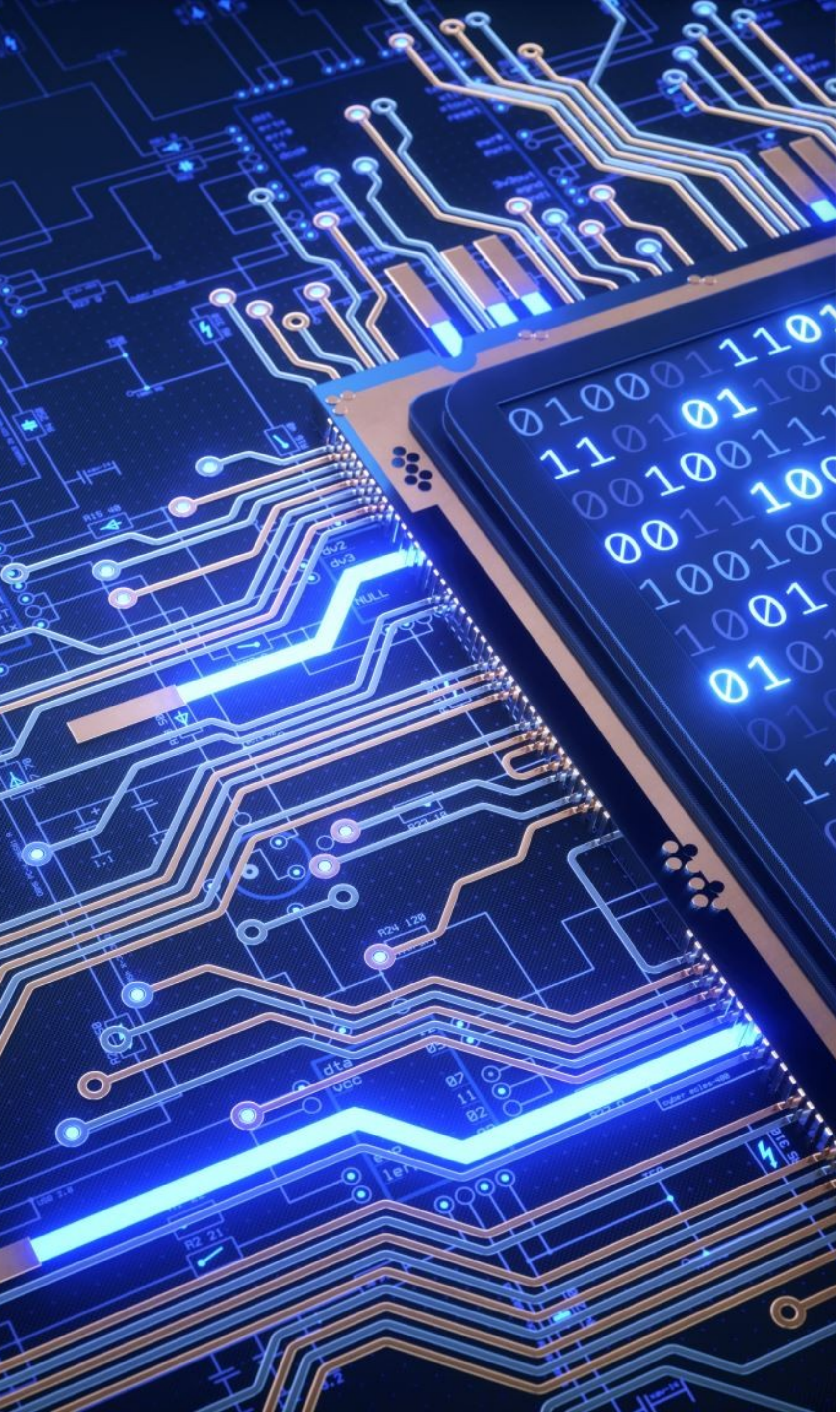
CSULB CECS 346

ARM Cortex M4 TM4C123

GPIO

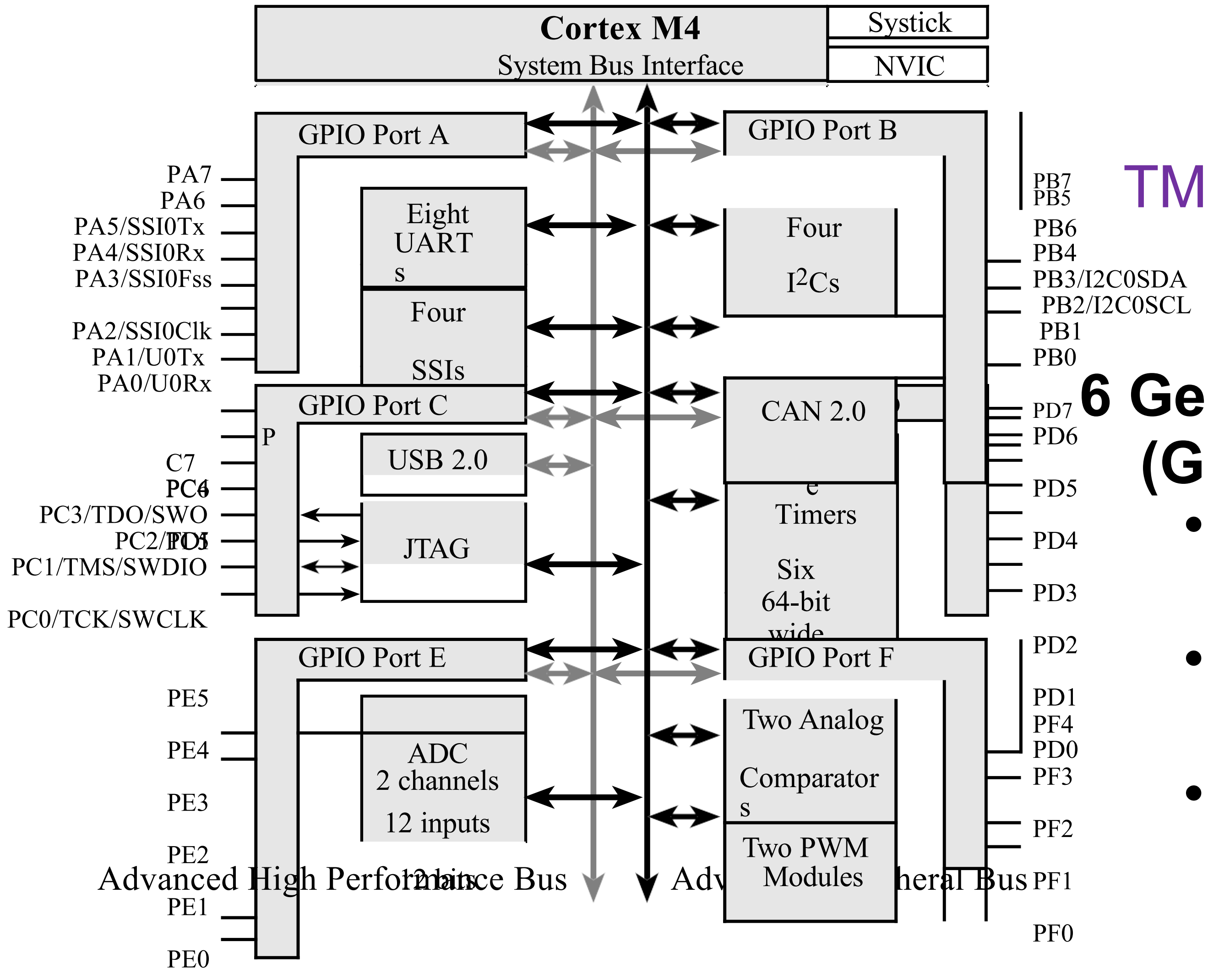
By Min He



Outline

- ❖ GPIO on TM4C123
- ❖ GPIO Pins and Alternative Functions
- ❖ GPIO Registers & Register Addresses
- ❖ GPIO Initialization
- ❖ Interface basic input/output devices: Switches and LEDs
- ❖ Writing friendly code and C review: bit-wise operators and C99 Data Types and Sizes
- ❖ References

General Purpose Input/Output on TM4C123

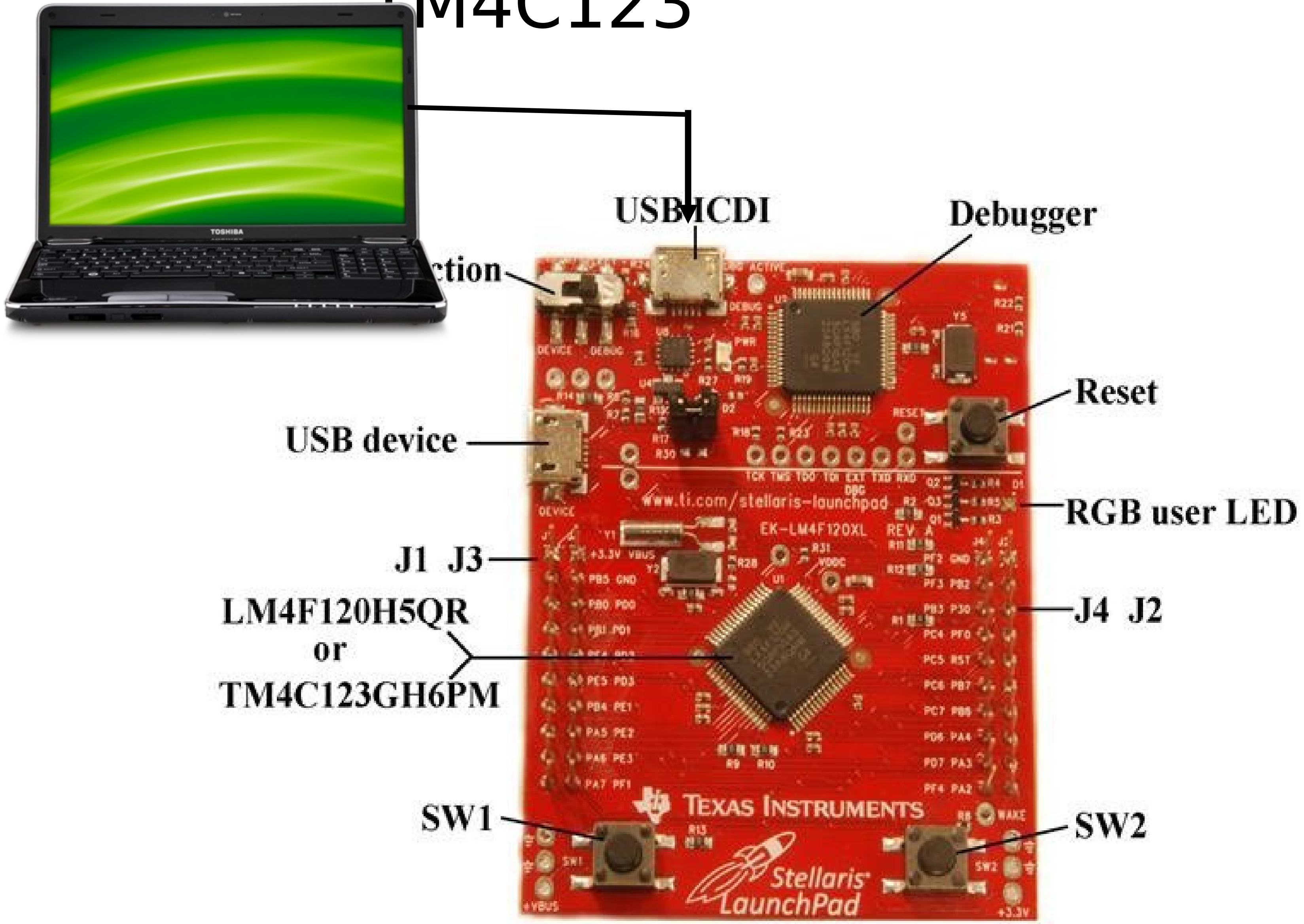


TM4C123GH6PM

6 General-Purpose I/O (GPIO) ports

- Four 8-bit ports (A, B, C, D)
- One 6-bit port (E)
- One 5-bit port (F)

Tiva C TM4C123



GPIO I/O Pins

- Pins can be assigned to as many as 8 different I/O functions: Digital I/O or an alternate function.
- Can be configured for digital I/O, analog input, timer I/O, or serial I/O.
- Two buses used for I/O: AHB & APB: Digital I/O ports are connected to both.
- 8 UART ports, 4 SSI (Synchronized Serial Interface) ports, 4 I^{2}C ports, two 12-bit ADCs, 12 timers, a CAN port, an USB interface, and 16 PWM outputs.
- 43 I/O lines: 4 x 8 + 5 + 6 = 43

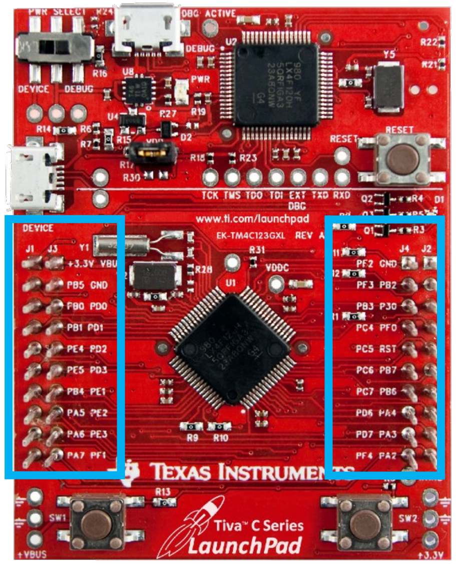
GPIO Pins: Alternative Functions

UART	Universal asynchronous receiver/transmitter. Used to interface to a computer or other devices.
SSI / SPI	Synchronous serial interface / serial peripheral interface. Used to interface eg SD cards, LCD screens. (Medium speed)
I2C	Inter-integrated circuit. (Low speed)
Timer	Periodic interrupts, input capture, output compare. Create periodic interrupts to eg measure period, pulse width, phase and frequency of a signal.
- Periodic Interrupts	Trigger interrupt when timer hits zero without busy waiting.
- Input Capture	Trigger interrupt when rising and/or falling edge is detected.
- Output Compare	Trigger interrupt with timer matches a specified value
PWM	Pulse width modulation. Control eg DC motor speeds.
ADC	Analog to digital converter. Measure the amplitude of analog signals.
Analog Comparator	Compares 2 analog signals and produces a digital output denoting which is greater.
QEI	Quadrature encoder interface. Used for tracking rotation.
USB	Universal serial bus. (High speed)
Ethernet	High speed wired networking to eg intenet or local area network (LAN).
CAN	Controller area network. (High speed)

* Digital alternative functions are specified in PCTL(Port Control Register) Map.

GPIO Connectors

- The LaunchPad has four 10-pin connectors (**J1 J2 J3 J4**) to which you can attach your external signals.
- The top side of these connectors has male pins, and the bottom side has female sockets: supports stacking boards together to make a layered system.
- 8 Pins not accessible: PA0-1, PC0-3, PD4-5.
- Special pins: 3.3v, 5.0v, 2 GND, RESET

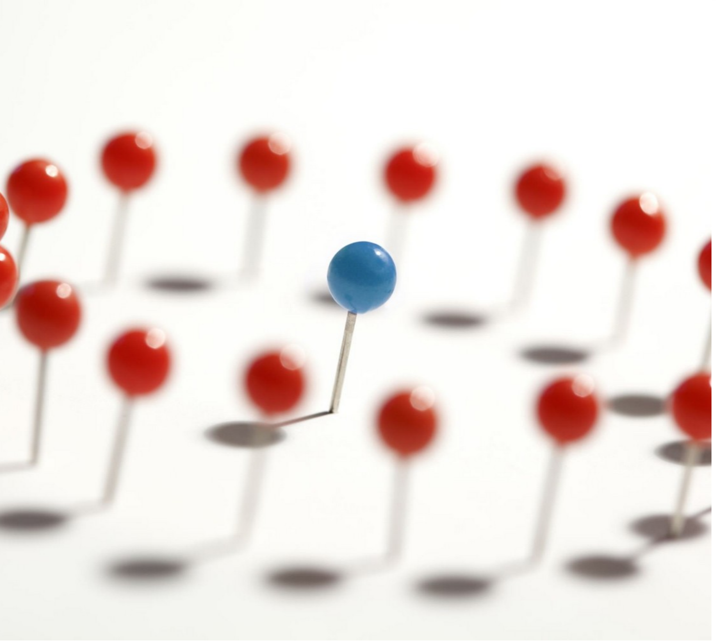


J1 Pin	GPIO
1.01	3.3 V
1.02	PB5
1.03	PB0
1.04	PB1
1.05	PE4
1.06	PE5
1.07	PB4
1.08	PA5
1.09	PA6
1.10	PA7

J3 Pin	GPIO
3.01	5.0 V
3.02	GND
3.03	PD0**
3.04	PD1*
3.05	PD2
3.06	PD3
3.07	PE1
3.08	PE2
3.09	PE3
3.10	PF1

J4 Pin	GPIO
4.01	PF2
4.02	PF3
4.03	PB3
4.04	PC4
4.05	PC5
4.06	PC6
4.07	PC7
4.08	PD6
4.09	PD7
4.10	PF4

J2 Pin	GPIO
2.01	GND
2.02	PB2
2.03	PE0
2.04	PF0
2.05	RESET
2.06	PB7*
2.07	PB6**
2.08	PA4
2.09	PA3
2.10	PA2



Special Pins on The LaunchPad

*From TM4C123 Datasheet [spms376e] Page 650 – 651.

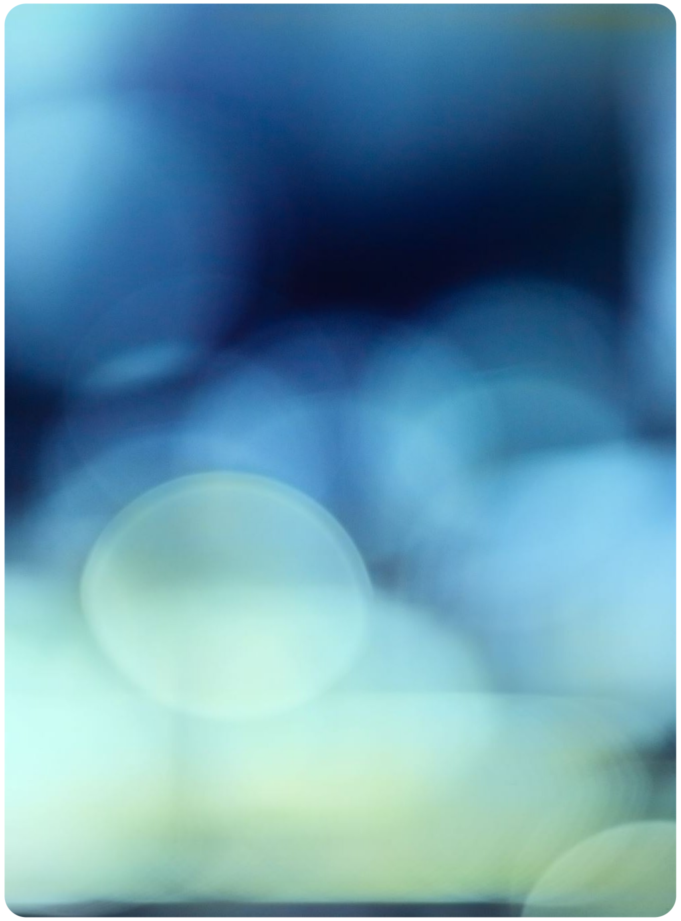
- **PB6/PD0,PB7/PD1**: The LaunchPad connects PB6 to PD0, and PB7 to PD1.
- **PF0-4**: Port F PF4 through PF0 connect to the onboard three LEDs and two Push Buttons. See details on the next slide.
- **PC[3:0]:** used by JTAG. Avoid using these pins when you need to do onboard debug.
- **PF0 & PD7**: needs to be unlocked in **LOCK** register and uncommitted it by setting **CR** register.
- **PA0,1**: they are connected to UART0.
- **PC0-3**: used for JTAG
- **PD4-5**: used for USB0
- **PD4,5/PB0,1**: All GPIO signals are 5V tolerant when configured as inputs except for PD4, PD5, PB0 and PB1, which are limited to 3.6V.

Switches and RGB LEDs on Launchpad

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

* TM4C123G User's Guide [spmu296] Page 9

Programmable Control For GPIO Pad Configuration



* Datasheet Page 1360

- Weak pull-up or pull-down resistor.
- 2 mA, 4 mA, and 8 mA pad drive for digital communication.
- Up to four pads can sink 18 mA for high-current application.
- Open drain enables.
- Digital / Analog input enables.
- The normal voltage range is (3.3V +/- 10%).

Registers

- System Control Registers
- GPIO Registers

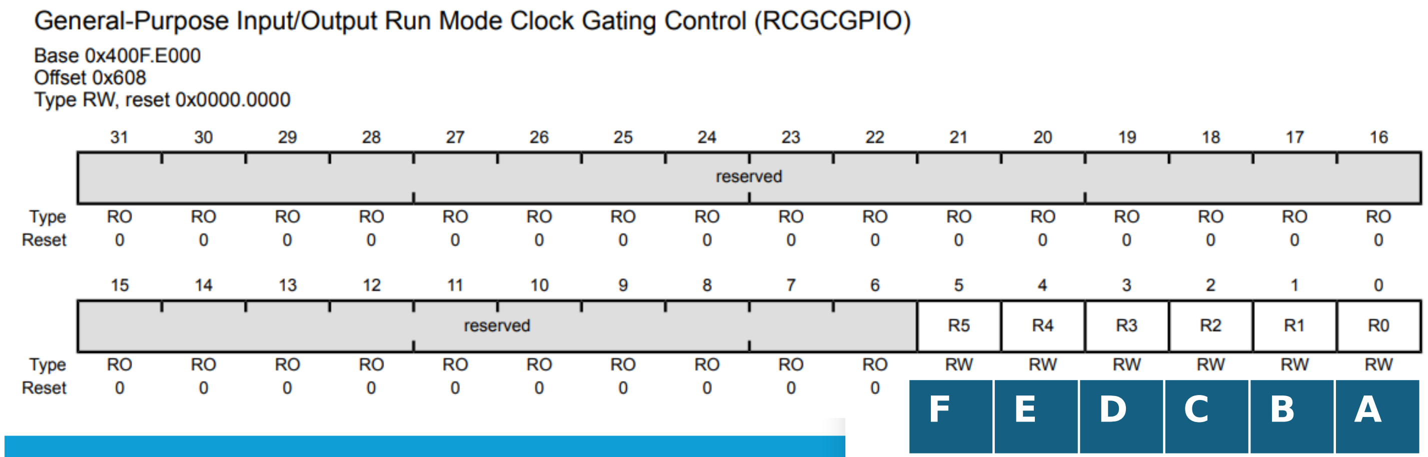
System Control Registers

RCGCGPIO

SYSCTL_RCGC2_R: legacy way
SYSCTL_RCGCGPIO: new way, recommended
Datasheet P. 340

RCGC:
Run mode **C**lock **G**ating **C**ontrol

- Enable clock for GPIO ports during normal operation.
- Common Error: You will get a bus fault if you access a port without enabling its clock.



General-Purpose Input/Output Run Mode Clock Gating Control (RCGCGPIO)
Base 0x400F.E000
Offset 0x608
Type RW, reset 0x0000.0000

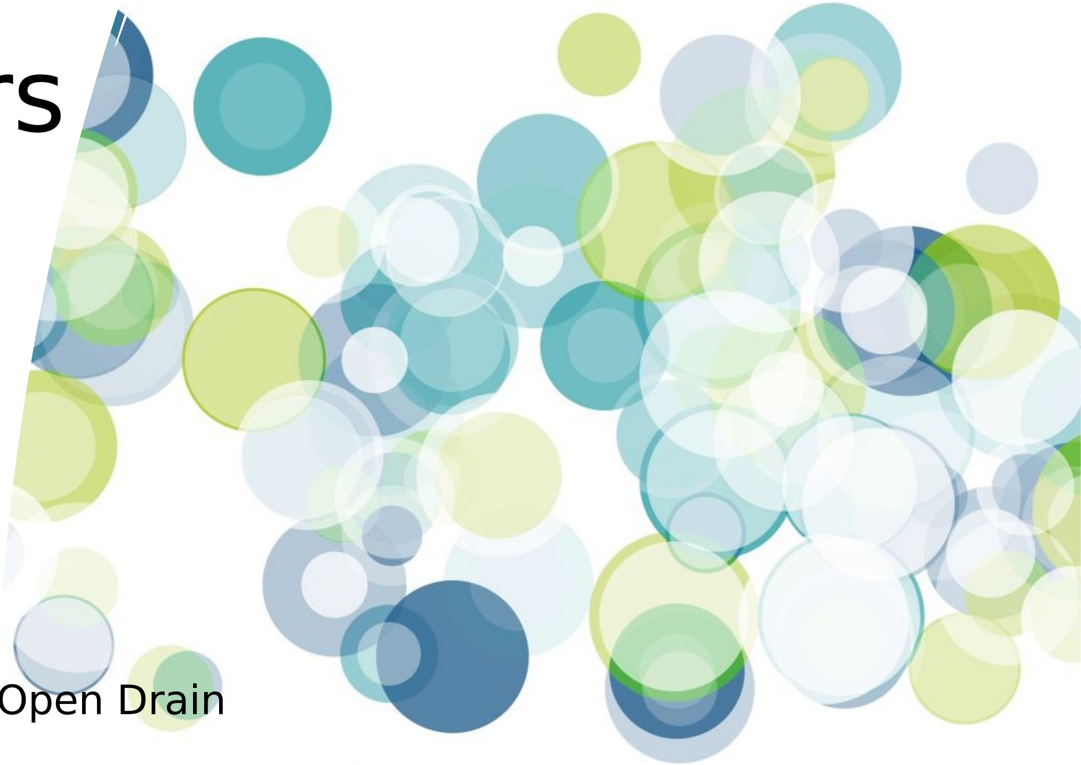
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

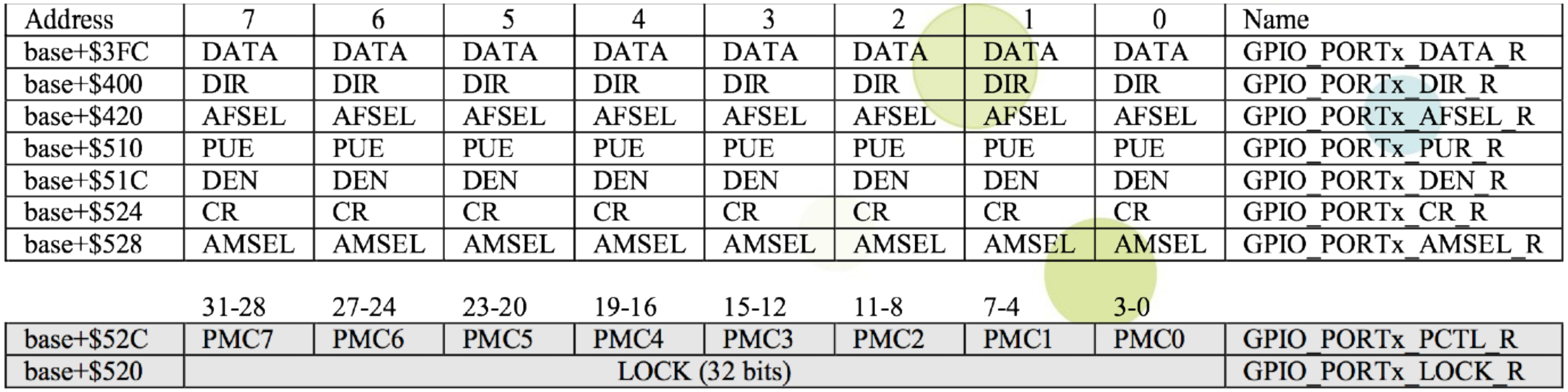
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved										R5	R4	R3	R2	R1	R0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	F	E	D	C	B	A

GPIO Registers

Datasheet: Chapter 10.5

- DEN: Digital Enable
- DIR: Direction
- DATA: data
- AFSEL: alternate function select
- PCTL: Port Control
- LOCK: Lock
- CR: Commit
- R2R/R4R/R8R: Drive Select
- PUR/PDR/ODR: Pull up/Pull Down/ Open Drain





Address	7	6	5	4	3	2	1	0	Name
base+$3FC	DATA	DATA	DATA	DATA	DATA	DATA	DATA	DATA	GPIO_PORTx_DATA_R
base+$400	DIR	DIR	DIR	DIR	DIR	DIR	DIR	DIR	GPIO_PORTx_DIR_R
base+$420	AFSEL	AFSEL	AFSEL	AFSEL	AFSEL	AFSEL	AFSEL	AFSEL	GPIO_PORTx_AFSEL_R
base+$510	PUE	PUE	PUE	PUE	PUE	PUE	PUE	PUE	GPIO_PORTx_PUR_R
base+$51C	DEN	DEN	DEN	DEN	DEN	DEN	DEN	DEN	GPIO_PORTx_DEN_R
base+$524	CR	CR	CR	CR	CR	CR	CR	CR	GPIO_PORTx_CR_R
base+$528	AMSEL	AMSEL	AMSEL	AMSEL	AMSEL	AMSEL	AMSEL	AMSEL	GPIO_PORTx_AMSEL_R

	31-28	27-24	23-20	19-16	15-12	11-8	7-4	3-0	
base+$52C	PMC7	PMC6	PMC5	PMC4	PMC3	PMC2	PMC1	PMC0	GPIO_PORTx_PCTL_R
base+$520	LOCK (32 bits)								GPIO_PORTx_LOCK_R

GPIO Port Base Addresses

Advanced

Peripheral

Bus

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

Advanced **H**igh-perf **B**us

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

See Microcontroller Data Sheet [spms376e] p678

See tm4c123gh6pm.h

GPIO ports are on **APB** by default

Digital Enable Register

GPIODEN / GPIO_PORTx_DEN_R

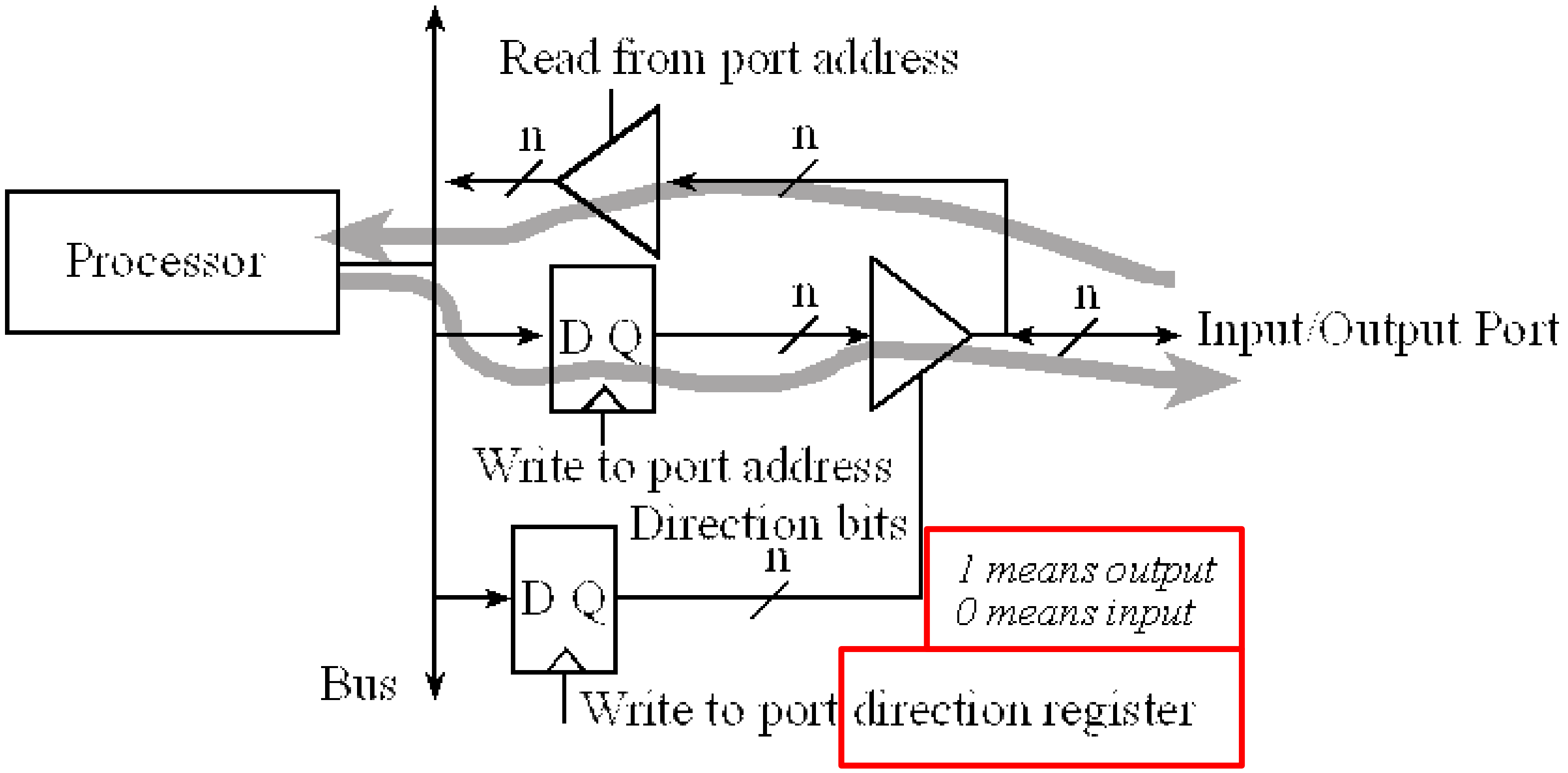
- Enable digital IO on specified GPIO port pins. 0 = disabled, 1 = enabled.
- See Table 10-10 GPIO Pins with Special Considerations for special cases.

Direction Register

GPIODIR/GPIO_PORTx_DIR_R

- Set digital IO direction. Input = 0, Output = 1.
- Only used if GPIODEN = 1 for that pin.

Port Direction Register

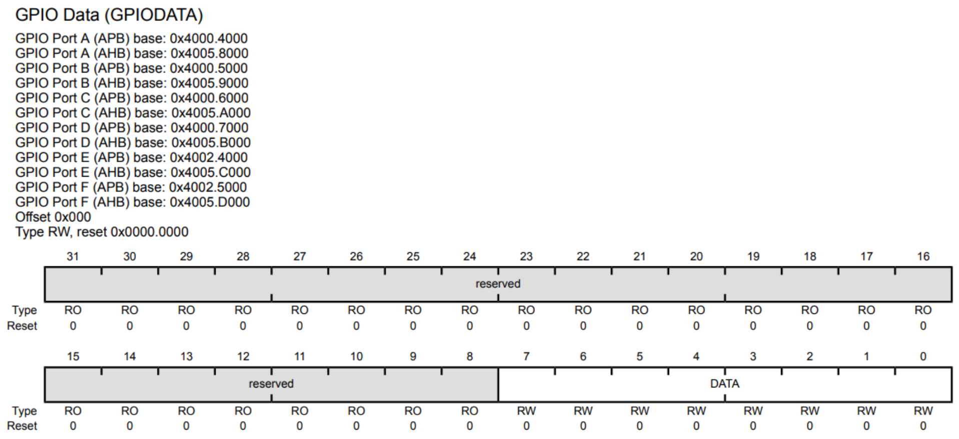


Each digital port pin has a direction bit. This means some pins on a port may be inputs while others are outputs.

Data Register

Datasheet: P.662

- **GPIODATA** / GPIO_PORTx_DATA_R
  - ✓ Used when read/write all 8 bits for this port.
  - ✓ Bits only valid if GPIODEN = 1 for that pin.
- **GPIODATA_BITS** / GPIO_PORTx_DATA_BITS_R
  - ✓ Used for read/write data from this port using **bit-specific addressing**.



GPIO Data (GPIODATA)

GPIO Port A (APB) base: 0x4000.4000
GPIO Port A (AHB) base: 0x4005.8000
GPIO Port B (APB) base: 0x4000.5000
GPIO Port B (AHB) base: 0x4005.9000
GPIO Port C (APB) base: 0x4000.6000
GPIO Port C (AHB) base: 0x4005.A000
GPIO Port D (APB) base: 0x4000.7000
GPIO Port D (AHB) base: 0x4005.B000
GPIO Port E (APB) base: 0x4002.4000
GPIO Port E (AHB) base: 0x4005.C000
GPIO Port F (APB) base: 0x4002.5000
GPIO Port F (AHB) base: 0x4005.D000
Offset 0x000
Type RW, reset 0x0000.0000

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved								DATA							
Type	RO	RO	RO	RO	RO	RO	RO	RO	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Alternate Function Select Register

GPIOAFSEL / GPIO_PORTx_AFSEL_R

- Enable alternate function (instead of digital IO) per pin.

Port Control Register

- **GPIOPCTL** / GPIO_PORTx_PCTL_R
  - Choose which alternate function to use, per pin.
  - Each pin has **4** bits set to specify the alternative function for that pin (0 means regular I/O port)
  - Alternate functions possibilities are different per pin; Function lookup defined in data sheet [spms376e] Table 23-5 or - Textbook Table 4.1.

PCTL Register Setting Map - Textbook Table 4.1

IO	Ain	0	1	2	3	4	5	6	7	8	9	14
PA2		Port		SSI0Clk								
PA3		Port		SSI0Fss								
PA4		Port		SSI0Rx								
PA5		Port		SSI0Tx								
PA6		Port			I$_2$C1SCL		M1PWM2					
PA7		Port			I$_2$C1SDA		M1PWM3					
PB0		Port	U1Rx						T2CCP0			
PB1		Port	U1Tx						T2CCP1			
PB2		Port			I$_2$C0SCL				T3CCP0			
PB3		Port			I$_2$C0SDA				T3CCP1			
PB4	Ain10	Port		SSI2Clk		M0PWM2			T1CCP0	CAN0Rx		
PB5	Ain11	Port		SSI2Fss		M0PWM3			T1CCP1	CAN0Tx		
PB6		Port		SSI2Rx		M0PWM0			T0CCP0			
PB7		Port		SSI2Tx		M0PWM1			T0CCP1			
PC4	C1-	Port	U4Rx	U1Rx		M0PWM6		IDX1	WT0CCP0	U1RTS		
PC5	C1+	Port	U4Tx	U1Tx		M0PWM7		PhA1	WT0CCP1	U1CTS		
PC6	C0+	Port	U3Rx					PhB1	WT1CCP0	USB0epen		
PC7	C0-	Port	U3Tx						WT1CCP1	USB0pflt		
PD0	Ain7	Port	SSI3Clk	SSI1Clk	I$_2$C3SCL	M0PWM6	M1PWM0		WT2CCP0			
PD1	Ain6	Port	SSI3Fss	SSI1Fss	I$_2$C3SDA	M0PWM7	M1PWM1		WT2CCP1			
PD2	Ain5	Port	SSI3Rx	SSI1Rx		M0Fault0			WT3CCP0	USB0epen		
PD3	Ain4	Port	SSI3Tx	SSI1Tx				IDX0	WT3CCP1	USB0pflt		
PD6		Port	U2Rx			M0Fault0		PhA0	WT5CCP0			
PD7		Port	U2Tx					PhB0	WT5CCP1	NMI		
PE0	Ain3	Port	U7Rx									
PE1	Ain2	Port	U7Tx									
PE2	Ain1	Port										
PE3	Ain0	Port										
PE4	Ain9	Port	U5Rx		I$_2$C2SCL	M0PWM4	M1PWM2			CAN0Rx		
PE5	Ain8	Port	U5Tx		I$_2$C2SDA	M0PWM5	M1PWM3			CAN0Tx		
PF0		Port	U1RTS	SSI1Rx	CAN0Rx		M1PWM4	PhA0	T0CCP0	NMI	C0o	
PF1		Port	U1CTS	SSI1Tx			M1PWM5	PhB0	T0CCP1		C1o	TRD1
PF2		Port		SSI1Clk		M0Fault0	M1PWM6		T1CCP0			TRD0
PF3		Port		SSI1Fss	CAN0Tx		M1PWM7		T1CCP1			TRCLK
PF4		Port					M1Fault0	IDX0	T2CCP0	USB0epen		

Lock & Commit Register

- **GPIOLOCK** / GPIO_PORTx_LOCK_R
  - Controls write access to GPIOCR register. Must write value 0x4C4F434B to unlock, any other value locks.
  - Read value 1 = locked, 0 = unlocked.
  - Only PC[3:0], PD7, PF0 are initially locked.
- **GPIOCR** / GPIO_PORTx_CR_R [**C**ommit **R**egister]
  - Determines if writes to bit is allowed in registers GPIOAFSEL, GPIOPUR, GPIOPDR, GPIODEN.
  - 0 = writes not allowed, 1 = writes allowed.
  - Pins must be unlocked in GPIOLOCK register first.
  - Default value is 0xFF for all GPIO ports except for special case pins in data sheet [spms376e] Table 23-1.

Drive Select Registers

- **GPIODR2R** / GPIO_PORTx_DR2R_R [**2** mA **Dr**ive select]
  - 1 = 2 mA drive, 0 = controlled by GPIODR4R or GPIODR8R.
  - Default is 2mA for all pins. Last set of GPIODRxR is what is used.
- **GPIODR4R** / GPIO_PORTx_DR4R_R [**4** mA **Dr**ive select]
  - 1 = 4 mA drive, 0 = controlled by GPIODR2R or GPIODR8R.
- **GPIODR8R** / GPIO_PORTx_DR8R_R [**8** mA **Dr**ive select]
  - 1 = 8 mA drive, 0 = controlled by GPIODR2R or GPIODR4R.

Pull-Up, Pull-Down, Open-Drain Registers

- **GPIOPDR** / GPIO_PORTx_PDR_R [**P**ull-**D**own select]
  - 1 = Enable internal pull down resistor, 0 = disable
- **GPIOPUR** / GPIO_PORTx_PUR_R [**P**ull-**U**p select]
  - 1 = Enable internal pull up resistor, 0 = disable
- **GPIOODR** / GPIO_PORTx_ODR_R [**O**pen **D**rain select]
  - 1 = open drain, 0 = not open drain
  - Open drain applies to output only.
    - 1 = HiZ, 0 = ground (current flows into sink). Max sink current is 18 mA (see data sheet [spms376e] p1360).

GPIO Port Base Addresses

Advanced

Peripheral
Bus

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

Advanced
High-perf
Bus

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

See Microcontroller Data Sheet [spms376e] p678
See tm4c123gh6pm.h

GPIO ports are on **APB** by default

Addresses for GPIO Registers

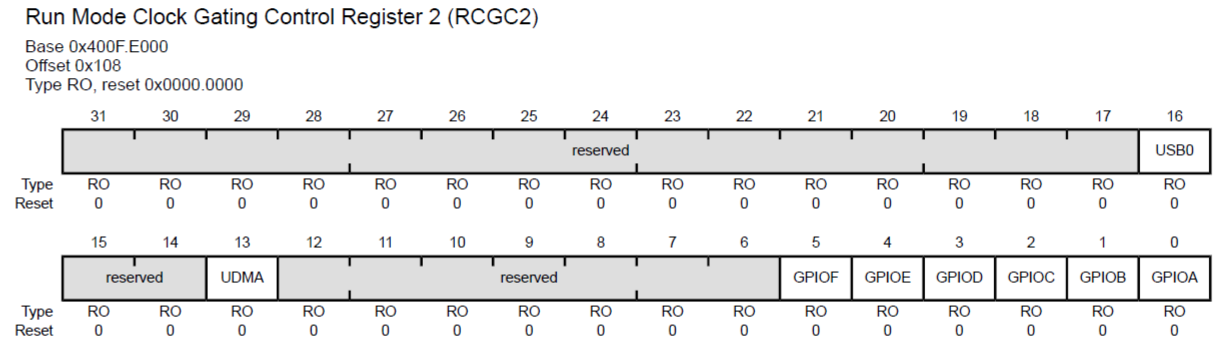
How to Calculate GPIO registers' addresses:
Register address = Port Base address + Register offset.

- **Ex1:** calculate GPIO_PORTF_PCTL_R address
  PCTL offset: 0x52C, see datasheet page 688
  GPIO Port F base address: 0x40025000
  GPIO_PORTF_PCTL_R address:
  0x40025000+0x52C=0x4002552C
- **Ex2:** calculate GPIO_PORTA_DIR_R address
  DIR offset: 0x400, see datasheet page 663
  GPIO Port A base address: 0x40004000
  GPIO_PORTF_PCTL_R address:
  0x40004000+0x400=0x40004400

Finding Address for a Register in Other Modules

1. Find the hardware module in tm4C123 datasheet.
2. In « Register Description » section find the corresponding register.
3. Register address = Base address + Register offset.

Ex: Find the address for SYSCTL_RCGCGPIO_R
Hardware module: system control
Register name: RCGCGPIO, see datasheet Page 340
Base address: 0x400FE000



Run Mode Clock Gating Control Register 2 (RCGC2)

Base 0x400F.E000
Offset 0x108
Type RO, reset 0x0000.0000

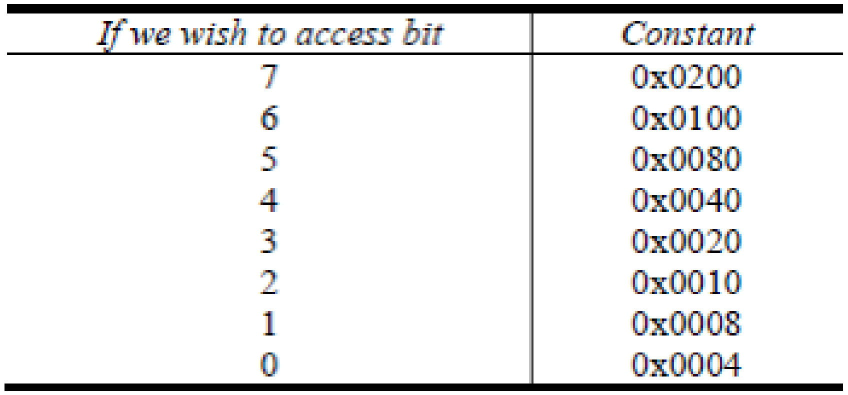
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	reserved															USB0
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	reserved		UDMA	reserved							GPIOF	GPIOE	GPIOD	GPIOC	GPIOB	GPIOA
Type	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

608

Data Register Bit-Specific Addressing

- I/O Port bit-specific addressing is used to access port data register
  - Define address offset as **$4*2^b$**, where **b** is the selected bit position
  - (Alternatively, think of it as $2^b << 2$)
  - 256 possible bit combinations (0-8)
  - Add offsets for each bit selected to base address for the port



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

If port base addr = 0x40025000

Bit specific address for bits 0 and 4
= 0x40025000+0x0004+0x0040
= 0x40025044

If all 8 bits: 0x400253FC

Provides friendly and atomic access to port pins.

Bit-Specific Addressing
Base Address vs. Data Address

- ❑ Calculate Port A Data register address: GPIO_PORTA_DATA_R
  - **0x400.43FC = 0x4000.4000+0x03FC**
  - Allows access to all 8 bits.
- ❑ To access only bit 5:

- **#define PA5 (*((volatile unsigned long *)0x40004080))**

```
PA5 = 0x20;      // make PA5 high
PA5 =            // make PA5 low
0x00;
PA5 = PA5^0x20;  // toggle PA5
```

Port	Base address
PortA	0x40004000
PortB	0x40005000
PortC	0x40006000
PortD	0x40007000
PortE	0x40024000
PortF	0x40025000

***These base addresses are for the APB bus.**

The **volatile** keyword is added because the value of a port can change beyond the direct action of the software. It forces the C compiler to read a new value each time through a loop and not rely on the previous value.

Initialization

- **Initialization (executed once at beginning)**

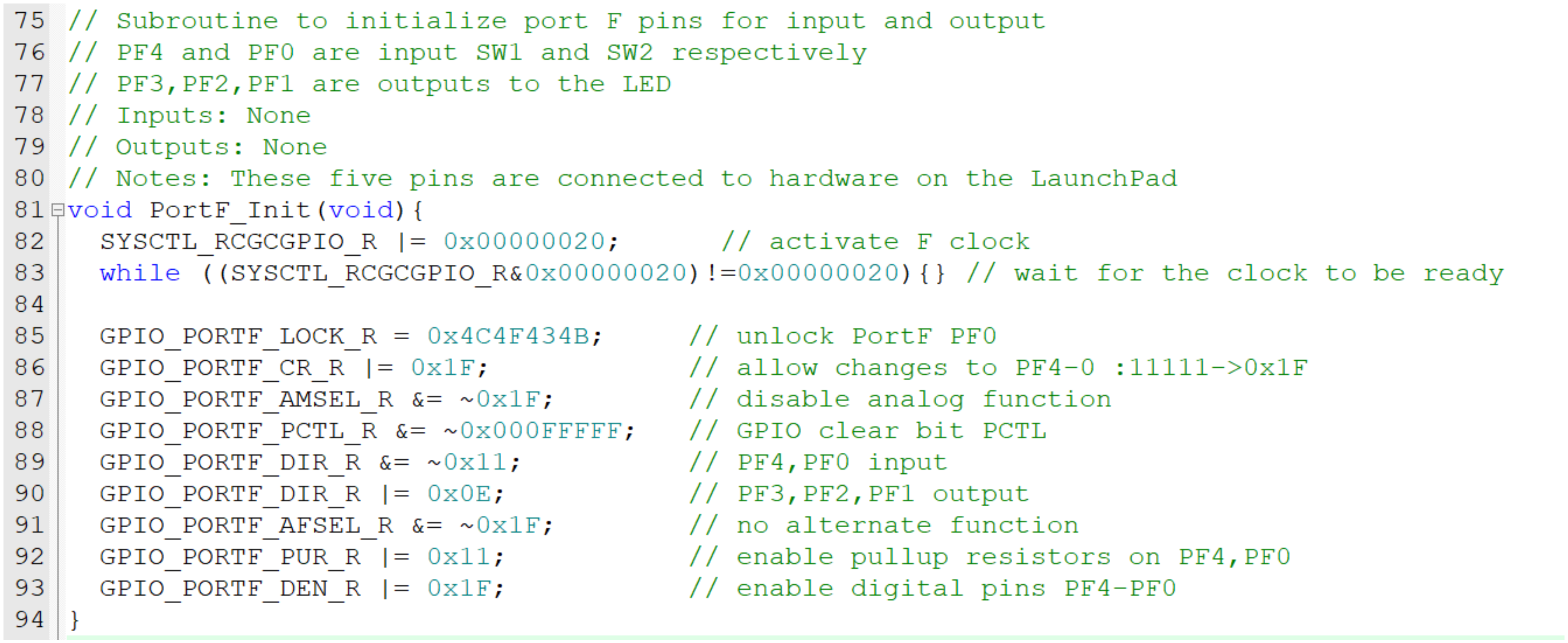
1. Turn on Port clocks in **`RCGCGPIO`**
   Wait two bus cycles (eg two **`NOP`**)
2. Unlock special pins (PF0, PD7, etc) + set CR register for bits used
3. Clear *AMSEL* to disable analog
4. Clear *PCTL* to select GPIO
5. Set *DIR* to 0 for input, 1 for
6. output
7. Clear *AFSEL* bits to 0 to select regular I/O
   Set *PUR/PDR/ODR* bits to 1 to enable internal
   weak pull-up/pull-down resistors or open drain
8. Set *DEN* bits to 1 to enable data pins

Steps 1, 5, 8 are the minimum steps required to enable digital IO

Example: Port F Initialization

`PortF_Init` function in **C** from example project HelloLaunchPad.

- PF4, PF0 are set to be inputs (onboard switches w/ internal pull-up resistors).
- PF3-1 are set to be outputs (onboard RGB LEDs).



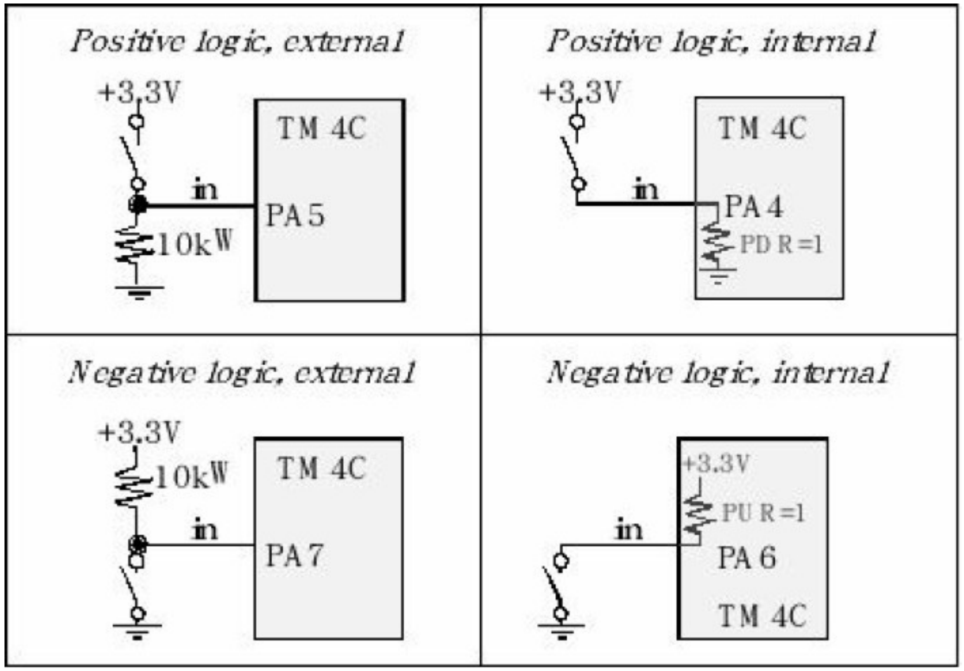
```c
// Subroutine to initialize port F pins for input and output
// PF4 and PF0 are input SW1 and SW2 respectively
// PF3,PF2,PF1 are outputs to the LED
// Inputs: None
// Outputs: None
// Notes: These five pins are connected to hardware on the LaunchPad
void PortF_Init(void){
  SYSCTL_RCGCGPIO_R |= 0x00000020;       // activate F clock
  while ((SYSCTL_RCGCGPIO_R&0x00000020)!=0x00000020){} // wait for the clock to be ready

  GPIO_PORTF_LOCK_R = 0x4C4F434B;     // unlock PortF PF0
  GPIO_PORTF_CR_R |= 0x1F;            // allow changes to PF4-0 :11111->0x1F
  GPIO_PORTF_AMSEL_R &= ~0x1F;        // disable analog function
  GPIO_PORTF_PCTL_R &= ~0x000FFFFF;   // GPIO clear bit PCTL
  GPIO_PORTF_DIR_R &= ~0x11;          // PF4,PF0 input
  GPIO_PORTF_DIR_R |= 0x0E;           // PF3,PF2,PF1 output
  GPIO_PORTF_AFSEL_R &= ~0x1F;        // no alternate function
  GPIO_PORTF_PUR_R |= 0x11;           // enable pullup resistors on PF4,PF0
  GPIO_PORTF_DEN_R |= 0x1F;           // enable digital pins PF4-PF0
}
```

Interface Basic Input/Output Devices

- Switches
- LEDs

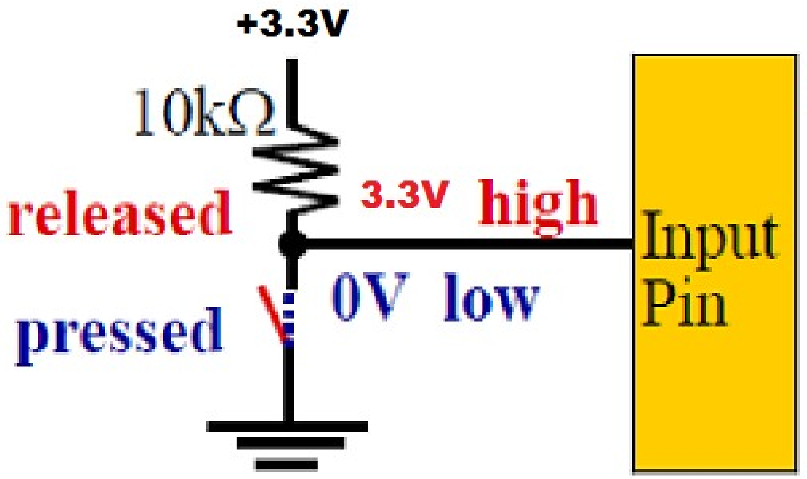
Switch Inputs

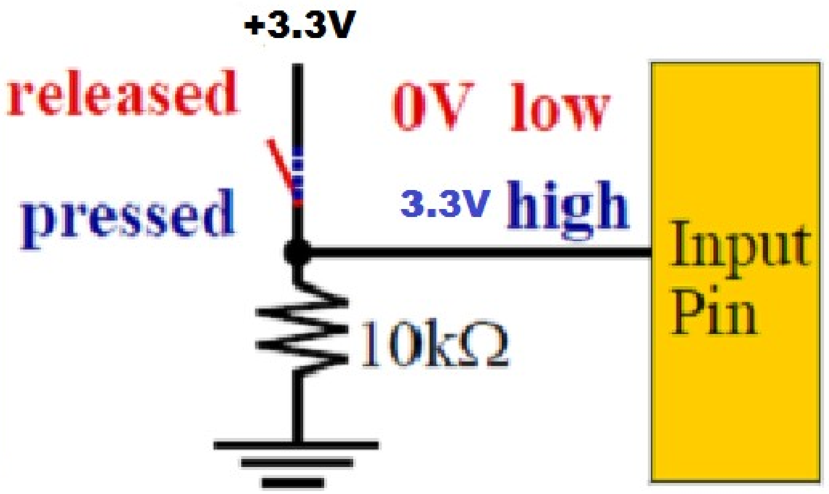
There are four ways to interface a switch to the microcontroller.



- ❑ We can use either positive or negative logic, and
- ❑ We can use an external resistor or select an internal resistor.
- ❑ External resistor is essentially the same as internal resistor;
- ❑ When use external resistor, use a 10 kΩ resistor; when use an internal resistor, set the corresponding PDR/PUR bit during software initialization.

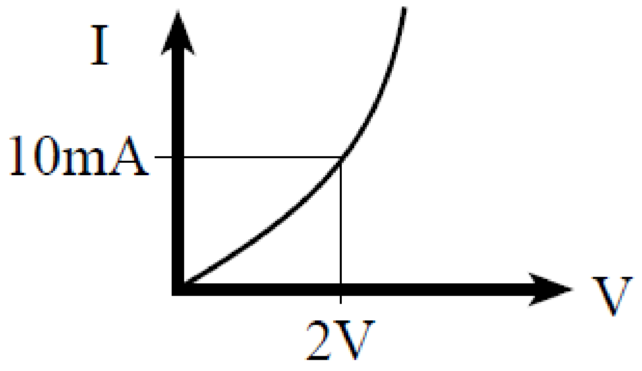
Switch Configuration with External Resisters





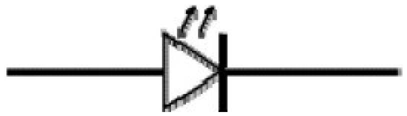
negative – pressed = ‘0’

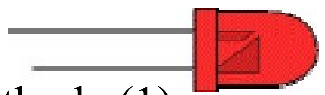
positive – pressed = ‘1’



LED current v. voltage

Brightness = power = V*I



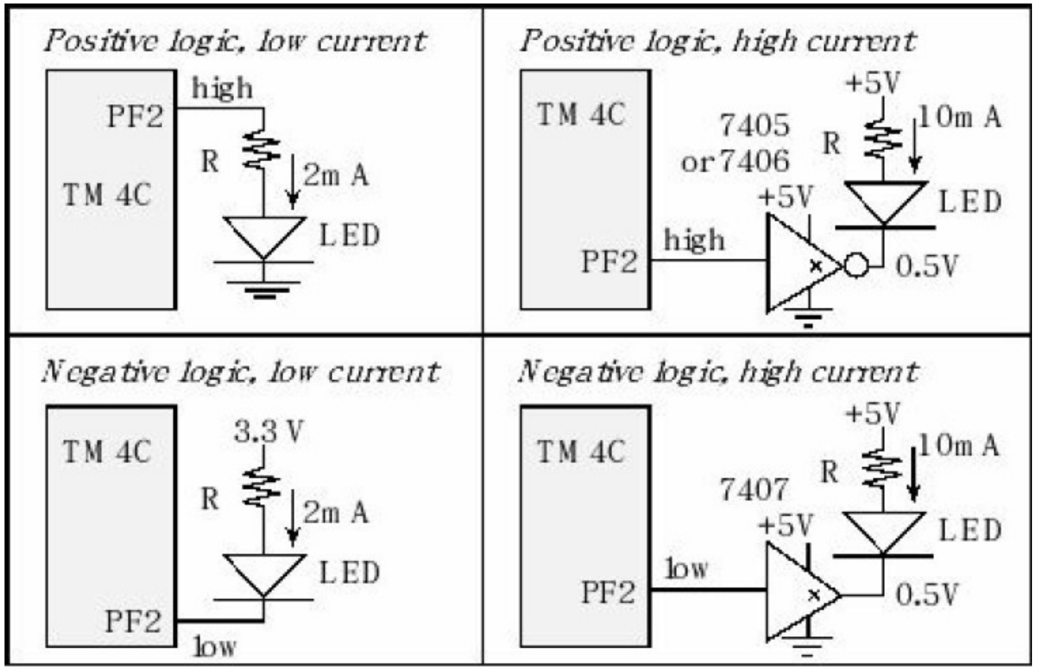


“*big voltage connects to big pin*”

How an LED Works

Interface an LED

There are four ways to interface an LED to the microcontroller.



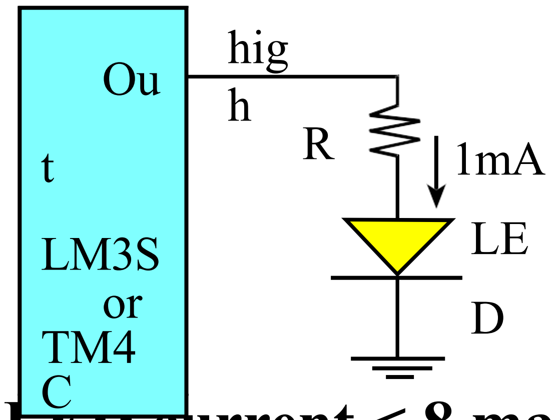
- If the microcontroller can output the current needed by the LED, we can use the two circuits on the left.
- If the LED current is more than can be supplied by our microcontroller, we can use the two circuits on the right .

On the TM4C123, the maximum output current is 8 mA, the default setting for an output pin is 2 mA maximum current. To activate 8 mA mode, we set bits in the **GPIO_PORTF_DR8R_R** register.

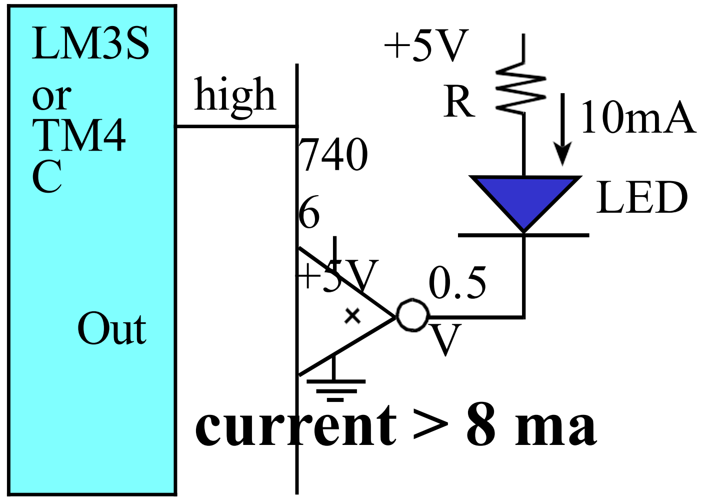
Calculate Resistor Values for an LED

R = (3.3V – 1.6)/0.001
= 1.7 kOhm

R = (5.0-2-0.5)/0.01
= 250 Ohm



LED current < 8 ma

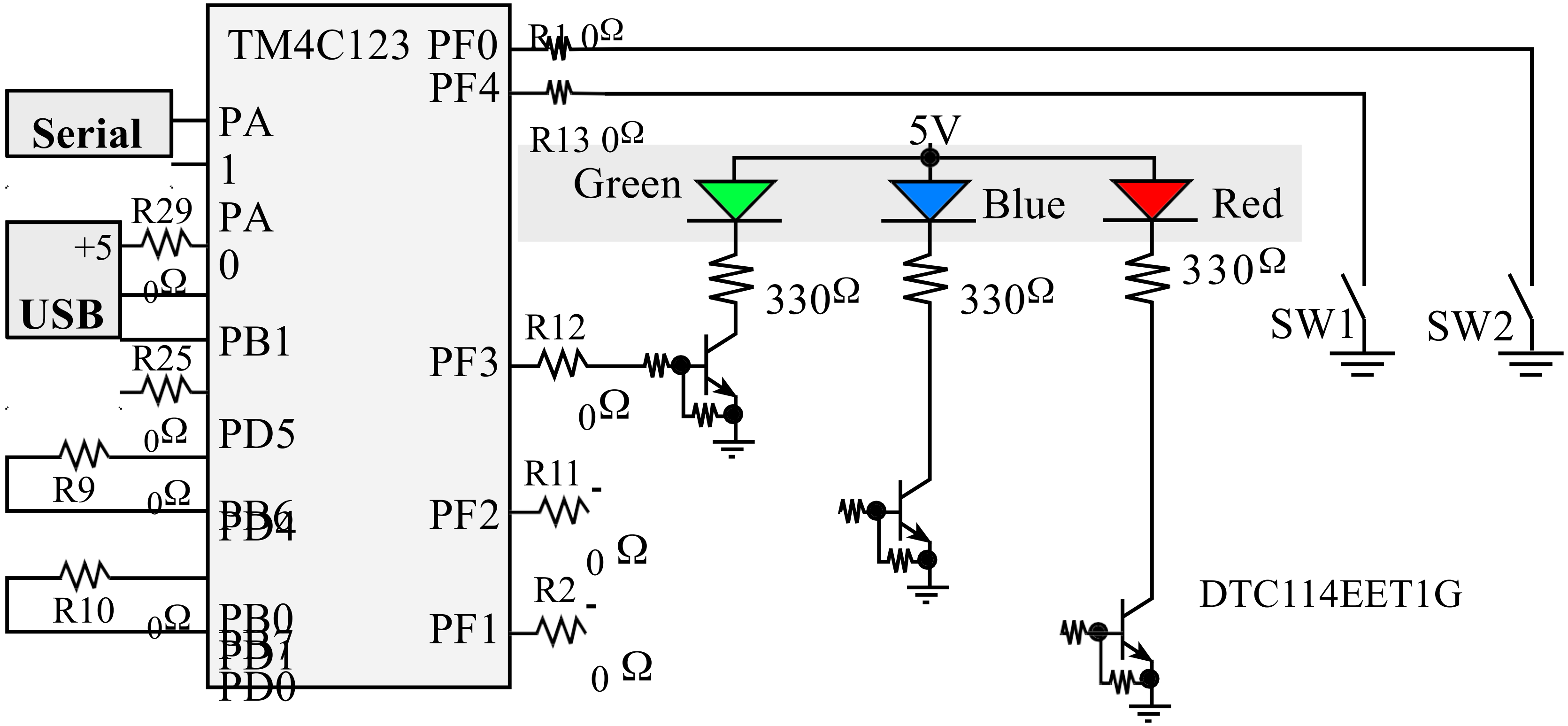


current > 8 ma

Example LED datasheet

Page 4: V_f=1.7v, test condition: 2mA

LaunchPad Switches and LEDs



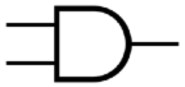
- The **switches** on the LaunchPad
  - **Negative logic**
  - Require internal pull-up (set bits in PUR)
- The PF3-1 **LEDs** are **positive logic**

Writing Friendly Code

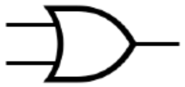
Two ways to write friendly code:

- **read-modify-write** instructions using bit-wise operators
- Use **Bit-specific Addressing**.
  The TM4C family implements a flexible way to access port pins. This bit-specific addressing doesn't work for all the I/O registers, just the parallel port **data** registers.

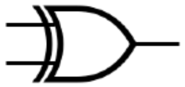
Review C: Bitwise Boolean Operations



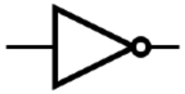
AND		
A	B	A & B
0	0	0
0	1	0
1	0	0
1	1	1



OR		
A	B	A | B
0	0	0
0	1	1
1	0	1
1	1	1



EOR		
A	B	A ^ B
0	0	0
0	1	1
1	0	1
1	1	0



NOT	
A	~A
0	1
1	0

Review C: Set, Clear and Toggle Bits

- Assume c =

b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0

- Clear bit 5
  - c &= ~0x20

b_7	b_6	0	b_4	b_3	b_2	b_1	b_0

  - Note: c &= 0xDF is equivalent to above but less obvious which bit is being modified
- Set bit 2
  - c |= 0x04

b_7	b_6	b_5	b_4	b_3	1	b_1	b_0

- Toggle bit 1
  - c ^= 0x02

b_7	b_6	b_5	b_4	b_3	b_2	$\sim b_1$	b_0

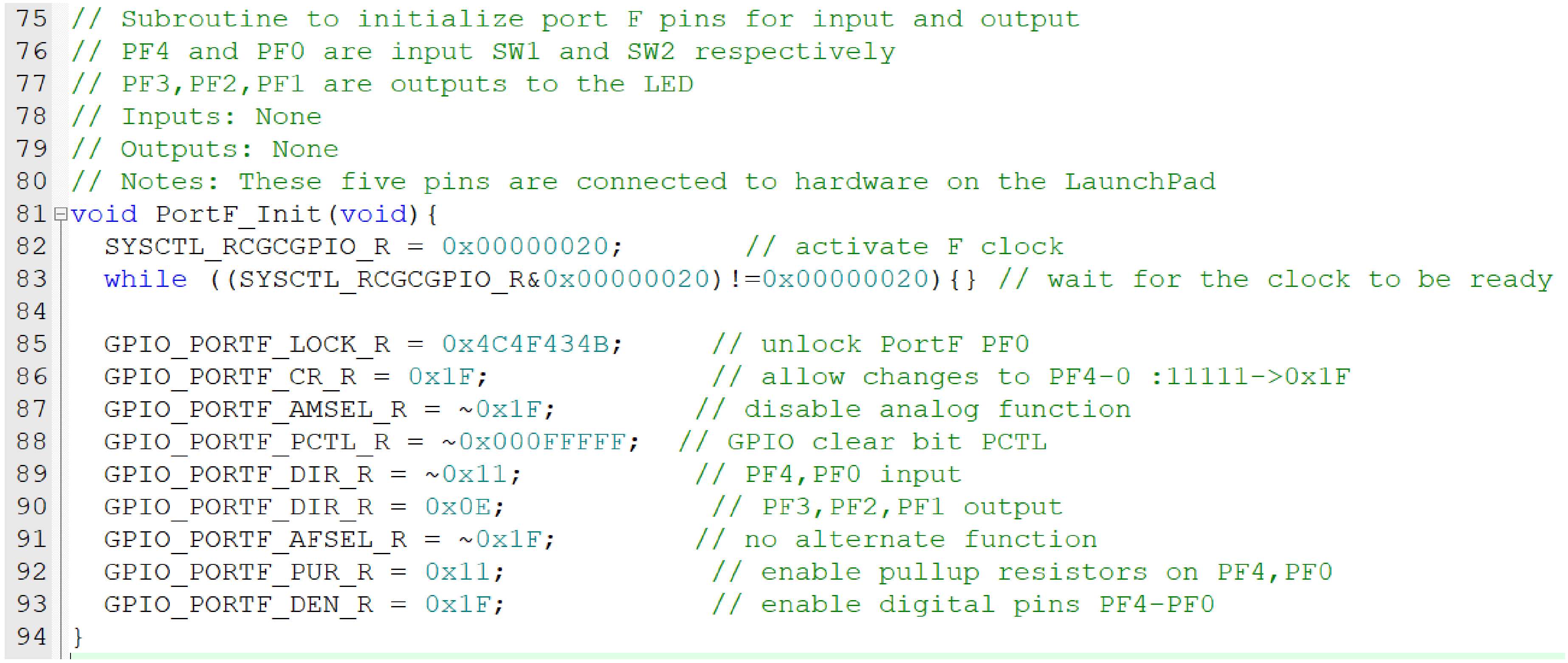
Review C: C99 Data Type Sizes

Name	Size (Bits)	C99 Type
char	8	int8_t
unsigned char	8	uint8_t
short	16	int16_t
unsigned short	16	uint16_t
int	16	int16_t
unsigned int	16	uint16_t
long	32	int32_t
unsigned long	32	uint32_t

Name	Size (Bits)
Nibble	4
Byte	8
Half Word	16
Word	32

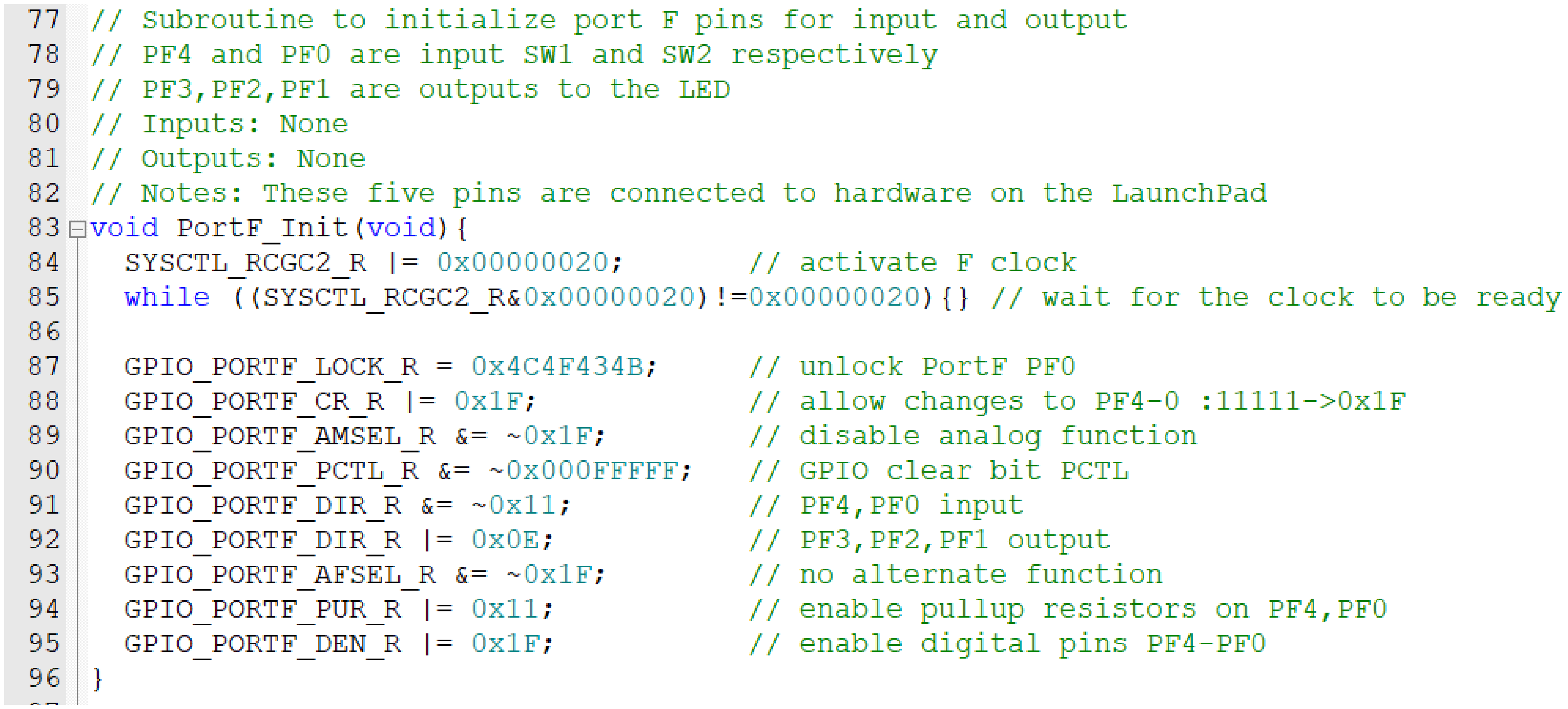
- ❑ Recommend to use C99 types in the code you write for clarity
- ❑ Types (especially int, long, word, half word) depend on processor + platform: These values are for TM4C
- ❑ To use C99 integer data type alias, #include "stdint.h"

Example: non-Friendly Code



```c
// Subroutine to initialize port F pins for input and output
// PF4 and PF0 are input SW1 and SW2 respectively
// PF3,PF2,PF1 are outputs to the LED
// Inputs: None
// Outputs: None
// Notes: These five pins are connected to hardware on the LaunchPad
void PortF_Init(void){
  SYSCTL_RCGCGPIO_R = 0x00000020;       // activate F clock
  while ((SYSCTL_RCGCGPIO_R&0x00000020)!=0x00000020){} // wait for the clock to be ready

  GPIO_PORTF_LOCK_R = 0x4C4F434B;     // unlock PortF PF0
  GPIO_PORTF_CR_R = 0x1F;             // allow changes to PF4-0 :11111->0x1F
  GPIO_PORTF_AMSEL_R = ~0x1F;        // disable analog function
  GPIO_PORTF_PCTL_R = ~0x000FFFFF;  // GPIO clear bit PCTL
  GPIO_PORTF_DIR_R = ~0x11;          // PF4,PF0 input
  GPIO_PORTF_DIR_R = 0x0E;            // PF3,PF2,PF1 output
  GPIO_PORTF_AFSEL_R = ~0x1F;        // no alternate function
  GPIO_PORTF_PUR_R = 0x11;            // enable pullup resistors on PF4,PF0
  GPIO_PORTF_DEN_R = 0x1F;            // enable digital pins PF4-PF0
}
```

Example: Friendly Code with Bit-Wise Operators



```c
// Subroutine to initialize port F pins for input and output
// PF4 and PF0 are input SW1 and SW2 respectively
// PF3,PF2,PF1 are outputs to the LED
// Inputs: None
// Outputs: None
// Notes: These five pins are connected to hardware on the LaunchPad
void PortF_Init(void){
  SYSCTL_RCGC2_R |= 0x00000020;        // activate F clock
  while ((SYSCTL_RCGC2_R&0x00000020)!=0x00000020){} // wait for the clock to be ready

  GPIO_PORTF_LOCK_R = 0x4C4F434B;      // unlock PortF PF0
  GPIO_PORTF_CR_R |= 0x1F;             // allow changes to PF4-0 :11111->0x1F
  GPIO_PORTF_AMSEL_R &= ~0x1F;         // disable analog function
  GPIO_PORTF_PCTL_R &= ~0x000FFFFF;    // GPIO clear bit PCTL
  GPIO_PORTF_DIR_R &= ~0x11;           // PF4,PF0 input
  GPIO_PORTF_DIR_R |= 0x0E;            // PF3,PF2,PF1 output
  GPIO_PORTF_AFSEL_R &= ~0x1F;         // no alternate function
  GPIO_PORTF_PUR_R |= 0x11;            // enable pullup resistors on PF4,PF0
  GPIO_PORTF_DEN_R |= 0x1F;            // enable digital pins PF4-PF0
}
```

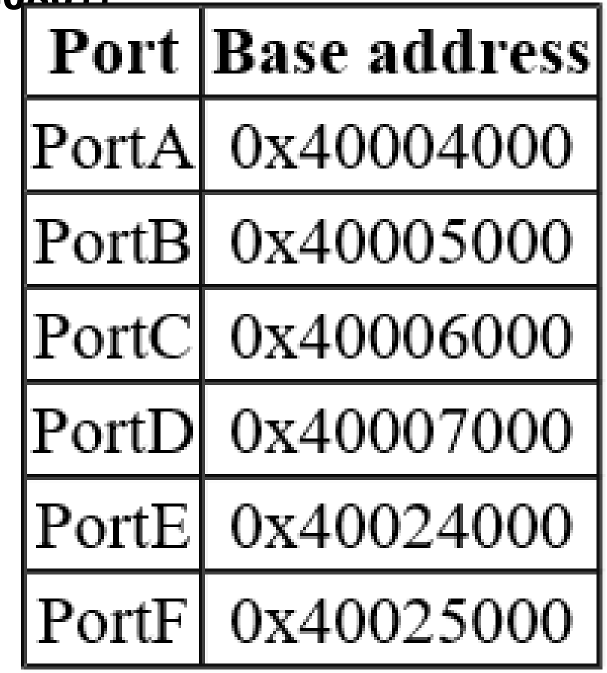
Compared to previous slide, this code only modifies the bits it is using instead of all bits for the port initialization.

Friendly code using Bit-Specific Addressing

To access only bit 5:

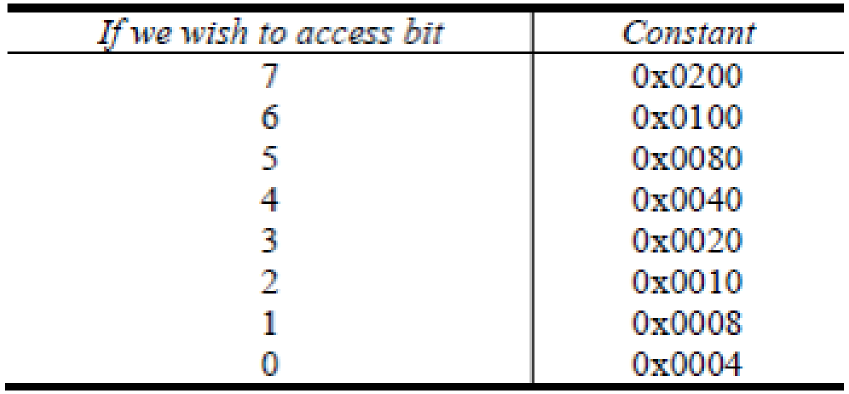
```
#define PA5          (*((volatile unsigned long *)0x40004080))

 PA5 = 0x20;       // make PA5 high
  PA5 =            // make PA5 low
 0x00;
 PA5 = PA5^0x20;   // toggle PA5
```

The **volatile** keyword is added because the value of a port can change beyond the direct action of the software. It forces the C compiler to read a new value each time through a loop and not rely on the previous value.



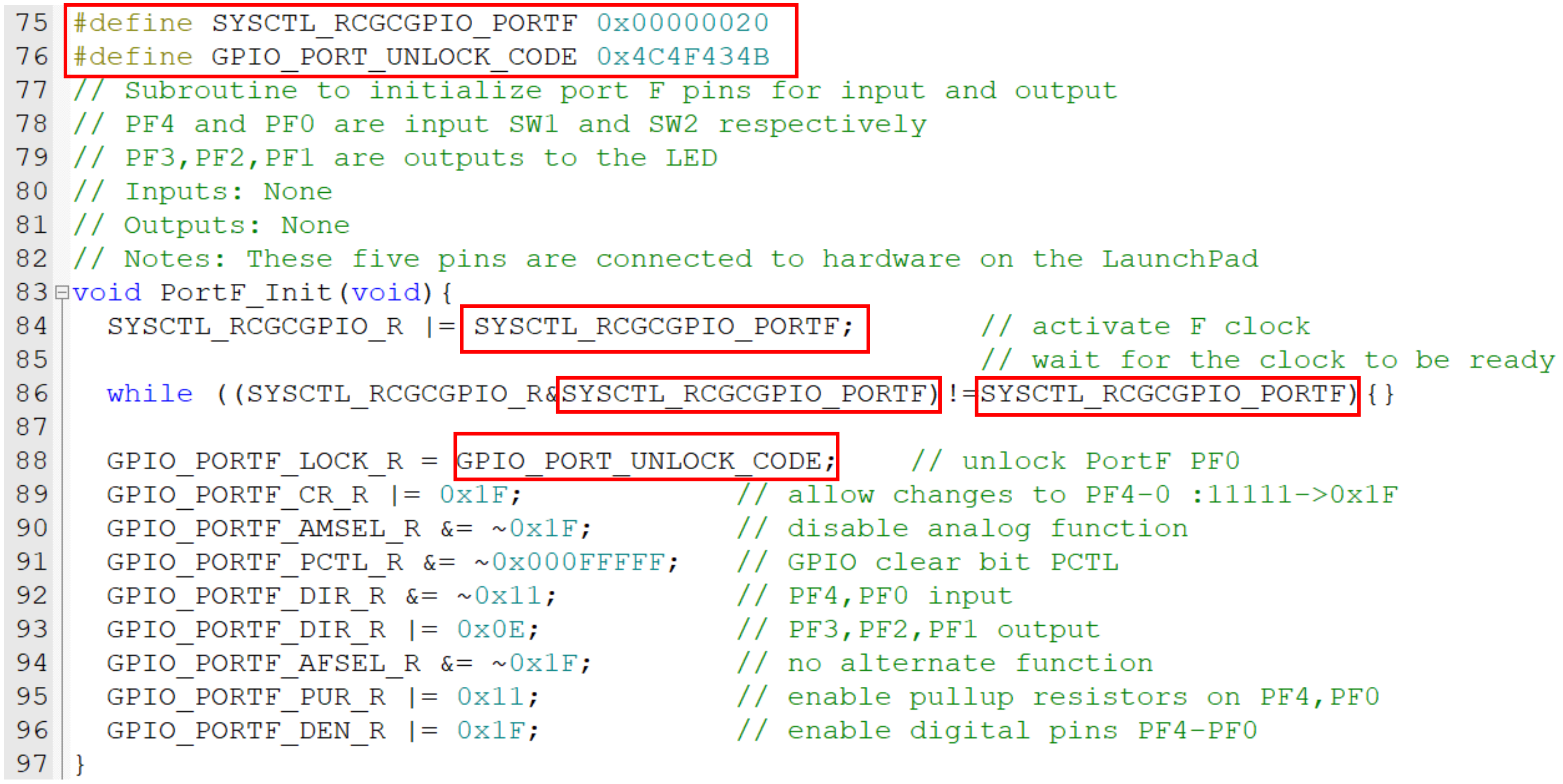
Port	Base address
PortA	0x40004000
PortB	0x40005000
PortC	0x40006000
PortD	0x40007000
PortE	0x40024000
PortF	0x40025000

*These base addresses are for the APB bus.

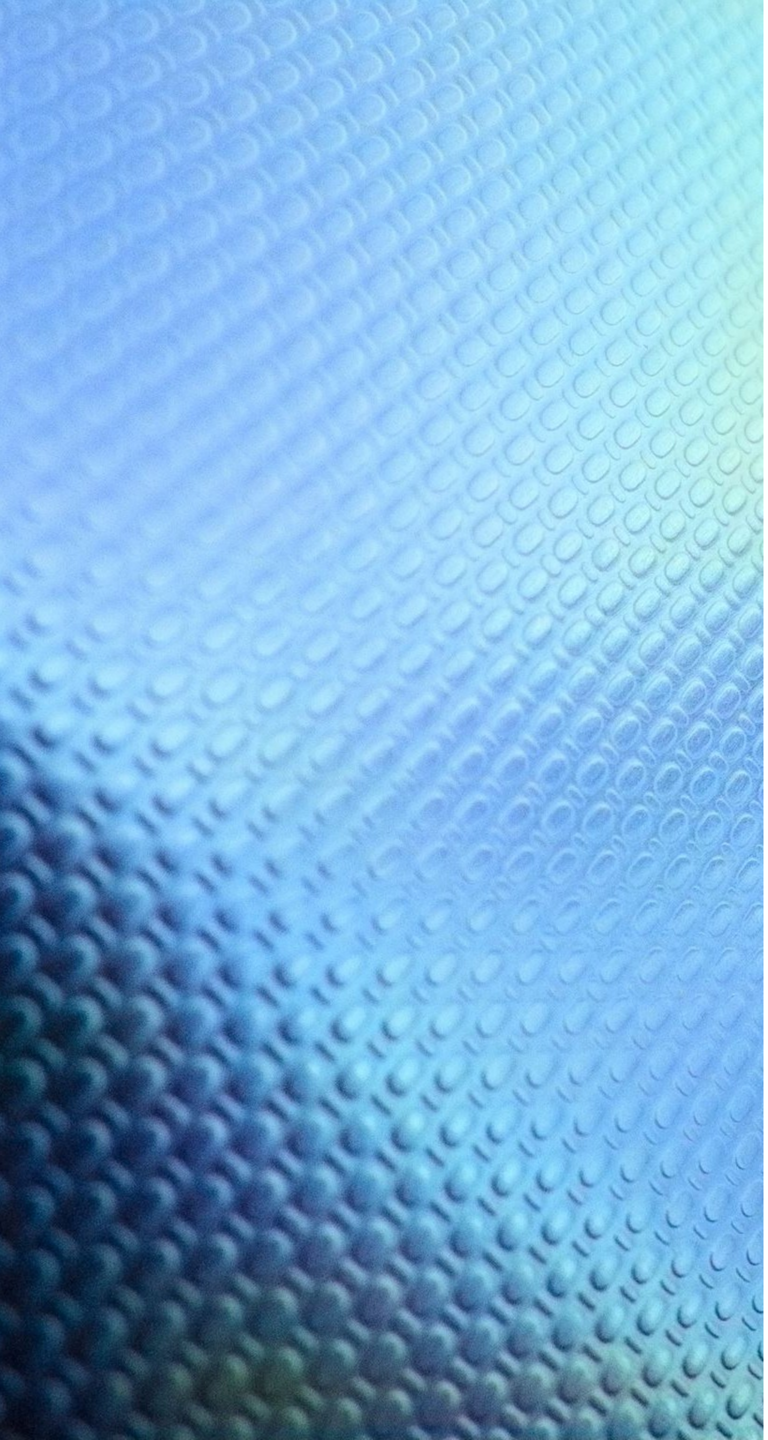


If we wish to access bit	*Constant*
7	0x0200
6	0x0100
5	0x0080
4	0x0040
3	0x0020
2	0x0010
1	0x0008
0	0x0004

Avoid Magic Numbers – Use Macros to Define Constants



```c
#define SYSCTL_RCGCGPIO_PORTF 0x00000020
#define GPIO_PORT_UNLOCK_CODE 0x4C4F434B
// Subroutine to initialize port F pins for input and output
// PF4 and PF0 are input SW1 and SW2 respectively
// PF3,PF2,PF1 are outputs to the LED
// Inputs: None
// Outputs: None
// Notes: These five pins are connected to hardware on the LaunchPad
void PortF_Init(void){
  SYSCTL_RCGCGPIO_R |= SYSCTL_RCGCGPIO_PORTF;        // activate F clock
                                                     // wait for the clock to be ready
  while ((SYSCTL_RCGCGPIO_R&SYSCTL_RCGCGPIO_PORTF)!=SYSCTL_RCGCGPIO_PORTF){}

  GPIO_PORTF_LOCK_R = GPIO_PORT_UNLOCK_CODE;     // unlock PortF PF0
  GPIO_PORTF_CR_R |= 0x1F;                  // allow changes to PF4-0 :11111->0x1F
  GPIO_PORTF_AMSEL_R &= ~0x1F;              // disable analog function
  GPIO_PORTF_PCTL_R &= ~0x000FFFFF;         // GPIO clear bit PCTL
  GPIO_PORTF_DIR_R &= ~0x11;                // PF4,PF0 input
  GPIO_PORTF_DIR_R |= 0x0E;                 // PF3,PF2,PF1 output
  GPIO_PORTF_AFSEL_R &= ~0x1F;              // no alternate function
  GPIO_PORTF_PUR_R |= 0x11;                 // enable pullup resistors on PF4,PF0
  GPIO_PORTF_DEN_R |= 0x1F;                 // enable digital pins PF4-PF0
}
```



Reading Materials and Assignments

- Textbook Chapter 4: 4.1 – 4.2.
- Tiva™ TM4C123GH6PM Microcontroller Data Sheet.
- TM4C123 Launchpad User's Guide
- Example Project: HelloLaunchPad
- Lab Assignment: Lab 2