

임베디드 MCU 프로그래밍 실습

AURIX TC275 보드 GPIO 사용

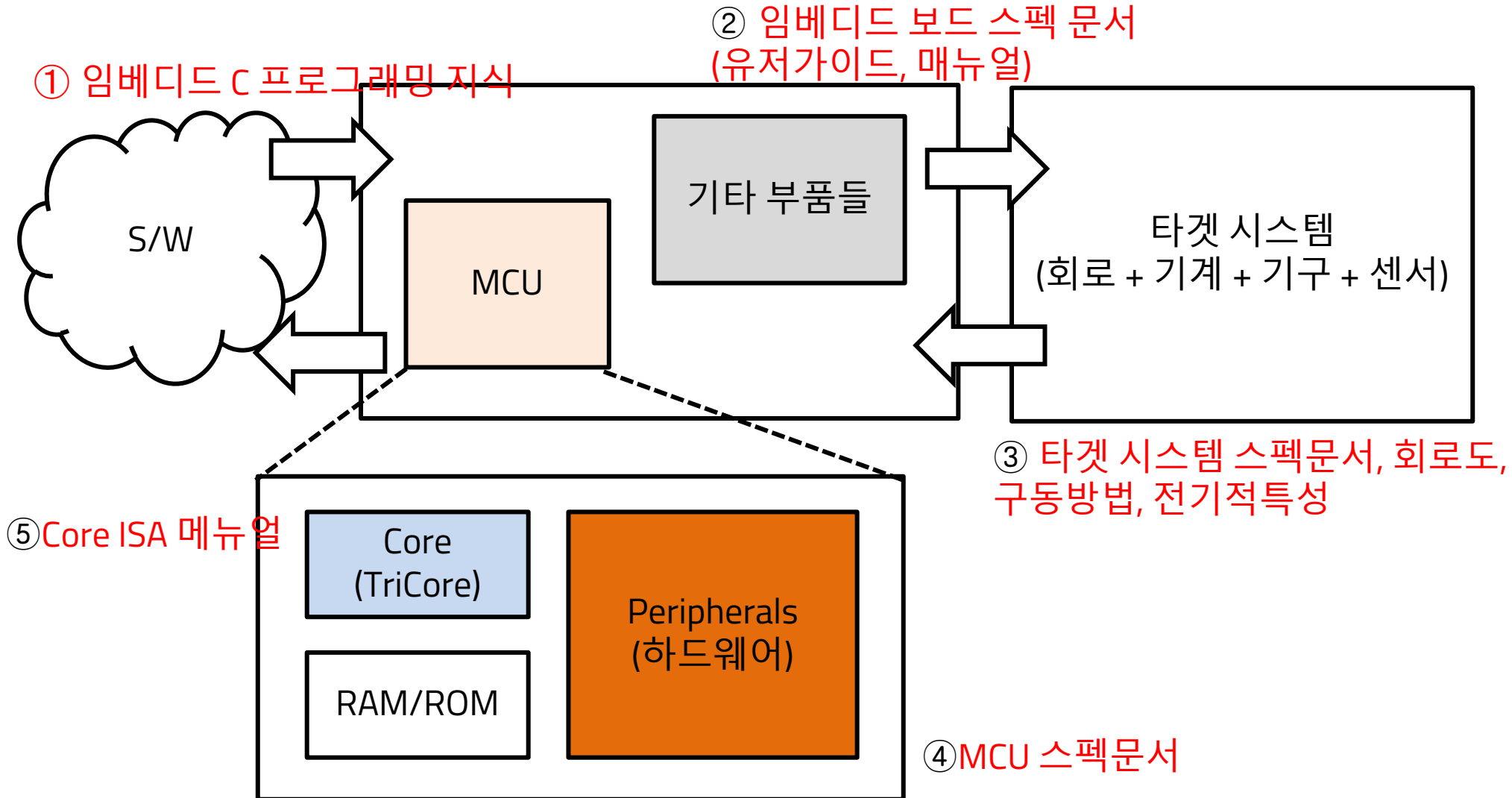
현대자동차 입문교육
박대진 교수

실습 목표

: 입력, 출력 제어

1. 보드의 **GPIO** 출력을 제어해서 **LED**를 **toggle** 해보도록 한다.
2. 확장 보드의 **Push 버튼 (SW2)** 상태를 **GPIO**로 입력 받아 버튼이 눌렸을 때 **LED**를 켜보도록 한다.

(1/3) MCU 프로그래밍 개요 및 읽어야 할 문서들



(2/3) 스펙을 읽고나서 코딩 - 주소 및 타입

```
372
373 /** \brief 10, Port Input/Output Control Register 0 */
374 #define P10_IOCR0 /*lint --e(923)*/ (*(volatile Ifx_P_IOCR0*) 0xF003B010u)
375
376 /** \brief 14, Port Input/Output Control Register 4 */
```

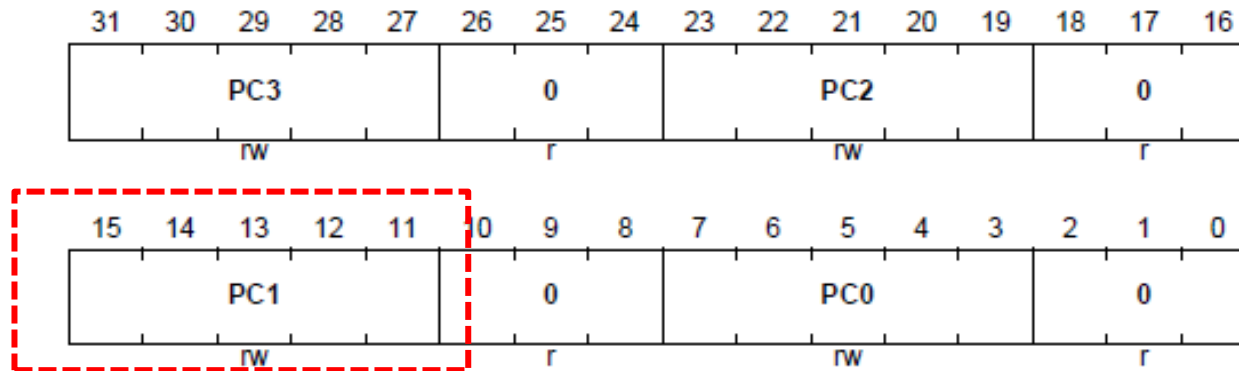
```
563
564 /** \brief Port Input/Output Control Register 0 */
565 typedef union
566 {
567     unsigned int U;
568     signed int I;
569     Ifx_P_IOCR0_Bits B;
570 } Ifx_P_IOCR0;
571
```

/**< \brief Unsigned access */
/**< \brief Signed access */
/**< \brief Bitfield access */

```
156
157 /** \brief Port Input/Output Control Register 0 Bits
```

```
158 typedef struct _Ifx_P_IOCR0_Bits
159 {
160     unsigned int reserved_0:3;
161     unsigned int PC0:5;
162     unsigned int reserved_8:3;
163     unsigned int PC1:5;
164     unsigned int reserved_16:3;
165     unsigned int PC2:5;
166     unsigned int reserved_24:3;
167     unsigned int PC3:5;
168 } Ifx_P_IOCR0_Bits;
169
```

(3/3) 스펙을 읽고나서 코딩 - 해당 비트 셋팅



여기에 10000b 를 쓰고 싶으면

또는

```
#define PC1_BIT_LSB_IDX 11
P10_IOCR0.U &= ~(0x1F << PC1_BIT_LSB_IDX )
```

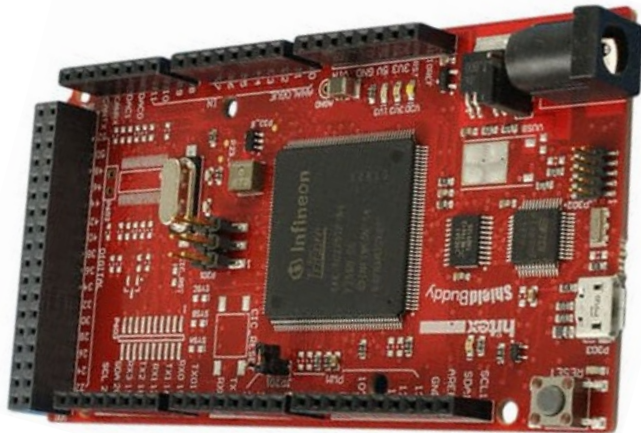
```
5
6 P10_IOCR0.U |= 0x10 << PC1_BIT_LSB_IDX
```

```
9
10 P10_IOCR0.B.PC1 = 0x10;
```

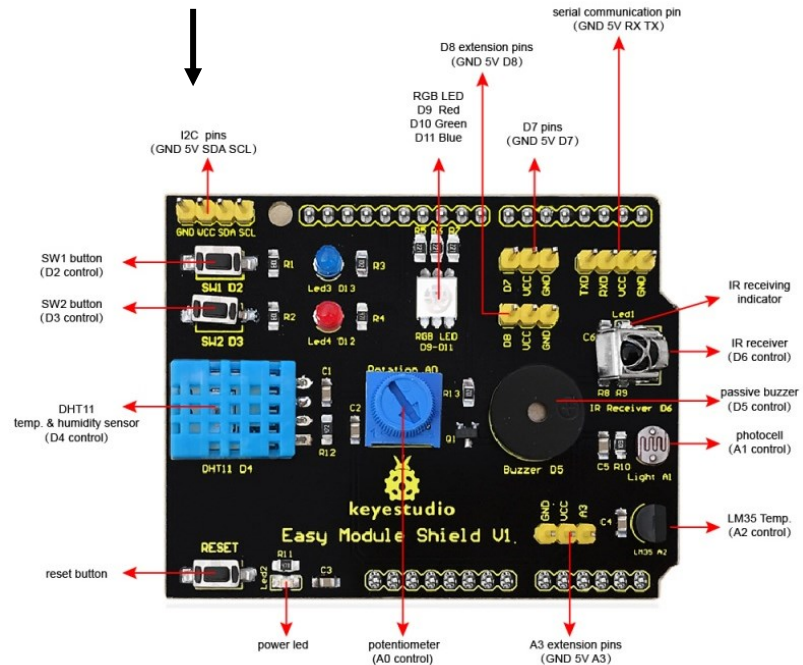
확장 보드 연결

: TC275 + ShieldBuddy + YwRobot Shield

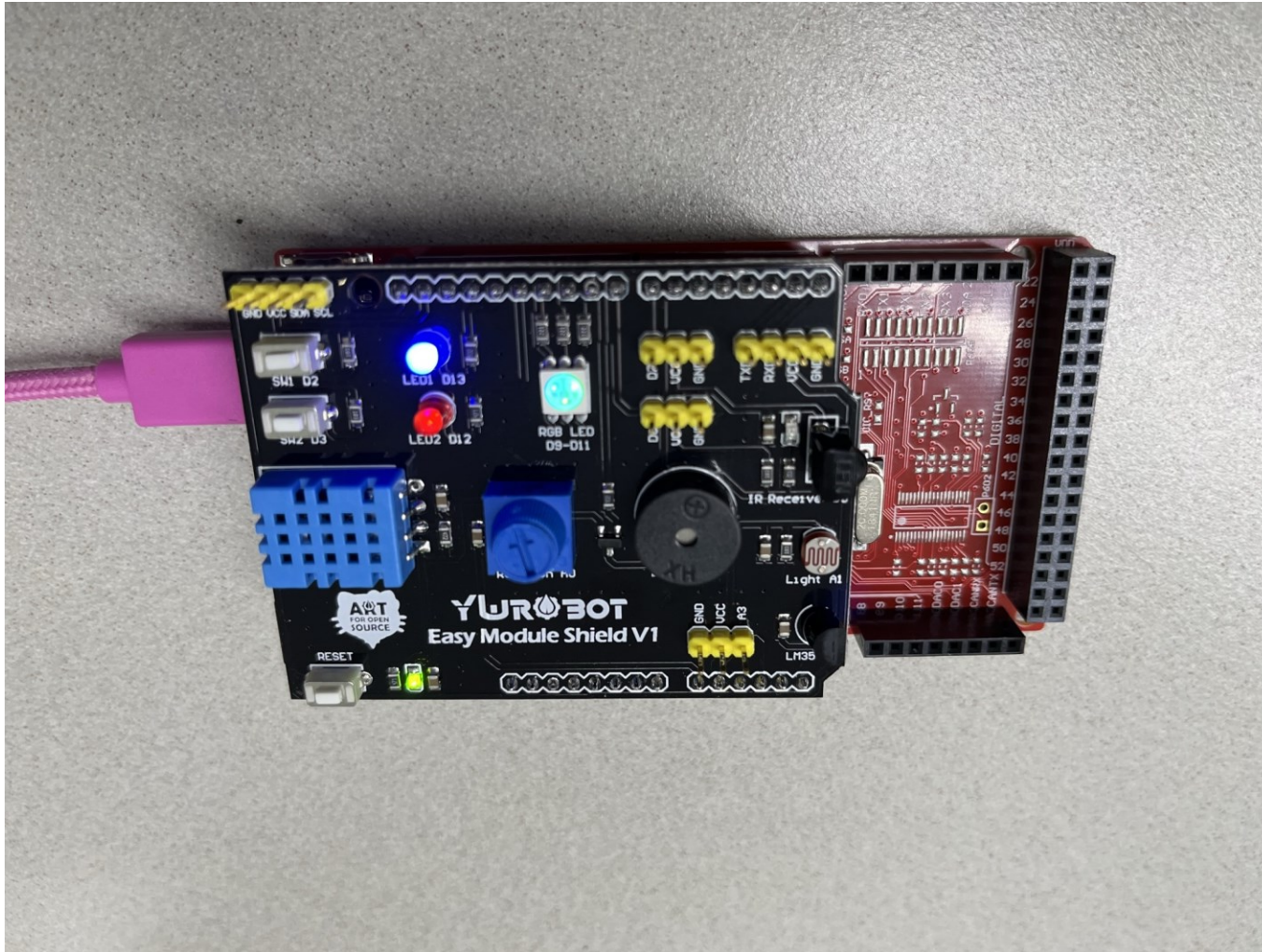
- 본 실습에서 사용하는 보드
 - Infineon 사의 **AURIX TriCore TC275** MCU 를 기반으로
 - Hitex 사에서 제작한 **ShieldBuddy TC275** 개발 보드
- 여러 가지 센서를 브레드보드 없이 연결하기 위해
 - YwRobot Easy Module Shield V1** 확장 보드 사용



+



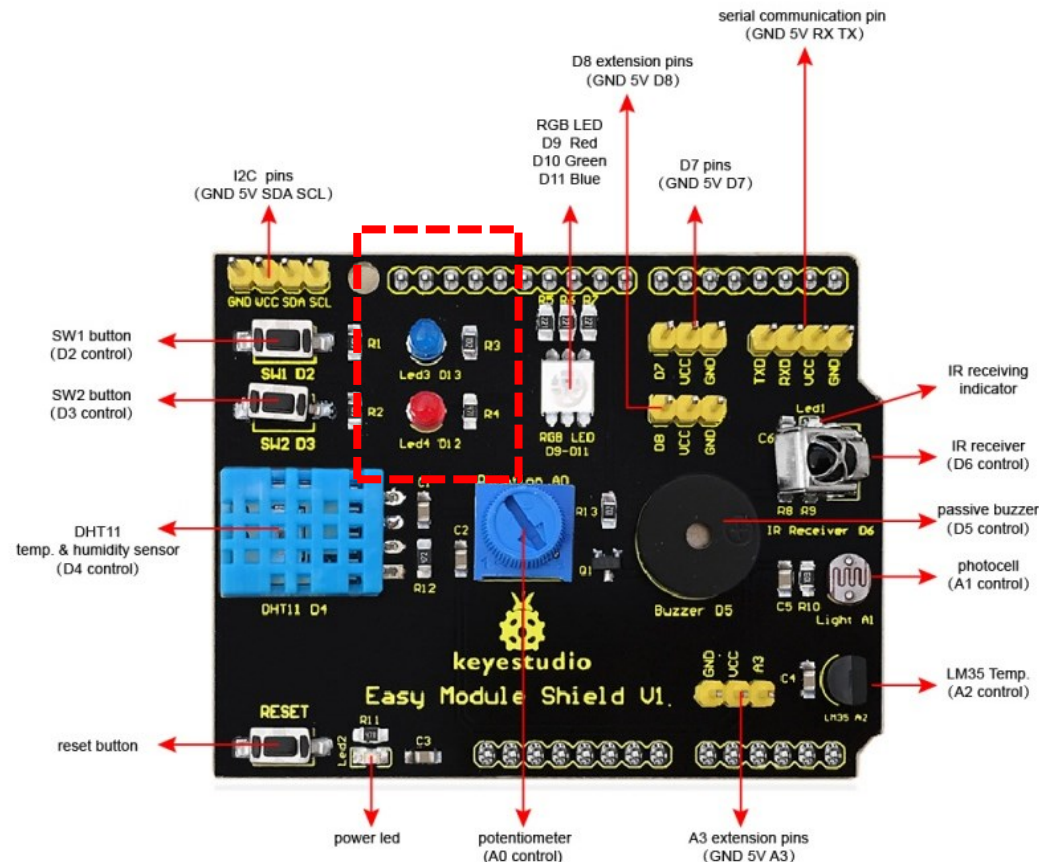
확장 보드를 TC275 보드에 연결한 모습



LED 회로 구조 파악

- 보드의 LED를 사용하기 위해서는 TC275 보드의 어떤 핀이 Easy Module Shield 확장 보드의 LED와 연결되어 있는지 알아야 함

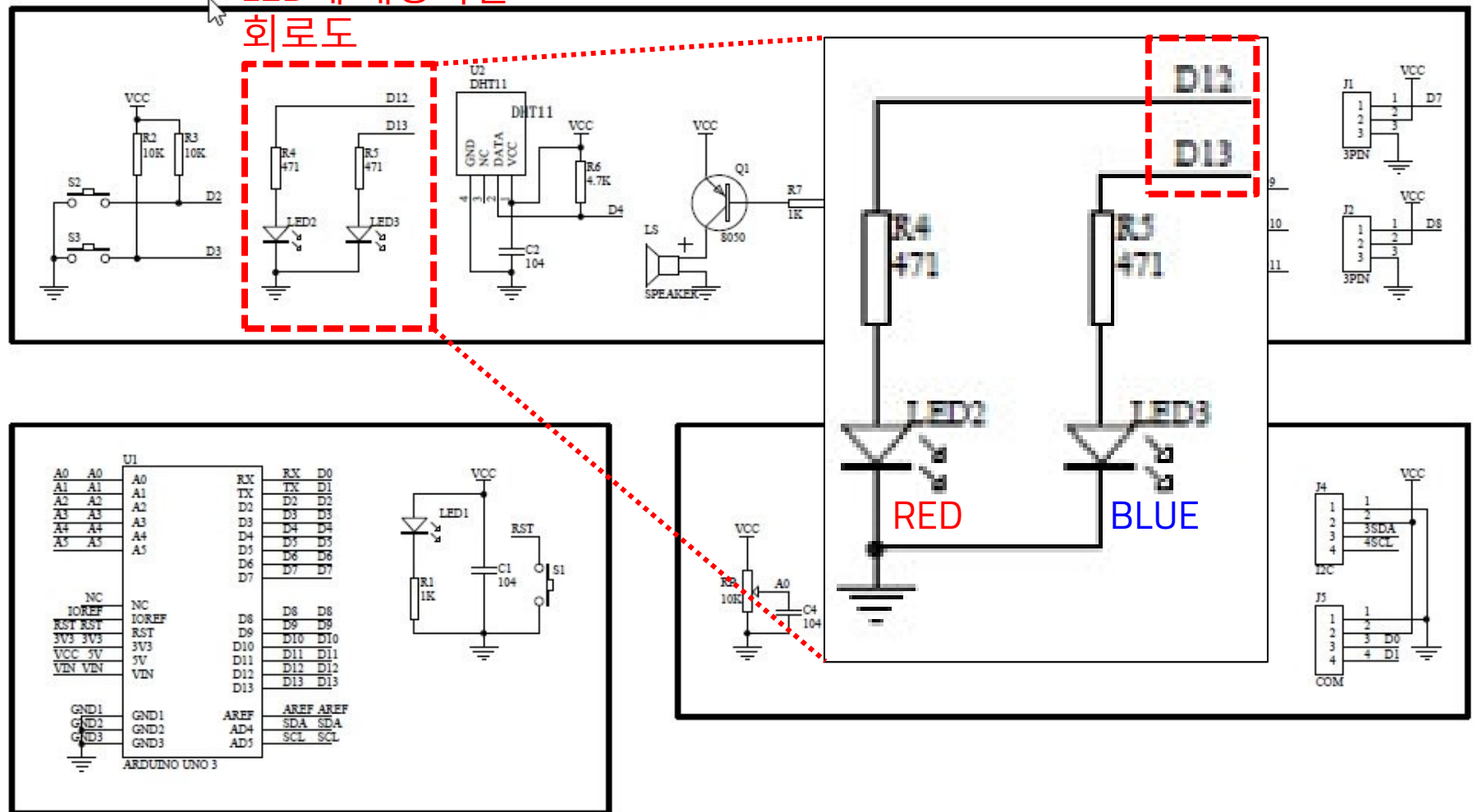
→ 데이터 시트 분석 필요



회로도 @ 확장 보드 데이터 시트

LED는
D12, D13 핀에 연결

LED에 해당하는
회로도



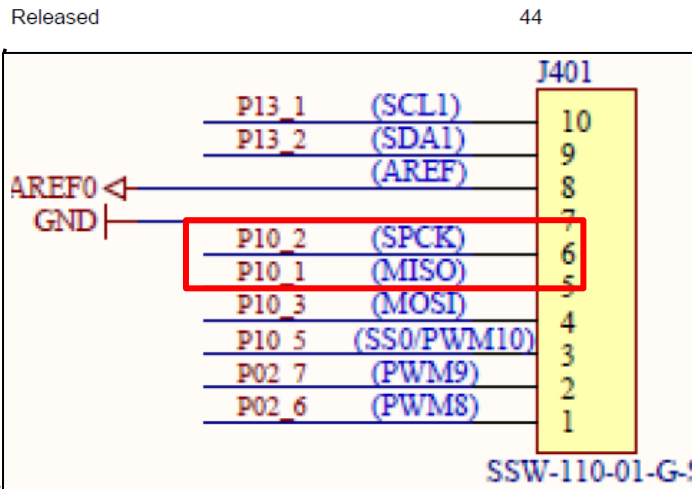
Pin Map @ TC275 보드

- 앞서 LED는 확장 보드의 핀 D12, D13에 연결 되어있는 것을 확인
→ 그렇다면 확장 보드의 핀 D12, D13는 TC275 보드의 어느 핀에 연결?

Infineon-ShieldBuddy_TC275 -UM-v02_08-EN.pdf p.44

Digital pin 8 (PWM)	PWMH.1	P2.6
Digital pin 9 (PWM)	PWMH.2	P2.7
Digital pin 10 (PWM/SS)	PWMH.3	P10.5
Digital pin 11 (PWM/MOSI)	PWMH.4	P10.3
Digital pin 12 (PWM/MISO)	PWMH.5	P10.1
Digital pin 13 (PWM/SPCK)	PWMH.6	P10.2

D12
D13

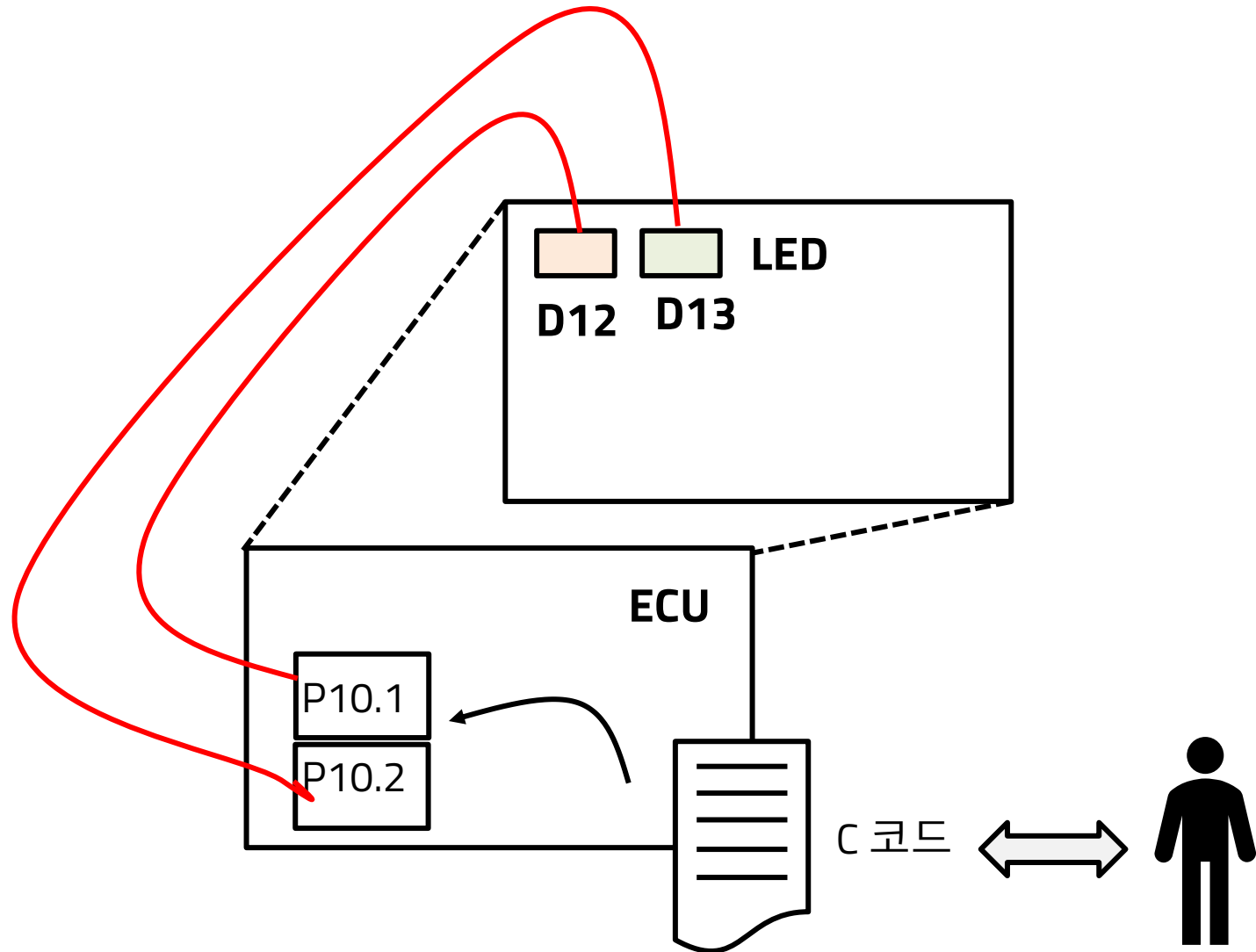


TC275 MCU I/O 중,
Port 10 pin 1 (P10.1)
Port 10 pin 2 (P10.2)
와 연결됨

Infineon-ShieldBuddy_TC275 -UM-v02_08-EN.pdf p.42

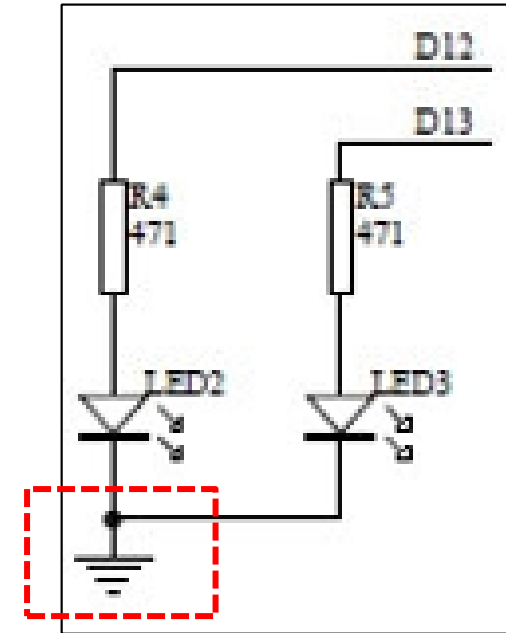
Infineon-ShieldBuddy_TC275 -UM-v02_08-EN.pdf p.46

ECU보드와 Sensor/Actuator보드의 연결구조



LED를 제어하려면...

- LED 회로를 보면 D12, D13 핀 반대쪽에 GND가 있으므로
 - HIGH 값을 인가해야 LED가 켜질 수 있음
 - 반대로 LOW 값을 인가하면 LED는 꺼짐
- 즉, LED를 켜려면 MCU의 P10.1, P10.2 핀에 HIGH 값을 인가해야 함
- 어떻게 MCU의 핀에 HIGH 또는 LOW 값을 인가?
 - **GPIO** 하드웨어 모듈을 설정해야 함



GPIO (General Purpose Input Output)

- MCU 칩 외부와 통신하기 위해 범용(general purpose)으로 사용되는 디지털 신호 입/출력 모듈

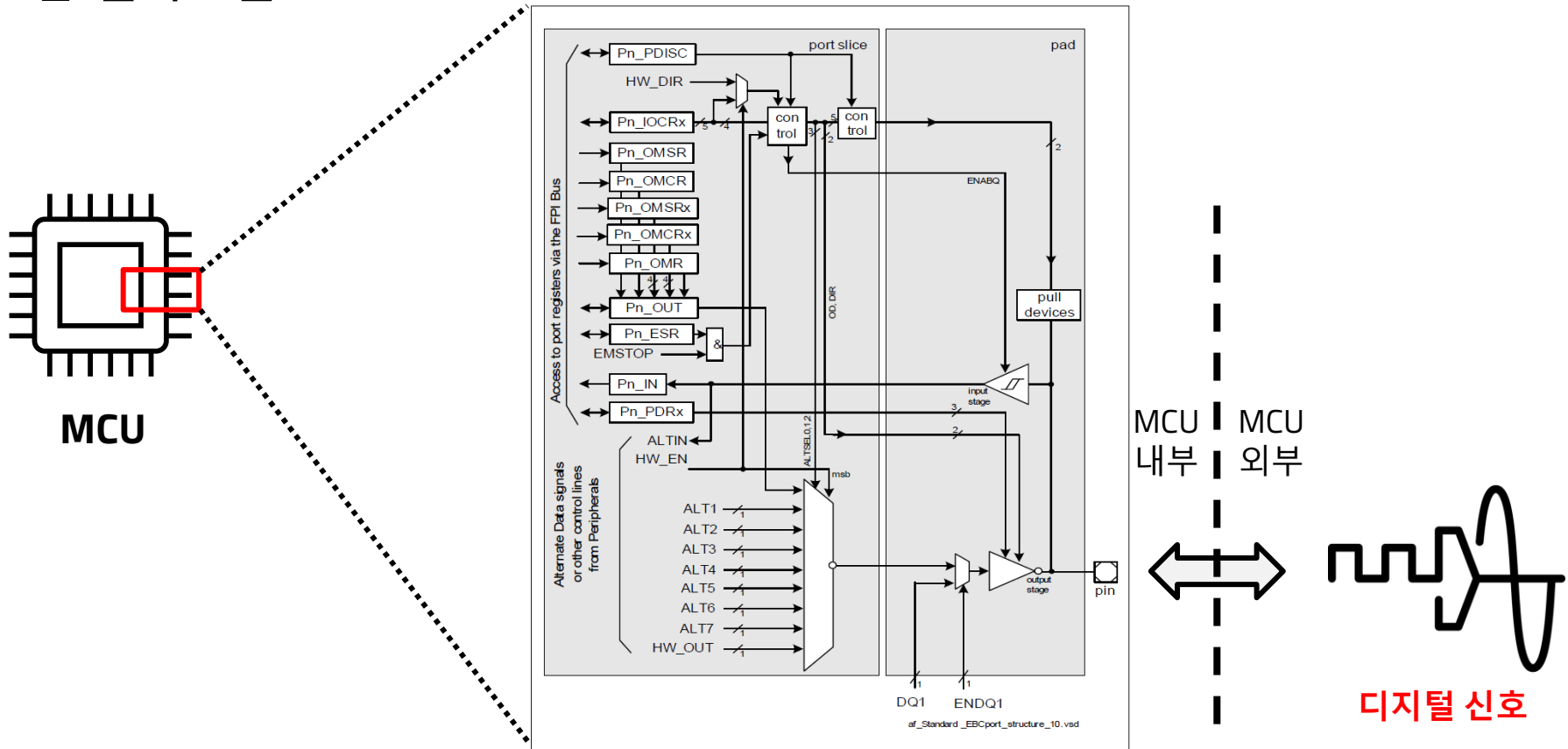


Figure 13-1 General Structure of a Port Pin

[Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.1067](#)

GPIO (General Purpose Input Output)

- GPIO 하드웨어 모듈의 다양한 기능을 원하는 대로 사용하기 위해

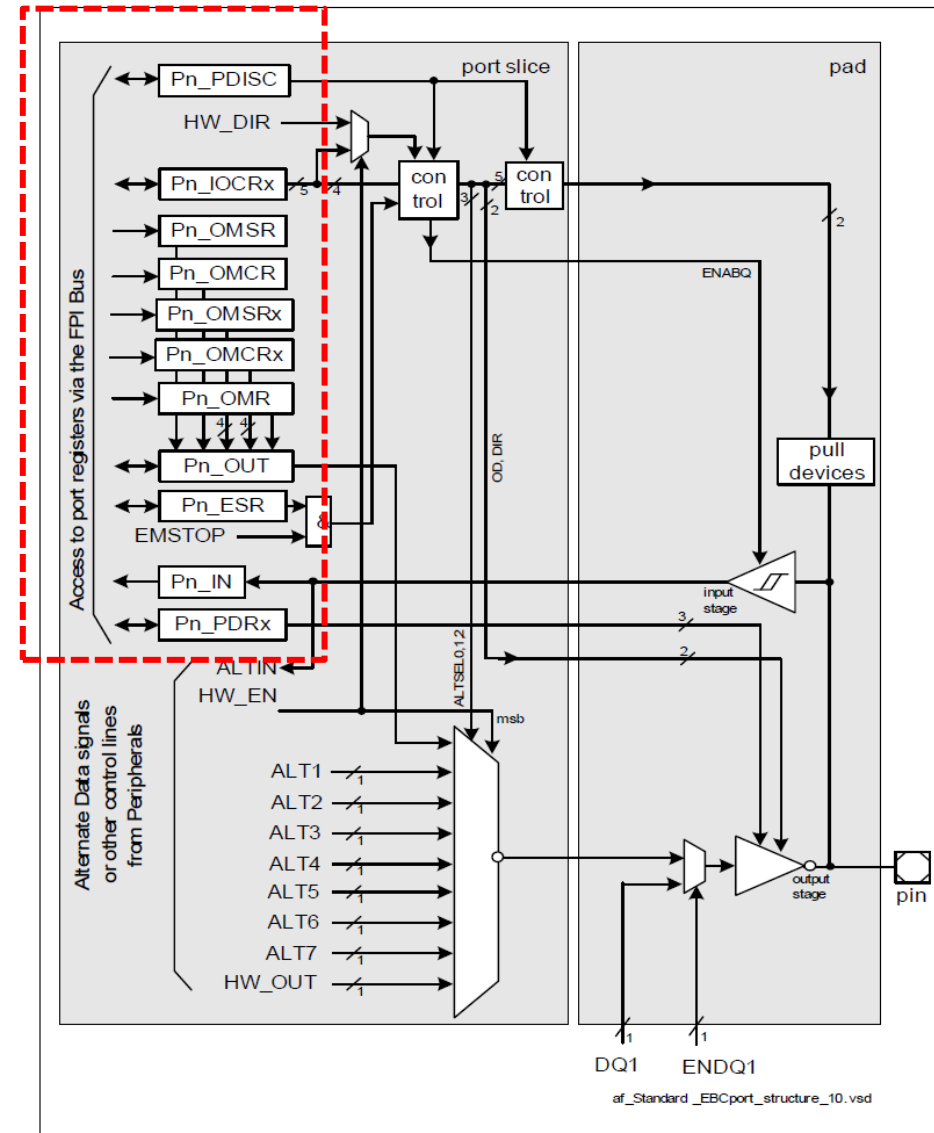
→ GPIO 해당하는 레지스터 설정 필요 →

- 입력으로 사용할 것인지, 출력으로 사용할 것인지
- 풀-업(pull-up)으로 사용할 것인지, 풀-다운(pull-down)으로 사용할 것인지
- 속도를 얼마나 빠르게 사용할 것인지
- etc ...

- MCU의 하드웨어가 “레지스터”라는 이름으로 특정 주소에 설정 (configuration) 가능하도록 매핑 되어 있음

- 본 실습에서 GPIO 동작 목표는

→ P10.1, P10.2 핀의 출력 모드 동작



GPIO 레지스터 설정

- P10.1, P10.2 핀의 출력 모드 동작을 위해서는 어떤 레지스터 설정이 필요?
→ 데이터 시트에서 찾을 수 있음
- GPIO Port 레지스터 항목에서
 - P10_OUT
 - P10_IOCR0
- 2가지 레지스터에 설정이 필요함

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.1163

Table 13-16 Port 10 Functions (cont'd)

Port Pin	I/O	Pin Functionality	Associated Reg./ I/O Line	Port I/O Control Select.	
				Reg./Bit Field	Value
P10.1	I	General-purpose input	P10_IN.P1	P10_IOCR0.PC1	0XXXX _B
		GTM input	TIN103		
		QSPI1 input	MRST1A		
		GPT120 input	T5EUDB		
	O	General-purpose output	P10_OUT.P1		1X000 _B
		GTM output	TOUT103		1X001 _B
		QSPI1 output	MTSR1		1X010 _B
		QSPI1 output	MRST1		1X011 _B
		MSC0	EN01		1X100 _B
		VADC output	VADCG6BFL1		1X101 _B
		MSC0 output	END03		1X110 _B
		Reserved	–		1X111 _B
P10.2	I	General-purpose input	P10_IN.P2	P10_IOCR0.PC2	0XXXX _B
		GTM input	TIN104		
		QSPI1 input	SCLK1A		
		SCU input	REQ2		
		MSC0 input	SDI01		
		GPT120 input	T6INB		
		CAN node 2 input	RXDCAN2E		
	O	General-purpose output	P10_OUT.P2		1X000 _B
		GTM output	TOUT104		1X001 _B
		Reserved	–		1X010 _B
		QSPI1 output	SCLK1		1X011 _B
		MSC0	EN00		1X100 _B
		VADC output	VADCG6BFL2		1X101 _B
		MSC0 output	END02		1X110 _B
		Reserved	–		1X111 _B

GPIO 레지스터 설정 - Address

- 레지스터 설정을 위해 레지스터가 위치한 주소 파악 필요

1. GPIO (Ports) 레지스터 영역 찾기

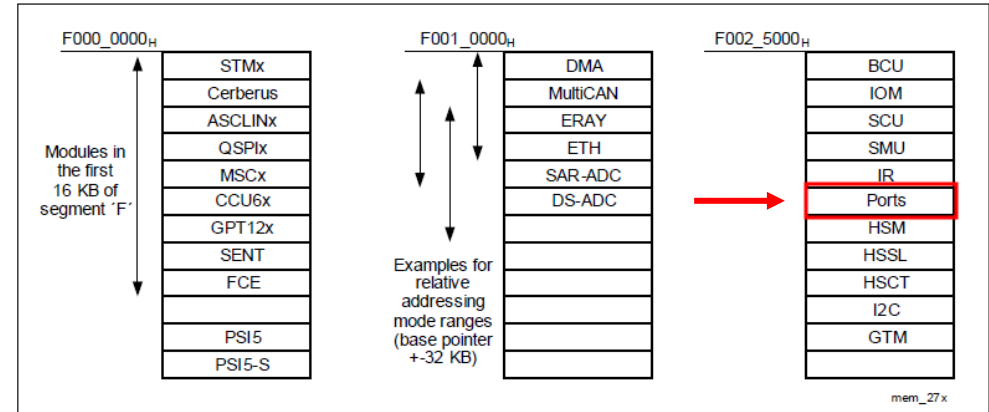


Figure 3-1 Segment F Structure

2. 실습에서 사용하는 Port 10 에 해당하는 레지스터 영역 찾기

- 시작 주소 (Base address)
- = **0xF003B000**

Table 3-3 On Chip Bus Address Map of Segment 15 (cont'd)

Unit	Address Range	Size	Access Type	
			Read	Write
Reserved	F003 A300 _H - F003 AFFF _H	-	SPBBE	SPBBE
Port 10	F003 B000 _H - F003 B0FF _H	256 byte	access	access
Port 11	F003 B100 _H - F003 B1FF _H	256 byte	access	access

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.231

GPIO 레지스터 설정

: P10_IOCRO Direction Reg Addr 계산

3. 사용할 특정 레지스터의 주소 찾기

Infinion-TC27x_D-step-UM-v02_02-EN.pdf p.1074

– P10_IOCRO

– Offset Address = 0x0010

– 즉, Base Address로부터 0x0010만큼 떨어져 있음

→ P10_IOCRO 레지스터 주소

→ = [P10 Base Addr] + [P10_IOCRO Offset Addr]

→ = 0xF003B000 + 0x0010 = **0xF003B010**

Table 13-4 Registers Overview

Register Short Name	Register Long Name	Offset Address	Access Mode		Reset	Desc. see
			Read	Write		
Pn_OUT	Port n Output Register	0000 _H	U, SV	U, SV, P	Application Reset	Page 13-38
Pn_OMR	Port n Output Modification Register	0004 _H	U, SV	U, SV, P	Application Reset	Page 13-39
ID	Module Identification Register	0008 _H	U, SV	BE	Application Reset	Page 13-13
Pn_IOCRO	Port n Input/Output Control Register 0	0010 _H	U, SV	U, SV, P	Application Reset	Page 13-14
Pn_IOCRA	Port n Input/Output Control Register 1	0014 _H	U, SV	U, SV, P	Application Reset	Page 13-14

왜 P10 이 아니라 Pn 으로 표기?

사용할 Port 번호 n = 10

→ Pn = P10

- MCU에서 사용할 수 있는 Port 는 여러 개 존재
✓ Port 1, Port 2, ... Port 10, ...
- 각 Port 들이 동일한 레지스터 구성을 가지기 때문!
- 모든 Port에 대한 레지스터는 동일한 **map**을 참고하기 위해 Pn (n 자리에는 Port 번호) 사용

GPIO 레지스터 설정

: P10_OUT Data Reg Addr 계산

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.1074

3. 사용할 특정 레지스터의 주소 찾기

– P10_OUT

– Offset Address = 0x0000

– 즉, Base Address로부터 0x0000만큼 떨어져 있음

→ P10_OUT 레지스터 주소

→ = [P10 Base Addr] + [P10_OUT Offset Addr]

→ = 0xF003B000 + 0x0000 = **0xF003B000**

Table 13-4 Registers Overview

Register Short Name	Register Long Name	Offset Address	Access Mode		Reset	Desc. see
			Read	Write		
Pn_OUT	Port n Output Register	0000 _H	U, SV	U, SV, P	Application Reset	Page 13-38
Pn_OMR	Port n Output Modification Register	0004 _H	U, SV	U, SV, P	Application Reset	Page 13-39
ID	Module Identification Register	0008 _H	U, SV	BE	Application Reset	Page 13-13
Pn_IDCR0	Port n Input/Output Control Register 0	0010 _H	U, SV	U, SV, P	Application Reset	Page 13-14
Pn_IDCR4	Port n Input/Output Control Register 4	0014 _H	U, SV	U, SV, P	Application Reset	Page 13-14

왜 P10 이 아니라 Pn 으로 표기?

- MCU에서 사용할 수 있는 Port 는 여러 개 존재
 - ✓ Port 1, Port 2, ... Port 10, ...
- 각 Port 들이 동일한 레지스터 구성을 가지기 때문!
- 모든 Port에 대한 레지스터는 동일한 **map**을 참고하기 위해 Pn (n 자리에는 Port 번호) 사용

사용할 Port 번호 n = 10

→ Pn = P10

GPIO 레지스터 설정

: P10_IOCRO에 값을 쓰기 → Output Direction 설정

- P10.1, P10.2를 General Purpose Output (push-pull)으로 사용하기 위해 P10_IOCRO 레지스터에 어떤 값을 써야 하는지 확인

where?

- Pin 1 과 Pin 2 이므로 PC1, PC2 영역에 write 필요



Field	Bits	Type	Description
PC0, PC1, PC2, PC3	[7:3], [15:11], [23:19], [31:27]	rw	Port Control for Port n Pin 0 to 3 This bit field determines the Port n line x functionality (x = 0-3) according to the coding table (see Table 13-5).
0	[2:0], [10:8], [18:16], [26:24]	r	Reserved Read as 0; should be written with 0.

Table 13-5 PCx Coding

PCx[4:0]	I/O	Characteristics	Selected Pull-up / Pull-down / Selected Output Function
10000 _B	Output	Push-pull	General-purpose output
10001 _B	what? - 출력 모드로 사용하기 위해 0x10 값을 PC1, PC2 영역에 write		Alternate output function 1
10010 _B			Alternate output function 2
10011 _B			Alternate output function 3
10100 _B			Alternate output function 4
10101 _B			Alternate output function 5
10110 _B			Alternate output function 6
10111 _B			Alternate output function 7
11000 _B		Open-drain	General-purpose output
11001 _B			Alternate output function 1
11010 _B			Alternate output function 2
11011 _B			Alternate output function 3
11100 _B			Alternate output function 4
11101 _B			Alternate output function 5
11110 _B			Alternate output function 6
11111 _B			Alternate output function 7

1) This is the default pull device setting after reset for powertrain applications.

[Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.1080](#)

[Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.1090](#)

Lab1 – Let's Do it by Yourself

: 프로젝트 환경 복사 및 셋업

현재 열려있는 project
마우스 오른쪽 click

빈 공간 마우스
오른쪽 click

Copy

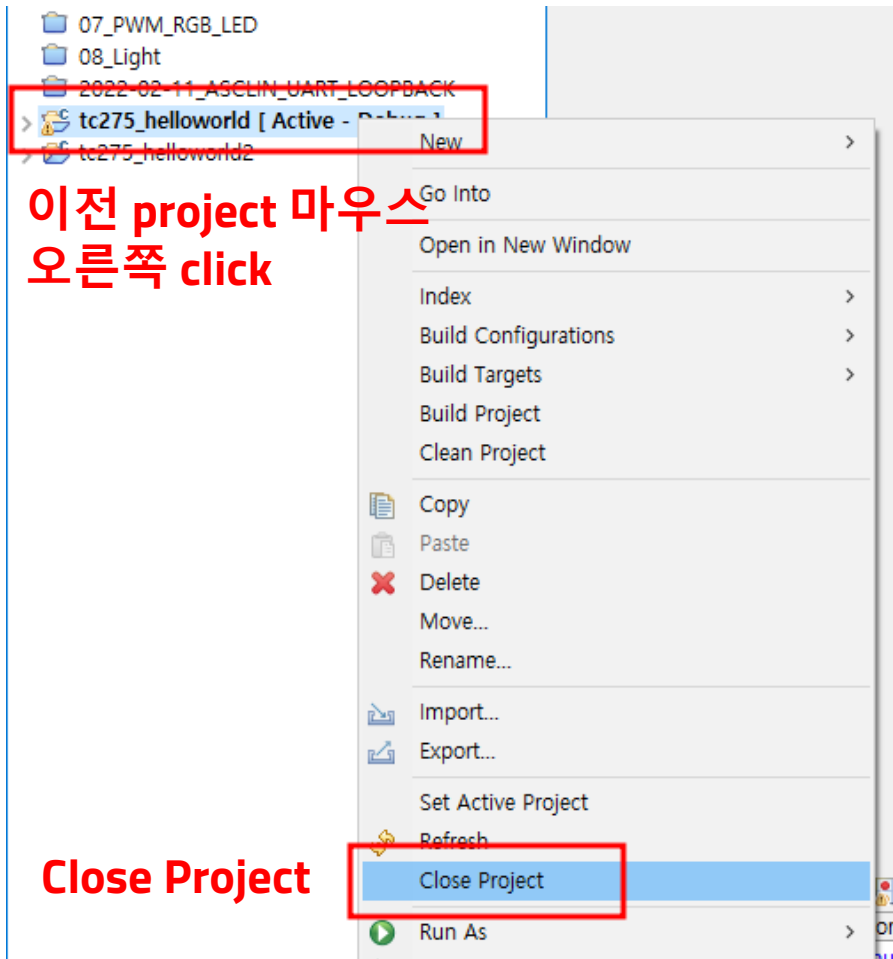
Paste

원하는 project 이름

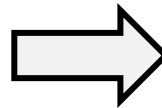
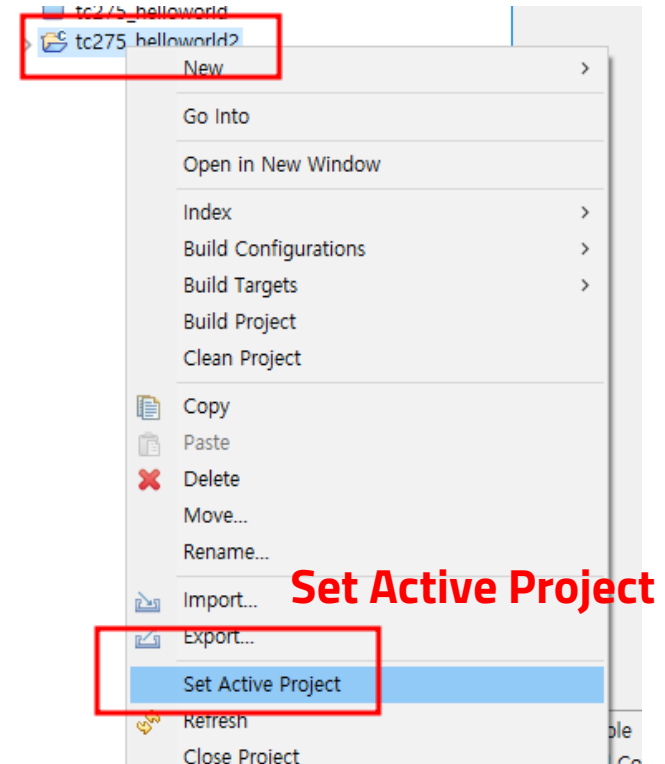
The first screenshot shows a project tree on the left with 'tc275_helloworld' selected. A right-click context menu is open, and the 'Copy' option is highlighted. The second screenshot shows a right-click in an empty space, opening a context menu with the 'Paste' option highlighted. The third screenshot shows a 'Copy Project' dialog box with 'tc275_helloworld2' entered in the 'Project name' field.

Lab1 – Let's Do it by Yourself

: 프로젝트 환경 복사 및 셋업



새로 복사한 project
마우스 오른쪽 click



Lab1 – Let's Do it by Yourself

: P10_IOCR0 레지스터에 값 쓰기

```
30
31 // Port registers
32 #define PC1_BIT_LSB_IDX      11
33 #define PC2_BIT_LSB_IDX      19
34 #define P1_BIT_LSB_IDX       1
35 #define P2_BIT_LSB_IDX       2
36
```

} Port 레지스터 bit shift offset

```
86
87 void initLED(void)
88 {
89     P10_IOCR0.U &= ~(0x1F << PC1_BIT_LSB_IDX); // reset P10_IOCR0 PC1
90     P10_IOCR0.U &= ~(0x1F << PC2_BIT_LSB_IDX); // reset P10_IOCR0 PC2
91
92     P10_IOCR0.U |= 0x10 << PC1_BIT_LSB_IDX; // set P10.1 push-pull general output
93     P10_IOCR0.U |= 0x10 << PC2_BIT_LSB_IDX; // set P10.2 push-pull general output
94 }
95
```


Lab 1 - Memory Map된 하드웨어 영역을 주소로 접근

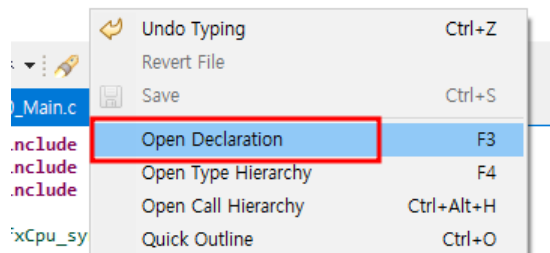
: P10_IOCRO 레지스터 선언부

- LED 사용을 위한 초기화 함수 *initLED()*

```
86
87 void initLED(void)
88 {
89     P10_IOCRO.U &= ~(0x1F << PC1_BIT_LSB_IDX); // reset P10_IOCRO PC1
90     P10_IOCRO.U &= ~(0x1F << PC2_BIT_LSB_IDX); // reset P10_IOCRO PC2
91
92     P10_IOCRO.U |= 0x10 << PC1_BIT_LSB_IDX; // set P10.1 push-pull general output
93     P10_IOCRO.U |= 0x10 << PC2_BIT_LSB_IDX; // set P10.2 push-pull general output
94 }
95
```

AURIX에서 제공하는 헤더파일의
레지스터 주소 확인

- 레지스터 이름 (P10_IOCRO) right click



'Open Declaration'
click



372
373
374
375

```
/** \brief 10, Port Input/Output Control Register 0 */
#define P10_IOCRO /*lint --e(923)*/ (*(volatile Ifx_P_IOCRO*) 0xF003B010u)
```

레지스터의 주소를 가리키는
변수는 *volatile* 키워드 사용



Libraries\Infra\Sfr\TC27D_Reg\IfxPort_reg.h

앞서 데이터 시트에서 찾았던 ↓
P10_IOCRO 레지스터의 주소와 동일
= 0xF003B010

Lab1 - 구조체를 이용한 비트 slice, 공용체 이용한 타입선택

- 레지스터 주소 pointer에서 사용되는 구조체 (struct) / 공용체 (union)

```
372  
373 /** \brief 10, Port Input/Output Control Register 0 */  
374 #define P10_IOCRO /*lint --e(923)*/ (*(volatile Ifx_P_IOCRO* 0xF003B010u)  
375  
376 /** \brief 14, Port Input/Output Control Register 1 */
```

Libraries\Infra\Sfr\TC27D_Reg\IfxPort_reg.h

*Ifx_P_IOCRO*이라는 자료형은 어디로부터?

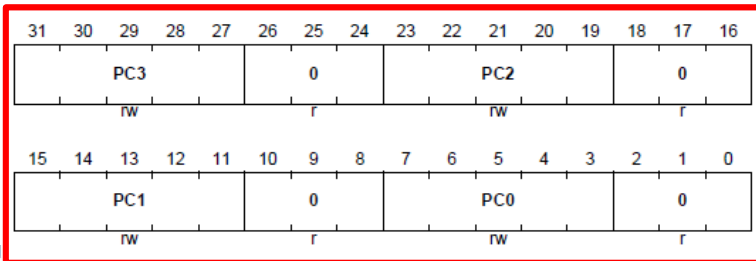
```
86  
87 void initLED(void)  
88 {  
89     P10_IOCRO.U = ~(0x1F << PC1_BIT_LSB_IDX);  
90     P10_IOCRO.U = ~(0x1F << PC2_BIT_LSB_IDX);  
91  
92     P10_IOCRO.U |= 0x10 << PC1_BIT_LSB_IDX;  
93     P10_IOCRO.U |= 0x10 << PC2_BIT_LSB_IDX;  
94 }  
95
```

```
564 /** \brief Port Input/Output Control Register 0 */  
565 typedef union  
566 {  
567     unsigned int U;  
568     Ifx_P_IOCRO_Bits B;  
569 } Ifx_P_IOCRO;  
570  
571
```

Libraries\Infra\Sfr\TC27D_Reg\IfxPort_regdef.h

공용체 (union)의 멤버 중, U 사용 =
unsigned int에 해당하는 크기만큼 사용
→ 변수 P02_IOCRO 은 32bit의 크기를 가짐

공용체 (union)의 형태로 선언되어 있음



```
156  
157 /** \brief Port Input/Output Control Register 0 */  
158 typedef struct Ifx_P_IOCRO_Bits  
159 {  
160     unsigned int reserved_0:3;  
161     unsigned int PC0:5;  
162     unsigned int reserved_8:3;  
163     unsigned int PC1:5;  
164     unsigned int reserved_16:3;  
165     unsigned int PC2:5;  
166     unsigned int reserved_24:3;  
167     unsigned int PC3:5;  
168 } Ifx_P_IOCRO_Bits;  
169
```

레지스터를 이루는 bit 필드 단위로 사용하고 싶으면 공용체 멤버 중, **B** 사용
(LSB가 작은 주소에 위치하므로 → Little Endian)

구조체 bit slice 이용하여 특정 비트만 접근하는 방법

- 레지스터 값을 수정할 수 있는 2가지 방법 - 두 코드는 동일한 결과를 가짐
→ P10_IOCRO 레지스터의 PC1 영역에 0x10 이라는 값을 write 하는 코드

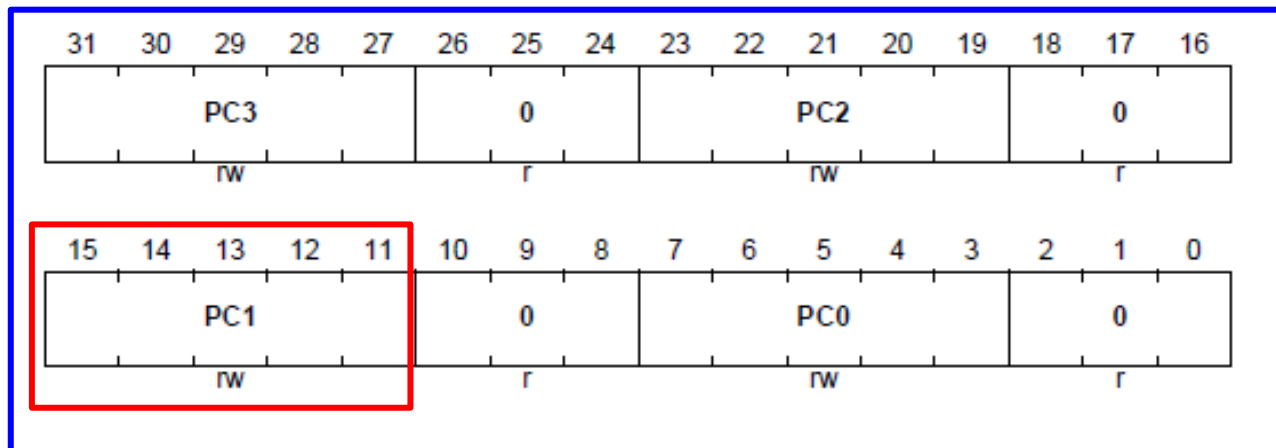
```
5  
6 P10_IOCRO.U |= 0x10 << PC1_BIT_LSB_IDX
```

```
9  
10 P10_IOCRO.B.PC1 = 0x10;
```

→ P10_IOCRO 레지스터에 unsigned int 의 32bit 크기
전체로 접근 (PC1 영역이 시작되는 11번째 bit까지
shift 필요)

P10_IOCRO 레지스터 중, PC1 영역에 해당하는 5개의 bit
에 구조체 필드로 접근 (다른 영역에는 영향 X)

P10_IOCRO 레지스터 32bit



Lab 설명: 비트 Clear

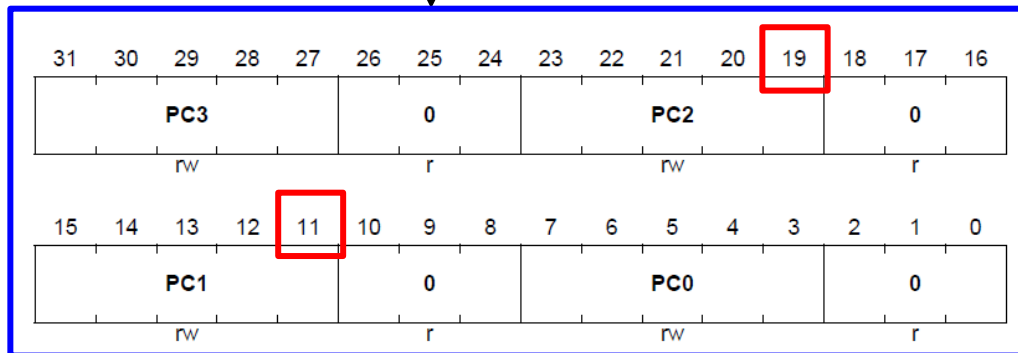
- LED 사용을 위한 초기화 함수 *initLED()*

```
86
87 void initLED(void)
88 {
89     P10_IOCR0.U &= ~(0x1F << PC1_BIT_LSB_IDX); // reset P10_IOCR0 PC1
90     P10_IOCR0.U &= ~(0x1F << PC2_BIT_LSB_IDX); // reset P10_IOCR0 PC2
91
92     P10_IOCR0.U |= 0x10 << PC1_BIT_LSB_IDX; // set P10.1 push-pull general output
93     P10_IOCR0.U |= 0x10 << PC2_BIT_LSB_IDX; // set P10.2 push-pull general output
94 }
95
```

```
30
31 // Port registers
32 #define PC1_BIT_LSB_IDX 11
33 #define PC2_BIT_LSB_IDX 19
34 #define P1_BIT_LSB_IDX 1
35 #define P2_BIT_LSB_IDX 2
36
```

```
86
87 void initLED(void)
88 {
89     P10_IOCR0.U &= ~(0x1F << PC1_BIT_LSB_IDX); // reset P10_IOCR0 PC1
90     P10_IOCR0.U &= ~(0x1F << PC2_BIT_LSB_IDX); // reset P10_IOCR0 PC2
91
92     P10_IOCR0.U |= 0x10 << PC1_BIT_LSB_IDX; // set P10.1 push-pull general output
93     P10_IOCR0.U |= 0x10 << PC2_BIT_LSB_IDX; // set P10.2 push-pull general output
94 }
95
```

P10_IOCR0 레지스터 32bit shift 하는 숫자 (BIT_LSB_IDX)의 의미



PC2, PC1의 bit 위치만큼 0x1F 값을 left shift

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.1080

"0"과 AND 연산의 의미

$\sim(0x1F \ll 11)$
 $= \sim 0x0000F800$
 $= 0xFFFF07FF$

PC1 영역 "0"을 AND 연산하면 반드시 0의 결과
(나머지는 "1"과 AND 연산하면 값 유지)

전체 레지스터 중, PC2 또는 PC1 등 특정 영역만을 0으로 clear

Lab 설명: Bit Set

- LED 사용을 위한 초기화 함수 *initLED()*

```
86
87 void initLED(void)
88 {
89     P10_IOCR0.U &= ~(0x1F << PC1_BIT_LSB_IDX); // reset P10_IOCR0 PC1
90     P10_IOCR0.U &= ~(0x1F << PC2_BIT_LSB_IDX); // reset P10_IOCR0 PC2
91
92     P10_IOCR0.U |= 0x10 << PC1_BIT_LSB_IDX; // set P10.1 push-pull general output
93     P10_IOCR0.U |= 0x10 << PC2_BIT_LSB_IDX; // set P10.2 push-pull general output
94 }
95
```

OR 연산을 사용한 레지스터 write

$0x10 \ll 11$
 $= 0x00008000$

PC1 영역에 0x10을 OR
연산하면 해당 위치의
비트만 값이 변경됨
(나머지는 "0"과 OR
연산하여 값 유지)

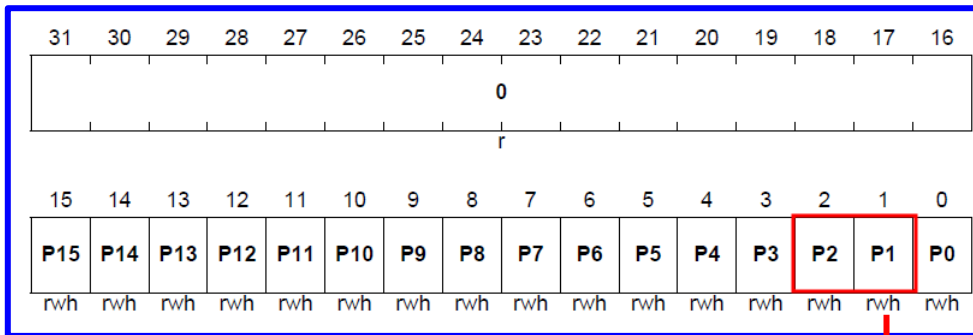
전체 레지스터 중, PC2 또는 PC1 등
특정 영역에만 값 write

GPIO 레지스터 설정

: P10_OUT 에 값을 쓰기 → Output Value 설정

- P10.1, P10.2 출력에 HIGH("1") 또는 LOW("0")의 값을 인가하기 위해 P10_OUT 레지스터에 어떤 값을 써야 하는지 확인

P10_OUT 레지스터의 32bit 전체 영역



Field	Bits	Type	Description
Px (x = 0-15)	x	rwh	Port n Output Bit x This bit determines the level at the output pin Pn.x if the output is selected as GPIO output. 0 _B The output level of Pn.x is 0. 1 _B The output level of Pn.x is 1. Pn.x can also be set or cleared by control bits of the Pn_OMSR, Pn_OMCR or Pn_OMR registers.
0	[31:16]	r	Reserved Read as 0; should be written with 0.

where?

- Pin 1 과 Pin 2 이므로 P1, P2 영역에 write 필요
- 전체 32bit 중, P2, P1 영역은 1bit 씩 차지

what?

- 출력하고자 하는 값("0" 또는 "1")을 P1, P2 영역에 write

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.1103

Lab2 – Let's Do it by Yourself

: P10_OUT에 값을 써서, 출력 제어해보자, (토글)

```
43
44 int core0_main(void)
45 {
46     IfxCpu_enableInterrupts();
47
48     /* !!WATCHDOG0 AND SAFETY WATCHDOG ARE DISABLED HERE!!
49      * Enable the watchdogs and service them periodically if it is required
50      */
51     IfxScuWdt_disableCpuWatchdog(IfxScuWdt_getCpuWatchdogPassword());
52     IfxScuWdt_disableSafetyWatchdog(IfxScuWdt_getSafetyWatchdogPassword());
53
54     /* Wait for CPU sync event */
55     IfxCpu_emitEvent(&g_cpuSyncEvent);
56     IfxCpu_waitEvent(&g_cpuSyncEvent, 1);
57
58     initLED();
59
60
61     while(1)
62     {
63         // 1. GPIO OUT
64         for(uint32 i = 0; i < 10000000; i++); // delay
65         P10_OUT.U ^= 0x1 << P1_BIT_LSB_IDX; // toggle P10.1 (LED D12 RED)
66
67         for(uint32 i = 0; i < 10000000; i++); // delay
68         P10_OUT.U ^= 0x1 << P2_BIT_LSB_IDX; // toggle P10.2 (LED D13 BLUE)
69
70     }
71     return (1);
72 }
73
74
```


Lab 2: P10_OUT 셋팅

```

30
31 // Port registers
32 #define PC1_BIT_LSB_IDX      11
33 #define PC2_BIT_LSB_IDX      19
34 #define P1_BIT_LSB_IDX       1
35 #define P2_BIT_LSB_IDX       2
36

```

- main 함수 while문 내에 원하는 동작에 대한 코드를 작성

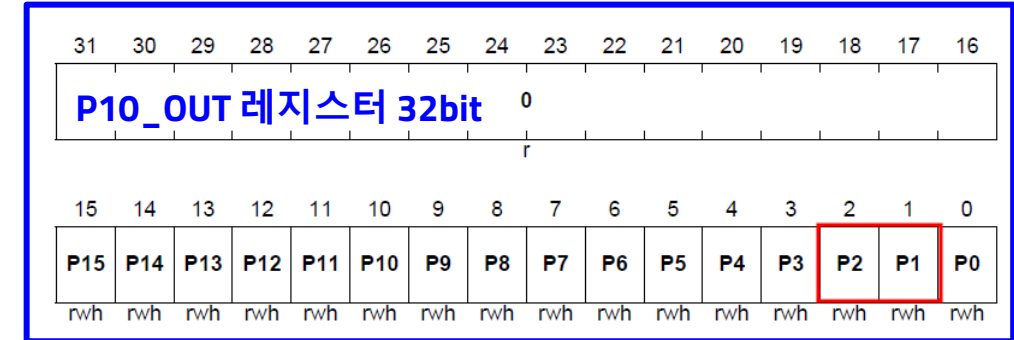
Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.1103

```

43
44 int core0_main(void)
45 {
46     IfxCpu_enableInterrupts();
47
48     /* !!WATCHDOG0 AND SAFETY WATCHDOG ARE DISABLED HERE!!
49      * Enable the watchdogs and service them periodically if it is required
50      */
51     IfxScuWdt_disableCpuWatchdog(IfxScuWdt_getCpuWatchdogPassword());
52     IfxScuWdt_disableSafetyWatchdog(IfxScuWdt_getSafetyWatchdogPassword());
53
54     /* Wait for CPU sync event */
55     IfxCpu_emitEvent(&g_cpuSyncEvent);
56     IfxCpu_waitEvent(&g_cpuSyncEvent, 1);
57     initLED();
58
59
60
61     while(1)
62     {
63         // 1. GPIO OUT
64         for(uint32 i = 0; i < 10000000; i++); // delay
65         P10_OUT.U ^= 0x1 << P1_BIT_LSB_IDX; // toggle P10.1 (LED D12 RED)
66
67         for(uint32 i = 0; i < 10000000; i++); // delay
68         P10_OUT.U ^= 0x1 << P2_BIT_LSB_IDX; // toggle P10.2 (LED D13 BLUE)
69
70     }
71     return (1);
72 }
73
74
408
409 /** \brief 0, Port Output Register */
410 #define P10_OUT /*lint --e(923)*/ (*(volatile Ifx_P_OUT*)0xF003B000u)
411
412 /** \brief 40. Port Pad Driver Mode 0 Register */

```

LED 초기화



XOR 연산을 사용한
bit toggle

P1 영역과 "1"값을 XOR
연산하면, 두 값이 다를
경우 1을 출력
→ 현재 bit값이 "0"인
경우 "1"을 write
→ 현재 bit값이 "1"인
경우 "0"을 write
toggle 동작을 수행

앞서 데이터 시트에서 찾았던 P10_OUT 레지스터의
주소와 동일 = 0xF003B000

Lab1, Lab2 수행 결과

SW가 HW를 제어

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.1067

- SW에서 변수의 형태로 특정 주소의 레지스터에 write한 값이 하드웨어에 미치는 영향
 - P10_IOCRO 레지스터에 의해 pin으로 출력되는 값이 P10_OUT으로 multiplexing 되도록 함

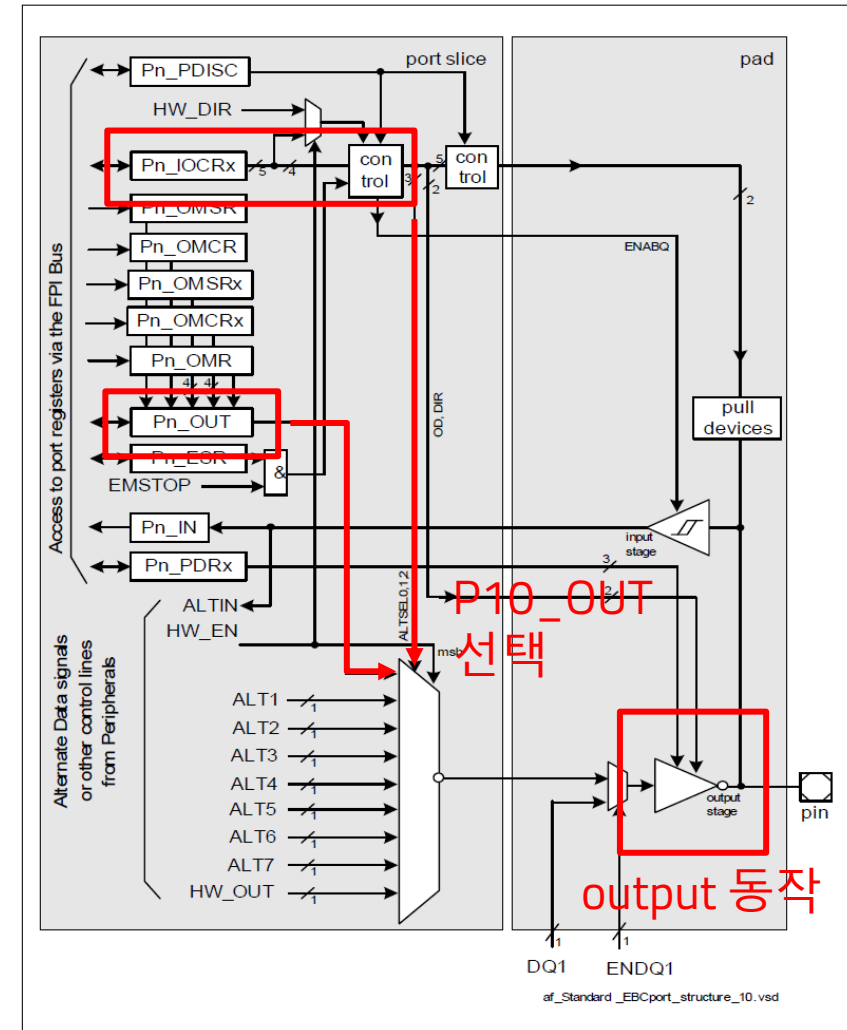
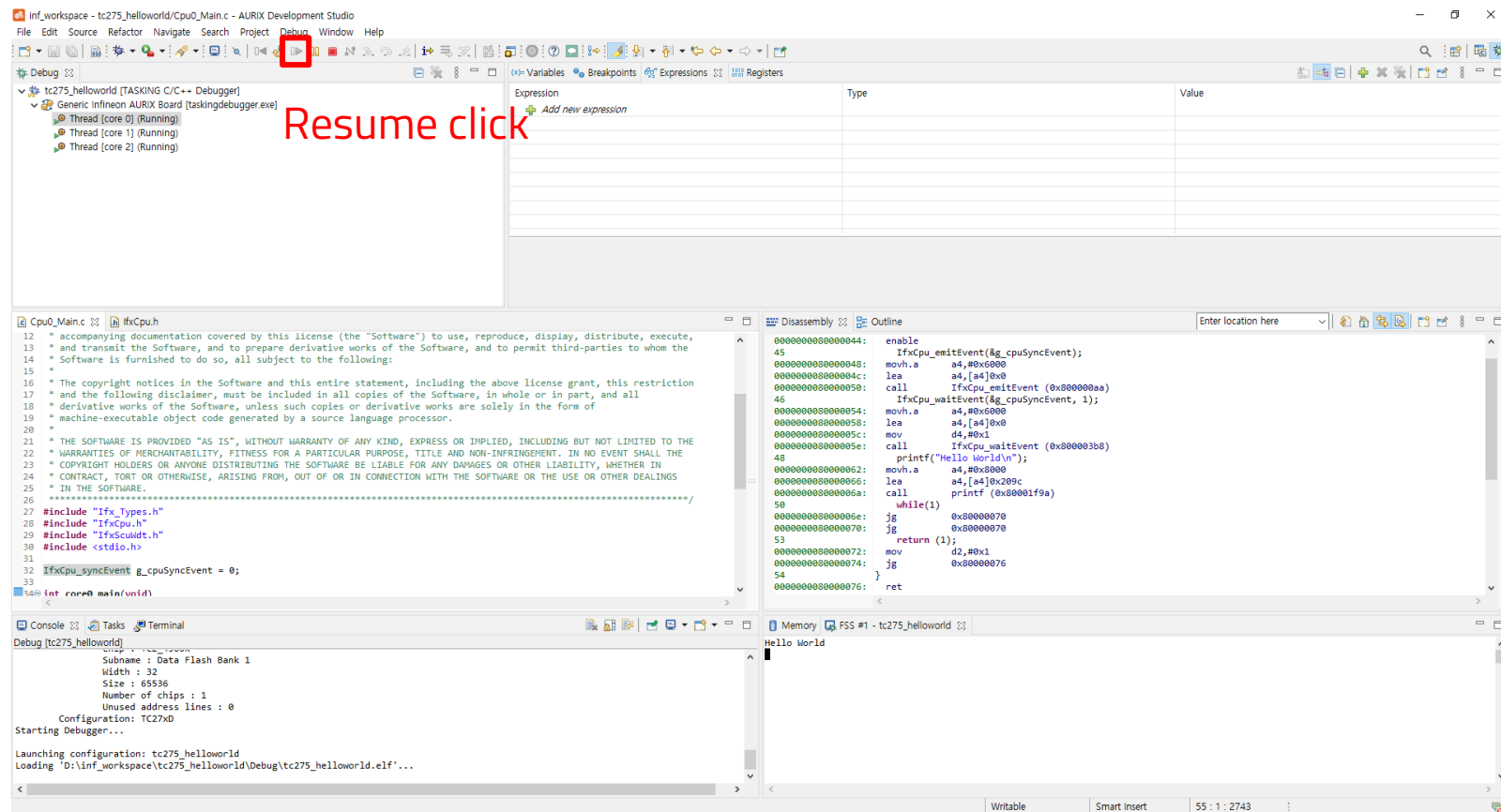


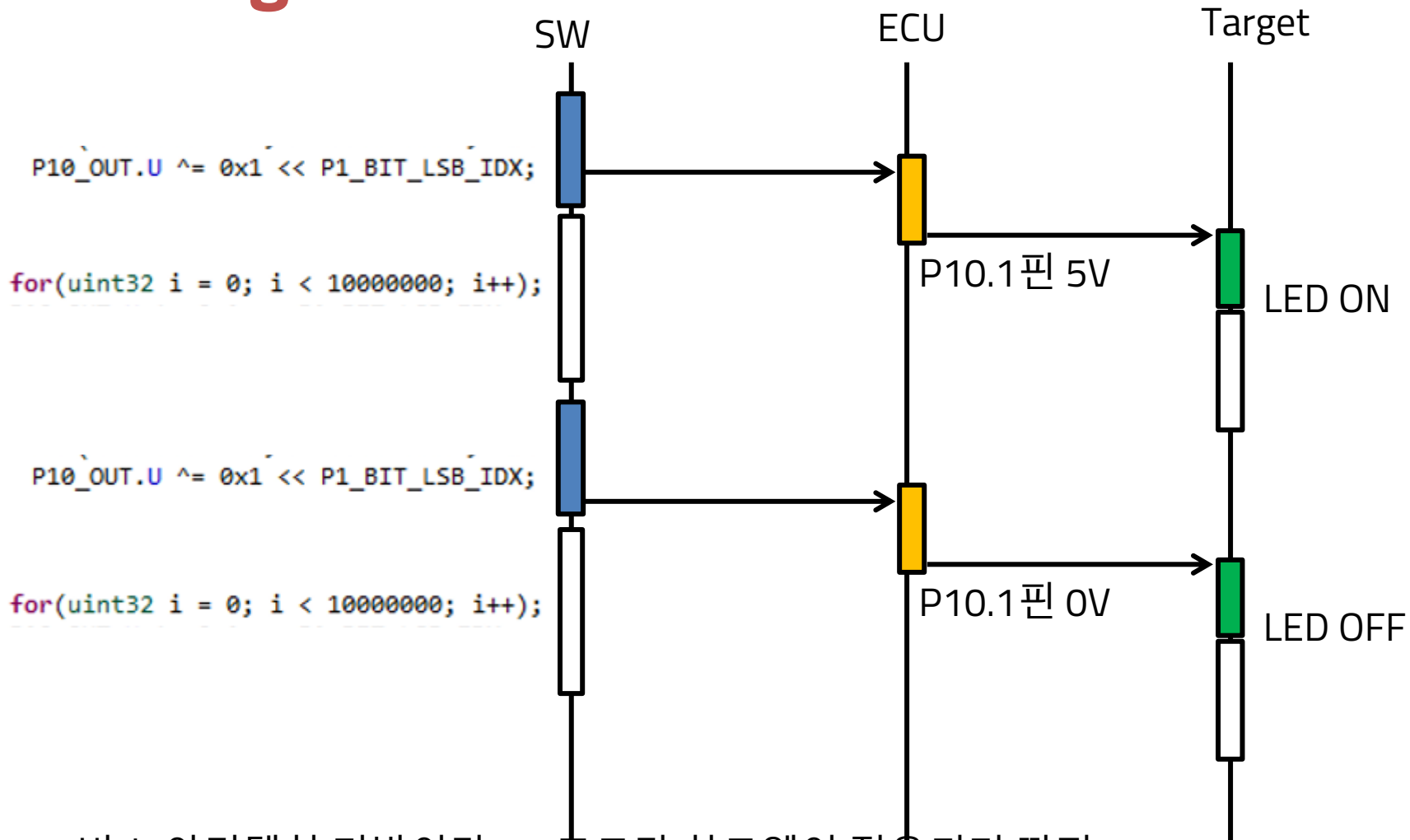
Figure 13-1 General Structure of a Port Pin

Build 및 Debug

- 프로젝트 빌드 (ctrl + b)
- 디버그 수행하여 보드에 실행 파일 flash



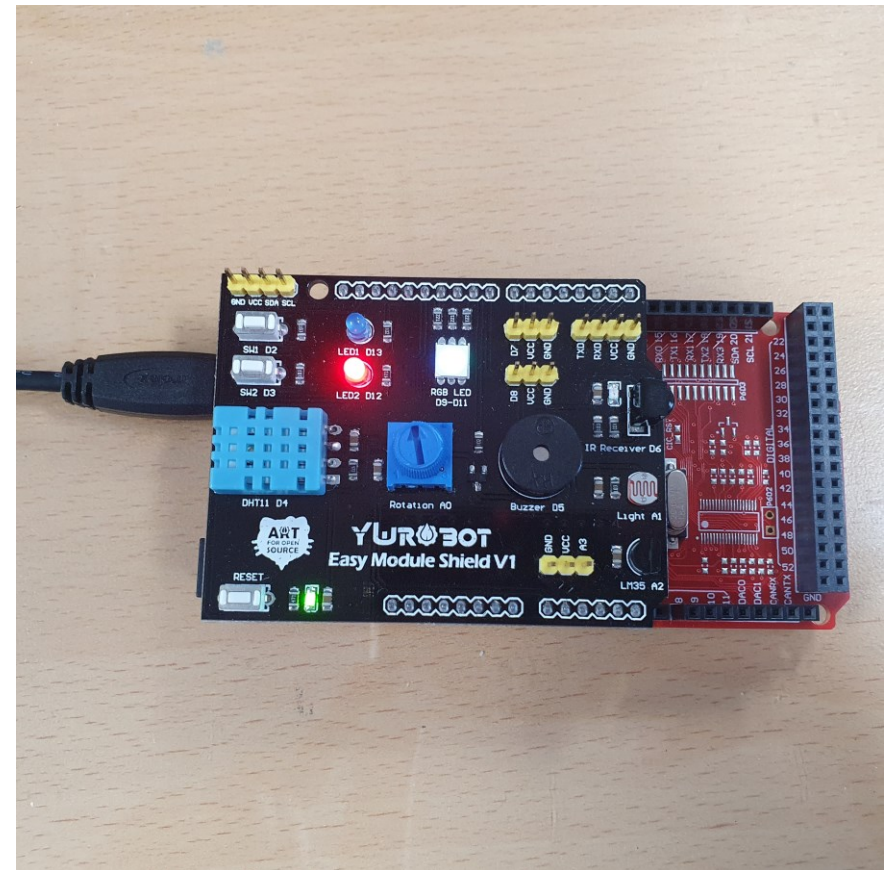
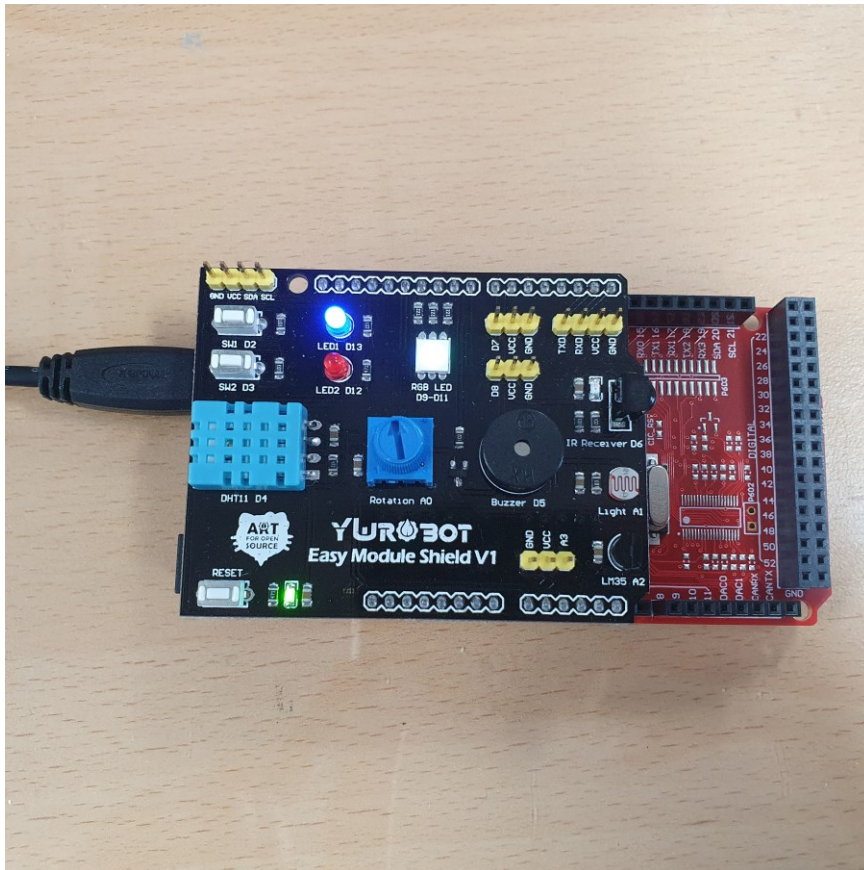
Polling-based SW-HW Interaction



버스 아키텍처 기반이라 SW코드가 하드웨어 적용되기 까지 시간이 걸림, overrun을 방지하기 위해 delay 필요함. (폴링 방식일때 사용, 다음에 인터럽트 방식일때는 flag체크 필요)

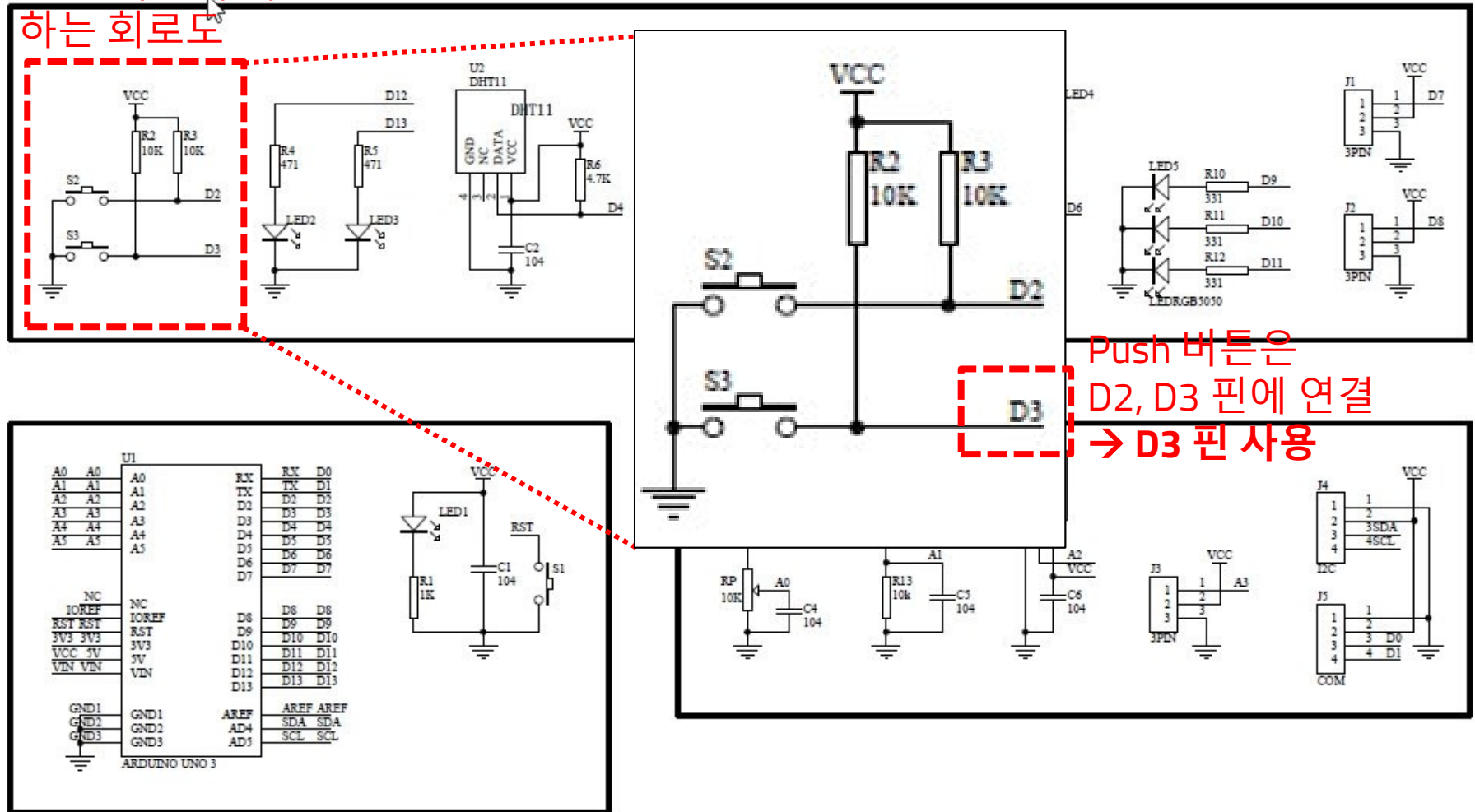
동작 확인

- 두 LED가 차례로 toggle 하는 모습 확인 가능



Push 버튼 회로 구조 파악

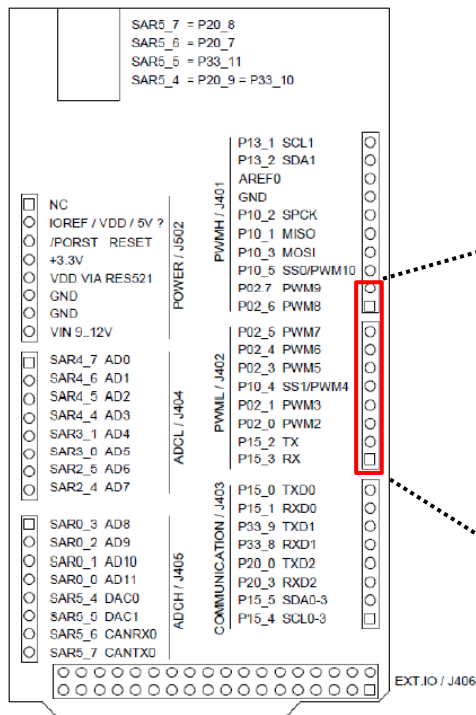
Push 버튼에 해당
하는 회로도



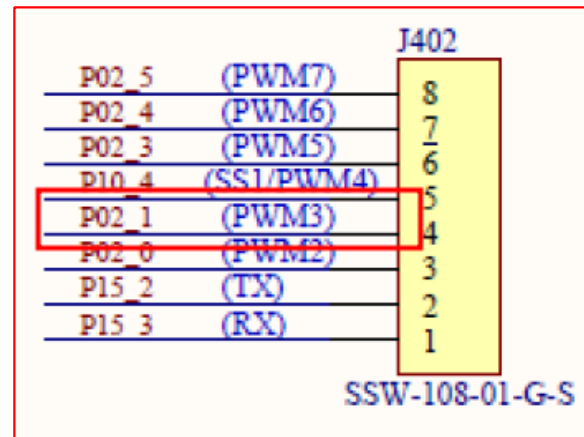
Pin Map @ TC275 보드

- 사용할 Push 버튼 (SW2) 은 확장 보드의 핀 D3에 연결 되어있는 것을 확인
→ 그렇다면 확장 보드의 핀 D3은 TC275 보드의 어느 핀에 연결?

[Infineon-ShieldBuddy_TC275 -UM-v02_08-EN.pdf p.44](#)



Digital pin 1 (TX0)	PWML.2	P15.2
Digital pin 2 (PWM)	PWML.3	P2.0
Digital pin 3 (PWM)	PWML.4	P2.1
Digital pin 5 (PWM)	PWML.6	P2.3
Digital pin 6 (PWM)	PWML.7	P2.4
Digital pin 7 (PWM)	PWML.8	P2.5



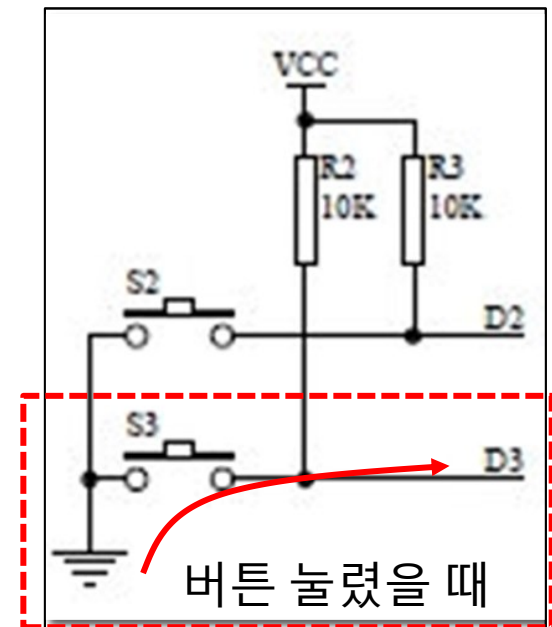
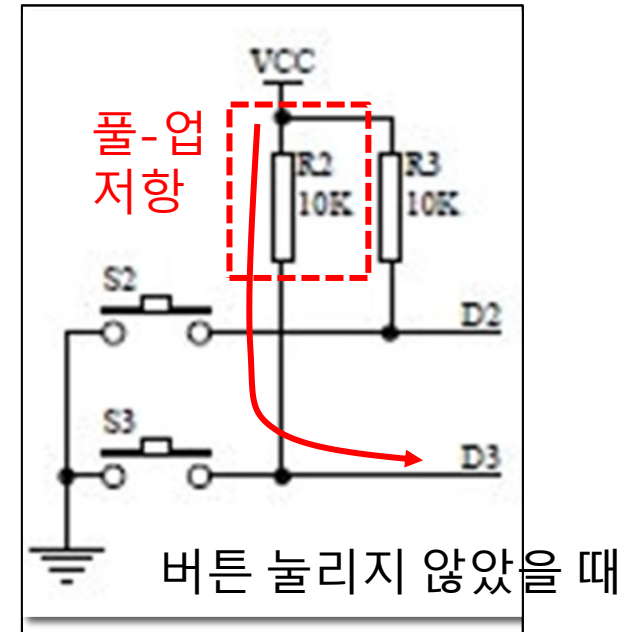
TC275 MCU I/O 중,
Port 2 pin 1 (P02.1)
와 연결됨

[Infineon-ShieldBuddy_TC275 -UM-v02_08-EN.pdf p.42](#)

[Infineon-ShieldBuddy_TC275 -UM-v02_08-EN.pdf p.46](#)

Push 버튼 상태를 읽기 위해

- Push 버튼 회로를 보면 D3 핀에 VCC가 pull-up으로 연결 되어있음
 - 버튼이 눌리지 않으면 핀에 HIGH가 전달됨
 - 반대로 버튼이 눌리면 핀에 LOW가 전달됨
- 즉, LOW 값이 입력될 때 Push 버튼이 눌러 있는 상태인 것
- **GPIO를 입력 모드로 사용해서 MCU의 핀이 HIGH 또는 LOW 값인지 확인해야 함**



GPIO 레지스터 설정

- P02.1 핀의 입력 모드 동작을 위해서는 어떤 레지스터 설정이 필요?
→ 데이터 시트에서 찾을 수 있음
- GPIO Port 레지스터 항목에서
 - **P02_IN**
 - **P02_IOCR0**
- 2가지 레지스터에 설정이 필요함

Table 13-14 Port 02 Functions (cont'd)

Port Pin	I/O	Pin Functionality	Associated Reg./ I/O Line	Port I/O Control Select.	
				Reg./Bit Field	Value
P02.1	I	General-purpose input	P02_IN.P1	P02_IOCR0. PC1	0XXXX _B
		GTM input	TIN1		
		ASCLIN2 input	ARX2B		
		CAN node 0 input	RXDCAN0A		
		ERAY input	RXDA2		
		CIF input	CIFD1		
		SCU input	REQ14		
	O	General-purpose output	P02_OUT.P1		1X000 _B
		GTM output	TOUT1		1X001 _B
		Reserved	–		1X010 _B
		QSPI3 output	SLSO32		1X011 _B
		DSADC output	DSCGPWMP		1X100 _B
		Reserved	–		1X101 _B
		Reserved	–		1X110 _B
		CCU60 output	COOUT60		1X111 _B
P02.2	I	General-purpose input	P02_IN.P2	P02_IOCR0	0XXXX

[Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.1151](#)

GPIO 레지스터 설정

Port02 Group Base Address

- 레지스터 설정을 위해 레지스터가 위치한 주소 파악 필요

1. GPIO (Ports) 레지스터 영역 찾기

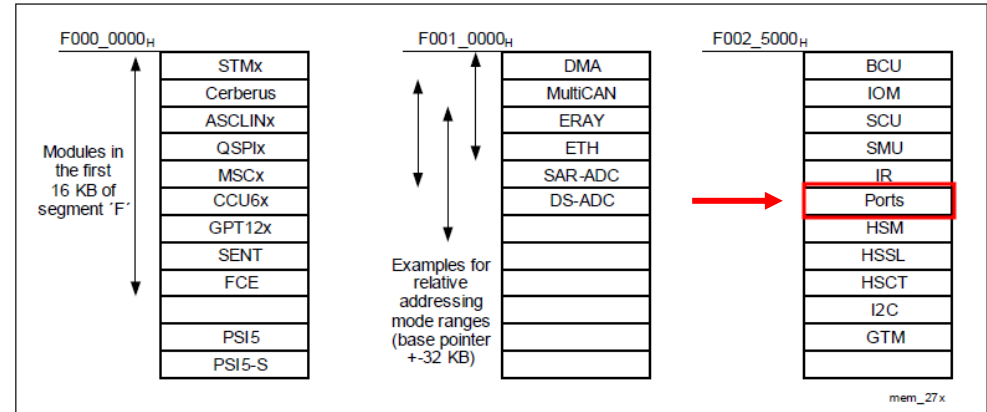


Figure 3-1 Segment F Structure

[Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.227](#)

2. 실습에서 사용하는 Port 02 에 해당하는 레지스터 영역 찾기

- 시작 주소 (Base address)
- = **0xF003A200**

Interrupt Router (IR) SRV Registers	F003_0000_H - F003_9FFF_H	0 Kbyte	access	access
Port 00	F003 A000_H - F003 A0FF_H	256 byte	access	access
Port 01	F003 A100_H - F003 A1FF_H	256 byte	access	access
Port 02	F003 A200_H - F003 A2FF_H	256 byte	access	access

[Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.230](#)

GPIO 레지스터 설정 - Address 계산

3. 사용할 특정 레지스터의 주소 찾기

- P02_IN

- Offset Address = 0x0024

- 즉, **Base Address**로부터 **0x0024**만큼 떨어져 있음

→ P02_IN 레지스터 주소

→ = [P02 Base Addr] + [P02_IN Offset Addr]

→ = 0xF003A200 + 0x0024 = **0xF003A224**

- P02_IOCR0

- Offset Address = 0x0010

- 즉, **Base Address**로부터 **0x0010**만큼 떨어져 있음

→ P02_IOCR0 레지스터 주소

→ = [P02 Base Addr] + [P02_IOCR0 Offset Addr]

→ = 0xF003A200 + 0x0010 = **0xF003A210**

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.1075

Table 13-4 Registers Overview (cont'd)

Register Short Name	Register Long Name	Offset Address	Access Mode		Reset	Desc. see
			Read	Write		
Pn_IOCRR12	Port n Input/Output Control Register 12	001C _H	U, SV	U, SV, P	Application Reset	Page 13-14
-	Reserved	0020 _H	BE	BE	-	-
Pn_IN	Port n Input Register	0024 _H	U, SV	BE	Application Reset	Page 13-52
-	Reserved	0028 _H - 003C _H	BE	BE	-	-

→ 사용할 Port 번호 n = 02

→ Pn = P02

Table 13-4 Registers Overview

Register Short Name	Register Long Name	Offset Address	Access Mode		Reset	Desc. see
			Read	Write		
Pn_OUT	Port n Output Register	0000 _H	U, SV	U, SV, P	Application Reset	Page 13-8
Pn_OMR	Port n Output Modification Register	0004 _H	U, SV	U, SV, P	Application Reset	Page 13-9
ID	Module Identification Register	0008 _H	U, SV	BE	Application Reset	Page 13-3
Pn_IOCRR0	Port n Input/Output Control Register 0	0010 _H	U, SV	U, SV, P	Application Reset	Page 13-14
Pn_IOCRR4	Port n Input/Output Control Register 4	0014 _H	U, SV	U, SV, P	Application Reset	Page 13-14

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.1074

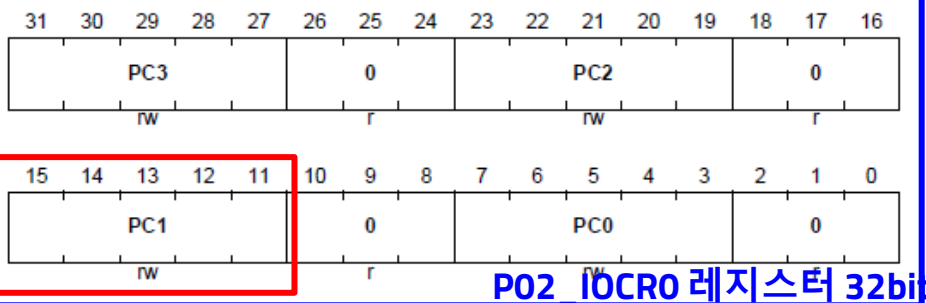
P02.1 포트 Direction 설정

: P02_IOCRO 레지스터에 값 쓰기

- P02.1를 General Purpose Input 으로 사용하기 위해 P02_IOCRO 레지스터에 어떤 값을 써야 하는지 확인

where?

- Pin 1 이므로 PC1 영역에 write 필요



Field	Bits	Type	Description
PC0, PC1, PC2, PC3	[7:3], [15:11], [23:19], [31:27]	rw	Port Control for Port n Pin 0 to 3 This bit field determines the Port n line x functionality (x = 0-3) according to the coding table (see Table 13-5).
0	[2:0], [10:8], [18:16], [26:24]	r	Reserved Read as 0; should be written with 0.

P02_IOCRO 레지스터 32bit 중,
PC1 영역에 해당하는 5bit

[Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.1080](#)

what?

- 풀-업 회로가 연결된 입력 모드로 사용하기 위해 0x02 값을 PC1 영역에 write
- (X 는 무관항(don't care)을 의미)

Table 13-5 PCx Coding

PCx[4:0]	I/O	Characteristics	Selected Pull-up / Pull-down / Selected Output Function
0XX00 _B	Input	–	No input pull device connected, tri-state mode
0XX01 _B			Input pull-down device connected
0XX10 _B			Input pull-up device connected ¹⁾
0XX11 _B			No input pull device connected, tri-state mode

[Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.1089](#)

Lab 3: P02_IOCR0이용하여 Direction 제어

```
95
96 void initButton(void)
97 {
98     P02_IOCR0.U &= ~(0x1F << PC1_BIT_LSB_IDX);    // reset P02_IOCR0 PC1
99
100    P02_IOCR0.U |= 0x02 << PC1_BIT_LSB_IDX;        // set P02.1 general input (pull-up connected)
101 }
102
```

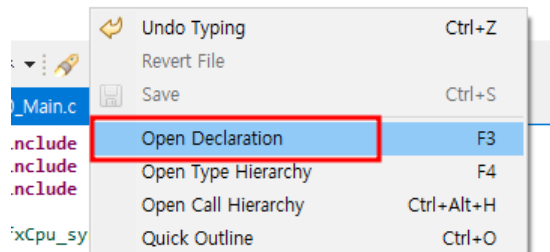
Lab 3 설명: P02_IOCRO 레지스터 정의부

- Push 버튼 사용을 위한 초기화 함수 *initButton()*

```
95
96 void initButton(void)
97 {
98     P02_IOCRO.U &= ~(0x1F << PC1_BIT_LSB_IDX);    // reset P02_IOCRO PC1
99
100    P02_IOCRO.U |= 0x02 << PC1_BIT_LSB_IDX;        // set P02.1 general input (pull-up connected)
101 }
102
```

AURIX에서 제공하는 헤더파일의
레지스터 주소 확인

- 레지스터 이름 (P02_IOCRO) right click



'Open Declaration'
click

```
306 #define P02_IN /*lint --e(923)*/ (*(volatile Ifx_P_IN*)0xF003A224u)
307
308 /** \brief 10, Port Input/Output Control Register 0 */
309 #define P02_IOCRO /*lint --e(923)*/ (*(volatile Ifx_P_IOCRO*)0xF003A210u)
310
311 /** \brief 14, Port Input/Output Control Register 4 */
312 #define P02_IOCRO4 /*lint --e(923)*/ (*(volatile Ifx_P_IOCRO4*)0xF003A214u)
```

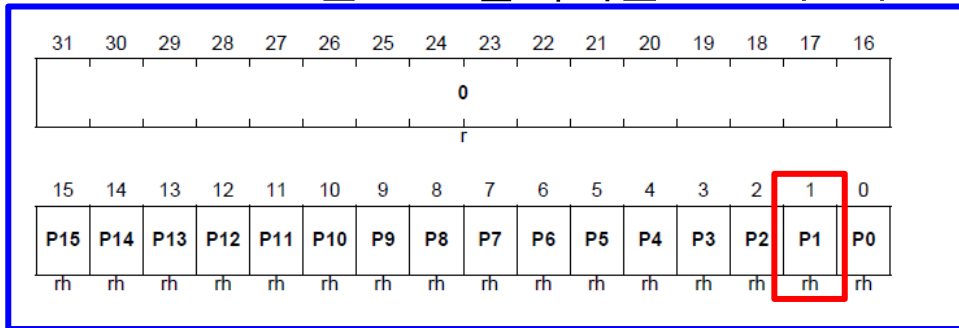
레지스터의 주소를 가리키는
변수는 *volatile* 키워드 사용

Libraries\Infra\Sfr\TC27D_Reg\IfxPort_reg.h

앞서 데이터 시트에서 찾았던
P02_IOCRO 레지스터의 주소와 동일
= 0xF003A210

GPIO 레지스터 설정 – P02_IN값 Read

- P02.1 핀으로 입력되는 HIGH("1") 또는 LOW("0")의 값을 읽기 위한 레지스터



where?

- Pin 1 이므로 P1 영역을 read

User's Manual
Ports, V2.0

13-52

P02_IN 레지스터 32bit

V2.2 2014-12



TC27x D-Step

General Purpose I/O Ports and Peripheral I/O Lines (Ports)

Field	Bits	Type	Description
Px (x = 0-15)	x	rh	Port n Input Bit x This bit indicates the level at the input pin Pn.x. 0 _B The input level of Pn.x is 0. 1 _B The input level of Pn.x is 1.
0	[31:16]	r	Reserved Read as 0.

what?

- 입력되는 값이 HIGH이면 P1 영역의 값이 "1"
- 입력되는 값이 LOW이면 P1 영역의 값이 "0"

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.1118

Lab 4: P02.1 포트 값에 따른 출력 제어

- main 함수 while문 내에 원하는 동작에 대한 코드를 작성

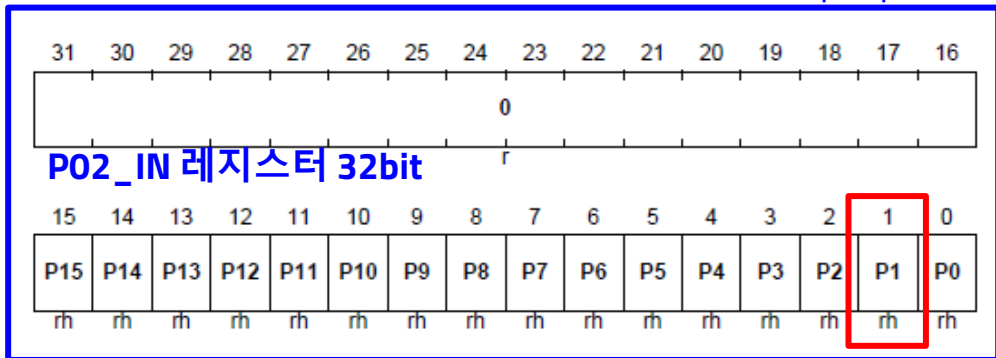
Infineon-TC27x_D-step-UM-
v02_02-EN.pdf p.1103

```
53
54 /* Wait for CPU sync event */
55 IfxCpu_emitEvent(&g_cpuSyncEvent);
56 IfxCpu_waitEvent(&g_cpuSyncEvent, 1);
57
58 initLED();
59 initButton();
60
61 while(1)
62 {
63     /*
64     // 1. GPIO OUT
65     for(uint32 i = 0; i < 10000000; i++); // delay
66     P10_OUT.U ^= 0x1 << P1_BIT_LSB_IDX; // toggle P10.1 (LED D12 RED)
67
68     for(uint32 i = 0; i < 10000000; i++); // delay
69     P10_OUT.U ^= 0x1 << P2_BIT_LSB_IDX; // toggle P10.2 (LED D13 BLUE)
70     */
71
72     // 2. GPIO IN
73     if( (P02_IN.U & (0x1 << P1_BIT_LSB_IDX)) == 0 // check button status, is it pushed?
74     {
75         P10_OUT.U |= 0x1 << P1_BIT_LSB_IDX; // set P10.1 (LED D13 RED)
76     }
77     else
78     {
79         P10_OUT.U &= ~(0x1 << P1_BIT_LSB_IDX); // clear P10.1 (LED D13 RED)
80     }
81 }
82
83 return (1);
84 }
85
86
```

LED, Button 초기화

AND 연산을 사용한
bit Read

다른 영역의 값과
관계없이, P1 영역만의
값을 읽을 때,
P1 영역에만 "1" 값을
채워 AND 연산하면, P1
영역의 값만 남음
(Bit Masking)



앞서 데이터 시트에서 찾았던 P02_IN 레지스터의
주소와 동일 = 0xF003A224

Lab 3, Lab 4 실행결과 SW가 HW를 제어함.

Infinion-TC27x_D-step-UM-v02_02-EN.pdf p.1067

- SW에서 변수의 형태로 특정 주소의 레지스터에 write한 값이 하드웨어에 미치는 영향
 - P02_IOCRO 레지스터에 의해 pin0이 입력으로 동작하도록 함

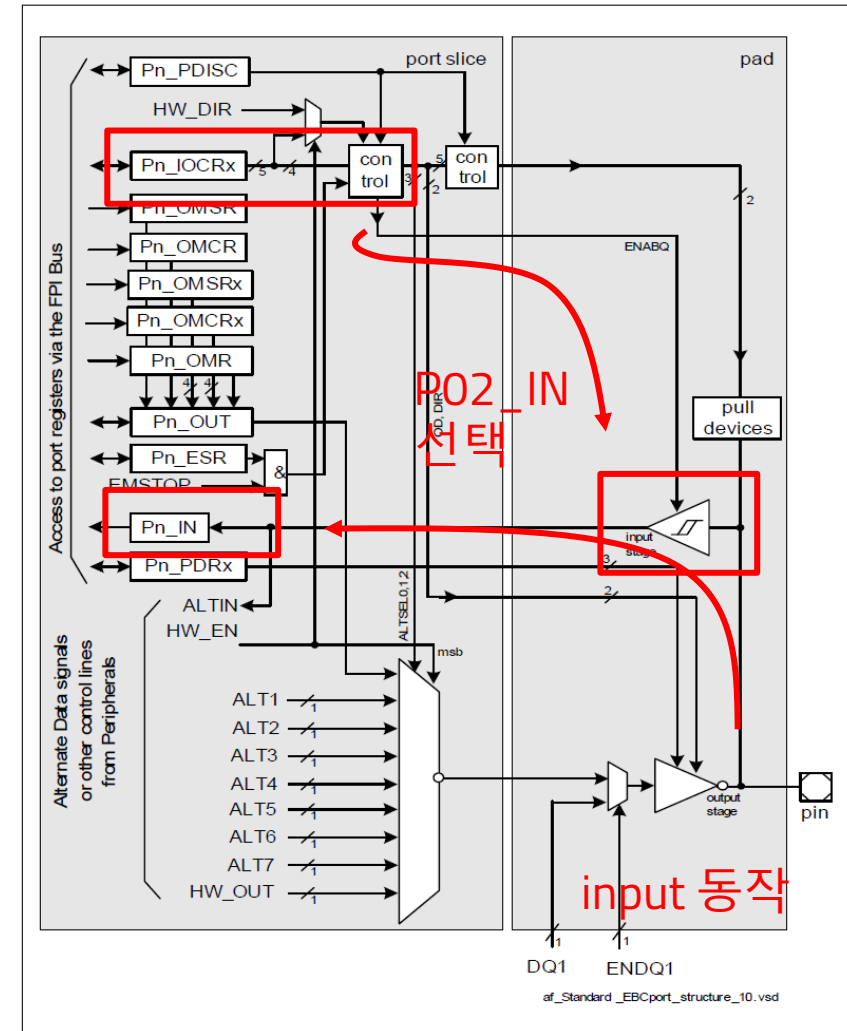
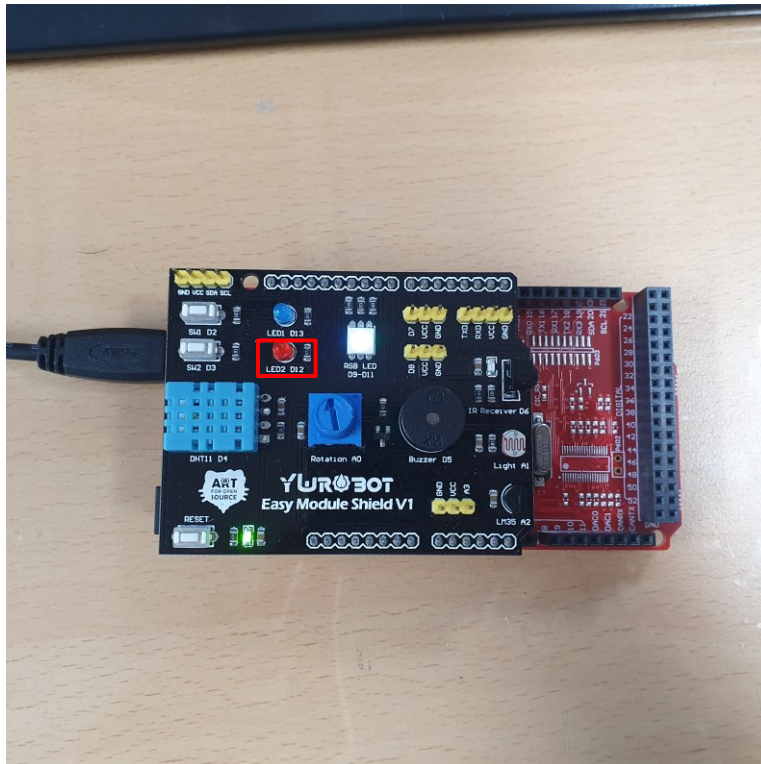


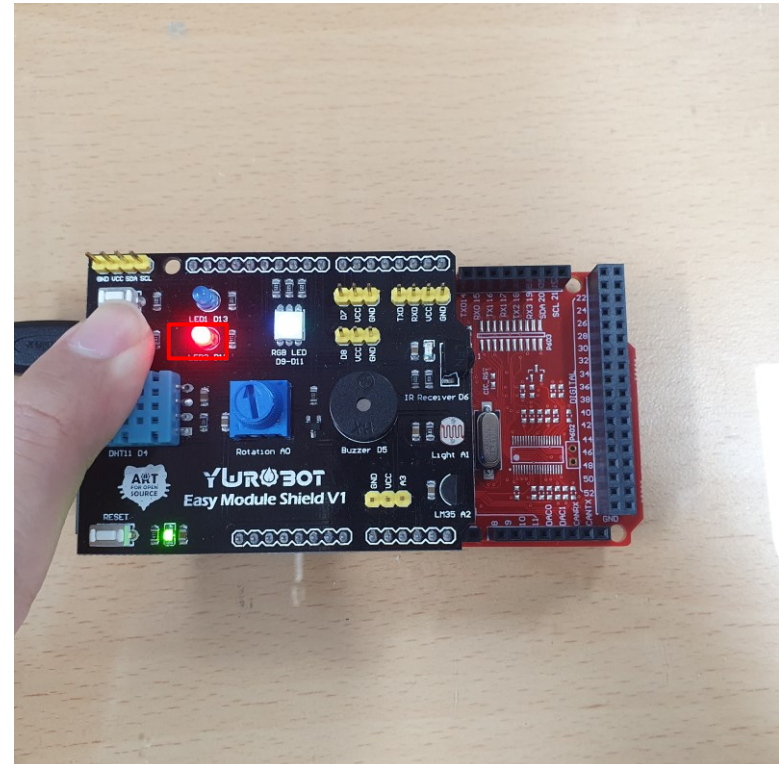
Figure 13-1 General Structure of a Port Pin

동작 확인

- Build & Debug 후, Push 버튼을 누를 때만 LED RED가 켜지는 모습 확인 가능

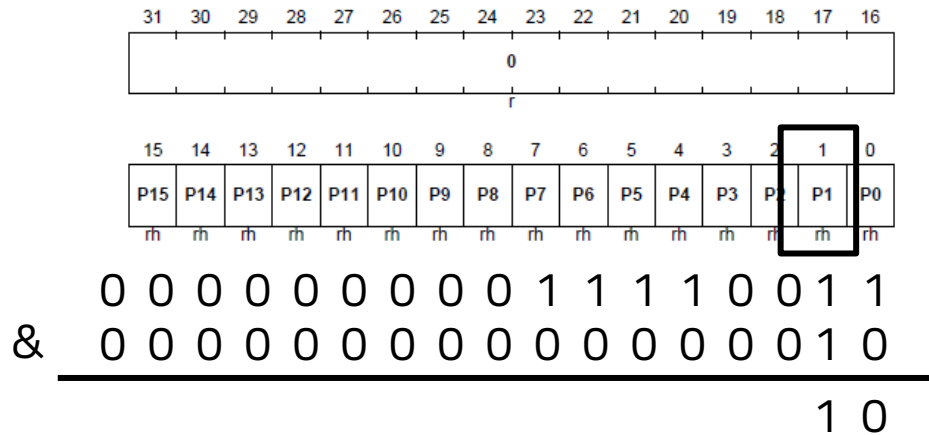


Switch is not pushed



Switch is pushed

비트 연산 후 0을 비교하는 이유



만약 P01_IN.U값이 0x0000_FF03일때
 $0x1 \ll P1_BIT_LSB_IDX$ 는 0x0000_0002이므로

⇒ 비트 연산결과가 1이 아니라 2임.

```
// 2. GPIO IN
if( (P02_IN.U & (0x1 << P1_BIT_LSB_IDX)) == 1 )
{
    P10_OUT.U &= ~(0x1 << P1_BIT_LSB_IDX);
}
else
{
    P10_OUT.U |= 0x1 << P1_BIT_LSB_IDX;
}
```

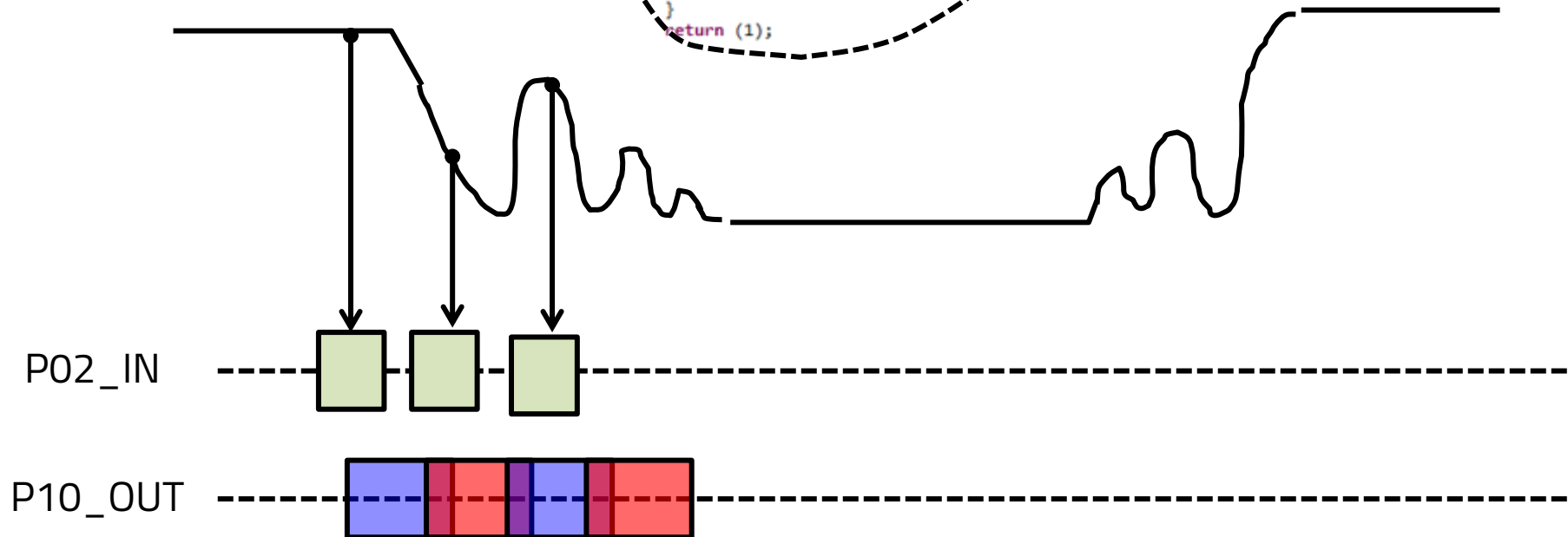
위치에 따라 2, 4, 8, 16
 등의 조건값을 비교해야
 하는 불편함

너무 빠른 Read Operation



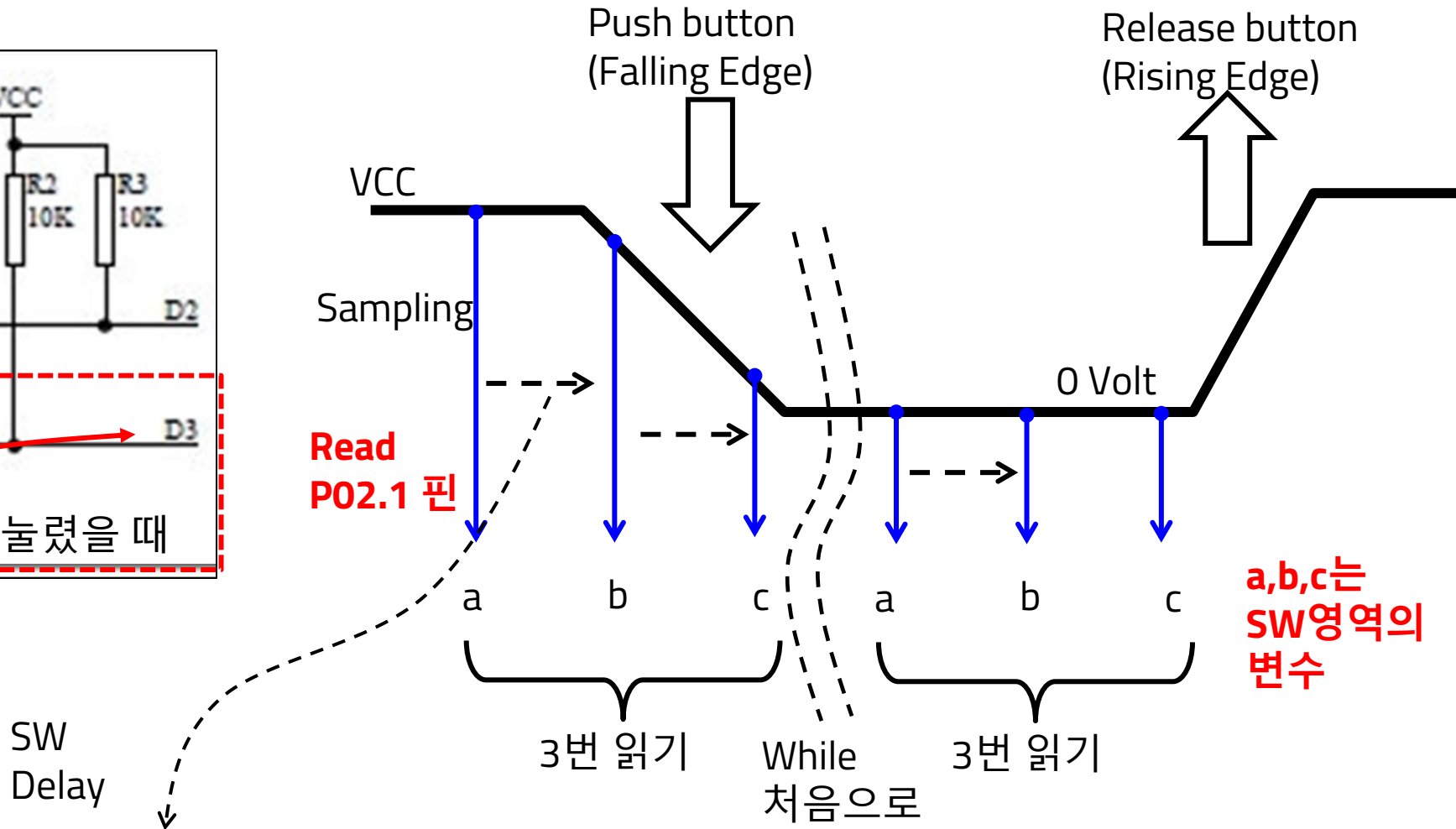
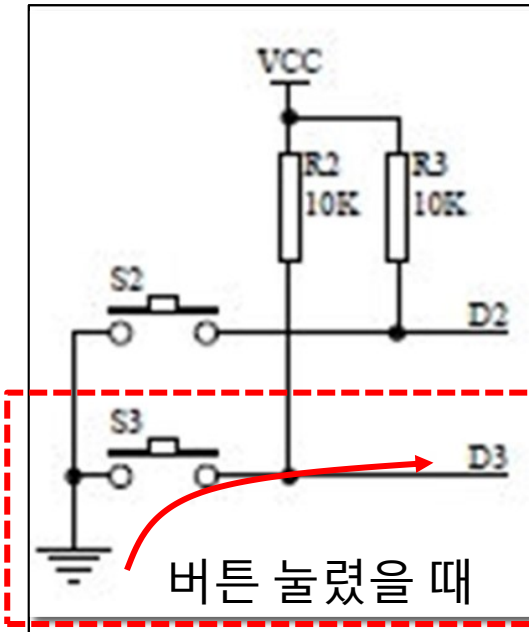
```
while(1)
{
    // 2. GPIO IN
    if( (P02_IN.U & (0x1 << P1_BIT_LSB_IDX)) == 0 )
    {
        P10_OUT.U |= 0x1 << P1_BIT_LSB_IDX;
    }
    else
    {
        P10_OUT.U &= ~(0x1 << P1_BIT_LSB_IDX);
    }
}
return (1);
```

빠른 cycle로 read



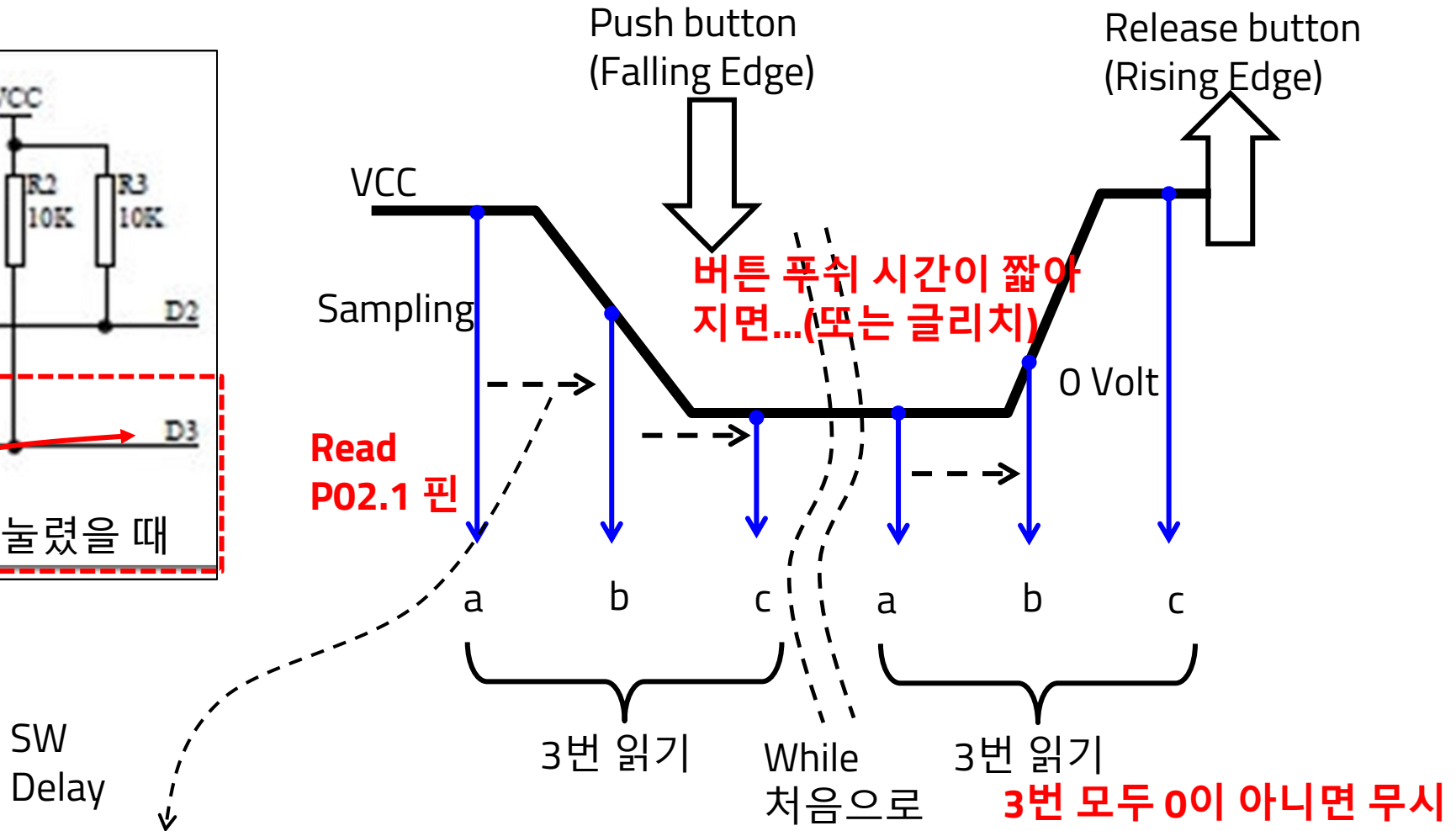
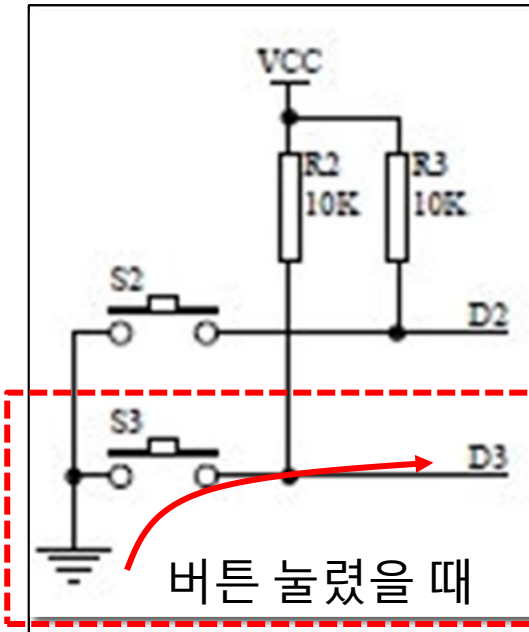
Overrun 됨 (아직 P10_OUT으로 값 전달하여 전압 구동 중에 다시, 다른 값을 씌)

Lab1: LPF – 샘플링 3번해서 유지되는지 판단



```
for(unsigned int i = 0; i < 100; i++);
```

Lab1: LPF – 샘플링 3번해서 유지되는지 판단



```
for(unsigned int i = 0; i < 100; i++);
```

Delay를 길게 하면 외부 버튼을 더 오랫동안 눌러주어야 버튼 인식될 것이다.

Lab1: LPF – 샘플링 3번해서 유지되는지 판단

```
while(1)
{

    a = P02_IN.U & (0x1 << P1_BIT_LSB_IDX);
    for(unsigned int i = 0; i < 100; i++);

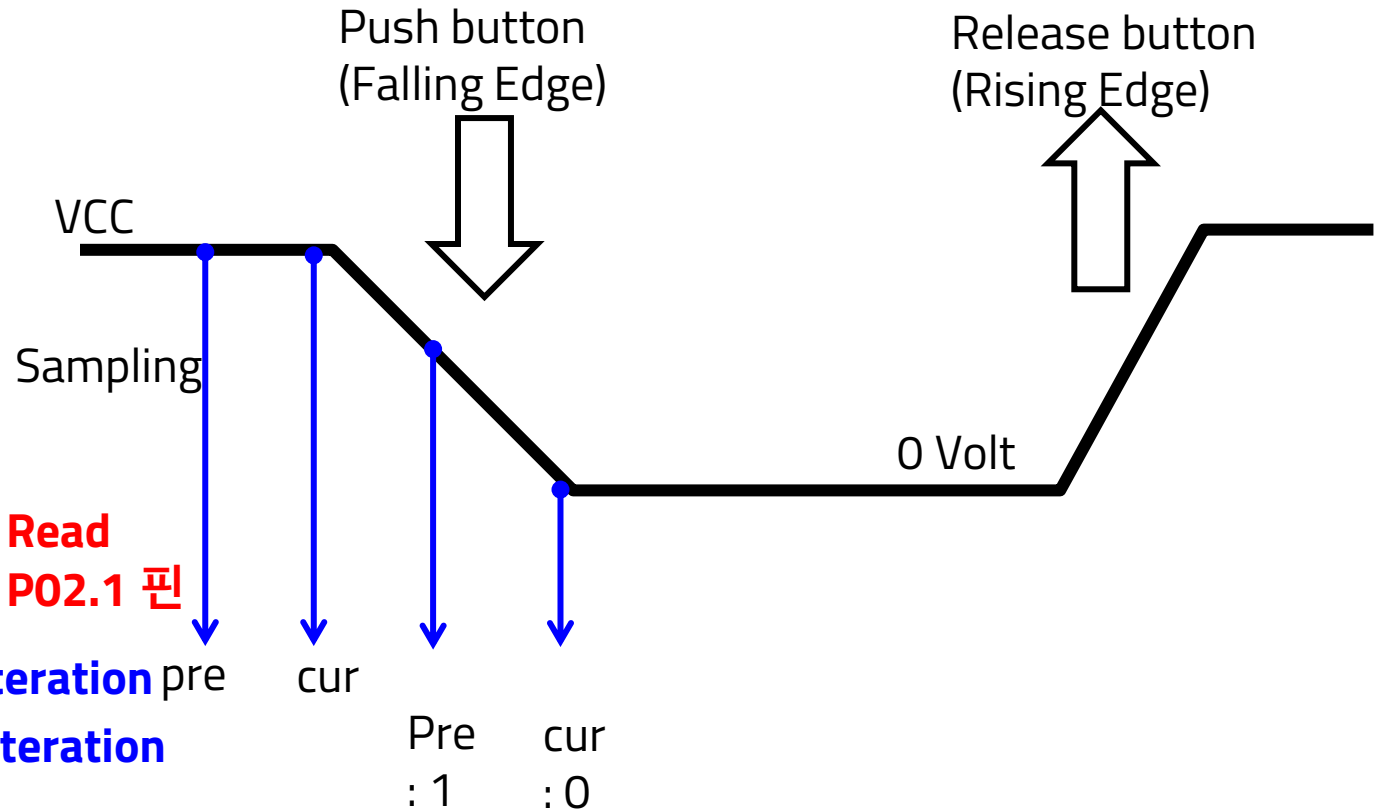
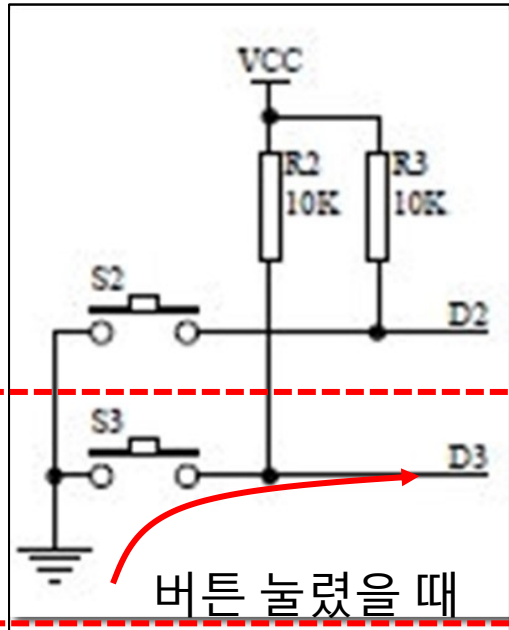
    b = P02_IN.U & (0x1 << P1_BIT_LSB_IDX);
    for(unsigned int i = 0; i < 100; i++);

    c = P02_IN.U & (0x1 << P1_BIT_LSB_IDX);
    for(unsigned int i = 0; i < 100; i++);

    if( a == 0 && b == 0 && c == 0 )
        P10_OUT.U |= (0x1 << P1_BIT_LSB_IDX);      // s
    else
        P10_OUT.U &= ~(0x1 << P1_BIT_LSB_IDX);      //

}
```


Lab2: Polling기반 Edge Detection



Pre cur 는 SW영역의 변수

Lab2: Polling기반 Edge Detection

```
unsigned char pre = 1, cur = 1;

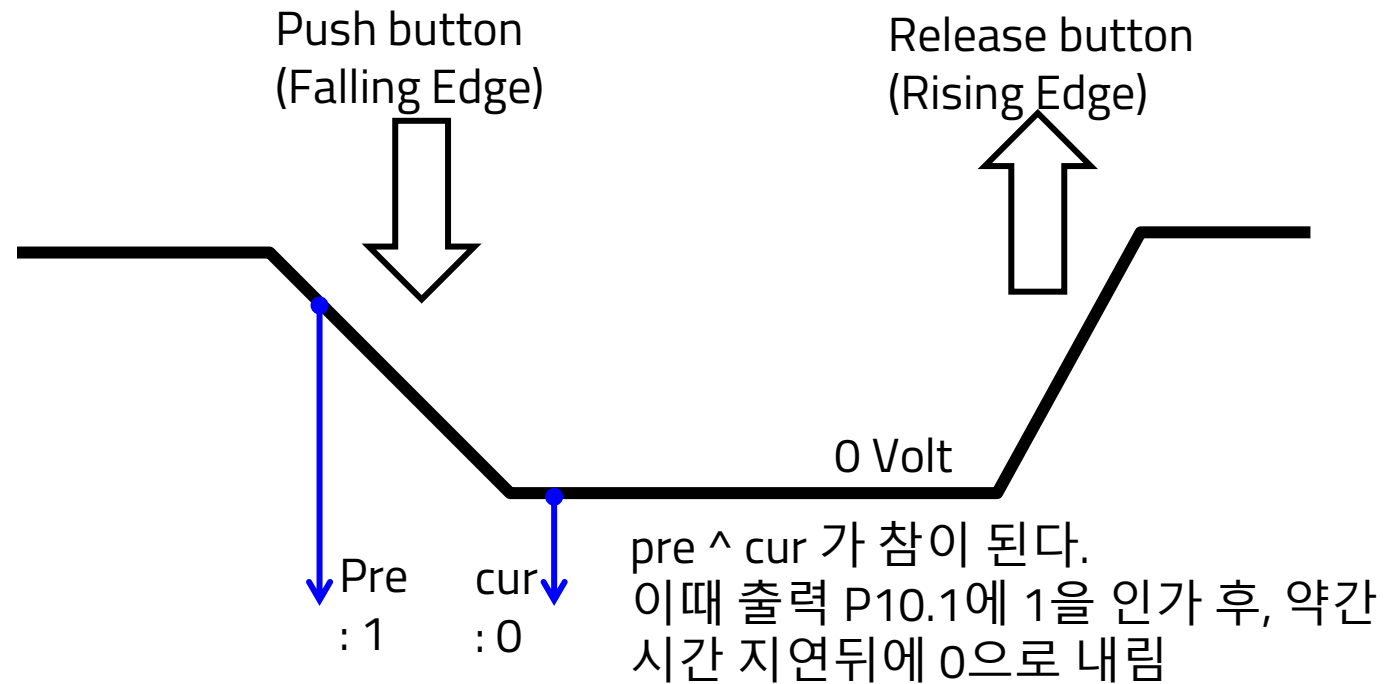
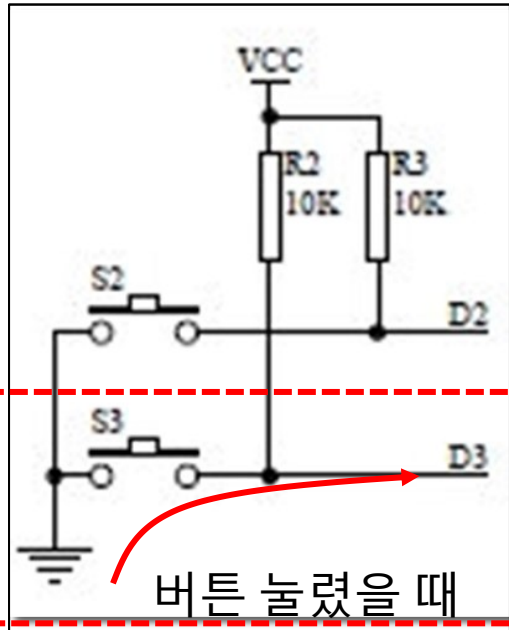
while(1)
{

    pre = P02_IN.U & (0x1 << P1_BIT_LSB_IDX);
    for(unsigned int i = 0; i < 1000; i++);
    cur = P02_IN.U & (0x1 << P1_BIT_LSB_IDX);

    // edge detection
    if( pre ^ cur )
    {
        P10_OUT.U |= (0x1 << P1_BIT_LSB_IDX);           // set P10.1 (LED D13 RED)
        for(unsigned int i = 0; i < 100000; i++);
        P10_OUT.U &= ~(0x1 << P1_BIT_LSB_IDX);           // clear P10.1 (LED D13 RED)
    }

}
```

Lab2: Polling기반 Edge Detection



P10.1포트 출력

SW Delay

```
for(unsigned int i = 0; i < 100000; i++);
```

감사합니다. 휴식~~

FAQ

- 상판 LED소자가 P10.1에 연결된 정보는 스펙 문서 2개 (타겟보드, MCU보드) 각각을 확인해야 한다
- 비트를 clear, set, toggle, 그리고 그 값을 확인 하는 4가지 비트 연산 방법을 꼭 기억하자!