Exercise Sheet 3 - Advanced Sensors

Background & Getting Started

After implementing the I2C driver, we're finally going to use it for reading some more advanced sensors: Accelerometers. To perform the following tasks, keep the wiring of exercise sheet 2. Just add a connection from `CON6:MMA_INT1` to any pin of the MSP430 with interrupt capability (please clearly name that pin in a comment in your `main.c`).

Task 1

Implement a library to control the accelerometer chip MMA8451Q using your I2C library. Consider the header file `mma.h` for detailed information on the functions. They can be categorized as follows:

a) Initialization: Here, you will basically set up the chip (`mma_init`, **1 pt.**), perform a self-test (`mma_selftest`, **1 pt.**) set the measurement range (2G, 4G or 8G maximum acceleration, `mma_setRange`, **0.5 pts.**) and the measurement resolution (8 or 14 bit, `mma_setResolution`, **0.5 pts.**)

b) Reading values: Develop a function to capture the value of all three axes and to store them internally (`mma_read`, **1 pt.**) and multiple functions to obtain the values from x-, y- or z-axis with particular resolution (`mma_get8[XYZ]`, `mma_get14[XYZ]` and `mma_getReal[XYZ]`, **1 pt. altogether**).

c) Reacting on tapping events: Write functions to configure the MMA's internal feature of reacting to a double tap (`mma_enableTapInterrupt`, **1 pt.** & `mma_disableTapInterrupt`, **0.5 pts.**)

Task 2

Write a main program which – by using the libraries of the exercise sheets 1, 2 and 3 – reacts to movements of the extension board. Your program should:

a) Initialize the `MMA` and execute the self-tests. Inform the user on the display that the self-tests are about to run. Acknowledge successful self-tests; show an error-message and repeat the self-test upon a failure. **(0.5 pts.)**

b) Next, we want to send the `MSP430` to sleep mode (also called low-power mode, LPM; always use the deepest sleep level you can use). The `MMA` should listen for an incoming double tap and wake the `MSP430` in case such an event is occuring.
As a result, do the following: Set up the double tap detection on the `MMA` using the function you created in task 1. Configure an interrupt service routine on the `MSP430` that will be executed when the double tap was detected by the `MMA`; in this routine, the MSP430 can exit from low-power mode to do something (see subtask d)). After the configuration, put the `MSP430` to sleep mode. Show an appropriate message on the LCD while the `MSP430` is sleeping. **(1 pt.)**

c) *Bonus:* Save some more power: Modify your I2C library is a way that the controller enters sleep mode while waiting for the transmission or reception of the next data byte (i.e. the next interrupt to occur). Having processed all the data, leave the low-power mode. **(1 bonus pt.)**

d) When double tapping the `MMA`, wake up the `MSP430`. From now on, query any of the three axes and display a bubble level on the LCD, corresponding to the sensor's readout. **(0.5 pts.)**

e) *Bonus*: Display a realistic two-axis bubble by using custom characters for your LCD display. The bubble should float around as if it was an analog air bubble (i.e. the bubble shouldn't jump from one character's position to the next, but rather float smoothly between the lines. See figure 1 for an example of the bubble in several y-positions). **(1 bonus pt.)**

f) If the accelerometer's data is not changing significantly within a time frame of five seconds, prepare for another double-tap interrupt and go back to sleep (don't forget the message on the LCD). Dim the display whenever the programm goes to sleep and turn it back on when the chip wakes up. **(0.5 pts.)**

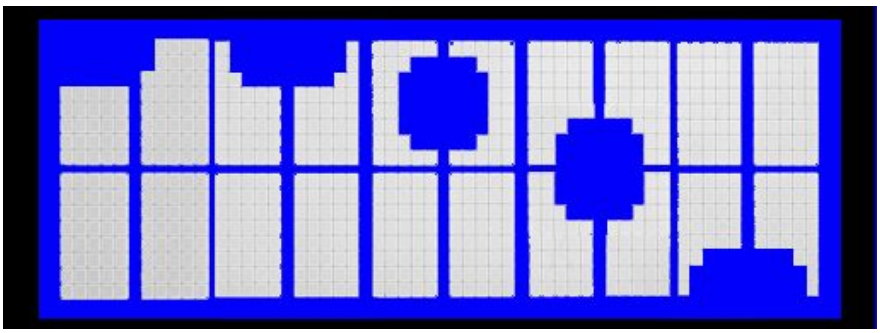

Abbildung 1: Custom character example of a bubble in several y-positions. Once realizing the bubble, just use a single bubble floating both in x- and y-direction.

Task 3

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.