TI DSP,MCU 및 Xilinux Zynq FPGA

프로그래밍 전문가 과정

이름	문지희
학생 이메일	mjh8127@naver.com
날짜	2018/5/3
수업일수	47 일차
담당강사	Innova Lee(이상훈)
강사 이메일	gcccompil3r@gmail.com

목차

보드 사용하기

[Code Composer Studio Version 8] 다운로드 링크

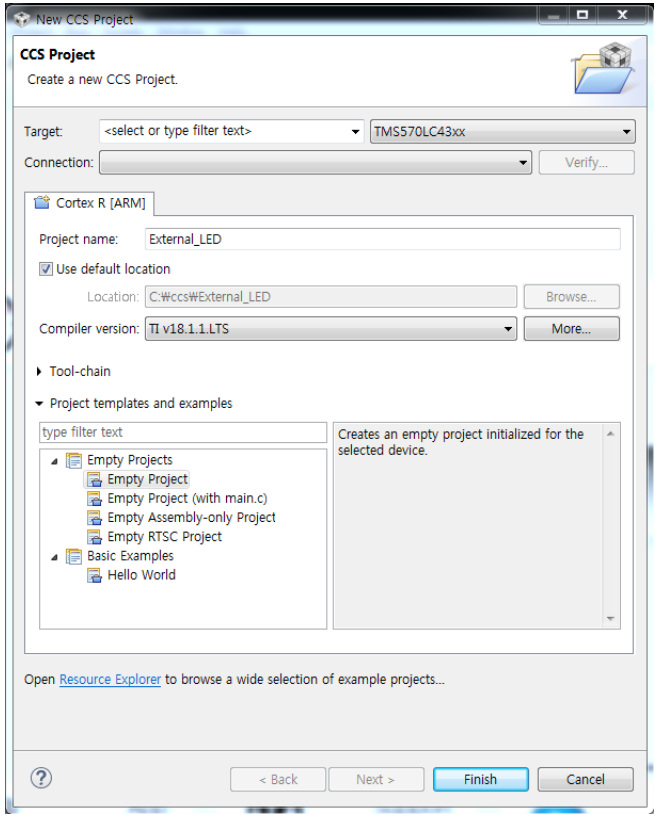
http://processors.wiki.ti.com/index.php/Download_CCS

Off-line Installers 의 Windows 를 클릭하고 다운 받음.

[HALcogen] 다운로드 링크

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

- 프로젝트 생성 (CSS)



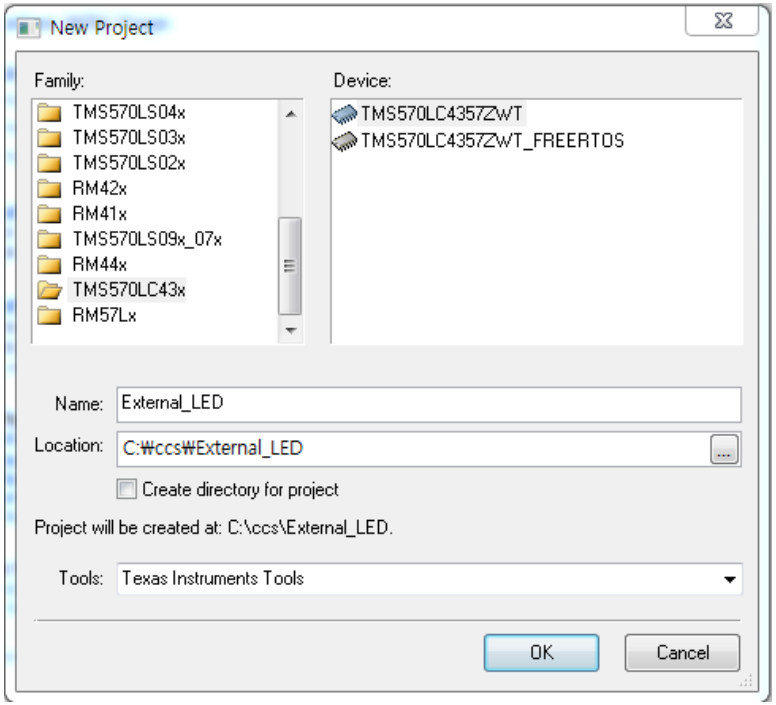
file - new - CSSproj

TMS570LC43xx 로 타겟을 바꾸고

project name 을 External_LED 로 설정한다.

Empty Project 를 클릭하고 Finish

-프로젝트 생성 (HAL)

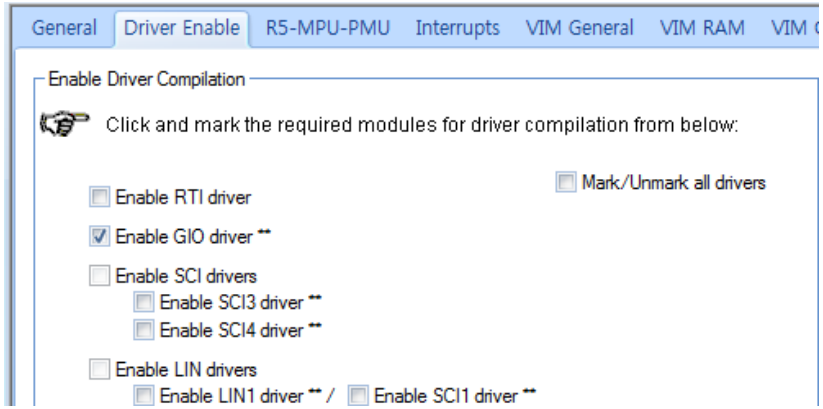


TMS570LC43x - TMS570LC4357ZwT 를 누름

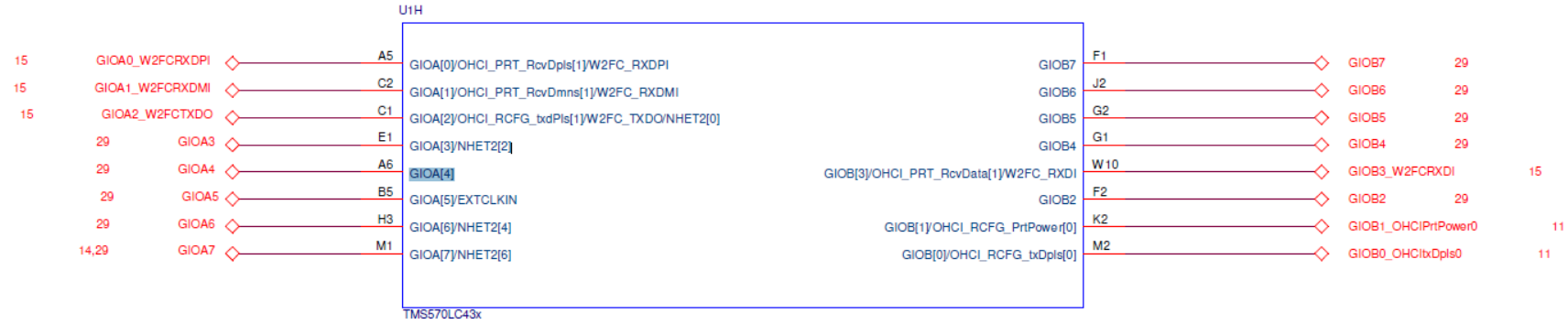
External_LED 로 이름 설정

Create directory for project 해제 뒤 OK

생성된 프로젝트에서 Driver Enable 클릭

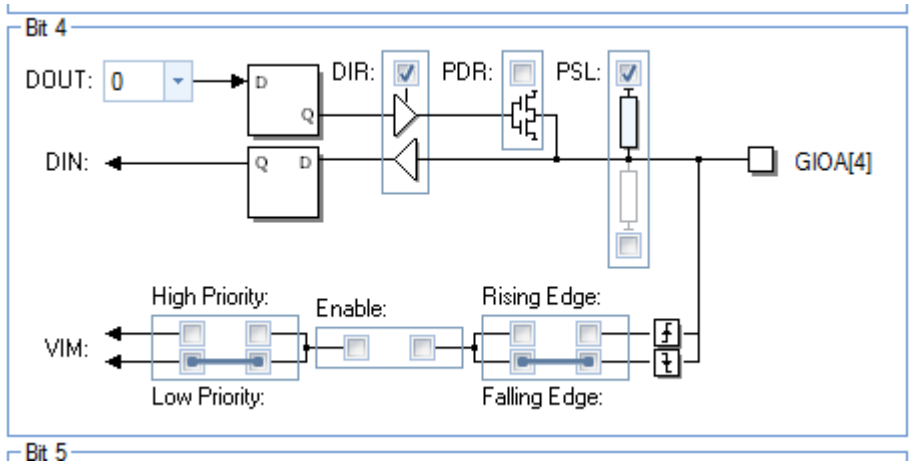


LED 를 제어할 것 이므로 GIO driver 외에는 필요 없으니 Mark/Unmark all drivers 를 클릭하여 모두 해제하고 Enable GIO driver 만 클릭



위의 그림은 우리가 사용하는 보드의 GIO 핀들인데, 살펴보면 GIO 동작만 하는 GIOA[4]를 사용할 것임.

GOIA[4] 핀설정



GIO 에서 GIOA[4]핀인 Bit4 에서 DIR 과 PSL 을 클릭한다.

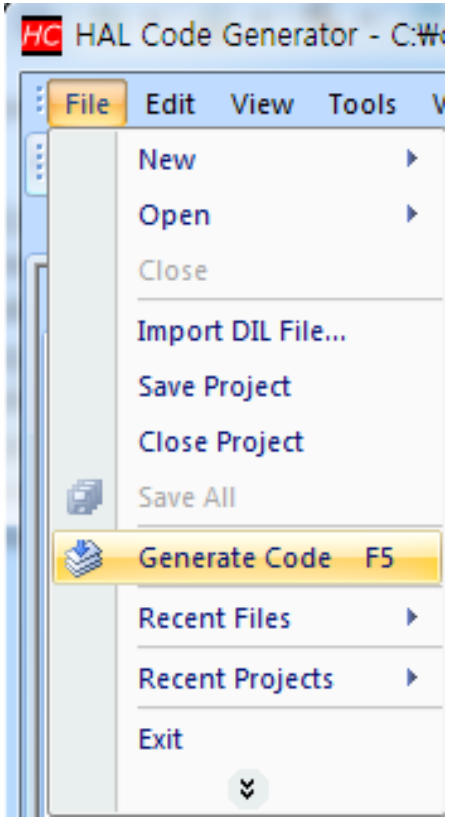
DIR(Data Direction Registers) - DOUT 의 방향을 정해줌

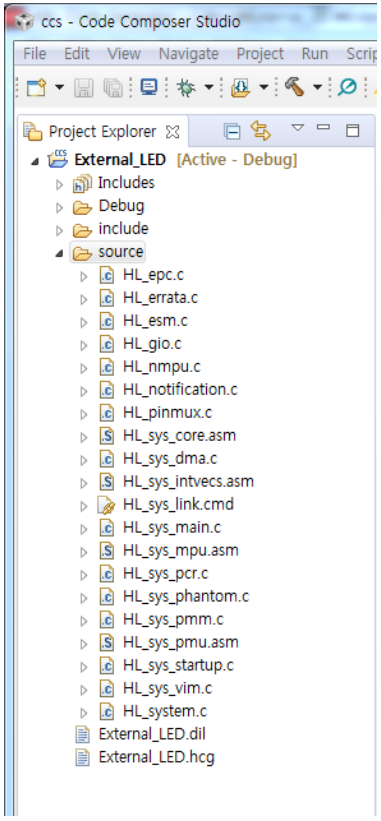
PSL(Pull Select Registers) - 플로팅 상태를 해결하기 위함. 위로 올리면 pull up, 아래로 내리면 pull down 이다.

(플로팅 상태 : HIGH 와 LOW 사이의 값을 가져 이도 저도 아닌 상태.

pull up : 플로팅 상태일 때 HIGH 로 끌어올려 줌

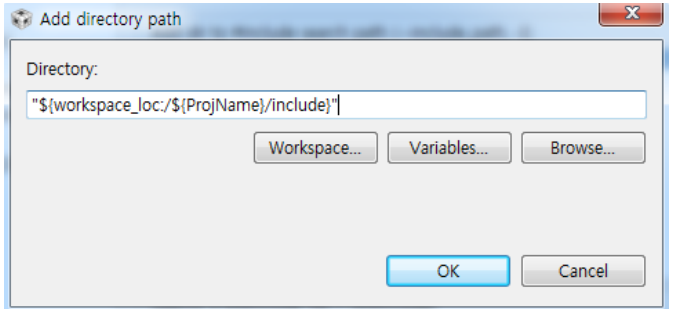
pull down : 플로팅 상태일 때 LOW 로 내려 줌)





핀 설정을 만지고 나서 file - Generate Code 를 누르면 External_LED 폴더에 source 와 include 가 생기고 c 파일과 헤더파일이 자동으로 만들어짐.

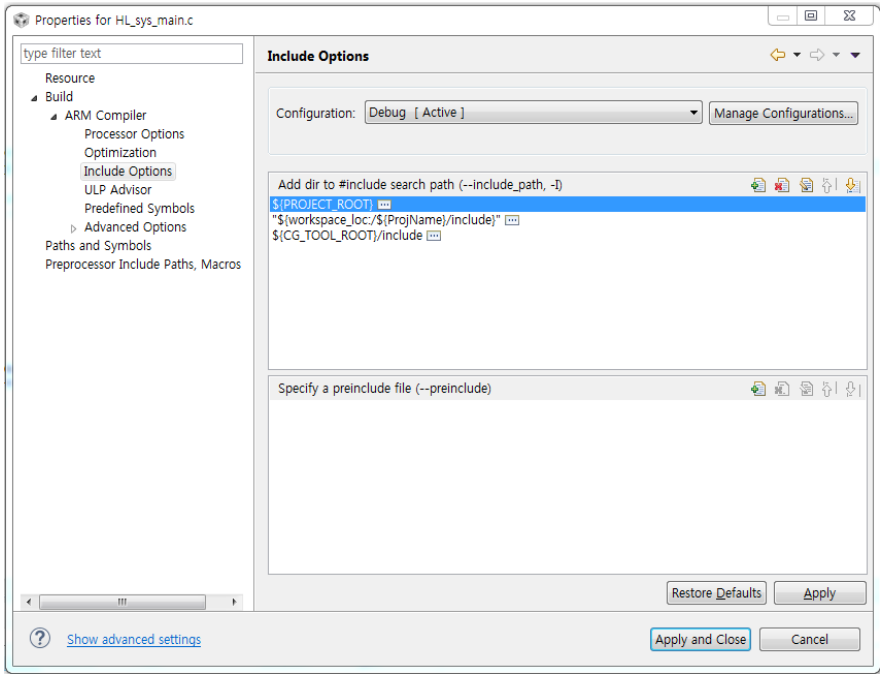
c 파일과 h 파일을 사용하기 위해 설정해야 하는 것이 있음



Alt + Enter(?)를 눌러 프로퍼티를 열고 Build-ARM Compiler - Include Options 에서

Add dir to #incldue search path 를 추가한다.

“${workspace_loc:/${ProjName}/include}”를 입력하고 OK.



-회로 구성

GIOA4 핀 - 저항 - LED - GND 핀으로 회로를 직렬 연결하여 회로를 구성함.

-소스코드 작성

project Explorer - External_LED - source - HL_main.c 을 눌러 아래 소스코드를 작성한다.

이후 망치 그림을 클릭하여 디버깅함.

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

int main(void)
{

    gioInit();
    gioSetDirection(gioPORTA, 0xffffffff);
    gioSetPort(gioPORTA, 0xffffffff);
    gioSetBit(gioPORTA, 4, 1);

    while(1);

    return 0;
}
```

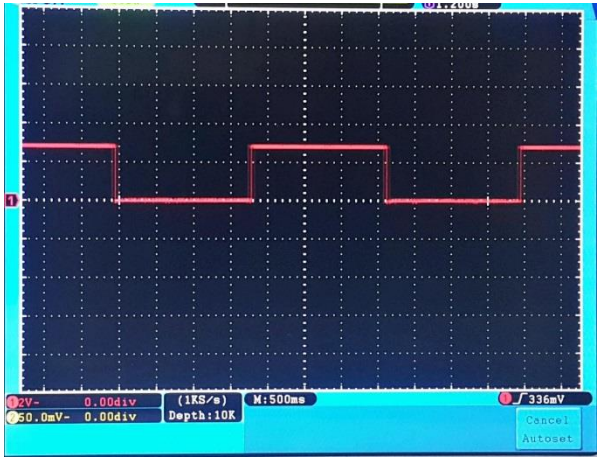
-결과

-LED 깜빡거리게 하기

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

int main(void)
{
    int i;
    gioInit();
    gioSetDirection(gioPORTA, 0xffffffff);

    while(1)
    {
        gioSetPort(gioPORTA, 0xffffffff);
        for(i=0; i<100000000; i++)
        gioSetPort(gioPORTA, 0x00000000);
        for(i=0; i<100000000; i++)
    }
    return 0;
}
```



오실로스코프로 회로의 양 끝에 프로브로 전압을 측정한 모습. 3V 에 못 미치는 출력파형이 나오게 됨.

약 3V, 0V 를 왔다갔다하면서 LED 에 신호를 보내고 있음.

데이트 시트 차원에서 GIO 분석하기

- gioInit (1)

```
void gioInit(void)
{
/* USER CODE BEGIN (2) */
/* USER CODE END */

    /** bring GIO module out of reset */
    gioREG->GCR0    = 1U;
    gioREG->ENACLR = 0xFFU;
    gioREG->LVLCLR = 0xFFU;

    /** @b initialize @b Port @b A */
```

gioREG->GCR0 = 1U; //GIO 가 정상적으로 동작함.

gioREG->ENACLR = 0xFFU; //Write 인터럽트를 수용하지 않음.

gioREG->LVLCLR = 0xFFU; //인터럽트를 low 레벨 인터럽트로 지정

[TMS570LC4357 Hercules MCU Technical Reference Manual - 25.5.1 GIO Global Control Register (GIOGCR0)]

Bit 1 에 모듈 리셋 상태를 제어하는 비트가 존재. 0 을 넣으면 모듈이 리셋되고, 1 은 GIO 가 정상 작동하고 있는 상태이다.

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.

[TMS570LC4357 Hercules MCU Technical Reference Manual - 25.5.4.2 GIOENACLR Register]

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	

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

Table 25-6. GIO Interrupt Enable Clear Register (GIOENACLR) Field Descriptions

Bit	Field	Value	Description
31-24	GIOENACLR 3		Interrupt disable for pins GIOD[7:0]
		0	Read: The interrupt is disabled. Write: Writing a 0 to this bit has no effect.
		1	Read: The interrupt is enabled. Write: Disables the interrupt.
23-16	GIOENACLR 2		Interrupt disable for pins GIOC[7:0]
		0	Read: The interrupt is disabled. Write: Writing a 0 to this bit has no effect.
		1	Read: The interrupt is enabled. Write: Disables the interrupt.
15-8	GIOENACLR 1		Interrupt disable for pins GIOB[7:0]
		0	Read: The interrupt is disabled. Write: Writing a 0 to this bit has no effect.
		1	Read: The interrupt is enabled. Write: Disables the interrupt.
7-0	GIOENACLR 0		Interrupt disable for pins GIOA[7:0]
		0	Read: The interrupt is disabled. Write: Writing a 0 to this bit has no effect.
		1	Read: The interrupt is enabled. Write: Disables the interrupt.

0xFFU 이므로 bit 0 ~ bit 31 까지

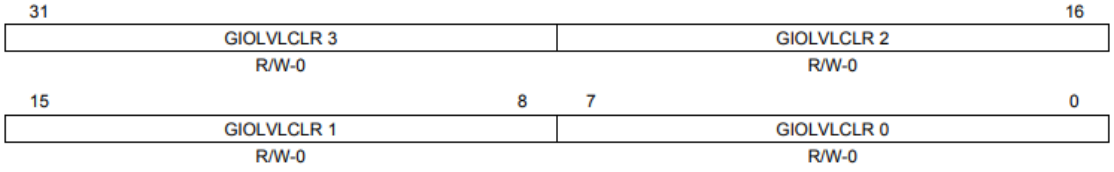
Read : 인터럽트 수신가능, Write : 인터럽트 수신 불가능,

[TMS570LC4357 Hercules MCU Technical Reference Manual - 25.5.5.2 GIOLVLCLR Register]

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]



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.
23-16	GIOLVLCLR 2		GIO low-priority interrupt for pins GIOC[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.
15-8	GIOLVLCLR 1		GIO low-priority interrupt for pins GIOB[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.
7-0	GIOLVLCLR 0		GIO low-priority interrupt for pins GIOA[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.

Read: 하이-레벨 인터럽트로 세팅되고 GIOOFF1 과 GIOEMU1 에 기록됨.

Write: 로우-레벨 인터럽트로 세팅되고 GIOOFF2 와 GIOEMU2 에 기록된다.

- gioInit (2)

```
/** - 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 */

/** - 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 */
               | (uint32)((uint32)0U << 5U)  /* Bit 5 */
               | (uint32)((uint32)0U << 6U)  /* Bit 6 */
               | (uint32)((uint32)0U << 7U); /* Bit 7 */

/** - 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 */

/** - 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 */

/** @b initialize @b Port @b B */

/** - Port B output values */
gioPORTB->DOUT = (uint32)((uint32)0U << 0U)  /* Bit 0 */
```

gioPORTA->DOUT 을 전부 0 으로 세팅

gioPORTA->DIR 의 bit 4 를 1 로 세팅하고 나머지는 전부 0 으로 세팅. 이게 GIOA[4]

gioPORTA->POR 을 전부 0 으로 세팅

gioPORTA->PULDIS 를 전부 0 으로 세팅

PORTB 는 사용하지 않음.

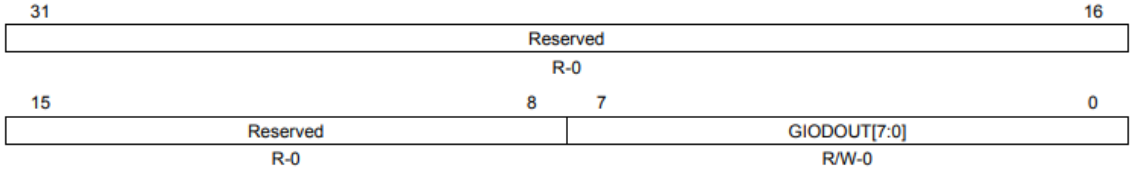
[TMS570LC4357 Hercules MCU Technical Reference Manual - 25.5.13 GIO Data Output Registers (GIODOUT[A-B])]

25.5.13 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. Figure 25-19 and Table 25-16 describe this register.

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

Figure 25-19. GIO Data Output Registers (GIODOUT[A-B]) [offset = 3Ch, 5Ch]



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

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.

GIODOUT 레지스터의 값들은 출력으로 구성될 때 포트 핀의 출력상태를 지정한다.

전부 0 으로 세팅했으므로 bit 31 ~ bit 8 은 읽을 때 0 을 반환하고 쓸 때는 아무런 동작을 하지 않음.

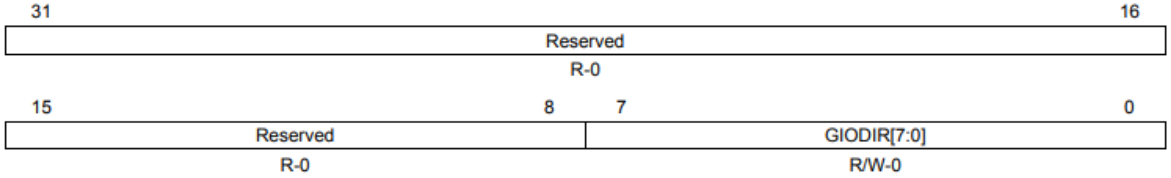
bit 7~ bit 0 은 0 일 때 low 로 구동되고 1 일 때 high 로 구동된다. 그래서 핀 0~핀 7 모두 0 으로 구동된다.

[TMS570LC4357 Hercules MCU Technical Reference Manual - 25.5.11 GIO Data Direction Registers (GIODIR[A-B])]

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

The GIODIR register controls whether the pins of a given port are configured as inputs or outputs. Figure 25-17 and Table 25-14 describe this register.

Figure 25-17. GIO Data Direction Registers (GIODIR[A-B]) [offset = 34h, 54h]



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

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.

GIODIR 레지스터는 주어진 포트 핀을 입력이나 출력으로 구성하도록 조정한다.

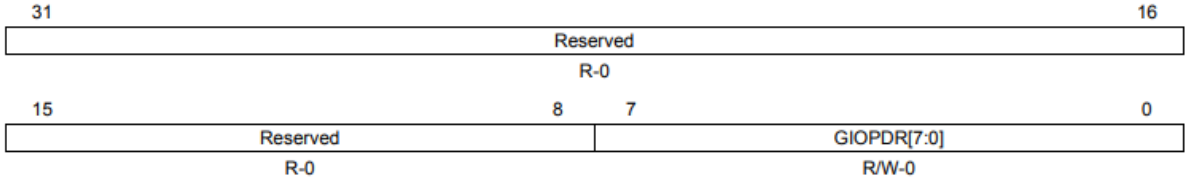
gioPORTA->DIR의 bit 4만 1로 세팅했으므로 핀 4가 출력 핀이 되고 나머지 핀들은 입력 핀이 된다.

[TMS570LC4357 Hercules MCU Technical Reference Manual - 25.5.16 GIO Open Drain Registers (GIOPDR[A-B])]

25.5.16 GIO Open Drain Registers (GIOPDR[A-B])

Values in this register enable or disable the open drain capability of the data pins. Figure 25-22 and Table 25-19 describe this register.

Figure 25-22. GIO Open Drain Registers (GIOPDR[A-B]) [offset = 48h, 68h]



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

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.

레지스터의 값들은 데이터 핀의 오픈 드레인 기능을 활성화 또는 비활성화 한다.

전부 0 으로 세팅.

0 : GIO 핀은 push 모드/pull 모드로 구성 되어있다. 출력전압은 GIODOUT 의 비트가 0 이면 V_{OL}보다 낮고 비트가 1 이면 V_{OL}보다 크다.

오픈 드레인(Open Drain)

: BJT 트랜지스터인 경우 컬렉터가 개방된 형태의 구조의 출력회로.

오픈 콜렉터(Open Collector)

: MOS FET 인 경우 드레인이 개방된 형태의 출력회로

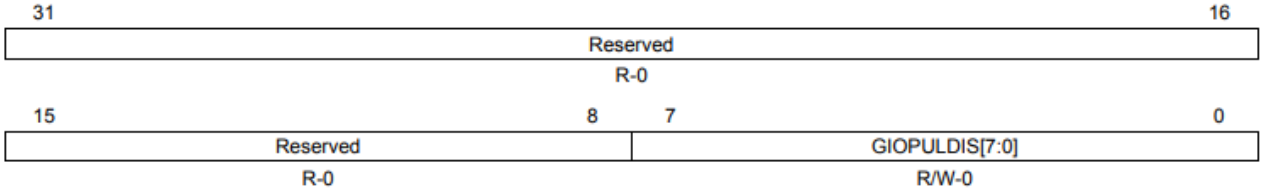
오픈 드레인이나 오픈 컬렉터의 출력은 스스로 논리 1 을 출력할 수 없다.

[TMS570LC4357 Hercules MCU Technical Reference Manual - 25.5.17 GIO Pull Disable Registers (GIOPULDIS[A-B])]

25.5.17 GIO Pull Disable Registers (GIOPULDIS[A-B])

Values in this register enable or disable the pull control capability of the pins. Figure 25-23 and Table 25-20 describe this register.

Figure 25-23. GIO Pull Disable Registers (GIOPULDIS[A-B]) [offset = 4Ch, 6Ch]



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

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.

GIOPULDIS 레지스터의 값들은 핀의 풀 제어 기능을 활성화 또는 비활성화 한다.

gioPORTA->PULDIS 를 0 으로 세팅했으므로 port 0~ port7 모두 플로팅 상태를 pull 하는 기능을 설정한다.

-gioSetDirection(gioPORTA, 0xffffffff);

```
void gioSetDirection(gioPORT_t *port, uint32 dir)
{
    port->DIR = dir;
}
```

gioPORTA->DIR 을 1 로 채움. 핀 모두를 output 으로 설정함.

-gioSetPort(gioPORTA, 0xffffffff);

```
void gioSetPort(gioPORT_t *port, uint32 value)
{
/* USER CODE BEGIN (6) */
/* USER CODE END */

    port->DOUT = value;

/* USER CODE BEGIN (7) */
/* USER CODE END */

}
```

gioPORTA->DOUT 을 전부 1 로 채움. 모든 포트가 low 로 구성됨.

-gioSetBit(gioPORTA, 4, 1);

```
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;
    }
}
```

value 가 1 이므로 else 문 만족하고 gioPORTA->DCLR 에 10000(2)를 넣음 bit4 를 1 로 채움.

-최적화

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

int main(void)
{
    gioREG->GCR0    = 1U;
    gioREG->ENACLR = 0xFFU;
    gioREG->LVLCLR = 0xFFU;
    gioPORTA->DIR   = (uint32)((uint32)1U << 4U);

    gioSetBit(gioPORTA, 4, 1);

    while(1);

    return 0;
}
```

이전 원본 코드를 보면 같은 동작을 하는 코드가 있어 성능 향상을 위해 최적화를 하게 되면 위의 코드와 같이 된다..

gioInit();
gioSetDirection(gioPORTA, 0xffffffff);
gioSetPort(gioPORTA, 0xffffffff);
	gioREG->GCR0 = 1U;
	gioREG->ENACLR = 0xFFU;
	gioREG->LVLCLR = 0xFFU;
	gioPORTA->DIR = (uint32)((uint32)1U << 4U);

왼쪽 3 줄의 코드를 오른쪽 4 줄의 코드로 바꾸었다.