Using LabVIEW[™]

with

BiPOM Boards

Quick Start Guide

Document Revision: 1.02

Date: 14 February, 2012



16301 Blue Ridge Road, Missouri City, Texas 77489
Telephone: 1-713-283-9970. Fax: 1-281-416-2806
E-mail: <u>info@bipom.com</u>
Web: <u>www.bipom.com</u>
© 2012 BiPOM Electronics, Inc. All Rights Reserved.
All trademarked names in this document are the property of respective owners.

Overview

LabVIEWTM (abbreviation for **Lab**oratory **V**irtual Instrumentation Engineering **W**orkbench) is a visual programming language and development environment from National InstrumentsTM. LabVIEWTM 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:

ClientAPI Software Architecture Overview

Client PC

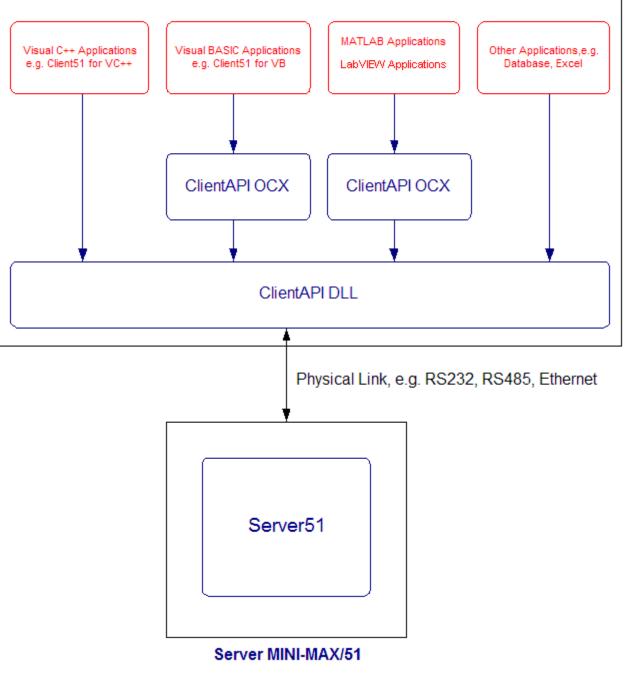


Figure 1

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

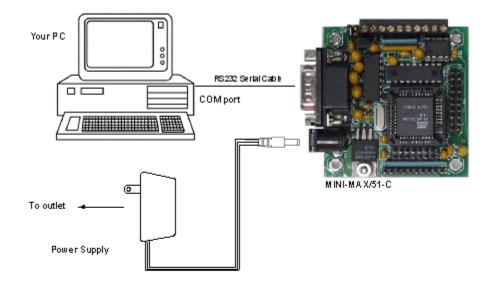
1. Place the MINI-MAX/51-C2 Microcontroller board on a clean, non-conductive surface.

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: <u>http://www.bipom.com/8051dev_down.php</u>

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

User Information		×
User Information	 You may lea 	er your name, company name and serial number below. ave the serial number field blank to run this software in ed demo mode.
		< <u>B</u> ack <u>N</u> ext > 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.

Micro-IDE - [server51.asm]								
 Eile Edit View Build Project Debuild Disc Edit View Build Project Debuild A man Call 		indow Help E 🏥 🎟 🖿 🔶 😨	1			M & % %		_ & ×
	- = ~		1	ብ ብ ብ		₽ % 🗗 📃 🗖		
server51.asm						erminal		×
;**************************************	******	********	*******	*****	_			<u> </u>
;*****	Server			****				
*****	06/22/ *******	′2UUb **********	*******	*****				
MULTIIO_SUPPORT	SET	DISABLE						
CPU ; ** SYSTEM **	SET	T89C51Rx2						
RESET_VECTOR	EQU	0						
RAM_MEMORY	EQU	8						
; ; ** COMMAND SET **								
INVALID_COMMAND	EQU	255						
CONTINUE_REPLY	EQU	254						
RESERVEDO_COMMAND READ_ANALOG_COMMAND	EQU EQU	0 1						
WRITE_ANALOG_COMMAND	EQU	2						
READ_PORT_COMMAND	EQU	3						
WRITE_PORT_COMMAND READ_MEMORY_COMMAND	EQU EQU	4 5						
WRITE_MEMORY_COMMAND	EQU	5 6						
SET_BIT_COMMAND	EQU	7						
CLR_BIT_COMMAND	EQU	8						
READ_SFR_COMMAND WRITE_SFR_COMMAND	EQU EQU	9 10						
CONTROL_I2C_COMMAND	EQU	11						
SET_BAUDRATE_COMMAND	EQU	12			-			
TIMED PHILSE COMMAND	FOU	13						v
×								
4								
4 •								F
Build Debug Find in Files 1 Find in	Files 2 Load	er						
Ready						Ln 10, Col 3	4 Disconnected	

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:

III Mici	o-IDE	- [ser	ver51.	asm]									
🔝 Eile	<u>E</u> dit	<u>V</u> iew	<u>B</u> uild	Project	<u>D</u> ebug	<u>T</u> ools	<u>W</u> indow	<u>H</u> elp					
] 🗅 🛛	j 🔒	Ø	X 🗈			9	۵ 🍪	۳C	0	9			
] 🖪 🕻		500 g	2 B							t t	1	{ }}	₽ (}
server	i1.asm												
;*	****	****	****	*****	*****	• ** *•	*****	****	****	*****	****	***	***
;*	****	×				Serv	zer51		\sim			**	***
;*	****	×				06/2	22/200)6	$\langle \rangle$			**	***
;**	****	****	****	****	****	(***)	*****	****	****	*****	****	***	***
, MU CPI		o_su		RΤ		SET SET)ISAE 78905	BLE 51Rx2	Dowi	nloa	d ic	on

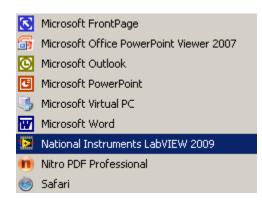
After the download completes, exit Micro-IDE and the board is ready to use with LabVIEW.

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 (<u>www.ni.com</u>)

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.

Lab¥IE₩		X
LabVIEW [®]		MATIONAL INSTRUMENTS [™]
License Status		
LabVIEW Development S Evaluation Application Builder Evaluation	16 days remaining 16 days remaining	Launch LabVIEW
		Activate Products
		Purchase Products
		Exit LabVIEW
Do not show this dialog again		More product information

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

🔛 Getting Started	
-------------------	--

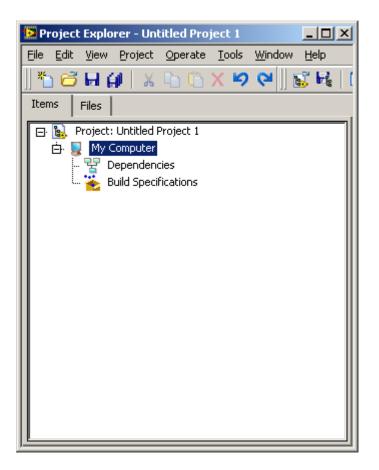
<u>File Operate Tools H</u>elp

LabVIEW 2009	LabVIEW Evaluation Mode
New Image: Ima	Latest from ni.com News (2) Technical Content Example Programs (1) Training Resources Online Support Discussion Forums Code Sharing KnowledgeBase Request Support Help LabVIEW Help List of All New Features
Browse	Find Examples

_ 🗆 🗡

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:

×
"Untitled Project 1" refers to new, unsaved files. You must save these files in order to save "Untitled Project 1".
Do you want to save the new, unsaved files?
Yes Cancel

Click Yes and give a name to your project:

Name the Projec	ct (Untitled Project 1)	<u>? ×</u>
Save jn:	📔 My Documents 💽 📀 🎓 🖽 -	
My Recent Documents Desktop My Documents My Computer	Arduino PDF files Dell WebCam Central rs232 DLS Guy Output Guy Ou	
My Network Places		OK Cancel

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

Name the ¥I (Un	itled 2)					<u>? ×</u>
Save jn:	🗎 My Documents 📃	6	1	P .	•	
My Recent Documents Desktop My Documents My Computer	Arduino PDF files Dell WebCam Central Srs232 DLS Downloads eagle GPS GpsGate GpsGateBuffer LabVIEW Data Music My Google Gadgets My Pictures My Videos My Webs					
My Network Places	File name: my_bipom_project					ОК
	Save as type: Vis (*.vi;*.vit)			•		Cancel New LLB

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

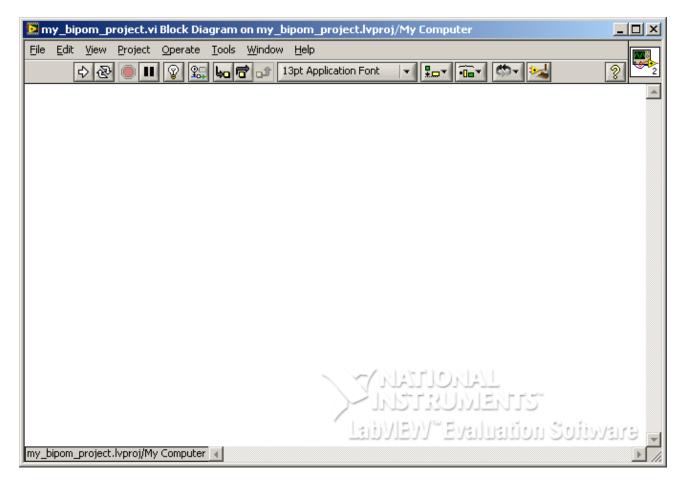
my_bipom_project.aliases my_bipom_project.lvlps my_bipom_project.lvproj my_bipom_project.vi

Your VI consists of 2 windows:

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

📴 my_bipom_project.vi Front Panel on my_bipom_project.lvproj/My Computer	
<u>Eile Edit View Project Operate Tools Window Help</u>	
🗘 🐼 🔘 💵 13pt Application Font 🖃 🏣 🖬 🖼 🏙 🕅	2
	<u> </u>
7 NATIONAL	
INSTRUMENTS	
LabVIEW "Evaluation Softwa	12
my_bipom_project.lvproj/My Computer	

• **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:

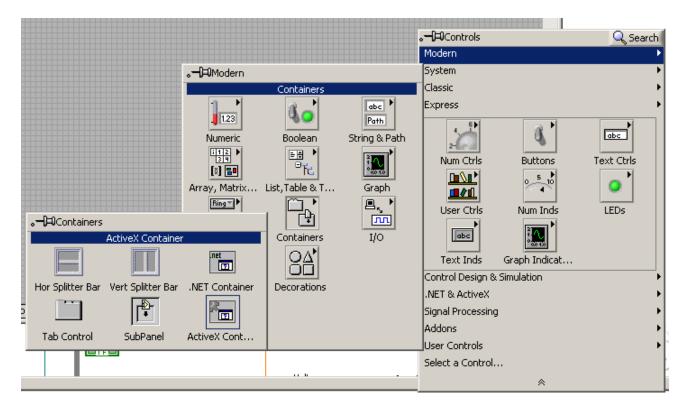
Insert the ActiveX control for ClientAPI into the Front Panel. To do this, right click on Front Panel. This will bring up *Controls* window:

₀-#Controls		🔍 Search
Express		•
4	C.	abc
Num Ctris	Buttons	Text Ctrls
	5 10	
User Ctrls	Num Inds	LEDs
abc	₹ 	
Text Inds	Graph Indicat	
User Controls		•
Select a Control.		
	*	

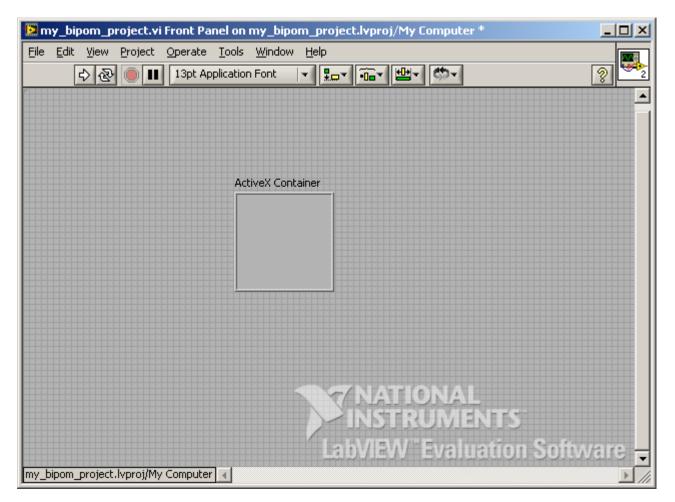
Click on the down arrow at the bottom of this window to expand the selections:

.−I¤Controls		🔍 Search
Modern		+
System		+
Classic		+
Express		+
Num Ctris	Buttons © 510 Num Inds Inds	Text Ctrls
Text Inds Control Design &	Graph Indicat Simulation	,
.NET & ActiveX		•
Signal Processing	l .	•
Addons		•
User Controls		•
Select a Control.		
	*	

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.

📴 my_bipom_project.vi Front Panel on my_bip	om_project.lvproj/My Comp	uter *	
File Edit View Project Operate Tools Window	v <u>H</u> elp		
다 🐼 🖲 💵 13pt Application Font	IT 🏪T 🎰 👑 🔅	*	8 2
ActiveX Cor	ntainer		
	Visible Items		
	Find Terminal		
	Change to Indicator		
	Description and Tip		
	Create		
	Replace	•	
	Data Operations		
	Advanced I		
	Fit Control to Pane		
	Scale Object with Pane	ENTS	
	✓ Autosizing	uation Softv	vare -
my_bipom_project.lvproj/My Computer	Insert ActiveX Object		
Inv_pipon_projeccivproj/my_compacer	Property Browser		

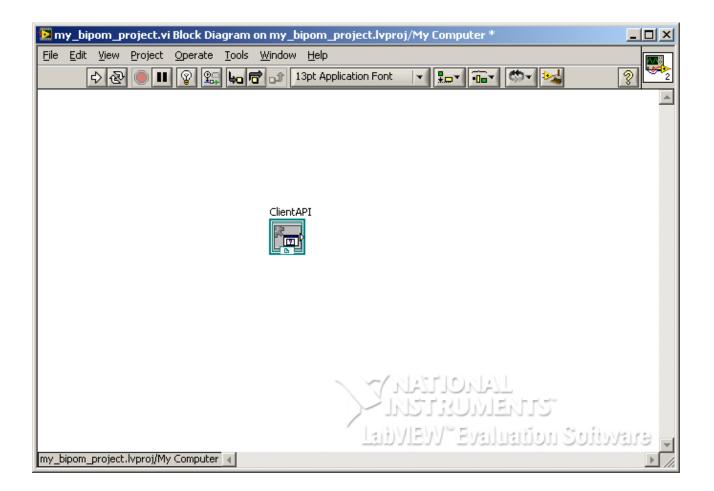
Select ClientAPI Control from the list:

📴 Select ActiveX Object 🛛 🗙							
Create Control							
Validate Servers							
Behavior Object							
BluEdit Control							
Calendar Control 8.0							
CDDBAppleControl Class							
Cdlg.RepCdg							
CfWheel Control							
ClientAPI Control							
CMS_OCX_H264							
COMNSView Class							
OK Cancel Help							

Click OK. Resize the ClientAPI object as needed:

😫 my_bipom_project.vi Front Panel on my_bipom_project.lvproj/My Computer *	
<u>Eile Edit View Project Operate Tools Window Help</u>	
2 2 2 13pt Application Font 🔽 🚛 🙃 🕊 🖘	2 2
ClientAPI	
API	
INSTRUMENTS	
LabVIEW [®] Evaluation Softwa	are 🚽
my_bipom_project.lvproj/My Computer 🔫	► ///

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.

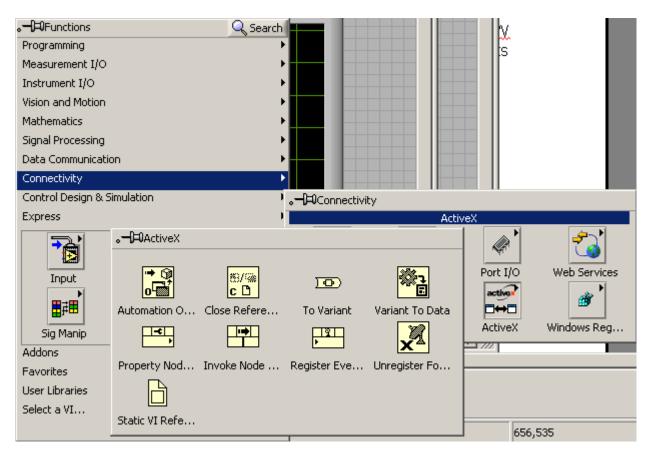
So far, we have successfully inserted the ClientAPI ActiveX Control into our LabVIEW[™] program. As next step, we will insert a Property Node that gives us access to the objects inside the ClientAPI ActiveX Control.

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

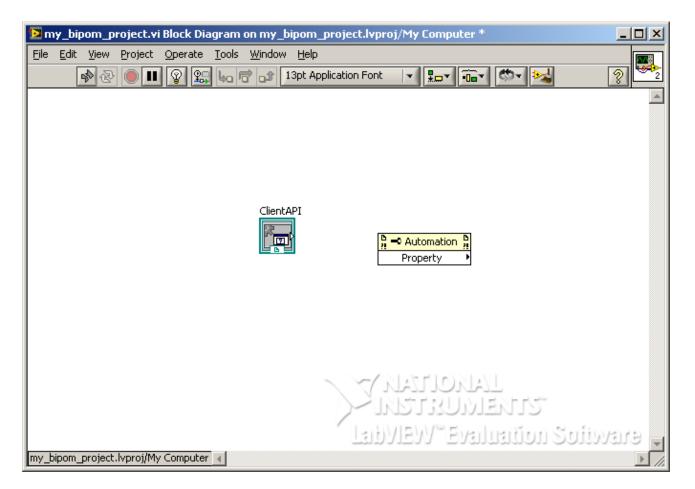


Click on the down arrow at the bottom of this window to expand the selections:

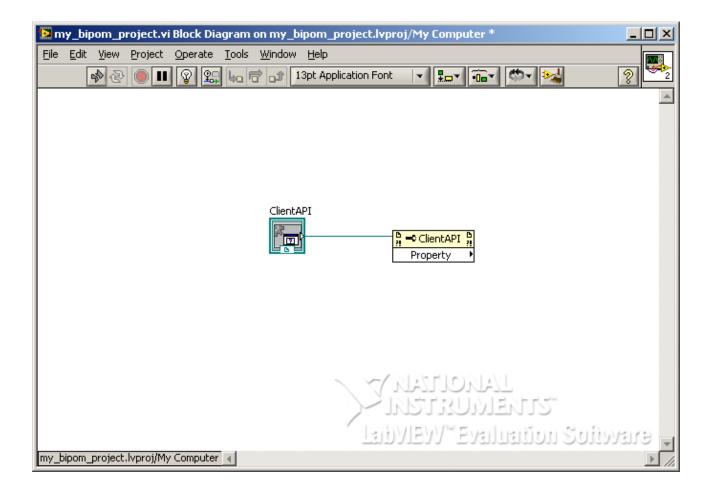
Select Connectivity->ActiveX->Property Node:



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



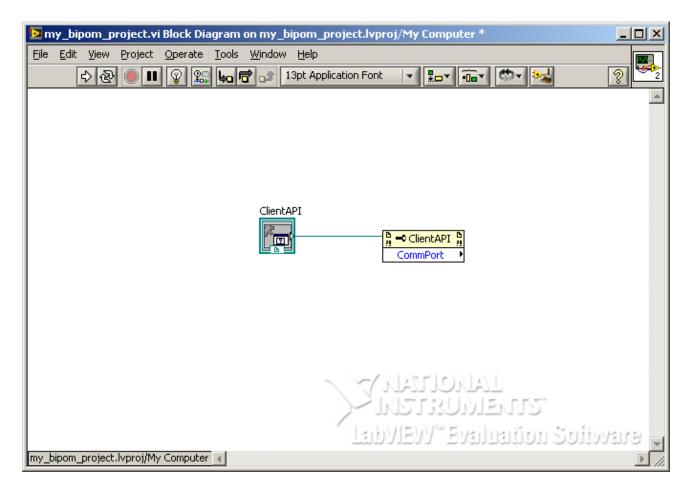
Connect ClientAPI ActiveX Control object to the *Property Node* using a wire. You will notice that the name of the *Property Node* will change to ClientAPI since it is now connected to and associated with ClientAPI ActiveX Control:



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

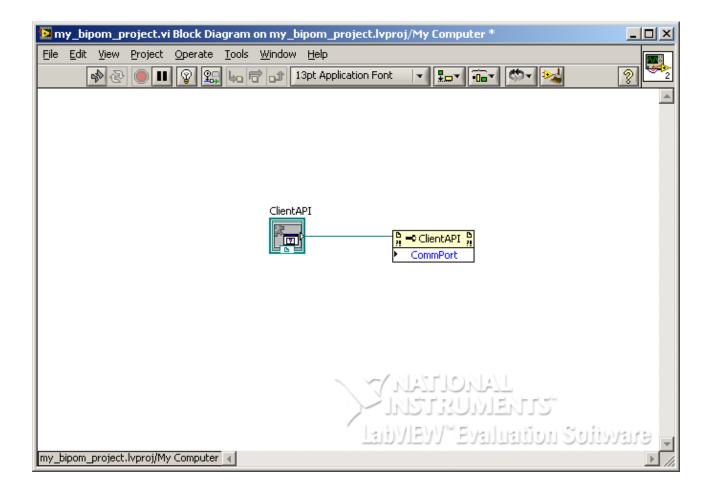
ClientAPI	
	Browse
	AnalogChannel
	BaudRate
	CommPort
	ErrorCode
	ErrorEvent
	ErrorString
/ WILLIN	IsInitialized
)~IRBTE	SFR
	Success
- WEINGET	Timer
	TimerMode

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

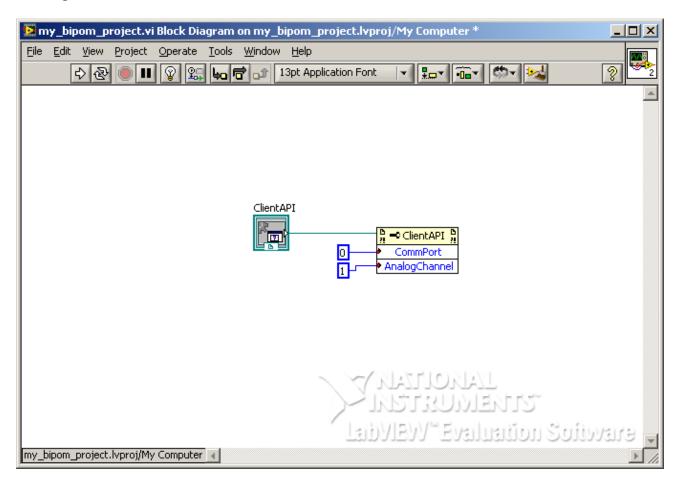
In this example, we will use COM1 so we assign a value of 0 to *CommPort* field. To do this, right click in Block Diagram area and select *Numeric Constant* under *Functions->Programming->Numeric*:

		°-	Functions		🔍 Search			
		P	rogramming		۱.			
.− ⊨0Numeric		•			•	-DProgrammin	g	
							Numeric	
⊳	Þ	\geqslant	\Rightarrow	÷ R 10] <u>I32</u> ►]DBL		B 1 2 0 3 4	
Add	Subtract	Multiply	Divide	Quotient & R	Conversion	Structures	Array	Cluster, Class
₽	₽	Σ	Þ			123		abc ►]a fi
Increment	Decrement	Add Array Ele	Multiply Array	Compound Ar	Data Manipul	Numeric	Boolean	String
				*	X+iY		Q,	
Absolute Value	Round To Ne	Round Towar	Round Towar	Scale By Pow	Complex	Comparison	Timing	Dialog & User
\triangleright	×		\geqslant		mx+b ~~+^v			
Square Root	Square	Negate	Reciprocal	Sign	Scaling	File I/O	Waveform	Application C
123	• Enum	Ring	Ð	EXPR]	FXP D.C	!	* }	, ₽ ₽
Numeric Cons	Enum Constant	Ring Constant	Random Num	Expression N	Fixed-Point	Synchronization	Graphics & So	Report Gener
+00	-00	C			π ► e			
+Inf	-Inf	Machine Epsilon			Math Constants		لفك	201200 20

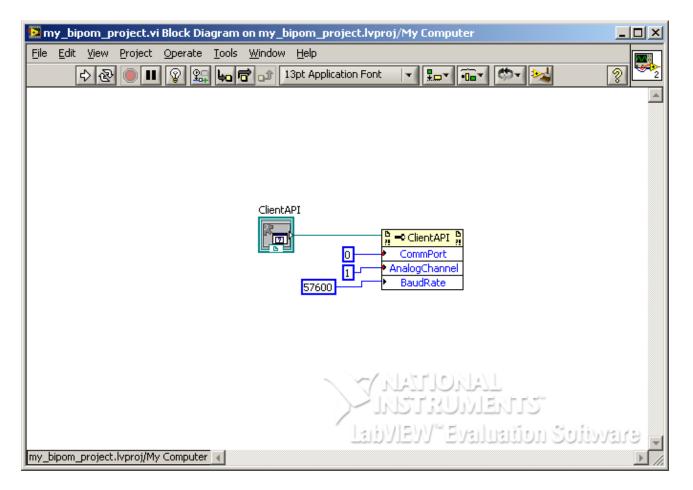
Drag and drop *Numeric Constant* onto the Block Diagram and assign it a value of 0. Also, connect the *Numeric Constant* to the *CommPort* field of *Property Node*:

📴 my_bipom_project.vi Block Diagram on my_bipom_project.lvproj/My Computer *	
<u>File E</u> dit <u>V</u> iew <u>P</u> roject <u>O</u> perate <u>T</u> ools <u>W</u> indow <u>H</u> elp	
🗘 🕸 🖲 💵 😵 🕵 🏎 🖻 🔐 13pt Application Font 🖃 🏣 🖬 🕬 🚧 🔗	2
ClientAPI	
ClientAPI 🙀 CommPort	
KLASTIONAL	
MIKELLEUMERLIZ.	
LabVIEW"Evaluation Softwar	(a
my_bipom_project.lvproj/My Computer	

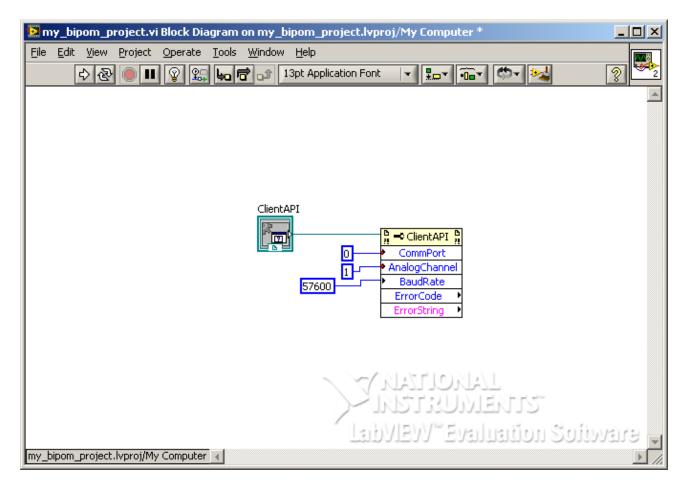
So far, we have successfully assigned a value of 0 to *CommPort* field. Now we want to add more fields to configure in our *Property Node*. Right click on *CommPort* field and select *Add Element* from the menu. This will add a second field, also with the name of *CommPort*. To change this, right click 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:



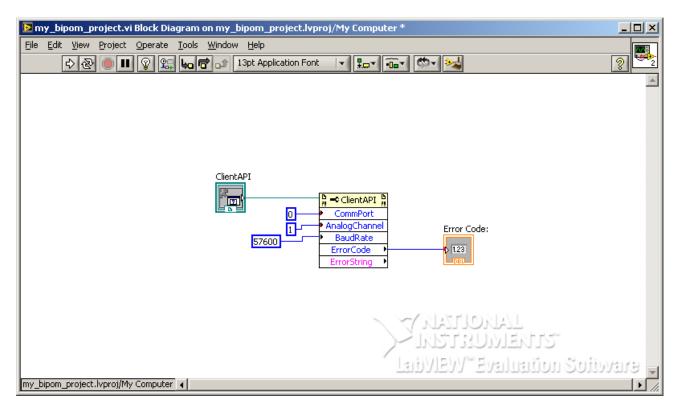
Add two more fields to the **Property Node**: **ErrorString** and **ErrorCode**. **ErrorCode** is the error code returned from ActiveX functions and **ErrorString** is human readable description of the error code. After adding these fields, change them to Read-only using **Change To Read** option from 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:*

😰 my_bipom_project.vi Front Panel on my_bipom_project.lvproj/My Computer *	<u> </u>
<u>File E</u> dit <u>V</u> iew <u>P</u> roject <u>O</u> perate <u>T</u> ools <u>W</u> indow <u>H</u> elp	
🗘 🐼 🛑 💵 13pt Application Font 🛛 🚛 🖬 🕮 🏧	2
Error Code:	ClientAPI Client API
NATIONAL INSTRUMENTS	
LabVIEW" Evaluation S	oftware 🖵
my_bipom_project.lvproj/My Computer 4	▶ //

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:

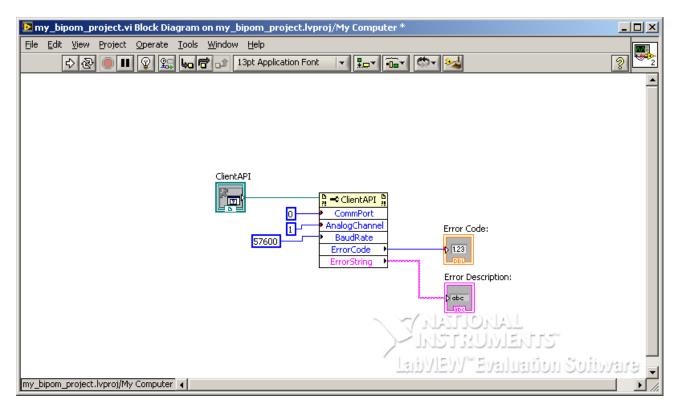
🔯 my_bipom_project.vi Front Panel on my_bipom_project.lvproj/My Computer *	
<u>File E</u> dit <u>V</u> iew <u>P</u> roject <u>O</u> perate <u>T</u> ools <u>W</u> indow <u>H</u> elp	
🗘 🕑 🔲 13pt Application Font 🖃 🏣 🖬 🖼	2
Run Continuously	ClientAPI Client API
Error Code:	
1059	
INSTRUMENTS	
INSTRUMENTS	
LabVIEW "Evaluation So	oftware 🖵
my_bipom_project.lvproj/My Computer 4	•

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:*

🔯 my_bipom_project.vi Front Panel on my_bipom_project.lvproj/My Computer *	
<u>File E</u> dit <u>V</u> iew <u>P</u> roject <u>O</u> perate <u>T</u> ools <u>W</u> indow <u>H</u> elp	
💠 🐼 🔘 💵 13pt Application Font 🛛 🛪 🚛 🖬 🐨	2 2
Error Description:	ClientAPI Client API
Error Code:	1
1059	
NATIONAL	
INSTRUMENTS	
LabVIEW [®] Evaluation S	offware -
my_bipom_project.lvproj/My Computer	▼

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:

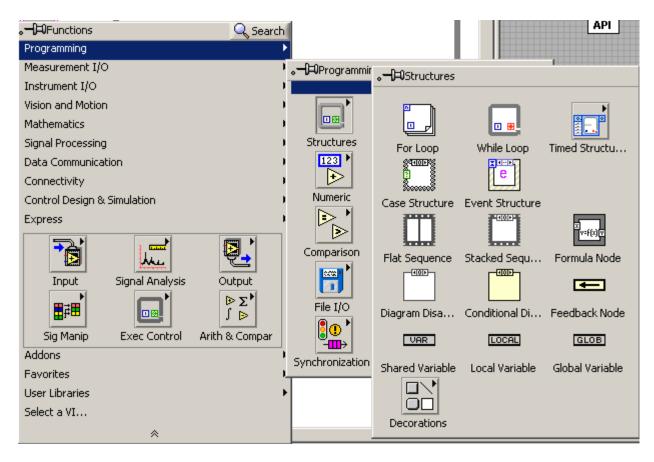
🔛 m	ny_bip	om_p	roject. v i					
Eile				<u>O</u> perate	<u>T</u> ools	<u>W</u> indow	Help	
	I	₽ 2						2 2
		Erro	^r Descripti	ion.				
		_		baud rate.	-	_		1
		Jene	Ji secong	bada rate.				
		Error	Code:					
		102	2					
							NATIONA	\L
							INSTRUM	ENTS
					_		Labvievv Eval	luation Software 🚽
my_t	pipom_	project.	lvproj/My	Computer	•			

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:

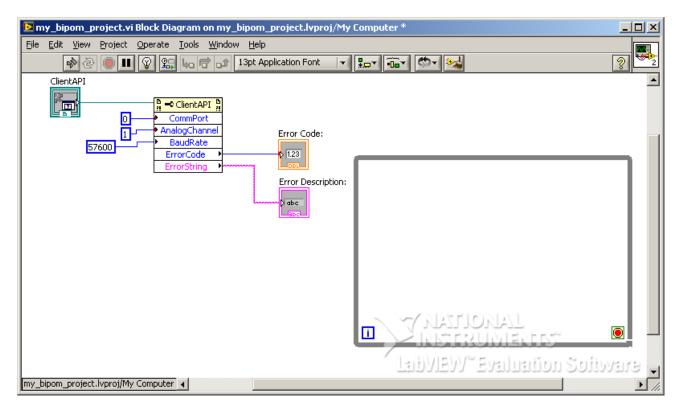
1	n y_b i	pom_p	roject. v i	i				
Eile				Operate	<u>T</u> ools	<u>W</u> indow	Help	
		₽						8 2
		Error	^r Descripti	ion				_
			boschpa		_	_		T I I I I I I I I I I I I I I I I I I I
		Error	Code:					
		JU						
								M
							INSTRUM	ENTS"
							Labvievv Eva	luation Software 🚽
my_	bipom_	project.	lvproj/My	/ Computer				

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:

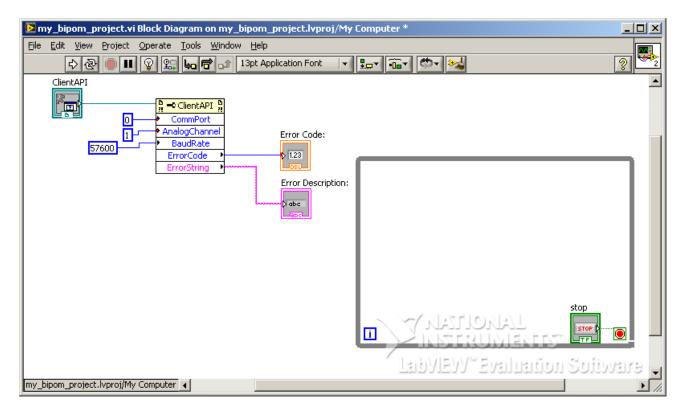


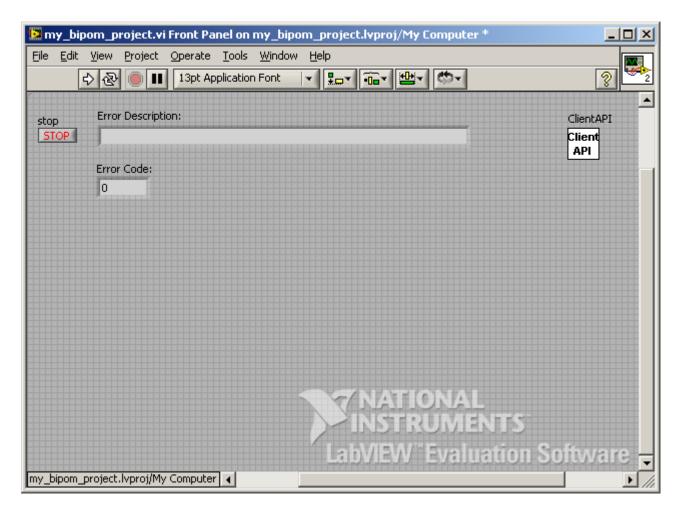
Drag and drop the *While Loop* into the Block Diagram and resize it bigger to allow other blocks to be inserted into the *While Loop*:



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:

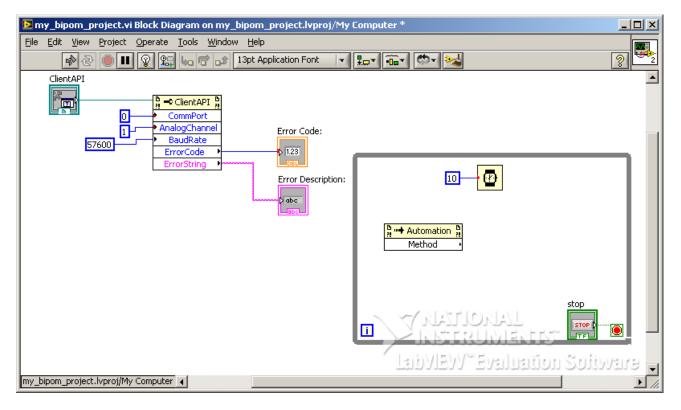
📴 my_bipom_project.vi Block Diagram on my_bipom_project.lvproj/My Computer *	<u> </u>
Eile Edit View Project Operate Tools Window Help	
🔊 🕙 🔲 😰 🕵 🏎 🗃 🔐 13pt Application Font 🖃 🚛 🖬 🐨 🛀	2
ClientAPI	stop
my_bipom_project.lvproj/My Computer 🔺	

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

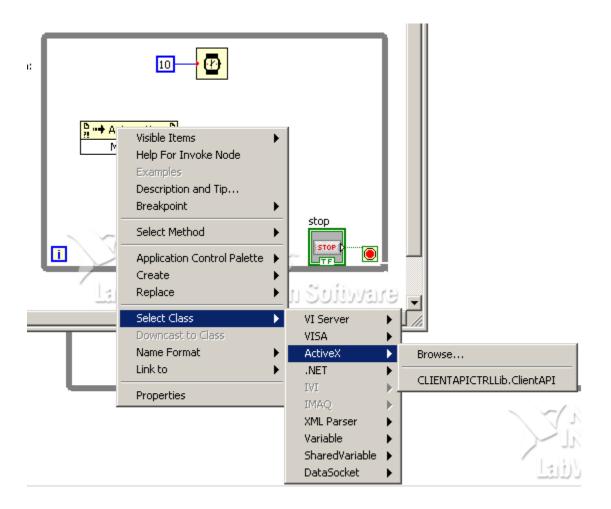
📴 my_bipom_project.vi Block Diagram on my_bipom_project.lvproj/My Computer *	<u> </u>
<u>Eile Edit View Project Operate Tools Window H</u> elp	
🗘 🕑 🔲 😰 🕵 🏎 🖻 🗗 13pt Application Font 🖃 🏣 🖬 🐨 🚧	2
	stop stop Store 1 Solityvara
my_bipom_project.lvproj/My Computer 🖌	▶ //.

This means that the loop will execute every 10 milliseconds.

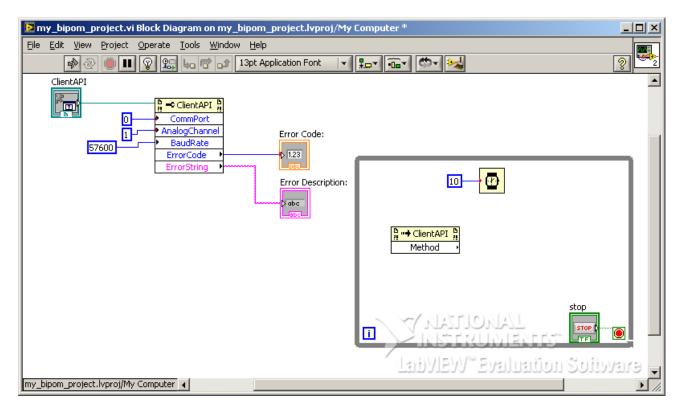
Now that we have a loop that runs every 10 milliseconds, we will add an *Invoke Node* to the *While Loop* to invoke the ClientAPI *ReadAnalog* function. This is the function that communicates with the board and reads one analog value. Right click in Block Diagram to bring the Functions menu and select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



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



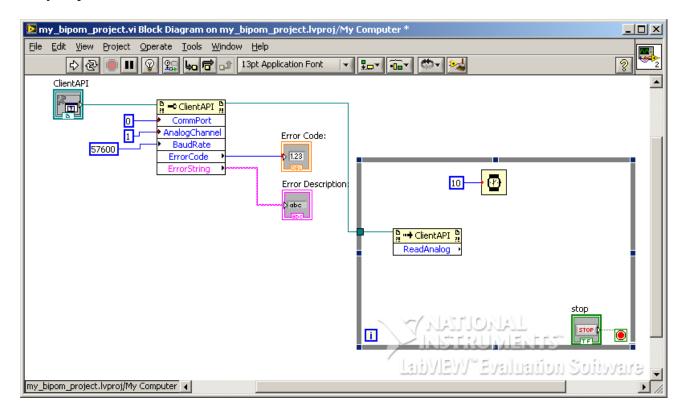
Note that the name of *Invoke Node* changes to ClientAPI:



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

📴 my_bipom_project.vi Block Diagram on my_bipom_project.lvproj/My Computer *	<u>- 🗆 ×</u>
<u>File Edit View Project Operate Tools Window Help</u>	
🔊 🕑 🔲 😰 🕵 🦕 🖻 🗗 13pt Application Font 🖃 🏣 🖬 🔂	
ClentAPI	
my_bipom_project.lvproj/My Computer 4	

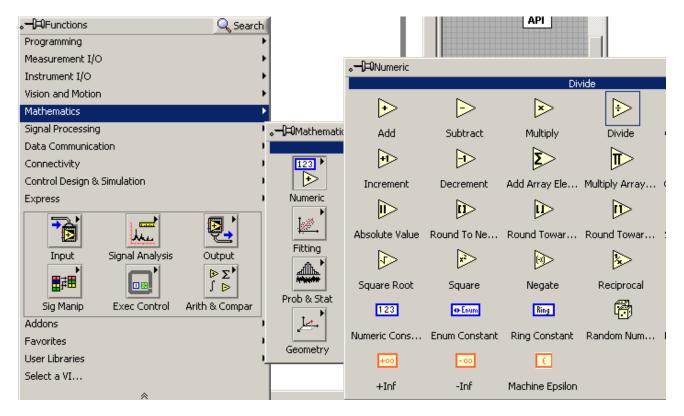
Using a wire, connect the *Reference* input of *Invoke Node* to *Reference Output* of *Property Node*:



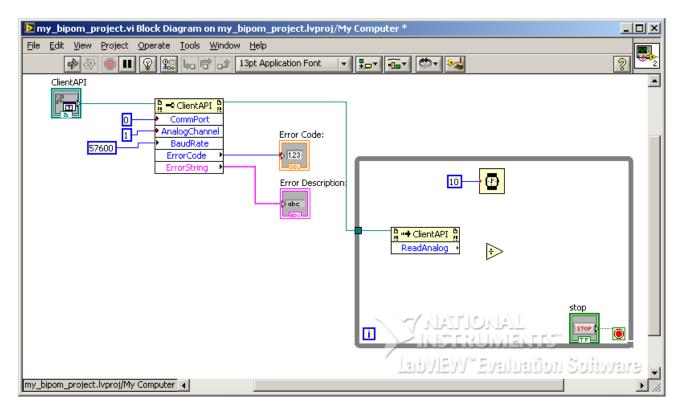
ReadAnalog function returns a raw value from 0 to 4095 corresponding to a voltage of 0 to 4.095 Volts. So to display real voltages, we use the following formula:

Voltage = Raw Value / 1000

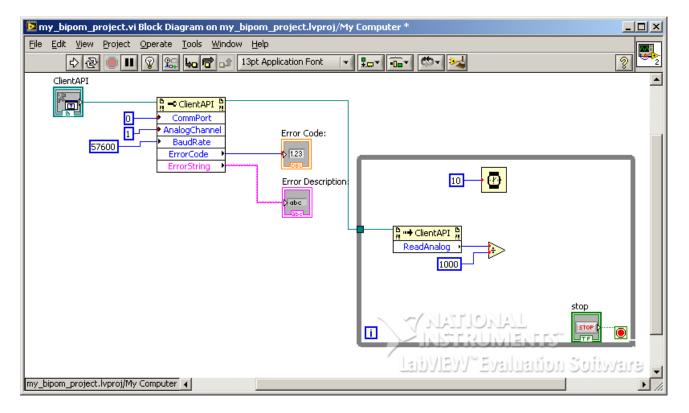
To implement this, we add a divider to the LabVIEW[™] program. Right click on Block Diagram and bring up the Functions menu. The *Divide* function is under *Mathematics->Numeric*:



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



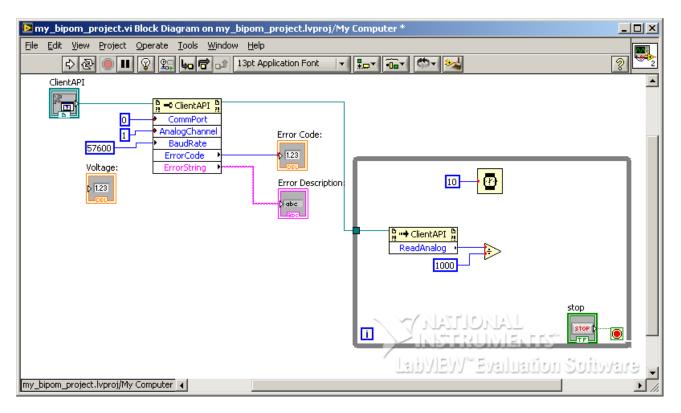
Connect the *ReadAnalog* output of *ReadAnalog* function to *X input* of *Divider* using a wire. Connect a *Numeric Constant* of 1000 to *Y input* of *Divider*. This will implement ReadAnalog/1000 operation:



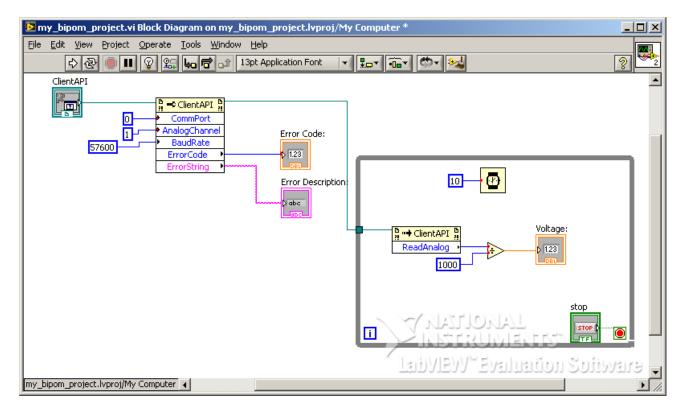
Now comes the exciting part. We are about to display our voltage value on the Front Panel. Add a *Numeric Indicator* to Front Panel by selecting *Num Ind* from under *Num Inds* in Controls menu. Rename the *Numeric Indicator* as Voltage:

🔯 my_bipom_project.vi Front Panel on my_bipom_pro	ject.lvproj/My Computer *
<u>File Edit View Project Operate Tools Window H</u> elp	
🗘 🚱 🛑 💵 13pt Application Font 🖃 🛼	- 🔤 🖳 🖉 🕺
stop Error Description:	ClientAPI Client API
Error Code:	
0	
Voltage:	
0	
	7 NATIONAL INSTRUMENTS
my_bipom_project.lvproj/My Computer 4	abVIEW "Evaluation Software

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



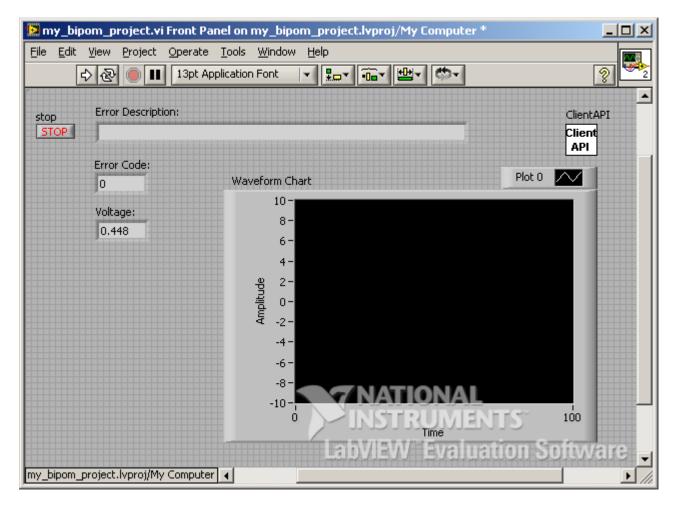
Move the *Numeric Indicator* to inside *While Loop* and connect its input to *X/Y output* of *Divider*:



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

	n y_ bi	pom_p	roject.vi	i				
Eile				Operate	<u>T</u> ools	<u>W</u> indow	Help	
								2 2
		Erro	r Descript	ion:				_
sti	TOP							Ĩ
		Erro	r Code:					-
		0	r coue.					
		Volta	ane:					
		4.0	_					
							INSTRUM	
							LabVIEW [™] Eva	luation Software 🖵
my_	bipom	_project	.lvproj/M _y	/ Computer				

As next step, we want to display the value from the board also on a graphical chart. With LabVIEWTM, 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*.

😥 Chart Properties: Waveform Chart	×
Appearance Display Format Plots Label ✓ Visible ✓ Visible ✓ MINI-MAX/51-C2 Channels ✓ Enabled State ✓ ⑥ Enabled ⑦ Disabled ⑦ Disabled & grayed	Scales Documentation Data Binding Caption Visible Size Height Width 252 374
 Show graph palette Show plot legend Auto size to plot names 1 Plots shown Show x scroll bar Show scale legend Show cursor legend 	Update mode Scope Chart Stack plots Show digital display(s) Show optional plane None Cartesian lines Optional plane labels Optional plane lines
	OK Cancel Help

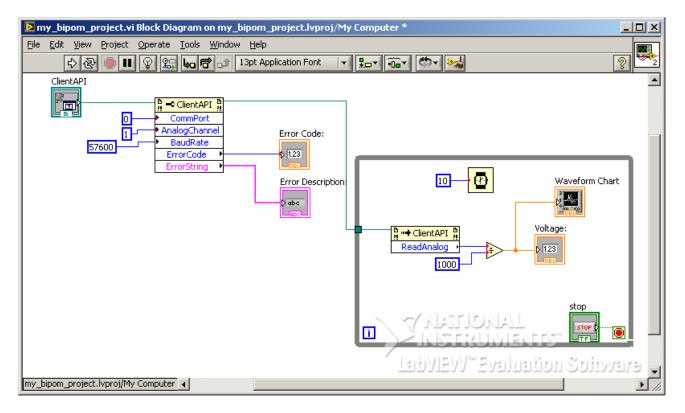
Click on *Scales* tab. Rename *X-Axis* as *Time (Milliseconds)* and *Y-Axis* as *Amplitude (Volts)*. Select green as the color for both X and Y axes.

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

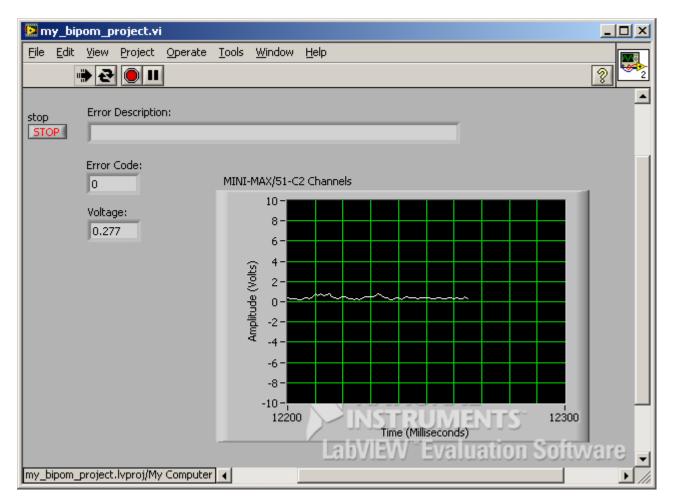
📴 Chart Properties: Waveform Chart	×
Appearance Display Format Plots Scales Documentation Data Binding	1
Amplitude (Volts) (Y-Axis)	
Name Amplitude (Volts)	
Show scale label	
Show scale -10 Minimum	
Log 10 Maximum	
Inverted Scaling Factors	
Expand digital buses 0 Offset	
1 Multiplier	
	_
1.0- Major tick	
0.5- Minor tick	
Marker text	
Ignore waveform time stamp on x-axes	
OK Cancel	Help

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*):

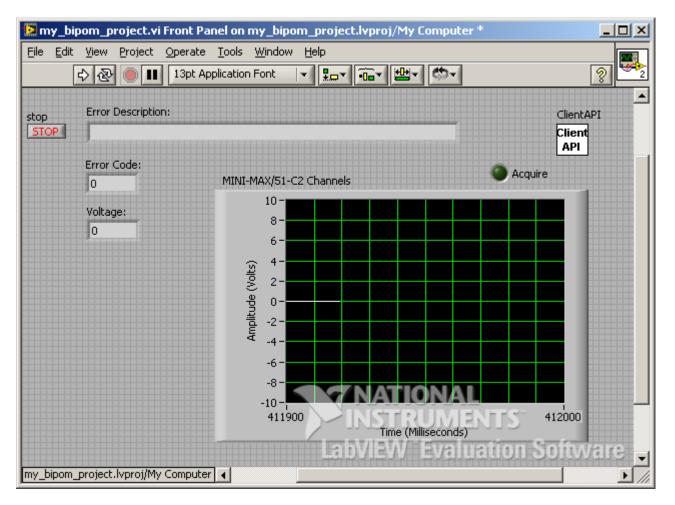


Now run the program and watch the voltage values from the board appear on the chart ${f J}$:

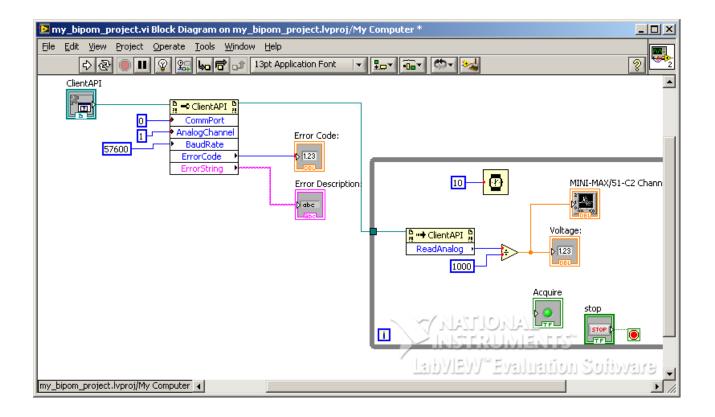


As next step, we want to add an indicator *LED* to the LabVIEW[™] program. This *LED* will toggle every time the loop executes. We will tie this *LED* to the loop iteration counter.

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



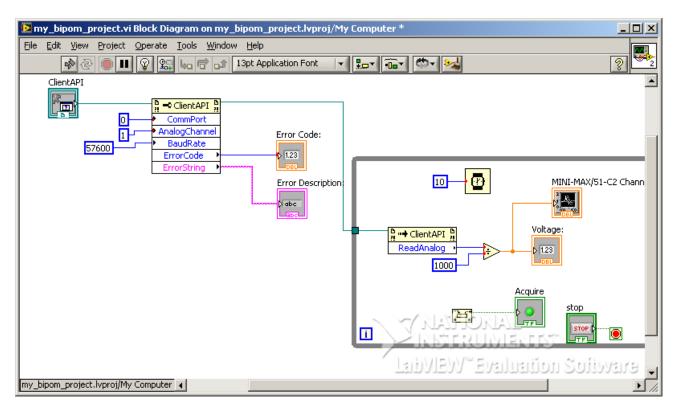
The LED will also appear on the Block Diagram. Move the LED to inside the While Loop:



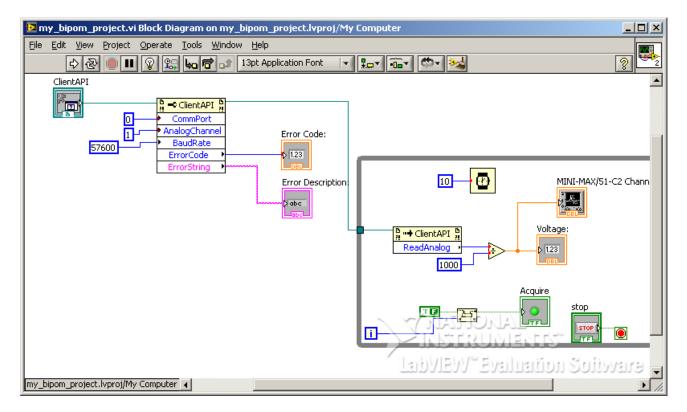
The iteration counter is an integer but the *LED* is a *Boolean* component (either on or off). To convert integer to *Boolean*, we feed the counter to a shift register. Add a *Rotate Right with Carry* function to the Block Diagram and place it inside the *While Loop*. *Rotate Right with Carry* function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

.− 戸Numeric								
		Data Ma						
\triangleright	Þ	\triangleright	Þ	÷R IQ] <u>I32</u> ►]DBL		IINI-MAX/51-C2 C	hann
Add	Subtract	Multiply	Divide	Quotient & R	Conversion	-D Mathematic	s	
+1 >		~	T	¶.×]			Numeric	
	₀—Ĵ≓0Data Manip	oulation				123	¢°γ► B <mark>≷s</mark> o	
Increment					Data Manipul			
	e for	<mark>∲</mark> ⊊;	<mark>,Z¶a</mark>	MANT 1{#<2 # =0 EXP	×+iY		Elementary	Linear Algebra
Absolute Value	Type Cast	Flatten To Str	Unflatten Fro	Mantissa & E	Complex			Jx(t)dt
\triangleright		<u> </u>	0+0+0	Ð	mx+b	Fitting	Interp & Extrap	Integ & Diff
Square Root	Rotate Left	Rotate Right	Logical Shift	Rotate	Scaling	Prob & Stat	Optimization	Differential Eqs
123	I	II			FXP		· •	
Numeria Cons					Fixed-Point	14-	$\sum_{i=0}^{n} \alpha_{i} x^{i}$	X 4(4) X=1; Y
Numeric Cons	Split Number	Join Numbers	Swap Bytes	Swap Words		Geometry	Polynomial	Script & Formula
+00	-00	E					- ciynomiai	
+Inf	-Inf	Machine Epsilon			Math Constants			

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.