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

날 짜 : 2018 . 5. 1

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

etPWM

<code>

```
#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;
}
```

---

etpwmInit()

```
/** - Sets high speed time-base clock prescale bits */
// TBCTL time base clock control register 분주 비트 설정이다.
// 7 번 비트는 HSPCLKDIV 관련한 분주 비트이다. 0 으로 설정되어 있다. → 나중에 이걸 이용해 분주를 더 할 수 있다. ( high speed 클럭 디비젼 ) TBCLK = VCLK3 / (HSPCLKDIV × CLKDIV)
    etpwmREG2->TBCTL = (uint16)0U << 7U;

/** - Sets time-base clock prescale bits */
// 10 번 비트는 CLKDIV 의 분주를 설정한다. TBCLK = VCLK3 / (HSPCLKDIV × CLKDIV)
    etpwmREG2->TBCTL |= (uint16)((uint16)0U << 10U);

    /** - Sets time period or frequency for ETPWM block both PWMA and PWMB*/
// 이 비트도 중요하다. count 되는 값을 설정하여 주기를 몇으로 줄지 설정한다.
    etpwmREG2->TBPRD = 374U;
```

```
    /** - Setup the duty cycle for PWMA */
// 비교 할 듀티 값을 설정해 준다. 위의 count 값의 ½ 정도의 값인데 duty 를 50 프로로 설정해서 그런듯 함.
    etpwmREG2->CMPA = 188U;

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



   /** - Force EPWMxA output high when counter reaches zero and low when counter reaches Compare A value */
// clear 함, CMPA 레지스터의 count 값을 증가 형으로 함.
    etpwmREG2->AQCTLA = ((uint16)((uint16)ActionQual_Set   << 0U)
                       | (uint16)((uint16)ActionQual_Clear << 4U));

    /** - Force EPWMxB output high when counter reaches zero and low when counter reaches Compare B value */
// count 가 0 일시 동작, CMPB 레지스터의 count 값을 증가 형으로 함.
    etpwmREG2->AQCTLB = ((uint16)((uint16)ActionQual_Set   << 0U)
                       | (uint16)((uint16)ActionQual_Clear << 8U));



// on-off 식 듀티 제어 50%로 설정한다. 1/8 duty 로 설정. Divide by 1 (no prescale, = 12.5 MHz // at 100 MHz VCLK3)
 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 */
```

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)

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.

dead – Band

delay 를 주어서 임계치를 넘은 파형을 잘라 내어 값이 들어오게 한다. 그러면 마치 사각파 형태로 나온다.

MUX 조작

mux 를 조작하기 위해서는 반드시(KICKER0, KICKER1) → **매직 넘버**를 설정해 주어야 한다.

etpwmStartTBCLK()

```
// 매직 넘버를 설정해 준다. ( mux 를 조작 하기 위해서)
pinMuxReg->KICKER0 = 0x83E70B13U;
pinMuxReg->KICKER1 = 0x95A4F1E0U;

// PINMUX 가 166 이면 Spetial 부분의 6 번을 나타낸다.
// 166 번 mux 는 ePWMx TBCLKSYNC Enable 를 나타낸다. 끄면 그핀에 연결한 녀석도 꺼진다.(모터비상시)
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;
```

VCLK3

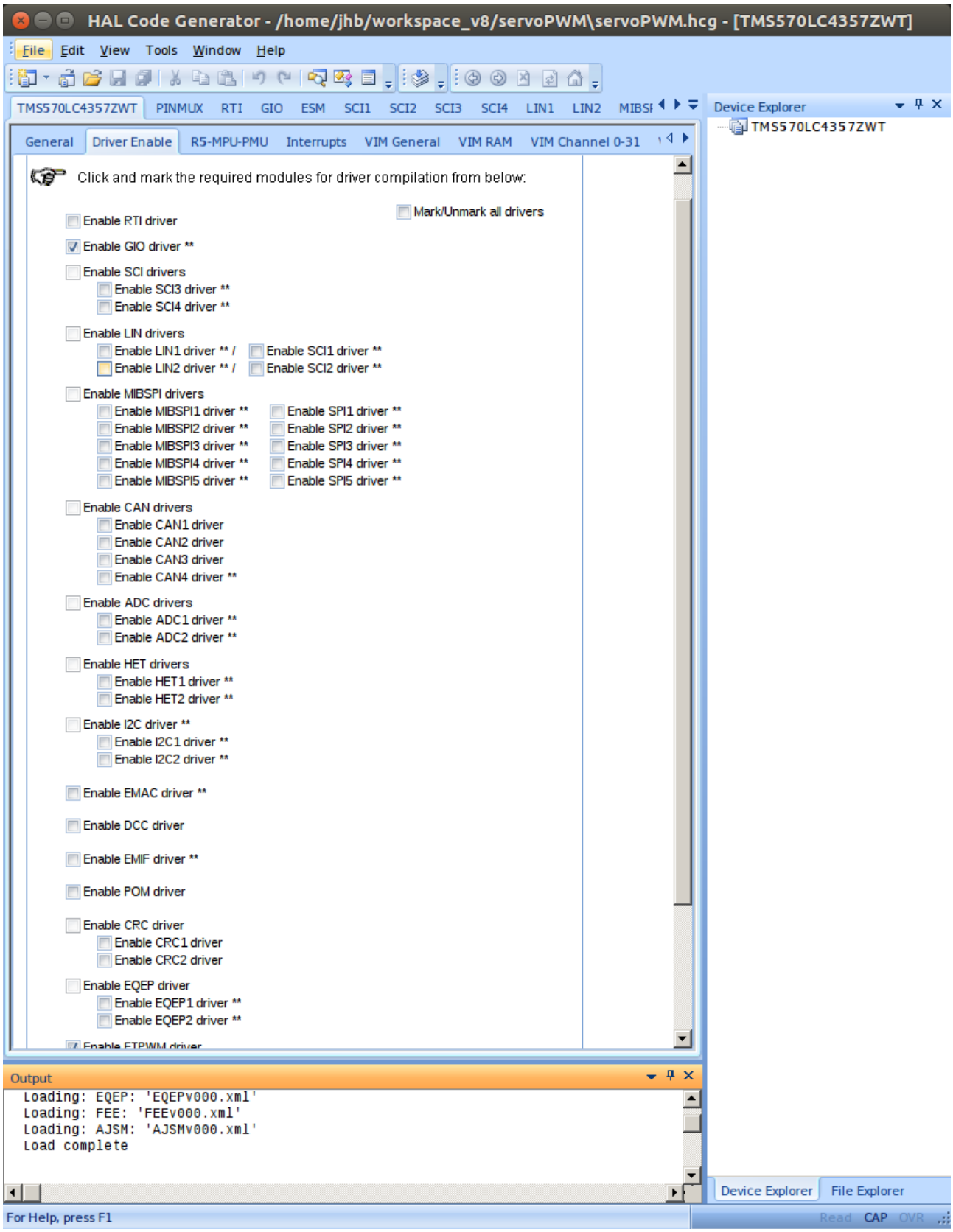
PWM 과 관련한 녀석을 쓸 때 클럭설정하는 레지스터 이다.

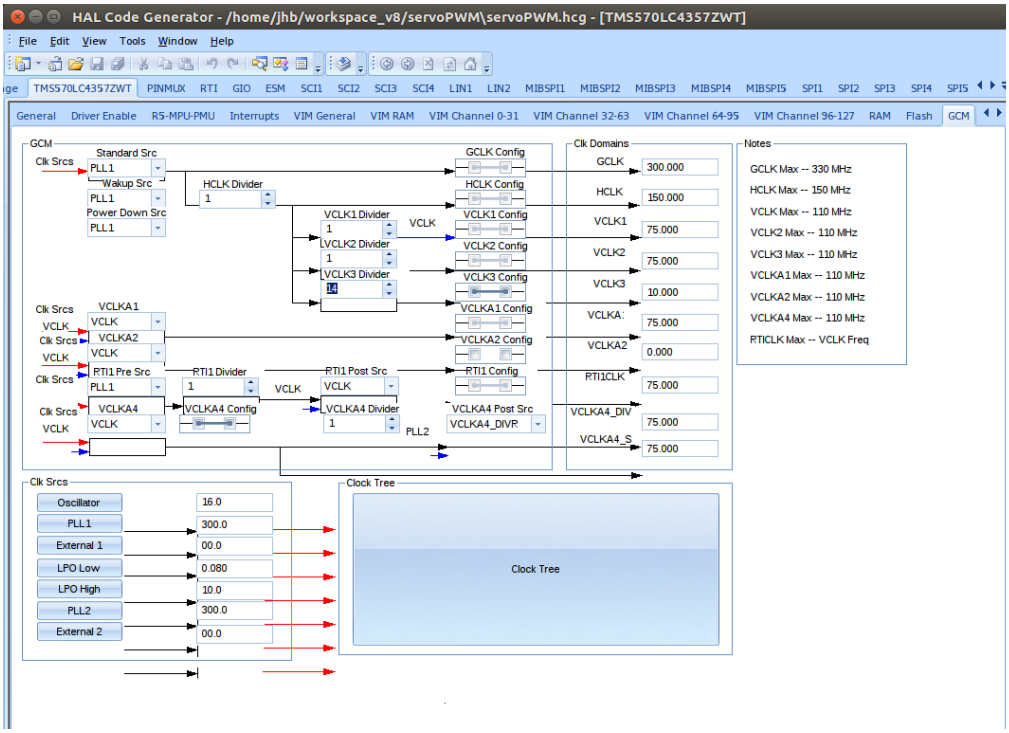
VCLK2

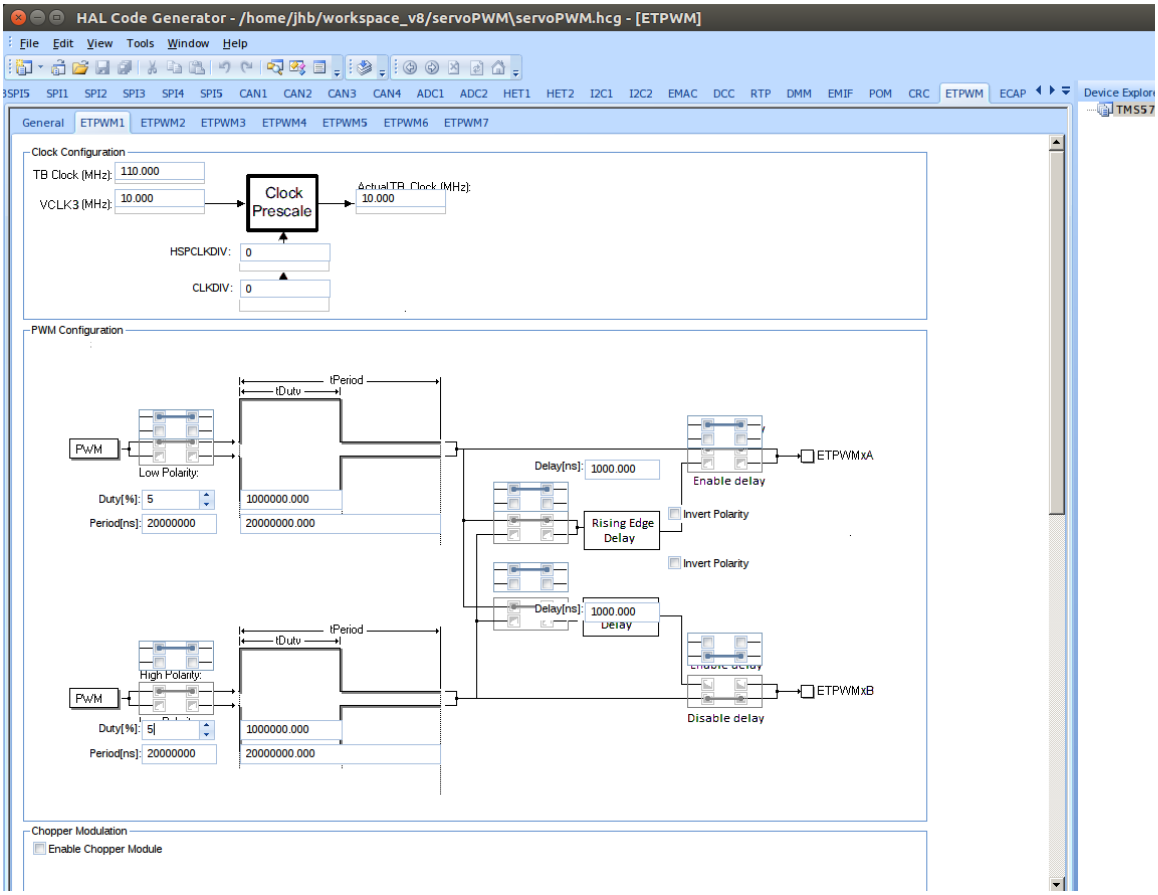
HET(HIGH END TIMER)를 쓸 때, 레지스터 이다.

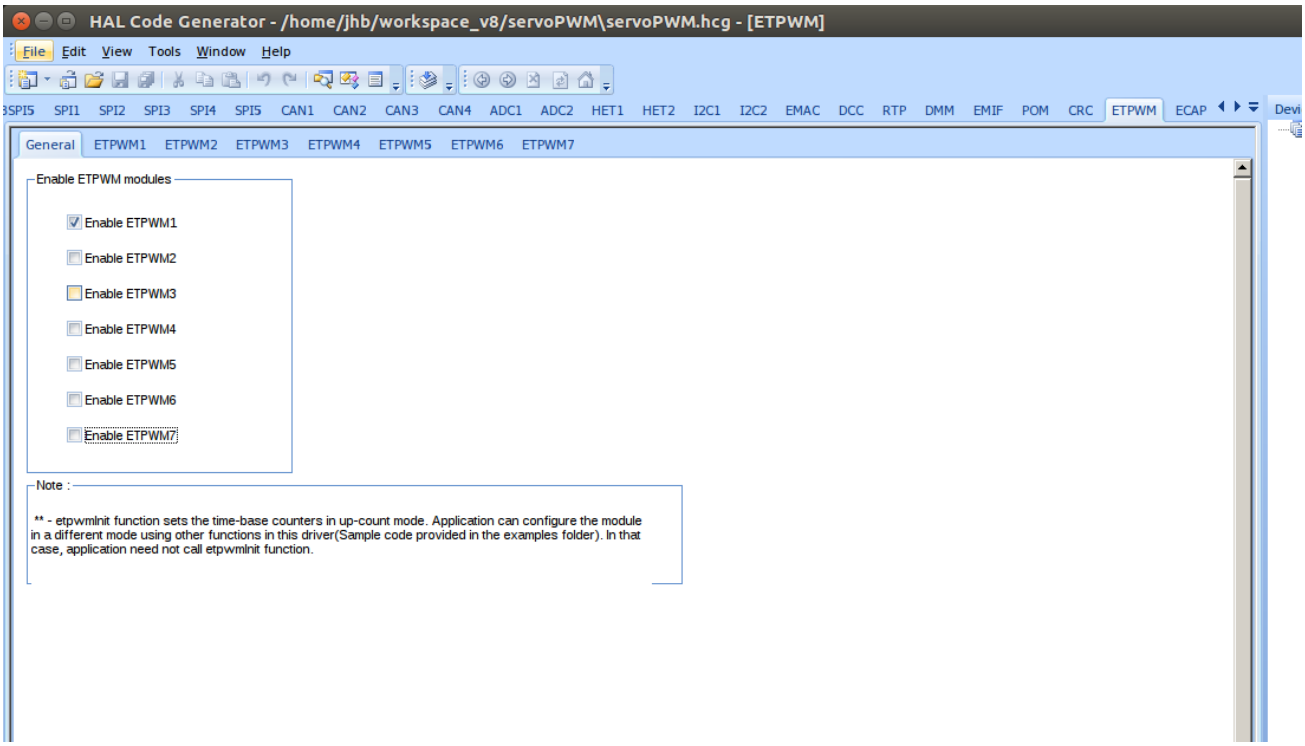
SERVO PWM

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

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


HAL Code Generator - /home/jhb/workspace_v8/servoPWM\servoPWM.hcg - [TMS570LC4357ZWT]
File Edit View Tools Window Help
TMS570LC4357ZWT PINMUX RTI GIO ESM SCI1 SCI2 SCI3 SCI4 LIN1 LIN2 MIBSI
Device Explorer
TMS570LC4357ZWT
General Driver Enable R5-MPU-PMU Interrupts VIM General VIM RAM VIM Channel 0-31
Click and mark the required modules for driver compilation from below:
Mark/Unmark all drivers
Enable RTI driver
Enable GIO driver **
Enable SCI drivers
Enable SCI3 driver **
Enable SCI4 driver **
Enable LIN drivers
Enable LIN1 driver ** /
Enable LIN2 driver ** /
Enable SCI1 driver **
Enable SCI2 driver **
Enable MIBSPI drivers
Enable MIBSPI1 driver **
Enable MIBSPI2 driver **
Enable MIBSPI3 driver **
Enable MIBSPI4 driver **
Enable MIBSPI5 driver **
Enable SPI1 driver **
Enable SPI2 driver **
Enable SPI3 driver **
Enable SPI4 driver **
Enable SPI5 driver **
Enable CAN drivers
Enable CAN1 driver
Enable CAN2 driver
Enable CAN3 driver
Enable CAN4 driver **
Enable ADC drivers
Enable ADC1 driver **
Enable ADC2 driver **
Enable HET drivers
Enable HET1 driver **
Enable HET2 driver **
Enable I2C driver **
Enable I2C1 driver **
Enable I2C2 driver **
Enable EMAC driver **
Enable DCC driver
Enable EMIF driver **
Enable POM driver
Enable CRC driver
Enable CRC1 driver
Enable CRC2 driver
Enable EQEP driver
Enable EQEP1 driver **
Enable EQEP2 driver **
Output
Loading: EQEP: 'EQEPv000.xml'
Loading: FEE: 'FEEv000.xml'
Loading: AJSM: 'AJSMv000.xml'
Load complete
Device Explorer File Explorer
For Help, press F1
Read CAP OVR


HAL Code Generator - /home/jhb/workspace_v8/servoPWM\servoPWM.hcg - [TMS570LC4357ZWT]
File Edit View Tools Window Help
TMS570LC4357ZWT PINMUX RTI GIO ESM SCI1 SCI2 SCI3 SCI4 LIN1 LIN2 MIBSPI1 MIBSPI2 MIBSPI3 MIBSPI4 MIBSPI5 SPI1 SPI2 SPI3 SPI4 SPI5
General Driver Enable R5-MPU-PMU Interrupts VIM General VIM RAM VIM Channel 0-31 VIM Channel 32-63 VIM Channel 64-95 VIM Channel 96-127 RAM Flash GCM
GCM
Clk Srcs
Standard Src
PLL1
Wakup Src
Power Down Src
HCLK Divider
VCLK1 Divider
VCLK2 Divider
VCLK3 Divider
VCLK
GCLK Config
HCLK Config
VCLK1 Config
VCLK2 Config
VCLK3 Config
VCLKA1
VCLKA2
RTI1 Pre Src
RTI1 Divider
RTI1 Post Src
RTI1 Config
VCLKA4
VCLKA4 Config
VCLKA4 Divider
VCLKA4 Post Src
VCLKA4_DIVR
PLL2
Clk Domains
GCLK 300.000
HCLK 150.000
VCLK1 75.000
VCLK2 75.000
VCLK3 10.000
VCLKA 75.000
VCLKA2 0.000
RTI1CLK 75.000
VCLKA4_DIV 75.000
VCLKA4_S 75.000
Notes
GCLK Max -- 330 MHz
HCLK Max -- 150 MHz
VCLK Max -- 110 MHz
VCLK2 Max -- 110 MHz
VCLK3 Max -- 110 MHz
VCLKA1 Max -- 110 MHz
VCLKA2 Max -- 110 MHz
VCLKA4 Max -- 110 MHz
RTICLK Max -- VCLK Freq
Clk Srcs
Oscillator 16.0
PLL1 300.0
External 1 00.0
LPO Low 0.080
LPO High 10.0
PLL2 300.0
External 2 00.0
Clock Tree
Clock Tree


HAL Code Generator - /home/jhb/workspace_v8/servoPWM\servoPWM.hcg - [ETPWM]
File Edit View Tools Window Help
SPI5 SPI1 SPI2 SPI3 SPI4 SPI5 CAN1 CAN2 CAN3 CAN4 ADC1 ADC2 HET1 HET2 I2C1 I2C2 EMAC DCC RTP DMM EMIF POM CRC ETPWM ECAP
Device Explorer
TMS57
General ETPWM1 ETPWM2 ETPWM3 ETPWM4 ETPWM5 ETPWM6 ETPWM7
Clock Configuration
TB Clock (MHz): 110.000
VCLK3 (MHz): 10.000
Clock Prescale
Actual TB Clock (MHz): 10.000
HSPCLKDIV: 0
CLKDIV: 0
PWM Configuration
tPeriod
tDuty
PWM
Low Polarity:
Duty[%]: 5
Period[ns]: 20000000
1000000.000
20000000.000
Delay[ns]: 1000.000
Enable delay
ETPWMxA
Rising Edge Delay
Invert Polarity
Invert Polarity
Delay[ns]: 1000.000
Delay
High Polarity:
PWM
Duty[%]: 5
Period[ns]: 20000000
1000000.000
20000000.000
Disable delay
ETPWMxB
Chopper Modulation
Enable Chopper Module



1. 처음에 프로그램 실행시 20ms 가나오지 않아 서보모터가 제대로 동작 하지 않는 것을 확인했다.
2. 모터는 원래 20ms 에서 동작해야 하기 때문에 설정에 문제가 있다.
3. TMS570LC4357 탭에서 PLL1→ HCLK 1 → VCLK3 14 를 사용한다.
4. 바꿔야 할 부분은 ETPWM 탭에서 HSPCLKDIV 값을 10 으로 설정해준다.
5. TB CLOCK 이 1M 가 나오게 해주어야 한다. (이유: TBPRD count 값이 2^16bit 까지만 표현 한다.)
   그래서 HSPCLKDIV 값으로 한번더 분주를 해주어 주파수를 낮추어 count 값이 작아지게 하는 것이다.
6. etpwmREG2->TBPRD = 19999U; 가 되어 있는지 확이 해야 한다. 약 20000 이라는 숫자가 나온것.
7. 이유: 1M clock 에서 몇번의 카운트가 되어야 20m/s 가 나오는가를 계산한다.
   T = 1/f 인데 여기서 주기는 1M 에서 x 번의 카운트 한 만큼이 되기때문.
   (20m/s = x/1M) 가 된다. (x = 20000) 이 나온다.