Infineon TC275 PWM (Pulse Width Modulation)

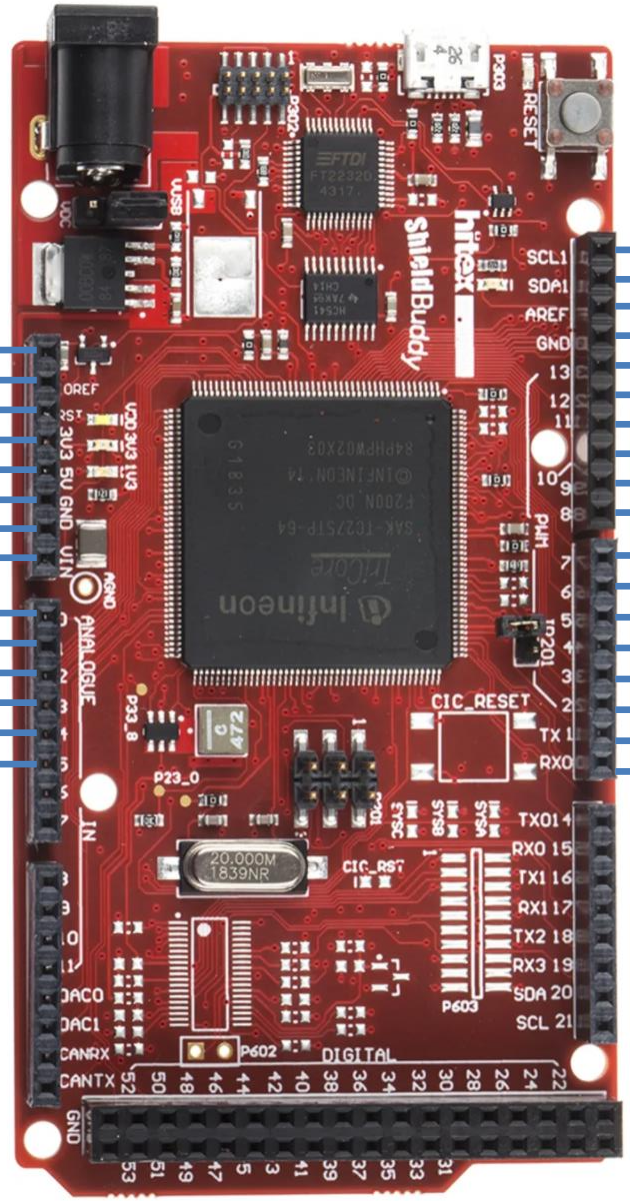
Hyeongrae Kim

Architecture and Compiler for Embedded System LAB.

School of Electronics Engineering, KNU, KOREA

2021-05-11

Hitex ShieldBuddy TC275



NC	NC
IOREF/VDD/5V	IOREF
RESET	Reset
+3.3V	+3V
VDD VIA RES521	+5V
GND	GND
GND	GND
VIN 9..12V	VIN

SAR4_7 AD0	A0|D14
SAR4_6 AD1	A1|D15
SAR4_5 AD2	A2|D16
SAR4_4 AD3	A3|D17
SAR3_1 AD4	A4|D18
SAR3_0 AD5	A5|D19

D19	P13_1 SCL1
D18	P13_2 SDA1
AREF	AREF0
GND	GND
D13	P10_2 SPCK
D12	P10_1 MISO
~D11	P10_3 MOSI
~D10	P10_5 SSO/PWM10
~D9	P02_7 PWM9
D8	P02_6 PWM8

D7	P02_5 PWM7
~D6	P02_4 PWM6
~D5	P02_3 PWM5
D4	P10_4 SS1/PWM4
~D3	P02_1 PWM3
D2	P02_0 PWM2
D1/TX	P15_2 TX
D0/RX	P15_3 RX

ShieldBuddy TC275 Pin Layout	Arduino Uno Pin Layout

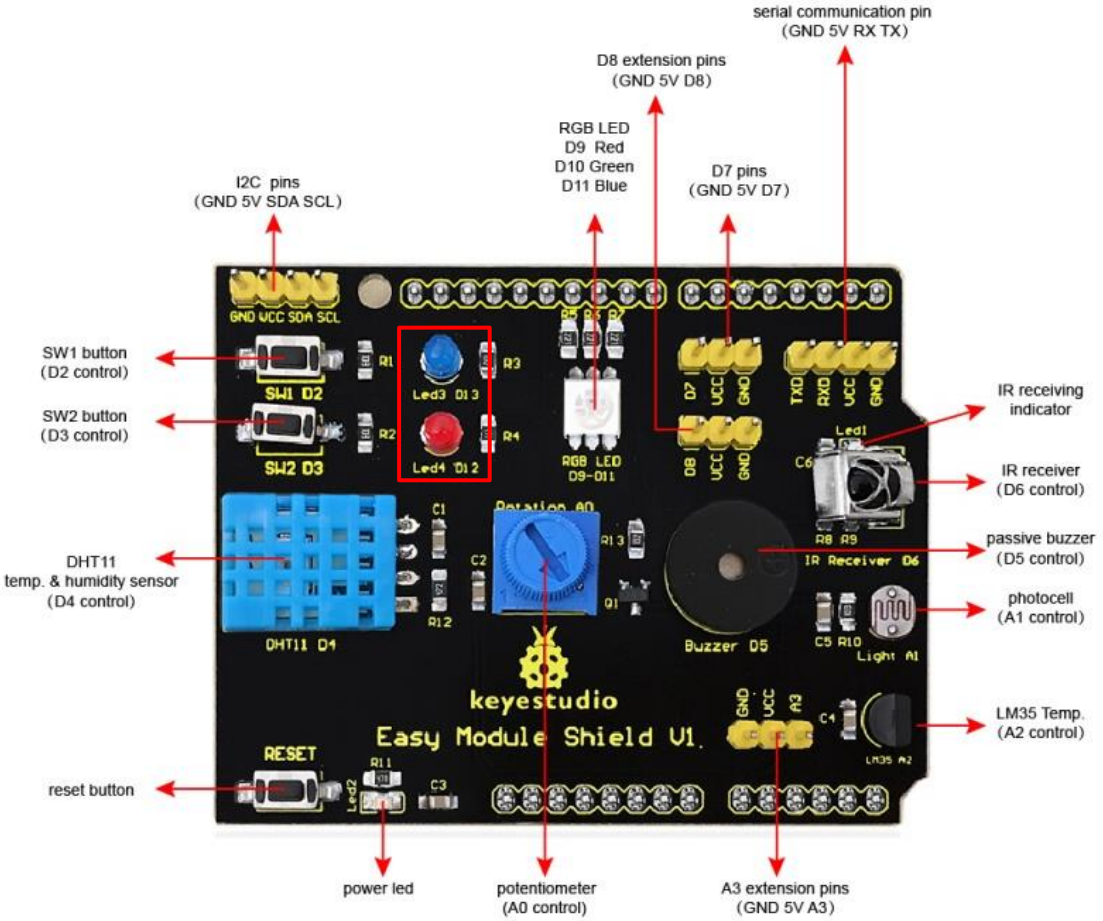
ACE Lab.

PWM Example

- PWM Duty Ratio에 따른 LED 밝기 변화
  1. 새로운 예제를 위한 프로젝트를 생성한다.
  2. 원하는 동작을 위해 레지스터와 메모리에 직접 접근해서 값을 써야한다.
  3. Board Schematic과 Datasheet를 통해 PWM 신호 출력에 대한 정보를 파악한다.
  4. PWM 신호 생성을 위해 사용할 GTM 모듈의 동작 원리를 파악하고 메모리 맵을 분석한다.
  5. 분석 결과를 활용해 임베디드 프로그래밍을 한다.

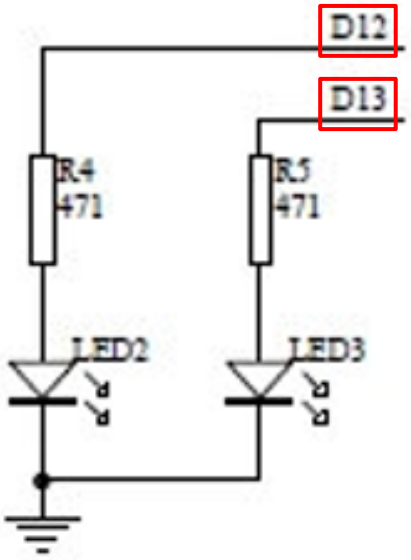
PWM Example

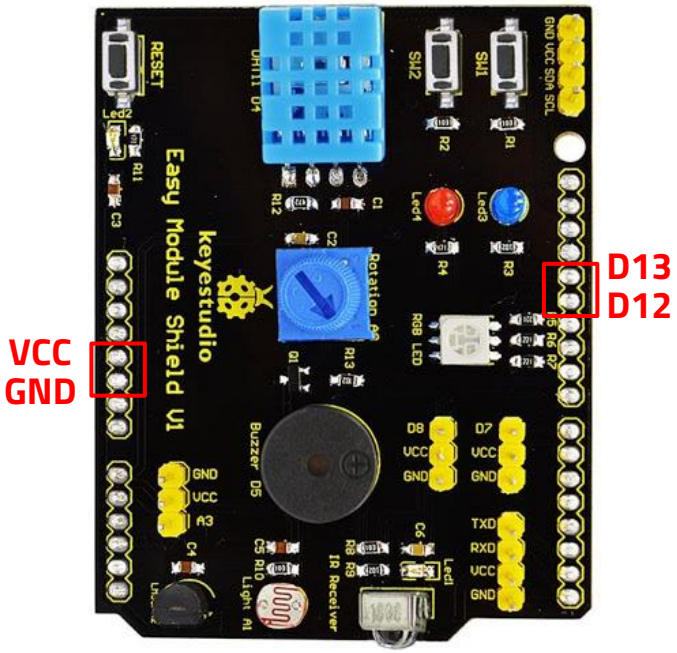
1. LED 연결 정보 파악
   - ✓ 여러 LED를 사용하기 위해 Target Board가 아닌 **Easy Module Shield V1 확장 보드**의 LED를 사용한다.



PWM Example

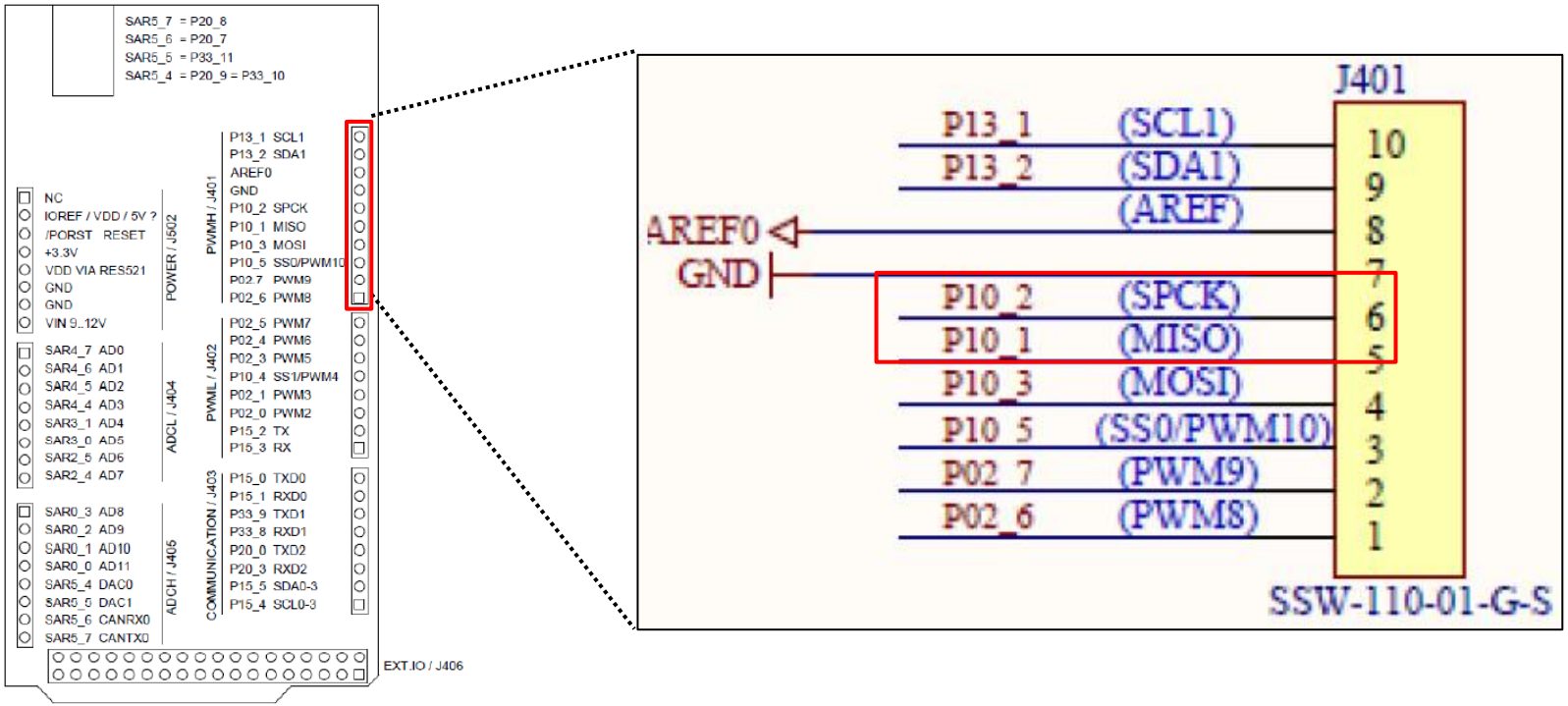
1. LED 연결 정보 파악
   - ✓ LED는 Easy Module Shield V1 확장 보드의 **Pin D12(RED)/D13(BLUE)**과 연결되어 있다.
   - ✓ 타겟 보드는 Easy Module Shield V1 확장 보드의 Pin D12/D13을 통해 LED 출력을 보낼 수 있다.





PWM Example

1. LED 연결 정보 파악
   - ✓ TC275 보드의 Schematic과 Datasheet를 확인했을 때, Easy Module Shield V1 확장 보드의 **Pin D12/D13**과 연결되는 IO는 PORT10**의 Pin 1-2**다.
   - ✓ 해당 Pin의 출력이 High-level 일 때 LED는 켜지고, Low-level 일 때 LED는 꺼진다.



PWM Example

P10.2	I	General-purpose input	P10_IN.P2	P10_IOCR0.PC2	$0XXXX_B$
		GTM input	TIN104		
		QSPI1 input	SCLK1A		
		SCU input	REQ2		
		MSC0 input	SDI01		
		GPT120 input	T6INB		
		CAN node 2 input	RXDCAN2E		
	O	General-purpose output	P10_OUT.P2		$1X000_B$
		GTM output	TOUT104		$1X001_B$
		Reserved	–		$1X010_B$
		QSPI1 output	SCLK1		$1X011_B$

1. PWM 신호 출력 정보 파악
   - ✓ LED가 연결된 PORT10 Pin 1는 GTM 모듈의 TOUT103과 연결되어 있다.
   - ✓ GTM 모듈의 TOUT103이 PWM 신호를 출력하면 PORT10 Pin 1을 통해 LED에 인가될 수 있다.
   - ✓ PWM 신호를 통해 LED 밝기를 제어하기 위해 해당 Pin을 **GTM 모듈의 TOUT103 (O1)**으로 설정해야 한다.

Pin	Symbol	Ctrl	Type	Function
169	P10.1	I	**MP+ / PU1 / VEXT**	General-purpose input
	TIN103			GTM input
	MRST1A			QSPI1 input
	T5EUDB			GPT120 input
	P10.1	O0		General-purpose output
	TOUT103	O1		GTM output
	MTSR1	O2		QSPI1 output
	MRST1	O3		QSPI1 output
	EN01	O4		MSC0 output
	VADCG6BFL1	O5		VADC output
	END03	O6		MSC0 output
	–	O7		Reserved

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

Port	Input	Output	Input Timer Mapped		Output Timer Mapped			
			A	B	A	B	C	D
P02.2	TIN2	TOUT2	TIM0_2	TIM1_2	TOM0_10	TOM1_10	ATOM 0_2	ATOM 1_2
P02.3	TIN3	TOUT3	TIM0_3	TIM1_3	TOM0_11	TOM1_11	ATOM 0_3	ATOM 1_3
P02.4	TIN4	TOUT4	TIM0_4	TIM1_4	TOM0_12	TOM1_12	ATOM 0_4	ATOM 1_4
P02.5	TIN5	TOUT5	TIM0_5	TIM1_5	TOM0_13	TOM1_13	ATOM 0_5	ATOM 1_5
P02.6	TIN6	TOUT6	TIM0_6	TIM1_6	TOM0_14	TOM1_14	ATOM 0_6	ATOM 1_6
P02.7	TIN7	TOUT7	TIM0_7	TIM1_7	TOM0_15	TOM1_15	ATOM 0_7	ATOM 1_7
P02.8	TIN8	TOUT8	TIM2_0	TIM3_0	TOM0_8	TOM1_0	ATOM 0_0	ATOM 1_0
P10.0	TIN102	TOUT102	TIM0_4	TIM1_4	TOM0_4	TOM2_12	ATOM 1_4	ATOM 4_4
P10.1	TIN103	TOUT103	TIM0_1	TIM1_1	TOM0_1	TOM2_9	ATOM 1_1	ATOM 4_1
P10.2	TIN104	TOUT104	TIM0_2	TIM1_2	TOM0_2	TOM2_10	ATOM 1_2	ATOM 4_2

PWM Example

2. Data sheet 분석 : PORT 설정 (1)

   ✓ P10_IOCR Register는 PORT10의 Input/Output을 설정한다.

   ✓ LED가 PORT10의 Pin 1에 연결되어 있기 때문에 **P10_IOCR0 Register의 PC1 bits**를 설정한다.

Table 13-3 Registers Address Space

Module	Base Address	End Address	Note
P00	$F003\ A000_H$	$F003\ A0FF_H$	13 pins
P01	$F003\ A100_H$	$F003\ A1FF_H$	5 pins
P02	$F003\ A200_H$	$F003\ A2FF_H$	12 pins
P10	$F003\ B000_H$	$F003\ B0FF_H$	9 pins
P11	$F003\ B100_H$	$F003\ B1FF_H$	16 pins
P12	$F003\ B200_H$	$F003\ B2FF_H$	2 pins
P13	$F003\ B300_H$	$F003\ B3FF_H$	4 pins
P14	$F003\ B400_H$	$F003\ B4FF_H$	11 pins
P15	$F003\ B500_H$	$F003\ B5FF_H$	9 pins

P10_IOCR0 Register 주소: F003_B010h (F003B000h + 10h)

P10_IOCR0 Register 구조:

Pn_IOCR0 (n=10-11)
Port n Input/Output Control Register 0
($F003\ A610_H + n^*100_H$) **Reset Value: $1010\ 1010_H$**

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.

PWM Example

2. Data sheet 분석 : PORT 설정 (2)

   ✓ PORT10의 Pin 1을 GTM 모듈의 TOUT103 (01)으로 설정하기 위해 **PC1 bits**를 **10001b**로 설정한다.

Table 13-5 PCx Coding

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

PWM Example

2. Data sheet 분석 : GTM Enable 설정
   - ✓ GTM_CLC Register는 GTM 모듈의 Enable 설정을 한다.
   - ✓ GTM 모듈을 Enable 하기 위해 **DISR bit**를 **0**으로 설정한다.
   - ✓ GTM 모듈이 Enable 되어 있는지 확인하기 위해 **DISS bit**가 **0**인지 확인한다.

GTM_CLC Register 주소: F019_FD00h (F0100000h + 9FD00h)

GTM_CLC Register 구조:

Table 25-63 Registers Address Space

Module	Base Address	End Address	Note
GTM	$F010\ 0000_H$	$F019\ FFFF_H$	

CLC
Clock Control Register **($9FD00_H$)** **Reset Value: $0000\ 0003_H$**

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0												EDIS		DISS	DISR
r												rw	r	r	rw

Field	Bits	Type	Description
DISR	0	rw	**Module Disable Request Bit** Used for enable/disable control of the GTM module. 0_B No disable requested 1_B Disable requested
DISS	1	r	**Module Disable Status Bit** Bit indicates the current status of the GTM module. 0_B GTM module is enabled 1_B GTM module is disabled
EDIS	3	rw	**Sleep Mode Enable Control** Used for module sleep mode control.
0	2, [31:4]	r	**Reserved** Read as 0; should be written with 0.

PWM Example

2. Data sheet 분석 : System Critical Register 설정 (1)
   - ✓ 설정해야 하는 GTM_CLC Register는 System Critical Register이기 때문에 Write Protected (System ENDINIT, End-of-Initialization) 되어 있다.
   - ✓ 해당 Register를 수정하기 위해서는 System ENDINIT을 해제해야 한다.
   - ✓ SCU_WDTCPU0CON0 Register는 **System Critical Register**의 **System ENDINIT을 설정/해제**한다.

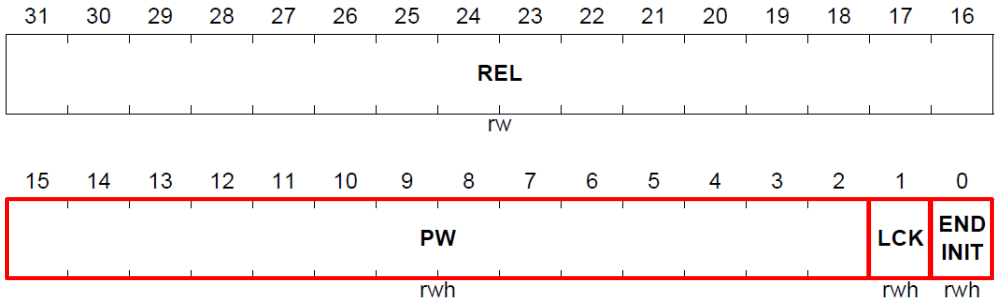
SCU_WDTCPU0CON0 Register 주소: F003_6100h
(F0036000h + 100h)

SCU_WDTCPU0CON0 Register 구조:

Table 7-27 Registers Address Spaces - SCU Kernel Registers

Module	Base Address	End Address	Note
SCU	$F003\ 6000_H$	$F003\ 63FF_H$	-

WDTCPU0CON0
CPU0 WDT Control Register 0 (100_H) Reset Value: $FFFC\ 000E_H$



PWM Example

2. Data sheet 분석 : System Critical Register 설정 (2)
   - ✓ **ENDINIT bit**는 System ENDINIT의 설정 상태를 나타내며 Modify Access를 통해서만 수정이 가능하다.
   - ✓ **LCK bit**는 SCU_WDTCPU0CON0 Register의 Lock 상태를 나타내며 해당 Register의 Lock 상태는 Password Access를 통해 Unlock 되고, Modify Access를 통해 Lock 된다.
   - ✓ **PW bits**는 SCU_WDTCPU0CON0 Register에 접근하기 위한 Password를 저장하며 해당 값을

Field	Bits	Type	Description
ENDINIT	0	rwh	**End-of-Initialization Control Bit** 0_B Access to Endinit-protected registers is permitted. 1_B Access to Endinit-protected registers is not permitted. This bit must be written with a '1' during a Password Access or Check Access (although this write is only used for the password-protection mechanism and is not stored). This bit must be written with the required ENDINIT update value during a Modify Access.
LCK	1	rwh	**Lock Bit to Control Access to WDTxCON0** 0_B Register WDTxCON0 is unlocked 1_B Register WDTxCON0 is locked (default after ApplicationReset) The current value of LCK is controlled by hardware. It is cleared after a valid Password Access to WDTxCON0 when WDTxSR.US is 0 (or when WDTxSR.US is 1 and the SMU is in RUN mode), and it is automatically set again after a valid Modify Access to WDTxCON0. During a write to WDTxCON0, the value written to this bit is only used for the password-protection mechanism and is not stored. This bit must be cleared during a Password Access to WDTxCON0, and set during a Modify Access to WDTxCON0. A Check Access does not clear LCK.

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. If corresponding WDTxSR.PAS = 0 then this bit field must be written with its current contents during a Password Access or Check Access. If corresponding WDTxSR.PAS = 1 then this bit field must be written with the next password in the LFSR sequence during a Password Access or Check Access The default password after Application Reset is 00000000111100_B A-step silicon: Bits [7:2] must be written with 111100_B during Password Access and Modify Access. Read returns 000011_B for these bits.

PWM Example

2. Data sheet 분석 : System Critical Register 설정 (3)
   - ✓ SCU_WDTCPU0CON0 Register에 적절한 값을 Write하여 **Password Access**를 수행한다.
   - ✓ **Password Access**는 **SCU_WDTCPU0CON0 Register의 Lock 상태를 해제**하며 과정은 다음과 같다.
     1. SCU_WDTCPU0CON0 Register의 값을 읽어 REL bits, PW bits를 파악한다.
     2. Bits[7:2] (PW bits의 일부)가 반전되어 읽히기 때문에 이를 반전시켜 정확한 PW bits를 얻는다.
     3. Write 할 값의 bits[31:16]은 읽혀진 REL bits 값으로 설정하고 bit[15:2]는 앞서 구한 정확한 PW bits 값으로 설정한다.
     4. Write 할 값의 bit[1]은 0으로 설정하고, bit[0]은 1로 설정한다.
     5. 설정된 값을 SCU_WDTCPU0CON0 Register에 한번에 쓴다.
     6. SCU_WDTCPU0CON0 Register의 LCK bit를 확인하여 Lock 상태가 해제되었는지 파악한다. (Password Access가 정상적으로 수행되면 Lock 상태가 해제되며 LCK bit가 0으로 설정된다.)
   - ✓ Password Access를 통해 SCU_WDTCPU0CON0 Register의 Lock 상태가 해제되면 Modify Access를 통해 System ENDINIT을 설정/해제할 수 있다.

PWM Example

2. Data sheet 분석 : System Critical Register 설정 (4)

✓ SCU_WDTCPU0CON0 Register에 적절한 값을 Write하여 **Modify Access**를 수행한다.

✓ **Modify Access**는 **System ENDINIT을 설정/해제**하며 과정은 다음과 같다.

1. SCU_WDTCPU0CON0 Register의 값을 읽어 REL bits, PW bits를 파악한다.
2. Bits[7:2] (PW bits의 일부)가 반전되어 읽히기 때문에 이를 반전시켜 정확한 PW bits를 얻는다.
3. Write 할 값의 bits[31:16]은 읽혀진 REL bits 값으로 설정하고 bit[15:2]는 앞서 구한 정확한 PW bits 값으로 설정한다.
4. Write 할 값의 bit[1]은 1로 설정하고, bit[0]은 적절한 값으로 설정한다.
   (System ENDINIT 설정: bit[0] = 1, System ENDINIT 해제 : bit[0] = 0)
5. 설정된 값을 SCU_WDTCPU0CON0 Register에 한번에 쓴다.
6. SCU_WDTCPU0CON0 Register의 LCK bit를 확인하여 Lock 상태가 다시 설정되었는지 파악한다.
   (Modify Access가 정상적으로 수행되면 Lock 상태가 설정되며 LCK bit가 1로 설정된다.)

✓ Modify Access를 통해 System ENDINIT을 해제하면 System Critical Register를 수정할 수 있으며 수정을 완료하면 System ENDINIT을 꼭 다시 설정해야 한다.

PWM Example

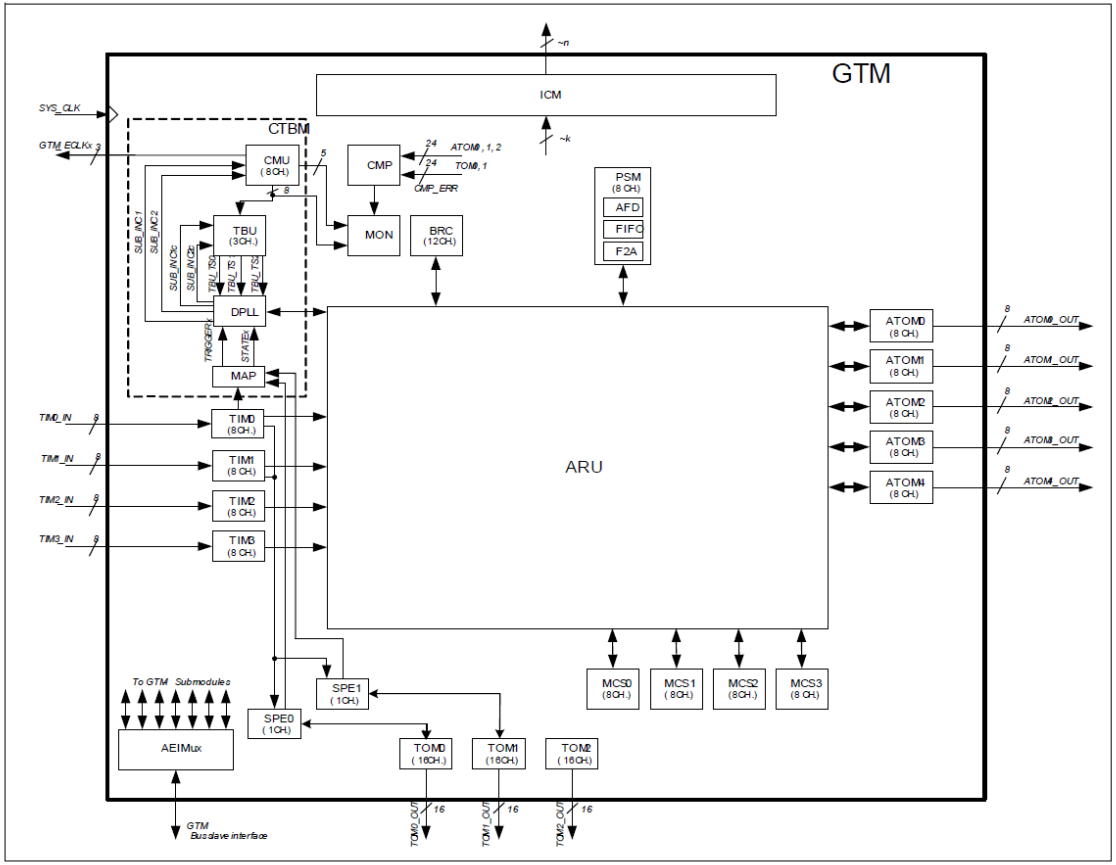


Figure 25-1 GTM Architecture Block Diagram

ACE Lab.

Table 25-1 Submodule groups

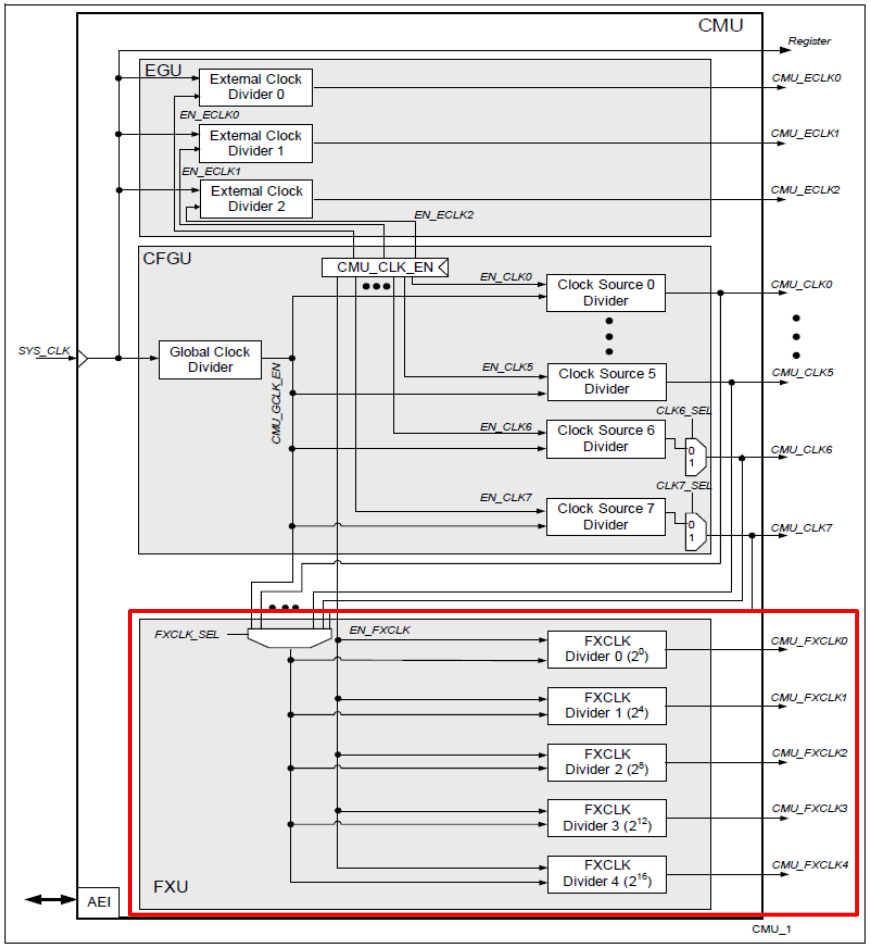
Section	Submodule	Group
Section 25.3	Advanced Routing Unit (ARU)	Infrastructural components
Section 25.4	Broadcast Module (BRC)	Infrastructural components
Section 25.5	First In First Out Module (FIFO)	Infrastructural components
Section 25.6	AEI-to-FIFO Data Interface (AFD)	Infrastructural components
Section 25.7	FIFO-to-ARU Interface (F2A)	Infrastructural components
Section 25.8	Clock Management Unit (CMU)	Infrastructural components
Section 25.9	Time Base Unit (TBU)	Infrastructural components
Section 25.10	Timer Input Module (TIM)	IO Modules
Section 25.11	Timer Output Module (TOM)	IO Modules
Section 25.12	ARU-connected Timer Output Module (ATOM)	IO Modules
Section 25.13	Multi Channel Sequencer (MCS)	Signal generation and processing

Table 25-1 Submodule groups (cont'd)

Section	Submodule	Group
Section 25.14	Memory Configuration Module (MCFG)	Infrastructural component for MCS
Section 25.15	TIM0 Input Mapping Module (MAP)	Dedicated
Section 25.16	Digital PLL (DPLL)	Dedicated
Section 25.17	Sensor Pattern Evaluation Module (SPE)	BLDC support
Section 25.18	Interrupt Concentrator Module (ICM)	Interrupt services
Section 25.19	Output Compare Unit (CMP)	Safety features
Section 25.20	Monitoring Unit (MON)	Safety features

PWM Example

2. Data sheet 분석 : GTM 내부 Clock 설정 (1)
   - ✓ GTM 모듈은 내부에 CMU (Clock Management Unit)을 포함하고 있다.
   - ✓ CMU는 GTM 입력 클럭을 분주하여 다양한 내부 클럭을 생성하고, GTM 내부의 하위 모듈에 공급한다.
   - ✓ 본 실습에서 PWM 신호 생성을 위해 사용할 하위 모듈인 **TOM (Timer Output Module)**은 **CMU_FXCLK**에 따라 동작한다.
   - ✓ 따라서, CMU의 **FXU에 대한 설정**을 해야 한다.



PWM Example

2. Data sheet 분석 : GTM 내부 Clock 설정 (2)
   - ✓ GTM_CMU_FXCLK_CTRL Register는 CMU_FXCLK의 소스 클럭을 설정한다.
   - ✓ CMU_FXCLK의 소스 클럭으로 GTM 모듈의 입력 클럭인 CMU_GCLK_EN 또는 GTM 모듈 내부에서 생성된 CMU_CLKx가 사용될 수 있다.
   - ✓ 소스 클럭을 CMU_GCLK_EN으로 설정하기 위해 **FXCLK_SEL bits**를 **0000b**로 설정한다.

GTM_CMU_FXCLK_CTRL Register 주소: F010_0344h
(F0100000h + 344h)

GTM_CMU_FXCLK_CTRL Register 구조:

Table 25-63 Registers Address Space

Module	Base Address	End Address	Note
GTM	$F010\ 0000_H$	$F019\ FFFF_H$	

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 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 FXCLK_SEL = 0.*

PWM Example

2. Data sheet 분석 : GTM 내부 Clock 설정 (3)

✓ GTM_CMU_CLK_EN Register는 CMU 내부의 클럭에 대한 Enable 설정을 한다.

✓ GTM_CMU_CLK_EN Register는 CMU 내부에서 생성된 다양한 클럭에 대한 Enable을 설정할 수 있다.

✓ CMU_FXCLK을 Enable 하기 위해 **EN_FXCLK bits**를 **10b**로 설정한다.

GTM_CMU_CLK_EN Register 주소: F010_0300h
(F0100000h + 300h)

GTM_CMU_CLK_EN Register 구조:

Table 25-63 Registers Address Space

Module	Base Address	End Address	Note
GTM	$F010\ 0000_H$	$F019\ FFFF_H$	

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	

Field	Bits	Type	Description
EN_CLK4	[9:8]	rw	**Enable clock source 4** see bits [1:0]
EN_CLK5	[11:10]	rw	**Enable clock source 5** see bits [1:0]
EN_CLK6	[13:12]	rw	**Enable clock source 6** see bits [1:0]
EN_CLK7	[15:14]	rw	**Enable clock source 7** see bits [1:0]
EN_ECLK0	[17:16]	rw	**Enable ECLK 0 generation subunit** see bits [1:0]
EN_ECLK1	[19:18]	rw	**Enable ECLK 1 generation subunit** see bits [1:0]
EN_ECLK2	[21:20]	rw	**Enable ECLK 2 generation subunit** see bits [1:0]
EN_FXCLK	[23:22]	rw	**Enable all CMU_FXCLK** see bits [1:0] *Note: An enable reset inte*

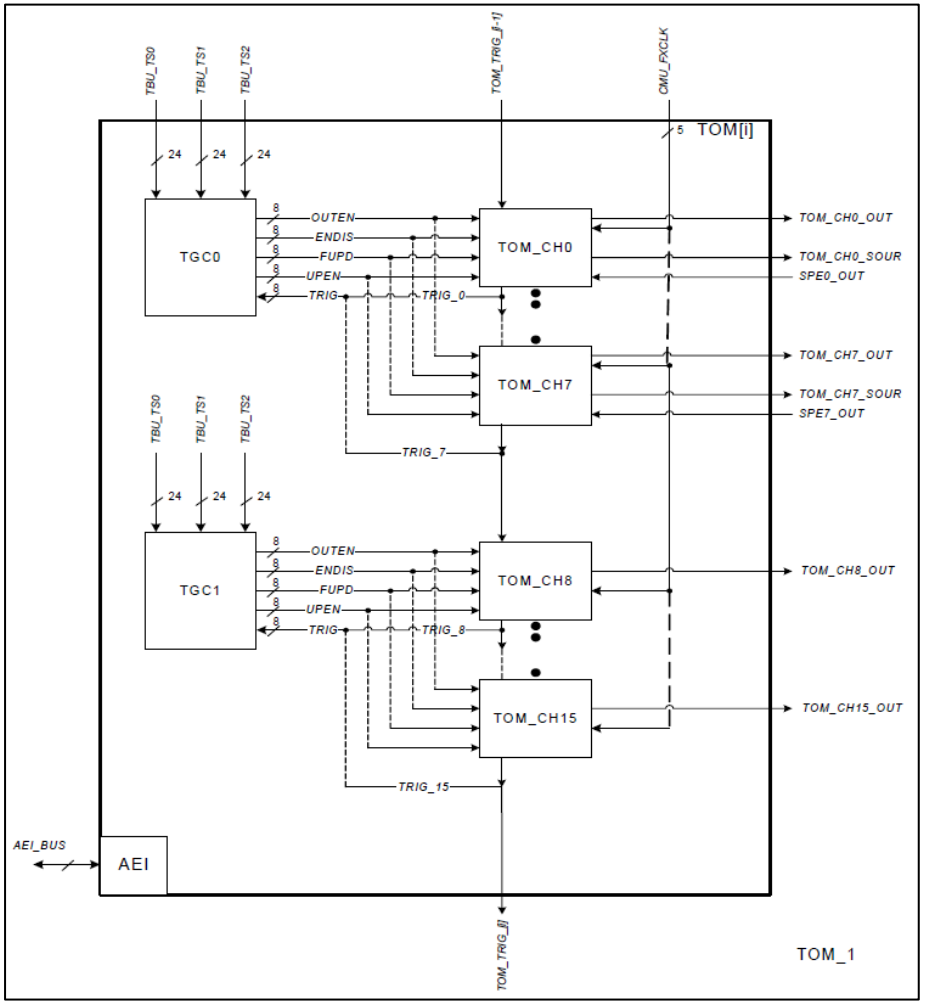
00_B clock source is disabled (ignore write access)
01_B disable clock signal and reset internal states
10_B enable clock signal
11_B clock signal enabled (ignore write access)

PWM Example

P10.0	TIN102	TOUT102	TIM0_4	TIM1_4	TOM0_4	TOM2_12	ATOM 1_4	ATOM 4_4
P10.1	TIN103	TOUT103	TIM0_1	TIM1_1	TOM0_1	TOM2_9	ATOM 1_1	ATOM 4_1
P10.2	TIN104	TOUT104	TIM0_2	TIM1_2	TOM0_2	TOM2_10	ATOM 1_2	ATOM 4_2

2. Data sheet 분석 : TOM 구조 분석

- ✓ PWM 신호 생성을 위해 GTM 모듈 내부의 TOM을 사용한다.
- ✓ GTM 모듈은 3개의 TOM을 포함하고 있고, 각 TOM은 2개의 TGC (TOM Global Channel Control)와 16개의 TOM Channel을 가지고 있다.
- ✓ **TGC**는 8개의 TOM Channel과 연결되어 있으며 이를 통해 **TOM Channel을 제어**할 수 있다.
- ✓ **TOM Channel**은 TGC의 제어에 따라 동작을 수행하며 **출력 신호를 생성**한다.
- ✓ 본 실습에서는 **TOM0_CH1**를 사용한다. (TOUT103과 연결되어 있기 때문이다.)
- ✓ 따라서, **TOM0_CH1를 사용하기 위한 설정**을 수행한다.



PWM Example

2. Data sheet 분석 : TOM 동작 분석

- ✓ TOM Channel은 **CN0 / CM0 / CM1**을 사용해 **PWM 신호를 생성**한다.
  - ✓ CN0 : 동작 클럭에 따라 증가하는 Count 값을 저장한다.
  - ✓ CM0 : PWM 신호의 주기를 결정하는 값을 저장한다.
  - ✓ CM1 : PWM 신호의 Duty Ratio를 결정하는 값을 저장한다.
- ✓ CN0는 동작 클럭에 따라 1씩 증가하며 CM0에 도달하면 0으로 초기화된다.
- ✓ CN0가 CM0에 도달했을 때, 출력 신호는 SL 값으로 설정된다.
- ✓ CN0가 CM1에 도달했을 때, 출력 신호는 SL 반전 값으로 설정된다.

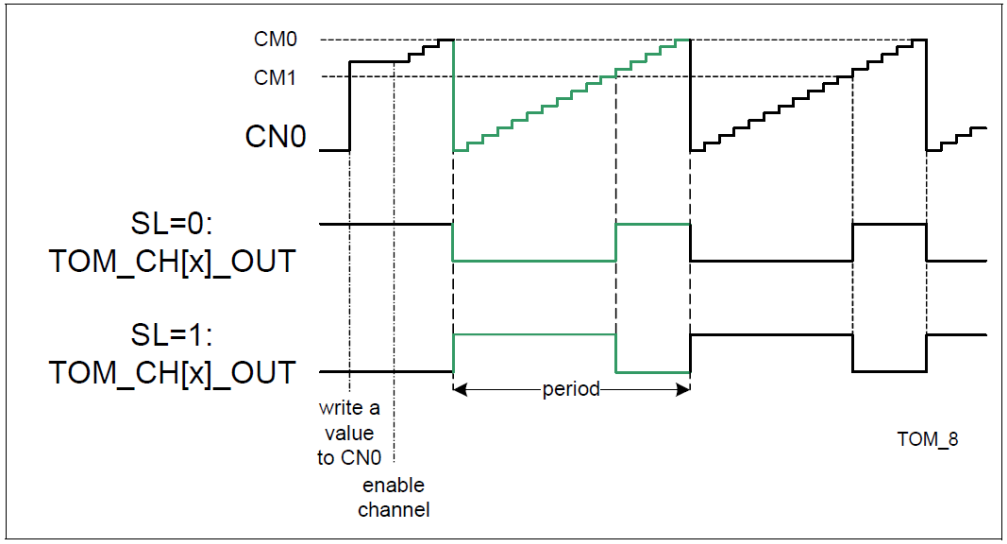


Figure 25-39 PWM Output with respect to configuration bit SL in continuos mode

PWM Example

2. Data sheet 분석 : TOM0 – TGC0 설정 (1), TOM Global Control
   - ✓ GTM_TOM0_TGC0_GLB_CTRL Register는 Channel 0-7을 제어하는 TGC0에 대한 설정을 한다.
   - ✓ Channel에 대한 Enable/Disable 설정 및 Output Enable 설정은 트리거 신호에 의해 일괄적으로 반영된다.
   - ✓ **HOST_TRIG bit를 1로 설정하여 사용자가 소프트웨어적으로 트리거 신호를 발생시킬 수 있다.**

GTM_TOM0_TGC0_GLB_CTRL Register 주소: F010_8030h (F0100000h + 8030h)

GTM_TOM0_TGC0_GLB_CTRL Register 구조:

Table 25-63 Registers Address Space

Module	Base Address	End Address	Note
GTM	$F010\ 0000_H$	$F019\ FFFF_H$	

GTM_TOMi_TGC0_GLB_CTRL (i=0-2)
TOMi TGC0 Global Control Register(08030_H+i*800_H) **Reset Value: 00000000_H**

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

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
Reserved	[7:1]	r	**Reserved** Read as zero, should be written as zero
RST_CH0	8	w	**Software reset of channel 0** 0_B No action 1_B Reset channel Read as 0. *Note: This bit is cleared automatically after write by CPU. The channel registers are set to their reset values and channel operation is stopped immediately. The S-r FlipFlop SOUR is set to '1'.*

PWM Example

2. Data sheet 분석 : TOM0 – TGC0 설정 (2)
   - ✓ TOM 동작을 위한 CM0 / CM1 / CLK_SRC 값은 먼저 Shadow Register에 저장된다.
   - ✓ 업데이트가 Enable 되어 있으면 업데이트를 할 때 Shadow Register에 저장되어 있는 값이 일괄적으로 반영되어 CM0 / CM1 / CLK_SRC가 설정된다.
   - ✓ TOM Channel 1이 동작하기 위해서는 해당 Channel에 대한 CM0 / CM1 / CLK_SRC 값이 설정되어야 하며 이를 위해 **UPEN_CTRL1 bits**를 **10b**로 설정하여 업데이트를 Enable 한다.

GTM_TOMi_TGC0_GLB_CTRL (i=0-2)
TOMi TGC0 Global Control Register(08030_H+i*800_H) **Reset Value: 00000000_H**

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

UPEN_CTRL1	[19:18]	rw	**TOM channel 1 enable update of register CM0, CM1 and CLK_SRC** See bits 17:

Write of following double bit values is possible:
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
Read of following double values means:
00_B channel disabled
11_B channel enabled

PWM Example

2. Data sheet 분석 : TOM0 – TGC0 설정 (3)
   - ✓ GTM_TOM0_TGC0_FUPD_CTRL Register는 트리거 신호에 따른 동작 설정을 한다.
   - ✓ FUPD_CTRLx bits는 CM0 / CM1 / CLK_SRC의 업데이트가 트리거 신호에 의해 실행되도록 설정하며 이를 Channel 1에 적용하기 위해 **FUPD_CTRL1 bits**를 **10b**로 설정한다.
   - ✓ RSTCN0_CHx bits는 CN0의 초기화가 트리거 신호에 의해 실행되도록 설정하며 이를 Channel 1에 적용하기 위해 **RSTCN0_CH1 bits**를 **10b**로 설정한다.

GTM_TOM0_TGC0_FUPD_CTRL Register 주소: F010_8038h (F0100000h + 8038h)

GTM_TOM0_TGC0_FUPD_CTRL Register 구조:

Table 25-63 Registers Address Space

Module	Base Address	End Address	Note
GTM	$F010\ 0000_H$	$F019\ FFFF_H$	

GTM_TOMi_TGC0_FUPD_CTRL (i=0-2)
TOMi TGC0 Force Update Control Register

($08038_H+i*800_H$) Reset Value: 00000000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSTCN0_CH7		RSTCN0_CH6		RSTCN0_CH5		RSTCN0_CH4		RSTCN0_CH3		RSTCN0_CH2		RSTCN0_CH1		RSTCN0_CH0	
rw		rw		rw		rw		rw		rw		rw		rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FUPD_CTRL7		FUPD_CTRL6		FUPD_CTRL5		FUPD_CTRL4		FUPD_CTRL3		FUPD_CTRL2		FUPD_CTRL1		FUPD_CTRL0	
rw		rw		rw		rw		rw		rw		rw		rw	

FUPD_CTRL1	[3:2]	rw	**Force update of (A)TOM channel 1 operation registers** See bits 1

Write of following double bit values is possible:
- 00_B don't care, bits 1:0 will not be changed
- 01_B force update disabled: is read as 00 (see below)
- 10_B force update enabled: is read as 11 (see below)
- 11_B don't care, bits 1:0 will not be changed

Read of following double values means:
- 00_B force update disabled
- 11_B force channel enabled

RSTCN0_CH1	[19:18]	rw	**Reset CN0 of channel 1 on force update event** See bits 1

Write of following double bit values is possible:
- 00_B don't care, bits 1:0 will not be changed
- 01_B CN0 is not reset on forced update: is read as 00 (see below)
- 10_B CN0 is reset on forced update: is read as 11 (see below)
- 11_B don't care, bits 1:0 will not be changed

Read of following double values means:
- 00_B CN0 is not reset on forced update
- 11_B CN0 is reset on forced update

PWM Example

2. Data sheet 분석 : TOM0 – TGC0 설정 (4)
   - ✓ GTM_TOM0_TGC0_ENDIS_CTRL Register는 트리거 신호에 따른 Enable/Disable을 설정한다.
   - ✓ 트리거 신호에 따라 각 Channel을 Enable 할지 Disable 할지 설정할 수 있다.
   - ✓ 트리거 신호 발생 시, Channel 1가 Enable 되게 **ENDIS_CTRL1 bits**를 **10b**로 설정한다.

GTM_TOM0_TGC0_ENDIS_CTRL Register 주소: F010_8070h
(F0100000h + 8070h)

GTM_TOM0_TGC0_ENDIS_CTRL Register 구조:

Table 25-63 Registers Address Space

Module	Base Address	End Address	Note
GTM	$F010\ 0000_H$	$F019\ FFFF_H$	

GTM_TOMi_TGC0_ENDIS_CTRL (i=0-2)
TOMi TGC0 Enable/Disable Control Register

(08070_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_CTRL7		ENDIS_CTRL6		ENDIS_CTRL5		ENDIS_CTRL4		ENDIS_CTRL3		ENDIS_CTRL2		ENDIS_CTRL1		ENDIS_CTRL0	
rw		rw		rw		rw		rw		rw		rw		rw	

ENDIS_CTRL1	[3:2]	rw	**(A)TOM channel 1 enable/disable update value**

See If a TOM channel is disabled, the counter CN0 is stopped and the FlipFlop SOUR is set to the inverse value of control bit SL. On an enable event, the counter CN0 starts counting from its current value.
Write of following double bit values is possible:
00_B don't care, bits 1:0 of register ENDIS_STAT will 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
Note: if the channel is disabled (ENDIS[0]=0) or the output is disabled (OUTEN[0]=0), the TOM channel 0 output TOM_OUT[0] is the inverted value of bit SL.

PWM Example

2. Data sheet 분석 : TOM0 – TGC0 설정 (5)
   - ✓ GTM_TOM0_TGC0_OUTEN_CTRL Register는 트리거 신호에 따른 Output Enable을 설정한다.
   - ✓ 트리거 신호에 따라 각 Channel의 Output을 Enable 할지 Disable 할지 설정할 수 있다.
   - ✓ 트리거 신호 발생 시, Channel 1의 Output이 Enable 되게 **OUTEN_CTRL1 bits**를 **10b**로 설정한다.

GTM_TOM0_TGC0_OUTEN_CTRL Register 주소: F010_8078h
(F0100000h + 8078h)

GTM_TOM0_TGC0_OUTEN_CTRL Register 구조:

Table 25-63 Registers Address Space

Module	Base Address	End Address	Note
GTM	$F010\ 0000_H$	$F019\ FFFF_H$	

GTM_TOMi_TGC0_OUTEN_CTRL (i=0-2)
TOMi TGC0 Output Enable Control Register
(08078_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	

Field	Bits	Type	Description
OUTEN_CTRL0	[1:0]	rw	**Output (A)TOM_OUT(0) enable/disable update value** Write of following double bit values is possible: 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 Note: if the channel is disabled (ENDIS[0]=0) or the output is disabled (OUTEN[0]=0), the TOM channel 0 output TOM_OUT[0] is the inverted value of bit SL.
OUTEN_CTRL1	[3:2]	rw	**Output (A)TOM_OUT(1)enable/disable update value** See bits 1:0
OUTEN_CTRL2	[5:4]	rw	**Output (A)TOM_OUT(2) enable/disable update value** See bits 1:0
OUTEN_CTRL3	[7:6]	rw	**Output (A)TOM_OUT(3) enable/disable update value** See bits 1:0
OUTEN_CTRL4	[9:8]	rw	**Output (A)TOM_OUT(4) enable/disable update value** See bits 1:0
OUTEN_CTRL5	[11:10]	rw	**Output (A)TOM_OUT(5) enable/disable update value** See bits 1:0

PWM Example

25.11.8.9 Register TOMi_CHx_CTRL (x:0…14)

GTM_TOM0_CHx_CTRL (x=0-14)
TOM0 Channel x Control Register'
(08000_H+x*0040_H) Reset Value: 00000800_H
GTM_TOM1_CHx_CTRL (x=0-14)
TOM1 Channel x Control Register'
(08800_H+x*0040_H) Reset Value: 00000800_H
GTM_TOM2_CHx_CTRL (x=0-14)
TOM2 Channel x Control Register'
(09000_H+x*0040_H) Reset Value: 00000800_H

2. Data sheet 분석 : TOM0 – Channel 1 설정 (1)
   - ✓ GTM_TOM0_CHx_CTRL Register는 TOM0의 각 Channel에 대한 동작 설정을 한다.
   - ✓ TOM Channel 1의 동작을 설정하기 위해 **GTM_TOM0_CH1_CTRL Register**를 설정한다.
   - ✓ 출력 신호의 Duty Cycle에 대한 Signal level을 High로 설정하기 위해 **SL bit**를 **1**로 설정한다.

GTM_TOM0_CH1_CTRL Register 주소: F010_8040h
(F0100000h + 8040h)

GTM_TOM0_CH1_CTRL Register 구조:

Table 25-63 Registers Address Space

Module	Base Address	End Address	Note
GTM	$F010\ 0000_H$	$F019\ FFFF_H$	

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.

PWM Example

2. Data sheet 분석 : TOM0 – Channel 1 설정 (2)

- ✓ TOM Channel 1의 동작 클럭을 CMU_FXCLK1로 설정하기 위해 **CLK_SRC_SR bits**를 **001b**로 설정한다.
- ✓ CMU_FXCLK1의 주파수는 100MHz / 16 = 6,250kHz 이다.
- ✓ CLK_SRC_SR bits가 업데이트를 할 때 반영되기 때문에 TOM Channel 1의 동작 클럭 또한 업데이트를 할 때 반영된다.

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										

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.

PWM Example

2. Data sheet 분석 : TOM0 – Channel 1 설정 (3)

- ✓ GTM_TOM0_CHx_SR0 Register는 CM0에 대한 Shadow Register이다.
- ✓ TOM Channel 1의 CM0를 설정하기 위해 **GTM_TOM0_CH1_SR0 Register**를 설정한다.
- ✓ GTM_TOM0_CH1_SR0 Register에 설정할 CM0 값을 저장하면 업데이트를 할 때 CM0에 반영된다.
- ✓ 본 실습에서는 PWM 신호의 주기를 2ms로 설정하기 위해 해당 Register의 값을 **(12500 – 1)**로 설정한다.

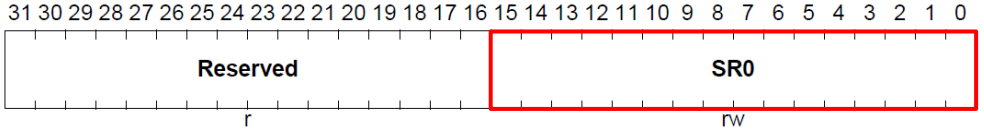
GTM_TOM0_CH1_SR0 Register 주소: F010_8044h
(F0100000h + 8044h)

GTM_TOM0_CH1_SR0 Register 구조:

Table 25-63 Registers Address Space

Module	Base Address	End Address	Note
GTM	$F010\ 0000_H$	$F019\ FFFF_H$	

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



$$(Period\ of\ PWM) = \frac{(Value\ of\ CM0) + 1}{(Freq.of\ CMU\_FXCLK1)}$$

$$= \frac{12500}{6250kHz} = 0.002s$$

PWM Example

2. Data sheet 분석 : TOM0 – Channel 1 설정 (4)

   ✓ GTM_TOM0_CHx_SR1 Register는 CM1에 대한 Shadow Register이다.

   ✓ TOM Channel 1의 CM1을 설정하기 위해 **GTM_TOM0_CH1_SR1 Register**를 설정한다.

   ✓ GTM_TOM0_CH1_SR1 Register에 설정할 CM1 값을 저장하면 업데이트를 할 때 CM1에 반영된다.

   ✓ CM1에 의한 **Duty Ratio**는 $\left(\frac{CM1+1}{CM0+1} \times 100\right)(\%)$ 이다.

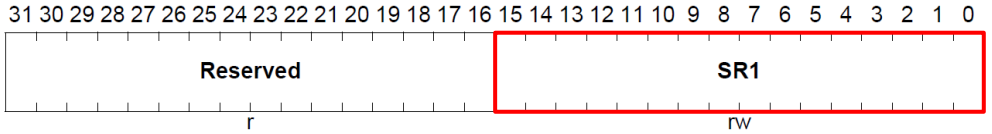
GTM_TOM0_CH1_SR1 Register 주소: F010_8048h
(F0100000h + 8048h)

GTM_TOM0_CH1_SR1 Register 구조:

Table 25-63 Registers Address Space

Module	Base Address	End Address	Note
GTM	$F010\ 0000_H$	$F019\ FFFF_H$	

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



PWM Example

2. Data sheet 분석 : TOUT 설정 (1)
   - ✓ GTM 모듈 내 하위 모듈에서 생성한 출력 신호를 외부에 전달하기 위해서는 GTM 모듈의 출력 포트 (TOUT)와 연결 설정을 해야 한다.
   - ✓ 하나의 출력 포트에는 하위 모듈에서 생성된 출력 신호 4개가 MUX를 통해 연결되어 있으며 MUX 제어를 통해 하나의 신호가 출력 포트와 연결된다.
   - ✓ **GTM_TOUTSEL Register**는 MUX에 제어 신호를 입력하며 하나의 Register가 16개의 MUX를 제어한다.
   - ✓ 따라서, LED가 연결된 TOUT103 (PORT10 Pin 1)은 **GTM_TOUTSEL6 Register**의 **SEL7 bits**를 통해 설정할 수 있다.

PWM Example

2. Data sheet 분석 : TOUT 설정 (2)

- ✓ GTM_TOUTSEL Register는 TOUT을 통해 출력될 신호를 설정한다.
- ✓ TOUT103에 대해 설정하기 위해 **GTM_TOUTSEL6 Register**의 **SEL7 bits**를 설정한다.
- ✓ TOM0 Channel 1를 통해 생성한 PWM 신호를 TOUT103로 출력하기 위해 **SEL7 bits**를 **00b**로 설정한다.

GTM_TOUTSEL6 Register 주소: F019_FD48h
(F0100000h + 9FD48h)

GTM_TOUTSEL6 Register 구조:

Table 25-63 Registers Address Space

Module	Base Address	End Address	Note
GTM	$F010\ 0000_H$	$F019\ FFFF_H$	

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.*

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
P10.1	TIN103	TOUT103	TIM0_1	TIM1_1	TOM0_1	TOM2_9	ATOM 1_1	ATOM 4_1

103/16 = 6, 103%16=7
104/16 = 6, 104%16=8

PWM Example

3. 프로그래밍

1) LED가 연결된 PORT에 대한 설정을 수행하는 함수를 구현한다.

```
74  // PORT10 Registers
75  #define PORT10_BASE      (0xF003B000)
76  #define PORT10_IOCR0     (*(volatile unsigned int*)(PORT10_BASE + 0x10))
77
78  #define PC1              11
```

PORT IO 설정관련 레지스터 주소 및 비트 필드 정의

```
109 void init_LED(void)
110 {
111     PORT10_IOCR0 &= ~((0x1F) << PC1);          // PORT10.1 : Alternate output function 1 (push-pull)
112     PORT10_IOCR0 |= ((0x11) << PC1);           // PORT10.1 : GTM_TOUT103
113
114 }
```

PORT IO 설정 함수

PWM Example

3. 프로그래밍

2) GTM을 설정하기 위한 함수를 구현한다.

① SCU_WDTCPU0CON0 Register를 통해 Password/Modify Access를 수행하여 System ENDINIT을 해제한다.

② GTM_CLC Register를 통해 GTM 모듈을 Enable 한다.

③ SCU_WDTCPU0CON0 Register를 통해 Password/Modify Access를 수행하여 System ENDINIT을 설정한다.

④ GTM_CMU_FXCLK_CTRL Register와 GTM_CMU_CLK_EN Register를 통해 CMU_FXCLK를 설정한다.

⑤ GTM_TOM0_TGC0_GLB_CTRL Register를 통해 CM0 / CM1 / CLK_SRC에 대한 업데이트를 Enable 한다.

⑥ GTM_TOM0_TGC0_FUPD_CTRL Register를 통해 트리거 신호에 따른 동작 (Force update, Clear CN0)을 설정한다.

⑦ GTM_TOM0_TGC0_ENDIS_CTRL Register와 GTM_TOM0_TGC0_OUTEN_CTRL Register를 통해 트리거 신호에 따른 동작 (Channel enable, Output enable)을 설정한다.

⑧ GTM_TOM0_CH1_CTRL Register를 통해 Signal level을 설정한다.

⑨ GTM_TOM0_CH1_CTRL / GTM_TOM0_CH1_SR0 / GTM_TOM0_CH1_SR1 Register를 통해 CM0 / CM1 / CLK_SRC에 대한 Shadow Register를 설정한다.

⑩ GTM_TOUTSEL6 Register를 통해 TOM0 Channel 1의 PWM 신호가 TOUT103로 출력되도록 설정한다.

⑪ GTM_TOM0_TGC0_GLB_CTRL Register를 통해 모든 설정이 반영되도록 HOST TRIGGER를 발생시킨다.

PWM Example

3. 프로그래밍

2) GTM을 설정하기 위한 함수를 구현한다.

```
/* Address of Registers */
// SCU Registers
#define SCU_BASE        (0xF0036000)
#define SCU_WDT_CPU0CON0 (*(volatile unsigned int*)(SCU_BASE + 0x100))

#define LCK             1
#define ENDINIT         0

// GTM Registers
// GTM - CMU
#define GTM_BASE                    (0xF0100000)
#define GTM_CMU_CLK_EN              (*(volatile unsigned int*)(GTM_BASE + 0x00300))
#define GTM_CMU_FXCLK_CTRL          (*(volatile unsigned int*)(GTM_BASE + 0x00344))

#define EN_FXCLK                    22
#define FXCLK_SEL                   0

// GTM - TOM0
#define GTM_TOM0_TGC0_GLB_CTRL      (*(volatile unsigned int*)(GTM_BASE + 0x08030))
#define GTM_TOM0_TGC0_ENDIS_CTRL    (*(volatile unsigned int*)(GTM_BASE + 0x08070))
#define GTM_TOM0_TGC0_OUTEN_CTRL    (*(volatile unsigned int*)(GTM_BASE + 0x08078))
#define GTM_TOM0_TGC0_FUPD_CTRL     (*(volatile unsigned int*)(GTM_BASE + 0x08038))
#define GTM_TOM0_CH1_CTRL           (*(volatile unsigned int*)(GTM_BASE + 0x08040))
#define GTM_TOM0_CH1_SR0            (*(volatile unsigned int*)(GTM_BASE + 0x08044))
#define GTM_TOM0_CH1_SR1            (*(volatile unsigned int*)(GTM_BASE + 0x08048))

#define UPEN_CTRL1                  18
#define HOST_TRIG                   0
#define ENDIS_CTRL1                 2
#define OUTEN_CTRL1                 2
#define RSTCN0_CH1                  18
#define FUPD_CTRL1                  2
#define CLK_SRC_SR                  12
#define SL                          11

// GTM
#define GTM_CLC                     (*(volatile unsigned int*)(GTM_BASE + 0x9FD00))
#define GTM_TOUTSEL6                (*(volatile unsigned int*)(GTM_BASE + 0x9FD48))

#define DISS                        1
#define DISR                        0
#define SEL7                        14
```

GTM 설정관련 레지스터 주소 및 비트 필드 정의

PWM Example

3. 프로그래밍

2) GTM을 설정하기 위한 함수를 구현한다.

```
116 void init_GTM_TOM0_PWM(void)
117 {
118     /* GTM Enable */
119     // Password Access to unlock WDTCPU0CON0
120 ①   SCU_WDT_CPU0CON0 = ((SCU_WDT_CPU0CON0 ^ 0xFC) & ~(1 << LCK)) | (1 << ENDINIT);
121     while((SCU_WDT_CPU0CON0 & (1 << LCK)) != 0);
122
123     // Modify Access to clear ENDINIT bit
124     SCU_WDT_CPU0CON0 = ((SCU_WDT_CPU0CON0 ^ 0xFC) | (1 << LCK)) & ~(1 << ENDINIT);
125     while((SCU_WDT_CPU0CON0 & (1 << LCK)) == 0);
126
127 ②   GTM_CLC &= ~(1 << DISR);                          // Enable GTM Module
128
129     // Password Access to unlock WDTCPU0CON0
130 ③   SCU_WDT_CPU0CON0 = ((SCU_WDT_CPU0CON0 ^ 0xFC) & ~(1 << LCK)) | (1 << ENDINIT);
131     while((SCU_WDT_CPU0CON0 & (1 << LCK)) != 0);
132
133     // Modify Access to set ENDINIT bit
134     SCU_WDT_CPU0CON0 = ((SCU_WDT_CPU0CON0 ^ 0xFC) | (1 << LCK)) | (1 << ENDINIT);
135     while((SCU_WDT_CPU0CON0 & (1 << LCK)) == 0);
136
137     while((GTM_CLC & (1 << DISS)) != 0);              // Wait until module is enabled
138
139     /* GTM Clock Setting */
140 ④   GTM_CMU_FXCLK_CTRL &= ~((0xF) << FXCLK_SEL);     // Input clock of CMU_FXCLK : CMU_GCLK_EN
141
142     GTM_CMU_CLK_EN |= ((0x2) << EN_FXCLK);           // Enable all CMU_FXCLK
```

PWM Example

3. 프로그래밍

2) GTM을 설정하기 위한 함수를 구현한다.

```
144     /* GTM TOM0 PWM Setting */
145 ⑤   GTM_TOM0_TGC0_GLB_CTRL |= ((0x2) << UPEN_CTRL1);     // TOM0 channel 1 enable update of
146                                                          // register CM0, CM1, CLK_SRC
147
148 ⑥   GTM_TOM0_TGC0_FUPD_CTRL |= ((0x2) << FUPD_CTRL1);    // Enable force update of TOM0 channel 1
149     GTM_TOM0_TGC0_FUPD_CTRL |= ((0x2) << RSTCN0_CH1);    // Reset CN0 of TOM0 channel 1 on force update
150
151 ⑦   GTM_TOM0_TGC0_ENDIS_CTRL |= ((0x2) << ENDIS_CTRL1); // Enable channel 1 on an update trigger
152     GTM_TOM0_TGC0_OUTEN_CTRL |= ((0x2) << OUTEN_CTRL1); // Enable channel 1 output on an update trigger
153
154 ⑧   GTM_TOM0_CH1_CTRL |= (1 << SL);                      // High signal level for duty cycle
155
156 ⑨   GTM_TOM0_CH1_CTRL &= ~((0x7) << CLK_SRC_SR);         // Clock source : CMU_FXCLK(1) = 6250 kHz
157     GTM_TOM0_CH1_CTRL |= (1 << CLK_SRC_SR);
158     GTM_TOM0_CH1_SR0 = 12500 - 1;                        // PWM freq. = 6250 kHz / 12500 = 500 Hz
159 //    GTM_TOM0_CH1_SR1 = 0;                                // Duty cycle = 0
160 //    GTM_TOM0_CH1_SR1 = 6250-1;                           // Duty cycle = 50
161     GTM_TOM0_CH1_SR1 = 12500 - 1;                        // Duty cycle = 100
162
163 Ⓐ   GTM_TOUTSEL6 &= ~((0x3) << SEL7);                    // TOUT103 : TOM0 channel 1
164
165 Ⓑ   GTM_TOM0_TGC0_GLB_CTRL |= (1 << HOST_TRIG);         // Trigger request signal to update
166 }
```

GTM 설정 함수

PWM Example

3. 프로그래밍

3) 동작에 따라 'main' 함수를 구현한다. (앞서 구현한 함수들을 호출한다.)

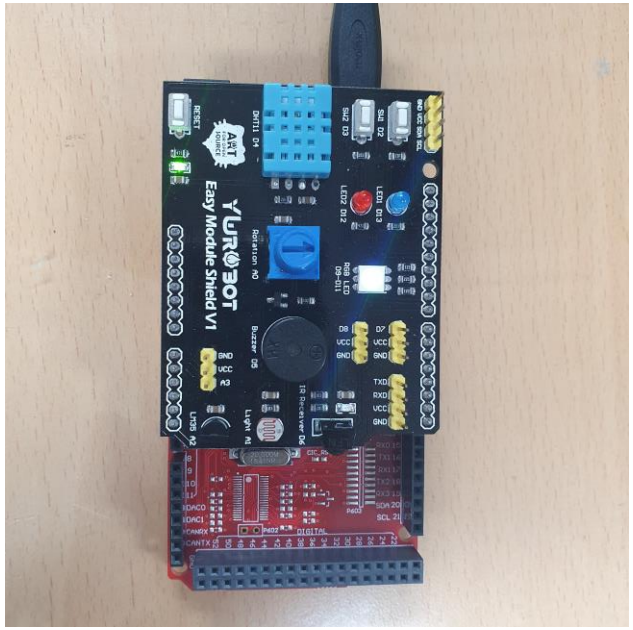
```
/* Address of Registers */
// SCU Registers
#define SCU_BASE        (0xF0036000)
#define SCU_WDT_CPU0CON0 (*(volatile unsigned int*)(SCU_BASE + 0x100))

#define LCK             1
#define ENDINIT         0

// GTM Registers
// GTM - CMU
#define GTM_BASE                    (0xF0100000)
#define GTM_CMU_CLK_EN              (*(volatile unsigned int*)(GTM_BASE + 0x00300))
#define GTM_CMU_FXCLK_CTRL          (*(volatile unsigned int*)(GTM_BASE + 0x00344))

#define EN_FXCLK                    22
#define FXCLK_SEL                   0

// GTM - TOM0
#define GTM_TOM0_TGC0_GLB_CTRL      (*(volatile unsigned int*)(GTM_BASE + 0x08030))
#define GTM_TOM0_TGC0_ENDIS_CTRL    (*(volatile unsigned int*)(GTM_BASE + 0x08070))
#define GTM_TOM0_TGC0_OUTEN_CTRL    (*(volatile unsigned int*)(GTM_BASE + 0x08078))
#define GTM_TOM0_TGC0_FUPD_CTRL     (*(volatile unsigned int*)(GTM_BASE + 0x08038))
#define GTM_TOM0_CH1_CTRL           (*(volatile unsigned int*)(GTM_BASE + 0x08040))
#define GTM_TOM0_CH1_SR0            (*(volatile unsigned int*)(GTM_BASE + 0x08044))
#define GTM_TOM0_CH1_SR1            (*(volatile unsigned int*)(GTM_BASE + 0x08048))

#define UPEN_CTRL1                  18
#define HOST_TRIG                   0
#define ENDIS_CTRL1                 2
#define OUTEN_CTRL1                 2
#define RSTCN0_CH1                  18
#define FUPD_CTRL1                  2
#define CLK_SRC_SR                  12
#define SL                          11

// GTM
#define GTM_CLC                     (*(volatile unsigned int*)(GTM_BASE + 0x9FD00))
#define GTM_TOUTSEL6                (*(volatile unsigned int*)(GTM_BASE + 0x9FD48))

#define DISS                        1
#define DISR                        0
#define SEL7                        14
```

```
// PORT10 Registers
#define PORT10_BASE     (0xF003B000)
#define PORT10_IOCR0    (*(volatile unsigned int*)(PORT10_BASE + 0x10))

#define PC1             11

/* Function Prototype */
void init_LED(void);
void init_GTM_TOM0_PWM(void);

IfxCpu_syncEvent g_cpuSyncEvent = 0;

int core0_main(void)
{
    IfxCpu_enableInterrupts();

    /* !!WATCHDOG0 AND SAFETY WATCHDOG ARE DISABLED HERE!!
     * Enable the watchdogs and service them periodically if it is required
     */
    IfxScuWdt_disableCpuWatchdog(IfxScuWdt_getCpuWatchdogPassword());
    IfxScuWdt_disableSafetyWatchdog(IfxScuWdt_getSafetyWatchdogPassword());

    /* Wait for CPU sync event */
    IfxCpu_emitEvent(&g_cpuSyncEvent);
    IfxCpu_waitEvent(&g_cpuSyncEvent, 1);

    init_LED();
    init_GTM_TOM0_PWM();

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

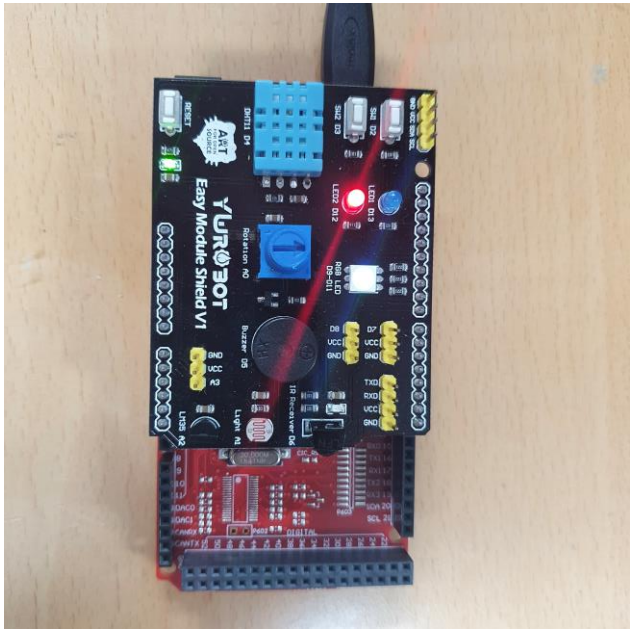
PWM Example

4. 동작 확인

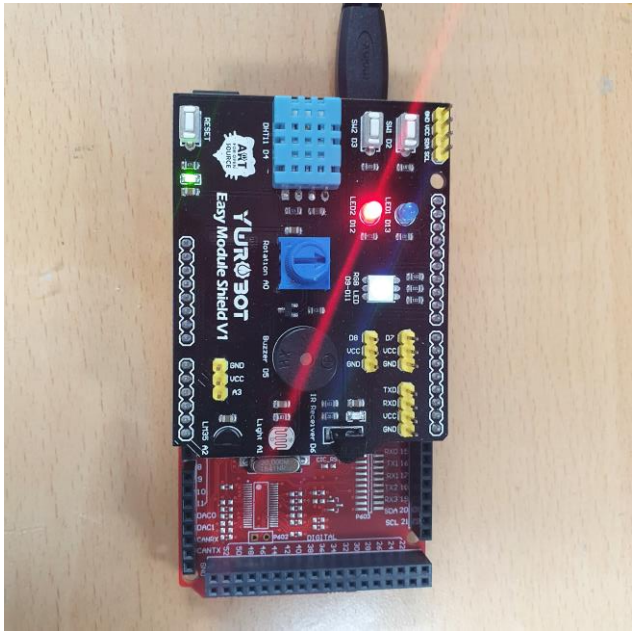
   ✓ Build 및 Debug 후 (‘Resume’ 버튼 클릭), CM1 값을 바꿔보며 Duty Ratio에 따른 LED 밝기를 확인한다.



Duty = 0%



Duty = 50%



Duty = 100%

PWM Example

1. import tc275_PWM
2. BLUE LED 추가

   P10.2 → TOUT104

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

Port	Input	Output	Input Timer Mapped		Output Timer Mapped			
			A	B	A	B	C	D
P02.2	TIN2	TOUT2	TIM0_2	TIM1_2	TOM0_10	TOM1_10	ATOM 0_2	ATOM 1_2
P02.3	TIN3	TOUT3	TIM0_3	TIM1_3	TOM0_11	TOM1_11	ATOM 0_3	ATOM 1_3
P02.4	TIN4	TOUT4	TIM0_4	TIM1_4	TOM0_12	TOM1_12	ATOM 0_4	ATOM 1_4
P02.5	TIN5	TOUT5	TIM0_5	TIM1_5	TOM0_13	TOM1_13	ATOM 0_5	ATOM 1_5
P02.6	TIN6	TOUT6	TIM0_6	TIM1_6	TOM0_14	TOM1_14	ATOM 0_6	ATOM 1_6
P02.7	TIN7	TOUT7	TIM0_7	TIM1_7	TOM0_15	TOM1_15	ATOM 0_7	ATOM 1_7
P02.8	TIN8	TOUT8	TIM2_0	TIM3_0	TOM0_8	TOM1_0	ATOM 0_0	ATOM 1_0
P10.0	TIN102	TOUT102	TIM0_4	TIM1_4	TOM0_4	TOM2_12	ATOM 1_4	ATOM 4_4
P10.1	TIN103	TOUT103	TIM0_1	TIM1_1	TOM0_1	TOM2_9	ATOM 1_1	ATOM 4_1
P10.2	TIN104	TOUT104	TIM0_2	TIM1_2	TOM0_2	TOM2_10	ATOM 1_2	ATOM 4_2
P10.3	TIN105	TOUT105	TIM0_3	TIM1_3	TOM0_3	TOM2_11	ATOM 1_3	ATOM 4_3
P10.4	TIN106	TOUT106	TIM0_6	TIM1_6	TOM0_6	TOM2_6	ATOM 0_6	ATOM 4_6
P10.5	TIN107	TOUT107	TIM0_2	TIM1_2	TOM0_2	TOM2_10	ATOM 1_2	ATOM 4_2
P10.6	TIN108	TOUT108	TIM0_3	TIM1_3	TOM0_3	TOM2_11	ATOM 1_3	ATOM 4_3
P10.7	TIN109	TOUT109	TIM0_0	TIM1_0	TOM0_0	TOM2_8	ATOM 1_0	ATOM 4_0
P10.8	TIN110	TOUT110	TIM0_5	TIM1_5	TOM0_5	TOM2_13	ATOM 1_5	ATOM 4_5

PWM Example

1. import tc275_PWM
2. BLUE LED 추가

   P10.2 → TOUT104
3. ADC4.7(Potential Meter) RED LED Duty
4. ADC4.6(Light Sensor) BLUE LED Duty

ADC → duty calculation

duty = ADC/4095*pwm period

https://github.com/swip6th/swip6th/mcu/tree/main/tc275_PWM

Cpu0_Main_LAB0.c

my_lib.h

my_lib.c

PWM Example

1. import tc275_PWM
2. BLUE LED 추가

   P10.2 → TOUT104
3. ADC4.7(Potential Meter) RED LED Duty
4. ADC4.6(Light Sensor) BLUE LED Duty

```
while(1)
{
    systick_curr = SYSTEM_TIMER_0_31_0;
    systick = systick_curr - systick_prev;

    if( systick > SYSTICK_100MHZ/10 )   // 100ms
    {
        systick_prev = systick_curr;

        potential_meter = GetVADC4(7);
        light_sensor = GetVADC4(6);

        // FIXME: light sensor scaling

        // FIXME: duty calculation

        GTM_TOM0_CH1_SR0 = 12500 - 1;
        GTM_TOM0_CH2_SR0 = 12500 - 1;

        if( potential_meter_duty == 0 )
            GTM_TOM0_CH1_SR1 = 0;
        else
            GTM_TOM0_CH1_SR1 = potential_meter_duty - 1;

        if( light_sensor_duty == 0 )
            GTM_TOM0_CH2_SR1 = 0;
        else
            GTM_TOM0_CH2_SR1 = light_sensor_duty - 1;
    }
}
```

dynamic range scaling

light_sensor_duty = ((light_sensor – light_sensor_min)*4095)/(max-min)

light_sensor_max, light_sensor_min 은 debugger break 기능 사용하여 측정

min 보다 작은 경우 0, max 보다 클 경우 4095 처리

ADC → duty calculation

duty = ADC/4095*pwm period

https://github.com/swip6th/swip6th/mcu/tree/main/tc275_PWM

Cpu0_Main_LAB1.c

my_lib.h

my_lib.c

RGB.PWM

RGB PWM	Red	TOUT7
	Green	TOUT107
	Blue	TOUT105

Red

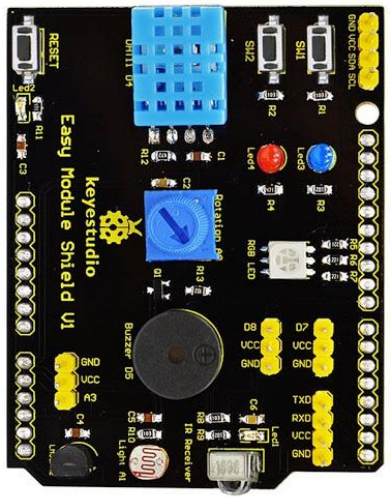
Blue

Green

P02.7	TIN7	TOUT7	TIM0_7	TIM1_7	TOM0_15	TOM1_15	ATOM 0_7	ATOM 1_7
P02.8	TIN8	TOUT8	TIM2_0	TIM3_0	TOM0_8	TOM1_0	ATOM 0_0	ATOM 1_0
P10.0	TIN102	TOUT102	TIM0_4	TIM1_4	TOM0_4	TOM2_12	ATOM 1_4	ATOM 4_4
P10.1	TIN103	TOUT103	TIM0_1	TIM1_1	TOM0_1	TOM2_9	ATOM 1_1	ATOM 4_1
P10.2	TIN104	TOUT104	TIM0_2	TIM1_2	TOM0_2	TOM2_10	ATOM 1_2	ATOM 4_2
P10.3	TIN105	TOUT105	TIM0_3	TIM1_3	TOM0_3	TOM2_11	ATOM 1_3	ATOM 4_3
P10.4	TIN106	TOUT106	TIM0_6	TIM1_6	TOM0_6	TOM2_6	ATOM 0_6	ATOM 4_6
P10.5	TIN107	TOUT107	TIM0_2	TIM1_2	TOM0_2	TOM2_10	ATOM 1_2	ATOM 4_2

SW1 누르면, Blue LED 켜고 RGB.Blue Duty 설정
SW2 누르면, Red LED 켜고 RGB.Red Duty 설정
둘다 누르면, Blue/Red LED 켜고 RGB.Green Duty 설정
SW Debounce: 1초, SW state Debounce: 1초

모듈		TC275 Shield Buddy Pin	TC275 Pin	TC275 관련기능
RGB LED	Red	D9	P02_7	GPIO, PWM
	Green	D10	P10_5	GPIO, PWM
	Blue	D11	P10_3	GPIO, PWM



tc275_PWM_RGB project 생성
https://github.com/swip6th/mcu/tree/main/tc275_PWM_RGB
Cpu0_Main.c
my_lib.h
my_lib.c

RGB.PWM

RGB PWM	Red	TOUT7
	Green	TOUT107
	Blue	TOUT105

SW1 누르면, Blue LED 켜고 RGB.Blue Duty 설정
SW2 누르면, Red LED 켜고 RGB.Red Duty 설정
둘다 누르면, Blue/Red LED 켜고 RGB.Green Duty 설정
SW Debounce: 1초, SW state Debounce: 1초

모듈		TC275 Shield Buddy Pin	TC275 Pin	TC275 관련기능
RGB LED	Red	D9	P02_7	GPIO, PWM
	Green	D10	P10_5	GPIO, PWM
	Blue	D11	P10_3	GPIO, PWM

```
111
112         // FIXME: 1s state debounce
113         SW_state_prev = SW_state_curr;
114         SW_state_curr = (SW2_debounce<<1) | (SW1_debounce<<0);
115
116         if( SW_state_debounce != 0 )
117         {
118             potential_meter = GetVADC4(7);
119             potential_meter_duty = (potential_meter*12500)/4095;
120
121             if( potential_meter_duty == 0 )
122                 potential_meter_duty  = 0;
123             else
124                 potential_meter_duty -= 1;
125
126             // FIXME:
127             // SW1 pushed     : blue     led on, RGB.blue  duty     TOM0_CH3
128             // SW2 pushed     : red      led on, RGB.red   duty     TOM0_CH15
129             // SW1&SW2 pushed : blue&red led on, RGB.green duty     TOM0_CH2
130
131             // End of FIXME
132         }
133         else                                                    // SW1 & SW2 are open
134         {
135                 PORT10_OMR = (1<<PCL1)  |                       // LED RED  off
136                              (1<<PCL2)  ;                       // LED BLUE off
137         }
138     }
139     return (1);
140 }
141
142
143 // 100ms timer
144 __interrupt( 0x0F ) __vector_table( 0 )
145 void CCU61_T12_ISR(void)
146 {
147     SW1_prev = SW1_curr;
148     SW1_curr = (PORT02_IN & (1<<P0)) == 0;
149     SW2_prev = SW2_curr;
150     SW2_curr = (PORT02_IN & (1<<P1)) == 0;
151
152     // FIXME: SW1, SW2 software debounce for 1 second
153
154     irq_timer = 1;
155 }
```

Buzzer

```
GTM_CMU_CLK_EN |= ((0x2) << EN_FXCLK);      // enable

GTM_CMU_CLK_EN &= ~((0x2) << EN_FXCLK);     // disable
```

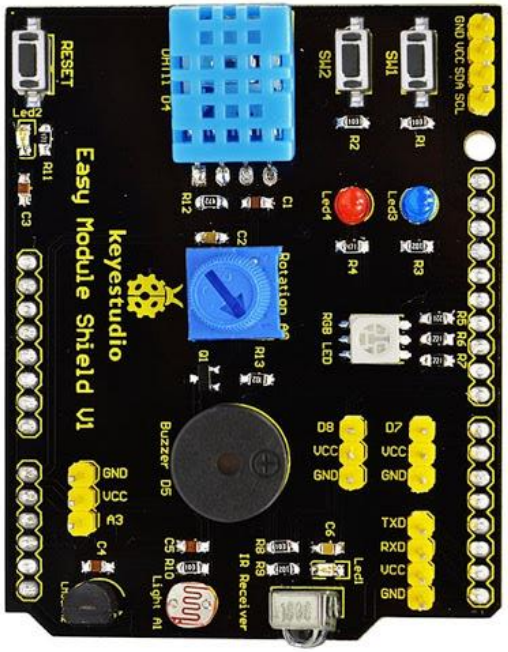
TOUT3 GTM_TOM0 Channel 11

delay 앞 Buzzer Enable

delay 뒤 Buzzer Disable

int pwm_cnt = PWM_FREQ / tones[i]; break 시 Buzzer 들리지 않게!!!!!

모듈		TC275 Shield Buddy Pin	TC275 Pin	TC275 관련기능
LED1		D13	P10_2	GPIO, PWM
LED2		D12	P10_1	GPIO, PWM
SW1		D2	P02_0	GPIO
SW2		D3	P02_1	GPIO
RGB LED	Red	D9	P02_7	GPIO, PWM
	Green	D10	P10_5	GPIO, PWM
	Blue	D11	P10_3	GPIO, PWM
Rotation		A0	SAR4_7	ADC
Buzzer		D5	P02_3	PWM
Light		A1	SAR4_6	ADC
LM35		A2	SAR4_5	ADC

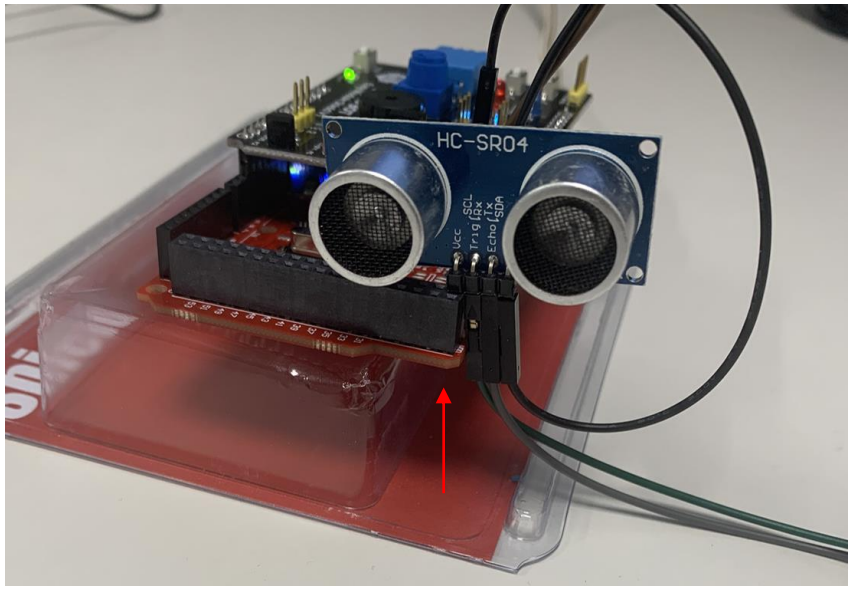


Ultrasonic

VDD(5V)

Trig	P15_4
Echo	P15_5
GND	GND

거리(cm) Live Update



https://github.com/swip6th/mcu/tree/main/tc275_Ultrasonic

Cpu0_Main.c

ACE Lab.

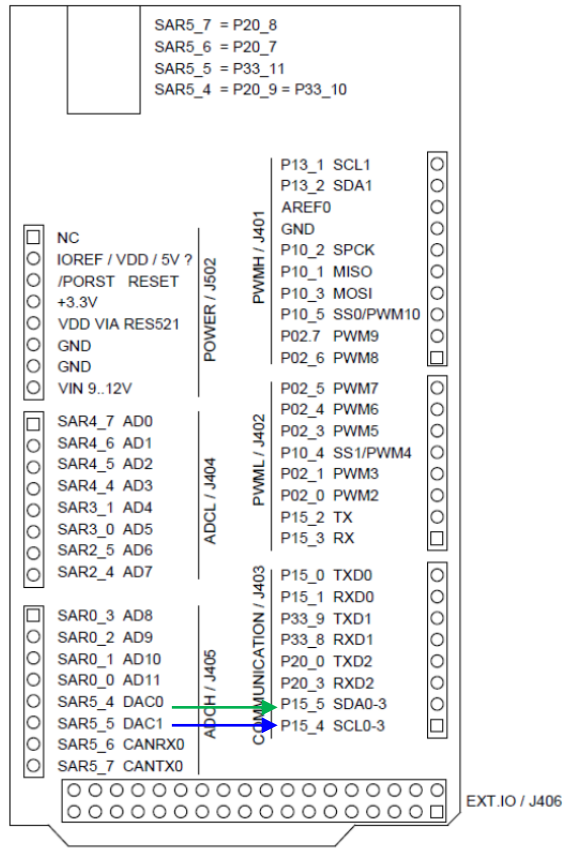
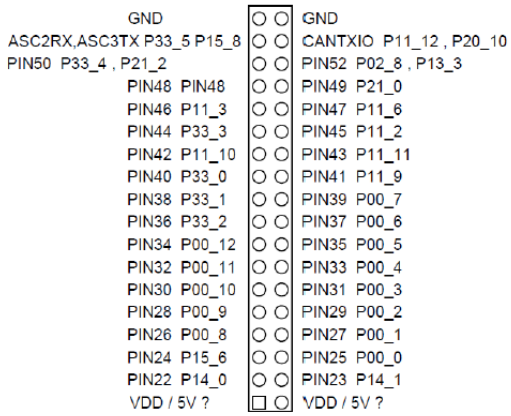


Figure 7 TC275 to Arduino Connector Mapping



Ultrasonic

Trig	P15_4	Output
Echo	P15_5	Input

```
void init_ultrasonic(void)
{
    /* Init TRIG Pin - P15.4 (Output) */
    PORT15_IOCR4 &= ~((0x1F) << PC4);
    PORT15_IOCR4 |= ((0x10) << PC4);

    /* Init ECHO Pin - P15.5 (Input) */
    PORT15_IOCR4 &= ~((0x1F) << PC5);
}
```

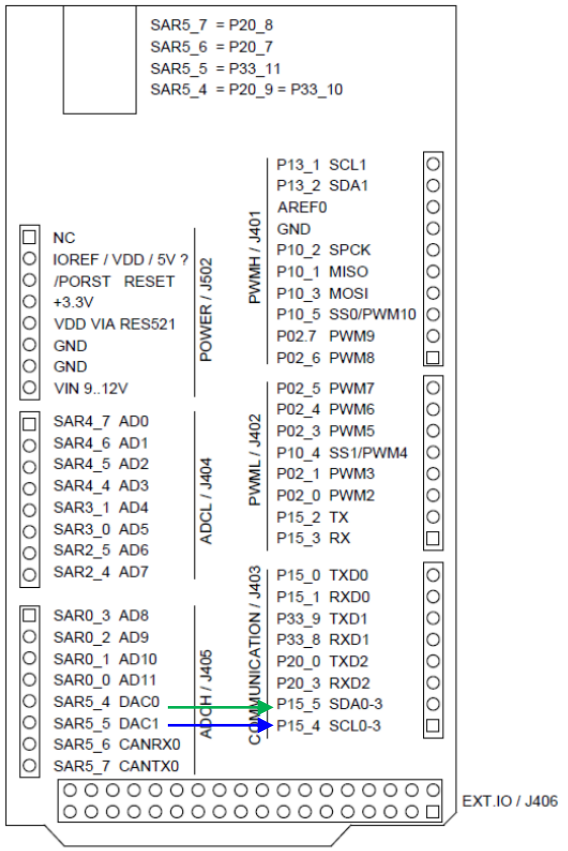
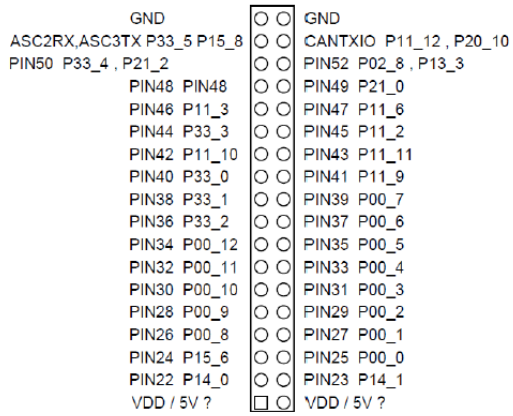
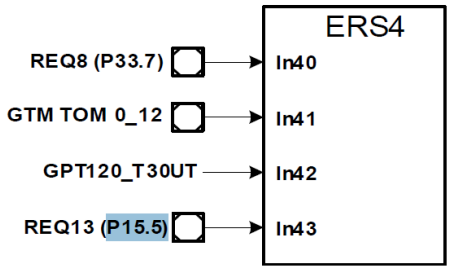


Figure 7 TC275 to Arduino Connector Mapping



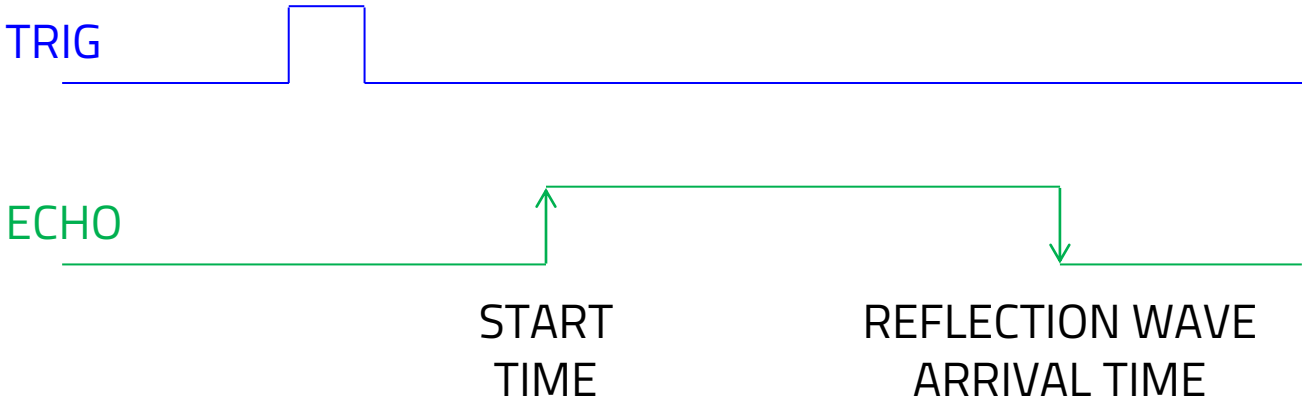
Ultrasonic




```
void init_ERU(void)
{
    /* ERU Input Channel 4 Setting */
    /* Password Access to unlock WDTSCON0 */
    SCU_WDTSCON0 = ((SCU_WDTSCON0 ^ 0xFC) & ~(1 << LCK)) | (1 << ENDINIT);
    while((SCU_WDTSCON0 & (1 << LCK)) != 0);

    // Modify Access to clear ENDINIT bit
    SCU_WDTSCON0 = ((SCU_WDTSCON0 ^ 0xFC) | (1 << LCK)) & ~ (1 << ENDINIT);
    while((SCU_WDTSCON0 & (1 << LCK)) == 0);

    SCU_EICR2 &= ~((0x7) << EXIS0);             // External input 3 is selected
    SCU_EICR2 |= ((0x3) << EXIS0);

    SCU_EICR2 |= (1 << REN0);                   // Rising edge enable
    SCU_EICR2 |= (1 << FEN0);                   // Falling edge enable

    SCU_EICR2 |= (1 << EIEN0);                  // The trigger event is enabled

    SCU_EICR2 &= ~((0x7) << INP0);              // An event from input ETL 4 triggers output OGU 0

    SCU_IGCR0 &= ~((0x3) << IGP0);              // IOUT(0) is activated in response to a trigger event
    SCU_IGCR0 |= (1 << IGP0);                   // The pattern is not considered

    /* Password Access to unlock WDTSCON0 */
    SCU_WDTSCON0 = ((SCU_WDTSCON0 ^ 0xFC) & ~(1 << LCK)) | (1 << ENDINIT);
    while((SCU_WDTSCON0 & (1 << LCK)) != 0);

    /* Modify Access to set ENDINIT bit */
    SCU_WDTSCON0 = ((SCU_WDTSCON0 ^ 0xFC) | (1 << LCK)) | (1 << ENDINIT);
    while((SCU_WDTSCON0 & (1 << LCK)) == 0);

    /* SRC Interrupt Setting For ECU */
    SRC_SCUERU0 &= ~((0xFF) << SRPN);           // Set Priority : 0x0B
    SRC_SCUERU0 |= ((0x0B) << SRPN);

    SRC_SCUERU0 &= ~((0x3) << TOS);             // CPU0 services

    SRC_SCUERU0 |= (1 << SRE);                  // Service Request is enabled
}
```

ACE Lab.

Ultrasonic



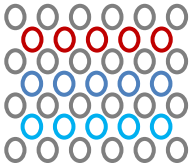
Distance = (ARRIVAL TIME – START TIME)/2 x 340m/s

$Distance_{10us}$ = (ARRIVAL TIME – START TIME)/2 x 340,00cm/1,000,000us
= (ARRIVAL TIME – START TIME) x 17~~0,00~~cm/1,~~000~~,000us
= (ARRIVAL TIME – START TIME) x 17/100cm/10us
= (ARRIVAL TIME – START TIME) x 17/10mm/10us
= (ARRIVAL TIME – START TIME) x 1700um/10us

$Distance_{10ns}$ = (ARRIVAL TIME – START TIME)/2 x 340,000mm/1,000,000,000ns
= (ARRIVAL TIME – START TIME) x 17~~0,000~~mm/1,~~000,0~~00,000ns
= (ARRIVAL TIME – START TIME) x 17/10000mm/10ns
= (ARRIVAL TIME – START TIME) x 17/10um/10ns

Q & A

Thank you for your attention



Architecture and
Compiler
for Embedded Systems Lab.

School of Electronics Engineering, KNU

ACE Lab (hn02301@gmail.com)