Lab 6

Michael Arboleda
Lab Section: 7F34

July 19, 2017

b. Answers to all pre-lab questions

1) What is the highest speed of communication the IMU can handle?

   ANS: The highest speed of communication the IMU can handle is 10MHz

2) In which order should you transmit data to the LSM330? LSB first or MSB first?

   ANS: Data for the LSM330 should be transmitted with MSB first

3) When using SPI, why do we have to write data in order to be able to read data?

   ANS: Since SPI is synchronous and transmits bi-directionally, the master must transmit data so data from the slave can be acquired.

4) How are the accelerometer and gyroscope enabled?

   ANS: The accelerometer is enabled by setting $\overline{SSA}$ to be true and SEN_SEL to 1. The gyroscope is enabled by setting $\overline{SSG}$ to be true and SEN_SEL to 0.

5) Why is it a good idea to modify global flag variables inside ISRs instead of doing everything inside of them?

   ANS: It is a good idea to not do everything inside an ISR because ISR stops processes that are running normally. Thus it is good to have the least amount of code/operation inside an ISR so that the program can continue to execute normally

c. Problems Encountered

Problems encounter was that the wrong data was being outputted. This is because in the LSM_READ, the return statement returned spi_read, but this happens after the selects for both the accelerometer or gyroscope are off, so it was never actually reading data from the LSM

d. Future Work/Applications

Future application would be using the LED with accelerometer. This could be change the color once a certain value with the accelerometer is reach. This could then be used for devices that are remote controlled.

e. Schematics

N/A

g. Pseudocode/Flowcharts

Pseudocode for Lab6b.c:

```
MAIN:
    * Call Change_CLK_32HZ Function
    * Call spi_init Function

    WHILE(TRUE){
        spiWrite(0x53)
    }
END

FUNCTION spi_init
    * Set up port F
    * Set up port A
    * Set up SPI ctrl

FUNCTION spiWrite
    * Write to DATA reg

    WHILE(IF flag not set){}

    * Return data in SPIF.DATA

FUNCTION spiRead
    * return spiWrite(0xFF)

FUNCTION Change_CLK_32HZ
    * Enable the new oscillator

    WHILE(OSC FLAG not set){}

    * Write the IOREG signature to the CPU_CCP reg
    * Select the new clock source in the CLK_CTRL reg
```

Pseudocode for Lab6f.c:

```
MAIN:
    * Call Change_CLK_32HZ Function
    * Call spi_init Function
    * Call USART_INIT Function
    * Call spi_init Function
    * Call accel_init Function
    * Call gyro_init Function
    * Set up interrupts
    * Clear data in Gyro

    WHILE(TRUE){
        IF(accelerometer data is ready){
            * Call READ_DATA_ACCEL() Function
        }
        IF(gyro data is ready){
            * Call READ_DATA_GYRO() Function
        }
    }

END

FUNCTION LSM_READ
    IF(GYRO selected){
        * Select/Enable Gyro
    }
    IF(ACCEL selected){
        * Select/Enable ACCEL
    }

    * write adrres
    * save data at address

    IF(GYRO selected){
        * disable Gyro
    }
    IF(ACCEL selected){
         * disable ACCEL
    }

    * return saved data
```

```
FUNCTION LSM_WRITE
    IF(GYRO selected){
        * Select/Enable Gyro
    }
    IF(ACCEL selected){
        * Select/Enable ACCEL
    }

    * write adrres
    * write data

    IF(GYRO selected){
        * disable Gyro
    }
    IF(ACCEL selected){
        * disable ACCEL
    }

FUNCTION accel_init
    * Set up PORT C interrupt
    * route the DRDY signal to INT1_A and enable
      INT1 with a rising edge (active high)
    * configure the accelerometer to have the highest
      possible output data rate as well as enable the X, Y,
      and Z axes.

FUNCTION gyro_init
    * Set up PORT A interrupt
    * GYRO_ENABLE bit on PORTA
    * configure the gyroscope to have the highest possible
      output data rate as well as enable the X, Y, and Z axes.
    * Set the I2_DRDY bit
    * Choose 2000 dps for the full-scale selection bits

ISR(PORTA_INT0_vect)
    * Preserve Status reg
    * Set int flags
    * Set global var flag
    * Restore Status reg

ISR(PORTC_INT0_vect)
    * Preserve Status reg
    * Set int flags
    * Set global var flag
    * Restore Status reg
```

```
FUNCTION DISPLAY_GYROACCEL_INFO
    * Transmit start byte
    * OUT CHAR ACCEL DATA
    * OUT_CHAR GYRO DATA
    * Transmit end byte

FUNCTION READ_DATA_ACCEL
    * Read accel data
    * Reset accel global flag

FUNCTION READ_DATA_GYRO
    * Read gyro data
    * Reset gyro global flag

FUNCTION CLR_GYRO
    * STORE GYRO info

FUNCTION USART_INIT
    * Set port D for USART com
    * Set up Ctrl B and C
    * Set up baud rate

FUNCTION OUT_CHAR
    While(Transmitting){}
    * Return USART Data in a CHAR

FUNCTION spi_init
    * Set up port F
    * Set up port A
    * Set up SPI ctrl

FUNCTION spiWrite
    * Write to DATA reg

    WHILE(IF flag not set){}

    * Return data in SPIF.DATA
```

```
FUNCTION spiRead
    * return spiWrite(0xFF)

FUNCTION Change_CLK_32HZ
    * Enable the new oscillator

    WHILE(OSC FLAG not set){}

    * Write the IOREG signature to the CPU_CCP reg
    * Select the new clock source in the CLK_CTRL reg
```

h. Program Code

Code for Lab6b.c:

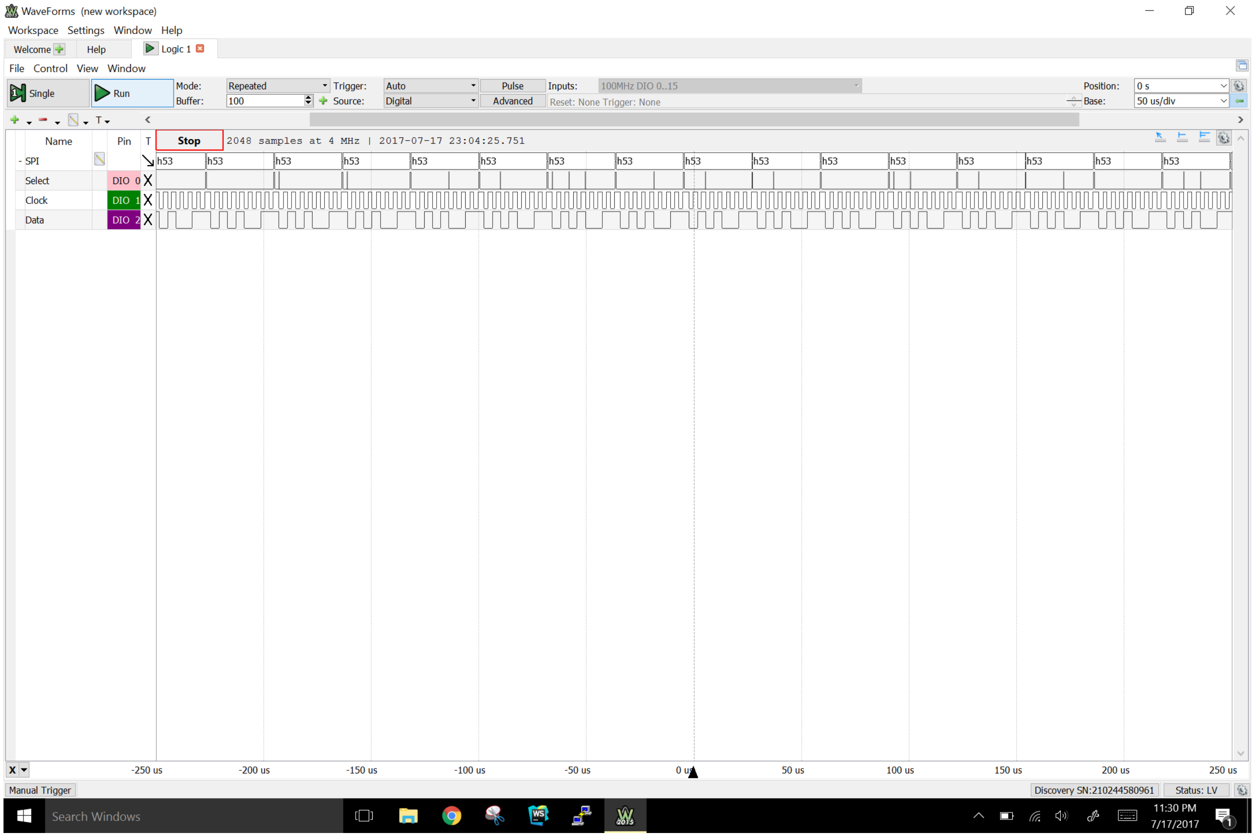
```
/*Lab 6 Part B
Name: Michael Arboleda
Section #: 7F34
TA Name: Wesley Piard
Description: SPI COMMUNICATION TESTING

Lab6b.c
Created: 7/17/2017 4:49:59 AM
*/

#include <avr/io.h>
#include "constants.h"
#include "Clk_32MHz.h"
#include "SPI.h"


int main(void){
    //Set up program
   Change_CLK_32HZ();
   spi_init();

   //Loop for sending 0x53 continuously
   while(TRUE){
      spiWrite(0x53);
   }
}
```

Code for Lab6f.c

```c
/* Lab6 Part F
Name: Michael Arboleda
Section #: 7F34
TA Name: Wesley Piard
Description: Output data for accel
   and gyro

Lab6f.c
Created: 7/18/2017 12:47:12 AM
*/

#include <avr/io.h>
#include <avr/interrupt.h>
#include "constants.h"
#include "Clk_32MHz.h"
#include "USART.h"
#include "SPI.h"
#include "LSM.h"
#include "LSM330.h"

int main(void){
   //Set up code
   Change_CLK_32HZ();
   USART_INIT();
   spi_init();
   accel_init();
   gyro_init();

   //Set up interrupts
   PMIC.CTRL = 0x07;
   sei();

   //Clear data in Gyro
   CLR_GYRO();

   //WHILE(1)
   while(TRUE){
      //IF accel data is ready
      if(accelDataReady){
         READ_DATA_ACCEL();
      }

      //IF Gyro data is ready
      if(gyroDataReady){
         READ_DATA_GYRO();
      }

      //Display data through USART
      DISPLAY_GYROACCEL_INFO();
   }
```

```
  // Return with status 0
  return 0;
}
```

i. Appendix



WaveForms (new workspace)

Mode: Repeated
Buffer: 100
Trigger: Auto
Source: Digital
Inputs: 100MHz DIO 0..15
Reset: None Trigger: None
Position: 0 s
Base: 50 us/div

```
2048 samples at 4 MHz | 2017-07-17 23:04:25.751
```

Name	Pin
SPI	
Select	DIO 0
Clock	DIO 1
Data	DIO 2

Discovery SN:210244580961
Status: LV

Figure 1: DAD reading of SPI

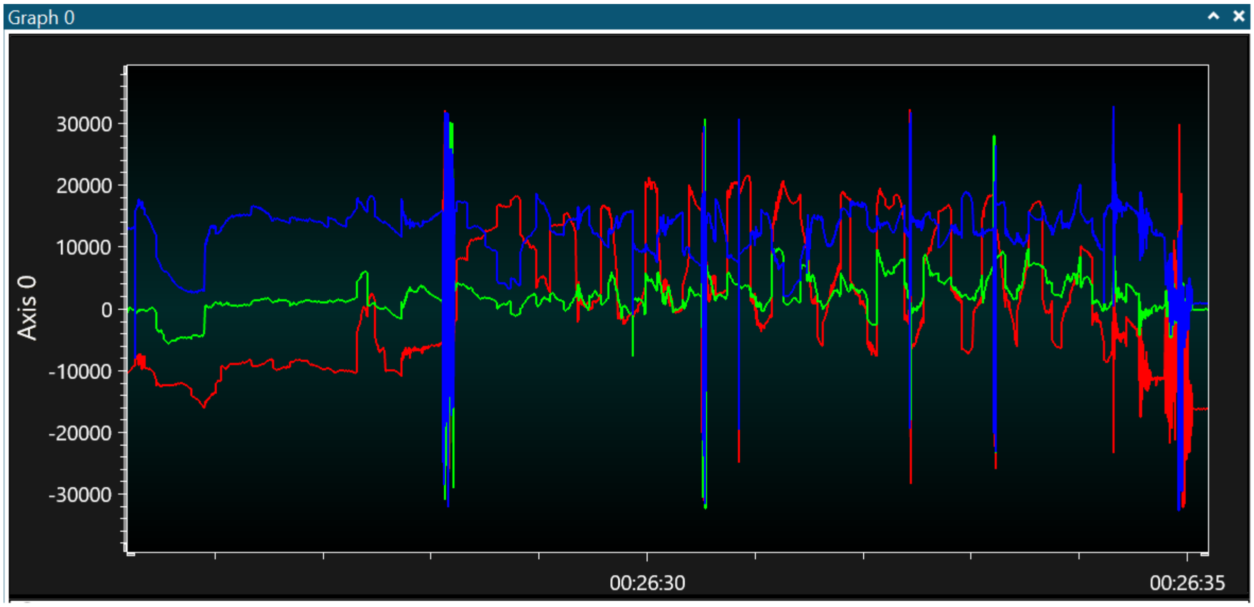


Figure 2: Graph of accelerometer data

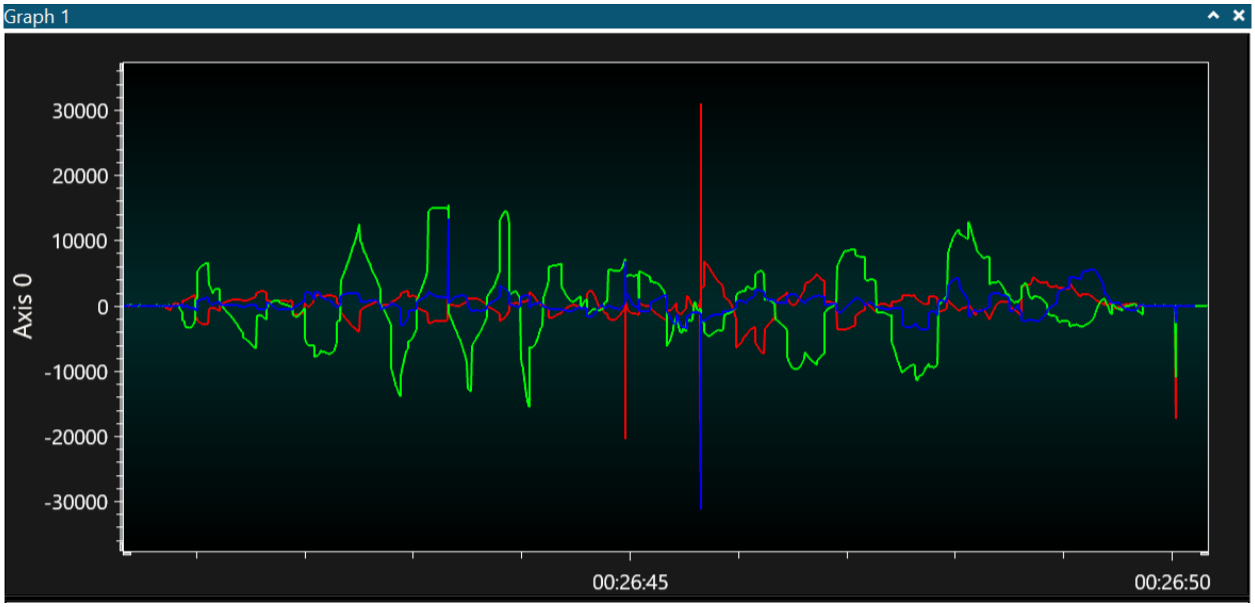


Figure 3: Graph of gyroscope data

Code for Clk_32MHz.h:

```
#ifndef CLK_32MHZ_H_
#define CLK_32MHZ_H_

/*
Name: Michael Arboleda
Section #: 7F34
TA Name: Wesley Piard
Description: Header for CLK_32MHz.c

Clk_32MHz.h
Created: 7/7/2017 11:20:13 PM
*/


//Function Prototype
void Change_CLK_32HZ(void);


#endif /* CLK_32MHZ_H_ */
```

Code for Clk_32MHz.c:

```
/*
Name: Michael Arboleda
Section #: 7F34
TA Name: Wesley Piard
Description: Changes uP freq to 32Mhz

* Clk_32MHz.c
* Created: 7/17/2017 6:21:29 AM
*/


#include <avr/io.h>

//include extern constants
extern const uint8_t NEW_CLOCK_FREQ;


/*********************Function**************************************
; Function Name: Change_CLK_32HZ
; Inputs: No direct input (from stack)
; Outputs: No direct outputs
*/
void Change_CLK_32HZ(void){

   //Set the clk config
   OSC_CTRL = NEW_CLOCK_FREQ;

   //Wait for the right flag to be set in the OSC_STATUS reg
   while((OSC_STATUS & PIN1_bm) != PIN1_bm);

   //Write the IOREG signature to the CPU_CCP reg
   CPU_CCP = CCP_IOREG_gc;

   //Select the new clock source in the CLK_CTRL reg
   CLK_CTRL = CLK_SCLKSEL_RC32M_gc;

   return;
}
```

Code for LSM.h:

```
#ifndef LMS_H_
#define LMS_H_

/*
Name: Michael Arboleda
Section #: 7F34
TA Name: Wesley Piard
Description: Header for LMS.c

LMS.h
Created: 7/18/2017 1:18:03 AM
*/

//Function Prototype

uint8_t LSM_READ(uint8_t SS_choice, uint8_t reg);
void LSM_WRITE(uint8_t SS_choice, uint8_t reg, uint8_t reg_data);
void accel_init(void);
void gyro_init(void);
void DISPLAY_GYROACCEL_INFO(void);
void READ_DATA_ACCEL(void);
void READ_DATA_GYRO(void);
void CLR_GYRO(void);


#endif /* LMS_H_ */
```

Code for LSM.c:

```
/*
Name: Michael Arboleda
Section #: 7F34
TA Name: Wesley Piard
Description: LMS functions

LMS.c
Created: 7/18/2017 1:17:10 AM
*/

#include <avr/io.h>
#include <avr/interrupt.h>
#include "USART.h"
#include "SPI.h"
#include "LSM330.h"


//include extern constants
extern const uint8_t PORTC_INTCTRL_CONFIG;
extern const uint8_t PORTC_INT0MASK_CONFIG;
extern const uint8_t PORTC_PIN7CTRL_CONFIG;
extern const uint8_t PORTA_INTCTRL_CONFIG;
extern const uint8_t PORTA_INT0MASK_CONFIG;
extern const uint8_t PORTA_PIN1CTRL_CONFIG;
//FLAGS
extern volatile uint8_t accelDataReady;
extern volatile uint8_t gyroDataReady;


//Set Global(THIS FILE) Variables
//COORDINATES
volatile uint8_t accel_xL = 0;
volatile uint8_t accel_xH = 0;
volatile uint8_t accel_yL = 0;
volatile uint8_t accel_yH = 0;
volatile uint8_t accel_zL = 0;
volatile uint8_t accel_zH = 0;
volatile uint8_t gyro_xL = 0;
volatile uint8_t gyro_xH = 0;
volatile uint8_t gyro_yL = 0;
volatile uint8_t gyro_yH = 0;
volatile uint8_t gyro_zL = 0;
volatile uint8_t gyro_zH = 0;



/*********************Function**************************************
; Function Name: LSM_READ
; Inputs: uint8_t:=SS_choice, uint8_t:=reg
; Outputs: uint8_:=spiRead()
```

```
*/
uint8_t LSM_READ(uint8_t SS_choice, uint8_t reg){
   //IF GYRO
   if(SS_choice == 0){
      PORTF.OUTCLR = PIN2_bm; //SENSOR_SEL
      PORTF.OUTCLR = PIN4_bm; //SSG
   }
   //ELSE IF accelerometer
   else if(SS_choice == 1){
      PORTF.OUTSET = PIN2_bm; //SENSOR_SEL
      PORTF.OUTCLR = PIN3_bm; //SSA
   }

   //Address OR with READ CYCLE enable
   spiWrite( (reg | PIN7_bm) );
   uint8_t result = spiRead();

   //IF GYRO
   if(SS_choice == 0){
      PORTF.OUTSET = PIN4_bm; //SSG
   }
   //ELSE IF accelerometer
   else if(SS_choice == 1){
      PORTF.OUTSET = PIN3_bm; //SSA
   }

   //return from function
   return(result);
}



/*********************Function**************************************
; Function Name: LSM_WRITE
; Inputs: uint8_t:=SS_choice, uint8_t:=reg, uint8_t:=reg_data
; Outputs: No direct outputs
*/
void LSM_WRITE(uint8_t SS_choice, uint8_t reg, uint8_t reg_data){
   //IF GYRO
   if(SS_choice == 0){
      PORTF.OUTCLR = PIN2_bm; //SENSOR_SEL
      PORTF.OUTCLR = PIN4_bm; //SSG
   }
   //ELSE IF accelerometer
   else if(SS_choice == 1){
      PORTF.OUTSET = PIN2_bm; //SENSOR_SEL
      PORTF.OUTCLR = PIN3_bm; //SSA
   }

   //Write reg address then data
   spiWrite(reg);
   spiWrite(reg_data);
   //IF GYRO
   if(SS_choice == 0){
      PORTF.OUTSET = PIN4_bm; //SSG
   }
   //ELSE IF accelerometer
   else if(SS_choice == 1){
      PORTF.OUTSET = PIN3_bm; //SSA
   }

   //return from function
   return;
}



/*********************Function**************************************
; Function Name: accel_init
; Inputs: No direct inputs
; Outputs: No direct outputs
*/
void accel_init(void){
   //Set up interrupt
   PORTC.DIRCLR = PIN7_bm;
   PORTC.INTCTRL = PORTC_INTCTRL_CONFIG;
   PORTC.INT0MASK = PORTC_INT0MASK_CONFIG;
   PORTC.PIN7CTRL = PORTC_PIN7CTRL_CONFIG;

   // route the DRDY signal to INT1_A and enable
   //INT1 with a rising edge (active high)
   LSM_WRITE(1, CTRL_REG4_A, 0xC8);

   //configure the accelerometer to have the highest
   //possible output data rate as well as enable the X, Y,
   //and Z axes.
   LSM_WRITE(1, CTRL_REG5_A, 0x97);

   //return from function
   return;
}



/*********************Function**************************************
; Function Name: gyro_init
; Inputs: No direct inputs
; Outputs: No direct outputs
*/
void gyro_init(void){
   //Set up interrupt
   PORTA.DIRCLR = PIN1_bm;
   PORTA.INTCTRL = PORTA_INTCTRL_CONFIG;
   PORTA.INT0MASK = PORTA_INT0MASK_CONFIG;
   PORTA.PIN1CTRL = PORTA_PIN1CTRL_CONFIG;

   //GYRO_ENABLE bit on PORTA
   PORTA.OUTSET = PIN3_bm;

   //configure the gyroscope to have the highest possible
   //output data rate as well as enable the X, Y, and Z axes.
   LSM_WRITE(0, CTRL_REG1_G, 0xCF);

   //Set the I2_DRDY bit
   LSM_WRITE(0, CTRL_REG3_G, 0x08);

   // Choose 2000 dps for the full-scale selection bits
   LSM_WRITE(0, CTRL_REG4_G, 0x30);

   //Return from function
   return;
}



/*********************ISR**************************************
; Interrupt Type: PORTA_INT0_vect
; Inputs: No direct inputs
; Outputs: No direct outputs
*/
ISR(PORTA_INT0_vect){
   //Preserve Status Reg
   uint8_t temp = CPU_SREG;

   //Set int flags
   PORTA.INTFLAGS = 0x01;

   //Set global var flag
   gyroDataReady = 0x01;

   //Restore Status Reg
   CPU_SREG = temp;

   //Return from function
   return;
}



/*********************ISR**************************************
; Interrupt Type: PORTC_INT0_vect
; Inputs: No direct inputs
; Outputs: No direct outputs
*/
ISR(PORTC_INT0_vect){
   //Preserve Status Reg
   uint8_t temp = CPU_SREG;

   //Set int flags
   PORTC.INTFLAGS = 0x01;

   //Set global var flag
   accelDataReady = 0x01;

   //Restore Status Reg
   CPU_SREG = temp;

   //Return from function
   return;
}



/*********************Function**************************************
; Function Name: DISPLAY_GYROACCEL_INFO
; Inputs: No direct inputs
; Outputs: No direct outputs
*/
void DISPLAY_GYROACCEL_INFO(void){
   //Transmit start byte as seen in example
   OUT_CHAR(0x03);

   //OUT CHAR ACCEL DATA
   OUT_CHAR(accel_xL);
   OUT_CHAR(accel_xH);
   OUT_CHAR(accel_yL);
   OUT_CHAR(accel_yH);
   OUT_CHAR(accel_zL);
   OUT_CHAR(accel_zH);

   //OUT_CHAR GYRO DATA
   OUT_CHAR(gyro_xL);
   OUT_CHAR(gyro_xH);
   OUT_CHAR(gyro_yL);
   OUT_CHAR(gyro_yH);
   OUT_CHAR(gyro_zL);
   OUT_CHAR(gyro_zH);

   //OUT_CHAR ~0xFC
   OUT_CHAR(0xFC);

   //Return from function
   return;
}
```

```
/*********************Function**************************************
; Function Name: READ_DATA_ACCEL
; Inputs: No direct inputs
; Outputs: No direct outputs
*/
void READ_DATA_ACCEL(void){
   //Read accel data
   accel_xL = LSM_READ(1, OUT_X_L_A);
   accel_xH = LSM_READ(1, OUT_X_H_A);
   accel_yL = LSM_READ(1, OUT_Y_L_A);
   accel_yH = LSM_READ(1, OUT_Y_H_A);
   accel_zL = LSM_READ(1, OUT_Z_L_A);
   accel_zH = LSM_READ(1, OUT_Z_H_A);

   //Reset accel global flag
   accelDataReady = 0;

   //Return from function
   return;
}



/*********************Function**************************************
; Function Name: READ_DATA_GYRO
; Inputs: No direct inputs
; Outputs: No direct outputs
*/
void READ_DATA_GYRO(void){
   //Read Gyro data
   gyro_xL = LSM_READ(0, OUT_X_L_G);
   gyro_xH = LSM_READ(0, OUT_X_H_G);
   gyro_yL = LSM_READ(0, OUT_Y_L_G);
   gyro_yH = LSM_READ(0, OUT_Y_H_G);
   gyro_zL = LSM_READ(0, OUT_Z_L_G);
   gyro_zH = LSM_READ(0, OUT_Z_H_G);

   //Reset gyro global flag
   gyroDataReady = 0;

   //Return from function
   return;
}



void CLR_GYRO(void){

   //STORE GYRO info
   gyro_xH = LSM_READ(0, OUT_X_H_G);
   gyro_xL = LSM_READ(0, OUT_X_L_G);
   gyro_yH = LSM_READ(0, OUT_Y_H_G);
   gyro_yL = LSM_READ(0, OUT_Y_L_G);
   gyro_zH = LSM_READ(0, OUT_Z_H_G);
   gyro_zL = LSM_READ(0, OUT_Z_L_G);

   //Return from function
   return;
}
```

Code for SPI.h:

```
#ifndef SPI_H_
#define SPI_H_

/*
Name: Michael Arboleda
Section #: 7F34
TA Name: Wesley Piard
Description: Header for SPI

SPI.h
Created: 7/17/2017 4:53:25 AM
*/


//Function Prototypes
void spi_init(void);
uint8_t spiWrite(uint8_t data);
uint8_t spiRead();


#endif /* SPI_H_ */
```

Code for SPI.c:

```
/*
Name: Michael Arboleda
Section #: 7F34
TA Name: Wesley Piard
Description: SPI functions

SPI.c
Created: 7/17/2017 6:25:57 AM
*/

#include <avr/io.h>

//include extern constants
extern const uint8_t PORTF_DIRSET_CONFIG;
extern const uint8_t PORTF_DIRCLR_CONFIG;
extern const uint8_t PORTF_OUTSET_CONFIG;
extern const uint8_t PORTA_DIRSET_CONFIG;
extern const uint8_t PORTA_OUTCLR_CONFIG;
extern const uint8_t SPI_CTRL_CONFIG;

/*********************Function**************************************
; Function Name: spi_init
; Inputs: No direct input
; Outputs: No direct outputs
*/
void spi_init(void){

   //Set up port F
   PORTF.DIRSET = PORTF_DIRSET_CONFIG;
   PORTF.DIRCLR = PORTF_DIRCLR_CONFIG;
   PORTF.OUTSET = PORTF_OUTSET_CONFIG;

   //Set up port A
   PORTA.DIRSET = PORTA_DIRSET_CONFIG;
   PORTA.OUTCLR = PORTA_OUTCLR_CONFIG;

   //Set up SPI
   SPIF.CTRL = SPI_CTRL_CONFIG;//0x5F

   //Return from function
   return;
}



/*********************Function**************************************
; Function Name: spiWrite
; Inputs: uint8_t:=DATA
; Outputs: uint8_t:=SPIF.DATA
*/
```

```
uint8_t spiWrite(uint8_t data){
   //Write to DATA reg
   SPIF.DATA = data;

   //Wait for transfer to finish
   while((SPIF.STATUS & 0x80) != 0x80);

   //Return data in SPIF.DATA
   return (SPIF.DATA);
}



/*********************Function**************************************
; Function Name: spiRead
; Inputs: No direct input
; Outputs: uint8_t:=SPIF.DATA
*/
uint8_t spiRead(){
   return(spiWrite(0xFF));
}
```

Code for USART.h:

```
#ifndef USART_H_
#define USART_H_
/*
Name: Michael Arboleda
Section #: 7F34
TA Name: Wesley Piard
Description: Header for USART.h

USART.h
Created: 7/8/2017 3:07:25 AM
 */

//Function Prototypes
void USART_INIT(void);
void OUT_CHAR(uint8_t c);

#endif /* USART_H_ */
```

Code for USART.c:

```
/*
Name: Michael Arboleda
Section #: 7F34
TA Name: Wesley Piard
Description: Set up USART

USART.c
Created: 7/18/2017 5:05:45 AM
*/
#include <avr/io.h>

extern const uint8_t pin_Tx;
extern const uint8_t pin_Rx;
extern const uint8_t TxRx_On;
extern const uint8_t usart_ctrl_C;
extern const uint8_t BSCALE;
extern const uint8_t upper_BSEL;
extern const uint8_t BSEL;
/*********************Function**************************************
; Function Name: USART_INIT
; Inputs: No inputs
; Outputs: No outputs
*/
void USART_INIT(void){
   // Set port D for USART com
   PORTD.DIRSET = pin_Tx;
   PORTD.OUTSET = pin_Tx;
   PORTD.DIRCLR = pin_Rx;

   //Set up Ctrl B and C
   USARTD0_CTRLB = TxRx_On;
   USARTD0_CTRLC = usart_ctrl_C;

   //Set up baud rate
   USARTD0.BAUDCTRLA = BSEL & 0xFF;
   USARTD0.BAUDCTRLB = (BSCALE <<4 & 0xF0);

   return;
}



/*********************Function**************************************
; Function Name: OUT_CHAR
; Inputs: Char c
; Outputs: No outputs
*/
void OUT_CHAR(uint8_t c){

   // Wait until prev receive done
   while((USARTD0.STATUS & PIN5_bm) == 0);

   //Output char thru USART
   USARTD0.DATA = c;

   return;
}
```

Code for constants.h:

```
#ifndef CONSTANTS_H_
#define CONSTANTS_H_

/*
Name: Michael Arboleda
Section #: 7F34
TA Name: Wesley Piard
Description: Changes uP freq to 32Mhz

constants.h
Created: 7/7/2017 11:53:20 PM
*/
#include <avr/io.h>

//OVERALL DEFS
#define TRUE 1
#define FALSE 0
#define NULL 0

//****************** CLK_32MHz.h ***********************************
const uint8_t NEW_CLOCK_FREQ = 0b00000010;



//****************** SPI.h ***********************************
//PORT F
const uint8_t PORTF_DIRSET_CONFIG = 0xBC;//0b10111100
const uint8_t PORTF_DIRCLR_CONFIG = 0x40;//0b01000000
const uint8_t PORTF_OUTSET_CONFIG = 0x18;//0b00011000
//PORT A
const uint8_t PORTA_DIRSET_CONFIG = 0x18;//0b00011000
const uint8_t PORTA_OUTCLR_CONFIG = 0x10;//0b00010000
//SPI
const uint8_t SPI_CTRL_CONFIG = 0x5C;//0b01011100


//****************** LSM.h ***********************************
//PORT C
const uint8_t PORTC_INTCTRL_CONFIG = 0x03;//0b00000011;
const uint8_t PORTC_INT0MASK_CONFIG = 0x80;//0b10000000;
const uint8_t PORTC_PIN7CTRL_CONFIG = 0x02;//0b00000010
//PORT A
const uint8_t PORTA_INTCTRL_CONFIG = 0x03;//0b00000011
const uint8_t PORTA_INT0MASK_CONFIG = 0x02;//0b00000010
const uint8_t PORTA_PIN1CTRL_CONFIG = 0x19;//0b00011001
//FLAGS
volatile uint8_t accelDataReady = 0;
volatile uint8_t gyroDataReady = 0;
```

```
//****************** USART.h ***************************************
const uint8_t pin_Tx = PIN3_bm;
const uint8_t pin_Rx = PIN2_bm;
const uint8_t TxRx_On = 0b00011000; // 0x18
// asynch, 8 databits, no parity, 1 start, and 1 stop
const uint8_t usart_ctrl_C = 0b00000011;//
// BSEL 1M
const uint8_t upper_BSEL = 0b00000000;
const uint8_t BSEL = 32;
const uint8_t BSCALE = -5; // 1 BSCALE


#endif /* CONSTANTS_H_ */
```