임베디드 MCU 프로그래밍 실습

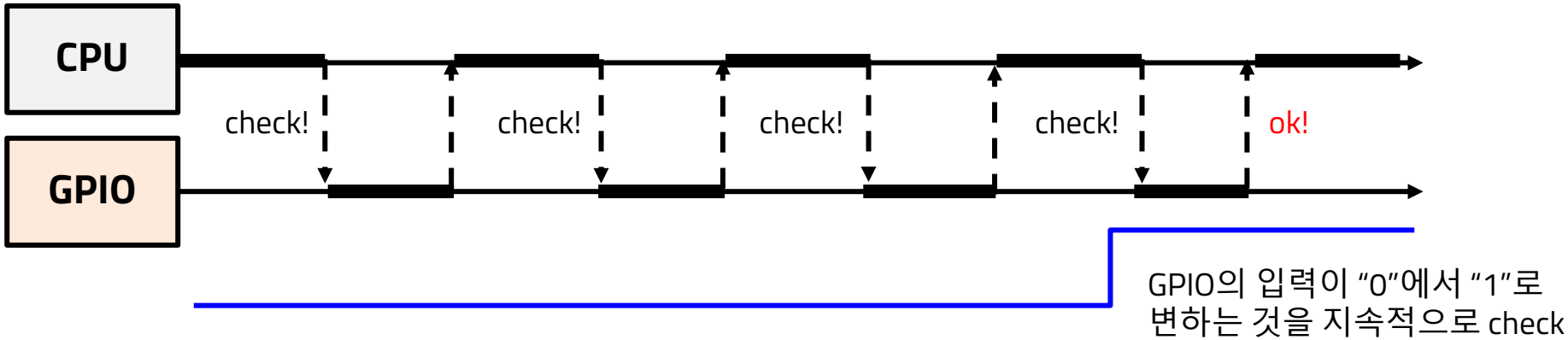
External Interrupt-Driven Asynchronous Hardware-Software Interaction

GPIO Interrupt-Driven Software Execution for TC275 Infineon Embedded Processors

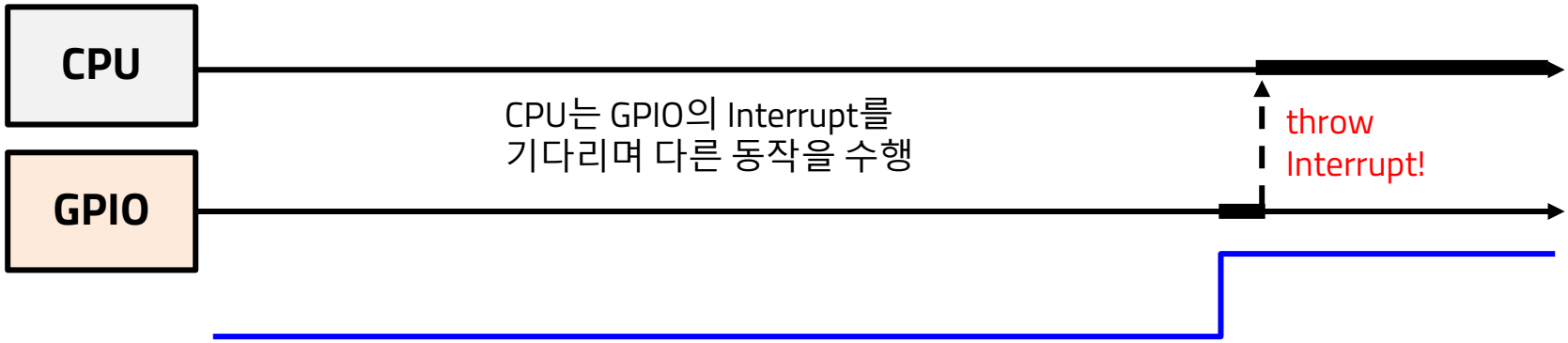
현대자동차 입문교육
박대진 교수

Polling vs Interrupt

- 지금까지 Push 버튼이 눌렸는지 확인하기 위해 사용한 **Polling 기반 방법**
  - CPU가 항상 GPIO를 신경 쓰고 있어야 함



- Push 버튼이 눌렸을 때 **Interrupt가 발생하도록 하는 방법**
  - GPIO 모니터링 필요없이 CPU가 다른 일을 하거나 잠잘수 있음 (IDLE)



실습 목표

1. 보드의 GPIO와 연결된 External Interrupt를 사용해서 Push 버튼이 눌리는 순간에 LED를 toggle하는 ISR 함수가 호출되도록 한다.

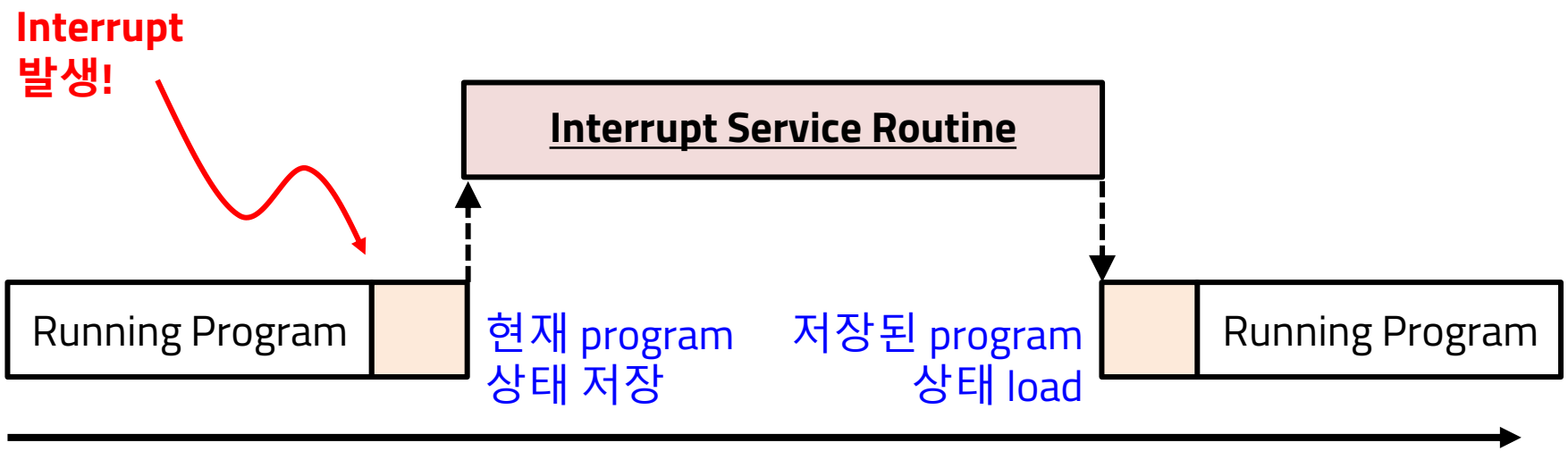
다음 3개의 레지스터를 Access하여 인터럽트 하드웨어를 구동한다.

- EICR
- IGCR
- SCUERU

ISR (Interrupt Service Routine)

- Interrupt가 발생하면, 원래 CPU에서 수행하던 프로그램은?

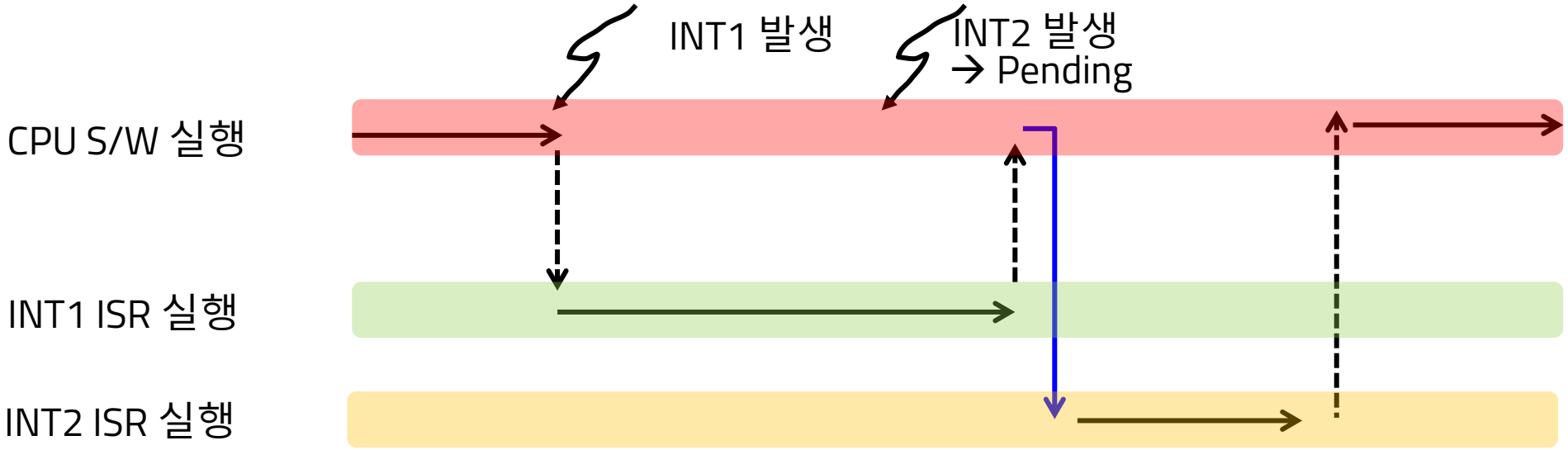
→ 잠시 CPU에서 수행하던 프로그램을 pause하고, **ISR로 jump**

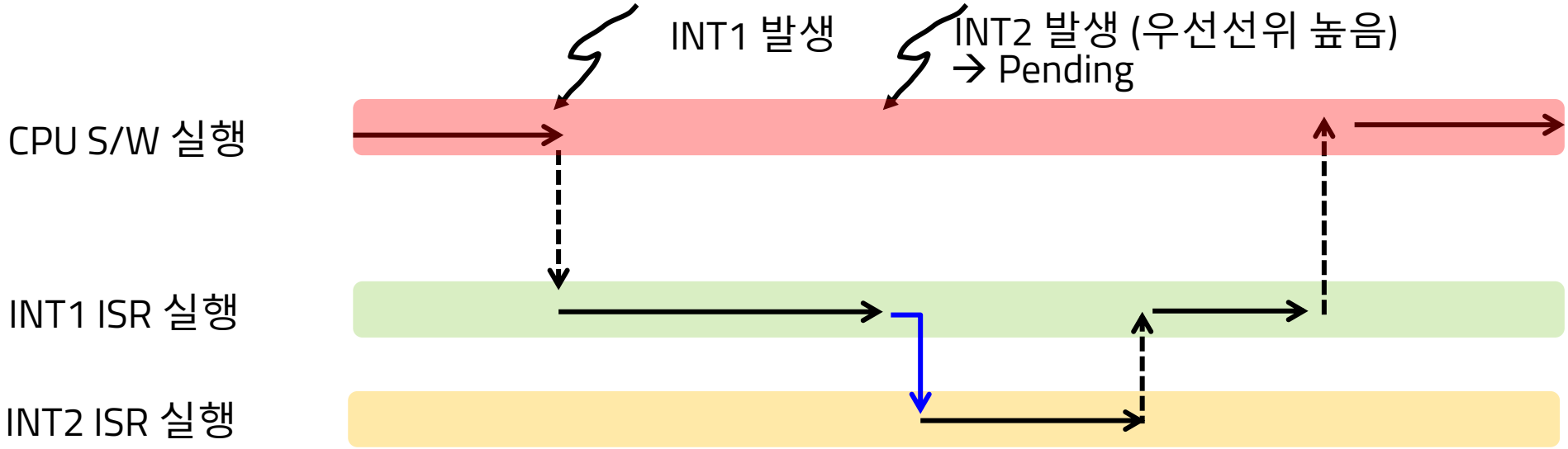


- Interrupt가 발생하면, **하드웨어에 의해** 해당 interrupt 에 지정된 ISR 함수가 자동으로 수행됨

→ External Interrupt에 반응할 수 있도록 하드웨어 레지스터 설정 필요

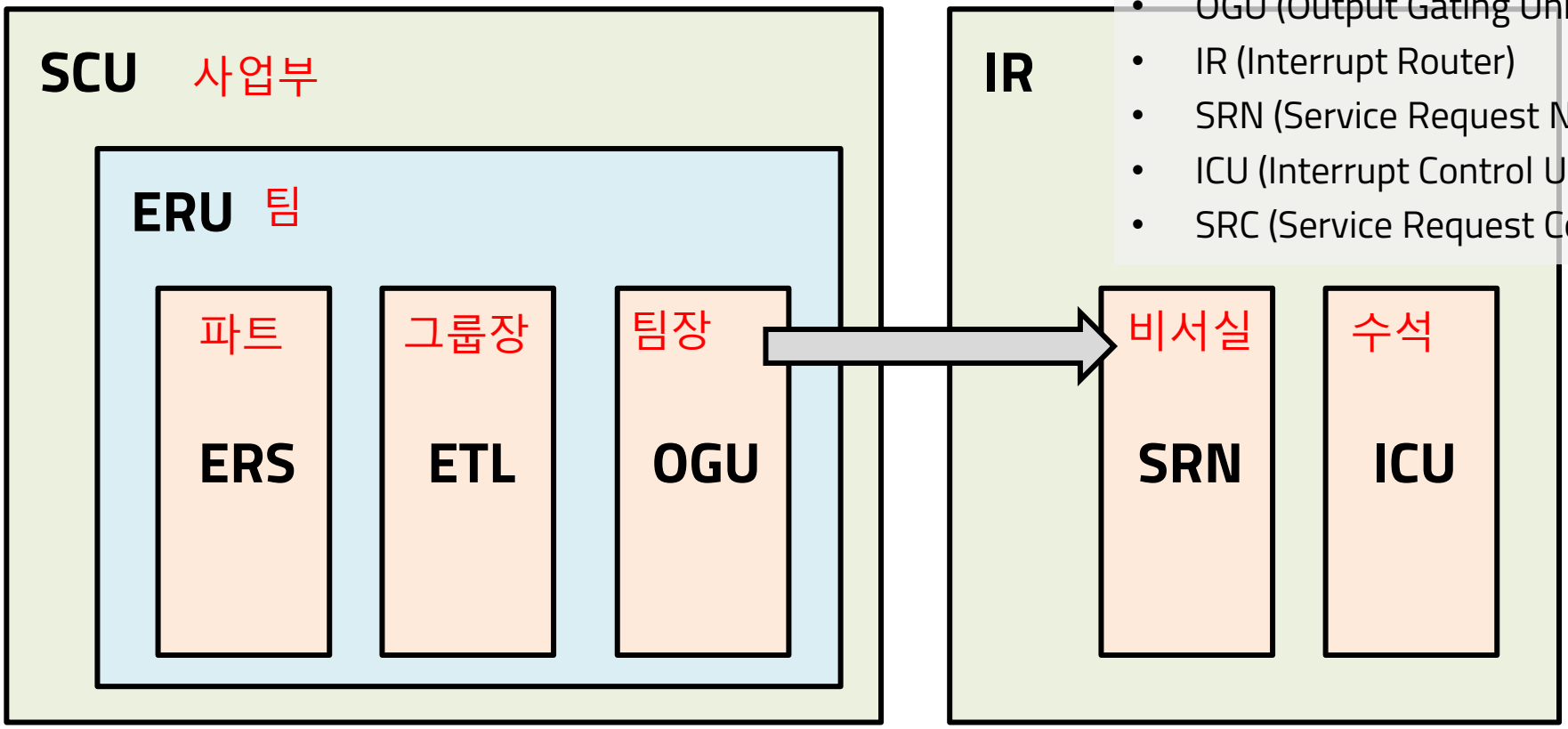
인터럽트 우선순위와 멀티인터럽트 ISR





TC275 Interrupt 모듈 계층 구조

- SCU (System Control Unit)
- ERU (External Request Unit)
- ERS (External Request Selection)
- ETL (External Trigger Logic)
- OGU (Output Gating Unit)
- IR (Interrupt Router)
- SRN (Service Request Node)
- ICU (Interrupt Control Units)
- SRC (Service Request Control)



External Interrupt 처리 Flow

: External Request Unit (ERU)

- MCU에서 사용가능한 Interrupt 는 여러가지 종류가 존재

→ 본 실습에서 Interrupt의 대상이 되는 외부 GPIO 입력은 “External Request”에 해당

- External Request 를 처리하는 곳 → **System Control Unit (SCU) 내 External Request Unit (ERU)**

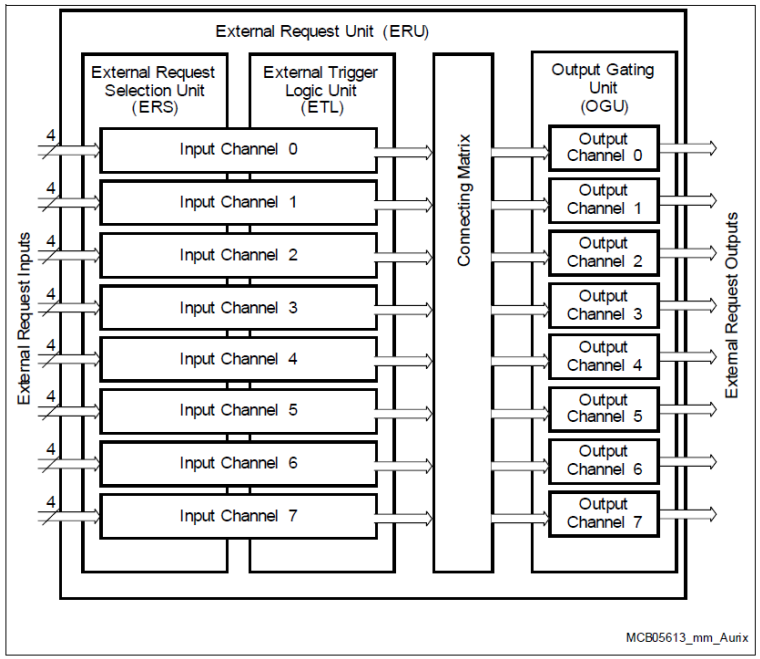


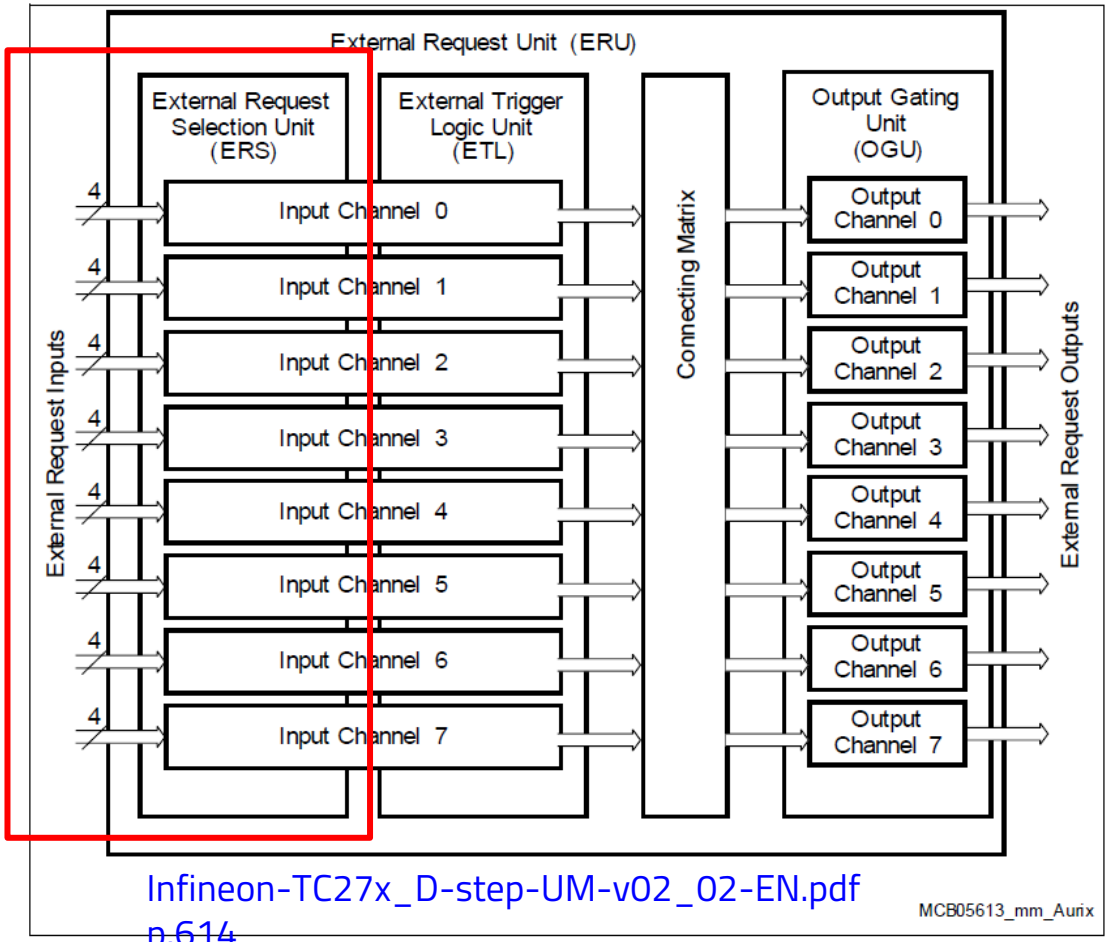
Figure 7-40 External Request Unit Overview

Infineon-TC27x_D-step-UM-v02_02-EN.pdf
p.612

External Interrupt 처리 Flow

: External Request Selection Unit (ERS)

1. **External Request Selection Unit (ERS)**의 역할
   - 각각의 Input Channel n 에 연결된 4개의 입력 중, 1개 입력을 선택



Infineon-TC27x_D-step-UM-v02_02-EN.pdf
p.614

Figure 7-40 External Request Unit Overview

ERS 구성의 일부

- GPIO Push 버튼이 어디에 연결되어 있는지 확인해야 함

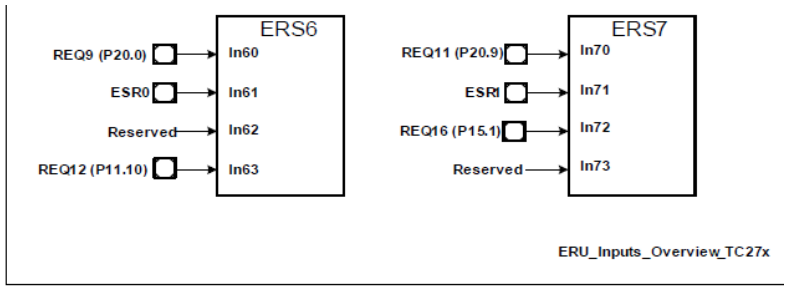
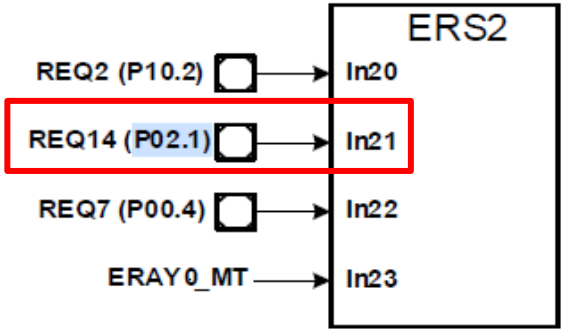


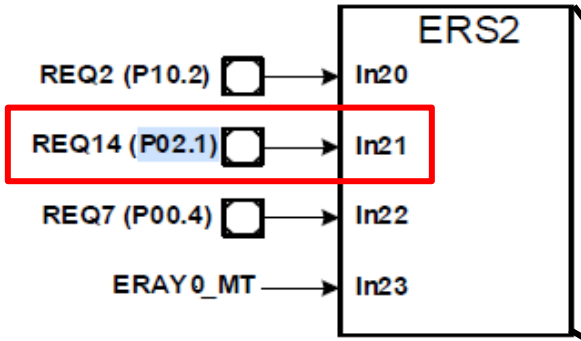
Figure 7-41 External Request Unit Input Connections for TC27x



GPIO와 External Interrupt 연결 위한 레지스터 설정

: ERU External Input Channel 설정 → EICR 레지스터 설정

- 본 실습에서 GPIO 입력으로 사용하는 핀 → **P02.1**
- 해당 핀은 어느 ERS에 입력? → **ERS2 의 In21 (REQ14)**



Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.616

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.614

- ERU의 External Input Channel 설정을 위해서는 어떤 레지스터 설정이 필요?
  → 데이터 시트에서 찾을 수 있음

- SCU 레지스터 항목에서
  - **EICR** 레지스터 설정 필요

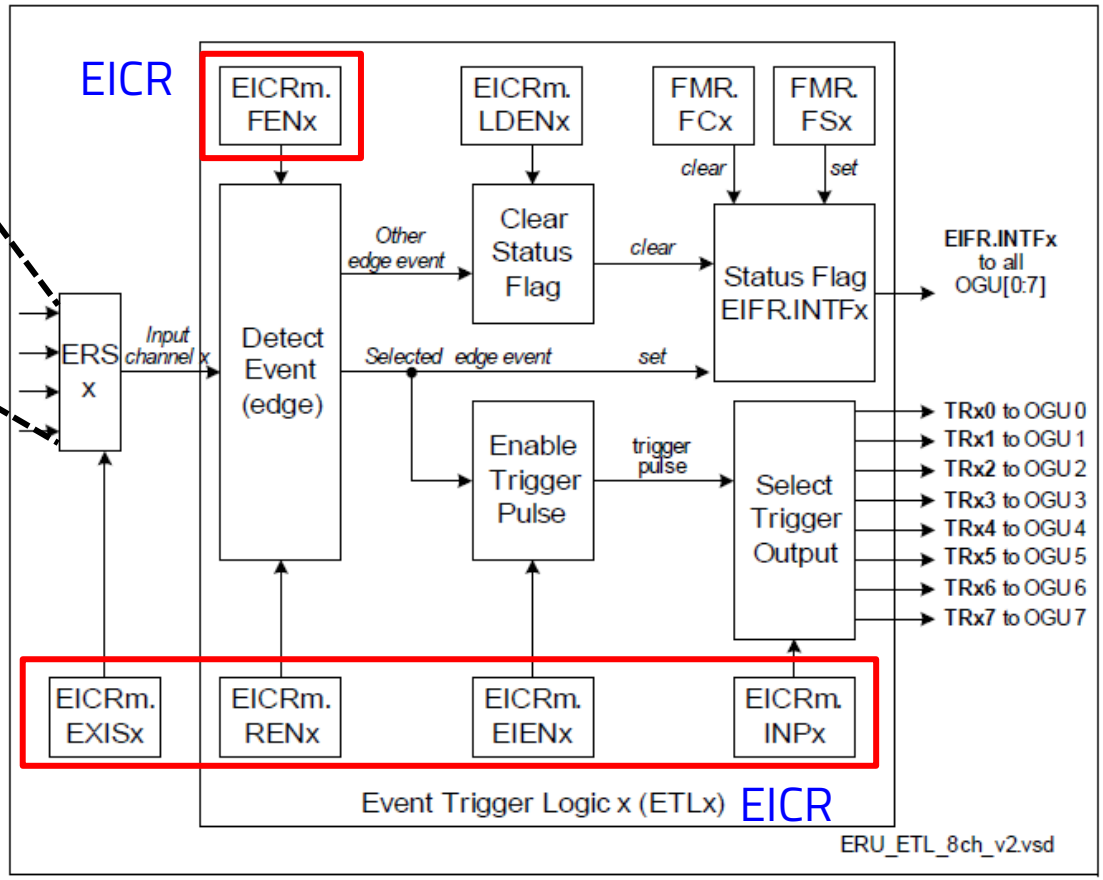


Figure 7-43 Event Trigger Logic Overview

SCU 레지스터 설정 – EICR
: 레지스터 주소 계산

1. SCU 레지스터 영역 시작 주소 (Base address) **= 0xF0036000**

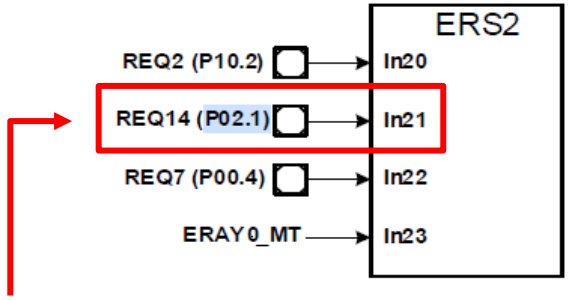
2. 사용할 레지스터의 주소 찾기
   - EICRi (i번째) 레지스터에는 ERS 채널 2개씩 할당
     - EICR0: ERS0-1 , **EICR1: ERS2-3** … , EICR2: ERS4-5

   - **EICR1** 의 Offset Address = **0x214**
   - **즉, Base Address로부터 0x214만큼 떨어져 있음**

   → EICR1 레지스터 주소

   = [SCU Base Addr] + [EICR1 Offset Addr]

   = 0xF0036000 + 0x214 = **0xF0036214**



본 실습의 P02.1 핀은 **ERS2**를 사용하므로

→ EICR1 사용

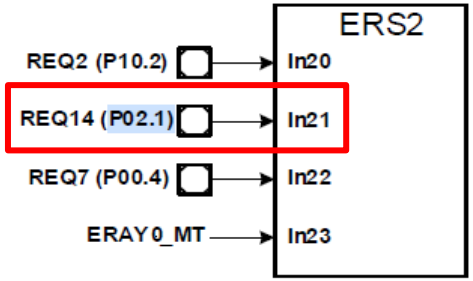
Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.624

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	$21C_H$	U, SV	U, SV, P	Application	Page

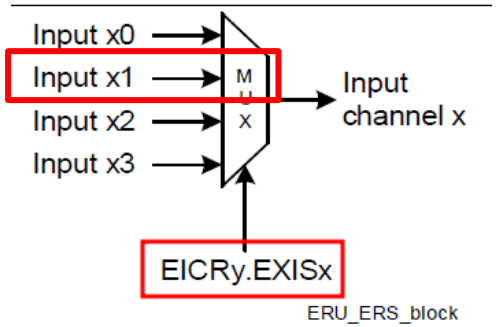
Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.698

SCU 레지스터 설정 – EICR
: 채널 선택 – EXIS0

3. 레지스터 write 값 결정
   - ERS Channel 2 의 4개 External Request 입력 중, 1번째 입력 선택



Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.614



Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.615

   - **ERS2의 1번째 입력을 선택하기 위해 EXIS0 영역의 값을 0x1로 (001b) 결정**

ERS3 설정 →

ERS2 설정 →

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			

EICR1 레지스터 @ 0xF0036214

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.624

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

Lab1: SCU 레지스터 설정 – EICR의 EXIS0 설정

```c
#define EXIS0_BIT_LSB_IDX          4
```

```c
void initERU(void)
{
    // ERU setting
    SCU_EICR1.U &= ~(0x7 << EXIS0_BIT_LSB_IDX);
    SCU_EICR1.U |= (0x1  << EXIS0_BIT_LSB_IDX);

    SCU_EICR1.U |= 0x1   << FEN0_BIT_LSB_IDX;

    SCU_EICR1.U |= 0x1   << EIEN0_BIT_LSB_IDX;

    SCU_EICR1.U &= ~(0x7 << INP0_BIT_LSB_IDX);

    SCU_IGCR0.U &= ~(0x3 << IGP0_BIT_LSB_IDX);
    SCU_IGCR0.U |= 0x1   << IGP0_BIT_LSB_IDX;


    // SRC Interrupt setting
    SRC_SCU_SCU_ERU0.U &= ~(0xFF << SRPN_BIT_LSB_IDX);
    SRC_SCU_SCU_ERU0.U |= 0x0A   << SRPN_BIT_LSB_IDX;

    SRC_SCU_SCU_ERU0.U &= ~(0x3  << TOS_BIT_LSB_IDX);

    SRC_SCU_SCU_ERU0.U |= 1      << SRE_BIT_LSB_IDX;
}
```

External Interrupt 처리 Flow

: External Trigger Logic Unit (ETL)

2. **External Trigger Logic Unit (ETL)**의 역할
   - Interrupt 이벤트가 Input Channel 입력의 상승(rising) 엣지에서 발생하는지 또는 하강(falling) 엣지에서 발생하는지 ETL에 설정 필요
   - ETL 설정에 따라 **Input Channel 에서 입력된 신호의 상승(rising) 또는 하강(falling) 엣지에서 Trigger 이벤트 발생 시킴**

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.616

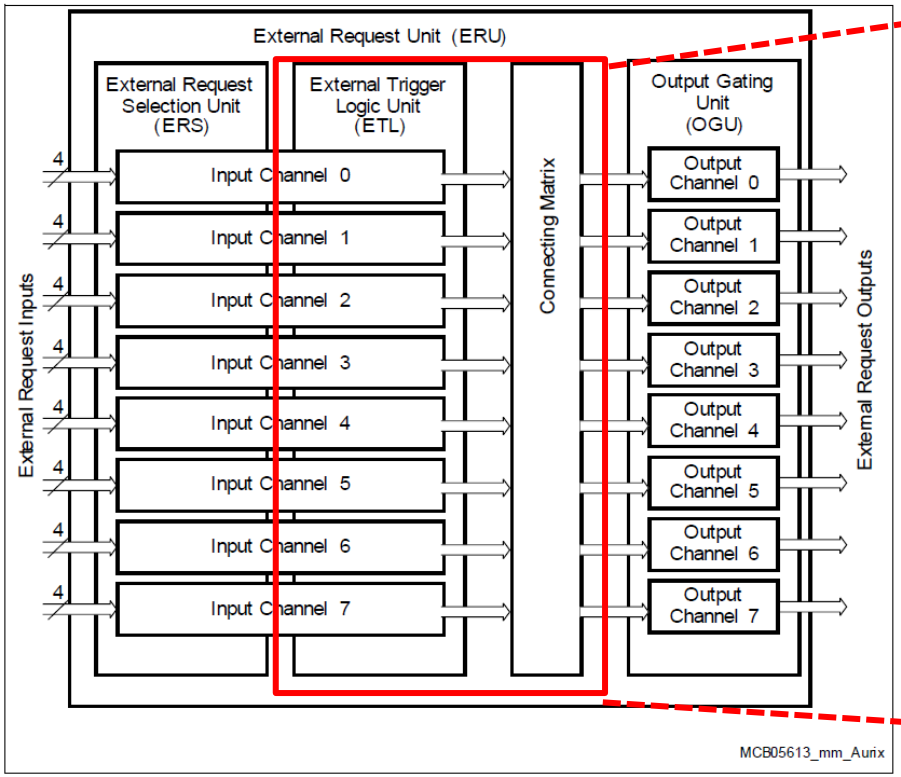


Figure 7-40 External Request Unit Overview

엣지 검출 (detection)

Rising

Falling

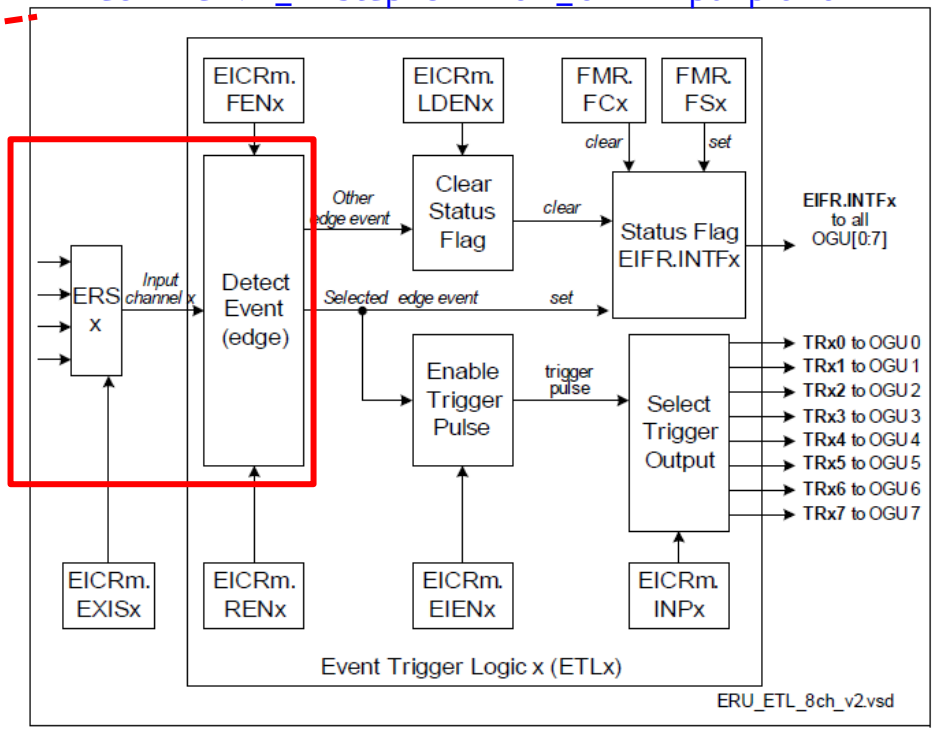


Figure 7-43 Event Trigger Logic Overview

SCU 레지스터 설정 – EICR

: Falling Edge 선택, Trigger Enable – FEN0, EIEN0

3. 레지스터 write 값 결정
   - 본 실습에서 사용하는 Push 버튼은 눌렸을 때 LOW값이 되며 하강(falling) 엣지가 발생
   - Falling 엣지가 발생했을 때 Trigger를 생성하도록 **FEN0 영역의 값을 "1"로 설정**
   - Trigger 생성을 enable 하기 위해 **EIEN0 영역의 값을 "1"로 설정**

EICR1 레지스터 @ 0xF0036214

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
ERS3 설정 →	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
ERS2 설정 →	0	INP0			EI EN0	LD EN0	R EN0	F EN0	0	EXIS0			0			
	r	rw			rw	rw	rw	rw	r	rw			r			

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.625

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.
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

Lab2: SCU 레지스터 설정 – EICR의 FEN0, EIEN0 설정

```
40  #define EXIS0_BIT_LSB_IDX          4
41  #define FEN0_BIT_LSB_IDX           8
42  #define EIEN0_BIT_LSB_IDX          11
```

```
105
106 void initERU(void)
107 {
108     // ERU setting
109     SCU_EICR1.U &= ~(0x7 << EXIS0_BIT_LSB_IDX);
110     SCU_EICR1.U |= (0x1  << EXIS0_BIT_LSB_IDX);
111
112     SCU_EICR1.U |= 0x1   << FEN0_BIT_LSB_IDX;
113
114     SCU_EICR1.U |= 0x1   << EIEN0_BIT_LSB_IDX;
115
116     SCU_EICR1.U &= ~(0x7 << INP0_BIT_LSB_IDX);
117
118     SCU_IGCR0.U &= ~(0x3 << IGP0_BIT_LSB_IDX);
119     SCU_IGCR0.U |= 0x1   << IGP0_BIT_LSB_IDX;
120
121
122     // SRC Interrupt setting
123     SRC_SCU_SCU_ERU0.U &= ~(0xFF << SRPN_BIT_LSB_IDX);
124     SRC_SCU_SCU_ERU0.U |= 0x0A   << SRPN_BIT_LSB_IDX;
125
126     SRC_SCU_SCU_ERU0.U &= ~(0x3  << TOS_BIT_LSB_IDX);
127
128     SRC_SCU_SCU_ERU0.U |= 1      << SRE_BIT_LSB_IDX;
129 }
130
```

External Interrupt 처리 Flow

: Output Gating Unit (OGU)

3. **Output Gating Unit (OGU)**의 역할

- Input Channel 로부터 생성된 Trigger 이벤트를 조합하여 **Request 출력 생성**
- 하나의 이벤트가 여러 채널로 연결되거나, 여러 이벤트가 하나의 채널에서 패턴을 만들 수 있음

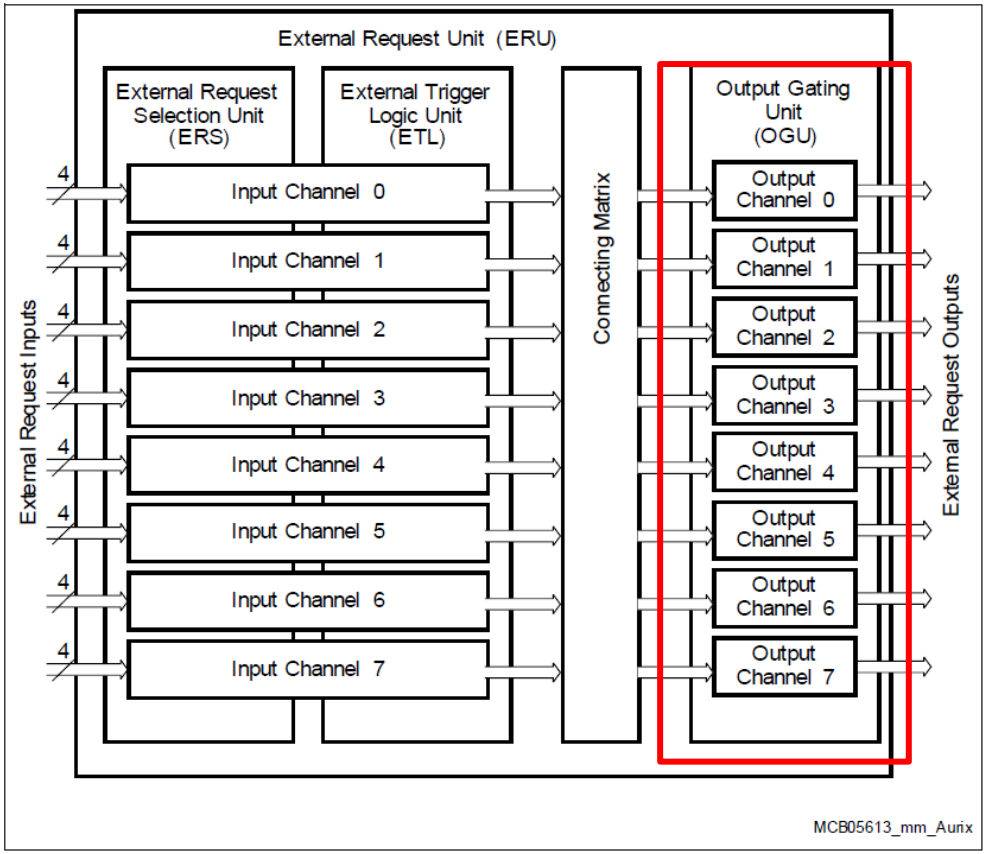


Figure 7-40 External Request Unit Overview

Infineon-TC27x_D-step-UM-v02_02-EN.pdf
p.612

SCU 레지스터 설정 – EICR
: 출력 채널 설정

3. 레지스터 write 값 결정
   - 생성되는 Trigger 신호를 Interrupt 신호를 출력으로 전달하기 위한 레지스터 설정
   - OGU의 Channel 0 으로 전달하기 위해 **INP0 영역의 값을 0x0 (000b) 으로 설정**

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
ERS3 설정 →	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
ERS2 설정 →	0	INP0			EI EN0	LD EN0	R EN0	F EN0	0	EXIS0			0			
	r	rw			rw	rw	rw	rw	r	rw			r			

EICR1 레지스터 @ 0xF0036214

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)

레지스터 설정에 의한 데이터 흐름

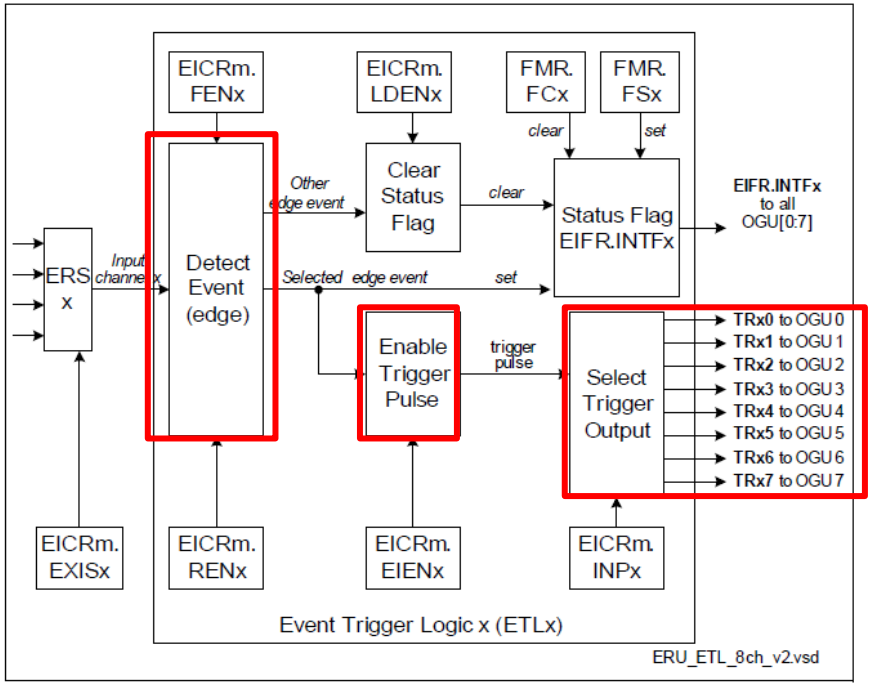


Figure 7-43 Event Trigger Logic Overview

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.625

Lab3: SCU 레지스터 설정 – EICR의 INFO 설정

```c
#define EXIS0_BIT_LSB_IDX          4
#define FEN0_BIT_LSB_IDX           8
#define EIEN0_BIT_LSB_IDX          11
#define INP0 BIT LSB IDX           12
```

```c
void initERU(void)
{
    // ERU setting
    SCU_EICR1.U &= ~(0x7 << EXIS0_BIT_LSB_IDX);
    SCU_EICR1.U |= (0x1  << EXIS0_BIT_LSB_IDX);

    SCU_EICR1.U |= 0x1   << FEN0_BIT_LSB_IDX;

    SCU_EICR1.U |= 0x1   << EIEN0_BIT_LSB_IDX;

    SCU_EICR1.U &= ~(0x7 << INP0_BIT_LSB_IDX);

    SCU_IGCR0.U &= ~(0x3 << IGP0_BIT_LSB_IDX);
    SCU_IGCR0.U |= 0x1   << IGP0_BIT_LSB_IDX;


    // SRC Interrupt setting
    SRC_SCU_SCU_ERU0.U &= ~(0xFF << SRPN_BIT_LSB_IDX);
    SRC_SCU_SCU_ERU0.U |= 0x0A   << SRPN_BIT_LSB_IDX;

    SRC_SCU_SCU_ERU0.U &= ~(0x3  << TOS_BIT_LSB_IDX);

    SRC_SCU_SCU_ERU0.U |= 1      << SRE_BIT_LSB_IDX;
}
```

GPIO와 External Interrupt 연결 위한 레지스터 설정

: ERU 최종 IOUT 출력 포트 설정

- SCU 레지스터 항목에서
  - **IGCR** 레지스터 설정 필요

- **OGU**에서 입력 받은 Trigger 이벤트를 **ERU의 IOUT 포트로 출력**하기 위한 레지스터 설정 필요함

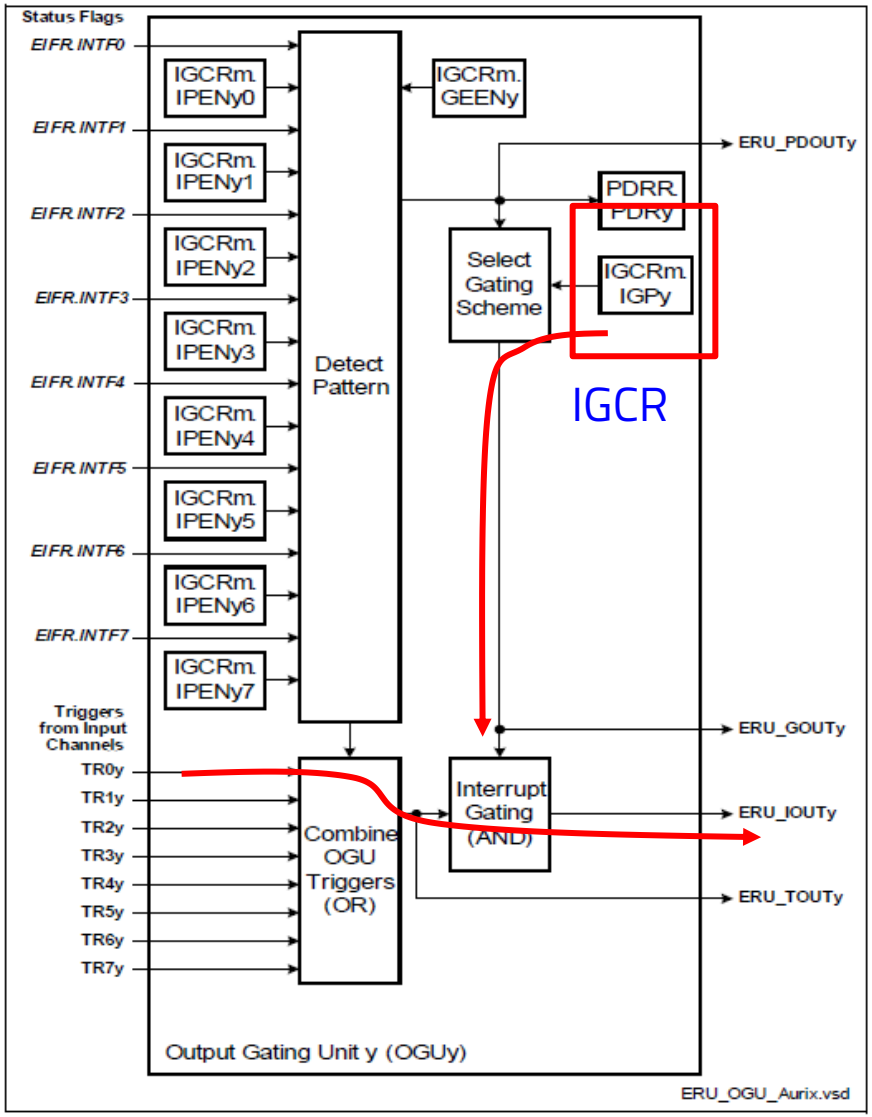


Figure 7-45 Output Gating Unit for Output Channel y

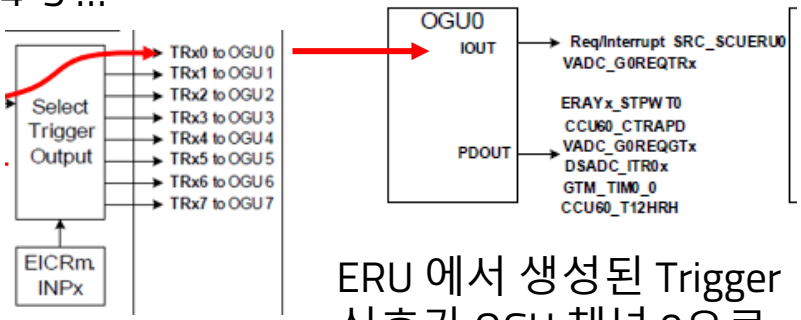
Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.620

SCU 레지스터 설정 – IGCR 레지스터 : 주소 계산

1. SCU 레지스터 영역 시작 주소 (Base address) **= 0xF0036000**

2. 사용할 레지스터의 주소 찾기
   - IGCRj (j번째) 레지스터는 OGU 채널 2개를 제어함
     - **IGCR0: OGU0-1** , IGCR1: OGU2-3 … , IGCR2: OGU4-5 …

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.623



ERU 에서 생성된 Trigger 신호가 OGU 채널 0으로 연결되므로
→ **IGCR0 사용**

   - **IGCR0** 의 Offset Address = **0x22C**
   - **즉, Base Address로부터 0x22만큼 떨어져 있음**

→ IGCR0 레지스터 주소
= [SCU Base Addr] + [IGCR0 Offset Addr]
= 0xF0036000 + 0x22 = **0xF003622C**

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.699

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

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

SCU 레지스터 설정 – IGCR

: Trigger 이벤트를 최종 출력으로 생성되도록 설정

3. 레지스터 write 값 결정
   - 입력되는 Trigger 신호를 OGU0의 IOUT 으로 연결 위해
   - **IGP0 영역의 값을 0x1 (01b) 으로 설정**

IGCR0 레지스터 @ 0xF0036022

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

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	IGP1		GE EN1	0					IPEN 17	IPEN 16	IPEN 15	IPEN 14	IPEN 13	IPEN 12	IPEN 11	IPEN 10
	rw		rw	r					rw	rw	rw	rw	rw	rw	rw	rw

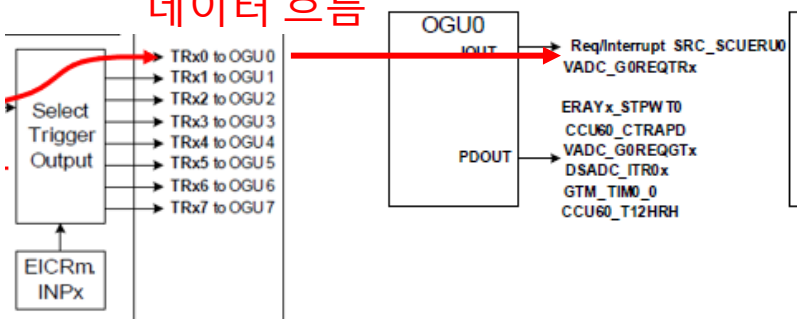
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	IGP0		GE EN0	0					IPEN 07	IPEN 06	IPEN 05	IPEN 04	IPEN 03	IPEN 02	IPEN 01	IPEN 00
	rw		rw	r					rw	rw	rw	rw	rw	rw	rw	rw

OGU1 설정

OGU0 설정

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.

레지스터 설정에 의한 데이터 흐름



Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.623

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.631

Lab4: SCU 레지스터 설정 – IGCR의 IGP0 설정

```c
#define EXIS0_BIT_LSB_IDX          4
#define FEN0_BIT_LSB_IDX           8
#define EIEN0_BIT_LSB_IDX          11
#define INP0_BIT_LSB_IDX           12
#define IGP0_BIT_LSB_IDX           14
```

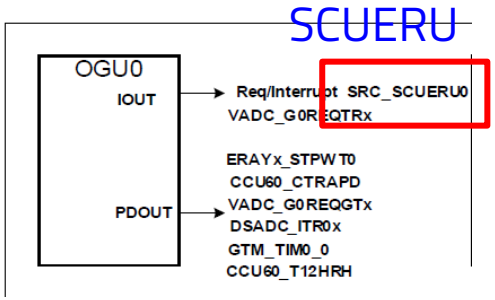
```c
void initERU(void)
{
    // ERU setting
    SCU_EICR1.U &= ~(0x7 << EXIS0_BIT_LSB_IDX);
    SCU_EICR1.U |= (0x1  << EXIS0_BIT_LSB_IDX);

    SCU_EICR1.U |= 0x1   << FEN0_BIT_LSB_IDX;

    SCU_EICR1.U |= 0x1   << EIEN0_BIT_LSB_IDX;

    SCU_EICR1.U &= ~(0x7 << INP0_BIT_LSB_IDX);

    SCU_IGCR0.U &= ~(0x3 << IGP0_BIT_LSB_IDX);
    SCU_IGCR0.U |= 0x1   << IGP0_BIT_LSB_IDX;


    // SRC Interrupt setting
    SRC_SCU_SCU_ERU0.U &= ~(0xFF << SRPN_BIT_LSB_IDX);
    SRC_SCU_SCU_ERU0.U |= 0x0A   << SRPN_BIT_LSB_IDX;

    SRC_SCU_SCU_ERU0.U &= ~(0x3  << TOS_BIT_LSB_IDX);

    SRC_SCU_SCU_ERU0.U |= 1      << SRE_BIT_LSB_IDX;
}
```

GPIO와 External Interrupt 연결 위한 레지스터 설정

: ERU IOUT 출력 Interrupt를 IR에 연결

- SRC 레지스터 항목에서
  - **SCUERU** 레지스터 설정 필요

- (1) ERU의 IOUT 포트로 출력된 Interrupt IR의 SRN과 연결.
- (2) IR에 입력된 Interrupt 신호를 특정 CPU에 연결하기 위한 레지스터 설정 필요함



Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.623

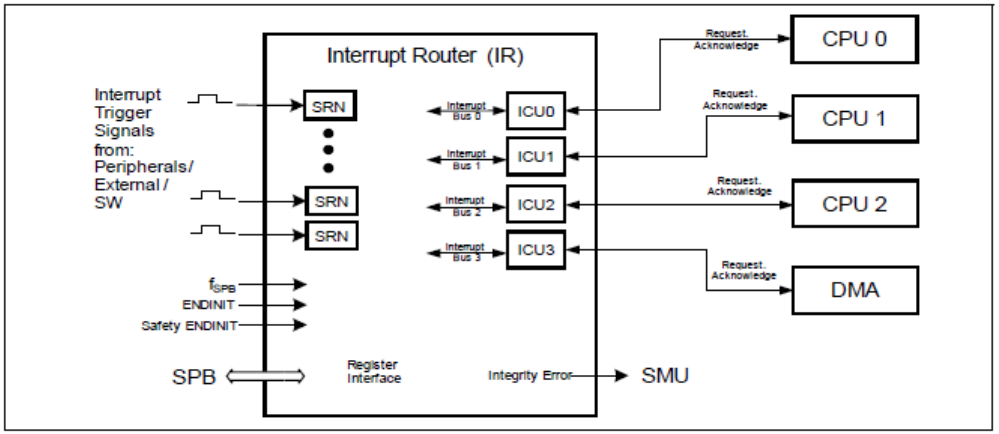


Figure 16-1 Block Diagram of the TC27x Interrupt System

SRC 레지스터 설정 – SCUERU0

: Base 주소 계산

1. SRC 레지스터 영역의 주소 찾기
   - 시작 주소 (Base address)
   - **= 0xF0038000**

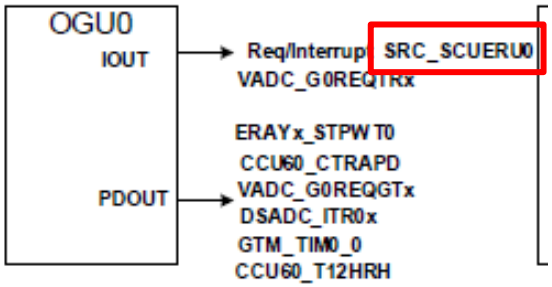
Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.230

Interrupt Router (IR)	$F003\ 7000_H$ - $F003\ 7FFF_H$	4 Kbyte	access	access
Interrupt Router (IR) SRC Registers	$F003\ 8000_H$ - $F003\ 9FFF_H$	8 Kbyte	access	access
Port 00	$F003\ A000_H$ - $F003\ A0FF_H$	256 byte	access	access

2. 사용할 레지스터의 주소 찾기
   - SCUERU"0" 이므로 m=0
   - **SCUERU0 (m=0)**의 Offset Address = 0xCD4 + (m * 0x4) = **0xCD4**
   - **즉, Base Address로부터 0xCD4만큼 떨어져 있음**

→ SCUERU0 레지스터 주소 = [SCU Base Addr] + [SCUERU0 Offset Addr]

= 0xF0038000 + 0xCD4 = **0xF0038CD4**



OGU Channel 0 의 IOUT 출력은 SRC 의 **SCUERU0 레지스터**와 연결

TS		Request		SV	P0, P1		
SRC_SCUERUm	SCU	SCU ERU Service Request x (m = 0-3)	$0CD4_H$+ (m x 4_H)	U, SV	SV, P0, P1	3	259 + m
		Reserved	$0CE4_H$ $0D0C_H$	BE	BE	-	-
SRC_SMU	SMU	SMU Service Request m	$0D10_H$+	U,	SV,	3	263

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.1577

SRC 레지스터 설정 – SCUERU0

: SRPN, SRE, TOS 설정

3. 레지스터 write 값 결정
   - Interrupt 의 우선순위 및 SW에서 ISR을 고유하게 인식하기 위한 ID 설정을 위해 **SRPN 영역의 값을 0xA 으로 설정 (임의의 값)**
   - Interrupt 를 enable 하기 위해 **SRE 영역의 값을 "1" 으로 설정**
   - Interrupt 를 CPU0에서 처리하기 위해 **TOS 영역의 값을 "0" 으로 설정**

SCUERU0 레지스터 @ 0xF0038CD4

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

레지스터 설정에 의한 데이터 흐름

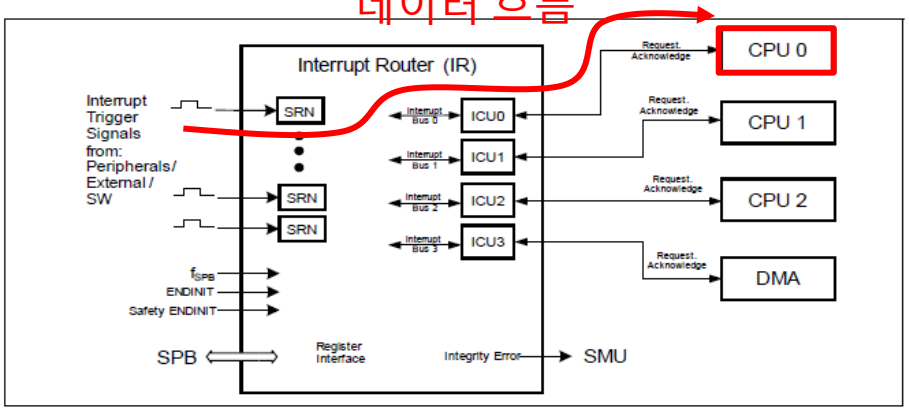


Figure 16-1 Block Diagram of the TC27x Interrupt System

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.1518

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.1608

External Interrupt 처리 Flow

: Interrupt Router (IR)

4. ERU의 출력이 연결되는 Interrupt Router (IR)의 역할
   - ERU 마지막 단계 OGU의 출력은 IR의 첫 번째 단계 **Service Request Node (SRN)**의 입력으로 연결
   - **SRN은 Interrupt Control Units (ICU)에 연결되어 Service Request Control (SRC) 레지스터 설정으로 각 CPU0 ~ 2에 연결**
   - ICU는 여러 개의 service request 간의 중재 역할도 수행 (priority 기반)

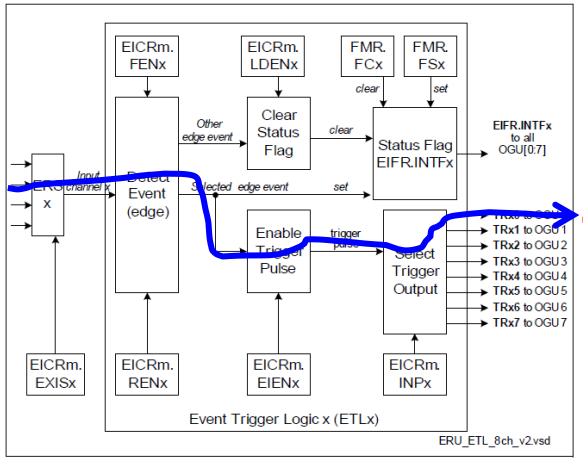
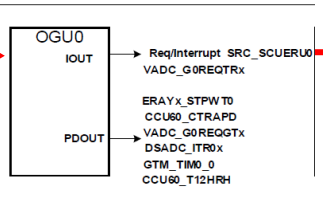


Figure 7-43 Event Trigger Logic Overview

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.623



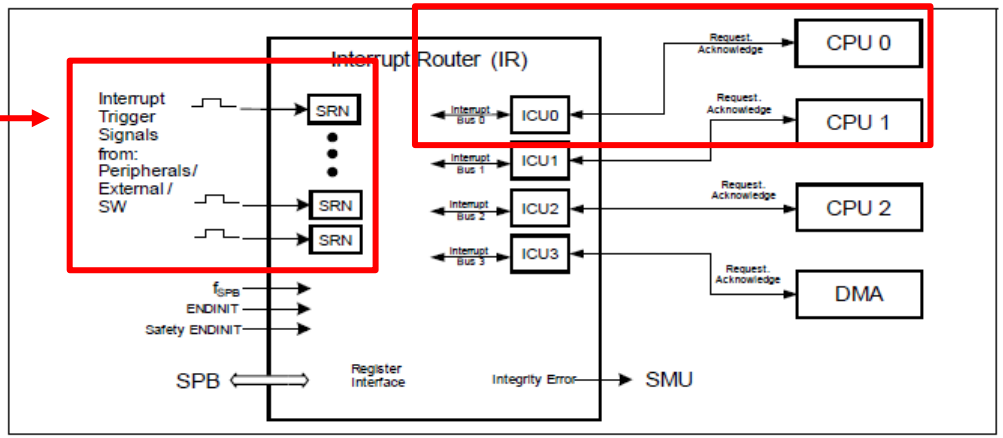


Figure 16-1 Block Diagram of the TC27x Interrupt System

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.1518

Lab5: SCU 레지스터 설정 – IGCR의 SRPN,TOS,SRE 설정

```c
// SRC registers
#define SRPN_BIT_LSB_IDX            0
#define TOS_BIT_LSB_IDX             11
#define SRE_BIT_LSB_IDX             10
```

```c
void initERU(void)
{
    // ERU setting
    SCU_EICR1.U &= ~(0x7 << EXIS0_BIT_LSB_IDX);
    SCU_EICR1.U |= (0x1  << EXIS0_BIT_LSB_IDX);

    SCU_EICR1.U |= 0x1   << FEN0_BIT_LSB_IDX;

    SCU_EICR1.U |= 0x1   << EIEN0_BIT_LSB_IDX;

    SCU_EICR1.U &= ~(0x7 << INP0_BIT_LSB_IDX);

    SCU_IGCR0.U &= ~(0x3 << IGP0_BIT_LSB_IDX);
    SCU_IGCR0.U |= 0x1   << IGP0_BIT_LSB_IDX;


    // SRC Interrupt setting
    SRC_SCU_SCU_ERU0.U &= ~(0xFF << SRPN_BIT_LSB_IDX);
    SRC_SCU_SCU_ERU0.U |= 0x0A   << SRPN_BIT_LSB_IDX;

    SRC_SCU_SCU_ERU0.U &= ~(0x3  << TOS_BIT_LSB_IDX);

    SRC_SCU_SCU_ERU0.U |= 1      << SRE_BIT_LSB_IDX;
}
```

Shift Offset 값 Define

- 레지스터에 값을 write할 때 shift되는 offset을 쉽게 사용하기 위한 define 작성

```
// Port registers
#define PC1_BIT_LSB_IDX          11
#define PC2_BIT_LSB_IDX          19
#define P1_BIT_LSB_IDX           1
#define P2_BIT_LSB_IDX           2

// SCU registers
#define LCK_BIT_LSB_IDX          1
#define ENDINIT_BIT_LSB_IDX      0
#define EXIS0_BIT_LSB_IDX        4
#define FEN0_BIT_LSB_IDX         8
#define EIEN0_BIT_LSB_IDX        11
#define INP0_BIT_LSB_IDX         12
#define IGP0_BIT_LSB_IDX         14

// SRC registers
#define SRPN_BIT_LSB_IDX         0
#define TOS_BIT_LSB_IDX          11
#define SRE_BIT_LSB_IDX          10
```

SCU 레지스터 bit shift offset

SRC 레지스터 bit shift offset

각 레지스터의 선언부 확인 (주소 및 비트 Slice)

- ERU Interrupt 사용을 위한 초기화 함수 *initERU()*

AURIX에서 제공하는 헤더파일의 레지스터 주소 확인
- 레지스터 이름 ctrl + left click

```c
void initERU(void)
{
    // ERU setting
    SCU_EICR1.U &= ~(0x7 << EXIS0_BIT_LSB_IDX);
    SCU_EICR1.U |= (0x1  << EXIS0_BIT_LSB_IDX);

    SCU_EICR1.U |= 0x1   << FEN0_BIT_LSB_IDX;

    SCU_EICR1.U |= 0x1   << EIEN0_BIT_LSB_IDX;

    SCU_EICR1.U &= ~(0x7 << INP0_BIT_LSB_IDX);

    SCU_IGCR0.U &= ~(0x3 << IGP0_BIT_LSB_IDX);
    SCU_IGCR0.U |= 0x1   << IGP0_BIT_LSB_IDX;


    // SRC Interrupt setting
    SRC_SCU_SCU_ERU0.U &= ~(0xFF << SRPN_BIT_LSB_IDX);
    SRC_SCU_SCU_ERU0.U |= 0x0A   << SRPN_BIT_LSB_IDX;

    SRC_SCU_SCU_ERU0.U &= ~(0x3  << TOS_BIT_LSB_IDX);

    SRC_SCU_SCU_ERU0.U |= 1      << SRE_BIT_LSB_IDX;
}
```

- SCU_EICR1

```c
/** \brief  214, External Input Channel Register */
#define SCU_EICR1 /*lint --e(923)*/ (*(volatile Ifx_SCU_EICR*)0xF0036214u)

/** \brief  218, External Input Channel Register */
```

- SCU_IGCR0

```c
/** \brief  22C, Flag Gating Register */
#define SCU_IGCR0 /*lint --e(923)*/ (*(volatile Ifx_SCU_IGCR*)0xF003622Cu)

/** \brief  230, Flag Gating Register */
```

- SRC_SCUERU0

```c
/** \brief  CD4, SCU ERU Service Request */
#define SRC_SCU_SCU_ERU0 /*lint --e(923)*/ (*(volatile Ifx_SRC_SRCR*)0xF0038CD4u)

/** Alias (User Manual Name) for SRC_SCU_SCU_ERU0.
* To use register names with standard convension, please use SRC_SCU_SCU_ERU0
```

전체적 코드 흐름

- ERU Interrupt 사용을 위한 초기화 함수 *initERU()*

```c
void initERU(void)
{
    // ERU setting
    SCU_EICR1.U &= ~(0x7 << EXIS0_BIT_LSB_IDX);
    SCU_EICR1.U |= (0x1  << EXIS0_BIT_LSB_IDX);

    SCU_EICR1.U |= 0x1   << FEN0_BIT_LSB_IDX;

    SCU_EICR1.U |= 0x1   << EIEN0_BIT_LSB_IDX;

    SCU_EICR1.U &= ~(0x7 << INP0_BIT_LSB_IDX);

    SCU_IGCR0.U &= ~(0x3 << IGP0_BIT_LSB_IDX);
    SCU_IGCR0.U |= 0x1   << IGP0_BIT_LSB_IDX;


    // SRC Interrupt setting
    SRC_SCU_SCU_ERU0.U &= ~(0xFF << SRPN_BIT_LSB_IDX);
    SRC_SCU_SCU_ERU0.U |= 0x0A   << SRPN_BIT_LSB_IDX;

    SRC_SCU_SCU_ERU0.U &= ~(0x3  << TOS_BIT_LSB_IDX);

    SRC_SCU_SCU_ERU0.U |= 1      << SRE_BIT_LSB_IDX;
}
```

ERU 설정
- ERU Channel 2의 입력으로 Input 1 사용
- Falling 엣지가 Trigger 신호 생성하도록 하고 이를 enable
- 생성된 Trigger 신호를 Output Channel 0 으로 전달

SRC Interrupt 설정
- Interrupt의 우선순위 설정
- Interrupt의 처리를 CPU0이 맡도록 설정
- Interrupt가 발생하도록 enable

인터럽트 발생시 호출할 ISR Function 정의

- Interrupt 발생 시 호출되는 ISR 함수 작성

```
__interrupt(0x0A) __vector_table(0)
void ERU0_ISR(void)
{
    P10_OUT.U ^= 0x1 << P1_BIT_LSB_IDX;  // toggle P10.1 (LED D12 RED)
}
```

→ **0x0A 고유 Interrupt ID를 가지며, CPU0에서 처리됨**

→ P10.1 핀의 출력 toggle

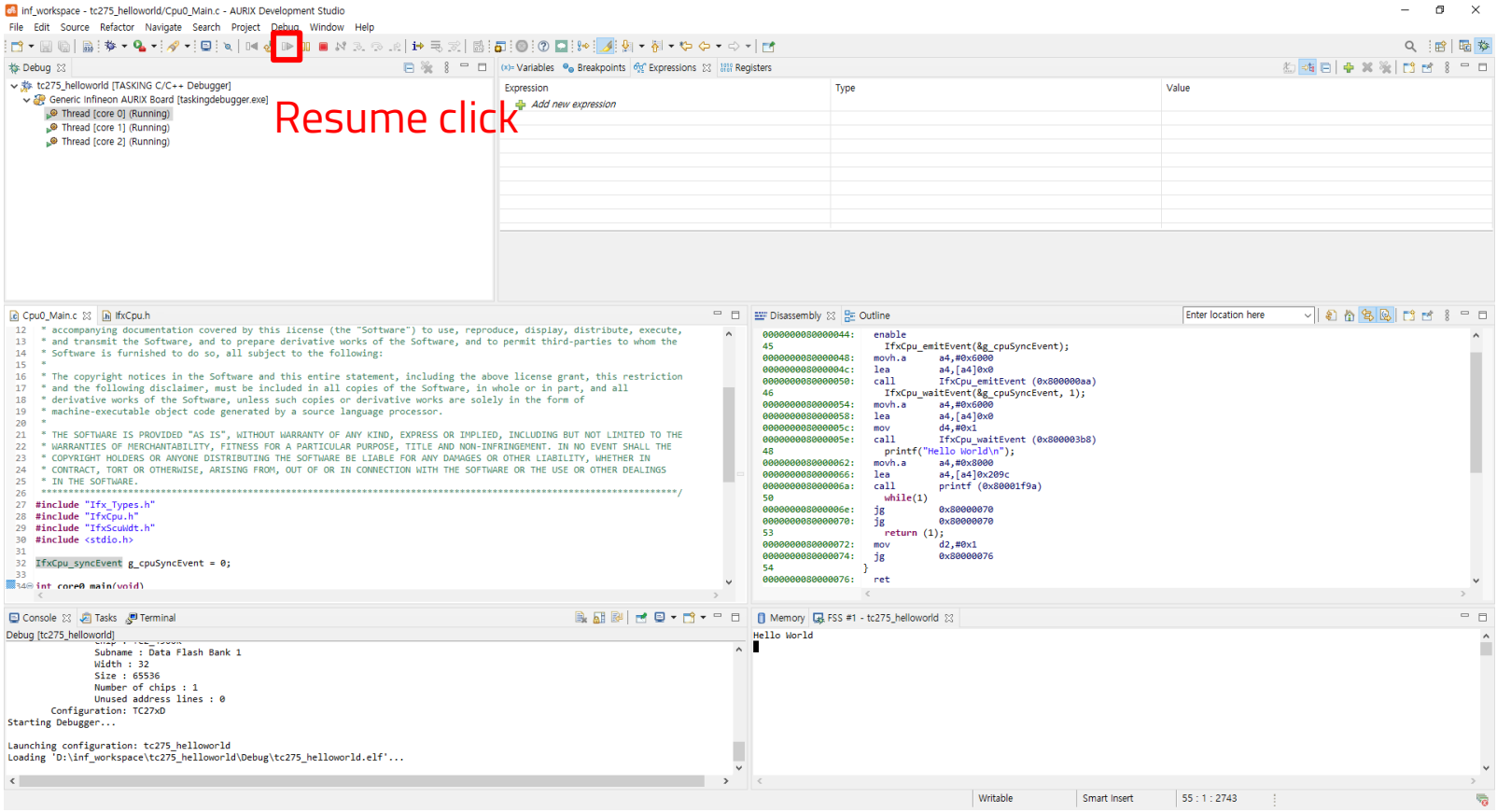
- 이제 main 함수 while 내에는 아무 코드도 필요하지 않음

```
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);

    initERU();
    initLED();
    initButton();


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

→ 초기화 함수 호출

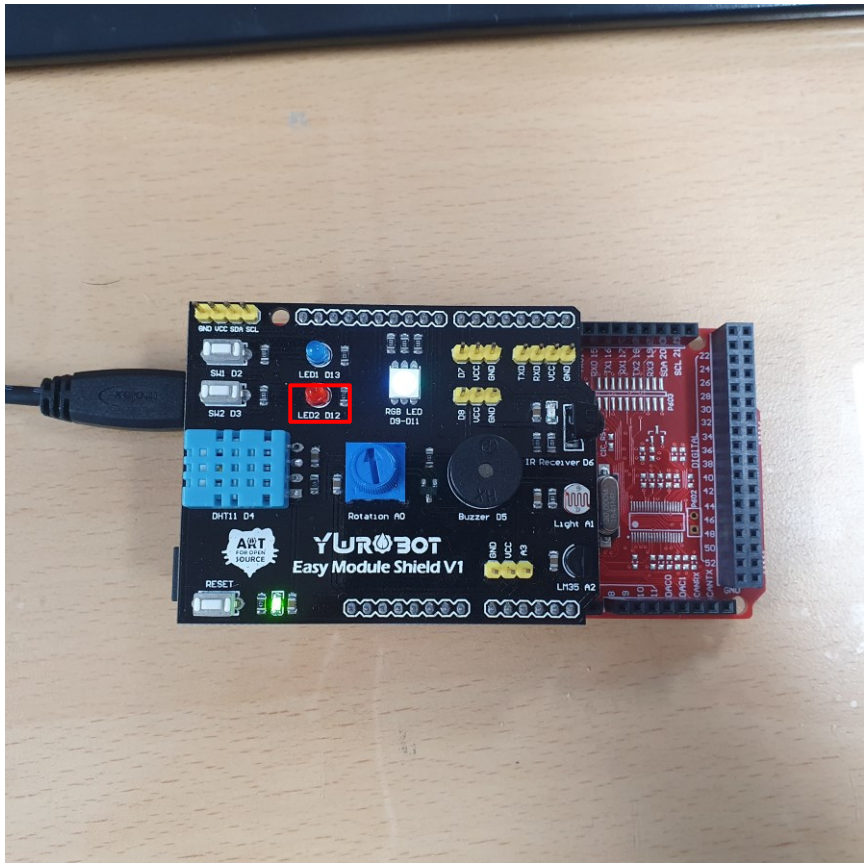
Build 및 Debug

- 프로젝트 빌드 (ctrl + b)
- 디버그 수행하여 보드에 실행 파일 flash

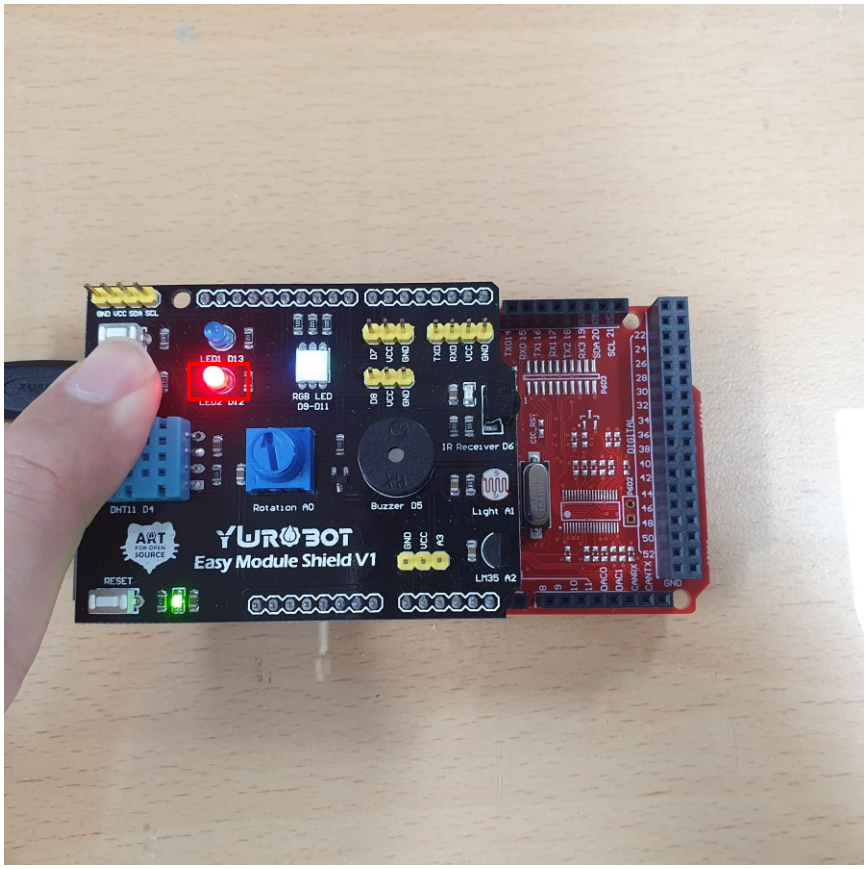


동작 확인

- Push 버튼 (SW2)를 누르면 LED RED가 toggle 하는 모습 확인 가능

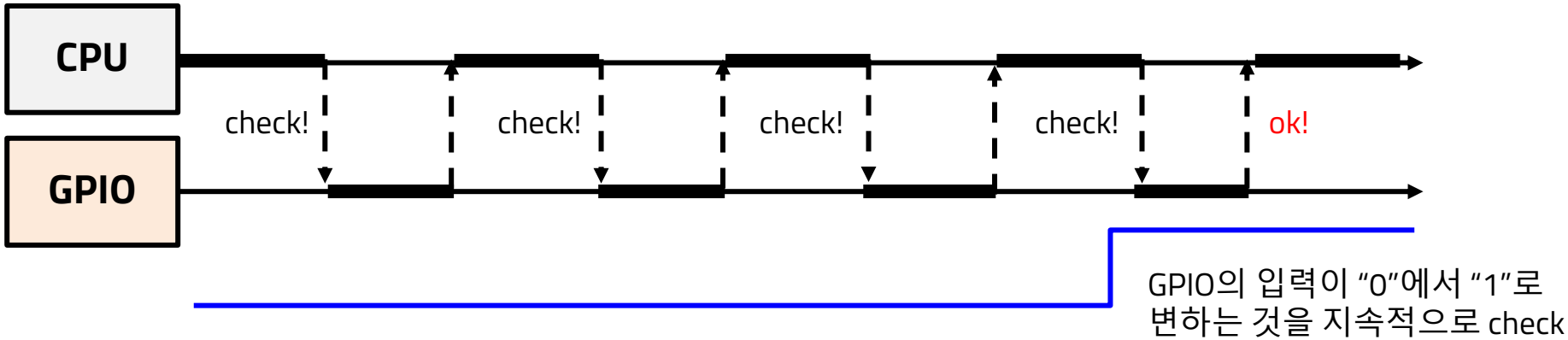


Switch Click!

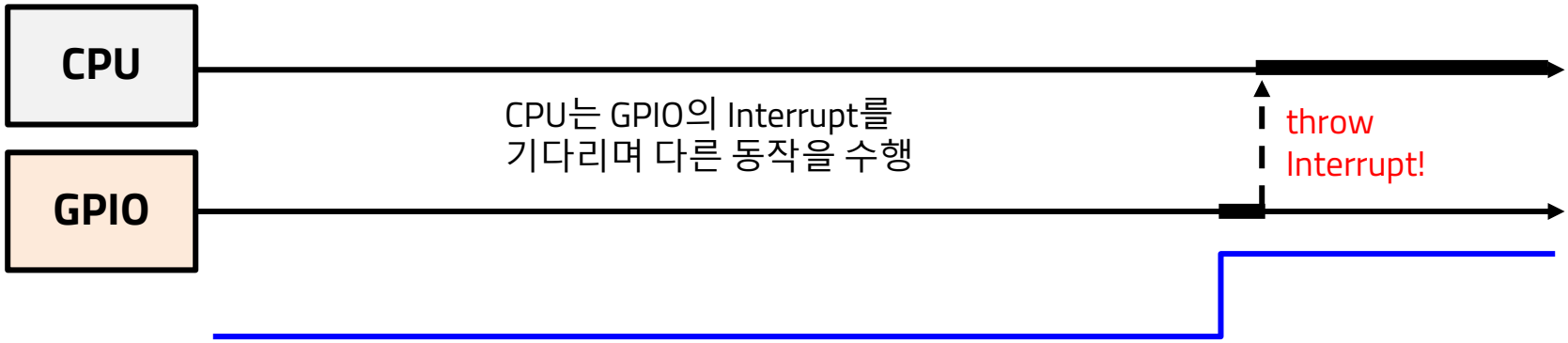


인트럽트 방식으로 구현시 장점 생각해보자!!

- 지금까지 Push 버튼이 눌렸는지 확인하기 위해 사용한 **Polling 기반 방법**
  - CPU가 항상 GPIO를 신경 쓰고 있어야 함



- Push 버튼이 눌렸을 때 **Interrupt가 발생하도록 하는 방법**
  - GPIO 모니터링 필요없이 CPU가 다른 일을 하거나 잠잘수 있음 (IDLE)



인터럽트 코드가 아닌 Polling 방식이 된다면~..

```
64
65 int core0_main(void)
66 {
67     IfxCpu_enableInterrupts();
68
69     /* !!WATCHDOG0 AND SAFETY WATCHDOG ARE DISABLED HERE!!
70      * Enable the watchdogs and service them periodically if it is required
71      */
72     IfxScuWdt_disableCpuWatchdog(IfxScuWdt_getCpuWatchdogPassword());
73     IfxScuWdt_disableSafetyWatchdog(IfxScuWdt_getSafetyWatchdogPassword());
74
75     /* Wait for CPU sync event */
76     IfxCpu_emitEvent(&g_cpuSyncEvent);
77     IfxCpu_waitEvent(&g_cpuSyncEvent, 1);
78
79     initERU();
80     initLED();
81     initButton();
82
83
84     while(1)
85     {
86     }
87     return (1);
88 }
```

보충Lab1: 버튼 눌렀을때 LED 한번 깜빡이기(펄스)

```
__interrupt(0x0A) __vector_table(0)
void ERU0_ISR(void)
{
    P10_OUT.U ^= 0x1 << P1_BIT_LSB_IDX;  // set P10.1 (LED D12 RED)
    for(unsigned int i = 0; i < 300000; i++);
    P10_OUT.U &= ~(0x1 << P1_BIT_LSB_IDX);  // clear P10.1 (LED D12 RED)
}
```

보충Lab2: 버튼 눌렀다 땔 때 반응하도록

```
void initERU(void)
{
    // ERU setting
    SCU_EICR1.U &= ~(0x7 << EXIS0_BIT_LSB_IDX);
    SCU_EICR1.U |= (0x1  << EXIS0_BIT_LSB_IDX);

    SCU_EICR1.U |= 0x1   << FEN0_BIT_LSB_IDX;

    /****/
    SCU_EICR1.U |= 0x1   << REN0_BIT_LSB_IDX;   // rising edge
    /****/

    SCU_EICR1.U |= 0x1   << EIEN0_BIT_LSB_IDX;
```

보충Lab3: 버튼을 누르고 한참 있으면 LED on

```c
__interrupt(0x0A) __vector_table(0)
void ERU0_ISR(void)
{
    unsigned char a, b, c;

    a = P02_IN.U & (0x1 << P1_BIT_LSB_IDX);
    for(unsigned int i = 0; i < 10000000; i++);

    b = P02_IN.U & (0x1 << P1_BIT_LSB_IDX);
    for(unsigned int i = 0; i < 10000000; i++);

    c = P02_IN.U & (0x1 << P1_BIT_LSB_IDX);
    for(unsigned int i = 0; i < 10000000; i++);

    if( a == 0 && b == 0 && c == 0 )
    {
        P10_OUT.U |= (0x1 << P1_BIT_LSB_IDX);        // set P10.1 (LED D13 RED)
        for(unsigned int i = 0; i < 500000; i++);
        P10_OUT.U &= ~(0x1 << P1_BIT_LSB_IDX);        // clear P10.1 (LED D13 RED)
    }

}
```

감사합니다. 휴식~~