CS 5331: Special Problems in CS: Cyber-Physical Systems Spring 2023

Assignment 8: Interrupt and Timer

Amir Faiyaz

R11772642

a.

Objective:
To write an Arduino Sketch code using bitmath, interrupt, and timer to generate a waveform output with a specific frequency and control it with push switches. The code should use the lowest possible power-consuming sleep mode.

Procedure:
1. Set up the hardware with a LED connected to Pin 6 and Push Switches connected to Pin 7 and Pin 8.
2. Write preliminary code to ensure the hardware setup of the push switches, where the LED blinks on/off when the push switch is pressed. Disable code for timer.
3. Use PCINT0 for PortB and PCINT2 for PortD and set the PCINT interrupt triggers on both edges of switch press.
4. Use a flag variable to check if the edge is even number or not, then toggle the LED.
5. Use Timer1 in OVF mode and set the maximum/minimum limit based on the corresponding count1 values.
6. For the timer 1 to be active, clkIO must be kept enabled. Test the functionality of the code without using the sleep mode first, then implement the sleep mode.
7. Use the push switches to control the frequency of the output waveform. Push Switch 1 will reduce the frequency of the output waveform by half until it reaches 0.5 Hz (lowest frequency) as soon as the switch is pressed. Push Switch 2 will increase the frequency of the output waveform by twice until it reaches 8 Hz (highest frequency) as soon as the switch is pressed.

Key results:
The final code will generate a waveform output with a specific frequency controlled by push switches. The LED connected to Pin 6 will blink at the rate of the frequency of the waveform generated by the code. Push Switch 1 will reduce the frequency of the output waveform by half, and Push Switch 2 will increase the frequency of the output waveform by twice. The code will use the lowest possible power-consuming sleep mode.

b.

```
#include <avr/interrupt.h>
#include <avr/sleep.h>


#define LED_PIN 6
#define SWITCH1_PIN 7
#define SWITCH2_PIN 8
```

```
volatile uint8_t frequency = 2; // Starting frequency at 2 Hz

// Timer1 ISR to generate waveform output
ISR(TIMER1_COMPA_vect) {
  static uint8_t counter = 0;
  if (++counter >= (F_CPU / 2 / frequency / 256)) {
    PORTD ^= (1 << LED_PIN);
    counter = 0;
  }
}

// PCIE0 ISR for switch 1 to reduce frequency by half
ISR(PCINT0_vect) {
  if (bit_is_clear(PINB, SWITCH1_PIN)) {
    if (frequency > 0.5) frequency /= 2;
  }
}

// PCIE2 ISR for switch 2 to increase frequency by twice
ISR(PCINT2_vect) {
  if (bit_is_clear(PIND, SWITCH2_PIN)) {
    if (frequency < 8) frequency *= 2;
  }
}

void setup() {
  // Set LED pin as output
  DDRD |= (1 << LED_PIN);

  // Enable PCIE0 and PCIE2 interrupts for switches 1 and 2
  PCICR |= (1 << PCIE0) | (1 << PCIE2);

  // Enable interrupt on switch pins
  PCMSK0 |= (1 << SWITCH1_PIN);
  PCMSK2 |= (1 << SWITCH2_PIN);

  // Setup Timer1 for CTC mode with prescaler 256
  TCCR1B |= (1 << WGM12) | (1 << CS12);
  OCR1A = F_CPU / 2 / 256 / frequency - 1; // Set compare match value for desired
frequency
  TIMSK1 |= (1 << OCIE1A); // Enable timer compare match interrupt

  // Enable global interrupts
  sei();
```

```
  // Set sleep mode to power-save mode
  set_sleep_mode(SLEEP_MODE_PWR_SAVE);
}

void loop() {
  sleep_mode(); // Enter sleep mode to save power
}
```

c.

```
etch_apr9a.ino
 1  #include <avr/interrupt.h>
 2  #include <avr/sleep.h>
 3
 4
 5  #define LED_PIN 6
 6  #define SWITCH1_PIN 7
 7  #define SWITCH2_PIN 8
 8
 9  volatile uint8_t frequency = 2; // Starting frequency at 2 Hz
10
11  // Timer1 ISR to generate waveform output
12  ISR(TIMER1_COMPA_vect) {
13    static uint8_t counter = 0;
14    if (++counter >= (F_CPU / 2 / frequency / 256)) {
15      PORTD ^= (1 << LED_PIN);
16      counter = 0;
17    }
18  }
19
20  // PCIE0 ISR for switch 1 to reduce frequency by half
21  ISR(PCINT0_vect) {
22    if (bit_is_clear(PINB, SWITCH1_PIN)) {
utput
Sketch uses 1210 bytes (3%) of program storage space. Maximum is 32256 bytes.
Global variables use 12 bytes (0%) of dynamic memory, leaving 2036 bytes for local variables. Maximum is 2048 bytes.
```

d.