Using the ePWM Module for 0% - 100% Duty Cycle Control

Application Report

Literature Number: SPRAAI1

December 2006

TEXAS INSTRUMENTS

Contents

List of Figures

Using the Enhanced Pulse Width Modulator (ePWM) Module for 0% to 100% Duty Cycle Control

Hrishikesh Nene

ABSTRACT

This document provides a guide for the use of the ePWM module to provide 0% to 100% duty cycle control and is applicable to the TMS320x280x family of processors.

1 Introduction

Achieving a full 0% to 100% duty cycle control can become critical in certain applications. The flexibility and the resources provided by the TMS320x280x family of processors enable system control and applications engineers to achieve such duty cycle control over the entire range.

The ePWM modules can provide 0% to 100% duty cycles with minimal overheads. These modules can operate in three modes; up-count mode, up-down count mode and down-count mode. This document focuses on the use of the first two modes. This document assumes user familiarity with the *TMS320x280xx, 28xxx Enhanced Pulse Width Modulator (ePWM) Reference Guide* (SPRU791).

2 Configuring the PWM Module

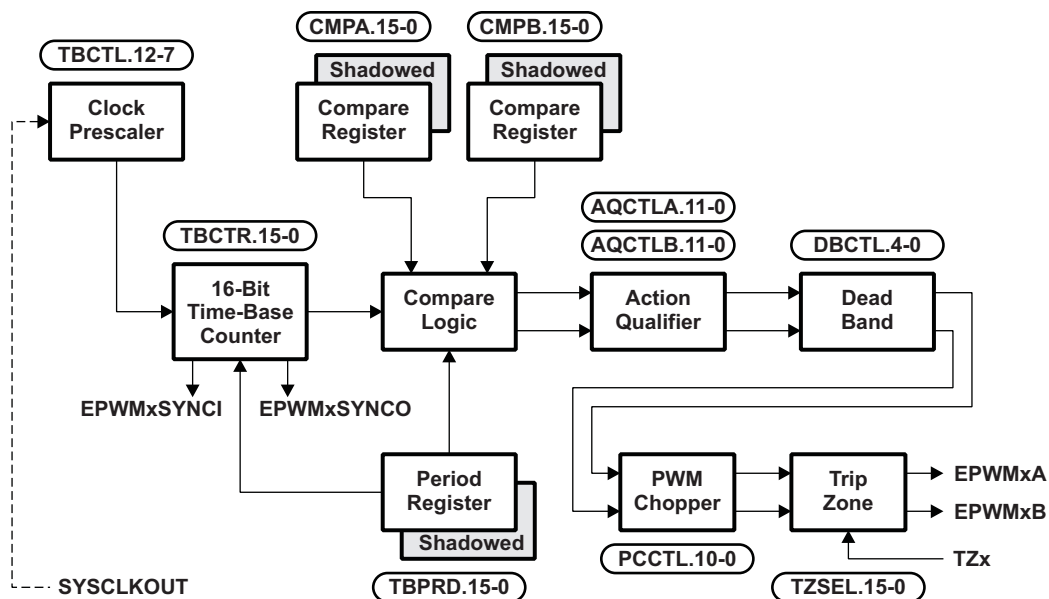Figure 1 depicts the PWM module block diagram.

Figure 1. EPWM Block Diagram

The ePWM module consists of the following submodules:
- Time-Base (TB) Submodule

- Counter-Compare (CC) Submodule
- Action-Qualifier (AQ) Submodule
- Dead-Band Generator (DB) Submodule
- PWM-Chopper (PC) Submodule
- Trip-Zone (TZ) Submodule
- Event-Trigger (ET) Submodule

Configuring an ePWM module requires initializing the registers found in these submodules. The control registers need to be configured properly to operate the ePWM module in one of the three modes mentioned in the previous section.

Configuring and using the ePWM module is simple for cases in which a 0% or a 100% duty cycle is not required. This can be achieved by following the procedures mentioned in the *TMS320x28xx, 28xxx Enhanced Pulse Width Modulator (ePWM) Reference Guide* ([SPRU791](link)). However, applications requiring 0% and/or 100% duty cycles are special cases and an additional procedure needs to be followed. This procedure is listed in the next section.

3 Achieving Full Range of Duty Ratios

Achieving a full range of duty ratios is possible by implementation of additional software that keeps track of the current and the next duty cycle values and makes use of the flexible configurability provided by the action qualifier submodule. The additional code is placed in the PWM interrupt service routine (ISR), and the register changes for the next cycle, if required, are made in the current ISR. The implementation of this software routine for the up-down count and up-count modes is explained next.

3.1 Up-Down Count Mode

The following is true for the up-down count mode (symmetric case) where a CMPA match on the up-count *sets* ePWMxA and a CMPA match on a down-count *resets* it. The ISR is called on a *zero* match and *shadow* loading is used.

In this case, a 100% duty cycle on ePWMxA is implemented by loading the CMPA register with a value of 0 and a 0% duty cycle is implemented by loading the CMPA register with a value $\geq$ Period.

The following explains the code implementation.

The compare register values for the next PWM cycle are calculated in the ISR of the current cycle. Therefore, the duty cycle values for the current and the next cycle are known in the present ISR. The knowledge of the next duty cycle can help in the current cycle as explained below.

When going from a non-zero CMPA value to a zero CMPA value:

- Change the action qualifier control register, AQCTLA.bit.ZRO = AQ_SET.
- This *sets* the PWM pin immediately in the next cycle.
- In the ISR for the next cycle (this is actually the first 100% cycle), change the action qualifier register back to its original value.

When coming out from a zero CMPA value to a non-zero CMPA value:

- Change the action qualifier control register to, AQCTLA.bit.ZRO = AQ_CLEAR, AQCTLA.bit.CAD = AQ_NO_ACTION.
- Change the LOADAMODE to load on zero or period.
- Note that the AQCTLA.bit.CAU = AQ_SET has not been altered. Therefore, for the cycle following the last 'CMPA=0' cycle, the ePWM pin sets high at a CMPA match even if the CMPA value is equal to 1 for this cycle.
- Change the action qualifier and control registers back to their original values at the start of the ISR for the next cycle. (This is actually the first non 0% cycle or the first non-zero CMPA cycle following the zero CMPA cycle)

No such action is needed when the CMPA value goes to *period* and comes out of it.

Note: *EPwm1Regs.ETPS.bit.INTPRD* should be initialized to *ET_1ST*, i.e., interrupt generated every event.

3.2 Up-Count Mode

The following is true for the up-count mode where a ZRO match *sets* the ePWMxA and a CMPA match *resets* it. The ISR is called on a *zero* match and *shadow* loading is used.

In this case, a 100% duty cycle on the ePWMxA is implemented by loading the CMPA register with a value > Period and a 0% duty cycle is implemented by loading the CMPA register with a value of 0.

The following explains the code implementation.

In this case, the compare register values for the next PWM cycle are calculated in the ISR of the current cycle. Therefore, the duty cycle values for the current and the next cycle are known in the present ISR. The knowledge of the next duty cycle can help in the current cycle as explained below.

When going from a non-zero CMPA value to a zero CMPA value:

- Change the action qualifier control register, AQCTLA.bit.ZRO = AQ_CLEAR.
- This *clears* the PWM pin immediately in the next cycle.
- In the ISR for the next cycle (this is actually the first 0% cycle), change the action qualifier register back to its original value.

When coming out from a zero CMPA value to a non-zero CMPA value:

- Change the action qualifier control register to, AQCTLA.bit.ZRO = AQ_SET.
- Change the LOADAMODE to load on zero or period.
- Note that the AQCTLA.bit.CAU = AQ_CLEAR has not been altered. Therefore, for the cycle following the last *CMPA=0* cycle, the ePWM pin gets cleared at a CMPA match even if CMPA value is equal to 1 for this cycle.
- Change the action qualifier and control registers back to their original values at the start of the ISR for the next cycle. (This is actually the first non 0% cycle or the first non-zero CMPA cycle following the zero CMPA cycle)

No such action is needed when the CMPA value goes to *Period* and comes out of it.

Note: *EPwm1Regs.ETPS.bit.INTPRD* should be initialized to *ET_1ST*, i.e., interrupt generated every event.

Note: The PWM time base sub-module should be configured such that the ISR code always gets executed within half the PWM period for the up-down count mode and within a PWM period for the up-count mode.

4 Software Flowchart

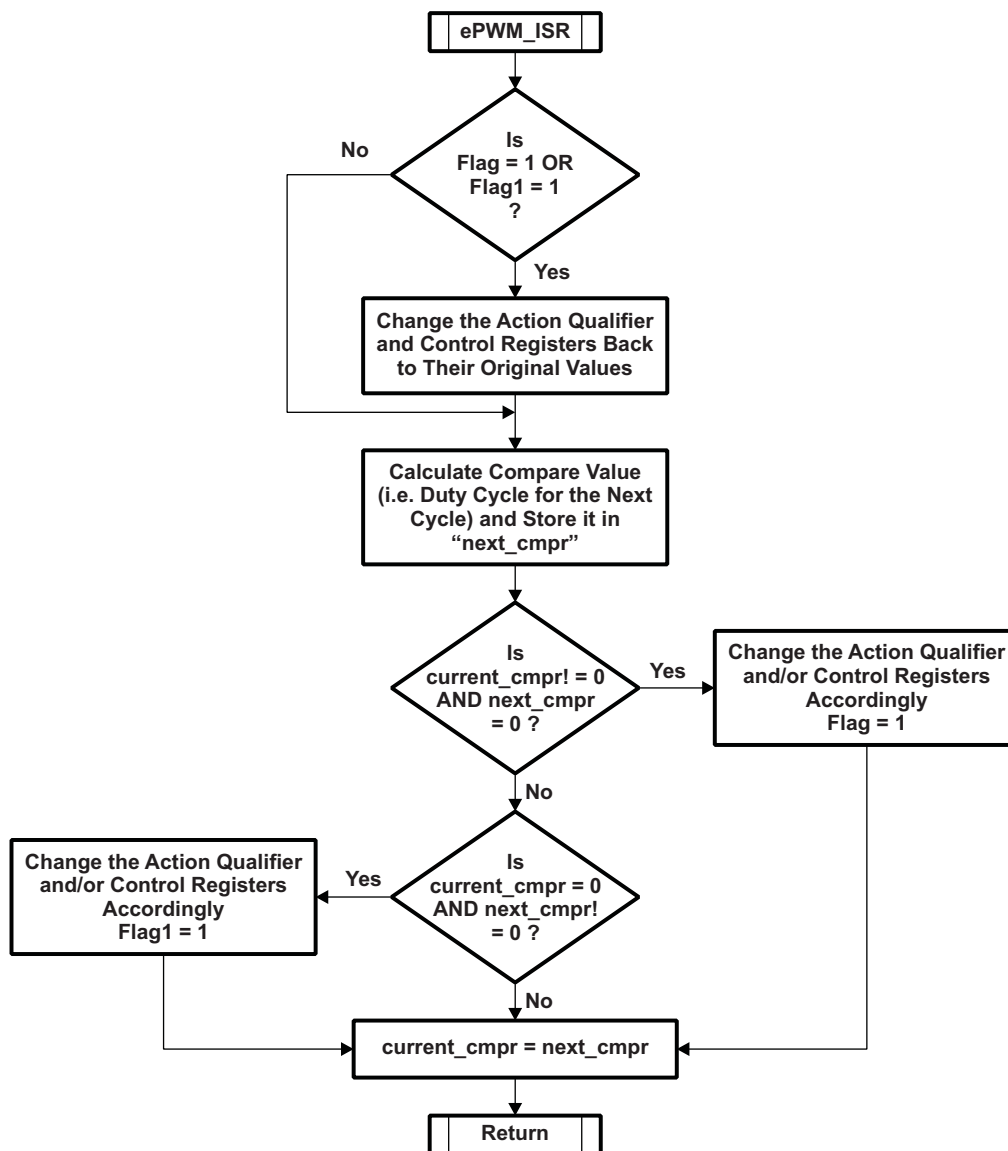A software flowchart for the code implementation is shown in Figure 2.

Figure 2. Software Flowchart

5   Sample Code

The following is a sample ISR code that can be used for implementing the ePWM module to generate duty cycles with 0% to 100% variation in the up-down count mode. This code provides independent control for the ePWM1A and ePWM1B and also makes the latter complementary to the ePWM1A.

```
void update_compare1(EPWM_INFO *epwm_info)
{

if (flag_outta_0 == 1 || flag_into_0 == 1)
        {
                EPwm1Regs.AQCTLA.bit.CAU = AQ_SET;      // Set PWM1A on event A, up count
                EPwm1Regs.AQCTLA.bit.CAD = AQ_CLEAR;
                EPwm1Regs.CMPCTL.bit.LOADAMODE = 0;     // Load on Zero
                flag_outta_0 = 0;
                flag_into_0 = 0;
        }

if (flag_outta_0_b == 1 || flag_into_0_b == 1)
        {
                EPwm1Regs.AQCTLB.bit.CBU = AQ_CLEAR;    // Set PWM1A on event A, up count
                EPwm1Regs.AQCTLB.bit.CBD = AQ_SET;
                EPwm1Regs.CMPCTL.bit.LOADBMODE = 0;     // Load on Zero
                flag_outta_0_b = 0;
                flag_into_0_b = 0;
        }

    // Every 6'th interrupt, change the CMPA/CMPB values

if(epwm_info->EPwmTimerIntCount == 5)
    {
        epwm_info->EPwmTimerIntCount = 0;

        // If we were increasing CMPA, check to see if
        // we reached the max value.  If not, increase CMPA
        // else, change directions and decrease CMPA
            if(epwm_info->EPwm_CMPA_Direction == EPWM_CMP_UP)
            {
                if(epwm_info->EPwmRegHandle->CMPA.half.CMPA < epwm_info->EPwmMaxCMPA)
                {
                    epwm_info->EPwmRegHandle->CMPA.half.CMPA = epwm_info->EPwmRegHandle
->CMPA.half.CMPA + Steps;                          //Steps = programmable duty cycle
                }                                          //step change defined at the start.
                else
                {
                    epwm_info->EPwm_CMPA_Direction = EPWM_CMP_DOWN;
                    epwm_info->EPwmRegHandle->CMPA.half.CMPA= epwm_info->EPwmRegHandle
->CMPA.half.CMPA - Steps;
                }
            }

        // If we were decreasing CMPA, check to see if
        // we reached the min value.  If not, decrease CMPA
        // else, change directions and increase CMPA

         else
         {
                if(epwm_info->EPwmRegHandle->CMPA.half.CMPA == epwm_info->EPwmMinCMPA)
                {
                    epwm_info->EPwm_CMPA_Direction = EPWM_CMP_UP;
                    epwm_info->EPwmRegHandle->CMPA.half.CMPA= epwm_info->EPwmRegHandle
->CMPA.half.CMPA + Steps;
                }
                else
                {
                    epwm_info->EPwmRegHandle->CMPA.half.CMPA= epwm_info->EPwmRegHandle
->CMPA.half.CMPA - Steps;
```

```
            }
        }

            //Coming out of CMPA = 0

if (temp == 0 && epwm_info->EPwmRegHandle->CMPA.half.CMPA != 0)
    {                                                       //temp=previous/current CMP value
        EPwm1Regs.AQCTLA.bit.ZRO = AQ_CLEAR;                // Set PWM1A on event A, up count
        EPwm1Regs.AQCTLA.bit.CAD = AQ_NO_ACTION;
        EPwm1Regs.CMPCTL.bit.LOADAMODE = 2;
        flag_outta_0 = 1;
    }

            //Going from CMPA != 0 to CMPA = 0

if (temp != 0 && epwm_info->EPwmRegHandle->CMPA.half.CMPA == 0)
    {
        EPwm1Regs.AQCTLA.bit.ZRO = AQ_SET;                  // Set PWM1A on event A, up count
        flag_into_0 = 1;
    }

temp = epwm_info->EPwmRegHandle->CMPA.half.CMPA;

        // If we were increasing CMPB, check to see if
        // we reached the max value.  If not, increase CMPB
        // else, change directions and decrease CMPB

        if(epwm_info->EPwm_CMPB_Direction == EPWM_CMP_UP)
        {
            if(epwm_info->EPwmRegHandle->CMPB < epwm_info->EPwmMaxCMPB)
            {
                epwm_info->EPwmRegHandle->CMPB = epwm_info->EPwmRegHandle->CMPB + Steps;
            }
            else
            {
                epwm_info->EPwm_CMPB_Direction = EPWM_CMP_DOWN;
                epwm_info->EPwmRegHandle->CMPB = epwm_info->EPwmRegHandle->CMPB - Steps;
            }
        }

        // If we were decreasing CMPB, check to see if
        // we reached the min value.  If not, decrease CMPB
        // else, change directions and increase CMPB

        else
        {
            if(epwm_info->EPwmRegHandle->CMPB == epwm_info->EPwmMinCMPB)
            {
            epwm_info->EPwm_CMPB_Direction = EPWM_CMP_UP;
            epwm_info->EPwmRegHandle->CMPB = epwm_info->EPwmRegHandle->CMPB + Steps;
        }
        else
        {
            epwm_info->EPwmRegHandle->CMPB = epwm_info->EPwmRegHandle->CMPB - Steps;
        }
        }

if (temp1 == 0 && epwm_info->EPwmRegHandle->CMPB != 0)
    {
        EPwm1Regs.AQCTLB.bit.ZRO = AQ_SET;                  // Set PWM1A on event A, up count
        EPwm1Regs.AQCTLB.bit.CBD = AQ_NO_ACTION;
        EPwm1Regs.CMPCTL.bit.LOADBMODE = 2;
        flag_outta_0_b = 1;
    }

if (temp1 != 0 && epwm_info->EPwmRegHandle->CMPB == 0)
```

```
        {
            EPwm1Regs.AQCTLB.bit.ZRO = AQ_CLEAR;                // Set PWM1A on event A, up count
            flag_into_0_b = 1;
        }

    temp1 = epwm_info->EPwmRegHandle->CMPB;

        }
        else
        {
            epwm_info->EPwmTimerIntCount++;
        }

        return;
    }
```

6   References

- *TMS320x28xx, 28xxx Enhanced Pulse Width Modulator (ePWM) Reference Guide* ([SPRU791](#)).

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

Products		**Applications**	
Amplifiers	amplifier.ti.com	Audio	www.ti.com/audio
Data Converters	dataconverter.ti.com	Automotive	www.ti.com/automotive
DSP	dsp.ti.com	Broadband	www.ti.com/broadband
Interface	interface.ti.com	Digital Control	www.ti.com/digitalcontrol
Logic	logic.ti.com	Military	www.ti.com/military
Power Mgmt	power.ti.com	Optical Networking	www.ti.com/opticalnetwork
Microcontrollers	microcontroller.ti.com	Security	www.ti.com/security
Low Power Wireless	www.ti.com/lpw	Telephony	www.ti.com/telephony
		Video & Imaging	www.ti.com/video
		Wireless	www.ti.com/wireless

Mailing Address: Texas Instruments
Post Office Box 655303 Dallas, Texas 75265