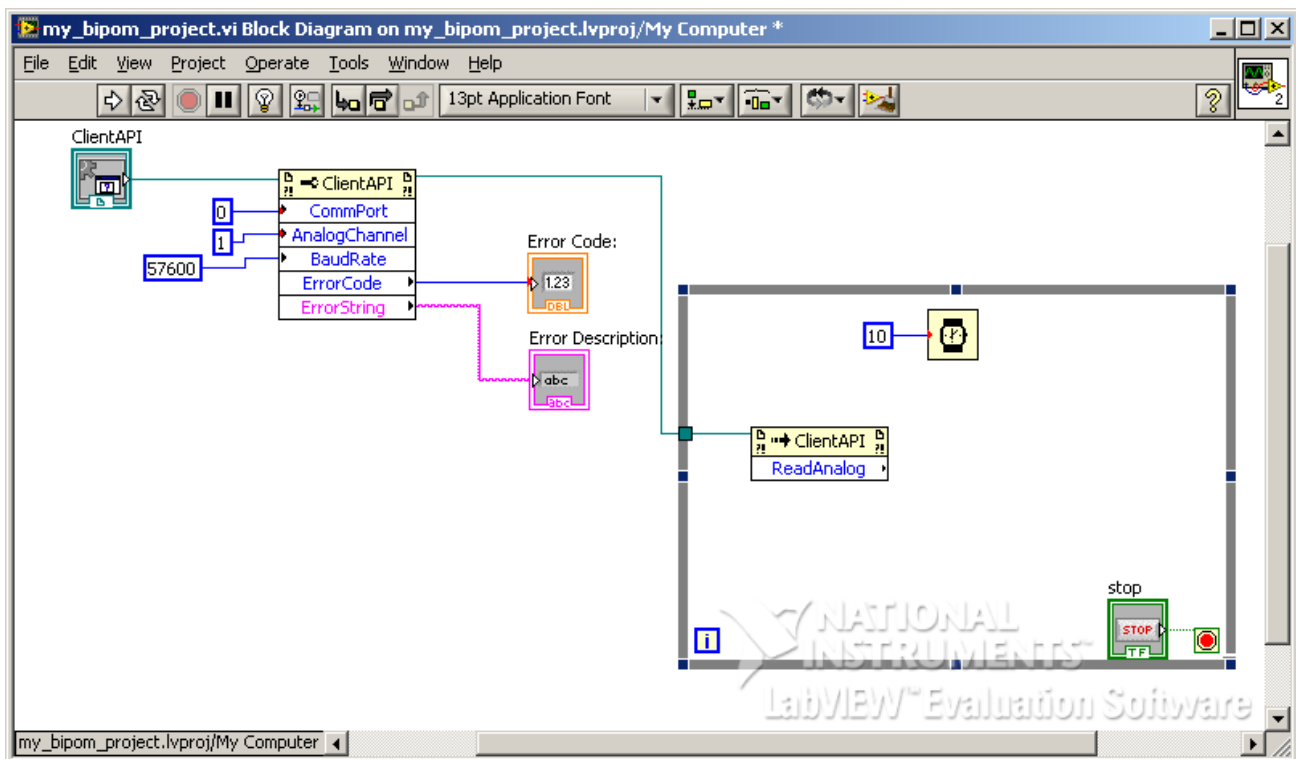
Note that the name of *Invoke Node* changes to ClientAPI:



Right click on **Method** field. From the menu, choose **Select Method->ReadAnalog**. The name of the **Method** field will change to **ReadAnalog**.



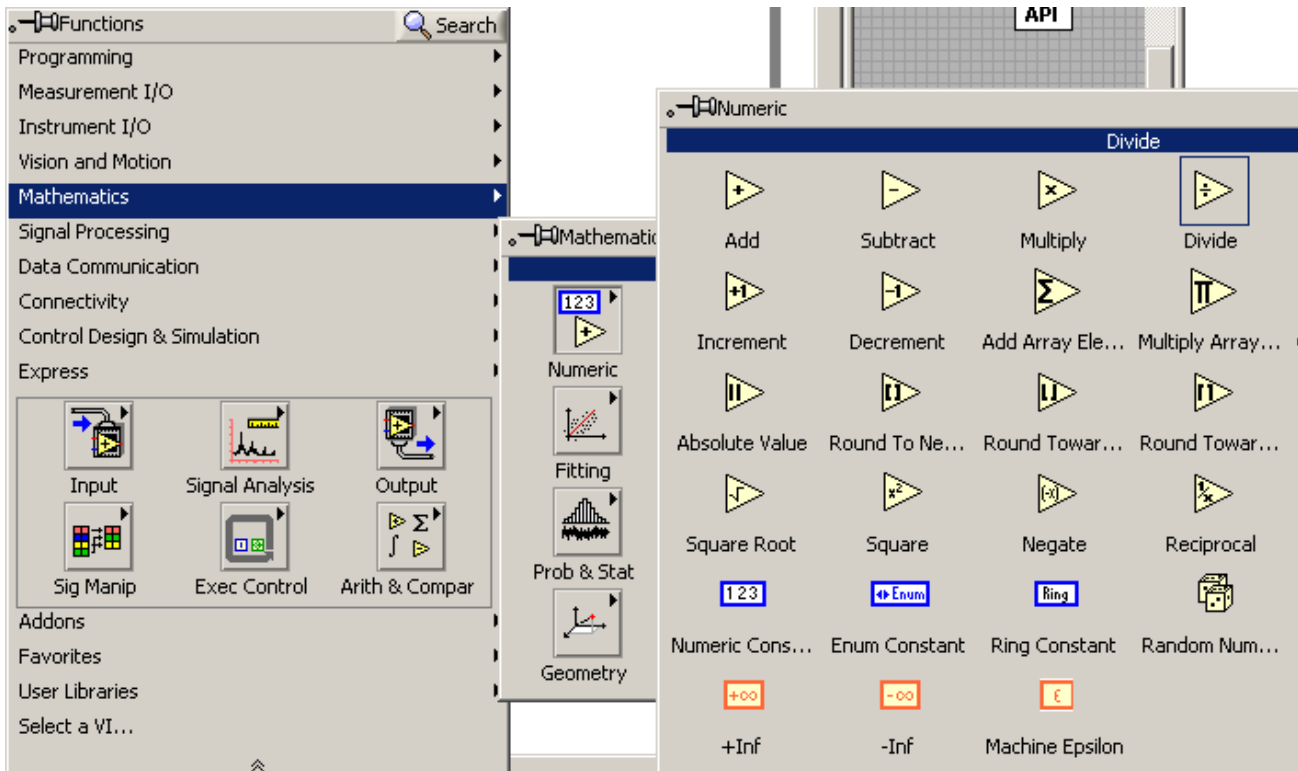
Using a wire, connect the **Reference** input of **Invoke Node** to **Reference Output** of **Property Node**:



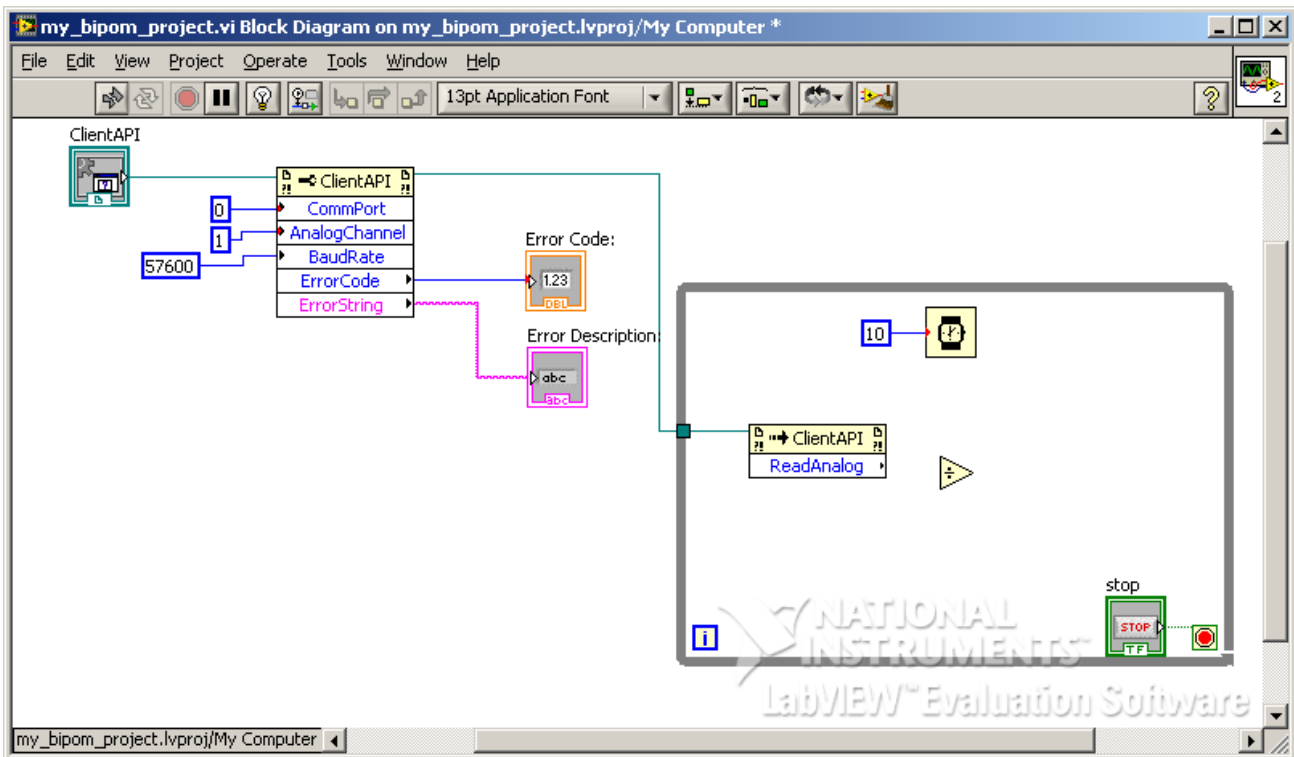
ReadAnalog function returns a raw value from 0 to 4095 corresponding to a voltage of 0 to 4.095 Volts. So to display real voltages, we use the following formula:

$$\text{Voltage} = \text{Raw Value} / 1000$$

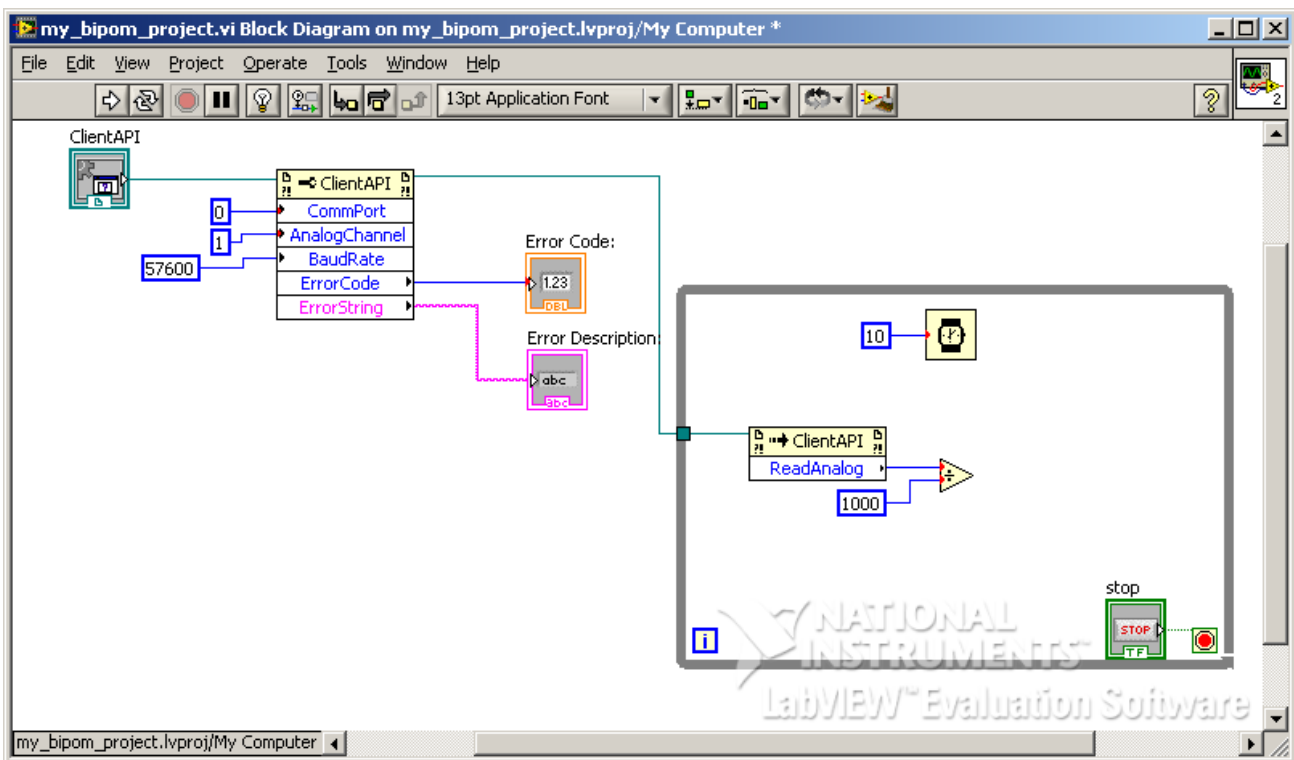
To implement this, we add a divider to the LabVIEW™ program. Right click on Block Diagram and bring up the Functions menu. The **Divide** function is under **Mathematics->Numeric**:



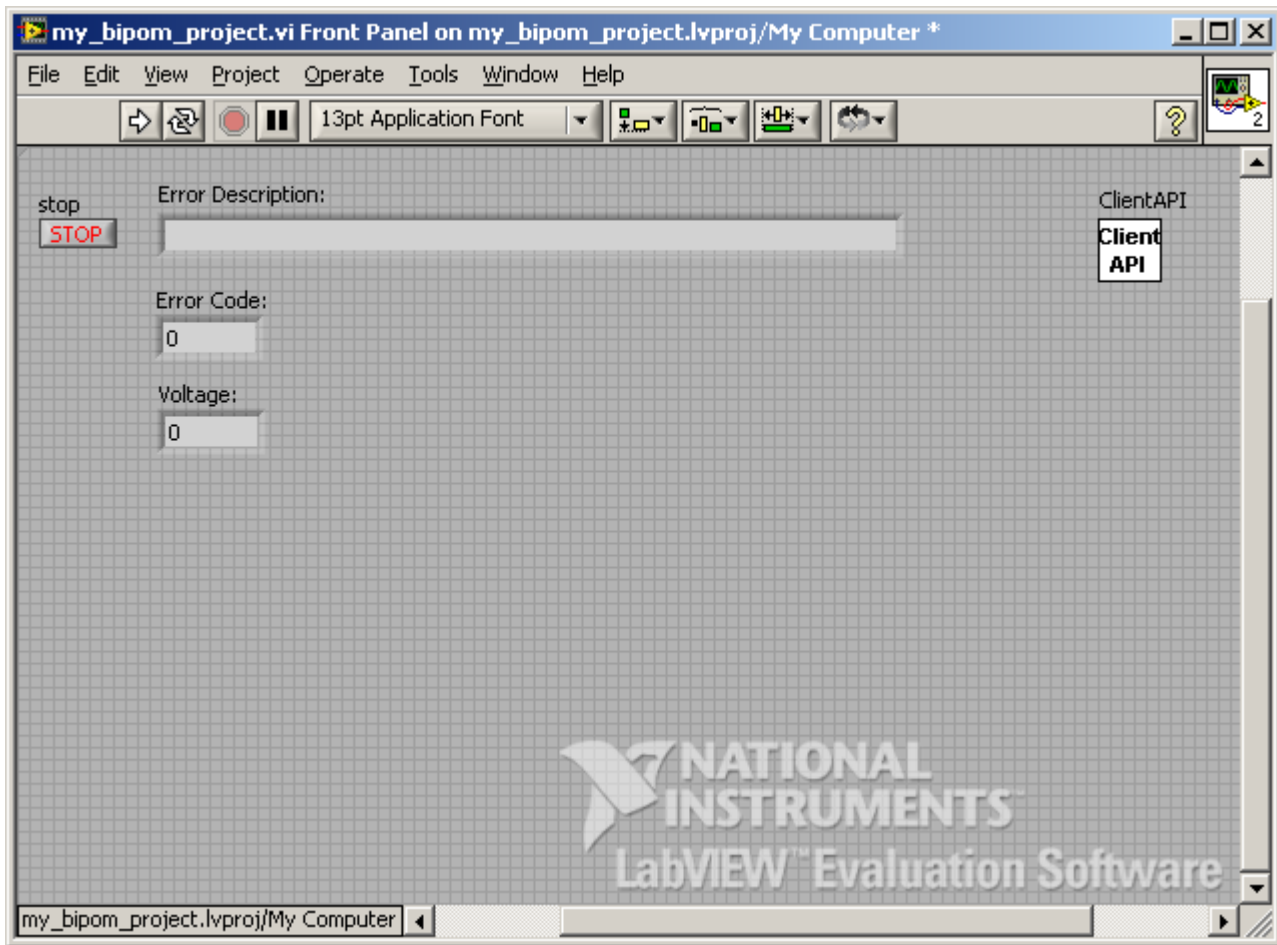
Drop the **Divide** function in the **While Loop** area:



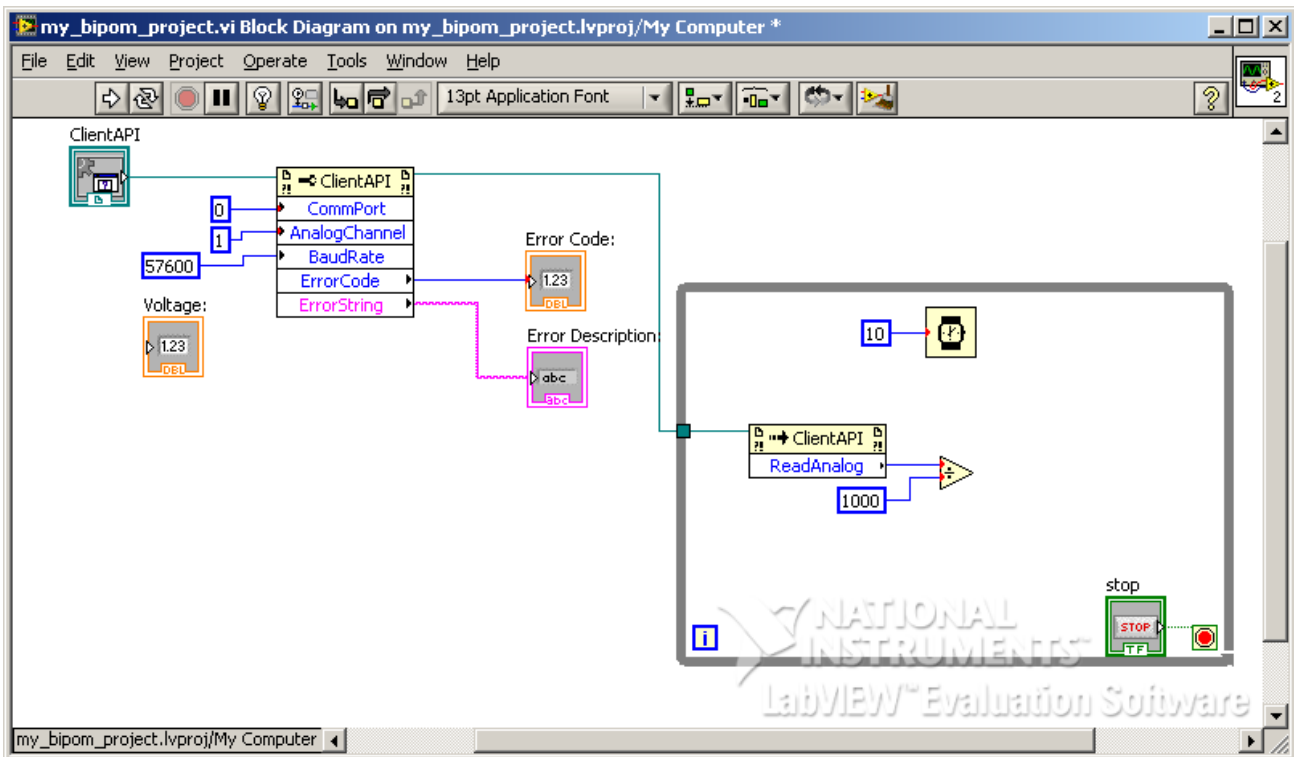
Connect the **ReadAnalog** output of **ReadAnalog** function to **X input** of **Divider** using a wire. Connect a **Numeric Constant** of 1000 to **Y input** of **Divider**. This will implement $\text{ReadAnalog}/1000$ operation:



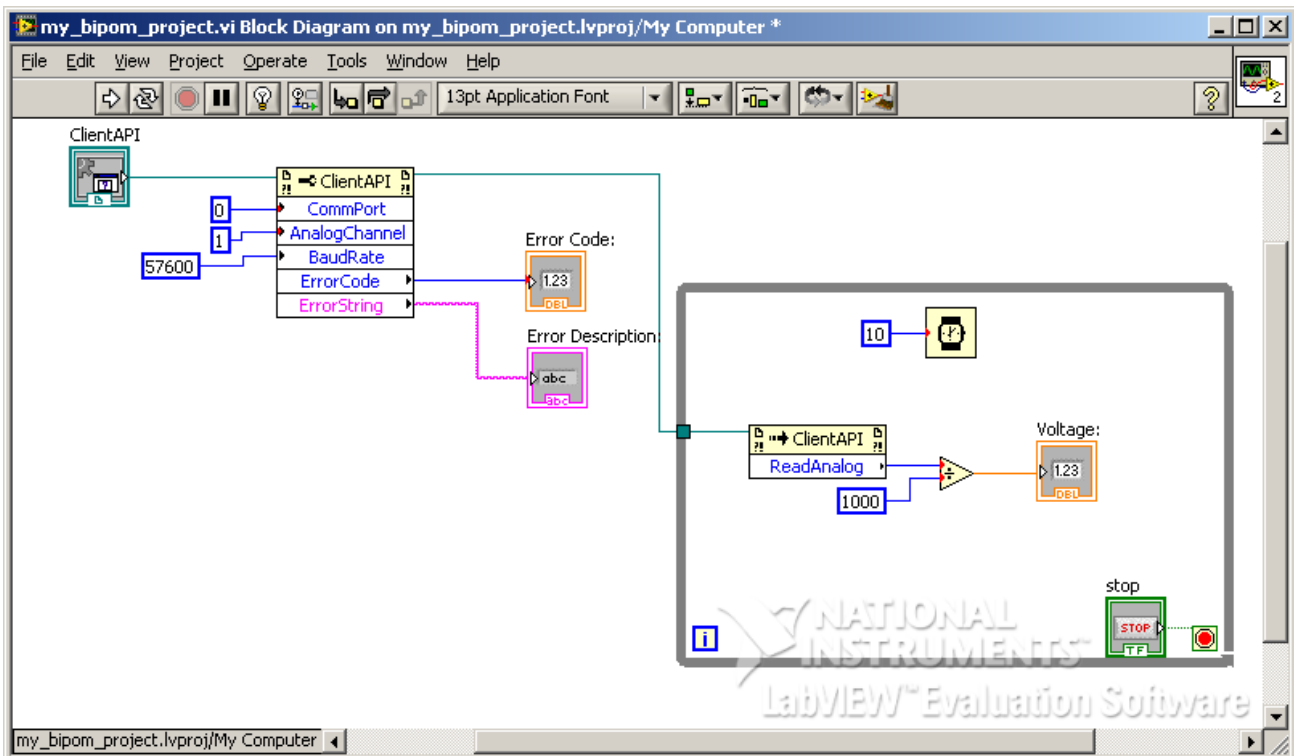
Now comes the exciting part. We are about to display our voltage value on the Front Panel. Add a **Numeric Indicator** to Front Panel by selecting **Num Ind** from under **Num Inds** in Controls menu. Rename the **Numeric Indicator** as Voltage:



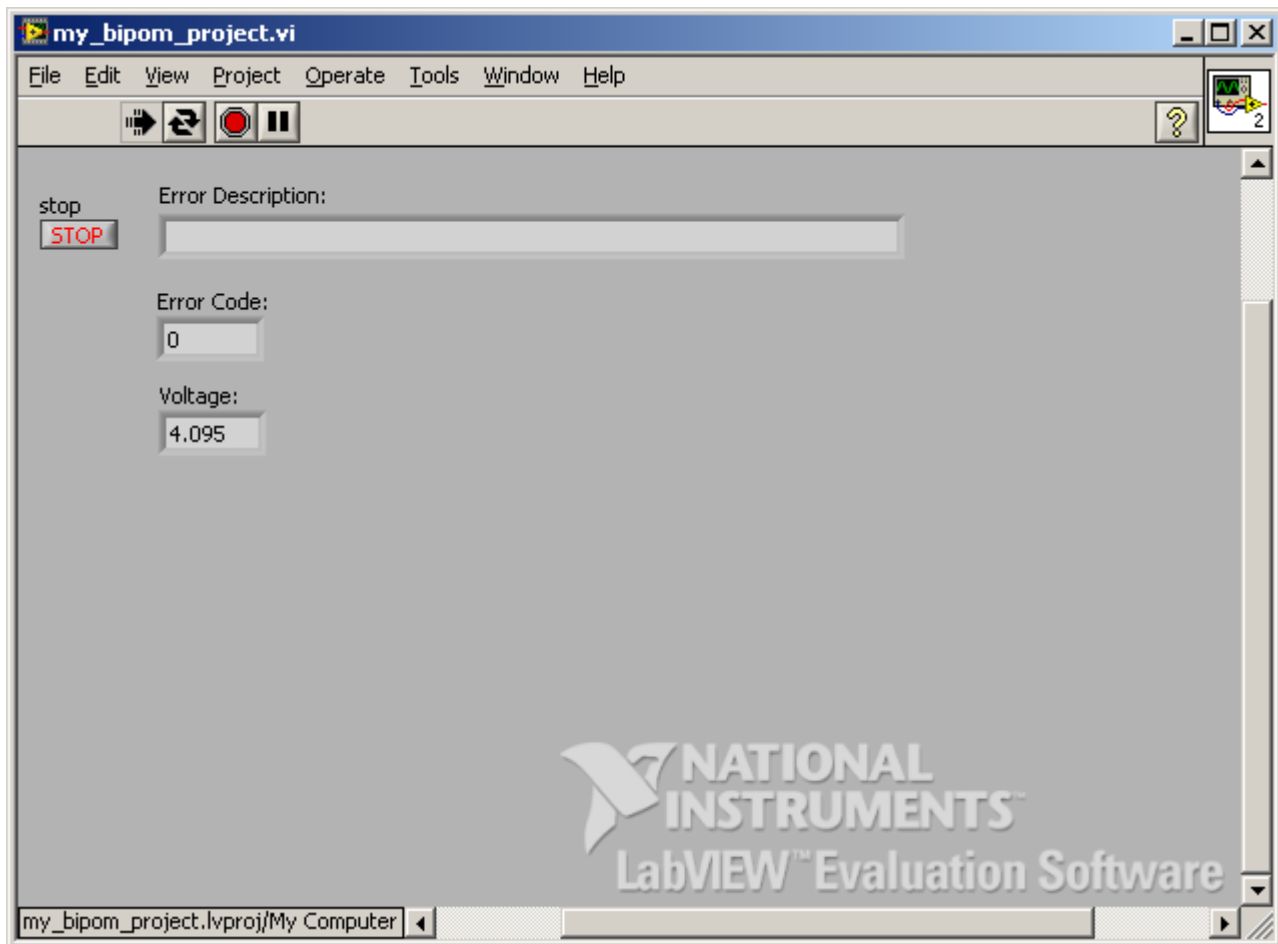
Notice that the **Numeric Indicator** was also added to Block Diagram:



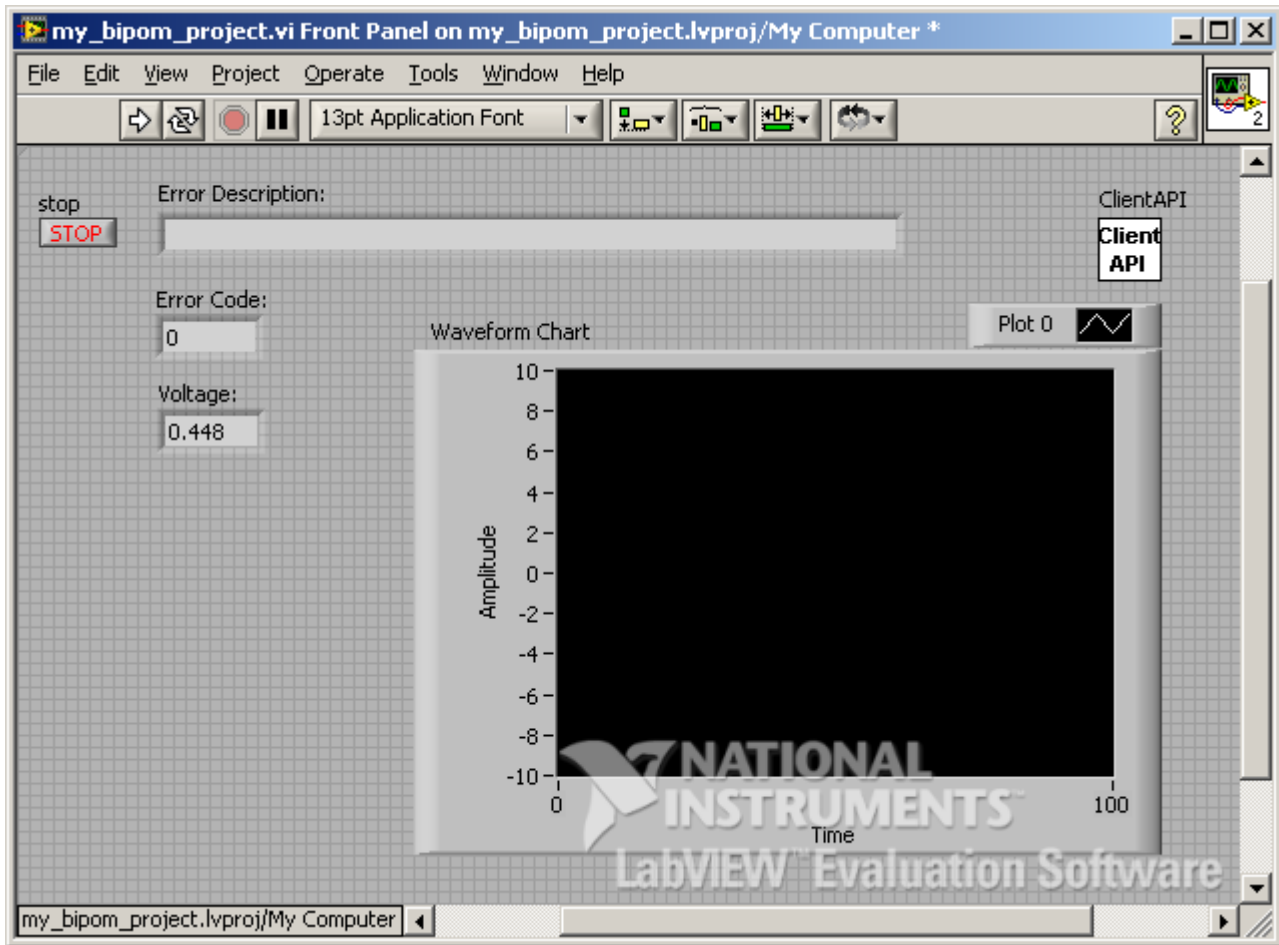
Move the **Numeric Indicator** to inside **While Loop** and connect its input to **X/Y output** of **Divider**.



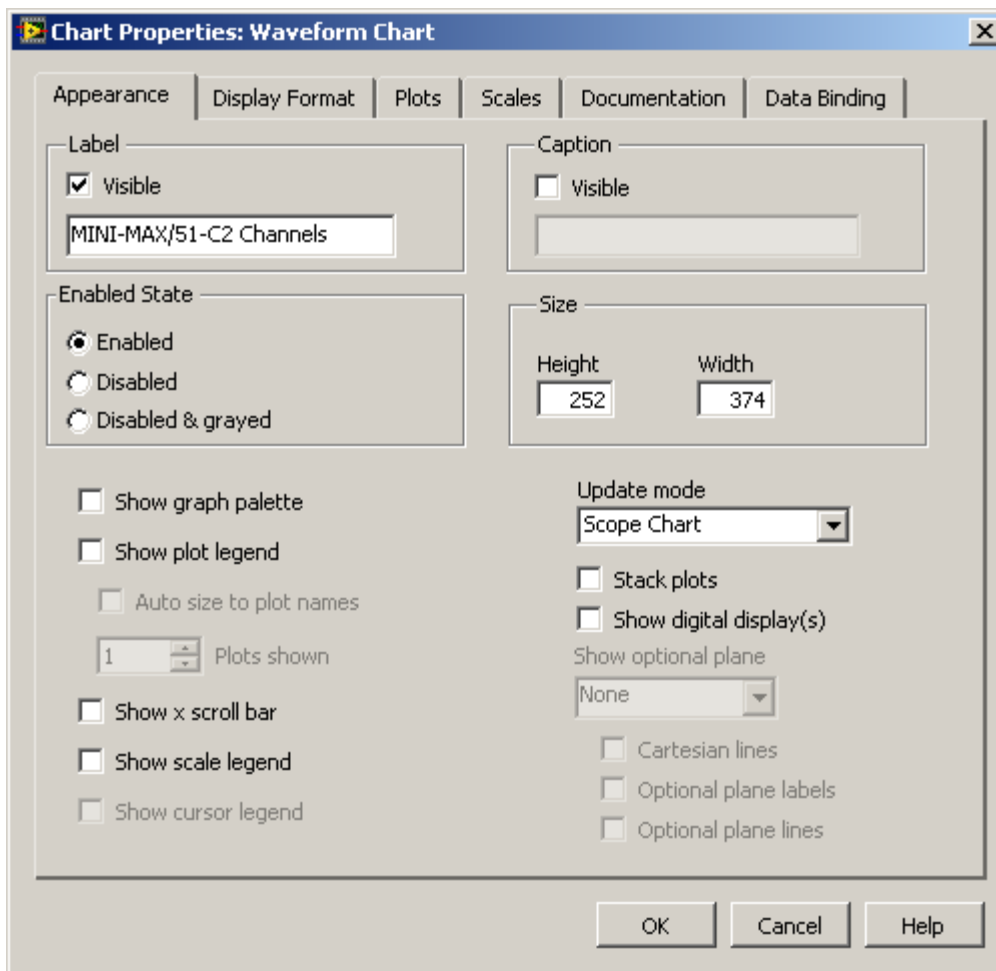
Now, run the program and you should see the voltage from the board inside the **Numeric Indicator** on the Front Panel:



As next step, we want to display the value from the board also on a graphical chart. With LabVIEW™, this is easy since there is a built-in **Chart** function. Right click on Front Panel and bring up the Controls menu. Select **Chart** from under **Graph Indicators**. Place the **Chart** on the Front Panel and resize as desired:

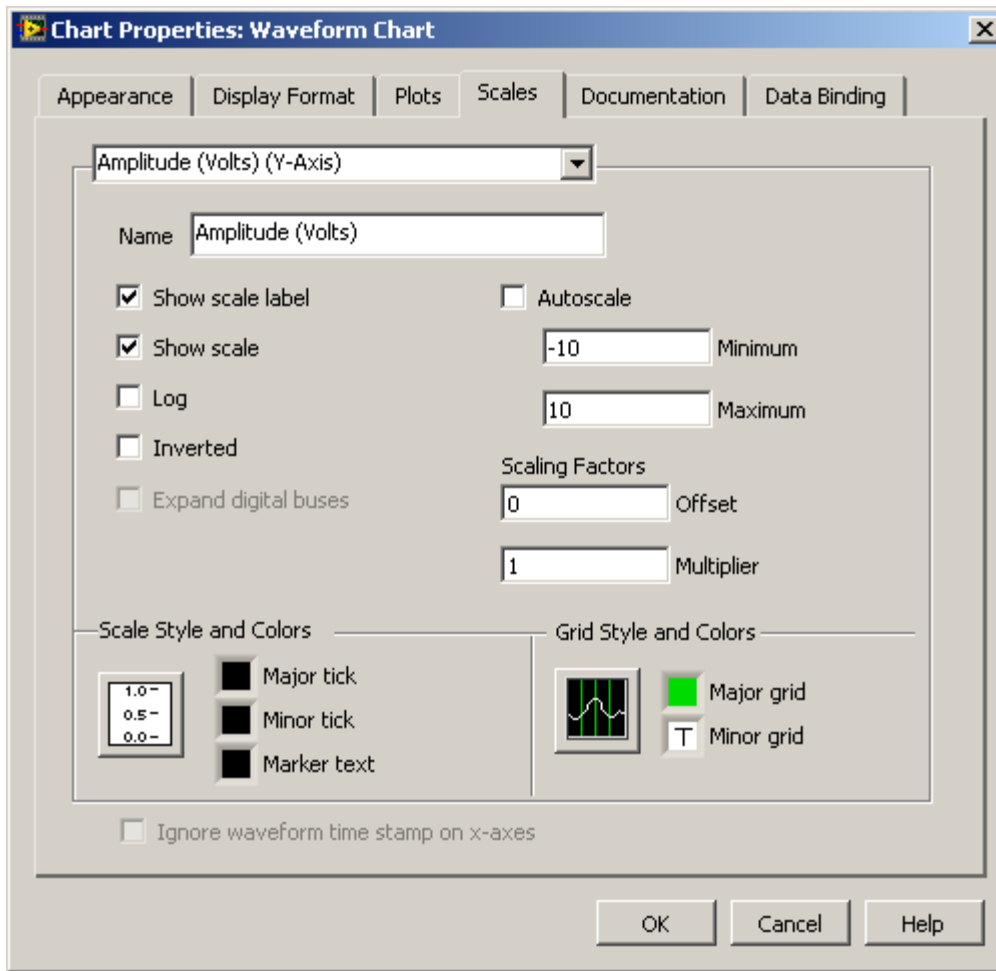


Right click on chart and select **Properties**. Rename chart as **MINI-MAX/51-C2 Channels**. Change **Update Mode** to **Scope Chart**. Uncheck **Show Plot Legend**.



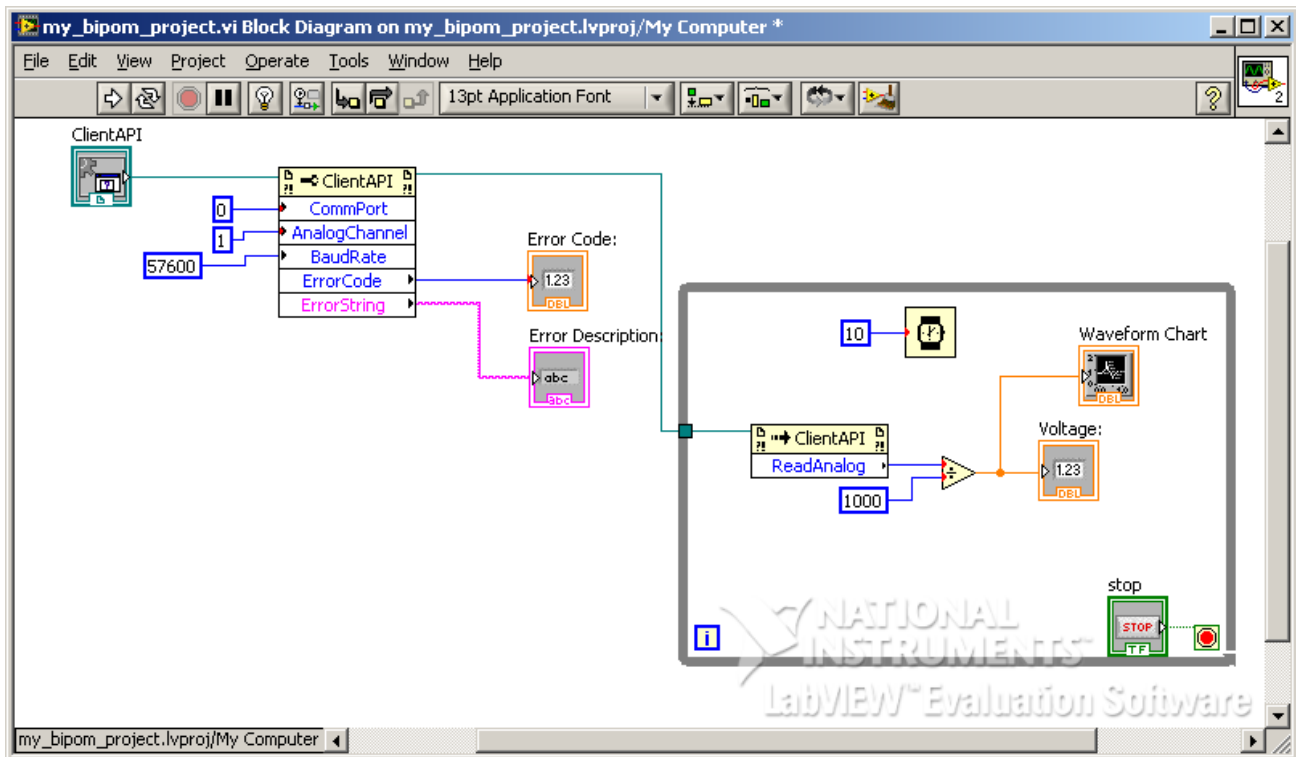
Click on **Scales** tab. Rename **X-Axis** as **Time (Milliseconds)** and **Y-Axis** as **Amplitude (Volts)**. Select green as the color for both X and Y axes.

In **Amplitude (Volts) Y-Axis**, uncheck **Autoscale** and enter **Minimum** as **0** and **Maximum** as **4.095**:

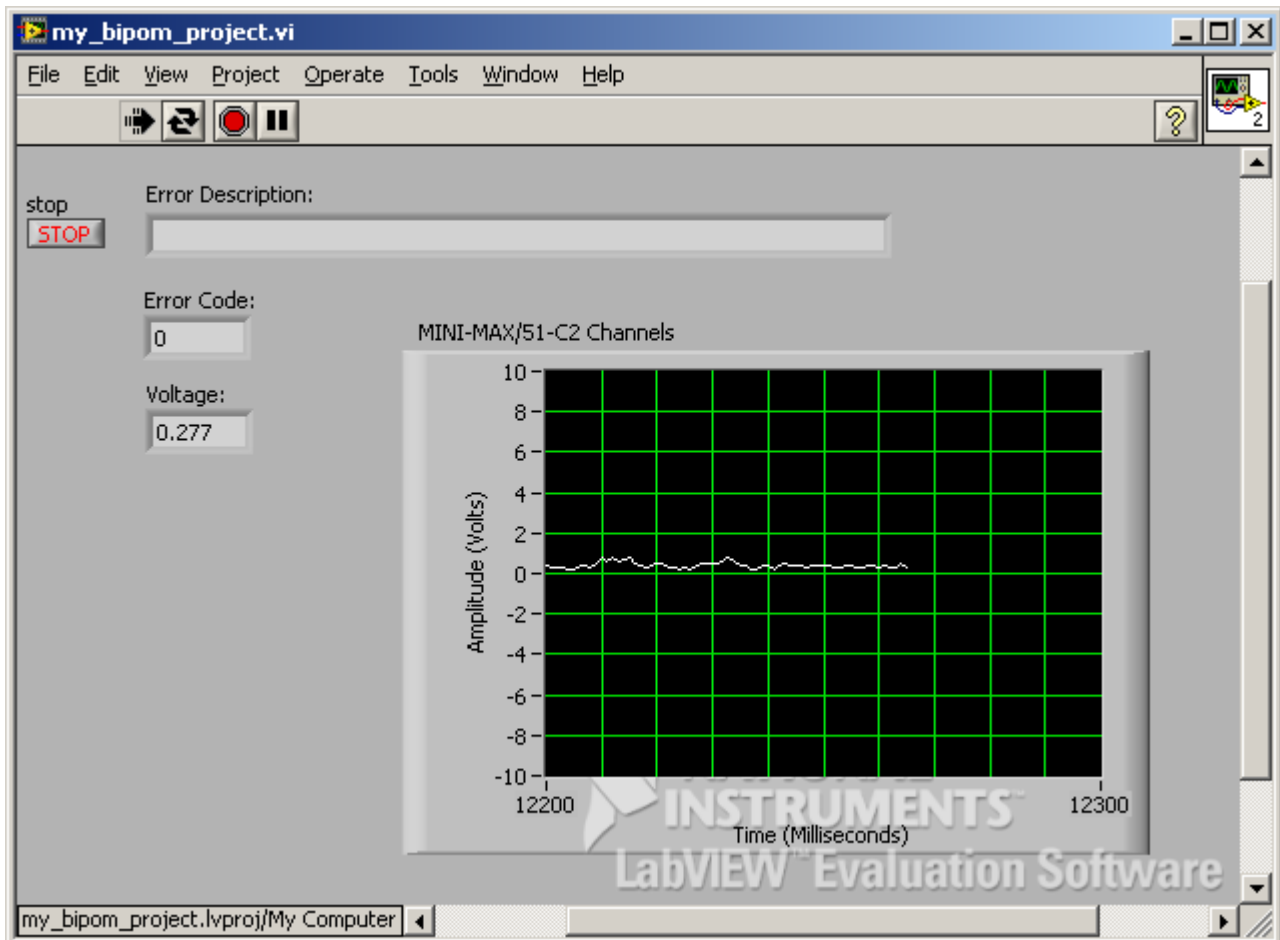


Click OK.

Notice that the **Chart** was also added to Block Diagram. Move the **Chart** inside the **While Loop** area and connect the input of the **Chart** to the output of the **Divider** (where we also connected the **Numeric Indicator**):

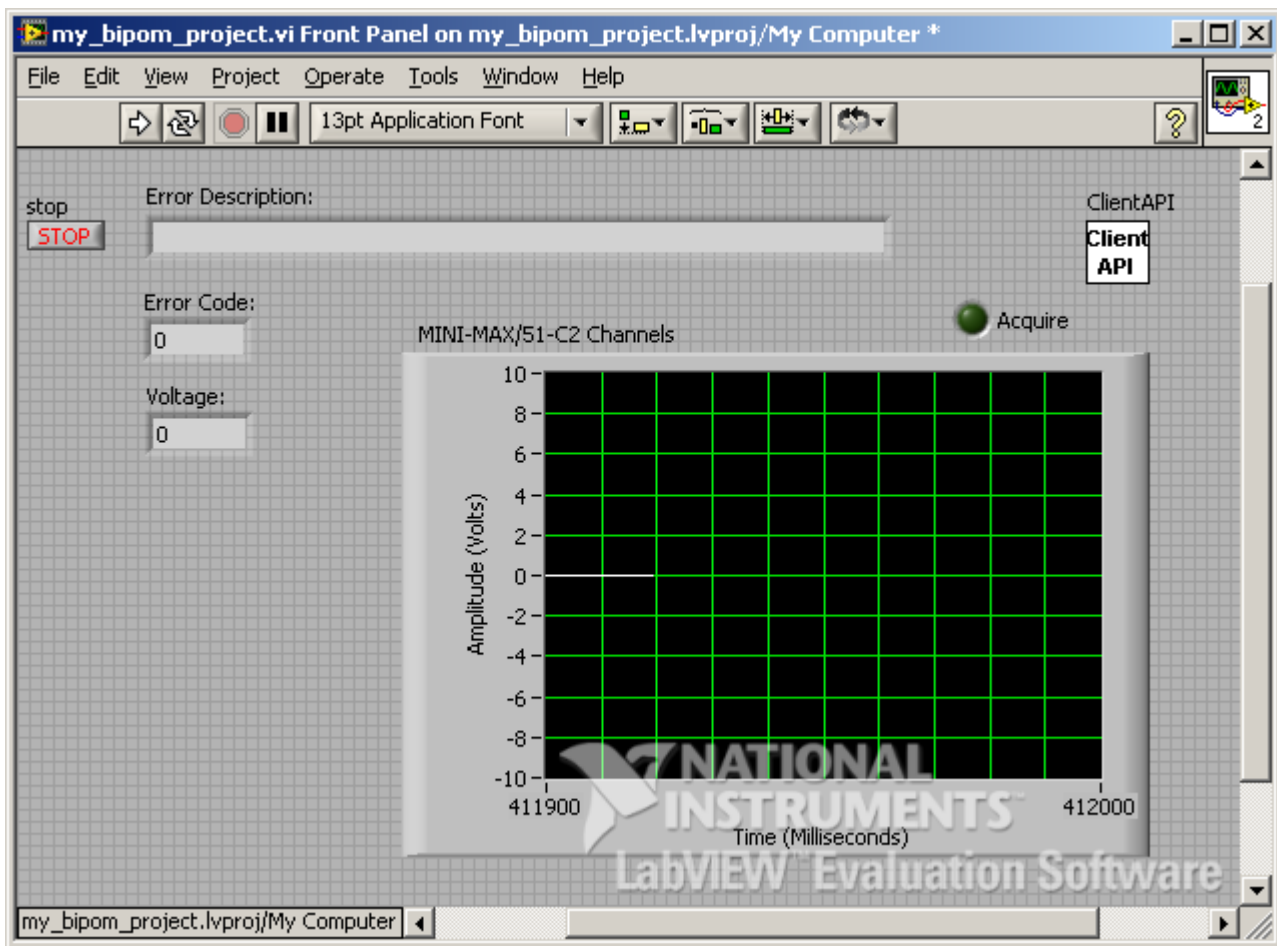


Now run the program and watch the voltage values from the board appear on the chart J :

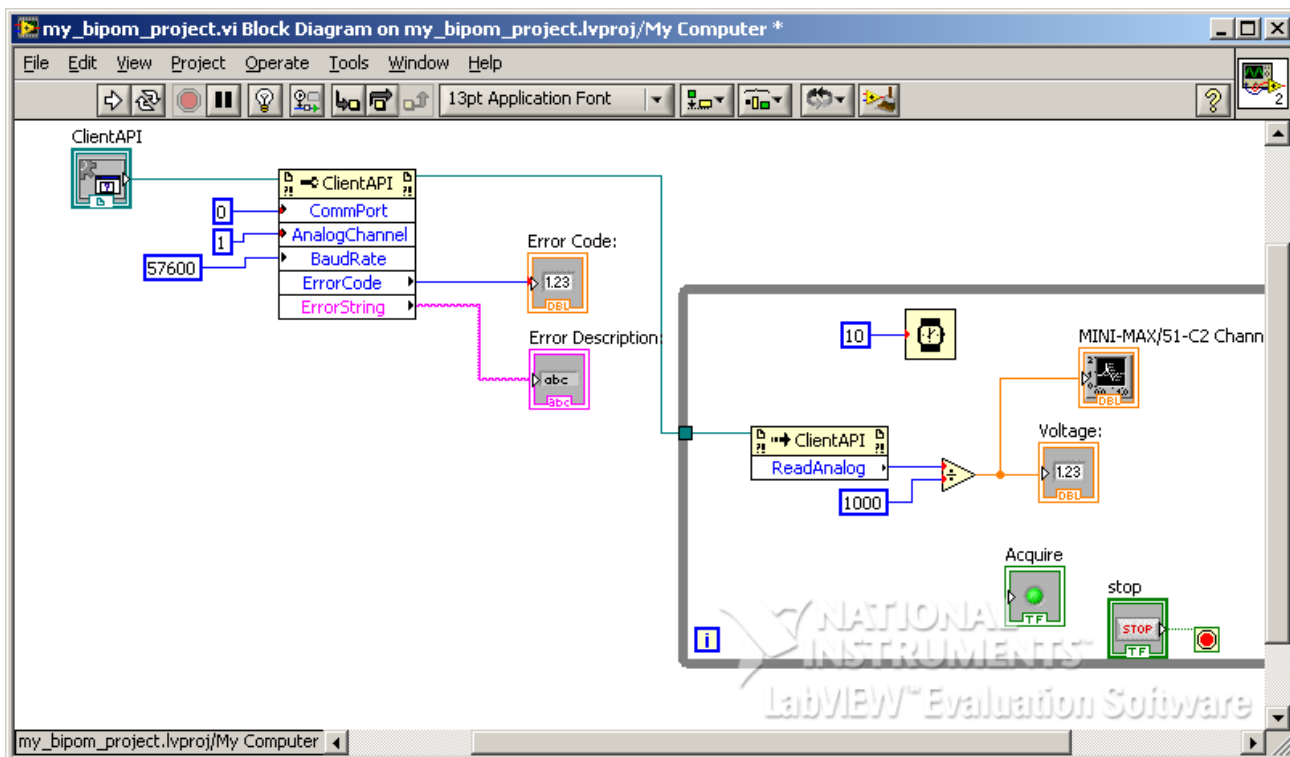


As next step, we want to add an indicator **LED** to the LabVIEW™ program. This **LED** will toggle every time the loop executes. We will tie this **LED** to the loop iteration counter.

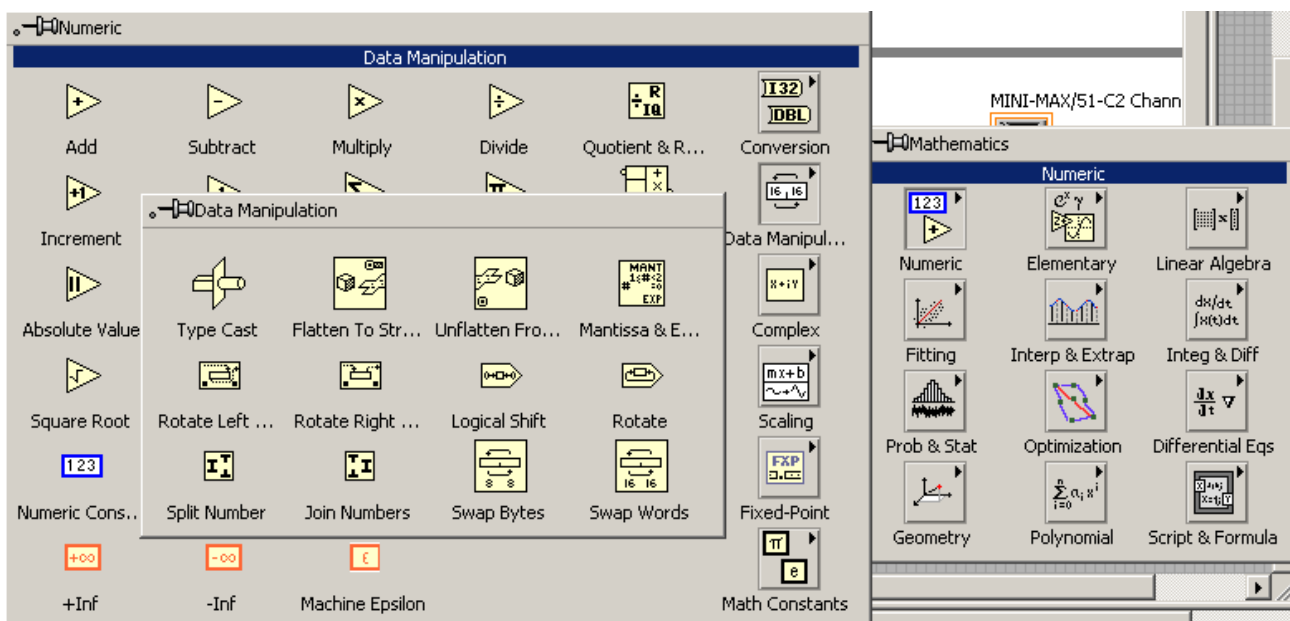
Add a **Round LED** to the Front Panel. Rename its label as **Acquire** and move the label to the right of the **LED**.



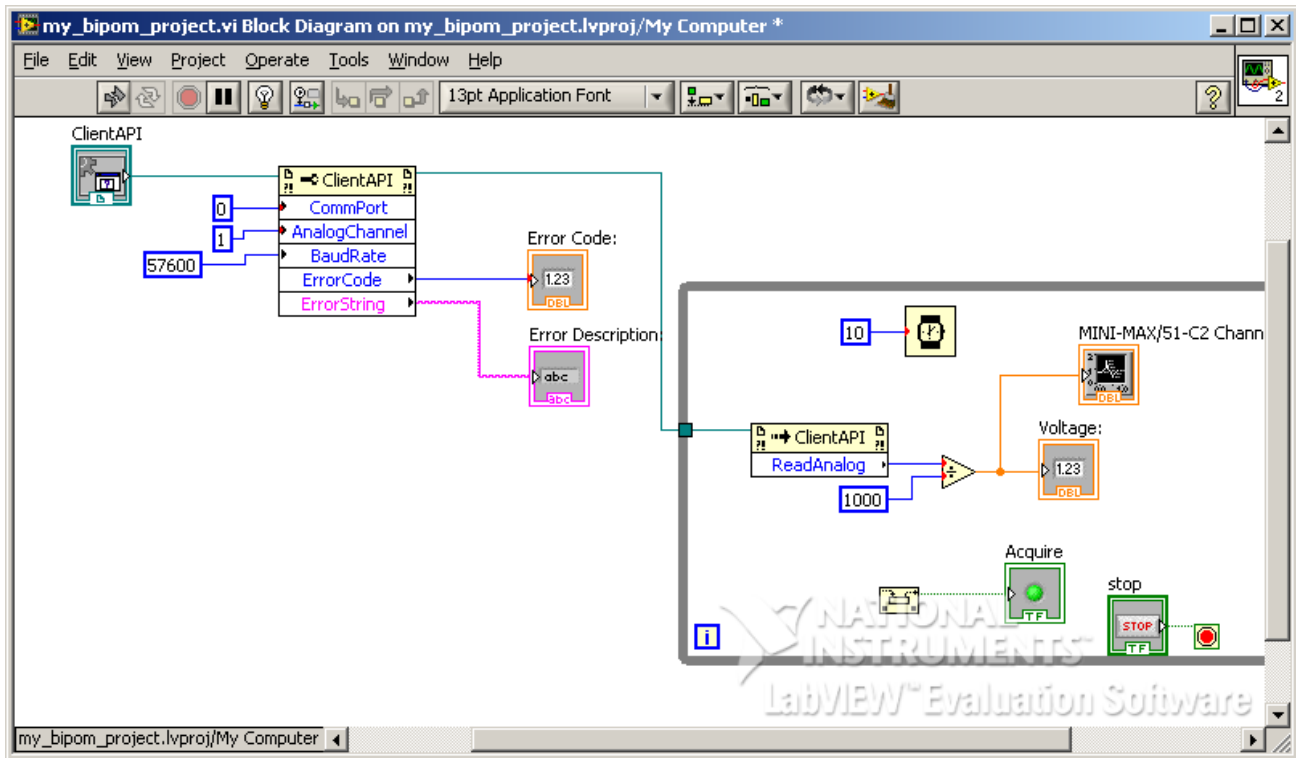
The **LED** will also appear on the Block Diagram. Move the **LED** to inside the **While Loop**:



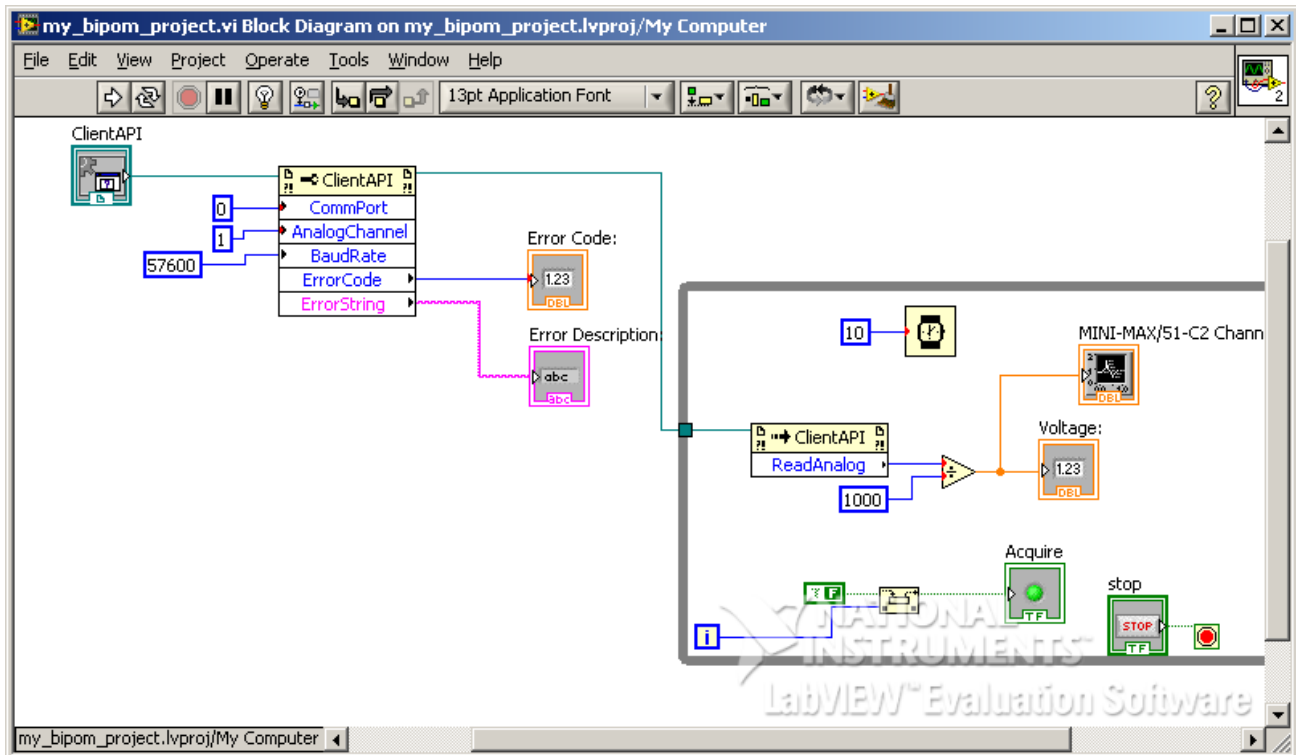
The iteration counter is an integer but the **LED** is a **Boolean** component (either on or off). To convert integer to **Boolean**, we feed the counter to a shift register. Add a **Rotate Right with Carry** function to the Block Diagram and place it inside the **While Loop**. **Rotate Right with Carry** function is under **Mathematics->Numeric->Data Manipulation** window of Functions menu:



Connect **LSB Carry Output** of **Rotate Right with Carry** function to the input of **LED**:



Connect **Value** input of **Rotate Right with Carry** to iteration counter. Connect a **Boolean FALSE** constant to carry input of **Rotate Right with Carry** function:



Now run the LabVIEW™ program. The green **LED** will blink as the data is being acquired.