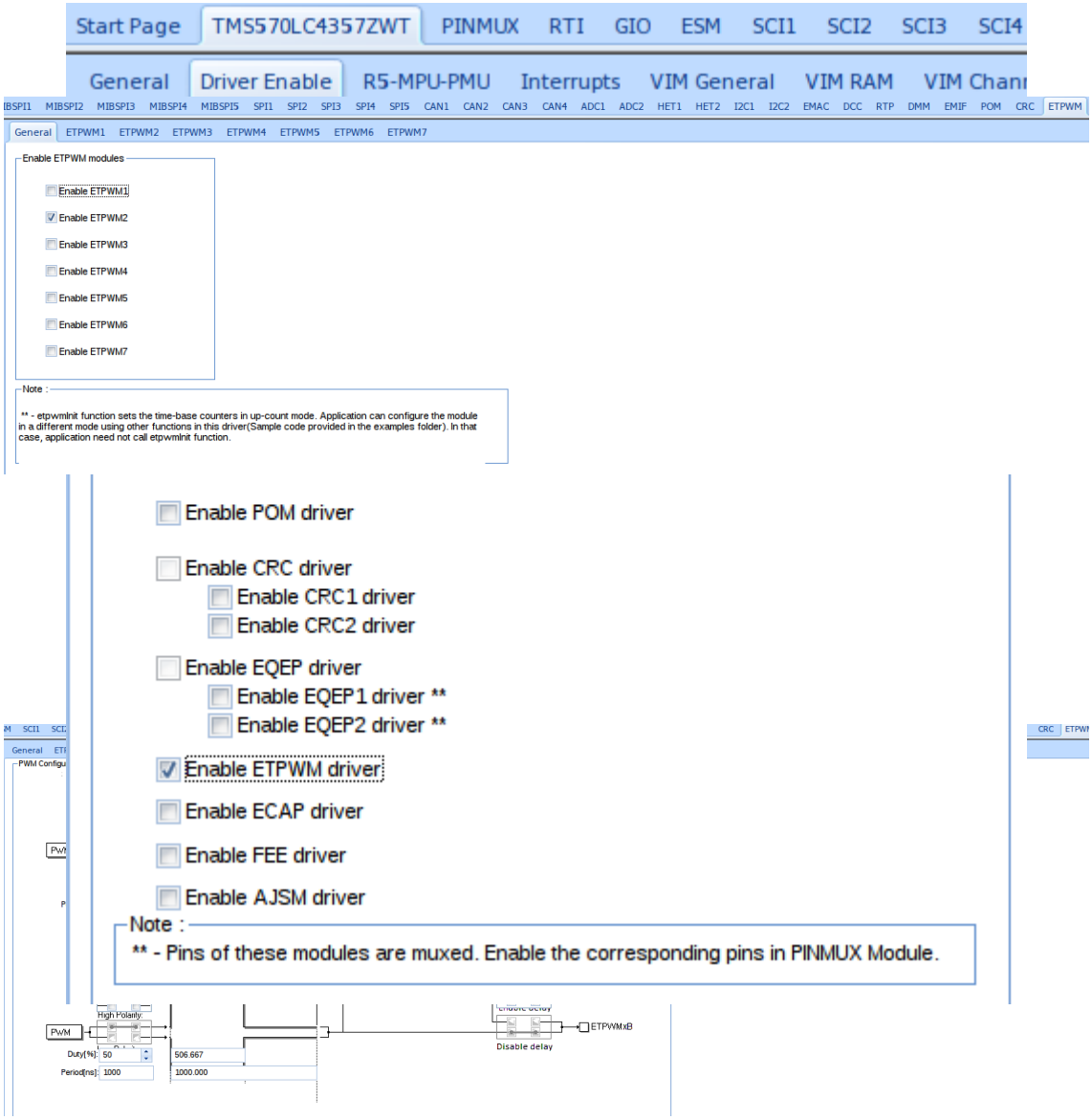
TI DSP, MCU 및 Xilinx Zynq FPGA
프로그래밍 전문가 과정

강사 – Innova Lee(이상훈)
gcccompil3r@gmail.com
학생 – 하성용
accept0108@naver.com

52 일차

RTI_etPWM

CCS, HALCoGen 생성



TBPRD // 주파수 결정

etPWM2 //
PWM 의 주기를 결정

749 = 10000ms
374 = 5000ns

CMPA // 연속적으로 비교를 한다 TBCTR 이랑
값이 같아지면 인터럽트 발생시킨다
563a → CMPA 검색

35.4.2.2 Counter-Compare A Register (CMPA)

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

15	0
CMPA	
R/W-0	

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.

```c
#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<10000; i++)
            ;

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

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

*HL_sys_main.c

```
54
55 /** @fn void main(void)
56 *   @brief Application main function
57 *   @note This function is empty by default.
58 *
59 *   This function is called after startup.
60 *   The user can use this function to implement the application.
61 */
62
63 /* USER CODE BEGIN (2) */
64 /* USER CODE END */
65
66 int main(void)
67 {
68     int i;
69     unsigned short val = 0;
70
71     etpwmInit();
72
73     for(;;)
74     {
75         etpwmStartTBCLK();
76
77         for(i=0; i<10000; i++)
78             ;
79
80         etpwmSetCmpP(etpwmREG2, val);
81         val++;
82         etpwmStopTBCLK();
83
84         if(val == 500)
85             val = 0;
86     }
87
88     return 0;
89 }
```

AQCTLA//

HL_etpwm.c

```
76     /** @b initialize @b ETPWM2 */
77
78     /** - Sets high speed time-base clock prescale bits */
79     etpwmREG2->TBCTL = (uint16)0U << 7U;
80
81     /** - Sets time-base clock prescale bits */
82     etpwmREG2->TBCTL |= (uint16)((uint16)0U << 10U);
83
84     /** - Sets time period or frequency for ETPWM block both PWMA a
85     etpwmREG2->TBPRD = 749U;
86
87     /** - Setup the duty cycle for PWMA */
88     etpwmREG2->CMPA = 375U;
89
90     /** - Setup the duty cycle for PWMB */
91     etpwmREG2->CMPB = 375U;
92
93     /** - Force EPWMxA output high when counter reaches zero and lo
94     etpwmREG2->AQCTLA = ((uint16)((uint16)ActionQual_Set   << 0U)
95                        | (uint16)((uint16)ActionQual_Clear << 4U));
96
```

AQCTLA

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 35-31. 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.

DBCTL//

```
106 etpwmREG2->DBCTL = ((uint16)((uint16)0U << 5U)    /* Source for Falling edge delay(0-PWMA, 1-PWMB) */
107                   | (uint16)((uint16)0U << 4U) /* Source for Rising edge delay(0-PWMA, 1-PWMB)  */
108                   | (uint16)((uint16)0U << 3U)  /* Enable/Disable EPWMxB invert       */
109                   | (uint16)((uint16)0U << 2U)  /* Enable/Disable EPWMxA invert       */
110                   | (uint16)((uint16)0U << 1U) /* Enable/Disable Rising Edge Delay   */
111                   | (uint16)((uint16)0U << 0U));   /* Enable/Disable Falling Edge Delay  */
112
```

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].

DBRED//

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

자기한계를 넘길경우, 데드밴드가 지나가는 그 시점을 계산해주는 레지스터
이 시간만큼 딜레이 넣어준다는거

ex) 회로를 만들었는제 회로가 버틸수있는 전압의 한계가 5 볼트인데
7 볼트가 들어가면 보드 고장

사인웨이브 값의 팍하고 올라가는 볼록한부분을 잘라줌
밑으로 볼록한부분도 같음

Table 35-36. Dead-Band Generator Falling Edge Delay Register (DBFED) Field Descriptions

Bits	Name	Description
15-10	Reserved	Reserved
9-0	DEL	Falling Edge Delay Count. 10-bit counter.

35.4.4.3 Dead-Band Generator Rising Edge Delay Register (DBRED)

Figure 35-77. Dead-Band Generator Rising Edge Delay Register (DBRED) [offset = 22h]

15	10	9	8
Reserved		DEL	
R-0		R/W-0	

7	0
DEL	
R/W-0	

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

Table 35-37. Dead-Band Generator Rising Edge Delay Register (DBRED) Field Descriptions

Bits	Name	Description
15-10	Reserved	Reserved
9-0	DEL	Rising Edge Delay Count. 10-bit counter.

PCCTL//

```
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 */
                  | (uint16)((uint16)0U << 5U));  /* Chopping Clock Frequency */
```

8 번 bit 와 9 번 bit 만 살리고 다 꺼버림
8 번 bit 와 9 번 bit 만 살아나면 duty 가 50%

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)

duty 는 그 사이클이 몇번 반복되면 꺼지겠다 는걸 의미

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

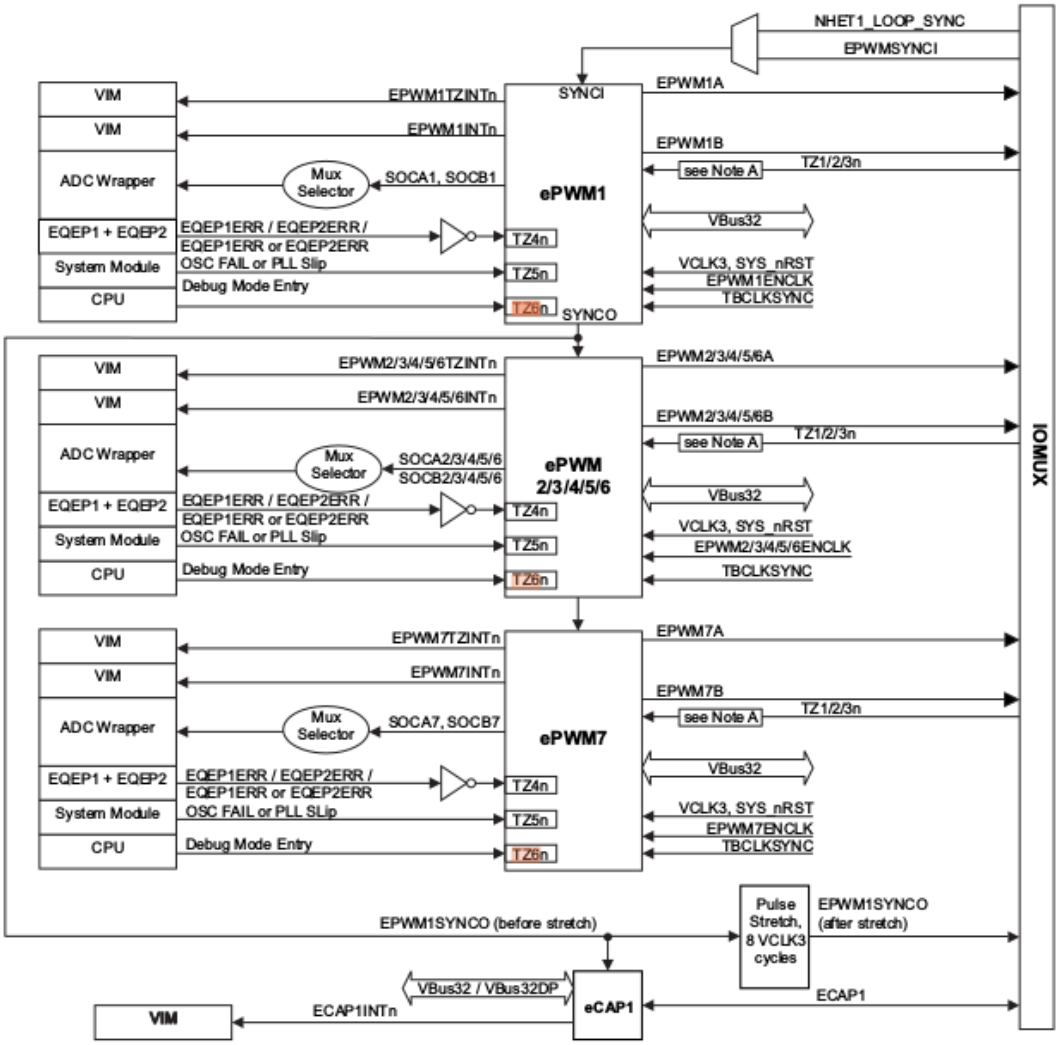
Bits	Name	Value	Description
4-1	OSHTWTH		One-Shot Pulse Width.
		0	1 x VCLK3 / 8 wide (= 80 nS at 100 MHz VCLK3)
		1h	2 x VCLK3 / 8 wide (= 160 nS at 100 MHz VCLK3)
		2h	3 x VCLK3 / 8 wide (= 240 nS at 100 MHz VCLK3)
		3h	4 x VCLK3 / 8 wide (= 320 nS at 100 MHz VCLK3)
		4h	5 x VCLK3 / 8 wide (= 400 nS 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.

vclk 를 8 로 나눔
값이 튈때 vclk 클록을 8 개 단위로 쪼개서 보겠다

TZSEL//

```
/** - Set trip source enable */
etpwmREG2->TZSEL =  0x0000U    /** - Enable/Disable TZ1 as a one-shot trip source */
```

Figure 35-1. Multiple ePWM Modules



TZEINT // 인터럽트 다막음

```
/** - Set interrupt enable */
etpwmREG2->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  */
```

ETSEL//

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

if ((etpwmREG2->ETSEL & 0x0007U) != 0U)
{
    etpwmREG2->ETSEL |= 0x0008U;
}
```

ETPS// 신호가 충돌할 상황을 방지

```
/** - Sets up the ADC SOC period */
etpwmREG2->ETPS |= ((uint16)((uint16)1U << 8U)
                 | (uint16)((uint16)1U << 12U));
```

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).

IOMM//

ePWMx TBCLKSYNC Enable	3A8h	PINMMR166[1]		
ePWM1 Trip Zone 4 Select	3ACh	PINMMR167[0]	PINMMR167[1]	PINMMR167[2]
ePWM2 Trip Zone 4 Select		PINMMR167[8]	PINMMR167[9]	PINMMR167[10]
ePWM3 Trip Zone 4 Select		PINMMR167[16]	PINMMR167[17]	PINMMR167[18]
ePWM4 Trip Zone 4 Select		PINMMR167[24]	PINMMR167[25]	PINMMR167[26]
ePWM5 Trip Zone 4 Select	3B0h	PINMMR168[0]	PINMMR168[1]	PINMMR168[2]
ePWM6 Trip Zone 4 Select		PINMMR168[8]	PINMMR168[9]	PINMMR168[10]
ePWM7 Trip Zone 4 Select		PINMMR168[16]	PINMMR168[17]	PINMMR168[18]

//문제가 발생했을때 이걸 사용해서 꺼버리면된다

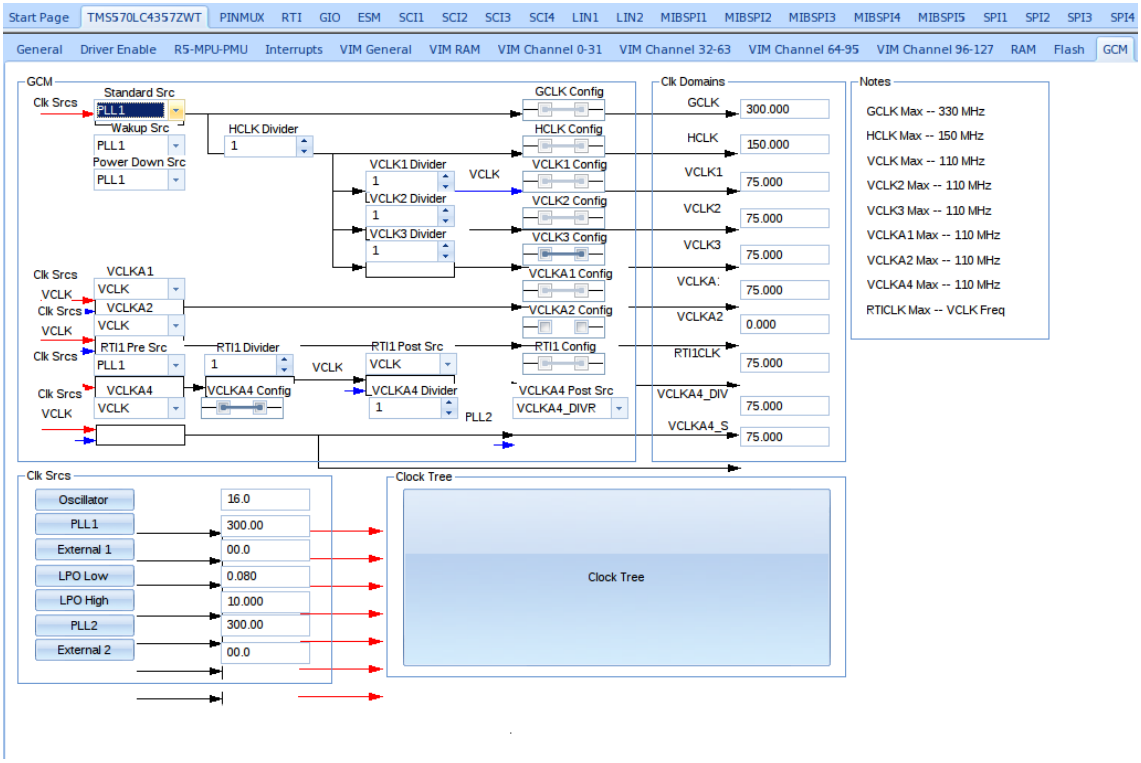
```
#include "HL_sys_common.h"
#include "HL_etpwm.h"

int main(void)
{
    int i;
    unsigned short val = 0;

    etpwmInit();

    for(;;)
    {
        etpwmStartTBCLK();

        for(i=0; i<10000; i++)
            ;

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

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

CmpB 에 값 375
값이 계속 더해진다 500 이 되기전까지

//처음에 떠있는시간이 시간이 지날수록 길어진다
//실제로는 꺼졌다 켜졌다는 반복하는건데
//우리눈에는 점점밝아지는거로 느껴지는거

주기를 20ms

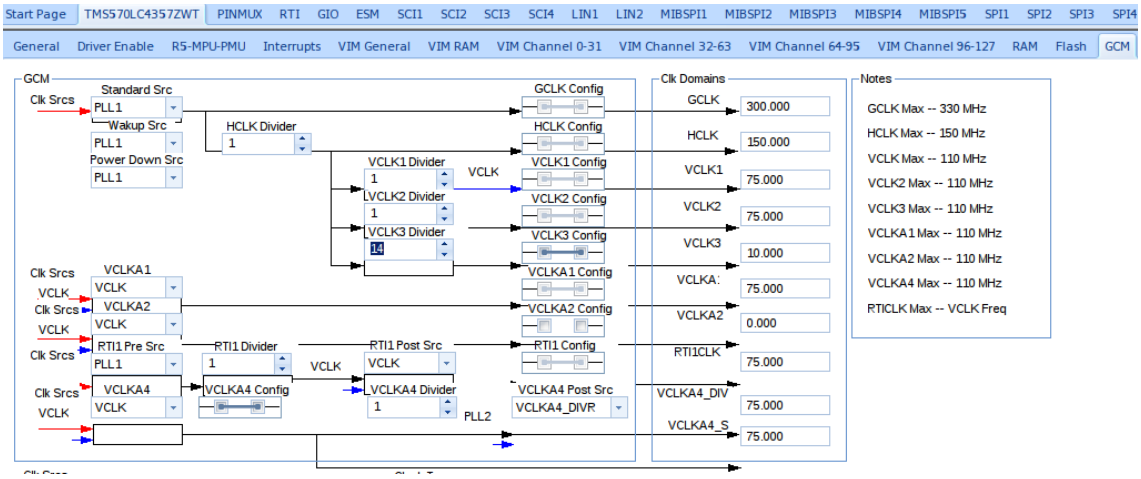
GCM//



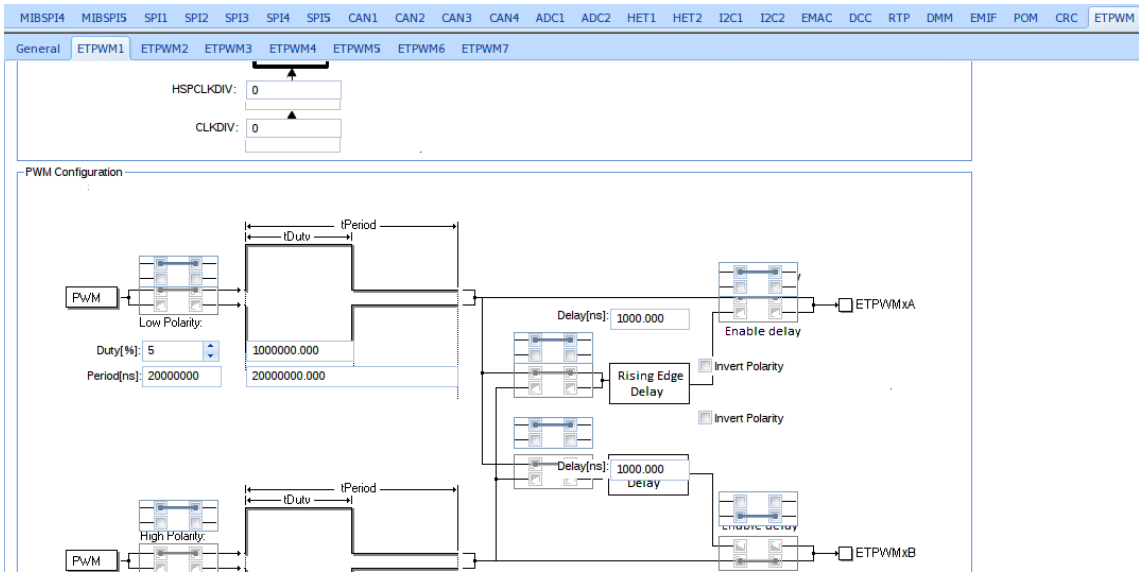
값을 조절하면 분주를 하니까 클록이 떨어지고있다
이것을 사용해서 50 헤르츠를 만들기위해 분주를 해 줄 수있다
우리가 쓰는 신호선

프리페럴 쓰는것 : VCLK3

servoPWM
CCS, HCG 구동



//GCM 14 분주



- 20ms 맞추기위해 period 값 변경필요

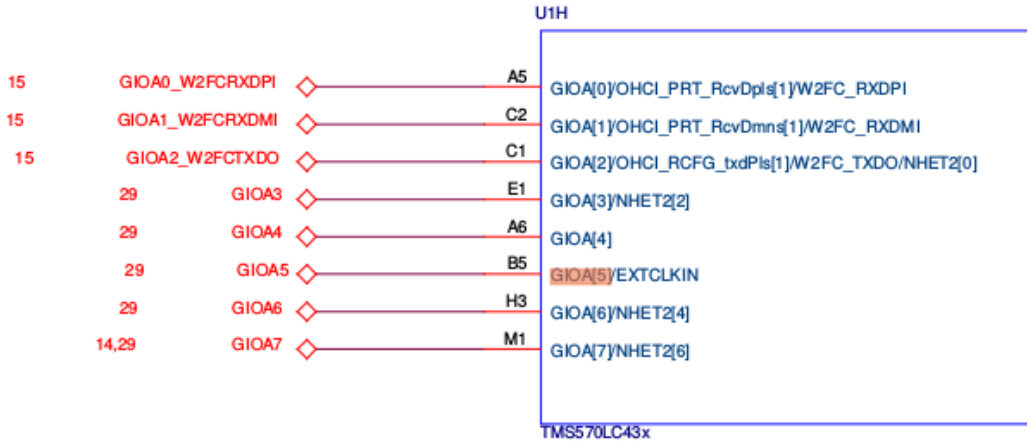
```
#include "HL_sys_common.h"
#include "HL_system.h"
#include "HL_etpwm.h"

uint8   emacAddress[6U] =   {0xFFU, 0xFFU, 0xFFU, 0xFFU, 0xFFU,
0xFFU};
uint32  emacPhyAddress  =   1U;

void pwmset(void);
void delay(uint32);

int main(void)
{
    etpwmInit();
    etpwmStartTBCLK();

    delay(10000);

    for(;;)
    {
        pwmSet();
        delay(50000000);
    }
    return 0;
}

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

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

propertie – include option
"${workspace_loc:/${ProjName}/include}"

그라운드는 갈색
그라운드는 아두 그라운드로

선하나 남은건 a 에 5 번
코어텍스 R 에서 a5 에 꽂으면됨

GIOA[5]




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

uint32 value =0;
uint32 idx =0;
uint32 duty_arr[6]={1000,1200,1400,1600,1800,2000};

void pwmset(void);
void delay(uint32);

int main(void)
{
    etpwmInit();
    etpwmStartTBCLK();
    delay(10000);

    for(;;)
    {
        pwmSet();
        delay(10000000);
    }
    return 0;
}

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

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

PLL1 을 VCLK3 14 분주후 ETPWM 에서 10 분주하고 수업에서 만든 코드 그대로 사용하면 동작