**REQUIREMENTS NOT MET**

N/A

**PROBLEMS ENCOUNTERED**

N/A

**FUTURE WORK/APPLICATIONS**

Applications of this includes development of communications for basically all microcontrollers and peripheral devices, as SPI is faster than UART.

**PRE-LAB EXERCISES**

**i. In regard to SPI communication that is to exist between the relevant ATxmega128A1U and IMU chips, answer each of the questions within the previously given bulleted list**

**Which device(s) should be given the role of master and which device(s) should be given the role of student?**

**The IMU should be the slave, and the ATX should be the master**

**How will the student device(s) be enabled? If a student select is utilized, rather than just have the device(s) be permanently enabled, which pin(s) will be used?**

**The slave will be enabled using its chip select.  
 The chip select of the slave (pin 12) will be connected to the slave select of the ATX(port F pin 5).**

**What is the order of data transmission? Is the MSb or LSb transmitted first?**

**The IMU transmits data MSB first.  
 The ATX can transmit data either LSB or MSB first**

**In regard to the relevant clock signal, should data be latched on a rising edge or on a falling edge?**

**The IMU transmits and receives data on a rising clock edge. So the data should be latched then.**

**What is the maximum serial clock frequency that can be utilized by the relevant devices?**

**The ATX can transmit/receive data at a max rate of 1MHZ.  
 However, the IMU can transmit at a max rate of 10MHZ  
 Therefore, the ATX will need to receive at its max frequency of 1MHZ**

**PSEUDOCODE/FLOWCHARTS**

**SECTION 2**

**A diagram of a flowchart

Description automatically generated  
Figure 1: Flowchart for “lab6\_2.C”**

**SECTION 3**

**A diagram of a program

Description automatically generated**

**Figure 2: flowchart for “lab6\_3.C”**

**PROGRAM CODE**

**SECTION 2**

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Lab 6, Section 2

//Name: Steven Miller

//Class #: 11318

//PI Name: Anthony Stross

//Description: continuously sends data over spi

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*DEPENDENCIES\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <avr/io.h>

#include "spi.h"

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*END OF DEPENDENCIES\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*MAIN\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int main(void)

{

spi\_init();

while(1)

{

//turn on chip select

PORTF.OUTCLR = SS\_bm;

spi\_write(0x2a);

//turn off chip select

PORTF.OUTSET = SS\_bm;

}

return 0;

}

**SECTION 3**

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Lab 6, Section 3

//Name: Steven Miller

//Class #: 11318

//PI Name: Anthony Stross

//Description: allows communication with IMU

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*DEPENDENCIES\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include "lsm6dsl.h"

#include "lsm6dsl\_registers.h"

#include <avr/io.h>

#include "spi.h"

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*END OF DEPENDENCIES\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int main(void)

{

spi\_init();

//read "who am i?" register

while(1)

{

*uint8\_t* identity = lsm\_read(WHO\_AM\_I);

}

//uint8\_t identity = lsm\_read(WHO\_AM\_I);

return 0;

}

**APPENDIX**

**A white rectangular object with black lines

Description automatically generated  
Figure x: Measurement of “lab6\_2.C”**

**SPI.H**

#ifndef SPI\_H\_ // Header guard.

#define SPI\_H\_

/\*------------------------------------------------------------------------------

spi.h --

Description:

Provides function prototypes and macro definitions for utilizing the SPI

system of the ATxmega128A1U.

Author(s): Dr. Eric M. Schwartz, Christopher Crary, Wesley Piard

Last modified by: Dr. Eric M. Schwartz

Last modified on: 8 Mar 2023

------------------------------------------------------------------------------\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*DEPENDENCIES\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <avr/io.h>

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*END OF DEPENDENCIES\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*MACROS\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#define SS\_bm (1<<4)

#define MOSI\_bm (1<<5)

#define MISO\_bm (1<<6)

#define SCK\_bm (1<<7)

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*END OF MACROS\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*FUNCTION PROTOTYPES\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*------------------------------------------------------------------------------

spi\_init --

Description:

Initializes the relevant SPI module to communicate with the LSM6DSL.

Input(s): N/A

Output(s): N/A

------------------------------------------------------------------------------\*/

void spi\_init(void);

/\*------------------------------------------------------------------------------

spi\_write --

Description:

Transmits a single byte of data via the relevant SPI module.

Input(s): `data` - 8-bit value to be written via the relevant SPI module.

Output(s): N/A

------------------------------------------------------------------------------\*/

void spi\_write(*uint8\_t* data);

/\*------------------------------------------------------------------------------

spi\_read --

Description:

Reads a byte of data via the relevant SPI module.

Input(s): N/A

Output(s): 8-bit value read from the relevant SPI module.

------------------------------------------------------------------------------\*/

*uint8\_t* spi\_read(void);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*END OF FUNCTION PROTOTYPES\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#endif // End of header guard.

**SPI.C**

/\*------------------------------------------------------------------------------

spi.c --

Description:

Provides useful definitions for manipulating the relevant SPI

module of the ATxmega128A1U.

Author(s): Dr. Eric M. Schwartz, Christopher Crary, Wesley Piard

Last modified by: Dr. Eric M. Schwartz

Last modified on: 8 Mar 2023

------------------------------------------------------------------------------\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*DEPENDENCIES\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <avr/io.h>

#include "spi.h"

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*END OF DEPENDENCIES\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*FUNCTION DEFINITIONS\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void spi\_init(void)

{

/\* Initialize the relevant SPI output signals to be in an "idle" state.

\* Refer to the relevant timing diagram within the LSM6DSL datasheet.

\* (You may wish to utilize the macros defined in `spi.h`.) \*/

PORTF.OUTSET = (SS\_bm|MOSI\_bm|SCK\_bm);

/\* Configure the pin direction of relevant SPI signals. \*/

PORTF.DIRSET = (SS\_bm|MOSI\_bm|SCK\_bm) ;

PORTF.DIRCLR = (MISO\_bm);

/\* Set the other relevant SPI configurations. \*/

SPIF.CTRL = SPI\_PRESCALER\_DIV4\_gc | SPI\_MASTER\_bm|SPI\_MODE\_0\_gc|SPI\_ENABLE\_bm| SPI\_CLK2X\_bm;

}

void spi\_write(*uint8\_t* data)

{

/\* Write to the relevant DATA register. \*/

SPIF.DATA = data;

/\* Wait for relevant transfer to complete. \*/

while(SPIF.STATUS != SPI\_IF\_bm)

{

//do nothing while we wait

}

/\* In general, it is probably wise to ensure that the relevant flag is

\* cleared at this point, but, for our contexts, this will occur the

\* next time we call the `spi\_write` (or `spi\_read`) routine.

\* Really, because of how the flag must be cleared within

\* ATxmega128A1U, it would probably make more sense to have some single

\* function, say `spi\_transceive`, that both writes and reads

\* data, rather than have two functions `spi\_write` and `spi\_read`,

\* but we will not concern ourselves with this possibility

\* during this semester of the course. \*/

}

*uint8\_t* spi\_read(void)

{

/\* Write some arbitrary data to initiate a transfer. \*/

SPIF.DATA = 0x37;

/\* Wait for relevant transfer to be complete. \*/

while(SPIF.STATUS != SPI\_IF\_bm)

{

//do nothing while we wait

}

/\* After the transmission, return the data that was received. \*/

return SPIF.DATA;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*END OF FUNCTION DEFINITIONS\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**lsm6dsl.h**

#ifndef LSM6DSL\_H\_ // Header guard.

#define LSM6DSL\_H\_

/\*------------------------------------------------------------------------------

lsm6dsl.h --

Description:

Provides custom data types to make it easier to handle any data

read from the LSM6DSL IMU.

The LSM6DSL can output accelerometer and gyroscope data. Data from both

of these sensors is represented in a 16-bit signed format.

Author(s): Wesley Piard & Leslye Castillo & Dr. Eric M. Schwartz

Last modified by: Dr. Eric M. Schwartz

Last modified on: 29 June 2022

------------------------------------------------------------------------------\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*MACROS\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#define LSM6DSL\_SPI\_READ\_STROBE\_bm 0x80

#define LSM6DSL\_SPI\_WRITE\_STROBE\_bm 0x00

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*END OF MACROS\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <avr/io.h>

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*CUSTOM DATA TYPES\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Used to differentiate the accelerometer and gyroscope within the LSM6DSL. \*/

typedef enum {LSM6DSL\_ACCEL, LSM6DSL\_GYRO} lsm6dsl\_module\_t;

/\* Can be used to contain the separated bytes of data as they are read from

\* the LSM6DSL. \*/

typedef struct lsm6dsl\_data\_raw

{

*uint8\_t* accel\_x\_low, accel\_x\_high;

*uint8\_t* accel\_y\_low, accel\_y\_high;

*uint8\_t* accel\_z\_low, accel\_z\_high;

*uint8\_t* gyro\_x\_low, gyro\_x\_high;

*uint8\_t* gyro\_y\_low, gyro\_y\_high;

*uint8\_t* gyro\_z\_low, gyro\_z\_high;

}lsm6dsl\_data\_raw\_t;

/\* Contains the full concatenated signed 16-bit words of data. \*/

typedef struct lsm6dsl\_data\_full

{

*int16\_t* accel\_x, accel\_y, accel\_z;

*int16\_t* gyro\_x, gyro\_y, gyro\_z;

}lsm6dsl\_data\_full\_t;

/\* Provides the ability to choose how to access the LSM6DSL data. \*/

typedef union lsm6dsl\_data

{

lsm6dsl\_data\_full\_t word;

lsm6dsl\_data\_raw\_t byte;

}lsm6dsl\_data\_t;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*END OF CUSTOM DATA TYPES\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*FUNCTION PROTOTYPES\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void lsm\_write(*uint8\_t* reg\_addr, *uint8\_t* data);

*uint8\_t* lsm\_read(*uint8\_t* reg\_addr);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*END OF FUNCTION PROTOTYPES\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#endif // End of header guard.

**lsm6dsl.C**

/\*------------------------------------------------------------------------------

lsm6dsl.c --

Description:

Brief description of file.

Extended description, if appropriate.

Author(s):

Last modified by: Dr. Eric M. Schwartz

Last modified on: 8 Mar 2023

------------------------------------------------------------------------------\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*DEPENDENCIES\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <avr/io.h>

#include "lsm6dsl.h"

#include "lsm6dsl\_registers.h"

#include "spi.h"

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*END OF DEPENDENCIES\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*FUNCTION DEFINITIONS\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void lsm\_write(*uint8\_t* reg\_addr, *uint8\_t* data)

{

//enable imu by enabling chip select

PORTF.OUTCLR = SS\_bm;

//send over the address bits

//keep in mind that writing to an address in the imu takes 16 cycles(16 bits of data need to be shifted)

//the first bit of the 8 bit address is the strobe bit. Which tells the imu whether we wanna read or write.

//1= read, 0 = write

spi\_write(reg\_addr|LSM6DSL\_SPI\_WRITE\_STROBE\_bm);

//our spi master data register is now filled with junk data,

//we now send out the data we wanna store in the imu.

spi\_write(data);

//disable imu by disabling chip select

PORTF.OUTSET = SS\_bm;

}

*uint8\_t* lsm\_read(*uint8\_t* reg\_addr)

{

//enable imu by enabling chip select

PORTF.OUTCLR = SS\_bm;

//send over the address bits

//keep in mind that reading from an address in the imu takes 16 cycles(16 bits of data need to be shifted)

//the first bit of the 8 bit address is the strobe bit. Which tells the imu whether we wanna read or write.

//1= read, 0 = write

spi\_write(reg\_addr|LSM6DSL\_SPI\_READ\_STROBE\_bm);

//our spi master data register is now filled with junk data,

//we need to perform another read so we can activate the clock and recieve our desired data

spi\_read();

//disable imu by disabling chip select

PORTF.OUTSET = SS\_bm;

return SPIF.DATA;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*END OF FUNCTION DEFINITIONS\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/