**REQUIREMENTS NOT MET**

N/A

**PROBLEMS ENCOUNTERED**

N/A

**FUTURE WORK/APPLICATIONS**

<insert a brief paragraph describing how the topics covered in this lab could potentially be used for other applications>

THE ABOVE SHOULD BE LIMITED TO THE FIRST PAGE, AND NOTHING ELSE SHOULD BE INCLUDED, WHICH ALSO IMPLIES THAT THIS SENTENCE OF TEXT SHOULD BE REMOVED.

**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 program

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

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*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)

{

//turn on chip select

PORTF.OUTCLR = SS\_bm;

/\* 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

}

//turn off chip select

PORTF.OUTSET = SS\_bm;

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

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

int main(void)

{

spi\_init();

while(1)

{

spi\_write(0x2a);

}

return 0;

}

**APPENDIX**

**A black line on a white background

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