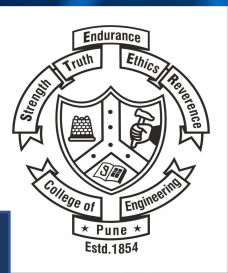
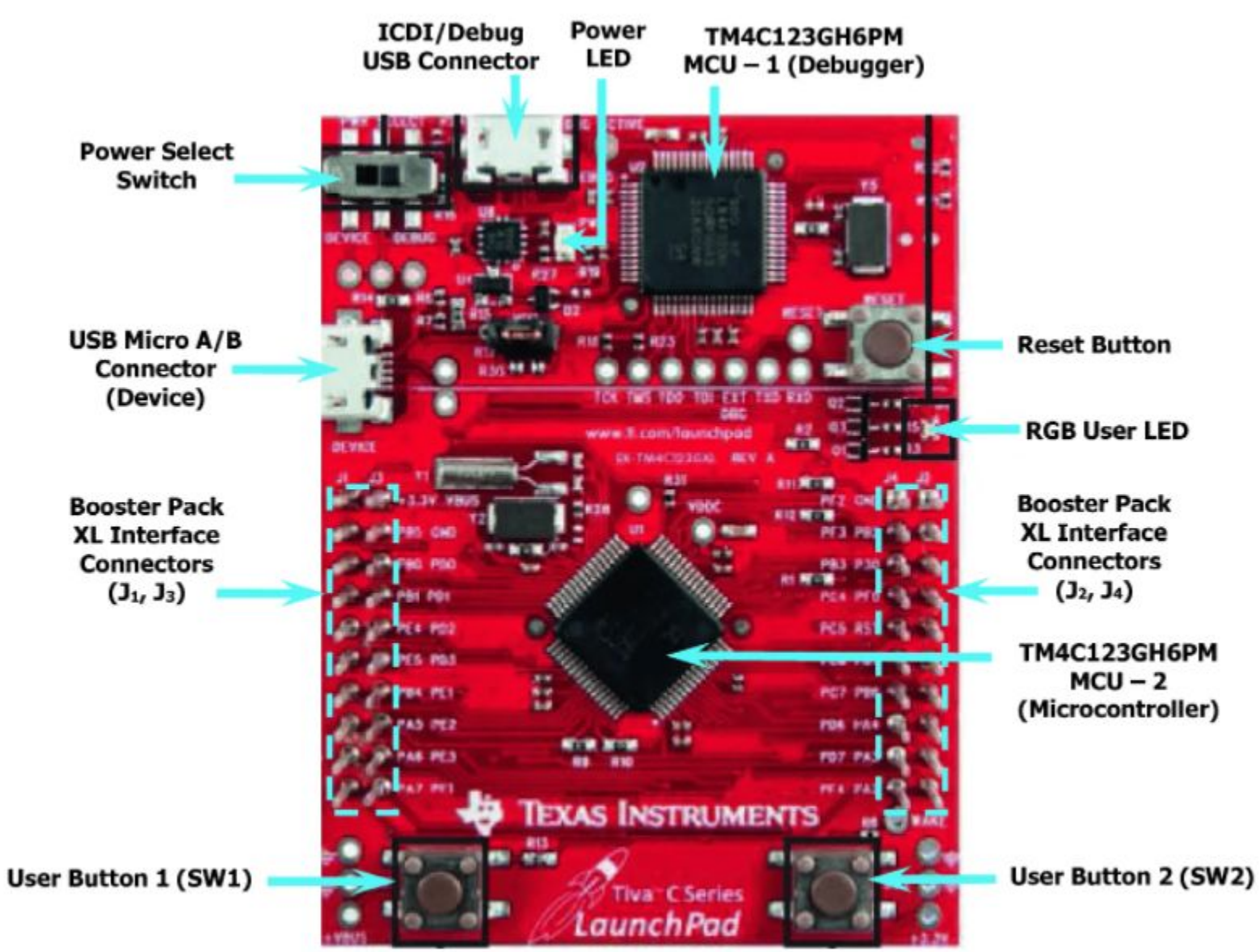
MAP Lab

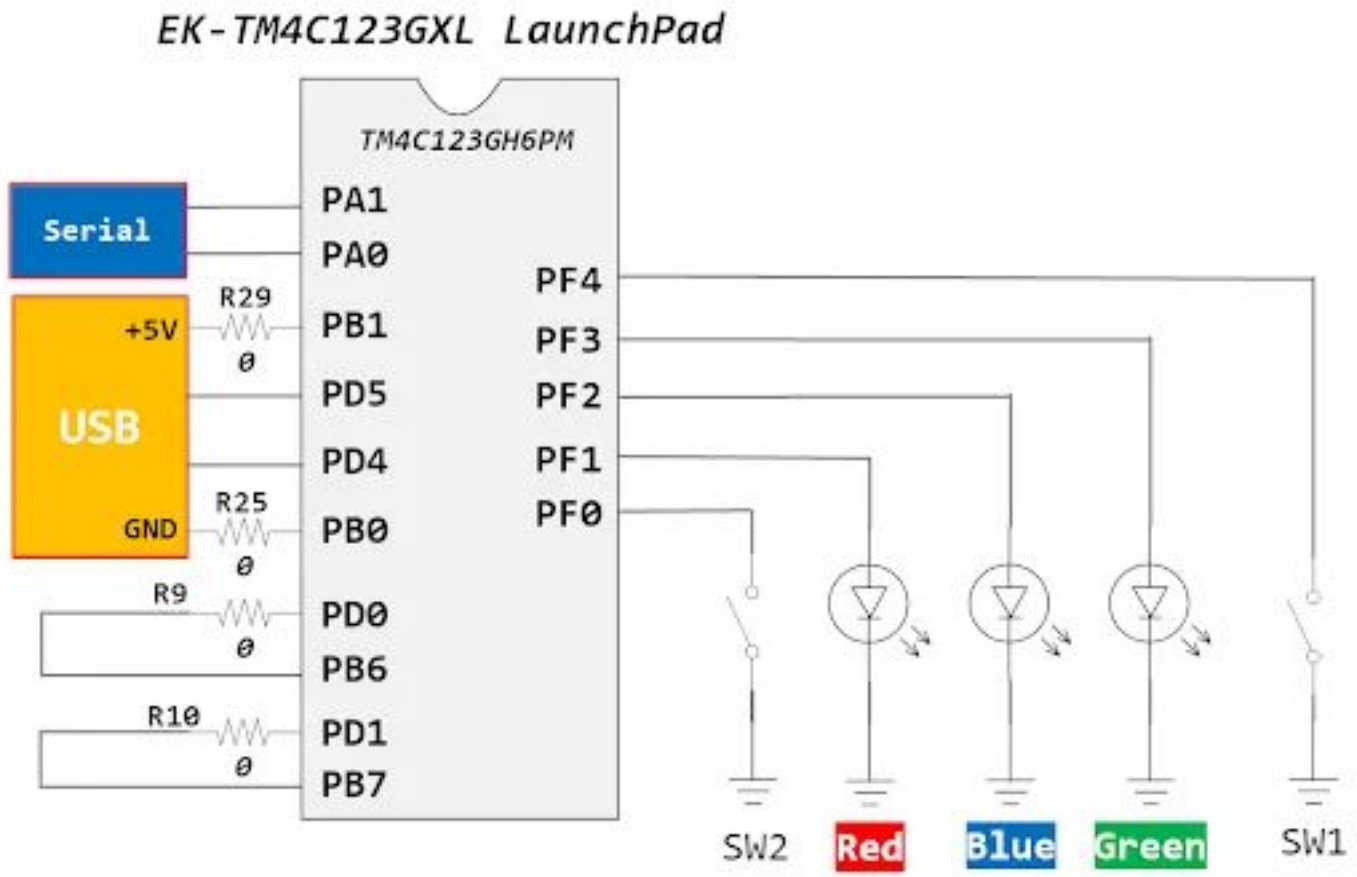
LAB 3 : Use Push Button to Control LED



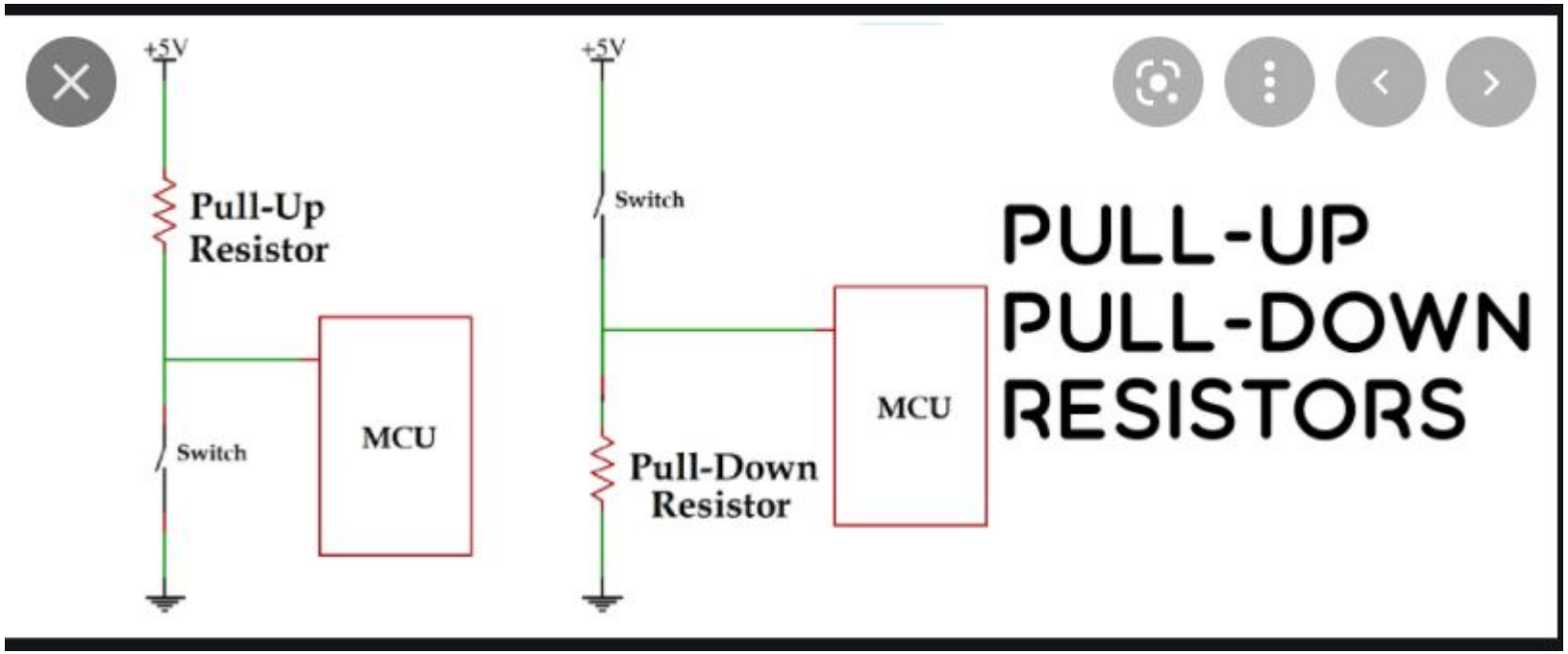
AIM

In this tutorial, we will learn how to use on-board Push buttons of TM4C123G TIVA launchpad to control onboard LEDs. Firstly, you should know how to use GPIO pins of TM4C123G6PM microcontroller as digital output pins


ICDI/Debug USB Connector
Power LED
TM4C123GH6PM MCU – 1 (Debugger)
Power Select Switch
USB Micro A/B Connector (Device)
Reset Button
RGB User LED
Booster Pack XL Interface Connectors (J1, J3)
Booster Pack XL Interface Connectors (J2, J4)
TM4C123GH6PM MCU – 2 (Microcontroller)
User Button 1 (SW1)
User Button 2 (SW2)
TEXAS INSTRUMENTS
Tiva C Series
LaunchPad


EK-TM4C123GXL LaunchPad
TM4C123GH6PM
Serial
+5V
USB
GND
R29
0
R25
0
R9
0
R10
0
PA1
PA0
PB1
PD5
PD4
PB0
PD0
PB6
PD1
PB7
PF4
PF3
PF2
PF1
PF0
SW2
Red
Blue
Green
SW1

There is no pull-up resistor available with switch-one and switch-two. But the good thing is that the TM4C1235G series microcontroller has internal pull-up and pull-down registers associated with each port. We can configure these resistors using a respective GPIO port register. In the programming section, we will see how to configure this register.



Mechanical switches are commonly used to feed any parameters to the digital systems. The switches can be interfaced to a microcontroller using digital inputs. The software program for switch interfacing can be implemented using one of the following methods.

1. Polling based method
2. Interrupt based method

We will discuss polling based switch interfacing in this tutorial. Before proceeding further, it is important to first make ourselves familiar with the physical behavior of switch and be will describe switch bouncing next, which is one of the critical attribute of its physical behavior

Switch bouncing

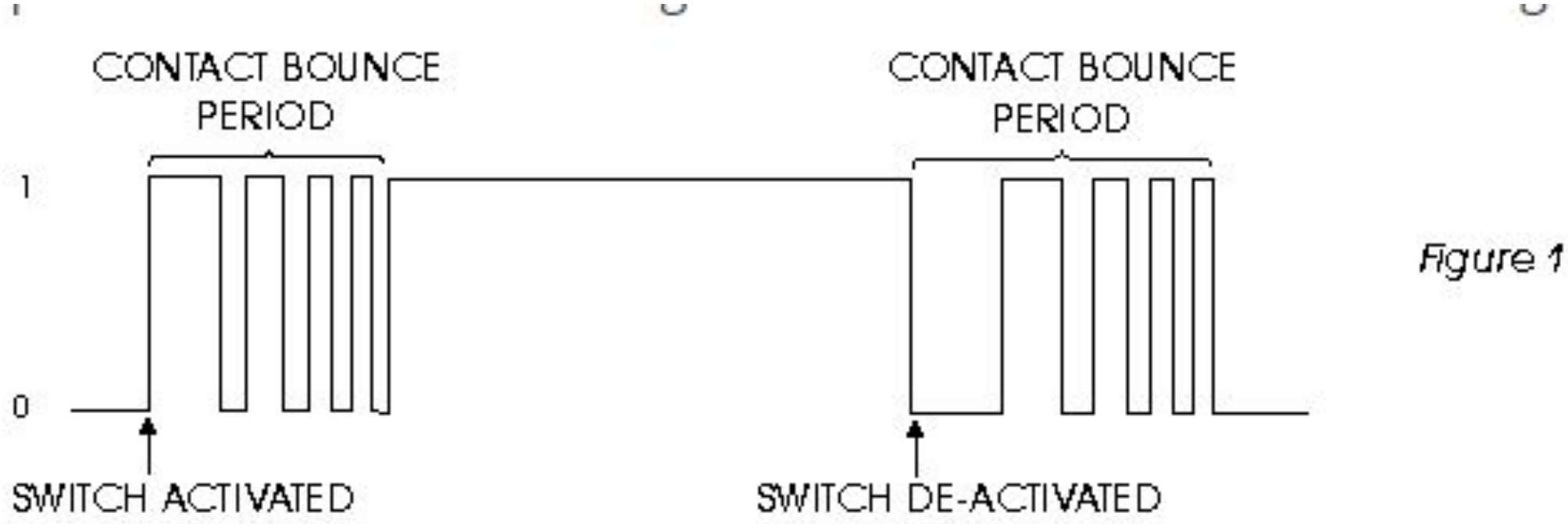


Figure 1

Controlling an LED with a push button using Tiva Launchpad

Pin	Function
PF1	LED – Red
PF4	On-Board Switch-2

whenever a user presses the push button that is connected with the PF0 pin of TM4C123G6PM microcontroller, LED will turn on. Moreover, as soon as the user releases the push button, the LED turns off.

Steps

- Include Header File TM4C123GH6PM.h
- GPIO Pins Clock Enable Register
- GPIO Lock and Commit Registers
- Pull-Up Resistor Register CR PUR
- Direction Control Register DIR DEN

```
#define WATCHDOG1_TEST_R        (*((volatile unsigned long *)0x40001418))
#define WATCHDOG1_LOCK_R        (*((volatile unsigned long *)0x40001C00))

//******************************************************************************
//
// GPIO registers (PORTA)
//
//******************************************************************************
#define GPIO_PORTA_DATA_BITS_R  ((volatile unsigned long *)0x40004000)
#define GPIO_PORTA_DATA_R       (*((volatile unsigned long *)0x400043FC))
#define GPIO_PORTA_DIR_R        (*((volatile unsigned long *)0x40004400))
#define GPIO_PORTA_IS_R         (*((volatile unsigned long *)0x40004404))
#define GPIO_PORTA_IBE_R        (*((volatile unsigned long *)0x40004408))
#define GPIO_PORTA_IEV_R        (*((volatile unsigned long *)0x4000440C))
#define GPIO_PORTA_IM_R         (*((volatile unsigned long *)0x40004410))
#define GPIO_PORTA_RIS_R        (*((volatile unsigned long *)0x40004414))
#define GPIO_PORTA_MIS_R        (*((volatile unsigned long *)0x40004418))
#define GPIO_PORTA_ICR_R        (*((volatile unsigned long *)0x4000441C))
#define GPIO_PORTA_AFSEL_R      (*((volatile unsigned long *)0x40004420))
#define GPIO_PORTA_DR2R_R       (*((volatile unsigned long *)0x40004500))
#define GPIO_PORTA_DR4R_R       (*((volatile unsigned long *)0x40004504))
#define GPIO_PORTA_DR8R_R       (*((volatile unsigned long *)0x40004508))
#define GPIO_PORTA_ODR_R        (*((volatile unsigned long *)0x4000450C))
#define GPIO_PORTA_PUR_R        (*((volatile unsigned long *)0x40004510))
#define GPIO_PORTA_PDR_R        (*((volatile unsigned long *)0x40004514))
#define GPIO_PORTA_SLR_R        (*((volatile unsigned long *)0x40004518))
#define GPIO_PORTA_DEN_R        (*((volatile unsigned long *)0x4000451C))
#define GPIO_PORTA_LOCK_R       (*((volatile unsigned long *)0x40004520))
#define GPIO_PORTA_CR_R         (*((volatile unsigned long *)0x40004524))
#define GPIO_PORTA_AMSEL_R      (*((volatile unsigned long *)0x40004528))
#define GPIO_PORTA_PCTL_R       (*((volatile unsigned long *)0x4000452C))
#define GPIO_PORTA_ADCCTL_R     (*((volatile unsigned long *)0x40004530))
#define GPIO_PORTA_DMACTL_R     (*((volatile unsigned long *)0x40004534))

//******************************************************************************
//
// GPIO registers (PORTB)
//
//******************************************************************************
#define GPIO_PORTB_DATA_BITS_R  ((volatile unsigned long *)0x40005000)
#define GPIO_PORTB_DATA_R       (*((volatile unsigned long *)0x400053FC))
#define GPIO_PORTB_DIR_R        (*((volatile unsigned long *)0x40005400))
#define GPIO_PORTB_IS_R         (*((volatile unsigned long *)0x40005404))
#define GPIO_PORTB_IBE_R        (*((volatile unsigned long *)0x40005408))
#define GPIO_PORTB_IEV_R        (*((volatile unsigned long *)0x4000540C))
```

Include Header File

```
#include "TM4C123GH6PM.h"
```

TM4C123GH6PM Microcontroller GPIO pins

TM4C123GH6PM belongs to the ARM Cortex M4 microcontroller series. It has six GPIO ports such as PORTA, PORTB, PORTC, PORTD, and PORTE. Each port has a different number of pins as given in this table.

GPIO Ports	Pins
PORTA	PA0 – PA7
PORTB	PB0- PB7
PORTC	PC0 – PC7
PORTD	PD0 – PD7
PORTE	PE0 – PE5
PORTF	PF0-PF7

GPIO Pins Clock Enable Register

GPIO Pins Clock Enable Register

The first step in GPIO configuration is to enable the clock for a particular peripheral you want to enable. A particular port can be enabled by setting an appropriate bit field for the required GPIO port in the RCGCGPIO register.

RCGCGPIO Register is on pg. No. 340 of datasheet

```
RCGCGPIO |= 0x01 //Enable clock for PORTA
RCGCGPIO |= 0x02 //Enable clock for PORTB
RCGCGPIO |= 0x04 //Enable clock for PORTC
RCGCGPIO |= 0x08 //Enable clock for PORTD
RCGCGPIO |= 0x01 //Enable clock for PORTE
RCGCGPIO |= 0x02 //Enable clock for PORTF
```

TM4C123G GPIODATA Register

The GPIO port pins of TM4C123G are multiplexed with different peripherals such as digital I/O, PWM, serial communication, etc. But each pin can be used for only one functionality at a time. GPIODEN register is used to enable GPIO pins as digital input-output pins.

When the port pin is configured as GPIO pins, the GPIODATA register is used to read and write data on the registers. If the pin is configured as a digital output pin, the data written to the GPIODATA register reflects on the corresponding output pin.

TM4C123G GPIODIR Direction Control Register

A GPIODIR register decides, which pins of the PORT will be configured either as a digital input or a digital output. The individual configuration capability of each GPIO is applicable to other registers of the GPIO port pins.

If we want to enable a pin of a port as digital input-output the corresponding bit on the GPIODIR register should be set or reset. If we want to configure a particular pin of any port as a digital input pin, the corresponding e data direction bit should be cleared. Similarly, if we want to configure a particular pin of any port as a digital output pin, the corresponding data direction bit should be set to one.

GPIO Lock and Commit Registers

The next register is a GPIOLOCK register. It enables write access to GPIOCR register. In order to unlock access to the GPIOCR register, we must initialize the GPIOLOCK register with 0x4C4F.434B value.

Pull-Up Resistor Register

Because we will be using an internal pull-up register with a PF4 pin which will be used as a digital input pin. GPIOPUR register is used to enable or disable internal pull-up register with any GPIO pin. But to enable write to GPIOPUR, we first need to enable GPIOCR register. Otherwise, write operation to GPIOPUR register will not commit

Assignment 2 :

1. Compute XOR from 1 to n

```
Input : n = 6
Output : 7
// 1 ^ 2 ^ 3 ^ 4 ^ 5 ^ 6  = 7

Input : n = 7
Output : 0
  // 1 ^ 2 ^ 3 ^ 4 ^ 5 ^ 6 ^ 7 = 0
```

```
if (n % 4 == 0)
        return n;

    // If n%4 gives
remainder 1
    if (n % 4 == 1)
        return 1;

    // If n%4 gives
remainder 2
    if (n % 4 == 2)
        return n + 1;

    // If n%4 gives
remainder 3
    return 0;
```

```
Number Binary-Repr  XOR-from-1-to-n
1           1           [0001]
2          10           [0011]
3          11           [0000]  <----- We get a 0
4         100           [0100]  <----- Equals to n
5         101           [0001]
6         110           [0111]
7         111           [0000]  <----- We get 0
8        1000           [1000]  <----- Equals to n
9        1001           [0001]
10       1010           [1011]
11       1011           [0000] <------ We get 0
12       1100           [1100] <------ Equals to n
```

Assignment 3 :

1. What do you understand by Interrupt Latency?
2. Is it possible for a variable to be both volatile and const?
3. What is a reentrant function?
4. What are the reasons for Interrupt Latency and how to reduce it?
5. swap two numbers without using a temporary variable?