ECE 461/561 EMBEDDED SYSTEM DESIGN PROJECT 1: OPTIMIZING I^2C COMMUNICATION CODE (V1.0)

OVERVIEW

For this project you will modify a program on your Freedom board which flashes the LED when the accelerometer detects motion. You will improve the scheduling approach of the I^2C driver code. The starter code uses busy-waiting; you will convert it to a polled state-machine version, and then to an interrupt-driven version.

DETAILS

REQUIREMENTS

MODE SELECTION INPUT SIGNALS

After reset your code must read configuration information from an input port. This will select which I^2C code to run. The starter code configures these bits as digital inputs with pull-up resistors. When floating (disconnected), the input will be read as 1. When connected to ground (pin 12 on connector J2), the input will be read as 0.

Port E Bit 3	Port E Bit 4	Port E Bit 5	I^2C Code to Use
Floating	Floating	Grounded	Mode 1 - Blocking
Floating	Grounded	Floating	Mode 2 – FSM with Polling
Grounded	Floating	Floating	Mode 3 – ISR

DEBUG OUTPUT SIGNALS

Use three GPIO output bits to provide debug information visible on your logic analyzer. These will be used in grading your project.

Debug Bit	Port Bit	Indication
1	PTB1	I^2C code executing
2	PTB2	I^2C message is on bus
3	PTB3	Busy-wait loop executing for I2C code

- Debug bit 1 must be 1 only when executing I^2C communication code. Set debug bit 1 to 1 when entering the I2C communication code and clear it to 0 when exiting that code. This will let you see how long each state takes to execute.
- Debug bit 2 must be 1 only when an I^2C message is being sent or received on the bus. Set debug bit 2 immediately before sending a start condition and clear it to 0 immediately after sending a stop condition.
- Debug bit 3 must be 1 only when executing a busy-wait loop in I^2C communication code.

I²C COMMUNICATIONS CODE

MODE 1 – BLOCKING

Use the existing code I²C code. Add code to control the three debug bits to provide debug information visible on your logic analyzer.

MODE 2 – FINITE STATE MACHINE WITH POLLING

Convert the I²C read bytes function (i2c_read_bytes_fsm) to a finite state machine as discussed in class and presented in the notes online. You do **not** need to convert the I2C write bytes function. You will need to manually inline the i2c_wait function into i2c_read_bytes_fsm to create the FSM.

You may add a parameter to i2c_read_bytes_fsm to indicate if the function should copy the parameters and start a new read transaction. Alternatively you may use a shared flag variable which the FSM reads to determine if there is new data.

Draw a flowchart of the function and identify each state by circling its code. A hand drawing is fine. Include a photo or scan of this diagram in your report.

Configure the debug bits to provide timing information visible on your logic analyzer. There should be no busy-waiting loops in the FSM, so debug bit 3 should remain 0.

How often must your code call the FSM in order to operate correctly? Insert a call to ShortDelay between each call to the FSM and increase the argument until the program does not work correctly. When submitting the code, use the largest delay which works reliably. Include this time in your report.

MODE 3 – INTERRUPT SERVICE ROUTINE

Create an interrupt service routine to handle the I²C communication according to the flowchart of Figure 38-42 in the KL25 Sub-Family Reference Manual. Note that you only need to write the code for the master mode (the left half of the flowchart). The ISR must set a shared global variable to indicate when the communication has completed.

PERIPHERALS USED

THREE-AXIS ACCELEROMETER

Use the Freedom board's accelerometer to determine board orientation. For details, see the datasheet for the device (MMA8451), Chapter 8 of **Embedded Systems Fundamentals**, the introductory course module on Serial Communication (Presentation, I²C and Accelerometer Demonstration Code) and the FRDM-KL25Z User's Manual.

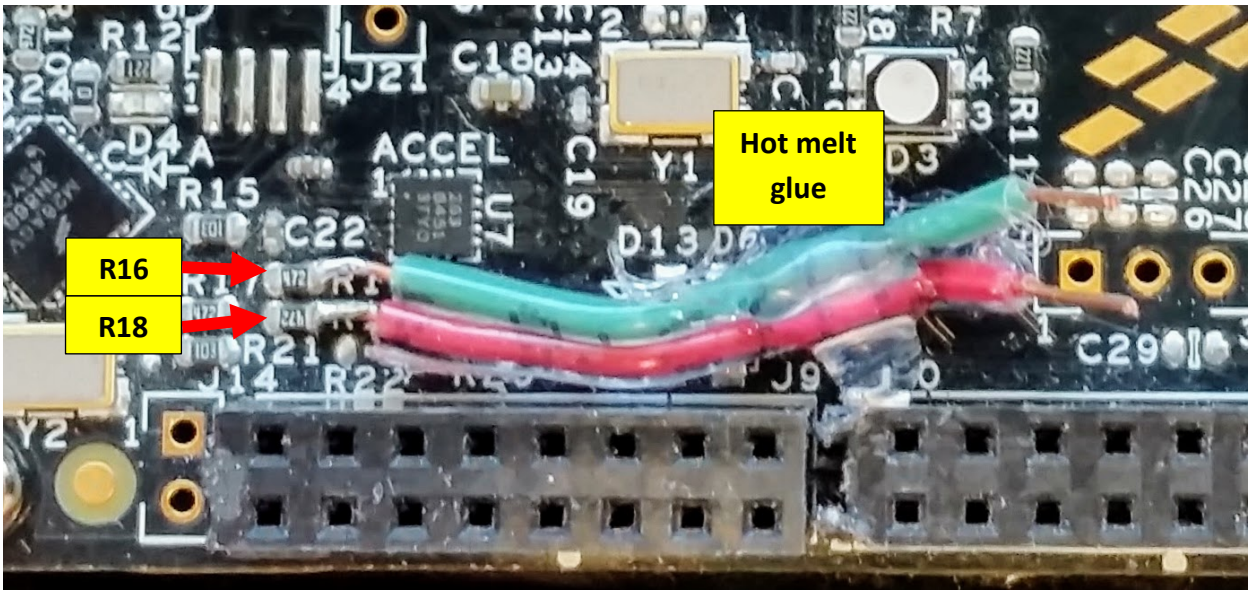
RGB LEDS

Use the RGB LEDs on the Freedom board to create the white light. Drive them with the GPIO modules.

LED	Port Connection
Red	Port B 18
Green	Port B 19
Blue	Port D 1

For details, see Chapter 2 of **Embedded Systems Fundamentals**, the introductory course module on digital interfacing with GPIO and the FRDM-KL25Z User's Manual.

OPTIONAL CIRCUIT MODIFICATIONS



Connect two wires to the I^2C bus signals to allow monitoring it with the logic analyzer. Solder solid wire (e.g. 20 gauge) to SCL via resistor R16 (upper resistor, right terminal) and SDA via R18 (lower resistor, right terminal). Be sure to provide strain relief by gluing the wire to the PCB as shown above.

DELIVERABLES

Submit these items through Moodle.

- Zipped archive of project files.
- PDF of project report (see template for ECE 461 or 561 for required contents).