

# Skyler MacDougall, Matthew Gerace

---

## Homework 6: Due 2/26/2020

---

1. Rewrite the reaction timer game program using only interrupts.

- A pin change interrupt on PD7 is used to start the game. When the active-low button is pressed, the yellow LED turns on after a random delay. When the LED turns on, `TCNT1=0`.
- `INT0` is attached to another active-low button. This is for the reaction button. If the button is pressed before a  $\frac{1}{2}$  second has elapsed, the green LED should light.
- A Timer1 interrupt should be used to count to  $\frac{1}{2}$  second. If  $\frac{1}{2}$  second passes before `INT0`, the red LED should be lit.

✓ Which pins will the 3 LEDs be connected to?

Green = `PC0`. Yellow = `PC1`. Red = `PC2`.

✓ Initialize the pins for the LEDs.

See line 8

✓ Initialize the two interrupt pins as inputs.

See line 5

✓ What mode should Timer1 be in to create a  $\frac{1}{2}$  second interrupt?

CTC, although PWM is also possible.

✓ Initialize for `TCCR1A` and `TCCR1B` for the correct mode.

See lines 11 and 12.

✓ Is a clock prescaler needed for  $\frac{1}{2}$  second interrupt?

Yes.

✓ If so, set it up in `TCCR1B`.

See line 12.

✓ What value goes in `OCR1A` for  $\frac{1}{2}$  second?

See line 14.

✓ Enable the Timer1 interrupt.

See line 18.

✓ Enable the pin change interrupt in the control register.

See line 19.

✓ Enable the external interrupt in the control register.

See line 21.

✓ Set the correct pin for the pin change interrupt in `PCMSK`

See line 20.

- ✓ Set the correct pin for the external interrupt in the `EIMSK`

See line 22.

- ✓ What are the names of the 3 interrupt vectors?

`PCINT2_vect`, `INT0_vect`, and `TIMER1_COMPA_vect`.

```
1 // Homework 6, Problem 1
2
3 void setup()
4 {
5     DDRD &= ~(0x82); // make PD6 and PD7 inputs -- reaction switch and start
switch
6     DDRC |= (1 << 0|1 << 1|1 << 2); //PC0, PC1, and PC2 outputs -- green,
yellow, and red light
7
8     PORTC &= ~(1 << 0|1 << 1|1 << 2); //start all lights off
9
10    // CTC mode timer setups
11    TCCR1A = 0;
12    TCCR1B = 0x0C;
13    TCNT1 = 0;
14    OCR1A = 31249;
15
16    cli(); // turn off global interrupts
17    // set the interrupt flag
18    TIMSK1 = 4;
19    PCICR = 4;
20    PCMSK2 = 0x80;
21    EIMSK = 1;
22    EICRA = 3;
23    sei(); // turn on global interrupts
24 }
25
26 void loop(){while(1)}
27
28 ISR(PCINT2_vect){
29     // wait a random amount of time
30     delay(random());
31
32     // turn the yellow LED on
33     PORTC |= 0x02;
34     //reset the timer
35     TCNT1 = 0;
36 }
37
38 ISR(INT0_vect){
39     PORTC &= ~(0x02); // turn yellow LED off
40     PORTC |= 0x01;    // turn red LED on
41 }
42
43 ISR(TIMER1_COMPA_vect)
44 {
45     // code for servicing the interrupt
46     PORTC &= ~(0x02); // turn yellow LED off
47     PORTC |= 0x04;    // turn red LED on
48 }
```

2. In the previous homework, you determined the constants that control your servo motor to rotate  $180^\circ$ . Starting from that point, complete the following:
  1. Once you've determined the constants that will move the servo a full  $180^\circ$ , modify your program to create a windshield wiper controller. As you know, a windshield wiper moves smoothly across  $180^\circ$ . It doesn't "snap" back to position like the servo did in lab 4. You can achieve a smooth motion by incrementally increasing and decreasing your pulsewidth.
  3. The last step is to add a pushbutton interrupt to start the windshield wiper. You can choose either a pin change or an external interrupt as a source.
  3. Submit a video of your working system (or bring it to me for a demonstration).

```
1 unsigned long servoValue=0, prevValue=0;
2
3 void setup()
4 {
5   pinMode(9,OUTPUT);
6   Serial.begin(9600);
7   TCCR1A=0xB2;
8   TCCR1B=0x1B;
9   ICR1=0x1387;
10 }
11
12 void loop()
13 {
14   delay(10);
15   prevValue = servoValue;
16   while ((servoValue>=525) || (prevValue > servoValue))
17   {
18     if (servoValue > 0){
19       servoValue--;
20       OCR1A=144+servoValue;
21       Serial.println(OCR1A);
22       delay(10);
23     }
24     else{break;}
25   }
26   if (prevValue == servoValue) {servoValue++;}
27   OCR1A=144+servoValue;
28   Serial.println(OCR1A);
29 }
```

[link to video](#)

3. Rewrite Q1 of HW#4, using PWM mode instead of CTC mode.

Because the code is quite lengthy, code modifications can be seen in lines 35, 36, and 44.

```
1 //Note-name/constant conversion
2 #define LA 36363
3 #define LAS 34323
4 #define LB 32396
5 #define LC 30578
6 #define LCS 28861
7 #define LD 27242
```

```
8 #define LDS 25713
9 #define LE 24270
10 #define LF 22908
11 #define LFS 21622
12 #define LG 20408
13 #define LGS 19263
14 #define MA 18182
15 #define MAS 17161
16 #define MB 16198
17 #define MC 15289
18 #define MCS 14430
19 #define MD 13620
20 #define MDS 12856
21 #define ME 12135
22 #define MF 11454
23 #define MFS 10811
24 #define MG 10204
25 #define MGS 9631
26 #define HA 9091
27
28 //Note Length definitions
29 #define QUARTER 400
30 #define EIGHTH 200
31 #define DOTQUARTER 600
32 #define HALF 800
33
34 void setup(){
35     TCCR1A=0b10100011;
36     TCCR1B=0b00011001;
37     TCCR1C=0;
38     TCNT1=0;
39     DDRB|=0x02;
40 }
41
42 void playNote(int note, int time){
43     OCR1A = note;
44     OCR1B = note / 2;
45     delay(time);
46     OCR1A = 0;
47     delay(50);
48 }
49
50 void loop(){
51     playNote(LD, EIGHTH);
52     playNote(LD, EIGHTH);
53     playNote(LG, QUARTER);
54     playNote(LG, EIGHTH);
55     playNote(MA, EIGHTH);
56     playNote(MB, EIGHTH);
57     playNote(LG, EIGHTH);
58     playNote(MB, EIGHTH);
59     playNote(MC, EIGHTH);
60     playNote(MD, QUARTER);
61     playNote(MD, EIGHTH);
62     playNote(MC, EIGHTH);
63     playNote(MB, QUARTER);
64     playNote(MA, EIGHTH);
65     playNote(LG, EIGHTH);
```

```
66 playNote(MA, QUARTER);
67 playNote(MA, QUARTER);
68 playNote(MA, QUARTER);
69 playNote(MA, QUARTER);
70 playNote(MA, EIGHTH);
71 playNote(MB, EIGHTH);
72 playNote(MA, EIGHTH);
73 playNote(LFS, EIGHTH);
74 playNote(LD, QUARTER);
75 playNote(LD, QUARTER);
76 playNote(LG, EIGHTH);
77 playNote(LFS, EIGHTH);
78 playNote(LG, EIGHTH);
79 playNote(MA, EIGHTH);
80 playNote(MB, QUARTER);
81 playNote(MA, EIGHTH);
82 playNote(MA, EIGHTH);
83 playNote(MB, QUARTER);
84 playNote(MC, QUARTER);
85 playNote(MD, DOTQUARTER);
86 playNote(MD, EIGHTH);
87 playNote(MA, QUARTER);
88 playNote(MA, EIGHTH);
89 playNote(MB, EIGHTH);
90 playNote(MA, QUARTER);
91 playNote(LG, EIGHTH);
92 playNote(LG, EIGHTH);
93 playNote(MA, QUARTER);
94 playNote(LG, EIGHTH);
95 playNote(LFS, EIGHTH);
96 playNote(LG, HALF);
97 playNote(MD, HALF);
98 playNote(MB, DOTQUARTER);
99 playNote(MA, EIGHTH);
100 playNote(MB, EIGHTH);
101 playNote(MC, EIGHTH);
102 playNote(MB, EIGHTH);
103 playNote(MA, EIGHTH);
104 playNote(LG, DOTQUARTER);
105 playNote(MA, EIGHTH);
106 playNote(MB, QUARTER);
107 playNote(MB, EIGHTH);
108 playNote(MC, EIGHTH);
109 playNote(MD, QUARTER);
110 playNote(MB, QUARTER);
111 playNote(MA, QUARTER);
112 playNote(MA, EIGHTH);
113 playNote(MB, EIGHTH);
114 playNote(MA, DOTQUARTER);
115 playNote(LD, EIGHTH);
116 playNote(LG, HALF);
117 playNote(LG, DOTQUARTER);
118 playNote(MA, EIGHTH);
119 playNote(MB, QUARTER);
120 playNote(MB, EIGHTH);
121 playNote(MC, EIGHTH);
122 playNote(MD, QUARTER);
123 playNote(MC, EIGHTH);
```

```
124     playNote(MB, EIGHTH);
125     playNote(MA, QUARTER);
126     playNote(MA, QUARTER);
127     playNote(MB, QUARTER);
128     playNote(MB, EIGHTH);
129     playNote(MA, EIGHTH);
130     playNote(LG, QUARTER);
131     playNote(LG, QUARTER);
132     playNote(LG, HALF);
133     delay(1000);
134 }
```