TI DSP, MCU, Xilinx Zynq FPGA 기반의 프로그래밍 전문가 과정

<펌웨어 프로그래밍>
2018.05.11 - 52일차

강사 – Innova Lee(이상훈)
gcccompil3r@gmail.com

학생 – 안상재
sangjae2015@naver.com

프로젝트 : <SERVO_PWM>

-> 서보모터 구동시키기

1. main()

```
#include "HL_sys_common.h"
#include "HL_system.h"
#include "HL_etpwm.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};  // 듀티비 조정하기 위한 CMPA 값 배열

void pwmSet(void);
void delay(uint32);

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

    delay(10000);

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

}

void pwmSet(void)
{
    value = duty_arr[idx % 20];
    etpwmSetCmpA(etpwmREG1, value);
    idx++;
}
void delay(uint32 delay)
{
    int i;
    for(i=0;i<delay;i++);
}
```

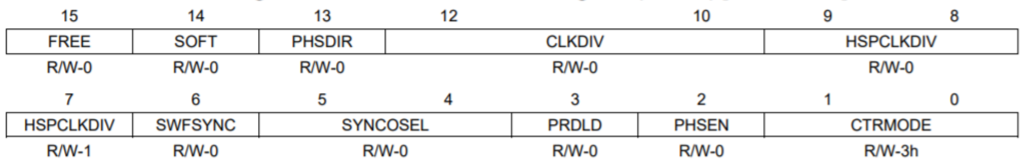
2. main() => etpwmInit()

```
void etpwmInit(void)
{
    /** @b initialize @b ETPWM1 */

    /** - Sets high speed time-base clock prescale bits */
    etpwmREG1->TBCTL = (uint16)5U << 7U;  // TBCLK = VCLK3 / (HSPCLKDIV × CLKDIV) 에서
                                             HSPCLKDIV 를 10분주로 설정 */

    /** - Sets time-base clock prescale bits */
    etpwmREG1->TBCTL |= (uint16)((uint16)0U << 10U); /* TBCLK = VCLK3 / (HSPCLKDIV × CLKDIV)
                                                     에서 CLKDIV를 1분주(default)로 설정 */
    /** - Sets time period or frequency for ETPWM block both PWMA and PWMB*/
    etpwmREG1->TBPRD = 19999U;  /* time-base counter 의 주기를 결정한다. 전체 pwm의 주기를
                                   time-base clock의 주기로 나눈 값 */

    /** - Setup the duty cycle for PWMA */
    etpwmREG1->CMPA = 1000U;    /* TBCTR값과 지속적으로 비교되면서 두 값이 같아지면
                                "time-base counter equal to counter compare A" 이벤트를
                                 발생시킨다. AQCTLA 레지스터 값에 따라 이벤트 action이 결정됨 */

    /** - Setup the duty cycle for PWMB */
    etpwmREG1->CMPB = 1000U;

    /** - Force EPWMxA output high when counter reaches zero and low when counter reaches
Compare A value */
    etpwmREG1->AQCTLA = ((uint16)((uint16)ActionQual_Set   << 0U)  /* counter가 0이 될 때,
                                                               EPWMxA 출력을 high로 셋팅 */
                       | (uint16)((uint16)ActionQual_Clear << 4U)); /* counter가 증가하다가
                                           CMPA 레지스터와 같아질 때, EPWMxA 출력이 low가 됨 */

    /** - 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));

    /** - 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*/

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

    /** - Set the falling edge delay  */
    etpwmREG1->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
    */
    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 */

    /** - Set trip source enable */
    etpwmREG1->TZSEL = 0x0000U;

    /** - Set interrupt enable */
    etpwmREG1->TZEINT = 0x0000U;
    /** - 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));
}
```

- TBCTL 레지스터

35.4.1.2 Time-Base Control Register (TBCTL)

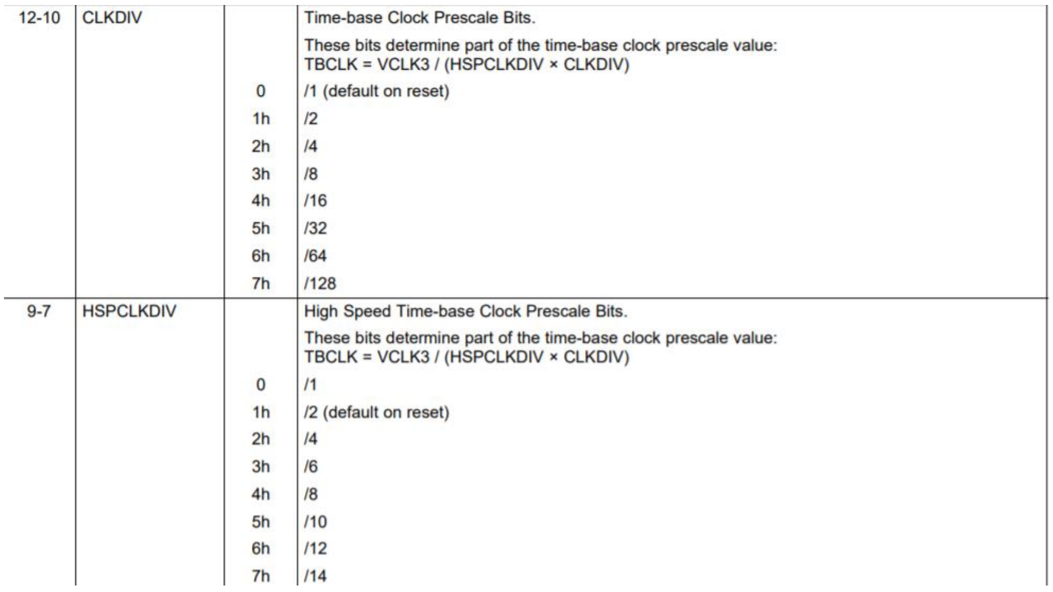
Figure 35-64. Time-Base Control Register (TBCTL) [offset = 02h]



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

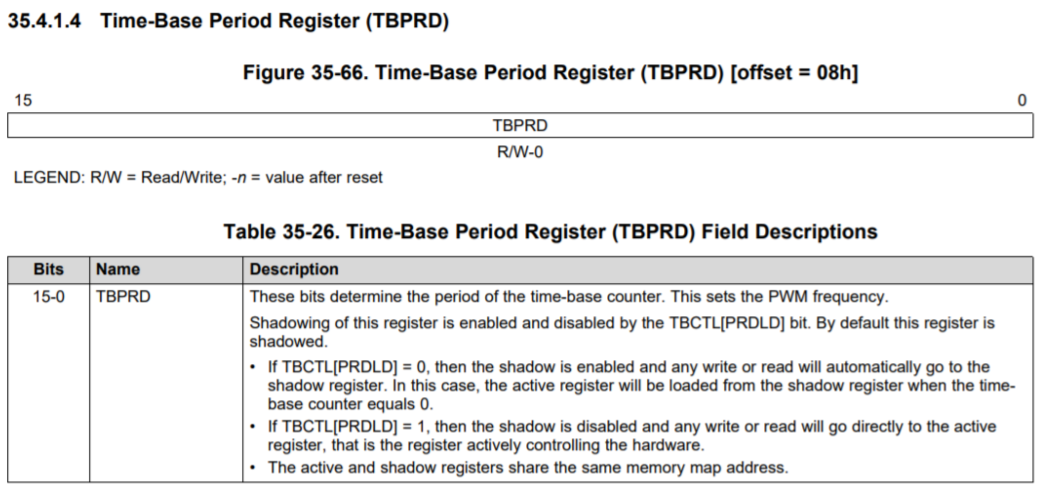
7	6	5-4	3	2	1-0
HSPCLKDIV	SWFSYNC	SYNCOSEL	PRDLD	PHSEN	CTRMODE
R/W-1	R/W-0	R/W-0	R/W-0	R/W-0	R/W-3h

- CLKDIV 와 HSPCLKDIV 는 VCLK3을 분주하는 값을 정함.



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

- TBPRD 레지스터 : time-base counter 의 주기를 결정함



35.4.1.4 Time-Base Period Register (TBPRD)

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

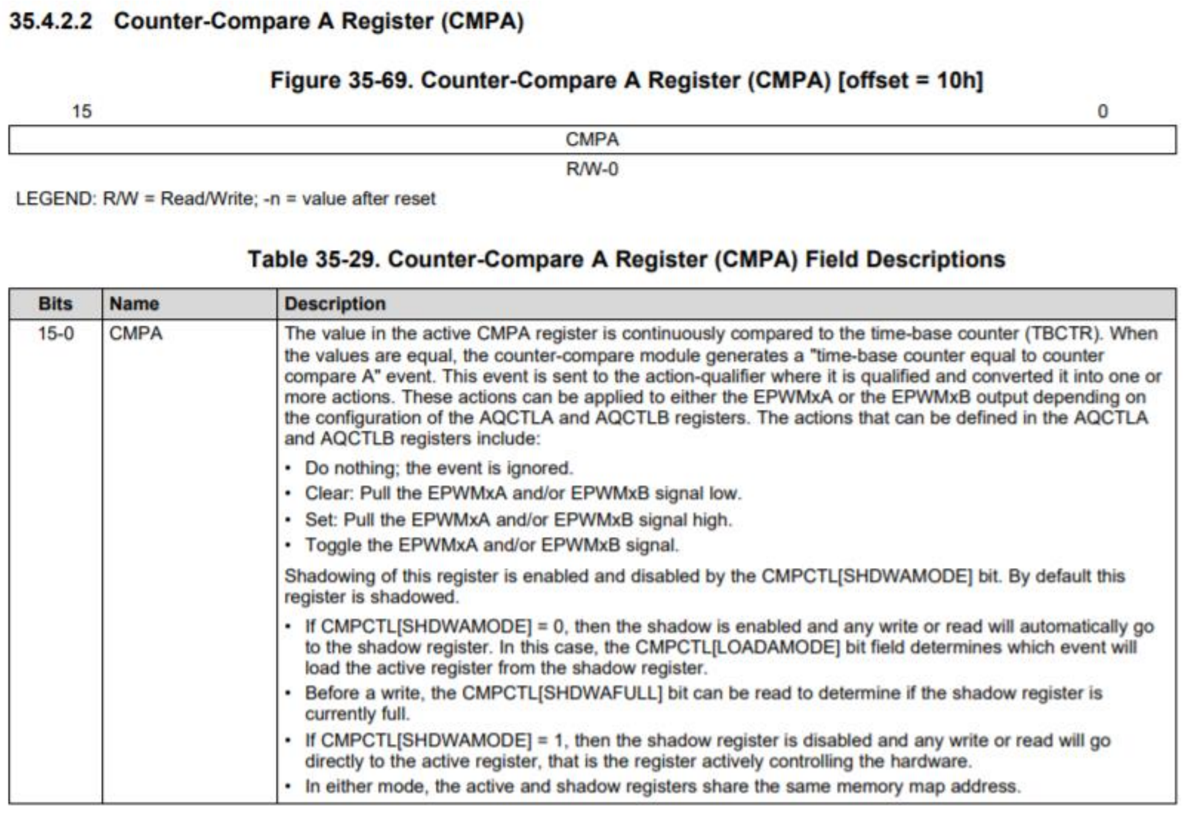
15 … 0
TBPRD
R/W-0

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.

- CMPA 레지스터 : 듀티비를 결정하는 레지스터



35.4.2.2 Counter-Compare A Register (CMPA)

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

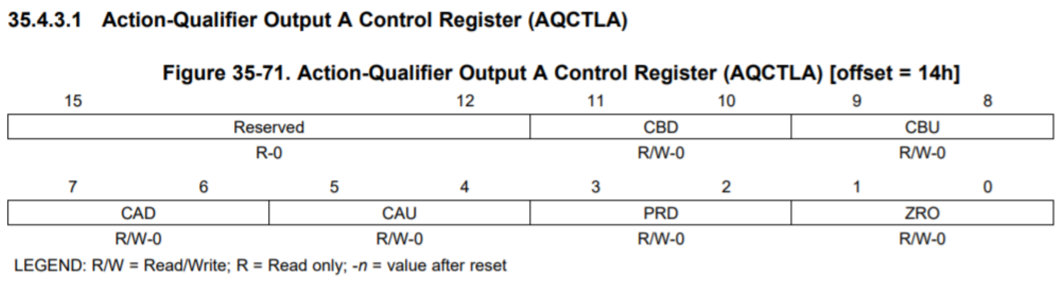
15 … 0
CMPA
R/W-0

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.

- AQCTLA 레지스터 : counter 가 0이 될 때와 CMPA 값과 같아질 때의 action 을 결정함.



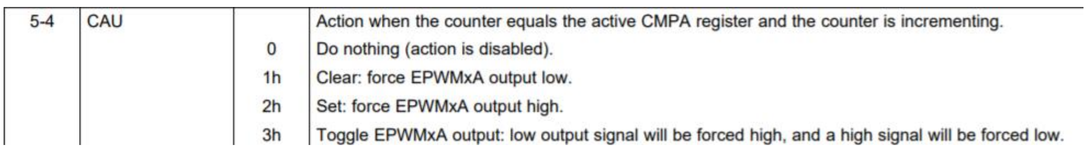
35.4.3.1 Action-Qualifier Output A Control Register (AQCTLA)

Figure 35-71. Action-Qualifier Output A Control Register (AQCTLA) [offset = 14h]

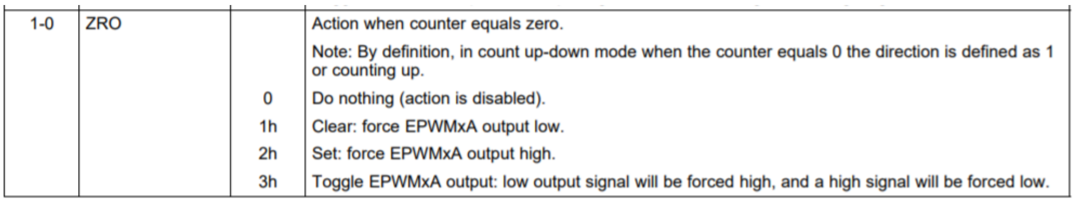
15–12	11–10	9–8
Reserved	CBD	CBU
R-0	R/W-0	R/W-0

7–6	5–4	3–2	1–0
CAD	CAU	PRD	ZRO
R/W-0	R/W-0	R/W-0	R/W-0

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



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.

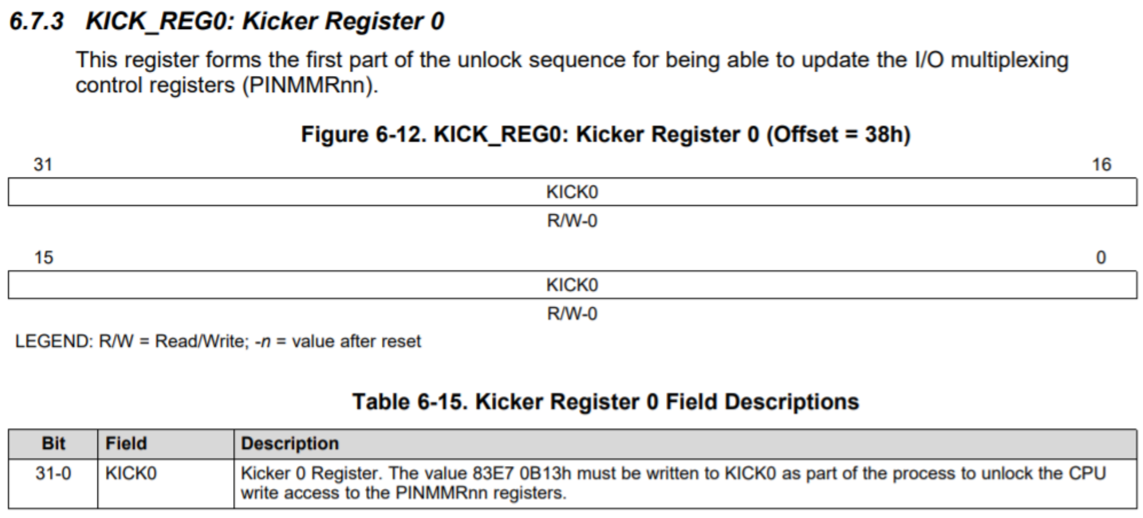


1-0	ZRO		Action when counter equals zero.
			Note: By definition, in count up-down mode when the counter equals 0 the direction is defined as 1 or counting up.
		0	Do nothing (action is disabled).
		1h	Clear: force EPWMxA output low.
		2h	Set: force EPWMxA output high.
		3h	Toggle EPWMxA output: low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

3. main() => `etpwmStartTBCLK()`

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

      // PINMUX_ETPWM_TBCLK_SYNC_MASK = 0x00, PINMUX_ETPWM_TBCLK_SYNC_ON = 0x2U
      pinMuxReg->PINMUX[166U] = (pinMuxReg->PINMUX[166U] & PINMUX_ETPWM_TBCLK_SYNC_MASK) |
(PINMUX_ETPWM_TBCLK_SYNC_ON);

    /* Disable Pin Muxing */
    pinMuxReg->KICKER0 = 0x00000000U;  // CPU가 PINMMRnn 레지스터에 접근하는 것을 lock시킴
    pinMuxReg->KICKER1 = 0x00000000U;
}
```

- KICKER0 : KICKER0 레지스터에 0x83E70B13U 를 write 하면 cpu 가 PINMMRnn 레지스터에 접근하는 것을 허용함.



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

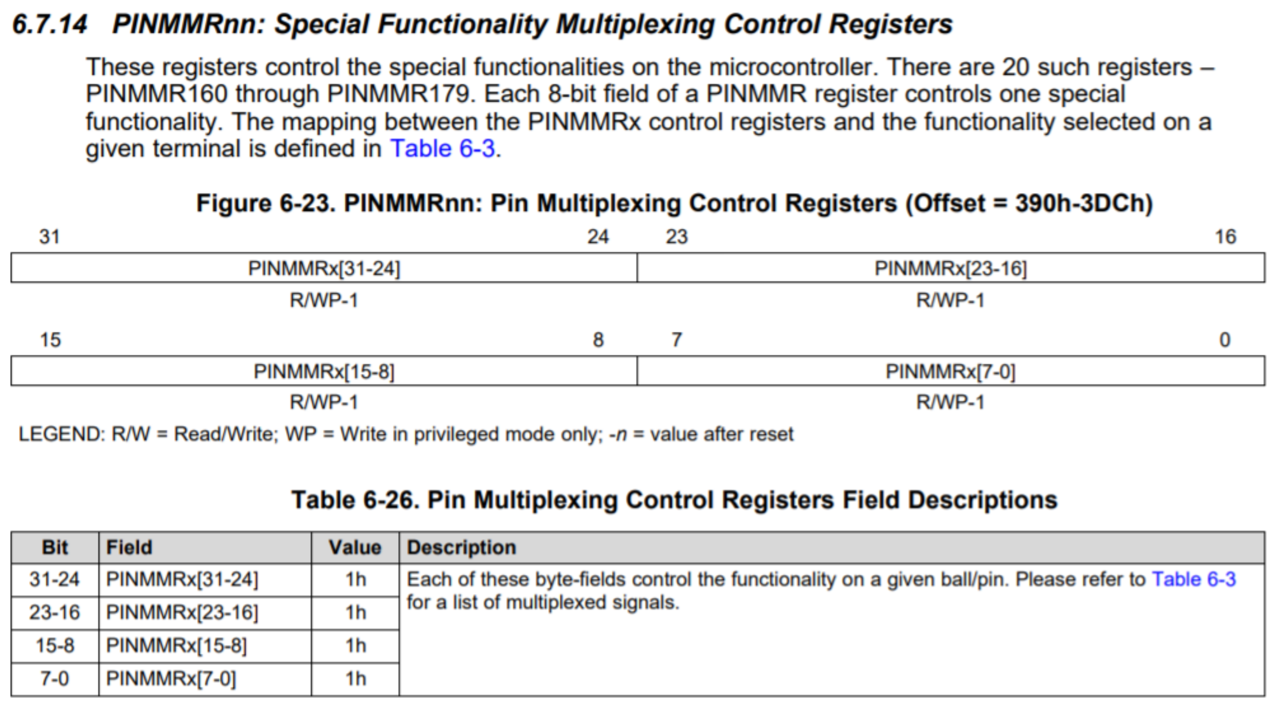
15 … 0
KICK0
R/W-0

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.

- PINMUX[166U] (Special Functionality Multilexing Control Register) 레지스터 찾기



6.7.14 *PINMMRnn: Special Functionality Multiplexing Control Registers*

These registers control the special functionalities on the microcontroller. There are 20 such registers – PINMMR160 through PINMMR179. Each 8-bit field of a PINMMR register controls one special functionality. The mapping between the PINMMRx control registers and the functionality selected on a given terminal is defined in Table 6-3.

Figure 6-23. PINMMRnn: Pin Multiplexing Control Registers (Offset = 390h-3DCh)

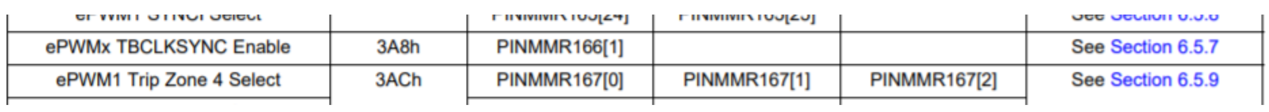
31 ... 24	23 ... 16
PINMMRx[31-24]	PINMMRx[23-16]
R/WP-1	R/WP-1
15 ... 8	**7 ... 0**
PINMMRx[15-8]	PINMMRx[7-0]
R/WP-1	R/WP-1

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

Table 6-26. Pin Multiplexing Control Registers Field Descriptions

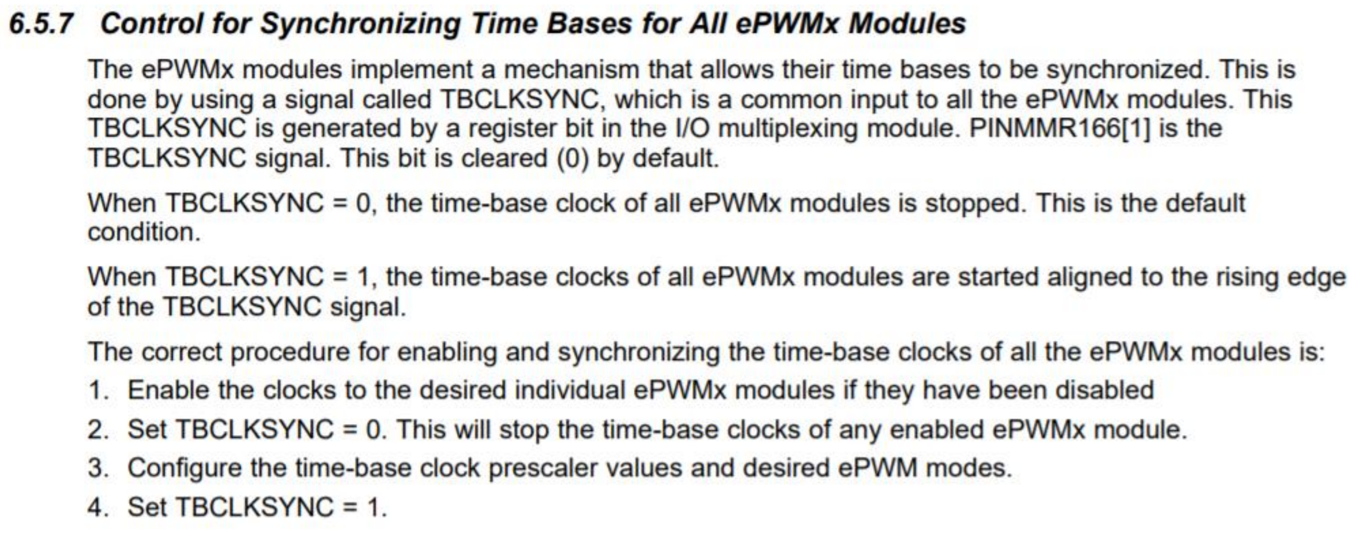
Bit	Field	Value	Description
31-24	PINMMRx[31-24]	1h	Each of these byte-fields control the functionality on a given ball/pin. Please refer to Table 6-3 for a list of multiplexed signals.
23-16	PINMMRx[23-16]	1h	
15-8	PINMMRx[15-8]	1h	
7-0	PINMMRx[7-0]	1h	

-> 위 그림에서 Table 6-3 클릭



ePWM1 SYNCI Select		PINMMR165[24]	PINMMR165[25]		See Section 6.5.8
ePWMx TBCLKSYNC Enable	3A8h	PINMMR166[1]			See Section 6.5.7
ePWM1 Trip Zone 4 Select	3ACh	PINMMR167[0]	PINMMR167[1]	PINMMR167[2]	See Section 6.5.9

-> PINMMR166[1] 을 보기 위해 Section 6.5.7 클릭



6.5.7 *Control for Synchronizing Time Bases for All ePWMx Modules*

The ePWMx modules implement a mechanism that allows their time bases to be synchronized. This is done by using a signal called TBCLKSYNC, which is a common input to all the ePWMx modules. This TBCLKSYNC is generated by a register bit in the I/O multiplexing module. PINMMR166[1] is the TBCLKSYNC signal. This bit is cleared (0) by default.

When TBCLKSYNC = 0, the time-base clock of all ePWMx modules is stopped. This is the default condition.

When TBCLKSYNC = 1, the time-base clocks of all ePWMx modules are started aligned to the rising edge of the TBCLKSYNC signal.

The correct procedure for enabling and synchronizing the time-base clocks of all the ePWMx modules is:

1. Enable the clocks to the desired individual ePWMx modules if they have been disabled
2. Set TBCLKSYNC = 0. This will stop the time-base clocks of any enabled ePWMx module.
3. Configure the time-base clock prescaler values and desired ePWM modes.
4. Set TBCLKSYNC = 1.

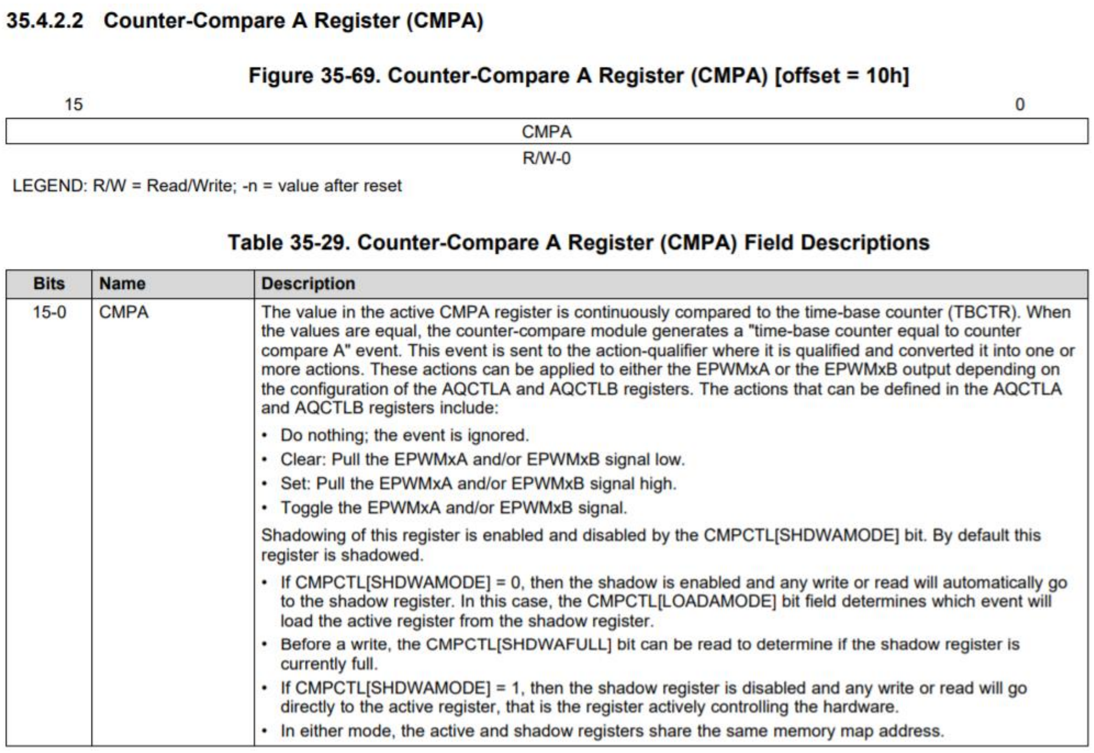
-> PINMMR166[1] 이라고 되어 있기 때문에 바로 위의 데이터 시트에서 언급하는 TBCLKSYNC 는 PINMMR166[1] 에 위치해 있는 것으로 추정됨.

-> PINMMR166[1] 에 1을 write 함으로써 ePWMx module 이 구동을 시작함.

4. main() => pwmSet()

```
void pwmSet(void)
{
    value = duty_arr[idx % 20];
    etpwmSetCmpA(etpwmREG1, value);  // CMPA 값을 duty_arr[] 배열의 값에서 차례로 변경함.
    idx++;
}
```

- CMPA 레지스터 : TBCTL 과 실시간으로 비교함



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.