TI DSP, MCU 및 Xilinx Zynq FPGA 프로그래밍 전문가 과정

2018-05-03 (47회차)

강사 - Innova Lee(이상훈)
gcccompil3r@gmail.com
학생 - 정유경
ucong@naver.com

TMS570LC4357 Hercules™ Microcontroller Based on the ARM® Cortex-R Core

[다운로드]

1. Code Composer Studio Version 8 Downloads

http://processors.wiki.ti.com/index.php/Download_CCS

Off-line Installers: Windows

: 설치 시 보안관련 설정을 모두 끈다

: Hercules Safety MCU 선택

: C:₩ti_tms570_workspace

2. HALCOGEN: HAL Code Generator tool

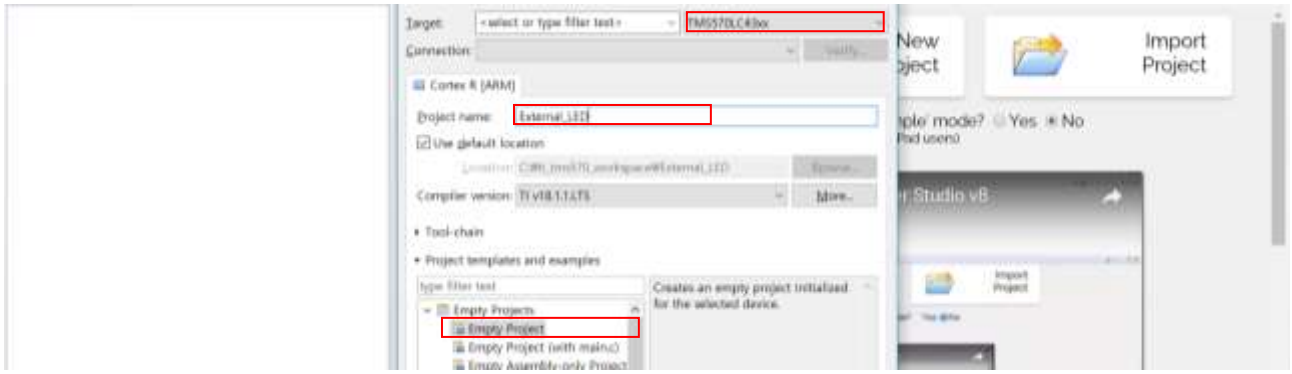
http://www.ti.com/tool/halcogen

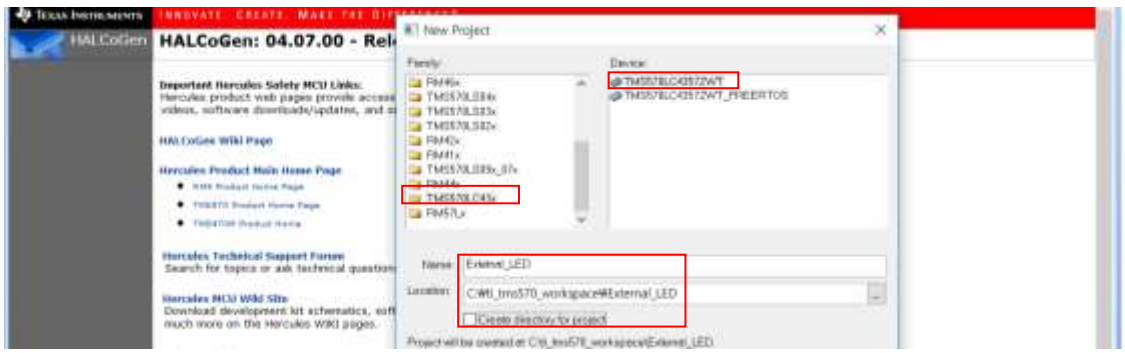
: component group the install 체크

[주의사항]

보드를 만지기 전에는 꼭 접지를 하자

[설치과정]

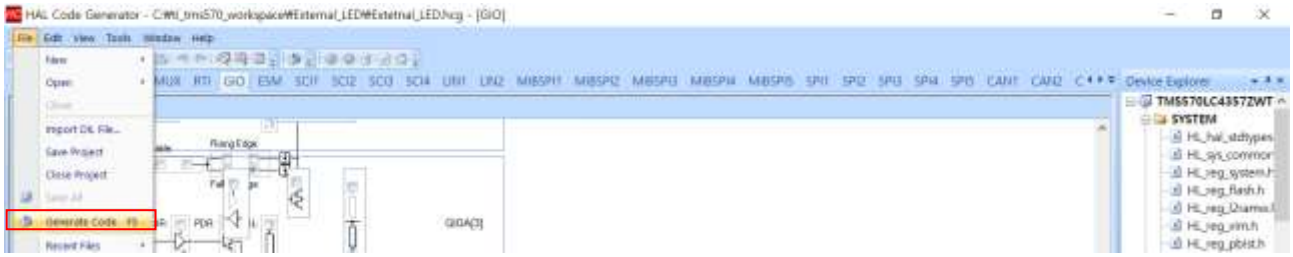


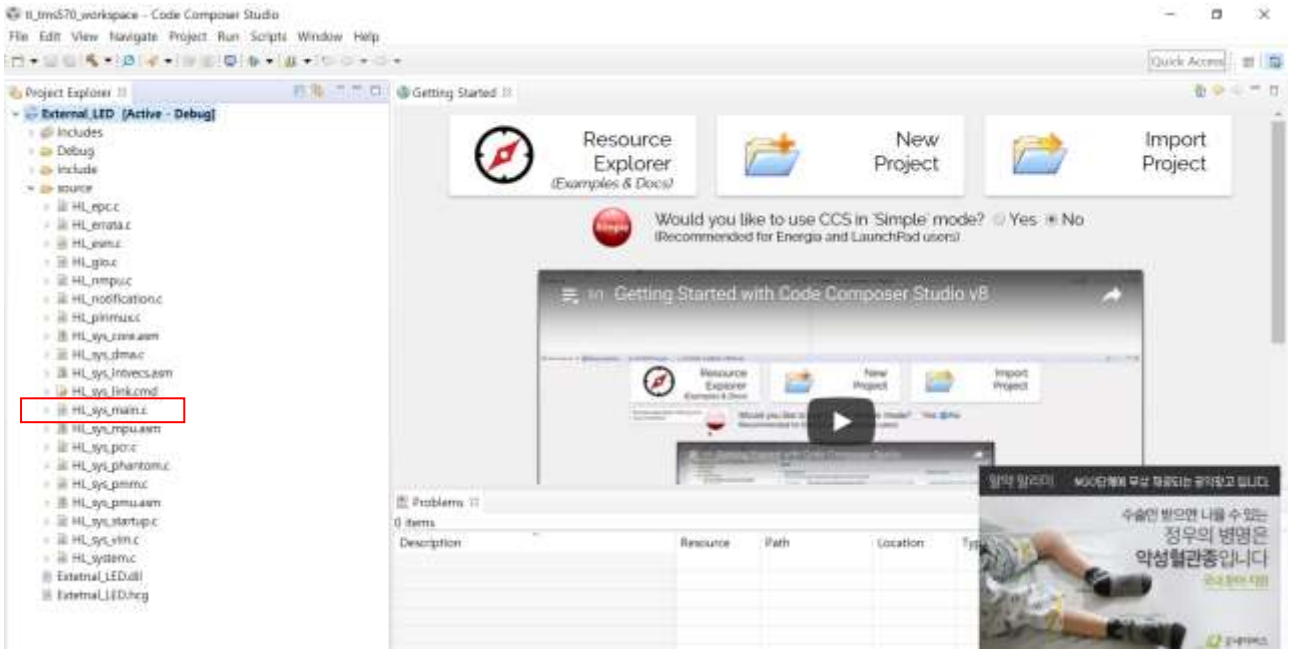


(Enable Driver Compilation : 사용가능한 모든 드라이버들을 나타낸다)

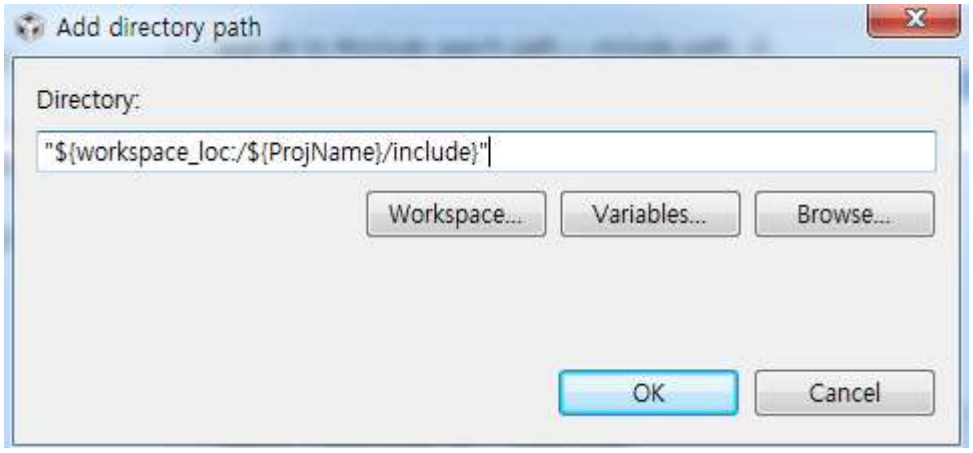
드라이버를 모두 끄고, LED 제어를 위해 GPIO 드라이버만 체크해준다

GIO 탭에서 우리가 사용할 PORTA > BIT4 > DIR(출력방향 설정) > PSL (풀업 설정)





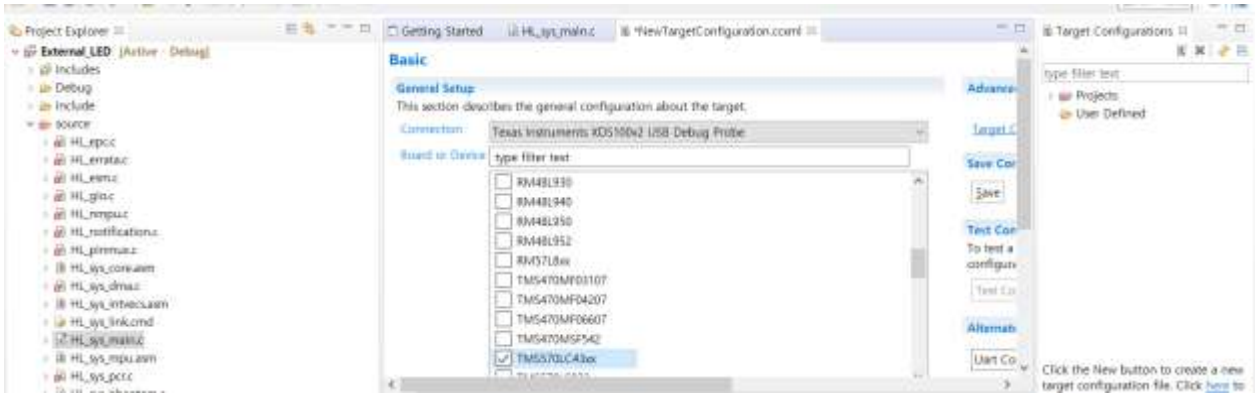
프로젝트에 추가된 Source Code 들 중 HL_sys_main.c 에서 프로그램 코드를 작성한다.



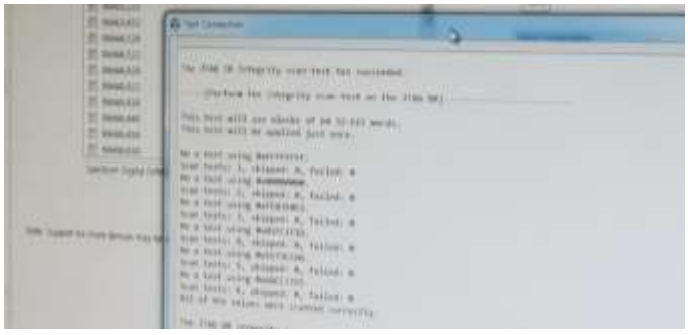
Property > Build > ARM Compiler > "${workspace_loc:/${ProjName}/include}" 입력

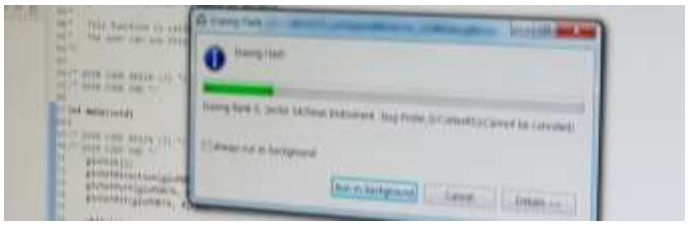
<끝>

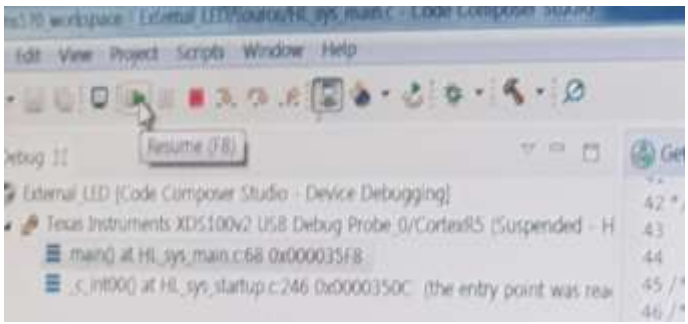
[HL_sys_main.c] 코드 작성하고 컴파일하여 실행하기



Debug > New Target Configuration 설정







[코드분석]

*. 들어갈땐 f3 나올땐 alt+방향키<-

[1] LED On/Off

```
int main(void)
{
/* USER CODE BEGIN (3) */
    int i;
    gioREG->GCR0 = 0x1; // GPIO가 정상적으로 동작한다
    gioREG->ENACLR = 0xFF; // 인터럽트를 수용하지 않는다
    gioREG->LVLCLR = 0xFF; // 저수준 인터럽트를 설정한다
```

```
    gioPORTA->DIR = 0x10; // 00010000 4번 비트를 출력으로 설정한다

    while(1)
    {
        gioPORTA->DOUT = 0x10; // 00010000 4번 비트 출력을 1로 설정한다
        for(i=0 ; i < 10000000 ; i++); // 정확한 값을 계산할 수없으므로 LED동작을 보면서 조절한다
        // 정확히 조절하고 싶으면 인터럽트를 사용해야 한다.

        gioPORTA->DOUT = 0x00; // 00010000 4번 비트 출력을 1로 설정한다
        for(i=0 ; i < 10000000 ; i++);

    }

/* USER CODE END */
    return 0;
}
```

[2] 최적화 한 예제

```
int main(void)
{
/* USER CODE BEGIN (3) */
    gioREG->GCR0 = 0x1; // GPIO가 정상적으로 동작한다
    gioREG->ENACLR = 0xFF; // 인터럽트를 수용하지 않는다
    gioREG->LVLCLR = 0xFF; // 저수준 인터럽트를 설정한다

    gioPORTA->DIR = 0x10; // 00010000 4번 비트를 출력으로 설정한다
    gioPORTA->DOUT = 0x10; // 00010000 4번 비트 출력을 1로 설정한다

    // gioSetBit(gioPORTA, 4, 1); 최적화시 필요없는 코드
        while(1);
/* USER CODE END */
    return 0;
}
```

[3] 기본예제 분석하기: GPIO PORTA 의 4 번 pin 에 LED 를 연결하고 LED ON 시킨다.

```
#include "HL_sys_common.h"
#include "HL_gio.h"

int main(void)
{
/* USER CODE BEGIN (3) */
    gioInit();
    gioSetDirection(gioPORTA, 0xffffffff);
    gioSetPort(gioPORTA, 0xffffffff);
    gioSetBit(gioPORTA, 4, 1);

        while(1);
/* USER CODE END */
    return 0;
}
```

1. gioInit() 안으로 들어간다

```
Void gioinit(void)
{
    /** bring GIO module out of reset */
    gioREG->GCR0    = 1U; // GPIO가 정상적으로 동작한다
```

TMS570LC4357 Hercules MCU Technical Reference Manual 을 본다.

Table 25-2. GIO Global Control Register (GIOGCR0) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reads return 0. Writes have no effect.
0	RESET		GIO reset.
		0	The GIO is in reset state.
		1	The GIO is operating normally.

gioREG->ENACLR = 0xFFU; // 인터럽트를 수용하지 않는다

25.5.4.2 GIOENACLR Register

This register disables the interrupt. Figure 25-9 and Table 25-6 describe this register.

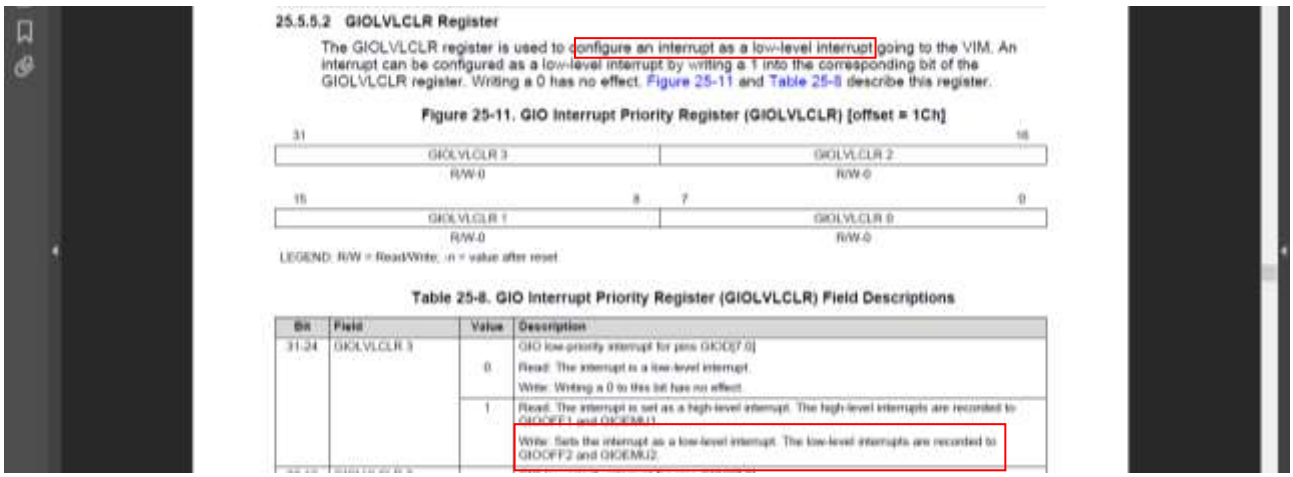
Figure 25-9. GIO Interrupt Enable Clear Register (GIOENACLR) [offset = 14h]

31	24	23	16
GIOENACLR 3		GIOENACLR 2	
R/W-0		R/W-0	

15	8	7	0
GIOENACLR 1		GIOENACLR 0	
R/W-0		R/W-0	

gioREG->LVLCLR = 0xFFU; // 저수준 인터럽트

(Low level 인터럽트가 발생하도록 설정해놓았다면 핀이 로우 레벨을 계속 유지하는 동안 연속적으로 인터럽트가 발생한다)



25.5.5.2 GIOLVLCLR Register

The GIOLVLCLR register is used to configure an interrupt as a low-level interrupt going to the VIM. An interrupt can be configured as a low-level interrupt by writing a 1 into the corresponding bit of the GIOLVLCLR register. Writing a 0 has no effect. Figure 25-11 and Table 25-8 describe this register.

Figure 25-11. GIO Interrupt Priority Register (GIOLVLCLR) [offset = 1Ch]

31	16
GIOLVLCLR 3	GIOLVLCLR 2
R/W-0	R/W-0
15	8 7 0
GIOLVLCLR 1	GIOLVLCLR 0
R/W-0	R/W-0

LEGEND: R/W = Read/Write; -n = value after reset

Table 25-8. GIO Interrupt Priority Register (GIOLVLCLR) Field Descriptions

Bit	Field	Value	Description
31-24	GIOLVLCLR 3		GIO low-priority interrupt for pins GIOD[7:0]
		0	Read: The interrupt is a low-level interrupt. Write: Writing a 0 to this bit has no effect.
		1	Read: The interrupt is set as a high-level interrupt. The high-level interrupts are recorded to GIOOFF1 and GIOEMU1. Write: Sets the interrupt as a low-level interrupt. The low-level interrupts are recorded to GIOOFF2 and GIOEMU2.

```
/** - Port A output values */
    gioPORTA->DOUT = (uint32)((uint32)0U << 0U)  /* Bit 0 */
                   | (uint32)((uint32)0U << 1U)  /* Bit 1 */
                   | (uint32)((uint32)0U << 2U)  /* Bit 2 */
                   | (uint32)((uint32)0U << 3U)  /* Bit 3 */
                   | (uint32)((uint32)0U << 4U)  /* Bit 4 */
                   | (uint32)((uint32)0U << 5U)  /* Bit 5 */
                   | (uint32)((uint32)0U << 6U)  /* Bit 6 */
                   | (uint32)((uint32)0U << 7U); /* Bit 7 */
```

GIO Data Output Registers (GIODOUT[A-B])

Values in the GIODOUT register specify the output state (high = 1 or low = 0) of the pins of the port

- GIO Data Output Registers 를 모두 0으로 셋팅한다, 처음에는 PORTA의 출력이 없다.

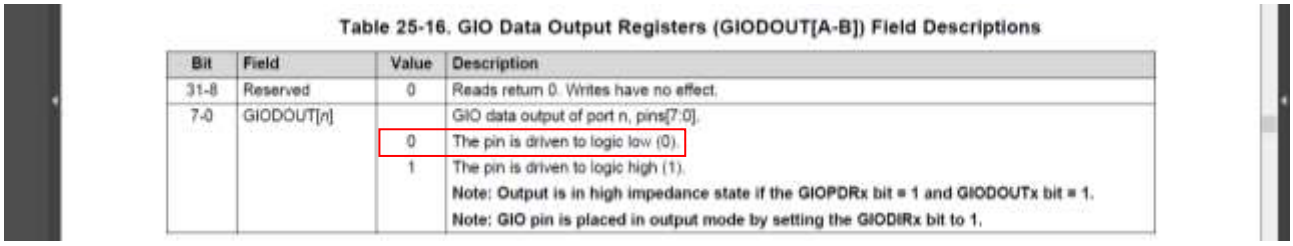


Table 25-16. GIO Data Output Registers (GIODOUT[A-B]) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reads return 0. Writes have no effect.
7-0	GIODOUT[n]		GIO data output of port n, pins[7:0].
		0	The pin is driven to logic low (0).
		1	The pin is driven to logic high (1). Note: Output is in high impedance state if the GIOPDRx bit = 1 and GIODOUTx bit = 1. Note: GIO pin is placed in output mode by setting the GIODIRx bit to 1.

```
    /** - Port A direction */
    gioPORTA->DIR  = (uint32)((uint32)0U << 0U)  /* Bit 0 */
                   | (uint32)((uint32)0U << 1U)  /* Bit 1 */
                   | (uint32)((uint32)0U << 2U)  /* Bit 2 */
                   | (uint32)((uint32)0U << 3U)  /* Bit 3 */
                   | (uint32)((uint32)1U << 4U)  /* Bit 4 */ // halcogen 에서 설정함
                   | (uint32)((uint32)0U << 5U)  /* Bit 5 */
                   | (uint32)((uint32)0U << 6U)  /* Bit 6 */
                   | (uint32)((uint32)0U << 7U); /* Bit 7 */
```

PORTA 의 4번 비트를 출력으로 설정한다.

(나머지 비트는 입력)

Table 25-14. GIO Data Direction Registers (GIODIR[A-B]) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reads return 0. Writes have no effect.
7-0	GIODIR[n]		GIO data direction of port n, pins [7:0]
		0	The GIO pin is an input. Note: If the pin direction is set as an input, the output buffer is tristated.
		1	The GIO pin is an output.

```
/** - Port A open drain enable */
gioPORTA->PDR  = (uint32)((uint32)0U << 0U)  /* Bit 0 */
               | (uint32)((uint32)0U << 1U)  /* Bit 1 */
               | (uint32)((uint32)0U << 2U)  /* Bit 2 */
               | (uint32)((uint32)0U << 3U)  /* Bit 3 */
               | (uint32)((uint32)0U << 4U)  /* Bit 4 */
               | (uint32)((uint32)0U << 5U)  /* Bit 5 */
               | (uint32)((uint32)0U << 6U)  /* Bit 6 */
               | (uint32)((uint32)0U << 7U); /* Bit 7 */
```

Table 25-19. GIO Open Drain Registers (GIOPDR[A-B]) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reads return 0. Writes have no effect.
7-0	GIOPDR[n]		GIO open drain for port n, pins[7:0]
		0	The GIO pin is configured in push/pull (normal GIO) mode. The output voltage is V_{OL} or lower if GIODOUT bit = 0 and V_{OH} or higher if GIODOUT bit = 1.
		1	The GIO pin is configured in open drain mode. The GIODOUTx bit controls the state of the GIO output buffer: GIODOUTx = 0, the GIO output buffer is driven low; GIODOUTx = 1, the GIO output buffer is tristated.

***. Open Collector**

: Open-collector(오픈콜렉터)/open-drain(오픈드레인)은 여러개의 장치(Device)를 하나의 연결선으로 양방향(bi-directionally) 통신할 수 있도록 하는 Circuit 테크닉

(http://egloos.zum.com/slgi97/v/10874166)

```
/** - Port A pullup / pulldown selection */
gioPORTA->PSL  = (uint32)((uint32)0U << 0U)  /* Bit 0 */
               | (uint32)((uint32)0U << 1U)  /* Bit 1 */
               | (uint32)((uint32)0U << 2U)  /* Bit 2 */
               | (uint32)((uint32)0U << 3U)  /* Bit 3 */
               | (uint32)((uint32)1U << 4U)  /* Bit 4 */ // halcogen 에서 설정함
               | (uint32)((uint32)0U << 5U)  /* Bit 5 */
               | (uint32)((uint32)0U << 6U)  /* Bit 6 */
               | (uint32)((uint32)0U << 7U); /* Bit 7 */
```

Table 25-21. GIO Pull Select Registers (GIOPSL[A-B]) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reads return 0. Writes have no effect.
7-0	GIOPSL[n]		GIO pull select for port n, pins[7:0]
		0	The pull down functionality is select, when pull up/pull down logic is enabled.
		1	The pull up functionality is select, when pull up/pull down logic is enabled.
			Note: The pull up/pull down functionality is enabled by clearing corresponding bit in GIOPULDIS to 0.

***. 풀업, 풀다운저항**

: 스위치가 OFF 될 때, floating 되어 있는 입력 상태를 없애는 용도

```
/** - Port A pullup / pulldown enable*/
gioPORTA->PULDIS  = (uint32)((uint32)0U << 0U)  /* Bit 0 */
                  | (uint32)((uint32)0U << 1U)  /* Bit 1 */
                  | (uint32)((uint32)0U << 2U)  /* Bit 2 */
                  | (uint32)((uint32)0U << 3U)  /* Bit 3 */
                  | (uint32)((uint32)0U << 4U)  /* Bit 4 */
                  | (uint32)((uint32)0U << 5U)  /* Bit 5 */
                  | (uint32)((uint32)0U << 6U)  /* Bit 6 */
                  | (uint32)((uint32)0U << 7U); /* Bit 7 */
```

Table 25-20. GIO Pull Disable Registers (GIOPULDIS[A-B]) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reads return 0. Writes have no effect.
7-0	GIOPULDIS[n]		GIO pull disable for port n, pins[7:0]. Writes to this bit will only take effect when the GIO pin configured as an input pin.
		0	The pull functionality is enabled.
		1	The pull functionality is disabled.
			Note: The GIO pin is placed in input mode by clearing the GIODIRx bit to 0.

2. gioSetDirection(gioPORTA, 0xffffffff)

port->DIR = dir;

GIO Data Direction Registers (GIODIR[A-B])

The GIODIR register controls whether the pins of a given port are configured as inputs or outputs.

PORTA 의 모든 비트를 출력으로 설정한다.

1번에서 4번 비트를 이미 출력으로 설정해 놓았는데도, 여기서 재설정해주는 이유는

gioInit()으로 직접 들어가지 않고도 커스텀이 가능하기 때문이다.

gioInit()에서 필요없었던 코드들을 작성할 필요가 없다.

3. gioSetPort(gioPORTA, 0xffffffff) // 모든 비트 출력을 1 로 설정

port->DOUT = value;

GIO Data Output Registers (GIODOUT[A-B])

Values in the GIODOUT register specify the output state (high = 1 or low = 0) of the pins of the port when

they are configured as outputs

4. gioSetBit(gioPORTA, 4, 1) // 4 번 비트 출력만 1 로 설정

```
void gioSetBit(gioPORT_t *port, uint32 bit, uint32 value)
{
/* USER CODE BEGIN (5) */
/* USER CODE END */

    if (value != 0U)
    {
        port->DSET = (uint32)1U << bit;
```

```
    }
    else
    {
        port->DCLR = (uint32)1U << bit;
    }
}
```

GIO Data Set Registers (GIODSET[A-B])

Values in this register set the data output control register bits to 1 regardless of the current value in the GIODOUT bits.

GIO Data Clear Registers (GIODCLR[A-B])

Values in this register clear the data output register (GIO Data Output Register [A-H]) bit to 0 regardless of its current value.