TM4C123 GPIO Programming

J.Shankarappa

GPIO & Special Purpose I/O

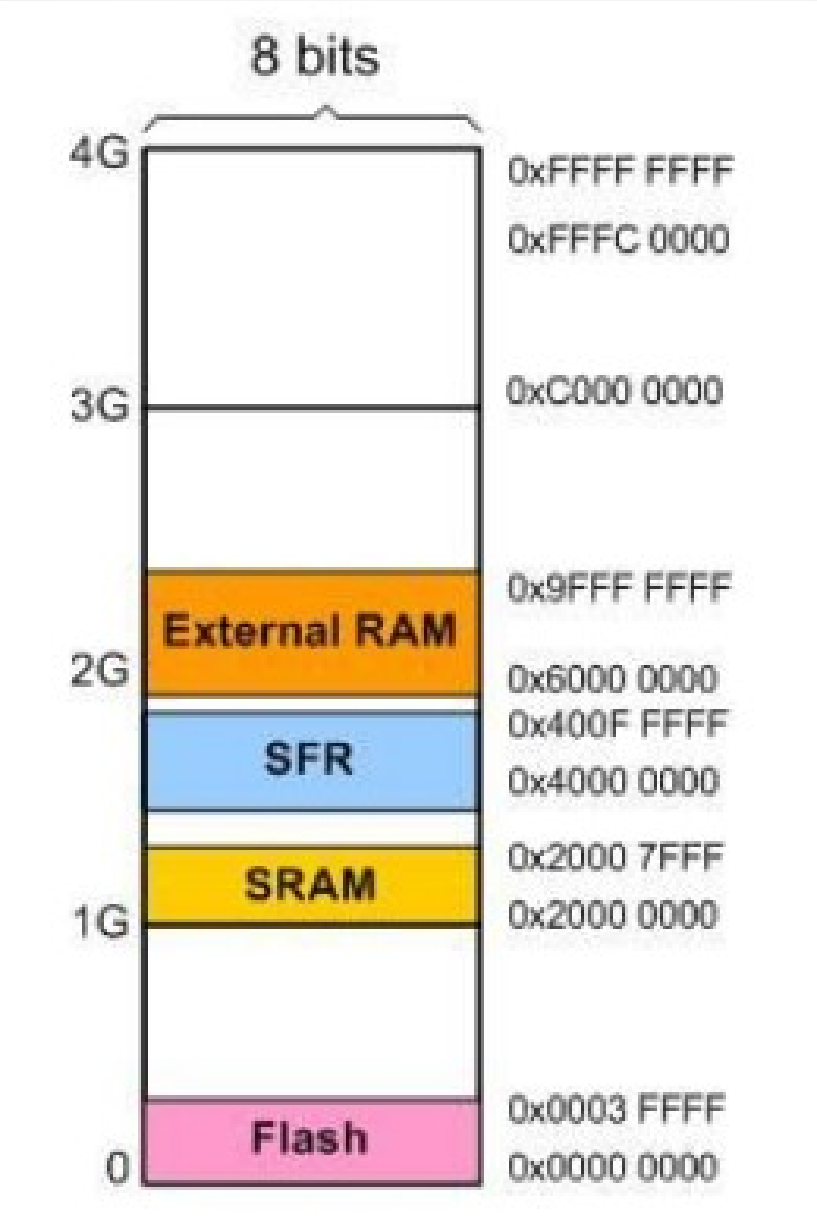
- While memory holds code and data for the CPU to process, the I/O ports are used by the CPU to access input and output devices.
- In the microcontroller we have two types of I/O :
  1. **General Purpose I/O (GPIO):** The GPIO ports are used for interfacing devices such as LEDs, Switches, LCD, Keypad, and so on.
  2. **Special purpose I/O:** These I/O ports have designated function such as ADC (Analog-to-Digital), Timer, UART, PWM and so on.

TM4C123GH6PM Memory Map

The TI LaunchPad uses the **TM4C123GH6PM** microcontroller, which has **256K** bytes (256KB) of on-chip Flash memory for code, **32KB** of on-chip SRAM for data, and a large number of on-chip peripherals.

	Allocated Size	**Allocated Address**
FLASH	256 KB	0x0000.0000 - 0x0003.FFFF
SRAM	32 KB	0x2000.0000 - 0x2000.7FFF
I/O	All the peripherals	0x4000.0000 - 0x400F.FFFF

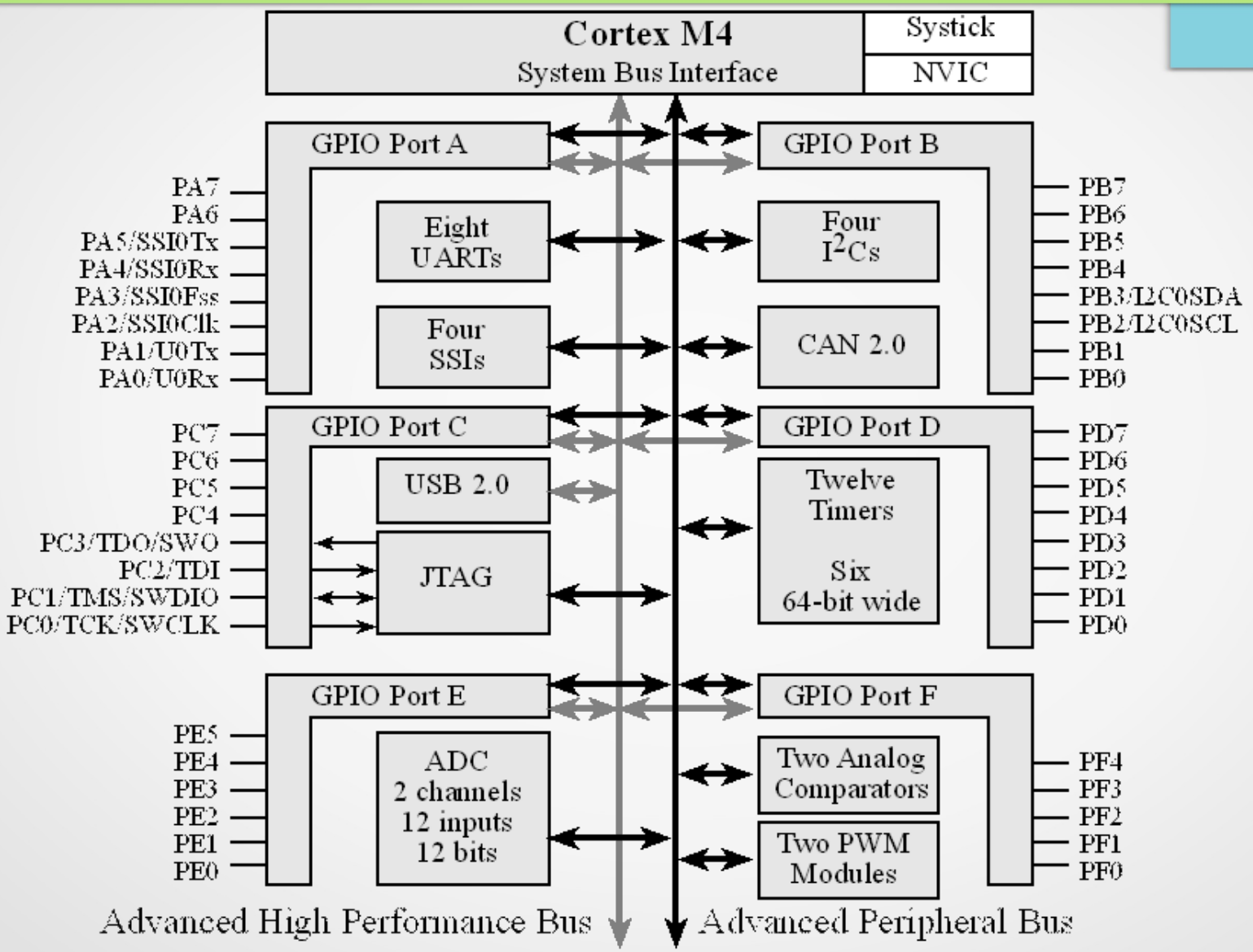
TM4C123GH6PM Memory Map ...



GPIO APB & AHB Bus

- The ARM chips have two buses:
  - Advanced Peripheral Bus (**APB**) and Advanced High-Performance Bus (**AHB**).
- The **AHB** bus is much faster than APB. The AHB allows one clock cycle to access the peripherals.
- The **APB** is slower and its access time is minimum of 2 clock cycles.

Input/Output Port Pins



GPIO APB Memory Map

- The I/O ports Base addresses assigned to the Port **A** - Port **F** for **APB** are as follows:
- GPIO Port **A** : 0x4000.4000
- GPIO Port **B** : 0x4000.5000
- GPIO Port **C** : 0x4000.6000
- GPIO Port **D** : 0x4000.7000
- GPIO Port **E** : 0x4002.4000
- GPIO Port **F** : 0x4002.5000

GPIO AHB Memory Map

- The I/O ports Base addresses assigned to the Port **A** - Port **F** for **AHB** are as follows:
- GPIO Port **A** : 0x4005.8000
- GPIO Port **B** : 0x4005.9000
- GPIO Port **C** : 0x4005.A000
- GPIO Port **D** : 0x4005.B000
- GPIO Port **E** : 0x4005.C000
- GPIO Port **F** : 0x4005.D000

GPIO Memory Map ...

- **4K** bytes of memory space is assigned to each of the GPIO port.
  - Each GPIO has a large number of **SFRs** associated with it and the GPIO DATA register supports bit-banding.
  - The GPIO DATA register is 8-bit wide. With bit-banding, it will need 256 words (4 bytes each, 1 KB total).
- There are many registers associated with each of the above GPIO ports and they have designated addresses in the memory map.
- The above addresses are the Base addresses meaning that within that base address we have registers associated with that port.

GPIO SFR Memory Map

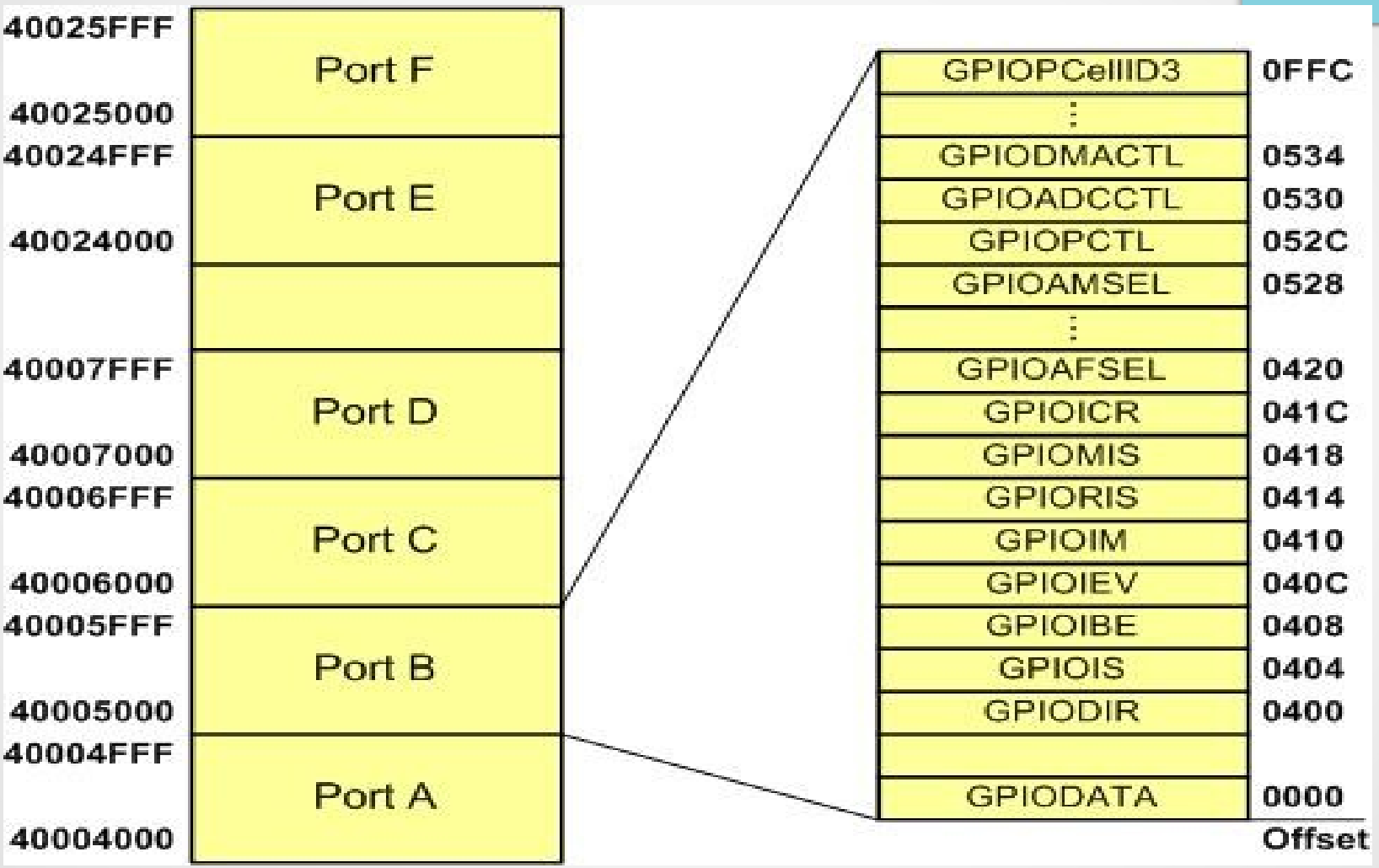


Table 10-6. GPIO Register Map, See Page 660

GPIO SFR Map

Name	Offset	Tivaware Name	Description
GPIODATA	0x000	GPIO_PORTx_DATA_R	GPIO Data
GPIODIR	0x400	GPIO_PORTx_DIR_R	GPIO Direction
GPIOIS	0x404	GPIO_PORTx_IS_R	GPIO Interrupt Sense
GPIOIBE	0x408	GPIO_PORTx_IBE_R	GPIO Interrupt Both Edges
GPIOIEV	0x40C	GPIO_PORTx_IEV_R	GPIO Interrupt Event
GPIOIM	0x410	GPIO_PORTx_IM_R	GPIO Interrupt Mask
GPIORIS	0x414	GPIO_PORTx_RIS_R	GPIO Raw Interrupt Status
GPIOMIS	0x418	GPIO_PORTx_MIS_R	GPIO Masked Interrupt Status
GPIOICR	0x41C	GPIO_PORTx_ICR_R	GPIO Interrupt Clear
GPIOAFSEL	0x420	GPIO_PORTx_AFSEL_R	GPIO Alternate Function Select
GPIODR2R	0x504	GPIO_PORTx_DR2R_R	GPIO 2-mA Drive Select
GPIODR4R	0x500	GPIO_PORTx_DR4R_R	GPIO 4-mA Drive Select
GPIODR8R	0x504	GPIO_PORTx_DR8R_R	GPIO 8-mA Drive Select
GPIOODR	0x508	GPIO_PORTx_ODR_R	GPIO Open Drain Select
GPIOPUR	0x50C	GPIO_PORTx_PUR_R	GPIO Pull-Up Select
GPIOPDR	0x510	GPIO_PORTx_PDR_R	GPIO Pull-Down Select
GPIOSLR	0x514	GPIO_PORTx_SLR_R	GPIO Slew Rate Control Select
GPIODEN	0x51C	GPIO_PORTx_DEN_R	GPIO Digital Enable
GPIOLOCK	0x520	GPIO_PORTx_LOCK_R	GPIO Lock
GPIOCR	0x524	GPIO_PORTx_CR_R	GPIO Commit
GPIOAMSEL	0x528	GPIO_PORTx_AMSEL_R	GPIO Analog Mode Select
GPIOPCTL	0X52C	GPIO_PORTx_PCTL_R	GPIO Port Control

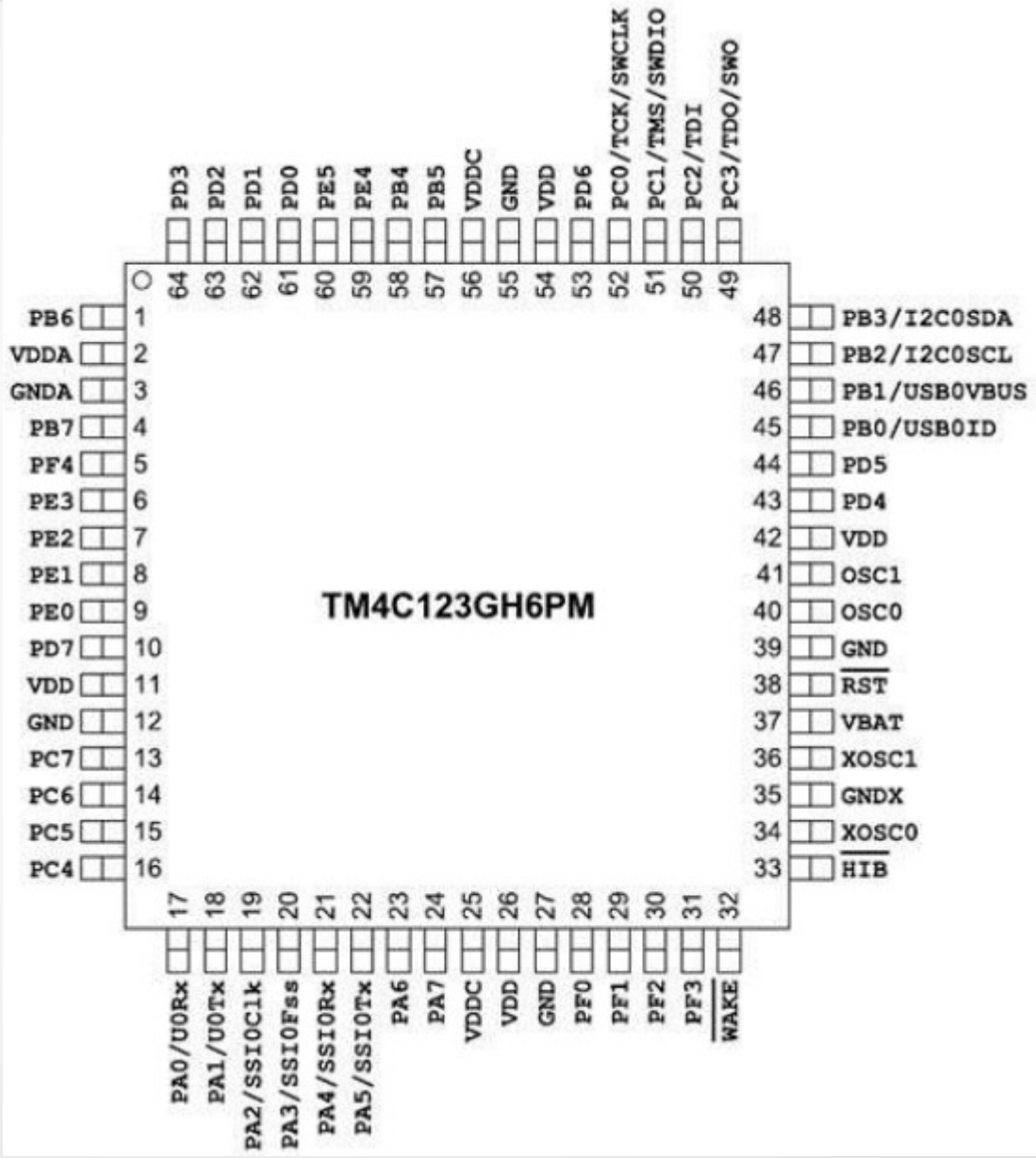
IO Pins on TI Tiva LaunchPad

- The ARM chip used in TI Tiva LaunchPad is Tiva C series **TM4C123GH6PM**.
- Ports **A**, **B**, **C**, **D**, **E**, and **F**.

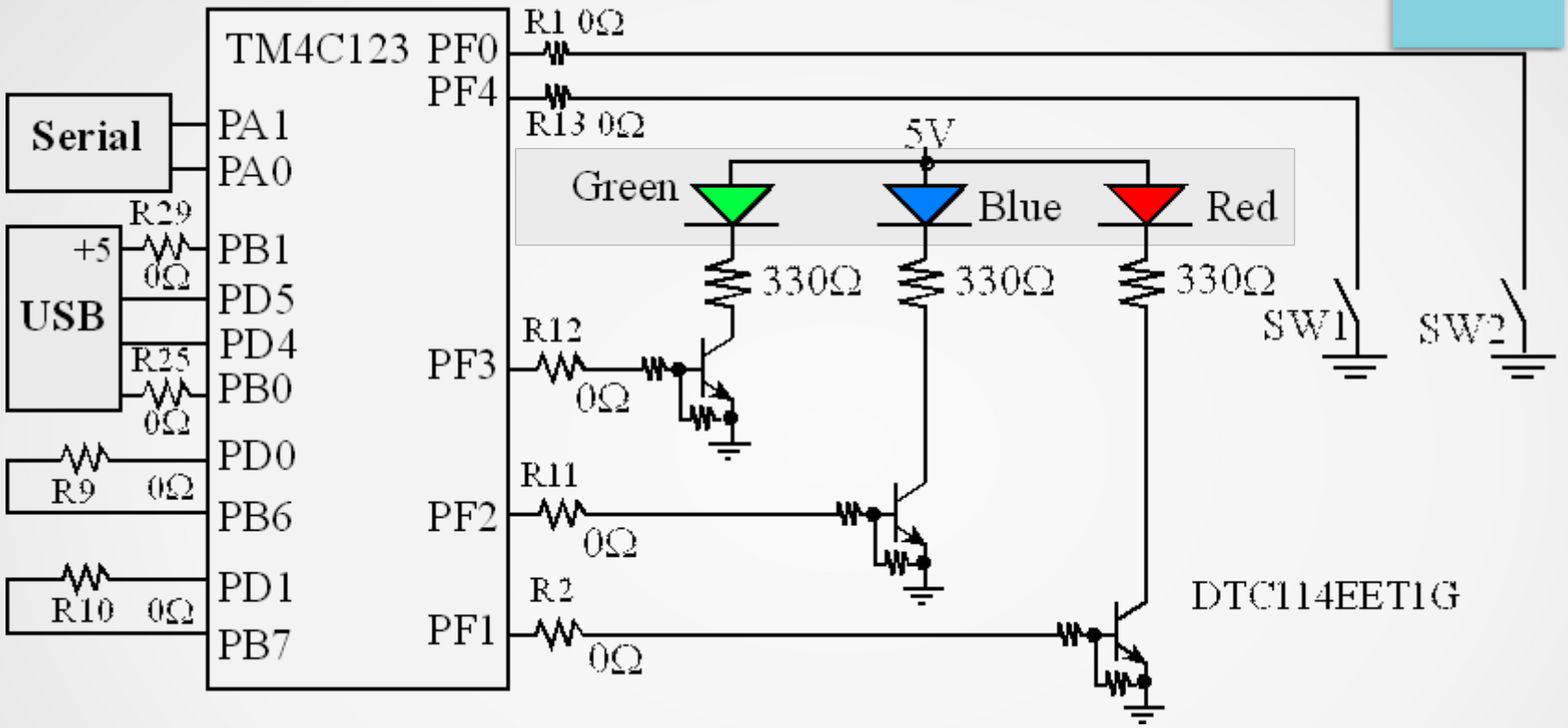
  The pins are designated as:

  - PA0 - PA7
  - PB0 - PB7
  - PC0 - PC7
  - PD0 - PD7
  - PE0 - PE5
  - PF0 – PF4

- Port-E and Port-F do not have all the 8 pins implemented.

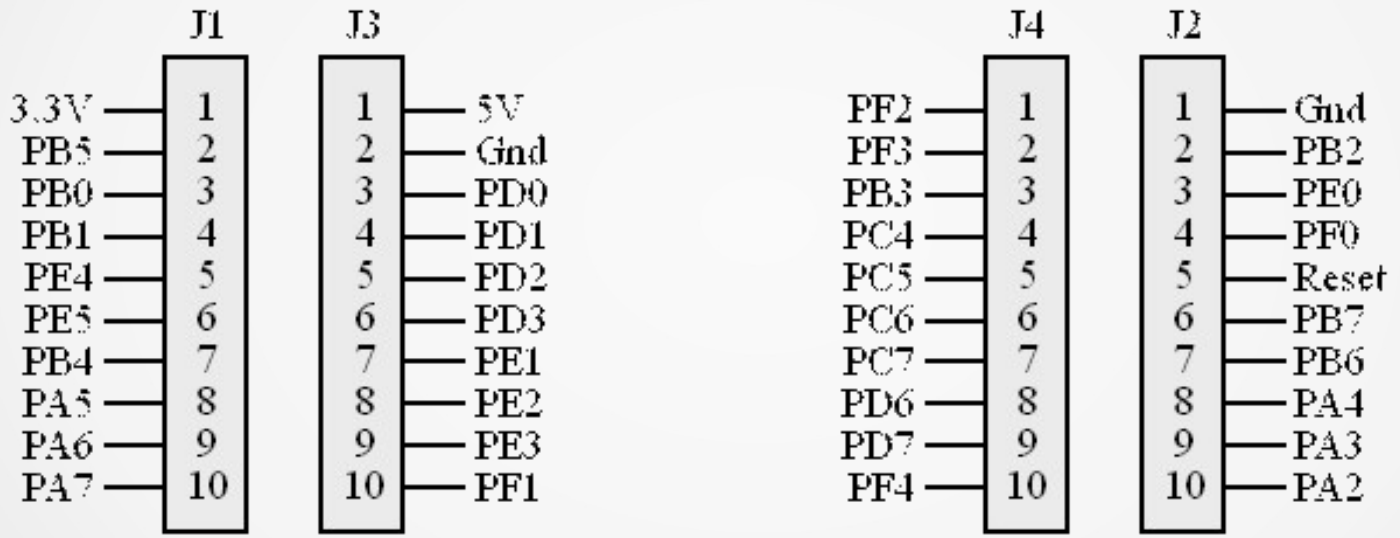


Switch & LED interfaces on LaunchPad



- **PC0-PC3** is used for **JTAG** connections to the debugger on the LaunchPad.
- The LaunchPad connects PB6 to PD0, and PB7 to PD1.
- If you wish to use both PB6 and PD0 you will need to remove the R9 resistor.
- Similarly, to use both PB7 and PD1 remove the R10 resistor.

Interface connectors on the Tiva TM4C123 LaunchPad Board



ICDI & Com Port

In-Circuit Debug Interface (ICDI)

GPIO Pin	Pin Function
PC0	TCK/SWCLK
PC1	TMS/SWDIO
PC2	TDI
PC3	TDO/SWO

Virtual COM Port

GPIO Pin	Pin Function
PA0	U0RX
PA1	U0TX

Switches & LEDs

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

GPIO Pins With Special Considerations

- The table below shows special consideration GPIO pins.
- Most GPIO pins are configured as GPIOs and tri-stated by default.
- Special consideration pins may be programed to a **non-GPIO** function or may have special commit controls out of reset.

GPIO Pins	Default Reset State	GPIOAFSEL	GPIODEN	GPIOPDR	GPIOPUR	GPIOPCTL	GPIOCR
PA[1:0]	UART0	0	0	0	0	0x1	1
PA[5:2]	SSI0	0	0	0	0	0x2	1
PB[3:2]	$I^{21}C0$	0	0	0	0	0x3	1
PC[3:0]	JTAG/SWD	1	1	0	1	0x1	0
PD[7]	GPIO[a]	0	0	0	0	0x0	0
PF[0]	GPIO[a]	0	0	0	0	0x0	0

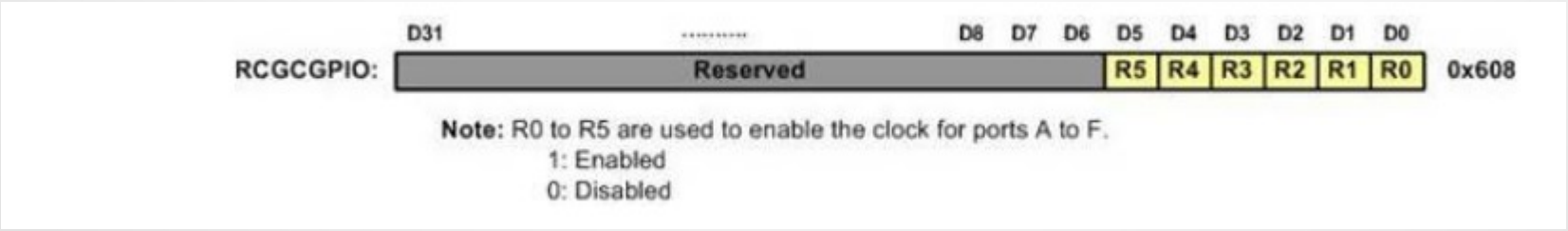
a. This pin is configured as a GPIO by default but is locked and can only be reprogrammed by unlocking the pin in the **GPIOLOCK** register and uncommitting it by setting the **GPIOCR** register.

GPIO Initialization

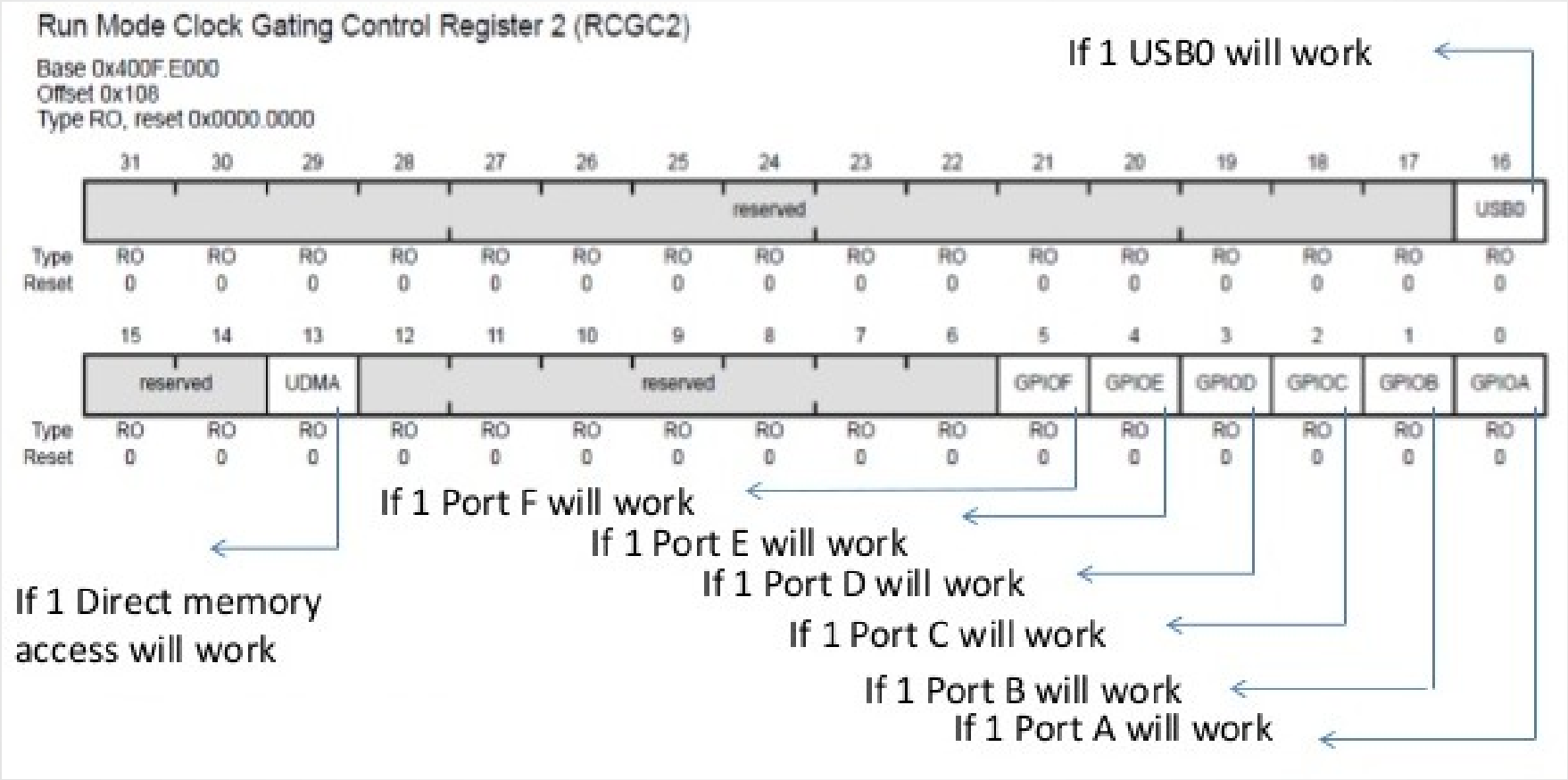
1. Set up the system clock for the specific GPIO Port to enable the clock to drive the port by setting the appropriate bits in the GPIO Run Mode Clock Gating Control (**RCGCG2**) Register.
2. Unlock the port (**GPIOLOCK = 0x4C4F434B**). This step is only needed for pins **PC0-3**, **PD7** and **PF0** on TM4C123GXL LaunchPad.
3. Set up the direction for each pin on the GPIO port by programming the **GPIODIR** register.
4. Optionally you can configure the **GPIOAFSEL** register to program each bit as a **GPIO** Mode or **Alternate** Mode. If an alternate pin is chosen for a bit, then the **PMCx** field must be programmed in the **GPIOPCTL** register for the specific peripheral function.
5. To enable GPIO pins as digital I/Os, set the appropriate **DEN** bit in the **GPIODEN** register. To enable GPIO pins to their analog function (if available), set the **GPIOAMSEL** bit in the **GPIOAMSEL** register.
6. Optionally, you can setup the drive strength for each pin through the GPIODR2R, GPIODR4R, or GPIODR8R registers. This step is optional since the default drive strength is 2 mA.
7. Optionally, you can configure each pad in the port to have pull-up, pull-down, or open drain function through the **GPIOPUR**, **GPIOPDR** , or **GPIOODR** register. Slew rate may also be configured, if needed, through the **GPIOSLR** register.
8. Optionally, you can configure the **GPIOIS**, **GPIOIBE**, **GPIOEV**, and **GPIOIM** registers to set up the type, event, and mask of the interrupts for each port if interrupts are used for the port.
9. Optionally, you can lock the configurations of the NMI and JTAG/SWD pins on the GPIO port pins by setting the LOCK bits in the **GPIOLOCK** register.

GPIO Clock for all I/O ports

- The Run Mode Clock Gating Control (**RCGC2**) register is used to enable the clock source for the I/O port circuitry.
- If an I/O port is not used, the clock source to it can be disabled in order to save power.
- There is only one **RCGC2** special function register for all the ports and each bit of that register is used to enable the clock source to one of the ports.
- In the case of TI TM4C123GH6PM since we have only ports **A** through **F**, the lower 6 bits of this register are used.
- The register bits are shown below.



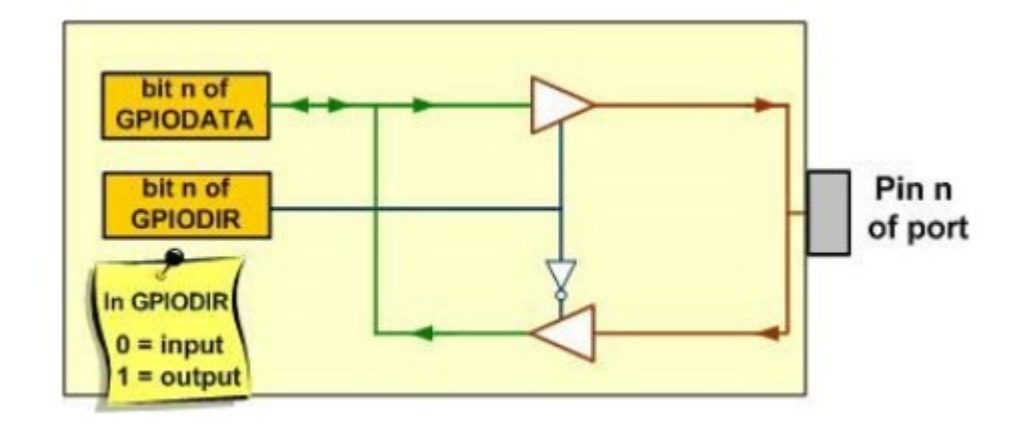
RCGC2 Register



Direction Register

- Generally every microcontroller has minimum of two registers associated with each of I/O port - **Data Register** and **Direction Register**.
- For output we need a latch and for input we need a tri-state buffer. The Direction register is used to make the pin either input or output.
- After the Direction register is properly configured, then we use the Data register to actually write to the pin or read data from the pin.
- It is the Direction register (when configured as output) that allows the information written to the Data register to be driven to the pins of CPU.
- The same way, it is the Direction register (when configured as input) that allows the signal present at the CPU pin to be brought into the Data register.

Direction Register ...



The Data and Direction Registers and a Simplified View of an I/O pin

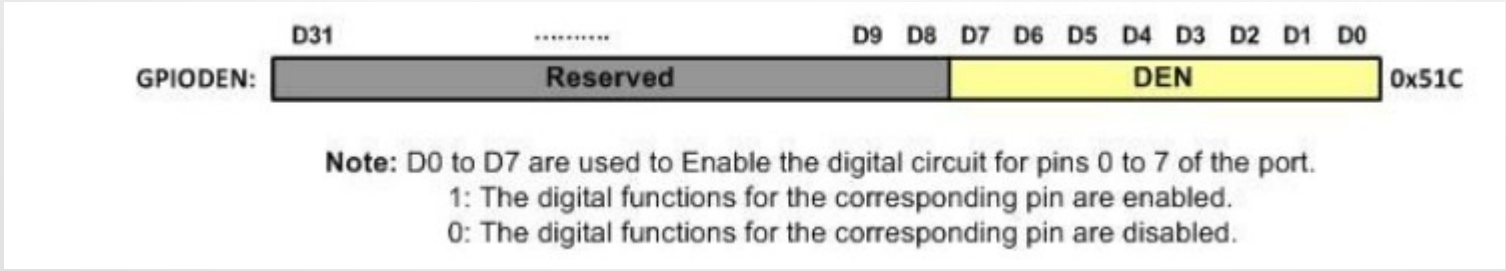
	D31	.....	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	
GPIODIR:	Reserved				DIR								0x400

Note: D0 to D7 are used to set the direction for pins 0 to 7 of the port.
1: Output
0: Input

Each of the Direction register bit needs to be a **0** to configure the port pin as input and a **1** as output.

Digital Enable Register (GPIODEN)

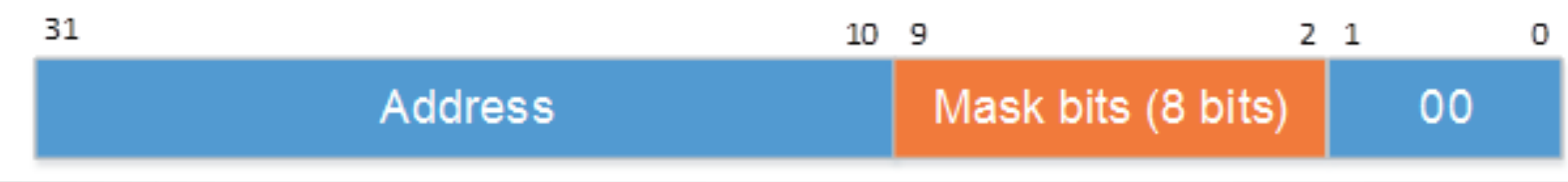
- Each pin of the TI ARM chip can have multiple functions. We choose the function by programming a special function register (**GPIODEN**).
- Using a single pin for multiple functions is called pin multiplexing and is widely used by microcontrollers.
- A given pin can be used as digital I/O, Analog input, or I2C pin. Of course not all at the same time.
- The **GPIODEN** register allows us to enable the pin to be used as digital I/O pin instead of analog function.
- Each PORT of **A** - **F** has its own **GPIODEN** register and one can enable the digital I/O for each pin of a given port.



Note: D0 to D7 are used to Enable the digital circuit for pins 0 to 7 of the port.
1: The digital functions for the corresponding pin are enabled.
0: The digital functions for the corresponding pin are disabled.

DATA Register (GPIODATA)

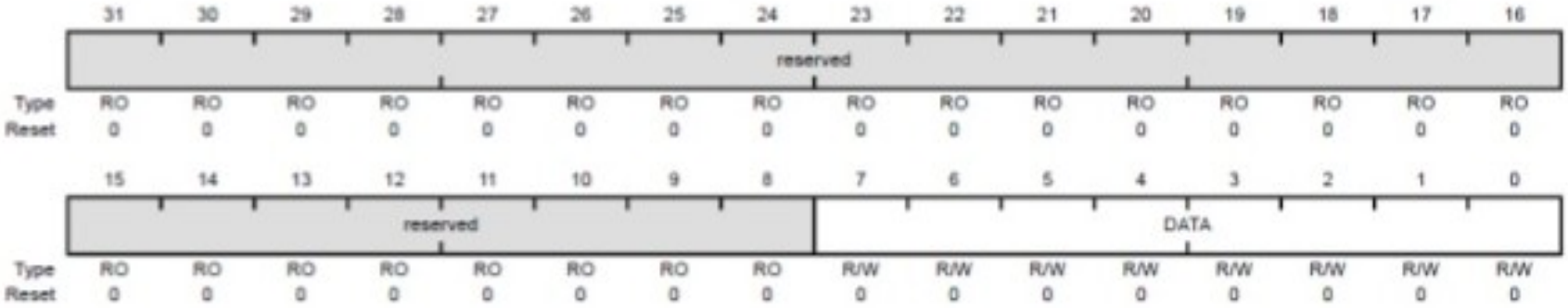
- The GPIO Data Register is located at the offset address of **0x000** from the base address of its port.
- The data register supports **bit-specific** addressing.
- In order to write to this register, the corresponding bits in the mask, resulting from the address bus bits[**9:2**], must be set. Otherwise, the bit values remain unchanged by the write.



- Writing to address 0x40004038 means that bits 1, 2 and 3 of **Port A** must be changed.
- If we want to read and write all 8 bits of a port, it means that we need to sum all these 8 offset constants, which makes the offset address of 0x3FC (001111111100B).
- If we are interested in just bit 5 of Port A, we add 0x0080 to 0x4000.4000, and we can define this in C as
- **#define PA5 (*((volatile uint32_t *)0x40004080))**

GPIO DATA

GPIODATA

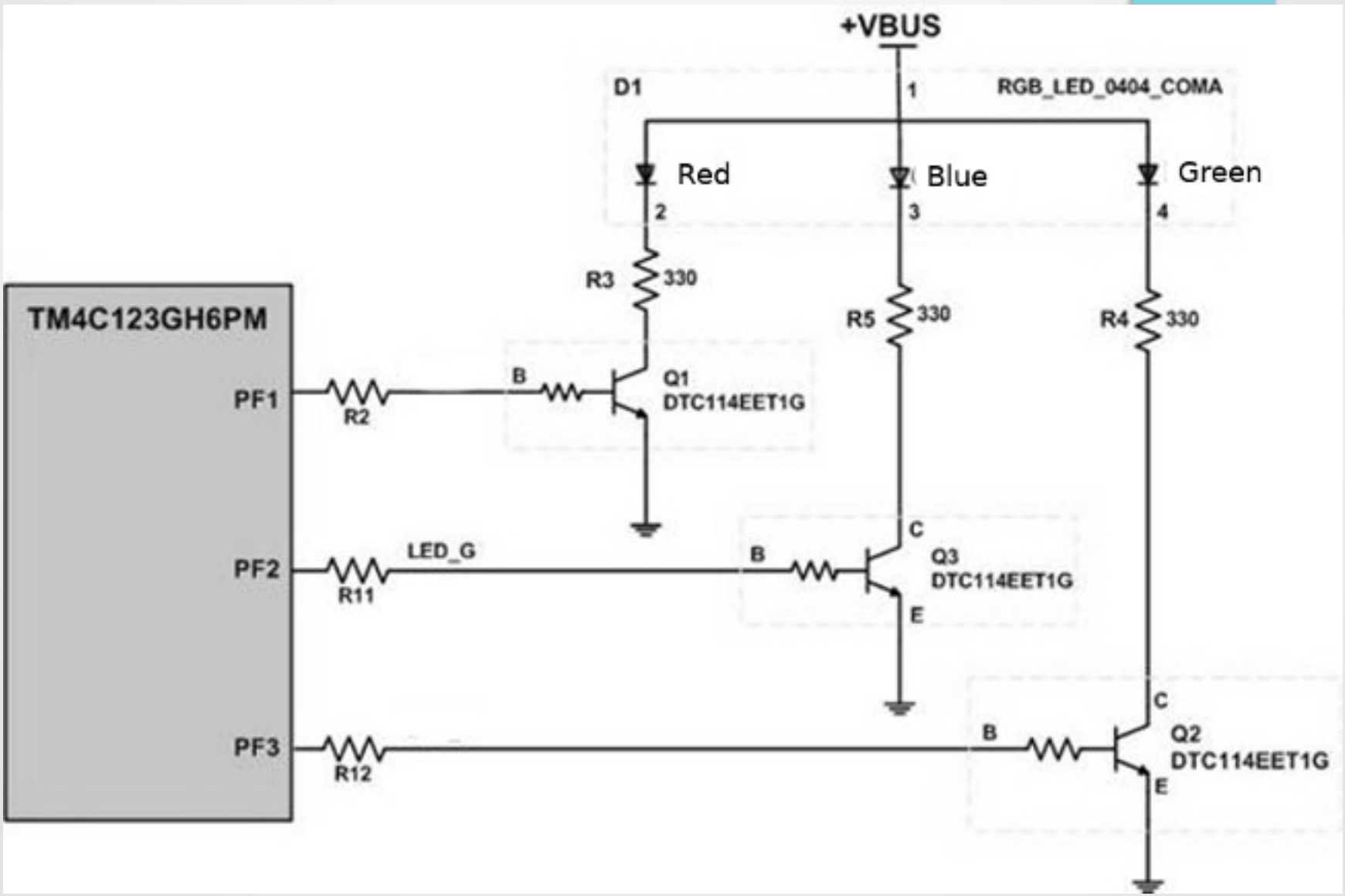


If we wish to access bit	*Constant*
7	0x0200
6	0x0100
5	0x0080
4	0x0040
3	0x0020
2	0x0010
1	0x0008
0	0x0004

ISO C99 integer data types and their ranges

Data Type	Size	Range
int8_t	1 byte	-128 to 127
uint8_t	1 byte	0 to 255
int16_t	2 bytes	-32,768 to 32,767
uint16_t	2 bytes	0 to 65,535
int32_t	4 bytes	0 to 65,535
uint32_t	4 bytes	-2,147,483,648 to 2,147,483,647
int64_t	8 bytes	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
uint64_t	8 bytes	0 to 18,446,744,073,709,551,615

LED connection to PORTF in TI Tiva LaunchPad



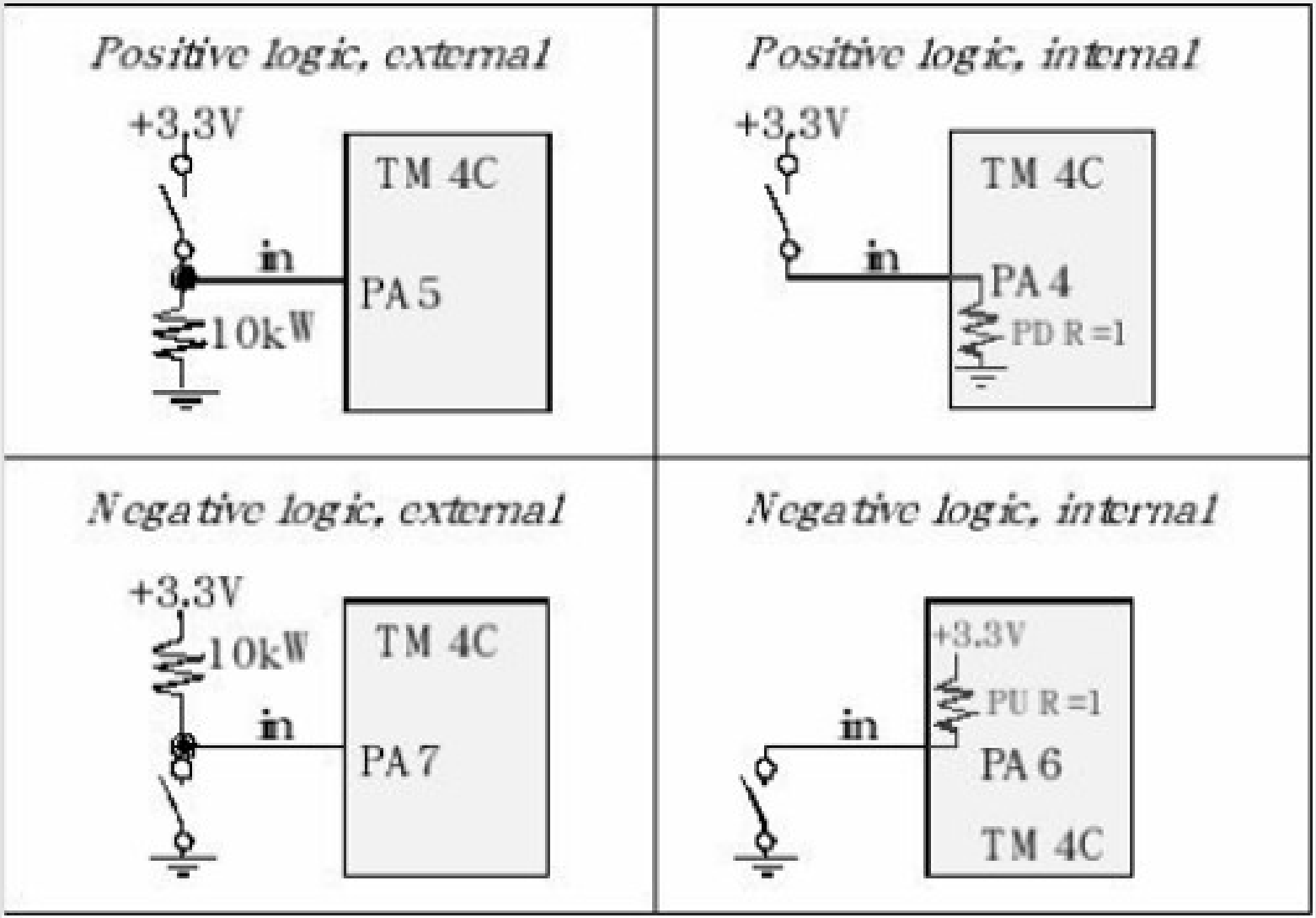
Toggling LEDs

```c
/* Toggling LEDs using special function registers by their names defined in the TivaWare header file */

#include <stdint.h>
#include "inc/tm4c123gh6pm.h"

void delayMs(int n);

int main(void)
{
    SYSCTL_RCGC2_R |= 0x00000020;     /* enable clock to GPIOF at clock gating control register */

    GPIO_PORTF_DIR_R = 0x0E;          /* enable the GPIO pins for the LED (PF3, 2, 1) as output */
    GPIO_PORTF_DEN_R = 0x0E;          /* enable the GPIO pins for digital function */

    while(1) {
        GPIO_PORTF_DATA_R = 0x0E;     /* turn on all LEDs */
        delayMs(500);
        GPIO_PORTF_DATA_R = 0;        /* turn off all LEDs */
        delayMs(500);
    }
}

/* delay n milliseconds (16 MHz CPU clock) */
void delayMs(int n)
{
    int i, j;
    for(i = 0 ; i < n; i++)
        for(j = 0; j < 3180; j++) {}   /* do nothing for 1 ms */
}
```

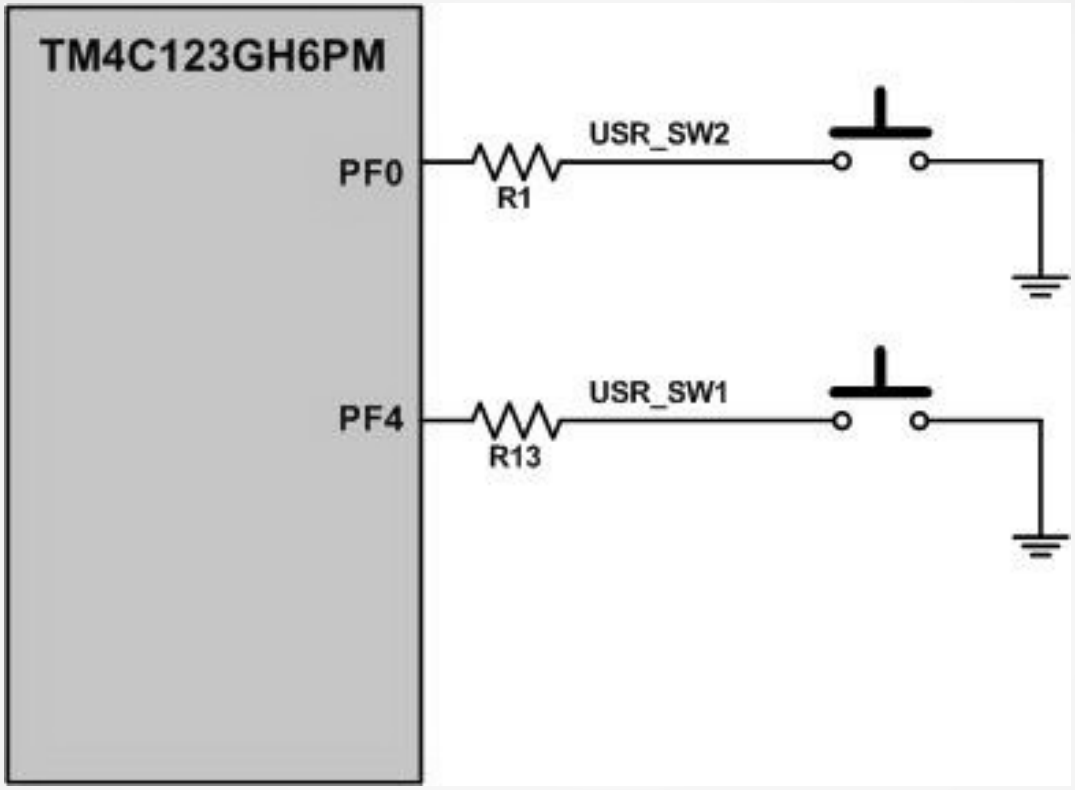
3 – Minute Quiz

- Toggle only one of the LED and shouldn‘t disturb other bits of the PortF
- GPIO PortF DATA REGISTER is at address 0x40025000. The corresponding bits in the mask, resulting from the address bus bits[9:2], must be set in order to read/write.
- Specify a #define that allows us to access bit 2 of Port F. Use this #define to toggle Green LED.

Interfacing a Switch



Switch Inputs and LED Outputs



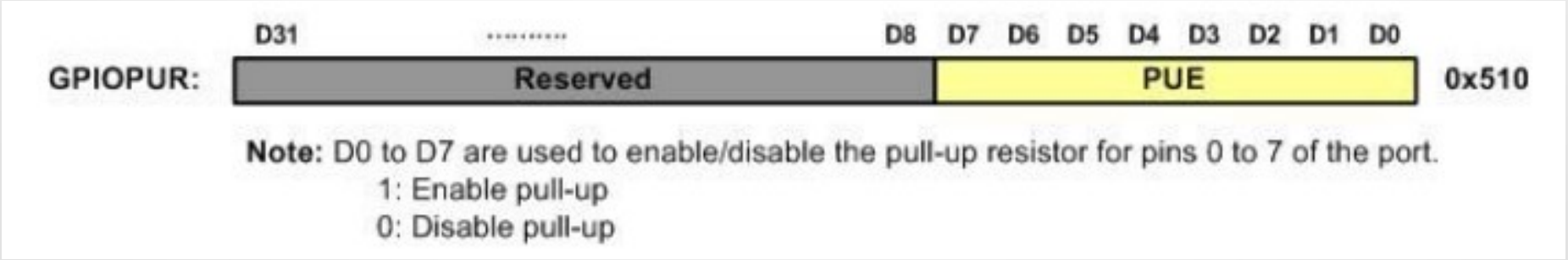
- **SW1** push-button switch is connected directly to PF0 pin
- **SW2** push-button switch is connected directly to PF4 pin.
- There is no pull-up resistor connected to SW1 & SW2
- To use the SW1 and SW2, we need to enable the internal pull-up resistor for PF0 and PF4 pins.

Pull-Up and Pull-Down Resistor

- We can use either positive or negative logic, and we can use an external resistor or select an internal resistor.
- Notice the positive/negative logic circuit with external resistor is essentially the same as the positive/negative logic circuit with internal resistance.
- The difference lies with whether the pull-up/pull-down resistor is connected externally as a 10 kΩ resistor or internally by setting the corresponding GPIOPUR/GPIOPDR bit during software initialization.

GPIO Pull-Up Select (GPIOPUR), GPIO Pull-Down Select (GPIOPDR)

- The **GPIOPUR** register is the pull-up control register.
  - When a bit is set, a weak pull-up resistor on the corresponding GPIO signal is enabled.
  - Setting a bit in GPIOPUR automatically clears the corresponding bit in the GPIO Pull-Down Select (GPIOPDR) register.
- The **GPIOPDR** register is the pull-down control register.
  - When a bit is set, a weak pull-down resistor on the corresponding GPIO signal is enabled.
  - Setting a bit in GPIOPDR automatically clears the corresponding bit in the GPIO Pull-Up Select (GPIOPUR) register.



Read SW1 & Display it on the Green LED

To read SW1 and display it on the green LED, the following steps must be taken.

1. Enable the clock to PortF
2. Set the Direction register PF4 as input, and PF3 as output
3. Enable the digital I/O feature of PortF
4. Enable the pull up resistor option in PUR register since the switch circuit does not have pull-up resistor
5. Read SW1 on PortF
6. Invert the value since the switch is active low and the LED is active high
7. Shift right the switch bit (PF4) to green LED bit(PF3) of the value
8. Write the value to green LED of PortF
9. Repeat steps 5 to 8

Read SW1 and Write it on LED

```
/* Read a switch and write it to the LED */
/* This program reads SW1 of Tiva LaunchPad and writes the inverse of the value to the green
   LED. SW1 is low when pressed (Normally High). LED is ON when high.
*/

#include <stdint.h>
#include "inc/tm4c123gh6pm.h"

int main(void)
{
    unsigned int value;

    SYSCTL_RCGC2_R |= 0x00000020;;   /* enable clock to GPIOF */

    GPIO_PORTF_DIR_R = 0x08;         /* set PORTF3 pin as output (LED) pin */
                                     /* and PORTF4 as input, SW1 is on PORTF4 */
    GPIO_PORTF_DEN_R = 0x18;         /* set PORTF pins 4-3 as digital pins */
    GPIO_PORTF_PUR_R = 0x10;         /* enable pull up for pin 4 */

    While(1) {
        value = GPIO_PORTF_DATA_R;  /* read data from PORTF */
        value = ~value;             /* switch is low active; LED is high active */
        value = value >> 1;         /* shift it right to display on green LED */
        GPIO_PORTF_DATA_R = value;  /* put it on the green LED */
    }
}
```

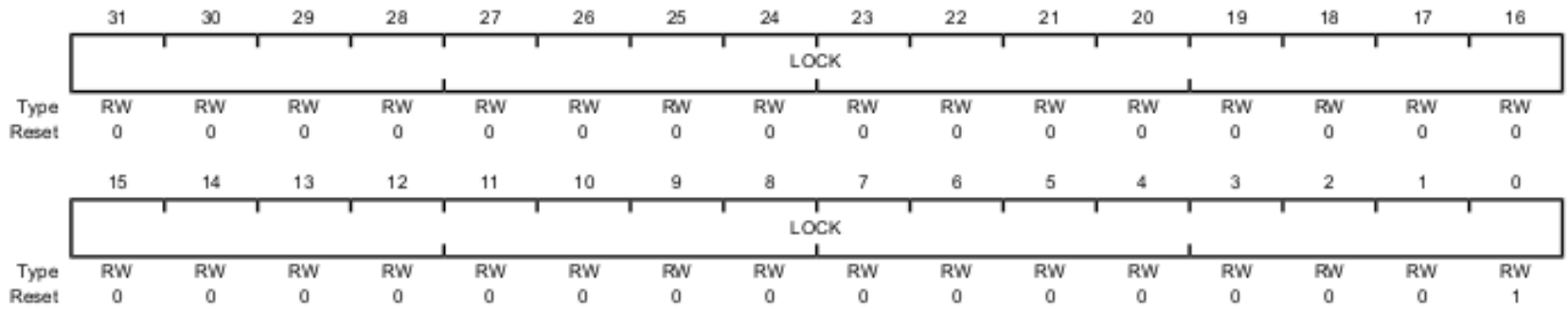
Read SW2 and Write it on Red LED

- To read **SW2** and display it on the red LED, the program is similar to previous except extra steps needed to take care of PortF0 as described below:
- The **SW2** is connected to PortF0 pin, which is shared with **NMI** (non-maskable interrupt).
- To prevent accidental write to configuration registers and thus disables **NMI**, the configuration register bits for PortF0 are normally locked.
- They may be unlocked by writing a passcode value of **0x4C4F434B** to the LOCK Register followed by setting bit 0 of the Commit Register (**GPIOCR**).

GPIOLOCK Register

- The **GPIOLOCK** register enables write access to the GPIO Commit Register (**GPIOCR**).
- Writing **0x4C4F434B** to the GPIOLOCK register unlocks the **GPIOCR** register.
  - Writing any other value to the GPIOLOCK register re-enables the locked state.
  - Reading the **GPIOLOCK** register returns the lock status rather than the 32-bit value that was previously written.
  - Therefore, when write accesses are disabled, or locked, reading the **GPIOLOCK** register returns 0x0000.0001.
  - When write accesses are enabled, or unlocked, reading the **GPIOLOCK** register returns 0x0000.0000

GPIOLOCK Register ...



Bit/Field	Name	Type	Reset	Description
31:0	LOCK	RW	0x0000.0001	GPIO Lock

A write of the value 0x4C4F.434B unlocks the **GPIO Commit (GPIOCR)** register for write access. A write of any other value or a write to the **GPIOCR** register reapplies the lock, preventing any register updates.

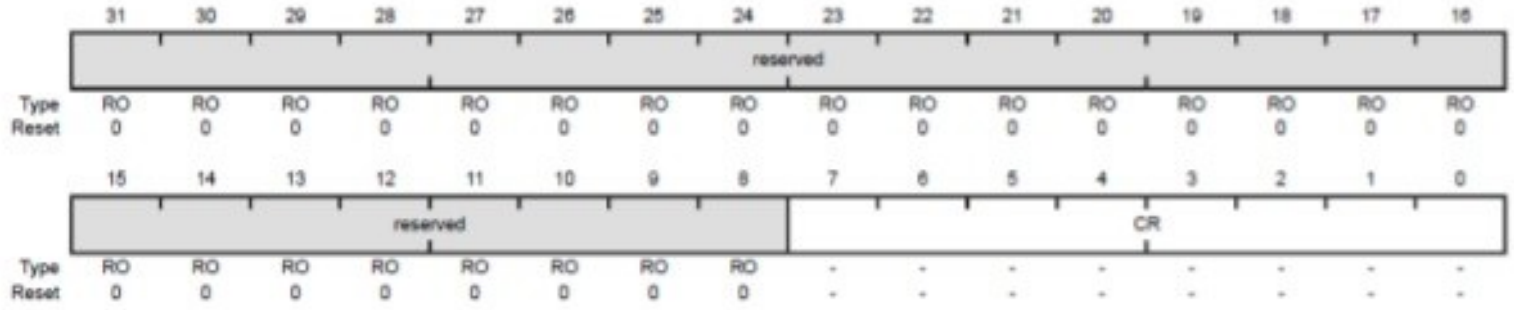
A read of this register returns the following values:

Value	Description
0x1	The **GPIOCR** register is locked and may not be modified.
0x0	The **GPIOCR** register is unlocked and may be modified.

GPIO Commit (GPIOCR) Register

- The value of the **GPIOCR** register determines which bits of the GPIOAFSEL, GPIOPUR, GPIOPDR, and GPIODEN registers are committed when a write to these registers is performed.
- If a bit in the GPIOCR register is cleared, the data being written to the corresponding bit in the GPIOAFSEL, GPIOPUR, GPIOPDR, or GPIODEN registers cannot be committed and retains its previous value.
- 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.

GPIOCR Register



This register is designed **to prevent accidental programming of the registers** that control connectivity to the NMI and JTAG/SWD debug hardware. By initializing the bits of the GPIOCR register to 0 for PD7, PF0, and PC[3:0].

Read SW2 and Write it on LED

```c
/* Read a switch and write it to the LED */
/* This program reads SW2 of Tiva LaunchPad and write the inverse of the value to the red LED.
   SW2 is low when pressed. LED is ON when high. */
/* SW2 is connected to PORTF0, which is an NMI pin. */
/* In order to use this pin for any function other than NMI, the pin needs be unlocked first. */

#include <stdint.h>
#include "inc/tm4c123gh6pm.h"

int main(void)
{
    unsigned int value;

    SYSCTL_RCGC2_R |= 0x00000020;       /* enable clock to GPIOF */

    GPIO_PORTF_LOCK_R = 0x4C4F434B;     /* unlock commit register */
    GPIO_PORTF_CR_R = 0x01;             /* make PORTF0 configurable */
    GPIO_PORTF_DIR_R = 0x02;            /* set PORTF1 pin as output (LED) pin */
                                        /* and PORTF0 as input, SW2 is on  PORTF0 */
    GPIO_PORTF_DEN_R = 0x03;            /* set PORTF pins 1-0 as digital pins */
    GPIO_PORTF_PUR_R = 0x01;            /* enable pull up for pin 0 */

    while(1) {
        value = GPIO_PORTF_DATA_R;      /* read data from PORTF */
        value = ~value;                 /* switch is low active; LED is high active */
        value = value << 1;             /* shift it left to display on red LED */
        GPIO_PORTF_DATA_R = value;      /* put it on red LED */
    }
}
```

GPIO Initialization Rituals

1. Set up the system clock for the specific GPIO Port to enable the clock to drive the port by setting the appropriate bits in the GPIO Run Mode Clock Gating Control (**RCGCG2**) Register.
2. Unlock the port (**GPIOLOCK = 0x4C4F434B**). This step is only needed for pins PC0-3, PD7 and PF0 on TM4C123GXL LaunchPad.
3. Set up the direction for each pin on the GPIO port by programming the **GPIODIR** register.
4. Optionally you can configure the GPIOAFSEL register to program each bit as a GPIO Mode or Alternate Mode. If an alternate pin is chosen for a bit, then the PMCx field must be programmed in the **GPIOPCTL** register for the specific peripheral function.
5. Enable GPIO pins as digital I/Os by setting the appropriate DEN bit in the **GPIODEN** register. To enable GPIO pins to their analog function, set the **GPIOAMSEL** bit in the **GPIOAMSEL** register.
6. Optionally, you can setup the drive strength for each pin through the GPIODR2R, GPIODR4R, or GPIODR8R registers. This step is optional since the default drive strength is 2 mA.
7. Optionally, you can configure each pad in the port to have pull-up, pull-down, or open drain function through the **GPIOPUR**, **GPIOPDR** , or **GPIOODR** register. Slew rate may also be configured, if needed, through the **GPIOSLR** register.
8. Optionally, you can configure the **GPIOIS**, **GPIOIBE**, **GPIOEV**, and **GPIOIM** registers to set up the type, event, and mask of the interrupts for each port if interrupts are used for the port.
9. Optionally, you can lock the configurations of the NMI and JTAG/SWD pins on the GPIO port pins by setting the LOCK bits in the **GPIOLOCK** register.

GPIO Init Example

1. SYSCTL_RCGC2_R |= 0x00000020; // activate clock for Port F
2. GPIO_PORTF_LOCK_R = 0x4C4F434B; // unlock GPIO Port F
3. GPIO_PORTF_CR_R = 0x1F; // allow changes to PF4-0
4. GPIO_PORTF_AMSEL_R = 0x00; // disable analog on PF
5. GPIO_PORTF_PCTL_R = 0x00000000; // PCTL GPIO on PF4-0
6. GPIO_PORTF_DIR_R = 0x0E; // PF4,PF0 in, PF3-1 out
7. GPIO_PORTF_AFSEL_R = 0x00; // disable alternate funct on PF7-0
8. GPIO_PORTF_PUR_R = 0x11; // enable pull-up on PF0 and PF4
9. GPIO_PORTF_DEN_R = 0x1F; // enable digital I/O on PF4-0

5 – Minute Test

- Write a program to light Blue LED when SW1 is pressed, light Green LED when SW2 is pressed, light white LED when both the switches are pressed and light RED LED when none is pressed .

The END

Thank you