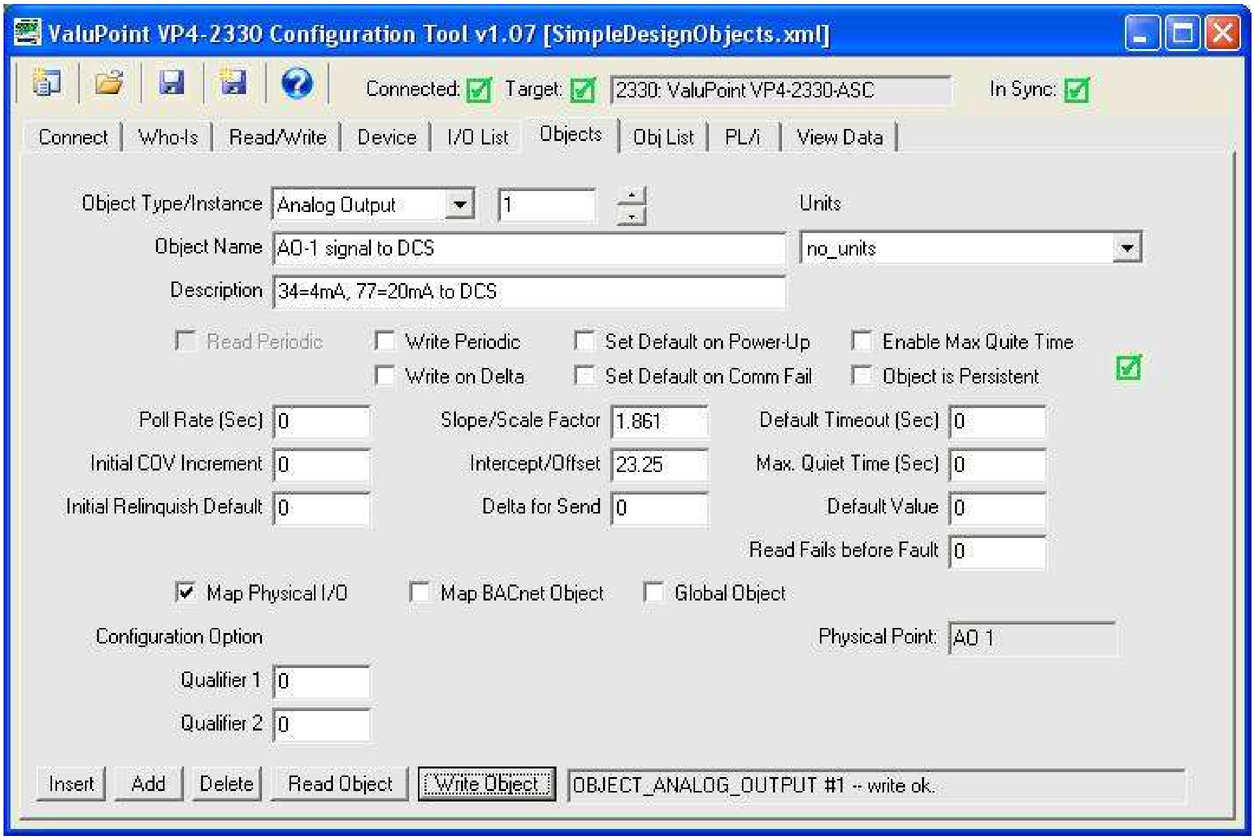
ValuPoint Application Example - 4-20mA Output Connected to External BACnet Device

Our customer had the following requirement: Read a value from another BACnet MS/TP device, and produce a 4-20mA signal proportional to a certain range of the value read. The ValuPoint VP4-2330 is capable of doing this with just minor programming.

An object in the ValuPoint can only be mapped to one function at a time, either physical I/O or mapping to an external BACnet MS/TP device. When mapped to an external BACnet device, the ValuPoint will poll that device periodically and store the data received in its own local object. The same mapped object cannot also interface with local physical I/O, so two objects are needed – one for mapping to the other BACnet device and one for interfacing to the physical I/O.

Linking the two objects so that data from the remote BACnet device effectively controls the analog output is a very simple control algorithm, but is still technically "control” because output is being autonomously manipulated based on external data. Some scaling is necessary, but object configuration in ValuPoint already provides for built-in scaling. Therefore, assuming the scaling is configured, the control program simply needs to copy data from the input object to the output object.

Mapping an AV object to read an AI from another MS/TP device would look like this using the ValuPoint configuration tool:



ValuPoint VP4-2330 Configuration Tool v1.07 [SimpleDesignObjects.xml]

Connected: ☑ Target: ☑ 2330: ValuPoint VP4-2330-ASC In Sync: ☑

Object Type/Instance: Analog Output 1

Units: no_units

Object Name: AO-1 signal to DCS

Description: 34=4mA, 77=20mA to DCS

□ Read Periodic □ Write Periodic □ Set Default on Power-Up □ Enable Max Quite Time

□ Write on Delta □ Set Default on Comm Fail □ Object is Persistent ☑

Poll Rate (Sec): 0 Slope/Scale Factor: 1.861 Default Timeout (Sec): 0

Initial COV Increment: 0 Intercept/Offset: 23.25 Max. Quiet Time (Sec): 0

Initial Relinquish Default: 0 Delta for Send: 0 Default Value: 0

Read Fails before Fault: 0

☑ Map Physical I/O □ Map BACnet Object □ Global Object

Configuration Option

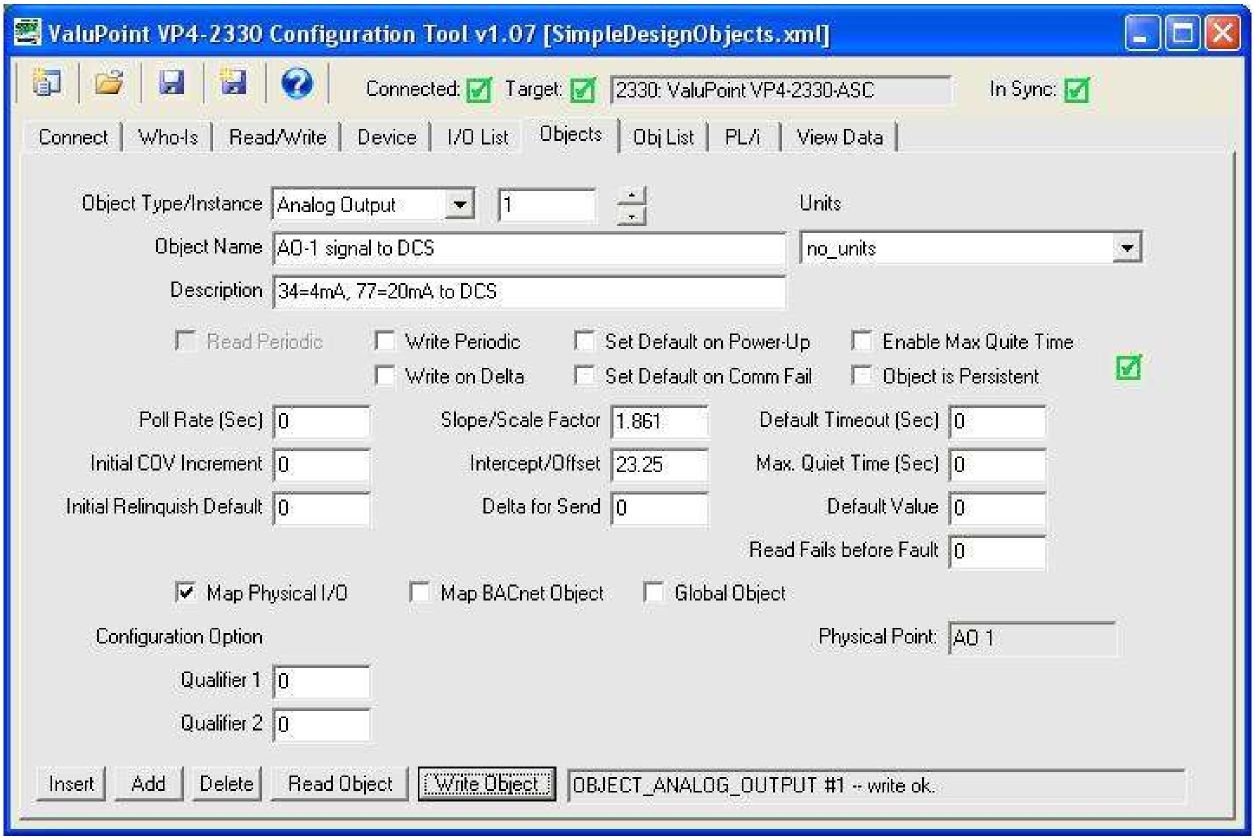
Physical Point: AO 1

Qualifier 1: 0

Qualifier 2: 0

OBJECT_ANALOG_OUTPUT #1 -- write ok.

Mapping the AO object to physical hardware will be done automatically if you loaded the default configuration file, or simply read the default configuration as shipped from the device. In any event, the AO should end up looking like this:



Connected: ☑ Target: ☑ 2330: ValuPoint VP4-2330-ASC In Sync: ☑

Object Type/Instance: Analog Output 1

Units: no_units

Object Name: AO-1 signal to DCS

Description: 34=4mA, 77=20mA to DCS

☐ Read Periodic ☐ Write Periodic ☐ Set Default on Power-Up ☐ Enable Max Quite Time

☐ Write on Delta ☐ Set Default on Comm Fail ☐ Object is Persistent ☑

Poll Rate (Sec): 0 Slope/Scale Factor: 1.861 Default Timeout (Sec): 0

Initial COV Increment: 0 Intercept/Offset: 23.25 Max. Quiet Time (Sec): 0

Initial Relinquish Default: 0 Delta for Send: 0 Default Value: 0

Read Fails before Fault: 0

☑ Map Physical I/O ☐ Map BACnet Object ☐ Global Object

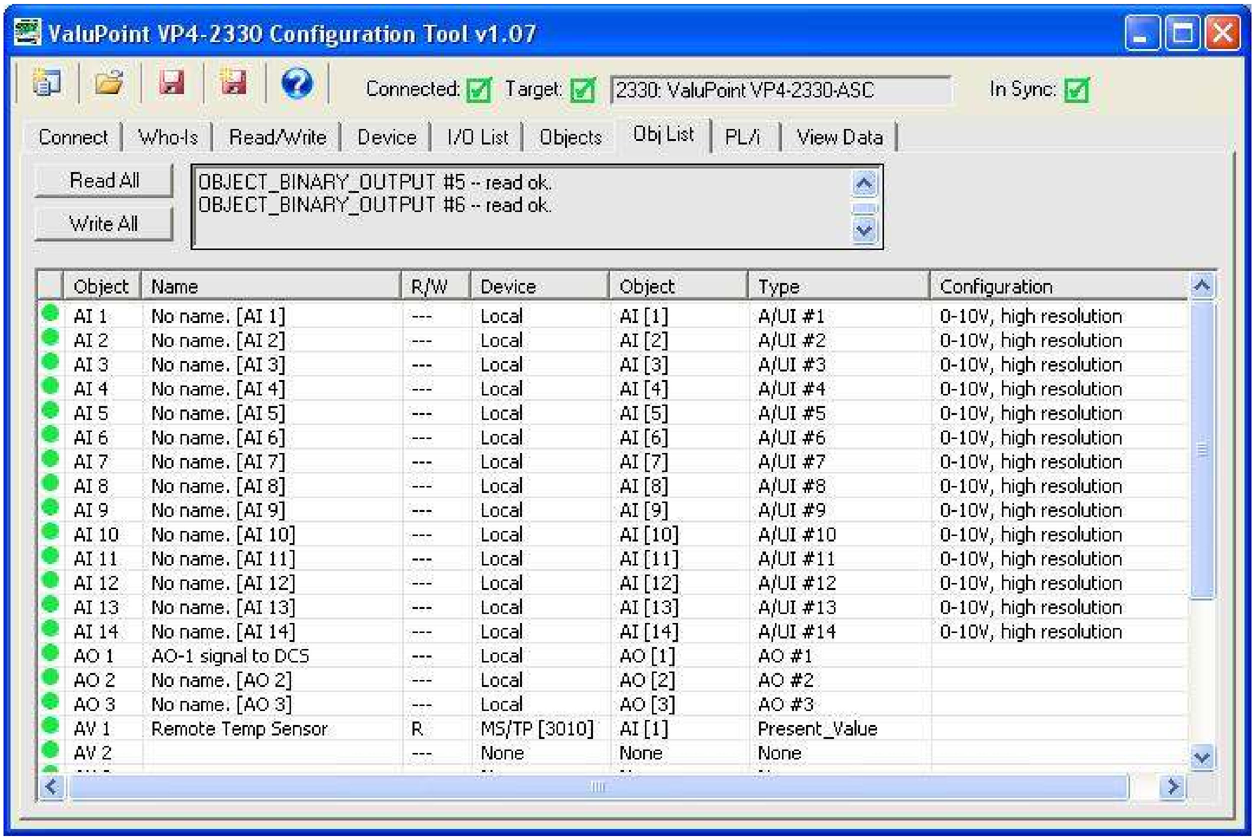
Configuration Option Physical Point: AO 1

Qualifier 1: 0

Qualifier 2: 0

OBJECT_ANALOG_OUTPUT #1 -- write ok.

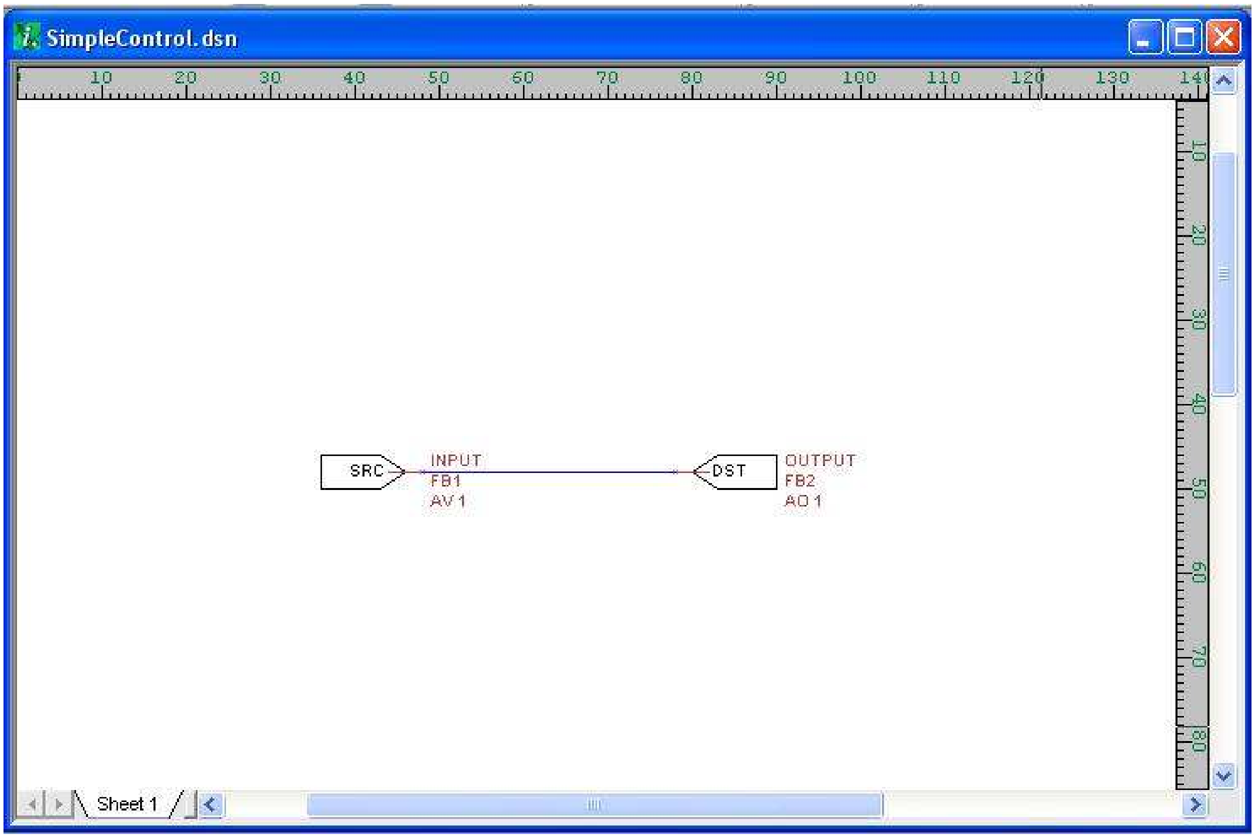
The object list in our simple test of this application now looks like this:



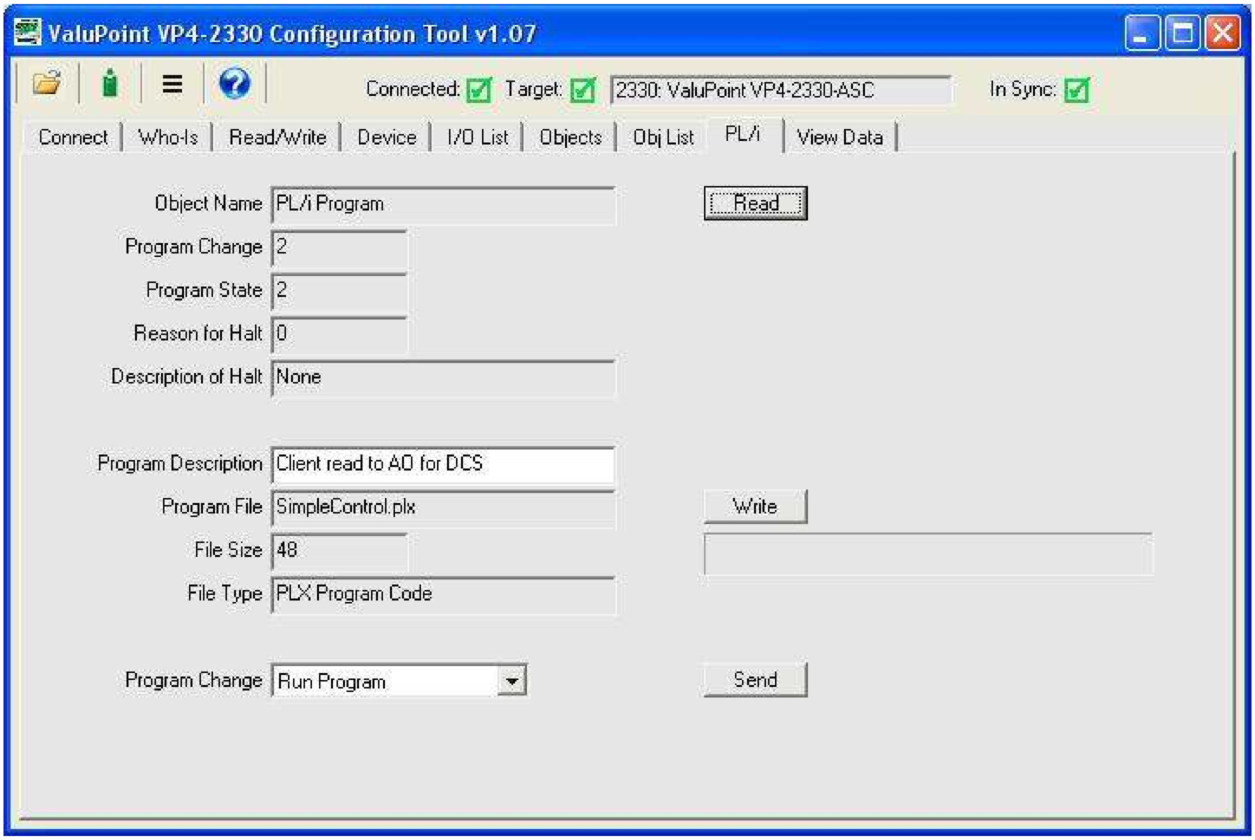
```
OBJECT_BINARY_OUTPUT #5 -- read ok.
OBJECT_BINARY_OUTPUT #6 -- read ok.
```

	Object	Name	R/W	Device	Object	Type	Configuration
●	AI 1	No name. [AI 1]	---	Local	AI [1]	A/UI #1	0-10V, high resolution
●	AI 2	No name. [AI 2]	---	Local	AI [2]	A/UI #2	0-10V, high resolution
●	AI 3	No name. [AI 3]	---	Local	AI [3]	A/UI #3	0-10V, high resolution
●	AI 4	No name. [AI 4]	---	Local	AI [4]	A/UI #4	0-10V, high resolution
●	AI 5	No name. [AI 5]	---	Local	AI [5]	A/UI #5	0-10V, high resolution
●	AI 6	No name. [AI 6]	---	Local	AI [6]	A/UI #6	0-10V, high resolution
●	AI 7	No name. [AI 7]	---	Local	AI [7]	A/UI #7	0-10V, high resolution
●	AI 8	No name. [AI 8]	---	Local	AI [8]	A/UI #8	0-10V, high resolution
●	AI 9	No name. [AI 9]	---	Local	AI [9]	A/UI #9	0-10V, high resolution
●	AI 10	No name. [AI 10]	---	Local	AI [10]	A/UI #10	0-10V, high resolution
●	AI 11	No name. [AI 11]	---	Local	AI [11]	A/UI #11	0-10V, high resolution
●	AI 12	No name. [AI 12]	---	Local	AI [12]	A/UI #12	0-10V, high resolution
●	AI 13	No name. [AI 13]	---	Local	AI [13]	A/UI #13	0-10V, high resolution
●	AI 14	No name. [AI 14]	---	Local	AI [14]	A/UI #14	0-10V, high resolution
●	AO 1	AO-1 signal to DCS	---	Local	AO [1]	AO #1	
●	AO 2	No name. [AO 2]	---	Local	AO [2]	AO #2	
●	AO 3	No name. [AO 3]	---	Local	AO [3]	AO #3	
●	AV 1	Remote Temp Sensor	R	MS/TP [3010]	AI [1]	Present_Value	
●	AV 2		---	None	None	None	

Next, use i.CanDrawIt to create a simple program to move the input value to the output object. The drawing will look like this:



Using the i.CanDrawIt tool, compile the file to create a .plx file. Then from the PL/i tab of the configuration tool, open the .plx file, and click Write to send the program to the ValuPoint. Once that is done, select Run Program from the Program Change list and click Send. The program should now be running, which you can verify by clicking the Read button. The status should appear as follows:


ValuPoint VP4-2330 Configuration Tool v1.07

Connected: ☑ Target: ☑ 2330: ValuPoint VP4-2330-ASC In Sync: ☑

Object Name: PL/i Program

Program Change: 2

Program State: 2

Reason for Halt: 0

Description of Halt: None

Program Description: Client read to AO for DCS

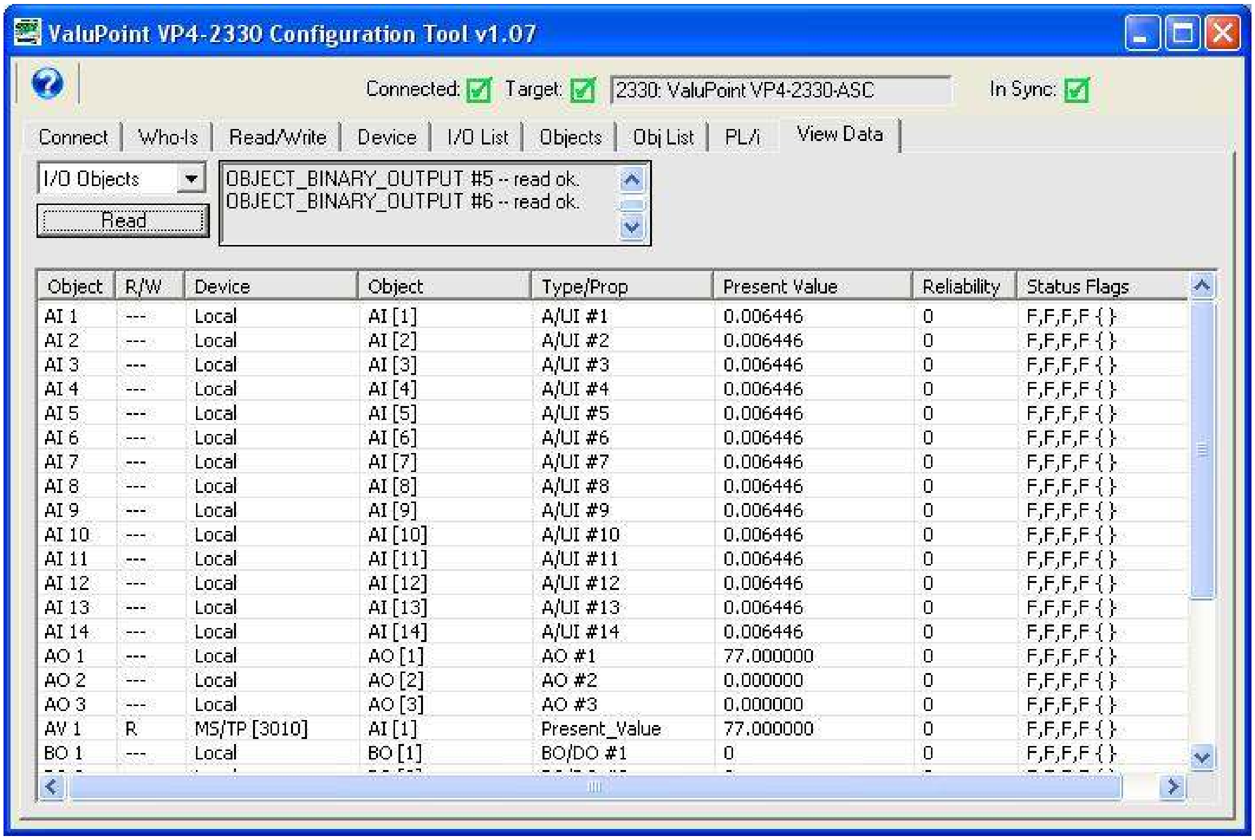
Program File: SimpleControl.plx

File Size: 48

File Type: PLX Program Code

Program Change: Run Program

Now you should be able to read present values on the Data List page and see the value of the AV being copied to the AO.



Connected: ☑ Target: ☑ 2330: ValuPoint VP4-2330-ASC In Sync: ☑

```
OBJECT_BINARY_OUTPUT #5 -- read ok.
OBJECT_BINARY_OUTPUT #6 -- read ok.
```

Object	R/W	Device	Object	Type/Prop	Present Value	Reliability	Status Flags
AI 1	---	Local	AI [1]	A/UI #1	0.006446	0	F,F,F,F { }
AI 2	---	Local	AI [2]	A/UI #2	0.006446	0	F,F,F,F { }
AI 3	---	Local	AI [3]	A/UI #3	0.006446	0	F,F,F,F { }
AI 4	---	Local	AI [4]	A/UI #4	0.006446	0	F,F,F,F { }
AI 5	---	Local	AI [5]	A/UI #5	0.006446	0	F,F,F,F { }
AI 6	---	Local	AI [6]	A/UI #6	0.006446	0	F,F,F,F { }
AI 7	---	Local	AI [7]	A/UI #7	0.006446	0	F,F,F,F { }
AI 8	---	Local	AI [8]	A/UI #8	0.006446	0	F,F,F,F { }
AI 9	---	Local	AI [9]	A/UI #9	0.006446	0	F,F,F,F { }
AI 10	---	Local	AI [10]	A/UI #10	0.006446	0	F,F,F,F { }
AI 11	---	Local	AI [11]	A/UI #11	0.006446	0	F,F,F,F { }
AI 12	---	Local	AI [12]	A/UI #12	0.006446	0	F,F,F,F { }
AI 13	---	Local	AI [13]	A/UI #13	0.006446	0	F,F,F,F { }
AI 14	---	Local	AI [14]	A/UI #14	0.006446	0	F,F,F,F { }
AO 1	---	Local	AO [1]	AO #1	77.000000	0	F,F,F,F { }
AO 2	---	Local	AO [2]	AO #2	0.000000	0	F,F,F,F { }
AO 3	---	Local	AO [3]	AO #3	0.000000	0	F,F,F,F { }
AV 1	R	MS/TP [3010]	AI [1]	Present_Value	77.000000	0	F,F,F,F { }
BO 1	---	Local	BO [1]	BO/DO #1	0	0	F,F,F,F { }

You will see that scaling for the AO has some unusual numbers in our example. The application illustrated here called for a 4-20mA signal corresponding to a temperature reading of 34-77F. In other words, if the value in the AV object is 34, the AO should be generating 4mA. If the value in the AV object is 77, the AO should be generating 20mA.

Scaling for analog inputs uses the traditional slope-intercept formula y=mx+b. Calculation of slope would be m=(y2-y1)/(x2-x1). Once slope (m) is calculated, solve for intercept (b).

Scaling for analog outputs would theoretically be the reverse of this process, meaning subtract intercept then divide by slope. However, the implementation was coded incorrectly. The ValuPoint correctly subtracts intercept first, but multiplies by slope instead of divide. Modifying firmware would result in the same model device in the field behaving two different ways. Rather than create this dilemma, we will simply revise the calculation needed.

Calculation of slope for analog outputs still uses the standard m=(y2-y1)/(x2-x1) formula. But when solving for intercept, the formula is y=(x-b)*m.

Analog outputs produce 0-20mA given a raw value of 0-100 (with percent being implied). To generate a traditional 4-20mA signal, a non-zero value is needed to get the 4mA offset. The 4mA offset corresponds to a raw value of 20 (or 20% of 20mA). Therefore, the Y values in the slope calculation should be 100 and 20, with X values corresponding to whatever range you wish to scale to for 20mA and 4mA respectively.

In our example, the slope (m) is then (100-80)/(77-34) = 1.86. Next, solving y=(x-b)*m for b, we find that the intercept is 23.25. The scaling could have been applied on the AV object instead using the input formula for intercept. But by scaling the AO instead, we can look at the Data List and see the actual value of the AV in the range of 34 to 77.

Qualifier 1 for the analog output should be the value of the dropping resistor at the other end of the loop. If left set to 0, it will assume 500 ohms by default. If the value is anything other than 500, enter that number as the qualifier.

Article ID: 16
Created On: Fri, Oct 17, 2014 at 2:13 PM
Last Updated On: Fri, Dec 12, 2014 at 11:24 PM