# Spring 2021 – ECE 487/587 Lab #5/#6 Grading Sheet

Name: ______________________________ CWID: _______________

Functionality (487 – 170 points; 587 – 270 points):

1) Timing – individual and average
2) Structure of the polling loop
3) Operation of potentiometer (full range of digital values are reported)
4) Format of output to the screen
5) Use of required techniques for each user input option
6) Student's understanding of code and hardware
7) Error checking user input
8) 4 second watchdog timeout
9) Cumulative stats (ECE 587 students only – 50 points)
10) Countdown timer (ECE 587 students only – 50 points)

Good Programming Practices (30 points):

1) Comments
2) Indentation
3) Good/meaningful variable names
4) Minimum/wise usage of global variables; unnecessary use of variables
5) Inefficient code
6) Structure of ISR

```
/*************************************************************************
 * lab_5_6
 * Created by Matt Mason,
 * CWID 11800439
 *************************************************************************/

#include <avr/wdt.h>

// potentiometer pin
#define POT A5

// ISR variables
volatile bool newValue = false;
volatile int value;
volatile unsigned long endTime;

/*************************************************************************
 * Function:        printConversion
 * Parameters:      int number - the conversion number
 *                  int value - the digital result of the conversion
 *                  unsigned long time - the conversion time
 * Return value:    none
 * Purpose:         Prints the required info for an individual conversion to
 *                  the serial terminal, nicely formatted.
 *************************************************************************/
void printConversion(int number, int value, unsigned long time)
{
    // print # and pad with space
    if (number < 10)
        Serial.print(" ");
    Serial.print("#");
    Serial.print(number);
    // print digital value in hex and pad with zeroes
    Serial.print(": digital value = ");
    if (value < 0x010)
        Serial.print("0");
    if (value < 0x100)
        Serial.print("0");
    Serial.print(value, HEX);
    // print conversion time
    Serial.print(", conversion time = ");
    Serial.print(time);
    Serial.println(" us");
}

/*************************************************************************
 * Function:        initializeADC
 * Parameters:      uint8_t channel - the desired input channel (0-15)
 * Return value:    none
 * Purpose:         Resets and initializes the ADC using direct port
 *                  manipulation.
 *************************************************************************/
void initializeADC(uint8_t channel)
{
    // reset ADC
    ADCSRA = 0x00;
    ADCSRB = 0x00;
    ADMUX  = 0x00;
    // set prescaler to 128
    ADCSRA |= 0x07;
    // set voltage reference to AVCC
    bitSet(ADMUX, REFS0);
    // set input channel
    ADMUX  |= channel & 0x07;
    ADCSRB |= channel & 0x08;
    // turn on ADC
    bitSet(ADCSRA, ADEN);
    // initialize ADC by doing first conversion
    bitSet(ADCSRA, ADSC);
    while(bitRead(ADCSRA, ADSC));
}

```

```
/**************************************************************************
 * Function:        doConversionsA
 * Parameters:      none
 * Return value:    none
 * Purpose:         Performs 30 A->D conversions for the POT analog input,
 *                  using analogRead(). Outputs the hex value and the
 *                  conversion time for each, as well as the average
 *                  conversion time at the end.
 **************************************************************************/
void doConversionsA()
{
    unsigned long totalTime = 0;
    unsigned long startTime, conversionTime;
    // do 30 conversions using analogRead()
    for (int i = 0; i < 30; i++)
    {
        // record start time
        startTime = micros();
        // do conversion
        int value = analogRead(POT);
        // calculate elapsed time
        conversionTime = micros() - startTime;
        // display conversion results
        printConversion(i + 1, value, conversionTime);
        // sum all conversion times
        totalTime += conversionTime;
    }
    // calculate average conversion time
    float averageTime = totalTime / 30.0f;
    // display average conversion time
    Serial.print("analogRead() avg conversion time = ");
    Serial.print(averageTime, 2);
    Serial.println(" us\n");
    // disregard any received input by clearing serial buffer
    while (Serial.read() != -1);
}

/**************************************************************************
 * Function:        doConversionsB
 * Parameters:      none
 * Return value:    none
 * Purpose:         Performs 30 A->D conversions for the POT analog input,
 *                  using polling and direct port manipulation. Outputs the
 *                  hex value and the conversion time for each, as well as
 *                  the average conversion time at the end.
 **************************************************************************/
void doConversionsB()
{
    unsigned long totalTime = 0;
    unsigned long startTime, conversionTime;
    // initialize ADC using port manipulation
    initializeADC(POT - A0);
    // do 30 conversions using polling
    for (int i = 0; i < 30; i++)
    {
        // record start time
        startTime = micros();
        // start conversion
        bitSet(ADCSRA, ADSC);
        // wait for conversion to finish
        while(bitRead(ADCSRA, ADSC));
        // retrieve conversion value
        int value = ADC;
        // calculate elapsed time
        conversionTime = micros() - startTime;
        // display conversion results
        printConversion(i + 1, value, conversionTime);
        // sum all conversion times
        totalTime += conversionTime;
    }
    // calculate average conversion time
    float averageTime = totalTime / 30.0f;
    // display average conversion time
    Serial.print("polling avg conversion time = ");
    Serial.print(averageTime, 2);
    Serial.println(" us\n");
    // disregard any received input by clearing serial buffer
    while (Serial.read() != -1);
}
```

```

/*****************************************************************************
 * Function:        doConversionsC
 * Parameters:      none
 * Return value:    none
 * Purpose:         Performs 30 A->D conversions for the POT analog input,
 *                  using interrupts and direct port manipulation. Outputs the
 *                  hex value and the conversion time for each, as well as
 *                  the average conversion time at the end.
 ******************************************************************************/
void doConversionsC()
{
    unsigned long totalTime = 0;
    unsigned long startTime, conversionTime;
    // initialize ADC using port manipulation
    initializeADC(POT - A0);
    // enable 'conversion complete' interrupt
    bitSet(ADCSRA, ADIE);
    // record start time
    startTime = micros();
    // start first conversion
    bitSet(ADCSRA, ADSC);
    // do 30 conversions using interrupts
    bool converting = true;
    int i = 0;
    while (converting)
    {
        if (newValue)
        {
            newValue = false;
            // calculate elapsed time
            conversionTime = endTime - startTime;
            // display conversion results
            printConversion(i + 1, value, conversionTime);
            // sum all conversion times
            totalTime += conversionTime;
            // increment conversion counter
            i++;
            // start next conversion if more to go
            if (i < 30)
            {
                // record start time
                startTime = micros();
                // start next conversion
                bitSet(ADCSRA, ADSC);
            }
            else // done
                converting = false;
        }
        // foreground app runs here
    }
    // disable 'conversion complete' interrupt
    bitClear(ADCSRA, ADIE);
    // calculate average conversion time
    float averageTime = totalTime / 30.0f;
    // display average conversion time
    Serial.print("interrupt-driven avg conversion time = ");
    Serial.print(averageTime, 2);
    Serial.println(" us\n");
    // disregard any received input by clearing serial buffer
    while (Serial.read() != -1);
}

/*****************************************************************************
 * Function:        ADC_vect ISR
 * Parameters:      none
 * Return value:    none
 * Purpose:         Triggered by ADC conversion completion. Reads the
 *                  conversion result, records the conversion end time, and
 *                  sets a flag to tell foreground app new data is available.
 ******************************************************************************/
ISR (ADC_vect)
{
    value = ADC;
    endTime = micros();
    newValue = true;
}

```

```
void setup()
{
    // setup input pin
    pinMode(POT, INPUT);

    // open serial connection
    Serial.begin(9600);
    Serial.println("\nlab_5_6 – Board Reset");

    // enable watchdog timer with a 4-second timeout
    wdt_enable(WDTO_4S);
}

void loop()
{
    // display prompt
    Serial.println("Select a type of conversion to perform:");
    Serial.print("'a' for analogRead(), 'b' for polling, 'c' for interrupts > ");
    // refresh watchdog
    wdt_reset();

    // loop condition variable
    bool waiting = true;
    // user input string
    String input = "";
    while (waiting)
    {
        // read serial into input string until buffer is empty or newline received
        char c = 0;
        while (Serial.available())
        {
            c = (char)Serial.read();
            if (c == '\n')
                break;
            input += c;
        }
        // if last character received was a newline, user input is ready
        if (c == '\n')
            waiting = false;
    }

    // refresh watchdog and respond to user input
    wdt_reset();
    Serial.println(input);
    if (input.equals("a"))
    {
        Serial.println("\nStarting a set of conversions using analogRead():");
        doConversionsA();
    }
    else if (input.equals("b"))
    {
        Serial.println("\nStarting a set of conversions using polling:");
        doConversionsB();
    }
    else if (input.equals("c"))
    {
        Serial.println("\nStarting a set of conversions using interupts:");
        doConversionsC();
    }
    else // invalid input
    {
        // display error message
        Serial.println("Invalid input!!");
    }
}
```