Lab 1: Getting Started with ARM Cortex M4 Microcontrollers

Objectives

- Use the TIVA C TM4C123G microcontroller GPIO to drive LEDs and read keys status
- Use the System Tick timer to implement precise delay
- Configure the microcontroller clocking system
- Interact with I/O peripherals by reading/writing I/O registers
- Develop a simple embedded application in C

Experiments

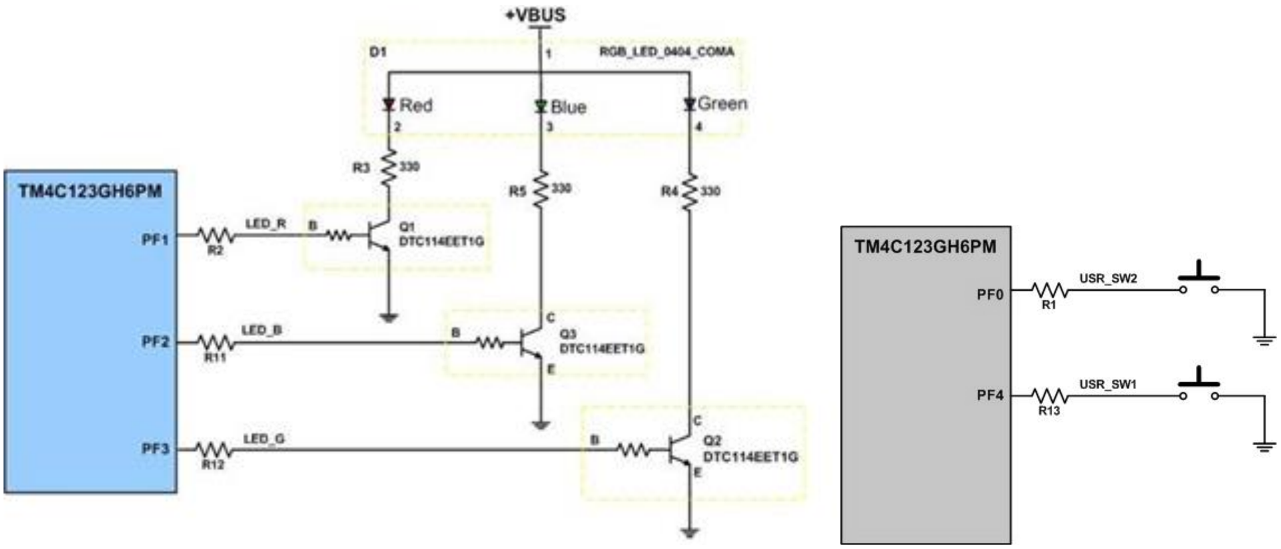
Experiment 1:

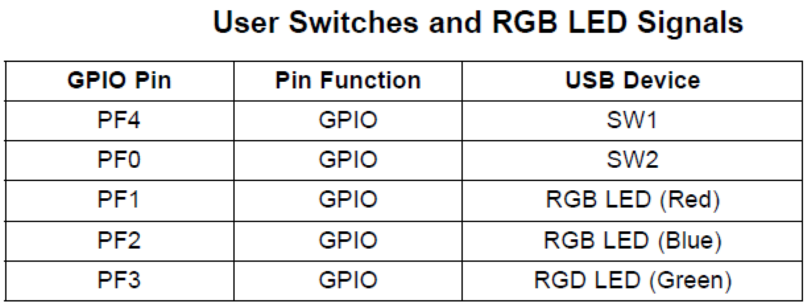
Follow the tutorial (µVision Tutorial.pdf) using Keil µVision IDE to develop and run a simple embedded application that blinks the red LED on the Launchpad with a blinking frequency of 2Hz. Replace delayMs() function with another that uses SysTick timer in order to implement precise delay.

Experiment 2:

Develop an embedded application using Keil µVision IDE to implement the requirements described below. Use System Tick timer to implement any needed delay.

a. When the application starts the blue LED (only) flashes with a frequency of 2Hz.
b. SW1 is used to change the color of the flashing LED (sequence: RED ➔ BLUE ➔ GREEN ➔ RED ➔ …). The change is done whenever SW1 gets pressed.
c. SW2 is used to control the flashing by pausing and resuming it. Pressing it while flashing stops the flashing. Pressing it again brings it back.





User Switches and RGB LED Signals

GPIO Pin	Pin Function	USB Device
PF4	GPIO	SW1
PF0	GPIO	SW2
PF1	GPIO	RGB LED (Red)
PF2	GPIO	RGB LED (Blue)
PF3	GPIO	RGD LED (Green)

General steps to follow for configuring GPIO:

1. **Provide clock** to the peripheral and access to peripheral registers.
2. For **locked pins** (like PF0): **unlock** the commit register then **commit** configuration changes.
3. Set the pin **alternate function select** to 0 for the pin to be used as a GPIO (set by default).
4. Set the **digital enable option** on the GPIO pin to use the pin as a digital input or output.
5. Set the **direction** of pins, to set each to be an input or output.
6. Set the **pulldown or pullup select option** for the input pins, to be set as active high or active low.

Notes on GPIO registers:

- The following GPIO registers are the ones you need to use in the experiments above: **GPIODATA, GPIOAFSEL, GPIODEN, GPIODIR, GPIOPUR, GPIOPDR, GPIOLOCK, GPIOCR**.
- The GPIO commit control registers (**GPIOCR**) provide a **layer of protection** against accidental programming of critical hardware peripherals. The GPIOCR register is designed to prevent accidental programming of the registers that control connectivity to the NMI and JTAG/SWD debug hardware.
- **PF0** is configured as a GPIO by default but is **locked** and can only be reprogrammed by **unlocking the pin in the GPIOLOCK register** and **committing it by setting the GPIOCR** (commit control) register. Check the Microcontroller datasheet to find any info you need.
- The value of the GPIOCR register determines which bits of the GPIO Alternate Function Select (GPIOAFSEL) register, GPIO Pull Up Select (GPIOPUR) register, GPIO Pull-Down Select (GPIOPDR) register, and GPIO Digital Enable (GPIODEN) registers are committed when a write to these registers is performed.
- **The contents of the GPIOCR register can only be modified if the status in the GPIOLOCK register is unlocked.** Writes to the GPIOCR register are ignored if the status in the GPIOLOCK register is locked.

References

- µVision Tutorial (Lab 0)
- Setting the System Clock notes (Lab 0)
- Tiva™ C Series TM4C123G LaunchPad Evaluation Board__User's Guide__spmu296
- Tiva™ C Series TM4C123GH6PM Microcontroller datasheet (Rev. E)__spms376e
- SysTick notes

Lab Report [10 pts]
(Deadline: Monday of next week 11:59 pm) (Individual submission)

1. [1 pts] Provide your C code of the experiments conducted in the lab.

2. [4.5 pts] Develop an application that generates a square wave on one of **PORTC** pins. Initially the square wave has a frequency of 1Hz. The user should be able to increase or decrease the frequency with a step pf 0.2Hz using SW1 (+0.2Hz) and SW2 (-0.2Hz). Use external LEDs to verify your work.

   Provide your code along with a small sized video of the application running on the Launchpad.

3. [4.5 pts] Develop a traffic light system on our TIVA Launchpad that uses the onboard RGB LED to continuously flash in this sequence without user intervention:
   a. Green for 4 sec
   b. Yellow for 1 sec
   c. Red for 4 sec
   d. Yellow for 1 sec
   e. And Repeat from step (a)

   Additionally, SW2 is used as the traffic light system on/off. If it gets pressed the system turns off and if it’s released the system turns on.

   Provide your code along with a small sized video of the application running on the Launchpad.