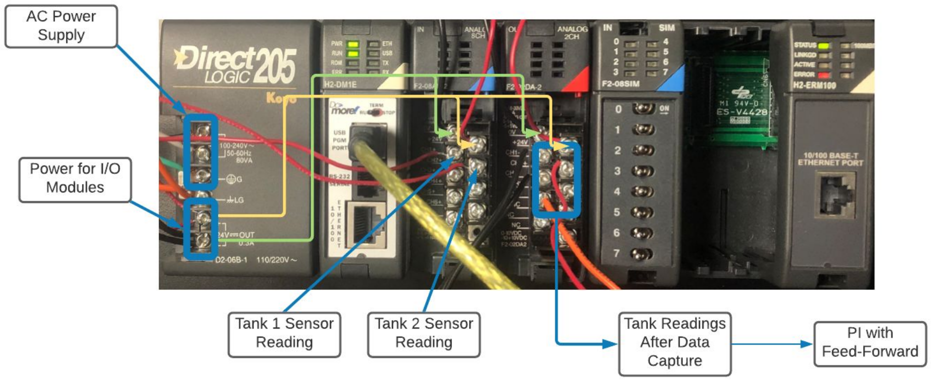
PLC For Data Capture



Method 1 *(pass signal through PLC and back out to system)*
----

1. **Powering the System**
   1.1. Power on the PLC and the I/O modules to be used *(8-channel analog in, 2 channel analog out).* Power the PLC directly from an outlet to the power module all the way on the left side of the controller. Connect the wires in the AC ports from top to bottom in the order black, white, then green. To make the connection, simply unscrew the screws, slide the correct wire into a position where it makes ample contact with the metal plate, then screw it down into place.

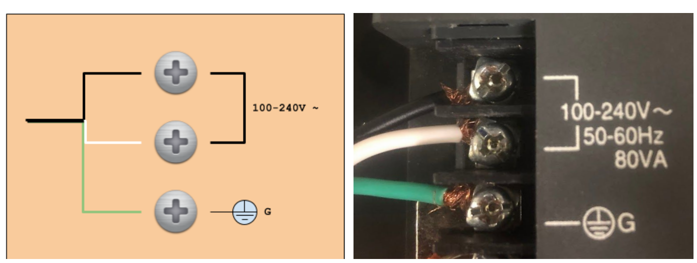


Figure 1a PLC Power Module

To power on the I/O modules, the 24V DC OUT needs to be used. This should be the bottom 2 ports on the power module. Each I/O module should have 2 pins at the top for these power lines. Connect the wires as shown in figure 1b following the same procedure as for the AC supply. Figure 1b shows how to connect to only 1 I/O module (analog in, for example). To power on any other modules, simply perform the same operation; multiple wires can connect to the same port.

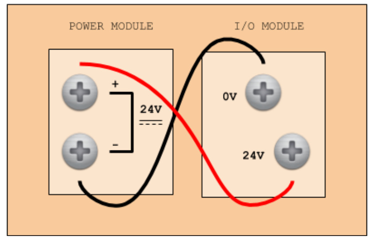


Figure 1b Powering I/O Modules

After setting up the wires as needed, it may be helpful to use tape and group them all together, to be pushed to the side since they will not need any further manipulation.

2. Input / Output Setup

2.1. To set up the PLC for data capture, the user needs to know how many signals they intend to pass through. With the F2-08AD and F2-08DA modules, method 1 can only deal with 2 signals (this will be dealt with in Method 2).

2.1.1. Analog In Setup

2.1.1.1. To take in analog input, simply directly connect the wire that carries it into an analog in port. There are 8 ports, allowing for 8 inputs. Be mindful of which port is chosen for which signal, as each port has a label in the PLC memory.

2.1.1.2. To read from something like a sensor, as was done with the quad-tank system, the signal must also connect to ground.

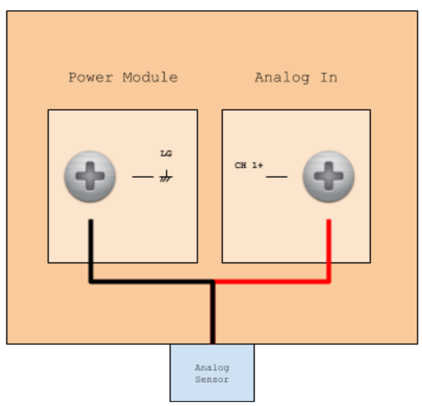


Figure 2a Analog in from Sensor

2.1.2. Analog Out Setup

2.1.2.1. The analog output module will return the processed signal as you define it in ladder logic. There are simple ports for each channel that are easily identified. Connect wires to the ground and signal port of the channel to be used, and simply connect those wires to their location in the system being captured.

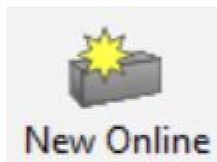
3. Ladder Logic and Do-More Designer

3.1. The ultimate goal of using the PLC for data capture is to take an input, and pass it back out the exact same signal. This means there is no need for any real processing to be done on the PLC.

3.2. Setting up a New Project

3.2.1. Power on the PLC by plugging it in, and connect the H2-DM1E processor module to the computer with the given USB.

3.2.2. Select "New Online" in the top left corner of the Do-More Designer. You will be prompted to make a connection to the PLC. Follow the instructions and connect to the PLC (creating and saving new names for the same link will not override anything).



3.3. Main Logic Blocks

3.3.1. The ladder logic you will code will execute sequentially.

3.3.2. The most important logic blocks for this task set will be SCALE and CLAMP.

3.3.2.1. SCALE takes an input (WXO for the first analog in port) and outputs the scaled value to the chosen variable (WY0 for the first analog output). The programmer then identifies the range of the input that is possible, and identifies the desired range of the output. For example, if you want to output half the voltage read in, In Min = 0, In Max = 2 and Out Min = 0, Out Max = 1. It is important to note that the I/O modules convert the analog signal to a digital value. Without changing the memory configuration, the min for the I/O modules should be 0 and the max should be 4095.

Figure 3a SCALE Instruction

3.3.2.2. CLAMP limits the range of a variable between given values. It can be used to clamp the output variable between 0 and 4095 so that negative outputs can be avoided.

Figure 3b CLAMP Instruction

3.3.3. These blocks can be found in the instruction toolbox on the right side under the analog process tab. Drop these blocks on the right side of the ladder rungs in the desired order. You can drop "normally closed contacts" on the actual rung but if not, when you try to save the project, you will be prompted to create "always on" contacts and can just say yes. Saying no will also work but merges rungs.

3.4. Actual Code Structure

3.4.1. Scale Inputs to Outputs

3.4.1.1. Ideally this would be 0-1 to 0-1 but because of noise in sensors and negative readings, this may need to be modified significantly. This will require some trial and error and comparing signals with a multimeter to make sure your input matches your output.

3.4.2. Clamp Outputs

3.4.3. Scale Outputs to R memory variable

3.4.3.1. This scaling will be to convert the digital output value to volts for viewing, and ultimately exporting to a csv.

3.5. Viewing Live Variable Values and Exporting

3.5.1. At the top menu, click on "Trend" and then 1 Pane

3.5.1.1. Here you will choose what variables you want to view.

3.5.2. In elements, enter the variable name you wish to see. R1 for example may be your output in V. You can also add constant values if you want to see a set-point. Line colors and axis ranges can also be customized.

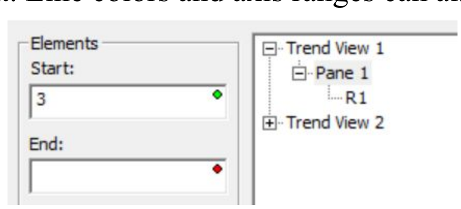


Figure 3c Trend View

3.5.3. To export data to a CSV, simply hit export and select the desired location.

3.5.3.1. Data Smoothing

3.5.3.1.1. The PLC does not perform any sort of mathematical smoothing but does offer the option to define the desired sample time. The default setting is 1 second which is enough for slow systems where behavior is observed over a long period of time but the PLC can also drop down to ms intervals.

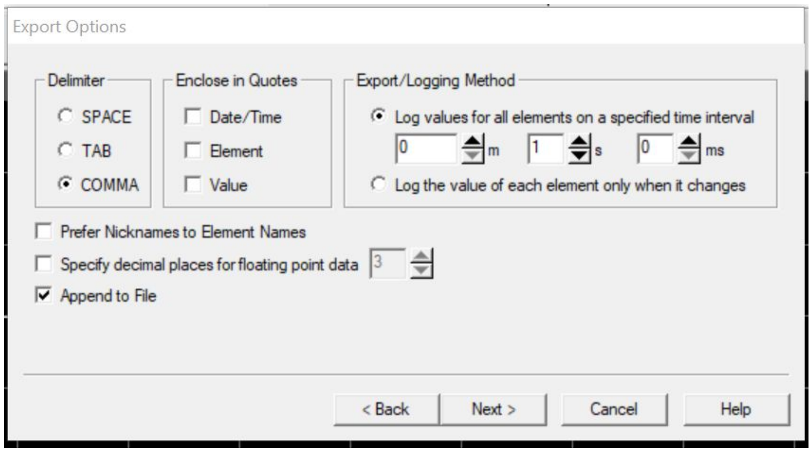


Figure 3d Export Options

3.5.3.1.2. After choosing a sample time, select the desired time range on the prompt, and export.

4. Method 1 Notes / Discussion

4.1. To simplify the issue of negative values in the input, it may be worth clamping the input before scaling. I tried this by clamping WX0 and storing it in WX0 after but this didn't seem to work as intended so it may need to be stored in R memory.

4.2. It is important to not have stripped wires be exposed more than they need to be. The PLC keeps so many wires in close proximity, making it extra important to avoid unintended contact

4.3. It is important to validate your inputs and outputs with a multimeter, to make sure all your code is working as intended.

4.4. When changing the setpoint with a step, this can be tapped with method 2 or added manually in the csv pretty easily, you just need to write down at what time in the trend view the step was made.

4.5. To help with debugging the built in Data View can be used.

Method 2 *(tap signal to PLC, no output)*

----

1. **Methodology**
    1.1. Method 2 follows the same layout as method 1 for the bulk of the set-up, except it does not output any signals. Rather than have the PLC act as a middle-man, method 2 treats it as a third party observer. Simply wire the desired signals to both the system and to the PLC and follow the same ideas from method 1. This is a much simpler method in theory but was tested out of fear that the tapping of the signal would inadvertently change the behavior of the system. If time allows, it is worth investigating as a stand alone method of data capture or a supplement to method 1 to read in things like a setpoint.
2. **Actual Code Structure**
    2.1. Scale Inputs to R memory variable
        2.1.1. This scaling will be to convert the digital input values to volts for viewing, and ultimately exporting to a csv.