CMPE 121L: Microprocessor System Design Lab

Fall 2016

Lab Exercise 1

Check-off Due in lab section, week of October 10, 2016

Lab Report due: Within 3 days after checkoff due

In this exercise, you will learn the basics of using the GPIO pins of the PSoC-5 kit and how to monitor and control the inputs/outputs from a program. You will also learn the basics of pulse-width modulation. PWM is used for control of motors, servos, etc. It can also be used to control the average DC voltage or current in a circuit (for example, it can be used for controlling the brightness of a light source) digitally.

Part 0: GPIO Pin Toggling

The purpose of this part is to familiarize with how to configure the GPIO pins as output ports and control them from a C program.

1. Open the schematic editor of PSoC Creator and add a digital output pin from the component catalog. Right click on the pin symbol and click "open datasheet". Review the data sheet to understand the various configuration options. Make sure to read the Application Programming Interface (API) section.
2. Assign the pin number to P6[2], which is connected to LED 3 on the board
    a. Under the project workspace open the file [project name].cydrw
    b. On the right hand side, set the Port as P6[2]
    c. Make sure to set the *Lock* option to force the assignment of the pins.
3. There are three ways to control the state of an output pin from a program: (i) using Per-Pin APIs, (ii) Using Component APIs, and (iii) using a Control Register. The first two are described in the data sheet. For the third option, add a Control Register from the component catalog to the schematic and connect it to the output pin.
4. Write three programs to toggle the output pin using the three different approaches, to make the LED go on and off.
5. In your report, describe the pros and cons of each approach. When will you use each of them?

CHECKOFFS:

1. Per-Pin method
2. Component API
3. Control Register

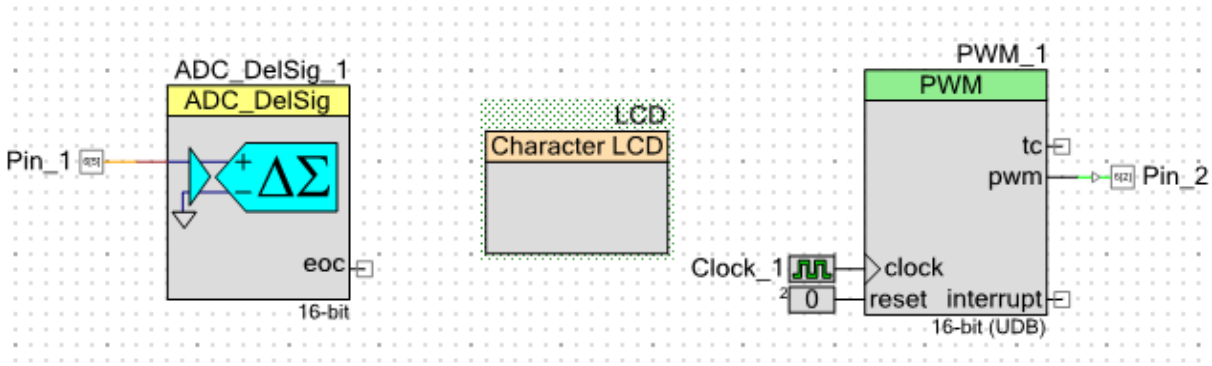
Part 1: LED Brightness Control using PWM

In this part, your program will continuously read the setting of a potentiometer and use it to set the brightness of one of the red LEDs. There is a potentiometer on the PSoC 5 board that you can use, but you will get better results with the potentiometer in the parts kit (the former has logarithmic behavior, while the latter is linear).

1. Using PSoC Creator, generate the datasheets of the following components and study them:
    a. PWM
    b. Delta Sigma ADC (this will be covered in more detail in class later).
    c. Character LCD
    d. Clock

    The datasheets also provide the API functions associated with each of the components. You need to be familiar with them to develop the program.

2. Create a schematic with the components as shown below:



Right click to configure the Analog-to-Digital Converter (ADC):

1. Set the ADC as single ended
2. Set the output resolution to 16 bits

Set the input range between Vssa(analog ground) and Vdda(analog supply voltage). The output of the ADC should range from 0 to 0xffff (16-bits). Define the clock frequency as 1 MHz and the period of the PWM signal as 999, which results in a 1 kHz signal on the PWM output.

Connect the input terminal of the ADC (Pin_1) to physical pin P6[5], which is connected to the potentiometer on the board, and the PWM output (Pin_2) to physical pin P6[2], which is connected to LED 3.

a. Under the project workspace open the file [project name].cydrw
b. On the right hand side, set the Port as P6[2]
c. Make sure to set the *Lock* option to force the assignment of the pins.

3. Write a program to read the ADC output and control the duty cycle of the PWM output in a continuous loop. The duty cycle can be controlled by keeping the period of the PWM signal constant and changing the ON interval between 1 cycle and 999 cycles, based on the value read from the potentiometer. For debug purposes, also display the value read from the potentiometer on the 2x16 LCD display.
4. Read the potentiometer in as a 16-bit and as a 32-bit value. What differences do you see? Which will you use and why?
5. Note that the on-board potentiometer has a logarithmic range. Replace it with the potentiometer supplied in the parts kit by connecting the middle pin of the potentiometer

to a GPIO pin of the board and connecting the outer pins of the potentiometer to ground and 5V.

CHECKOFFS:

1. LED is controlled by on board potentiometer.
2. LED can go from completely off to completely on.
3. LCD displays ADC value from 0 to FFFF (hex) (check edge cases)
4. LED is controlled by off board potentiometer

Part 2: Frequency Meter

In this second part, you will generate a variable frequency signal and measure its frequency using a timer and counter.

1. Using PSoC Creator, generate the datasheets of the following additional components and study them:
    a. Counter
    b. Timer
    c. Control and Status Registers
2. Modify the design in part 2 to control the frequency of the PWM output using the external potentiometer. The PWM must generate a square wave with frequency in the range 1 kHz – 12 MHz, based on the setting of the potentiometer (this requires changing both the period and the ON interval of the PWM block based on the value read from the ADC).
3. Add a counter block to the design.
    a. Use the UDB option to implement the counter, This will make the count input appear.
    b. Take the PWM output (NOT TC) and set that as the count input.
4. Add a timer block to the design. Set it up to generate an interrupt every 2 milliseconds.
5. Write the interrupt service routine (ISR). This code should read the counter value, calculate the frequency (by dividing the counter value by the period of the timer), and display the value on the LCD display.
6. Note that, to measure the frequency continuously, you should either clear the counter or remember its current value before returning from the ISR. This gives you two options to determine the number of pulses during the measurement interval. Which of these do you think will be more accurate? Try out the two options in your design and report the percentage difference in the measured values, at several different points. Include this information in your lab report.
7. Did you observe that the measurements at the low end of the range are not very accurate? How can you improve the accuracy when the frequency is low?

CHECKOFFS:

1. ISR routine is short (No print functions!)

2. LCD displays potentiometer value, expected frequency, calculated frequency.
3. Clear counter method is working
4. Track counter method is working

What to submit?

- Schematics and code for all parts as PNG or PDF (Please review guidelines in lecture slides to decide which files to submit).
- Report containing descriptions of your designs and answers to the questions above, as a PDF file
- Your project folder. Make sure to delete all generated files.
- You should also demonstrate the working design for all parts to your TA.