Benjamin Linam

Bml42@students.uwf.edu

EEL 4744L: Microprocessor Applications Laboratory

Lab 8: Output Compare

4/10/2018

Objective

The object of this lab is to study and understand the output compare functions of the 68HC11.

Introduction/Background/Theory

The 68HC11 has five output compare function pins that are able to be controlled to produce a waveform. Realistically, that wave can be used to control a device such as one involved in the operation of a vehicle. The output compare pins (OC2) can be set with certain commands such as a toggle once the TCNT counter reaches a set TOC2 value. Being able to toggle between 0 V and 5 V easily allows control over a waveform with a certain frequency and duty cycle. This lab requires creation of a program that is controlled by a 3-switch dip switch to output 8 different waves shown in Table 1. To show that the desired waveforms are created, an oscilloscope is to be used to measure frequencies and duty cycles, which are shown in figures 3 through 10.

Table 1: Dip Switch values represented by C2:C0 with the corresponding frequencies and duty cycles.

C2:C0	Frequency	Duty Cycle
000	1kHz	10%
001	1kHz	30%
010	2kHz	40%
011	2kHz	50%
100	2kHz	60%
101	4kHz	70%
110	4kHz	80%
111	4kHz	90%

Procedure

1. Much of the code for this lab requires adjusting binary values being read by accumulators and action registers, so the hexadecimal values saved to the initial variables should be looked at as binary instead of hexadecimal. Figure 1 shows that the first operation is setting Port C, which reads input values from a 3-switch dipswitch, to input by setting pins PC0-PC2 to 0.
2. The next operation is preparing OC2 to toggle after a set interval defined by the HITIME and LOTIME arrays shown in Figure 2. Before toggling the OC2 pin, the requesting input value from the dip switch is read, ANDed with 0's to force any undesired input to 0, and the offset required to produce each specific signal is calculated and stored at TOC2 for the next toggle. Since each high or low time is a 2-byte value, the offset is found by multiplying the dip switch value with 2 (performed by a logical shift left). After a toggle, the TFLG1 is cleared in preparation for the next toggle.
3. Once the wave generation has begun, a loop will cause the wave to alternate between high and low to produce the specified frequency and duty cycle.

```
;lab 8 Benjamin Linam 4/4/18

REGBAS  EQU     $1000   ;BASE ADDRESS OF I/O REG BLOCK
PORTA   EQU     $00     ;OFFSET OF PORTA
PORTC   EQU     $03     ;OFFSET OF PORTC
DDRC    EQU     $07     ;OFFSET OF DATA DIRECTION REG FOR PORT C
TOC2    EQU     $18     ;OFFSET OF TOC2 FROM REGBAS
TCNT    EQU     $0E     ;OFFSET OF TCNT FROM REGBAS
TCTL1   EQU     $20     ;OFFSET OF TCTL1 FROM REGBAS
TFLG1   EQU     $23     ;OFFSET OF TFLG1 FROM REGBAS
TOGGLE  EQU     $40     ;VALUE TO SELECT TOGGLE ACTION OF PIN OC2
OC2     EQU     $40     ;MASK TO SELECT OC2 PIN AND OC2F FLAG
CLEAR   EQU     $40     ;VALUE TO CLEAR OC2F FLAG

        ORG     $B600
        LDX     #REGBAS         ;X -> BASE PORT REGISTER

        LDAA    #$F8            ;SET PORT C AS INPUT
        STAA    DDRC,X          ;       ""
        BSET    PORTA,X OC2     ;SET OC2 PIN TO HIGH
        LDAA    #TOGGLE         ;SELECT TOGGLE AS OUTPUT COMPARE ACTION
        STAA    TCTL1,X         ;       ""

        LDAB    PORTC,X         ;RETRIEVES VALUE AT PORTC AND STORE IN [B]
        ANDB    #$07            ;RETURNS 00000XXX
        LDY     #HITIME         ;SET Y TO HITIME ARRAY
        LSLB                    ;MULTIPLY [B] BY 2, ARRAY HAS 2-BYTE NUMBERS
        ABY                     ;ADD [B] TO Y, Y WILL POINT TO CORRECT HITIME VALUE

        LDD     TCNT,X          ;START OC2 WITH HIGH TIME DELAY
        ADDD    0,Y             ;       ""
        STD     TOC2,X          ;       ""
        LDAA    #CLEAR          ;CLEAR OC2F FLAG
        STAA    TFLG1,X         ;       ""
```

Figure 1: ASM code showcasing code controlling initialization of variables and setting of initial signal generation

```
high    BRCLR   TFLG1,X OC2 high        ;WAIT UNTIL OC2F FLAG SET TO 1

        LDAB    PORTC,X         ;RETRIEVES VALUE AT PORTC AND STORE IN [B]
        ANDB    #$07            ;RETURNS 00000XXX
        LDY     #LOTIME         ;SET Y TO LOTIME ARRAY
        LSLB                    ;MULTIPLY [B] BY 2, ARRAY HAS 2-BYTE NUMBERS
        ABY                     ;ADD [B] TO Y, Y WILL POINT TO CORRECT LOTIME VALUE

        LDD     TOC2,X          ;TOGGLE OC2 PIN AFTER LOW CYCLE
        ADDD    0,Y             ;       ""
        STD     TOC2,X          ;       ""
        LDAA    #CLEAR          ;CLEAR THE OC2F FLAG
        STAA    TFLG1,X         ;       ""
low     BRCLR   TFLG1,X OC2 low         ;WAIT UNTIL OC2F SET TO 1
        LDAA    #CLEAR                  ;CLEAR OC2F FLAG
        STAA    TFLG1,X                 ;       ""

        LDAB    PORTC,X         ;RETRIEVES VALUE AT PORTC AND STORE IN [B]
        ANDB    #$07            ;RETURNS 00000XXX
        LDY     #HITIME         ;SET Y TO HITIME ARRAY
        LSLB                    ;MULTIPLY [B] BY 2, ARRAY HAS 2-BYTE NUMBERS
        ABY                     ;ADD [B] TO Y, Y WILL POINT TO CORRECT HITIME VALUE

        LDD     TOC2,X          ;START THE NEXT OC2 COMPARE OPERATION
        ADDD    0,Y             ;WILL TOGGLE OC2 PIN HIGH
        STD     TOC2,X                  ;       ""
        BRA     high

HITIME  FDB     200,600,400,500,600,350,400,450         ;HIGH TIME VALUES
LOTIME  FDB     1800,1400,600,500,400,150,100,50        ;LOW TIME VALUES
```

Figure 2: ASM code showcasing code controlling dip switch testing and wave generation loop

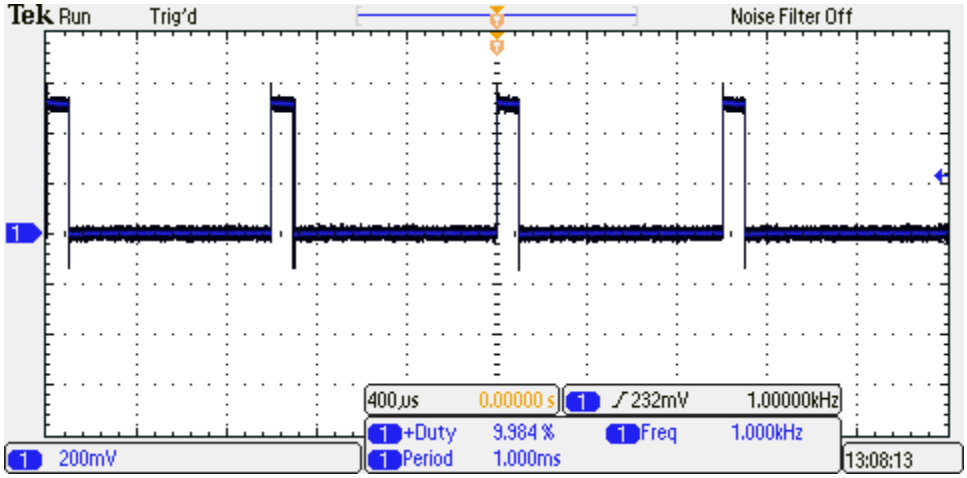

Figure 3: Oscilloscope representation of 1 kHz wave with 10% Duty Cycle

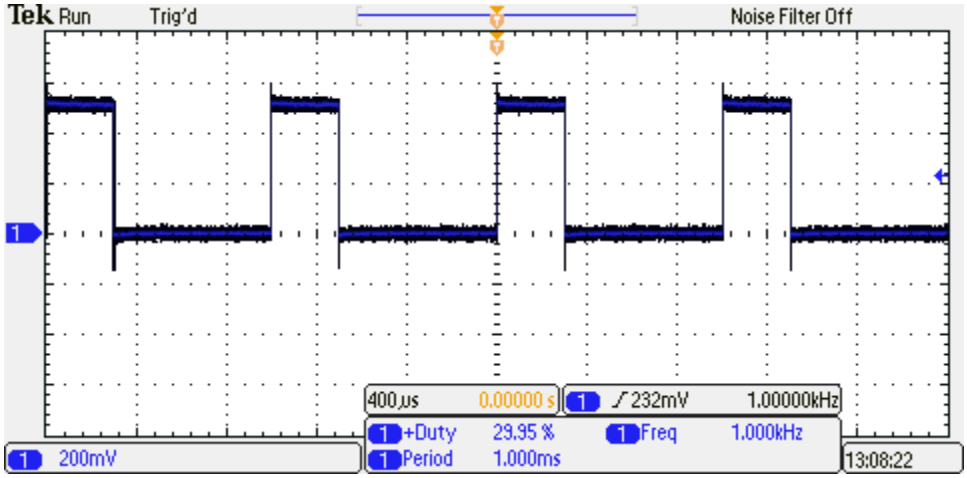

Figure 4: Oscilloscope representation of 1 kHz wave with 30% Duty Cycle

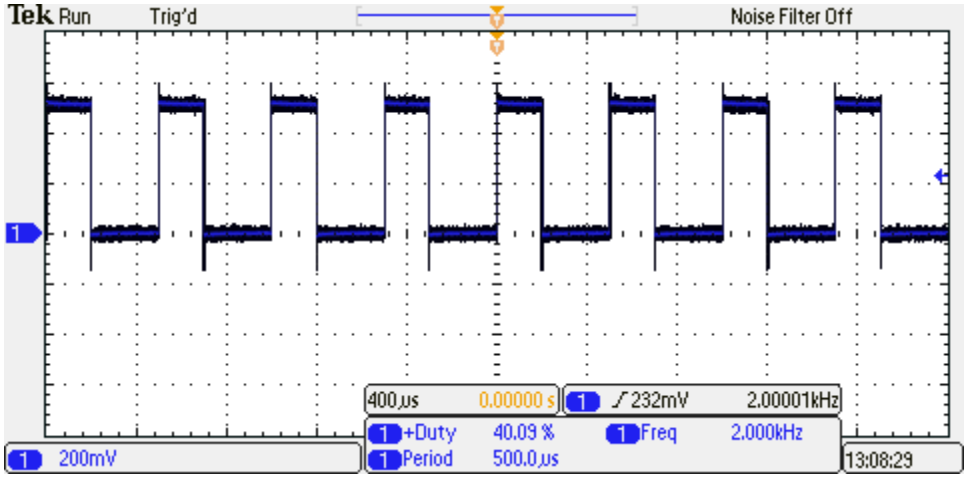

Figure 5: Oscilloscope representation of 2 kHz wave with 40% Duty Cycle

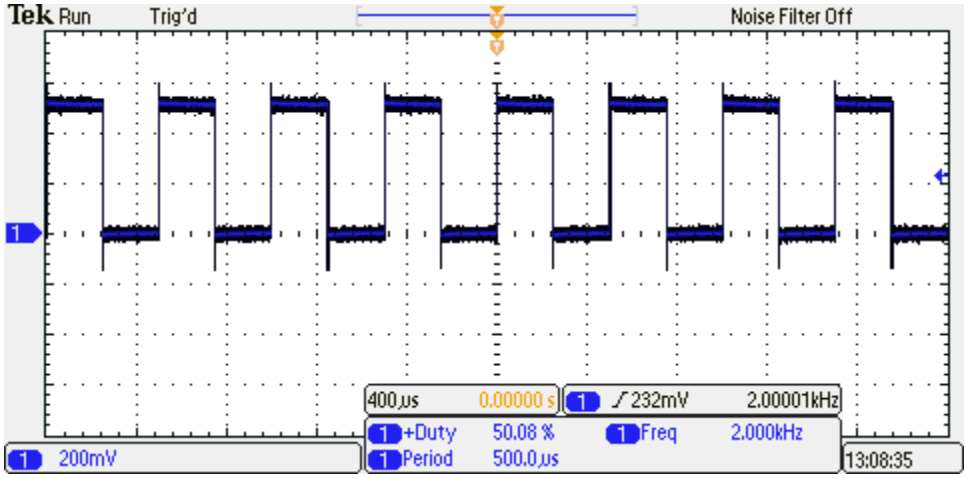

Figure 6: Oscilloscope representation of 2 kHz wave with 50% Duty Cycle

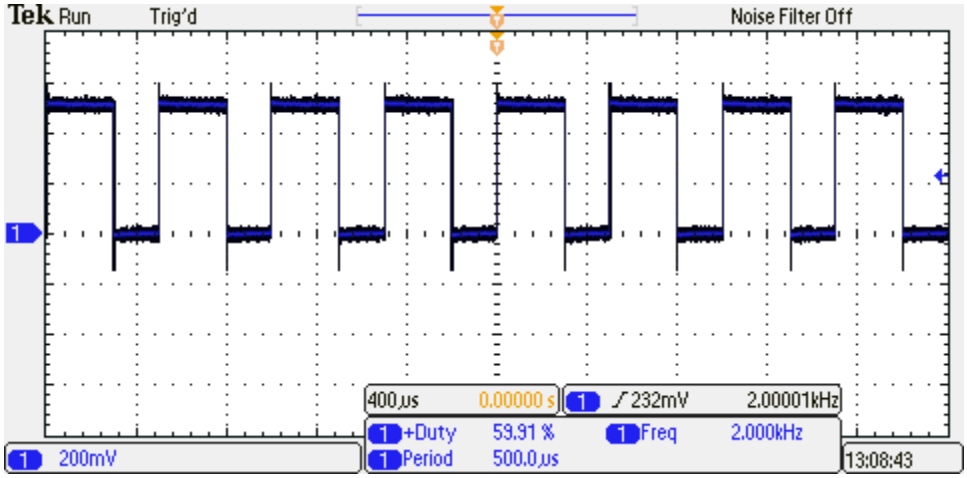

Figure 7: Oscilloscope representation of 2 kHz wave with 60% Duty Cycle

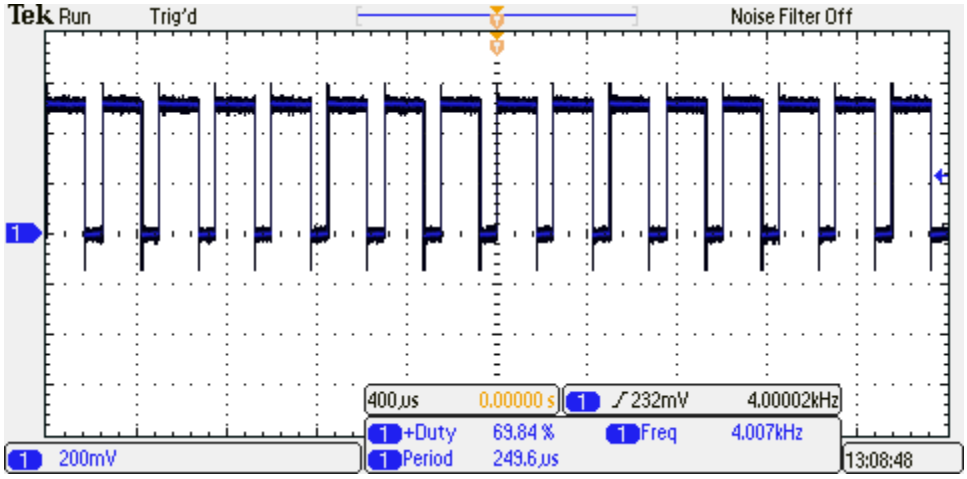

Figure 8: Oscilloscope representation of 4 kHz wave with 70% Duty Cycle

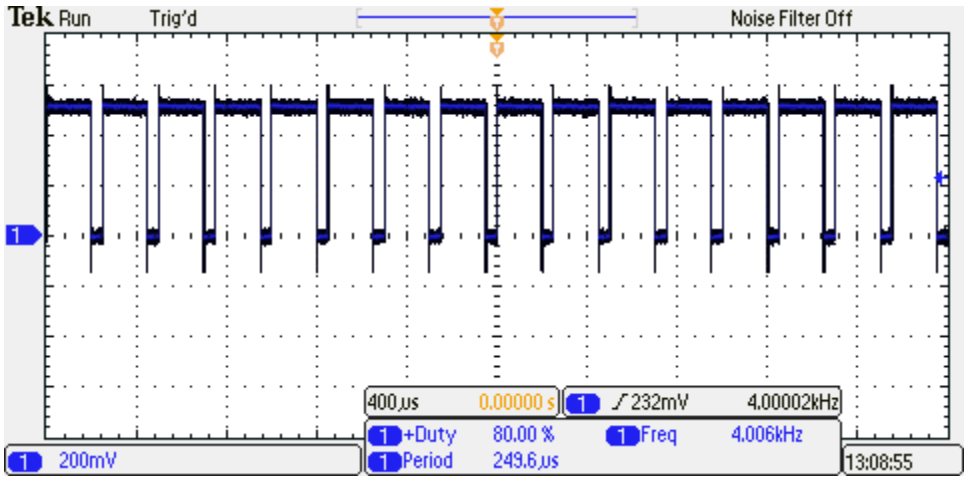

Figure 9: Oscilloscope representation of 4 kHz wave with 80% Duty Cycle

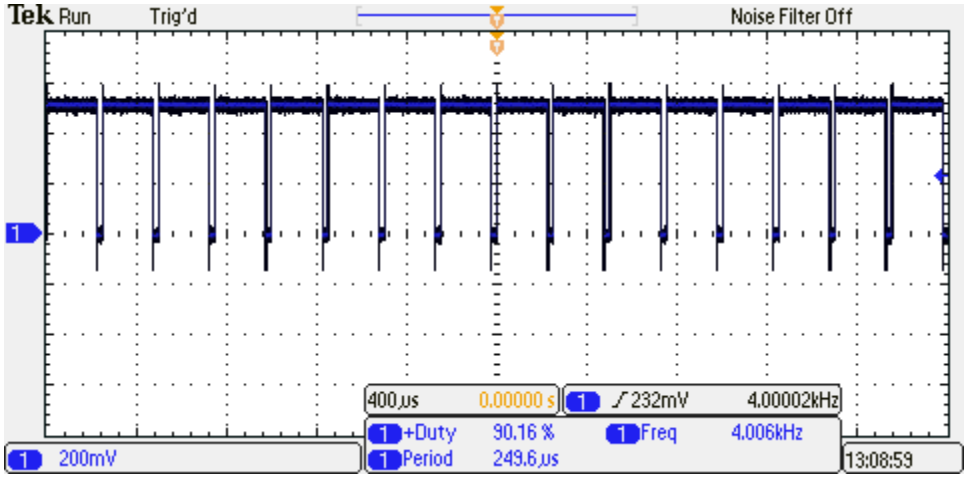

Figure 10: Oscilloscope representation of 4 kHz wave with 90% Duty Cycle

Conclusions

Not very many problems were encountered during the creation and debugging of this program. Initially I created the dip switch checking portion of the code with several branch statements, but I streamlined the code by testing the dip switch values and determining the desired wave just before each toggle was activated. During testing of the code in the lab, the only problem I had was that I forgot to include a pound sign (#) before initializing Port C as well as ANDing the input values.