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

#47

강사 : Innova Lee(이 상훈)

학생 : 김 시윤

1.배운내용 복습.

ARM LED 제어

External_LED main

```
#include “HL_sys_common.h”
#include “HL_gio.h”

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

        return 0;
}
```

init 함수

```
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 */

    /** - 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 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 */
```

```
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 */
```

gioREG->GCR0 = 1;

25.5.1 GIO Global Control Register (GIOGCR0)

The GIOGCR0 register contains one bit that controls the module reset status. Writing a 0 to this bit puts the module in a reset state. After system reset, this bit must be set to 1 before configuring any other register of the GIO 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	Reads return 0. Writes have no effect.
0	RESET		GIO reset.
		0	The GIO is in reset state.
		1	The GIO is operating normally.

GIOGCR0 레지스터는 0번째 비트만을 사용한다. 이 비트가 0이면 모듈을 reset 한다. 다른 gio레지스터를 사용하기 위해서는 이 레지스터를 1로 셋팅하여야한다. 여기서는 다른 레지스터를 사용하여 gio를 제어해야 하기 때문에 1로 셋팅 한다.

gioREG->ENACLR=0xFF;

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.

0x000000ff 이므로 GIOENCLR0 이 1로 셋팅
1이면 인터럽트를 무시한다는 설정이다.
다른 GIOENCLRn들은 0이기 때문에 아무 effect가 없다고 되어있다.

gioREG->LVLCLR = 0xFF;

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	15 – 8	7 – 0
GIOLVLCLR 3	GIOLVLCLR 2	GIOLVLCLR 1	GIOLVLCLR 0
R/W-0	R/W-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.
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.

인터럽트를 로우 레벨 인터럽트로 설정합니다. 저수준 인터럽트는 GIOOFF2 및 GIOEMU2에 기록됩니다.

GIODOUT 레지스터는 output을 설정해줍니다.
GIODIR 레지스터는 전류방향을 설정해줍니다.
GIOPSL 풀업저항 풀다운 저항을 설정해줍니다.(플로팅방지)

```
258 void gioSetDirection(gioPORT_t *port, uint32 dir)
259 {
260     port->DIR = dir;
261 }
262
```

gioSetDirection 함수에 모든값을 1로 셋팅하여 DIR로 넣어줍니다.
근데 이럴필요 없는게 init함수에서 이미 셋팅을 해주었기 때문에 불필요한 함수입니다. 또한 우리가 사용할 포트만 켜줘도 동작이 가능하기 때문에 이렇게 함수를 쓰는건 쓸모없는 소스코드 낭비였습니다.

밑에 PORT 설정도 마찬가지로 DOUT를 모두 1로 셋팅하였습니다.
하지만 우리가 사용할 gio PORTA 의 4번만 1로 셋팅해도 동작하는 것을 확인하였습니다.