Week 6 Practice

Attempt these exercises and then:

* Record a short video – show the system works on the BOARD (no need to show the screen). This is for me to see if your system works.
* Submit code and this video by Thursday of Week 7.

# Exercise 1

*This exercise is adopted from Exercise 2 of Lab 1*

*USE CTC with interrupts to generate* a square wave with the frequency of 0.125 Hz.

# Exercise 2

Write a program to use the Arduino Uno board and a button to control a built-in LED

* If the button is not pressed, the LED is blinking with the frequency of 1Hz
* If the button is pressed, the LED will blink with the frequency of 0.125 Hz

*For this exercise you are required to use ONLY ONE Timer Interrupt to generate the frequencies above.*

*Hints – You can use some variable (or flag) to check if the button is pressed and then change the TOP VALUE OF THE Timer* ***inside ISR***

# Exercise 3

*This exercise is adopted from Exercise 1 of Lab 1*

Write a program to use the Arduino Uno board and a button to control a single 7-segment LED. The control scheme is as follows:

* 1. When no buttons are pressed, the 7-segment LED shows number 5 (we refer to this as *Pattern 1*).
  2. If the button 1 is pressed, the 7-segment LED shows number 1 and the number flashes with the frequency of about 1Hz for 10 times (we refer to this as *Pattern 2*) then after that returns to *Pattern 1.*

Unlike in Lab 1 you are required to use:

* Button 1 is connected to External Interrupt INT0, and we will use External Interrupt to capture its change.
* You are required to use Timer INTERRUPT to generate the delay for 1Hz (0.5s ON and 0.5s OFF). You can use CTC mode if you wish.

*Hints – You can use some variable (or flag) to check if the button is pressed and then change the TOP VALUE OF THE Timer* ***inside ISR***

**EXERCISE 1**

#include <avr/io.h>

#include <avr/interrupt.h>

volatile int half\_second = 1;

volatile bool four\_elapsed = false;

int main(void)

{

DDRB |= (1 << 5); // Set LED as output

TCCR1B |= (1 << WGM12); // Turn on the CTC mode for Timer 1

TCCR1B |= (1 << CS12 ); // Set up Timer 1 with the prescaler of 256

OCR1A = 31249; // Set CTC compare value to 1Hz at 16 MHz AVR clock , with a prescaler of 256

TIMSK1 = 1<<OCIE1A; // Enable Output Compare A Match Interrupt

sei(); //Enable the Global Interrupt Bit

while (1)

{

if(four\_elapsed) //When 4 seconds

{

PORTB ^= (1 << 5); // Toggle the LED

four\_elapsed = false;

}

}

}

ISR ( TIMER1\_COMPA\_vect ) //0.5 seconds? how can we expand to 4 seconds?

{

if (half\_second == 8) // 8 x 0.5 = 4 second

{

four\_elapsed = true; // Indicate has 4 seconds already

half\_second =1;

}

else

{

half\_second++;

}

}

**Exercise 2**

#include <avr/io.h>

#include <avr/interrupt.h>

volatile int half\_second = 1;

volatile bool four\_elapsed = false;

volatile bool halfsec\_elapsed = false;

int main(void)

{

DDRB |= (1 << 5); // Set LED as output

DDRD &= ~(1<<2);

TCCR1B |= (1 << WGM12); // Turn on the CTC mode for Timer 1

TCCR1B |= (1 << CS12 ); // Set up Timer 1 with the prescaler of 256

OCR1A = 31249; // Set CTC compare value to 1Hz at 16 MHz AVR clock , with a prescaler of 256

TIMSK1 = 1<<OCIE1A; // Enable Output Compare A Match Interrupt

sei(); //Enable the Global Interrupt Bit

while (1)

{

if (!(PIND & (1<<PIND2))) //if button is pressed

{

if(four\_elapsed) //When 4 seconds

{

PORTB ^= (1 << 5); // Toggle the LED

four\_elapsed = false;

}

}

else //Button is not pressed

{

if (halfsec\_elapsed)

{

PORTB ^= (1<<5);

halfsec\_elapsed = false;

}

}

}

}

ISR ( TIMER1\_COMPA\_vect ) //0.5 seconds? how can we expand to 4 seconds?

{

halfsec\_elapsed = true; //0.5 seconds has been elapsed.

if (half\_second == 8) // 8 x 0.5 = 4 second

{

four\_elapsed = true; // Indicate has 4 seconds already

half\_second =1;

}

else

{

half\_second++;

}

}

**Exercise 3**

#include <avr/io.h>

#include <avr/interrupt.h>

volatile bool halfsec\_elapsed = false;

volatile bool button\_pressed = false;

int main(void)

{

DDRB = 0xFF;

DDRC |= (1<<0);

DDRD &= ~(1<<2); // INT0 external interrupt

//Display number 5

PORTC |= (1<<0);

PORTB = B00101101;

//enable interrupts

EIMSK |= (1<<INT0); //enable PORTD2 as external interrupt

EICRA |= (1<<ISC01); // Detect on falling edge

EIFR |= (1<<INTF0); // Clear flag

TCCR1B |= (1 << WGM12); // Turn on the CTC mode for Timer 1

TCCR1B |= (1 << CS12 ); // Set up Timer 1 with the prescaler of 256

OCR1A = 31249; // Set CTC compare value to 1Hz at 16 MHz AVR clock , with a prescaler of 256

sei(); //Enable the Global Interrupt Bit

int count = 0; //initialize couting value

while (1){

if (button\_pressed) { // button\_pressed 1 pressed

PORTB = 0x00;

PORTC &= ~(1<<0);

while (count < 20) { // pattern 2 flashes as 1Hz frequency 10 times

if (halfsec\_elapsed)

{

if (PORTB == B00000110)

{

PORTB = 0x00;

PORTC &= ~(1<<0);

}

else

{

PORTB = B00000110;

PORTC |= ~(1<<0);

}

count++;

halfsec\_elapsed = false;

}

}

count = 0; // reset count value

// change to default pattern - Number 5

PORTC |= (1<<0);

PORTB = B00101101;

button\_pressed = false;

}

}

}

ISR (INT0\_vect)

{

button\_pressed = true;

TCNT1 = 0;

TIMSK1 = 1<<OCIE1A; // Enable Output Compare A Match Interrupt

}

ISR ( TIMER1\_COMPA\_vect )

{

halfsec\_elapsed = true;

}