ECE 367 - Experiment #7
Introduction to Pulse Width Modulation

Spring 2006 Semester

Introduction

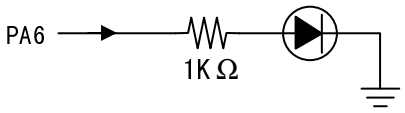
In this experiment you will write code to raise and lower the perceived intensity of an LED using Pulse Width Modulation (PWM). This will be accomplished using software delay loops to generate the desired waveform.

Required Hardware

In addition to the MicroStamp11 module, this experiment requires the docking module.

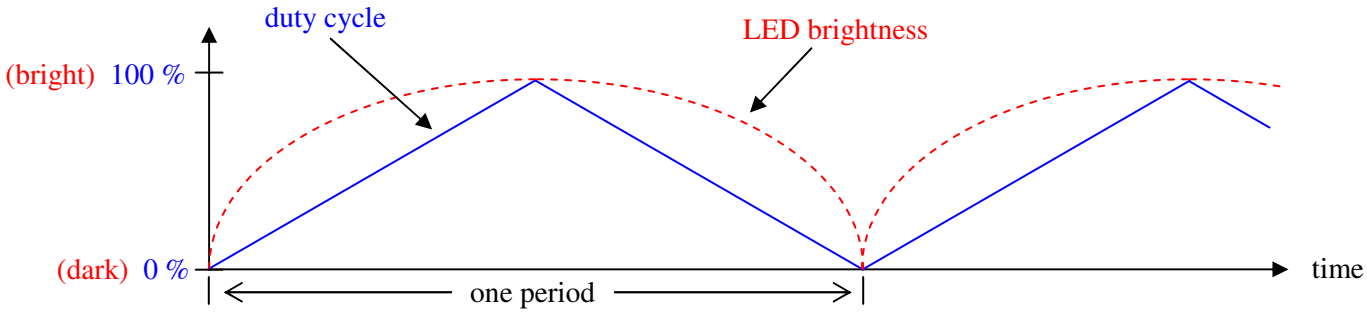
Wiring Diagram

This experiment does not require any circuit construction, as there is already a red LED connected to PA6 output on the docking module:



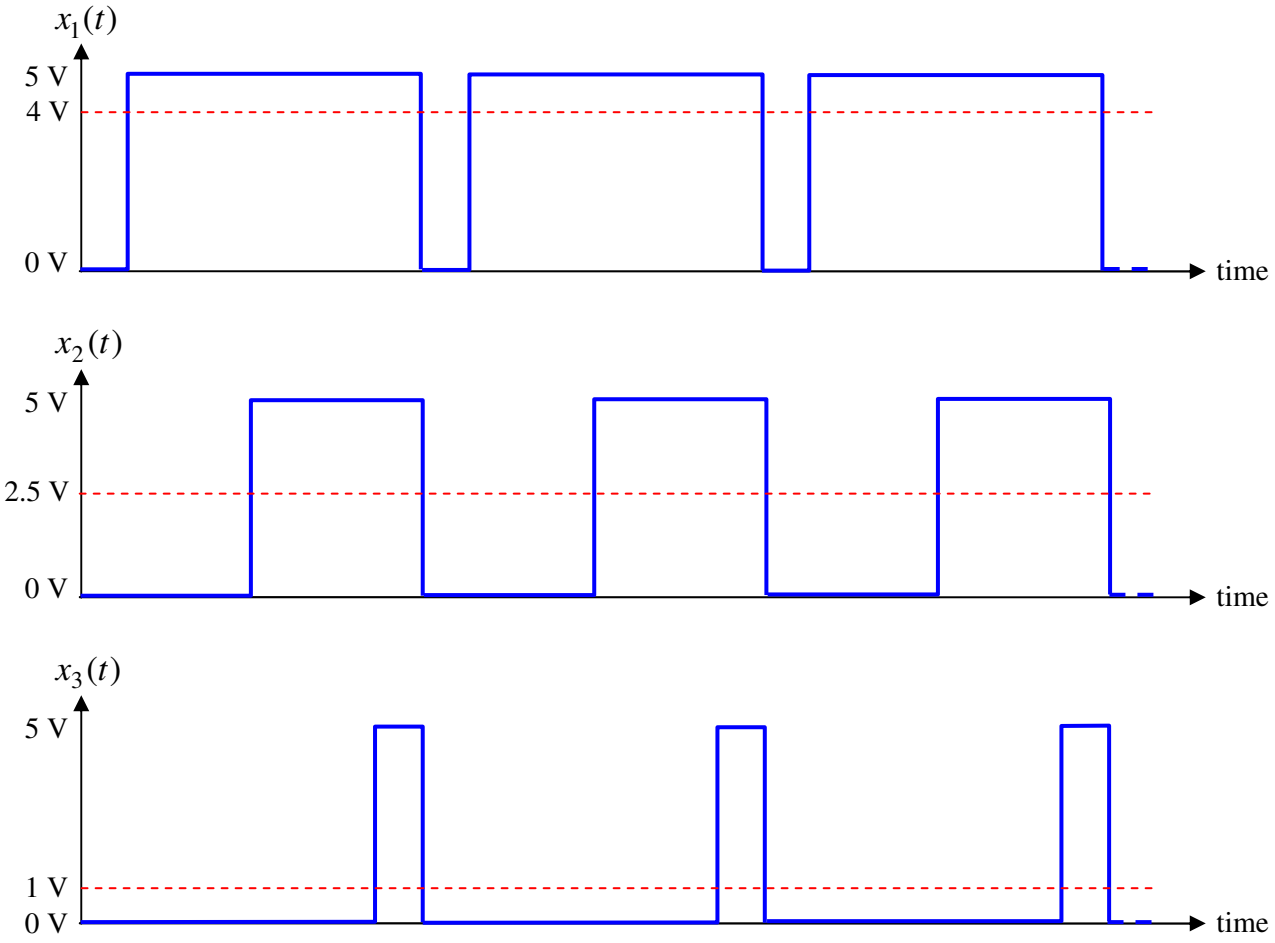
Goal

You are to generate a periodic signal at output port PA6 whose duty cycle varies with time as shown by the solid blue line on the graph below. This will cause the perceived brightness of the LED to vary approximately as shown by the red dotted line (as the visual system is nonlinear in its perception of brightness as a function of duty cycle):



Pulse Width Modulation

Even though the 68HC11 has only digital outputs, we may still create analog-style output effects through the use of pulse width modulation. Consider the three waveforms below:



These three waveforms are periodic and have the same period, but not the same average value (as represented by the horizontal dashed lines). The fraction of time that a periodic binary waveform spends in its high state is called the "duty cycle" of the waveform, and is normally expressed in percent. Therefore: $x_1(t)$ has a 80% duty cycle, $x_2(t)$ has a 50% duty cycle and $x_3(t)$ has an 20% duty cycle. If each of these waveforms is passed through a lowpass filter to extract only the d.c. term, we would obtain the constant values represented by the dashed lines. In fact, producing a periodic binary waveform having the desired duty cycle and then lowpass filtering this signal is a simple method of digital-to-analog conversion. In some applications the electrical lowpass filter is not required due to lowpass effects that already exist. Such is the case with LED intensity control: the slowness of response of the human visual system inherently performs lowpass filtering to make a rapidly-flashing LED appear to be lit at constant intensity level proportional to the waveform's duty cycle.

PWM Generation in the 68HC11 using a Software Delay Loop

There are different methods that may be used to generate pulse width modulation using software. The best way to do this, as shown in the waveform diagrams above, is to keep the period of the PWM constant and independent of the duty cycle. Therefore the method we will use will be as follows:

1. Clear the output pin low
2. Wait for t_1 sec
3. Set the output pin high
4. Wait for $(T-t_1)$ sec
5. Repeat from Step 1.

The duty cycle for the generated waveform is then $(t_1 \div T) \times 100\%$, and the period of this waveform is T sec (frequency = $1/T$ Hz).

Here is an 68HC11 assembly language subroutine to implement Steps 1-4 above:

```
PWM_Cycle:

      BCLR   PortA,X,$40    ; PA6 <-- 0
      LDY    Count2         ; Count1, Count2 = 16 bit values in memory
B0:   CPY    Count1         ; Compare Y to Count1
      BHI    B1             ; if (Y > Count1) then B1
      BSET   PortA,X,$40    ; PA6 <-- 1
      BRA    B2
B1:   NOP
      NOP
      NOP
      NOP
      NOP
B2:   DEY
      BNE    B0
      RTS
```

This subroutine produces one period of a PWM waveform, having duty cycle equal to Count1/Count2 × 100%, at output pin PA6. It works this way: register Y is initialized to Count2 and PA6 is reset to zero. With each iteration of the loop, Y is decremented by 1. Once it is detected that Y ≤ Count1 then PA6 is set to 1. Eventually when Y reaches zero the subroutine returns to the calling routine.

The NOP (No Operation) cycle-wasting instructions are used to give a constant number of clock cycles per iteration, regardless of what branch is taken after comparing Y to Count1. This way the period of the PWM waveform will only be a function of Count2, regardless of Count1 (that controls duty cycle).

Method

To keep the LED that is tied to PA6 from visibly flashing, the PWM frequency should be 30 Hz or higher. By having Count2 = 500 this requirement is met. Count1 is then varied over the range [0, Count2].

Write code to realize this pseudocode:

```
Program 7 pseudocode:

        Count2 ← 500

        repeat
              Count1 ← 0
              while (Count1 < Count2)
                    call  PWM_Cycle              ; generate one waveform cycle
                    Count1 ←  Count1 + 1         ; increase duty cycle
              end while

              Count1 ← Count2
              while (Count1 > 0)
                    call  PWM_Cycle              ; generate one waveform cycle
                    Count1 ←  Count1 – 1         ; decrease duty cycle
              end while
        end repeat
```

Questions

1. Find the number of clock cycles per instruction within the iterative portion of subroutine PWM_Cycle. How many clock cycles are there in one iteration?

2. Find a formula for the frequency of the pulse-width modulated waveform in terms Count2, when the clock cycle is assumed to be 0.5 μsec in duration (2.0 MHz master clock frequency). What is the frequency of the PWM waveform in this experiment?

3. Calculate the approximate period of the LED perceived brightness waveform, in sec.

4. How would you propose to modify the code so that *perceived brightness* is varied linearly with time? (In its present form the code varies *duty cycle* linearly with time.)

Extra

Plug the microcontroller module into your breadboard, and hook up an oscilloscope between PA6 and ground. With proper triggering level, you will clearly see the PWM duty cycle changing with time.