Exercise sheet 2 - I2C

Background

In this exercise sheet, you are going to implement communications with external devices using the I2C bus technology. I2C is a commonly used standard to interface sensors and peripherals by just two communication lines, having multiple slave devices connected to the same lines. The extension board of the advanced microcontroller laboratory features several I2C devices, such as an external EEPROM, a digital potentiometer and an ADC-DAC-converter. In this exercise, you will have to implement an application using the hardware I2C interface of the MSP430.

Getting started

Please follow these steps before starting with the exercise tasks:

1) First of all, please familiarize yourself with the I2C protocol before writing any code for this exercise. A good documentation can be found here: `http://i2c.info/i2c-bus-specification`.

2) As the I2C module uses the same peripheral module as the UART interface, disable UART in the initialization of `templateEMP.h` To do so, you can either modify the header file or you can type `#define NO_TEMPLATE_UART` before including it.

3) To be on the safe side for future exercises, we recommend to increase the clock speed of the MCU to full-speed, i.e. 16 MHz. Therefore, modify `templateEMP.c` appropriately.
For future use, keep in mind that this might have impact on delay cycle functions being used in already existing libraries.

4) Perform the following wiring:

   - Connect the LCD as stated in exercise sheet 1.
   - On the extension board: connect `CON6:I2C_SPI` to `CON2:P1.3`, `CON6:XSCL` to `CON2:P1.6` and `CON6:XSDA` to `CON2:P1.7`.
   - Connect `CON6:UDAC` to `CON5:BCKL`. Set `JP2:BKL_ON` to the right position.
   - On the base board: Connect `CON3:P3.6` to `X1:Buzzer` (just for bonus task).

5) In order to connect the I2C pins of the MSP430 to the I2C bus on the extension board, you have to make sure that the `IC4:74HCT4066` routes the correct pins to the I2C bus. You can switch between the different input lines by using `CON6:I2C_SPI` (you just wired it to `CON2:P1.3`). Familiarize yourself with the functionality of the `IC4:74HCT4066` by reading the appropriate sections of the datasheet.

6) In Code Composer Studio, import the project archive `Exercise_2.zip`, which you can find on the ILIAS web platform.

Task 1

Implement a I2C library based upon the given header file `i2c.h`. You may ignore the extensions of the I2C specification such as 10 bit addresses, Fast Mode and High-Speed Mode.
To get started, consider the *MSP430x2xx Family User's Guide* (see the datasheet section for board I on ILIAS) or the given code examples from Texas Instruments on the I2C module[1]. Read and write the individual data bytes with the interrupt service routines given in `i2c.c`; write and read functions should be blocking, i.e. they just finish once all bytes are transmitted or received. Using the IKALOGIC logic analyzer is highly recommended to observe the SDA and SCL lines. Check the signal's time sequence for debugging purposes. **(6 pts. in total - see header for distribution)**

Task 2

Implement the ADC/DAC library `adac.h` which controls `IC2:PCF8591T`. Use the i2c.h-library which you created previously. For details, consider the given datasheet.
(2 pts. in total - see header for distribution)

Task 3

Write a main program which reacts to changes to the joystick `JOY1`. The following functions should be realized:

a) Set the brightness of the LCD based on the x-position of the joystick. To read the joystick, use the ADC; to control the brightness, use the DAC. **(1 pt.)**

b) *Bonus*: Play a sound on the buzzer. The sound's frequency should directly correlate with the y-position of the joystick. Pause that sound whenever the joystick is pressed. You may use PWM for playing the sound. **(2 bonus pts.)**

Task 4

Prepare your exercise sheet for submission using the following subtasks **(1 pt.)**:

a) Fill the file `feedback.txt` with a brief feedback statement, which contains specific problems and issues you experienced while solving the exercise, additional requests, positive remarks and alike.

b) Please assure that every file contains a header comment including...
   - your personal information (i.e. name, matriculation number, e-mail contact).
   - the exercise sheet number.
   - a brief description of the given code and its purpose; the description of the file *main.c* should also list the required pin connections.

c) Rename your project to *Exercise_[ExerciseNo]_[YourLastName]* within Code Composer Studio.

d) Export your project to an archive file (ZIP) using Code Composer Studio (File > Export... > General > Archive File).

e) Upload your file to ILIAS considering the individual deadline.

[1] http://www.ti.com/product/MSP430G2553/toolssoftware