Embedded and Ambient Systems

Topic of exercise 2

The task to be accomplished during the exercise is that an LED lights up when a button is pressed, and goes out when the button is released. We solve the task starting from zero register-level settings.

Peripheral programming at register level

- Based on the circuit diagram, find out which input of which port the LEDs and buttons are connected to (EFM32GG-BRD2200A-A03-schematic.pdf).
- Create an empty project.
- Collect the necessary registers. The registers can be found in the processor documentation (EFM32GG-RM.pdf). The following information should be used and tasks to be done:
    - Registers are memory embedded registers. You can calculate the address of the registers in the following way. Each function group has a base address, which is described in page 17, Fig. 5.2. Here you will need the GPIO and CMU (Clock Management Unit) base addresses. The offset of the individual function registers compared to the base address is given in the documentation as follows:
        - CMU registers: 11.4. chapter, page 136
        - GPIO registers: 32.4. chapter, page 764

      In the above chapters, you can find the functions of the individual registers, their offsets compared to the base address, and which parameters can be set in which bit fields.
      Note: if an address is known (e.g. 0x400c8000+0x044), then according to the C language it can be referred to as a 32-bit variable in the following way:
      (*(volatile long unsigned int *) (0x400c8000+0x044))
      You may want to name your register references as, for example:
      #define REG_A (*(volatile long unsigned int *) (0x400c8044))
      which can be referred to, for example, like this (setting the 13th bit to 1 in the REG_A register):
      REG_A |= 1<<13;
    - The clock signal of the GPIO peripheral must be enabled (clock signal distribution network: page 128):
        - GPIO clock: CMU_HFPERCLKEN0 register GPIO bit.
    - The pins corresponding to the push buttons must be set as inputs (INPUT).
        - The input values can be read from the xx_DIN register belonging to the given port.
    - The legs for the LEDs must be set as outputs (PUSHPULL).
        - The output value can be written directly by the xx_DOUTS register belonging to the given port, it can be set to a high level with the xx_DOUTSET register, and it can be set to a low level in the xx_DOUTCLR register (by writing 1 to the given location).
- Write the program:
    - Enable the GPIO clock,
    - Initialize the LEDs,
    - Set the state of at least one LED to high.
    - Additional task:
        - Initialize the push buttons,
        - In an endless cycle, query the value of the push button(s) and set the status of the LED(s) according to the buttons.

```
//#include "em_device.h"
//#include "em_chip.h"


#define CMU_BASE_ADDR 0x400c8000
#define GPIO_BASE_ADDR 0x40006000

#define CMU_HFPERCLKDIV (*(volatile unsigned long int*)(0x400c8000 + 0x008))
#define CMU_HFPERCLKEN0 (*(volatile unsigned long int*)(0x400c8000 + 0x044))

#define GPIO_PB_MODEH (*(volatile unsigned long int*)(0x40006000 + 0x02C))
#define GPIO_PB_DIN (*(volatile unsigned long int*)(0x40006000 + 0x040))

#define GPIO_PE_MODEL (*(volatile unsigned long int*)(0x40006000 + 0x094))
#define GPIO_PE_DOUT (*(volatile unsigned long int*)(0x40006000 + 0x09C))



int main(void)
{
/* Chip errata */
//CHIP_Init();
CMU_HFPERCLKDIV |= 1 << 8; // peripheral clk enable
CMU_HFPERCLKEN0 |= 1 << 13; // GPIO signal enable

//
GPIO_PE_MODEL |= 4 << 8; // port E pin 2: pushpull output: page 766
GPIO_PE_MODEL |= 4 << 12;// port E pin 3: pushpull output

GPIO_PE_DOUT |= 1 << 2; // port E pin 2: high
GPIO_PE_DOUT |= 1 << 3; // port E pin 3: high

GPIO_PB_MODEH |= 1 << 4; // port B pin 9: input: page 767
GPIO_PB_MODEH |= 1 << 8;// port B pin 10: input


/* Infinite loop */
while (1) {

        if (GPIO_PB_DIN & (1<<9)){
                GPIO_PE_DOUT &= ~(1 << 3);
        } else {
                GPIO_PE_DOUT |= 1 << 3;
       }

        if(GPIO_PB_DIN & (1<<10)){
                GPIO_PE_DOUT &= ~(1 << 2);
        } else {
                GPIO_PE_DOUT |= 1 << 2;
       }

}
}
```