임베디드 기반 SW 개발 프로젝트

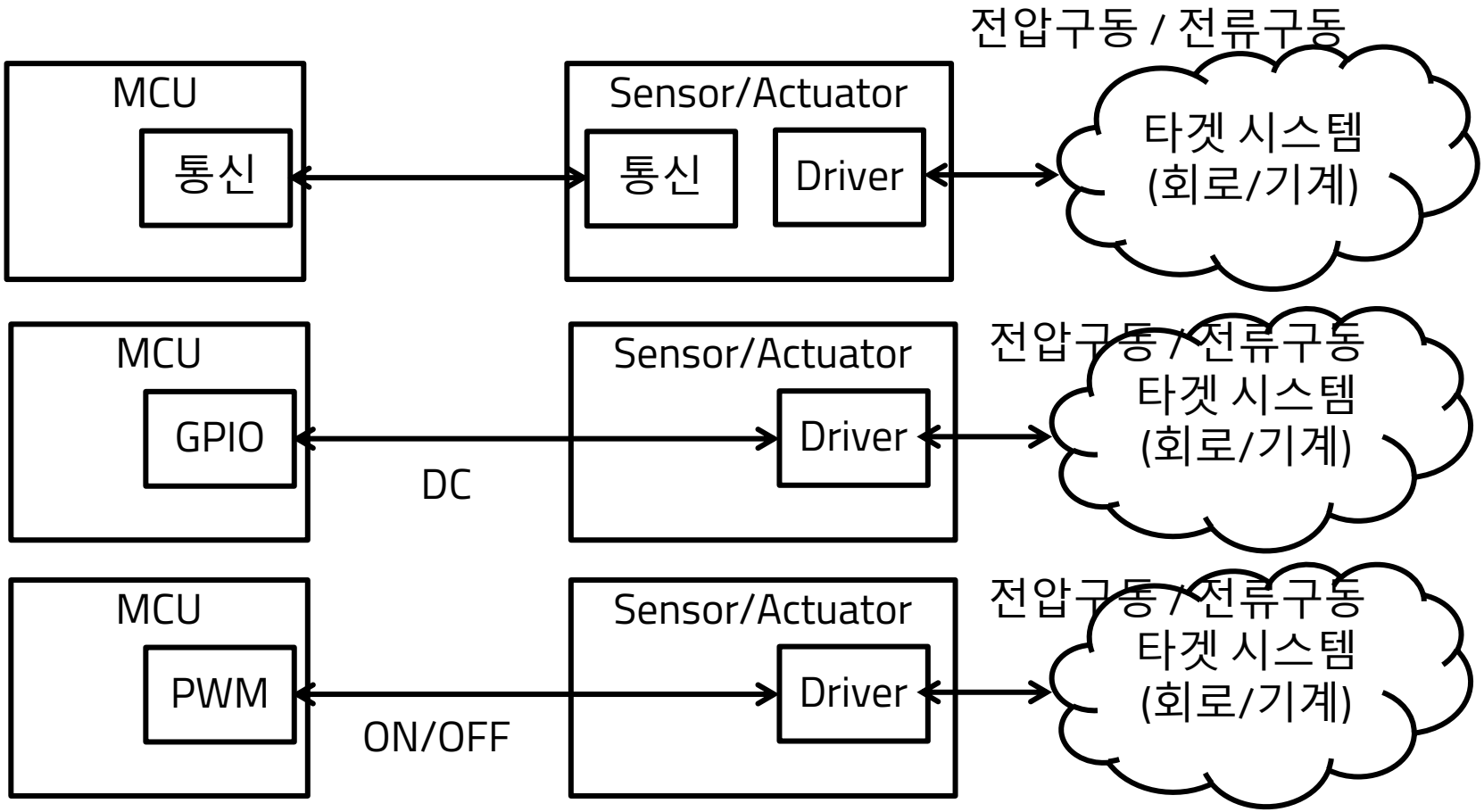
AURIX TC275 보드 PWM 사용하여 Buzzer 제어

- Buzzer -

현대자동차 입문교육
박대진 교수

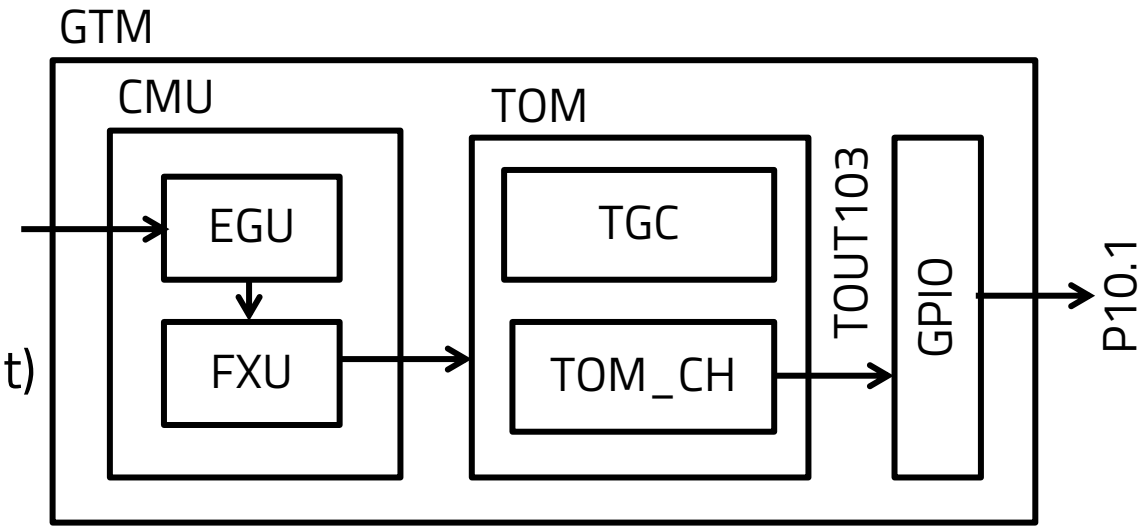
실습 목표

1. PWM 출력의 주기(period)를 제어해서 부저(Buzzer)의 음계를 변화시켜 본다.



사용된 약어 정리

- PWM (Pulse Width Modulation)
- GTM (Generic Timer Module)
- CMU (Clock Management Unit)
- EGU (External Generation Unit)
- FXU (Fixed Clock Generation Unit)
- TOM (Timer Output Module)
- TGC (TOM Global Control Unit)

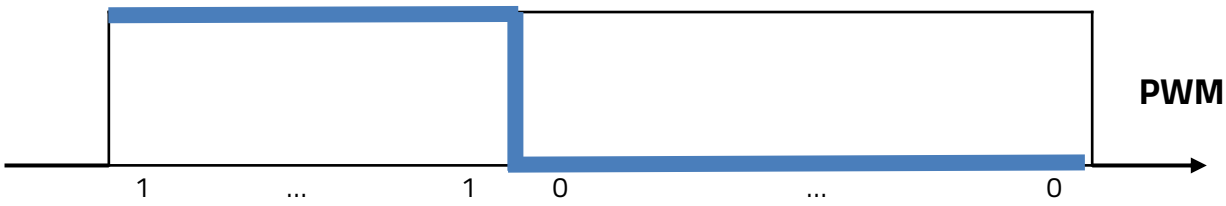


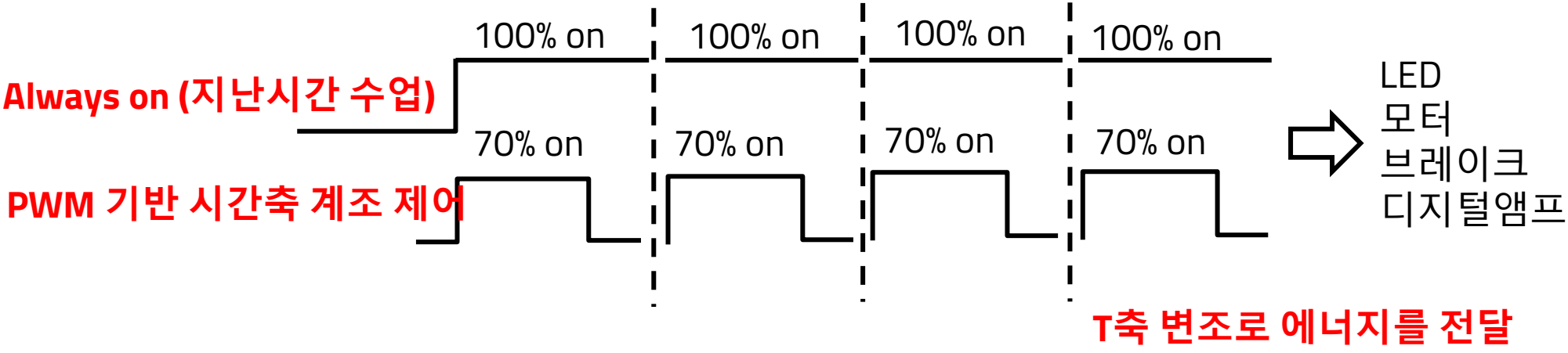
PWM (Pulse Width Modulation) 이란?

- PWM (Pulse Width Modulation) : One sample ➔ express **ratio of HIGH value** in one cycle
  = duty cycle

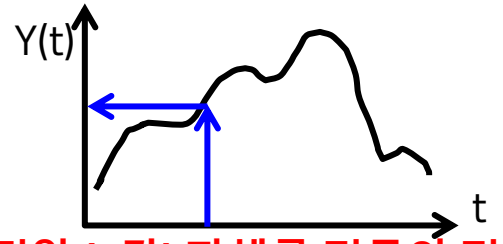
sample value : 100

256 listed '1' and '0'
(100 '1' and 156 '0')



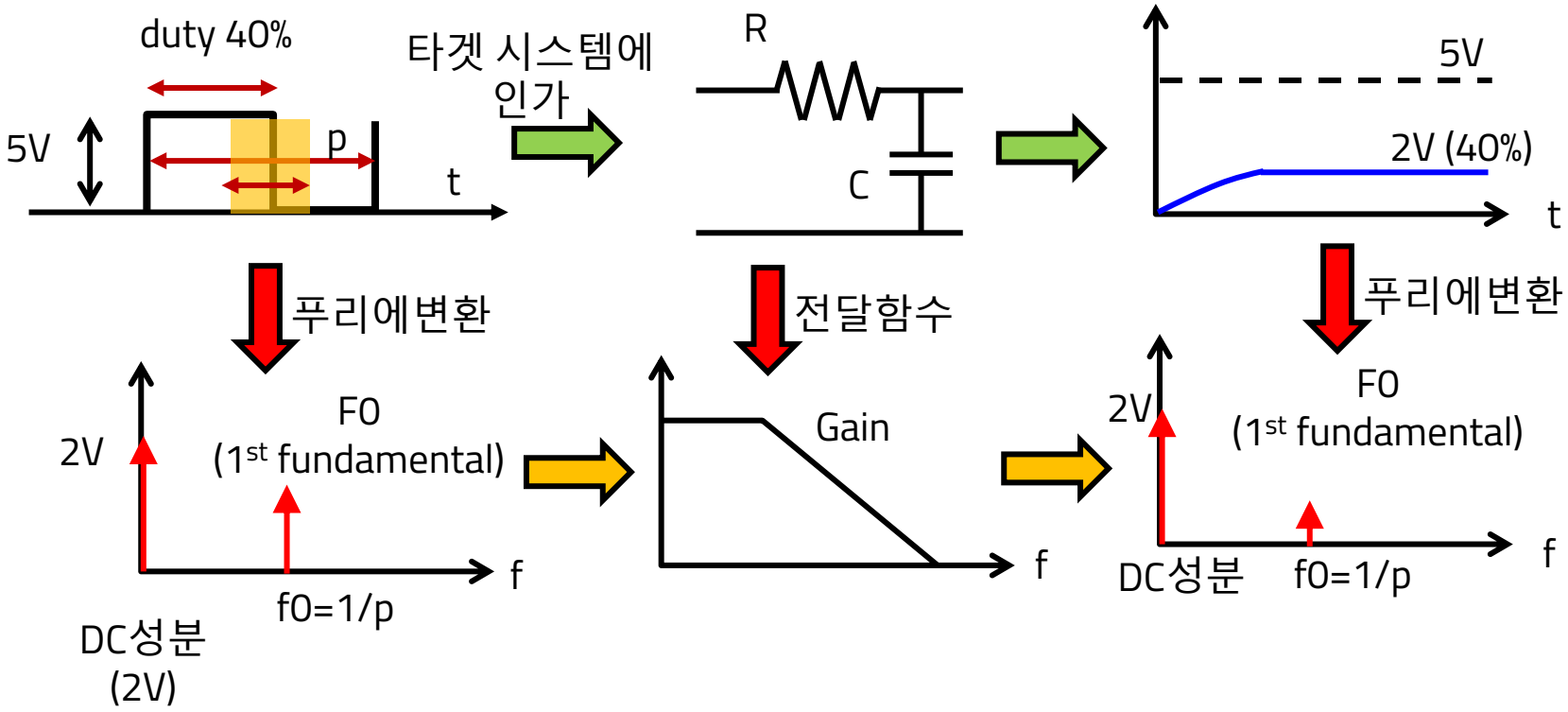


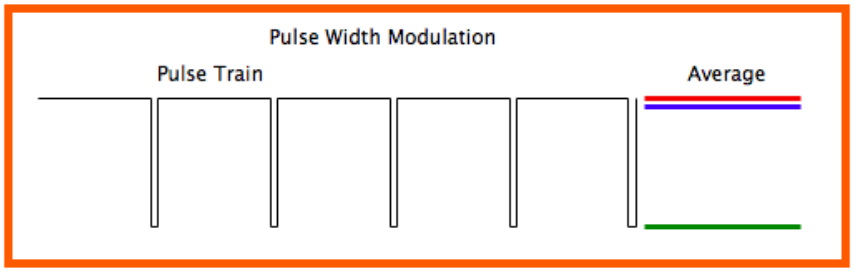
T축의 변조로, Y값을 제어하는 방법



전압 Y를 고정(5V) 시켜두고
t축 Edge를modulation

전압 (Y값) 자체를 전류와 전압 제어로modulation (DAC이용)



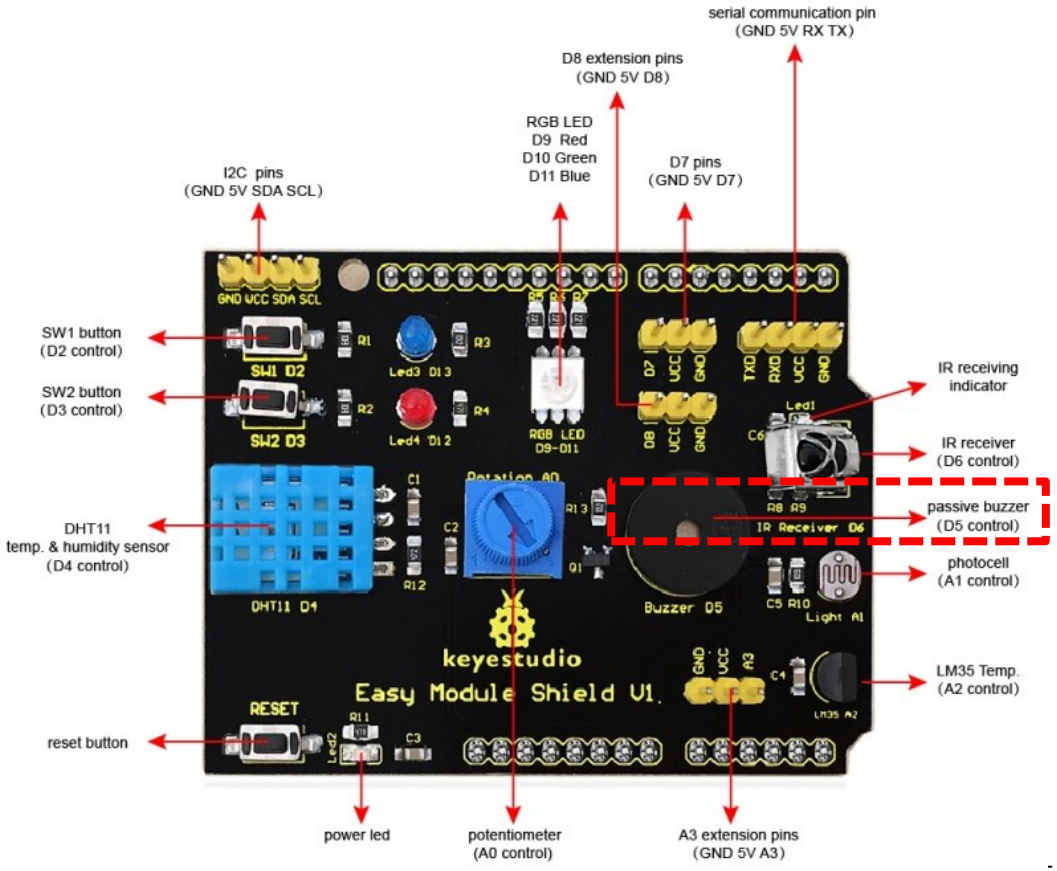


Buzzer 회로 구조 파악

: 확장 보드 PCB에 Buzzer가 연결된 회로

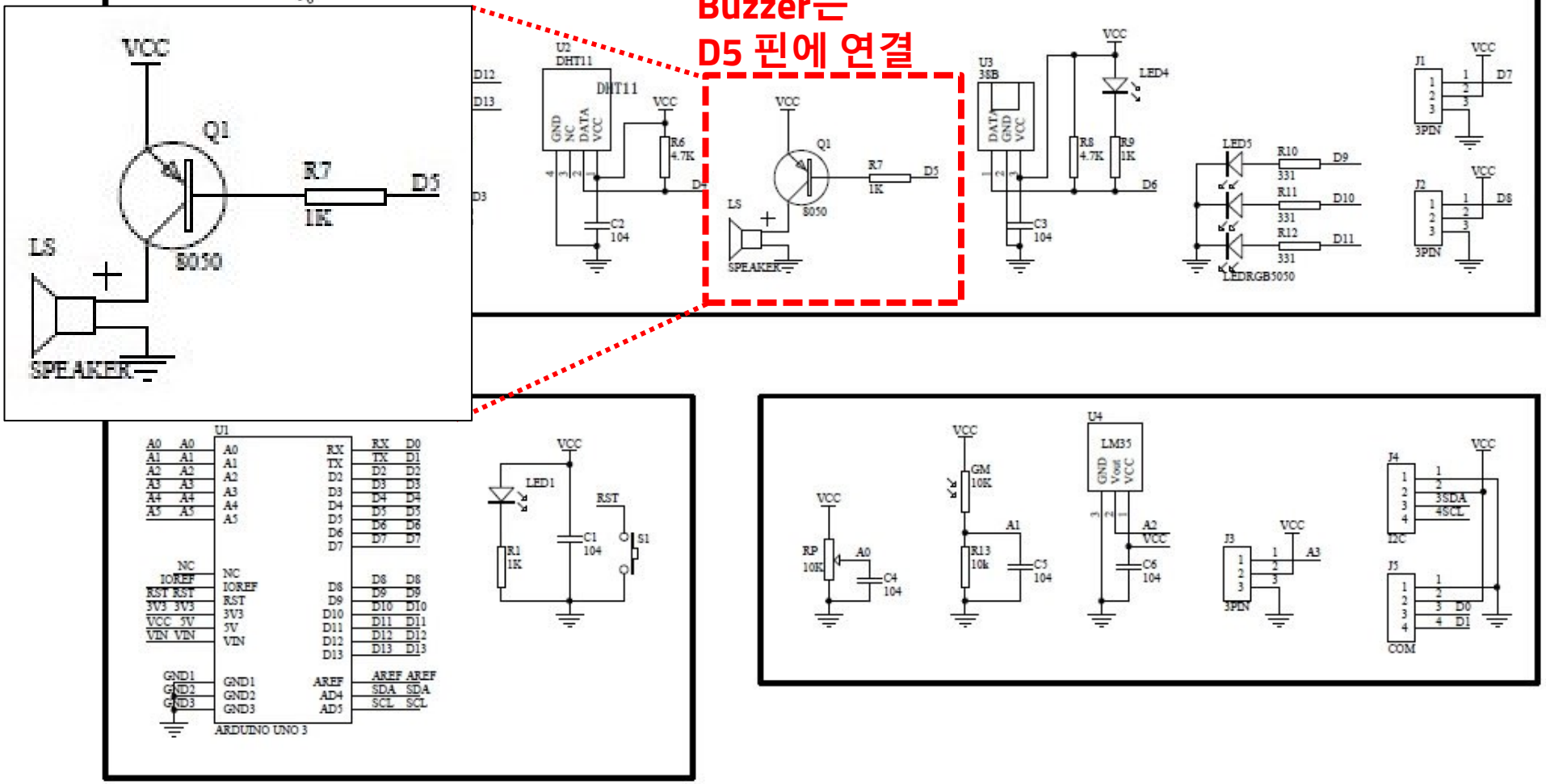
- 확장 보드의 Buzzer를 사용하기 위해서는 TC275 보드의 어떤 핀이 Easy Module Shield 확장 보드의 Buzzer와 연결되어 있는지 알아야 함

  → 데이터 시트 분석 필요



회로도 @ 확장 보드 데이터 시트

: Buzzer 회로 부분 확인



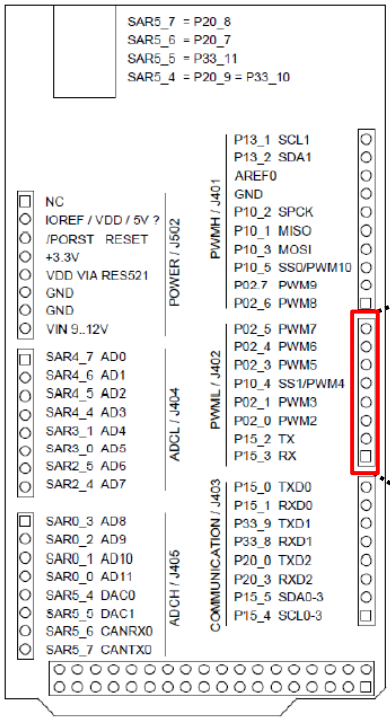
Pin Map @ TC275 보드

: Buzzer와 TC275 보드 핀과의 회로 연결 정보 확인

- 앞서 Buzzer는 확장 보드의 핀 D5에 연결 되어있는 것을 확인

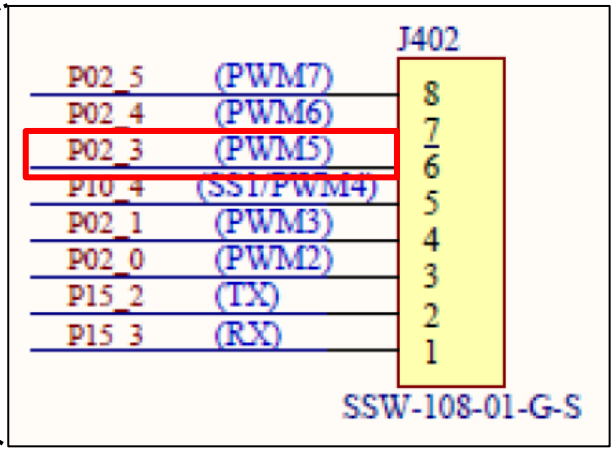
→ 그렇다면 확장 보드의 핀 **D5**는 **TC275** 보드의 어느 핀에 연결?

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



D5

Digital pin 3 (PWM)	PWML.4	P2.1
Digital pin 5 (PWM)	PWML.6	P2.3
Digital pin 6 (PWM)	PWML.7	P2.4



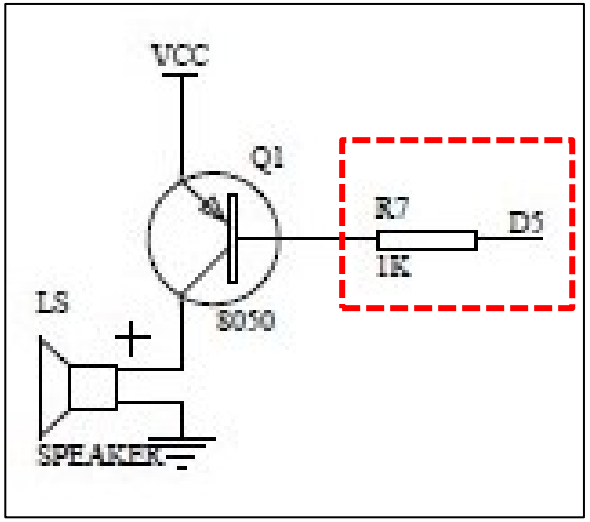
TC275 MCU I/O 중,
Port 02 pin 3 (P02.3)
와 연결됨

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

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

Buzzer를 제어하려면...

- **Buzzer의 음계를 결정하려면 MCU P02.3핀에 적합한 주기(period)의 PWM 신호를 인가해야 함**



(단위 : Hz)

옥타브 / 음계	1	2	3	4	5	6	7	8
C(도)	32.7032	65.4064	130.8128	261.6256	523.2511	1046.502	2093.005	4186.009
C#	34.6478	69.2957	138.5913	277.1826	554.3653	1108.731	2217.461	4434.922
D(레)	36.7081	73.4162	146.8324	293.6648	587.3295	1174.659	2349.318	4698.636
D#	38.8909	77.7817	155.5635	311.1270	622.2540	1244.508	2489.016	4978.032
E(미)	41.2034	82.4069	164.8138	329.6276	659.2551	1318.510	2637.020	5274.041
F(파)	43.6535	87.3071	174.6141	349.2282	698.4565	1396.913	2793.826	5587.652
F#	46.2493	92.4986	184.9972	369.9944	739.9888	1479.978	2959.955	5919.911
G(솔)	48.9994	97.9989	195.9977	391.9954	783.9909	1567.982	3135.963	6271.927
G#	51.9130	103.8262	207.6523	415.3047	830.6094	1661.219	3322.438	6644.875
A(라)	55.0000	110.0000	220.0000	440.0000	880.0000	1760.000	3520.000	7040.000
A#	58.2705	116.5409	233.0819	466.1638	932.3275	1864.655	3729.310	7458.620
B(시)	61.7354	123.4708	246.9417	493.8833	987.7666	1975.533	3951.066	7902.133

GPIO 레지스터 설정
: 핀의 입출력 방향 및 출력값 설정

- P02.3, 핀의 PWM 출력 모드 동작을 위해서는 GPIO Port 레지스터 항목에서
  - **P02_IOCR0**

- 1가지 레지스터에 설정이 필요함
  - **GTM 모듈의 PWM 출력 신호를 사용하도록**

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

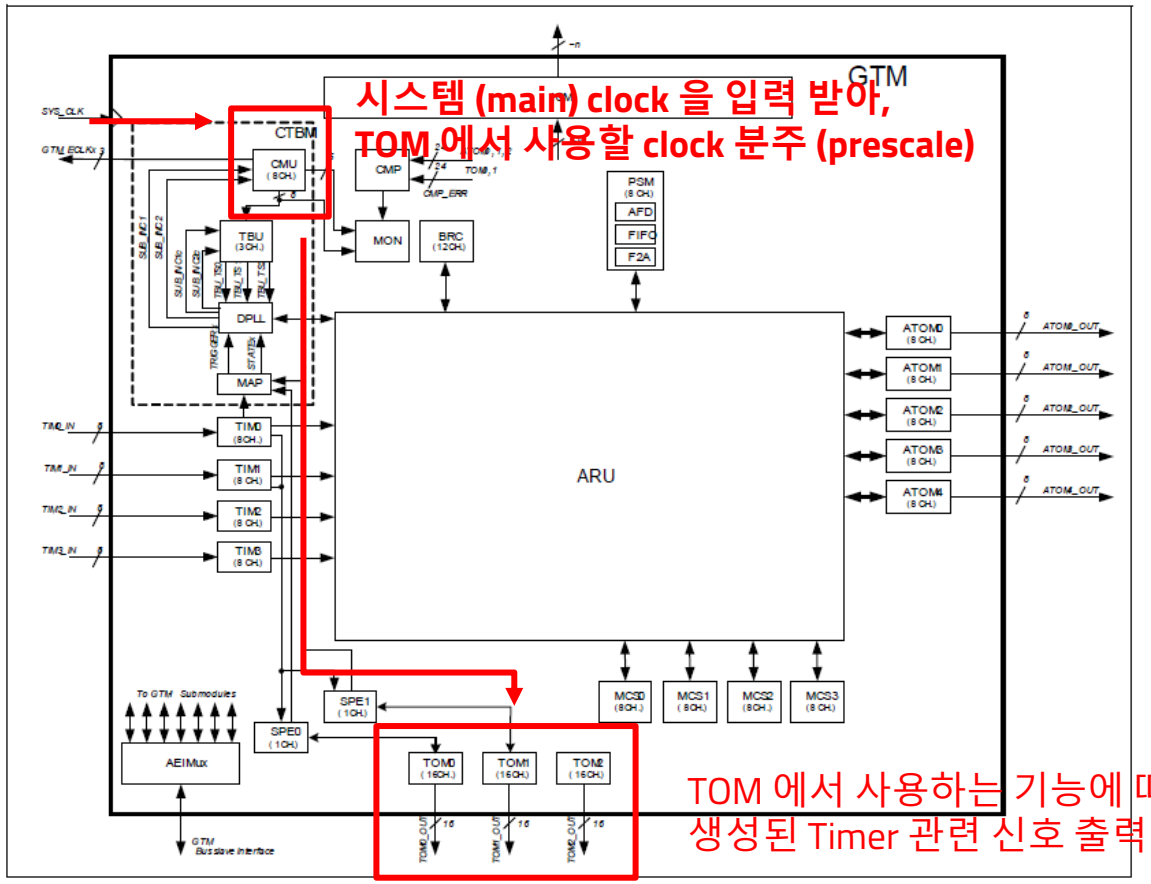
Port Pin	I/O	Pin Functionality	Associated Reg./ I/O Line	Port I/O Control Select.	
				Reg./Bit Field	Value
P02.3	I	General-purpose input	P02_IN.P3	P02_IOCR0. PC3	$0XXXX_B$
		GTM input	TIN3		
		CAN node 2 input	RXDCAN2B		
		ERAY input	RXDB2		
		PSI5 input	PSIRX0B		
		DSADC input	DSCIN5B		
		MSC1 input	SDI11		
		CIF input	CIFD3		
		ASCLIN1 input	ARX1G		
	O	General-purpose output	P02_OUT.P3		$1X000_B$
		GTM output	TOUT3		$1X001_B$
		ASCLIN2 output	ASLSO2		$1X010_B$
		QSPI3 output	SLSO34		$1X011_B$
		DSADC output	DSCOUT5		$1X100_B$

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.1152

PWM 신호 생성 flow

: GTM 모듈의 시스템 clock 사용한 PWM 신호 출력

- GTM (Generic Timer Module) 에는 다양한 기능을 가지는 submodule 들이 존재
  - ATOM, BRC, MCS, PSM, SPE, TIM, TOM 등 …
- 본 실습에서 사용하는 **PWM** 기능은 **TOM (Timer Output Module)** 을 사용함



Infineon-TC27x_D-step-UM-v02_02-EN.pdf
p.2731

Figure 25-1 GTM Architecture Block Diagram

GTM (Generic Timer Module) 사용을 위한 레지스터 설정 : GTM 모듈 enable 위해서는 보호 레지스터 잠금 해제 필요

- GTM 모듈 사용을 위해 Clock Control 레지스터에서 **GTM 모듈 enable** 필요

- GTM 레지스터 항목에서
  - **CLC** 레지스터 설정 필요

- GTM_CLC 레지스터는 System Critical 레지스터이므로 CPU ENDINIT 해제 후 수정해야함
- (CPU0 만을 사용하므로) SCU 레지스터 항목에서
  - **WDTCPU0CON0** 레지스터 설정 필요

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.3469

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 25-749
TIM0INSEL	TIM0 Input Select Register	$9FD10_H$	U, SV	U, SV, P	Application	Page 25-773

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

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

SCU 레지스터 설정 – WDTCPU0CON0

: GTM 모듈 사용 설정 과정을 보호하는 레지스터

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

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

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

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

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

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

Short Name	Long Name	Offset Addr. 1)	Access Mode Read	Access Mode Write	Reset	Description See
EMSR	Emergency Stop Register	0FC$_H$	U, SV	SV, SE, P	Application Reset	Page 7-291
WDTCPU0CON0	CPU0 WDT Control Register 0	100$_H$	U, SV	U, SV, 32,(CPU 0[2)])	Application Reset	Page 7-276
WDTCPU0CON1	CPU0 WDT Control	104$_H$	U, SV	SV,	Application	Page

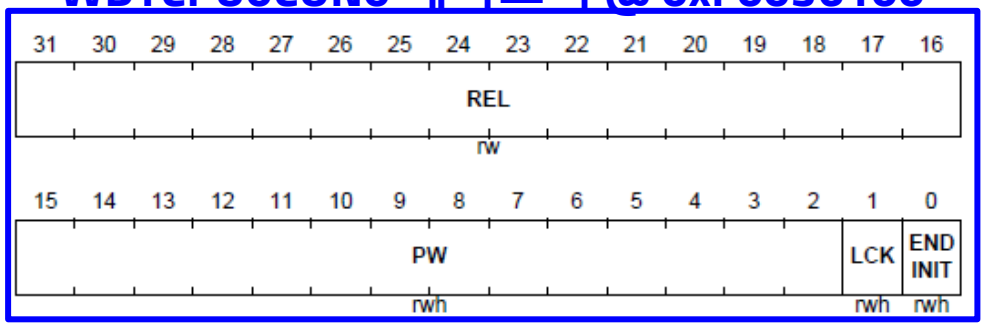
Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.697

SCU 레지스터 설정 – WDTCPU0CON0
:GTM 모듈 사용하도록 설정 위해 보호 레지스터 잠금 해제

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

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

WDTCPU0CON0 레지스터 @ 0xF0036100



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

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

PW	[15:2]	rwh	**User-Definable Password Field for Access to WDTxCON0** This bit field is written with an initial password value during a Modify Access. A read from this bitfield returns this initial password, but bits [7:2] are inverted (toggled) to ensure that a simple read/write is not sufficient to service the WDT.

단, PW 영역의 일부 PW[7:2] 는 반전해서 읽어야 함

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

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

3) 결정한 32bit 값을 WDTCPU0CON0 레지스터에 한 번에 write

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

Lock 상태가 해제되면 LCK bit 가 0으로 읽힘

Lab1: 헤더 파일 작성

:GTM 레지스터의 각 영역(필드) LSB 비트 시작 위치 define 정의

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

→ 헤더 파일 참조 추가

```
// GTM registers
#define DISS_BIT_LSB_IDX            1
#define DISR_BIT_LSB_IDX            0
#define SEL3_BIT_LSB_IDX            6
#define EN_FXCLK_BIT_LSB_IDX        22
#define FXCLK_SEL_BIT_LSB_IDX       0

// GTM - TOM0 registers
#define UPEN_CTRL3_BIT_LSB_IDX      22
#define HOST_TRIG_BIT_LSB_IDX       0
#define ENDIS_CTRL3_BIT_LSB_IDX     6
#define OUTEN_CTRL3_BIT_LSB_IDX     6
#define FUPD_CTRL1_BIT_LSB_IDX      2
#define CLK_SRC_SR_BIT_LSB_IDX      12
#define SL_BIT_LSB_IDX              11
#define OSM_BIT_LSB_IDX             26
#define TRIGOUT_BIT_LSB_IDX         24
```

} GTM 레지스터 bit shift offset

Lab2: SCU 레지스터 설정 – WDTCPU0CON0
:GTM 모듈 사용하도록 설정 위해 보호 레지스터 잠금 해제

```
372
373 void initGTM(void)
374 {
375     // Password Access to unlock SCU_WDTSCON0
376     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
377     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0);    // wait until unlocked
378
379     // Modify Access to clear ENDINIT
380     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) & ~(1 << ENDINIT_BIT_LSB_IDX);
381     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0);    // wait until locked
382
383     GTM_CLC.U &= ~(1 << DISR_BIT_LSB_IDX);    // enable GTM
384
385     // Password Access to unlock SCU_WDTSCON0
386     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
387     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0);    // wait until unlocked
388
389     // Modify Access to set ENDINIT
390     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
391     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0);
392
393     while((GTM_CLC.U & (1 << DISS_BIT_LSB_IDX)) != 0); // wait until GTM module enabled
394
395
396     // GTM clock configuration
397     GTM_CMU_FXCLK_CTRL.U &= ~(0xF << FXCLK_SEL_BIT_LSB_IDX);        // input clock of CMU_FXCLK --> CMU_GCLK_EN
398     GTM_CMU_CLK_EN.U |= 0x2 << EN_FXCLK_BIT_LSB_IDX;                // enable all CMU_FXCLK
399
400
401     // set GTM TOM0 channel 11 - Buzzer
402     GTM_TOM0_TGC1_GLB_CTRL.U |= 0x2 << UPEN_CTRL3_BIT_LSB_IDX;      // TOM0 channel 11 enable
403     GTM_TOM0_TGC1_ENDIS_CTRL.U |= 0x2 << ENDIS_CTRL3_BIT_LSB_IDX;   // enable channel 11 on update trigger
404     GTM_TOM0_TGC1_OUTEN_CTRL.U |= 0x2 << OUTEN_CTRL3_BIT_LSB_IDX;   // enable channel 11 output on update trigger
405
406
407     // TOM0 CH11
408     GTM_TOM0_CH11_CTRL.U |= 0x1 << SL_BIT_LSB_IDX;                  // high signal level for duty cycle
409     GTM_TOM0_CH11_CTRL.U |= 0x1 << CLK_SRC_SR_BIT_LSB_IDX;          // clock source --> CMU_FXCLK(1) = 6250 kHz
410     GTM_TOM0_CH11_CTRL.U &= ~(0x0 << OSM_BIT_LSB_IDX);              // continuous mode
411     GTM_TOM0_CH11_CTRL.U &= ~(0x0 << TRIGOUT_BIT_LSB_IDX);
412     GTM_TOM0_CH11_SR0.U = 12500 - 1;                                // PWM freq. = 6250 kHz / 12500 = 500 Hz (temp)
413     GTM_TOM0_CH11_SR1.U = 6250 - 1;                                 // duty cycle = 6250 / 12500 = 50 % (temp)
414
415     // TOUT pin selection
416     GTM_TOUTSEL0.U &= ~(0x3 << SEL3_BIT_LSB_IDX);                   // TOUT3  --> TOM0 channel 11
417 }
418
```

GTM 레지스터 설정 – CLC

1. GTM 레지스터 영역의 주소 찾기
   - 시작 주소 (Base address)
   - **= 0xF0100000**

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.232

(I2C0)	$F00D\ 00FF_H$	byte		
Reserved	$F00D\ 0100_H$ - $F00F\ FFFF_H$	–	SPBBE	SPBBE
Global Timer Module (GTM)	$F010\ 0000_H$ - $F019\ FFFF_H$	640 Kbyte	access	access
Reserved	$F01A\ 0000_H$ - $F7FF\ FFFF_H$	–	SPBBE	SPBBE
Reserved	$F800\ 0000$		SPIRE	SPIRE

2. 사용할 레지스터의 주소 찾기
   - **CLC** 의 Offset Address = **0x9FD00**

→ CLC 레지스터 주소 = 0xF0100000 + 0x9FD00 = **0xF019FD00**

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 25-749
TIM0INSEL	TIM0 Input Select Register	$9FD10_H$	U, SV	U, SV, P	Application	Page 25-773
TIM1INSEL	TIM1 Input Select Register	$9FD14_H$	U, SV	U, SV, P	Application	Page 25-77

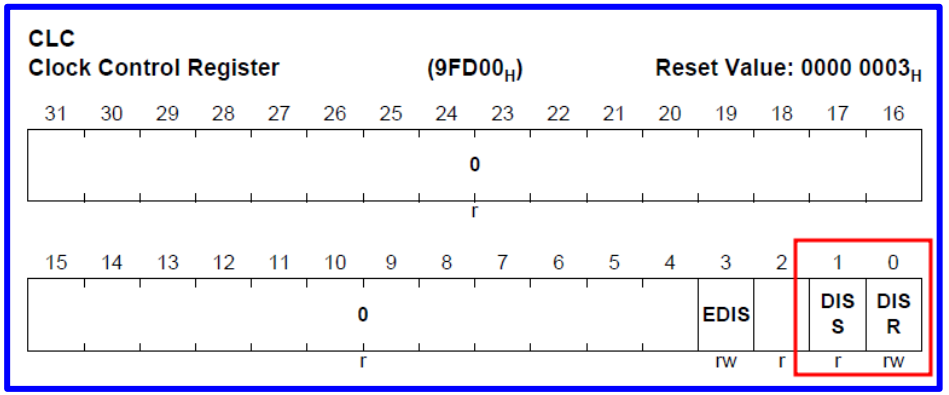
Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.3469

GTM 레지스터 설정 – CLC

:GTM 모듈 사용 설정

3. 레지스터 write 값 결정
   - GTM 모듈을 enable 하기 위해 **DISR 영역에 0x1 write**
   - GTM 모듈을 enable 된 것을 확인하기 위해 **DISS 영역의 값이 "0"인지 확인 (만약 enable 되지 않았다면 "1"의 값 유지)**

CLC 레지스터 @ 0xF019FD00



Field	Bits	Type	Description
DISR	0	rw	**Module Disable Request Bit** Used for enable/disable control of the module. 0_B Module disable is not requested. 1_B Module disable is requested.
DISS	1	rh	**Module Disable Status Bit** Bit indicates the current status of the module. 0_B Module is enabled. 1_B Module is disabled.
EDIS	3	rw	**Sleep Mode Enable Control**

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.3476

Lab3: GTM 레지스터 설정 – CLC
:GTM 모듈 사용 설정

```
372
373 void initGTM(void)
374 {
375     // Password Access to unlock SCU_WDTSCON0
376     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
377     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0);    // wait until unlocked
378
379     // Modify Access to clear ENDINIT
380     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) & ~(1 << ENDINIT_BIT_LSB_IDX);
381     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0);    // wait until locked
382
383     GTM_CLC.U &= ~(1 << DISR_BIT_LSB_IDX);    // enable GTM
384
385     // Password Access to unlock SCU_WDTSCON0
386     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
387     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0);    // wait until unlocked
388
389     // Modify Access to set ENDINIT
390     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
391     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0);
392
393     while((GTM_CLC.U & (1 << DISS_BIT_LSB_IDX)) != 0); // wait until GTM module enabled
394
395
396     // GTM clock configuration
397     GTM_CMU_FXCLK_CTRL.U &= ~(0xF << FXCLK_SEL_BIT_LSB_IDX);        // input clock of CMU_FXCLK --> CMU_GCLK_EN
398     GTM_CMU_CLK_EN.U |= 0x2 << EN_FXCLK_BIT_LSB_IDX;                // enable all CMU_FXCLK
399
400
401     // set GTM TOM0 channel 11 - Buzzer
402     GTM_TOM0_TGC1_GLB_CTRL.U |= 0x2 << UPEN_CTRL3_BIT_LSB_IDX;      // TOM0 channel 11 enable
403     GTM_TOM0_TGC1_ENDIS_CTRL.U |= 0x2 << ENDIS_CTRL3_BIT_LSB_IDX;   // enable channel 11 on update trigger
404     GTM_TOM0_TGC1_OUTEN_CTRL.U |= 0x2 << OUTEN_CTRL3_BIT_LSB_IDX;   // enable channel 11 output on update trigger
405
406
407     // TOM0 CH11
408     GTM_TOM0_CH11_CTRL.U |= 0x1 << SL_BIT_LSB_IDX;                  // high signal level for duty cycle
409     GTM_TOM0_CH11_CTRL.U |= 0x1 << CLK_SRC_SR_BIT_LSB_IDX;          // clock source --> CMU_FXCLK(1) = 6250 kHz
410     GTM_TOM0_CH11_CTRL.U &= ~(0x0 << OSM_BIT_LSB_IDX);              // continuous mode
411     GTM_TOM0_CH11_CTRL.U &= ~(0x0 << TRIGOUT_BIT_LSB_IDX);
412     GTM_TOM0_CH11_SR0.U = 12500 - 1;                                // PWM freq. = 6250 kHz / 12500 = 500 Hz (temp)
413     GTM_TOM0_CH11_SR1.U = 6250 - 1;                                 // duty cycle = 6250 / 12500 = 50 % (temp)
414
415     // TOUT pin selection
416     GTM_TOUTSEL0.U &= ~(0x3 << SEL3_BIT_LSB_IDX);                   // TOUT3  --> TOM0 channel 11
417 }
418
```

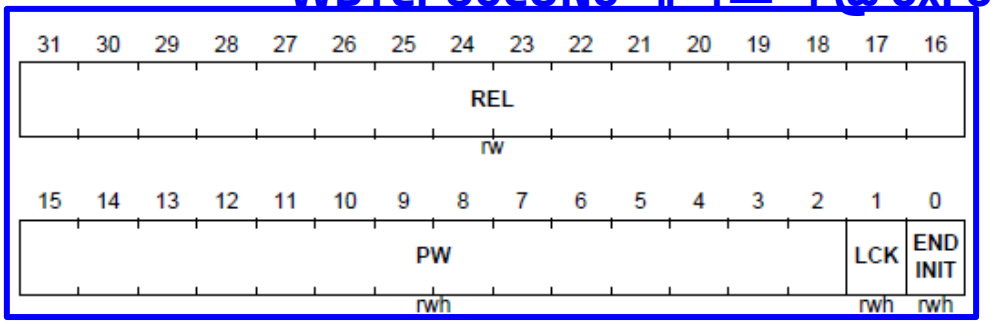
SCU 레지스터 설정 – WDTCPU0CON0

:GTM 모듈 사용 설정 후, 보호 레지스터 잠금 설정

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

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

WDTCPU0CON0 레지스터 @ 0xF0036100



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

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

PW	[15:2]	rwh	**User-Definable Password Field for Access to WDTxCON0** This bit field is written with an initial password value during a Modify Access. ~~A read from this bitfield returns this initial password,~~ but bits [7:2] are inverted (toggled) to ensure that a ~~simple read/write is not sufficient to service the WDT.~~

단, PW 영역의 일부 PW[7:2] 는 반전해서 읽어야 함

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

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

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

4) 결정한 32bit 값을 WDTCPU0CON0 레지스터에 한 번에 write

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

Lock 상태가 설정되면 LCK bit 가 1로 읽힘

5. Modify Access 를 통해 CPU0 ENDINIT을 해제하면 레지스터 수정 후 반드시 CPU0 ENDINIT을 재설정해줘야 함

Lab4: SCU 레지스터 설정 – WDTCPU0CON0

:GTM 모듈 사용 설정 후, 보호 레지스터 잠금 설정

```
372
373 void initGTM(void)
374 {
375     // Password Access to unlock SCU_WDTSCON0
376     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
377     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0);    // wait until unlocked
378
379     // Modify Access to clear ENDINIT
380     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) & ~(1 << ENDINIT_BIT_LSB_IDX);
381     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0);    // wait until locked
382
383     GTM_CLC.U &= ~(1 << DISR_BIT_LSB_IDX);    // enable GTM
384
385     // Password Access to unlock SCU_WDTSCON0
386     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
387     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0);    // wait until unlocked
388
389     // Modify Access to set ENDINIT
390     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
391     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0);
392
393     while((GTM_CLC.U & (1 << DISS_BIT_LSB_IDX)) != 0); // wait until GTM module enabled
394
395
396     // GTM clock configuration
397     GTM_CMU_FXCLK_CTRL.U &= ~(0xF << FXCLK_SEL_BIT_LSB_IDX);        // input clock of CMU_FXCLK --> CMU_GCLK_EN
398     GTM_CMU_CLK_EN.U |= 0x2 << EN_FXCLK_BIT_LSB_IDX;                // enable all CMU_FXCLK
399
400
401     // set GTM TOM0 channel 11 - Buzzer
402     GTM_TOM0_TGC1_GLB_CTRL.U |= 0x2 << UPEN_CTRL3_BIT_LSB_IDX;      // TOM0 channel 11 enable
403     GTM_TOM0_TGC1_ENDIS_CTRL.U |= 0x2 << ENDIS_CTRL3_BIT_LSB_IDX;   // enable channel 11 on update trigger
404     GTM_TOM0_TGC1_OUTEN_CTRL.U |= 0x2 << OUTEN_CTRL3_BIT_LSB_IDX;   // enable channel 11 output on update trigger
405
406
407     // TOM0 CH11
408     GTM_TOM0_CH11_CTRL.U |= 0x1 << SL_BIT_LSB_IDX;                  // high signal level for duty cycle
409     GTM_TOM0_CH11_CTRL.U |= 0x1 << CLK_SRC_SR_BIT_LSB_IDX;          // clock source --> CMU_FXCLK(1) = 6250 kHz
410     GTM_TOM0_CH11_CTRL.U &= ~(0x0 << OSM_BIT_LSB_IDX);              // continuous mode
411     GTM_TOM0_CH11_CTRL.U &= ~(0x0 << TRIGOUT_BIT_LSB_IDX);
412     GTM_TOM0_CH11_SR0.U = 12500 - 1;                                // PWM freq. = 6250 kHz / 12500 = 500 Hz (temp)
413     GTM_TOM0_CH11_SR1.U = 6250 - 1;                                 // duty cycle = 6250 / 12500 = 50 % (temp)
414
415     // TOUT pin selection
416     GTM_TOUTSEL0.U &= ~(0x3 << SEL3_BIT_LSB_IDX);                   // TOUT3  --> TOM0 channel 11
417 }
418
```

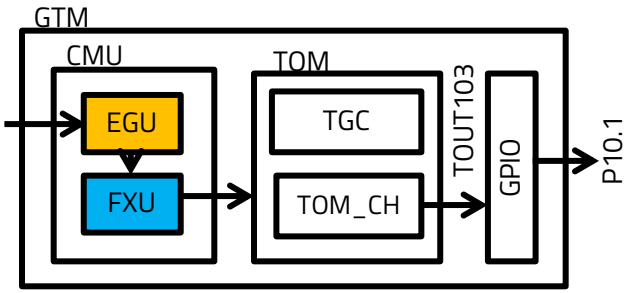
PWM 신호 생성 flow

:CMU 내부에 FXU 에서 TOM이 사용할 clock 생성

- GTM – TOM 에서 사용되는 clock 신호의 생성은 **CMU (Clock Management Unit)** 가 담당

- TOM은 CMU 에서 생성되는 clock 신호 중, **FXU (Fixed Clock Generation Unit)** 에서 생성되는 **clock** 들을 사용함

→ FXU 에 입력되는 clock 신호, FXU 에서 TOM으로 공급되는 clock 신호에 대한 설정 필요



Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.2829

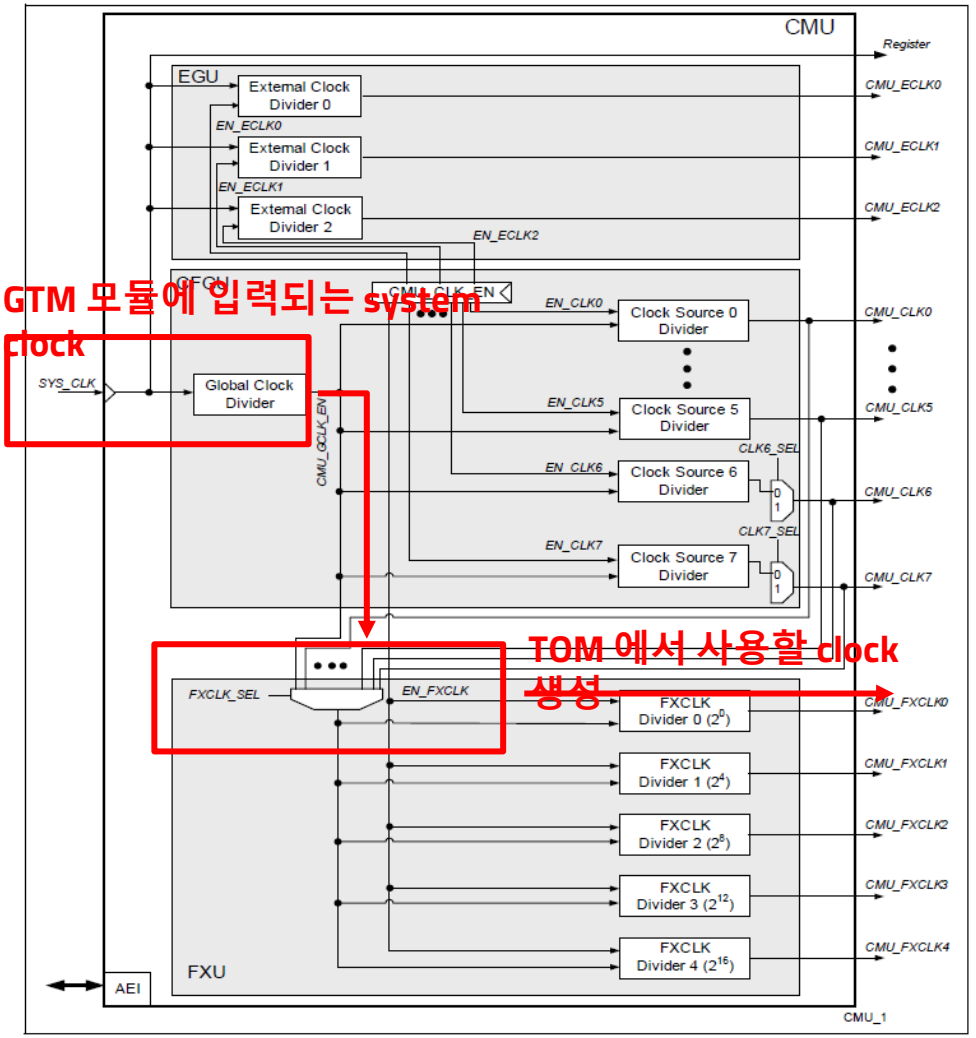


Figure 25-17 CMU Block Diagram

GTM 사용을 위한 레지스터 설정
:FXU 에서 GTM 내부 TOM이 사용할 clock 생성

- GTM 모듈에서 TOM 으로 clock 을 전달해주는 역할은 CMU 가 담당하며, **TOM은 FXU에서 생성하는 FXCLK를 사용함**

→ **FXU가 사용할 Clock, FXCLK의 주파수, 번호 등의 결정 필요**

- GTM 레지스터 항목에서
  - **CMU_CLK_EN** 레지스터 설정 필요
  - **CMU_FXCLK_CTRL** 레지스터 설정 필요

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.2829

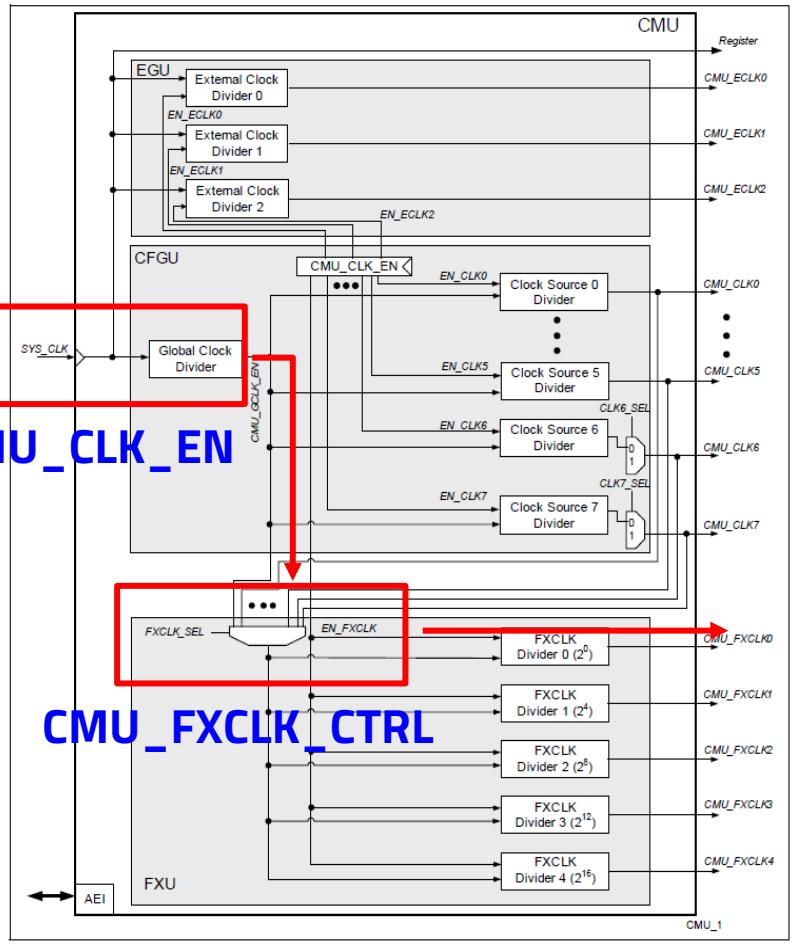


Figure 25-17 CMU Block Diagram

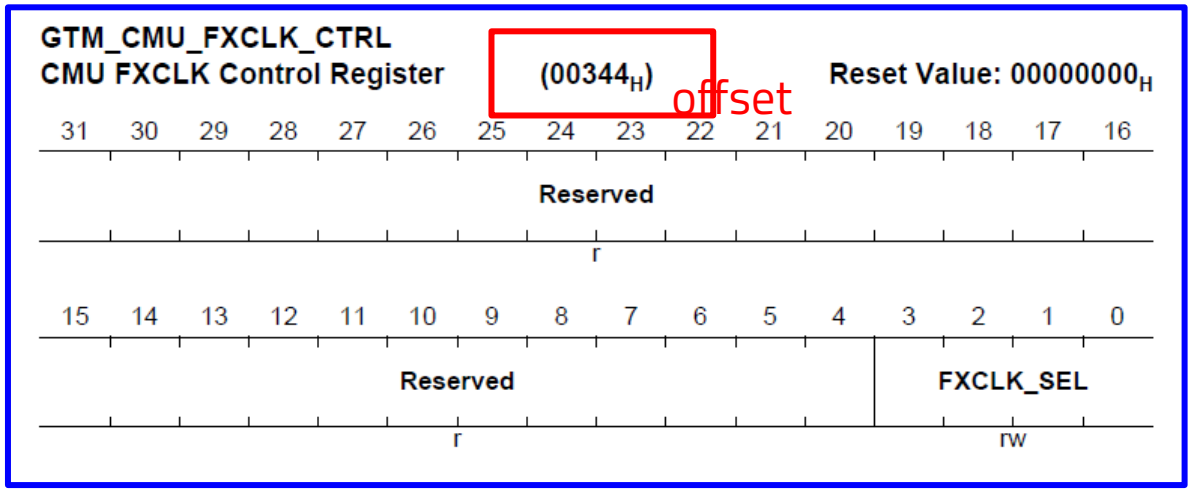
GTM 레지스터 설정 – CMU_FXCLK_CTRL

1. GTM 레지스터 영역의 주소 찾기
   - 시작 주소 (Base address) **= 0xF0100000**

2. 사용할 레지스터의 주소 찾기
   - **CMU_FXCLK_CTRL** 의 Offset Address = **0x344**

   → CMU_FXCLK_CTRL 레지스터 주소 = 0xF0100000 + 0x344 = **0xF0100344**

CMU_FXCLK_CTRL 레지스터 @ 0xF0100344



Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.2844

GTM 레지스터 설정 – CMU_FXCLK_CTRL

:FXU에서 생성할 clock 에 대한 설정

3. 레지스터 write 값 결정
   - CMU 내부의 FXU 에 대한 clock으로 GTM 모듈의 입력 clock인 **CMU_GCLK_EN** 을 사용하기 위해 **FXCLK_SEL 영역에 0x0 write**

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.2829

CMU_FXCLK_CTRL 레지스터 @ 0xF0100344

GTM_CMU_FXCLK_CTRL
CMU FXCLK Control Register (00344$_H$) **Reset Value: 00000000$_H$**

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

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

Field	Bits	Type	Description
FXCLK_SEL	[3:0]	rw	**Input clock selection for EN_FXCLK line** 0000_BCMU_GCLK_EN selected **100MHz** 0001_BCMU_CLK0 selected 0010_BCMU_CLK1 selected 0011_BCMU_CLK2 selected 0100_BCMU_CLK3 selected 0101_BCMU_CLK4 selected 0110_BCMU_CLK5 selected 0111_BCMU_CLK6 selected 1000_BCMU_CLK7 selected *Note: This value can only be written, when the CMU_FXCLK generation is disabled. See bits 23…22 in register CMU_CLK_EN.* *Note: Other values for FXCLK_SEL are reserved and should not be used. but they behave like*

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.2844

GTM 모듈에 입력되는 system clock을 FXU로 입력

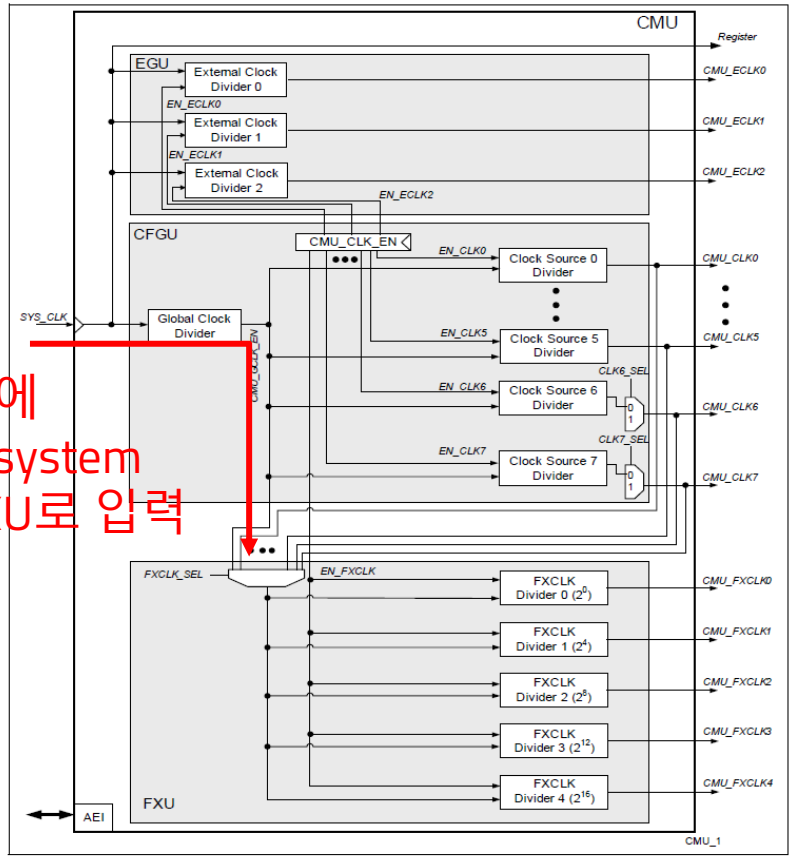


Figure 25-17 CMU Block Diagram

Lab5: GTM 레지스터 설정 – CMU_FXCLK_CTRL
:FXU에서 생성할 clock 에 대한 설정

```
372
373 void initGTM(void)
374 {
375     // Password Access to unlock SCU_WDTSCON0
376     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
377     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0);    // wait until unlocked
378
379     // Modify Access to clear ENDINIT
380     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) & ~(1 << ENDINIT_BIT_LSB_IDX);
381     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0);    // wait until locked
382
383     GTM_CLC.U &= ~(1 << DISR_BIT_LSB_IDX);    // enable GTM
384
385     // Password Access to unlock SCU_WDTSCON0
386     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
387     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0);    // wait until unlocked
388
389     // Modify Access to set ENDINIT
390     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
391     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0);
392
393     while((GTM_CLC.U & (1 << DISS_BIT_LSB_IDX)) != 0); // wait until GTM module enabled
394
395
396     // GTM clock configuration
397     GTM_CMU_FXCLK_CTRL.U &= ~(0xF << FXCLK_SEL_BIT_LSB_IDX);        // input clock of CMU_FXCLK --> CMU_GCLK_EN
398     GTM_CMU_CLK_EN.U |= 0x2 << EN_FXCLK_BIT_LSB_IDX;                // enable all CMU_FXCLK
399
400
401     // set GTM TOM0 channel 11 - Buzzer
402     GTM_TOM0_TGC1_GLB_CTRL.U |= 0x2 << UPEN_CTRL3_BIT_LSB_IDX;      // TOM0 channel 11 enable
403     GTM_TOM0_TGC1_ENDIS_CTRL.U |= 0x2 << ENDIS_CTRL3_BIT_LSB_IDX;   // enable channel 11 on update trigger
404     GTM_TOM0_TGC1_OUTEN_CTRL.U |= 0x2 << OUTEN_CTRL3_BIT_LSB_IDX;   // enable channel 11 output on update trigger
405
406
407     // TOM0 CH11
408     GTM_TOM0_CH11_CTRL.U |= 0x1 << SL_BIT_LSB_IDX;                  // high signal level for duty cycle
409     GTM_TOM0_CH11_CTRL.U |= 0x1 << CLK_SRC_SR_BIT_LSB_IDX;          // clock source --> CMU_FXCLK(1) = 6250 kHz
410     GTM_TOM0_CH11_CTRL.U &= ~(0x0 << OSM_BIT_LSB_IDX);              // continuous mode
411     GTM_TOM0_CH11_CTRL.U &= ~(0x0 << TRIGOUT_BIT_LSB_IDX);
412     GTM_TOM0_CH11_SR0.U = 12500 - 1;                                // PWM freq. = 6250 kHz / 12500 = 500 Hz (temp)
413     GTM_TOM0_CH11_SR1.U = 6250 - 1;                                 // duty cycle = 6250 / 12500 = 50 % (temp)
414
415     // TOUT pin selection
416     GTM_TOUTSEL0.U &= ~(0x3 << SEL3_BIT_LSB_IDX);                   // TOUT3  --> TOM0 channel 11
417 }
418
```

```
95 #define FXCLK_SEL_BIT_LSB_IDX       0
```

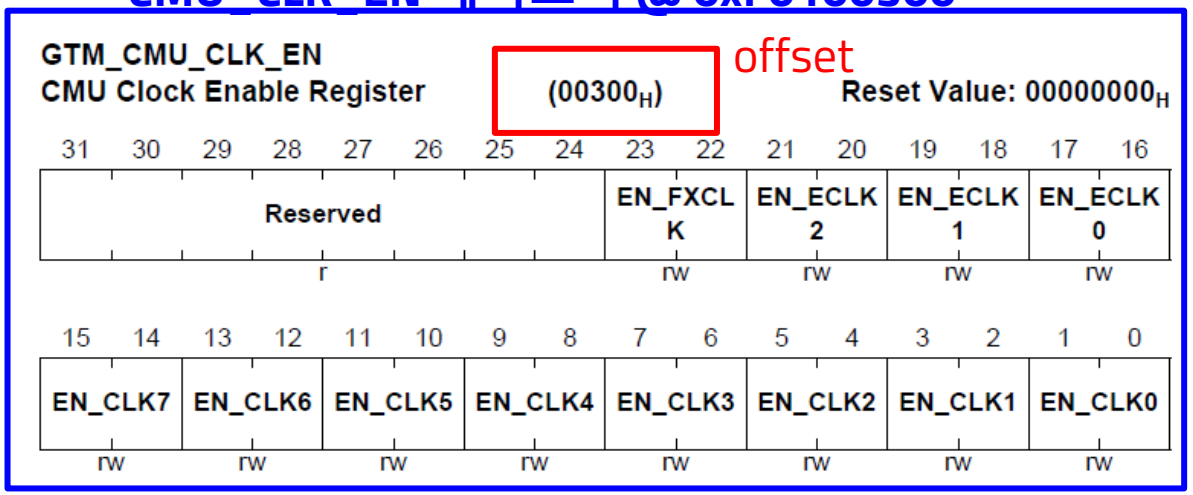
GTM 레지스터 설정 – CMU_CLK_EN

1. GTM 레지스터 영역의 주소 찾기
   - 시작 주소 (Base address) **= 0xF0100000**

2. 사용할 레지스터의 주소 찾기
   - **CMU_CLK_EN** 의 Offset Address = **0x300**

   → CMU_CLK_EN 레지스터 주소 = 0xF0100000 + 0x300 = **0xF0100300**

CMU_CLK_EN 레지스터 @ 0xF0100300



Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.2835

GTM 레지스터 설정 – CMU_CLK_EN

:FXU에서 생성할 clock 에 대한 설정

3. 레지스터 write 값 결정
   - TOM 은 FXU 에서 생성하는 FXCLK 를 사용
   - FXCLK를 사용하기 위해 **EN_FXCLK_SEL 영역에 0x2 write**

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.2829

CMU_CLK_EN 레지스터 @ 0xF0100300

GTM_CMU_CLK_EN
CMU Clock Enable Register (00300$_H$) **Reset Value: 00000000$_H$**

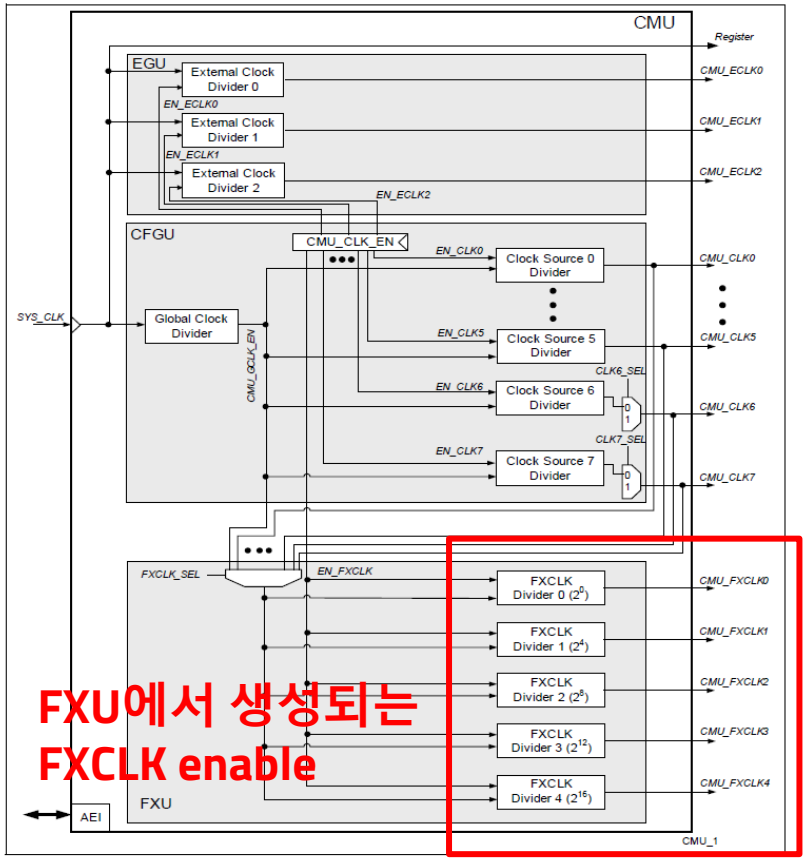
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved								EN_FXCLK		EN_ECLK2		EN_ECLK1		EN_ECLK0	
r								rw		rw		rw		rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EN_CLK7		EN_CLK6		EN_CLK5		EN_CLK4		EN_CLK3		EN_CLK2		EN_CLK1		EN_CLK0	
rw		rw		rw		rw		rw		rw		rw		rw	

2			see bits [1:0]
EN_FXCLK	[23:22]	rw	**Enable all CMU_FXCLK** see bits [1:0]
			Note: An enable to EN_FXCLK from disable state will be reset internal fixed clock counters.
Reserved	[31:24]	r	Reserved

00_B clock source is disabled (ignore write access)
01_B disable clock signal and reset internal states
10_B enable clock signal
11_B clock signal enabled (ignore write access)

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.2835



FXU에서 생성되는 FXCLK enable

Figure 25-17 CMU Block Diagram

Lab6: GTM 레지스터 설정 – CMU_CLK_EN

:FXU에서 생성할 clock 에 대한 설정

```
372
373 void initGTM(void)
374 {
375     // Password Access to unlock SCU_WDTSCON0
376     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
377     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0);    // wait until unlocked
378
379     // Modify Access to clear ENDINIT
380     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) & ~(1 << ENDINIT_BIT_LSB_IDX);
381     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0);    // wait until locked
382
383     GTM_CLC.U &= ~(1 << DISR_BIT_LSB_IDX);    // enable GTM
384
385     // Password Access to unlock SCU_WDTSCON0
386     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
387     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0);    // wait until unlocked
388
389     // Modify Access to set ENDINIT
390     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
391     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0);
392
393     while((GTM_CLC.U & (1 << DISS_BIT_LSB_IDX)) != 0); // wait until GTM module enabled
394
395
396     // GTM clock configuration
397     GTM_CMU_FXCLK_CTRL.U &= ~(0xF << FXCLK_SEL_BIT_LSB_IDX);        // input clock of CMU_FXCLK --> CMU_GCLK_EN
398     GTM_CMU_CLK_EN.U |= 0x2 << EN_FXCLK_BIT_LSB_IDX;                // enable all CMU_FXCLK
399
400
401     // set GTM TOM0 channel 11 - Buzzer
402     GTM_TOM0_TGC1_GLB_CTRL.U |= 0x2 << UPEN_CTRL3_BIT_LSB_IDX;      // TOM0 channel 11 enable
403     GTM_TOM0_TGC1_ENDIS_CTRL.U |= 0x2 << ENDIS_CTRL3_BIT_LSB_IDX;   // enable channel 11 on update trigger
404     GTM_TOM0_TGC1_OUTEN_CTRL.U |= 0x2 << OUTEN_CTRL3_BIT_LSB_IDX;   // enable channel 11 output on update trigger
405
406
407     // TOM0 CH11
408     GTM_TOM0_CH11_CTRL.U |= 0x1 << SL_BIT_LSB_IDX;                  // high signal level for duty cycle
409     GTM_TOM0_CH11_CTRL.U |= 0x1 << CLK_SRC_SR_BIT_LSB_IDX;          // clock source --> CMU_FXCLK(1) = 6250 kHz
410     GTM_TOM0_CH11_CTRL.U &= ~(0x0 << OSM_BIT_LSB_IDX);              // continuous mode
411     GTM_TOM0_CH11_CTRL.U &= ~(0x0 << TRIGOUT_BIT_LSB_IDX);
412     GTM_TOM0_CH11_SR0.U = 12500 - 1;                                // PWM freq. = 6250 kHz / 12500 = 500 Hz (temp)
413     GTM_TOM0_CH11_SR1.U = 6250 - 1;                                 // duty cycle = 6250 / 12500 = 50 % (temp)
414
415     // TOUT pin selection
416     GTM_TOUTSEL0.U &= ~(0x3 << SEL3_BIT_LSB_IDX);                   // TOUT3  --> TOM0 channel 11
417 }
418
```

```
94 #define EN_FXCLK_BIT_LSB_IDX        22
```

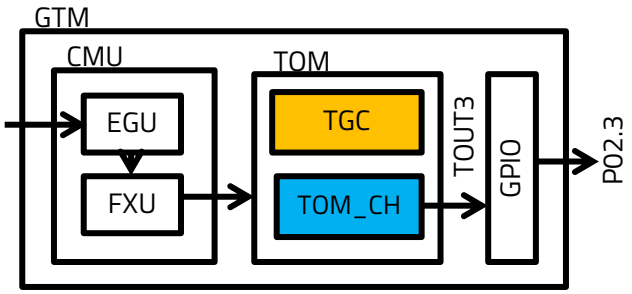
PWM 신호 생성 flow
:TOM에서 PWM 출력 신호 생성

- 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

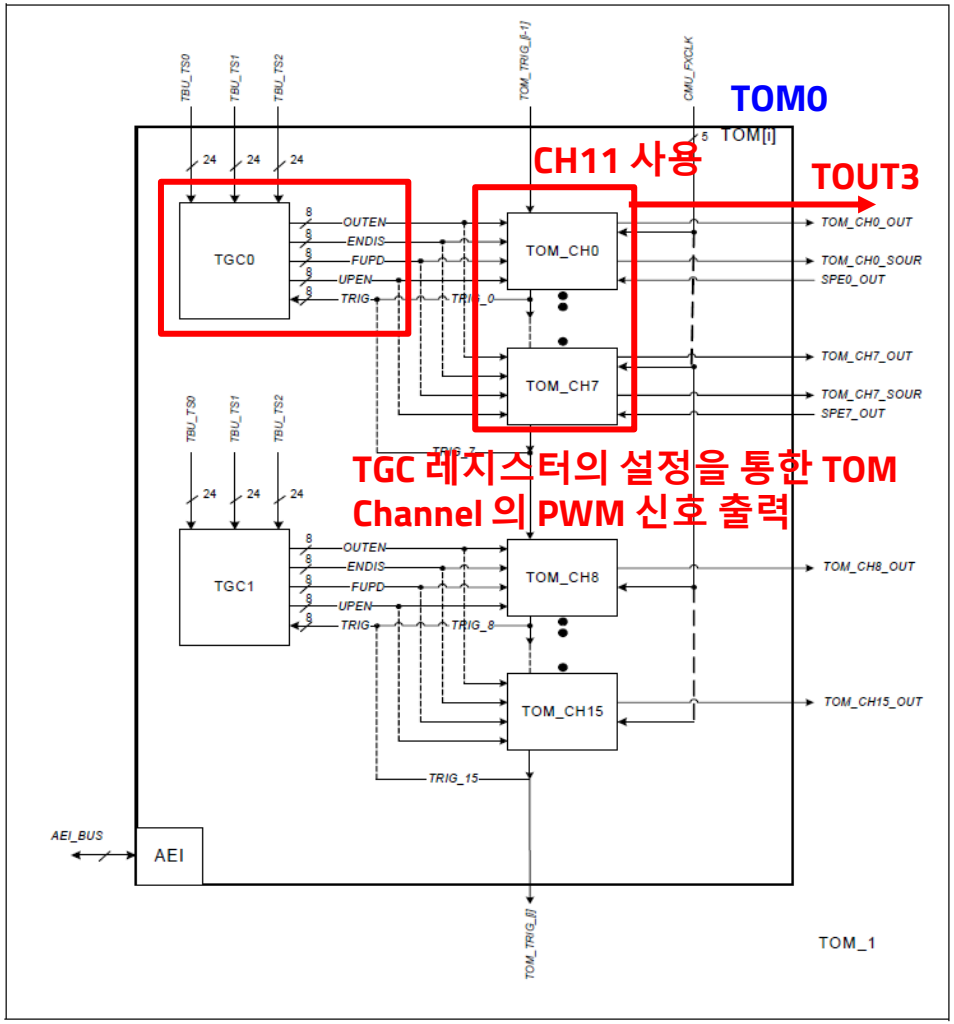


Figure 25-32 TOM Block diagram

GTM 사용을 위한 레지스터 설정
:GPIO 출력을 GTM 출력으로 사용하는 설정

- PWM 신호를 통해 밝기를 제어할 LED RED가 연결된 핀은 **P10.1**

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
P02.3	I	General-purpose input	P02_IN.P3	P02_IOCR0. PC3	$0XXXX_B$
		GTM input	TIN3		
		CAN node 2 input	RXDCAN2B		
		ERAY input	RXDB2		
		PSI5 input	PSIRX0B		
		DSADC input	DSCIN5B		
		MSC1 input	SDI11		
		CIF input	CIFD3		
		ASCLIN1 input	ARX1G		
	O	General-purpose output	P02_OUT.P3		$1X000_B$
		GTM output	TOUT3		$1X001_B$
		ASCLIN2 output	ASLSO2		$1X010_B$
		QSPI3 output	SLSO34		$1X011_B$
		DSADC output	DSCOUT5		$1X100_B$
		Reserved	–		$1X101_B$
		Reserved	–		$1X110_B$
		CCU60 output	COUT61		$1X111_B$

P02.3 핀의 GPIO 출력을 GTM 출력으로 사용하기 위해 →
포트의 I/O Control 레지스터 설정 필요

- GPIO P02 레지스터 항목에서
  - **IOCR0** 레지스터 설정 필요

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.1152

GPIO 레지스터 설정 – P02 Address 계산

: Port 02의 각 레지스터가 위치한 주소 확인

3. 사용할 특정 레지스터의 주소 찾기
   - **P02_IOCR0** Offset Address = 0x0010

   → P02_IOCR0 레지스터 주소

   → = 0xF003A200 + 0x0010 = **0xF003A210**

Table 13-4 Registers Overview

Register Short Name	Register Long Name	Offset Address	Access Mode Read	Access Mode Write	Reset	Desc. see
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

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

GPIO 레지스터 설정 – P02 I/O 모드

: Port 02의 핀 3을 GTM 출력 모드로 사용

- **P02.3** 을 **General Purpose Output (push-pull)가 아닌 GTM 모듈의 출력으로 사용하기 위해** P02_IOCR0 레지스터에 어떤 값을 써야 하는지 확인 → **0x11 write**
  (예전에 단순 GPIO출력으로 쓸 때는 0x10를 썼다)

P02_IOCR0 레지스터 @ 0xF003A210

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PC3					0			PC2					0		
rw					r			rw					r		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PC1					0			PC0					0		
rw					r			rw					r		

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

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.1080

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	Port I/O Control Select. Value
P02.3	I	General-purpose input	P02_IN.P3	P02_IOCR0. PC3	$0XXXX_B$
		GTM input	TIN3		
		CAN node 2 input	RXDCAN2B		
		ERAY input	RXDB2		
		PSI5 input	PSIRX0B		
		DSADC input	DSCIN5B		
		MSC1 input	SDI11		
		CIF input	CIFD3		
		ASCLIN1 input	ARX1G		
	O	~~General-purpose output~~	~~P02_OUT.P3~~		~~$1X000_B$~~
		GTM output	TOUT3		$1X001_B$
		~~ASCLIN2 output~~	~~ASLSO2~~		~~$1X010_B$~~
		QSPI3 output	SLSO34		$1X011_B$
		DSADC output	DSCOUT5		$1X100_B$
		Reserved	–		$1X101_B$
		Reserved	–		$1X110_B$
		CCU60 output	COUT61		$1X111_B$

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

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.1152

Lab7: P02 레지스터 설정 – IOCR0
:GPIO 출력 모드를 GTM 생성 출력으로 설정

initBuzzer() 작성

```
void initBuzzer(void)
{
    P02_IOCR0.B.PC3 = 0x11;                    // GTM output mode
}
```

GTM 사용을 위한 레지스터 설정
:GTM 모듈의 PWM 출력을 MCU 외부 핀으로 연결

- PWM 신호를 통해 밝기를 제어할 LED RED가 연결된 핀은 **P02.3**
- 핀 P02.3 에 **GPIO 대신 GTM 출력으로 연결 필요 → TOUT3**

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
P02.3	I	General-purpose input	P02_IN.P3	P02_IOCR0. PC3	$0XXXX_B$
		GTM input	TIN3		
		CAN node 2 input	RXDCAN2B		
		ERAY input	RXDB2		
		PSI5 input	PSIRX0B		
		DSADC input	DSCIN5B		
		MSC1 input	SDI11		
		CIF input	CIFD3		
		ASCLIN1 input	ARX1G		
	O	General-purpose output	P02_OUT.P3		$1X000_B$
		GTM output	TOUT3		$1X001_B$
		ASCLIN2 output	ASLSO2		$1X010_B$
		QSPI3 output	SLSO34		$1X011_B$
		DSADC output	DSCOUT5		$1X100_B$
		Reserved	–		$1X101_B$
		Reserved	–		$1X110_B$
		CCU60 output	COUT61		$1X111_B$

→ **GTM 출력으로 사용하기 위해 TOUT3 관련 레지스터 설정 필요**

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.1152

GTM 사용을 위한 레지스터 설정
:사용하는 GPIO 핀이 TOM의 어느 출력과 연결되는지 확인

- 앞서 핀 P02.3 에 연결되는 GTM의 출력은 TOUT3 인 것을 확인
- TOUT 출력은 GTM – TOM 에서 사용하는 TOM 번호와 채널이 정해져 있음

→ P02.3은 TOUT3

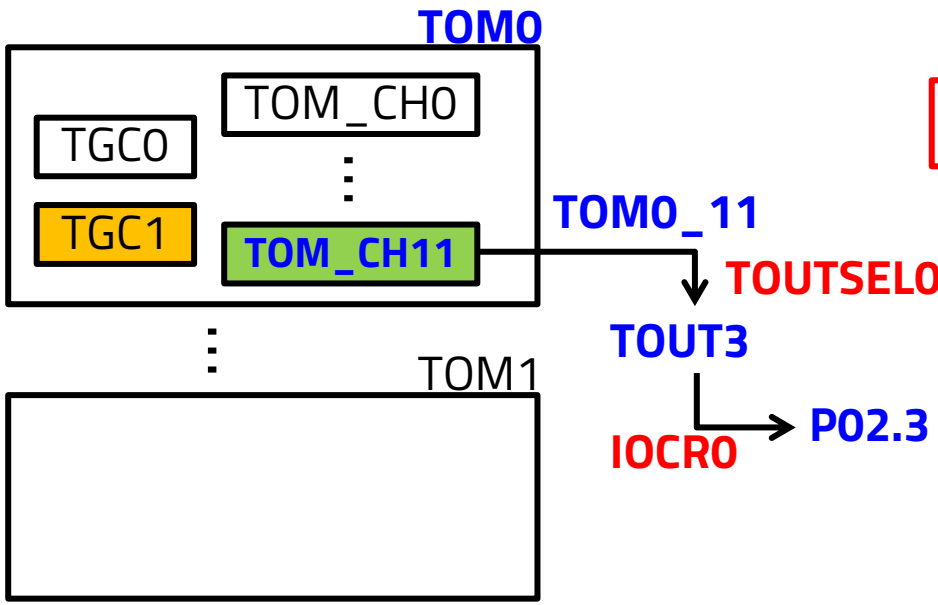


Table 25-67 GTM to Port Mapping for QFP-176

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	ATOM

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

GTM 레지스터 설정

: Timer Output Selection Register (TOUTSELx)

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 **TOUT3** (P02.31)은 **TOUTSEL0**에 포함되어 있음(**TOUT0~ TOUT15 안에 TOU3이 포함)**
   - **TOUTSEL0** 의 Offset Address = **0x9FD30**

   → TOUTSEL0 레지스터 주소 = 0xF0100000 + 0x9FD30 = **0xF019FD30**

TOUTSEL0 ⟶ TOUT00 ~~ TOUT15
TOUTSEL1 ⟶ TOUT16 ~~ TOUT31
TOUTSEL2 ⟶ TOUT32 ~~ TOUT47
TOUTSEL3 ⟶ TOUT48 ~~ TOUT63
TOUTSEL4 ⟶ TOUT64 ~~ TOUT79
TOUTSEL5 ⟶ TOUT80 ~~ TOUT95
TOUTSEL6 ⟶ TOUT96 ~~ TOUT103

TIM30INSEL	TIM3 Input Select Register	$9FD1C_H$	U, SV	U, SV, P	Application	Page 25-773
TOUTSEL0	Timer Output Select 0 Register	$9FD30_H$	U, SV	U, SV, P	Application	Page 25-814
TOUTSEL1	Timer Output Select 1 Register	$9FD34_H$	U, SV	U, SV, P	Application	Page 25-814
TOUTSEL2	Timer Output Select 2	$9FD38_H$	U, SV	U, SV	Application	Page 25-81

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.3469

GTM 레지스터 설정 – TOUTSEL0

:TOM에서 출력되는 PWM 신호를 MCU 핀에 연결 설정

3. 레지스터 write 값 결정
   - TOM0 채널11로부터 생성된 PWM 신호를 TOUT3 핀으로 출력하기 위한 설정
   - **TOUTSEL0은 TOUT0부터 TOUT15까지 제어함.**
   - **TOUT3 핀은 3번째임 (0번부터)**
     - **Output Timer Mapped 에서 A 항목을 사용하도록 설정하기 위해 SEL3 영역에 0x0 write**

TOUTSEL0 레지스터 @ 0xF019FD30

TOUTSELn (n = 0-14)
Timer Output Select Register ($9FD30_H+n*4_H$) Reset Value: $0000\ 0000_H$

31 30	29 28	27 26	25 24	23 22	21 20	19 18	17 16
SEL15	SEL14	SEL13	SEL12	SEL11	SEL10	SEL9	SEL8
rw	rw	rw	rw	rw	rw	rw	rw

15 14	13 12	11 10	9 8	7 6	5 4	3 2	1 0
SEL7	SEL6	SEL5	SEL4	SEL3	SEL2	SEL1	SEL0
rw	rw	rw	rw	rw	rw	rw	rw

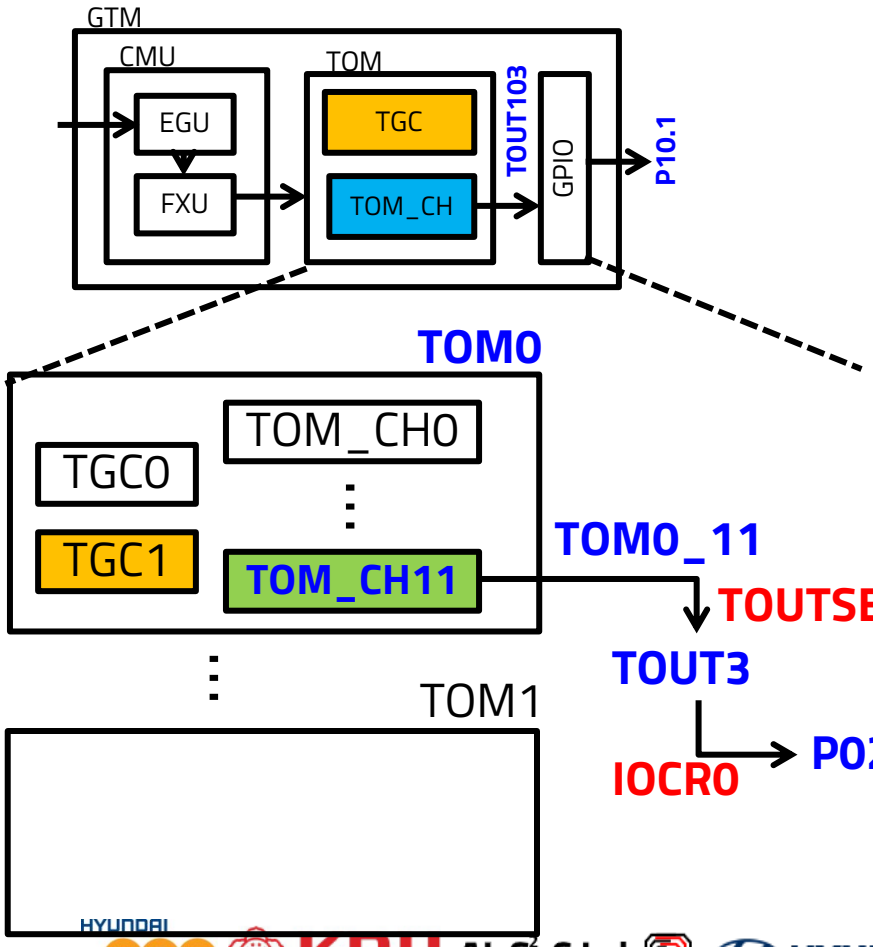
Field	Bits	Type	Description
SELx (x = 0-15)	[x*2+1: x*2]	rw	**TOUT(n*16+x) Output Selection** This bit defines which timer out is connected as TOUT(n*16+x). The mapping for each pin is defined by Table 25-67Table 25-68 00_B Timer A form Table 25-67Table 25-68 is connected as TOUT(n*16+x) to the ports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

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

GTM 사용을 위한 레지스터 설정

: P02.3은 TOUT3에 연결되어 있으며, TOM0 모듈의 채널 11을 사용하도록 설정됨

→ P02.3 (TOUT3)은 **TOM0** 의 채널 **11** 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	ATOM

Output Timer Mapped 의 **A** 항목 사용하도록 설정 시

→ **0번째 TOM (TOM0)의 채널 11 사용**

Lab8: GTM 레지스터 설정 – TOUTSEL0

:TOM에서 출력되는 (TOM0모듈의 채널11) PWM 신호를 MCU 핀에 연결 설정

```
372
373 void initGTM(void)
374 {
375     // Password Access to unlock SCU_WDTSCON0
376     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
377     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0);    // wait until unlocked
378
379     // Modify Access to clear ENDINIT
380     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) & ~(1 << ENDINIT_BIT_LSB_IDX);
381     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0);    // wait until locked
382
383     GTM_CLC.U &= ~(1 << DISR_BIT_LSB_IDX);    // enable GTM
384
385     // Password Access to unlock SCU_WDTSCON0
386     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
387     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0);    // wait until unlocked
388
389     // Modify Access to set ENDINIT
390     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
391     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0);
392
393     while((GTM_CLC.U & (1 << DISS_BIT_LSB_IDX)) != 0); // wait until GTM module enabled
394
395
396     // GTM clock configuration
397     GTM_CMU_FXCLK_CTRL.U &= ~(0xF << FXCLK_SEL_BIT_LSB_IDX);        // input clock of CMU_FXCLK --> CMU_GCLK_EN
398     GTM_CMU_CLK_EN.U |= 0x2 << EN_FXCLK_BIT_LSB_IDX;                // enable all CMU_FXCLK
399
400
401     // set GTM TOM0 channel 11 - Buzzer
402     GTM_TOM0_TGC1_GLB_CTRL.U |= 0x2 << UPEN_CTRL3_BIT_LSB_IDX;      // TOM0 channel 11 enable
403     GTM_TOM0_TGC1_ENDIS_CTRL.U |= 0x2 << ENDIS_CTRL3_BIT_LSB_IDX;   // enable channel 11 on update trigger
404     GTM_TOM0_TGC1_OUTEN_CTRL.U |= 0x2 << OUTEN_CTRL3_BIT_LSB_IDX;   // enable channel 11 output on update trigger
405
406
407     // TOM0 CH11
408     GTM_TOM0_CH11_CTRL.U |= 0x1 << SL_BIT_LSB_IDX;                  // high signal level for duty cycle
409     GTM_TOM0_CH11_CTRL.U |= 0x1 << CLK_SRC_SR_BIT_LSB_IDX;          // clock source --> CMU_FXCLK(1) = 6250 kHz
410     GTM_TOM0_CH11_CTRL.U &= ~(0x0 << OSM_BIT_LSB_IDX);              // continuous mode
411     GTM_TOM0_CH11_CTRL.U &= ~(0x0 << TRIGOUT_BIT_LSB_IDX);
412     GTM_TOM0_CH11_SR0.U = 12500 - 1;                                // PWM freq. = 6250 kHz / 12500 = 500 Hz (temp)
413     GTM_TOM0_CH11_SR1.U = 6250 - 1;                                 // duty cycle = 6250 / 12500 = 50 % (temp)
414
415     // TOUT pin selection
416     GTM_TOUTSEL0.U &= ~(0x3 << SEL3_BIT_LSB_IDX);                   // TOUT3  --> TOM0 channel 11
417 }
418
```

```
93  #define SEL3_BIT_LSB_IDX        6
```

GTM 사용을 위한 레지스터 설정
:TOM에서 출력으로 사용할 채널 설정

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

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

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

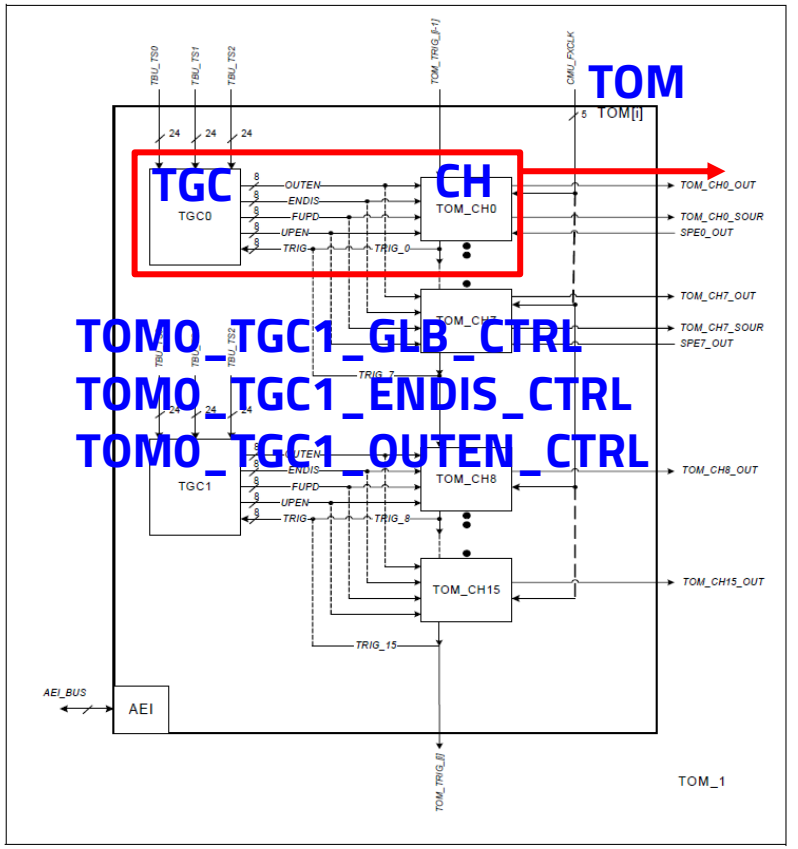


Figure 25-32 TOM Block diagram

- 본 실습에서는 **TOM0의 채널 11**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

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) = **0x8230**

   → TOM0_TGC1_GLB_CTRL 레지스터 주소 = 0xF0100000 + 0x8230 = **0xF0108230**

TOM0_TGC1_GLB_CTRL 레지스터 @ 0xF0108230

GTM_TOMi_TGC1_GLB_CTRL (i=0..2)
TOMi TGC1 Global Control Register (08230_H+i*800_H) offset
Reset Value: 00000000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UPEN_CTRL7		UPEN_CTRL6		UPEN_CTRL5		UPEN_CTRL4		UPEN_CTRL3		UPEN_CTRL2		UPEN_CTRL1		UPEN_CTRL0	
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_CH7	RST_CH6	RST_CH5	RST_CH4	RST_CH3	RST_CH2	RST_CH1	RST_CH0	Reserved							HOST_TRIG
w	w	w	w	w	w	w	w	r							w

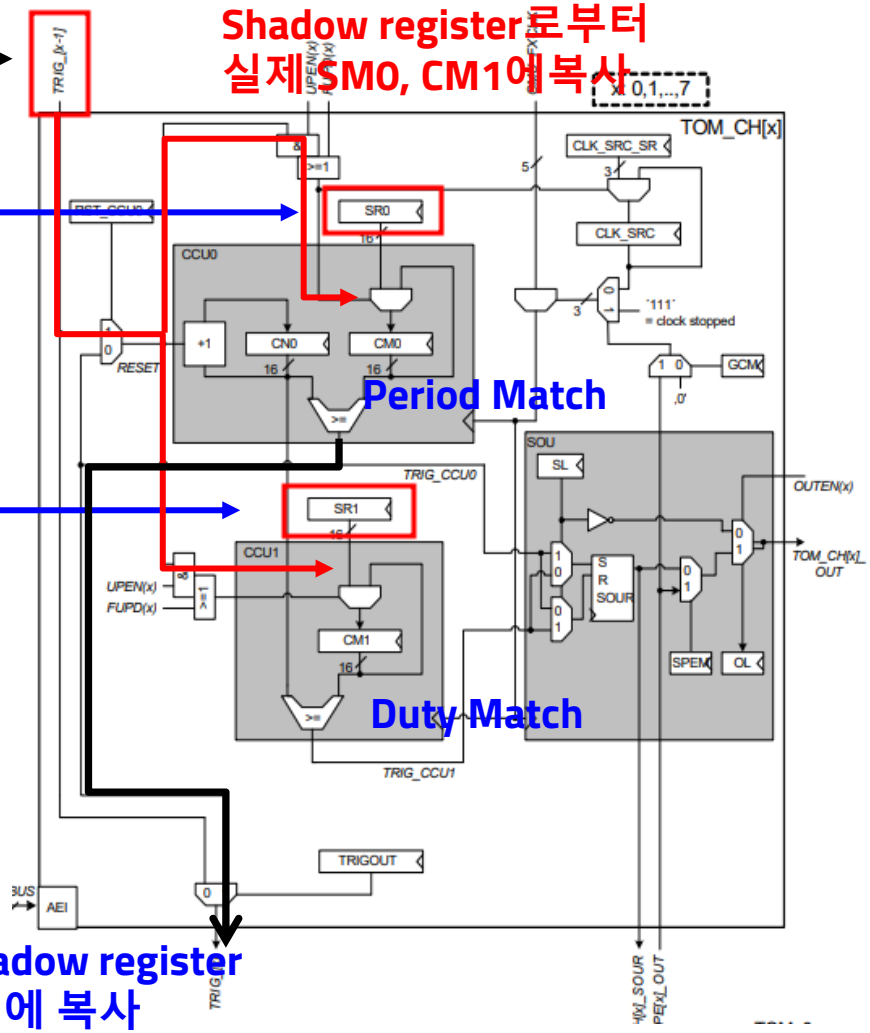
Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.2937

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 11에서 이러한 update가 가능하도록 설정 필요**

Period Match 신호에서 Trig (즉 shadow register에 셋팅된 SR0, SR1값을 CM0, CM1에 복사

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

GTM 레지스터 설정 – TOM0_TGC1_GLB_CTRL
:TOM 에서 사용할 채널 설정

3. 레지스터 write 값 결정
   - TOM0의 채널 11이 동작하기 위해서는 해당 채널에서 PWM 설정 값들이 update 되어야 함
   - Update가 가능하도록 설정하기 위해 **UPEN_CTRL3 영역에 0x2 write**

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

TOM0_TGC1_GLB_CTRL 레지스터 @ 0xF0108230

GTM_TOMi_TGC1_GLB_CTRL (i=0-2)
TOMi TGC1 Global Control Register(08230_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_CTRL7		UPEN_CTRL6		UPEN_CTRL5		UPEN_CTRL4		UPEN_CTRL3		UPEN_CTRL2		UPEN_CTRL1		UPEN_CTRL0	
rw		rw		rw		rw		rw		rw		rw		rw	

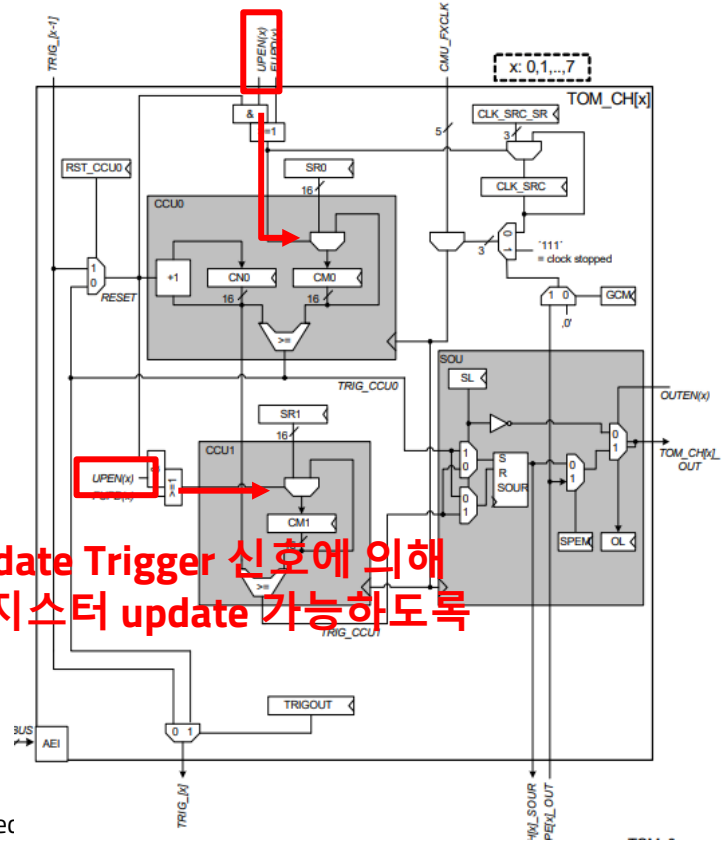
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RST_CH7	RST_CH6	RST_CH5	RST_CH4	RST_CH3	RST_CH2	RST_CH1	RST_CH0	Reserved							HOST_TRIG
w	w	w	w	w	w	w	w	r							w

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

UPEN_CTRL3	[23:22]	rw	TOM channel 3 enable update of register CM0, CM1 and CLK_SRC See bits 17:16

TGC1이 제어하는 채널8~15
채널11은 8부터 시작하면 **3번째**

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



Update Trigger 신호에 의해 레지스터 update 가능하도록

Lab9: GTM 레지스터 설정 – TOM0_TGC1_GLB_CTRL
:TOM 에서 사용할 채널11의 update enable 설정

```
372
373 void initGTM(void)
374 {
375     // Password Access to unlock SCU_WDTSCON0
376     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
377     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0);    // wait until unlocked
378
379     // Modify Access to clear ENDINIT
380     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) & ~(1 << ENDINIT_BIT_LSB_IDX);
381     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0);    // wait until locked
382
383     GTM_CLC.U &= ~(1 << DISR_BIT_LSB_IDX);    // enable GTM
384
385     // Password Access to unlock SCU_WDTSCON0
386     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
387     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0);    // wait until unlocked
388
389     // Modify Access to set ENDINIT
390     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
391     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0);
392
393     while((GTM_CLC.U & (1 << DISS_BIT_LSB_IDX)) != 0); // wait until GTM module enabled
394
395
396     // GTM clock configuration
397     GTM_CMU_FXCLK_CTRL.U &= ~(0xF << FXCLK_SEL_BIT_LSB_IDX);        // input clock of CMU_FXCLK --> CMU_GCLK_EN
398     GTM_CMU_CLK_EN.U |= 0x2 << EN_FXCLK_BIT_LSB_IDX;                // enable all CMU_FXCLK
399
400
401     // set GTM TOM0 channel 11 - Buzzer
402     GTM_TOM0_TGC1_GLB_CTRL.U |= 0x2 << UPEN_CTRL3_BIT_LSB_IDX;      // TOM0 channel 11 enable
403     GTM_TOM0_TGC1_ENDIS_CTRL.U |= 0x2 << ENDIS_CTRL3_BIT_LSB_IDX;   // enable channel 11 on update trigger
404     GTM_TOM0_TGC1_OUTEN_CTRL.U |= 0x2 << OUTEN_CTRL3_BIT_LSB_IDX;   // enable channel 11 output on update trigger
405
406
407     // TOM0 CH11
408     GTM_TOM0_CH11_CTRL.U |= 0x1 << SL_BIT_LSB_IDX;                  // high signal level for duty cycle
409     GTM_TOM0_CH11_CTRL.U |= 0x1 << CLK_SRC_SR_BIT_LSB_IDX;          // clock source --> CMU_FXCLK(1) = 6250 kHz
410     GTM_TOM0_CH11_CTRL.U &= ~(0x0 << OSM_BIT_LSB_IDX);              // continuous mode
411     GTM_TOM0_CH11_CTRL.U &= ~(0x0 << TRIGOUT_BIT_LSB_IDX);
412     GTM_TOM0_CH11_SR0.U = 12500 - 1;                                // PWM freq. = 6250 kHz / 12500 = 500 Hz (temp)
413     GTM_TOM0_CH11_SR1.U = 6250 - 1;                                 // duty cycle = 6250 / 12500 = 50 % (temp)
414
415     // TOUT pin selection
416     GTM_TOUTSEL0.U &= ~(0x3 << SEL3_BIT_LSB_IDX);                   // TOUT3  --> TOM0 channel 11
417 }
418
```

```
98 #define UPEN_CTRL3_BIT_LSB_IDX      22
```

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)

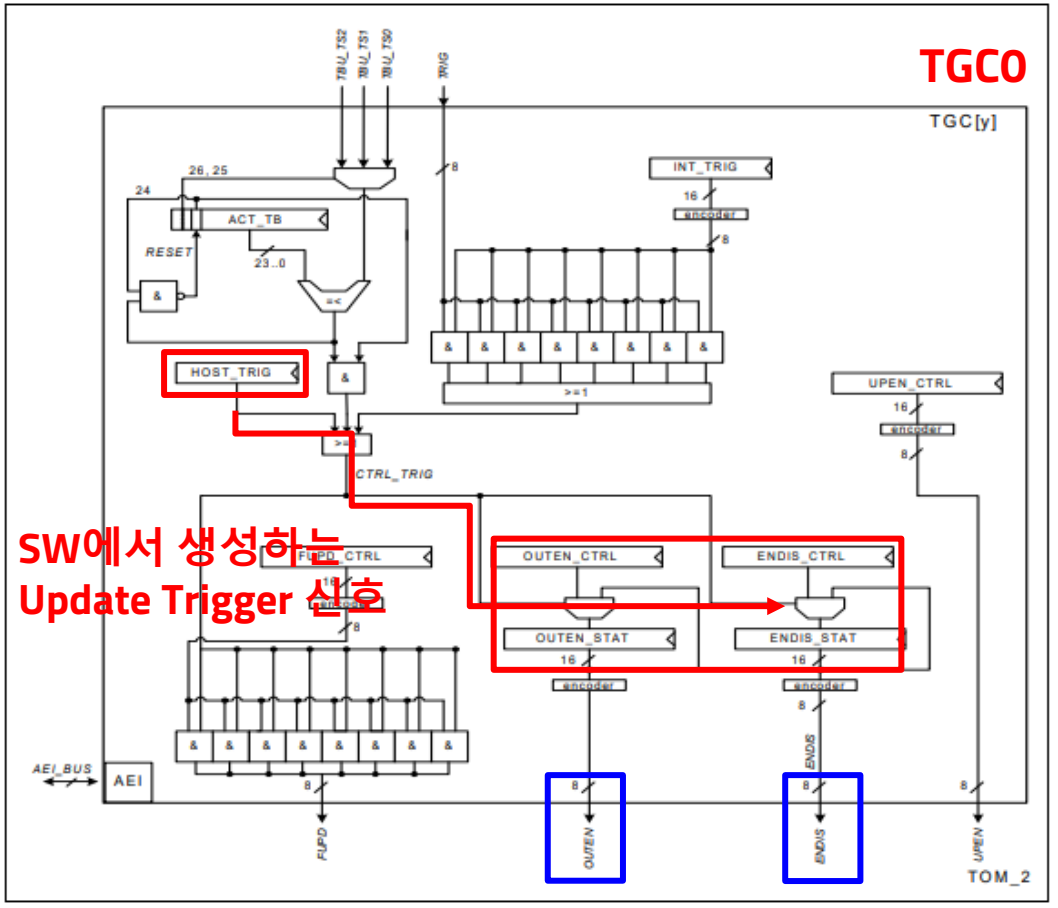


Figure 25-33 TOM Global channel control mechanism

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.

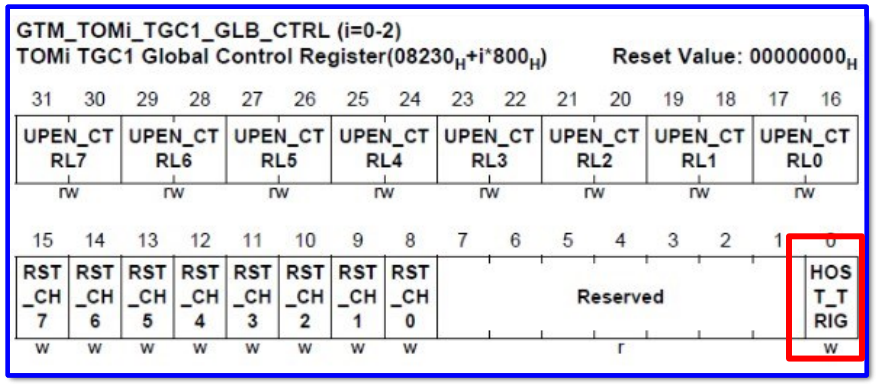
Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.2919

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.2921

GTM 레지스터 설정 – TOM0_TGC1_GLB_CTRL
:TOM 레지스터 설정이 적용되도록 update trigger event 발생

3. 레지스터 write 값 결정
   - TOM0의 채널 11에 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



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

Lab10: GTM 레지스터 설정 – TOM0_TGC1_GLB_CTRL

:TOM 에서 출력 생성 시작하도록 trigger event 발생

```
99  #define HOST_TRIG_BIT_LSB_IDX       0
```

```
151     initRGBLED();
152     //initVADC();
153     initBuzzer();
154     initGTM();
155     //initButton();
156
157     GTM_TOM0_TGC1_GLB_CTRL.U |= 0x1 << HOST_TRIG_BIT_LSB_IDX;       // trigger update request signal
158
159     // from 3 octave 도C ~ 4 octave 도C {도C, 레D, 미E, 파F, 솔G, 라A, 시B, 도C}
160     unsigned int duty[8] = {130, 146, 164, 174, 195, 220, 246, 262};
161
162     while(1)
163     {
164
```

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) = **0x8270**

   → TOM0_TGC1_ENDIS_CTRL 레지스터 주소 = 0xF0100000 + 0x8270 = **0xF0108270**

TOM0_TGC1_ENDIS_CTRL 레지스터 @ 0xF0108270

GTM_TOMi_TGC1_ENDIS_CTRL (i=0-2)
TOMi TGC1 Enable/Disable Control Register
(08270_H+i*800_H)

offset

Reset Value: 00000000_H

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ENDIS_CTRL7		ENDIS_CTRL6		ENDIS_CTRL5		ENDIS_CTRL4		ENDIS_CTRL3		ENDIS_CTRL2		ENDIS_CTRL1		ENDIS_CTRL0	
rw		rw		rw		rw		rw		rw		rw		rw	

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

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 11이 enable** 되도록 하기 위해 **ENDIS_CTRL3 영역에 0x2 write**

TOM0_TGC1_ENDIS_CTRL 레지스터 @ 0xF0108270

GTM_TOMi_TGC1_ENDIS_CTRL (i=0-2)
TOMi TGC1 Enable/Disable Control Register
(08270_H+i*800_H) Reset Value: 00000000_H

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ENDIS_CT RL7		ENDIS_CT RL6		ENDIS_CT RL5		ENDIS_CT RL4		ENDIS_CT RL3		ENDIS_CT RL2		ENDIS_CT RL1		ENDIS_CT RL0	
rw		rw		rw		rw		rw		rw		rw		rw	

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

TGC1이 제어하는 채널8~15
채널11은 8부터 시작하면 **3번째**

RL2			See bits 1:0
ENDIS_CT RL3	[7:6]	rw	(A)TOM channel 3 enable/disable update value See bits 1:0
ENDIS_CT	[9:8]	rw	(A)TOM channel 4 enable/disable update value

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

Lab11: GTM 레지스터 설정 – TOM0_TGC1_ENDIS_CTRL

:TOM에서 사용할 채널에 대한 설정

```
372
373 void initGTM(void)
374 {
375     // Password Access to unlock SCU_WDTSCON0
376     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
377     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0);    // wait until unlocked
378
379     // Modify Access to clear ENDINIT
380     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) & ~(1 << ENDINIT_BIT_LSB_IDX);
381     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0);    // wait until locked
382
383     GTM_CLC.U &= ~(1 << DISR_BIT_LSB_IDX);    // enable GTM
384
385     // Password Access to unlock SCU_WDTSCON0
386     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
387     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0);    // wait until unlocked
388
389     // Modify Access to set ENDINIT
390     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
391     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0);
392
393     while((GTM_CLC.U & (1 << DISS_BIT_LSB_IDX)) != 0); // wait until GTM module enabled
394
395
396     // GTM clock configuration
397     GTM_CMU_FXCLK_CTRL.U &= ~(0xF << FXCLK_SEL_BIT_LSB_IDX);        // input clock of CMU_FXCLK --> CMU_GCLK_EN
398     GTM_CMU_CLK_EN.U |= 0x2 << EN_FXCLK_BIT_LSB_IDX;                // enable all CMU_FXCLK
399
400
401     // set GTM TOM0 channel 11 - Buzzer
402     GTM_TOM0_TGC1_GLB_CTRL.U |= 0x2 << UPEN_CTRL3_BIT_LSB_IDX;      // TOM0 channel 11 enable
403     GTM_TOM0_TGC1_ENDIS_CTRL.U |= 0x2 << ENDIS_CTRL3_BIT_LSB_IDX;   // enable channel 11 on update trigger
404     GTM_TOM0_TGC1_OUTEN_CTRL.U |= 0x2 << OUTEN_CTRL3_BIT_LSB_IDX;   // enable channel 11 output on update trigger
405
406
407     // TOM0 CH11
408     GTM_TOM0_CH11_CTRL.U |= 0x1 << SL_BIT_LSB_IDX;                  // high signal level for duty cycle
409     GTM_TOM0_CH11_CTRL.U |= 0x1 << CLK_SRC_SR_BIT_LSB_IDX;          // clock source --> CMU_FXCLK(1) = 6250 kHz
410     GTM_TOM0_CH11_CTRL.U &= ~(0x0 << OSM_BIT_LSB_IDX);              // continuous mode
411     GTM_TOM0_CH11_CTRL.U &= ~(0x0 << TRIGOUT_BIT_LSB_IDX);
412     GTM_TOM0_CH11_SR0.U = 12500 - 1;                                // PWM freq. = 6250 kHz / 12500 = 500 Hz (temp)
413     GTM_TOM0_CH11_SR1.U = 6250 - 1;                                 // duty cycle = 6250 / 12500 = 50 % (temp)
414
415     // TOUT pin selection
416     GTM_TOUTSEL0.U &= ~(0x3 << SEL3_BIT_LSB_IDX);                   // TOUT3  --> TOM0 channel 11
417 }
418
```

```
100 #define ENDIS_CTRL3_BIT_LSB_IDX     6
```

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) = **0x8278**

   → TOM0_TGC1_OUTEN_CTRL 레지스터 주소 = 0xF0100000 + 0x8278 = **0xF0108278**

TOM0_TGC1_OUTEN_CTRL 레지스터 @ 0xF0108278

GTM_TOMi_TGC1_OUTEN_CTRL (i=0-2)

TOMi TGC1 Output Enable Control Register

$(08278_H+i*800_H)$ offset

Reset Value: 00000000_H

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OUTEN_CTRL7		OUTEN_CTRL6		OUTEN_CTRL5		OUTEN_CTRL4		OUTEN_CTRL3		OUTEN_CTRL2		OUTEN_CTRL1		OUTEN_CTRL0	
rw		rw		rw		rw		rw		rw		rw		rw	

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

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 11의 출력이 enable** 되도록 하기 위해 **OUTEN_CTRL3 영역에 0x2 write**

TOM0_TGC1_OUTEN_CTRL 레지스터 @ 0xF0108278

GTM_TOMi_TGC1_OUTEN_CTRL (i=0-2)
TOMi TGC1 Output Enable Control Register
(08278$_H$+i*800$_H$) Reset Value: 00000000$_H$

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OUTEN_CTRL7		OUTEN_CTRL6		OUTEN_CTRL5		OUTEN_CTRL4		OUTEN_CTRL3		OUTEN_CTRL2		OUTEN_CTRL1		OUTEN_CTRL0	
rw		rw		rw		rw		rw		rw		rw		rw	

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

TGC1이 제어하는 채널8~15
채널11은 8부터 시작하면 **3번째**

OUTEN_CTRL3	[7:6]	rw	**Output (A)TOM_OUT(3) 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

Lab12: GTM 레지스터 설정 – TOM0_TGC1_OUTEN_CTRL

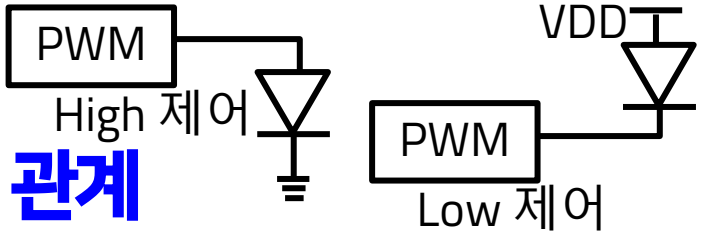
:TOM의 출력이 trigger 이벤트에 반응하도록 설정

```
372
373 void initGTM(void)
374 {
375     // Password Access to unlock SCU_WDTSCON0
376     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
377     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0);    // wait until unlocked
378
379     // Modify Access to clear ENDINIT
380     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) & ~(1 << ENDINIT_BIT_LSB_IDX);
381     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0);    // wait until locked
382
383     GTM_CLC.U &= ~(1 << DISR_BIT_LSB_IDX);    // enable GTM
384
385     // Password Access to unlock SCU_WDTSCON0
386     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
387     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0);    // wait until unlocked
388
389     // Modify Access to set ENDINIT
390     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
391     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0);
392
393     while((GTM_CLC.U & (1 << DISS_BIT_LSB_IDX)) != 0); // wait until GTM module enabled
394
395
396     // GTM clock configuration
397     GTM_CMU_FXCLK_CTRL.U &= ~(0xF << FXCLK_SEL_BIT_LSB_IDX);        // input clock of CMU_FXCLK --> CMU_GCLK_EN
398     GTM_CMU_CLK_EN.U |= 0x2 << EN_FXCLK_BIT_LSB_IDX;                // enable all CMU_FXCLK
399
400
401     // set GTM TOM0 channel 11 - Buzzer
402     GTM_TOM0_TGC1_GLB_CTRL.U |= 0x2 << UPEN_CTRL3_BIT_LSB_IDX;      // TOM0 channel 11 enable
403     GTM_TOM0_TGC1_ENDIS_CTRL.U |= 0x2 << ENDIS_CTRL3_BIT_LSB_IDX;   // enable channel 11 on update trigger
404     GTM_TOM0_TGC1_OUTEN_CTRL.U |= 0x2 << OUTEN_CTRL3_BIT_LSB_IDX;   // enable channel 11 output on update trigger
405
406
407     // TOM0 CH11
408     GTM_TOM0_CH11_CTRL.U |= 0x1 << SL_BIT_LSB_IDX;                  // high signal level for duty cycle
409     GTM_TOM0_CH11_CTRL.U |= 0x1 << CLK_SRC_SR_BIT_LSB_IDX;          // clock source --> CMU_FXCLK(1) = 6250 kHz
410     GTM_TOM0_CH11_CTRL.U &= ~(0x0 << OSM_BIT_LSB_IDX);              // continuous mode
411     GTM_TOM0_CH11_CTRL.U &= ~(0x0 << TRIGOUT_BIT_LSB_IDX);
412     GTM_TOM0_CH11_SR0.U = 12500 - 1;                                // PWM freq. = 6250 kHz / 12500 = 500 Hz (temp)
413     GTM_TOM0_CH11_SR1.U = 6250 - 1;                                 // duty cycle = 6250 / 12500 = 50 % (temp)
414
415     // TOUT pin selection
416     GTM_TOUTSEL0.U &= ~(0x3 << SEL3_BIT_LSB_IDX);                   // TOUT3  --> TOM0 channel 11
417 }
418
```

```
101 #define OUTEN_CTRL3_BIT_LSB_IDX     6
```

PWM 신호 생성 flow

:PWM 생성과정의 counter와 duty cycle 관계



- TOM에서 이루어지는 PWM 동작 개요

- TOM Channel은 PWM 신호를 생성하기 위해 3가지 값 사용
  - CN0 : TOM clock 주기마다 1씩 증가하는 counter 값
  - CM0 : PWM 신호의 주기를 결정하는 값
  - CM1 : PWM 신호의 Duty Cycle 을 결정하는 값

- CN0 이 clock 주기마다 증가하다가
  - → **CM1 (PWM Duty Cycle 길이)에 도달하면 출력 값 반전**
  - → **CM0 (PWM 1주기 길이)에 도달하면 0으로 초기화**

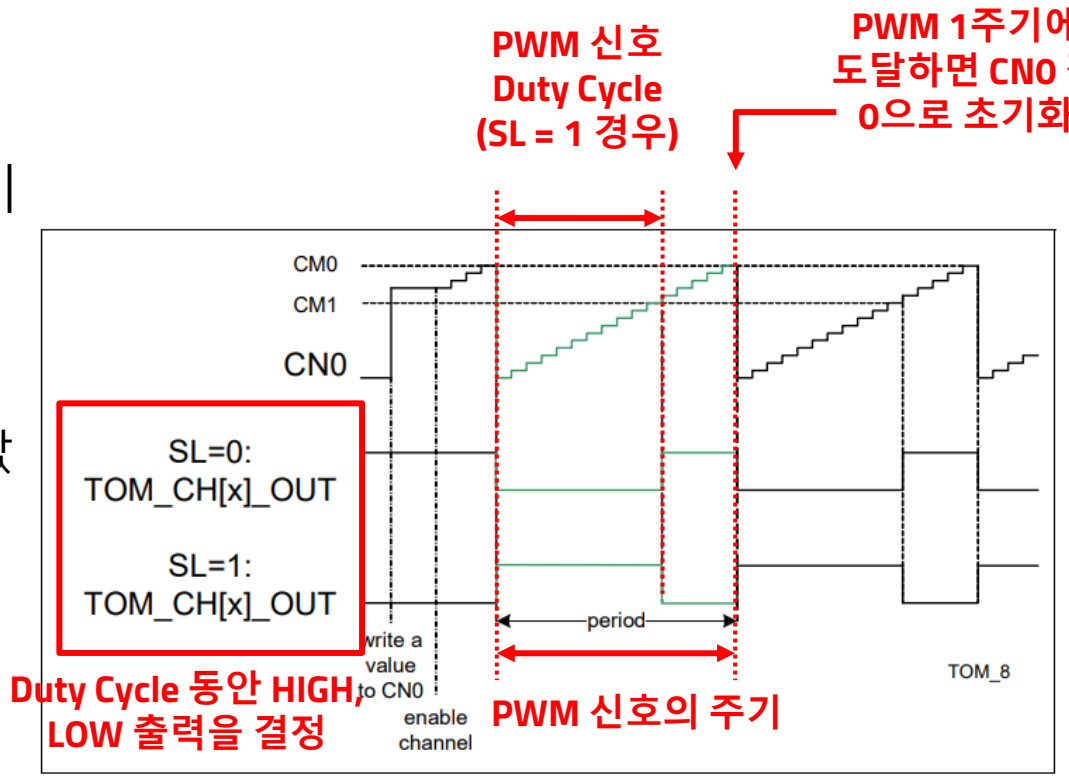


Figure 25-39 PWM Output with respect to configuration bit SL in continuos mode

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 11**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_CH11_CTRL** 레지스터 설정 필요
  - **TOM0_CH11_SR0** 레지스터 설정 필요
  - **TOM0_CH11_SR1** 레지스터 설정 필요

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

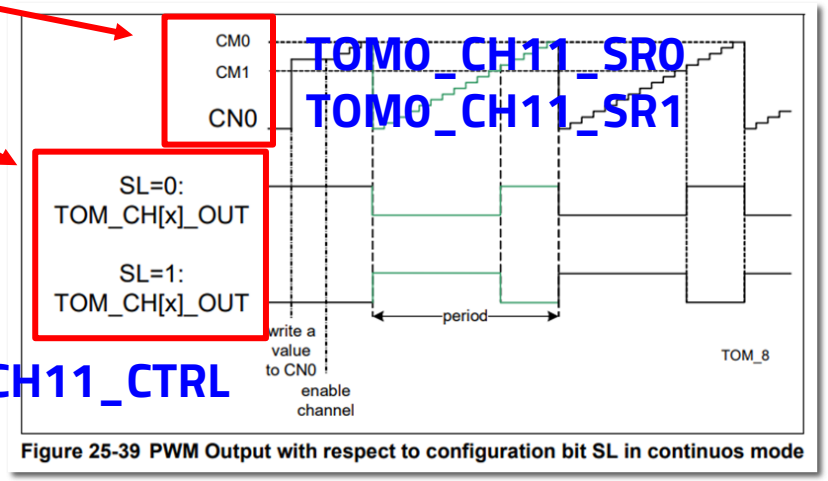


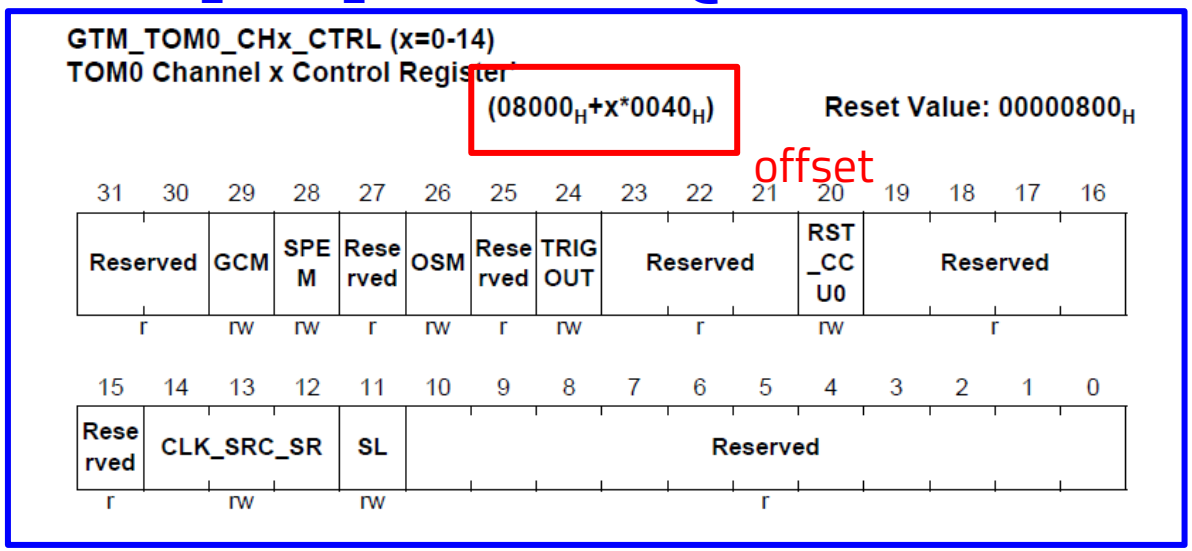
Figure 25-39 PWM Output with respect to configuration bit SL in continuos mode

GTM 레지스터 설정 – TOM0_CH11_CTRL

1. GTM 레지스터 영역의 주소 찾기
   - 시작 주소 (Base address) **= 0xF0100000**
2. 사용할 레지스터의 주소 찾기 (채널 11 이므로 x = 11)
   - **TOM0_CH11_CTRL** 의 Offset Address = 0x8000 + (x * 0x40) = **0x82C0**

   → TOM0_CH11_CTRL 레지스터 주소 = 0xF0100000 + 0x82C0 = **0xF01082C0**

TOM0_CH11_CTRL 레지스터 @ 0xF01082C0



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

GTM 레지스터 설정 – TOM0_CH11_CTRL
:PWM 신호 중 duty cycle을 어떤 값으로 출력할 지 설정

3. 레지스터 write 값 결정
   - TOM0 채널11의 동작을 설정
   - PWM 신호의 Duty Cycle 영역에서 신호의 값을 HIGH로 출력하기 위해 **SL 영역에 0x1 write**

TOM0_CH11_CTRL 레지스터 @ 0xF01082C0

GTM_TOM0_CHx_CTRL (x=0-14)
TOM0 Channel x Control Register'
(08000_H+x*0040_H) Reset Value: 00000800_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		GCM	SPEM	Reserved	OSM	Reserved	TRIGOUT	Reserved			RST_CCU0	Reserved			
r		rw	rw	r	rw	r	rw	r			rw	r			

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

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.2953

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

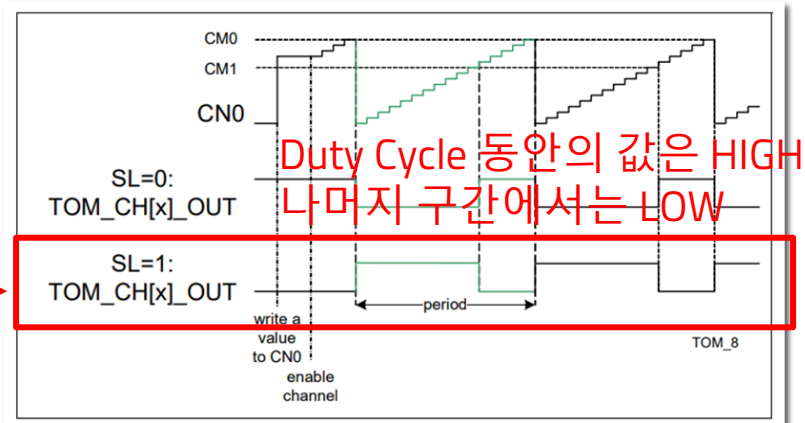


Figure 25-39 PWM Output with respect to configuration bit SL in continuos mode

Lab13: GTM 레지스터 설정 – TOM0_CH11_CTRL

:PWM 신호 중 duty cycle을 어떤 값으로 출력할 지 설정

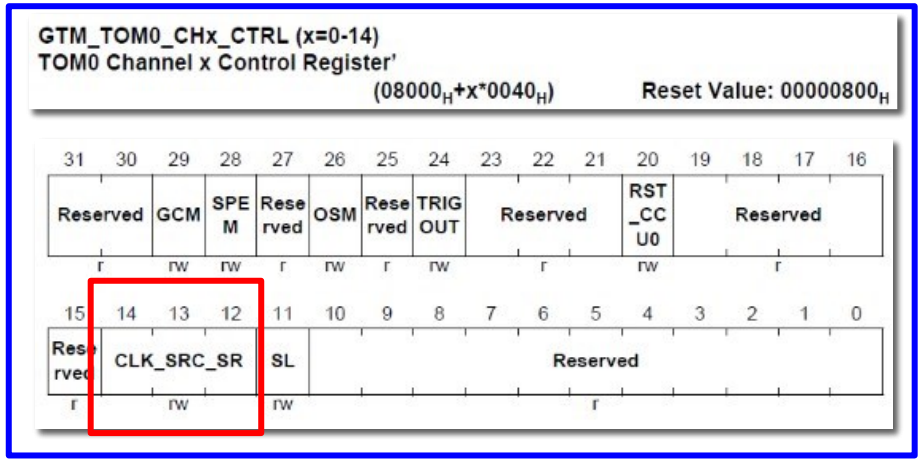
```
372
373 void initGTM(void)
374 {
375     // Password Access to unlock SCU_WDTSCON0
376     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
377     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0);    // wait until unlocked
378
379     // Modify Access to clear ENDINIT
380     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) & ~(1 << ENDINIT_BIT_LSB_IDX);
381     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0);    // wait until locked
382
383     GTM_CLC.U &= ~(1 << DISR_BIT_LSB_IDX);    // enable GTM
384
385     // Password Access to unlock SCU_WDTSCON0
386     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
387     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0);    // wait until unlocked
388
389     // Modify Access to set ENDINIT
390     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
391     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0);
392
393     while((GTM_CLC.U & (1 << DISS_BIT_LSB_IDX)) != 0); // wait until GTM module enabled
394
395
396     // GTM clock configuration
397     GTM_CMU_FXCLK_CTRL.U &= ~(0xF << FXCLK_SEL_BIT_LSB_IDX);        // input clock of CMU_FXCLK --> CMU_GCLK_EN
398     GTM_CMU_CLK_EN.U |= 0x2 << EN_FXCLK_BIT_LSB_IDX;                // enable all CMU_FXCLK
399
400
401     // set GTM TOM0 channel 11 - Buzzer
402     GTM_TOM0_TGC1_GLB_CTRL.U |= 0x2 << UPEN_CTRL3_BIT_LSB_IDX;      // TOM0 channel 11 enable
403     GTM_TOM0_TGC1_ENDIS_CTRL.U |= 0x2 << ENDIS_CTRL3_BIT_LSB_IDX;   // enable channel 11 on update trigger
404     GTM_TOM0_TGC1_OUTEN_CTRL.U |= 0x2 << OUTEN_CTRL3_BIT_LSB_IDX;   // enable channel 11 output on update trigger
405
406
407     // TOM0 CH11
408     GTM_TOM0_CH11_CTRL.U |= 0x1 << SL_BIT_LSB_IDX;                  // high signal level for duty cycle
409     GTM_TOM0_CH11_CTRL.U |= 0x1 << CLK_SRC_SR_BIT_LSB_IDX;          // clock source --> CMU_FXCLK(1) = 6250 kHz
410     GTM_TOM0_CH11_CTRL.U &= ~(0x0 << OSM_BIT_LSB_IDX);              // continuous mode
411     GTM_TOM0_CH11_CTRL.U &= ~(0x0 << TRIGOUT_BIT_LSB_IDX);
412     GTM_TOM0_CH11_SR0.U = 12500 - 1;                                // PWM freq. = 6250 kHz / 12500 = 500 Hz (temp)
413     GTM_TOM0_CH11_SR1.U = 6250 - 1;                                 // duty cycle = 6250 / 12500 = 50 % (temp)
414
415     // TOUT pin selection
416     GTM_TOUTSEL0.U &= ~(0x3 << SEL3_BIT_LSB_IDX);                   // TOUT3  --> TOM0 channel 11
417 }
418
```

```
105 #define SL_BIT_LSB_IDX              11
106
```

GTM 레지스터 설정 – TOM0_CH11_CTRL
:FXU에서 생성하고 TOM에서 사용할 clock 주파수 설정

3. 레지스터 write 값 결정
   - TOM0 채널11의 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



TOM0_CH11_CTRL 레지스터 @ 0xF01082C0

FXU로 입력되는 system clock을 2^4 (= 16)으로 나눠서 FXCLK 로 사용
→ **100 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.

Lab14: GTM 레지스터 설정 – TOM0_CH11_CTRL

:FXU에서 생성하고 TOM에서 사용할 clock 주파수 설정

```
372
373 void initGTM(void)
374 {
375     // Password Access to unlock SCU_WDTSCON0
376     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
377     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0);    // wait until unlocked
378
379     // Modify Access to clear ENDINIT
380     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) & ~(1 << ENDINIT_BIT_LSB_IDX);
381     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0);    // wait until locked
382
383     GTM_CLC.U &= ~(1 << DISR_BIT_LSB_IDX);    // enable GTM
384
385     // Password Access to unlock SCU_WDTSCON0
386     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
387     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0);    // wait until unlocked
388
389     // Modify Access to set ENDINIT
390     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
391     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0);
392
393     while((GTM_CLC.U & (1 << DISS_BIT_LSB_IDX)) != 0); // wait until GTM module enabled
394
395
396     // GTM clock configuration
397     GTM_CMU_FXCLK_CTRL.U &= ~(0xF << FXCLK_SEL_BIT_LSB_IDX);        // input clock of CMU_FXCLK --> CMU_GCLK_EN
398     GTM_CMU_CLK_EN.U |= 0x2 << EN_FXCLK_BIT_LSB_IDX;                // enable all CMU_FXCLK
399
400
401     // set GTM TOM0 channel 11 - Buzzer
402     GTM_TOM0_TGC1_GLB_CTRL.U |= 0x2 << UPEN_CTRL3_BIT_LSB_IDX;      // TOM0 channel 11 enable
403     GTM_TOM0_TGC1_ENDIS_CTRL.U |= 0x2 << ENDIS_CTRL3_BIT_LSB_IDX;   // enable channel 11 on update trigger
404     GTM_TOM0_TGC1_OUTEN_CTRL.U |= 0x2 << OUTEN_CTRL3_BIT_LSB_IDX;   // enable channel 11 output on update trigger
405
406
407     // TOM0 CH11
408     GTM_TOM0_CH11_CTRL.U |= 0x1 << SL_BIT_LSB_IDX;                  // high signal level for duty cycle
409     GTM_TOM0_CH11_CTRL.U |= 0x1 << CLK_SRC_SR_BIT_LSB_IDX;          // clock source --> CMU_FXCLK(1) = 6250 kHz
410     GTM_TOM0_CH11_CTRL.U &= ~(0x0 << OSM_BIT_LSB_IDX);              // continuous mode
411     GTM_TOM0_CH11_CTRL.U &= ~(0x0 << TRIGOUT_BIT_LSB_IDX);
412     GTM_TOM0_CH11_SR0.U = 12500 - 1;                                // PWM freq. = 6250 kHz / 12500 = 500 Hz (temp)
413     GTM_TOM0_CH11_SR1.U = 6250 - 1;                                 // duty cycle = 6250 / 12500 = 50 % (temp)
414
415     // TOUT pin selection
416     GTM_TOUTSEL0.U &= ~(0x3 << SEL3_BIT_LSB_IDX);                   // TOUT3  --> TOM0 channel 11
417 }
418
```

```
102 #define CLK_SRC_SR_BIT_LSB_IDX      12
```

GTM 레지스터 설정 – TOM0_CH11_CTRL
:연속적으로 PWM 신호가 생성되는 Continuous 모드

3. 레지스터 write 값 결정
   - TOM0 채널 11에서 생성되는 PWM 신호를 continuous 모드로 사용하기 위해 **OSM 영역에 0x0 write**

PWM 주기가 연속으로 생성되는 모드

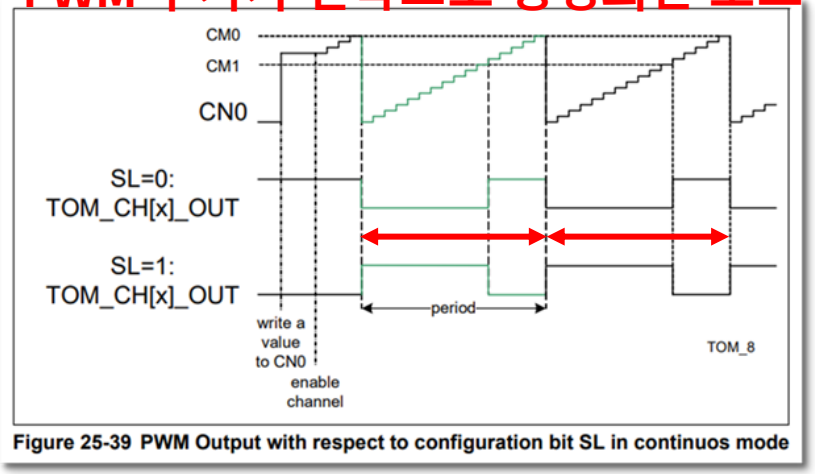


Figure 25-39 PWM Output with respect to configuration bit SL in continuos mode

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

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

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		GCM	SPEM	Reserved	OSM	Reserved	TRIGOUT	Reserved			RST_CCU0	Reserved			
r		rw	rw	r	rw	r	rw	r			rw	r			

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

TOM0_CH11_CTRL 레지스터 @ 0xF01082C0

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 0_B One-shot mode disabled 1_B One-shot mode enabled

One-shot 모드 disable
= Continuous 모드

Lab15: GTM 레지스터 설정 – TOM0_CH11_CTRL

:연속적으로 PWM 신호가 생성되는 Continuous 모드

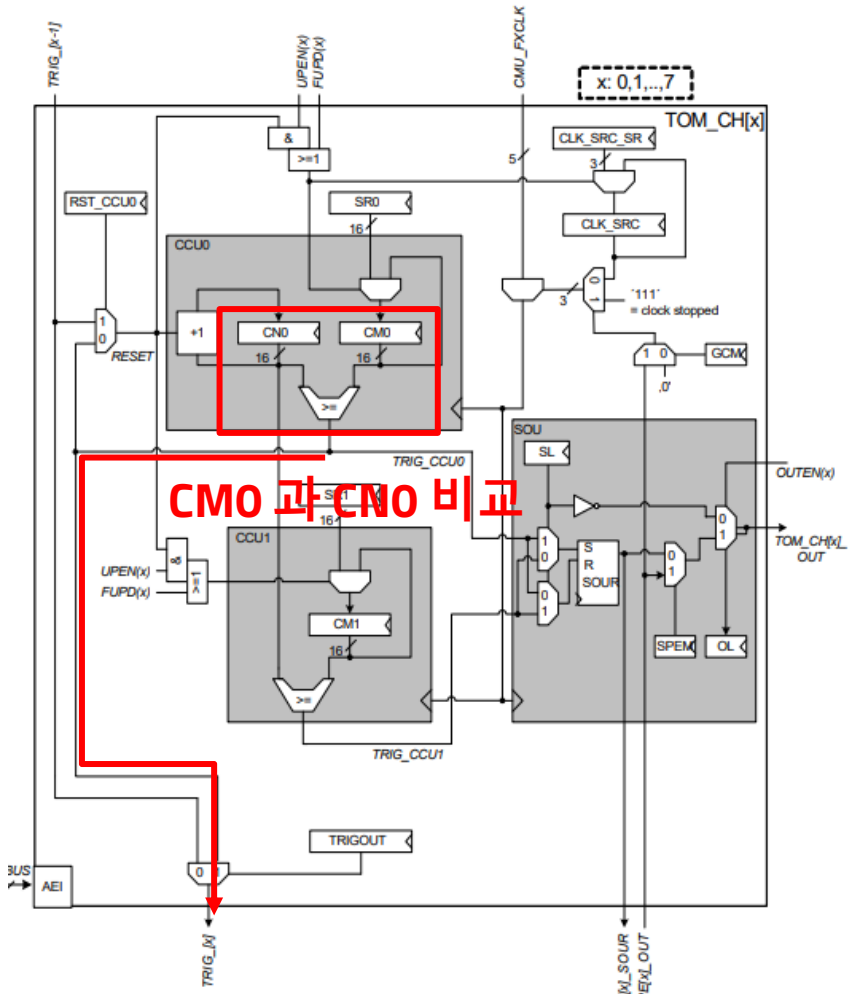
```
372
373 void initGTM(void)
374 {
375     // Password Access to unlock SCU_WDTSCON0
376     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
377     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0);    // wait until unlocked
378
379     // Modify Access to clear ENDINIT
380     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) & ~(1 << ENDINIT_BIT_LSB_IDX);
381     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0);    // wait until locked
382
383     GTM_CLC.U &= ~(1 << DISR_BIT_LSB_IDX);    // enable GTM
384
385     // Password Access to unlock SCU_WDTSCON0
386     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
387     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0);    // wait until unlocked
388
389     // Modify Access to set ENDINIT
390     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
391     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0);
392
393     while((GTM_CLC.U & (1 << DISS_BIT_LSB_IDX)) != 0); // wait until GTM module enabled
394
395
396     // GTM clock configuration
397     GTM_CMU_FXCLK_CTRL.U &= ~(0xF << FXCLK_SEL_BIT_LSB_IDX);        // input clock of CMU_FXCLK --> CMU_GCLK_EN
398     GTM_CMU_CLK_EN.U |= 0x2 << EN_FXCLK_BIT_LSB_IDX;                // enable all CMU_FXCLK
399
400
401     // set GTM TOM0 channel 11 - Buzzer
402     GTM_TOM0_TGC1_GLB_CTRL.U |= 0x2 << UPEN_CTRL3_BIT_LSB_IDX;      // TOM0 channel 11 enable
403     GTM_TOM0_TGC1_ENDIS_CTRL.U |= 0x2 << ENDIS_CTRL3_BIT_LSB_IDX;   // enable channel 11 on update trigger
404     GTM_TOM0_TGC1_OUTEN_CTRL.U |= 0x2 << OUTEN_CTRL3_BIT_LSB_IDX;   // enable channel 11 output on update trigger
405
406
407     // TOM0 CH11
408     GTM_TOM0_CH11_CTRL.U |= 0x1 << SL_BIT_LSB_IDX;                  // high signal level for duty cycle
409     GTM_TOM0_CH11_CTRL.U |= 0x1 << CLK_SRC_SR_BIT_LSB_IDX;          // clock source --> CMU_FXCLK(1) = 6250 kHz
410     GTM_TOM0_CH11_CTRL.U &= ~(0x0 << OSM_BIT_LSB_IDX);              // continuous mode
411     GTM_TOM0_CH11_CTRL.U &= ~(0x0 << TRIGOUT_BIT_LSB_IDX);
412     GTM_TOM0_CH11_SR0.U = 12500 - 1;                                // PWM freq. = 6250 kHz / 12500 = 500 Hz (temp)
413     GTM_TOM0_CH11_SR1.U = 6250 - 1;                                 // duty cycle = 6250 / 12500 = 50 % (temp)
414
415     // TOUT pin selection
416     GTM_TOUTSEL0.U &= ~(0x3 << SEL3_BIT_LSB_IDX);                   // TOUT3  --> TOM0 channel 11
417 }
418
```

```
103 #define OSM_BIT_LSB_IDX             26
```

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

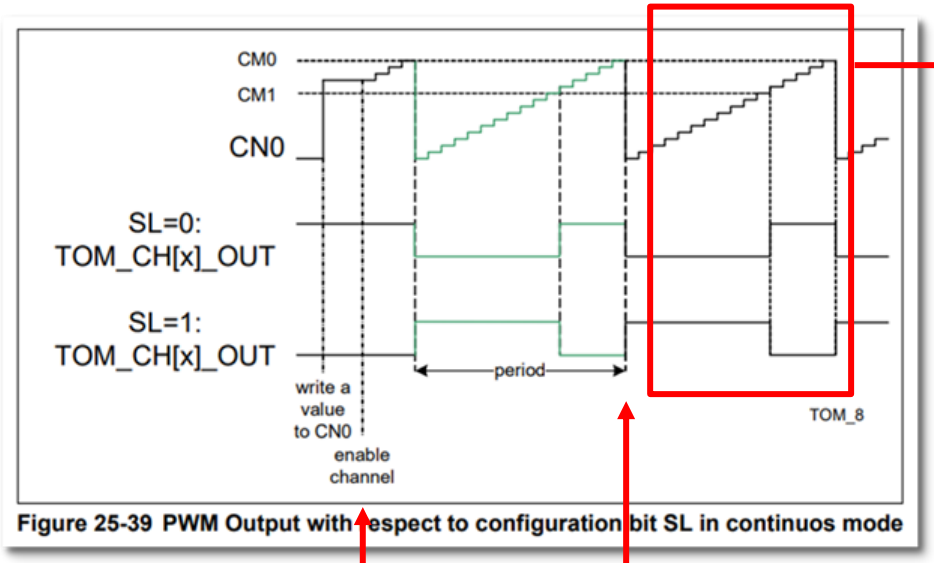


Figure 25-39 PWM Output with respect to configuration bit SL in continuos mode

새로 업데이트된
CM0, CM1 값으로
PWM 출력

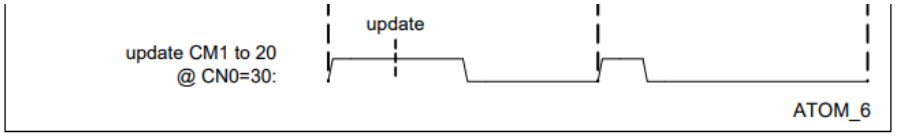


Figure 25-37 synchronous update of duty cycle

Infineon-TC27x_D-step-UM-v02_02-EN.pdf p.2928

최초 SW에서 발생한
Update Trigger에 의해
시작

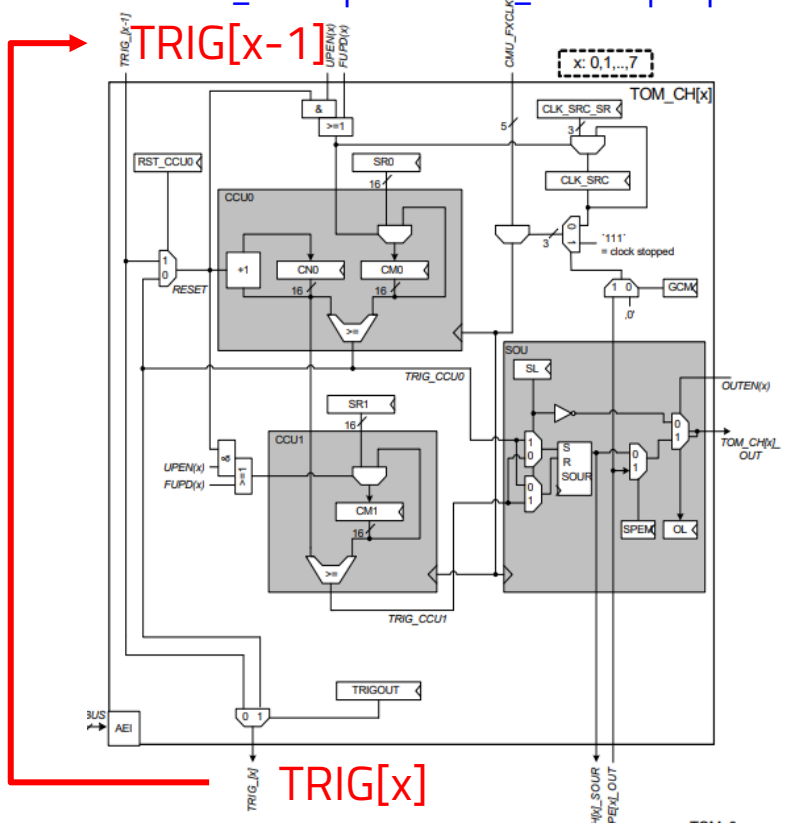
PWM 1주기 도달
→ SR0, SR1 레지스터 (shadow) 에 저장된 값을 각각 CM0, CM1로 copy

GTM 레지스터 설정 – TOM0_CH11_CTRL

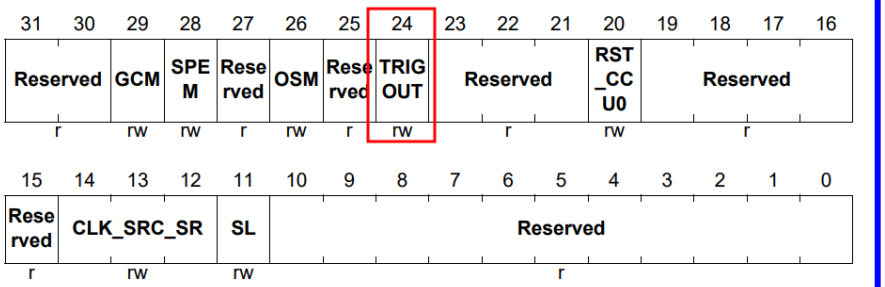
:PWM 신호에서 발생하는 trigger 신호 설정

3. 레지스터 write 값 결정
   - **PWM 신호의 1주기 도달 (CN0 = 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.2923



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



31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		GCM	SPEM	Reserved	OSM	Reserved	TRIGOUT	Reserved			RST_CCU0	Reserved			
r		rw	rw	r	rw	r	rw	r			rw	r			

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

TOM0_CH11_CTRL 레지스터 @ 0xF01082C0

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_CCU0

TRIG 신호 선택

Lab16: GTM 레지스터 설정 – TOM0_CH11_CTRL

:PWM 신호에서 발생하는 trigger 신호 설정

```
372
373 void initGTM(void)
374 {
375     // Password Access to unlock SCU_WDTSCON0
376     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
377     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0);    // wait until unlocked
378
379     // Modify Access to clear ENDINIT
380     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) & ~(1 << ENDINIT_BIT_LSB_IDX);
381     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0);    // wait until locked
382
383     GTM_CLC.U &= ~(1 << DISR_BIT_LSB_IDX);    // enable GTM
384
385     // Password Access to unlock SCU_WDTSCON0
386     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
387     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0);    // wait until unlocked
388
389     // Modify Access to set ENDINIT
390     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
391     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0);
392
393     while((GTM_CLC.U & (1 << DISS_BIT_LSB_IDX)) != 0); // wait until GTM module enabled
394
395
396     // GTM clock configuration
397     GTM_CMU_FXCLK_CTRL.U &= ~(0xF << FXCLK_SEL_BIT_LSB_IDX);        // input clock of CMU_FXCLK --> CMU_GCLK_EN
398     GTM_CMU_CLK_EN.U |= 0x2 << EN_FXCLK_BIT_LSB_IDX;                // enable all CMU_FXCLK
399
400
401     // set GTM TOM0 channel 11 - Buzzer
402     GTM_TOM0_TGC1_GLB_CTRL.U |= 0x2 << UPEN_CTRL3_BIT_LSB_IDX;      // TOM0 channel 11 enable
403     GTM_TOM0_TGC1_ENDIS_CTRL.U |= 0x2 << ENDIS_CTRL3_BIT_LSB_IDX;   // enable channel 11 on update trigger
404     GTM_TOM0_TGC1_OUTEN_CTRL.U |= 0x2 << OUTEN_CTRL3_BIT_LSB_IDX;   // enable channel 11 output on update trigger
405
406
407     // TOM0 CH11
408     GTM_TOM0_CH11_CTRL.U |= 0x1 << SL_BIT_LSB_IDX;                  // high signal level for duty cycle
409     GTM_TOM0_CH11_CTRL.U |= 0x1 << CLK_SRC_SR_BIT_LSB_IDX;          // clock source --> CMU_FXCLK(1) = 6250 kHz
410     GTM_TOM0_CH11_CTRL.U &= ~(0x3 << OSM_BIT_LSB_IDX);              // continuous mode
411     GTM_TOM0_CH11_CTRL.U &= ~(0x0 << TRIGOUT_BIT_LSB_IDX);
412     GTM_TOM0_CH11_SR0.U = 12500 - 1;                                // PWM freq. = 6250 kHz / 12500 = 500 Hz (temp)
413     GTM_TOM0_CH11_SR1.U = 6250 - 1;                                 // duty cycle = 6250 / 12500 = 50 % (temp)
414
415     // TOUT pin selection
416     GTM_TOUTSEL0.U &= ~(0x3 << SEL3_BIT_LSB_IDX);                   // TOUT3  --> TOM0 channel 11
417 }
418
```

```
104 #define TRIGOUT_BIT_LSB_IDX          24
```

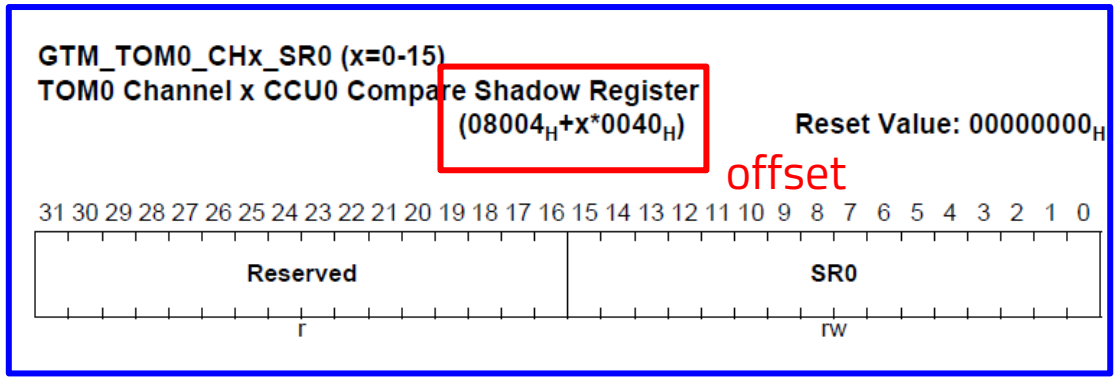
GTM 레지스터 설정 – TOM0_CH11_SR0

1. GTM 레지스터 영역의 주소 찾기
   - 시작 주소 (Base address) **= 0xF0100000**

2. 사용할 레지스터의 주소 찾기 (채널 11 이므로 x = 11)
   - **TOM0_CH11_SR0** 의 Offset Address = 0x8004 + (x * 0x40) = **0x82C4**

   → TOM0_CH11_SR0 레지스터 주소 = 0xF0100000 + 0x82C4 = **0xF01082C4**

TOM0_CH11_SR0 레지스터 @ 0xF01082C4



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

GTM 레지스터 설정 – TOM0_CH11_SR0
:PWM 신호의 주기 설정

3. 레지스터 write 값 결정
   - TOM0 채널1의 동작을 설정

$$PWM\ Period = \frac{\boldsymbol{CM0} + 1}{Freq.of\ CMU\_FXCLK1} = \frac{12500}{6250\ kHz} = 2ms$$

   - PWM 신호의 주기를 2ms 로 설정하기 위해 **SR0 영역에 10진수 (12500 – 1) write**
   - SR0 레지스터에 설정할 CM0 값을 저장하면 Trigger로 update 시 CM0에 반영됨

TOM0_CH11_SR0 레지스터@ 0xF01082C4

GTM_TOM0_CHx_SR0 (x=0-15)
TOM0 Channel x CCU0 Compare Shadow Register
(08004_H+x*0040_H) Reset Value: 00000000_H

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Reserved	SR0
r	rw

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.2962

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

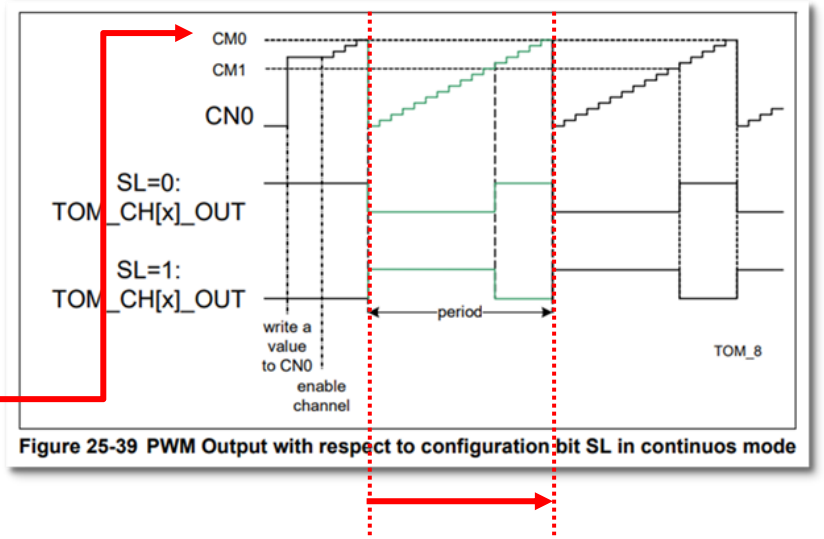


Figure 25-39 PWM Output with respect to configuration bit SL in continuos mode

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

Lab17: GTM 레지스터 설정 – TOM0_CH11_SR0

:PWM 신호의 주기 설정 (임시, 음계 설정을 위해 추후 변경 예정)

```
372
373 void initGTM(void)
374 {
375     // Password Access to unlock SCU_WDTSCON0
376     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
377     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0);    // wait until unlocked
378
379     // Modify Access to clear ENDINIT
380     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) & ~(1 << ENDINIT_BIT_LSB_IDX);
381     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0);    // wait until locked
382
383     GTM_CLC.U &= ~(1 << DISR_BIT_LSB_IDX);    // enable GTM
384
385     // Password Access to unlock SCU_WDTSCON0
386     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
387     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0);    // wait until unlocked
388
389     // Modify Access to set ENDINIT
390     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
391     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0);
392
393     while((GTM_CLC.U & (1 << DISS_BIT_LSB_IDX)) != 0); // wait until GTM module enabled
394
395
396     // GTM clock configuration
397     GTM_CMU_FXCLK_CTRL.U &= ~(0xF << FXCLK_SEL_BIT_LSB_IDX);        // input clock of CMU_FXCLK --> CMU_GCLK_EN
398     GTM_CMU_CLK_EN.U |= 0x2 << EN_FXCLK_BIT_LSB_IDX;                // enable all CMU_FXCLK
399
400
401     // set GTM TOM0 channel 11 - Buzzer
402     GTM_TOM0_TGC1_GLB_CTRL.U |= 0x2 << UPEN_CTRL3_BIT_LSB_IDX;      // TOM0 channel 11 enable
403     GTM_TOM0_TGC1_ENDIS_CTRL.U |= 0x2 << ENDIS_CTRL3_BIT_LSB_IDX;   // enable channel 11 on update trigger
404     GTM_TOM0_TGC1_OUTEN_CTRL.U |= 0x2 << OUTEN_CTRL3_BIT_LSB_IDX;   // enable channel 11 output on update trigger
405
406
407     // TOM0 CH11
408     GTM_TOM0_CH11_CTRL.U |= 0x1 << SL_BIT_LSB_IDX;                  // high signal level for duty cycle
409     GTM_TOM0_CH11_CTRL.U |= 0x1 << CLK_SRC_SR_BIT_LSB_IDX;          // clock source --> CMU_FXCLK(1) = 6250 kHz
410     GTM_TOM0_CH11_CTRL.U &= ~(0x0 << OSM_BIT_LSB_IDX);              // continuous mode
411     GTM_TOM0_CH11_CTRL.U &= ~(0x0 << TRIGOUT_BIT_LSB_IDX);
412     GTM_TOM0_CH11_SR0.U = 12500 - 1;                                // PWM freq. = 6250 kHz / 12500 = 500 Hz (temp)
413     GTM_TOM0_CH11_SR1.U = 6250 - 1;                                 // duty cycle = 6250 / 12500 = 50 % (temp)
414
415     // TOUT pin selection
416     GTM_TOUTSEL0.U &= ~(0x3 << SEL3_BIT_LSB_IDX);                   // TOUT3  --> TOM0 channel 11
417 }
418
```

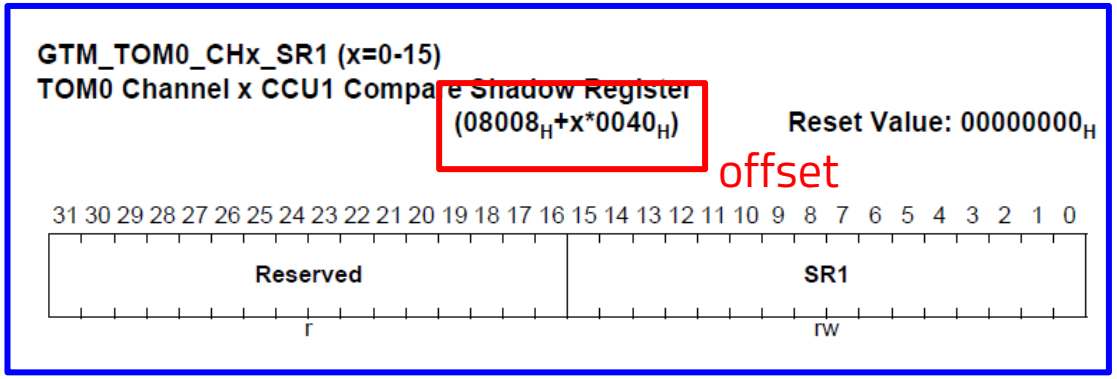
GTM 레지스터 설정 – TOM0_CH11_SR1

1. GTM 레지스터 영역의 주소 찾기
   - 시작 주소 (Base address) **= 0xF0100000**

2. 사용할 레지스터의 주소 찾기 (채널 1 이므로 x = 1)
   - **TOM0_CH11_SR1** 의 Offset Address = 0x8008 + (x * 0x40) = **0x82C8**

   → TOM0_CH11_SR1 레지스터 주소 = 0xF0100000 + 0x82C8 = **0xF01082C8**

TOM0_CH11_SR1 레지스터@ 0xF01082C8



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

GTM 레지스터 설정 – TOM0_CH11_SR1

:PWM 신호의 duty cycle % 설정

3. 레지스터 write 값 결정
   - TOM0 채널11의 동작을 설정
   - PWM 신호의 Duty Cycle을 설정하기 위해 **SR1 영역에 write**
   - SR1 레지스터에 설정할 CM1 값을 저장하면 Trigger로 update 시 CM1에 반영됨

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

TOM0_CH11_SR1 레지스터@ 0xF01082C8

GTM_TOM0_CHx_SR1 (x=0-15)
TOM0 Channel x CCU1 Compare Shadow Register
(08008_H+x*0040_H) Reset Value: 00000000_H

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Reserved	SR1
r	rw

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.2964

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

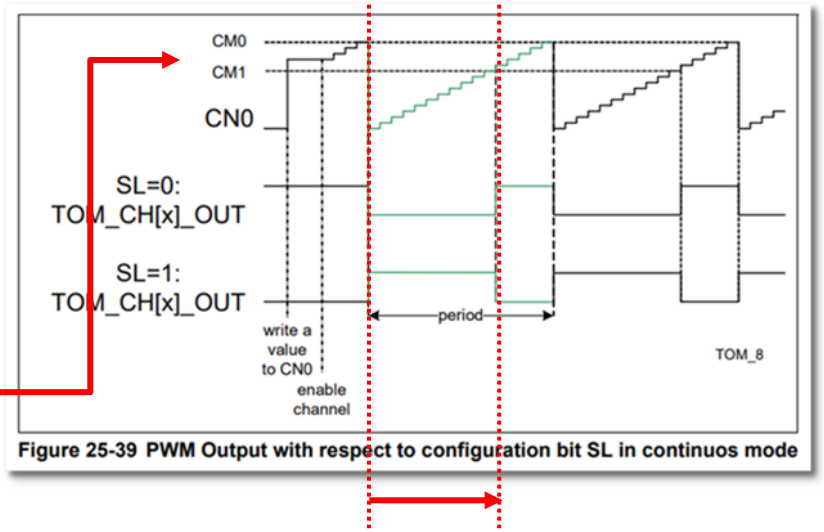


Figure 25-39 PWM Output with respect to configuration bit SL in continuos mode

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

Lab18: GTM 레지스터 설정 – TOM0_CH11_SR1

:PWM 신호의 duty cycle % 설정 (임시, 추후 변경 예정)

```
372
373 void initGTM(void)
374 {
375     // Password Access to unlock SCU_WDTSCON0
376     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
377     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0);    // wait until unlocked
378
379     // Modify Access to clear ENDINIT
380     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) & ~(1 << ENDINIT_BIT_LSB_IDX);
381     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0);    // wait until locked
382
383     GTM_CLC.U &= ~(1 << DISR_BIT_LSB_IDX);    // enable GTM
384
385     // Password Access to unlock SCU_WDTSCON0
386     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
387     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0);    // wait until unlocked
388
389     // Modify Access to set ENDINIT
390     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
391     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0);
392
393     while((GTM_CLC.U & (1 << DISS_BIT_LSB_IDX)) != 0); // wait until GTM module enabled
394
395
396     // GTM clock configuration
397     GTM_CMU_FXCLK_CTRL.U &= ~(0xF << FXCLK_SEL_BIT_LSB_IDX);        // input clock of CMU_FXCLK --> CMU_GCLK_EN
398     GTM_CMU_CLK_EN.U |= 0x2 << EN_FXCLK_BIT_LSB_IDX;                // enable all CMU_FXCLK
399
400
401     // set GTM TOM0 channel 11 - Buzzer
402     GTM_TOM0_TGC1_GLB_CTRL.U |= 0x2 << UPEN_CTRL3_BIT_LSB_IDX;      // TOM0 channel 11 enable
403     GTM_TOM0_TGC1_ENDIS_CTRL.U |= 0x2 << ENDIS_CTRL3_BIT_LSB_IDX;   // enable channel 11 on update trigger
404     GTM_TOM0_TGC1_OUTEN_CTRL.U |= 0x2 << OUTEN_CTRL3_BIT_LSB_IDX;   // enable channel 11 output on update trigger
405
406
407     // TOM0 CH11
408     GTM_TOM0_CH11_CTRL.U |= 0x1 << SL_BIT_LSB_IDX;                  // high signal level for duty cycle
409     GTM_TOM0_CH11_CTRL.U |= 0x1 << CLK_SRC_SR_BIT_LSB_IDX;          // clock source --> CMU_FXCLK(1) = 6250 kHz
410     GTM_TOM0_CH11_CTRL.U &= ~(0x0 << OSM_BIT_LSB_IDX);              // continuous mode
411     GTM_TOM0_CH11_CTRL.U &= ~(0x0 << TRIGOUT_BIT_LSB_IDX);
412     GTM_TOM0_CH11_SR0.U = 12500 - 1;                                // PWM freq. = 6250 kHz / 12500 = 500 Hz (2ms)
413     GTM_TOM0_CH11_SR1.U = 6250 - 1;                                 // duty cycle = 6250 / 12500 = 50 % (temp)
414
415     // TOUT pin selection
416     GTM_TOUTSEL0.U &= ~(0x3 << SEL3_BIT_LSB_IDX);                   // TOUT3  --> TOM0 channel 11
417 }
418
```

SW 프로그래밍

:GTM 레지스터의 각 영역(필드) LSB 비트 시작 위치 define 정의

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

→ 헤더 파일 참조 추가

```
// GTM registers
#define DISS_BIT_LSB_IDX            1
#define DISR_BIT_LSB_IDX            0
#define SEL3_BIT_LSB_IDX            6
#define EN_FXCLK_BIT_LSB_IDX        22
#define FXCLK_SEL_BIT_LSB_IDX       0

// GTM - TOM0 registers
#define UPEN_CTRL3_BIT_LSB_IDX      22
#define HOST_TRIG_BIT_LSB_IDX       0
#define ENDIS_CTRL3_BIT_LSB_IDX     6
#define OUTEN_CTRL3_BIT_LSB_IDX     6
#define FUPD_CTRL1_BIT_LSB_IDX      2
#define CLK_SRC_SR_BIT_LSB_IDX      12
#define SL_BIT_LSB_IDX              11
#define OSM_BIT_LSB_IDX             26
#define TRIGOUT_BIT_LSB_IDX         24
```

GTM 레지스터 bit shift offset

SW 프로그래밍

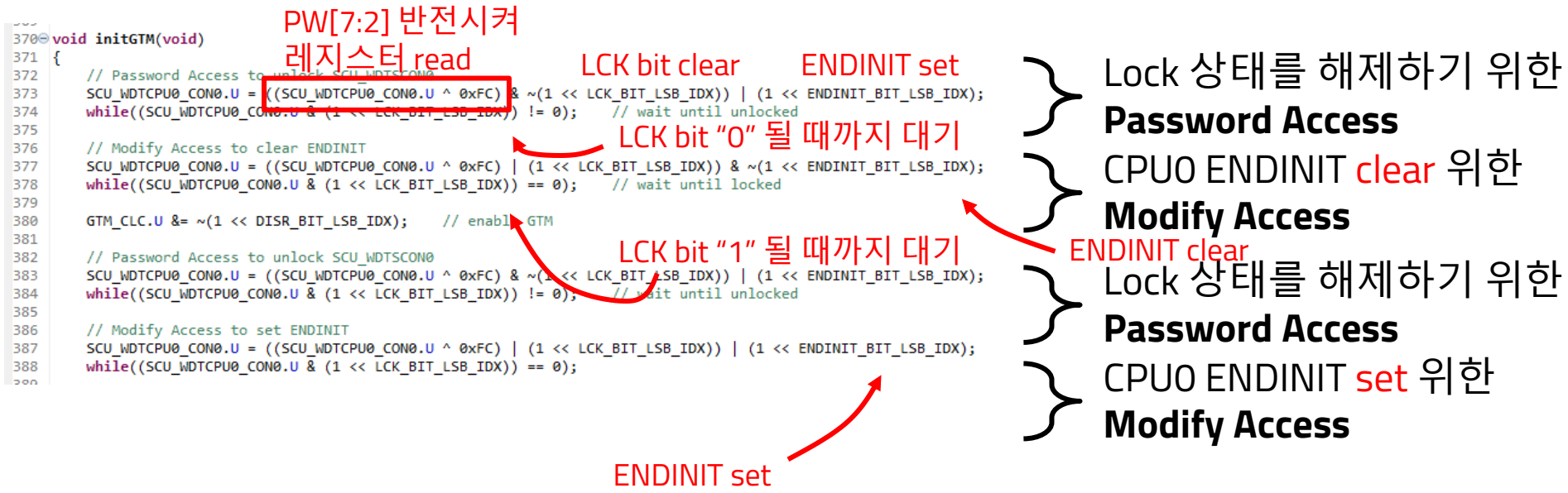
- GTM 모듈 사용을 위한 초기화 함수 *initGTM()*

```
372
373 void initGTM(void)
374 {
375     // Password Access to unlock SCU_WDTSCON0
376     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
377     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0);    // wait until unlocked
378
379     // Modify Access to clear ENDINIT
380     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) & ~(1 << ENDINIT_BIT_LSB_IDX);
381     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0);    // wait until locked
382
383     GTM_CLC.U &= ~(1 << DISR_BIT_LSB_IDX);    // enable GTM
384
385     // Password Access to unlock SCU_WDTSCON0
386     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) & ~(1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
387     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) != 0);    // wait until unlocked
388
389     // Modify Access to set ENDINIT
390     SCU_WDTCPU0_CON0.U = ((SCU_WDTCPU0_CON0.U ^ 0xFC) | (1 << LCK_BIT_LSB_IDX)) | (1 << ENDINIT_BIT_LSB_IDX);
391     while((SCU_WDTCPU0_CON0.U & (1 << LCK_BIT_LSB_IDX)) == 0);
392
393     while((GTM_CLC.U & (1 << DISS_BIT_LSB_IDX)) != 0); // wait until GTM module enabled
394
395
396     // GTM clock configuration
397     GTM_CMU_FXCLK_CTRL.U &= ~(0xF << FXCLK_SEL_BIT_LSB_IDX);        // input clock of CMU_FXCLK --> CMU_GCLK_EN
398     GTM_CMU_CLK_EN.U |= 0x2 << EN_FXCLK_BIT_LSB_IDX;                // enable all CMU_FXCLK
399
400
401     // set GTM TOM0 channel 11 - Buzzer
402     GTM_TOM0_TGC1_GLB_CTRL.U |= 0x2 << UPEN_CTRL3_BIT_LSB_IDX;      // TOM0 channel 11 enable
403     GTM_TOM0_TGC1_ENDIS_CTRL.U |= 0x2 << ENDIS_CTRL3_BIT_LSB_IDX;   // enable channel 11 on update trigger
404     GTM_TOM0_TGC1_OUTEN_CTRL.U |= 0x2 << OUTEN_CTRL3_BIT_LSB_IDX;   // enable channel 11 output on update trigger
405
406
407     // TOM0 CH11
408     GTM_TOM0_CH11_CTRL.U |= 0x1 << SL_BIT_LSB_IDX;                  // high signal level for duty cycle
409     GTM_TOM0_CH11_CTRL.U |= 0x1 << CLK_SRC_SR_BIT_LSB_IDX;          // clock source --> CMU_FXCLK(1) = 6250 kHz
410     GTM_TOM0_CH11_CTRL.U &= ~(0x0 << OSM_BIT_LSB_IDX);              // continuous mode
411     GTM_TOM0_CH11_CTRL.U &= ~(0x0 << TRIGOUT_BIT_LSB_IDX);
412     GTM_TOM0_CH11_SR0.U = 12500 - 1;                                // PWM freq. = 6250 kHz / 12500 = 500 Hz (temp)
413     GTM_TOM0_CH11_SR1.U = 6250 - 1;                                 // duty cycle = 6250 / 12500 = 50 % (temp)
414
415     // TOUT pin selection
416     GTM_TOUTSEL0.U &= ~(0x3 << SEL3_BIT_LSB_IDX);                   // TOUT3  --> TOM0 channel 11
417 }
418
```

SW 프로그래밍

:GTM 모듈 사용 설정 전, 보호 레지스터 잠금 해제 필요
:GTM 모듈 사용 설정 후, 보호 레지스터 잠금 재설정 필요

- GTM 모듈 사용을 위한 초기화 함수 *initGTM()*




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

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

    GTM_CLC.U &= ~(1 << DISR_BIT_LSB_IDX);    // enable GTM

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

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

SW 프로그래밍
:GTM 모듈 설정

- GTM 모듈 사용을 위한 초기화 함수 *initGTM()*

```
393     while((GTM_CLC.U & (1 << DISS_BIT_LSB_IDX)) != 0); // wait until GTM module enabled
394
395
396     // GTM clock configuration
397     GTM_CMU_FXCLK_CTRL.U &= ~(0xF << FXCLK_SEL_BIT_LSB_IDX);        // input clock of CMU_FXCLK --> CMU_GCLK_EN
398     GTM_CMU_CLK_EN.U |= 0x2 << EN_FXCLK_BIT_LSB_IDX;                // enable all CMU_FXCLK
399
400
401     // set GTM TOM0 channel 11 - Buzzer
402     GTM_TOM0_TGC1_GLB_CTRL.U |= 0x2 << UPEN_CTRL3_BIT_LSB_IDX;      // TOM0 channel 11 enable
403     GTM_TOM0_TGC1_ENDIS_CTRL.U |= 0x2 << ENDIS_CTRL3_BIT_LSB_IDX;   // enable channel 11 on update trigger
404     GTM_TOM0_TGC1_OUTEN_CTRL.U |= 0x2 << OUTEN_CTRL3_BIT_LSB_IDX;   // enable channel 11 output on update trigger
405
406
407     // TOM0 CH11
408     GTM_TOM0_CH11_CTRL.U |= 0x1 << SL_BIT_LSB_IDX;                  // high signal level for duty cycle
409     GTM_TOM0_CH11_CTRL.U |= 0x1 << CLK_SRC_SR_BIT_LSB_IDX;          // clock source --> CMU_FXCLK(1) = 6250 kHz
410     GTM_TOM0_CH11_CTRL.U &= ~(0x0 << OSM_BIT_LSB_IDX);              // continuous mode
411     GTM_TOM0_CH11_CTRL.U &= ~(0x0 << TRIGOUT_BIT_LSB_IDX);
412     GTM_TOM0_CH11_SR0.U = 12500 - 1;                                // PWM freq. = 6250 kHz / 12500 = 500 Hz (temp)
413     GTM_TOM0_CH11_SR1.U = 6250 - 1;                                 // duty cycle = 6250 / 12500 = 50 % (temp)
414
415     // TOUT pin selection
416     GTM_TOUTSEL0.U &= ~(0x3 << SEL3_BIT_LSB_IDX);                   // TOUT3  --> TOM0 channel 11
417 }
418
```

GTM 모듈 설정
- GTM 모듈 enable
- TOM에서 사용하는 FXCLK 주파수 설정
- TOM0의 채널11을 제어하는 TGC1에 대한 설정
- TOM0의 채널11에 대한 설정
- PWM 신호의 주기, Duty Cycle 설정
- PWM 신호가 출력될 핀 설정

SW 프로그래밍
:VADC 모듈에서 변환된 디지털 값 사용 PWM Duty Cycle 제어

초기화 함수 호출 ←

```
//initERU();
initCCU60();
initLED();
initRGBLED();
//initVADC();
initBuzzer();
initGTM();
//initButton();
```

- main 함수 작성

```
GTM_TOM0_TGC1_GLB_CTRL.U |= 0x1 << HOST_TRIG_BIT_LSB_IDX;       // trigger update request s

// from 3 octave 도C ~ 4 octave 도C {도C, 레D, 미E, 파F, 솔G, 라A, 시B, 도C}
unsigned int duty[8] = {130, 146, 164, 174, 195, 220, 246, 262};

while(1)
{

    for(unsigned int i = 0; i < 50000000; i++);
    GTM_TOM0_CH11_SR0.U = (6250000 / duty[0]) - 1;
    GTM_TOM0_CH11_SR1.U = (3125000 / duty[0]) - 1;
```

특정 음계 소리를 내기 위한 PWM 신호의 주기 (Hz)
→ 3옥타브 도C 부터 4옥타브 도C 까지 필요한 주기를 미리 저장

(단위 : Hz)

옥타브 / 음계	1	2	3	4	5	6	7	8
C(도)	32.7032	65.4064	130.8128	261.6256	523.2511	1046.502	2093.005	4186.009
C#	34.6478	69.2957	138.5913	277.1826	554.3653	1108.731	2217.461	4434.922
D(레)	36.7081	73.4162	146.8324	293.6648	587.3295	1174.659	2349.318	4698.636
D#	38.8909	77.7817	155.5635	311.1270	622.2540	1244.508	2489.016	4978.032
E(미)	41.2034	82.4069	164.8138	329.6276	659.2551	1318.510	2637.020	5274.041
F(파)	43.6535	87.3071	174.6141	349.2282	698.4565	1396.913	2793.826	5587.652
F#	46.2493	92.4986	184.9972	369.9944	739.9888	1479.978	2959.955	5919.911
G(솔)	48.9994	97.9989	195.9977	391.9954	783.9909	1567.982	3135.963	6271.927
G#	51.9130	103.8262	207.6523	415.3047	830.6094	1661.219	3322.438	6644.875
A(라)	55.0000	110.0000	220.0000	440.0000	880.0000	1760.000	3520.000	7040.000
A#	58.2705	116.5409	233.0819	466.1638	932.3275	1864.655	3729.310	7458.620
B(시)	61.7354	123.4708	246.9417	493.8833	987.7666	1975.533	3951.066	7902.133

음계마다 필요한 PWM 주기가 미리 정해져 있음
→ 필요한 PWM 주기 선택

SW 프로그래밍
:VADC 모듈에서 변환된 디지털 값 사용
PWM Duty Cycle 제어

6250kHz -> 초당 카운터값 6250000번 증가
➔ SR0 (period) 6250000 설정시 1초 지나면 period match됨, 따라서 주기 1Hz PWM 생성됨

- main 함수 작성

```
GTM_TOM0_TGC1_GLB_CTRL.U |= 0x1 << HOST_TRIG_BIT_LSB_IDX;      // trigger update request s

// from 3 octave 도C ~ 4 octave 도C {도C, 레D, 미E, 파F, 솔G, 라A, 시B, 도C}
unsigned int duty[8] = {130, 146, 164, 174, 195, 220, 246, 262};

while(1)
{
    for(unsigned int i = 0; i < 50000000; i++);
    GTM_TOM0_CH11_SR0.U = (6250000 / duty[0]) - 1;
    GTM_TOM0_CH11_SR1.U = (3125000 / duty[0]) - 1;

    for(unsigned int i = 0; i < 50000000; i++);
    GTM_TOM0_CH11_SR0.U = (6250000 / duty[1]) - 1;
    GTM_TOM0_CH11_SR1.U = (3125000 / duty[1]) - 1;

    for(unsigned int i = 0; i < 50000000; i++);
    GTM_TOM0_CH11_SR0.U = (6250000 / duty[2]) - 1;
    GTM_TOM0_CH11_SR1.U = (3125000 / duty[2]) - 1;

    for(unsigned int i = 0; i < 50000000; i++);
    GTM_TOM0_CH11_SR0.U = (6250000 / duty[3]) - 1;
    GTM_TOM0_CH11_SR1.U = (3125000 / duty[3]) - 1;

    for(unsigned int i = 0; i < 50000000; i++);
    GTM_TOM0_CH11_SR0.U = (6250000 / duty[4]) - 1;
    GTM_TOM0_CH11_SR1.U = (3125000 / duty[4]) - 1;

    for(unsigned int i = 0; i < 50000000; i++);
    GTM_TOM0_CH11_SR0.U = (6250000 / duty[5]) - 1;
    GTM_TOM0_CH11_SR1.U = (3125000 / duty[5]) - 1;

    for(unsigned int i = 0; i < 50000000; i++);
    GTM_TOM0_CH11_SR0.U = (6250000 / duty[6]) - 1;
    GTM_TOM0_CH11_SR1.U = (3125000 / duty[6]) - 1;

    for(unsigned int i = 0; i < 50000000; i++);
    GTM_TOM0_CH11_SR0.U = (6250000 / duty[7]) - 1;
    GTM_TOM0_CH11_SR1.U = (3125000 / duty[7]) - 1;

}
return (1);
```

특정 음계 소리를 내기 위한 PWM 신호의 주기 (Hz) 설정
duty는 50%로 설정

시간 간격으로 음계를 변화시켜가며 부저 작동
도 → 레 → … → 시 → 도

ex) 3옥타브 도C 주기
(6250000 Hz / 130 Hz) - 1

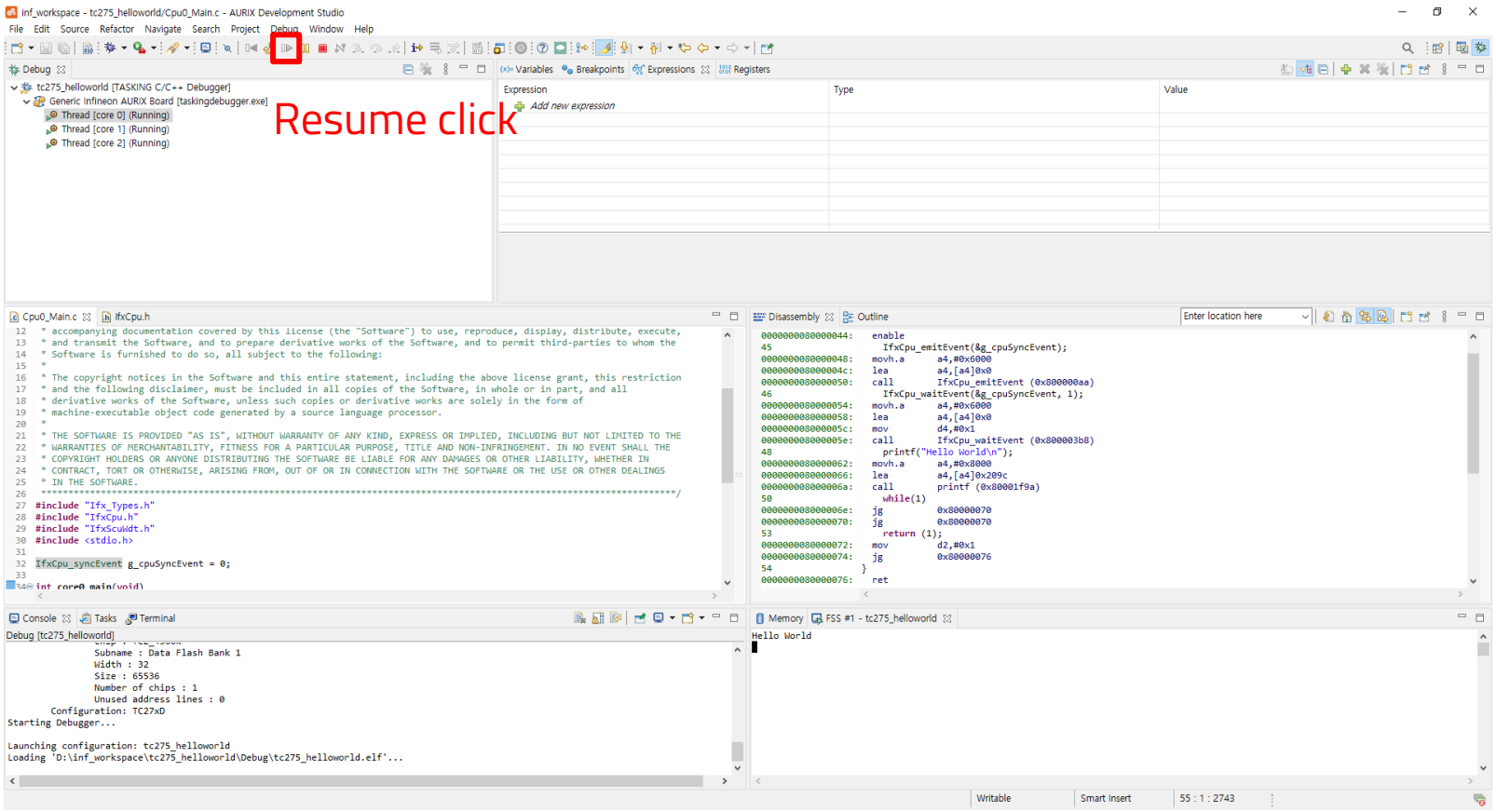
ex) 3옥타브 도C duty
(6250000 Hz / 130 Hz / 2) - 1

→ **SR0 을 130을 나누면 카운터 match값이 1/130로 줄어듦**
→ **130배 빠른 속도로 match**
→ **130Hz 주기 PWM 생성**

SR1은 50%로 해서 duty 50%로 설정

Build 및 Debug

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



동작 확인

- 부저의 소리를 들어보자

감사합니다. 휴식~~