CPTR-480
Lab 6 notes

The proper connection between the LIS2MDL break-out board and port D on the microcontroller is this:

Name on PCB	SPI name	Port D pin#	Function
SCL	SCK	5	Clock
SDA	MOSI	6	Master Out Slave In
SDO	MISO	7	Master In Slave out
CS	SS	4	Slave Select (chip select)

The magnitometer defaults to three-wire SPI mode which means that one pin, MOSI, is used by the master to send commands and data and the same pin used to receive data coming from the slave device. For this assignment using only three-wire mode to write and read data is adequate but credit given if you switch to 4-wire mode and that works as well as 3-wire mode.

The magnitometer has a bunch of registers. To do a read, the desired register number (one data byte) is sent to the magnitometer and the magnitometer responds with a one-byte reply (multiple byte replies are also possible, but not required for this assignment). Because a single pin (and hence wire) is being used to both transmit and receive data, the master must configure the MOSI pin as an output before transmitting the register number and then change the pin back to input mode to receive the response from the magnitometer. Also, the chip select must be asserted continuously during the sending of the register number and receiving the data byte from the magnitometer.

NOTE: the first byte sent contains the register number which is a 7-bit value sent right justified in an 8-bit field. The most significant bit is sent first (the MCU must be configured for SPI to send the MSB first). The first bit sent, i.e. the MSB of the byte, is used to specify if the register number being sent is a request to read data from that register or a request to write data into that register (see figures 6, 7, and 8 in the LIS2MDL data sheet). If the first bit sent, the MSB, is zero then a write request is specified with the first byte being sent used as the register number and the second byte sent as the data to be written into the register. If the first bit sent is a one then a read request is specified and the first byte being sent is used as the register from which data will be read. Thus the byte you send must have its MSB set correctly as well as the lower 7 bits with the register number.

The example code in the textbook for initializing the SPI module is close to complete except the configuration needs 3-wire mode, the MOSI pin should begin as an input, and the chip select pin (also called SS) which is Port D pin4 needs to be configures as GPIO in output mode and initially set to logic 1 (rather than being configured for SPI via mux set to Alt2).

The example textbook code for transmitting and receiving data via SPI illustrates the basics. It does need to be extended to handle transmitting multiple bytes (2 bytes) and dealing with the single wire send/receive.

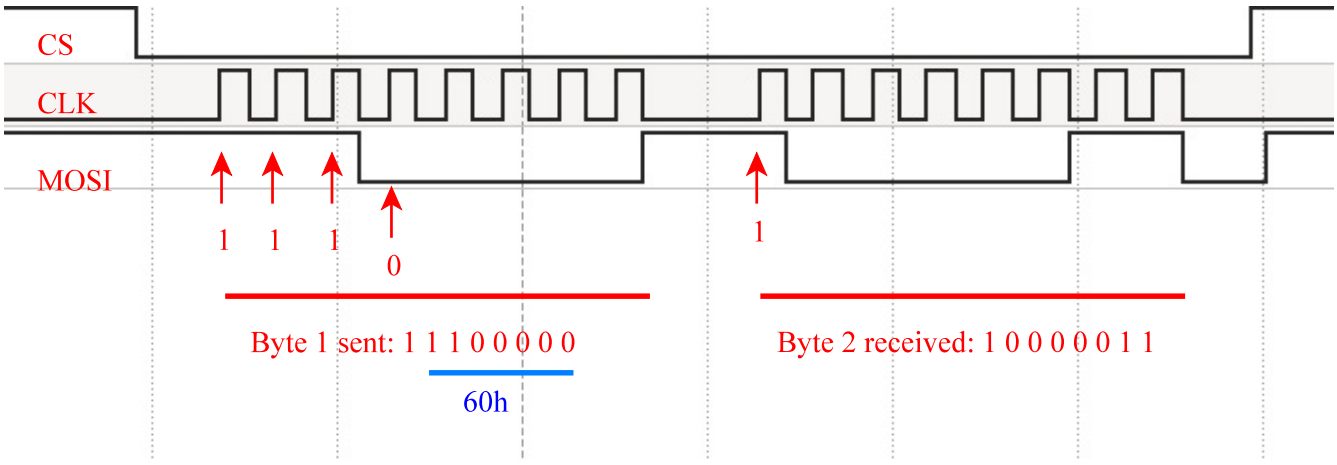
Here is an outline of what is needed to send a read request:

Assert chip select (i.e. SS)
Wait for transmitter empty flag to be set
Turn on the output driver (set a bit in the SPI C2 register)
Put the byte to send in the XMIT register (this is the register number with MSB =1)
Wait for the receive flag to be set
Turn off the output driver
Read SPI data (to empty the receive buffer). Through away, not good data.
Send a second byte (this is a dummy, i.e. needed to cause 8 more clock cycles to be output. Any value will do. For example, 0xFF)
Wait for the receive flag to be set
Read SPI data (this is the good data you are requesting)
De-assert chip select

The outline for sending a write request is similar with these differences:

- Two useful bytes will be sent: register number and data to go into the register
- The MSB of the first byte sent must be zero
- The MOSI pin must remain in output mode until both bytes have been sent
- The second byte sent is not a dummy, it is the data to be placed into a register
- Two reads will still be done but the data read will be the byte values being sent rather than data coming back from the slave device.

Below is an example waveform captured from a read request and the returned data:



The first bit sent is a one, thus this is a read request going to the magnitometer asking for the value in register 60h which is named CFG_REG_A. The second data byte, 83h, is coming from the magnitometer and happens to be a value that was previously written to the register (the value in register 60h was 00h initially).

Note that there is a short time between data bytes when the clock does not change. This is different than the data sheet picture but normal (clock rate here is 654 Khz, about 1.5us per cycle)

The lab 6 handout did not state that you should create functions to do SPI communications, but that is how you should structure your work and the expectation. There should be a function to initialize SPI for the way you are using it and it would be good to have a function for sending a read request to the magnitometer and a function for writing data to a register in the magnitometer.

As you read the magnitometer data sheet you will note that the magnitometer can operate in a mode where it does its calculations and periodically signals the master that new data is available. For this assignment you can use it in single mode. A measurement is initiated by setting the MD0 bit of CFG_REG_A to a one and some time later there is new data. It appears that the STATUS_REG can be read to see if there is new data. And it is possible in 3-wire mode to configure the SDO pin of the magnitometer to be asserted when new data is available. In that case what would be the MCU MISO pin (PortD pin 7) can be configured as GPIO and used to check if new data is available.

And one final thing, make sure you do not write to register numbers listed as reserved in Table-22. There may be content in some register that is loaded at power-up which if over-written may inhibit proper operation of the magnitometer.