ECEN 3360
Digital Design Lab #3a update-1
I2C and Load Power Management
Spring 2019

Objective: Develop an I2C ladder flow chart and I2C driver to access the Si7021. Lab 3a is to learn how to write an I2C driver and verify functionality by writing and reading the Si7021

Note: This assignment will begin with the completed Lab 2, Energy Modes, Assignment

Due Date 3a: February 9th, 2018 at 11:59pm

Instructions:

1. Make any changes required to the Managing Energy Mode Assignment
    a. Replace "magic numbers" with #define statements
    b. Create a .c and .h for each peripheral for ease of readability and code reuse, best practices. For example:
        i. cmu.h/cmu.c
        ii. gpio.h/gpio.c
        iii. letimer.h/letimer.c
        iv. i2c.h/i2c.c
        v. sleep.h/sleep.c
        vi. si7021.h/si7021.c

2. Add the following lines of code where shown to help provide a more consistent current measurement from one board or implementation to the next.

```
EMU_DCDCInit_TypeDef dcdcInit = EMU_DCDCINIT_DEFAULT;
CMU_HFXOInit_TypeDef hfxoInit = CMU_HFXOINIT_DEFAULT;

/* Chip errata */
CHIP_Init();

/* Init DCDC regulator and HFXO with kit specific parameters */
/* Initialize DCDC. Always start in low-noise mode. */
EMU_EM23Init_TypeDef em23Init = EMU_EM23INIT_DEFAULT;
EMU_DCDCInit(&dcdcInit);
em23Init.vScaleEM23Voltage = emuVScaleEM23_LowPower;
EMU_EM23Init(&em23Init);
CMU_HFXOInit(&hfxoInit);
```

3. Set LETIMER0 should be set to the following conditions at startup / reset
    a. Period = 4.0 seconds
    b. LED on-time = 500ms / 0.5s
        i. Use LED0 for LETIMER0 LED blinking
    c. Lowest Energy mode of operation = EM1

4. Develop software ladder flow chart for I2C driver to write to the Si7021, User Register 1

5. Develop software ladder flow chart for I2C driver to read Si7021, User Register 1

6. Initialize I2C peripheral to perform the following:
    a. Pearl Gecko to be I2C master
    b. The I2C is a peripheral within the chip whose peripheral's SCL and SDA lines must be routed to the external pins
        i. You can find the external pins to route to by using the dev kit quick start user guide and search for the block diagram of the EFM32 I2C connection to the Si7021 or trace the wires back from the Si7021 to the Pearl Gecko on the dev kit schematic.
        ii. With having the external port and pin names, you must now determine the location number to indicate to the Pearl Gecko how to route SCL and SDK from the internal I2C0 peripheral to these GPIO pins. You can find this information in the Pearl Gecko datasheet. Search for the desired port numbers and you will find a relocation table. For the port and pin name desired, find the desired functionality you are requiring in the communication column. The number to the right of your desired signal is the routing #. You must find these for both SCL and SDA.
        iii. With the routing information, you must now specify the peripheral the routing number for both SCL and SDA. You must write directly to the register and position the routing number in the proper position of the relocation register. You can find the information by using the Pearl Gecko Reference Manual. At the end of the I2C chapter, you will find the register map and then go to register ROUTELOC0. You will need to be writing to this register directly
            1. I2C0->ROUTELOC0 = ……
        iv. With the internal peripheral now routed to the external GPIO, you must enable the internal peripheral SCL and SDA lines on the internal bus. You can determine the bits to ASSERT by using the Pearl Gecko Reference Manual and reviewing the ROUTEPEN register information at the end of the I2C chapter. To specify the appropriate bit you use the following nomenclature
            1. I2C_"register name"_"desired bit"
            2. Remember this nomenclature is identical to the LETIMERO where you enabled an interrupt bit using LETIMER_IEN_COMP0.

v. You have routed the internal I2C signals to the external GPIO pins. You now must set the port of the GPIO pins to the proper configuration which for the I2C signals is "Wired AND."
vi. Remember that you must also set the port information for SENSOR ENABLE bit and then ASSERT it to enable the Si7021.

c. Match the I2C data rate to the fastest Si7021 mode
d. Set the I2C CLTO, Clock Low Time Out, in the CNTL register to 1024PCC, prescaled clock cycles
e. Set the I2C interrupt for CLTO, Clock Low Time Out, for debug
f. Set the I2C BITO, Bus Idle Time Out, in the CNTL register to 1024PCC, prescaled clock cycles
g. Set the I2C interrupt for BITO, Bus Idle Time Out, for debug
h. Not, the CLTO and BITO interrupt enable bits should only be enabled after SCL and SDA pins have been initialized on
i. Implement I2C driver using wait states
    i. Not energy efficient, but focusing on functionality
j. EXTRA CREDIT: Implement an I2C interrupt based I2C read and write driver using RXDATAV, ACK, and NACK interrupts
k. You must determine what are the GPIO pins connected to the Si7021 via the dev kit schematic or dev kit user guide and configure in the gpio.h/gpio.c program files
    i. You must properly initialize the GPIO SCL and SDA pins to the proper I/O configuration
    ii. The driver must route the SCL and SDA pins to the proper GPIO pins and the peripheral SCL and SDA signals must be enabled

7. Implement I2C interrupt handler
    a. Use CLTO, BITO for debug in the development of the I2C driver
        i. If CLTO is set, turn-on / latch-on LED1 and LED0 off if in the process of an I2C read operation
        ii. If CLT0 is set, turn-on / latch-on LED1 and turn-on / latch-on LED0 if in the process of an I2C write operation
        iii. Enter while (1) loop to effectively freeze firmware in the interrupt handler and indicating an error
    b. Extra Credit: Implement I2C interrupt handler to enable read and write drivers of I2C registers

8. Write to the Si7021 User Register 1 to <u>set the temperature resolution to 12-bits</u>
    a. The Si7021 is designed to be implemented in a low energy design. You must assert SENSOR_ENABLE before reading or writing to the Si7021. For this assignment, you can turn it on and keep it on, but for lab 3b, you will deassert SENSOR_ENABLE when the Si7021 is not being used

b. You must determine what is the GPIO pin used for SENSOR_ENABLE via the dev kit schematic or the dev kit user guide and configure it in the gpio.h/gpio.c program files
c. You can assert SENSOR_ENABLE using the same emlib routine to turn-on an LED
d. Write to the Si7021 User Register 1 on COMP1 interrupt

9. Read the Si7021 User Register 1 to verify that the Si7021 has been set to a temperature resolution of 12-bits
    a. The Si7021 is designed to be implemented in a low energy design. You must assert SENSOR_ENABLE before reading or writing to the Si7021. For this assignment, you can turn it on and keep it on, but for lab 3b, you will deassert SENSOR_ENABLE when the Si7021 is not being used
    b. You must determine what is the GPIO pin used for SENSOR_ENABLE via the dev kit schematic or the dev kit user guide and configure it in the gpio.h/gpio.c program files
    c. You can assert SENSOR_ENABLE using the same emlib routine to turn-on an LED
    d. Read the Si7021 User Register 1 immediately after writing to the Si7021 User Register 1
        i. If the value read does not match expected result, turn-on LED1 and enter while(1) loop indicating an error for debug

10. Development / Debug suggestions:
    a. Set LETIMER0 period to 1 second to have a faster “heart beat”
    b. Set the write and read operation of the Si7021 User Register 1 every LETIMER0 COMP1 interrupt to enable ease of debug by allowing multiple accesses to the Si7021
    c. You will not be able to step through the I2C handler, it will result in CLTO and BITO interrupts. The best way to debug an I2C driver is to use a breakpoint to determine how far you have gone into the driver, and if the break point occurs, create a breakpoint further down the driver, clear the current breakpoint, and reset you firmware.
    d. If the temperature resolution from the Si7021 User Register 1 <u>does not</u> equal 12-bits, turn-on LED1 and turn off LED0 and enter while(1) loop
    e. Before writing to the Si7021 User Register 1, turn-on LED1 and turn-off LED0, then you know that an error occurred during the I2C write operation if this configuration of LEDs remains constant
    f. Before writing to the Si7021 User Register 1, turn-off LED1 and turn-on LED0, then you know that an error occurred during the I2C read operation if this configuration of LEDs remains constant

11. Set your LETIMER0 to the requirements of the assignment described above in item (2)

12. Set your write and confirmation of the Si7021 User Register 1 to occur only once while configuring your I2C device.

Questions:

1. There are no questions for Lab 3a

Deliverables:

1. One document that includes the Ladder flow chart for read and one for write to the Si7021 User Register 1

2. Export you project as an archive, .zip file, and upload into canvas

Rubric: (Base grade is 20 pts)

1.	I2C read and write flow charts	6 pts
2.	I2C software project	14 pts
	a. Total	20 pts
3.	Extra credit	
	a. I2C interrupt based read driver	3 pts
	b. I2C interrupt based write driver	3 pts
	a. Total	26 pts max
4.	Deductions:	
	a. Magic numbers	- 3 pts
	b. Not unique files per peripheral	- 3 pts
	c. No acknowledgement of IP	- 2 pts