**CS 5331-003: Special Problems in Computer Science: Embedded Systems**

**Instructor: Dr. Morshed, Associate Professor**

**Assignment-8: Interrupt and Timer**

**Anvesh Muppeda**

**R11840667**

[**amuppeda@ttu.edu**](mailto:amuppeda@ttu.edu)

**Submission Date: 04/12/2024**

A black and white logo

Description automatically generated

**Task:**

- Arduino Pin 8 is connected to Push Switch 1. Connect a 1 kΩ resistance to the same pin to GND (0 V) of the Arduino. The other side pin of that push switch will connect to 5 V of the Arduino.

- Arduino Pin 7 is connected to Push Switch 2. Connect a 1 kΩ resistance to the same pin to GND (0 V) of the Arduino. The other side pin of that push switch will connect to 5 V of the Arduino.

- Arduino Pin 6 is connected to the LED. Use 220 Ω or 330 Ω for the resistance in series with the LED. The other side of this resistor is connected to GND (0 V) of the Arduino. 1

For the above hardware setup, write an Arduino Sketch code using bitmath, interrupt, and timer to generate a waveform output with following specifications:

1) The LED connected on Pin 6 will blink at a rate of frequency (either at 0.5, 1, 2, 4, or 8 Hz) of the waveform generated by the code (using Timer 1 ISR).

2) The Push Switch 1 (using PCIE0 ISR) 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.

3) The Push Switch 2 (using PCIE2 ISR) will increase the frequency of the output waveform by twice until it reaches 8 Hz (highest frequency) as soon as the switch is pressed.

4) The microcontroller must use the lowest possible power consuming sleep mode.

1. **Briefly summarize the work including your objective, procedure, and key results.**

**Objective:**

The aim was to develop an Arduino sketch capable of generating a waveform output through Timer1 ISR, synchronizing LED blinking with the waveform's frequency. Furthermore, the sketch required integrating features for two push switches: one to halve the frequency and the other to double it. Additionally, the microcontroller was to be utilized to achieve the lowest feasible power consumption during idle periods.

**Procedure:**

The initialization phase establishes pin assignments, constant values, and global variables. It sets up pin configurations, activates switch interrupts, and initializes Timer1 for generating the waveform. During the main loop, the microcontroller transitions into low-power sleep mode. Interrupt Service Routines (ISRs) manage switch inputs and Timer1 overflow to control LED toggling and frequency adjustments. Functions for frequency modification ensure updates to Timer1 align with desired changes. Ultimately, the microcontroller transitions back to low-power sleep mode, optimizing energy usage.

**Key Results:**

The sketch generates a waveform output with a blinking LED synchronized with Timer1 ISR. Users can adjust the LED's blinking frequency using two push switches, one decreasing it by half and the other increasing it by double. The microcontroller efficiently utilizes low-power sleep mode during idle periods, conserving energy when not actively processing inputs or generating the waveform. The code handles switch inputs and adjusts the LED's blinking frequency seamlessly, ensuring a smooth user experience without noticeable delays or glitches.

1. **Provide your code for this assignment with sufficient comments to ensure the code is understandable.**

**Code:**

#include <avr/sleep.h>

// Defining pin numbers

// Pin connected to the LED

#define LED\_PIN 6

// Pin connected to switch 1

#define SWITCH\_1\_PIN 8

// Pin connected to switch 2

#define SWITCH\_2\_PIN 7

// Defining frequency limits

// Low limit of frequency (Hz)

#define LOW\_FREQUENCY 0.5

// High limit of frequency (Hz)

#define HIGH\_FREQUENCY 8.0

// Defining timer constants

const unsigned long TIMER1\_COUNT1 = 15625; // Corresponds to 1Hz frequency

// Declaration of Global variables

volatile float frequency = 1.0; // Initial frequency

// Flag indicating if switch 1 is pressed

volatile bool g\_Switch1Pressed = false;

// Flag indicating if switch 2 is pressed

volatile bool g\_Switch2Pressed = false;

// Function Setup

void setup() {

pinMode(LED\_PIN, OUTPUT);

pinMode(SWITCH\_1\_PIN, INPUT\_PULLUP);

pinMode(SWITCH\_2\_PIN, INPUT\_PULLUP);

// Enable pin change interrupts

// Selecting the port B and port D

PCICR |= B00000101;

// Enabling the PCINT0 (PortB-D8)

PCMSK0 |= B00000001;

// Enabling the PCINT23 (PortD-D7)

PCMSK2 |= B10000000;

// Initialize Timer1

TCCR1A = 0;

TCCR1B = 0;

// Initializing the timer count to achieve 1Hz frequency

TCNT1 = 0xFFFF - TIMER1\_COUNT1;

// Starting the Timer1 with no prescaler

TCCR1B |= (1 << CS10);

// Enable Timer1 overflow interrupt

TIMSK1 |= (1 << TOIE1);

// Enable interrupts Globally

sei();

}

// Loop function

void loop() {

// Enter sleep mode

set\_sleep\_mode(SLEEP\_MODE\_IDLE);

sleep\_enable();

sleep\_mode();

}

// Pin Change Interrupt for SWITCH\_1\_PIN

ISR(PCINT0\_vect)

{

if (digitalRead(SWITCH\_1\_PIN) == LOW)

{

// Set flag indicating switch 1 is pressed

g\_Switch1Pressed = true;

}

}

// Pin Change Interrupt for SWITCH\_2\_PIN

ISR(PCINT2\_vect)

{

if (digitalRead(SWITCH\_2\_PIN) == LOW)

{

// Set flag indicating switch 2 is pressed

g\_Switch2Pressed = true;

}

}

// Timer1 overflow interrupt

ISR(TIMER1\_OVF\_vect)

{

// Increase the elapsed time since the last toggle

static unsigned long s\_previousToggleMillis = 0;

// Current time in milliseconds

unsigned long currentMillis = millis();

// Time elapsed since last toggle

unsigned long elapsedTime = currentMillis - s\_previousToggleMillis;

// Check if it's time to toggle the LED based on frequency

if (elapsedTime >= (1000 / frequency))

{

// Toggle the LED Func

toggleLED();

// Update last toggle time

s\_previousToggleMillis = currentMillis;

}

// Adjust frequency if switches are pressed

if (g\_Switch1Pressed)

{

// Halve the frequency

frequency /= 2;

if (frequency < LOW\_FREQUENCY)

{

// Ensure frequency doesn't go below the lower limit

frequency = LOW\_FREQUENCY;

}

// Update Timer1 with the new frequency

updateTimer();

// Reset switch 1 flag

g\_Switch1Pressed = false;

}

if (g\_Switch2Pressed)

{

// Double the frequency

frequency \*= 2;

if (frequency > HIGH\_FREQUENCY)

{

// Ensure frequency doesn't exceed the upper limit

frequency = HIGH\_FREQUENCY;

}

// Update Timer1 with the new frequency

updateTimer();

// Reset switch 2 flag

g\_Switch2Pressed = false;

}

}

// Function to toggle LED

void toggleLED()

{

// Variable to store the current state of LED

static bool s\_ledState = false;

// Toggle the LED state

digitalWrite(LED\_PIN, s\_ledState);

// Update LED state for next toggle

s\_ledState = !s\_ledState;

}

// Function to update Timer1 based on the new frequency

void updateTimer()

{

// Calculate new timer count value

unsigned long updatedCountOne = TIMER1\_COUNT1 \* (1 / frequency);

// Disable the interrupts

cli();

// Updating the Timer1 count

TCNT1 = 0xFFFF - updatedCountOne;

// Enable the interrupts

sei();

}

1. **Provide snapshots confirming successful download of the code from computer to Arduino board.**

**Successful Compiling Screenshot:**

**A screenshot of a computer

Description automatically generated**

**Successful Uploading Screenshot:**

**A screenshot of a computer

Description automatically generated**

1. **Provide a few pictures as evidence of successful prototyping of the hardware and operation of the code.**

**Setup Screenshots:**

A circuit board with wires

Description automatically generated

A circuit board with wires and a cable

Description automatically generated

A circuit board with wires connected to it

Description automatically generated

**Output Screenshots:**

A person's hands connecting wires to a circuit board

Description automatically generated

A person's hands connecting wires to a circuit board

Description automatically generated

A person holding a circuit board with wires

Description automatically generated

**Video:**

