```
void etpwmInit(void);
void etpwmStartTBCLK(void);
void etpwmStopTBCLK(void);
void etpwmSetClkDiv(etpwmBASE_t *etpwm, etpwmClkDiv_t clkdiv, etpwmHspClkDiv_t
hspclkdiv);
void etpwmSetTimebasePeriod(etpwmBASE_t *etpwm, uint16 period);
void etpwmSetCount(etpwmBASE_t *etpwm, uint16 count);
void etpwmDisableTimebasePeriodShadowMode(etpwmBASE_t *etpwm);
void etpwmEnableTimebasePeriodShadowMode(etpwmBASE_t *etpwm);
void etpwmEnableCounterLoadOnSync(etpwmBASE_t *etpwm, uint16 phase, uint16
direction);
void etpwmDisableCounterLoadOnSync(etpwmBASE_t *etpwm);
void etpwmSetSyncOut(etpwmBASE_t *etpwm, etpwmSyncOut_t syncOutSrc);
void etpwmSetCounterMode(etpwmBASE_t *etpwm, etpwmCounterMode_t countermode);
void etpwmTriggerSWSync(etpwmBASE_t *etpwm);
void etpwmSetRunMode(etpwmBASE_t *etpwm, etpwmRunMode_t runmode);
void etpwmSetCmpA(etpwmBASE_t *etpwm, uint16 value);
void etpwmSetCmpB(etpwmBASE_t *etpwm, uint16 value);
void etpwmEnableCmpAShadowMode(etpwmBASE_t *etpwm, etpwmLoadMode_t loadmode);
void etpwmDisableCmpAShadowMode(etpwmBASE_t *etpwm);
void etpwmEnableCmpBShadowMode(etpwmBASE_t *etpwm, etpwmLoadMode_t loadmode);
void etpwmDisableCmpBShadowMode(etpwmBASE_t *etpwm);
void etpwmSetActionQualPwmA(etpwmBASE_t *etpwm, etpwmActionQualConfig_t
actionqualconfig);
void etpwmSetActionQualPwmB(etpwmBASE_t *etpwm, etpwmActionQualConfig_t
actionqualconfig);
void etpwmEnableDeadBand(etpwmBASE_t *etpwm, etpwmDeadBandConfig_t
deadbandconfig);
void etpwmDisableDeadband(etpwmBASE_t *etpwm);
void etpwmSetDeadBandDelay(etpwmBASE_t *etpwm, uint16 Rdelay, uint16 Fdelay);
void etpwmEnableChopping(etpwmBASE_t *etpwm, etpwmChoppingConfig_t
choppingconfig);
void etpwmDisableChopping(etpwmBASE_t *etpwm);
void etpwmEnableTripZoneSources(etpwmBASE_t *etpwm, etpwmTripZoneSrc_t sources);
void etpwmDisableTripZoneSources(etpwmBASE_t *etpwm, etpwmTripZoneSrc_t sources);
void etpwmSetTripAction(etpwmBASE_t *etpwm, etpwmTripActionConfig_t
tripactionconfig);
void etpwmEnableTripInterrupt(etpwmBASE_t *etpwm, etpwmTrip_t interrupts);
void etpwmDisableTripInterrupt(etpwmBASE_t *etpwm, etpwmTrip_t interrupts);
void etpwmClearTripCondition(etpwmBASE_t *etpwm, etpwmTrip_t trips);
void etpwmClearTripInterruptFlag(etpwmBASE_t *etpwm);
void etpwmForceTripEvent(etpwmBASE_t *etpwm, etpwmTrip_t trip);
void etpwmEnableSOCA(etpwmBASE_t *etpwm, etpwmEventSrc_t eventsource,
etpwmEventPeriod_t eventperiod);
void etpwmDisableSOCA(etpwmBASE_t *etpwm);
void etpwmEnableSOCB(etpwmBASE_t *etpwm, etpwmEventSrc_t eventsource,
etpwmEventPeriod_t eventperiod);
void etpwmDisableSOCB(etpwmBASE_t *etpwm);
void etpwmEnableInterrupt(etpwmBASE_t *etpwm, etpwmEventSrc_t eventsource,
etpwmEventPeriod_t eventperiod);
void etpwmDisableInterrupt(etpwmBASE_t *etpwm);
uint16 etpwmGetEventStatus(etpwmBASE_t *etpwm);
void etpwmClearEventFlag(etpwmBASE_t *etpwm, etpwmEvent_t events);
void etpwmTriggerEvent(etpwmBASE_t *etpwm, etpwmEvent_t events);
void etpwmEnableDigitalCompareEvents(etpwmBASE_t *etpwm,
etpwmDigitalCompareConfig_t digitalcompareconfig);
void etpwm1GetConfigValue(etpwm_config_reg_t *config_reg, config_value_type_t type);
```

```
void etpwmInit(void);

void etpwmInit(void)
{
/** - Sets high speed time-base clock prescale bits */
    etpwmREG1->TBCTL = (uint16)5U << 7U;
    etpwmREG1->TBCTL |= (uint16)((uint16)1U << 10U);
```

12-10	CLKDIV		Time-base Clock Prescale Bits. These bits determine part of the time-base clock prescale value: TBCLK = VCLK3 / (HSPCLKDIV × CLKDIV)
		0	/1 (default on reset)
		1h	/2
		2h	/4
		3h	/8
		4h	/16
		5h	/32
		6h	/64
		7h	/128
9-7	HSPCLKDIV		High Speed Time-base Clock Prescale Bits. These bits determine part of the time-base clock prescale value: TBCLK = VCLK3 / (HSPCLKDIV × CLKDIV)
		0	/1
		1h	/2 (default on reset)
		2h	/4
		3h	/6
		4h	/8
		5h	/10
		6h	/12
		7h	/14

TBCLK = VCLK3 / (HSPCLKDIV * CLKDIV) 가 된다.

따라서 VCLK3 / (10 * 2)가 된다. 분주를 해주기 위함.

(나머지는 다 0 입력 중요한 것 하나만 보자면 CTRMODE 가 0 이므로 Up-count 모드로 동작)

```
    /** - Sets time period or frequency for ETPWM block both PWMA and PWMB*/
    etpwmREG1->TBPRD = 4999U;
```

35.4.1.4 Time-Base Period Register (TBPRD)

Figure 35-66. Time-Base Period Register (TBPRD) [offset = 08h]

15		0
TBPRD		
R/W-0		

LEGEND: R/W = Read/Write; *-n* = value after reset

Table 35-26. Time-Base Period Register (TBPRD) Field Descriptions

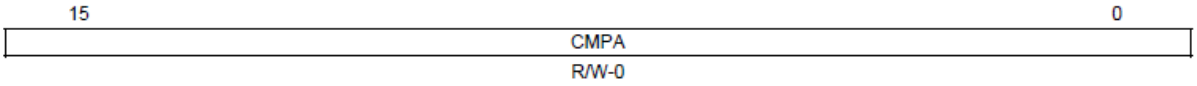
Bits	Name	Description
15-0	TBPRD	These bits determine the period of the time-base counter. This sets the PWM frequency. Shadowing of this register is enabled and disabled by the TBCTL[PRDLD] bit. By default this register is shadowed. • If TBCTL[PRDLD] = 0, then the shadow is enabled and any write or read will automatically go to the shadow register. In this case, the active register will be loaded from the shadow register when the time-base counter equals 0. • If TBCTL[PRDLD] = 1, then the shadow is disabled and any write or read will go directly to the active register, that is the register actively controlling the hardware. • The active and shadow registers share the same memory map address.

주기를 설정한다. TBCLK의 반복 횟수를 주기로 잡는다. TBCTL의 PRDLD 필드를 활용하여 쉐도우 기법을 활용할지 말지 결정한다.

```
/** - Setup the duty cycle for PWMA */
etpwmREG1->CMPA = 250U;
```

35.4.2.2 Counter-Compare A Register (CMPA)

Figure 35-69. Counter-Compare A Register (CMPA) [offset = 10h]



LEGEND: R/W = Read/Write; -n = value after reset

Table 35-29. Counter-Compare A Register (CMPA) Field Descriptions

Bits	Name	Description
15-0	CMPA	The value in the active CMPA register is continuously compared to the time-base counter (TBCTR). When the values are equal, the counter-compare module generates a "time-base counter equal to counter compare A" event. This event is sent to the action-qualifier where it is qualified and converted it into one or more actions. These actions can be applied to either the EPWMxA or the EPWMxB output depending on the configuration of the AQCTLA and AQCTLB registers. The actions that can be defined in the AQCTLA and AQCTLB registers include: • Do nothing; the event is ignored. • Clear: Pull the EPWMxA and/or EPWMxB signal low. • Set: Pull the EPWMxA and/or EPWMxB signal high. • Toggle the EPWMxA and/or EPWMxB signal. Shadowing of this register is enabled and disabled by the CMPCTL[SHDWAMODE] bit. By default this register is shadowed. • If CMPCTL[SHDWAMODE] = 0, then the shadow is enabled and any write or read will automatically go to the shadow register. In this case, the CMPCTL[LOADAMODE] bit field determines which event will load the active register from the shadow register. • Before a write, the CMPCTL[SHDWAFULL] bit can be read to determine if the shadow register is currently full. • If CMPCTL[SHDWAMODE] = 1, then the shadow register is disabled and any write or read will go directly to the active register, that is the register actively controlling the hardware. • In either mode, the active and shadow registers share the same memory map address.

해당 레지스터의 값을 활용하여 TBCTR 레지스터 (현재 TBCLK의 카운터 횟수가 저장되어 있음)와 비교한다. (즉, duty cycle을 정해준다는 뜻이다.)
CMPA에 250을 넣음으로써, TBCTR이 250이 되면 counter compare A 이벤트가 발생한다. 이 이벤트는 AQ 로 보내지게 된다. 해당 AQ에 의해 PWM 신호가 만들어 진다. AQCTLA와 AQCTLB의 레지스터를 통하여 EPMWxA와 EPMWxB의 기능을 정해 줄 수 있다.(clear, set, toggle, do nothing) CMPCTL[SHDWAMODE]의 기능을 활용하여 쉐도우 기법을 유/무를 정한다.

```
/** - Setup the duty cycle for PWMB */
etpwmREG1->CMPB = 2500U;
```

CMPA와 같은 기능을 해준다.

```
    /** - Force EPWMxA output high when counter reaches zero and low when counter
reaches Compare A value */
    etpwmREG1->AQCTLA = ((uint16)((uint16)ActionQual_Set   << 0U)
                       | (uint16)((uint16)ActionQual_Clear << 4U));
```

비트는 1번과 4번을 온 시킨다.

5-4	CAU		Action when the counter equals the active CMPA register and the counter is incrementing.
		0	Do nothing (action is disabled).
		1h	Clear: force EPWMxA output low.
		2h	Set: force EPWMxA output high.
		3h	Toggle EPWMxA output: low output signal will be forced high, and a high signal will be forced low.
3-2	PRD		Action when the counter equals the period. Note: By definition, in count up-down mode when the counter equals period the direction is defined as 0 or counting down.
		0	Do nothing (action is disabled).
		1h	Clear: force EPWMxA output low.
		2h	Set: force EPWMxA output high.
		3h	Toggle EPWMxA output: low output signal will be forced high, and a high signal will be forced low.
1-0	ZRO		Action when counter equals zero. Note: By definition, in count up-down mode when the counter equals 0 the direction is defined as 1 or counting up.
		0	Do nothing (action is disabled).
		1h	Clear: force EPWMxA output low.
		2h	Set: force EPWMxA output high.
		3h	Toggle EPWMxA output: low output signal will be forced high, and a high signal will be forced low.

ZRO는 2h이므로 increase 모드에서 counter가 0이 되는 순간 output을 high로 지정한다.
CAU는 1h이므로 increase 모드에서 counter = CPMA 인 순간, output을 low로 지정한다.
즉, 처음에 high, CPMA까지 유지후에 주기 만큼 0이 되는 PWM 신호가 완성된다.
해당 레지스터는 전체적으로 카운터의 방향과 CMPA, CMPB, zero, period 도달 시, 이벤트를 설정 해 줄 수 있다.

```
    /** - Force EPWMxB output high when counter reaches zero and low when counter
reaches Compare B value */
    etpwmREG1->AQCTLB = ((uint16)((uint16)ActionQual_Set   << 0U)
                       | (uint16)((uint16)ActionQual_Clear << 8U));
```

AQCTLA와 같은 역할을 해준다.
다만 output 출력이 EPWMxB이다.

```
    /** - Mode setting for Dead Band Module
    *     -Select the input mode for Dead Band Module
    *     -Select the output mode for Dead Band Module
    *     -Select Polarity of the output PWMs
    */
    etpwmREG1->DBCTL = ((uint16)((uint16)0U << 5U)    /* Source for Falling edge
delay(0-PWMA, 1-PWMB) */
                       | (uint16)((uint16)0u << 4U)    /* Source for Rising edge
delay(0-PWMA, 1-PWMB)  */
                       | (uint16)((uint16)0U << 3U)    /* Enable/Disable EPWMxB
invert      */
                       | (uint16)((uint16)0U << 2U)    /* Enable/Disable EPWMxA
invert      */
                       | (uint16)((uint16)0U << 1U)    /* Enable/Disable Rising Edge
Delay   */
                       | (uint16)((uint16)0U << 0U));  /* Enable/Disable Falling Edge
Delay  */
```

35.4.4.1 Dead-Band Generator Control Register (DBCTL)

Figure 35-75. Dead-Band Generator Control Register (DBCTL) [offset = 1Ch]

15	14						8
HALFCYCLE	Reserved						
R/W-0	R-0						

7	6	5	4	3	2	1	0
Reserved		IN_MODE		POLSEL		OUT_MODE	
R-0		R/W-0		R/W-0		R/W-0	

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 35-35. Dead-Band Generator Control Register (DBCTL) Field Descriptions

Bits	Name	Value	Description
15	HALFCYCLE		Half Cycle Clocking Enable Bit.
		0	Full cycle clocking enabled. The dead-band counters are clocked at the TBCLK rate.
		1	Half cycle clocking enabled. The dead-band counters are clocked at TBCLK × 2.
14-6	Reserved	0	Reserved
5-4	IN_MODE		Dead Band Input Mode Control. Bit 5 controls the S5 switch and bit 4 controls the S4 switch shown in Figure 35-28. This allows you to select the input source to the falling-edge and rising-edge delay. To produce classical dead-band waveforms the default is EPWMxA In is the source for both falling and rising-edge delays.
		0	EPWMxA In (from the action-qualifier) is the source for both falling-edge and rising-edge delay.
		1h	EPWMxB In (from the action-qualifier) is the source for rising-edge delayed signal. EPWMxA In (from the action-qualifier) is the source for falling-edge delayed signal.
		2h	EPWMxA In (from the action-qualifier) is the source for rising-edge delayed signal. EPWMxB In (from the action-qualifier) is the source for falling-edge delayed signal.
		3h	EPWMxB In (from the action-qualifier) is the source for both rising-edge delay and falling-edge delayed signal.
3-2	POLSEL		Polarity Select Control. Bit 3 controls the S3 switch and bit 2 controls the S2 switch shown in Figure 35-28. This allows you to selectively invert one of the delayed signals before it is sent out of the dead-band submodule. The following descriptions correspond to classical upper/lower switch control as found in one leg of a digital motor control inverter. These assume that DBCTL[OUT_MODE] = 1,1 and DBCTL[IN_MODE] = 0,0. Other enhanced modes are also possible, but not regarded as typical usage modes.
		0	Active high (AH) mode. Neither EPWMxA nor EPWMxB is inverted (default).
		1h	Active low complementary (ALC) mode. EPWMxA is inverted.
		2h	Active high complementary (AHC). EPWMxB is inverted.
		3h	Active low (AL) mode. Both EPWMxA and EPWMxB are inverted.

해당 입력된 비트를 보면,
모두 0이 입력된다. IN_MODE 필드에 0으로 상승, 하강 엣지에 딜레이를 삽입하며, POLSEL이 0이므로 둘다 반전되어 나오지 않는다.

Table 35-35. Dead-Band Generator Control Register (DBCTL) Field Descriptions (continued)

Bits	Name	Value	Description
1-0	OUT_MODE		Dead-band Output Mode Control. Bit 1 controls the S1 switch and bit 0 controls the S0 switch shown in Figure 35-28. This allows you to selectively enable or bypass the dead-band generation for the falling-edge and rising-edge delay.
		0	Dead-band generation is bypassed for both output signals. In this mode, both the EPWMxA and EPWMxB output signals from the action-qualifier are passed directly to the PWM-chopper submodule. In this mode, the POLSEL and IN_MODE bits have no effect.
		1h	Disable rising-edge delay. The EPWMxA signal from the action-qualifier is passed straight through to the EPWMxA input of the PWM-chopper submodule. The falling-edge delayed signal is seen on output EPWMxB. The input signal for the delay is determined by DBCTL[IN_MODE].
		2h	The rising-edge delayed signal is seen on output EPWMxA. The input signal for the delay is determined by DBCTL[IN_MODE]. Disable falling-edge delay. The EPWMxB signal from the action-qualifier is passed straight through to the EPWMxB input of the PWM-chopper submodule.
		3h	Dead-band is fully enabled for both rising-edge delay on output EPWMxA and falling-edge delay on output EPWMxB. The input signal for the delay is determined by DBCTL[IN_MODE].

이와 동시에, dead-band 생성기는 output을 위해서 우회된다. 즉 AQ에서 나온 신호는 PWM-chopper 서브 모듈로 다이렉트로 전달된다. (따라서 dead-band 생성기를 사용하지 않는다는 의미이다. 사용하려면 OUT_MODE 필드에 적절한 값을 넣어야 한다.)

```
/** - Set the rising edge delay  */
etpwmREG1->DBRED = 110U;

/** - Set the falling edge delay  */
etpwmREG1->DBFED = 110U ;
```

두 레지스터는 Rising Edge와 Falling Edge의 크기를 정해주는 delay counter이다.
10-bit counter이므로, 0-9 비트만 사용한다. 0x0 ~ 0x1FF까지 크기를 정할 수 있다.

```
    /** - Enable the chopper module for ETPWMx
    *     -Sets the One shot pulse width in a chopper modulated wave
    *     -Sets the dutycycle for the subsequent pulse train
    *     -Sets the period for the subsequent pulse train
    */
    etpwmREG1->PCCTL = ((uint16)((uint16)0U << 0U)    /* Enable/Disable chopper
module */
                     | (uint16)((uint16)0U << 1U)    /* One-shot Pulse Width */
                     | (uint16)((uint16)3U << 8U)    /* Chopping Clock Duty Cycle
*/
                     | (uint16)((uint16)0U << 5U));  /* Chopping Clock Frequency */
```

PWM-chopper 서브 모듈 컨트롤 레지스터이다. 8, 9번 비트를 온 시킨다.

35.4.7 PWM-Chopper Submodule Register

35.4.7.1 PWM-Chopper Control Register (PCCTL)

Figure 35-90. PWM-Chopper Control Register (PCCTL) [offset = 3Eh]

15	11	10	8
Reserved		CHPDUTY	
R-0		R/W-0	

7	5	4	1	0
CHPFREQ		OSHTWTH		CHPEN
R/W-0		R/W-0		R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 35-50. PWM-Chopper Control Register (PCCTL) Bit Descriptions

Bits	Name	Value	Description
15-11	Reserved	0	Reserved
10-8	CHPDUTY		Chopping Clock Duty Cycle.
		0	Duty = 1/8 (12.5%)
		1h	Duty = 2/8 (25.0%)
		2h	Duty = 3/8 (37.5%)
		3h	Duty = 4/8 (50.0%)
		4h	Duty = 5/8 (62.5%)
		5h	Duty = 6/8 (75.0%)
		6h	Duty = 7/8 (87.5%)
		7h	Reserved
7-5	CHPFREQ		Chopping Clock Frequency.
		0	Divide by 1 (no prescale, = 12.5 MHz at 100 MHz VCLK3)
		1h	Divide by 2 (6.25 MHz at 100 MHz VCLK3)
		2h	Divide by 3 (4.16 MHz at 100 MHz VCLK3)
		3h	Divide by 4 (3.12 MHz at 100 MHz VCLK3)
		4h	Divide by 5 (2.50 MHz at 100 MHz VCLK3)
		5h	Divide by 6 (2.08 MHz at 100 MHz VCLK3)
		6h	Divide by 7 (1.78 MHz at 100 MHz VCLK3)
		7h	Divide by 8 (1.56 MHz at 100 MHz VCLK3)

CHPDUTY에 3h의 값을 입력하게 된다.

Chopping clock Duty cycle을 50%로 설정.

Table 35-50. PWM-Chopper Control Register (PCCTL) Bit Descriptions (continued)

Bits	Name	Value	Description
4-1	OSHTWTH		One-Shot Pulse Width.
		0	1 x VCLK3 / 8 wide (= 80 nS at 100 MHz VCLK3)
		1h	2 x VCLK3 / 8 wide (= 160 nS at 100 MHz VCLK3)
		2h	3 x VCLK3 / 8 wide (= 240 nS at 100 MHz VCLK3)
		3h	4 x VCLK3 / 8 wide (= 320 nS at 100 MHz VCLK3)
		4h	5 x VCLK3 / 8 wide (= 400 nS at 100 MHz VCLK3)
		5h	6 x VCLK3 / 8 wide (= 480 nS at 100 MHz VCLK3)
		6h	7 x VCLK3 / 8 wide (= 560 nS at 100 MHz VCLK3)
		7h	8 x VCLK3 / 8 wide (= 640 nS at 100 MHz VCLK3)
		8h	9 x VCLK3 / 8 wide (= 720 nS at 100 MHz VCLK3)
		9h	10 x VCLK3 / 8 wide (= 800 nS at 100 MHz VCLK3)
		Ah	11 x VCLK3 / 8 wide (= 880 nS at 100 MHz VCLK3)
		Bh	12 x VCLK3 / 8 wide (= 960 nS at 100 MHz VCLK3)
		Ch	13 x VCLK3 / 8 wide (= 1040 nS at 100 MHz VCLK3)
		Dh	14 x VCLK3 / 8 wide (= 1120 nS at 100 MHz VCLK3)
		Eh	15 x VCLK3 / 8 wide (= 1200 nS at 100 MHz VCLK3)
		Fh	16 x VCLK3 / 8 wide (= 1280 nS at 100 MHz VCLK3)
0	CHPEN		PWM-chopping Enable.
		0	Disable (bypass) PWM chopping function.
		1	Enable chopping function.

다만 CHPEN의 필드를 0으로 하기 때문에 PWM chopping 기능을 사용하지 않는다.

```
    /** - Set trip source enable */
    etpwmREG1->TZSEL = 0x0000U    /** - Enable/Disable TZ1 as a one-shot trip
source */
                     | 0x0000U    /** - Enable/Disable TZ2 as a one-shot trip source
*/
                     | 0x0000U    /** - Enable/Disable TZ3 as a one-shot trip source
*/
                     | 0x0000U    /** - Enable/Disable TZ4 as a one-shot trip source
*/
                     | 0x0000U    /** - Enable/Disable TZ5 as a one-shot trip source
*/
                     | 0x0000U    /** - Enable/Disable TZ6 as a one-shot trip source
*/
                     | 0x0000U    /** - Enable/Disable TZ1 as a CBC trip source
*/
                     | 0x0000U    /** - Enable/Disable TZ2 as a CBC trip source
*/
                     | 0x0000U    /** - Enable/Disable TZ3 as a CBC trip source
*/
                     | 0x0000U    /** - Enable/Disable TZ4 as a CBC trip source
*/
                     | 0x0000U    /** - Enable/Disable TZ5 as a CBC trip source
*/
                     | 0x0000U;   /** - Enable/Disable TZ6 as a CBC trip source
*/
```

Trip-Zone 선택 레지스터이다.
해당 필드를 보면,

35.4.5.2 Trip-Zone Select Register (TZSEL)

Figure 35-79. Trip-Zone Select Register (TZSEL) [offset = 26h]

15	14	13	12	11	10	9	8
DCBEVT1	DCAEVT1	OSHT6	OSHT5	OSHT4	OSHT3	OSHT2	OSHT1
R-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
DCBEVT2	DCAEVT2	CBC6	CBC5	CBC4	CBC3	CBC2	CBC1
R-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 35-39. Trip-Zone Submodule Select Register (TZSEL) Field Descriptions

Bits	Name	Value	Description
One-Shot (OSHT) Trip-zone enable/disable. When any of the enabled pins go low, a one-shot trip event occurs for this ePWM module. When the event occurs, the action defined in the TZCTL register is taken on the EPWMxA and EPWMxB outputs. The one-shot trip condition remains latched until the user clears the condition via the TZCLR register.			
15	DCBEVT1		Digital Compare Output B Event 1 Select.
		0	Disable DCBEVT1 as one-shot-trip source for this ePWM module.
		1	Enable DCBEVT1 as one-shot-trip source for this ePWM module.
14	DCAEVT1		Digital Compare Output A Event 1 Select.
		0	Disable DCAEVT1 as one-shot-trip source for this ePWM module.
		1	Enable DCAEVT1 as one-shot-trip source for this ePWM module.
13	OSHT6		Trip-zone 6 ($\overline{TZ6}$) Select.
		0	Disable $\overline{TZ6}$ as a one-shot trip source for this ePWM module.
		1	Enable $\overline{TZ6}$ as a one-shot trip source for this ePWM module.
12	OSHT5		Trip-zone 5 ($\overline{TZ5}$) Select.
		0	Disable $\overline{TZ5}$ as a one-shot trip source for this ePWM module.
		1	Enable $\overline{TZ5}$ as a one-shot trip source for this ePWM module.
11	OSHT4		Trip-zone 4 ($\overline{TZ4}$) Select.
		0	Disable $\overline{TZ4}$ as a one-shot trip source for this ePWM module.
		1	Enable $\overline{TZ4}$ as a one-shot trip source for this ePWM module.
10	OSHT3		Trip-zone 3 ($\overline{TZ3}$) Select.
		0	Disable $\overline{TZ3}$ as a one-shot trip source for this ePWM module.
		1	Enable $\overline{TZ3}$ as a one-shot trip source for this ePWM module.
9	OSHT2		Trip-zone 2 ($\overline{TZ2}$) Select.
		0	Disable $\overline{TZ2}$ as a one-shot trip source for this ePWM module.
		1	Enable $\overline{TZ2}$ as a one-shot trip source for this ePWM module.
8	OSHT1		Trip-zone 1 ($\overline{TZ1}$) Select.
		0	Disable $\overline{TZ1}$ as a one-shot trip source for this ePWM module.
		1	Enable $\overline{TZ1}$ as a one-shot trip source for this ePWM module.
Cycle-by-Cycle (CBC) Trip-zone enable/disable. When any of the enabled pins go low, a cycle-by-cycle trip event occurs for this ePWM module. When the event occurs, the action defined in the TZCTL register is taken on the EPWMxA and EPWMxB outputs. A cycle-by-cycle trip condition is automatically cleared when the time-base counter reaches zero.			
7	DCBEVT2		Digital Compare Output B Event 2 Select.
		0	Disable DCBEVT2 as a CBC trip source for this ePWM module.
		1	Enable DCBEVT2 as a CBC trip source for this ePWM module.
6	DCAEVT2		Digital Compare Output A Event 2 Select.
		0	Disable DCAEVT2 as a CBC trip source for this ePWM module.
		1	Enable DCAEVT2 as a CBC trip source for this ePWM module.

Table 35-39. Trip-Zone Submodule Select Register (TZSEL) Field Descriptions (continued)

Bits	Name	Value	Description
5	CBC6		Trip-zone 6 ($\overline{TZ6}$) Select.
		0	Disable $\overline{TZ6}$ as a CBC trip source for this ePWM module.
		1	Enable $\overline{TZ6}$ as a CBC trip source for this ePWM module.
4	CBC5		Trip-zone 5 ($\overline{TZ5}$) Select.
		0	Disable $\overline{TZ5}$ as a CBC trip source for this ePWM module.
		1	Enable $\overline{TZ5}$ as a CBC trip source for this ePWM module.
3	CBC4		Trip-zone 4 ($\overline{TZ4}$) Select.
		0	Disable $\overline{TZ4}$ as a CBC trip source for this ePWM module.
		1	Enable $\overline{TZ4}$ as a CBC trip source for this ePWM module.
2	CBC3		Trip-zone 3 ($\overline{TZ3}$) Select.
		0	Disable $\overline{TZ3}$ as a CBC trip source for this ePWM module.
		1	Enable $\overline{TZ3}$ as a CBC trip source for this ePWM module.
1	CBC2		Trip-zone 2 ($\overline{TZ2}$) Select.
		0	Disable $\overline{TZ2}$ as a CBC trip source for this ePWM module.
		1	Enable $\overline{TZ2}$ as a CBC trip source for this ePWM module.
0	CBC1		Trip-zone 1 ($\overline{TZ1}$) Select.
		0	Disable $\overline{TZ1}$ as a CBC trip source for this ePWM module.
		1	Enable $\overline{TZ1}$ as a CBC trip source for this ePWM module.

One-Shot(OSHT) 기능과 Cycle-by-Cycle(CBC) trip기능을 수행하게 한다.
OSHT와 CBC는 이 레지스터에서 enable한 핀이 low상태로 변할 때, 해당 ePWM 모듈에 해당 trip기능을 수행한다. 그리고 TZCTL 레지스터에 정의된 기능이 EPWMxAB에 수행되게 된다.

OSHT는 유저가 TZCLR을 통하여 0으로 클리어 할 때까지 OSHT의 trip상태가 유지된다.
CBC는 time-base counter가 0이 되는 순간에 자동적으로 trip 상태가 초기화 된다.

```
    /** - Set interrupt enable */
    etpwmREG1->TZEINT = 0x0000U     /** - Enable/Disable Digital Comparator
Output A Event 1  */
                      | 0x0000U     /** - Enable/Disable Digital Comparator Output
A Event 2  */
                      | 0x0000U     /** - Enable/Disable Digital Comparator Output
A Event 1  */
                      | 0x0000U     /** - Enable/Disable Digital Comparator Output
A Event 2  */
                      | 0x0000U     /** - Enable/Disable one-shot interrupt
generation       */
                      | 0x0000U;    /** - Enable/Disable cycle-by-cycle interrupt
generation  */
```

35.4.5.3 Trip-Zone Enable Interrupt Register (TZEINT)

Figure 35-80. Trip-Zone Enable Interrupt Register (TZEINT) [offset = 28h]

15							8
Reserved							
R -0							
7	6	5	4	3	2	1	0
Reserved	DCBEVT2	DCBEVT1	DCAEVT2	DCAEVT1	OST	CBC	Reserved
R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 35-40. Trip-Zone Enable Interrupt Register (TZEINT) Field Descriptions

Bits	Name	Value	Description
15-3	Reserved	0	Reserved
6	DCBEVT2		Digital Comparator Output B Event 2 Interrupt Enable.
		0	Disabled
		1	Enabled
5	DCBEVT1		Digital Comparator Output B Event 1 Interrupt Enable.
		0	Disabled
		1	Enabled
4	DCAEVT2		Digital Comparator Output A Event 2 Interrupt Enable.
		0	Disabled
		1	Enabled
3	DCAEVT1		Digital Comparator Output A Event 1 Interrupt Enable.
		0	Disabled
		1	Enabled
2	OST		Trip-zone One-Shot Interrupt Enable.
		0	Disable one-shot interrupt generation.
		1	Enable Interrupt generation; a one-shot trip event will cause a EPWMx_TZINT VIM interrupt.
1	CBC		Trip-zone Cycle-by-Cycle Interrupt Enable.
		0	Disable cycle-by-cycle interrupt generation.
		1	Enable interrupt generation; a cycle-by-cycle trip event will cause an EPWMx_TZINT VIM interrupt.
0	Reserved	0	Reserved

해당 레지스터는 DCBEVT1,2 DCAEVT1,2Digital comparator 이벤트 enable 및 OHST, CBC를 인터럽트로 동작하게 하는 레지스터이다.

TZCTL과 TZCLR로 해당 digital comparator의 action과 세팅을 해 줄 수 있다.
(+ TZFLG, TZFRC)

```
    /** - Sets up the event for interrupt */
    etpwmREG1->ETSEL = (uint16)NO_EVENT;

    if ((etpwmREG1->ETSEL & 0x0007U) != 0U)
    {
        etpwmREG1->ETSEL |= 0x0008U;
    }
    /** - Setup the frequency of the interrupt generation */
    etpwmREG1->ETPS = 1U;

    /** - Sets up the ADC SOC interrupt */
    etpwmREG1->ETSEL |= ((uint16)(0x0000U)
                      | (uint16)(0x0000U)
                      | (uint16)((uint16)DCAEVT1 << 8U)
                      | (uint16)((uint16)DCBEVT1 << 12U));

    /** - Sets up the ADC SOC period */
    etpwmREG1->ETPS |= ((uint16)((uint16)1U << 8U)
```

```
                    | (uint16)((uint16)1U << 12U));


/* USER CODE BEGIN (2) */
/* USER CODE END */
}
```

ETSEL의 기능은 이벤트 트리거를 선택하는 레지스터이다. NO_EVENT는 0을 가리킨다.

TB, CC에서 생성하는 이벤트(DC counter등)를 어떻게 사용할지 선택한다.

각각의 필드는 ePWM 인터럽트 신호 발생 유/무, 언제 인터럽트가 발생하게 할건지(주기, CMPA등등) DC counter 사용 유/무, ADC 시작신호 사용 유/무 등이 있다.

```
if ((etpwmREG1->ETSEL & 0x0007U) != 0U)
    {
        etpwmREG1->ETSEL |= 0x0008U;
    }
```

의 기능은 ETSEL[INTSEL] 레지스터의 EPWM interrupt 선택 옵션으로 주기, CMPA등등 의 경우 이벤트를 발생 enable 레지스터로 해당 비트가 0이 아니라면 (enable event 한 경우)

ETSEL[INTEN] 에서 interrupt 생성을 하도록 세팅을 해주는 동작이다.

또한 DCBEVT1.soc과 DCAEVT1.soc의 기능 사용한다고 선언하였다.

```
etpwmREG1->ETPS = 1U;
```
이벤트 카운터 인터럽트를 해제, 해당 필드는 ETPS[INTCNT]와 함께 이벤트가 몇번 발생하야 인터럽트가 동작하는지 설정하는 곳이다.

```
    /** - Sets up the ADC SOC interrupt */
    etpwmREG1->ETSEL |= ((uint16)(0x0000U)
                      | (uint16)(0x0000U)
                      | (uint16)((uint16)DCAEVT1 << 8U)
                      | (uint16)((uint16)DCBEVT1 << 12U));

    /** - Sets up the ADC SOC period */
    etpwmREG1->ETPS |= ((uint16)((uint16)1U << 8U)
```
해당 동작은 ADC 시작 이벤트를 disable 하였다.

여기도 마찬가지로 이벤트 횟수에 따른 인터럽트 동작을 설정할 수 있다.

35.4.6 Event-Trigger Submodule Registers

35.4.6.1 Event-Trigger Selection Register (ETSEL)

Figure 35-85. Event-Trigger Selection Register (ETSEL) [offset = 30h]

15	14 12	11	10 8
SOCBEN	SOCBSEL	SOCAEN	SOCASEL
R/W-0	R/W-0	R/W-0	R/W-0

7 4	3	2 0
Reserved	INTEN	INTSEL
R-0	R/W-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Table 35-45. Event-Trigger Selection Register (ETSEL) Field Descriptions

Bits	Name	Value	Description
15	SOCBEN		Enable the ADC Start of Conversion B (EPWMxSOCB) Pulse.
		0	Disable EPWMxSOCB.
		1	Enable EPWMxSOCB pulse.
14-12	SOCBSEL		EPWMxSOCB Selection Options.
			These bits determine when a EPWMxSOCB pulse will be generated.
		0	Enable DCBEVT1.soc event.
		1h	Enable event time-base counter equal to zero. (TBCTR = 0x0000).
		2h	Enable event time-base counter equal to period (TBCTR = TBPRD).
		3h	Enable event time-base counter equal to zero or period (TBCTR = 0x0000 or TBCTR = TBPRD). This mode is useful in up-down count mode.
		4h	Enable event time-base counter equal to CMPA when the timer is incrementing.
		5h	Enable event time-base counter equal to CMPA when the timer is decrementing.
		6h	Enable event: time-base counter equal to CMPB when the timer is incrementing.
		7h	Enable event: time-base counter equal to CMPB when the timer is decrementing.
11	SOCAEN		Enable the ADC Start of Conversion A (EPWMxSOCA) Pulse.
		0	Disable EPWMxSOCA.
		1	Enable EPWMxSOCA pulse.
10-8	SOCASEL		EPWMxSOCA Selection Options.
			These bits determine when a EPWMxSOCA pulse will be generated.
		0	Enable DCAEVT1.soc event.
		1h	Enable event time-base counter equal to zero. (TBCTR = 0x0000).
		2h	Enable event time-base counter equal to period (TBCTR = TBPRD).
		3h	Enable event time-base counter equal to zero or period (TBCTR = 0x0000 or TBCTR = TBPRD). This mode is useful in up-down count mode.
		4h	Enable event time-base counter equal to CMPA when the timer is incrementing.
		5h	Enable event time-base counter equal to CMPA when the timer is decrementing.
		6h	Enable event: time-base counter equal to CMPB when the timer is incrementing.
		7h	Enable event: time-base counter equal to CMPB when the timer is decrementing.
7-4	Reserved	0	Reserved
3	INTEN		Enable ePWM Interrupt (EPWMx_INT) Generation.
		0	Disable EPWMx_INT generation.
		1	Enable EPWMx_INT generation.

Table 35-45. Event-Trigger Selection Register (ETSEL) Field Descriptions (continued)

Bits	Name	Value	Description
2-0	INTSEL		ePWM Interrupt (EPWMx_INT) Selection Options.
		0	Reserved
		1h	Enable event time-base counter equal to zero. (TBCTR = 0x0000).
		2h	Enable event time-base counter equal to period (TBCTR = TBPRD).
		3h	Enable event time-base counter equal to zero or period (TBCTR = 0x0000 or TBCTR = TBPRD). This mode is useful in up-down count mode.
		4h	Enable event time-base counter equal to CMPA when the timer is incrementing.
		5h	Enable event time-base counter equal to CMPA when the timer is decrementing.
		6h	Enable event: time-base counter equal to CMPB when the timer is incrementing.
		7h	Enable event: time-base counter equal to CMPB when the timer is decrementing.

35.4.6.3 Event-Trigger Prescale Register (ETPS)

Figure 35-87. Event-Trigger Prescale Register (ETPS) [offset = 36h]

15 14	13 12	11 10	9 8
SOCBCNT	SOCBPRD	SOCACNT	SOCAPRD
R-0	R/W-0	R-0	R/W-0

7 4	3 2	1 0
Reserved	INTCNT	INTPRD
R-0	R-0	R/W-0

LEGEND: R/W = Read/Write; R = Read only; *-n* = value after reset

Table 35-47. Event-Trigger Prescale Register (ETPS) Field Descriptions

Bits	Name	Value	Description
15-14	SOCBCNT		ePWM ADC Start-of-Conversion B Event (EPWMxSOCB) Counter Register.
			These bits indicate how many selected ETSEL[SOCBSEL] events have occurred.
		0	No events have occurred.
		1h	1 event has occurred.
		2h	2 events have occurred.
		3h	3 events have occurred.
13-12	SOCBPRD		ePWM ADC Start-of-Conversion B Event (EPWMxSOCB) Period Select.
			These bits determine how many selected ETSEL[SOCBSEL] events need to occur before an EPWMxSOCB pulse is generated. To be generated, the pulse must be enabled (ETSEL[SOCBEN] = 1). The SOCB pulse will be generated even if the status flag is set from a previous start of conversion (ETFLG[SOCB] = 1). Once the SOCB pulse is generated, the ETPS[SOCBCNT] bits will automatically be cleared.
		0	Disable the SOCB event counter. No EPWMxSOCB pulse will be generated.
		1h	Generate the EPWMxSOCB pulse on the first event: ETPS[SOCBCNT] = 0,1.
		2h	Generate the EPWMxSOCB pulse on the second event: ETPS[SOCBCNT] = 1,0.
		3h	Generate the EPWMxSOCB pulse on the third event: ETPS[SOCBCNT] = 1,1.
11-10	SOCACNT		ePWM ADC Start-of-Conversion A Event (EPWMxSOCA) Counter Register.
			These bits indicate how many selected ETSEL[SOCASEL] events have occurred.
		0	No events have occurred.
		1h	1 event has occurred.
		2h	2 events have occurred.
		3h	3 events have occurred.
9-8	SOCAPRD		ePWM ADC Start-of-Conversion A Event (EPWMxSOCA) Period Select.
			These bits determine how many selected ETSEL[SOCASEL] events need to occur before an EPWMxSOCA pulse is generated. To be generated, the pulse must be enabled (ETSEL[SOCAEN] = 1). The SOCA pulse will be generated even if the status flag is set from a previous start of conversion (ETFLG[SOCA] = 1). Once the SOCA pulse is generated, the ETPS[SOCACNT] bits will automatically be cleared.
		0	Disable the SOCA event counter. No EPWMxSOCA pulse will be generated.
		1h	Generate the EPWMxSOCA pulse on the first event: ETPS[SOCACNT] = 0,1.
		2h	Generate the EPWMxSOCA pulse on the second event: ETPS[SOCACNT] = 1,0.
		3h	Generate the EPWMxSOCA pulse on the third event: ETPS[SOCACNT] = 1,1.

Table 35-47. Event-Trigger Prescale Register (ETPS) Field Descriptions (continued)

Bits	Name	Value	Description
3-2	INTCNT		ePWM Interrupt Event (EPWMx_INT) Counter Register. These bits indicate how many selected ETSEL[INTSEL] events have occurred. These bits are automatically cleared when an interrupt pulse is generated. If interrupts are disabled, ETSEL[INT] = 0 or the interrupt flag is set, ETFLG[INT] = 1, the counter will stop counting events when it reaches the period value ETPS[INTCNT] = ETPS[INTPRD].
		0	No events have occurred.
		1h	1 event has occurred.
		2h	2 events have occurred.
		3h	3 events have occurred.
1-0	INTPRD		ePWM Interrupt (EPWMx_INT) Period Select. These bits determine how many selected ETSEL[INTSEL] events need to occur before an interrupt is generated. To be generated, the interrupt must be enabled (ETSEL[INT] = 1). If the interrupt status flag is set from a previous interrupt (ETFLG[INT] = 1) then no interrupt will be generated until the flag is cleared via the ETCLR[INT] bit. This allows for one interrupt to be pending while another is still being serviced. Once the interrupt is generated, the ETPS[INTCNT] bits will automatically be cleared. Writing a INTPRD value that is the same as the current counter value will trigger an interrupt if it is enabled and the status flag is clear. Writing a INTPRD value that is less than the current counter value will result in an undefined state. If a counter event occurs at the same instant as a new zero or non-zero INTPRD value is written, the counter is incremented.
		0	Disable the interrupt event counter. No interrupt will be generated and ETFRC[INT] is ignored.
		1h	Generate an interrupt on the first event INTCNT = 01 (first event).
		2h	Generate interrupt on ETPS[INTCNT] = 1,0 (second event).
		3h	Generate interrupt on ETPS[INTCNT] = 1,1 (third event).

```
void etpwmStartTBCLK(void);
```

```
void etpwmStartTBCLK(void)
{
    /* Enable Pin Muxing */
    pinMuxReg->KICKER0 = 0x83E70B13U;
    pinMuxReg->KICKER1 = 0x95A4F1E0U;

        pinMuxReg->PINMUX[166U] = (pinMuxReg->PINMUX[166U] &
PINMUX_ETPWM_TBCLK_SYNC_MASK) | (PINMUX_ETPWM_TBCLK_SYNC_ON);

    /* Disable Pin Muxing */
    pinMuxReg->KICKER0 = 0x00000000U;
    pinMuxReg->KICKER1 = 0x00000000U;
}
```

6.7.3 KICK_REG0: Kicker Register 0

This register forms the first part of the unlock sequence for being able to update the I/O multiplexing control registers (PINMMRnn).

Figure 6-12. KICK_REG0: Kicker Register 0 (Offset = 38h)

31	16
KICK0	
R/W-0	

15	0
KICK0	
R/W-0	

LEGEND: R/W = Read/Write; *-n* = value after reset

Table 6-15. Kicker Register 0 Field Descriptions

Bit	Field	Description
31-0	KICK0	Kicker 0 Register. The value 83E7 0B13h must be written to KICK0 as part of the process to unlock the CPU write access to the PINMMRnn registers.

해당 KICKER 레지스터에 지정된 값 `0x83E70B13U` 을 삽입 하여 CPU가 Pin MUX 레지스터에 접근할 수 있도록 허가를 내준다.

따라서 `pinMuxReg->PINMUX[166U]` 에 접근이 가능하다.

해당 pinMux 설정으로 gpio핀에 etPWM을 연결한다.

그리고 다시 KICKER 레지스터를 닫아서 Mux 설정을 건드리지 못하게 한다.

```
void etpwmStopTBCLK(void);
```

```
void etpwmStopTBCLK(void)
{
    /* Enable Pin Muxing */
    pinMuxReg->KICKER0 = 0x83E70B13U;
    pinMuxReg->KICKER1 = 0x95A4F1E0U;

        pinMuxReg->PINMUX[166U] = (pinMuxReg->PINMUX[166U] &
PINMUX_ETPWM_TBCLK_SYNC_MASK) | (PINMUX_ETPWM_TBCLK_SYNC_OFF);

    /* Disable Pin Muxing */
    pinMuxReg->KICKER0 = 0x00000000U;
    pinMuxReg->KICKER1 = 0x00000000U;
}
```

위의 함수와 똑 같은 작동을 하지만 Mux 설정을 0으로 만들어서 gpio와 etPWM 신호 연결을 막는다.

```
void etpwmSetClkDiv(etpwmBASE_t *etpwm, etpwmClkDiv_t clkdiv, etpwmHspClkDiv_t hspclkdiv);
```

```
void etpwmSetClkDiv(etpwmBASE_t *etpwm, etpwmClkDiv_t clkdiv, etpwmHspClkDiv_t
hspclkdiv)
{
    etpwm->TBCTL &= (uint16)~(uint16)0x1F80U;
    etpwm->TBCTL |= (uint16)clkdiv | (uint16)hspclkdiv;
}
```

리턴값은 void형.

매개변수는 etpwm 모듈 이름, clock 분주 값, hsp 클락 분주 값을 넣어준다.

TBCTL 레지스터의 7 ~ 12 비트의 값을 0으로 초기화 한다.

12-10	CLKDIV		Time-base Clock Prescale Bits. These bits determine part of the time-base clock prescale value: TBCLK = VCLK3 / (HSPCLKDIV × CLKDIV)
		0	/1 (default on reset)
		1h	/2
		2h	/4
		3h	/8
		4h	/16
		5h	/32
		6h	/64
		7h	/128
9-7	HSPCLKDIV		High Speed Time-base Clock Prescale Bits. These bits determine part of the time-base clock prescale value: TBCLK = VCLK3 / (HSPCLKDIV × CLKDIV)
		0	/1
		1h	/2 (default on reset)
		2h	/4
		3h	/6
		4h	/8
		5h	/10
		6h	/12
		7h	/14

해당 레지스터의 필드를 초기화 하여 기존 VCLK의 값을 그대로 초기화를 한다.

이후에 매개변수로 입력한 clkdiv, hspclkdiv의 값을 토대로 분주를 다시 결정해준다.

값의 범위는 각각 0~7의 값을 넣어줘야 한다.

```
void etpwmSetTimebasePeriod(etpwmBASE_t *etpwm, uint16 period);
```

```
void etpwmSetTimebasePeriod(etpwmBASE_t *etpwm, uint16 period)
{
    etpwm->TBPRD = period;
}
```

35.4.1.4 Time-Base Period Register (TBPRD)

Figure 35-66. Time-Base Period Register (TBPRD) [offset = 08h]

15	0
TBPRD	
R/W-0	

LEGEND: R/W = Read/Write; *-n* = value after reset

Table 35-26. Time-Base Period Register (TBPRD) Field Descriptions

Bits	Name	Description
15-0	TBPRD	These bits determine the period of the time-base counter. This sets the PWM frequency. Shadowing of this register is enabled and disabled by the TBCTL[PRDLD] bit. By default this register is shadowed. • If TBCTL[PRDLD] = 0, then the shadow is enabled and any write or read will automatically go to the shadow register. In this case, the active register will be loaded from the shadow register when the time-base counter equals 0. • If TBCTL[PRDLD] = 1, then the shadow is disabled and any write or read will go directly to the active register, that is the register actively controlling the hardware. • The active and shadow registers share the same memory map address.

리턴형은 없다.

매개변수는 PWM 모듈번호와 주기를 넣는다.

해당 TBPRD 레지스터의 주기를 결정 짓는다.

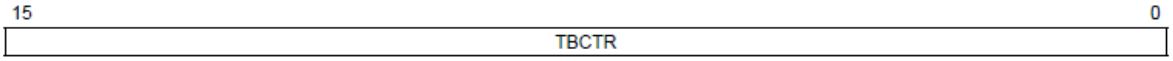
계산방법은 TBCLK가 1주기로 몇 번 동작하는지 넣어 주는 것이다.

```
void etpwmSetCount(etpwmBASE_t *etpwm, uint16 count);
```

```
void etpwmSetCount(etpwmBASE_t *etpwm, uint16 count)
{
    etpwm->TBCTR = count;
}
```

35.4.1.5 Time-Base Counter Register (TBCTR)

Figure 35-67. Time-Base Counter Register (TBCTR) [offset = 0Ah]



LEGEND: R/W = Read/Write; *-n* = value after reset

Table 35-27. Time-Base Counter Register (TBCTR) Field Descriptions

Bits	Name	Description
15-0	TBCTR	Reading these bits gives the current time-base counter value. Writing to these bits sets the current time-base counter value. The update happens as soon as the write occurs; the write is NOT synchronized to the time-base clock (TBCLK) and the register is not shadowed.

리턴형은 void

매개변수는 모듈 넘버와 지정하고 싶은 카운트 숫자를 넣으면 된다.

해당 카운터 레지스터를 통하여 현재 카운터를 수정할 수 있다.

TBCLK의 클록 만큼 카운터가 증가하게 되는데 이를 수정해 주는 것이다.

```
void etpwmDisableTimebasePeriodShadowMode(etpwmBASE_t *etpwm);
```

```
void etpwmDisableTimebasePeriodShadowMode(etpwmBASE_t *etpwm)
{
    etpwm->TBCTL |= 0x0008U;
}
```

리턴형은 void

매개변수는 PWM 모듈 번호만 넣으면 된다.

3번째 비트를 ON 시킨다.

3	PRDLD		Active Period Register Load From Shadow Register Select.
		0	The period register (TBPRD) is loaded from its shadow register when the time-base counter, TBCTR, is equal to zero. A write or read to the TBPRD register accesses the shadow register.
		1	Load the TBPRD register immediately without using a shadow register. A write or read to the TBPRD register directly accesses the active register.

해당 TBCLT[PRDLD] 를 1로 세트하여 shadow register 사용없이 바로 active register에 접근 할 수 있도록 한다.

요약하자면, etpwm 의 shadowMode를 disable 하는 것이다.

```
void etpwmEnableTimebasePeriodShadowMode(etpwmBASE_t *etpwm);
```

```
void etpwmEnableTimebasePeriodShadowMode(etpwmBASE_t *etpwm)
{
    etpwm->TBCTL &= (uint16)~(uint16)0x0008U;
}
```

리턴형은 void 매개변수는 모듈 번호만 들어간다.

3번째 비트를 무조건 0으로 만든다.

위의 필드를 확인하면, TBPRD 레지스터가 shadow register에 접근하도록 한다.

```
void etpwmEnableCounterLoadOnSync(etpwmBASE_t *etpwm, uint16 phase, uint16 direction);
```

```
void etpwmEnableCounterLoadOnSync(etpwmBASE_t *etpwm, uint16 phase, uint16
direction)
{
    etpwm->TBCTL &= (uint16)~(uint16)0x2000U;
    etpwm->TBCTL |= 0x0004U | direction;
    etpwm->TBPHS = phase;
}
```

리턴형은 void 이며, 매개변수는 모듈번호, 위상, 방향이 들어간다.

TBCTL에 13번째 비트를 0으로 클리어, 2번째 비트를 ON 그리고 direction 값이 들어 감.

TBPHS에 phase 값이 들어 감.

13	PHSDIR		Phase Direction Bit. This bit is only used when the time-base counter is configured in the up-down-count mode. The PHSDIR bit indicates the direction the time-base counter (TBCTR) will count after a synchronization event occurs and a new phase value is loaded from the phase (TBPHS) register. This is irrespective of the direction of the counter before the synchronization event.. In the up-count and down-count modes this bit is ignored.
		0	Count down after the synchronization event.
		1	Count up after the synchronization event.

13 비트를 0으로 하기에 sync 이벤트 발생 이후에(동기화 후) counter down으로 동작하게 한다.

(Up-Down count 모드에서만 동작한다. sync 이벤트 이전 counter 방향은 상관이 없다. sync 이벤트 발생 이후에 새로운 위상 값이 TBPHS 레지스터에서 로드 된다.)

2	PHSEN		Counter Register Load From Phase Register Enable.
		0	Do not load the time-base counter (TBCTR) from the time-base phase register (TBPHS).
		1	Load the time-base counter with the phase register when an EPWMxSYNCI input signal occurs or when a software synchronization is forced by the SWFSYNC bit, or when a digital compare sync event occurs.

2번 비트를 ON 시키기에 EPWMxSYNCI input signal // SWFSYNC bit // digital compare sync event 가 발생한 경우 time-base counter에 phase register을 로드 한다.

35.4.1.3 Time-Base Phase Register (TBPHS)

Figure 35-65. Time-Base Phase Register (TBPHS) [offset = 04h]

15	0
TBPHS	
R/W-0	

LEGEND: R/W = Read/Write; *-n* = value after reset

Table 35-25. Time-Base Phase Register (TBPHS) Field Descriptions

Bits	Name	Description
15-0	TBPHS	These bits set time-base counter phase of the selected ePWM relative to the time-base that is supplying the synchronization input signal. • If TBCTL[PHSEN] = 0, then the synchronization event is ignored and the time-base counter is not loaded with the phase. • If TBCTL[PHSEN] = 1, then the time-base counter (TBCTR) will be loaded with the phase (TBPHS) when a synchronization event occurs. The synchronization event can be initiated by the input synchronization signal (EPWMxSYNCI) or by a software forced synchronization.

해당 레지스터에 위상 값을 넣는다.

동기화 입력 신호를 공급하는 시간 축에 상대적으로 선택한 ePWM의 시간 기준 위상을 설정.

TBCTL[PHSEN] = 1 이어야 한다. input synchronization signal (EPWMxSYNCI)에 의한 이벤트나 강제 sync 에 의해서 해당 레지스터의 위상 값이 TBCTR에 로드 된다.

direction은 2를 넣어주면 될 것 같다. up-down-count mode 동작을 위해서

```
void etpwmDisableCounterLoadOnSync(etpwmBASE_t *etpwm);
```

```
void etpwmDisableCounterLoadOnSync(etpwmBASE_t *etpwm)
{
    etpwm->TBCTL &= (uint16)~(uint16)0x0004U;
}
```

해당 함수에서 리턴 값은 void, 매개변수는 해당 모듈을 삽입하면 된다.
비트 2번째를 무조건 0으로 클리어 하는 명령이다.

2	PHSEN		Counter Register Load From Phase Register Enable.
		0	Do not load the time-base counter (TBCTR) from the time-base phase register (TBPHS).
		1	Load the time-base counter with the phase register when an EPWMxSYNCI input signal occurs or when a software synchronization is forced by the SWFSYNC bit, or when a digital compare sync event occurs.

따라서 위상 레지스터 (TBPHS)에서 TBCTR 레지스터로 로드 하지 않는다. (disable)

```
void etpwmSetSyncOut(etpwmBASE_t *etpwm, etpwmSyncOut_t syncOutSrc)
```

```
void etpwmSetSyncOut(etpwmBASE_t *etpwm, etpwmSyncOut_t syncOutSrc)
{
    etpwm->TBCTL &= (uint16)~(uint16)0x0030U;
    etpwm->TBCTL |= syncOutSrc;
}
```

리턴 값은 void , 매개변수는 etpwmSyncOut_t 타입의 변수가 들어온다.

```
typedef enum
{
    SyncOut_EPWMxSYNCI = 0x00U, /** EPWMxSYNCI               */
    SyncOut_CtrEqZero  = 0x10U, /** CTR = zero               */
    SyncOut_CtrEqCmpB  = 0x20U, /** CTR = CMPB               */
    SyncOut_Disable    = 0x30U  /** Disable EPWMxSYNCO signal */
} etpwmSyncOut_t;
```

etpwmSyncOut_t 매개 변수는 위와 같은 타입을 가지고 있다.

4, 5 번 비트를 0으로 클리어 하고, syncOutSrc의 변수(0x00, 0x10, 0x20, 0x30)를 집어 넣는다. (즉 아래 레지스터를 0 초기화 하고 syncOutSrc의 변수로 동기화 출력 신호를 선택한다.)

5-4	SYNCOSEL		Synchronization Output Select. These bits select the source of the EPWMxSYNCO signal.
		0	EPWMxSYNC
		1h	CTR = zero: Time-base counter equal to zero (TBCTR = 0x0000)
		2h	CTR = CMPB : Time-base counter equal to counter-compare B (TBCTR = CMPB)
		3h	Disable EPWMxSYNCO signal

동기화 출력 신호를 선택 할 수 있다. 4, 5 비트 0 클리어 시 EPWMxSYNC signal를 선택한다. 아니면, zero 상태에 발생하거나 cmpb 상태에 발생하는 EPWMxSYNCO signal. 혹은 disable 할 수 있다.

```
void etpwmSetCounterMode(etpwmBASE_t *etpwm, etpwmCounterMode_t countermode);
```

```
void etpwmSetCounterMode(etpwmBASE_t *etpwm, etpwmCounterMode_t countermode)
{
    etpwm->TBCTL &= (uint16)~(uint16)0x0003U;
    etpwm->TBCTL |= countermode;
}
```

리턴값 void, 매개변수는 모듈 넘버와 countermode 값이다.

0,1 비트 클리어, countermode 를 TBCLT에 추가.

```
typedef enum
{
    CounterMode_Up     = 0U, /** Up-count mode                */
    Countermode_Down   = 1U, /** Down-count mode               */
    CounterMode_UpDown = 2U, /** Up-down-count mode            */
    CounterMode_Stop   = 3U  /** Stop - freeze counter operaton */
} etpwmCounterMode_t;
```

카운터 모드의 변수는 0, 1, 2, 3 이 존재한다.

1-0	CTRMODE		Counter Mode. The time-base counter mode is normally configured once and not changed during normal operation. If you change the mode of the counter, the change will take effect at the next TBCLK edge and the current counter value shall increment or decrement from the value before the mode change. These bits set the time-base counter mode of operation as follows:
		0	Up-count mode
		1h	Down-count mode
		2h	Up-down-count mode
		3h	Stop-freeze counter operation (default on reset)

해당 TBCTL[CTRMODE] 의 값을 0으로 초기화 한다.

이후에 countermode 의 매개 변수를 이용하여

평상시 모드에서 count 모드를 변경 시킨다.

up/ down/ up-down/ stop-freeze counter 모드가 있다.

```
void etpwmTriggerSWSync(etpwmBASE_t *etpwm)
```

```
void etpwmTriggerSWSync(etpwmBASE_t *etpwm)
{
    etpwm->TBCTL |= 0x0040U;
}
```

6번 비트를 활성화 시킨다.

6	SWFSYNC		Software Forced Synchronization Pulse.
		0	Writing a 0 has no effect and reads always return a 0.
		1	Writing a 1 forces a one-time synchronization pulse to be generated. This event is ORed with the EPWMxSYNCI input of the ePWM module. SWFSYNC is valid (operates) only when EPWMxSYNCI is selected by SYNCOSEL = 00.

해당 강제 one-time 동기화 펄스를 소프트웨어 적으로 삽입한다.

해당 이벤트는 ePWM 모듈의 EPWMxSYNCI 펄스 입력과 OR 작용을 한다.

그리고 EPWMxSYNCI가 TBCTL[SYNCOSEL] == 00 인 경우에만 동작한다.

```
void etpwmSetRunMode(etpwmBASE_t *etpwm, etpwmRunMode_t runmode);

void etpwmSetRunMode(etpwmBASE_t *etpwm, etpwmRunMode_t runmode)
{
    etpwm->TBCTL &= (uint16)~(uint16)0xC000U;
    etpwm->TBCTL |= runmode;
}
```

리턴형은 void

14, 15 비트를 0으로 클리어 시킨다. 그리고 runmode 매개변수를 TBCTL에 세트한다.

```
typedef enum
{
    RunMode_SoftStopAfterIncr  = ((uint16)0U << 14U), /** Stop after the next time-base counter increment */
    RunMode_SoftStopAfterDecr  = ((uint16)0U << 14U), /** Stop after the next time-base counter decrement */
    RunMode_SoftStopAfterCycle = ((uint16)1U << 14U), /** Stop when counter completes a whole cycle       */
    RunMode_FreeRun            = ((uint16)2U << 14U)  /** Free run */
} etpwmRunMode_t;
```

해당 매개변수 타입은 위와 같다.

Bit	Field	Value	Description
15-14	FREE, SOFT		Emulation Mode Bits. These bits select the behavior of the ePWM time-base counter during emulation events:
		0	Stop after the next time-base counter increment or decrement.
		1h	Stop when counter completes a whole cycle: • Up-count mode: stop when the time-base counter = period (TBCTR = TBPRD) • Down-count mode: stop when the time-base counter = 0x0000 (TBCTR = 0x0000) • Up-down-count mode: stop when the time-base counter = 0x0000 (TBCTR = 0x0000)
		2h-3h	Free run

해당 매개변수는 전체 사이클이 끝난 후에 정지하는 모드, free run, 한주기 후 정지 모드 등으로 만들 수 있다.

```
void etpwmSetCmpA(etpwmBASE_t *etpwm, uint16 value);

void etpwmSetCmpA(etpwmBASE_t *etpwm, uint16 value)
{
    etpwm->CMPA = value;
}
```

리턴값은 void
매개변수는 모듈번호와 원하는 카운터의 값을 입력한다.

CMPA 의 값을 value로 변경한다.

```
void etpwmSetCmpB(etpwmBASE_t *etpwm, uint16 value);

void etpwmSetCmpB(etpwmBASE_t *etpwm, uint16 value)
{
    etpwm->CMPB = value;
}
```

리턴값은 void
매개변수는 모듈번호와 원하는 카운터의 값을 입력한다.

CMPB의 값을 value로 변경한다.

```
void etpwmEnableCmpAShadowMode(etpwmBASE_t *etpwm, etpwmLoadMode_t loadmode);

void etpwmEnableCmpAShadowMode(etpwmBASE_t *etpwm, etpwmLoadMode_t loadmode)
{
    etpwm->CMPCTL &= (uint16)~(uint16)0x0013U;
    etpwm->CMPCTL |= loadmode;
}
```

매개변수로 pwm 모듈번호 및 loadmode 라는 매개 변수를 넣는다. loadmode는 아래와 같다.

```
typedef enum
{
    LoadMode_CtrEqZero       = 0U, /** Load on CTR = Zero             */
    LoadMode_CtrEqPeriod     = 1U, /** Load on CTR = PRD              */
    LoadMode_CtrEqZeroPeriod = 2U, /** Load on CTR = Zero or CTR = PRD */
    LoadMode_Freeze          = 3U  /** Freeze (no loads possible)     */
} etpwmLoadMode_t;
```

CMPTL 의 0, 1, 4번 비트를 0으로 클리어 한 후에 loadmode를 넣는다.

4	SHDWAMODE		Counter-compare A (CMPA) Register Operating Mode.
		0	Shadow mode. Operates as a double buffer. All writes via the CPU access the shadow register.
		1	Immediate mode. Only the active compare register is used. All writes and reads directly access the active register for immediate compare action.
3-2	LOADBMODE		Active Counter-Compare B (CMPB) Load From Shadow Select Mode. This bit has no effect in immediate mode (CMPCTL[SHDWBMODE] = 1).
		0	Load on CTR = Zero: Time-base counter equal to zero (TBCTR = 0x0000)
		1h	Load on CTR = PRD: Time-base counter equal to period (TBCTR = TBPRD)
		2h	Load on either CTR = Zero or CTR = PRD
		3h	Freeze (no loads possible)
1-0	LOADAMODE		Active Counter-Compare A (CMPA) Load From Shadow Select Mode. This bit has no effect in immediate mode (CMPCTL[SHDWAMODE] = 1).
		0	Load on CTR = Zero: Time-base counter equal to zero (TBCTR = 0x0000)
		1h	Load on CTR = PRD: Time-base counter equal to period (TBCTR = TBPRD)
		2h	Load on either CTR = Zero or CTR = PRD
		3h	Freeze (no loads possible)

CMPCTL[SHDWAMODE] 을 0 으로 하여 shadow mode로 동작하게 한다.
CMPCTL[LOADAMODE] 도 0 으로 만들어서 초기화 한다.

그리고 CMPCTL[LOADAMODE]에 loadmode 매개변수 (0, 1, 2, 3)을 넣어 해당 동작이 이루어지게 한다.

해당 동작은 Zero, 주기 인 순간 shadow 레지스터에서 CMPA를 읽는다는 것이다.
(보호나 이상 현상이 발생하지 않도록 하는 목적 같이 보인다.)

```
void etpwmDisableCmpAShadowMode(etpwmBASE_t *etpwm);

void etpwmDisableCmpAShadowMode(etpwmBASE_t *etpwm)
{
    etpwm->CMPCTL |= 0x0010U;
}
```

위 그림에 있는 CMPCTL[SHDWAMODE] 을 1 으로 하여 immediate mode로 동작하게 한다.

```
void etpwmSetActionQualPwmA(etpwmBASE_t *etpwm, etpwmActionQualConfig_t
actionqualconfig);

void etpwmSetActionQualPwmA(etpwmBASE_t *etpwm, etpwmActionQualConfig_t
actionqualconfig)
{
  etpwm->AQCTLA =((uint16)((uint16)actionqualconfig.CtrEqZero_Action     << 0U)  |
                  (uint16)((uint16)actionqualconfig.CtrEqPeriod_Action   << 2U) |
                  (uint16)((uint16)actionqualconfig.CtrEqCmpAUp_Action   << 4U) |
                  (uint16)((uint16)actionqualconfig.CtrEqCmpADown_Action << 6U)|
                  (uint16)((uint16)actionqualconfig.CtrEqCmpBUp_Action   << 8U) |
                  (uint16)((uint16)actionqualconfig.CtrEqCmpBDown_Action<<10U));
}
```

리턴은 void,

매개변수는 etpwm 모듈, actionqualconfig 변수를 넣는다.

해당 변수는 구조체 형태로 다음과 같다.

```
typedef struct
{
    etpwmActionQual_t CtrEqZero_Action;
    etpwmActionQual_t CtrEqPeriod_Action;
    etpwmActionQual_t CtrEqCmpAUp_Action;
    etpwmActionQual_t CtrEqCmpADown_Action;
    etpwmActionQual_t CtrEqCmpBUp_Action;
    etpwmActionQual_t CtrEqCmpBDown_Action;
}etpwmActionQualConfig_t;
```

구조체 내부의 각각의 변수는 다음과 같다.

```
typedef enum
{
    ActionQual_Disabled = 0U, /** Do nothing (action disabled)          */
    ActionQual_Clear    = 1U, /** Clear: force EPTWMxA/ETPWMB output low */
    ActionQual_Set      = 2U, /** Set: force ETPWMxA/ETPWMxB output high */
    ActionQual_Toggle   = 3U, /** Toggle EPWMxA/ETPWMxB output          */
    ForceSize_ActionQual = 0xFFFFU  /** Do not use (Makes sure that
etpwmActionQual_t is at least 16 bits wide)  */
} etpwmActionQual_t;
```

Table 35-31. Action-Qualifier Output A Control Register (AQCTLA) Field Descriptions

Bits	Name	Value	Description
15-12	Reserved	0	Reserved
11-10	CBD		Action when the time-base counter equals the active CMPB register and the counter is decrementing.
		0	Do nothing (action is disabled).
		1h	Clear: force EPWMxA output low.
		2h	Set: force EPWMxA output high.
		3h	Toggle EPWMxA output: low output signal will be forced high, and a high signal will be forced low.
9-8	CBU		Action when the counter equals the active CMPB register and the counter is incrementing.
		0	Do nothing (action is disabled).
		1h	Clear: force EPWMxA output low.
		2h	Set: force EPWMxA output high.
		3h	Toggle EPWMxA output: low output signal will be forced high, and a high signal will be forced low.
7-6	CAD		Action when the counter equals the active CMPA register and the counter is decrementing.
		0	Do nothing (action is disabled).
		1h	Clear: force EPWMxA output low.
		2h	Set: force EPWMxA output high.
		3h	Toggle EPWMxA output: low output signal will be forced high, and a high signal will be forced low.
5-4	CAU		Action when the counter equals the active CMPA register and the counter is incrementing.
		0	Do nothing (action is disabled).
		1h	Clear: force EPWMxA output low.
		2h	Set: force EPWMxA output high.
		3h	Toggle EPWMxA output: low output signal will be forced high, and a high signal will be forced low.
3-2	PRD		Action when the counter equals the period. Note: By definition, in count up-down mode when the counter equals period the direction is defined as 0 or counting down.
		0	Do nothing (action is disabled).
		1h	Clear: force EPWMxA output low.
		2h	Set: force EPWMxA output high.
		3h	Toggle EPWMxA output: low output signal will be forced high, and a high signal will be forced low.
1-0	ZRO		Action when counter equals zero. Note: By definition, in count up-down mode when the counter equals 0 the direction is defined as 1 or counting up.
		0	Do nothing (action is disabled).
		1h	Clear: force EPWMxA output low.
		2h	Set: force EPWMxA output high.
		3h	Toggle EPWMxA output: low output signal will be forced high, and a high signal will be forced low.

해당 구조체의 변수를 건드려 (주기, 제로, CMPA, CMPB) 일 때, HIGH, LOW, TOGGLE, DO NOTTHING 동작을 하도록 한다.

```
void etpwmEnableDeadBand(etpwmBASE_t *etpwm, etpwmDeadBandConfig_t
deadbandconfig);

void etpwmEnableDeadBand(etpwmBASE_t *etpwm, etpwmDeadBandConfig_t deadbandconfig)
{
    uint16 halfCycleMask = (uint16)((deadbandconfig.halfCycleEnable) ? 0x8000U :
0U);
    etpwm->DBCTL = ((uint16)deadbandconfig.inputmode |
                    (uint16)deadbandconfig.outputmode |
                    (uint16)deadbandconfig.polarity   |
                    halfCycleMask);
}
```

리턴형은 void 이고,

매개변수는 pwm 모듈, deadbandconfig를 넣는다.

해당 매개변수는 다음과 같다.

```
typedef struct
{
    etpwmDeadBandInputMode_t inputmode;
    etpwmDeadBandOutputMode_t outputmode;
    etpwmDeadBandPolarity_t polarity;
    boolean halfCycleEnable;
}etpwmDeadBandConfig_t;

typedef enum
{
    PWMA_RED_FED      = 0x00U,   /** Source of Rising edge delay: ETPWMxA, Source
of Falling edge delay: ETPWMxA */
    PWMA_FED_PWMB_RED = 0x10U,   /** Source of Rising edge delay: ETPWMxB, Source
of Falling edge delay: ETPWMxA */
    PWMA_RED_PWMB_FED = 0x20U,   /** Source of Rising edge delay: ETPWMxA, Source
of Falling edge delay: ETPWMxB */
    PWMB_RED_FED      = 0x30U,   /** Source of Rising edge delay: ETPWMxB, Source
of Falling edge delay: ETPWMxB */

    ForceSize_DBInput = 0xFFFFU  /** Do not use (Makes sure that
etpwmDeadBandInputMode_t is at least 16 bits wide)  */
} etpwmDeadBandInputMode_t;

typedef enum
{
    PWMA_PWMB_NIL      = 0U,     /** Deadband generation is bypassed for both
output signals                                    */
    PWMA_NIL_PWMB_FED  = 1U,     /** Disable rising-edge delay. The falling-edge
delayed signal is seen on output EPWMxB.          */
    PWMA_RED_PWMB_NIL  = 2U,     /** Disable falling-edge delay. The rising-edge
delayed signal is seen on output EPWMxA.          */
    PWMB_FED_PWMA_RED  = 3U,     /** Rising-edge delayed signal on output EPWMxA
and falling-edge delayed signal on output EPWMxB. */

       ForceSize_DBOutput = 0xFFFFU /** Do not use (Makes sure that
etpwmDeadBandOutputMode_t is at least 16 bits wide)  */
} etpwmDeadBandOutputMode_t;
```

```
typedef enum
{
    DisableInvert    = ((uint16)0U << 2U), /** Neither EPWMxA nor EPWMxB is
inverted */
    Invert_PWMA      = ((uint16)1U << 2U), /** EPWMxA is inverted
*/
    Invert_PWMB      = ((uint16)2U << 2U), /** EPWMxB is inverted
*/
    Invert_PWMA_PWMB = ((uint16)3U << 2U), /** Both EPWMxA and EPWMxB are inverted
*/

    ForceSize_DBPol = 0xFFFFU /** Do not use (Makes sure that
etpwmDeadBandPolarity_t is at least 16 bits wide)  */
} etpwmDeadBandPolarity_t;
```

해당 구조체의 주석을 보고 설정하여 넣으면 그것에 따라서 dead-band가 형성되어 출력된다.

```
void etpwmDisableDeadband(etpwmBASE_t *etpwm);

void etpwmDisableDeadband(etpwmBASE_t *etpwm)
{
    etpwm->DBCTL = 0U;
}
```

Table 35-35. Dead-Band Generator Control Register (DBCTL) Field Descriptions (continued)

Bits	Name	Value	Description
1-0	OUT_MODE		Dead-band Output Mode Control. Bit 1 controls the S1 switch and bit 0 controls the S0 switch shown in Figure 35-28. This allows you to selectively enable or bypass the dead-band generation for the falling-edge and rising-edge delay.
		0	Dead-band generation is bypassed for both output signals. In this mode, both the EPWMxA and EPWMxB output signals from the action-qualifier are passed directly to the PWM-chopper submodule. In this mode, the POLSEL and IN_MODE bits have no effect.
		1h	Disable rising-edge delay. The EPWMxA signal from the action-qualifier is passed straight through to the EPWMxA input of the PWM-chopper submodule. The falling-edge delayed signal is seen on output EPWMxB. The input signal for the delay is determined by DBCTL[IN_MODE].
		2h	The rising-edge delayed signal is seen on output EPWMxA. The input signal for the delay is determined by DBCTL[IN_MODE]. Disable falling-edge delay. The EPWMxB signal from the action-qualifier is passed straight through to the EPWMxB input of the PWM-chopper submodule.
		3h	Dead-band is fully enabled for both rising-edge delay on output EPWMxA and falling-edge delay on output EPWMxB. The input signal for the delay is determined by DBCTL[IN_MODE].

해당 동작은 DBCTL[OUT_MODE] = 0 이므로, DEAD-BAND 생성을 무시하고 지나친다.
즉, Disable한다.

```
void etpwmSetDeadBandDelay(etpwmBASE_t *etpwm, uint16 Rdelay, uint16 Fdelay);

void etpwmSetDeadBandDelay(etpwmBASE_t *etpwm, uint16 Rdelay, uint16 Fdelay)
{
    etpwm->DBRED = Rdelay;
    etpwm->DBFED = Fdelay;
}
```

Faliing Edge delay = DBFED 설정 0x 000 ~ 0x1FF

Rising Edge delay = DBRED 설정 0x 000 ~ 0x1FF 범위

```
void etpwmEnableChopping(etpwmBASE_t *etpwm, etpwmChoppingConfig_t
choppingconfig);


void etpwmEnableChopping(etpwmBASE_t *etpwm, etpwmChoppingConfig_t choppingconfig)
{
    etpwm->PCCTL = ((uint16)0x0001U              |
                    (uint16)choppingconfig.oswdth |
                    (uint16)choppingconfig.freq   |
                    (uint16)choppingconfig.duty);
}


typedef struct
{
    etpwmChoppingPulseWidth_t oswdth;
    etpwmChoppingClkFreq_t freq;
    etpwmChoppingDutyCycle_t duty;
}etpwmChoppingConfig_t;

typedef enum
{
    ChoppingPulseWidth_8_VCLK4   = ((uint16)0U << 1U),  /** 1 x VCLK4/8 wide  */
    ChoppingPulseWidth_16_VCLK4  = ((uint16)1U << 1U),  /** 2 x VCLK4/8 wide  */
    ChoppingPulseWidth_24_VCLK4  = ((uint16)2U << 1U),  /** 3 x VCLK4/8 wide  */
    ChoppingPulseWidth_32_VCLK4  = ((uint16)3U << 1U),  /** 4 x VCLK4/8 wide  */
    ChoppingPulseWidth_40_VCLK4  = ((uint16)4U << 1U),  /** 5 x VCLK4/8 wide  */
    ChoppingPulseWidth_48_VCLK4  = ((uint16)5U << 1U),  /** 6 x VCLK4/8 wide  */
    ChoppingPulseWidth_56_VCLK4  = ((uint16)6U << 1U),  /** 7 x VCLK4/8 wide  */
    ChoppingPulseWidth_64_VCLK4  = ((uint16)7U << 1U),  /** 8 x VCLK4/8 wide  */
    ChoppingPulseWidth_72_VCLK4  = ((uint16)8U << 1U),  /** 9 x VCLK4/8 wide  */
    ChoppingPulseWidth_80_VCLK4  = ((uint16)9U << 1U),  /** 10 x VCLK4/8 wide */
    ChoppingPulseWidth_88_VCLK4  = ((uint16)10U << 1U), /** 11 x VCLK4/8 wide */
    ChoppingPulseWidth_96_VCLK4  = ((uint16)11U << 1U), /** 12 x VCLK4/8 wide */
    ChoppingPulseWidth_104_VCLK4 = ((uint16)12U << 1U), /** 13 x VCLK4/8 wide */
    ChoppingPulseWidth_112_VCLK4 = ((uint16)13U << 1U), /** 14 x VCLK4/8 wide */
    ChoppingPulseWidth_120_VCLK4 = ((uint16)14U << 1U), /** 15 x VCLK4/8 wide */
    ChoppingPulseWidth_128_VCLK4 = ((uint16)15U << 1U), /** 16 x VCLK4/8 wide */

    ForceSize_ChopPulseWidth = 0xFFFFU /** Do not use (Makes sure that
etpwmChoppingPulseWidth_t is at least 16 bits wide)  */
} etpwmChoppingPulseWidth_t;
```

```
typedef enum
{
    ChoppingClkFreq_VCLK4_by_8  = ((uint16)0U << 5U), /** VCLK4/8 divided by 1 */
    ChoppingClkFreq_VCLK4_by_16 = ((uint16)1U << 5U), /** VCLK4/8 divided by 2 */
    ChoppingClkFreq_VCLK4_by_24 = ((uint16)2U << 5U), /** VCLK4/8 divided by 3 */
    ChoppingClkFreq_VCLK4_by_32 = ((uint16)3U << 5U), /** VCLK4/8 divided by 4 */
    ChoppingClkFreq_VCLK4_by_40 = ((uint16)4U << 5U), /** VCLK4/8 divided by 5 */
    ChoppingClkFreq_VCLK4_by_48 = ((uint16)5U << 5U), /** VCLK4/8 divided by 6 */
    ChoppingClkFreq_VCLK4_by_56 = ((uint16)6U << 5U), /** VCLK4/8 divided by 7 */
    ChoppingClkFreq_VCLK4_by_64 = ((uint16)7U << 5U), /** VCLK4/8 divided by 8 */

    ForceSize_ChopClkFreq = 0xFFFFU /** Do not use (Makes sure that
etpwmChoppingClkFreq_t is at least 16 bits wide)  */
}etpwmChoppingClkFreq_t;

typedef enum
{
    ChoppingDutyCycle_One_Eighth    = 0x0000U, /** Duty = 1/8 (12.5%) */
    ChoppingDutyCycle_Two_Eighths   = 0x0100U, /** Duty = 2/8 (25.0%) */
    ChoppingDutyCycle_Three_Eighths = 0x0200U, /** Duty = 3/8 (37.5%) */
    ChoppingDutyCycle_Four_Eighths  = 0x0300U, /** Duty = 4/8 (50.0%) */
    ChoppingDutyCycle_Five_Eighths  = 0x0400U, /** Duty = 5/8 (62.5%) */
    ChoppingDutyCycle_Six_Eighths   = 0x0500U, /** Duty = 6/8 (75.0%) */
    ChoppingDutyCycle_Seven_Eighths = 0x0600U, /** Duty = 7/8 (87.5%) */

    ForceSize_ChopDuty = 0xFFFFU /** Do not use (Makes sure that
etpwmChoppingDutyCycle_t is at least 16 bits wide)  */
} etpwmChoppingDutyCycle_t;
```

0	CHPEN		PWM-chopping Enable.
		0	Disable (bypass) PWM chopping function.
		1	Enable chopping function.

0번 비트를 세트 하므로, chopping function을 활성화 한다.

이외 매개변수 choppingconfig 변수를 위에 기술한 구조체를 보고 값을 집어넣어 설정한다. PCCTL 레지스터의 chopping clock bit, chopping clock frequency, one-shot pulse width를 설정한다.

즉 원하는 사양의 chop 된 pwm 신호를 생성한다.

```
void etpwmDisableChopping(etpwmBASE_t *etpwm);
```

해당 etPWM모듈의 chop 기능을 disable 한다.

```
void etpwmEnableTripZoneSources(etpwmBASE_t *etpwm, etpwmTripZoneSrc_t sources);

void etpwmEnableTripZoneSources(etpwmBASE_t *etpwm, etpwmTripZoneSrc_t sources)
{
    etpwm->TZSEL |= sources;
}
```

Table 35-39. Trip-Zone Submodule Select Register (TZSEL) Field Descriptions

Bits	Name	Value	Description
One-Shot (OSHT) Trip-zone enable/disable. When any of the enabled pins go low, a one-shot trip event occurs for this ePWM module. When the event occurs, the action defined in the TZCTL register is taken on the EPWMxA and EPWMxB outputs. The one-shot trip condition remains latched until the user clears the condition via the TZCLR register.			
15	DCBEVT1		Digital Compare Output B Event 1 Select.
		0	Disable DCBEVT1 as one-shot-trip source for this ePWM module.
		1	Enable DCBEVT1 as one-shot-trip source for this ePWM module.
14	DCAEVT1		Digital Compare Output A Event 1 Select.
		0	Disable DCAEVT1 as one-shot-trip source for this ePWM module.
		1	Enable DCAEVT1 as one-shot-trip source for this ePWM module.
13	OSHT6		Trip-zone 6 ($\overline{TZ6}$) Select.
		0	Disable $\overline{TZ6}$ as a one-shot trip source for this ePWM module.
		1	Enable $\overline{TZ6}$ as a one-shot trip source for this ePWM module.
12	OSHT5		Trip-zone 5 ($\overline{TZ5}$) Select.
		0	Disable $\overline{TZ5}$ as a one-shot trip source for this ePWM module.
		1	Enable $\overline{TZ5}$ as a one-shot trip source for this ePWM module.
Cycle-by-Cycle (CBC) Trip-zone enable/disable. When any of the enabled pins go low, a cycle-by-cycle trip event occurs for this ePWM module. When the event occurs, the action defined in the TZCTL register is taken on the EPWMxA and EPWMxB outputs. A cycle-by-cycle trip condition is automatically cleared when the time-base counter reaches zero.			
7	DCBEVT2		Digital Compare Output B Event 2 Select.
		0	Disable DCBEVT2 as a CBC trip source for this ePWM module.
		1	Enable DCBEVT2 as a CBC trip source for this ePWM module.
6	DCAEVT2		Digital Compare Output A Event 2 Select.
		0	Disable DCAEVT2 as a CBC trip source for this ePWM module.
		1	Enable DCAEVT2 as a CBC trip source for this ePWM module.

간략화 하여 표기한 레지스터 정보이다. OSHT는 쇼트방지, CBC는 허용 초과전류 방지를 위해서 사용한다. OSHT는 사용자가 TZCLR로 클리어 해주기 전까지 해당 이벤트가 유지되고, CBC는 한 주기마다 이벤트 초기화가 된다.(이벤트 액션은 TZCTL로 정한다.)

리턴값은 void,
매개변수는 모듈넘버와 sources 변수를 넣는다.

```
typedef enum
{
    CycleByCycle_TZ1     = ((uint16)1U << 0U),
    CycleByCycle_TZ2     = ((uint16)1U << 1U),
    CycleByCycle_TZ3     = ((uint16)1U << 2U),
    CycleByCycle_TZ4     = ((uint16)1U << 3U),
    CycleByCycle_TZ5     = ((uint16)1U << 4U),
    CycleByCycle_TZ6     = ((uint16)1U << 5U),
    CycleByCycle_DCAEVT2 = ((uint16)1U << 6U),
    CycleByCycle_DCBEVT2 = ((uint16)1U << 7U),
    OneShot_TZ1          = ((uint16)1U << 8U),
    OneShot_TZ2          = ((uint16)1U << 9U),
    OneShot_TZ3          = ((uint16)1U << 10U),
    OneShot_TZ4          = ((uint16)1U << 11U),
```

```
    OneShot_TZ5          = ((uint16)1U << 12U),
    OneShot_TZ6          = ((uint16)1U << 13U),
    OneShot_DCAEVT1      = ((uint16)1U << 14U),
    OneShot_DCBEVT1      = ((uint16)1U << 15U)
} etpwmTripZoneSrc_t;
```

해당 sources 변수는 다음과 같은 열거형 데이터를 삽입한다.

이를 통하여 digital compare output x Event나 해당하는 Trip-zone을 선택하게 된다.

```
void etpwmDisableTripZoneSources(etpwmBASE_t *etpwm, etpwmTripZoneSrc_t sources);


void etpwmSetTripAction(etpwmBASE_t *etpwm, etpwmTripActionConfig_t
tripactionconfig);
```