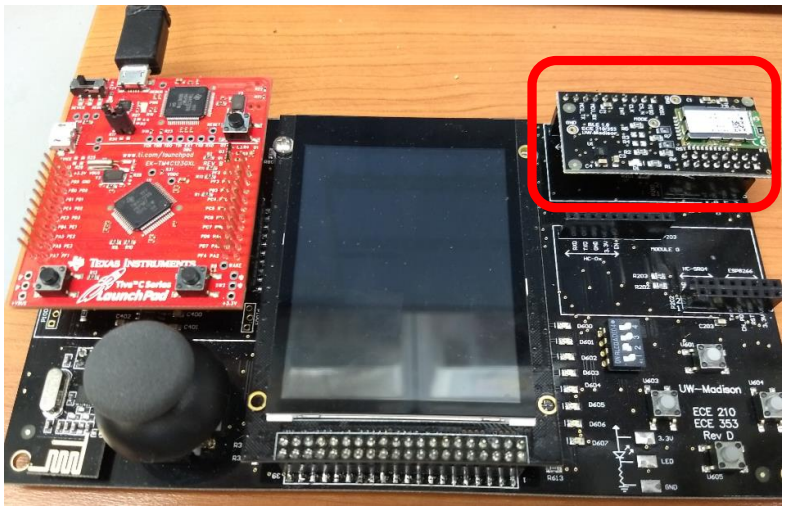
SPI ICE

Overview

This ICE will introduce you to the SPI peripheral on the TM4C123. You will learn how to configure the interface so that it can communicate with a LSM6DS3H accelerometer found on the ECE353 BLE mezzanine card.

Please connect your ECE353 BLE mezzanine card to Module Site 1 on your ECE353 Development platform. If you do not have an ECE353 BLE mezzanine card, please see your instructor.



SPI Basics

1. *Before you begin writing code for this exercise, complete the questions on page 1 of the SPI ICE questions.*

TM4C123 SPI Driver

Our first step in writing software to communicate with the accelerometer is to write a driver that can send/receive data to the TM4C123 SPI peripheral. The initialization of the GPIO pins and a function that sends/receives data has been completed for you.

You will need to complete the function that defines the behavior of the SPI interface Using Chapter 15 from the TM4C123 datasheet and/or the ECE353 text, complete page 2 of the SPI ICE questions.

2. ***Open spi.c from the drivers group**. Using the answers from page 2 of the SPI ICE, edit the function initialize_spi(). Complete the function based on the comments found in the function.*

   Look for the comments with ADD CODE to determine how to modify the function.

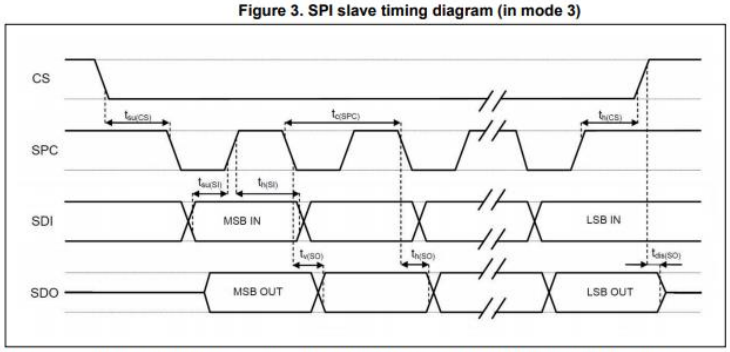
Accelerometer (LSM6DS3H) Driver

Once the TM4C123 spi driver has been completed, we will use the spiTx() function to send and receive data to/from the accelerometer.

3. *If we open the LSM6DS3H datasheet, we first need to determine which SPI mode the is supported. Device datasheets will contain a section that defines the supported interface characteristics. In this case, we can determine the supported SPI mode by looking at section 4.4.1, figure 3.*

tested in production

Figure 3. SPI slave timing diagram (in mode 3)



Note: Measurement points are done at 0.2·Vdd_IO and 0.8·Vdd_IO, for both input and output ports.

Supported SPI MODE: _____

Open peripherals/include/accel.h and modify line 53 accordingly.

4. *Normally, the SPI peripheral would control the Chip Select when sending data. Unfortunately for us, there is an errata in the TM4C123 that prevents it from working properly for some SPI modes. As a result, we are going to control the CS signal by manually forcing the CS signal high/low.*

   To so this, you will need to complete the following functions in `peripherals/c/accel.c`

   ```
   static __INLINE void  accel_CSN_low(void)

   static __INLINE void  accel_CSN_high(void)
   ```

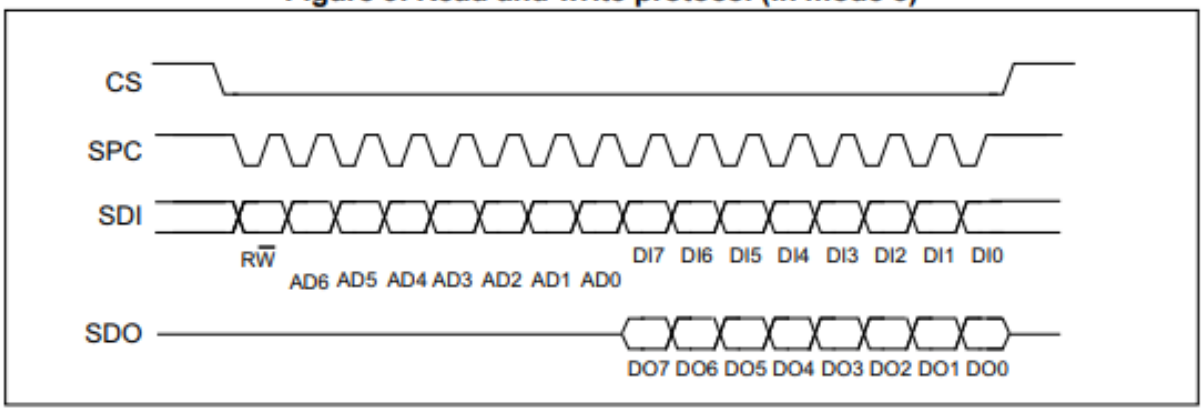
   You can write to the CS pin using the macros defined in accel.h. Below is an example of how to write to the register.

   ```
   ACCEL_CS_PORT->DATA = 0x00;
   ```

   When you implement these functions, be sure to ONLY modify the pin the CS signal is connected to.

5. *Section 6.2 LSM6DS3H datasheet describes the Read/Write Protocol required access a register. Use this diagram to answer the questions on page 3 of the SPI ICE.*

Figure 9. Read and write protocol (in mode 3)



6. *Complete Page 3 of the SPI ICE questions.*

7. *Using the answers from page 3 of the SPI ICE,* ***complete the two functions in accel.c listed below****. Make use of the function* ***`spiTx()`*** *to send the correctly formatted data packets to the accelerometer. These functions are used to read or write a register the register specified as the first parameter.*

   ```
   static __INLINE uint8_t accel_reg_read(uint8_t reg)

   static __INLINE void accel_reg_write(uint8_t reg, uint8_t data)
   ```

8. *Use the information found in Section 8 of the datasheet (Register Mapping) and the descriptions of those registers in the Register Description section to answer the Page 4 SPI questions.*

9. ***Complete three functions in accel.c*** *that return the acceleration values of the X, Y, and Z directions. Be sure to use the function you just completed,* ***`accel_reg_read()`****, to read the registers in each function. You can determine which registers we need to read by looking at Section 8 (Register mapping) of the LSM6DS3H datasheet. Search until you find "Accelerometer output registers" in the Comment column of the table.*

   ```
   int16_t accel_read_x(void)

   int16_t accel_read_y(void)

   int16_t accel_read_z(void)
   ```

10. *When you are finished, compile and load your ICE-20 code. After the code has been loaded, press the reset button on the Tiva Launchpad. As you move the ECE353 board, you will be able to observe debug messages in Putty (115200 8N1) that show you the movement of the board.*

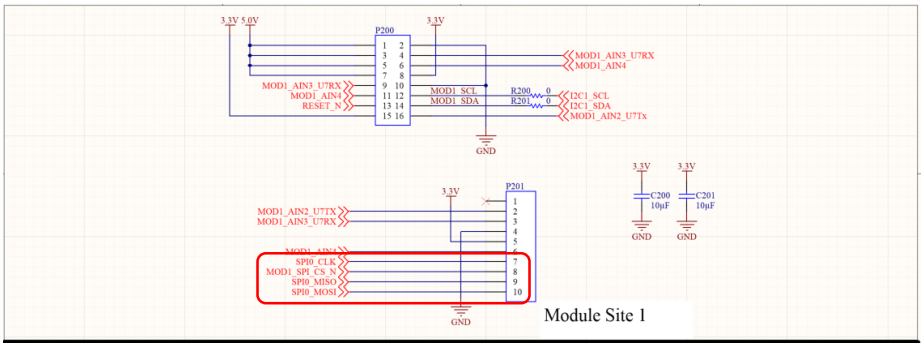
    The project will only display the X direction accelerations. Move the ECE353 board left/right and observe that as you move the board more quickly, the reported values will change more positive (as you tilt it left) and more negative (as you tilt it right).

    If your project does not demonstrate that you can read data from the accelerometer, re-examine your code and/or use the debugger to correct any errors.

11. *Turn in accel.c and spi.c to the SPI ICE dropbox.*

Appendix: SPI Schematic Analysis

1. *In order to determine which pins the accelerometer (LSM6DS3H) is connected to, we need to look at the connectors for Module Site 1*



If you look at page 10 of the ECE353 Schematics, it provides a summary of all the connections to the ECE353 Development platform. Find the signal names from above in this able and make note of which pins they are connected to.

Port A	
PIN	Net
PA0	UART0_RX
PA1	UART0_TX
PA2	SPI0_CLK
PA3	SPI0_CS_N
PA4	SPI0_MISO
PA5	SPI0_MOSI
PA6	I2C1_SCL
PA7	I2C1_SDA

Port B	
PIN	Net
PB0	LCD_DATA_0
PB1	LCD_DATA_1
PB2	LCD_DATA_2
PB3	LCD_DATA_3
PB4	LCD_DATA_4
PB5	LCD_DATA_5
PB6	LCD_DATA_6
PB7	LCD_DATA_7

Port C	
PIN	Net
PC0	No Connect
PC1	No Connect
PC2	No Connect
PC3	No Connect
PC4	LCD_CS
PC5	LCD_DATA_INST
PC6	LCD_WRITE_N
PC7	PC7

Port D	
PIN	Net
PD0	SPI_SELECT_0
PD1	SPI_SELECT_1
PD2	MOD0_AIN5
PD3	MOD1_ANN4
PD4	No Connect
PD5	No Connect
PD6	RF_CE
PD7	WS2812B

Port E	
PIN	Net
PE0	MOD1_AIN3_U7RX
PE1	MOD1_AIN2_U7TX
PE2	PS2_Y_AIN1
PE3	PS2_X_AIN0
PE4	MOD0_AIN9_U5RX
PE5	MOD0_AIN8_U5TX
PE6	No Connect
PE7	No Connect

Port F	
PIN	Net
PF0	SW2_IO_EXPANDER_INT
PF1	LED_RED
PF2	LED_BLUE
PF3	LED_GREEN
PF4	SW1_TP_INT_N

Using this information, the macros were defined in accel.h to aid in the development of a driver for the LSM6DS3H. Make use of these macros when completing the coding exercises to follow.