“본 강의 동영상 및 자료는 대한민국 저작권법을 준수합니다. 본 강의 동영상 및 자료는 상명대학교 재학생들의 수업목적으로 제작 · 배포되는 것이므로, 수업목적으로 내려받은 강의 동영상 및 자료는 수업목적 이외에 다른 용도로 사용할 수 없으며, 다른 장소 및 타인에게 복제, 전송하여 공유할 수 없습니다. 이를 위반해서 발생하는 모든 법적 책임은 행위 주체인 본인에게 있습니다.”

피지컬 컴퓨팅

Lec. 4. GPIO: LED & Switch

Heenam Yoon

Department of
Human-Centered Artificial Intelligence

E-mail) h-yoon@smu.ac.kr
Room) 0112

IAR 프로젝트 설정

- 주차별로 새로 구성하기

- 참조하는 파일들이 달라짐
- +익숙해지기 위해

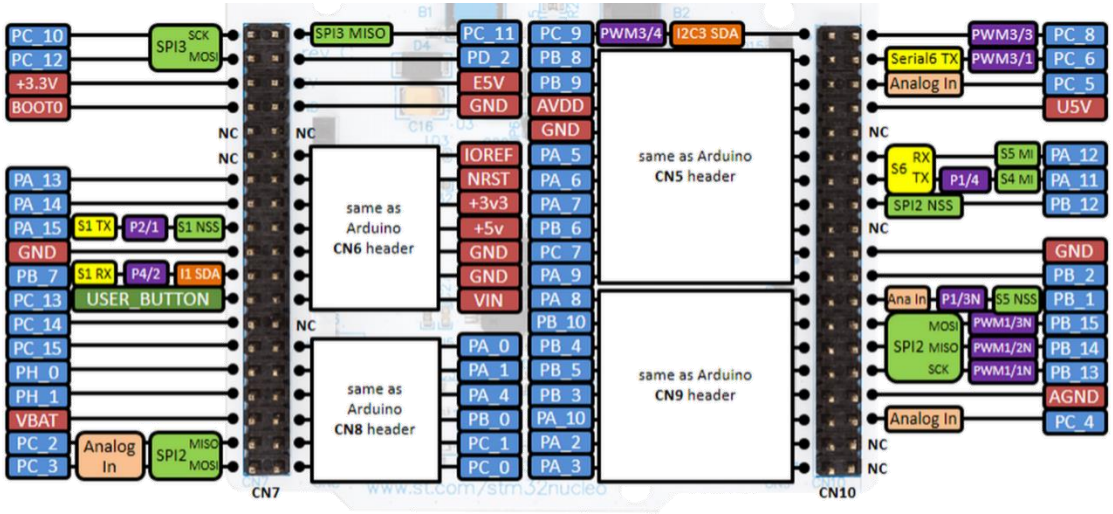
- 오늘은 지난시간의 연장이므로 지난 IAR프로젝트에 이어서 하겠습니다

Internal Use Only

Review: GPIO

- MCU는 우리가 어떤 포트 & 핀을 어떤 목적으로 사용할지 모른다
- 어디를 어떻게 사용할지 알려주어야 함 = 레지스터 설정

- Port / Pin
- Input / output / alternate function / analog

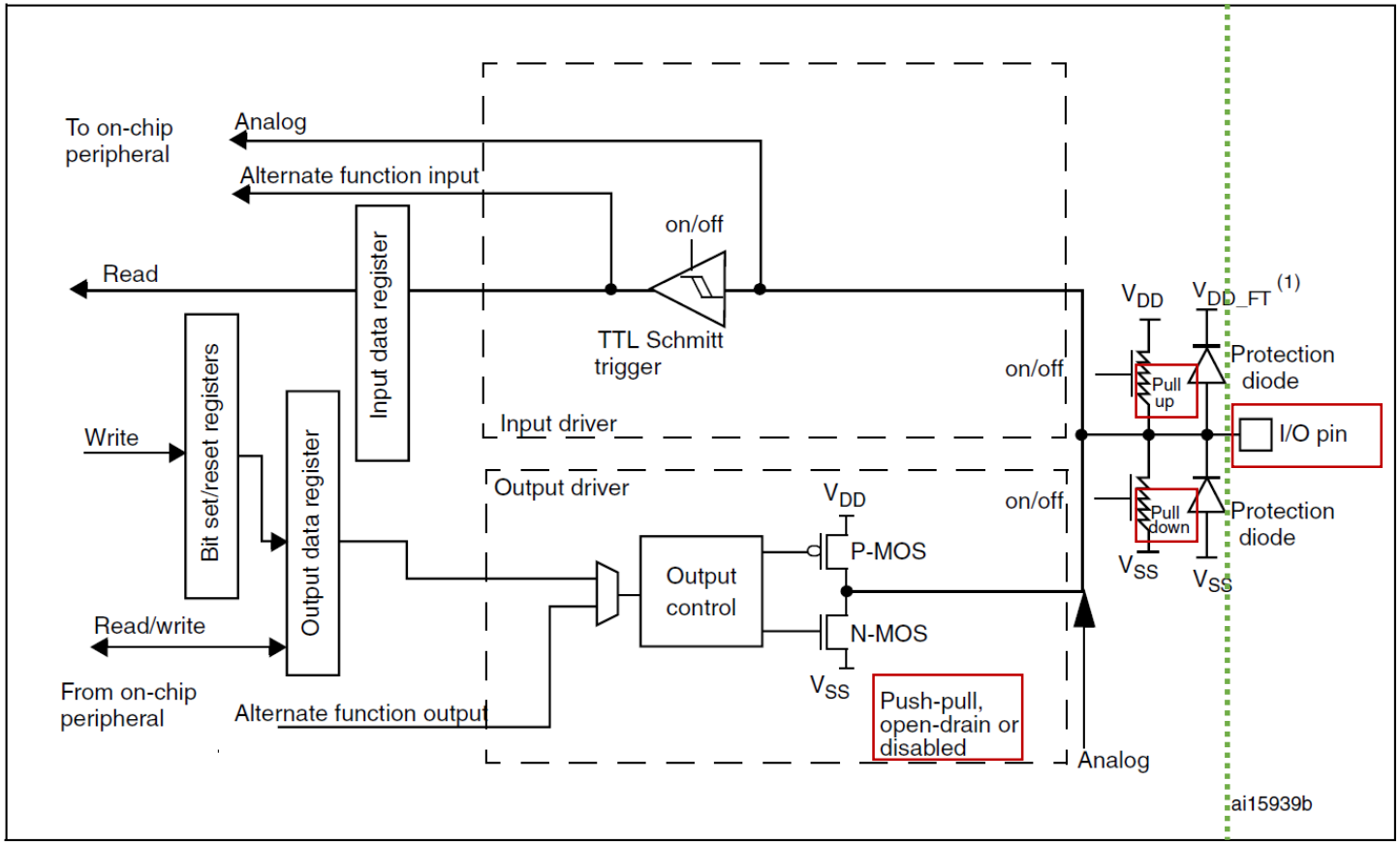


Internal Use Only

상명대학교 SANGMYUNG UNIVERSITY

Review: GPIO

Figure 25. Basic structure of a five-volt tolerant I/O port bit



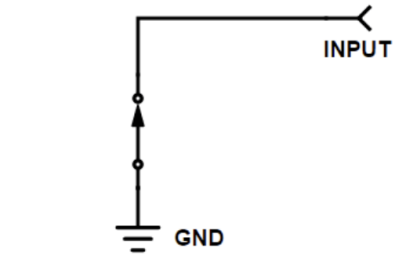
1. V_{DD_FT} is a potential specific to five-volt tolerant I/Os and different from V_{DD}.

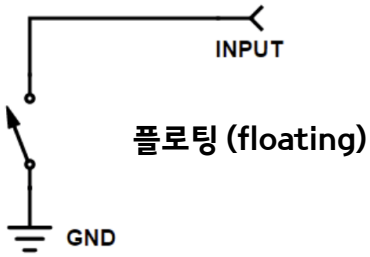
Internal Use Only

상명대학교 SANGMYUNG UNIVERSITY

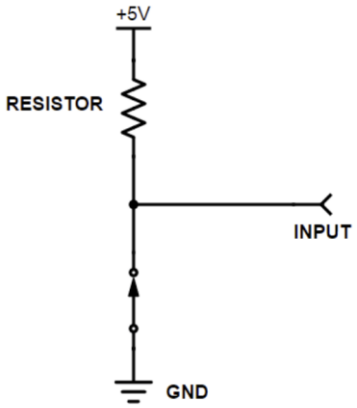
Review: GPIO

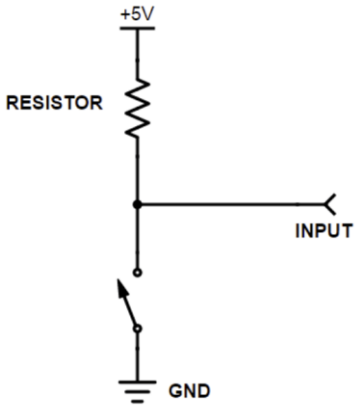
- Input
  - pull-up / pull-down





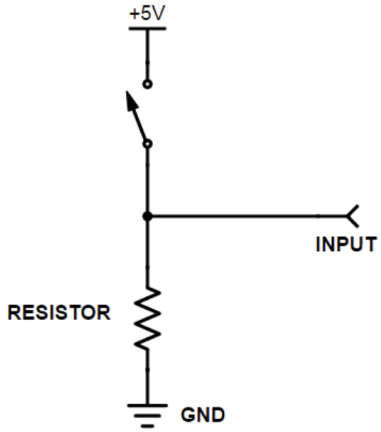
플로팅 (floating)





pull-up

스위치를 당겼더니 (pull), 공급 전압이 5v (up)가 됐다



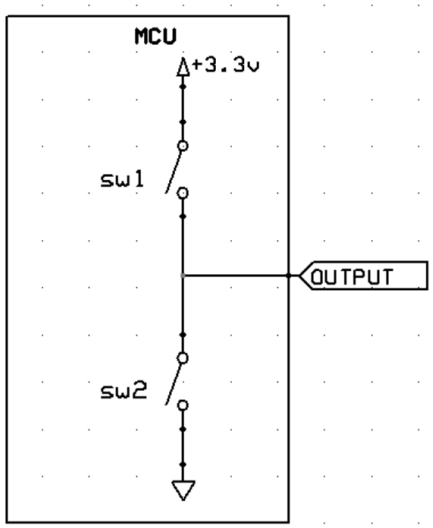
pull-down

스위치를 당겼더니 (pull), 공급 전압이 0v (down)가 됐다

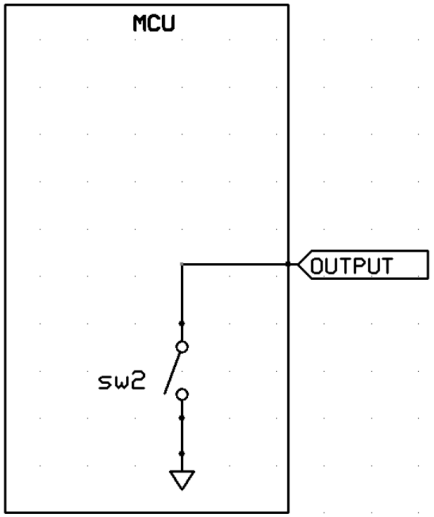
Internal Use Only

Review: GPIO

- Output
  - push-pull / open-drain
  - pull-up / pull-down
  - …



push-pull



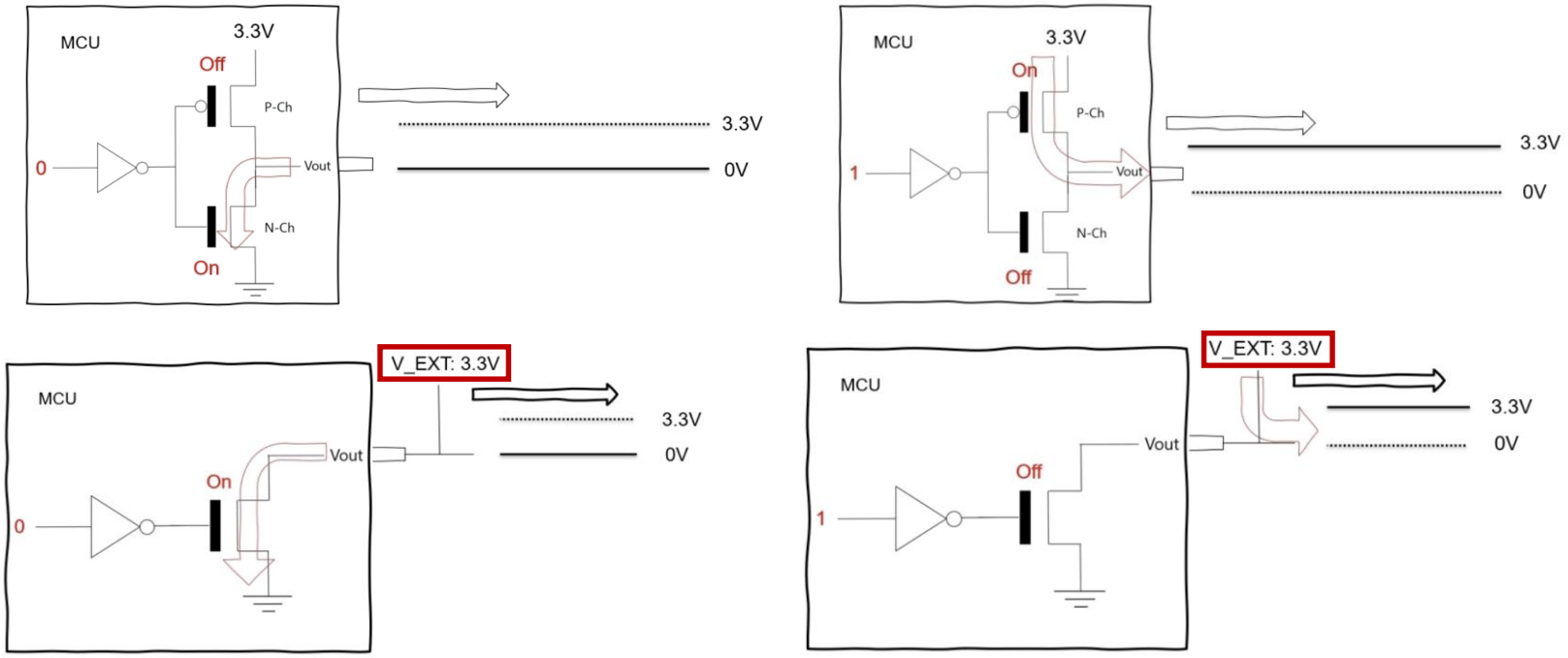
open-drain

Internal Use Only

상명대학교 SANGMYUNG UNIVERSITY

Review: GPIO

- Output
  - push-pull / open-drain
  - pull-up / pull-down
  - …



Internal Use Only

상명대학교 SANGMYUNG UNIVERSITY

Review: GPIO

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

MODER(i) [1:0]	OTYPER(i)	OSPEEDR(i) [B:A]	PUPDR(i) [1:0]		I/O configuration	
01	0	SPEED [B:A]	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)	

Internal Use Only

상명대학교 SANGMYUNG UNIVERSITY

Review: GPIO

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

MODER(i) [1:0]	OTYPER(i)	OSPEEDR(i) [B:A]		PUPDR(i) [1:0]		I/O configuration	
10	0	SPEED [B:A]		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		

Internal Use Only

Review: GPIO

GPIO registers

- **GPIO port mode register (GPIOx_MODER) (x = A..I/J/K)**

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

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

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

00: Input (reset state)

01: General purpose output mode

10: Alternate function mode

11: Analog mode

상명대학교 SANGMYUNG UNIVERSITY

Review: GPIO

GPIO registers

- **GPIO port output type register (GPIOx_OTYPER) (x = A..I/J/K)**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
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 output type of the I/O port.

0: Output push-pull (reset state)

1: Output open-drain

상명대학교 SANGMYUNG UNIVERSITY

Review: GPIO

GPIO registers

- GPIO port pull-up/pull-down register (GPIOx_PUPDR)(x = A..I/J/K)

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

Bits 2y:2y+1 **PUPDRy[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

상명대학교 SANGMYUNG UNIVERSITY

Review: GPIO

GPIO registers

- **GPIO port input data register (GPIOx_IDR) (x = A..I/J/K)**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IDR15	IDR14	IDR13	IDR12	IDR11	IDR10	IDR9	IDR8	IDR7	IDR6	IDR5	IDR4	IDR3	IDR2	IDR1	IDR0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

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

Bits 15:0 **IDRy**: Port input data (y = 0..15)

These bits are read-only and can be accessed in word mode only. They contain the input value of the corresponding I/O port.

상명대학교 SANGMYUNG UNIVERSITY

Review: GPIO

GPIO registers

- **GPIO port output data register (GPIOx_ODR) (x = A..I/J/K)**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ODR15	ODR14	ODR13	ODR12	ODR11	ODR10	ODR9	ODR8	ODR7	ODR6	ODR5	ODR4	ODR3	ODR2	ODR1	ODR0
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 **ODRy**: Port output data (y = 0..15)

These bits can be read and written by software.

Note: For atomic bit set/reset, the ODR bits can be individually set and reset by writing to the GPIOx_BSRR register (x = A..I/J/K).

상명대학교 SANGMYUNG UNIVERSITY

Review: GPIO

GPIO registers

- GPIO port bit set/reset register (GPIOx_BSRR) (x = A..I/J/K)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BS15	BS14	BS13	BS12	BS11	BS10	BS9	BS8	BS7	BS6	BS5	BS4	BS3	BS2	BS1	BS0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:16 **BRy:** Port x reset bit y (y = 0..15)

These bits are write-only and can be accessed in word, half-word or byte mode. A read to these bits returns the value 0x0000.

0: No action on the corresponding ODRx bit

1: Resets the corresponding ODRx bit

Note: If both BSx and BRx are set, BSx has priority.

Bits 15:0 **BSy:** Port x set bit y (y= 0..15)

These bits are write-only and can be accessed in word, half-word or byte mode. A read to these bits returns the value 0x0000.

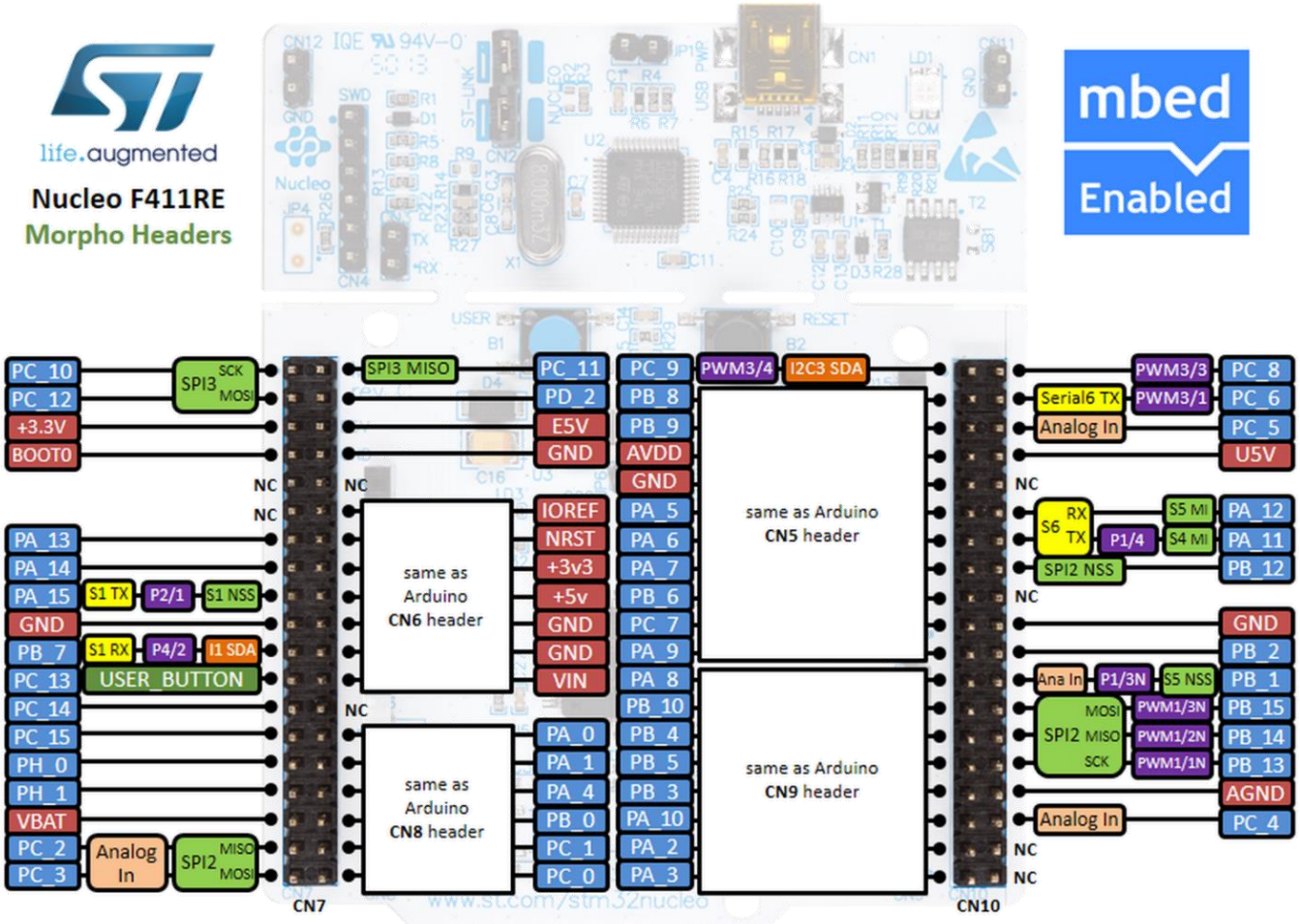
0: No action on the corresponding ODRx bit

1: Sets the corresponding ODRx bit

상명대학교 SANGMYUNG UNIVERSITY

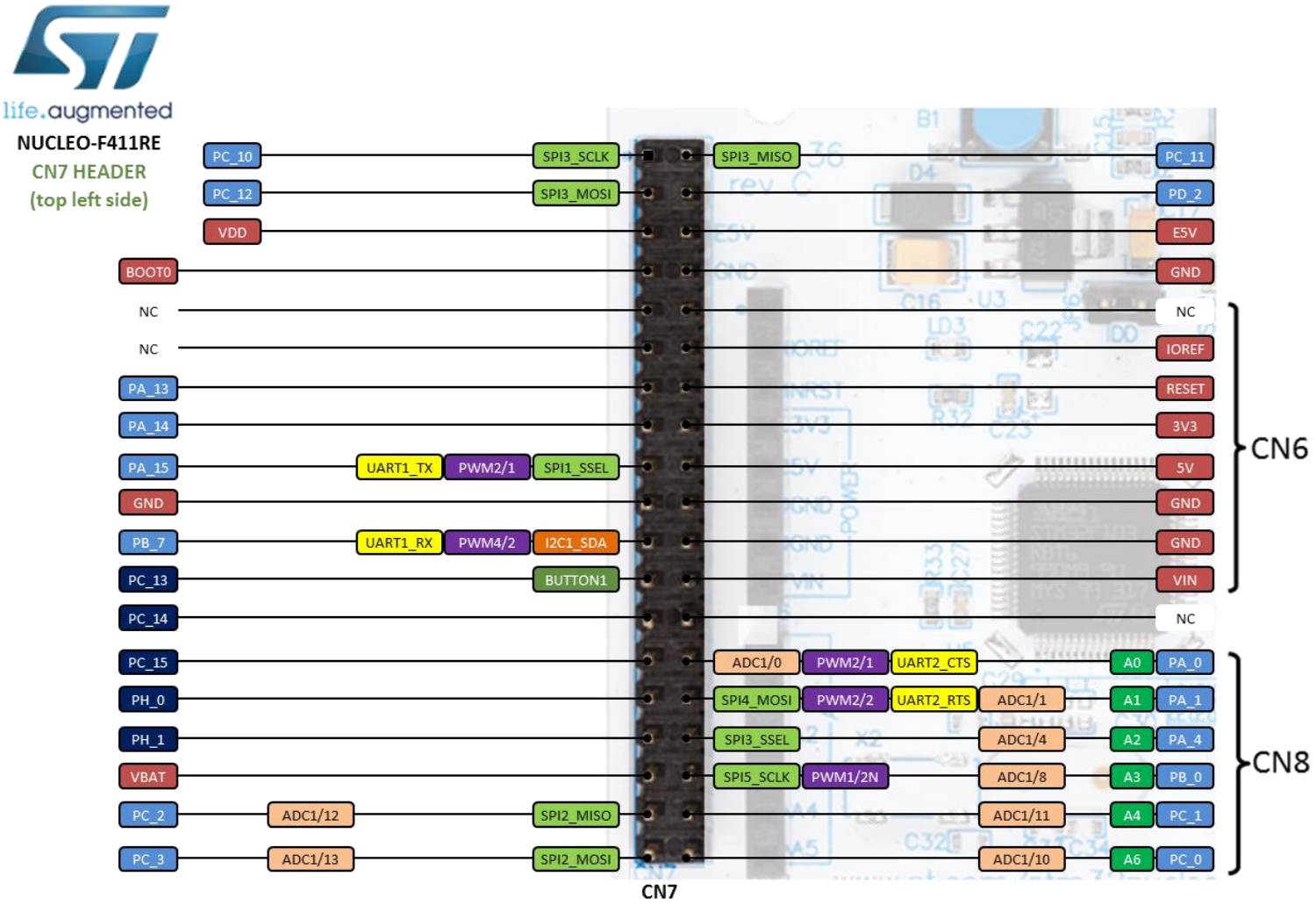
Review: GPIO

내부 LED 제어



Review: GPIO

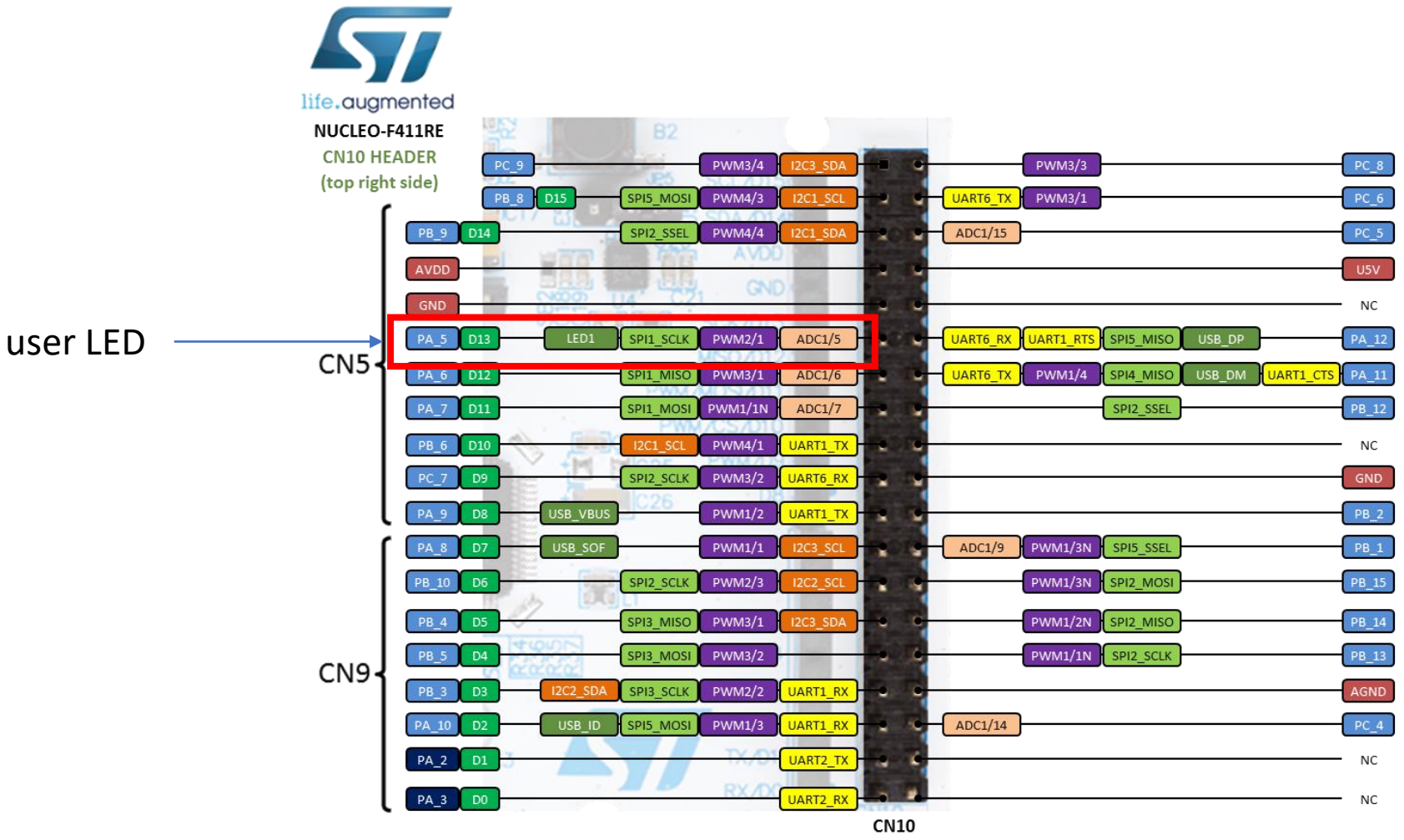
내부 LED 제어



상명대학교 SANGMYUNG UNIVERSITY

Review: GPIO

user LED 제어



Review: GPIO

user LED 제어 코드 설명

- Output
  - push-pull / open-drain
  - pull-up / pull-down

- User LED에 전원을 공급하면 LED가 켜질 것
- User LED는 Port A 5번 핀에 연결되어 있음
- MCU는 Port A 5번 핀에 전원을 공급해야 함
- 즉, Port A 5번 핀을 output으로 사용해야 함

```
#include "stm32f4xx.h"

void Delay(__IO uint32_t nCount)
{
 for(; nCount != 0; nCount--);
}

int main()
{
  GPIO_InitTypeDef GPIO_InitStructure;

  RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOA, ENABLE);

  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_OUT;
  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_5;
  GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
  GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL;

  GPIO_Init(GPIOA, &GPIO_InitStructure);

  while(1)
  {
   GPIO_SetBits(GPIOA, GPIO_Pin_5);
   Delay(5000000);
   GPIO_ResetBits(GPIOA, GPIO_Pin_5);
   Delay(5000000);
  }

 return 0;
}
```

Port를 MCU와 동기화하여 사용하기 위해 clock 공급

5번 핀을 Output으로 사용
Push-pull을 이용
Pull-up/down은 사용X

위 설정 정보를 Port A에 적용하라

Port A 5번 핀에 값을 주어라

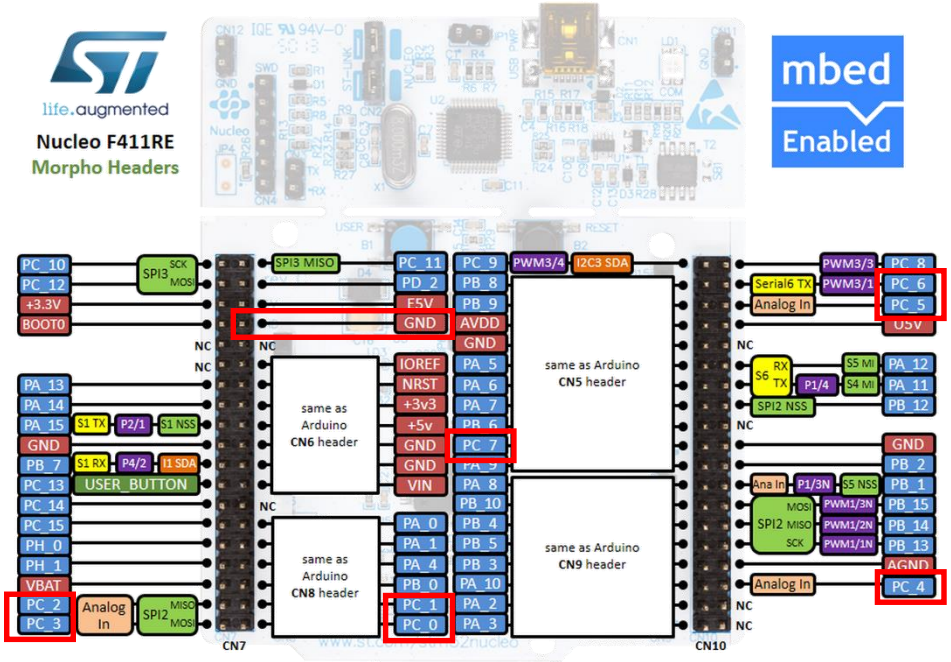
Port A 5번 핀에 값을 지워라

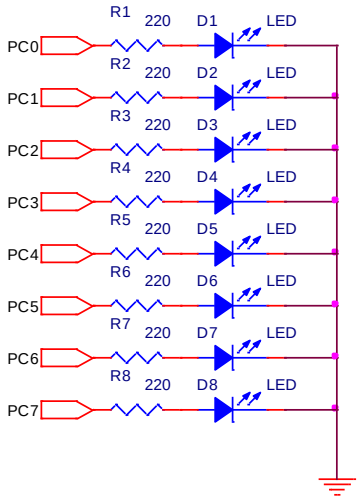
Delay함수
숫자를 조절하면 깜박이 속도가 달라짐

GPIO: 실습 1

외부 LED 제어

- HW 구성
  - 저항(220Ω), LED를 아래와 같이 연결
  - 포트 C PC0~PC7을 각각 LED와 연결

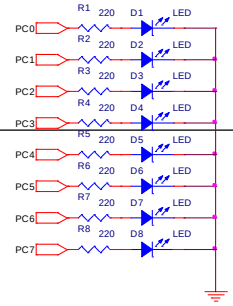


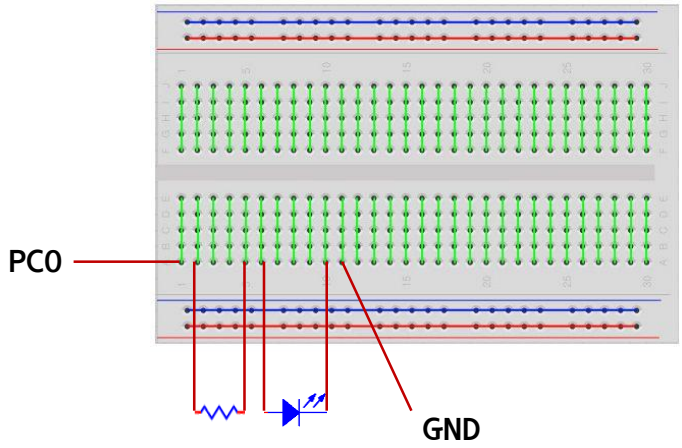


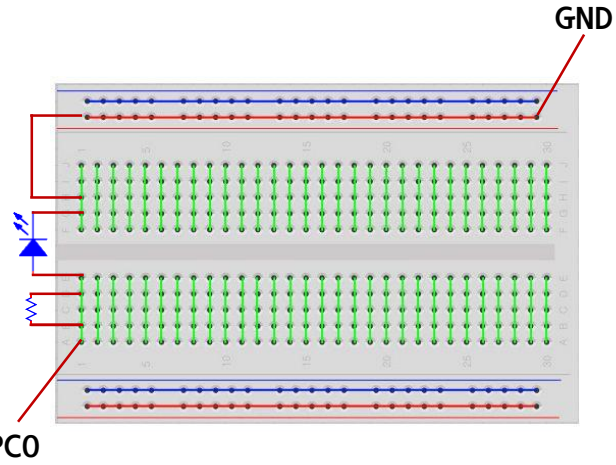
상명대학교 SANGMYUNG UNIVERSITY

GPIO: 실습 1

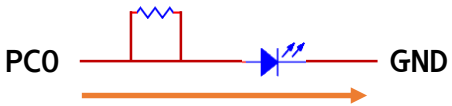
외부 LED 제어



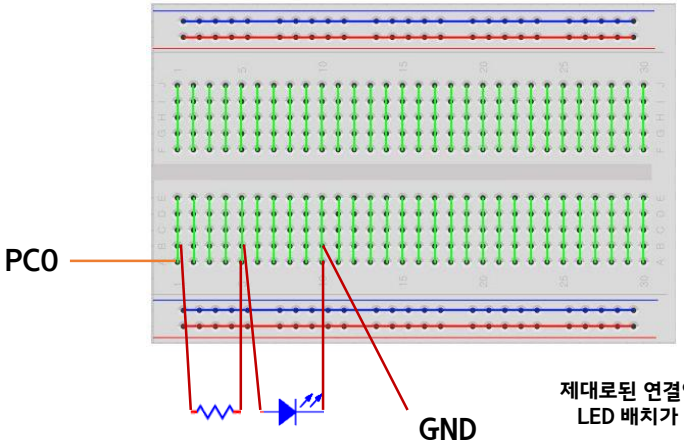




아무것도 연결되어 있지 않음



전류의 흐름
(굳이 저항이 있는 곳으로
갈 필요 없음 = 저항의
의미가 없음)

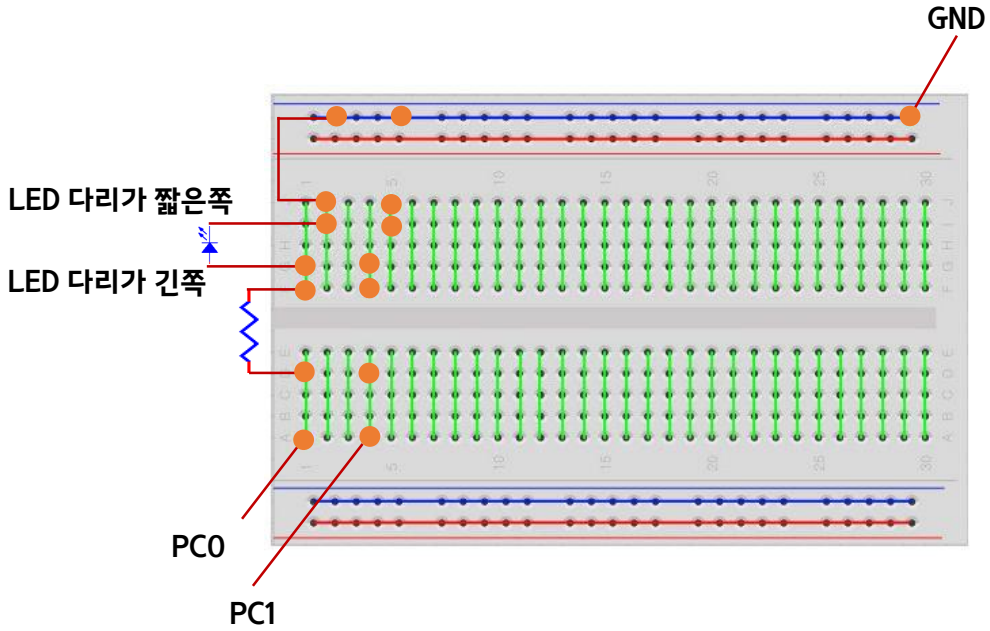


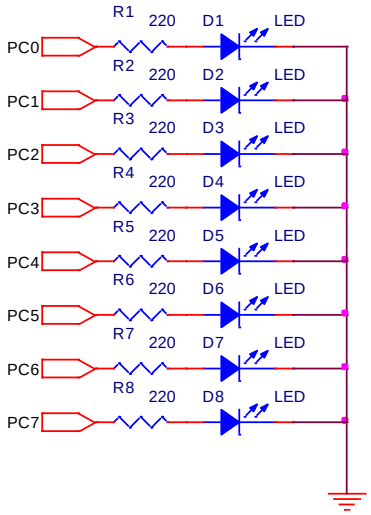
제대로된 연결인데 8개를
LED 배치가 어려워짐

상명대학교 SANGMYUNG UNIVERSITY

GPIO: 실습 1

외부 LED 제어, HW 구성 제안 (다른 방법도 있음)





상명대학교 SANGMYUNG UNIVERSITY

GPIO: 실습 1

외부 LED 제어, Software

- LED가 on/off됨 = 전원 공급 on / off
- Port C의 0~ 7번 핀을 "Output"으로 사용

- 고려해야 할 레지스터?

상명대학교 SANGMYUNG UNIVERSITY

GPIO: 실습 1-1

외부 LED 제어, Software

- 8개의 LED가 ON-OFF 되는 프로그램 작성

```
#include "stm32f4xx.h"

void Delay(__IO uint32_t nCount)
{
  for(; nCount != 0; nCount--);
}

int main()
{
   GPIO_InitTypeDef GPIO_InitStructure;

   RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOC, ENABLE);

   GPIO_InitStructure.GPIO_Mode = GPIO_Mode_OUT;
   GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0|GPIO_Pin_1|GPIO_Pin_2|GPIO_Pin_3|GPIO_Pin_4|GPIO_Pin_5|GPIO_Pin_6|GPIO_Pin_7;
   GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
   GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL;

   GPIO_Init(GPIOC, &GPIO_InitStructure);

   while(1)
   {
    GPIO_Write(GPIOC, 0x0000);
    Delay(5000000);
    GPIO_Write(GPIOC, 0x00FF);
    Delay(5000000);
   }

  return 0;
}
```


상명대학교 SANGMYUNG UNIVERSITY

GPIO: 실습 1-1

외부 LED 제어, Software

- 8개의 LED가 순차적으로 하나씩 켜지도록 프로그램 수정

```
while(1)
  {
    GPIO_Write(GPIOC, GPIO_Pin_0);
    Delay(500000);
    GPIO_Write(GPIOC, GPIO_Pin_1);
    Delay(500000);
    GPIO_Write(GPIOC, GPIO_Pin_2);
    Delay(500000);
    GPIO_Write(GPIOC, GPIO_Pin_3);
    Delay(500000);
    GPIO_Write(GPIOC, GPIO_Pin_4);
    Delay(500000);
    GPIO_Write(GPIOC, GPIO_Pin_5);
    Delay(500000);
    GPIO_Write(GPIOC, GPIO_Pin_6);
    Delay(500000);
    GPIO_Write(GPIOC, GPIO_Pin_7);
    Delay(500000);
  }
```

GPIO_Pin_0 vs. 0x0001

상명대학교 SANGMYUNG UNIVERSITY

GPIO: 실습 1-1

참고: GPIO_SetBits(), GPIO_Write() 차이

- 하나의 포트는 16개의 핀이 있음 (0 ~ 15 핀)

PC15	PC14	PC13	PC12	PC11	PC10	PC9	PC8	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0
1	1	1	1	0	0	0	0	1	1	0	0	0	0	0	0

2진수 표현: 1111 0000 1100 0000

16진수 표현: 0xF0C0

15, 14, 13, 12, 7, 6에 1로 세팅되어 있음

상명대학교 SANGMYUNG UNIVERSITY

GPIO: 실습 1-1

참고: GPIO_SetBits(), GPIO_Write() 차이

- 두 함수 모두 해당 핀의 값을 설정하는 것

PC15	PC14	PC13	PC12	PC11	PC10	PC9	PC8	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

GPIO_SetBits(GPIOC, GPIO_Pin_5); **= GPIO_SetBits(GPIOC, 0x0020);**

PC15	PC14	PC13	PC12	PC11	PC10	PC9	PC8	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0
0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0

GPIO_SetBits(GPIOC, GPIO_Pin_4); **= GPIO_SetBits(GPIOC, 0x0010);**

PC15	PC14	PC13	PC12	PC11	PC10	PC9	PC8	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0
0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0

레지스터 값이 누적됨. 결과적으로 GPIO_SetBits(GPIOC, 0x0030);과 같아짐

따라서, GPIO_ResetBits() 함수와 함께 사용

상명대학교 SANGMYUNG UNIVERSITY

GPIO: 실습 1-1

참고: GPIO_SetBits(), GPIO_Write() 차이

- 두 함수 모두 해당 핀의 값을 설정하는 것

PC15	PC14	PC13	PC12	PC11	PC10	PC9	PC8	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

GPIO_Write(GPIOC, GPIO_Pin_5); = GPIO_ Write(GPIOC, 0x0020);

PC15	PC14	PC13	PC12	PC11	PC10	PC9	PC8	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0
0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0

GPIO_ Write(GPIOC, GPIO_Pin_4); = GPIO_ Write(GPIOC, 0x0010);

PC15	PC14	PC13	PC12	PC11	PC10	PC9	PC8	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0
0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

레지스터 값이 덮어 씌워짐

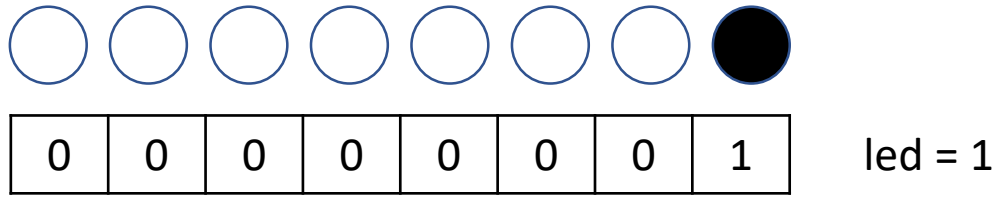
상명대학교 SANGMYUNG UNIVERSITY

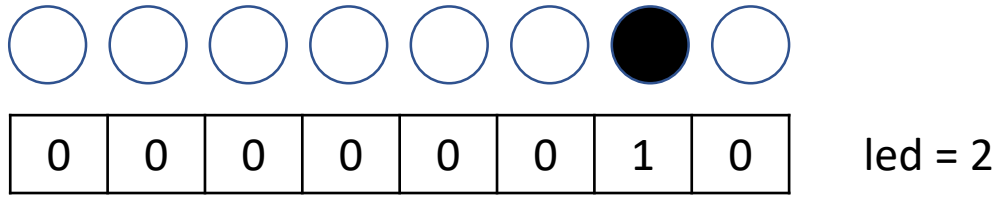
GPIO: 실습 1-2

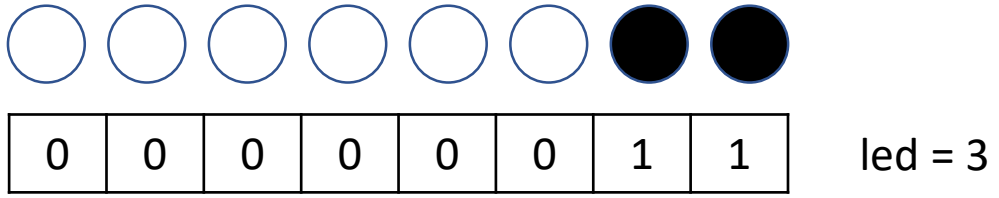
외부 LED 제어, Software

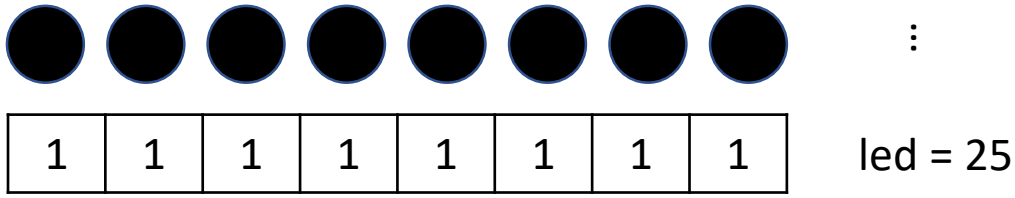
- 8개의 LED가 순차적으로 하나씩 켜지도록 프로그램 수정
- <u>변수로 제어</u>, 무엇이 문제인가

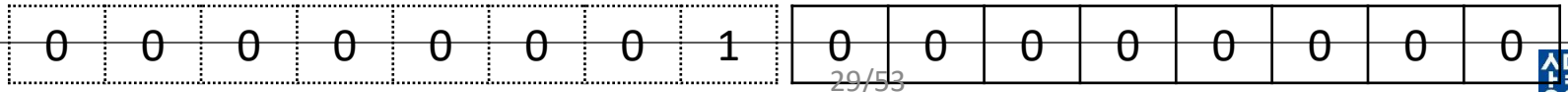
```
int led = 0;

while(1)
{
  GPIO_Write(GPIOC, led);
  Delay(500000);
  led++;
  if(led > 256)
    led = 0;
}
```









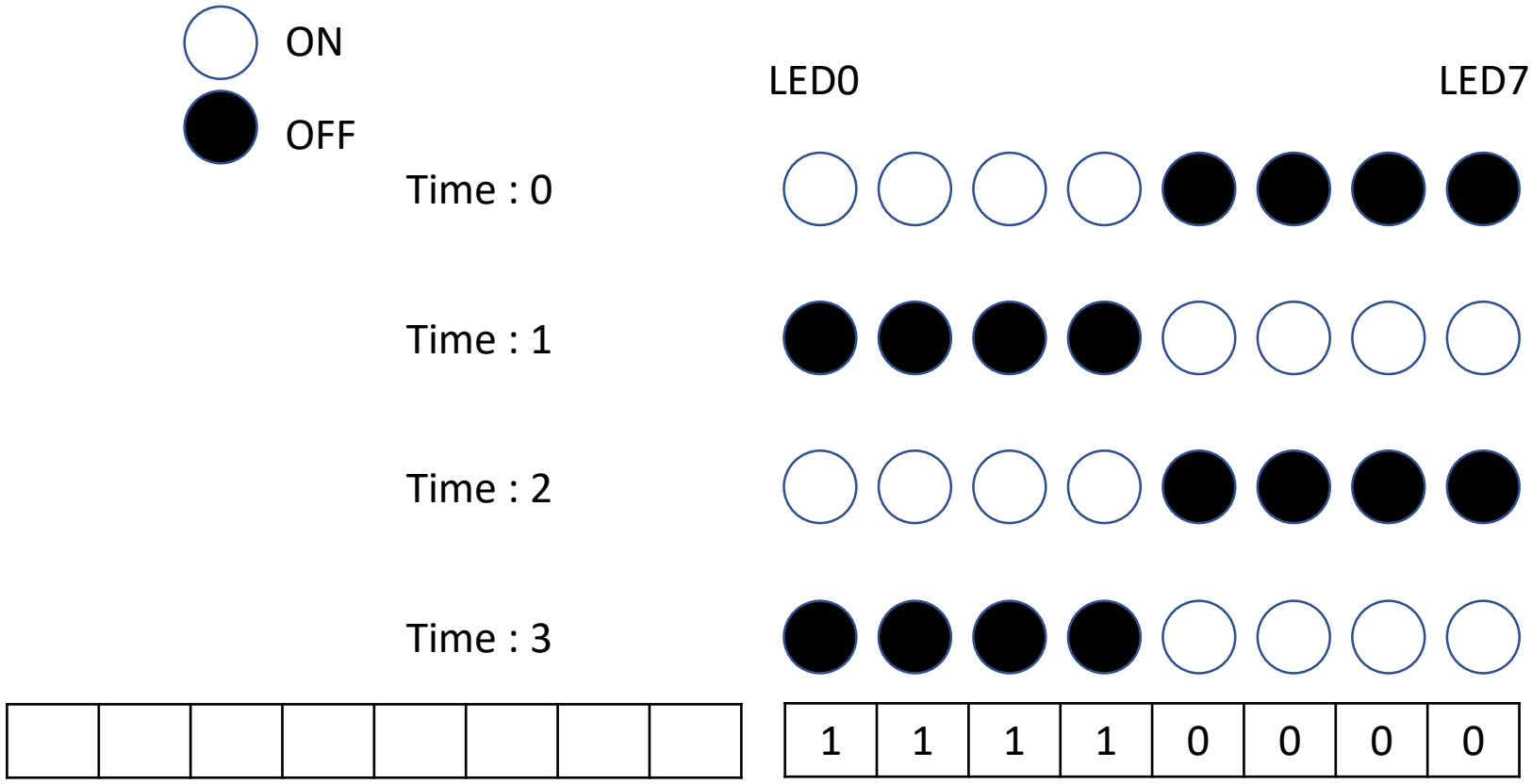


상명대학교 SANGMYUNG UNIVERSITY

GPIO: 실습 1-3

외부 LED

- 8개의 LED가 아래와 같은 순서로 ON 되도록 프로그램을 수정

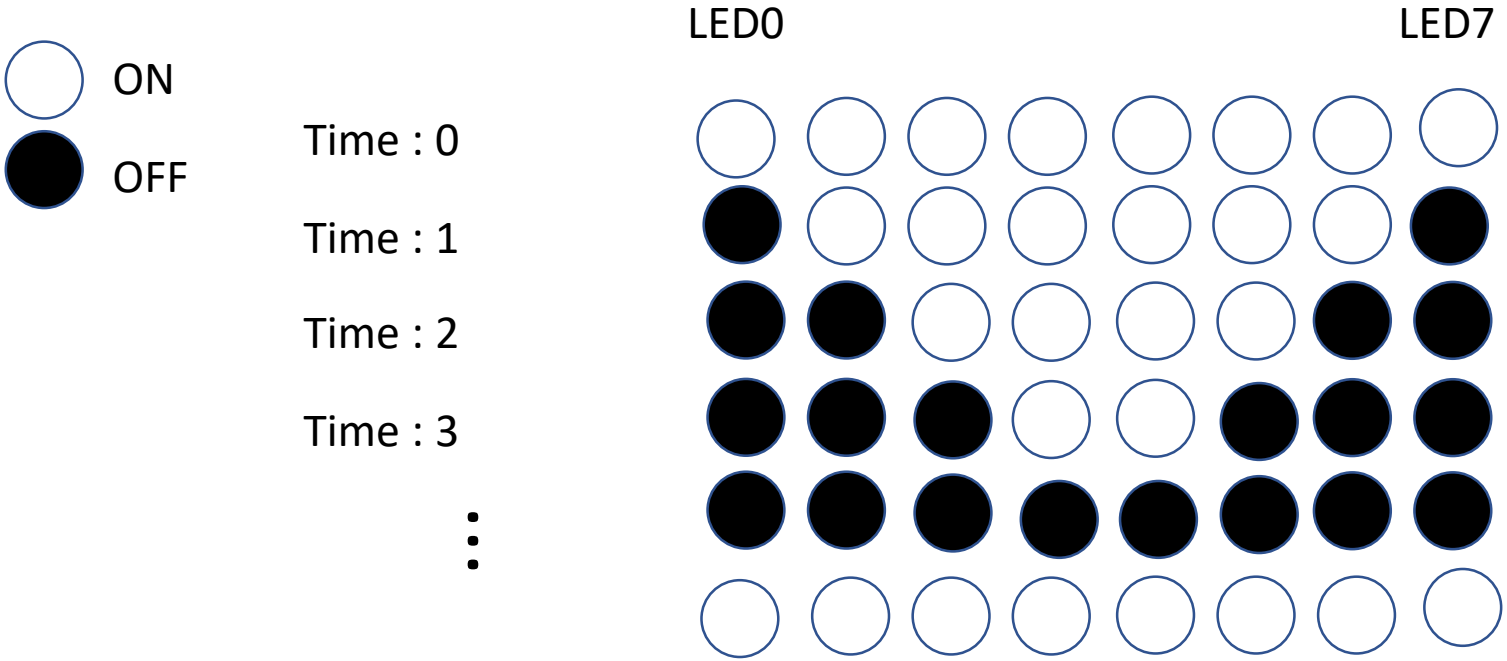


Internal Use Only

GPIO: 실습 1-4

외부 LED

- 8개의 LED가 아래와 같은 순서로 ON 되도록 프로그램을 수정



Internal Use Only

LED + Switch

- Output 제어 실습을 해봤으니,

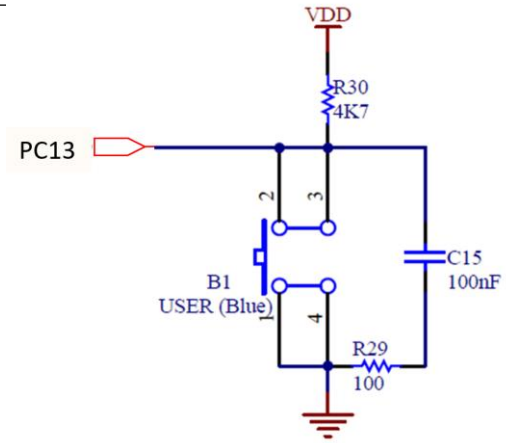
input + output 실습을 해봅시다

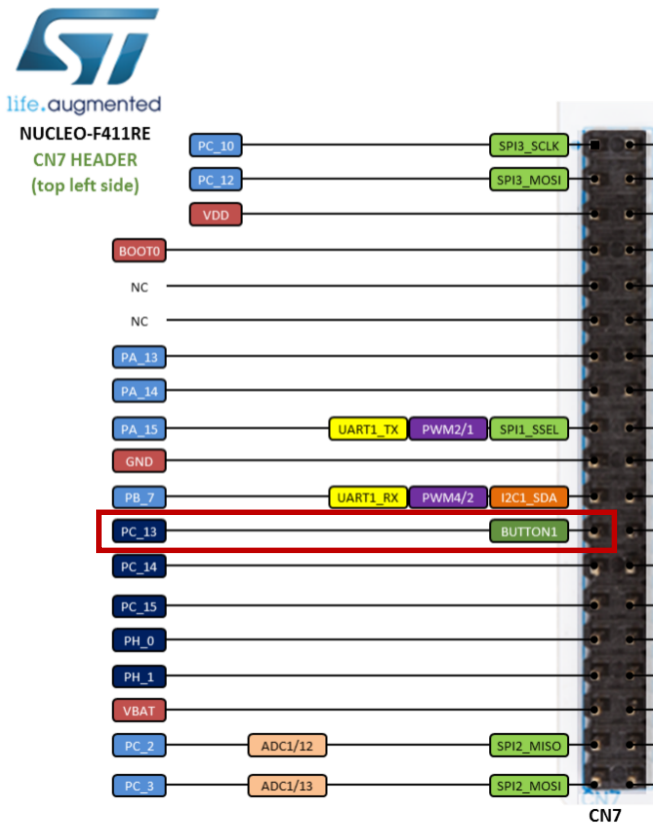
Internal Use Only

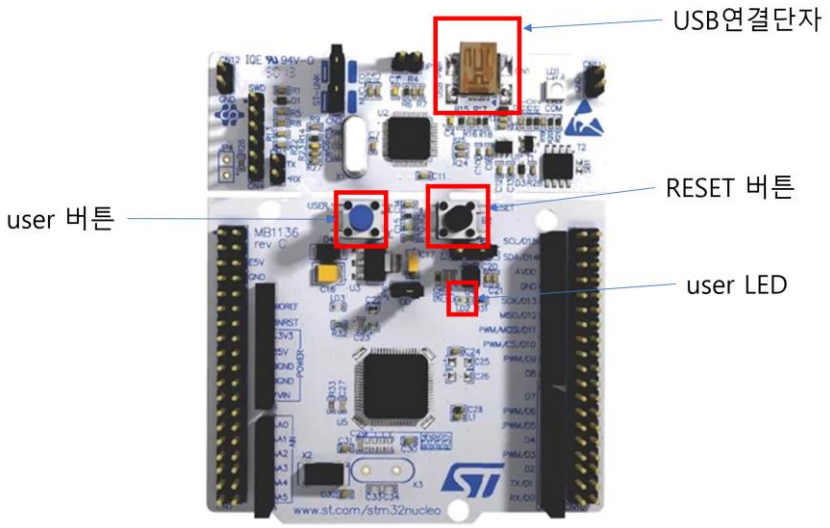
LED + Switch 실습 1-1

User Switch

- User 버튼을 누르면 8개 LED OFF
- User 버튼을 누르지 아니하면 8개 LED ON







LED + Switch 실습 1-1

User Switch

- User 버튼은 어떤 포트 몇 번 핀인가? Input인가 output인가?
- LED 8개는 어떤 포트 몇 번 핀에 연결되어 있는가? Input인가 output인가?
- Input / output으로 사용할 때 고려해야 할 레지스터는?

- Input
  - pull-up / pull-down

- Output
  - push-pull / open-drain
  - pull-up / pull-down

Internal Use Only

LED + Switch 실습 1-1

User Switch

- User 스위치를 이용하여 8개 LED ON/OFF
- Output 파트 (LED 제어)

```
#include "stm32f4xx.h"

void Delay(__IO uint32_t nCount)
{
  for(; nCount != 0; nCount--);
}

void LED_init(void)
{
   RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOC, ENABLE);

   GPIO_InitTypeDef GPIO_InitStructure;

   GPIO_InitStructure.GPIO_Mode = GPIO_Mode_OUT;
   GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0|GPIO_Pin_1|GPIO_Pin_2|GPIO_Pin_3|GPIO_Pin_4|GPIO_Pin_5|GPIO_Pin_6|GPIO_Pin_7;
   GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
   GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL;

   GPIO_Init(GPIOC, &GPIO_InitStructure);
}
```

Internal Use Only

LED + Switch 실습 1-1

User Switch

- User 스위치를 이용하여 8개 LED ON/OFF
- Input 파트 (스위치 제어)

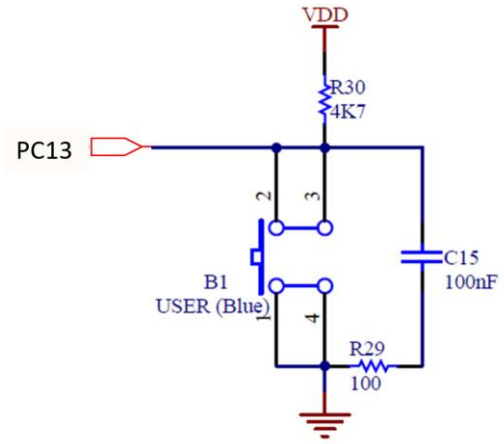
```
void SW_init(void)
{
   RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOC, ENABLE);

   GPIO_InitTypeDef GPIO_InitStructure;

   GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN;
   GPIO_InitStructure.GPIO_Pin = GPIO_Pin_13;
   GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL;

   GPIO_Init(GPIOC, &GPIO_InitStructure);
}
```

Internal Use Only

LED + Switch 실습 1-1

User Switch

- User 스위치를 이용하여 8개 LED ON/OFF
- main




```
int main()
{
   LED_init();
   SW_init();

   while(1)
   {
    if(GPIO_ReadInputDataBit(GPIOC, GPIO_Pin_13) == 1)
      GPIO_Write(GPIOC, 0x00FF);
    else
      GPIO_ Write(GPIOC, 0x0000);
   }

  return 0;
}
```

스위치를 누르지 않으면 1
누르면 0

Internal Use Only

LED + Switch 실습 1-2

User Switch

- User 스위치를 누르면, LED가 한칸씩 이동하는 코드 작성해보기
- 아래 코드 참고 + 무언가 추가해야 함

```
int main()
{
   int led = 0;
   …

   LED_init();
   SW_init();

   while(1)
   {
      …
      GPIO_Write(GPIOC, led);
   }
return 0;
}
```

+a

```
if(led == 0) // 1개도 안켜져 있으면
    {
      led++;
    }
    else if(led == 128) // 맨 끝까지가면
    {
      led = 0;
    }
    else // 1칸 shift
    {
      led = led << 1;
    }
```

1	0	0	0	0	0	0	0

0	0	0	0	0	0	0	1

0	0	0	0	0	0	1	0

Internal Use Only

LED + Switch 실습 1-2

User Switch

- 잘못된 코드

```
while(1)
{

 if(GPIO_ReadInputDataBit(GPIOC, GPIO_Pin_13) == 0)
 {
  if(led == 0)
  {
   led++;
  }
  else if(led == 128)
  {
   led = 0;
  }
  else
  {
   led = led << 1;
  }
 }

 GPIO_Write(GPIOC, led);
}
```

스위치를 누르면 0
이 값을 클럭 속도로 받아옴
우리는 한번 살짝 눌렀다고 생각하지만
(스위치를 누르고 있다면 더욱)
MCU관점에서는 많은 값을 읽어옴

엄청 빠르게 led변수가 변한 것이고,
우리 눈에 보이지 않을 정도로 빠르게
LED가 변하고 있음

Internal Use Only

LED + Switch 실습 1-2

User Switch

- **추천 코드**

```
int main()
{
  LED_init();
  SW_init();

  int led = 0;
  int sw_new = 1;
  int sw_old = 1;

  sw_new = GPIO_ReadInputDataBit(GPIOC, GPIO_Pin_13);
```

스위치를 누르면 sw_new는 0이 됨
그러다가 떼면 sw_new는 1이 됨

```
// 눌렸다 떨어질때 action
  if( (sw_old == 0) & (sw_new == 1) )
  {
    if(led == 0)
    {
      led++;
    }
    else if(led == 128)
    {
      led = 0;
    }
    else
    {
      led = led << 1;
    }
  }
  sw_old = sw_new;
  GPIO_Write(GPIOC, led);
 }
 return 0;
}
```

Internal Use Only

LED + Switch 실습 1-2

참고

PC15	PC14	PC13	PC12	PC11	PC10	PC9	PC8	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

GPIO_Write(GPIOC, GPIO_Pin_0);

2진수: 0000 0000 0000 0001
16진수: 0x0001

GPIO_Pin_0와 0x0001이 왜 같지?

Internal Use Only

상명대학교 SANGMYUNG UNIVERSITY

LED + Switch 실습 1-2

참고

- RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOC, ENABLE);
  - GPIOC를 사용하기 위해, 이 주변장치에 클럭을 공급(ENABLE)
  - 만일 사용을 더 이상 하지 않는 경우, DISABLE로 설정 시 전력소비 감소
- GPIO_InitTypeDef GPIO_InitStructure;
  - GPIO를 설정하기 위한 GPIO_InitTypeDef 구조체 선언

```
typedef struct
{
  uint32_t GPIO_Pin;           /*!< Specifies the GPIO pins to be configured.
                                This parameter can be any value of @ref GPIO_pins_define */

  GPIOMode_TypeDef GPIO_Mode;     /*!< Specifies the operating mode for the selected pins.
                                   This parameter can be a value of @ref GPIOMode_TypeDef */

  GPIOSpeed_TypeDef GPIO_Speed;   /*!< Specifies the speed for the selected pins.
                                   This parameter can be a value of @ref GPIOSpeed_TypeDef */

  GPIOOType_TypeDef GPIO_OType;   /*!< Specifies the operating output type for the selected pins.
                                   This parameter can be a value of @ref GPIOOType_TypeDef */

  GPIOPuPd_TypeDef GPIO_PuPd;     /*!< Specifies the operating Pull-up/Pull down for the selected pins.
                                   This parameter can be a value of @ref GPIOPuPd_TypeDef */
}GPIO_InitTypeDef;
```

Internal Use Only

LED + Switch 실습 1-2

참고

- GPIO_InitStructure.GPIO_Pin = GPIO_Pin_5;
  - GPIO의 핀을 선택하는 부분 GPIO_Pin_0부터 GPIO_Pin_15까지 지정 가능
  - 16개의 핀을 모두 선택하는 경우 GPIO_Pin_All
- GPIO_InitStructure.GPIO_Mode = GPIO_Mode_OUT;
  - GPIO모드로 4가지 형태로 사용 가능

```
typedef enum
{
  GPIO_Mode_IN   = 0x00, /*!< GPIO Input Mode */
  GPIO_Mode_OUT  = 0x01, /*!< GPIO Output Mode */
  GPIO_Mode_AF   = 0x02, /*!< GPIO Alternate function Mode */
  GPIO_Mode_AN   = 0x03  /*!< GPIO Analog Mode */
}GPIOMode_TypeDef;
```

- GPIO port mode register (GPIOx_MODER) (x = A..I/J/K)

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

Bits 2y:2y+1 **MODERy[1:0]:** Port x configuration bits (y = 0..15)
These bits are written by software to configure the I/O direction mode.
00: Input (reset state)
01: General purpose output mode
10: Alternate function mode
11: Analog mode

Internal Use Only

LED + Switch 실습 1-2

참고

- **GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;**
  - **GPIO출력 형태를 지정**

```
typedef enum
{
  GPIO_OType_PP = 0x00,
  GPIO_OType_OD = 0x01
}GPIOOType_TypeDef;
```

- **GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL;**
  - **GPIO의 Pull-up, pull-down에 대한 설정**

```
typedef enum
{
  GPIO_PuPd_NOPULL = 0x00,
  GPIO_PuPd_UP     = 0x01,
  GPIO_PuPd_DOWN   = 0x02
}GPIOPuPd_TypeDef;
```

Internal Use Only

LED + Switch 실습 1-2

참고

- GPIO_Init(GPIOC, &GPIO_InitStructure);
  - GPIOC를 앞에서 설정한 GPIO_InitStructure의 내용으로 초기화하는 함수

```
void GPIO_Init(GPIO_TypeDef* GPIOx, GPIO_InitTypeDef* GPIO_InitStruct)
{
 uint32_t pinpos = 0x00, pos = 0x00 , currentpin = 0x00;

 /* Check the parameters */
 assert_param(IS_GPIO_ALL_PERIPH(GPIOx));
 assert_param(IS_GPIO_PIN(GPIO_InitStruct->GPIO_Pin));
 assert_param(IS_GPIO_MODE(GPIO_InitStruct->GPIO_Mode));
 assert_param(IS_GPIO_PUPD(GPIO_InitStruct->GPIO_PuPd));

 /* ------------------------ Configure the port pins ---------------- */
 /*-- GPIO Mode Configuration --*/
 for (pinpos = 0x00; pinpos < 0x10; pinpos++)
 {
  pos = ((uint32_t)0x01) << pinpos;
  /* Get the port pins position */
  currentpin = (GPIO_InitStruct->GPIO_Pin) & pos;

  if (currentpin == pos)
  {
   GPIOx->MODER  &= ~(GPIO_MODER_MODER0 << (pinpos * 2));
   GPIOx->MODER |= (((uint32_t)GPIO_InitStruct->GPIO_Mode) << (pinpos * 2));

   if ((GPIO_InitStruct->GPIO_Mode == GPIO_Mode_OUT) || (GPIO_InitStruct->GPIO_Mode == GPIO_Mode_AF))
   {
    /* Check Speed mode parameters */
    assert_param(IS_GPIO_SPEED(GPIO_InitStruct->GPIO_Speed));

    /* Speed mode configuration */
    GPIOx->OSPEEDR &= ~(GPIO_OSPEEDER_OSPEEDR0 << (pinpos * 2));
    GPIOx->OSPEEDR |= ((uint32_t)(GPIO_InitStruct->GPIO_Speed) << (pinpos * 2));

    /* Check Output mode parameters */
    assert_param(IS_GPIO_OTYPE(GPIO_InitStruct->GPIO_OType));

    /* Output mode configuration*/
    GPIOx->OTYPER  &= ~((GPIO_OTYPER_OT_0) << ((uint16_t)pinpos)) ;
    GPIOx->OTYPER |= (uint16_t)(((uint16_t)GPIO_InitStruct->GPIO_OType) << ((uint16_t)pinpos));
   }

   /* Pull-up Pull down resistor configuration*/
   GPIOx->PUPDR &= ~(GPIO_PUPDR_PUPDR0 << ((uint16_t)pinpos * 2));
   GPIOx->PUPDR |= (((uint32_t)GPIO_InitStruct->GPIO_PuPd) << (pinpos * 2));
  }
 }
}
```

Internal Use Only

LED + Switch 실습 1-2

참고

- GPIO_ReadInputDataBit(GPIOC, GPIO_Pin_13)
  - GPIO의 Pin에서 값을 읽어오는 함수

```
uint8_t GPIO_ReadInputDataBit(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin)
{
  uint8_t bitstatus = 0x00;

  /* Check the parameters */
  assert_param(IS_GPIO_ALL_PERIPH(GPIOx));
  assert_param(IS_GET_GPIO_PIN(GPIO_Pin));

  if ((GPIOx->IDR & GPIO_Pin) != (uint32_t)Bit_RESET)
  {
    bitstatus = (uint8_t)Bit_SET;
  }
  else
  {
    bitstatus = (uint8_t)Bit_RESET;
  }
  return bitstatus;
}
```

Internal Use Only

LED + Switch 실습 1-2

참고

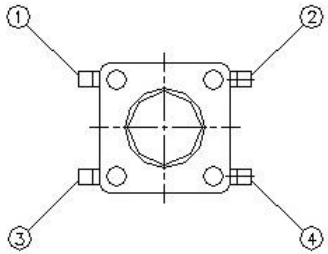
- GPIO_ReadInputDataBit(GPIOC, GPIO_Pin_13)
  - GPIO_SetBits(GPIOC, GPIO_Pin_5);
  - GPIO_ResetBits(GPIOC, GPIO_Pin_5);

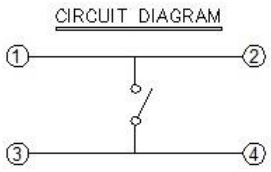
```
void GPIO_SetBits(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin)
{
  /* Check the parameters */
  assert_param(IS_GPIO_ALL_PERIPH(GPIOx));
  assert_param(IS_GPIO_PIN(GPIO_Pin));

  GPIOx->BSRRL = GPIO_Pin;
}


void GPIO_ResetBits(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin)
{
  /* Check the parameters */
  assert_param(IS_GPIO_ALL_PERIPH(GPIOx));
  assert_param(IS_GPIO_PIN(GPIO_Pin));

  GPIOx->BSRRH = GPIO_Pin;
}
```

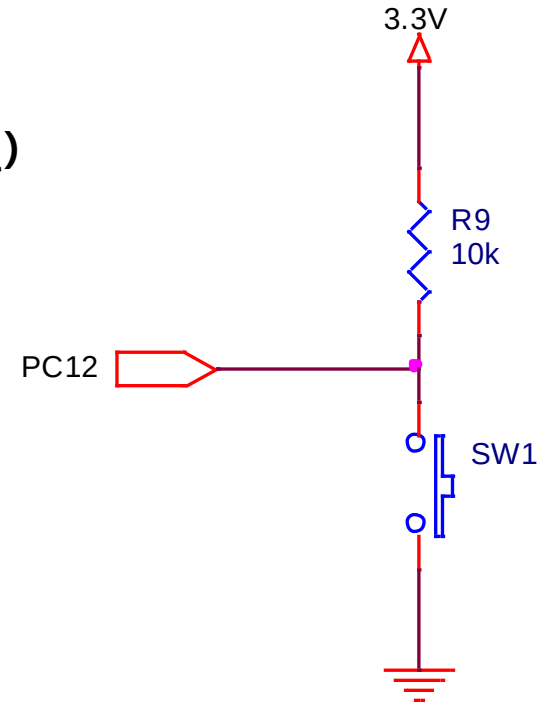
Internal Use Only

LED + Switch 실습 1-3

- 외부 스위치를 이용하여 포트 C0~C7에 연결된 LED의 값을 변경하는 프로그램 작성
- 외부 스위치가 눌릴 때마다 LED가 한칸씩 이동
- HW 구성
  - 저항(220Ω), LED를 아래와 같이 연결
  - 포트 C PC0~PC7을 각각 LED와 연결
  - 택트 스위치를 포트 C PC12에 연결 (저항 10k 연결)

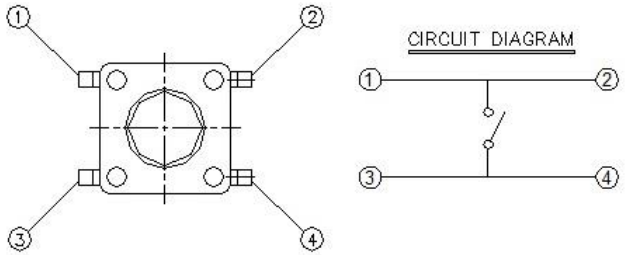


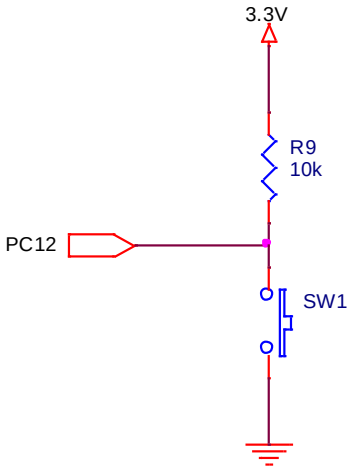


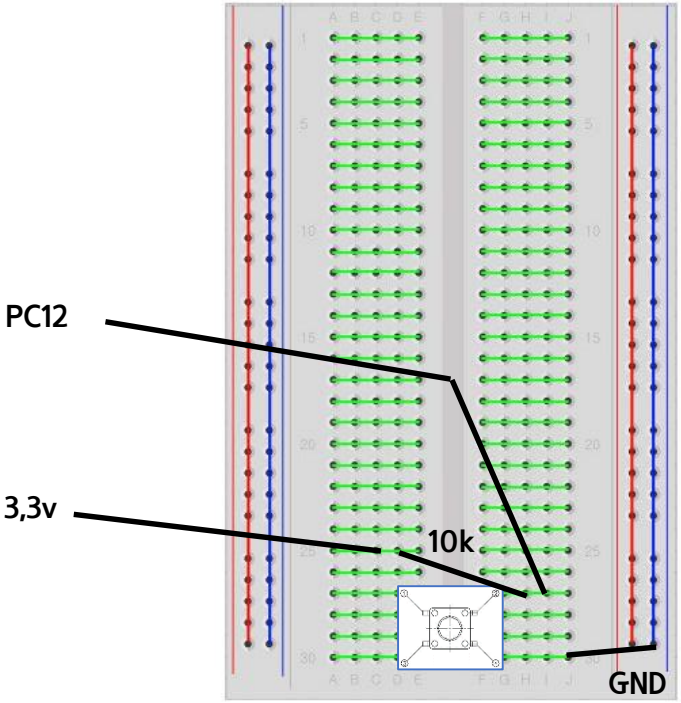


Internal Use Only

LED + Switch 실습 1-3







Internal Use Only

상명대학교 SANGMYUNG UNIVERSITY

LED + Switch 실습 1-3

참고

- PC12 외부에 구현된 회로는 Pull-up의 모습이니, PC12설정을 GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_UP; 으로 해야할까?

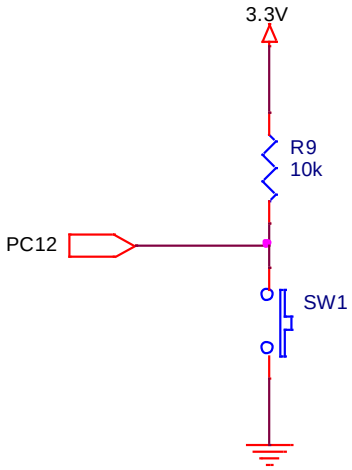
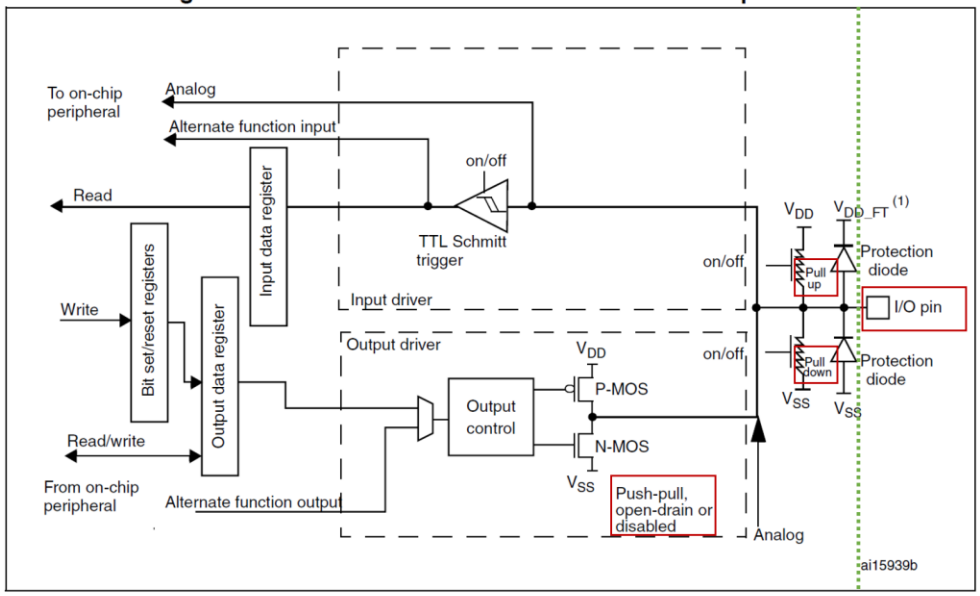
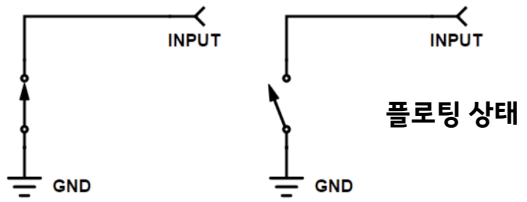


Figure 25. Basic structure of a five-volt tolerant I/O port bit



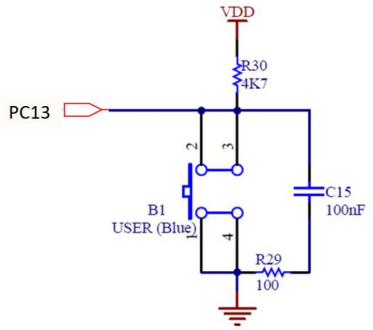
1. V_{DD_FT} is a potential specific to five-volt tolerant I/Os and different from V_{DD}.

외부에 회로를 구현하지 않았을 때 고려하면 됨



Internal Use Only
상명대학교 SANGMYUNG UNIVERSITY

과제

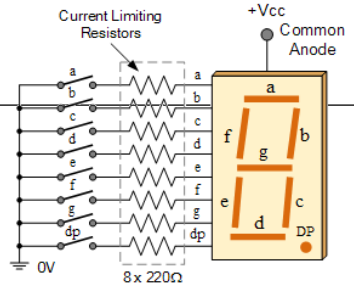


- 과제 1
  - 발생하는 문제를 코드적으로 해결해보자
    - 이 문제는 사실 하드웨어적으로 해결되야 함
    - 하지만, 어느정도 코드적으로 보완할 수 있음
    - 정답은 없음
    - (hint) 이 문제가 어떤 상황이기에 발생하는 것인지 생각해보기
  - 외부 스위치를 "누를 때" 마다 LED가 한 칸씩 이동하는 프로그램 작성 (뗄 때 아니고, 누를 때임!)
  - main.c 파일만 e-campus에 업로드
- 과제 2
  - FND 연결해오기
  - 10월 16일에 진행할 실습 때 사용할 IAR 프로젝트 미리 만들어오기

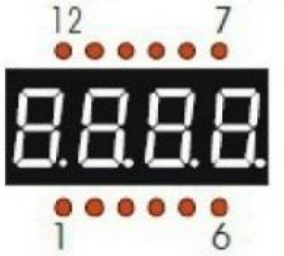
Internal Use Only

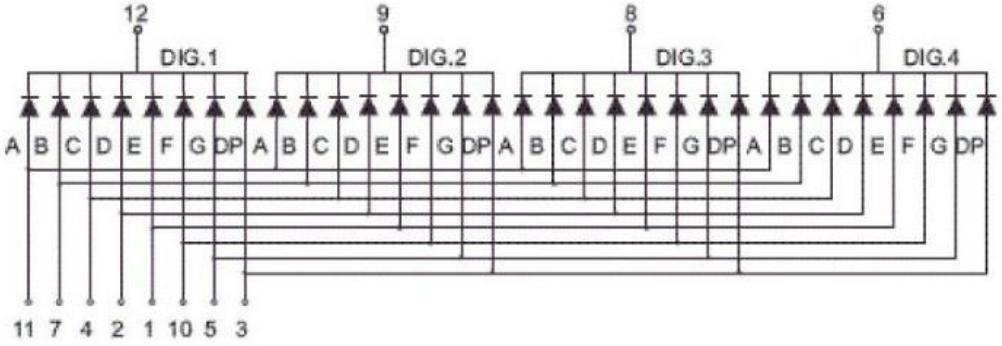
과제

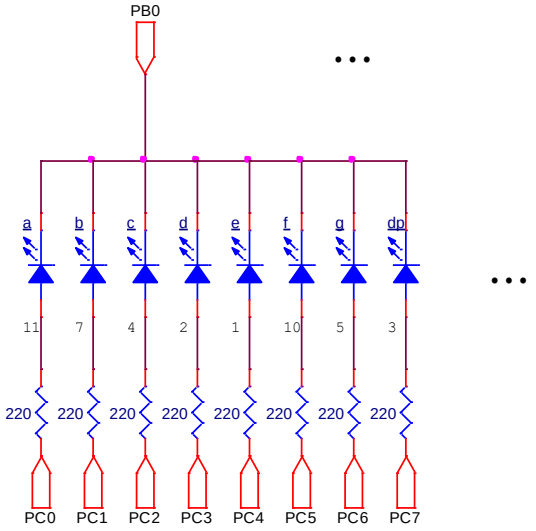
과제 2



- 7-세그먼트(FND)
  - FND의 A,B,C,D,E,F,G,DP를 저항(220Ω)을 연결한 후 PC0~PC7과 연결
  - FND의 common인 12,9,8,6핀을 PB0~PB3과 연결







Internal Use Only

Thank you.