TI DSP, MCU 및 Xilinx Zynq FPGA 프로그래밍 전문가 과정

2018-05-11 (52회차)

강사 - Innova Lee(이상훈)

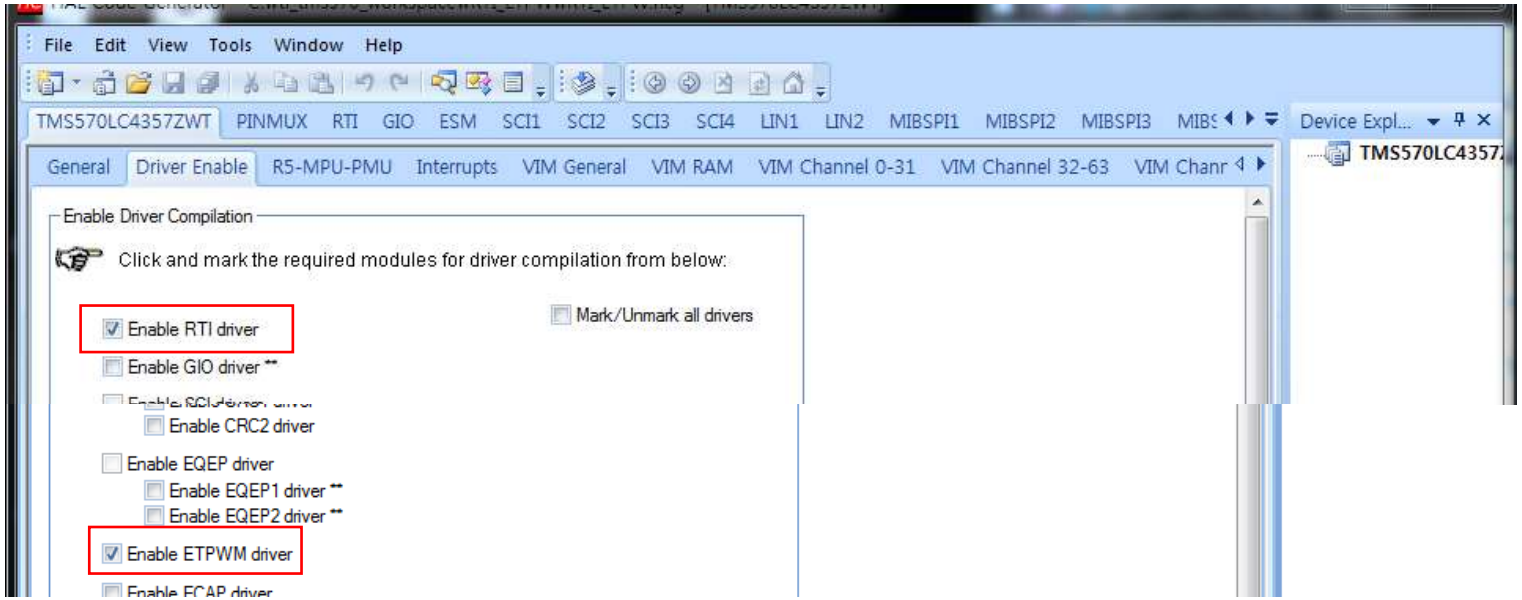
gcccompil3r@gmail.com

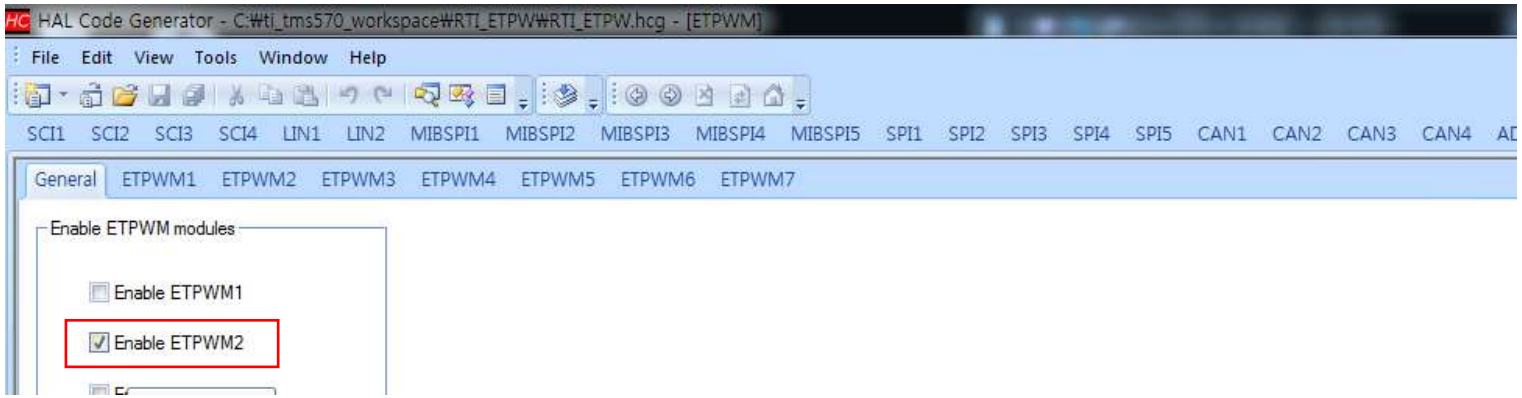
학생 - 정유경

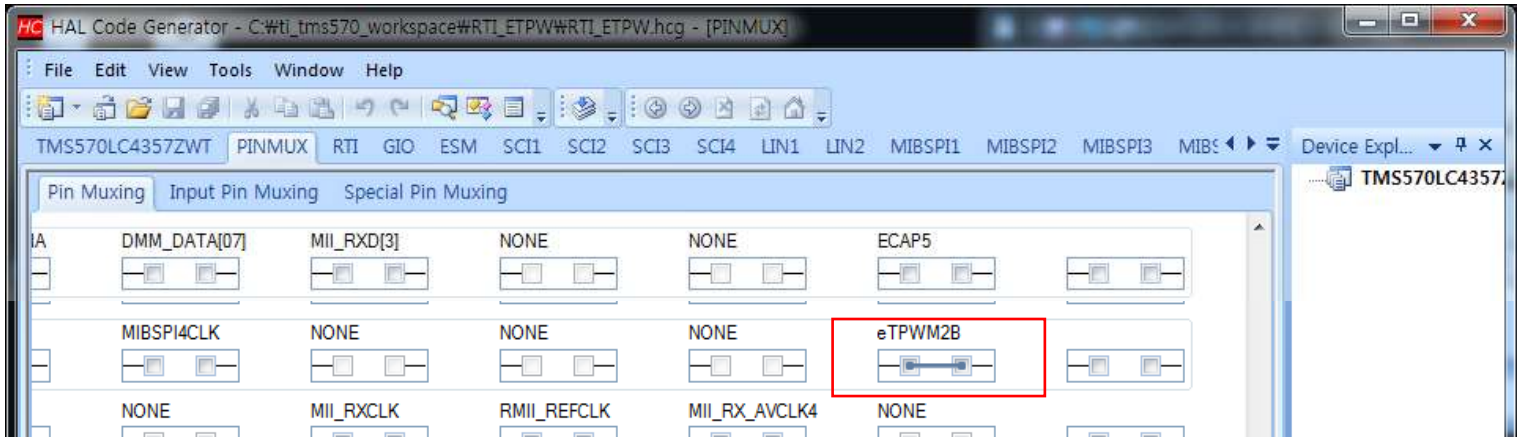
ucong@naver.com

NHET과 ETPWM을 모두 PWM으로 사용할 수 있기 때문에
PWM을 15개 사용할 수 있다

1. PLL 베이스로 PWM이용하여 LED 밝기 제어하기

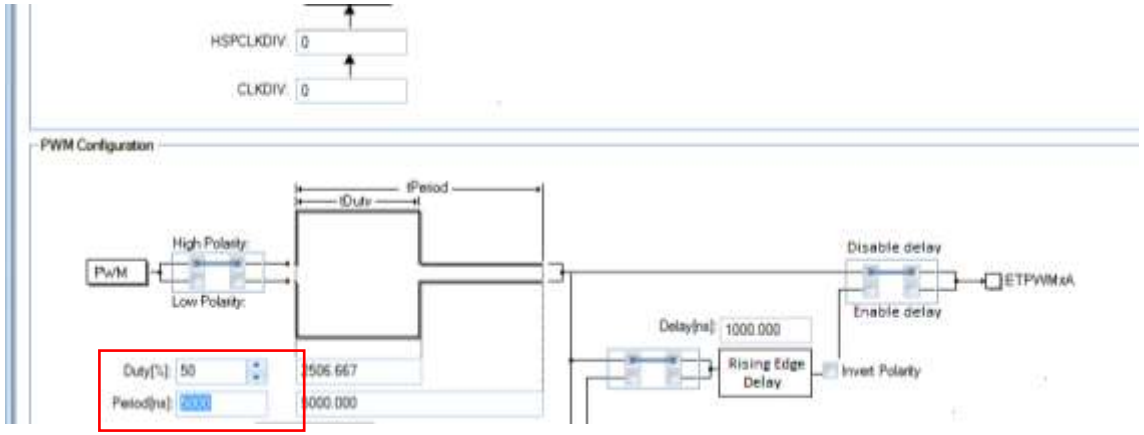






부트코드에서 PINMUX[186] 셋팅했던걸 여기서 이렇게 간편하게 해줄 수 있다.

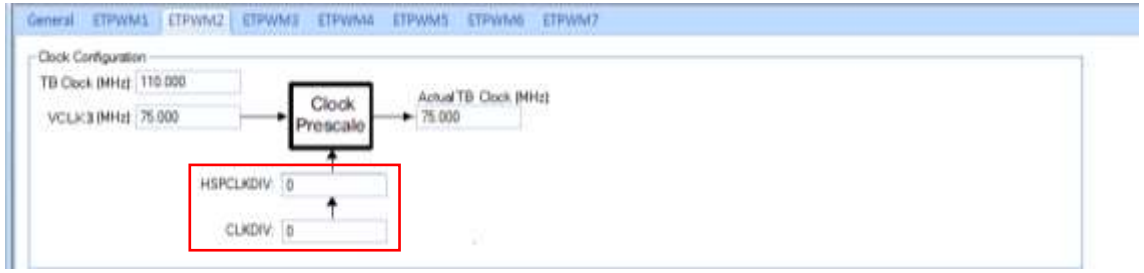
period 5000으로 변경하고 Generate Code



코드를 작성한다.

```
#include "HL_sys_common.h"
#include "HL_etpwm.h"

int main(void)
{

    int i;
    unsigned short val = 0;
    etpwmInit();

    for(;;)
    {
      etpwmStartTBCLK();
      for(i=0; i<100000; i++)
      ;
      etpwmSetCmpB(etpwmREG2, val);
      val++;
      etpwmStopTBCLK();

      if(val == 500)
        val = 0;
    }

    return 0;
}
```

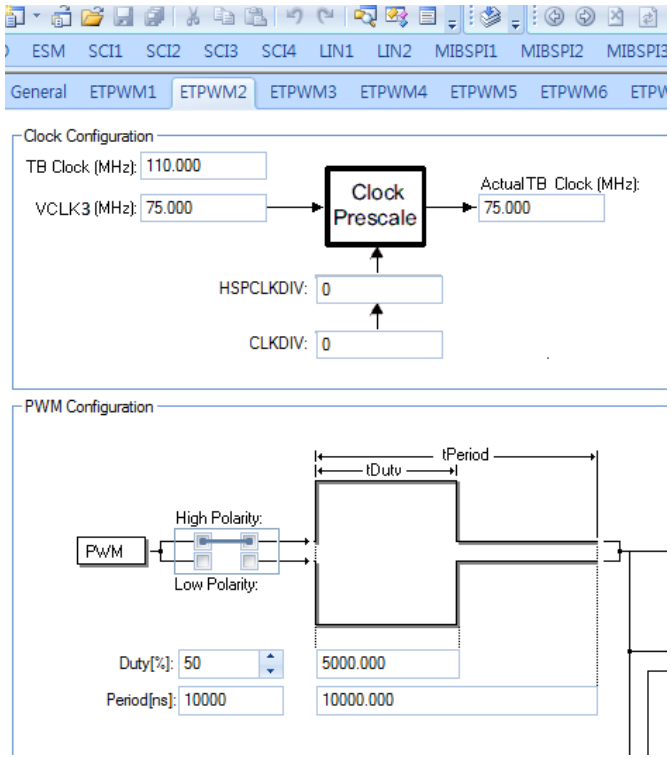
```
void etpwmInit(void)
{

    /** ETPWM2를 초기화한다. */

    /** - Sets high speed time-base clock prescale bits */
    etpwmREG2->TBCTL = (uint16)0U << 7U; // 분주없음
    /** - Sets time-base clock prescale bits */
    etpwmREG2->TBCTL |= (uint16)((uint16)0U << 10U); // 분주없음
```




```
    /** - Sets time period or frequency for ETPWM block both PWMA and PWMB*/
    etpwmREG2->TBPRD = 374U; // PWM의 주파수를 결정한다.
```

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.



Period를 10000으로 바꿔주고 다시 Generate Code하면

```
/** - Sets time period or frequency for ETPWM block both PWMA and PWM
etpwmREG2->TBPRD = 749U;

/**   Setup the duty cycle for PWMA */
```

TBPRD 값이 749로 바뀌는 것을 알 수 있다.

```
/** - Setup the duty cycle for PWMA */
    etpwmREG2->CMPA = 375U
/** - Setup the duty cycle for PWMB */
    etpwmREG2->CMPB = 375U;
```

타임베이스 카운터(TBCTR)가 카운터 비교기 A(CMPA)와 일치한다는 인터럽트를 발생시킨다.

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:

카운터가 샘플링타임마다 세면서 CMPA (375)랑 같아지면 인터럽트를 발생시킨다, CMPB에 대해서도 마찬가지

```
  /** - Force EPWMxA output high when counter reaches zero and low when counter reaches Compare A value */
    etpwmREG2->AQCTLA = ((uint16)((uint16)ActionQual_Set   << 0U)
                       | (uint16)((uint16)ActionQual_Clear << 4U));// 할코젠 PINMUX에서 EPTWMxB 지정했었다.
EPTWMxA안쓴다
    /** - Force EPWMxB output high when counter reaches zero and low when counter reaches Compare B value */
    etpwmREG2->AQCTLB = ((uint16)((uint16)ActionQual_Set   << 0U)  // 1
                      | (uint16)((uint16)ActionQual_Clear << 8U)); // 8
```

카운터가 0일 때, 카운터랑 CMPB 레지스터랑 일치할 때 EPWMxB의 출력을 0으로 떨어뜨린다.

```
/** - 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
    */
    etpwmREG2->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  */
```

Dead-band generation is bypassed for both output signals모든 출력신호에 대해 바이패스 된다

PWM chopper 즉, 파형정형안하고 바이패스 시킨다

Active high (AH) mode. Neither EPWMxA nor EPWMxB is inverted

EPWMxA In (from the action-qualifier) is the source for both falling-edge and rising-edge delay.

여기서 특별히 한일이 없다.

```
/** - Set the rising edge delay  */
    etpwmREG2->DBRED = 110U;

    /** - Set the falling edge delay  */
    etpwmREG2->DBFED = 110U; //2084p
```

dead band: 회로가 버틸수 있는 값 이상의 전압을 넣으면 보드가 고장이 난다. 따라서 데드밴드가 지나가는 시점을 계산해 주려고 한다

목적은 데드밴드를 계산해서 딜레이를 넣는것이다.

Saturation Instruction -127부터 +128까지 사각파가 만들어진다

지연시켜서 (딜레이를 110을 주어서) 윗부분 자르고 아랫부분도 지연 110 주어서 잘라낸다

*. DEAD BAND: 일반적으로, 불감대역은 측정된 값이 변화되는 다시 말하면 디바이스나 컴퓨터에 대해 레지스터가 변경되는 주어진 범위나 총합의 최소 퍼센트로써 정의 할 수 있다. 예를 들면, 0~100V의 범위를 갖는 디바이스를 가지고 있고, 1%의 불감대역으로 설정하면 들어오는 전압은 값이 변경되는 것을 알아차리기 위해 최소 1 Volt의 변화여야 한다.

```
/** - 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
    */
    etpwmREG2->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 */ // 8, 9
                     | (uint16)((uint16)0U << 5U));  /* Chopping Clock Frequency */
```

PWM-Chopper Control Register (PCCTL) 8 , 9 bit set이므로, Duty = 4/8 (50.0%)

Chopping Clock Frequency는 분주하지 않음 (no prescale, = 12.5 MHz at 100 MHz VCLK3)

One-Shot Pulse(한번 펄스 내보낼 때 VCLK3를 8개 단위로 쪼개서 보겠다) Width.

: VCLK3 / 8 wide (= 80 nS at 100 MHz VCLK3) 100MHz를 8분주했다는 의미이다.
Disable (bypass) PWM chopping function.

```
/** - Set trip source enable */
    etpwmREG2->TZSEL = 0x00;
```

Trip-Zone Submodule Select Register (TZSEL) 를 0으로 셋팅하여 인터럽트를 다 끈다

(1997p 이런부분도 다 읽어주어야 MCU 할 수 있다, 언더바가 있는 애들은 논리회로가 들어있다고 추측할 수 있다.)

```
/** - Set interrupt enable */
    etpwmREG2->TZEINT = 0x00;
```

Trip-Zone Enable Interrupt Register (TZEINT)를 0으로 셋팅하여 인터럽트를 다 막았다

```
/** - Sets up the event for interrupt */
    etpwmREG2->ETSEL = (uint16)NO_EVENT;

    if ((etpwmREG2->ETSEL & 0x0007U) != 0U) // ETSEL은 NO_EVENT(0)이므로 if 안들어감
```

Event-Trigger Selection Register (ETSEL)
0 Disable EPWMxSOCB
0 Enable DCBEVT1.soc event
1h Enable event time-base counter equal to zero. (TBCTR = 0x0000).

```
/** - Setup the frequency of the interrupt generation */
    etpwmREG2->ETPS = 1U;
```

Event-Trigger Prescale Register (ETPS) 1들어간다 여기서 인터럽트 생성을 시작

1들어가면 인터럽트가 자동으로 활성화

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. 신호가 충돌 날 상황을 방지하겠다는 의미이다

```
/** - Sets up the ADC SOC interrupt */ 사용하지 않는다.
```

```
etpwmStartTBCLK(); // This function starts the time-base clocks of all eTPWMx modules.
```

```
    /* Enable Pin Muxing */
    pinMuxReg->KICKER0 = 0x83E70B13U;
    pinMuxReg->KICKER1 = 0x95A4F1E0U;
```

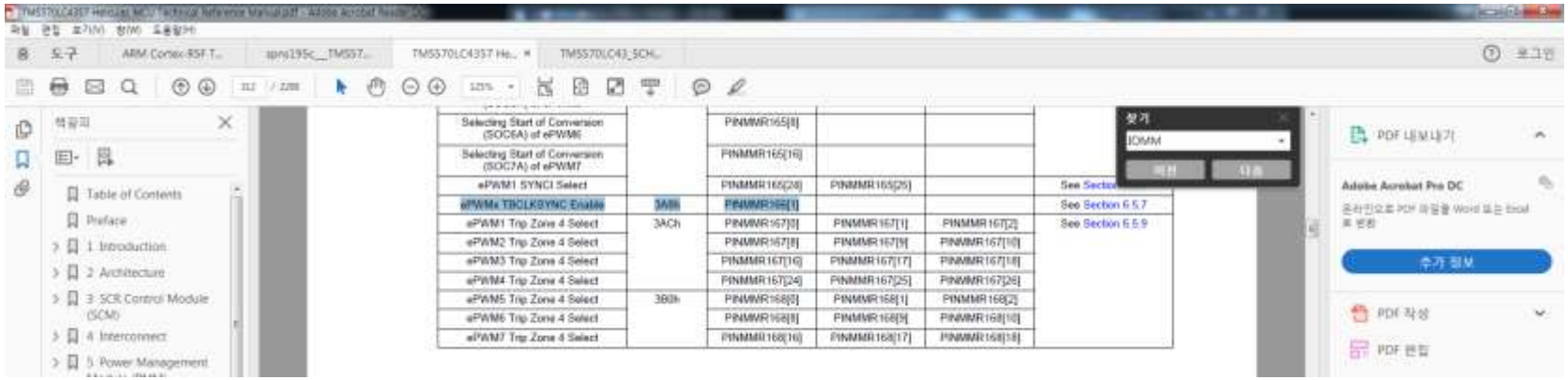
MUX를 사용하기 위해 KICKER값을 설정하는게 필요했었다. 여기서도 마찬가지

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

```
uint32 PINMUX[180]
/**< 0x110 - 1A4 : Output Pin Multiplexing Control Registers (38 registers); 0x250 - 0x29C : Input Pin Multiplexing
Control Registers (20); 0X390 - 3DC : Special Functionality Control Registers (20) */
```

이었다 따라서 180이 20번째, 160은 0번째, 166은 6번째 레지스터가 된다.

PINMMRnn: Special Functionality Multiplexing Control Registers (MUX 테이블의 스페셜 레지스터)를 살펴보면 (312p)



이걸 활성화 시켜야 PWM신호가 나온다

이상신호가 감지되면 인터럽트, 트리거, 유아트등을 사용해서 PINMMR166 꺼야 한다

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

```
    pinMuxReg->KICKER1 = 0x00000000U;
```

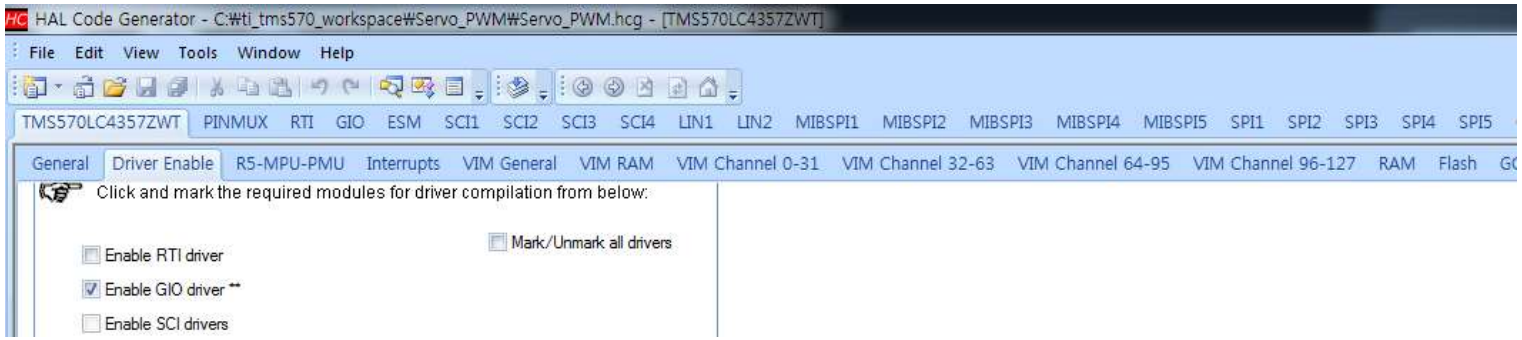
여기까지 PWM 싱크신호를 주었다.

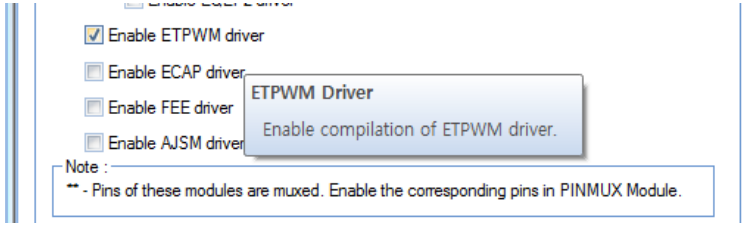
SetCompB랑 비교하면서 375U넣었으니까
375될때까지 value값을 증가시킨다 그러다가 500되면 떨어진다. 이를 코드에서 확인해보면 다음과 같다.

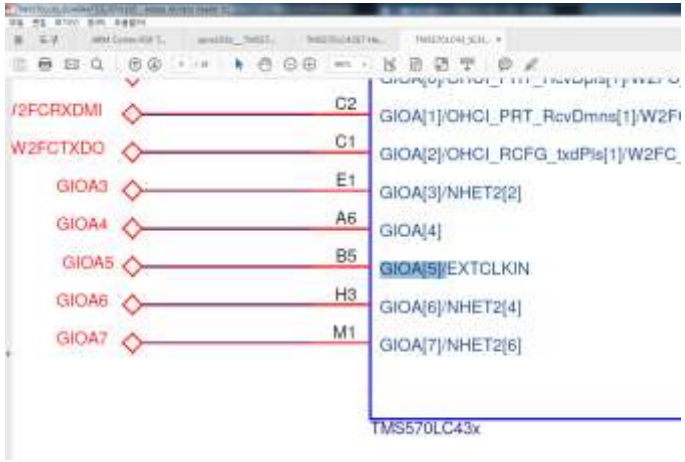
```
 for(;;)
    {
        etpwmStartTBCLK();
        for(i=0;i<100000;i++);

           etpwmSetCmpB(etpwmREG2, val);
           val++;
           etpwmStopTBCLK();

              if(val==500)
                  val=0;
    }
```

For문 전체가 하나의 주기가 된다. val값의 증가에 따라 떠있는 시간이 길어지면서 LED는 점점 밝아진다

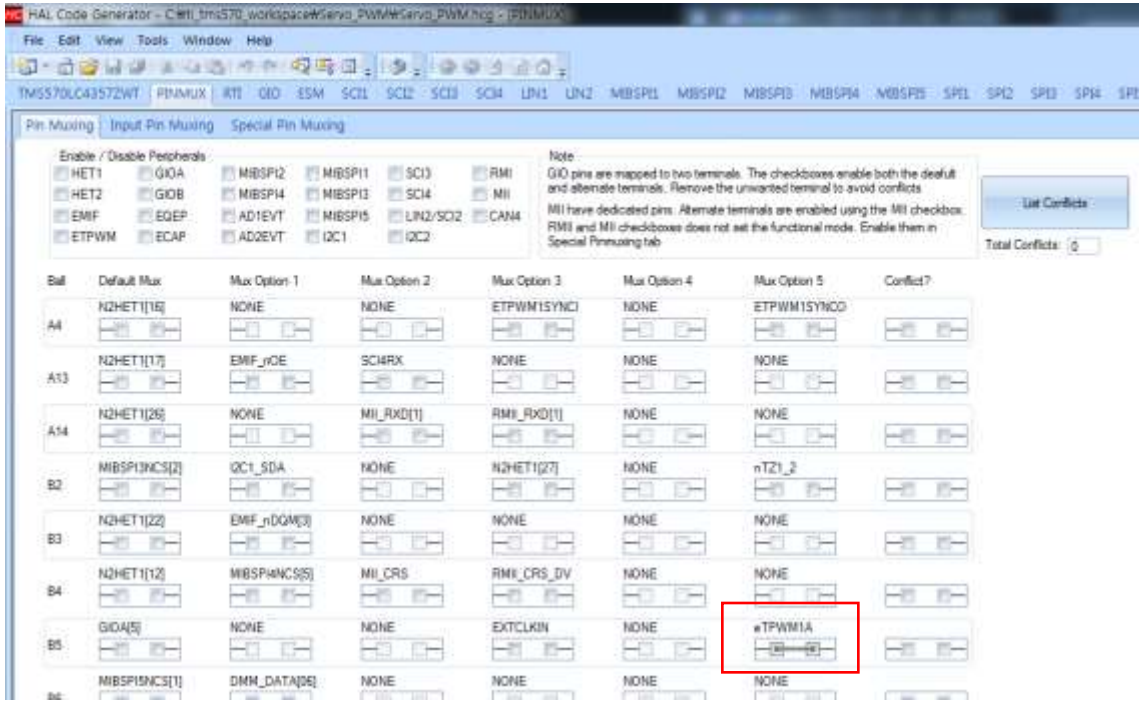
2. PLL 베이스로 서보모터 제어하기(20ms, 50Hz)



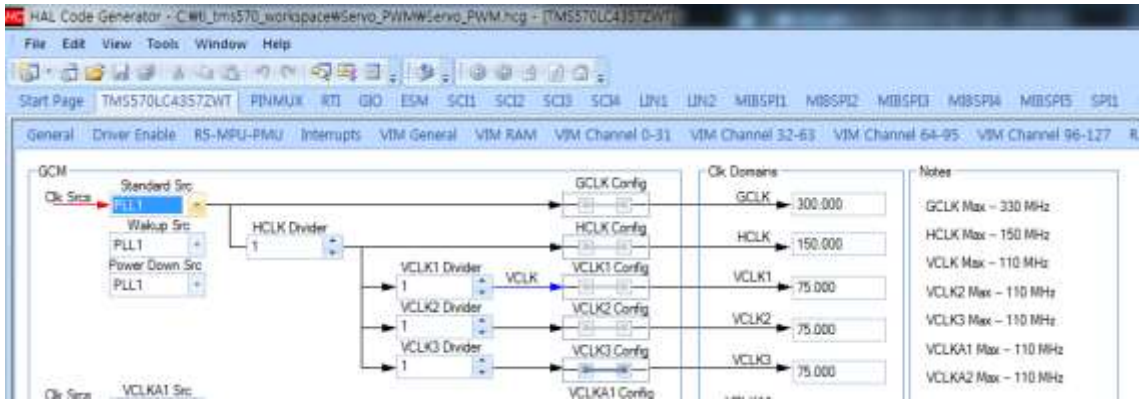




GPIO[5]를 PWM으로 쓴다.



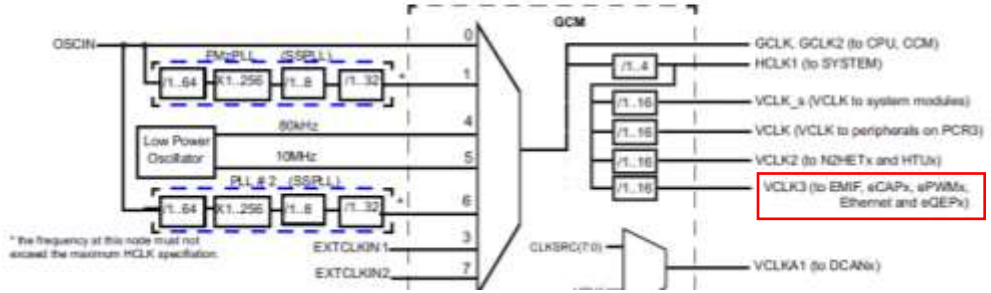
모터를 20ms로 주기를 맞추는게 중요 50Hz 전체동작주파수를 50Hz 맞취주기 위해서는 PLL을 제어해야 한다.
PLL을 제어하기 위해서는 Global Clock Module (GCM)을 건드려야 한다



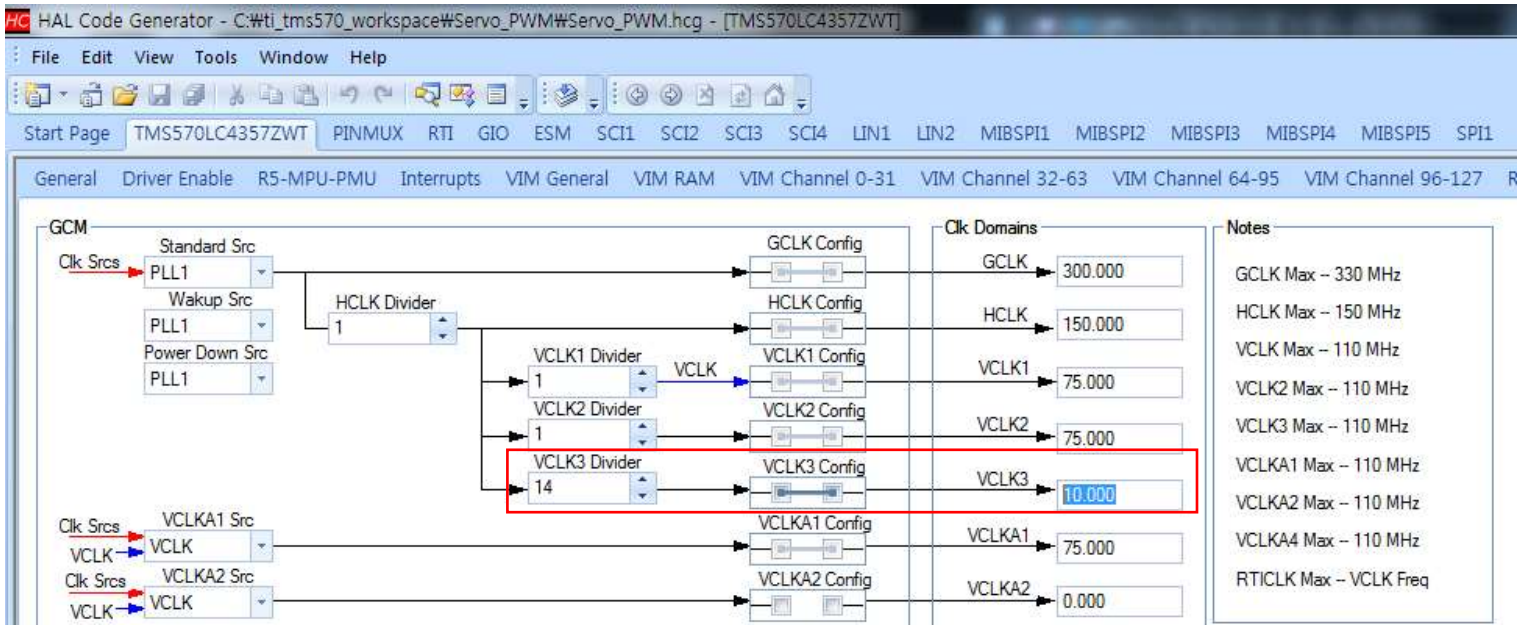
VCLK가 75MHz로 출력된다. Divider를 조정하여 주파수를 변화시킬 수 있다.

6.6.2.2 Mapping of Clock Domains to Device Modules

Each clock domain has a dedicated functionality as shown in Figure 6-6.

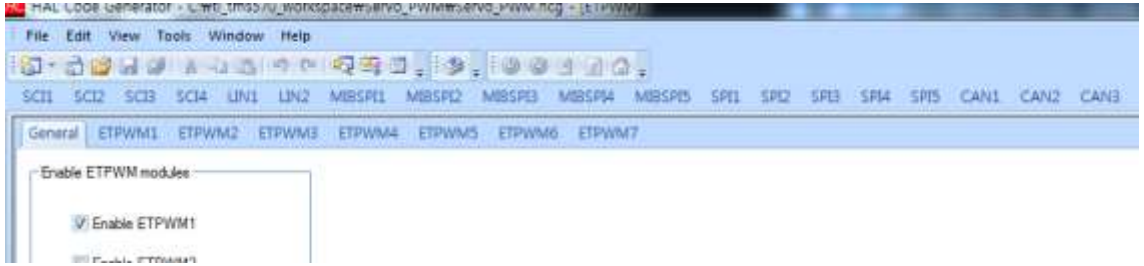


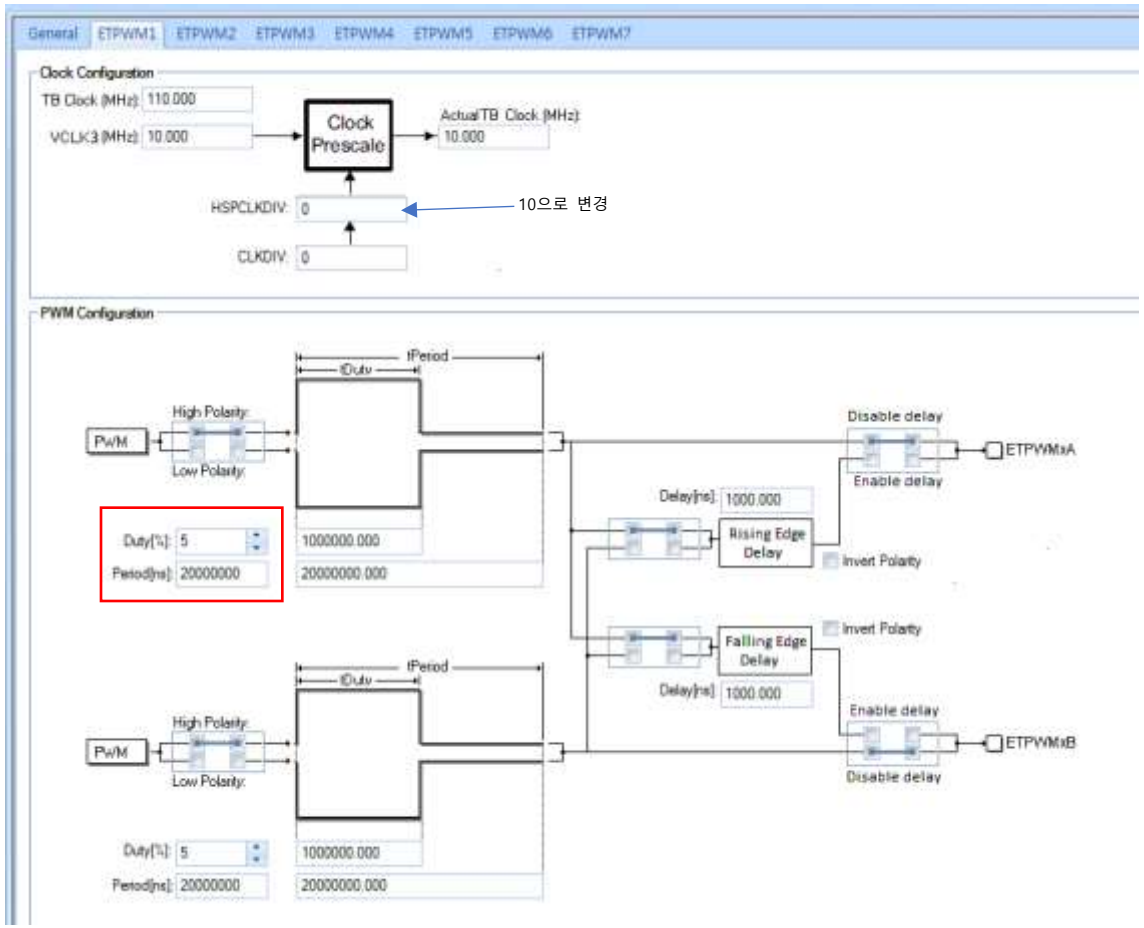
VCLK1 은 peripheral, VCLK2 는 N2HETx and HTUx , VCLK3 는 EMIF, ePWMx에 사용된다. 따라서 PWM은 VCLK 3이니까 3을 바꿔줘야 한다



VCLK3를 10Mhz로 맞춰주기 위해 VCLK3 Divider를 14로 고친다

모터제어에서 가장 중요한 것은 주기를 20ms 50Hz 주는것이다. 따라서 다음과 같이 설정한다.





20ms period를 설정하고 Duty를 5로 하면 1ms동안 1이 설정된다

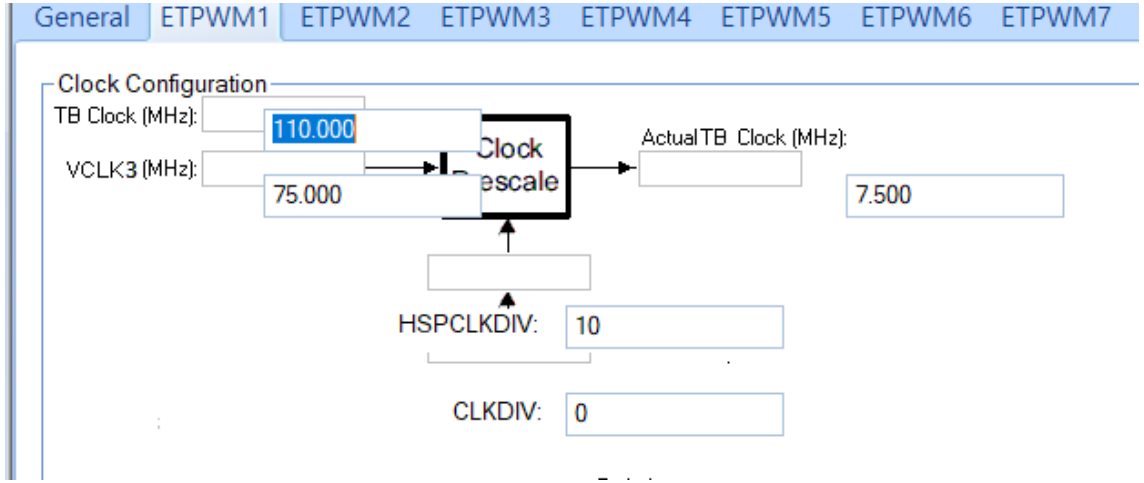
A만 사용하지만, 간섭을 막기 위해 B도 똑같이 설정해준다

코드를 작성한다.

```
#include "HL_system.h"
#include "HL_etpwm.h"
#include "HL_sys_common.h"

int idx = 0;
uint32 value = 0;
uint32 duty_arr[20] = {1000, 1100, 1200, 1300, 1400,1500, 1600, 1700, 1800, 1900,2000, 1900, 1800, 1700,
1600,1500, 1400, 1300, 1200, 1100};
void pwmSet(void);
void delay(uint32);

int main(void)
{
    etpwmInit();
    etpwmStartTBCLK();
    delay(10000);
    for(;;)
    {
    pwmSet();
    delay(5000000);
    }
    return 0;
}

void delay(uint32 delay)
{
    int i;
    for(i = 0; i < delay; i++)
       ;
}

void pwmSet(void)
{
    value = duty_arr[idx % 20];
    etpwmSetCmpA(etpwmREG1, value);
    idx++;
}
```

코드분석

```
/** @b initialize @b ETPWM1 */

    /** - Sets high speed time-base clock prescale bits */
    etpwmREG1->TBCTL = (uint16)5U << 7U; // 7, 9 : HSPCLKDIV 10분주

    /** - Sets time-base clock prescale bits */
    etpwmREG1->TBCTL |= (uint16)((uint16)0U << 10U); // 10 : CLKDIV 분주없음
```




```
/** - Sets time period or frequency for ETPWM block both PWMA and PWMB*/
    etpwmREG1->TBPRD = 1499999U;

    /** - Setup the duty cycle for PWMA */
    etpwmREG1->CMPA = 75000U;

    /** - Setup the duty cycle for PWMB */
    etpwmREG1->CMPB = 750000U;
```

나머지 부분은 LED 예제와 동일, 메인함수 부분은 분석이 필요