CPE301 – SPRING 2021

Design Assignment 6

Student Name: Brian Wolak

Student #: 2000509437

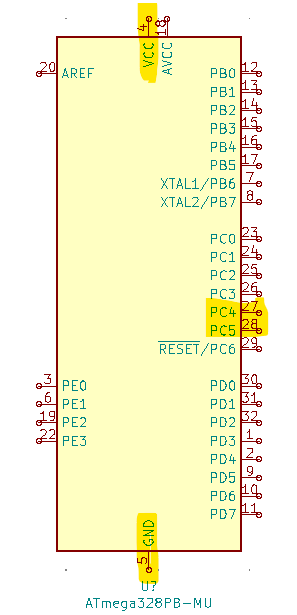
Student Email: wolak@unlv.nevada.edu

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

Directory: [submission\_da/DA\_6 at main · brianwolak/submission\_da (github.com)](https://github.com/brianwolak/submission_da/tree/main/DA_6)

**Task 1:**

In this design assignment a C code is written to interface a 9-DOF ICM-20948 sensor with the ATmega328pb using i2C. The goal was to gather information from the accelerometer and gyro for each axis (x,y,z). Below you can see the code, demonstration video link, and outputs verifying the proper operation.



*Design Assignment 6 Ports Used*

**Video Link:**

<https://youtu.be/EF3NKGJFo8Q>

**C Code:**

#define *F\_CPU* 16000000UL //CPU clock speed

#define BAUD 9600 //Baud rate is 9600

#include <avr/common.h>

#include <avr/io.h>

#include <avr/interrupt.h>

#include <util/setbaud.h>

#include <stdio.h>

#include <util/delay.h>

#include <inttypes.h>

#include <util/twi.h>

#define SCL\_CLOCK 100000L

#define TW\_STATUS\_PB (TWSR0 & 0xF8)

char display[20], display2[20];

*uint8\_t* who\_am\_i; //device address

*uint16\_t* Xacc, Yacc, Zacc, Xgyr, Ygyr, Zgyr;

float Xaccf, Yaccf, Zaccf, Xgyrf, Ygyrf, Zgyrf;

//UART initialize

void USART\_init(void)

{

UBRR0H = *UBRRH\_VALUE*;

UBRR0L = *UBRRL\_VALUE*;

UCSR0C = \_BV(UCSZ01) | \_BV(UCSZ00);

UCSR0B = \_BV(RXEN0) | \_BV(TXEN0);

}

//UART print function

void USART\_tx\_string(char \*data){

while((\*data != '\0')){ //while the data string is not empty

while(!(UCSR0A & (1 << UDRE0))); //while the data regsiter is not empty

UDR0 = \*data; //UDR0 register receives data

data++; //next data value

}

}

void i2c\_init(void)

{/\* initialize TWI clock: 100 kHz clock, TWPS = 0 => prescaler = 1 \*/

TWSR0 = 0; /\* no prescaler \*/

TWBR0 = ((*F\_CPU*/SCL\_CLOCK)-16)/2; /\* must be > 10 for stable operation 12\*/

}/\* i2c\_init \*/

unsigned char i2c\_start(unsigned char address)

{

*uint8\_t* twst;

// send START condition

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

// wait until transmission completed

while(!(TWCR0 & (1<<TWINT)));

// check value of TWI Status Register. Mask prescaler bits.

twst = TW\_STATUS\_PB & 0xF8;

if ( (twst != *TW\_START*) && (twst != *TW\_REP\_START*)) return 1;

// send device address

TWDR0 = address;

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

// wail until transmission completed and ACK/NACK has been received

while(!(TWCR0 & (1<<TWINT)));

// check value of TWI Status Register. Mask prescaler bits.

twst = TW\_STATUS\_PB & 0xF8;

if ( (twst != *TW\_MT\_SLA\_ACK*) && (twst != *TW\_MR\_SLA\_ACK*) ) return 1;

return 0;

}/\* i2c\_start \*/

unsigned char i2c\_rep\_start(unsigned char address)

{

return i2c\_start( address );

}/\* i2c\_rep\_start \*/

void i2c\_stop(void)

{

/\* send stop condition \*/

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

// wait until stop condition is executed and bus released

while(TWCR0 & (1<<TWSTO));

}/\* i2c\_stop \*/

unsigned char i2c\_write( unsigned char data )

{

*uint8\_t* twst;

// send data to the previously addressed device

TWDR0 = data;

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

// wait until transmission completed

while(!(TWCR0 & (1<<TWINT)));

// check value of TWI Status Register. Mask prescaler bits

twst = TW\_STATUS\_PB & 0xF8;

if( twst != *TW\_MT\_DATA\_ACK*) return 1;

return 0;

}//\* i2c\_write \*

unsigned char i2c\_readAck(void)

{

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

while(!(TWCR0 & (1<<TWINT)));

return TWDR0;

}/\* i2c\_readAck \*/

unsigned char i2c\_readNak(void)

{

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

while(!(TWCR0 & (1<<TWINT)));

return TWDR0;

}/\* i2c\_readNak \*/

void i2c\_start\_wait(unsigned char address)

{

*uint8\_t* twst;

while ( 1 )

{

// send START condition

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

// wait until transmission completed

while(!(TWCR0 & (1<<TWINT)));

// check value of TWI Status Register. Mask prescaler bits.

twst = TW\_STATUS\_PB & 0xF8;

if ( (twst != *TW\_START*) && (twst != *TW\_REP\_START*)) continue;

// send device address

TWDR0 = address;

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

// wail until transmission completed

while(!(TWCR0 & (1<<TWINT)));

// check value of TWI Status Register. Mask prescaler bits.

twst = TW\_STATUS\_PB & 0xF8;

if ( (twst == *TW\_MT\_SLA\_NACK* )||(twst ==*TW\_MR\_DATA\_NACK*) )

{/\* device busy, send stop condition to terminate write operation \*/

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

// wait until stop condition is executed and bus released

while(TWCR0 & (1<<TWSTO));

continue;

}

//if( twst != TW\_MT\_SLA\_ACK) return 1;

break;

}

}/\* i2c\_start\_wait \*/

//who\_am\_i function

*uint16\_t* who\_am\_i\_func(void){

*uint16\_t* data;

i2c\_start\_wait(0xD0); //open write communication with 20948

i2c\_write(0x00); //select register address

i2c\_start\_wait(0xD1); //open communication again

data = i2c\_readNak(); //read data value

i2c\_stop(); //stop transmission

return data;

}

//get 16-bit data function

int get\_data(*uint16\_t* addressL, *uint16\_t* addressH){

*uint16\_t* data;

i2c\_start\_wait(0xD0); //start communication with sensor

i2c\_write(addressH); //send high register address

i2c\_start\_wait(0xD1); //open communication

data = i2c\_readNak(); //read high register

i2c\_stop(); //stop transmission

i2c\_start\_wait(0xD0); //open write

i2c\_write(addressL); //send low address

i2c\_start\_wait(0xD1); //open communication

data = (8 << data) | i2c\_readNak(); //read values

i2c\_stop(); //stop transmission

return data;

}

int main(void)

{

int device; //device read address

USART\_init(); //function call

i2c\_init(); //initialize i2c

USART\_tx\_string("Connection Established!\r\n"); //UART connection confirm

i2c\_start\_wait(0xD0); //open communication with sensor

i2c\_write(0x06); //select register 6

i2c\_write(0x00); //write value to the register location

i2c\_stop(); //stop i2c

//who\_am\_i

who\_am\_i = who\_am\_i\_func();

//print who\_am\_i info

*sprintf*(display, "%d", who\_am\_i);

USART\_tx\_string("i2c device address is: ");

USART\_tx\_string(display);

USART\_tx\_string("\r\n");

device = who\_am\_i;

while (1){

who\_am\_i = who\_am\_i\_func(); //who\_am\_i check

if (who\_am\_i == device) //if true, device connected

USART\_tx\_string("ICM-20948 is still Connected\r\n");

else //if false, device not connected

USART\_tx\_string("ICM-20948 is NOT Connected\r\n");

//gyro X

Xgyr = get\_data(0x33, 0x34);

Xgyrf = Xgyr / 131.0; //scale with sensitivity factor

//print gyro X info

*sprintf*(display, "%.2f", Xgyrf);

USART\_tx\_string("Gyro X Value: ");

USART\_tx\_string(display);

USART\_tx\_string("\r\n");

//gyro Y

Ygyr = get\_data(0x35, 0x36);

Ygyrf = Ygyr / 131.0; //scale with sensitivity factor

//print gyro Y info

*sprintf*(display, "%.2f", Ygyrf);

USART\_tx\_string("Gyro Y Value: ");

USART\_tx\_string(display);

USART\_tx\_string("\r\n");

//gyro Z

Zgyr = get\_data(0x37, 0x38);

Zgyrf = Zgyr / 131.0; //scale with sensitivity factor

//print gyro Z info

*sprintf*(display, "%.2f", Zgyrf);

USART\_tx\_string("Gyro Z Value: ");

USART\_tx\_string(display);

USART\_tx\_string("\r\n");

//accelerometer X

Xacc = get\_data(0x2D, 0x2E);

Xaccf = Xacc / 16384.0; //scale with sensitivity factor

//print accel X info

*sprintf*(display, "%.2f", Xaccf);

USART\_tx\_string("Accelerometer X Value: ");

USART\_tx\_string(display);

USART\_tx\_string("\r\n");

//accelerometer Y

Yacc = get\_data(0x2F, 0x30);

Yaccf = Yacc / 16384.0; //scale with sensitivity factor

//print accel Y info

*sprintf*(display, "%.2f", Yaccf);

USART\_tx\_string("Accelerometer Y Value: ");

USART\_tx\_string(display);

USART\_tx\_string("\r\n");

//Accelerometer Z

Zacc = get\_data(0x31, 0x32);

Zaccf = Zacc / 16384.0; //scale with sensitivity factor

//print accel Z info

*sprintf*(display, "%.2f", Zaccf);

USART\_tx\_string("Accelerometer Z Value: ");

USART\_tx\_string(display);

USART\_tx\_string("\r\n");

USART\_tx\_string("\r\n");

*\_delay\_ms*(500); //terminal delay

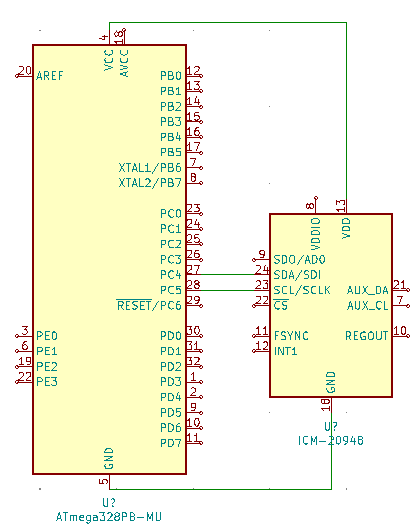
}

}

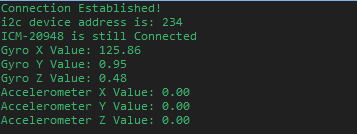
A picture containing text

Description automatically generated

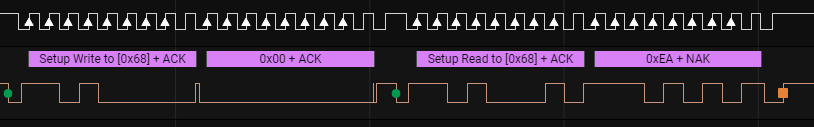
*Design Assignment 6 Board Setup*



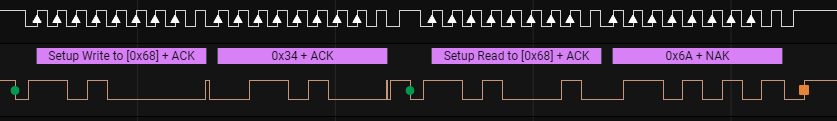
*Design Assignment 6 Circuit*

**

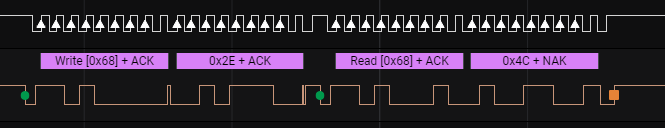
*Terminal Output Showing Initial Connection and Confirmation of Connection*

**

*Logic Output Showing who\_am\_i Reading*

**

*Logic Output Showing X Gyro Scope L Read*

**

*Logic Output Showing X Accelerometer L Read*

**GitHub Link:**

[submission\_da/DA\_6 at main · brianwolak/submission\_da (github.com)](https://github.com/brianwolak/submission_da/tree/main/DA_6)

**Video Link:**

<https://youtu.be/EF3NKGJFo8Q>