CPE301 – FALL 2019

Design Assignment 6

Student Name: Cameron Kirk

Student #: 2000660135

Student Email: kirkc1@unlv.nevada.edu

Primary Github address: <https://github.com/kirkster96/submission_da>

Directory: <https://github.com/kirkster96/submission_da/tree/master/DesignAssignment/DA6>

1. **COMPONENTS LIST AND CONNECTION BLOCK DIAGRAM w/ PINS**

Atmega328PB

Mutli-function shield

LM35

1. **AVR C DEVELOPED CODE OF TASK 1**

/\*\*

\* \DA3\_a Cameron Kirk

\*

\* \This is a AVR C program that will use the LM35 to print temperatur to the terminal

\*

\*/

#define *F\_CPU* 16000000UL

#define I2C\_SCL\_CLOCK 100000L

#define BAUD\_RATE 9600

#define BAUD\_PRESCALE (((*F\_CPU* / (BAUD\_RATE \* 16UL))) - 1)

#define TRUE 1

#define FALSE 0

#define CHAR\_NEWLINE '\n'

#define CHAR\_RETURN '\r'

#define RETURN\_NEWLINE "\r\n"

#include <avr/io.h>

#include <avr/interrupt.h>

#include <util/delay.h>

#include <stdlib.h>

#include <string.h>

#include <stdio.h>

void usart\_init ();

void usart\_send (char ch);

void UART\_sendString(const char \*str);

void check\_OK();

volatile unsigned char data\_in[8];

volatile unsigned char command\_in[8];

volatile unsigned char data\_count;

volatile unsigned char command\_ready;

int main (void)

{

//initialize the usart

usart\_init ();

//initialize the TWI for I2C

TWBR0 = ((*F\_CPU*/I2C\_SCL\_CLOCK)-16)/2; //set module clock freq.

TWCR0 |= (1<<TWEN); //enable the TWI module

//Set port

volatile unsigned char TEMP [4];

while (1){

UART\_sendString(TEMP);

*\_delay\_ms*(2000);

UART\_sendString("\r\n\r\n");

}

return 0;

}

void I2C\_Start\_Wait(int value)

{

// send the START condition

TWCR0 |= (1<<TWINT)|(1<<TWEN)|(1<<TWSTA);

while((TWCR0 & (1 << TWINT)) == 0);

}

void I2C\_WAIT\_CLEAR(int a, int b)

{

while((TWCR0 & (1 << TWINT)) == 0);

}

void I2C\_Write(int value)

{

//Send device address

TWDR0 |= value;

TWCR0 |= (1<<TWINT)|(1<<TWEN);

//wait until transmission completed and ACK/NACK has been recieved

while ((TWCR0 & (1<<TWINT)) == 0);

}

void I2C\_Stop()

{

TWCR0 |= (1<<TWINT)|(1<<TWEN)|(1<<TWSTO);

}

unsigned char I2C\_Read(unsigned char isLast){

if (isLast == 0)

TWCR0 |= (1<<TWINT)|(1<<TWEN)|(1<<TWEA);

else

TWCR0 |= (1<<TWINT)|(1<<TWEN);

while ((TWCR0 & (1<<TWINT)) == 0);

return TWDR0;

}

void MPU6050\_Init ()/\* Gyro initialization function\*/

{

int INT\_ENABLE;

int GYRO\_CONFIG;

int CONFIG;

int PWR\_MGMT\_1;

int SMPLRT\_DIV;

*\_delay\_ms* ( 150);/\* Power up time >100ms\*/

I2C\_Start\_Wait(0xD0);/\* Start with device write address\*/

I2C\_Write(SMPLRT\_DIV);/\* Write to sample rate register\*/

I2C\_Write(0x07);/\* 1KHz sample rate\*/

I2C\_Stop();

I2C\_Start\_Wait(0xD0);

I2C\_Write(PWR\_MGMT\_1);/\* Write to power management register\*/

I2C\_Write(0x01);/\* X axis gyroscope reference frequency\*/

I2C\_Stop();

I2C\_Start\_Wait(0xD0);

I2C\_Write(CONFIG);/\* Write to Configuration register\*/

I2C\_Write(0x00);/\* Fs = 8KHz\*/

I2C\_Stop();

I2C\_Start\_Wait(0xD0);

I2C\_Write(GYRO\_CONFIG);/\* Write to Gyro configuration register\*/

I2C\_Write(0x18);/\* Full scale range 2000 degree/C \*/

I2C\_Stop();

I2C\_Start\_Wait(0xD0);

I2C\_Write(INT\_ENABLE);/\* Write to interrupt enable register\*/

I2C\_Write(0x01);

I2C\_Stop();

}

void I2C\_Start(void)

{

// send the START condition

TWCR0 |= (1<<TWINT)|(1<<TWEN)|(1<<TWSTA);

while((TWCR0 & (1 << TWINT)) == 0);

}

/\*

void MPU6050\_writereg(uint8\_t reg, uint8\_t val){

I2C\_Start(MPU6050+I2C\_Write);

I2C\_Write(reg);//go to register

I2C\_Write(val);//set value

I2C\_Stop();//set stop condition = bus release

}

uint8\_t MPU6050\_readreg(uint8\_t reg){

I2C\_Start\_Wait(MPU6050+I2C\_Write);

I2C\_Write(reg);

I2C\_rep\_start(MPU6050+I2C\_Read);

raw = I2C\_ReadACK();

raw = (raw<<8)|i2creadNAK();

I2C\_Stop();

return raw;

}\*/

void usart\_init(void){

//turn on Rx, TX

UCSR0B |= (1<<RXEN0)|(1<<TXEN0);

//8bit char sizes

UCSR0C |= (1<<UCSZ01)|(1<<UCSZ00);

//set baud rate

UBRR0H = (BAUD\_PRESCALE >> 8);

UBRR0L = BAUD\_PRESCALE;

//Enable the USART Receive interrupt

UCSR0B |= (1<<RXCIE0);

//globally enable interrupts

//sei();

}

void usart\_send(char ch){

while(! (UCSR0A & (1<<UDRE0))); //wait until UDR0 is empty

UDR0 = ch; //transmit ch

}

void UART\_sendString(const char \*str){

while(\*str){

usart\_send(\*str++);

}

}

1. **GITHUB LINK OF THIS DA**

<https://github.com/kirkster96/submission_da/tree/master/DesignAssignment/DA6>

**Student Academic Misconduct Policy**

<http://studentconduct.unlv.edu/misconduct/policy.html>

“This assignment submission is my own, original work”.

Cameron Kirk