GPIO

Part of the slides are
by Prof. Shiao-Li Tsao
NCTU CS 2016

GPIO講解大綱

- GPIO 硬體架構與外接電路
- GPIO 組合語言控制register，register位址和值
  - Reference manual和programming manual定義reg和function (重要! 花十分鐘帶你看，以後要學會自己查)
  - 記憶體分配，GPIO register設定，程式範例
  - 補充資料-如何看register設定
- Lab電路
- 範例程式，練習與作業

GPIO講解大綱

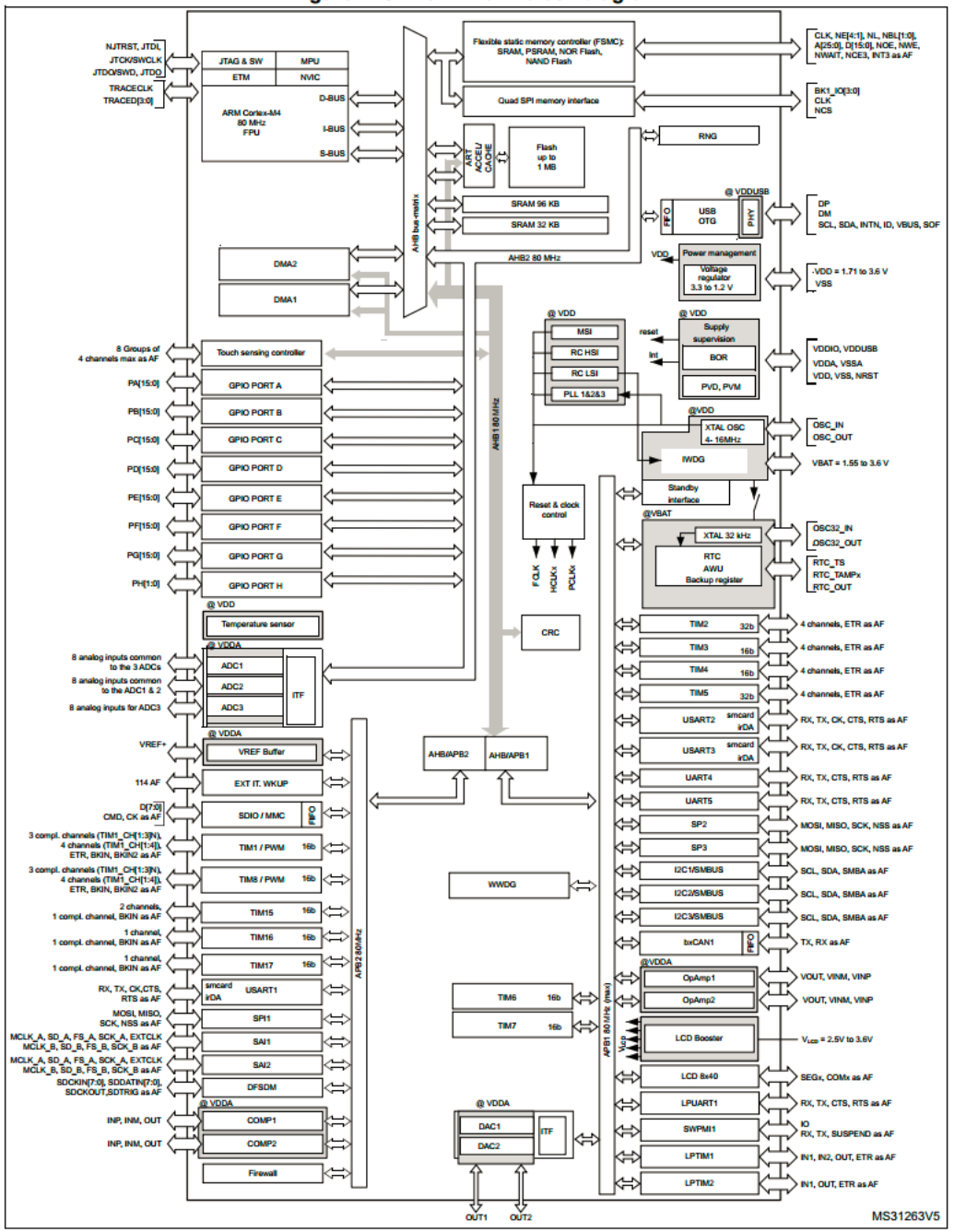
- GPIO 硬體架構與外接電路
- GPIO 組合語言控制register，register位址
  - Reference manual和programming manual定義reg和function (重要! 花十分鐘帶你看，以後要學會自己查)
  - 記憶體分配，GPIO register設定，程式範例
  - 補充資料-如何看register設定
- 輸出電路
- 範例程式，練習與作業

Block diagram

- Advanced Microcontroller Bus Architecture (AMBA)
  - Advanced High-performance Bus (AHB)
  - Advanced Peripheral Bus (APB)

只要follow "ARM bus protocol"
不同公司的device也可以相容
I/O快的離CPU近，I/O慢的離CPU遠

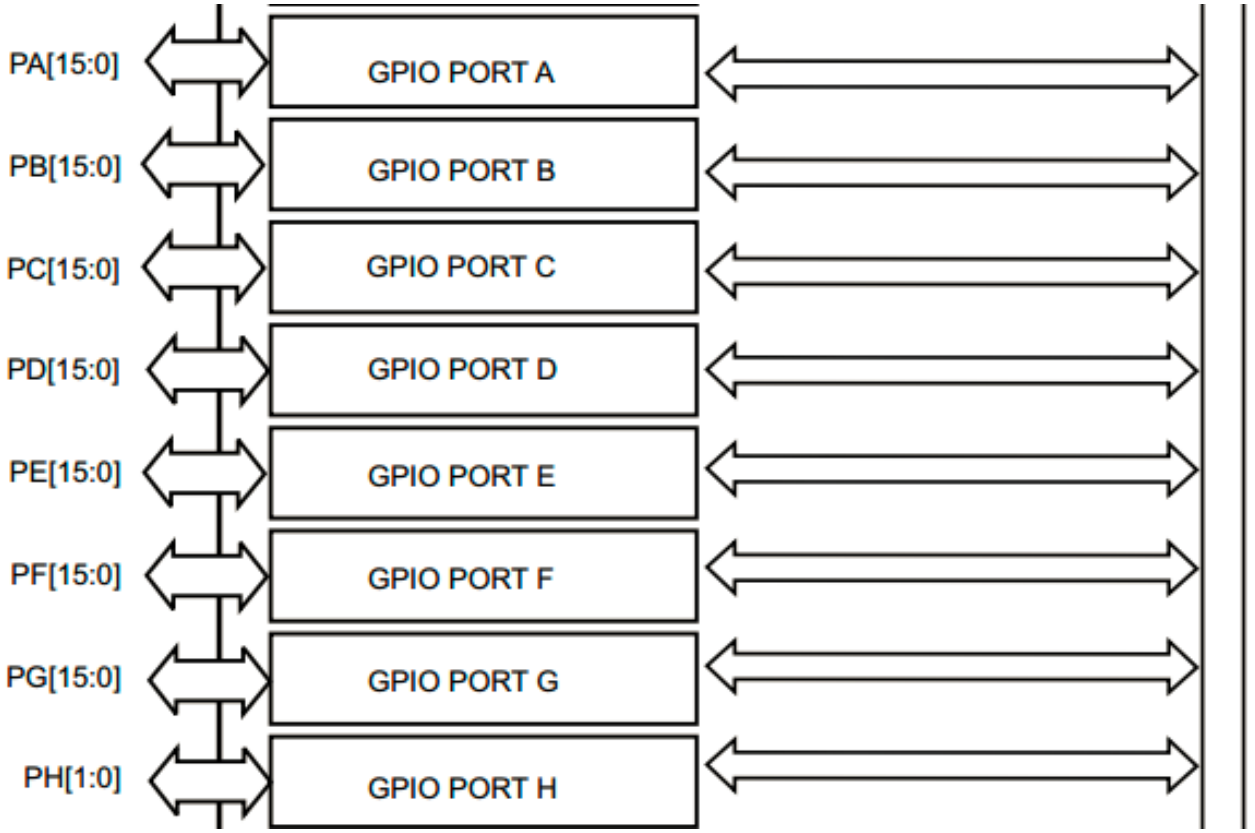
Figure 1. STM32L476xx block diagram



Note: AF: alternate function on I/O pins.

General-purpose inputs/outputs (GPIO)

- • STM32L476 have port A~H GPIO port connect on AHB2 bus
- • Except port H, each port have 16 pins
- • Our STM32L476RG chip can use
  - • **PA[0..15], PB[0..15], PC[0..15]**
  - • **PF[0..1], PF[4..7], PD2, PD8**



Nucleo Board Extension Connector

- • 用於連接GPIO與外部電路
- • 同學可參考Reference manual了解內部連接方式

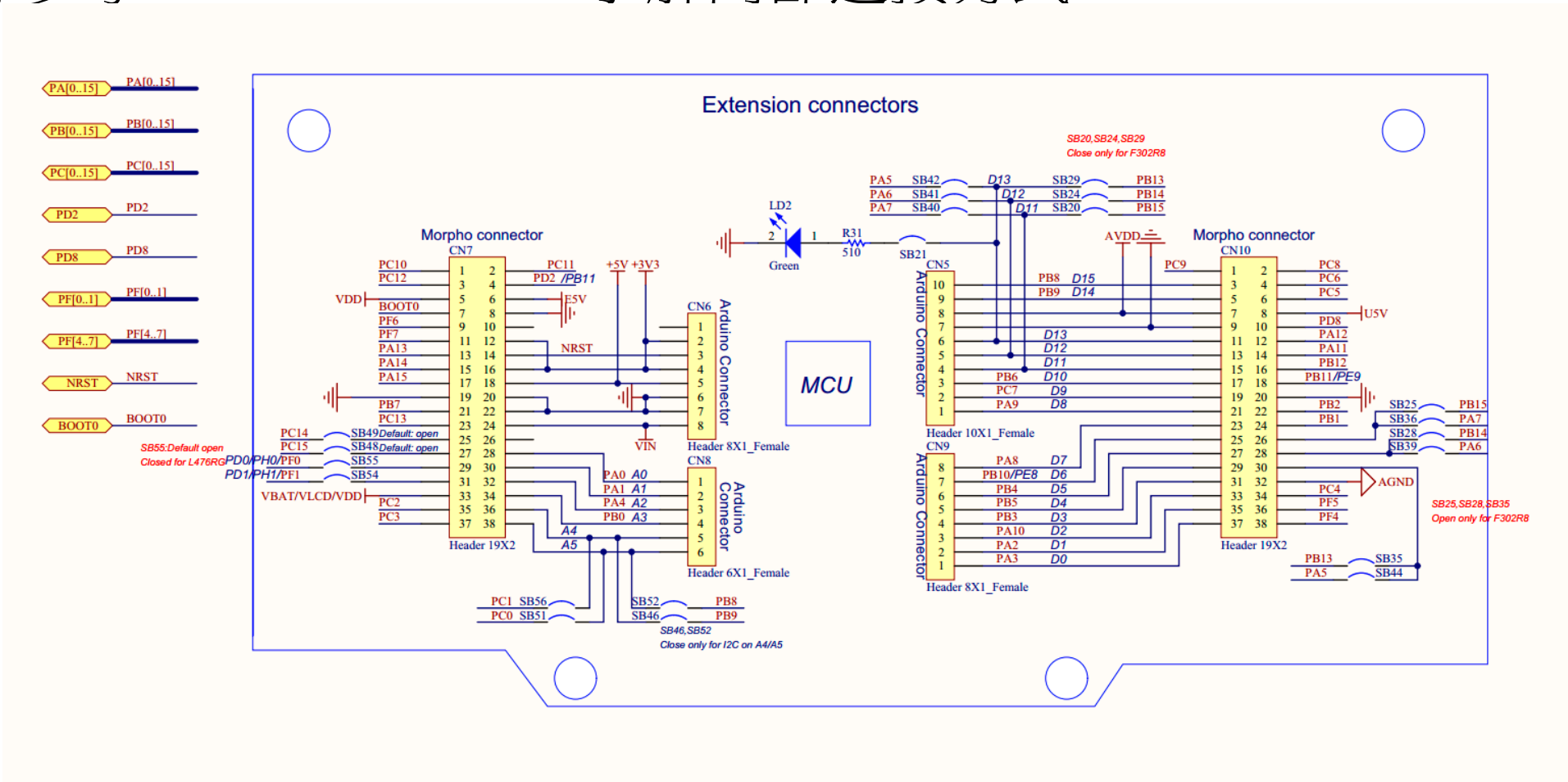
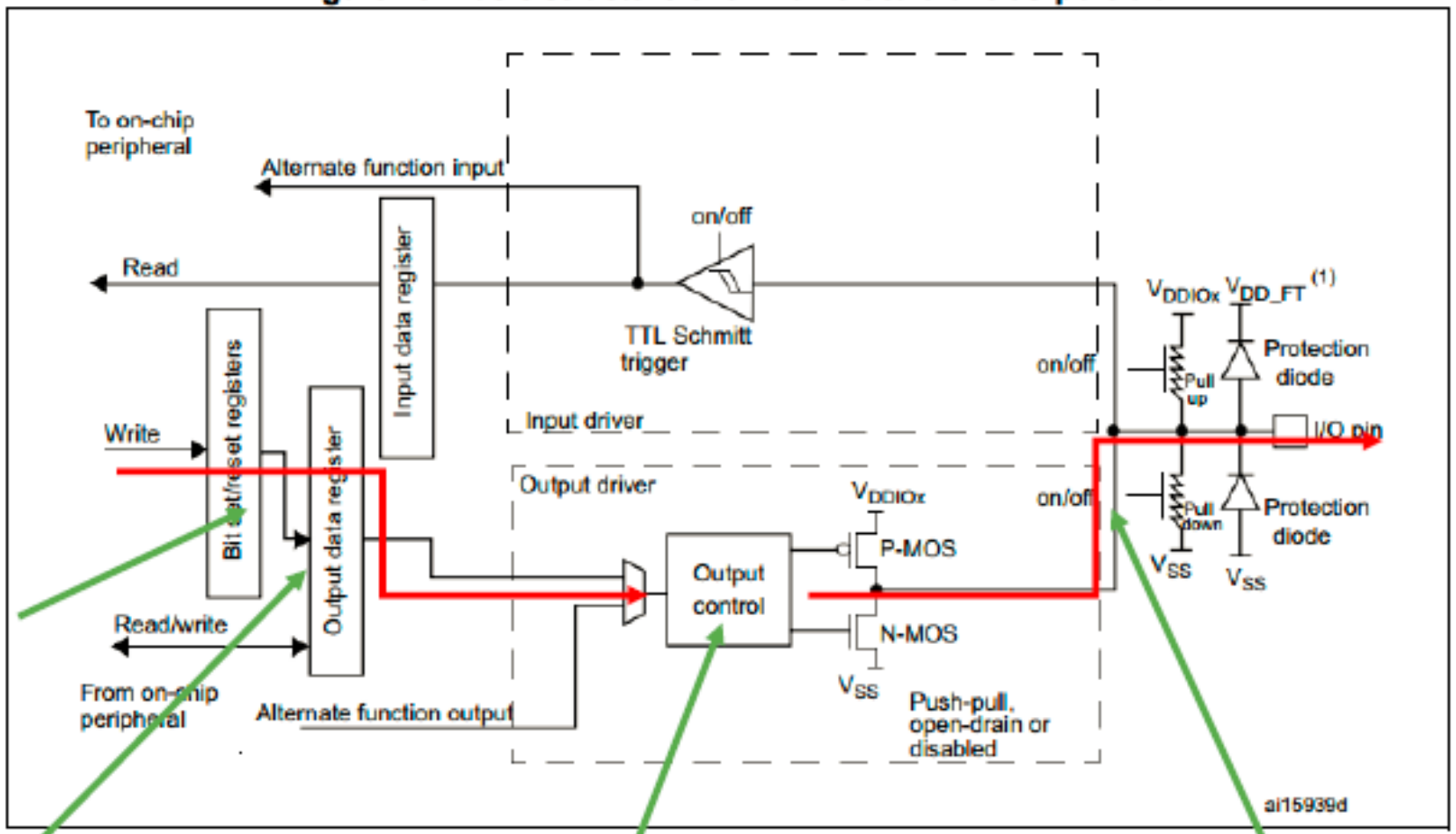


Figure 18. Basic structure of a five-volt tolerant I/O port bit



1. V_{DD_FT} is a potential specific to five-volt tolerant I/Os and different from V_{DD}.

Bit set/reset register(BSRR)

Output data register(ODR)

Output type register(OTYPER)

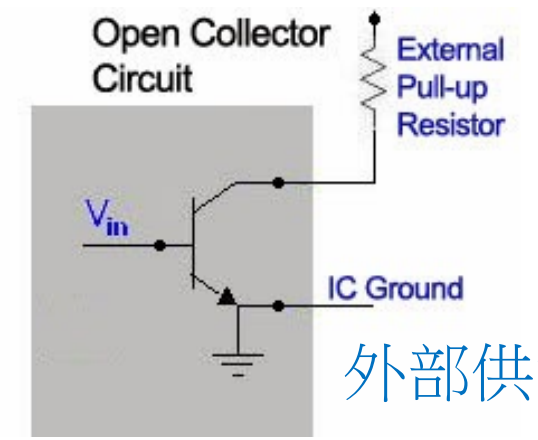
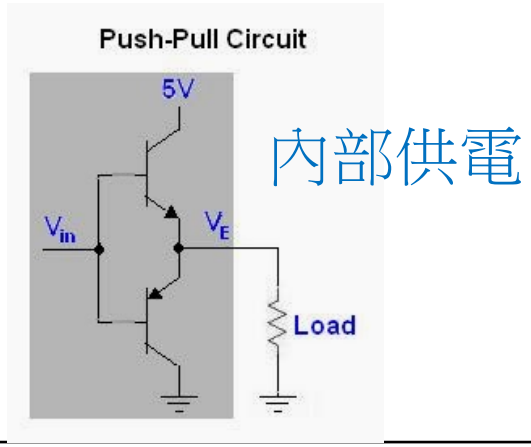
Pull-up control register(PUPDR)

本slide和slide26 Configuration Reference印出供同學參考

GPIO講解大綱

- GPIO 硬體架構
  - Open-drain vs. push-pull
  - Schmitt trigger vs. OP-AMP Comparator
  - Protection Diode

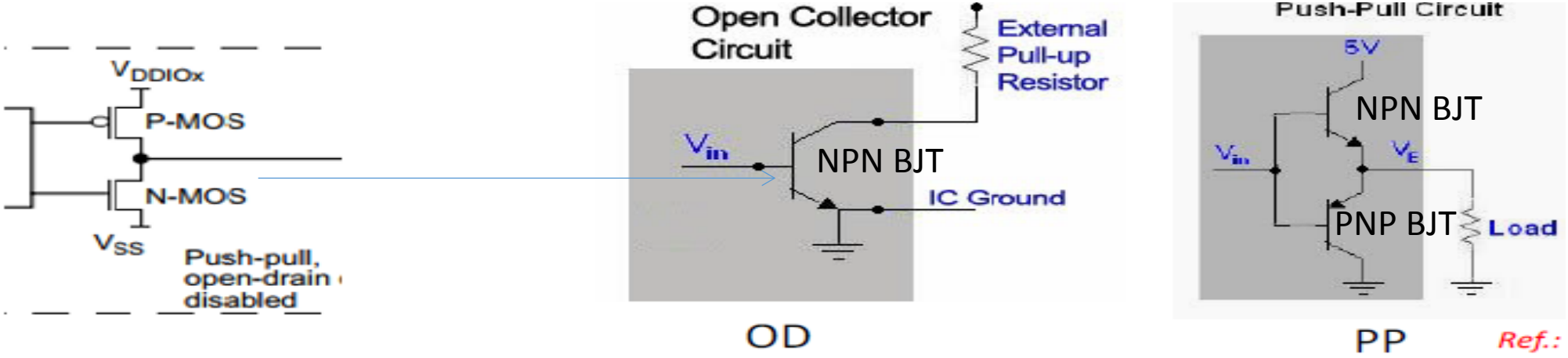
Open-Drain vs. Push-Pull

名稱	Open-drain (MOSFET) or open collector (BJT)	Push-Pull
電路架構		
輸出	輸出要加上拉電阻，否則是浮接。	輸出可為Hi或Low準位。
特點	1. 可做電壓轉換-Level shift 2. IC內部僅需很小的閘極驅動電流	1.可以吸電流。 2.可以灌電流。 3.輸出電壓由IC電源決定。

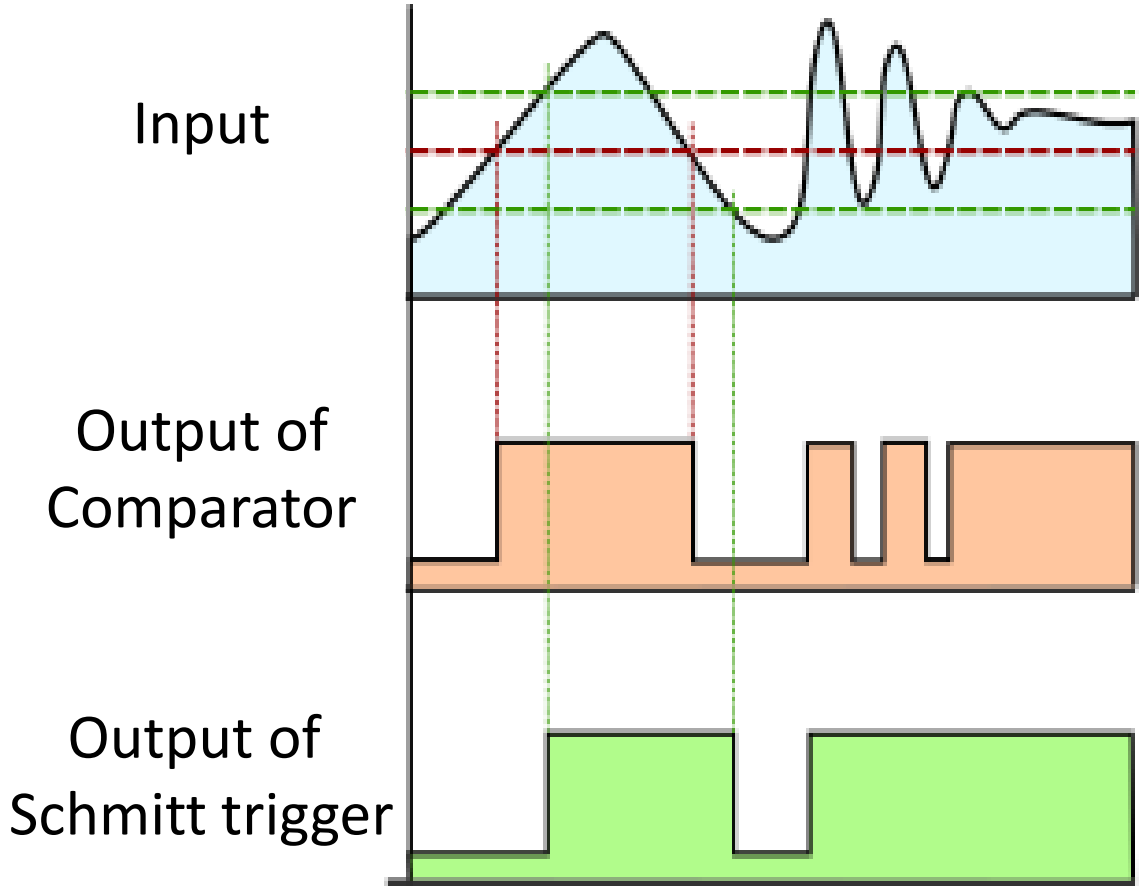
http://cary1120.blogspot.tw/2013/11/open-drain-push-pull.html

Push-Pull vs Open-Drain Output

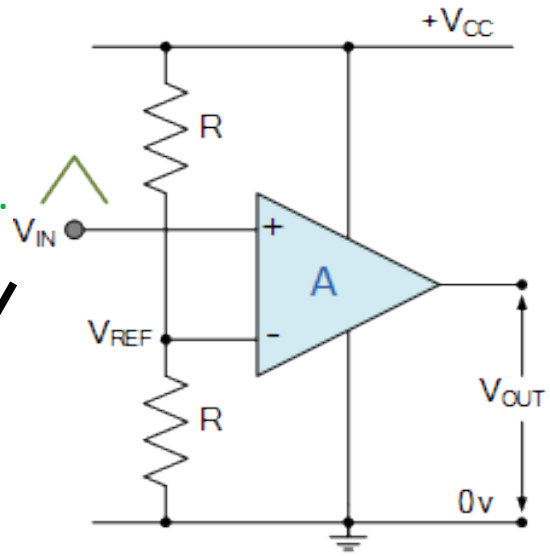
- Open-Drain
  - Output voltage level determine by external circuit
  - A “1” in the Vin activates the N-MOS/NPN-BJT whereas a “0” in the Vin leaves the port in Hi-Z (the P-MOS/PNP-BJT is never activated)
- Push-Pull
  - Output voltage level determine by internal Vdd_io
  - A “1” in the Vin activates the N-MOS/NPN-BJT whereas a “0” in the Vin activates the P-MOS/PNP-BJT

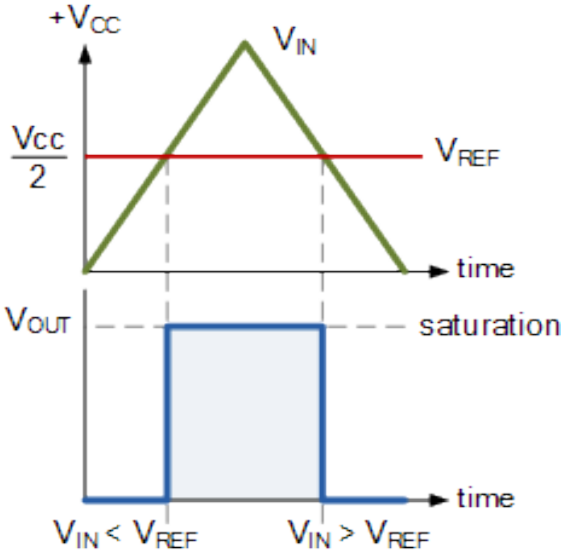


補充資料: Schmitt trigger vs. OP-AMP Comparator

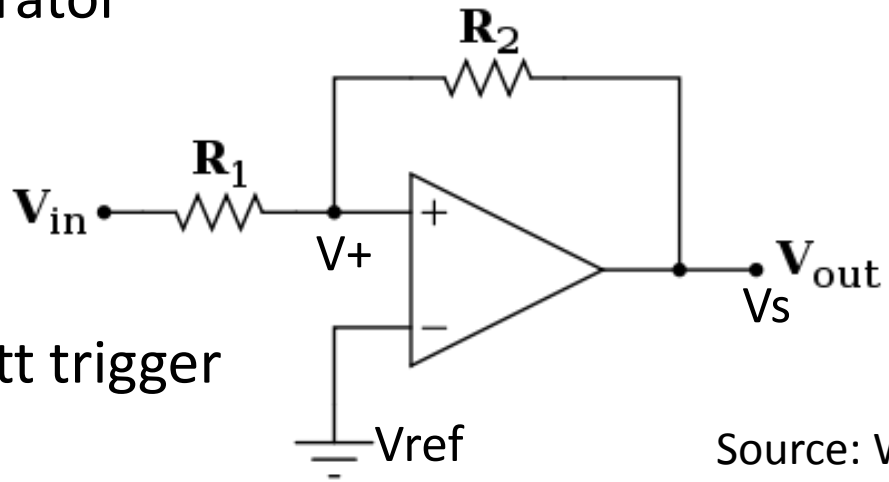


Comparator





Schmitt trigger



Source: Wikipedia

Schmitt trigger 運作原理

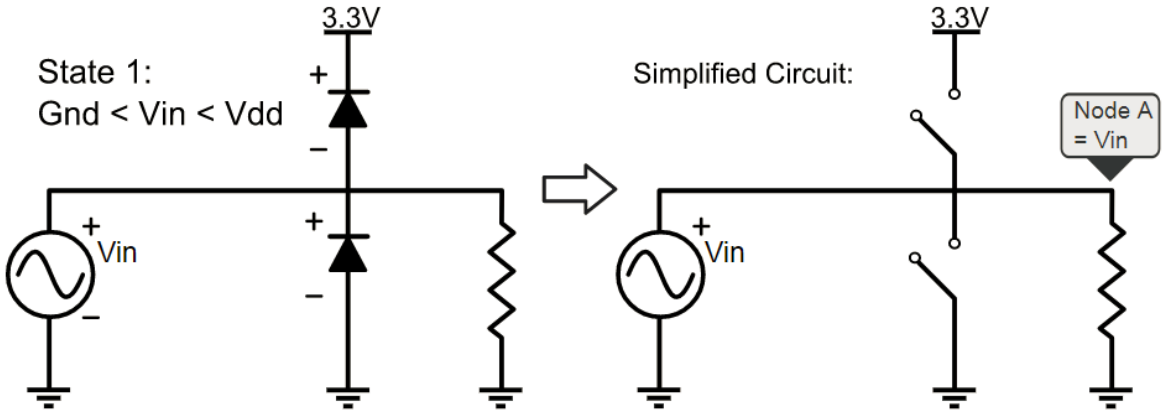
如果施密特觸發器的當前狀態是高電平，輸出會處於正電源軌（$+V_S$）上。這時V_+就會成為V_{in}和$+V_S$間的分壓器。在這種情況下，只有當$V_+=0$（接地）時，比較器才會翻轉到低電平。由電流守恆，可知此時滿足下列關係：$\frac{V_{\text{in}}}{R_1} = -\frac{V_S}{R_2}$

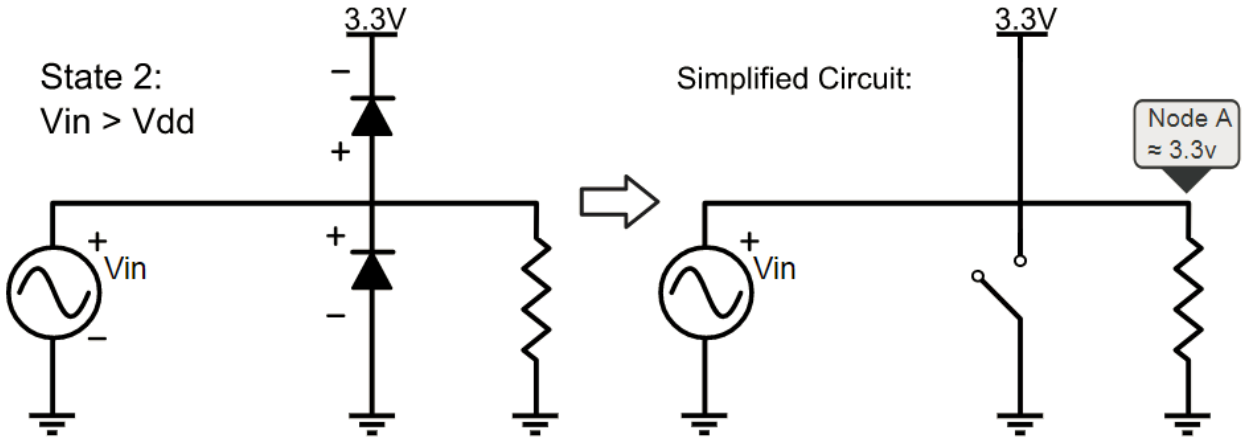
因此 必須降低到低於 $-\frac{R_1}{R_2}V_s$ 時，輸出才會翻轉狀態。一旦比較器的輸出翻轉到$-V_S$，翻轉回高電平的閾值就變成了 $+\frac{R_1}{R_2}V_s$

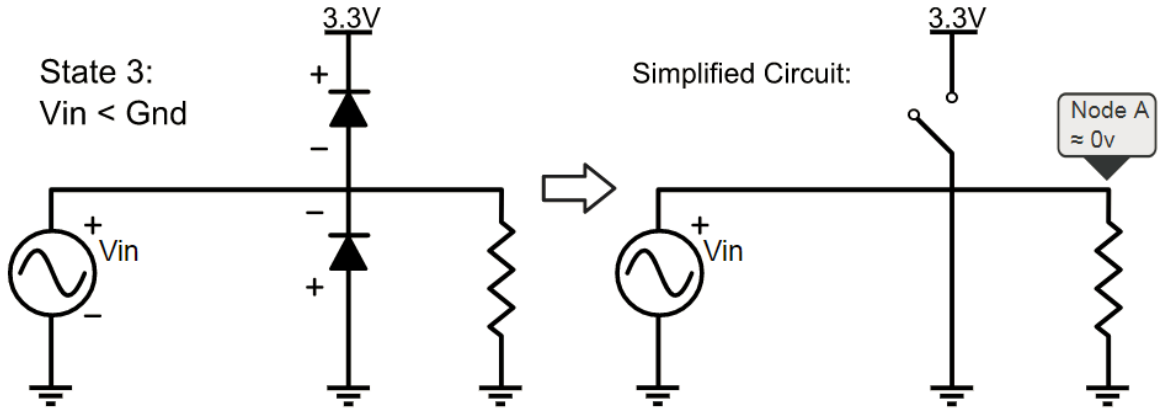
補充資料: Schmitt trigger vs. OP-AMP Comparator

為何Input port使用TTL Schmidt trigger?

補充資料: Protection Diode





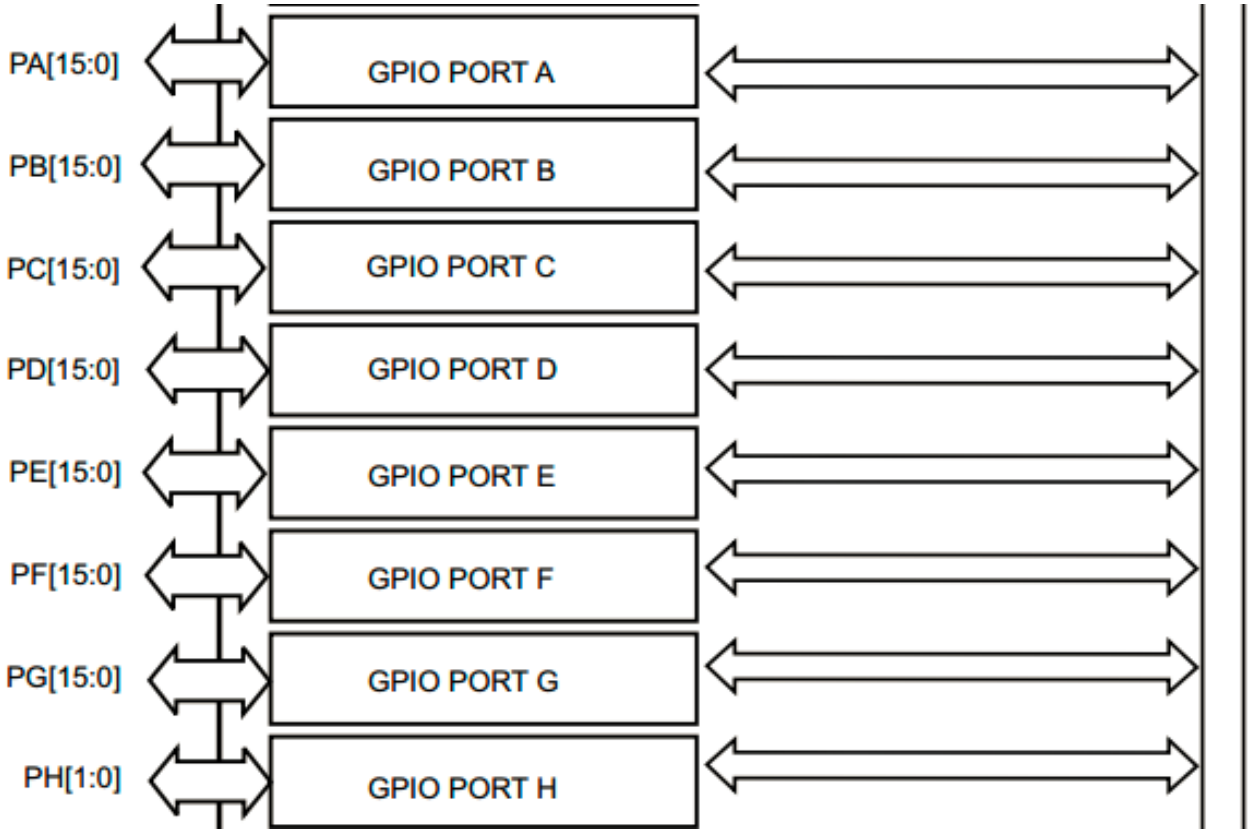


GPIO講解大綱

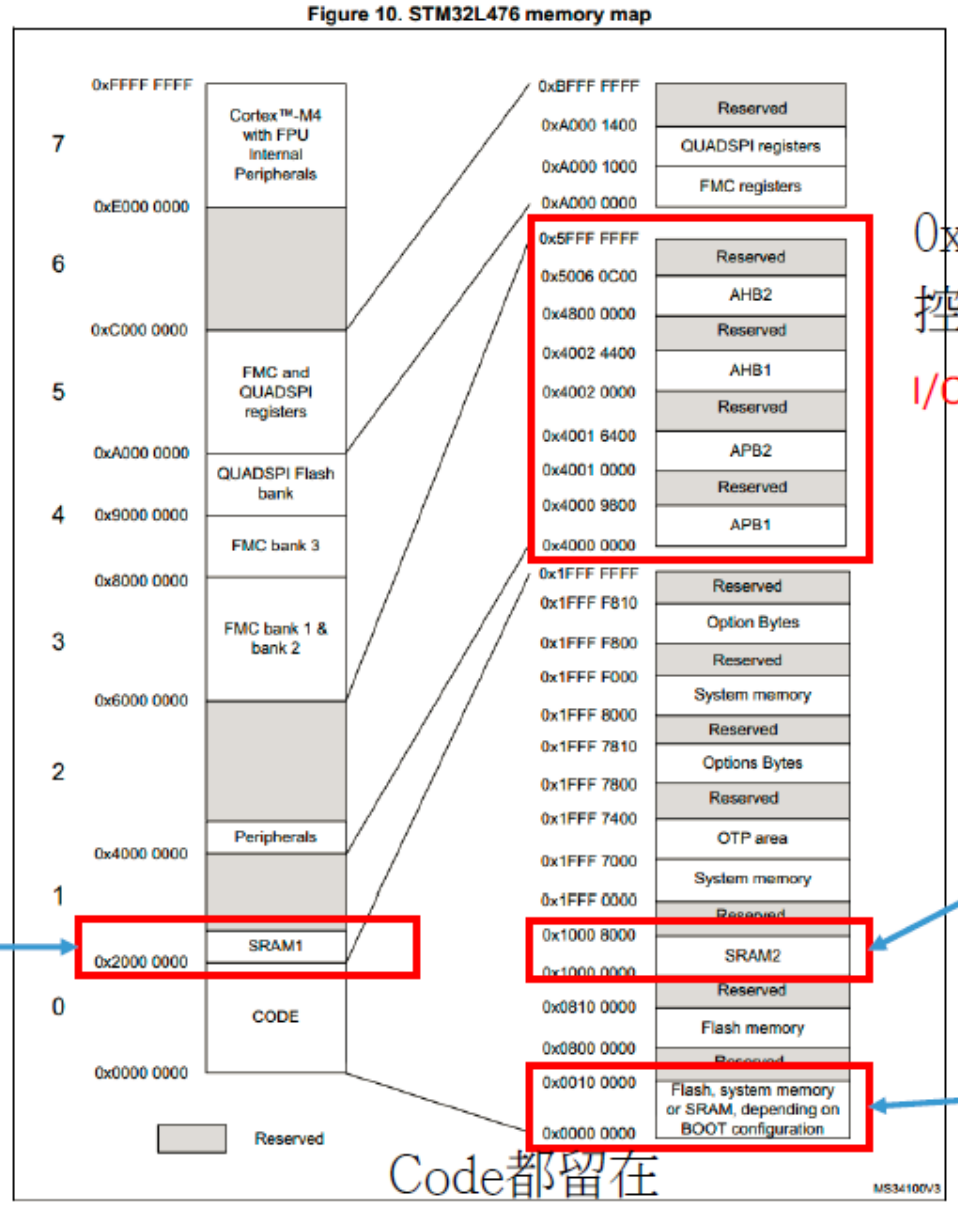
- GPIO 硬體架構與外接電路
- GPIO 組合語言控制register，register位址
  - Reference manual和programming manual定義reg和function (重要! 花十分鐘帶你看，以後要學會自己查)
  - 記憶體分配，GPIO register設定，程式範例
  - 補充資料-如何看register設定
- 輸出電路
- 範例程式，練習與作業

General-purpose inputs/outputs (GPIO)

- • STM32L476 have port A~H GPIO port connect on AHB2 bus
- • Except port H, each port have 16 pins
- • Our STM32L476RG chip can use
  - • **PA[0..15], PB[0..15], PC[0..15]**
  - • **PF[0..1], PF[4..7], PD2, PD8**



Memory Map



0x48000000 是GPIOA
控制I/O訊號用的記憶體

I/O memory-mapping area

Internal 92KB SRAM1

Internal 32KB SRAM2

Flash ROM-area

Code都留在
Flash ROM內執行

Reference manual STM32L4x6 IO周邊register操作相關參考文件 => P67 Figure 2. Memory map

GPIO Memory Address

- In STM32L4 system all GPIO port connect on AHB2 bus
- Port A system memory address start from ***0x48000000***

C語言指令:

char *a

a=0x4800 0000

*a = 1

=> 0x4800 0000 的輸出變成1

Table 1. STM32L4x6 memory map and peripheral register boundary addresses

Bus	Boundary address	Size (bytes)	Peripheral	Peripheral register map
AHB2	0x5006 0800 - 0x5006 0BFF	1 KB	RNG	*Section 24.4.4: RNG register map*
	0x5006 0400 - 0x5006 07FF	1 KB	Reserved	-
	0x5006 0000 - 0x5006 03FF	1 KB	AES	*Section 25.14.18: AES register map*
	0x5004 0400 - 0x5005 FFFF	127 KB	Reserved	-
	0x5004 0000 - 0x5004 03FF	1 KB	ADC	*Section 16.6.4: ADC register map*
	0x5000 0000 - 0x5003 FFFF	16 KB	OTG_FS	*Section 43.15.54: OTG_FS register map*
	0x4800 2000 - 0x4FFF FFFF	~127 MB	Reserved	-
	0x4800 1C00 - 0x4800 1FFF	1 KB	GPIOH	*Section 7.4.13: GPIO register map*
	0x4800 1800 - 0x4800 1BFF	1 KB	GPIOG	*Section 7.4.13: GPIO register map*
	0x4800 1400 - 0x4800 17FF	1 KB	GPIOF	*Section 7.4.13: GPIO register map*
	0x4800 1000 - 0x4800 13FF	1 KB	GPIOE	*Section 7.4.13: GPIO register map*
	0x4800 0C00 - 0x4800 0FFF	1 KB	GPIOD	*Section 7.4.13: GPIO register map*
	0x4800 0800 - 0x4800 0BFF	1 KB	GPIOC	*Section 7.4.13: GPIO register map*
	0x4800 0400 - 0x4800 07FF	1 KB	GPIOB	*Section 7.4.13: GPIO register map*
	0x4800 0000 - 0x4800 03FF	1 KB	GPIOA	*Section 7.4.13: GPIO register map*
	0x4002 4400 - 0x47FF FFFF	~127 MB	Reserved	-

ADC 在這裡

GPIOA 在這裡

GPIO Register

- Clock enable register
  - AHB2 peripheral clock enable register (RCC_AHB2ENR) 打開bus，GPIO才可使用
- Control registers 控制用
  - GPIO port mode register (GPIOx_MODER) (x =A..H) 輸出，輸入, AF…
  - GPIO port output type register (GPIOx_OTYPER) (x = A..H) pull up, drain…
  - GPIO port output speed register (GPIOx_OSPEEDR) 讀取速度
  - GPIO port pull-up/pull-down register (GPIOx_PUPDR) 輸出電路選擇
  - …
- Data registers 資料傳輸用
  - Output: GPIOx_ODR, 16bits
  - Input: GPIOx_IDR, 16bits

請同學們把這個記在印出的講義上

位址

ldr r1, =GPIOB_ODR

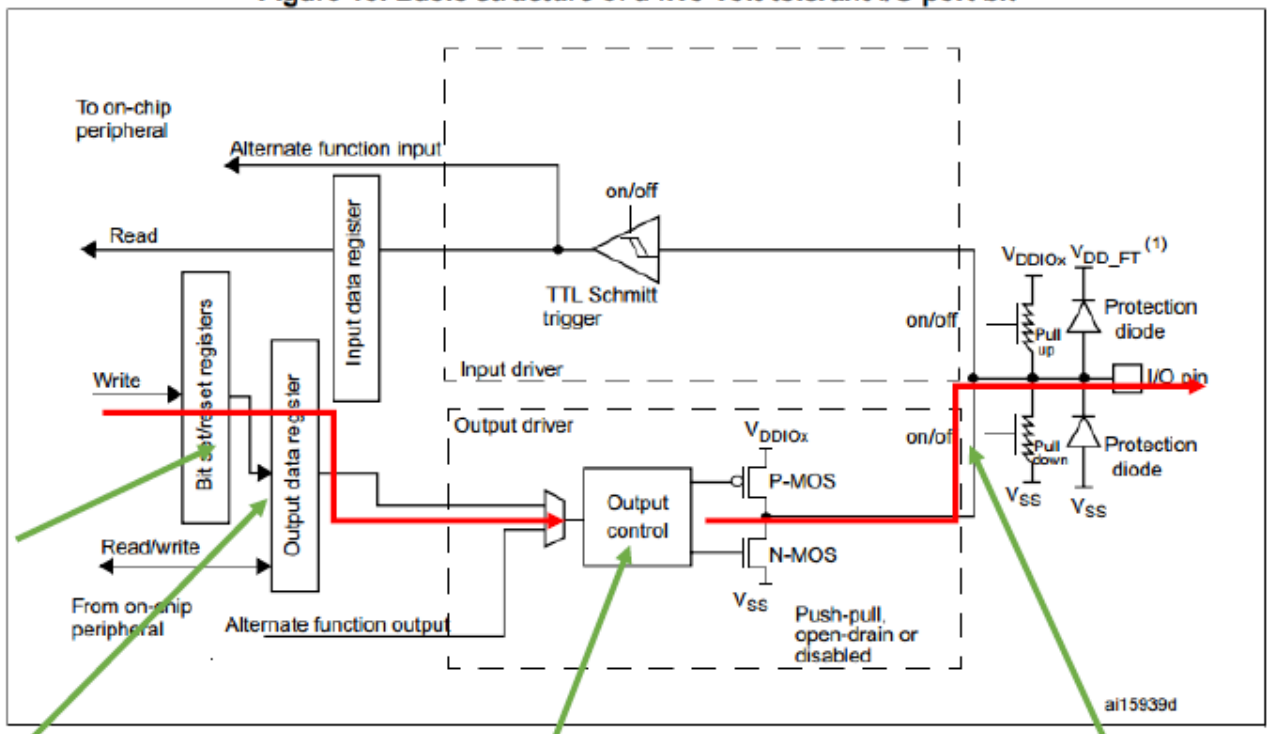
ldr r0, [r1]

and r0, #0xFFFFC03F

orr r0, #0x1540值

str r0, [r1]

Figure 18. Basic structure of a five-volt tolerant I/O port bit



1. V_{DD_FT} is a potential specific to five-volt tolerant I/Os and different from V_{DD}.

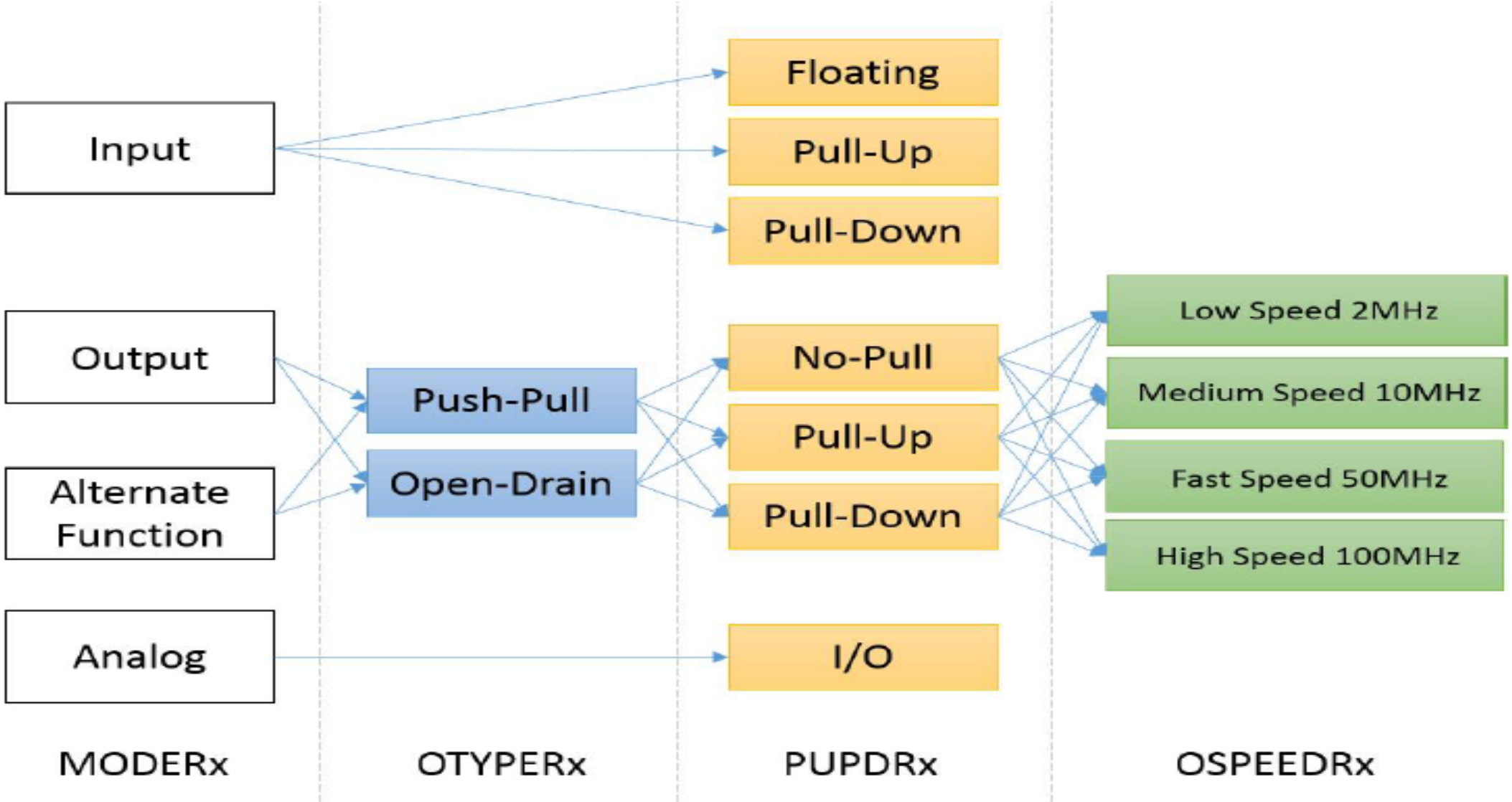
Bit set/reset register(BSRR)

Output data register(ODR)

Output type register(OTYPER)

Pull-up control register(PUPDR)

Configuration Reference



slide26 Configuration Reference印出供同學參考

Code Example

- Configure
  - Output
  - Pull-up
- Set PA pin 5 as output high

```
.syntax unified
.cpu cortex-m4
.thumb

.text
.global main
.equ RCC_AHB2ENR, 0x4002104C
.equ GPIOA_MODER, 0x48000000
.equ GPIOA_OTYPER, 0x48000004
.equ GPIOA_OSPEEDR, 0x48000008
.equ GPIOA_PUPDR, 0x4800000C
.equ GPIOA_ODR, 0x48000014

//LED on PA5
main:
    //Enable AHB2 clock
    movs    r0, #0x1
    ldr     r1, =RCC_AHB2ENR
    str     r0, [r1]

    //Set PA5 as output mode
    movs    r0, #0x400
    ldr     r1, =GPIOA_MODER
    ldr     r2, [r1]
    and     r2, #0xFFFFF3FF //Mask MODER5
    orrs    r2, r2, r0
    str     r2, [r1]

    //Default PA5 is Pull-up output, no need to set

    //Set PA5 as high speed mode
    movs    r0, #0x800
    ldr     r1, =GPIOA_OSPEEDR
    strh    r0, [r1]

    ldr     r1, =GPIOA_ODR
L1:
    movs    r0, #(1<<5)
    strh    r0, [r1]

    B L1
```

Memory mapped I/O register addresses

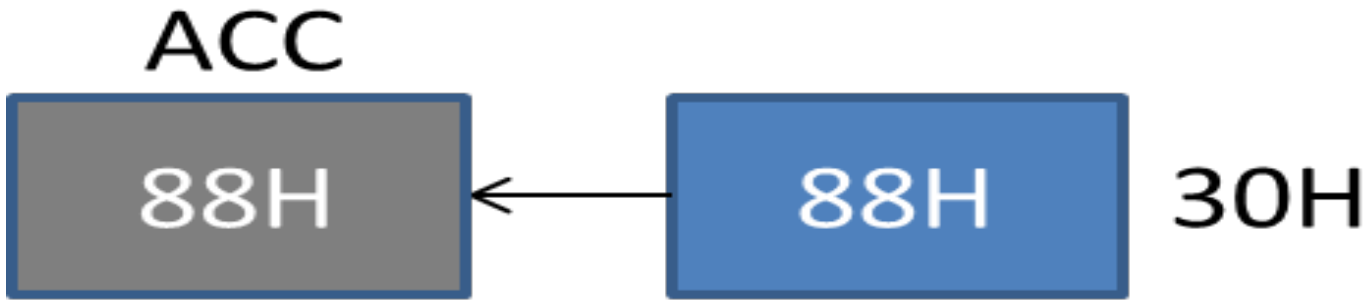
GPIOA_MODER = (GPIOA_MODER &0xFFFFF3FF) | 0x400

F3FF= 1111 0011 1111 1111 確認P5的位置是00

High speed 0000 1000 0000 0000

Why we need read this register value fist?
Ans: JTAG/SWD is use PA13,14 as debug port, can't modify its mode configuration

複習: 1直接定址法



- ex: MOV A,30H ;把位址 30H 的內容存入累加器A。所以A裡面是88H

```
ldr  r1, =X
ldr  r0, [r1]
```

```
movs r2, #AA
adds r2, r2, r0
str  r2, [r1]
```

複習: Register使用 []

- 若是 label 加上兩個中括號(例如：[L1])，表示所使用的為該記憶體空間中儲存的值

- Ex: 如何寫入值打開GPIOA?

```
ldr r1, =RCC_AHB2ENR
ldr r2, [r1]
movs r2, #1
str r2, [r1]
```

C語言指令:
char *a
a=0x4800 0000
*a = 1
=> 0x4800 0000 的儲存值變成1

http://godleon.blogspot.tw/2008/01/machine-language-cpu-machine-language.html


OSPEEDER: 0x08
OTYPER: 0x04
MODER: 0x00
GPIOC: 0x48000800

如何寫入值控制一個GPIO Port?

```
ldr r1, =RCC_AHB2ENR
ldr r2, [r1]
movs r2, #1
str r2, [r1]
```

```
movs    r0, #0x400
ldr     r1, =GPIOA_MODER
ldr     r2, [r1]
and     r2, #0xFFFFF3FF ;/Mask MODERS
orrs    r2, r2, r0
str     r2, [r1]
```

- 位址: 控制該GPIO功能的位址
- 值: 不同功能要寫入什麼值

位址: 控制該GPIO功能的位址

找GPIO的Register位址

- **Reference manual STM32**

Page68

Table 1. STM32L4x6 memory map and peripheral register boundary addresses

Table 1. STM32L4x6 memory map and peripheral register boundary addresses

Bus	Boundary address	Size (bytes)	Peripheral	Peripheral register map
AHB2	0x5006 0800 - 0x5006 0BFF	1 KB	RNG	*Section 24.4.4: RNG register map*
	0x5006 0400 - 0x5006 07FF	1 KB	Reserved	-
	0x5006 0000 - 0x5006 03FF	1 KB	AES	*Section 25.14.18: AES register map*
	0x5004 0400 - 0x5005 FFFF	127 KB	Reserved	-
	0x5004 0000 - 0x5004 03FF	1 KB	ADC	*Section 16.6.4: ADC register map*
	0x5000 0000 - 0x5003 FFFF	16 KB	OTG_FS	*Section 43.15.54: OTG_FS register map*
	0x4800 2000 - 0x4FFF FFFF	~127 MB	Reserved	-
	0x4800 1C00 - 0x4800 1FFF	1 KB	GPIOH	*Section 7.4.13: GPIO register map*
	0x4800 1800 - 0x4800 1BFF	1 KB	GPIOG	*Section 7.4.13: GPIO register map*
	0x4800 1400 - 0x4800 17FF	1 KB	GPIOF	*Section 7.4.13: GPIO register map*
	0x4800 1000 - 0x4800 13FF	1 KB	GPIOE	*Section 7.4.13: GPIO register map*
	0x4800 0C00 - 0x4800 0FFF	1 KB	GPIOD	*Section 7.4.13: GPIO register map*
	0x4800 0800 - 0x4800 0BFF	1 KB	GPIOC	*Section 7.4.13: GPIO register map*
	0x4800 0400 - 0x4800 07FF	1 KB	GPIOB	*Section 7.4.13: GPIO register map*
	0x4800 0000 - 0x4800 03FF	1 KB	GPIOA	*Section 7.4.13: GPIO register map*
	0x4002 4400 - 0x47FF FFFF	~127 MB	Reserved	-

7.4.13 GPIO register map

The following table gives the GPIO register map and reset values.

Table 33. GPIO register map and reset values

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	GPIOA_MODER	MODE15[1:0]		MODE14[1:0]		MODE13[1:0]		MODE12[1:0]		MODE11[1:0]		MODE10[1:0]		MODE9[1:0]		MODE8[1:0]		MODE7[1:0]		MODE6[1:0]		MODE5[1:0]		MODE4[1:0]		MODE3[1:0]		MODE2[1:0]		MODE1[1:0]		MODE0[1:0]	
	Reset value	1	0	1	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x00	GPIOB_MODER	MODER15[1:0]		MODER14[1:0]		MODER13[1:0]		MODER12[1:0]		MODER11[1:0]		MODER10[1:0]		MODER9[1:0]		MODER8[1:0]		MODER7[1:0]		MODER6[1:0]		MODER5[1:0]		MODER4[1:0]		MODER3[1:0]		MODER2[1:0]		MODER1[1:0]		MODER0[1:0]	
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	0	1	1	1	1	1	1
0x00	GPIOx_MODER (where x = C..H)	MODE15[1:0]		MODE14[1:0]		MODE13[1:0]		MODE12[1:0]		MODE11[1:0]		MODE10[1:0]		MODE9[1:0]		MODE8[1:0]		MODE7[1:0]		MODE6[1:0]		MODE5[1:0]		MODE4[1:0]		MODE3[1:0]		MODE2[1:0]		MODE1[1:0]		MODE0[1:0]	
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x04	GPIOx_OTYPER (where x = A..H)	Res.		Res.		Res.		Res.		Res.		Res.		Res.		Res.		OT15	OT14	OT13	OT12	OT11	OT10	OT9	OT8	OT7	OT6	OT5	OT4	OT3	OT2	OT1	OT0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	GPIOA_OSPEEDR	OSPEED15[1:0]		OSPEED14[1:0]		OSPEED13[1:0]		OSPEED12[1:0]		OSPEED11[1:0]		OSPEED10[1:0]		OSPEED9[1:0]		OSPEED8[1:0]		OSPEED7[1:0]		OSPEED6[1:0]		OSPEED5[1:0]		OSPEED4[1:0]		OSPEED3[1:0]		OSPEED2[1:0]		OSPEED1[1:0]		OSPEED0[1:0]	
	Reset value	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	GPIOx_OSPEEDR (where x = B..H)	OSPEED15[1:0]		OSPEED14[1:0]		OSPEED13[1:0]		OSPEED12[1:0]		OSPEED11[1:0]		OSPEED10[1:0]		OSPEED9[1:0]		OSPEED8[1:0]		OSPEED7[1:0]		OSPEED6[1:0]		OSPEED5[1:0]		OSPEED4[1:0]		OSPEED3[1:0]		OSPEED2[1:0]		OSPEED1[1:0]		OSPEED0[1:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	GPIOA_PUPDR	PUPD15[1:0]		PUPD14[1:0]		PUPD13[1:0]		PUPD12[1:0]		PUPD11[1:0]		PUPD10[1:0]		PUPD9[1:0]		PUPD8[1:0]		PUPD7[1:0]		PUPD6[1:0]		PUPD5[1:0]		PUPD4[1:0]		PUPD3[1:0]		PUPD2[1:0]		PUPD1[1:0]		PUPD0[1:0]	
	Reset value	0	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	GPIOB_PUPDR	PUPD15[1:0]		PUPD14[1:0]		PUPD13[1:0]		PUPD12[1:0]		PUPD11[1:0]		PUPD10[1:0]		PUPD9[1:0]		PUPD8[1:0]		PUPD7[1:0]		PUPD6[1:0]		PUPD5[1:0]		PUPD4[1:0]		PUPD3[1:0]		PUPD2[1:0]		PUPD1[1:0]		PUPD0[1:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
0x0C	GPIOx_PUPDR (where x = C..H	PUPD15[1:0]		PUPD14[1:0]		PUPD13[1:0]		PUPD12[1:0]		PUPD11[1:0]		PUPD10[1:0]		PUPD9[1:0]		PUPD8[1:0]		PUPD7[1:0]		PUPD6[1:0]		PUPD5[1:0]		PUPD4[1:0]		PUPD3[1:0]		PUPD2[1:0]		PUPD1[1:0]		PUPD0[1:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	GPIOx_IDR (where x = A..H)	Res.		Res.		Res.		Res.		Res.		Res.		Res.		Res.		ID15	ID14	ID13	ID12	ID11	ID10	ID9	ID8	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0
	Reset value																	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

Table 33. GPIO register map and reset values (continued)

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x14	GPIOx_ODR (where x = A..H)	Res.		Res.		Res.		Res.		Res.		Res.		Res.		Res.		OD15	OD14	OD13	OD12	OD11	OD10	OD9	OD8	OD7	OD6	OD5	OD4	OD3	OD2	OD1	OD0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	GPIOx_BSRR (where x = A..H)	BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0	BS15	BS14	BS13	BS12	BS11	BS10	BS9	BS8	BS7	BS6	BS5	BS4	BS3	BS2	BS1	BS0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	GPIOx_LCKR (where x = A..H)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LCKK	LCK15	LCK14	LCK13	LCK12	LCK11	LCK10	LCK9	LCK8	LCK7	LCK6	LCK5	LCK4	LCK3	LCK2	LCK1	LCK0
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	GPIOx_AFRL (where x = A..H)	AFSEL7[3:0]				AFSEL6[3:0]				AFSEL5[3:0]				AFSEL4[3:0]				AFSEL3[3:0]				AFSEL2[3:0]				AFSEL1[3:0]				AFSEL0[3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x24	GPIOx_AFRH (where x = A..H)	AFSEL15[3:0]				AFSEL14[3:0]				AFSEL13[3:0]				AFSEL12[3:0]				AFSEL11[3:0]				AFSEL10[3:0]				AFSEL9[3:0]				AFSEL8[3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	GPIOx_BRR (where x = A..H)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	GPIOx_ASCR (where x = A..H)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ASC15	ASC14	ASC13	ASC12	ASC11	ASC10	ASC9	ASC8	ASC7	ASC6	ASC5	ASC4	ASC3	ASC2	ASC1	ASC0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Port 的address, data, mode等配置都要查manual
在我們Reference manual STM32裡面是一個port先疊完各設定(e.g. MODER) ，再下一個port 。然後下一個port
PortA MODER ...OTYPER ...SPEEDR...
PortB MODER...

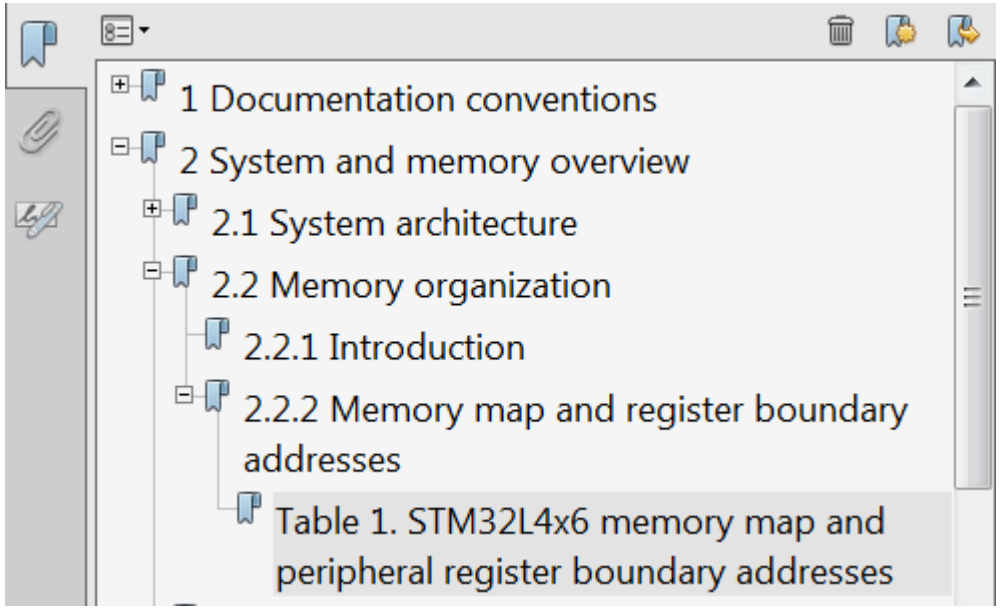
看一下pdf p272 GPIO register map

Offset就是起始值再加上多少，例如GPIOB_OTYPER就是0x4800 0400 (GPIOB, 見P9 GPIO Memory Address) 再加上 0x04 (OTYPER) 請大家找看看。 練習: 找出GPIOC_OSPEEDR的位址，加分。 然後用pin.s找。


OSPEEDER: 0x08
OTYPER: 0x04
MODER: 0x00
GPIOC: 0x48000800

找 RCC_AHB2ENR位址:

- Reference manual STM32L4x6 IO周邊register操作相關參考文件.pdf

→ 

pin.s | led_button.h | stm32l476x

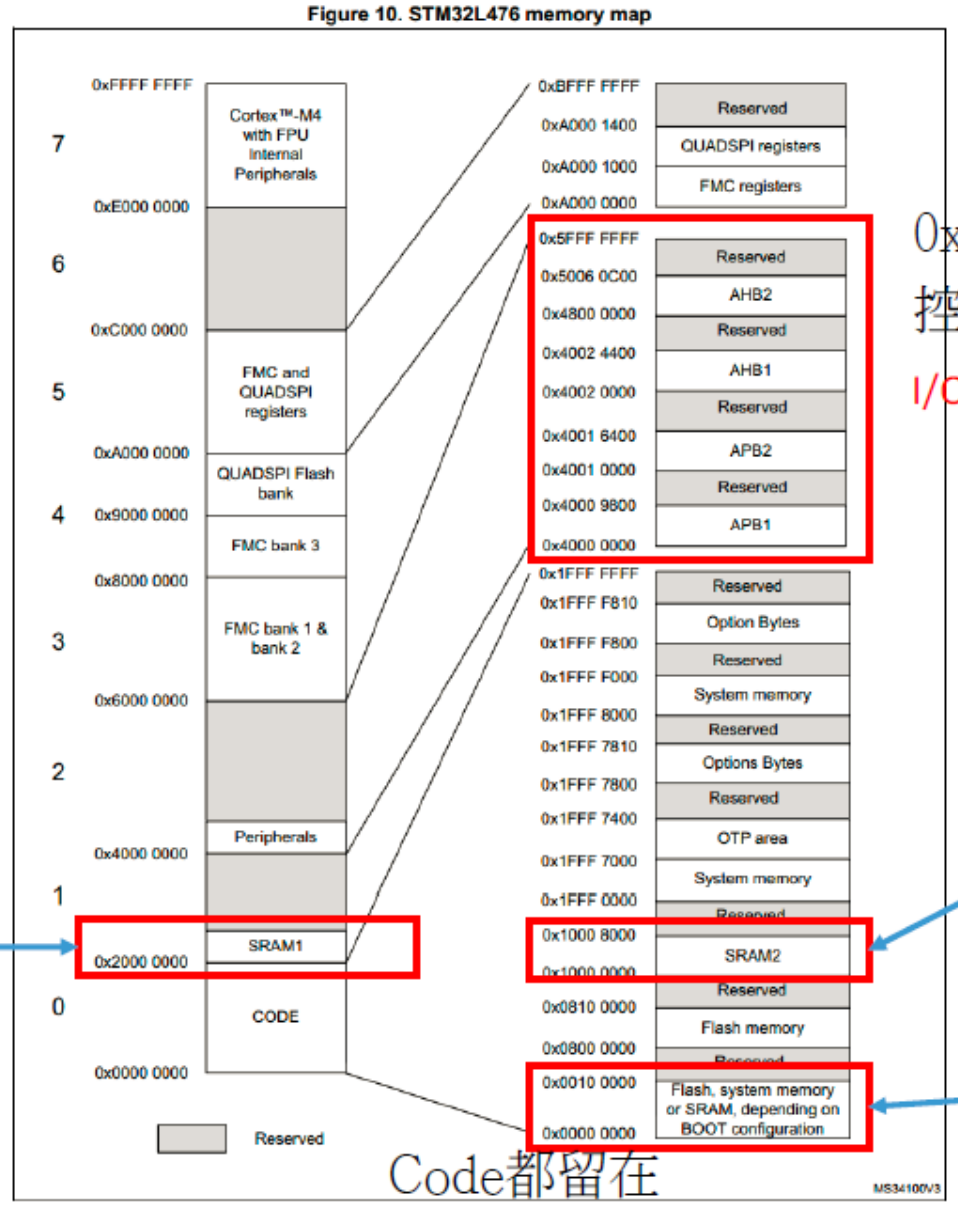
```
118 .equ RCC_APB1RSTR2, 0x4002103C
119 .equ RCC_APB2RSTR,  0x40021040
120 .equ RCC_AHB1ENR,   0x40021048
121 .equ RCC_AHB2ENR,   0x4002104C
```

→ GPIO*EN在RCC_AHB1裡面，點Section 6.4.31: RCC register map Page253 →找到RCC_AHB2ENR是0x4C (另外GPIOA~H在AHB2裡面Page68，不一樣)

→而RCC在AHB1裡面 Page69，找到RCC範圍是0x4002 1000 - 0x4002 13FF

→ 所以RCC_AHB2ENR位址是0x4002 1000 + 0x4C = 0x4002 104C

Memory Map



0x48000000 是GPIOA
控制I/O訊號用的記憶體

I/O memory-mapping area

Internal 92KB SRAM1

Internal 32KB SRAM2

Flash ROM-area

Code都留在
Flash ROM內執行

Reference manual STM32L4x6 IO周邊register操作相關參考文件 => P67 Figure 2. Memory map

AHB1	0x4002 4000 - 0x4002 43FF	1 KB	TSC	*Section 23.6.11: TSC register map*
	0x4002 3400 - 0x4002 3FFF	1 KB	Reserved	-
	0x4002 3000 - 0x4002 33FF	1 KB	CRC	*Section 13.4.6: CRC register map*
	0x4002 2400 - 0x4002 2FFF	3 KB	Reserved	-
	0x4002 2000 - 0x4002 23FF	1 KB	FLASH registers	*Section 3.7.17: FLASH register map*
	0x4002 1400 - 0x4002 1FFF	3 KB	Reserved	-
	0x4002 1000 - 0x4002 13FF	1 KB	RCC	*Section 6.4.31: RCC register map*
	0x4002 0800 - 0x4002 0FFF	2 KB	Reserved	-

設定GPIO的Enable

0x4C	RCC_AHB2 ENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RNGEN	Res.	AESEN	Res.	Res.	ADCEN	OTGFSEN	Res.	Res.	Res.	Res.	GPIOHEN	GPIOGEN	GPIOFEN	GPIOEEN	GPIODEN	GPIOCEN	GPIOBEN	GPIOAEN
	Reset value														0		0			0	0					0	0	0	0	0	0	0	0

0000 0110

=0x06

```
    // Enable AHB2 clock
    movs r0, #0x6
    ldr r1, =RCC_AHB2ENR
    str r0, [r1]
```

GPIO IO Setting 値Value

Reference *manual STM32* *P257*

Table 32. Port bit configuration table(1)

MODE(i) [1:0]	OTYPER(i)	OSPEED(i) [1:0]		PUPD(i) [1:0]		I/O configuration	
01	0	SPEED [1:0]		0	0	GP output	PP
	0			0	1	GP output	PP + PU
	0			1	0	GP output	PP + PD
	0			1	1	Reserved	
	1			0	0	GP output	OD
	1			0	1	GP output	OD + PU
	1			1	0	GP output	OD + PD
	1			1	1	Reserved (GP output OD)	
10	0	SPEED [1:0]		0	0	AF	PP
	0			0	1	AF	PP + PU
	0			1	0	AF	PP + PD
	0			1	1	Reserved	
	1			0	0	AF	OD
	1			0	1	AF	OD + PU
	1			1	0	AF	OD + PD
	1			1	1	Reserved	
00	x	x	x	0	0	Input	Floating
	x	x	x	0	1	Input	PU
	x	x	x	1	0	Input	PD
	x	x	x	1	1	Reserved (input floating)	
11	x	x	x	0	0	Input/output	Analog
	x	x	x	0	1	Reserved	
	x	x	x	1	0		
	x	x	x	1	1		

1. GP = general-purpose, PP = push-pull, PU = pull-up, PD = pull-down, OD = open-drain, AF = alternate function.

值: 不同功能要寫入對應的值

RCC_AHB2ENR

- Use for enable clock of GPIO bus

總開關

6.4.17 AHB2 peripheral clock enable register (RCC_AHB2ENR)

Address offset: 0x4C

Reset value: 0x0000 0000

Access: no wait state, word, half-word and byte access

Note: When the peripheral clock is not active, the peripheral registers read or write access is not supported.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RNG EN	Res.	AESEN
													rw		rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	ADCEN	OTGFS EN	Res.	Res.	Res.	Res.	GPIOH EN	GPIOG EN	GPIOF EN	GPIOE EN	GPIOD EN	GPIOC EN	GPIOB EN	GPIOA EN
		rw	rw					rw	rw	rw	rw	rw	rw	rw	rw

設定GPIOB_MODER

Reference *manual STM32* *P266*

Port IO mode

7.4.1 GPIO port mode register (GPIOx_MODER) (x =A..H)

Address offset:0x00

重開機時的default值

Reset values:

- 0xABFF FFFF for port A
- 0xFFFF FEBF for port B
- 0xFFFF FFFF for ports C..G,
- 0x0000 000F for port H

不作任何更動時的default值

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MODE15[1:0]		MODE14[1:0]		MODE13[1:0]		MODE12[1:0]		MODE11[1:0]		MODE10[1:0]		MODE9[1:0]		MODE8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MODE7[1:0]		MODE6[1:0]		MODE5[1:0]		MODE4[1:0]		MODE3[1:0]		MODE2[1:0]		MODE1[1:0]		MODE0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 2y+1:2y **MODEy[1:0]:** Port x configuration bits (y = 0..15)

These bits are written by software to configure the I/O mode.

00: Input mode

01: General purpose output mode

10: Alternate function mode

11: Analog mode (reset state)

一個port有16個pin

Pin5=MODE5, P5 output就是

0000 0100 0000 0000 = 0x0400

設定GPIOB_MODER

Reference *manual STM32* ***P257***

```
38      // Set PB3 ~ PB6 to output mode
39      ldr r1, =GPIOB_MODER
40      ldr r0, [r1]
41      and r0, #0xFFFFC03F
42      orr r0, #0x1540
43      str r0, [r1]
```

0xC03F = 1100 0000 0011 1111

0x1540 = 0001 0101 0100 0000

7 6 5 4 3 2 1 0

PB6 ~ PB3

設定GPIOB_OTYPER

Reference *manual STM32* *P265*

7.4.2 GPIO port output type register (GPIOx_OTYPER) (x = A..H)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
OT15	OT14	OT13	OT12	OT11	OT10	OT9	OT8	OT7	OT6	OT5	OT4	OT3	OT2	OT1	OT0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

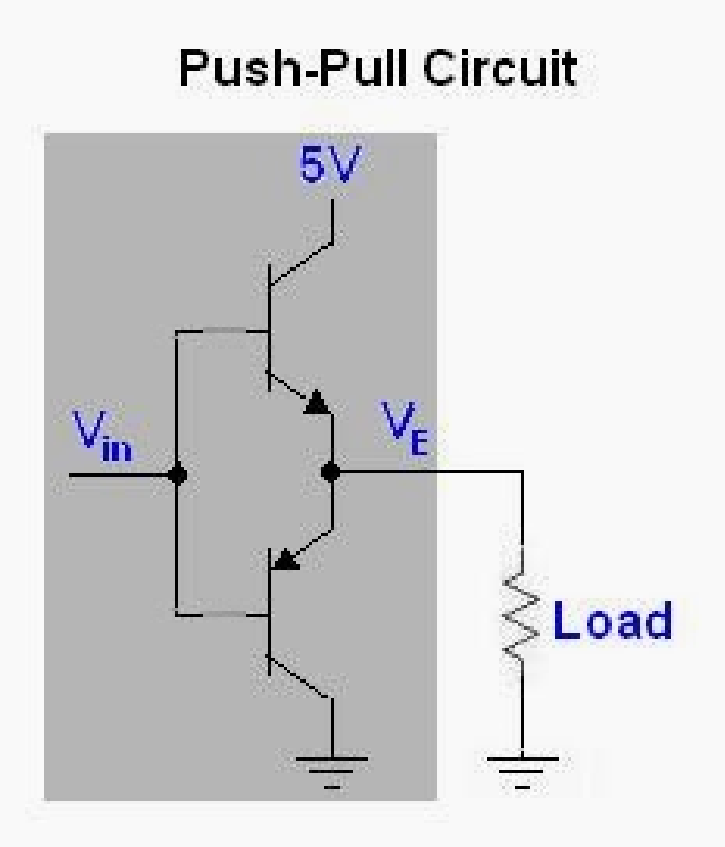
Bits 15:0 **OTy:** Port x configuration bits (y = 0..15)

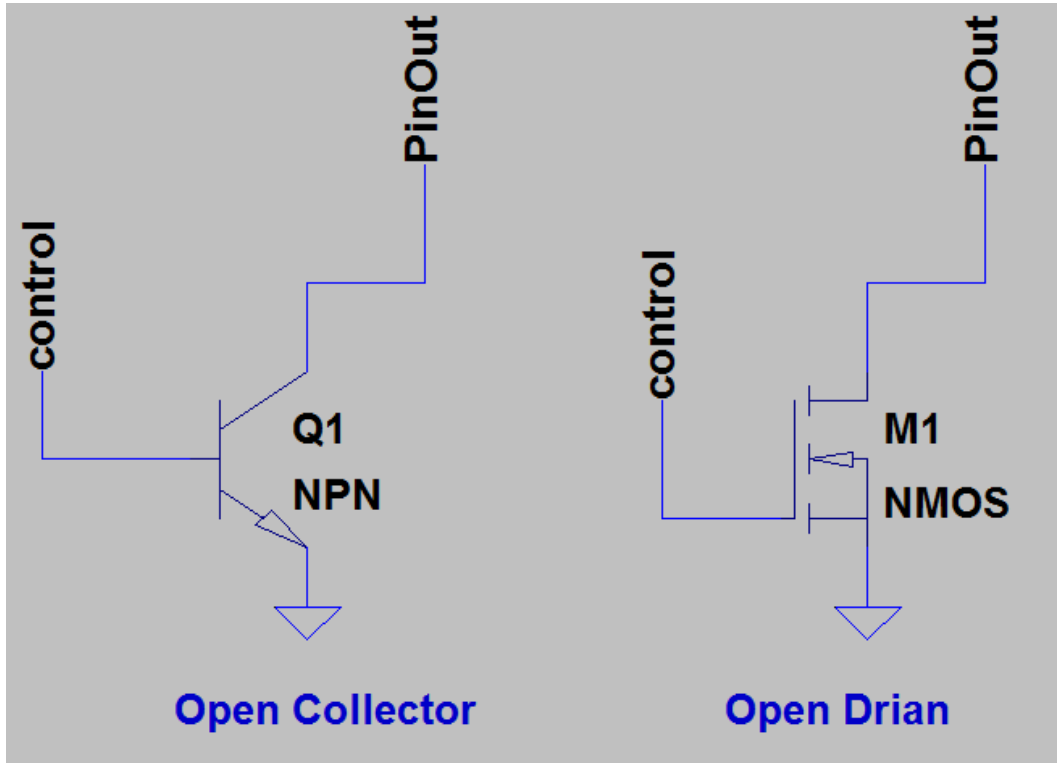
These bits are written by software to configure the I/O output type.

0: Output push-pull (reset state)

1: Output open-drain

Push-Pull v.s. Open-Drain





Current sink & current pull

Only current sink, no current pull

設定GPIOB_OSPEEDR

Reference *manual STM32* *P266*

7.4.3 GPIO port output speed register (GPIOx_OSPEEDR) (x = A..H)

Address offset: 0x08

Reset value:

- 0x0C00 0000 for port A
- 0x0000 0000 for the other ports

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OSPEED15 [1:0]		OSPEED14 [1:0]		OSPEED13 [1:0]		OSPEED12 [1:0]		OSPEED11 [1:0]		OSPEED10 [1:0]		OSPEED9 [1:0]		OSPEED8 [1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OSPEED7 [1:0]		OSPEED6 [1:0]		OSPEED5 [1:0]		OSPEED4 [1:0]		OSPEED3 [1:0]		OSPEED2 [1:0]		OSPEED1 [1:0]		OSPEED0 [1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 2y+1:2y **OSPEEDy[1:0]**: Port x configuration bits (y = 0..15)

These bits are written by software to configure the I/O output speed.

00: Low speed

01: Medium speed

10: High speed

11: Very high speed

Note: Refer to the device datasheet for the frequency specifications and the power supply and load conditions for each speed.

設定GPIOB_OSPEEDER

Reference *manual STM32* ***P257***

```
44      // Set PB3 ~ PB6 to high speed
45      ldr r1, =GPIOB_OSPEEDR
46      ldr r0, [r1]
47      and r0, #0xFFFFC03F
48      orr r0, #0x2A80
49      str r0, [r1]
```

0xC03F = 1100 0000 0011 1111

0x2A80 = 0010 1010 1000 0000

7 6 5 4 3 2 1 0

PB6 ~ PB3

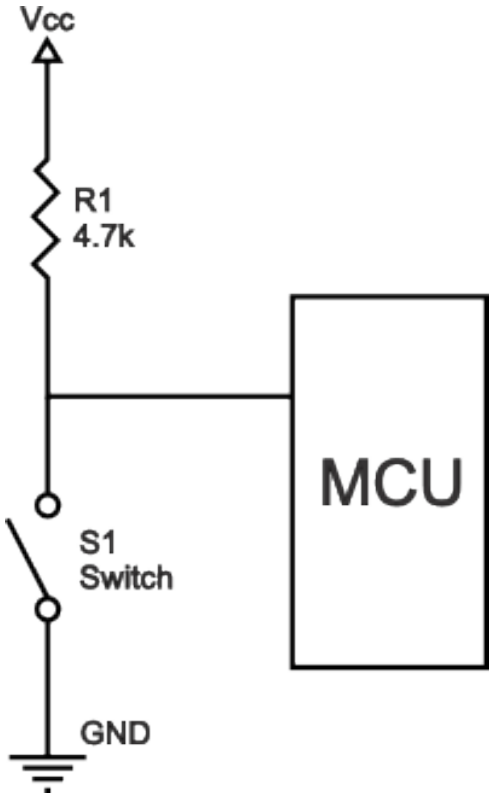
設定GPIOB_PUPDR

Question: 如何設定pin是pullup?

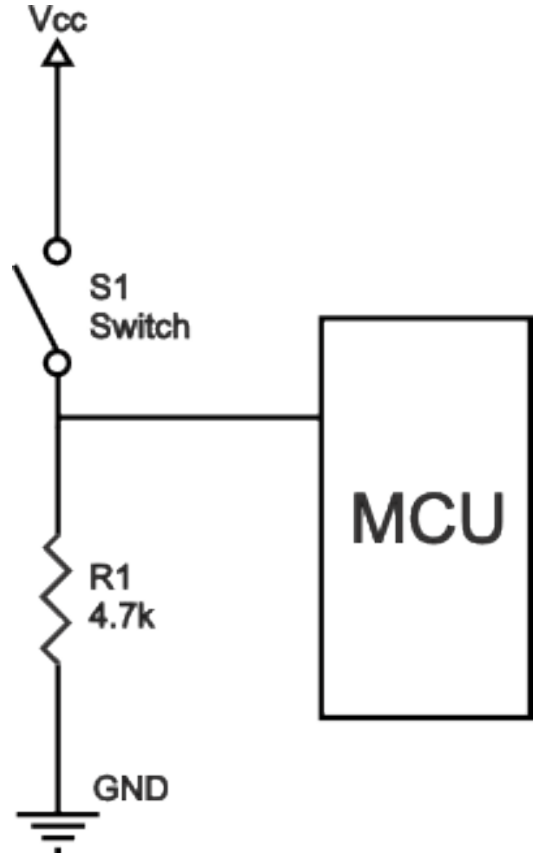
Reference *manual STM32* ***P???***

```
50      // Set PB3 ~ PB6 to pullup
51      ldr r1, =GPIOB_PUPDR
52      ldr r0, [r1]
53      and r0, #0xFFFFC03F
54      orr r0, ??????
55      str r0, [r1]
```

Pull up Circuit vs Pull down Circuit



Pull up input



Pull down input

設定GPIOB_PUPDR

Reference *manual STM32* *P266*

7.4.4 GPIO port pull-up/pull-down register (GPIOx_PUPDR) (x = A..H)

Address offset: 0x0C

Reset values:

- 0x6400 0000 for port A
- 0x0000 0100 for port B
- 0x0000 0000 for other ports

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PUPD15[1:0]		PUPD14[1:0]		PUPD13[1:0]		PUPD12[1:0]		PUPD11[1:0]		PUPD10[1:0]		PUPD9[1:0]		PUPD8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PUPD7[1:0]		PUPD6[1:0]		PUPD5[1:0]		PUPD4[1:0]		PUPD3[1:0]		PUPD2[1:0]		PUPD1[1:0]		PUPD0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 2y+1:2y **PUPDy[1:0]:** Port x configuration bits (y = 0..15)

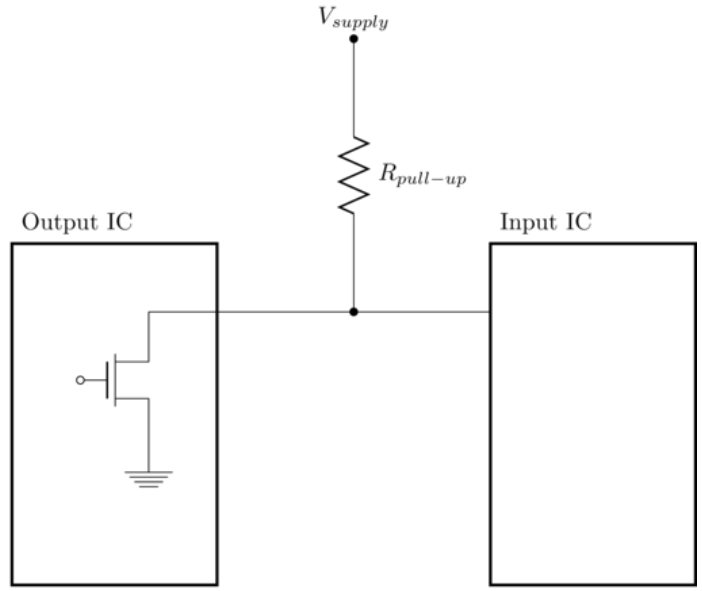
These bits are written by software to configure the I/O pull-up or pull-down

00: No pull-up, pull-down

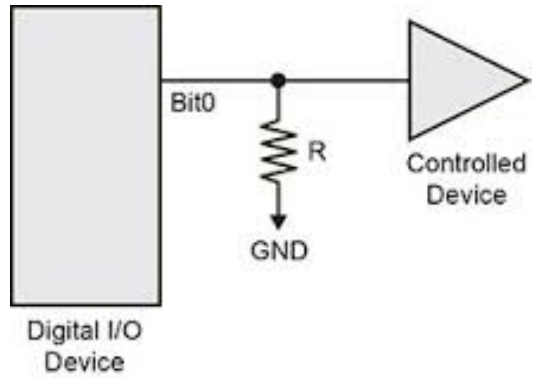
01: Pull-up

10: Pull-down

11: Reserved



Pull up output

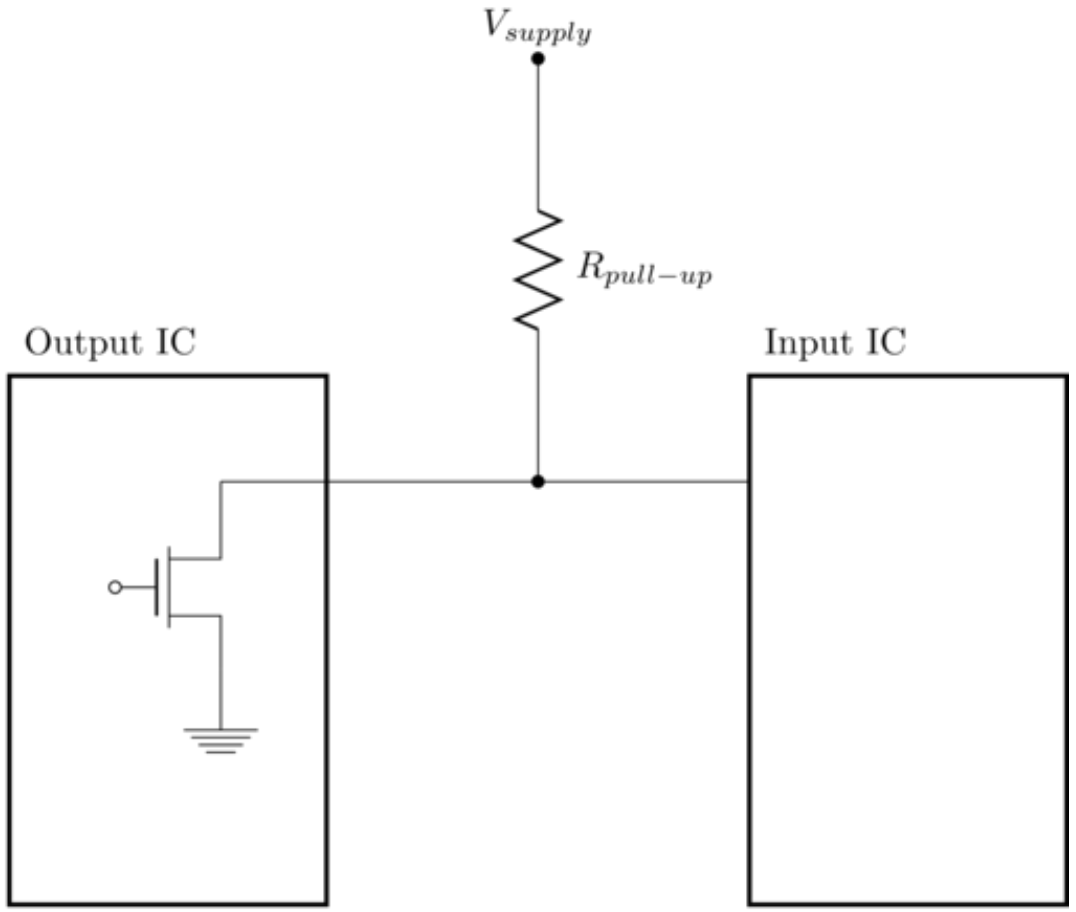


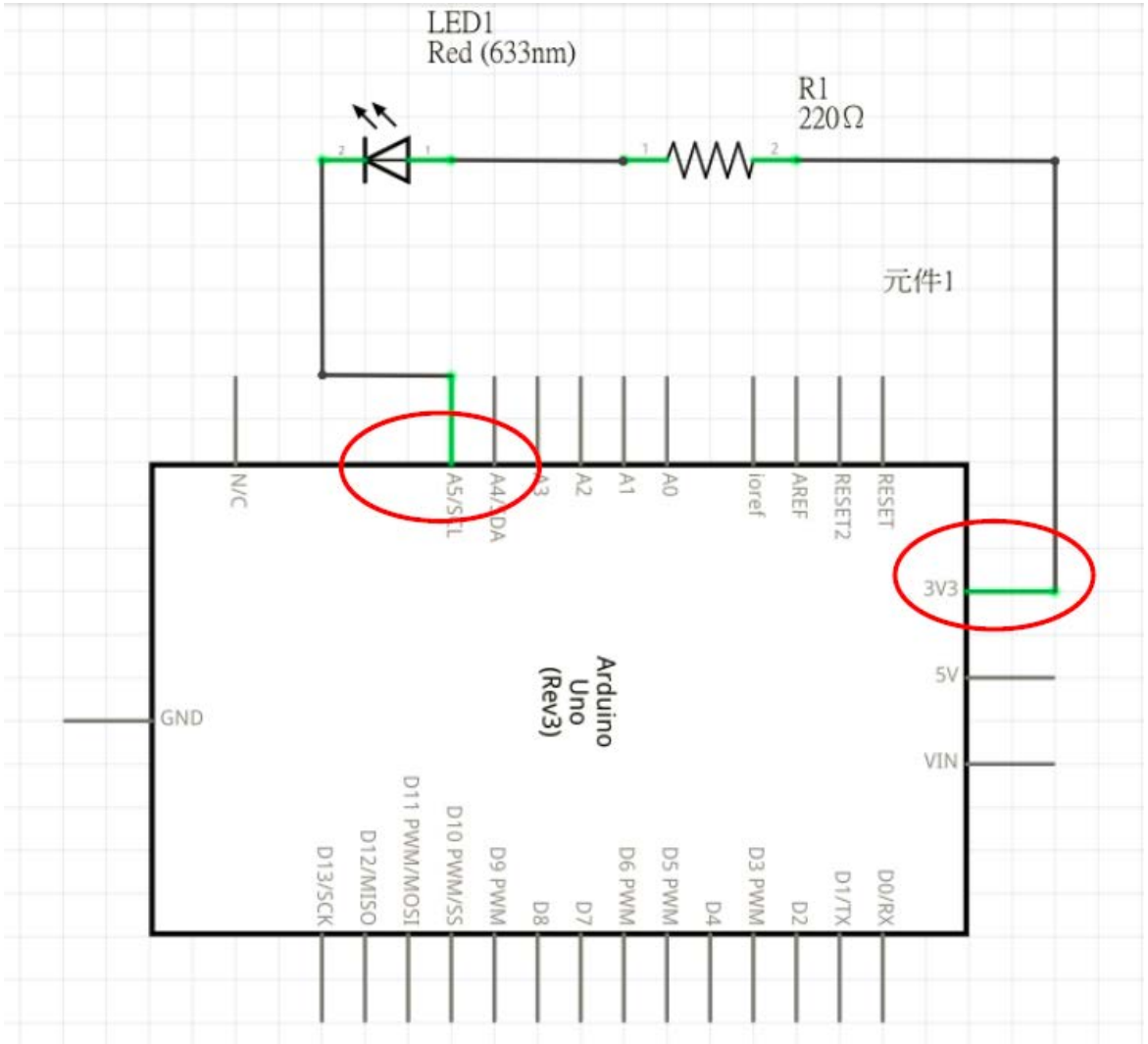
Pull-Down Current Sourcing = $\frac{5\ V}{R}$

Pull down output

http://ben.artins.org/electronics/pullup-resistors/

Pull-up Output Circuit for LED Control





設定GPIOB_PUPDR

Reference *manual STM32* ***P???***

```
50      // Set PB3 ~ PB6 to pullup
51      ldr r1, =GPIOB_PUPDR
52      ldr r0, [r1]
53      and r0, #0xFFFFC03F
54      orr r0, #0x1540
55      str r0, [r1]
```

0xC03F = 1100 0000 0011 1111

0x1540 = 0001 0101 0100 0000

7 6 5 4 3 2 1 0

PB6 ~ PB3

GPIO IO Setting

Reference *manual STM32* *P257*

Output setting

Input setting

Table 32. Port bit configuration table[1]

MODE(i) [1:0]	OTYPER(i)	OSPEED(i) [1:0]		PUPD(i) [1:0]		I/O configuration	
01	0	SPEED [1:0]		0	0	GP output	PP
	0			0	1	GP output	PP + PU
	0			1	0	GP output	PP + PD
	0			1	1	Reserved	
	1			0	0	GP output	OD
	1			0	1	GP output	OD + PU
	1			1	0	GP output	OD + PD
	1			1	1	Reserved (GP output OD)	
10	0	SPEED [1:0]		0	0	AF	PP
	0			0	1	AF	PP + PU
	0			1	0	AF	PP + PD
	0			1	1	Reserved	
	1			0	0	AF	OD
	1			0	1	AF	OD + PU
	1			1	0	AF	OD + PD
	1			1	1	Reserved	
00	x	x	x	0	0	Input	Floating
	x	x	x	0	1	Input	PU
	x	x	x	1	0	Input	PD
	x	x	x	1	1	Reserved (input floating)	
11	x	x	x	0	0	Input/output	Analog
	x	x	x	0	1	Reserved	
	x	x	x	1	0		
	x	x	x	1	1		

1. GP = general-purpose, PP = push-pull, PU = pull-up, PD = pull-down, OD = open-drain, AF = alternate function.

設定GPIOC_MODER

Question: 如何設定GPIOC P13是input?

値=?

位址=?

Reference *manual STM32* ***P257***

```
56      // Set PC13 to input mode
57      ldr r1, =GPIOC_MODER
58      ldr r0, [r1]
59      and r0,
60      orr r0,
61      str r0, [r1]
```

設定GPIOC_MODER

Reference *manual STM32* ***P???***

```
56      // Set PC13 to input mode
57      ldr r1, =GPIOC_MODER
58      ldr r0, [r1]
59      and r0, #0xF3FFFFFF
60      orr r0, #0x0
61      str r0, [r1]
```

0xF3FF = 1111 0011 1111 1111

0x00 = 0000 0000 0000 0000

1514 13

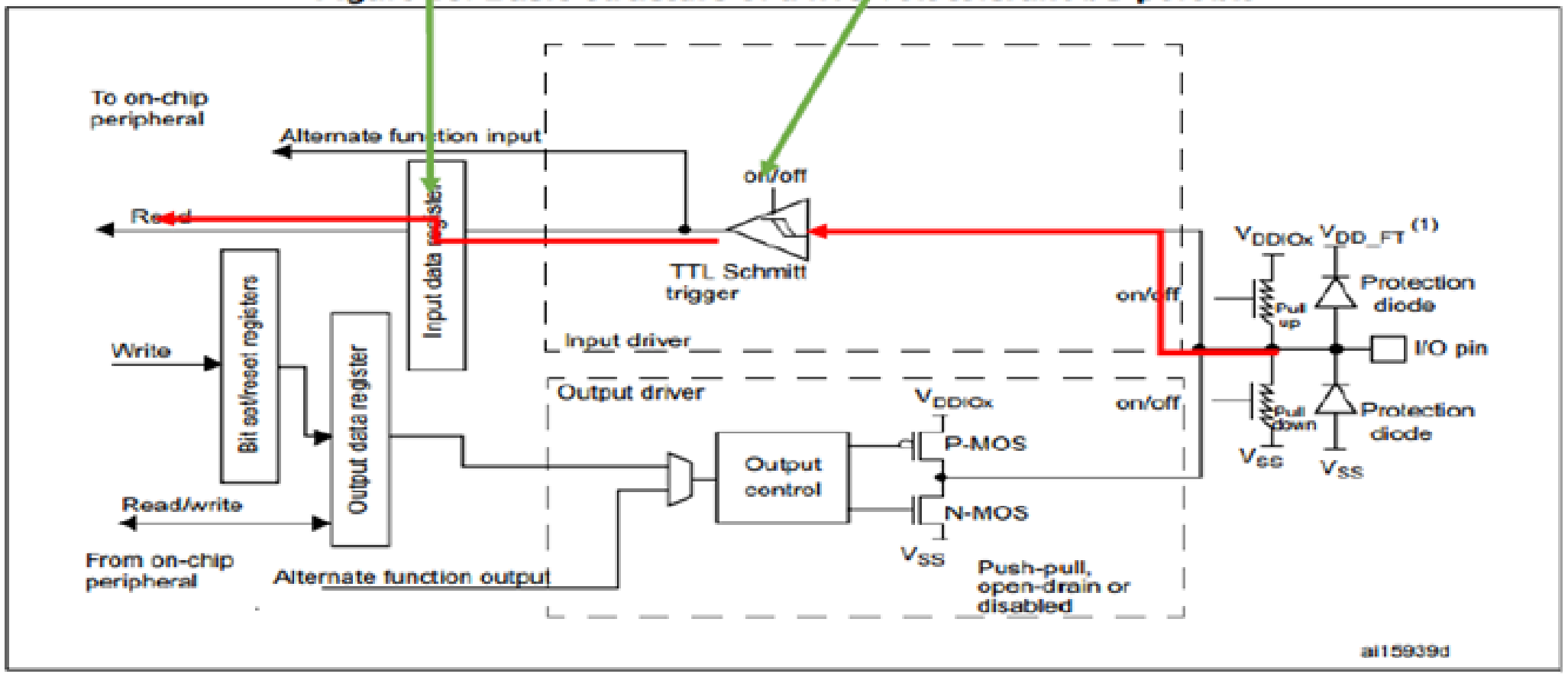
PC13

Input Signal Path

Input data register(IDR)

Mode register(MODER)

Figure 18. Basic structure of a five-volt tolerant I/O port bit



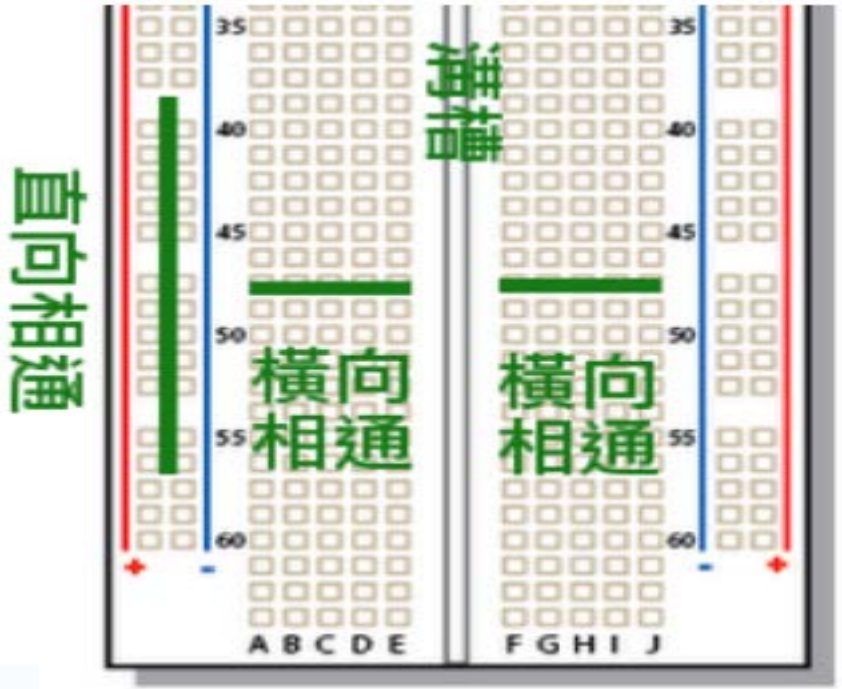
1. V_{DD_FT} is a potential specific to five-volt tolerant I/Os and different from V_{DD}.

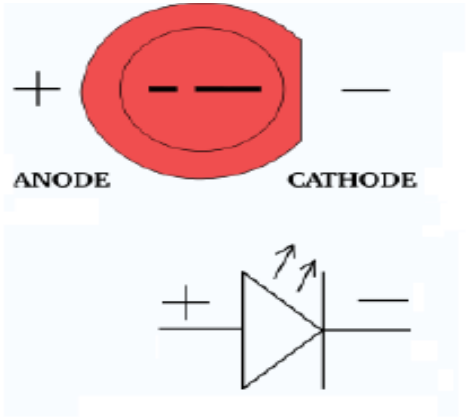
GPIO講解大綱

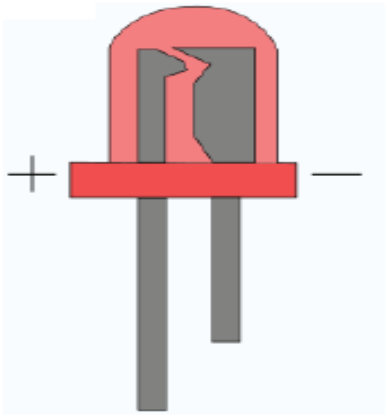
- GPIO 硬體架構與外接電路
- GPIO 組合語言控制register，register位址
  - Reference manual和programming manual定義reg和function (重要! 花十分鐘帶你看，以後要學會自己查)
  - 記憶體分配，GPIO register設定，程式範例
  - 補充資料-如何看register設定
- 實驗電路
- 範例程式，練習與作業

Lab 2 實驗零件

- Nucleo-L476RG board
- 麵包板
- 4DIP Switch
  - 1K排阻*1
- LED *4
  - 220歐姆電阻*4

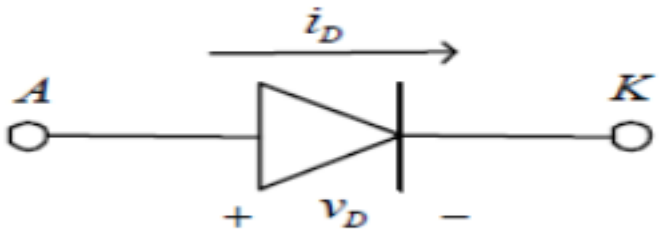


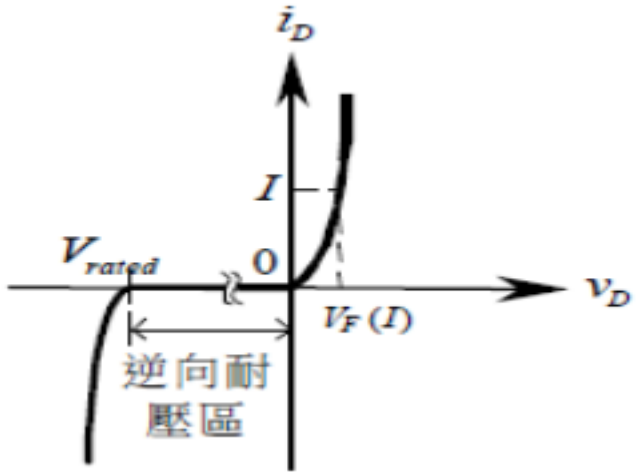


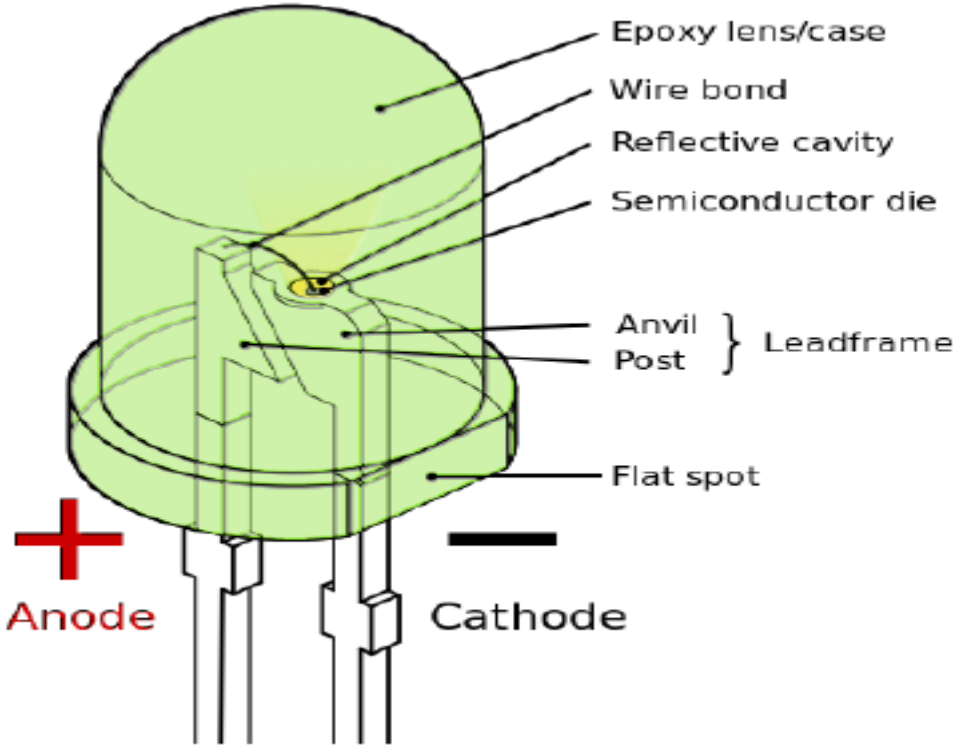


LED

- 特性類似二極體，導通時發光，導通電壓約為0.3 or 0.7V
- 二極體內阻小，使用上通常會加上限流電阻避免LED燒毀

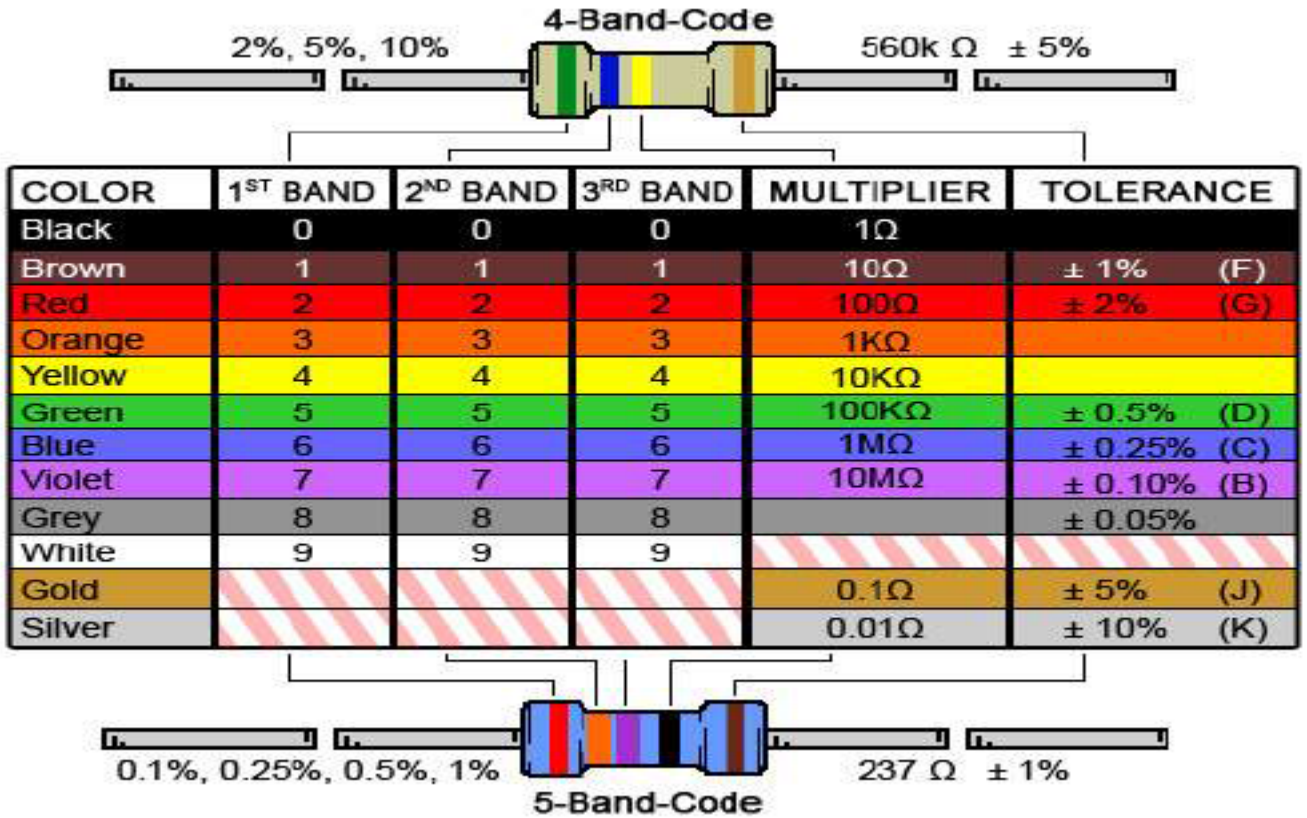






電阻

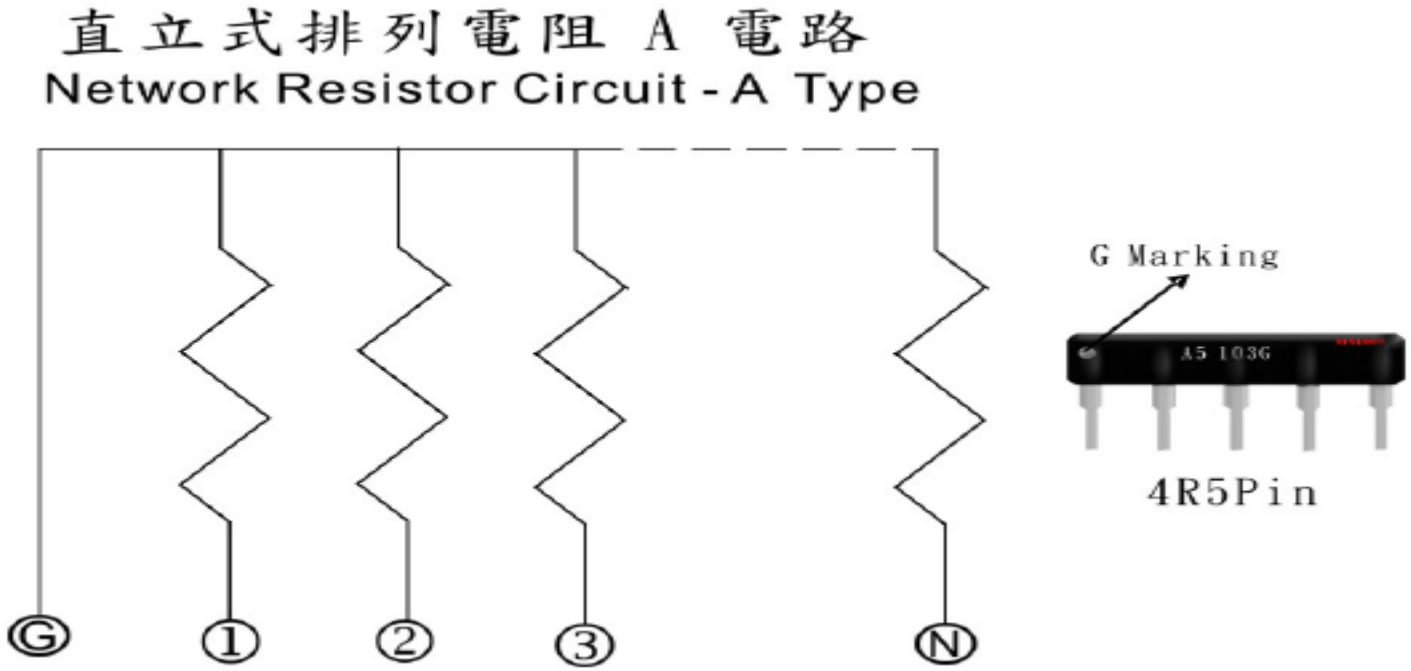
- 利用色碼標示電阻值



COLOR	1ST BAND	2ND BAND	3RD BAND	MULTIPLIER	TOLERANCE
Black	0	0	0	1Ω	
Brown	1	1	1	10Ω	± 1% (F)
Red	2	2	2	100Ω	± 2% (G)
Orange	3	3	3	1KΩ	
Yellow	4	4	4	10KΩ	
Green	5	5	5	100KΩ	± 0.5% (D)
Blue	6	6	6	1MΩ	± 0.25% (C)
Violet	7	7	7	10MΩ	± 0.10% (B)
Grey	8	8	8		± 0.05%
White	9	9	9		
Gold				0.1Ω	± 5% (J)
Silver				0.01Ω	± 10% (K)

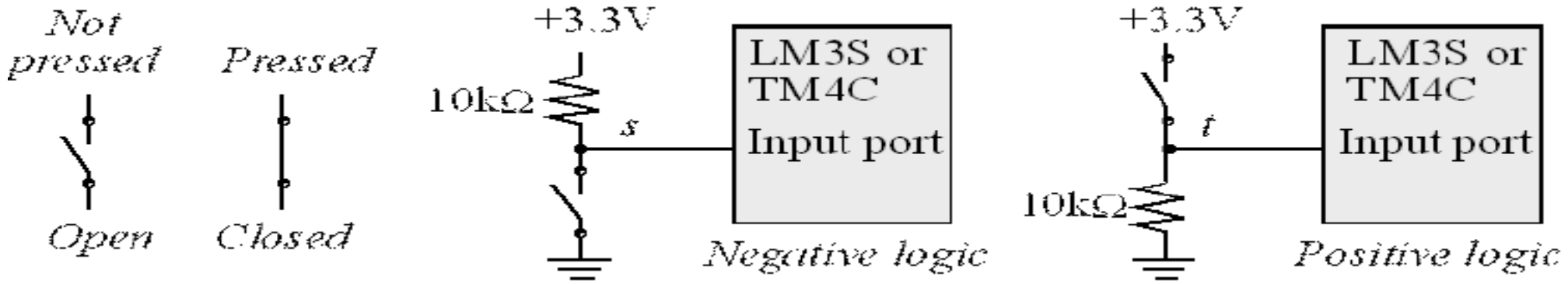
排阻

- 集合式電阻
- 用數字標記電阻值，例如：103=10*10^3 = 10K歐姆



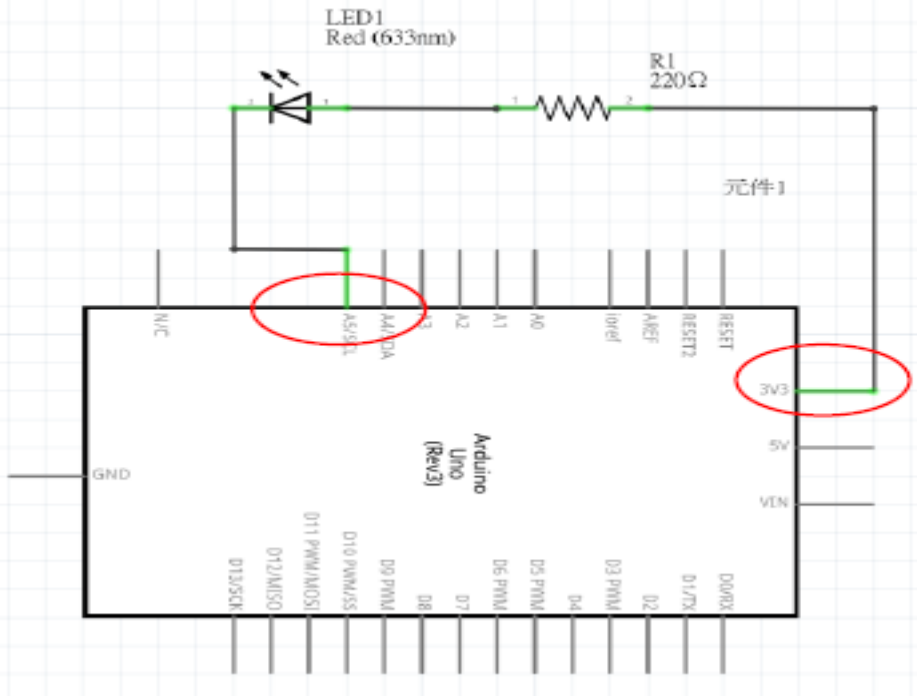
Negative logic and Positive logic

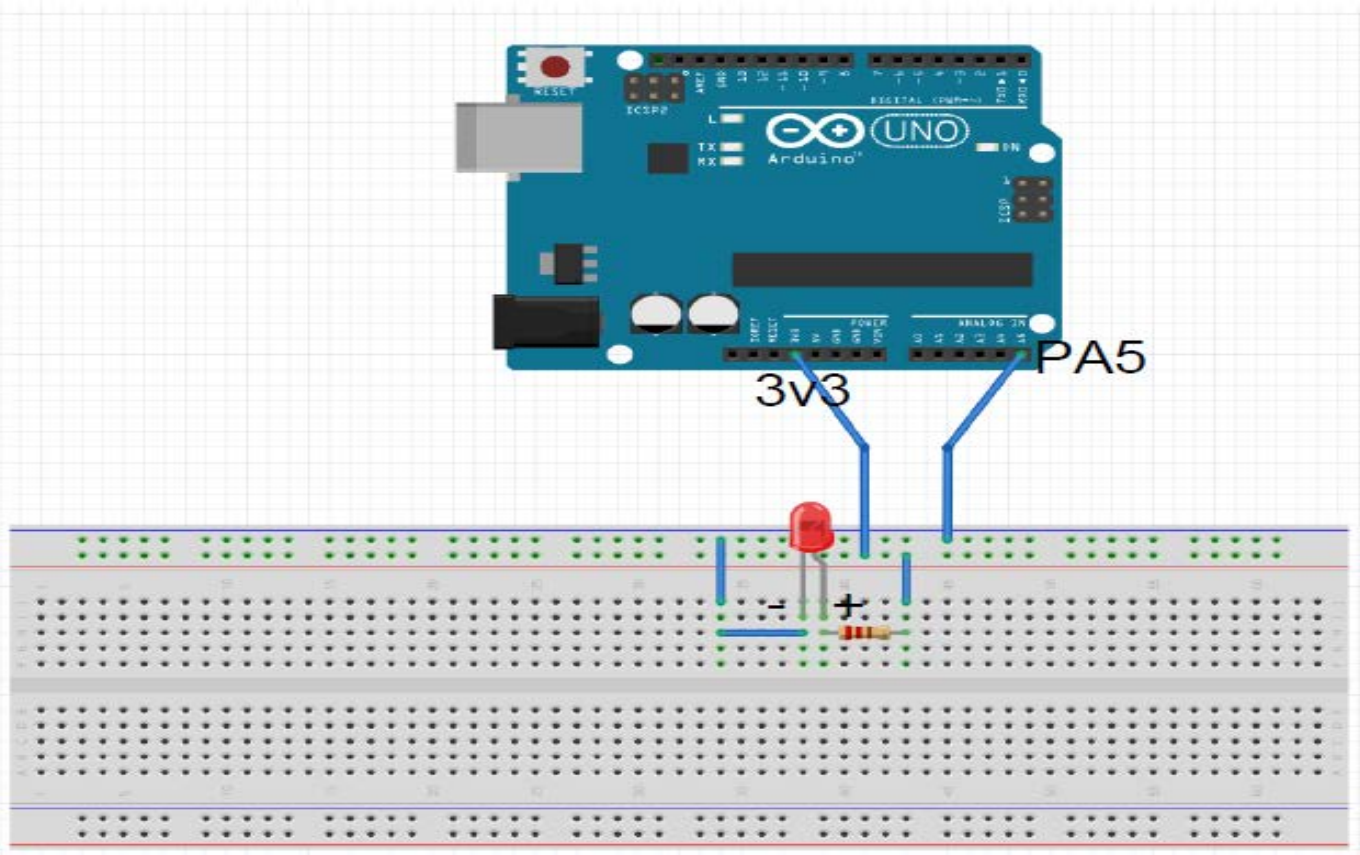
- logic 可指某個零件”動作”時CPU所收到邏輯準位
- 若某裝置動作時CPU收到的是High ”1”準位則稱Positive logic或稱Active High
- 反之裝置位動作CPU收到的是Low “0”準位則稱Negative logic或稱Active Low



How to connect breadboard, LEDs and STM32

- An active low circuit
  - Output ‘0’ LED燈亮

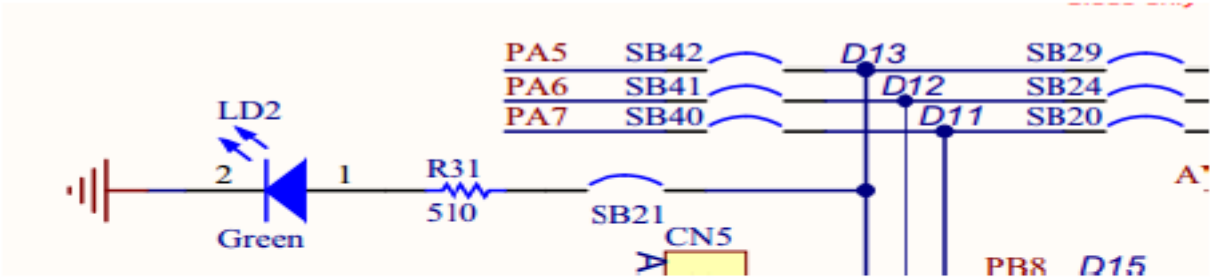




低電位驅動電路圖與照片，本次實驗使用

How to turn on single LED?

- Nucleo-L476RG has a onboard LED(LD2) connect at *GPIOA pin5* which is an active high circuit



高電位驅動

非本次使用的電路

```
        .syntax unified
        .cpu cortex-m4
        .thumb

        .text
        .global main
        .equ RCC_AHB2ENR, 0x4002104C
        .equ GPIOA_MODER, 0x48000000
        .equ GPIOA_OTYPER, 0x48000004
        .equ GPIOA_OSPEEDR, 0x48000008
        .equ GPIOA_PUPDR, 0x4800000C
        .equ GPIOA_ODR, 0x48000014

//LED on PA5
main:
        //Enable AHB2 clock
        movs    r0, #0x1
        ldr     r1, =RCC_AHB2ENR
        str     r0, [r1]

        //Set PA5 as output mode
        movs    r0, #0x400
        ldr     r1, =GPIOA_MODER
        ldr     r2, [r1]
        and     r2, #0xFFFFF3FF //Mask MODER5
        orrs    r2, r2, r0
        str     r2, [r1]

        //Default PA5 is Pull-up output, no need to set

        //Set PA5 as high speed mode
        movs    r0, #0x800
        ldr     r1, =GPIOA_OSPEEDR
        strh    r0, [r1]

        ldr     r1, =GPIOA_ODR
L1:
        movs    r0, #(1<<5)          1變成100000
        strh    r0, [r1]

        B L1
```

How to move a single LED?

- Example codes

LED Blink

Set PA5 output level via ODR

```
                //Set data register address
                ldr         r1, =GPIOA_ODR
LED:
                //Set PA5 as low then delay
                movs        r0, #0
                strh        r0, [r1]
                bl          delay
                //Set PA5 as high then delay
                movs        r0, #(1<<5)
                strh        r0, [r1]
                bl          delay

                B LED
```

Note: 修改ODR會一次改到整個GPIO port的值，若只需改動到某一個pin腳時可利用BSRR register存取

How to delay 1 second?

- Each instruction has own execution cycles(e.g. MOV take 1 cycle, LDR/STR take 2 cycles,… etc.)
- By default, our CPU(STM32L476) runs on 4MHz, 1cycle = 0.25uS
- So se can simply write a busy loop code as a delay function.
- Example codes

Cycles				
	Delay:			
2		**LDR**	**R3, =X**	
2	**L1:**	**LDR**	**R4, =Y**	
1	**L2:**	**SUBS**	**R4, #1**	
1 or 3		**BNE**	**L2**	跳去標註L2那行
1		**SUBS**	**R3, #1**	
1 or 3		**BNE**	**L1**	
1		**BX**	**LR**	

Why ≅?

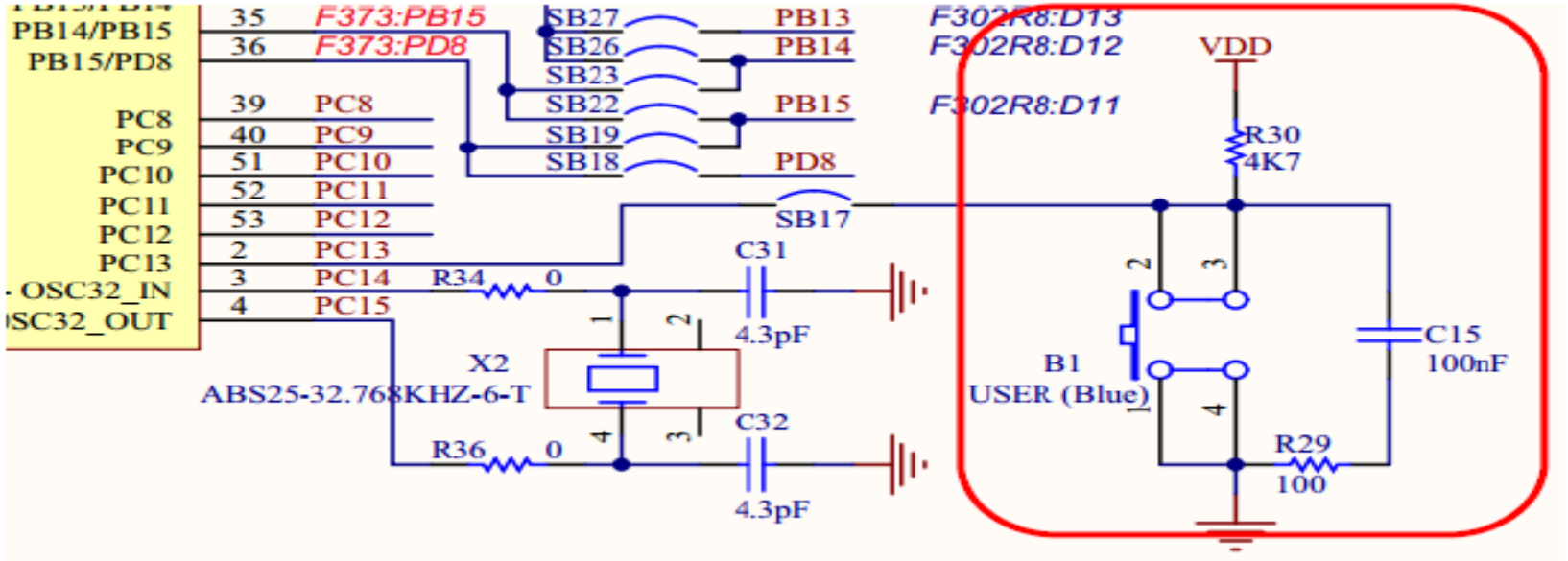
$$Time \cong 2 + (2 + (1 + 3) * Y - 1 + 1 + 3) * X - 1 + 1$$

BNE L2 not taken

BNE L1 not taken

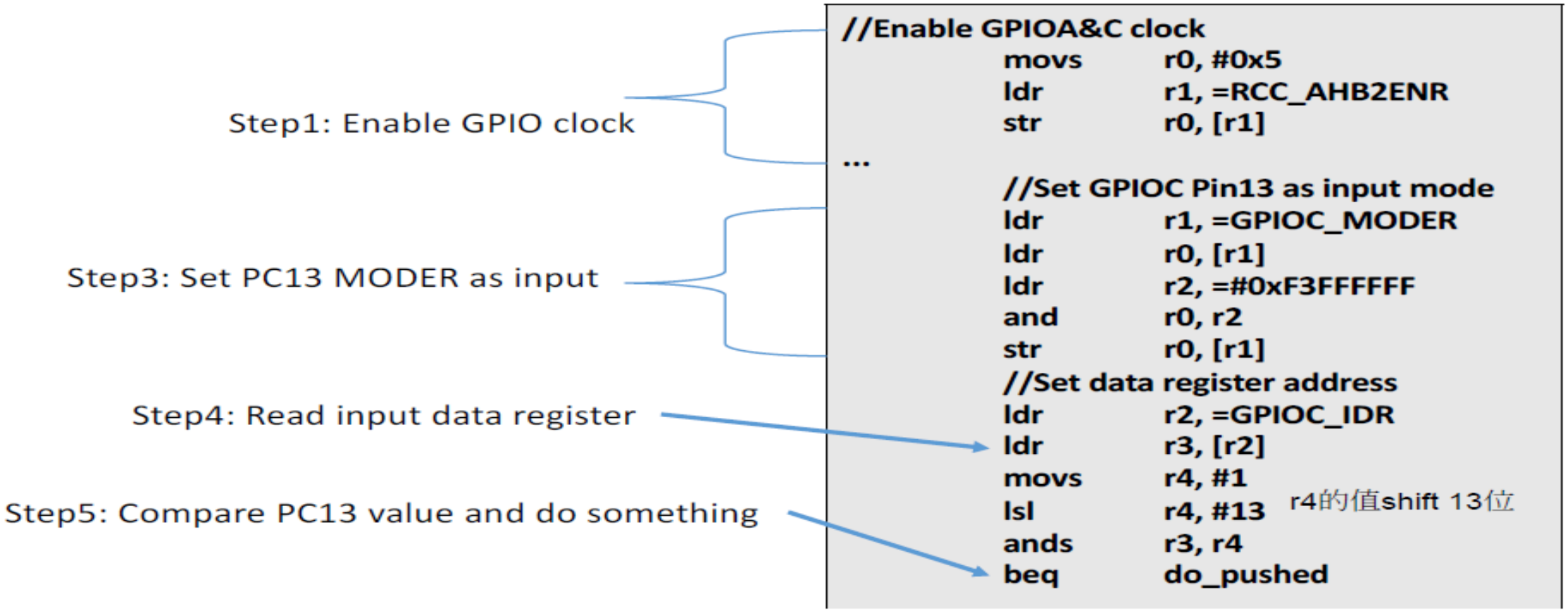
How to read user button?

- Configure a GPIO pin as input
  - If external circuit has pull-up resister, the pin can configured as floating input state.



GPIO Input Configure Example

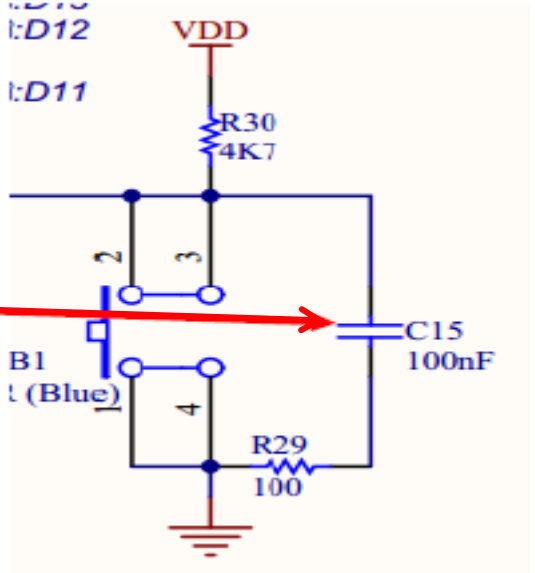
- Onboard user button connect on PC13

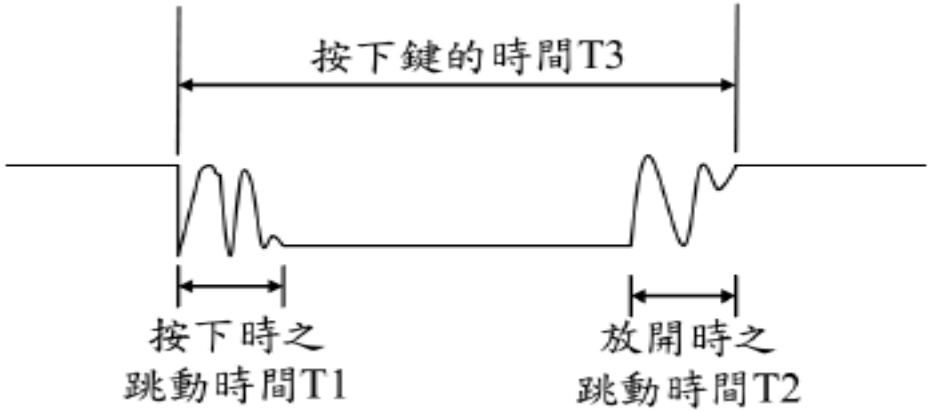


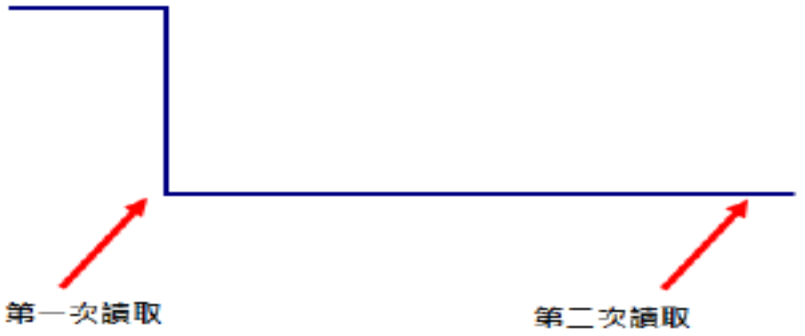
Note: Input預設為floating狀態且不需設Speed register

Debounce

- Hardware method
  - Add a 濾波電容
- Software method
  - 讀取GPIO Pin後間隔一段時間再
  - 讀取一次確認
  - 連續讀取N次, 看讀值是否穩定無改變



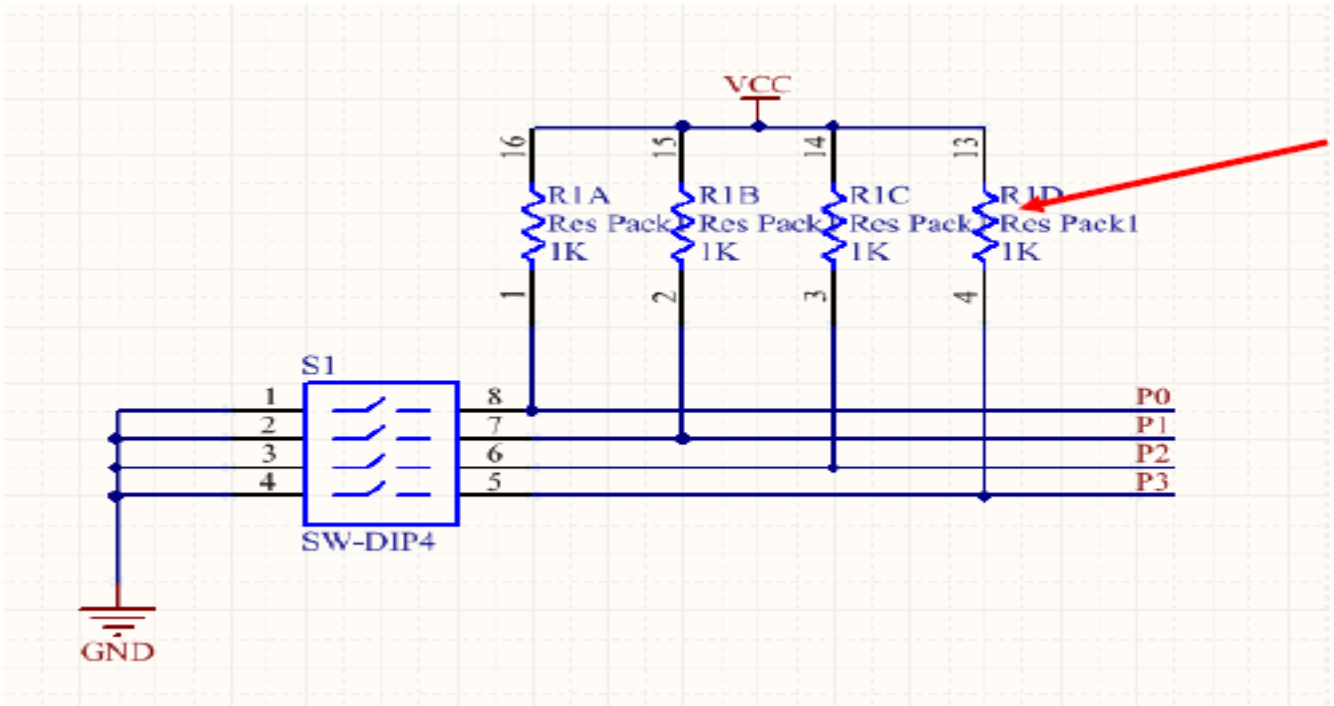




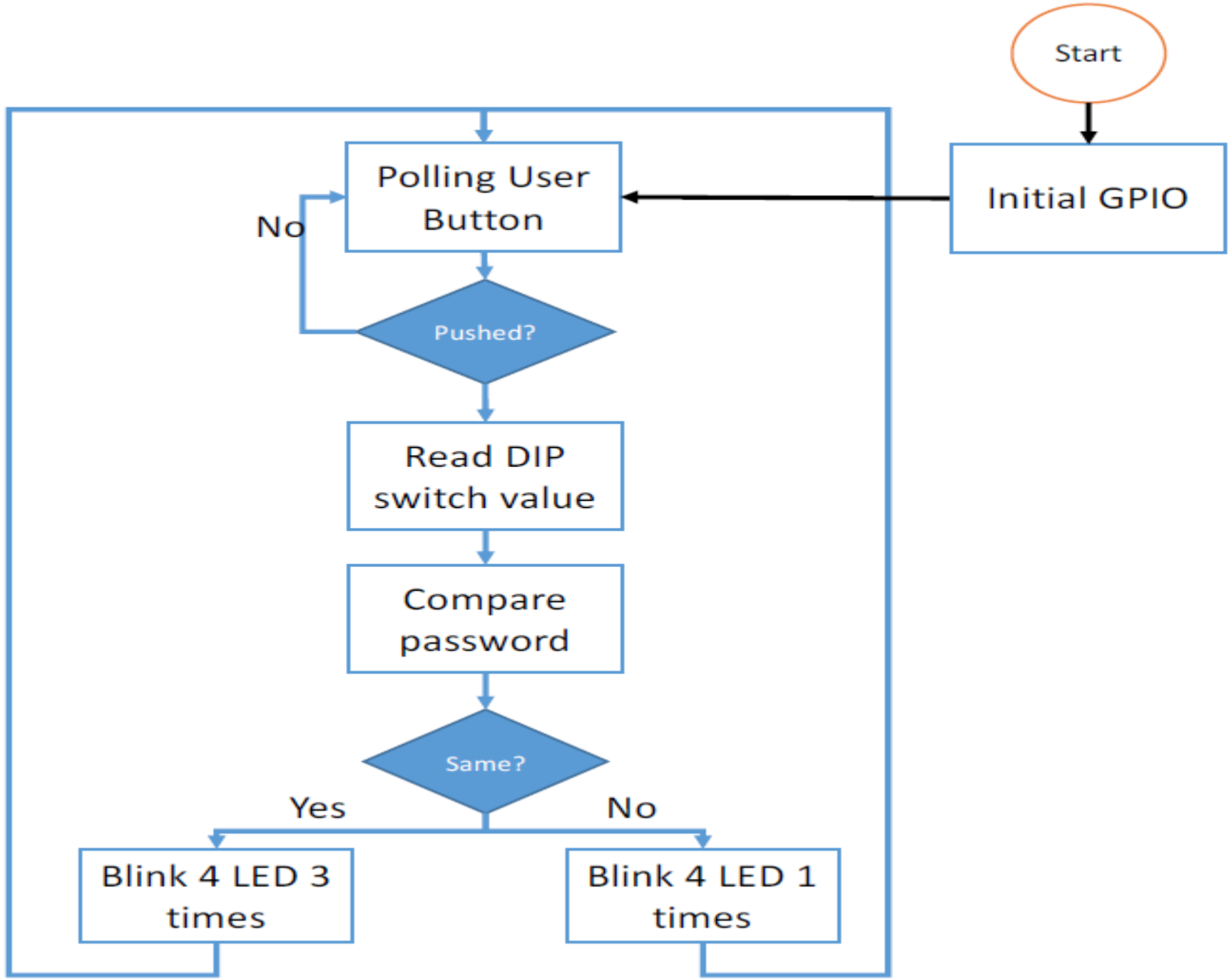
How to connect DIP switch and STM32

A simple DIP Switch Circuit

- When switch ‘ON’ Px get GND level (0), ‘OFF’ get VCC level(‘1’)
  - It an active low circuit



Pull-up resistance
(可不接,由GIPO內部設定)


Start
Initial GPIO
Polling User Button
No
Pushed?
Read DIP switch value
Compare password
Same?
Yes
No
Blink 4 LED 3 times
Blink 4 LED 1 times

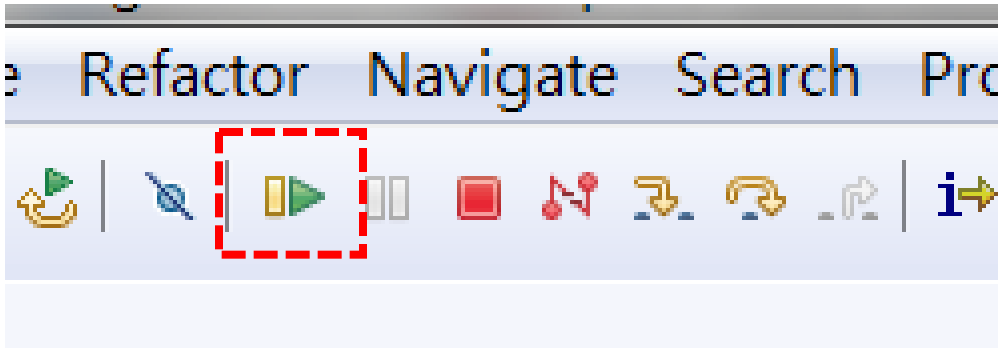
Reference

- STM32L4x6 Reference manual
  - http://www.st.com/resource/en/reference_manual/dm00083560.pdf
- Embedded system course from NCKU
  - http://wiki.csie.ncku.edu.tw/embedded/GPIO

Lab2.1 Assembly GPIO
範例程式

Code provided by NCTU CS 謝明恩 吳赫倫
Slide made by NCTU ME 助教 林穎毅
20170314

- 記得如果是要不斷執行的程式，在run debug後，要按resume (F8)，不然會自動停止在某break point.



起始的參數跟環境設定

```
1 .syntax unified
2 .cpu cortex-m4
3 .thumb
4
5 .data
6     Leds: .byte 0
7
8 .text
9     .global main
10    .equ mini_sec,  0x96      // 150
11    .include "../src/pin.s"   // import GPIO constants
12
```

MAIN

```
138 main:
139     bl GPIO_init     // Setup GPIO
140     movs r0, #0x3    // write 0b'000011 to Leds
141     ldr r5, =Leds    // r5 = Leds addr
142     strb r0, [r5]
143     movs r6, #0x1    // r6 = lock (0=lock)
144     movs r3, #0      // r3 = last button stat
145     b Loop
146
147 L: b L
```

第一步為進GPIO_init 把參數調成我們需要的樣子

GPIO_init

```
GPIO_init:
    // Enable AHB2 clock
    movs r0, #0x6
    ldr r1, =RCC_AHB2ENR
    str r0, [r1]
    // Set PB3 ~ PB6 to output mode
    ldr r1, =GPIOB_MODER
    ldr r0, [r1]
    and r0, #0xFFFFC03F
    orr r0, #0x1540  //??
    str r0, [r1]
    // Set PB3 ~ PB6 to high speed
    ldr r1, =GPIOB_OSPEEDR
    ldr r0, [r1]
    and r0, #0xFFFFC03F
    orr r0, #0x2A80
    str r0, [r1]
    // Set PB3 ~ PB6 to pullup
    ldr r1, =GPIOB_PUPDR
    ldr r0, [r1]
    and r0, #0xFFFFC03F
    orr r0, #0x1540
    str r0, [r1]
    // Set PC13 to input mode
    ldr r1, =GPIOC_MODER
    ldr r0, [r1]
    and r0, #0xF3FFFFFF
    orr r0, #0x0
    str r0, [r1]
    bx lr
```

GPIO_init AHB2ENR

```
GPIO_init:
    // Enable AHB2 clock
    movs r0, #0x6
    ldr r1, =RCC_AHB2ENR
    str r0, [r1]
```

1.設r0 為6
2.然後把r1存成RCC_AHB2ENR的位置
3.把r0 的值存入r1所指的位置(也就是RCC_AHB2ENR=0x6)

6.4.17 AHB2 peripheral clock enable register (RCC_AHB2ENR)

Address offset: 0x4C

Reset value: 0x0000 0000

Access: no wait state, word, half-word and byte access

Note: When the peripheral clock is not active, the peripheral registers read or write access is not supported.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RNG EN	Res.	AESEN
													rw		rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	ADCEN	OTGFS EN	Res.	Res.	Res.	Res.	GPIOH EN	GPIOG EN	GPIOF EN	GPIOE EN	GPIOD EN	GPIOC EN	GPIOB EN	GPIOA EN
		rw	rw					rw	rw	rw	rw	rw	rw	rw	rw

RCC_AHB2ENR=0000 0110 對應到上表即為GPIOB跟GPIOC的clock enable

```
38      // Set PB3 ~ PB6 to output mode
39      ldr r1, =GPIOB_MODER
40      ldr r0, [r1]
41      and r0, #0xFFFFC03F
42      orr r0, #0x1540
43      str r0, [r1]
```

1. 把r1存GPIOB_MODER位置
2.把r0存入r1內的位置的值
3.把r0存成r0跟0xFFFFC03F作and的結果
4.把r0存成r0跟0x2A80作or的結果
5.把r0的值存入r1指的位置(意即GPIOB_MODER)

P.278, GPIO port mode register (GPIOx_MODER) in Reference manual

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MODER15[1:0]		MODER14[1:0]		MODER13[1:0]		MODER12[1:0]		MODER11[1:0]		MODER10[1:0]		MODER9[1:0]		MODER8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MODER7[1:0]		MODER6[1:0]		MODER5[1:0]		MODER4[1:0]		MODER3[1:0]		MODER2[1:0]		MODER1[1:0]		MODER0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

MODERy[1:0]決定第y pin GPIO

使用的configuration，2個bit為一組

- 00: Input
- 01: output mode
- 10: Alternate function mode
- 11: Analog mode
- 當reset後，GPIOA_MODER=0xA800 0000、GPIOB_MODER=0x0000 0280，其他皆為0。
- PA3, PA4, PB13, PB14, PB15 為 Debug Pin in AF

GPIOB_MODER為0x0000 1540(=0001 0101 0100 0000)即設PB3~6為OUTPUT

GPIO 寫值進register運算方式

R0 = 0X 0000 0280
= 0X 0000 0000 0000 0000　　0000 0010 1000 0000

0X FFFF C03F
= 0X 1111 1111 1111 1111　　1100 0000 0011 1111

and R0, #0XFFFFC03F
R0 = 0X 0000 0000 0000 0000　　0000 0000 0000 0000

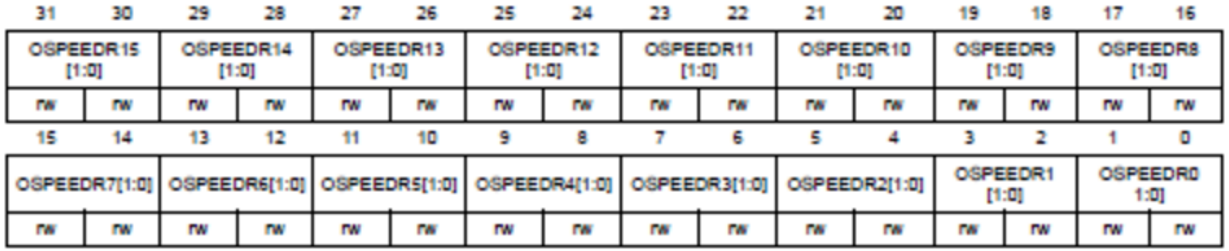
orr R0, #1540
R0 = 0X 0100 1000 0000 0000　　0001 0101 0100 0000

也就是Pin3,4,5,6的MODER設定成output mode

```
44      // Set PB3 ~ PB6 to high speed
45      ldr r1, =GPIOB_OSPEEDR
46      ldr r0, [r1]
47      and r0, #0xFFFFC03F
48      orr r0, #0x2A80
49      str r0, [r1]
```

1. 把r1存GPIOB_OSPEEDR位置
2.把r0存入r1內的位置的值
3.把r0存成r0跟0xFFFFC03F作and的結果
4.把r0存成r0跟0x2A80作or的結果
5.把r0的值存入r1指的位置(意即GPIOB_OSPEEDR)

P.279, GPIO port output speed register (GPIOx_OSPEEDR) in Reference manual



31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OSPEEDR15 [1:0]		OSPEEDR14 [1:0]		OSPEEDR13 [1:0]		OSPEEDR12 [1:0]		OSPEEDR11 [1:0]		OSPEEDR10 [1:0]		OSPEEDR9 [1:0]		OSPEEDR8 [1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OSPEEDR7[1:0]		OSPEEDR6[1:0]		OSPEEDR5[1:0]		OSPEEDR4[1:0]		OSPEEDR3[1:0]		OSPEEDR2[1:0]		OSPEEDR1 [1:0]		OSPEEDR0 1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

OSPEEDRy[1:0]決定output的速

度。在不同的電容與電壓(VDD)下，輸出頻率可能不同，以下只列出代表值。

- 00: Low speed (2MHz)
- 01: Medium speed (10MHz)
- 10: Fast speed (50MHz)
- 11: High speed (100MHz)
- reset GPIOB_OSPEEDR=0x0000 00C0，其他皆為皆為0x0000 0000
- 速度越高，雜訊與耗電量越多。
- 在其他狀態下的速度，可以參考 datasheet p134 table 58. I/O AC Characteristics

GPIOB_OSPEEDR=0x0000 2A80(0010 1010 1000 0000)即PB3~6為fast mode

```
50      // Set PB3 ~ PB6 to pullup
51      ldr r1, =GPIOB_PUPDR
52      ldr r0, [r1]
53      and r0, #0xFFFFC03F
54      orr r0, #0x1540
55      str r0, [r1]
```

1. 把r1存GPIOB_PUPDR位置
2.把r0存入r1內的位置的值
3.把r0存成r0跟0xFFFFC03F作and的結果
4.把r0存成r0跟0x2A80作or的結果
5.把r0的值存入r1指的位置(意即GPIOB_PUPDR)

P.280, GPIO port pull-up/pull-down register (GPIOx_PUPDR) in Reference manual

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PUPDR15[1:0]		PUPDR14[1:0]		PUPDR13[1:0]		PUPDR12[1:0]		PUPDR11[1:0]		PUPDR10[1:0]		PUPDR9[1:0]		PUPDR8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PUPDR7[1:0]		PUPDR6[1:0]		PUPDR5[1:0]		PUPDR4[1:0]		PUPDR3[1:0]		PUPDR2[1:0]		PUPDR1[1:0]		PUPDR0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

PUPDRy[1:0]決定pin y是否pull-up/pull-down

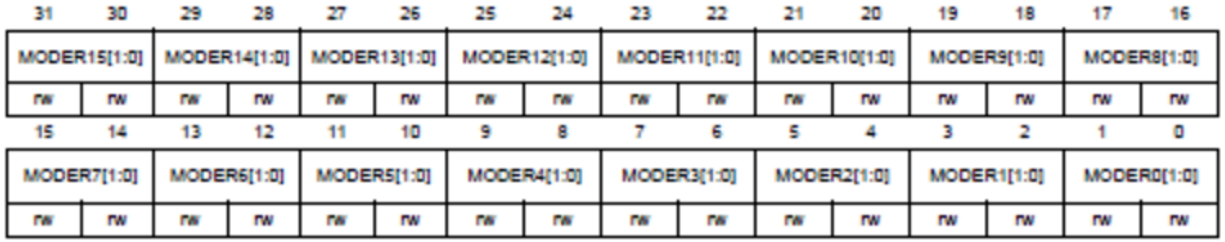
- 00: No pull-up, pull-down
- 01: Pull-up
- 10: Pull-down
- 11: Reserved
- reset GPIOA_PUPDR=0x6400 0000、GPIOB_PUPDR=0x0000 0100，其他0x0000 0000。

GPIOB_OSPEEDR=0x0000 2A80(0010 1010 1000 0000)即PB3~6為pull up

```
// Set PC13 to input mode
ldr r1, =GPIOC_MODER
ldr r0, [r1]
and r0, #0xF3FFFFFF
orr r0, #0x0
str r0, [r1]
bx lr
```

1. 把r1存GPIOC_MODER位置
2.把r0存入r1內的位置的值
3.把r0存成r0跟0xFFFFC03F作and的結果
4.把r0存成r0跟0x2A80作or的結果
5.把r0的值存入r1指的位置(意即GPIOC_MODER)

P.278, **GPIO port mode register (GPIOx_MODER)** in Reference manual



31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MODER15[1:0]		MODER14[1:0]		MODER13[1:0]		MODER12[1:0]		MODER11[1:0]		MODER10[1:0]		MODER9[1:0]		MODER8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MODER7[1:0]		MODER6[1:0]		MODER5[1:0]		MODER4[1:0]		MODER3[1:0]		MODER2[1:0]		MODER1[1:0]		MODER0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

MODERy[1:0]決定第y pin GPIO

使用的configuration，2個bit為一組

- 00: Input
- 01: output mode
- 10: Alternate function mode
- 11: Analog mode
- 當reset後，GPIOA_MODER=0xA800 0000、GPIOB_MODER=0x0000 0280，其他皆為0。
- PA3, PA4, PB13, PB14, PB15 為 Debug Pin in AF

GPIOB_MODER為0x0000 0000(=0000 0000 0000 0000)即設PB13為INPUT

```
138 main:
139     bl GPIO_init    // Setup GPIO
140     movs r0, #0x3   // write 0b'000011 to Leds
141     ldr r5, =Leds   // r5 = Leds addr
142     strb r0, [r5]
143     movs r6, #0x1   // r6 = lock (0=lock)
144     movs r3, #0     // r3 = last button stat
145     b Loop
146
147 L: b L
```

1.設r0為3
2.把r5存入Leds 位置
3.把r0的值存入r5指的位置
4.令r6為1(unlock的狀態)
5.令r3為0(last button 狀態)
6.前往Loop

```
// r3 = last button status
// r4 = shift direction (0:left, 1:right)
// r5 = Leds Address
// r6 = lock shifting (0=lock)
Loop:
    ldrb r0, [r5]         // Load Leds into r0
    and r0, r0, #0x1E     // Get the center 4 bits
    lsl r0, #2            //     which are the next led pattern
    bl DisplayLED         // Branch to change led display
    movs r2, #100         // Check For 100 times of Button pressing
```

1.把Leds的值讀到r0中
2.把r0存成r0跟0X1E作and的結果
3.把r0往左位移2位元(bit)
4.前往 DisplayLED
5設r2為100

```
DisplayLED:
    PUSH {r1, r2, lr}
    ldr r1, =GPIOB_ODR
    ldr r2, [r1]          // Get Current Output
    and r2, #0xFFFFFF87   // Clear PB3 to PB6
    eor r0, #0xFFFFFFFF   // Reverse r0 (Active Low)
    and r0, #0x78         // Select the needed bits
    orr r2, r0            // Set the output bits
    str r2, [r1]          // Write back to Output Reg
    POP {r1, r2, pc}
```

1.從暫存器取出位置
PUSH r1, r2 //空間騰出來 lr代表從哪個loop跳來這裡
2.把r1存成GPIOB_ODR的位置
3.r2存取r1指向位置的數值
4.r2跟0xFFFFFF87作and
5.r0跟0xFFFFFFFF作exclusive or
6.r0跟0x78作and
7.r2跟r0作or
8把r2資料存到GPIOB_ODR
9.把從暫存器拿出來位置的值還給電腦(POP)

R2= 0X 00000000

R0 = 0X 0000 1000(最後八碼)

eor R0, #0XFFFFFFFF
R0 = 0X 1111 1111 1111 1111　　1111 1111 1111 0111

and R0, #078
R0 = 0X 0000 0000 0000 0000　　0000 0000 0111 0000

orr R2, R0
R2 = 0X 0000 0000 0000 0000　　0000 0000 0111 0000

Loop_check

```
loop_check:
    bl check_button       // Branch to Ckeck button press
    subs r2, r2, #1       // Decrement counter
    bne loop_check        // Check if loop is finished
    ldrb r0, [r5]         // Load old Leds into r0
    and r1, r0, #0x20     // Check if need to change shifting direction
    cbnz r1, Loop_right   // Hit left border
    and r1, r0, #0x01
    cbnz r1, Loop_left    // Hit right border
    b Loop_end
```

1.前往 check_button
2.r2 = r2-1
3.若r2!=0 則回到loop_check
4.r0存進之前狀態的LED
5.r1存成r0跟0x20 and的結果
6.若r1=0則前往loop_right
7.r1存成r0跟0x01 and的結果
6.若r1=0則前往loop_left
8.前往 Loop_end

```
check_button:
    PUSH {r0, r1, r2, lr}
    movs r0, #100          // Check for 100 times for debounce
    movs r1, #0            // r1 = The amount of times button equals 1

check_button_loop:
    bl Delay               // Delay for a short period of time
    ldr r2, =GPIOC_IDR     // Check if button is pressed
    ldr r2, [r2]           // R2為按鈕input訊號，有按是1；沒按反之
    ands r2, #0x2000       // 0x2000, GPIOC第13個bit是user button
    bne check_button_end   // If button = 0, branch to check_button_end
    add r1, r1, #1         // Else, increment button press counter
check_button_end:
    subs r0, r0, #1        // Decrement counter
    bne check_button_loop
    cmp r1, #50            // Check if there are more than 50 ones
    blt check_button_change   // Negative edge trigger
    movs r3, #1            // Last button status is 1
    POP {r0, r1, r2, pc}
```

```
Delay:
    PUSH {r0, lr}
    ldr r0, =mini_sec      // Use to control total delay time
Delay_loop:
    subs r0, r0, #1        // 1 cycle
    bne Delay_loop         // 1~3 cycle
    POP {r0, pc}
```

```
check_button_change:
    cbz r3, check_button_nochange  // Check if switch from 1 to 0
    movs r3, #0            // Last button status is 0
    cbz r6, check_button_change_1
    movs r6, #0x0          // Unlock Leds
    POP {r0, r1, r2, pc}
check_button_change_1:
    movs r6, #0x1          // Lock up Leds
    POP {r0, r1, r2, pc}
check_button_nochange:
    POP {r0, r1, r2, pc}
```

Cbz：確認值是否為0

```
Loop_left:
    mov r4, #0              // Change shift direction to left
    b Loop_end
Loop_right:
    mov r4, #1              // Change shift direction to right
    b Loop_end
Loop_end:
    cbz r4, mov_left        // Choose which way to shift Leds
mov_right:
    lsr r0, #1              // Shift Leds right
    b mov_done
mov_left:
    lsl r0, #1              // Shift Leds left
    b mov_done
mov_done:
    cbz r6, mov_end         // If locked, don't move
    strb r0, [r5]           // Else, write new Leds back to Leds
mov_end:
    b Loop
```

練習

- Lab 2.1:學號最後一碼轉為二進制，控制使四個LED亮起。上傳程式碼、word檔內放LED和電路照片 。(40%)
- Lab 2.2: 範例程式原本的pin腳是PB3~PB6，換一根pin腳(ex: PB2~PB5)做一樣的事情。上傳demo影片、程式碼、和word檔說明修改的地方。(40%)
- 2.3: GPIOB Pin8，要enable、設為輸出、pulldown、open-drain、medium speed。 逐項列出位址和值各為多少。(20%)

- Main.s和需要include的pin.s檔請一併上傳。檔名請加上組別、姓名和作業編號。