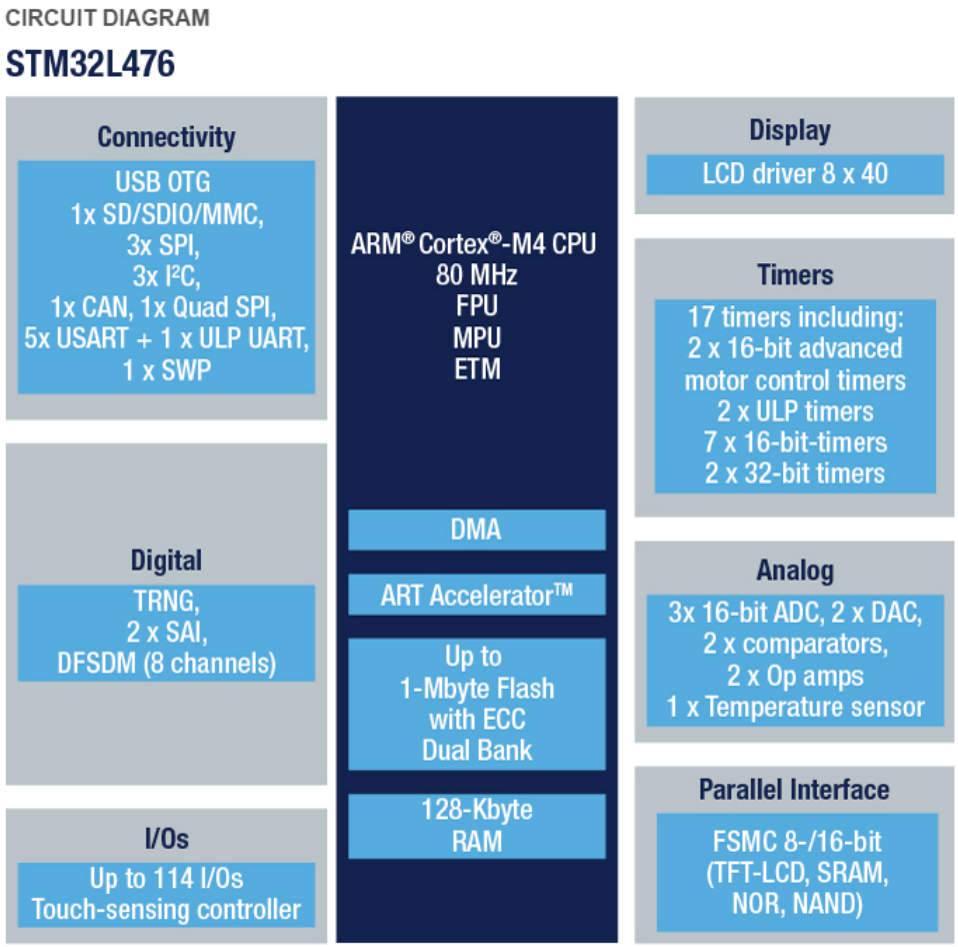
ARM STM32 GPIO

Features

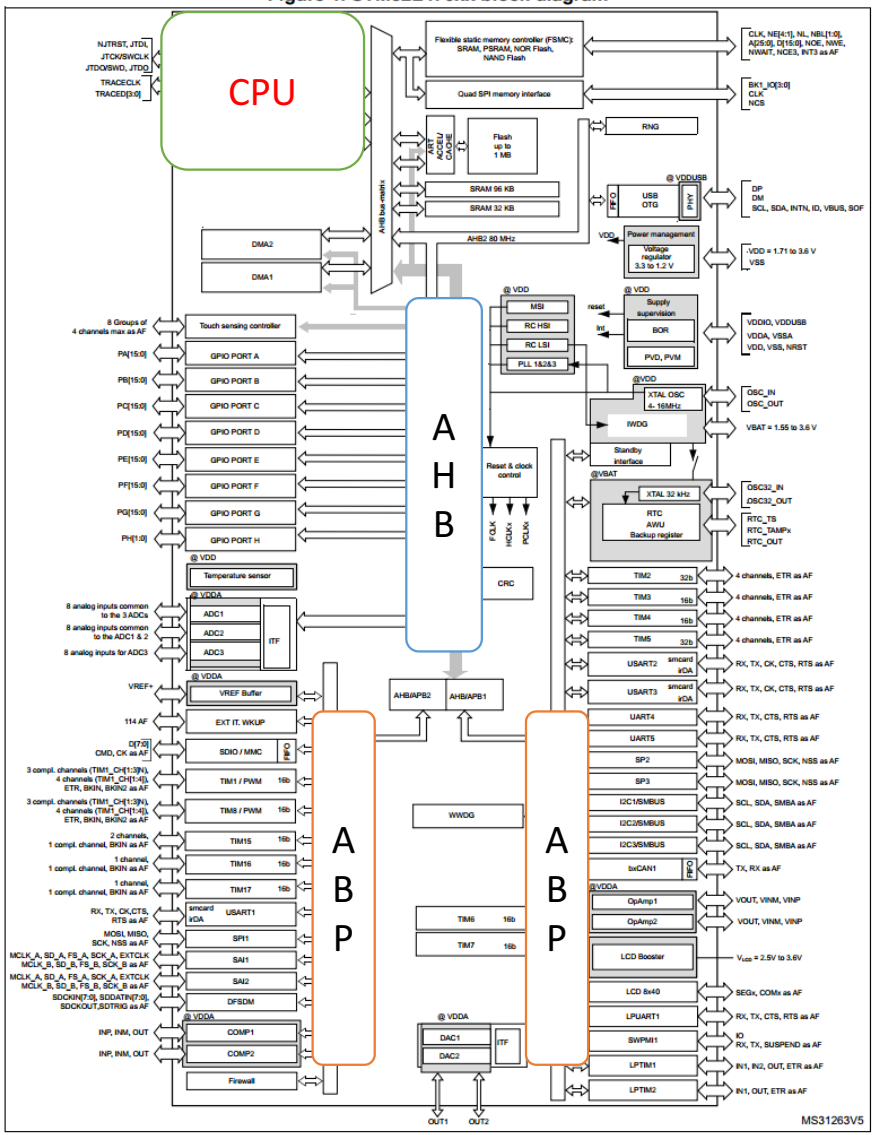
- ARM Cortex-M4
  - frequency up to 80 MHz
  - MPU
  - 100DMIPS/1.25DMIPS/MHz (Dhrystone 2.1)
  - DSP instructions
- Memories
  - 1MB Flash
  - 128KB SRAM
- 51 GPIO pins
- Timers
- Communication interfaces
  - I2C, SPI, CAN, USART, USB,…
- 12-bit ADC and DAC



Block diagram

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

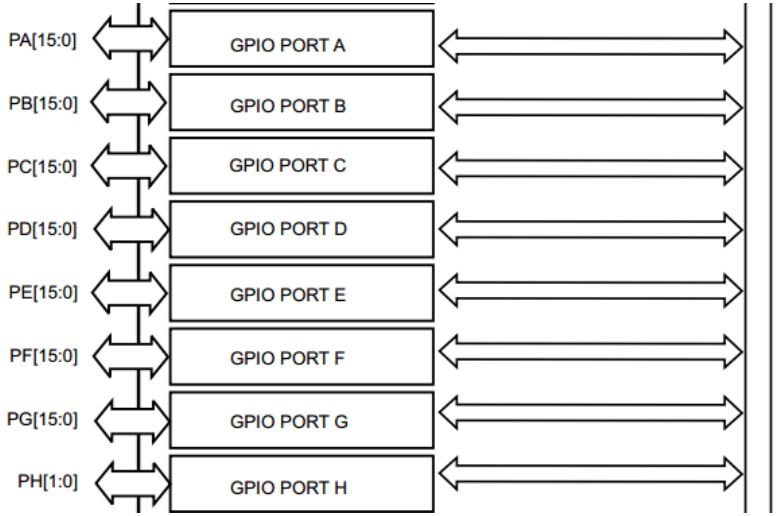
Figure 1. STM32L476xx block diagram



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

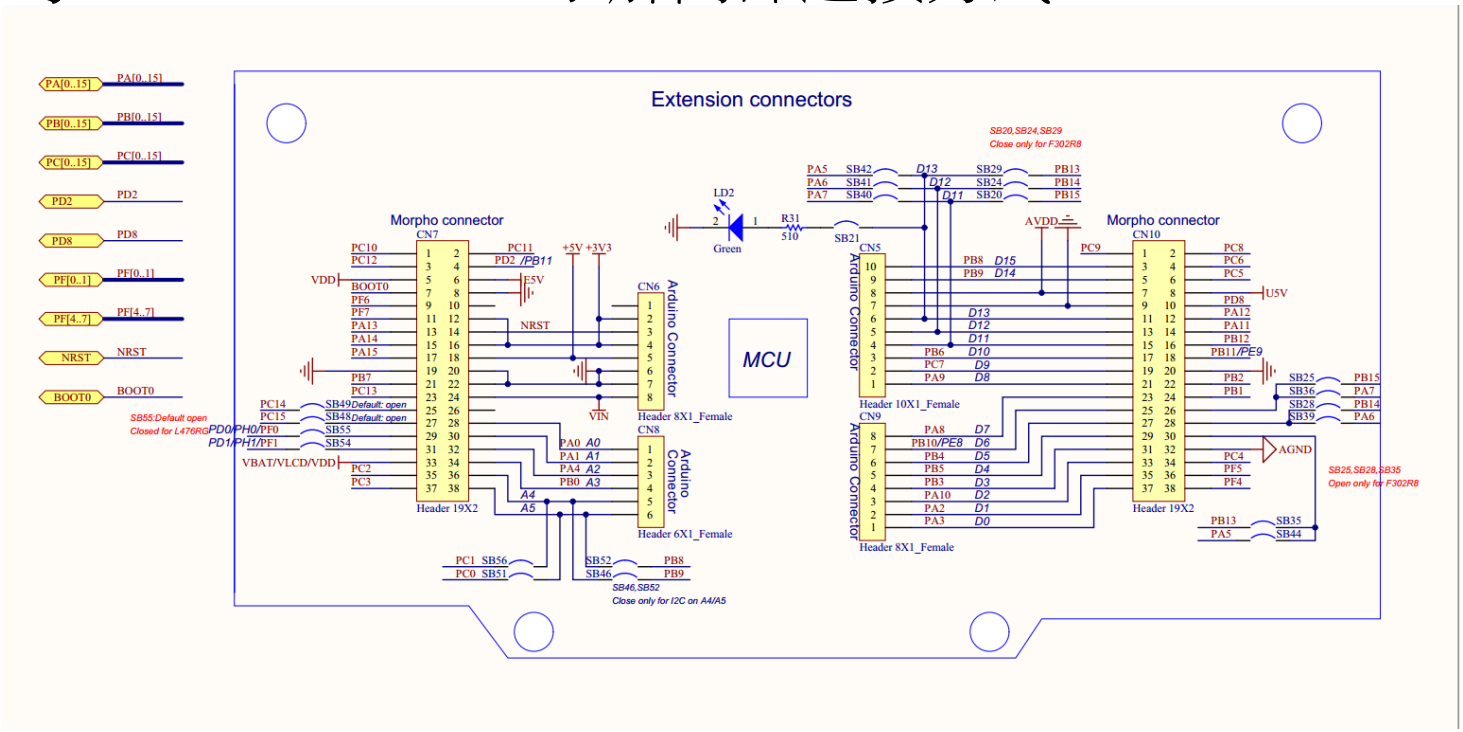
General-purpose inputs/outputs (GPIO)

- 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

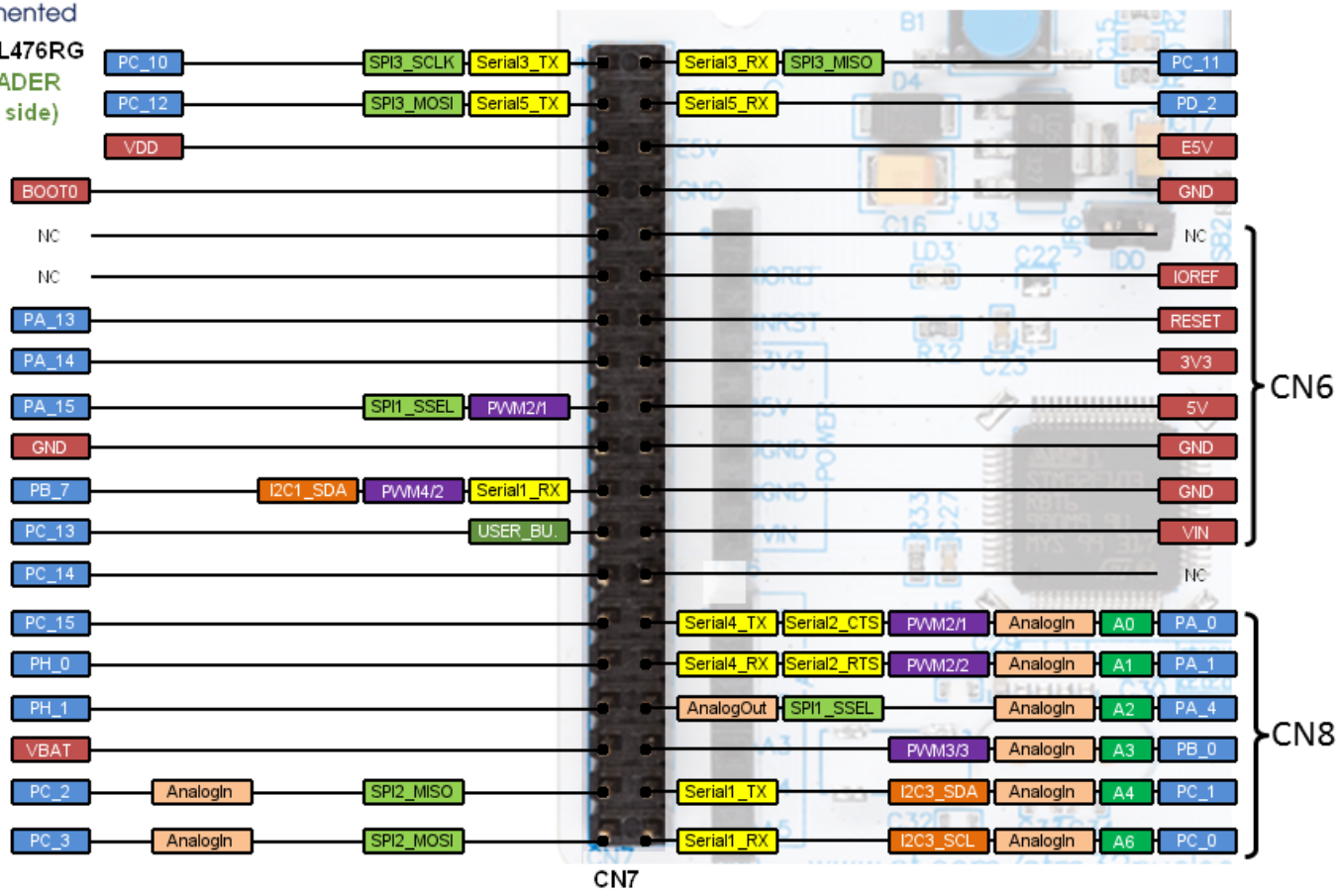


life.augmented

NUCLEO-L476RG

CN7 HEADER

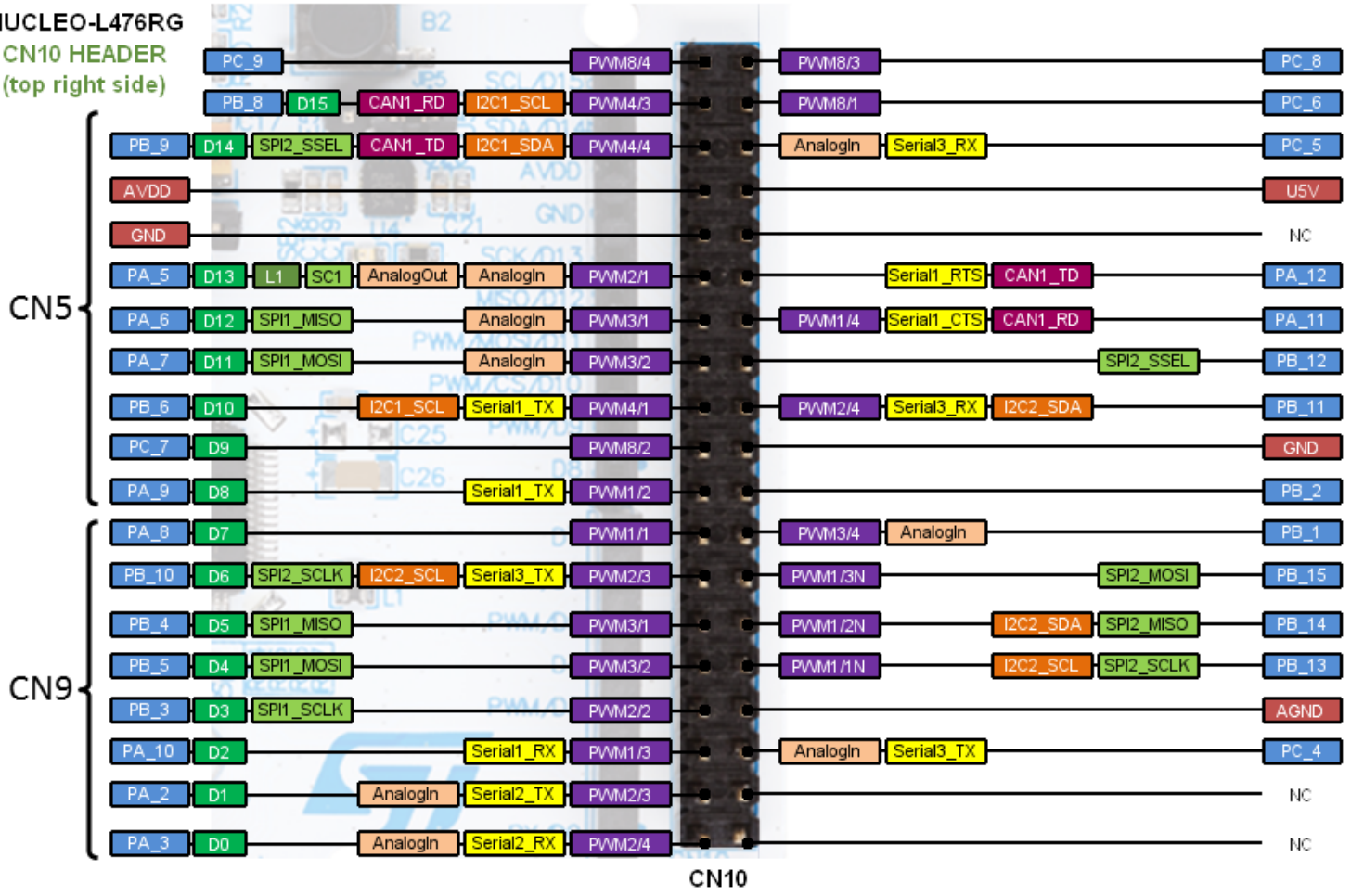
(top left side)



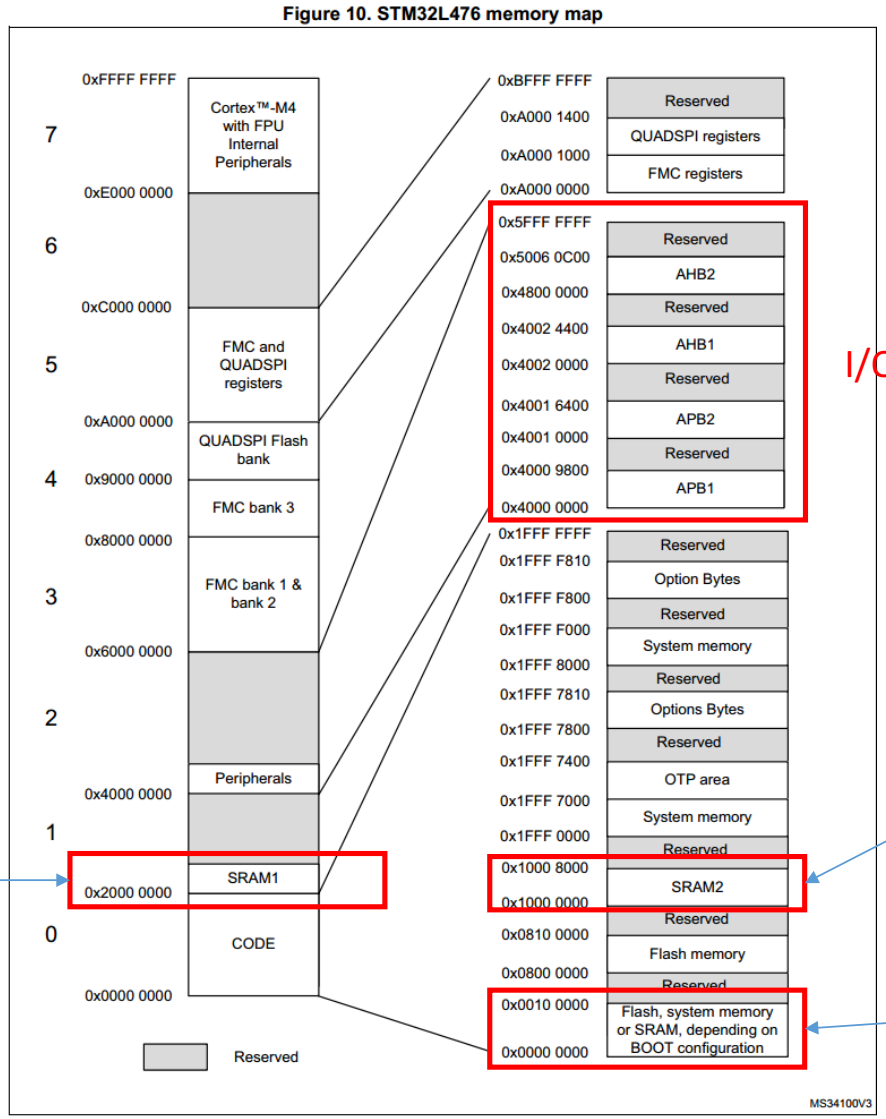
life.augmented

NUCLEO-L476RG

CN10 HEADER
(top right side)



Memory Map



Internal 92KB SRAM1

I/O memory-mapping area

Internal 32KB SRAM2

Flash ROM area

GPIO Memory Address

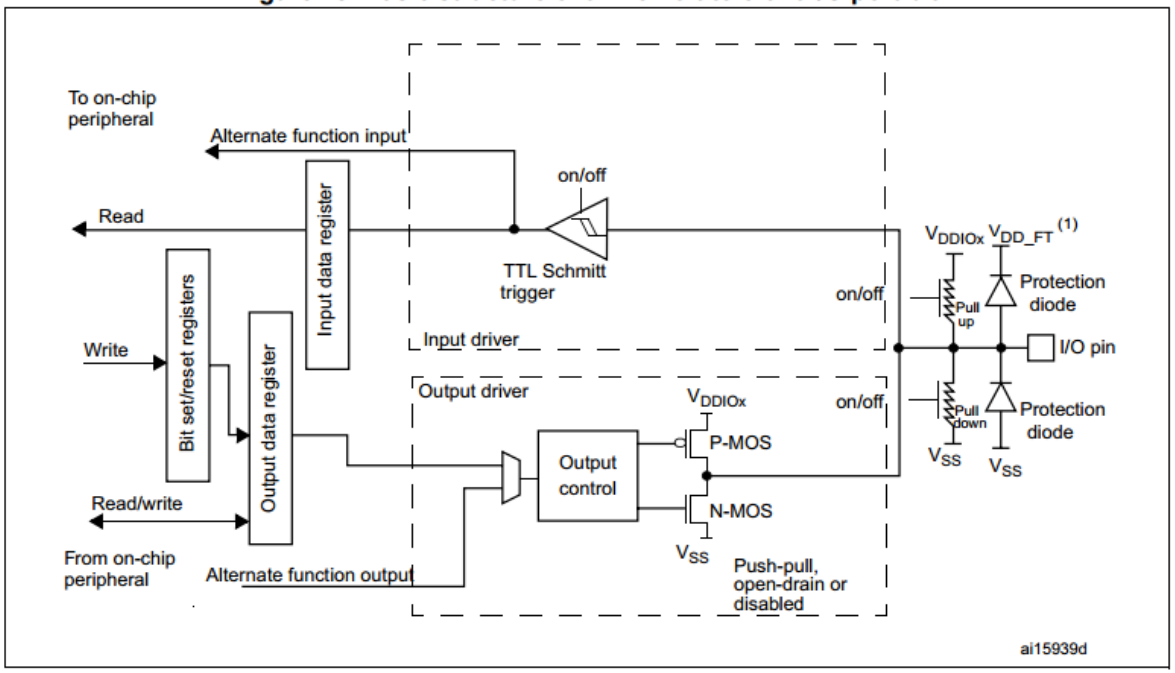
- In STM32L4 system all GPIO port connect on AHB2 bus
- Port A system memory address start from ***0x48000000***

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	-

GPIO Pin Structure

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 Registers

- Clock enable register
  - AHB2 peripheral clock enable register (RCC_AHB2ENR)
- Control registers
  - GPIO port mode register (GPIOx_MODER) (x =A..H)
  - GPIO port output type register (GPIOx_OTYPER) (x = A..H)
  - GPIO port output speed register (GPIOx_OSPEEDR)
  - GPIO port pull-up/pull-down register (GPIOx_PUPDR)
  - …
- Data registers
  - Output: GPIOx_ODR, 16bits
  - Input: GPIOx_IDR, 16bits

Output Signal Path

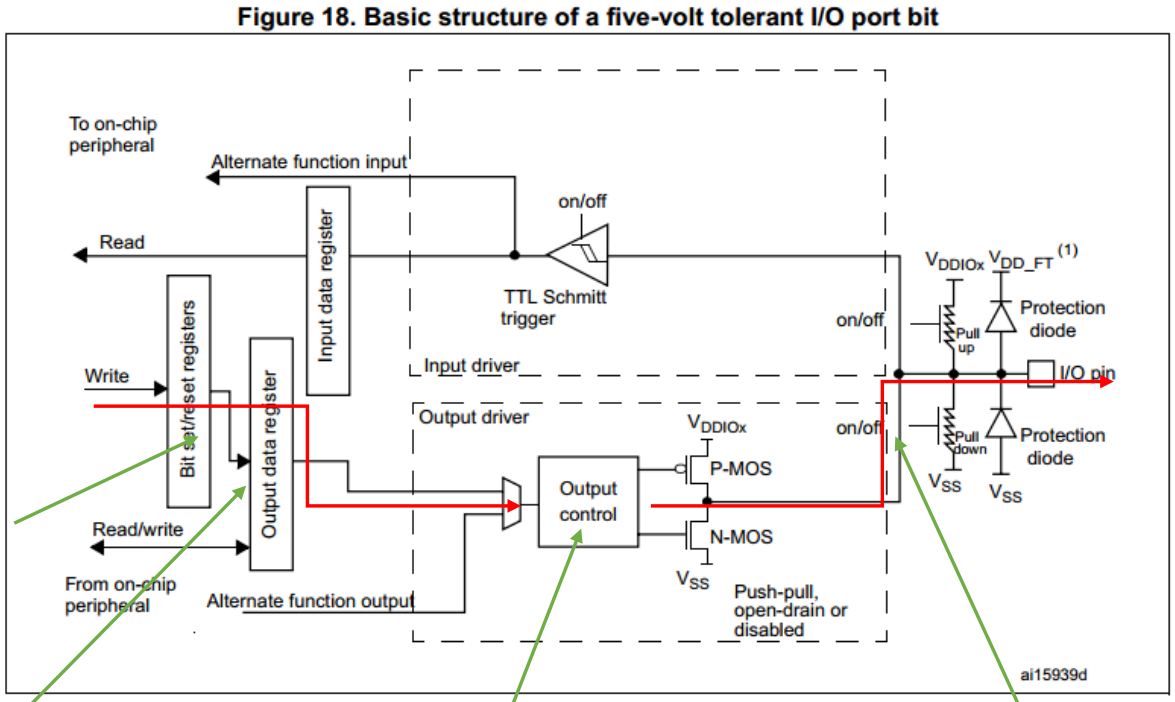


Figure 18. Basic structure of a five-volt tolerant I/O port bit

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

Bit set/reset register(BSRR)

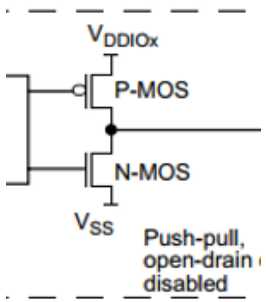
Output data register(ODR)

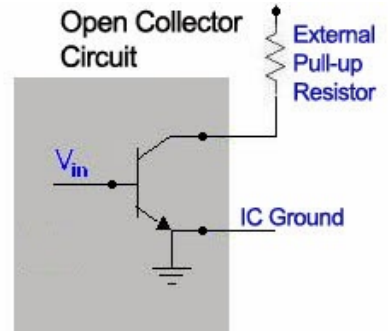
Output type register(OTYPER)

Pull-up control register(PUPDR)

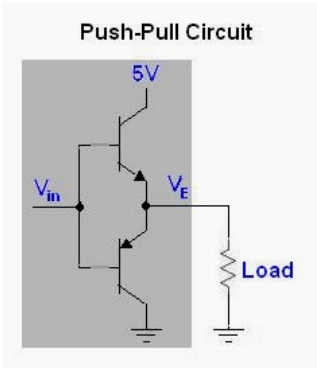
Push-Pull vs Open-Drain Output

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





OD



PP

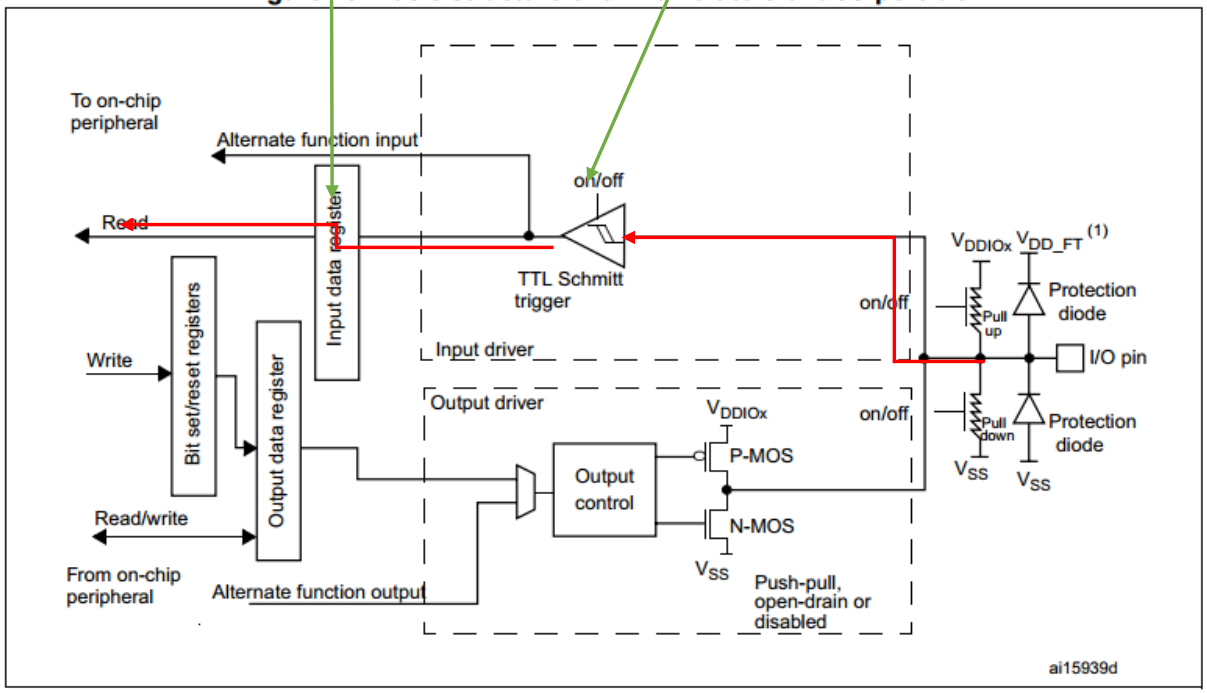
Ref.: http://cary1120.blogspot.tw/2013/11/open-drain-push-pull.html

Input Signal Path

Input data register(IDR)

Mode register(MODER)

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

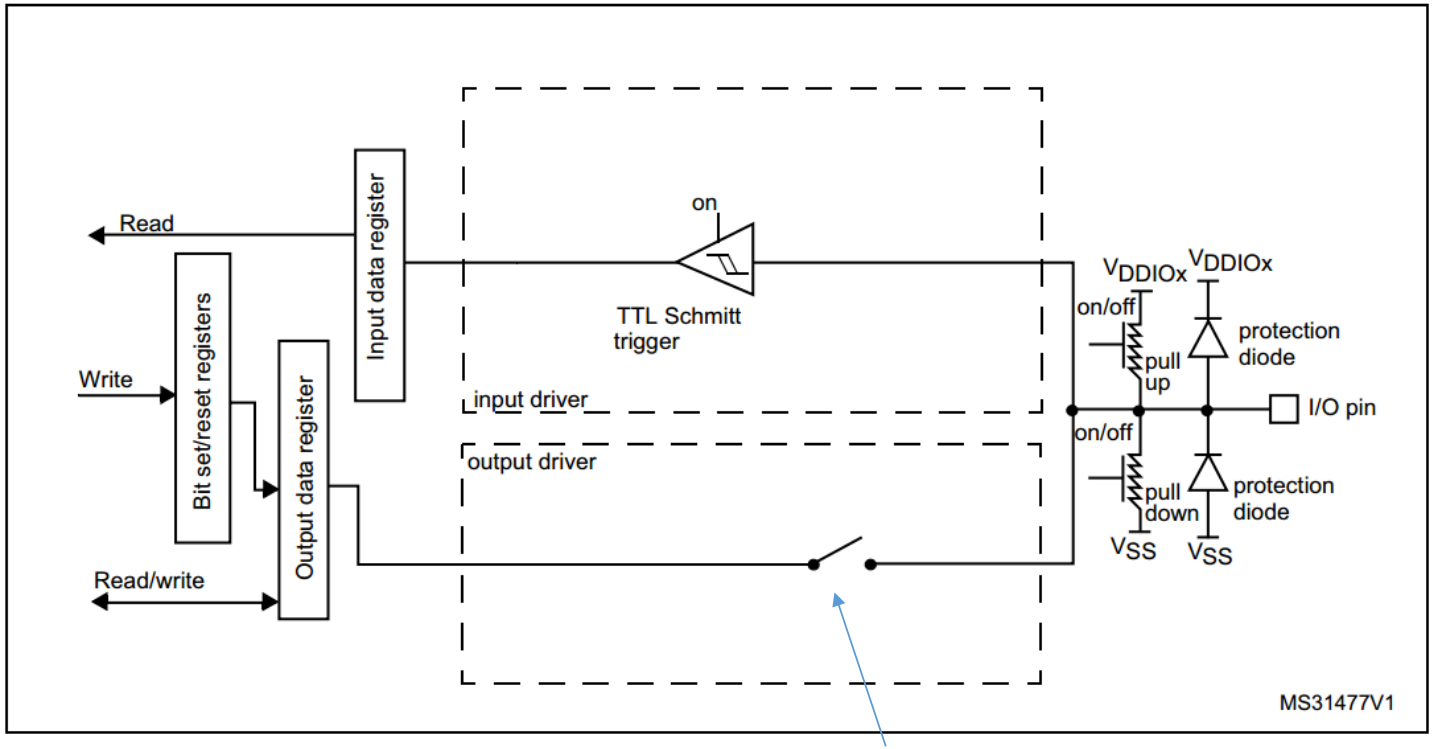


1. V_{DD_FT} is a potential specific to five-volt tolerant I/Os and different from V_{DD}.

Input Configure Example

- Mode=00
- PUPD= 00

Figure 19. Input floating/pull up/pull down configurations



Output disable

Basic GPIO Configuration

Output setting

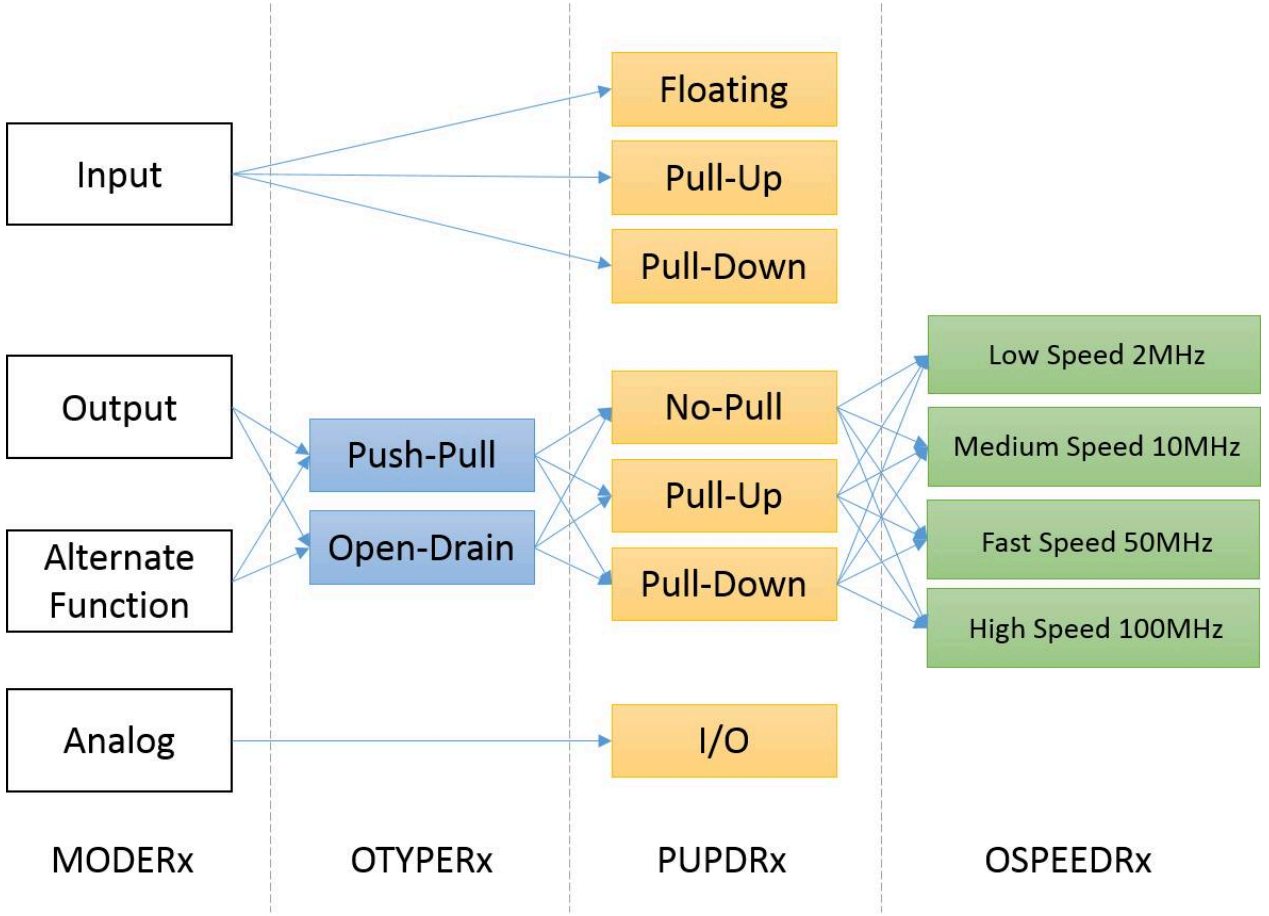
Input setting

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.

Configuration Reference



Reference: http://wiki.csie.ncku.edu.tw/embedded/GPIO

RCC_AHB2ENR

- Use for enable clock of GPIO bus

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

GPIOx_MODER

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

Address offset:0x00

Reset values:

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

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)

GPIOx_OTYPER

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

GPIOx_OSPEEDR

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.

GPIOx_PUPDR

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

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.	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.		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.		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

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

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

RCC_AHB2ENR

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

0x01

GPIOA_MODER

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

0xFFFF**F3**FF

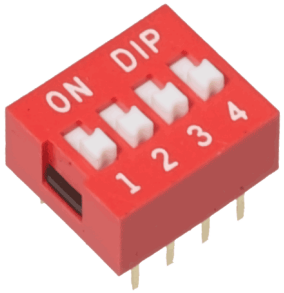
1 1 1 1 0 0 1 1 (F3)

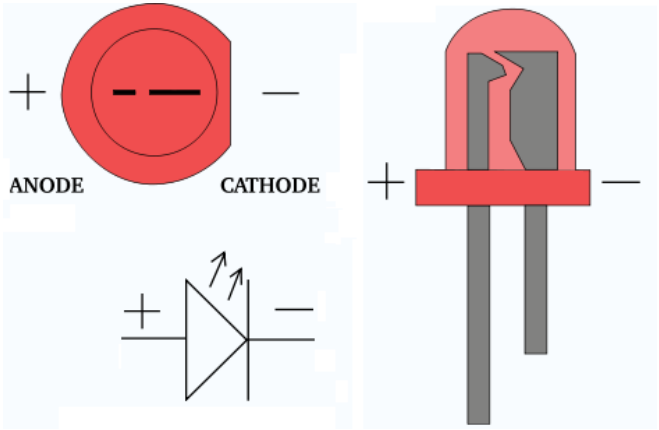
0 0 0 0 0 1 0 0 (04)

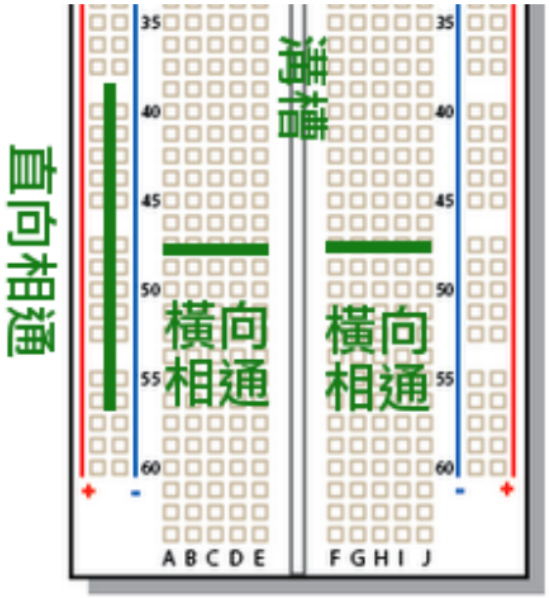
GPIOA_MODER = (GPIOA_MODER &0xFFFFF3FF) | 0x400

Lab 4 實驗零件

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

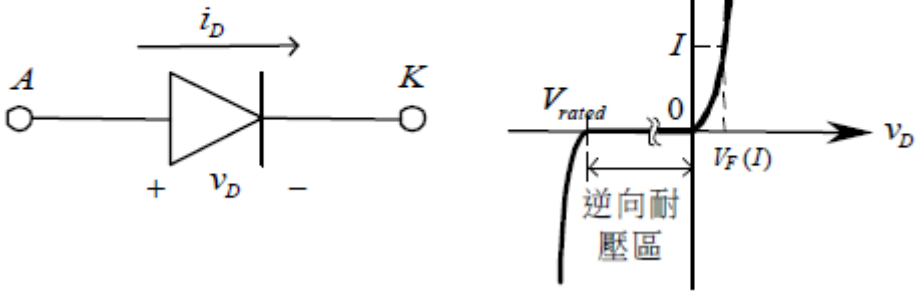


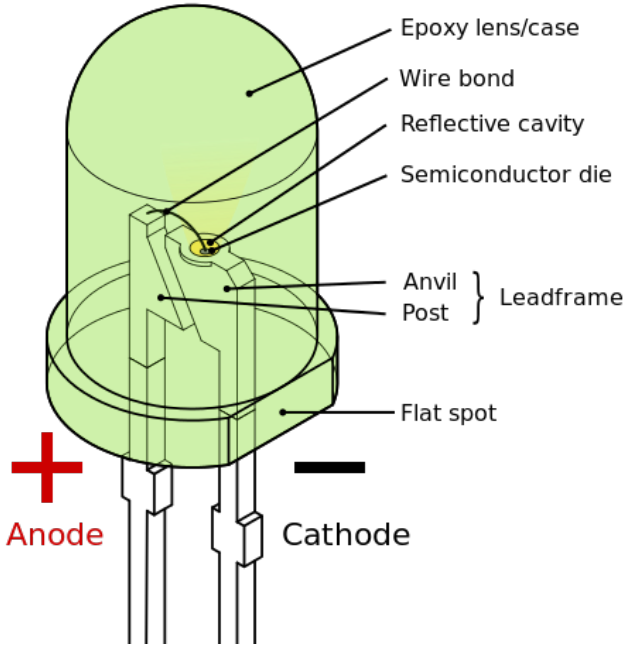




LED

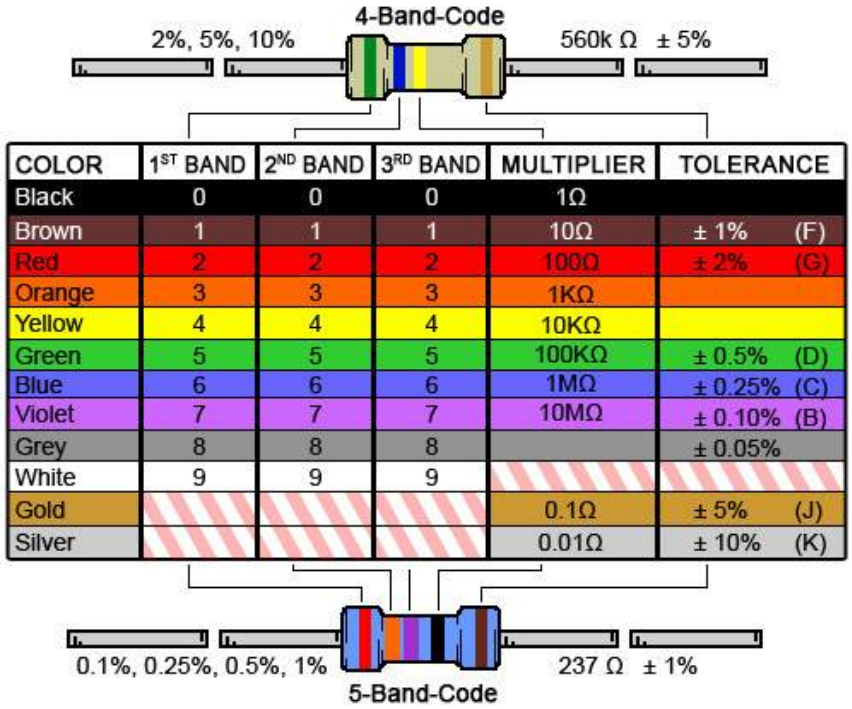
- 特性類似二極體，導通時發光，導通電壓約為0.3 or 0.7V
- 二極體內阻小，使用上通常會加上限流電阻避免LED燒毀





電阻

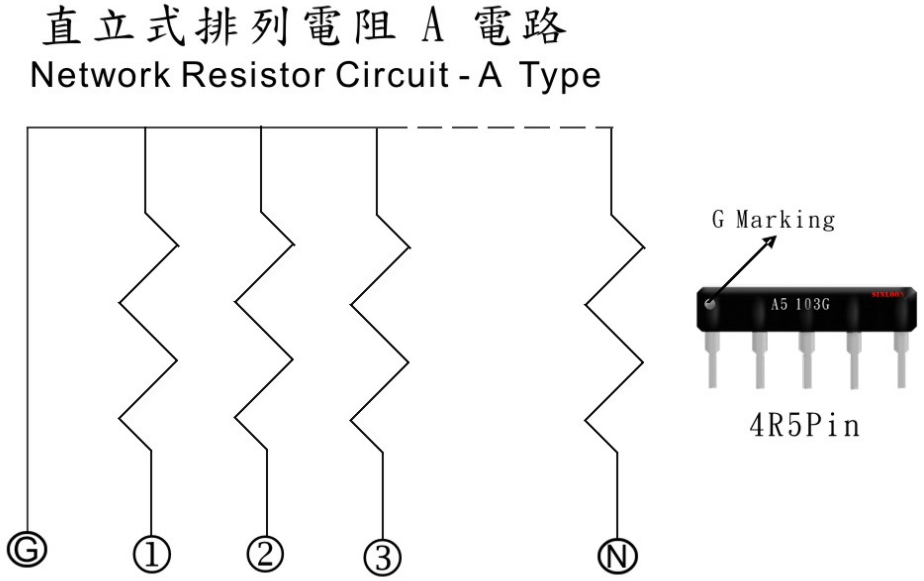
- 利用色碼標示電阻值



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)

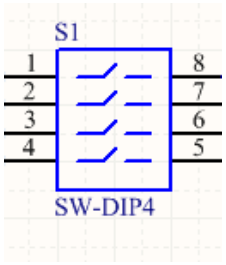
排阻

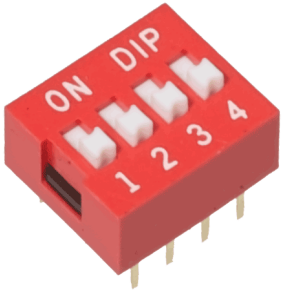
- 集合式電阻
- 用數字標記電阻值，例如：103=10*10^3 = 10K歐姆



DIP Switch

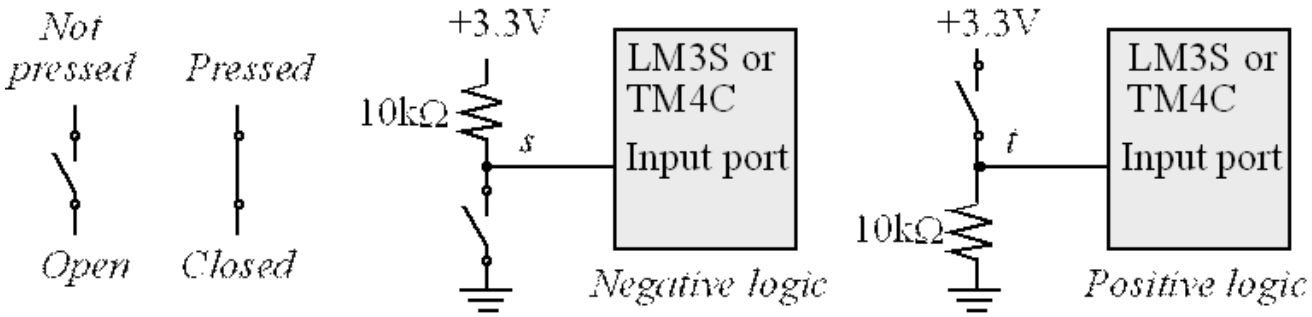
- 用途類似開關
- 當切到ON時PIN腳兩端連通





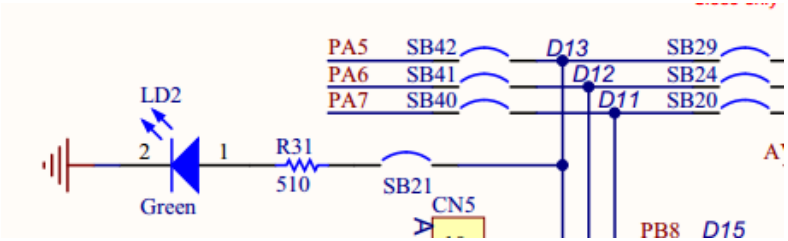
Negative logic and Positive logic

- 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 turn on single LED?

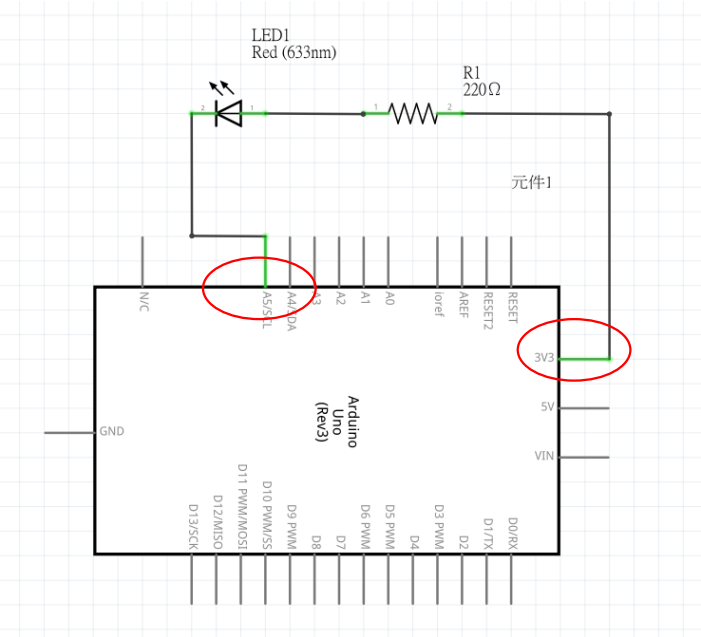
- Nucleo-L476RG has a onboard LED(LD2) connect at *GPIOA pin5* which is an active high circuit

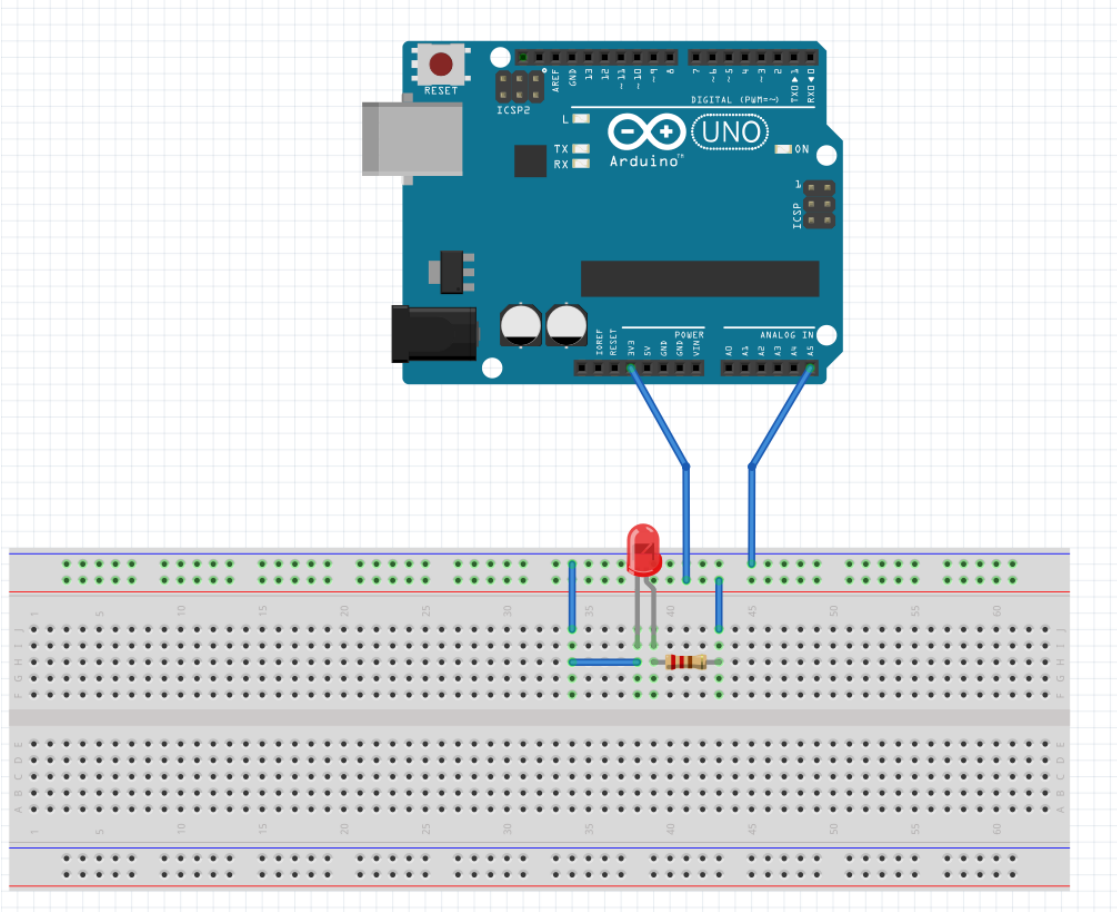



```
    .syntax unified
    .cpu cortex-m4
    .thumb

    .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
```

How to connect breadboard, LEDs and STM32

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





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

Set Selected Bits

- Through ODR

```
GPIOD_ODR |= 0x0080; // PD7 = 1
GPIOD_ODR &= ~0x0400; // PD10 = 0
```

- Through BSSR

```
LDR R0,=GPIOD ; GPIOD base address
MOV R1,#0x0080 ; select PD7
STRH R1,[R0,#BSSRL] ; set PD7 = 1
MOV R1,#0x0400 ;selectPD10
STRH R1,[R0,#BSSRH] ; reset PD10 = 0
```

```
/* BSRRL refer to bits 0-15 of BSRR which
set the corresponding bit of the port
bits 0 - 15 of a port */
/* BSRRH refer to bits 16-31 of BSRR
which reset the corresponding bit of the
port bits 0 - 15 of a port */
```

		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OD15	OD14	OD13	OD12	OD11	OD10	OD9	OD8	OD7	OD6	OD5	OD4	OD3	OD2	OD1	OD0
0x14	**GPIOx_ODR** (where x = A..H)																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Reset value																																
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

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**
1		**SUBS**	**R3, #1**
1 or 3		**BNE**	**L1**
1		**BX**	**LR**

Why ≅?

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

BNE L2 not taken

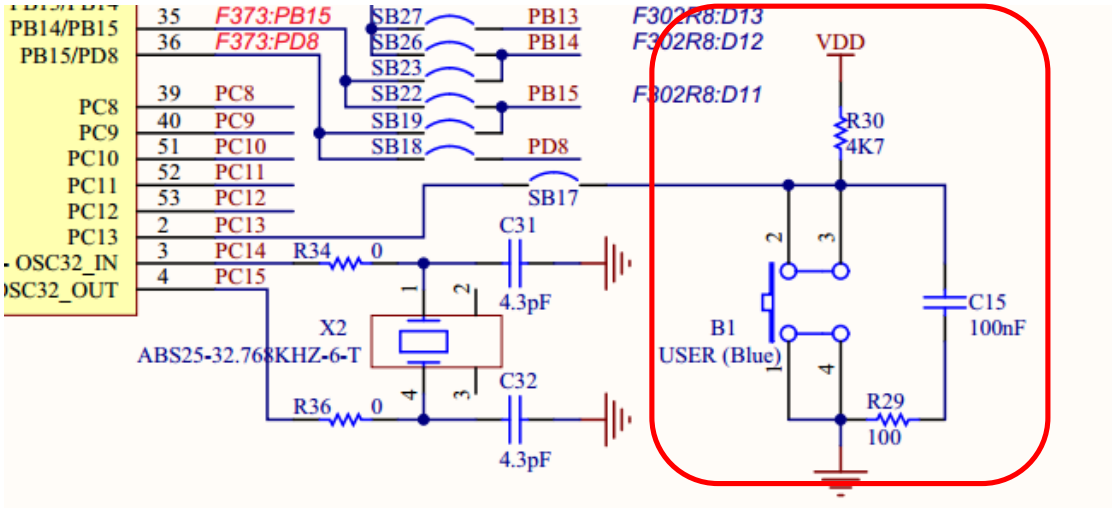
BNE L1 not taken

Branch Hazards

```
pre_gcd:
        push {R1}
        push {R0}
        bl GCD
L:
        b L
GCD:
        pop {R0}
        pop {R1}
        bx lr
```

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

Step1: Enable GPIO clock

Step3: Set PC13 MODER as input

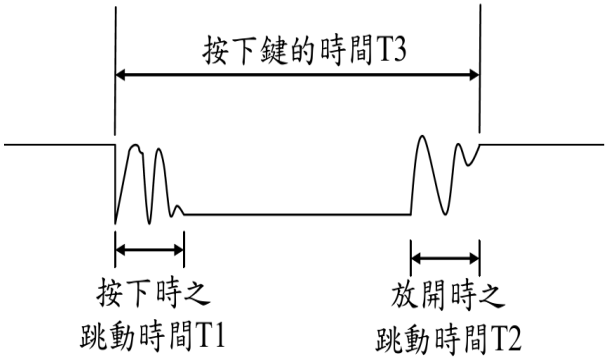
Step4: Read input data register

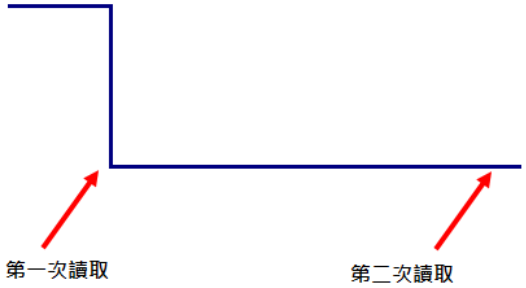
Step5: Compare PC13 value and do something

```
//Enable GPIOA&C clock
        movs    r0, #0x5
        ldr     r1, =RCC_AHB2ENR
        str     r0, [r1]
...
        //Set GPIOC Pin13 as input mode
        ldr     r1, =GPIOC_MODER
        ldr     r0, [r1]
        ldr     r2, =#0xF3FFFFFF
        and     r0, r2
        str     r0, [r1]
        //Set data register address
        ldr     r2, =GPIOC_IDR
        ldr     r3, [r2]
        movs    r4, #1
        lsl     r4, #13
        ands    r3, r4
        beq     do_pushed
```

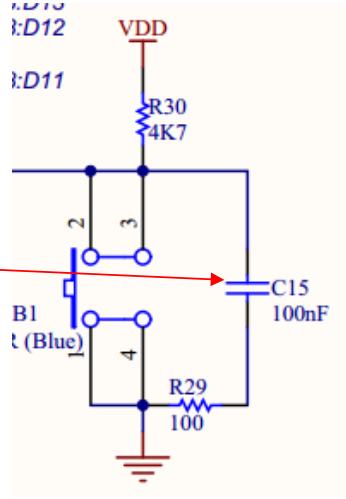
Note: Input預設為floating狀態且不需設Speed register

Debounce





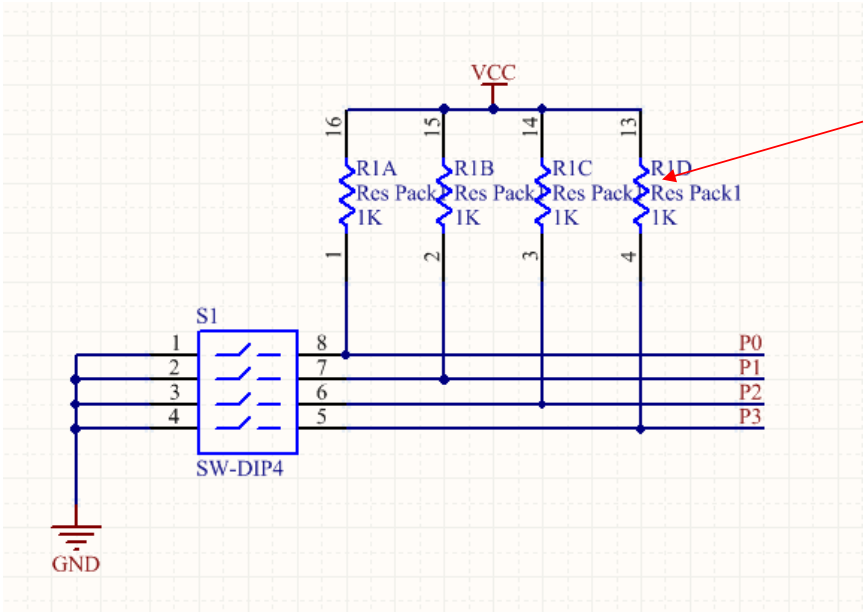
- Hardware method
  - Add a 濾波電容
- Software method
  - 讀取GPIO Pin後間隔一段時間再讀取一次確認
  - 連續讀取N次, 看讀值是否穩定無改變



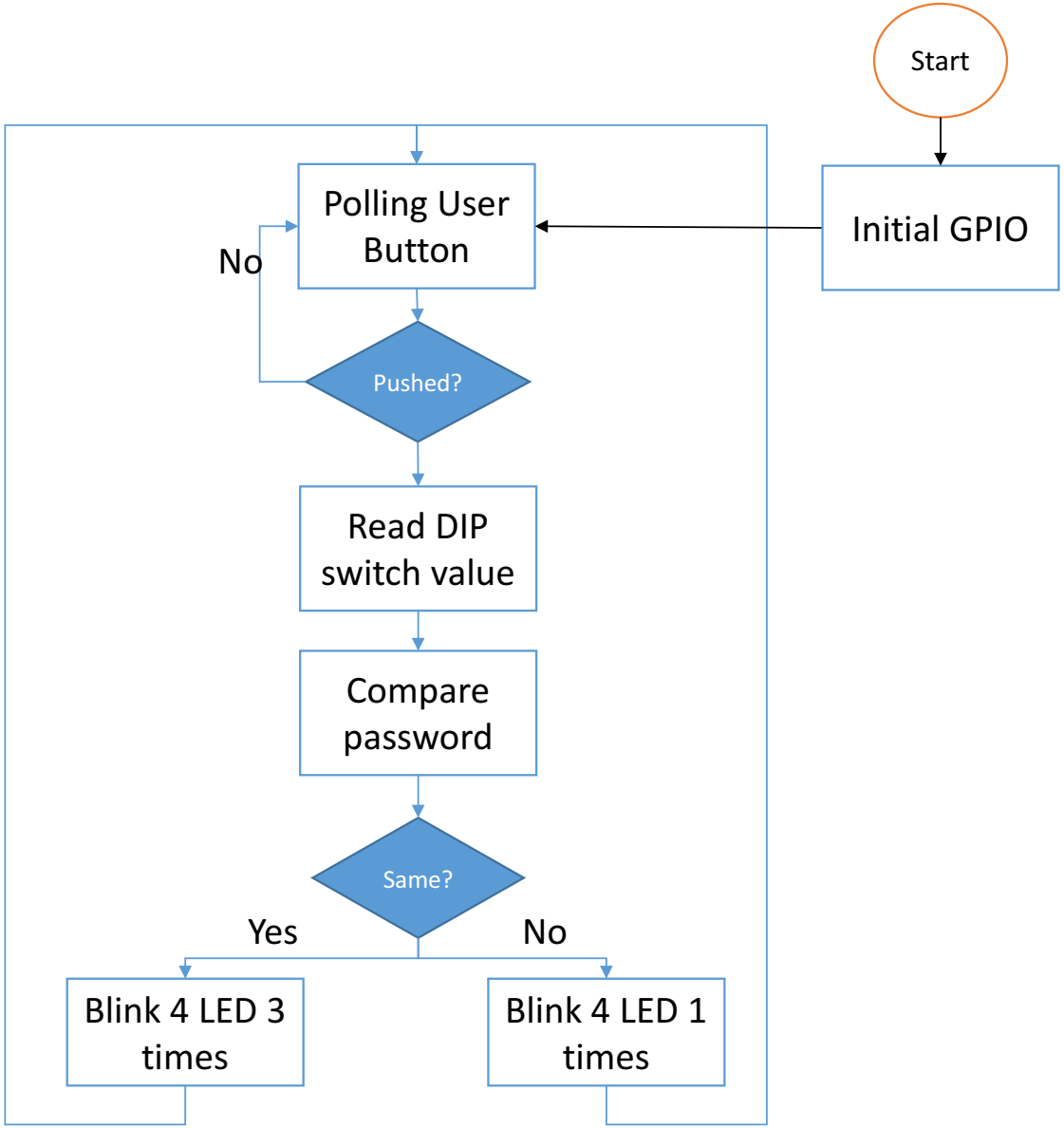
How to connect DIP switch and STM32

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