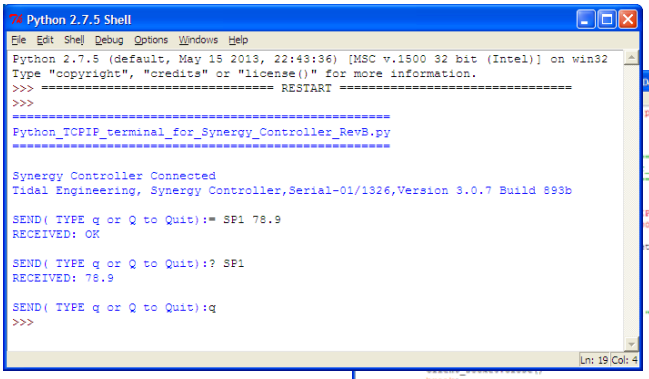
Synergy Controllers and the Python Programing Language



Introduction

This application note demonstrates the versatility of the various communication ports available on the Synergy Controller family; i.e. Ethernet, GPIB, and RS-232 and the simplicity of the Python programming language.

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. Two Python example applications are provided.

1. The first example demonstrates a simple Ethernet TCP/IP control example using Python and Sockets.

2. The second example using Python with VISA (PyVISA) demonstrates simultaneous Synergy Controller communications using Ethernet (TCP/IP), GPIB (IEEE 488), and RS-232.

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 try these example programs, follow these setup steps:

1. Python language setup is described in Appendix A.
2. PyVISA setup is described in Appendix B.
3. VISA setup is described in Appendix C.

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.

Example 1: Synergy Controller Programming with Python and Sockets

Program Window

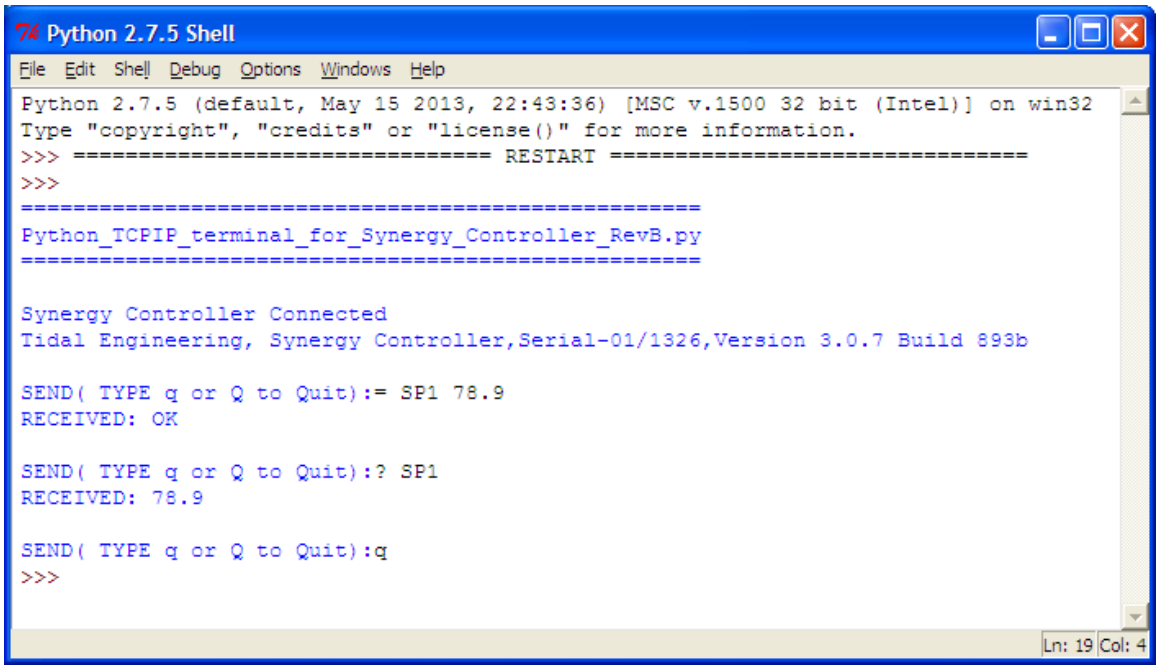
Python_TCPIP_terminal_for_Synergy_Controller_RevB.py - C:\Documents and Settings\Admini...

File Edit Format Run Options Windows Help

```
#Python_TCPIP_terminal_for_Synergy_Controller_RevB.py
#Python 2.7.5
import socket

print '=================================================\r'
print 'Python_TCPIP_terminal_for_Synergy_Controller_RevB.py\r'
print '=================================================\r'
print '\r'

HOST = '172.16.10.119'    # 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))
client_socket.send("*idn?\r\n")
data = client_socket.recv(1024)
print 'Synergy Controller Connected\r\n', data
while 1:
    data = raw_input ( "SEND( TYPE q or Q to Quit):" )
    if (data <> 'Q' and data <> 'q'):
        client_socket.send(data + "\r\n")
    else:
            client_socket.send(data + "\r\n")
            client_socket.close()
            break;
    data = client_socket.recv(512)
    print 'RECEIVED:', data
```

Ln: 27 Col: 16

Output Window



Example 1 Source Code

```
#Python_TCPIP_terminal_for_Synergy_Controller_RevB.py
#Python 2.7.5
import socket

print '==========\r'
print 'Python_TCPIP_terminal_for_Synergy_Controller_RevB.py\r'
print '==========\r'
print '\r'

HOST = '172.16.10.119'   # 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))
client_socket.send("*idn?\r\n")
data = client_socket.recv(1024)
print 'Synergy Controller Connected\r\n', data
while 1:
   data = raw_input ( "SEND( TYPE q or Q to Quit):" )
   if (data <> 'Q' and data <> 'q'):
      client_socket.send(data + "\r\n")
   else:
         client_socket.send(data + "\r\n")
         client_socket.close()
         break;
   data = client_socket.recv(512)
   print 'RECEIVED:', data
```

The second example shown on the following page shows off the power of the Python Programming language in conjunction with the VISA library for controlling the multiple 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 C 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.

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

Program Window

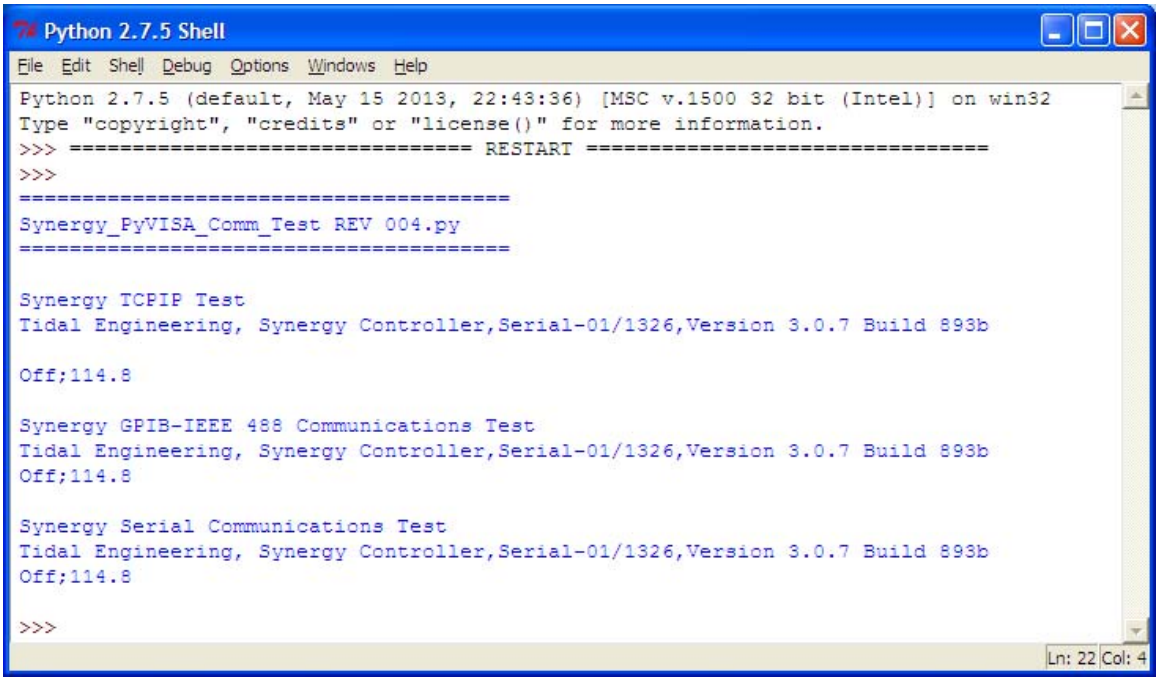
Synergy_PyVISA_Comm_Test REV 004.py - C:/Python27/Synergy_PyVISA_Comm_Test REV ...

File Edit Format Run Options Windows Help

```
# Synergy_PyVISA_Comm_Test REV 004.py
# Python VISA application tests Synergy TCP/IP, GPIB, and Serial communications

from visa import *
import time

while 1:
    print "=======================================\r"
    print "Synergy_PyVISA_Comm_Test REV 004.py"
    print "=======================================\r"
    print "\r"

    my_instrument = instrument("TCPIP0::172.16.10.119::5000::SOCKET", term_chars
    print "Synergy TCPIP Test\r"
    my_instrument.write("*IDN?\r\n")
    print my_instrument.read()
    my_instrument.write("? C1;? C2\r\n")
    print my_instrument.read()
    print "\r\r"

    my_instrument = instrument("GPIB0::3::INSTR", term_chars = "\r")
    print "Synergy GPIB-IEEE 488 Communications Test\r"
    my_instrument.write("*IDN?\r\n")
    time.sleep (50.0 / 1000.0)
    print my_instrument.read()
    my_instrument.write("? C1;? C2\r\n")
    time.sleep (50.0 / 1000.0)
    print my_instrument.read()
    print "\r\r"

    my_instrument = instrument("ASRL1", term_chars = "\r", baud_rate = 19200)
    print "Synergy Serial Communications Test\r"
    my_instrument.write("*IDN?\r")
    print my_instrument.read()
    my_instrument.write("? C1;? C2\r")
    print my_instrument.read()
    print "\r"
```

Ln: 22 Col: 32

Example 2: Output Window



Example 2: Source Code

```
# Synergy_PyVISA_Comm_Test REV 004.py
# Python VISA application tests Synergy TCP/IP, GPIB, and Serial communications

from visa import *
import time

while 1:
    print "======================================\r"
    print "Synergy_PyVISA_Comm_Test REV 004.py"
    print "======================================\r"
    print "\r"

    my_instrument = instrument("TCPIP0::172.16.10.119::5000::SOCKET", term_chars = "\r")
    print "Synergy TCPIP Test\r"
    my_instrument.write("*IDN?\r\n")
    print my_instrument.read()
    my_instrument.write("? C1;? C2\r\n")
    print my_instrument.read()
    print "\r\r"

    my_instrument = instrument("GPIB0::3::INSTR", term_chars = "\r")
    print "Synergy GPIB-IEEE 488 Communications Test\r"
    my_instrument.write("*IDN?\r\n")
    time.sleep (50.0 / 1000.0)
    print my_instrument.read()
    my_instrument.write("? C1;? C2\r\n")
    time.sleep (50.0 / 1000.0)
    print my_instrument.read()
    print "\r\r"

    my_instrument = instrument("ASRL1", term_chars = "\r", baud_rate = 19200)
    print "Synergy Serial Communications Test\r"
    my_instrument.write("*IDN?\r")
    print my_instrument.read()
    my_instrument.write("? C1;? C2\r")
    print my_instrument.read()
    print "\r"
```

Appendix A: **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 here:
http://www.python.org/ftp/python/2.7.5/python-2.7.5.msi

Appendix B: **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.

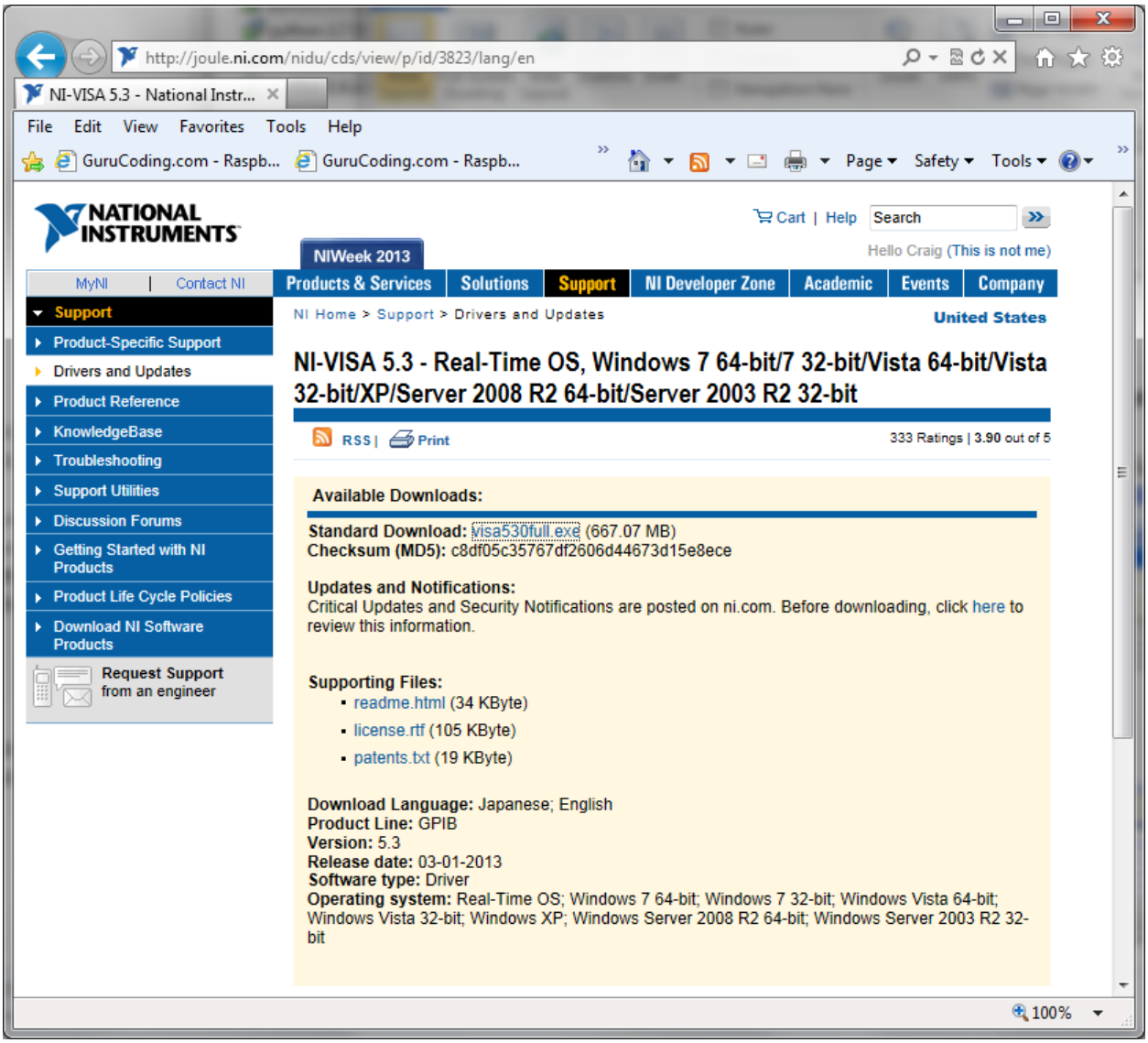
http://pyvisa.sourceforge.net/

Download the Installation File here:

http://sourceforge.net/projects/pyvisa/

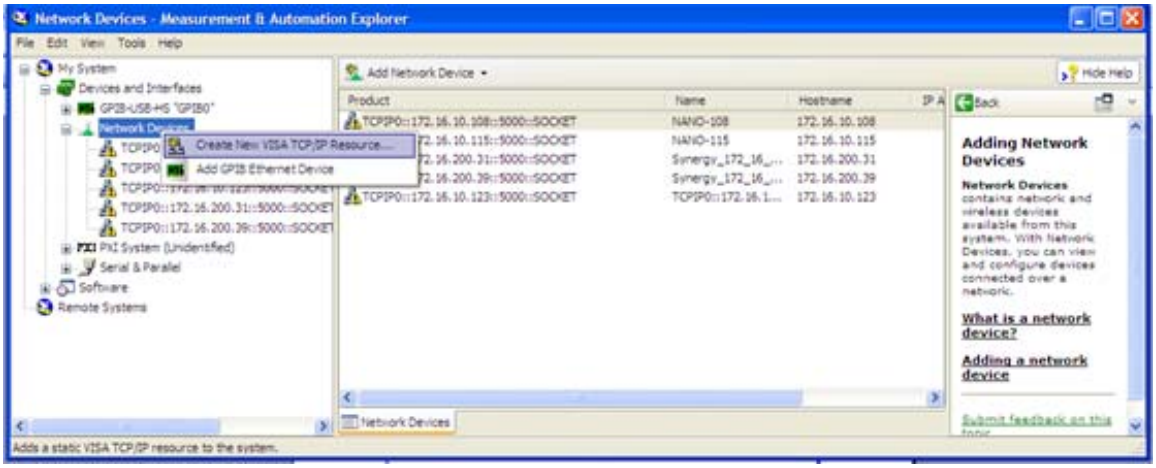
Appendix C: **NI Measurement and Automation Explorer (NIMAX) Setup**

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

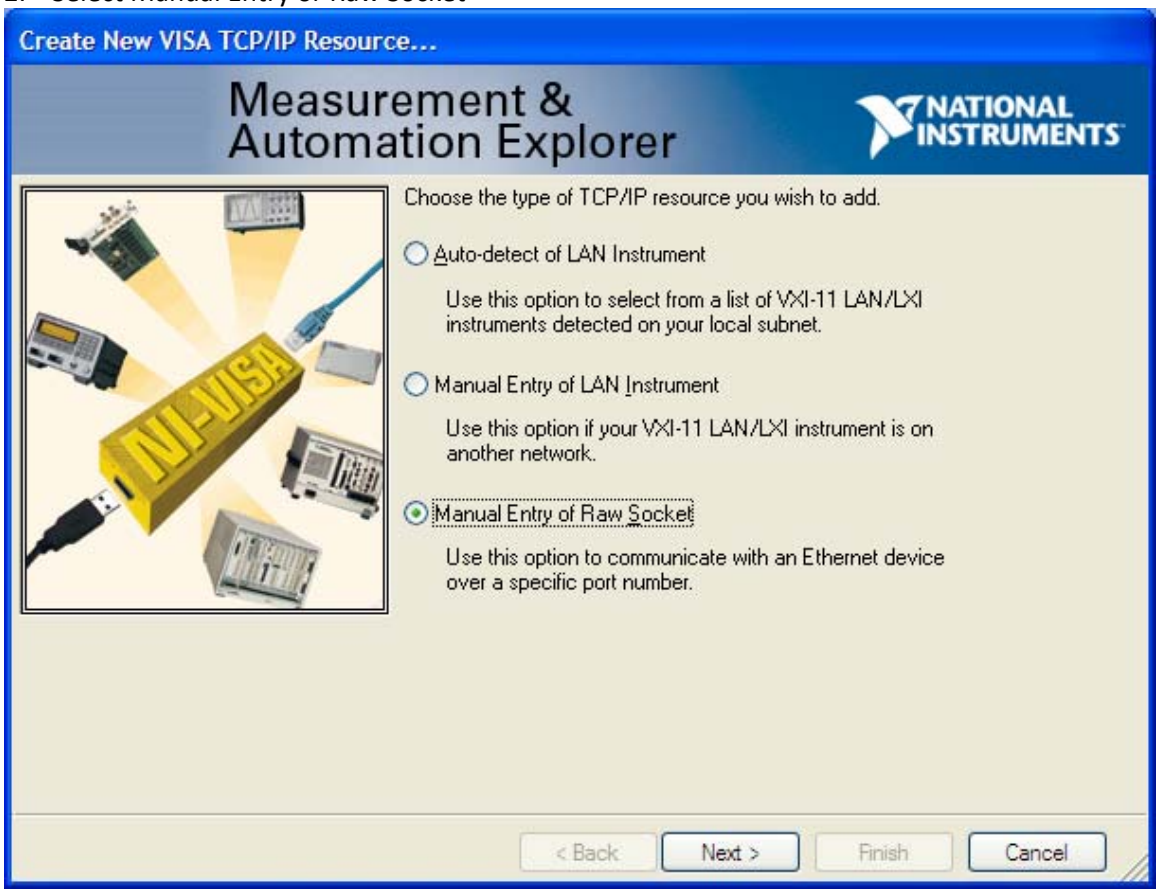


NI Measurement and Automation Explorer (NIMAX) Setup for TCP/IP

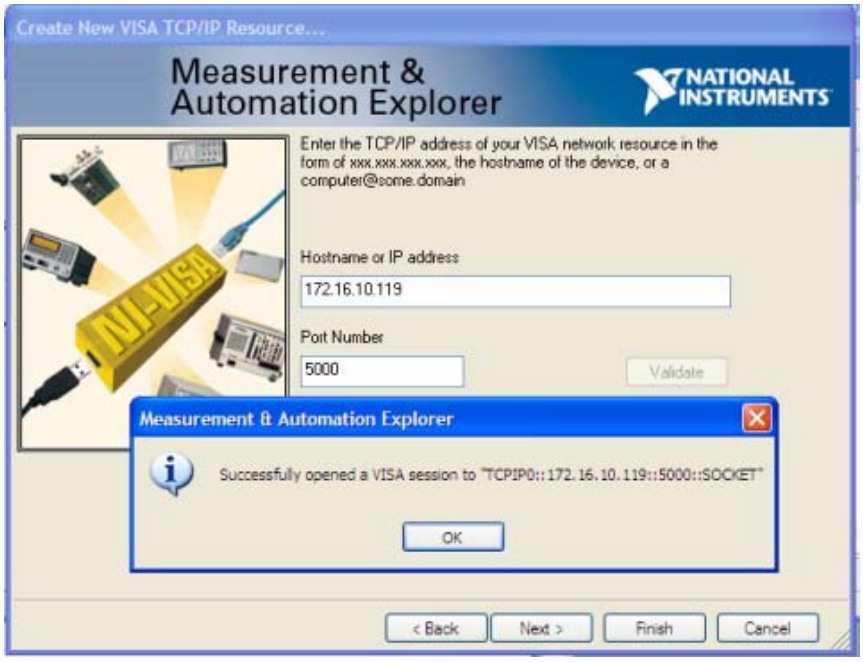
1. Right-Click on the **Network Devices** and select Create a new Network Device



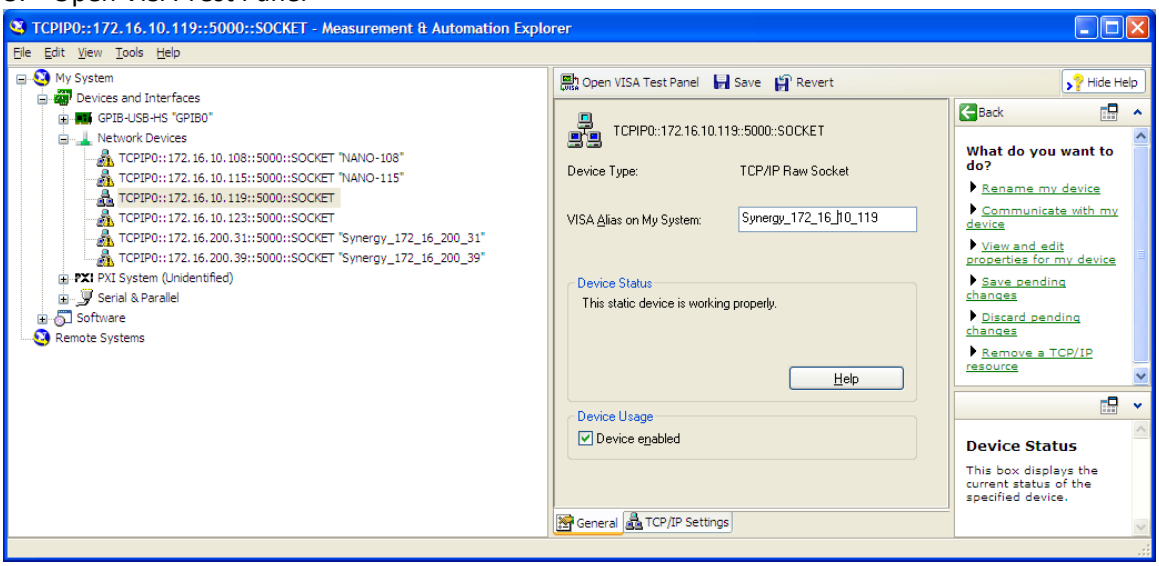
2. Select Manual Entry or Raw Socket



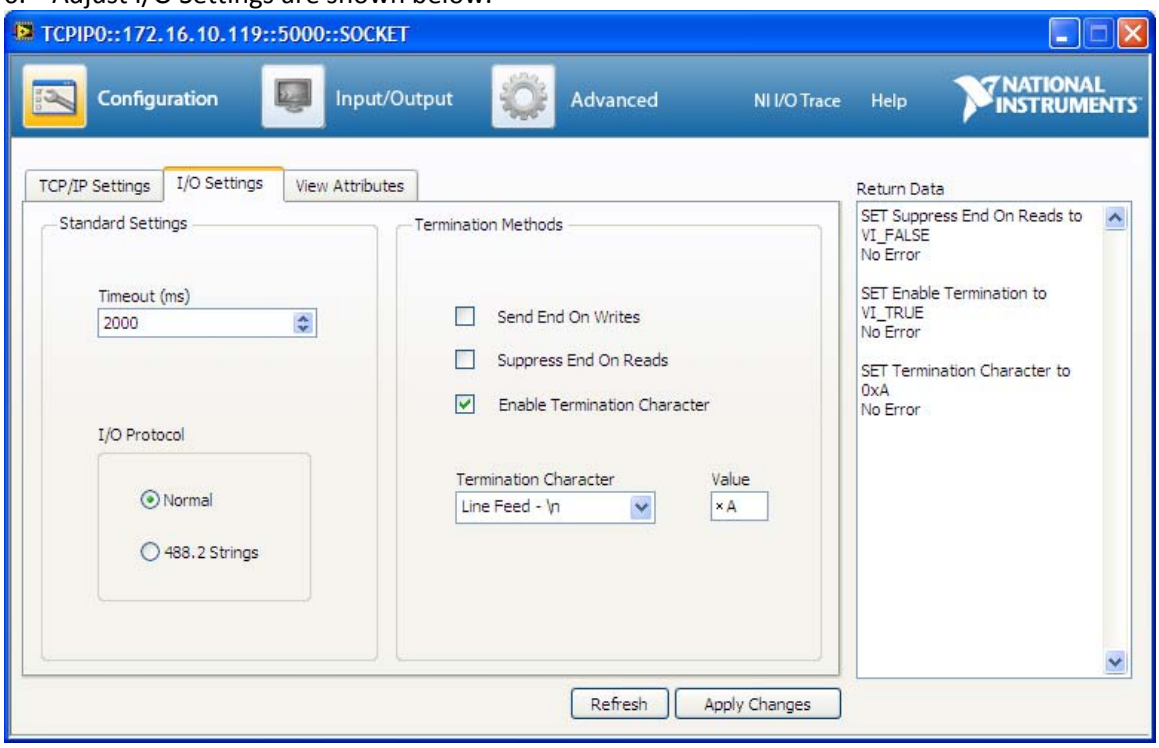
3. Enter the IP address, Port Number 500 and click **Validate.**

4. 

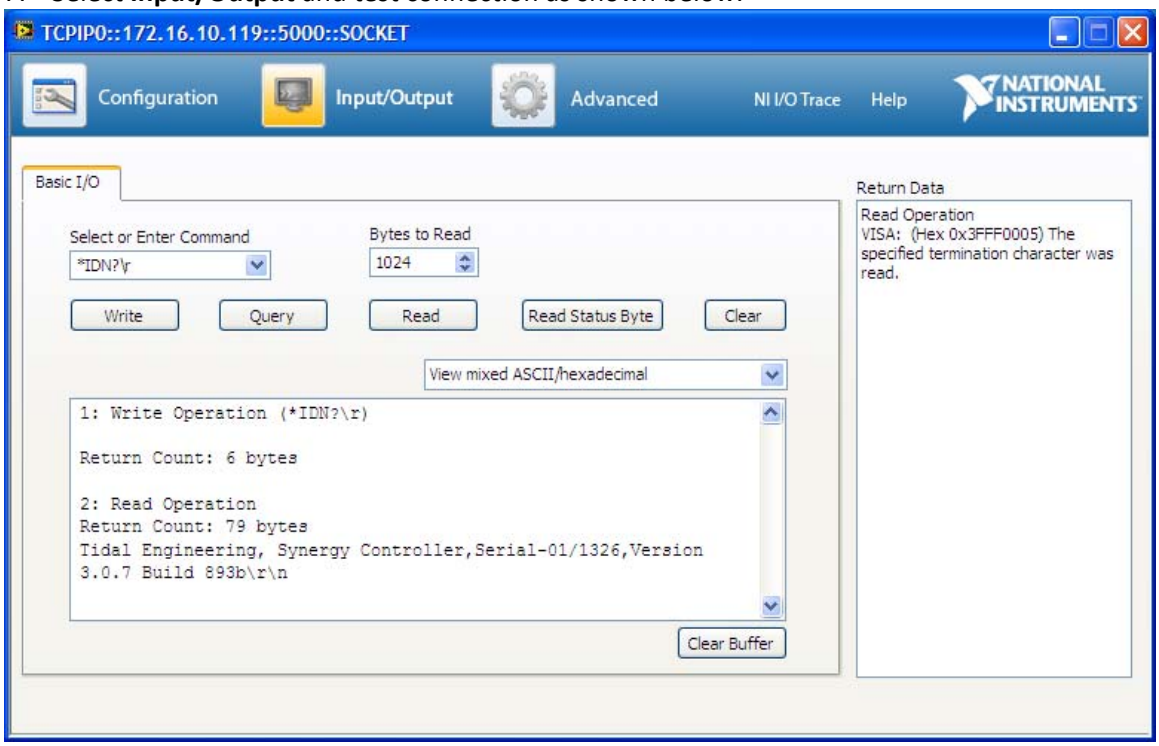
5. Open VISA Test Panel



6. Adjust I/O Settings are shown below.

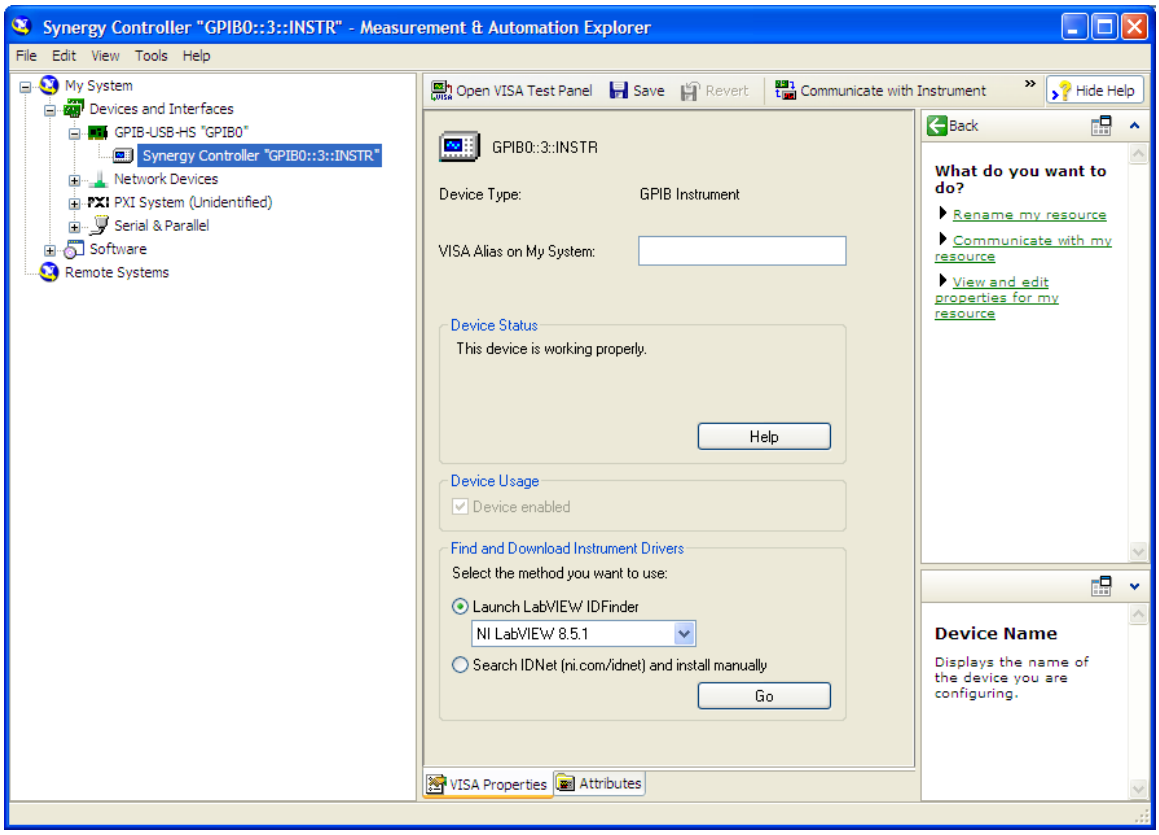


7. Select **Input/Output** and test connection as shown below.

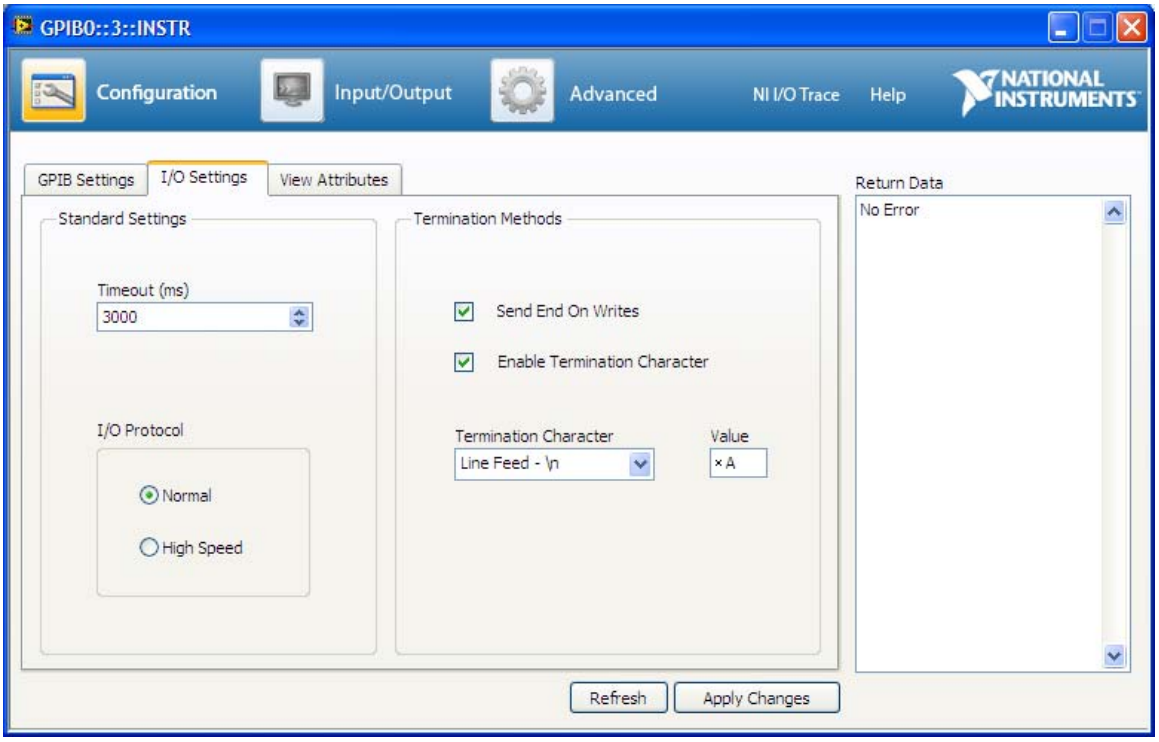


NI Measurement and Automation Explorer (NIMAX) Setup for GPIB

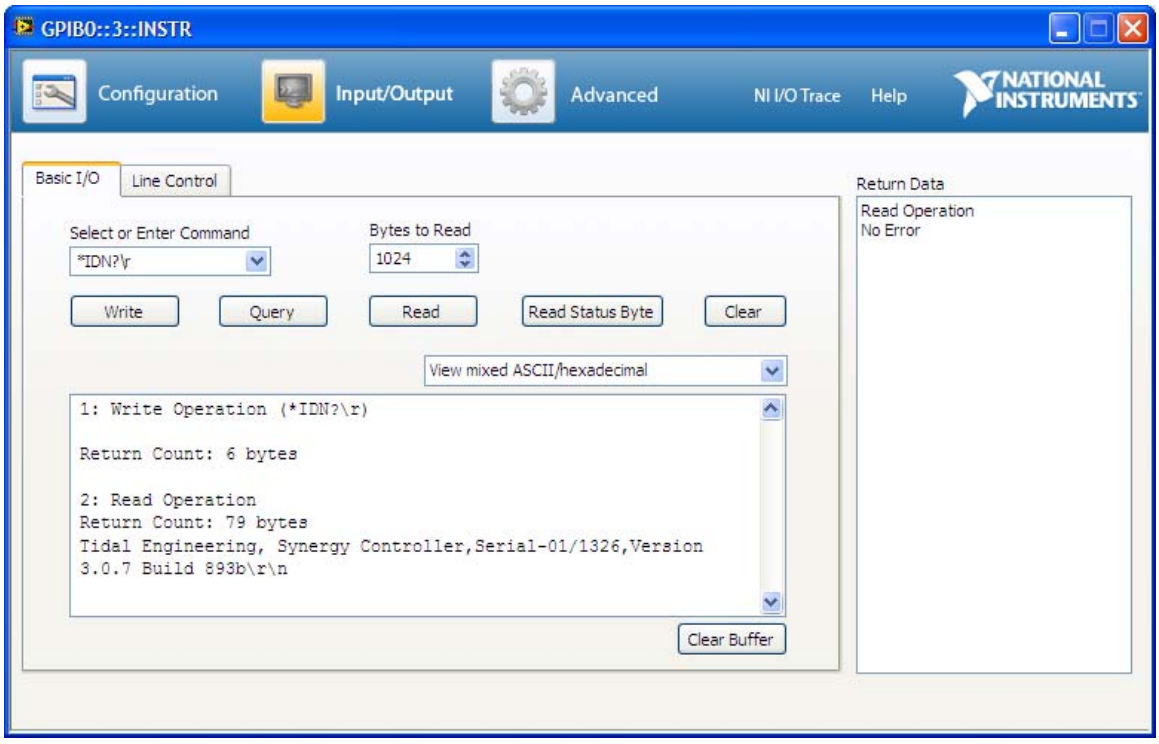
1. Select the GPIB controller in the list with the right-mouse and select **Scan for Instruments** from the menu.



2. Setup the I/O Settings as follows:

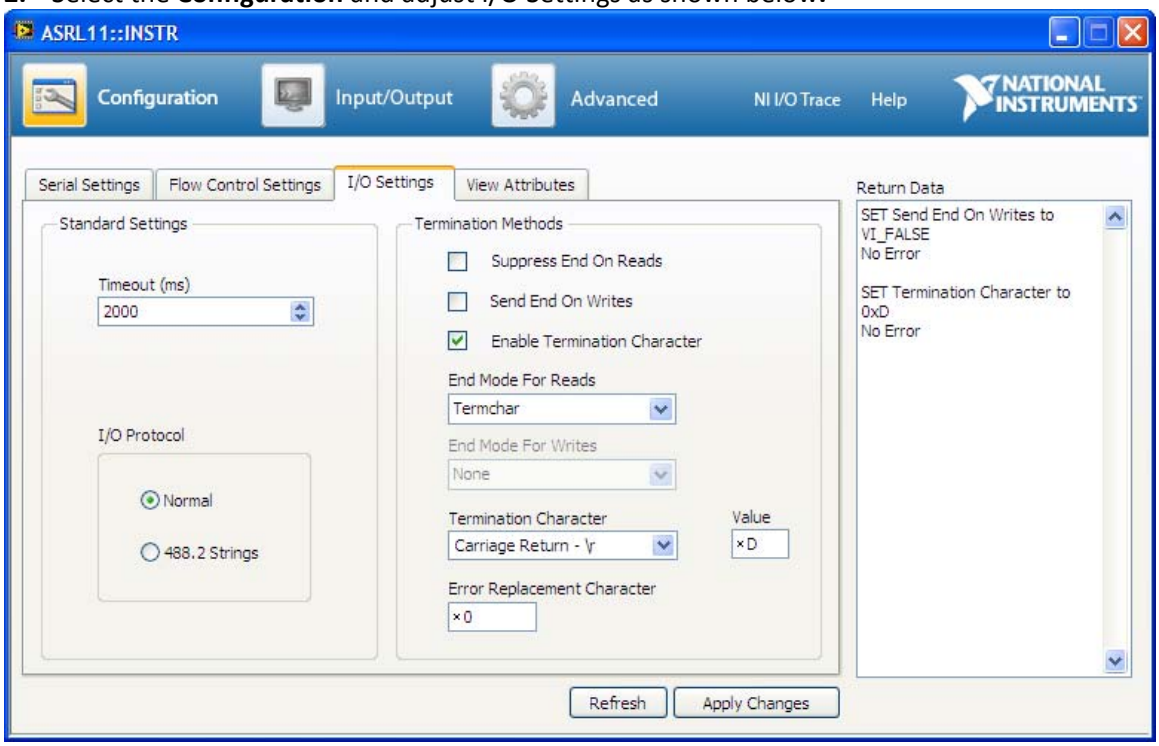


3. Select **Input/Output** and test the connection as follows:

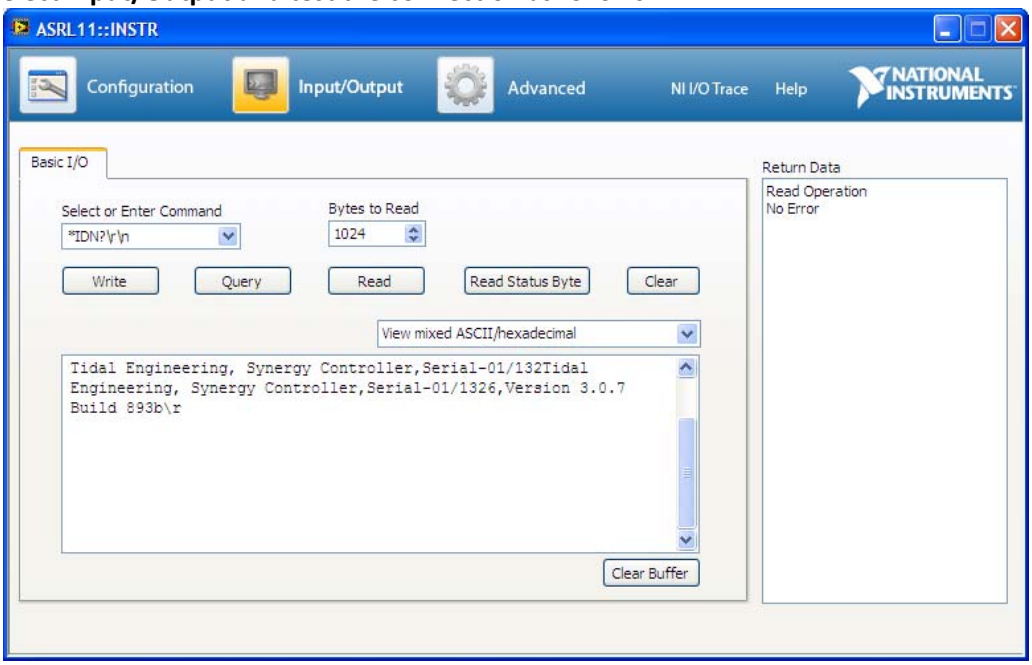


NI Measurement and Automation Explorer (NIMAX) Setup for RS-232

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 Controller Family

Tidal Engineering's Synergy Controllers, both the Synergy Micro 2 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 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
Fax: 973/328-2302
www.TidalEng.com
info@tidaleng.com