Infineon TC275 Interrupt

Hyeongrae Kim
Architecture and Compiler for Embedded System LAB.
School of Electronics Engineering, KNU, KOREA

2021-05-11

ACE Lab.

0x1F 십진수로 ?

7번 비트 1로 설정하기 위해서 ?

7번 비트 0으로 설정하기 위해서는 ?

7번 비트 토글하기 위해서는 ?

7번 비트가 0인지 아닌지 판별하기 위해서는 ?

10진수 20을 16진수로 ?

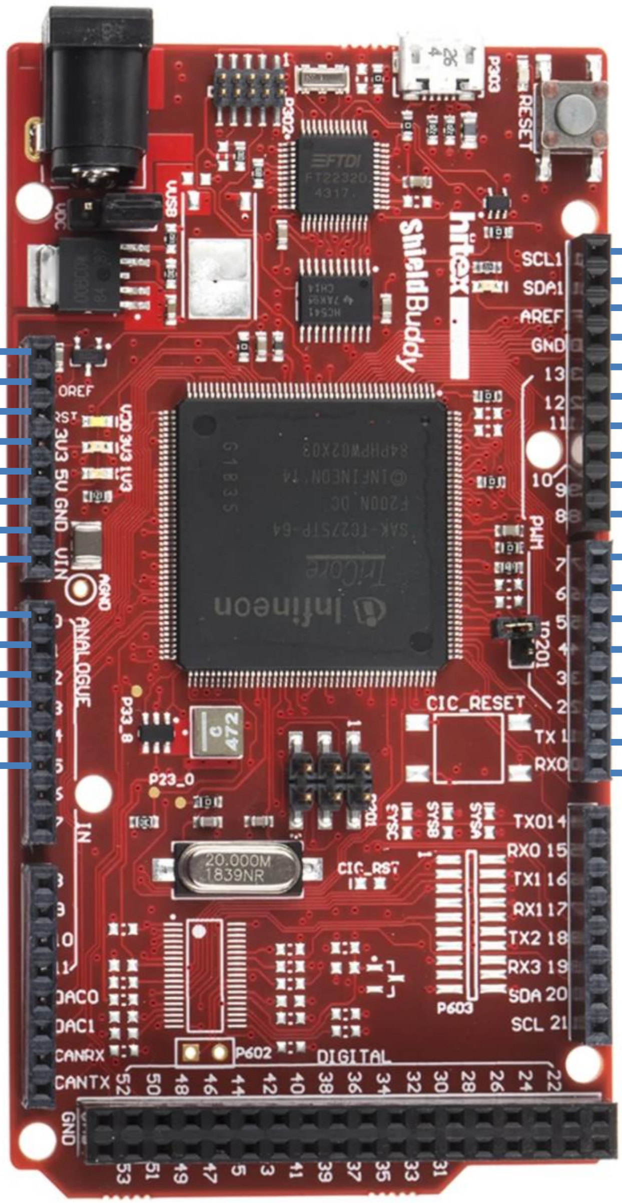
0b11111을 10진수로 ?

5번과 7번 비트 1로 설정하기 위해서 ?

5번과 7번 비트 0으로 설정하기 위해서는 ?

5번과 7번 비트 토글하기 위해서는 ?

Hitex ShieldBuddy TC275



ShieldBuddy TC275 Pin Layout	Arduino Uno Pin Layout
NC	NC
IOREF/VDD/5V	IOREF
RESET	Reset
+3.3V	+3V
VDD VIA RES521	+5V
GND	GND
GND	GND
VIN 9..12V	VIN
SAR4_7 AD0	A0|D14
SAR4_6 AD1	A1|D15
SAR4_5 AD2	A2|D16
SAR4_4 AD3	A3|D17
SAR3_1 AD4	A4|D18
SAR3_0 AD5	A5|D19

Arduino Uno Pin Layout	ShieldBuddy TC275 Pin Layout
D19	P13_1 SCL1
D18	P13_2 SDA1
AREF	AREF0
GND	GND
D13	P10_2 SPCK
D12	P10_1 MISO
~D11	P10_3 MOSI
~D10	P10_5 SS0/PWM10
~D9	P02_7 PWM9
D8	P02_6 PWM8
D7	P02_5 PWM7
~D6	P02_4 PWM6
~D5	P02_3 PWM5
D4	P10_4 SS1/PWM4
~D3	P02_1 PWM3
D2	P02_0 PWM2
D1/TX	P15_2 TX
D0/RX	P15_3 RX

ACE Lab.

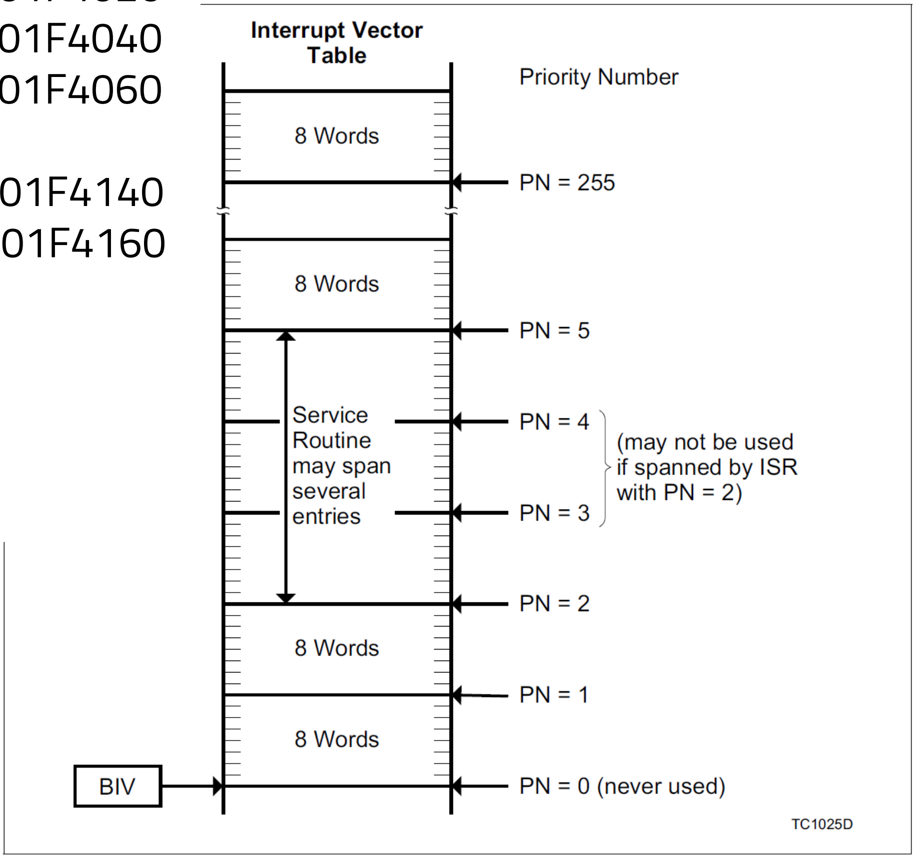
Interrupt System

Interrupt 발생 시 CPU는 사전에 정의된 주소로 JUMP
- 이때, Stack에 필요한 정보 저장
TC275 Base Interrupt Vector = 0x801F4000

0x00 Interrupt Vector Table: 0x801F4000 + 0x0A*0x20 = 0x801F4000
0x01 Interrupt Vector Table: 0x801F4000 + 0x0A*0x20 = 0x801F4020
0x02 Interrupt Vector Table: 0x801F4000 + 0x0A*0x20 = 0x801F4040
0x03 Interrupt Vector Table: 0x801F4000 + 0x0A*0x20 = 0x801F4060
~
0x0A Interrupt Vector Table: 0x801F4000 + 0x0A*0x20 = 0x801F4140
0x0B Interrupt Vector Table: 0x801F4000 + 0x0B*0x20 = 0x801F4160

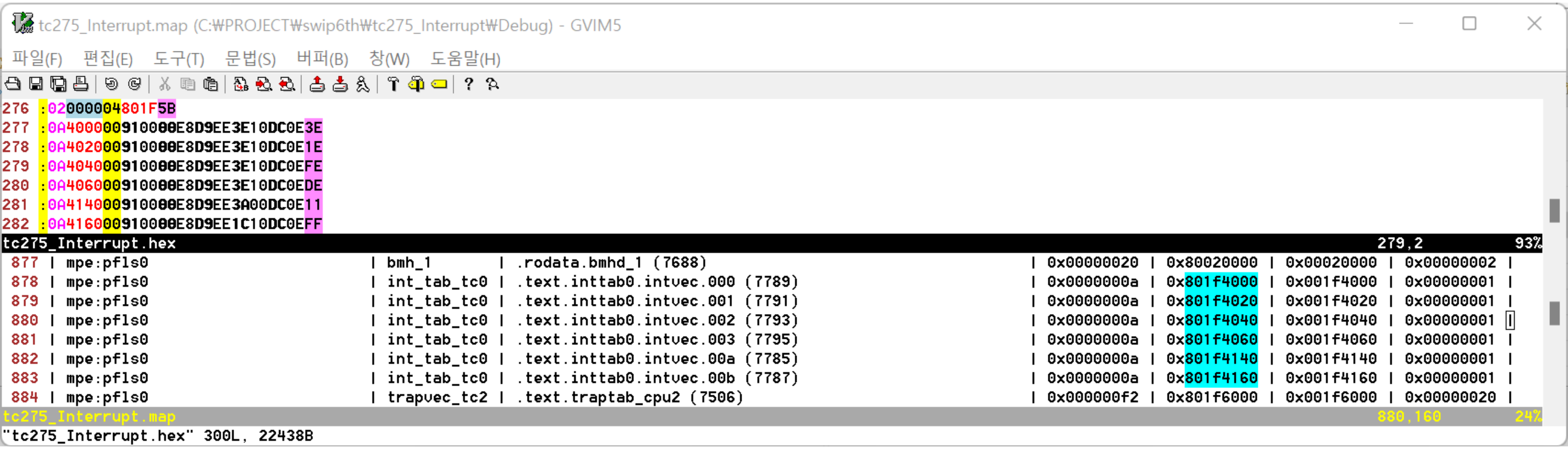
0x01 ~ 0xFF 까지 Interrupt Vector Allocation 가능함

낮은 Interrupt 일 수록 우선 순위(Priority)가 높음



ACE Lab.

Interrupt System


tc275_Interrupt.map (C:₩PROJECT₩swip6th₩tc275_Interrupt₩Debug) - GVIM5
파일(F) 편집(E) 도구(T) 문법(S) 버퍼(B) 창(W) 도움말(H)

```
:020000004801F5B
:0A400000910000E8D9EE3E10DC0E3E
:0A402000910000E8D9EE3E10DC0E1E
:0A404000910000E8D9EE3E10DC0EFE
:0A406000910000E8D9EE3E10DC0EDE
:0A414000910000E8D9EE3A00DC0E11
:0A416000910000E8D9EE1C10DC0EFF
tc275_Interrupt.hex                                                        279,2          93%
| mpe:pfls0        | bmh_1         | .rodata.bmhd_1 (7688)          | 0x00000020 | 0x80020000 | 0x00020000 | 0x00000002 |
| mpe:pfls0        | int_tab_tc0   | .text.inttab0.intvec.000 (7789) | 0x0000000a | 0x801f4000 | 0x001f4000 | 0x00000001 |
| mpe:pfls0        | int_tab_tc0   | .text.inttab0.intvec.001 (7791) | 0x0000000a | 0x801f4020 | 0x001f4020 | 0x00000001 |
| mpe:pfls0        | int_tab_tc0   | .text.inttab0.intvec.002 (7793) | 0x0000000a | 0x801f4040 | 0x001f4040 | 0x00000001 |
| mpe:pfls0        | int_tab_tc0   | .text.inttab0.intvec.003 (7795) | 0x0000000a | 0x801f4060 | 0x001f4060 | 0x00000001 |
| mpe:pfls0        | int_tab_tc0   | .text.inttab0.intvec.00a (7785) | 0x0000000a | 0x801f4140 | 0x001f4140 | 0x00000001 |
| mpe:pfls0        | int_tab_tc0   | .text.inttab0.intvec.00b (7787) | 0x0000000a | 0x801f4160 | 0x001f4160 | 0x00000001 |
| mpe:pfls0        | trapvec_tc2   | .text.traptab_cpu2 (7506)      | 0x000000f2 | 0x801f6000 | 0x001f6000 | 0x00000020 |
tc275_Interrupt.map                                                        880,160        24%
"tc275_Interrupt.hex" 300L, 22438B
```

0x0A Interrupt 발생 시
1. 0x801F4140 JUMP
2.1 수행할 내용이 8WORD(32bytes) 초과 시
    ISR로 JUMP
2.2 수행할 내용이 8WORD(32bytes) 미만 시
    수행 후 RFE(Return From Exception)

```
00000000801f4140:   movh.a      a14,#0x8000
00000000801f4144:   lea         a14,[a14]0x3a
00000000801f4148:   ji          a14
```

```
191                 void ERU0_ISR(void)
000000008000003a:   svlcx
193                   PORT10_OMR |= ((1<<PCL1) | (1<<PS1));
000000008000003e:   movh.a      a15,#0xf004
0000000080000042:   ld.w        d15,[a15]-0x4ffc
0000000080000046:   mov         d8,#0x2
0000000080000048:   addih       d8,d8,#0x2
000000008000004c:   or          d15,d8
000000008000004e:   movh.a      a15,#0xf004
0000000080000052:   st.w        [a15]-0x4ffc,d15
194                 }
0000000080000056:   rslcx
000000008000005a:   rfe
```

ACE Lab.

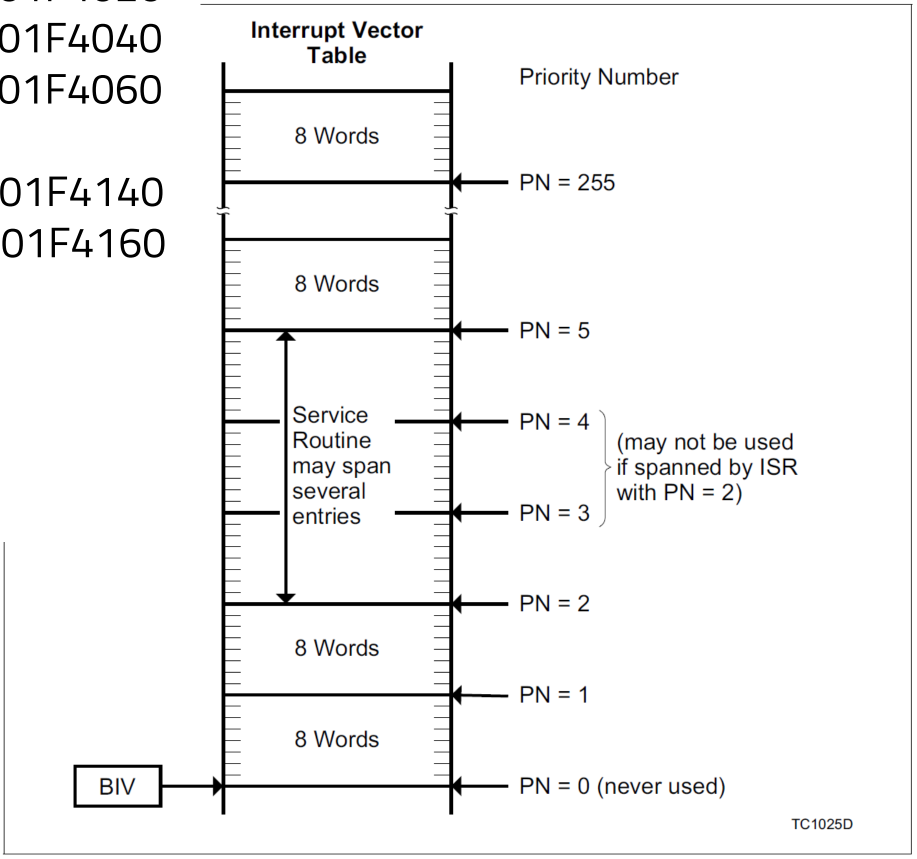
Interrupt System

Interrupt 발생 시 CPU는 사전에 정의된 주소로 JUMP
- 이때, Stack에 필요한 정보 저장
TC275 Base Interrupt Vector = 0x801F4000

0x00 Interrupt Vector Table: 0x801F4000 + 0x0A*0x20 = 0x801F4000
0x01 Interrupt Vector Table: 0x801F4000 + 0x0A*0x20 = 0x801F4020
0x02 Interrupt Vector Table: 0x801F4000 + 0x0A*0x20 = 0x801F4040
0x03 Interrupt Vector Table: 0x801F4000 + 0x0A*0x20 = 0x801F4060
~
0x0A Interrupt Vector Table: 0x801F4000 + 0x0A*0x20 = 0x801F4140
0x0B Interrupt Vector Table: 0x801F4000 + 0x0B*0x20 = 0x801F4160

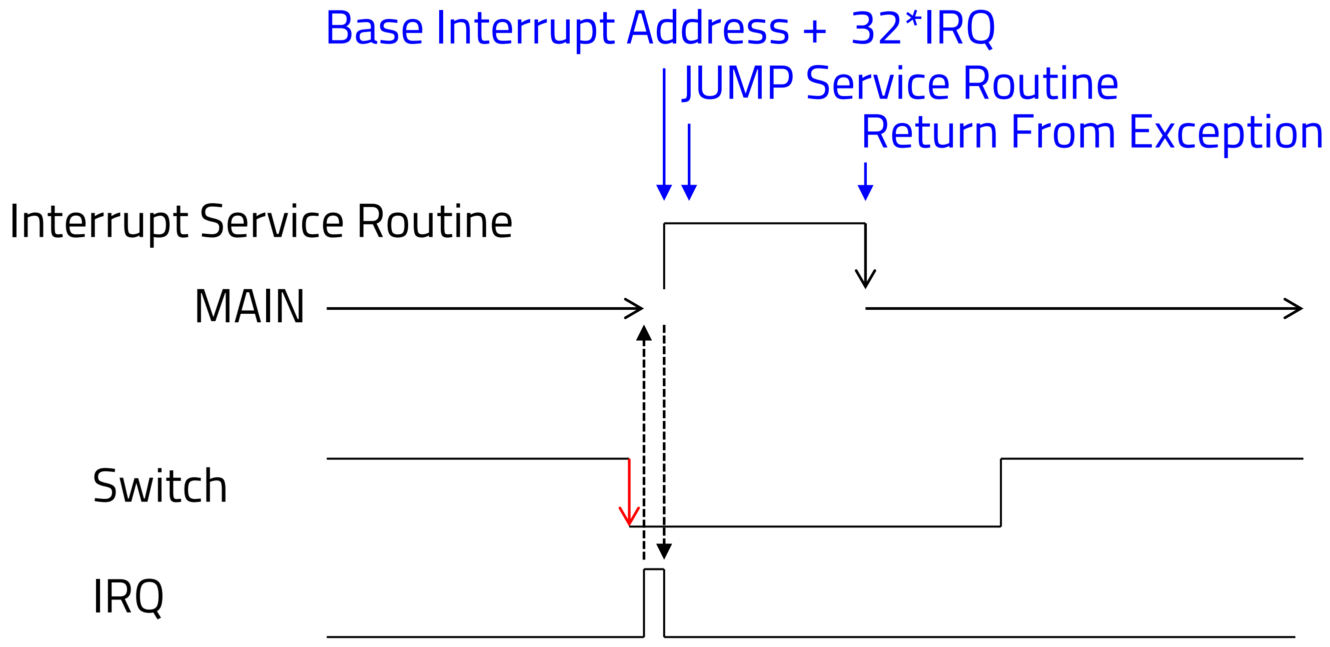
0x01 ~ 0xFF 까지 Interrupt Vector Allocation 가능함

낮은 Interrupt 일 수록 우선 순위(Priority)가 높음

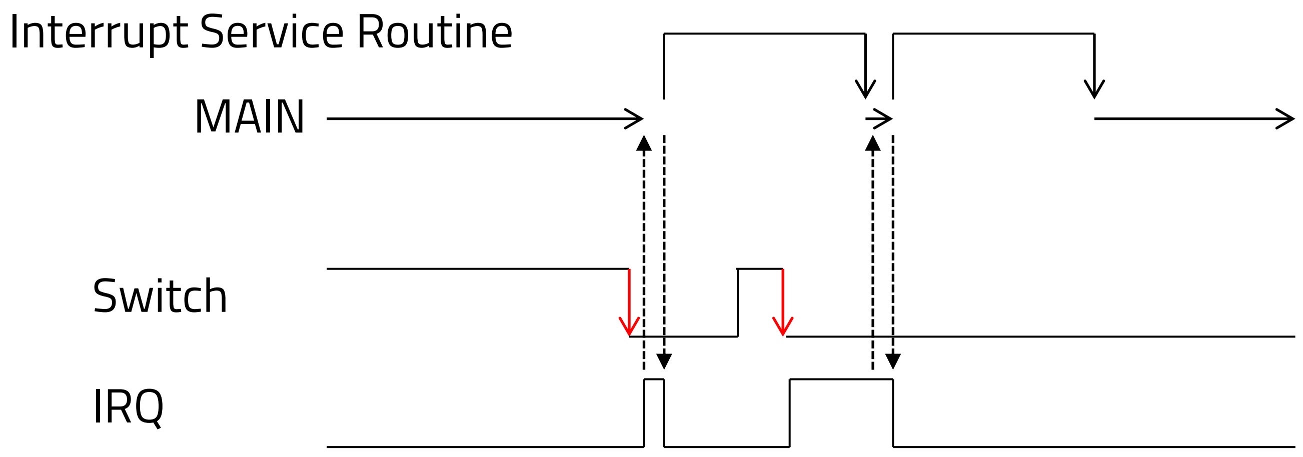


ACE Lab.

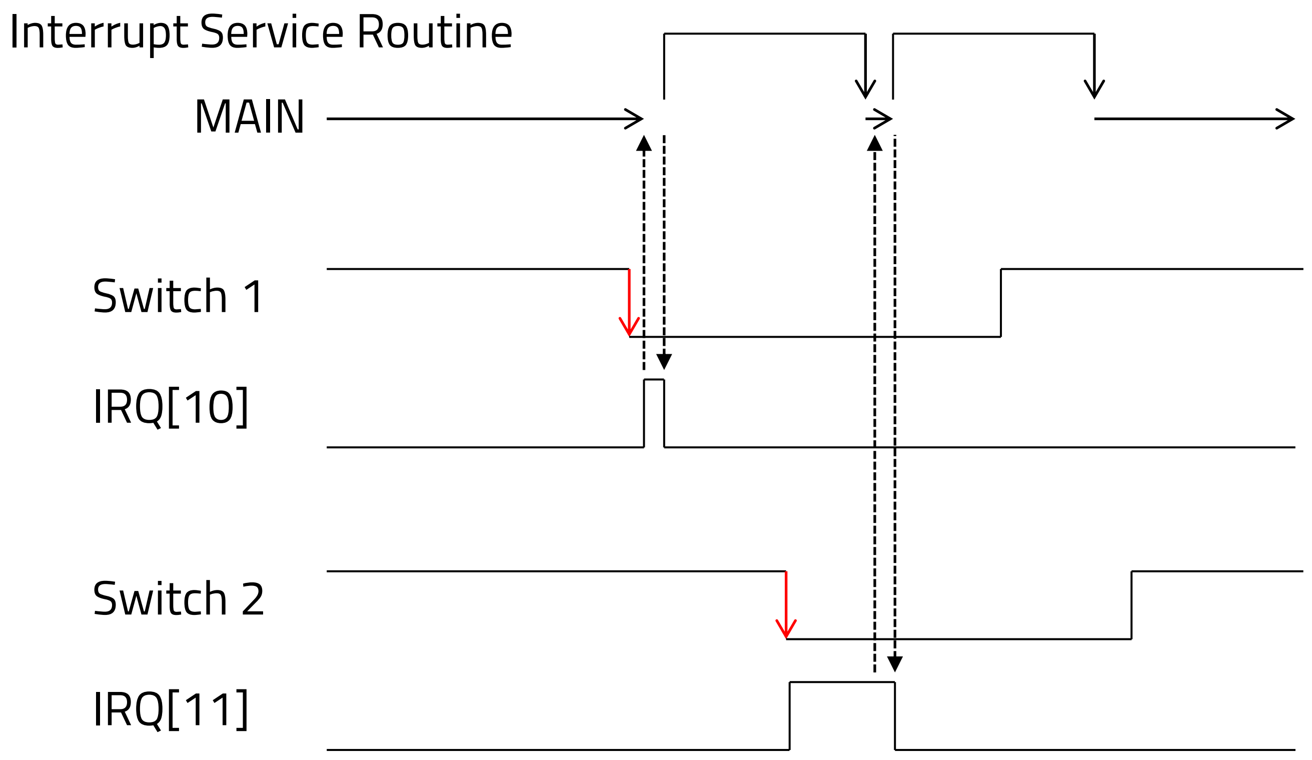
Interrupt System



Interrupt System

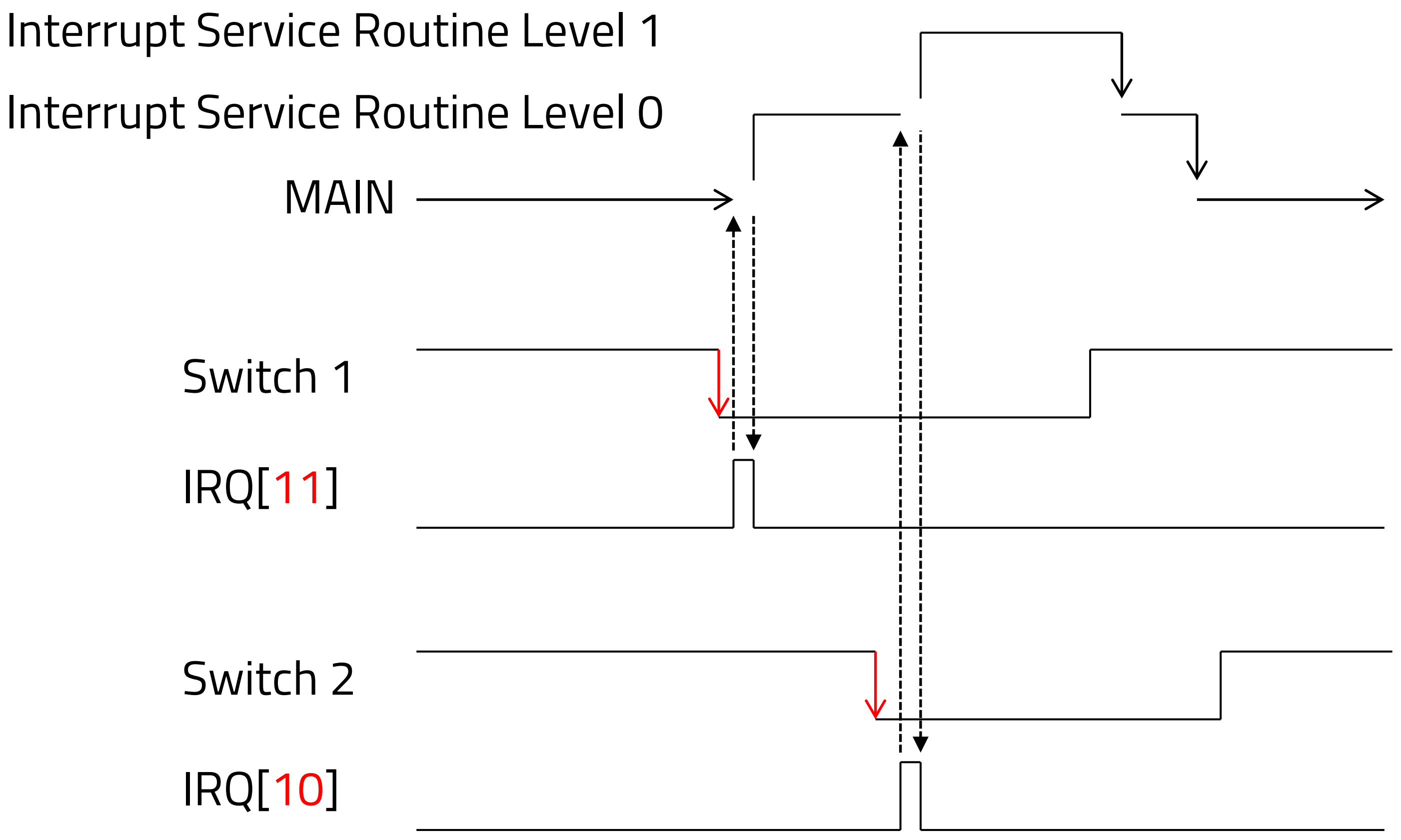


Interrupt System



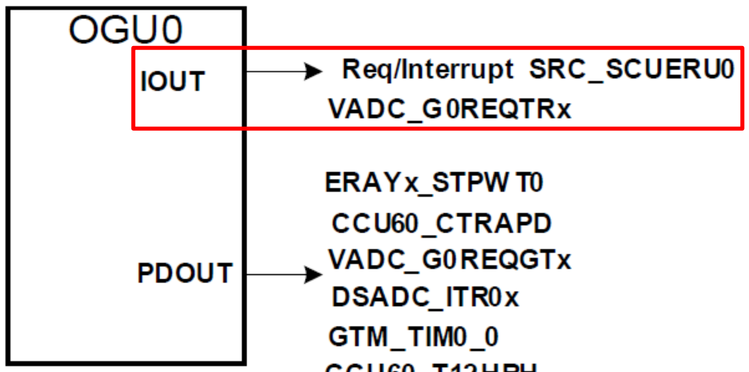
ACE Lab.

Interrupt System


Interrupt Service Routine Level 1
Interrupt Service Routine Level 0
MAIN
Switch 1
IRQ[11]
Switch 2
IRQ[10]

Interrupt Flow in TC275

2. ERU의 출력이 Interrupt Router (IR)의 입력이 됨
   - ✓ OGU에서 나온 출력이 연결된 Service Request Nodes (SRN)의 입력이 됨
   - ✓ SRN은 모든 Interrupt Control Units (ICU)에 연결되어 있고 Service Request Control Register (SRC) 설정을 통해 가능한 Service Provider (CPU0-2, DMA)에 매핑함
   - ✓ 각 ICU는 ICU에 매핑된 SRN의 Service Request 간의 인터럽트 중재를 처리함



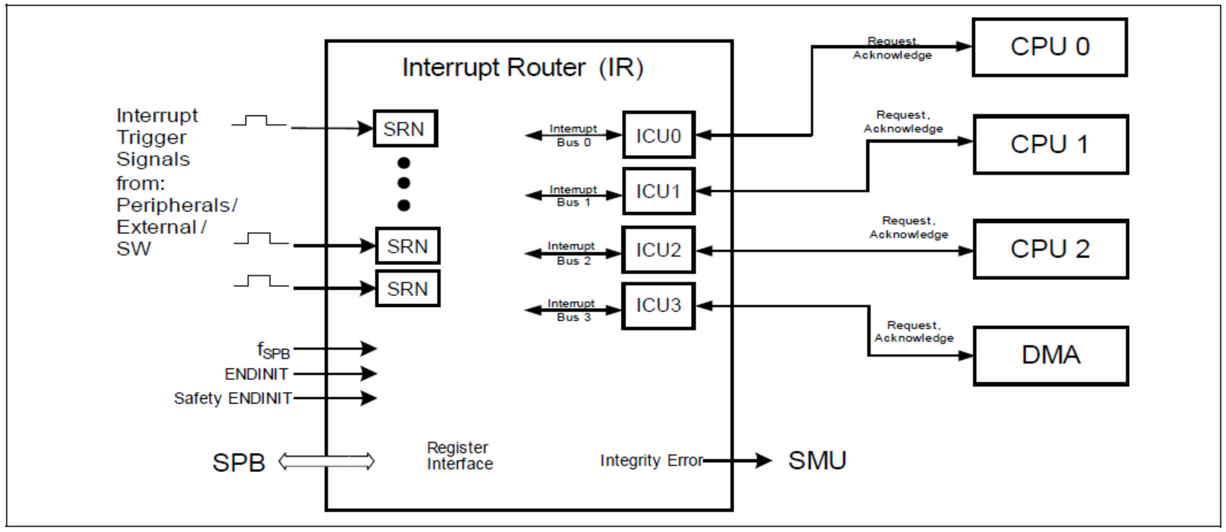


Figure 16-1 Block Diagram of the TC27x Interrupt System

ACE Lab.

Interrupt Flow in TC275

1. External Request Input을 External Request Unit (ERU)이 처리함
   - ✓ 각 Input Channel에 있는 ERS에서 4개의 가능한 입력 중 하나의 입력 벡터를 선택함

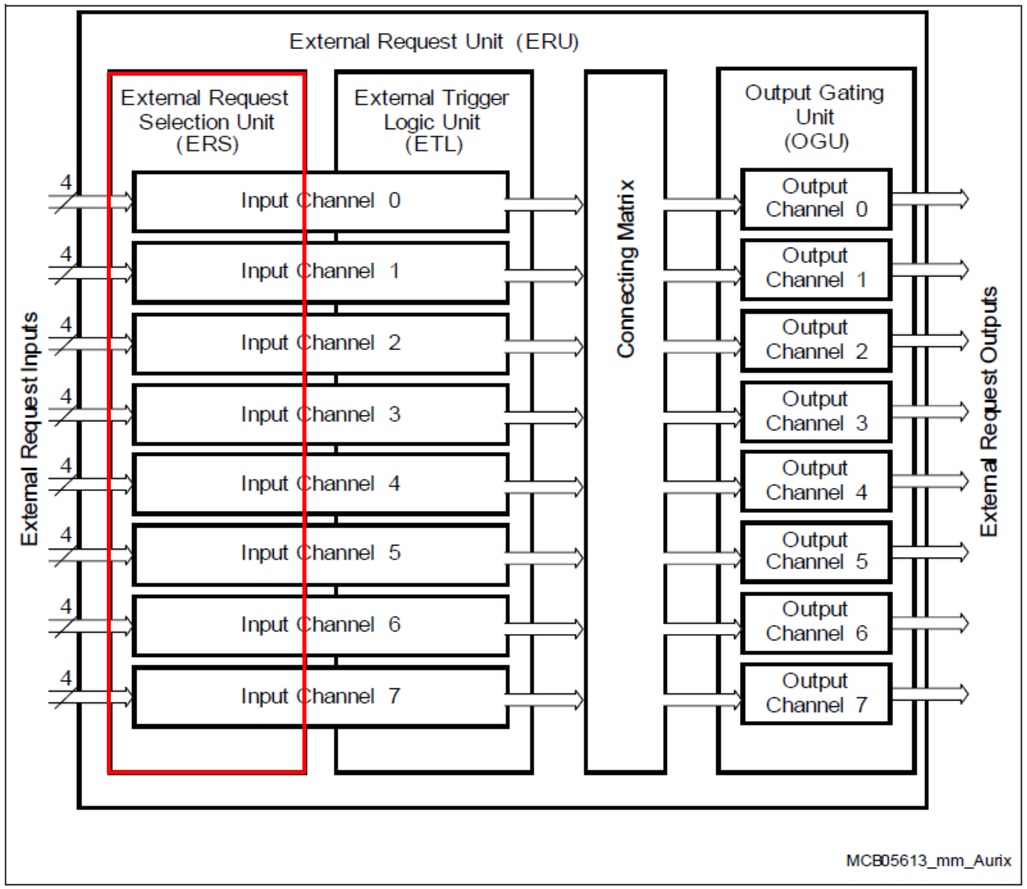


Figure 7-40 External Request Unit Overview

ACE Lab.

Interrupt Flow in TC275

1. External Request Input을 External Request Unit (ERU)이 처리함
   - ✓ 각 Input Channel에 있는 ETL이 지정된 엣지에서 입력을 트리거 이벤트로 전환함

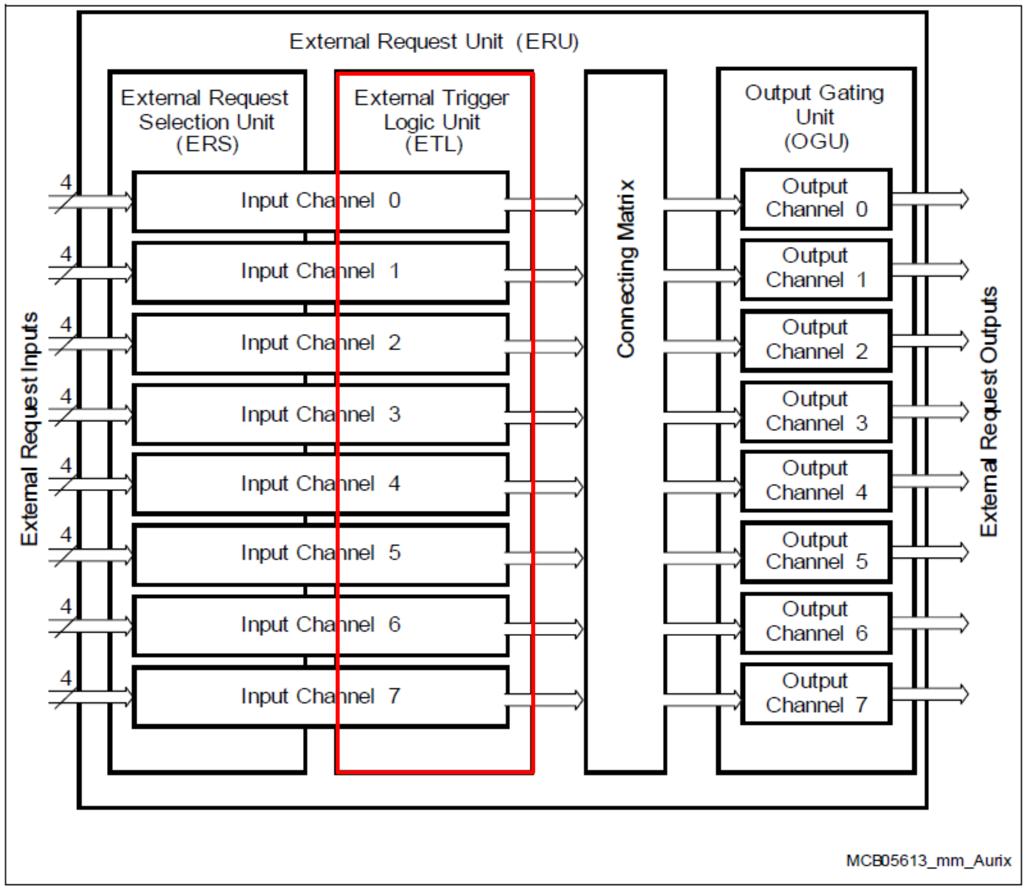


Figure 7-40 External Request Unit Overview

ACE Lab.

Interrupt Flow in TC275

1. External Request Input을 External Request Unit (ERU)이 처리함
   - ✓ Connecting Matrix는 입력 채널에서 생성된 이벤트를 출력 채널로 배포함

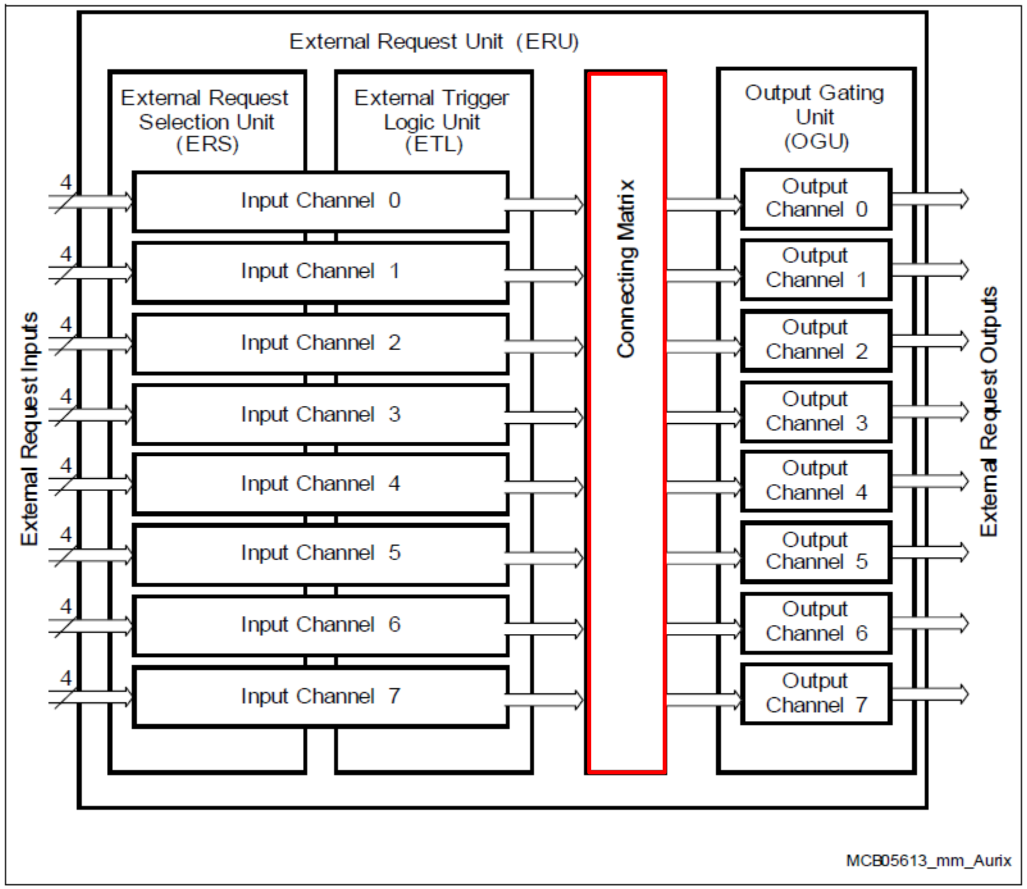


Figure 7-40 External Request Unit Overview

ACE Lab.

Interrupt Flow in TC275

1. External Request Input을 External Request Unit (ERU)이 처리함
   - ✓ Output Gating Unit은 Input Channel로 부터의 트리거 이벤트와 상태 정보를 조합하여 출력을 내보냄
   - ✓ 하나의 이벤트가 여러 채널에 갈 수 있고 여러 이벤트가 하나의 채널에서 패턴을 만들 수 있음

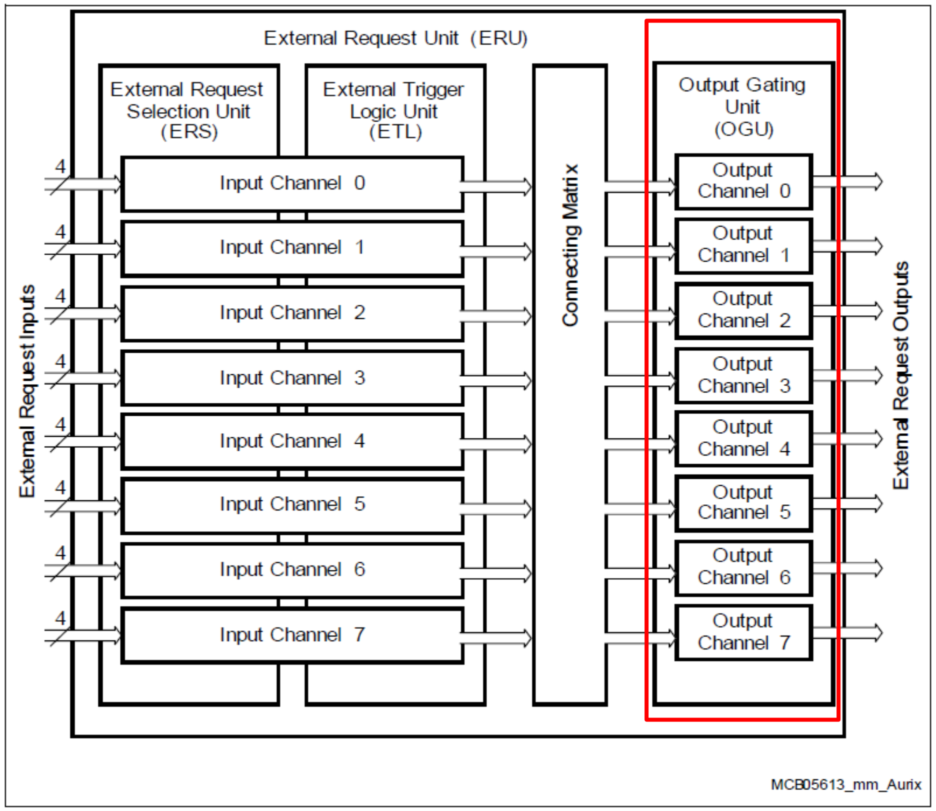


Figure 7-40 External Request Unit Overview

ACE Lab.

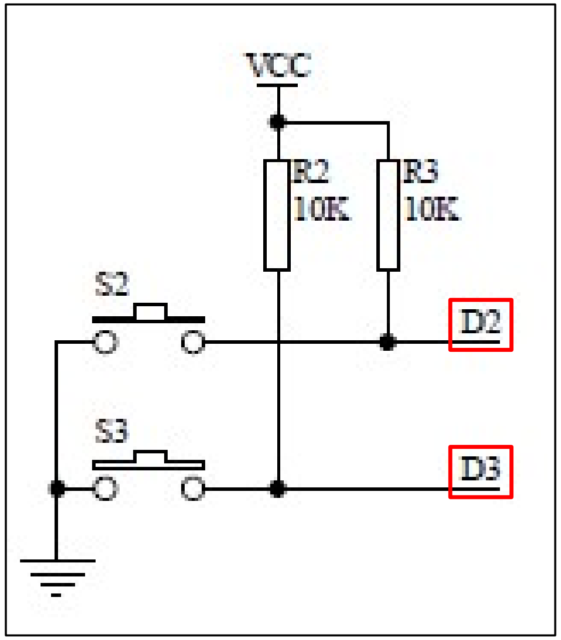
External Interrupt Example

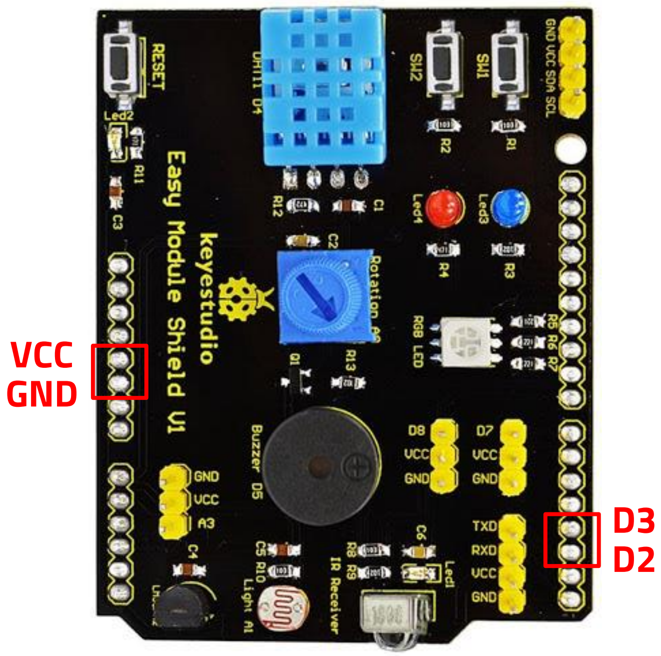
- External Interrupt를 사용하여 Switch를 눌렀을 때 LED Toggle

1. 새로운 예제를 위한 프로젝트를 생성한다.
2. 원하는 동작을 위해 레지스터와 메모리에 직접 접근해서 값을 써야한다.
3. Switch 사용을 위해 Board Schematic과 Datasheet에서 Switch 연결 정보를 파악한다.
4. External Interrupt를 사용하기 위해 Datasheet를 분석한다.
5. 분석 결과를 활용해 임베디드 프로그래밍을 한다.

External Interrupt Example

1. Switch 연결 정보 파악
   - ✓ Switch는 Easy Module Shield V1 확장 보드의 **Pin D2/D3**과 연결되어 있다.
   - ✓ Switch가 눌리면 연결된 Pin은 Low-level이 되고, Switch가 눌리지 않으면 연결된 Pin은 High-level이 된다.
   - ✓ 타겟 보드는 Easy Module Shield V1 확장 보드의 Pin D2/D3을 통해 Switch 입력을 받을 수 있다. (정상적인 Switch 동작을 위해 VCC 및 GND도 연결해야 한다.)



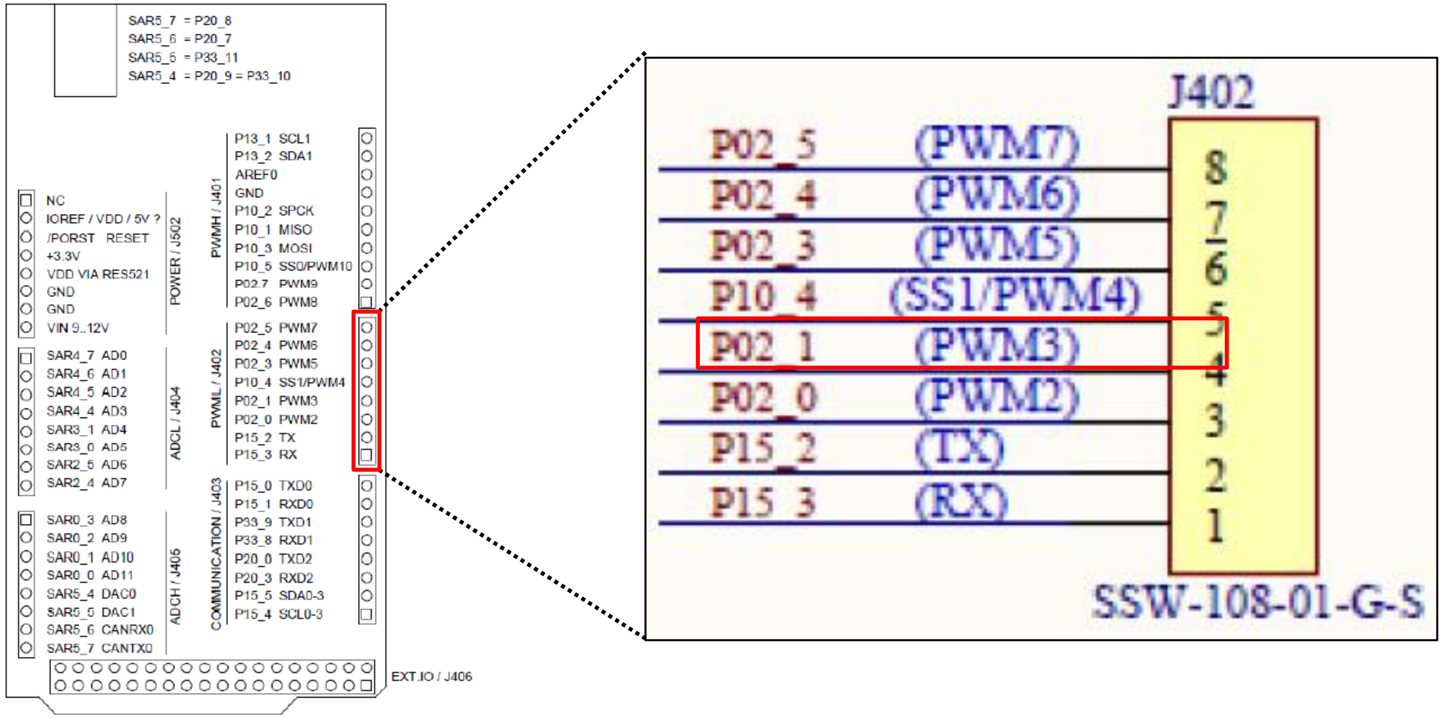


ACE Lab.

External Interrupt Example

1. Switch 연결 정보 파악

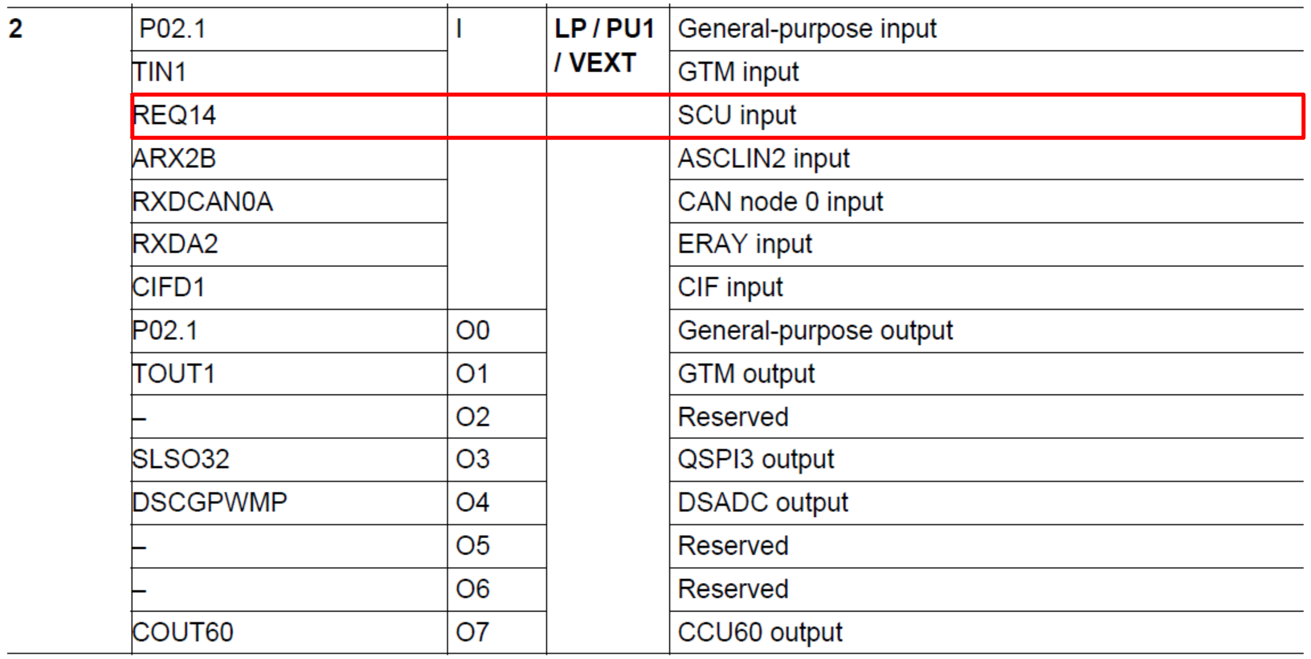
   ✓ TC275 보드의 Schematic과 Datasheet를 확인했을 때, Easy Module Shield V1 확장 보드의 **Pin D3**와 연결되는 IO는 **PORT02의 Pin 1**다.



External Interrupt Example

2. Data sheet 분석 : IO 설정

- ✓ Switch에 의한 External Interrupt를 사용하기 위해 연결된 Pin의 IO 설정이 필요하다.
- ✓ Switch가 연결된 PORT02 Pin 1은 External Interrupt를 관리하는 **SCU (System Control Unit) 내 ERU (External Request Unit)의 REQ14**와 연결되어 있다.
- ✓ 따라서, PORT02 Pin 1을 **Input**으로 설정하여 Switch 신호를 **ERU의 입력**으로 설정해야 한다.



2	P02.1	I	**LP / PU1 / VEXT**	General-purpose input
	TIN1			GTM input
	REQ14			SCU input
	ARX2B			ASCLIN2 input
	RXDCAN0A			CAN node 0 input
	RXDA2			ERAY input
	CIFD1			CIF input
	P02.1	O0		General-purpose output
	TOUT1	O1		GTM output
	–	O2		Reserved
	SLSO32	O3		QSPI3 output
	DSCGPWMP	O4		DSADC output
	–	O5		Reserved
	–	O6		Reserved
	COUT60	O7		CCU60 output

ACE Lab.

External Interrupt Example

2. Data sheet 분석 : PORT 설정 (1)
   - ✓ P02_IOCR Register는 PORT02의 Input/Output을 설정한다.
   - ✓ Switch가 PORT02의 Pin 1에 연결되어 있기 때문에 **P02_IOCR0 Register의 PC1 bits**를 설정한다.

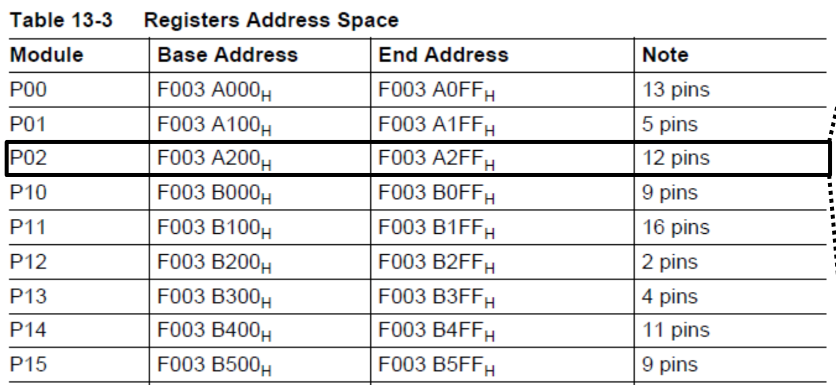


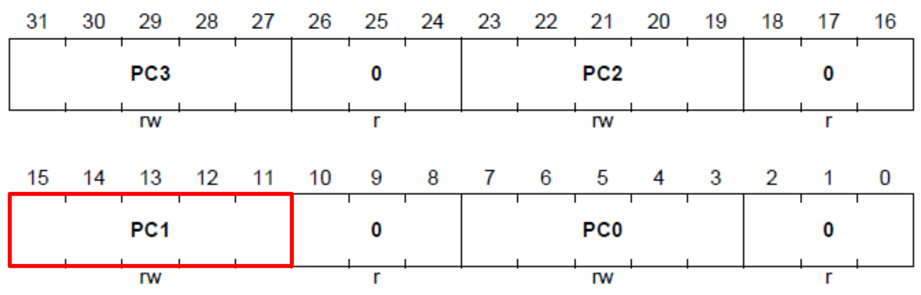
Table 13-3 Registers Address Space

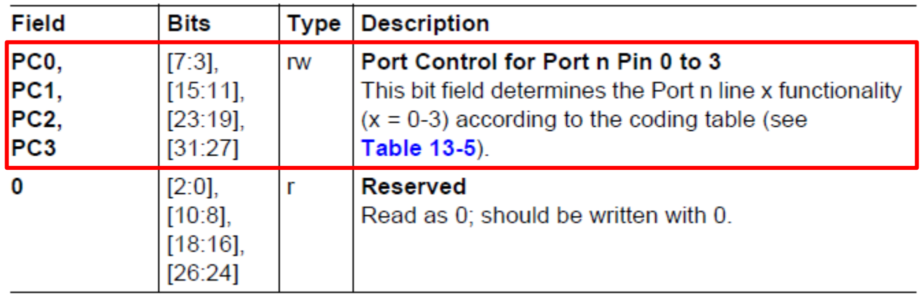
Module	Base Address	End Address	Note
P00	F003 A000$_H$	F003 A0FF$_H$	13 pins
P01	F003 A100$_H$	F003 A1FF$_H$	5 pins
P02	F003 A200$_H$	F003 A2FF$_H$	12 pins
P10	F003 B000$_H$	F003 B0FF$_H$	9 pins
P11	F003 B100$_H$	F003 B1FF$_H$	16 pins
P12	F003 B200$_H$	F003 B2FF$_H$	2 pins
P13	F003 B300$_H$	F003 B3FF$_H$	4 pins
P14	F003 B400$_H$	F003 B4FF$_H$	11 pins
P15	F003 B500$_H$	F003 B5FF$_H$	9 pins

P02_IOCR0 Register 주소: F003_A210h (F003A200h + 10h)

P02_IOCR0 Register 구조:

P02_IOCR0
Port 02 Input/Output Control Register 0 **(10$_H$)** **Reset Value: 1010 1010$_H$**





Field	Bits	Type	Description
PC0, PC1, PC2, PC3	[7:3], [15:11], [23:19], [31:27]	rw	**Port Control for Port n Pin 0 to 3** This bit field determines the Port n line x functionality (x = 0-3) according to the coding table (see Table 13-5).
0	[2:0], [10:8], [18:16], [26:24]	r	**Reserved** Read as 0; should be written with 0.

ACE Lab.

External Interrupt Example

2. Data sheet 분석 : PORT 설정 (2)
   - ✓ Easy Module Shield V1의 Switch는 pull-up device이다.
   - ✓ 따라서, PORT02의 Pin 1을 Input으로 설정할 때 **PC1 bits**를 **0XX10b**로 설정한다.

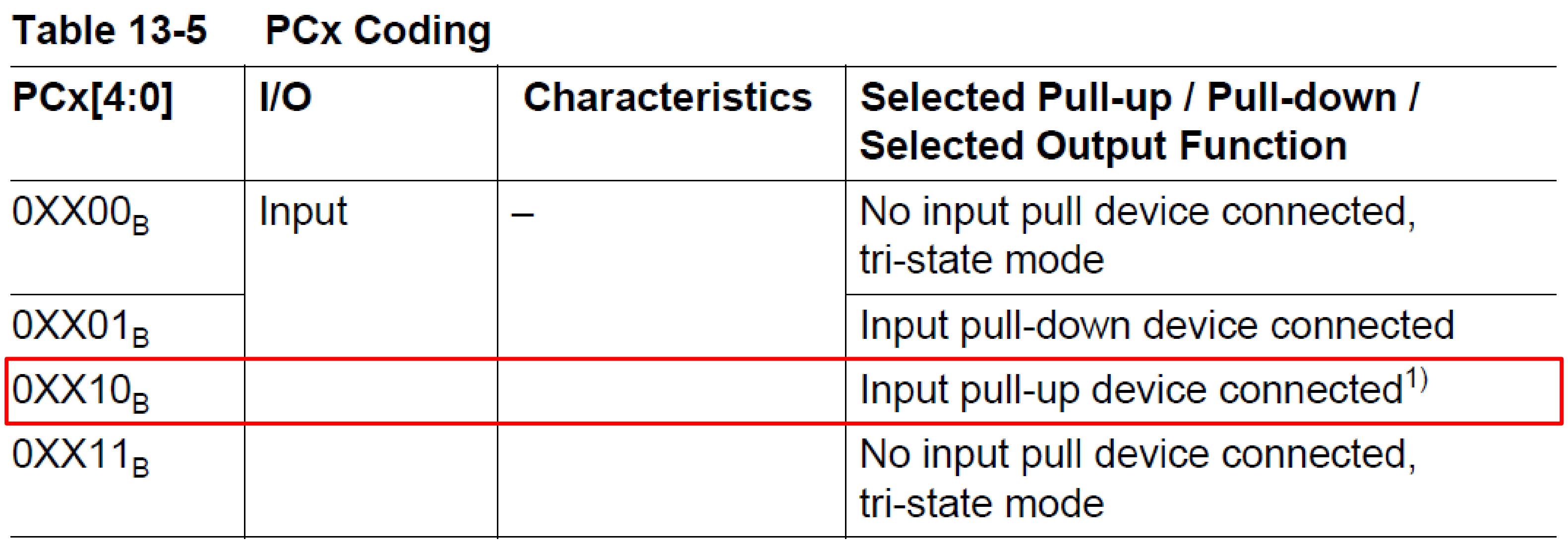


Table 13-5 PCx Coding

PCx[4:0]	I/O	Characteristics	Selected Pull-up / Pull-down / Selected Output Function
$0XX00_B$	Input	–	No input pull device connected, tri-state mode
$0XX01_B$			Input pull-down device connected
$0XX10_B$			Input pull-up device connected[1)]
$0XX11_B$			No input pull device connected, tri-state mode

7.4.1.2 ERU Input Pin Connections

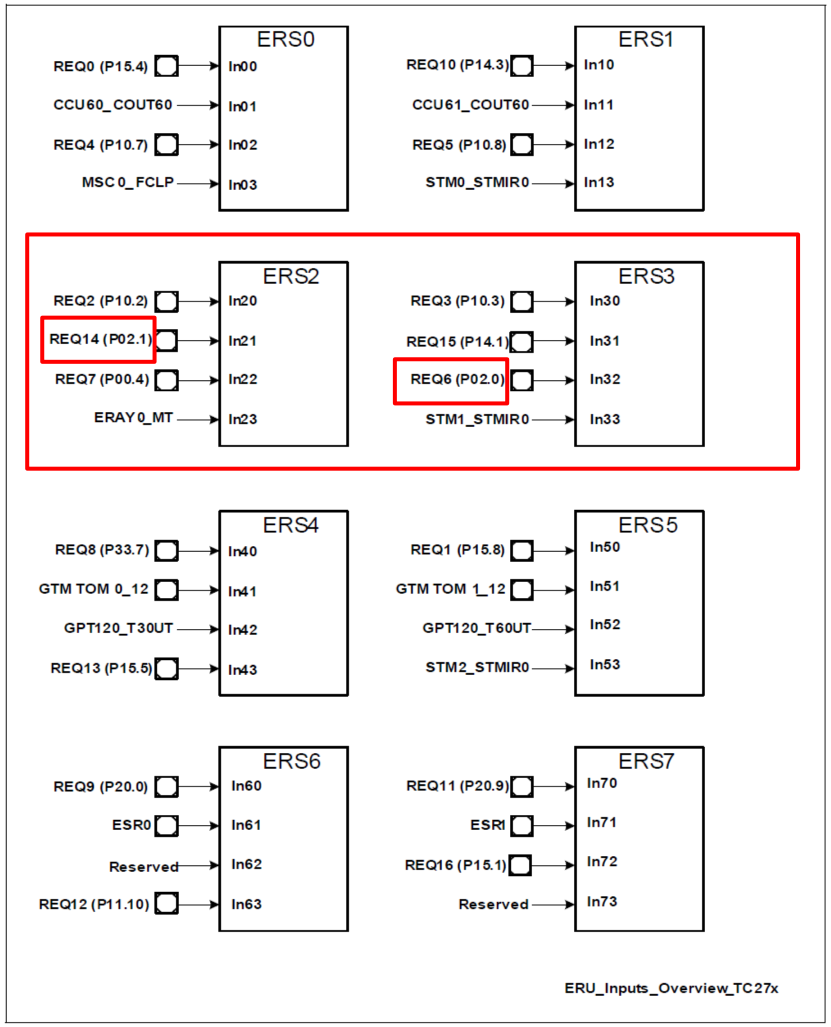


Figure 7-41 External Request Unit Input Connections for TC27x

External Interrupt Example

2. Data sheet 분석 : ERU External Input Channel 설정 (1)
   - ✓ SCU_EICR Register는 ERU의 External input Channel 0-7에 대한 설정을 한다.
   - ✓ 하나의 SCU_EICR Register는 2개의 Channel에 대한 설정을 한다.
     (SCU_EICR0: Channel 0-1, SCU_EICR1: Channel 2-3, SCU_EICR2: Channel 4-5, ...)
   - ✓ PORT02 Pin 1과 연결된 REQ14가 **ERU의 Channel 2 Input 1**과 연결되어 있기 때문에 **SCU_EICR1 Register**의 **INP0 bits / EIEN0 bit / FEN0 bit / EXIS0 bits**를 설정한다.

SCU_EICR1 Register 주소: F003_6214h (F0036000h + 214h)

SCU_EICR1 Register 구조:

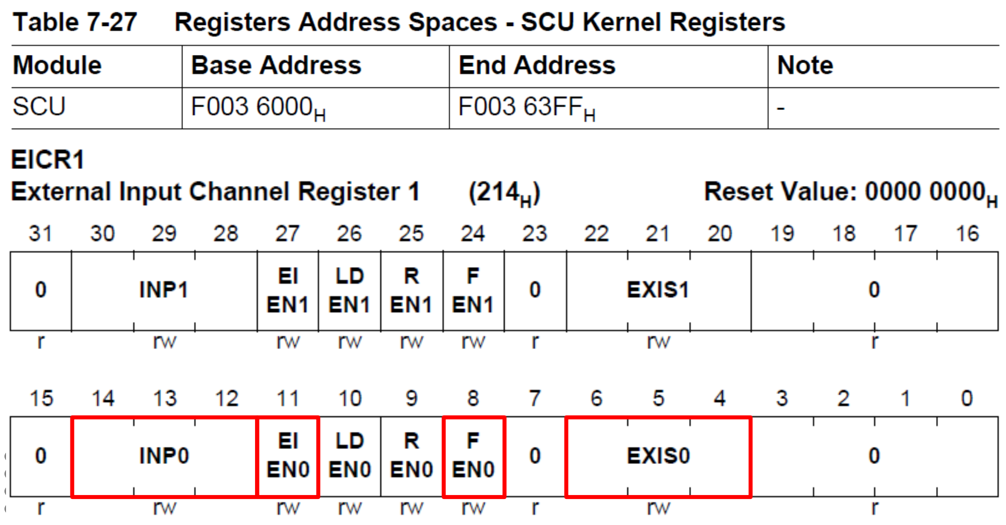


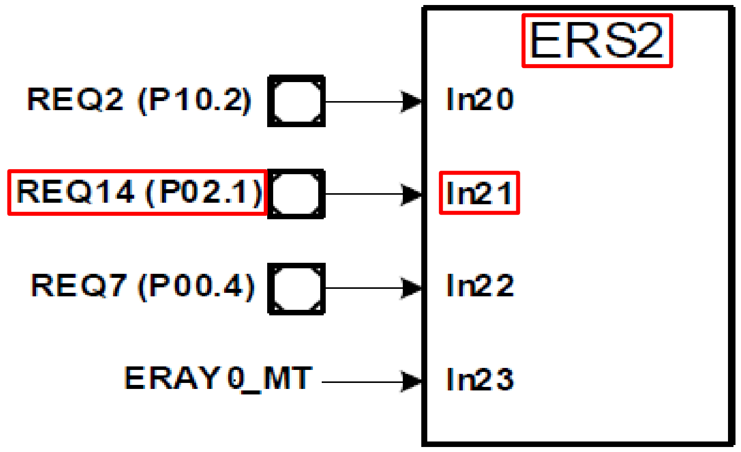
Table 7-27 Registers Address Spaces - SCU Kernel Registers

Module	Base Address	End Address	Note
SCU	F003 6000_H	F003 $63FF_H$	-

EICR1
External Input Channel Register 1 (214_H) Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	INP1			EIEN1	LDEN1	REN1	FEN1	0	EXIS1			0			
r	rw			rw	rw	rw	rw	r	rw			r			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	INP0			EIEN0	LDEN0	REN0	FEN0	0	EXIS0			0			
r	rw			rw	rw	rw	rw	r	rw			r			



7.4.1.8 External Request Unit Registers

The External Input Channel Registers EICRi (i=0 to 3) for the 4 external input channels contain bits to configure the external request selection ERS and the event trigger logic ETL.

EICR0
External Input Channel Register 0 (210_H) Reset Value: $0000\ 0000_H$
EICR1
External Input Channel Register 1 (214_H) Reset Value: $0000\ 0000_H$
EICR2
External Input Channel Register 2 (218_H) Reset Value: $0000\ 0000_H$
EICR3
External Input Channel Register 3 ($21C_H$) Reset Value: $0000\ 0000_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	INP1			EI EN1	LD EN1	R EN1	F EN1	0	EXIS1			0			
r	rw			rw	rw	rw	rw	r	rw			r			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	INP0			EI EN0	LD EN0	R EN0	F EN0	0	EXIS0			0			
r	rw			rw	rw	rw	rw	r	rw			r			

Field	Bits	Type	Description
EXIS0	[6:4]	rw	**External Input Selection 0** This bit field determines which input line is selected for Input Channel (2i). 000_B Input (2i) 0 is selected 001_B Input (2i) 1 is selected 010_B Input (2i) 2 is selected 011_B Input (2i) 3 is selected 100_B Reserved 101_B Reserved 110_B Reserved 111_B Reserved

Field	Bits	Type	Description
FEN0	8	rw	**Falling Edge Enable 0** This bit determines if the falling edge of Input Channel (2i) is used to set bit INTF(2i). 0_B The falling edge is not used 1_B The detection of a falling edge of Input Channel 0 generates a trigger event. INTF(2i) becomes set.
REN0	9	rw	**Rising Edge Enable 0** This bit determines if the rising edge of Input Channel (2*i) is used to set bit INTF(2i). 0_B The rising edge is not used 1_B The detection of a rising edge of Input Channel (2*i) generates a trigger event. INTF(2*i) becomes set
LDEN0	10	rw	**Level Detection Enable 0** This bit determines if bit INTF(2i) is cleared automatically if an edge of the input Input Channel (2i) is detected, which has not been selected (rising edge with REN0 = 0 or falling edge with FEN0 = 0). 0_B Bit INTF(2i) will not be cleared 1_B Bit INTF(2i) will be cleared
EIEN0	11	rw	**External Input Enable 0** This bit enables the generation of a trigger event for request channel (2i) (e.g. for interrupt generation) when a selected edge is detected. 0_B The trigger event is disabled 1_B The trigger event is enabled

Field	Bits	Type	Description
INP0	[14:12]	rw	**Input Node Pointer** This bit field determines the destination (output channel) for trigger event (2i) (if enabled by EIEN(2i)). 000_B An event from input ETL 2i triggers output OGU0 (signal TR(2i) 0) 001_B An event from input ETL 2i triggers output OGU1 (signal TR(2i) 1) 010_B An event from input ETL 2i triggers output OGU2 (signal TR(2i) 2) 011_B An event from input ETL 2i triggers output OGU3 (signal TR(2i) 3) 100_B An event from input ETL 2i triggers output OGU4 (signal TR(2i) 0) 101_B An event from input ETL 2i triggers output OGU5 (signal TR(2i) 0) 110_B An event from input ETL 2i triggers output OGU6 (signal TR(2i) 0) 111_B An event from input ETL 2i triggers output OGU7 (signal TR(2i) 0)

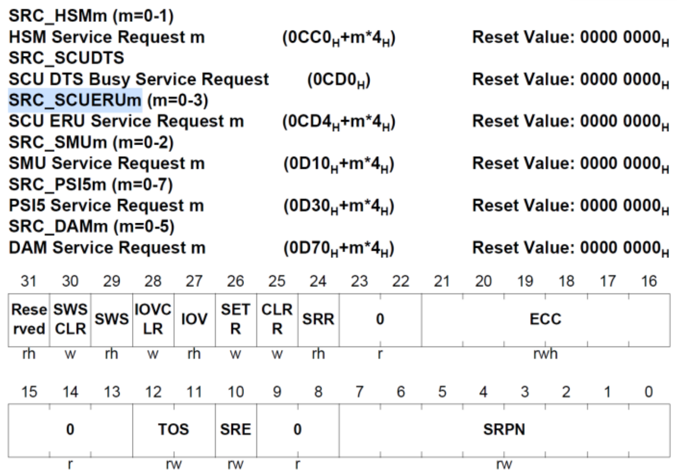
Page 1608

External Interrupt Example

2. Data sheet 분석 : ERU External Input Channel 설정 (2)

   ✓ ERU의 각 External Input Channel은 여러 개의 External R
   하나의 입력을 결정해야 한다.

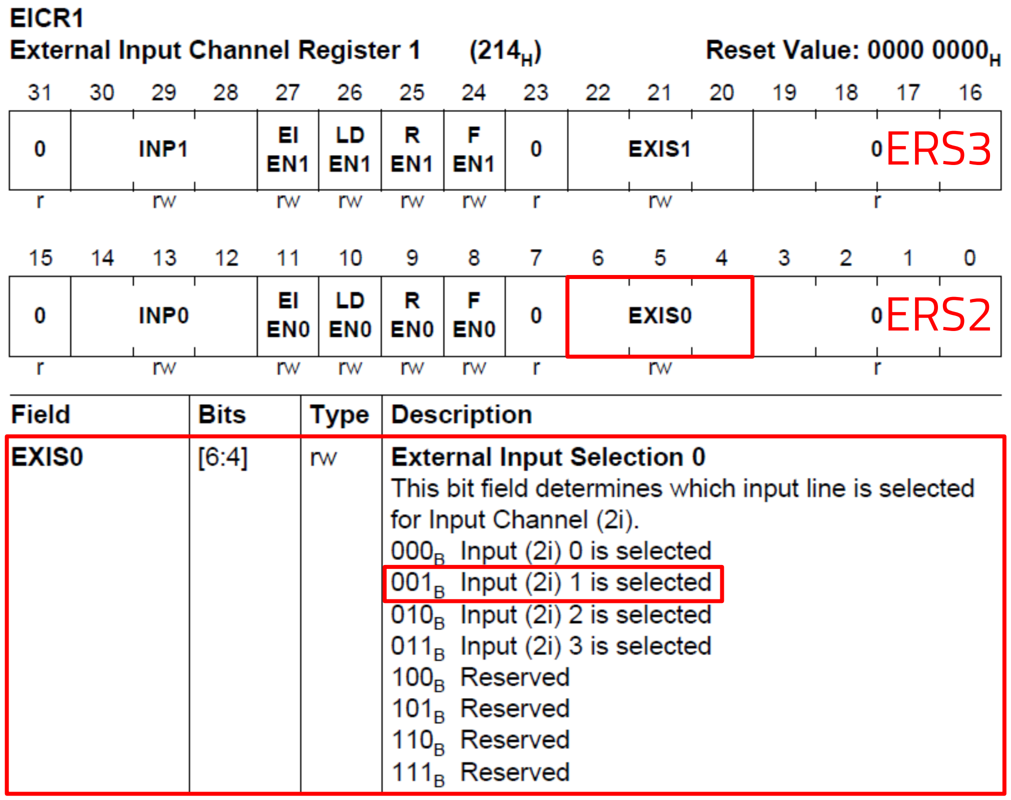
   ✓ PORT02 Pin 1이 ERU의 Channel 2 Input 1과 연결되어 있
   설정한다.



SRC_HSMm (m=0-1) HSM Service Request m	$(0CC0_H+m*4_H)$	Reset Value: $0000\ 0000_H$
SRC_SCUDTS SCU DTS Busy Service Request	$(0CD0_H)$	Reset Value: $0000\ 0000_H$
SRC_SCUERUm (m=0-3) SCU ERU Service Request m	$(0CD4_H+m*4_H)$	Reset Value: $0000\ 0000_H$
SRC_SMUm (m=0-2) SMU Service Request m	$(0D10_H+m*4_H)$	Reset Value: $0000\ 0000_H$
SRC_PSI5m (m=0-7) PSI5 Service Request m	$(0D30_H+m*4_H)$	Reset Value: $0000\ 0000_H$
SRC_DAMm (m=0-5) DAM Service Request m	$(0D70_H+m*4_H)$	Reset Value: $0000\ 0000_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Rese rved	SWS CLR	SWS	IOVC LR	IOV	SET R	CLR R	SRR	0			ECC				
rh	w	rh	w	rh	w	w	rh	r			rwh				

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0			TOS		SRE	0		SRPN							
r			rw		rw	r		rw							

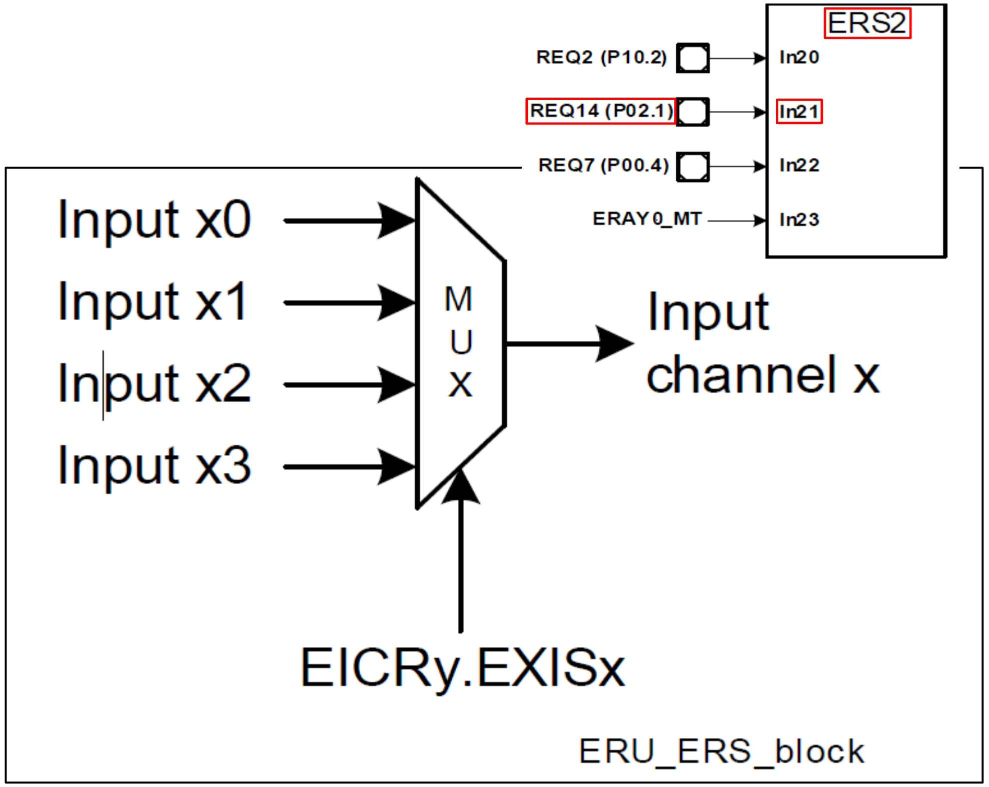


EICR1
External Input Channel Register 1 (214_H) Reset Value: $0000\ 0000_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	INP1			EI EN1	LD EN1	R EN1	F EN1	0	EXIS1			0 ERS3			
r	rw			rw	rw	rw	rw	r	rw			r			

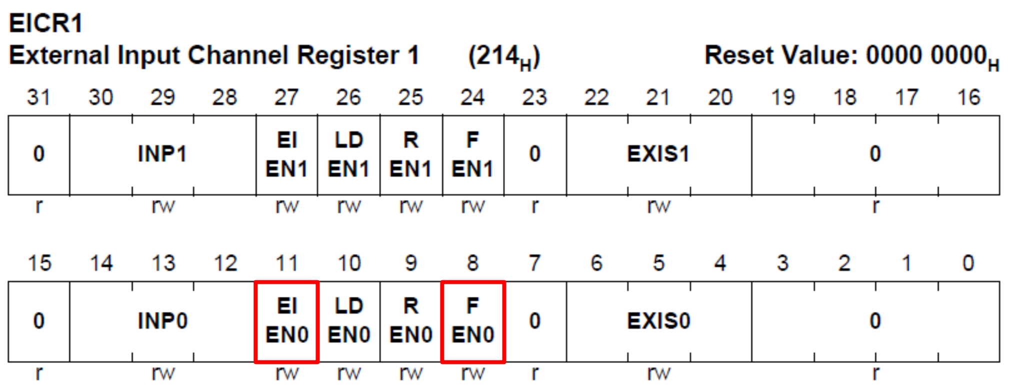
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	INP0			EI EN0	LD EN0	R EN0	F EN0	0	EXIS0			0 ERS2			
r	rw			rw	rw	rw	rw	r	rw			r			

Field	Bits	Type	Description
EXIS0	[6:4]	rw	**External Input Selection 0** This bit field determines which input line is selected for Input Channel (2i). 000_B Input (2i) 0 is selected 001_B Input (2i) 1 is selected 010_B Input (2i) 2 is selected 011_B Input (2i) 3 is selected 100_B Reserved 101_B Reserved 110_B Reserved 111_B Reserved



External Interrupt Example

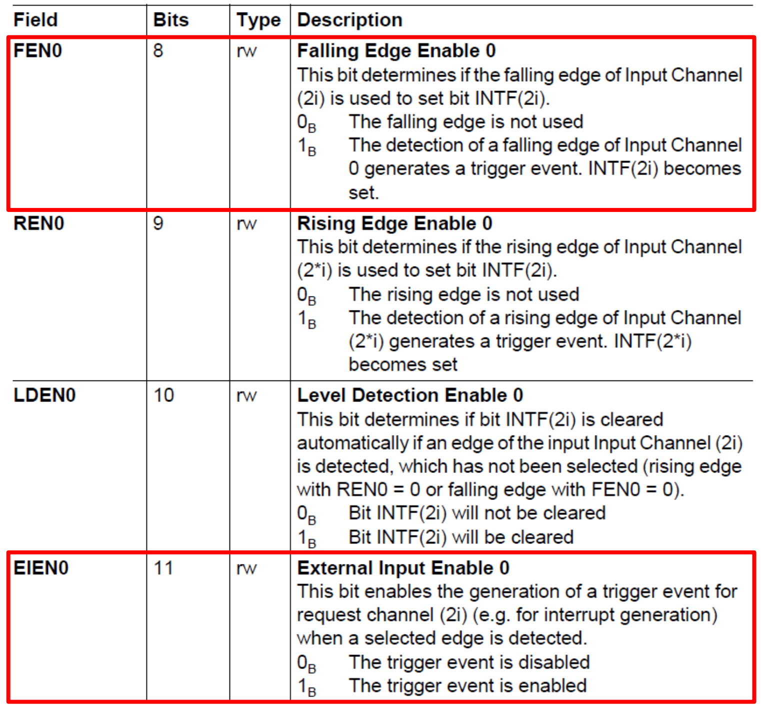
2. Data sheet 분석 : ERU External Input Channel 설정 (3)
   - ✓ Switch가 pull-up device이기 때문에 Switch가 눌렸을 때, 신호는 High-level에서 Low-level로 바뀌며 Falling edge가 발생한다.
   - ✓ 따라서, Falling edge가 검출되었을 때 트리거 신호 (for External Interrupt)를 생성하기 위해 **FEN0 bit**를 **1**로 설정한다.
   - ✓ 생성된 트리거 신호를 Enable 하기 위해 **EIEN0 bit**를 **1**로 설정한다.



EICR1
External Input Channel Register 1 (214$_H$) Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	INP1			EI EN1	LD EN1	R EN1	F EN1	0	EXIS1			0			
r	rw			rw	rw	rw	rw	r	rw			r			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	INP0			EI EN0	LD EN0	R EN0	F EN0	0	EXIS0			0			
r	rw			rw	rw	rw	rw	r	rw			r			



Field	Bits	Type	Description
FEN0	8	rw	**Falling Edge Enable 0** This bit determines if the falling edge of Input Channel (2i) is used to set bit INTF(2i). 0$_B$ The falling edge is not used 1$_B$ The detection of a falling edge of Input Channel 0 generates a trigger event. INTF(2i) becomes set.
REN0	9	rw	**Rising Edge Enable 0** This bit determines if the rising edge of Input Channel (2*i) is used to set bit INTF(2i). 0$_B$ The rising edge is not used 1$_B$ The detection of a rising edge of Input Channel (2*i) generates a trigger event. INTF(2*i) becomes set
LDEN0	10	rw	**Level Detection Enable 0** This bit determines if bit INTF(2i) is cleared automatically if an edge of the input Input Channel (2i) is detected, which has not been selected (rising edge with REN0 = 0 or falling edge with FEN0 = 0). 0$_B$ Bit INTF(2i) will not be cleared 1$_B$ Bit INTF(2i) will be cleared
EIEN0	11	rw	**External Input Enable 0** This bit enables the generation of a trigger event for request channel (2i) (e.g. for interrupt generation) when a selected edge is detected. 0$_B$ The trigger event is disabled 1$_B$ The trigger event is enabled

ACE Lab.

7.4.1.8 External Request Unit Registers

The External Input Channel Registers EICRi (i=0 to 3) for the 4 external input channels contain bits to configure the external request selection ERS and the event trigger logic ETL.

EICR0
External Input Channel Register 0 (210_H) Reset Value: $0000\ 0000_H$
EICR1
External Input Channel Register 1 (214_H) Reset Value: $0000\ 0000_H$
EICR2
External Input Channel Register 2 (218_H) Reset Value: $0000\ 0000_H$
EICR3
External Input Channel Register 3 ($21C_H$) Reset Value: $0000\ 0000_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	INP1			EIEN1	LDEN1	REN1	FEN1	0	EXIS1			0			
r	rw			rw	rw	rw	rw	r	rw			r			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	INP0			EIEN0	LDEN0	REN0	FEN0	0	EXIS0			0			
r	rw			rw	rw	rw	rw	r	rw			r			

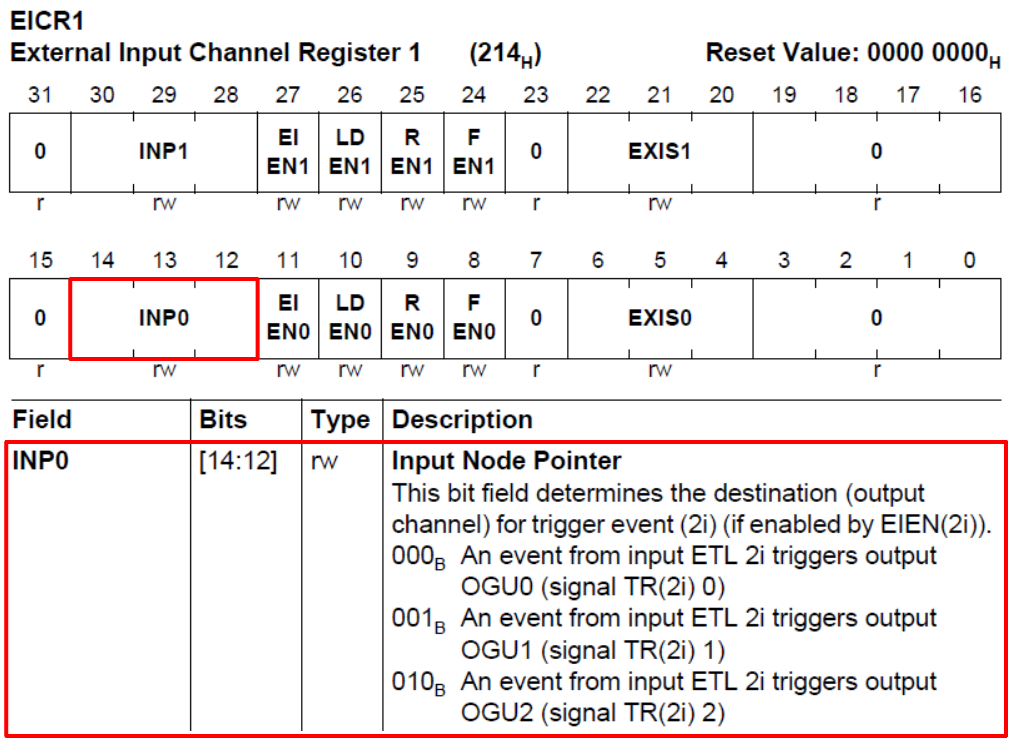
Field	Bits	Type	Description
EXIS0	[6:4]	rw	**External Input Selection 0** This bit field determines which input line is selected for Input Channel (2i). 000_B Input (2i) 0 is selected 001_B Input (2i) 1 is selected 010_B Input (2i) 2 is selected 011_B Input (2i) 3 is selected 100_B Reserved 101_B Reserved 110_B Reserved 111_B Reserved

Field	Bits	Type	Description
FEN0	8	rw	**Falling Edge Enable 0** This bit determines if the falling edge of Input Channel (2i) is used to set bit INTF(2i). 0_B The falling edge is not used 1_B The detection of a falling edge of Input Channel 0 generates a trigger event. INTF(2i) becomes set.
REN0	9	rw	**Rising Edge Enable 0** This bit determines if the rising edge of Input Channel (2*i) is used to set bit INTF(2i). 0_B The rising edge is not used 1_B The detection of a rising edge of Input Channel (2*i) generates a trigger event. INTF(2*i) becomes set
LDEN0	10	rw	**Level Detection Enable 0** This bit determines if bit INTF(2i) is cleared automatically if an edge of the input Input Channel (2i) is detected, which has not been selected (rising edge with REN0 = 0 or falling edge with FEN0 = 0). 0_B Bit INTF(2i) will not be cleared 1_B Bit INTF(2i) will be cleared
EIEN0	11	rw	**External Input Enable 0** This bit enables the generation of a trigger event for request channel (2i) (e.g. for interrupt generation) when a selected edge is detected. 0_B The trigger event is disabled 1_B The trigger event is enabled

Field	Bits	Type	Description
INP0	[14:12]	rw	**Input Node Pointer** This bit field determines the destination (output channel) for trigger event (2i) (if enabled by EIEN(2i)). 000_B An event from input ETL 2i triggers output OGU0 (signal TR(2i) 0) 001_B An event from input ETL 2i triggers output OGU1 (signal TR(2i) 1) 010_B An event from input ETL 2i triggers output OGU2 (signal TR(2i) 2) 011_B An event from input ETL 2i triggers output OGU3 (signal TR(2i) 3) 100_B An event from input ETL 2i triggers output OGU4 (signal TR(2i) 0) 101_B An event from input ETL 2i triggers output OGU5 (signal TR(2i) 0) 110_B An event from input ETL 2i triggers output OGU6 (signal TR(2i) 0) 111_B An event from input ETL 2i triggers output OGU7 (signal TR(2i) 0)

External Interrupt Example

2. Data sheet 분석 : ERU External Input Channel 설정 (4)
   - ✓ ERU의 각 External Input Channel에서 생성된 트리거 신호는 Connecting Matrix를 통해 Output Channel에 전달된다.
   - ✓ Output Channel은 입력 받은 트리거 신호를 Interrupt 신호로 전달할 수 있다.
   - ✓ 생성된 트리거 신호를 Output Channel 0에 전달하기 위해 **INP0 bits**를 **000b**로 설정한다.



EICR1
External Input Channel Register 1 (214_H) Reset Value: 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0		INP1		EIEN1	LDEN1	REN1	FEN1	0		EXIS1			0		
r		rw		rw	rw	rw	rw	r		rw			r		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0		INP0		EIEN0	LDEN0	REN0	FEN0	0		EXIS0			0		
r		rw		rw	rw	rw	rw	r		rw			r		

Field	Bits	Type	Description
INP0	[14:12]	rw	**Input Node Pointer** This bit field determines the destination (output channel) for trigger event (2i) (if enabled by EIEN(2i)). 000_B An event from input ETL 2i triggers output OGU0 (signal TR(2i) 0) 001_B An event from input ETL 2i triggers output OGU1 (signal TR(2i) 1) 010_B An event from input ETL 2i triggers output OGU2 (signal TR(2i) 2)

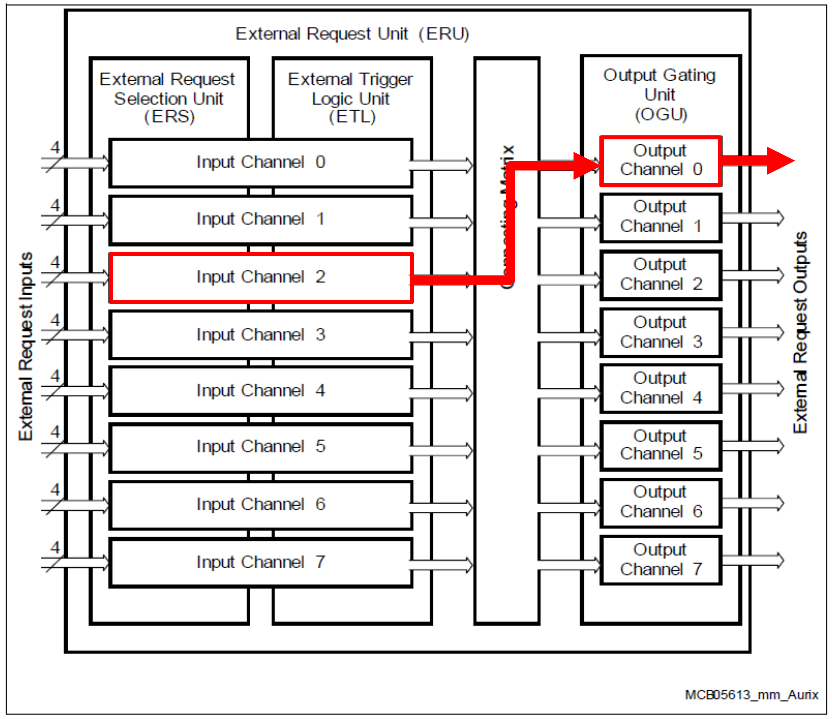


Figure 7-40 External Request Unit Overview

External Interrupt Example

2. Data sheet 분석 : ERU Flag Gating 설정 (1)
   - ✓ SCU_IGCR Register는 ERU의 Output Channel 0-7에 대한 설정을 한다.
   - ✓ 하나의 SCU_IGCR Register는 2개의 Channel에 대한 설정을 한다.
     (SCU_IGCR0: Channel 0-1, SCU_IGCR1: Channel 2-3, SCU_IGCR2: Channel 4-5, ...)
   - ✓ External Input Channel 2에서 생성된 트리거 신호가 Output Channel 0에 전달되기 때문에 **SCU_IGCR0 Register**의 **IGP0 bits**를 설정한다.

SCU_IGCR0 Register 주소: F003_622Ch (F0036000h + 22Ch)

SCU_IGCR0 Register 구조:

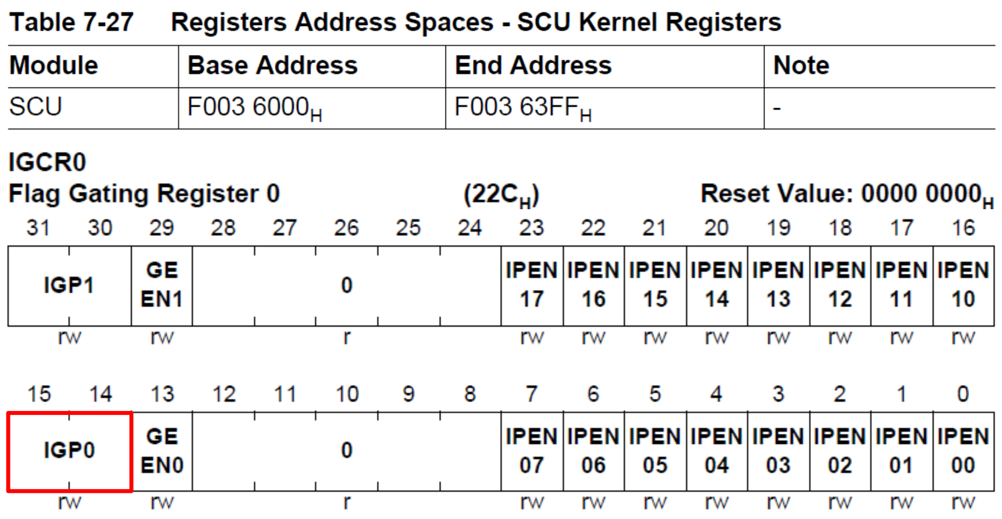


Table 7-27 Registers Address Spaces - SCU Kernel Registers

Module	Base Address	End Address	Note
SCU	F003 6000$_H$	F003 63FF$_H$	-

IGCR0
Flag Gating Register 0 (22C$_H$) **Reset Value: 0000 0000$_H$**

31-30	29	28-24	23	22	21	20	19	18	17	16
IGP1	GEEN1	0	IPEN17	IPEN16	IPEN15	IPEN14	IPEN13	IPEN12	IPEN11	IPEN10
rw	rw	r	rw	rw	rw	rw	rw	rw	rw	rw

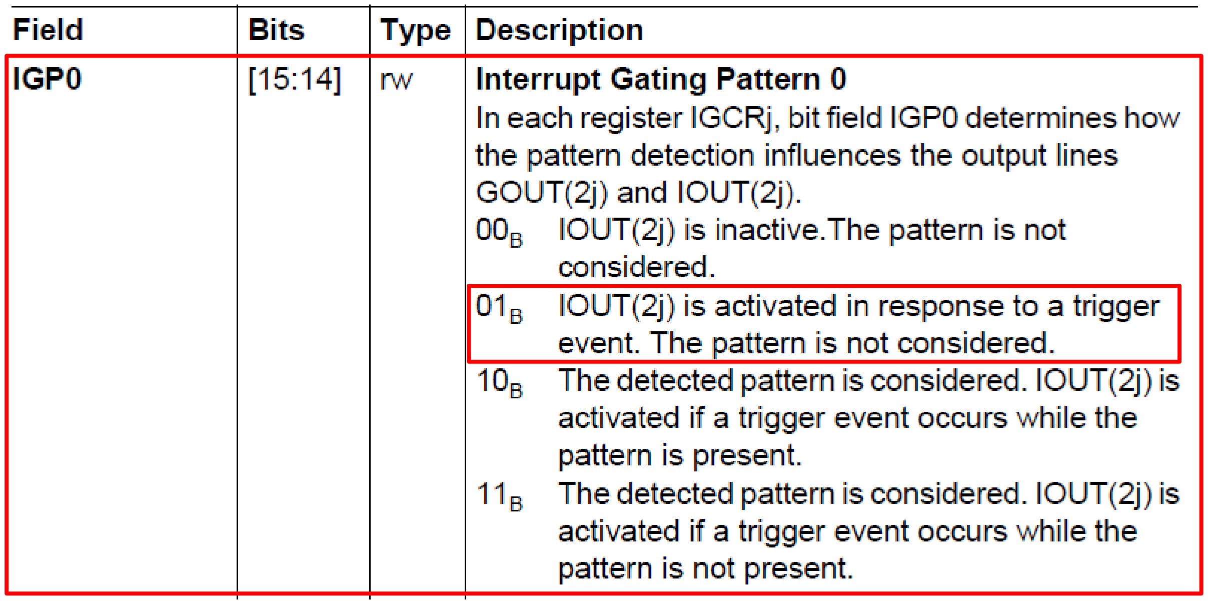
15-14	13	12-8	7	6	5	4	3	2	1	0
IGP0	GEEN0	0	IPEN07	IPEN06	IPEN05	IPEN04	IPEN03	IPEN02	IPEN01	IPEN00
rw	rw	r	rw	rw	rw	rw	rw	rw	rw	rw

ACE Lab.

External Interrupt Example

2. Data sheet 분석 : ERU Flag Gating 설정 (2)

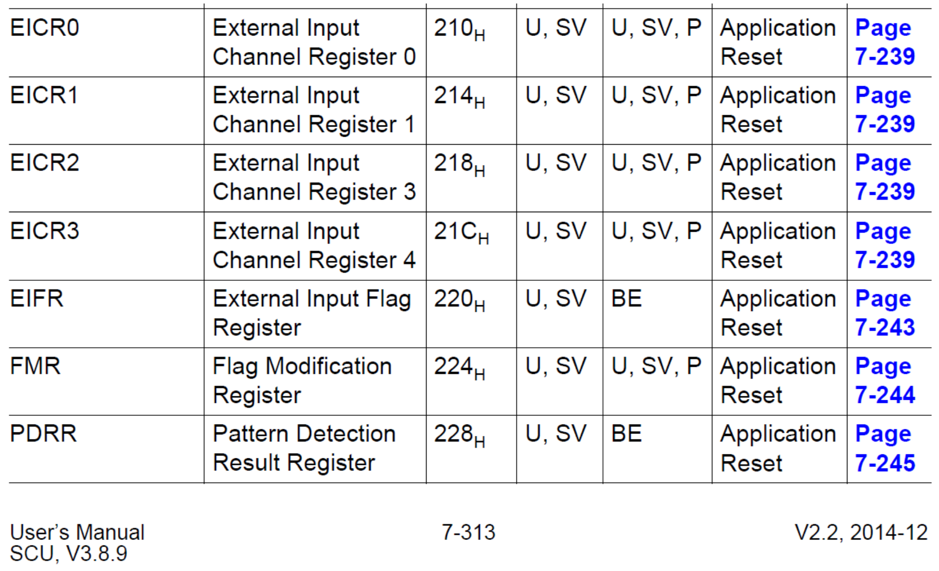
✓ 트리거 신호를 IOUT (for External Interrupt)으로 출력하기 위해 **IGP0 bits**를 **01b**로 설정한다.



Field	Bits	Type	Description
IGP0	[15:14]	rw	**Interrupt Gating Pattern 0** In each register IGCRj, bit field IGP0 determines how the pattern detection influences the output lines GOUT(2j) and IOUT(2j). 00_B IOUT(2j) is inactive.The pattern is not considered. 01_B IOUT(2j) is activated in response to a trigger event. The pattern is not considered. 10_B The detected pattern is considered. IOUT(2j) is activated if a trigger event occurs while the pattern is present. 11_B The detected pattern is considered. IOUT(2j) is activated if a trigger event occurs while the pattern is not present.

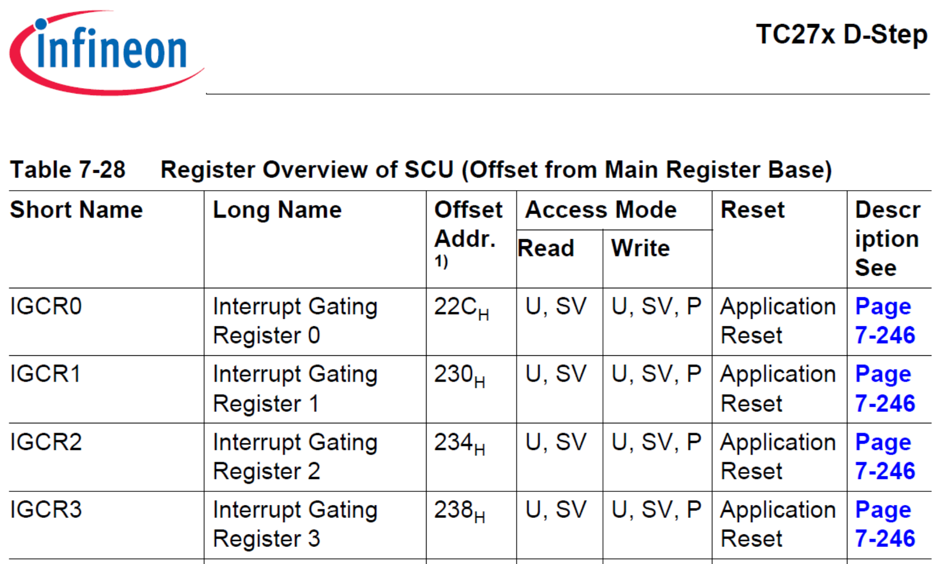
ACE Lab.

External Interrupt Example



EICR0	External Input Channel Register 0	210_H	U, SV	U, SV, P	Application Reset	Page 7-239
EICR1	External Input Channel Register 1	214_H	U, SV	U, SV, P	Application Reset	Page 7-239
EICR2	External Input Channel Register 3	218_H	U, SV	U, SV, P	Application Reset	Page 7-239
EICR3	External Input Channel Register 4	$21C_H$	U, SV	U, SV, P	Application Reset	Page 7-239
EIFR	External Input Flag Register	220_H	U, SV	BE	Application Reset	Page 7-243
FMR	Flag Modification Register	224_H	U, SV	U, SV, P	Application Reset	Page 7-244
PDRR	Pattern Detection Result Register	228_H	U, SV	BE	Application Reset	Page 7-245

User's Manual 7-313 V2.2, 2014-12
SCU, V3.8.9



TC27x D-Step

Table 7-28 Register Overview of SCU (Offset from Main Register Base)

Short Name	Long Name	Offset Addr. 1)	Access Mode		Reset	Description See
			Read	Write		
IGCR0	Interrupt Gating Register 0	$22C_H$	U, SV	U, SV, P	Application Reset	Page 7-246
IGCR1	Interrupt Gating Register 1	230_H	U, SV	U, SV, P	Application Reset	Page 7-246
IGCR2	Interrupt Gating Register 2	234_H	U, SV	U, SV, P	Application Reset	Page 7-246
IGCR3	Interrupt Gating Register 3	238_H	U, SV	U, SV, P	Application Reset	Page 7-246

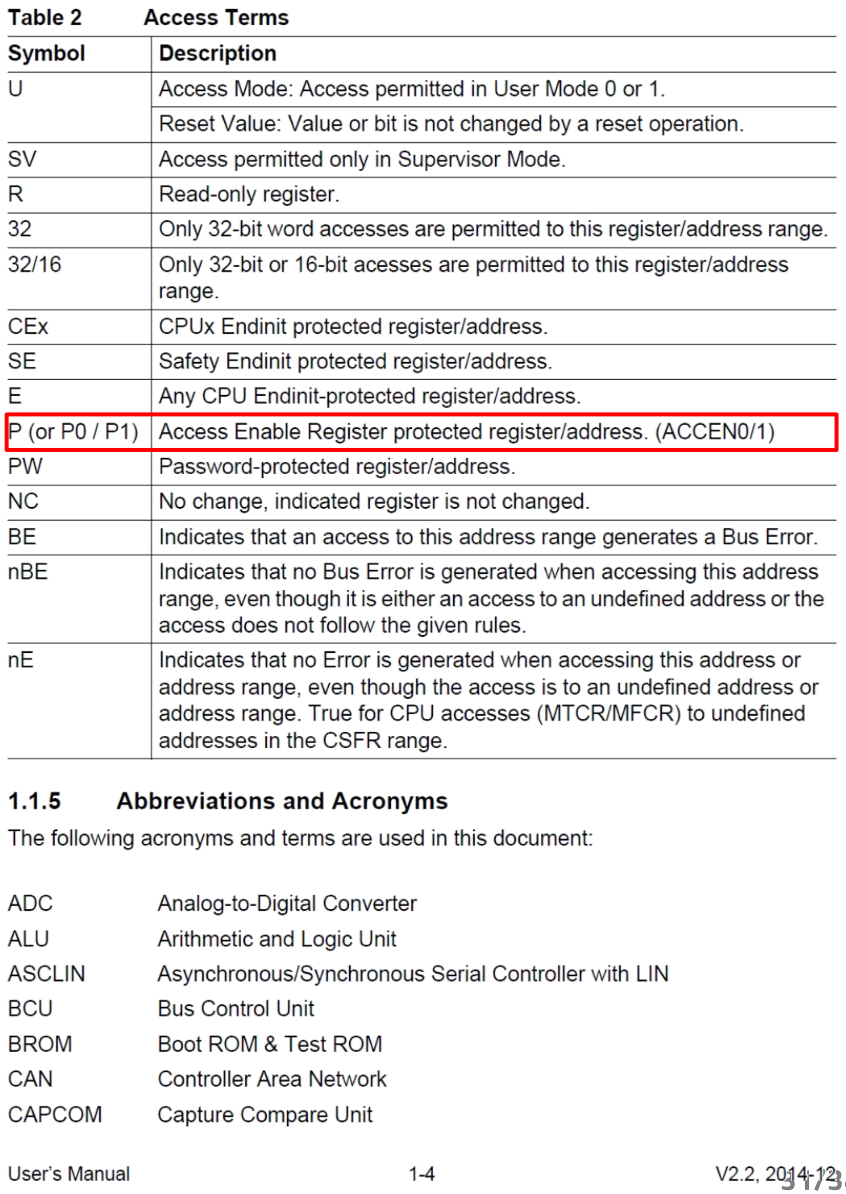


Table 2 Access Terms

Symbol	Description
U	Access Mode: Access permitted in User Mode 0 or 1.
	Reset Value: Value or bit is not changed by a reset operation.
SV	Access permitted only in Supervisor Mode.
R	Read-only register.
32	Only 32-bit word accesses are permitted to this register/address range.
32/16	Only 32-bit or 16-bit accesses are permitted to this register/address range.
CEx	CPUx Endinit protected register/address.
SE	Safety Endinit protected register/address.
E	Any CPU Endinit-protected register/address.
P (or P0 / P1)	Access Enable Register protected register/address. (ACCEN0/1)
PW	Password-protected register/address.
NC	No change, indicated register is not changed.
BE	Indicates that an access to this address range generates a Bus Error.
nBE	Indicates that no Bus Error is generated when accessing this address range, even though it is either an access to an undefined address or the access does not follow the given rules.
nE	Indicates that no Error is generated when accessing this address or address range, even though the access is to an undefined address or address range. True for CPU accesses (MTCR/MFCR) to undefined addresses in the CSFR range.

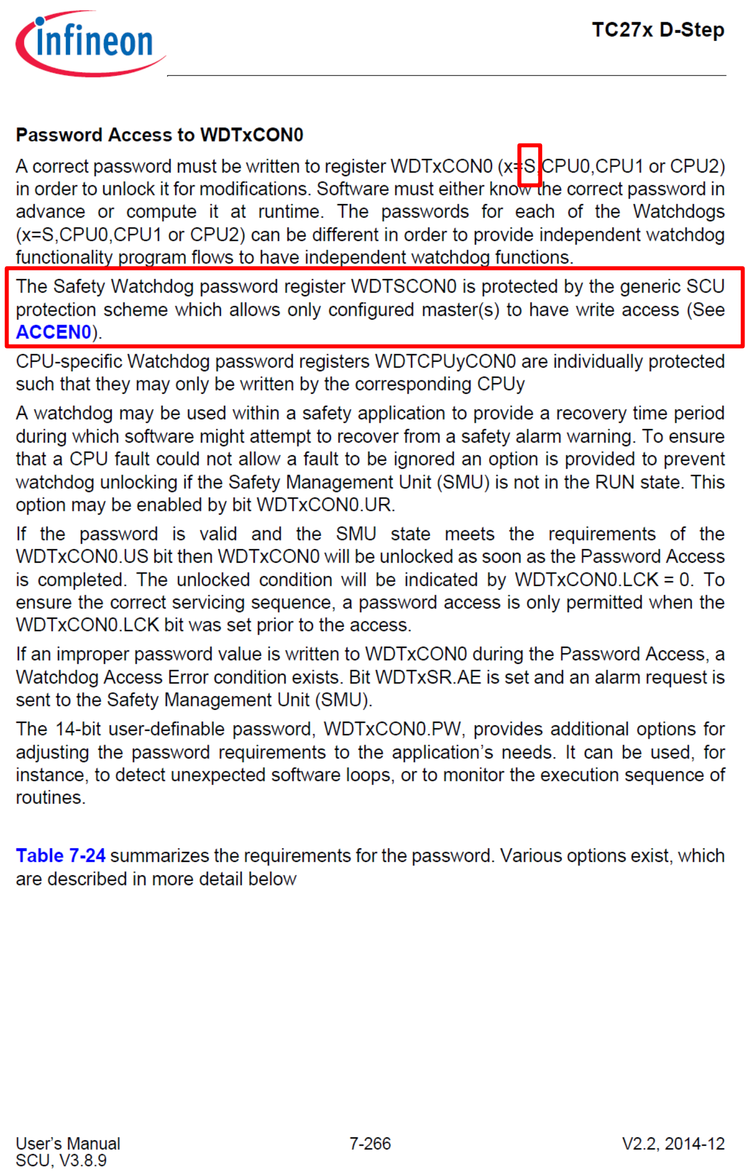
1.1.5 Abbreviations and Acronyms

The following acronyms and terms are used in this document:

ADC	Analog-to-Digital Converter
ALU	Arithmetic and Logic Unit
ASCLIN	Asynchronous/Synchronous Serial Controller with LIN
BCU	Bus Control Unit
BROM	Boot ROM & Test ROM
CAN	Controller Area Network
CAPCOM	Capture Compare Unit

User's Manual 1-4 V2.2, 2014-12

External Interrupt Example



Password Access to WDTxCON0

A correct password must be written to register WDTxCON0 (x=S,CPU0,CPU1 or CPU2) in order to unlock it for modifications. Software must either know the correct password in advance or compute it at runtime. The passwords for each of the Watchdogs (x=S,CPU0,CPU1 or CPU2) can be different in order to provide independent watchdog functionality program flows to have independent watchdog functions.

The Safety Watchdog password register WDTSCON0 is protected by the generic SCU protection scheme which allows only configured master(s) to have write access (See **ACCEN0**).

CPU-specific Watchdog password registers WDTCPUyCON0 are individually protected such that they may only be written by the corresponding CPUy

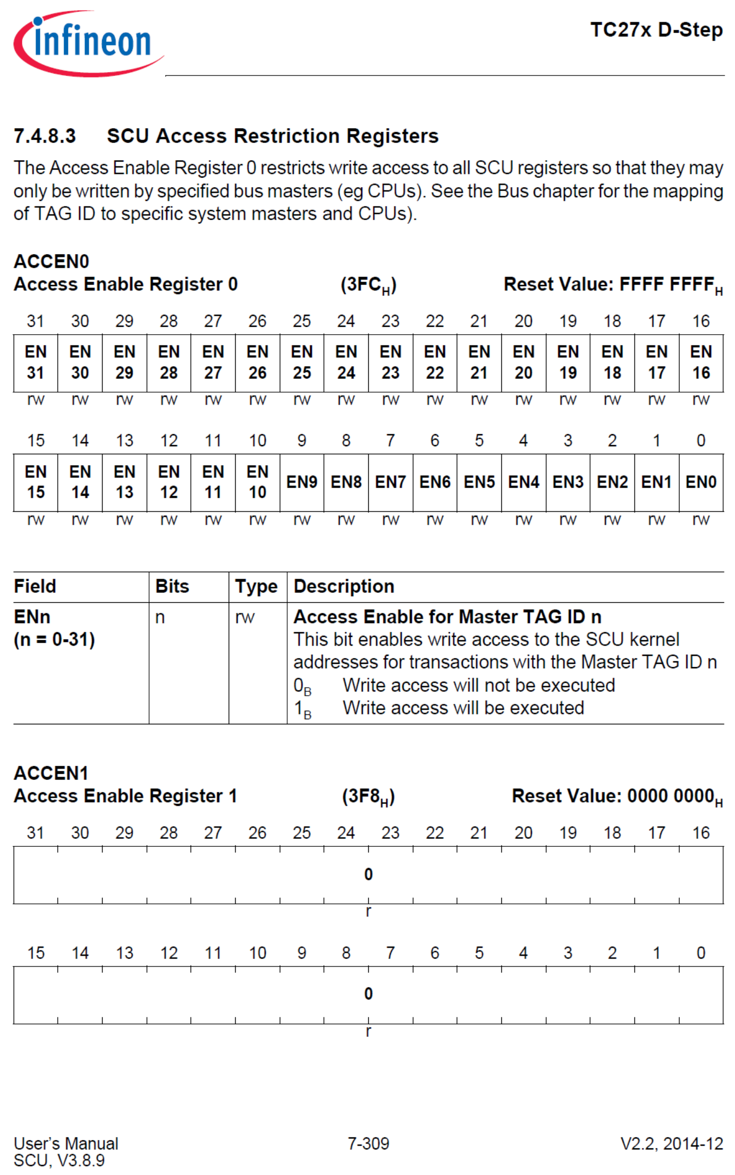
A watchdog may be used within a safety application to provide a recovery time period during which software might attempt to recover from a safety alarm warning. To ensure that a CPU fault could not allow a fault to be ignored an option is provided to prevent watchdog unlocking if the Safety Management Unit (SMU) is not in the RUN state. This option may be enabled by bit WDTxCON0.UR.

If the password is valid and the SMU state meets the requirements of the WDTxCON0.US bit then WDTxCON0 will be unlocked as soon as the Password Access is completed. The unlocked condition will be indicated by WDTxCON0.LCK = 0. To ensure the correct servicing sequence, a password access is only permitted when the WDTxCON0.LCK bit was set prior to the access.

If an improper password value is written to WDTxCON0 during the Password Access, a Watchdog Access Error condition exists. Bit WDTxSR.AE is set and an alarm request is sent to the Safety Management Unit (SMU).

The 14-bit user-definable password, WDTxCON0.PW, provides additional options for adjusting the password requirements to the application's needs. It can be used, for instance, to detect unexpected software loops, or to monitor the execution sequence of routines.

Table 7-24 summarizes the requirements for the password. Various options exist, which are described in more detail below



7.4.8.3 SCU Access Restriction Registers

The Access Enable Register 0 restricts write access to all SCU registers so that they may only be written by specified bus masters (eg CPUs). See the Bus chapter for the mapping of TAG ID to specific system masters and CPUs).

ACCEN0
Access Enable Register 0 **($3FC_H$)** **Reset Value: FFFF $FFFF_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EN31	EN30	EN29	EN28	EN27	EN26	EN25	EN24	EN23	EN22	EN21	EN20	EN19	EN18	EN17	EN16
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EN15	EN14	EN13	EN12	EN11	EN10	EN9	EN8	EN7	EN6	EN5	EN4	EN3	EN2	EN1	EN0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
ENn (n = 0-31)	n	rw	**Access Enable for Master TAG ID n** This bit enables write access to the SCU kernel addresses for transactions with the Master TAG ID n 0_B Write access will not be executed 1_B Write access will be executed

ACCEN1
Access Enable Register 1 **($3F8_H$)** **Reset Value: 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0															
r															

External Interrupt Example

2. Data sheet 분석 : Safety Critical Register 설정 (1)
   - ✓ 설정해야 하는 SCU_EICR1 / SCU_IGCR0 Register는 Safety Critical Register이기 때문에 Write Protected (Safety ENDINIT, End-of-Initialization) 되어 있다.
   - ✓ 해당 Register를 수정하기 위해서는 Safety ENDINIT을 해제해야 한다.
   - ✓ SCU_WDTSCON0 Register는 **Safety Critical Register**에 대한 **Safety ENDINIT을 설정/해제**한다.

SCU_WDTSCON0 Register 주소: F003_60F0h
(F0036000h + 0F0h)

SCU_WDTSCON0 Register 구조:

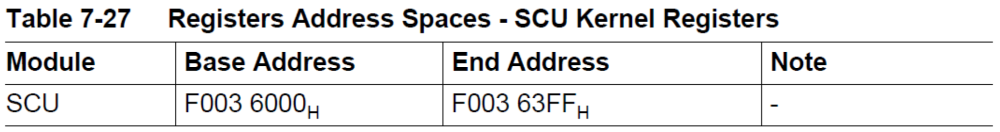
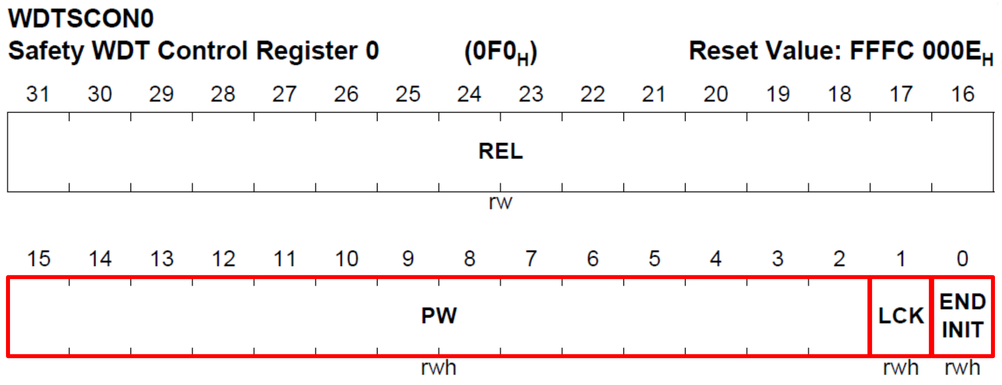


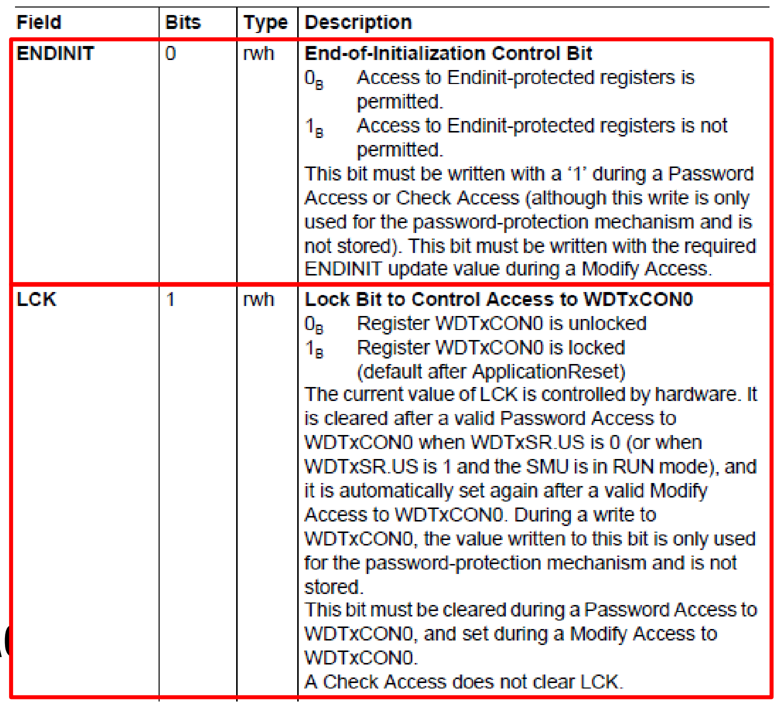
Table 7-27 Registers Address Spaces - SCU Kernel Registers

Module	Base Address	End Address	Note
SCU	F003 6000_H	F003 $63FF_H$	-

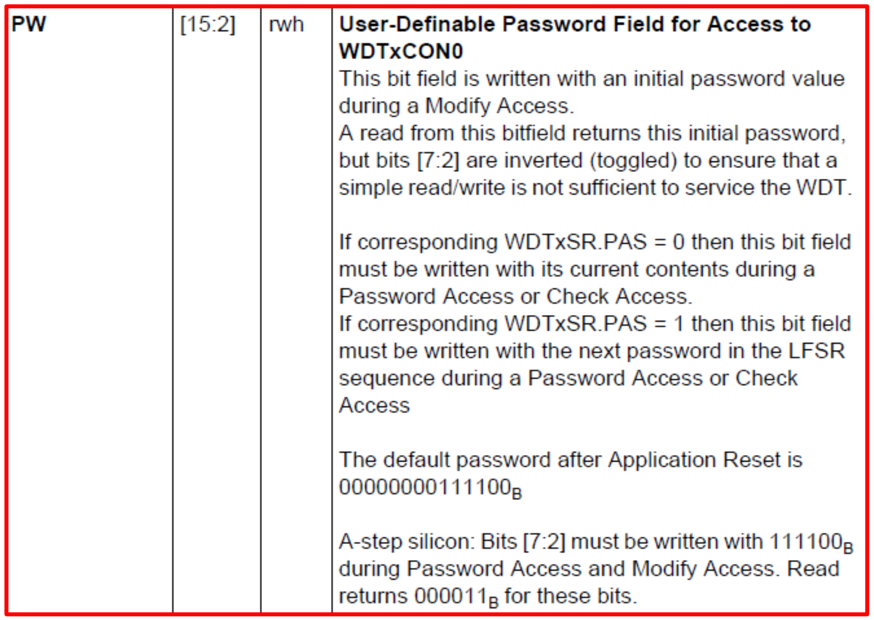


External Interrupt Example

2. Data sheet 분석 : Safety Critical Register 설정 (2)
   - ✓ **ENDINIT bit**는 Safety ENDINIT의 설정 상태를 나타내며 Modify Access를 통해서만 수정이 가능하다.
   - ✓ **LCK bit**는 SCU_WDTSCON0 Register의 Lock 상태를 나타내며 해당 Register의 Lock 상태는 Password Access를 통해 Unlock 되고, Modify Access를 통해 Lock 된다.
   - ✓ **PW bits**는 SCU_WDTSCON0 Register에 접근하기 위한 Password를 저장하며 해당 값을 읽으면



Field	Bits	Type	Description
ENDINIT	0	rwh	**End-of-Initialization Control Bit** 0_B Access to Endinit-protected registers is permitted. 1_B Access to Endinit-protected registers is not permitted. This bit must be written with a '1' during a Password Access or Check Access (although this write is only used for the password-protection mechanism and is not stored). This bit must be written with the required ENDINIT update value during a Modify Access.
LCK	1	rwh	**Lock Bit to Control Access to WDTxCON0** 0_B Register WDTxCON0 is unlocked 1_B Register WDTxCON0 is locked (default after ApplicationReset) The current value of LCK is controlled by hardware. It is cleared after a valid Password Access to WDTxCON0 when WDTxSR.US is 0 (or when WDTxSR.US is 1 and the SMU is in RUN mode), and it is automatically set again after a valid Modify Access to WDTxCON0. During a write to WDTxCON0, the value written to this bit is only used for the password-protection mechanism and is not stored. This bit must be cleared during a Password Access to WDTxCON0, and set during a Modify Access to WDTxCON0. A Check Access does not clear LCK.



PW	[15:2]	rwh	**User-Definable Password Field for Access to WDTxCON0** This bit field is written with an initial password value during a Modify Access. A read from this bitfield returns this initial password, but bits [7:2] are inverted (toggled) to ensure that a simple read/write is not sufficient to service the WDT. If corresponding WDTxSR.PAS = 0 then this bit field must be written with its current contents during a Password Access or Check Access. If corresponding WDTxSR.PAS = 1 then this bit field must be written with the next password in the LFSR sequence during a Password Access or Check Access The default password after Application Reset is 00000000111100_B A-step silicon: Bits [7:2] must be written with 111100_B during Password Access and Modify Access. Read returns 000011_B for these bits.

External Interrupt Example

2. Data sheet 분석 : Safety Critical Register 설정 (3)

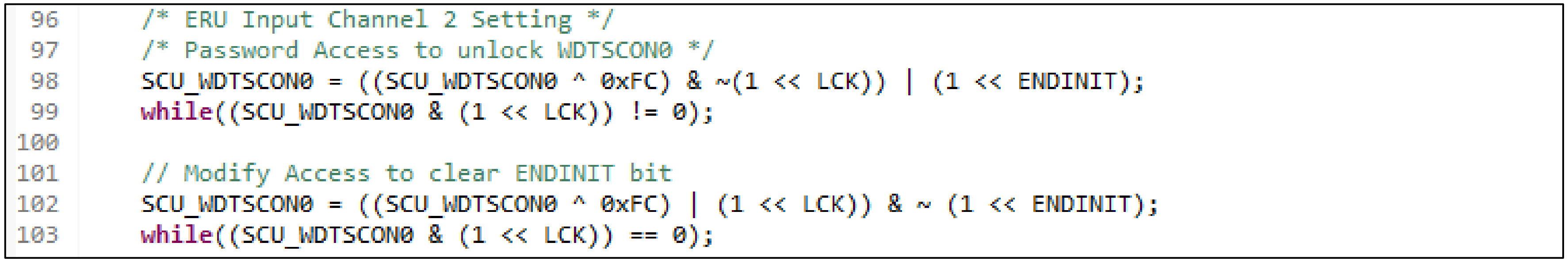
- ✓ SCU_WDTSCON0 Register에 적절한 값을 Write하여 **Password Access**를 수행한다.
- ✓ **Password Access는 SCU_WDTSCON0 Register의 Lock 상태를 해제**하며 과정은 다음과 같다.
  1. SCU_WDTSCON0 Register의 값을 읽어 REL bits, PW bits를 파악한다.
  2. Bits[7:2] (PW bits의 일부)가 반전되어 읽히기 때문에 이를 반전시켜 정확한 PW bits를 얻는다.
  3. Write 할 값의 bits[31:16]은 읽혀진 REL bits 값으로 설정하고 bit[15:2]는 앞서 구한 정확한 PW bits 값으로 설정한다.
  4. Write 할 값의 bit[1]은 unlock(write 0) 후, bit[0]은 0으로 설정한다.
     (Safety ENDINIT 설정: bit[0] = 1, Safety ENDINIT 해제 : bit[0] = 0)
  5. 설정된 값을 SCU_WDTSCON0 Register에 한번에 쓴다.
  6. SCU_WDTSCON0 Register의 LCK bit를 확인하여 Lock 상태가 해제되었는지 파악한다.
     (Password Access가 정상적으로 수행되면 Lock 상태가 해제되며 LCK bit가 0으로 설정된다.)
- ✓ Password Access를 통해 SCU_WDTSCON0 Register의 Lock 상태가 해제되면 Modify Access를 통해 Safety ENDINIT을 설정/해제할 수 있다.

External Interrupt Example

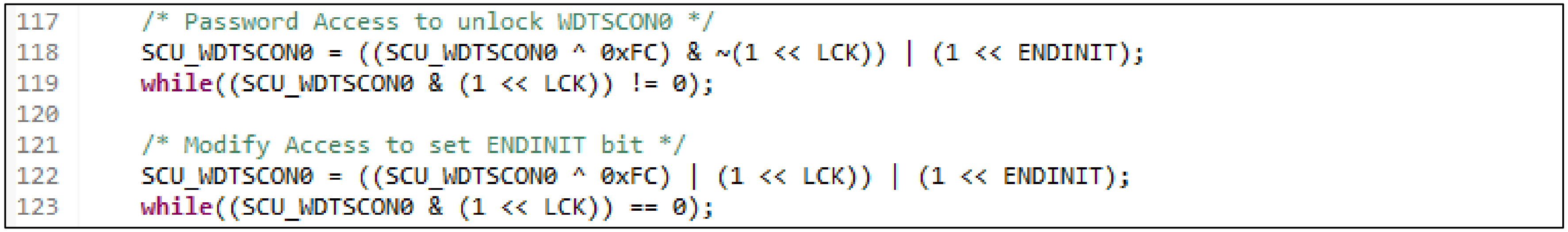
2. Data sheet 분석 : Safety Critical Register 설정 (4)

- ✓ SCU_WDTSCON0 Register에 적절한 값을 Write하여 **Modify Access**를 수행한다.
- ✓ **Modify Access**는 **Safety ENDINIT을 설정/해제**하며 과정은 다음과 같다.
    1. SCU_WDTSCON0 Register의 값을 읽어 REL bits, PW bits를 파악한다.
    2. Bits[7:2] (PW bits의 일부)가 반전되어 읽히기 때문에 이를 반전시켜 정확한 PW bits를 얻는다.
    3. Write 할 값의 bits[31:16]은 읽혀진 REL bits 값으로 설정하고 bit[15:2]는 앞서 구한 정확한 PW bits 값으로 설정한다.
    4. Write 할 값의 bit[1]은 unlock(write 0) 후, bit[0]은 1로 설정한다.
       (Safety ENDINIT 설정: bit[0] = 1, Safety ENDINIT 해제 : bit[0] = 0)
    5. 설정된 값을 SCU_WDTSCON0 Register에 한번에 쓴다.
    6. SCU_WDTSCON0 Register의 LCK bit를 확인하여 Lock 상태가 다시 설정되었는지 파악한다.
       (Modify Access가 정상적으로 수행되면 Lock 상태가 설정되며 LCK bit가 1로 설정된다.)
- ✓ Modify Access를 통해 Safety ENDINIT을 해제하면 Safety Critical Register를 수정할 수 있으며 수정을 완료하면 Safety ENDINIT을 꼭 다시 설정해야 한다.

External Interrupt Example



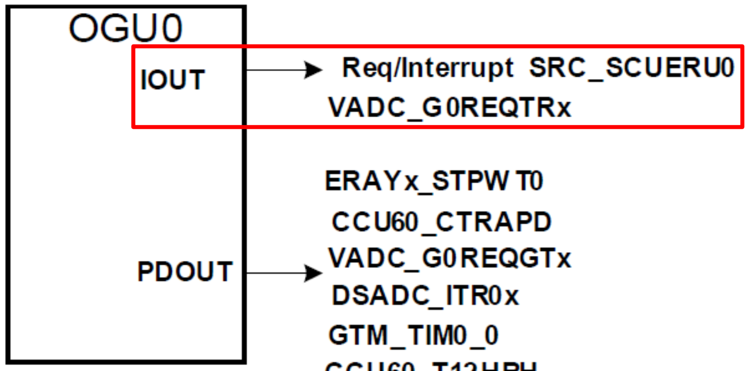
```
    /* ERU Input Channel 2 Setting */
    /* Password Access to unlock WDTSCON0 */
    SCU_WDTSCON0 = ((SCU_WDTSCON0 ^ 0xFC) & ~(1 << LCK)) | (1 << ENDINIT);
    while((SCU_WDTSCON0 & (1 << LCK)) != 0);

    // Modify Access to clear ENDINIT bit
    SCU_WDTSCON0 = ((SCU_WDTSCON0 ^ 0xFC) | (1 << LCK)) & ~ (1 << ENDINIT);
    while((SCU_WDTSCON0 & (1 << LCK)) == 0);
```



```
    /* Password Access to unlock WDTSCON0 */
    SCU_WDTSCON0 = ((SCU_WDTSCON0 ^ 0xFC) & ~(1 << LCK)) | (1 << ENDINIT);
    while((SCU_WDTSCON0 & (1 << LCK)) != 0);

    /* Modify Access to set ENDINIT bit */
    SCU_WDTSCON0 = ((SCU_WDTSCON0 ^ 0xFC) | (1 << LCK)) | (1 << ENDINIT);
    while((SCU_WDTSCON0 & (1 << LCK)) == 0);
```

External Interrupt Example

2. Data sheet 분석 : Interrupt Router 설정 (1)
   - ✓ Interrupt Router는 Interrupt Trigger를 Service Providers (CPU 0-2, DMA)에 연결한다.
   - ✓ Switch 신호에 따라 ERU에서 생성된 트리거 신호는 **Output Channel 0의 IOUT**으로 출력된다.
   - ✓ 해당 출력은 Interrupt Router의 **SCUERU0 SRN (Service Request Node)**와 연결된다.
   - ✓ 해당 노드에 대한 설정을 하기 위해 **SRC_SCUERU0 Register**를 설정해야 한다.



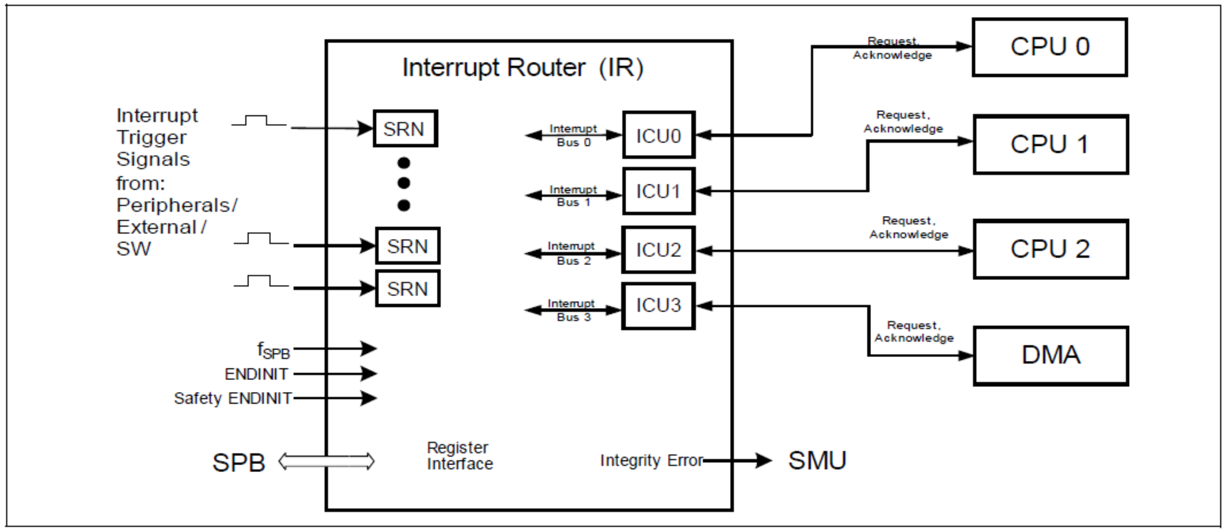


Figure 16-1 Block Diagram of the TC27x Interrupt System

External Interrupt Example

2. Data sheet 분석 : Interrupt Router 설정 (2)
   - ✓ SRC_SCUERU0 Register는 SCUERU0 SRN에 대한 Interrupt 설정을 한다.
   - ✓ 해당 Interrupt의 우선순위를 설정하기 위해 **SRPN bits**를 **Ah** (임의의 값)로 설정한다.
     (우선순위는 해당 Interrupt가 할당된 Service Provider에서 Interrupt Vector Table의 Index가 된다.)
   - ✓ 해당 Interrupt가 CPU0에서 처리되도록 하기 위해 **TOS bits**를 **0h**로 설정한다.
   - ✓ 해당 Interrupt를 Enable 하기 위해 **SRE bit**를 **1**로 설정한다.

SRC_SCUERU0 Register 주소: F003_8CD4h
(F0038000h + CD4h)

SRC_SCUERU0 Register 구조:

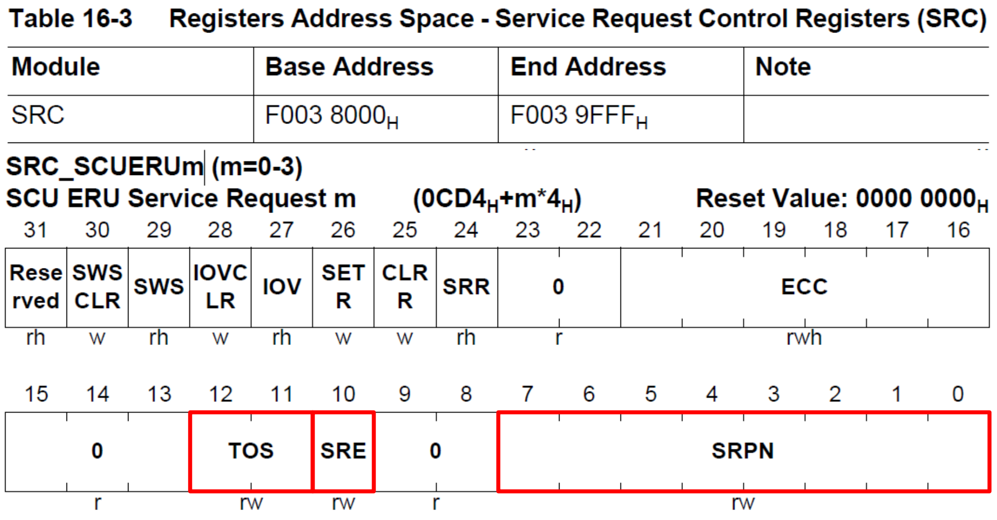


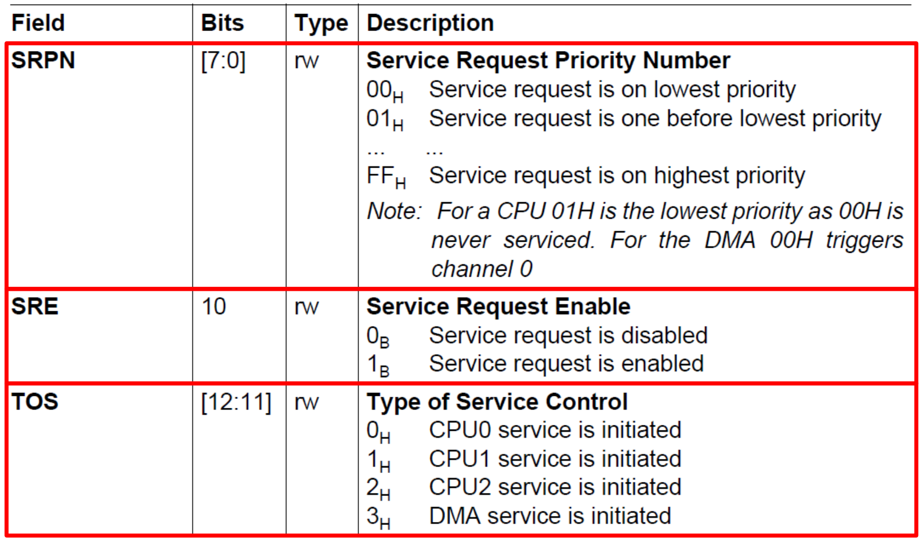
Table 16-3 Registers Address Space - Service Request Control Registers (SRC)

Module	Base Address	End Address	Note
SRC	F003 8000$_H$	F003 9FFF$_H$	

SRC_SCUERUm (m=0-3)
SCU ERU Service Request m (0CD4$_H$+m*4$_H$) Reset Value: 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved	SWSCLR	SWS	IOVCLR	IOV	SETR	CLRR	SRR	0		ECC					
rh	w	rh	w	rh	w	w	rh	r		rwh					

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0			TOS		SRE	0		SRPN							
r			rw		rw	r		rw							



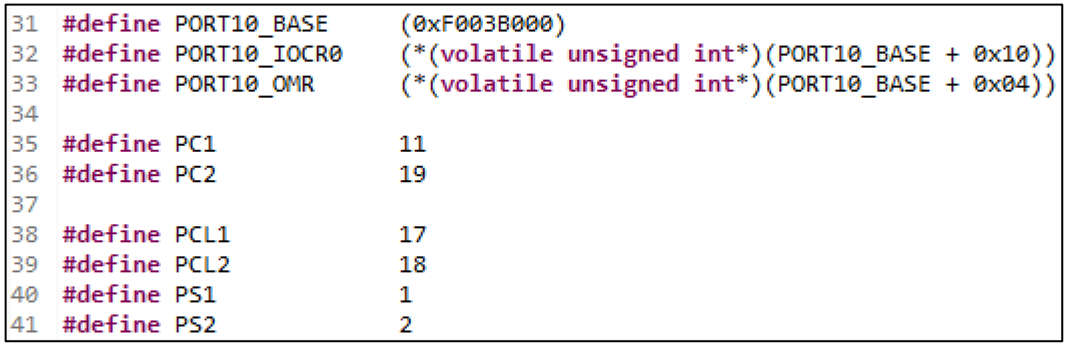
Field	Bits	Type	Description
SRPN	[7:0]	rw	**Service Request Priority Number** 00$_H$ Service request is on lowest priority 01$_H$ Service request is one before lowest priority ... ... FF$_H$ Service request is on highest priority *Note: For a CPU 01H is the lowest priority as 00H is never serviced. For the DMA 00H triggers channel 0*
SRE	10	rw	**Service Request Enable** 0$_B$ Service request is disabled 1$_B$ Service request is enabled
TOS	[12:11]	rw	**Type of Service Control** 0$_H$ CPU0 service is initiated 1$_H$ CPU1 service is initiated 2$_H$ CPU2 service is initiated 3$_H$ DMA service is initiated

External Interrupt Example

3. 프로그래밍

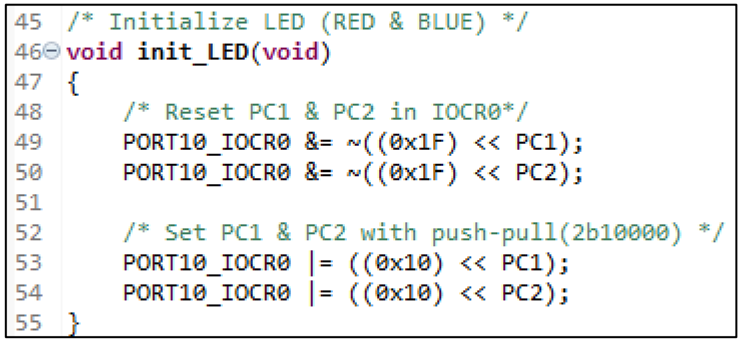
   1) Switch 및 LED가 연결된 PORT에 대한 설정을 수행하는 함수를 구현한다.

      ✓ 자세한 내용은 이전 강의자료 (GPIO)를 참고한다.



```c
#define PORT10_BASE     (0xF003B000)
#define PORT10_IOCR0    (*(volatile unsigned int*)(PORT10_BASE + 0x10))
#define PORT10_OMR      (*(volatile unsigned int*)(PORT10_BASE + 0x04))

#define PC1             11
#define PC2             19

#define PCL1            17
#define PCL2            18
#define PS1             1
#define PS2             2
```

PORT13 IO (LED RED) 설정관련 레지스터 주소 및 비트 필드 정의



```c
/* Initialize LED (RED & BLUE) */
void init_LED(void)
{
    /* Reset PC1 & PC2 in IOCR0*/
    PORT10_IOCR0 &= ~((0x1F) << PC1);
    PORT10_IOCR0 &= ~((0x1F) << PC2);

    /* Set PC1 & PC2 with push-pull(2b10000) */
    PORT10_IOCR0 |= ((0x10) << PC1);
    PORT10_IOCR0 |= ((0x10) << PC2);
}
```

PORT13 IO (LED RED) 설정 초기화 코드

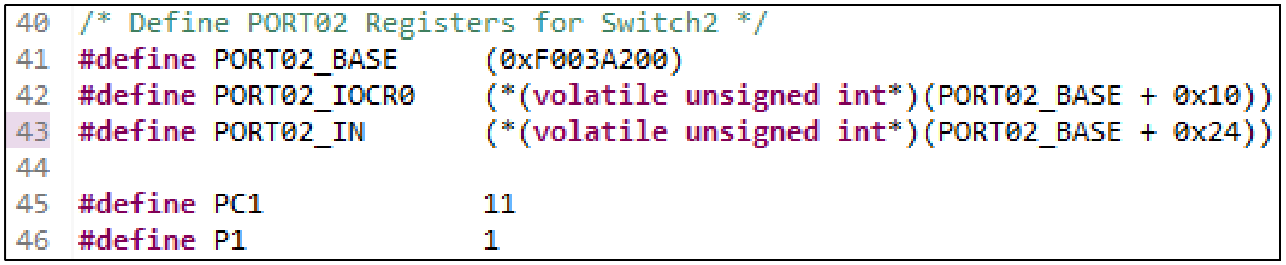
ACE Lab.

External Interrupt Example

3. 프로그래밍

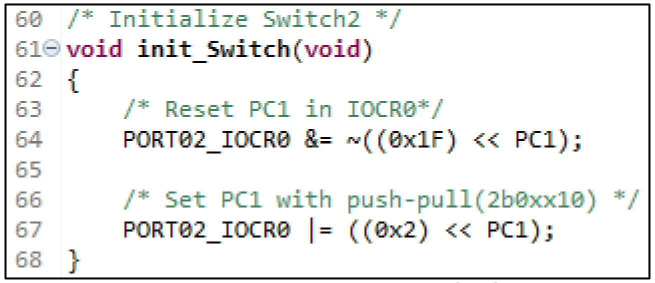
1) Switch 및 LED가 연결된 PORT에 대한 설정을 수행하는 함수를 구현한다.

✓ 자세한 내용은 이전 강의자료 (GPIO)를 참고한다.



```c
/* Define PORT02 Registers for Switch2 */
#define PORT02_BASE     (0xF003A200)
#define PORT02_IOCR0    (*(volatile unsigned int*)(PORT02_BASE + 0x10))
#define PORT02_IN       (*(volatile unsigned int*)(PORT02_BASE + 0x24))

#define PC1             11
#define P1              1
```

PORT02 IO (Switch2) 설정관련 레지스터 주소 및 비트 필드 정의



```c
/* Initialize Switch2 */
void init_Switch(void)
{
    /* Reset PC1 in IOCR0*/
    PORT02_IOCR0 &= ~((0x1F) << PC1);

    /* Set PC1 with push-pull(2b0xx10) */
    PORT02_IOCR0 |= ((0x2) << PC1);
}
```

PORT02 IO (Switch2) 설정 코드

ACE Lab.

External Interrupt Example

3. 프로그래밍

2) ERU를 설정하기 위한 함수를 구현한다.

① SCU_WDTSCON0 Register를 통해 Password/Modify Access를 수행하여 Safety ENDINIT을 해제한다.

② SCU_EICR1 Register를 통해 ERU의 Channel 2의 입력으로 Input 1을 설정한다.

③ SCU_EICR1 Register를 통해 Falling edge가 트리거 신호를 생성하도록 설정하고 이를 Enable 한다.

④ SCU_EICR1 Register를 통해 생성된 트리거 신호가 Output Channel 0에 전달되도록 한다.

⑤ SCU_IGCR0 Register를 통해 전달된 트리거 신호가 IOUT으로 출력되도록 설정한다.

⑥ SCU_WDTSCON0 Register를 통해 Password/Modify Access를 수행하여 Safety ENDINIT을 설정한다.

⑦ SRC_SCUERU0 Register를 통해 SCUERU0 SRN의 우선순위를 설정한다.

⑧ SRC_SCUERU0 Register를 통해 SCUERU0 SRN의 처리가 CPU0에서 수행되도록 설정한다.

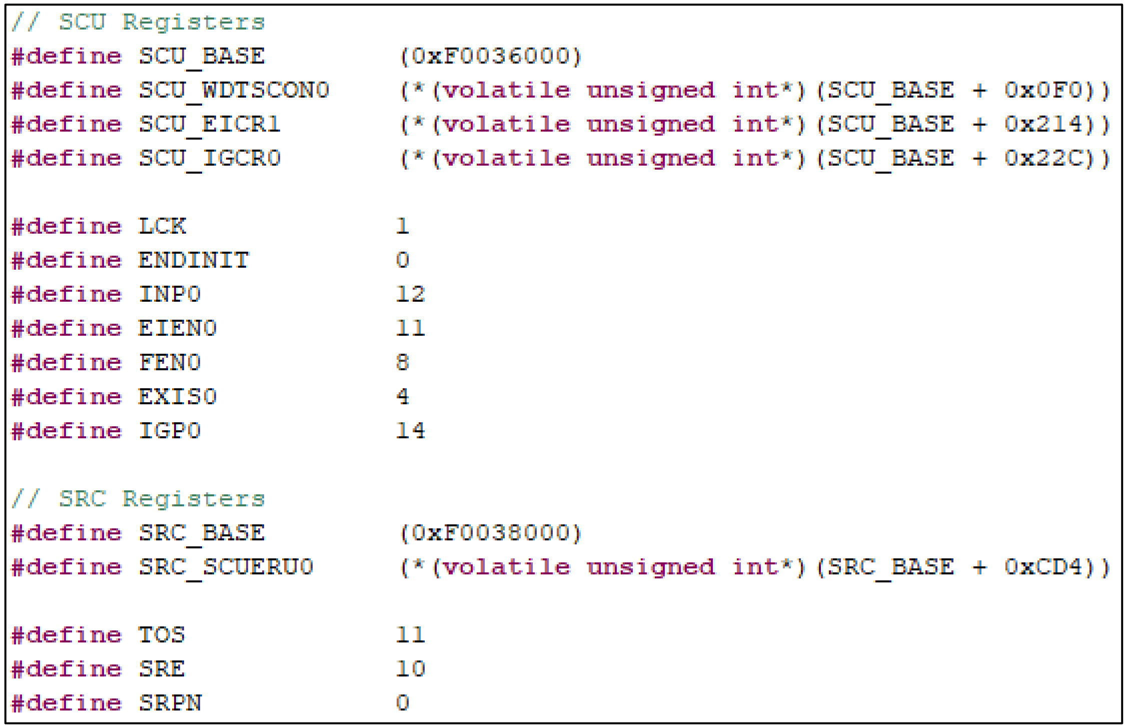
⑨ SRC_SCUERU0 Register를 통해 SCUERU0 SRN의 Interrupt를 Enable 한다.

ACE Lab.

External Interrupt Example

3. 프로그래밍

   2) ERU를 설정하기 위한 함수를 구현한다.



```
// SCU Registers
#define SCU_BASE         (0xF0036000)
#define SCU_WDTSCON0     (*(volatile unsigned int*)(SCU_BASE + 0x0F0))
#define SCU_EICR1        (*(volatile unsigned int*)(SCU_BASE + 0x214))
#define SCU_IGCR0        (*(volatile unsigned int*)(SCU_BASE + 0x22C))

#define LCK             1
#define ENDINIT         0
#define INP0            12
#define EIEN0           11
#define FEN0            8
#define EXIS0           4
#define IGP0            14

// SRC Registers
#define SRC_BASE         (0xF0038000)
#define SRC_SCUERU0      (*(volatile unsigned int*)(SRC_BASE + 0xCD4))

#define TOS             11
#define SRE             10
#define SRPN            0
```

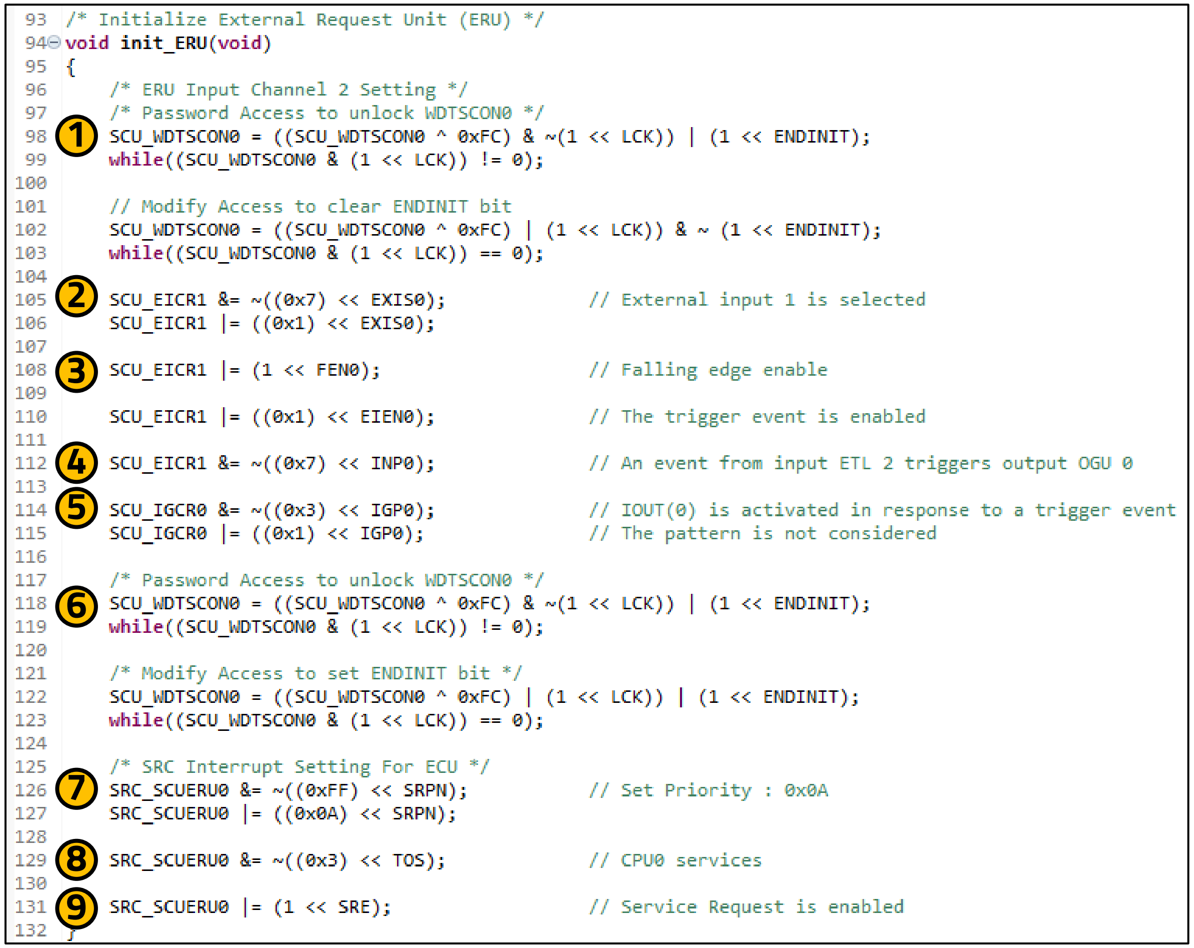
ERU 설정관련 레지스터 주소 및 비트 필드 정의

ACE Lab.

External Interrupt Example

3. 프로그래밍

   2) ERU를 설정하기 위한 함수를 구현한다.



```c
/* Initialize External Request Unit (ERU) */
void init_ERU(void)
{
    /* ERU Input Channel 2 Setting */
    /* Password Access to unlock WDTSCON0 */
①   SCU_WDTSCON0 = ((SCU_WDTSCON0 ^ 0xFC) & ~(1 << LCK)) | (1 << ENDINIT);
    while((SCU_WDTSCON0 & (1 << LCK)) != 0);

    // Modify Access to clear ENDINIT bit
    SCU_WDTSCON0 = ((SCU_WDTSCON0 ^ 0xFC) | (1 << LCK)) & ~ (1 << ENDINIT);
    while((SCU_WDTSCON0 & (1 << LCK)) == 0);

②   SCU_EICR1 &= ~((0x7) << EXIS0);                // External input 1 is selected
    SCU_EICR1 |= ((0x1) << EXIS0);

③   SCU_EICR1 |= (1 << FEN0);                      // Falling edge enable

    SCU_EICR1 |= ((0x1) << EIEN0);                 // The trigger event is enabled

④   SCU_EICR1 &= ~((0x7) << INP0);                 // An event from input ETL 2 triggers output OGU 0

⑤   SCU_IGCR0 &= ~((0x3) << IGP0);                 // IOUT(0) is activated in response to a trigger event
    SCU_IGCR0 |= ((0x1) << IGP0);                  // The pattern is not considered

    /* Password Access to unlock WDTSCON0 */
⑥   SCU_WDTSCON0 = ((SCU_WDTSCON0 ^ 0xFC) & ~(1 << LCK)) | (1 << ENDINIT);
    while((SCU_WDTSCON0 & (1 << LCK)) != 0);

    /* Modify Access to set ENDINIT bit */
    SCU_WDTSCON0 = ((SCU_WDTSCON0 ^ 0xFC) | (1 << LCK)) | (1 << ENDINIT);
    while((SCU_WDTSCON0 & (1 << LCK)) == 0);

    /* SRC Interrupt Setting For ECU */
⑦   SRC_SCUERU0 &= ~((0xFF) << SRPN);              // Set Priority : 0x0A
    SRC_SCUERU0 |= ((0x0A) << SRPN);

⑧   SRC_SCUERU0 &= ~((0x3) << TOS);                // CPU0 services

⑨   SRC_SCUERU0 |= (1 << SRE);                     // Service Request is enabled
}
```

ERU 설정 함수

External Interrupt Example

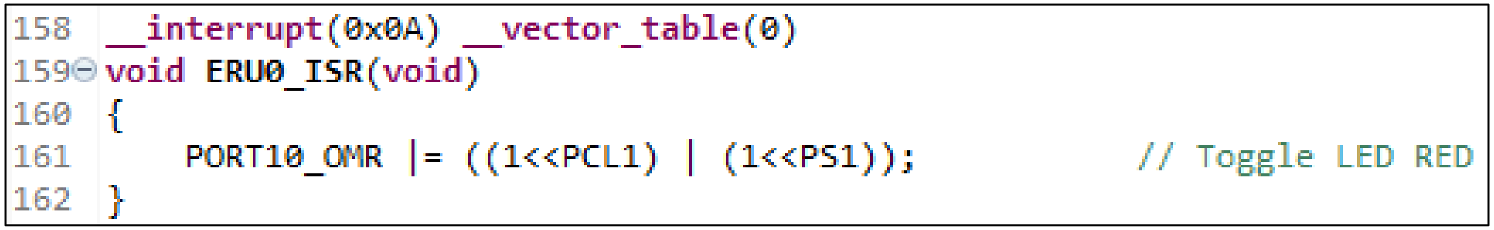
3. 프로그래밍

   3) ERU를 통한 External Interrupt에 대한 ISR를 구현한다.

      ✓ 해당 함수가 ISR 임을 나타내기 위해 컴파일러 지시자를 앞에 붙인다.

      - __interrupt(PRIORITY) : 괄호 안에는 ISR에 대응되는 Interrupt의 우선순위를 입력한다.
      - __vector_table(CPU_NUM) : 괄호 안에는 해당 ISR을 수행하는 CPU 번호를 입력한다.

      ✓ ISR이 수행된 후, 해당 Interrupt Flag가 자동으로 Clear 되기 때문에 이를 위한 코드가 필요하지 않다.



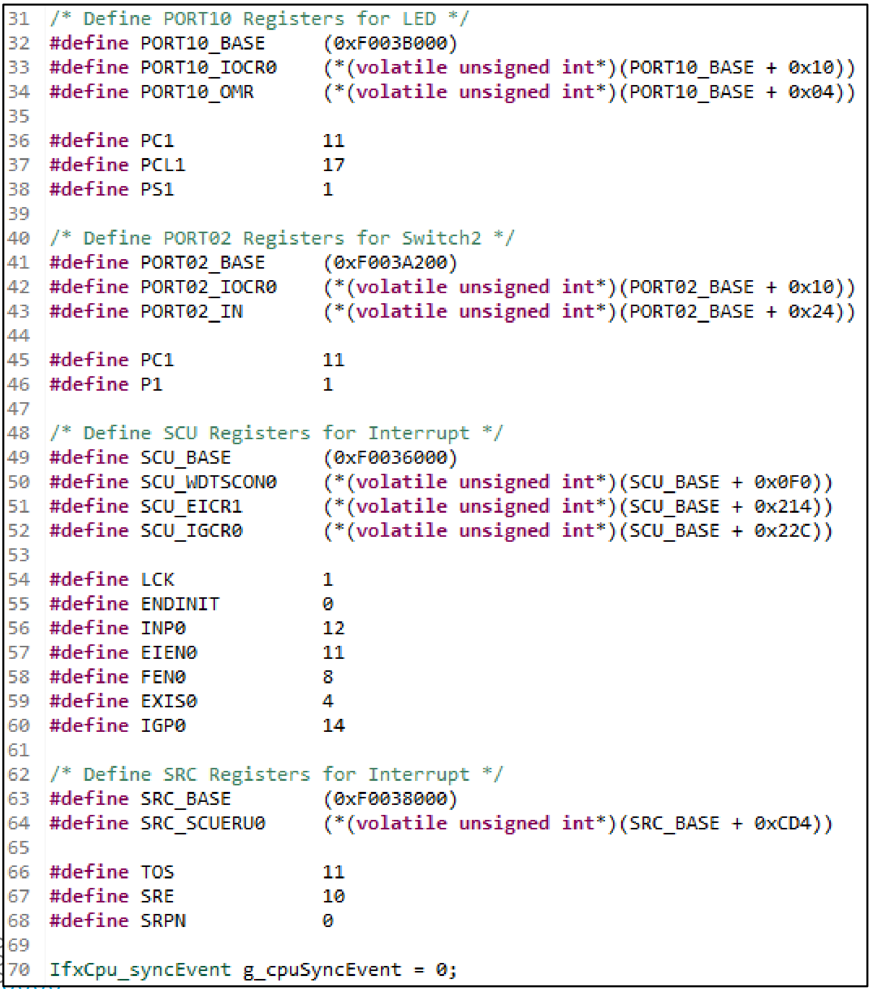
```
__interrupt(0x0A) __vector_table(0)
void ERU0_ISR(void)
{
    PORT10_OMR |= ((1<<PCL1) | (1<<PS1));          // Toggle LED RED
}
```

ERU0 Interrupt Service Routine

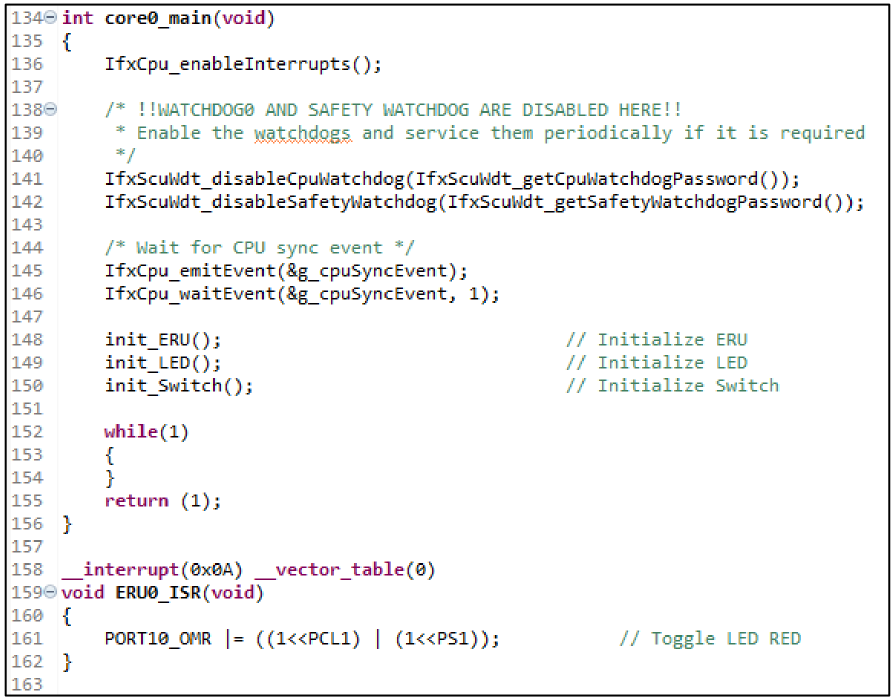
External Interrupt Example

3. 프로그래밍

   4) 동작에 따라 'main' 함수를 구현한다.



```c
/* Define PORT10 Registers for LED */
#define PORT10_BASE     (0xF003B000)
#define PORT10_IOCR0    (*(volatile unsigned int*)(PORT10_BASE + 0x10))
#define PORT10_OMR      (*(volatile unsigned int*)(PORT10_BASE + 0x04))

#define PC1             11
#define PCL1            17
#define PS1             1

/* Define PORT02 Registers for Switch2 */
#define PORT02_BASE     (0xF003A200)
#define PORT02_IOCR0    (*(volatile unsigned int*)(PORT02_BASE + 0x10))
#define PORT02_IN       (*(volatile unsigned int*)(PORT02_BASE + 0x24))

#define PC1             11
#define P1              1

/* Define SCU Registers for Interrupt */
#define SCU_BASE        (0xF0036000)
#define SCU_WDTSCON0    (*(volatile unsigned int*)(SCU_BASE + 0x0F0))
#define SCU_EICR1       (*(volatile unsigned int*)(SCU_BASE + 0x214))
#define SCU_IGCR0       (*(volatile unsigned int*)(SCU_BASE + 0x22C))

#define LCK             1
#define ENDINIT         0
#define INP0            12
#define EIEN0           11
#define FEN0            8
#define EXIS0           4
#define IGP0            14

/* Define SRC Registers for Interrupt */
#define SRC_BASE        (0xF0038000)
#define SRC_SCUERU0     (*(volatile unsigned int*)(SRC_BASE + 0xCD4))

#define TOS             11
#define SRE             10
#define SRPN            0

IfxCpu_syncEvent g_cpuSyncEvent = 0;
```



```c
int core0_main(void)
{
    IfxCpu_enableInterrupts();

    /* !!WATCHDOG0 AND SAFETY WATCHDOG ARE DISABLED HERE!!
     * Enable the watchdogs and service them periodically if it is required
     */
    IfxScuWdt_disableCpuWatchdog(IfxScuWdt_getCpuWatchdogPassword());
    IfxScuWdt_disableSafetyWatchdog(IfxScuWdt_getSafetyWatchdogPassword());

    /* Wait for CPU sync event */
    IfxCpu_emitEvent(&g_cpuSyncEvent);
    IfxCpu_waitEvent(&g_cpuSyncEvent, 1);

    init_ERU();                                 // Initialize ERU
    init_LED();                                 // Initialize LED
    init_Switch();                              // Initialize Switch

    while(1)
    {
    }
    return (1);
}

__interrupt(0x0A) __vector_table(0)
void ERU0_ISR(void)
{
    PORT10_OMR |= ((1<<PCL1) | (1<<PS1));           // Toggle LED RED
}
```

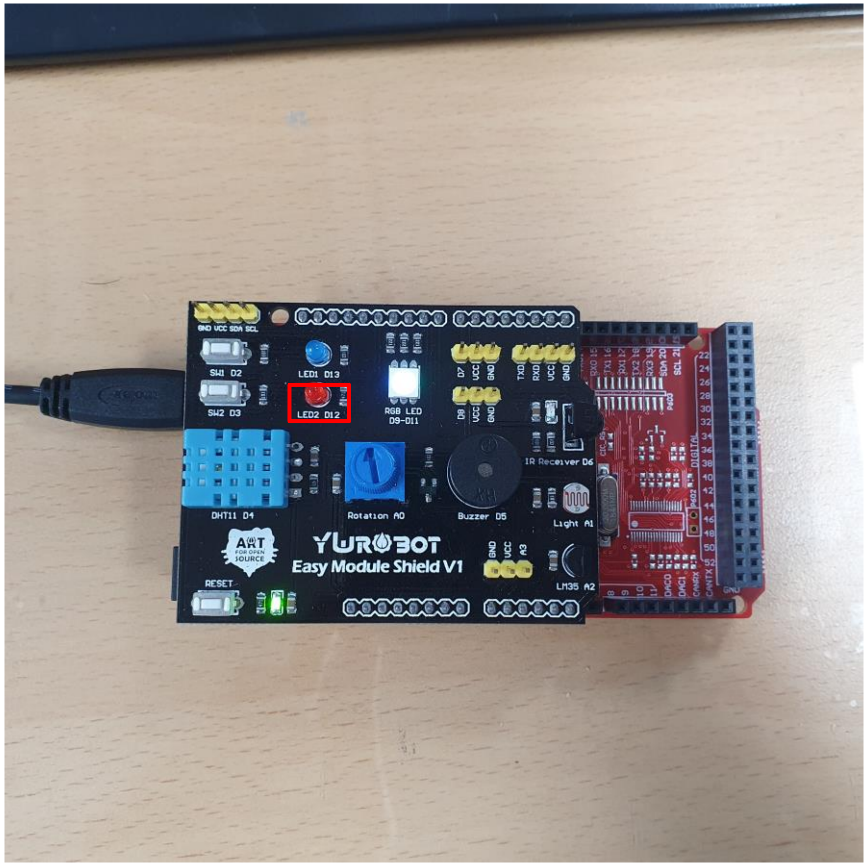
✓ '__enable()'을 통해 CPU의 Global Interrupt Enable을 수행한다.

✓ 앞서 구현한 함수들을 호출한다.

External Interrupt Example

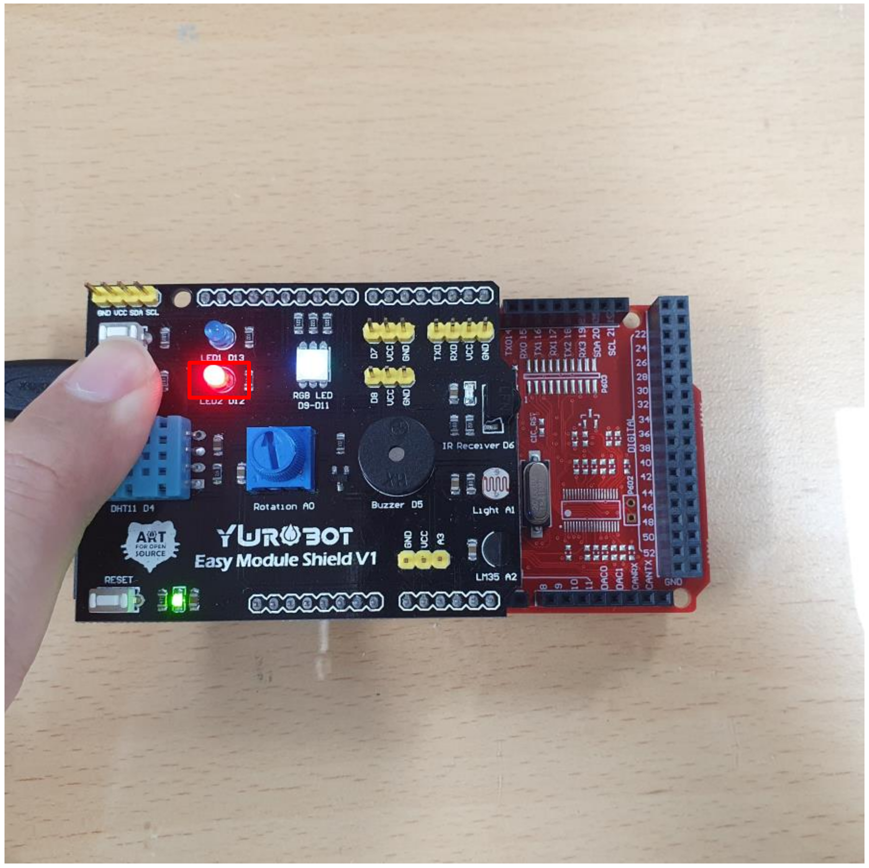
4. 동작 확인

✓ Build 및 Debug 후 ('Resume' 버튼 클릭), Switch를 누를 때마다 LED가 Toggle 되는 것을 확인한다.



Switch Click!

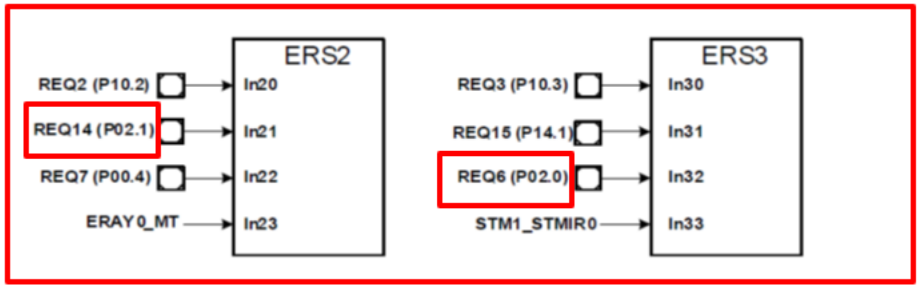
$\rightleftarrows$



ACE Lab.

External Interrupt Example

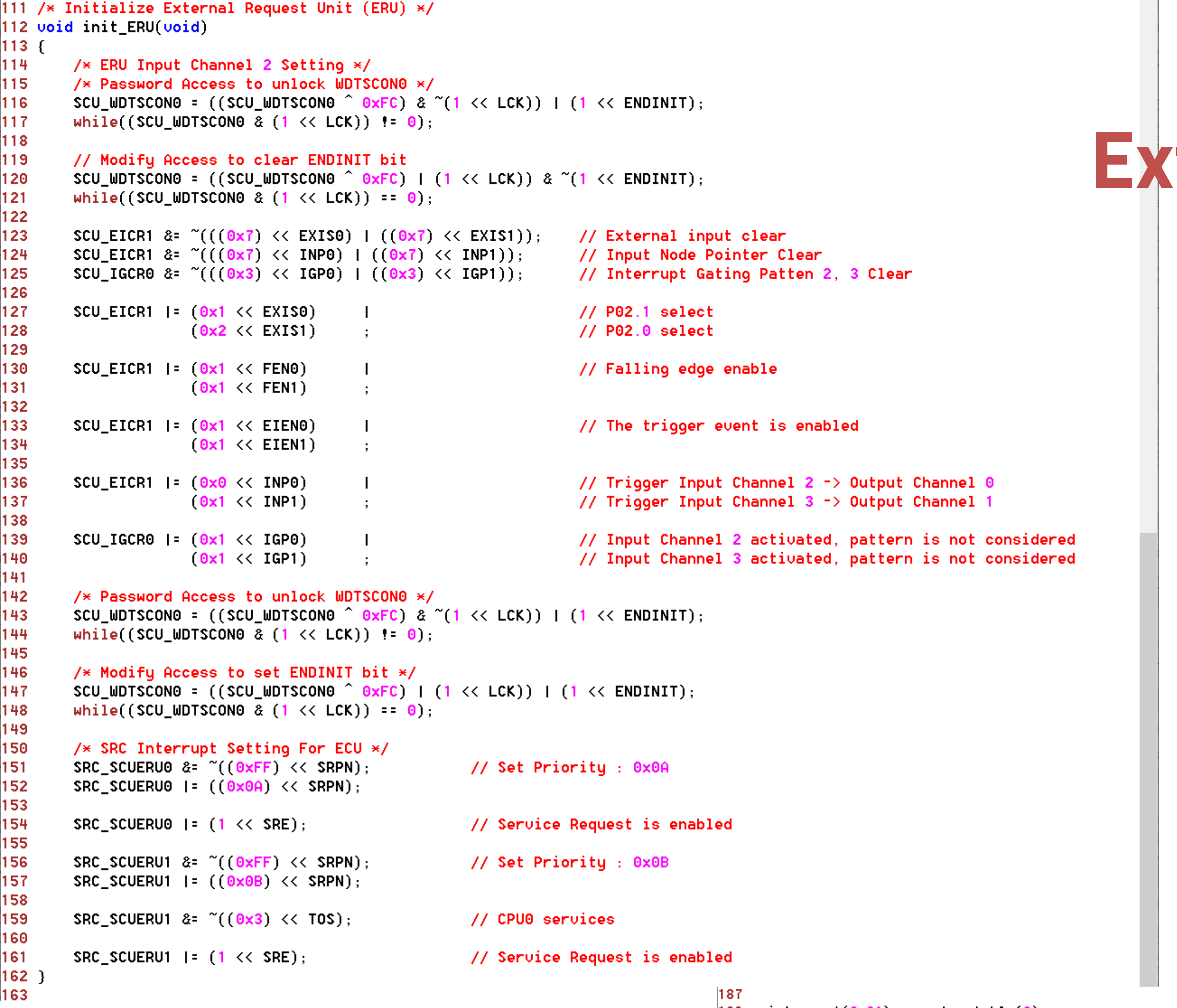
SW1 & SW2 External Interrupt



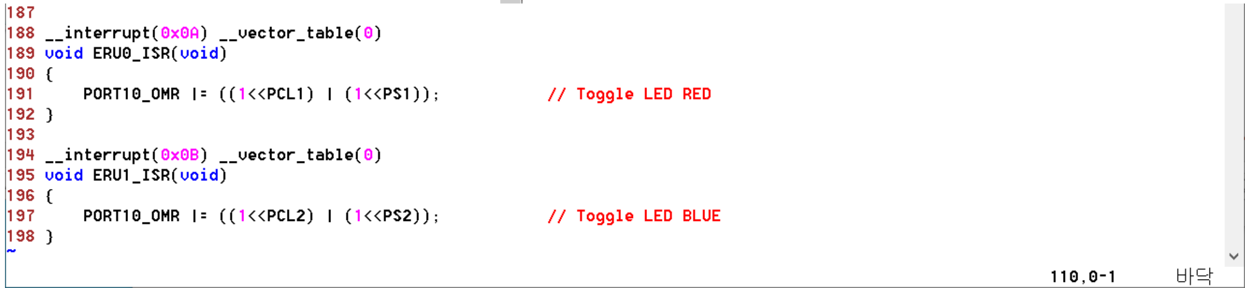
ECU_EICR1	ESR3 Input Channel 3 Select as P02.0, Then Interrupt Setting
	ESR3 → Output Channel 1
SRC_SCUERU1	CPU0.0x0B Interrupt

Interrupt Service Routine

SW1 & SW2 External Interrupt



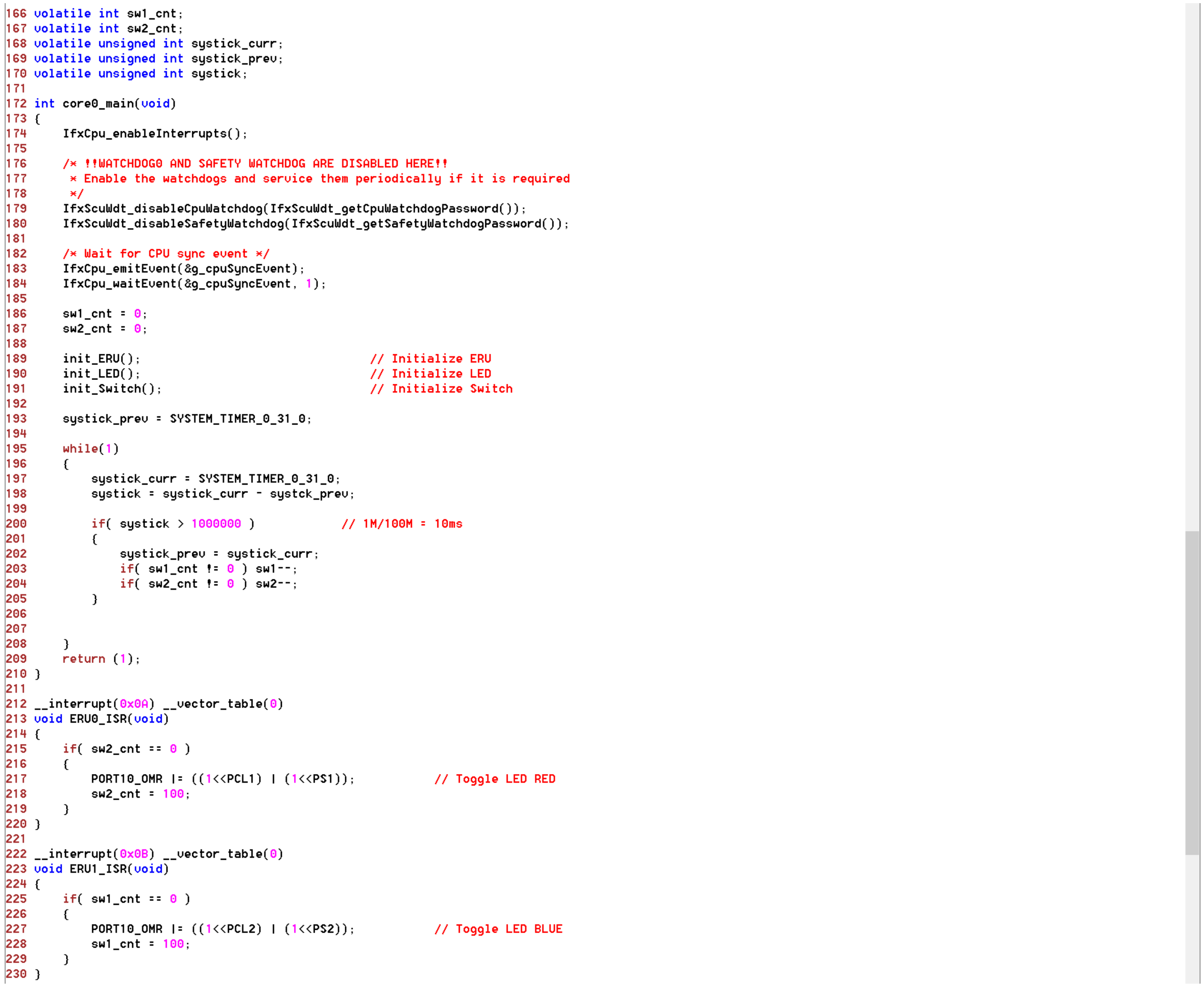
```c
/* Initialize External Request Unit (ERU) */
void init_ERU(void)
{
    /* ERU Input Channel 2 Setting */
    /* Password Access to unlock WDTSCON0 */
    SCU_WDTSCON0 = ((SCU_WDTSCON0 ^ 0xFC) & ~(1 << LCK)) | (1 << ENDINIT);
    while((SCU_WDTSCON0 & (1 << LCK)) != 0);

    // Modify Access to clear ENDINIT bit
    SCU_WDTSCON0 = ((SCU_WDTSCON0 ^ 0xFC) | (1 << LCK)) & ~(1 << ENDINIT);
    while((SCU_WDTSCON0 & (1 << LCK)) == 0);

    SCU_EICR1 &= ~(((0x7) << EXIS0) | ((0x7) << EXIS1));    // External input clear
    SCU_EICR1 &= ~(((0x7) << INP0) | ((0x7) << INP1));      // Input Node Pointer Clear
    SCU_IGCR0 &= ~(((0x3) << IGP0) | ((0x3) << IGP1));      // Interrupt Gating Patten 2, 3 Clear

    SCU_EICR1 |= (0x1 << EXIS0)     |                       // P02.1 select
                 (0x2 << EXIS1)     ;                       // P02.0 select

    SCU_EICR1 |= (0x1 << FEN0)      |                       // Falling edge enable
                 (0x1 << FEN1)      ;

    SCU_EICR1 |= (0x1 << EIEN0)     |                       // The trigger event is enabled
                 (0x1 << EIEN1)     ;

    SCU_EICR1 |= (0x0 << INP0)      |                       // Trigger Input Channel 2 -> Output Channel 0
                 (0x1 << INP1)      ;                       // Trigger Input Channel 3 -> Output Channel 1

    SCU_IGCR0 |= (0x1 << IGP0)      |                       // Input Channel 2 activated, pattern is not considered
                 (0x1 << IGP1)      ;                       // Input Channel 3 activated, pattern is not considered

    /* Password Access to unlock WDTSCON0 */
    SCU_WDTSCON0 = ((SCU_WDTSCON0 ^ 0xFC) & ~(1 << LCK)) | (1 << ENDINIT);
    while((SCU_WDTSCON0 & (1 << LCK)) != 0);

    /* Modify Access to set ENDINIT bit */
    SCU_WDTSCON0 = ((SCU_WDTSCON0 ^ 0xFC) | (1 << LCK)) | (1 << ENDINIT);
    while((SCU_WDTSCON0 & (1 << LCK)) == 0);

    /* SRC Interrupt Setting For ECU */
    SRC_SCUERU0 &= ~((0xFF) << SRPN);           // Set Priority : 0x0A
    SRC_SCUERU0 |= ((0x0A) << SRPN);

    SRC_SCUERU0 |= (1 << SRE);                  // Service Request is enabled

    SRC_SCUERU1  &= ~((0xFF) << SRPN);          // Set Priority : 0x0B
    SRC_SCUERU1 |= ((0x0B) << SRPN);

    SRC_SCUERU1  &= ~((0x3) << TOS);            // CPU0 services

    SRC_SCUERU1  |= (1 << SRE);                 // Service Request is enabled
}
```



```c

__interrupt(0x0A) __vector_table(0)
void ERU0_ISR(void)
{
    PORT10_OMR |= ((1<<PCL1) | (1<<PS1));           // Toggle LED RED
}

__interrupt(0x0B) __vector_table(0)
void ERU1_ISR(void)
{
    PORT10_OMR |= ((1<<PCL2) | (1<<PS2));           // Toggle LED BLUE
}
~
                                                          110,0-1       바닥
```

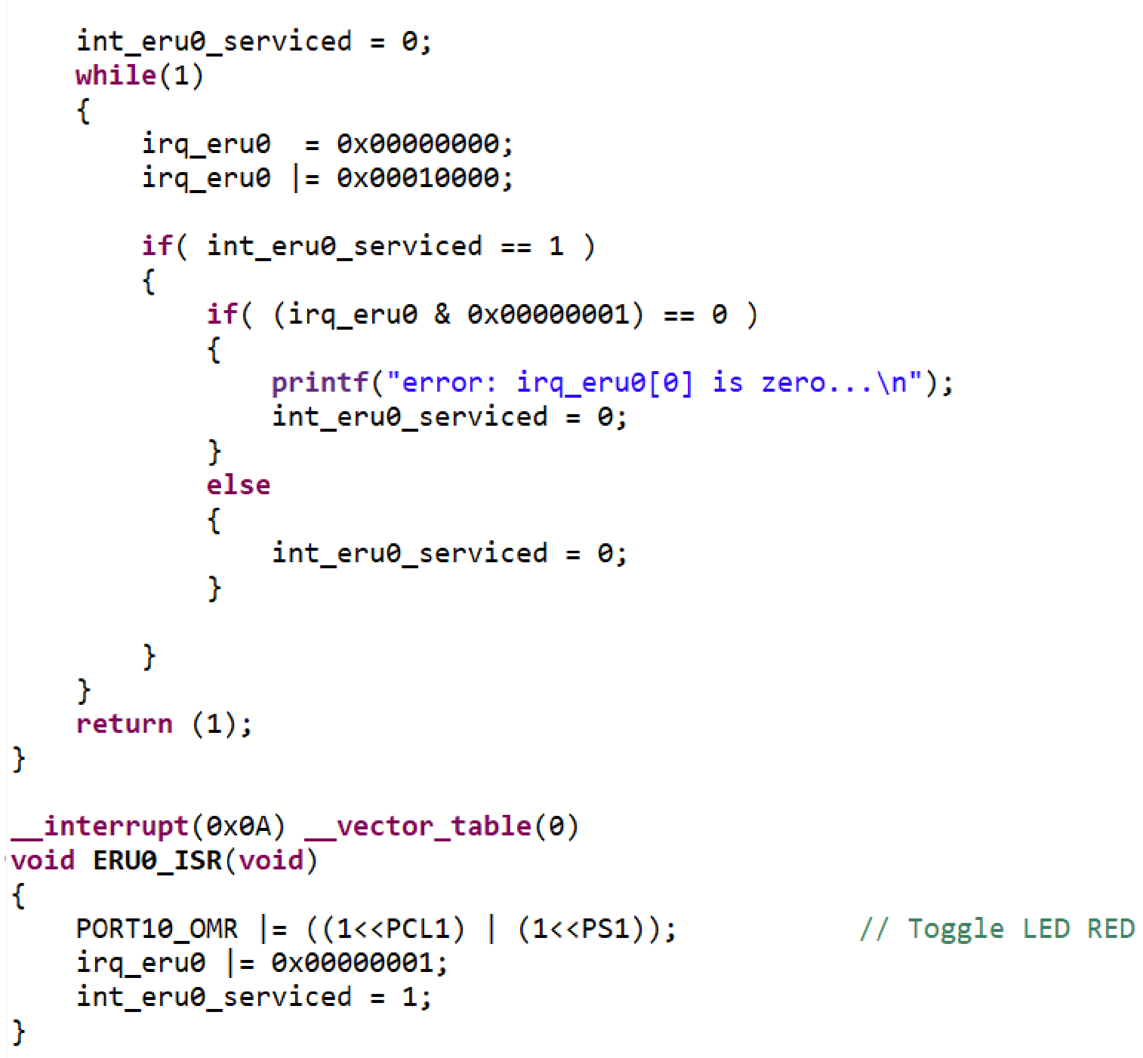
ACE Lab.

Debounce (Software)



```c
volatile int sw1_cnt;
volatile int sw2_cnt;
volatile unsigned int systick_curr;
volatile unsigned int systick_prev;
volatile unsigned int systick;

int core0_main(void)
{
    IfxCpu_enableInterrupts();

    /* !!WATCHDOG0 AND SAFETY WATCHDOG ARE DISABLED HERE!!
     * Enable the watchdogs and service them periodically if it is required
     */
    IfxScuWdt_disableCpuWatchdog(IfxScuWdt_getCpuWatchdogPassword());
    IfxScuWdt_disableSafetyWatchdog(IfxScuWdt_getSafetyWatchdogPassword());

    /* Wait for CPU sync event */
    IfxCpu_emitEvent(&g_cpuSyncEvent);
    IfxCpu_waitEvent(&g_cpuSyncEvent, 1);

    sw1_cnt = 0;
    sw2_cnt = 0;

    init_ERU();                                 // Initialize ERU
    init_LED();                                 // Initialize LED
    init_Switch();                              // Initialize Switch

    systick_prev = SYSTEM_TIMER_0_31_0;

    while(1)
    {
        systick_curr = SYSTEM_TIMER_0_31_0;
        systick = systick_curr - systck_prev;

        if( systick > 1000000 )            // 1M/100M = 10ms
        {
            systick_prev = systick_curr;
            if( sw1_cnt != 0 ) sw1--;
            if( sw2_cnt != 0 ) sw2--;
        }


    }
    return (1);
}

__interrupt(0x0A) __vector_table(0)
void ERU0_ISR(void)
{
    if( sw2_cnt == 0 )
    {
        PORT10_OMR |= ((1<<PCL1) | (1<<PS1));           // Toggle LED RED
        sw2_cnt = 100;
    }
}

__interrupt(0x0B) __vector_table(0)
void ERU1_ISR(void)
{
    if( sw1_cnt == 0 )
    {
        PORT10_OMR |= ((1<<PCL2) | (1<<PS2));           // Toggle LED BLUE
        sw1_cnt = 100;
    }
}
```

Bit Operation @ Interrupt



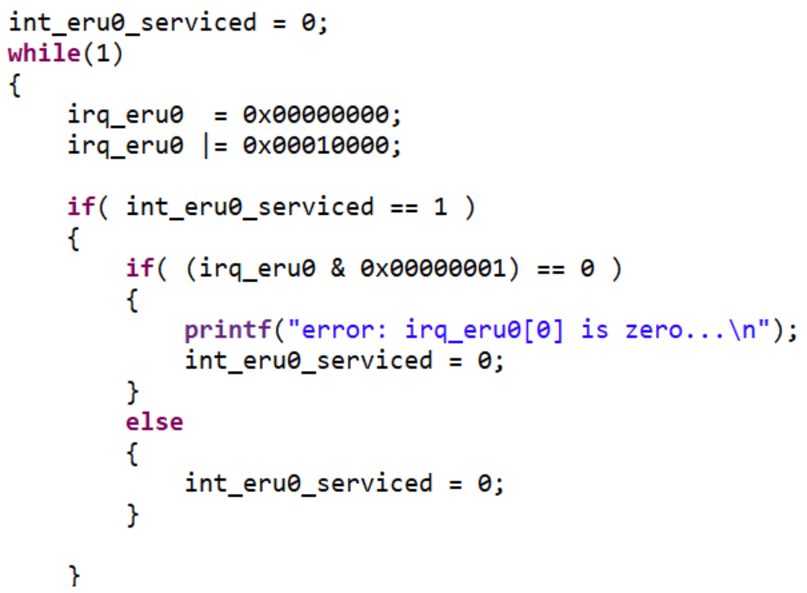
```c
    int_eru0_serviced = 0;
    while(1)
    {
        irq_eru0  = 0x00000000;
        irq_eru0 |= 0x00010000;

        if( int_eru0_serviced == 1 )
        {
            if( (irq_eru0 & 0x00000001) == 0 )
            {
                printf("error: irq_eru0[0] is zero...\n");
                int_eru0_serviced = 0;
            }
            else
            {
                int_eru0_serviced = 0;
            }

        }
    }
    return (1);
}

__interrupt(0x0A) __vector_table(0)
void ERU0_ISR(void)
{
    PORT10_OMR |= ((1<<PCL1) | (1<<PS1));           // Toggle LED RED
    irq_eru0 |= 0x00000001;
    int_eru0_serviced = 1;
}
```

ACE Lab.

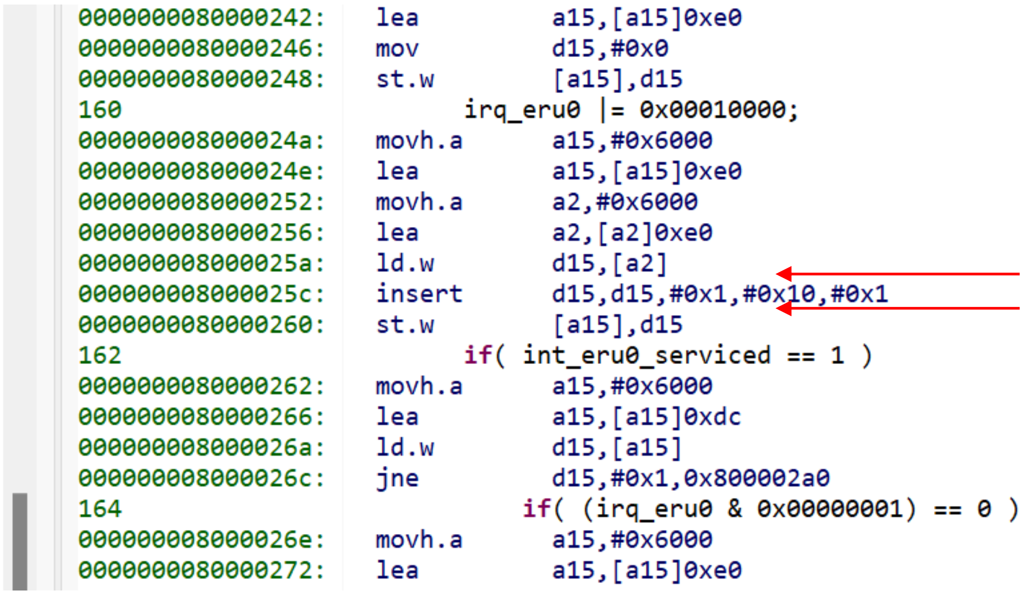
Bit Operation @ Interrupt

Read
Modify
Write

Read와 Write 사이에 Interrupt 발생 시 Abnormal Operation
해결방법:
- Write Only: Set, Clear 와 같은 형식으로
- Read 전 Interrupt Disable, Write 이후 Interrupt Enable



```
int_eru0_serviced = 0;
while(1)
{
    irq_eru0  = 0x00000000;
    irq_eru0 |= 0x00010000;

    if( int_eru0_serviced == 1 )
    {
        if( (irq_eru0 & 0x00000001) == 0 )
        {
            printf("error: irq_eru0[0] is zero...\n");
            int_eru0_serviced = 0;
        }
        else
        {
            int_eru0_serviced = 0;
        }

    }
```



```
0000000080000242:   lea         a15,[a15]0xe0
0000000080000246:   mov         d15,#0x0
0000000080000248:   st.w        [a15],d15
160                       irq_eru0 |= 0x00010000;
000000008000024a:   movh.a      a15,#0x6000
000000008000024e:   lea         a15,[a15]0xe0
0000000080000252:   movh.a      a2,#0x6000
0000000080000256:   lea         a2,[a2]0xe0
000000008000025a:   ld.w        d15,[a2]
000000008000025c:   insert      d15,d15,#0x1,#0x10,#0x1
0000000080000260:   st.w        [a15],d15
162                       if( int_eru0_serviced == 1 )
0000000080000262:   movh.a      a15,#0x6000
0000000080000266:   lea         a15,[a15]0xdc
000000008000026a:   ld.w        d15,[a15]
000000008000026c:   jne         d15,#0x1,0x800002a0
164                           if( (irq_eru0 & 0x00000001) == 0 )
000000008000026e:   movh.a      a15,#0x6000
0000000080000272:   lea         a15,[a15]0xe0
```

NG

ACE Lab.

Q & A

Thank you for your attention

Architecture and
Compiler
for Embedded Systems Lab.

School of Electronics Engineering, KNU

ACE Lab (hn02301@gmail.com)