

```
1  /*
2   * FileName: main.c
3   * Version: 1
4   *
5   * Created: 10/26/2022 1:34:18 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_initInterrupt();
36
37      IO_init();
38
39      initDebug();
40
41      sei();
42
43      while(1)
44      {
45          if(DLY_getTick() > 500)
46          {
47              PORTB = ~PORTB & 0x80;
48
49              DLY_setTick(0);
50          }
51      }
52  }
53
54  /* NOTE: Function implementations */
55  void IO_init(void)
56  {
57      // set port B.7 as an output
```

```
58 |     DDRB = 0x80;  
59 |     PORTB = 0x00;  
60 | }  
61 |
```

```
1  /*
2   * FileName: main.c
3   * Version: 1
4   *
5   * Created: 10/26/2022 2:23:36 PM
6   * Author: Ethan Zeronik
7   *
8   * Operations: ADC using interrupts
9   *
10  * Hardware:
11  *   Atmega2560          micro controller
12  *   PORTA              LED bar
13  *   PORTF.0            Potentiometer in
14  */
15
16 /* NOTE: Includes */
17 #include <avr/io.h>
18
19 #include "AnalogToDigital.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     ADC_initInterrupt();
36
37     IO_init();
38
39     sei();
40
41     while(1)
42     {
43         // get the value and bit shift it right 2
44         PORTA = ADC_getTenBitValueInterrupt(0) >> 2;
45     }
46 }
47
48 /* NOTE: Function implementations */
49 void IO_init(void)
50 {
51     // set portA as an output
52     DDRA = 0xFF;
53     PORTA = 0x00;
54 }
55
```

```
1  /*
2  * FileName: main.c
3  * Version: 1
4  *
5  * Created: 10/26/2022 2:46:31 PM
6  * Author: Ethan Zeronik
7  *
8  * Operations: Encoder module
9  *
10 * Hardware:
11 *   Atmega2560          micro controller
12 *   PORTA               LED bar
13 *   PORTD1              Direction encoder pin
14 *   PORTD2              Encoder clock pin
15 */
16
17 /* NOTE: Includes */
18 #include <avr/io.h>
19
20 #include "Encoder.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     ENC_init();
37
38     IO_init();
39
40     sei();
41
42     while(1)
43     {
44         // get the flagged byte
45         uint8_t directionFlag = ENC_getDirection();
46
47         // if the flag has been set
48         if(directionFlag & 0x10)
49         {
50             // get the direction flag
51             if((directionFlag & 0x01))
52             {
53                 // move the bar left one
54                 PORTA = PORTA << 1 | 0x01;
55             }
56             else
57             {
```

```
58 |             // move the bar right one
59 |             PORTA = PORTA >> 1;
60 |         }
61 |     }
62 | }
63 |
64 |
65 | /* NOTE: Function implementations */
66 | void IO_init(void)
67 | {
68 |     // set portA as an output
69 |     DDRA = 0xFF;
70 |     PORTA = 0x00;
71 | }
```

```
1  /*
2  * FileName: main.c
3  * Version: 1
4  *
5  * Created: 10/28/2022 4:32:43 PM
6  * Author: Ethan Zeronik
7  *
8  * Operations: Mode enabled combination of previous parts
9  *
10 * Hardware:
11 *   Atmega2560          micro controller
12 *   PORTA               LED bar
13 *   PORTF.0            Potentiometer in
14 *   PORTD1             Direction encoder pin
15 *   PORTD2             Encoder clock pin
16 *   PORTD2             Mode pushbutton
17 */
18
19 /* NOTE: Includes */
20 #include <avr/io.h>
21
22 #include "Encoder.h"
23 #include "AnalogToDigital.h"
24 #include "Delay.h"
25
26 /* NOTE: Custom Macros */
27 // TODO: None
28
29 /* NOTE: Global Variables */
30 static uint8_t modeFlag = 0;
31
32 /* NOTE: Function prototypes */
33 // inits IO ports
34 void IO_init(void);
35
36 /* NOTE: Application implementation */
37 // the main loop of the function, provided to us
38 int main(void)
39 {
40     IO_init();
41
42     ENC_init();
43     ADC_initInterrupt();
44     DLY_initInterrupt();
45
46     sei();
47
48     while(1)
49     {
50         // the watchdog led
51         if(DLY_getTick() > 500)
52         {
53             PORTB = ~PORTB & 0x80;
54
55             if(modeFlag == 0)
56             {
57                 PORTA = ~PORTA & 0x01;
```

```
58         }
59
60         DLY_setTick(0);
61     }
62
63     // the mode switcher
64     if(modeFlag == 1)
65     {
66         // get the value and bit shift it right 2
67         PORTA = ADC_getTenBitValueInterrupt(0) >> 2;
68     }
69     else if(modeFlag == 2)
70     {
71         // get the flagged byte
72         uint8_t directionFlag = ENC_getDirection();
73
74         // if the flag has been set
75         if(directionFlag & 0x10)
76         {
77             // get the driection flag
78             if((directionFlag & 0x01))
79             {
80                 // move the bar left one
81                 PORTA = PORTA << 1 | 0x01;
82             }
83             else
84             {
85                 // move the bar right one
86                 PORTA = PORTA >> 1;
87             }
88         }
89     }
90 }
91
92
93 // the mode button interupt
94 ISR(INT3_vect)
95 {
96     if(modeFlag < 2)
97     {
98         modeFlag++;
99     }
100     else
101     {
102         modeFlag = 0;
103     }
104
105     PORTA = 0x00;
106 }
107
108 /* NOTE: Function implementations */
109 void IO_init(void)
110 {
111     // set port B.7 as an output
112     DDRB = 0x80;
113     PORTB = 0x00;
114 }
```

```
115 // set port A.7 as an output
116 DDRA = 0xFF;
117 PORTA = 0x00;
118
119 // set port D.3 as an input
120 DDRD = 0x00;
121 PORTD = 0x08;
122
123 // set interrupt 3 to on rising edge
124 EIMSK = (1 << INT3);
125 EICRA = (1 << ISC31);
126 }
127
```



```
1  /*
2   * FileName: Delay.c
3   * Version: 1
4   *
5   * Created: 10/26/2022 2:28 PM
6   * Author: Ethan Zeronik
7   *
8   * Operations: encoder definition
9   */
10
11 #ifndef Encoder_h_INCLUDED
12 #define Encoder_h_INCLUDED
13
14 #if defined(__cplusplus)
15 extern "C" {
16 #endif
17
18 #pragma message("WARNING: this module uses PORTD1:2 for interrupt")
19
20 #include <avr/interrupt.h>
21 #include <stdio.h>
22
23 /* NOTE: Custom Types */
24 // TODO: None
25
26 /* NOTE: Function prototypes */
27 // init for the encoder
28 void ENC_init(void);
29 // gets the raw value for the port
30 uint8_t ENC_getValue(void);
31 // returns the flagged direction
32 uint8_t ENC_getDirection(void);
33
34
35 #if defined(__cplusplus)
36 } /* extern "C" */
37 #endif
38
39 #endif // Encoder_h_INCLUDED
```

```
1  /*
2   * FileName: Delay.c
3   * Version: 1
4   *
5   * Created: 10/26/2022 2:28 PM
6   * Author: Ethan Zeronik
7   *
8   * Operations: encoder implementation
9   */
10
11 /* NOTE: Includes */
12 #include "Encoder.h"
13
14 #include <avr/io.h>
15
16 /* NOTE: Local declarations */
17 // TODO: None
18
19 /* NOTE: Global Variables */
20 static uint8_t value = 0;
21 // upper byte is flag lower is direction
22 static uint8_t direction = 0;
23
24 /* NOTE: Local function implementations */
25 void ENC_init(void)
26 {
27     // turn on the clock and direction inputs
28     DDRD |= 0x06;
29     PORTD |= 0x06;
30
31     // interrupt 2 to enabled falling edge
32     EIMSK |= (1 << INT2);
33     EICRA |= (1 << ISC21);
34 }
35
36 uint8_t ENC_getValue(void)
37 {
38     return value;
39 }
40
41 uint8_t ENC_getDirection(void)
42 {
43     // cache the flag state
44     uint8_t dir = direction;
45
46     // reset the flag in the global
47     direction = 0x00;
48
49     // return the cached state
50     return dir;
51 }
52
53 ISR(INT2_vect)
54 {
55     // if pin is high
56     if(PIND & 0x02)
57     {
```

```
58     // set the flag
59     direction = 0x11;
60
61     // increment if it won't overflow
62     if(value < 255)
63     {
64         value++;
65     }
66 }
67 else
68 {
69     // set the flag
70     direction = 0x10;
71
72     // decrement if it won't overflow
73     if(value > 0)
74     {
75         value--;
76     }
77 }
78 }
```

```
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 <avr/interrupt.h>
21 #include <stdio.h>
22
23 #define F_CPU 16000000UL
24
25 /* NOTE: Custom Types */
26 // TODO: None
27
28 /* NOTE: Function prototypes */
29 // init registers for delay
30 void DLY_init(void);
31 // init for the 1ms interrupt service
32 void DLY_initInterrupt(void);
33 // delay for an amount of ms
34 void DLY_ms(double ms);
35 // gets the ISR tick value
36 uint16_t DLY_getTick(void);
37 // sets the ISR tick value
38 void DLY_setTick(uint16_t t);
39
40 #if defined(__cplusplus)
41 } /* extern "C" */
42 #endif
43
44 #endif // Delay_h_INCLUDED
```

```
1  /*
2  * FileName: Delay.c
3  * Version: 1
4  *
5  * Created: 10/18/2022 7:26 PM
6  * Author: Ethan Zeronik
7  *
8  * Operations: create a custom delay function
9  */
10
11 /* NOTE: Includes */
12 #include "Delay.h"
13
14 #include <avr/io.h>
15
16 /* NOTE: Local declarations */
17 // TODO: None
18
19 /* NOTE: Global Variables */
20 static uint16_t tick = 0;
21
22 /* NOTE: Local function implementations */
23 void DLY_init(void)
24 {
25     // reset counter to 0
26     TCNT0 = 0;
27
28     // normal mode
29     TCCR0A = 0x00;
30     TCCR0B = 0x00;
31
32     OCR0A = 0;
33 }
34
35 void DLY_initInterrupt(void)
36 {
37     // interrupt mode
38     TCCR0A = 0x00;
39     TCCR0B = 0x04;
40     TCNT0 = 0;
41     OCR0A = 62;
42     TIMSK0 = (1 << OCIE0A);
43 }
44
45 void DLY_ms(double ms)
46 {
47     size_t time = ((ms / 1000.0) * F_CPU) / 1024;
48
49     if(ms <= 16)
50     {
51         OCR0A = time;
52
53         // prescalar of 1024
54         TCCR0B = 0x05;
55
56         while((TIFR0 & (1 << OCF0A)) == 0)
57         {
```

```
58         // do nothing
59     }
60
61     // stop the timer
62     TCCR0B = 0x00;
63     // clear the overflow flag
64     TIFR0 |= (1 << OCF0A);
65     TCNT0 = 0;
66     OCR0A = 0;
67 }
68 else
69 {
70     OCR0A = (((1 / 1000.0) * F_CPU) / 1024);
71
72     for(size_t i = 0; i < ms; i++)
73     {
74         // prescalar of 1024
75         TCCR0B = 0x05;
76
77         while((TIFR0 & (1 << OCF0A)) == 0)
78         {
79             // do nothing
80         }
81
82         // stop the timer
83         TCCR0B = 0x00;
84         // clear the overflow flag
85         TIFR0 |= (1 << OCF0A);
86         TCNT0 = 0;
87     }
88
89     OCR0A = 0;
90 }
91 }
92
93 uint16_t DLY_getTick(void)
94 {
95     return tick;
96 }
97
98 void DLY_setTick(uint16_t t){
99     tick = t;
100 }
101
102 ISR(TIMER0_COMPA_vect)
103 {
104     TCNT0 = 0;
105     tick++;
106 }
107
```

```
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 <avr/interrupt.h>
19 #include <stdio.h>
20
21 #define F_CPU 16000000UL
22
23 /* NOTE: Custom Types */
24 // TODO: None
25
26 /* NOTE: Function prototypes */
27 // init registers for adc
28 void ADC_init(void);
29 // init adc for interrupt mode
30 void ADC_initInterrupt(void);
31 // returns the value of the given channel
32 double ADC_getTenBitValue(uint16_t channel);
33 // gets the 10 bit value on channel o
34 uint16_t ADC_getTenBitValueInterrupt(uint16_t channel);
35
36 #if defined(__cplusplus)
37 } /* extern "C" */
38 #endif
39
40 #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 static uint16_t readInterrupt = 0;
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 void ADC_initInterrupt(void)
35 {
36     ADC_init();
37
38     ADCSRA |= (1 << ADIE);
39 }
40
41 double ADC_getTenBitValue(uint16_t channel)
42 {
43     uint16_t result = 0;
44
45     // select the channel
46     ADMUX = (ADMUX & 0xe0) | channel;
47     ADCSRB = (ADCSRB & 0xf7) | (channel >> 2);
48
49     // start conversion
50     ADCSRA |= (1 << ADSC);
51
52     // wait for conversion
53     while((ADCSRA & (1 << ADSC)) == 1)
54     {
55         // do nothing
56     }
57 }
```



```
58     // save result
59     result = ADCL;
60     result = result | (ADCH << 8);
61
62     return result / 1024.0;
63 }
64
65 uint16_t ADC_getTenBitValueInterrupt(uint16_t channel)
66 {
67     // select the channel
68     ADMUX = (ADMUX & 0xe0) | channel;
69     ADCSRB = (ADCSRB & 0xf7) | (channel >> 2);
70
71     // start conversion
72     ADCSRA |= (1 << ADSC);
73
74     return readInterrupt;
75 }
76
77 ISR(ADC_vect)
78 {
79     readInterrupt = ADCL;
80     readInterrupt = readInterrupt | (ADCH << 8);
81 }
```