

임베디드 기반 SW 개발 프로젝트

AURIX TC275 보드 PWM 신호로부터
아날로그 신호 생성

Digital DAC를 구현하여 스피커를 디지털로 구동

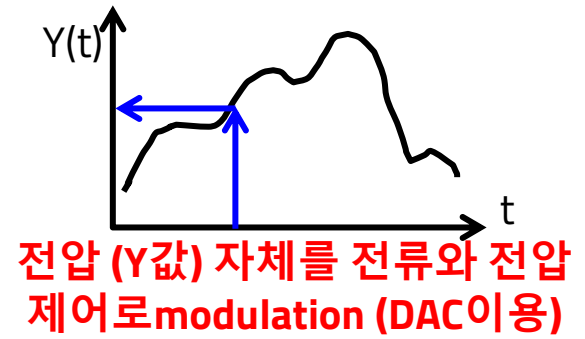
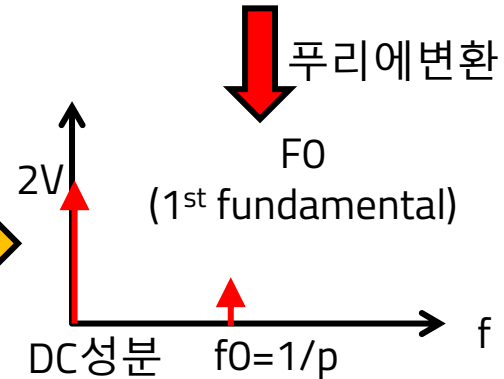
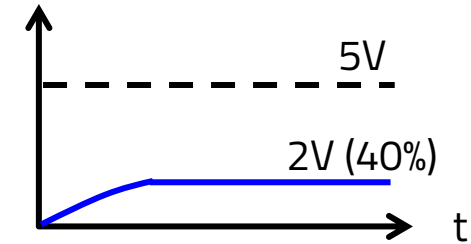
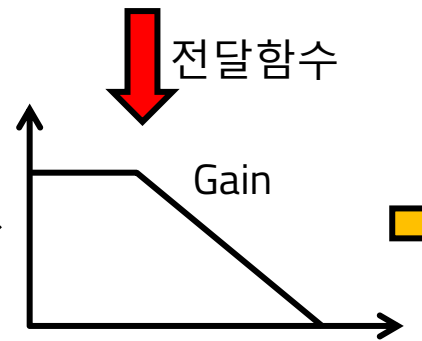
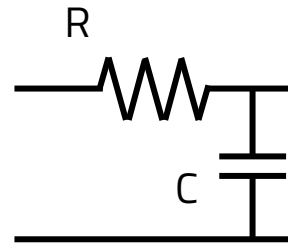
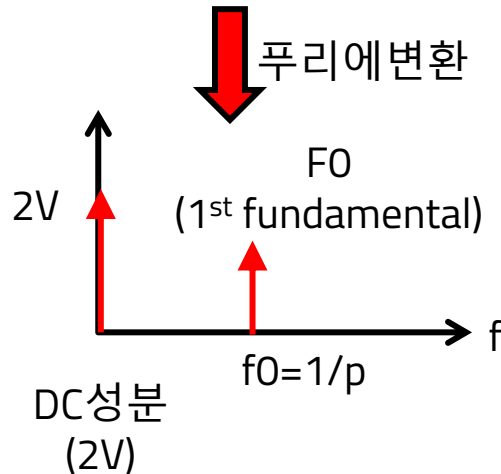
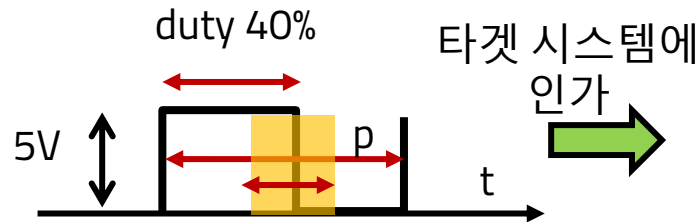
현대자동차 입문교육
박대진 교수

실습 목표

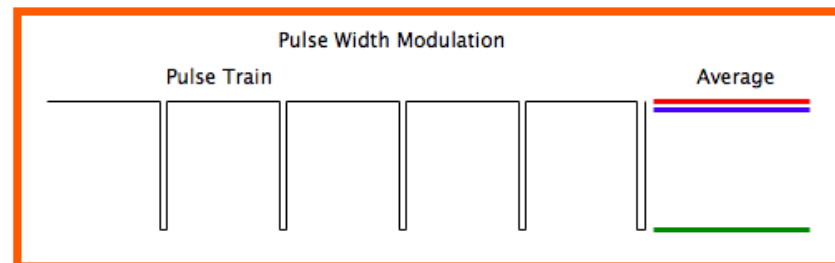
- 보드에서 생성되는 PWM 신호와 저항(R), 커패시터(C)로 구성된 저주파 통과 필터 (LPF: Low Pass Filter)를 결합하여 **아날로그 sin 신호를 생성해본다.**

T축의 변조로, Y값을 제어하는 방법

전압 Y를 고정(5V) 시켜두고
t축 Edge를 modulation

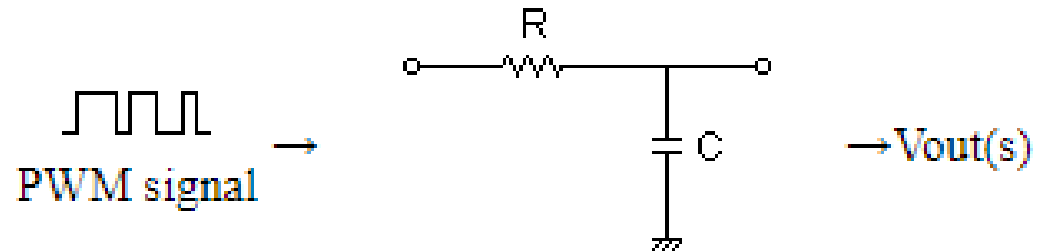
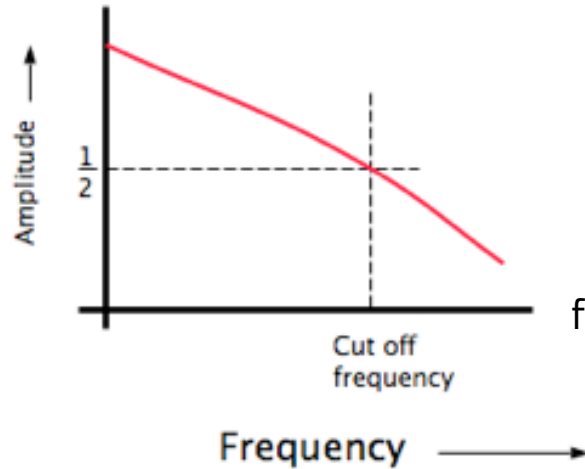


전압 (Y값) 자체를 전류와 전압
제어로 modulation (DAC이용)



Cut-off Frequency

- Cut-off freq.
 - $f = 1/(2\pi RC)$



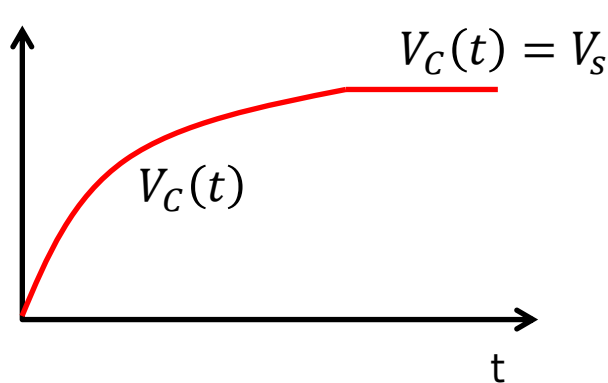
$$G(s) = \frac{1/sC}{R + 1/sC} = \frac{1}{1 + RCs} = \frac{\frac{1}{RC}}{s + \frac{1}{RC}}$$

$s = \frac{1}{RC}$ 일 때 $G(s)$ 가 $\frac{1}{2}$ 이 된다.

$2\pi f = \frac{1}{RC} \rightarrow f_c = \frac{1}{2\pi RC}$ 가 cut off 주파수가 된다

$R=3.3k, C=11\mu$ 일 때 $f_c = 4.37Hz$

Rising/Falling (Settling) Time 발생 이유



$$V_C(t) = V_s(1 - e^{-\frac{1}{RC}t})$$

$$t = RC \text{ 일 때 } e^{-\frac{1}{RC}t} = e^{-1} = 0.368$$

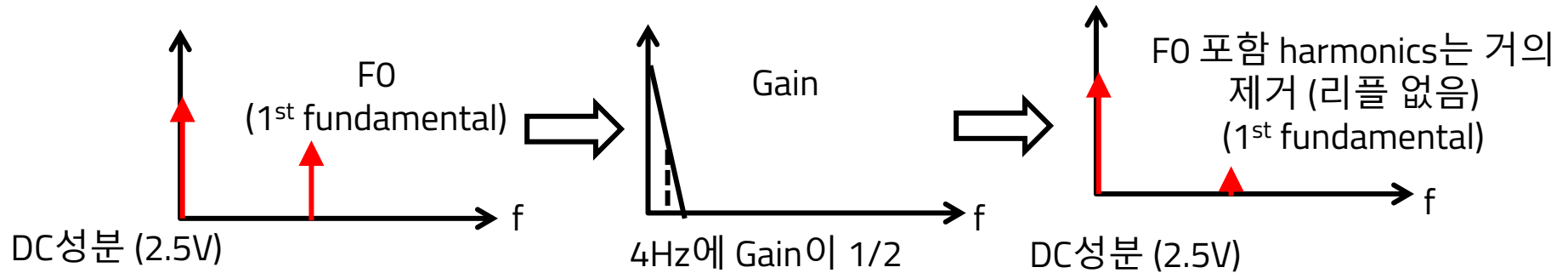
RC time constant : $t = RC \rightarrow 63.2\%$ charge from zero

$$\text{rise time (20\% to 80\%)} \quad t_r \approx 1.4\tau \approx \frac{0.22}{f_c}$$

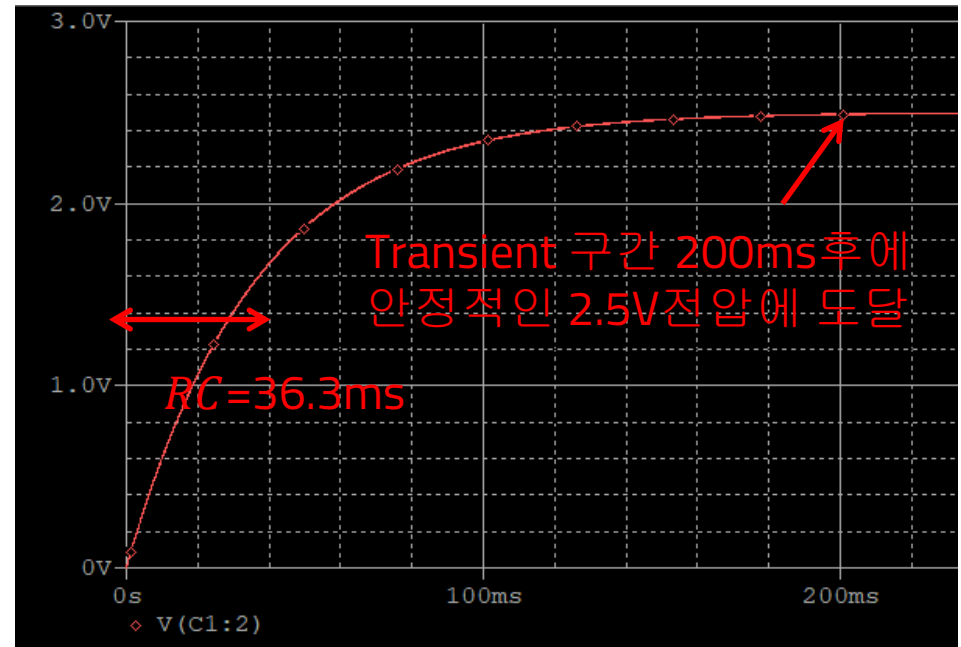
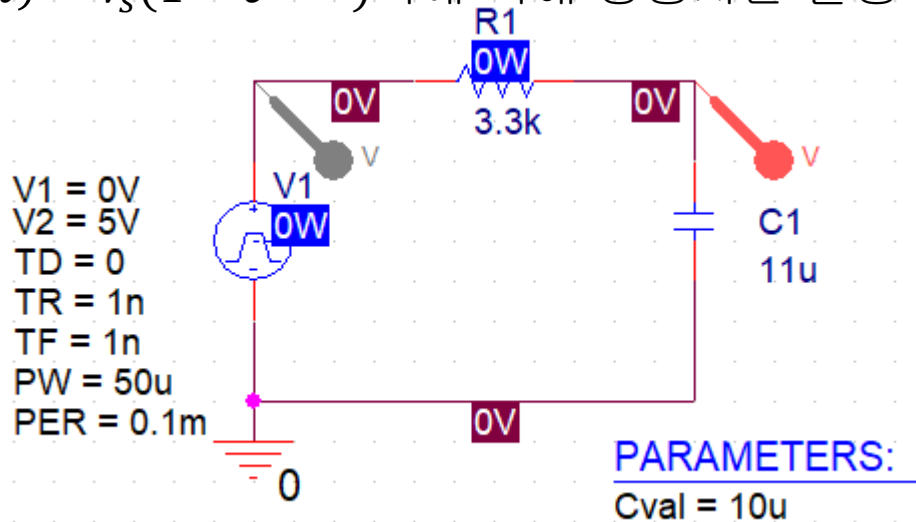
$$\text{rise time (10\% to 90\%)} \quad t_r \approx 2.2\tau \approx \frac{0.35}{f_c}$$

Settling 에 걸리는 시간

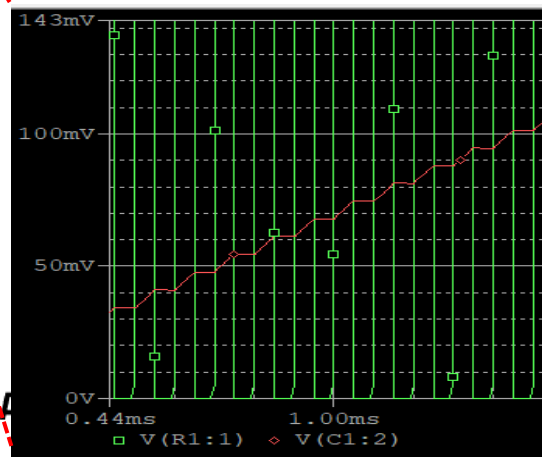
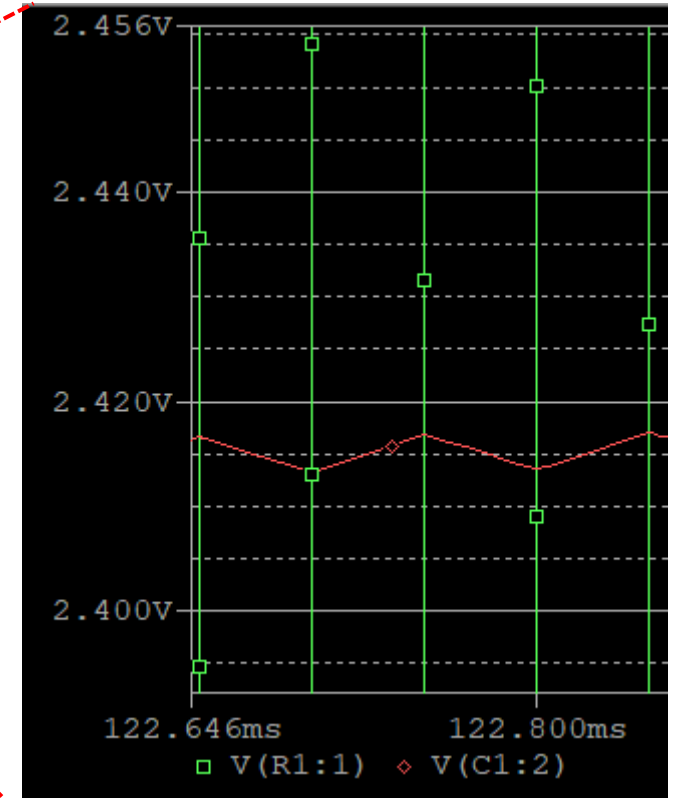
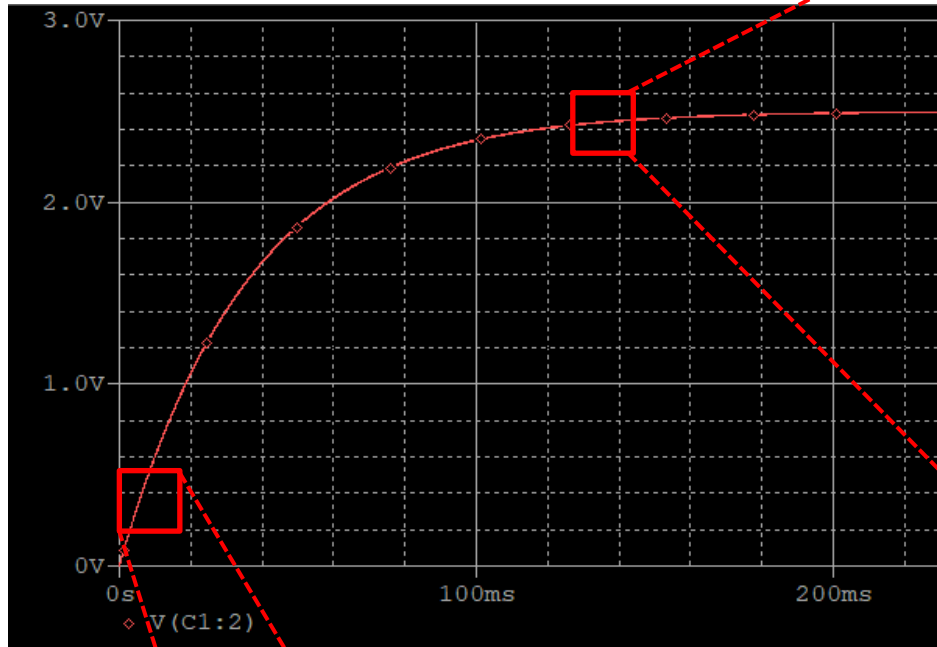
- 10kHz PWM input, duty 50%, 약 cut-freq. 4Hz, settling time 약 200ms



$V_c(t) = V_s(1 - e^{-\frac{1}{RC}t})$ 식에 의해 상승시간 발생



Transient Simulation

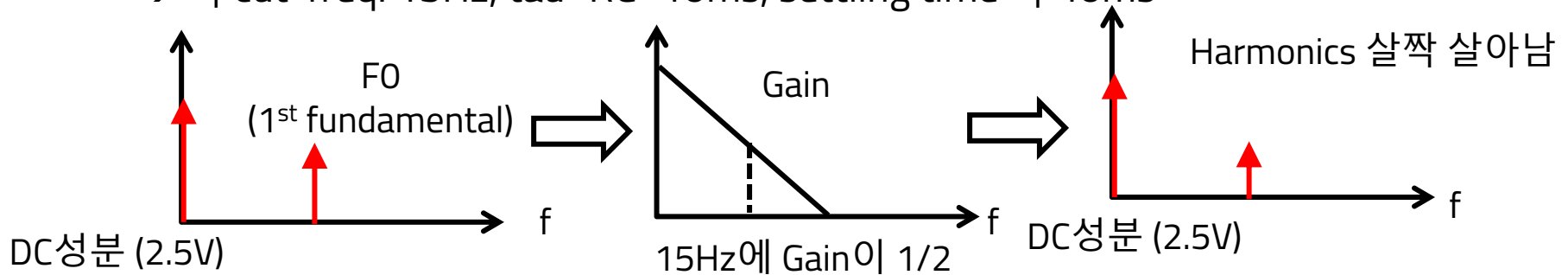


PWM Pulse Duty 바꾸면 최대 대략 200ms
씩 Delay가 생김 (R,C필터에 의해)
→ Delay줄 일려면 filter 설계 조정

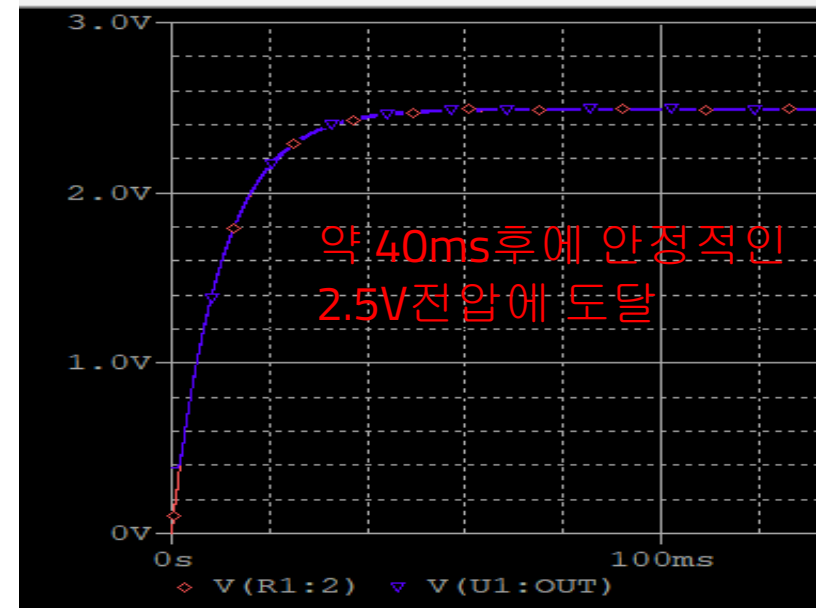
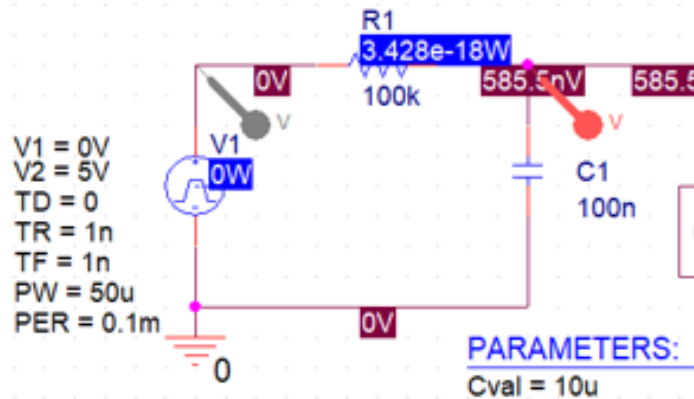
→ PWM을 200ms 정도 유지해야 하니 5Hz 속
도로 전압 가변가능

Settling 도달시간 줄이기 (Pole 이동)

- 10kHz PWM input, duty 50%, R=100k, C=100nF
 - 약 cut-freq. 15Hz, $\tau=RC=10\text{ms}$, settling time 약 40ms



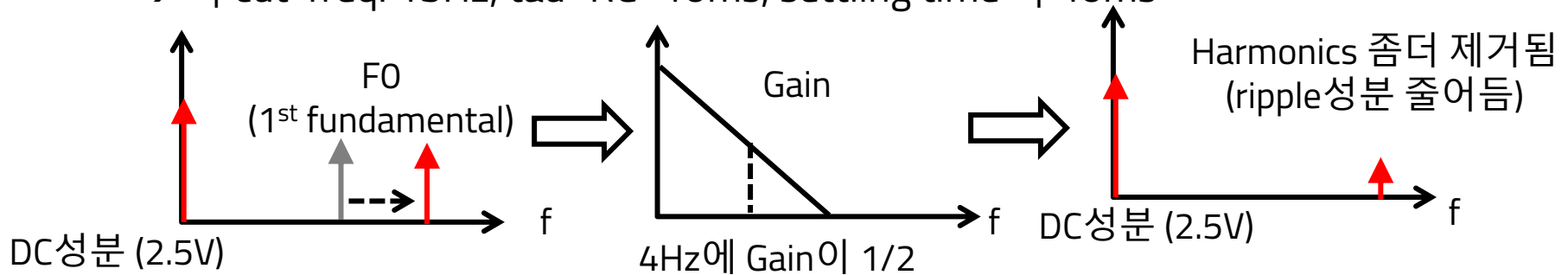
$V_c(t) = V_s(1 - e^{-\frac{1}{RC}t})$ 식에 의해 상승시간 발생
RC가 작아져서 100%도달하기까지 t도 작아짐



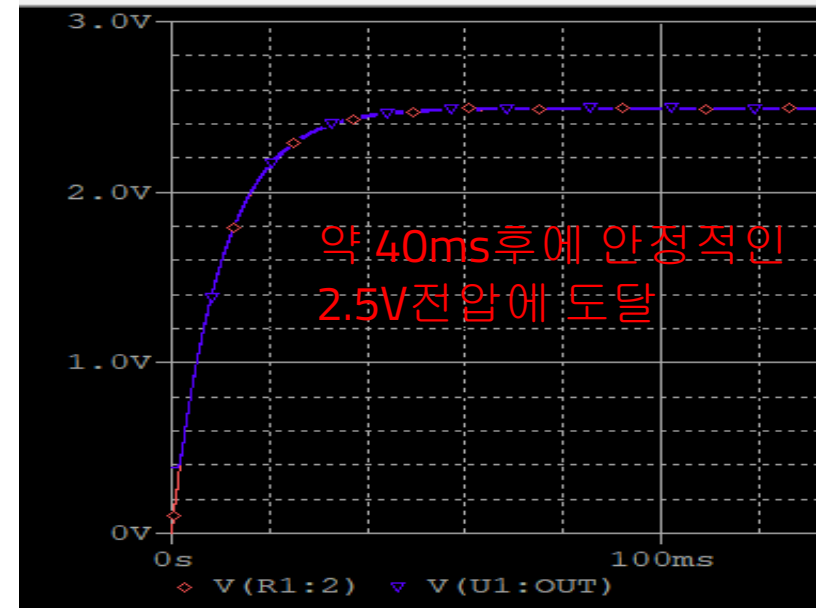
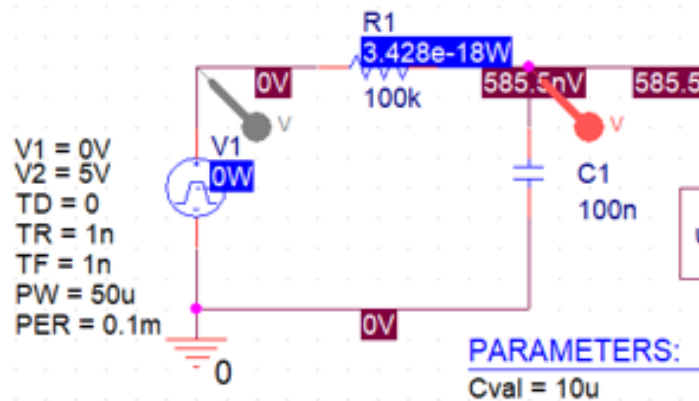
→ 10kHz PWM pulse ($1/10\text{k}=0.1\text{ms}$) 반복을 40ms 정도 유지해야 하니 pulse 400번 반복해야 정상 전압 도착. $1/40\text{m}=25\text{Hz}$ 속도로 전압 가변가능

리플 특성 개선 (입력 PWM속도 상승)

- 10k → 100KHz PWM input, duty 50%, R=100k, C=100nF
 - 약 cut-freq. 15Hz, $\tau=RC=10\text{ms}$, settling time 약 40ms



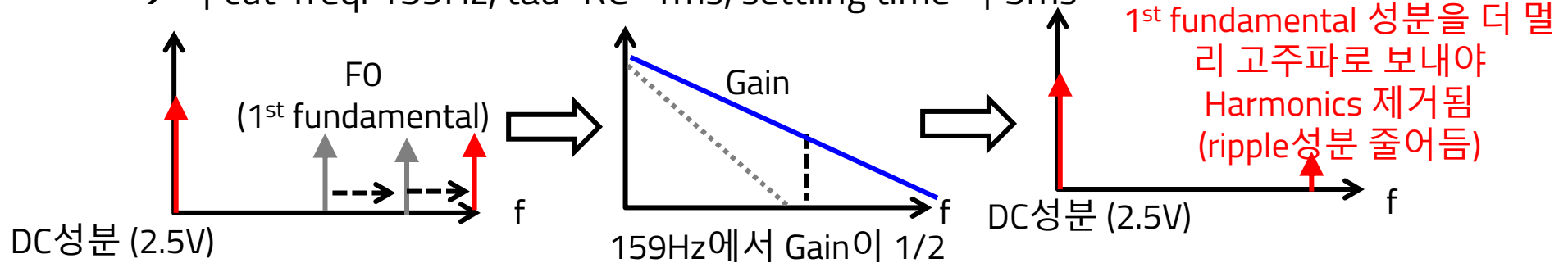
$V_c(t) = V_s(1 - e^{-\frac{1}{RC}t})$ 식에 의해 상승시간 발생
RC는 같은 값이므로, 상승시간 변화 없음.



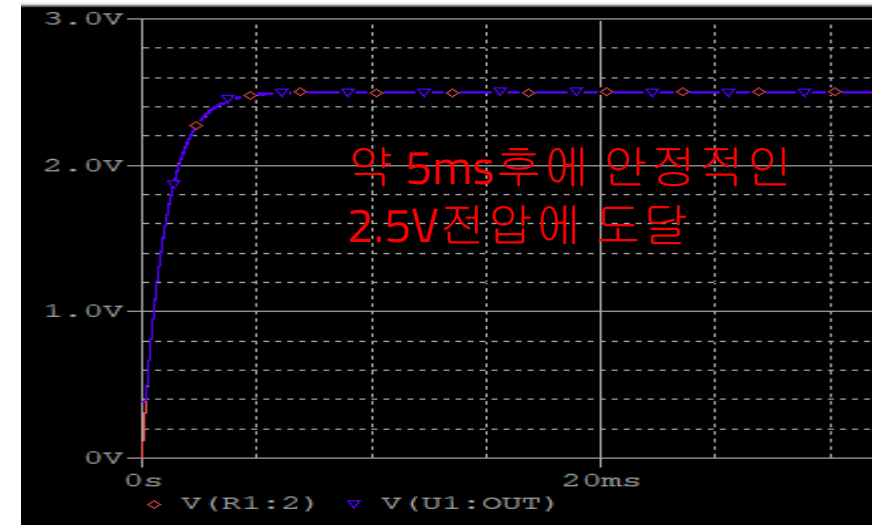
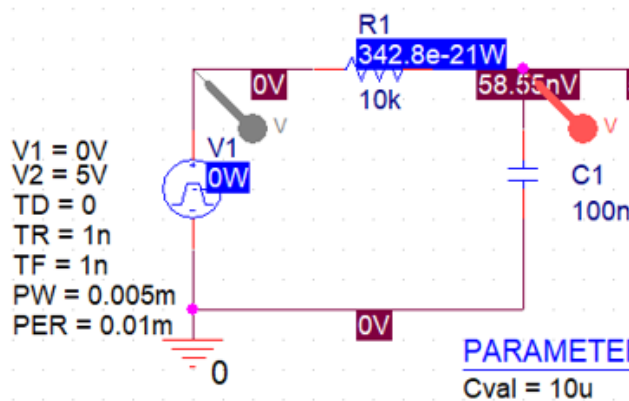
→ RC는 동일하므로, 상승하는 시간에는 영향을 줄 수 없음, 리플 특성은 좀더 좋아짐.

상승속도, 리플특성 개선 (Pole이동, PWM고속)

- 10k → 100KHz → 1000kHz PWM input, duty 50%, R=10k, C=100nF
 - 약 cut-freq. 159Hz, $\tau=RC=1\text{ms}$, settling time 약 5ms



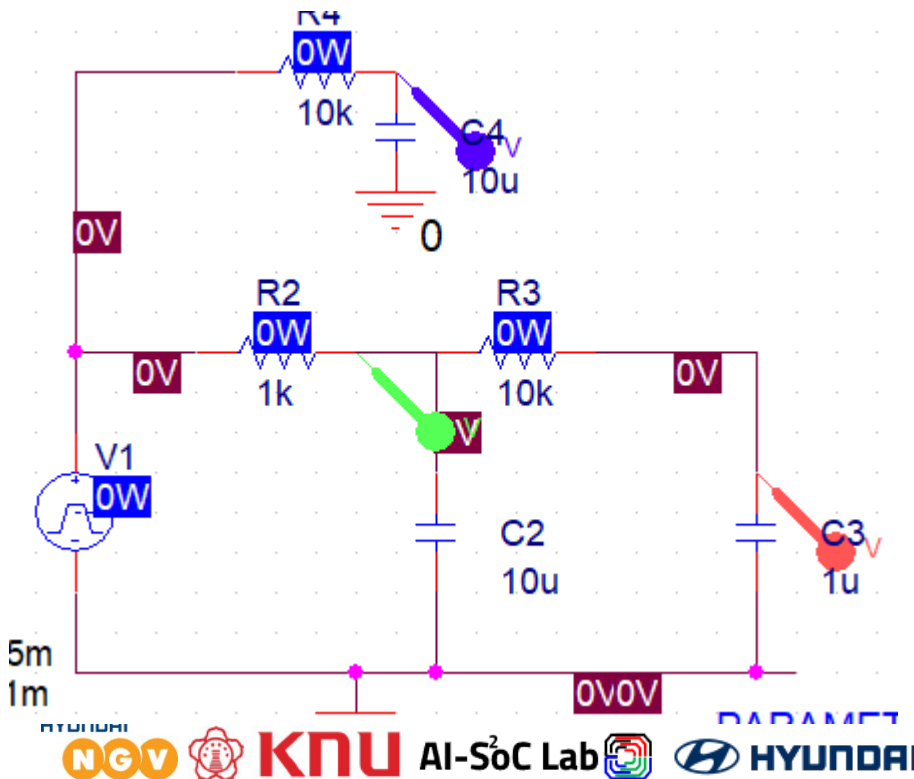
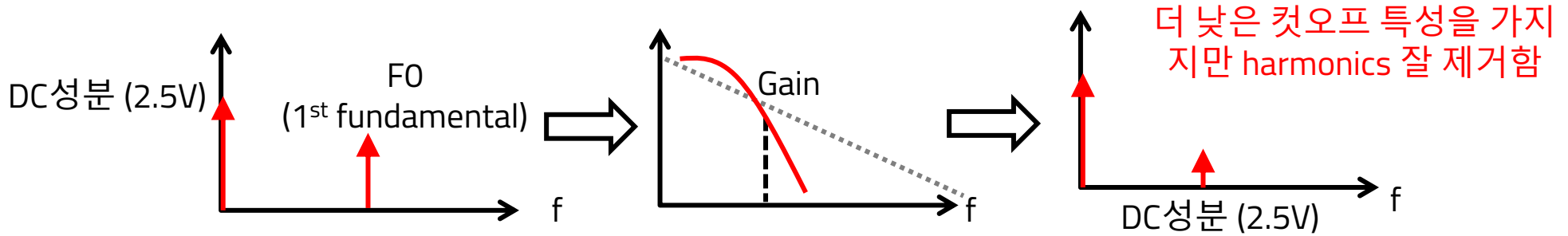
$V_c(t) = V_s(1 - e^{-\frac{1}{RC}t})$ 식에 의해 상승시간 발생
RC는 작아져, 상승시간도 짧아짐.



→ RC는 동일하므로, 상승하는 시간에는 영향을 줄 수 없음, 리플 특성은 좀더 좋아짐.

회로 개선을 통해 PWM 주파수 다운 가능

- 2nd order circuit settling time



Response도 빠르면서 리플 제거



PWM 신호 생성 flow

:TOM에서 PWM 출력 신호 생성

- PWM 기능을 사용하기 위한 TOM의 구조
- TOM은 TGC (TOM Global Channel Control)의 제어를 받아 출력 신호를 생성
 - TOM의 기능 중, PWM 기능을 사용하기 위한 설정 필요
 - TOM의 출력은 MCU의 핀으로 바로 출력 가능, 사용하려는 핀이 TOM의 어떤 Channel과 연결되어 있는지 확인 필요

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.2918

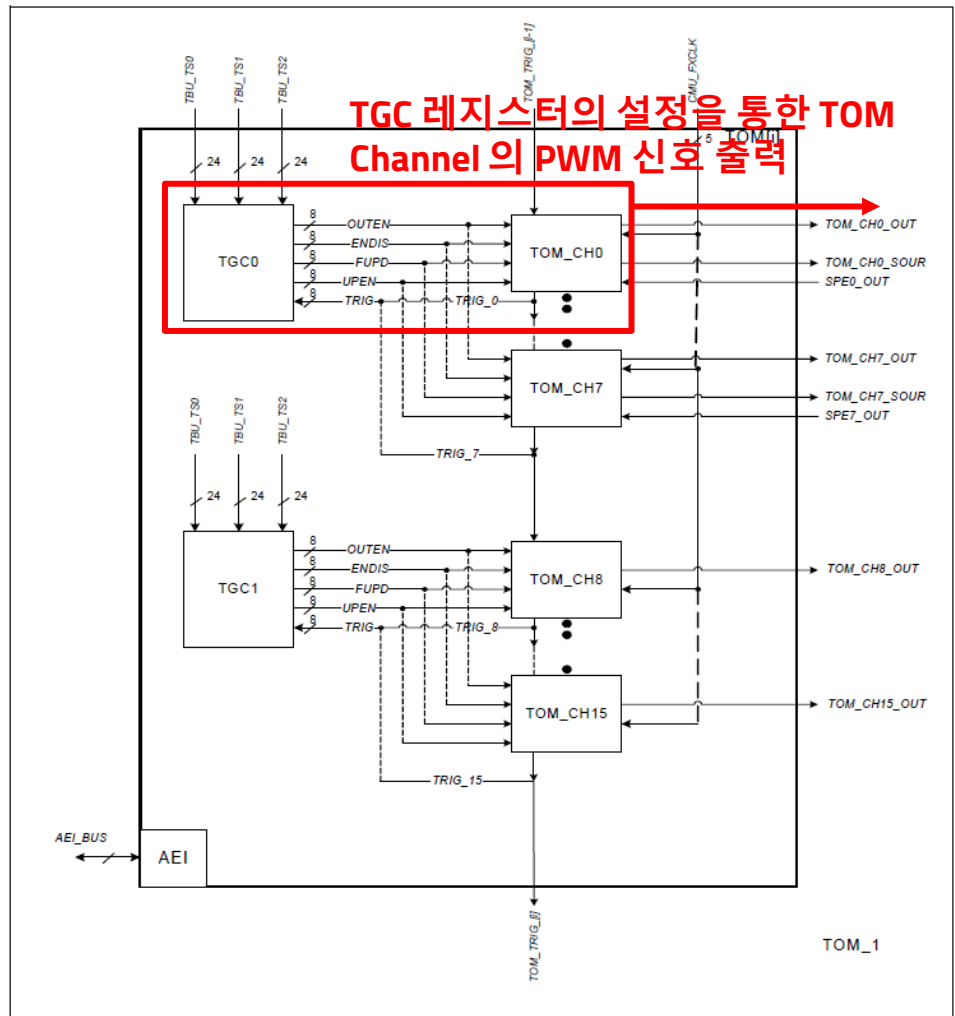
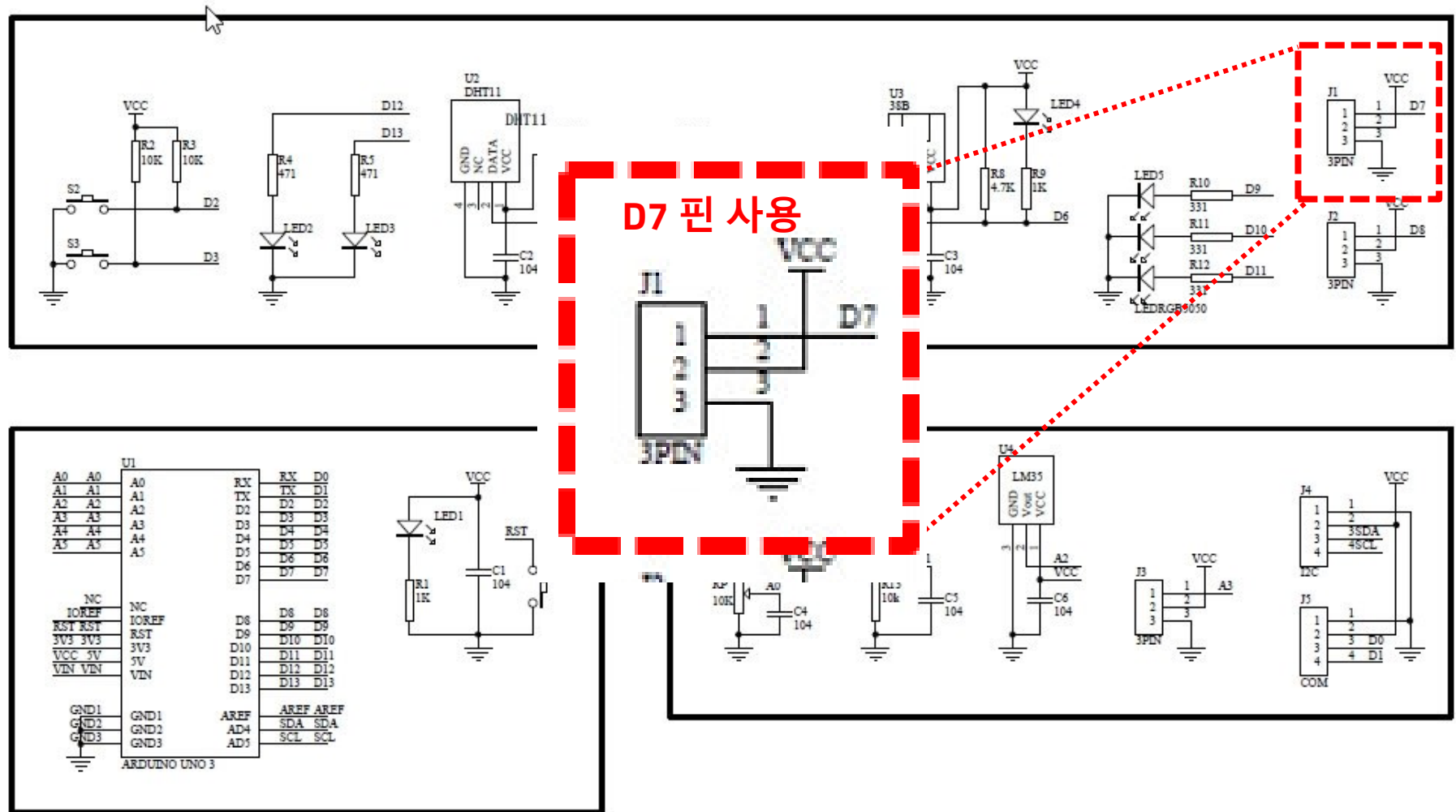


Figure 25-32 TOM Block diagram

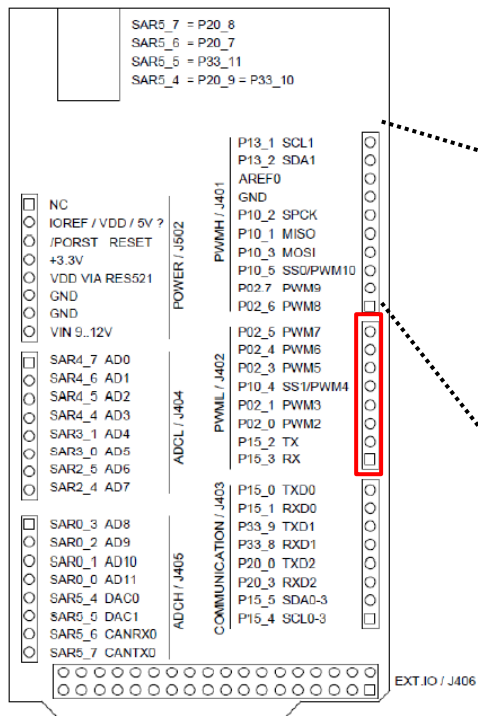
회로도 @ 확장 보드 데이터 시트



Pin Map @ TC275 보드

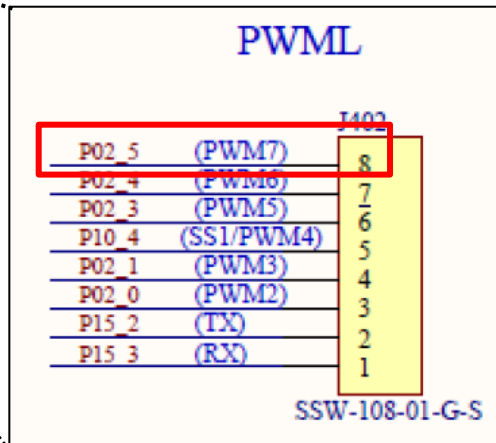
- 확장 보드의 핀 D7는 TC275 보드의 어느 핀에 연결?

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



D7

| | | |
|---------------------|--------|------|
| Digital pin 6 (PWM) | PWML.7 | P2.4 |
| Digital pin 7 (PWM) | PWML.8 | P2.5 |
| Digital pin 8 (PWM) | PWMH.1 | P2.6 |



TC275 MCU I/O 중,
Port 2 pin 5 (P02.5)
와 연결됨

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

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

GTM 사용을 위한 레지스터 설정

:GPIO 출력을 GTM 출력으로 사용하는 설정

- PWM 신호를 출력할 핀은 **P02.5**

Table 13-14 Port 02 Functions (cont'd)

| Port Pin | I/O | Pin Functionality | Associated Reg./ I/O Line | Port I/O Control Select. | |
|----------|-----|------------------------|---------------------------|--------------------------|--------------------|
| | | | | Reg./Bit Field | Value |
| P02.5 | I | General-purpose input | P02_IN.P5 | P02_IOC4. PC5 | 0XXXX _B |
| | | GTM input | TIN5 | | |
| | | QSPI3 input | MRST3A | | |
| | | PSI5 input | PSIRX1B | | |
| | | SENT input | SENT3C | | |
| | | DSADC input | DSCIN4B | | |
| | | I2C0 input | SCL0A | | |
| | | CIF input | CIFD5 | | |
| | | PSI5-S input | PSISRxB | | |
| | O | General-purpose output | P02_OUT.P5 | | 1X000 _B |
| | | GTM output | TOUT5 | | 1X001 _B |
| | | CAN node 0 output | TXDCAN0 | | 1X010 _B |
| | | QSPI3 output | MRST3 | | 1X011 _B |
| | | DSADC output | DSCOUT4 | | 1X100 _B |
| | | I2C0 output | SCL0 | | 1X101 _B |
| | | ERAY output | TXENB | | 1X110 _B |
| | | CCU60 output | COUT62 | | 1X111 _B |

**P02.5 핀의 GPIO 출력을 GTM 출력으로 사용하기 위해 →
포트의 I/O Control 레지스터 설정 필요**

- GPIO P02 레지스터 항목에서
– **IOC4** 레지스터 설정 필요

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.1154

GPIO 레지스터 설정 – P02 Address 계산

: Port 02의 각 레지스터가 위치한 주소 확인

3. 사용할 특정 레지스터의 주소 찾기

- **P02_IOCR4** Offset Address = 0x0014
- P02_IOCR4 레지스터 주소
- = **P02 Base address + IOCR4 Offset**
- = 0xF003A200 + 0x0014 = **0xF003A214**

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_OMR | 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_IOCR0 | Port n Input/Output Control Register 0 | 0010 _H | U, SV | U, SV, P | Application Reset | Page 13-14 |
| Pn_IOCR4 | Port n Input/Output Control Register 4 | 0014 _H | U, SV | U, SV, P | Application Reset | Page 13-14 |
| Pn_IOCR8 | Port n Input/Output Control Register 8 | 0018 _H | U, SV | U, SV, P | Application Reset | Page 13-14 |



[Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.1074](#)

Table 13-3 Registers Address Space

| Module | Base Address | End Address | Note |
|--------|------------------------|------------------------|---------|
| P00 | F003 A000 _H | F003 A0FF _H | 13 pins |
| P01 | F003 A100 _H | F003 A1FF _H | 5 pins |
| P02 | F003 A200 _H | F003 A2FF _H | 12 pins |
| P10 | F003 B000 _H | F003 B0FF _H | 9 pins |
| P11 | F003 B100 _H | F003 B1FF _H | 16 pins |

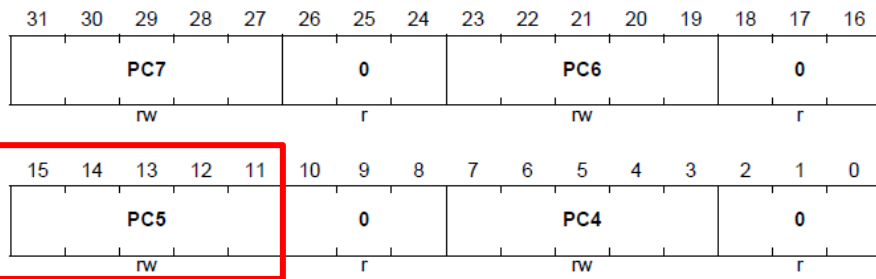
[Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.1073](#)

GPIO 레지스터 설정 – P02 I/O 모드

: Port 02의 핀 5를 GTM 출력 모드로 사용

- P02.5를 General Purpose Output (push-pull)가 아닌 GTM 모듈의 출력으로 사용하기 위해 P10_IOCRO 레지스터에 어떤 값을 써야 하는지 확인 → **0x11 write**

P02_IOCRO4 레지스터 @ 0xF003A214



| 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 13-14 Port 02 Functions (cont'd)

| Port Pin | I/O | Pin Functionality | Associated Reg./ I/O Line | Port I/O Control Select. | |
|----------|-----|------------------------|---------------------------|--------------------------|--------------------|
| | | | | Reg./Bit Field | Value |
| P02.5 | I | General-purpose input | P02_IN.P5 | P02_IOCRO4. PC5 | 0XXXX _B |
| | | GTM input | TIN5 | | |
| | | QSPI3 input | MRST3A | | |
| | | PSI5 input | PSIRX1B | | |
| | | SENT input | SENT3C | | |
| | | DSADC input | DSCIN4B | | |
| | | I2C0 input | SCL0A | | |
| | | CIF input | CIFD5 | | |
| | | PSI5-S input | PSISRXB | | |
| | | General-purpose output | P02_OUT.P5 | | 1X000 _B |
| O | | GTM output | TOUT5 | | 1X001 _B |
| | | CAN node 0 output | TXDCAN0 | | 1X010 _B |
| | | QSPI3 output | MRST3 | | 1X011 _B |
| | | DSADC output | DSCOUT4 | | 1X100 _B |
| | | I2C0 output | SCL0 | | 1X101 _B |
| | | ERAY output | TXENB | | 1X110 _B |
| | | CCU60 output | COUT62 | | 1X111 _B |

what?

- GTM 출력 모드로 사용하기 위해 0x11 (10001b) 값을 PC1 영역에 write

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.1082

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.1154

Lab1: P10 레지스터 설정 – IOCR0

:GPIO 출력 모드를 GTM 생성 출력으로 설정

initPWM() 작성

```
417 void initPWM(void)
418 {
419     P02_IOCR4.B.PC5 = 0x11;
420 }
421
```

GTM 사용을 위한 레지스터 설정

:GTM 모듈의 PWM 출력을 MCU 외부 핀으로 연결

- PWM 신호를 출력할 핀은 **P02.5**
- 핀 P02.5에 **GPIO** 대신 **GTM** 출력으로 연결 필요 → **TOUT5**

Table 13-14 Port 02 Functions (cont'd)

| Port Pin | I/O | Pin Functionality | Associated Reg./ I/O Line | Port I/O Control Select. | |
|----------|-----|------------------------|---------------------------|--------------------------|--------------------|
| | | | | Reg./Bit Field | Value |
| P02.5 | I | General-purpose input | P02_IN.P5 | P02_IOCRA4. PC5 | 0XXXX _B |
| | | GTM input | TIN5 | | |
| | | QSPI3 input | MRST3A | | |
| | | PSI5 input | PSIRX1B | | |
| | | SENT input | SENT3C | | |
| | | DSADC input | DSCIN4B | | |
| | | I2C0 input | SCL0A | | |
| | | CIF input | CIFD5 | | |
| | | PSI5-S input | PSISRXB | | |
| | O | General-purpose output | P02_OUT.P5 | TOUTSEL | 1X000 _B |
| | | GTM output | TOUT5 | | 1X001 _B |
| | | CAN node 0 output | TXDCAN0 | | 1X010 _B |
| | | QSPI3 output | MRST3 | | 1X011 _B |
| | | DSADC output | DSCOUT4 | | 1X100 _B |
| | | I2C0 output | SCL0 | | 1X101 _B |
| | | ERAY output | TXENB | | 1X110 _B |
| | | CCU60 output | COUT62 | | 1X111 _B |

**GTM 출력으로 사용하기 위해
TOUT5 관련 레지스터 설정 필요**

- GTM 레지스터 항목에서
– **TOUTSEL** 레지스터 설정 필요

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.1154

GTM 사용을 위한 레지스터 설정

:사용하는 GPIO 핀이 TOM의 어느 출력과 연결되는지 확인

- 앞서 핀 P02.5 에 연결되는 GTM의 출력은 TOUT5 인 것을 확인
- TOUT 출력은 GTM – TOM 에서 사용하는 TOM 번호와 채널이 정해져 있음

→ P02.5는 TOUT5

[Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.3484](#)

Table 25-67 GTM to Port Mapping for QFP-176

| Port | Input | Output | Input Timer Mapped | | Output Timer Mapped | | | |
|-------|-------|--------|--------------------|--------|---------------------|---------|----------|----------|
| | | | A | B | A | B | C | D |
| P02.2 | TIN2 | TOUT2 | TIM0_2 | TIM1_2 | TOM0_10 | TOM1_10 | ATOM 0_2 | ATOM 1_2 |
| P02.3 | TIN3 | TOUT3 | TIM0_3 | TIM1_3 | TOM0_11 | TOM1_11 | ATOM 0_3 | ATOM 1_3 |
| P02.4 | TIN4 | TOUT4 | TIM0_4 | TIM1_4 | TOM0_12 | TOM1_12 | ATOM 0_4 | ATOM 1_4 |
| P02.5 | TIN5 | TOUT5 | TIM0_5 | TIM1_5 | TOM0_13 | TOM1_13 | ATOM 0_5 | ATOM 1_5 |
| P02.6 | TIN6 | TOUT6 | TIM0_6 | TIM1_6 | TOM0_14 | TOM1_14 | ATOM 0_6 | ATOM 1_6 |

GTM 레지스터 설정 – TOUTSELO

1. GTM 레지스터 영역의 주소 찾기

- 시작 주소 (Base address) = **0xF0100000**

2. 사용할 레지스터의 주소 찾기

- 1개의 **TOUTSELx** 레지스터는 16개의 **TOUT** 핀을 제어함
- TOUT0 부터 16개씩 묶을 때 핀 **TOUT5** (P02.5)는 **TOUTSELO**에 포함되어 있음(**TOUT0~TOUT15** 안에 **TOUT5**가 포함)
- **TOUTSELO** 의 Offset Address = **0x9FD30**
- TOUTSELO 레지스터 주소 = $0xF0100000 + 0x9FD30 = \mathbf{0xF019FD30}$

Table 25-64 Registers Overview - GTM Control Registers

| Short Name | Description | Offset Addr. | Access Mode | | Reset | Description See |
|------------|--------------------------------|--------------------|-------------|----------|-------------|-----------------|
| | | | Read | Write | | |
| CLC | Clock Control Register | 9FD00 _H | U, SV | SV, E, P | Application | Page 25-74 9 |
| TIM0INSEL | TIM0 Input Select Register | 9FD10 _H | U, SV | U, SV, P | Application | Page 25-77 3 |
| TIM1INSEL | TIM1 Input Select Register | 9FD14 _H | U, SV | U, SV, P | Application | Page 25-77 3 |
| TIM2INSEL | TIM2 Input Select Register | 9FD18 _H | U, SV | U, SV, P | Application | Page 25-77 3 |
| TIM30INSEL | TIM3 Input Select Register | 9FD1C _H | U, SV | U, SV, P | Application | Page 25-77 3 |
| TOUTSEL0 | Timer Output Select 0 Register | 9FD30 _H | U, SV | U, SV, P | Application | Page 25-81 4 |
| TOUTSEL1 | Timer Output Select 1 Register | 9FD34 _H | U, SV | U, SV, P | Application | Page 25-81 4 |

[Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.3469](#)

GTM 사용을 위한 레지스터 설정

: P02.5는 TOUT5에 연결되어 있으며, TOM0 모듈의 채널 13을 사용하도록 설정됨

→ P02.5 (TOUT5)은 **TOM0**의 채널 13 사용

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.3484

Table 25-67 GTM to Port Mapping for QFP-176

| Port | Input | Output | Input Timer Mapped | | Output Timer Mapped | | | |
|-------|-------|--------|--------------------|--------|---------------------|---------|----------|----------|
| | | | A | B | A | B | C | D |
| P02.2 | TIN2 | TOUT2 | TIM0_2 | TIM1_2 | TOM0_10 | TOM1_10 | ATOM 0_2 | ATOM 1_2 |
| P02.3 | TIN3 | TOUT3 | TIM0_3 | TIM1_3 | TOM0_11 | TOM1_11 | ATOM 0_3 | ATOM 1_3 |
| P02.4 | TIN4 | TOUT4 | TIM0_4 | TIM1_4 | TOM0_12 | TOM1_12 | ATOM 0_4 | ATOM 1_4 |
| P02.5 | TIN5 | TOUT5 | TIM0_5 | TIM1_5 | TOM0_13 | TOM1_13 | ATOM 0_5 | ATOM 1_5 |
| P02.6 | TIN6 | TOUT6 | TIM0_6 | TIM1_6 | TOM0_14 | TOM1_14 | ATOM 0_6 | ATOM 1_6 |

Output Timer Mapped의 **A** 항목 사용하도록 설정 시
→ 0번째 TOM (TOM0)의 채널 13 사용

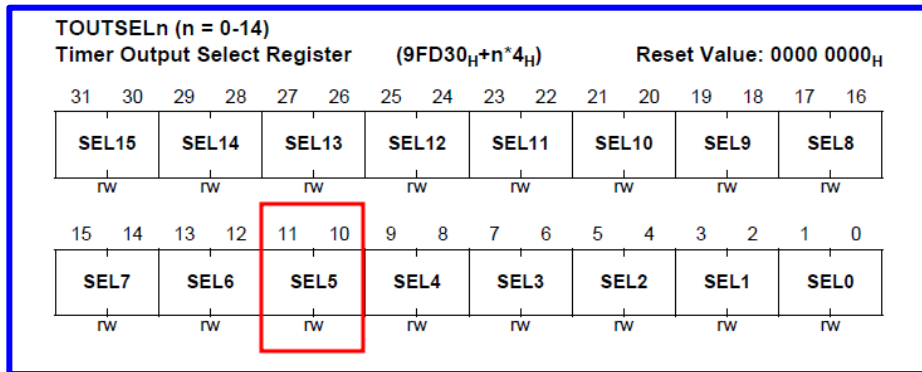
GTM 레지스터 설정 – TOUTSELO

:TOM에서 출력되는 PWM 신호를 MCU 핀에 연결 설정

3. 레지스터 write 값 결정

- TOM0 채널13으로부터 생성된 PWM 신호를 TOUT5 핀으로 출력하기 위한 설정
- TOUTSELO는 TOUT0부터 TOUT15까지 제어함.
- TOUT5 핀은 5번째 (0번째부터)
 - Output Timer Mapped 에서 A 항목을 사용하도록 설정하기 위해 SEL5 영역에 0x0 write

TOUTSELO 레지스터 @ 0xF019FD30



| Field | Bits | Type | Description |
|--------------------|-----------------|------|---|
| SELx (x = 0-15) | [x*2+1: x*2] | rw | TOUT(n*16+x) Output Selection This bit defines which timer out is connected as TOUT(n*16+x). The mapping for each pin is defined by Table 25-67 Table 25-68 . <div>00_B Timer A form Table 25-67Table 25-68 is connected as TOUT(n*16+x) to the ports</div> <div>01_B Timer B form Table 25-67Table 25-68 is connected as TOUT(n*16+x) to the ports</div> <div>10_B Timer C form Table 25-67Table 25-68 is connected as TOUT(n*16+x) to the ports</div> <div>11_B Timer D form Table 25-67Table 25-68 is connected as TOUT(n*16+x) to the ports</div> <i>Note: If TOUT(n*16+x) is not defined in Table 25-67Table 25-68 this bit field has to be treated as reserved.</i> |

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.3541

Lab2: GTM 레지스터 설정 – TOUTSELO

:TOM에서 출력되는 (TOM0모듈의 채널13) PWM 신호를 MCU 핀에 연결 설정

```
70 void initGTM(void)
71 {
72     // Password Access to unlock SCU_WDTSCON0
73     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
74     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0); // wait until unlocked
75
76     // Modify Access to clear ENDINIT
77     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) & ~(1 << ENDINIT_BIT_LSB_IDX);
78     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0); // wait until locked
79
80     GTM_CLC.U &= ~(1 << DISR_BIT_LSB_IDX); // enable GTM
81
82     // Password Access to unlock SCU_WDTSCON0
83     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
84     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0); // wait until unlocked
85
86     // Modify Access to set ENDINIT
87     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
88     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0);
89
90     while((GTM_CLC.U & (1 << DISS_BIT_LSB_IDX)) != 0); // wait until GTM module enabled
91
92
93     // GTM clock configuration
94     GTM_CMU_FXCLK_CTRL.U &= ~(0xF << FXCLK_SEL_BIT_LSB_IDX); // input clock of CMU_FXCLK --> CMU_GCLK_EN
95     GTM_CMU_CLK_EN.U |= 0x2 << EN_FXCLK_BIT_LSB_IDX; // enable all CMU_FXCLK
96
97     // GTM TOM0 PWM configuration
98     GTM_TOM0_TGC0_GLB_CTRL.U |= 0x2 << UPEN_CTRL1_BIT_LSB_IDX; // TOM channel 1 update enable
99
100    GTM_TOM0_TGC0_ENDIS_CTRL.U |= 0x2 << ENDIS_CTRL1_BIT_LSB_IDX; // enable channel 1 on update trigger
101
102    GTM_TOM0_TGC0_OUTEN_CTRL.U |= 0x2 << OUTEN_CTRL1_BIT_LSB_IDX; // enable channel 1 output on update trigger
103
104    GTM_TOM0_CH1_CTRL.U |= 0x1 << SL_BIT_LSB_IDX; // high signal level for duty cycle
105    GTM_TOM0_CH1_CTRL.U |= 0x1 << CLK_SRC_SR_BIT_LSB_IDX; // clock source --> CMU_FXCLK(1) = 6250 kHz
106    GTM_TOM0_CH1_CTRL.U &= ~(0x1 << OSM_BIT_LSB_IDX); // continuous mode enable
107    GTM_TOM0_CH1_CTRL.U &= ~(0x1 << TRIGOUT_BIT_LSB_IDX); // TRIG[x] = TRIG[x-1]
108
109    GTM_TOM0_CH1_SR0.U = 12500 - 1; // PWM freq. = 6250 kHz / 12500 = 500 Hz
110
111    GTM_TOM0_CH1_SR1.U = 1250 - 1; // duty cycle = 1250 / 12500 = 10 % (temporary)
112
113    GTM_TOUTSEL0.U &= ~(0x3 << SEL7_BIT_LSB_IDX); // 100/100 --> TOM0 channel 1
114    // 103 = 16 * 6 + 7
115
116
```

initGTM() 함수 일부 수정

GTM_TOUTSEL0.B.SEL5 = 0x0;

// TOUT5 --> TOM0 channel 13
// 5 = 16 * 0 + 5

GTM 사용을 위한 레지스터 설정

:TOM에서 출력으로 사용할 채널 설정

Infineon-TC27x_D-step-UM-v02_02-EN.pdf
p.2918

- TOM 설정은 TGC (TOM Global Control Unit) 가 담당

25.11.2 TOM Global Channel Control (TGC0, TGC1)

25.11.2.1 Overview

There exist two global channel control units (TGC0 and TGC1) to drive a number of individual TOM channels synchronously by external or internal events.

Each TGC[y] can drive up to eight TOM channels where TGC0 controls TOM channels 0 to 7 and TGC1 controls TOM channels 8 to 15.

The TOM submodule supports four different kinds of signalling mechanisms:

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.2919

- 본 실습에서는 **TOM0**의 채널 **13**을 사용하기 위해 **TGC1**에 해당하는 레지스터 설정 필요
- GTM 레지스터 항목에서
 - TOM0_TGC1_GLB_CTRL** 레지스터 설정 필요
 - TOM0_TGC1_ENDIS_CTRL** 레지스터 설정 필요
 - TOM0_TGC1_OUTEN_CTRL** 레지스터 설정 필요

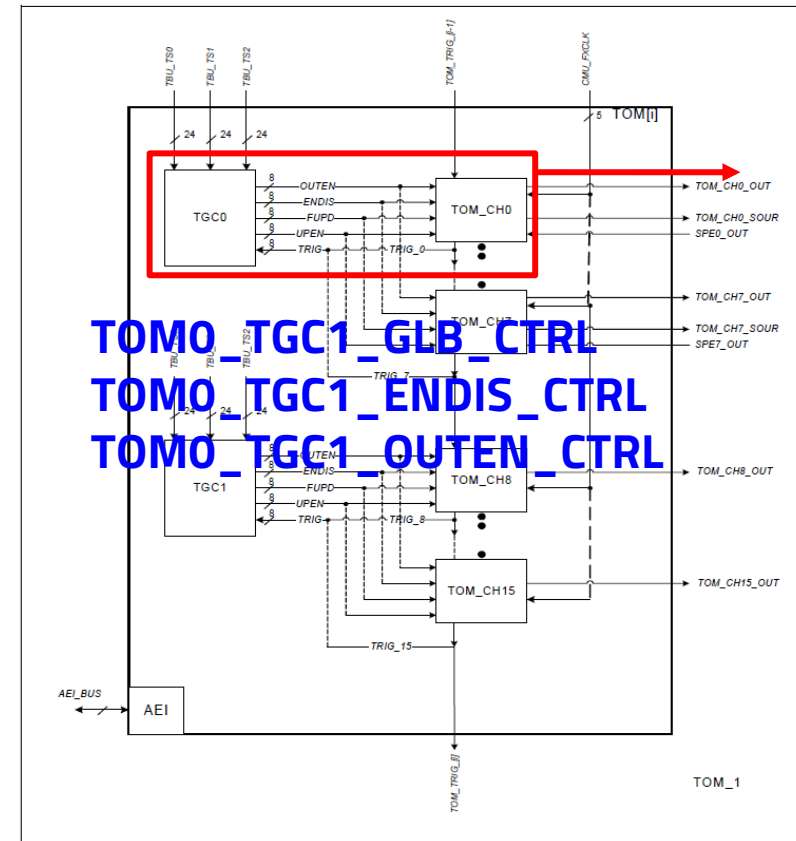


Figure 25-32 TOM Block diagram

GTM 레지스터 설정 – TOM0_TGC1_GLB_CTRL

1. GTM 레지스터 영역의 주소 찾기

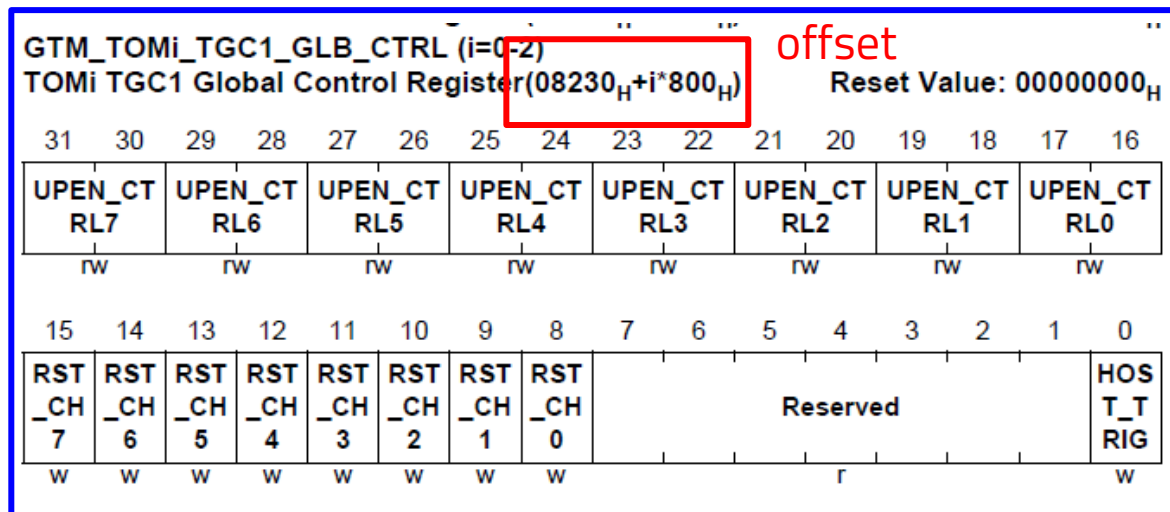
– 시작 주소 (Base address) = **0xF0100000**

2. 사용할 레지스터의 주소 찾기 (TOM0 이므로 i = 0)

– **TOM0_TGC1_GLB_CTRL** 의 Offset Address = $0x8230 + (i * 0x800) = \mathbf{0x8230}$

→ TOM0_TGC1_GLB_CTRL 레지스터 주소 = $0xF0100000 + 0x8230 = \mathbf{0xF0108230}$

TOM0_TGC0_GLB_CTRL 레지스터 @ 0xF0108230



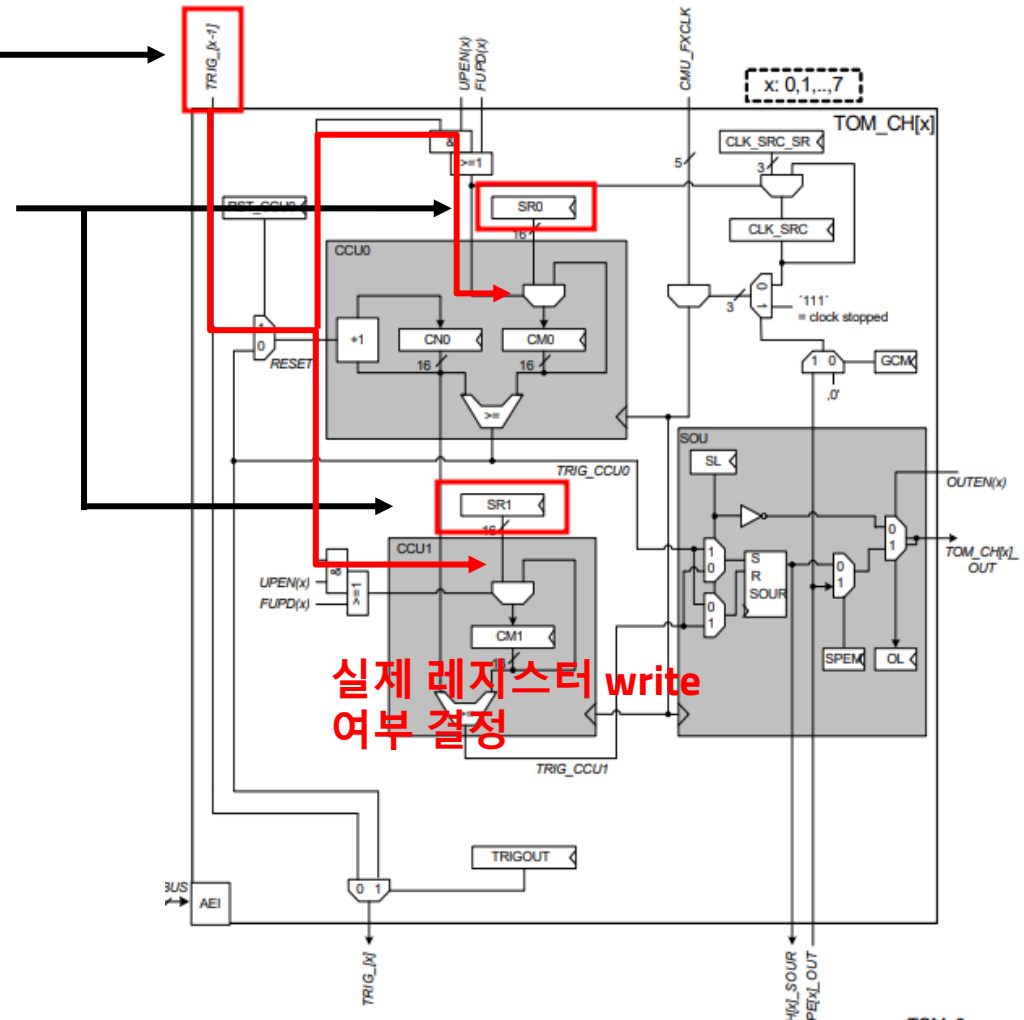
GTM 설정 레지스터의 write 구조

:shadow 레지스터 write되어 update 신호 발생해야 실제 write

Update Trigger 신호

PWM 동작 설정에 필요한
shadow 레지스터

- SW 에서 레지스터 write를 하면
실제 레지스터에 **write** 되기 전,
shadow 레지스터에 저장
- Update Trigger 신호 발생하면 실제
레지스터로 옮겨 write 해야 함
- TOM0 의 채널 13에서 이러한
update가 가능하도록 설정 필요



GTM 레지스터 설정 – TOM0_TGC1_GLB_CTRL

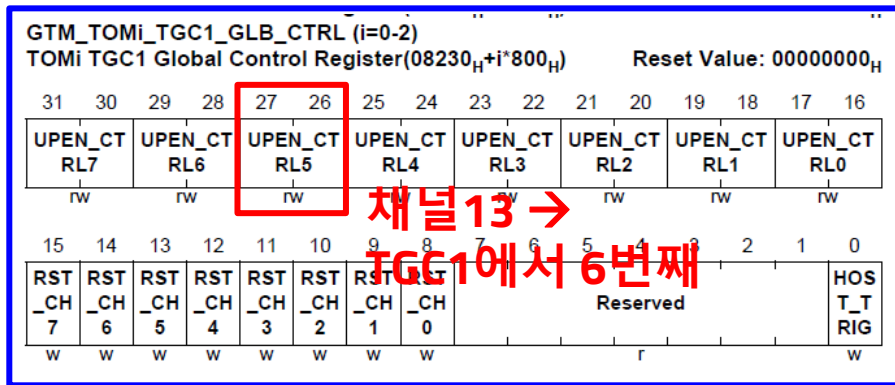
:TOM 에서 사용할 채널 설정

3. 레지스터 write 값 결정

- TOM0의 채널 13이 동작하기 위해서는 해당 채널에서 PWM 설정 값들이 update 되어야 함
- Update가 가능하도록 설정하기 위해 **UPEN_CTRL1** 영역에 **0x2 write**

Infineon-TC27x_D-
step-UM-v02_02-
EN.pdf p.2923

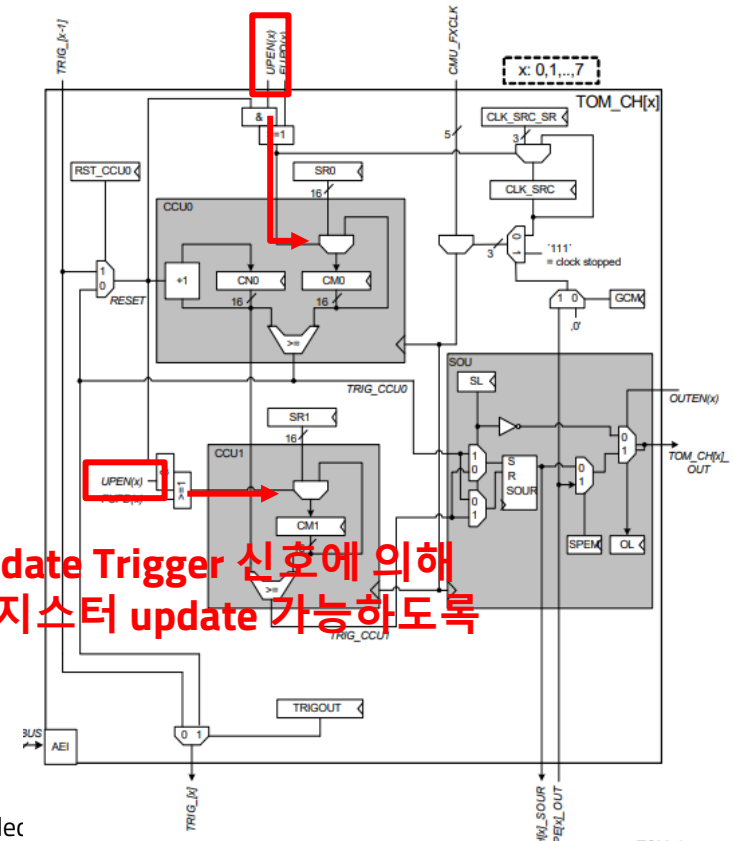
TOM0_TGC1_GLB_CTRL 레지스터 @ 0xF0108230



Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.2937

| Field | Bits | Type | Description |
|-------------|---------|------|--|
| UPEN_CT RL5 | [27:26] | rw | TOM channel 5 enable update of register CM0, CM1 and CLK_SRC See bits 17:16 |

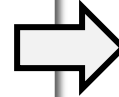
00_B don't care, bits 1:0 will not be changed
01_B update disabled: is read as 00 (see below)
10_B update enabled: is read as 11 (see below)
11_B don't care, bits 1:0 will not be changed



Lab3: GTM 레지스터 설정 – TOM0_TGC1_GLB_CTRL

:TOM 에서 사용할 채널13의 update enable 설정

```
70 void initGTM(void)
71 {
72     // Password Access to unlock SCU_WDTSCON0
73     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
74     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0); // wait until unlocked
75
76     // Modify Access to clear ENDINIT
77     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) & ~(1 << ENDINIT_BIT_LSB_IDX);
78     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0); // wait until locked
79
80     GTM_CLC.U &= ~(1 << DISR_BIT_LSB_IDX); // enable GTM
81
82     // Password Access to unlock SCU_WDTSCON0
83     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
84     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0); // wait until unlocked
85
86     // Modify Access to set ENDINIT
87     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
88     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0);
89
90     while((GTM_CLC.U & (1 << DISS_BIT_LSB_IDX)) != 0); // wait until GTM module enabled
91
92
93     // GTM clock configuration
94     GTM_CMU_FXCLK_CTRL.U &= ~(0xF << FXCLK_SEL_BIT_LSB_IDX); // input clock of CMU_FXCLK --> CMU_GCLK_EN
95     GTM_CMU_CLK_EN.U |= 0x2 << EN_FXCLK_BIT_LSB_IDX; // enable all CMU_FXCLK
96
97     // GTM TOM0 PWM configuration
98     GTM_TOM0_TGC0_GLB_CTRL.U |= 0x2 << UPEN_CTRL1_BIT_LSB_IDX; // TOM channel 1 update enable
99
100     GTM_TOM0_TGC0_ENDIS_CTRL.U |= 0x2 << ENDIS_CTRL1_BIT_LSB_IDX; // enable channel 1 on update trigger
101
102     GTM_TOM0_TGC0_OUTEN_CTRL.U |= 0x2 << OUTEN_CTRL1_BIT_LSB_IDX; // enable channel 1 output on update trigger
103
104     GTM_TOM0_CH1_CTRL.U |= 0x1 << SL_BIT_LSB_IDX; // high signal level for duty cycle
105     GTM_TOM0_CH1_CTRL.U |= 0x1 << CLK_SRC_SR_BIT_LSB_IDX; // clock source --> CMU_FXCLK(1) = 6250 kHz
106     GTM_TOM0_CH1_CTRL.U &= ~(0x1 << OSM_BIT_LSB_IDX); // continuous mode enable
107     GTM_TOM0_CH1_CTRL.U &= ~(0x1 << TRIGOUT_BIT_LSB_IDX); // TRIG[x] = TRIG[x-1]
108
109     GTM_TOM0_CH1_SR0.U = 12500 - 1; // PWM freq. = 6250 kHz / 12500 = 500 Hz
110
111     GTM_TOM0_CH1_SR1.U = 1250 - 1; // duty cycle = 1250 / 12500 = 10 % (temporary)
112
113     GTM_TOUTSEL6.U &= ~(0x3 << SEL7_BIT_LSB_IDX); // TOUT103 --> TOM0 channel 1
114     // 103 = 16 * 6 + 7
115 }
116
```



GTM_TOM0_TGC1_GLB_CTRL.B.UPEN_CTRL5 = 0x2;

// TOM channel 13 update enable

GTM 레지스터 update 하는 Trigger 신호 생성

:1) SW에서 생성하는 Update Trigger 신호

- 앞으로 설정할 GTM 레지스터들이 실제 하드웨어에 적용될 수 있도록 하는 **Update Trigger** 신호를
- 1) SW에서 생성할 수 있음**
- 2) PWM의 1주기에 도달했을 때 생성할 수 있음 (뒤에서 설명)**
- Update Trigger 신호 발생 전까지는 레지스터에 write 해도 실제 하드웨어에 적용되지 않음 (shadow register 개념)

25.11.2.2 TGC Subunit

Each of the first three individual mechanisms (enable/disable of the channel, output enable and force update) can be driven by three different trigger sources.

The three trigger sources are:

- the host CPU (bit HOST_TRIG of register TOMi_TGCy_GLB_CTRL)
- the TBU time stamp (signal TBU_TS0., TBU_TS1, TBU_TS2)
- the internal trigger signal TRIG (bunch of trigger signals TRIG[x])

Note: The trigger signal is only active for one configured CMU clock period.

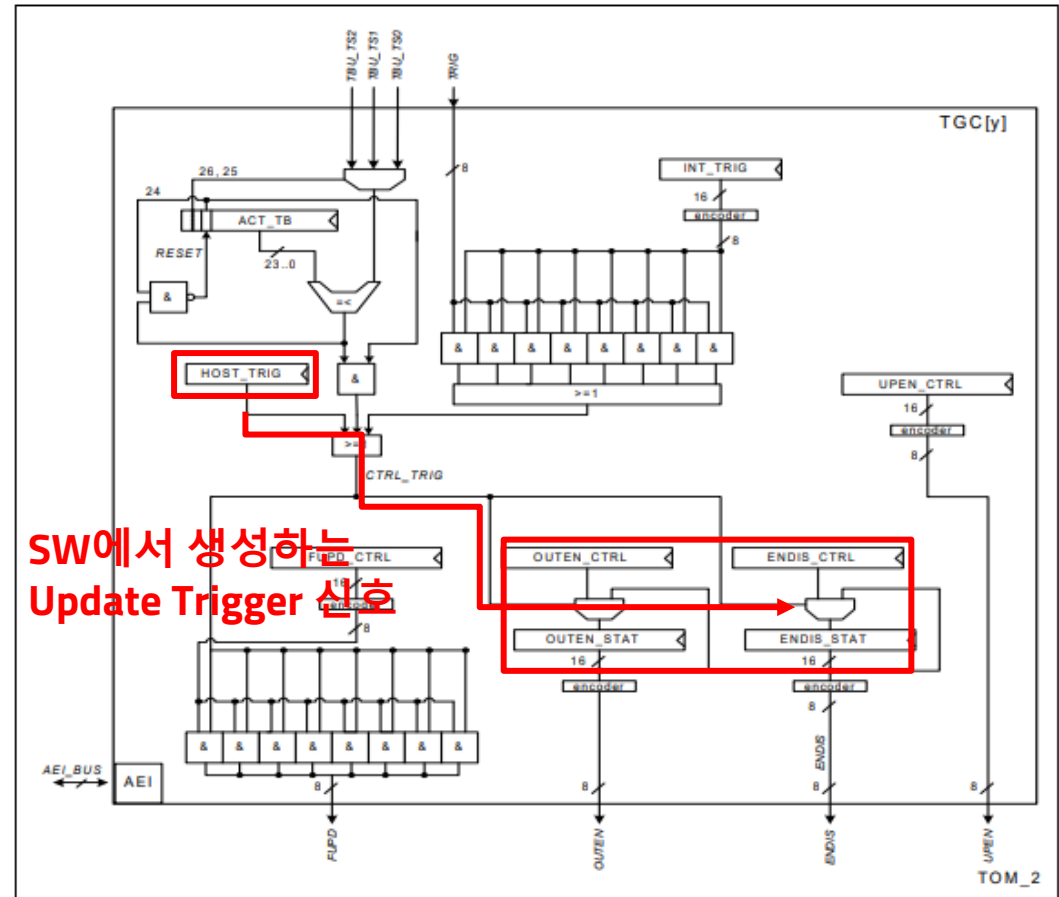


Figure 25-33 TOM Global channel control mechanism

GTM 레지스터 설정 – TOM0_TGC1_GLB_CTRL

:TOM 레지스터 설정이 적용되도록 update trigger event 발생

3. 레지스터 write 값 결정

- TOM0의 채널 13에 대한 레지스터 설정이 shadow 레지스터로부터 하드웨어에 적용되도록 하기 위해 **HOST_TRIG** 영역에 **0x1 write** (GTM 레지스터 설정 후 마지막에 수행)

TOM0_TGC1_GLB_CTRL 레지스터@ 0xF0108230

| GTM_TOMi_TGC0_GLB_CTRL (i=0-2) | | | | | | | | | | | | | | | | |
|--|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|----------------|----|----------------|----|----------------|----|----------------|-------------------|--|
| TOMi TGC0 Global Control Register(08030 _H +i*800 _H) | | | | | | | | | | | | | | | | |
| Reset Value: 00000000 _H | | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | |
| UPEN_CT RL7 | | UPEN_CT RL6 | | UPEN_CT RL5 | | UPEN_CT RL4 | | UPEN_CT RL3 | | UPEN_CT RL2 | | UPEN_CT RL1 | | UPEN_CT RL0 | | |
| rw | | rw | | rw | | rw | | rw | | rw | | rw | | rw | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
| RST _CH 7 | RST _CH 6 | RST _CH 5 | RST _CH 4 | RST _CH 3 | RST _CH 2 | RST _CH 1 | RST _CH 0 | Reserved | | | | | | | HOS T_T RIG | |
| w | w | w | w | w | w | w | w | r | | | | | | | w | |

| Field | Bits | Type | Description |
|-----------|------|------|---|
| HOST_TRIG | 0 | w | Trigger request signal (see TGC0, TGC1) to update the register ENDIS_STAT and OUTEN_STAT 0 _B no trigger request 1 _B set trigger request Read as 0. Note: This flag is cleared automatically after triggering the update |

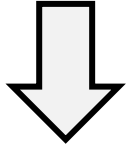
Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.2937

Lab4: GTM 레지스터 설정 – TOM0_TGC1_GLB_CTRL

:TOM 에서 출력 생성 시작하도록 trigger event 발생

```
147 //initERU();
148 initCCU60();
149 initLED();
150 initRGBLED();
151 initVADC();
152 initGTM();
153 //initButton();
154
155 GTM_TOM0_TGC0_GLB_CTRL.U |= 0x1 << HOST_TRIG_BIT_LSB_IDX; // trigger update request signal
156
157 while(1)
158 {
159     VADC_startConversion();
160     unsigned int adcResult = VADC_readResult();
161     for(unsigned int i = 0; i < 100; i++);
162
163     GTM_TOM0_TGC1_GLB_CTRL.B.HOST_TRIG = 0x1; // trigger update request signal
```

PWM 신호가 생성되기 원하는 시점에
trigger 발생시켜야 함



GTM 레지스터 설정 – TOM0_TGC1_ENDIS_CTRL

1. GTM 레지스터 영역의 주소 찾기

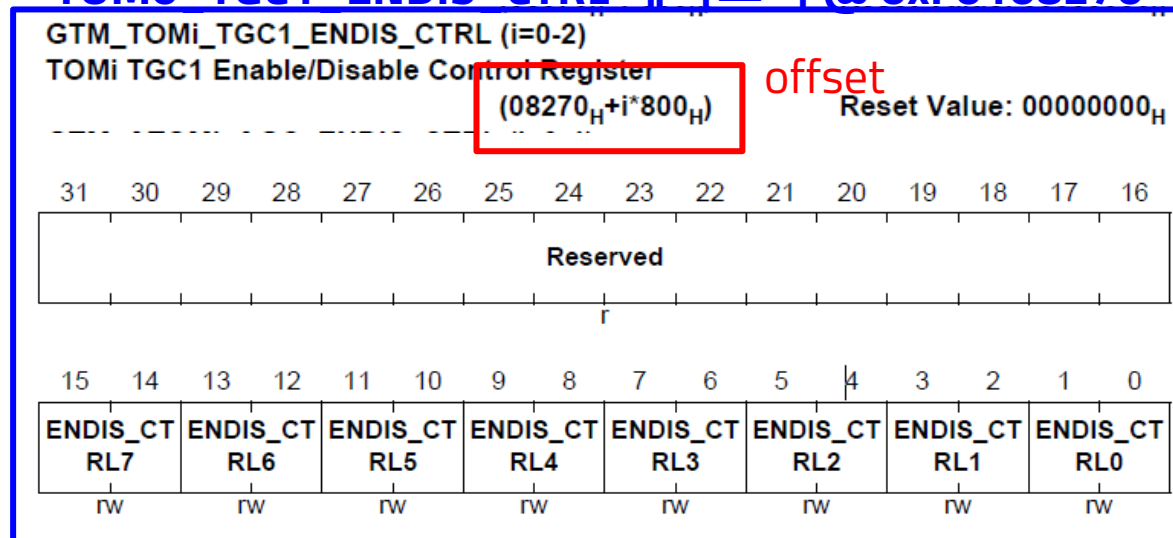
– 시작 주소 (Base address) = **0xF0100000**

2. 사용할 레지스터의 주소 찾기 (TOM0 이므로 i = 0)

– **TOM0_TGC1_ENDIS_CTRL** 의 Offset Address = $0x8270 + (i * 0x800) = \mathbf{0x8270}$

→ TOM0_TGC1_ENDIS_CTRL 레지스터 주소 = $0xF0100000 + 0x8270 = \mathbf{0xF0108270}$

TOM0 TGC1 ENDIS CTRL 레지스터 @ 0xF0108270



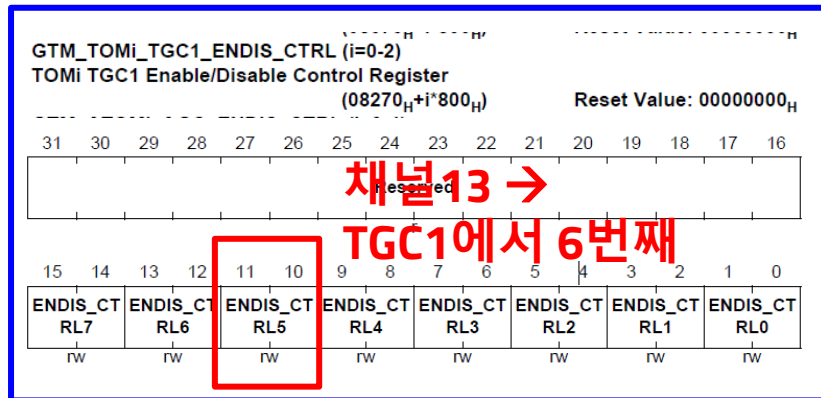
GTM 레지스터 설정 – TOM0_TGC1_ENDIS_CTRL

:TOM에서 사용할 채널에 대한 설정

3. 레지스터 write 값 결정

- TOM0_TGC1_GLB_CTRL 레지스터에서 Update Trigger 신호 발생 시, 하드웨어에 적용되어 **TOM0 채널 130**이 **enable** 되도록 하기 위해 **ENDIS_CTRL5** 영역에 **0x2** write

TOM0_TGC1_ENDIS_CTRL 레지스터 @ 0xF0108270



| RLx | Field | Width | Access | Description |
|---------------|---------|-------|--------|--|
| ENDIS_CTL RL5 | [11:10] | 2 | rw | (A)TOM channel 5 enable/disable update value See bits 1:0 |

00_B don't care, bits 1:0 of register ENDIS_ not be changed on an update trigger
01_B disable channel on an update trigger
10_B enable channel on an update trigger
11_B don't change bits 1:0 of this register

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.2940

Lab5: GTM 레지스터 설정 – TOM0_TGCO_ENDIS_CTRL

:TOM에서 사용할 채널에 대한 설정

```
70 void initGTM(void)
71 {
72     // Password Access to unlock SCU_WDTSCON0
73     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
74     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0); // wait until unlocked
75
76     // Modify Access to clear ENDINIT
77     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) & ~(1 << ENDINIT_BIT_LSB_IDX);
78     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0); // wait until locked
79
80     GTM_CLC.U &= ~(1 << DISR_BIT_LSB_IDX); // enable GTM
81
82     // Password Access to unlock SCU_WDTSCON0
83     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
84     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0); // wait until unlocked
85
86     // Modify Access to set ENDINIT
87     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
88     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0);
89
90     while((GTM_CLC.U & (1 << DISS_BIT_LSB_IDX)) != 0); // wait until GTM module enabled
91
92
93     // GTM clock configuration
94     GTM_CMU_FXCLK_CTRL.U &= ~(0xF << FXCLK_SEL_BIT_LSB_IDX); // input clock of CMU_FXCLK --> CMU_GCLK_EN
95     GTM_CMU_CLK_EN.U |= 0x2 << EN_FXCLK_BIT_LSB_IDX; // enable all CMU_FXCLK
96
97     // GTM TOM0 PWM configuration
98     GTM_TOM0_TGCO_GLB_CTRL.U |= 0x2 << UPEN_CTRL1_BIT_LSB_IDX; // TOM channel 1 update enable
99
100     GTM_TOM0_TGCO_ENDIS_CTRL.U |= 0x2 << ENDIS_CTRL1_BIT_LSB_IDX; // enable channel 1 on update trigger
101
102     GTM_TOM0_TGCO_OUTEN_CTRL.U |= 0x2 << OUTEN_CTRL1_BIT_LSB_IDX; // enable channel 1 output on update trigger
103
104     GTM_TOM0_CH1_CTRL.U |= 0x1 << SL_BIT_LSB_IDX; // high signal level for duty cycle
105     GTM_TOM0_CH1_CTRL.U |= 0x1 << CLK_SRC_SR_BIT_LSB_IDX; // clock source --> CMU_FXCLK(1) = 6250 kHz
106     GTM_TOM0_CH1_CTRL.U &= ~(0x1 << OSM_BIT_LSB_IDX); // continuous mode enable
107     GTM_TOM0_CH1_CTRL.U &= ~(0x1 << TRIGOUT_BIT_LSB_IDX); // TRIG[x] = TRIG[x-1]
108
109     GTM_TOM0_CH1_SR0.U = 12500 - 1; // PWM freq. = 6250 kHz / 12500 = 500 Hz
110
111     GTM_TOM0_CH1_SR1.U = 1250 - 1; // duty cycle = 1250 / 12500 = 10 % (temporary)
112
113     GTM_TOUTSEL6.U &= ~(0x3 << SEL7_BIT_LSB_IDX); // TOUT103 --> TOM0 channel 1
114     // 103 = 16 * 6 + 7
115 }
116
```

GTM_TOM0_TGC1_ENDIS_CTRL.B.ENDIS_CTRL5 = 0x2;

GTM 레지스터 설정 – TOM0_TGC1_OUTEN_CTRL

1. GTM 레지스터 영역의 주소 찾기

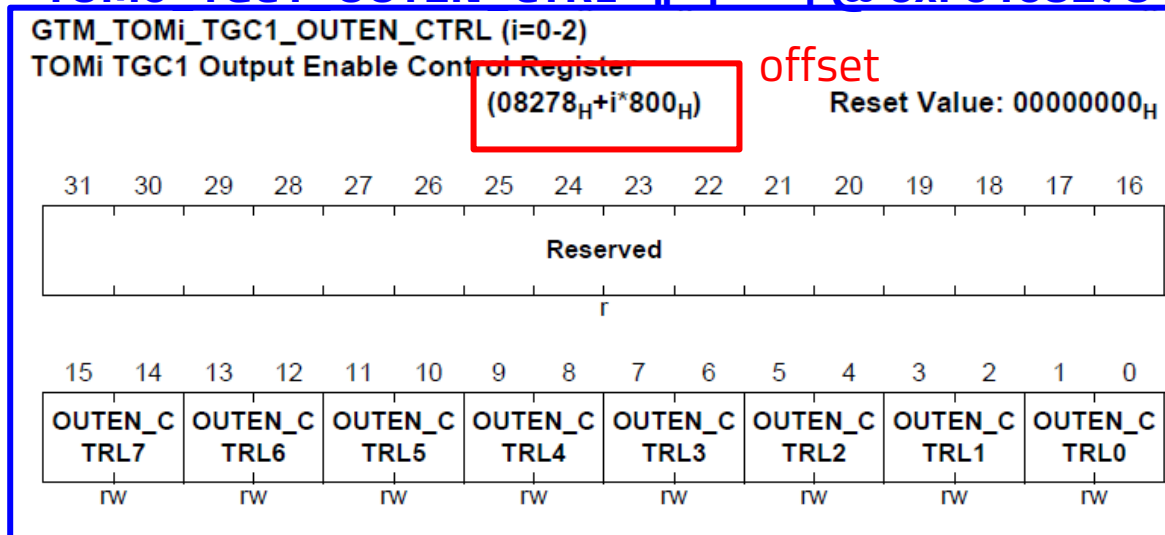
– 시작 주소 (Base address) = **0xF0100000**

2. 사용할 레지스터의 주소 찾기 (TOM0 이므로 i = 0)

– **TOM0_TGC1_OUTEN_CTRL** 의 Offset Address = $0x8278 + (i * 0x800) = \mathbf{0x8278}$

→ TOM0_TGC1_OUTEN_CTRL 레지스터 주소 = $0xF0100000 + 0x8278 = \mathbf{0xF0108278}$

TOM0 TGC1 OUTEN_CTRL 레지스터 @ 0xF0108278



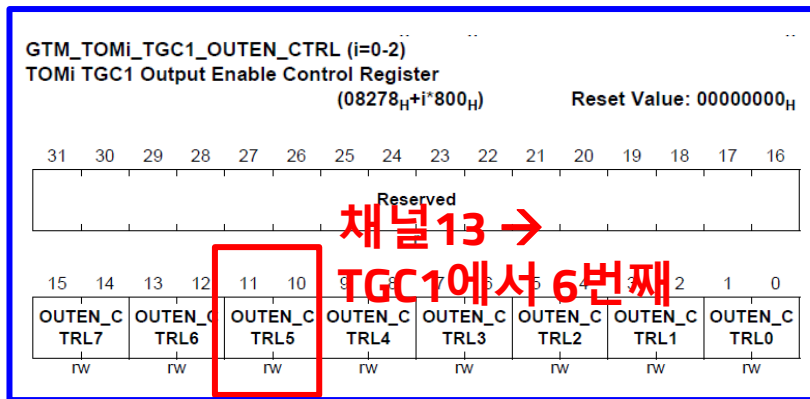
GTM 레지스터 설정 – TOM0_TGC1_OUTEN_CTRL

:TOM의 출력이 trigger 이벤트에 반응하도록 설정

3. 레지스터 write 값 결정

- TOM0_TGC1_GLB_CTRL 레지스터에서 Update Trigger 신호 발생 시, 하드웨어에 적용되어 **TOM0 채널 13**의 출력이 **enable** 되도록 하기 위해 **OUTEN_CTRL5** 영역에 **0x2** write

TOM0_TGC1_OUTEN_CTRL 레지스터 @ 0xF0108278



| | | | |
|--------------|---------|----|--|
| OUTEN_C TRL5 | [11:10] | rw | Output (A)TOM_OUT(5) enable/disable update value See bits 1:0 |
|--------------|---------|----|--|

| | |
|-----------------|--|
| 00 _B | don't care, bits 1:0 of register OUTEN_STAT will not be changed on an update trigger |
| 01 _B | disable channel output on an update trigger |
| 10 _B | enable channel output on an update trigger |
| 11 _B | don't change bits 1:0 of this register |

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.2945

Lab6: GTM 레지스터 설정 – TOM0_TGCO_OUTEN_CTRL

:TOM의 출력이 trigger 이벤트에 반응하도록 설정

```
70 void initGTM(void)
71 {
72     // Password Access to unlock SCU_WDTSCON0
73     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
74     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0); // wait until unlocked
75
76     // Modify Access to clear ENDINIT
77     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) & ~(1 << ENDINIT_BIT_LSB_IDX);
78     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0); // wait until locked
79
80     GTM_CLC.U &= ~(1 << DISR_BIT_LSB_IDX); // enable GTM
81
82     // Password Access to unlock SCU_WDTSCON0
83     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
84     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0); // wait until unlocked
85
86     // Modify Access to set ENDINIT
87     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
88     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0);
89
90     while((GTM_CLC.U & (1 << DISS_BIT_LSB_IDX)) != 0); // wait until GTM module enabled
91
92
93     // GTM clock configuration
94     GTM_CMU_FXCLK_CTRL.U &= ~(0xF << FXCLK_SEL_BIT_LSB_IDX); // input clock of CMU_FXCLK --> CMU_GCLK_EN
95     GTM_CMU_CLK_EN.U |= 0x2 << EN_FXCLK_BIT_LSB_IDX; // enable all CMU_FXCLK
96
97     // GTM TOM0 PWM configuration
98     GTM_TOM0_TGCO_GLB_CTRL.U |= 0x2 << UPEN_CTRL1_BIT_LSB_IDX; // TOM channel 1 update enable
99
100    GTM_TOM0_TGCO_ENDIS_CTRL.U |= 0x2 << ENDIS_CTRL1_BIT_LSB_IDX; // enable channel 1 on update trigger
101
102    GTM_TOM0_TGCO_OUTEN_CTRL.U |= 0x2 << OUTEN_CTRL1_BIT_LSB_IDX; // enable channel 1 output on update trigger
103
104    GTM_TOM0_CH1_CTRL.U |= 0x1 << SL_BIT_LSB_IDX; // high signal level for duty cycle
105    GTM_TOM0_CH1_CTRL.U |= 0x1 << CLK_SRC_SR_BIT_LSB_IDX; // clock source --> CMU_FXCLK(1) = 6250 kHz
106    GTM_TOM0_CH1_CTRL.U &= ~(0x1 << OSM_BIT_LSB_IDX); // continuous mode enable
107    GTM_TOM0_CH1_CTRL.U &= ~(0x1 << TRIGOUT_BIT_LSB_IDX); // TRIG[x] = TRIG[x-1]
108
109    GTM_TOM0_CH1_SR0.U = 12500 - 1; // PWM freq. = 6250 kHz / 12500 = 500 Hz
110
111    GTM_TOM0_CH1_SR1.U = 1250 - 1; // duty cycle = 1250 / 12500 = 10 % (temporary)
112
113    GTM_TOUTSEL6.U &= ~(0x3 << SEL7_BIT_LSB_IDX); // TOUT103 --> TOM0 channel 1
114    // 103 = 16 * 6 + 7
115 }
116
```

GTM_TOM0_TGCO_OUTEN_CTRL.B.OUTEN_CTRL5 = 0x2;

:PWM 생성과정의 counter와 duty cycle 관계

- PWM 신호 Duty Cycle (SL = 1 경우)**
- PWM 1주기에 도달하면 CNO 값 0으로 초기화**
- SL=0:
TOM_CH[x]_OUT
- SL=1:
TOM_CH[x]_OUT
- write a value to CNO
- enable channel
- period
- TOM_8
- Duty Cycle 동안 HIGH, LOW 출력을 결정**
- PWM 신호의 주기**
-
- The diagram shows the timing of the TOM_8 module. It includes signals for CM0, CM1, and CN0. The output signals for SL=0 and SL=1 are shown as green and purple waveforms. The period of the PWM signal is indicated by a red double-headed arrow. The duty cycle is also indicated by a red double-headed arrow. The text explains that when the PWM signal reaches 1 period, the CNO value is reset to 0.

Figure 25-39 PWM Output with respect to configuration bit SL in continuos mode

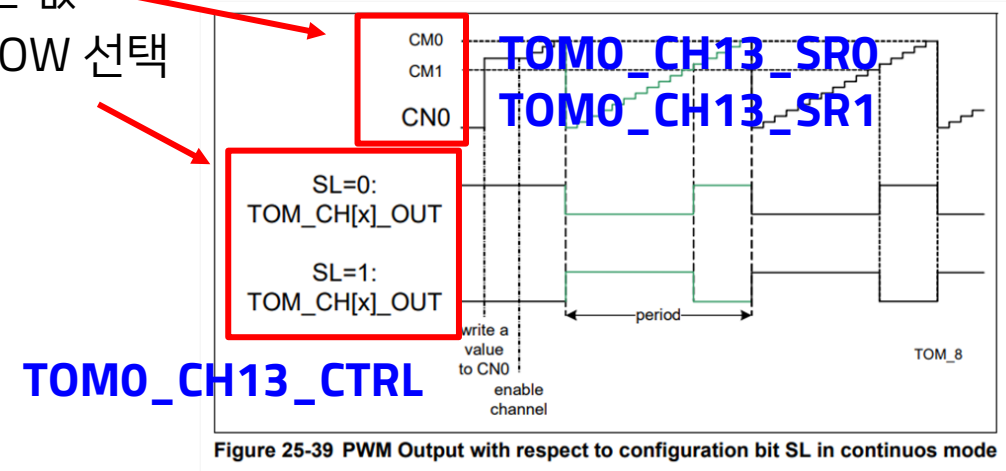
Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.2930

GTM 사용을 위한 레지스터 설정

:원하는 PWM 신호 생성 위한 TOM 설정

- 본 실습에서 사용하는 **TOM0**의 **채널 13**에 대한 레지스터 설정 필요
 - TOM0에 입력될 FXCLK 주파수 선택
 - PWM 신호의 주기 및 Duty Cycle 결정을 위한 값
 - Duty Cycle 동안 출력될 신호의 HIGH 또는 LOW 선택
- GTM 레지스터 항목에서
 - TOM0_CH13_CTRL** 레지스터 설정 필요
 - TOM0_CH13_SR0** 레지스터 설정 필요
 - TOM0_CH13_SR1** 레지스터 설정 필요

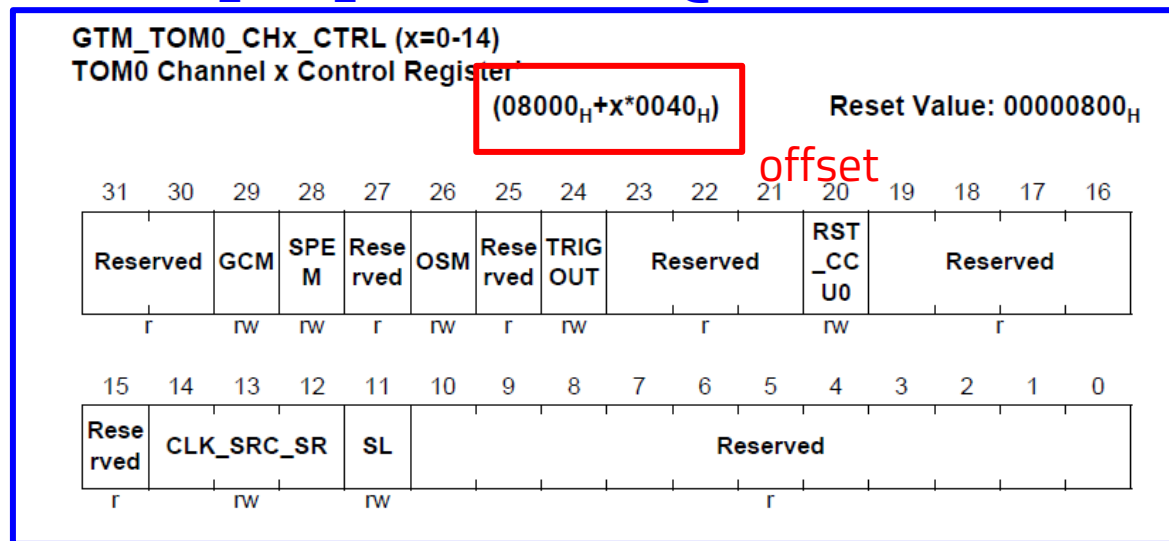
Infineon-TC27x_D-step-UM-v02_02-EN.pdf
p.2930



GTM 레지스터 설정 – TOM0_CH13_CTRL

1. GTM 레지스터 영역의 주소 찾기
 - 시작 주소 (Base address) = **0xF0100000**
2. 사용할 레지스터의 주소 찾기 (채널 13 이므로 x = 13)
 - **TOM0_CH13_CTRL** 의 Offset Address = $0x8000 + (x * 0x40) = \mathbf{0x8340}$
 - TOM0_CH13_CTRL 레지스터 주소 = $0xF0100000 + 0x8340 = \mathbf{0xF0108340}$

TOM0_CH1_CTRL 레지스터 @ 0xF0108340



[Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.2953](#)

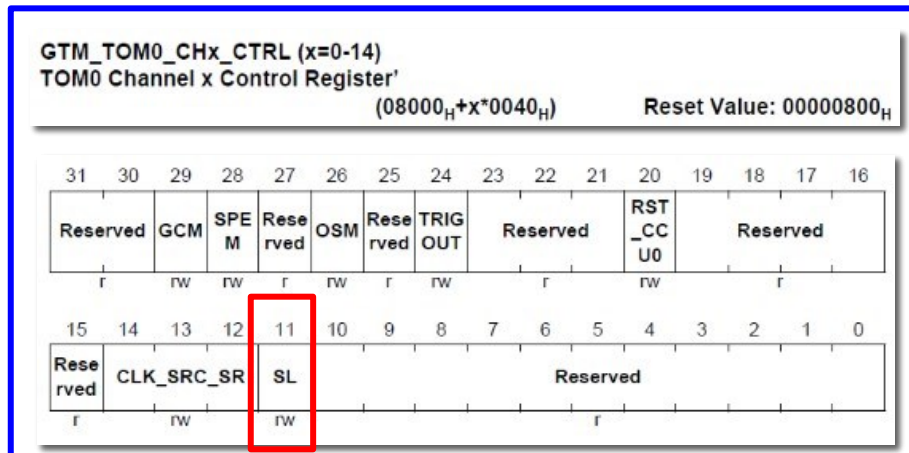
GTM 레지스터 설정 – TOM0_CH13_CTRL

:PWM 신호 중 duty cycle을 어떤 값으로 출력할 지 설정

3. 레지스터 write 값 결정

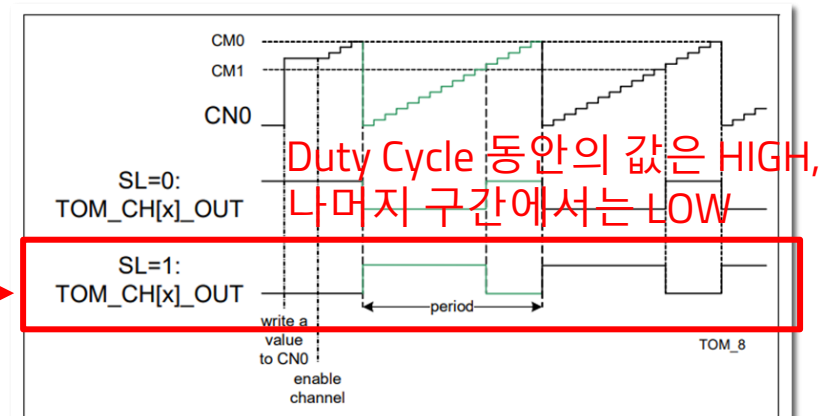
- TOM0 채널13의 동작을 설정
- PWM 신호의 Duty Cycle 영역에서 신호의 값을 HIGH로 출력하기 위해 **SL** 영역에 **0x1 write**

TOM0_CH13_CTRL 레지스터 @ 0xF0108340



| Field | Bits | Type | Description |
|-------|------|------|--|
| SL | 11 | rw | Signal level for duty cycle 0 _B Low signal level 1 _B High signal level If the output is disabled, the output TOM_OUT[x] is set to inverse value of SL. |

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.2930



Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.2953

Figure 25-39 PWM Output with respect to configuration bit SL in continuous mode

GTM 레지스터 설정 – TOM0_CH13_CTRL

:FXU에서 생성하고 TOM에서 사용할 clock 주파수 설정

3. 레지스터 write 값 결정

- TOM0 채널13의 동작을 설정
- TOM0 에서 사용할 FXCLK clock 신호의 주파수를 결정하기 위해 **CLK_SRC_SR 영역에 0x0 write**
- 현재 GTM 에서 사용하는 system clock 주파수는 100 MHz → **FXCLK = 6250 kHz**

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.2953

| GTM_TOM0_CHx_CTRL (x=0-14) TOM0 Channel x Control Register' | | | | | | | | | | | | | | | |
|--|------------|-------|----------|----------|----------|----------|----------|----------|----------|----------|-----------|----------|----------|----------|----------|
| (08000_H+x*0040_H) | | | | | | | | | | | | | | | |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | GCM | SPE M | Reserved | OSM | Reserved | TRIG OUT | Reserved | Reserved | Reserved | Reserved | RST_CC U0 | Reserved | Reserved | Reserved | Reserved |
| r | rw | rw | r | rw | r | rw | r | r | r | r | rw | r | r | r | r |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | CLK_SRC_SR | SL | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved |
| r | rw | rw | r | r | r | r | r | r | r | r | r | r | r | r | r |

TOM0_CH13_CTRL 레지스터
@ 0xF0108340

FXU로 입력되는
system clock을
 $2^4 (= 16)$ 으로
나눠서 FXCLK 로
사용
→ $100 \text{ MHz} / 16$
= 6250 kHz

| CLK_SRC_SR | [14:12] | rw | |
|------------|---------|----|---|
| | | | Clock source select for channel The register CLK_SRC is updated with the value of CLK_SRC_SR together with the update of register CM0 and CM1. The input of the FX clock divider depends on the value of FXCLK_SEL (see CMU). 000 _B CMU_FXCLK(0) selected: clock selected by FXCLKSEL 001 _B CMU_FXCLK(1) selected: clock selected by FXCLKSEL / 2 ⁴ 010 _B CMU_FXCLK(2) selected: clock selected by FXCLKSEL / 2 ⁸ 011 _B CMU_FXCLK(3) selected: clock selected by FXCLKSEL / 2 ¹² 100 _B CMU_FXCLK(4) selected: clock selected by FXCLKSEL / 2 ¹⁶ 101 _B no CMU_FXCLK selected, clock of channel stopped 110 _B no CMU_FXCLK selected, clock of channel stopped 111 _B no CMU_FXCLK selected, clock of channel stopped Note: if clock of channel is stopped (i.e. CLK_SRC = 101/110/111), the channel can only be restarted by resetting CLK_SRC_SR to a value of 000 to 100 and forcing an update via the force update mechanism. |

GTM 레지스터 설정 - TOM0_CH13_CTRL

:연속적으로 PWM 신호가 생성되는 Continuous 모드

3. 레지스터 write 값 결정

- TOM0 채널13에서 생성되는 PWM 신호를 continuous 모드로 사용하기 위해 **OSM 영역에 0x0 write**

PWM 주기가 연속으로 생성되는 모드

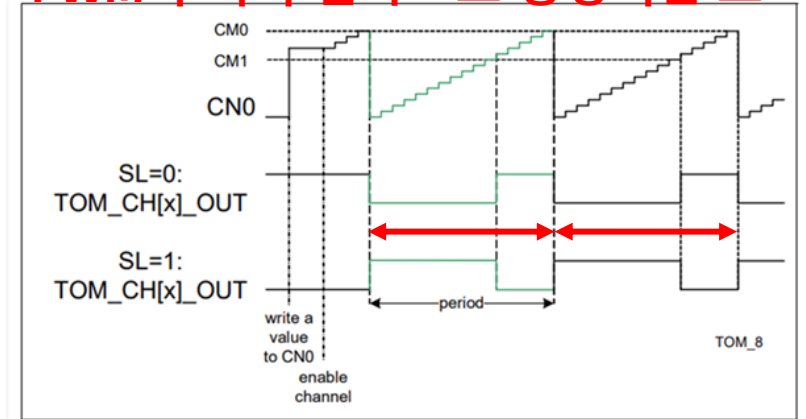


Figure 25-39 PWM Output with respect to configuration bit SL in continuous mode

[Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.2953](#)

[Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.2930](#)

| | | | | | | | | | | | | | | | |
|----------|------------|-------|----------|----------|----------|----------|----------|----------|-----------|----------|----------|----------|----------|----------|----------|
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | GCM | SPE M | Reserved | Reserved | OSM | Reserved | TRIG OUT | Reserved | RST_CC U0 | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved |
| r | rw | rw | r | r | rw | r | rw | r | rw | r | r | r | r | r | r |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | CLK_SRC_SR | SL | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved |
| r | rw | rw | r | r | r | r | r | r | r | r | r | r | r | r | r |

| | | | |
|-----|----|----|---|
| OSM | 26 | rw | One-shot mode In this mode the counter CN0 counts for only one period The length of period is defined by CM0 A write access to the register CN0 triggers the start of counting <div>0_B One-shot mode disabled</div> <div>1_B One-shot mode enabled</div> |
|-----|----|----|---|

TOM0_CH13_CTRL 레지스터 @ 0xF0108340

**One-shot 모드 disable
= Continuous 모드**

GTM 레지스터 update 하는 Trigger 신호 생성

:PWM의 1주기에서 생성하는 Update Trigger 신호

- 앞으로 설정할 GTM 레지스터 (SR0, SR1 등...) 들이 실제 하드웨어에 적용될 수 있도록 하는 **Update Trigger** 신호를
- 1) SW에서 생성할 수 있음
- 2) PWM의 1주기에 도달했을 때 생성할 수 있음

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.2923

25.11.2.2 TGC Subunit

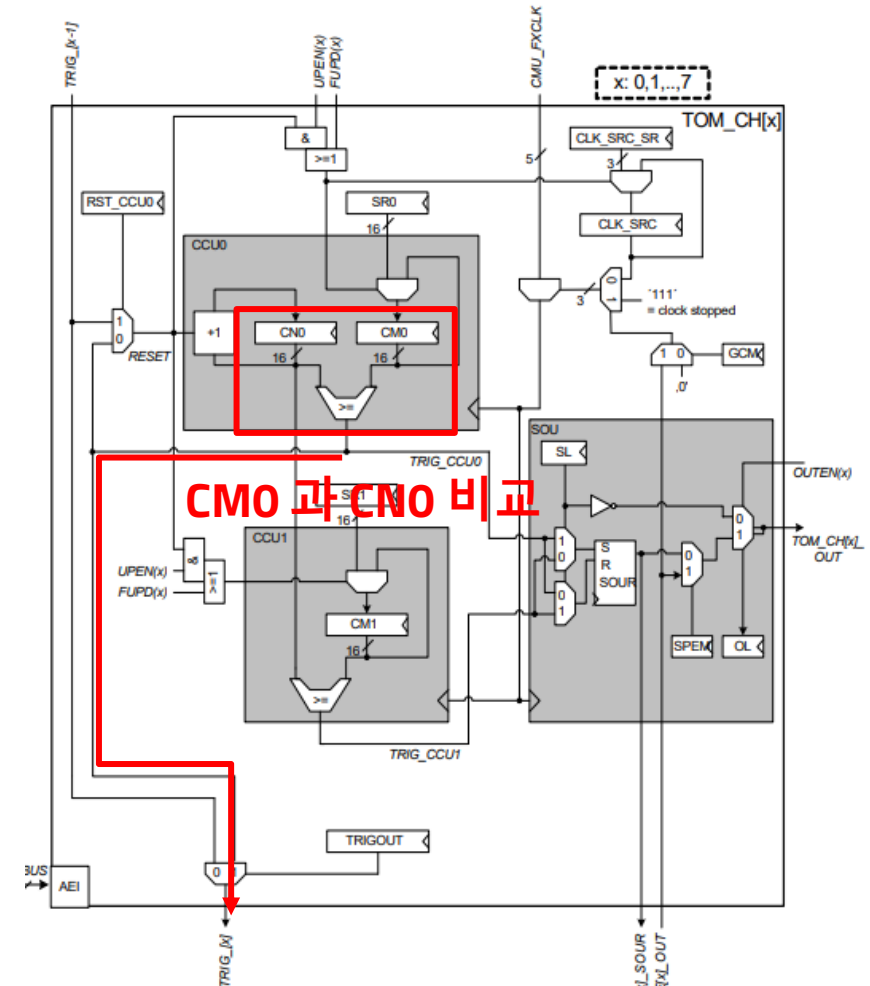
Each of the first three individual mechanisms (enable/disable of the channel, output enable and force update) can be driven by three different trigger sources.

The three trigger sources are:

- the host CPU (bit HOST_TRIG of register TOMi_TGCy_GLB_CTRL)
- the TBU time stamp (signal TBU_TS0..TBU_TS1..TBU_TS2)
- the internal trigger signal TRIG (bunch of trigger signals TRIG_[x])

Note: The trigger signal is only active for one configured CMU clock period.

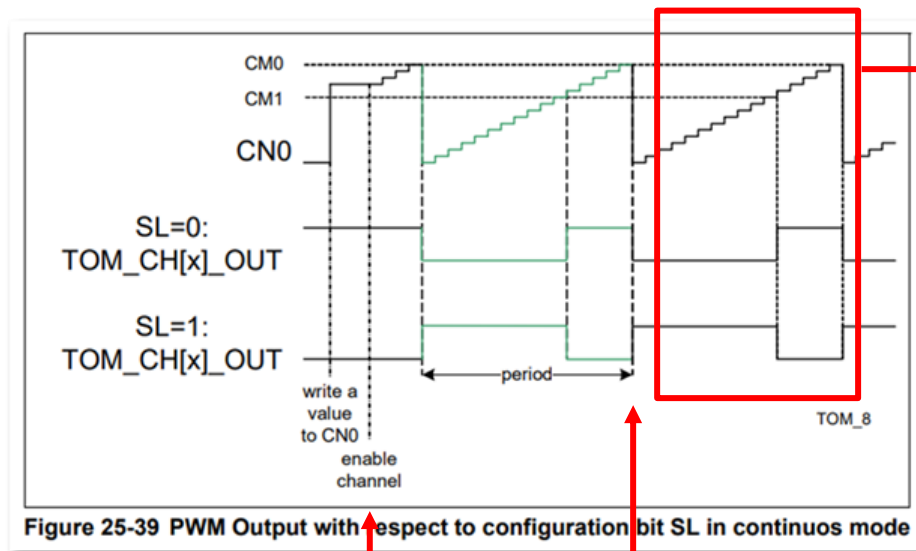
Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.2919



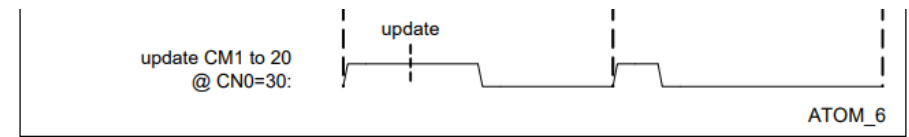
GTM 레지스터 update 하는 Trigger 신호

:Continuous 모드에서 PWM 주기, duty cycle 적용

- PWM의 1주기에 도달할 때마다 update trigger 신호 발생
= CN0 카운터 값이 CM0에 도달하는 시점



새로 업데이트된
CM0, CM1 값으로
PWM 출력



[Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.2928](#)

최초 SW에서 발생한
Update Trigger에 의해
시작

PWM 1주기 도달
→ SR0, SR1 레지스터 (shadow) 에
저장된 값을 각각 CM0, CM1로 copy

GTM 레지스터 설정 – TOM0_CH13_CTRL

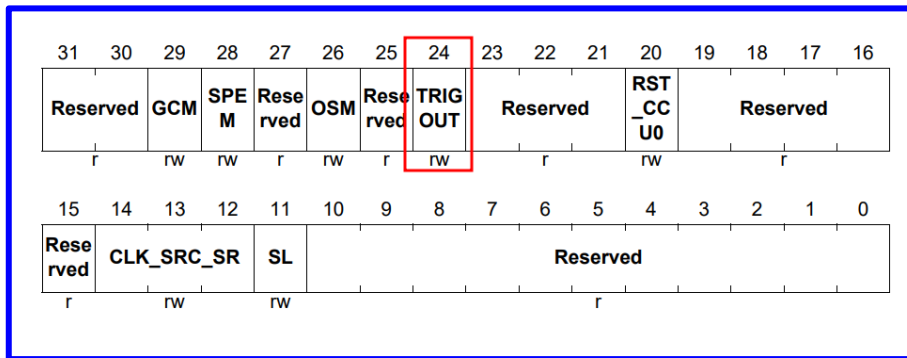
:PWM 신호에서 발생하는 trigger 신호 설정

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.2923

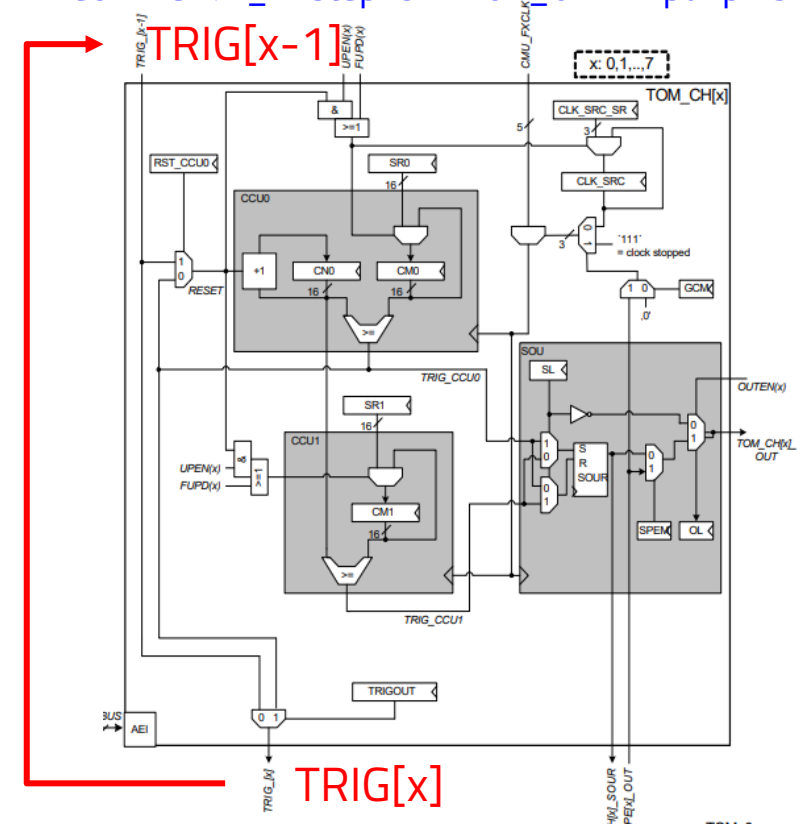
3. 레지스터 write 값 결정

- **PWM 신호의 1주기 도달 (CNO = CM0)시 발생하는 trigger (= TRIG) 신호를 Next trigger 신호**로 사용하기 위해 **TRIGOUT** 영역에 **0x0** write

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.2953



TOM0_CH13_CTRL 레지스터 @ 0xF0108340



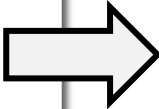
| | | | |
|---------|----|----|---|
| TRIGOUT | 24 | rw | Trigger output selection (output signal TRIG_[x]) of module TOM_CH[x] |
| | | | 0 _B TRIG_[x] is TRIG_[x-1] |
| | | | 1 _B TRIG_[x] is TRIG_CC U0 |

TRIG 신호 선택

Lab7: GTM 레지스터 설정 – TOM0_CH13_CTRL

:PWM 신호에서 발생하는 trigger 신호 설정

```
70 void initGTM(void)
71 {
72     // Password Access to unlock SCU_WDTSCON0
73     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
74     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0); // wait until unlocked
75
76     // Modify Access to clear ENDINIT
77     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) & ~(1 << ENDINIT_BIT_LSB_IDX);
78     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0); // wait until locked
79
80     GTM_CLC.U &= ~(1 << DISR_BIT_LSB_IDX); // enable GTM
81
82     // Password Access to unlock SCU_WDTSCON0
83     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
84     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0); // wait until unlocked
85
86     // Modify Access to set ENDINIT
87     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
88     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0);
89
90     while((GTM_CLC.U & (1 << DISS_BIT_LSB_IDX)) != 0); // wait until GTM module enabled
91
92
93     // GTM clock configuration
94     GTM_CMU_FXCLK_CTRL.U &= ~(0xF << FXCLK_SEL_BIT_LSB_IDX); // input clock of CMU_FXCLK --> CMU_GCLK_EN
95     GTM_CMU_CLK_EN.U |= 0x2 << EN_FXCLK_BIT_LSB_IDX; // enable all CMU_FXCLK
96
97     // GTM TOM0 PWM configuration
98     GTM_TOM0_TGC0_GLB_CTRL.U |= 0x2 << UPEN_CTRL1_BIT_LSB_IDX; // TOM channel 1 update enable
99
100    GTM_TOM0_TGC0_ENDIS_CTRL.U |= 0x2 << ENDIS_CTRL1_BIT_LSB_IDX; // enable channel 1 on update trigger
101
102    GTM_TOM0_TGC0_OUTEN_CTRL.U |= 0x2 << OUTEN_CTRL1_BIT_LSB_IDX; // enable channel 1 output on update trigger
103
104    GTM_TOM0_CH1_CTRL.U |= 0x1 << SL_BIT_LSB_IDX; // high signal level for duty cycle
105    GTM_TOM0_CH1_CTRL.U |= 0x1 << CLK_SRC_SR_BIT_LSB_IDX; // clock source --> CMU_FXCLK(1) = 6250 kHz
106    GTM_TOM0_CH1_CTRL.U &= ~(0x1 << OSM_BIT_LSB_IDX); // continuous mode enable
107    GTM_TOM0_CH1_CTRL.U &= ~(0x1 << TRIGOUT_BIT_LSB_IDX); // TRIG[x] = TRIG[x-1]
108
109    GTM_TOM0_CH1_SR0.U = 12500 - 1; // PWM freq. = 6250 kHz / 12500 = 500 Hz
110
111    GTM_TOM0_CH1_SR1.U = 1250 - 1; // duty cycle = 1250 / 12500 = 10 % (temporary)
112
113    GTM_TOUTSEL6.U &= ~(0x3 << SEL7_BIT_LSB_IDX); // TOUT103 --> TOM0 channel 1
114    // 103 = 16 * 6 + 7
115 }
116
```

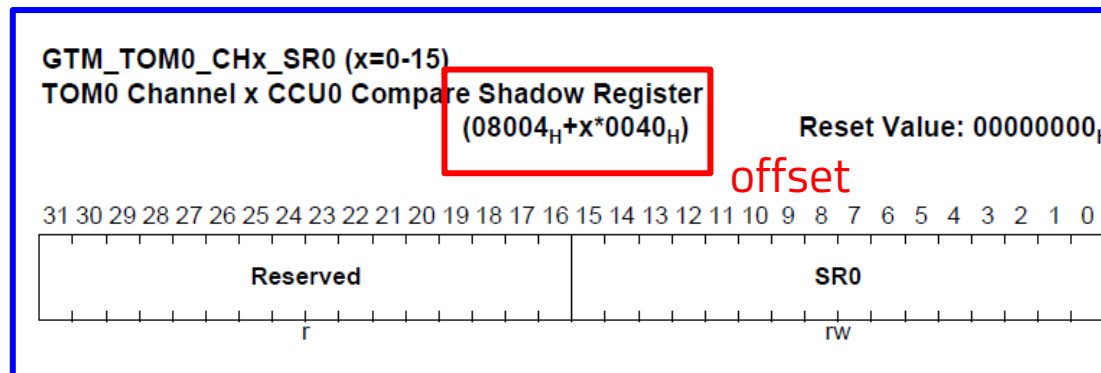


```
GTM_TOM0_CH13_CTRL.B.SL = 0x1;
GTM_TOM0_CH13_CTRL.B.CLK_SRC_SR = 0x1;
GTM_TOM0_CH13_CTRL.B.OSM = 0x0;
GTM_TOM0_CH13_CTRL.B.TRIGOUT = 0x0;
```


GTM 레지스터 설정 – TOM0_CH13_SR0

1. GTM 레지스터 영역의 주소 찾기
 - 시작 주소 (Base address) = **0xF0100000**
2. 사용할 레지스터의 주소 찾기 (채널 13 이므로 $x = 13$)
 - **TOM0_CH13_SR0** 의 Offset Address = $0x8004 + (x * 0x40) = \mathbf{0x8344}$
 - TOM0_CH13_SR0 레지스터 주소 = $0xF0100000 + 0x8344 = \mathbf{0xF0108344}$

TOM0_CH13_SR0 레지스터 @ 0xF0108344



[Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.2962](#)

GTM 레지스터 설정 – TOM0_CH13_SRO

:PWM 신호의 주기 설정

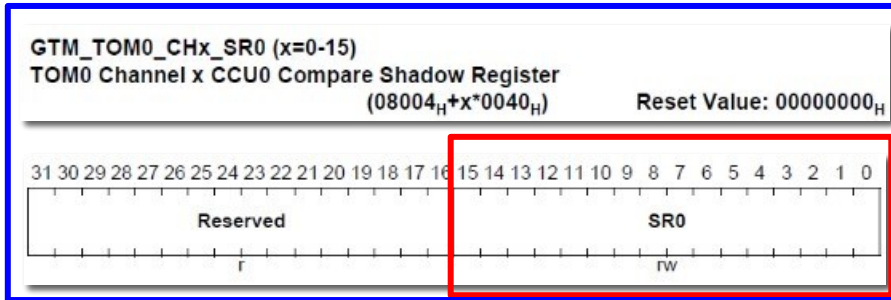
3. 레지스터 write 값 결정

- TOM0 채널13의 동작을 설정
- PWM 신호의 주기를 100us 로 설정하기 위해 **SRO 영역에 10진수 (625 – 1) write**
- SRO 레지스터에 설정할 CM0 값을 저장하면 Trigger로 update 시 CM0에 반영됨

$$PWM\ Period = \frac{CM0 + 1}{Freq.\ of\ CMU_FXCLK1}$$

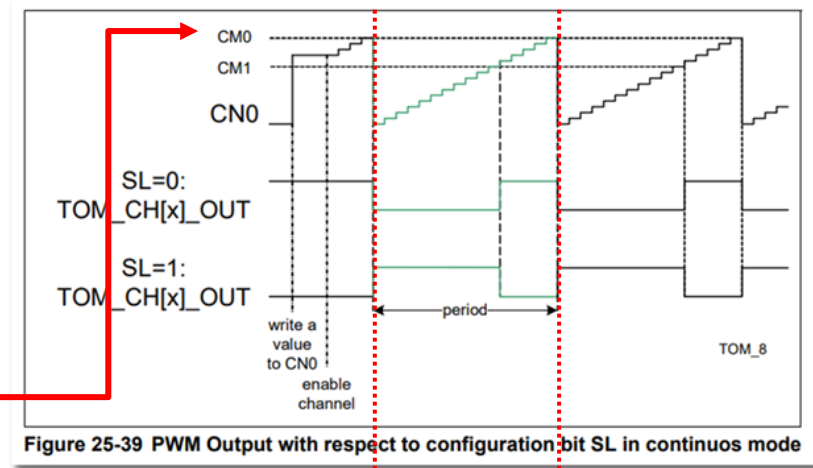
$$= \frac{625}{6250\ kHz} = 100\mu s$$

TOM0_CH1_SRO 레지스터@ 0xF0108044



| Field | Bits | Type | Description |
|----------|---------|------|--|
| SR0 | [15:0] | rw | TOM channel x shadow register SR0 for update of compare register CM0 |
| Reserved | [31:16] | r | Reserved Read as zero, should be written as zero |

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.2930



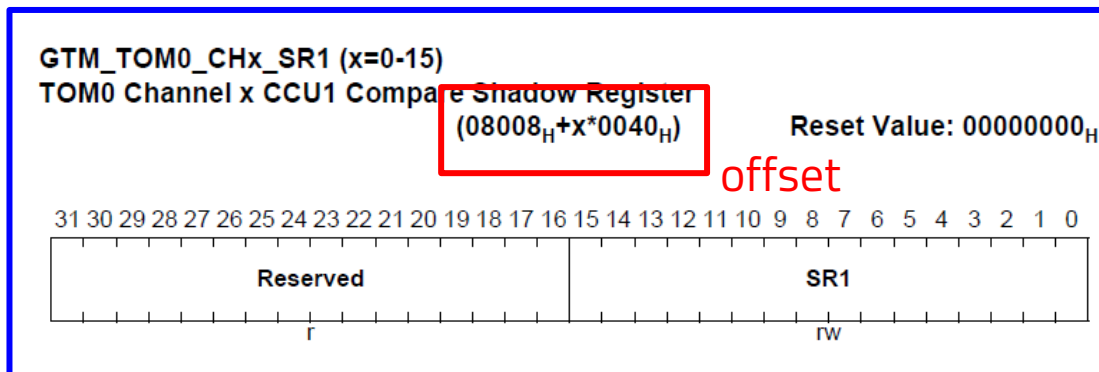
Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.2962

CN0 값이 0부터 증가하기 시작하여 CM0 값에 만나기 까지의 시간이 PWM 신호의 1주기

GTM 레지스터 설정 – TOM0_CH13_SR1

1. GTM 레지스터 영역의 주소 찾기
 - 시작 주소 (Base address) = **0xF0100000**
2. 사용할 레지스터의 주소 찾기 (채널 13 이므로 $x = 13$)
 - **TOM0_CH13_SR1** 의 Offset Address = $0x8008 + (x * 0x40) = \mathbf{0x8348}$
 - TOM0_CH13_SR1 레지스터 주소 = $0xF0100000 + 0x8348 = \mathbf{0xF0108348}$

TOM0_CH13_SR1 레지스터@ 0xF0108348



[Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.2964](#)

GTM 레지스터 설정 – TOM0_CH13_SR1

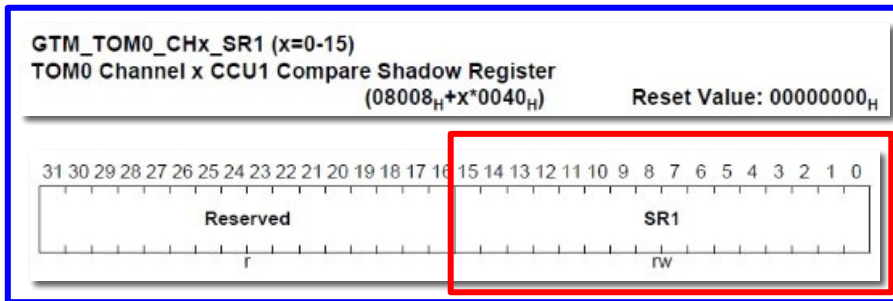
:PWM 신호의 duty cycle % 설정

3. 레지스터 write 값 결정

- TOM0 채널13의 동작을 설정
- PWM 신호의 Duty Cycle을 설정하기 위해 **SR1 영역에 write**
- SR1 레지스터에 설정할 CM1 값을 저장하면 Trigger로 update 시 CM1에 반영됨

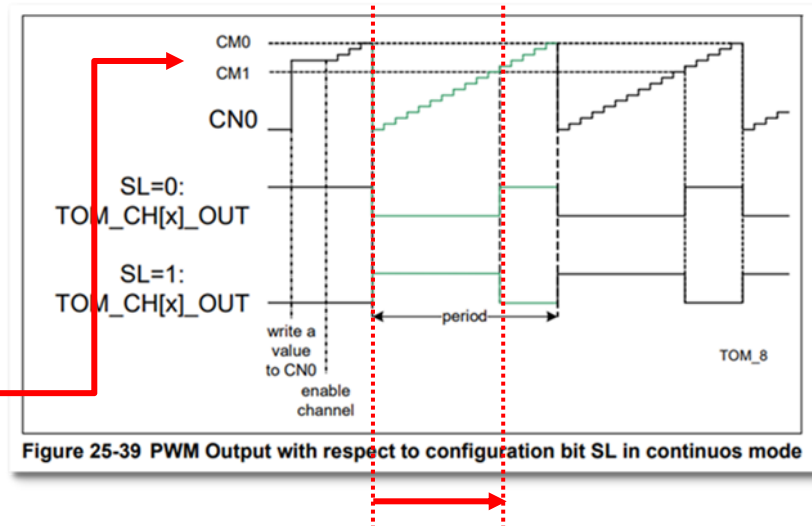
$$PWM \text{ Duty Cycle} = \frac{CM1 + 1}{CM0 + 1} \times 100 [\%]$$

TOM0_CH13_SR1 레지스터@ 0xF0108348



| Field | Bits | Type | Description |
|----------|---------|------|--|
| SR1 | [15:0] | rw | TOM channel x shadow register SR1 for update of compare register CM1 |
| Reserved | [31:16] | r | Reserved Read as zero, should be written as zero |

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.2930



Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.2964

CN0 값이 0부터 증가하기 시작하여 CM1 값에 만나기 까지의 시간이, PWM 신호의 1주기 시간 중 차지하는 비율이 Duty Cycle (%)

Lab8: GTM 레지스터 설정 – TOM0_CH13_SR1

:PWM 신호의 duty cycle % 설정

```
70 void initGTM(void)
71 {
72     // Password Access to unlock SCU_WDTSCON0
73     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
74     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0); // wait until unlocked
75
76     // Modify Access to clear ENDINIT
77     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) & ~(1 << ENDINIT_BIT_LSB_IDX);
78     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0); // wait until locked
79
80     GTM_CLC.U &= ~(1 << DISR_BIT_LSB_IDX); // enable GTM
81
82     // Password Access to unlock SCU_WDTSCON0
83     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
84     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0); // wait until unlocked
85
86     // Modify Access to set ENDINIT
87     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
88     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0);
89
90     while((GTM_CLC.U & (1 << DISS_BIT_LSB_IDX)) != 0); // wait until GTM module enabled
91
92
93     // GTM clock configuration
94     GTM_CMU_FXCLK_CTRL.U &= ~(0xF << FXCLK_SEL_BIT_LSB_IDX); // input clock of CMU_FXCLK --> CMU_GCLK_EN
95     GTM_CMU_CLK_EN.U |= 0x2 << EN_FXCLK_BIT_LSB_IDX; // enable all CMU_FXCLK
96
97     // GTM TOM0 PWM configuration
98     GTM_TOM0_TGC0_GLB_CTRL.U |= 0x2 << UPEN_CTRL1_BIT_LSB_IDX; // TOM channel 1 update enable
99
100    GTM_TOM0_TGC0_ENDIS_CTRL.U |= 0x2 << ENDIS_CTRL1_BIT_LSB_IDX; // enable channel 1 on update trigger
101
102    GTM_TOM0_TGC0_OUTEN_CTRL.U |= 0x2 << OUTEN_CTRL1_BIT_LSB_IDX; // enable channel 1 output on update trigger
103
104    GTM_TOM0_CH1_CTRL.U |= 0x1 << SL_BIT_LSB_IDX; // high signal level for duty cycle
105    GTM_TOM0_CH1_CTRL.U |= 0x1 << CLK_SRC_SR_BIT_LSB_IDX; // clock source --> CMU_FXCLK(1) = 6250 kHz
106    GTM_TOM0_CH1_CTRL.U &= ~(0x1 << OSM_BIT_LSB_IDX); // continuous mode enable
107    GTM_TOM0_CH1_CTRL.U &= ~(0x1 << TRIGOUT_BIT_LSB_IDX); // TRIG[x] = TRIG[x-1]
108
109    GTM_TOM0_CH1_SR0.U = 12500 - 1; // PWM freq. = 6250 kHz / 12500 = 500 Hz
110    GTM_TOM0_CH1_SR1.U = 1250 - 1; // duty cycle = 1250 / 12500 = 10 % (temporary)
111
112    GTM_TOUTSEL6.U &= ~(0x3 << SEL7_BIT_LSB_IDX); // TOUT103 --> TOM0 channel 1
113    // 103 = 16 * 6 + 7
114
115 }
```

임시로 설정
→ 나중에 duty cycle 변경

GTM_TOM0_CH13_SR0.U = 625 - 1;
GTM_TOM0_CH13_SR1.U = 312 - 1;

SW 프로그래밍

:sin 곡선에 해당하는 duty 값이 저장된 배열

- 1주기의 sin 곡선을 이루는 500개의 샘플링 point
- 1부터 625까지의 크기를 가짐

```
134
135 unsigned short sin_wave[500] = {
136     313, 317, 321, 325, 329, 333, 337, 340, 344, 348, 352, 356, 360, 364, 368, 372, 375, 379, 383, 387, 391, 395, 398, 402, 406, 410, 413, 417, 421, 424, 428,
137     432, 435, 439, 443, 446, 450, 453, 457, 460, 464, 467, 470, 474, 477, 480, 484, 487, 490, 494, 497, 500, 503, 506, 509, 512, 515, 518, 521, 524, 527, 530,
138     533, 535, 538, 541, 543, 546, 549, 551, 554, 556, 559, 561, 563, 566, 568, 570, 572, 575, 577, 579, 581, 583, 585, 587, 589, 590, 592, 594, 596, 597, 599,
139     600, 602, 603, 605, 606, 607, 609, 610, 611, 612, 613, 614, 615, 616, 617, 618, 619, 620, 620, 621, 622, 622, 623, 623, 624, 624, 624, 624, 625, 625, 625,
140     625, 625, 625, 625, 625, 625, 624, 624, 624, 623, 623, 622, 622, 621, 621, 620, 619, 618, 618, 617, 616, 615, 614, 613, 612, 611, 609, 608, 607, 605, 604, 603,
141     601, 600, 598, 596, 595, 593, 591, 590, 588, 586, 584, 582, 580, 578, 576, 574, 571, 569, 567, 565, 562, 560, 557, 555, 552, 550, 547, 545, 542, 539, 537,
142     534, 531, 528, 525, 523, 520, 517, 514, 511, 508, 505, 501, 498, 495, 492, 489, 485, 482, 479, 475, 472, 469, 465, 462, 458, 455, 451, 448, 444, 441, 437,
143     434, 430, 426, 423, 419, 415, 411, 408, 404, 400, 396, 393, 389, 385, 381, 377, 374, 370, 366, 362, 358, 354, 350, 346, 342, 339, 335, 331, 327, 323, 319,
144     315, 311, 307, 303, 299, 295, 291, 287, 284, 280, 276, 272, 268, 264, 260, 256, 252, 249, 245, 241, 237, 233, 230, 226, 222, 218, 215, 211, 207, 203, 200,
145     196, 192, 189, 185, 182, 178, 175, 171, 168, 164, 161, 157, 154, 151, 147, 144, 141, 137, 134, 131, 128, 125, 121, 118, 115, 112, 109, 106, 103, 101, 98,
146     95, 92, 89, 87, 84, 81, 79, 76, 74, 71, 69, 66, 64, 61, 59, 57, 55, 52, 50, 48, 46, 44, 42, 40, 38, 36, 35, 33, 31, 30, 28, 26, 25, 23, 22, 21, 19, 18, 17,
147     15, 14, 13, 12, 11, 10, 9, 8, 8, 7, 6, 5, 5, 4, 4, 3, 3, 2, 2, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 3, 3, 4, 4, 5, 6, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15,
148     16, 17, 19, 20, 21, 23, 24, 26, 27, 29, 30, 32, 34, 36, 37, 39, 41, 43, 45, 47, 49, 51, 54, 56, 58, 60, 63, 65, 67, 70, 72, 75, 77, 80, 83, 85, 88, 91, 93,
149     96, 99, 102, 105, 108, 111, 114, 117, 120, 123, 126, 129, 132, 136, 139, 142, 146, 149, 152, 156, 159, 162, 166, 169, 173, 176, 180, 183, 187, 191, 194, 198, 202,
150     205, 209, 213, 216, 220, 224, 228, 231, 235, 239, 243, 247, 251, 254, 258, 262, 266, 270, 274, 278, 282, 286, 289, 293, 297, 301, 305, 309, 313
151 };
152
```

SW 프로그래밍

:VADC 모듈에서 변환된 디지털 값 사용 PWM Duty Cycle 제어

- main 함수 작성

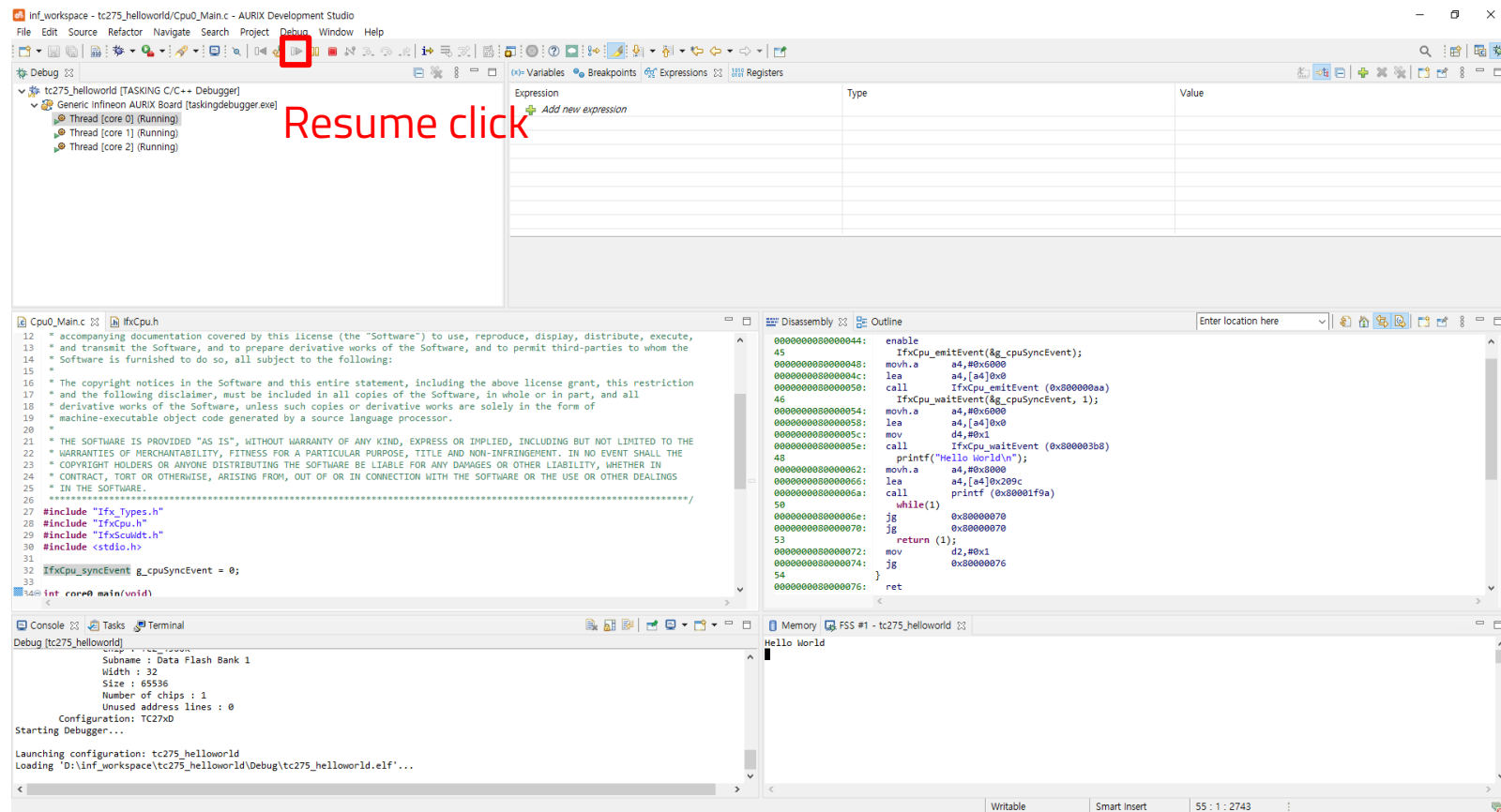
초기화 함수 호출

```
167 //initERU();
168 initCCU60();
169 initLED();
170 initRGBLED();
171 initVADC();
172 initGTM();
173 //initButton();
174 initPWM();
175
176
177
178 GTM_TOM0_TGC1_GLB_CTRL.B.HOST_TRIG = 0x1; // trigger update request signal
179
180
181 unsigned int idx = 0;
182
183 while(1)
184 {
185     for(unsigned int i = 0; i < 3000; i++)
186     {
187         GTM_TOM0_CH13_SR1.U = sin_wave[idx] - 1;
188     }
189     idx++;
190     idx %= 500;
191 }
192 return (1);
193 }
194
195 }
```

미리 저장된 sin
곡선에 해당하는
duty에서 선택

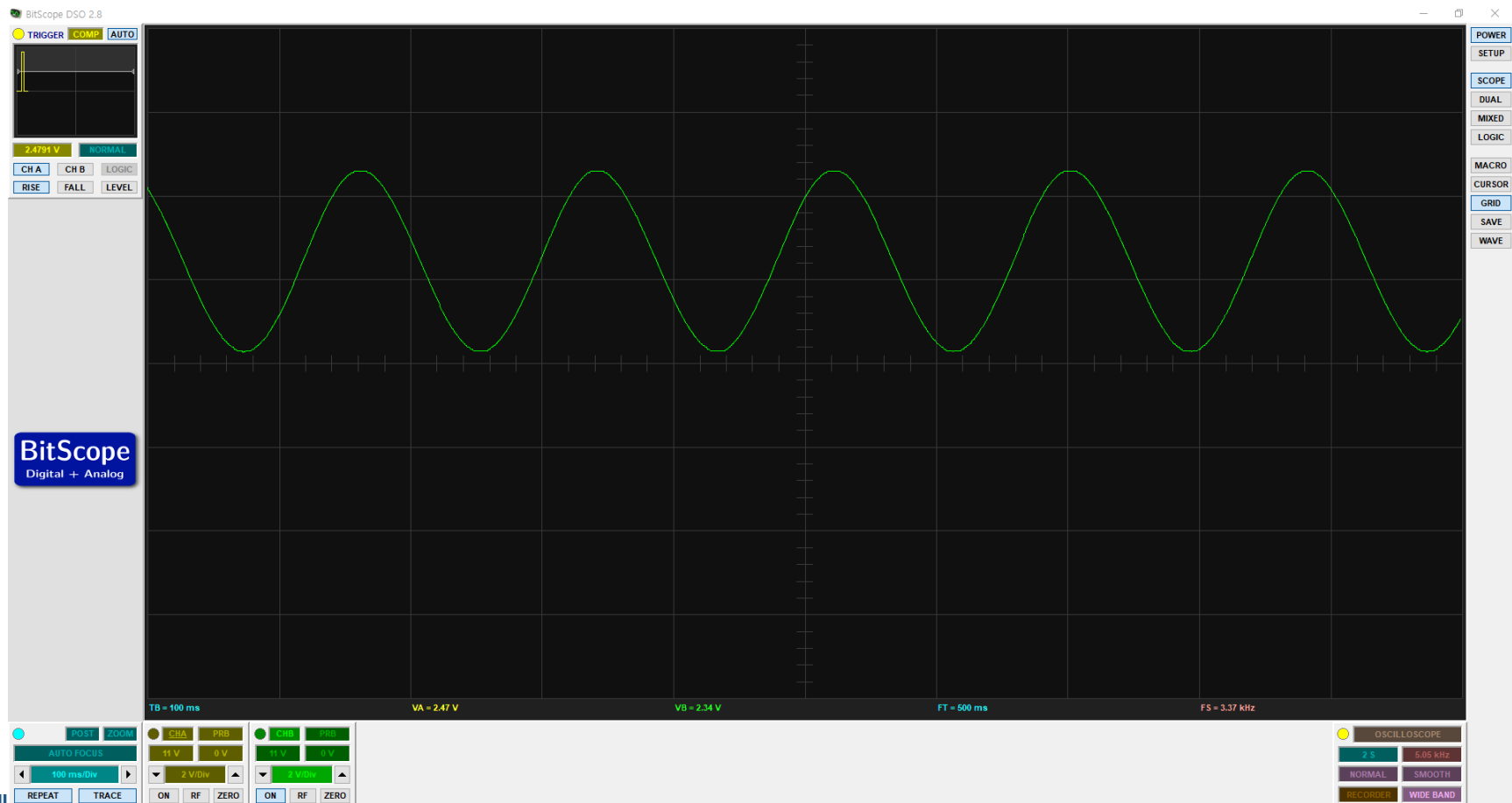
Build 및 Debug

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



동작 확인

- 오실로스코프를 사용해서 PWM신호를 출력하는 핀의 아날로그 측정값이 sin곡선을 그리는 것을 확인



감사합니다. 휴식~~