Lab Assignment: GPIO

Objective

To grow your skills in the following:

1. Manipulating a microcontroller's registers in order to access and control physical pins
2. Use implemented driver to sense input signals and control LEDs.

Assignment

Test your knowledge by doing the following:

Part 0. Basic GPIO Driver to blink an onboard LED

```
 void vControlLED( void * pvParameters )
{
     //Choose one of the onboard LEDS by looking into schematics and write code for the below
    1) Set the PINSEL register bits for that LED
    2) Set the FIODIR register bit for that LED
    while(1)
    {
        3) Set the bit in FIOSET register to turn the LED ON
        vTaskDelay(500);

        4) Set the bit in FIOCLR register to turn the LED OFF
        vTaskDelay(500);
    }
}
int main()
{
```

```
    xTaskCreate(vControlLED, “internal_led_task”, 1024, NULL, PRIORITY_LOW, NULL);
    /* Start Scheduler */
    vTaskStartScheduler();
    return 0;
}
```

Part 1. Implement the **GPIO_0** Driver

Using the following class template

1. Implement ALL class methods.
2. All methods must function work as expected of their method name.

```
#ifndef LABGPIO_H
#define LABGPIO_H
class LabGPIO_0
{
private:
    /**
     * port, pin and any other variables should be placed here.
     * NOTE: that the state of the pin should not be recorded here. The true
     *      source of that information is embedded in the hardware registers
     */
public:
    /**
     * You should not modify any hardware registers at this point
     * You should store the port and pin using the constructor.
     *
     * @param {uint8_t} pin  - pin number between 0 and 32
     */
LabGPIO_0(uint8_t pin);
    /**
     * Should alter the hardware registers to set the pin as an input
     */
void setAsInput();
    /**
     * Should alter the hardware registers to set the pin as an input
```

```
     */
    void setAsOutput();
    /**
     * Should alter the set the direction output or input depending on the input.
     *
     * @param {bool} output - true => output, false => set pin to input
     */
    void setDirection(bool output);
    /**
     * Should alter the hardware registers to set the pin as high
     */
    void setHigh();
    /**
     * Should alter the hardware registers to set the pin as low
     */
    void setLow();
    /**
     * Should alter the hardware registers to set the pin as low
     *
     * @param {bool} high - true => pin high, false => pin low
     */
    void set(bool high);
    /**
     * Should return the state of the pin (input or output, doesn't matter)
     *
     * @return {bool} level of pin high => true, low => false
     */
    bool getLevel();
    ~LabGPIO_0();
};
#endif
```

Code Block 1: ***LabGPIO_0.hpp*** template

Part 2. Use Driver for an application

The application is to use an internal **AND** external switch to control an on board and external LED, respectively.

```
void vControlLED( void * pvParameters )
{
    /* Get Parameter */
    uint32_t param = (uint32_t)(pvParameters);
    /* Define Constants Here */
    /* Define Local Variables and Objects */

    /* Initialization Code */
    while(1)
    {
        /* Insert Loop Code */
    }
    /* Only necessary if above loop has a condition */
    xTaskDelete(NULL);
}
void vReadSwitch( void * pvParameters )
{
    /* Get Parameter */
    uint32_t param = (uint32_t)(pvParameters);
    /* Define Constants Here */
    /* Define Local Variables and Objects */

    /* Initialization Code */
    while(1)
    {
        /* Insert Loop Code */
    }
    /* Only necessary if above loop has a condition */
    xTaskDelete(NULL);}
```

Code Block 2: FreeRTOS Task Templates

Requirements:

? You **MUST NOT** use any pre-existing library such as a GPIO class for this assignment.

> You **MAY USE** LPC17xx.h it is not a library but a list of registers mapped to the appropriate locations.

1. You **MUST** use the two task templates above, you MUST pass a parameter to these tasks to inform them which switch or led they are reading/controlling.
2. You will need to create a **GPIO_1** library as well.
3. The task above must do the following:
    1. ***vReadSwitch:*** Will read from the internal **OR** external GPIO, depending on the input parameter, connected to the switches. If a button is **RELEASED** (not when pressed, but when it released), this task will **set** a global variable.
    2. ***vControlLED***: Will loop and toggle the state of your designated LED, specified by the input parameter, and clear that global variable flag.

4. In total, you will need to create 4 tasks, two **vReadSwitch** tasks, one for reading external switch and another for reading the internal switch two **vControlLED** tasks, one for toggling the internal and another for toggling the external LED.
5. Pass in a parameter to change the behavior of each task. See *code block 2.*

Upload only relevant `.cpp` files into canvas. A good example is: *main.cpp, LabGPIO.hpp, LabGPIO.cpp*. See Canvas for rubric and grade breakdown.

> **Extra Credit**
>
> Add a flashy easter egg feature to your assignment, with your new found LED and switch powers! The extra credit is subject to the instructor's, ISA's and TA's discretion about what is worth the extra credit.
>
> Consider using additional switches and/or LEDs.

> **Extra Credit**
>
> Be able to handle multiple ports (port 1 and port 2) with a single class.

?

Revision #14
Created 1 year ago by Admin
Updated 2 weeks ago by sree harsha

?