Galil RIO PLC Analog Input/Output Exercises and Motor Control Exercises

MTRN3500 Computing Applications in Mechatronic Systems

Use your Galil class functions to do the following.

Problems

Analog I/O Exercises

1. The signal lines of the low byte of the Galil digital outputs are connected to a DAC (Digital to Analog Converter) and the resulting voltage is displayed on a digital voltmeter (DVM) (lower digital display of the laboratory set up). Write a `main()` function that will allow you to obtain the relationship between the digital values you would write to the lower byte and the voltage values displayed on the DVM.

2. The value generated by the above mentioned DAC is connected to the analog input channel 0 (AI0) of Galil. Write a `main()` function that reads the AIO voltage and display it on the computer screen.

3. Write a `main()` function to demonstrate that when you write different digital outputs to the lower byte of Galil digital outputs;
   - the voltage displayed on the computer screen changes and
   - the same voltage values are displayed on the DVM, and
   - the value displayed on the DVM corresponds to the relationship you built in problem 1. above.

4. Write a `main()` function that will allow you to write to the digital output low byte and then to read the incoming AI0 and then to send the voltage read form AI0 to AO7. This way you can control the motor movement using a digital output to the low byte.

5. Write a `main()` function that will control the motor speed using `SpeedControl()` function. You need to carry out the following steps to make this work.

   (a) Make sure the motor is stopped.

   (b) Reset the encoder to zero.

   (c) Set a value to `setPoint`. This has to be a realistic number expressed in encoder counts per second.

   (d) Call `setKp(), setKi(), setKd()` functions to set Kp, Ki and Kd values. Start by setting Kp to 1, Ki = 0 and Kd = 0 and then experiment to achieve good values for Kp, Ki and Kd to get you good motor speed control.

6. Write a `main()` function that will control the motor position using `PositionControl()` function. You need to carry out the following steps to make this work.

   (a) Make sure the motor is stopped.

   (b) Reset the encoder to zero.

   (c) Set a value to `setPoint`. This is the desired final position of the motor expressed in encoder counts.

   (d) Call `setKp(), setKi(), setKd()` functions to set Kp, Ki and Kd values. Start by setting Kp to 1, Ki = 0 and Kd = 0 and then experiment to achieve good values for Kp, Ki and Kd to get you good motor position control.