임베디드 MCU 프로그래밍 실습

Internal Interrupt-Driven Hardware-Software Interaction

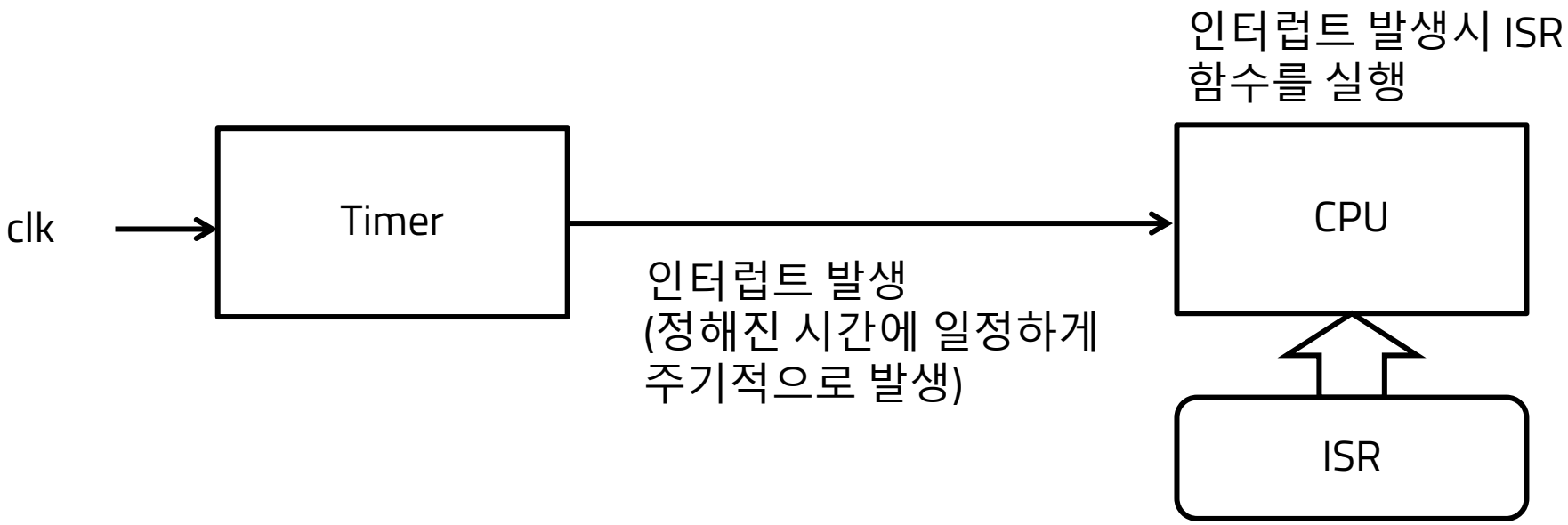
Timer Interrupt-Driven Periodic Software Execution for TC275 Infineon Embedded Processors

현대자동차 입문교육
박대진 교수

실습 목표

보드의 **CCU6** 하드웨어에서 발생하는 **Interrupt**를 사용해서 일정한 시간 간격(정확한) **(0.5초)** 으로 **BLUE LED**를 **toggle**하는 **ISR** 함수가 호출되도록 한다.

1. 타이머 회로 설정하여 **0.5**초마다 인터럽트 발생시키도록 설정
2. 발생된 인터럽트를 **CPU0**로 배송되도록 인터럽트 회로 설정 **(**어제 한 내용 상기**)**



사용된 약어 정리

- CCU6 (Capture / Compare Unit 6)
- IR (Interrupt Router)
- SRN (Service Request Node)
- SCU (System Control Unit)
- PM (Period Match)

CCU6 하드웨어 동작 원리

: Counter Match Interrupt 발생시키기

- 정확한 시간을 계산해야 될 필요성 존재 , 그것을 실현시키는 회로가 타이머.

→ **CCU6 (Capture / Compare Unit 6) 하드웨어 사용 – CCU6 은 고유명사!**

- MCU 내부에 매 clock 마다 증가하는 counter 존재, counter의 값이 원하는 목표치에 도달했는지 비교하는 회로가 필요함
- **Counter 값이 원하는 값에 도달하면 Interrupt 발생**

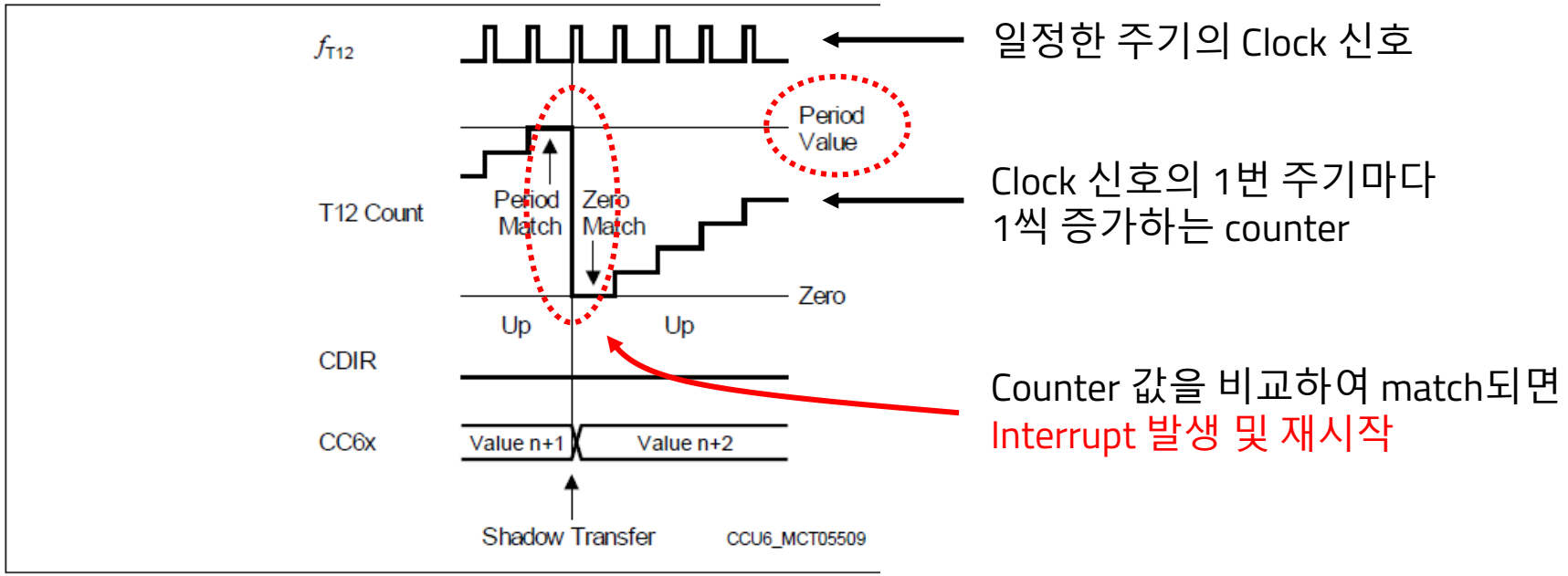


Figure 26-5 T12 Operation in Edge-Aligned Mode

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.3648

CCU6 하드웨어 동작 원리

: CCU6 내부 T12 모듈

CCU6 기능 중, 본 실습에서 사용할 **T12 모듈**
(T12 는 고유명사!)

CCU6 하드웨어에 입력되는 clock 신호 → 시간 계산의 기준이 됨 = **50 MHz**

Interrupt 발생시키는 역할

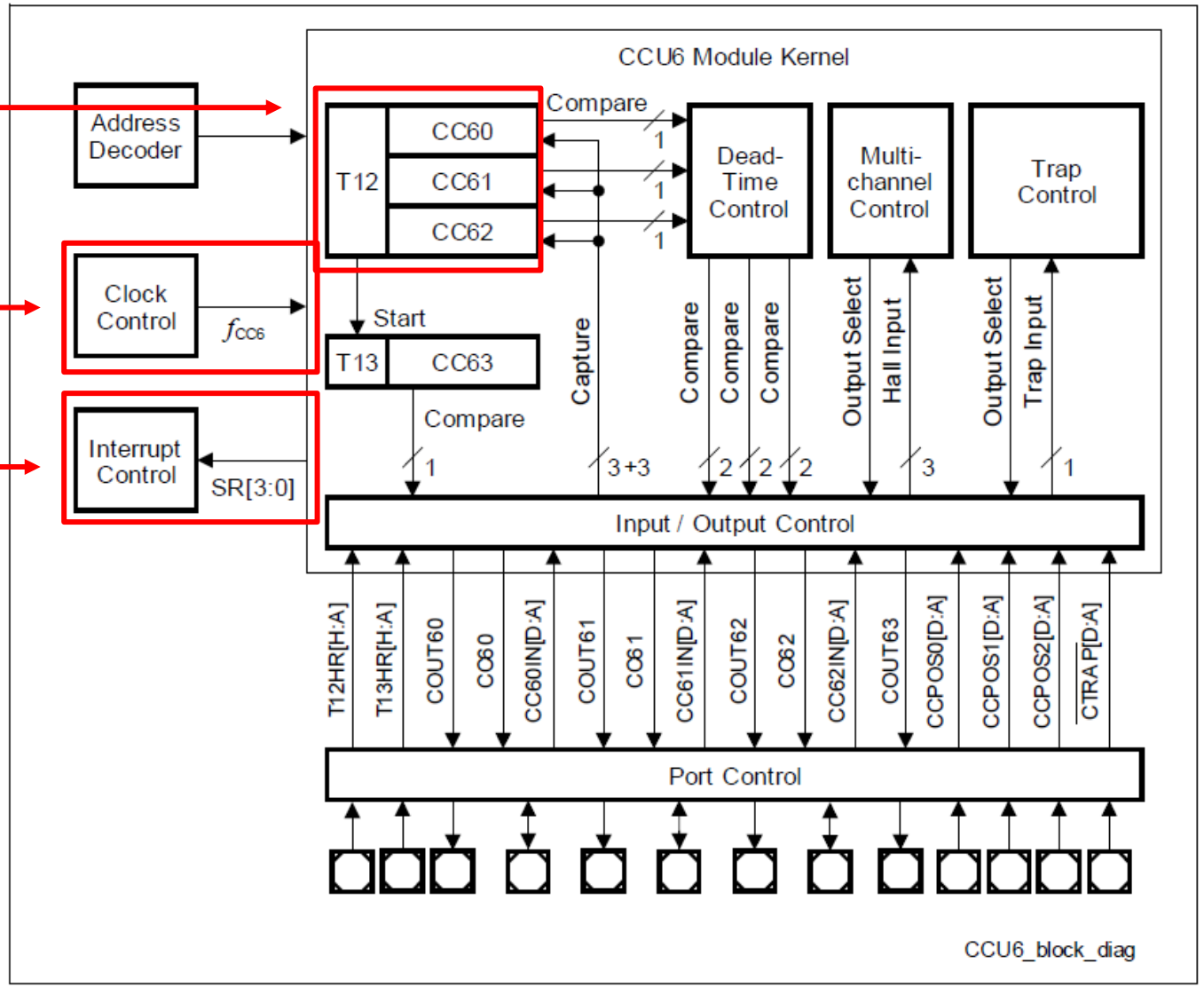


Figure 26-1 CCU6 Block Diagram

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.3639

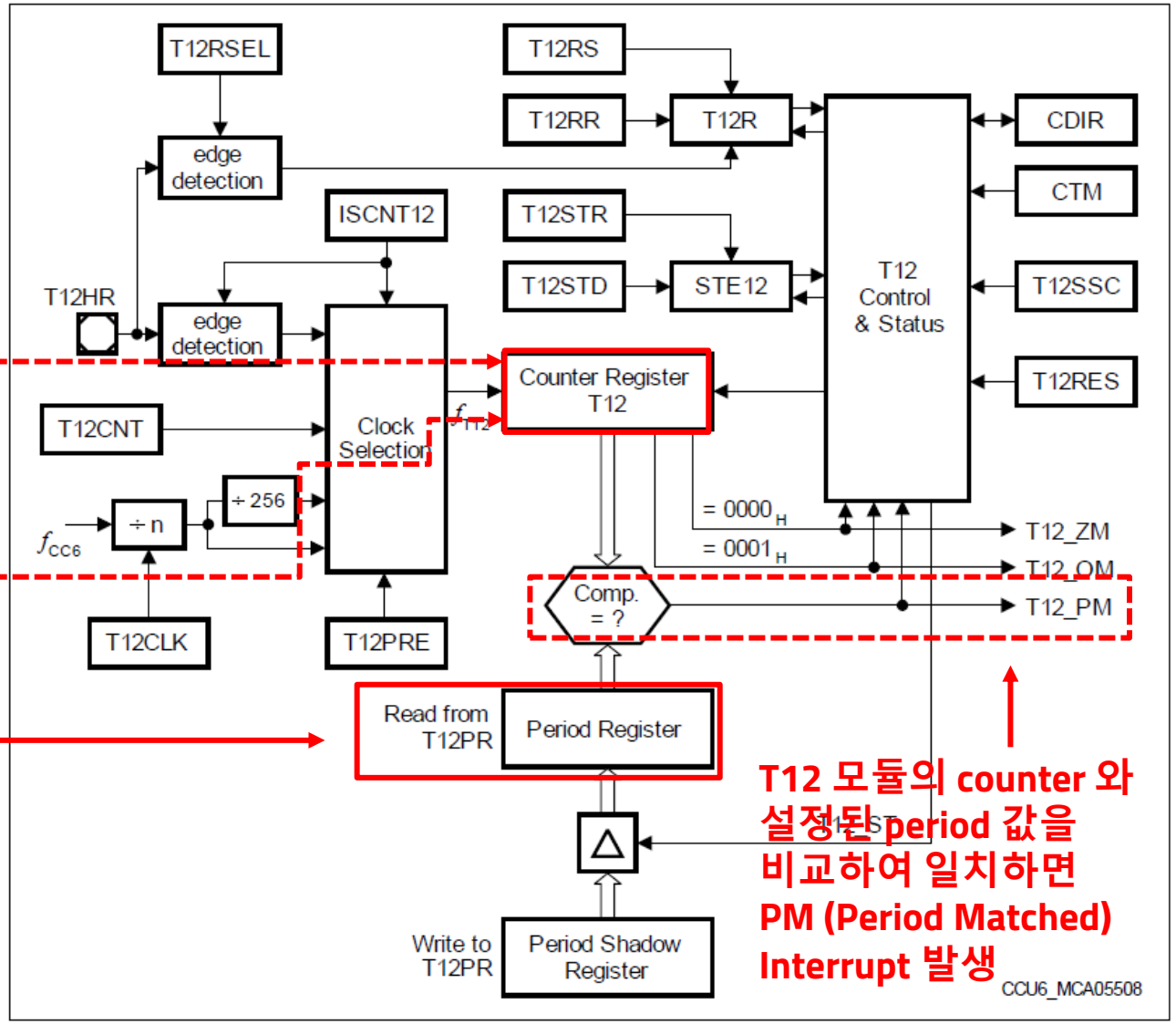
CCU6 내부 T12 하드웨어 동작 원리

: T12 모듈 → Counter 구동 및 목표 설정값 도달여부 비교

T12 모듈의 counter 값

CCU6 으로 입력되는 clock 신호를 분주하여 T12 에서 counter 증가에 사용될 clock 주파수를 결정

T12 모듈의 counter 와 비교할 목표값 period



T12 모듈의 counter 와 설정된 period 값을 비교하여 일치하면 PM (Period Matched) Interrupt 발생

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.3645

Figure 26-4 Timer T12 Logic and Period Comparators

T12에서 생성되는 Interrupt 설정
: 인터럽트 Enable, 인터럽트 출력 포트 설정

- T12에서 counter 가 증가하여 period 레지스터의 값과 match 되었을 때 발생하는 PM (Period Matched) Interrupt 를 사용하기 위한 설정

1. **T12_PM** Interrupt Enable
2. 발생한 Interrupt를 **SR0** 출력으로 전달

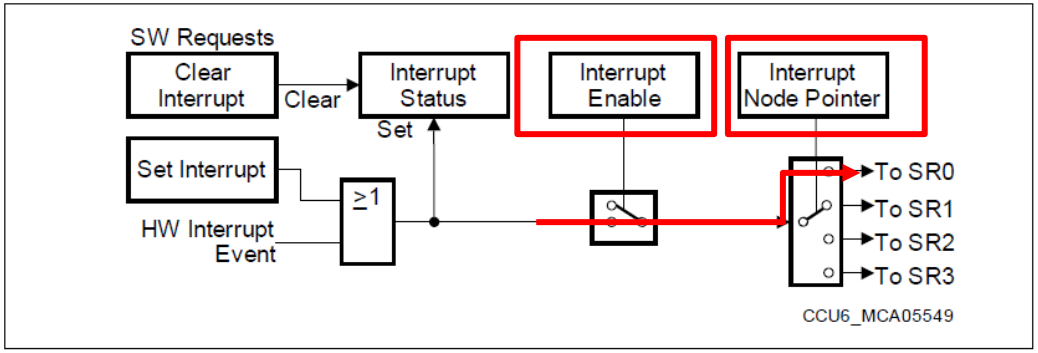
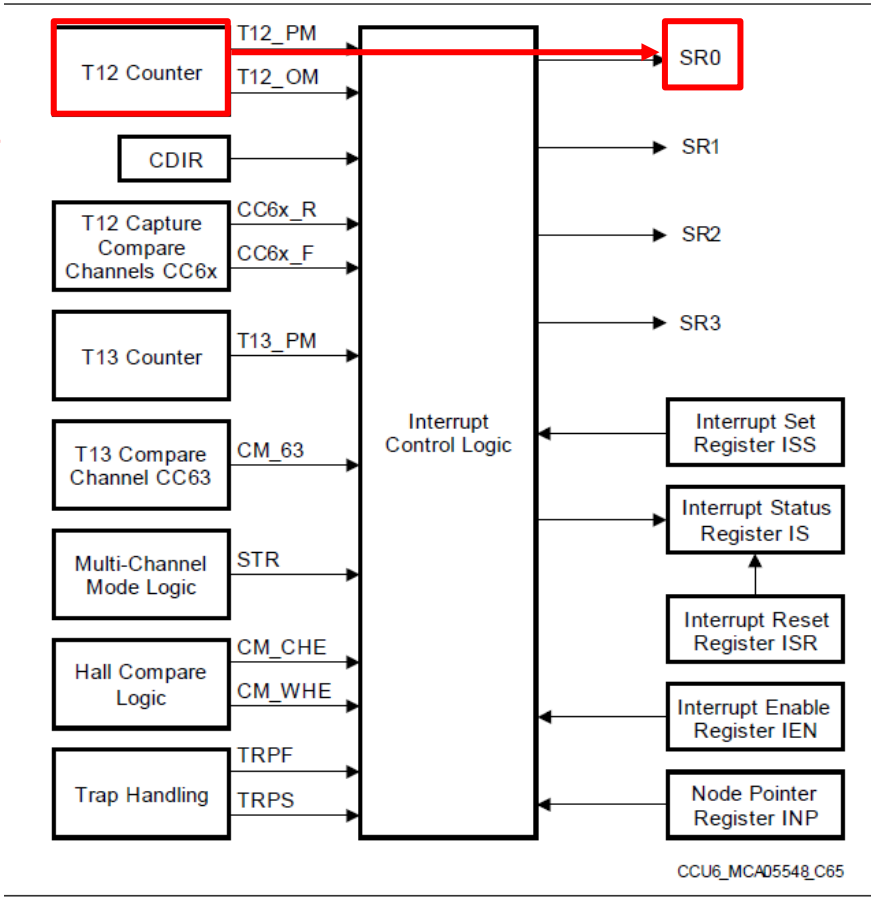


Figure 26-42 General Interrupt Structure

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.3732



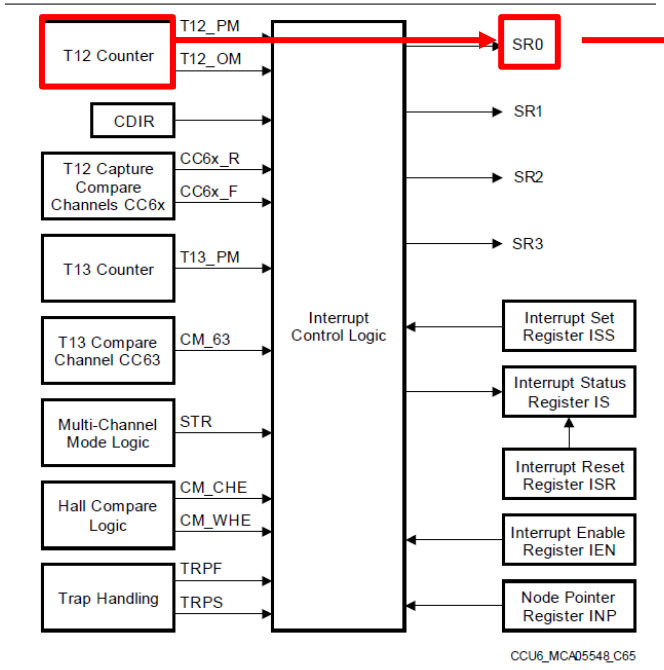
Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.3733

6-43 Interrupt Sources and Events

T12에서 생성되는 Interrupt를 IR로 연결

- CCU6 모듈 내부 T12에서 생성되는 Interrupt를 IR의 입력(SRN)으로 연결
  - **SRN 레지스터를 설정하여 Interrupt를 CPU0으로 전달**

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.3733



3-43 Interrupt Sources and Events

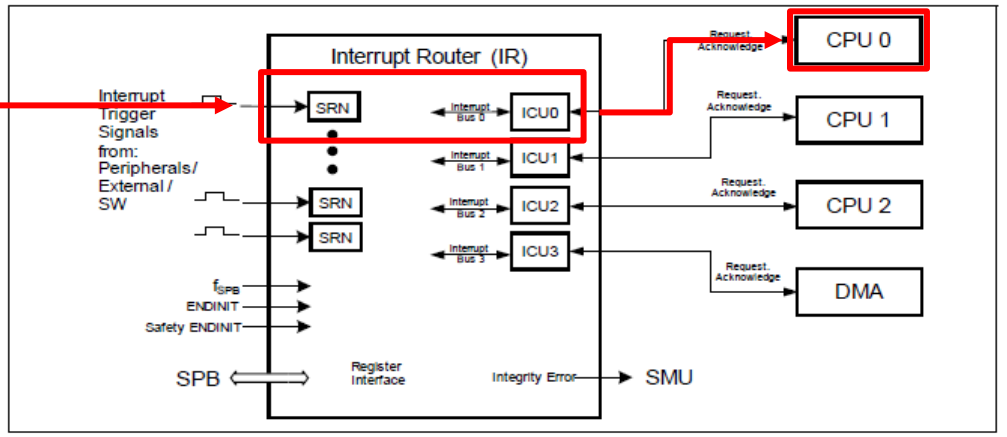


Figure 16-1 Block Diagram of the TC27x Interrupt System

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

0번째 CCU6 모듈 사용 위한 레지스터 설정
→ CCU60 모듈의 Clock 입력 설정

- CCU6 는 0 – 1 까지 2개의 모듈 존재, 그중 CCU60
- CCU60 위한 레지스터 중에 **CLC** 레지스터
  - Clock Control enable 필요함
- **CCU60_CLC 레지스터는 System Critical 레지스터이므로 CPU ENDINIT 해제 후 수정가능**
- CPU ENDINIT 비트 clear위해 SCU 레지스터 항목에서 **WDTCPU0CON0** 레지스터 설정 필요

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.3639

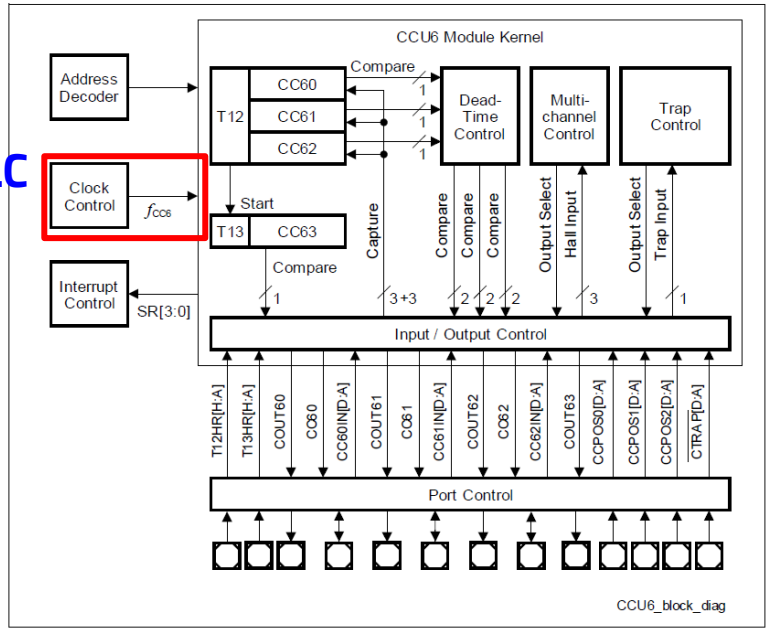


Figure 26-1 CCU6 Block Diagram

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.650

- "CE0"- writeable only when CPU0 ENDINIT bit is zero
- "CE1" - writeable only when CPU1 ENDINIT bit is zero
- "CE2" - writeable only when CPU2 ENDINIT bit is zero
- "E" - writeable when any (one or more) CPUx ENDINIT bit is zero
- "SE" - writeable only when Safety ENDINIT bit is zero
- None of the above - accessible at any time

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.3764

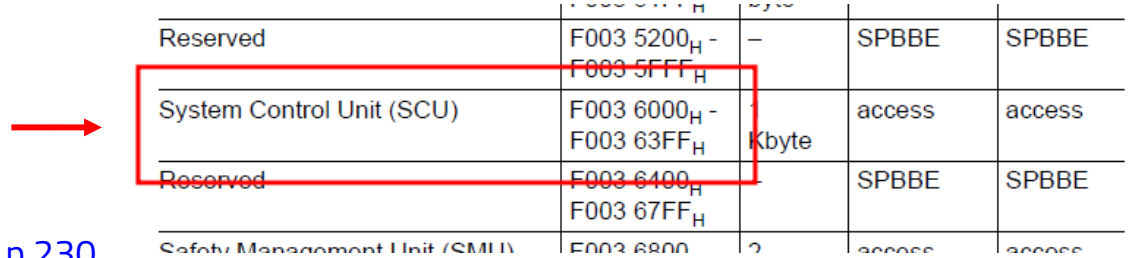
Table 26-14 Registers Overview - BPI Registers

Register Short Name	Description	Offset Addr.	Access Mode		Reset	Page Num.
			Read	Write		
CLC	Clock Control Register	00_H	U, SV	SV, E, P	Application Reset	26-130
Kernel Registers						

Lab: SCU 레지스터 설정 – WDTCPU0CON0

1. SCU 레지스터 영역의 주소 찾기
   - 시작 주소 (Base address)
   - **= 0xF0036000**

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



Reserved	F003 5200$_H$ - F003 5FFF$_H$	–	SPBBE	SPBBE
System Control Unit (SCU)	F003 6000$_H$ - F003 63FF$_H$	1 Kbyte	access	access
Reserved	F003 6400$_H$ F003 67FF$_H$	–	SPBBE	SPBBE
Safety Management Unit (SMU)	F003 6800	2	access	access

2. 사용할 레지스터의 주소 찾기
   - **WDTCPU0CON0** 의 Offset Address = **0x100**

→ WDTCPU0CON0 레지스터 주소 = 0xF0036000 + 0x100 = **0xF0036100**

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		
EMSR	Emergency Stop Register	0FC$_H$	U, SV	SV, SE, P	Application Reset	Page 7-291
WDTCPU0CON0	CPU0 WDT Control Register 0	100$_H$	U, SV	U, SV, 32,(CPU 0[2)])	Application Reset	Page 7-276
WDTCPU0CON1	CPU0 WDT Control	104$_H$	U, SV	SV,	Application	Page

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.697

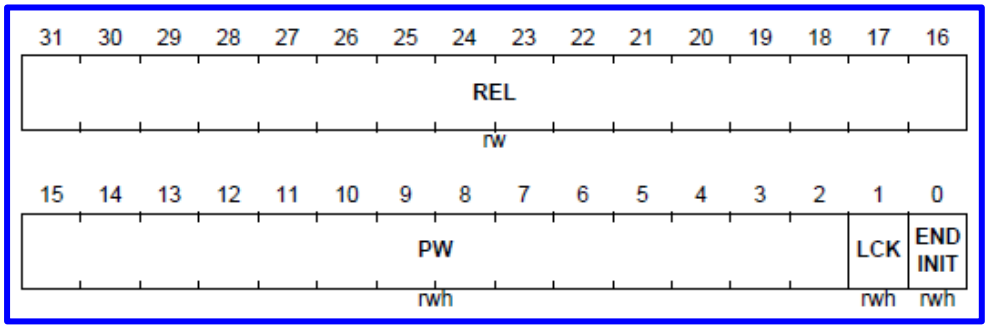
SCU 레지스터 설정 – WDTCPU0CON0

: WDTCPU0CON0 변경위해서 보호 레지스터 잠금 해제 필요함

3. **Password Access** 를 통해 WDTCPU0CON0 레지스터의 Lock 상태를 해제해야 함

1) WDTCPU0CON0 레지스터를 읽어 **WDTCPU0CON0.REL, WDTCPU0CON0.PW** 영역의 값을 확인

WDTCPU0CON0 레지스터 @ 0xF0036100



Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.661

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.662

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.

단, PW 영역의 일부 PW[7:2] 는 반전해서 읽어야 함

2) Write 할 값은 1)에서 읽은 REL 과 PW 의 조합으로 결정

[31:16] = REL, [15:2] = PW, [1] = “0”, [0] = “1”

3) 결정한 32bit 값을 WDTCPU0CON0 레지스터에 한 번에 write

4) WDTCPU0CON0 레지스터의 1번째 bit LCK 를 읽어서 Lock 상태가 해제되었는지 확인

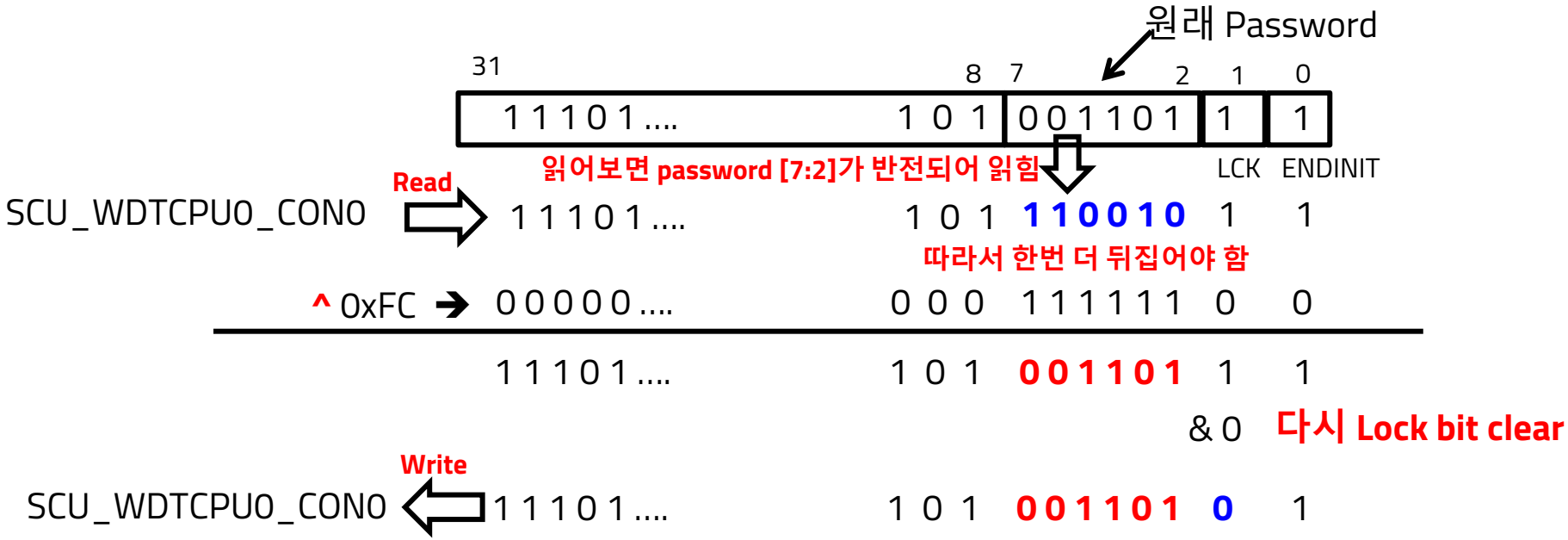
Lock 상태가 해제되면 LCK bit 가 0으로 읽힘

Lab1: SCU 레지스터 설정 – WDTCPU0CON0

보호 레지스터 잠금 해제

```
39 // SCU registers
40 #define LCK_BIT_LSB_IDX            1
41 #define ENDINIT_BIT_LSB_IDX        0
```

```
150 void initCCU60(void)
151 {
152     // Password Access to unlock SCU_WDTSCON0
153     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
154     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0);    // wait until unlocked
155
156     // Modify Access to clear ENDINIT
157     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) & ~(1 << ENDINIT_BIT_LSB_IDX);
158     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0);    // wait until locked
```



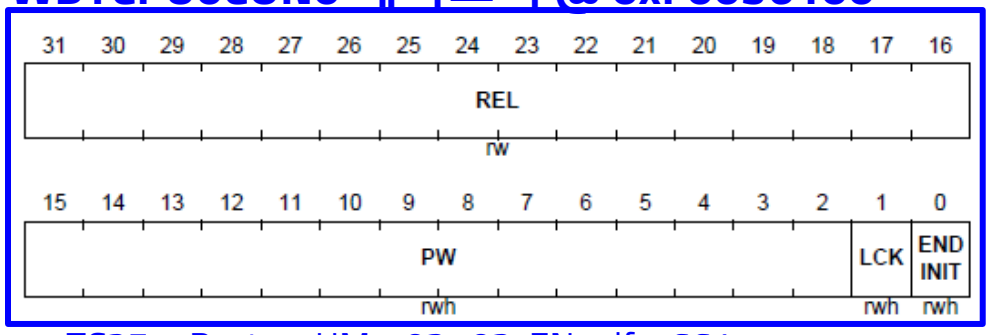
SCU 레지스터 설정 – WDTCPU0CON0

: Lock해제 후 ENDINIT 값을 바꾼 뒤 바로 잠금 설정

4. Modify Access 를 통해 WDTCPU0CON0 레지스터의 CPU0 ENDINIT을 set/clear

1) WDTCPU0CON0 레지스터를 읽어 **WDTCPU0CON0.REL, WDTCPU0CON0.PW** 영역의 값을 확인

WDTCPU0CON0 레지스터 @ 0xF0036100



Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.661

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.662

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.

단, PW 영역의 일부 PW[7:2] 는 반전해서 읽어야 함

2) Write 할 값은 1)에서 읽은 REL 과 PW 의 조합으로 결정

[31:16] = REL, [15:2] = PW, [1] = "1"

3) 0번째 bit **ENDINIT을 설정하려면 [0] = "1", 해제하려면 "0" 값을 write**

➔ **변경함과 동시에 바로 Lock시킴**

4) 결정한 32bit 값을 WDTCPU0CON0 레지스터에 한 번에 write

5) WDTCPU0CON0 레지스터의 1번째 bit LCK 를 읽어서 Lock 상태가 설정되었는지 확인

Lock 상태가 설정되면 LCK bit 가 1로 읽힘

Lab2: SCU 레지스터 설정 – WDTCPU0CON0

Lock해제 후 ENDINT clear후 다시 잠금

```
39 // SCU registers
40 #define LCK_BIT_LSB_IDX            1
41 #define ENDINIT_BIT_LSB_IDX        0
```

```
150 void initCCU60(void)
151 {
152     // Password Access to unlock SCU_WDTSCON0
153     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
154     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0);    // wait until unlocked
155
156     // Modify Access to clear ENDINIT
157     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) & ~(1 << ENDINIT_BIT_LSB_IDX);
158     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0);    // wait until locked
159
160     CCU60_CLC.U &= ~(1 << DISR_BIT_LSB_IDX);    // enable CCY
161
162     // Password Access to unlock SCU_WDTSCON0
163     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
164     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0);    // wait until unlocked
165
166     // Modify Access to set ENDINIT
167     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
168     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0);    // wait until locked
169
```

ENDINIT값이 0이므로 보호 레지스터에 값을 쓸 수 있음
→ CLC 값을 변경가능
→ 이후에 다시 ENDINIT값을 1로 셋팅하기 위해 보호 레지스터 Lock풀고,
ENDINIT값을 변경한뒤 다시 Lock

CCU60 레지스터 설정 – CLC

ENDINIT가 0이 되었으므로 보호 레지스터 CLC 변경가능

1. CCU60 레지스터 영역의 주소 찾기
   - 시작 주소 (Base address)
   - **= 0xF0002A00**

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.228

(MSC1)	$F000\ 27FF_H$	byte		
Reserved	$F000\ 2800_H$ - $F000\ 29FF_H$	–	SPBBE	SPBBE
Capture/Compare Unit 6 0(CCU60)	$F000\ 2A00_H$ - $F000\ 2AFF_H$	256 byte	access	access
Capture/Compare Unit 6 1 (CCU61)	$F000\ 2B00_H$ - $F000\ 2BFF_H$	256 byte	access	access

2. 사용할 레지스터의 주소 찾기
   - **CLC** 의 Offset Address = **0x00**

   → CLC 레지스터 주소 = 0xF0002A00 + 0x00 = **0xF0002A00**

Table 26-14 Registers Overview - BPI Registers

Register Short Name	Description	Offset Addr.	Access Mode		Reset	Page Num.
			Read	Write		
CLC	Clock Control Register	00_H	U, SV	SV, E, P	Application Reset	26-130
-	Kernel Registers	-	-	-	-	-

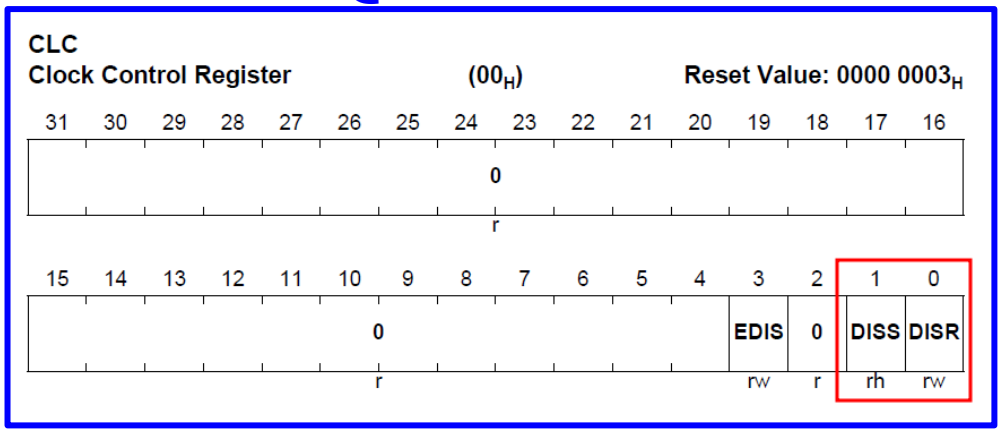
Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.3764

CCU60 레지스터 설정 – CLC

: Clock enable

3. 레지스터 write 값 결정
   - CCU60 모듈을 위한 clock을 enable 하기 위해 **DISR 영역에 0x0 write**
   - CCU60 모듈의 **DISS 영역의 값을 읽었을 때 "0"이면 모듈이 enable 되었음**
     - 만약 읽었을 때 **"1"이라면 모듈이 disable 되어있다는 뜻**

CLC 레지스터 @ 0xF0002A00



Field	Bits	Type	Description
DISR	0	rw	**Module Disable Request Bit** Used for enable/disable control of the module. 0_B Module disable is not requested. 1_B Module disable is requested.
DISS	1	rh	**Module Disable Status Bit** Bit indicates the current status of the module. 0_B Module is enabled. 1_B Module is disabled.

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.3766

Lab3: CCU60 레지스터 설정 – CLC

: CCU60 Clock Enable 요청

헤더 파일 참조 추가

```
30
31 #include "IfxCcu6_reg.h"
```

```
53 // CCU60 registers
54 #define DISS_BIT_LSB_IDX            1
55 #define DISR_BIT_LSB_IDX            0
```

```
150 void initCCU60(void)
151 {
152     // Password Access to unlock SCU_WDTSCON0
153     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
154     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0);    // wait until unlocked
155
156     // Modify Access to clear ENDINIT
157     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) & ~(1 << ENDINIT_BIT_LSB_IDX);
158     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0);    // wait until locked
159
160     CCU60_CLC.U &= ~(1 << DISR_BIT_LSB_IDX);    // enable CCY
161
162     // Password Access to unlock SCU_WDTSCON0
163     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
164     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0);    // wait until unlocked
165
166     // Modify Access to set ENDINIT
167     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
168     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0);    // wait until locked
169
```

ENDINIT값이 0이므로 보호 레지스터에 값을 쓸 수 있음
→ CLC 값을 변경가능
→ 이후에 다시 ENDINIT값을 1로 셋팅하기 위해 보호 레지스터 Lock풀고,
ENDINIT값을 변경한뒤 다시 Lock

Lab4: SCU 레지스터 설정 – WDTCPU0CON0

보호 레지스터 잠금 해제한 뒤 ENDINT set후 다시 잠금

```
150 void initCCU60(void)
151 {
152     // Password Access to unlock SCU_WDTSCON0
153     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
154     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0);    // wait until unlocked
155
156     // Modify Access to clear ENDINIT
157     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) & ~(1 << ENDINIT_BIT_LSB_IDX);
158     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0);    // wait until locked
159
160     CCU60_CLC.U &= ~(1 << DISR_BIT_LSB_IDX);    // enable CCY
161
162     // Password Access to unlock SCU_WDTSCON0
163     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
164     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0);    // wait until unlocked
165
166     // Modify Access to set ENDINIT
167     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
168     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0);    // wait until locked
169
```

5. Modify Access 를 통해 CPU0 ENDINIT을 해제한 뒤 보호 레지스터값 수정 후 반드시 CPU0 ENDINIT을 1로 다시 설정 해줘야 함
   (1) ENDINIT를 0으로 하면서 바로 Lock을 했기 때문에
   (2) 보호 레지스터 값 변경후
   (3) ENDINIT를 1로 하기 위해 다시 Password Access 방법이용 Lock 해제후
   (4) 다시 ENDINIT를 1로 하면서 동시에 Lock 시킴

Lab5: CCU60 레지스터 설정 – CLC

CCU6 모듈 enable 여부 확인

```
150 void initCCU60(void)
151 {
152     // Password Access to unlock SCU_WDTSCON0
153     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
154     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0);    // wait until unlocked
155
156     // Modify Access to clear ENDINIT
157     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) & ~(1 << ENDINIT_BIT_LSB_IDX);
158     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0);    // wait until locked
159
160     CCU60_CLC.U &= ~(1 << DISR_BIT_LSB_IDX);    // enable CCY
161
162     // Password Access to unlock SCU_WDTSCON0
163     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
164     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0);    // wait until unlocked
165
166     // Modify Access to set ENDINIT
167     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
168     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0);    // wait until locked
169
170
171     // CCU60 T12 configurations
172     while((CCU60_CLC.U & (1 << DISS_BIT_LSB_IDX)) != 0);// wait until CCU60 module enabled
173
174     CCU60_TCTR0.U &= ~(0x7 << T12CLK_BIT_LSB_IDX);      // f_T12 = f_CCU6 / prescaler
175     CCU60_TCTR0.U |= 0x2 << T12CLK_BIT_LSB_IDX;         // f_CCU6 = 50 MHz, prescaler = 1024
176     CCU60_TCTR0.U |= 0x1 << T12PRE_BIT_LSB_IDX;         // f_T12 = 48,828 Hz
177
178     CCU60_TCTR0.U &= ~(0x1 << CTM_BIT_LSB_IDX);         // T12 auto reset when period match (PM) occur
179
```

CCU60 내부 T12 모듈 사용 위한 레지스터 설정
: T12 Counter Increment 방식 선택 (Auto Clear Mode)

- CCU60를 위한 **T12** 레지스터
  - CCU60 내부 T12 모듈에서 1 clock 주기마다 1씩 값이 증가
  - **counter 레지스터의 초기화 필요함**
  - 0으로 초기화

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.3645

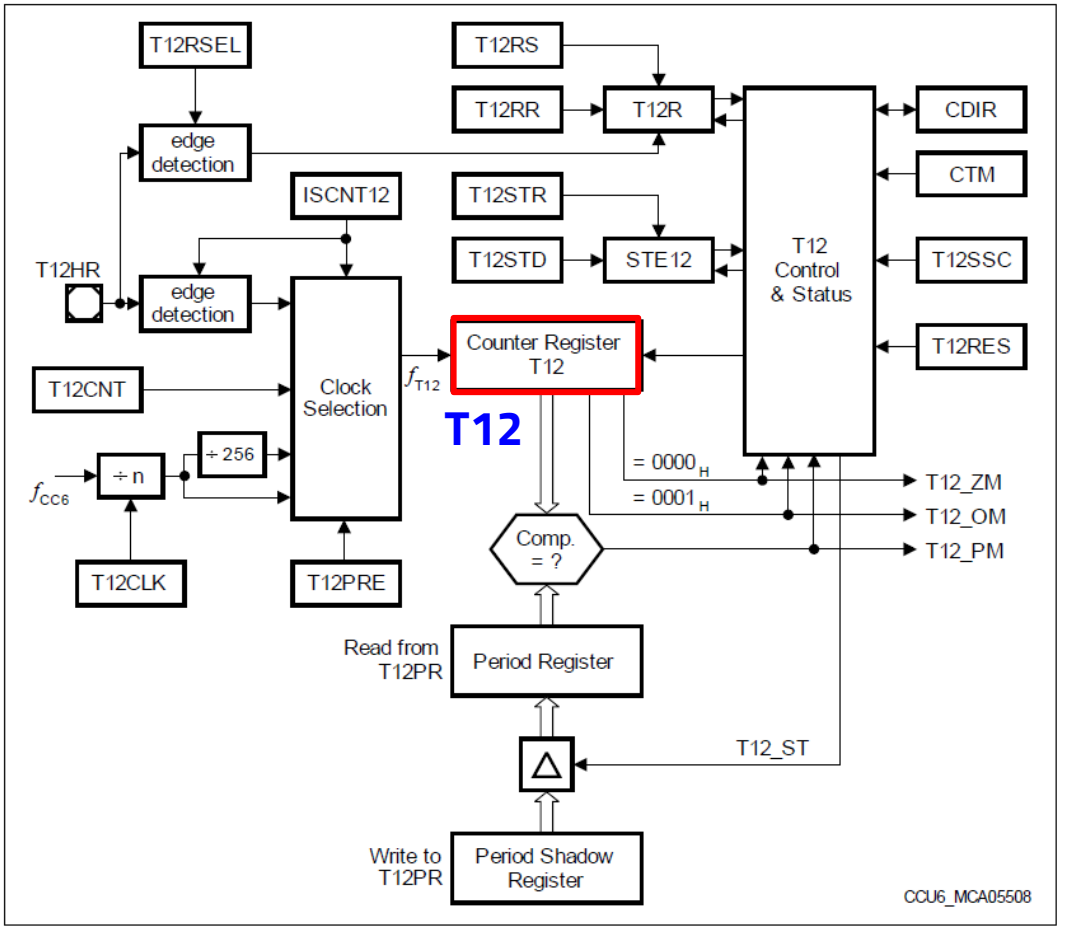


Figure 26-4 Timer T12 Logic and Period Comparators

CCU60 내부 T12 모듈 사용 위한 레지스터 설정

: T12 Control Register – TCTR0

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.3645

26.2.1 T12 Overview

Figure 26-4 shows a detailed block diagram of Timer T12. The functions of the timer T12 block are controlled by bits in registers TCTR0, TCTR2, and PISEL0.

Timer T12 receives its input clock (f_{T12}) from the module clock f_{CC6} via a programmable

- CCU60 레지스터 항목에서
  - **TCTR0** 레지스터 설정 필요

- T12의 Counter 레지스터와 Period Match 가 발생했을 때, **T12 counter 레지스터를 자동으로 초기화하고 다시 증가하도록 설정**

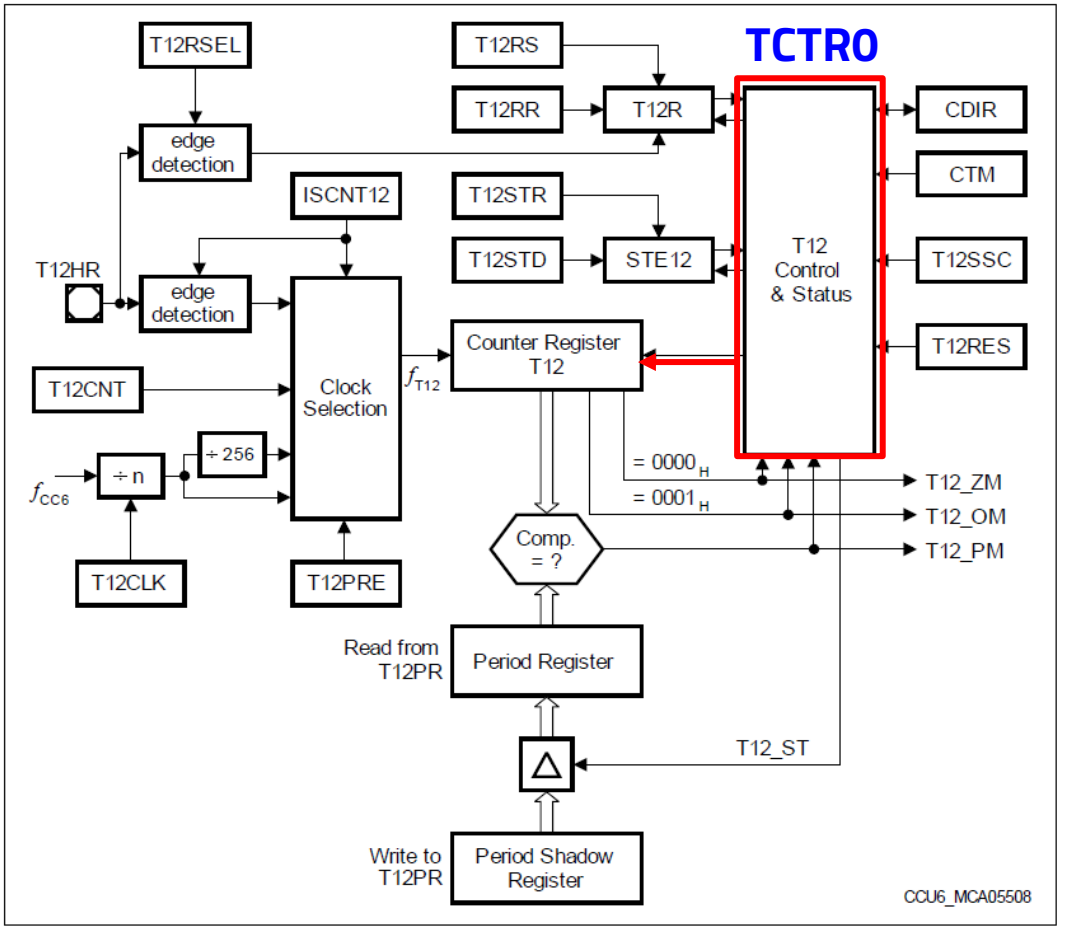


Figure 26-4 Timer T12 Logic and Period Comparators

CCU60 레지스터 설정 – TCTR0

1. CCU60 레지스터 영역의 주소 찾기
   - 시작 주소 (Base address) **= 0xF0002A00**

2. 사용할 레지스터의 주소 찾기
   - **TCTR0** 의 Offset Address = **0x70**

   → TCTR0 레지스터 주소 = 0xF0002A00 + 0x70 = **0xF0002A70**

Capture/Compare Control Registers

CMPSTAT	Compare State Register	60_H	U, SV	U, SV, P	Application Reset	26-39
CMPMODIF	Compare State Modification Register	64_H	U, SV	U, SV, P	Application Reset	26-42
T12MSEL	T12 Capture/Compare Mode Select Register	68_H	U, SV	U, SV, P	Application Reset	26-43
TCTR0	Timer Control Register 0	70_H	U, SV	U, SV, P	Application Reset	26-44
TCTR2	Timer Control Register 2	74_H	U, SV	U, SV, P	Application Reset	26-48
TCTR4	Timer Control Register 4	78...	U, SV	U	Application	26-51

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.3642

CCU60 레지스터 설정 – TCTR0

: Counter Auto Clear 모드 설정

3. 레지스터 write 값 결정
   - T12 모듈에서 counter와 period를 비교하여 match가 발생한 이후에, T12 counter 레지스터를 자동으로 0으로 초기화하고 다시 증가하도록 설정
   - **CTM 영역에 "0"값을 write**

TCTR0 레지스터 @ 0xF0002A70

TCTR0
Timer Control Register 0 (70_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		STE 13	T13R	T13 PRE	T13CLK			CTM	CDIR	STE 12	T12R	T12 PRE	T12CLK		
r		rh	rh	rw	rw			rw	rh	rh	rh	rw	rw		

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.3680

			T12. 0_B T12 counts up. 1_B T12 counts down.
CTM	7	rw	**T12 Operating Mode** 0_B Edge-aligned Mode: T12 always counts up and continues counting from zero after reaching the period value. 1_B Center-aligned Mode: T12 counts down after detecting a period-match and counts up after detecting a one-match.

CCU60 내부 T12 모듈 사용 위한 레지스터 설정

: T12 Counter를 구동하는 클럭 속도 조절

- CCU60를 위한 레지스터 항목에서
  - **T12CLK** 레지스터 설정 필요
  - **T12PRE** 레지스터 설정 필요

- CCU60 내부 T12 모듈에 입력되는 clock 주파수 설정을 위한 레지스터 설정 필요함

- **CCU60 모듈에 처음 입력되는 clock 신호의 주파수는 50 MHz**
- 이를 어떻게 분주(prescale)해서 **T12 모듈에 얼마의 clock 주파수로 전달할지 설정**

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.3645

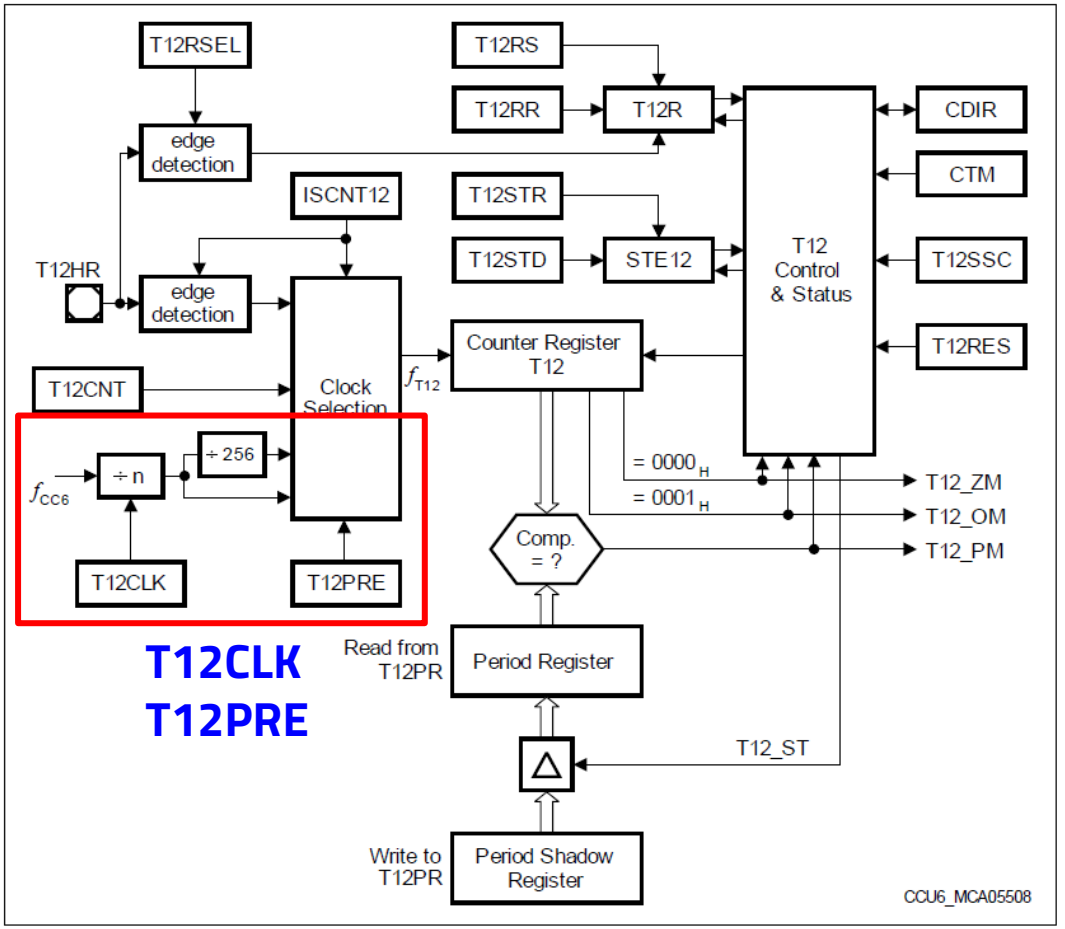


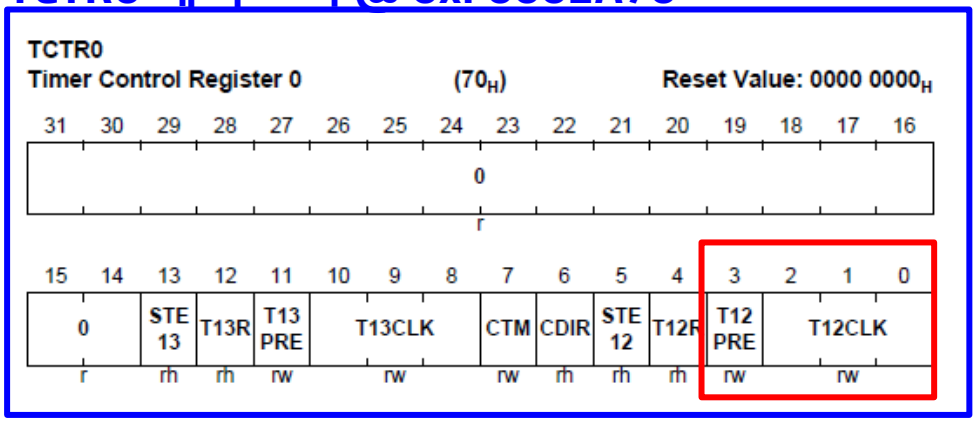
Figure 26-4 Timer T12 Logic and Period Comparators

CCU60 레지스터 설정 – TCTR0

: Clock 선택 (분주 or Prescaling)

3. 레지스터 write 값 결정
   - T12 모듈에서 사용할 clock 주파수 설정
   - CCU60 모듈에 처음 입력되는 clock 주파수가 50 MHz 이므로, **T12CLK 영역에 0x2값을 write** 하여 50 MHz / 4 = 12.5 MHz 의 clock 생성
   - **T12PRE 영역에 "1"값을 write**하여 **12.5 MHz / 256 = 48,828 Hz** 의 **T12 clock 생성 (최종)**

TCTR0 레지스터 @ 0xF0002A70



Field	Bits	Type	Description
T12CLK	[2:0]	rw	**Timer T12 Input Clock Select** Selects the input clock for timer T12 that is derived from the peripheral clock according to the equation $f_{T12} = f_{CC6} / 2^{<T12CLK>}$. 000_B $f_{T12} = f_{CC6}$ 001_B $f_{T12} = f_{CC6} / 2$ 010_B $f_{T12} = f_{CC6} / 4$ 011_B $f_{T12} = f_{CC6} / 8$ 100_B $f_{T12} = f_{CC6} / 16$ 101_B $f_{T12} = f_{CC6} / 32$ 110_B $f_{T12} = f_{CC6} / 64$ 111_B $f_{T12} = f_{CC6} / 128$
T12PRE	3	rw	**Timer T12 Prescaler Bit** In order to support higher clock frequencies, an additional prescaler factor of 1/256 can be enabled for the prescaler for T12. 0_B The additional prescaler for T12 is disabled. 1_B The additional prescaler for T12 is enabled.

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.3680

Lab6: CCU60 레지스터 설정 – TCTR0

```
56  #define CTM_BIT_LSB_IDX             7
57  #define T12PRE_BIT_LSB_IDX          3
58  #define T12CLK_BIT_LSB_IDX          0
```

```
150 void initCCU60(void)
151 {
152     // Password Access to unlock SCU_WDTSCON0
153     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
154     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0);    // wait until unlocked
155
156     // Modify Access to clear ENDINIT
157     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) & ~(1 << ENDINIT_BIT_LSB_IDX);
158     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0);    // wait until locked
159
160     CCU60_CLC.U &= ~(1 << DISR_BIT_LSB_IDX);    // enable CCY
161
162     // Password Access to unlock SCU_WDTSCON0
163     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
164     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0);    // wait until unlocked
165
166     // Modify Access to set ENDINIT
167     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
168     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0);    // wait until locked
169
170
171     // CCU60 T12 configurations
172     while((CCU60_CLC.U & (1 << DISS_BIT_LSB_IDX)) != 0);// wait until CCU60 module enabled
173
174     CCU60_TCTR0.U &= ~(0x7 << T12CLK_BIT_LSB_IDX);      // f_T12 = f_CCU6 / prescaler
175     CCU60_TCTR0.U |= 0x2 << T12CLK_BIT_LSB_IDX;         // f_CCU6 = 50 MHz, prescaler = 1024
176     CCU60_TCTR0.U |= 0x1 << T12PRE_BIT_LSB_IDX;         // f_T12 = 48,828 Hz
177
178     CCU60_TCTR0.U &= ~(0x1 << CTM_BIT_LSB_IDX);         // T12 auto reset when period match (PM) occur
179
```

CCU60 내부 T12 모듈 사용 위한 레지스터 설정

: Period 값 설정 레지스터 – T12PR

- CCU60 레지스터 항목에서
  - **T12PR** 레지스터 설정 필요

- CCU60 내부 T12 모듈의 counter 레지스터와 비교될 period 레지스터

- **<u>T12 모듈의 clock 주파수를 바탕으로</u>, 얼마의 시간 간격으로 Match 인터럽트를 발생시킬지 고려하여 Period Register를 셋팅해야 함**

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.3645

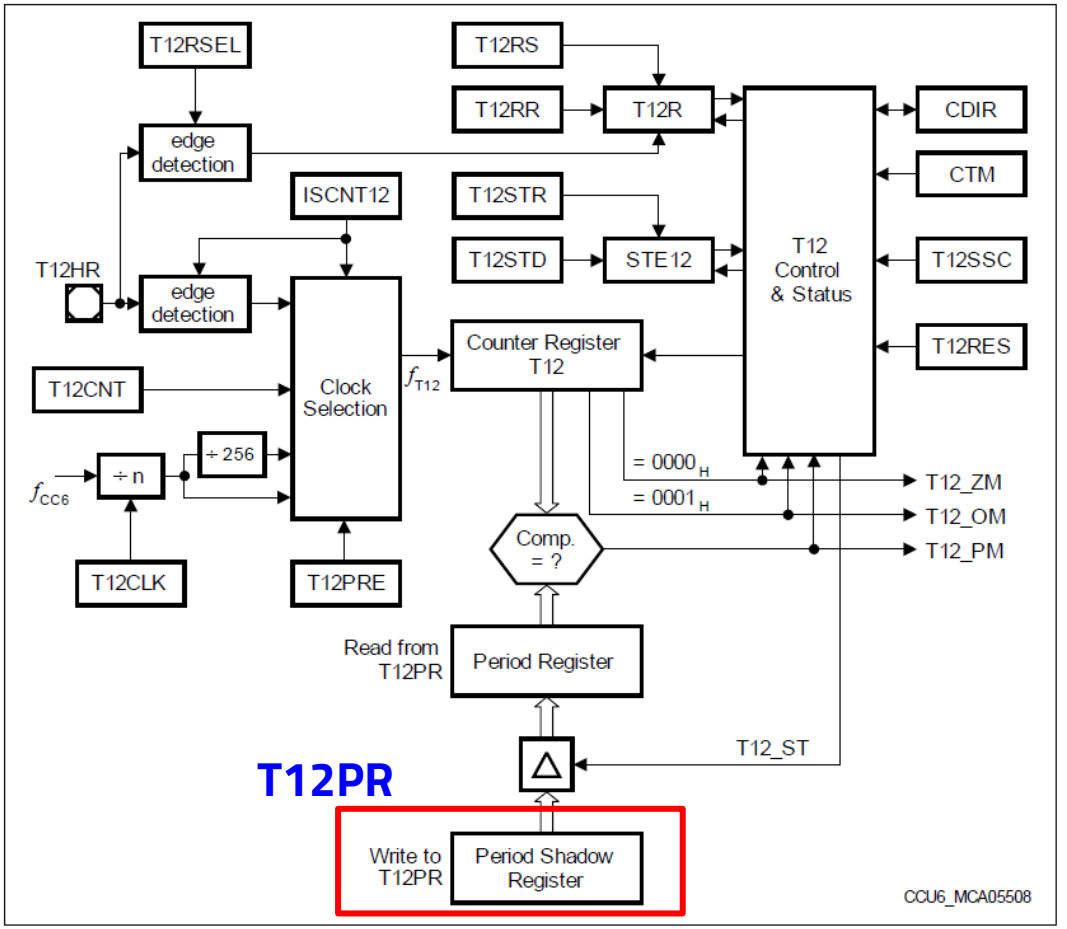


Figure 26-4 Timer T12 Logic and Period Comparators

CCU60 레지스터 설정 – T12PR

1. CCU60 레지스터 영역의 주소 찾기
   - 시작 주소 (Base address) **= 0xF0002A00**

2. 사용할 레지스터의 주소 찾기
   - **T12PR** 의 Offset Address = **0x24**

   → T12PR 레지스터 주소 = 0xF0002A00 + 0x24 = **0xF0002A24**

Timer T12 related Registers

T12	Timer 12 Counter Register	20_H	U, SV	U, SV, P	Application Reset	26-33
T12PR	Timer 12 Period Register	24_H	U, SV	U, SV, P	Application Reset	26-34
T12DTC	Dead-Time Control Register for Timer T12	28_H	U, SV	U, SV, P	Application Reset	26-37

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.3641

CCU60 레지스터 설정 – T12PR

: Clock 주파수와 Period를 고려하여 Match빈도 결정

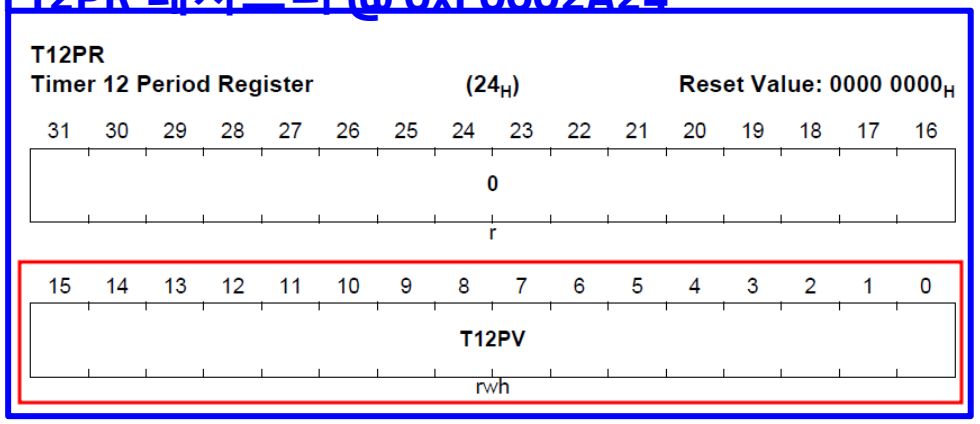
3. 레지스터 write 값 결정
   - 계획하는 Period Match 의 발생 주시를 고려하여 **T12PV 영역에 T12 모듈의 counter 레지스터와 match 여부 비교될 값을 write**

$$\text{Period Match가 발생하는 빈도 } (Freq.of\ Period\ Match) = \frac{(Freq.of\ Timer\ T12\ Clock)}{(Value\ of\ Period\ Register) + 1}$$

→ T12 모듈의 clock 주파수(frequency)가 **48,828 Hz** 이므로, 1초에 48,828 만큼 counter 증가함 → **0.5초간격으로** Period Match를 발생하려면 counter 가 **48,828 / 2 = 24,414** 에서 **Period Match 발생해야 함**

→ **T12PV 영역에 write할 값 = (48,828 / 2) - 1 = 24,413 (0x5F5D)**

T12PR 레지스터 @ 0xF0002A24



Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.3670

Field	Bits	Type	Description
T12PV	[15:0]	rwh	**T12 Period Value** The value T12PV defines the counter value for T12 leading to a period-match. When reaching this value, the timerT12 is set to zero (edge-aligned mode) or changes its count direction to down counting (center-aligned mode).
0	[31:16]	r	**Reserved;** Returns 0 if read; should be written with 0.

Lab7: CCU60 레지스터 설정 – T12PR

```
150 void initCCU60(void)
151 {
152     // Password Access to unlock SCU_WDTSCON0
153     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
154     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0);    // wait until unlocked
155
156     // Modify Access to clear ENDINIT
157     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) & ~(1 << ENDINIT_BIT_LSB_IDX);
158     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0);    // wait until locked
159
160     CCU60_CLC.U &= ~(1 << DISR_BIT_LSB_IDX);    // enable CCY
161
162     // Password Access to unlock SCU_WDTSCON0
163     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
164     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0);    // wait until unlocked
165
166     // Modify Access to set ENDINIT
167     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
168     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0);    // wait until locked
169
170
171     // CCU60 T12 configurations
172     while((CCU60_CLC.U & (1 << DISS_BIT_LSB_IDX)) != 0);// wait until CCU60 module enabled
173
174     CCU60_TCTR0.U &= ~(0x7 << T12CLK_BIT_LSB_IDX);      // f_T12 = f_CCU6 / prescaler
175     CCU60_TCTR0.U |= 0x2 << T12CLK_BIT_LSB_IDX;         // f_CCU6 = 50 MHz, prescaler = 1024
176     CCU60_TCTR0.U |= 0x1 << T12PRE_BIT_LSB_IDX;         // f_T12 = 48,828 Hz
177
178     CCU60_TCTR0.U &= ~(0x1 << CTM_BIT_LSB_IDX);         // T12 auto reset when period match (PM) occur
179
180
181     CCU60_T12PR.U = 24414 -1;                          // PM interrupt freq. = f_T12 / (T12PR + 1)
182     CCU60_TCTR4.U |= 0x1 << T12STR_BIT_LSB_IDX;         // load T12PR from shadow register
183
184     CCU60_T12.U = 0;                                   // clear T12 counter register
185
```

CCU60 내부 T12 모듈 사용 위한 레지스터 설정

: Counting Stop/Start제어, Period적용 → TCTR4

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.3646

enables an automatic stop of the timer when the current counting period is finished (see **Figure 26-7** and **Figure 26-8**).

The start or stop of T12 is controlled by the Run bit T12R that can be modified by bits in register **TCTR4**. The run bit can be set/cleared by software via the associated set/clear bits T12RS or T12RR, it can be set by a selectable edge of the input signal T12HR (**TCTR2**.T12RSEL), or it is cleared by hardware according to preselected conditions. The timer T12 run bit T12R must not be set while the applied T12 period value is zero. Timer T12 can be cleared via control bit T12RES. Setting this write-only bit does only

- CCU60 레지스터 항목에서
  - **TCTR4** 레지스터 설정 필요

- **T12의 counter 레지스터 증가 시작을 위한 설정**
- **Shadow Register에 저장된 period 설정 값을 T12PR 레지스터에 실제 적용**

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.3645

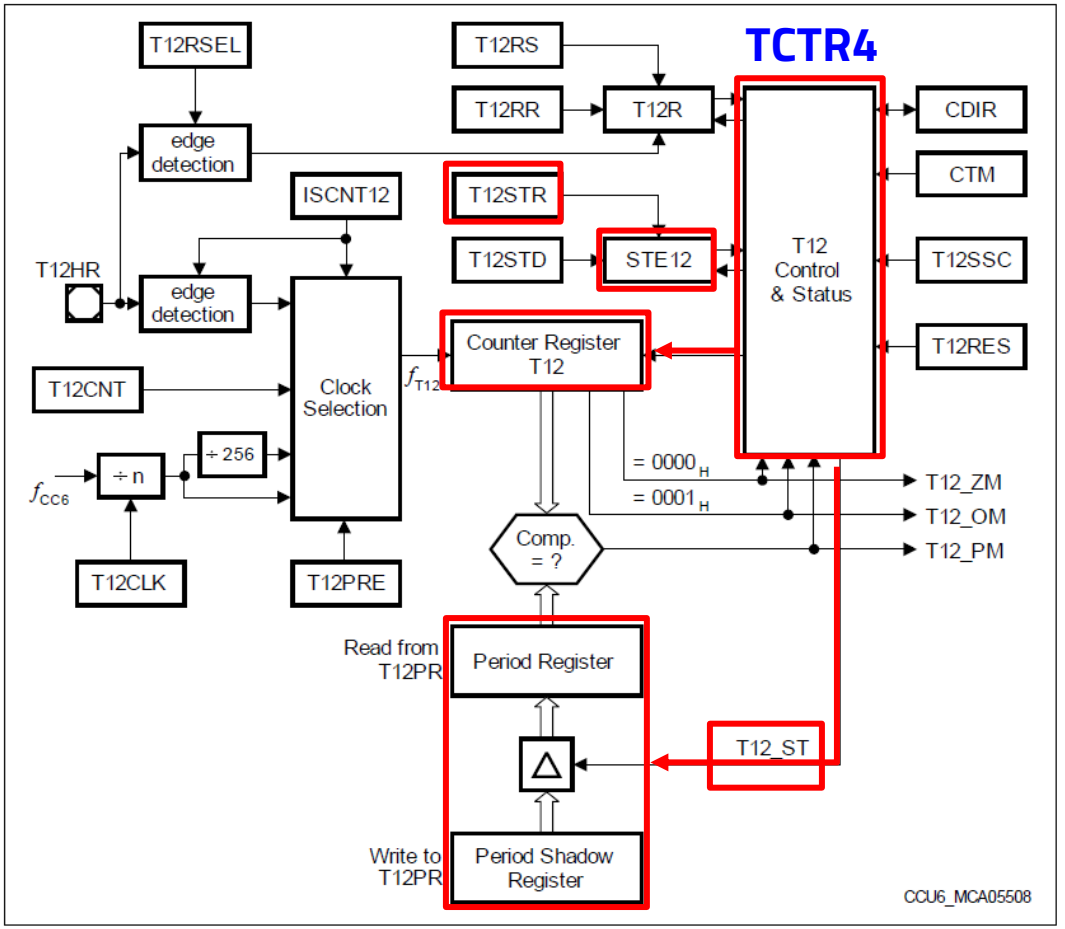


Figure 26-4 Timer T12 Logic and Period Comparators

CCU60 레지스터 설정 – TCTR4

1. CCU60 레지스터 영역의 주소 찾기
   - 시작 주소 (Base address) **= 0xF0002A00**

2. 사용할 레지스터의 주소 찾기
   - **TCTR4** 의 Offset Address = **0x78**

   → TCTR4 레지스터 주소 = 0xF0002A00 + 0x78 = **0xF0002A78**

TCTR2	Timer Control Register 2	74_H	U, SV	U, SV, P	Application Reset	26-48
TCTR4	Timer Control Register 4	78_H	U, SV	U, SV, P	Application Reset	26-51
Timer T13 related Registers						
T13	Timer 13 Counter	50_H	U, SV	U,	Application	26-66

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.3642

CCU60 레지스터 설정 – TCTR4

: Shadow값을 T12PR에 Store

3. 레지스터 write 값 결정
   - 앞서 T12PR 레지스터에 값을 write하면 바로 적용되지 않고 Period Shadow Register에 머무르게 됨
   - Period Shadow Register에 저장된 period 설정 값을 T12PR 레지스터에 옮기기 위해 **T12STR 영역에 "1"값을 write**

TCTR4 레지스터 @ 0xF0002A78

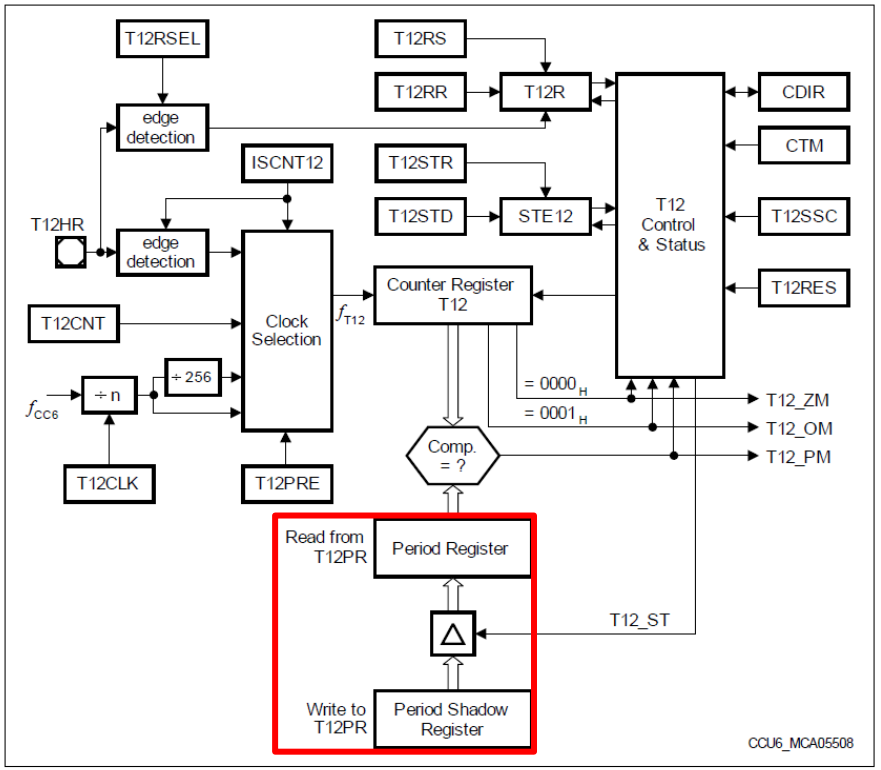
TCTR4
Timer Control Register 4 (78_H) **Reset Value: 0000 0000$_H$**

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
T13 STD	T13 STR	T13 CNT	0		T13 RES	T13 RS	T13 RR	T12 STD	T12 STR	T12 CNT	0	DT RES	T12 RES	T12 RS	T12 RR
w	w	w	r		w	w	w	w	w	w	r	w	w	w	w

Field	Bits	Type	Description
T12STR	6	w	Timer T12 Shadow Transfer Request 0_B No action 1_B STE12 is set, enabling the shadow transfer.
T12STD	7	w	Timer T12 Shadow Transfer Disable 0_B No action

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.3688



2022 **Figure 26-4 Timer T12 Logic and Period Comparators**

Lab8: CCU60 레지스터 설정 – TCTR4

```
150 void initCCU60(void)
151 {
152     // Password Access to unlock SCU_WDTSCON0
153     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
154     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0);    // wait until unlocked
155
156     // Modify Access to clear ENDINIT
157     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) & ~(1 << ENDINIT_BIT_LSB_IDX);
158     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0);    // wait until locked
159
160     CCU60_CLC.U &= ~(1 << DISR_BIT_LSB_IDX);    // enable CCY
161
162     // Password Access to unlock SCU_WDTSCON0
163     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
164     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0);    // wait until unlocked
165
166     // Modify Access to set ENDINIT
167     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
168     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0);    // wait until locked
169
170
171     // CCU60 T12 configurations
172     while((CCU60_CLC.U & (1 << DISS_BIT_LSB_IDX)) != 0);// wait until CCU60 module enabled
173
174     CCU60_TCTR0.U &= ~(0x7 << T12CLK_BIT_LSB_IDX);      // f_T12 = f_CCU6 / prescaler
175     CCU60_TCTR0.U |= 0x2 << T12CLK_BIT_LSB_IDX;         // f_CCU6 = 50 MHz, prescaler = 1024
176     CCU60_TCTR0.U |= 0x1 << T12PRE_BIT_LSB_IDX;         // f_T12 = 48,828 Hz
177
178     CCU60_TCTR0.U &= ~(0x1 << CTM_BIT_LSB_IDX);         // T12 auto reset when period match (PM) occur
179
180
181     CCU60_T12PR.U = 24414 - 1;                      // PM interrupt freq. = f_T12 / (T12PR + 1)
182     CCU60_TCTR4.U |= 0x1 << T12STR_BIT_LSB_IDX;         // load T12PR from shadow register
183
184     CCU60_T12.U = 0;                                // clear T12 counter register
185
186
187     // CCU60 T12 PM interrupt setting
188     CCU60_INP.U &= ~(0x3 << INPT12_BIT_LSB_IDX);        // service request output SR0 selected
189     CCU60_IEN.U |= 0x1 << ENT12PM_BIT_LSB_IDX;          // enable T12 PM interrupt
190
191
192     // SRC setting for CCU60
193     SRC_CCU6_CCU60_SR0.U &= ~(0xFF << SRPN_BIT_LSB_IDX);
194     SRC_CCU6_CCU60_SR0.U |= 0x0B << SRPN_BIT_LSB_IDX;   // set priority 0x0B
195
196     SRC_CCU6_CCU60_SR0.U &= ~(0x3 << TOS_BIT_LSB_IDX);  // CPU0 service T12 PM interrupt
197
198     SRC_CCU6_CCU60_SR0.U |= 0x1 << SRE_BIT_LSB_IDX;     // SR0 enabled
199
200
201     // CCU60 T12 counting start
202     CCU60_TCTR4.U = 0x1 << T12RS_BIT_LSB_IDX;           // T12 start counting
203 }
204
```

```
59 #define T12STR_BIT_LSB_IDX          6
```

CCU60 레지스터 설정 – T12

1. CCU60 레지스터 영역의 주소 찾기
   - 시작 주소 (Base address) **= 0xF0002A00**

2. 사용할 레지스터의 주소 찾기
   - **T12** 의 Offset Address = **0x20**

   → T12 레지스터 주소 = 0xF0002A00 + 0x20 = **0xF0002A20**

				SV, P	Reset	

Timer T12 related Registers

T12	Timer 12 Counter Register	20_H	U, SV	U, SV, P	Application Reset	26-33
T12PR	Timer 12 Period Register	24_H	U, SV	U, SV, P	Application Reset	26-34
T12DTC	Dead-Time Control	28_H	U, SV	U,	Application	26-37

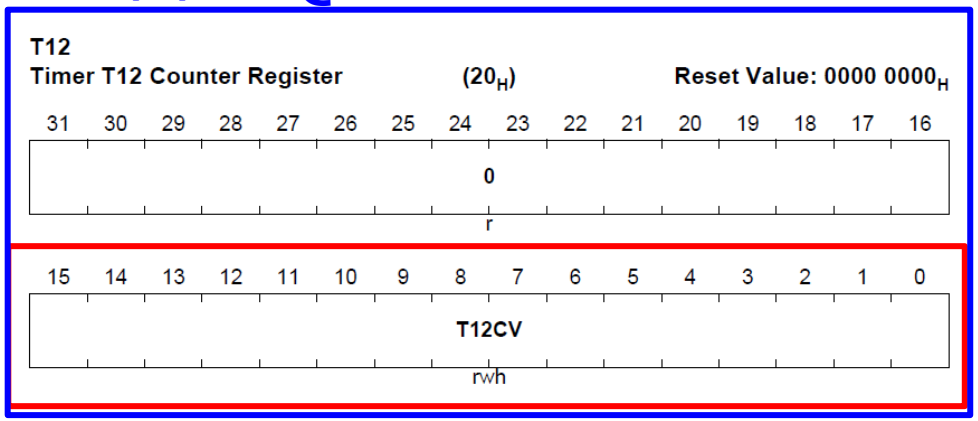
Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.3641

CCU60 레지스터 설정 – T12

: Counter Value 초기화

3. 레지스터 write 값 결정
   - T12 모듈의 counter 값을 초기화하기 위해 **T12CV 영역에 "0" write**

T12 레지스터 @ 0xF0002A20



Field	Bits	Type	Description
T12CV	[15:0]	rwh	**Timer 12 Counter Value** This register represents the 16-bit counter value of Timer12.
0	[31:16]	r	**Reserved;** Returns 0 if read; should be written with 0.

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.3669

CCU60 레지스터 설정 – T12

```
150 void initCCU60(void)
151 {
152     // Password Access to unlock SCU_WDTSCON0
153     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
154     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0);    // wait until unlocked
155
156     // Modify Access to clear ENDINIT
157     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) & ~(1 << ENDINIT_BIT_LSB_IDX);
158     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0);    // wait until locked
159
160     CCU60_CLC.U &= ~(1 << DISR_BIT_LSB_IDX);    // enable CCY
161
162     // Password Access to unlock SCU_WDTSCON0
163     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
164     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0);    // wait until unlocked
165
166     // Modify Access to set ENDINIT
167     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
168     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0);    // wait until locked
169
170
171     // CCU60 T12 configurations
172     while((CCU60_CLC.U & (1 << DISS_BIT_LSB_IDX)) != 0);// wait until CCU60 module enabled
173
174     CCU60_TCTR0.U &= ~(0x7 << T12CLK_BIT_LSB_IDX);      // f_T12 = f_CCU6 / prescaler
175     CCU60_TCTR0.U |= 0x2 << T12CLK_BIT_LSB_IDX;         // f_CCU6 = 50 MHz, prescaler = 1024
176     CCU60_TCTR0.U |= 0x1 << T12PRE_BIT_LSB_IDX;         // f_T12 = 48,828 Hz
177
178     CCU60_TCTR0.U &= ~(0x1 << CTM_BIT_LSB_IDX);         // T12 auto reset when period match (PM) occur
179
180
181     CCU60_T12PR.U = 24414 -1;                       // PM interrupt freq. = f_T12 / (T12PR + 1)
182     CCU60_TCTR4.U |= 0x1 << T12STR_BIT_LSB_IDX;         // load T12PR from shadow register
183
184     CCU60_T12.U = 0;                                // clear T12 counter register
185
186
187     // CCU60 T12 PM interrupt setting
188     CCU60_INP.U &= ~(0x3 << INPT12_BIT_LSB_IDX);        // service request output SR0 selected
189     CCU60_IEN.U |= 0x1 << ENT12PM_BIT_LSB_IDX;          // enable T12 PM interrupt
190
191
192     // SRC setting for CCU60
193     SRC_CCU6_CCU60_SR0.U &= ~(0xFF << SRPN_BIT_LSB_IDX);
194     SRC_CCU6_CCU60_SR0.U |= 0x0B << SRPN_BIT_LSB_IDX;   // set priority 0x0B
195
196     SRC_CCU6_CCU60_SR0.U &= ~(0x3 << TOS_BIT_LSB_IDX);  // CPU0 service T12 PM interrupt
197
198     SRC_CCU6_CCU60_SR0.U |= 0x1 << SRE_BIT_LSB_IDX;     // SR0 enabled
199
200
201     // CCU60 T12 counting start
202     CCU60_TCTR4.U = 0x1 << T12RS_BIT_LSB_IDX;           // T12 start counting
203 }
204
```

CCU60 레지스터 설정 – TCTR4

3. 레지스터 write 값 결정
   - T12 의 counter 증가 동작을 시작하기 위해 **T12RS 영역에 "1"값을 write**

TCTR4 레지스터 @ 0xF0002A78

TCTR4
Timer Control Register 4 (78_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
T13 STD	T13 STR	T13 CNT	0		T13 RES	T13 RS	T13 RR	T12 STD	T12 STR	T12 CNT	0	DT RES	T12 RES	T12 RS	T12 RR
w	w	w	r		w	w	w	w	w	w	r	w	w	w	w

Field	Bits	Type	Description
T12STR	6	w	**Timer T12 Shadow Transfer Request** 0_B No action 1_B STE12 is set, enabling the shadow transfer.
T12STD	7	w	**Timer T12 Shadow Transfer Disable** 0_B No action

Field	Bits	Type	Description
T12RR	0	w	**Timer T12 Run Reset** Setting this bit clears the T12R bit. 0_B T12R is not influenced. 1_B T12R is cleared, T12 stops counting.
T12RS	1	w	**Timer T12 Run Set** Setting this bit sets the T12R bit. 0_B T12R is not influenced. 1_B T12R is set, T12 starts counting.
T12RES	2	w	**Timer T12 Reset**

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.3687

Lab8: CCU60 레지스터 설정 – TCTR4

: Timer Start를 맨마지막에 해야 함

```
150 void initCCU60(void)
151 {
152     // Password Access to unlock SCU_WDTSCON0
153     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
154     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0);    // wait until unlocked
155
156     // Modify Access to clear ENDINIT
157     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) & ~(1 << ENDINIT_BIT_LSB_IDX);
158     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0);    // wait until locked
159
160     CCU60_CLC.U &= ~(1 << DISR_BIT_LSB_IDX);    // enable CCY
161
162     // Password Access to unlock SCU_WDTSCON0
163     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
164     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0);    // wait until unlocked
165
166     // Modify Access to set ENDINIT
167     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
168     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0);    // wait until locked
169
170
171     // CCU60 T12 configurations
172     while((CCU60_CLC.U & (1 << DISS_BIT_LSB_IDX)) != 0);// wait until CCU60 module enabled
173
174     CCU60_TCTR0.U &= ~(0x7 << T12CLK_BIT_LSB_IDX);      // f_T12 = f_CCU6 / prescaler
175     CCU60_TCTR0.U |= 0x2 << T12CLK_BIT_LSB_IDX;         // f_CCU6 = 50 MHz, prescaler = 1024
176     CCU60_TCTR0.U |= 0x1 << T12PRE_BIT_LSB_IDX;         // f_T12 = 48,828 Hz
177
178     CCU60_TCTR0.U &= ~(0x1 << CTM_BIT_LSB_IDX);         // T12 auto reset when period match (PM) occur
179
180
181     CCU60_T12PR.U = 24414 -1;                       // PM interrupt freq. = f_T12 / (T12PR + 1)
182     CCU60_TCTR4.U |= 0x1 << T12STR_BIT_LSB_IDX;         // load T12PR from shadow register
183
184     CCU60_T12.U = 0;                                // clear T12 counter register
185
186
187     // CCU60 T12 PM interrupt setting
188     CCU60_INP.U &= ~(0x3 << INPT12_BIT_LSB_IDX);        // service request output SR0 selected
189     CCU60_IEN.U |= 0x1 << ENT12PM_BIT_LSB_IDX;          // enable T12 PM interrupt
190
191
192     // SRC setting for CCU60
193     SRC_CCU6_CCU60_SR0.U &= ~(0xFF << SRPN_BIT_LSB_IDX);
194     SRC_CCU6_CCU60_SR0.U |= 0x0B << SRPN_BIT_LSB_IDX;   // set priority 0x0B
195
196     SRC_CCU6_CCU60_SR0.U &= ~(0x3 << TOS_BIT_LSB_IDX);  // CPU0 service T12 PM interrupt
197
198     SRC_CCU6_CCU60_SR0.U |= 0x1 << SRE_BIT_LSB_IDX;     // SR0 enabled
199
200
201     // CCU60 T12 counting start
202     CCU60_TCTR4.U = 0x1 << T12RS_BIT_LSB_IDX;           // T12 start counting
203 }
204
```

타이머 인터럽트 발생 설정

```
60 #define T12RS_BIT_LSB_IDX          1
```

T12 Interrupt 사용 위한 레지스터 설정
: 인터럽트 Enable → IEN 레지스터

- CCU60 레지스터 항목에서
  - **IEN** 레지스터 설정 필요

- CCU60 내부 T12 모듈이 Interrupt 를 생성할 수 있도록 enable 하기 위한 레지스터
- **T12에서 period match가 발생할 때마다 Interrupt를 생성**하기 위한 설정 필요

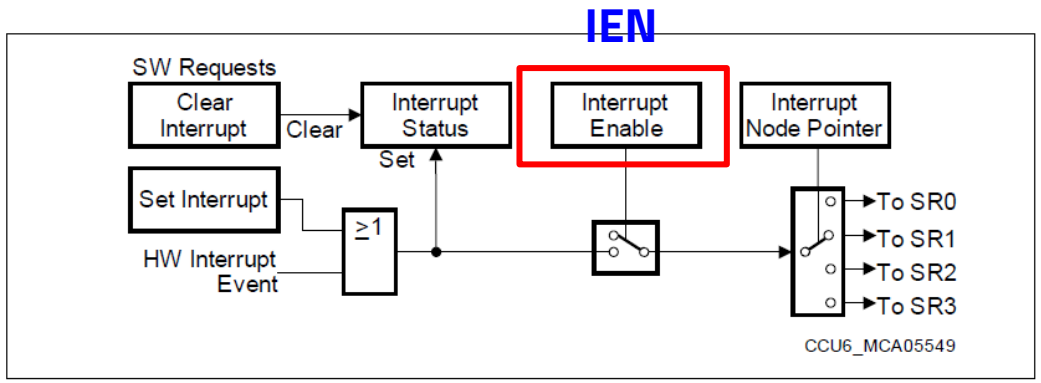


Figure 26-42 General Interrupt Structure

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.3732

CCU60 레지스터 설정 – IEN

1. CCU60 레지스터 영역의 주소 찾기
   - 시작 주소 (Base address) **= 0xF0002A00**

2. 사용할 레지스터의 주소 찾기
   - **IEN** 의 Offset Address = **0xB0**

   → IEN 레지스터 주소 = 0xF0002A00 + 0xB0 = **0xF0002AB0**

Interrupt Status and Node Registers

IS	Interrupt Status Register	$A0_H$	U, SV	U,SV, P	Application Reset	26-98
ISS	Interrupt Status Set Register	$A4_H$	U, SV	U, SV, P	Application Reset	26-101
ISR	Interrupt Status Reset Register	$A8_H$	U, SV	U, SV, P	Application Reset	26-103
INP	Interrupt Node Pointer Register	AC_H	U, SV	U, SV, P	Application Reset	26-108
IEN	Interrupt Node Pointer Register	$B0_H$	U, SV	U, SV, P	Application Reset	26-105

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.3643

CCU60 레지스터 설정 – IEN

: Timer Interrupt 발생하도록 설정

3. 레지스터 write 값 결정
   - T12 모듈에서 Period Match 가 발생할 때마다 Interrupt를 생성하도록 enable 하기 위해 **ENT12PM 영역에 "1"값을 write**

IEN 레지스터 @ 0xF0002AB0

IEN
Interrupt Enable Register **($B0_H$)** **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
ENSTR	ENIDLE	ENWHE	ENCHE	0	ENTRPF	ENT13PM	ENT13CM	ENT12PM	ENT12OM	ENCC62F	ENCC62R	ENCC61F	ENCC61R	ENCC60F	ENCC60R
rw	rw	rw	rw	r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Field	Bits	Type	Description
ENT12OM	6	rw	**Enable Interrupt for T12 One-Match** 0_B No interrupt will be generated if the set condition for bit T12OM in register IS occurs. 1_B An interrupt will be generated if the set condition for bit T12OM in register IS occurs. The service request output that will be activated is selected by bit field INPT12.
ENT12PM	7	rw	**Enable Interrupt for T12 Period-Match** 0_B No interrupt will be generated if the set condition for bit T12PM in register IS occurs. 1_B An interrupt will be generated if the set condition for bit T12PM in register IS occurs. The service request output that will be activated is selected by bit field INPT12.
ENT13CM	8	rw	**Enable Interrupt for T13 Compare-Match**

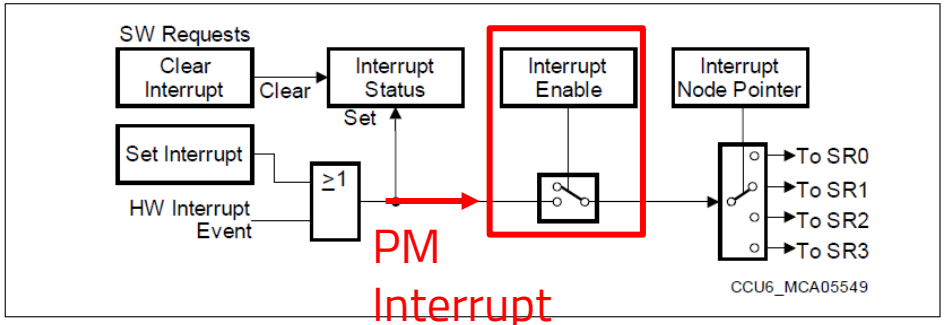


Figure 26-42 General Interrupt Structure

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.3741

Lab9: CCU60 레지스터 설정 – IEN

```c
void initCCU60(void)
{
    // Password Access to unlock SCU_WDTSCON0
    SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
    while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0);    // wait until unlocked

    // Modify Access to clear ENDINIT
    SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) & ~(1 << ENDINIT_BIT_LSB_IDX);
    while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0);    // wait until locked

    CCU60_CLC.U &= ~(1 << DISR_BIT_LSB_IDX);    // enable CCY

    // Password Access to unlock SCU_WDTSCON0
    SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
    while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0);    // wait until unlocked

    // Modify Access to set ENDINIT
    SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
    while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0);    // wait until locked


    // CCU60 T12 configurations
    while((CCU60_CLC.U & (1 << DISS_BIT_LSB_IDX)) != 0);// wait until CCU60 module enabled

    CCU60_TCTR0.U &= ~(0x7 << T12CLK_BIT_LSB_IDX);      // f_T12 = f_CCU6 / prescaler
    CCU60_TCTR0.U |= 0x2 << T12CLK_BIT_LSB_IDX;         // f_CCU6 = 50 MHz, prescaler = 1024
    CCU60_TCTR0.U |= 0x1 << T12PRE_BIT_LSB_IDX;         // f_T12 = 48,828 Hz

    CCU60_TCTR0.U &= ~(0x1 << CTM_BIT_LSB_IDX);         // T12 auto reset when period match (PM) occur


    CCU60_T12PR.U = 24414 -1;                       // PM interrupt freq. = f_T12 / (T12PR + 1)
    CCU60_TCTR4.U |= 0x1 << T12STR_BIT_LSB_IDX;         // load T12PR from shadow register

    CCU60_T12.U = 0;                                // clear T12 counter register


    // CCU60 T12 PM interrupt setting
    CCU60_INP.U &= ~(0x3 << INPT12_BIT_LSB_IDX);        // service request output SR0 selected
    CCU60_IEN.U |= 0x1 << ENT12PM_BIT_LSB_IDX;          // enable T12 PM interrupt


    // SRC setting for CCU60
    SRC_CCU6_CCU60_SR0.U &= ~(0xFF << SRPN_BIT_LSB_IDX);
    SRC_CCU6_CCU60_SR0.U |= 0x0B << SRPN_BIT_LSB_IDX;   // set priority 0x0B

    SRC_CCU6_CCU60_SR0.U &= ~(0x3 << TOS_BIT_LSB_IDX);  // CPU0 service T12 PM interrupt

    SRC_CCU6_CCU60_SR0.U |= 0x1 << SRE_BIT_LSB_IDX;     // SR0 enabled


    // CCU60 T12 counting start
    CCU60_TCTR4.U = 0x1 << T12RS_BIT_LSB_IDX;           // T12 start counting
}
```

```c
#define ENT12PM_BIT_LSB_IDX         7
```

T12 Interrupt 사용 위한 레지스터 설정
: 발생된 인터럽트를 출력으로 연결 – INP레지스터

- CCU60 레지스터 항목에서
  - **INP** 레지스터 설정 필요

- CCU60 - T12 모듈에서 발생한 Interrupt 를 SR0 노드로 연결하기 위한 레지스터 설정
- T12에서 발생한 period match Interrupt를 4개의 Node Pointer SR0-3 중, **SR0에 연결하기 위한 설정 필요**

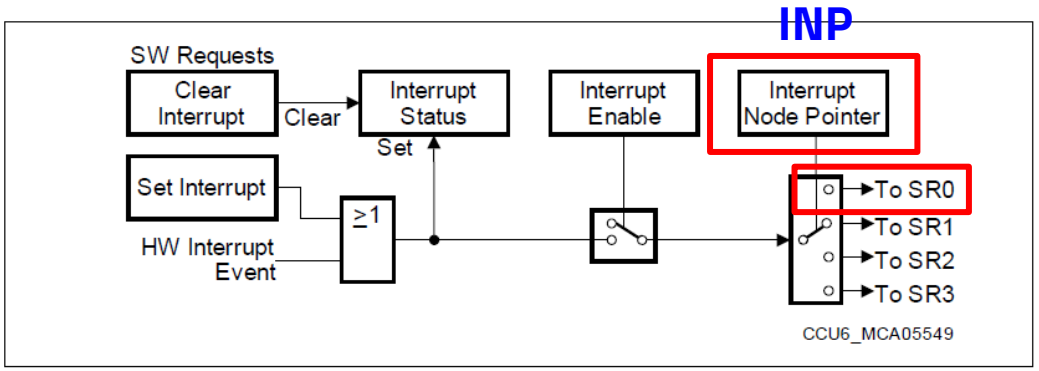
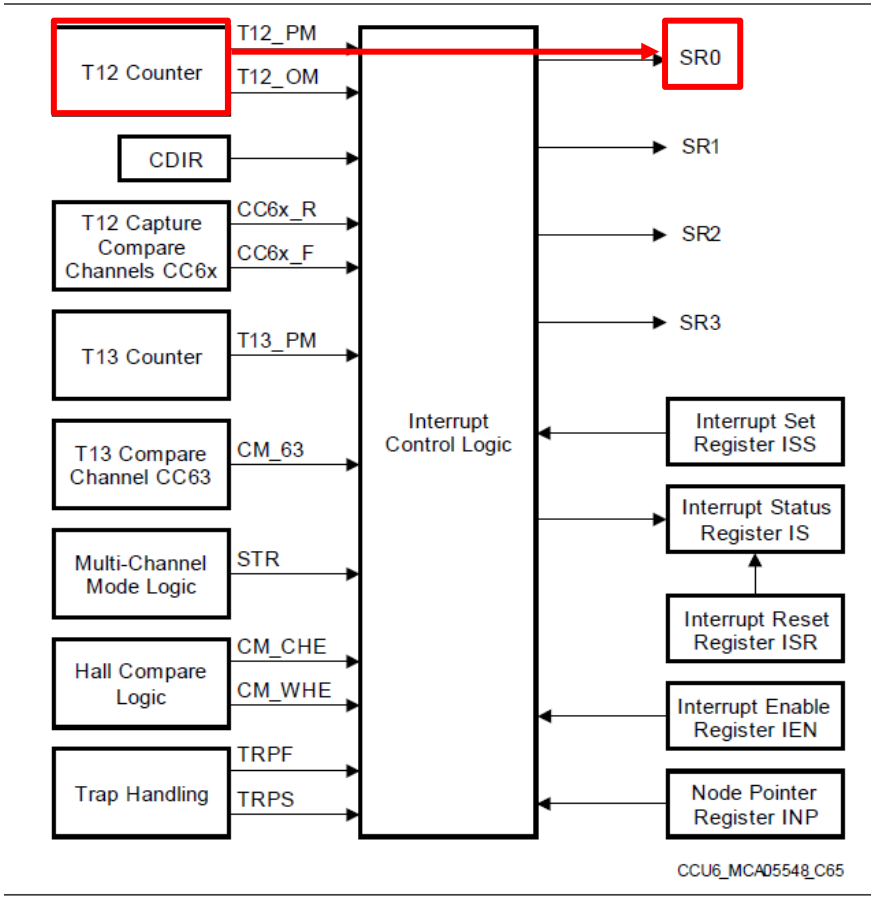


Figure 26-42 General Interrupt Structure

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.3732



6-43 Interrupt Sources and Events

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.3733

CCU60 레지스터 설정 – INP

1. CCU60 레지스터 영역의 주소 찾기
   - 시작 주소 (Base address) **= 0xF0002A00**

2. 사용할 레지스터의 주소 찾기
   - **INP** 의 Offset Address = **0xAC**

   → INP 레지스터 주소 = 0xF0002A00 + 0xAC = **0xF0002AAC**

Interrupt Status and Node Registers

IS	Interrupt Status Register	$A0_H$	U, SV	U,SV, P	Application Reset	26-98
ISS	Interrupt Status Set Register	$A4_H$	U, SV	U, SV, P	Application Reset	26-101
ISR	Interrupt Status Reset Register	$A8_H$	U, SV	U, SV, P	Application Reset	26-103
INP	Interrupt Node Pointer Register	AC_H	U, SV	U, SV, P	Application Reset	26-108
IEN	Interrupt Node Pointer Register	$B0_H$	U, SV	U, SV, P	Application Reset	26-105

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.3643

CCU60 레지스터 설정 – INP

3. 레지스터 write 값 결정
   - T12 모듈에서 발생하는 Period Match Interrupt를 Interrupt Router (IR)에 전달할 때, **SR0 Node**로 전달하기 위해 Node Pointer 설정 필요
   - **INPT12 영역에 0x0 값을 write (SR0 이므로 "0"값)**

INP 레지스터 @ 0xF0002AAC

INP
Interrupt Node Pointer Register (AC_H) **Reset Value: 0000 3940$_H$**

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		INP T13		INP T12		INP ERR		INP CHE		INP CC62		INP CC61		INP CC60	
r		rw		rw		rw		rw		rw		rw		rw	

			Coding see INPCC6x.
INPT12	[11:10]	rw	**Interrupt Node Pointer for Timer12 Interrupts** This bit field defines the service request output activated due to a set condition for bit T12OM (if enabled by bit ENT12OM) or for bit T12PM (if enabled by bit ENT12PM). Coding see INPCC6x.

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.3744

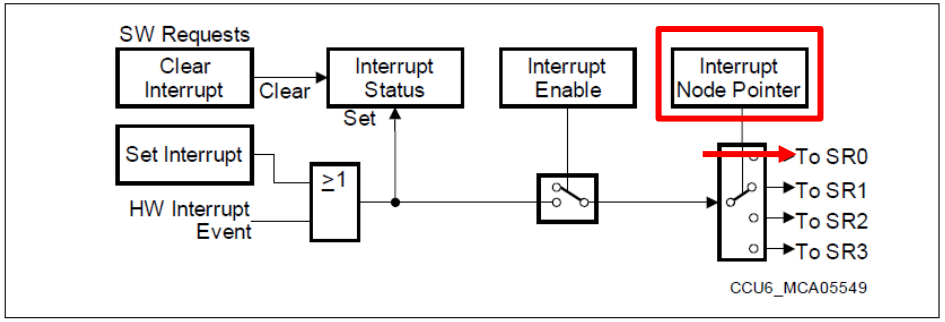


Figure 26-42 General Interrupt Structure

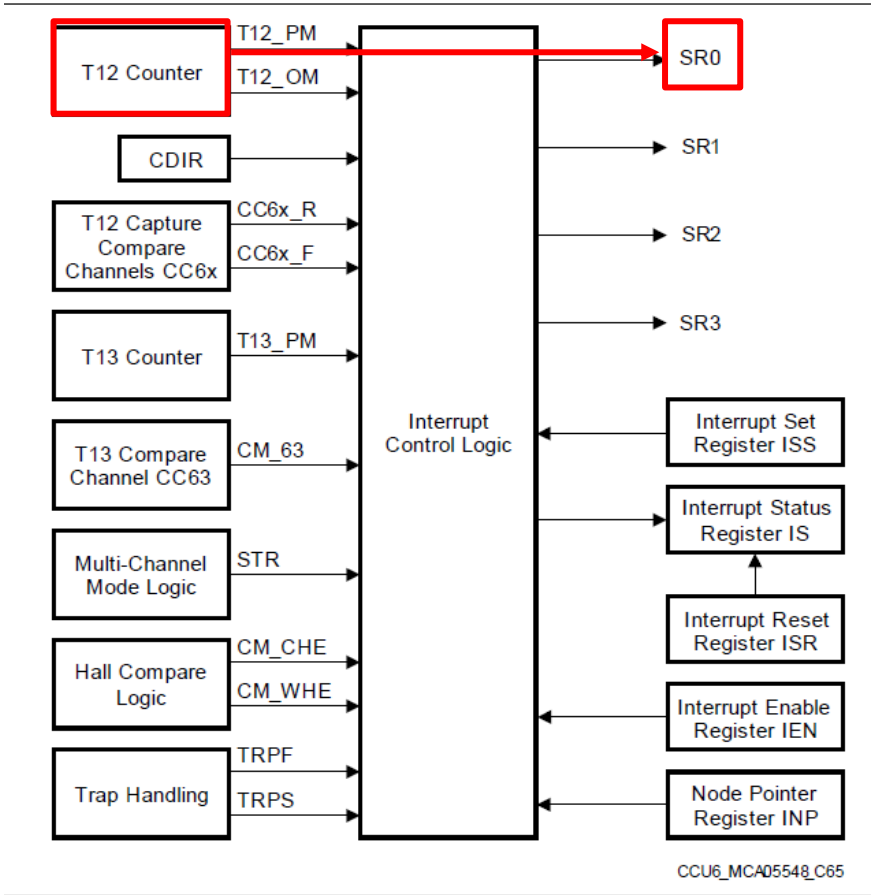
Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.3732

Lab10: CCU60 레지스터 설정 – INP

```
150 void initCCU60(void)
151 {
152     // Password Access to unlock SCU_WDTSCON0
153     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
154     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0);    // wait until unlocked
155
156     // Modify Access to clear ENDINIT
157     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) & ~(1 << ENDINIT_BIT_LSB_IDX);
158     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0);    // wait until locked
159
160     CCU60_CLC.U &= ~(1 << DISR_BIT_LSB_IDX);    // enable CCY
161
162     // Password Access to unlock SCU_WDTSCON0
163     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
164     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0);    // wait until unlocked
165
166     // Modify Access to set ENDINIT
167     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
168     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0);    // wait until locked
169
170
171     // CCU60 T12 configurations
172     while((CCU60_CLC.U & (1 << DISS_BIT_LSB_IDX)) != 0);// wait until CCU60 module enabled
173
174     CCU60_TCTR0.U &= ~(0x7 << T12CLK_BIT_LSB_IDX);      // f_T12 = f_CCU6 / prescaler
175     CCU60_TCTR0.U |= 0x2 << T12CLK_BIT_LSB_IDX;         // f_CCU6 = 50 MHz, prescaler = 1024
176     CCU60_TCTR0.U |= 0x1 << T12PRE_BIT_LSB_IDX;         // f_T12 = 48,828 Hz
177
178     CCU60_TCTR0.U &= ~(0x1 << CTM_BIT_LSB_IDX);         // T12 auto reset when period match (PM) occur
179
180
181     CCU60_T12PR.U = 24414 -1;                       // PM interrupt freq. = f_T12 / (T12PR + 1)
182     CCU60_TCTR4.U |= 0x1 << T12STR_BIT_LSB_IDX;         // load T12PR from shadow register
183
184     CCU60_T12.U = 0;                                // clear T12 counter register
185
186
187     // CCU60 T12 PM interrupt setting
188     CCU60_INP.U &= ~(0x3 << INPT12_BIT_LSB_IDX);        // service request output SR0 selected
189     CCU60_IEN.U |= 0x1 << ENT12PM_BIT_LSB_IDX;          // enable T12 PM interrupt
190
191
192     // SRC setting for CCU60
193     SRC_CCU6_CCU60_SR0.U &= ~(0xFF << SRPN_BIT_LSB_IDX);
194     SRC_CCU6_CCU60_SR0.U |= 0x0B << SRPN_BIT_LSB_IDX;   // set priority 0x0B
195
196     SRC_CCU6_CCU60_SR0.U &= ~(0x3 << TOS_BIT_LSB_IDX);  // CPU0 service T12 PM interrupt
197
198     SRC_CCU6_CCU60_SR0.U |= 0x1 << SRE_BIT_LSB_IDX;     // SR0 enabled
199
200
201     // CCU60 T12 counting start
202     CCU60_TCTR4.U = 0x1 << T12RS_BIT_LSB_IDX;           // T12 start counting
203 }
204
```

```
61 #define INPT12_BIT_LSB_IDX          10
62 #define ENT12PM_BIT_LSB_IDX         7
63
```

T12 Interrupt를 Interrupt Router로 전달위한 레지스터 설정 : CCU60SR0 레지스터 설정



5-43 Interrupt Sources and Events

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.3733

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.3783

OTGB1[0.15]	OCDS	O	OCDS Trigger Bus 1
SR0	Interrupt Router: SRC_CCU60SR0	O	CCU60 Service Request 0

- SRC 레지스터 항목에서
  - **CCU60SR0** 레지스터 설정 필요
- CCU60 – T12 모듈에서 생성된 Interrupt를 특정 CPU에 연결하기 위해 Interrupt Router (IR) 설정
- **T12 PM Interrupt에 Interrupt 고유 ID 부여 및 CPU0에서 처리하도록 설정**

SRC 레지스터 설정 – CCU60SR0

생성된 인터럽트의 ID부여 및 cpu 할당

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

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

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

2. 사용할 레지스터의 주소 찾기
   - 2개의 CCU6 모듈 중, 0번째 CCU6 모듈 사용하므로 **CCU6"0" (m = 0)**
   - **CCU60SR0** 의 Offset Address = 0x420 + (m * 0x10) = **0x420**

   → CCU60SR0 레지스터 주소 = 0xF0038000 + 0x420 = **0xF0038420**

-	-	Reserved	0406_H-$041C_H$	BE	BE	-	-
SRC_CCU6 mSR0	CCU6 m	CCU6 m Service Request 0 (m = 0-1)	0420_H+ (m × 10_H)	U, SV	SV, P0, P1	3	89 + m*4
SRC_CCU6 mSR1	CCU6 m	CCU6 m Service Request 1 (m = 0-1)	0424_H+ (m × 10_H)	U, SV	SV, P0	3	90 + m*4

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.1572

SRC 레지스터 설정 – CCU60SR0

생성된 인터럽트의 ID부여 및 cpu 할당

3. 레지스터 write 값 결정
   - Interrupt 의 우선순위 및 SW에서 ISR을 고유하게 인식하기 위한 ID 설정을 위해 **SRPN 영역의 값을 0xB 로 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**

CCU60SR0 레지스터 @ 0xF0038420

SRC_CCU6mSR0 (m=0-1)

CCU6 m Service Request 0 (0420_H+m*10_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

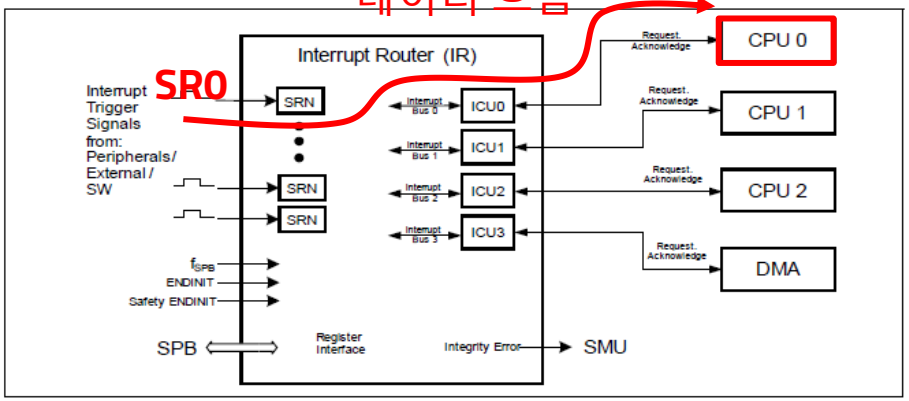


Figure 16-1 Block Diagram of the TC27x Interrupt System

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

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

Lab11: SRC 레지스터 설정 – CCU60SR0

Interrupt 설정

```
150 void initCCU60(void)
151 {
152     // Password Access to unlock SCU_WDTSCON0
153     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
154     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0);    // wait until unlocked
155
156     // Modify Access to clear ENDINIT
157     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) & ~(1 << ENDINIT_BIT_LSB_IDX);
158     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0);    // wait until locked
159
160     CCU60_CLC.U &= ~(1 << DISR_BIT_LSB_IDX);    // enable CCY
161
162     // Password Access to unlock SCU_WDTSCON0
163     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
164     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0);    // wait until unlocked
165
166     // Modify Access to set ENDINIT
167     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
168     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0);    // wait until locked
169
170
171     // CCU60 T12 configurations
172     while((CCU60_CLC.U & (1 << DISS_BIT_LSB_IDX)) != 0);// wait until CCU60 module enabled
173
174     CCU60_TCTR0.U &= ~(0x7 << T12CLK_BIT_LSB_IDX);      // f_T12 = f_CCU6 / prescaler
175     CCU60_TCTR0.U |= 0x2 << T12CLK_BIT_LSB_IDX;         // f_CCU6 = 50 MHz, prescaler = 1024
176     CCU60_TCTR0.U |= 0x1 << T12PRE_BIT_LSB_IDX;         // f_T12 = 48,828 Hz
177
178     CCU60_TCTR0.U &= ~(0x1 << CTM_BIT_LSB_IDX);         // T12 auto reset when period match (PM) occur
179
180
181     CCU60_T12PR.U = 24414 -1;                       // PM interrupt freq. = f_T12 / (T12PR + 1)
182     CCU60_TCTR4.U |= 0x1 << T12STR_BIT_LSB_IDX;         // load T12PR from shadow register
183
184     CCU60_T12.U = 0;                                // clear T12 counter register
185
186
187     // CCU60 T12 PM interrupt setting
188     CCU60_INP.U &= ~(0x3 << INPT12_BIT_LSB_IDX);        // service request output SR0 selected
189     CCU60_IEN.U |= 0x1 << ENT12PM_BIT_LSB_IDX;          // enable T12 PM interrupt
190
191
192     // SRC setting for CCU60
193     SRC_CCU6_CCU60_SR0.U &= ~(0xFF << SRPN_BIT_LSB_IDX);
194     SRC_CCU6_CCU60_SR0.U |= 0x0B << SRPN_BIT_LSB_IDX;   // set priority 0x0B
195
196     SRC_CCU6_CCU60_SR0.U &= ~(0x3 << TOS_BIT_LSB_IDX);  // CPU0 service T12 PM interrupt
197
198     SRC_CCU6_CCU60_SR0.U |= 0x1 << SRE_BIT_LSB_IDX;     // SR0 enabled
199
200
201     // CCU60 T12 counting start
202     CCU60_TCTR4.U = 0x1 << T12RS_BIT_LSB_IDX;           // T12 start counting
203 }
204
```

```
48 // SRC registers
49 #define SRPN_BIT_LSB_IDX            0
50 #define TOS_BIT_LSB_IDX             11
51 #define SRE_BIT_LSB_IDX             10
52
```

Bit Slice Index 정의

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

```
30
31 #include "IfxCcu6_reg.h"
32
33 // Port registers
34 #define PC1_BIT_LSB_IDX            11
35 #define PC2_BIT_LSB_IDX            19
36 #define P1_BIT_LSB_IDX             1
37 #define P2_BIT_LSB_IDX             2
38
39 // SCU registers
40 #define LCK_BIT_LSB_IDX            1
41 #define ENDINIT_BIT_LSB_IDX        0
42 #define EXIS0_BIT_LSB_IDX          4
43 #define FEN0_BIT_LSB_IDX           8
44 #define EIEN0_BIT_LSB_IDX          11
45 #define INP0_BIT_LSB_IDX           12
46 #define IGP0_BIT_LSB_IDX           14
47
48 // SRC registers
49 #define SRPN_BIT_LSB_IDX           0
50 #define TOS_BIT_LSB_IDX            11
51 #define SRE_BIT_LSB_IDX            10
52
53 // CCU60 registers
54 #define DISS_BIT_LSB_IDX           1
55 #define DISR_BIT_LSB_IDX           0
56 #define CTM_BIT_LSB_IDX            7
57 #define T12PRE_BIT_LSB_IDX         3
58 #define T12CLK_BIT_LSB_IDX         0
59 #define T12STR_BIT_LSB_IDX         6
60 #define T12RS_BIT_LSB_IDX          1
61 #define INPT12_BIT_LSB_IDX         10
62 #define ENT12PM_BIT_LSB_IDX        7
63
```

헤더 파일 참조 추가

CCU60 레지스터 bit shift offset index

최종 Timer 준비 코드

- CCU60 모듈 및 Interrupt 사용을 위한 초기화 함수 *initCCU60()*

```
150 void initCCU60(void)
151 {
152     // Password Access to unlock SCU_WDTSCON0
153     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
154     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0);    // wait until unlocked
155
156     // Modify Access to clear ENDINIT
157     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) & ~(1 << ENDINIT_BIT_LSB_IDX);
158     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0);    // wait until locked
159
160     CCU60_CLC.U &= ~(1 << DISR_BIT_LSB_IDX);    // enable CCY
161
162     // Password Access to unlock SCU_WDTSCON0
163     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
164     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0);    // wait until unlocked
165
166     // Modify Access to set ENDINIT
167     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
168     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0);    // wait until locked
169
170
171     // CCU60 T12 configurations
172     while((CCU60_CLC.U & (1 << DISS_BIT_LSB_IDX)) != 0);// wait until CCU60 module enabled
173
174     CCU60_TCTR0.U &= ~(0x7 << T12CLK_BIT_LSB_IDX);      // f_T12 = f_CCU6 / prescaler
175     CCU60_TCTR0.U |= 0x2 << T12CLK_BIT_LSB_IDX;         // f_CCU6 = 50 MHz, prescaler = 1024
176     CCU60_TCTR0.U |= 0x1 << T12PRE_BIT_LSB_IDX;         // f_T12 = 48,828 Hz
177
178     CCU60_TCTR0.U &= ~(0x1 << CTM_BIT_LSB_IDX);         // T12 auto reset when period match (PM) occur
179
180
181     CCU60_T12PR.U = 24414 -1;                       // PM interrupt freq. = f_T12 / (T12PR + 1)
182     CCU60_TCTR4.U |= 0x1 << T12STR_BIT_LSB_IDX;         // load T12PR from shadow register
183
184     CCU60_T12.U = 0;                                // clear T12 counter register
185
186
187     // CCU60 T12 PM interrupt setting
188     CCU60_INP.U &= ~(0x3 << INPT12_BIT_LSB_IDX);        // service request output SR0 selected
189     CCU60_IEN.U |= 0x1 << ENT12PM_BIT_LSB_IDX;          // enable T12 PM interrupt
190
191
192     // SRC setting for CCU60
193     SRC_CCU6_CCU60_SR0.U &= ~(0xFF << SRPN_BIT_LSB_IDX);
194     SRC_CCU6_CCU60_SR0.U |= 0x0B << SRPN_BIT_LSB_IDX;   // set priority 0x0B
195
196     SRC_CCU6_CCU60_SR0.U &= ~(0x3 << TOS_BIT_LSB_IDX);  // CPU0 service T12 PM interrupt
197
198     SRC_CCU6_CCU60_SR0.U |= 0x1 << SRE_BIT_LSB_IDX;     // SR0 enabled
199
200
201     // CCU60 T12 counting start
202     CCU60_TCTR4.U = 0x1 << T12RS_BIT_LSB_IDX;           // T12 start counting
203 }
204
```

SW 프로그래밍
레지스터 주소 확인

- CCU60 모듈 및 Interrupt 사용을 위한 초기화 함수 *initCCU60()*

```
150 void initCCU60(void)
151 {
152     // Password Access to unlock SCU_WDTSCON0
153     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
154     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0);    // wait until unlocked
155
156     // Modify Access to clear ENDINIT
157     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) & ~(1 << ENDINIT_BIT_LSB_IDX);
158     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0);    // wait until locked
159
160     CCU60_CLC.U &= ~(1 << DISR_BIT_LSB_IDX);    // enable CCY
161
162     // Password Access to unlock SCU_WDTSCON0
163     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
164     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0);    // wait until unlocked
165
166     // Modify Access to set ENDINIT
167     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
168     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0);    // wait until locked
169
```

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

- SCU_WDTCPU0CON0

```
351
352 /** \brief  100, CPU WDT Control Register 0 */
353 #define SCU_WDTCPU0_CON0 /*lint --e(923)*/ (*(volatile Ifx_SCU_WDTCPU_CON0*)0xF0036100u)
354
355 /** Alias (User Manual Name) for SCU_WDTCPU0_CON0.
```

- CCU60_CLC

```
102
103 /** \brief  0, Clock Control Register */
104 #define CCU60_CLC /*lint --e(923)*/ (*(volatile Ifx_CCU6_CLC*)0xF0002A00u)
105
106 /** \brief  64, Compare State Modification Register */
```

SW 프로그래밍
레지스터 주소 확인

- CCU60 모듈 및 Interrupt 사용을 위한 초기화 함수 *initCCU60()*

```
171  // CCU60 T12 configurations
172  while((CCU60_CLC.U & (1 << DISS_BIT_LSB_IDX)) != 0);// wait until CCU60 module enabled
173
174  CCU60_TCTR0.U &= ~(0x7 << T12CLK_BIT_LSB_IDX);      // f_T12 = f_CCU6 / prescaler
175  CCU60_TCTR0.U |= 0x2 << T12CLK_BIT_LSB_IDX;         // f_CCU6 = 50 MHz, prescaler = 1024
176  CCU60_TCTR0.U |= 0x1 << T12PRE_BIT_LSB_IDX;         // f_T12 = 48,828 Hz
177
178  CCU60_TCTR0.U &= ~(0x1 << CTM_BIT_LSB_IDX);         // T12 auto reset when period match (PM) occur
179
180
181  CCU60_T12PR.U = 24414 -1;                        // PM interrupt freq. = f_T12 / (T12PR + 1)
182  CCU60_TCTR4.U |= 0x1 << T12STR_BIT_LSB_IDX;         // load T12PR from shadow register
183
184  CCU60_T12.U = 0;                                 // clear T12 counter register
185
186
187  // CCU60 T12 PM interrupt setting
188  CCU60_INP.U &= ~(0x3 << INPT12_BIT_LSB_IDX);        // service request output SR0 selected
189  CCU60_IEN.U |= 0x1 << ENT12PM_BIT_LSB_IDX;          // enable T12 PM interrupt
```

- CCU60_TCTR0

```
195
196  /** \brief  70, Timer Control Register 0 */
197  #define CCU60_TCTR0 /*lint --e(923)*/ (*(volatile Ifx_CCU6_TCTR0*)0xF0002A70u)
198
199  /** \brief  74, Timer Control Register 2 */
```

- CCU60_T12PR

```
186
187  /** \brief  24, Timer 12 Period Register */
188  #define CCU60_T12PR /*lint --e(923)*/ (*(volatile Ifx_CCU6_T12PR*)0xF0002A24u)
189
190  /** \brief  50, Timer T13 Counter Register */
```

- CCU60_TCTR4

```
201
202  /** \brief  78, Timer Control Register 4 */
203  #define CCU60_TCTR4 /*lint --e(923)*/ (*(volatile Ifx_CCU6_TCTR4*)0xF0002A78u)
204
205  /** \brief  84, Trap Control Register */
```

- CCU60_T12

```
176  #define CCU60_PSLR /*lint --e(923)*/ (*(volatile Ifx_CCU6_PSLR*)0xF0002A88u)
177
178  /** \brief  20, Timer T12 Counter Register */
179  #define CCU60_T12 /*lint --e(923)*/ (*(volatile Ifx_CCU6_T12*)0xF0002A20u)
180
181  /** \brief  28, Dead-Time Control Register for Timer12 */
182  #define CCU60_T12DTC /*lint --e(923)*/ (*(volatile Ifx_CCU6_T12DTC*)0xF0002A2
```

- CCU60_INP

```
120
121  /** \brief  AC, Interrupt Node Pointer Register */
122  #define CCU60_INP /*lint --e(923)*/ (*(volatile Ifx_CCU6_INP*)0xF0002AACu)
123
124  /** \brief  A0, Interrupt Status Register */
```

SW 프로그래밍
레지스터 주소 확인

- CCU60 모듈 및 Interrupt 사용을 위한 초기화 함수 *initCCU60()*

```
    // SRC setting for CCU60
    SRC_CCU6_CCU60_SR0.U &= ~(0xFF << SRPN_BIT_LSB_IDX);
    SRC_CCU6_CCU60_SR0.U |= 0x0B << SRPN_BIT_LSB_IDX;    // set priority 0x0B

    SRC_CCU6_CCU60_SR0.U &= ~(0x3 << TOS_BIT_LSB_IDX);   // CPU0 service T12 PM interrupt

    SRC_CCU6_CCU60_SR0.U |= 0x1 << SRE_BIT_LSB_IDX;      // SR0 enabled


    // CCU60 T12 counting start
    CCU60_TCTR4.U = 0x1 << T12RS_BIT_LSB_IDX;            // T12 start counting
}
```

- CCU60_IEN

```
114
115 /** \brief  B0, Interrupt Enable Register */
116 #define CCU60_IEN /*lint --e(923)*/ (*(volatile Ifx_CCU6_IEN*)0xF0002AB0u)
117
118 /** \brief  98, Input Monitoring Register */
119 #define CCU60_TMON /*lint --e(923)*/ (*(volatile Ifx_CCU6_TMON*)0xF0002A98u)
```

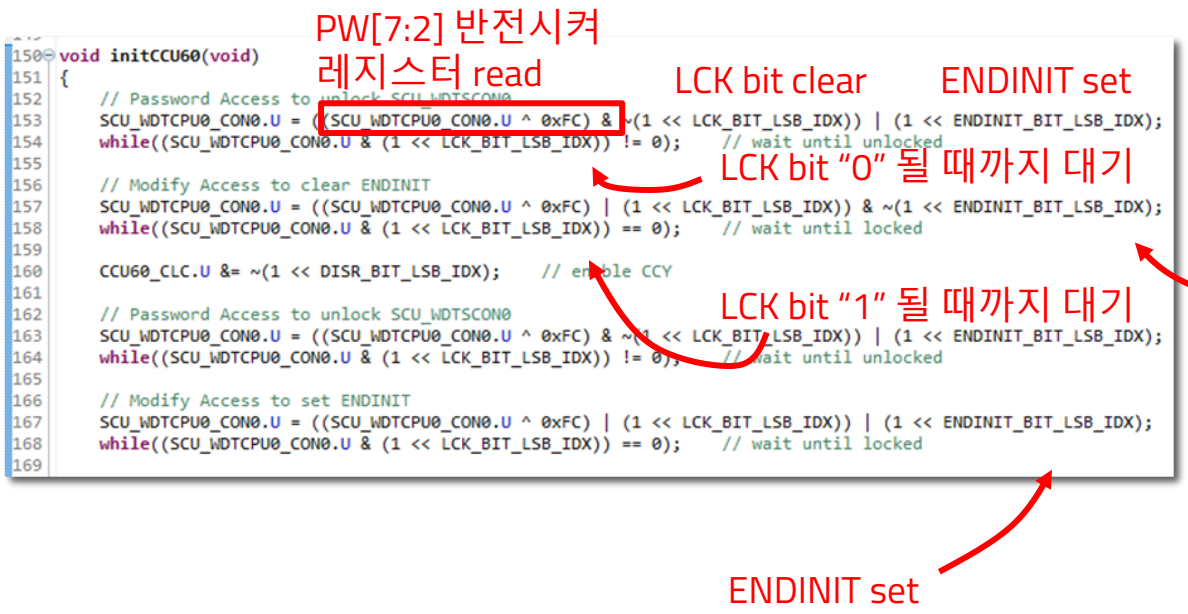
- SRC_CCU60SR0

```
306
307 /** \brief  420, CCU6 Service Request 0 */
308 #define SRC_CCU6_CCU60_SR0 /*lint --e(923)*/ (*(volatile Ifx_SRC_SRCR*)0xF0038420u)
309
310/** Alias (User Manual Name) for SRC_CCU6_CCU60_SR0.
311 * To use register names with standard convension, please use SRC CCU6 CCU60 SR0.
```

SW 프로그래밍

레지스터 보호 해제 및 잠금 과정

- CCU60 모듈 및 Interrupt 사용을 위한 초기화 함수 *initCCU60()*




```
150 void initCCU60(void)
151 {
152     // Password Access to unlock SCU_WDTSCON0
153     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
154     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0);    // wait until unlocked
155
156     // Modify Access to clear ENDINIT
157     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) & ~(1 << ENDINIT_BIT_LSB_IDX);
158     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0);    // wait until locked
159
160     CCU60_CLC.U &= ~(1 << DISR_BIT_LSB_IDX);    // enable CCY
161
162     // Password Access to unlock SCU_WDTSCON0
163     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
164     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0);    // wait until unlocked
165
166     // Modify Access to set ENDINIT
167     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
168     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0);    // wait until locked
169
```

Lock 상태를 해제하기 위한 **Password Access**

Safety ENDINIT clear 위한 **Modify Access**

Lock 상태를 해제하기 위한 **Password Access**

Safety ENDINIT set 위한 **Modify Access**

SW 프로그래밍

- CCU60 모듈 및 Interrupt 사용을 위한 초기화 함수 *initCCU60()*

```
170
171   // CCU60 T12 configurations
172   while((CCU60_CLC.U & (1 << DISS_BIT_LSB_IDX)) != 0);// wait until CCU60 module enabled
173
174   CCU60_TCTR0.U &= ~(0x7 << T12CLK_BIT_LSB_IDX);      // f_T12 = f_CCU6 / prescaler
175   CCU60_TCTR0.U |= 0x2 << T12CLK_BIT_LSB_IDX;         // f_CCU6 = 50 MHz, prescaler = 1024
176   CCU60_TCTR0.U |= 0x1 << T12PRE_BIT_LSB_IDX;         // f_T12 = 48,828 Hz
177
178   CCU60_TCTR0.U &= ~(0x1 << CTM_BIT_LSB_IDX);         // T12 auto reset when period match (PM) occur
179
180
181   CCU60_T12PR.U = 24414 -1;                     // PM interrupt freq. = f_T12 / (T12PR + 1)
182   CCU60_TCTR4.U |= 0x1 << T12STR_BIT_LSB_IDX;         // load T12PR from shadow register
183
184   CCU60_T12.U = 0;                              // clear T12 counter register
185
```

CCU60 - T12 모듈 설정

- CCU60 모듈 enable
- CCU60 모듈에 입력되는 clock 주파수 기반 T12 모듈 clock 주파수 설정
- Counter 가 Period 값과 일치하면 자동으로 counter 리셋 후 증가 시작
- 원하는 PM Interrupt 발생 빈도에 따른 period 값 설정
- T12 counter 레지스터 값 0으로 초기화

SW 프로그래밍

- CCU60 모듈 및 Interrupt 사용을 위한 초기화 함수 *initCCU60()*

```
186
187     // CCU60 T12 PM interrupt setting
188     CCU60_INP.U &= ~(0x3 << INPT12_BIT_LSB_IDX);        // service request output SR0 selected
189     CCU60_IEN.U |= 0x1 << ENT12PM_BIT_LSB_IDX;          // enable T12 PM interrupt
190
191
192     // SRC setting for CCU60
193     SRC_CCU6_CCU60_SR0.U &= ~(0xFF << SRPN_BIT_LSB_IDX);
194     SRC_CCU6_CCU60_SR0.U |= 0x0B << SRPN_BIT_LSB_IDX;   // set priority 0x0B
195
196     SRC_CCU6_CCU60_SR0.U &= ~(0x3 << TOS_BIT_LSB_IDX);  // CPU0 service T12 PM interrupt
197
198     SRC_CCU6_CCU60_SR0.U |= 0x1 << SRE_BIT_LSB_IDX;     // SR0 enabled
199
200
201     // CCU60 T12 counting start
202     CCU60_TCTR4.U = 0x1 << T12RS_BIT_LSB_IDX;           // T12 start counting
203 }
```

CCU60 T12 counter 증가 시작

CCU60 - T12 Interrupt 설정
- Interrupt 로 SR0 사용하도록 설정
- T12 모듈에서 PM Interrupt enable

SRC 에서 CCU60 에 해당하는 Interrupt Router (IR) 설정
- Interrupt의 우선순위 설정
- Interrupt의 처리를 CPU0이 맡도록 설정
- Interrupt가 발생하도록 enable

ISR 함수 설정

- CCU60 내부의 T12 모듈 PM Interrupt 발생 시 호출되는 ISR 함수 작성

```
71
72  //__interrupt(0x0A) __vector_table(0)
73  //void ERU0_ISR(void)
74  //{
75  //    P10_OUT.U ^= 0x1 << P1_BIT_LSB_IDX;  // toggle P10.1 (LED D12 RED)
76  //}
77
78  __interrupt(0x0B) __vector_table(0)
79  void CCU60_T12_ISR(void)
80  {
81      P10_OUT.U ^= 0x1 << P2_BIT_LSB_IDX;  // toggle P10.2 (LED D13 BLUE)
82  }
83
84
```

GPIO Interrupt 실습에서 사용했던 *ERU0_ISR()* 함수 주석처리

→ **0x0B 고유 Interrupt ID를 가지며, CPU0에서 처리**

→ P10.2 핀의 출력 toggle

- 이제 main 함수 while 내에는 아무 코드도 필요하지 않음

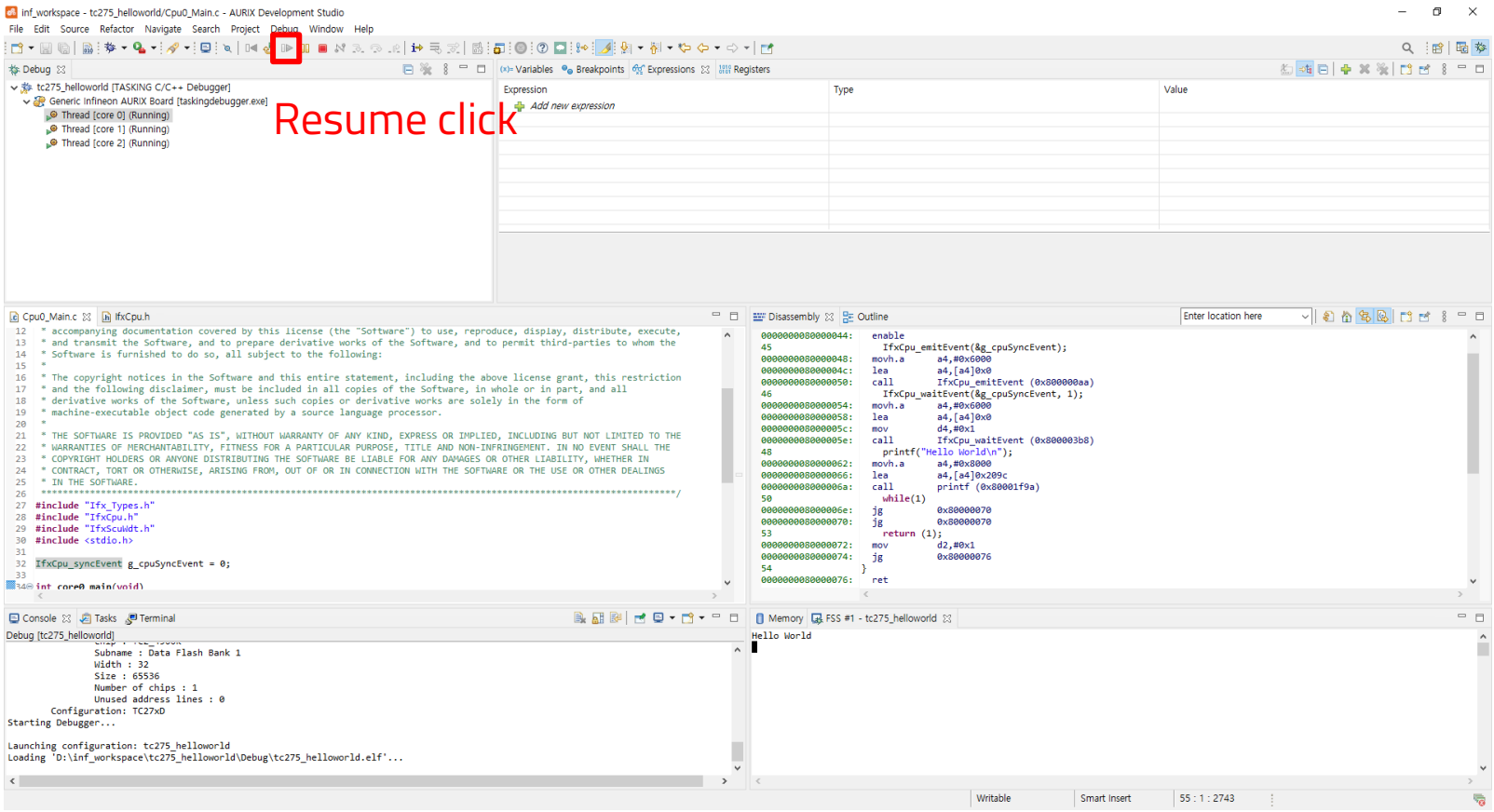
```
84
85  int core0_main(void)
86  {
87      IfxCpu_enableInterrupts();
88
89      /* !!WATCHDOG0 AND SAFETY WATCHDOG ARE DISABLED HERE!!
90       * Enable the watchdogs and service them periodically if it is required
91       */
92      IfxScuWdt_disableCpuWatchdog(IfxScuWdt_getCpuWatchdogPassword());
93      IfxScuWdt_disableSafetyWatchdog(IfxScuWdt_getSafetyWatchdogPassword());
94
95      /* Wait for CPU sync event */
96      IfxCpu_emitEvent(&g_cpuSyncEvent);
97      IfxCpu_waitEvent(&g_cpuSyncEvent, 1);
98
99      //initERU();
100     initCCU60();
101     initLED();
102     //initButton();
103
104     while(1)
105     {
106     }
107     return (1);
108 }
```

→ 초기화 함수 호출

GPIO Interrupt 실습에서 사용했던 *initERU()* *initButton()* 함수 주석처리

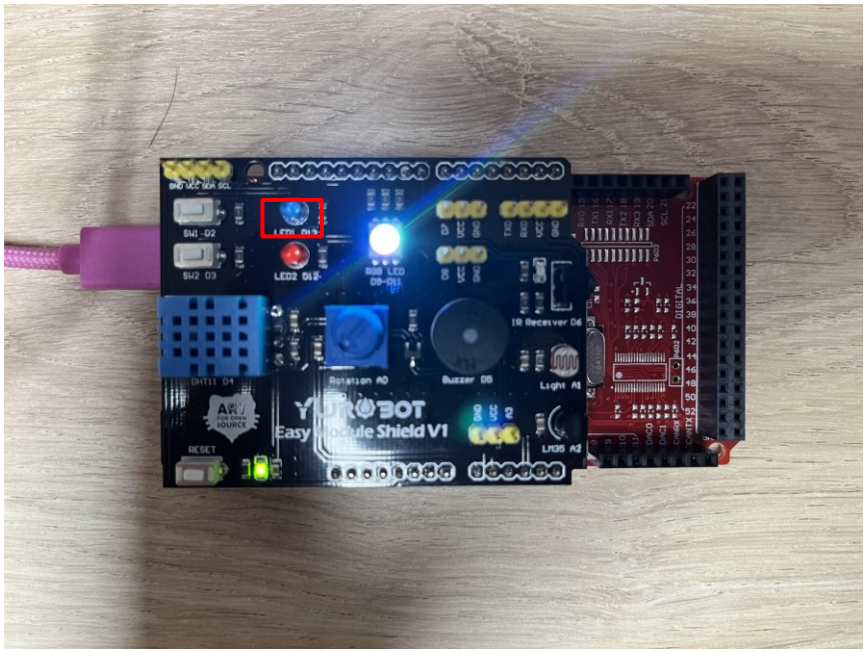
Build 및 Debug

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

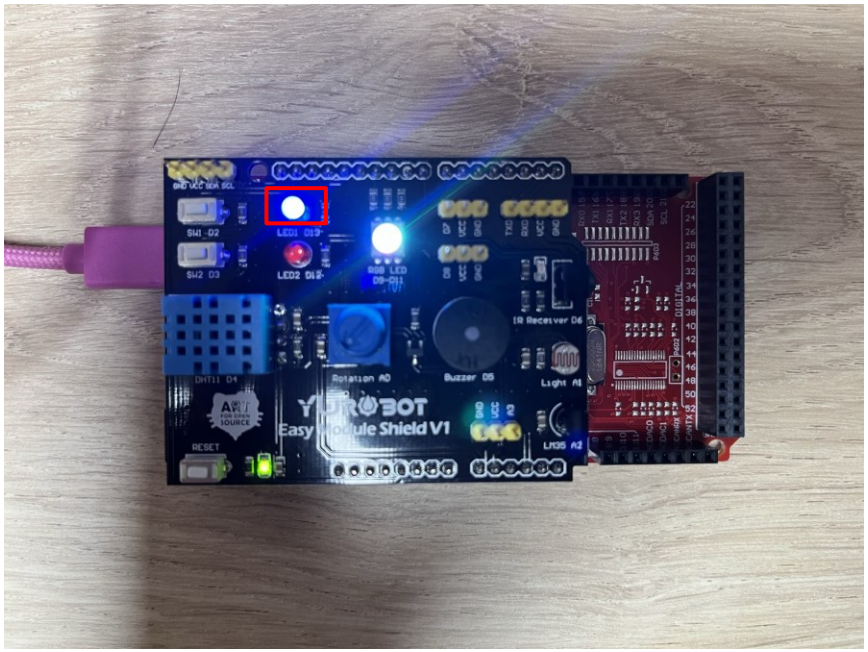


동작 확인

- LED BLUE가 0.5초 주기로 toggle 하는 모습 확인 가능



0.5 sec



보충Lab: 버튼 P02.1핀 누르면 blink 속도증가

: P02.1 포트 입력 활성화 및 외부 인터럽트 지정

```
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();
    initCCU60();
    initLED();
    initButton();

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

P02.1 포트 입력 활성화 및
외부 인터럽트 설정

보충Lab: 버튼 P02.1핀 누르면 blink 속도증가

: P02.1 포트 입력 활성화 및 외부 인터럽트 지정 (Rising 및 Falling모두)

```
void initButton(void)
{
    P02_IOCR0.U &= ~(0x1F << PC1_BIT_LSB_IDX);

    P02_IOCR0.U |= 0x02 << PC1_BIT_LSB_IDX;
}
```

P02.1핀에 대해 Rising 및 Falling 인터럽트 수신 준비

```
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;

    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;
}
```

보충Lab: 버튼 P02.1핀 누르면 blink 속도증가

: 버튼 눌렸을때/땔때 Timer Period 값 다르게 수정

```
__interrupt(0x0A) __vector_table(0)
void ERU0_ISR(void)
{
    // button status check
    if( (P02_IN.U & (0x1 << P1_BIT_LSB_IDX)) == 0 )  // button pushed
    {
        P10_OUT.B.P1 = 0x1;
        CCU60_T12PR.U = 12207 - 1;                      // PM interrupt freq. = f_T12 / (T12PR + 1)
        CCU60_TCTR4.U |= 0x1 << T12STR_BIT_LSB_IDX;     // load T12PR from shadow register
    }
    else                                               // button released
    {
        P10_OUT.B.P1 = 0x0;
        CCU60_T12PR.U = 24414 - 1;                      // PM interrupt freq. = f_T12 / (T12PR + 1)
        CCU60_TCTR4.U |= 0x1 << T12STR_BIT_LSB_IDX;     // load T12PR from shadow register
    }
}


__interrupt(0x0B) __vector_table(0)
void CCU60_T12_ISR(void)
{
    P10_OUT.U ^= 0x1 << P2_BIT_LSB_IDX;  // toggle P10.2 (LED D13 BLUE)
}
```

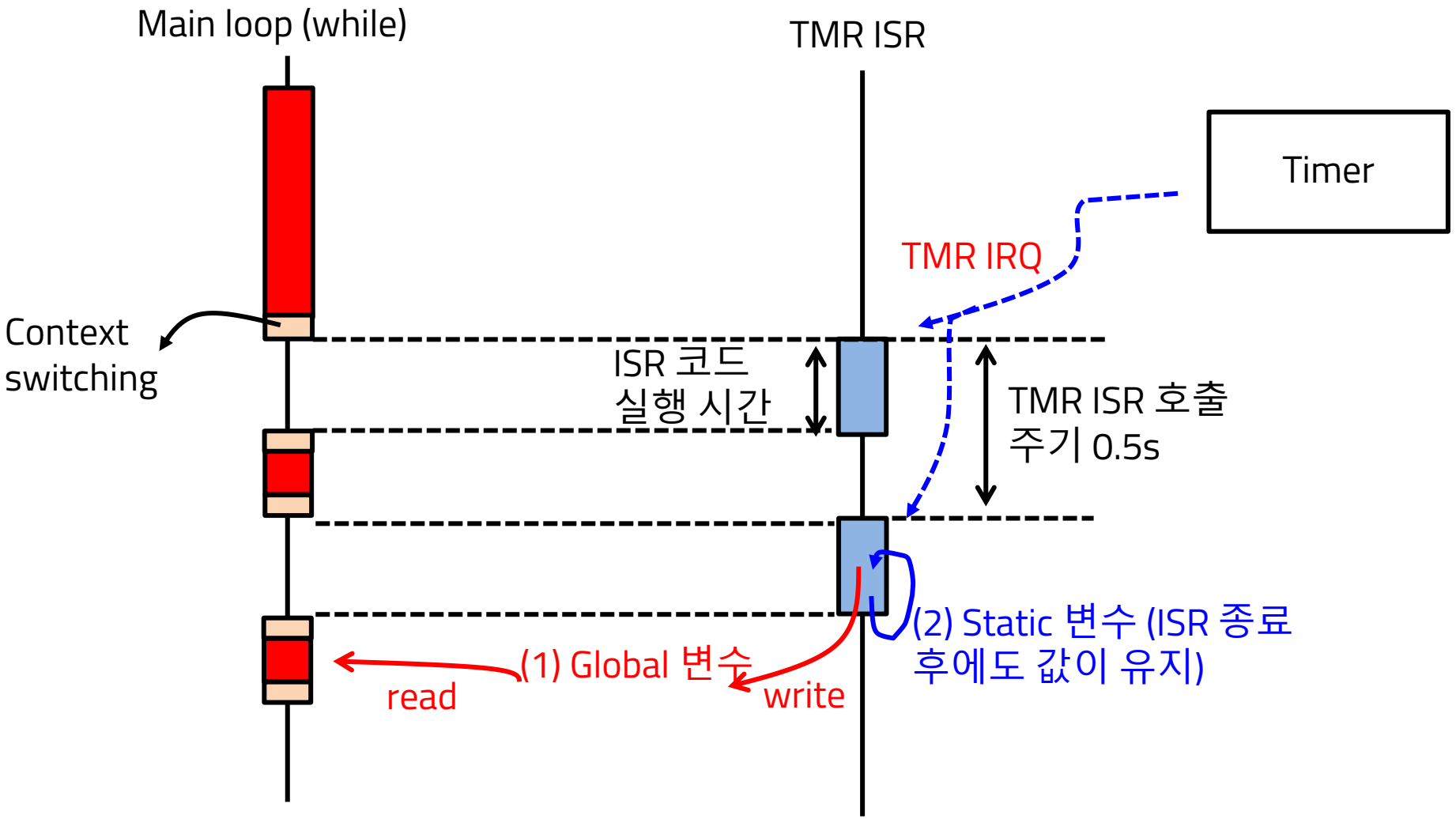
보충 Lab: ISR함수 실행하는 세션을 관리

- ISR함수 호출할때마다 항상 새로 호출하는 것과 같다. (로컬 변수는 다 지워지므로)
- 따라서 static변수를 사용하면, ISR 호출하는 순서관계를 관리할 수 있다.

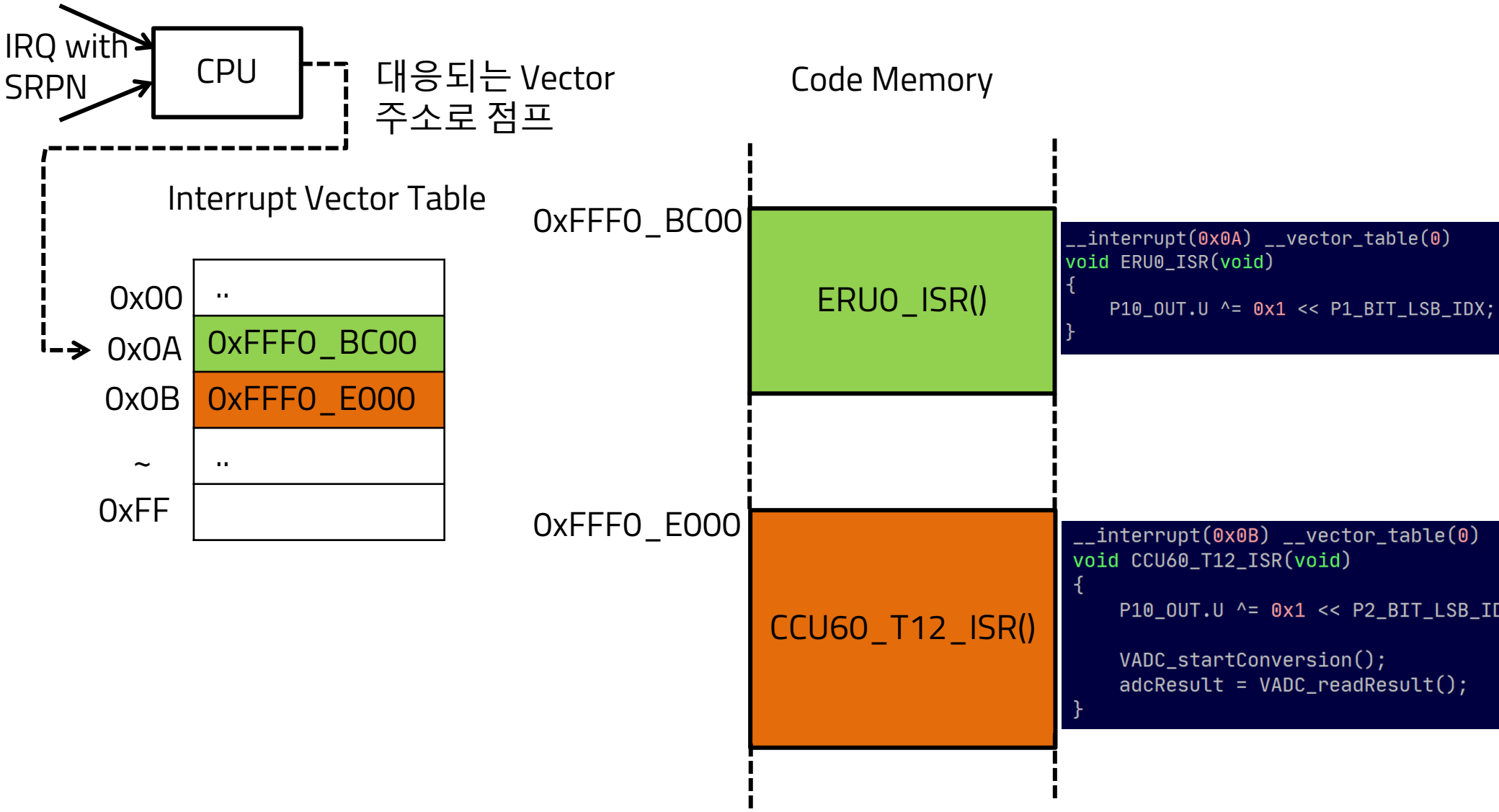
```
__interrupt(0x0A) __vector_table(0)
void ERU0_ISR(void)
{
    static unsigned int session_down_cnt = 0;
    static unsigned int session_up_cnt = 0;

    // button status check
    if( (P02_IN.U & (0x1 << P1_BIT_LSB_IDX)) == 0 )  // button pushed
    {
        P10_OUT.B.P1 = 0x1;
        CCU60_T12PR.U = 12207 - 1;                    // PM interrupt freq. = f_T12 / (T12PR + 1)
        CCU60_TCTR4.U |= 0x1 << T12STR_BIT_LSB_IDX;   // load T12PR from shadow register
        session_down_cnt++;
    }
    else                                              // button released
    {
        P10_OUT.B.P1 = 0x0;
        CCU60_T12PR.U = 24414 - 1;                    // PM interrupt freq. = f_T12 / (T12PR + 1)
        CCU60_TCTR4.U |= 0x1 << T12STR_BIT_LSB_IDX;   // load T12PR from shadow register
        session_down_cnt++;
    }

    if( session_down_cnt % 4 == 0 ) {
        P10_OUT.U ^= 0x1 << P2_BIT_LSB_IDX;  // toggle P10.2 (LED D13 BLUE)
    }
}


__interrupt(0x0B) __vector_table(0)
void CCU60_T12_ISR(void)
{
    //P10_OUT.U ^= 0x1 << P2_BIT_LSB_IDX;  // toggle P10.2 (LED D13 BLUE)
}
```

ISR함수 호출할 때마다의 기록을 유지 하기 위해 Global 변수를 사용할 수도 있으나 함수내로 제한하기 위해서 static을 사용

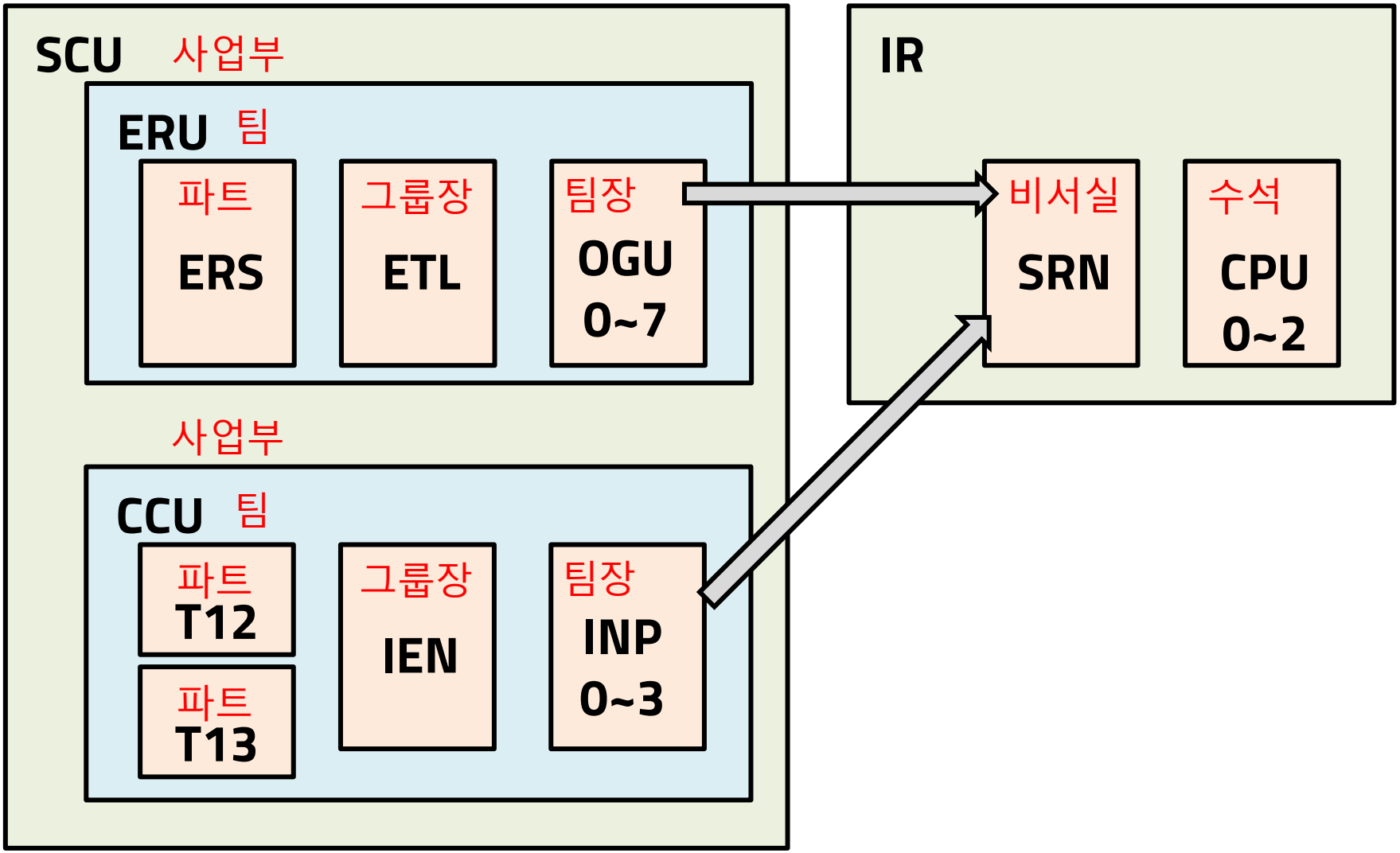
Main loop와 ISR의 소통



Interrupt Vector Table로부터 ISR을 호출하는 원리



TC275 Interrupt 모듈 계층 구조



감사합니다. 휴식~~