Ian Sbar

4-1-2024

Professor Muhn

RBT211

LED visualizer

## Code

/\*

\* Assignment 9.1- bit visualizer.cpp

\*

\* Created: 3/28/2024 5:30:39 PM

\* Author : iansb

\*/

#include <avr/io.h>

#ifndef F\_CPU

#define F\_CPU 1000000UL

#endif

#include <avr/io.h>

int main(void)

{

ADMUX|= (0 << REFS2) | (0 << REFS1 ) | (0 << REFS0 ) | (1 << ADLAR) | (0 << MUX3 ) | (0 << MUX2 ) | (1 << MUX1 ) | (0 << MUX0 ); //REFS pins are voltage reference selection bits. seting (REFS1, REFS0) as (0,0)

// ADLAR- left adjust bit

// MUX pins select which ADC pin to use. (0010) means we are suing ADC2, which is PB4 (pin 3). this pin is connected to the potentiometer

ADCSRA |= (1 << ADEN ) | (0 << ADSC ) |(0 << ADATE) |(0 << ADIF ) |(0 << ADIE ) |(1 << ADPS2 ) |(1 << ADPS1 ) |(0 << ADPS0 ); // ADEN: ADC enable

// ADSC: starts conversions in setup code if enabled. look at page 136 on datasheet for more info

// ADATE: Auto trigger enable. trigger source selected in ADCSRB

// ADIF: intterupt flag. makes it run in an interupt block

// ADPS bits: Prescaler select bits. determines the prescaler division factor between system clock frequecy and input. minimum divion is 2 (0,0,0)

ADCSRB |= (0 << ADTS2 ) | ( 0 << ADTS1) | (0 << ADTS0 ); //ADTS = auto triger source: (0,0,0) puts it in free running mode

//START ADC CONVERSION (Free runnign)

// because the ATtiny is bineg used, PWM is needed

TCCR0A |= (1 << COM0B1) | (0 << COM0B0) | (1 << WGM01) | (1 << WGM00); //WGM02, WGM01, WGM00 are set to (0,1,1) to enable fast PWM mode

// The COM ports (1,0), represent OC0A/OC0B depending if its C0M0Bx oe COM0Ax. (1,0) puts the timer in compare match mode

TCCR0B |=(0 << CS02) | (0 << CS01) | (1 << CS00) | (0 << WGM02);

// CS02, CS01, CS00vare set to (0,0,1), which sets no prescaler.

DDRB |= (1 << PB1); // sets PB1 (pin 6) as output

while (1)

{

int Analog\_value\_H = ADCH; // reads ADC the 8-bit high (most sig bits) value into this integer. if ADLAR = 0, there is only two that can be read (0b000000xx)

int Analog\_value\_L = ADCL; // reads ADC low value (least significant bits) into this integer (0bxxxxxxxx)

int Analog\_value = ADC; // 10 bit number 0bxxxxxxxxxx

//Analog\_value

// for 10 bits my max value for is 2^10 = 1024 -1 because start 0 to 102

// ADCH

// be the two highest bits = 2^2

//ADCL

// 8 least significant bits 2^8. between 0-255

//0 will be 0 V read

// MAX value (1023) will be 5 volts

ADCSRA |= (1 << ADSC); // start ADC measurement

while (ADCSRA & (1 << ADSC) ); // wait till conversion complete

OCR0B = ADCH;

}

}

## Video

Video will be sent separately