EE-3306 HC6811 Lab #4

Counters and Pulse Width Modulation

**Objectives:** The purpose of this lab is to learn how to generate a pulse width modulated signal and generate a timer to keep track of real time using 2 output compare functions. The OC function used for the timer is also used to control the rate at which the duty cycle of the PWM wave is changed.

---

1. **Prelab Questions:**
   1. What values are necessary (High, Low, Period) to generate a 20% duty cycle, 800 Hz waveform? Look the section in the Reference manual that talks about output compare function.
   2. What is pulse width modulation?
   3. Name some uses of pulse width modulation.

**References**

**Equipment for this lab:**
1• 68HC11 trainer kit, to include 68HC11 EVBU and prototyping strips.
2• IBM compatible PC to connect to the trainer kit via an RS-232 serial cable.
3• Agilent 54621D oscilloscope.
1• Floppy disk provided by student.

**Laboratory Exercise**

Notes:
1• Make sure your development EVBU board is connected to power (green LED on board ON) and the serial port of the EVBU is connected to the serial port of your development PC containing the AXIDE software, configured to the correct port at the correct baud rate, etc.
• The lab TA has a copy of the .s19 files for each part of the lab. The TA will demonstrate the waveforms to you at the beginning of the class.

**Task for the lab:**
In the previous labs you learnt how the output compare functions work. An interrupt is generated at the OCx pin when the value in the corresponding TOCx register equals the value in the free running counter TCNT.
In this lab the same output compare functions will be used again to generate a PWM signal and also a real time clock. The duty cycle is changed by 1% and it is changed at the rate of 1s. In the following sections, you will learn how to generate the PWM wave and the real time clock.

Program: **stud_PWM_timer.RTF**

Before the main program begins, assign labels to all the registers that will be used in the program including TCTL1, TMSK1, and TFLG1…

Assign variables needed in the program and also the interrupt vector addresses.

In the main program the first thing that should be done is that the control registers for the interrupts should be configured (TCTL1, TMSK1 and TFLG1).

Initialize the necessary variables needed.

To generate a real time clock:

- The first step should be to assign separate variables to keep count of seconds and minutes, say ‘sec’ and ‘min’
- Fix the time rate at which interrupts are generated by adding an integer number into the TOC2 register. For ex: To generate interrupts every 5ms or 10ms, you need to add 10000 or 20000 to TOC2 respectively.
- Set a counter which counts the number of interrupts generated so that it can act as a reference point to keep track of ‘seconds’. For ex, if an interrupt is being generated every 10ms and the counter reads 100 at some point, it means a second has passed.
- The ‘sec’ variable is updated every time the counter reads 100 and the counter is immediately reset back to 0 (in case the rate of interrupt generation chosen is 10ms).
- All this is done in one interrupt vector subroutine say int_TOC2 (since you will be using OC2).
- Once the ‘sec’ variable reads 60, the ‘min’ variable is updated and ‘sec’ is reset back to 0.

Note: Every time an interrupt occurs it is important to reset the interrupt flag to enable continuous generation of interrupts. This should be done at the end of the interrupt vector subroutine.

To generate a PWM wave:

The procedure to generate a square wave is similar to what was done in Part 2 of the previous lab. TOC3 is used to generate this square wave. The initial duty cycle will be 5%.

To change the duty cycle every second:
To change the duty cycle by 1% every second the high and low variables have to be updated every second.

The ‘high’ variable is increased by 1% of the total period and the ‘low’ variable is decreased by 1% of the total period.

To update the ‘high’ and ‘low’ variables every second, the int_TOC2 subroutine can be used.

In this particular subroutine, which keeps track of every passing second, the ‘high’ and ‘low’ variables are updated every time the ‘clks’ variable is incremented.

When the duty cycle reaches 95%, the ‘high’ and ‘low’ variables are initialized back to their original values (5% duty cycle).

Note: When you want to change the duty cycle in the int_TOC2 subroutine, note that the integer values used for interrupt generation might not be same again. This is because the program run time affects the interrupt generation. So, choose the values of the counters carefully.

The program provided to the students (stud_PWM_timer) will have the subroutine which will display the time on the LCD screen. The rest of the code should be filled in by the students using the hints given above. The ‘disp_time’ subroutine used to display time is explained below:

The label ‘TIME’ is first displayed on the screen using ‘cputs’.

The ‘clkm’ variable which keeps track of the minutes is displayed next by converting the value to decimal.

The symbol “:” symbol is displayed next followed by the ‘clks’ variable which keeps track of the seconds (by converting it to decimal).

If the number of seconds passed is less than 10, a ‘0’ is added before the number.

The display format will be as follows:

TIME—23:06

Complete the code and show the output waveform and the time on the LCD to the TA.

TA Initials:____________