Willis Allstead
CPE 301-1001
Lab #10
November 28, 2016

Assignment Description:

In this lab we were supposed to go off of the work we did in Lab 7, but instead using our own kbhit, getchar, and putchar along with interrupts instead of just basic timers. We were to use the ISR function for the TIMER1 OVF to stop the timer, reload the timer counter, toggle portB.6, and restart the timer. Then we were supposed to explain why the sound we hear is different than when we use a sine wave generator. I chose to implement the sharp notes as well.

Problems Encountered:

The number one problem I had in this lab was changing from just using the timer to using the interrupt and the timer's overflow flag. I decided to put the number of ticks I would use for the timer's counter in a global variable which I change in the switch statement within the loop() function. Then when the ISR is triggered it just uses that global variable to reload counter1. This worked fine but I feel like there is a better (or cleaner) solution.

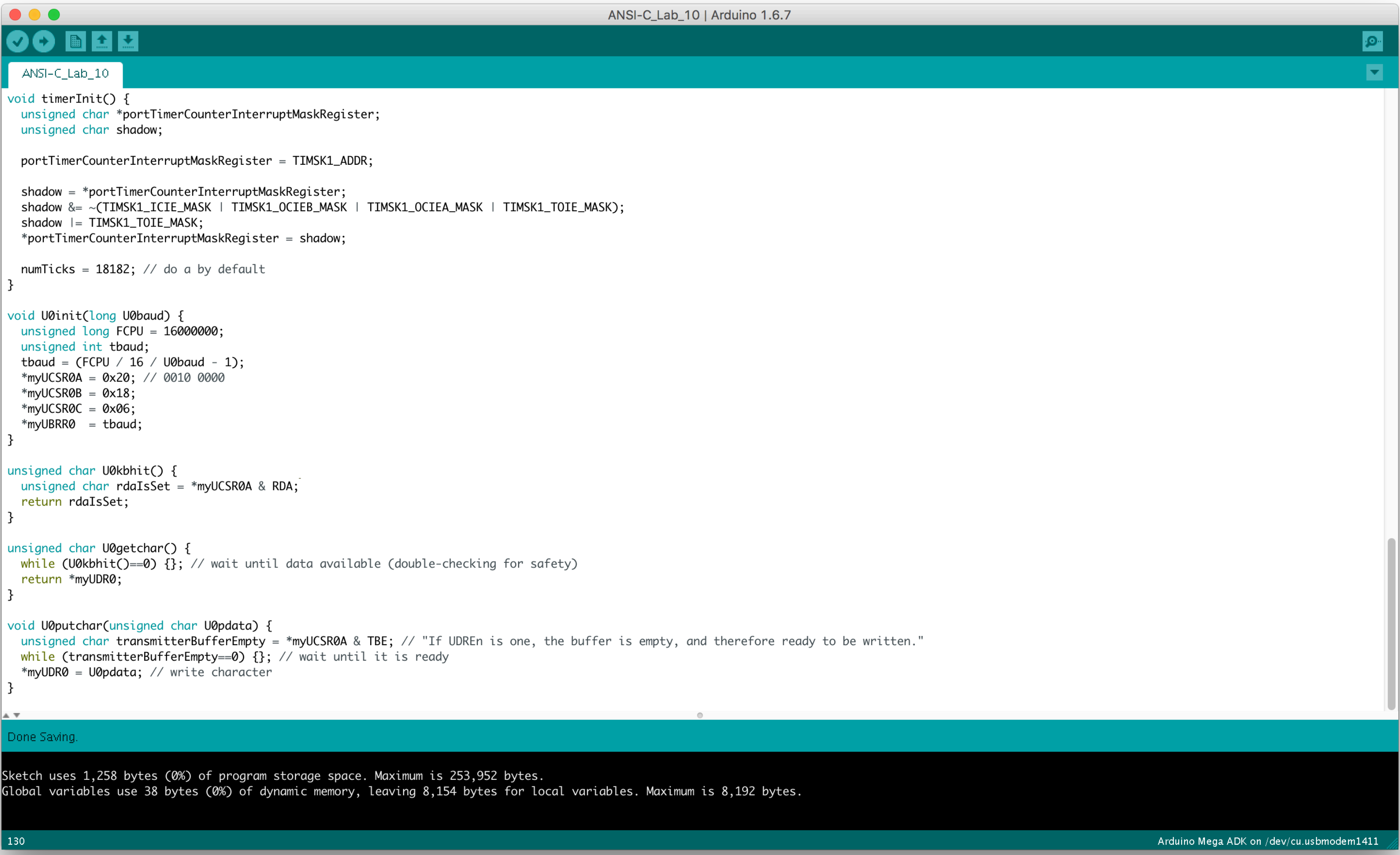
Lessons Learned:

I learned that using interrupts can be very useful in our programs. We can execute actions the moment they occur rather than planning for them ahead of time, which I could see being very helpful in the future of my programming for Arduino.

Description of Completed Lab: (sorry for length of code)

```
#define RDA 0x80 // Received Data Available ~= RXC
#define TBE 0x20 // Transmitter Buffer Empty ~= DRE (Data Register
Empty)
#define TIMSK1_ADDR (unsigned char *) 0x6F
#define TIMSK1_ICIE_MASK  0x20
#define TIMSK1_OCIEB_MASK 0x04
#define TIMSK1_OCIEA_MASK 0x02
#define TIMSK1_TOIE_MASK  0x01

volatile unsigned char *myTCCR1A = (unsigned char *) 0x80; // Timer/
Counter Control Register A
volatile unsigned char *myTCCR1B = (unsigned char *) 0x81; // Timer/
Counter Control Register B
volatile unsigned char *myTCCR1C = (unsigned char *) 0x82; // Timer/
Counter Control Register C
volatile unsigned char *myTIMSK1 = (unsigned char *) 0x6F; // Timer/
Counter Interrupt Mask Register
volatile unsigned int  *myTCNT1  = (unsigned  int *) 0x84; // Timer/
Counter Count Register (low & high)
volatile unsigned char *myTIFR1 =  (unsigned char *) 0x36; // Timer/
Counter Interrupt Flag Register

volatile unsigned char *myUCSR0A = (unsigned char *)0xC0; // USART
MSPIM Control and Status Register A
volatile unsigned char *myUCSR0B = (unsigned char *)0xC1; // USART
MSPIM Control and Status Register B
volatile unsigned char *myUCSR0C = (unsigned char *)0xC2; // USART
MSPIM Control and Status Register C
volatile unsigned int  *myUBRR0  = (unsigned int *) 0xC4; // USART0
Baud Rate Register Low Byte
volatile unsigned char *myUDR0   = (unsigned char *)0xC6; // USART I/O
Data Register

volatile unsigned char *portDDRB = (unsigned char *) 0x24;
volatile unsigned char *portB =    (unsigned char *) 0x25;
volatile unsigned char charRead;
volatile int numTicks;

void timerInit();
void U0init(long U0baud);
unsigned char U0kbhit();
unsigned char U0getchar();
void U0putchar(unsigned char U0pdata);

ISR(TIMER1_OVF_vect) {
  *myTCCR1B &= 0xF8; // stop timer
  *myTCNT1 = (unsigned int) (65536 - numTicks); // reload timer
  *myTCCR1B |= 0x01; // turn timer ON with pre-scalar of 1
  *portB ^= 0x40; // toggle portb
  *myTIFR1 |= 0x01; // clear overflow flag bit (by setting to 1 for
some reason)
}

void setup() {
  /* Initialize Timer1 for NORMAL mode */
  *myTCCR1A = 0;
  *myTCCR1B = 0;
  *myTCCR1C = 0;
  *myTIMSK1 = 0; // Timer 1 should have no interrupts

  *portDDRB |= 0x40;  // Initialize GPIO PortB as output

  U0init(9600); // initialize serial port on USART0
  timerInit();

}

// the loop function runs over and over again forever
void loop() {
  while (U0kbhit()==0){}; // wait for RDA = true
  charRead = U0getchar();    // read character

  switch(charRead) {
    case 'a':
      numTicks = 18182;
      break;
    case 'A': // A#
      numTicks = 17167;
      break;
    case 'b':
      numTicks = 16194;
      break;
    case 'c':
      numTicks = 15296;
      break;
    case 'C': // C#
      numTicks = 14440;
      break;
    case 'd':
      numTicks = 13629;
      break;
    case 'D': // D#
      numTicks = 12821;
      break;
    case 'e':
      numTicks = 12140;
      break;
    case 'f':
      numTicks = 11461;
      break;
    case 'F': // F#
      numTicks = 10811;
      break;
    case 'g':
      numTicks = 10204;
      break;
    case 'G': // G#
      numTicks = 9627;
      break;
    default:
      U0putchar('a');
      numTicks = 18182;
  }
}

void timerInit() {
  unsigned char *portTimerCounterInterruptMaskRegister;
  unsigned char shadow;

  portTimerCounterInterruptMaskRegister = TIMSK1_ADDR;

  shadow = *portTimerCounterInterruptMaskRegister;
  shadow &= ~(TIMSK1_ICIE_MASK | TIMSK1_OCIEB_MASK | TIMSK1_OCIEA_MASK
| TIMSK1_TOIE_MASK);
  shadow |= TIMSK1_TOIE_MASK;
  *portTimerCounterInterruptMaskRegister = shadow;

  numTicks = 18182; // do a by default
}

void U0init(long U0baud) {
  unsigned long FCPU = 16000000;
  unsigned int tbaud;
  tbaud = (FCPU / 16 / U0baud - 1);
  *myUCSR0A = 0x20; // 0010 0000
  *myUCSR0B = 0x18;
  *myUCSR0C = 0x06;
  *myUBRR0  = tbaud;
}

unsigned char U0kbhit() {
  unsigned char rdaIsSet = *myUCSR0A & RDA;
  return rdaIsSet;
}

unsigned char U0getchar() {
  while (U0kbhit()==0) {}; // wait until data available (double-
checking for safety)
  return *myUDR0;
}

void U0putchar(unsigned char U0pdata) {
  unsigned char transmitterBufferEmpty = *myUCSR0A & TBE; // "If UDREn
is one, the buffer is empty, and therefore ready to be written."
  while (transmitterBufferEmpty==0) {}; // wait until it is ready
  *myUDR0 = U0pdata; // write character
}
```

Here is the Compiled code screenshot:



```
void timerInit() {
  unsigned char *portTimerCounterInterruptMaskRegister;
  unsigned char shadow;

  portTimerCounterInterruptMaskRegister = TIMSK1_ADDR;

  shadow = *portTimerCounterInterruptMaskRegister;
  shadow &= ~(TIMSK1_ICIE_MASK | TIMSK1_OCIEB_MASK | TIMSK1_OCIEA_MASK | TIMSK1_TOIE_MASK);
  shadow |= TIMSK1_TOIE_MASK;
  *portTimerCounterInterruptMaskRegister = shadow;

  numTicks = 18182; // do a by default
}

void U0init(long U0baud) {
  unsigned long FCPU = 16000000;
  unsigned int tbaud;
  tbaud = (FCPU / 16 / U0baud - 1);
  *myUCSR0A = 0x20; // 0010 0000
  *myUCSR0B = 0x18;
  *myUCSR0C = 0x06;
  *myUBRR0  = tbaud;
}

unsigned char U0kbhit() {
  unsigned char rdaIsSet = *myUCSR0A & RDA;
  return rdaIsSet;
}

unsigned char U0getchar() {
  while (U0kbhit()==0) {}; // wait until data available (double-checking for safety)
  return *myUDR0;
}

void U0putchar(unsigned char U0pdata) {
  unsigned char transmitterBufferEmpty = *myUCSR0A & TBE; // "If UDREn is one, the buffer is empty, and therefore ready to be written."
  while (transmitterBufferEmpty==0) {}; // wait until it is ready
  *myUDR0 = U0pdata; // write character
}
```

```
Done Saving.

Sketch uses 1,258 bytes (0%) of program storage space. Maximum is 253,952 bytes.
Global variables use 38 bytes (0%) of dynamic memory, leaving 8,154 bytes for local variables. Maximum is 8,192 bytes.

130                                                        Arduino Mega ADK on /dev/cu.usbmodem1411
```

As you can see, the code compiled and ran as I wanted (most of the time). I have a feeling that the alligator wires I was using were not working correctly because the volume of the sound was lower than in project7. Still, the frequencies were correct.

As to why the sound produced by my code is different than that of the sine wave generator, thats because I am producing square waves. Square waves have a more harsh sound to them because of their nature, going from the extreme of 0 to the extreme of 1 with no in-between. A type of wave that sounds more near that of the sine is the triangle, which has a slope of 1 from 0 to 1 and a slope of -1 from 1 to 0.