CMPE 443 PRINCIPLES OF EMBEDDED SYSTEMS DESIGN

LAB #009

"I2C"

Motivation

In this experiment, we will use the I2C interface to access the data of the temperature on the LPC4088 Experiment Base Board. Temperature sensor is one of the common components that are found in mobile embedded devices. I2C is frequently used by a lot of sensor vendors because it provides a serial digital interface between the sensor and the microcontroller. Hence, in this experiment you will learn:

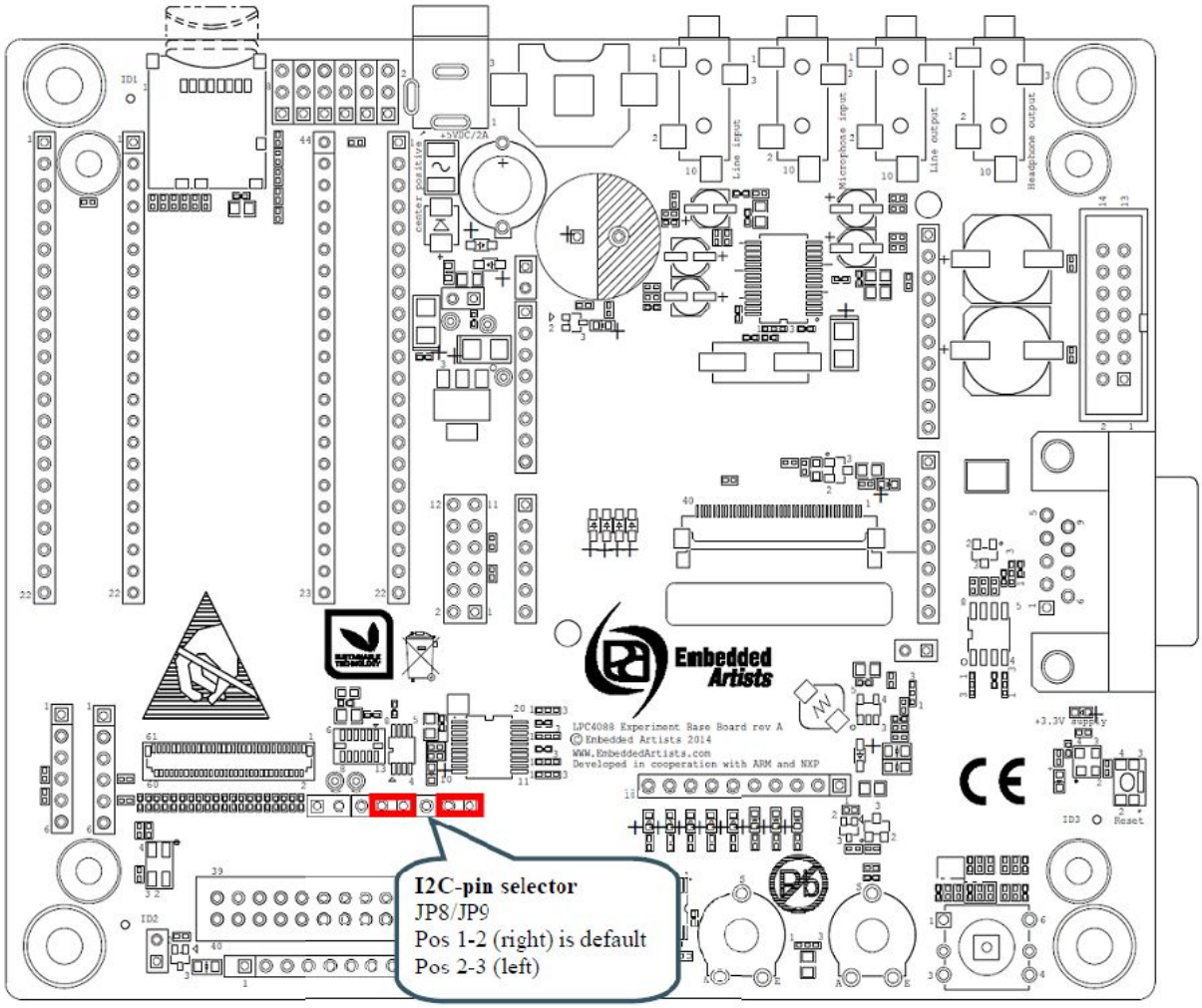
- configuring I2C for one of the I2C communication standards.
- configuring the temperature sensor through I2C
- working with I2C so as to access the data from the temperature sensor

1) LM75B *4 pts*

- Temperature accuracy of this sensor? **(?)** _____ *0,5 pts*
- Temperatures range of this sensor? **(?)** _____ *0,5 pts*
- According to Temperature register, which byte and bit is used for checking the temperature is positive or negative? **(?)** _____ *1 pt*
- What is the formula for the correct temperature value (**Celsius**) from the sensor data, if the temperature is positive? **(?)** _____ *1 pt*
- What is the formula for the correct temperature value (**Celsius**) from the sensor data, if the temperature is negative? **(?)** _____ *1 pt*

2) Initialize I2C *17 pts*

In order to use LM75B temperature sensor, I2C should be initialized. In Experimental base board, I2C channel 0 will be used and JP8/JP9 should be in position 1-2 on your board.



For JP8/JP9 in position 1-2:

- Which pins are used for I2C bus? **(?)** _____ *0,5 pts*
- Which pin is used for I2C Data Input/Output? **(?)** _____ *0,5 pts*
- Which pin is used for I2C Clock Input/Output? **(?)** _____ *0,5 pts*
- Which I2C is used for these pins? **(?)** _____ *0,5 pts*

In **Library/I2C.h** file,

- Write the base address of the I2C. **(*?)** _____ *1 pt*
- Write the IOCON address of I2C Data Input/Output. **(*?)** _____ *0,5 pts*
- Write the IOCON address of I2C Clock Input/Output. **(*?)** _____ *0,5 pts*

I2C can work with 3 different rates which are Standard, Fast Mode and Fast Mode Plus. We want to use I2C with Standard mode. (PCLK = 60 MHz)

- What is the I2CSCLL + I2CSCLH value? **(?)** _____ *0,5 pts*
- In **Library/I2C.h** file, write I2CDutyCycle which is (I2CSCLL + I2CSCLH) / 2 **(*?)** _____ *0,5 pts*

In **Library/I2C.c** file and **I2C_Init** method:

- Turn on I2C **(*?)** _____ *0,5 pts*
- Clear Assert acknowledge, I2C interrupt, START flag and I2C interface **(*?)** _____ *0,5 pts*
- Write correct value to CONSET register for Master only functions. (In Initialization routine of I2C). **(*?)** _____ *0,5 pts*
- Change the functionality of the I2C_SDA_PIN. **(*)** _____ *0,5 pts*
- Change the functionality of the I2C_SCL_PIN. **(*)** _____ *0,5 pts*

In **Library/I2C.c** file and **I2C_Start** method:

- Write Correct Value to CONSET register (In Start Master Transmit function). **(*?)** _____ *0,5 pts*

In **Library/I2C.c** file and **I2C_Stop** method:

- Write the correct value to CONSET for transmitting a STOP condition in master mode **(*?)** _____ *0,5 pts*

In **Library/I2C.c** file and **I2C_Wait_SI** method:

- Fill the condition in the while statement for waiting until I2C state changes. **(*?)** _____ *0,5 pts*

In **Library/I2C.c** file and **I2C_WriteData** method (Master Transmitter states) (You can use *I2C_Stop()* method for stopping I2C):

- Find the correct states for a START or a repeated START Condition **(?)** _____ *0,5 pts*
- Find the correct state for First Data Byte is Transmitted **(?)** _____ *0,5 pts*
- Find the correct states for Data is Transmitted **(?)** _____ *0,5 pts*

- Stop and Return -1 when A START or a repeated START condition has not been transmitted. (You check this from *status* variable) **(*)** _____ *0,5 pts*
- Write the correct address for writing data to I2C device. (The address of the device is different while reading and writing.) **(*)** _____ *0,5 pts*
- Stop and Return -1 When First Data Byte is not Transmitted (You check this from *status* variable) **(*)** _____ *0,5 pts*
- Stop and Return the index variable when Data is not Transmitted (You check this from *status* variable) **(*)** _____ *0,5 pts*

In **Library/I2C.c** file and **I2C_Do_Write** method:

- Which I2C register contains the data which will be transmitted during master transmit mode, and the data which will received during master receive mode **(?)** _____ *0,5 pts*
- Write the value the a register for transmitting the data. **(*?)** _____ *0,5 pts*

In **Library/I2C.c** file and **I2C_ReadData** method (Master Receiver states):

- Find the correct state for ACK has been received. Data will be received and ACK returned. **(?)** _____ *0,5 pts*
- Find the correct state for Data has been received, ACK has been returned. **(?)** _____ *0,5 pts*
- Find the correct state for Data has been received, NOT ACK has been returned. **(?)** _____ *0,5 pts*
- Write the correct address for reading data from I2C device. (The address of the device is different while reading and writing.) **(*)** _____ *0,5 pts*
- Stop and Return -1 when ACK is not received **(*)** _____ *0,5 pts*
- Stop and Return index when Data is not received **(*)** _____ *0,5 pts*
- Stop and Return (length - 1) When the Last Data is not received **(*)** _____ *0,5 pts*

3) Initialize LM75B *9 pts*

LM75B I2C address should be known for the communication. In **Library/LM75B.h** file:

- Write the I2C address of LM75B (7 bits) **(?*)** _____ *2 pts*

In **Library/LM75B.c** file:

- Combine the buffers into value variable. **(?*)** _____ *2 pts*
- Return correct temperature value for positive and negative temperature values. **(?*)** _____ *5 pts*

4) Receiving Data from LM75B *10 pts*

In this section, you will get the temperature sensor value from the LM75B Temperature sensor (LM75B_Read), and send it to the Putty via serial communication. The code for the serial communication is already in your project and it is configured as the LAB 8. You will write the temperature sensor value the Putty with this format (in every line):

"Temperature sensor value is 00"