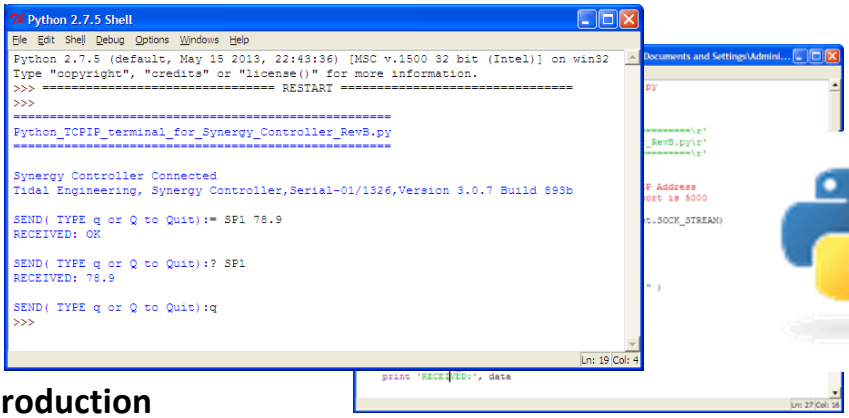
Synergy Controllers and the Python Programing Language



Introduction

This application note demonstrates the simplicity of the Python programming language the versatility of the various communication ports available on the Synergy Controller family; and.

- Ethernet (TCP/IP)
- GPIB (IEEE 488)
- RS-232
- ModbusTCP

Tidal Engineering's Synergy Controllers, including the Synergy Micro 2, Synergy Quattro, and the ¼ DIN Synergy Nano provide state-of-the-art usability and connectivity for environmental test control and data acquisition. They combine the functions of a chamber controller and a data logger. They are designed to improve test efficiency by supporting both factory automation and test and measurement protocols and standards. Offering the flexibility of multiple communication ports including Ethernet, GPIB, and RS-232 make these controllers perfect for today's changing testing environments.

Python is an Open Source (Free) programming language that can run on virtually any modern platform; Windows, Linux, Mac, etc. Three Python example applications are included in this application note as follows:

Example 1:

Ethernet TCP/IP control example using Python and Sockets.

Example 2:

VISA (PyVISA) to support simultaneous Synergy Controller communications using:

Ethernet (TCP/IP)
GPIB (IEEE 488)
RS-232.

Example 3

ModbusTCP

The Virtual Instrument Software Architecture (VISA) is a popular and powerful I/O framework for communicating with test and measurement instruments from a PC. VISA is an industry standard implemented by multiple Test & Measurement companies including National Instruments and Agilent Technologies.

PyVISA provides a remarkably simple and straightforward means to interface to the Synergy Controllers multiple communications ports using a VISA driver.

To use these example programs, follow these setup steps:

1. Appendix B Python language setup i.
2. Appendix C PyVISA setup.
3. Appendix D PyModbusTCP setup.
4. Appendix E NI VISA setup.

The first example on the following page shows off the simplicity of the Python Programming language and the power of the Synergy Controller's TCP/IP capability. In 18 lines, the program establishes communications with the controller and then waits for the user to type commands, displaying the controller's response to each one.

The source code for this example is shown below the screenshots so that it can be quickly cut and pasted into the reader's program.

The command syntax for the Synergy Controller is the straightforward ASCII and the same for all three models.

For example:
To set the chamber temperature (Channel 1) to 78.9, the command is "= SP1 78.9\r\n".
To query the chamber temperature setpoint (Channel 1), "? SP1\r\n".
To query the chamber temperature (Channel 1) "? C1\r\n"
To turn the chamber on "= ON\r\n"

To turn the chamber off “= OFF\r\n”

All commands and queries should be terminated with carriage return <cr>and line feed <lf>. (\r\n). Also note that the controller will interpret temperature commands and send responses in the current units of measure, C or F.

See Appendix A for a list of Synergy Controller frequently used commands.

Example 1: Synergy Controller Programming with Python and TCPIP Sockets

Program Window

Python_TCPIP_Example_for_Synergy_Controller_Rev_D.py - //tidal01/Tidal/Users/TIDAL ENGINEERING...

File Edit Format Run Options Window Help

```
# Python_TCPIP_Example_for_Synergy_Controller_Rev_D.py
#Python 3.7.4
import socket

HOST = '172.16.200.109'   # The Synergy Controller IP Address
PORT = 5000               # The Synergy Controller port is 5000

client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client_socket.connect((HOST, PORT))
msg="*idn?\r\n"
client_socket.send(msg.encode('ascii'))
data = client_socket.recv(1024)
print ('Synergy Controller Connected\r\n', data)
while 1:
    data = input ( "SEND( TYPE q or Q to Quit):" )
    if (data != 'Q' and data != 'q'):
        msg=(data + "\r\n")
        client_socket.send(msg.encode('ascii'))
    else:
            msg=(data + "\r\n")
            client_socket.send(msg.encode('ascii'))
            client_socket.close()
            break;
    data = client_socket.recv(512)
    print ('RECEIVED:', data)
```

Ln: 25 Col: 17

Output Window

Python 2.7.5 Shell

File Edit Shell Debug Options Windows Help

```
Python 2.7.5 (default, May 15 2013, 22:43:36) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> ================================ RESTART ================================
>>>
==========================================================
Python_TCPIP_terminal_for_Synergy_Controller_RevB.py
==========================================================

Synergy Controller Connected
Tidal Engineering, Synergy Controller,Serial-01/1326,Version 3.0.7 Build 893b

SEND( TYPE q or Q to Quit):= SP1 78.9
RECEIVED: OK

SEND( TYPE q or Q to Quit):? SP1
RECEIVED: 78.9

SEND( TYPE q or Q to Quit):q
>>>
```

Ln: 19 Col: 4

Example 1 Source Code

```
# Python_TCPIP_Example_for_Synergy_Controller_Rev_D.py
# Python 3.7.4
import socket

HOST = '172.16.200.109'   # The Synergy Controller IP Address
PORT = 5000             # The Synergy Controller port is 5000

client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client_socket.connect((HOST, PORT))
msg="*idn?\r\n"
client_socket.send(msg.encode('ascii'))
data = client_socket.recv(1024)
print ('Synergy Controller Connected\r\n', data)
while 1:
   data = input ( "SEND( TYPE q or Q to Quit):" )
   if (data != 'Q' and data != 'q'):
      msg=(data + "\r\n")
      client_socket.send(msg.encode('ascii'))
   else:
         msg=(data + "\r\n")
         client_socket.send(msg.encode('ascii'))
         client_socket.close()
         break;
   data = client_socket.recv(512)
   print ('RECEIVED:', data)
```

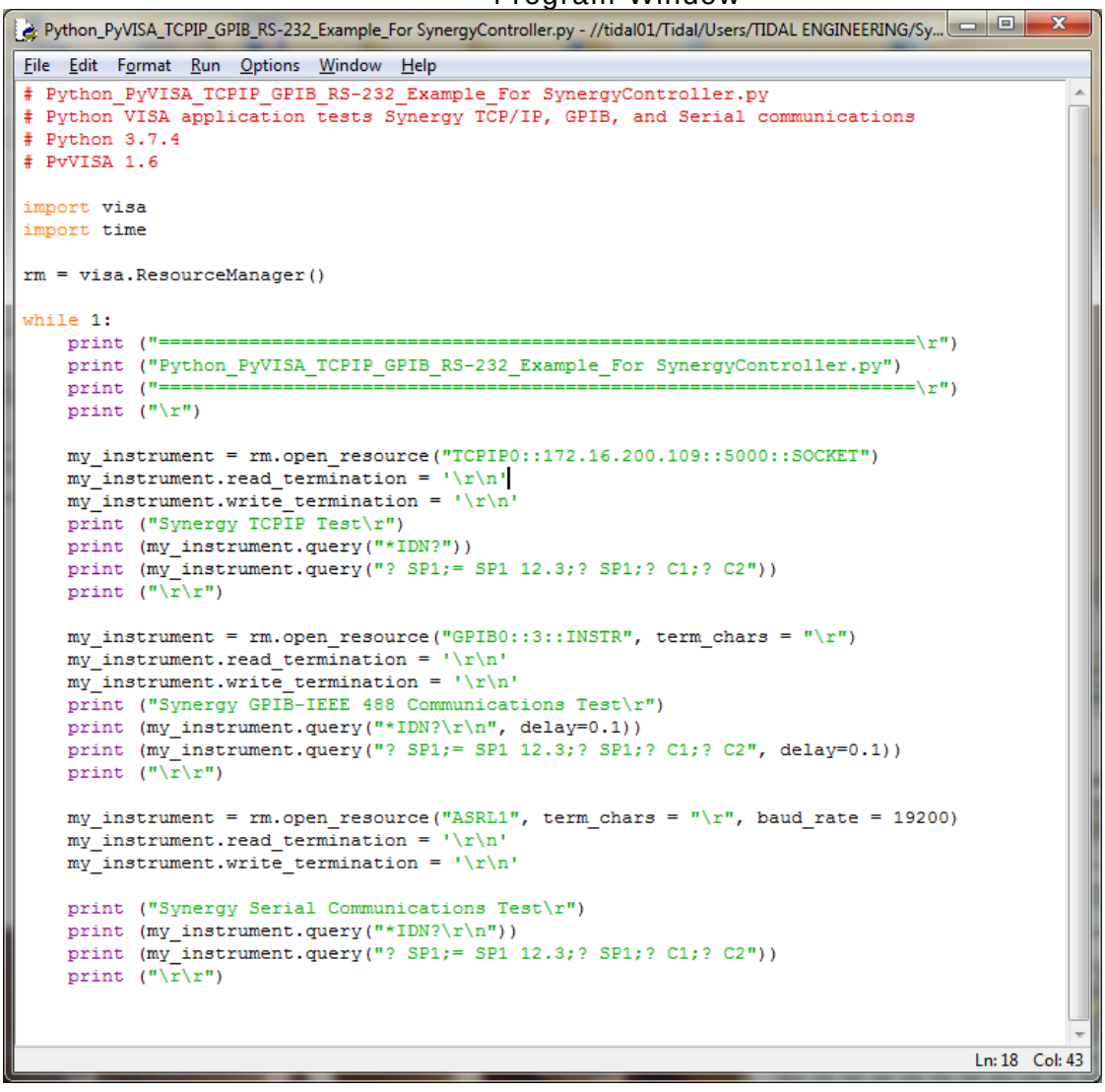
Example 2: Synergy Controller Programming with Python and VISA (PyVISA)

The second example shown on the following page shows off the Python VISA library and controls the three communications ports on the Synergy Controller. In 14 lines (excluding print statements), the program establishes communications with the controller's three ports sequentially, queries the *IDN? Message, the Channel 1 and Channel 2 Process Variables for each and displays the controller's response.

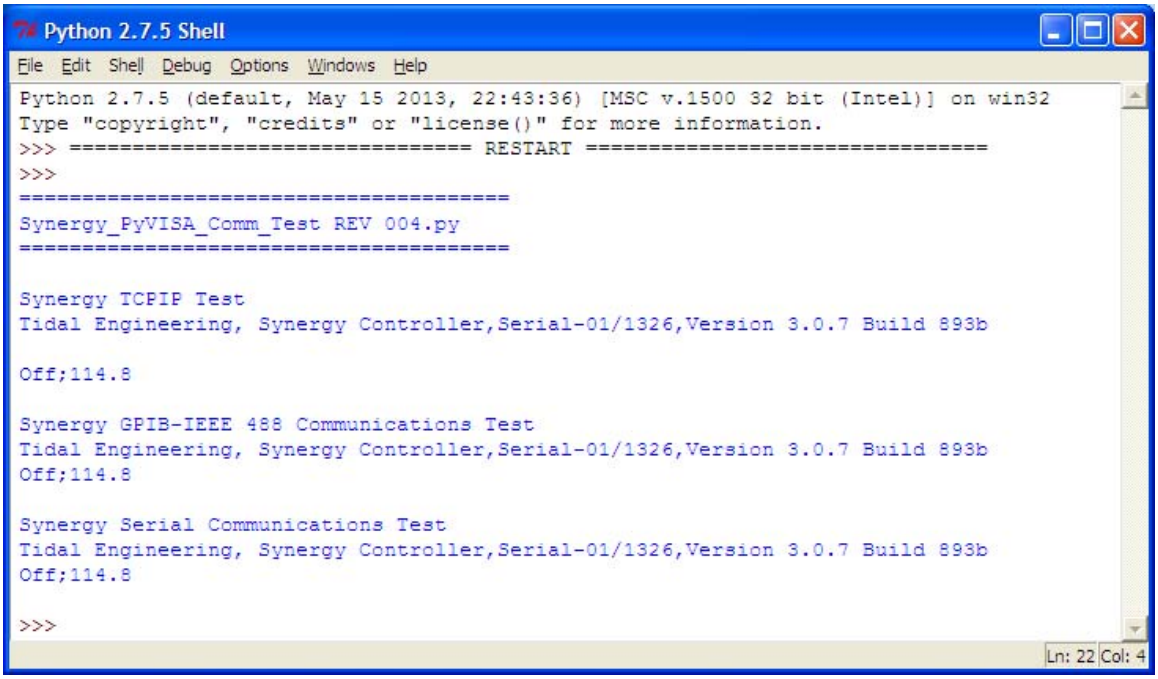
VISA communications for each port are setup prior to running the application. The Screenshots provided in Appendix D show how this is done with National Instruments Measurement and Automation Explorer (NI-MAX).

The source code for this example is included below the screenshots so that it can be quickly cut and pasted into the reader's program.

Program Window



Example 2: Output Window



Example 2: Source Code

```
# Python_PyVISA_TCPIP_GPIB_RS-232_Example_For SynergyController.py
# Python VISA application tests Synergy TCP/IP, GPIB, and Serial communications
# Python 3.7.4
# PvVISA 1.6

import visa
import time

rm = visa.ResourceManager()

while 1:
    print ("=====================================================================\r")
    print ("Python_PyVISA_TCPIP_GPIB_RS-232_Example_For SynergyController.py")
    print ("=====================================================================\r")
    print ("\r")

    my_instrument = rm.open_resource("TCPIP0::172.16.200.109::5000::SOCKET")
    my_instrument.read_termination = '\r\n'
    my_instrument.write_termination = '\r\n'
    print ("Synergy TCPIP Test\r")
    print (my_instrument.query("*IDN?"))
    print (my_instrument.query("? SP1;= SP1 12.3;? SP1;? C1;? C2"))
    print ("\r\r")

    my_instrument = rm.open_resource("GPIB0::3::INSTR", term_chars = "\r")
    my_instrument.read_termination = '\r\n'
    my_instrument.write_termination = '\r\n'
    print ("Synergy GPIB-IEEE 488 Communications Test\r")
    print (my_instrument.query("*IDN?\r\n", delay=0.1))
    print (my_instrument.query("? SP1;= SP1 12.3;? SP1;? C1;? C2", delay=0.1))
    print ("\r\r")

    my_instrument = rm.open_resource("ASRL1", term_chars = "\r", baud_rate = 19200)
    my_instrument.read_termination = '\r\n'
    my_instrument.write_termination = '\r\n'

    print ("Synergy Serial Communications Test\r")
    print (my_instrument.query("*IDN?\r\n"))
    print (my_instrument.query("? SP1;= SP1 12.3;? SP1;? C1;? C2"))
    print ("\r\r")
```

Example 3: Synergy Controller Programming with Python and ModbusTCP

Program Window

Python_ModbusTCP_Example_for_Synergy_Controller_Rev_A.py - \\tidal01\Tidal\Users\TIDAL ENGINEERING\Synergy Controller\Application No...

File Edit Format Run Options Window Help

```
Python_ModbusTCP_Example_for_Synergy_Controller_Rev_A.py
#Python 3.7.4
#Synergy Controller Software Version 5.4.3 Build 1412

from pyModbusTCP.client import ModbusClient
from pyModbusTCP import utils
import time

c = ModbusClient(host='172.16.200.109', port=502, auto_open=True)

while 1:
    time.sleep(0.5)
    print()
    reply = c.read_holding_registers(40899, 6)
    print("Real Time Clock--Address 40899:  ", reply)

    reply = c.read_holding_registers(40800, 4)
    print("Network Address--Address 40800:  ", '{:03}.{:03}.{:03}.{:03}'.format(*reply))

    reply = c.read_holding_registers(40804, 4)
    print("Subnet Mask------Address 40804:  ", '{:03}.{:03}.{:03}.{:03}'.format(*reply))

    reply = c.read_holding_registers(40808, 4)
    print("Gateway----------Address 40808:  ", '{:03}.{:03}.{:03}.{:03}'.format(*reply))

    reply = c.read_holding_registers(40821, 6)
    print("MAC Address------Address 40821:  ", '{:02X}:{:02X}:{:02X}:{:02X}:{:02X}:{:02X}'.format(*reply))

c.close()
```

Ln: 11 Col: 8

Example 3- ModbusTCP: Output Window

C:\Windows\py.exe

```
Real Time Clock--Address 40899:   [8, 13, 12, 9, 1, 2019]
Network Address--Address 40800:   172.016.200.109
Subnet Mask------Address 40804:   255.255.255.000
Gateway----------Address 40808:   172.016.200.254
MAC Address------Address 40821:   00:14:2D:25:1E:EE

Real Time Clock--Address 40899:   [8, 13, 13, 9, 1, 2019]
Network Address--Address 40800:   172.016.200.109
Subnet Mask------Address 40804:   255.255.255.000
Gateway----------Address 40808:   172.016.200.254
MAC Address------Address 40821:   00:14:2D:25:1E:EE

Real Time Clock--Address 40899:   [8, 13, 14, 9, 1, 2019]
Network Address--Address 40800:   172.016.200.109
Subnet Mask------Address 40804:   255.255.255.000
Gateway----------Address 40808:   172.016.200.254
MAC Address------Address 40821:   00:14:2D:25:1E:EE

Real Time Clock--Address 40899:   [8, 13, 14, 9, 1, 2019]
Network Address--Address 40800:   172.016.200.109
Subnet Mask------Address 40804:   255.255.255.000
Gateway----------Address 40808:   172.016.200.254
MAC Address------Address 40821:   00:14:2D:25:1E:EE
```

Example 3 - ModbusTCP: Source Code

```
#Python_ModbusTCP_Example_for_Synergy_Controller_Rev_B.py
#Python 3.7.4
#Synergy Controller Software Version 5.4.3 Build 1414

from pyModbusTCP.client import ModbusClient
from pyModbusTCP import utils
import time

c = ModbusClient(host='172.16.200.109', port=502, auto_open=True)

while 1:
    time.sleep(0.5)
    print()
    reply = c.read_holding_registers(40900, 6)
    print("Real Time Clock--Address 40900:  ", reply)

    reply = c.read_holding_registers(40801, 4)
    print("Network Address--Address 40801:  ", '{:03}.{:03}.{:03}.{:03}'.format(*reply))

    reply = c.read_holding_registers(40805, 4)
    print("Subnet Mask------Address 40805:  ", '{:03}.{:03}.{:03}.{:03}'.format(*reply))

    reply = c.read_holding_registers(40809, 4)
    print("Gateway----------Address 40809:  ", '{:03}.{:03}.{:03}.{:03}'.format(*reply))

    reply = c.read_holding_registers(40821, 6)
    print("MAC Address------Address 40821:  ",
'{:02X}:{:02X}:{:02X}:{:02X}:{:02X}:{:02X}'.format(*reply))

c.close()
```

Appendix A: Synergy Controller Frequently Used Commands

Setpoints and Process Var. Commands	**Syntax**	**Example**	**Response**
Query Temperature (Chan.1)	? C1	? C1	25.0
Query Humidity (Chan.2)	? C2	? C2	50.0
Set Temperature Setpoint (Chan.1)	= SP1 n	= SP1 25.0	OK
Query Temperature Setpoint (Chan.1)	? SP1	? SP1	25.0
Set Humidity Setpoint (Chan. Chan.2)	= SP2 n	= SP2 50.0	OK
Query Humidity Setpoint (Channel 2)	? SP2	? SP2	50.0
On/Off Commands	**Syntax**	**Example**	**Response**
Turn Chamber ON	= ON	= ON	OK
Query Chamber ON state	? ON	? ON	0 or 1
Turn Chamber OFF	= OFF	= OFF	OK
Event Output Commands	**Syntax**	**Example**	**Response**
Set Event Output n ON	= EVENTS n, 1	= EVENTS 1, 1	OK
Set Event Output n OFF	= EVENTS n, 0	= EVENTS 1, 0	OK
Query State of Event Outputs	? EVENTS	? EVENTS	00FF0003
Program Commands	**Syntax**	**Example**	**Response**
Query Program State	? RUN PROFILE_STATE		
Load a Program	= FILEOPEN 1 “program-name”	= FILEOPEN 1 "Product1"	OK
Start a Program	= RUN	= RUN	OK
Start a Program at a specific line	= RUNFROM n	= RUNFROM 2	OK
Query Program state	? RUN 1= RUN, 2 = PAUSE, 3 = Steady State	? RUN	1
Alarm Commands	**Syntax**	**Example**	**Response**
Check Status of Alarms	? ALM	? ALM	OK
Acknowledge Alarms	= ACKALM	= ACKALM	OK

Notes:

1. Terminate commands and queries with Carriage Return (cr) and Line Feed (lf) (\r\n).
2. Visit www.TidalEng.com\synergy.htm for the complete command set.

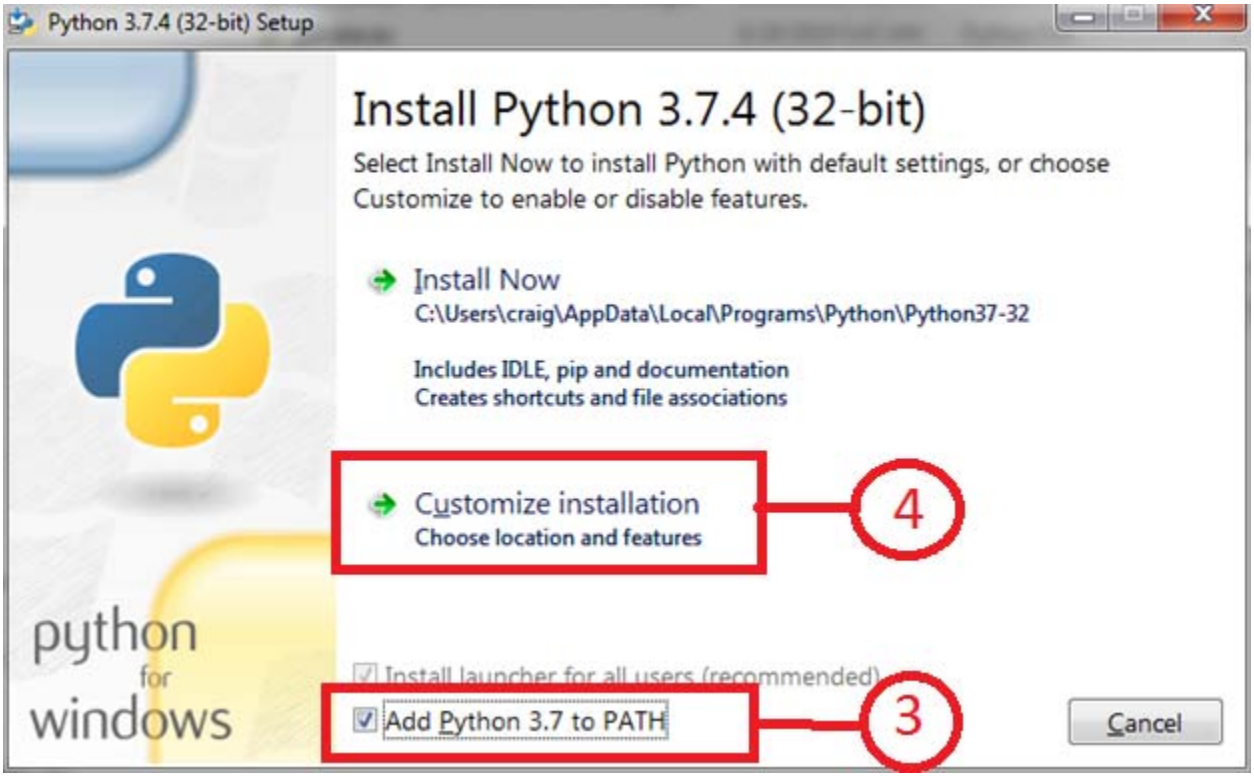
Appendix B: **Python Setup**

Python is an Open Source and freely available programming language that can be using for almost any task. http://www.python.org/about/

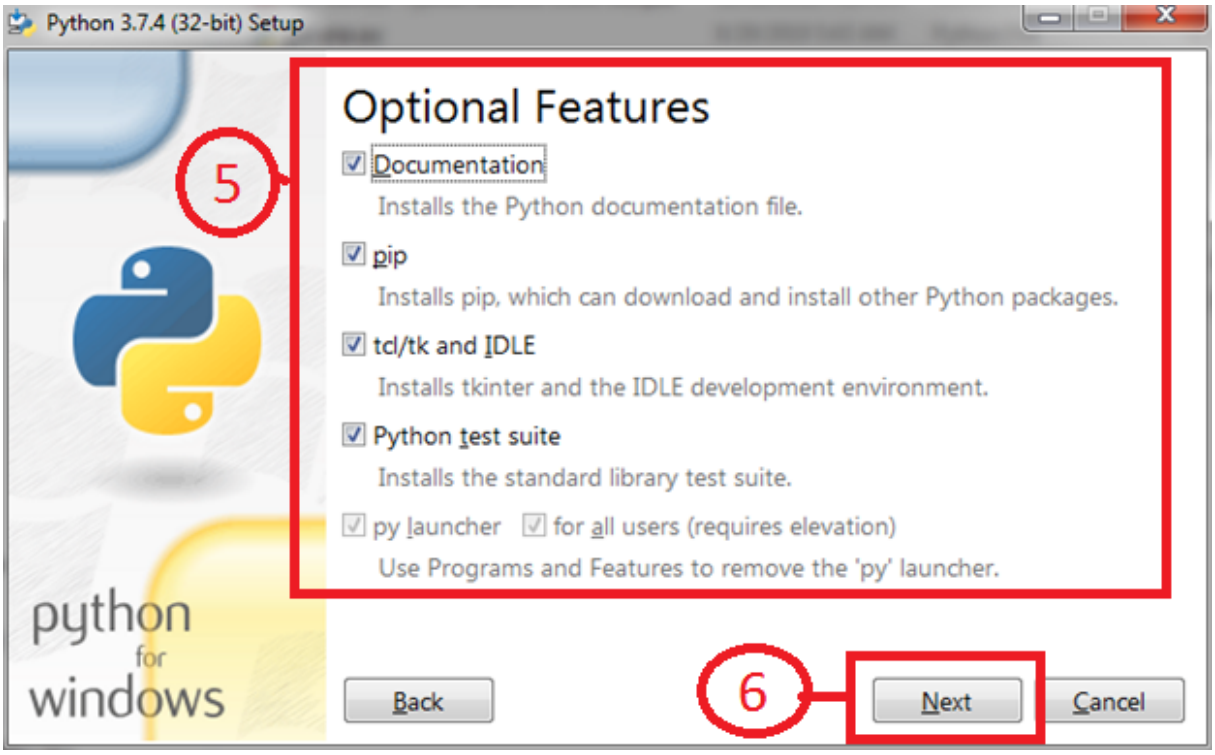
For this demonstration, download the Python Software for Windows https://www.python.org/:

Or click this link to download 3.7.4 for Windows: https://www.python.org/downloads/

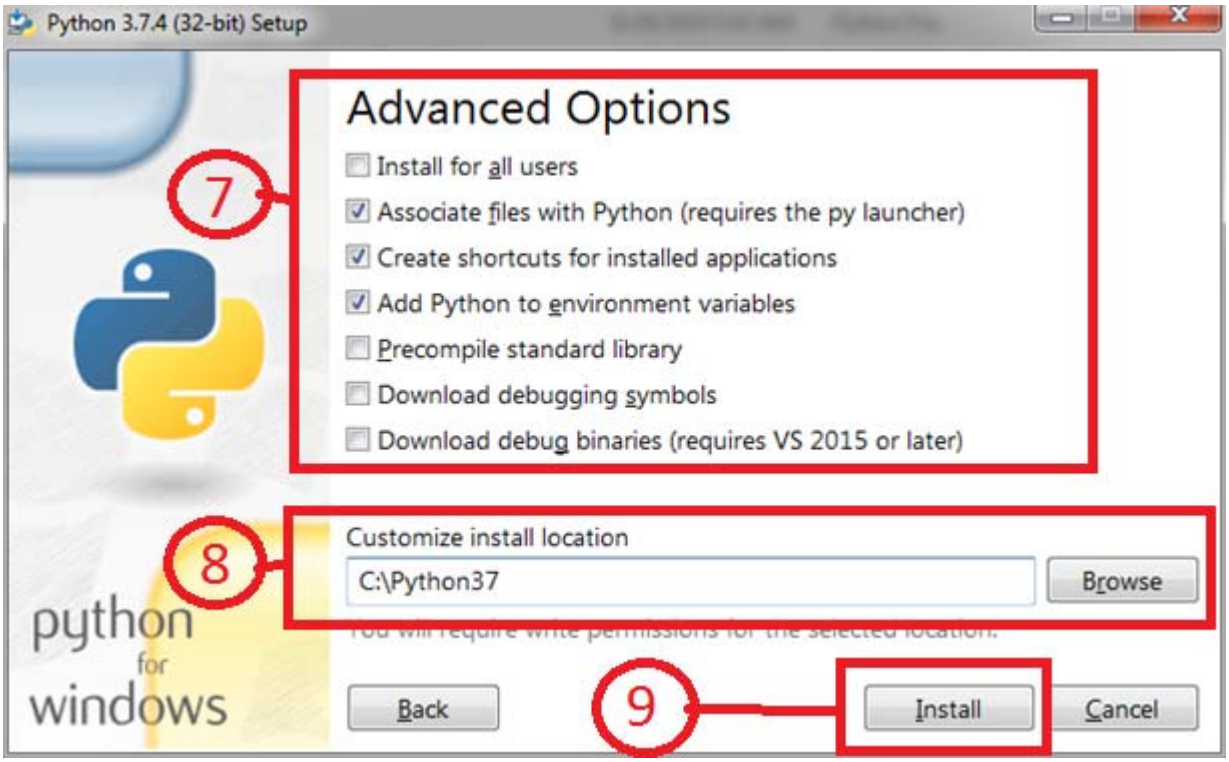
1. Save the Installer, python-3.7.4.exe to your desktop
2. Click on the installer to start it.
3. Check the box “Add Python 3.7 to PATH”.
4. Click “Customize Installation”.



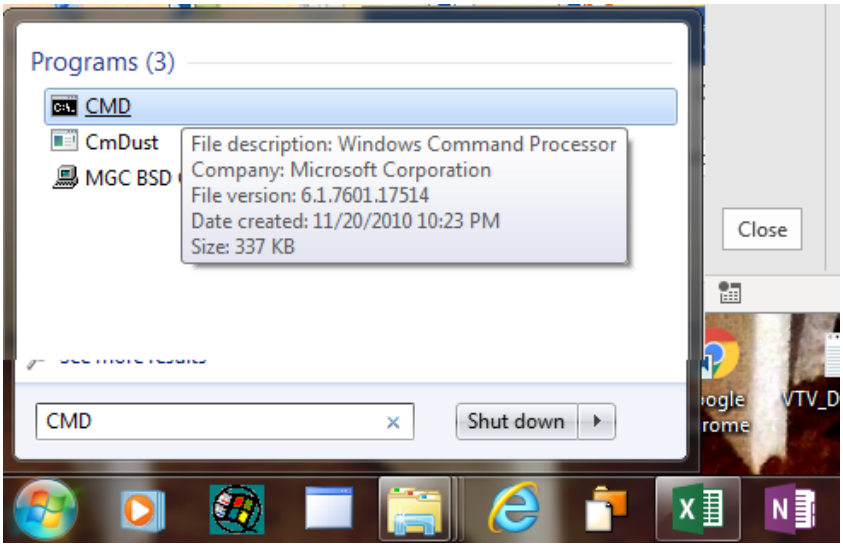
5. Select the options below.
6. Click “Next”.

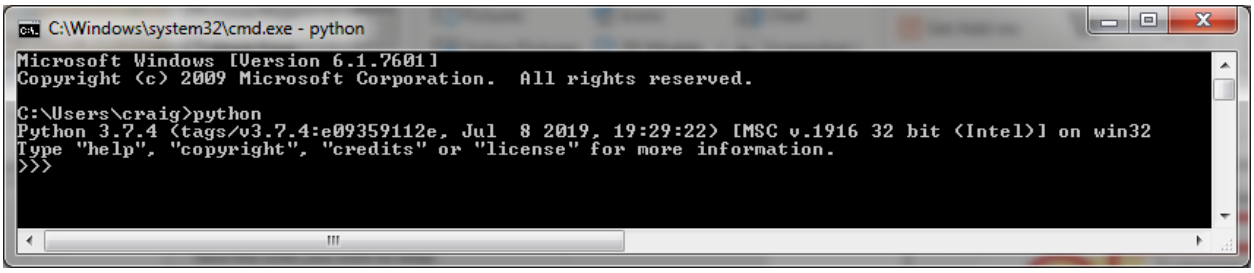


7. Select the Advanced options below.
8. Type C:\Python37” to customize the install location.
9. Click “Install”



10. To Test Installation open a Command prompt and Type Python as shown below
    Look for the Python Version in the response.



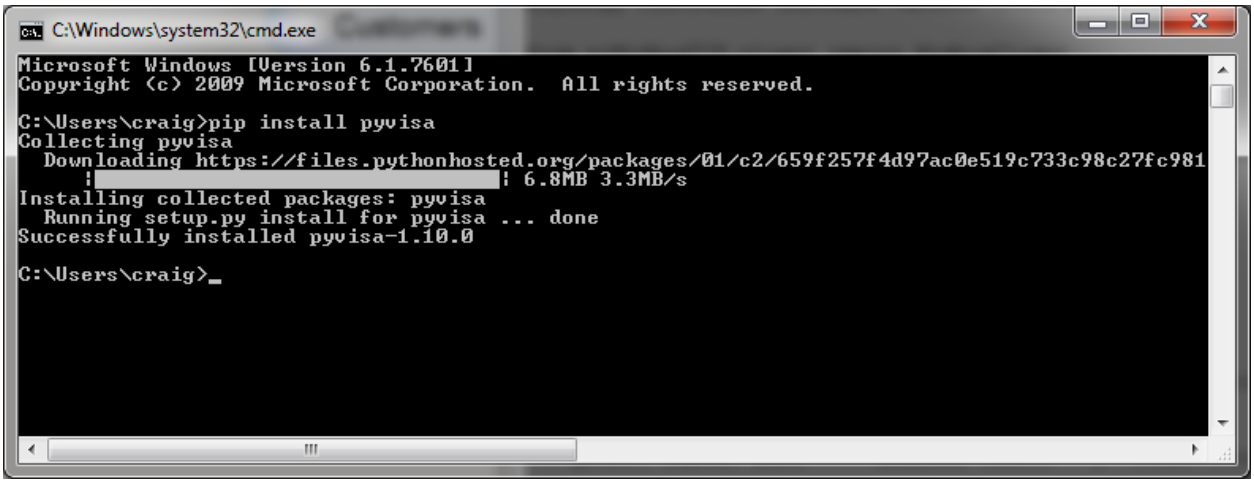


Appendix C: **PyVISA Setup**

The PyVISA package augments the Python language and equips it to easily control virtually any test and measurement device through variety of busses including; GPIB, RS232, TCP/IP and USB.

First, Install Python as shown in Appendix B.

Then, open a command prompt and type “pip install PyVISA” as shown below.
(Note that a network connection is required for this installation)

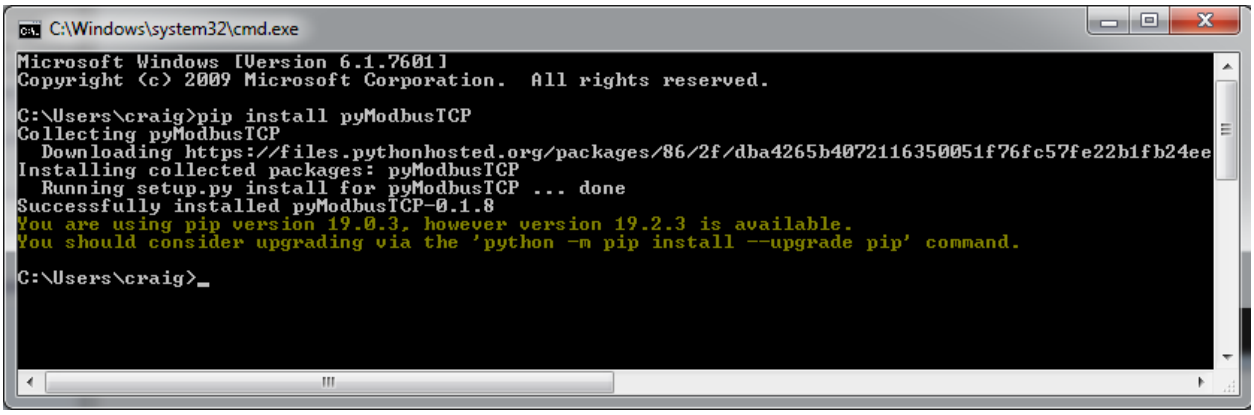


Appendix D: **PyModbusTCP Setup**

The PyModbusTCP package augments the Python language ModbusTCP features.

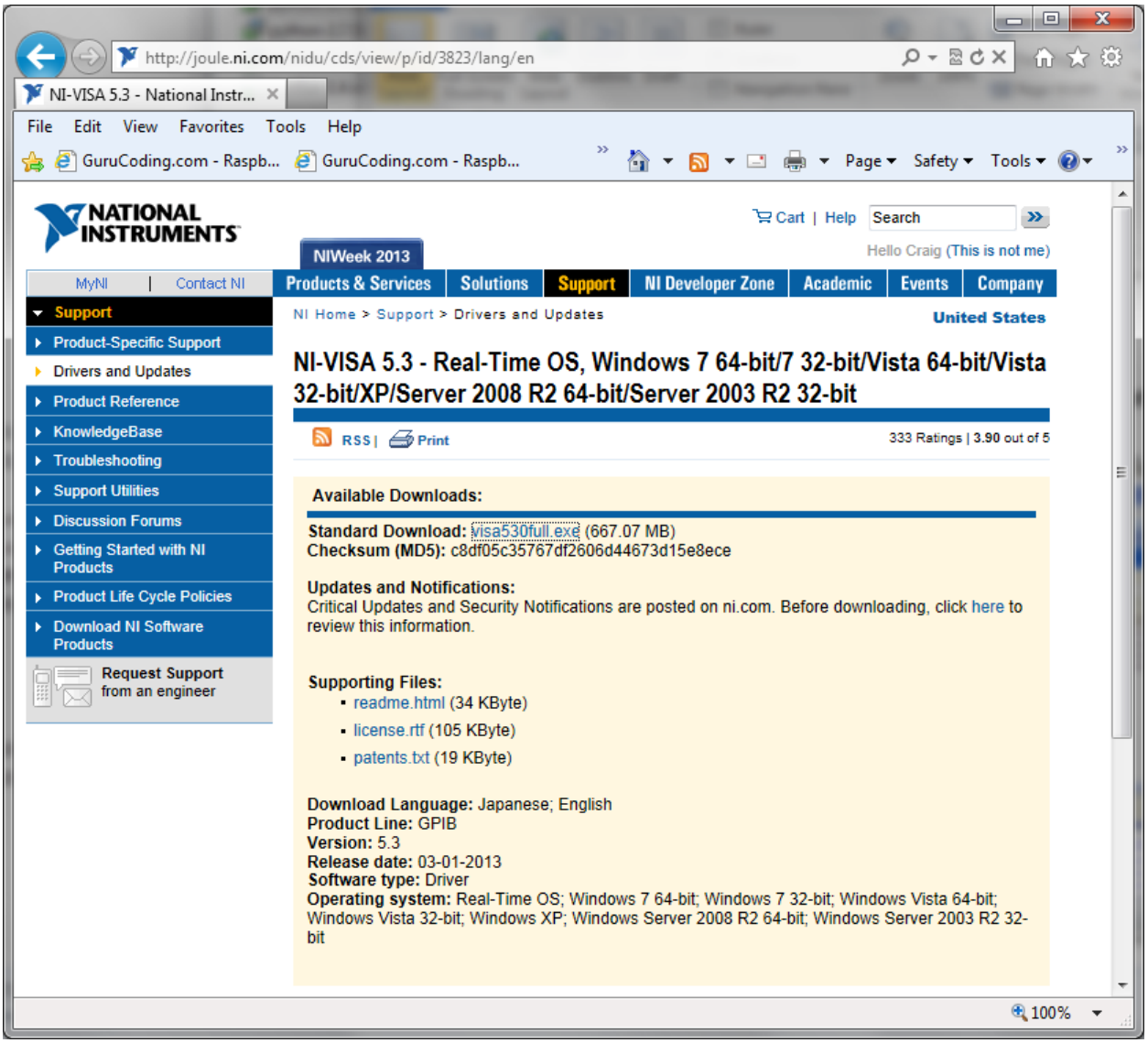
First, Install Python as shown in Appendix B.

Then, open a command prompt and type “pip install pyModbusTCP” as shown below.
(Note that a network connection is required for this installation)



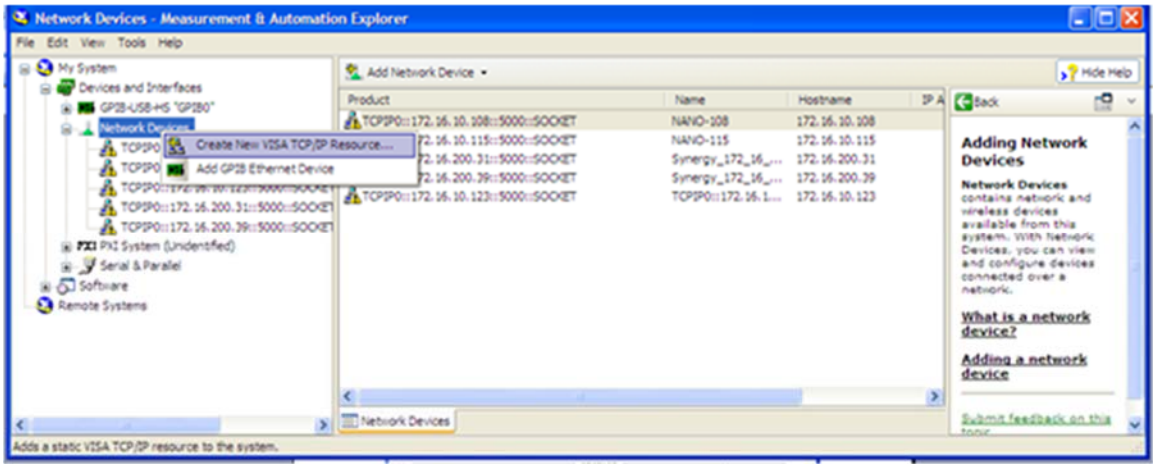
Appendix E: **NI Measurement and Automation Explorer (NIMAX) Setup**

The NI VISA download page is here: http://joule.ni.com/nidu/cds/view/p/id/3823/lang/en

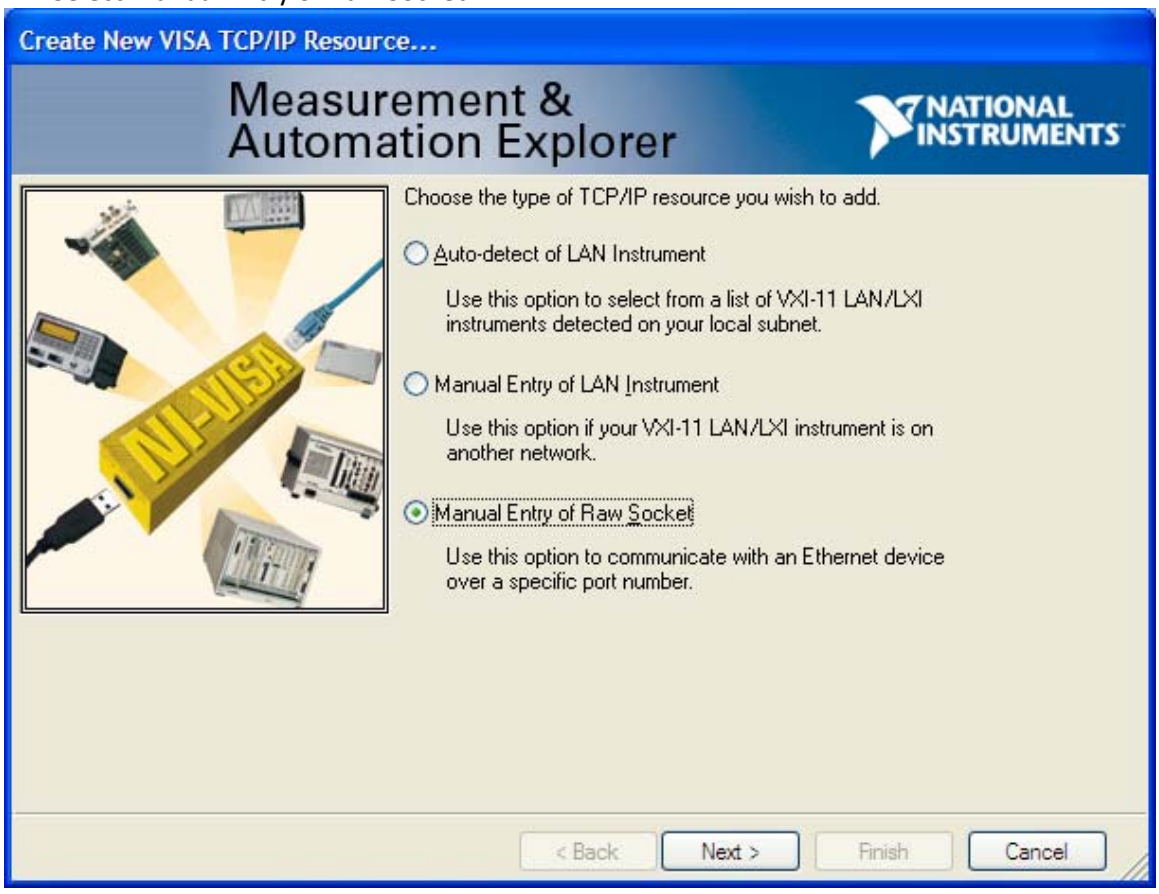


NI Measurement and Automation Explorer (NIMAX) Setup for TCP/IP

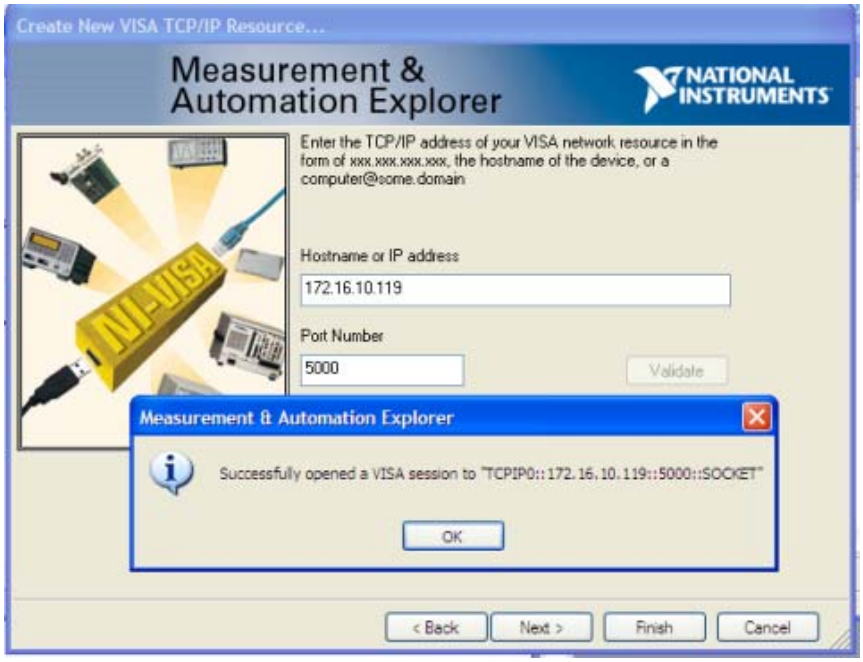
1. Right-Click on the **Network Devices** and select Create a new Network Device



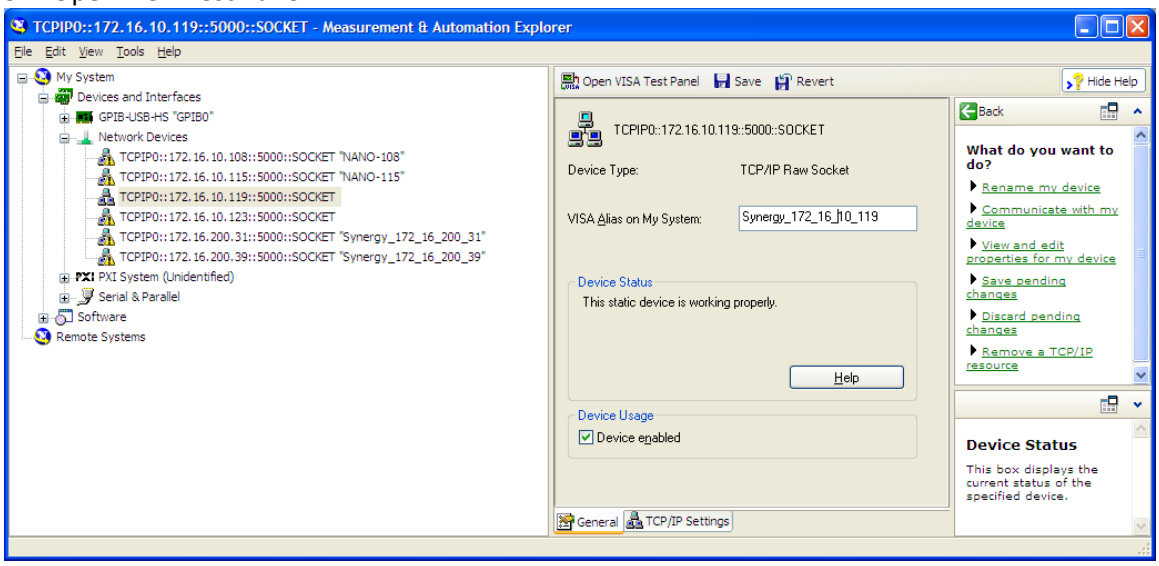
2. Select Manual Entry or Raw Socket



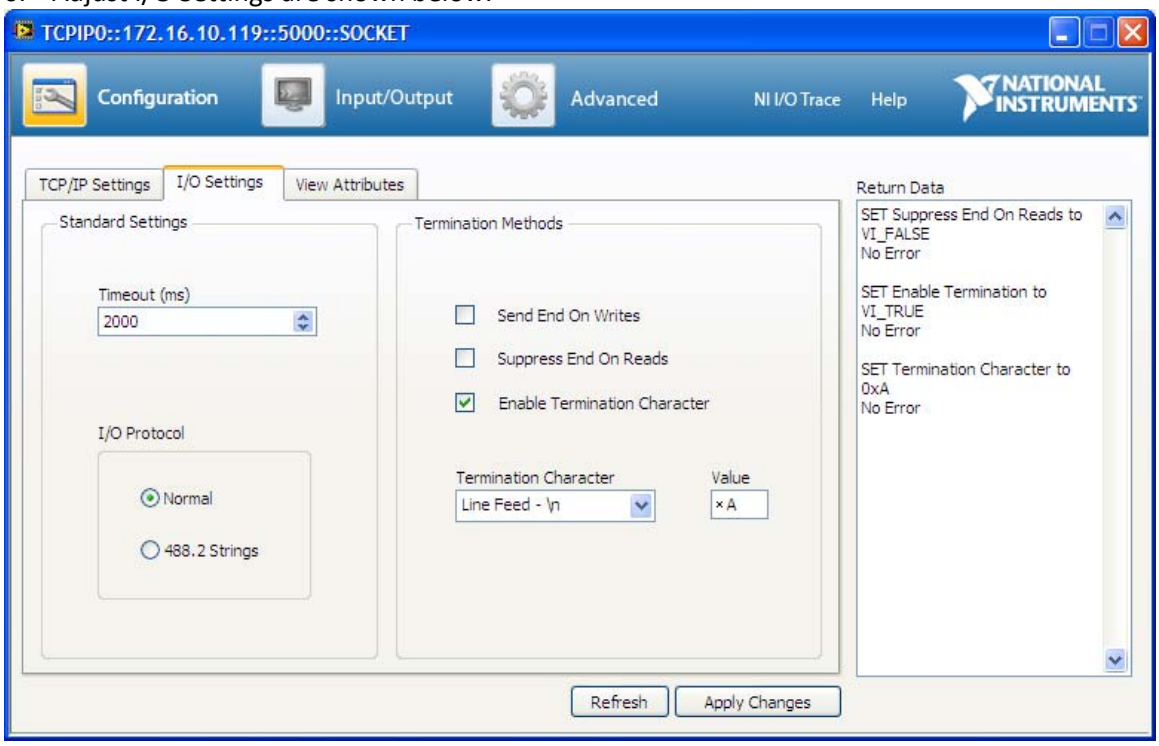
3. Enter the IP address, Port Number 500 and click **Validate.**

4. 

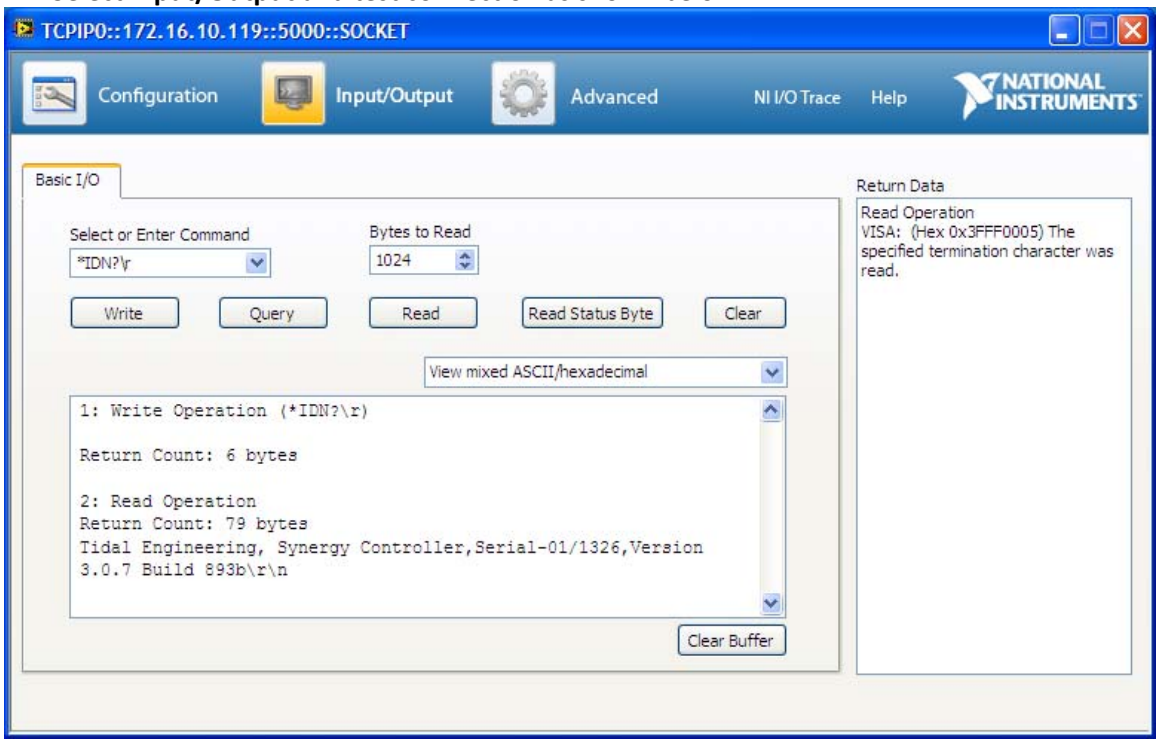
5. Open VISA Test Panel



6. Adjust I/O Settings are shown below.

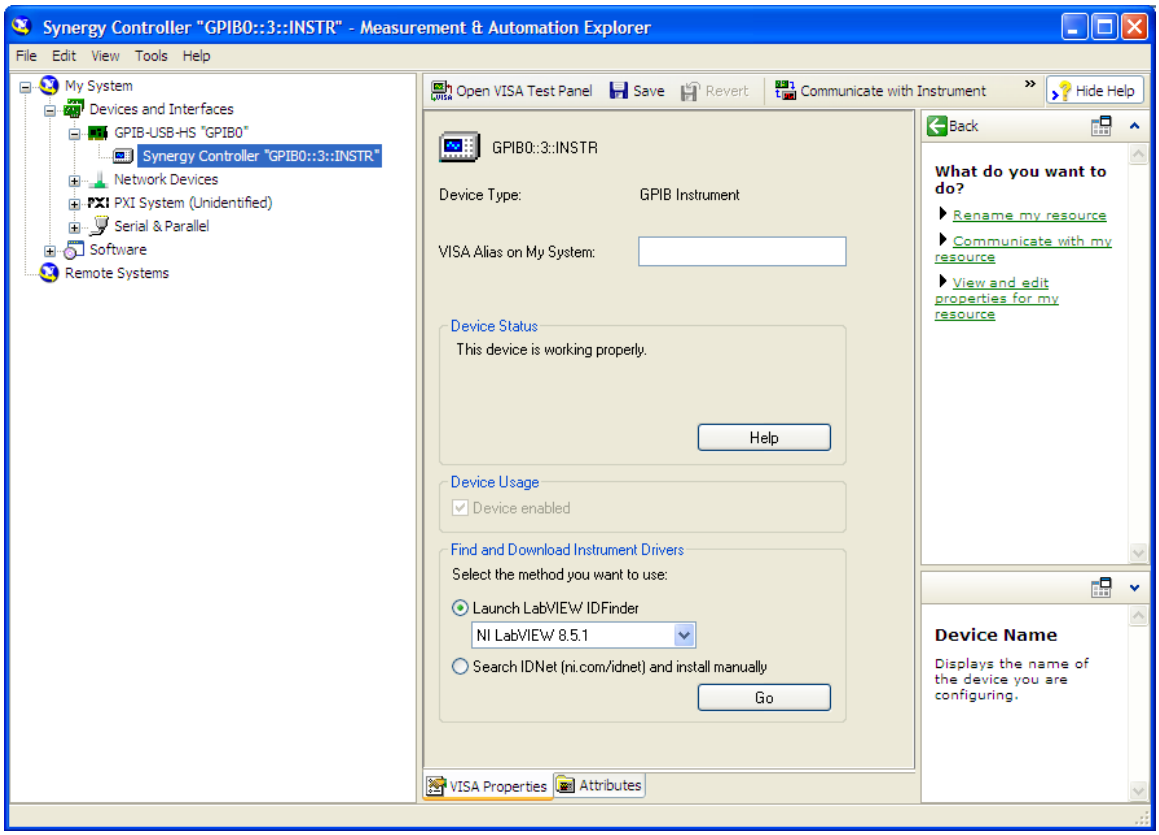


7. Select **Input/Output** and test connection as shown below.

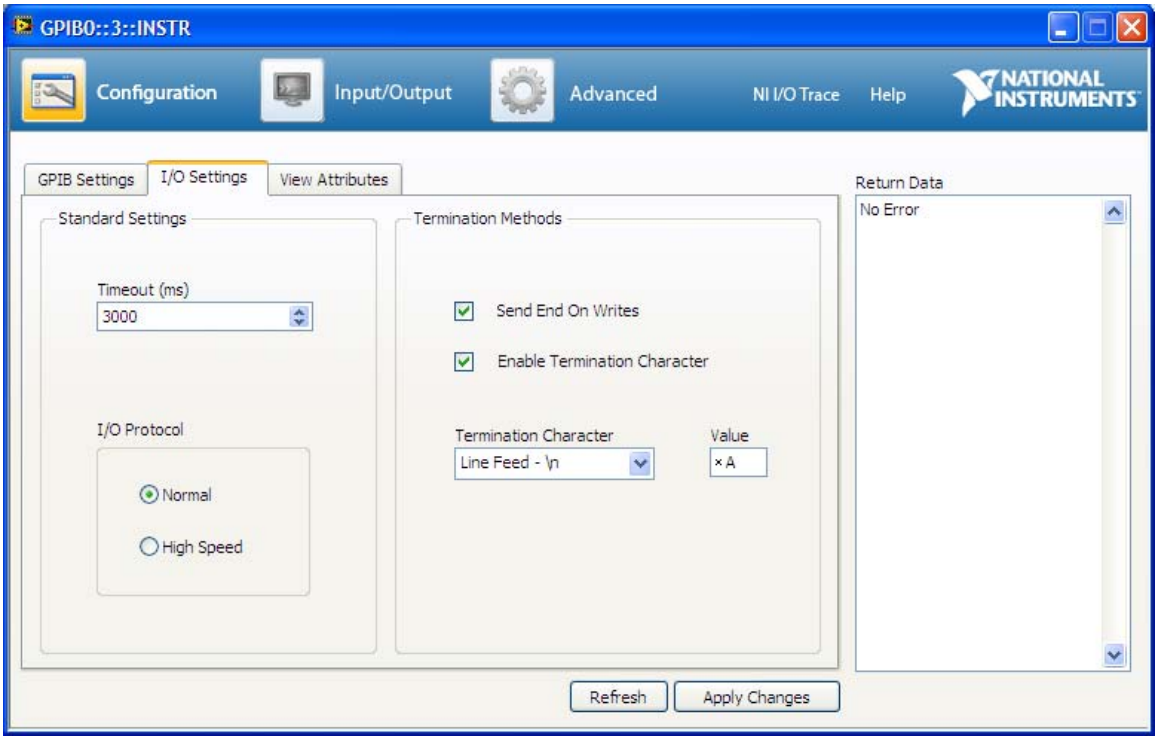


NI Measurement and Automation Explorer (NIMAX) Setup for GPIB

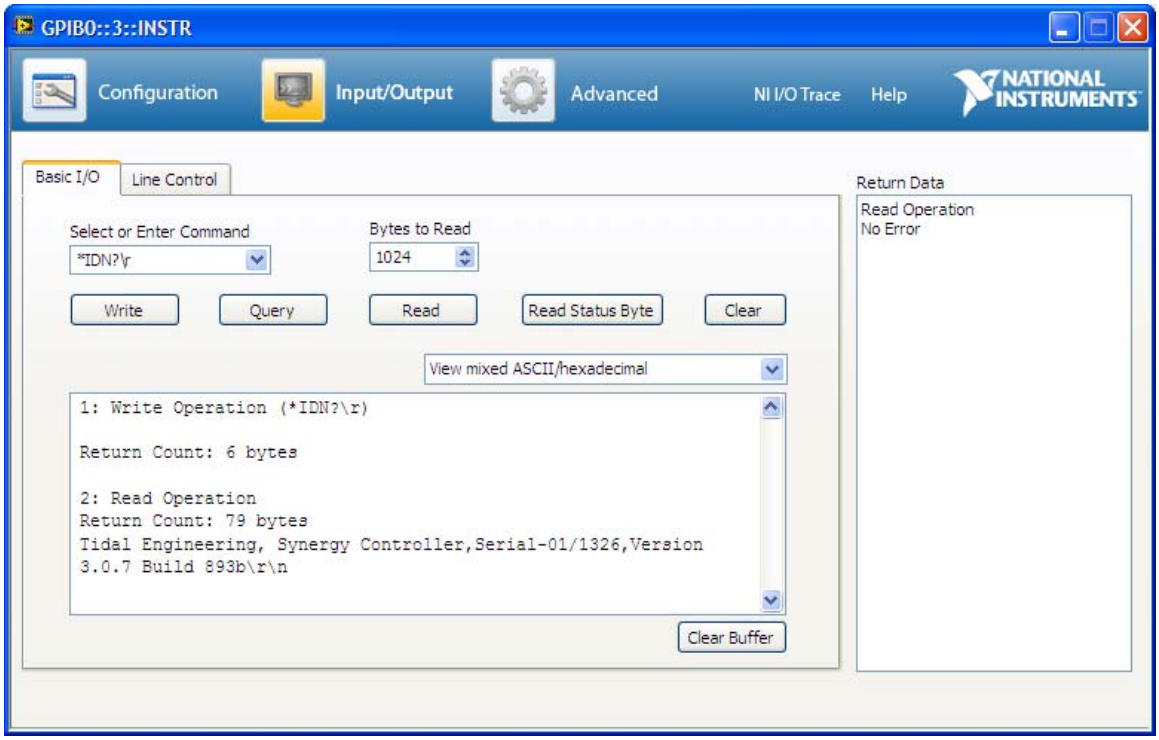
1. Select the GPIB controller in the list with the right-mouse and select **Scan for Instruments** from the menu.



2. Setup the I/O Settings as follows:

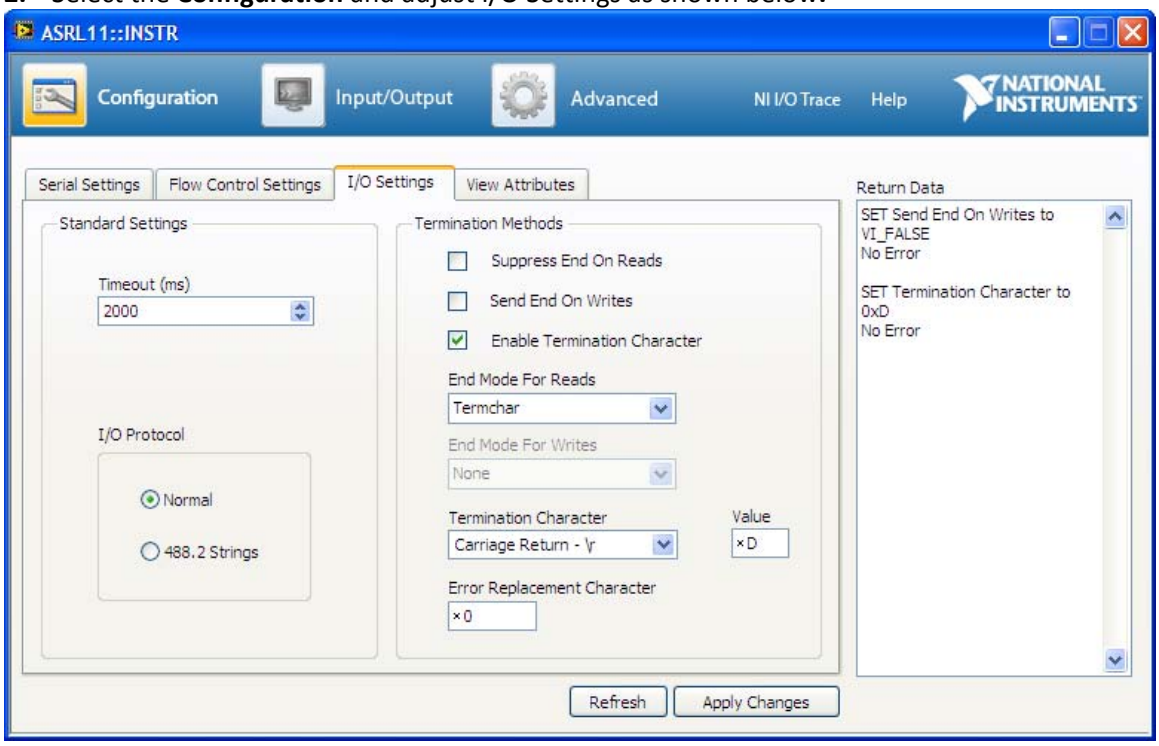


3. Select **Input/Output** and test the connection as follows:

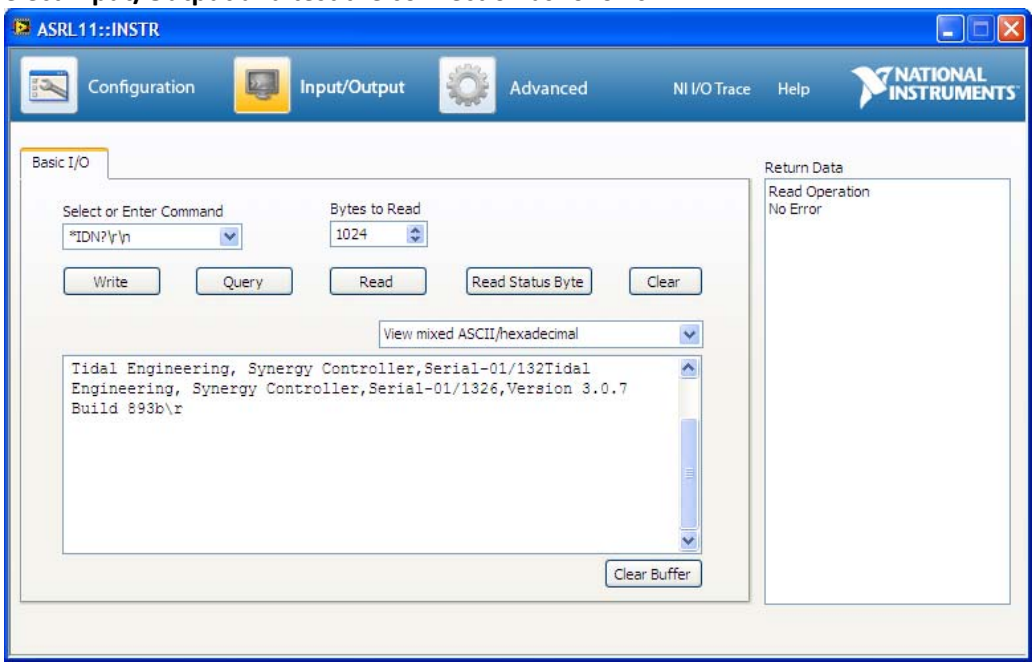


NI Measurement and Automation Explorer (NIMAX) Setup for RS-232

1. Select the appropriate COM port from the **Serial & Parallel** devices list and click **Open VISA Test Panel.**
2. Select the **Configuration** and adjust I/O Settings as shown below**.**



3. Select **Input/Output** and test the connection as follows:



About the Synergy Family
Tidal Engineering's Synergy Controllers, the ¼ DIN Synergy Nano, Synergy Micro 2 and the Synergy Quattro provide state-of-the-art usability and connectivity for environmental test control and data acquisition. They combine the functions of a chamber controller and a data logger and are designed to improve test efficiency by supporting both factory automation and test and measurement protocols and standards.

Synergy Controller feature highlights includes:

- ➔ Color touch screen
- ➔ Ethernet, RS-232 and GPIB communications
- ➔ Built in 100 MB Data logger with USB drive support
- ➔ Data Acquisition, up to 64 T-type thermocouples (Optional)
- ➔ Built-in Web Server for remote control; WebTouch Remote ™
- ➔ Compatible with Synergy Manager for PC based control, monitoring and programming.
- ➔ Built-in FTP Server for factory automation and test and measurement applications

For more information regarding these controllers please see the full Synergy Controller Technical Manual on our website at http://www.tidaleng.com/synergy.htm

About Tidal Engineering
Headquartered in Randolph, NJ, Tidal Engineering Corporation has been designing and building award-winning embedded hardware and software for test and measurement and data acquisition applications since 1992. The company is recognized for technical expertise in such areas as Embedded IEEE 488, and turnkey SCADA (Supervisory Control and Data Acquisition) systems.

Tidal Engineering Corporation
2 Emery Avenue
Randolph, NJ 07869
Tel: 973.328.1173
www.TidalEng.com
info@tidaleng.com