

전자종합설계2 실험 보고서 1

전자공학도의 윤리 강령 (IEEE Code of Ethics)

(출처: <http://www.ieee.org>)

나는 전자공학도로서, 전자공학이 전 세계 인류의 삶에 끼치는 심대한 영향을 인식하여 우리의 직업, 동료와 사회에 대한 나의 의무를 짐에 있어 최고의 윤리적, 전문적 행위를 수행할 것을 다짐하면서, 다음에 동의한다.

1. **공중의 안전, 건강 복리에 대한 책임:** 공중의 안전, 건강, 복리에 부합하는 결정을 할 책임을 질 것이며, 공중 또는 환경을 위협할 수 있는 요인을 신속히 공개한다.
2. **지위 남용 배제:** 실존하거나 예기되는 이해 상충을 가능한 한 피하며, 실제로 이해가 상충할 때에는 이를 이해 관련 당사자에게 알린다. (이해 상충: conflicts of interest, 공적인 지위를 사적 이익에 남용할 가능성)
3. **정직성:** 청구 또는 견적을 함에 있어 입수 가능한 자료에 근거하여 정직하고 현실적으로 한다.
4. **뇌물 수수 금지:** 어떠한 형태의 뇌물도 거절한다.
5. **기술의 영향력 이해:** 기술과 기술의 적절한 응용 및 잠재적 영향에 대한 이해를 높인다.
6. **자기계발 및 책무성:** 기술적 능력을 유지, 증진하며, 훈련 또는 경험을 통하여 자격이 있는 경우이거나 관련 한계를 전부 밝힌 뒤에만 타인을 위한 기술 업무를 수행한다.
7. **엔지니어로서의 자세:** 기술상의 업무에 대한 솔직한 비평을 구하고, 수용하고, 제공하며, 오류를 인정하고 수정하며, 타인의 기여를 적절히 인정한다.
8. **차별 안하기:** 인종, 종교, 성별, 장애, 연령, 출신국 등의 요인에 관계없이 모든 사람을 공평하게 대한다.
9. **도덕성:** 허위 또는 악의적인 행위로 타인, 타인의 재산, 명예, 또는 취업에 해를 끼치지 않는다.
10. **동료애:** 동료와 협력자가 전문분야에서 발전하도록 도우며, 이 윤리 헌장을 준수하도록 지원한다.

위 IEEE 윤리헌장 정신에 입각하여 report를 작성하였음을 서약합니다.

학 부: 전자공학부

제출일: 2020.04.27.(월)

과목명: 융합캡스톤디자인2

교수명: 이교범 교수님

조 : 4 조

학 번: 201420820, 201623432

성 명: 이승복, 김희서

[실험 : GPIO Input]

1. 코드 분석

예제 1. GPIO01_Input

```
// 선행처리 지시
//-----
#include "DSP28x_Project.h"           // Device Headerfile and Examples Include File
//=====

//-----
// 시스템에서 사용할 전역 변수 선언
//-----
Uint16 Loop_cnt;
Uint16 SW1, SW2, SW3, SW4;
//=====
```

28X DSP의 기능을 제어하는 레지스터에 쉽게 접근하기 위한 헤더파일을 추가한다. 이후 코딩이 잘 시행되고 있는지 변화를 나타내는 Loop_cnt 함수와 스위치 상태를 나타내는 각각의 네 변수 SW1, SW2, SW3, SW4를 Uint16을 이용해서 선언해준다. Uint16으로 구성된 변수는 2바이트의 부호 없는 정수로 저장된다.

```
void main(void)
{
    //-----
    // Step 1. Disable Global Interrupt
    //-----
    DINT;
    //-----

    //-----
    // Step 2. 시스템 컨트롤 초기화:
    //-----
    InitSysCtrl();
    //-----
}
```

* **DINT**: 칩의 활성 여부를 결정하는 전역 인터럽트 스위치를 Off하고, 칩의 비정상적인 작동을 방지한다.

* **InitSysCtrl**: 시스템 컨트롤을 초기화한다. 이 함수가 하는 역할은 다음과 같다.

- (1) 시스템에 이상이 생기면 시스템을 리셋하는 Watchdog기능을 disable한다.
- (2) PLLCR 레지스터를 설정하여 적당한 SYSCLKOUT을 생성한다.
- (3) 프리스케일러를 설정하여 고속 주변장치 클럭과 저속 주변장치 클럭을 조정한다.
- (4) 클럭을 ON하여 주변장치에 공급한다.

```
// Step 3. GPIO 초기화
//-----
EALLOW;
GpioCtrlRegs.GPBMUX1.bit.GPI044 = 0;    // 핀 기능선택: GPI044
GpioCtrlRegs.GPBMUX1.bit.GPI045 = 0;    // 핀 기능선택: GPI045
GpioCtrlRegs.GPBMUX1.bit.GPI046 = 0;    // 핀 기능선택: GPI046
GpioCtrlRegs.GPBMUX1.bit.GPI047 = 0;    // 핀 기능선택: GPI047
GpioCtrlRegs.GPBDIR.bit.GPI044 = 0;     // GPI044 입출력 선택: Input
GpioCtrlRegs.GPBDIR.bit.GPI045 = 0;     // GPI045 입출력 선택: Input
GpioCtrlRegs.GPBDIR.bit.GPI046 = 0;     // GPI046 입출력 선택: Input
GpioCtrlRegs.GPBDIR.bit.GPI047 = 0;     // GPI047 입출력 선택: Input
EDIS;
//=====
```

* **EALLOW-EDIS**: DSP에서 일정 Protected 영역에 값을 쓰기 위해서는 Protect를 해지하고 다시 Protect를 해주는 작업이 필요하다. GpioCtrlRegs 그룹의 레지스터는 모두 Protected 영역에 있으므로 레지스터를 설정하

기 위해서는 EALLOW로 보호설정을 풀어주고, EDIS로 보호를 해주어야 한다.

Table 52. GPIO Port B MUX 1 (GPBMUX1) Register Field Descriptions

Bit	Field	Value	Description
31:30	GPIO47	00	Configure this pin as:
		01	GPIO 47 - general purpose I/O 47 (default)
		Reserved	
		10 or 11	XA7 - External interface (XINTF) address line 7 (O)
29:28	GPIO46	00	Configure this pin as:
		01	GPIO 46 - general purpose I/O 46 (default)
		Reserved	
		10 or 11	XA6 - External interface (XINTF) address line 6 (O)
27:26	GPIO45	00	Configure this pin as:
		01	GPIO 45 - general purpose I/O 45 (default)
		Reserved	
		10 or 11	XA5 - External interface (XINTF) address line 5 (O)
25:24	GPIO44	00	Configure this pin:
		01	GPIO 44 - general purpose I/O 44 (default)
		Reserved	
		10 or 11	XA4 - External interface (XINTF) address line 4 (O)

* **GPBMUX**: 핀의 역할을 정해주는 레지스터로, 0을 입력하면 핀을 GPIO로 사용하는 것이고, 1을 입력하면 핀을 GPIO가 아닌 다른 용도로 사용한다는 의미이다. 이번 예제에서는 모든 핀이 GPIO로 사용하기 때문에 0을 입력해주었다.

Table 63. GPIO Port B Direction (GPBDIR) Register Field Descriptions

Bits	Field	Value	Description ⁽¹⁾
31-0	GPIO63-GPIO32		Controls direction of GPIO pin when GPIO mode is selected. Reading the register returns the current value of the register setting
		0	Configures the GPIO pin as an input. (default)
		1	Configures the GPIO pin as an output

* **GPBDIR**: GPIO핀의 입출력을 선택하는 레지스터로, 0을 입력하면 GPIO핀을 입력으로 이용하고, 1을 입력하면 GPIO핀을 출력으로 이용한다는 의미이다. 이번 예제에서 GPIO핀은 스위치를 누르는 입력 역할로 동작하기 때문에 0을 입력해주었다.

```
// Step 4. Initialize Application Variables
//=====
SW1 = 0;
SW2 = 0;
SW3 = 0;
SW4 = 0;
Loop_cnt = 0;
//=====

//=====
// Enable global interrupts and higher priority real-time debug events:
//=====
EINT; // Enable Global interrupt INTM
ERTM; // Enable Global realtime interrupt DBGM
//=====
```

앞서 선언해준 변수 SW1, SW2, SW3, SW4, Loop_cnt의 값을 초기화해준다.

- * **EINT**: 앞의 DINT와 반대로 전역 인터럽트를 활성화 시킨다.
- * **ERTM**: DBGCM을 clear하여 디버그 이벤트를 Enable 시킨다.

```
// IDLE loop. Just sit and loop forever :
//=====
for(;;)
{
    SW1 = GpioDataRegs.GPBDAT.bit.GPIO47;
    SW2 = GpioDataRegs.GPBDAT.bit.GPIO46;
    SW3 = GpioDataRegs.GPBDAT.bit.GPIO45;
    SW4 = GpioDataRegs.GPBDAT.bit.GPIO44;
    Loop_cnt++;
}
//=====
```

무한루프를 통해 SW1, SW2, SW3, SW4에 각각 GPIO47, 46, 45, 44번 핀의 GPBDAT 값을 대입해주고, Loop_cnt의 값을 증가시킨다. Loop_cnt는 0부터 2^{16} 까지 확인 가능하다.

Table 69. GPIO Port B Data (GPBDAT) Register Field Descriptions

Bit	Field	Value	Description
31-0	GPIO63-GPIO32	0	Each bit corresponds to one GPIO port B pin (GPIO32-GPIO63) as shown in Figure 66 . Reading a 0 indicates that the state of the pin is currently low, irrespective of the mode the pin is configured for. Writing a 0 will force an output of 0 if the pin is configured as a GPIO output in the appropriate GPBMUX1 and GPBDIR registers; otherwise, the value is latched but not used to drive the pin.
		1	Reading a 1 indicates that the state of the pin is currently high irrespective of the mode the pin is configured for. Writing a 1 will force an output of 1 if the pin is configured as a GPIO output in the GPBMUX1 and GPBDIR registers; otherwise, the value is latched but not used to drive the pin.

* **GPBDAT:** Pin이 low일 때 0을 read하고 pin이 high일 때 1을 read한다. 해당 예제에서 GPBDAT 레지스터를 read하여 스위치를 누르지 않을 때는 SW가 0의 값을 갖고, 스위치를 누를 때는 SW의 값이 1로 변하는 것을 확인할 수 있다.

예제 2. GPIO02_Input_Xint

```

#include "DSP28x_Project.h"           // Device Headerfile and Examples Include File
//=====

//=====
// 함수 선언
//=====
interrupt void Xint3_isr(void);
interrupt void Xint4_isr(void);
interrupt void Xint5_isr(void);
interrupt void Xint6_isr(void);
//=====

// ISR 함수 정의
//=====
interrupt void Xint3_isr(void)
{
    SW1_cnt++;
    PieCtrlRegs.PIEACK.all = PIEACK_GROUP12;
}

interrupt void Xint4_isr(void)
{
    SW2_cnt++;
    PieCtrlRegs.PIEACK.all = PIEACK_GROUP12;
}

interrupt void Xint5_isr(void)
{
    SW3_cnt++;
    PieCtrlRegs.PIEACK.all = PIEACK_GROUP12;
}

interrupt void Xint6_isr(void)
{
    SW4_cnt++;
    PieCtrlRegs.PIEACK.all = PIEACK_GROUP12;
}
//=====

```

예제 1과 같이 헤더파일을 추가해준 뒤 네 개의 ISR 함수를 선언해준다.

* **ISR**: Interrupt Service Routine의 약자로, 인터럽트가 발생할 때 수행할 callback 함수이다.

Table 115. PIE Interrupt Acknowledge Register (PIEACK) Field Descriptions

Bits	Field	Value	Description
15-12	Reserved		Reserved
11-0	PIEACK	bit x = 0 ⁽¹⁾ bit x = 1	Each bit in PIEACK refers to a specific PIE group. Bit 0 refers to interrupts in PIE group 1 that are MUXed into INTT up to Bit 11, which refers to PIE group 12 which is MUXed into CPUINT12. If a bit reads as a 0, it indicates that the PIE can send an interrupt from the respective group to the CPU. Writes of 0 are ignored. Reading a 1 indicates if an interrupt from the respective group has been sent to the CPU and all other interrupts from the group are currently blocked. Writing a 1 to the respective interrupt bit clears the bit and enables the PIE block to drive a pulse into the CPU interrupt input if an interrupt is pending for that group.

⁽¹⁾ bit x = PIEACK bit 0 - PIEACK bit 11. Bit 0 refers to CPU INTT up to Bit 11, which refers to CPU INT12

* **PIEACK**: PIE controller를 거친 인터럽트 신호를 한 번에 통제하기 위한 장치이다. PIEIER을 거친 신호를 다음 단계로 전달하기 위해서는 PIEACK에 1값을 write해야 한다. 왜냐하면 동작이 Active Low이기 때문에 1을 write함으로써 해당 영역이 0으로 clear되기 때문이다. 인터럽트 요청 신호가 PIEACK를 통과하고 나면 하드웨어적으로 PIEACK는 1로 set되기 때문에 다음 인터럽트 신호가 통과하지 못하게 된다. 따라서 인터럽트를 사용할 때는 인터럽트 서비스 루틴의 마지막 부분에 PIEACK 레지스터를 Clear 해주어야 한다.

```
// Step 3. 인터럽트 초기화:
//-----
InitPieCtrl();
IER = 0x0000;
IFR = 0x0000;
InitPieVectTable();
```

예제 1과 Step1, 2 과정은 동일하며, Step3은 인터럽트를 초기화 하는 과정이다.

- * **InitPieCtrl**: PIE회로를 초기화하기 위한 함수이다. 먼저 PIE회로를 disable 시키고, PIEIER과 PIEIFR 레지스터를 clear 시켜 놓는다.
- * **IER = 0x0000, IFR = 0x0000**: 인터럽트를 사용하기 위한 준비단계로, 레지스터의 상태를 0으로 설정하여 혹시 모를 시스템 오동작을 방지한다.
- * **InitPieVectTable**: PIE회로를 활성화 시키며, 28x DSP에서 인터럽트를 쓰기 위한 기본 단계이다. 제조사인 TI에서 모든 인터럽트 벡터들을 모두 Mapping 시켜놨기 때문에 이 함수를 사용하게 되면 모든 인터럽트 벡터들이 TI에서 정한 default 인터럽트 서비스 루틴에 연결된다.

```
// Vector Remapping
EALLOW;
PieVectTable.XINT3 = &Xint3_isr;
PieVectTable.XINT4 = &Xint4_isr;
PieVectTable.XINT5 = &Xint5_isr;
PieVectTable.XINT6 = &Xint6_isr;
EDIS;
//=====
```

Reserved	Reserved	Reserved	Reserved
XINT6	XINT5	XINT4	XINT3
Ext. Int. 6	Ext. Int. 5	Ext. Int. 4	Ext. Int. 3
0xDF6	0xDF4	0xDF2	0xDF0

Default로 설정된 인터럽트 벡터를 Remapping 해준다. 외부 인터럽트 3이 걸렸을 때 Xint3_isr이라는 ISR 함수를 실행하도록 설정한 코드이다. PIE Vector Table을 살펴보면 우측 하단 낮은 우선순위에 위치한 것을 알 수 있다.

```
// Step 4. GPIO 초기화
//-----
EALLOW;
GpioCtrlRegs.GPMUX1.bit.GP1044 = 0; // 핀 기능선택: GP1044
GpioCtrlRegs.GPMUX1.bit.GP1045 = 0; // 핀 기능선택: GP1045
GpioCtrlRegs.GPMUX1.bit.GP1046 = 0; // 핀 기능선택: GP1046
GpioCtrlRegs.GPMUX1.bit.GP1047 = 0; // 핀 기능선택: GP1047
GpioCtrlRegs.GPDIR.bit.GP1044 = 0; // GP1044 입출력 선택: Input
GpioCtrlRegs.GPDIR.bit.GP1045 = 0; // GP1045 입출력 선택: Input
GpioCtrlRegs.GPDIR.bit.GP1046 = 0; // GP1046 입출력 선택: Input
GpioCtrlRegs.GPDIR.bit.GP1047 = 0; // GP1047 입출력 선택: Input
EDIS;
//=====
```

예제 1에서 했던 방식과 동일하게 핀 기능은 GPIO로, 입출력 선택은 입력으로 선택하였다.

```
// Step 5. XINT 초기화
```

```
//
EALLOW;
GpioIntRegs.GPIOXINT3SEL.bit.GPIOSEL = 47; // 외부 인터럽트 XINT3로 사용할 핀 선택: GPIO47
GpioIntRegs.GPIOXINT4SEL.bit.GPIOSEL = 46; // 외부 인터럽트 XINT4로 사용할 핀 선택: GPIO46
GpioIntRegs.GPIOXINT5SEL.bit.GPIOSEL = 45; // 외부 인터럽트 XINT5로 사용할 핀 선택: GPIO45
GpioIntRegs.GPIOXINT6SEL.bit.GPIOSEL = 44; // 외부 인터럽트 XINT6로 사용할 핀 선택: GPIO44
EDIS;
```

Table 82. GPIO XINT3 - XINT7 Interrupt Select (GPIOXINTnSEL) Register Field Descriptions⁽¹⁾

Bits	Field	Value	Description ⁽²⁾
15-5	Reserved		Reserved
4-0	GPIOXINTnSEL		Select the port B GPIO signal (GPIO32 - GPIO63) that will be used as the XINTn interrupt source. In addition, you can configure the interrupt in the XINTnCR register described in Section 8.6.
		00000	Select the GPIO32 pin as the XINTn interrupt source (default)
		00001	Select the GPIO33 pin as the XINTn interrupt source
	
		11110	Select the GPIO62 pin as the XINTn interrupt source
		11111	Select the GPIO63 pin as the XINTn interrupt source

⁽¹⁾ n = 3, 4, 5, 6, or 7

⁽²⁾ This register is EALLOW protected. See Section 7.2 for more information.

Table 83. XINT3 - XINT7 Interrupt Select and Configuration Registers

n	Interrupt	Interrupt Select Register	Configuration Register
3	XINT3	GPIOXINT3SEL	XINT3CR
4	XINT4	GPIOXINT4SEL	XINT4CR
5	XINT5	GPIOXINT5SEL	XINT5CR
6	XINT6	GPIOXINT6SEL	XINT6CR
7	XINT7	GPIOXINT7SEL	XINT7CR

* **GPIOXINnSEL**: 외부 인터럽트로 사용할 핀을 선택한다. XINT3부터 XINT6까지 외부 인터럽트로 사용하며 그에 해당하는 Interrupt Select Register와 Configuration Register는 위 표와 같다.

```
XIntruptRegs.XINT3CR.bit.POLARITY = 0; // XINT3 인터럽트 발생 조건 설정: 입력 신호의 하강 엣지
XIntruptRegs.XINT4CR.bit.POLARITY = 1; // XINT4 인터럽트 발생 조건 설정: 입력 신호의 상승 엣지
XIntruptRegs.XINT5CR.bit.POLARITY = 2; // XINT5 인터럽트 발생 조건 설정: 입력 신호의 하강 엣지
XIntruptRegs.XINT6CR.bit.POLARITY = 3; // XINT6 인터럽트 발생 조건 설정: 입력 신호의 하강 & 상승 엣지

XIntruptRegs.XINT3CR.bit.ENABLE = 1; // XINT3 인터럽트 : Enable
XIntruptRegs.XINT4CR.bit.ENABLE = 1; // XINT4 인터럽트 : Enable
XIntruptRegs.XINT5CR.bit.ENABLE = 1; // XINT5 인터럽트 : Enable
XIntruptRegs.XINT6CR.bit.ENABLE = 1; // XINT6 인터럽트 : Enable
```

Table 121. External Interrupt n Control Register (XINTnCR) Field Descriptions

Bits	Field	Value	Description
15-4	Reserved		Reads return zero; writes have no effect.
3-2	Polarity		This read/write bit determines whether interrupts are generated on the rising edge or the falling edge of a signal on the pin.
		00	Interrupt generated on a falling edge (high-to-low transition)
		01	Interrupt generated on a rising edge (low-to-high transition)
		10	Interrupt is generated on a falling edge (high-to-low transition)
		11	Interrupt generated on both a falling edge and a rising edge (high-to-low and low-to-high transition)
1	Reserved		Reads return zero; writes have no effect
0	Enable		This read/write bit enables or disables external interrupt XINTn.
		0	Disable interrupt
		1	Enable interrupt

* **Polarity**: 입력값에 따라 인터럽트가 언제 발생할 것인지를 설정하게 된다. 0일 경우 하강edge, 1일 경우 상승 edge, 2일 경우 하강 edge, 3일 경우 하강&상승 edge에서 인터럽트가 발생하게 된다.

* **Enable**: 1값을 입력하게 되면 인터럽트를 enable한다.


```

// 외부 인터럽트 포함된 벡터 활성화
PieCtrlRegs.PIEIER12.bit.INTx1 = 1;      // PIE 인터럽트(XINT3) : Enable
PieCtrlRegs.PIEIER12.bit.INTx2 = 1;      // PIE 인터럽트(XINT4) : Enable
PieCtrlRegs.PIEIER12.bit.INTx3 = 1;      // PIE 인터럽트(XINT5) : Enable
PieCtrlRegs.PIEIER12.bit.INTx4 = 1;      // PIE 인터럽트(XINT6) : Enable
IER |= M_INT12;                          // CPU 인터럽트(INT12) : Enable
//=====

```

- * **PIEIER12**: Group Enable Register이며, INTx1, 2, 3, 4를 사용하여 외부 인터럽트 XINT3, 4, 5, 6에 해당하는 각각의 PIE 인터럽트를 Enable 상태로 설정해준다.
- * **IER |= M_INT12**: CPU 인터럽트를 Enable 상태로 설정해준다.

이후 Switch와 Loop_cnt 변수를 초기화 해준 뒤 무한루프를 실행시킨다.

예제 3. GPIO02_Input_Qual

```

// Step 5: Qualification 초기화
//=====
EALLOW;
GpioCtrlRegs.GPBCTRL.bit.QUALPRD1 = 0xFF; // (GPIO40~GPIO47) Qual period 설정
GpioCtrlRegs.GPBQSEL1.bit.GPIO44 = 2;    // Qualification using 6 samples
GpioCtrlRegs.GPBQSEL1.bit.GPIO45 = 2;    // Qualification using 6 samples
GpioCtrlRegs.GPBQSEL1.bit.GPIO46 = 2;    // Qualification using 6 samples
GpioCtrlRegs.GPBQSEL1.bit.GPIO47 = 2;    // Qualification using 6 samples
EDIS;
//=====

```

예제 2의 'GPIO 초기화'와 'XINT 초기화' 사이에 'Qualification 초기화'가 추가된 예제이다.

Table 57. GPIO Port B Qualification Control (GPBCTRL) Register Field Descriptions

Bits	Field	Value	Description ⁽¹⁾
15-8	QUALPRD1	0x00 0x01 0x02 ... 0xFF	Specifies the sampling period for pins GPIO40 to GPIO47 Sampling Period = $T_{SYSCLKOUT}$ ⁽²⁾ Sampling Period = $2 \times T_{SYSCLKOUT}$ Sampling Period = $4 \times T_{SYSCLKOUT}$... Sampling Period = $510 \times T_{SYSCLKOUT}$

- * **GPBCTRL**: 샘플링 주기를 설정 가능하다. 0xFF값을 갖는 경우 샘플링 주기는 $510 \times T_{SYSCLKOUT}$ 값을 갖는다.

Table 60. GPIO Port B Qualification Select 1 (GPBQSEL1) Register Field Descriptions

Bits	Field	Value	Description ⁽¹⁾
31-0	GPIO47-GPIO32	00 01 10 11	Select input qualification type for GPIO32 to GPIO47. The input qualification of each GPIO input is controlled by two bits as shown in Figure 55. 00 Synchronize to SYSCLKOUT only. Valid for both peripheral and GPIO pins. 01 Qualification using 3 samples. Valid for pins configured as GPIO or a peripheral function. The time between samples is specified in the GPACTRL register. 10 Qualification using 6 samples. Valid for pins configured as GPIO or a peripheral function. The time between samples is specified in the GPACTRL register. 11 Asynchronous. (no synchronization or qualification). This option applies to pins configured as peripherals only. If the pin is configured as a GPIO input, then this option is the same as 0,0 or synchronize to SYSCLKOUT.

⁽¹⁾ This register is EALLOW protected. See Section 7.2 for more information.

- * **GPBQSEL**: Input qualification type을 선택한다. 샘플의 사용 횟수를 정하는데, 1값을 입력할 경우 3개의 샘플을 이용하고 2값을 입력할 경우 6개의 샘플을 이용한다. 해당 예제에서는 2값을 입력했기 때문에 6개의 샘플을 qualification에 이용한다.

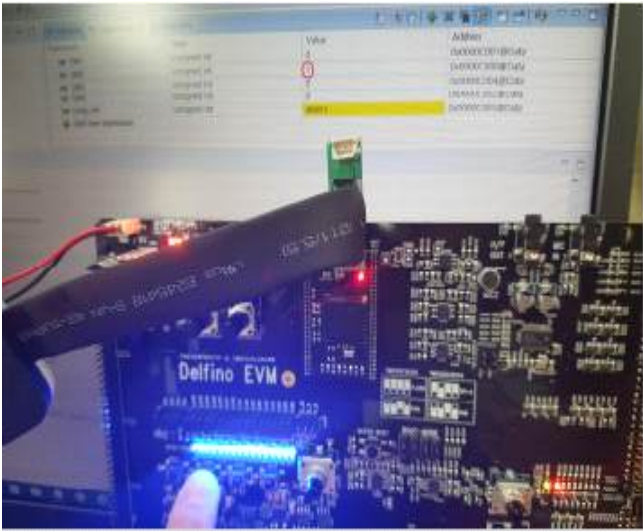
2. 실험 결과

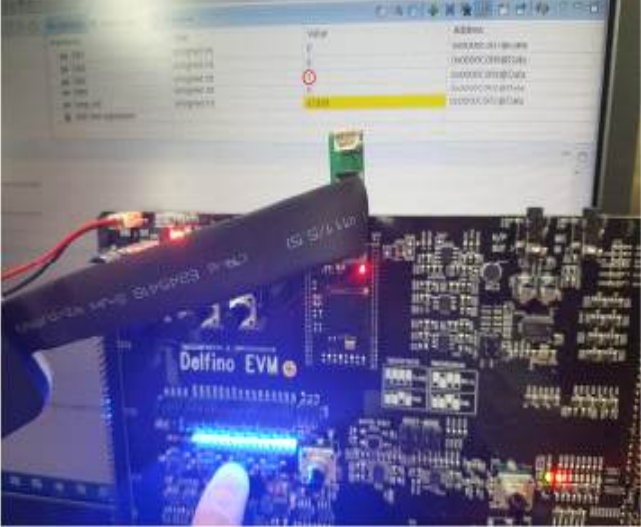
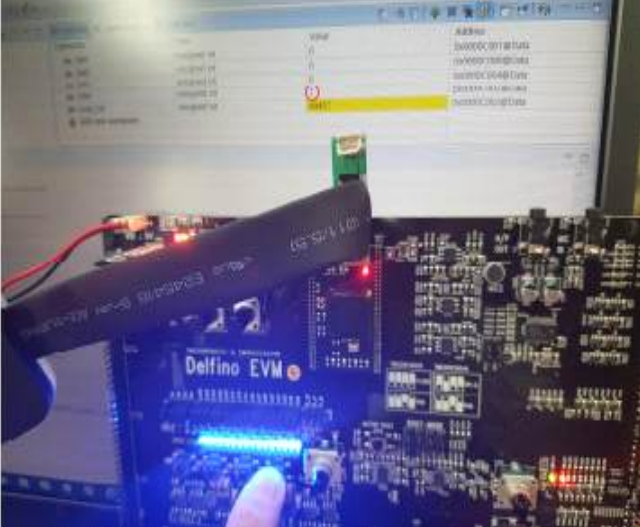
예제 1. GPIO01_Input

- 실험 목표:

GPIO 핀을 입력으로 사용하여 Delfino EVM의 스위치 4개에서 들어오는 신호를 확인한다. CCS의 Watch Window를 통해서 스위치 상태 변화에 따른 SW 변수 값을 확인한다.

- 실험 분석:

Input All	Input SW1
	
SW1:1, SW2:1, SW3:1, SW4:1	SW1:1, SW2:0, SW3:0, SW4:0
Input SW1, SW2	Input SW2
	
SW1:1, SW2:1, SW3:0, SW4:0	SW1:0, SW2:1, SW3:0, SW4:0

Input SW3	Input SW4
	
SW1:0, SW2:0, SW3:1, SW4:0	SW1:0, SW2:0, SW3:0, SW4:1

Delfino EVM 보드의 스위치를 왼쪽부터 SW1, SW2, SW3, SW4, 값으로 설정해주었다. 설정해준 변수에는 GPIO47, 46, 45, 44의 GPBDAT 값이 입력되게 된다.

GPBDAT은 Pin이 low일 때 0을 read하고 pin이 high일 때 1을 read한다. 따라서 위 실험 결과와 같이 GPBDAT 레지스터를 read하여 스위치를 누르지 않을 때는 SW가 0의 값을 갖고, 스위치를 누를 때는 SW의 값이 1로 변하는 것을 확인할 수 있다. 또한 루프가 계속 돌아가고 있음을 Loop_cnt 값이 바뀌는 것을 통해 확인할 수 있다.

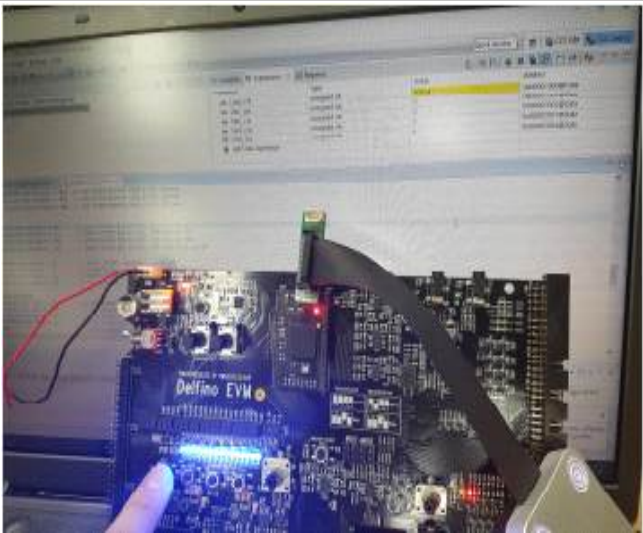
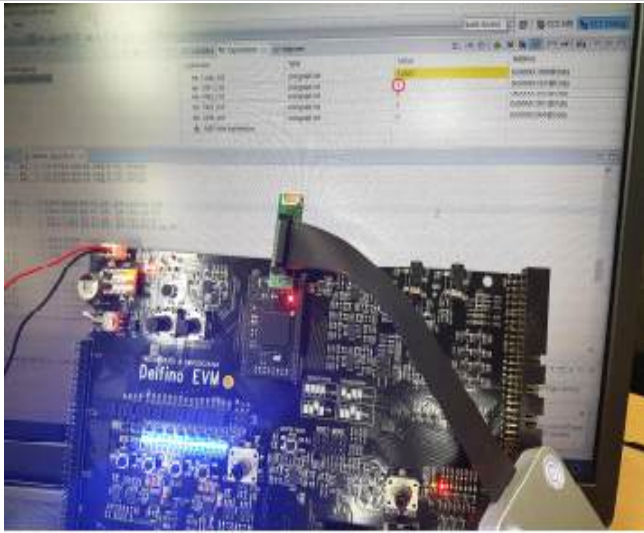
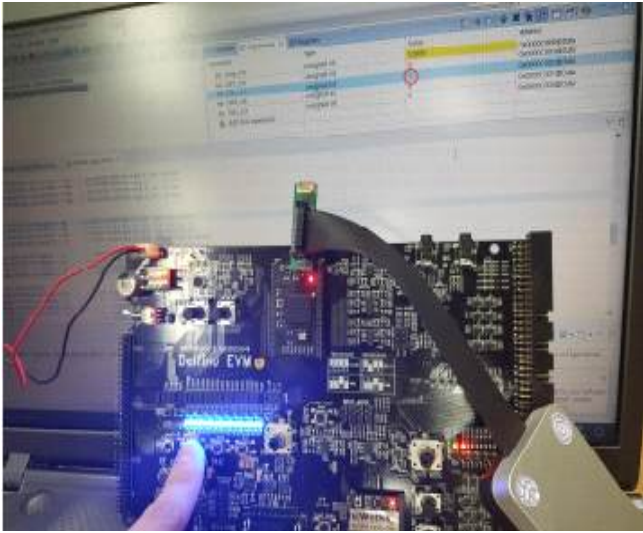
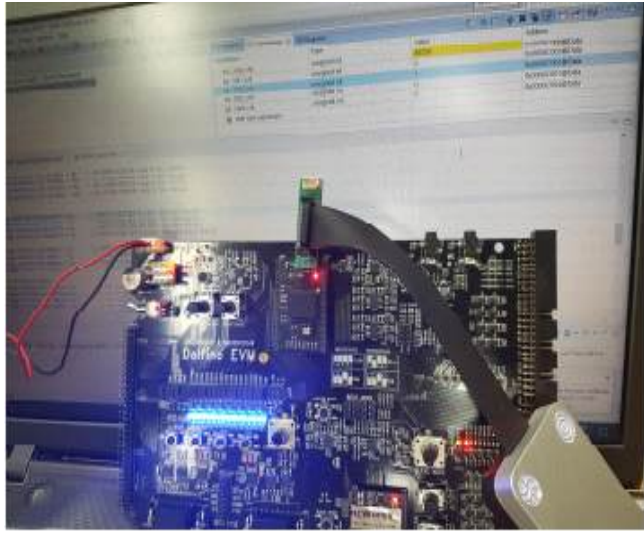
예제 2. GPIO02_Input_Xint

• 실험 목표:

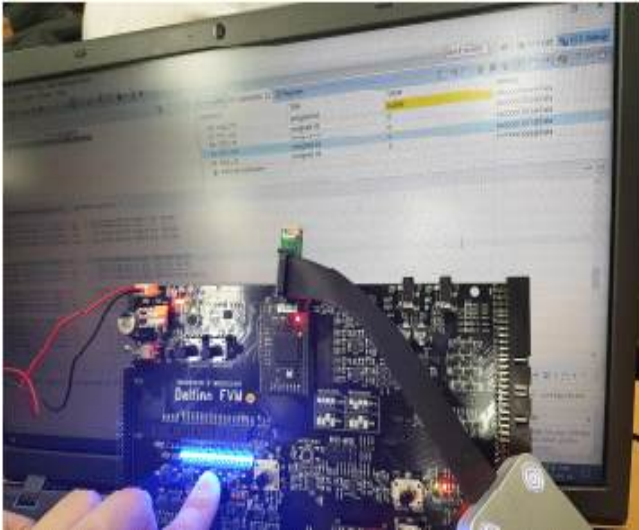
GPIO 핀을 입력으로 사용하고, 네 개의 스위치에서 들어오는 신호를 외부 인터럽트로 사용한다. 외부 인터럽트의 입력에 따른 SW 변수 값을 확인하고 Polarity, Enable의 값 변화에 따른 입력 특성을 알아본다.

• 실험 분석:

[실험 2-1] Polarity: SW1부터 SW4까지 순서대로 0(하강), 1(상승), 2(하강), 3(하강, 상승), Enable: 모두 1

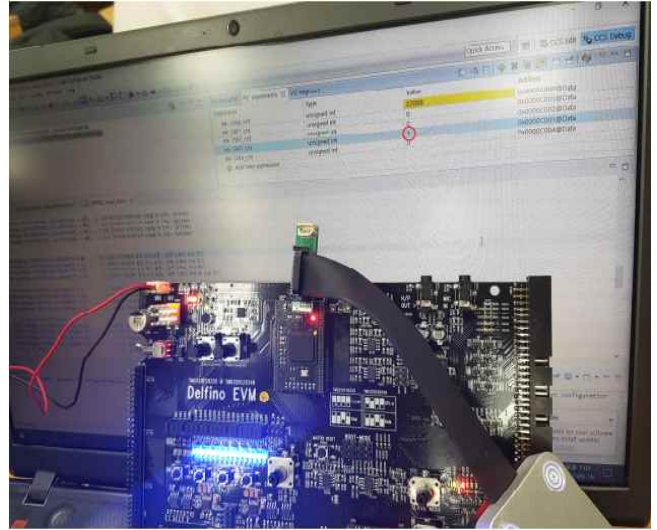
SW1 상승 edge (Polarity: 0, Enable: 1)	SW1 하강 edge (Polarity: 0, Enable: 1)
	
SW1_cnt:0, SW2_cnt:0, SW3_cnt:0, SW4_cnt:0	SW1_cnt:1, SW2_cnt:0, SW3_cnt:0, SW4_cnt:0
SW2 상승 edge (Polarity: 1, Enable: 1)	SW2 하강 edge (Polarity: 1, Enable: 1)
	
SW1_cnt:0, SW2_cnt:1, SW3_cnt:0, SW4_cnt:0	SW1_cnt:0, SW2_cnt:1, SW3_cnt:0, SW4_cnt:0

SW3 상승 edge (Polarity: 2, Enable: 1)



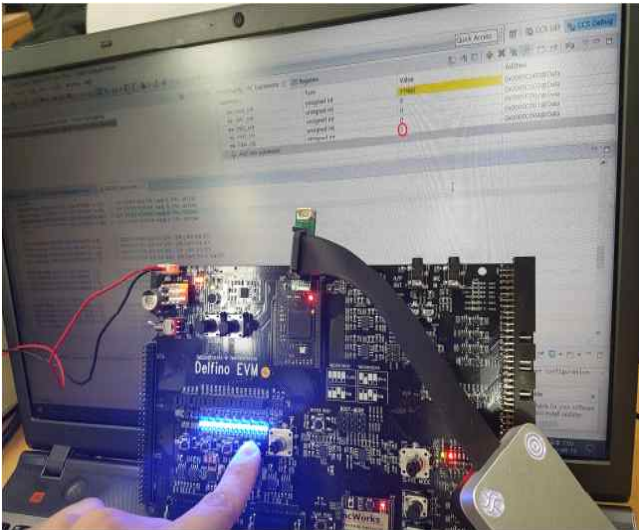
SW1_cnt:0, SW2_cnt:0, SW3_cnt:0, SW4_cnt:0

SW3 하강 edge (Polarity: 2, Enable: 1)



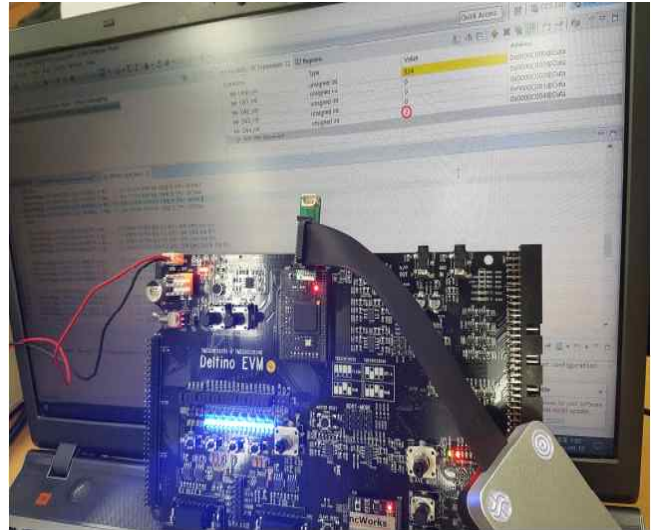
SW1_cnt:0, SW2_cnt:0, SW3_cnt:1, SW4_cnt:0

SW4 상승 edge (Polarity: 3, Enable: 1)



SW1_cnt:0, SW2_cnt:0, SW3_cnt:0, SW4_cnt:1

SW4 하강 edge (Polarity: 3, Enable: 1)



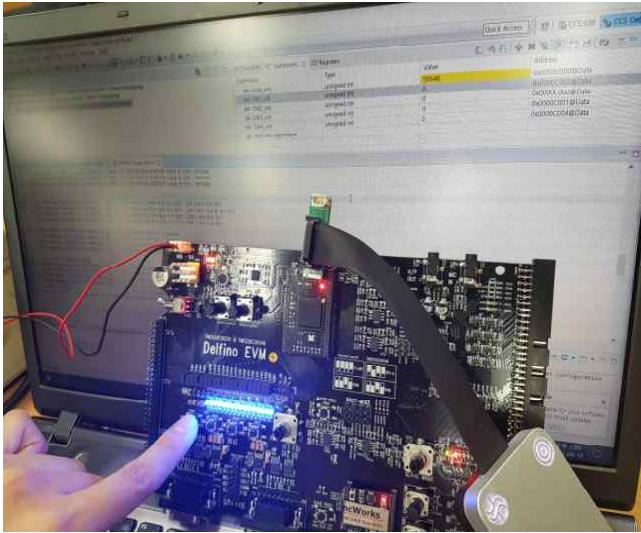
SW1_cnt:0, SW2_cnt:0, SW3_cnt:0, SW4_cnt:2

SW1부터 SW4에 외부 인터럽트로 사용할 GPIO pin 47~44을 배정한 뒤 ISR을 Remapping했다. 이후 SW1부터 SW4까지 순서대로 0(하강), 1(상승), 2(하강), 3(하강, 상승)의 Polarity 값을 입력하고 모든 Enable은 1값을 주어 인터럽트를 enable했다.

- SW1에서는 Polarity 값이 0이기 때문에 falling edge에서만 count값이 1 증가하는 것을 확인할 수 있다.
- SW2에서는 Polarity 값이 1이기 때문에 rising edge에서만 count값이 1 증가하는 것을 확인할 수 있다.
- SW3에서는 Polarity 값이 2이기 때문에 falling edge에서만 count값이 1 증가하는 것을 확인할 수 있다.
- SW4에서는 Polarity 값이 3이기 때문에 rising edge와 falling edge에서 각각 count값이 1씩 증가하는 것을 확인할 수 있다.

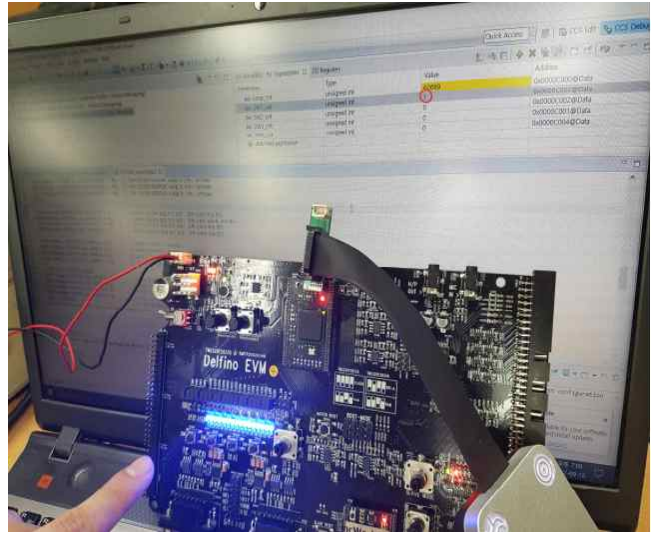
[실험 2-2] Polarity: SW1부터 SW4까지 순서대로 2(하강), 3(하강, 상승), 0(하강), 1(상승), Enable: 모두 1

SW1 상승 edge (Polarity: 2, Enable: 1)



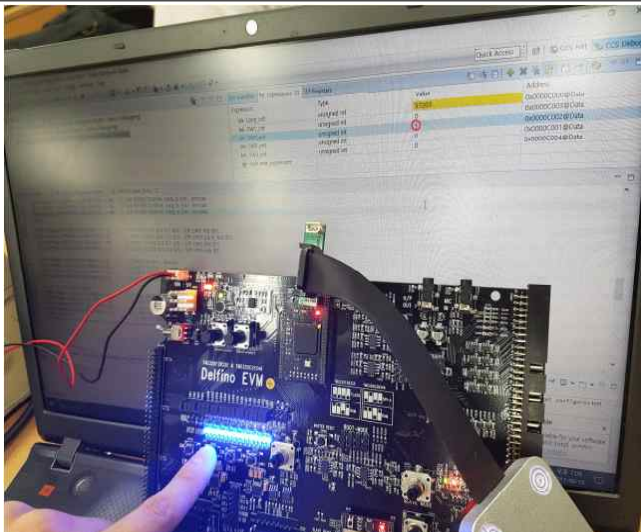
SW1_cnt:0, SW2_cnt:0, SW3_cnt:0, SW4_cnt:0

SW1 하강 edge (Polarity: 2, Enable: 1)



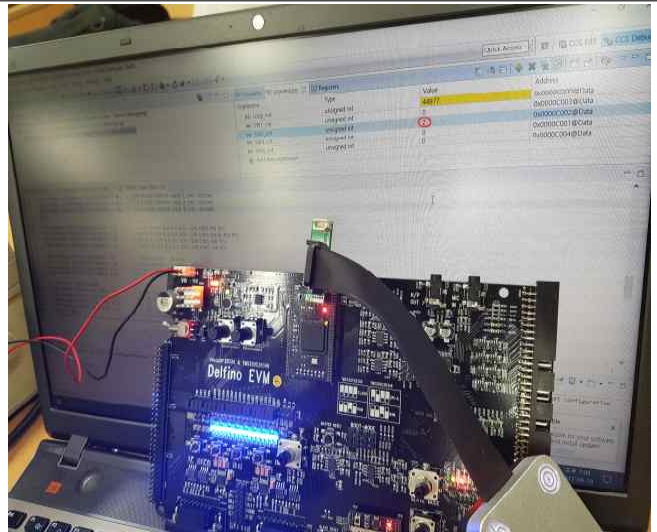
SW1_cnt:1, SW2_cnt:0, SW3_cnt:0, SW4_cnt:0

SW2 상승 edge (Polarity: 3, Enable: 1)



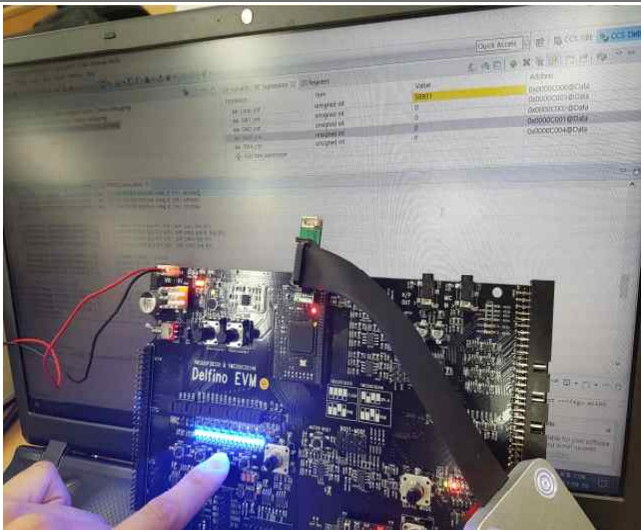
SW1_cnt:0, SW2_cnt:1, SW3_cnt:0, SW4_cnt:0

SW2 하강 edge (Polarity: 3, Enable: 1)



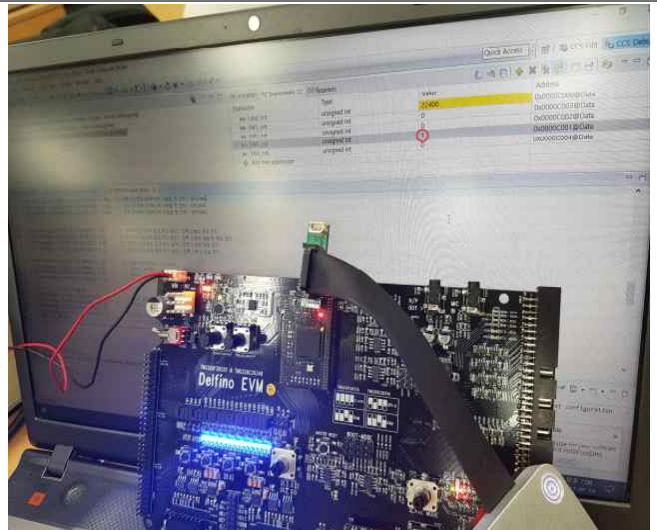
SW1_cnt:0, SW2_cnt:2, SW3_cnt:0, SW4_cnt:0

SW3 상승 edge (Polarity: 0, Enable: 1)

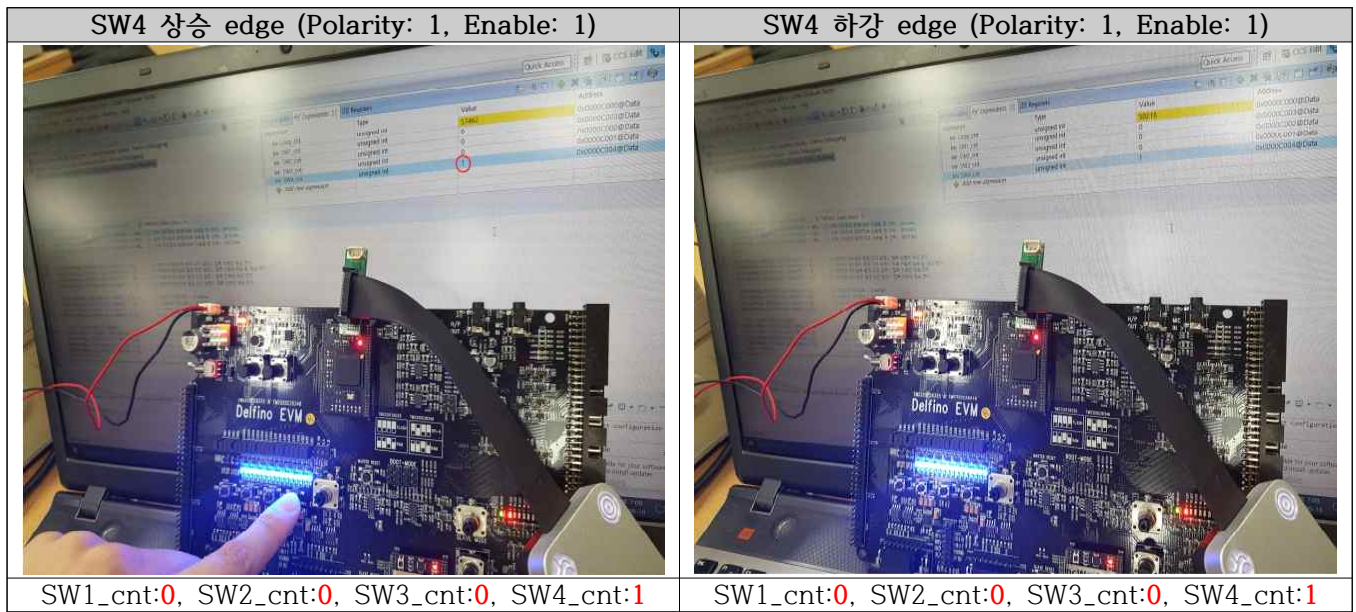


SW1_cnt:0, SW2_cnt:0, SW3_cnt:0, SW4_cnt:0

SW3 하강 edge (Polarity: 0, Enable: 1)



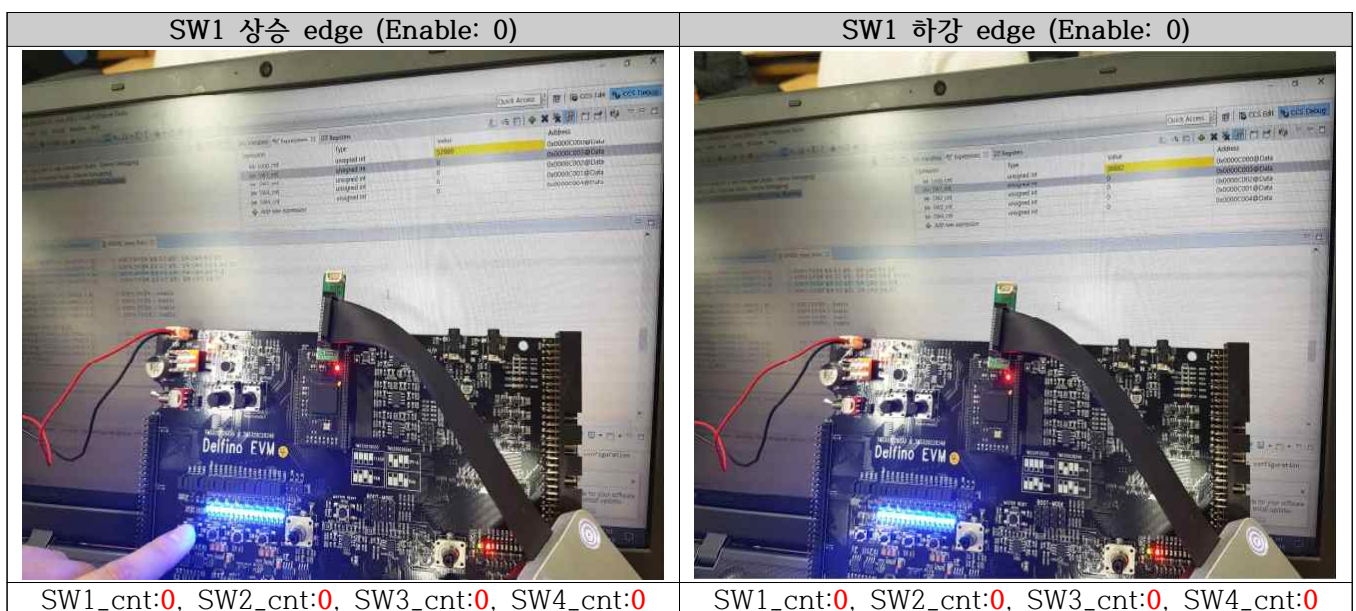
SW1_cnt:0, SW2_cnt:0, SW3_cnt:1, SW4_cnt:0



실험 2-1 과 같은 코드에 Polarity 값만 변경한 뒤 입력 특성을 확인해보았다. SW1부터 SW4까지 순서대로 2 (하강), 3(하강, 상승), 0(하강), 1(상승)의 Polarity 값을 입력하고 모든 Enable은 1값을 주어 인터럽트를 enable 했다.

- SW1에서는 Polarity 값이 2이기 때문에 falling edge에서만 count값이 1 증가하는 것을 확인할 수 있다.
- SW2에서는 Polarity 값이 3이기 때문에 rising edge와 falling edge에서 각각 count값이 1씩 증가하는 것을 확인할 수 있다.
- SW3에서는 Polarity 값이 0이기 때문에 falling edge에서만 count값이 1 증가하는 것을 확인할 수 있다.
- SW4에서는 Polarity 값이 1이기 때문에 rising edge에서만 count값이 1 증가하는 것을 확인할 수 있다.

[실험 2-3] XINT3의 Enable값이 0 일 경우

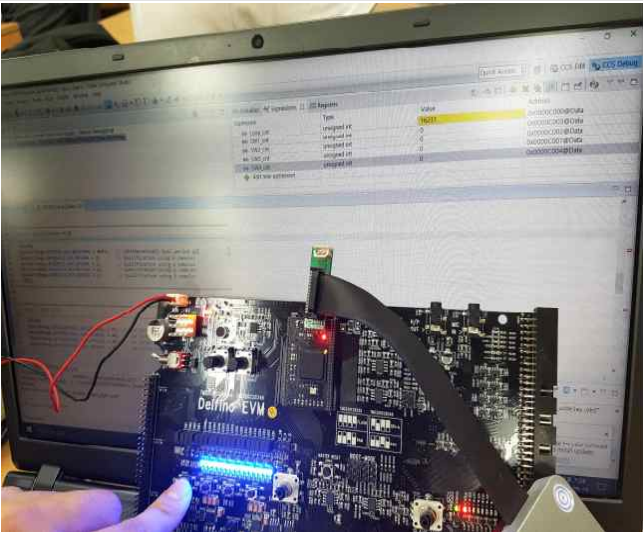
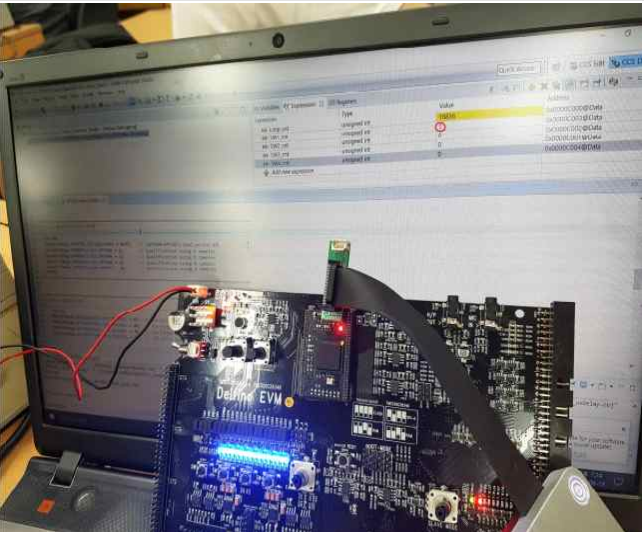
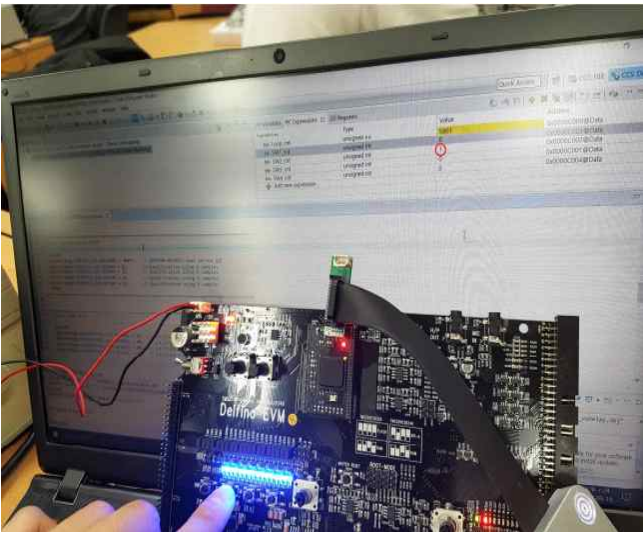
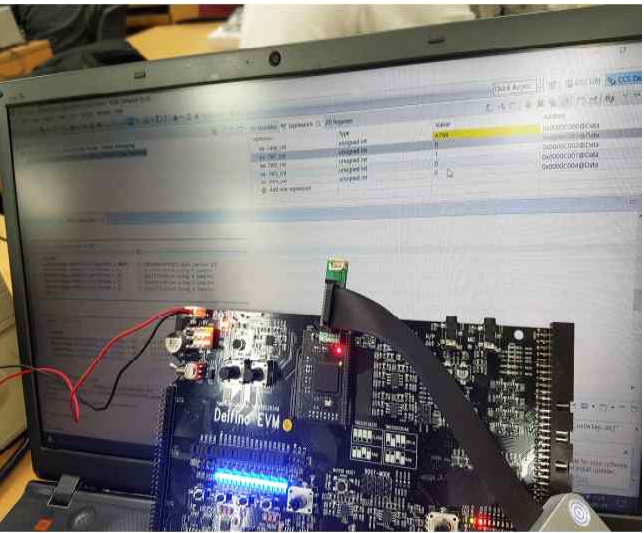


Enable 값에 0을 대입하면 'Disable Interrupt'를 시행하므로 스위치를 눌러도 아무 변화가 없는 것을 확인할 수 있다.

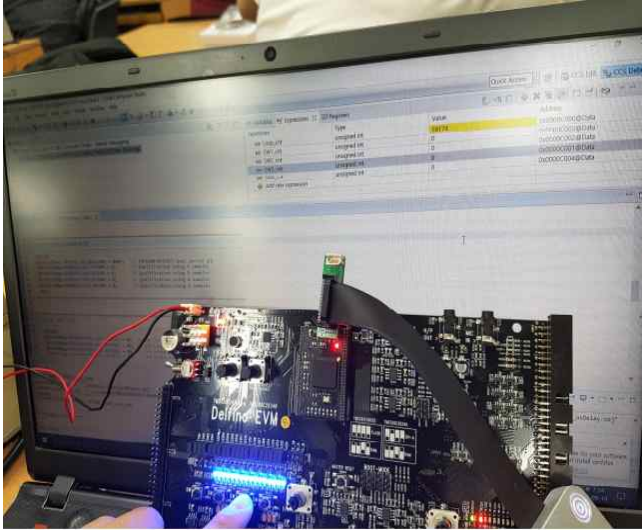
예제 3. GPIO03_Input_Qual

- 실험 목표:
GPIO 핀을 입력으로 사용하고, 네 개의 스위치에서 들어오는 신호를 외부 인터럽트로 사용한다. Qualification 기능을 사용했을 때 스위치 입력에서 발생하는 노이즈가 감소되는 것을 확인한다.
- 실험 분석:

[실험 3-1]
GPBQSEL1에 2의 값을 입력, Polarity: SW1부터 SW4까지 순서대로 0(하강), 1(상승), 2(하강), 3(하강, 상승), Enable: 모두 1

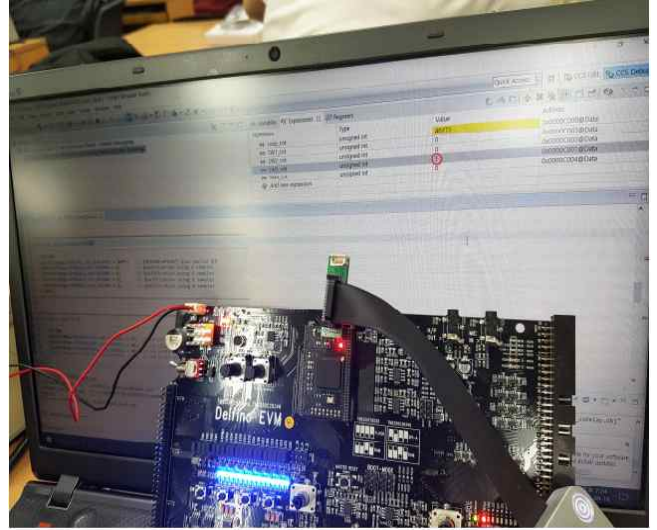
<div>SW1 상승 edge (GPBQSEL1 = 2, Polarity: 0)</div> 	<div>SW1 하강 edge (GPBQSEL1 = 2, Polarity: 0)</div> 
<div>SW2 상승 edge (GPBQSEL1 = 2, Polarity: 1)</div> 	<div>SW2 하강 edge (GPBQSEL1 = 2, Polarity: 1)</div> 

SW3 상승 edge (GPBQSEL1 = 2, Polarity: 2)



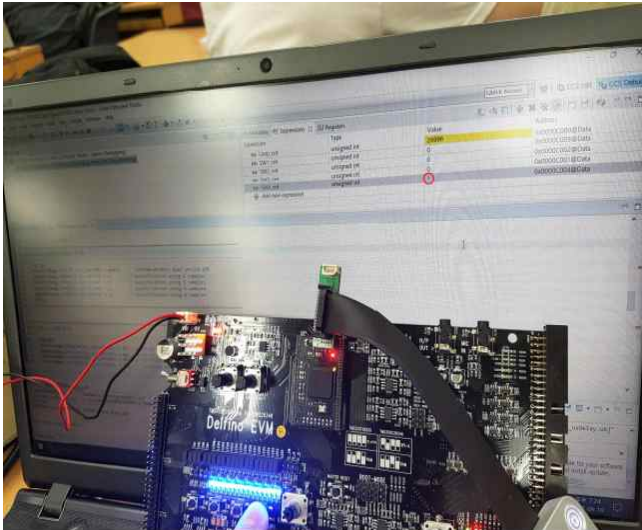
SW1_cnt:0, SW2_cnt:0, SW3_cnt:0, SW4_cnt:0

SW3 하강 edge (GPBQSEL1 = 2, Polarity: 2)



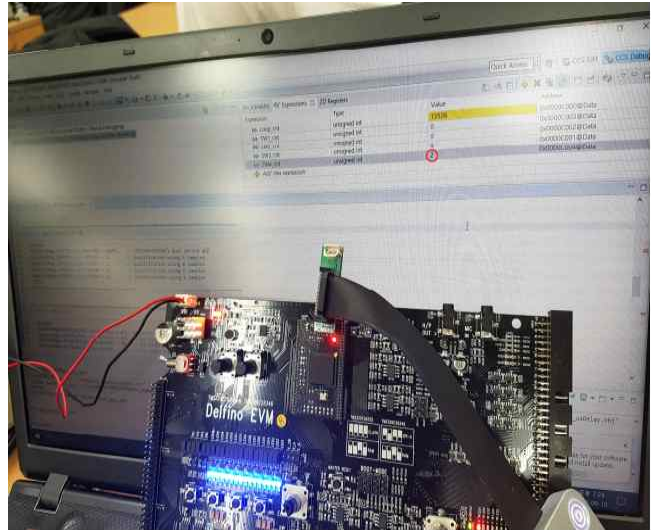
SW1_cnt:0, SW2_cnt:0, SW3_cnt:1, SW4_cnt:0

SW4 상승 edge (GPBQSEL1 = 2, Polarity: 3)



SW1_cnt:0, SW2_cnt:0, SW3_cnt:0, SW4_cnt:1

SW4 하강 edge (GPBQSEL1 = 2, Polarity: 3)



SW1_cnt:0, SW2_cnt:0, SW3_cnt:0, SW4_cnt:2

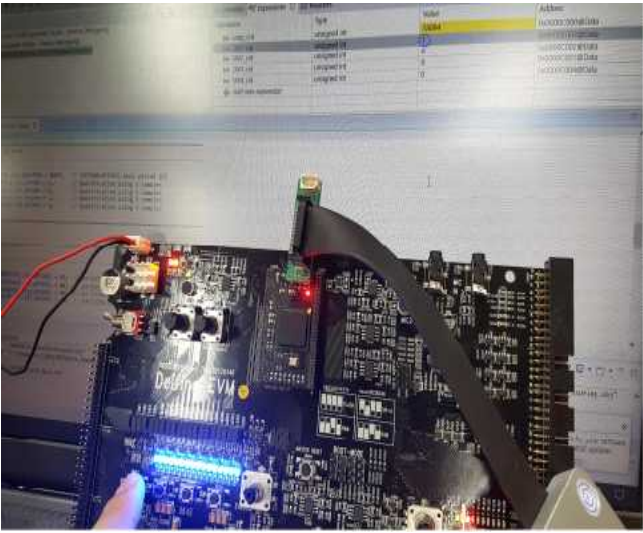
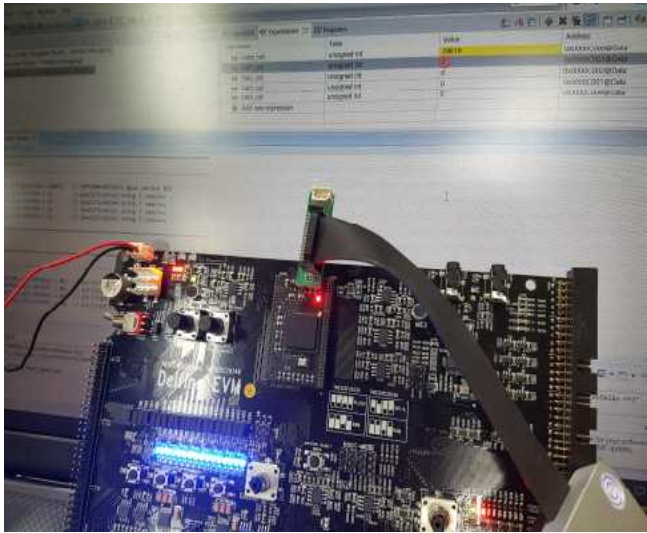
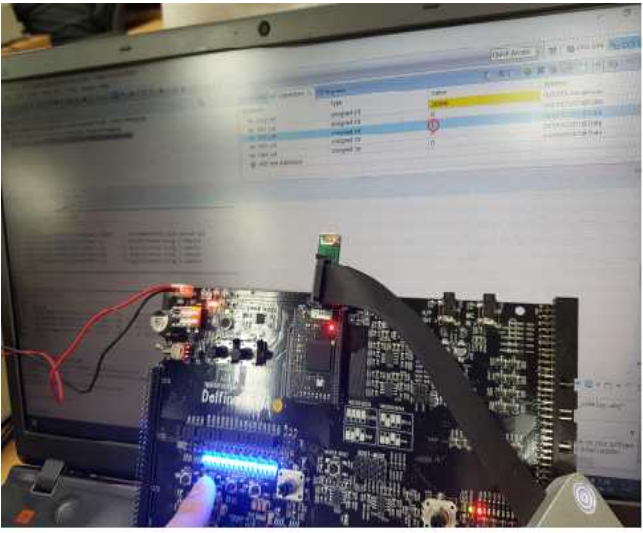
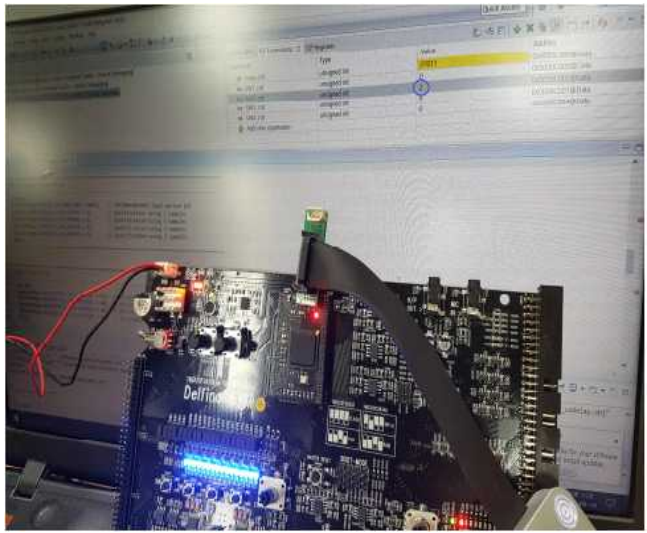
SW1부터 SW4에 외부 인터럽트로 사용할 GPIO pin 47~44을 배정한 뒤 ISR을 Remapping했다. 이후 SW1부터 SW4까지 순서대로 0(하강), 1(상승), 2(하강), 3(하강, 상승)의 Polarity 값을 입력하고 모든 Enable은 1값을 주어 인터럽트를 enable했다. 여기에 샘플링 사용 횟수를 정하는 레지스터인 GPBQSEL1을 추가했다.

GPBQSEL1 레지스터에 1값을 입력할 경우 3개의 샘플을 이용하고 2값을 입력할 경우 6개의 샘플을 이용한다. 해당 실험에서는 2값을 입력했기 때문에 6개의 샘플을 qualification에 이용한다. 그 결과 각 polarity 특성에 맞게 count값의 변화가 rising 혹은 falling edge에서 1씩 있는 것을 확인할 수 있었다.

- SW1에서는 Polarity 값이 0이기 때문에 falling edge에서만 count값이 1 증가하는 것을 확인할 수 있다.
- SW2에서는 Polarity 값이 1이기 때문에 rising edge에서만 count값이 1 증가하는 것을 확인할 수 있다.
- SW3에서는 Polarity 값이 2이기 때문에 falling edge에서만 count값이 1 증가하는 것을 확인할 수 있다.
- SW4에서는 Polarity 값이 3이기 때문에 rising edge와 falling edge에서 각각 count값이 1씩 증가하는 것을 확인할 수 있다.

[실험 3-2]

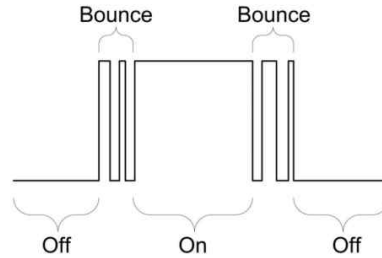
GPBQSEL1에 1의 값을 입력, Polarity: SW1부터 SW4까지 순서대로 0(하강), 1(상승), 2(하강), 3(하강, 상승), Enable: 모두 1

SW1 상승 edge (GPBQSEL1 = 1, Polarity: 0)	SW1 하강 edge (GPBQSEL1 = 1, Polarity: 0)
	
SW1_cnt:1, SW2_cnt:0, SW3_cnt:0, SW4_cnt:0	SW1_cnt:2, SW2_cnt:0, SW3_cnt:0, SW4_cnt:0
SW2 상승 edge (GPBQSEL1 = 1, Polarity: 1)	SW2 하강 edge (GPBQSEL1 = 1, Polarity: 1)
	
SW1_cnt:0, SW2_cnt:1, SW3_cnt:0, SW4_cnt:0	SW1_cnt:0, SW2_cnt:2, SW3_cnt:0, SW4_cnt:0

[실험 3-1]의 조건에서 GPBBQSEL1의 값을 1로 바꿔 3개의 샘플을 이용했다. 그 결과 6개의 샘플을 이용할 때보다 노이즈가 섞은 것을 확인할 수 있었다. SW1의 경우 Polarity가 0이기 때문에 rising edge에서는 값의 변화가 일어나지 않아야 하는데 0에서 1로 증가한 것을 확인할 수 있다. 마찬가지로 SW2의 경우 Polarity가 1이기 때문에 falling edge에서는 값의 변화가 일어나지 않아야 하는데 1에서 2로 증가한 것을 확인할 수 있다.

이러한 오류는 채터링 현상에 의한 것이다. 채터링 현상이란 아래 <그림 1>과 같이 스위치가 열려있는 상태에서 닫거나 닫혀있는 상태에서 열 때, 스위치의 상태가 변하는 짧은 순간에 열림과 닫힘이 수회 반복되는 현상을 의미한다.

채터링 현상을 방지해주기 위한 용도로 Qualification을 사용하며, [실험 3-1]과 [실험 3-2]를 통해 6개의 샘플을 사용하는 것이 3개의 샘플을 사용하는 것보다 노이즈 감소 효과가 좋다는 것을 확인할 수 있었다.



<그림 1> 채터링 현상

3. 실험 고찰

이번 실험에서는 CCS를 이용한 GPIO 입력 특성을 알아보았다. Power Supply를 통해 9V, 1A의 전압과 전류를 Delfino DSP 보드에 인가한 뒤 CCS의 코딩을 통해 빌드된 코드를 디버깅하여 타겟 보드에 입력을 시켜주었다. 이후 스위치의 On, Off에 따른 변수 값 SW의 변화를 살펴보았다.

실험 1에서는 GPBDIR 레지스터를 통해 GPIO핀을 스위치를 누르는 입력 역할로 설정했다. Delfino EVM 보드의 스위치를 왼쪽부터 SW1, SW2, SW3, SW4, 값으로 설정해주고, 설정해준 변수에는 GPIO47, 46, 45, 44의 GPBDAT 값이 입력되게 하였다. 이 과정을 통해 스위치를 누르지 않을 때는 SW가 0의 값을 갖고, 스위치를 누를 때는 SW의 값이 1로 변하게 된다. 실제 실험 결과를 보면 스위치를 누르면 해당 스위치의 변수 값인 SW가 1의 값을 갖고, 누르지 않으면 0의 값을 갖는 것을 확인 할 수 있다. 개별적인 스위치의 변화 뿐만 아니라 동시에 스위치 네 개를 모두 눌러도 동작하는 모습을 확인하였다. 루프가 계속 돌아가고 있다는 것은 Loop_cnt 값의 변화를 통해 알 수 있었다.

실험 2은 GPIO 핀을 입력으로 사용하고, 네 개의 스위치에서 들어오는 신호를 외부 인터럽트로 사용했다. 외부 인터럽트의 입력에 따른 스위치 변수 값을 확인하고 Polarity, Enable의 값 변화에 따른 입력 특성을 알아보았다. 먼저 SW1부터 SW4에 외부 인터럽트로 사용할 GPIO pin 47~44을 배정한 뒤 ISR을 Remapping했다. 이후 SW1부터 SW4까지 순서대로 0(하강), 1(상승), 2(하강), 3(하강, 상승)의 Polarity 값을 입력하고 모든 Enable은 1값을 주어 인터럽트를 enable했다. 그 결과 SW1_cnt는 falling edge에서만, SW2_cnt는 rising edge에서만, SW3_cnt는 falling edge에서만, SW4_cnt는 rising과 falling edge 각각에서 count값이 1씩 증가하는 것을 확인할 수 있었다. 각 스위치의 Polarity 값의 변화를 통해 인터럽트 발생 조건을 설정 할 수 있었다.

추가로 XINT3의 Enable값이 0 일 경우에 대해서도 실험을 진행했는데, Enable 값에 0을 대입하면 'Disable Interrupt'를 시행하므로 스위치를 눌러도 아무 변화가 없었다.

실험 3은 실험 2의 조건에서 Qualification 기능을 추가했을 때 스위치 입력에서 발생하는 노이즈가 감소되는 것을 확인하기 위한 실험이다. Qualification에는 GPBCTRL과 GPBQSEL 레지스터를 사용했다. GPBCTRL은 샘플링 주기를 설정 가능한 레지스터이며 이번 실험에서는 0xFF 값을 가졌다. 이 경우 샘플링 주기는 $510 * T_{SYSCLKOUT}$ 값을 갖는다. GPBQSEL 레지스터를 통해 Input qualification type을 선택할 수 있었다. [실험 3-1]에서는 GPBQSEL의 값을 2로 설정하여 6개의 샘플을 이용했고, [실험 3-2]에서는 GPBQSEL의 값을 1로 설정하여 3개의 샘플을 이용했다. 그 결과 6개의 샘플을 사용하는 것이 3개의 샘플을 사용하는 것보다 노이즈 감소 효과가 좋다는 것을 확인할 수 있었다.

이번 실험을 통해 CCS를 통한 보드와의 연동, 사용법을 익힐 수 있었다. 또한 예제 코드를 통해 레지스터의 기능과 동작 조건 등을 확인하는 실험을 통해 GPIO 입력에 관한 이해를 높일 수 있었다.

4. 참고문헌

- 인터럽트
<https://ko.wikipedia.org/wiki/%EC%9D%B8%ED%84%B0%EB%9F%BD%ED%8A%B8>
- 채터링
<https://terms.naver.com/entry.nhn?docId=2763430&cid=50307&categoryId=50307>
- DSP 보드
<http://www.mcublog.co.kr/>
- 데이터 시트
TMS320F28335 Datasheet