829H1 Real-Time Embedded Systems Exercise 4

Candidate No: 105936

12th January 2019

Contents

1. Introduction **3**

2. Experiments **3**
2.1. Serial Communications . . . 3
2.1.1. Using Different SPI Modes . . . 3
2.1.2. Using a Different Number of bits . . . 3
2.2. Linking Two Boards together . . . 4
2.2.1. What happens when MOSI is disconnected . . . 4
2.2.2. What happens when MISO is disconnected . . . 4
2.2.3. What happens when clock is disconnected . . . 5
2.2.4. What happens when the slave select/chip select is removed? . . . 5
2.3. USB . . . 5
2.3.1. Page Scroller . . . 5
2.3.2. Mouse Buttons . . . 5
2.3.3. Accelerometer Mouse . . . 6

3. Conclusion **6**

Appendices **7**

A. Lab Exercise 1 **7**
A.1. Part 1 . . . 7
A.2. Part 2 . . . 7
A.3. Part 3 . . . 8

B. Lab Exercise 2 **9**
B.1. Part 1 - Master Program . . . 9
B.2. Part 2 - Slave Program . . . 10

C. Lab Exercise 3 **11**
C.1. Part 1 . . . 11
C.2. Part 2 . . . 11
C.3. Part 3 . . . 12
C.4. Part 4 . . . 12

References **13**

1. Introduction

This report focuses on using serial communications on the board and shows the work completed during the laboratory sessions and what was learnt. Code listings for some of the created programs can be found in section 3.

2. Experiments

2.1. Serial Communications

For this experiment I created a SPI master object which allowed me to output data using serial methods on three pins, these pins were PTD2 which was the MOSI, PTD3 the MISO and PTD1 which was the clock. fig. 1 demonstrates the output of sending a word out

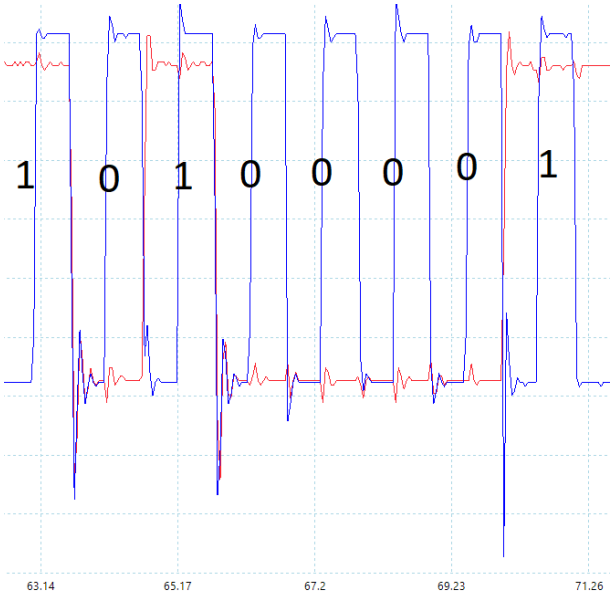

Figure 1: An Image showing the output on the oscilloscope when running code from section A.1

using serial communication. Here the blue line represents the clock and red the signals.

2.1.1. Using Different SPI Modes

This changes the clock signal as shown in fig. 2, fig. 3, fig. 4 and fig. 5. As you can see from the graphs the same word is being transmitted in each configuration. However, the clock signal is changing. This is because we are changing the clock mode through the different phases of the signal.

2.1.2. Using a Different Number of bits

The code for 12-bit operation can be found in section A.2. I am now sending the binary number 100010100001 this is equivalent to 0x8A1 or 2209 in decimal. The oscilloscope output for this can be seen in fig. 6. The figure clearly shows that more bits are being transmitted per word and if you read the values you can see it's transmitting correctly. I

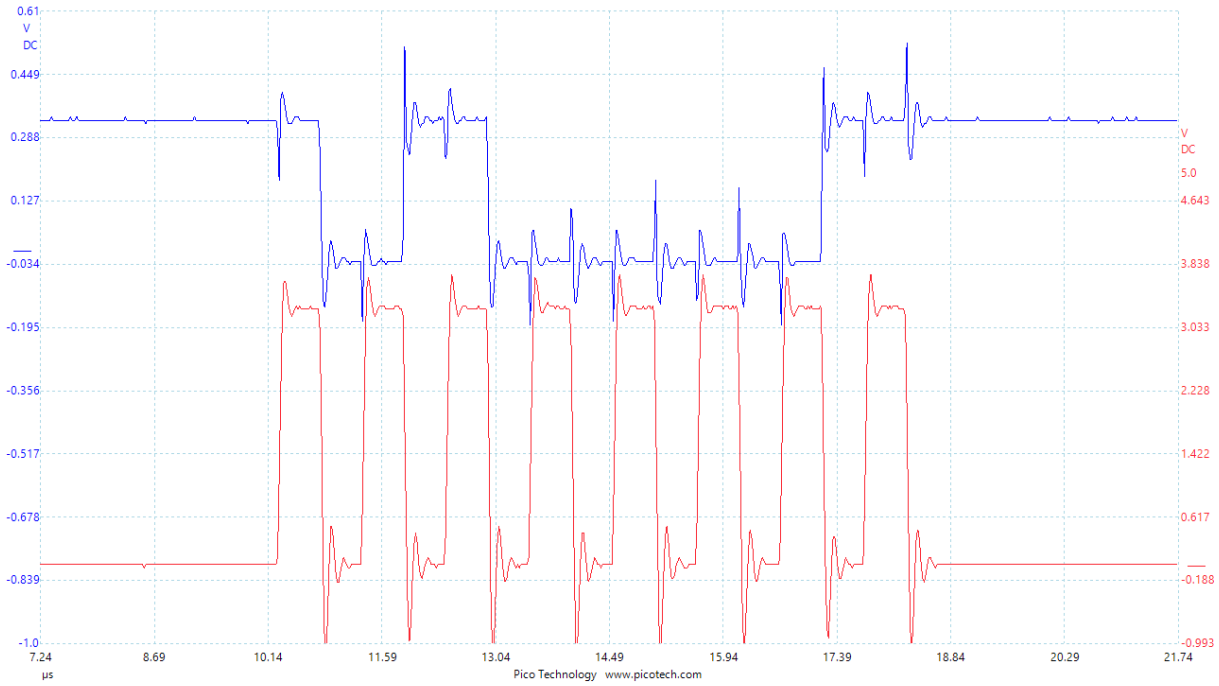

Figure 2: This figure shows clock mode 0 please note that the red is the clock and blue is the word being transmitted

also attempted to display a 16-bit word this was 0x8AA1 or 1000101010100001 in binary this can be seen in fig. 7 and the code can be found in section A.3.

2.2. Linking Two Boards together

This requires the use of a master and slave program the master program allows a slave to be attached to the board using the outputs. It sends a word to the slave board and gets a reply and sets the LEDs accordingly. The slave program receives a word from the master and sets the LEDs accordingly then sends a word back depending on the switch position code for these two programs can be found in section B. This is an example of a serial connection as we have the clock and two data lines one for input and one for output.

2.2.1. What happens when MOSI is disconnected

When the master output/slave input is disconnected the master board still lights up this is because the slave output line is still connected therefore as the master does not know that it is not transmitting it assumes the system is still working correctly. Also, the clock line is still connected so the slave knows when and how to transmit back.

2.2.2. What happens when MISO is disconnected

When the master input/slave output is disconnected nothing works this is a bit of a surprise to me as I expected the master to still be able to control the slave's lights. as the slave is still receiving the master's output.

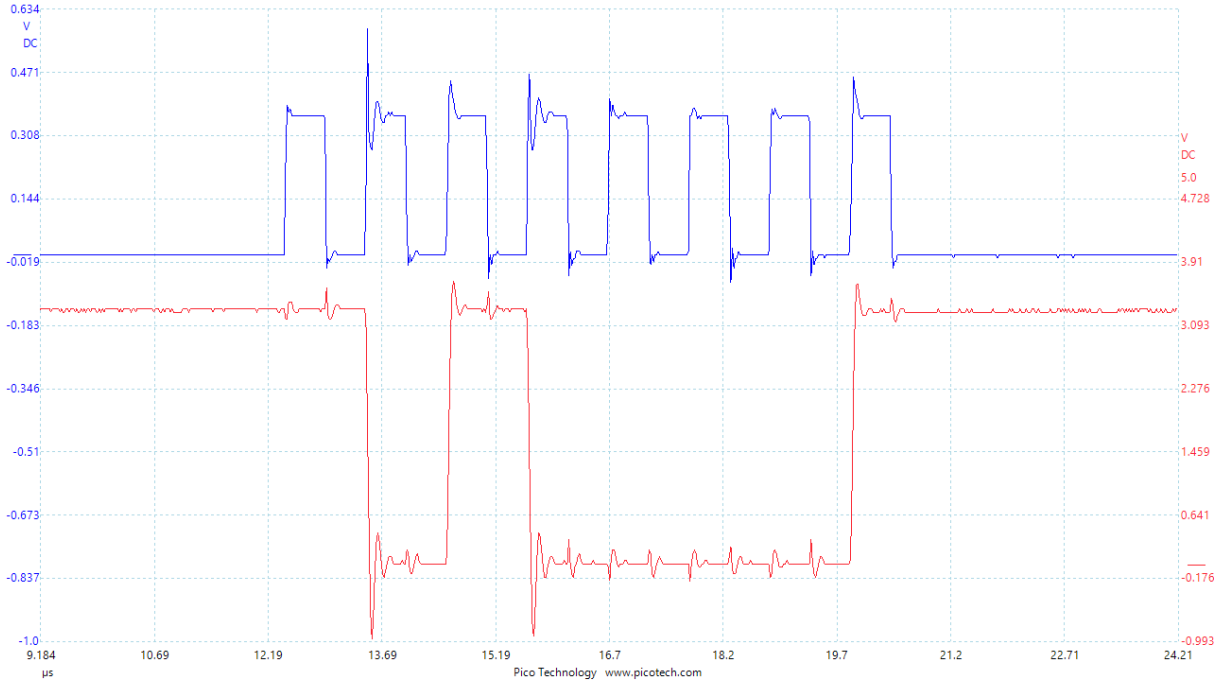

Figure 3: This shows clock mode 1

2.2.3. What happens when clock is disconnected

When the clock line is disconnected none of the functions work this is because the slave is not able to tell where one bit ends, and another begins and it also does not know how to send the data back to the master.

2.2.4. What happens when the slave select/chip select is removed?

When the slave/chip select line is removed again none of the functions work this is because the SPI library system we are using requires this line to be connected.5

2.3. USB

This implements a simple program where the computers mouse is moved on each iteration of the for loop this allows us to test that the USBMouse library is working properly the code for this can be found at section C.1.

2.3.1. Page Scroller

This is another simple program which when run simply continuously scrolls to the bottom of the page. Code for this can be found in section C.2.

2.3.2. Mouse Buttons

For this task I had to use knowledge I gained from earlier exercises. Therefore, I added interrupts on the switches which called the USBMouse.press() and USBMouse.release() the code for this can be found in section C.3.

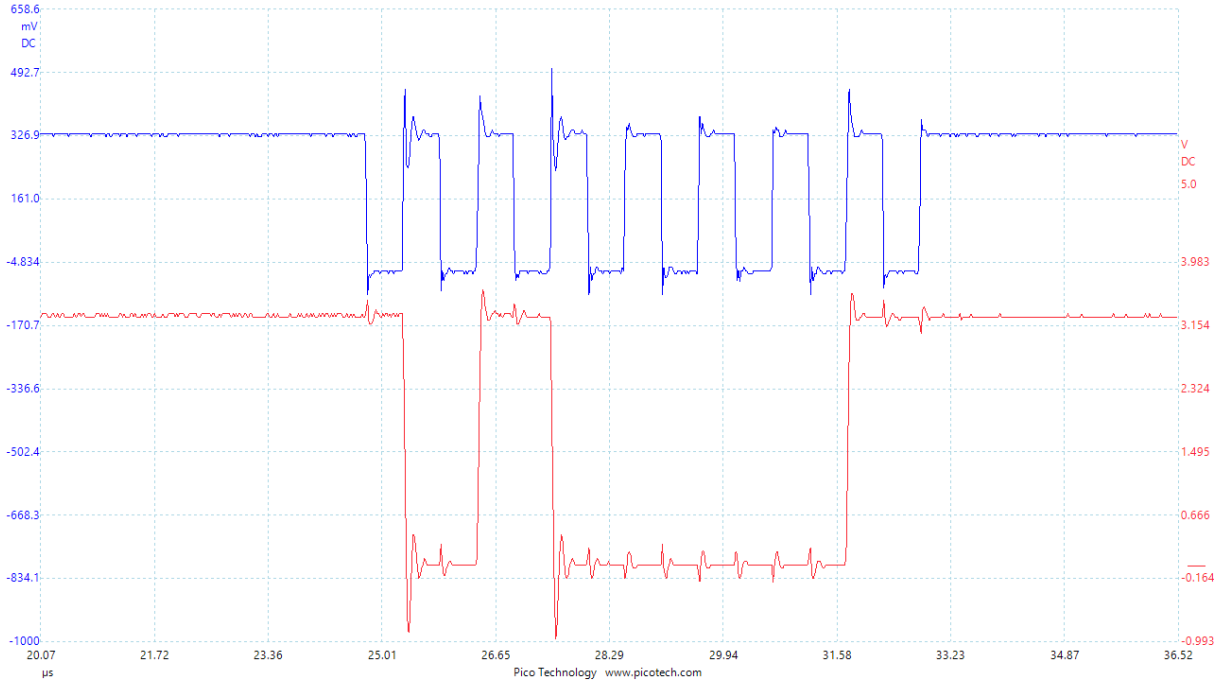

Figure 4: This shows clock mode 2

2.3.3. Accelerometer Mouse

For this task I had to use an additional library to access the accelerometer data. I used the FXOS8700Q[1] library. I then needed to use these values in the accelerator data to make the mouse move, the code for this can be found in section C.4.

3. Conclusion

In conclusion, this was a useful exercise on using serial communications and then learning how to use the USBDevice functions and also how to use the accelerometer values. I would have liked to go on and use the joystick to control the mouse however I ran out of time.

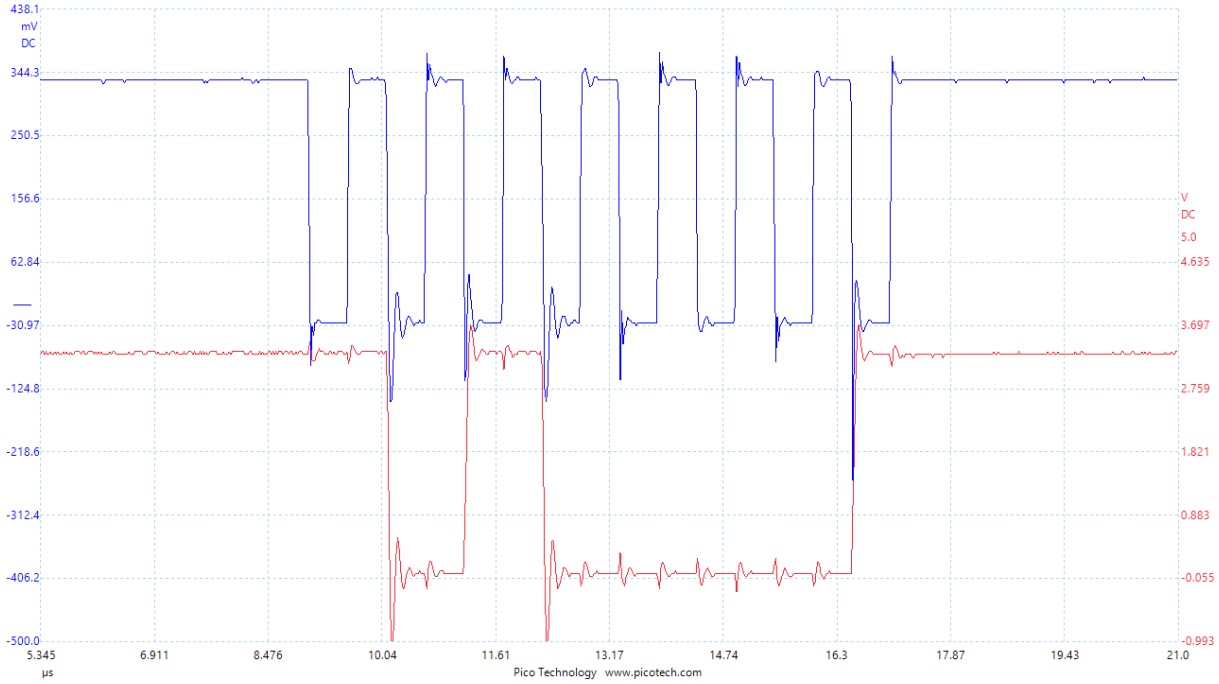

Figure 5: This shows clock mode 3

Appendices

A. Lab Exercise 1

A.1. Part 1

```
/* Sets up the mbed as SPI master, and continuously sends a single byte*/
#include "mbed.h"
SPI ser_port(PTD2, PTD3, PTD1); // mosi, miso, sclk
char switch_word ; //word we will send

int main()
{
    ser_port.format(8,0); // Setup the SPI for 8 bit data, Mode 0 operation
    ser_port.frequency(1000000); // Clock frequency is 1MHz
    while (1) {
        switch_word=0b10100001; // 0b means that the following number is in
    binary
        ser_port.write(switch_word); //send switch_word
        wait_us(50);
    }
}
```

A.2. Part 2

```
/* Sets up the mbed as SPI master, and continuously sends a single byte*/
#include "mbed.h"
SPI ser_port(PTD2, PTD3, PTD1); // mosi, miso, sclk
char switch_word ; //word we will send

int main()
```

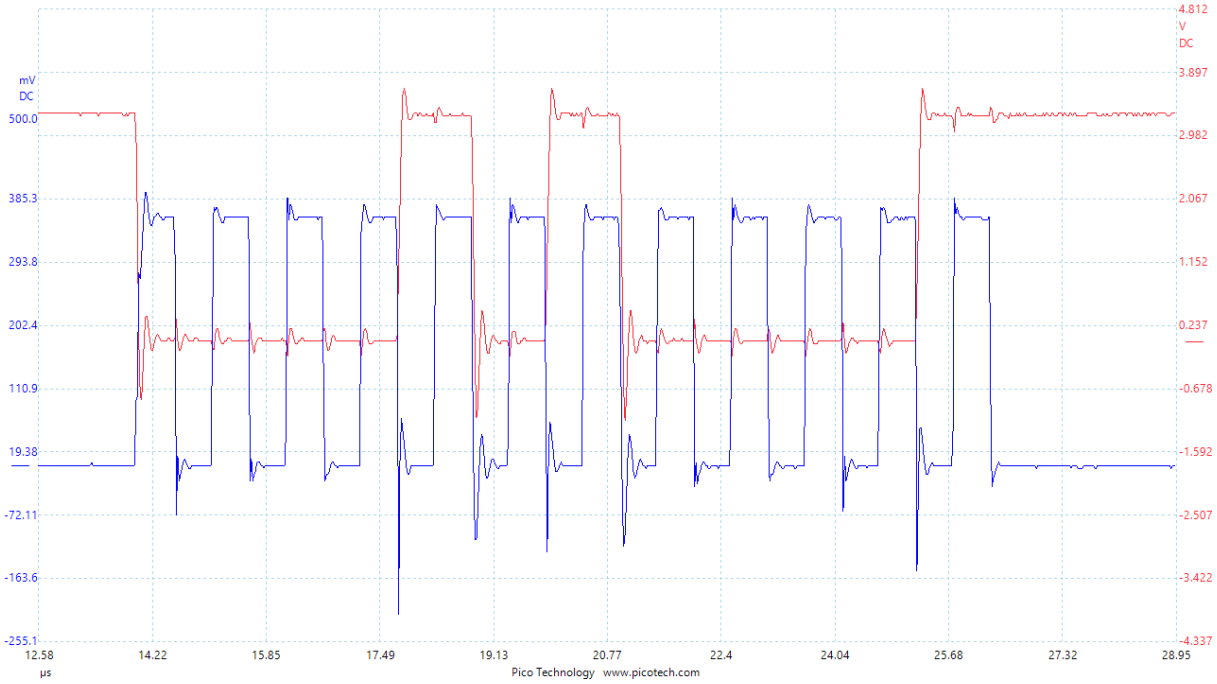

Figure 6: This shows 0x8A1 being transmitted as 12 bits with clock mode 0

```
{
    ser_port.format(12,0); // Setup the SPI for 8 bit data, Mode 0
    operation
    ser_port.frequency(1000000); // Clock frequency is 1MHz
    while (1) {
        switch_word=0b100010100001; // 0b means that the following number
    is in binary
        ser_port.write(switch_word); //send switch_word
        wait_us(50);
    }
}
```

A.3. Part 3

```
/* Sets up the mbed as SPI master, and continuously sends a single byte*/
#include "mbed.h"
SPI ser_port(PTD2, PTD3, PTD1); // mosi, miso, sclk
char switch_word ; //word we will send

int main()
{
    ser_port.format(16,0); // Setup the SPI for 8 bit data, Mode 0
    operation
    ser_port.frequency(1000000); // Clock frequency is 1MHz
    while (1) {
        switch_word=0b1000101010100001; // 0b means that the following
    number is in binary
        ser_port.write(switch_word); //send switch_word
        wait_us(50);
    }
}
```

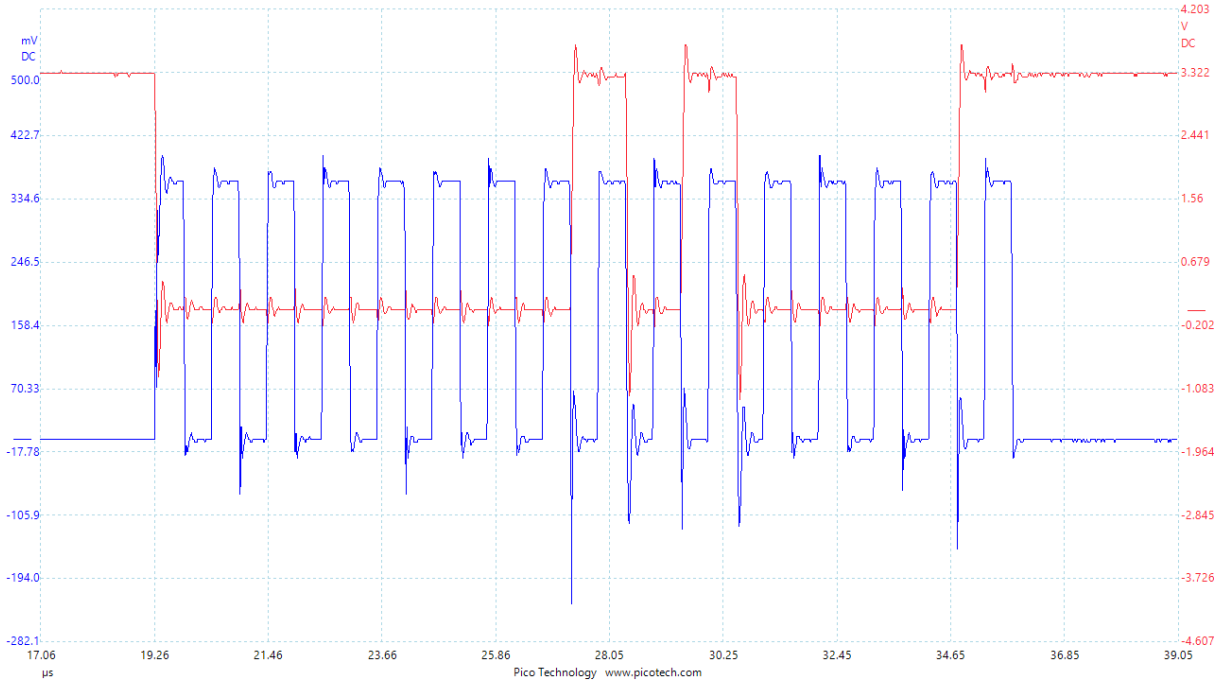

Figure 7: This shows 0x8AA1 being transmitted as 16 bits with clock mode 0

B. Lab Exercise 2

B.1. Part 1 - Master Program

```
/*Sets the mbed up as Master, and exchanges data with a slave, sending its
own switch positions, and displaying those of the slave.*/
#include "mbed.h"
SPI ser_port(PTD2, PTD3, PTD1); //mosi, miso, sclk
DigitalOut red_led(LED1);       //red led
DigitalOut green_led(LED2);     //green led
DigitalOut cs(PTD0);            //this acts as "slave select"
DigitalIn switch_ip1(SW2);
DigitalIn switch_ip2(SW3);
char switch_word; //word we will send
char recd_val;    //value return from slave

int main()
{
    ser_port.frequency(1000000); // 1MHz
    while (1)
    {
        //Default settings for SPI Master chosen, no need for further
    configuration
        //Set up the word to be sent, by testing switch inputs
        switch_word = 0xa0; //set up a recognizable output pattern
        if (switch_ip1 == 1) {switch_word = switch_word | 0x01;} //OR in
    lsb, check if the botton is pressed

        if (switch_ip2 == 1){ switch_word = switch_word | 0x02;}    //OR in
    next lsb, check if the botton is pressed
        cs = 0;                                 //select slave ("active low
    ")
        recd_val = ser_port.write(switch_word); //send switch_word and
```

```
    receive data. set a initial value for recd_val (dummy)
26          cs = 1;
27          wait(0.01);  //wait for slave to response
28                       //set leds according to incoming word from slave
29          red_led = 0; //preset both to 0
30          green_led = 0;
31          recd_val = recd_val & 0x03; //AND out unwanted bits
32          if (recd_val == 1)
33          {
34              red_led = 1;
35              green_led = 0;
36          }
37          if (recd_val == 2)
38          {
39              red_led = 0;
40              green_led = 1;
41          }
42          if (recd_val == 3)
43          {
44              red_led = 1;
45              green_led = 1;
46          }
47      }
48  }
```

B.2. Part 2 - Slave Program

```
1  /* Sets the mbed up as Slave, and exchanges data with a Master,
2  sending its own switch positions, and displaying those of the Master. as
       SPI
3  slave.*/
4  #include "mbed.h"
5  SPISlave ser_port(PTD2, PTD3, PTD1, PTD0);          // mosi, miso, sclk,
       ssel (fill by your own)
6  DigitalOut red_led(LED1);   //red led
7  DigitalOut green_led(LED2); //green led
8  DigitalIn switch_ip1(SW2);
9  DigitalIn switch_ip2(SW3);
10 char switch_word; //word we will send
11 char recd_val;    //value received from master
12
13 int main()
14 {
15     //default formatting applied
16     while (1)
17     {
18         //set up switch_word from switches that are pressed
19         switch_word = 0xa0; //set up a recognizable output pattern
20         if (switch_ip1 == 1)
21             switch_word = switch_word | 0x01;
22         if (switch_ip2 == 1)
23             switch_word = switch_word | 0x02;
24         if (ser_port.receive())
25         {                                  //test if data transfer has
       occurred
26             recd_val = ser_port.read();  // Read byte from master
27             ser_port.reply(switch_word); // Make this the next reply
28         }
```

```
        // set leds according to incoming word from slave
        red_led = 0; //preset both to 0
        green_led = 0;
        recd_val = recd_val & 0x03; //AND out unwanted bits
        if (recd_val == 1)
        {
            red_led = 1;
            green_led = 0;
        }
        if (recd_val == 2)
        {
            red_led = 0;
            green_led = 1;
        }
        if (recd_val == 3)
        {
            red_led = 1;
            green_led = 1;
        }
    }
}
```

C. Lab Exercise 3

C.1. Part 1

```
/* Emulating a USB mouse */
#include "mbed.h" // include mbed library
#include "USBMouse.h" // include USB Mouse library
USBMouse mouse; // define USBMouse interface
int dx[]={40,0,-40,0}; // relative x position coordinates
int dy[]={0,40,0,-40}; // relative y position coordinates

int main() {
    while (1) {
        for (int i=0; i<4; i++) { // scroll through position coordinates
            mouse.move(dx[i],dy[i]); // move mouse to coordinate
            wait(0.2);
        }
    }
}
```

C.2. Part 2

```
/* Emulating a USB mouse */
#include "mbed.h"           // include mbed library
#include "USBMouse.h"       // include USB Mouse library
USBMouse mouse;             // define USBMouse interface

int main()
{
    while (1)
    {
        mouse.scroll(1); // scroll page by parameter
        wait(0.2);
    }
}
```

C.3. Part 3

```
/* Emulating a USB mouse */
#include "mbed.h"            // include mbed library
#include "USBMouse.h"        // include USB Mouse library
USBMouse mouse;              // define USBMouse interface
InterruptIn leftButton(SW2);
InterruptIn rightButton(SW3);

void LeftDown();
void LeftUp();
void RightDown();
void RightUp();

int main()
{
    leftButton.fall(&LeftDown);
    leftButton.rise(&LeftUp);
    rightButton.fall(&RightDown);
    rightButton.rise(&RightUp);

    while(1){wait(1);}
}

void LeftDown(){
    mouse.press(MOUSE_LEFT);
}

void LeftUp(){
    mouse.release(MOUSE_LEFT);
}

void RightDown(){
    mouse.press(MOUSE_RIGHT);
}

void RightUp(){
    mouse.release(MOUSE_RIGHT);
}
```

C.4. Part 4

```
#include "mbed.h"
#include "USBMouse.h"
#include "FXOS8700Q.h"

//I2C lines for FXOS8700Q accelerometer/magnetometer
FXOS8700Q_acc acc( PTE25, PTE24, FXOS8700CQ_SLAVE_ADDR1);

USBMouse mouse;

int main()
{
    acc.enable();
    float faX, faY;
    int16_t x = 0;
    int16_t y = 0;

```

```
17      while (1)
18      {
19          acc.getX(&faX);
20          acc.getY(&faY);
21          x = 3*faX;
22          y = 3*faY;
23
24          mouse.move(x, y);
25          wait(0.001);
26      }
27  }
```

References

[1] J. Carver, *Fxos8700q*, https://os.mbed.com/users/JimCarver/code/FXOS8700Q/, May 2014.