香港中文大學
The Chinese University of Hong Kong

CENG2400 Embedded System Design

Lab 02: GPIO

Han ZHAO

hzhao@cse.cuhk.edu.hk

Step 0: Before the Lab beginning

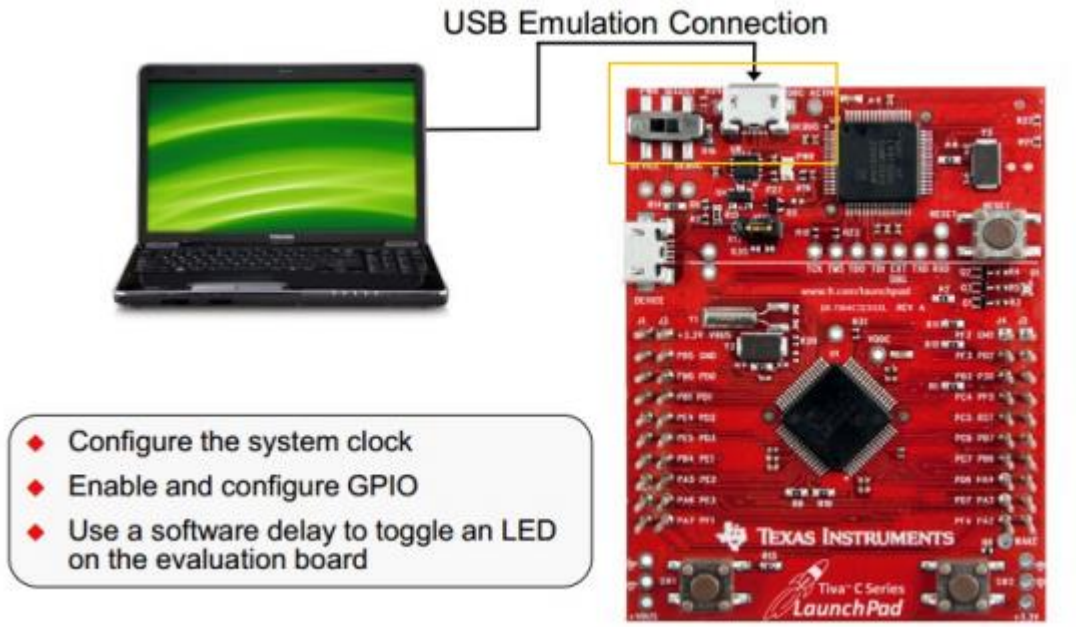
- Please proceed to the TA's desk to sign in, and provide your signature on the attendance sheet (Kindly refrain from signing in **on behalf of others**).
- Upon completing the sign-in process, each student should collect a board (Each individual is responsible for acquiring their own board and should not collect boards on behalf of others).

Outline

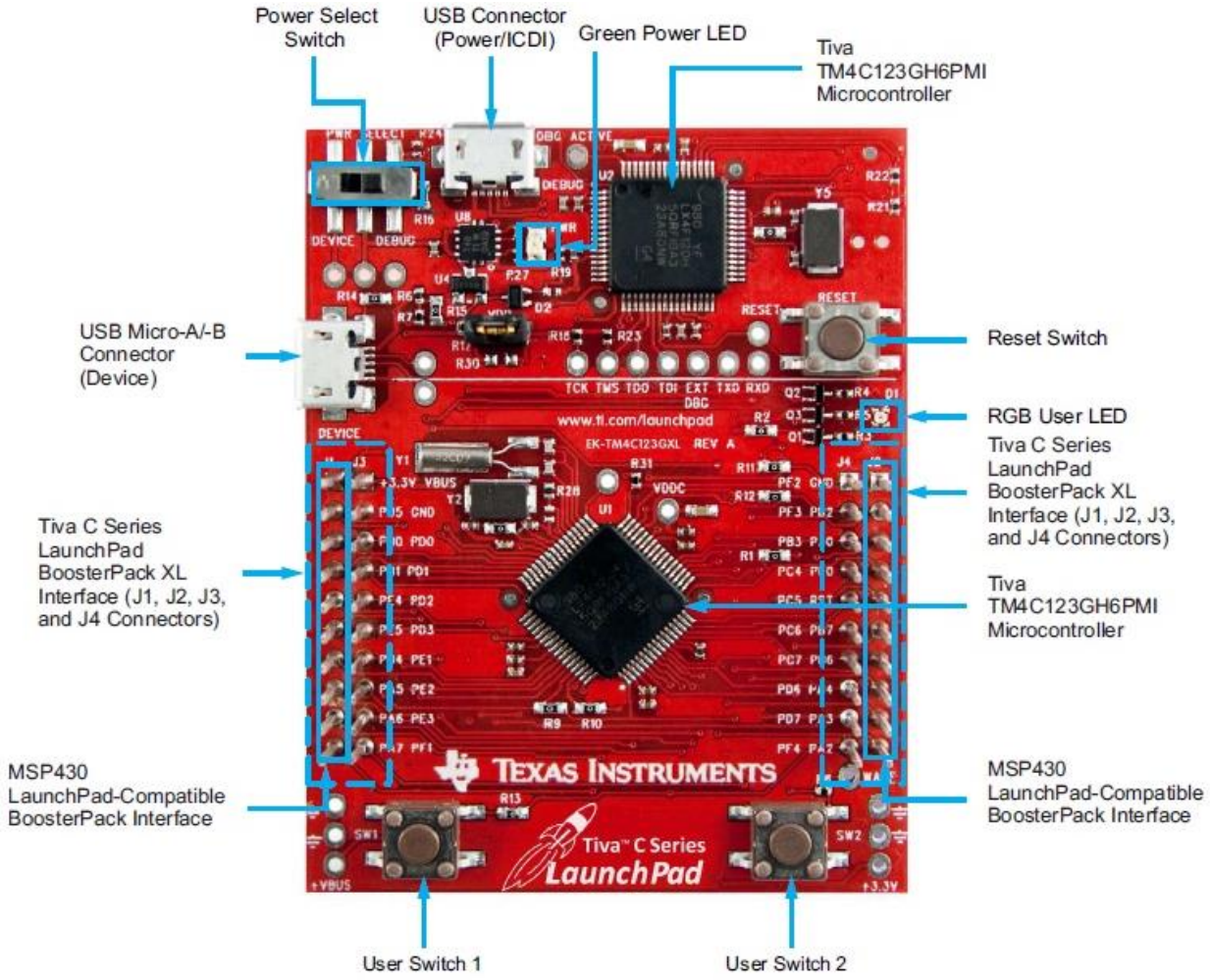
- **Step 1:** Learning GPIO
- **Step 2:** Building and running an example and learning about it
- **Step 3:** Doing your assignment (upload your code and video on blackboard between next lab)

Step 1: Learning GPIO

- In this lab we'll learn how to initialize the GPIO peripheral using TivaWare.
- We'll then use the GPIO output to blink an LED on the evaluation board.



Step 1: Learning GPIO



Tiva C Series TM4C123G LaunchPad Evaluation Board

Step 1: Learning GPIO

- GPIO (General Purpose Input/Output)
  - Can be configured as an input or an output. On reset, GPIOs default to being inputs.
  - In input mode, can generate interrupts on high level, low level, rising edge, falling edge, or both edges.
  - In output mode, can be configured for 2-mA, 4-mA, or 8-mA drive strength. The 8-mA drive strength configuration has optional slew rate control to limit the rise and fall times of the signal. On reset, GPIOs default to 2-mA drive strength.
  - Optional weak pull-up or pull-down resistors. On reset, GPIOs default to no pull-up or pull-down resistors.
  - Optional open-drain operation. On reset, GPIOs default to standard push/pull operation.
  - Can be configured to be a GPIO or a peripheral pin. On reset, the default is GPIO. Note that not all pins on all parts have peripheral functions, in which case the pin is only useful as a GPIO.

Step 1: Learning GPIO

- GPIO (General Purpose Input/Output)
  - Most useful: **GPIOPinRead()**
  - Function from `driverlib/gpio.h`

14.2.3.18 GPIOPinRead

Reads the values present of the specified pin(s).

Prototype: From TivaWave Peripheral Deiver Library Use Guide

```
int32_t
GPIOPinRead(uint32_t ui32Port,
            uint8_t ui8Pins)
```

Parameters:

ui32Port is the base address of the GPIO port.

ui8Pins is the bit-packed representation of the pin(s).

Description:

The values at the specified pin(s) are read, as specified by *ui8Pins*. Values are returned for both input and output pin(s), and the value for pin(s) that are not specified by *ui8Pins* are set to 0.

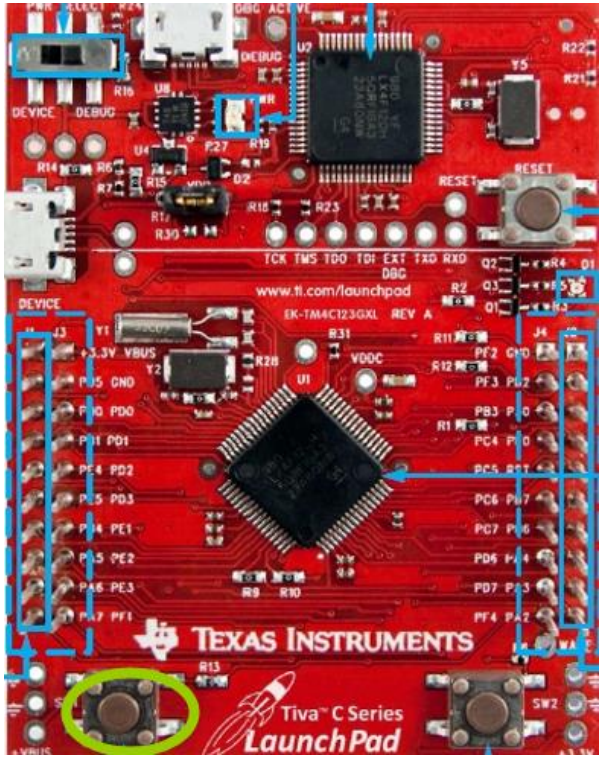
The pin(s) are specified using a bit-packed byte, where each bit that is set identifies the pin to be accessed, and where bit 0 of the byte represents GPIO port pin 0, bit 1 represents GPIO port pin 1, and so on.

Returns:

Returns a bit-packed byte providing the state of the specified pin, where bit 0 of the byte represents GPIO port pin 0, bit 1 represents GPIO port pin 1, and so on. Any bit that is not specified by *ui8Pins* is returned as a 0. Bits 31:8 should be ignored.

Step 1: Learning GPIO

- GPIO (General Purpose Input/Output)
- Example:
  - ButtonState = GPIOPinRead(GPIO_PORTF_BASE, GPIO_PIN_4);



Step 1: Learning GPIO

- GPIO (General Purpose Input/Output)
  - Most useful: **GPIOPinWrite()**
  - Function from `driverlib/gpio.h`

14.2.3.46 GPIOPinWrite

Writes a value to the specified pin(s).

Prototype:

```
void
GPIOPinWrite(uint32_t ui32Port,
             uint8_t ui8Pins,
             uint8_t ui8Val)
```

Parameters:

ui32Port is the base address of the GPIO port.

ui8Pins is the bit-packed representation of the pin(s).

ui8Val is the value to write to the pin(s).

Description:

Writes the corresponding bit values to the output pin(s) specified by *ui8Pins*. Writing to a pin configured as an input pin has no effect.

The pin(s) are specified using a bit-packed byte, where each bit that is set identifies the pin to be accessed, and where bit 0 of the byte represents GPIO port pin 0, bit 1 represents GPIO port pin 1, and so on.

Returns:

None.

ATTENTION: MASK + VALUE

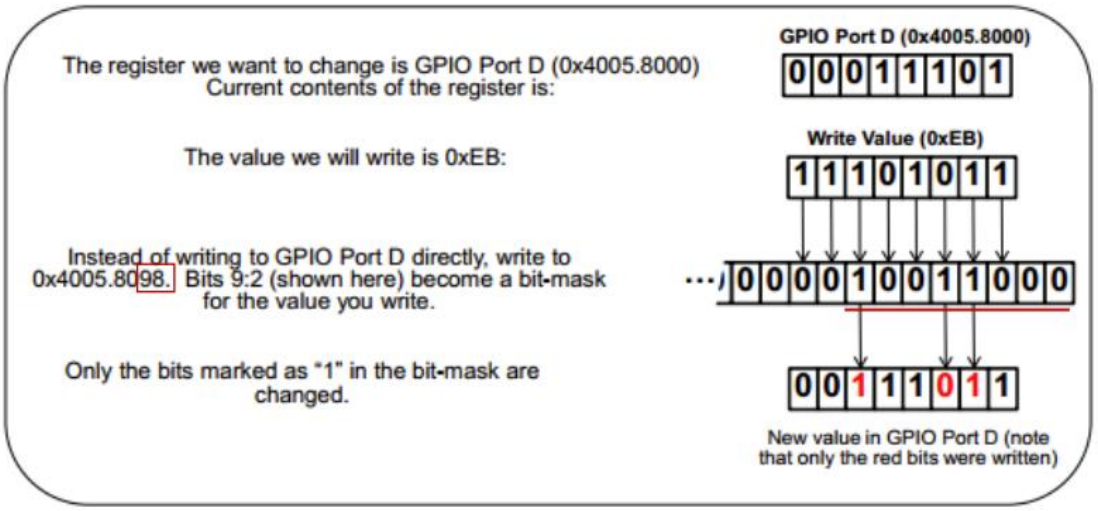
Value 0xEB ->	11101011
Mask 0x98 shift2	0010011000
Effect	1 01

Step 1: Learning GPIO

- GPIO (General Purpose Input/Output)
  - Most useful: **GPIOPinWrite()**

Each GPIO port has a base address. You can write an 8-bit value directly to this base address and all eight pins are modified. If you want to modify specific bits, you can use a bit-mask to indicate which bits are to be modified. This is done in hardware by mapping each GPIO port to 256 addresses. Bits 9:2 of the address bus are used as the bit mask.

In hardware:



Interface: **GPIOPinWrite(GPIO_PORTD_BASE, GPIO_PIN_5|GPIO_PIN_2|GPIO_PIN_1, 0xEB);** This is much easier

Note: you specify base address, bit mask, and value to write.
The GPIOPinWrite() function determines the correct address for the mask.

Step 1: Learning GPIO

- Example:
  - GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, magic_number);

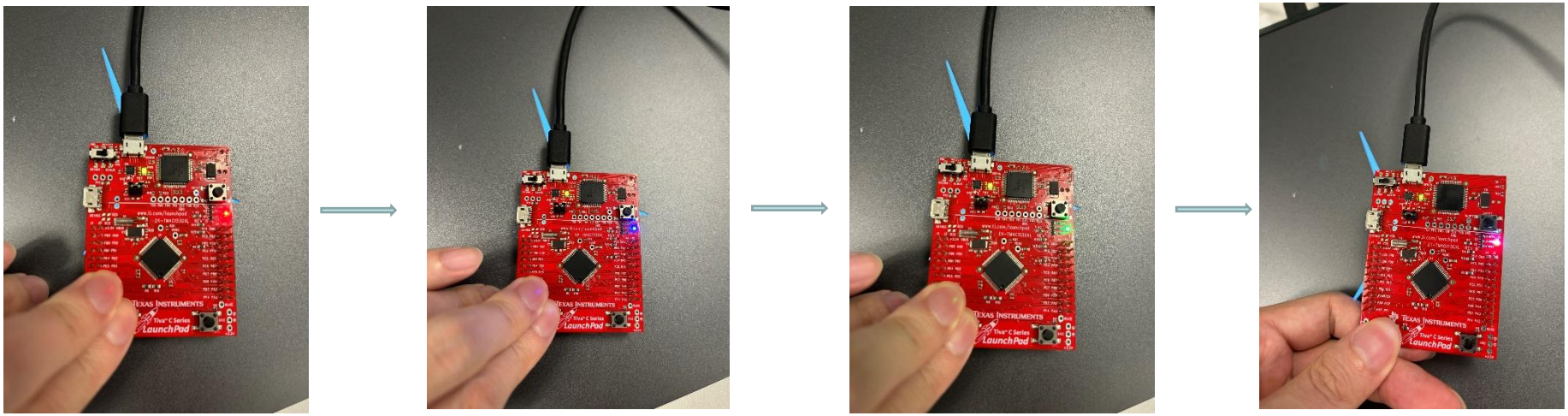
					G	B	R		
Value	PF7	PF6	PF5	PF4	PF3	PF2	PF1	PF0	LED
0	0	0	0	0	0	0	0	0	Off
2	0	0	0	0	0	0	1	0	Red
4	0	0	0	0	0	1	0	0	Blue
6	0	0	0	0	0	1	1	0	Purple (Red + Blue)
8	0	0	0	0	1	0	0	0	Green
10	0	0	0	0	1	0	1	0	Yellow (Red + Green)
12	0	0	0	0	1	1	0	0	Cyan (Green + Blue)
14	0	0	0	0	1	1	1	0	White (Red + Green + Blue)
16	0	0	0	1	0	0	0	0	Off

Step2: Running an example

- So far, we only use GPIOPinWrite(), so the GPIOs are under output mode. We can use GPIOPinRead() to get input from user buttons. We want:
  - When switch 1 is not pressed, the system displays a repeating sequence of colors (same as before).
  - When switch 1 is pressed, the LED flash white (all LEDs on) and off (all LEDs off) until the switch is released.

Step2: Running an example

- However, what will happen if switch 1 is pressed when the red light is on?



- Only a loop of red blue green is completed , the white light will flicker.

Step2: Running an example

- WHY?

```
while(1)
{
    ButtonState = GPIOPinRead(GPIO_PORTF_BASE, GPIO_PIN_4);
    if (ButtonState == 0) {//flash white
        //Control_RGB_LEDs(1, 1, 1);
        GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3);
        SysCtlDelay(W_DELAY);
        //Control_RGB_LEDs(0, 0, 0);
        GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, 0x00);
        SysCtlDelay(W_DELAY);
    }
    else {// sequence R,G,B
        //Control_RGB_LEDs(1, 0, 0);
        GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, GPIO_PIN_1);
        SysCtlDelay(RGB_DELAY);
        // Control_RGB_LEDs(0, 1, 0);
        GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, GPIO_PIN_2);
        SysCtlDelay(RGB_DELAY);
        //Control_RGB_LEDs(0, 0, 1);
        GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, GPIO_PIN_3);
        SysCtlDelay(RGB_DELAY);
    }
}
```

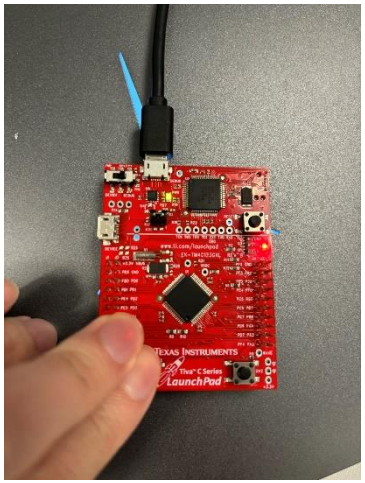
  - In each iteration of the **while** loop, if the **if** condition is satisfied, all the steps within the **red box** must be executed in their entirety before proceeding to the next iteration where the condition is re-evaluated.

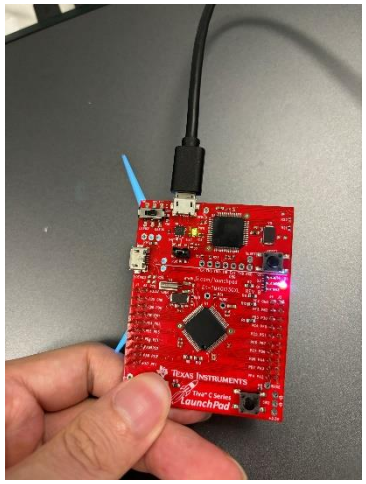
- More details: in **Lec03**

Step3: Assignment

- Assignment
  - To improve the previous code, ensure that the white light starts blinking after the current stage/color of the RGB sequence when the button is pressed, without waiting for the full RGB sequence to finish illuminating.

- For example:
  - If the red light is currently on and switch 1 is pressed, the system should automatically start blinking the white light after the red light turns off, without waiting for the blue and green lights to complete.





Step3: Assignment

- Notice
  - Upload your **code** and **a video** showing the effect of your code on blackboard.
- Deadline:
  - Before the next lab (next Tuesday, Sep. 24th).
- Requirements of the video:
  - The video should demonstrate that the white light starts blinking immediately when the button is pressed, without waiting for the red, blue, and green lights to finish illuminating.
  - For instance, you can press switch 1 separately when the red, blue, or green light is illuminated to observe whether the LED enters the white light flashing state immediately after the current color is extinguished.

Thanks for listening!

Q & A