0.2 Polling Function

/\*

 \* LED Blink1.c

 \*

 \* Created: 12-09-2018 02:05:19 PM

 \* Author : Tushar & Sourajit

 \*/

#define F\_CPU 16000000L

#include <avr/io.h>

/\* Replace with your library code \*/

int main(void)

{

DDRB = 0x20; //0x20; 0b00100000 PB5 is set at op

PORTB = 0x01;   // Pin 1 is set High

while(1)

{

if(!(PINB & 0x01))

{

PORTB |= 0x20; // Switch is pressed and LED turns ON

}

else

{

PORTB &= 0xDF; // Switch is not pressed and LED is OFF

}

}

//return 0;

}

0.3 Buzzer Code

/\*

\* BuzzerTimer0 0.3 ESE519 SD TGH.c

\*

\* Created: 9/13/2018 2:32:31 AM

\* Author : tghed

\*/

#define *F\_CPU* 16000000L

#include <avr/io.h>

int main(void)

{

int a;

DDRD = (1 << PD6); //PD6 is the output pin

PORTD = 0; //Port D initialized

TCCR0B |= (1 << CS02)|(1 << CS00); //(Timer setup with prescaler of 1024)

TCNT0 = 0; // counter initialized

a = 0;

/\* Replace with your application code \*/

while (1)

{

if(TCNT0 >= 124)

{

a = 1-a; //State toggling

PORTD = (a << PD6); //Toggling PD6

TCNT0 = 0; //Reset counter

}

}

}

* 1. Interrupt Code

\* 1.1 Interrupt LED swicth.c

\*

\* Created: 9/19/2018 8:47:16 PM

\* Author : tghed

\*/

#define *F\_CPU* 16000000L

#include <avr/io.h>

#include <avr/interrupt.h>

#include <stdio.h>

void timer1\_init()

{

TCCR1B |= (1 << CS00); //starting the timer1

TCCR1B |= (0 << 6); // set to capture falling edge

TIFR1 |= (1 << 5); // Input capture flag reset

TIMSK1 |= (1 << 5); // Enabled interrupt for input capture

sei(); // global interrupt enabled

}

int main(void)

{

DDRB = (1 << 5); //PB5 is our LED which is set to give output

PORTB = (1 << 0); //PB0 is pulled high (here input capture is to occur)

timer1\_init();

while (1)

{

}

}

ISR(TIMER1\_CAPT\_vect)

{

PORTB ^= (1 << 5); //First as it is set to capture falling edge, the LED is set to glow, the LED output is toggled upon further interrupts

TIFR1 |= (1 << 5); // Input capture flag reset

TCCR1B ^= (1 << 6); // Toggling between raising and falling edge function

}

* 1. Interrupt 440 Hz waveform

/\*

\* 1.2 Interrupt 440Hz Buzzer.c

\*

\* Created: 9/19/2018 9:48:44 PM

\* Author : tghed

\*/

#define *F\_CPU* 16000000L

#include <avr/io.h>

#include <stdio.h>

#include <avr/interrupt.h>

void timer0\_init()

{

TCCR0B |= (1 << CS02); // Pre scaler setting of 256

TCCR0A = (1 << 6); //Output Compare setting

TIMSK0 = (1 << 1);//Enable output compare A interrupt

TCNT0 = 0; //counter initialized

sei();

}

int main(void)

{

DDRD = (1 << 6); // Setting PD6 to output

timer0\_init();

OCR0A = 70; //70 Counts after which interrupt to be enabled (approx. half cycle)

while (1)

{

}

}

ISR(TIMER0\_COMPA\_vect)

{

PORTD ^= (1 << 6);

TCNT0 = 0;

OCR0A = 70;

TIFR0 = (1 << 2);

}

2.1 CTC mode waveform

/\*

\* 2.1 CTC mode waveform.c

\*

\* Created: 9/20/2018 12:49:13 AM

\* Author : tghed

\*/

#define *F\_CPU* 16000000L

#include <avr/io.h>

#include <stdio.h>

#include <avr/interrupt.h>

void timer0\_init()

{

TCCR0B |= (1 << CS02); // Pre scaler setting of 256

TCCR0A |= (1 << WGM01); // CTC mode on

TIMSK0 = (1 << 1);//Enable output compare A interrupt

sei();//global interrupts

}

int main(void)

{

DDRD |= (1 << 6); // Setting PD6 to output

timer0\_init();

OCR0A = 70; //70 Counts after which interrupt to be enabled (approx. half cycle)

while (1)

{

}

}

ISR(TIMER0\_COMPA\_vect)

{

PORTD ^= (1 << 6); //toggle waveform

OCR0A = 70;

}

2.2 Pulse width rangefinder

/\*

\* 2.2 Ultrasonic range finder.c

\*

\* Created: 9/20/2018 3:24:26 PM

\* Author : tghed

\*/

#include <avr/io.h>

#include <stdio.h>

#include <avr/interrupt.h>

#include "uart.h"

int e1, e2, d;

void timer1\_init()

{

TCCR1A |= (1 << 6)| (1 << 4); //output compare set A and B

TCCR1B |= (1 << 0); // Pre scaler 1 timer starter

TIMSK1 |= (1 << 1) | (1 << 2); // output compare interrupt enabled

TCNT1 = 0; // timer count initialization

sei(); //global interrupt enabled

}

int main(void)

{

uart\_init();

DDRB |= (1 << 1) | (0 << 0); // PB1 output, PB0 input

PORTB |= (1 << 1); // PB1 set high

OCR1A = 79; //Amounting to 5us

//OCR1B = 12708; //Amounting to 750us + 5us(initial)

timer1\_init();

while (1)

{*printf*("The difference between e1 %u and e2 %u \n", e1, e2);

*printf*("is %u \n", d);

}

}

ISR(TIMER1\_COMPA\_vect)

{

PORTB &= ~(1 << 1); // set output low

TCCR1B |= (1 << 6); // input capture set to high

TIMSK1 |= (1 << 5); // enable input capture interrupt

}

ISR(TIMER1\_COMPB\_vect)

{

if (!(PINB0 & 0)) //if pulse not being read

TCNT1 = 0; // counter reset for new sequence

}

ISR(TIMER1\_CAPT\_vect)

{

if (TCCR1B & (1 << 6)) // check if input capture high

{

e1 = ICR1;

TCCR1B &= ~(1 << 6); // input capture set to low

}

else

{

e2 = ICR1;

TCNT1 = 0; // counter reset for new sequence

}

d = e2 - e1;

}

2.3 Continuous Frequency generation using ping sensor

/\*

\* 2.3 Theremin Continuous.c

\*

\* Created: 9/23/2018 2:09:05 PM

\* Author : tghed

\*/

#include <avr/io.h>

#include <stdio.h>

#include <avr/interrupt.h>

#include "uart.h"

int e1, e2, d;

void timer0\_init()

{

TCCR0A |= (1 << WGM01); //CTC mode

TCCR0B |= (1 << CS01) | (1 << CS00);// Prescaler 64

TIMSK0 |= (1 << 1); //Output Compare A interrupt enable

TCNT0 = 0; //timer initialized

}

void timer1\_init()

{

TCCR1A |= (1 << 6)| (1 << 4); //output compare set A and B

TCCR1B |= (1 << 0); // Pre scaler 1 timer starter

TIMSK1 |= (1 << 1) | (1 << 2); // output compare interrupt enabled

TCNT1 = 0; // timer count initialization

sei(); //global interrupt enabled

}

int main(void)

{

uart\_init();

DDRB |= (1 << 1); // PB1 output, PB0 input

DDRD |= (1 << 6); //PD6 output, PD2 input

PORTB |= (1 << 1); // PB1 set high

//PORTD |= (1 << 2); //PD2 set high

OCR1A = 79; //Amounting to 5us

//OCR1B = 12708; //Amounting to 750us + 5us(initial)

timer0\_init();

timer1\_init();

while (1)

{

OCR0A = d\*0.01;

*printf*("The difference between e1 %u and e2 %u \n", e1, e2);

*printf*("is %u \n", d);

}

}

ISR(TIMER0\_COMPA\_vect)

{

PORTD ^= (1 << 6);

}

ISR(TIMER1\_COMPA\_vect)

{

PORTB &= ~(1 << 1); // set output low

TCCR1B |= (1 << 6); // input capture set to high

TIMSK1 |= (1 << 5); // enable input capture interrupt

}

ISR(TIMER1\_COMPB\_vect)

{

if (!(PINB0 & 0)) //if pulse not being read

TCNT1 = 0; // counter reset for new sequence

}

ISR(TIMER1\_CAPT\_vect)

{

if (TCCR1B & (1 << 6)) // check if input capture high

{

e1 = ICR1;

TCCR1B &= ~(1 << 6); // input capture set to low

}

else

{

e2 = ICR1;

TCNT1 = 0; // counter reset for new sequence

}

d = e2 - e1;

}

2.4 Discrete Frequency generation as per the given theremin notes

/\*

\* 2.4 Theremin Discrete output.c

\*

\* Created: 9/23/2018 2:07:02 PM

\* Author : tghed

\*/

#include <avr/io.h>

#include <stdio.h>

#include <avr/interrupt.h>

#include "uart.h"

int e1, e2, d;

void timer0\_init()

{

TCCR0A |= (1 << WGM01); //CTC mode

TCCR0B |= (1 << CS01) | (1 << CS00);// Prescaler 64

TIMSK0 |= (1 << 1); //Output Compare A interrupt enable

TCNT0 = 0; //timer initialized

}

void timer1\_init()

{

TCCR1A |= (1 << 6)| (1 << 4); //output compare set A and B

TCCR1B |= (1 << 0); // Pre scaler 1 timer starter

TIMSK1 |= (1 << 1) | (1 << 2); // output compare interrupt enabled

TCNT1 = 0; // timer count initialization

sei(); //global interrupt enabled

}

int main(void)

{

uart\_init();

DDRB |= (1 << 1); // PB1 output, PB0 input

DDRD |= (1 << 6); //PD6 output, PD2 input

PORTB |= (1 << 1); // PB1 set high

//PORTD |= (1 << 2); //PD2 set high

OCR1A = 79; //Amounting to 5us

//OCR1B = 12708; //Amounting to 750us + 5us(initial)

timer0\_init();

timer1\_init();

while (1)

{

if (d < 10000)

OCR0A = 119;

else if (d > 10000 && d < 13000)

OCR0A = 106;

else if (d > 13000 && d < 16000)

OCR0A = 94;

else if (d > 16000 && d < 19000)

OCR0A = 89;

else if (d > 19000 && d < 22000)

OCR0A = 79;

else if (d > 22000 && d < 25000)

OCR0A = 71;

else if (d > 25000 && d < 28000)

OCR0A = 63;

else

OCR0A = 59;

*printf*("The difference between e1 %u and e2 %u \n", e1, e2);

*printf*("is %u \n", d);

}

}

ISR(TIMER0\_COMPA\_vect)

{

PORTD ^= (1 << 6);

}

ISR(TIMER1\_COMPA\_vect)

{

PORTB &= ~(1 << 1); // set output low

TCCR1B |= (1 << 6); // input capture set to high

TIMSK1 |= (1 << 5); // enable input capture interrupt

}

ISR(TIMER1\_COMPB\_vect)

{

if (!(PINB0 & 0)) //if pulse not being read

TCNT1 = 0; // counter reset for new sequence

}

ISR(TIMER1\_CAPT\_vect)

{

if (TCCR1B & (1 << 6)) // check if input capture high

{

e1 = ICR1;

TCCR1B &= ~(1 << 6); // input capture set to low

}

else

{

e2 = ICR1;

TCNT1 = 0; // counter reset for new sequence

}

d = e2 - e1;

}