

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

```

1  /*****
2  * lab_5_6
3  * Created by Matt Mason,
4  * CWID 11800439
5  *****/
6
7  #include <avr/wdt.h>
8
9  // potentiometer pin
10 #define POT A5
11
12 // ISR variables
13 volatile bool newValue = false;
14 volatile int value;
15 volatile unsigned long endTime;
16
17 /*****
18 * Function:      printConversion
19 * Parameters:    int number — the conversion number
20 *                int value — the digital result of the conversion
21 *                unsigned long time — the conversion time
22 * Return value:  none
23 * Purpose:       Prints the required info for an individual conversion to
24 *                the serial terminal, nicely formatted.
25 *****/
26 void printConversion(int number, int value, unsigned long time)
27 {
28     // print # and pad with space
29     if (number < 10)
30         Serial.print(" ");
31     Serial.print("#");
32     Serial.print(number);
33     // print digital value in hex and pad with zeroes
34     Serial.print(": digital value = ");
35     if (value < 0x010)
36         Serial.print("0");
37     if (value < 0x100)
38         Serial.print("0");
39     Serial.print(value, HEX);
40     // print conversion time
41     Serial.print(", conversion time = ");
42     Serial.print(time);
43     Serial.println(" us");
44 }
45
46 /*****
47 * Function:      initializeADC
48 * Parameters:    uint8_t channel — the desired input channel (0–15)
49 * Return value:  none
50 * Purpose:       Resets and initializes the ADC using direct port
51 *                manipulation.
52 *****/
53 void initializeADC(uint8_t channel)
54 {
55     // reset ADC
56     ADCSRA = 0x00;
57     ADCSRB = 0x00;
58     ADMUX = 0x00;
59     // set prescaler to 128
60     ADCSRA |= 0x07;
61     // set voltage reference to AVCC
62     bitSet(ADMUX, REFS0);
63     // set input channel
64     ADMUX |= channel & 0x07;
65     ADCSRB |= channel & 0x08;
66     // turn on ADC
67     bitSet(ADCSRA, ADEN);
68     // initialize ADC by doing first conversion
69     bitSet(ADCSRA, ADSC);
70     while(bitRead(ADCSRA, ADSC));
71 }
72

```

```

73 /*****
74 * Function:      doConversionsA
75 * Parameters:    none
76 * Return value:  none
77 * Purpose:       Performs 30 A->D conversions for the POT analog input,
78 *               using analogRead(). Outputs the hex value and the
79 *               conversion time for each, as well as the average
80 *               conversion time at the end.
81 *****/
82 void doConversionsA()
83 {
84     unsigned long totalTime = 0;
85     unsigned long startTime, conversionTime;
86     // do 30 conversions using analogRead()
87     for (int i = 0; i < 30; i++)
88     {
89         // record start time
90         startTime = micros();
91         // do conversion
92         int value = analogRead(POT);
93         // calculate elapsed time
94         conversionTime = micros() - startTime;
95         // display conversion results
96         printConversion(i + 1, value, conversionTime);
97         // sum all conversion times
98         totalTime += conversionTime;
99     }
100    // calculate average conversion time
101    float averageTime = totalTime / 30.0f;
102    // display average conversion time
103    Serial.print("analogRead() avg conversion time = ");
104    Serial.print(averageTime, 2);
105    Serial.println(" us\n");
106    // disregard any received input by clearing serial buffer
107    while (Serial.read() != -1);
108 }
109
110 /*****
111 * Function:      doConversionsB
112 * Parameters:    none
113 * Return value:  none
114 * Purpose:       Performs 30 A->D conversions for the POT analog input,
115 *               using polling and direct port manipulation. Outputs the
116 *               hex value and the conversion time for each, as well as
117 *               the average conversion time at the end.
118 *****/
119 void doConversionsB()
120 {
121     unsigned long totalTime = 0;
122     unsigned long startTime, conversionTime;
123     // initialize ADC using port manipulation
124     initializeADC(POT - A0);
125     // do 30 conversions using polling
126     for (int i = 0; i < 30; i++)
127     {
128         // record start time
129         startTime = micros();
130         // start conversion
131         bitSet(ADCSRA, ADSC);
132         // wait for conversion to finish
133         while(bitRead(ADCSRA, ADSC));
134         // retrieve conversion value
135         int value = ADC;
136         // calculate elapsed time
137         conversionTime = micros() - startTime;
138         // display conversion results
139         printConversion(i + 1, value, conversionTime);
140         // sum all conversion times
141         totalTime += conversionTime;
142     }
143    // calculate average conversion time
144    float averageTime = totalTime / 30.0f;
145    // display average conversion time
146    Serial.print("polling avg conversion time = ");
147    Serial.print(averageTime, 2);
148    Serial.println(" us\n");
149    // disregard any received input by clearing serial buffer
150    while (Serial.read() != -1);
151 }

```

```

152
153 /*****
154  * Function:      doConversionsC
155  * Parameters:    none
156  * Return value:  none
157  * Purpose:      Performs 30 A->D conversions for the POT analog input,
158  *               using interrupts and direct port manipulation. Outputs the
159  *               hex value and the conversion time for each, as well as
160  *               the average conversion time at the end.
161  *****/
162 void doConversionsC()
163 {
164     unsigned long totalTime = 0;
165     unsigned long startTime, conversionTime;
166     // initialize ADC using port manipulation
167     initializeADC(POT - A0);
168     // enable 'conversion complete' interrupt
169     bitSet(ADCSRA, ADIE);
170     // record start time
171     startTime = micros();
172     // start first conversion
173     bitSet(ADCSRA, ADSC);
174     // do 30 conversions using interrupts
175     bool converting = true;
176     int i = 0;
177     while (converting)
178     {
179         if (newValue)
180         {
181             newValue = false;
182             // calculate elapsed time
183             conversionTime = endTime - startTime;
184             // display conversion results
185             printConversion(i + 1, value, conversionTime);
186             // sum all conversion times
187             totalTime += conversionTime;
188             // increment conversion counter
189             i++;
190             // start next conversion if more to go
191             if (i < 30)
192             {
193                 // record start time
194                 startTime = micros();
195                 // start next conversion
196                 bitSet(ADCSRA, ADSC);
197             }
198             else // done
199                 converting = false;
200         }
201         // foreground app runs here
202     }
203     // disable 'conversion complete' interrupt
204     bitClear(ADCSRA, ADIE);
205     // calculate average conversion time
206     float averageTime = totalTime / 30.0f;
207     // display average conversion time
208     Serial.print("interrupt-driven avg conversion time = ");
209     Serial.print(averageTime, 2);
210     Serial.println(" us\n");
211     // disregard any received input by clearing serial buffer
212     while (Serial.read() != -1);
213 }
214
215 /*****
216  * Function:      ADC_vect ISR
217  * Parameters:    none
218  * Return value:  none
219  * Purpose:      Triggered by ADC conversion completion. Reads the
220  *               conversion result, records the conversion end time, and
221  *               sets a flag to tell foreground app new data is available.
222  *****/
223 ISR (ADC_vect)
224 {
225     value = ADC;
226     endTime = micros();
227     newValue = true;
228 }
229

```

```

230 void setup()
231 {
232     // setup input pin
233     pinMode(POT, INPUT);
234
235     // open serial connection
236     Serial.begin(9600);
237     Serial.println("\nlab_5_6 - Board Reset");
238
239     // enable watchdog timer with a 4-second timeout
240     wdt_enable(WDTO_4S);
241 }
242
243 void loop()
244 {
245     // display prompt
246     Serial.println("Select a type of conversion to perform:");
247     Serial.print("'a' for analogRead(), 'b' for polling, 'c' for interrupts > ");
248     // refresh watchdog
249     wdt_reset();
250
251     // loop condition variable
252     bool waiting = true;
253     // user input string
254     String input = "";
255     while (waiting)
256     {
257         // read serial into input string until buffer is empty or newline received
258         char c = 0;
259         while (Serial.available())
260         {
261             c = (char)Serial.read();
262             if (c == '\n')
263                 break;
264             input += c;
265         }
266         // if last character received was a newline, user input is ready
267         if (c == '\n')
268             waiting = false;
269     }
270
271     // refresh watchdog and respond to user input
272     wdt_reset();
273     Serial.println(input);
274     if (input.equals("a"))
275     {
276         Serial.println("\nStarting a set of conversions using analogRead()");
277         doConversionsA();
278     }
279     else if (input.equals("b"))
280     {
281         Serial.println("\nStarting a set of conversions using polling");
282         doConversionsB();
283     }
284     else if (input.equals("c"))
285     {
286         Serial.println("\nStarting a set of conversions using interrupts");
287         doConversionsC();
288     }
289     else // invalid input
290     {
291         // display error message
292         Serial.println("Invalid input!!");
293     }
294 }

```