

임베디드 기반 SW 개발 프로젝트

AURIX TC275 보드와 초음파 센서 사용

현대자동차 입문교육
박대진 교수

실습 목표

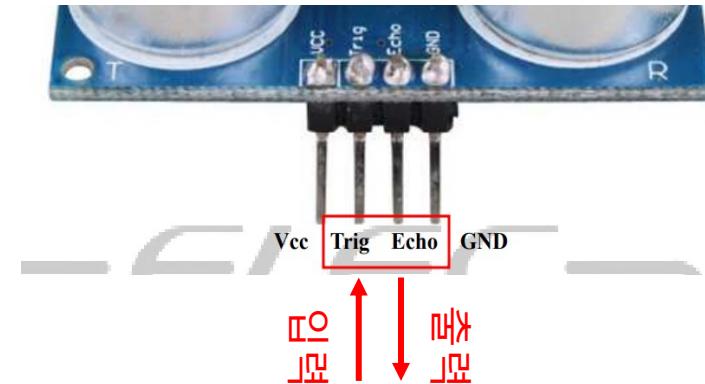
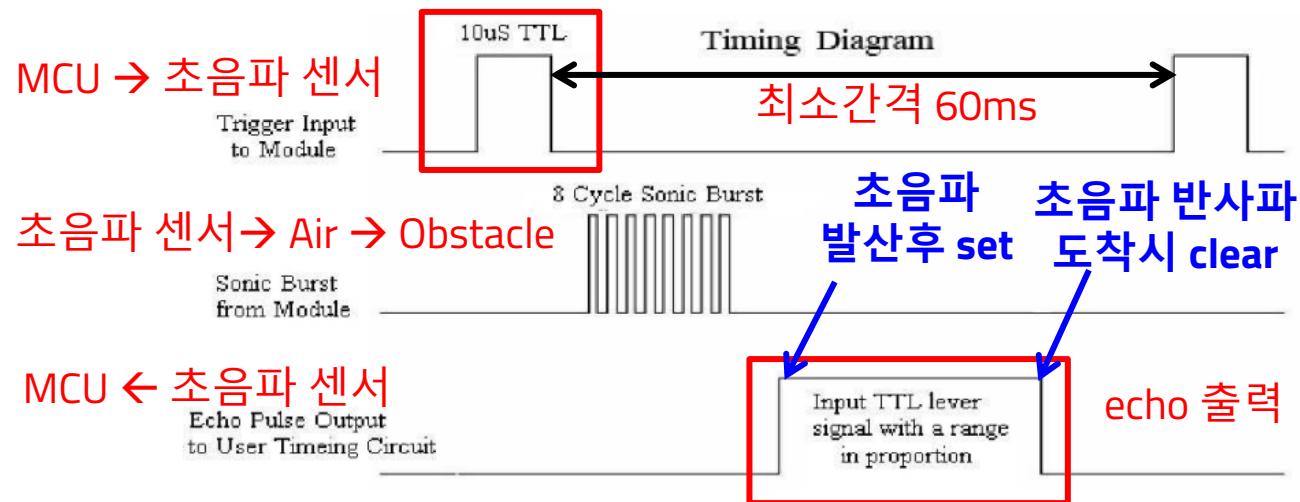
- 초음파 센서의 데이터 시트를 읽고 동작 방식을 파악한 뒤, 보드의 CCU6 하드웨어 2개와 GPIO Interrupt를 모두 사용해서 초음파 센서로 거리를 측정해본다.
 - CCU6 타이머 회로 설정하여 초음파 센서 구동하는 10us 간격 trigger펄스 생성
 - 초음파 센서로부터 입력되는 echo 신호 HIGH 시간 측정위해 ERU dual-edge Interrupt, CCU6 타이머 사용
 - 측정한 echo 신호 HIGH 시간으로 초음파 센서와 물체와의 거리 계산

HC-SR04 초음파 센서 데이터 시트 참고 (HCSR04.pdf)

초음파 센서 동작 원리 파악

초음파 센서 거리 측정 원리

: 물체로부터 반사되는 초음파 사용

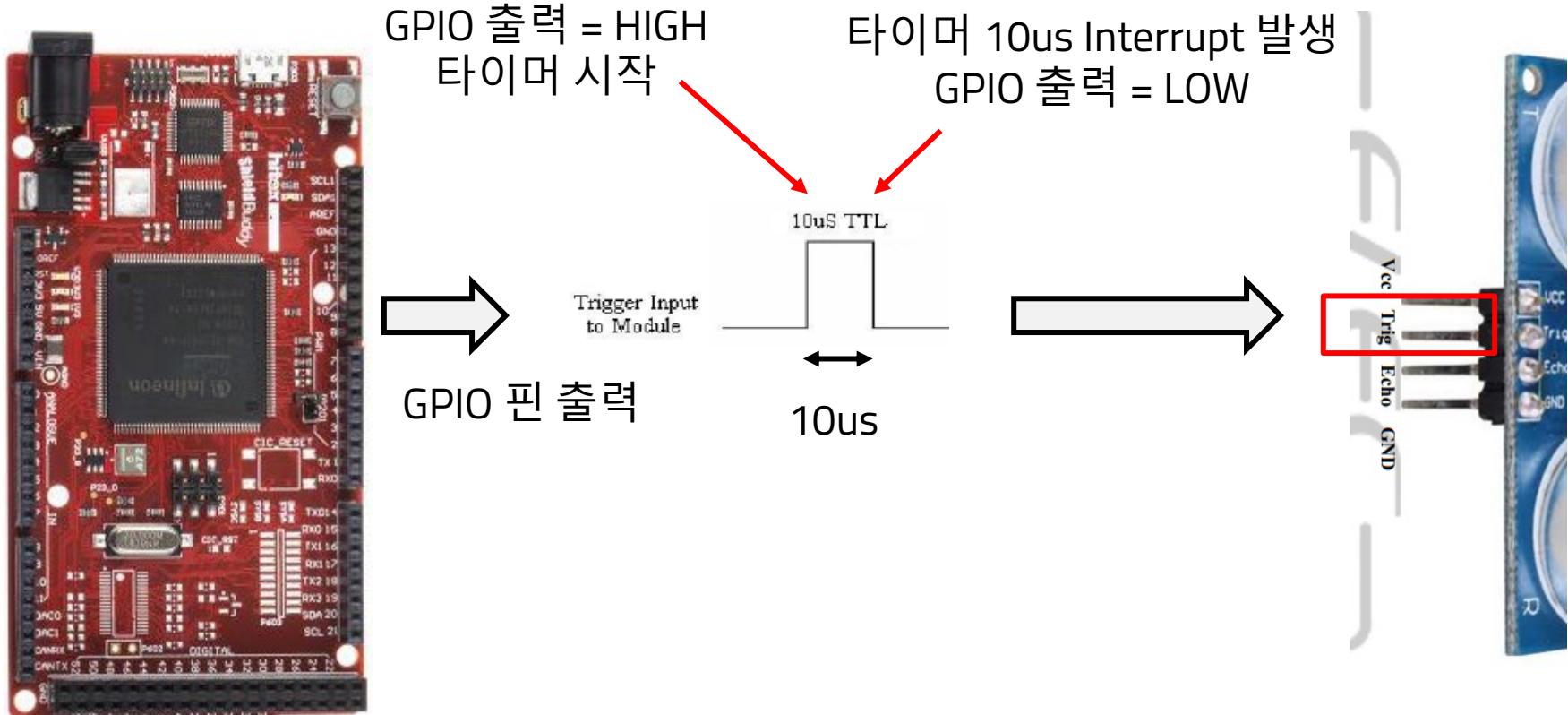


1. 초음파 센서가 물체와의 거리를 측정하려면 초음파 센서의 Trigger 핀으로 10 μs 길이의 펄스 입력
2. 초음파 센서는 초음파를 발사하여 되돌아오기까지의 시간동안 Echo 핀으로 HIGH 신호 출력
3. Echo 신호가 HIGH로 유지되어 있는 시간을 (μs 단위) 측정하면 물체와의 거리를 계산 가능 ($\mu\text{s} / 58 = \text{centimeter}$)

초음파 센서 거리 측정 기능 구현

: 10us trigger 펄스 생성하는 타이머 CCU6

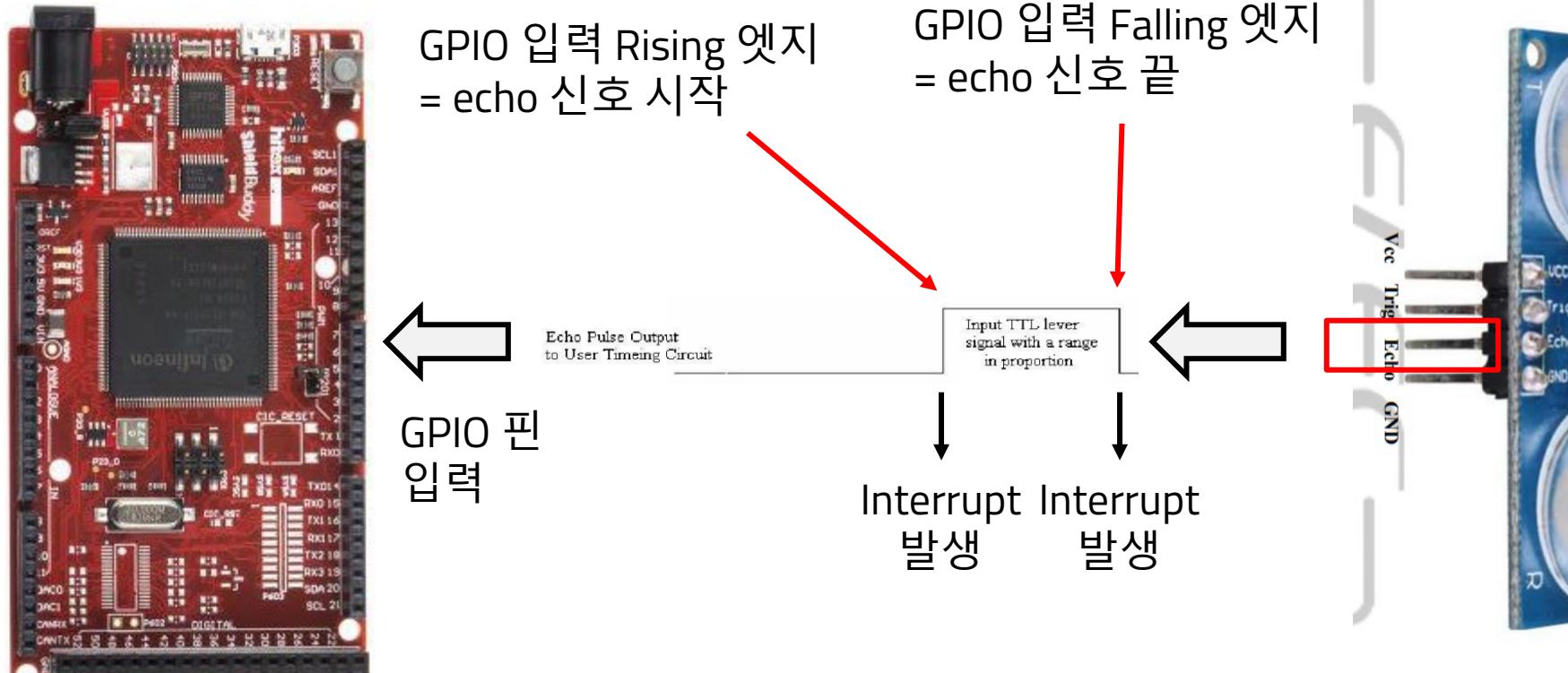
- 초음파 센서의 측정을 시작하려면 HIGH 펄스를 10us 동안 유지해야 함
→ CCU6 모듈의 타이머 기능 사용, GPIO 핀의 출력이 HIGH 되는 시점부터 타이머 카운팅을 시작, 10us가 되는 순간 Interrupt를 발생시켜 GPIO핀의 출력을 LOW로 변경



초음파 센서 거리 측정 기능 구현

:echo 신호 수신을 check하는 ERU Interrupt

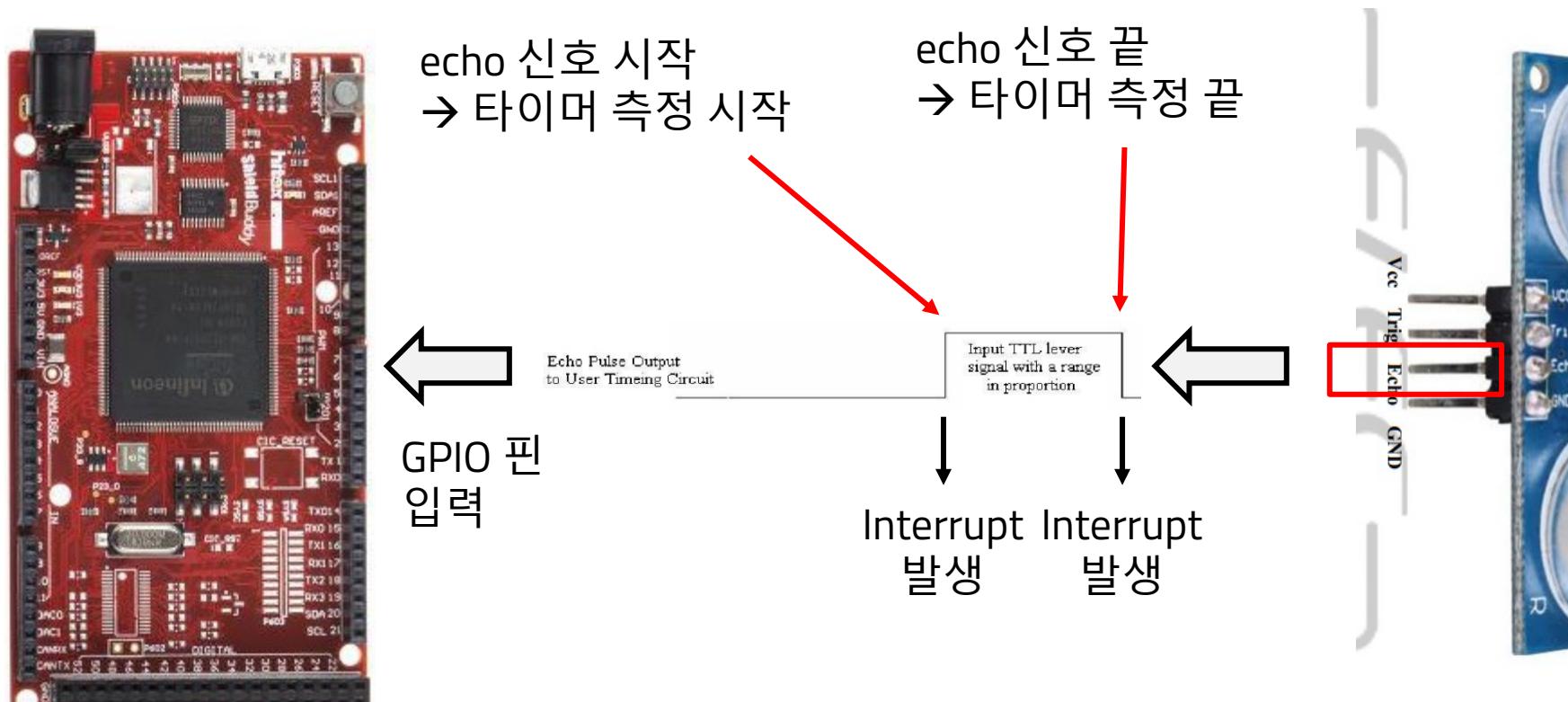
- 초음파 센서는 거리를 echo 신호의 HIGH 길이로 표현
→ echo 신호 HIGH 값이 시작되는 지점, LOW로 변하며 끝나는 지점 인식 필요
→ GPIO 핀으로 입력되는 echo 신호의 rising 엣지, falling 엣지 모두를 ERU Interrupt로 인식



초음파 센서 거리 측정 가능 구현

:echo 신호 길이를 측정하는 타이머 CCU6

- echo 신호를 CCU6 타이머로 카운팅하여 echo 신호가 종료된 시점의 값으로 길이 측정
→ echo 신호 시작 지점에서 CCU6 타이머 카운팅 시작
→ echo 신호 끝 지점에서 CCU6 타이머 카운팅 종료 및 카운터 값으로 시간 계산



초음파 센서 거리 측정 기능 구현

:추가로 사용하는 하드웨어 모듈 CCU6

- 지금까지의 실습에서 사용한 CCU6 타이머 모듈은 **CCU60** (0번) 1개
- 초음파 센서 사용을 위해서는 2곳에서 타이머 기능 필요
 - 1. Trigger 펄스 신호 10us 길이 측정
 - 2. Echo 신호 HIGH 길이 측정
- CCU61 (1번) 모듈을 추가로 사용**
 - CCU60과 동일하게
CCU61 내부 T12 사용

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

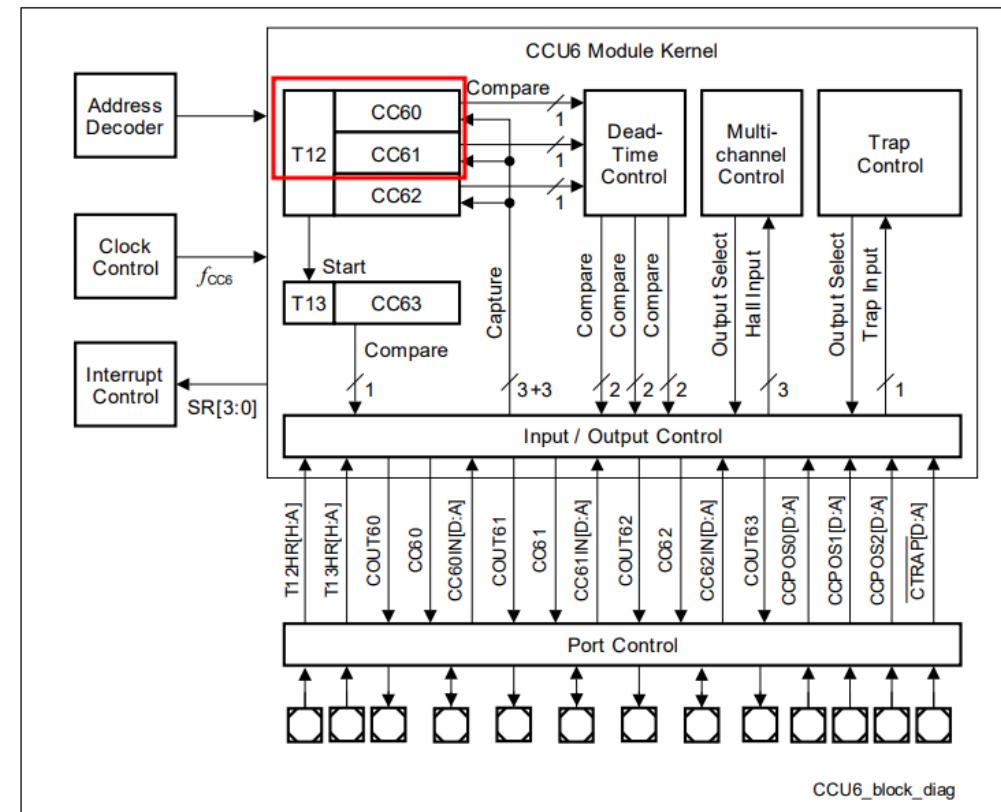


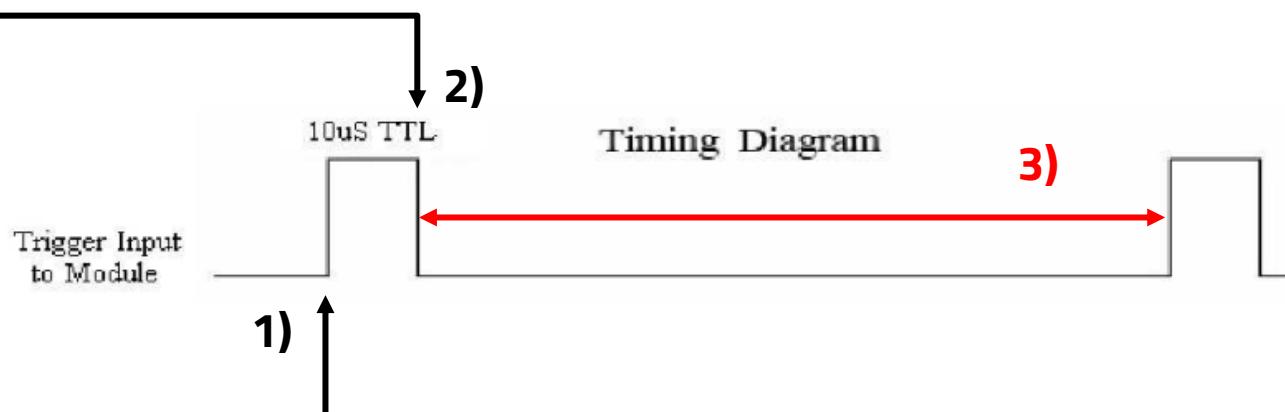
Figure 26-1 CCU6 Block Diagram

MCU에서 초음파센서 실행 위한 과정 정리

:Trigger 10us 펄스 생성

- 초음파 센서 1회 거리 측정을 시작하도록 하는 HIGH 펄스를 10us 동안 출력하기 위해

- 1) GPIO 출력을 HIGH로 설정하는 동시에 CCU60 모듈의 T12 타이머 카운터 증가 시작
- 2) CCU60 모듈 T12 카운터 값이 10us에 해당하는 값에 도달하는 Period Match Interrupt가 발생하면 GPIO 핀 출력을 LOW로 설정
- 3) CCU60의 T12 타이머는 다음 펄스 생성 전까지 reset 상태 유지 (1번만 카운팅 수행)
 - 이전의 CCU60 레지스터 설정에서 수정 필요

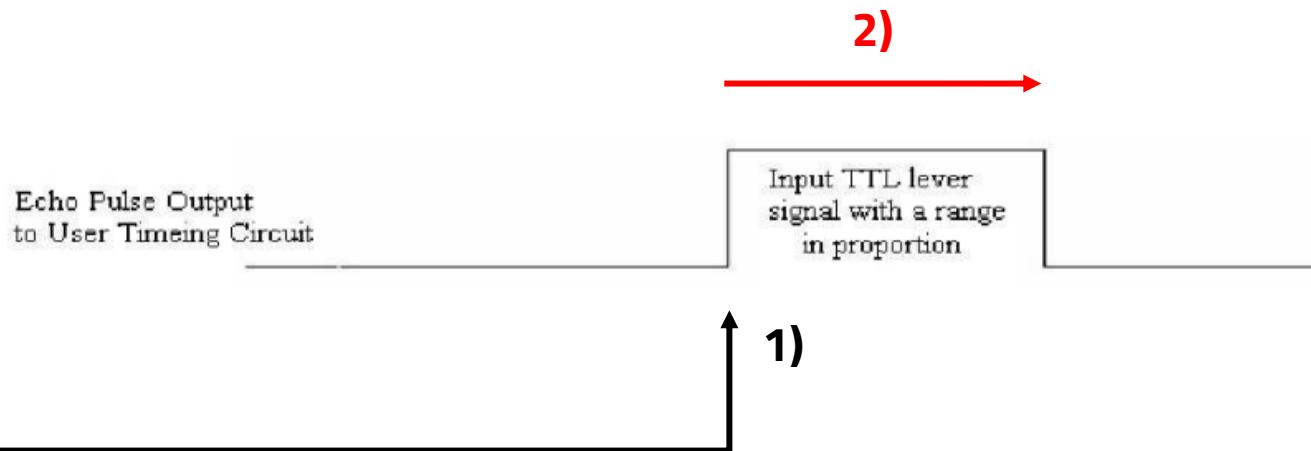


MCU에서 초음파센서 실행 위한 과정 정리

:Echo 신호 입력 시작 인식

2. 초음파 센서 거리 측정이 시작된 것을 의미하는 echo 신호 시작을 인식하고, 길이를 측정하는 타이머 카운팅 시작을 위해

- 1) GPIO 입력이 LOW에서 HIGH로 바뀌는 Rising 엣지에서 ERU Interrupt 발생
- 2) CCU61 모듈 T12 카운터 증가 시작

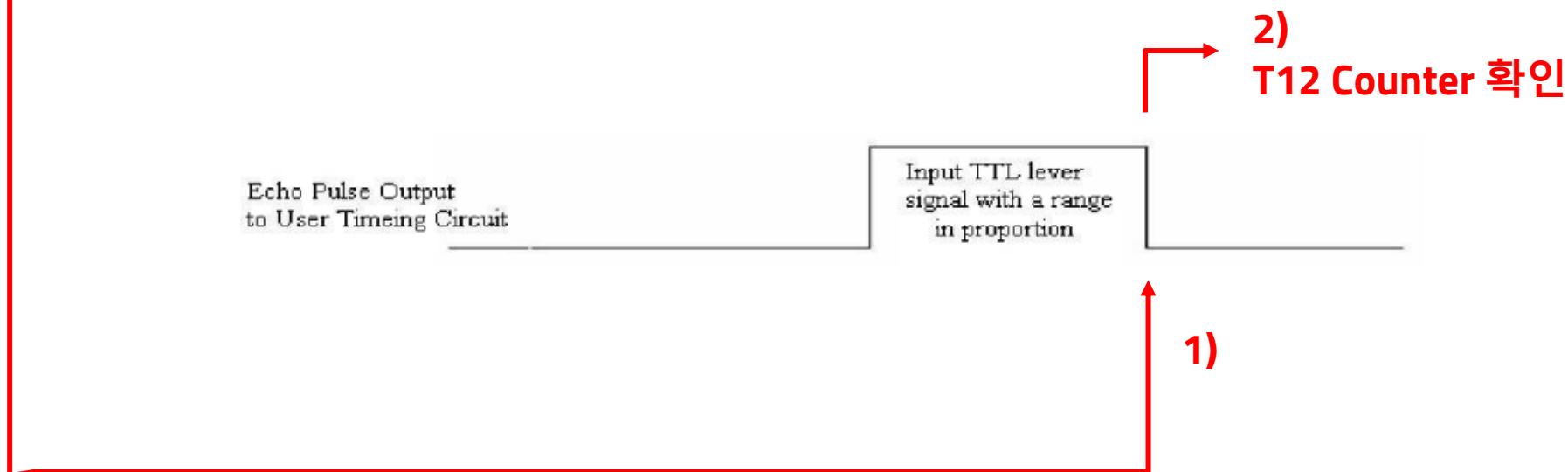


MCU에서 초음파센서 실행 위한 과정 정리

:Echo 신호 입력 종료 인식

3. 초음파 센서 거리 측정이 종료된 것을 의미하는 echo 신호 종료를 인식하고, 타이머 카운터 값을 확인하기 위해

- 1) GPIO 입력이 HIGH에서 LOW로 바뀌는 Falling 엣지에서 ERU Interrupt 발생 (Rising, Falling 모든 엣지에서 ERU Interrupt 발생)
 - 이전의 ERU 레지스터 설정에서 수정 필요
- 2) CCU61 모듈 T12 카운터 정지 및 T12 카운터 값 확인하여 거리 계산



HC-SR04 초음파 센서 데이터 시트 참고 (HCSR04.pdf)

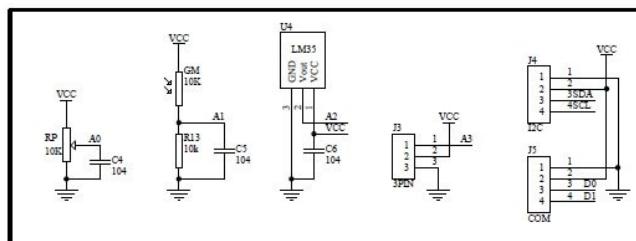
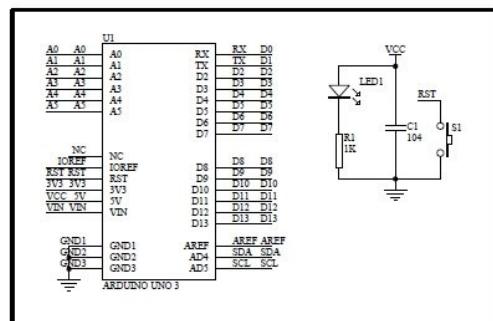
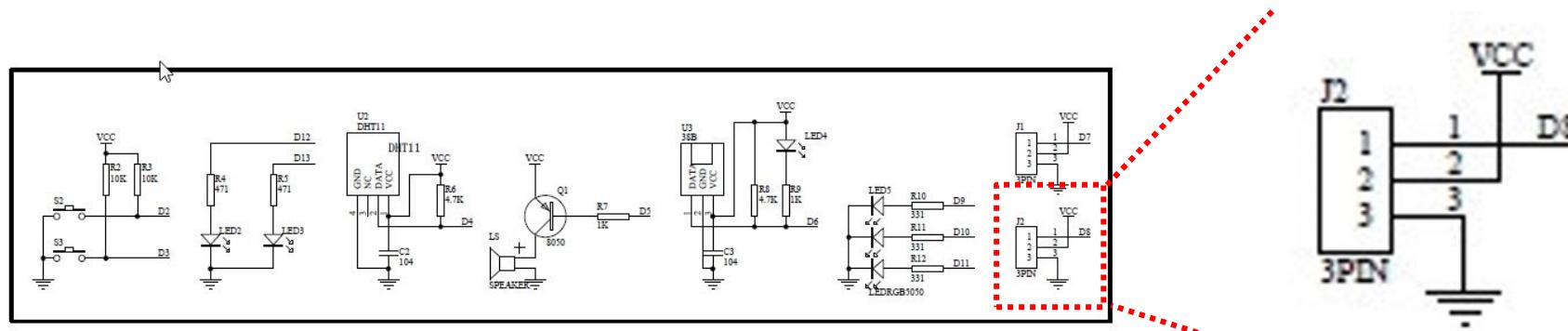
초음파 센서와 보드 연결

TC275 보드와 초음파 센서 연결

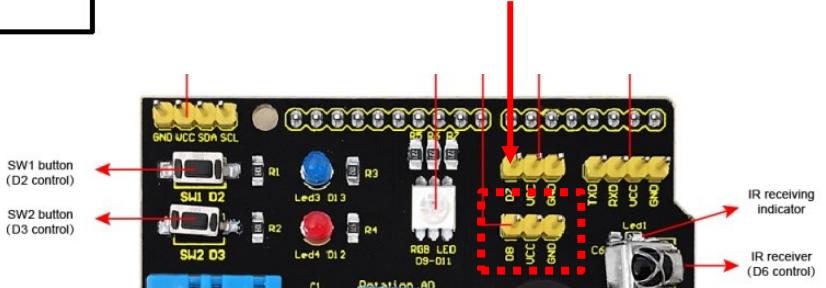
:확장 보드의 핀과 초음파 센서 연결 – Trigger 핀

- 초음파 센서 구동을 위해 필요한 핀은 2개 (Trigger, Echo)

- Trigger 핀과 연결되려면 TC275 MCU에서 **GPIO 출력**이 가능해야 함

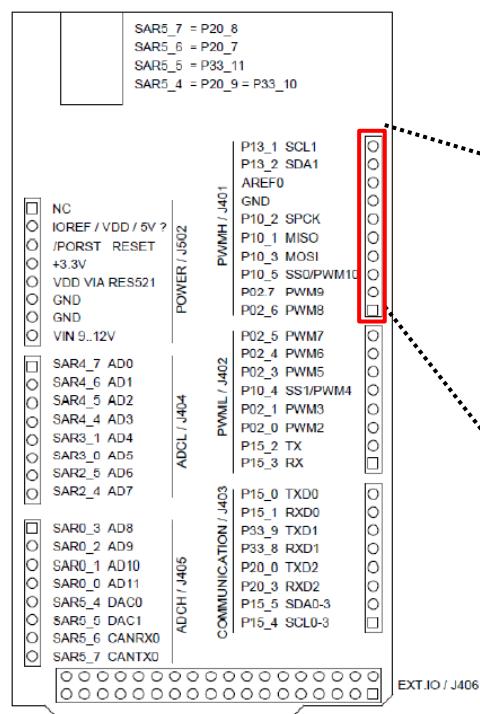


TC275 MCU에서 GPIO 출력
기능을 사용하기 위해
- 확장 보드의 D8 핀을 선택

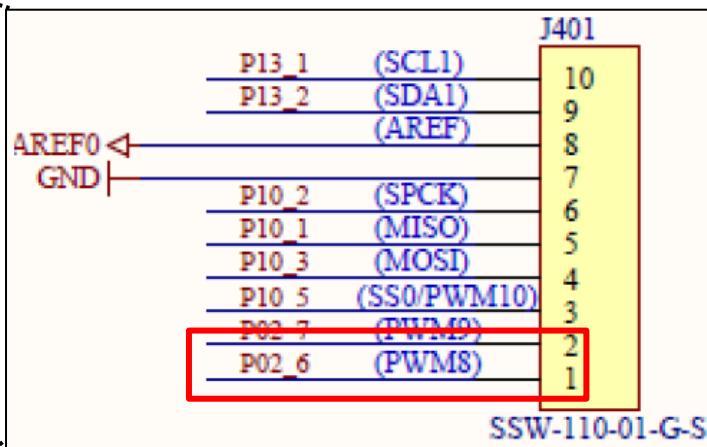
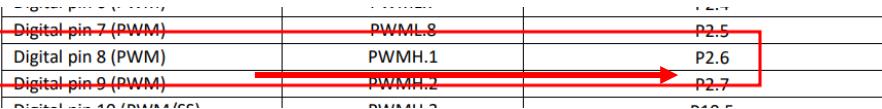


Pin Map @ TC275 보드

→ 그렇다면 확장 보드의 핀 D8은 TC275 보드의 어느 핀에 연결?



Infineon-ShieldBuddy_TC275 -UM-v02_08-EN.pdf p.44



TC275 MCU I/O 중,
Port 02 pin 6 (P02.6)
와 연결됨

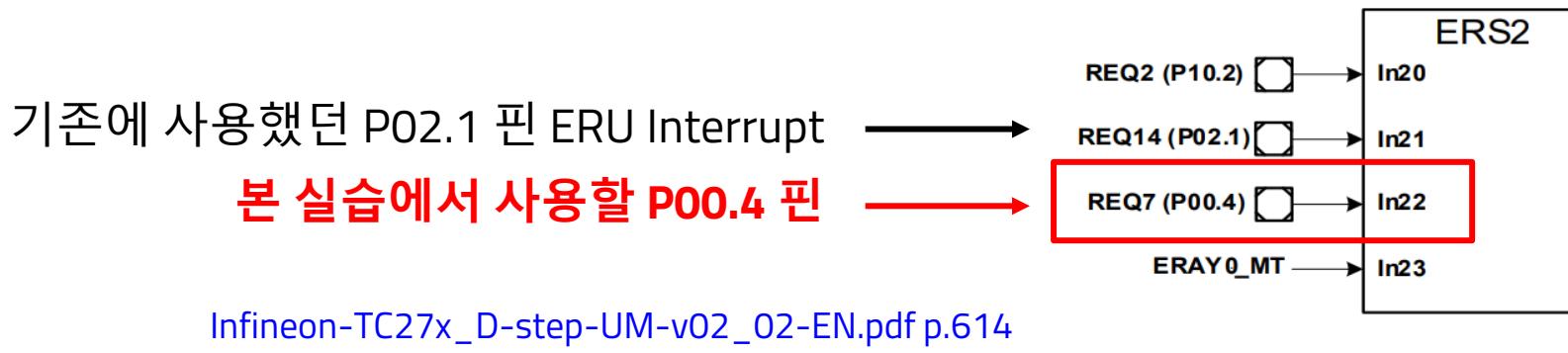
Infineon-ShieldBuddy_TC275 -UM-v02_08-EN.pdf p.42

Infineon-ShieldBuddy_TC275 -UM-v02_08-EN.pdf p.46

TC275 보드와 초음파 센서 연결

: 확장 보드의 핀과 초음파 센서 연결 – Echo 핀

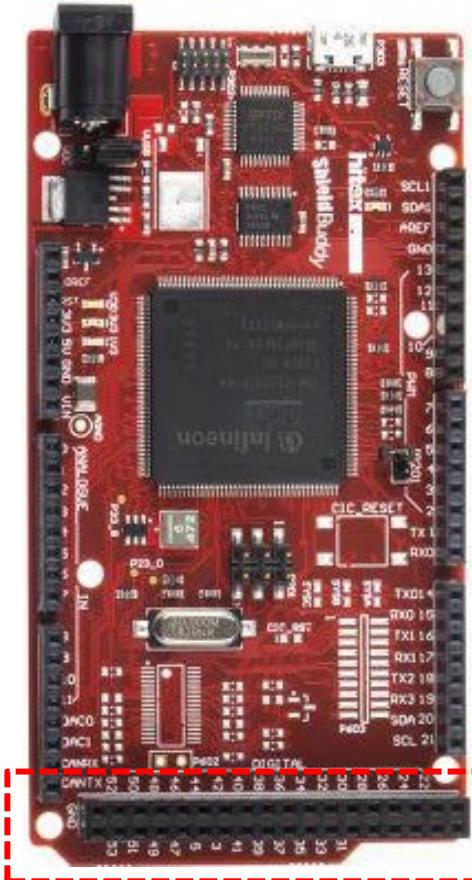
- 초음파 센서 구동을 위해 필요한 핀은 **2개 (Trigger, Echo)**
2. Echo 핀과 연결되려면 TC275 MCU에서 **GPIO 입력 및 ERU Interrupt 설정이 가능해야 함**
- 앞서 [Push 버튼을 사용한 GPIO 입력 ERU Interrupt 발생] 예제에서 사용한 ERU 설정 코드를 최대한 재사용하기 위해 ERU2 부분을 데이터 시트에서 확인



- P02.1 를 ERU로 연결했던 레지스터 설정들을 P00.4 가 ERU에 연결되도록 변경해주면 됨
- **TC275 보드의 핀 P00.4 (Port 00, 핀 4) 를 초음파 센서의 Echo 신호를 수신하는 핀으로 사용**

Pin Map @ TC275 보드

→ 그렇다면 TC275 보드의 P00.4 핀은 TC275 보드 어디에 위치?



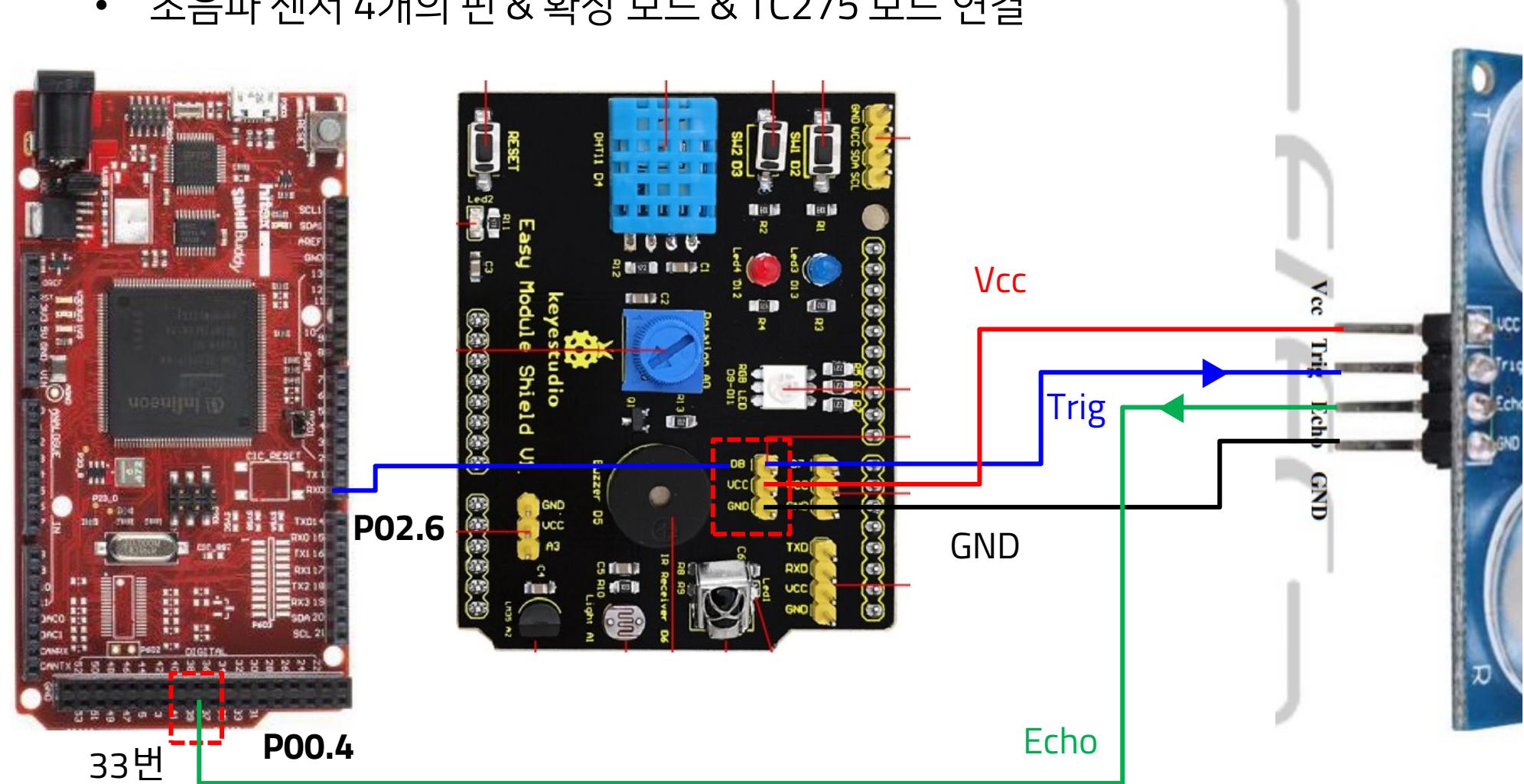
GND	GND
ASC2RX,ASC3TX P33_5 P15_8	CANTXIO P11_12 , P20_10
PIN50 P33_4 , P21_2	PIN52 P02_8 , P13_3
	PIN49 P21_0
PIN48 PIN48	PIN47 P11_6
PIN46 P11_3	PIN45 P11_2
PIN44 P33_3	PIN43 P11_11
PIN42 P11_10	PIN41 P11_9
PIN40 P33_0	PIN39 P00_7
PIN38 P33_1	PIN37 P00_6
PIN36 P33_2	PIN35 P00_5
PIN34 P00_12	PIN33 P00_4
PIN32 P00_11	PIN31 P00_3
PIN30 P00_10	PIN29 P00_2
PIN28 P00_9	PIN27 P00_1
PIN26 P00_8	PIN25 P00_0
PIN24 P15_6	PIN23 P14_1
PIN22 P14_0	VDD / 5V ?
VDD / 5V ?	

TC275 보드의 33번째 핀
(숫자 33 적혀 있음)
→ P00.4

TC275 보드와 초음파 센서 연결

:확장 보드의 핀과 초음파 센서 연결 - 모든 핀 연결한 모습

- 초음파 센서 4개의 핀 & 확장 보드 & TC275 보드 연결



초음파 센서 핀 입출력 연결위한 GPIO 레지스터 설정

GPIO 레지스터 출력 설정

:초음파 센서 Trigger와 연결

- 초음파 센서의 Trigger 핀에 10us 펄스를 보내주기 위해 P02.6 핀을 GPIO 출력으로 사용하기로 결정
- P02.6 핀의 출력 모드 동작을 위해서는 어떤 레지스터 설정이 필요?
- GPIO Port 02 레지스터 항목에서
 - P02_OUT**
 - P02_IOCR4**
- 2가지 레지스터에 설정이 필요함

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.1155

Port Pin	I/O	Pin Functionality	Associated Reg./ I/O Line	Port I/O Control Select.	
				Reg./Bit Field	Value
P02.6	I	General-purpose input	P02_IN.P6	P02_IOCR4 PC6	0XXXX _B
		GTM input	TIN6		
		QSPI3 input	MTSR3A		
		SENT input	SENT2C		
		CCU60 input	CC60INC		
		CCU60 input	CCPOS0A		
		CCU61 input	T12HRB		
		GPT120 input	T3INA		
		DSADC input	DSDIN4B		
		CIF input	CIFD6		
P02.6	O	General-purpose output	P02_OUT.P6	1X000 _B	1X001 _B
		GTM output	TOUT6		
		PSI5-S output	PSISTX		
		QSPI3 output	MTSR3		
		PSI5 output	PSITX1		
		VADC output	VADCEMUX00		
		Reserved	-		
		CCU60 output	CC60		

GPIO 레지스터 출력 설정

:Port 02 레지스터 주소 확인

- 레지스터 설정을 위해 레지스터가 위치한 주소 파악 필요

1. GPIO (Ports) 레지스터 영역 찾기

[Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.227](#)

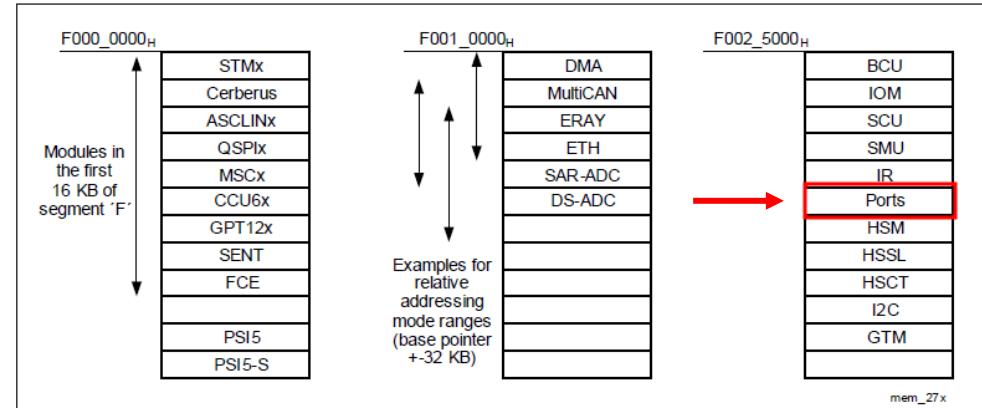


Figure 3-1 Segment F Structure

2. 실습에서 사용하는 Port 02에 해당하는 레지스터 영역 찾기

- 시작 주소 (Base address)
- = **0xF003A200**

HJ...		
Port 00	F003 A000 _H - F003 A0FF _H	256 byte	access	access
Port 01	F003 A100 _H - F003 A1FF _H	256 byte	access	access
Port 02	F003 A200 _H - F003 A2FF _H	256 byte	access	access

[Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.231](#)

GPIO 레지스터 출력 설정

: P02_IOCR4 Direction Reg Addr 계산

3. 사용할 특정 레지스터의 주소 찾기

- P02_IOCR4
- Offset Address = 0x0014
- 즉, Base Address로부터 0x0014만큼 떨어져 있음

→ P02_IOCR4 레지스터 주소

→ = [P02 Base Addr] + [P02_IOCR4 Offset Addr]

→ = 0xF003A200 + 0x0014 = **0xF003A214**

사용할 Port 번호 n = 02

→ Pn = P02

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.1074

ID	Register	Module Identification Register	0008 _H	U, SV	BE	Application Reset	Page 13-1 3
Pn_IOCR0	Port n Input/Output Control Register 0	0010 _H	U, SV	U, SV, P	Application Reset	Page 13-1 4	
Pn_IOCR4	Port n Input/Output Control Register 4	0014_H	U, SV	U, SV, P	Application Reset	Page 13-1 4	
Pn_IOCR8	Port n Input/Output Control Register 8	0018 _H	U, SV	U, SV, P	Application Reset	Page 13-1 4	



왜 P02 이 아니라 Pn 으로 표기?

- MCU에서 사용할 수 있는 Port는 여러 개 존재
 - ✓ Port 1, Port 2, ... Port 10, ...
- 각 Port들이 동일한 레지스터 구성을 가지기 때문!
- 모든 Port에 대한 레지스터는 동일한 map을 참고하기 위해 Pn (n 자리에는 Port 번호) 사용

GPIO 레지스터 출력 값 설정

: P02_OUT Data Reg Addr 계산

3. 사용할 특정 레지스터의 주소 찾기

- P02_OUT

- Offset Address = 0x0000

- 즉, Base Address로부터 0x0000만큼 떨어져 있음

→ P02_OUT 레지스터 주소

→ = [P02 Base Addr] + [P02_OUT Offset Addr]

→ = 0xF003A200 + 0x0000 = **0xF003A200**

사용할 Port 번호 n = 02

→ Pn = P02

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.1074

Table 13-4 Registers Overview

Register Short Name	Register Long Name	Offset Address	Access Mode		Reset	Desc. see
			Read	Write		
Pn_OUT	Port n Output Register	0000 _H	U, SV	U, SV, P	Application Reset	Page 13-38
Pn_QMR	Port n Output Modification Register	0004 _H	U, SV	U, SV, P	Application Reset	Page 13-39
ID	Module Identification Register	0008 _H	U, SV	BE	Application Reset	Page 13-13
Pn_IDCR0	Port n Input/Output Control Register 0	0010 _H	U, SV	U, SV, P	Application Reset	Page 13-14
Pn_IDCR4	Port n Input/Output Control Register 4	0014 _H	U, SV	U, SV, P	Application Reset	Page 13-14

왜 P02 이 아니라 Pn 으로 표기?

- MCU에서 사용할 수 있는 Port는 여러 개 존재
 - ✓ Port 1, Port 2, ... Port 10, ...
- 각 Port들이 동일한 레지스터 구성을 가지기 때문!
- 모든 Port에 대한 레지스터는 동일한 map을 참고하기 위해 Pn (n 자리에는 Port 번호) 사용

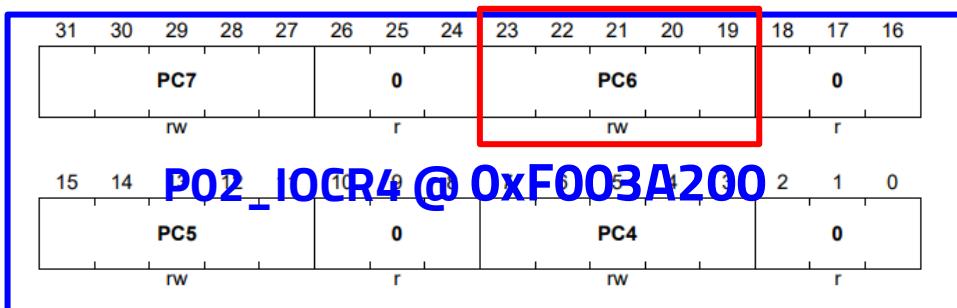
GPIO 레지스터 출력 설정

: P02_IOCR4 에 값을 쓰기 → Output Direction 설정

- P02.6 을 General Purpose Output (push-pull)으로 사용하기 위해 P02_IOCR4 레지스터에 어떤 값을 써야 하는지 확인

where?

- Pin 6 이므로 PC6 영역에 write 필요



Field	Bits	Type	Description
PC4, PC5, PC6, PC7	[7:3], [15:11], [23:19], [31:27]	rw	Port Control for Port n Pin 4 to 7 This bit field determines the Port n line x functionality (x = 4-7) according to the coding table (see Table 13-5).
0	[2:0], [10:8], [18:16], [26:24]	r	Reserved Read as 0; should be written with 0.

Table 13-5 PCx Coding

PCx[4:0]	I/O	Characteristics	Selected Pull-up / Pull-down / Selected Output Function
10000 _B	Output	Push-pull	General-purpose output
10001 _B	what? - 출력 모드로 사용하기 위해 0x10 값을 PC6 영역에 write	Open-drain	Alternate output function 1
10010 _B			Alternate output function 2
10011 _B			Alternate output function 3
10100 _B			Alternate output function 4
10101 _B			Alternate output function 5
10110 _B			Alternate output function 6
10111 _B			Alternate output function 7
11000 _B			General-purpose output
11001 _B			Alternate output function 1
11010 _B			Alternate output function 2
11011 _B			Alternate output function 3
11100 _B			Alternate output function 4
11101 _B			Alternate output function 5
11110 _B			Alternate output function 6
11111 _B			Alternate output function 7

1) This is the default pull device setting after reset for powertrain applications.

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.1083

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.1090

Lab1: 헤더 파일 작성

```
26 // -----
27 #include "Ifx_Types.h"
28 #include "IfxCpu.h"
29 #include "IfxScuWdt.h"
30
31 #include "IfxCcu6_reg.h"
32 #include "IfxVadc_reg.h"
33 #include "IfxGtm_reg.h"
34
35
36 // Port registers
37 #define PC1_BIT_LSB_IDX 11
38 #define PC2_BIT_LSB_IDX 19
39 #define PC3_BIT_LSB_IDX 27
40 #define PC4_BIT_LSB_IDX 3
41 #define PC5_BIT_LSB_IDX 11
42 #define PC6_BIT_LSB_IDX 19
43 #define PC7_BIT_LSB_IDX 27
44 #define P1_BIT_LSB_IDX 1
45 #define P2_BIT_LSB_IDX 2
46 #define P3_BIT_LSB_IDX 3
47 #define P4_BIT_LSB_IDX 4
48 #define P5_BIT_LSB_IDX 5
49 #define P6_BIT_LSB_IDX 6
50 #define P7_BIT_LSB_IDX 7
51
52 // SCU registers
53 #define LCK_BIT_LSB_IDX 1
54 #define ENDINIT_BIT_LSB_IDX 0
55 #define EXISO_BIT_LSB_IDX 4
56 #define FEN0_BIT_LSB_IDX 8
57 #define EIENO_BIT_LSB_IDX 11
58 #define INP0_BIT_LSB_IDX 12
59 #define IGPO_BIT_LSB_IDX 14
60 #define REN0_BIT_LSB_IDX 9
61
62 // SRC registers
63 #define SRPN_BIT_LSB_IDX 0
64 #define TOS_BIT_LSB_IDX 11
65 #define SRE_BIT_LSB_IDX 10
66
67 // CCU60 registers
68 #define DISS_BIT_LSB_IDX 1
69 #define DISR_BIT_LSB_IDX 0
70 #define CTM_BIT_LSB_IDX 7
71 #define T12PRE_BIT_LSB_IDX 3
72 #define T12CLK_BIT_LSB_IDX 0
73 #define T12STR_BIT_LSB_IDX 6
74 #define T12RS_BIT_LSB_IDX 1
75 #define INPT12_BIT_LSB_IDX 10
76 #define ENT12PM_BIT_LSB_IDX 7
77 #define T12SSC_BIT_LSB_IDX 0
78
79 // VADC registers
80 #define DISS_BIT_LSB_IDX 1
81 #define DISR_BIT_LSB_IDX 0
82 #define ANONC_BIT_LSB_IDX 0
83 #define ASEN0_BIT_LSB_IDX 24
84 #define CSM0_BIT_LSB_IDX 3
85 #define PRI00_BIT_LSB_IDX 0
86 #define CMS_BIT_LSB_IDX 8
87 #define FLUSH_BIT_LSB_IDX 10
88 #define TREV_BIT_LSB_IDX 9
89 #define ENGT_BIT_LSB_IDX 0
90 #define RESPOS_BIT_LSB_IDX 21
91 #define RESREG_BIT_LSB_IDX 16
92 #define ICLSEL_BIT_LSB_IDX 0
93 #define VF_BIT_LSB_IDX 31
94 #define RESULT_BIT_LSB_IDX 0
95
96 // GTM registers
97 #define DISS_BIT_LSB_IDX 1
98 #define DISR_BIT_LSB_IDX 0
99 #define SEL7_BIT_LSB_IDX 14
100 #define EN_FXCLK_BIT_LSB_IDX 22
101 #define FXCLK_SEL_BIT_LSB_IDX 0
102
103 // GTM - TOM0 registers
104 #define UPEN_CTRL1_BIT_LSB_IDX 18
105 #define HOST_TRIGGER_BIT_LSB_IDX 0
106 #define ENDIS_CTRL1_BIT_LSB_IDX 2
107 #define OUTEN_CTRL1_BIT_LSB_IDX 2
108 #define RSCNT0_CNI1_BIT_LSB_IDX 18
109 #define FUPD_CTRL1_BIT_LSB_IDX 2
110 #define CLK_SRC_SR_BIT_LSB_IDX 12
111 #define SL_BIT_LSB_IDX 11
112
```

Lab2: GPIO 출력 설정

: P02_IOCR4 레지스터에 값 쓰기

```
503  
504 void initUSonic(void)  
505 {  
506     P02_IOCR4.U &= ~(0x1F << PC6_BIT_LSB_IDX);          // reset P02_IOCR4 PC6  
507     P00_IOCR4.U &= ~(0x1F << PC4_BIT_LSB_IDX);          // reset P00_IOCR4 PC4  
508  
509     P00_TOCR4.II |= 0x01 << PC4_RTTR_LSB_IDX;           // set P00.4 general input (pull-down connected) [Echo]  
510     P02_IOCR4.U |= 0x10 << PC6_BIT_LSB_IDX;             // set P02.6 push-pull general output      [Trig]  
511  
512     P02_OUT.U &= ~(0x1 << P6_BIT_LSB_IDX);  
513 }  
514 ...
```



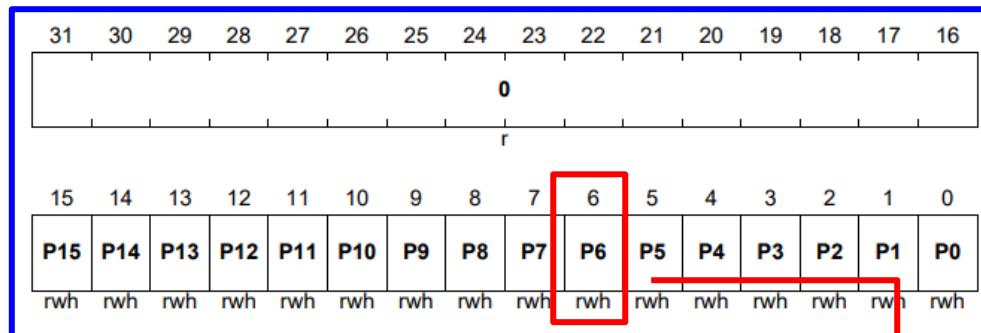
P02.6 핀의 초기 출력값을 0x0 으로 초기화

GPIO 레지스터 설정

: P02_OUT 에 값을 쓰기 → Output Value 설정

- P02.6 출력에 HIGH("1") 또는 LOW("0")의 값을 인가하기 위해 P02_OUT 레지스터에 어떤 값을 써야 하는지 확인

P02_OUT 레지스터의 32bit 전체 영역



Field	Bits	Type	Description
Px (x = 0-15)	x	rwh	Port n Output Bit x This bit determines the level at the output pin Pn.x if the output is selected as GPIO output. 0 _B The output level of Pn.x is 0. 1 _B The output level of Pn.x is 1. Pn.x can also be set or cleared by control bits of the Pn_OMSR, Pn_OMCR or Pn_OMR registers.
0	[31:16]	r	Reserved Read as 0; should be written with 0.

where?

- Pin 6 이므로 P6 영역에 write 필요
- 전체 32bit 중, P6 영역은 1bit 차지

what?

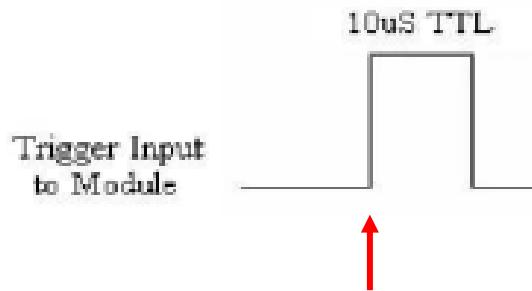
- 출력하고자 하는 값("0" 또는 "1")을 P6 영역에 write

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.1103

Lab3: 초음파 센서 Trigger 펄스 생성 함수

:Trigger 펄스 생성 시작 – HIGH 출력

```
513 void usonicTrigger(void)
514 {
515     // start of 10us Trigger Pulse
516     // GPIO P02.0 > HIGH
517     P02_OUT.U |= 0x1 << P6_BIT_LSB_IDX;
518     range_valid_flag = 0;
519     CCU60_TCTR4.U = 0x1 << T12RS_BIT_LSB_IDX;           // T12 start counting
520 }
521
```



GPIO 레지스터 입력 설정

:초음파 센서 Echo 와 연결

- 초음파 센서의 Echo 신호를 입력 받기 위해 P00.4 핀을 GPIO 입력으로 사용하기로 결정
- P00.4 핀의 입력 모드 동작을 위해서는 어떤 레지스터 설정이 필요?
- GPIO Port 00 레지스터 항목에서
 - P00_IN
 - P00_IOCR4
- 2가지 레지스터에 설정이 필요함

[Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.1125](#)

Table 13-10 Port 00 Functions (cont'd)

Port Pin	I/O	Pin Functionality	Associated Reg./ I/O Line	Port I/O Control Select.	
				Reg./Bit Field	Value
P00.4	I	General-purpose input	P00_IN.P4	P00_IOCR4. PC4	0XXXX _B
		GTM input	TIN13		
		SCU input	REQ7		
		SENT input	SENT3B		
		DSADC input	DS DIN3A		

GPIO 레지스터 입력 설정

:Port 00 레지스터 주소 확인

- 레지스터 설정을 위해 레지스터가 위치한 주소 파악 필요

1. GPIO (Ports) 레지스터 영역 찾기

[Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.227](#)

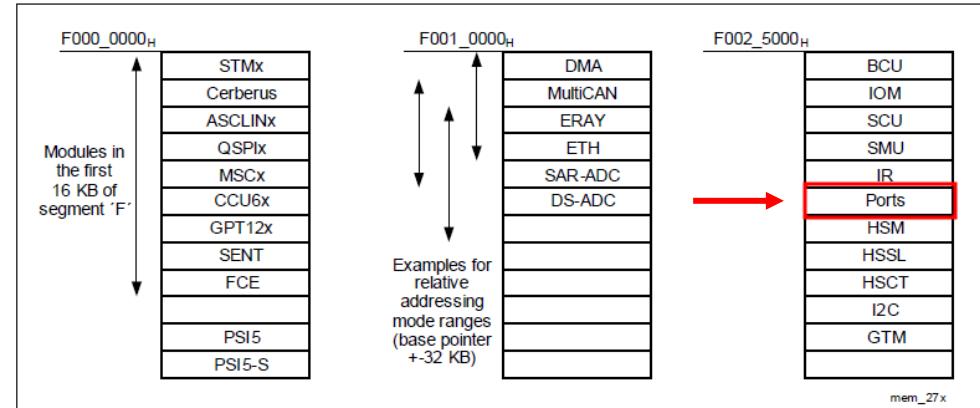


Figure 3-1 Segment F Structure

2. 실습에서 사용하는 Port 00에 해당하는 레지스터 영역 찾기

- 시작 주소 (Base address)
- = **0xF003A000**

	Port 00	F003 A000 _H - F003 A0FF _H	256 byte	access	access
	Port 01	F003 A100 _H - F003 A1FF _H	256 byte	access	access
	Port 02	F003 A200 _H - F003 A2FF _H	256 byte	access	access

[Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.231](#)

GPIO 레지스터 입력 설정

: P00_IOCR4 Direction Reg Addr 계산

3. 사용할 특정 레지스터의 주소 찾기

- **P00_IOCR4**
- **Offset Address = 0x0014**
- 즉, **Base Address**로부터 **0x0014**만큼 떨어져 있음

→ P00_IOCR4 레지스터 주소

→ = [P00 Base Addr] + [P00_IOCR4 Offset Addr]

→ = 0xF003A000 + 0x0014 = **0xF003A014**

사용할 Port 번호 n = 00

→ Pn = P00

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.1074

ID	Register	Module Identification Register	0008 _H	U, SV	BE	Application Reset	Page 13-1 3
Pn_IOCR0	Port n Input/Output Control Register 0	0010 _H	U, SV	U, SV, P	Application Reset	Page 13-1 4	
Pn_IOCR4	Port n Input/Output Control Register 4	0014_H	U, SV	U, SV, P	Application Reset	Page 13-1 4	
Pn_IOCR8	Port n Input/Output Control Register 8	0018 _H	U, SV	U, SV, P	Application Reset	Page 13-1 4	

왜 P00 이 아니라 Pn 으로 표기?

- MCU에서 사용할 수 있는 Port는 여러 개 존재
 - ✓ Port 1, Port 2, ... Port 10, ...
- 각 Port들이 동일한 레지스터 구성을 가지기 때문!
- 모든 Port에 대한 레지스터는 동일한 map을 참고하기 위해 Pn (n 자리에는 Port 번호) 사용

GPIO 레지스터 입력 값 read

: P00_IN Data Reg Addr 계산

3. 사용할 특정 레지스터의 주소 찾기

- **P00_IN**
- Offset Address = 0x0024
- 즉, **Base Address**로부터 0x0024만큼 떨어져 있음

→ P00_IN 레지스터 주소

→ = [P00 Base Addr] + [P00_IN Offset Addr]

→ = 0xF003A000 + 0x0024 = **0xF003A024**

사용할 Port 번호 n = 00

→ Pn = P00

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.1075

Table 13-4 Registers Overview (cont'd)

Register Short Name	Register Long Name	Offset Address	Access Mode		Reset	Desc. see
			Read	Write		
Pn_IOCR12	Port n Input/Output Control Register 12	001C _H	U, SV	U, SV, P	Application Reset	Page 13-14
Reserved		0020 _H	BE	BE	-	-
Pn_IN	Port n Input Register	0024 _H	U, SV	BE	Application Reset	Page 13-52
-	Reserved	0028 _H	BE	BE	-	-

왜 P00 이 아니라 Pn 으로 표기?

- MCU에서 사용할 수 있는 Port는 여러 개 존재
 - ✓ Port 1, Port 2, ... Port 10, ...
- 각 Port들이 동일한 레지스터 구성을 가지기 때문!
- 모든 Port에 대한 레지스터는 동일한 map을 참고하기 위해 Pn (n 자리에는 Port 번호) 사용

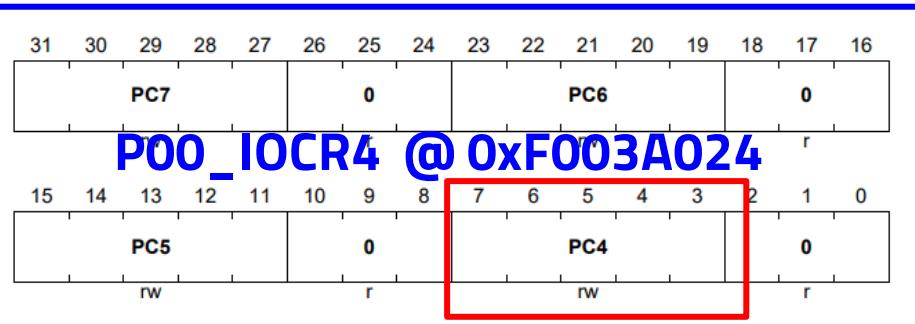
P00.4 포트 Direction 설정

: P00_IOCR4 레지스터에 값 쓰기 – Input Direction 설정

- P00.4 를 General Purpose Input 으로 사용하기 위해 P00_IOCR4 레지스터에 어떤 값을 써야 하는지 확인

where?

- Pin 4 이므로 PC4 영역에 write 필요



Field	Bits	Type	Description
PC4,	[7:0],	rw	Port Control for Port n Pin 4 to 7
PC5,	[15:11],		This bit field determines the Port n line x functionality (x = 4-7) according to the coding table (see Table 13-5).
PC6,	[23:19],		
PC7	[31:27]		
0	[2:0], [10:8], [18:16], [26:24]	r	Reserved Read as 0; should be written with 0.

P00_IOCR4 레지스터 32bit 중,
PC4 영역에 해당하는 5bit

[Infineon-TC27x_D-step-UM-v02_02-EN.pdf](#) p.1083

what?

- 풀-다운 회로가 연결된 입력 모드로 사용하기 위해 0x01 값을 PC4 영역에 write
- (X 는 무관함(don't care)을 의미)

Table 13-5 PCx Coding

PCx[4:0]	I/O	Characteristics	Selected Pull-up / Pull-down / Selected Output Function
0XX00 _B	Input	–	No input pull device connected, tri-state mode
0XX01 _B			Input pull-down device connected
0XX10 _B			Input pull-up device connected ¹⁾
0XX11 _B			No input pull device connected, tri-state mode

[Infineon-TC27x_D-step-UM-v02_02-EN.pdf](#) p.1089

Lab4: GPIO 입력 설정

: P00_IOCR4 레지스터에 값 쓰기

```
503  
504 void initUSonic(void)  
505 {  
506     P02_TOCR4.U &= ~(0x1F << PC6_BIT_LSB_IDX);           // reset P02_TOCR4 PC6  
507     P00_IOCR4.U &= ~(0x1F << PC4_BIT_LSB_IDX);           // reset P00_IOCR4 PC4  
508  
509     P00_IOCR4.U |= 0x01 << PC4_BIT_LSB_IDX;                // set P00.4 general input (pull-down connected) [Echo]  
510     P02_TOCR4.U |= 0x10 << PC6_BIT_LSB_IDX;                // set P02.6 push-pull general output [!rig]  
511  
512     P02_OUT.U &= ~(0x1 << P6_BIT_LSB_IDX);  
513 }  
514 ...
```

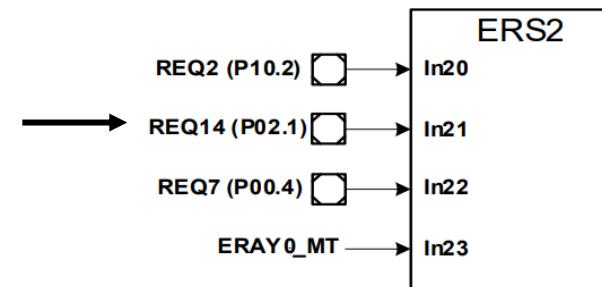
이전에 작성했던 레지스터 설정 코드에서 새롭게 추가 & 변경되는 부분

초음파 센서 ECHO 핀 GPIO INTERRUPT 발생 위한 ERU 레지스터 설정

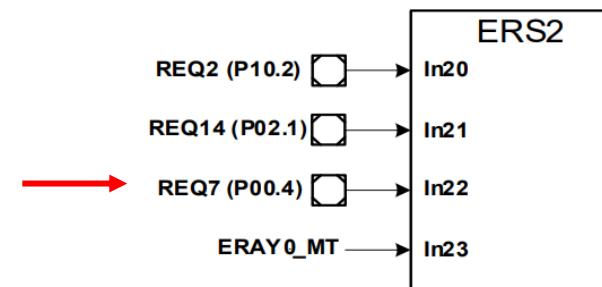
초음파 센서 Echo 입력위한 레지스터 설정

:Interrupt 채널 선택 설정 변경 필요

- 이전 Push 버튼 실습에서는 ERS2로 연결되는 Interrupt 중, P02.1에서 발생하는 GPIO 입력 Interrupt (REQ14)를 사용함



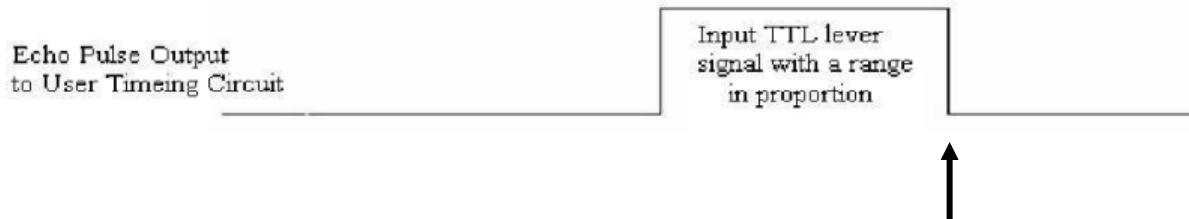
- 이번 실습에서는 P00.4 (REQ7)에서 발생하는 GPIO 입력 Interrupt 사용



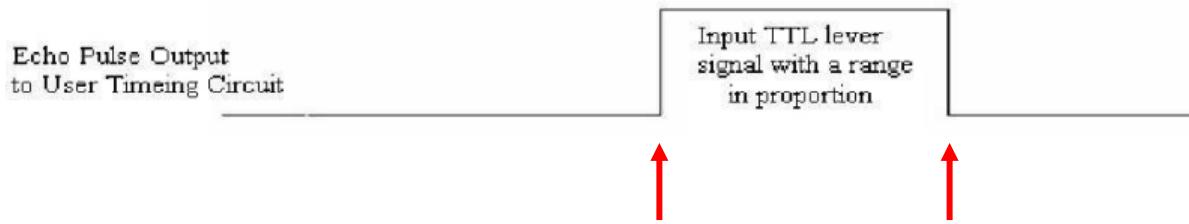
초음파 센서 Echo 입력위한 레지스터 설정

:Interrupt 발생 조건 설정 변경 필요

- 이전 Push 버튼 실습에서는 GPIO 입력의 falling edge에서만 Interrupt 발생



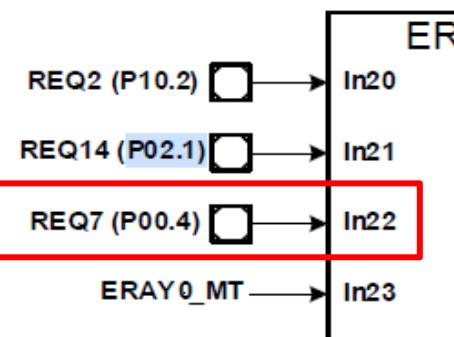
- 이번 실습에서는 falling edge 뿐만 아니라, rising edge에서도 Interrupt 발생 필요



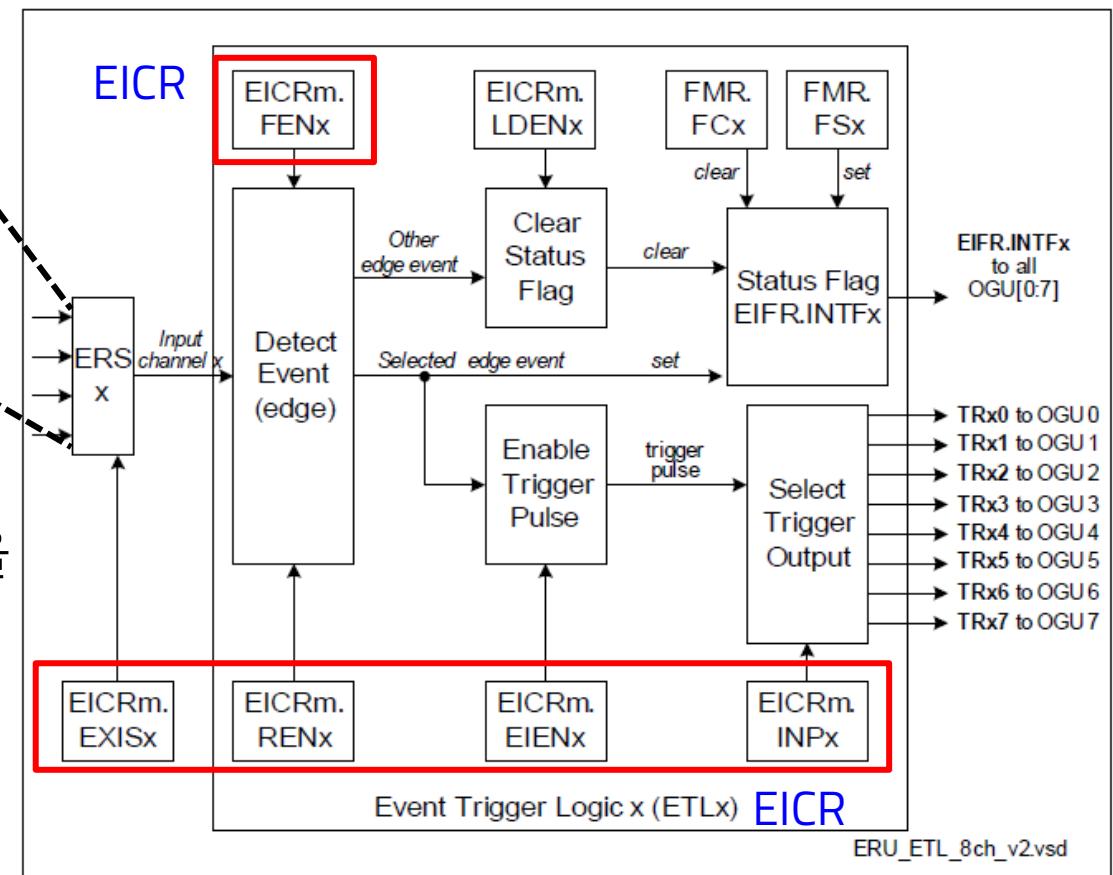
GPIO와 External Interrupt 연결 위한 레지스터 설정

: ERU External Input Channel 설정 → EICR 레지스터 설정

- 본 실습에서 GPIO 입력으로 사용하는 핀 → P00.4
- 해당 핀은 어느 ERS에 입력? → ERS2 의 In22 (REQ7)



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



- ERU의 External Input Channel 설정을 위해서 SCU 레지스터 항목에서
 - EICR 레지스터 설정 필요

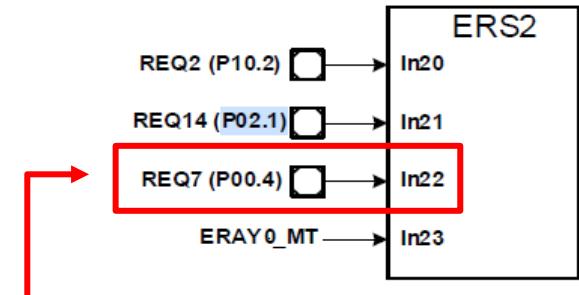
SCU 레지스터 설정 – EICR

: 레지스터 주소 계산

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

2. 사용할 레지스터의 주소 찾기

- EICR_i (i번째) 레지스터에는 ERS 채널 2개씩 할당
 - EICR0: ERS0-1, **EICR1: ERS2-3** ..., EICR2: ERS4-5



본 실습의 P00.4 핀은 **ERS2**를 사용하므로
→ **EICR1** 사용

[Infineon-TC27x_D-step-UM-v02_02-EN.pdf](#)
p.624

→ EICR1 레지스터 주소

$$= [\text{SCU Base Addr}] + [\text{EICR1 Offset Addr}]$$

$$= 0xF0036000 + 0x214 = \textcolor{red}{\mathbf{0xF0036214}}$$

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	$21C_{\text{H}}$	U, SV	U, SV, P	Application	Page

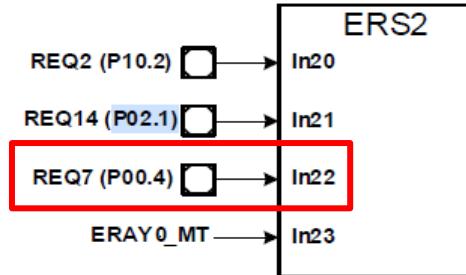
[Infineon-TC27x_D-step-UM-v02_02-EN.pdf](#) p.698

SCU 레지스터 설정 – EICR

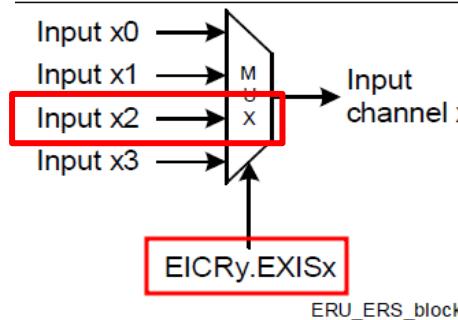
: 채널 선택 – EXISO

3. 레지스터 write 값 결정

- ERS Channel 2 의 4개 External Request 입력 중, 2번째 입력 (REQ7) 선택



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



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

- ERS2의 2번째 입력을 선택하기 위해 EXISO 영역에 0x2 write

EICR1 External Input Channel Register 1 (214H)																Reset Value: 0000 0000H			
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	rw	r	rw	rw	r	r	r	r	r	r	r			

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
0	INP0		EI EN0	LD EN0	R EN0	F EN0	0	EXIS0		0						
r	rw	rw	rw	rw	rw	rw	r	rw	rw	r	r	r	r	r	r	r

EICR1 레지스터 @ 0xF0036214

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

Field	Bits	Type	Description
EXISO	[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

Lab5: SCU 레지스터 설정 – EICR의 EXISO 설정 :기존 *initERU()* 함수 수정

```
106 void initERU(void)
107 {
108     // ERU setting
109     SCU_EICR1.U &= ~(0x7 << EXISO_BIT_LSB_IDX);
110     SCU_EICR1.U |= (0x1 << EXISO_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
```

```
254 void initERU(void)
255 {
256     // ERU setting
257     SCU_EICR1.U &= ~(0x7 << EXISO_BIT_LSB_IDX);
258
259     SCU_EICR1.U |= (0x2 << EXISO_BIT_LSB_IDX); → // ERS2 - Input 2
260
261
262     SCU_EICR1.U |= 0x1 << FEN0_BIT_LSB_IDX; // falling edge
263
264     SCU_EICR1.U |= 0x1 << REN0_BIT_LSB_IDX; // rising edge
265
266
267     SCU_EICR1.U |= 0x1 << EIEN0_BIT_LSB_IDX;
268
269     SCU_EICR1.U &= ~(0x7 << INP0_BIT_LSB_IDX);
270
271     SCU_IGCR0.U &= ~(0x3 << IGP0_BIT_LSB_IDX);
272     SCU_IGCR0.U |= 0x1 << IGP0_BIT_LSB_IDX;
273
274     // SRC Interrupt setting
275     SRC_SCU_SCU_ERU0.U &= ~(0xFF << SRPN_BIT_LSB_IDX);
276     SRC_SCU_SCU_ERU0.U |= 0x0A << SRPN_BIT_LSB_IDX;
277
278     SRC_SCU_SCU_ERU0.U &= ~(0x3 << TOS_BIT_LSB_IDX);
279
280     SRC_SCU_SCU_ERU0.U |= 1 << SRE_BIT_LSB_IDX;
281
282 }
```

Push 버튼 실습

초음파 센서 실습

SCU 레지스터 설정 – EICR

: Falling Edge, Rising Edge 모두 발생 조건으로 선택

3. 레지스터 write 값 결정

- 본 실습 초음파센서의 Echo 신호의 시작과 끝을 모두 인식해야 하므로 Rising, Falling Edge 모두에서 Interrupt가 발생해야 함
- Falling 엣지가 발생했을 때 Event를 생성하도록 **FENO** 영역에 **0x1 write**
- Rising 엣지가 발생했을 때 Event를 생성하도록 **RENO** 영역에 **0x1 write**

EICR1 레지스터 @ 0xF0036214

EICR1 External Input Channel Register 1 (214H)																Reset Value: 0000 0000H															
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	rw	rw	r	rw	rw	rw	rw	rw	rw	r	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ERS3 설정	→	0	INP0	EI EN0	LD EN0	R EN0	F EN0	0	EXIS0	0																					
ERS2 설정	→	0	INP0	EI EN0	LD EN0	R EN0	F EN0	0	EXIS0	0																					

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). 0B The falling edge is not used. 1B The detection of a falling edge of Input Channel 0 generates a trigger event. INTF(2i) becomes set.
RENO	9	rw	Rising Edge Enable 0 This bit determines if the rising edge of Input Channel (2*i) is used to set bit INTF(2i). 0B The rising edge is not used. 1B The detection of a rising edge of Input Channel (2*i) generates a trigger event. INTF(2*i) becomes set

Lab6: SCU 레지스터 설정 – EICR의 FEN, REN 설정 :기존 *initERU()* 함수 수정

```
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; // 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
```

```
254 void initERU(void)
255 {
256     // ERU setting
257     SCU_EICR1.U &= ~(0x7 << EXIS0_BIT_LSB_IDX);
258
259     SCU_EICR1.U |= (0x2 << EXIS0_BIT_LSB_IDX); // ERS2 - Input 2
260
261     SCU_EICR1.U |= 0x1 << FEN0_BIT_LSB_IDX; // falling edge
262
263     SCU_EICR1.U |= 0x1 << REN0_BIT_LSB_IDX; // rising edge
264
265
266     SCU_EICR1.U |= 0x1 << EIEN0_BIT_LSB_IDX;
267
268     SCU_EICR1.U &= ~(0x7 << INP0_BIT_LSB_IDX);
269
270     SCU_IGCR0.U &= ~(0x3 << IGP0_BIT_LSB_IDX);
271     SCU_IGCR0.U |= 0x1 << IGP0_BIT_LSB_IDX;
272
273
274     // SRC Interrupt setting
275     SRC_SCU_SCU_ERU0.U &= ~(0xFF << SRPN_BIT_LSB_IDX);
276     SRC_SCU_SCU_ERU0.U |= 0x0A << SRPN_BIT_LSB_IDX;
277
278     SRC_SCU_SCU_ERU0.U &= ~(0x3 << TOS_BIT_LSB_IDX);
279
280     SRC_SCU_SCU_ERU0.U |= 1 << SRE_BIT_LSB_IDX;
281 }
```

Push 버튼 실습

초음파 센서 실습

이전에 작성했던 레지스터 설정 코드에서 새롭게 추가 & 변경되는 부분

초음파 센서 TRIGGER 펄스 생성 위한 타이머 CCU60 레지스터 설정

CCU60 내부 T12 모듈 사용 위한 레지스터 설정

: T12 Counter를 구동하는 클럭 속도 조절

- CCU60 내부 T12 설정 위한 TCTR0 레지스터에서
 - **T12CLK** 영역 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

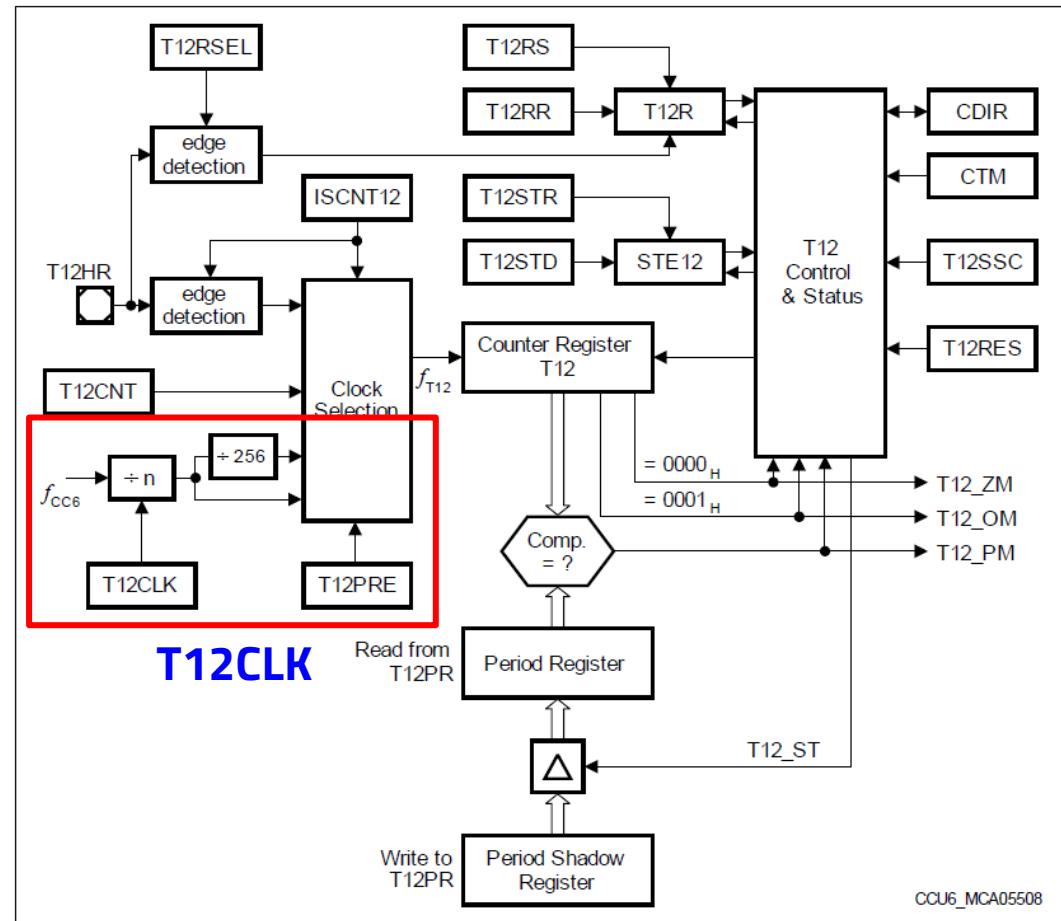


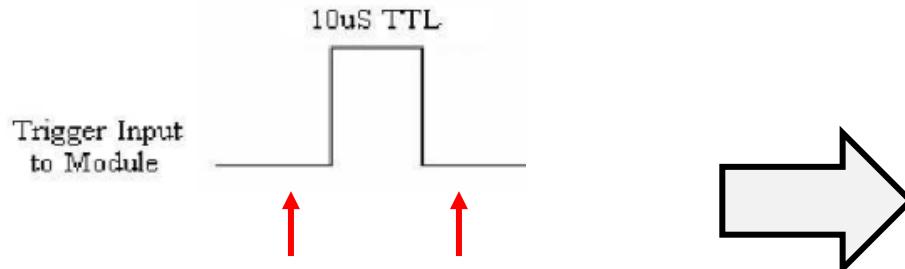
Figure 26-4 Timer T12 Logic and Period Comparators

CCU60 레지스터 설정 – TCTRO

: Clock 선택 (분주 or Prescaling)

- 기존 실습에서 레지스터 write 값 결정
 - CCU60 모듈에 처음 입력되는 clock 주파수가 50 MHz 이므로, **T12CLK** 영역에 **0x2값을 write** 하여 $50\text{MHz} / 4 = 12.5\text{MHz}$ 의 clock 생성

기존 실습에서는 **T12PRE** 영역 **0x1 write**하여
 $12.5\text{MHz} / 256 = 48,828\text{Hz}$ 의 **T12 clock** 생성



$$\frac{1}{freq.} = period = \frac{1}{48,828} = 20.48\mu s$$

T12PRE 영역에 값을
write하지 않고 **12.5 MHz**
clock을 **T12**에서 그대로
사용

하지만 48,828 Hz 주파수로는 10us 길이 펄스 생성
불가능 (주기가 너무 느려 10us 시간을 초과함)

CCU60 레지스터 설정 – TCTRO

: Clock 선택 (분주 or Prescaling)

3. 레지스터 write 값 결정

- CCU60 모듈에 처음 입력되는 clock 주파수가 50 MHz 이므로, **T12CLK** 영역에 **0x2값을 write** 하여 $50\text{ MHz} / 4 = 12.5\text{ MHz}$ 의 clock 생성
 - **T12PRE** 영역은 사용하지 않음

TCTR0 레지스터 @ 0xF0002A70

TCTR0 Timer Control Register 0																(70H)		
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					

Field	Bits	Type	Description
T12CLK	[2:0]	rw	<p>Timer T12 Input Clock Select</p> <p>Selects the input clock for timer T12 that is derived from the peripheral clock according to the equation</p> $f_{T12} = f_{CC6} / 2^{<T12CLK>}$ <ul style="list-style-type: none"> 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 $f_{T12} = f_{CC6} / 128$

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

Lab7: CCU60 레지스터 설정 - TCTR0의 T12CLK 설정

:기존 *initCCU60()* 함수 수정

```

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     // CCU60 T12 configurations
171     while((CCU60_CLC.U & (1 << DISS_BIT_LSB_IDX)) != 0); // wait until CCU60 module enabled
172
173     CCU60_TCTR0.U &= ~(0x7 << T12CLK_BIT_LSB_IDX);    // f_T12 = f_CCU6 / prescaler
174     CCU60_TCTR0.U |= 0x2 << T12CLK_BIT_LSB_IDX;        // f_CCU6 = 50 MHz, prescaler = 1024
175     CCU60_TCTR0.U |= 0x1 << T12PRE_BIT_LSB_IDX;        // f_T12 = 48,828 Hz
176
177     CCU60_TCTR0.U &= ~(0x1 << CTM_BIT_LSB_IDX);        // T12 auto reset when period match (PM) occur
178
179

```

```

303     // CCU60 T12 configurations
304     while((CCU60_CLC.U & (1 << DISS_BIT_LSB_IDX)) != 0); // wait until CCU60 module enabled
305
306     CCU60_TCTR0.U &= ~(0x7 << T12CLK_BIT_LSB_IDX);    // f_T12 = f_CCU6 / prescaler
307     CCU60_TCTR0.U |= 0x2 << T12CLK_BIT_LSB_IDX;        // f_CCU6 = 50 MHz, prescaler = 4
308     //CCU60_TCTR0.U |= 0x1 << T12PRE_BIT_LSB_IDX;        // f_T12 --> 12.5 MHz
309
310
311     CCU60_TCTR0.U &= ~(0x1 << CTM_BIT_LSB_IDX);        // T12 auto reset when period match (PM) occur
312
313
314     CCU60_T12PR.U = 125 -1;                            // PM interrupt freq. = f_T12 / (T12PR + 1)
315     CCU60_TCTR4.U |= 0x1 << T12STR_BIT_LSB_IDX;        // load T12PR from shadow register
316
317
318     CCU60_TCTR2.B.T12SSC = 0x1;                        // Single Shot Control
319
320
321
322     CCU60_T12.U = 0;                                  // clear T12 counter register
323

```

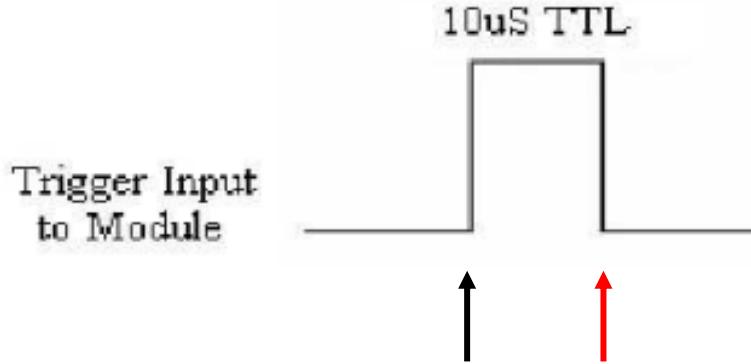
타이머 실습

초음파 센서 실습

CCU60 내부 T12 모듈 사용 위한 레지스터 설정

: 초음파 센서 Trigger 신호 생성 과정에서 Period Match Interrupt 사용

- Period Match Interrupt는 매 clock마다 1씩 증가하는 counter가 period 값에 도달하면 발생
→ Trigger 펄스의 10us 길이 측정하는데 사용



Trigger 핀에 해당하는 GPIO 출력을 HIGH로 변경하며 카운터 증가 시작

Counter 값이 10us에 해당하는 길이만큼의 period 값에 도달하면 PM Interrupt 발생
Trigger 핀에 해당하는 GPIO 출력을 LOW로 변경

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
- T12 모듈의 clock 주파수를 바탕으로, 얼마의 시간이후 타이머 Interrupt를 발생시킬지 고려하여 Period Register를 셋팅해야 함

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

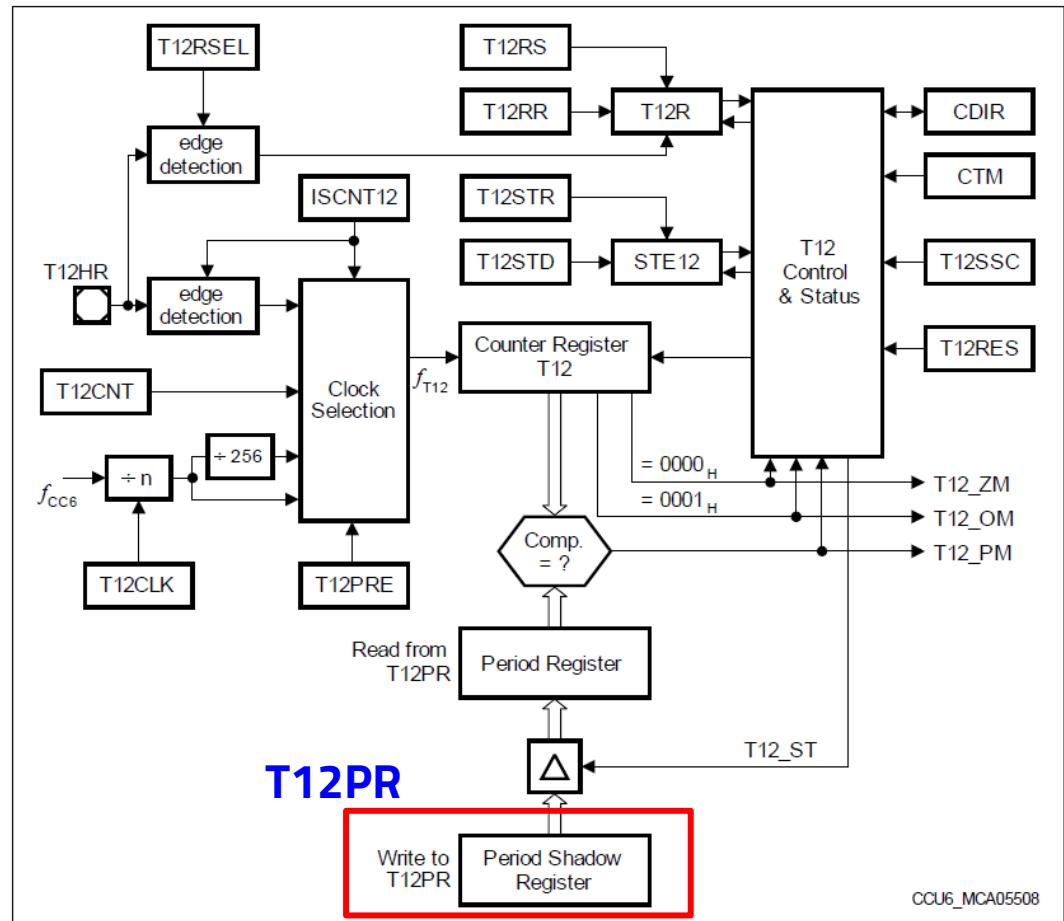


Figure 26-4 Timer T12 Logic and Period Comparators

CCU60 레지스터 설정 – T12PR

: 레지스터 주소 찾기 – 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 = $0xF0002A00 + 0x24 = \textcolor{red}{0xF0002A24}$

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
CCSRR	Control/Status Register	20	U, SV	U, SV	Application	26-35

[Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.3641](#)

CCU60 레지스터 설정 – T12PR

: Clock 주파수와 Period를 고려하여 Match 시간 간격 결정

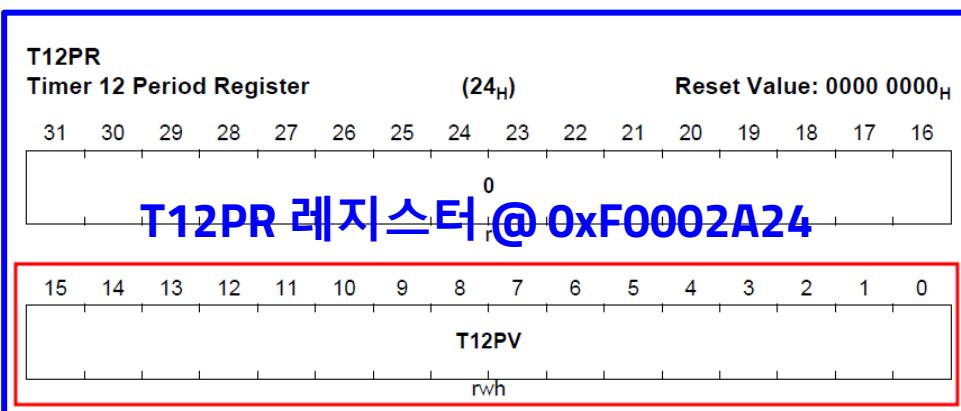
3. 레지스터 write 값 결정

- 계획하는 Period Match 의 발생 시간 간격을 고려하여 T12PV 영역에 T12 모듈의 counter 레지스터와 match 여부 비교될 값을 write

→ T12 모듈의 clock 주파수(frequency)가 **12.5 MHz** 이므로, **0.08us** 마다 counter 증가함

→ 타이머 동작 시작 이후, **10us** 후에 Period Match를 발생하려면 counter 가 **125** 값에 도달했을 때 Period Match 발생해야 함 ($10\text{us} / 0.08\text{us} = 125$ 이므로)

→ T12PV 영역에 write할 값 = $125 - 1 = 124$ (**0x7C**)



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 timer T12 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.

Lab8: CCU60 레지스터 설정 - T12PR의 T12PV 설정 :기존 *initCCU60()* 함수 수정

```

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_CCY / prescaler
175     CCU60_TCTR0.U |= 0x2 << T12CLK_BIT_LSB_IDX; // f_CCY = 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     CCU60_T12PR.U = 24414 -1; // PM interrupt freq. = f_T12 / (T12PR + 1)
181     CCU60_TCTR0.U |= 0x1 << T12SSC_BIT_LSB_10X; // load T12PR from shadow register
182
183     CCU60_T12.U = 0; // clear T12 counter register
184

```

```

303
304     // CCU60 T12 configurations
305     while((CCU60_CLC.U & (1 << DISS_BIT_LSB_IDX)) != 0); // wait until CCU60 module enabled
306
307     CCU60_TCTR0.U &= ~(0x7 << T12CLK_BIT_LSB_IDX); // f_T12 = f_CCY / prescaler
308     CCU60_TCTR0.U |= 0x2 << T12CLK_BIT_LSB_IDX; // f_CCY = 50 MHz, prescaler = 4
309     //CCU60_TCTR0.U |= 0x1 << T12PRE_BIT_LSB_IDX; // f_T12 --> 12.5 MHz
310
311
312     CCU60_TCTR0.U &= ~(0x1 << CTM_BIT_LSB_IDX); // T12 auto reset when period match (PM) occur
313
314
315     CCU60_T12PR.U = 125 -1; // PM interrupt freq. = f_T12 / (T12PR + 1)
316     CCU60_TCTR0.U |= 0x1 << T12SSC_BIT_LSB_10X; // load T12PR from shadow register
317
318
319     CCU60_TCTR2.B.T12SSC = 0x1; // Single Shot Control
320
321
322     CCU60_T12.U = 0; // clear T12 counter register
323

```

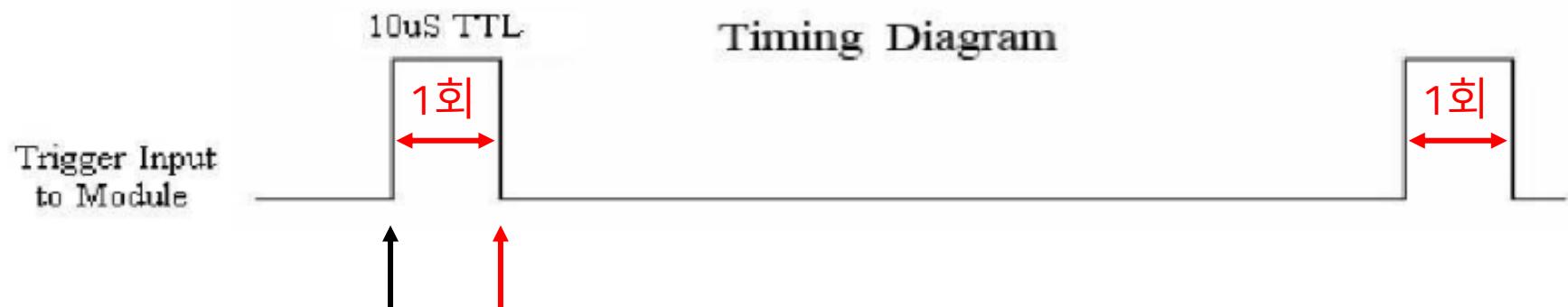
타이머 실습

초음파 센서 실습

CCU60 내부 T12 모듈 사용 위한 레지스터 설정

: 초음파 센서 Trigger 신호 1번 생성할 때만 타이머 1회 (single shot) 구동

- 초음파 센서 동작을 위한 Trigger 펄스가 시작하는 시점은 우리가 정해야 함
→ 10us 동안 타이머 동작 후, 1번 Trigger 펄스 출력이 끝나면 다음 Trigger 펄스 보내기 전까지 대기 필요
- CCU60 타이머는 한번 시작-종료 후 자동으로 다시 시작하지 않는 single shot 모드로 동작해야 함



Trigger 핀에 해당하는 GPIO 출력을 HIGH로 변경하면 카운터 증가 시작

Counter 값이 10us에 해당하는 길이만큼의 period 값에 도달하면 PM Interrupt 발생
Trigger 핀에 해당하는 GPIO 출력을 LOW로 변경
타이머 1회 수행 후, 다음 수행까지 대기

CCU60 내부 T12 모듈 사용 위한 레지스터 설정

: Single Shot 설정 레지스터 – TCTR2

- CCU60 레지스터 항목에서
 - TCTR2 레지스터 설정 필요
- CCU60 내부 T12 모듈의 counter 기능을 **single-shot** 모드로 사용하기 위한 T12SSC 영역 설정

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

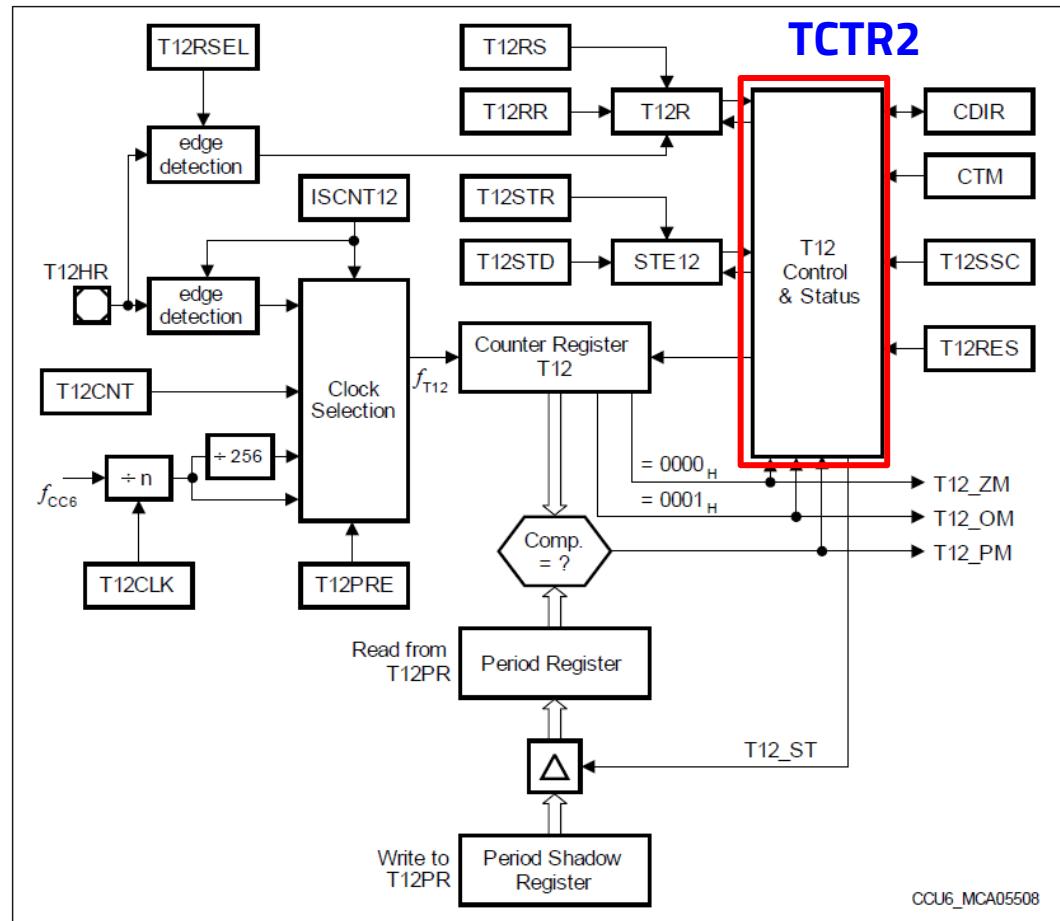


Figure 26-4 Timer T12 Logic and Period Comparators

CCU60 레지스터 설정 – TCTR2

: 레지스터 주소 찾기 – TCTR2

1. CCU60 레지스터 영역의 주소 찾기

- 시작 주소 (Base address) = **0xF0002A00**

2. 사용할 레지스터의 주소 찾기

- TCTR2 의 Offset Address = **0x74**

$$\rightarrow \text{TCTR2 레지스터 주소} = 0xF0002A00 + 0x74 = \textcolor{red}{\mathbf{0xF0002A74}}$$

Register	TCR2 Capture/Compare Mode Select Register	00H	U, SV	U, SV, P	Application Reset	26-44
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	76 _H	U, SV	U, SV, P	Application Reset	26-51

[Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.3642](#)

CCU60 레지스터 설정 – TCTR2

:Single-Shot 모드 설정

3. 레지스터 write 값 결정

- Trigger 펄스 시간 길이 10us를 1번 카운팅 후, 다음 번 trigger 펄스 출력때까지 카운터 값이 0인 상태에서 대기하도록 **T12SSC** 영역에 **0x1 write**
- **Period Match Interrupt** 가 발생하면, **T12** 모듈의 카운터는 **0**으로 초기화되고, 레지스터 설정으로 **T12** 카운터를 다시 시작할 때까지 **0**인 상태를 유지함

TCTR2 Timer Control Register 2																(74H)	Reset Value: 0000 0000H
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16		
																0	
TCTR2 레지스터 @ 0xF0002A74																	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
																T13 RSEL	T12 RSEL
																r	rw
																0	
																T13 TED	T13 TEC
																rw	rw
																T13 SSC	T12 SSC
																rw	rw

[Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.3684](#)

Field	Bits	Type	Description
T12SSC	0	rw	Timer T12 Single Shot Control This bit controls the single shot-mode of T12. 0_B The single-shot mode is disabled, no HW action on T12R. 1_B The single shot mode is enabled, the bit T12R is cleared by HW if - T12 reaches its period value in edge-aligned mode - T12 reaches the value 1 while down counting in center-aligned mode. In parallel to the clear action of bit T12R, the bits CC6xST (x=0, 1, 2) are cleared.

Lab9: CCU60 레지스터 설정 - TCTR2의 T12SSC 설정 :기존 *initCCU60()* 함수 수정

```

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     CCU60_T12PR.U = 24414 -1; // PM interrupt freq. = f_T12 / (T12PR + 1)
181     CCU60_TCTR4.U |= 0x1 << T12STR_BIT_LSB_IDX; // load T12PR from shadow register
182
183     CCU60_T12.U = 0; // clear T12 counter register
184
185 }

```

```

303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
589 추가

```

타이머 실습

초음파 센서 실습

Lab10: CCU60 레지스터 설정 – TCTR4의 T12RS 설정

:기존 *initCCU60()* 함수 수정

:CCU60에서 10us 길이 타이머 시작

```

150 void initCCU60(void)
151 {
152     // Password Access to unlock SCU_WDTSCON0
153     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFF) & ~(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 ^ 0xFF) | (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 ^ 0xFF) & ~(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 ^ 0xFF) | (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

```

삭제

CCU60 타이머 작동을 시작하는 부분을
초음파 센서 거리 측정 시작 함수로 분리

```

512
513 void usonicTrigger(void)
514 {
515     // start of 10us Trigger Pulse
516     // GPIO P02.6 --> HIGH
517     P02_OUT.U |= 0x1 << P6_BIT_LSB_IDX;
518     range_valid_flag = 0;
519     CCU60_TCTR4.U = 0x1 << T12RS_BIT_LSB_IDX; // T12 start counting
520 }
521

```

타이머 실습

HYUNDAI



초음파 센서 실습

이전에 작성했던 레지스터 설정 코드에서 새롭게 추가 & 변경되는 부분

초음파 센서 ECHO 신호 길이 측정 위한 타이머 CCU61 레지스터 설정

**별도의 타이머 1개 추가로 필요
:Echo 신호의 길이 측정을 위한 타이머 CCU61**

- Echo 신호의 Rising Edge에서 타이머 카운터가 증가하기 시작하여, Falling Edge 까지의 counter 값을 기록할 **CCU61** 의 **T12** 타이머 사용

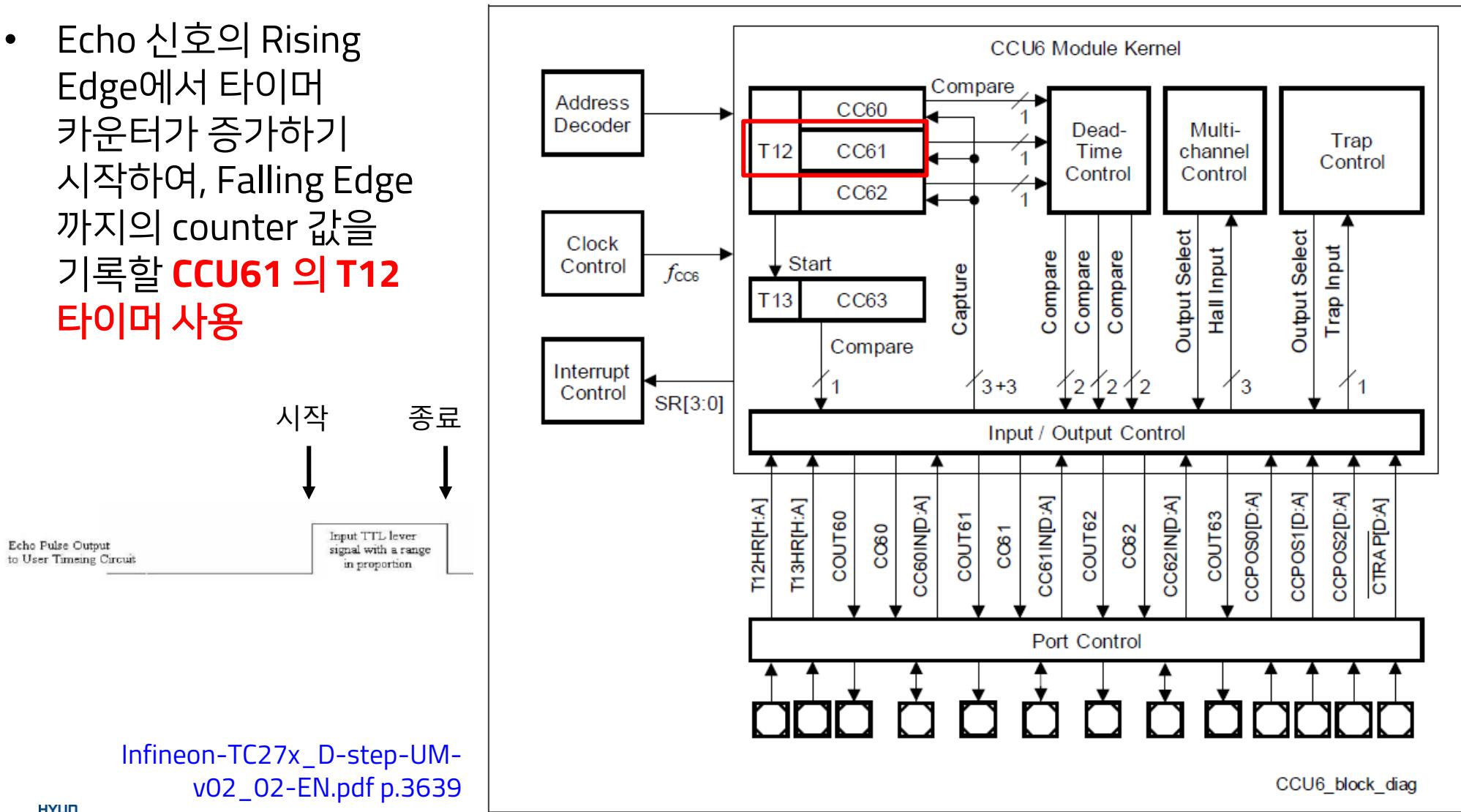


Figure 26-1 CCU6 Block Diagram

Lab11: CCU61 레지스터 설정

:기존 *initCCU60()* 함수 재사용 - *initCCU61()* 작성

```
338
339 void initCCU61(void)
340 {
341     // Password Access to unlock SCU_WDTSCON0
342     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
343     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0);    // wait until unlocked
344
345     // Modify Access to clear ENDINIT
346     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) & ~(1 << ENDINIT_BIT_LSB_IDX);
347     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0);    // wait until locked
348
349     CCU61_CLC.U &= ~(1 << DISR_BIT_LSB_IDX);    // enable CCU
350
351     // Password Access to unlock SCU_WDTSCON0
352     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
353     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0);    // wait until unlocked
354
355     // Modify Access to set ENDINIT
356     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
357     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0);    // wait until locked
358
359     // CCU60 T12 configurations
360     while((CCU61_CLC.U & (1 << DISS_BIT_LSB_IDX)) != 0); // wait until CCU60 module enabled
361
362
363     CCU61_TCTR0.U &= ~(0x7 << T12CLK_BIT_LSB_IDX);        // f_T12 = f_CCU6 / prescaler = 12.5 MHz
364     CCU61_TCTR0.U |= 0x2 << T12CLK_BIT_LSB_IDX;            // f_CCU6 = 50 MHz, prescaler = 4
365
366     CCU61_TCTR0.U |= 0x1 << T12PRE_BIT_LSB_IDX;           // f_T12 = f_CCU6 / 256 = 48,828 Hz
367
368
369     CCU61_T12PR.U = 100000 -1;                            // PM interrupt freq. = f_T12 / (T12PR + 1)
370     CCU61_TCTR4.U |= 0x1 << T12STR_BIT_LSB_IDX;          // load T12PR from shadow register
371
372     CCU61_T12.U = 0;                                     // clear T12 counter register
373 }
```

Lab12: CCU61 레지스터 설정

:CCU61 모듈 사용 설정

```
338
339 void initCCU61(void)
340 {
341     // Password Access to unlock SCU_WDTSCON0
342     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
343     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0);    // wait until unlocked
344
345     // Modify Access to clear ENDINIT
346     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) & ~(1 << ENDINIT_BIT_LSB_IDX);
347     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0);    // wait until locked
348
349     CCU61_CLC.U &= ~(1 << DISR_BIT_LSB_IDX);    // enable CCU
350
351     // Password Access to unlock SCU_WDTSCON0
352     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
353     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0);    // wait until unlocked
354
355     // Modify Access to set ENDINIT
356     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
357     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0);    // wait until locked
358
359     // CCU60 T12 configurations
360     while((CCU61_CLC.U & (1 << DISS_BIT_LSB_IDX)) != 0); // wait until CCU60 module enabled
361
362
363     CCU61_TCTR0.U &= ~(0x7 << T12CLK_BIT_LSB_IDX);        // f_T12 = f_CCU6 / prescaler = 12.5 MHz
364     CCU61_TCTR0.U |= 0x2 << T12CLK_BIT_LSB_IDX;            // f_CCU6 = 50 MHz, prescaler = 4
365
366     CCU61_TCTR0.U |= 0x1 << T12PRE_BIT_LSB_IDX;           // f_T12 = f_CCU6 / 256 = 48,828 Hz
367
368
369     CCU61_T12PR.U = 100000 -1;                            // PM interrupt freq. = f_T12 / (T12PR + 1)
370     CCU61_TCTR4.U |= 0x1 << T12STR_BIT_LSB_IDX;          // load T12PR from shadow register
371
372     CCU61_T12.U = 0;                                     // clear T12 counter register
373 }
```

CCU61 모듈 사용을
위한 보호 레지스터
잠금 및 해제

Lab13: CCU61 레지스터 설정

:CCU61 모듈 T12 타이머 clock 설정

```
338
339 void initCCU61(void)
340 {
341     // Password Access to unlock SCU_WDTSCON0
342     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
343     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0);    // wait until unlocked
344
345     // Modify Access to clear ENDINIT
346     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) & ~(1 << ENDINIT_BIT_LSB_IDX);
347     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0);    // wait until locked
348
349     CCU61_CLC.U &= ~(1 << DISR_BIT_LSB_IDX);    // enable CCU
350
351     // Password Access to unlock SCU_WDTSCON0
352     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
353     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0);    // wait until unlocked
354
355     // Modify Access to set ENDINIT
356     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
357     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0);    // wait until locked
358
359     // CCU60 T12 configurations
360     while((CCU61_CLC.U & (1 << DISS_BIT_LSB_IDX)) != 0); // wait until CCU60 module enabled
361
362
363     CCU61_TCTR0.U &= ~(0x7 << T12CLK_BIT_LSB_IDX);        // f_T12 = f_CCU6 / prescaler = 12.5 MHz
364     CCU61_TCTR0.U |= 0x2 << T12CLK_BIT_LSB_IDX;            // f_CCU6 = 50 MHz, prescaler = 4
365
366     CCU61_TCTR0.U |= 0x1 << T12PRE_BIT_LSB_IDX;           // f_T12 = f_CCU6 / 256 = 48,828 Hz
367
368
369     CCU61_T12PR.U = 100000 -1;                            // PM interrupt freq. = f_T12 / (T12PR + 1)
370     CCU61_TCTR4.U |= 0x1 << T12STR_BIT_LSB_IDX;          // load T12PR from shadow register
371
372     CCU61_T12.U = 0;                                     // clear T12 counter register
373 }
```

CCU61 모듈의 T12 카운터 사용을 위한 레지스터 설정

CCU61 모듈의 T12에서 사용하는 clock은 48,828 Hz

Lab14: CCU61 레지스터 설정

: CCU61 모듈 T12 타이머 Period 값 설정 (의미없는 값)

```
338
339 void initCCU61(void)
340 {
341     // Password Access to unlock SCU_WDTSCON0
342     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
343     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0);    // wait until unlocked
344
345     // Modify Access to clear ENDINIT
346     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) & ~(1 << ENDINIT_BIT_LSB_IDX);
347     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0);    // wait until locked
348
349     CCU61_CLC.U &= ~(1 << DISR_BIT_LSB_IDX);    // enable CCU
350
351     // Password Access to unlock SCU_WDTSCON0
352     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
353     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0);    // wait until unlocked
354
355     // Modify Access to set ENDINIT
356     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
357     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0);    // wait until locked
358
359     // CCU60 T12 configurations
360     while((CCU61_CLC.U & (1 << DISS_BIT_LSB_IDX)) != 0); // wait until CCU60 module enabled
361
362
363     CCU61_TCTR0.U &= ~(0x7 << T12CLK_BIT_LSB_IDX);        // f_T12 = f_CCU6 / prescaler = 12.5 MHz
364     CCU61_TCTR0.U |= 0x2 << T12CLK_BIT_LSB_IDX;            // f_CCU6 = 50 MHz, prescaler = 4
365
366     CCU61_TCTR0.U |= 0x1 << T12PRE_BIT_LSB_IDX;           // f_T12 = f_CCU6 / 256 = 48,828 Hz
367
368
369     CCU61_T12PR.U = 100000 -1;                            // PM interrupt freq. = f_T12 / (T12PR + 1)
370     CCU61_TCTR4.U |= 0x1 << T12STR_BIT_LSB_IDX;          // load T12PR from shadow register
371
372     CCU61_T12.U = 0;                                     // clear T12 counter register
373 }
```

CCU61 모듈의 T12 카운터 사용을 위한 레지스터 설정

CCU61 모듈 T12 에서는 Period Match를 사용하지 않으므로 → 임의의 큰 값으로 설정

Lab15: CCU61 레지스터 설정

: CCU61 모듈 T12 타이머 카운터 초기화

```
338
339 void initCCU61(void)
340 {
341     // Password Access to unlock SCU_WDTSCON0
342     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
343     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0);      // wait until unlocked
344
345     // Modify Access to clear ENDINIT
346     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) & ~(1 << ENDINIT_BIT_LSB_IDX);
347     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0);      // wait until locked
348
349     CCU61_CLC.U &= ~(1 << DISR_BIT_LSB_IDX);    // enable CCU
350
351     // Password Access to unlock SCU_WDTSCON0
352     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
353     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0);      // wait until unlocked
354
355     // Modify Access to set ENDINIT
356     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
357     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0);      // wait until locked
358
359     // CCU60 T12 configurations
360     while((CCU61_CLC.U & (1 << DISS_BIT_LSB_IDX)) != 0); // wait until CCU60 module enabled
361
362
363     CCU61_TCTR0.U &= ~(0x7 << T12CLK_BIT_LSB_IDX);          // f_T12 = f_CCU6 / prescaler = 12.5 MHz
364     CCU61_TCTR0.U |= 0x2 << T12CLK_BIT_LSB_IDX;              // f_CCU6 = 50 MHz, prescaler = 4
365
366     CCU61_TCTR0.U |= 0x1 << T12PRE_BIT_LSB_IDX;             // f_T12 = f_CCU6 / 256 = 48,828 Hz
367
368
369     CCU61_T12PR.U = 100000 -1;                                // PM interrupt freq. = f_T12 / (T12PR + 1)
370     CCU61_TCTR4.U |= 0x1 << T12STR_BIT_LSB_IDX;            // load T12PR from shadow register
371
372     CCU61_T12.U = 0;                                         // clear T12 counter register
373 }
```

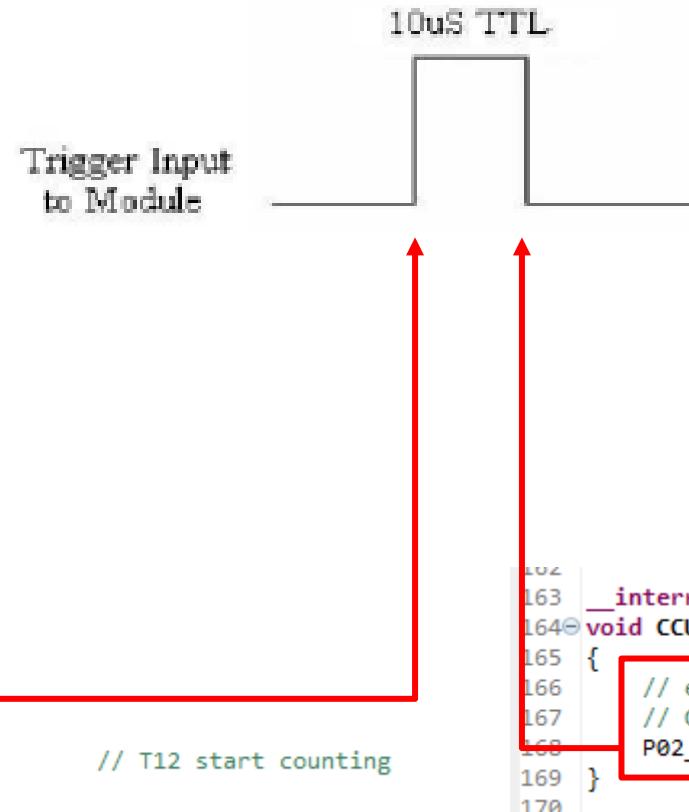
CCU61 모듈의 T12
카운터 사용을 위한
레지스터 설정

CCU61 모듈 T12
타이머 카운터
0으로 초기화
(아직 카운팅
시작하지 않음)

초음파 센서 거리 측정 함수 및 ISR 구현

Lab16: Trigger 펄스 생성 함수 작성

:CCU60 모듈 10us Period Match Interrupt 사용



10us 후, CCU61 모듈 T12에서 PM
Interrupt 발생으로 수행되는 ISR

Lab17: Echo 펄스 시간 측정 함수 작성

:ERU Rising / Falling Edge ISR 사용

```
132
133     unsigned int range;
134     unsigned char range_valid_flag = 0;
135
136
137     _interrupt(0x0A) __vector_table(0)
138 @ void ERU0_ISR(void)
139 {
140     if( (P00_IN.U & (0x1 << P4_BIT_LSB_IDX)) )
141     {
142         // echo
143         CCU61_TCTR4.U = 0x1 << T12RS_BIT_LSB
144     }
145     else
146     {
147         // echo
148     }
149
150
151
152
153     CCU61_TCTR4.B.T12RR = 0x1;           // stop CCU61 T12 counter
154
155     // (1 / t_freq) * counter * 1000000 / 58 = centimeter
156     range = ((CCU61_T12.B.T12CV * 1000000) / 48828) / 58;
157     range_valid_flag = 1;
158
159     CCU61_TCTR4.B.T12RES = 0x1;           // reset CCU61 T12 counter
160
161 }
162 }
```

거리 결과 저장할 전역 변수
거리 측정이 완료되었음을 main 함수에 알려줄 전역 변수

The diagram shows a timing sequence. On the left, there is a logic level diagram for the 'echo' signal. It starts at a low level, goes high for a short duration (the trigger pulse), and then returns to a low level. On the right, there is a sequence of assembly code lines from 506 to 514. Line 506 calls the 'usonicTrigger' function. Line 511 contains the assignment 'range_valid_flag = 0;'. A red box highlights this assignment, and a red arrow points from it to the text '거리 측정 시작 시점에서 flag clear' (Clear flag at the start of distance measurement).

```
506 @ void usonicTrigger(void)
507 {
508     // start of 10us Trigger Pulse
509     // GPIO P02.6 --> HIGH
510     P02_OUT.U |= 0x1 << P6_BIT_LSB_IDX;
511     range_valid_flag = 0;
512     CCU61_TCTR4.U = 0x1 << T12RS_BIT_LSB_IDX;
513 }
514 // T12 start counting
```

거리 측정 시작 시점에서 flag clear

Lab18: Echo 펄스 시간 측정 함수 작성

:ERU Rising / Falling Edge ISR 사용

```
132  
133 unsigned int range;  
134 unsigned char range_valid_flag = 0;  
135  
136  
137 __interrupt(0x0A) __vector_table(0)  
138 void ERU0_ISR(void)  
139 {  
140     if( (P00_IN.U & (0x1 << P4_BIT_LSB_IDX)) != 0 )      // rising edge of echo  
141     {  
142         //  
143         // echo _____|  
144         //           ^  
145         CCU61_TCTR4.U = 0x1 << T12RS_BIT_LSB_IDX;          // start CCU61 T12 counter  
146     }  
147     else  
148     {  
149         //  
150         // echo _____|  
151         //           ^  
152  
153         CCU61_TCTR4.B.T12RR = 0x1;             // stop CCU61 T12 counter  
154  
155         // (1 / t_freq) * counter * 1000000 / 58 = centimeter  
156         range = ((CCU61_T12.B.T12CV * 1000000) / 48828) / 58;  
157         range_valid_flag = 1;  
158  
159         CCU61_TCTR4.B.T12RES = 0x1;             // reset CCU61 T12 counter  
160     }  
161 }  
162 }
```

Echo Pulse Output
to User Timing Circuit

Input TTL lever
signal with a range
in proportion

Rising Edge 조건

Falling Edge 조건

Lab19: Echo 펄스 시간 측정 함수 작성

:CCU61 모듈 타이머 카운터 사용 - 카운팅 시작

```

132
133 unsigned int range;
134 unsigned char range_valid_flag = 0;
135
136
137 __interrupt(0x0A) __vector_table(0)
138 void ERU0_ISR(void)
139 {
140     if( (P00_IN.U & (0x1 << P4_BIT_LSB_IDX)) != 0 )      // rising edge of echo
141     {
142         //
143         // echo _____|_____
144         //
145         CCU61_TCTR4.U = 0x1 << T12RS_BIT_LSB_IDX;           // start CCU61 T12 counter
146     }
147     else
148     {
149         //
150         // echo _____|_____
151         //
152
153         CCU61_TCTR4.B.T12RR = 0x1;                         // stop CCU61 T12 counter
154
155         // (1 / t_freq) * counter * 1000000 / 58 = centimeter
156         range = ((CCU61_T12.B.T12CV * 1000000) / 48828) / 58;
157         range_valid_flag = 1;
158
159         CCU61_TCTR4.B.T12RES = 0x1;                         // reset CCU61 T12 counter
160
161     }
162 }
```

Echo Pulse Output
to User Timing Circuit

Input TTL lever
signal with a range
in proportion

Echo 신호의
시작과 동시에
카운팅 시작

TCTR4 Timer Control Register 4 (78H)																Reset Value: 0000 0000H			
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16				
TCTR4 레지스터 @ 0xF0002B78																			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0				
T13	T13	T13	0	T13	T13	T13	T12	T12	T12	T12	0	DT	T12	T12	T12	RR	RR	RR	
STD	STR	CNT	r	RES	RS	RR	STD	STR	CNT	w	w	RES	RS	RS	w	w	w	w	

T12RS 1 w Timer T12 Run Set
Setting this bit sets the T12R bit.
0 T12R is not influenced.
1 T12R is set, T12 starts counting.

Lab20: Echo 펄스 시간 측정 함수 작성

:CCU61 모듈 타이머 카운터 사용 - 카운팅 종료

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

```

132
133 unsigned int range;
134 unsigned char range_valid_flag = 0;
135
136
137 __interrupt(0x0A) __vector_table(0)
138 void ERU0_ISR(void)
139 {
140     if( (P00_IN.U & (0x1 << P4_BIT_LSB_IDX)) != 0 )      // rising edge of echo
141     {
142         //
143         // echo _____
144         //
145         CCU61_TCTR4.U = 0x1 << T12RS_BIT_LSB_IDX
146
147     }
148     else
149     {
150         //
151         // echo _____
152
153         CCU61_TCTR4.B.T12RR = 0x1;          // stop CCU61 T12 counter
154
155         // (1 / t_freq) * counter * 1000000 / 58 = centimeter
156         range = ((CCU61_T12.B.T12CV * 1000000) / 48828) / 58;
157         range_valid_flag = 1;
158
159         CCU61_TCTR4.B.T12RES = 0x1;          // reset CCU61 T12 counter
160
161     }
162 }
```

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
TCTR4 레지스터 @ 0xF0002B78															
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	w

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.

Echo Pulse Output
to User Timing Circuit

Input TTL lever
signal with a range
in proportion

Echo 신호가 끝나면 카운팅 종료

Lab21: Echo 펄스 시간 측정 함수 작성

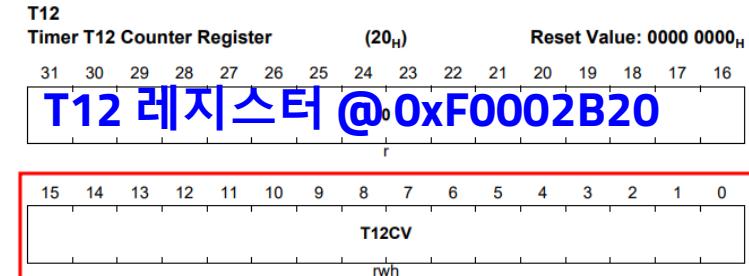
:CCU61 모듈 타이머 카운터 값 us 단위 환산

```

136
137 __interrupt(0x0A) __vector_table(0)
138 void ERU0_ISR(void)
139 {
140     if( (P00_IN.U & (0x1 << P4_BIT_LSB_IDX)) != 0 )      // rising edge of echo
141     {
142         //
143         // echo _____|_____
144         //
145         CCU61_TCTR4.U = 0x1 << T12RS_BIT_LSB_IDX;           // start CCU61 T12 counter
146     }
147     else
148     {
149         //
150         // echo _____|_____
151         //
152         CCU61_TCTR4.B.T12RR = 0x1;                          // stop CCU61 T12 counter
153
154         // (1 / t_freq) * counter * 1000000 / 58 = centimeter
155         range = ((CCU61_T12.B.T12CV * 1000000) / 48828) / 58;
156         range_valid_flag = 1;
157
158         CCU61_TCTR4.B.T12RES = 0x1;                          // reset CCU61 T12 counter
159     }
160 }
161
162

```

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



Field	Bits	Type	Description
T12CV	[15:0]	rwh	Timer 12 Counter Value This register represents the 16-bit counter value of Timer12.

T12 레지스터의 T12CV 영역에서
counter 값 확인

Counter는 $(1 / 48,828)$ 초 당 1씩 증가 (CCU61 모듈 T12 clock 주파수가 48,828 Hz 이므로)
→ 마이크로 초(us) 단위로 환산하려면
 $(\text{counter} * 1000000) * (1 / 48828) = \text{Echo 신호 시간 길이 (us)}$

Lab22: Echo 펄스 시간 측정 함수 작성 :타이머로 측정한 us 시간으로 거리(cm) 계산

```

136
137 __interrupt(0x0A) __vector_table(0)
138 void ERU0_ISR(void)
139 {
140     if( (P00_IN.U & (0x1 << P4_BIT_LSB_IDX)) != 0 )      // rising edge of echo
141     {
142         //
143         // echo _____|_____
144         CCU61_TCTR4.U = 0x1 << T12RS_BIT_LSB_IDX;           // start CCU61 T12 counter
145     }
146     else
147     {
148         //
149         // echo _____|_____
150
151         CCU61_TCTR4.B.T12RR = 0x1;                         // stop CCU61 T12 counter
152
153         // (1 / t_freq) * counter * 1000000 / 58 = centimeter
154         range = ((CCU61_T12.B.T12CV * 1000000) / 48828) / 58;
155         range_valid_flag = 1;
156
157         CCU61_TCTR4.B.T12RES = 0x1;                         // reset CCU61 T12 counter
158     }
159 }
160
161
162

```

초음파 센서 거리 측정이
완료되었음을 flag로 표시



HC-SR04.pdf p.2

초음파 센서 데이터 시트에 따르면,
거리를 구하는 공식
→ $us / 58 = \text{센티미터 거리(cm)}$
→ 거리 global 변수에 저장

Lab23: Echo 펄스 시간 측정 함수 작성

:CCU61 모듈 타이머 카운터 리셋

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

```

136
137     __interrupt(0x0A) __vector_table(0)
138     void ERU0_ISR(void)
139     {
140         if( (P00_IN.U & (0x1 << P4_BIT_LSB_IDX)) != 0 )      // rising edge of echo
141         {
142             //
143             // echo _____|_____
144             //
145             CCU61_TCTR4.U = 0x1 << T12RS_BIT_LSB_IDX;           // start CCU61 T12 counter
146         }
147     else
148     {
149         //
150         // echo _____|_____
151         //
152
153         CCU61_TCTR4.B.T12RR = 0x1;          // stop CCU61 T12 counter
154
155         // (1 / t_freq) * counter * 1000000 / 58 = centimeter
156         range = ((CCU61_T12.B.T12CV * 1000000) / 48828) / 58;
157         range_valid_flag = 1;
158
159         CCU61_TCTR4.B.T12RES = 0x1;          // reset CCU61 T12 counter
160     }
161 }
162

```

TCTR4
Timer Control Register 4 (78H)
Reset Value: 0000 0000H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TCTR4 레지스터 @ 0xF0002B78															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
T13	T13	T13		0	T13	T13	T13	T12	T12	T12	0	DT	T12	T12	T12
STD	STR	CNT		r	RES	RS	RR	STD	STR	CNT	r	RES	RS	RR	
w	w	w		r	w	w	w	w	w	w	r	w	w	w	w

T12RES | 2 | w | Timer T12 Reset
⁰ⁿ No effect on T12.
^{1B} The T12 counter register is cleared to zero. The switching of the output signals is according to the switching rules. Setting of T12RES has no impact on bit T12R.

다음 Echo 신호 측정이 가능하도록 CCU61 T12 타이머 카운터 리셋

초음파 센서 거리 측정 MAIN 함수 구현

Lab24: 초음파 센서 거리 측정 main 함수 작성

:전체적 코드 흐름

```
173 int core0_main(void)
174 {
175     IfxCpu_enableInterrupts();
176
177     /* !!WATCHDOG AND SAFETY WATCHDOG ARE DISABLED HERE!!
178     * Enable the watchdogs and service them periodically if it is required
179     */
180     IfxScuWdt_disableCpuWatchdog(IfxScuWdt_getCpuWatchdogPassword());
181     IfxScuWdt_disableSafetyWatchdog(IfxScuWdt_getSafetyWatchdogPassword());
182
183     /* Wait for CPU sync event */
184     IfxCpu_emitEvent(&g_cpuSyncEvent);
185     IfxCpu_waitEvent(&g_cpuSyncEvent, 1);
186
187     initERU();
188     initCCU60();
189     initCCU61();
190     initLED();
191     initRGBLED();
192     //initVADC();
193     //initGTM();
194     //initButton();
195     initUSonic();
196 }
```

초기화 함수 호출

Lab25: 초음파 센서 거리 측정 main 함수 작성 :전체적 코드 흐름

```
196
197
198
199 while(1)
{
200     for(unsigned int i = 0; i < 10000000; i++)
201         usonicTrigger();
202         while( range_valid_flag == 0 );
203
204         if( range >= 60 ) // red
205         {
206             P02_OUT.U |= 0x1 << P7_BIT_LSB_IDX;
207             P10_OUT.U &= ~(0x1 << P5_BIT_LSB_IDX);
208             P10_OUT.U &= ~(0x1 << P3_BIT_LSB_IDX);
209         }
210         else if( range >= 40 ) // green
211         {
212             P02_OUT.U &= ~(0x1 << P7_BIT_LSB_IDX);
213             P10_OUT.U |= 0x1 << P5_BIT_LSB_IDX;
214             P10_OUT.U &= ~(0x1 << P3_BIT_LSB_IDX);
215         }
216         else if( range >= 20 ) // blue
217         {
218             P02_OUT.U &= ~(0x1 << P7_BIT_LSB_IDX);
219             P10_OUT.U &= ~(0x1 << P5_BIT_LSB_IDX);
220             P10_OUT.U |= 0x1 << P3_BIT_LSB_IDX;
221         }
222         else // white
223         {
224             P02_OUT.U |= 0x1 << P7_BIT_LSB_IDX;
225             P10_OUT.U |= 0x1 << P5_BIT_LSB_IDX;
226             P10_OUT.U |= 0x1 << P3_BIT_LSB_IDX;
227         }
228     }
229     return (1);
230 }
```

초음파 센서 Trigger 펄스 생성
→ 거리 측정 시작

Lab26: 초음파 센서 거리 측정 main 함수 작성 :전체적 코드 흐름

```
196
197     while(1)
198     {
199         for(unsigned int i = 0; i < 10000000; i++)
200             usonicTrigger();
201         while( range_valid_flag == 0 );
202
203         if( range >= 60 ) // red
204         {
205             P02_OUT.U |= 0x1 << P7_BIT_LSB_IDX;
206             P10_OUT.U &= ~(0x1 << P5_BIT_LSB_IDX);
207             P10_OUT.U &= ~(0x1 << P3_BIT_LSB_IDX);
208         }
209         else if( range >= 40 ) // green
210         {
211             P02_OUT.U &= ~(0x1 << P7_BIT_LSB_IDX);
212             P10_OUT.U |= 0x1 << P5_BIT_LSB_IDX;
213             P10_OUT.U &= ~(0x1 << P3_BIT_LSB_IDX);
214         }
215         else if( range >= 20 ) // blue
216         {
217             P02_OUT.U &= ~(0x1 << P7_BIT_LSB_IDX);
218             P10_OUT.U &= ~(0x1 << P5_BIT_LSB_IDX);
219             P10_OUT.U |= 0x1 << P3_BIT_LSB_IDX;
220         }
221         else // white
222         {
223             P02_OUT.U |= 0x1 << P7_BIT_LSB_IDX;
224             P10_OUT.U |= 0x1 << P5_BIT_LSB_IDX;
225             P10_OUT.U |= 0x1 << P3_BIT_LSB_IDX;
226         }
227
228     }
229     return (1);
230 }
```

거리 측정 완료될 때까지 대기

Lab27: 초음파 센서 거리 측정 main 함수 작성 :전체적 코드 흐름

```
196
197     while(1)
198     {
199         for(unsigned int i = 0; i < 10000000; i++);
200         usonicTrigger();
201         while( range_valid_flag == 0 );
202
203         if( range >= 60 ) // red
204         {
205             P02_OUT.U |= 0x1 << P7_BIT_LSB_IDX;
206             P10_OUT.U &= ~(0x1 << P5_BIT_LSB_IDX);
207             P10_OUT.U &= ~(0x1 << P3_BIT_LSB_IDX);
208         }
209         else if( range >= 40 ) // green
210         {
211             P02_OUT.U &= ~(0x1 << P7_BIT_LSB_IDX);
212             P10_OUT.U |= 0x1 << P5_BIT_LSB_IDX;
213             P10_OUT.U &= ~(0x1 << P3_BIT_LSB_IDX);
214         }
215         else if( range >= 20 ) // blue
216         {
217             P02_OUT.U &= ~(0x1 << P7_BIT_LSB_IDX);
218             P10_OUT.U &= ~(0x1 << P5_BIT_LSB_IDX);
219             P10_OUT.U |= 0x1 << P3_BIT_LSB_IDX;
220         }
221         else // white
222         {
223             P02_OUT.U |= 0x1 << P7_BIT_LSB_IDX;
224             P10_OUT.U |= 0x1 << P5_BIT_LSB_IDX;
225             P10_OUT.U |= 0x1 << P3_BIT_LSB_IDX;
226         }
227     }
228
229     return (1);
230 }
```

측정한 거리 값에 따라 RGB LED 색상
결정

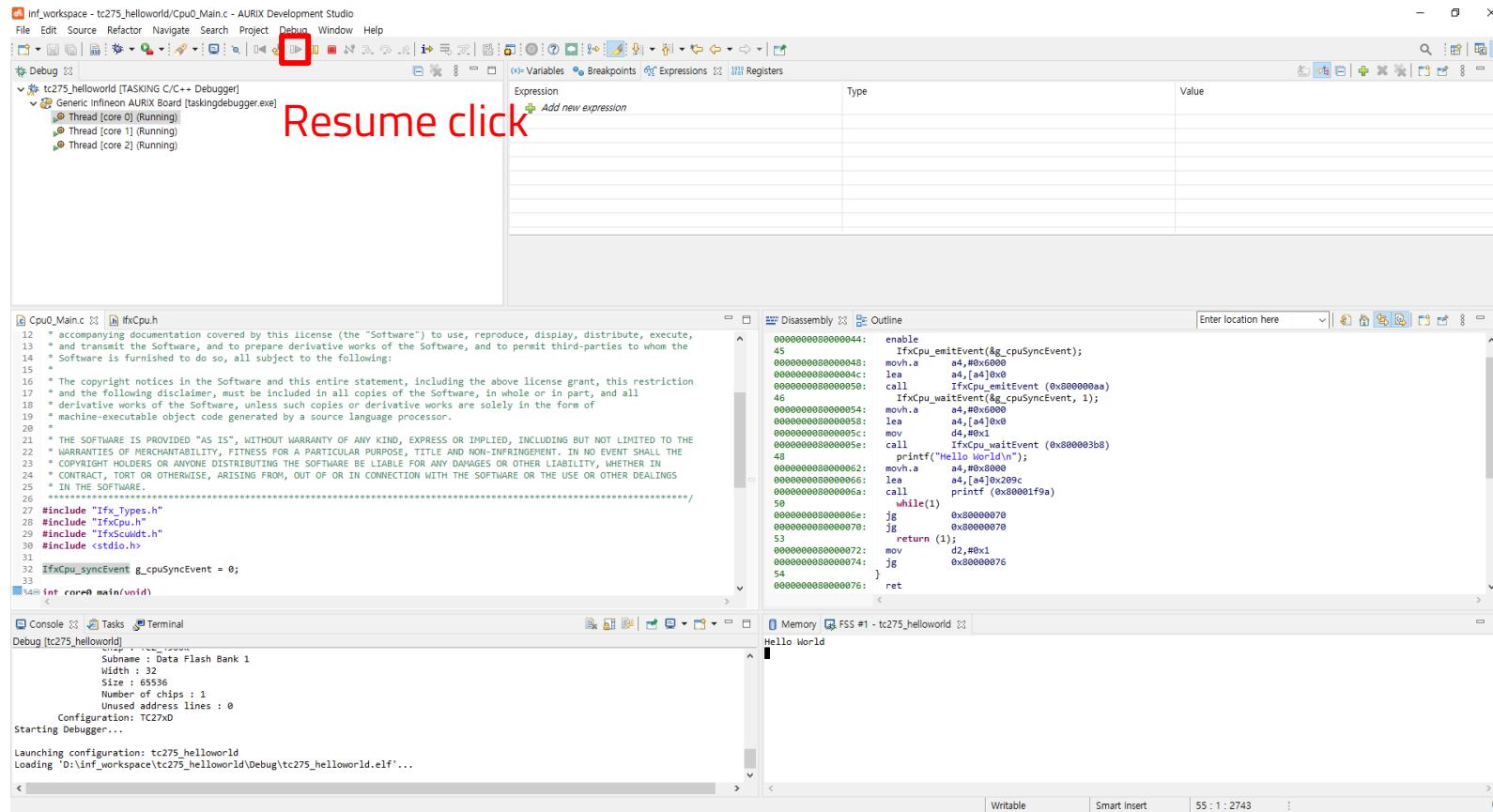
[HC-SR04.pdf p.1](#)

Working Voltage	DC 5 V
Working Current	15mA
Working Frequency	40Hz
Max Range	4m
Min Range	2cm
MeasuringAngle	15 degree
Trigger Input Signal	10uS TTL pulse
Echo Output Signal	Input TTL lever signal and the range in proportion
Dimension	45*20*15mm

초음파 센서 스펙 참고

Build 및 Debug

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



동작 확인

- 초음파 센서와의 거리에 따라 RGB LED의 색상이 달라지는 모습 확인 가능

감사합니다. 휴식~~