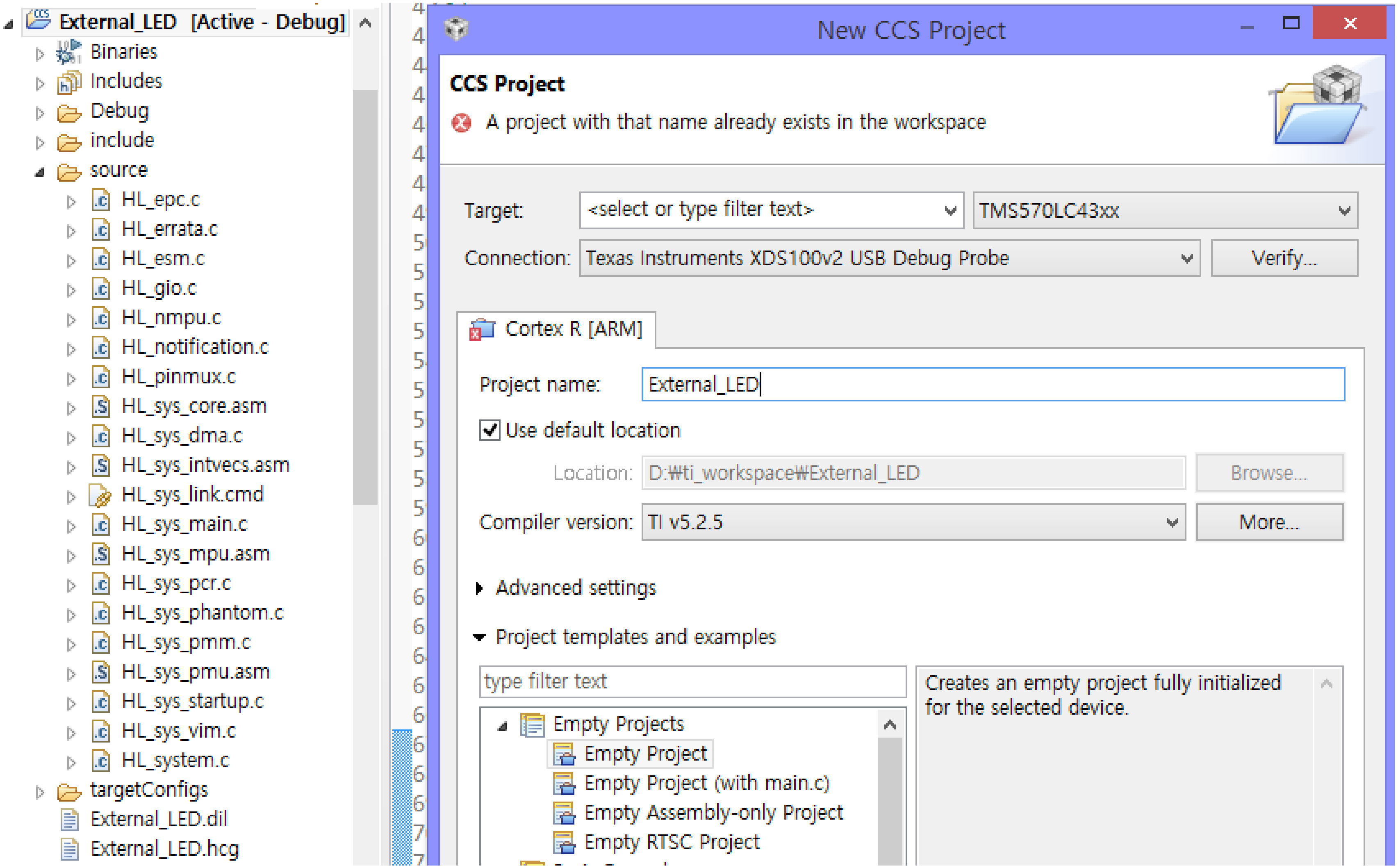
Xilinx Zynq FPGA, TI DSP, MCU 기반의 회로 설계 및 임베디드 전문가 과정

강사 – Innova Lee(이상훈)
gcccompil3r@gmail.com

GPIO Control with TMS570LC4357

1. **File – New – CCS Project -> Target, Connection 설정**
2. **Compiler Version – TI v5.2.6(이하 버전도 가능)**
3. **Empty Project 선택**
4. **생성하기**



New CCS Project

CCS Project

A project with that name already exists in the workspace

Target: <select or type filter text> TMS570LC43xx

Connection: Texas Instruments XDS100v2 USB Debug Probe

Cortex R [ARM]

Project name: External_LED

☑ Use default location

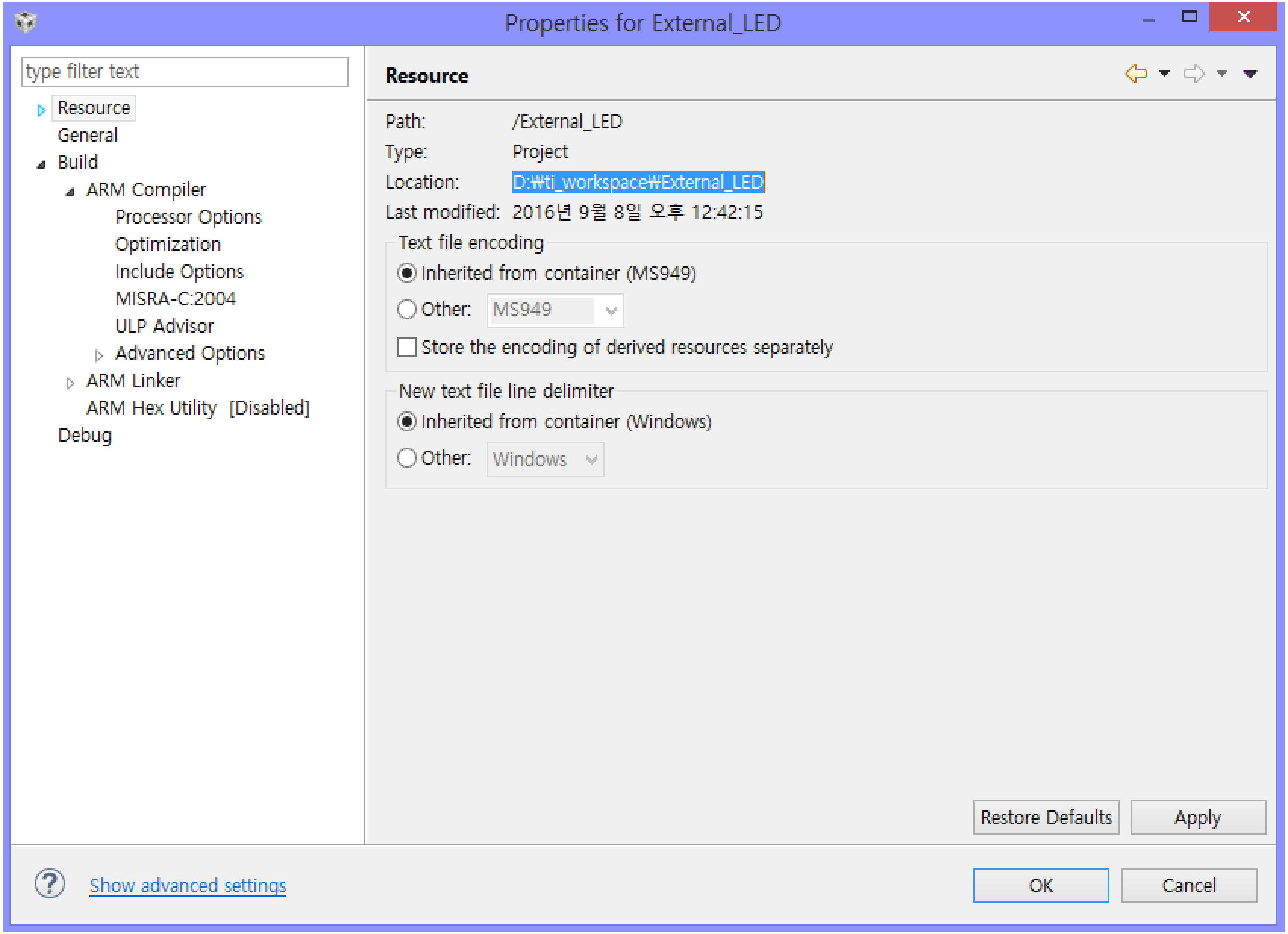
Location: D:₩ti_workspace₩External_LED

Compiler version: TI v5.2.5

Project templates and examples

Creates an empty project fully initialized for the selected device.

CCS에서 생성한 Project의 주소를 복사한다.



Properties for External_LED

type filter text

- Resource
- General
- Build
  - ARM Compiler
    - Processor Options
    - Optimization
    - Include Options
    - MISRA-C:2004
    - ULP Advisor
    - Advanced Options
  - ARM Linker
  - ARM Hex Utility [Disabled]
- Debug

Resource

Path: /External_LED
Type: Project
Location: D:₩ti_workspace₩External_LED
Last modified: 2016년 9월 8일 오후 12:42:15

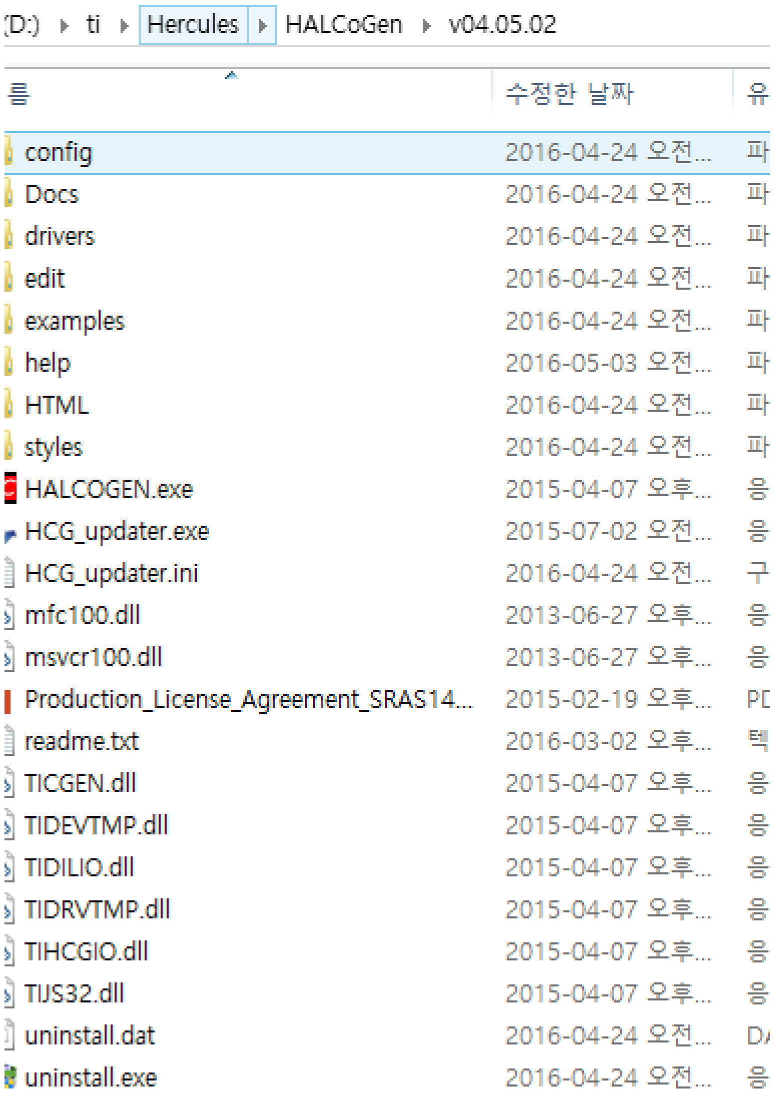
Text file encoding

● Inherited from container (MS949)
○ Other: MS949
□ Store the encoding of derived resources separately

New text file line delimiter

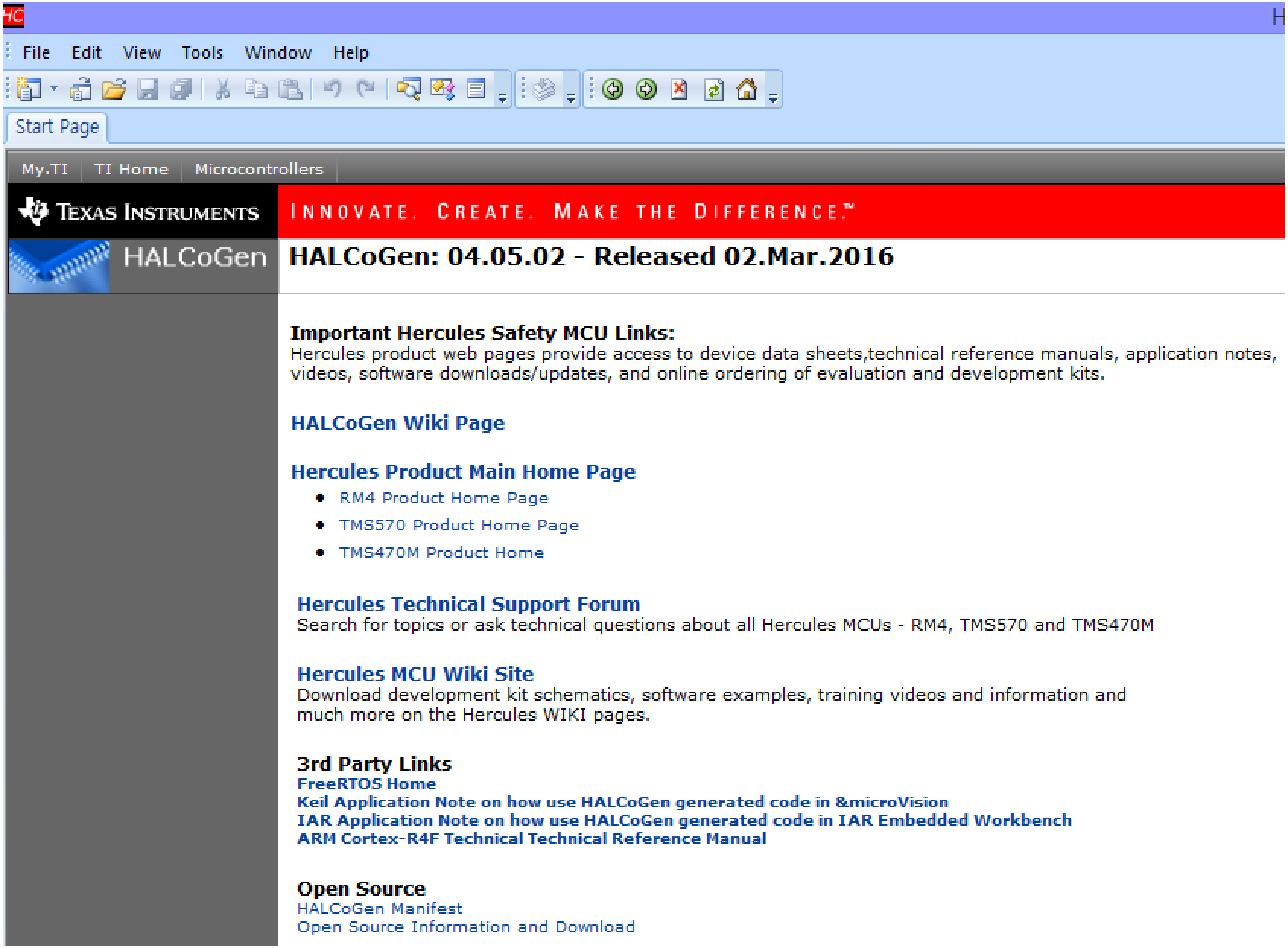
● Inherited from container (Windows)
○ Other: Windows

복사한 상태로 HALCoGen을 동작시킨다.



(D:) ▸ ti ▸ Hercules ▸ HALCoGen ▸ v04.05.02

름	수정한 날짜	유
config	2016-04-24 오전...	파
Docs	2016-04-24 오전...	파
drivers	2016-04-24 오전...	파
edit	2016-04-24 오전...	파
examples	2016-04-24 오전...	파
help	2016-05-03 오전...	파
HTML	2016-04-24 오전...	파
styles	2016-04-24 오전...	파
HALCOGEN.exe	2015-04-07 오후...	응
HCG_updater.exe	2015-07-02 오전...	응
HCG_updater.ini	2016-04-24 오전...	구
mfc100.dll	2013-06-27 오후...	응
msvcr100.dll	2013-06-27 오후...	응
Production_License_Agreement_SRAS14...	2015-02-19 오후...	PD
readme.txt	2016-03-02 오후...	텍
TICGEN.dll	2015-04-07 오후...	응
TIDEVTMP.dll	2015-04-07 오후...	응
TIDILIO.dll	2015-04-07 오후...	응
TIDRVTMP.dll	2015-04-07 오후...	응
TIHCGIO.dll	2015-04-07 오후...	응
TIJS32.dll	2015-04-07 오후...	응
uninstall.dat	2016-04-24 오전...	DA
uninstall.exe	2016-04-24 오전...	응



HALCoGen: 04.05.02 - Released 02.Mar.2016

Important Hercules Safety MCU Links:
Hercules product web pages provide access to device data sheets,technical reference manuals, application notes, videos, software downloads/updates, and online ordering of evaluation and development kits.

HALCoGen Wiki Page

Hercules Product Main Home Page

- RM4 Product Home Page
- TMS570 Product Home Page
- TMS470M Product Home

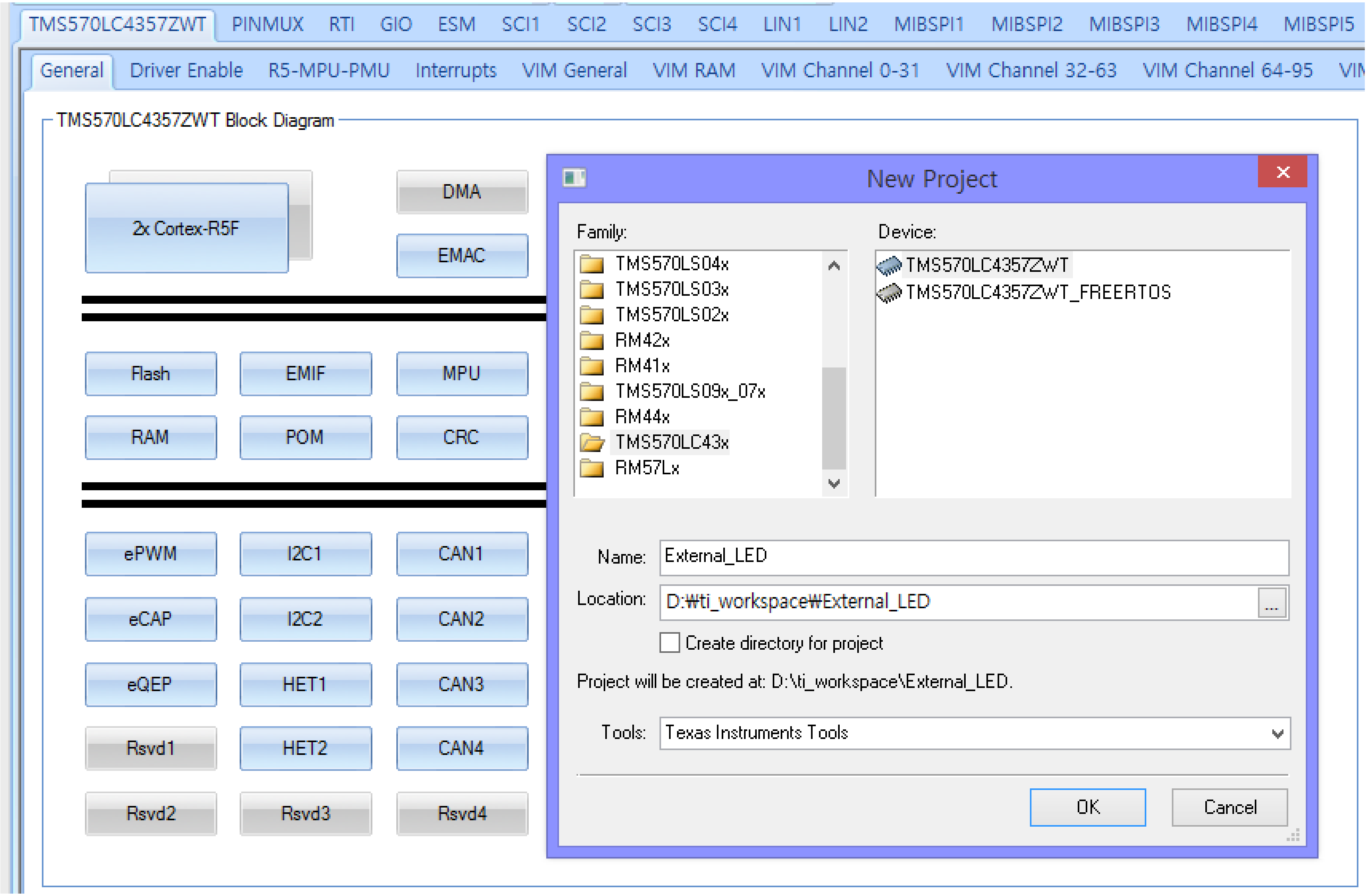
Hercules Technical Support Forum
Search for topics or ask technical questions about all Hercules MCUs - RM4, TMS570 and TMS470M

Hercules MCU Wiki Site
Download development kit schematics, software examples, training videos and information and much more on the Hercules WIKI pages.

3rd Party Links
FreeRTOS Home
Keil Application Note on how use HALCoGen generated code in &microVision
IAR Application Note on how use HALCoGen generated code in IAR Embedded Workbench
ARM Cortex-R4F Technical Technical Reference Manual

Open Source
HALCoGen Manifest
Open Source Information and Download

1. **HALCoGen에서 File – New – Project**
2. **Device 선택(TMS570LC4357ZWT)**
3. **Location에 복사한 주소 붙여넣기(Create Directory for Project 해제)**
4. **Project 생성**



New Project

Family:

- TMS570LS04x
- TMS570LS03x
- TMS570LS02x
- RM42x
- RM41x
- TMS570LS09x_07x
- RM44x
- TMS570LC43x
- RM57Lx

Device:

- TMS570LC4357ZWT
- TMS570LC4357ZWT_FREERTOS

Name: External_LED

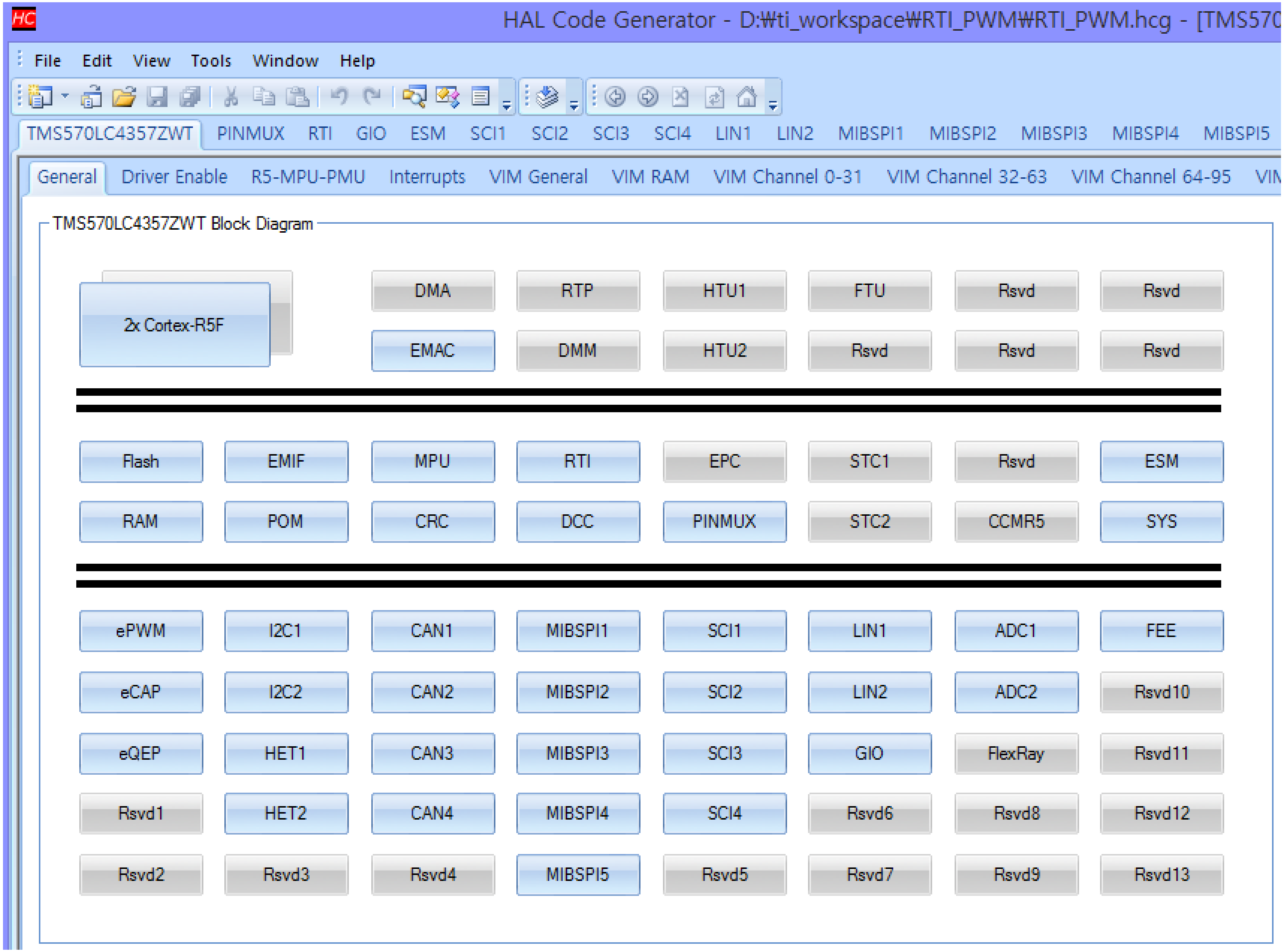
Location: D:₩ti_workspace₩External_LED

□ Create directory for project

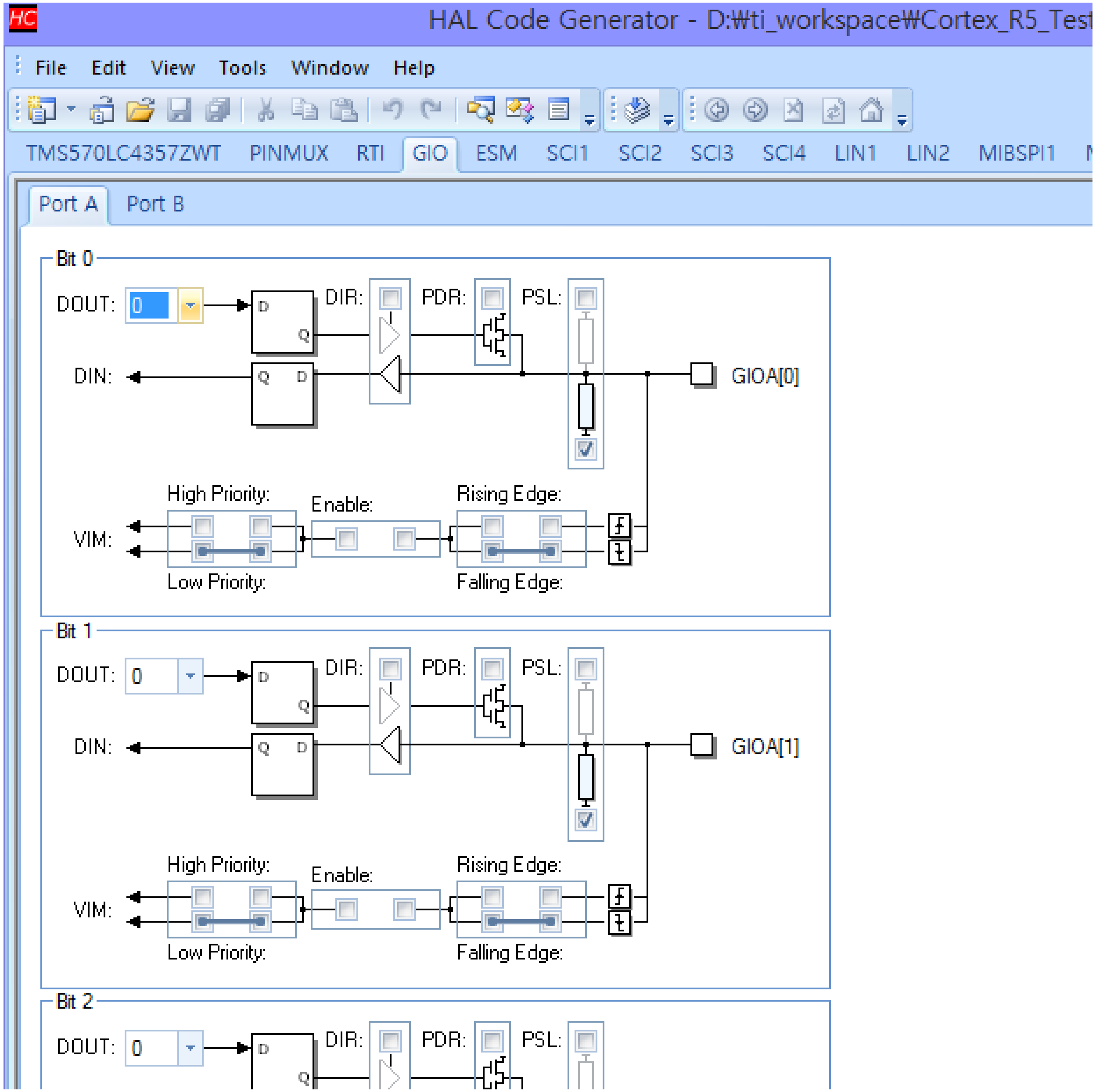
Project will be created at: D:\ti_workspace\External_LED.

Tools: Texas Instruments Tools

이번엔 PWM 을 활용하여 LED 를 제어해보도록 하자!

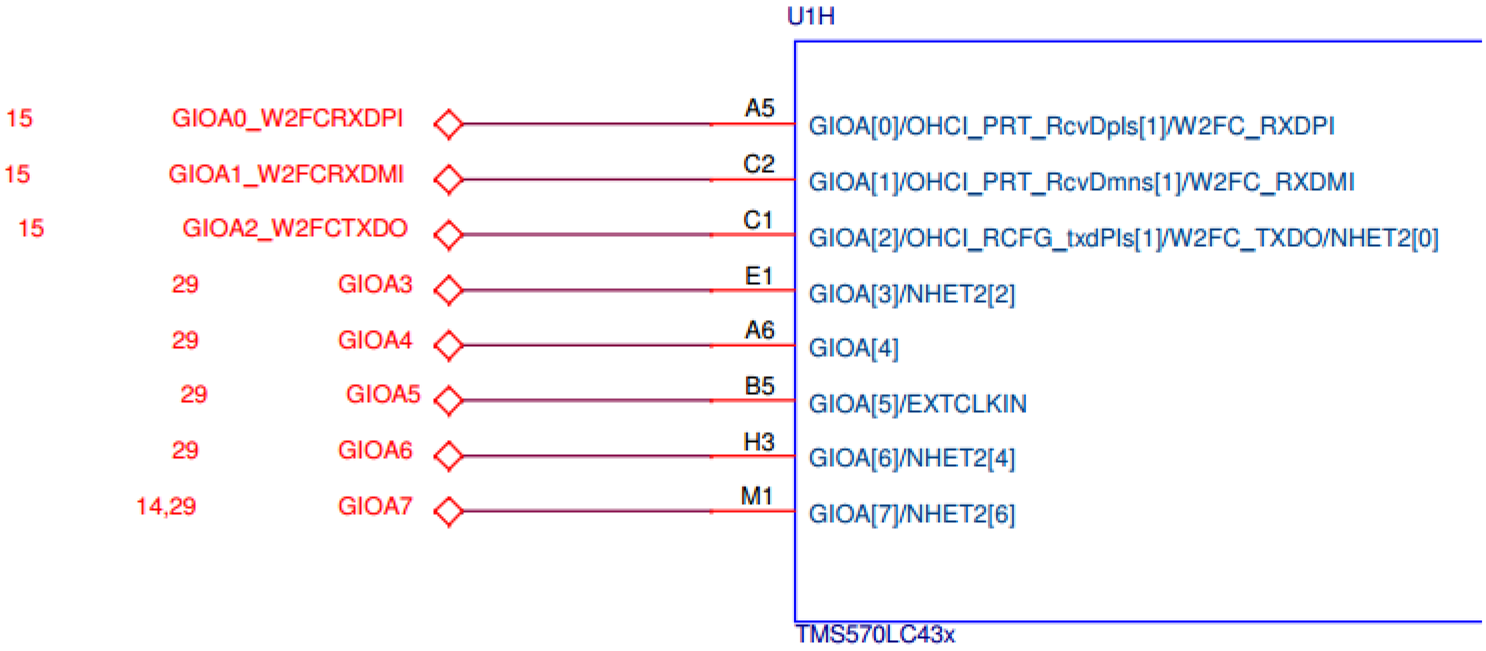


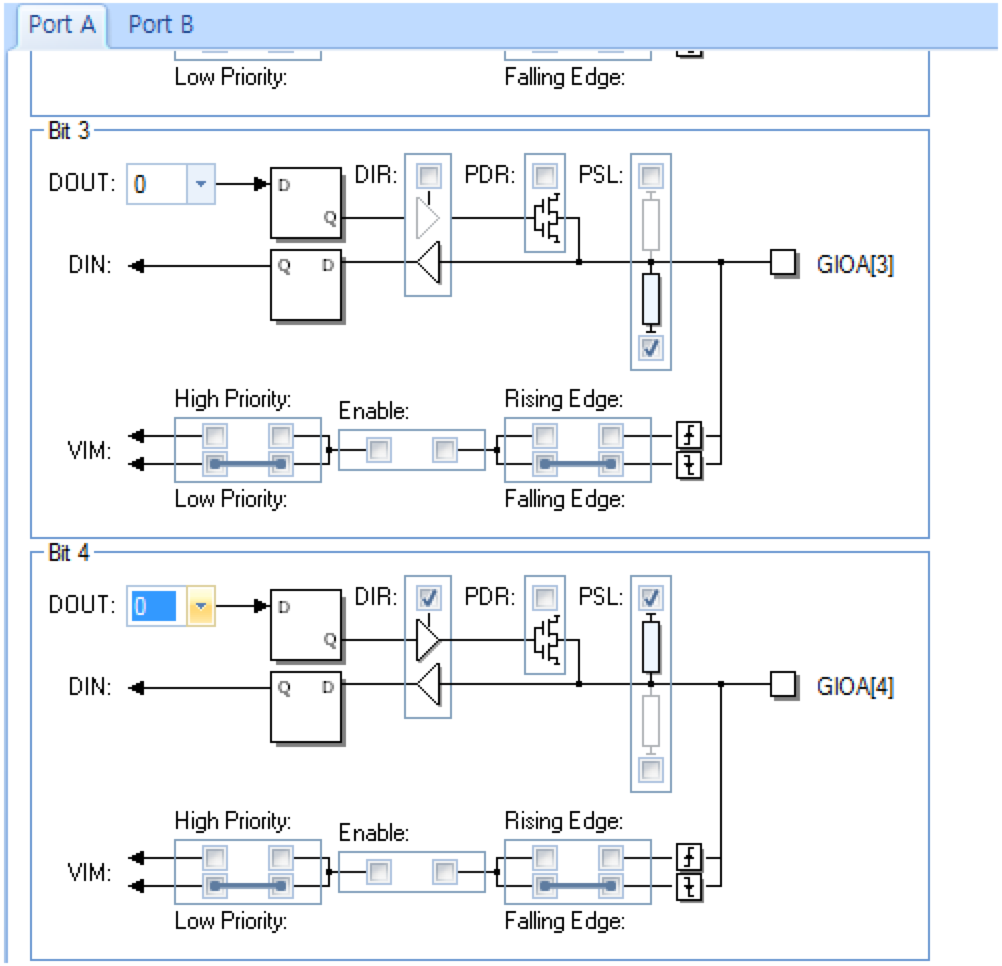
이제 GPIO를 제어해서 빵판의 LED를 제어해보도록 하자!
GIO를 누른다.



Cortex–R5의 Datasheet를 살펴보도록 한다.
http://www.ti.com/tool/TMDX570LC43HDK#technicaldocuments

무난하게 GIOA[4]를 GPIO 핀으로 활용하도록 하자!


U1H
15 GIOA0_W2FCRXDPI A5 GIOA[0]/OHCI_PRT_RcvDpls[1]/W2FC_RXDPI
15 GIOA1_W2FCRXDMI C2 GIOA[1]/OHCI_PRT_RcvDmns[1]/W2FC_RXDMI
15 GIOA2_W2FCTXDO C1 GIOA[2]/OHCI_RCFG_txdPls[1]/W2FC_TXDO/NHET2[0]
29 GIOA3 E1 GIOA[3]/NHET2[2]
29 GIOA4 A6 GIOA[4]
29 GIOA5 B5 GIOA[5]/EXTCLKIN
29 GIOA6 H3 GIOA[6]/NHET2[4]
14,29 GIOA7 M1 GIOA[7]/NHET2[6]
TMS570LC43x



Bit 3

DOUT: 0

DIR: □

PDR: □

PSL: □ ☑

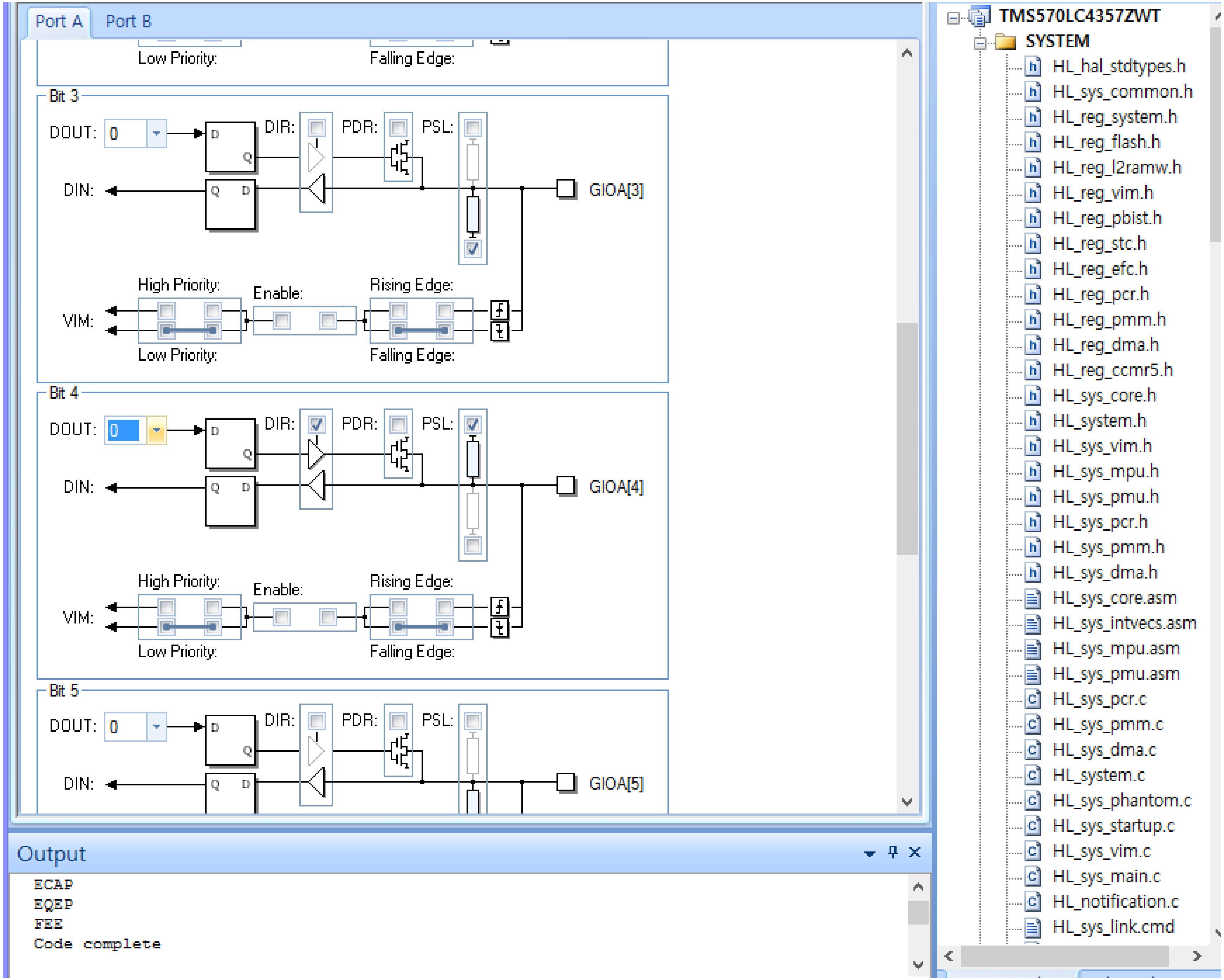
Bit 4

DOUT: 0

DIR: ☑

PDR: □

PSL: ☑ □

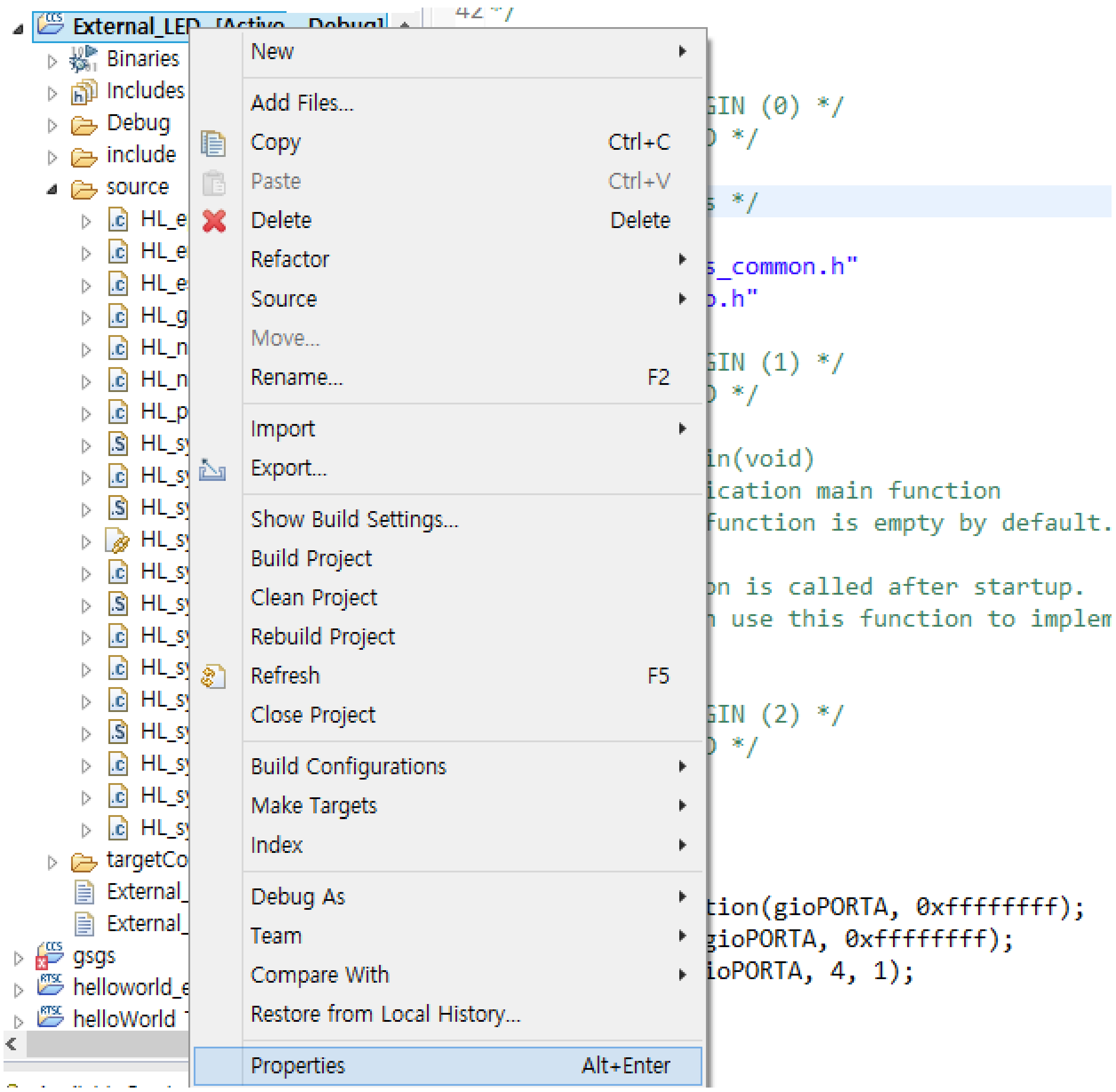


Bit 3: DOUT: 0; DIR: □; PDR: □; PSL: □ ☑

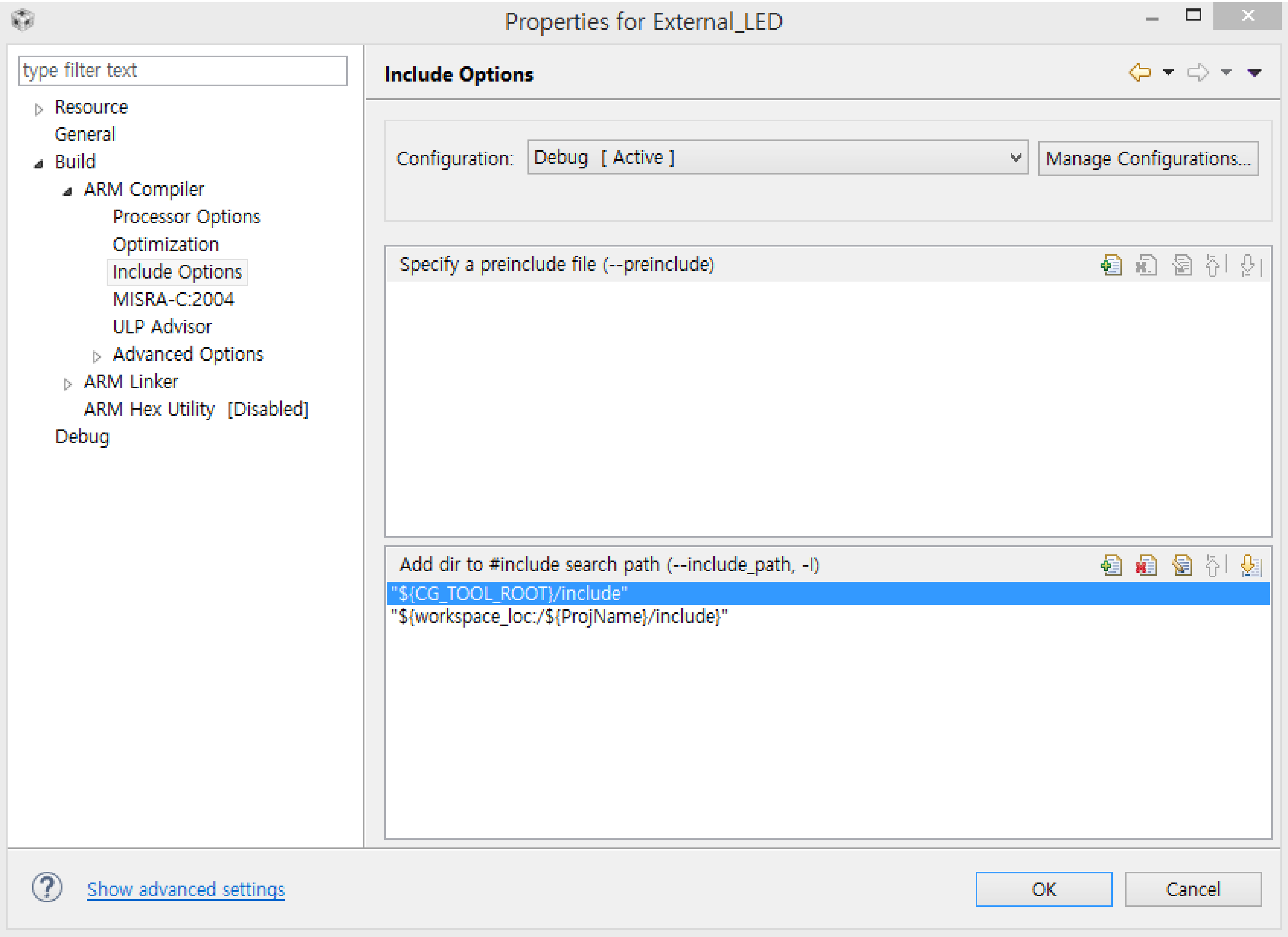
Bit 4: DOUT: 0; DIR: ☑; PDR: □; PSL: ☑ □

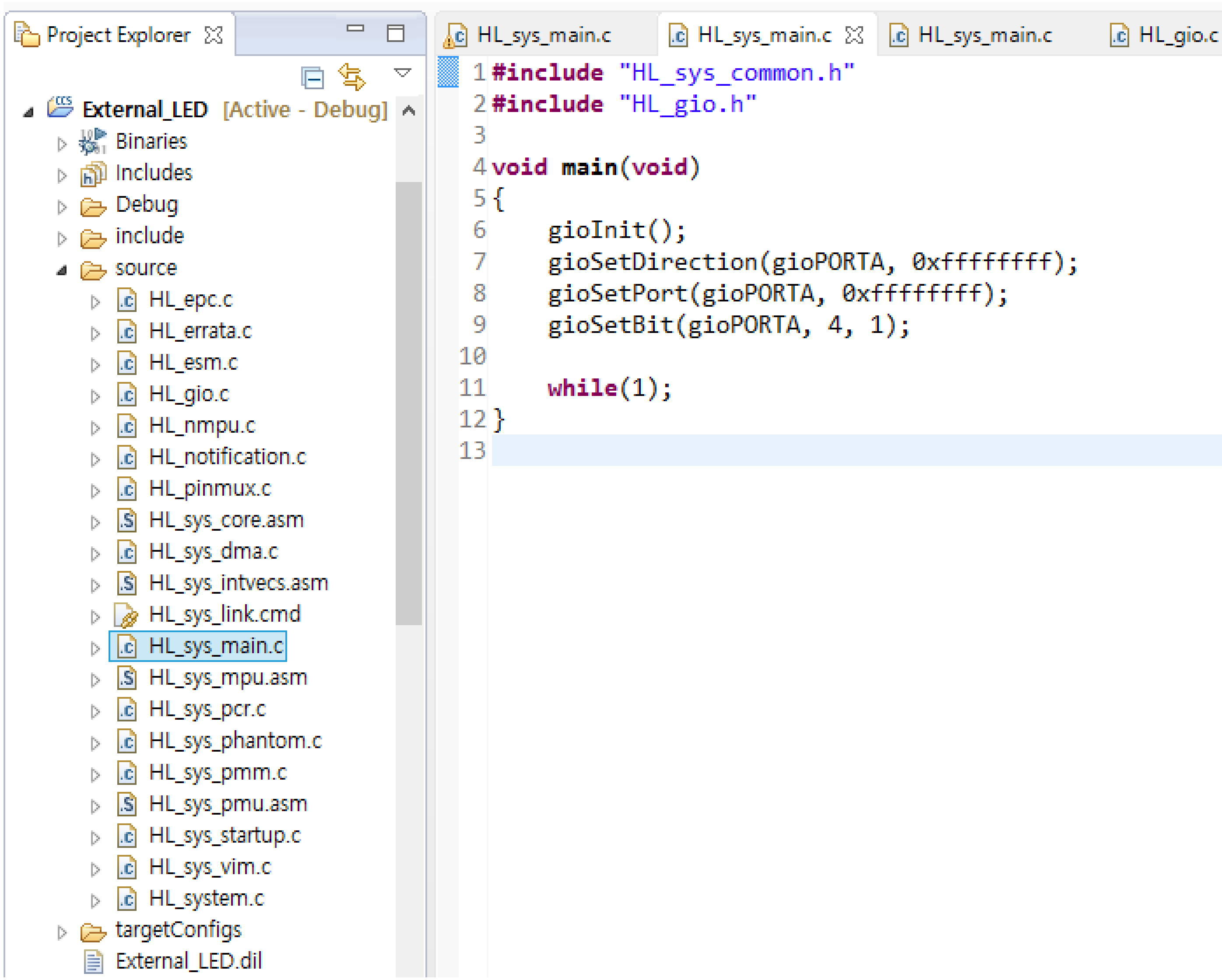
Bit 5: DOUT: 0; DIR: □; PDR: □; PSL: □

```
ECAP
EQEP
FEE
Code complete
```



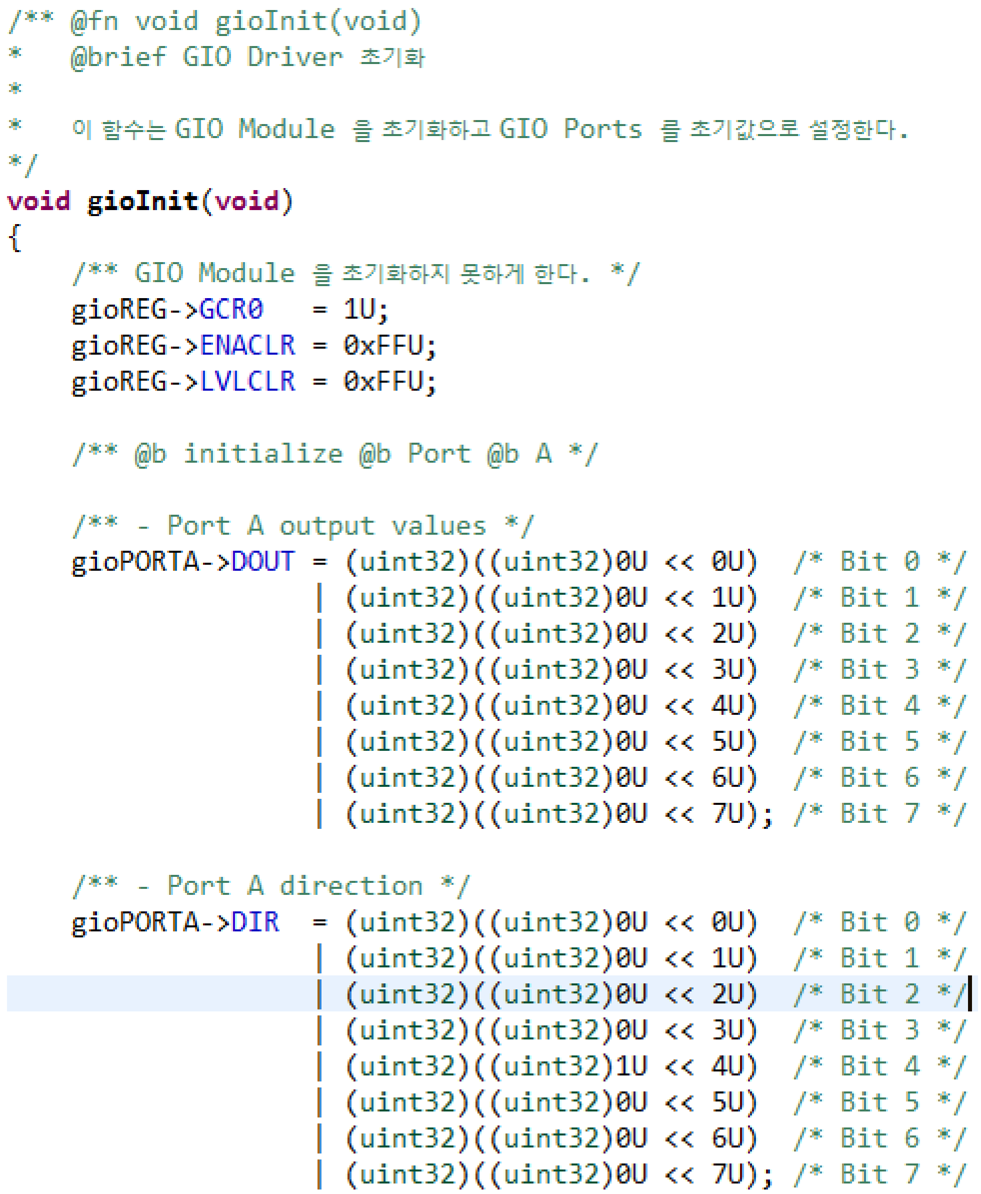
```
42 */
GIN (0) */
) */

s */

s_common.h"
o.h"

GIN (1) */
) */

in(void)
ication main function
function is empty by default.

on is called after startup.
n use this function to implem

GIN (2) */
) */

tion(gioPORTA, 0xffffffff);
gioPORTA, 0xffffffff);
ioPORTA, 4, 1);
```



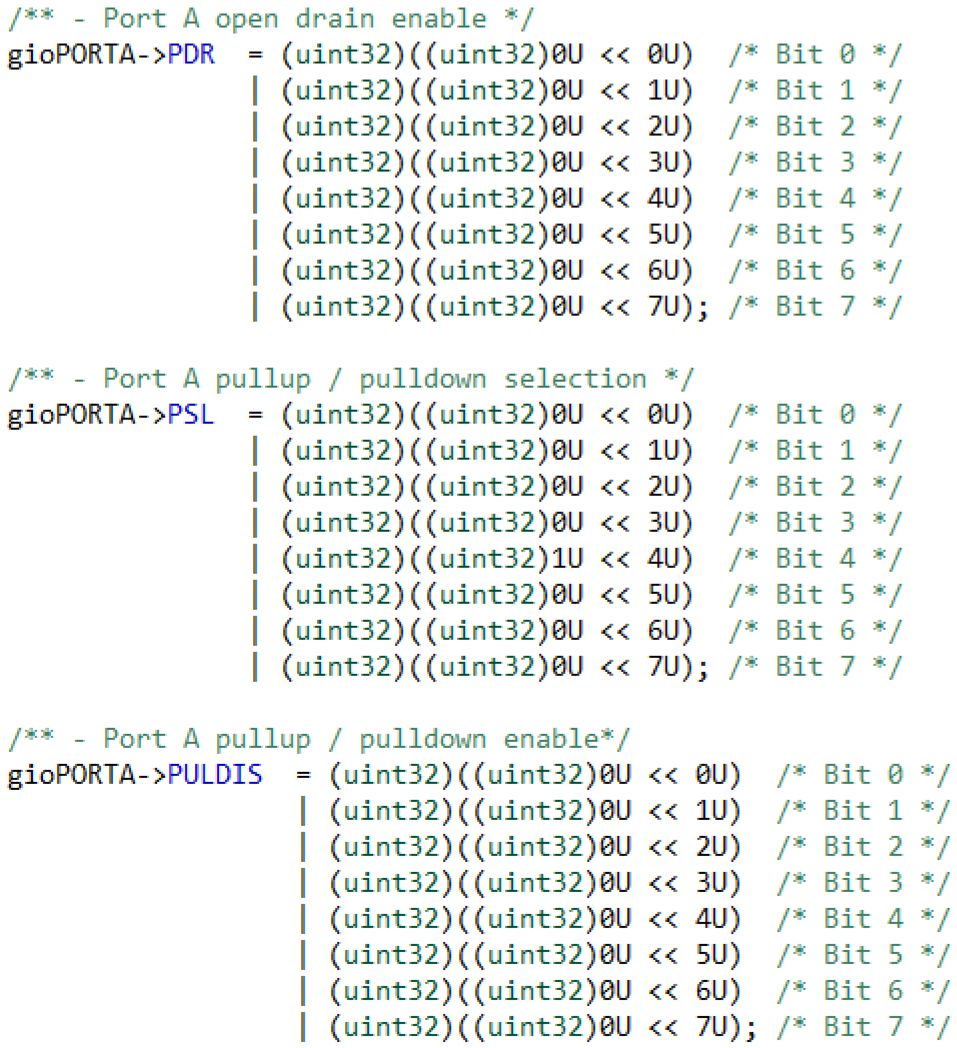


```c
#include "HL_sys_common.h"
#include "HL_gio.h"

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

    while(1);
}

```

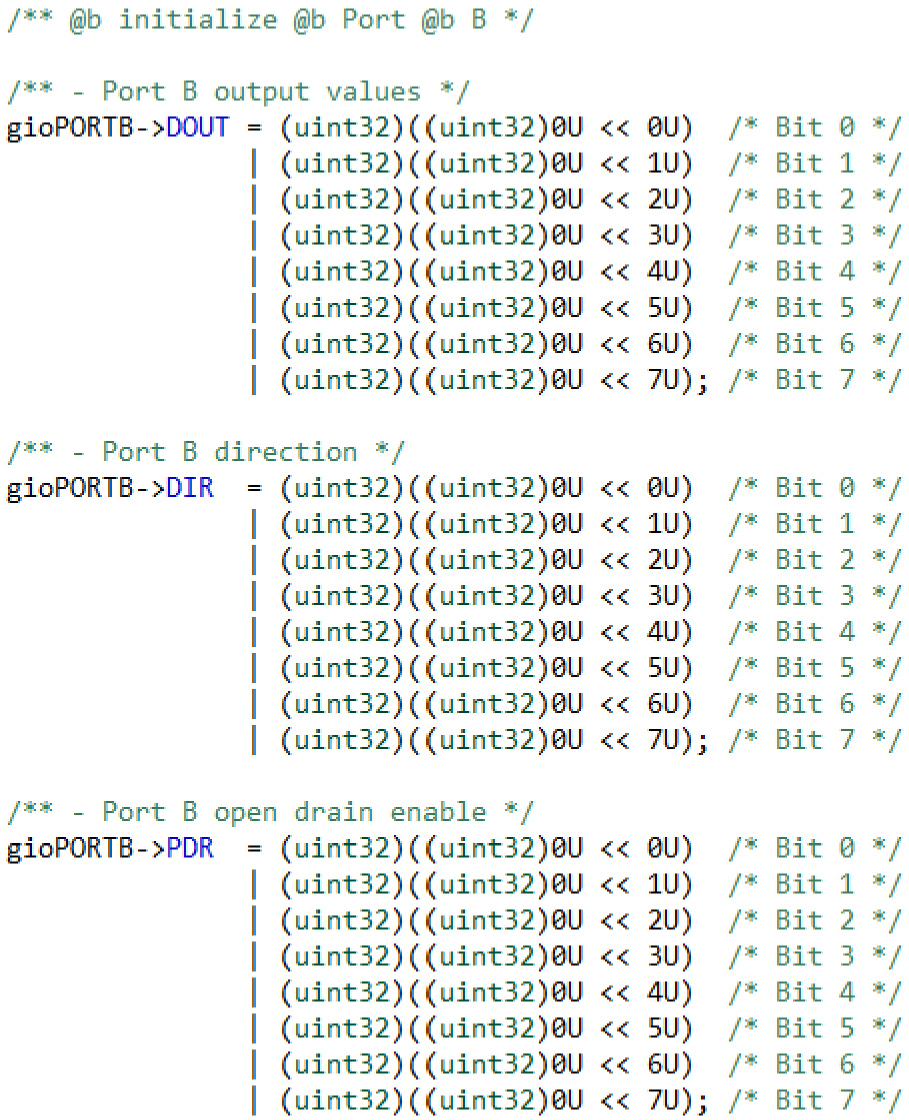
Vb
Vc
Microsoft
Autodesk



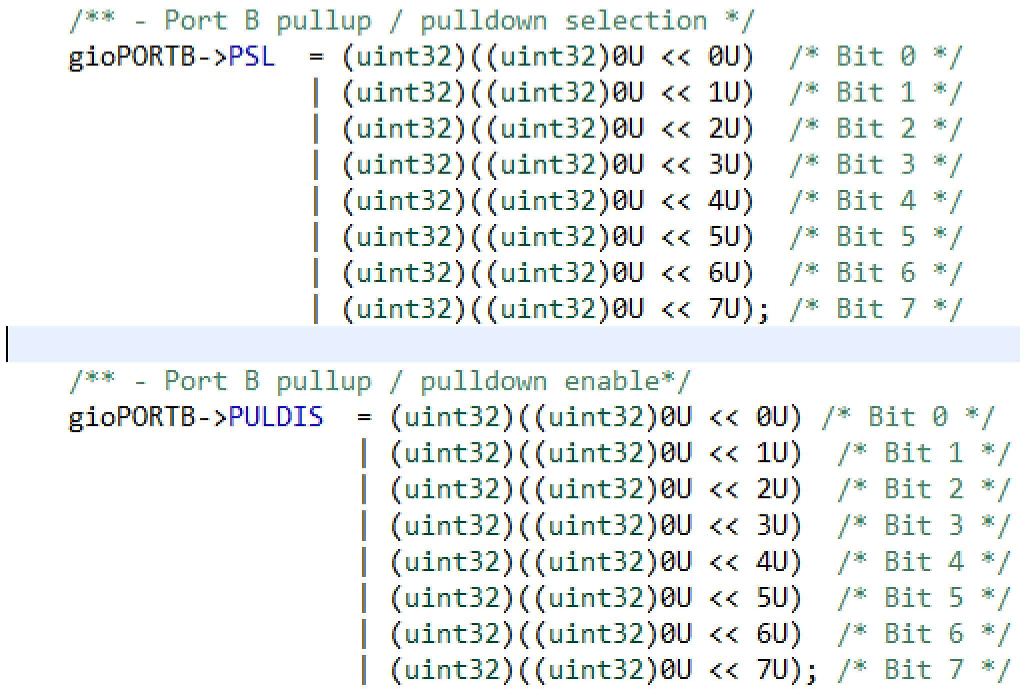
```c
/** @fn void gioInit(void)
*   @brief GIO Driver 초기화
*
*   이 함수는 GIO Module 을 초기화하고 GIO Ports 를 초기값으로 설정한다.
*/
void gioInit(void)
{
    /** GIO Module 을 초기화하지 못하게 한다. */
    gioREG->GCR0   = 1U;
    gioREG->ENACLR = 0xFFU;
    gioREG->LVLCLR = 0xFFU;

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

    /** - 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 */
```



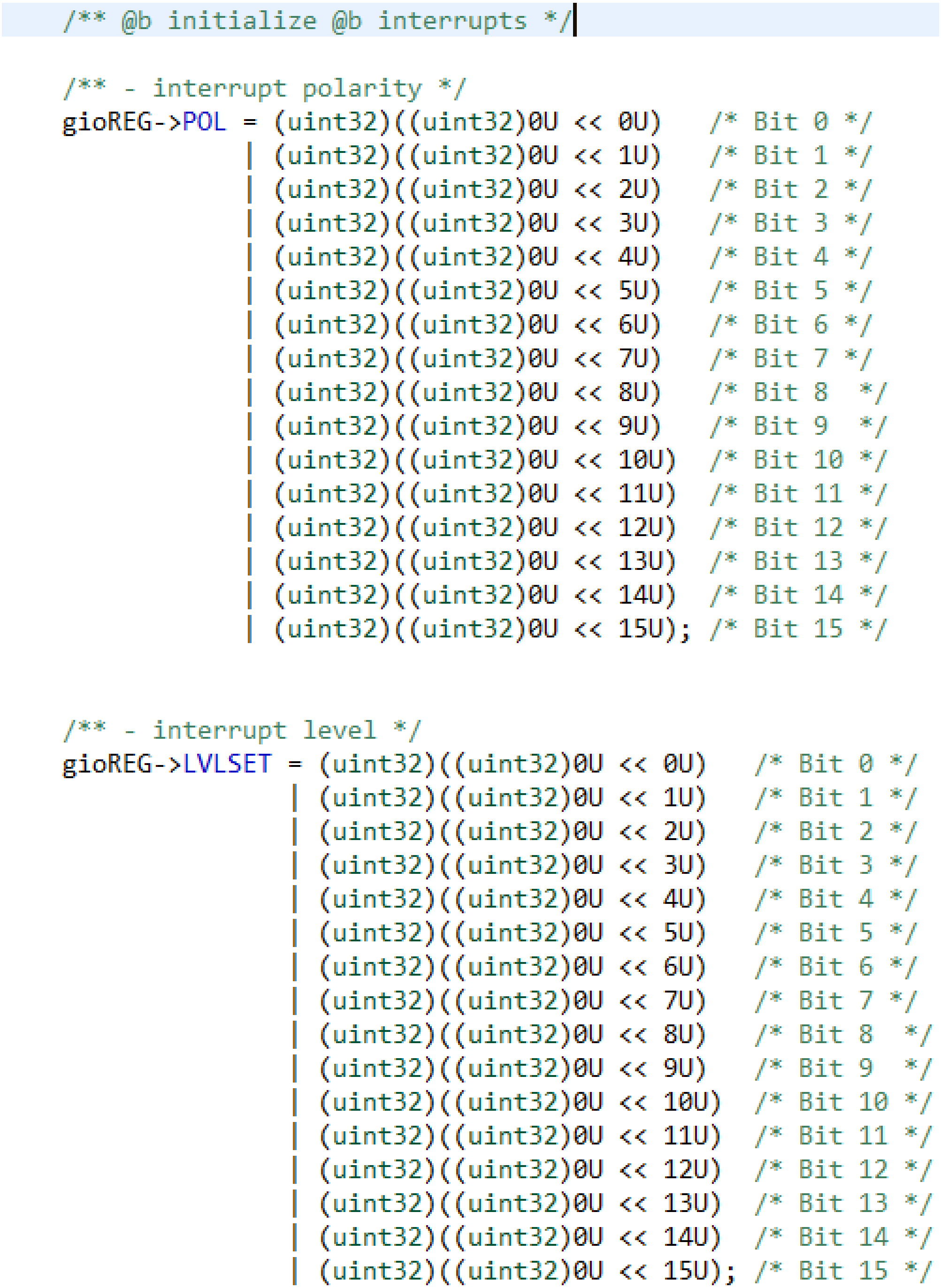
```c
/** - Port A open drain enable */
gioPORTA->PDR  = (uint32)((uint32)0U << 0U)  /* Bit 0 */
               | (uint32)((uint32)0U << 1U)  /* Bit 1 */
               | (uint32)((uint32)0U << 2U)  /* Bit 2 */
               | (uint32)((uint32)0U << 3U)  /* Bit 3 */
               | (uint32)((uint32)0U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



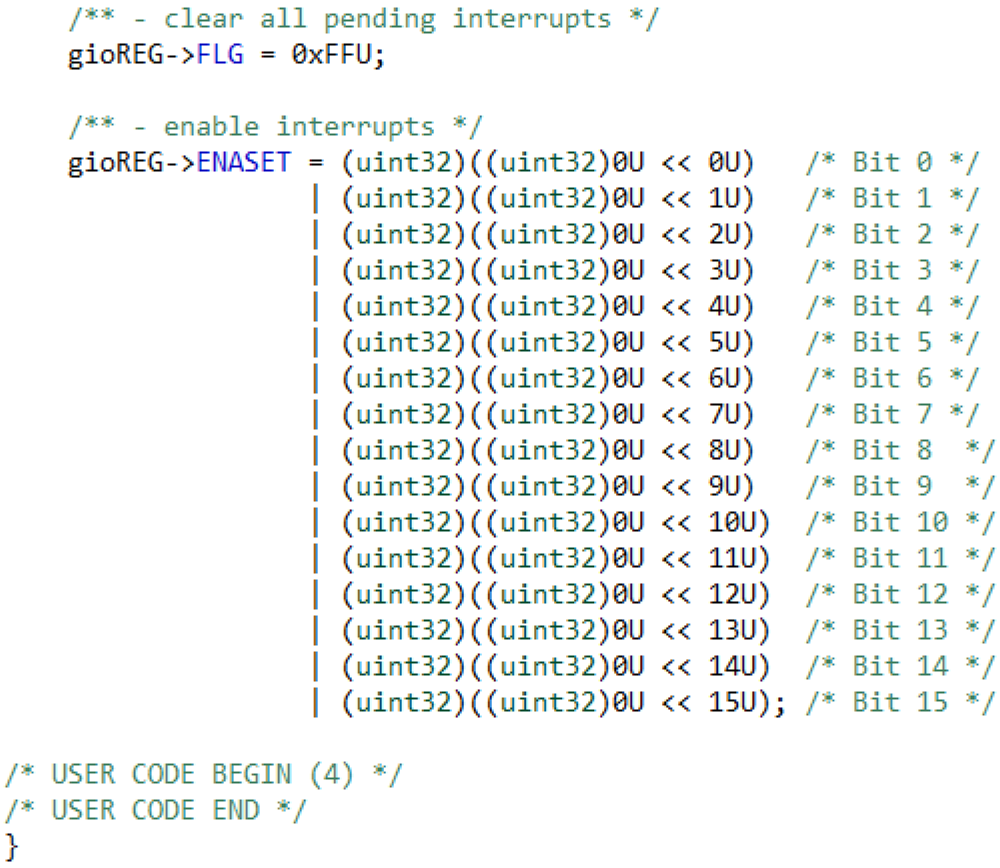
```c
/** @b initialize @b Port @b B */

/** - Port B output values */
gioPORTB->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 B direction */
gioPORTB->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)0U << 4U)  /* Bit 4 */
               | (uint32)((uint32)0U << 5U)  /* Bit 5 */
               | (uint32)((uint32)0U << 6U)  /* Bit 6 */
               | (uint32)((uint32)0U << 7U); /* Bit 7 */

/** - Port B open drain enable */
gioPORTB->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 B pullup / pulldown selection */
    gioPORTB->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)0U << 4U)  /* Bit 4 */
                   | (uint32)((uint32)0U << 5U)  /* Bit 5 */
                   | (uint32)((uint32)0U << 6U)  /* Bit 6 */
                   | (uint32)((uint32)0U << 7U); /* Bit 7 */

    /** - Port B pullup / pulldown enable*/
    gioPORTB->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 */
```

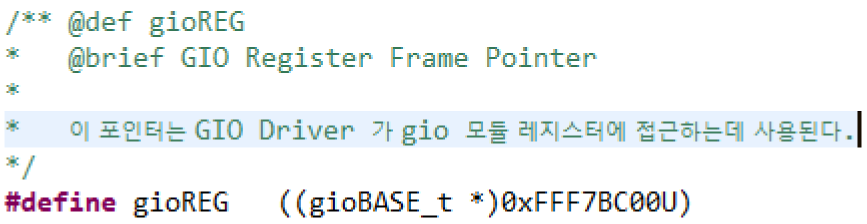


```
    /** @b initialize @b interrupts */

    /** - interrupt polarity */
    gioREG->POL = (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 */
                | (uint32)((uint32)0U << 8U)   /* Bit 8  */
                | (uint32)((uint32)0U << 9U)   /* Bit 9  */
                | (uint32)((uint32)0U << 10U)  /* Bit 10 */
                | (uint32)((uint32)0U << 11U)  /* Bit 11 */
                | (uint32)((uint32)0U << 12U)  /* Bit 12 */
                | (uint32)((uint32)0U << 13U)  /* Bit 13 */
                | (uint32)((uint32)0U << 14U)  /* Bit 14 */
                | (uint32)((uint32)0U << 15U); /* Bit 15 */


    /** - interrupt level */
    gioREG->LVLSET = (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 */
                   | (uint32)((uint32)0U << 8U)   /* Bit 8  */
                   | (uint32)((uint32)0U << 9U)   /* Bit 9  */
                   | (uint32)((uint32)0U << 10U)  /* Bit 10 */
                   | (uint32)((uint32)0U << 11U)  /* Bit 11 */
                   | (uint32)((uint32)0U << 12U)  /* Bit 12 */
                   | (uint32)((uint32)0U << 13U)  /* Bit 13 */
                   | (uint32)((uint32)0U << 14U)  /* Bit 14 */
                   | (uint32)((uint32)0U << 15U); /* Bit 15 */
```

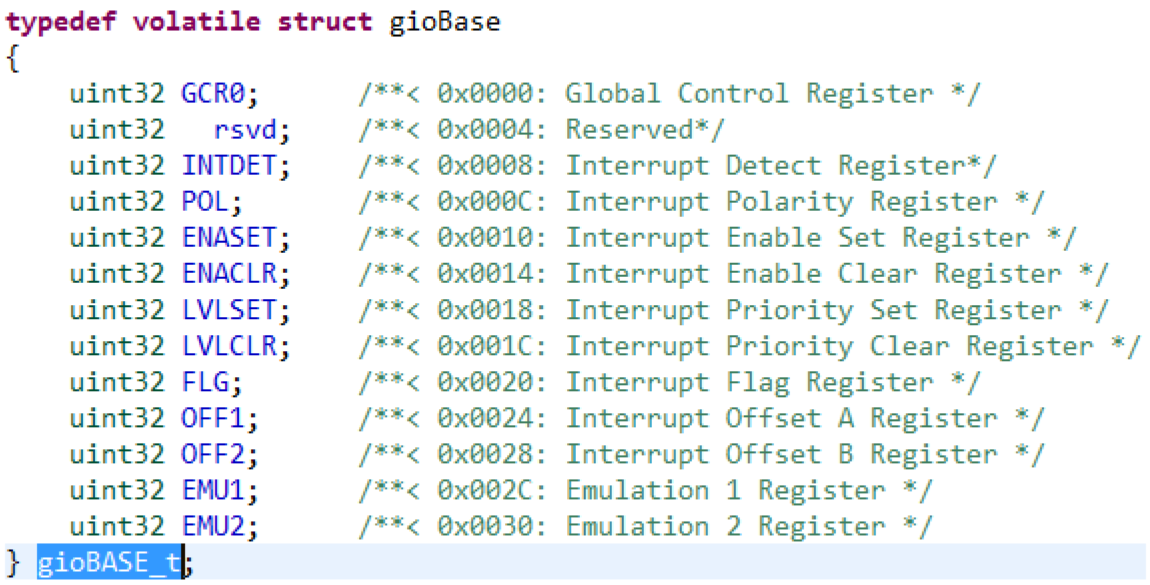


```
    /** - clear all pending interrupts */
    gioREG->FLG = 0xFFU;

    /** - enable interrupts */
    gioREG->ENASET = (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 */
                   | (uint32)((uint32)0U << 8U)   /* Bit 8  */
                   | (uint32)((uint32)0U << 9U)   /* Bit 9  */
                   | (uint32)((uint32)0U << 10U)  /* Bit 10 */
                   | (uint32)((uint32)0U << 11U)  /* Bit 11 */
                   | (uint32)((uint32)0U << 12U)  /* Bit 12 */
                   | (uint32)((uint32)0U << 13U)  /* Bit 13 */
                   | (uint32)((uint32)0U << 14U)  /* Bit 14 */
                   | (uint32)((uint32)0U << 15U); /* Bit 15 */

/* USER CODE BEGIN (4) */
/* USER CODE END */
}
```

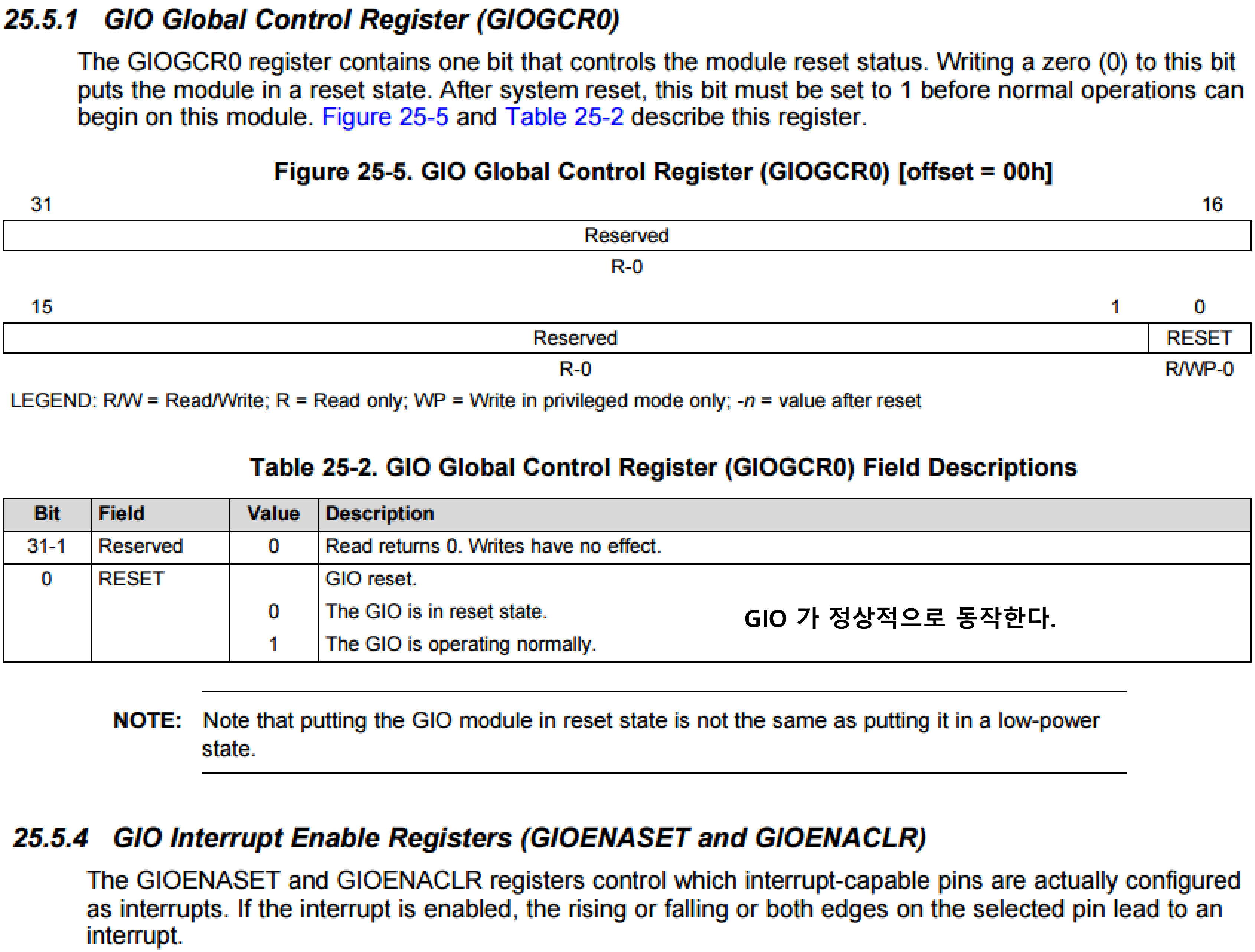
gioREG 에 대해 살펴보도록 하자!



```
/** @def gioREG
*   @brief GIO Register Frame Pointer
*
*   이 포인터는 GIO Driver 가 gio 모듈 레지스터에 접근하는데 사용된다.
*/
#define gioREG   ((gioBASE_t *)0xFFF7BC00U)
```



```
typedef volatile struct gioBase
{
    uint32 GCR0;      /**< 0x0000: Global Control Register */
    uint32   rsvd;    /**< 0x0004: Reserved*/
    uint32 INTDET;    /**< 0x0008: Interrupt Detect Register*/
    uint32 POL;       /**< 0x000C: Interrupt Polarity Register */
    uint32 ENASET;    /**< 0x0010: Interrupt Enable Set Register */
    uint32 ENACLR;    /**< 0x0014: Interrupt Enable Clear Register */
    uint32 LVLSET;    /**< 0x0018: Interrupt Priority Set Register */
    uint32 LVLCLR;    /**< 0x001C: Interrupt Priority Clear Register */
    uint32 FLG;       /**< 0x0020: Interrupt Flag Register */
    uint32 OFF1;      /**< 0x0024: Interrupt Offset A Register */
    uint32 OFF2;      /**< 0x0028: Interrupt Offset B Register */
    uint32 EMU1;      /**< 0x002C: Emulation 1 Register */
    uint32 EMU2;      /**< 0x0030: Emulation 2 Register */
} gioBASE_t;
```



25.5.1 GIO Global Control Register (GIOGCR0)

The GIOGCR0 register contains one bit that controls the module reset status. Writing a zero (0) to this bit puts the module in a reset state. After system reset, this bit must be set to 1 before normal operations can begin on this module. Figure 25-5 and Table 25-2 describe this register.

Figure 25-5. GIO Global Control Register (GIOGCR0) [offset = 00h]

31 … 16
Reserved
R-0

15 … 1	0
Reserved	RESET
R-0	R/WP-0

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

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

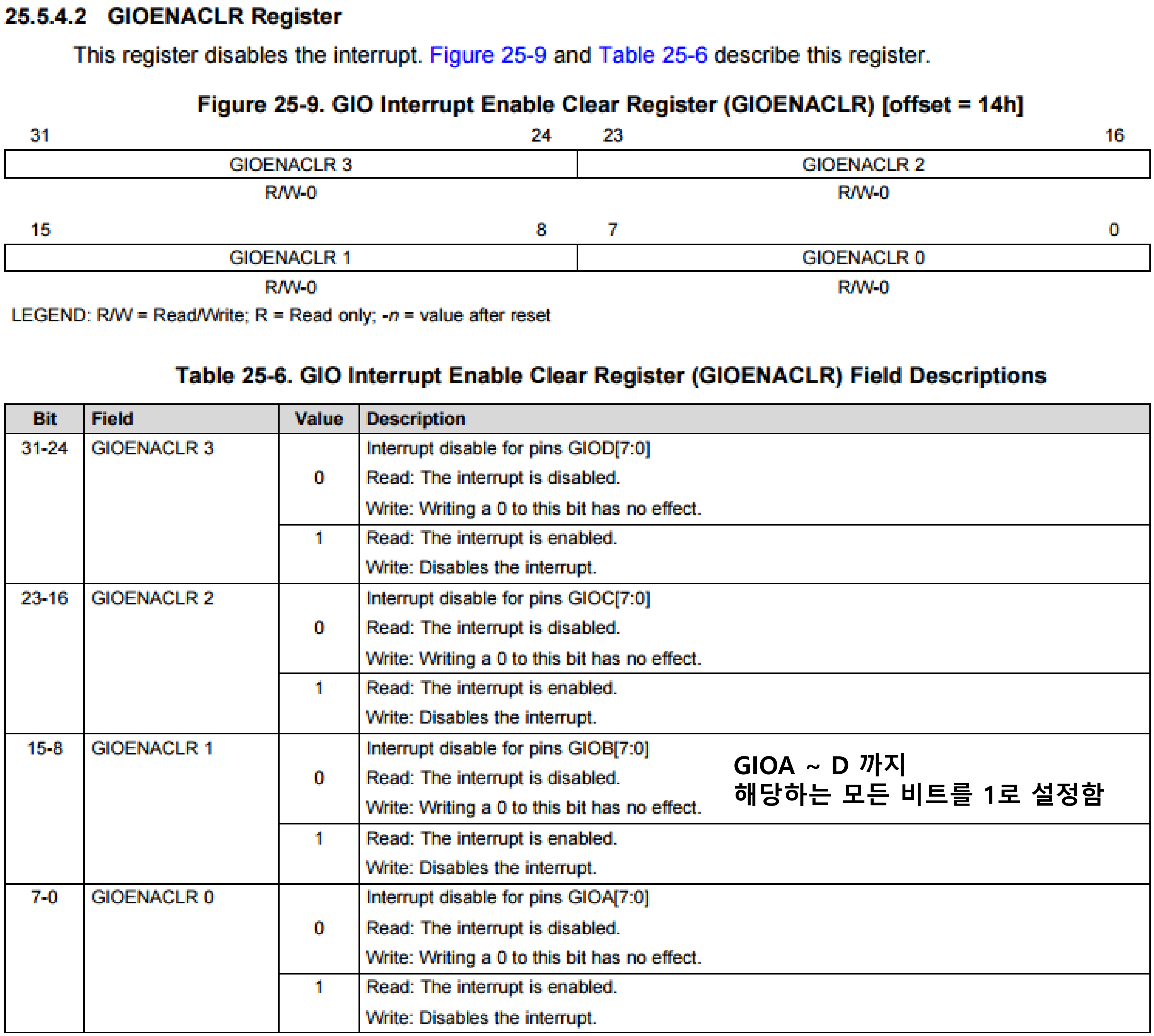
Bit	Field	Value	Description
31-1	Reserved	0	Read returns 0. Writes have no effect.
0	RESET		GIO reset.
		0	The GIO is in reset state.
		1	The GIO is operating normally.

GIO 가 정상적으로 동작한다.

NOTE: Note that putting the GIO module in reset state is not the same as putting it in a low-power state.

25.5.4 GIO Interrupt Enable Registers (GIOENASET and GIOENACLR)

The GIOENASET and GIOENACLR registers control which interrupt-capable pins are actually configured as interrupts. If the interrupt is enabled, the rising or falling or both edges on the selected pin lead to an interrupt.



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

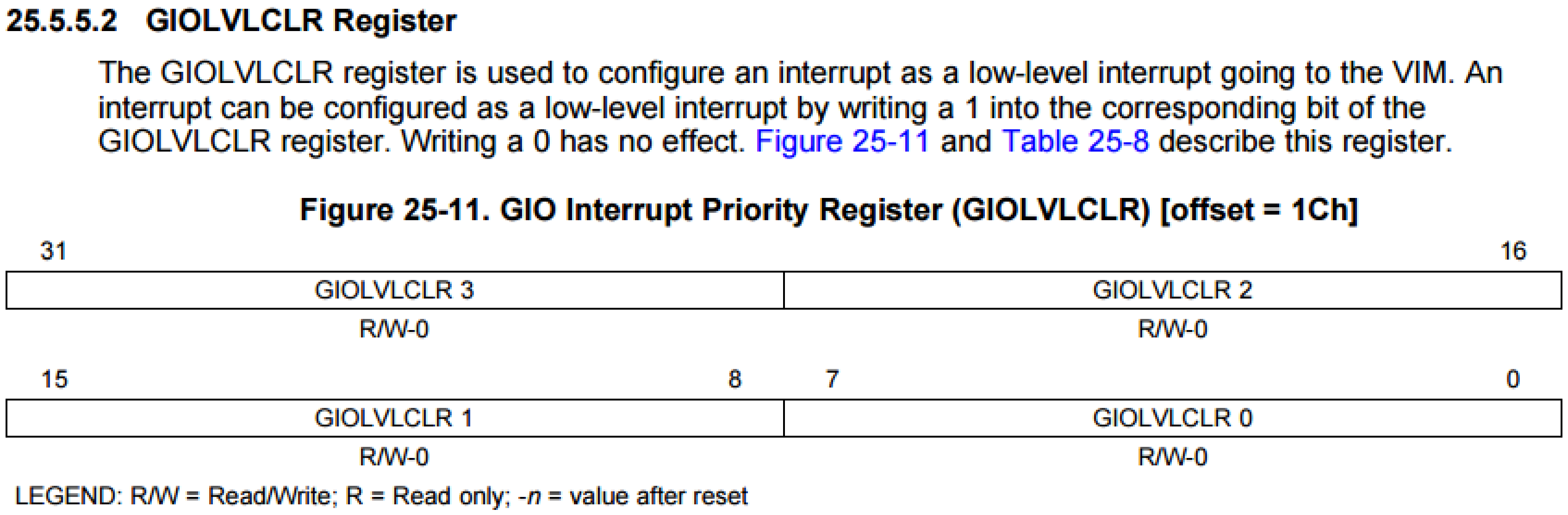
15 – 8	7 – 0
GIOENACLR 1	GIOENACLR 0
R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*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.

GIOA ~ D 까지
해당하는 모든 비트를 1로 설정함



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]

31–24	23–16
GIOLVLCLR 3	GIOLVLCLR 2
R/W-0	R/W-0

15–8	7–0
GIOLVLCLR 1	GIOLVLCLR 0
R/W-0	R/W-0

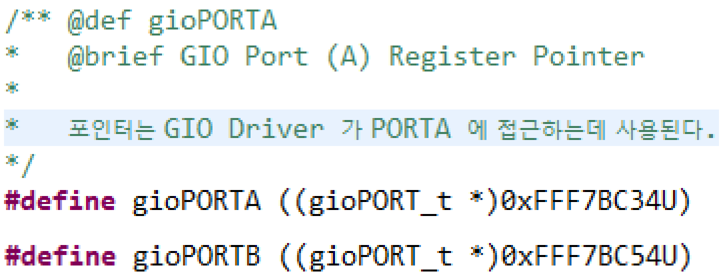
LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

GIOOFF1 ~ 2 과 GIOEMU1 ~ 2 에 따라 우선순위가 부여되게 된다.

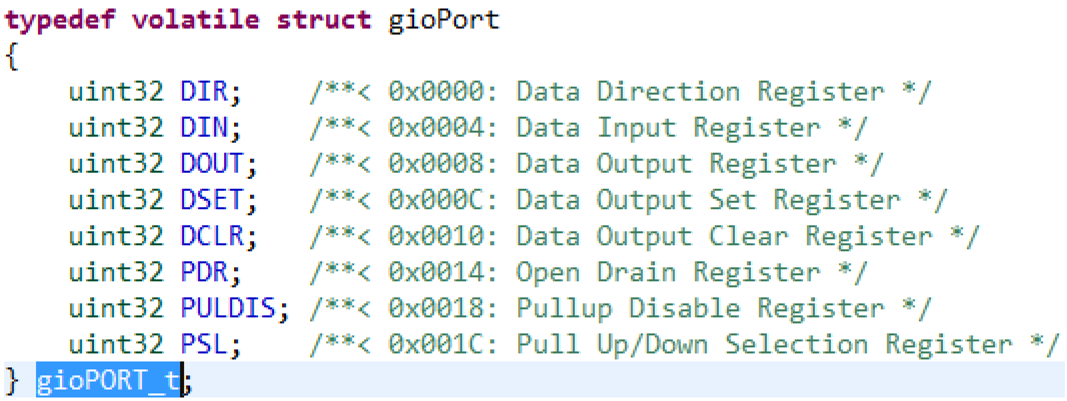
Table 25-8. GIO Interrupt Priority Register (GIOLVLCLR) Field Descriptions

Bit	Field	Value	Description
31-24	GIOLVLCLR 3	0	GIO low-priority interrupt for pins GIOD[7:0] 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	0	GIO low-priority interrupt for pins GIOC[7:0] 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	0	GIO low-priority interrupt for pins GIOB[7:0] 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	0	GIO low-priority interrupt for pins GIOA[7:0] 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.

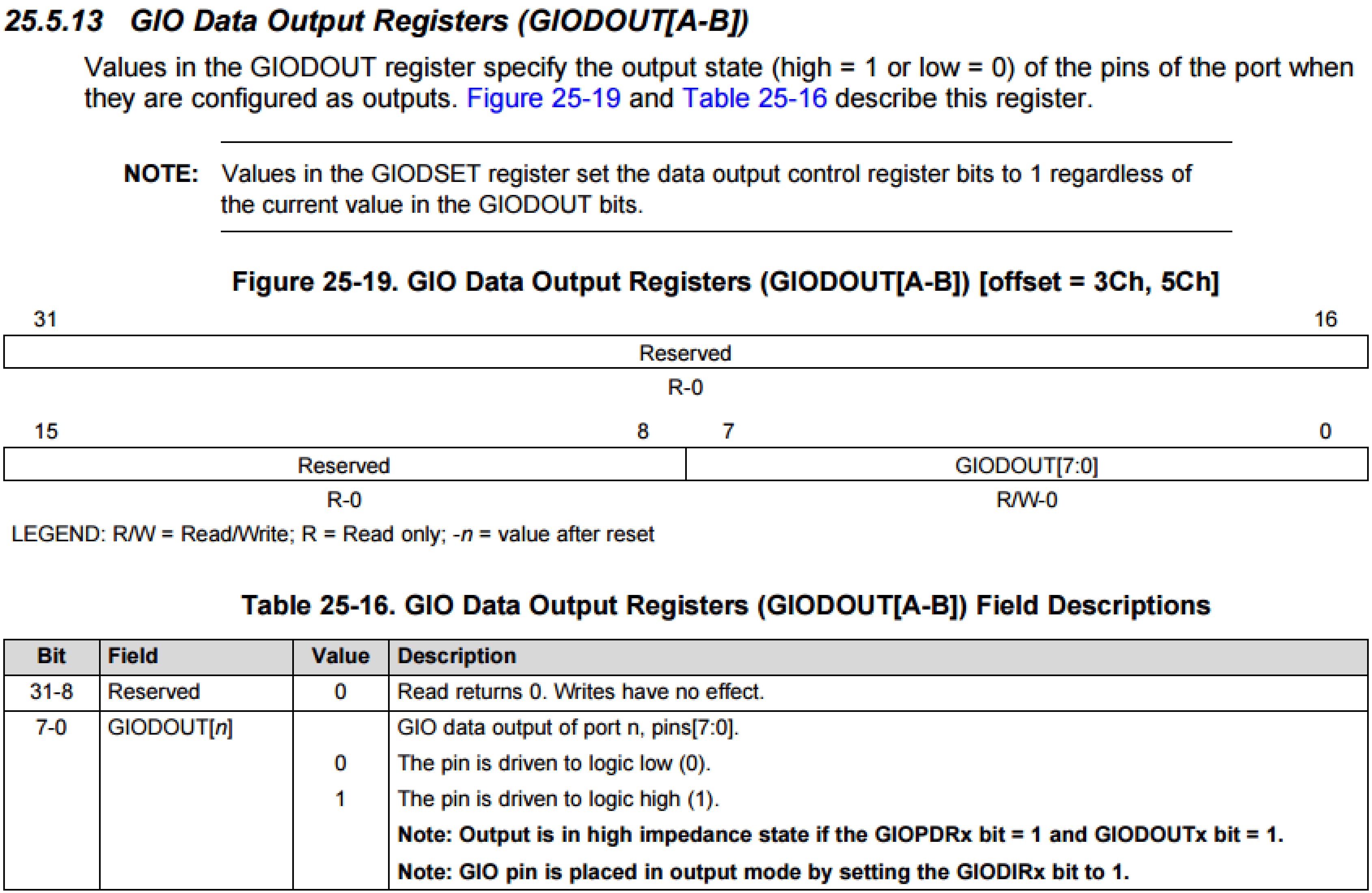
gioPORTA & gioPORTB 에 대해 살펴보도록 하자!



```c
/** @def gioPORTA
*   @brief GIO Port (A) Register Pointer
*
*   포인터는 GIO Driver 가 PORTA 에 접근하는데 사용된다.
*/
#define gioPORTA ((gioPORT_t *)0xFFF7BC34U)

#define gioPORTB ((gioPORT_t *)0xFFF7BC54U)
```



```c
typedef volatile struct gioPort
{
    uint32 DIR;    /**< 0x0000: Data Direction Register */
    uint32 DIN;    /**< 0x0004: Data Input Register */
    uint32 DOUT;   /**< 0x0008: Data Output Register */
    uint32 DSET;   /**< 0x000C: Data Output Set Register */
    uint32 DCLR;   /**< 0x0010: Data Output Clear Register */
    uint32 PDR;    /**< 0x0014: Open Drain Register */
    uint32 PULDIS; /**< 0x0018: Pullup Disable Register */
    uint32 PSL;    /**< 0x001C: Pull Up/Down Selection Register */
} gioPORT_t;
```



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]

<table>
<tr><td>31</td><td>16</td></tr>
<tr><td colspan="2">Reserved</td></tr>
<tr><td colspan="2">R-0</td></tr>
</table>

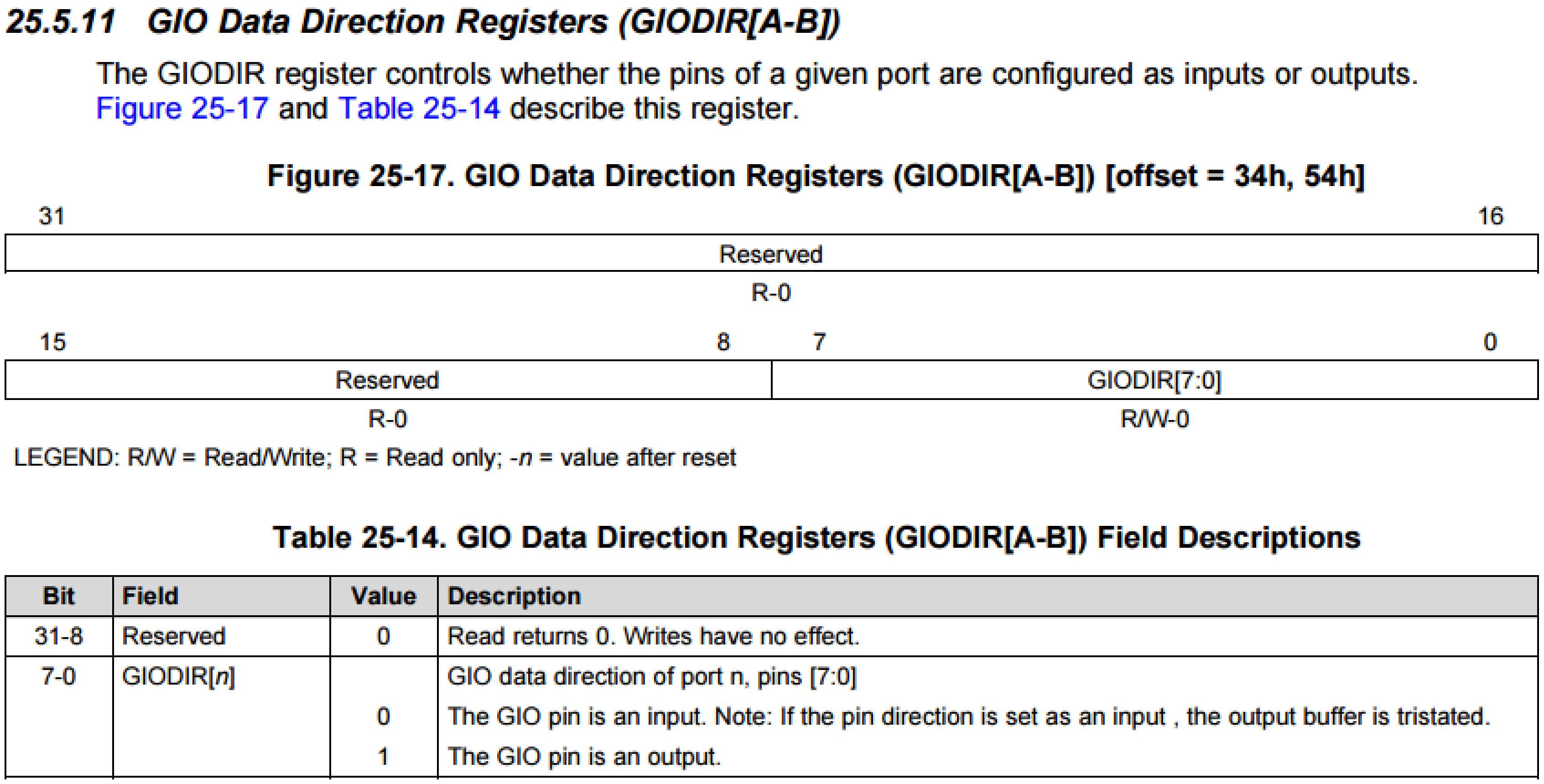
<table>
<tr><td>15</td><td>8</td><td>7</td><td>0</td></tr>
<tr><td colspan="2">Reserved</td><td colspan="2">GIODOUT[7:0]</td></tr>
<tr><td colspan="2">R-0</td><td colspan="2">R/W-0</td></tr>
</table>

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	Read returns 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.

GIOA ~ B 의 출력이 모두 Low 로 구동된다.



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]

31	16
Reserved	
R-0	

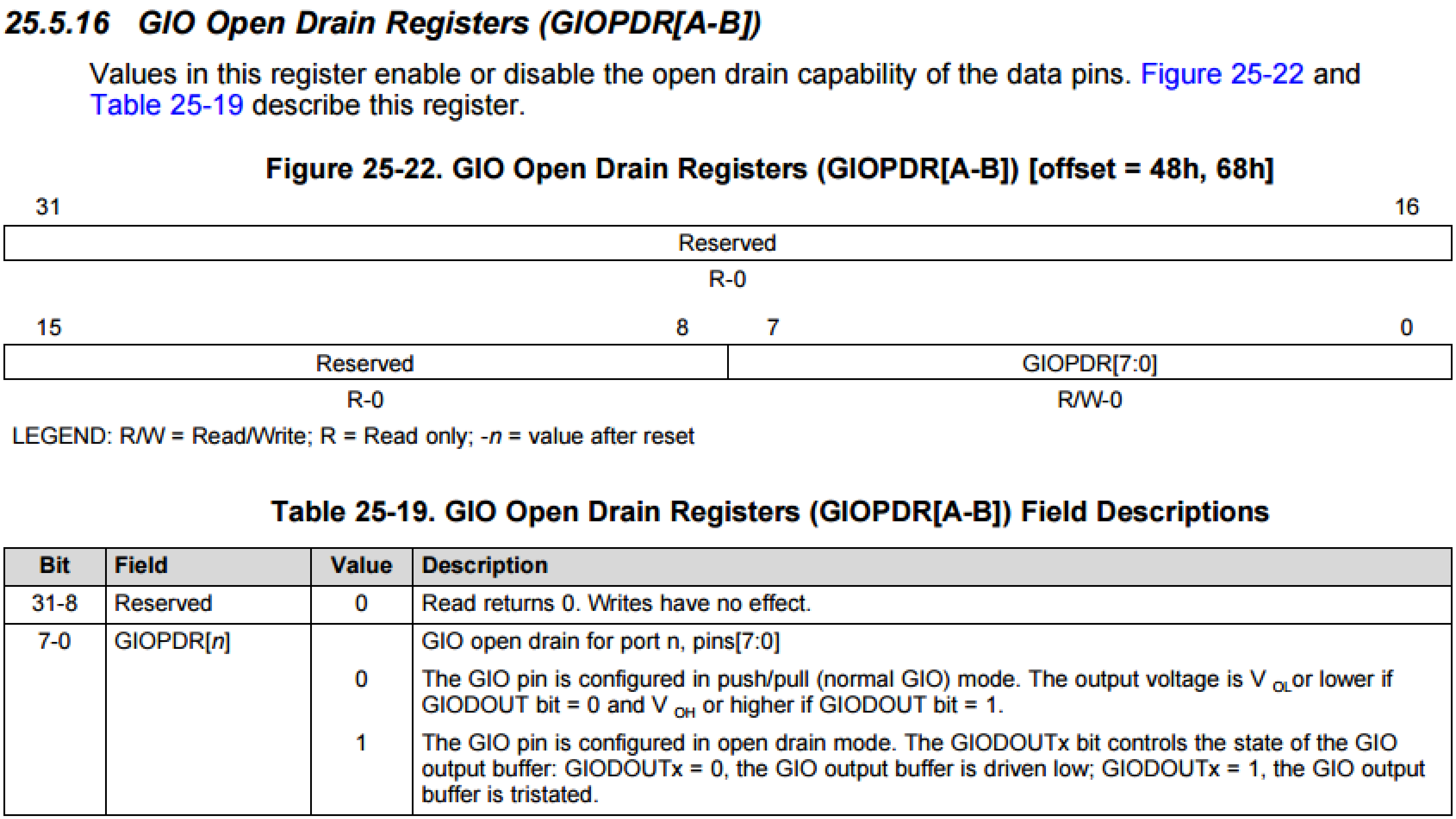
15	8	7	0
Reserved		GIODIR[7:0]	
R-0		R/W-0	

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	Read returns 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.

GIOA[4] 이 출력으로 설정되었다.



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]

31–16
Reserved
R-0

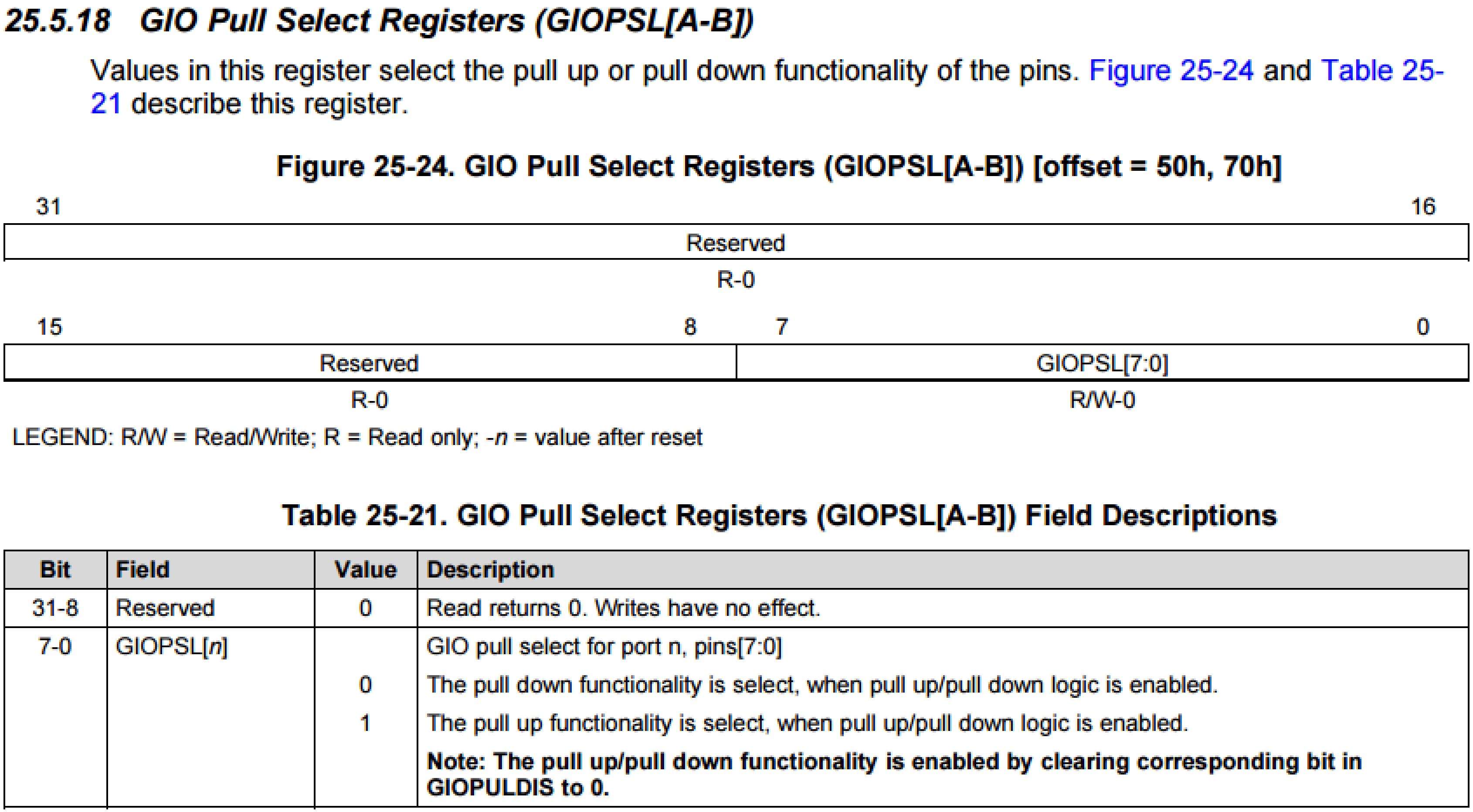
15–8	7–0
Reserved	GIOPDR[7:0]
R-0	R/W-0

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	Read returns 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.

0 인 경우 Push/Pull 방식으로 설정되며 1 인 경우 Open Drain 방식으로 설정된다.



25.5.18 *GIO Pull Select Registers (GIOPSL[A-B])*

Values in this register select the pull up or pull down functionality of the pins. Figure 25-24 and Table 25-21 describe this register.

Figure 25-24. GIO Pull Select Registers (GIOPSL[A-B]) [offset = 50h, 70h]

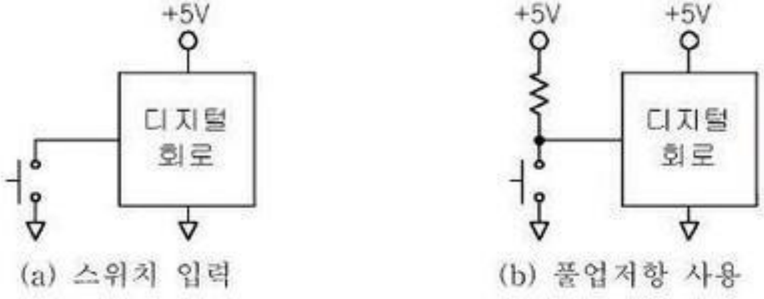
31–16	
Reserved	
R-0	
15–8	**7–0**
Reserved	GIOPSL[7:0]
R-0	R/W-0

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

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

Bit	Field	Value	Description
31-8	Reserved	0	Read returns 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.

GIOA[4] 쪽을 Pull Up 으로 설정한다.

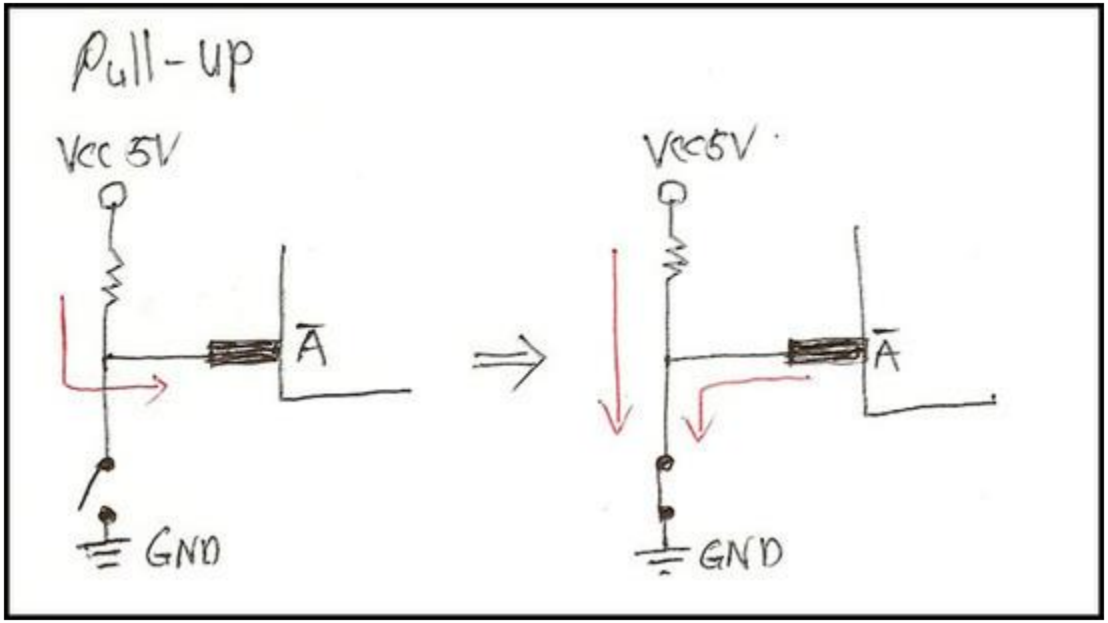


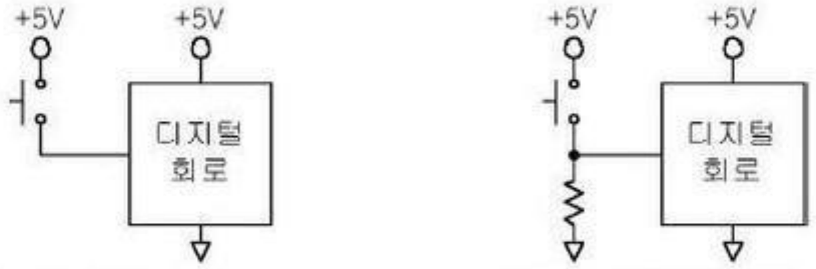
(a) 스위치 입력

(b) 풀업저항 사용

<그림 1> L 스위치 입력과 풀업 저항

스위치	ON	OFF
(a)그림	0V(Low)	Floating
(b)그림	0V(Low)	+5V(High)

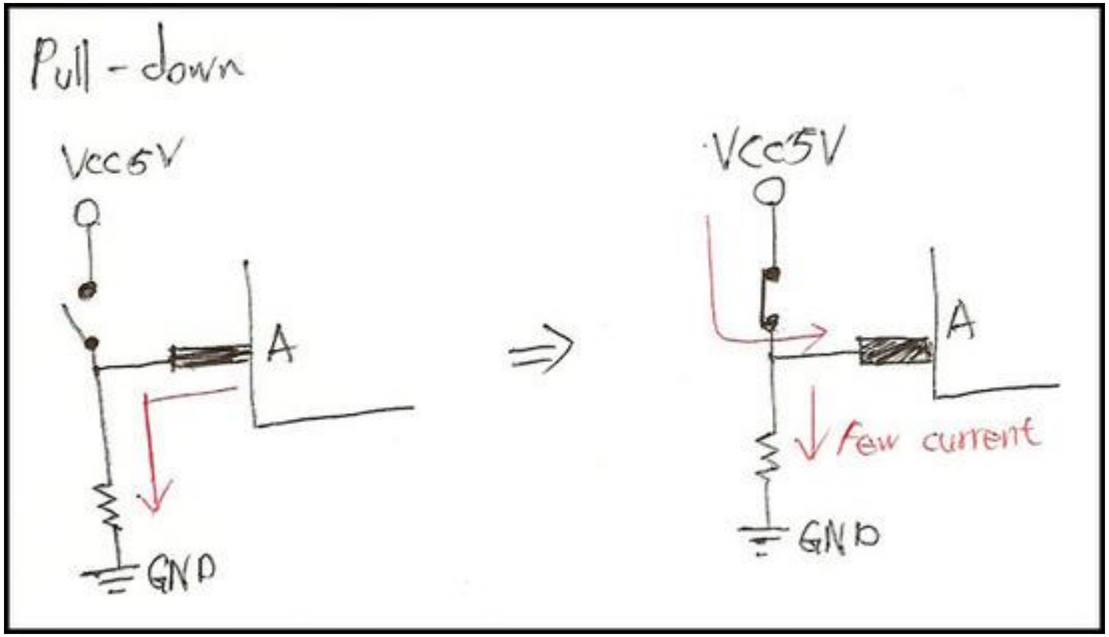




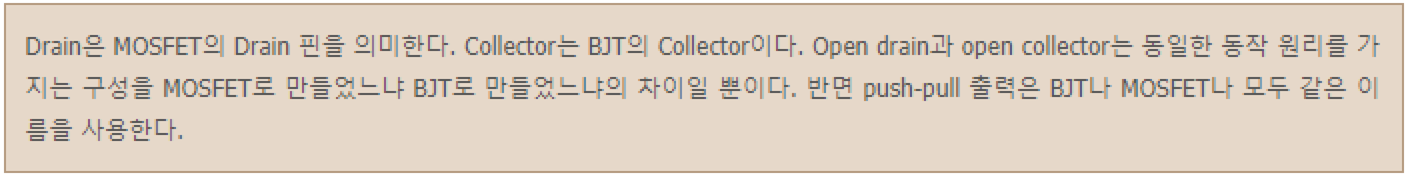
(a) 스위치 입력 (b) 풀다운저항 사용

<그림 2> H 스위치 입력과 풀다운 저항

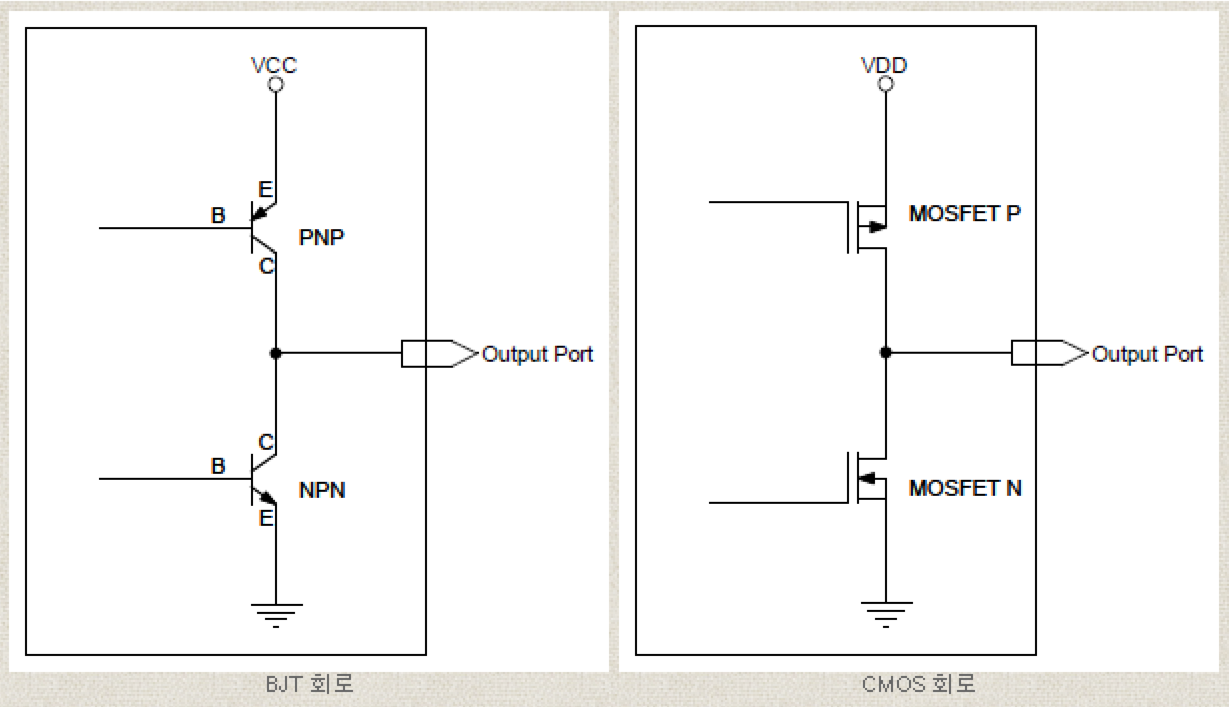
스위치	ON	OFF
(a)그림	+5V(High)	Floating
(b)그림	+5V(High)	0V(Low)



출처: http://irmus.tistory.com/76



Drain은 MOSFET의 Drain 핀을 의미한다. Collector는 BJT의 Collector이다. Open drain과 open collector는 동일한 동작 원리를 가지는 구성을 MOSFET로 만들었느냐 BJT로 만들었느냐의 차이일 뿐이다. 반면 push-pull 출력은 BJT나 MOSFET나 모두 같은 이름을 사용한다.

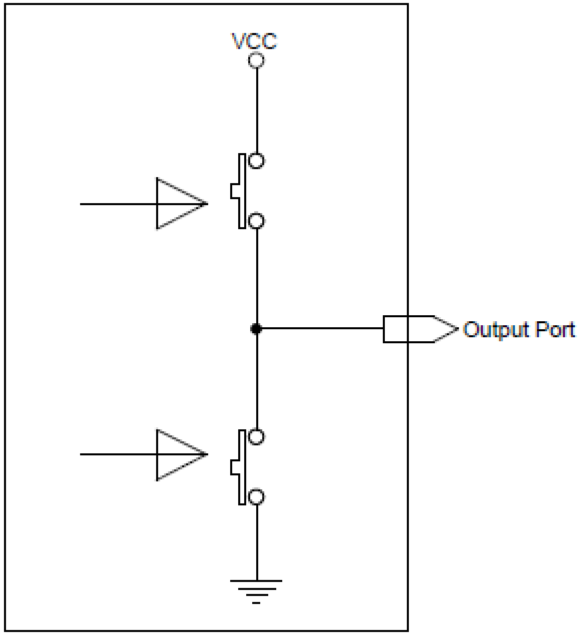


BJT 회로

CMOS 회로

위 그림이 push-pull 출력단을 간단하게 그려본 것이다.
push-pull 출력 포트는 2개의 TR로 구성된다. 아래쪽의 NPN TR(N-MOS)이 pull을 담당하고, 위쪽의 PNP TR(P-MOS)이 push를 담당한다. 단어의 의미를 생각해 보면 쉽게 이해할 수 있다. NPN TR은 GND쪽으로 끌어 당기는 역할을 하고, PNP TR은 VCC쪽으로 밀어 올리는 역할을 한다.

로직 회로에서는 TR을 switch로 사용한다는 것을 생각해 보면 쉽게 이해할 수 있다.

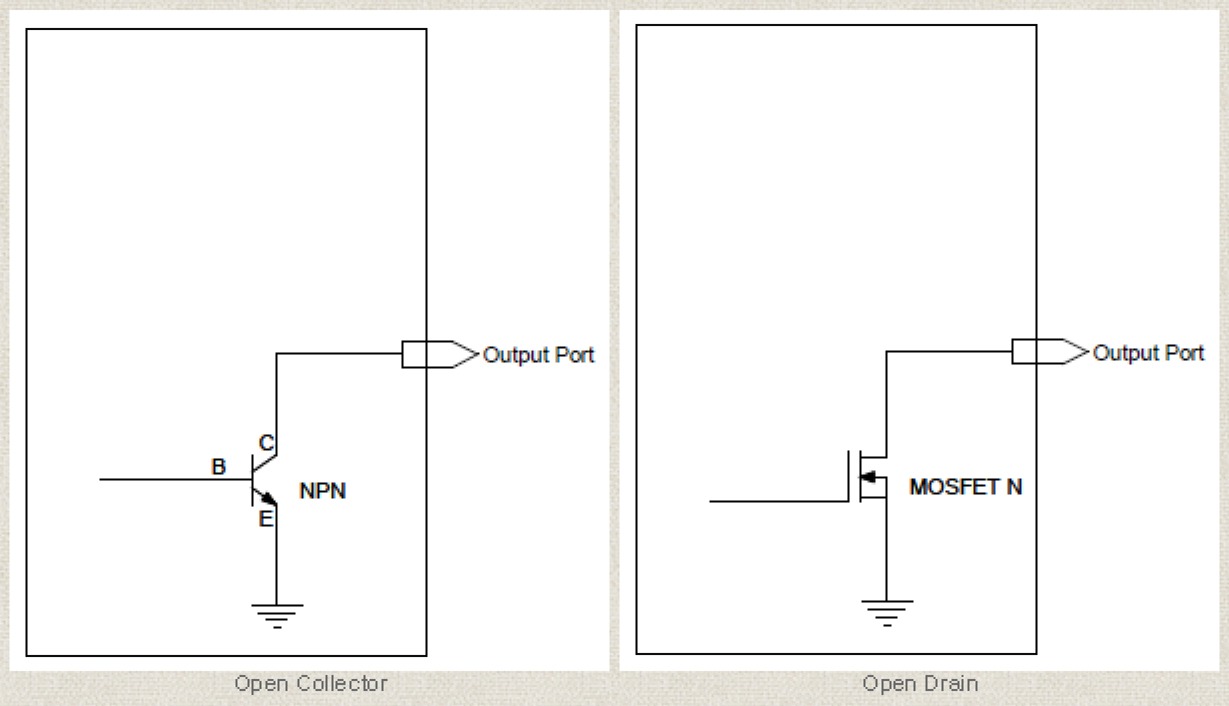


스위치로 치환해보자!

두개의 스위치 중 아래쪽 스위치를 ON 시키면 output port는 GND와 바로 연결된다. 즉, Output port를 GND쪽으로 끌어내린 것으로 0V가 출력된다. 반면 위쪽 스위치를 ON 시키면 VCC와 연결되어 high가 출력된다. VCC쪽으로 밀어 올린 것이다.

Push-pull 출력단은 그 자체로 동작한다. 이게 무슨 말이냐 하면... Output port에 아무것도 연결되어 있지 않더라도 output port의 전압이 의도한 바 대로 움직인다는 뜻이다. Low를 출력할 때엔 0V가 출력되고, high를 출력하면 VCC혹은 VDD가 출력된다. Output port에 아무것도 연결하지 말고 멀티메터나 오실로스코프로 관찰해 보면 출력 전압이 움직이는 것을 볼 수 있다.

Open collector 혹은 open drain은 이와 다르다.



Open Collector

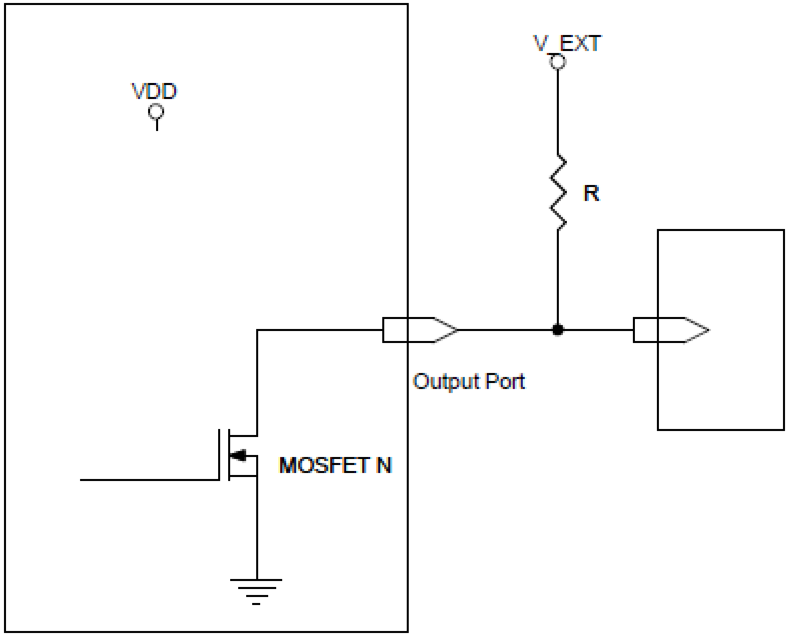
Open Drain

이름 그대로 collector 혹은 drain 핀이 외부로 노출(open)되어 있다. Push-pull 출력단과 달리 밀어올려주는(PUSH) 위쪽 절반이 없다. 더군다나 VCC나 VDD와도 전혀 연결되어 있지 않다. 반쪽짜리 미완성 회로인 것이다.

다시 스위치 모델을 생각하면서 생각해 보자. NPN TR이나 N-MOS가 ON 되었을 경우에는 Output port가 GND로 당겨지므로 0V 출력이 나가게 된다. 이 반쪽은 잘 동작한다(엄밀히 따지면 또 그렇지도 않지만 설명을 위해^^). 반면 NPN TR이나 N-MOS가 OFF되었을 경우 Output port는 그냥 무주공산으로 떠버리게 된다. 소위 Unknown state가 된다. 0V도 아니고 VCC나 VDD도 아닌 상태.

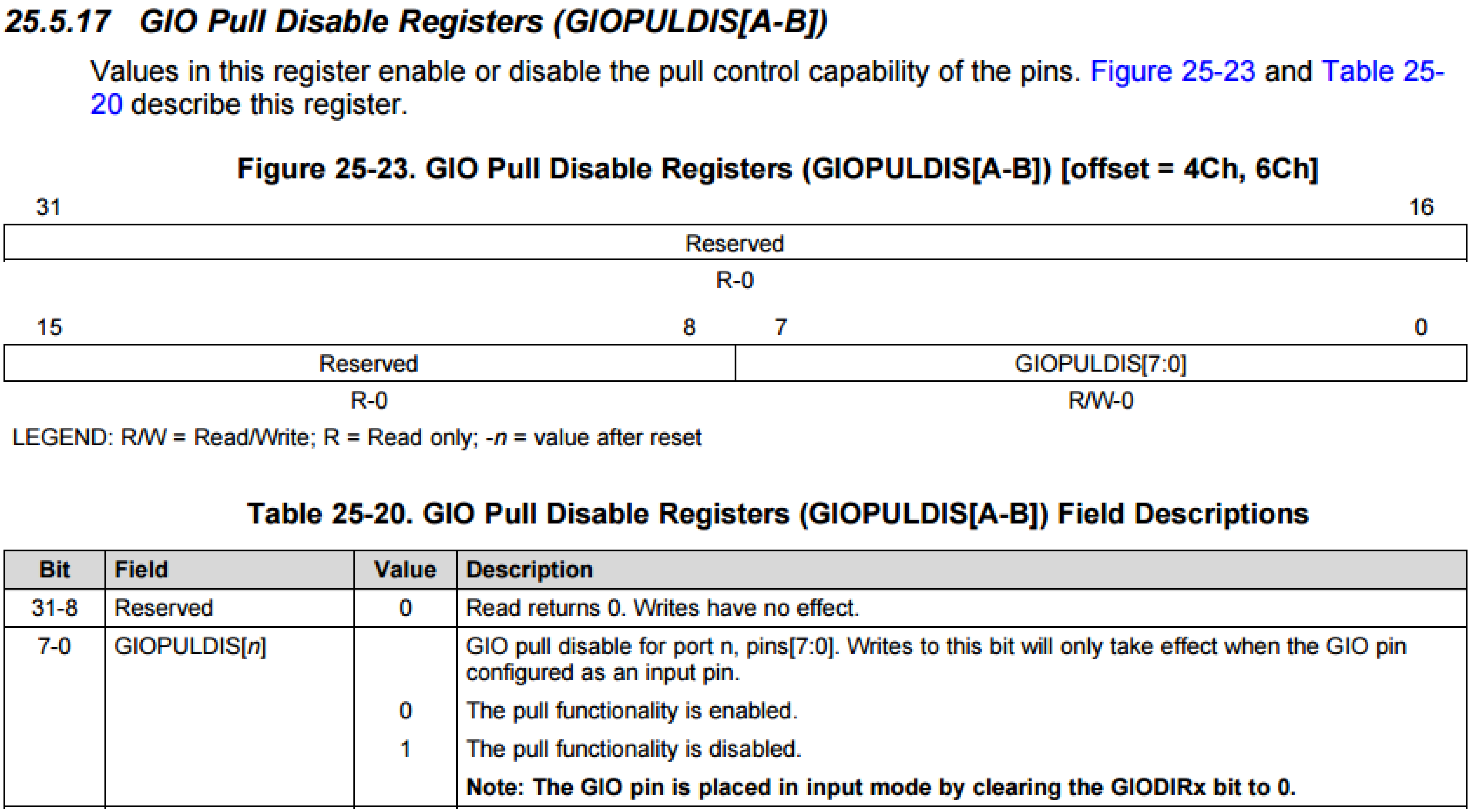
Open drain 출력단은 말 그대로 미완성 회로다. 이 상태 만으로는 제대로 동작하지 않는다. 외부에 부가 회로가 필요하다. 그러면 왜 이렇게 사용하는 것일까? 그것은 바로 입맛에 맞게 알아서 꾸며 쓰기 위해서이다.
Open drain 출력단이 필요한 경우는 여러가지가 있다. 대표적인 예로 level converter(level shifer), bus 구성 등이 있다.

Level converter로 사용하는 예를 살펴보자



VDD != V_EXT

위 그림에서 open drain 출력의 Output Port를 오른쪽에 있는 다른 장치의 입력 포트에 연결하려고 한다. 간단하게 왼쪽의 큰 상자를 MCU라 생각하고 오른쪽의 작은 상자를 센서 칩이라 생각해 보자. MCU가 사용하는 VDD는 3.3V이다. 센서 칩이 사용하는 전원(V_EXT)는 5V다. 이처럼 칩들이 사용하는 전원 전압이 다른 경우 바로 연결하면 문제가 발생할 수 있다. 이를 해결하는 한가지 수단으로 open drain 혹은 open collector가 사용될 수 있다. 위 회로처럼 외부에 풀업 저항(pull-up resisitor)을 하나 달아주게 되면 level shifter가 구성된다.
MCU 내부의 N-MOS 스위치가 ON되었을 경우 Output Port는 0V로 당겨지므로(PULL) low 출력이 나가게 된다. 이 경우는 push-pull 출력단과 동일하다. 반면 N-MOS 스위치가 OFF 되었을 경우 Output port가 unknown 상태가 되는 대신 외부 풀업저항에 의해 V_EXT로 묶여 올라가게 되는 것이다. 이것을 소극적인 push라고 생각하면 여러가지로 도움이 된다. 반면 push-pull 출력단의 위쪽 스위치는 적극적인 push인 셈이다.



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]

31 … 16
Reserved
R-0

15 … 8	7 … 0
Reserved	GIOPULDIS[7:0]
R-0	R/W-0

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	Read returns 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.

Pull Up 혹은 Pull Down 을 활성화시킨다.

25.5.3 GIO Interrupt Polarity Register (GIOPOL)

The GIOPOL register controls the polarity - rising edge (low to high) or falling edge (high to low) - that sets the flag. To ensure recognition of the signal as an edge, the signal must maintain the new level for at least one VCLK cycle. When the device is in low power mode, the interrupts are no longer triggered by an edge, but instead by a level. Therefore, in low power mode, the GIOPOL register controls the ***level***, high or low, which will trigger the interrupt. Figure 25-7 and Table 25-4 describe this register.

Figure 25-7. GIO Interrupt Polarity Register (GIOPOL) [offset = 0Ch]

31 24	23 16
GIOPOL 3	GIOPOL 2
R/W-0	R/W-0
15 8	7 0
GIOPOL 1	GIOPOL 0
R/W-0	R/W-0

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

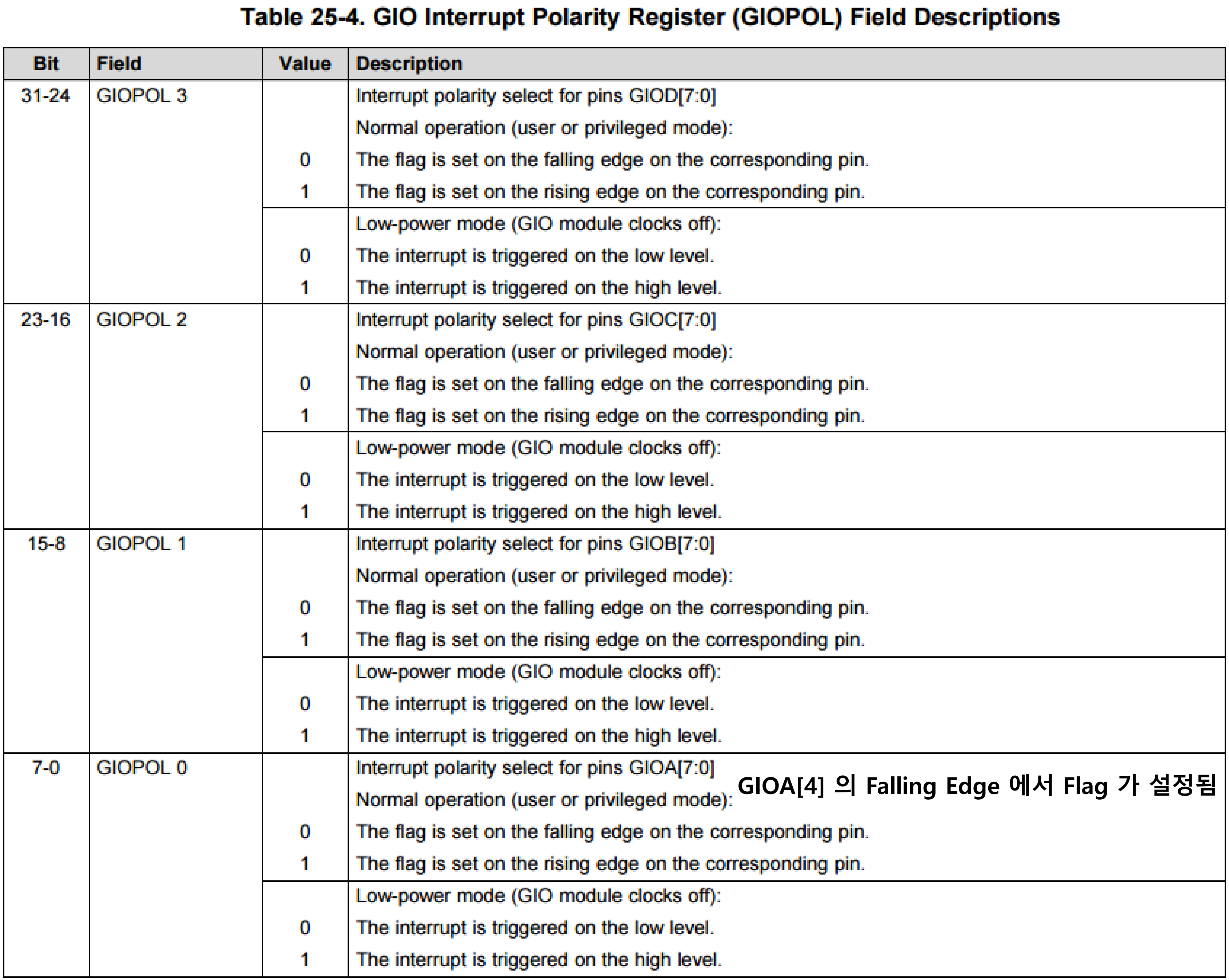


Table 25-4. GIO Interrupt Polarity Register (GIOPOL) Field Descriptions

Bit	Field	Value	Description
31-24	GIOPOL 3		Interrupt polarity select for pins GIOD[7:0]
			Normal operation (user or privileged mode):
		0	The flag is set on the falling edge on the corresponding pin.
		1	The flag is set on the rising edge on the corresponding pin.
			Low-power mode (GIO module clocks off):
		0	The interrupt is triggered on the low level.
		1	The interrupt is triggered on the high level.
23-16	GIOPOL 2		Interrupt polarity select for pins GIOC[7:0]
			Normal operation (user or privileged mode):
		0	The flag is set on the falling edge on the corresponding pin.
		1	The flag is set on the rising edge on the corresponding pin.
			Low-power mode (GIO module clocks off):
		0	The interrupt is triggered on the low level.
		1	The interrupt is triggered on the high level.
15-8	GIOPOL 1		Interrupt polarity select for pins GIOB[7:0]
			Normal operation (user or privileged mode):
		0	The flag is set on the falling edge on the corresponding pin.
		1	The flag is set on the rising edge on the corresponding pin.
			Low-power mode (GIO module clocks off):
		0	The interrupt is triggered on the low level.
		1	The interrupt is triggered on the high level.
7-0	GIOPOL 0		Interrupt polarity select for pins GIOA[7:0]
			Normal operation (user or privileged mode):
		0	The flag is set on the falling edge on the corresponding pin.
		1	The flag is set on the rising edge on the corresponding pin.
			Low-power mode (GIO module clocks off):
		0	The interrupt is triggered on the low level.
		1	The interrupt is triggered on the high level.

GIOA[4] 의 Falling Edge 에서 Flag 가 설정됨

25.5.5.1 GIOLVLSET Register

The GIOLVLSET register is used to configure an interrupt as a high-level interrupt going to the VIM. An interrupt can be configured as a high-level interrupt by writing a 1 into the corresponding bit of the GIOLVLSET register. Writing a 0 has no effect. Figure 25-10 and Table 25-7 describe this register.

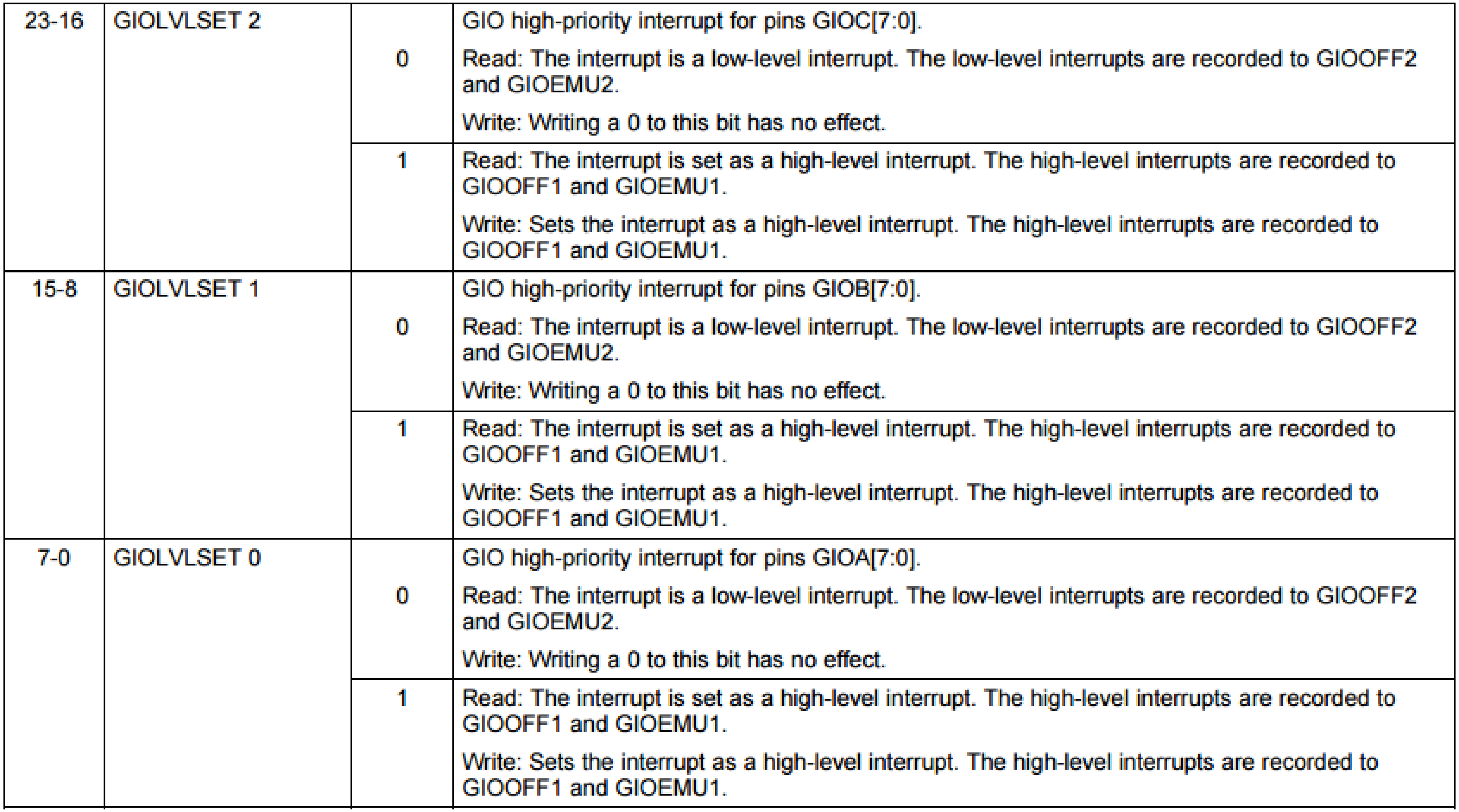
Figure 25-10. GIO Interrupt Priority Register (GIOLVLSET) [offset = 18h]

<table>
<tr><td colspan="2">31</td><td colspan="2">16</td></tr>
<tr><td colspan="2">GIOLVLSET 3</td><td colspan="2">GIOLVLSET 2</td></tr>
<tr><td colspan="2">R/W-0</td><td colspan="2">R/W-0</td></tr>
<tr><td>15</td><td>8</td><td>7</td><td>0</td></tr>
<tr><td colspan="2">GIOLVLSET 1</td><td colspan="2">GIOLVLSET 0</td></tr>
<tr><td colspan="2">R/W-0</td><td colspan="2">R/W-0</td></tr>
</table>

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

Table 25-7. GIO Interrupt Priority Register (GIOLVLSET) Field Descriptions

Bit	Field	Value	Description
31-24	GIOLVLSET 3		GIO high-priority interrupt for pins GIOD[7:0].
		0	Read: The interrupt is a low-level interrupt. The low-level interrupts are recorded to GIOOFF2 and GIOEMU2. 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 high-level interrupt. The high-level interrupts are recorded to GIOOFF1 and GIOEMU1.



23-16	GIOLVLSET 2	0	GIO high-priority interrupt for pins GIOC[7:0]. Read: The interrupt is a low-level interrupt. The low-level interrupts are recorded to GIOOFF2 and GIOEMU2. 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 high-level interrupt. The high-level interrupts are recorded to GIOOFF1 and GIOEMU1.
15-8	GIOLVLSET 1	0	GIO high-priority interrupt for pins GIOB[7:0]. Read: The interrupt is a low-level interrupt. The low-level interrupts are recorded to GIOOFF2 and GIOEMU2. 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 high-level interrupt. The high-level interrupts are recorded to GIOOFF1 and GIOEMU1.
7-0	GIOLVLSET 0	0	GIO high-priority interrupt for pins GIOA[7:0]. Read: The interrupt is a low-level interrupt. The low-level interrupts are recorded to GIOOFF2 and GIOEMU2. 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 high-level interrupt. The high-level interrupts are recorded to GIOOFF1 and GIOEMU1.

Low Level Interrupt 가 발생하며 GIOOFF2 와 GIOEMU2 를 통해 살펴볼 수 있다.

25.5.6 GIO Interrupt Flag Register (GIOFLG)

The GIOFLG register contains flags indicating that the transition edge (as set in GIOINTDET and GIOPOL) has occurred. The flag is also cleared by reading the appropriate interrupt offset register (GIOOFF1 or GIOOFF2). Figure 25-12 and Table 25-9 describe this register.

Figure 25-12. GIO Interrupt Flag Register (GIOFLG) [offset = 20h]

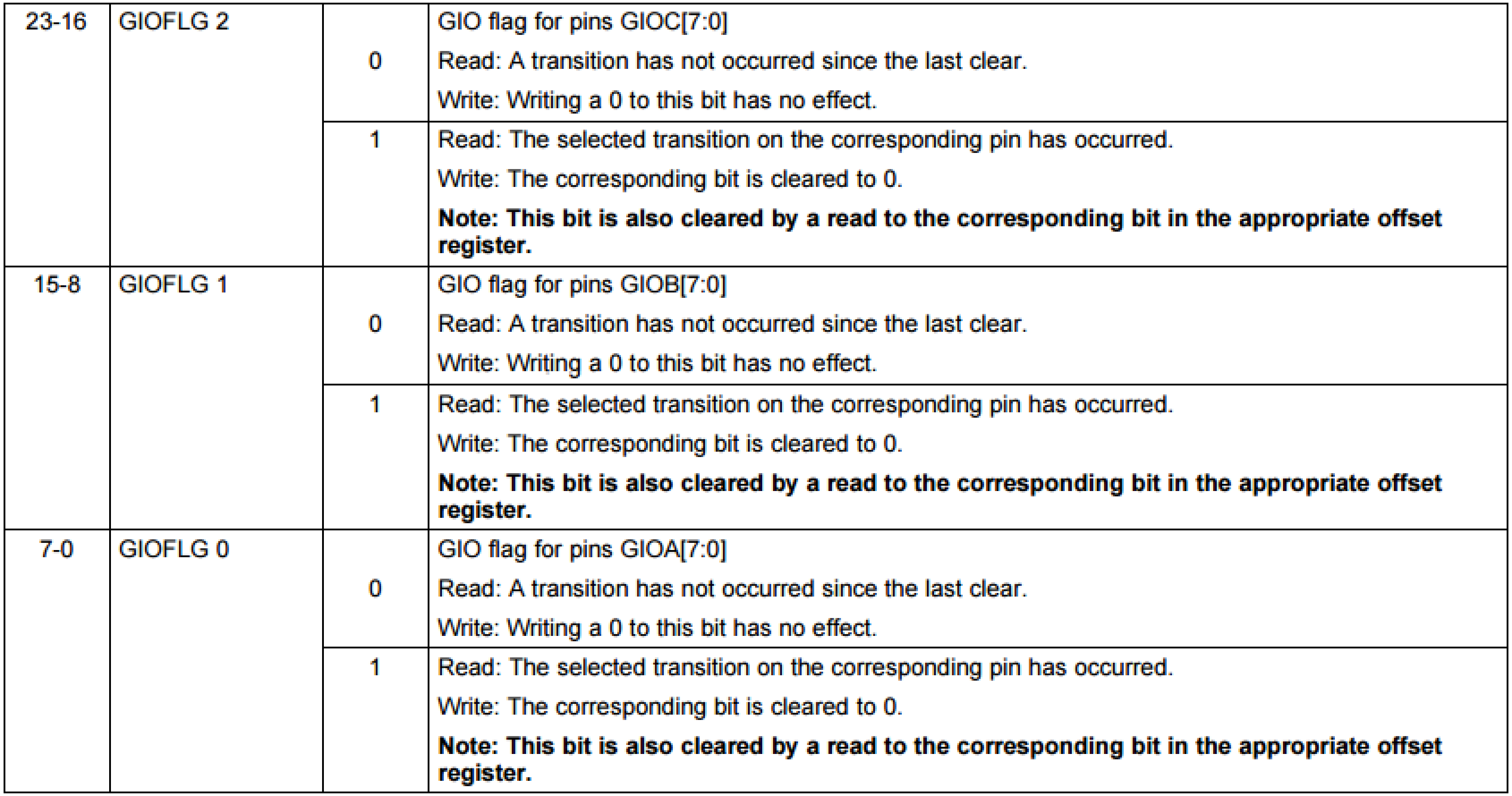
31 24	23 16
GIOFLG 3	GIOFLG 2
R/WC-0	R/WC-0

15 8	7 0
GIOFLG 1	GIOFLG 0
R/WC-0	R/WC-0

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

Table 25-9. GIO Interrupt Flag Register (GIOFLG) Field Descriptions

Bit	Field	Value	Description
31-24	GIOFLG 3		GIO flag for pins GIOD[7:0]
		0	Read: A transition has not occurred since the last clear. Write: Writing a 0 to this bit has no effect.
		1	Read: The selected transition on the corresponding pin has occurred. Write: The corresponding bit is cleared to 0. **Note: This bit is also cleared by a read to the corresponding bit in the appropriate offset register.**



23-16	GIOFLG 2	0	GIO flag for pins GIOC[7:0] Read: A transition has not occurred since the last clear. Write: Writing a 0 to this bit has no effect.
		1	Read: The selected transition on the corresponding pin has occurred. Write: The corresponding bit is cleared to 0. **Note: This bit is also cleared by a read to the corresponding bit in the appropriate offset register.**
15-8	GIOFLG 1	0	GIO flag for pins GIOB[7:0] Read: A transition has not occurred since the last clear. Write: Writing a 0 to this bit has no effect.
		1	Read: The selected transition on the corresponding pin has occurred. Write: The corresponding bit is cleared to 0. **Note: This bit is also cleared by a read to the corresponding bit in the appropriate offset register.**
7-0	GIOFLG 0	0	GIO flag for pins GIOA[7:0] Read: A transition has not occurred since the last clear. Write: Writing a 0 to this bit has no effect.
		1	Read: The selected transition on the corresponding pin has occurred. Write: The corresponding bit is cleared to 0. **Note: This bit is also cleared by a read to the corresponding bit in the appropriate offset register.**

GIOA[4] 에서 우리가 지정한 전이(Falling Edge)가 발생했음을 파악할 수 있다.

25.5.4 GIO Interrupt Enable Registers (GIOENASET and GIOENACLR)

The GIOENASET and GIOENACLR registers control which interrupt-capable pins are actually configured as interrupts. If the interrupt is enabled, the rising or falling or both edges on the selected pin lead to an interrupt.

25.5.4.1 GIOENASET Register

Figure 25-8 and Table 25-5 describe this register.

NOTE: Enabling Interrupt at the Device Level

GIO can be mapped to two different device level interrupts, the GIO high-level (level A) and low-level (level B) interrupts through programming registers GIOLVLSET and GIOLVLCLR. The corresponding bit to the mapped device level interrupt must be set within the vectored interrupt manager (VIM) in the interrupt mask register (REQMASK) to enable the appropriate interrupts. Additionally, the ARM CPU (CPSR bit 7 or 6) must be cleared to recognize interrupt requests (IRQ/FIQ).

Figure 25-8. GIO Interrupt Enable Set Register (GIOENASET) [offset = 10h]

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

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

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

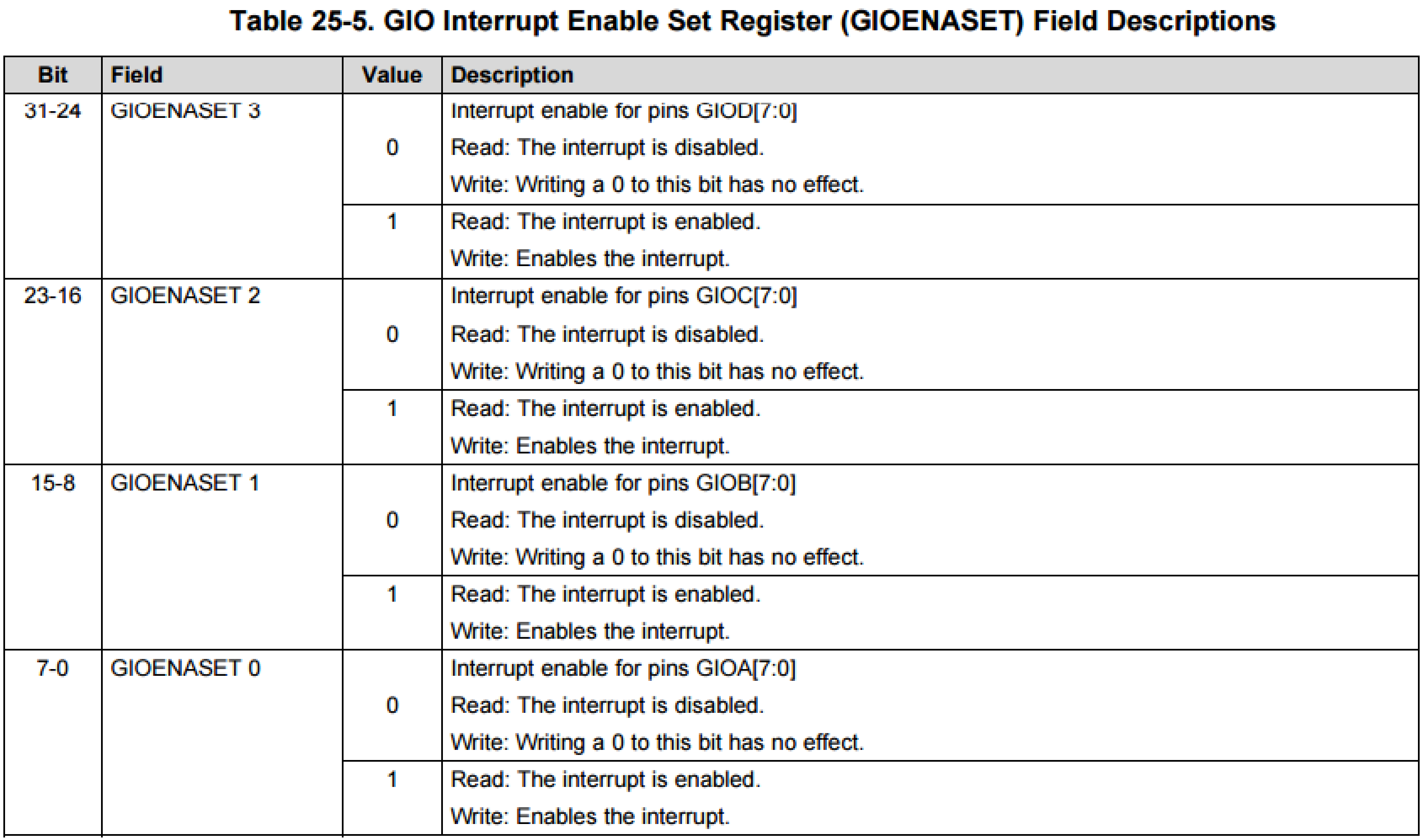
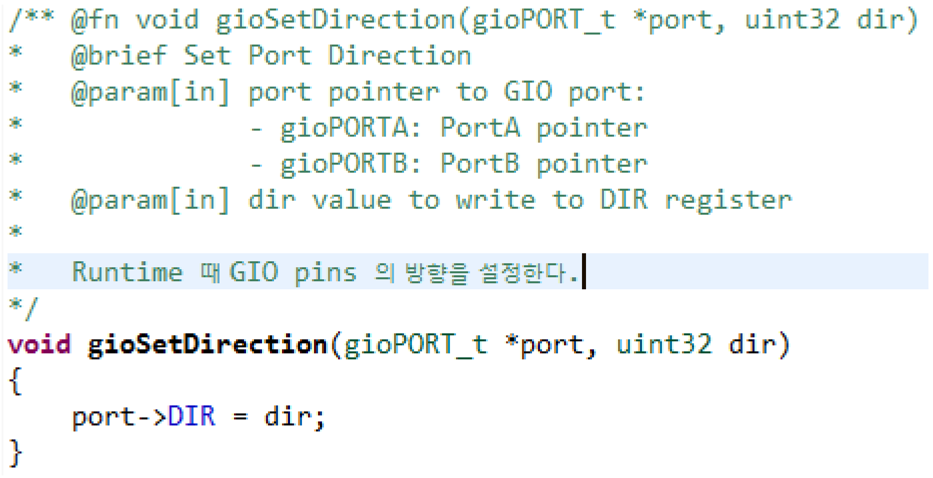


Table 25-5. GIO Interrupt Enable Set Register (GIOENASET) Field Descriptions

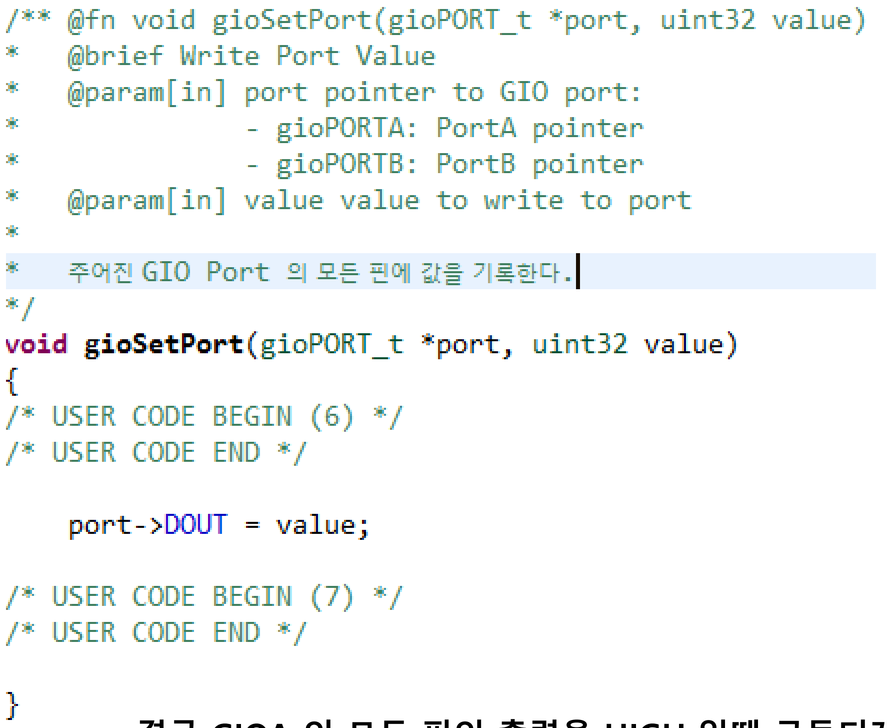
Bit	Field	Value	Description
31-24	GIOENASET 3		Interrupt enable for pins GIOD[7:0]
		0	Read: The interrupt is disabled. Write: Writing a 0 to this bit has no effect.
		1	Read: The interrupt is enabled. Write: Enables the interrupt.
23-16	GIOENASET 2		Interrupt enable for pins GIOC[7:0]
		0	Read: The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

인터럽트를 비활성화한다.



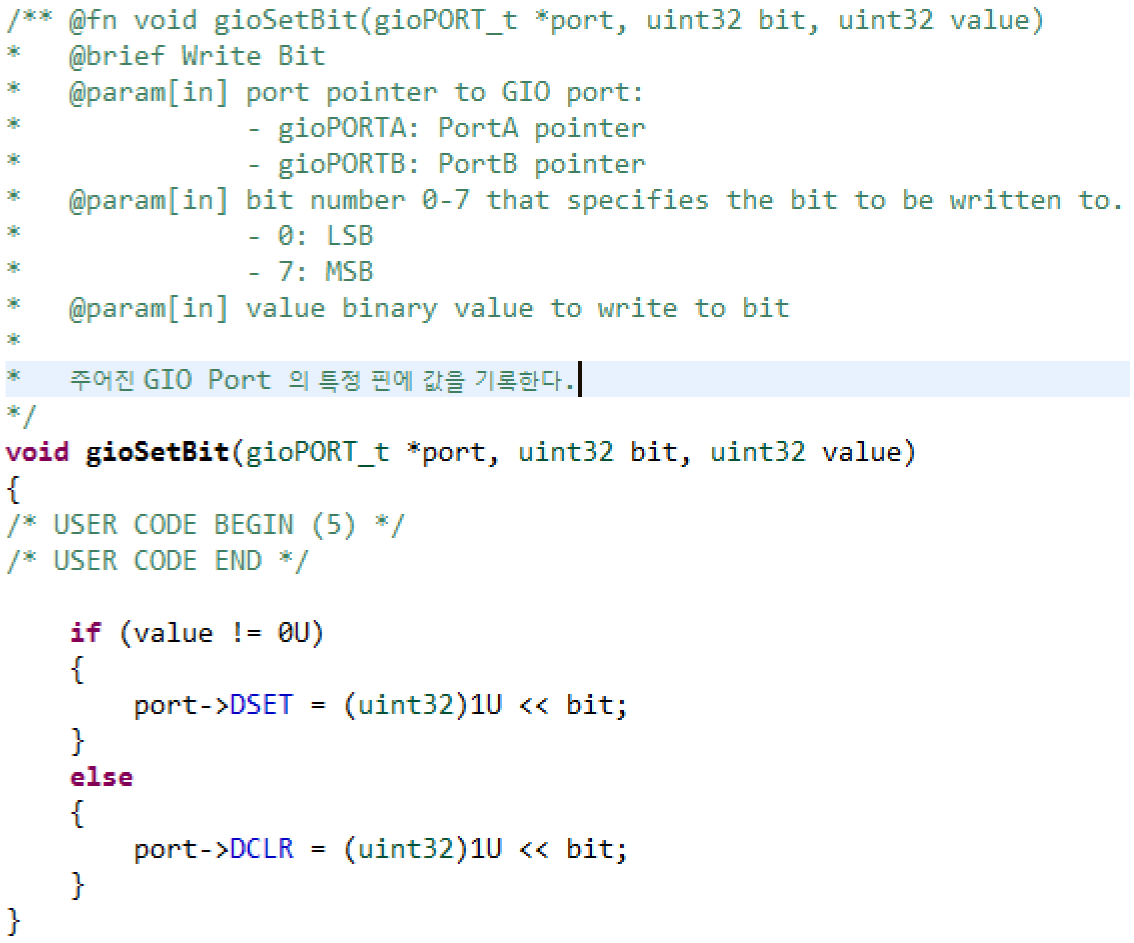
```
/** @fn void gioSetDirection(gioPORT_t *port, uint32 dir)
*   @brief Set Port Direction
*   @param[in] port pointer to GIO port:
*               - gioPORTA: PortA pointer
*               - gioPORTB: PortB pointer
*   @param[in] dir value to write to DIR register
*
*   Runtime 때 GIO pins 의 방향을 설정한다.
*/
void gioSetDirection(gioPORT_t *port, uint32 dir)
{
    port->DIR = dir;
}
```

결국 GIOA 의 모든 핀을 출력으로 만든다.



```
/** @fn void gioSetPort(gioPORT_t *port, uint32 value)
*   @brief Write Port Value
*   @param[in] port pointer to GIO port:
*               - gioPORTA: PortA pointer
*               - gioPORTB: PortB pointer
*   @param[in] value value to write to port
*
*   주어진 GIO Port 의 모든 핀에 값을 기록한다.
*/
void gioSetPort(gioPORT_t *port, uint32 value)
{
/* USER CODE BEGIN (6) */
/* USER CODE END */

    port->DOUT = value;

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

}
```

결국 GIOA 의 모든 핀의 출력을 HIGH 일때 구동되게 만든다.



```
/** @fn void gioSetBit(gioPORT_t *port, uint32 bit, uint32 value)
*   @brief Write Bit
*   @param[in] port pointer to GIO port:
*               - gioPORTA: PortA pointer
*               - gioPORTB: PortB pointer
*   @param[in] bit number 0-7 that specifies the bit to be written to.
*               - 0: LSB
*               - 7: MSB
*   @param[in] value binary value to write to bit
*
*   주어진 GIO Port 의 특정 핀에 값을 기록한다.
*/
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;
    }
}
```

GIOA[4] 에 HIGH 신호를 준다.
결국 LED 에 불이 켜진다.

25.5.14 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. The contents of this register reflect the contents of GIODOUT. Figure 25-20 and Table 25-17 describe this register.

Figure 25-20. GIO Data Set Registers (GIODSET[A-B]) [offset = 40h, 60h]

31	16
Reserved	
R-0	

15	8	7	0
Reserved		GIODSET[7:0]	
R-0		R/W-0	

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

Table 25-17. GIO Data Set Registers (GIODSET[A-B]) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Read returns 0. Writes have no effect.
7-0	GIODSET[*n*]		GIO data set for port n, pins[7:0]. This bit drives the output of GIO pin high.
		0	Write: Writing a 0 has no effect.
		1	Write: The corresponding GIO pin is driven to logic high (1).
			Note: The current logic state of the GIODOUT bit will also be displayed by this bit.
			Note: GIO pin is placed in output mode by setting the GIODIRx bit to 1.