

```
1  /*
2   * FileName: main.c
3   * Version: 1
4   *
5   * Created: 10/11/2022 5:36:00 PM
6   * Author: Ethan Zeronik
7   *
8   * Operations: set the button IO
9   *
10  * Hardware:
11  *   Atmega2560          micro controller
12  *   PORTA.0             mode switch
13  *   PORTA.4             start pushbutton
14  *   PORTA.5             stop pushbutton
15  */
16
17 /* NOTE: Includes */
18 #include <avr/io.h>
19
20 #include "Debugger.h"
21
22 /* NOTE: Custom Macros */
23 // TODO: None
24
25 /* NOTE: Global Variables */
26 // TODO: None
27
28 /* NOTE: Function prototypes */
29 // inits IO ports
30 void IO_init(void);
31
32 /* NOTE: Application implementation */
33 // the main loop of the function, provided to us
34 int main(void)
35 {
36     IO_init();
37
38     initDebug();
39
40     while(1)
41     {
42     }
43 }
44
45 /* NOTE: Function implementations */
46 void IO_init(void)
47 {
48     // set port A as all inputs
49     DDRA = 0x00;
50     // turn all pullup resisistors
51     PORTA = 0xFF;
52 }
```

```
1  /*
2   * FileName: main.c
3   * Version: 1
4   *
5   * Created: 10/18/2022 8:38:44 PM
6   * Author: Ethan Zeronik
7   *
8   * Operations: Blink and LED every 500ms
9   *
10  * Hardware:
11  *   Atmega2560          micro controller
12  *   PORTB.7            LED13 active high
13  */
14
15  /* NOTE: Includes */
16  #include <avr/io.h>
17
18  #include "Delay.h"
19  #include "Debugger.h"
20
21  /* NOTE: Custom Macros */
22  // TODO: None
23
24  /* NOTE: Global Variables */
25  // TODO: None
26
27  /* NOTE: Function prototypes */
28  // inits IO ports
29  void IO_init(void);
30
31  /* NOTE: Application implementation */
32  // the main loop of the function, provided to us
33  int main(void)
34  {
35      DLY_init();
36
37      IO_init();
38
39      initDebug();
40
41      while(1)
42      {
43          DLY_ms(500);
44
45          PORTB = ~(0x80 & PORTB);
46      }
47  }
48
49  /* NOTE: Function implementations */
50  void IO_init(void)
51  {
52      // set port B.7 as an output
53      DDRB = 0x80;
54      PORTB = 0x00;
55  }
56
```

```
1  /*
2   * FileName: main.c
3   * Version: 1
4   *
5   * Created: 10/18/2022 8:55:13 PM
6   * Author: Ethan Zeronik
7   *
8   * Operations: make a basic PWM controller
9   *
10  * Hardware:
11  *   Atmega2560          micro controller
12  *   PORTA.4             start pushbutton
13  *   PORTA.5             stop pushbutton
14  *   PORTB.5             PWM output
15  */
16
17 /* NOTE: Includes */
18 #include <avr/io.h>
19
20 #include "PulseWidthModulation.h"
21 #include "Delay.h"
22 #include "Debugger.h"
23
24 /* NOTE: Custom Macros */
25 // TODO: None
26
27 /* NOTE: Global Variables */
28 // TODO: None
29
30 /* NOTE: Function prototypes */
31 // inits IO ports
32 void IO_init(void);
33 // makes a ramp
34 void rampUpDelayWithSteps(double start, double end, uint16_t duration, uint8_t stepCount);
35
36 /* NOTE: Application implementation */
37 // the main loop of the function, provided to us
38 int main(void)
39 {
40     IO_init();
41     PWM_init();
42     DLY_init();
43
44     initDebug();
45
46     while(1)
47     {
48         while((PINA & 0x10) == 0)
49         {
50             // do nothing
51         }
52
53         PWM_enable();
54
55         rampUpDelayWithSteps(.1, .5, 8000, 5);
56
57         while((PINA & 0x20) == 0)
```

```
58     {
59         // do nothing
60     }
61
62     PWM_disable();
63 }
64 }
65
66 /* NOTE: Function implementations */
67 void IO_init(void)
68 {
69     // set port B.7 as an output
70     DDRA = ~0x30;
71     PORTA = 0x30;
72 }
73
74 void rampUpDelayWithSteps(double start, double end, uint16_t duration, uint8_t stepCount)
75 {
76     double stepIncrement = (end - start) / (stepCount - 1);
77
78     for(size_t i = 0; i < stepCount; i++)
79     {
80         DLY_ms(duration / stepCount);
81         PWM_dutyCycle((stepIncrement * i) + start);
82     }
83 }
```

```
1  /*
2   * FileName: main.c
3   * Version: 1
4   *
5   * Created: 10/19/2022 12:21:39 PM
6   * Author: Ethan Zeronik
7   *
8   * Operations: ADC to PWM
9   *
10  * Hardware:
11  *   Atmega2560          micro controller
12  *   PORTB.5             PWM output
13  *   PORTB.7             LED13 active high
14  *   PORTA.0             mode switch
15  *   PORTA.4             start pushbutton
16  *   PORTA.5             stop pushbutton
17  *   PORTF.0             pot in
18  */
19
20 /* NOTE: Includes */
21 #include <avr/io.h>
22
23 #include "PulseWidthModulation.h"
24 #include "AnalogToDigital.h"
25 #include "Delay.h"
26 #include "Debugger.h"
27
28 /* NOTE: Custom Macros */
29 // TODO: None
30
31 /* NOTE: Global Variables */
32 // TODO: None
33
34 /* NOTE: Function prototypes */
35 // inits IO ports
36 void IO_init(void);
37
38 /* NOTE: Application implementation */
39 // the main loop of the function, provided to us
40 int main(void)
41 {
42     IO_init();
43     PWM_init();
44     DLY_init();
45     ADC_init();
46
47     initDebug();
48
49     while(1)
50     {
51         if(PINA & 0x01)
52         {
53             PWM_enable();
54
55             // set pwm to pot adc
56             PWM_dutyCycle(ADC_getTenBitValue(0x00));
57         }
58     }
59 }
```

```
58     else
59     {
60         PWM_disable();
61
62         // toggle led every 500 ms
63         DLY_ms(500);
64         PORTB = ~PORTB & 0x80;
65     }
66 }
67 }
68
69 /* NOTE: Function implementations */
70 void IO_init(void)
71 {
72     // set port B.7 as an output
73     DDRB = 0x80;
74     PORTB = 0x00;
75
76     // set port A.0 as an input
77     DDRA = 0x00;
78     PORTA = 0x01;
79
80     // set port F.0 as an input
81     DDRF = 0x00;
82     PORTF = 0x00;
83 }
84
```

```
1  /*
2   * FileName: Delay.h
3   * Version: 1
4   *
5   * Created: 10/18/2022 7:26 PM
6   * Author: Ethan Zeronik
7   *
8   * Operations: header for the delay submodule
9   */
10
11 #ifndef Delay_h_INCLUDED
12 #define Delay_h_INCLUDED
13
14 #if defined(__cplusplus)
15 extern "C" {
16 #endif
17
18 #pragma message("WARNING: this module uses Timer 0 for delays")
19
20 #include "stdio.h"
21
22 #define F_CPU 16000000UL
23
24 /* NOTE: Custom Types */
25 // TODO: None
26
27 /* NOTE: Function prototypes */
28 // init registers for delay
29 void DLY_init(void);
30 // delay for an amount of ms
31 void DLY_ms(double ms);
32
33 #if defined(__cplusplus)
34 } /* extern "C" */
35 #endif
36
37 #endif // Delay_h_INCLUDED
```

```
60     OCR0A = (((1 / 1000.0) * F_CPU) / 1024);
61
62     for(size_t i = 0; i < ms; i++)
63     {
64         // prescalar of 1024
65         TCCR0B = 0x05;
66
67         while((TIFR0 & (1 << OCF0A)) == 0)
68         {
69             // do nothing
70         }
71
72         // stop the timer
73         TCCR0B = 0x00;
74         // clear the overflow flag
75         TIFR0 |= (1 << OCF0A);
76         TCNT0 = 0;
77     }
78
79     OCR0A = 0;
```



```
1  /*
2   * FileName: PulseWidthModulation.h
3   * Version: 1
4   *
5   * Created: 10/18/2022 10:14 PM
6   * Author: Ethan Zeronik
7   *
8   * Operations: header for the pwm submodule
9   */
10
11 #ifndef PulseWidthModulation_h_INCLUDED
12 #define PulseWidthModulation_h_INCLUDED
13
14 #if defined(__cplusplus)
15 extern "C" {
16 #endif
17
18 #pragma message("WARNING: this module uses Timer 1 for PWM")
19
20 #include "stdio.h"
21
22 #define F_CPU 16000000UL
23
24 /* NOTE: Custom Types */
25 // TODO: None
26
27 /* NOTE: Function prototypes */
28 // init registers for PWM
29 void PWM_init(void);
30 // makes a pwm for a given duty cycle
31 void PWM_dutyCycle(double percent);
32 // enable pwm output
33 void PWM_enable(void);
34 // disable pwm output
35 void PWM_disable(void);
36
37 #if defined(__cplusplus)
38 } /* extern "C" */
39 #endif
40
41 #endif // PulseWidthModulation_h_INCLUDED
```

```
1  /*
2   * FileName: PulseWidthModulation.c
3   * Version: 1
4   *
5   * Created: 10/18/2022 10:14 PM
6   * Author: Ethan Zeronik
7   *
8   * Operations: create a custom pwm function
9   */
10
11 /* NOTE: Includes */
12 #include "PulseWidthModulation.h"
13
14 #include <avr/io.h>
15
16 /* NOTE: Local declarations */
17 // TODO: None
18
19 /* NOTE: Global Variables */
20 // TODO: None
21
22 /* NOTE: Local function implementations */
23 void PWM_init(void)
24 {
25     // set up PORTB.5 as an output and 0V
26     DDRB |= 0x20;
27     PORTB |= PORTB & ~0x20;
28
29     // 512 @ 8
30     // set frequency to 3900hz
31     ICR1 = 512;
32
33     // fast pwm set on compare
34     TCCR1A = 0x02;
35     // prescaler set to 8
36     TCCR1B = 0x1a;
37 }
38
39 void PWM_dutyCycle(double percent)
40 {
41     OCR1A = percent * 512;
42 }
43
44 void PWM_enable(void)
45 {
46     TCCR1A |= 0x80;
47 }
48
49 void PWM_disable(void)
50 {
51     OCR1A = 0x00;
52
53     TCCR1A = (TCCR1A & ~0x80);
54 }
55
```

```
1  /*
2   * FileName: AnalogToDigital.h
3   * Version: 1
4   *
5   * Created: 10/19/2022 12:47 AM
6   * Author: Ethan Zeronik
7   *
8   * Operations: header for the adc submodule
9   */
10
11 #ifndef AnalogToDigital_h_INCLUDED
12 #define AnalogToDigital_h_INCLUDED
13
14 #if defined(__cplusplus)
15 extern "C" {
16 #endif
17
18 #include "stdio.h"
19
20 #define F_CPU 16000000UL
21
22 /* NOTE: Custom Types */
23 // TODO: None
24
25 /* NOTE: Function prototypes */
26 // init registers for adc
27 void ADC_init(void);
28 // returns the value of the given channel
29 double ADC_getTenBitValue(uint16_t channel);
30
31 #if defined(__cplusplus)
32 } /* extern "C" */
33 #endif
34
35 #endif // AnalogToDigital_h_INCLUDED
```

```
1  /*
2   * FileName: AnalogToDigital.c
3   * Version: 1
4   *
5   * Created: 10/19/2022 12:47 AM
6   * Author: Ethan Zeronik
7   *
8   * Operations: basic adc implementation
9   */
10
11 /* NOTE: Includes */
12 #include "AnalogToDigital.h"
13
14 #include <avr/io.h>
15
16 /* NOTE: Local declarations */
17 // TODO: None
18
19 /* NOTE: Global Variables */
20 // TODO: None
21
22 /* NOTE: Local function implementations */
23 void ADC_init(void)
24 {
25     // ten bit one way mode
26     ADCSRA = (1 << ADEN) | (1 << ADPS1) | (1 << ADPS0);
27
28     // 5v reference
29     ADMUX = (1 << REFS0);
30
31     ADCSRB = 0x00;
32 }
33
34 double ADC_getTenBitValue(uint16_t channel)
35 {
36     uint16_t result = 0;
37
38     // select the channel
39     ADMUX = (ADMUX & 0xe0) | channel;
40     ADCSRB = (ADCSRB & 0xf7) | (channel >> 2);
41
42     // start conversion
43     ADCSRA |= (1 << ADSC);
44
45     // wait for conversion
46     while((ADCSRA & (1 << ADSC)) == 1)
47     {
48         // do nothing
49     }
50
51     // save result
52     result = ADCL;
53     result = result | (ADCH << 8);
54
55     return result / 1024.0;
56 }
```