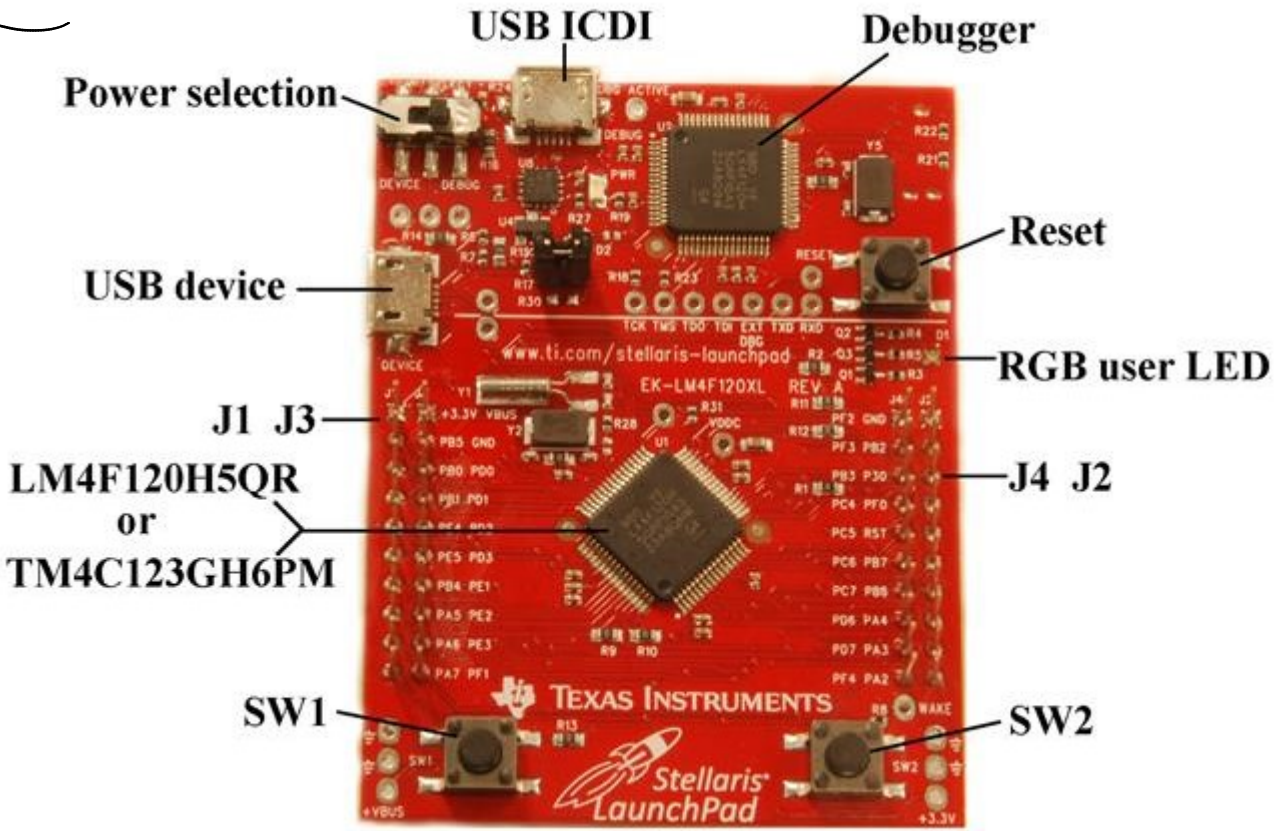
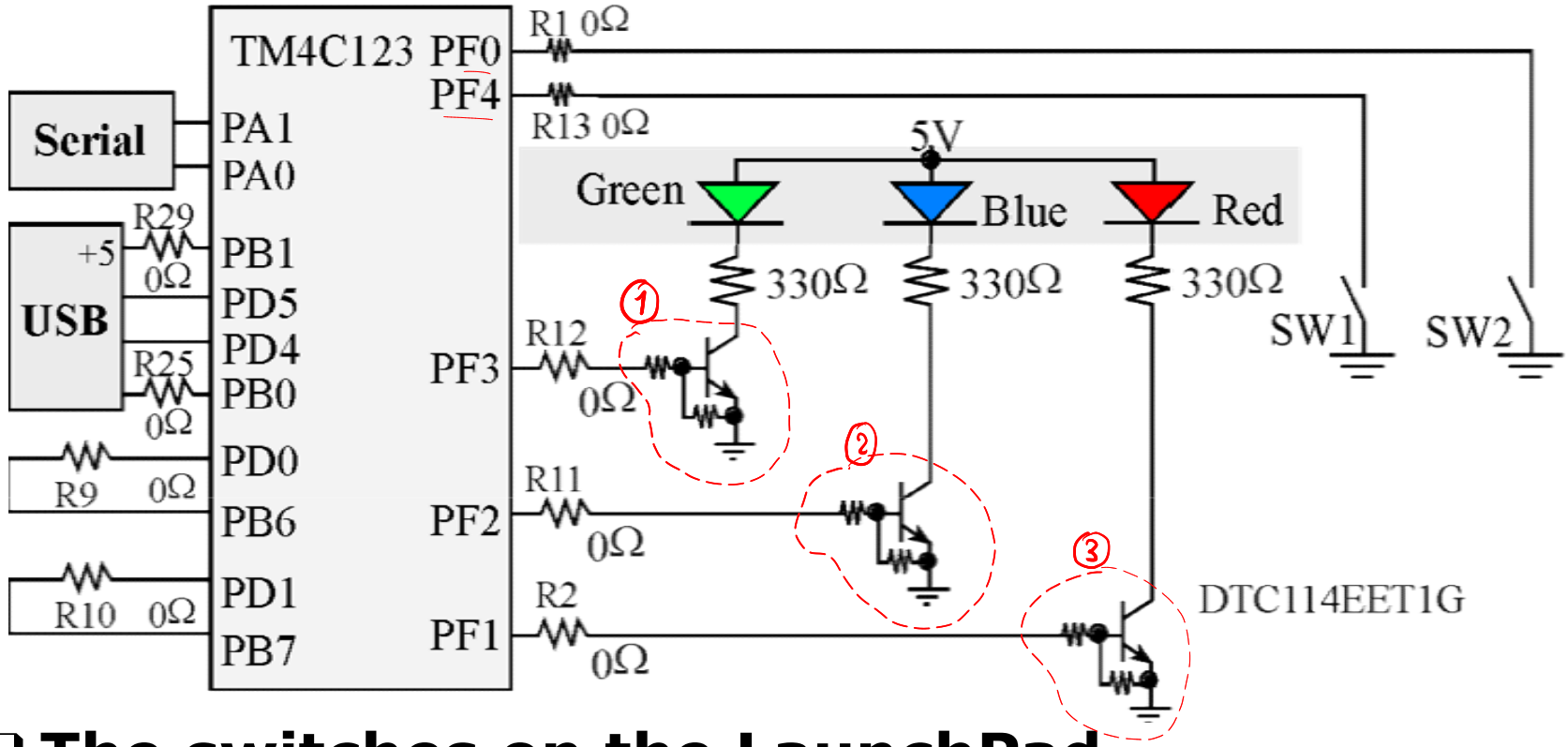
بسم الله الرحمن الرحيم

* in these slides we will discuss How to Program "Tiva C"

Tiva C Board



LaunchPad Switches and LEDs

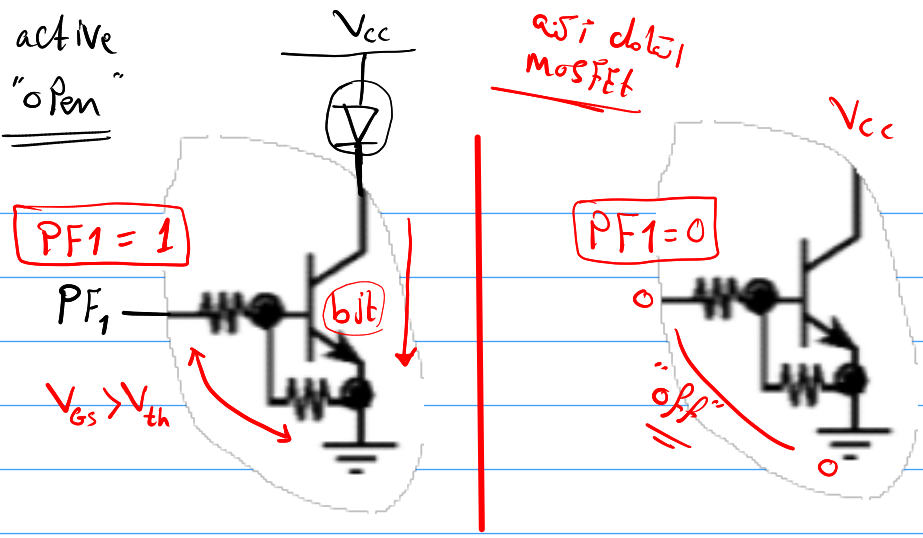


- **The switches on the LaunchPad**
  - **Negative logic**
  - **Require internal pull-up (set bits in PUR)**
- **The PF3-1 LEDs are positive logic**

On most embedded microcontrollers, the I/O ports are memory mapped. This means the software can access an input/output port simply by reading from or writing to the appropriate address. It is important to realize that even though I/O operations "look" like reads and writes to memory variables, the I/O ports often DO NOT act like memory. For example, some bits are read-only, some are write-only, some can only be cleared, others can only be set, and some bits cannot be modified. To make our software easier to understand we include symbolic definitions for the I/O ports. We set the direction register(e.g., **GPIO_PORTF_DIR_R**) to specify which pins are input and which are output. Individual port pins can be general purpose I/O (GPIO) or have an alternate function. We will set bits in the alternate function register (e.g., **GPIO_PORTF_AFSEL_R**) when we wish to activate the alternate functions listed in Tables 4.1, 4.3, and 4.4. To use a pin as a digital input or output, we must set the corresponding bit in the digital enable register(e.g., **GPIO_PORTF_DEN_R**). To use a pin as an analog input we must set the corresponding bit in theanalog mode select register (e.g., **GPIO_PORTF_AMSEL_R**). Typically, we write to the direction and alternate function registers once during the initialization phase. We use the data register(e.g., **GPIO_PORTF_DATA_R**) to perform the actual input/output on the port. Table 4.5 shows some of the parallel port registers for the TM4C123. Each of the ports has a clock, which can be separately enabled by writing to the **SYSCTL_RCGCGPIO_R** register.

The only differences among the TM4C family are the number of ports and available pins in each port. For example, the TM4C1294 has fifteen digital I/O ports A (8 bits), B (6 bits), C (8 bits), D (8 bits), E (6 bits), F (5 bits), G (2 bits), H (4 bits), J (2 bits), K (8 bits), L (8 bits), M (8 bits), N(6 bits), P (6 bits), and Q (5 bits). Furthermore, the TM4C1294 has different addresses for ports. Refer to the file **tm4c1294ncpdt.h** or to the data sheet for more the specific addresses of its I/O ports.

* if we need leds to be active
So, tranzistor should be "open"
for this
PF1 should equal "1"



8-bits

address	7	6	5	4	3	2	1	0	"Port A"
$4000.43FC	DATA	DATA	DATA	DATA	DATA	DATA	DATA	DATA	GPIO_PORTA_DATA_R ①
$4000.4400	DIR	DIR	DIR	DIR	DIR	DIR	DIR	DIR	GPIO_PORTA_DIR_R ②
$4000.4420	SEL	SEL	SEL	SEL	SEL	SEL	SEL	SEL	GPIO_PORTA_AFSEL_R ③
$4000.4510	PUE	PUE	PUE	PUE	PUE	PUE	PUE	PUE	GPIO_PORTA_PUR_R ④
$4000.451C	DEN	DEN	DEN	DEN	DEN	DEN	DEN	DEN	GPIO_PORTA_DEN_R ⑤
$4000.4524	1	1	1	1	1	1	1	1	GPIO_PORTA_CR_R ⑥
$4000.4528	0	0	0	0	0	0	0	0	GPIO_PORTA_AMSEL_R ⑦
$4000.53FC	DATA	DATA	DATA	DATA	DATA	DATA	DATA	DATA	GPIO_PORTB_DATA_R
$4000.5400	DIR	DIR	DIR	DIR	DIR	DIR	DIR	DIR	GPIO_PORTB_DIR_R
$4000.5420	SEL	SEL	SEL	SEL	SEL	SEL	SEL	SEL	GPIO_PORTB_AFSEL_R
$4000.5510	PUE	PUE	PUE	PUE	PUE	PUE	PUE	PUE	GPIO_PORTB_PUR_R
$4000.551C	DEN	DEN	DEN	DEN	DEN	DEN	DEN	DEN	GPIO_PORTB_DEN_R
$4000.5524	1	1	1	1	1	1	1	1	GPIO_PORTB_CR_R
$4000.5528	0	0	AMSEL	AMSEL	0	0	0	0	GPIO_PORTB_AMSEL_R
$4000.63FC	DATA	DATA	DATA	DATA	JTAG	JTAG	JTAG	JTAG	GPIO_PORTC_DATA_R
$4000.6400	DIR	DIR	DIR	DIR	JTAG	JTAG	JTAG	JTAG	GPIO_PORTC_DIR_R

Port B

F ، E ، D ، C ، B

GPIO_PORTA_DATA_R
GPIO_PORTA_DIR_R
GPIO_PORTA_AFSEL_R
GPIO_PORTA_PUR_R
GPIO_PORTA_DEN_R
GPIO_PORTA_CR_R
GPIO_PORTA_AMSEL_R

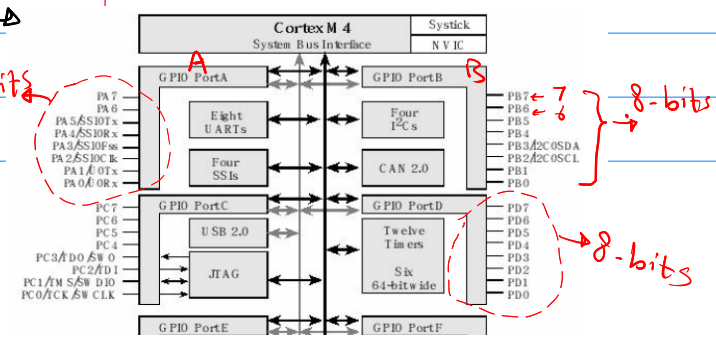
لوقتي لازم تكون فاهم كل واحد من دول بيعمل إيه ووظيفته إيه على سبيل المثال الـ Ports الموجودة في الـ tiva لو هتتحكم بـ Comparator أو هتدخلها جهزي الصوت أو أي حاجه analog فأنت لازم تعرفها إن إنت شغال digital ولا analog وبالتالي عندك إتنين Registers لازم تحددهم

→ DEN-R → for digital enable "0"
→ AMSEL → for analog selection "1"

وعلى كده قيس ولازم تفهم إن كل Port سواء A ولا B ولا C وغيرهم ليهم 8-bits ودول من الـ Registers كل bit بتمثل Pin واحد في البوردة بمعنى

7- Register [---- - DIR - DEN]

↳ if Port-A → [PA0 - PA1 - ... - PA7]



```
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 |= 0x010//Enable clock for PORTE
RCGCGPIO |= 0x020//Enable clock for PORTF
```

ده اللى بيحدد الـ Port اللى هيشتغل إذا كان A ، F ، E ، ---
يعنى لو هشتغل بـ F هروح أخلى (bit5) فى الجدول بتاع الـ Ports بواحد

← SYSCTL_RCGCGPIO_R register

GPIO_PORTF_DIR_R — **Which pins are input or output.**
GPIO_PORTF_AFSEL_R — **Activate the alternate functions**
GPIO_PORTF_DEN_R: — **Digital port**
GPIO_PORTF_DATA_R — **Perform input/output on the port.**

PF4 PF3 PF2 PF1 PF0

Address	7	6	5	4	3	2	1	0	Name
$400F.E608	-	-	GPIOF	GPIOE	GPIOD	GPIOC	GPIOB	GPIOA	SYSCTL_RCGCGPIO_R
$400F.EA08	-	-	GPIOF	GPIOE	GPIOD	GPIOC	GPIOB	GPIOA	SYSCTL_PRGPIO_R
$4002.53FC	-	-	-	DATA	DATA	DATA	DATA	DATA	GPIO_PORTF_DATA_R
$4002.5400	-	-	-	DIR	DIR	DIR	DIR	DIR	GPIO_PORTF_DIR_R
$4002.5420	-	-	-	SEL	SEL	SEL	SEL	SEL	GPIO_PORTF_AFSEL_R
$4002.5510	-	-	-	PUE	PUE	PUE	PUE	PUE	GPIO_PORTF_PUR_R
$4002.551C	-	-	-	DEN	DEN	DEN	DEN	DEN	GPIO_PORTF_DEN_R
$4002.5524	-	-	-	1	1	1	1	CR	GPIO_PORTF_CR_R
$4002.5528	-	-	-	0	0	0	0	0	GPIO_PORTF_AMSEL_R

* عشان أعرفه إنه أنا شغال "F"

SYSCTL_RCGCGPIO_R register

* يبقى عشان أشغل "F" القيمة بتاعت هتكون 00100000 = 0x20

هنحتاجها بعدين

* F-Port → only has 5 bits → PF0 - PF4 → for 2-switches
PF1, PF2, PF3 → for LEDs.

ده بيوضحلك فكرة الـ AFSEL بس مش لزمانى دلوقتى لأنى بعملها disable واحنا الهدف من السلايدز دى "test LEDs"

Port F

Regular	Alternate Pin Name	Alternate Function
PA0 – PA1	U0RX, U0TX	Universal Asynchronous Receiver/Transmit, UART0
PA2 – PA5	S0CLK, S0FS, S0RX, S0TX	Synchronous Serial Interface, SSI0
PA6 – PA7	SCL1, SDA1	Inter-Integrated Circuit, I²C1
PB0	CCP0	Timer 0A Capture/Compare
PB1	CCP2	Timer 1A Capture/Compare
PB2 – PB3	SCL0, SDA0	Inter-Integrated Circuit, I²C0
PB4, PB6, PF4	C0-, C0+, C0o	Analog Comparator 0

Address	7	6	5	4	3	2	1	0	Name
$400F.E108	--	--	GPIOF	GPIOE	GPIOD	GPIOC	GPIOB	GPIOA	SYSCTL_RCGC2_R
$4000.43FC	DATA	DATA	DATA	DATA	DATA	DATA	DATA	DATA	GPIO_PORTA_DATA_R
$4000.4400	DIR	DIR	DIR	DIR	DIR	DIR	DIR	DIR	GPIO_PORTA_DIR_R
$4000.4420	SEL	SEL	SEL	SEL	SEL	SEL	SEL	SEL	GPIO_PORTA_AFSEL_R
$4000.4510	PUE	PUE	PUE	PUE	PUE	PUE	PUE	PUE	GPIO_PORTA_PUR_R
$4000.451C	DEN	DEN	DEN	DEN	DEN	DEN	DEN	DEN	GPIO_PORTA_DEN_R
$4000.4524	1	1	1	1	1	1	1	1	GPIO_PORTA_CR_R
$4000.4528	0	0	0	0	0	0	0	0	GPIO_PORTA_AMSEL_R
$4000.53FC	DATA	DATA	DATA	DATA	DATA	DATA	DATA	DATA	GPIO_PORTB_DATA_R
$4000.5400	DIR	DIR	DIR	DIR	DIR	DIR	DIR	DIR	GPIO_PORTB_DIR_R
$4000.5420	SEL	SEL	SEL	SEL	SEL	SEL	SEL	SEL	GPIO_PORTB_AFSEL_R
$4000.5510	PUE	PUE	PUE	PUE	PUE	PUE	PUE	PUE	GPIO_PORTB_PUR_R
$4000.551C	DEN	DEN	DEN	DEN	DEN	DEN	DEN	DEN	GPIO_PORTB_DEN_R
$4000.5524	1	1	1	1	1	1	1	1	GPIO_PORTB_CR_R
$4000.5528	0	0	AMSEL	AMSEL	0	0	0	0	GPIO_PORTB_AMSEL_R
$4000.63FC	DATA	DATA	DATA	DATA	JTAG	JTAG	JTAG	JTAG	GPIO_PORTC_DATA_R
$4000.6400	DIR	DIR	DIR	DIR	JTAG	JTAG	JTAG	JTAG	GPIO_PORTC_DIR_R
$4000.6420	SEL	SEL	SEL	SEL	JTAG	JTAG	JTAG	JTAG	GPIO_PORTC_AFSEL_R
$4000.6510	PUE	PUE	PUE	PUE	JTAG	JTAG	JTAG	JTAG	GPIO_PORTC_PUR_R
$4000.651C	DEN	DEN	DEN	DEN	JTAG	JTAG	JTAG	JTAG	GPIO_PORTC_DEN_R

---

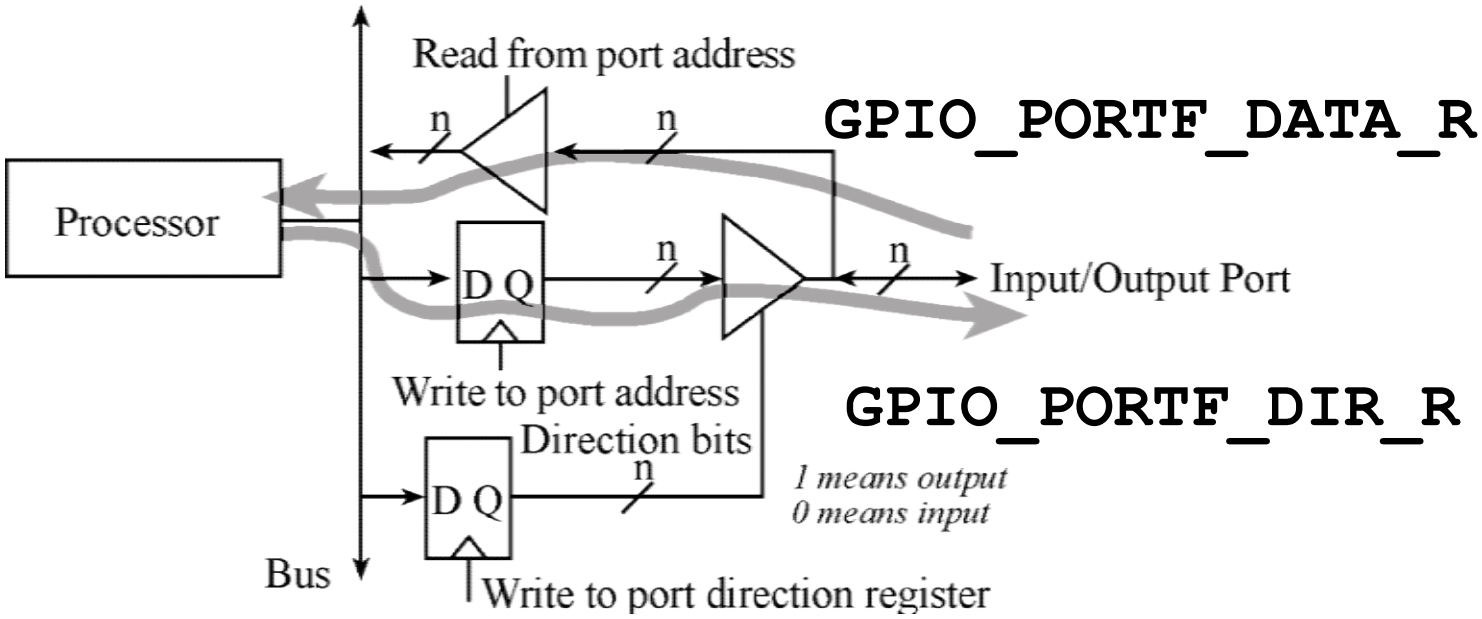
① SYSCTL_RCGCGPIO_R register

② PRGPIO

First, we activate the clock for the port by setting the corresponding bit in **RCGCGPIO** register. Because it takes time for the clock to stabilize, we next will wait for its status bit in the **PRGPIO** to be true. Second, we unlock the port;

* يعنى الـ PRGPIO ده بخليني أستنى ومكملش لحد ما الـ Port بجهز [Port بياخد clock ويكون جاهز]

I/O Ports and Control Registers



The input/output direction of a bidirectional port is specified by its direction register.

GPIO_PORTF_DIR_R , specify if corresponding pin is input or output:

- 0 means input
- 1 means output

I/O Ports and Control Registers

Address	7	6	5	4	3	2	1	0	Name
400F.E608	-	-	**GPIOF**	GPIOE	GPIOD	GPIOC	GPIOB	GPIOA	SYSCTL_RCGCGPIO_R
4002.53FC	-	-	-	DATA	DATA	DATA	DATA	DATA	GPIO_PORTF_DATA_R
4002.5400	-	-	-	DIR	DIR	DIR	DIR	DIR	GPIO_PORTF_DIR_R
4002.551C	-	-	-	DEN	DEN	DEN	DEN	DEN	GPIO_PORTF_DEN_R

خطوات برمجة البورده

- **Initialization (executed once at beginning)**
  1. Write *DIR* bit, 1 for output or 0 for input
  2. Set *DEN* bits to 1 to enable data pins
- **Input/output from pin**
  Input: Read from **`GPIO_PORTF_DATA_R`**
  Output: Write **`GPIO_PORTF_DATA_R`**

١- تحدد له أنت output ولا input
٢- ،، ،، analog ولا digital
٣- ،، ،، Read ولا Write

Initialization of an I/O port

To initialize an I/O port for general use:

1. Activate the clock for the port in the Run Mode Clock Gating Control Register 2 (**RCGC2**).
2. Unlock the port (**LOCK** = **0x4C4F434B**). This step is only needed for pins **PC0-3**, **PD7** and **PF0** on TM4C123GXL LaunchPad.
3. Disable the analog function of the pin in the Analog Mode Select register (**AMSEL**), because we want to use the pin for digital I/O. If this pin is connected to the ADC or analog comparator, its corresponding bit in **AMSEL** must be set as **1**. In our case, this pin is used as digital I/O, so its corresponding bit must be set as **0**.
4. Clear bits in the port control register (**PCTL**) to select regular digital function. Each GPIO pin needs four bits in its corresponding PCTL register. Not every pin can be configured to every alternative function.
5. Set its direction register (**DIR**). A DIR bit of **0** means input, and **1** means output.
6. Clear bits in the alternate Function Select register (**AFSEL**). → disable
7. Enable digital port in the Digital Enable register (**DEN**).

- Please note that we need to add a short delay between activating the clock and setting the port registers.
- **PC0-PC3** is used for JTAG connections to the debugger on the LaunchPad. So we'd better do not use these pins normally.

Set Port Direction & Port Type

```
LDR     R1,= GPIO_PORTF_DIR_R
MOV     R0,#0x0E
STR     R0,[R1]

LDR     R1,=GPIO_PORTF_DEN_R
MOV     R0,#0xFF
STR     R0,[R1]
```

Set Port Direction & Port Type

```
GPIO_PORTF_DIR_R = 0x0E;          // PF4,PF0 in, PF3-1 out
GPIO_PORTF_AFSEL_R = 0x00;        // disable alt funct on PF7-0
GPIO_PORTF_DEN_R = 0x1F;          // enable digital I/O on PF4-0
```

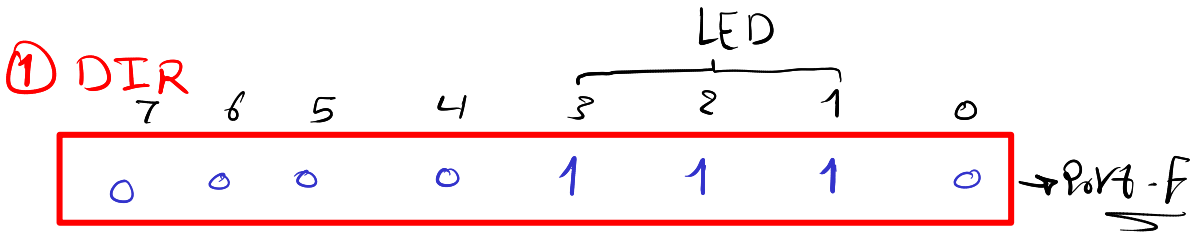
Port F LED Programming

Set LED

```
DIR ←(1) DR R1, =GPIO_PORTF_DIR_R       ; R1 -> GPIO_PORTE_DIR_R
         MOV R0, #0X0E                    ;PF0 , PF4 in and PF3-1 out
         STR R0, [R1]                    ; set direction register

(2)      LDR R1, =GPIO_PORTF_DEN_R       ; R1 -> GPIO_PORTE_DEN_R
         MOV R0, R0, #0xFF               ; enable digital port
         STR R0, [R1]                    ; set digital enable register

(3)      LDR R1, =GPIO_PORTF_DATA_R
         MOV R0,#0x02 → 0000 0010
         STR R0,[R1]
                          ↳PF1 → Red LED
```



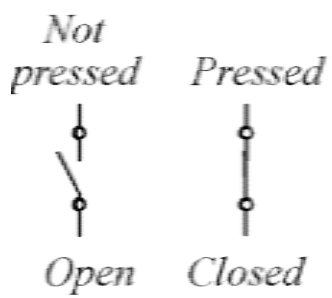
0X0E ← القيمة المخزنة في DIR

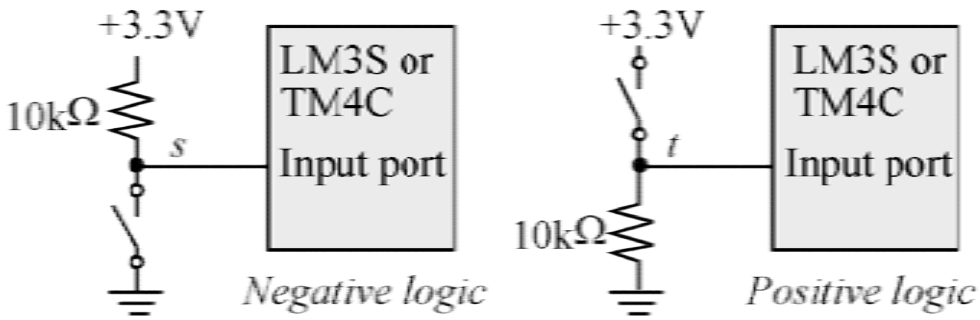
2 DEN → 0xFF → activation all Port-F

3

* بالنسبة لل switches و هم inputs طب هم read ولا write ؟ ← هم read لأن ال switch أفعل الـ
أنا مش بكتب عليه حاجه هو
على طول بيروح يجيب جوا
PF0-PF4 → "0" ولا "1"
وبالتالى أنا عاوز
أقرأ القيم
المتخزنه دى
عشان بمجرد ما
switch أفعل الـ
بتاعتى data الـ
تظهر سليمة

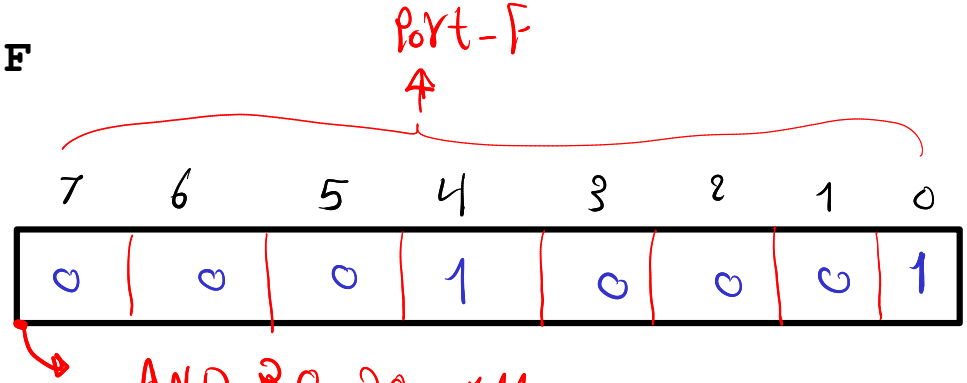
Switch Interfacing





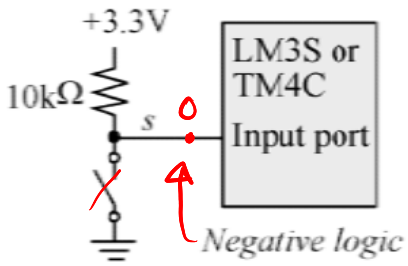
Assembly:

```
LDR   R1,=GPIO_PORTF_DATA_R
LDR   R0,[R1]        ; read port F
AND   R0,R0,#0x11    ; PF4-PF0
```

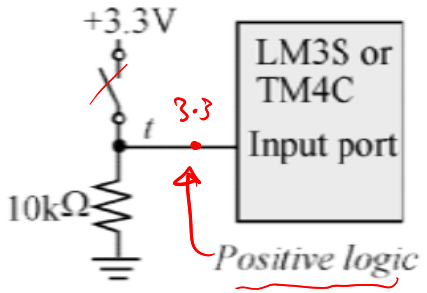


كده R0 ← بيحتوى على القيمة اللى بتمثل
إذا كان الـ switches مقفولة
ولا مفتوحة.

Pull up resistor



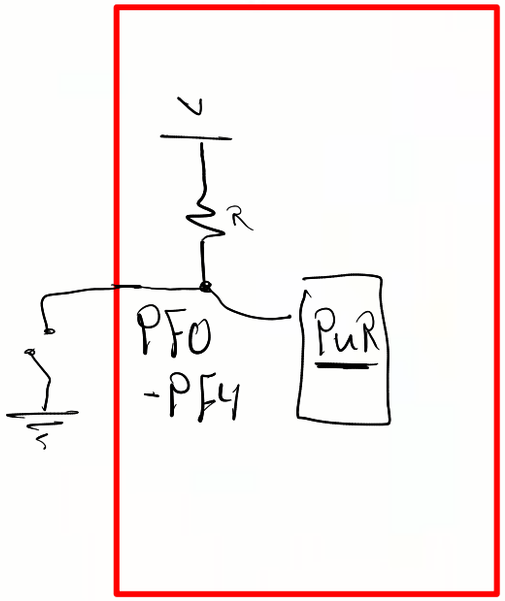
Pull down

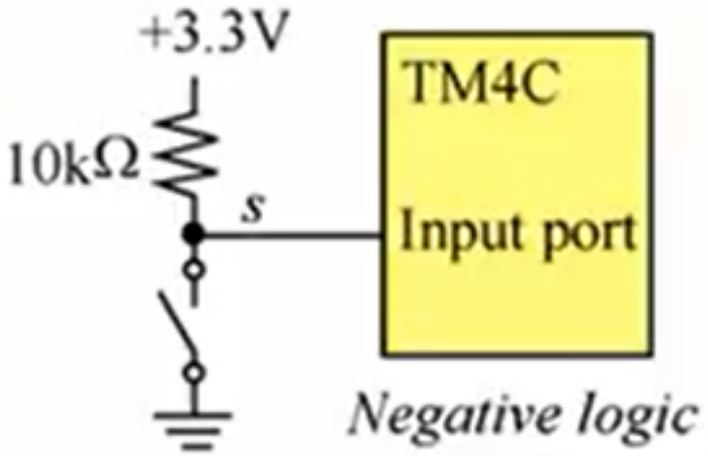


Pull Up resistor – Switch Interface

(Pull up) ← switches ← Tiva و ال down up * في الصفحة اللي قبل دي عرفنا

* PUR ← ده لو "1" ← يبقى أنت مفعل ال resis. ولو "0" يبقى disable





If we want to read and write all 8 bits of Port A, the constants will add up to 0x03FC. Notice that the sum of the base address (0x4000.4000) and all the constants yields the 0x4000.43FC address used in Table 4.5 and Program 4.1. In other words, read and write operations to **GPIO_PORTA_DATA_R** will access all 8 bits of Port A. If we are interested in just bit 5 of Port A, we add 0x0080 to 0x4000.4000, and we can define this in C and in assembly as

#define PA5 (*((volatile uint32_t *)0x40004080)) → C Progr.
PA5 EQU 0x40004080 → assembly — (=)

#define PA5 (*((volatile uint32_t *)0X40004080))

Base address

PA0 → 0X40004000
to get PA5 → (like array) You should add (0X0080)
على الـ Base register بس في مشكلة وهي إن الرقم الهيكسا ده الـ compiler مش عارف هو Data ولا address

*(→ معناه هات الـ Value اللي مشاور عليها بـ address

*( ) اللي موجود جوا

type casting عشان أحدد إن الرقم ده address

Quick recall

① `int* x = &y; // x is pointer, get address y`
`//and store it in Pointer x`

② `int z = *x; // get Value Pointed to by Pointer x`
`//and store Value in Var. "z"`

---

* أنا دلوقتي بعد ما عملت define لـ PA5 ← أقدر أكتب فيها عادي

imp. Note →

To understand the port definitions in C, we remember **#define** is simply a copy paste. E.g.,

data = PA5;

becomes

data = (*((volatile uint32_t *)0x40004080));

To understand why we define ports this way, let's break this port definition into pieces. First, 0x40004080 is the address of Port Abit 5. If we write just **#define PA5 0x40004080** it will create

data = 0x40004080;

which does not read the contents of PA5 as desired. This means we need to dereference the address. If wewrite **#define PA5 (*0x40004080)** it will create

data = (*0x40004080);

This will attempt to read the contents at 0x40004080, but doesn't know whether to read 8, 16, or 32 bits. So the compiler gives a syntax error because the type of data does not match the type of (*0x40004080). To solve a type mismatch in C we **typecast**, placing a (new type) in front of the object we wish to convert. We wish force the type conversion to unsigned 32 bits, so we modify the definition to include the typecast,

#define PA5 (*((volatile uint32_t *)0x40004080))

The **volatile** 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.

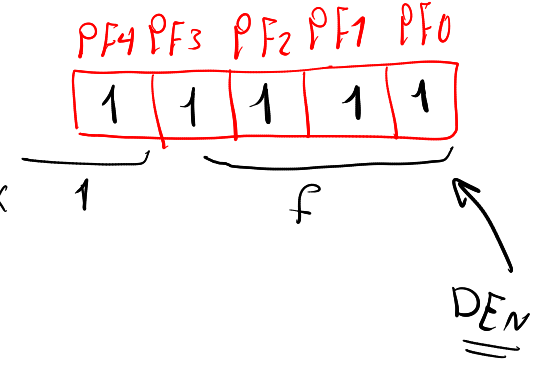
```
#include "inc/tm4c123gh6pm.h"
void PortF_Init(void){
  SYSCTL_RCGCGPIO_R |= 0x00000020; // 1) activate clock for Port F
  while((SYSCTL_PRGPIO_R&0x00000020) == 0){};// ready?
  GPIO_PORTF_LOCK_R = 0x4C4F434B;   // 2) unlock GPIO Port F
  GPIO_PORTF_CR_R = 0x1F;       // allow changes to PF4-0
  GPIO_PORTF_DIR_R = 0x0E;        // 5) PF4,PF0 in, PF3-1 out
  GPIO_PORTF_PUR_R = 0x11;        // enable pull-up on PF0 and PF4
  GPIO_PORTF_DEN_R = 0x1F;        // 7) enable digital I/O on PF4-0
}
uint32_t PortF_Input(void){
  return (GPIO_PORTF_DATA_R&0x11); // read PF4,PF0 inputs
}
void PortF_Output(uint32_t data){   // write PF3-PF1 outputs
  GPIO_PORTF_DATA_R = data;
}
```

header file has all #defines for all Ports

Logic

Wait Port F to be ready

switches → check again



ده شرح الـ c-fuct. اللي فوق ف كود الـ ؟

The function PortF_Input will read from all five Port F pins, and return a value depending on the status of the input pins at the time of the read. As shown in Figure 4.13, when writing to an I/O port, the input pins are not affected, and the output pins are changed to the value written to the port. That value remains until written again. The function PortF_Output will write new values to the three output pins.The #include will define symbolic names for all the I/O ports for that microcontroller. The header files are in the inc folder. Use the one for your microcontroller.
#include "inc/tm4c123gh6pm.h"

ده نفس كود الـ ؟ اللي فوق بس الـ assembly

لازم يعمل Wait واستعمل اتنين no operation

2- clock cycles

لأن الـ Port عشانه يجهز محتاج

```
PortF_Init
  LDR R1, =SYSCTL_RCGCGPIO_R    ; 1) activate clock for Port F
  LDR R0, [R1]
  ORR R0, R0, #0x20             ; set bit 5 to turn on clock
  STR R0, [R1]
  NOP
  NOP                           ; allow time for clock to finish
  LDR R1, =GPIO_PORTF_LOCK_R    ; 2) unlock the lock register
  LDR R0, =0x4C4F434B           ; unlock GPIO Port F Commit Register
  STR R0, [R1]
  LDR R1, =GPIO_PORTF_CR_R      ; enable commit for Port F
  MOV R0, #0xFF                 ; 1 means allow access
  STR R0, [R1]
  LDR R1, =GPIO_PORTF_DIR_R     ; 5) set direction register
  MOV R0, #0x0E                 ; PF0 and PF7-4 input, PF3-1 output
  STR R0, [R1]
  LDR R1, =GPIO_PORTF_PUR_R     ; pull-up resistors for PF4,PF0
  MOV R0, #0x11                 ; enable weak pull-up on PF0 and PF4
  STR R0, [R1]
  LDR R1, =GPIO_PORTF_DEN_R     ; 7) enable Port F digital port
  MOV R0, #0xFF                 ; 1 means enable digital I/O
  STR R0, [R1]

  BX  LR

PortF_Input
  LDR R1, =GPIO_PORTF_DATA_R ; pointer to Port F data
  LDR R0, [R1]               ; read all of Port F
  AND R0,R0,#0x11            ; just the input pins, bits 4,0
  BX  LR                  ; return R0 with inputs
PortF_Output
  LDR R1, =GPIO_PORTF_DATA_R ; pointer to Port F data
  STR R0, [R1]               ; write to PF3-1
  BX  LR
```