TI DSP, MCU 및 Xilinx Zynq FPGA 프로그래밍 전문가 과정

강사 : Innova Lee(이상훈)
gcccompil3r@gmail.com
학생 : 황수정
sue100012@naver.com
52일차 (2018. 05. 11)

목차

타이머와 카운터

타이머와 카운터는 비슷한 개념을 가지고 있지만, 사용방법이 다르다. 타이머 MCU의 내부클럭을 이용하여 일정 시간 간격의 펄스를 만들어 내거나 일정시간 경과 후에 인터럽트를 발생시키는 기능을 한다.

카운터는 외부핀을 통해서 들어오는 펄수의 수를 세는 역할을 한다. 타이머와 달리 외부 클럭을 이용하며 주기적으로 같은 동작을 반복하는 곳이나 여러 번에 나누어 해야 할 행동이 있을 때 timer를 이용한다.

- 관련 링크

https://wlsgk123123.blog.me/10090312114

https://blog.naver.com/wldino/30137921250

ETPWM 을 통한 LED 제어

PWM = pulse width modulation

```
#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;
}
```

- duty비가 바뀌면서 LED 가 점점 강하게 들어오는 것을 확인할 수 있다.

```
void etpwmInit(void)
{    /** @b initialize @b ETPWM2 */
   /** - Sets high speed time-base clock prescale bits */
```

35.4.1.2 Time-Base Control Register (TBCTL)

Figure 35-64. Time-Base Control Register (TBCTL) [offset = 02h]

15	14	13	12	10	9	8
FREE	SOFT	PHSDIR	CLKDIV		HSPCLKDIV	
R/W-0	R/W-0	R/W-0	R/W-0		R/W-0	

7	6	5	4	3	2	1	0
HSPCLKDIV	SWFSYNC	SYNCOSEL		PRDLD	PHSEN	CTRMODE	
R/W-1	R/W-0	R/W-0		R/W-0	R/W-0	R/W-3h	

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

- 분주 비트 설정

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

```
   etpwmREG2->TBCTL = (uint16)0U << 7U;
```

- 7비트에 0을 셋팅하겠다는 뜻이다.

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

- 10비트에 0을 셋팅한다.

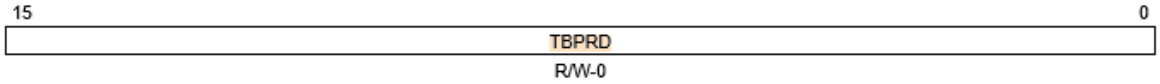
time-base clock prescale value : TBCLK = VCLK3 / (HSPCLKDIV × CLKDIV) 을 결정하는 비트이다. 두 비트 다 0으로 셋팅했으므로 분주를 하지 않는다는 뜻이 된다. HAL에서 HSPCLKDIV와 CLKDIV의 값을 확인할 수 있다.

```
   /** - Sets time period or frequency for ETPWM block both PWMA and PWMB*/
   etpwmREG2->TBPRD = 374U; /*주파수 설정*/
```

- 이 비트는 time-base 카운터의 주기를 결정하며 PWM의 주파수를 설정한다. HAL에서 period를 1000으로 바꾸면 374가 749로 바뀐다. 이 레지스터의 백업은 TBCTL[PRDLD]비트에 의해 활성화 및 비활성화가 된다.

35.4.1.4 Time-Base Period Register (TBPRD)

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



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

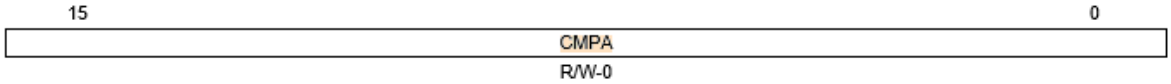
Table 35-26. Time-Base Period Register (TBPRD) Field Descriptions

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.

```
/** - Setup the duty cycle for PWMA */
etpwmREG2->CMPA = 188U;
```

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.

구동되는 CMPA의 값과 연속적으로 time-base카운터(TBCTR)과 비교한다. 이벤트가 발생하면 전송한다는 것으로 인터럽트를 발생시킨다는 것이다. 즉, CMPA와 TBCTR의 레지스터 값이 같을 때, "time-base counter equal to counter compare A"이벤트(인터럽트)를 발생시킨다는 것이다.
CMPCTL 비트에 의해 뒤에서 백업하고 있다.

```
/** - Setup the duty cycle for PWMB */
etpwmREG2->CMPB = 188U;
```

- CMPA와 같다.

/** - 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));

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.
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.

➢ Action-Qualifier Output A Control Register 로, 0과 4비트의 값을 알면 된다. 0비트는 카운터가 0과 같을 때 작업을 한다. 4비트는 카운터가 활동중인 CMPA 레지스터같고 카운터가 증가할 때 작동한다. 비트 1, 비트 4를 1로 세팅한다. 그러나 HAL에서 B를 지정했기 때문에 A는 사용하지 않는다.

/** - Force EPWMxB output high when counter reaches zero and low when counter reaches Compare B value */

etpwmREG2->AQCTLB = ((uint16)((uint16)*ActionQual_Set* << 0U)

| (uint16)((uint16)*ActionQual_Clear* << 8U));

		3h	Toggle EPWMxB 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 EPWMxB output low.
		2h	Set: force EPWMxB output high.
		3h	Toggle EPWMxB 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.

➢ 0비트는 AQCTLA와 같다. 8비트도 AQCTLA 4비트와 같은 역할을 한다. 카운터가 0으로 같을 때 EPWMxB의 출력을 low로 한다.

/** - 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

*/

➢ DEAD-BAND란 지나가는 시점을 계산해서 delay를 넣어주라는 것이다.

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  */
```

➢ 5~0비트를 0으로 셋팅한다.

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

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

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

    /** - Set trip source enable */
    etpwmREG2->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      */

    /** - 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  */

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

    if ((etpwmREG2->ETSEL & 0x0007U) != 0U)
    {
        etpwmREG2->ETSEL |= 0x0008U;
    }
    /** - Setup the frequency of the interrupt generation */
    etpwmREG2->ETPS = 1U;

    /** - Sets up the ADC SOC interrupt */
    etpwmREG2->ETSEL |= ((uint16)(0x0000U)
                      | (uint16)(0x0000U)
                      | (uint16)((uint16)DCAEVT1 << 8U)
                      | (uint16)((uint16)DCBEVT1 << 12U));

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

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

int idx = 0;
uint32 value = 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(5000000000);
    }
// 하드웨어적으로 FPGA를 할 때, 좋은 점 중에 하나이다. 소프트웨어적으로 하게 되면 코드를 기계어로 바꾸면서 실행하는데 시간이 걸리는데, 하드웨어적으로 만들어주면 바로 처리가 가능하기 때문이다. 소프트웨어적으로 무엇을 하던(delay를 주던) 하드웨어적으로 실행이 가능하기 때문이다.
    return 0;
}

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

void pwmSet(void)
{
    value = duty_arr[idx % 6];
    idx++;
    etpwmSetCmpA(etpwmREG1, value);
}
```