TMS320x28xx, 28xxx Enhanced Pulse Width
Modulator (ePWM) Module

Reference Guide

TEXAS
INSTRUMENTS

Contents

List of Figures

List of Tables

Read This First

This guide describes the Enhanced Pulse Width Modulator (ePWM) Module. It includes an overview of the module and information about each of the sub-modules:

- Time-Base Module
- Counter Compare Module
- Action Qualifier Module
- Dead-Band Generator Module
- PWM Chopper (PC) Module
- Trip Zone Module
- Event Trigger Module

Related Documentation From Texas Instruments

The following books describe the TMS320x280x and related support tools that are available on the TI website:

Data Manuals—

SPRS230— TMS320F2809, F2808, F2806, F2802, F2801, C2802, C2801, and F2801x DSPs Data Manual contains the pinout, signal descriptions, as well as electrical and timing specifications for the F280x devices.

SPRS357— TMS320F28044 Digital Signal Processor Data Manual contains the pinout, signal descriptions, as well as electrical and timing specifications for the F28044 device.

CPU User's Guides—

SPRU430— TMS320C28x DSP CPU and Instruction Set Reference Guide describes the central processing unit (CPU) and the assembly language instructions of the TMS320C28x fixed-point digital signal processors (DSPs). It also describes emulation features available on these DSPs.

SPRU712— TMS320x280x, 2801x, 2804x System Control and Interrupts Reference Guide describes the various interrupts and system control features of the 280x digital signal processors (DSPs).

Peripheral Guides—

SPRU566— TMS320x28xx, 28xxx Peripheral Reference Guide describes the peripheral reference guides of the 28x digital signal processors (DSPs).

SPRU716— TMS320x280x, 2801x, 2804x Analog-to-Digital Converter (ADC) Reference Guide describes how to configure and use the on-chip ADC module, which is a 12-bit pipelined ADC.

SPRU791— TMS320x28xx, 28xxx Enhanced Pulse Width Modulator (ePWM) Module Reference Guide describes the main areas of the enhanced pulse width modulator that include digital motor control, switch mode power supply control, UPS (uninterruptible power supplies), and other forms of power conversion

SPRU924— TMS320x28xx, 28xxx High-Resolution Pulse Width Modulator (HRPWM) describes the operation of the high-resolution extension to the pulse width modulator (HRPWM)

SPRU807— TMS320x28xx, 28xxx Enhanced Capture (eCAP) Module Reference Guide describes the enhanced capture module. It includes the module description and registers.

SPRU790— [TMS320x28xx, 28xxx Enhanced Quadrature Encoder Pulse (eQEP) Reference Guide](#) describes the eQEP module, which is used for interfacing with a linear or rotary incremental encoder to get position, direction, and speed information from a rotating machine in high performance motion and position control systems. It includes the module description and registers

SPRU074— [TMS320x28xx, 28xxx Enhanced Controller Area Network (eCAN) Reference Guide](#) describes the eCAN that uses established protocol to communicate serially with other controllers in electrically noisy environments.

SPRU051— [TMS320x28xx, 28xxx Serial Communication Interface (SCI) Reference Guide](#) describes the SCI, which is a two-wire asynchronous serial port, commonly known as a UART. The SCI modules support digital communications between the CPU and other asynchronous peripherals that use the standard non-return-to-zero (NRZ) format.

SPRU059— [TMS320x28xx, 28xxx Serial Peripheral Interface (SPI) Reference Guide](#) describes the SPI - a high-speed synchronous serial input/output (I/O) port - that allows a serial bit stream of programmed length (one to sixteen bits) to be shifted into and out of the device at a programmed bit-transfer rate.

SPRU721— [TMS320x28xx, 28xxx Inter-Integrated Circuit (I2C) Reference Guide](#) describes the features and operation of the inter-integrated circuit (I2C) module that is available on the TMS320x280x digital signal processor (DSP).

SPRU722— [TMS320x280x, 2801x, 2804x Boot ROM Reference Guide](#) describes the purpose and features of the bootloader (factory-programmed boot-loading software). It also describes other contents of the device on-chip boot ROM and identifies where all of the information is located within that memory.

Tools Guides—

SPRU513— [TMS320C28x Assembly Language Tools User's Guide](#) describes the assembly language tools (assembler and other tools used to develop assembly language code), assembler directives, macros, common object file format, and symbolic debugging directives for the TMS320C28x device.

SPRU514— [TMS320C28x Optimizing C Compiler User's Guide](#) describes the TMS320C28x™ C/C++ compiler. This compiler accepts ANSI standard C/C++ source code and produces TMS320 DSP assembly language source code for the TMS320C28x device.

SPRU608— [The TMS320C28x Instruction Set Simulator Technical Overview](#) describes the simulator, available within the Code Composer Studio for TMS320C2000 IDE, that simulates the instruction set of the C28x™ core.

SPRU625— [TMS320C28x DSP/BIOS Application Programming Interface (API) Reference Guide](#) describes development using DSP/BIOS.

Application Reports—

SPRAAM0— [Getting Started With TMS320C28x™ Digital Signal Controllers](#) is organized by development flow and functional areas to make your design effort as seamless as possible. Tips on getting started with C28x™ DSP software and hardware development are provided to aid in your initial design and debug efforts. Each section includes pointers to valuable information including technical documentation, software, and tools for use in each phase of design.

SPRAAD5— [Power Line Communication for Lighting Apps using BPSK w/ a Single DSP Controller](#) presents a complete implementation of a power line modem following CEA-709 protocol using a single DSP.

SPRAA85— [Programming TMS320x28xx and 28xxx Peripherals in C/C++](#) explores a hardware abstraction layer implementation to make C/C++ coding easier on 28x DSPs. This method is compared to traditional #define macros and topics of code efficiency and special case registers are also addressed.

SPRA958— [Running an Application from Internal Flash Memory on the TMS320F28xx DSP](#) covers the requirements needed to properly configure application software for execution from on-chip flash memory. Requirements for both DSP/BIOS™ and non-DSP/BIOS projects are presented. Example code projects are included.

SPRAA91— [TMS320F280x DSC USB Connectivity Using TUSB3410 USB-to-UART Bridge Chip](#) presents hardware connections as well as software preparation and operation of the development system using a simple communication echo program.

SPRAA58— [TMS320x281x to TMS320x280x Migration Overview](#) describes differences between the Texas Instruments TMS320x281x and TMS320x280x DSPs to assist in application migration from the 281x to the 280x. While the main focus of this document is migration from 281x to 280x, users considering migrating in the reverse direction (280x to 281x) will also find this document useful.

SPRAAD8— [TMS320280x and TMS320F2801x ADC Calibration](#) describes a method for improving the absolute accuracy of the 12-bit ADC found on the TMS320280x and TMS3202801x devices. Inherent gain and offset errors affect the absolute accuracy of the ADC. The methods described in this report can improve the absolute accuracy of the ADC to levels better than 0.5%. This application report has an option to download an example program that executes from RAM on the F2808 EzDSP.

SPRAAI1— [Using Enhanced Pulse Width Modulator (ePWM) Module for 0-100% Duty Cycle Control](#) 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.

SPRAA88— [Using PWM Output as a Digital-to-Analog Converter on a TMS320F280x](#) presents a method for utilizing the on-chip pulse width modulated (PWM) signal generators on the TMS320F280x family of digital signal controllers as a digital-to-analog converter (DAC).

SPRAAH1— [Using the Enhanced Quadrature Encoder Pulse (eQEP) Module](#) provides a guide for the use of the eQEP module as a dedicated capture unit and is applicable to the TMS320x280x, 28xxx family of processors.

SPRA820— [Online Stack Overflow Detection on the TMS320C28x DSP](#) presents the methodology for online stack overflow detection on the TMS320C28x™ DSP. C-source code is provided that contains functions for implementing the overflow detection on both DSP/BIOS™ and non-DSP/BIOS applications.

SPRA806— [An Easy Way of Creating a C-callable Assembly Function for the TMS320C28x DSP](#) provides instructions and suggestions to configure the C compiler to assist with understanding of parameter-passing conventions and environments expected by the C compiler.

Trademarks

TMS320C28x, C28x are trademarks of Texas Instruments.

Introduction

The enhanced pulse width modulator (ePWM) peripheral is a key element in controlling many of the power-related systems found in both commercial and industrial equipments. These systems include digital motor control, switch mode power supply control, uninterruptible power supplies (UPS), and other forms of power conversion. The ePWM peripheral performs a digital to analog (DAC) function, where the duty cycle is equivalent to a DAC analog value; it is sometimes referred to as a Power DAC.

This reference guide is applicable for the ePWM found on the TMS320x280x, TMS320x2801x and TMS320x2804x processors. This includes all Flash-based, ROM-based, and RAM-based devices.

1.1 Introduction

An effective PWM peripheral must be able to generate complex pulse width waveforms with minimal CPU overhead or intervention. It needs to be highly programmable and very flexible while being easy to understand and use. The ePWM unit described here addresses these requirements by allocating all needed timing and control resources on a per PWM channel basis. Cross coupling or sharing of resources has been avoided; instead, the ePWM is built up from smaller single channel modules with separate resources and that can operate together as required to form a system. This modular approach results in an orthogonal architecture and provides a more transparent view of the peripheral structure, helping users to understand its operation quickly.

In this document the letter x within a signal or module name is used to indicate a generic ePWM instance on a device. For example output signals EPWMxA and EPWMxB refer to the output signals from the ePWMx instance. Thus, EPWM1A and EPWM1B belong to ePWM1 and likewise EPWM4A and EPWM4B belong to ePWM4.

1.2 Submodule Overview

The ePWM module represents one complete PWM channel composed of two PWM outputs: EPWMxA and EPWMxB. Multiple ePWM modules are instanced within a device as shown in Figure 1-1. Each ePWM instance is identical with one exception. Some instances include a hardware extension that allows more precise control of the PWM outputs. This extension is the high-resolution pulse width modulator (HRPWM) and is described in the *TMS320x28xx, 28xxx High-Resolution Pulse Width Modulator (HRPWM) Reference Guide* (SPRU924). See the device-specific data manual to determine which ePWM instances include this feature. Each ePWM module is indicated by a numerical value starting with 1. For example ePWM1 is the first instance and ePWM3 is the 3rd instance in the system and ePWMx indicates any instance.

The ePWM modules are chained together via a clock synchronization scheme that allows them to operate as a single system when required. Additionally, this synchronization scheme can be extended to the capture peripheral modules (eCAP). The number of modules is device-dependent and based on target application needs. Modules can also operate stand-alone.

Each ePWM module supports the following features:
- Dedicated 16-bit time-base counter with period and frequency control
- Two PWM outputs (EPWMxA and EPWMxB) that can be used in the following configurations::
  - Two independent PWM outputs with single-edge operation
  - Two independent PWM outputs with dual-edge symmetric operation
  - One independent PWM output with dual-edge asymmetric operation
- Asynchronous override control of PWM signals through software.
- Programmable phase-control support for lag or lead operation relative to other ePWM modules.
- Hardware-locked (synchronized) phase relationship on a cycle-by-cycle basis.
- Dead-band generation with independent rising and falling edge delay control.
- Programmable trip zone allocation of both cycle-by-cycle trip and one-shot trip on fault conditions.
- A trip condition can force either high, low, or high-impedance state logic levels at PWM outputs.
- All events can trigger both CPU interrupts and ADC start of conversion (SOC)
- Programmable event prescaling minimizes CPU overhead on interrupts.
- PWM chopping by high-frequency carrier signal, useful for pulse transformer gate drives.

Each ePWM module is connected to the input/output signals shown in Figure 1-1. The signals are described in detail in subsequent sections.

Figure 1-1. Multiple ePWM Modules
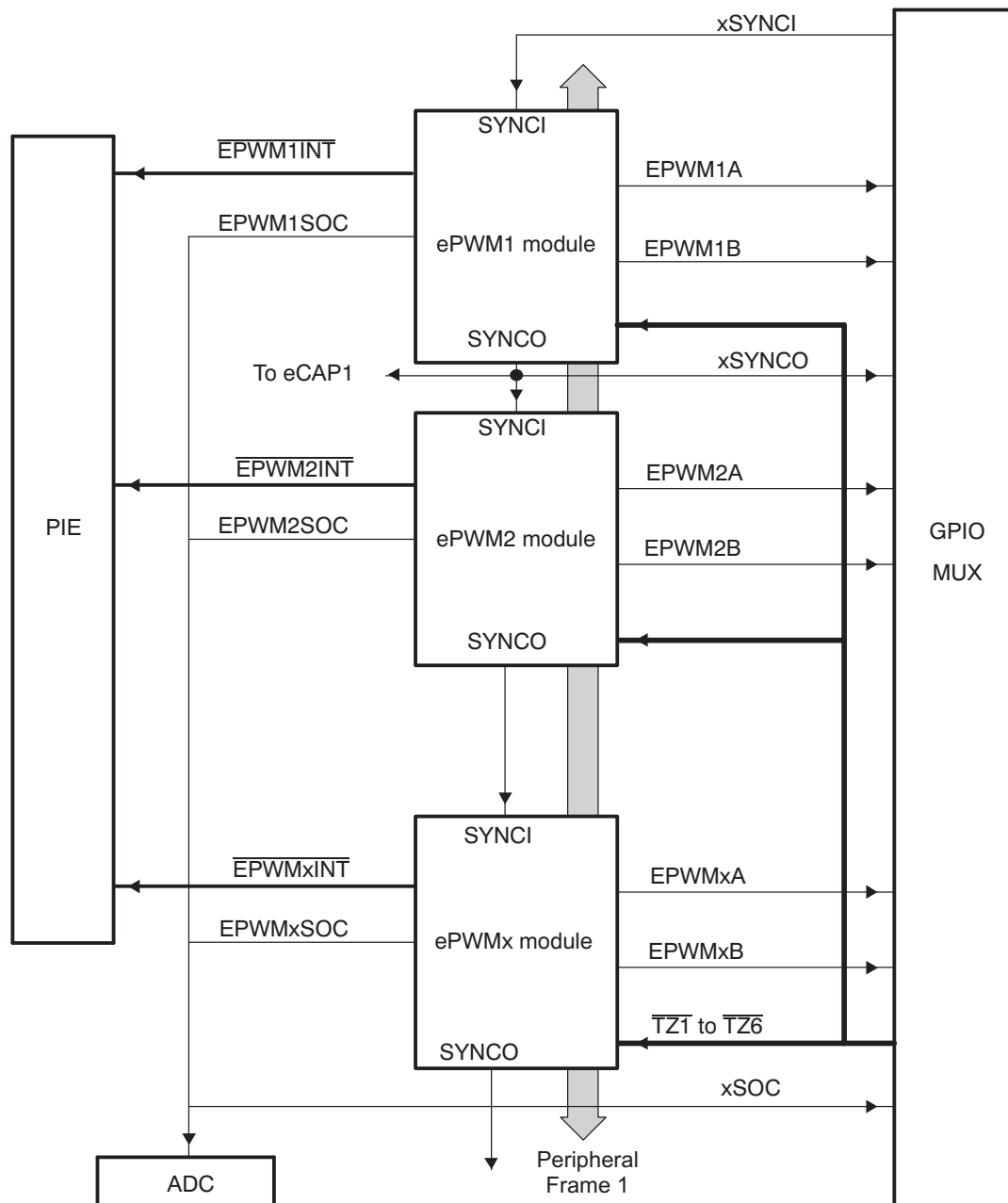
The order in which the ePWM modules are connected may differ from what is shown in Figure 1-1. See Section 2.2.3.2 for the synchronization scheme for a particular device. Each ePWM module consists of seven submodules and is connected within a system via the signals shown in Figure 1-2.

Figure 1-2. Submodules and Signal Connections for an ePWM Module
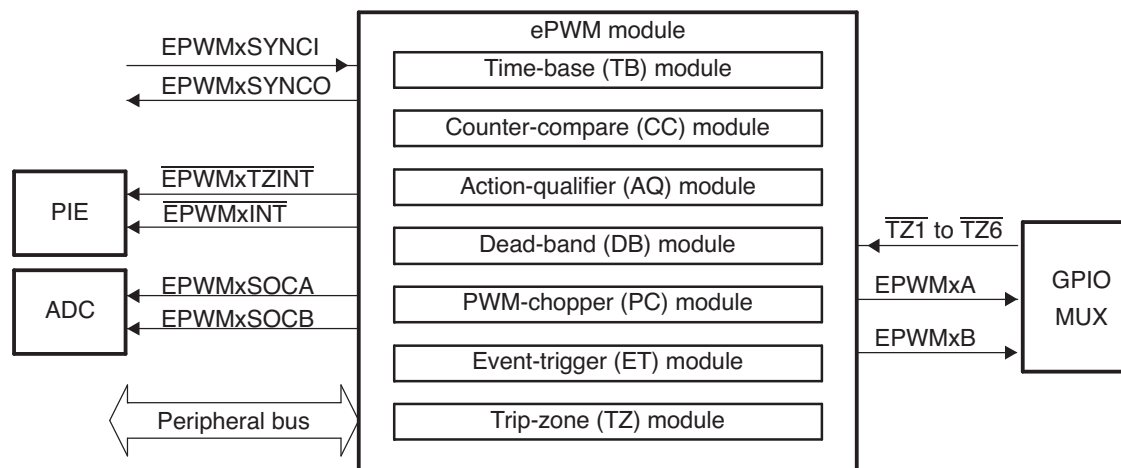
Figure 1-3 shows more internal details of a single ePWM module. The main signals used by the ePWM module are:

- **PWM output signals (EPWMxA and EPWMxB).**

  The PWM output signals are made available external to the device through the GPIO peripheral described in the system control and interrupts guide for your device.

- **Trip-zone signals ($\overline{TZ1}$ to $\overline{TZ6}$).**

  These input signals alert the ePWM module of an external fault condition. Each module on a device can be configured to either use or ignore any of the trip-zone signals. The trip-zone signals can be configured as asynchronous inputs through the GPIO peripheral.

- **Time-base synchronization input (EPWMxSYNCI) and output (EPWMxSYNCO) signals**.

  The synchronization signals daisy chain the ePWM modules together. Each module can be configured to either use or ignore its synchronization input. The clock synchronization input and output signal are brought out to pins only for ePWM1 (ePWM module #1). The synchronization output for ePWM1 (EPWM1SYNCO) is also connected to the SYNCI of the first enhanced capture module (eCAP1).

- **ADC start-of-conversion signals (EPWMxSOCA and EPWMxSOCB).**

  Each ePWM module has two ADC start of conversion signals (one for each sequencer). Any ePWM module can trigger a start of conversion for either sequencer. Which event triggers the start of conversion is configured in the Event-Trigger submodule of the ePWM.

- **Peripheral Bus**

  The peripheral bus is 32-bits wide and allows both 16-bit and 32-bit writes to the ePWM register file.

Figure 1-3. ePWM Submodules and Critical Internal Signal Interconnects
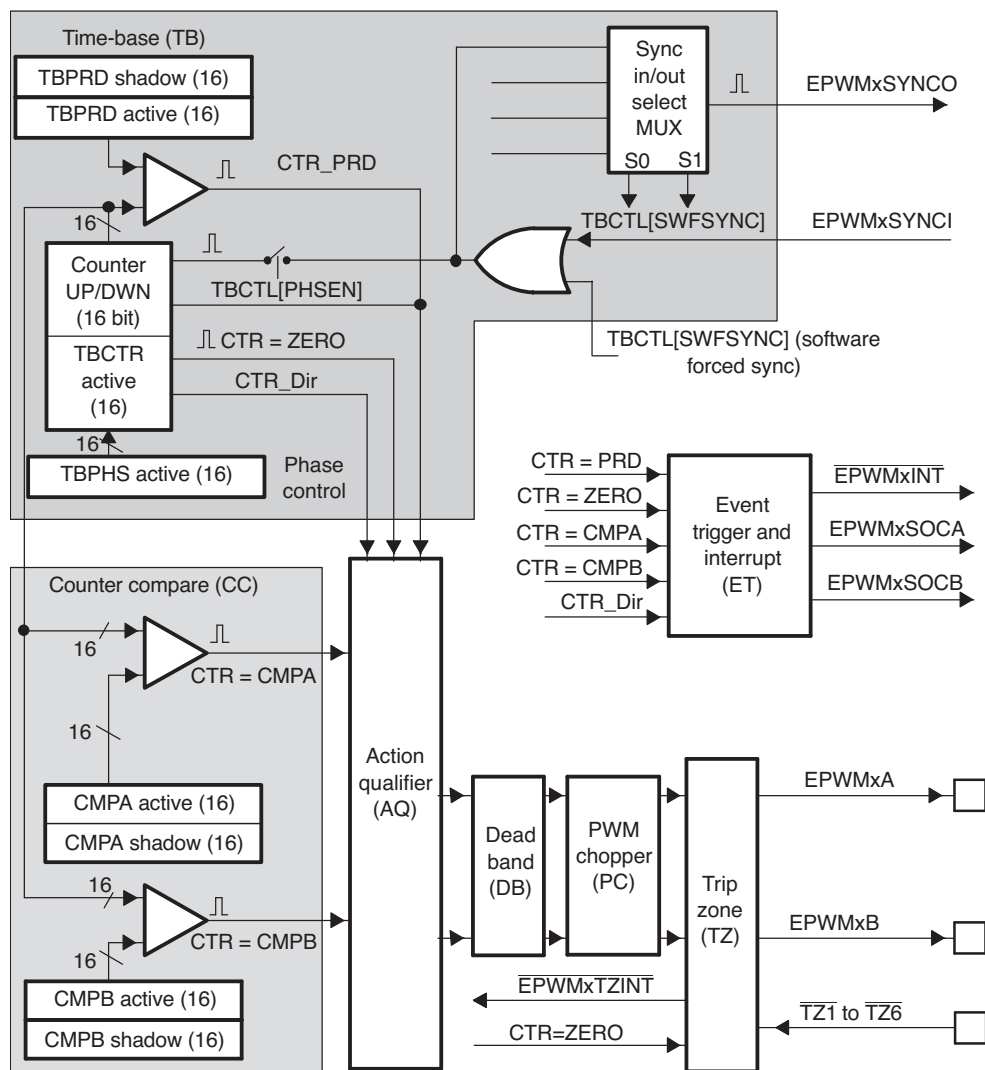
Figure 1-3 also shows the key internal submodule interconnect signals. Each submodule is described in detail in its respective section.

1.3   Register Mapping

The complete ePWM module control and status register set is grouped by submodule as shown in Table 1-1. Each register set is duplicated for each instance of the ePWM module. The start address for each ePWM register file instance on a device is specified in the appropriate data manual.

Table 1-1. ePWM Module Control and Status Register Set Grouped by Submodule

Name	Offset [1]	Size (x16)	Shadow	Description
				Time-Base Submodule Registers
TBCTL	0x0000	1	No	Time-Base Control Register
TBSTS	0x0001	1	No	Time-Base Status Register
TBPHSHR	0x0002	1	No	Extension for HRPWM Phase Register [2]
TBPHS	0x0003	1	No	Time-Base Phase Register
TBCTR	0x0004	1	No	Time-Base Counter Register
TBPRD	0x0005	1	Yes	Time-Base Period Register
				Counter-Compare Submodule Registers
CMPCTL	0x0007	1	No	Counter-Compare Control Register
CMPAHR	0x0008	1	No	Extension for HRPWM Counter-Compare A Register [2]
CMPA	0x0009	1	Yes	Counter-Compare A Register
CMPB	0x000A	1	Yes	Counter-Compare B Register
				Action-Qualifier Submodule Registers
AQCTLA	0x000B	1	No	Action-Qualifier Control Register for Output A (EPWMxA)
AQCTLB	0x000C	1	No	Action-Qualifier Control Register for Output B (EPWMxB)
AQSFRC	0x000D	1	No	Action-Qualifier Software Force Register
AQCSFRC	0x000E	1	Yes	Action-Qualifier Continuous S/W Force Register Set
				Dead-Band Generator Submodule Registers
DBCTL	0x000F	1	No	Dead-Band Generator Control Register
DBRED	0x0010	1	No	Dead-Band Generator Rising Edge Delay Count Register
DBFED	0x0011	1	No	Dead-Band Generator Falling Edge Delay Count Register
				Trip-Zone Submodule Registers
TZSEL	0x0012	1	No	Trip-Zone Select Register
TZCTL	0x0014	1	No	Trip-Zone Control Register [3]
TZEINT	0x0015	1	No	Trip-Zone Enable Interrupt Register [3]
TZFLG	0x0016	1	No	Trip-Zone Flag Register [3]
TZCLR	0x0017	1	No	Trip-Zone Clear Register [3]
TZFRC	0x0018	1	No	Trip-Zone Force Register [3]
				Event-Trigger Submodule Registers
ETSEL	0x0019	1	No	Event-Trigger Selection Register
ETPS	0x001A	1	No	Event-Trigger Pre-Scale Register
ETFLG	0x001B	1	No	Event-Trigger Flag Register
ETCLR	0x001C	1	No	Event-Trigger Clear Register
ETFRC	0x001D	1	No	Event-Trigger Force Register
				PWM-Chopper Submodule Registers
PCCTL	0x001E	1	No	PWM-Chopper Control Register
				High-Resolution Pulse Width Modulator (HRPWM) Extension Registers
HRCNFG	0x0020	1	No	HRPWM Configuration Register [2] [3]

[1] Locations not shown are reserved.
[2] These registers are only available on ePWM instances that include the high-resolution PWM extension. Otherwise these locations are reserved. These registers are described in the *TMS320x28xx, 28xxx High-Resolution Pulse Width Modulator (HRPWM) Reference Guide* (SPRU924). See the device specific data manual to determine which instances include the HRPWM.
[3] EALLOW protected registers as described in the specific device version of the *System Control and Interrupts Reference Guide* listed in Section 1.

ePWM Submodules

Seven submodules are included in every ePWM peripheral. Each of these submodules performs specific tasks that can be configured by software.

2.1 Overview

Table 2-1 lists the seven key submodules together with a list of their main configuration parameters. For example, if you need to adjust or control the duty cycle of a PWM waveform, then you should see the counter-compare submodule in Section 2.3 for relevant details.

Table 2-1. Submodule Configuration Parameters

Submodule	Configuration Parameter or Option

Time-base (TB)	• Scale the time-base clock (TBCLK) relative to the system clock (SYSCLKOUT).
	• Configure the PWM time-base counter (TBCTR) frequency or period.
	• Set the mode for the time-base counter:
	  – count-up mode: used for asymmetric PWM
	  – count-down mode: used for asymmetric PWM
	  – count-up-and-down mode: used for symmetric PWM
	• Configure the time-base phase relative to another ePWM module.
	• Synchronize the time-base counter between modules through hardware or software.
	• Configure the direction (up or down) of the time-base counter after a synchronization event.
	• Configure how the time-base counter will behave when the device is halted by an emulator.
	• Specify the source for the synchronization output of the ePWM module:
	  – Synchronization input signal
	  – Time-base counter equal to zero
	  – Time-base counter equal to counter-compare B (CMPB)
	  – No output synchronization signal generated.

Counter-compare (CC)	• Specify the PWM duty cycle for output EPWMxA and/or output EPWMxB • Specify the time at which switching events occur on the EPWMxA or EPWMxB output

Action-qualifier (AQ)	• Specify the type of action taken when a time-base or counter-compare submodule event occurs:
	  – No action taken
	  – Output EPWMxA and/or EPWMxB switched high
	  – Output EPWMxA and/or EPWMxB switched low
	  – Output EPWMxA and/or EPWMxB toggled
	• Force the PWM output state through software control
	• Configure and control the PWM dead-band through software

Dead-band (DB)	• Control of traditional complementary dead-band relationship between upper and lower switches • Specify the output rising-edge-delay value • Specify the output falling-edge delay value • Bypass the dead-band module entirely. In this case the PWM waveform is passed through without modification.
PWM-chopper (PC)	• Create a chopping (carrier) frequency. • Pulse width of the first pulse in the chopped pulse train. • Duty cycle of the second and subsequent pulses. • Bypass the PWM-chopper module entirely. In this case the PWM waveform is passed through without modification.

Trip-zone (TZ)	• Configure the ePWM module to react to one, all, or none of the trip-zone pins.
	• Specify the tripping action taken when a fault occurs:
	  – Force EPWMxA and/or EPWMxB high
	  – Force EPWMxA and/or EPWMxB low
	  – Force EPWMxA and/or EPWMxB to a high-impedance state
	  – Configure EPWMxA and/or EPWMxB to ignore any trip condition.
	• Configure how often the ePWM will react to each trip-zone pin:
	  – One-shot
	  – Cycle-by-cycle
	• Enable the trip-zone to initiate an interrupt.
	• Bypass the trip-zone module entirely.

Table 2-1. Submodule Configuration Parameters (continued)

Submodule	Configuration Parameter or Option
Event-trigger (ET)	• Enable the ePWM events that will trigger an interrupt. • Enable ePWM events that will trigger an ADC start-of-conversion event. • Specify the rate at which events cause triggers (every occurrence or every second or third occurrence) • Poll, set, or clear event flags

Code examples are provided in the remainder of this document that show how to implement various ePWM module configurations. These examples use the constant definitions shown in Example 2-1. These definitions are also used in the *C280x C/C++ Header Files and Peripheral Examples* (SPRC191).

Example 2-1. Constant Definitions Used in the Code Examples

```
// TBCTL (Time-Base Control)
// = = = = = = = = = = = = = = = = = = = = = = = = = = =
// TBCTR MODE bits
#define          TB_COUNT_UP        0x0
#define          TB_COUNT_DOWN      0x1
#define          TB_COUNT_UPDOWN    0x2
#define          TB_FREEZE          0x3
// PHSEN bit
#define          TB_DISABLE         0x0
#define          TB_ENABLE          0x1
// PRDLD bit
#define          TB_SHADOW          0x0
#define          TB_IMMEDIATE       0x1
// SYNCOSEL bits
#define          TB_SYNC_IN         0x0
#define          TB_CTR_ZERO        0x1
#define          TB_CTR_CMPB        0x2
#define          TB_SYNC_DISABLE    0x3
// HSPCLKDIV and CLKDIV bits
#define          TB_DIV1            0x0
#define          TB_DIV2            0x1
#define          TB_DIV4            0x2
// PHSDIR bit
#define          TB_DOWN            0x0
#define          TB_UP              0x1

// CMPCTL (Compare Control)
// = = = = = = = = = = = = = = = = = = = = = = = = = = =
// LOADAMODE and LOADBMODE bits
#define          CC_CTR_ZERO        0x0
#define          CC_CTR_PRD         0x1
#define          CC_CTR_ZERO_PRD    0x2
#define          CC_LD_DISABLE      0x3
// SHDWAMODE and SHDWBMODE bits
#define          CC_SHADOW          0x0
#define          CC_IMMEDIATE       0x1
// AQCTLA and AQCTLB (Action-qualifier Control)
// = = = = = = = = = = = = = = = = = = = = = = = = = = =
// ZRO, PRD, CAU, CAD, CBU, CBD bits
#define          AQ_NO_ACTION       0x0
#define          AQ_CLEAR           0x1
#define          AQ_SET             0x2
#define          AQ_TOGGLE          0x3
// DBCTL (Dead-Band Control)
// = = = = = = = = = = = = = = = = = = = = = = = = = = =
// MODE bits
#define          DB_DISABLE         0x0
#define          DBA_ENABLE         0x1
#define          DBB_ENABLE         0x2
#define DB_FULL_ENABLE 0x3
// POLSEL bits
#define          DB_ACTV_HI         0x0
#define          DB_ACTV_LOC        0x1
#define          DB_ACTV_HIC        0x2
```

Example 2-1. Constant Definitions Used in the Code Examples (continued)

```
#define          DB_ACTV_LO          0x3
// PCCTL (chopper control)
// = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
// CHPEN bit
#define          CHP_ENABLE          0x0
#define CHP_DISABLE 0x1
// CHPFREQ bits
#define          CHP_DIV1            0x0
#define          CHP_DIV2            0x1
#define          CHP_DIV3            0x2
#define          CHP_DIV4            0x3
#define          CHP_DIV5            0x4
#define          CHP_DIV6            0x5
#define          CHP_DIV7            0x6
#define          CHP_DIV8            0x7
// CHPDUTY bits
#define          CHP1_8TH            0x0
#define          CHP2_8TH            0x1
#define          CHP3_8TH            0x2
#define          CHP4_8TH            0x3
#define          CHP5_8TH            0x4
#define          CHP6_8TH            0x5
#define          CHP7_8TH            0x6
// TZSEL (Trip-zone Select)
// = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
// CBCn and OSHTn bits
#define          TZ_ENABLE           0x0
#define          TZ_DISABLE          0x1
// TZCTL (Trip-zone Control)
// = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
// TZA and TZB bits
#define          TZ_HIZ              0x0
#define          TZ_FORCE_HI         0x1
#define          TZ_FORCE_LO         0x2
#define          TZ_DISABLE          0x3
// ETSEL (Event-trigger Select)
// = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
// INTSEL, SOCASEL, SOCBSEL bits
#define          ET_CTR_ZERO         0x1
#define          ET_CTR_PRD          0x2
#define          ET_CTRU_CMPA        0x4
#define          ET_CTRD_CMPA        0x5
#define          ET_CTRU_CMPB        0x6
#define          ET_CTRD_CMPB        0x7
// ETPS (Event-trigger Prescale)
// = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
// INTPRD, SOCAPRD, SOCBPRD bits
#define          ET_DISABLE          0x0
#define          ET_1ST              0x1
#define          ET_2ND              0x2
#define          ET_3RD              0x3
```

2.2 Time-Base (TB) Submodule

Each ePWM module has its own time-base submodule that determines all of the event timing for the ePWM module. Built-in synchronization logic allows the time-base of multiple ePWM modules to work together as a single system. Figure 2-1 illustrates the time-base module's place within the ePWM.

Figure 2-1. Time-Base Submodule Block Diagram
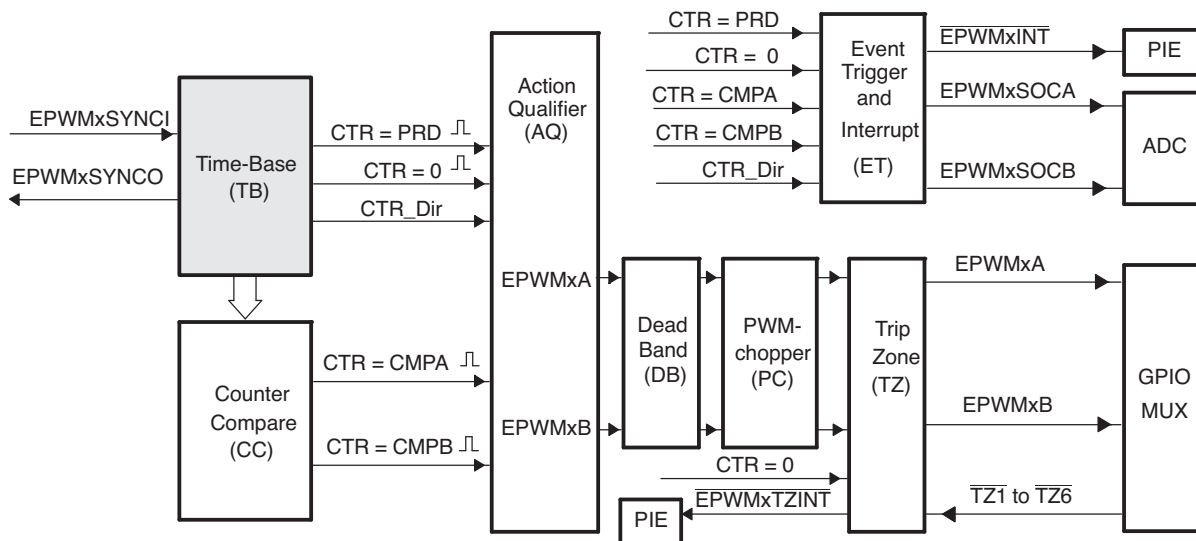
2.2.1 Purpose of the Time-Base Submodule

You can configure the time-base submodule for the following:

- Specify the ePWM time-base counter (TBCTR) frequency or period to control how often events occur.
- Manage time-base synchronization with other ePWM modules.
- Maintain a phase relationship with other ePWM modules.
- Set the time-base counter to count-up, count-down, or count-up-and-down mode.
- Generate the following events:
  - CTR = PRD: Time-base counter equal to the specified period (TBCTR = TBPRD) .
  - CTR = Zero: Time-base counter equal to zero (TBCTR = 0x0000).
- Configure the rate of the time-base clock; a prescaled version of the CPU system clock (SYSCLKOUT). This allows the time-base counter to increment/decrement at a slower rate.

2.2.2 *Controlling and Monitoring the Time-base Submodule*

Table 2-2 shows the registers used to control and monitor the time-base submodule.

Table 2-2. Time-Base Submodule Registers

Register	Address offset	Shadowed	Description
TBCTL	0x0000	No	Time-Base Control Register
TBSTS	0x0001	No	Time-Base Status Register
TBPHSHR	0x0002	No	HRPWM extension Phase Register [1]
TBPHS	0x0003	No	Time-Base Phase Register
TBCTR	0x0004	No	Time-Base Counter Register
TBPRD	0x0005	Yes	Time-Base Period Register

[1] This register is available only on ePWM instances that include the high-resolution extension (HRPWM). On ePWM modules that do not include the HRPWM, this location is reserved. This register is described in the *TMS320x28xx, 28xxx High-Resolution Pulse Width Modulator (HRPWM) Reference Guide* (SPRU924). See the device specific data manual to determine which ePWM instances include this feature.

The block diagram in Figure 2-2 shows the critical signals and registers of the time-base submodule. Table 2-3 provides descriptions of the key signals associated with the time-base submodule.

Figure 2-2. Time-Base Submodule Signals and Registers
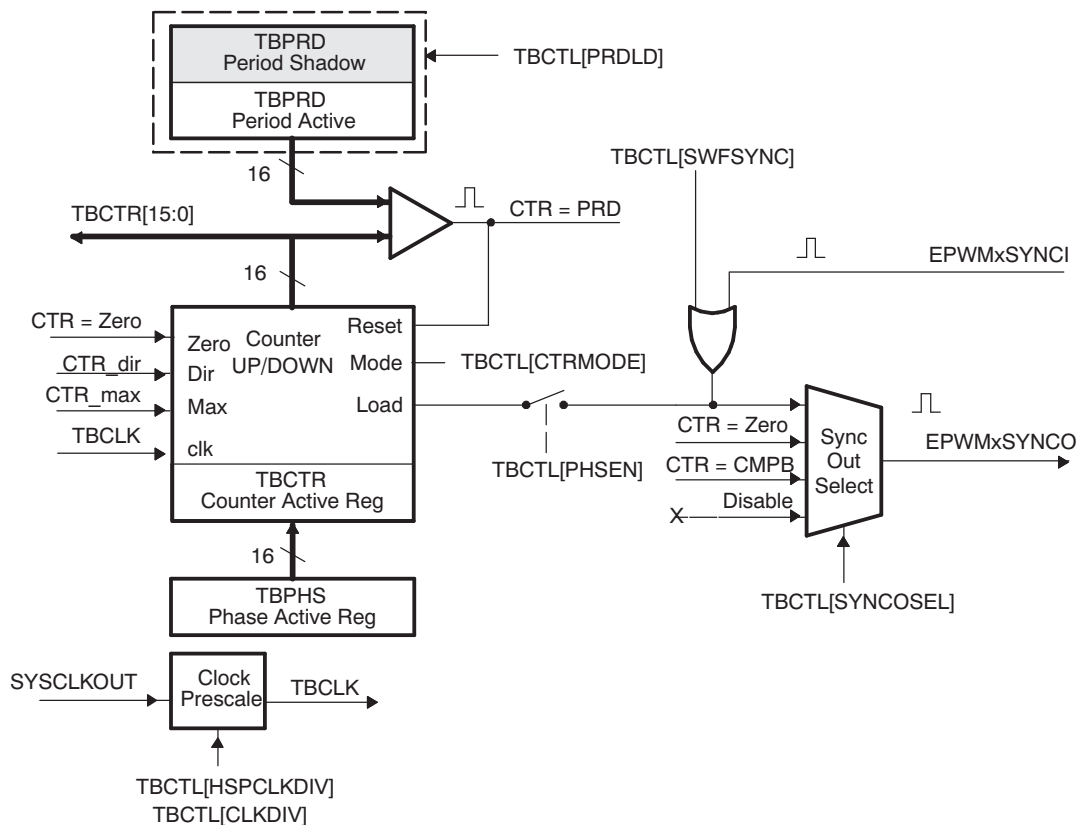
Table 2-3. Key Time-Base Signals

Signal	Description
EPWMxSYNCI	Time-base synchronization input.
	Input pulse used to synchronize the time-base counter with the counter of ePWM module earlier in the synchronization chain. An ePWM peripheral can be configured to use or ignore this signal. For the first ePWM module (EPWM1) this signal comes from a device pin. For subsequent ePWM modules this signal is passed from another ePWM peripheral. For example, EPWM2SYNCI is generated by the ePWM1 peripheral, EPWM3SYNCI is generated by ePWM2 and so forth. See Section 2.2.3.2 for information on the synchronization order of a particular device.
EPWMxSYNCO	Time-base synchronization output.
	This output pulse is used to synchronize the counter of an ePWM module later in the synchronization chain. The ePWM module generates this signal from one of three event sources:
	1. EPWMxSYNCI (Synchronization input pulse)
	2. CTR = Zero: The time-base counter equal to zero (TBCTR = 0x0000).
	3. CTR = CMPB: The time-base counter equal to the counter-compare B (TBCTR = CMPB) register.
CTR = PRD	Time-base counter equal to the specified period.
	This signal is generated whenever the counter value is equal to the active period register value. That is when TBCTR = TBPRD.
CTR = Zero	Time-base counter equal to zero
	This signal is generated whenever the counter value is zero. That is when TBCTR equals 0x0000.
CTR = CMPB	Time-base counter equal to active counter-compare B register (TBCTR = CMPB).
	This event is generated by the counter-compare submodule and used by the synchronization out logic
CTR_dir	Time-base counter direction.
	Indicates the current direction of the ePWM's time-base counter. This signal is high when the counter is increasing and low when it is decreasing.
CTR_max	Time-base counter equal max value. (TBCTR = 0xFFFF)
	Generated event when the TBCTR value reaches its maximum value. This signal is only used only as a status bit
TBCLK	Time-base clock.
	This is a prescaled version of the system clock (SYSCLKOUT) and is used by all submodules within the ePWM. This clock determines the rate at which time-base counter increments or decrements.

2.2.3 Calculating PWM Period and Frequency

The frequency of PWM events is controlled by the time-base period (TBPRD) register and the mode of the time-base counter. Figure 2-3 shows the period (T_{pwm}) and frequency (F_{pwm}) relationships for the up-count, down-count, and up-down-count time-base counter modes when when the period is set to 4 (TBPRD = 4). The time increment for each step is defined by the time-base clock (TBCLK) which is a prescaled version of the system clock (SYSCLKOUT).

The time-base counter has three modes of operation selected by the time-base control register (TBCTL):

- **Up-Down-Count Mode:**

    In up-down-count mode, the time-base counter starts from zero and increments until the period (TBPRD) value is reached. When the period value is reached, the time-base counter then decrements until it reaches zero. At this point the counter repeats the pattern and begins to increment.

- **Up-Count Mode:**

    In this mode, the time-base counter starts from zero and increments until it reaches the value in the period register (TBPRD). When the period value is reached, the time-base counter resets to zero and begins to increment once again.

- **Down-Count Mode:**

    In down-count mode, the time-base counter starts from the period (TBPRD) value and decrements until it reaches zero. When it reaches zero, the time-base counter is reset to the period value and it begins to decrement once again.

Figure 2-3. Time-Base Frequency and Period

For Up Count and Down Count

$$T_{PWM} = (TBPRD + 1) \times T_{TBCLK}$$
$$F_{PWM} = 1 / (T_{PWM})$$

For Up and Down Count

$$T_{PWM} = 2 \times TBPRD \times T_{TBCLK}$$
$$F_{PWM} = 1 / (T_{PWM})$$
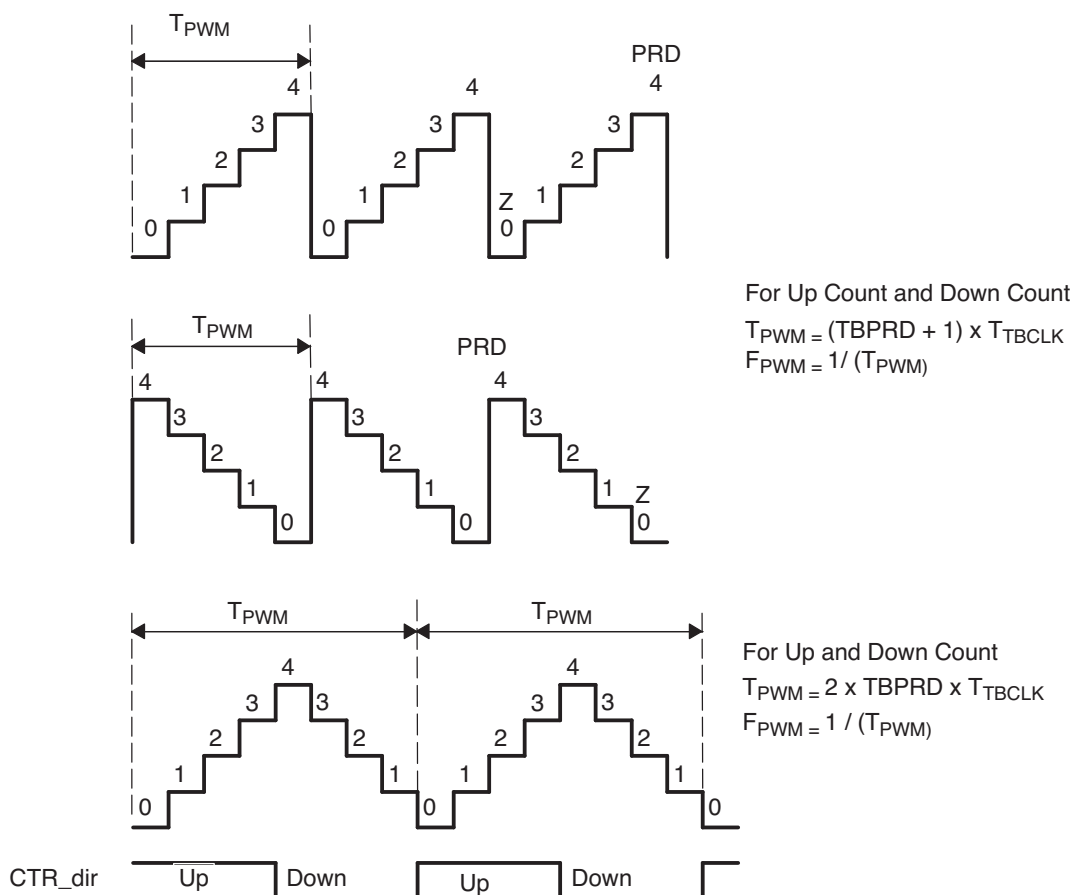
2.2.3.1 Time-Base Period Shadow Register

The time-base period register (TBPRD) has a shadow register. Shadowing allows the register update to be synchronized with the hardware. The following definitions are used to describe all shadow registers in the ePWM module:

- **Active Register**

  The active register controls the hardware and is responsible for actions that the hardware causes or invokes.

- **Shadow Register**

  The shadow register buffers or provides a temporary holding location for the active register. It has no direct effect on any control hardware. At a strategic point in time the shadow register's content is transferred to the active register. This prevents corruption or spurious operation due to the register being asynchronously modified by software.

The memory address of the shadow period register is the same as the active register. Which register is written to or read from is determined by the TBCTL[PRDLD] bit. This bit enables and disables the TBPRD shadow register as follows:

- **Time-Base Period Shadow Mode:**

  The TBPRD shadow register is enabled when TBCTL[PRDLD] = 0. Reads from and writes to the TBPRD memory address go to the shadow register. The shadow register contents are transferred to the active register (TBPRD (Active) ← TBPRD (shadow)) when the time-base counter equals zero (TBCTR = 0x0000). By default the TBPRD shadow register is enabled.

- **Time-Base Period Immediate Load Mode:**

  If immediate load mode is selected (TBCTL[PRDLD] = 1), then a read from or a write to the TBPRD memory address goes directly to the active register.

2.2.3.2 Time-Base Counter Synchronization

A time-base synchronization scheme connects all of the ePWM modules on a device. Each ePWM module has a synchronization input (EPWMxSYNCI) and a synchronization output (EPWMxSYNCO). The input synchronization for the first instance (ePWM1) comes from an external pin. The possible synchronization connections for the remaining ePWM modules are shown in Figure 2-4, Figure 2-5, and Figure 2-6.

Scheme 1 shown in Figure 2-4 applies to the 280x and 2801x devices. Scheme 1 also applies to the 2804x devices when the ePWM pinout is configured for 280x compatible mode (GPAMCFG[EPWMMODE] = 0).

Figure 2-4. Time-Base Counter Synchronization Scheme 1
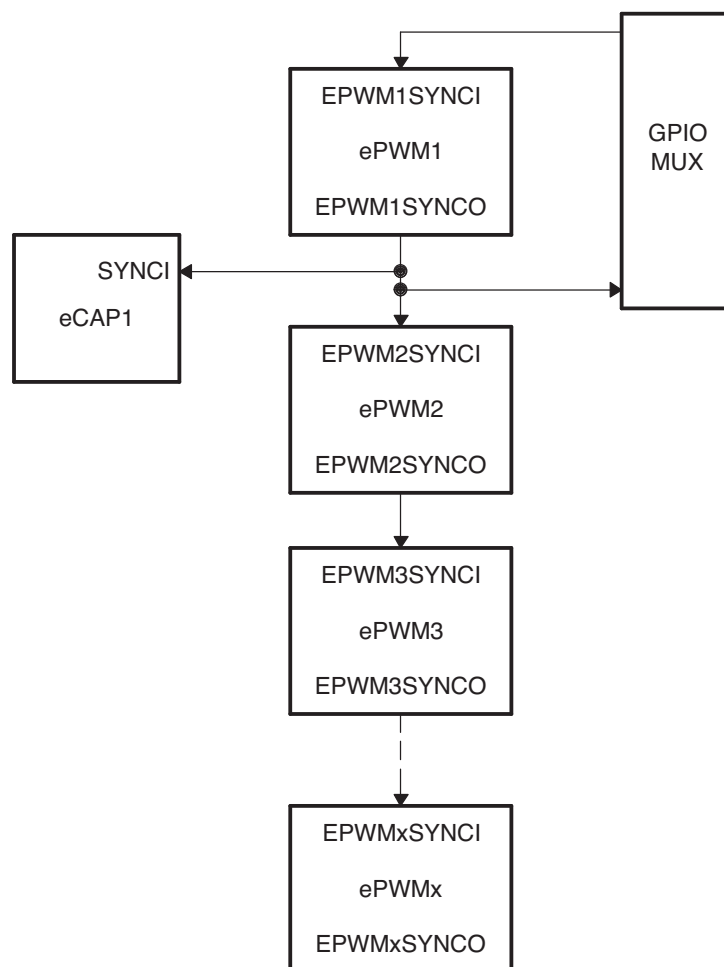
Scheme 2 shown in Figure 2-5 is used by the 2804x devices when the ePWM pinout is configured for A-channel only mode (GPAMCFG[EPWMMODE] = 3). If the 2804x ePWM pinout is configured for 280x compatible mode (GPAMCFG[EPWMMODE] = 0), then Scheme 1 is used.

Figure 2-5. Time-Base Counter Synchronization Scheme 2
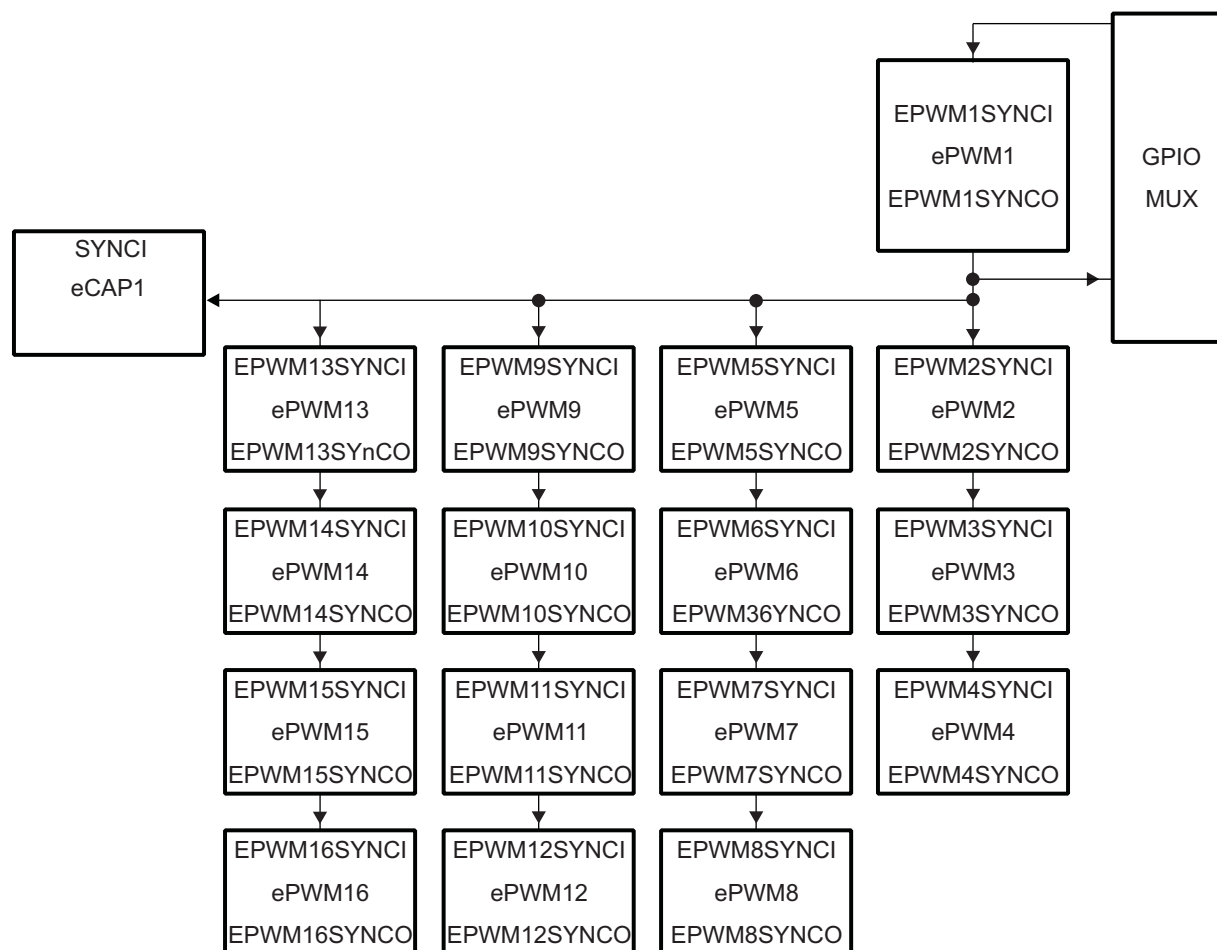
Scheme 3, shown in Figure 2-6, is used by all other devices.

Figure 2-6. Time-Base Counter Synchronization Scheme 3
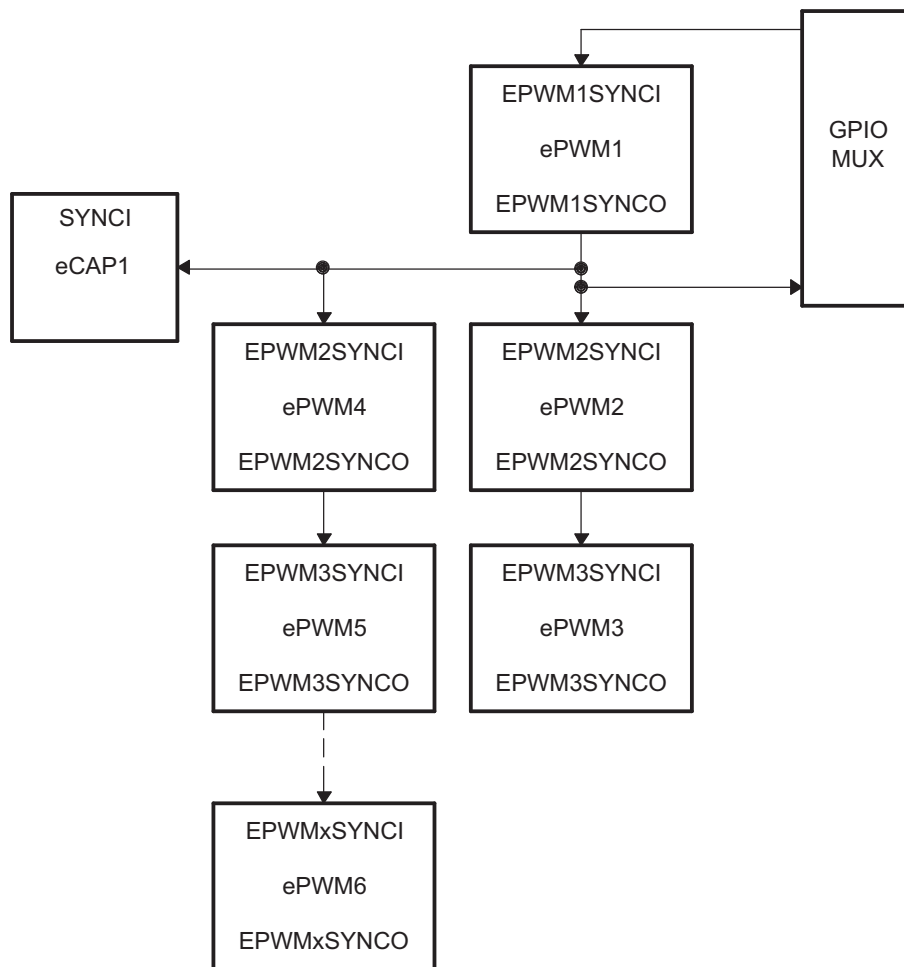
Each ePWM module can be configured to use or ignore the synchronization input. If the TBCTL[PHSEN] bit is set, then the time-base counter (TBCTR) of the ePWM module will be automatically loaded with the phase register (TBPHS) contents when one of the following conditions occur:

- **EPWMxSYNCI: Synchronization Input Pulse:**

    The value of the phase register is loaded into the counter register when an input synchronization pulse is detected (TBPHS → TBCNT). This operation occurs on the next valid time-base clock (TBCLK) edge.

- **Software Forced Synchronization Pulse:**

    Writing a 1 to the TBCTL[SWFSYNC] control bit invokes a software forced synchronization. This pulse is ORed with the synchronization input signal, and therefore has the same effect as a pulse on EPWMxSYNCI.

This feature enables the ePWM module to be automatically synchronized to the time base of another ePWM module. Lead or lag phase control can be added to the waveforms generated by different ePWM modules to synchronize them. In up-down-count mode, the TBCTL[PSHDIR] bit configures the direction of the time-base counter immediately after a synchronization event. The new direction is independent of the direction prior to the synchronization event. The TBPHS bit is ignored in count-up or count-down modes. See Figure 2-7 through Figure 2-10 for examples.

Clearing the TBCTL[PHSEN] bit configures the ePWM to ignore the synchronization input pulse. The synchronization pulse can still be allowed to flow-through to the EPWMxSYNCO and be used to synchronize other ePWM modules. In this way, you can set up a master time-base (for example, ePWM1) and downstream modules (ePWM2 - ePWMx) may elect to run in synchronization with the master. See the Application to Power Topologies Chapter 3 for more details on synchronization strategies.

2.2.4 Phase Locking the Time-Base Clocks of Multiple ePWM Modules

The TBCLKSYNC bit can be used to globally synchronize the time-base clocks of all enabled ePWM modules on a device. This bit is part of the DSPs clock enable registers and is described in the specific device version of the *System Control and Interrupts Reference Guide* listed in Section 1. When TBCLKSYNC = 0, the time-base clock of all ePWM modules is stopped (default). When TBCLKSYNC = 1, all ePWM time-base clocks are started with the rising edge of TBCLK aligned. For perfectly synchronized TBCLKs, the prescaler bits in the TBCTL register of each ePWM module must be set identically. The proper procedure for enabling the ePWM clocks is as follows:

1. Enable the individual ePWM module clocks. This is described in the specific device version of the *System Control and Interrupts Reference Guide* listed in Section 1.
2. Set TBCLKSYNC = 0. This will stop the time-base clock within any enabled ePWM module.
3. Configure the prescaler values and desired ePWM modes.
4. Set TBCLKSYNC = 1.

2.2.5 Time-base Counter Modes and Timing Waveforms

The time-base counter operates in one of four modes:

- Up-count mode which is asymmetrical.
- Down-count mode which is asymmetrical.
- Up-down-count which is symmetrical
- Frozen where the time-base counter is held constant at the current value

To illustrate the operation of the first three modes, the following timing diagrams show when events are generated and how the time-base responds to an EPWMxSYNCI signal.

Figure 2-7. Time-Base Up-Count Mode Waveforms
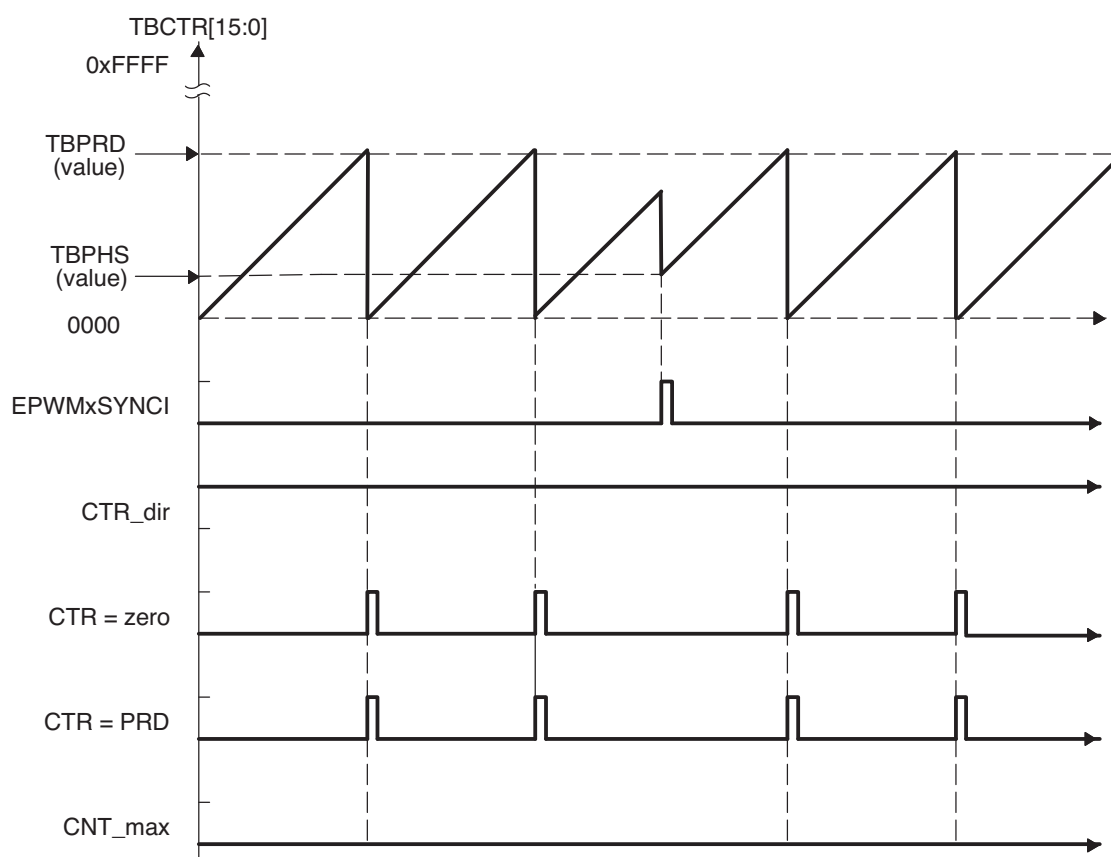
Figure 2-8. Time-Base Down-Count Mode Waveforms
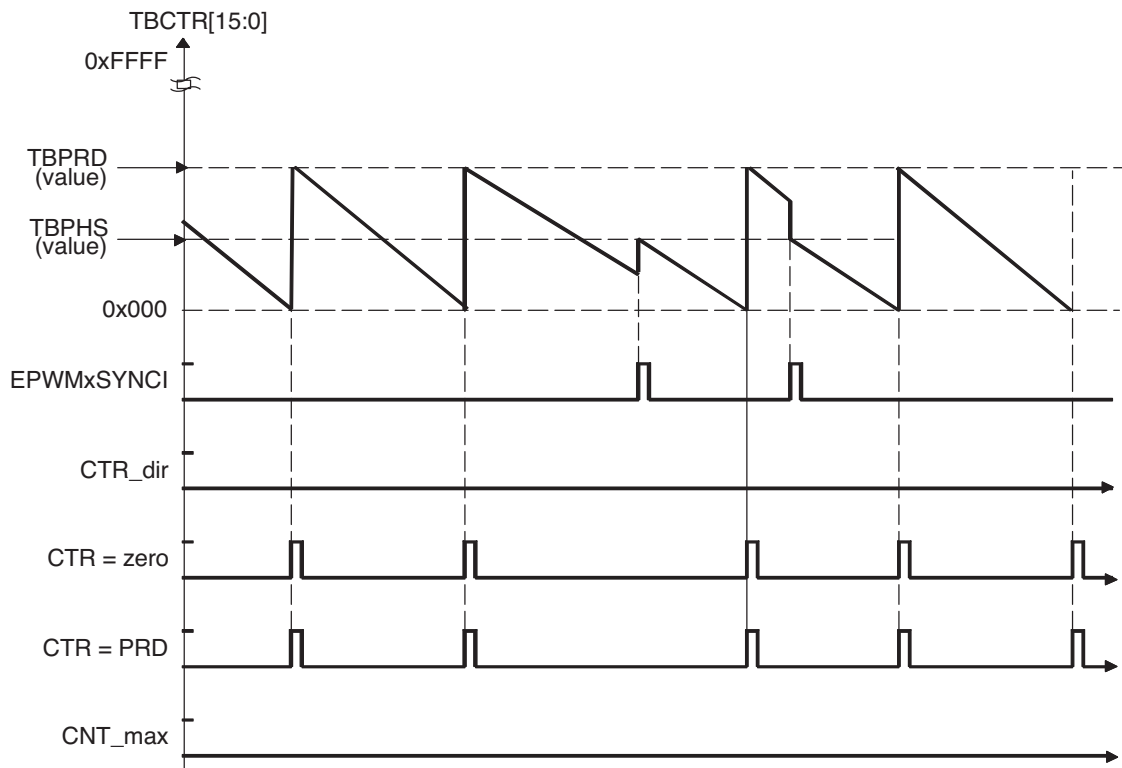
Figure 2-9. Time-Base Up-Down-Count Waveforms, TBCTL[PHSDIR = 0] Count Down On Synchronization Event
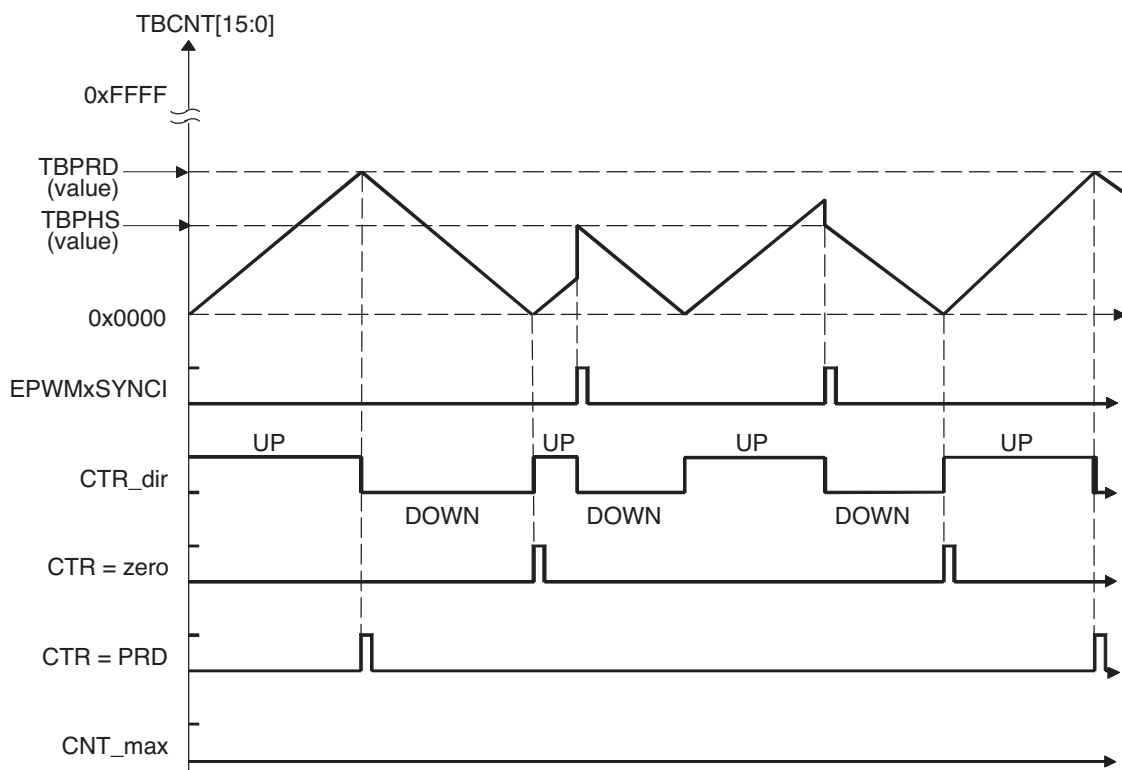
Figure 2-10. Time-Base Up-Down Count Waveforms, TBCTL[PHSDIR = 1] Count Up On Synchronization Event
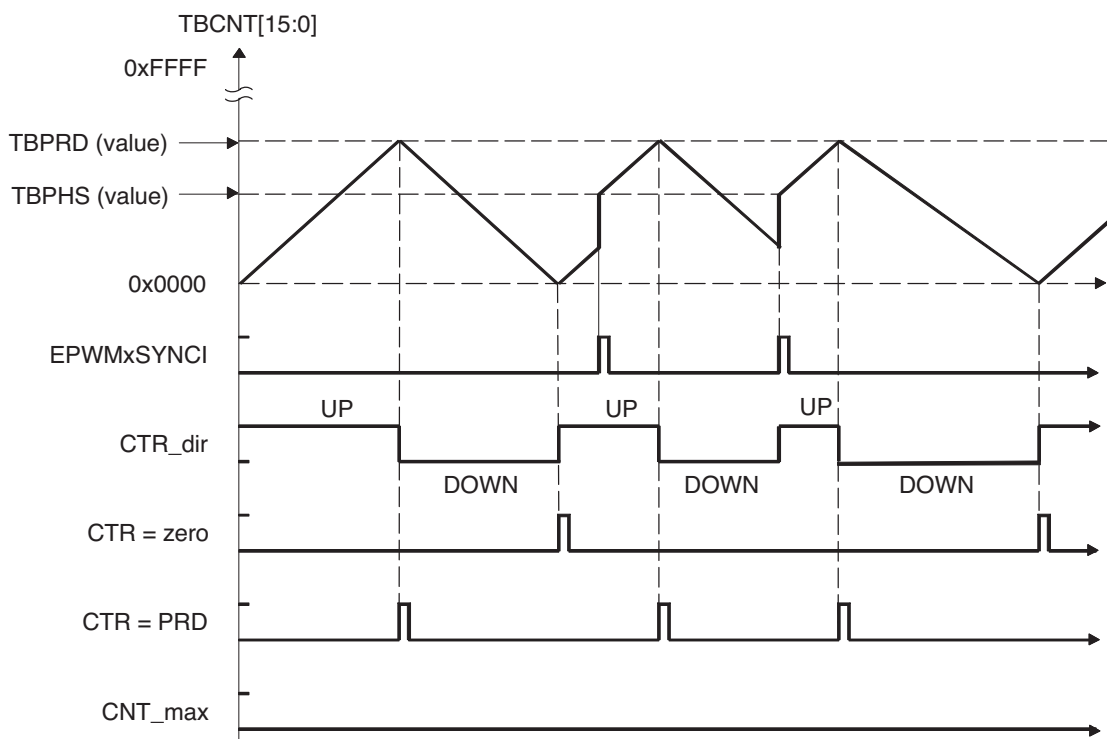
2.3 Counter-Compare (CC) Submodule

Figure 2-11 illustrates the counter-compare submodule within the ePWM.

Figure 2-11. Counter-Compare Submodule
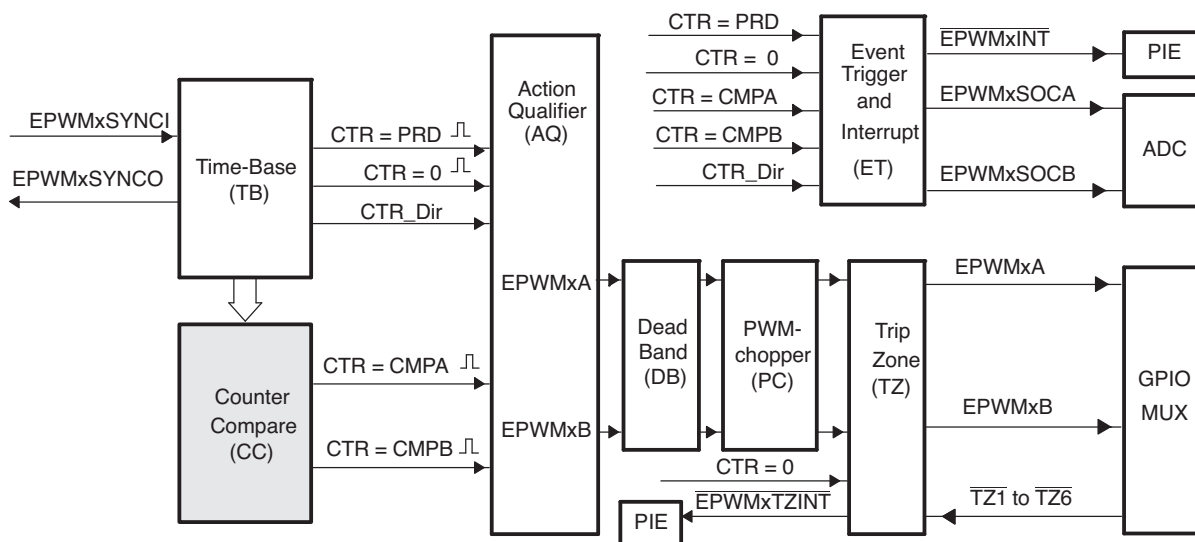
Figure 2-12 shows the basic structure of the counter-compare submodule.

2.3.1 Purpose of the Counter-Compare Submodule

The counter-compare submodule takes as input the time-base counter value. This value is continuously compared to the counter-compare A (CMPA) and counter-compare B (CMPB) registers. When the time-base counter is equal to one of the compare registers, the counter-compare unit generates an appropriate event.

The counter-compare:

- Generates events based on programmable time stamps using the CMPA and CMPB registers
  - CTR = CMPA: Time-base counter equals counter-compare A register (TBCTR = CMPA).
  - CTR = CMPB: Time-base counter equals counter-compare B register (TBCTR = CMPB)
- Controls the PWM duty cycle if the action-qualifier submodule is configured appropriately
- Shadows new compare values to prevent corruption or glitches during the active PWM cycle

2.3.2 Controlling and Monitoring the Counter-Compare Submodule

The counter-compare submodule operation is controlled and monitored by the registers shown in Table 2-4:

Table 2-4. Counter-Compare Submodule Registers

Register Name	Address Offset	Shadowed	Description
CMPCTL	0x0007	No	Counter-Compare Control Register.
CMPAHR	0x0008	Yes	HRPWM Counter-Compare A Extension Register [1]
CMPA	0x0009	Yes	Counter-Compare A Register
CMPB	0x000A	Yes	Counter-Compare B Register

[1] This register is available only on ePWM modules with the high-resolution extension (HRPWM). On ePWM modules that do not include the HRPWM this location is reserved. This register is described in the *TMS320x28xx, 28xxx High-Resolution Pulse Width Modulator (HRPWM) Reference Guide* (SPRU924). Refer to the device specific data manual to determine which ePWM instances include this feature.

Figure 2-12. Detailed View of the Counter-Compare Submodule
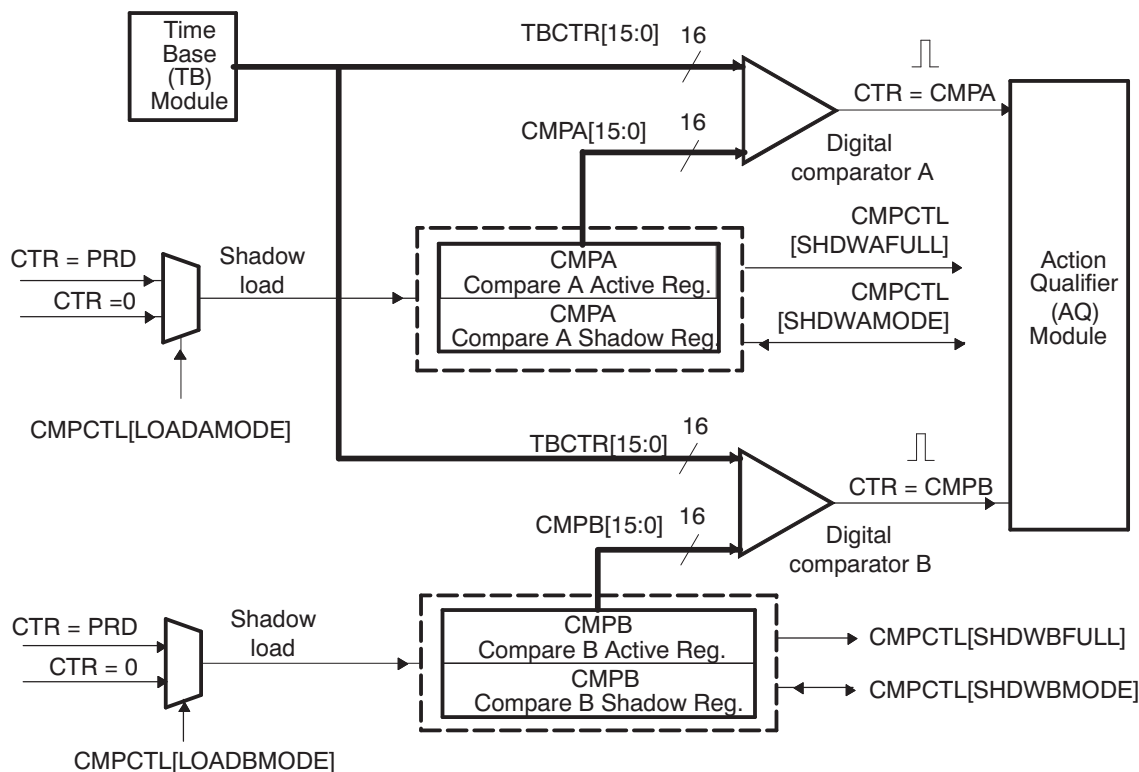
The key signals associated with the counter-compare submodule are described in Table 2-5.

Table 2-5. Counter-Compare Submodule Key Signals

Signal	Description of Event	Registers Compared
CTR = CMPA	Time-base counter equal to the active counter-compare A value	TBCTR = CMPA
CTR = CMPB	Time-base counter equal to the active counter-compare B value	TBCTR = CMPB
CTR = PRD	Time-base counter equal to the active period. Used to load active counter-compare A and B registers from the shadow register	TBCTR = TBPRD
CTR = ZERO	Time-base counter equal to zero. Used to load active counter-compare A and B registers from the shadow register	TBCTR = 0x0000

2.3.3 Operational Highlights for the Counter-Compare Submodule

The counter-compare submodule is responsible for generating two independent compare events based on two compare registers:

1. CTR = CMPA: Time-base counter equal to counter-compare A register (TBCTR = CMPA).
2. CTR = CMPB: Time-base counter equal to counter-compare B register (TBCTR = CMPB).

For up-count or down-count mode, each event occurs only once per cycle. For up-down-count mode each event occurs twice per cycle if the compare value is between 0x0000-TBPRD and once per cycle if the compare value is equal to 0x0000 or equal to TBPRD. These events are fed into the action-qualifier submodule where they are qualified by the counter direction and converted into actions if enabled. Refer to Section 2.4.1 for more details.

The counter-compare registers CMPA and CMPB each have an associated shadow register. Shadowing provides a way to keep updates to the registers synchronized with the hardware. When shadowing is used, updates to the active registers only occurs at strategic points. This prevents corruption or spurious operation due to the register being asynchronously modified by software. The memory address of the active register and the shadow register is identical. Which register is written to or read from is determined by the CMPCTL[SHDWAMODE] and CMPCTL[SHDWBMODE] bits. These bits enable and disable the CMPA shadow register and CMPB shadow register respectively. The behavior of the two load modes is described below:

- **Shadow Mode:**

  The shadow mode for the CMPA is enabled by clearing the CMPCTL[SHDWAMODE] bit and the shadow register for CMPB is enabled by clearing the CMPCTL[SHDWBMODE] bit. Shadow mode is enabled by default for both CMPA and CMPB.

  If the shadow register is enabled then the content of the shadow register is transferred to the active register on one of the following events:

  – CTR = PRD: Time-base counter equal to the period (TBCTR = TBPRD).
  – CTR = Zero: Time-base counter equal to zero (TBCTR = 0x0000)
  – Both CTR = PRD and CTR = Zero

  Which of these three events is specified by the CMPCTL[LOADAMODE] and CMPCTL[LOADBMODE] register bits. Only the active register contents are used by the counter-compare submodule to generate events to be sent to the action-qualifier.

- **Immediate Load Mode:**

  If immediate load mode is selected (i.e., TBCTL[SHADWAMODE] = 1 or TBCTL[SHADWBMODE] = 1), then a read from or a write to the register will go directly to the active register.

2.3.4 Count Mode Timing Waveforms

The counter-compare module can generate compare events in all three count modes:

- Up-count mode: used to generate an asymmetrical PWM waveform.
- Down-count mode: used to generate an asymmetrical PWM waveform.

- Up-down-count mode: used to generate a symmetrical PWM waveform.

To best illustrate the operation of the first three modes, the timing diagrams in Figure 2-13 through Figure 2-16 show when events are generated and how the EPWMxSYNCI signal interacts.

Figure 2-13. Counter-Compare Event Waveforms in Up-Count Mode
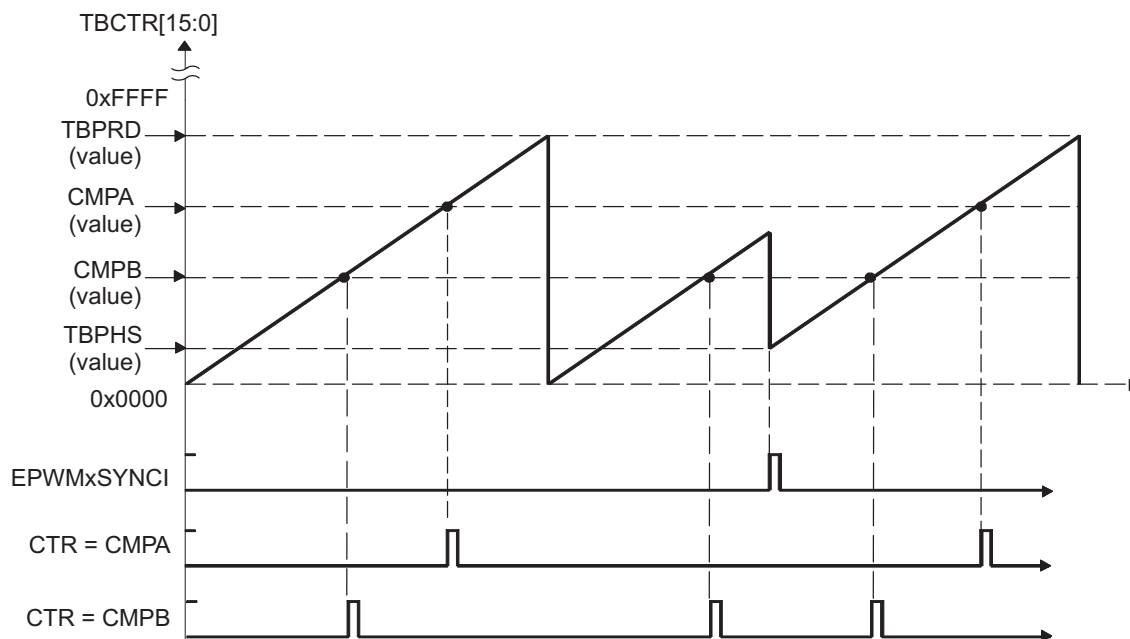
NOTE: An EPWMxSYNCI external synchronization event can cause a discontinuity in the TBCTR count sequence. This can lead to a compare event being skipped. This skipping is considered normal operation and must be taken into account.

Figure 2-14. Counter-Compare Events in Down-Count Mode
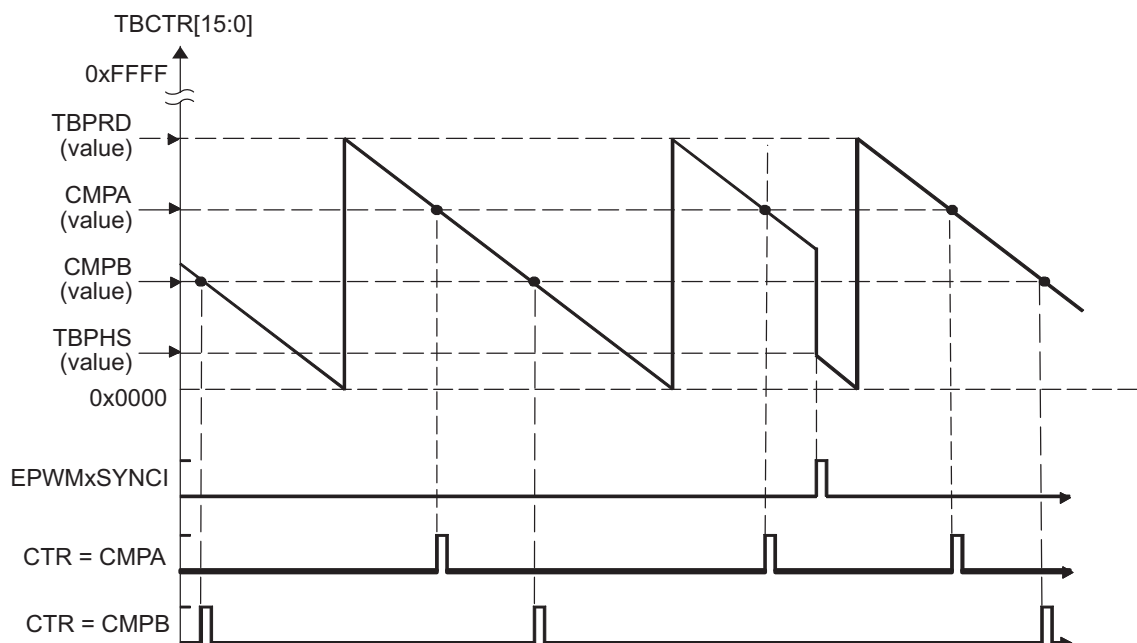
Figure 2-15. Counter-Compare Events In Up-Down-Count Mode, TBCTL[PHSDIR = 0] Count Down On Synchronization Event
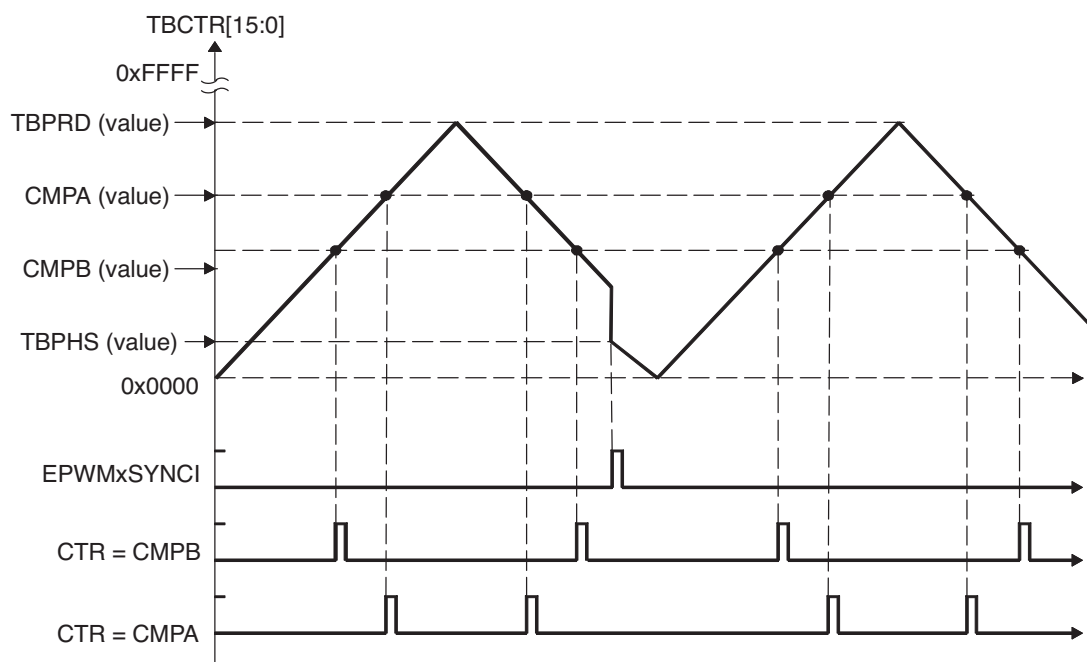
Figure 2-16. Counter-Compare Events In Up-Down-Count Mode, TBCTL[PHSDIR = 1] Count Up On Synchronization Event

Submit Documentation Feedback
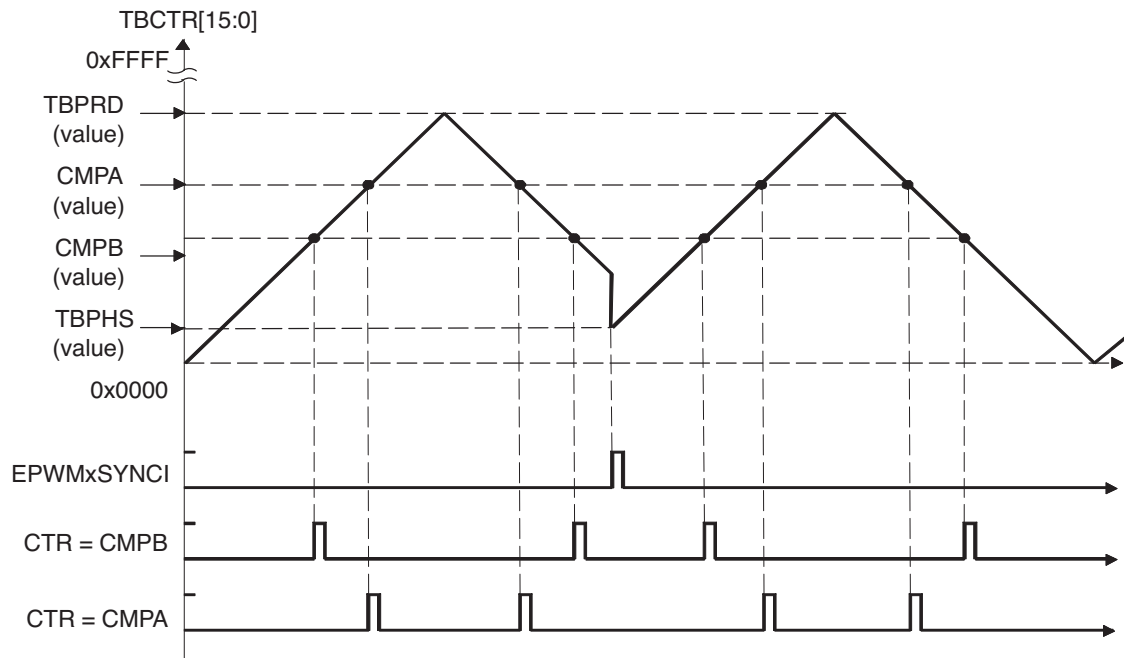
2.4 Action-Qualifier (AQ) Submodule

Figure 2-17 shows the action-qualifier (AQ) submodule (see shaded block) in the ePWM system.

Figure 2-17. Action-Qualifier Submodule
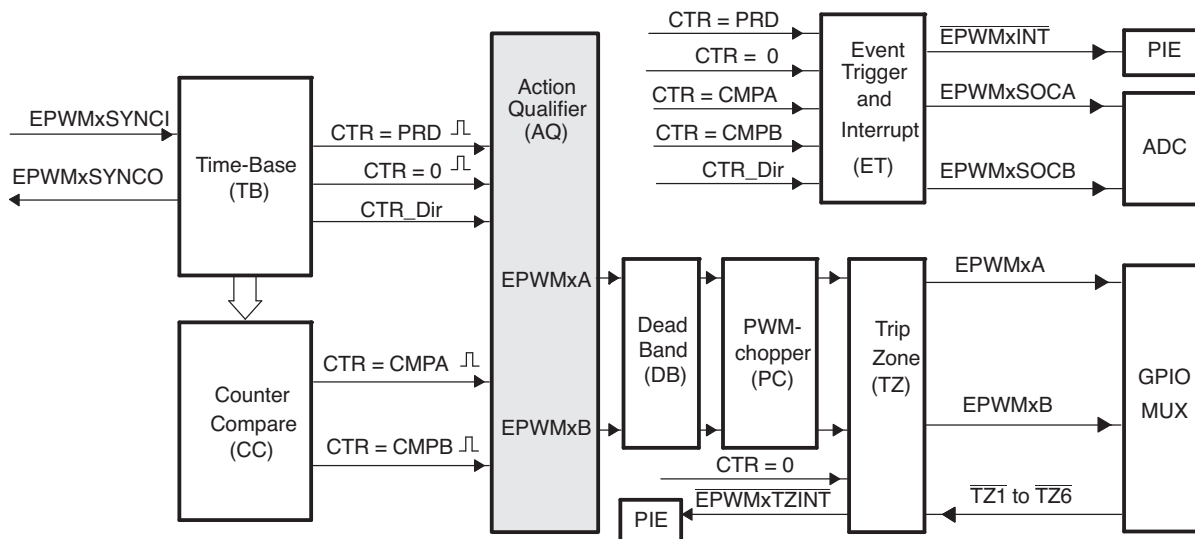
The action-qualifier submodule has the most important role in waveform construction and PWM generation. It decides which events are converted into various action types, thereby producing the required switched waveforms at the EPWMxA and EPWMxB outputs.

2.4.1 Purpose of the Action-Qualifier Submodule

The action-qualifier submodule is responsible for the following:

- Qualifying and generating actions (set, clear, toggle) based on the following events:
  - CTR = PRD: Time-base counter equal to the period (TBCTR = TBPRD).
  - CTR = Zero: Time-base counter equal to zero (TBCTR = 0x0000)
  - CTR = CMPA: Time-base counter equal to the counter-compare A register (TBCTR = CMPA)
  - CTR = CMPB: Time-base counter equal to the counter-compare B register (TBCTR = CMPB)
- Managing priority when these events occur concurrently
- Providing independent control of events when the time-base counter is increasing and when it is decreasing. .

2.4.2 Action-Qualifier Submodule Control and Status Register Definitions

The action-qualifier submodule operation is controlled and monitored via the registers in Table 2-6.

Table 2-6. Action-Qualifier Submodule Registers

Register Name	Address offset	Shadowed	Description
AQCTLA	0x000B	No	Action-Qualifier Control Register For Output A (EPWMxA)
AQCTLB	0x000C	No	Action-Qualifier Control Register For Output B (EPWMxB)
AQSFRC	0x000D	No	Action-Qualifier Software Force Register
AQCSFRC	0x000E	Yes	Action-Qualifier Continuous Software Force

The action-qualifier submodule is based on event-driven logic. It can be thought of as a programmable cross switch with events at the input and actions at the output, all of which are software controlled via the set of registers shown in Table 2-6.

Figure 2-18. Action-Qualifier Submodule Inputs and Outputs
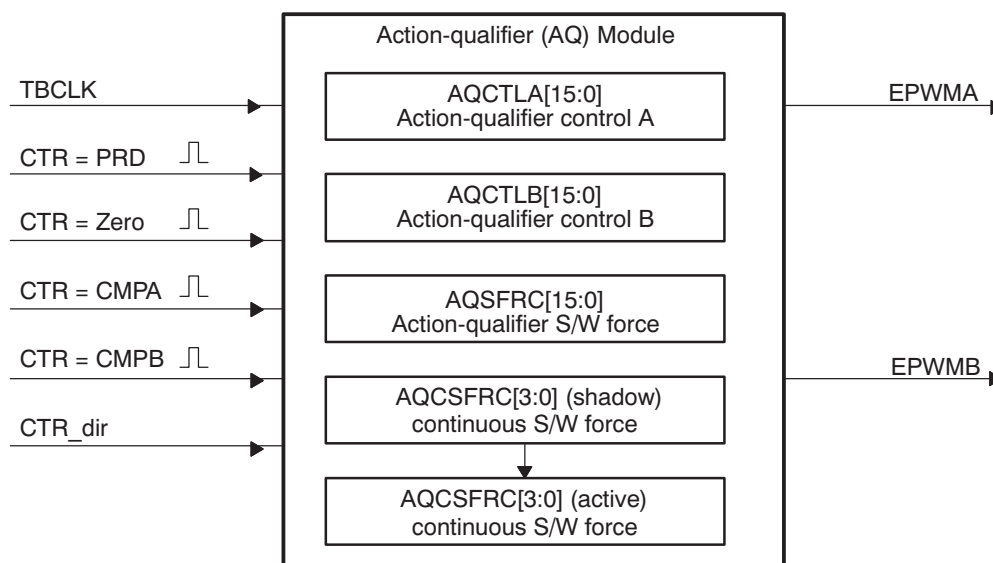
For convenience, the possible input events are summarized again in Table 2-7.

Table 2-7. Action-Qualifier Submodule Possible Input Events

Signal	Description	Registers Compared
CTR = PRD	Time-base counter equal to the period value	TBCTR = TBPRD
CTR = Zero	Time-base counter equal to zero	TBCTR = 0x0000
CTR = CMPA	Time-base counter equal to the counter-compare A	TBCTR = CMPA
CTR = CMPB	Time-base counter equal to the counter-compare B	TBCTR = CMPB
Software forced event	Asynchronous event initiated by software	

The software forced action is a useful asynchronous event. This control is handled by registers AQSFRC and AQCSFRC.

The action-qualifier submodule controls how the two outputs EPWMxA and EPWMxB behave when a particular event occurs. The event inputs to the action-qualifier submodule are further qualified by the counter direction (up or down). This allows for independent action on outputs on both the count-up and count-down phases.

The possible actions imposed on outputs EPWMxA and EPWMxB are:
- **Set High:**
  Set output EPWMxA or EPWMxB to a high level.
- **Clear Low:**
  Set output EPWMxA or EPWMxB to a low level.
- **Toggle:**
  If EPWMxA or EPWMxB is currently pulled high, then pull the output low. If EPWMxA or EPWMxB is currently pulled low, then pull the output high.
- **Do Nothing:**
  Keep outputs EPWMxA and EPWMxB at same level as currently set. Although the "Do Nothing" option prevents an event from causing an action on the EPWMxA and EPWMxB outputs, this event can still trigger interrupts and ADC start of conversion. See the Event-trigger Submodule description in Section 2.8 for details.

Actions are specified independently for either output (EPWMxA or EPWMxB). Any or all events can be configured to generate actions on a given output. For example, both CTR = CMPA and CTR = CMPB can operate on output EPWMxA. All qualifier actions are configured via the control registers found at the end of this section.

For clarity, the drawings in this document use a set of symbolic actions. These symbols are summarized in Figure 2-19. Each symbol represents an action as a marker in time. Some actions are fixed in time (zero and period) while the CMPA and CMPB actions are moveable and their time positions are programmed via the counter-compare A and B registers, respectively. To turn off or disable an action, use the "Do Nothing option"; it is the default at reset.

Figure 2-19. Possible Action-Qualifier Actions for EPWMxA and EPWMxB Outputs

S/W force	TB Counter equals:				Actions
	Zero	Comp A	Comp B	Period	
SW ✕	Z ✕	CA ✕	CB ✕	P ✕	Do Nothing
SW ↓	Z ↓	CA ↓	CB ↓	P ↓	Clear Low
SW ↑	Z ↑	CA ↑	CB ↑	P ↑	Set High
SW T	Z T	CA T	CB T	P T	Toggle

2.4.3 Action-Qualifier Event Priority

It is possible for the ePWM action qualifier to receive more than one event at the same time. In this case events are assigned a priority by the hardware. The general rule is events occurring later in time have a higher priority and software forced events always have the highest priority. The event priority levels for up-down-count mode are shown in Table 2-8. A priority level of 1 is the highest priority and level 7 is the lowest. The priority changes slightly depending on the direction of TBCTR.

Table 2-8. Action-Qualifier Event Priority for Up-Down-Count Mode

Priority Level	Event If TBCTR is Incrementing TBCTR = Zero up to TBCTR = TBPRD	Event If TBCTR is Decrementing TBCTR = TBPRD down to TBCTR = 1
1 (Highest)	Software forced event	Software forced event
2	Counter equals CMPB on up-count (CBU)	Counter equals CMPB on down-count (CBD)
3	Counter equals CMPA on up-count (CAU)	Counter equals CMPA on down-count (CAD)
4	Counter equals zero	Counter equals period (TBPRD)
5	Counter equals CMPB on down-count (CBD) [1]	Counter equals CMPB on up-count (CBU) [1]
6 (Lowest)	Counter equals CMPA on down-count (CAD) [1]	Counter equals CMPA on up-count (CBU) [1]

[1] To maintain symmetry for up-down-count mode, both up-events (CAU/CBU) and down-events (CAD/CBD) can be generated for TBPRD. Otherwise, up-events can occur only when the counter is incrementing and down-events can occur only when the counter is decrementing.

Table 2-9 shows the action-qualifier priority for up-count mode. In this case, the counter direction is always defined as up and thus down-count events will never be taken.

Table 2-9. Action-Qualifier Event Priority for Up-Count Mode

Priority Level	Event
1 (Highest)	Software forced event
2	Counter equal to period (TBPRD)
3	Counter equal to CMPB on up-count (CBU)
4	Counter equal to CMPA on up-count (CAU)
5 (Lowest)	Counter equal to Zero

Table 2-10 shows the action-qualifier priority for down-count mode. In this case, the counter direction is always defined as down and thus up-count events will never be taken.

Table 2-10. Action-Qualifier Event Priority for Down-Count Mode

Priority Level	Event
1 (Highest)	Software forced event
2	Counter equal to Zero
3	Counter equal to CMPB on down-count (CBD)
4	Counter equal to CMPA on down-count (CAD)
5 (Lowest)	Counter equal to period (TBPRD)

It is possible to set the compare value greater than the period. In this case the action will take place as shown in Table 2-11.

Table 2-11. Behavior if CMPA/CMPB is Greater than the Period

Counter Mode	Compare on Up-Count Event CAU/CBU	Compare on Down-Count Event CAU/CBU
Up-Count Mode	If CMPA/CMPB ≤ TBPRD period, then the event occurs on a compare match (TBCTR=CMPA or CMPB).	Never occurs.
	If CMPA/CMPB > TBPRD, then the event will not occur.	
Down-Count Mode	Never occurs.	If CMPA/CMPB < TBPRD, the event will occur on a compare match (TBCTR=CMPA or CMPB).
		If CMPA/CMPB ≥ TBPRD, the event will occur on a period match (TBCTR=TBPRD).
Up-Down-Count Mode	If CMPA/CMPB < TBPRD and the counter is incrementing, the event occurs on a compare match (TBCTR=CMPA or CMPB).	If CMPA/CMPB < TBPRD and the counter is decrementing, the event occurs on a compare match (TBCTR=CMPA or CMPB).
	If CMPA/CMPB is ≥ TBPRD, the event will occur on a period match (TBCTR = TBPRD).	If CMPA/CMPB ≥ TBPRD, the event occurs on a period match (TBCTR=TBPRD).

2.4.4 Waveforms for Common Configurations

Note: The waveforms in this document show the ePWMs behavior for a static compare register value. In a running system, the active compare registers (CMPA and CMPB) are typically updated from their respective shadow registers once every period. The user specifies when the update will take place; either when the time-base counter reaches zero or when the time-base counter reaches period. There are some cases when the action based on the new value can be delayed by one period or the action based on the old value can take effect for an extra period. Some PWM configurations avoid this situation. These include, but are not limited to, the following:

Use up-down-count mode to generate a symmetric PWM:

- If you load CMPA/CMPB on zero, then use CMPA/CMPB values greater than or equal to 1.
- If you load CMPA/CMPB on period, then use CMPA/CMPB values less than or equal to TBPRD-1.

  This means there will always be a pulse of at least one TBCLK cycle in a PWM period which, when very short, tend to be ignored by the system.

Use up-down-count mode to generate an asymmetric PWM:

- To achieve 50%-0% asymmetric PWM use the following configuration: Load CMPA/CMPB on period and use the period action to clear the PWM and a compare-up action to set the PWM. Modulate the compare value from 0 to TBPRD to achieve 50%-0% PWM duty.

When using up-count mode to generate an asymmetric PWM:

- To achieve 0-100% asymmetric PWM use the following configuration: Load CMPA/CMPB on TBPRD. Use the Zero action to set the PWM and a compare-up action to clear the PWM. Modulate the compare value from 0 to TBPRD+1 to achieve 0-100% PWM duty.

  See the *Using Enhanced Pulse Width Modulator (ePWM) Module for 0-100% Duty Cycle Control* Application Report (literature number SPRAAI1)

Figure 2-20 shows how a symmetric PWM waveform can be generated using the up-down-count mode of the TBCTR. In this mode 0%-100% DC modulation is achieved by using equal compare matches on the up count and down count portions of the waveform. In the example shown, CMPA is used to make the comparison. When the counter is incrementing the CMPA match will pull the PWM output high. Likewise, when the counter is decrementing the compare match will pull the PWM signal low. When CMPA = 0, the PWM signal is low for the entire period giving the 0% duty waveform. When CMPA = TBPRD, the PWM signal is high achieving 100% duty.

When using this configuration in practice, if you load CMPA/CMPB on zero, then use CMPA/CMPB values greater than or equal to 1. If you load CMPA/CMPB on period, then use CMPA/CMPB values less than or equal to TBPRD-1. This means there will always be a pulse of at least one TBCLK cycle in a PWM period which, when very short, tend to be ignored by the system.

Figure 2-20. Up-Down-Count Mode Symmetrical Waveform
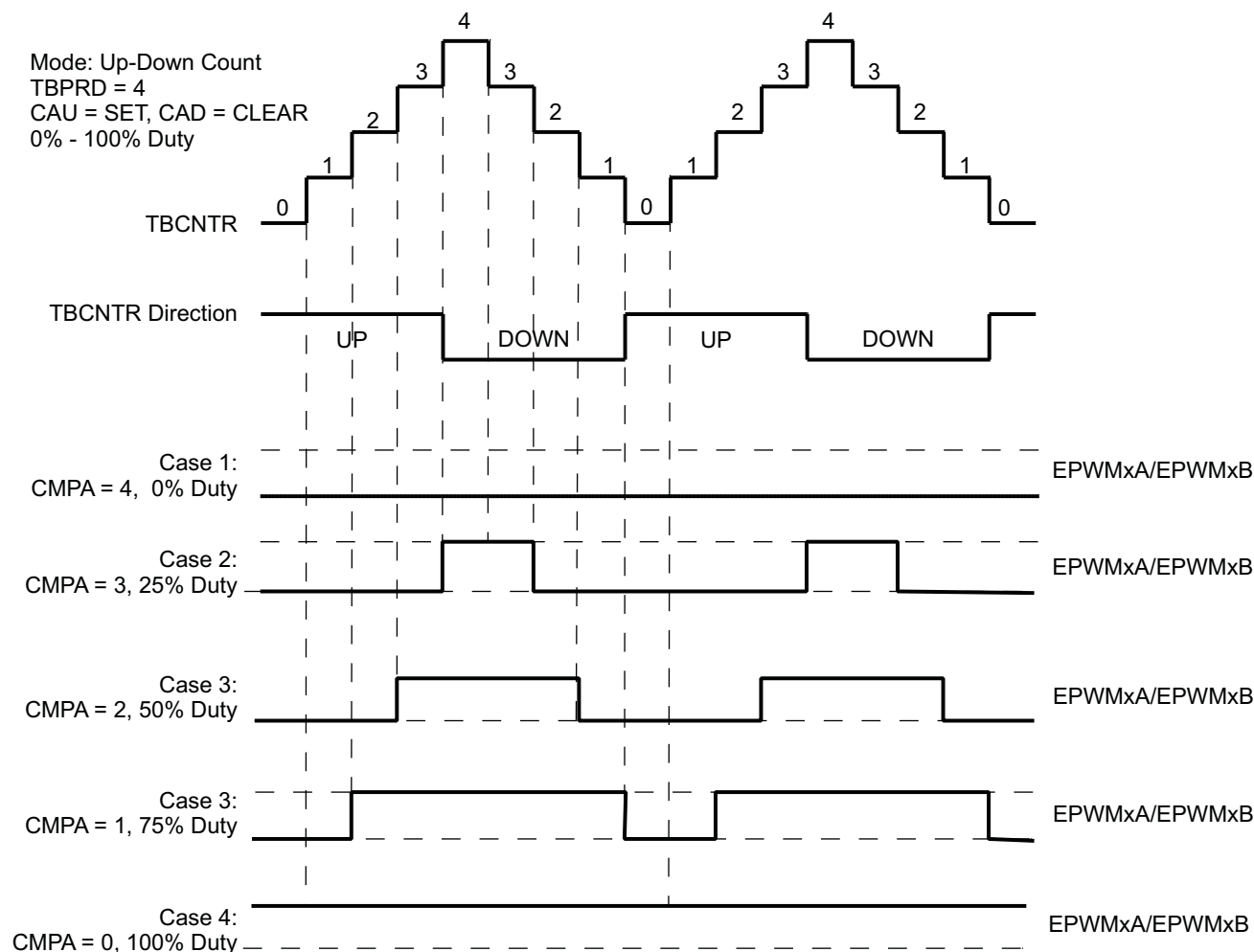
The PWM waveforms in Figure 2-21 through Figure 2-26 show some common action-qualifier configurations. The C-code samples in Example 2-2 through Example 2-7 shows how to configure an ePWM module for each case. Some conventions used in the figures and examples are as follows:

- TBPRD, CMPA, and CMPB refer to the value written in their respective registers. The active register, not the shadow register, is used by the hardware.
- CMPx, refers to either CMPA or CMPB.
- EPWMxA and EPWMxB refer to the output signals from ePWMx
- Up-Down means Count-up-and-down mode, Up means up-count mode and Dwn means down-count mode
- Sym = Symmetric, Asym = Asymmetric

Figure 2-21. Up, Single Edge Asymmetric Waveform, With Independent Modulation on EPWMxA and EPWMxB—Active High
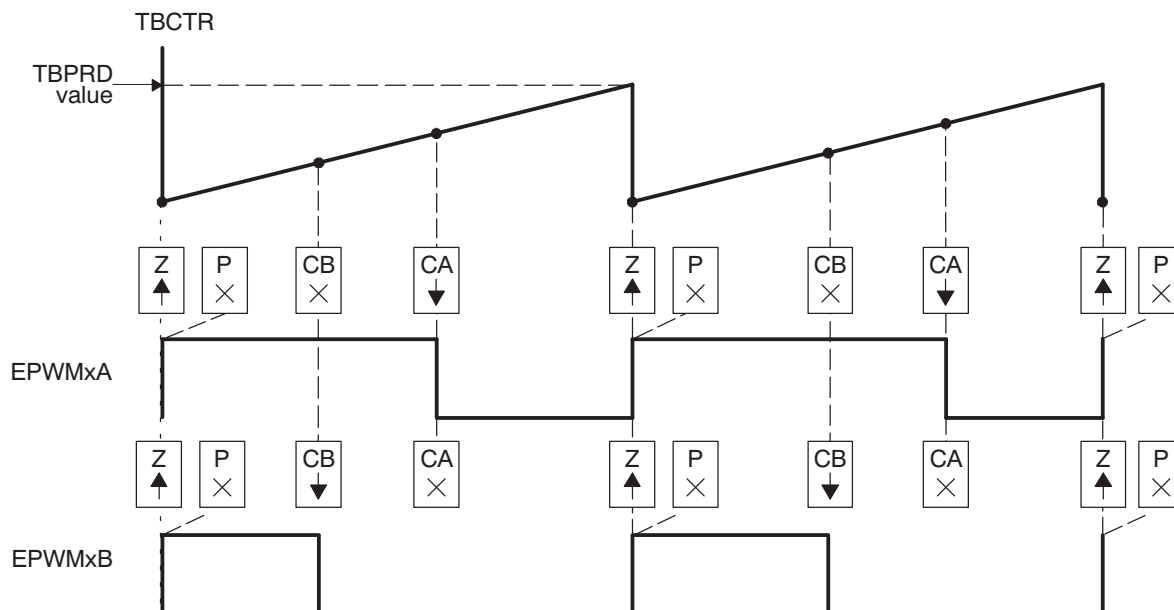
A   PWM period = (TBPRD + 1) × T_{TBCLK}

B   Duty modulation for EPWMxA is set by CMPA, and is active high (that is, high time duty proportional to CMPA).

C   Duty modulation for EPWMxB is set by CMPB and is active high (that is, high time duty proportional to CMPB).

D   The "Do Nothing" actions (X) are shown for completeness, but will not be shown on subsequent diagrams.

E   Actions at zero and period, although appearing to occur concurrently, are actually separated by one TBCLK period. TBCTR wraps from period to 0000.

Example 2-2 contains a code sample showing initialization and run time for the waveforms in Figure 2-21.

Example 2-2. Code Sample for Figure 2-21

```
// Initialization Time
// = = = = = = = = = = = = = = = = = = = = = = =  =
EPwm1Regs.TBPRD = 600;                       // Period = 601 TBCLK counts
EPwm1Regs.CMPA.half.CMPA = 350;              // Compare A = 350 TBCLK counts
EPwm1Regs.CMPB = 200;                        // Compare B = 200 TBCLK counts
EPwm1Regs.TBPHS = 0;                         // Set Phase register to zero
EPwm1Regs.TBCTR = 0;                         // clear TB counter
EPwm1Regs.TBCTL.bit.CTRMODE = TB_UP;
EPwm1Regs.TBCTL.bit.PHSEN = TB_DISABLE;      // Phase loading disabled
EPwm1Regs.TBCTL.bit.PRDLD = TB_SHADOW;
EPwm1Regs.TBCTL.bit.SYNCOSEL = TB_SYNC_DISABLE;
EPwm1Regs.TBCTL.bit.HSPCLKDIV = TB_DIV1;     // TBCLK = SYSCLK
EPwm1Regs.TBCTL.bit.CLKDIV = TB_DIV1;
EPwm1Regs.CMPCTL.bit.SHDWAMODE = CC_SHADOW;
EPwm1Regs.CMPCTL.bit.SHDWBMODE = CC_SHADOW;
EPwm1Regs.CMPCTL.bit.LOADAMODE = CC_CTR_ZERO; // load on CTR = Zero
EPwm1Regs.CMPCTL.bit.LOADBMODE = CC_CTR_ZERO; // load on CTR = Zero
EPwm1Regs.AQCTLA.bit.ZRO = AQ_SET;
EPwm1Regs.AQCTLA.bit.CAU = AQ_CLEAR;
EPwm1Regs.AQCTLB.bit.ZRO = AQ_SET;
EPwm1Regs.AQCTLB.bit.CBU = AQ_CLEAR;
//
// Run Time
// = = = = = = = = = = = = = = = = = = = = = = =  =
EPwm1Regs.CMPA.half.CMPA = Duty1A;           // adjust duty for output EPWM1A
EPwm1Regs.CMPB = Duty1B;                      // adjust duty for output EPWM1B
```

Figure 2-22. Up, Single Edge Asymmetric Waveform With Independent Modulation on EPWMxA and EPWMxB—Active Low
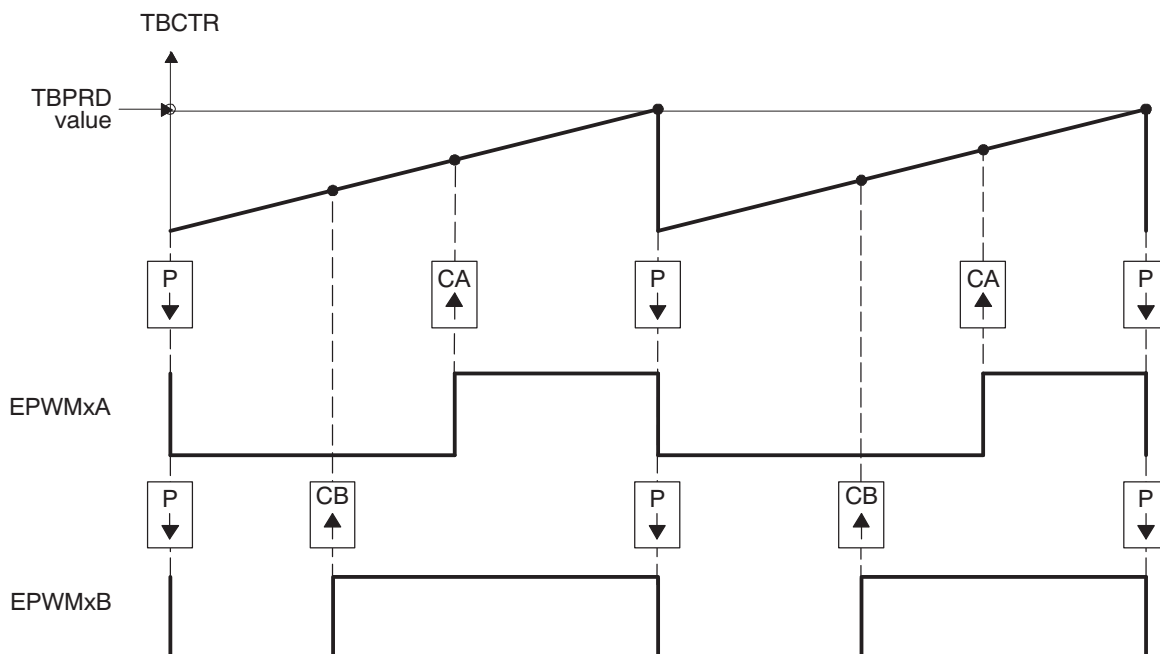
A   PWM period = (TBPRD + 1) × T_{TBCLK}

B   Duty modulation for EPWMxA is set by CMPA, and is active low (that is, the low time duty is proportional to CMPA).

C   Duty modulation for EPWMxB is set by CMPB and is active low (that is, the low time duty is proportional to CMPB).

D   The Do Nothing actions (X) are shown for completeness here, but will not be shown on subsequent diagrams.

E   Actions at zero and period, although appearing to occur concurrently, are actually separated by one TBCLK period. TBCTR wraps from period to 0000.

Example 2-3 contains a code sample showing initialization and run time for the waveforms in Figure 2-22.

Example 2-3. Code Sample for Figure 2-22

```
// Initialization Time
// = = = = = = = = = = = = = = = = = = = = = = =  =
EPwm1Regs.TBPRD = 600;                     // Period = 601 TBCLK counts
EPwm1Regs.CMPA.half.CMPA = 350;            // Compare A = 350 TBCLK counts
EPwm1Regs.CMPB = 200;                      // Compare B = 200 TBCLK counts
EPwm1Regs.TBPHS = 0;                       // Set Phase register to zero
EPwm1Regs.TBCTR = 0;                       // clear TB counter
EPwm1Regs.TBCTL.bit.CTRMODE = TB_UP;
EPwm1Regs.TBCTL.bit.PHSEN = TB_DISABLE;  // Phase loading disabled
EPwm1Regs.TBCTL.bit.PRDLD = TB_SHADOW;
EPwm1Regs.TBCTL.bit.SYNCOSEL = TB_SYNC_DISABLE;
EPwm1Regs.TBCTL.bit.HSPCLKDIV = TB_DIV1; // TBCLK = SYSCLKOUT
EPwm1Regs.TBCTL.bit.CLKDIV = TB_DIV1;
EPwm1Regs.CMPCTL.bit.SHDWAMODE = CC_SHADOW;
EPwm1Regs.CMPCTL.bit.SHDWBMODE = CC_SHADOW;
EPwm1Regs.CMPCTL.bit.LOADAMODE = CC_CTR_ZERO; // load on TBCTR = Zero
EPwm1Regs.CMPCTL.bit.LOADBMODE = CC_CTR_ZERO; // load on TBCTR = Zero
EPwm1Regs.AQCTLA.bit.PRD = AQ_CLEAR;
EPwm1Regs.AQCTLA.bit.CAU = AQ_SET;
EPwm1Regs.AQCTLB.bit.PRD = AQ_CLEAR;
EPwm1Regs.AQCTLB.bit.CBU = AQ_SET;
//
// Run Time
// = = = = = = = = = = = = = = = = = = = = = = = =
EPwm1Regs.CMPA.half.CMPA = Duty1A;     // adjust duty for output EPWM1A
EPwm1Regs.CMPB = Duty1B;               // adjust duty for output EPWM1B
```

Figure 2-23. Up-Count, Pulse Placement Asymmetric Waveform With Independent Modulation on EPWMxA
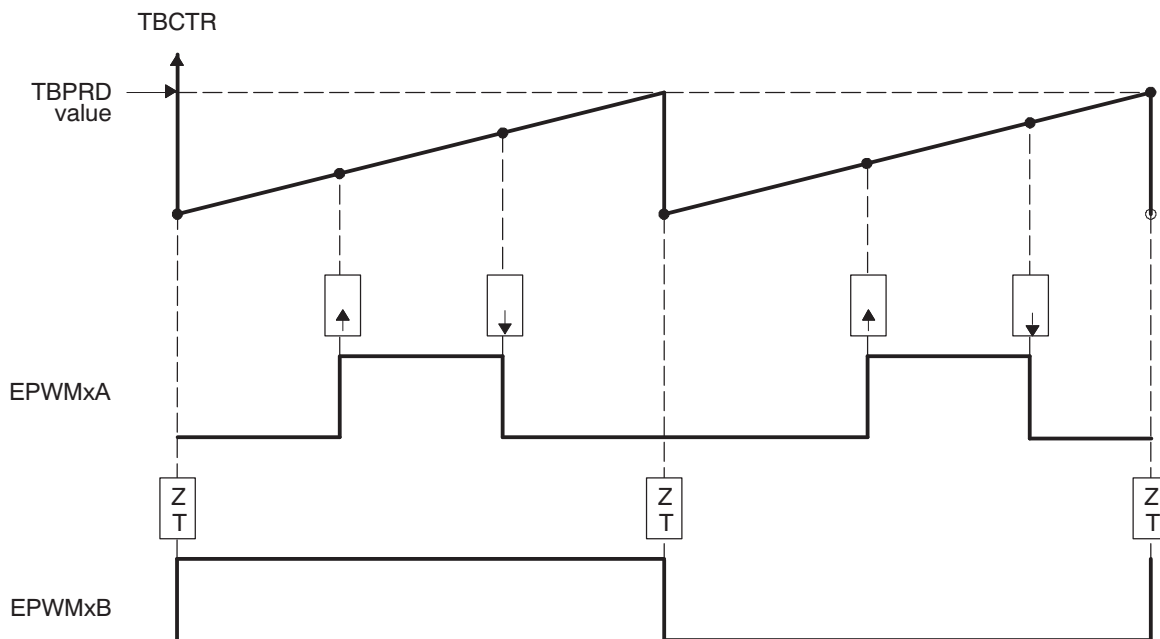
A   PWM frequency = 1/((TBPRD + 1) × T_{TBCLK})

B   Pulse can be placed anywhere within the PWM cycle (0000 - TBPRD)

C   High time duty proportional to (CMPB - CMPA)

D   EPWMxB can be used to generate a 50% duty square wave with frequency = 1/2 × ((TBPRD + 1) × TBCLK)

Example 2-4 contains a code sample showing initialization and run time for the waveforms Figure 2-23. Use the code in Example 2-1 to define the headers.

Example 2-4. Code Sample for Figure 2-23

```
// Initialization Time
// = = = = = = = = = = = = = = = = = = = = = = = = =  =
EPwm1Regs.TBPRD = 600;                           // Period = 601 TBCLK counts
EPwm1Regs.CMPA.half.CMPA = 200;                  // Compare A = 200 TBCLK counts
EPwm1Regs.CMPB = 400;                            // Compare B = 400 TBCLK counts
EPwm1Regs.TBPHS = 0;                             // Set Phase register to zero
EPwm1Regs.TBCTR = 0;                             // clear TB counter
EPwm1Regs.TBCTL.bit.CTRMODE = TB_UP;
EPwm1Regs.TBCTL.bit.PHSEN = TB_DISABLE;          // Phase loading disabled
EPwm1Regs.TBCTL.bit.PRDLD = TB_SHADOW;
EPwm1Regs.TBCTL.bit.SYNCOSEL = TB_SYNC_DISABLE;
EPwm1Regs.TBCTL.bit.HSPCLKDIV = TB_DIV1;         // TBCLK = SYSCLKOUT
EPwm1Regs.TBCTL.bit.CLKDIV = TB_DIV1;
EPwm1Regs.CMPCTL.bit.SHDWAMODE = CC_SHADOW;
EPwm1Regs.CMPCTL.bit.SHDWBMODE = CC_SHADOW;
EPwm1Regs.CMPCTL.bit.LOADAMODE = CC_CTR_ZERO;    // load on TBCTR = Zero
EPwm1Regs.CMPCTL.bit.LOADBMODE = CC_CTR_ZERO;    // load on TBCTR = Zero
EPwm1Regs.AQCTLA.bit.CAU = AQ_SET;
EPwm1Regs.AQCTLA.bit.CBU = AQ_CLEAR;
EPwm1Regs.AQCTLB.bit.ZRO = AQ_TOGGLE;
//
// Run Time
// = = = = = = = = = = = = = = = = = = = = = = = =  =
EPwm1Regs.CMPA.half.CMPA = EdgePosA;             // adjust duty for output EPWM1A only
EPwm1Regs.CMPB = EdgePosB;
```

Figure 2-24. Up-Down-Count, Dual Edge Symmetric Waveform, With Independent Modulation on EPWMxA and EPWMxB — Active Low

A   PWM period = 2 x TBPRD × T_{TBCLK}

B   Duty modulation for EPWMxA is set by CMPA, and is active low (that is, the low time duty is proportional to CMPA).

C   Duty modulation for EPWMxB is set by CMPB and is active low (that is, the low time duty is proportional to CMPB).

D   Outputs EPWMxA and EPWMxB can drive independent power switches

Example 2-5 contains a code sample showing initialization and run time for the waveforms in Figure 2-24. Use the code in Example 2-1 to define the headers.

Example 2-5. Code Sample for Figure 2-24

```
// Initialization Time
// = = = = = = = = = = = = = = = = = = = = = = =  =
EPwm1Regs.TBPRD = 600;                      // Period = 2×600 TBCLK counts
EPwm1Regs.CMPA.half.CMPA = 400;          // Compare A = 400 TBCLK counts
EPwm1Regs.CMPB = 500;                       // Compare B = 500 TBCLK counts
EPwm1Regs.TBPHS = 0;                        // Set Phase register to zero
EPwm1Regs.TBCNT = 0;                        // clear TB counter
EPwm1Regs.TBCTL.bit.CTRMODE = TB_UPDOWN;  // Symmetric
xEPwm1Regs.TBCTL.bit.PHSEN = TB_DISABLE;   // Phase loading disabled
xEPwm1Regs.TBCTL.bit.PRDLD = TB_SHADOW;
EPwm1Regs.TBCTL.bit.SYNCOSEL = TB_SYNC_DISABLE;
EPwm1Regs.TBCTL.bit.HSPCLKDIV = TB_DIV1;  // TBCLK = SYSCLKOUT
EPwm1Regs.TBCTL.bit.CLKDIV = TB_DIV1;
EPwm1Regs.CMPCTL.bit.SHDWAMODE = CC_SHADOW;
EPwm1Regs.CMPCTL.bit.SHDWBMODE = CC_SHADOW;
EPwm1Regs.CMPCTL.bit.LOADAMODE = CC_CTR_ZERO; // load on CTR = Zero
EPwm1Regs.CMPCTL.bit.LOADBMODE = CC_CTR_ZERO; // load on CTR = Zero
EPwm1Regs.AQCTLA.bit.CAU = AQ_SET;
EPwm1Regs.AQCTLA.bit.CAD = AQ_CLEAR;
EPwm1Regs.AQCTLB.bit.CBU = AQ_SET;
EPwm1Regs.AQCTLB.bit.CBD = AQ_CLEAR;
//
// Run Time
// = = = = = = = = = = = = = = = = = = = = = = = =
EPwm1Regs.CMPA.half.CMPA = Duty1A;       // adjust duty for output EPWM1A
EPwm1Regs.CMPB = Duty1B;                  // adjust duty for output EPWM1B
```

Figure 2-25. Up-Down-Count, Dual Edge Symmetric Waveform, With Independent Modulation on EPWMxA and EPWMxB — Complementary
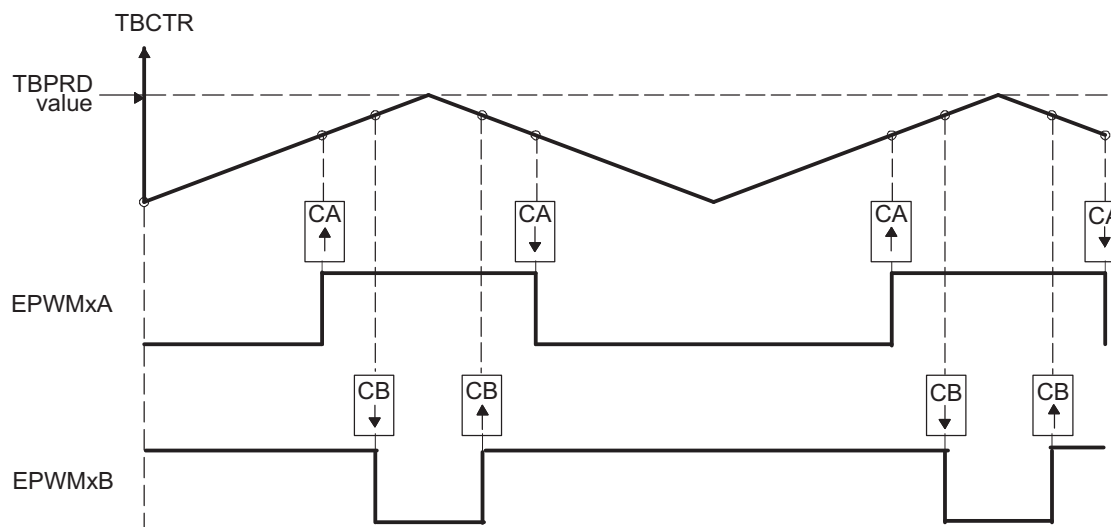
A   PWM period = 2 × TBPRD × T_{TBCLK}

B   Duty modulation for EPWMxA is set by CMPA, and is active low, i.e., low time duty proportional to CMPA

C   Duty modulation for EPWMxB is set by CMPB and is active high, i.e., high time duty proportional to CMPB

D   Outputs EPWMx can drive upper/lower (complementary) power switches

E   Dead-band = CMPB - CMPA (fully programmable edge placement by software). Note the dead-band module is also available if the more classical edge delay method is required.

Example 2-6 contains a code sample showing initialization and run time for the waveforms in Figure 2-25. Use the code in Example 2-1 to define the headers.

Example 2-6. Code Sample for Figure 2-25

```
// Initialization Time
// = = = = = = = = = = = = = = = = = = = = = = =  =
EPwm1Regs.TBPRD = 600;                          // Period = 2×600 TBCLK counts
EPwm1Regs.CMPA.half.CMPA = 350;                 // Compare A = 350 TBCLK counts
EPwm1Regs.CMPB = 400;                           // Compare B = 400 TBCLK counts
EPwm1Regs.TBPHS = 0;                            // Set Phase register to zero
EPwm1Regs.TBCNT = 0;                            // clear TB counter
EPwm1Regs.TBCTL.bit.CTRMODE = TB_UPDOWN;        // Symmetric
EPwm1Regs.TBCTL.bit.PHSEN = TB_DISABLE;         // Phase loading disabled
EPwm1Regs.TBCTL.bit.PRDLD = TB_SHADOW;
EPwm1Regs.TBCTL.bit.SYNCOSEL = TB_SYNC_DISABLE;
EPwm1Regs.TBCTL.bit.HSPCLKDIV = TB_DIV1;        // TBCLK = SYSCLKOUT
EPwm1Regs.TBCTL.bit.CLKDIV = TB_DIV1;
EPwm1Regs.CMPCTL.bit.SHDWAMODE = CC_SHADOW;
EPwm1Regs.CMPCTL.bit.SHDWBMODE = CC_SHADOW;
EPwm1Regs.CMPCTL.bit.LOADAMODE = CC_CTR_ZERO;  // load on CTR = Zero
EPwm1Regs.CMPCTL.bit.LOADBMODE = CC_CTR_ZERO;  // load on CTR = Zero
EPwm1Regs.AQCTLA.bit.CAU = AQ_SET;
EPwm1Regs.AQCTLA.bit.CAD = AQ_CLEAR;
EPwm1Regs.AQCTLB.bit.CBU = AQ_CLEAR;
EPwm1Regs.AQCTLB.bit.CBD = AQ_SET;
// Run Time
// = = = = = = = = = = = = = = = = = = = = = = =  =
EPwm1Regs.CMPA.half.CMPA = Duty1A;              // adjust duty for output EPWM1A
EPwm1Regs.CMPB = Duty1B;                        // adjust duty for output EPWM1B
```

Figure 2-26. Up-Down-Count, Dual Edge Asymmetric Waveform, With Independent Modulation on EPWMxA—Active Low
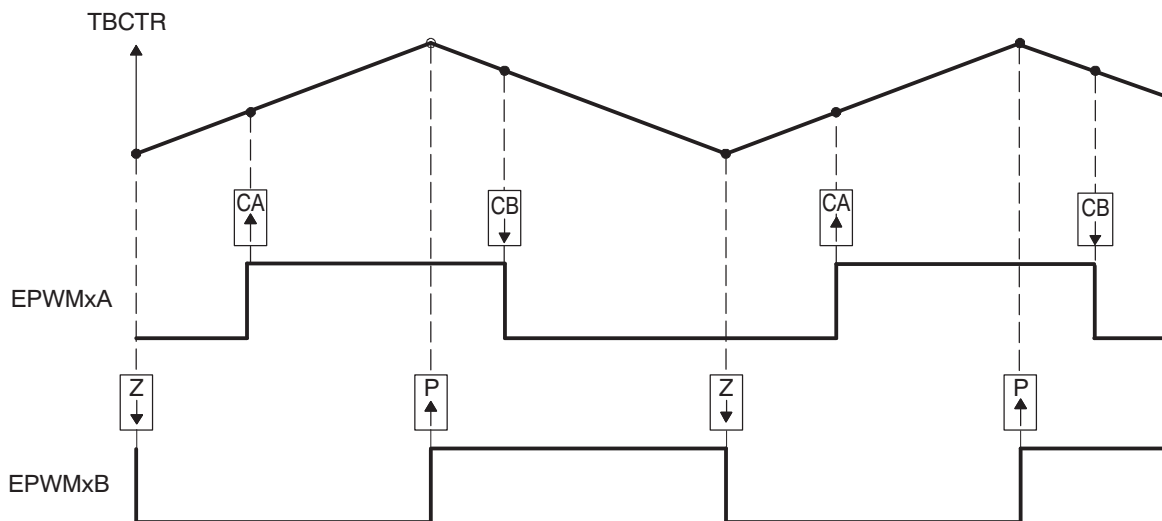
A   PWM period = 2 × TBPRD × TBCLK

B   Rising edge and falling edge can be asymmetrically positioned within a PWM cycle. This allows for pulse placement techniques.

C   Duty modulation for EPWMxA is set by CMPA and CMPB.

D   Low time duty for EPWMxA is proportional to (CMPA + CMPB).

E   To change this example to active high, CMPA and CMPB actions need to be inverted (i.e., Set ! Clear and Clear Set).

F   Duty modulation for EPWMxB is fixed at 50% (utilizes spare action resources for EPWMxB)

Example 2-7 contains a code sample showing initialization and run time for the waveforms in Figure 2-26. Use the code in Example 2-1 to define the headers.

Example 2-7. Code Sample for Figure 2-26

```
// Initialization Time
// = = = = = = = = = = = = = = = = = = = = = = =  =
EPwm1Regs.TBPRD = 600;                          // Period = 2 × 600 TBCLK counts
EPwm1Regs.CMPA.half.CMPA = 250;                 // Compare A = 250 TBCLK counts
EPwm1Regs.CMPB = 450;                           // Compare B = 450 TBCLK counts
EPwm1Regs.TBPHS = 0;                            // Set Phase register to zero
EPwm1Regs.TBCNT = 0;                            // clear TB counter
EPwm1Regs.TBCTL.bit.CTRMODE = TB_UPDOWN;        // Symmetric
EPwm1Regs.TBCTL.bit.PHSEN = TB_DISABLE;         // Phase loading disabled
EPwm1Regs.TBCTL.bit.PRDLD = TB_SHADOW;
EPwm1Regs.TBCTL.bit.SYNCOSEL = TB_SYNC_DISABLE;
EPwm1Regs.TBCTL.bit.HSPCLKDIV = TB_DIV1;        // TBCLK = SYSCLKOUT
EPwm1Regs.TBCTL.bit.CLKDIV = TB_DIV1;
EPwm1Regs.CMPCTL.bit.SHDWAMODE = CC_SHADOW;
EPwm1Regs.CMPCTL.bit.SHDWBMODE = CC_SHADOW;
EPwm1Regs.CMPCTL.bit.LOADAMODE = CC_CTR_ZERO; // load on CTR = Zero
EPwm1Regs.CMPCTL.bit.LOADBMODE = CC_CTR_ZERO; // load on CTR = Zero
EPwm1Regs.AQCTLA.bit.CAU = AQ_SET;
EPwm1Regs.AQCTLA.bit.CBD = AQ_CLEAR;
EPwm1Regs.AQCTLB.bit.ZRO = AQ_CLEAR;
EPwm1Regs.AQCTLB.bit.PRD = AQ_SET;
// Run Time
// = = = = = = = = = = = = = = = = = = = = = = =  =
EPwm1Regs.CMPA.half.CMPA = EdgePosA;            // adjust duty for output EPWM1A only
EPwm1Regs.CMPB = EdgePosB;
```

2.5 Dead-Band Generator (DB) Submodule

Figure 2-27 illustrates the dead-band submodule within the ePWM module.

Figure 2-27. Dead_Band Submodule
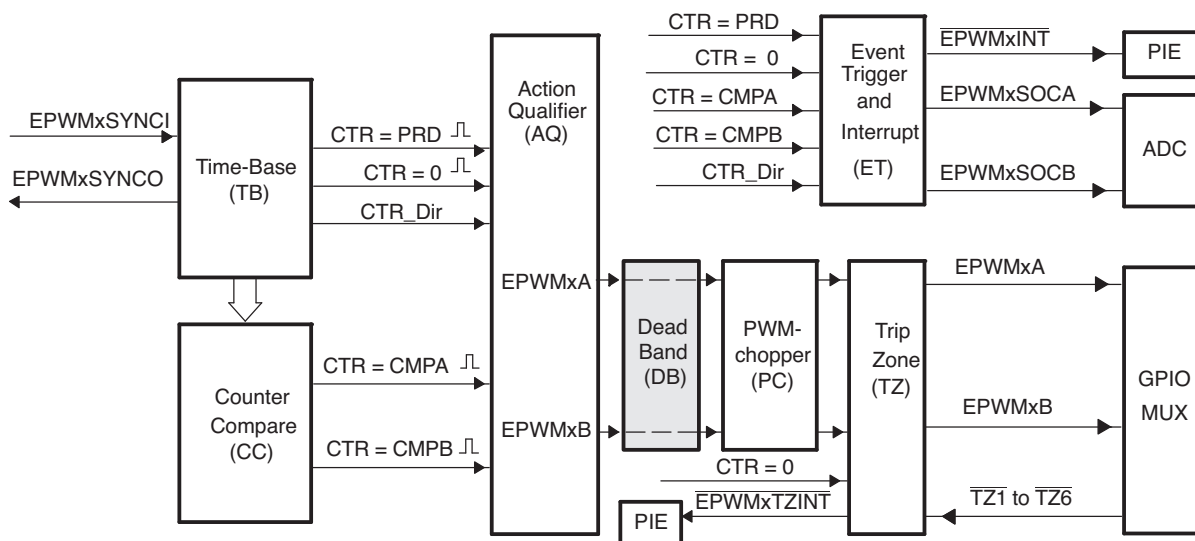
2.5.1 Purpose of the Dead-Band Submodule

The "Action-qualifier (AQ) Module" section discussed how it is possible to generate the required dead-band by having full control over edge placement using both the CMPA and CMPB resources of the ePWM module. However, if the more classical edge delay-based dead-band with polarity control is required, then the dead-band submodule described here should be used.

The key functions of the dead-band module are:

*   Generating appropriate signal pairs (EPWMxA and EPWMxB) with dead-band relationship from a single EPWMxA input
*   Programming signal pairs for:
    *   Active high (AH)
    *   Active low (AL)
    *   Active high complementary (AHC)
    *   Active low complementary (ALC)
*   Adding programmable delay to rising edges (RED)
*   Adding programmable delay to falling edges (FED)
*   Can be totally bypassed from the signal path (note dotted lines in diagram)

2.5.2 Controlling and Monitoring the Dead-Band Submodule

The dead-band submodule operation is controlled and monitored via the following registers:

Table 2-12. Dead-Band Generator Submodule Registers

Register Name	Address offset	Shadowed	Description
DBCTL	0x000F	No	Dead-Band Control Register
DBRED	0x0010	No	Dead-Band Rising Edge Delay Count Register
DBFED	0x0011	No	Dead-Band Falling Edge Delay Count Register

2.5.3 Operational Highlights for the Dead-Band Submodule

The following sections provide the operational highlights.

The dead-band submodule has two groups of independent selection options as shown in Figure 2-28.

- **Input Source Selection:**

  The input signals to the dead-band module are the EPWMxA and EPWMxB output signals from the action-qualifier. In this section they will be referred to as EPWMxA In and EPWMxB In. Using the DBCTL[IN_MODE] control bits, the signal source for each delay, falling-edge or rising-edge, can be selected:

  – EPWMxA In is the source for both falling-edge and rising-edge delay. This is the default mode.
  – EPWMxA In is the source for falling-edge delay, EPWMxB In is the source for rising-edge delay.
  – EPWMxA In is the source for rising edge delay, EPWMxB In is the source for falling-edge delay.
  – EPWMxB In is the source for both falling-edge and rising-edge delay.

- **Output Mode Control:**

  The output mode is configured by way of the DBCTL[OUT_MODE] bits. These bits determine if the falling-edge delay, rising-edge delay, neither, or both are applied to the input signals.

- **Polarity Control:**

  The polarity control (DBCTL[POLSEL]) allows you to specify whether the rising-edge delayed signal and/or the falling-edge delayed signal is to be inverted before being sent out of the dead-band submodule.

Figure 2-28. Configuration Options for the Dead-Band Submodule
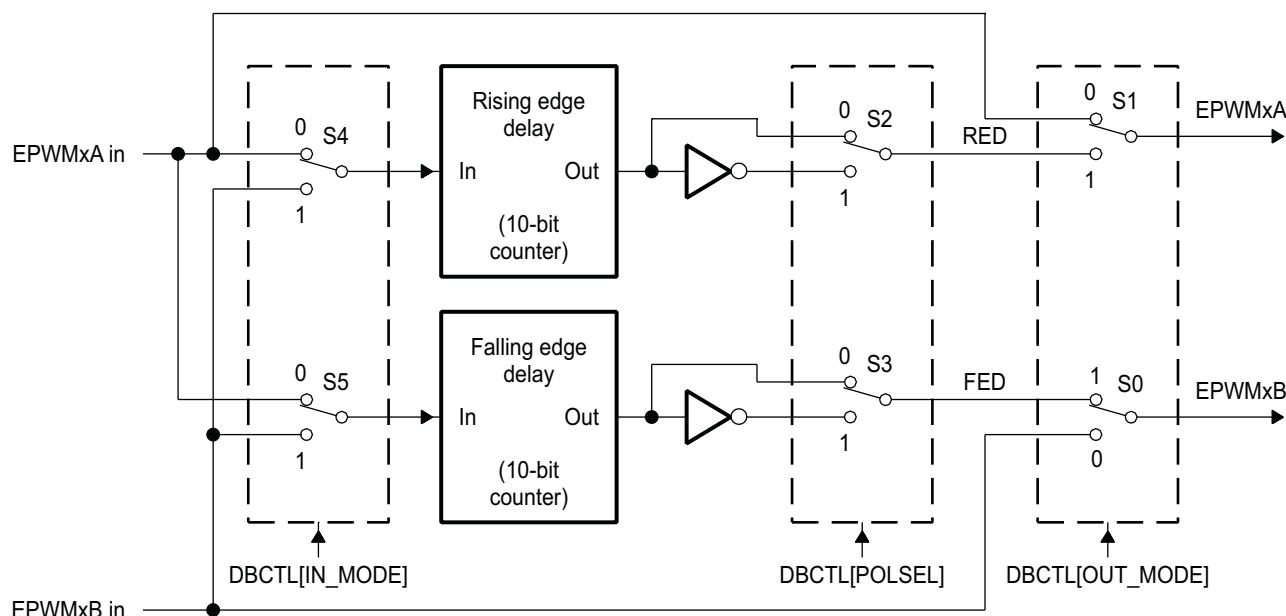
Although all combinations are supported, not all are typical usage modes. Table 2-13 documents some classical dead-band configurations. These modes assume that the DBCTL[IN_MODE] is configured such that EPWMxA In is the source for both falling-edge and rising-edge delay. Enhanced, or non-traditional modes can be achieved by changing the input signal source. The modes shown in Table 2-13 fall into the following categories:

- **Mode 1: Bypass both falling-edge delay (FED) and rising-edge delay (RED)**

  Allows you to fully disable the dead-band submodule from the PWM signal path.

- **Mode 2-5: Classical Dead-Band Polarity Settings:**

  These represent typical polarity configurations that should address all the active high/low modes required by available industry power switch gate drivers. The waveforms for these typical cases are shown in Figure 2-29. Note that to generate equivalent waveforms to Figure 2-29, configure the

action-qualifier submodule to generate the signal as shown for EPWMxA.

- **Mode 6: Bypass rising-edge-delay and Mode 7: Bypass falling-edge-delay**

    Finally the last two entries in Table 2-13 show combinations where either the falling-edge-delay (FED) or rising-edge-delay (RED) blocks are bypassed.

Table 2-13. Classical Dead-Band Operating Modes

Mode	Mode Description [1]	DBCTL[POLSEL]		DBCTL[OUT_MODE]	
		S3	S2	S1	S0
1	EPWMxA and EPWMxB Passed Through (No Delay)	X	X	0	0
2	Active High Complementary (AHC)	1	0	1	1
3	Active Low Complementary (ALC)	0	1	1	1
4	Active High (AH)	0	0	1	1
5	Active Low (AL)	1	1	1	1
6	EPWMxA Out = EPWMxA In (No Delay) EPWMxB Out = EPWMxA In with Falling Edge Delay	0 or 1	0 or 1	0	1
7	EPWMxA Out = EPWMxA In with Rising Edge Delay EPWMxB Out = EPWMxB In with No Delay	0 or 1	0 or 1	1	0

[1]   These are classical dead-band modes and assume that DBCTL[IN_MODE] = 0,0. That is, EPWMxA in is the source for both the falling-edge and rising-edge delays. Enhanced, non-traditional modes can be achieved by changing the IN_MODE configuration.

Figure 2-29 shows waveforms for typical cases where 0% < duty < 100%.

Figure 2-29. Dead-Band Waveforms for Typical Cases (0% < Duty < 100%)
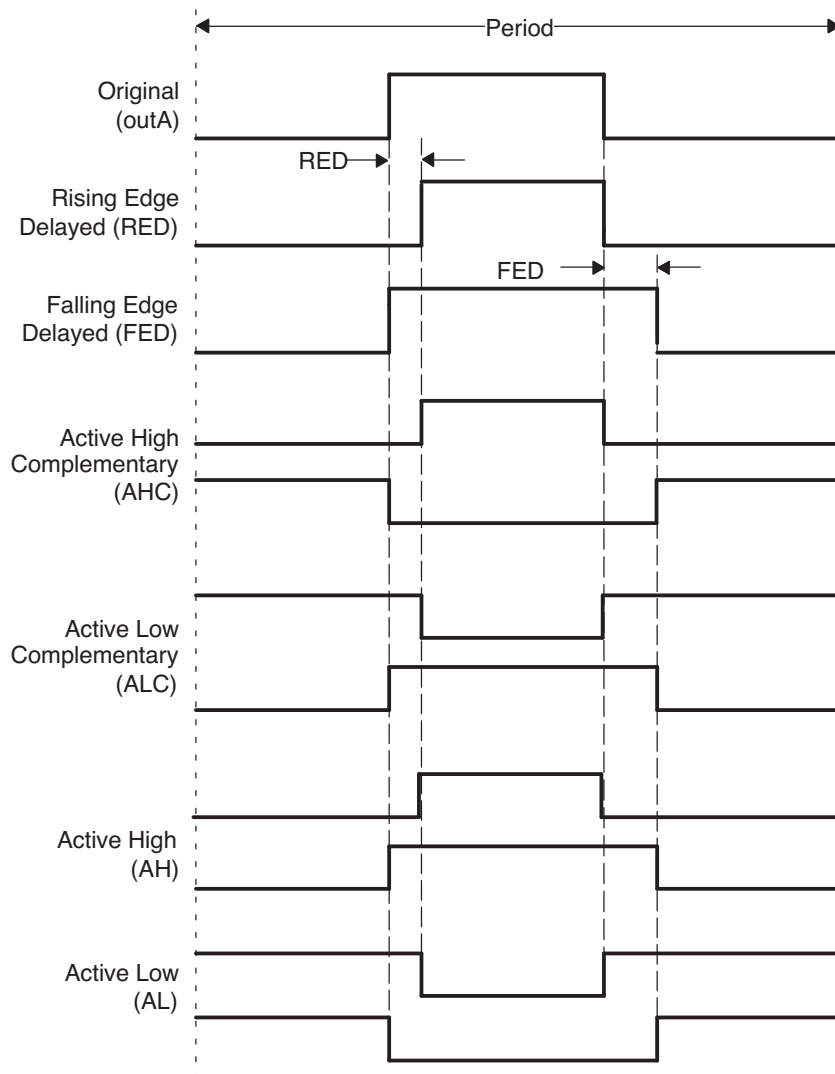
The dead-band submodule supports independent values for rising-edge (RED) and falling-edge (FED) delays. The amount of delay is programmed using the DBRED and DBFED registers. These are 10-bit registers and their value represents the number of time-base clock, TBCLK, periods a signal edge is delayed by. For example, the formula to calculate falling-edge-delay and rising-edge-delay are:

$$FED = DBFED \times T_{TBCLK}$$
$$RED = DBRED \times T_{TBCLK}$$

Where T_{TBCLK} is the period of TBCLK, the prescaled version of SYSCLKOUT.

For convenience, delay values for various TBCLK options are shown in Table 2-14.

Table 2-14. Dead-Band Delay Values in µS as a Function of DBFED and DBRED

Dead-Band Value	Dead-Band Delay in µS [1]		
DBFED, DBRED	TBCLK = SYSCLKOUT/1	TBCLK = SYSCLKOUT /2	TBCLK = SYSCLKOUT/4
1	0.01 µS	0.02 µS	0.04 µS
5	0.05 µS	0.10 µS	0.20 µS
10	0.10 µS	0.20 µS	0.40 µS
100	1.00 µS	2.00 µS	4.00 µS
200	2.00 µS	4.00 µS	8.00 µS
300	3.00 µS	6.00 µS	12.00 µS
400	4.00 µS	8.00 µS	16.00 µS
500	5.00 µS	10.00 µS	20.00 µS
600	6.00 µS	12.00 µS	24.00 µS
700	7.00 µS	14.00 µS	28.00 µS
800	8.00 µS	16.00 µS	32.00 µS
900	9.00 µS	18.00 µS	36.00 µS
1000	10.00 µS	20.00 µS	40.00 µS

[1]   Table values are calculated based on SYSCLKOUT = 100 MHz.

2.6 PWM-Chopper (PC) Submodule

Figure 2-30 illustrates the PWM-chopper (PC) submodule within the ePWM module.

Figure 2-30. PWM-Chopper Submodule
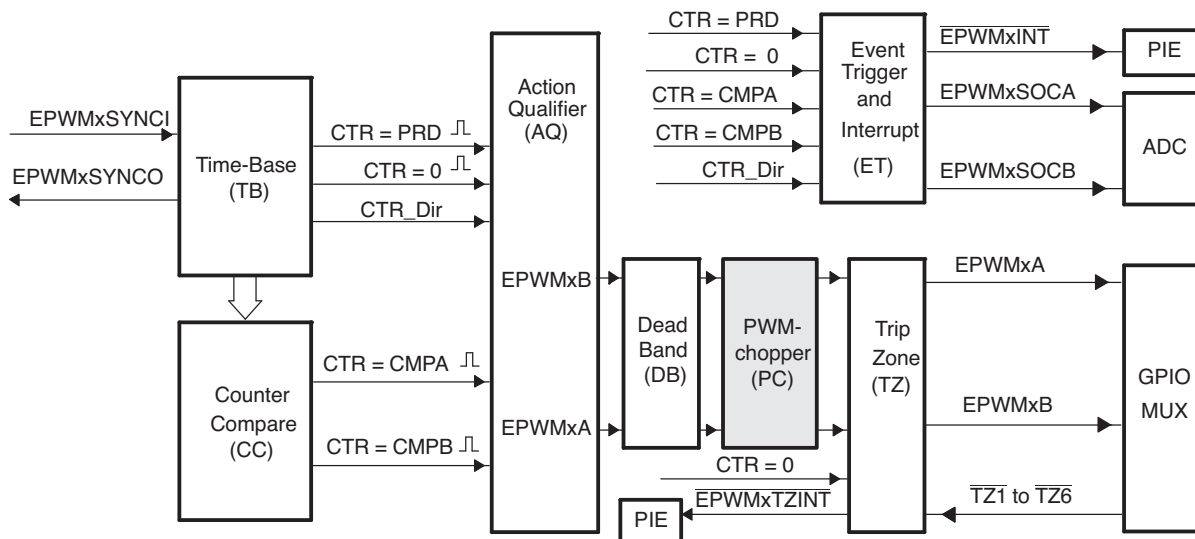
The PWM-chopper submodule allows a high-frequency carrier signal to modulate the PWM waveform generated by the action-qualifier and dead-band submodules. This capability is important if you need pulse transformer-based gate drivers to control the power switching elements.

2.6.1 Purpose of the PWM-Chopper Submodule

The key functions of the PWM-chopper submodule are:

- Programmable chopping (carrier) frequency
- Programmable pulse width of first pulse
- Programmable duty cycle of second and subsequent pulses
- Can be fully bypassed if not required

2.6.2 Controlling the PWM-Chopper Submodule

The PWM-chopper submodule operation is controlled via the registers in Table 2-15.

Table 2-15. PWM-Chopper Submodule Registers

mnemonic	Address offset	Shadowed	Description
PCCTL	0x001E	No	PWM-chopper Control Register

2.6.3 Operational Highlights for the PWM-Chopper Submodule

Figure 2-31 shows the operational details of the PWM-chopper submodule. The carrier clock is derived from SYSCLKOUT. Its frequency and duty cycle are controlled via the CHPFREQ and CHPDUTY bits in the PCCTL register. The one-shot block is a feature that provides a high energy first pulse to ensure hard and fast power switch turn on, while the subsequent pulses sustain pulses, ensuring the power switch remains on. The one-shot width is programmed via the OSHTWTH bits. The PWM-chopper submodule can be fully disabled (bypassed) via the CHPEN bit.

Figure 2-31. PWM-Chopper Submodule Operational Details
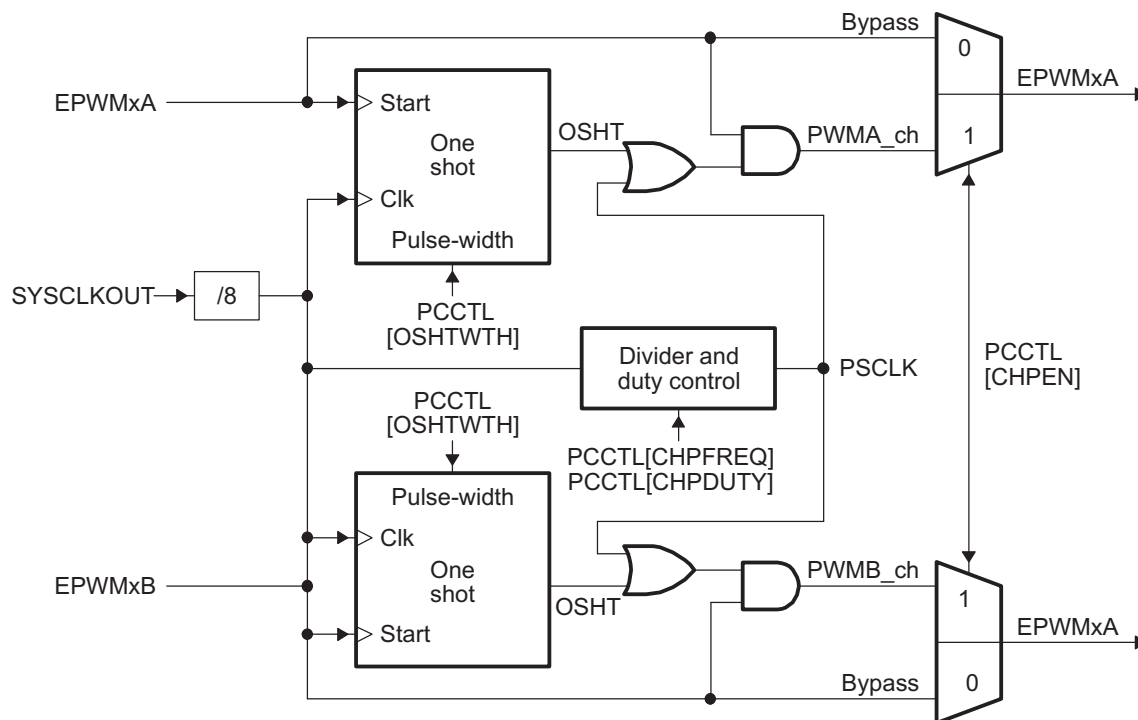
2.6.4 Waveforms

Figure 2-32 shows simplified waveforms of the chopping action only; one-shot and duty-cycle control are not shown. Details of the one-shot and duty-cycle control are discussed in the following sections.

Figure 2-32. Simple PWM-Chopper Submodule Waveforms Showing Chopping Action Only
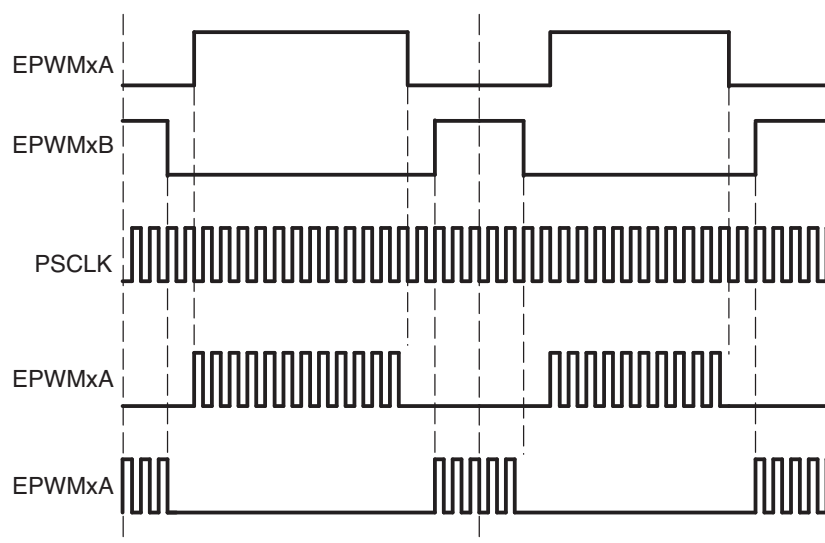
2.6.4.1 One-Shot Pulse

The width of the first pulse can be programmed to any of 16 possible pulse width values. The width or period of the first pulse is given by:

$$T_{1stpulse} = T_{SYSCLKOUT} \times 8 \times OSHTWTH$$

Where $T_{SYSCLKOUT}$ is the period of the system clock (SYSCLKOUT) and OSHTWTH is the four control bits (value from 1 to 16)

Figure 2-33 shows the first and subsequent sustaining pulses and Table 7.3 gives the possible pulse width values for a SYSCLKOUT = 100 MHz.

Figure 2-33. PWM-Chopper Submodule Waveforms Showing the First Pulse and Subsequent Sustaining Pulses
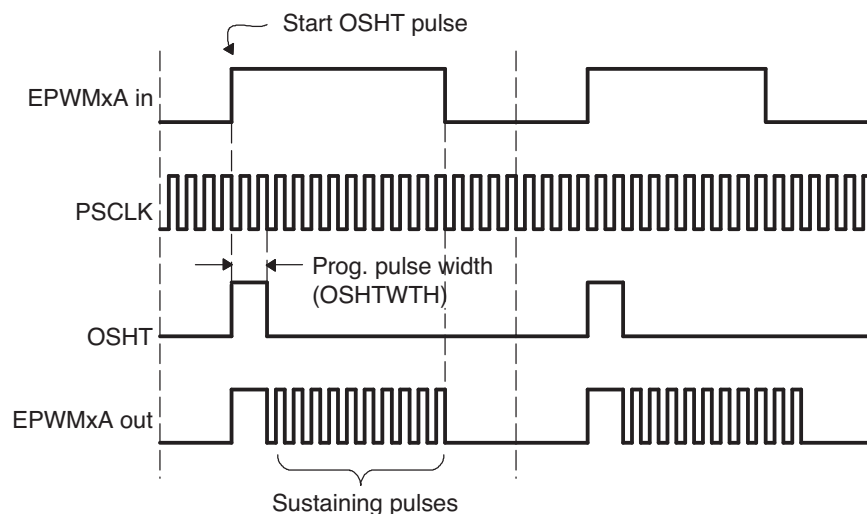
Table 2-16. Possible Pulse Width Values for SYSCLKOUT = 100 MHz

OSHTWTHz (hex)	Pulse Width (nS)
0	80
1	160
2	240
3	320
4	400
5	480
6	560
7	640
8	720
9	800
A	880
B	960
C	1040
D	1120
E	1200
F	1280

2.6.4.2 Duty Cycle Control

Pulse transformer-based gate drive designs need to comprehend the magnetic properties or characteristics of the transformer and associated circuitry. Saturation is one such consideration. To assist the gate drive designer, the duty cycles of the second and subsequent pulses have been made programmable. These sustaining pulses ensure the correct drive strength and polarity is maintained on the power switch gate during the on period, and hence a programmable duty cycle allows a design to be tuned or optimized via software control.

Figure 2-34 shows the duty cycle control that is possible by programming the CHPDUTY bits. One of seven possible duty ratios can be selected ranging from 12.5% to 87.5%.

Figure 2-34. PWM-Chopper Submodule Waveforms Showing the Pulse Width (Duty Cycle) Control of Sustaining Pulses
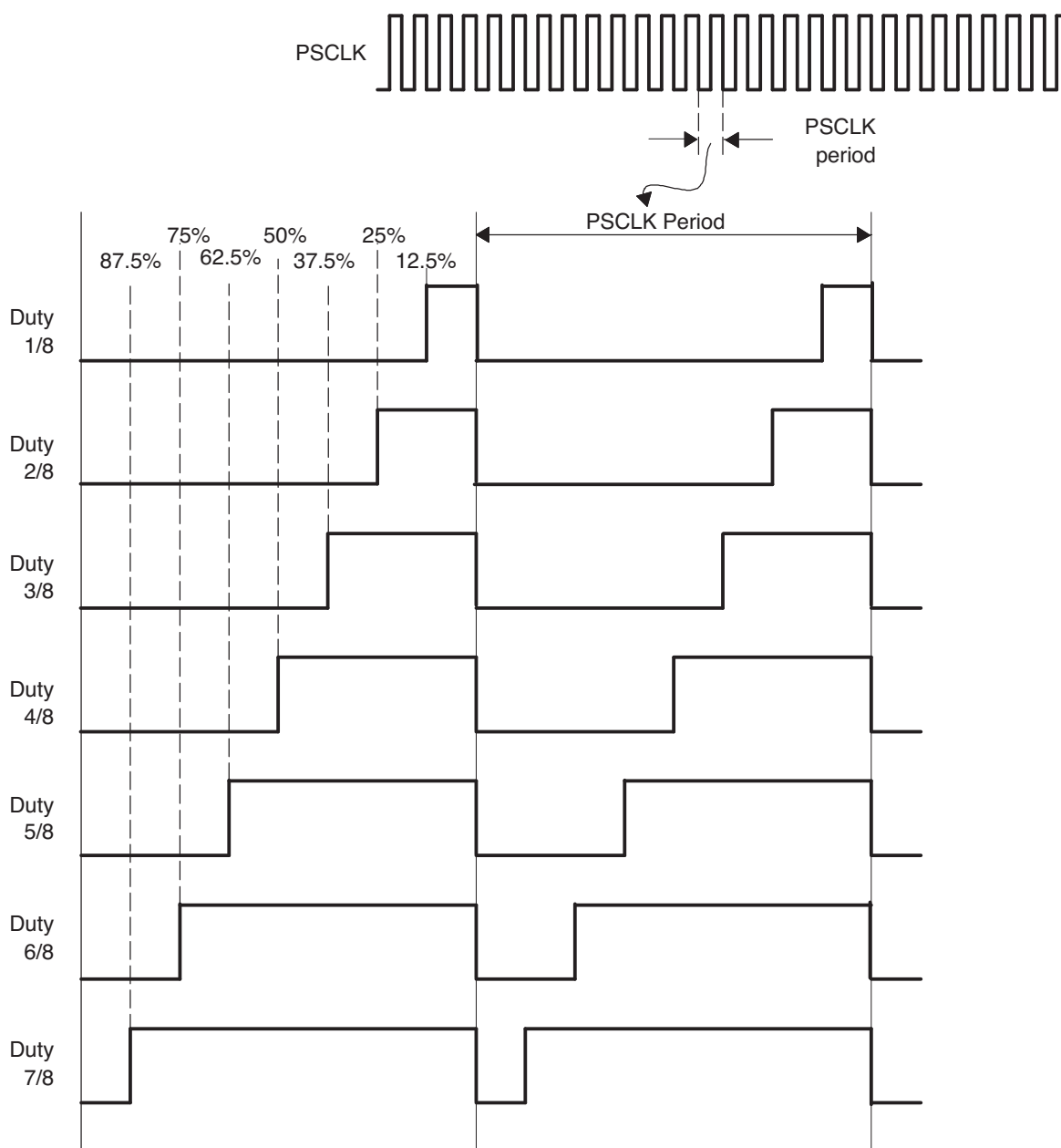
2.7 Trip-Zone (TZ) Submodule

Figure 2-35 shows how the trip-zone (TZ) submodule fits within the ePWM module.

Figure 2-35. Trip-Zone Submodule
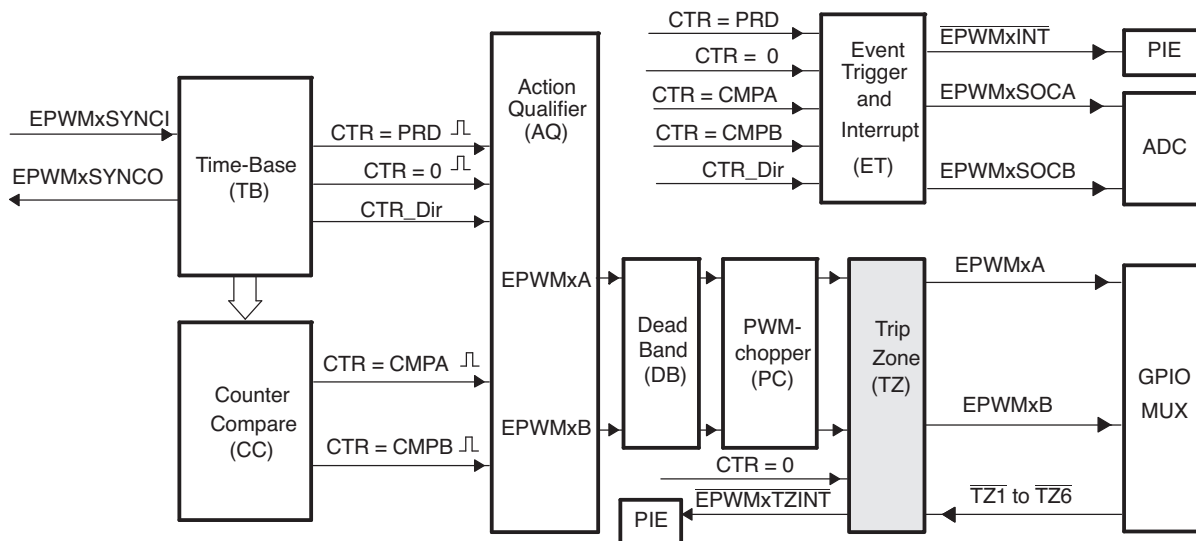
Each ePWM module is connected to six $\overline{TZn}$ signals ($\overline{TZ1}$ to $\overline{TZ6}$) that are sourced from the GPIO MUX. These signals indicate external fault or trip conditions, and the ePWM outputs can be programmed to respond accordingly when faults occur.

2.7.1 Purpose of the Trip-Zone Submodule

The key functions of the Trip-Zone submodule are:

*   Trip inputs $\overline{TZ1}$ to $\overline{TZ6}$ can be flexibly mapped to any ePWM module.
*   Upon a fault condition, outputs EPWMxA and EPWMxB can be forced to one of the following:
    *   High
    *   Low
    *   High-impedance
    *   No action taken
*   Support for one-shot trip (OSHT) for major short circuits or over-current conditions.
*   Support for cycle-by-cycle tripping (CBC) for current limiting operation.
*   Each trip-zone input pin can be allocated to either one-shot or cycle-by-cycle operation.
*   Interrupt generation is possible on any trip-zone pin.
*   Software-forced tripping is also supported.
*   The trip-zone submodule can be fully bypassed if it is not required.

2.7.2 Controlling and Monitoring the Trip-Zone Submodule

The trip-zone submodule operation is controlled and monitored through the following registers:

Table 2-17. Trip-Zone Submodule Registers

Register Name	Address offset	Shadowed	Description [1]
TZSEL	0x0012	No	Trip-Zone Select Register
reserved	0x0013		
TZCTL	0x0014	No	Trip-Zone Control Register
TZEINT	0x0015	No	Trip-Zone Enable Interrupt Register
TZFLG	0x0016	No	Trip-Zone Flag Register
TZCLR	0x0017	No	Trip-Zone Clear Register
TZFRC	0x0018	No	Trip-Zone Force Register

[1] All trip-zone registers are EALLOW protected and can be modified only after executing the EALLOW instruction. For more information, see the device-specific version of the System Control and Interrupts Reference Guide listed in Section 1.

2.7.3 Operational Highlights for the Trip-Zone Submodule

The following sections describe the operational highlights and configuration options for the trip-zone submodule.

The trip-zone signals at pins $\overline{TZ1}$ to $\overline{TZ6}$ (also collectively referred to as $\overline{TZn}$) are active low input signals. When one of these pins goes low, it indicates that a trip event has occurred. Each ePWM module can be individually configured to ignore or use each of the trip-zone pins. Which trip-zone pins are used by a particular ePWM module is determined by the TZSEL register for that specific ePWM module. The trip-zone signals may or may not be synchronized to the system clock (SYSCLKOUT) and digitally filtered within the GPIO MUX block. A minimum 1 SYSCLKOUT low pulse on $\overline{TZn}$ inputs is sufficient to trigger a fault condition in the ePWM module. The asynchronous trip makes sure that if clocks are missing for any reason, the outputs can still be tripped by a valid event present on $\overline{TZn}$ inputs, providing the GPIO is appropriately configured. For more information, see the GPIO section of the specific device version of the *System Control and Interrupts Reference Guide* listed in Section 1.

Each $\overline{TZn}$ input can be individually configured to provide either a cycle-by-cycle or one-shot trip event for a ePWM module. The configuration is determined by the TZSEL[CBCn] and TZSEL[OSHTn] control bits (where n corresponds to the trip pin) respectively.

- **Cycle-by-Cycle (CBC):**

  When a cycle-by-cycle trip event occurs, the action specified in the TZCTL register is carried out immediately on the EPWMxA and/or EPWMxB output. Table 2-18 lists the possible actions. In addition, the cycle-by-cycle trip event flag (TZFLG[CBC]) is set and a EPWMx_TZINT interrupt is generated if it is enabled in the TZEINT register and PIE peripheral.

  The specified condition on the pins is automatically cleared when the ePWM time-base counter reaches zero (TBCTR = 0x0000) if the trip event is no longer present. Therefore, in this mode, the trip event is cleared or reset every PWM cycle. The TZFLG[CBC] flag bit will remain set until it is manually cleared by writing to the TZCLR[CBC] bit. If the cycle-by-cycle trip event is still present when the TZFLG[CBC] bit is cleared, then it will again be immediately set.

- **One-Shot (OSHT):**

  When a one-shot trip event occurs, the action specified in the TZCTL register is carried out immediately on the EPWMxA and/or EPWMxB output. Table 2-18 lists the possible actions. In addition, the one-shot trip event flag (TZFLG[OST]) is set and a EPWMx_TZINT interrupt is generated if it is enabled in the TZEINT register and PIE peripheral. The one-shot trip condition must be cleared manually by writing to the TZCLR[OST] bit.

The action taken when a trip event occurs can be configured individually for each of the ePWM output pins by way of the TZCTL[TZA] and TZCTL[TZB] register bits. One of four possible actions, shown in Table 2-18, can be taken on a trip event.

Table 2-18. Possible Actions On a Trip Event

TZCTL[TZA] and/or TZCTL[TZB]	EPWMxA and/or EPWMxB	Comment
0,0	High-Impedance	Tripped
0,1	Force to High State	Tripped
1,0	Force to Low State	Tripped
1,1	No Change	Do Nothing. No change is made to the output.

Example 2-8. Trip-Zone Configurations

Scenario A:

A one-shot trip event on $\overline{TZ1}$ pulls both EPWM1A, EPWM1B low and also forces EPWM2A and EPWM2B high.

- Configure the ePWM1 registers as follows:
    - TZSEL[OSHT1] = 1: enables $\overline{TZ1}$ as a one-shot event source for ePWM1
    - TZCTL[TZA] = 2: EPWM1A will be forced low on a trip event.
    - TZCTL[TZB] = 2: EPWM1B will be forced low on a trip event.
- Configure the ePWM2 registers as follows:
    - TZSEL[OSHT1] = 1: enables $\overline{TZ1}$ as a one-shot event source for ePWM2
    - TZCTL[TZA] = 1: EPWM2A will be forced high on a trip event.
    - TZCTL[TZB] = 1: EPWM2B will be forced high on a trip event.

Scenario B:

A cycle-by-cycle event on $\overline{TZ5}$ pulls both EPWM1A, EPWM1B low.
A one-shot event on $\overline{TZ1}$ or $\overline{TZ6}$ puts EPWM2A into a high impedance state.

- Configure the ePWM1 registers as follows:
    - TZSEL[CBC5] = 1: enables $\overline{TZ5}$ as a one-shot event source for ePWM1
    - TZCTL[TZA] = 2: EPWM1A will be forced low on a trip event.
    - TZCTL[TZB] = 2: EPWM1B will be forced low on a trip event.
- Configure the ePWM2 registers as follows:
    - TZSEL[OSHT1] = 1: enables $\overline{TZ1}$ as a one-shot event source for ePWM2
    - TZSEL[OSHT6] = 1: enables $\overline{TZ6}$ as a one-shot event source for ePWM1
    - TZCTL[TZA] = 0: EPWM1A will be put into a high-impedance state on a trip event.
    - TZCTL[TZB] = 3: EPWM1B will ignore the trip event.

2.7.4 Generating Trip Event Interrupts

Figure 2-36 and Figure 2-37 illustrate the trip-zone submodule control and interrupt logic, respectively.

Figure 2-36. Trip-Zone Submodule Mode Control Logic
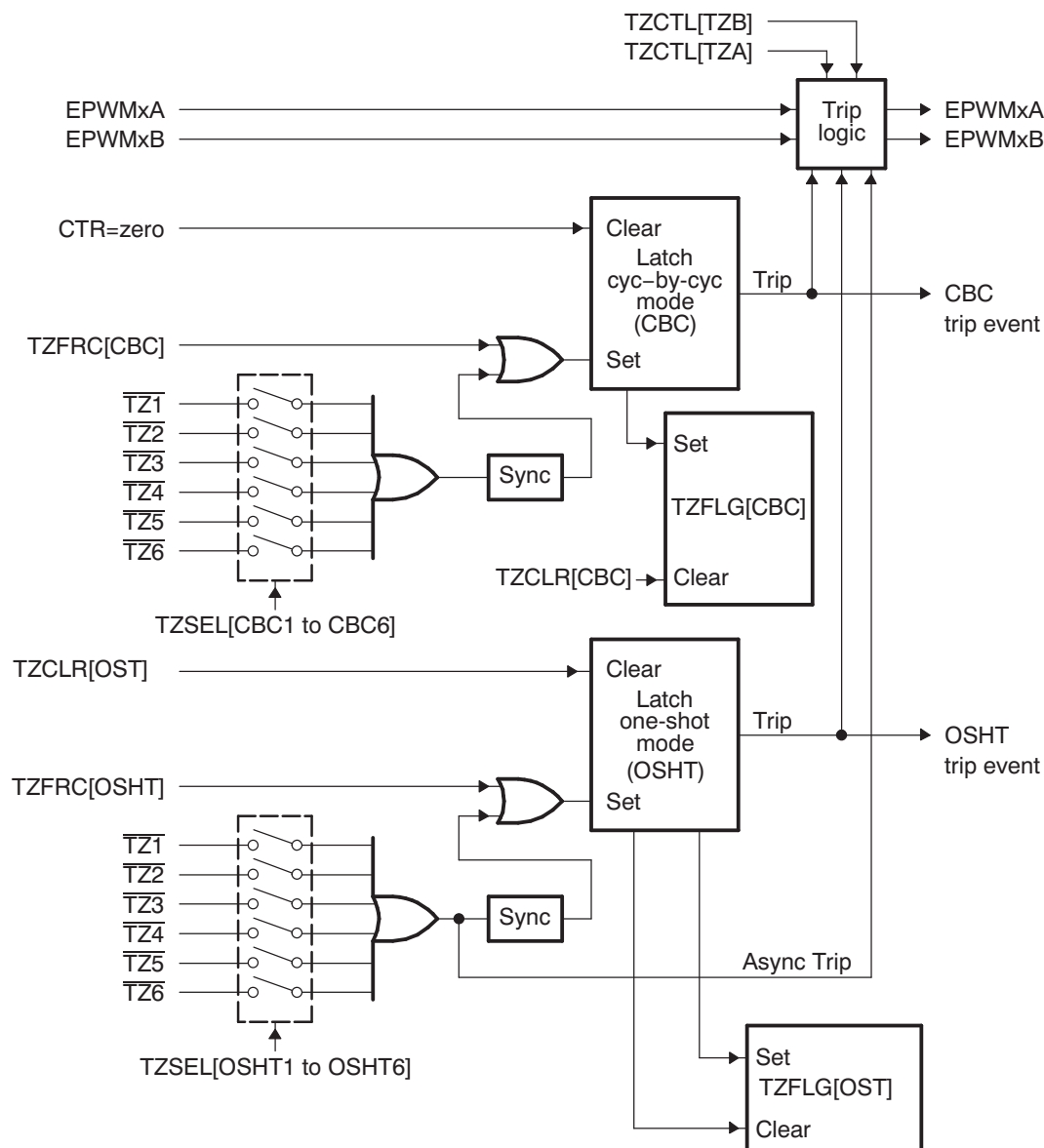
Figure 2-37. Trip-Zone Submodule Interrupt Logic
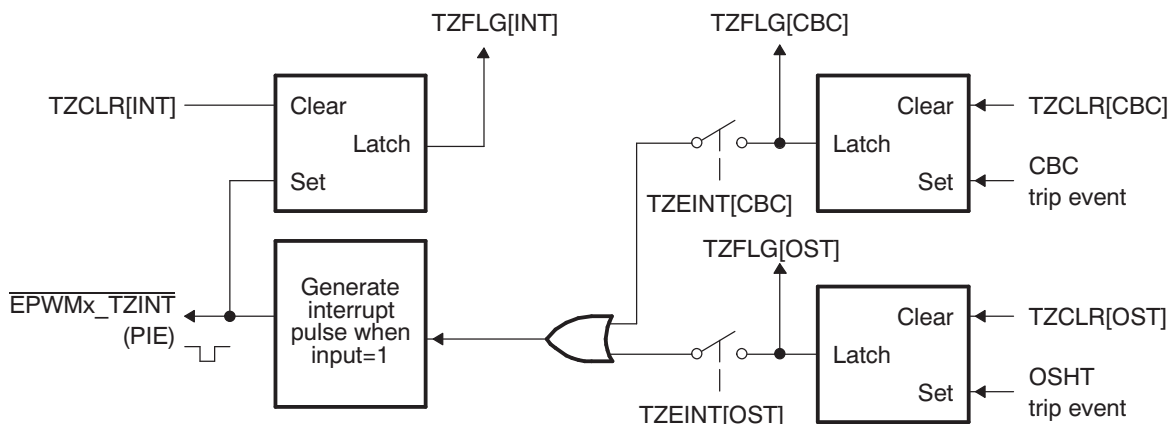
2.8 Event-Trigger (ET) Submodule

The key functions of the event-trigger submodule are:

- Receives event inputs generated by the time-base and counter-compare submodules
- Uses the time-base direction information for up/down event qualification
- Uses prescaling logic to issue interrupt requests and ADC start of conversion at:
  - Every event
  - Every second event
  - Every third event
- Provides full visibility of event generation via event counters and flags
- Allows software forcing of Interrupts and ADC start of conversion

The event-trigger submodule manages the events generated by the time-base submodule and the counter-compare submodule to generate an interrupt to the CPU and/or a start of conversion pulse to the ADC when a selected event occurs. Figure 2-38 illustrates where the event-trigger submodule fits within the ePWM system.

Figure 2-38. Event-Trigger Submodule
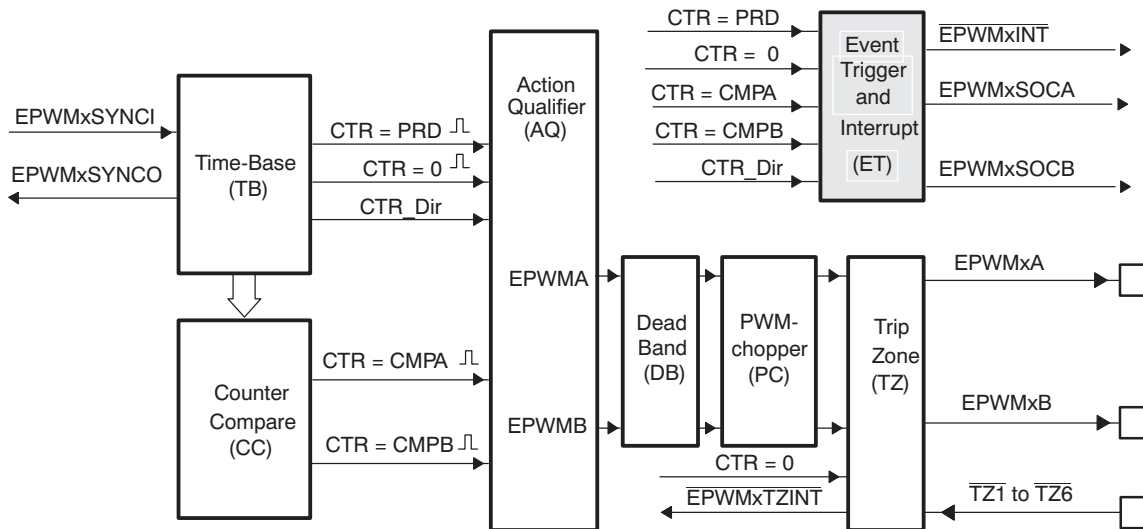
2.8.1 Operational Overview of the Event-Trigger Submodule

The following sections describe the event-trigger submodule's operational highlights.

Each ePWM module has one interrupt request line connected to the PIE and two start of conversion signals (one for each sequencer) connected to the ADC module. As shown in Figure 2-39, ADC start of conversion for all ePWM modules are ORed together and hence multiple modules can initiate an ADC start of conversion. If two requests occur on one start of conversion line, then only one will be recognized by the ADC.

Figure 2-39. Event-Trigger Submodule Inter-Connectivity of ADC Start of Conversion and Interrupt Signals
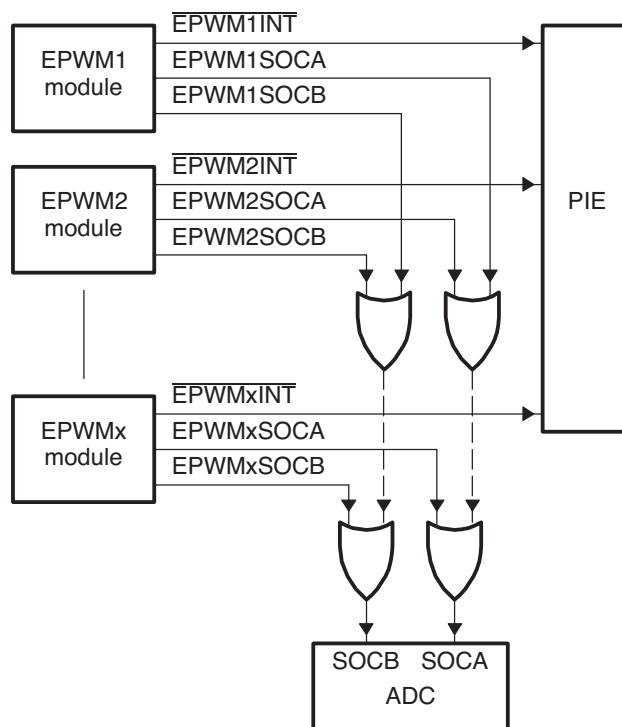
The event-trigger submodule monitors various event conditions (the left side inputs to event-trigger submodule shown in Figure 2-40) and can be configured to prescale these events before issuing an Interrupt request or an ADC start of conversion. The event-trigger prescaling logic can issue Interrupt requests and ADC start of conversion at:

- Every event
- Every second event
- Every third event

Figure 2-40. Event-Trigger Submodule Showing Event Inputs and Prescaled Outputs
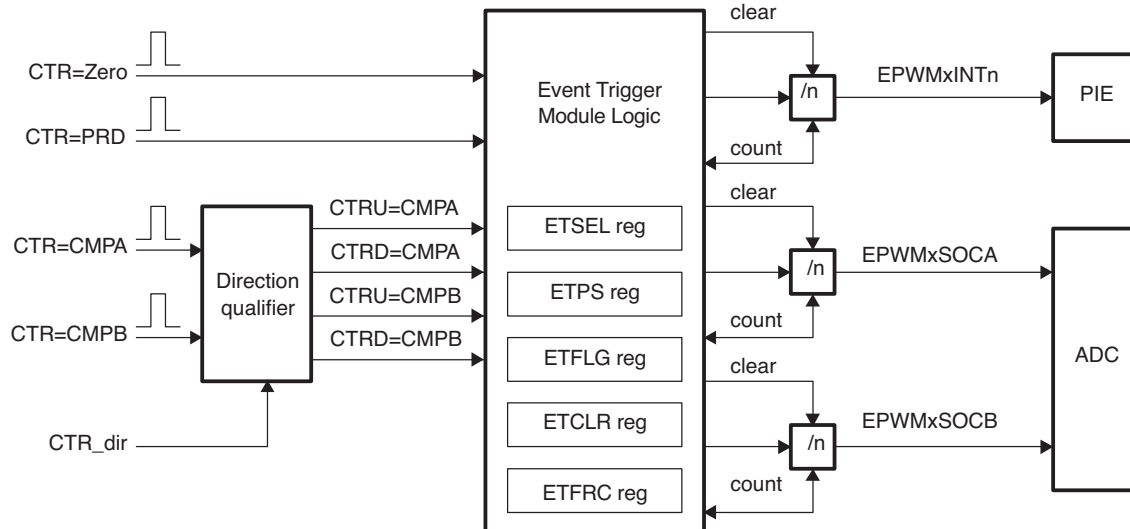
The key registers used to configure the event-trigger submodule are shown in Table 2-19:

Table 2-19. Event-Trigger Submodule Registers

Register Name	Address offset	Shadowed	Description
ETSEL	0x0019	No	Event-trigger Selection Register
ETPS	0x001A	No	Event-trigger Prescale Register
ETFLG	0x001B	No	Event-trigger Flag Register
ETCLR	0x001C	No	Event-trigger Clear Register
ETFRC	0x001D	No	Event-trigger Force Register

- ETSEL—This selects which of the possible events will trigger an interrupt or start an ADC conversion
- ETPS—This programs the event prescaling options mentioned above.
- ETFLG—These are flag bits indicating status of the selected and prescaled events.
- ETCLR—These bits allow you to clear the flag bits in the ETFLG register via software.
- ETFRC—These bits allow software forcing of an event. Useful for debugging or s/w intervention.

A more detailed look at how the various register bits interact with the Interrupt and ADC start of conversion logic are shown in Figure 2-41, Figure 2-42, and Figure 2-43.

Figure 2-41 shows the event-trigger's interrupt generation logic. The interrupt-period (ETPS[INTPRD]) bits specify the number of events required to cause an interrupt pulse to be generated. The choices available are:
- Do not generate an interrupt.
- Generate an interrupt on every event
- Generate an interrupt on every second event
- Generate an interrupt on every very third event

Which event can cause an interrupt is configured by the interrupt selection (ETSEL[INTSEL]) bits. The event can be one of the following:
- Time-base counter equal to zero (TBCTR = 0x0000).
- Time-base counter equal to period (TBCTR = TBPRD).
- Time-base counter equal to the compare A register (CMPA) when the timer is incrementing.
- Time-base counter equal to the compare A register (CMPA) when the timer is decrementing.
- Time-base counter equal to the compare B register (CMPB) when the timer is incrementing.
- Time-base counter equal to the compare B register (CMPB) when the timer is decrementing.

The number of events that have occurred can be read from the interrupt event counter (ETPS[INTCNT]) register bits. That is, when the specified event occurs the ETPS[INTCNT] bits are incremented until they reach the value specified by ETPS[INTPRD]. When ETPS[INTCNT] = ETPS[INTPRD] the counter stops counting and its output is set. The counter is only cleared when an interrupt is sent to the PIE.

When ETPS[INTCNT] reaches ETPS[INTPRD] the one of the following behaviors will occur:

- If interrupts are enabled, ETSEL[INTEN] = 1 and the interrupt flag is clear, ETFLG[INT] = 0, then an interrupt pulse is generated and the interrupt flag is set, ETFLG[INT] = 1, and the event counter is cleared ETPS[INTCNT] = 0. The counter will begin counting events again.
- If interrupts are disabled, ETSEL[INTEN] = 0, or the interrupt flag is set, ETFLG[INT] = 1, the counter stops counting events when it reaches the period value ETPS[INTCNT] = ETPS[INTPRD].
- If interrupts are enabled, but the interrupt flag is already set, then the counter will hold its output high until the ENTFLG[INT] flag is cleared. This allows for one interrupt to be pending while one is serviced.

Writing to the INTPRD bits will automatically clear the counter INTCNT = 0 and the counter output will be reset (so no interrupts are generated). Writing a 1 to the ETFRC[INT] bit will increment the event counter INTCNT. The counter will behave as described above when INTCNT = INTPRD. When INTPRD = 0, the counter is disabled and hence no events will be detected and the ETFRC[INT] bit is also ignored.

The above definition means that you can generate an interrupt on every event, on every second event, or on every third event. An interrupt cannot be generated on every fourth or more events.

Figure 2-41. Event-Trigger Interrupt Generator
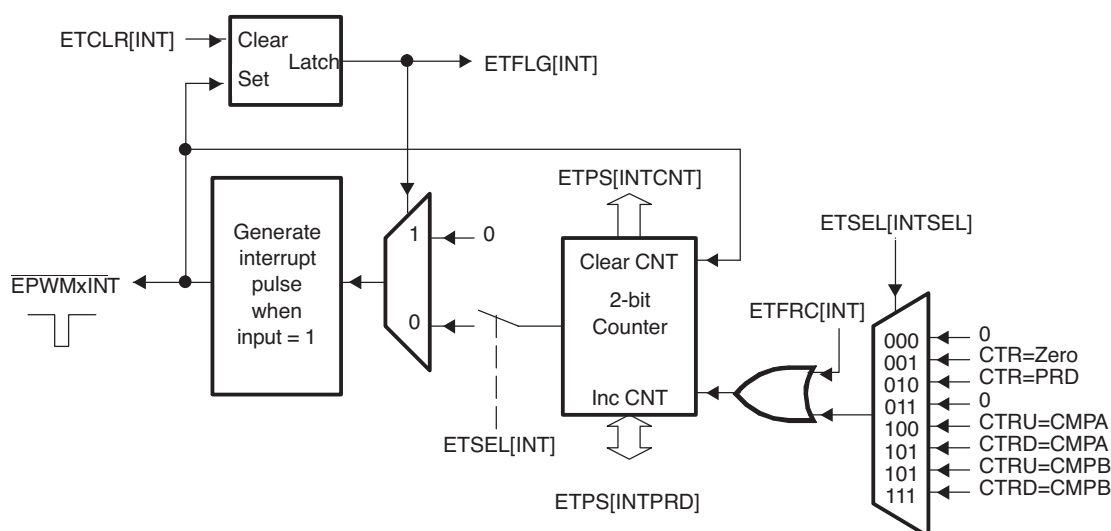
Figure 2-42 shows the operation of the event-trigger's start-of-conversion-A (SOCA) pulse generator. The ETPS[SOCACNT] counter and ETPS[SOCAPRD] period values behave similarly to the interrupt generator except that the pulses are continuously generated. That is, the pulse flag ETFLG[SOCA] is latched when a pulse is generated, but it does not stop further pulse generation. The enable/disable bit ETSEL[SOCAEN] stops pulse generation, but input events can still be counted until the period value is reached as with the interrupt generation logic. The event that will trigger an SOCA and SOCB pulse can be configured separately in the ETSEL[SOCASEL] and ETSEL[SOCBSEL] bits. The possible events are the same events that can be specified for the interrupt generation logic.

Figure 2-42. Event-Trigger SOCA Pulse Generator
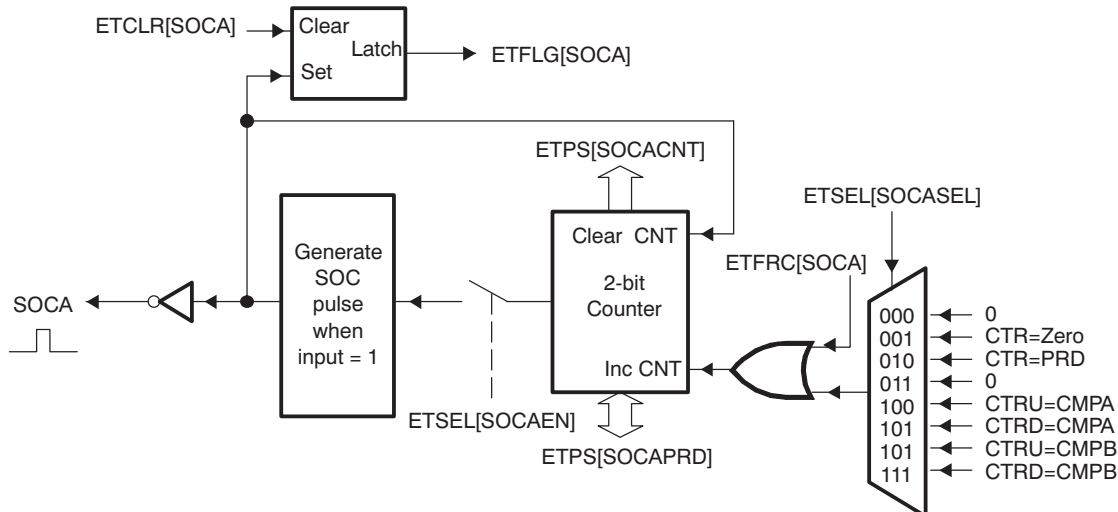
[Figure 2-43](#) shows the operation of the event-trigger's start-of-conversion-B (SOCB) pulse generator. The event-trigger's SOCB pulse generator operates the same way as the SOCA.

Figure 2-43. Event-Trigger SOCB Pulse Generator
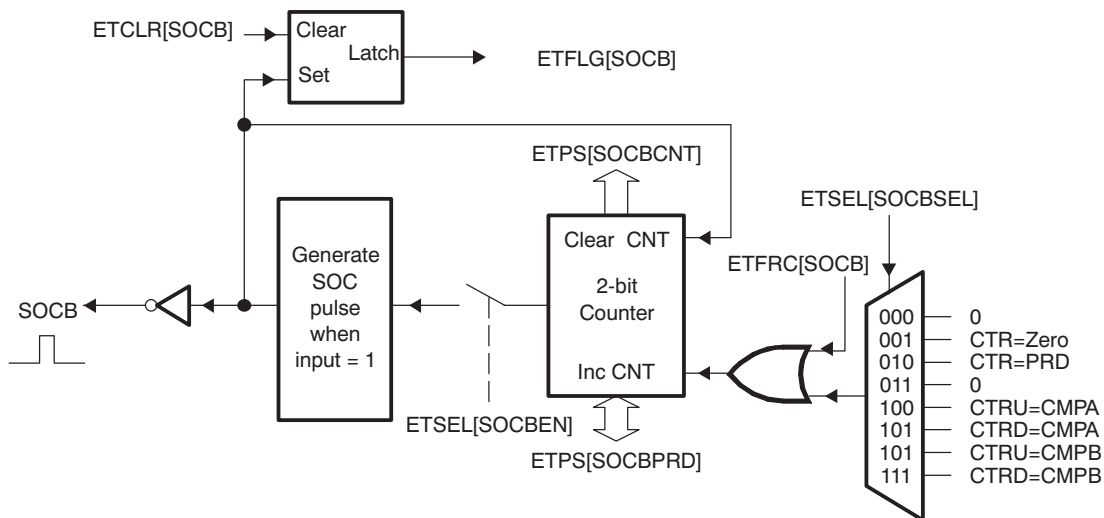
Applications to Power Topologies

An ePWM module has all the local resources necessary to operate completely as a standalone module or to operate in synchronization with other identical ePWM modules.

3.1 Overview of Multiple Modules

Previously in this user's guide, all discussions have described the operation of a single module. To facilitate the understanding of multiple modules working together in a system, the ePWM module described in reference is represented by the more simplified block diagram shown in Figure 3-1. This simplified ePWM block shows only the key resources needed to explain how a multiswitch power topology is controlled with multiple ePWM modules working together.

Figure 3-1. Simplified ePWM Module
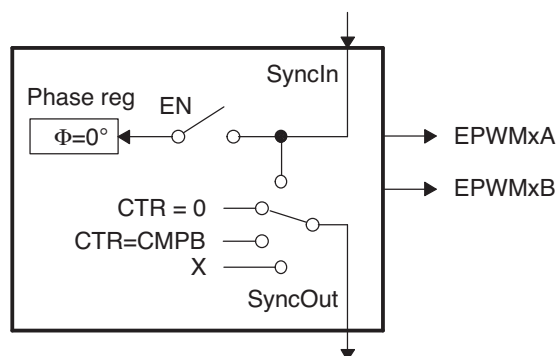
3.2 Key Configuration Capabilities

The key configuration choices available to each module are as follows:

- Options for SyncIn
  - Load own counter with phase register on an incoming sync strobe—enable (EN) switch closed
  - Do nothing or ignore incoming sync strobe—enable switch open
  - Sync flow-through - SyncOut connected to SyncIn
  - Master mode, provides a sync at PWM boundaries—SyncOut connected to CTR = PRD
  - Master mode, provides a sync at any programmable point in time—SyncOut connected to CTR = CMPB
  - Module is in standalone mode and provides No sync to other modules—SyncOut connected to X (disabled)
- Options for SyncOut
  - Sync flow-through - SyncOut connected to SyncIn
  - Master mode, provides a sync at PWM boundaries—SyncOut connected to CTR = PRD
  - Master mode, provides a sync at any programmable point in time—SyncOut connected to CTR = CMPB
  - Module is in standalone mode and provides No sync to other modules—SyncOut connected to X (disabled)

For each choice of SyncOut, a module may also choose to load its own counter with a new phase value on a SyncIn strobe input or choose to ignore it, i.e., via the enable switch. Although various combinations are possible, the two most common—master module and slave module modes—are shown in Figure 3-2.

Figure 3-2. EPWM1 Configured as a Typical Master, EPWM2 Configured as a Slave
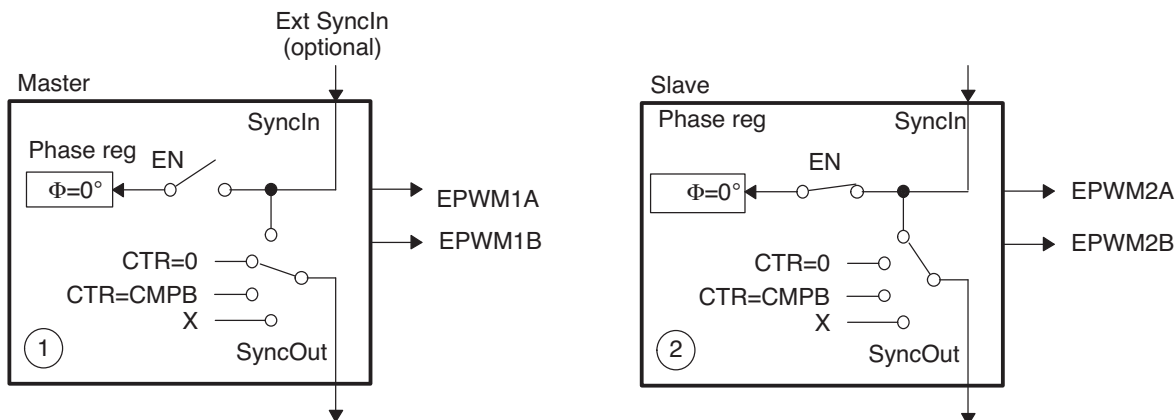
3.3 Controlling Multiple Buck Converters With Independent Frequencies

One of the simplest power converter topologies is the buck. A single ePWM module configured as a master can control two buck stages with the same PWM frequency. If independent frequency control is required for each buck converter, then one ePWM module must be allocated for each converter stage. Figure 3-3 shows four buck stages, each running at independent frequencies. In this case, all four ePWM modules are configured as Masters and no synchronization is used. Figure 3-4 shows the waveforms generated by the setup shown in Figure 3-3; note that only three waveforms are shown, although there are four stages.

Figure 3-3. Control of Four Buck Stages. Here $F_{PWM1} \neq F_{PWM2} \neq F_{PWM3} \neq F_{PWM4}$
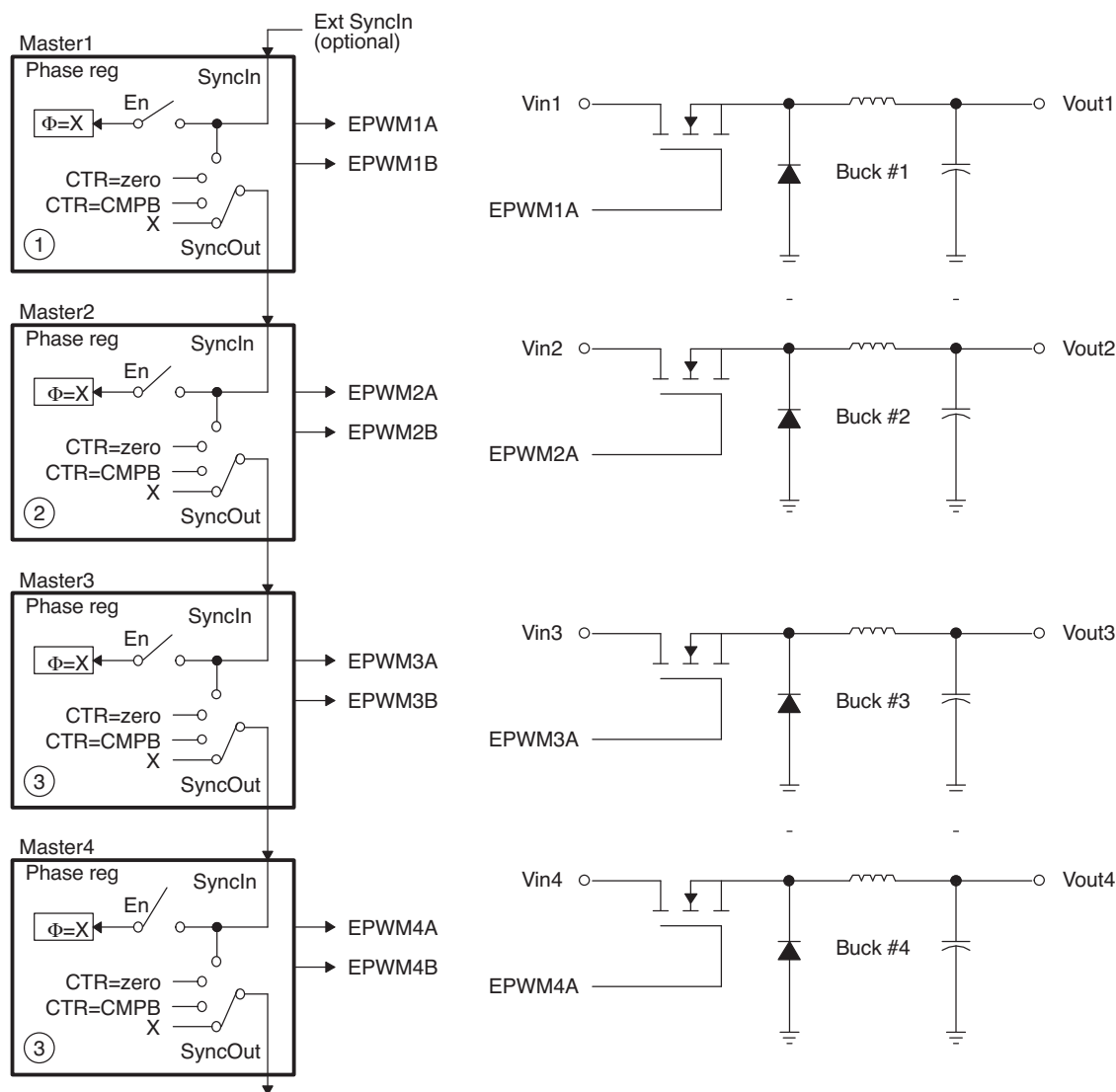
NOTE: $\Theta = X$ indicates value in phase register is a "don't care"

Figure 3-4. Buck Waveforms for Figure 3-3 (Note: Only three bucks shown here)
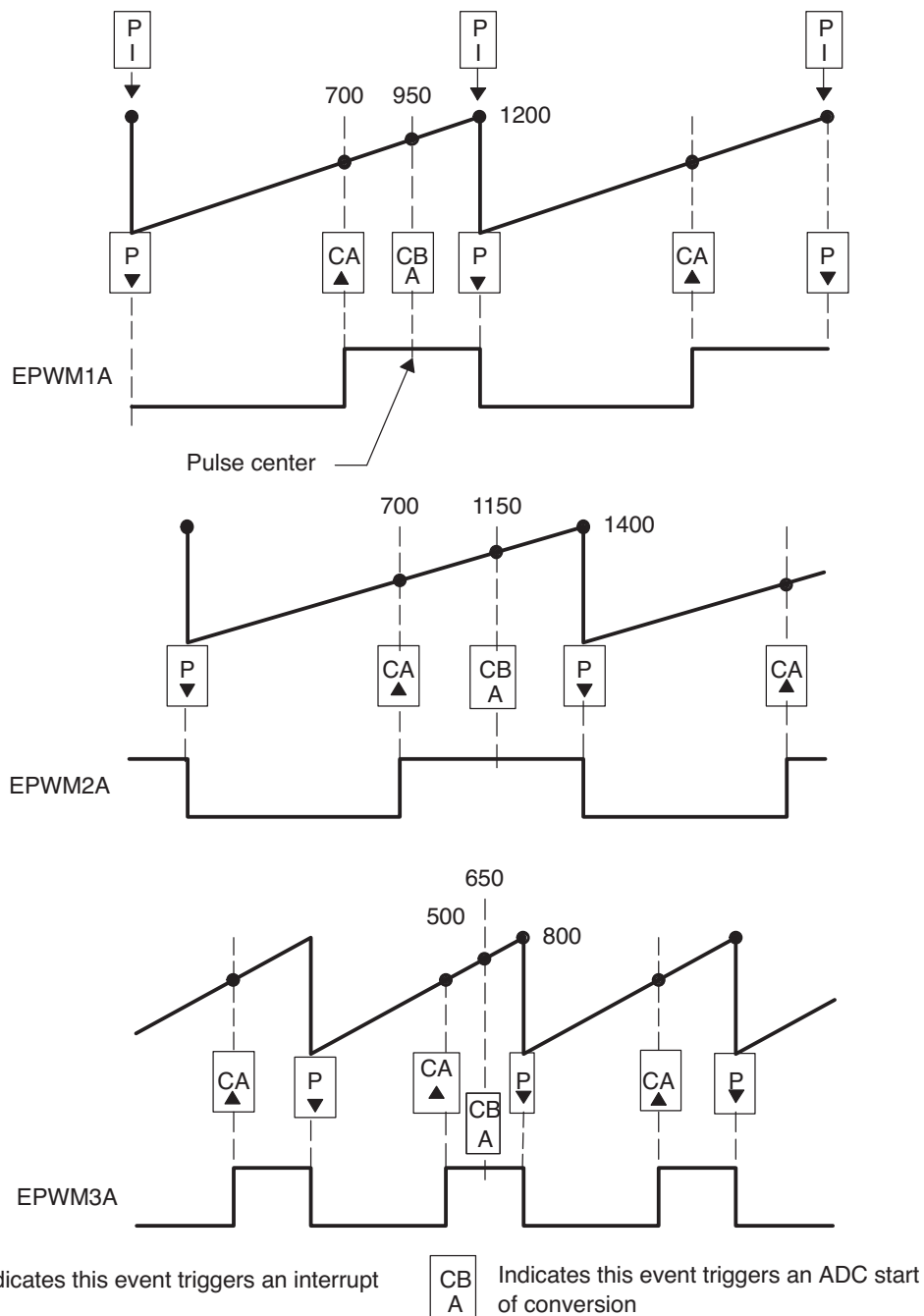
Example 3-1. Configuration for Example in Figure 3-4

```
//=====================================================================
//  (Note: code for only 3 modules shown)
// Initialization Time
//=======================
// EPWM Module 1 config
    EPwm1Regs.TBPRD = 1200;                     // Period = 1201 TBCLK counts
    EPwm1Regs.TBPHS = 0;                        // Set Phase register to zero
    EPwm1Regs.TBCTL.bit.CTRMODE = TB_COUNT_UP;  // Asymmetrical mode
    EPwm1Regs.TBCTL.bit.PHSEN = TB_DISABLE;     // Phase loading disabled
    EPwm1Regs.TBCTL.bit.PRDLD = TB_SHADOW;
    EPwm1Regs.TBCTL.bit.SYNCOSEL = TB_SYNC_DISABLE;
    EPwm1Regs.CMPCTL.bit.SHDWAMODE = CC_SHADOW;
    EPwm1Regs.CMPCTL.bit.SHDWBMODE = CC_SHADOW;
    EPwm1Regs.CMPCTL.bit.LOADAMODE = CC_CTR_ZERO; // load on CTR=Zero
    EPwm1Regs.CMPCTL.bit.LOADBMODE = CC_CTR_ZERO; // load on CTR=Zero
    EPwm1Regs.AQCTLA.bit.PRD = AQ_CLEAR;
    EPwm1Regs.AQCTLA.bit.CAU = AQ_SET;
// EPWM Module 2 config
    EPwm2Regs.TBPRD = 1400;                     // Period = 1401 TBCLK counts
    EPwm2Regs.TBPHS = 0;                        // Set Phase register to zero
    EPwm2Regs.TBCTL.bit.CTRMODE = TB_COUNT_UP;  // Asymmetrical mode
    EPwm2Regs.TBCTL.bit.PHSEN = TB_DISABLE;     // Phase loading disabled
    EPwm2Regs.TBCTL.bit.PRDLD = TB_SHADOW;
    EPwm2Regs.TBCTL.bit.SYNCOSEL = TB_SYNC_DISABLE;
    EPwm2Regs.CMPCTL.bit.SHDWAMODE = CC_SHADOW;
    EPwm2Regs.CMPCTL.bit.SHDWBMODE = CC_SHADOW;
    EPwm2Regs.CMPCTL.bit.LOADAMODE = CC_CTR_ZERO; // load on CTR=Zero
    EPwm2Regs.CMPCTL.bit.LOADBMODE = CC_CTR_ZERO; // load on CTR=Zero
    EPwm2Regs.AQCTLA.bit.PRD = AQ_CLEAR;
    EPwm2Regs.AQCTLA.bit.CAU = AQ_SET;
// EPWM Module 3 config
    EPwm3Regs.TBPRD = 800;                      // Period = 801 TBCLK counts
    EPwm3Regs.TBPHS = 0;                        // Set Phase register to zero
    EPwm3Regs.TBCTL.bit.CTRMODE = TB_COUNT_UP;
    EPwm3Regs.TBCTL.bit.PHSEN = TB_DISABLE;     // Phase loading disabled
    EPwm3Regs.TBCTL.bit.PRDLD = TB_SHADOW;
    EPwm3Regs.TBCTL.bit.SYNCOSEL = TB_SYNC_DISABLE;
    EPwm3Regs.CMPCTL.bit.SHDWAMODE = CC_SHADOW;
    EPwm3Regs.CMPCTL.bit.SHDWBMODE = CC_SHADOW;
    EPwm3Regs.CMPCTL.bit.LOADAMODE = CC_CTR_ZERO; // load on CTR=Zero
    EPwm3Regs.CMPCTL.bit.LOADBMODE = CC_CTR_ZERO; // load on CTR=Zero
    EPwm3Regs.AQCTLA.bit.PRD = AQ_CLEAR;
    EPwm3Regs.AQCTLA.bit.CAU = AQ_SET;
//
// Run Time (Note: Example execution of one run-time instant)
//=========================================================
    EPwm1Regs.CMPA.half.CMPA = 700;             // adjust duty for output EPWM1A
    EPwm2Regs.CMPA.half.CMPA = 700;             // adjust duty for output EPWM2A
    EPwm3Regs.CMPA.half.CMPA = 500;             // adjust duty for output EPWM3A
```

3.4   Controlling Multiple Buck Converters With Same Frequencies

If synchronization is a requirement, ePWM module 2 can be configured as a slave and can operate at integer multiple (N) frequencies of module 1. The sync signal from master to slave ensures these modules remain locked. Figure 3-5 shows such a configuration; Figure 3-6 shows the waveforms generated by the configuration.

Figure 3-5. Control of Four Buck Stages. (Note: $F_{PWM2} = N \times F_{PWM1}$)
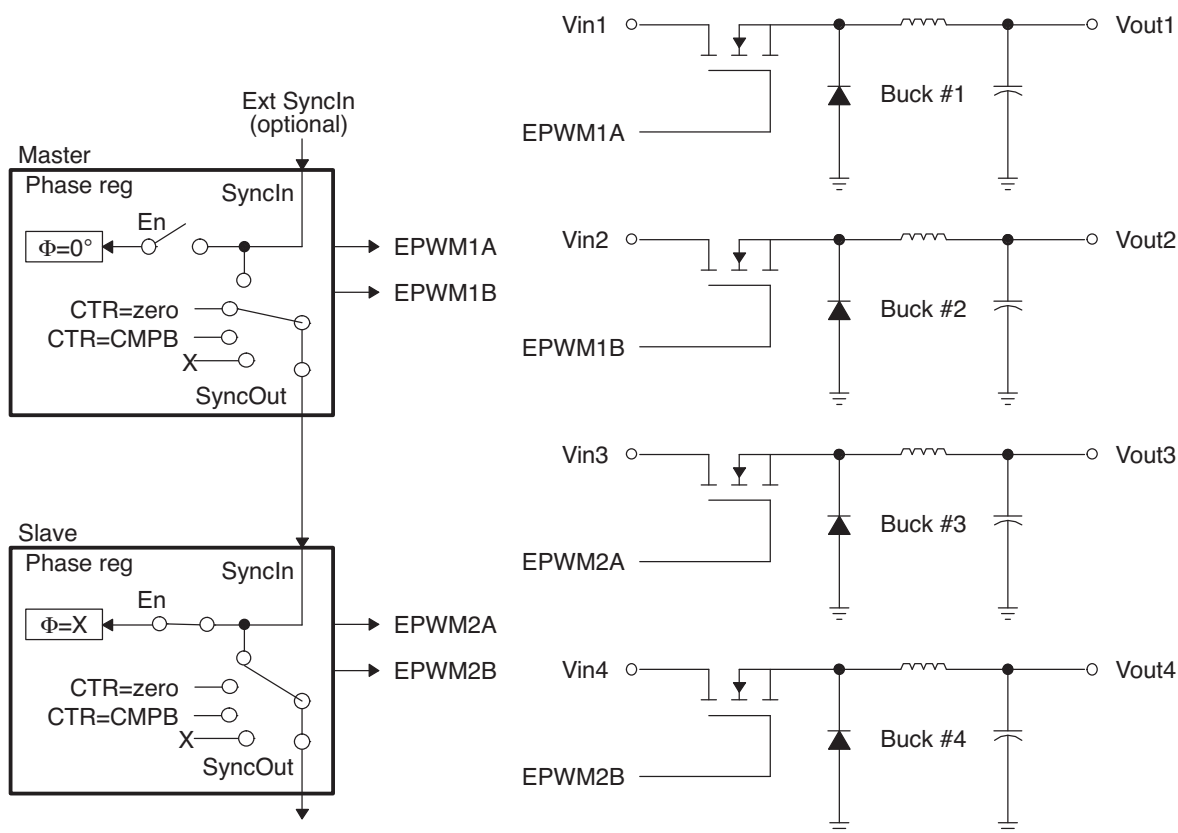
Figure 3-6. Buck Waveforms for Figure 3-5 (Note: $F_{PWM2} = F_{PWM1}$)
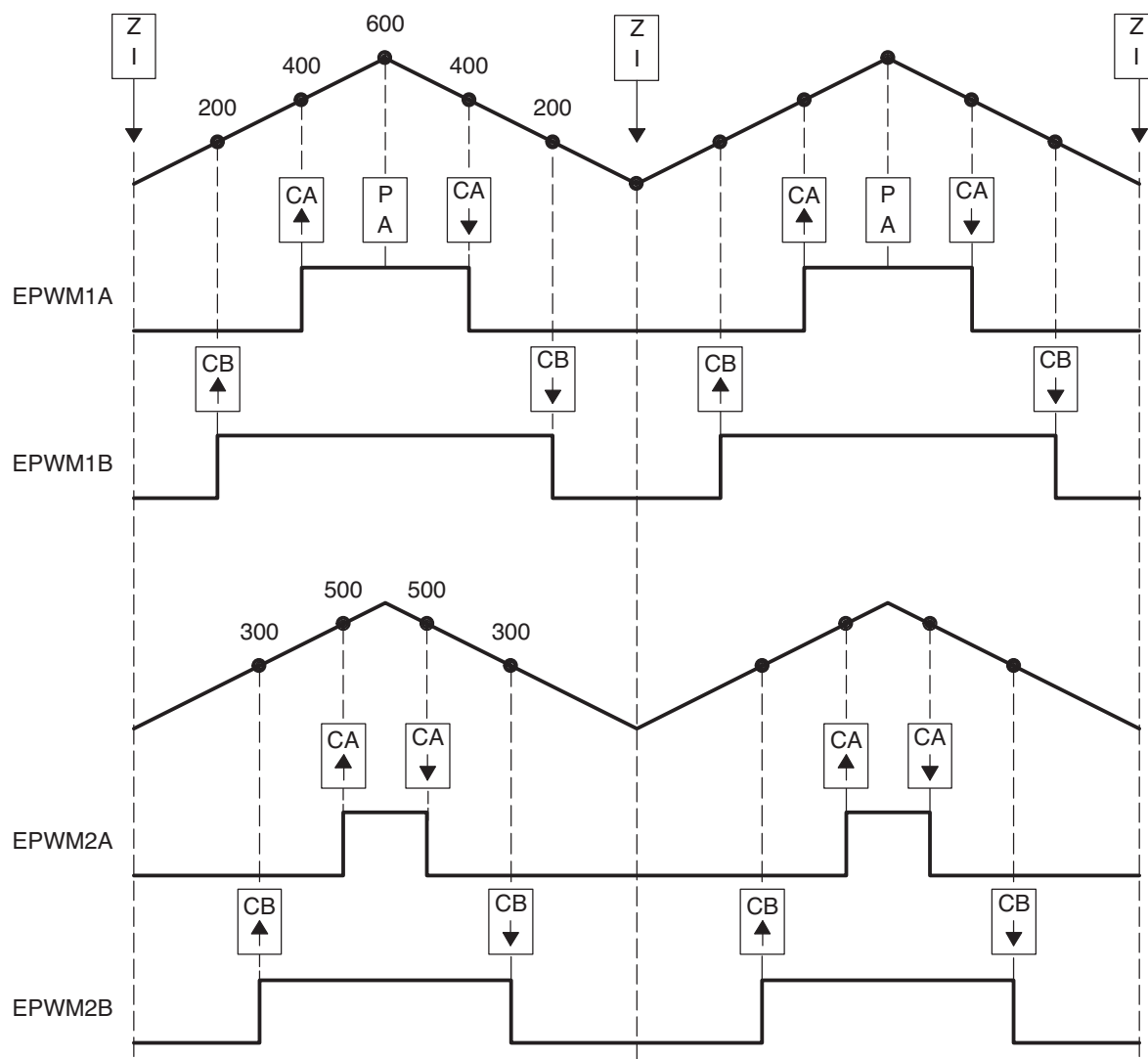
Example 3-2. Code Snippet for Configuration in Figure 3-5

```
//=====================================================================
// Config
//=====================================================================
// Initialization Time
//======================
// EPWM Module 1 config
    EPwm1Regs.TBPRD = 600;                       // Period = 1200 TBCLK counts
    EPwm1Regs.TBPHS = 0;                         // Set Phase register to zero
    EPwm1Regs.TBCTL.bit.CTRMODE = TB_COUNT_UPDOWN; // Symmetrical mode
    EPwm1Regs.TBCTL.bit.PHSEN = TB_DISABLE;      // Master module
    EPwm1Regs.TBCTL.bit.PRLD = TB_SHADOW;
    EPwm1Regs.TBCTL.bit.SYNCOSEL = TB_CTR_ZERO;  // Sync down-stream module
    EPwm1Regs.CMPCTL.bit.SHDWAMODE = CC_SHADOW;
    EPwm1Regs.CMPCTL.bit.SHDWBMODE = CC_SHADOW;
    EPwm1Regs.CMPCTL.bit.LOADAMODE = CC_CTR_ZERO;  // load on CTR=Zero
    EPwm1Regs.CMPCTL.bit.LOADBMODE = CC_CTR_ZERO;  // load on CTR=Zero
    EPwm1Regs.AQCTLA.bit.CAU = AQ_SET;           // set actions for EPWM1A
    EPwm1Regs.AQCTLA.bit.CAD = AQ_CLEAR;
    EPwm1Regs.AQCTLB.bit.CBU = AQ_SET;           // set actions for EPWM1B
    EPwm1Regs.AQCTLB.bit.CBD = AQ_CLEAR;
// EPWM Module 2 config
    EPwm2Regs.TBPRD = 600;                       // Period = 1200 TBCLK counts
    EPwm2Regs.TBPHS = 0;                         // Set Phase register to zero
    EPwm2Regs.TBCTL.bit.CTRMODE = TB_COUNT_UPDOWN; // Symmetrical mode
    EPwm2Regs.TBCTL.bit.PHSEN = TB_ENABLE;       // Slave module
    EPwm2Regs.TBCTL.bit.PRLD = TB_SHADOW;
    EPwm2Regs.TBCTL.bit.SYNCOSEL = TB_SYNC_IN;   // sync flow-through
    EPwm2Regs.CMPCTL.bit.SHDWAMODE = CC_SHADOW;
    EPwm2Regs.CMPCTL.bit.SHDWBMODE = CC_SHADOW;
    EPwm2Regs.CMPCTL.bit.LOADAMODE = CC_CTR_ZERO;  // load on CTR=Zero
    EPwm2Regs.CMPCTL.bit.LOADBMODE = CC_CTR_ZERO;  // load on CTR=Zero
    EPwm2Regs.AQCTLA.bit.CAU = AQ_SET;           // set actions for EPWM2A
    EPwm2Regs.AQCTLA.bit.CAD = AQ_CLEAR;
    EPwm2Regs.AQCTLB.bit.CBU = AQ_SET;           // set actions for EPWM2B
    EPwm2Regs.AQCTLB.bit.CBD = AQ_CLEAR;
//
// Run Time (Note: Example execution of one run-time instance)
//==========================================================
    EPwm1Regs.CMPA.half.CMPA = 400;       // adjust duty for output EPWM1A
    EPwm1Regs.CMPB = 200;                 // adjust duty for output EPWM1B
    EPwm2Regs.CMPA.half.CMPA = 500;       // adjust duty for output EPWM2A
    EPwm2Regs.CMPB = 300;                 // adjust duty for output EPWM2B
```

3.5 Controlling Multiple Half H-Bridge (HHB) Converters

Topologies that require control of multiple switching elements can also be addressed with these same ePWM modules. It is possible to control a Half-H bridge stage with a single ePWM module. This control can be extended to multiple stages. Figure 3-7 shows control of two synchronized Half-H bridge stages where stage 2 can operate at integer multiple (N) frequencies of stage 1. Figure 3-8 shows the waveforms generated by the configuration shown in Figure 3-7.

Module 2 (slave) is configured for Sync flow-through; if required, this configuration allows for a third Half-H bridge to be controlled by PWM module 3 and also, most importantly, to remain in synchronization with master module 1.

Figure 3-7. Control of Two Half-H Bridge Stages ($F_{PWM2} = N \times F_{PWM1}$)
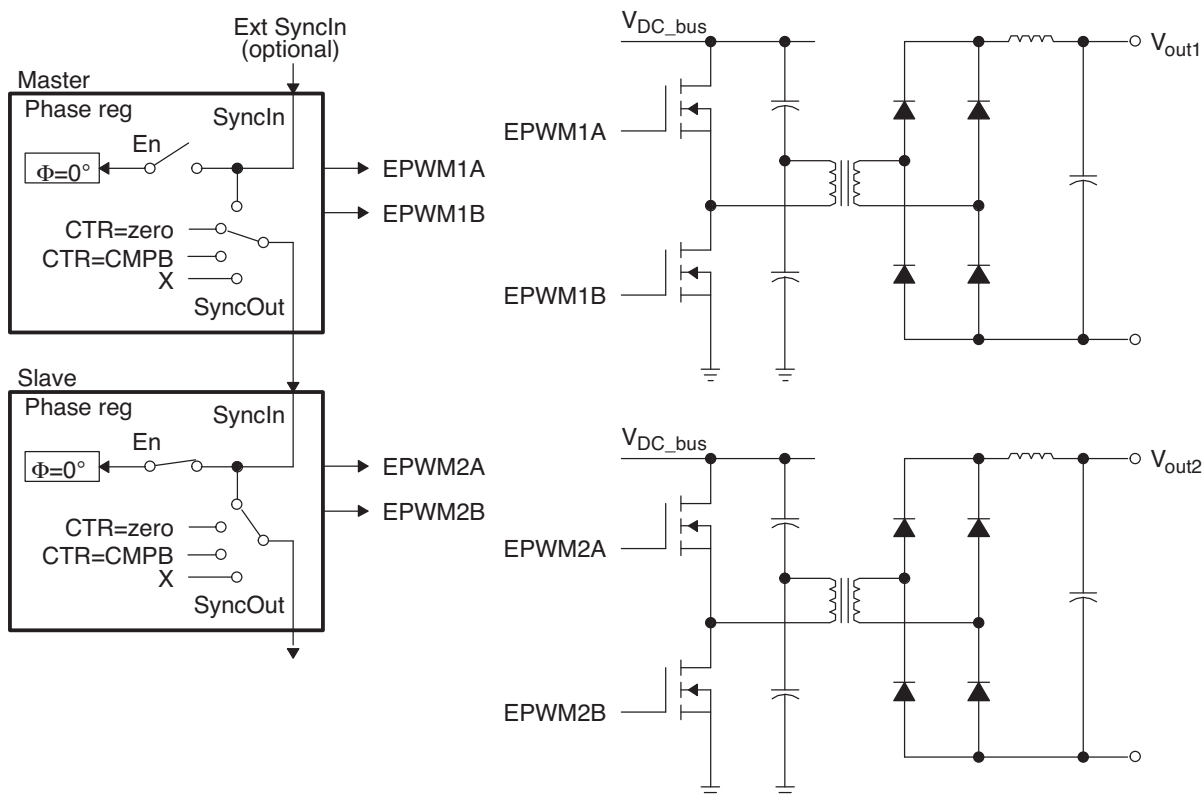
Figure 3-8. Half-H Bridge Waveforms for Figure 3-7 (Note: Here F$_{PWM2}$ = F$_{PWM1}$)
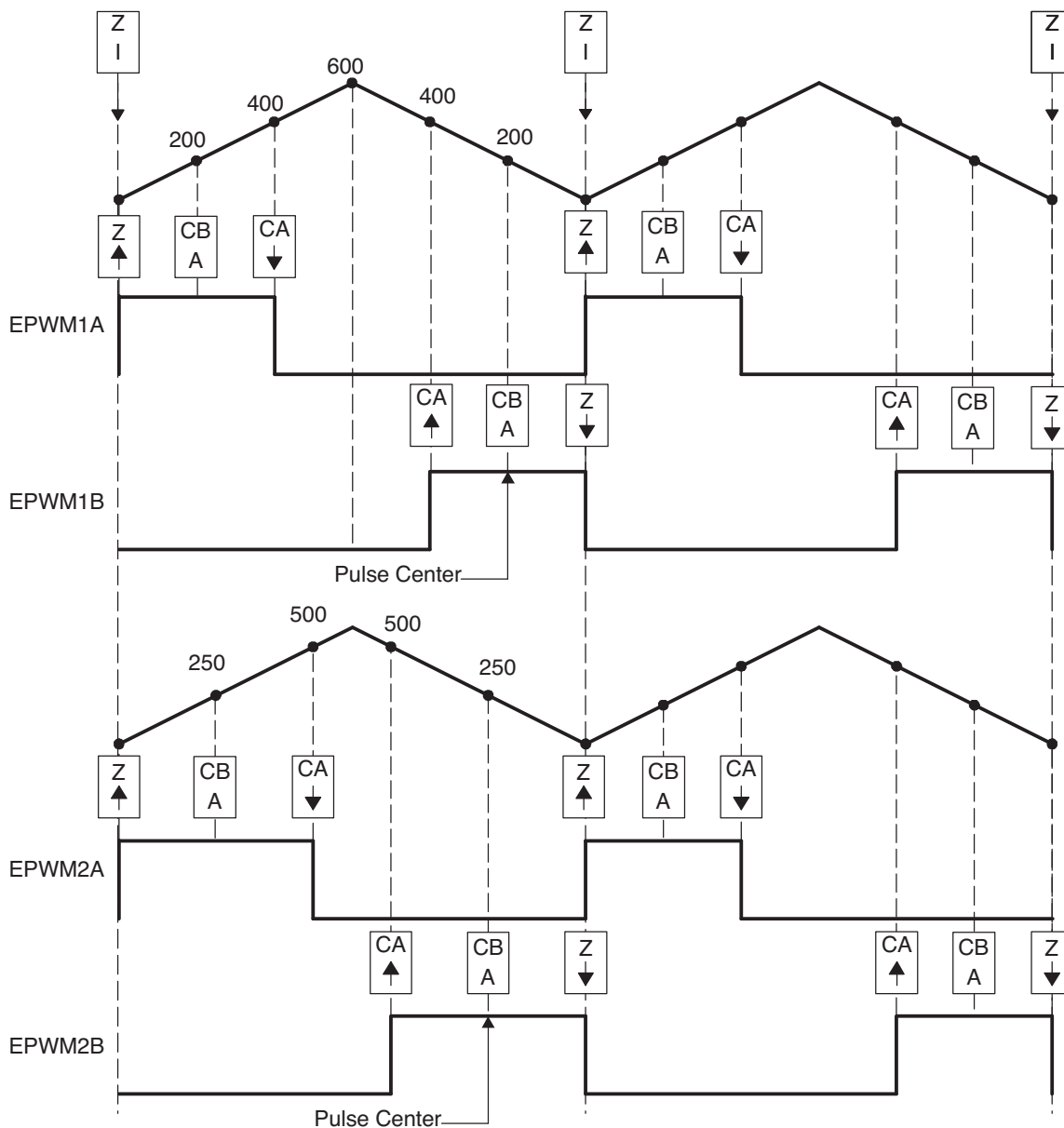
Example 3-3. Code Snippet for Configuration in *Figure 3-7*

```
//=====================================================================
// Config
//=====================================================================
// Initialization Time
//=======================
// EPWM Module 1 config
    EPwm1Regs.TBPRD = 600;                       // Period = 1200 TBCLK counts
    EPwm1Regs.TBPHS = 0;                         // Set Phase register to zero
    EPwm1Regs.TBCTL.bit.CTRMODE = TB_COUNT_UPDOWN; // Symmetrical mode
    EPwm1Regs.TBCTL.bit.PHSEN = TB_DISABLE;      // Master module
    EPwm1Regs.TBCTL.bit.PRLD = TB_SHADOW;
    EPwm1Regs.TBCTL.bit.SYNCOSEL = TB_CTR_ZERO;   // Sync down-stream module
    EPwm1Regs.CMPCTL.bit.SHDWAMODE = CC_SHADOW;
    EPwm1Regs.CMPCTL.bit.SHDWBMODE = CC_SHADOW;
    EPwm1Regs.CMPCTL.bit.LOADAMODE = CC_CTR_ZERO;  // load on CTR=Zero
    EPwm1Regs.CMPCTL.bit.LOADBMODE = CC_CTR_ZERO;  // load on CTR=Zero
    EPwm1Regs.AQCTLA.bit.ZRO = AQ_SET;           // set actions for EPWM1A
    EPwm1Regs.AQCTLA.bit.CAU = AQ_CLEAR;
    EPwm1Regs.AQCTLB.bit.ZRO = AQ_CLEAR;          // set actions for EPWM1B
    EPwm1Regs.AQCTLB.bit.CAD = AQ_SET;
// EPWM Module 2 config
    EPwm2Regs.TBPRD = 600;                       // Period = 1200 TBCLK counts
    EPwm2Regs.TBPHS = 0;                         // Set Phase register to zero
    EPwm2Regs.TBCTL.bit.CTRMODE = TB_COUNT_UPDOWN;// Symmetrical mode
    EPwm2Regs.TBCTL.bit.PHSEN = TB_ENABLE;       // Slave module
    EPwm2Regs.TBCTL.bit.PRLD = TB_SHADOW;
    EPwm2Regs.TBCTL.bit.SYNCOSEL = TB_SYNC_IN;    // sync flow-through
    EPwm2Regs.CMPCTL.bit.SHDWAMODE = CC_SHADOW;
    EPwm2Regs.CMPCTL.bit.SHDWBMODE = CC_SHADOW;
    EPwm2Regs.CMPCTL.bit.LOADAMODE = CC_CTR_ZERO; // load on CTR=Zero
    EPwm2Regs.CMPCTL.bit.LOADBMODE = CC_CTR_ZERO; // load on CTR=Zero
    EPwm2Regs.AQCTLA.bit.ZRO = AQ_SET;           // set actions for EPWM1A
    EPwm2Regs.AQCTLA.bit.CAU = AQ_CLEAR;
    EPwm2Regs.AQCTLB.bit.ZRO = AQ_CLEAR;          // set actions for EPWM1B
    EPwm2Regs.AQCTLB.bit.CAD = AQ_SET;
//========================================================
    EPwm1Regs.CMPA.half.CMPA = 400; // adjust duty for output EPWM1A & EPWM1B
    EPwm1Regs.CMPB = 200;           // adjust point-in-time for ADCSOC trigger
    EPwm2Regs.CMPA.half.CMPA = 500; // adjust duty for output EPWM2A & EPWM2B
    EPwm2Regs.CMPB = 250;           // adjust point-in-time for ADCSOC trigger
```

3.6   Controlling Dual 3-Phase Inverters for Motors (ACI and PMSM)

The idea of multiple modules controlling a single power stage can be extended to the 3-phase Inverter case. In such a case, six switching elements can be controlled using three PWM modules, one for each leg of the inverter. Each leg must switch at the same frequency and all legs must be synchronized. A master + two slaves configuration can easily address this requirement. Figure 3-9 shows how six PWM modules can control two independent 3-phase Inverters; each running a motor.

As in the cases shown in the previous sections, we have a choice of running each inverter at a different frequency (module 1 and module 4 are masters as in Figure 3-9), or both inverters can be synchronized by using one master (module 1) and five slaves. In this case, the frequency of modules 4, 5, and 6 (all equal) can be integer multiples of the frequency for modules 1, 2, 3 (also all equal).

Figure 3-9. Control of Dual 3-Phase Inverter Stages as Is Commonly Used in Motor Control
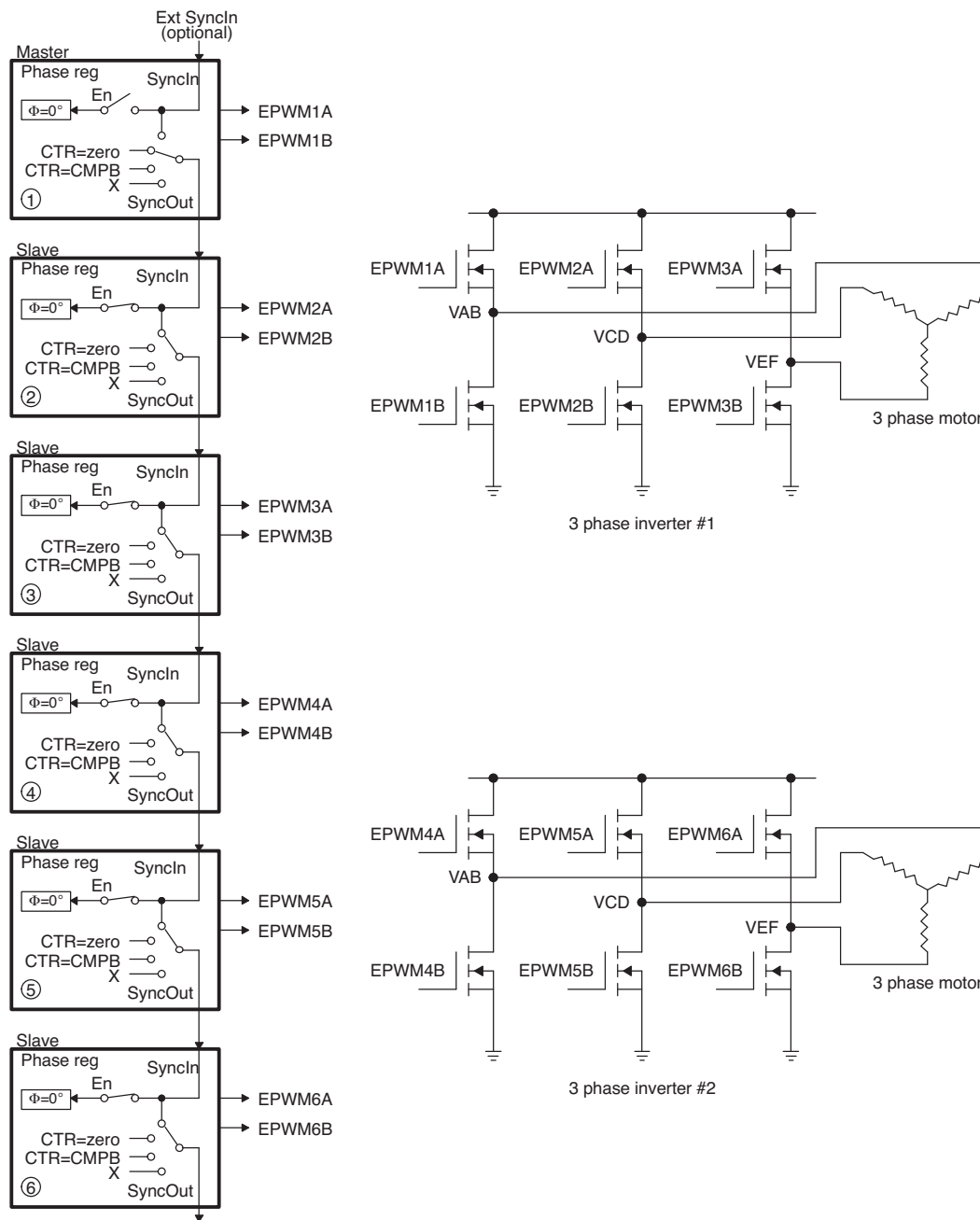
Figure 3-10. 3-Phase Inverter Waveforms for Figure 3-9 (Only One Inverter Shown)
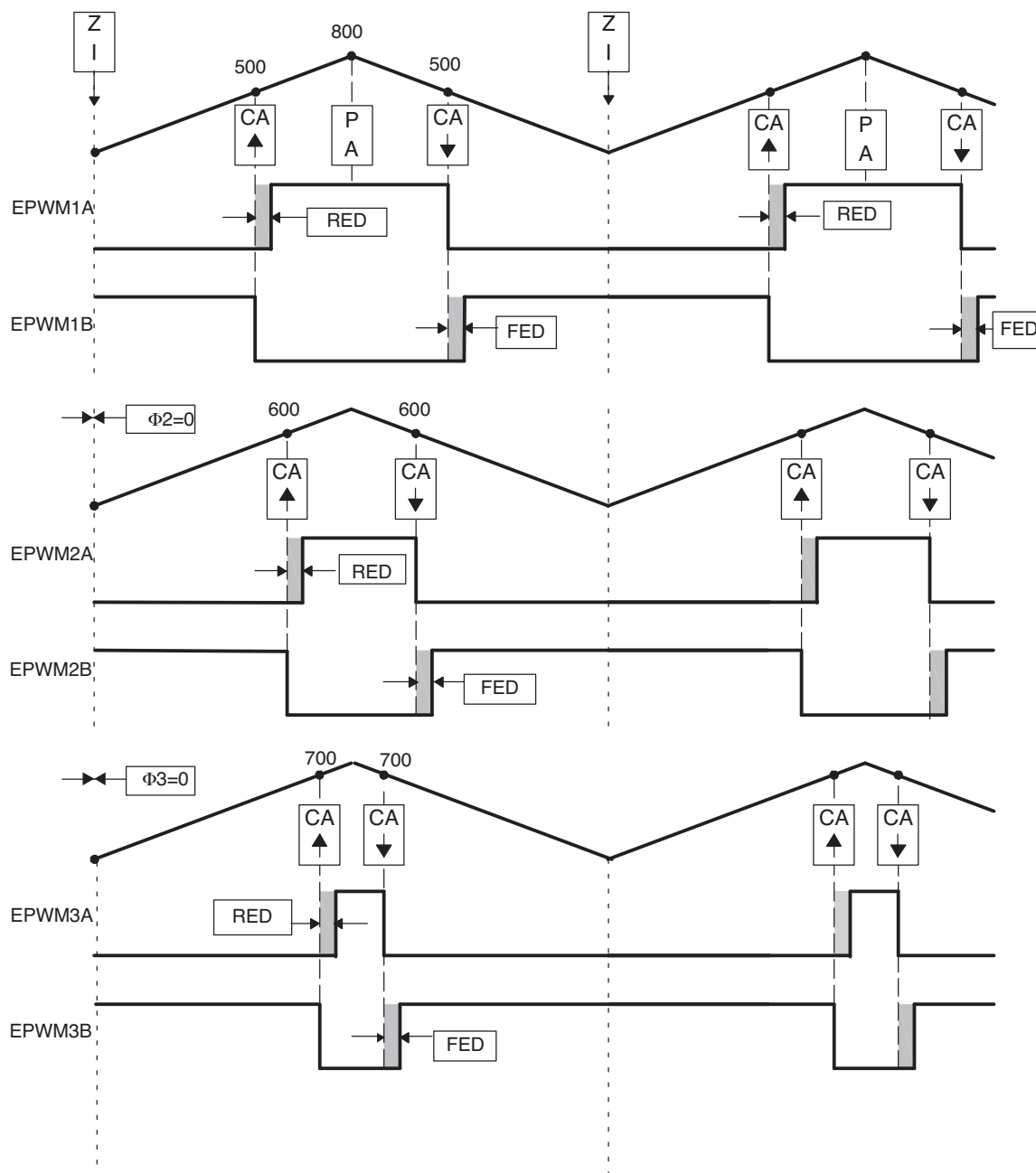
Example 3-4. Code Snippet for Configuration in **Figure 3-9**

```
//======================================================================
// Configuration
//======================================================================
// Initialization Time
//=======================// EPWM Module 1 config
    EPwm1Regs.TBPRD = 800;                       // Period = 1600 TBCLK counts
    EPwm1Regs.TBPHS = 0;                         // Set Phase register to zero
    EPwm1Regs.TBCTL.bit.CTRMODE = TB_COUNT_UPDOWN;// Symmetrical mode
    EPwm1Regs.TBCTL.bit.PHSEN = TB_DISABLE;      // Master module
    EPwm1Regs.TBCTL.bit.PRDLD = TB_SHADOW;
    EPwm1Regs.TBCTL.bit.SYNCOSEL = TB_CTR_ZERO;  // Sync down-stream module
    EPwm1Regs.CMPCTL.bit.SHDWAMODE = CC_SHADOW;
    EPwm1Regs.CMPCTL.bit.SHDWBMODE = CC_SHADOW;
    EPwm1Regs.CMPCTL.bit.LOADAMODE = CC_CTR_ZERO; // load on CTR=Zero
    EPwm1Regs.CMPCTL.bit.LOADBMODE = CC_CTR_ZERO; // load on CTR=Zero
    EPwm1Regs.AQCTLA.bit.CAU = AQ_SET;           // set actions for EPWM1A
    EPwm1Regs.AQCTLA.bit.CAD = AQ_CLEAR;
    EPwm1Regs.DBCTL.bit.MODE = DB_FULL_ENABLE;    // enable Dead-band module
    EPwm1Regs.DBCTL.bit.POLSEL = DB_ACTV_HIC;     // Active Hi complementary
    EPwm1Regs.DBFED = 50;                         // FED = 50 TBCLKs
    EPwm1Regs.DBRED = 50;                         // RED = 50 TBCLKs
// EPWM Module 2 config
    EPwm2Regs.TBPRD = 800;                       // Period = 1600 TBCLK counts
    EPwm2Regs.TBPHS = 0;                         // Set Phase register to zero
    EPwm2Regs.TBCTL.bit.CTRMODE = TB_COUNT_UPDOWN; // Symmetrical mode
    EPwm2Regs.TBCTL.bit.PHSEN = TB_ENABLE;        // Slave module
    EPwm2Regs.TBCTL.bit.PRDLD = TB_SHADOW;
    EPwm2Regs.TBCTL.bit.SYNCOSEL = TB_SYNC_IN;    // sync flow-through
    EPwm2Regs.CMPCTL.bit.SHDWAMODE = CC_SHADOW;
    EPwm2Regs.CMPCTL.bit.SHDWBMODE = CC_SHADOW;
    EPwm2Regs.CMPCTL.bit.LOADAMODE = CC_CTR_ZERO; // load on CTR=Zero
    EPwm2Regs.CMPCTL.bit.LOADBMODE = CC_CTR_ZERO; // load on CTR=Zero
    EPwm2Regs.AQCTLA.bit.CAU = AQ_SET;           // set actions for EPWM2A
    EPwm2Regs.AQCTLA.bit.CAD = AQ_CLEAR;
    EPwm2Regs.DBCTL.bit.MODE = DB_FULL_ENABLE;    // enable Dead-band module
    EPwm2Regs.DBCTL.bit.POLSEL = DB_ACTV_HIC;     // Active Hi complementary
    EPwm2Regs.DBFED = 50; // FED = 50 TBCLKs
    EPwm2Regs.DBRED = 50; // RED = 50 TBCLKs
// EPWM Module 3 config
    EPwm3Regs.TBPRD = 800;                       // Period = 1600 TBCLK counts
    EPwm3Regs.TBPHS = 0;                         // Set Phase register to zero
    EPwm3Regs.TBCTL.bit.CTRMODE = TB_COUNT_UPDOWN;// Symmetrical mode
    EPwm3Regs.TBCTL.bit.PHSEN = TB_ENABLE;        // Slave module
    EPwm3Regs.TBCTL.bit.PRDLD = TB_SHADOW;
    EPwm3Regs.TBCTL.bit.SYNCOSEL = TB_SYNC_IN;    // sync flow-through
    EPwm3Regs.CMPCTL.bit.SHDWAMODE = CC_SHADOW;
    EPwm3Regs.CMPCTL.bit.SHDWBMODE = CC_SHADOW;
    EPwm3Regs.CMPCTL.bit.LOADAMODE = CC_CTR_ZERO; // load on CTR=Zero
    EPwm3Regs.CMPCTL.bit.LOADBMODE = CC_CTR_ZERO; // load on CTR=Zero
    EPwm3Regs.AQCTLA.bit.CAU = AQ_SET;           // set actions for EPWM3A
    EPwm3Regs.AQCTLA.bit.CAD = AQ_CLEAR;
    EPwm3Regs.DBCTL.bit.MODE = DB_FULL_ENABLE;    // enable Dead-band module
    EPwm3Regs.DBCTL.bit.POLSEL = DB_ACTV_HIC;     // Active Hi complementary
    EPwm3Regs.DBFED = 50; // FED = 50 TBCLKs
    EPwm3Regs.DBRED = 50; // RED = 50 TBCLKs
// Run Time (Note: Example execution of one run-time instant)
//========================================================
EPwm1Regs.CMPA.half.CMPA = 500; // adjust duty for output EPWM1A
EPwm2Regs.CMPA.half.CMPA = 600; // adjust duty for output EPWM2A
EPwm3Regs.CMPA.half.CMPA = 700; // adjust duty for output EPWM3A
```

3.7 Practical Applications Using Phase Control Between PWM Modules

So far, none of the examples have made use of the phase register (TBPHS). It has either been set to zero or its value has been a don't care. However, by programming appropriate values into TBPHS, multiple PWM modules can address another class of power topologies that rely on phase relationship between legs (or stages) for correct operation. As described in the TB module section, a PWM module can be configured to allow a SyncIn pulse to cause the TBPHS register to be loaded into the TBCTR register. To illustrate this concept, Figure 3-11 shows a master and slave module with a phase relationship of 120°, i.e., the slave leads the master.

Figure 3-11. Configuring Two PWM Modules for Phase Control
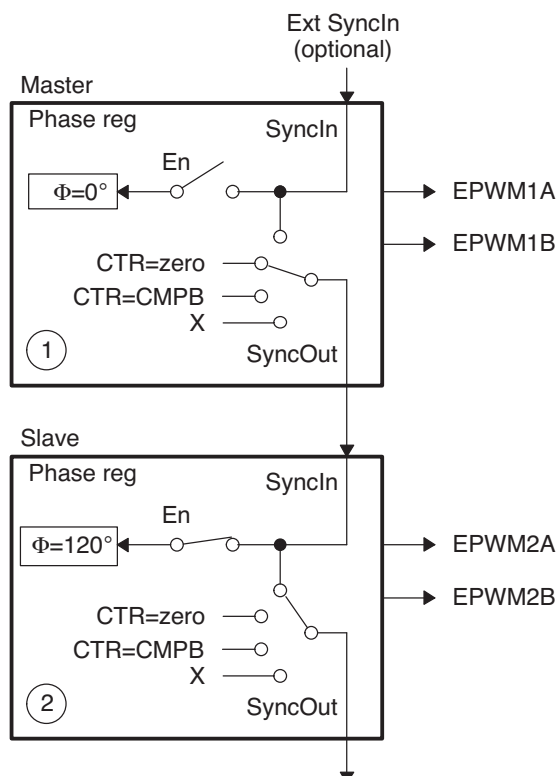
Figure 3-12 shows the associated timing waveforms for this configuration. Here, TBPRD = 600 for both master and slave. For the slave, TBPHS = 200 (i.e., 200/600 X 360° = 120°). Whenever the master generates a SyncIn pulse (CTR = PRD), the value of TBPHS = 200 is loaded into the slave TBCTR register so the slave time-base is always leading the master's time-base by 120°.

Figure 3-12. Timing Waveforms Associated With Phase Control Between 2 Modules
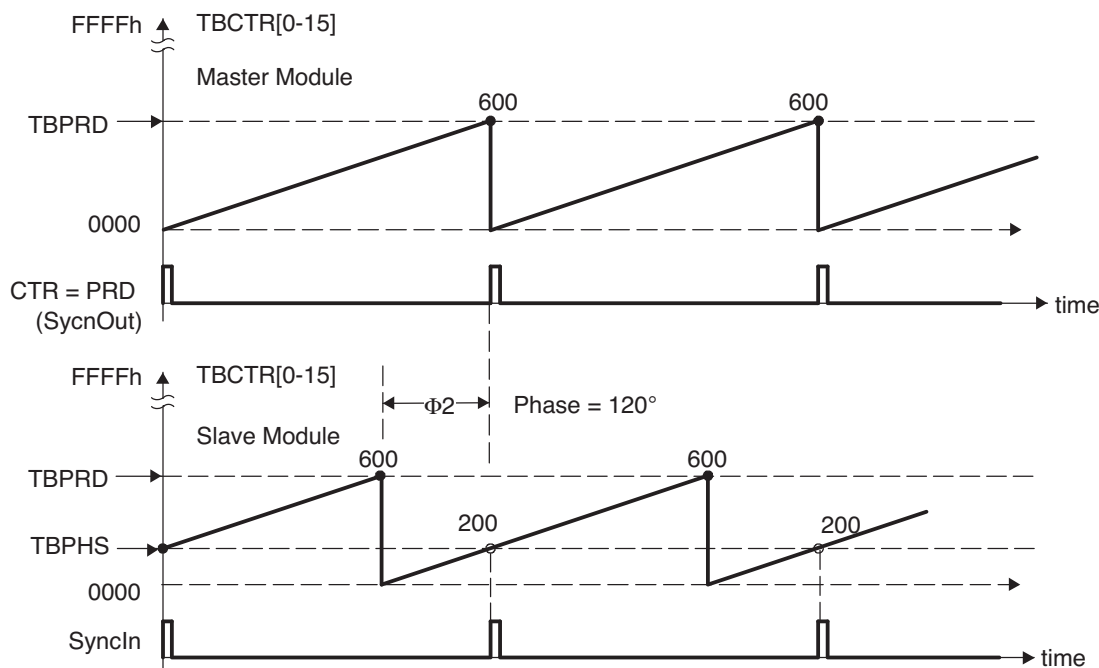
3.8 Controlling a 3-Phase Interleaved DC/DC Converter

A popular power topology that makes use of phase-offset between modules is shown in Figure 3-13. This system uses three PWM modules, with module 1 configured as the master. To work, the phase relationship between adjacent modules must be F = 120°. This is achieved by setting the slave TBPHS registers 2 and 3 with values of 1/3 and 2/3 of the period value, respectively. For example, if the period register is loaded with a value of 600 counts, then TBPHS (slave 2) = 200 and TBPHS (slave 3) = 400. Both slave modules are synchronized to the master 1 module.

This concept can be extended to four or more phases, by setting the TBPHS values appropriately. The following formula gives the TBPHS values for N phases:

TBPHS(N,M) = (TBPRD/N) x (—1)

Where:

N = number of phases

M = PWM module number

For example, for the 3-phase case (N=3), TBPRD = 600,

TBPHS(3,2) = (600/3) x (2-1) = 200 (i.e., Phase value for Slave module 2)

TBPHS(3,3) = 400 (i.e., Phase value for Slave module 3)

Figure 3-14 shows the waveforms for the configuration in Figure 3-13.

Figure 3-13. Control of a 3-Phase Interleaved DC/DC Converter
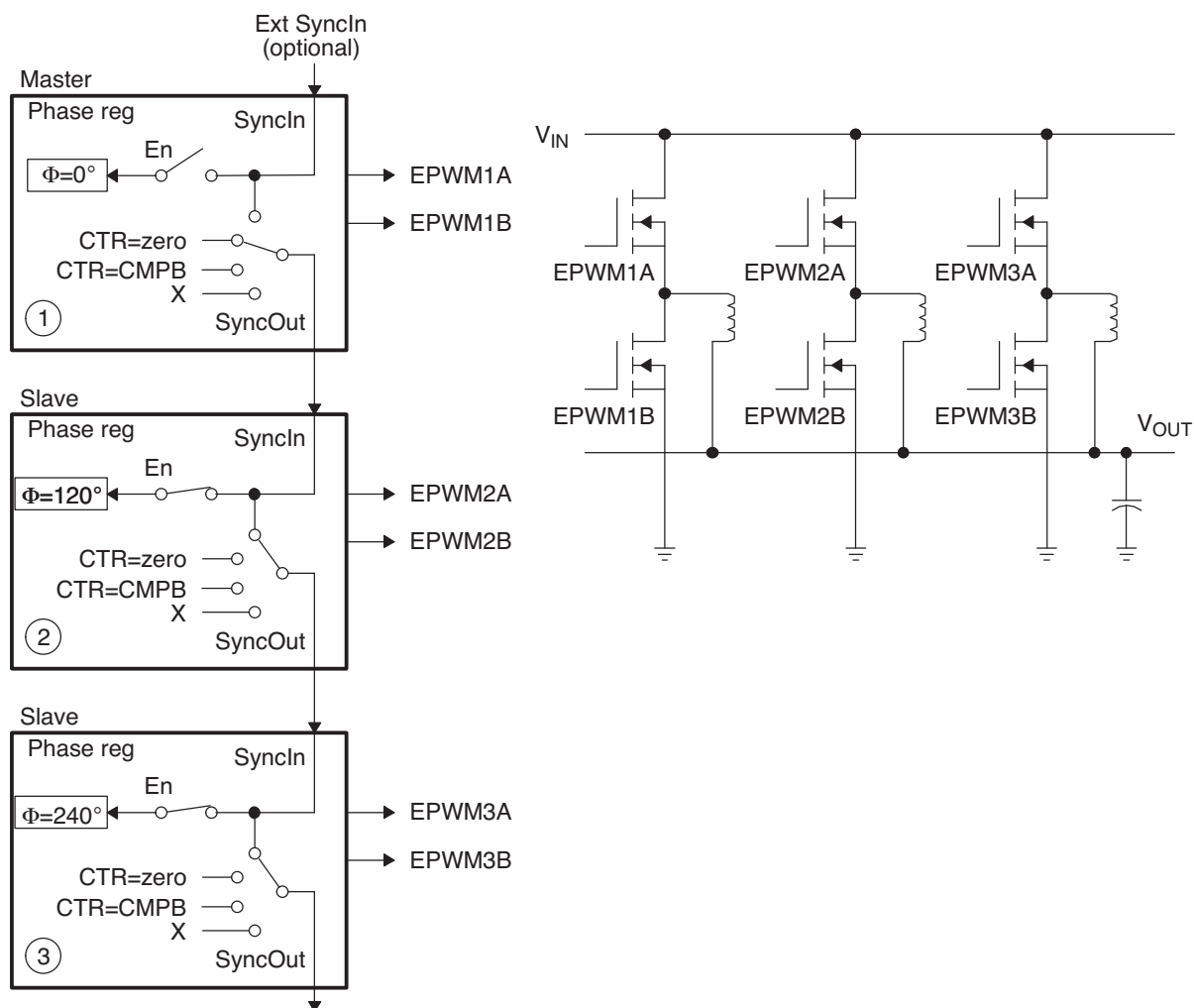
Figure 3-14. 3-Phase Interleaved DC/DC Converter Waveforms for Figure 3-13
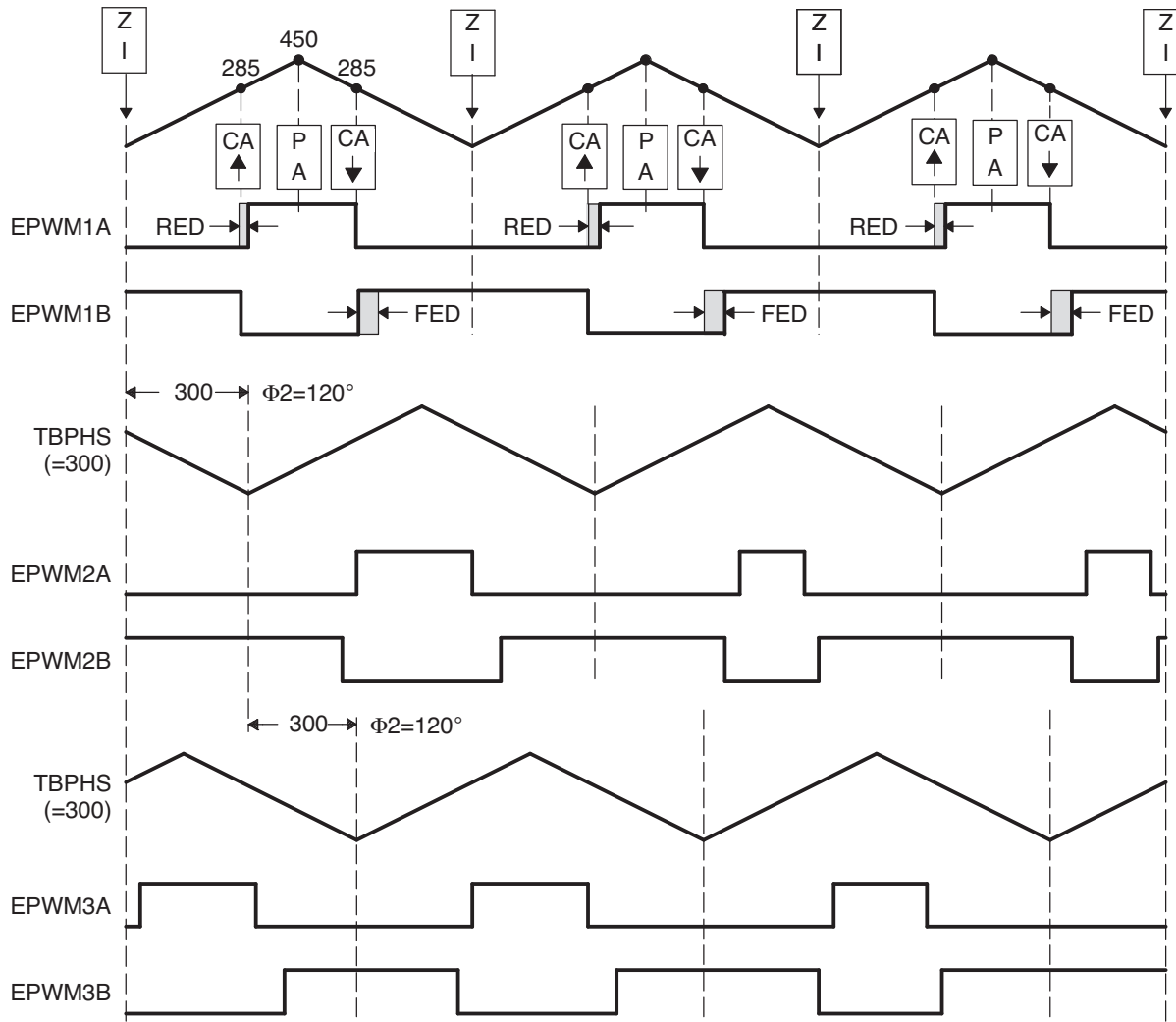
Example 3-5. Code Snippet for Configuration in Figure 3-13

```
//=======================================================================
// Config
// Initialization Time
//=======================
// EPWM Module 1 config
EPwm1Regs.TBPRD = 450;                          // Period = 900 TBCLK counts
EPwm1Regs.TBPHS = 0;                            // Set Phase register to zero
EPwm1Regs.TBCTL.bit.CTRMODE = TB_COUNT_UPDOWN;  // Symmetrical mode
EPwm1Regs.TBCTL.bit.PHSEN = TB_DISABLE;         // Master module
EPwm1Regs.TBCTL.bit.PRDLD = TB_SHADOW;
EPwm1Regs.TBCTL.bit.SYNCOSEL = TB_CTR_ZERO;     // Sync down-stream module
EPwm1Regs.CMPCTL.bit.SHDWAMODE = CC_SHADOW;
EPwm1Regs.CMPCTL.bit.SHDWBMODE = CC_SHADOW;
EPwm1Regs.CMPCTL.bit.LOADAMODE = CC_CTR_ZERO;   // load on CTR=Zero
EPwm1Regs.CMPCTL.bit.LOADBMODE = CC_CTR_ZERO;   // load on CTR=Zero
EPwm1Regs.AQCTLA.bit.CAU = AQ_SET;              // set actions for EPWM1A
EPwm1Regs.AQCTLA.bit.CAD = AQ_CLEAR;
EPwm1Regs.DBCTL.bit.MODE = DB_FULL_ENABLE;      // enable Dead-band module
EPwm1Regs.DBCTL.bit.POLSEL = DB_ACTV_HIC;       // Active Hi complementary
EPwm1Regs.DBFED = 20;                           // FED = 20 TBCLKs
EPwm1Regs.DBRED = 20;                           // RED = 20 TBCLKs
// EPWM Module 2 config
EPwm2Regs.TBPRD = 450;                          // Period = 900 TBCLK counts
EPwm2Regs.TBPHS = 300;                          // Phase = 300/900 * 360 = 120 deg
EPwm2Regs.TBCTL.bit.CTRMODE =    TB_COUNT_UPDOWN;  // Symmetrical mode
EPwm2Regs.TBCTL.bit.PHSEN =    TB_ENABLE;        // Slave module
EPwm2Regs.TBCTL.bit.PHSDIR =    TB_DOWN;          // Count DOWN on sync (=120 deg)
EPwm2Regs.TBCTL.bit.PRDLD =    TB_SHADOW;
EPwm2Regs.TBCTL.bit.SYNCOSEL =    TB_SYNC_IN;     // sync flow-through
EPwm2Regs.CMPCTL.bit.SHDWAMODE = CC_SHADOW;
EPwm2Regs.CMPCTL.bit.SHDWBMODE = CC_SHADOW;
EPwm2Regs.CMPCTL.bit.LOADAMODE = CC_CTR_ZERO;   // load on CTR=Zero
EPwm2Regs.CMPCTL.bit.LOADBMODE = CC_CTR_ZERO;   // load on CTR=Zero
EPwm2Regs.AQCTLA.bit.CAU = AQ_SET;              // set actions for EPWM2A
EPwm2Regs.AQCTLA.bit.CAD = AQ_CLEAR;
EPwm2Regs.DBCTL.bit.MODE = DB_FULL_ENABLE;      // enable Dead-band module
EPwm2Regs.DBCTL.bit.POLSEL = DB_ACTV_HIC;       // Active Hi Complementary
EPwm2Regs.DBFED = 20;                           // FED = 20 TBCLKs
EPwm2Regs.DBRED = 20;                           // RED = 20 TBCLKs
// EPWM Module 3 config
EPwm3Regs.TBPRD = 450;                          // Period = 900 TBCLK counts
EPwm3Regs.TBPHS = 300;                          // Phase = 300/900 * 360 = 120 deg
EPwm3Regs.TBCTL.bit.CTRMODE = TB_COUNT_UPDOWN;   // Symmetrical mode
EPwm3Regs.TBCTL.bit.PHSEN = TB_ENABLE;          // Slave module
EPwm2Regs.TBCTL.bit.PHSDIR = TB_UP;             // Count UP on sync (=240 deg)
EPwm3Regs.TBCTL.bit.PRDLD = TB_SHADOW;
EPwm3Regs.TBCTL.bit.SYNCOSEL = TB_SYNC_IN;      // sync flow-through
EPwm3Regs.CMPCTL.bit.SHDWAMODE = CC_SHADOW;
EPwm3Regs.CMPCTL.bit.SHDWBMODE = CC_SHADOW;
EPwm3Regs.CMPCTL.bit.LOADAMODE = CC_CTR_ZERO;   // load on CTR=Zero
EPwm3Regs.CMPCTL.bit.LOADBMODE = CC_CTR_ZERO;   // load on CTR=Zero
EPwm3Regs.AQCTLA.bit.CAU = AQ_SET;              // set actions for EPWM3Ai
EPwm3Regs.AQCTLA.bit.CAD = AQ_CLEAR;
EPwm3Regs.DBCTL.bit.MODE = DB_FULL_ENABLE;      // enable Dead-band module
EPwm3Regs.DBCTL.bit.POLSEL = DB_ACTV_HIC;       // Active Hi complementary
EPwm3Regs.DBFED = 20;                           // FED = 20 TBCLKs
EPwm3Regs.DBRED = 20;                           // RED = 20 TBCLKs
// Run Time (Note: Example execution of one run-time instant)
//============================================================
EPwm1Regs.CMPA.half.CMPA = 285;       // adjust duty for output EPWM1A
EPwm2Regs.CMPA.half.CMPA = 285;       // adjust duty for output EPWM2A
EPwm3Regs.CMPA.half.CMPA = 285;       // adjust duty for output EPWM3A
```

3.9 Controlling Zero Voltage Switched Full Bridge (ZVSFB) Converter

The example given in Figure 3-15 assumes a static or constant phase relationship between legs (modules). In such a case, control is achieved by modulating the duty cycle. It is also possible to dynamically change the phase value on a cycle-by-cycle basis. This feature lends itself to controlling a class of power topologies known as *phase-shifted full bridge*, or *zero voltage switched full bridge.* Here the controlled parameter is not duty cycle (this is kept constant at approximately 50 percent); instead it is the phase relationship between legs. Such a system can be implemented by allocating the resources of two PWM modules to control a single power stage, which in turn requires control of four switching elements. Figure 3-16 shows a master/slave module combination synchronized together to control a full H-bridge. In this case, both master and slave modules are required to switch at the same PWM frequency. The phase is controlled by using the slave's phase register (TBPHS). The master's phase register is not used and therefore can be initialized to zero.

Figure 3-15. Controlling a Full-H Bridge Stage ($F_{PWM2} = F_{PWM1}$)
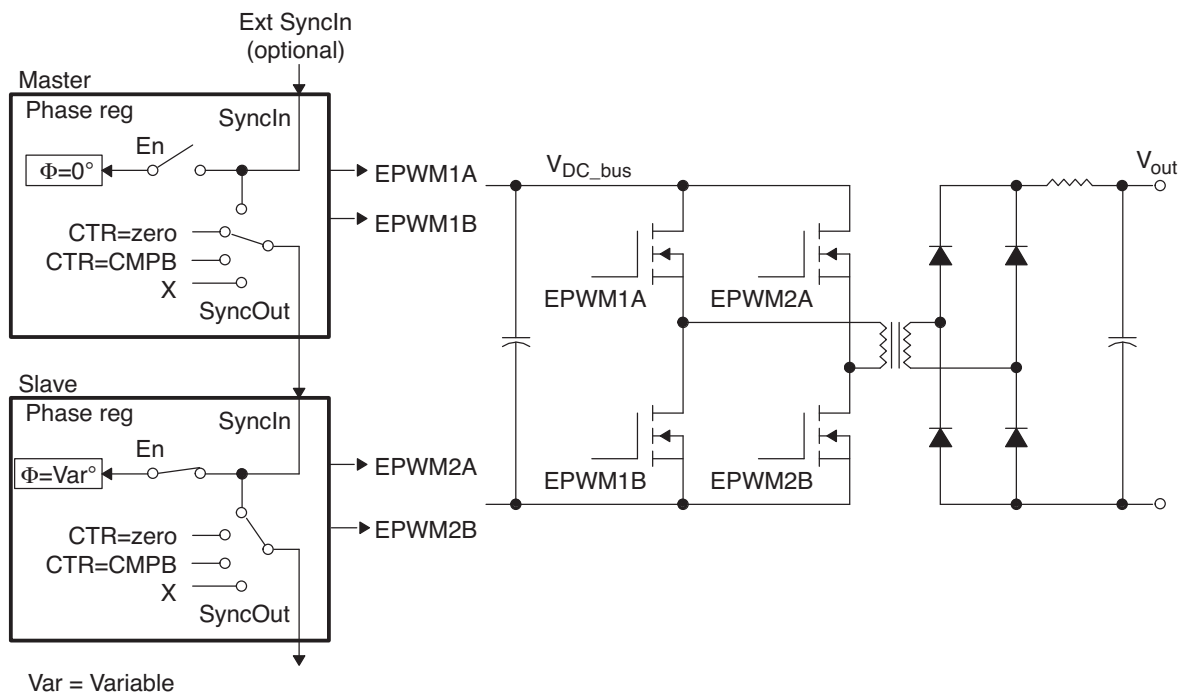
Figure 3-16. ZVS Full-H Bridge Waveforms
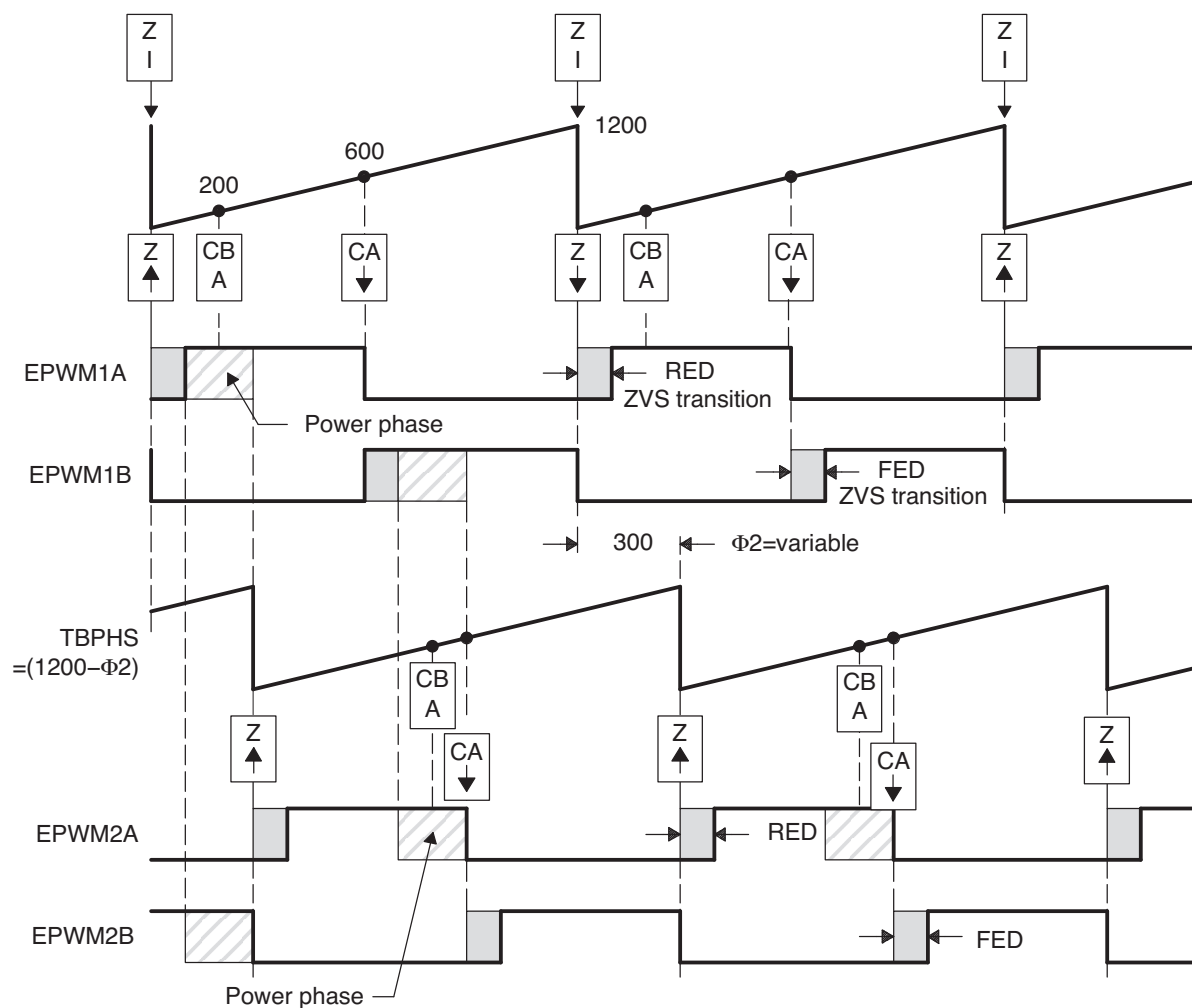
Example 3-6. Code Snippet for Configuration in Figure 3-15

```
//====================================================================
// Config
//====================================================================
// Initialization Time
//=======================
// EPWM Module 1 config
EPwm1Regs.TBPRD = 1200;                          // Period = 1201 TBCLK counts
EPwm1Regs.CMPA = 600;                            // Set 50% fixed duty for EPWM1A
EPwm1Regs.TBPHS = 0;                             // Set Phase register to zero
EPwm1Regs.TBCTL.bit.CTRMODE = TB_COUNT_UP;       // Asymmetrical mode
EPwm1Regs.TBCTL.bit.PHSEN = TB_DISABLE;          // Master module
EPwm1Regs.TBCTL.bit.PRDLD = TB_SHADOW;
EPwm1Regs.TBCTL.bit.SYNCOSEL = TB_CTR_ZERO;      // Sync down-stream module
EPwm1Regs.CMPCTL.bit.SHDWAMODE = CC_SHADOW;
EPwm1Regs.CMPCTL.bit.SHDWBMODE = CC_SHADOW;
EPwm1Regs.CMPCTL.bit.LOADAMODE = CC_CTR_ZERO;    // load on CTR=Zero
EPwm1Regs.CMPCTL.bit.LOADBMODE = CC_CTR_ZERO;    // load on CTR=Zero
EPwm1Regs.AQCTLA.bit.ZRO = AQ_SET;               // set actions for EPWM1A
EPwm1Regs.AQCTLA.bit.CAU = AQ_CLEAR;
EPwm1Regs.DBCTL.bit.MODE = DB_FULL_ENABLE;       // enable Dead-band module
EPwm1Regs.DBCTL.bit.POLSEL = DB_ACTV_HIC;        // Active Hi complementary
EPwm1Regs.DBFED = 50;                            // FED = 50 TBCLKs initially
EPwm1Regs.DBRED = 70;                            // RED = 70 TBCLKs initially
// EPWM Module 2 config
EPwm2Regs.TBPRD = 1200;                          // Period = 1201 TBCLK counts
EPwm2Regs.CMPA.half.CMPA = 600;                  // Set 50% fixed duty EPWM2A
EPwm2Regs.TBPHS = 0;                             // Set Phase register to zero initially
EPwm2Regs.TBCTL.bit.CTRMODE = TB_COUNT_UP;       // Asymmetrical mode
EPwm2Regs.TBCTL.bit.PHSEN = TB_ENABLE;           // Slave module
EPwm2Regs.TBCTL.bit.PRDLD = TB_SHADOW;
EPwm2Regs.TBCTL.bit.SYNCOSEL = TB_SYNC_IN;       // sync flow-through
EPwm2Regs.CMPCTL.bit.SHDWAMODE = CC_SHADOW;
EPwm2Regs.CMPCTL.bit.SHDWBMODE = CC_SHADOW;
EPwm2Regs.CMPCTL.bit.LOADAMODE = CC_CTR_ZERO;    // load on CTR=Zero
EPwm2Regs.CMPCTL.bit.LOADBMODE = CC_CTR_ZERO;    // load on CTR=Zero
EPwm2Regs.AQCTLA.bit.ZRO = AQ_SET;               // set actions for EPWM2A
EPwm2Regs.AQCTLA.bit.CAU = AQ_CLEAR;
EPwm2Regs.DBCTL.bit.MODE = DB_FULL_ENABLE;       // enable Dead-band module
EPwm2Regs.DBCTL.bit.POLSEL = DB_ACTV_HIC;        // Active Hi complementary
EPwm2Regs.DBFED = 30;                            // FED = 30 TBCLKs initially
EPwm2Regs.DBRED = 40;                            // RED = 40 TBCLKs initially


// Run Time (Note: Example execution of one run-time instant)
//============================================================
EPwm2Regs.TBPHS = 1200-300;                      // Set Phase reg to 300/1200 * 360 = 90 deg
EPwm1Regs.DBFED = FED1_NewValue;                 // Update ZVS transition interval
EPwm1Regs.DBRED = RED1_NewValue;                 // Update ZVS transition interval
EPwm2Regs.DBFED = FED2_NewValue;                 // Update ZVS transition interval
EPwm2Regs.DBRED = RED2_NewValue;                 // Update ZVS transition interval
EPwm1Regs.CMPB = 200;                            // adjust point-in-time for ADCSOC trigger
```

Registers

This chapter includes the register layouts and bit description for the submodules.

4.1 Time-Base Submodule Registers

Figure 4-1 through Figure 4-5 and Table 4-1 through Table 4-5 provide the time-base register definitions.

Figure 4-1. Time-Base Period Register (TBPRD)

15	0
TBPRD	
R/W-0	
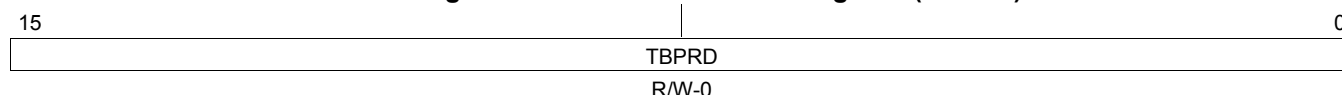
LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 4-1. Time-Base Period Register (TBPRD) Field Descriptions

Bits	Name	Value	Description
15-0	TBPRD	0000-FFFF	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 zero. • 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.

Figure 4-2. Time-Base Phase Register (TBPHS)

15	0
TBPHS	
R/W-0	
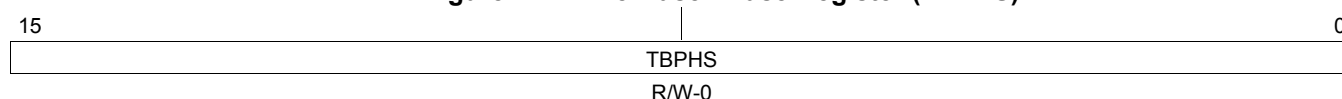
LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 4-2. Time-Base Phase Register (TBPHS) Field Descriptions

Bits	Name	Value	Description
15-0	TBPHS	0000-FFFF	These bits set time-base counter phase of the selected ePWM relative to the time-base that is supplying the synchronization input signal. • If TBCTL[PHSEN] = 0, then the synchronization event is ignored and the time-base counter is not loaded with the phase. • If TBCTL[PHSEN] = 1, then the time-base counter (TBCTR) will be loaded with the phase (TBPHS) when a synchronization event occurs. The synchronization event can be initiated by the input synchronization signal (EPWMxSYNCI) or by a software forced synchronization.

Figure 4-3. Time-Base Counter Register (TBCTR)

15	0
TBCTR	
R/W-0	
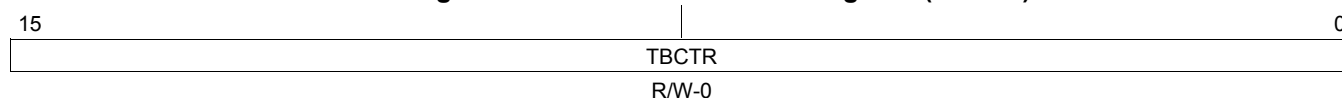
LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 4-3. Time-Base Counter Register (TBCTR) Field Descriptions

Bits	Name	Value	Description
15-0	TBCTR	0000-FFFF	Reading these bits gives the current time-base counter value. Writing to these bits sets the current time-base counter value. The update happens as soon as the write occurs; the write is NOT synchronized to the time-base clock (TBCLK) and the register is not shadowed.

Figure 4-4. Time-Base Control Register (TBCTL)

15	14	13	12		10	9	8
FREE, SOFT		PHSDIR	CLKDIV			HSPCLKDIV	
R/W-0		R/W-0	R/W-0			R/W-0,0,1	

7	6	5	4	3	2	1	0
HSPCLKDIV	SWFSYNC	SYNCOSEL		PRDLD	PHSEN	CTRMODE	
R/W-0,0,1	R/W-0	R/W-0		R/W-0	R/W-0	R/W-11	

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

Table 4-4. Time-Base Control Register (TBCTL) Field Descriptions

Bit	Field	Value	Description
15:14	FREE, SOFT		Emulation Mode Bits. These bits select the behavior of the ePWM time-base counter during emulation events:
		00	Stop after the next time-base counter increment or decrement
		01	Stop when counter completes a whole cycle:
			• Up-count mode: stop when the time-base counter = period (TBCTR = TBPRD)
			• Down-count mode: stop when the time-base counter = 0x0000 (TBCTR = 0x0000)
			• Up-down-count mode: stop when the time-base counter = 0x0000 (TBCTR = 0x0000)
		1X	Free run
13	PHSDIR		Phase Direction Bit.
			This bit is only used when the time-base counter is configured in the up-down-count mode. The PHSDIR bit indicates the direction the time-base counter (TBCTR) will count after a synchronization event occurs and a new phase value is loaded from the phase (TBPHS) register. This is irrespective of the direction of the counter before the synchronization event..
			In the up-count and down-count modes this bit is ignored.
		0	Count down after the synchronization event.
		1	Count up after the synchronization event.
12:10	CLKDIV		Time-base Clock Prescale Bits
			These bits determine part of the time-base clock prescale value. TBCLK = SYSCLKOUT / (HSPCLKDIV × CLKDIV)
		000	/1 (default on reset)
		001	/2
		010	/4
		011	/8
		100	/16
		101	/32
		110	/64
		111	/128
9:7	HSPCLKDIV		High Speed Time-base Clock Prescale Bits
			These bits determine part of the time-base clock prescale value. TBCLK = SYSCLKOUT / (HSPCLKDIV × CLKDIV)
			This divisor emulates the HSPCLK in the TMS320x281x system as used on the Event Manager (EV) peripheral.
		000	/1
		001	/2 (default on reset)
		010	/4
		011	/6
		100	/8
		101	/10
		110	/12
		111	/14

Table 4-4. Time-Base Control Register (TBCTL) Field Descriptions (continued)

Bit	Field	Value	Description
6	SWFSYNC		Software Forced Synchronization Pulse
		0	Writing a 0 has no effect and reads always return a 0.
		1	Writing a 1 forces a one-time synchronization pulse to be generated.
			This event is ORed with the EPWMxSYNCI input of the ePWM module.
			SWFSYNC is valid (operates) only when EPWMxSYNCI is selected by SYNCOSEL = 00.
5:4	SYNCOSEL		Synchronization Output Select. These bits select the source of the EPWMxSYNCO signal.
		00	EPWMxSYNC:
		01	CTR = zero: Time-base counter equal to zero (TBCTR = 0x0000)
		10	CTR = CMPB : Time-base counter equal to counter-compare B (TBCTR = CMPB)
		11	Disable EPWMxSYNCO signal
3	PRDLD		Active Period Register Load From Shadow Register Select
		0	The period register (TBPRD) is loaded from its shadow register when the time-base counter, TBCTR, is equal to zero.
			A write or read to the TBPRD register accesses the shadow register.
		1	Load the TBPRD register immediately without using a shadow register.
			A write or read to the TBPRD register directly accesses the active register.
2	PHSEN		Counter Register Load From Phase Register Enable
		0	Do not load the time-base counter (TBCTR) from the time-base phase register (TBPHS)
		1	Load the time-base counter with the phase register when an EPWMxSYNCI input signal occurs or when a software synchronization is forced by the SWFSYNC bit.
1:0	CTRMODE		Counter Mode
			The time-base counter mode is normally configured once and not changed during normal operation. If you change the mode of the counter, the change will take effect at the next TBCLK edge and the current counter value shall increment or decrement from the value before the mode change.
			These bits set the time-base counter mode of operation as follows:
		00	Up-count mode
		01	Down-count mode
		10	Up-down-count mode
		11	Stop-freeze counter operation (default on reset)

Figure 4-5. Time-Base Status Register (TBSTS)

15		8
	Reserved	
	R-0	

7		3	2	1	0
	Reserved		CTRMAX	SYNCI	CTRDIR
	R-0		R/W1C-0	R/W1C-0	R-1
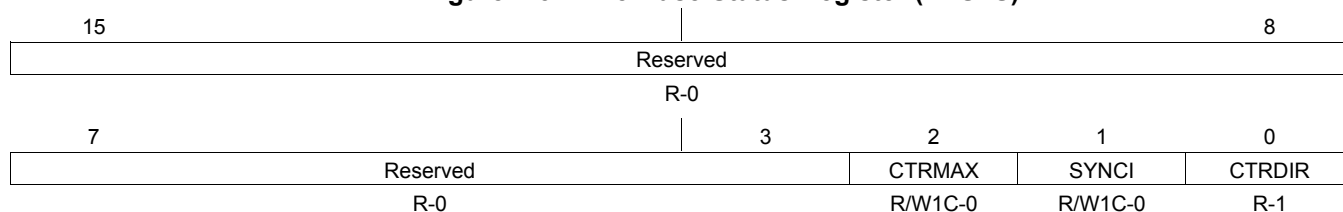
LEGEND: R/W = Read/Write; R = Read only; R/W1C = Read/Write 1 to clear; -n = value after reset

Table 4-5. Time-Base Status Register (TBSTS) Field Descriptions

Bit	Field	Value	Description
15:3	Reserved		Reserved
2	CTRMAX		Time-Base Counter Max Latched Status Bit
		0	Reading a 0 indicates the time-base counter never reached its maximum value. Writing a 0 will have no effect.
		1	Reading a 1 on this bit indicates that the time-base counter reached the max value 0xFFFF. Writing a 1 to this bit will clear the latched event.
1	SYNCI		Input Synchronization Latched Status Bit
		0	Writing a 0 will have no effect. Reading a 0 indicates no external synchronization event has occurred.
		1	Reading a 1 on this bit indicates that an external synchronization event has occurred (EPWMxSYNCI). Writing a 1 to this bit will clear the latched event.
0	CTRDIR		Time-Base Counter Direction Status Bit. At reset, the counter is frozen; therefore, this bit has no meaning. To make this bit meaningful, you must first set the appropriate mode via TBCTL[CTRMODE].
		0	Time-Base Counter is currently counting down.
		1	Time-Base Counter is currently counting up.

4.2 Counter-Compare Submodule Registers

Figure 4-6 through Figure 4-8 and Table 4-6 through Table 4-8 illustrate the counter-compare submodule control and status registers.

Figure 4-6. Counter-Compare A Register (CMPA)

15	0
CMPA	
R/W-0	
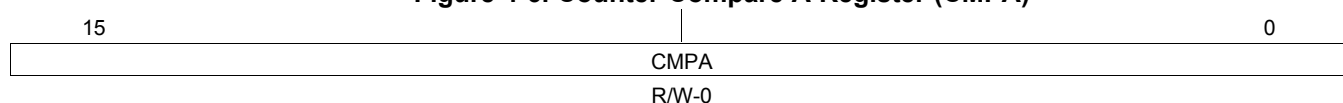
LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 4-6. 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.

Figure 4-7. Counter-Compare B Register (CMPB)

15	0
CMPB	

R/W-0
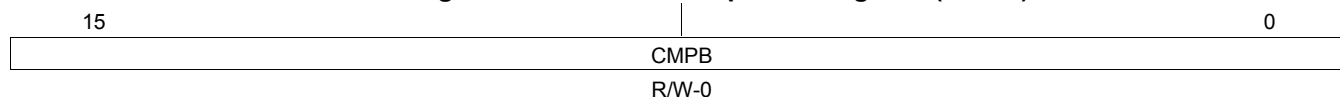
LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 4-7. Counter-Compare B Register (CMPB) Field Descriptions

Bits	Name	Description
15-0	CMPB	The value in the active CMPB 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 B" 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. 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[SHDWBMODE] bit. By default this register is shadowed. • If CMPCTL[SHDWBMODE] = 0, then the shadow is enabled and any write or read will automatically go to the shadow register. In this case, the CMPCTL[LOADBMODE] bit field determines which event will load the active register from the shadow register: • Before a write, the CMPCTL[SHDWBFULL] bit can be read to determine if the shadow register is currently full. • If CMPCTL[SHDWBMODE] = 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.

Figure 4-8. Counter-Compare Control Register (CMPCTL)

15					10	9	8
Reserved						SHDWBFULL	SHDWAFULL
R-0						R-0	R-0

7	6	5	4	3	2	1	0
Reserved	SHDWBMODE	Reserved	SHDWAMODE	LOADBMODE		LOADAMODE	
R-0	R/W-0	R-0	R/W-0	R/W-0		R/W-0	

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

Table 4-8. Counter-Compare Control Register (CMPCTL) Field Descriptions

Bits	Name	Value	Description
15-10	Reserved		Reserved
9	SHDWBFULL		Counter-compare B (CMPB) Shadow Register Full Status Flag
			This bit self clears once a load-strobe occurs.
		0	CMPB shadow FIFO not full yet
		1	Indicates the CMPB shadow FIFO is full; a CPU write will overwrite current shadow value.
8	SHDWAFULL		Counter-compare A (CMPA) Shadow Register Full Status Flag
			The flag bit is set when a 32-bit write to CMPA:CMPAHR register or a 16-bit write to CMPA register is made. A 16-bit write to CMPAHR register will not affect the flag.
			This bit self clears once a load-strobe occurs.
		0	CMPA shadow FIFO not full yet
		1	Indicates the CMPA shadow FIFO is full, a CPU write will overwrite the current shadow value.
7	Reserved		Reserved
6	SHDWBMODE		Counter-compare B (CMPB) Register Operating Mode
		0	Shadow mode. Operates as a double buffer. All writes via the CPU access the shadow register.
		1	Immediate mode. Only the active compare B register is used. All writes and reads directly access the active register for immediate compare action.
5	Reserved		Reserved
4	SHDWAMODE		Counter-compare A (CMPA) Register Operating Mode
		0	Shadow mode. Operates as a double buffer. All writes via the CPU access the shadow register.
		1	Immediate mode. Only the active compare register is used. All writes and reads directly access the active register for immediate compare action
3-2	LOADBMODE		Active Counter-Compare B (CMPB) Load From Shadow Select Mode This bit has no effect in immediate mode (CMPCTL[SHDWBMODE] = 1).
		00	Load on CTR = Zero: Time-base counter equal to zero (TBCTR = 0x0000)
		01	Load on CTR = PRD: Time-base counter equal to period (TBCTR = TBPRD)
		10	Load on either CTR = Zero or CTR = PRD
		11	Freeze (no loads possible)
1-0	LOADAMODE		Active Counter-Compare A (CMPA) Load From Shadow Select Mode. This bit has no effect in immediate mode (CMPCTL[SHDWAMODE] = 1).
		00	Load on CTR = Zero: Time-base counter equal to zero (TBCTR = 0x0000)
		01	Load on CTR = PRD: Time-base counter equal to period (TBCTR = TBPRD)
		10	Load on either CTR = Zero or CTR = PRD
		11	Freeze (no loads possible)

4.3 Action-Qualifier Submodule Registers

Figure 4-9 through Figure 4-12 and Table 4-9 through Table 4-12 provide the action-qualifier submodule register definitions.

Figure 4-9. Action-Qualifier Output A Control Register (AQCTLA)

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		RW-0	

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

Table 4-9. Action-Qualifier Output A Control Register (AQCTLA) Field Descriptions

Bits	Name	Value	Description
15-12	Reserved		Reserved
11-10	CBD		Action when the time-base counter equals the active CMPB register and the counter is decrementing.
		00	Do nothing (action disabled)
		01	Clear: force EPWMxA output low.
		10	Set: force EPWMxA output high.
		11	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.
		00	Do nothing (action disabled)
		01	Clear: force EPWMxA output low.
		10	Set: force EPWMxA output high.
		11	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.
		00	Do nothing (action disabled)
		01	Clear: force EPWMxA output low.
		10	Set: force EPWMxA output high.
		11	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.
		00	Do nothing (action disabled)
		01	Clear: force EPWMxA output low.
		10	Set: force EPWMxA output high.
		11	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.
		00	Do nothing (action disabled)
		01	Clear: force EPWMxA output low.
		10	Set: force EPWMxA output high.
		11	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.
		00	Do nothing (action disabled)
		01	Clear: force EPWMxA output low.
		10	Set: force EPWMxA output high.
		11	Toggle EPWMxA output: low output signal will be forced high, and a high signal will be forced low.

Figure 4-10. Action-Qualifier Output B Control Register (AQCTLB)

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

Table 4-10. Action-Qualifier Output B Control Register (AQCTLB) Field Descriptions

Bits	Name	Value	Description
15-12	Reserved		
11-10	CBD		Action when the counter equals the active CMPB register and the counter is decrementing.
		00	Do nothing (action disabled)
		01	Clear: force EPWMxB output low.
		10	Set: force EPWMxB output high.
		11	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.
		00	Do nothing (action disabled)
		01	Clear: force EPWMxB output low.
		10	Set: force EPWMxB output high.
		11	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.
		00	Do nothing (action disabled)
		01	Clear: force EPWMxB output low.
		10	Set: force EPWMxB output high.
		11	Toggle EPWMxB 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.
		00	Do nothing (action disabled)
		01	Clear: force EPWMxB output low.
		10	Set: force EPWMxB output high.
		11	Toggle EPWMxB 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.
		00	Do nothing (action disabled)
		01	Clear: force EPWMxB output low.
		10	Set: force EPWMxB output high.
		11	Toggle EPWMxB 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.
		00	Do nothing (action disabled)
		01	Clear: force EPWMxB output low.
		10	Set: force EPWMxB output high.
		11	Toggle EPWMxB output: low output signal will be forced high, and a high signal will be forced low.

Figure 4-11. Action-Qualifier Software Force Register (AQSFRC)

15							8
Reserved							
R-0							

7	6	5	4	3	2	1	0
RLDCSF		OTSFB	ACTSFB		OTSFA	ACTSFA	
R/W-0		R/W-0	R/W-0		R/W-0	R/W-0	
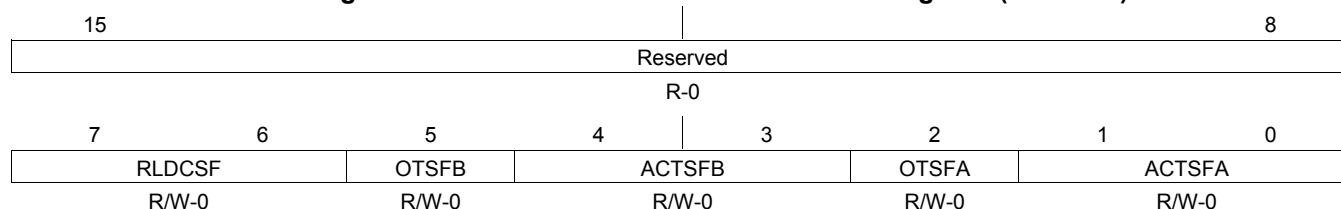
LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 4-11. Action-Qualifier Software Force Register (AQSFRC) Field Descriptions

Bit	Field	Value	Description
15:8	Reserved		
7:6	RLDCSF		AQCSFRC Active Register Reload From Shadow Options
		00	Load on event counter equals zero
		01	Load on event counter equals period
		10	Load on event counter equals zero or counter equals period
		11	Load immediately (the active register is directly accessed by the CPU and is not loaded from the shadow register).
5	OTSFB		One-Time Software Forced Event on Output B
		0	Writing a 0 (zero) has no effect. Always reads back a 0
			This bit is auto cleared once a write to this register is complete, i.e., a forced event is initiated.)
			This is a one-shot forced event. It can be overridden by another subsequent event on output B.
		1	Initiates a single s/w forced event
4:3	ACTSFB		Action when One-Time Software Force B Is invoked
		00	Does nothing (action disabled)
		01	Clear (low)
		10	Set (high)
		11	Toggle (Low -> High, High -> Low)
			Note: This action is not qualified by counter direction (CNT_dir)
2	OTSFA		One-Time Software Forced Event on Output A
		0	Writing a 0 (zero) has no effect. Always reads back a 0.
			This bit is auto cleared once a write to this register is complete (i.e., a forced event is initiated).
		1	Initiates a single software forced event
1:0	ACTSFA		Action When One-Time Software Force A Is Invoked
		00	Does nothing (action disabled)
		01	Clear (low)
		10	Set (high)
		11	Toggle (Low → High, High → Low)
			Note: This action is not qualified by counter direction (CNT_dir)

Figure 4-12. Action-Qualifier Continuous Software Force Register (AQCSFRC)

15							8
Reserved							
R-0							

7			4	3	2	1	0
Reserved				CSFB		CSFA	
R-0				R/W-0		R/W-0	
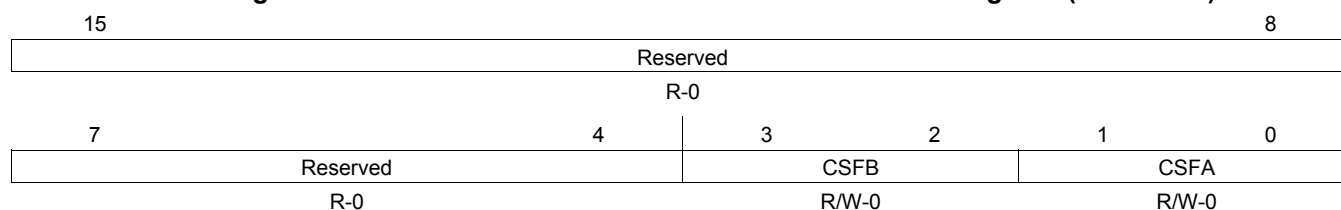
LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 4-12. Action-qualifier Continuous Software Force Register (AQCSFRC) Field Descriptions

Bits	Name	Value	Description
15-4	Reserved		Reserved
3-2	CSFB		Continuous Software Force on Output B
			In immediate mode, a continuous force takes effect on the next TBCLK edge.
			In shadow mode, a continuous force takes effect on the next TBCLK edge after a shadow load into the active register. To configure shadow mode, use AQSFRC[RLDCSF].
		00	Forcing disabled, i.e., has no effect
		01	Forces a continuous low on output B
		10	Forces a continuous high on output B
		11	Software forcing is disabled and has no effect
1-0	CSFA		Continuous Software Force on Output A
			In immediate mode, a continuous force takes effect on the next TBCLK edge.
			In shadow mode, a continuous force takes effect on the next TBCLK edge after a shadow load into the active register.
		00	Forcing disabled, i.e., has no effect
		01	Forces a continuous low on output A
		10	Forces a continuous high on output A
		11	Software forcing is disabled and has no effect

4.4 Dead-Band Submodule Registers

Figure 4-13 through Figure 4-15 and Table 4-13 through Table 4-15 provide the register definitions.

Figure 4-13. Dead-Band Generator Control Register (DBCTL)

15							8
Reserved							
R-0							

7	6	5	4	3	2	1	0
Reserved		IN_MODE		POLSEL		OUT_MODE	
R-0		R/W-0		R/W-0		R/W-0	
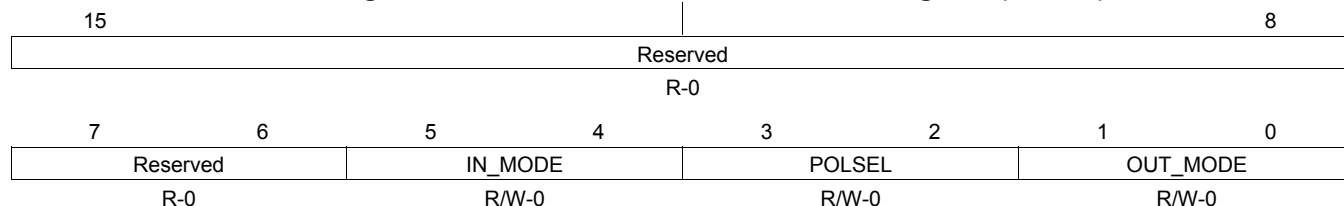
LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 4-13. Dead-Band Generator Control Register (DBCTL) Field Descriptions

Bits	Name	Value	Description
15-6	Reserved		Reserved
5-4	IN_MODE		Dead Band Input Mode Control
			Bit 5 controls the S5 switch and bit 4 controls the S4 switch shown in Figure 2-28.
			This allows you to select the input source to the falling-edge and rising-edge delay.
			To produce classical dead-band waveforms the default is EPWMxA In is the source for both falling and rising-edge delays.
		00	EPWMxA In (from the action-qualifier) is the source for both falling-edge and rising-edge delay.
		01	EPWMxB In (from the action-qualifier) is the source for rising-edge delayed signal.
			EPWMxA In (from the action-qualifier) is the source for falling-edge delayed signal.
		10	EPWMxA In (from the action-qualifier) is the source for rising-edge delayed signal.
			EPWMxB In (from the action-qualifier) is the source for falling-edge delayed signal.
		11	EPWMxB In (from the action-qualifier) is the source for both rising-edge delay and falling-edge delayed signal.
3-2	POLSEL		Polarity Select Control
			Bit 3 controls the S3 switch and bit 2 controls the S2 switch shown in Figure 2-28.
			This allows you to selectively invert one of the delayed signals before it is sent out of the dead-band submodule.
			The following descriptions correspond to classical upper/lower switch control as found in one leg of a digital motor control inverter.
			These assume that DBCTL[OUT_MODE] = 1,1 and DBCTL[IN_MODE] = 0,0. Other enhanced modes are also possible, but not regarded as typical usage modes.
		00	Active high (AH) mode. Neither EPWMxA nor EPWMxB is inverted (default).
		01	Active low complementary (ALC) mode. EPWMxA is inverted.
		10	Active high complementary (AHC). EPWMxB is inverted.
		11	Active low (AL) mode. Both EPWMxA and EPWMxB are inverted.
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 2-28.
			This allows you to selectively enable or bypass the dead-band generation for the falling-edge and rising-edge delay.
		00	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.
		01	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].
		10	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.
		11	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].

Figure 4-14. Dead-Band Generator Rising Edge Delay Register (DBRED)

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 4-14. Dead-Band Generator Rising Edge Delay Register (DBRED) Field Descriptions

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

Figure 4-15. Dead-Band Generator Falling Edge Delay Register (DBFED)

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

7	0
DEL	
R/W-0	
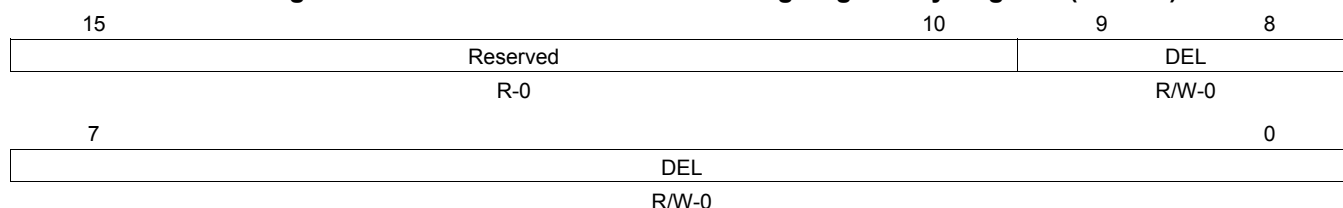
LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 4-15. 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

4.5 PWM-Chopper Submodule Control Register

and provide the definitions for the PWM-chopper submodule control register.

Figure 4-16. PWM-Chopper Control Register (PCCTL)

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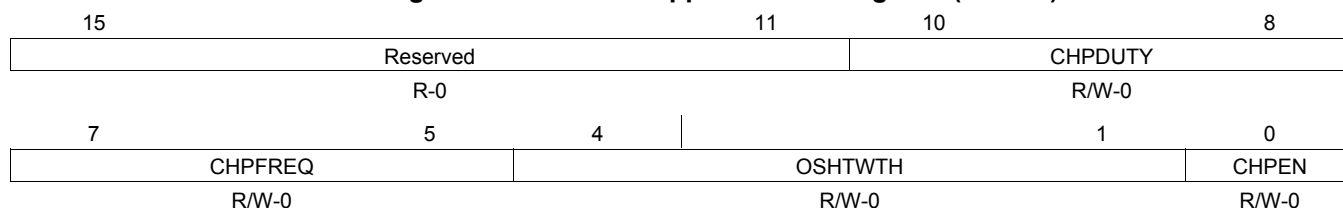
LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 4-16. PWM-Chopper Control Register (PCCTL) Bit Descriptions

Bits	Name	Value	Description
15-11	Reserved		Reserved

Table 4-16. PWM-Chopper Control Register (PCCTL) Bit Descriptions (continued)

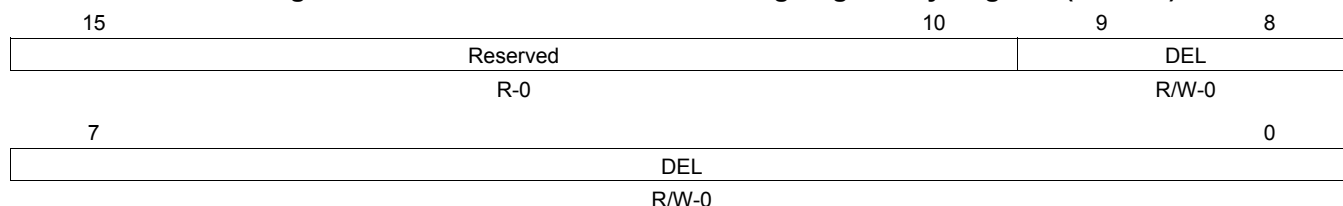| Bits	Name	Value	Description
10-8	CHPDUTY		Chopping Clock Duty Cycle
		000	Duty = 1/8 (12.5%)
		001	Duty = 2/8 (25.0%)
		010	Duty = 3/8 (37.5%)
		011	Duty = 4/8 (50.0%)
		100	Duty = 5/8 (62.5%)
		101	Duty = 6/8 (75.0%)
		110	Duty = 7/8 (87.5%)
		111	Reserved
7:5	CHPFREQ		Chopping Clock Frequency
		000	Divide by 1 (no prescale, = 12.5 MHz at 100 MHz SYSCLKOUT)
		001	Divide by 2 (6.25 MHz at 100 MHz SYSCLKOUT)
		010	Divide by 3 (4.16 MHz at 100 MHz SYSCLKOUT)
		011	Divide by 4 (3.12 MHz at 100 MHz SYSCLKOUT)
		100	Divide by 5 (2.50 MHz at 100 MHz SYSCLKOUT)
		101	Divide by 6 (2.08 MHz at 100 MHz SYSCLKOUT)
		110	Divide by 7 (1.78 MHz at 100 MHz SYSCLKOUT)
		111	Divide by 8 (1.56 MHz at 100 MHz SYSCLKOUT)
4:1	OSHTWTH		One-Shot Pulse Width
		0000	1 x SYSCLKOUT / 8 wide (= 80 nS at 100 MHz SYSCLKOUT)
		0001	2 x SYSCLKOUT / 8 wide (= 160 nS at 100 MHz SYSCLKOUT)
		0010	3 x SYSCLKOUT / 8 wide (= 240 nS at 100 MHz SYSCLKOUT)
		0011	4 x SYSCLKOUT / 8 wide (= 320 nS at 100 MHz SYSCLKOUT)
		0100	5 x SYSCLKOUT / 8 wide (= 400 nS at 100 MHz SYSCLKOUT)
		0101	6 x SYSCLKOUT / 8 wide (= 480 nS at 100 MHz SYSCLKOUT)
		0110	7 x SYSCLKOUT / 8 wide (= 560 nS at 100 MHz SYSCLKOUT)
		0111	8 x SYSCLKOUT / 8 wide (= 640 nS at 100 MHz SYSCLKOUT)
		1000	9 x SYSCLKOUT / 8 wide (= 720 nS at 100 MHz SYSCLKOUT)
		1001	10 x SYSCLKOUT / 8 wide (= 800 nS at 100 MHz SYSCLKOUT)
		1010	11 x SYSCLKOUT / 8 wide (= 880 nS at 100 MHz SYSCLKOUT)
		1011	12 x SYSCLKOUT / 8 wide (= 960 nS at 100 MHz SYSCLKOUT)
		1100	13 x SYSCLKOUT / 8 wide (= 1040 nS at 100 MHz SYSCLKOUT)
		1101	14 x SYSCLKOUT / 8 wide (= 1120 nS at 100 MHz SYSCLKOUT)
		1110	15 x SYSCLKOUT / 8 wide (= 1200 nS at 100 MHz SYSCLKOUT)
		1111	16 x SYSCLKOUT / 8 wide (= 1280 nS at 100 MHz SYSCLKOUT)
0	CHPEN		PWM-chopping Enable
		0	Disable (bypass) PWM chopping function
		1	Enable chopping function

4.6 Trip-Zone Submodule Control and Status Registers

Figure 4-17 and Table 4-17 provide the trip-zone control and status register definitions.

Figure 4-17. Trip-Zone Select Register (TZSEL)

15	14	13	12	11	10	9	8
Reserved		OSHT6	OSHT5	OSHT4	OSHT3	OSHT2	OSHT1
R-0		R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

7	6	5	4	3	2	1	0
Reserved		CBC6	CBC5	CBC4	CBC3	CBC2	CBC1
R-0		R/W-0	R/W-0	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 4-17. Trip-Zone Submodule Select Register (TZSEL) Field Descriptions

Bits	Name	Value	Description
One-Shot (OSHT) Trip-zone enable/disable. When any of the enabled pins go low, a one-shot trip event occurs for this ePWM module. When the event occurs, the action defined in the TZCTL register (Table 4-18) is taken on the EPWMxA and EPWMxB outputs. The one-shot trip condition remains latched until the user clears the condition via the TZCLR register (Table 4-21).			
15:14	Reserved		Reserved
13	OSHT6		Trip-zone 6 ($\overline{TZ6}$) Select
		0	Disable $\overline{TZ6}$ as a one-shot trip source for this ePWM module.
		1	Enable $\overline{TZ6}$ as a one-shot trip source for this ePWM module.
12	OSHT5		Trip-zone 5 ($\overline{TZ5}$) Select
		0	Disable $\overline{TZ5}$ as a one-shot trip source for this ePWM module
		1	Enable $\overline{TZ5}$ as a one-shot trip source for this ePWM module
11	OSHT4		Trip-zone 4 ($\overline{TZ4}$) Select
		0	Disable $\overline{TZ4}$ as a one-shot trip source for this ePWM module
		1	Enable $\overline{TZ4}$ as a one-shot trip source for this ePWM module
10	OSHT3		Trip-zone 3 ($\overline{TZ3}$) Select
		0	Disable $\overline{TZ3}$ as a one-shot trip source for this ePWM module
		1	Enable $\overline{TZ3}$ as a one-shot trip source for this ePWM module
9	OSHT2		Trip-zone 2 ($\overline{TZ2}$) Select
		0	Disable $\overline{TZ2}$ as a one-shot trip source for this ePWM module
		1	Enable $\overline{TZ2}$ as a one-shot trip source for this ePWM module
8	OSHT1		Trip-zone 1 ($\overline{TZ1}$) Select
		0	Disable $\overline{TZ1}$ as a one-shot trip source for this ePWM module
		1	Enable $\overline{TZ1}$ as a one-shot trip source for this ePWM module
Cycle-by-Cycle (CBC) Trip-zone enable/disable. When any of the enabled pins go low, a cycle-by-cycle trip event occurs for this ePWM module. When the event occurs, the action defined in the TZCTL register (Table 4-18) is taken on the EPWMxA and EPWMxB outputs. A cycle-by-cycle trip condition is automatically cleared when the time-base counter reaches zero.			
7:6	Reserved		Reserved
5	CBC6		Trip-zone 6 ($\overline{TZ6}$) Select
		0	Disable $\overline{TZ6}$ as a CBC trip source for this ePWM module
		1	Enable $\overline{TZ6}$ as a CBC trip source for this ePWM module
4	CBC5		Trip-zone 5 ($\overline{TZ5}$) Select
		0	Disable $\overline{TZ5}$ as a CBC trip source for this ePWM module
		1	Enable $\overline{TZ5}$ as a CBC trip source for this ePWM module
3	CBC4		Trip-zone 4 ($\overline{TZ4}$) Select
		0	Disable $\overline{TZ4}$ as a CBC trip source for this ePWM module
		1	Enable $\overline{TZ4}$ as a CBC trip source for this ePWM module
2	CBC3		Trip-zone 3 ($\overline{TZ3}$) Select
		0	Disable $\overline{TZ3}$ as a CBC trip source for this ePWM module
		1	Enable $\overline{TZ3}$ as a CBC trip source for this ePWM module

Table 4-17. Trip-Zone Submodule Select Register (TZSEL) Field Descriptions   (continued)

Bits	Name	Value	Description
1	CBC2		Trip-zone 2 ($\overline{TZ2}$) Select
		0	Disable $\overline{TZ2}$ as a CBC trip source for this ePWM module
		1	Enable $\overline{TZ2}$ as a CBC trip source for this ePWM module
0	CBC1		Trip-zone 1 ($\overline{TZ1}$) Select
		0	Disable $\overline{TZ1}$ as a CBC trip source for this ePWM module
		1	Enable $\overline{TZ1}$ as a CBC trip source for this ePWM module

Figure 4-18. Trip-Zone Control Register (TZCTL)

15						8
Reserved						
R-0						

7			4	3	2	1	0
Reserved				TZB		TZA	
R-0				R/W-0		R/W-0	

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

Table 4-18. Trip-Zone Control Register (TZCTL) Field Descriptions

Bits	Name	Value	Description
15–4	Reserved		Reserved
3–2	TZB		When a trip event occurs the following action is taken on output EPWMxB. Which trip-zone pins can cause an event is defined in the TZSEL register (Table 4-17).
		00	High impedance (EPWMxB = High-impedance state)
		01	Force EPWMxB to a high state
		10	Force EPWMxB to a low state
		11	Do nothing, no action is taken on EPWMxB.
1–0	TZA		When a trip event occurs the following action is taken on output EPWMxA. Which trip-zone pins can cause an event is defined in the TZSEL register (Table 4-17).
		00	High impedance (EPWMxA = High-impedance state)
		01	Force EPWMxA to a high state
		10	Force EPWMxA to a low state
		11	Do nothing, no action is taken on EPWMxA.

Figure 4-19. Trip-Zone Enable Interrupt Register (TZEINT)

15						8
Reserved						
R -0						

7		3	2	1	0
Reserved			OST	CBC	Reserved
R-0			R/W-0	R/W-0	R-0

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

Table 4-19. Trip-Zone Enable Interrupt Register (TZEINT) Field Descriptions

Bits	Name	Value	Description
15-3	Reserved		Reserved
2	OST		Trip-zone One-Shot Interrupt Enable
		0	Disable one-shot interrupt generation

Table 4-19. Trip-Zone Enable Interrupt Register (TZEINT) Field Descriptions (continued)

Bits	Name	Value	Description
		1	Enable Interrupt generation; a one-shot trip event will cause a EPWMx_TZINT PIE interrupt. [1]
1	CBC		Trip-zone Cycle-by-Cycle Interrupt Enable
		0	Disable cycle-by-cycle interrupt generation.
		1	Enable interrupt generation; a cycle-by-cycle trip event will cause an EPWMx_TZINT PIE interrupt. [1]
0	Reserved		Reserved

[1]   The Peripheral Interrupt Expansion (PIE) module is described in the specific device version of the *System Control and Interrupts Reference Guide* listed in Section 1.

Figure 4-20. Trip-Zone Flag Register (TZFLG)

15							8
Reserved							
R-0							

7			3	2	1	0
Reserved				OST	CBC	INT
R-0				R-0	R-0	R-0

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

Table 4-20. Trip-Zone Flag Register (TZFLG) Field Descriptions

Bits	Name	Value	Description
15-3	Reserved		Reserved
2	OST		Latched Status Flag for A One-Shot Trip Event
		0	No one-shot trip event has occurred.
		1	Indicates a trip event has occurred on a pin selected as a one-shot trip source.
			This bit is cleared by writing the appropriate value to the TZCLR register (Table 4-21).
1	CBC		Latched Status Flag for Cycle-By-Cycle Trip Event
		0	No cycle-by-cycle trip event has occurred.
		1	Indicates a trip event has occurred on a pin selected as a cycle-by-cycle trip source. The TZFLG[CBC] bit will remain set until it is manually cleared by the user. If the cycle-by-cycle trip event is still present when the CBC bit is cleared, then CBC will be immediately set again. The specified condition on the pins is automatically cleared when the ePWM time-base counter reaches zero (TBCTR = 0x0000) if the trip condition is no longer present. The condition on the pins is only cleared when the TBCTR = 0x0000 no matter where in the cycle the CBC flag is cleared.
			This bit is cleared by writing the appropriate value to the TZCLR register (Table 4-21).
0	INT		Latched Trip Interrupt Status Flag
		0	Indicates no interrupt has been generated.
		1	Indicates an EPWMx_TZINT PIE interrupt was generated because of a trip condition.
			No further EPWMx_TZINT PIE interrupts will be generated until this flag is cleared. If the interrupt flag is cleared when either CBC or OST is set, then another interrupt pulse will be generated. Clearing all flag bits will prevent further interrupts.
			This bit is cleared by writing the appropriate value to the TZCLR register (Table 4-21).

Figure 4-21. Trip-Zone Clear Register (TZCLR)

15							8
Reserved							
R-0							

7		3	2	1	0
Reserved			OST	CBC	INT
R-0			R/W-0	R/W-0	R/W-0

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

Table 4-21. Trip-Zone Clear Register (TZCLR) Field Descriptions

Bits	Name	Value	Description
15-3	Reserved		Reserved
2	OST		Clear Flag for One-Shot Trip (OST) Latch
		0	Has no effect. Always reads back a 0.
		1	Clears this Trip (set) condition.
1	CBC		Clear Flag for Cycle-By-Cycle (CBC) Trip Latch
		0	Has no effect. Always reads back a 0.
		1	Clears this Trip (set) condition.
0	INT		Global Interrupt Clear Flag
		0	Has no effect. Always reads back a 0.
		1	Clears the trip-interrupt flag for this ePWM module (TZFLG[INT]).
			NOTE: No further EPWMx_TZINT PIE interrupts will be generated until the flag is cleared. If the TZFLG[INT] bit is cleared and any of the other flag bits are set, then another interrupt pulse will be generated. Clearing all flag bits will prevent further interrupts.

Figure 4-22. Trip-Zone Force Register (TZFRC)

15							8
Reserved							
R-0							

7		3	2	1	0
Reserved			OST	CBC	Reserved
R-0			R/W-0	R/W-0	R- 0

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

Table 4-22. Trip-Zone Force Register (TZFRC) Field Descriptions

Bits	Name	Value	Description
15-3	Reserved		Reserved
2	OST		Force a One-Shot Trip Event via Software
		0	Writing of 0 is ignored. Always reads back a 0.
		1	Forces a one-shot trip event and sets the TZFLG[OST] bit.
1	CBC		Force a Cycle-by-Cycle Trip Event via Software
		0	Writing of 0 is ignored. Always reads back a 0.
		1	Forces a cycle-by-cycle trip event and sets the TZFLG[CBC] bit.
0	Reserved		Reserved

4.7 Event-Trigger Submodule Registers

Figure 4-23 through Figure 4-27 and Table 4-23 through Table 4-27 describe the registers for the event-trigger submodule.

Figure 4-23. Event-Trigger Selection Register (ETSEL)

15	14		12	11	10		8
SOCBEN	SOCBSEL			SOCAEN	SOCASEL		
R/W-0	R/W-0			R/W-0	R/W-0		

7			4	3	2		0
Reserved				INTEN	INTSEL		
R-0				R/W-0	R/W-0		

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

Table 4-23. Event-Trigger Selection Register (ETSEL) Field Descriptions

Bits	Name	Value	Description
15	SOCBEN		Enable the ADC Start of Conversion B (EPWMxSOCB) Pulse
		0	Disable EPWMxSOCB.
		1	Enable EPWMxSOCB pulse.
14-12	SOCBSEL		EPWMxSOCB Selection Options
			These bits determine when a EPWMxSOCB pulse will be generated.
		000	Reserved
		001	Enable event time-base counter equal to zero. (TBCTR = 0x0000)
		010	Enable event time-base counter equal to period (TBCTR = TBPRD)
		011	Reserved
		100	Enable event time-base counter equal to CMPA when the timer is incrementing.
		101	Enable event time-base counter equal to CMPA when the timer is decrementing.
		110	Enable event: time-base counter equal to CMPB when the timer is incrementing.
		111	Enable event: time-base counter equal to CMPB when the timer is decrementing.
11	SOCAEN		Enable the ADC Start of Conversion A (EPWMxSOCA) Pulse
		0	Disable EPWMxSOCA.
		1	Enable EPWMxSOCA pulse.
10-8	SOCASEL		EPWMxSOCA Selection Options
			These bits determine when a EPWMxSOCA pulse will be generated.
		000	Reserved
		001	Enable event time-base counter equal to zero. (TBCTR = 0x0000)
		010	Enable event time-base counter equal to period (TBCTR = TBPRD)
		011	Reserved
		100	Enable event time-base counter equal to CMPA when the timer is incrementing.
		101	Enable event time-base counter equal to CMPA when the timer is decrementing.
		110	Enable event: time-base counter equal to CMPB when the timer is incrementing.
		111	Enable event: time-base counter equal to CMPB when the timer is decrementing.
7-4	Reserved		Reserved
3	INTEN		Enable ePWM Interrupt (EPWMx_INT) Generation
		0	Disable EPWMx_INT generation
		1	Enable EPWMx_INT generation

Table 4-23. Event-Trigger Selection Register (ETSEL) Field Descriptions (continued)

Bits	Name	Value	Description
2-0	INTSEL		ePWM Interrupt (EPWMx_INT) Selection Options
		000	Reserved
		001	Enable event time-base counter equal to zero. (TBCTR = 0x0000)
		010	Enable event time-base counter equal to period (TBCTR = TBPRD)
		011	Reserved
		100	Enable event time-base counter equal to CMPA when the timer is incrementing.
		101	Enable event time-base counter equal to CMPA when the timer is decrementing.
		110	Enable event: time-base counter equal to CMPB when the timer is incrementing.
		111	Enable event: time-base counter equal to CMPB when the timer is decrementing.

Figure 4-24. Event-Trigger Prescale Register (ETPS)

15	14	13	12	11	10	9	8
SOCBCNT		SOCBPRD		SOCACNT		SOCAPRD	
R-0		R/W-0		R-0		R/W-0	

7			4	3	2	1	0
Reserved				INTCNT		INTPRD	
R-0				R-0		R/W-0	

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

Table 4-24. Event-Trigger Prescale Register (ETPS) Field Descriptions

Bits	Name		Description
15-14	SOCBCNT		ePWM ADC Start-of-Conversion B Event (EPWMxSOCB) Counter Register
			These bits indicate how many selected ETSEL[SOCBSEL] events have occurred:
		00	No events have occurred.
		01	1 event has occurred.
		10	2 events have occurred.
		11	3 events have occurred.
13-12	SOCBPRD		ePWM ADC Start-of-Conversion B Event (EPWMxSOCB) Period Select
			These bits determine how many selected ETSEL[SOCBSEL] events need to occur before an EPWMxSOCB pulse is generated. To be generated, the pulse must be enabled (ETSEL[SOCBEN] = 1). The SOCB pulse will be generated even if the status flag is set from a previous start of conversion (ETFLG[SOCB] = 1). Once the SOCB pulse is generated, the ETPS[SOCBCNT] bits will automatically be cleared.
		00	Disable the SOCB event counter. No EPWMxSOCB pulse will be generated
		01	Generate the EPWMxSOCB pulse on the first event: ETPS[SOCBCNT] = 0,1
		10	Generate the EPWMxSOCB pulse on the second event: ETPS[SOCBCNT] = 1,0
		11	Generate the EPWMxSOCB pulse on the third event: ETPS[SOCBCNT] = 1,1
11-10	SOCACNT		ePWM ADC Start-of-Conversion A Event (EPWMxSOCA) Counter Register
			These bits indicate how many selected ETSEL[SOCASEL] events have occurred:
		00	No events have occurred.
		01	1 event has occurred.
		10	2 events have occurred.
		11	3 events have occurred.

Table 4-24. Event-Trigger Prescale Register (ETPS) Field Descriptions (continued)

Bits	Name		Description
9-8	SOCAPRD		ePWM ADC Start-of-Conversion A Event (EPWMxSOCA) Period Select
			These bits determine how many selected ETSEL[SOCASEL] events need to occur before an EPWMxSOCA pulse is generated. To be generated, the pulse must be enabled (ETSEL[SOCAEN] = 1). The SOCA pulse will be generated even if the status flag is set from a previous start of conversion (ETFLG[SOCA] = 1). Once the SOCA pulse is generated, the ETPS[SOCACNT] bits will automatically be cleared.
		00	Disable the SOCA event counter. No EPWMxSOCA pulse will be generated
		01	Generate the EPWMxSOCA pulse on the first event: ETPS[SOCACNT] = 0,1
		10	Generate the EPWMxSOCA pulse on the second event: ETPS[SOCACNT] = 1,0
		11	Generate the EPWMxSOCA pulse on the third event: ETPS[SOCACNT] = 1,1
7-4	Reserved		Reserved
3-2	INTCNT		ePWM Interrupt Event (EPWMx_INT) Counter Register
			These bits indicate how many selected ETSEL[INTSEL] events have occurred. These bits are automatically cleared when an interrupt pulse is generated. If interrupts are disabled, ETSEL[INT] = 0 or the interrupt flag is set, ETFLG[INT] = 1, the counter will stop counting events when it reaches the period value ETPS[INTCNT] = ETPS[INTPRD].
		00	No events have occurred.
		01	1 event has occurred.
		10	2 events have occurred.
		11	3 events have occurred.
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.
		00	Disable the interrupt event counter. No interrupt will be generated and ETFRC[INT] is ignored.
		01	Generate an interrupt on the first event INTCNT = 01 (first event)
		10	Generate interrupt on ETPS[INTCNT] = 1,0 (second event)
		11	Generate interrupt on ETPS[INTCNT] = 1,1 (third event)

Figure 4-25. Event-Trigger Flag Register (ETFLG)

15					8
		Reserved			
		R-0			

7		4	3	2	1	0
	Reserved		SOCB	SOCA	Reserved	INT
	R-0		R-0	R-0	R-0	R-0
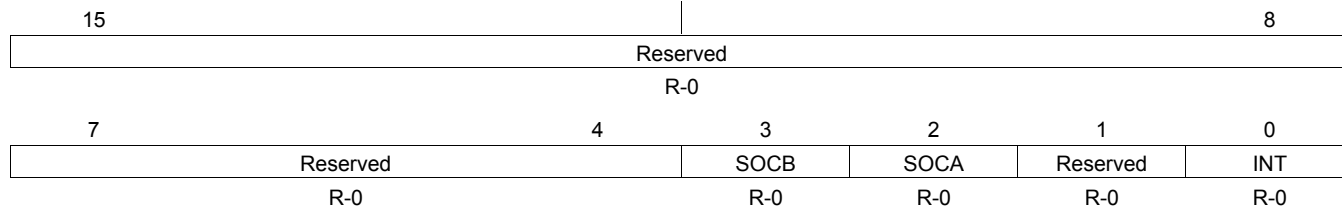
LEGEND: R/W = Read/Write; R = Read only; -n = value after reset

Table 4-25. Event-Trigger Flag Register (ETFLG) Field Descriptions

Bits	Name	Value	Description
15-4	Reserved		Reserved
3	SOCB		Latched ePWM ADC Start-of-Conversion B (EPWMxSOCB) Status Flag
		0	Indicates no EPWMxSOCB event occurred
		1	Indicates that a start of conversion pulse was generated on EPWMxSOCB. The EPWMxSOCB output will continue to be generated even if the flag bit is set.
2	SOCA		Latched ePWM ADC Start-of-Conversion A (EPWMxSOCA) Status Flag
			Unlike the ETFLG[INT] flag, the EPWMxSOCA output will continue to pulse even if the flag bit is set.
		0	Indicates no event occurred
		1	Indicates that a start of conversion pulse was generated on EPWMxSOCA. The EPWMxSOCA output will continue to be generated even if the flag bit is set.
1	Reserved		Reserved
0	INT		Latched ePWM Interrupt (EPWMx_INT) Status Flag
		0	Indicates no event occurred
		1	Indicates that an ePWMx interrupt (EWPMx_INT) was generated. No further interrupts will be generated until the flag bit is cleared. Up to one interrupt can be pending while the ETFLG[INT] bit is still set. If an interrupt is pending, it will not be generated until after the ETFLG[INT] bit is cleared. Refer to Figure 2-41.

Figure 4-26. Event-Trigger Clear Register (ETCLR)

15					8
		Reserved			
		R = 0			

7		4	3	2	1	0
	Reserved		SOCB	SOCA	Reserved	INT
	R-0		R/W-0	R/W-0	R-0	R/W-0

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

Table 4-26. Event-Trigger Clear Register (ETCLR) Field Descriptions

Bits	Name	Value	Description
15-4	Reserved		Reserved
3	SOCB		ePWM ADC Start-of-Conversion B (EPWMxSOCB) Flag Clear Bit
		0	Writing a 0 has no effect. Always reads back a 0
		1	Clears the ETFLG[SOCB] flag bit
2	SOCA		ePWM ADC Start-of-Conversion A (EPWMxSOCA) Flag Clear Bit
		0	Writing a 0 has no effect. Always reads back a 0
		1	Clears the ETFLG[SOCA] flag bit
1	Reserved		Reserved
0	INT		ePWM Interrupt (EPWMx_INT) Flag Clear Bit
		0	Writing a 0 has no effect. Always reads back a 0
		1	Clears the ETFLG[INT] flag bit and enable further interrupts pulses to be generated

4.8 Proper Interrupt Initialization Procedure

When the ePWM peripheral clock is enabled it may be possible that interrupt flags may be set due to spurious events due to the ePWM registers not being properly initialized. The proper procedure for initializing the ePWM peripheral is as follows:

1. Disable Global Interrupts (CPU INTM flag)
2. Disable ePWM Interrupts
3. Initialize Peripheral Registers
4. Clear Any Spurious ePWM Flags (including PIEIFR)
5. Enable ePWM Interrupts
6. Enable Global Interrupts

Figure 4-27. Event-Trigger Force Register (ETFRC)

15					8
Reserved					
R-0					

7	4	3	2	1	0
Reserved		SOCB	SOCA	Reserved	INT
R-0		R-0	R-0	R-0	R-0

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

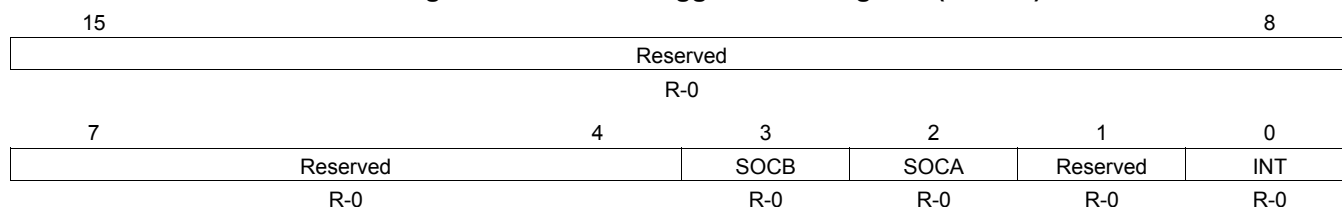#### Table 4-27. Event-Trigger Force Register (ETFRC) Field Descriptions

Bits	Name	Value	Description
15-4	Reserved		Reserved
3	SOCB		SOCB Force Bit. The SOCB pulse will only be generated if the event is enabled in the ETSEL register. The ETFLG[SOCB] flag bit will be set regardless.
		0	Has no effect. Always reads back a 0.
		1	Generates a pulse on EPWMxSOCB and sets the SOCBFLG bit. This bit is used for test purposes.
2	SOCA		SOCA Force Bit. The SOCA pulse will only be generated if the event is enabled in the ETSEL register. The ETFLG[SOCA] flag bit will be set regardless.
		0	Writing 0 to this bit will be ignored. Always reads back a 0.
		1	Generates a pulse on EPWMxSOCA and set the SOCAFLG bit. This bit is used for test purposes.
1	Reserved	0	Reserved
0	INT		INT Force Bit. The interrupt will only be generated if the event is enabled in the ETSEL register. The INT flag bit will be set regardless.
		0	Writing 0 to this bit will be ignored. Always reads back a 0.
		1	Generates an interrupt on $\overline{\text{EPWMxINT}}$ and set the INT flag bit. This bit is used for test purposes.

Revision History

This document was revised to SPRU791D from SPRU791C. The scope of the revision was limited to technical changes as shown in Table A-1.

Table A-1. Changes for Revision D

Location	Modifications, Additions, and Deletions
Figure 2-12	Modified the Detailed View of the Counter-compare Submodule figure
Section 2.2.3	Corrected register name from TBCTRL to TBCTL in the second paragraph of calculating PWM Period and Frequency
Figure 2-2	Modified figure
Figure 2-13	Modified the Counter-compare Event Waveforms in Up-count Mode figure and the Counter-compare Events in Down-count Mode figure
Figure 2-5	Modified figure
Figure 2-6	Added new figure for synchronization scheme 3
Figure 2-12	Modified figure
Figure 2-18	Modified figure
Table 2-7	Corrected register name from AQCSF to AQCSFRC in the paragraph following the Action-qualifier submodule Possible Input Events
Section 2.4.3	Corrected register name from TBCNTR to TBCTR in first paragraph and the Acton-qualifier Event Priority table
Figure 2-21	Modified figure
Figure 2-22	Modified figure
Figure 2-23	Modified figure
Figure 2-25	Modified figure
Figure 2-26	Modified figure
Section 2.6.3	Corrected register name from CHPCTL to PCCTL in the first paragraph of the section on operational highlights for the PWM-chopper Submodule
Figure 2-31	Modified figure
Figure 2-36	Modified figure
Figure 2-37	Modified figure
Table 4-24	Modified the descriptive name of the Event-Trigger Prescale Register INTPRD field
Figure 2-41	Corrected bit name ESOOCA to SOCAEN in the paragraph following the Event-trigger Interrupt Generator figure
Figure 2-42	Modified figure
Figure 2-43	Modified figure
Table 4-2	Modified the description of the Time-base Phase Register TBPHS field
Figure 4-4	Modified the reset value of the CTRMODE field in the Time-Base Control Register (TBCTL) figure
Table 4-4	Modified the bit description of SYNCOSEL in the TBCTL register field descriptions
Table 4-5	Modified two field descriptions in the TBSTS register field descriptions table
Figure 4-6	Corrected the CMPA Register figure, changing 7 to 15
Table 4-7	Modified descriptions in CMPB Field Descriptions table
Table 4-11	Changed AQCSF to AQCSFRC in the AQSFRC field descriptions table
Table 4-12	Modified description of CSFB field in the AQCSFRC field descriptions table

Table A-1. Changes for Revision D (continued)

Location	Modifications, Additions, and Deletions
Table 4-13	Modified the "10" description of the Dead-Band Generator Control Register OUT_MODE field.
Table 4-14	Modified the bit numbers of the Dead-Band Generator Rising Edge Delay Register Reserved field
Table 4-15	Modified the bit numbers of the Dead-Band Generator Falling Edge Delay Register Reserved field
Section 2.8.1	Updated description of event counter
Table 4-24	Updated INTPRD description
Table 4-25	Updated the INTFLG[INT[bit description
Table 4-27	Modified descriptions in the Event-trigger Force register field descriptions

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 TI 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. Information of third parties may be subject to additional restrictions.

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.

TI products are not authorized for use in safety-critical applications (such as life support) where a failure of the TI product would reasonably be expected to cause severe personal injury or death, unless officers of the parties have executed an agreement specifically governing such use. Buyers represent that they have all necessary expertise in the safety and regulatory ramifications of their applications, and acknowledge and agree that they are solely responsible for all legal, regulatory and safety-related requirements concerning their products and any use of TI products in such safety-critical applications, notwithstanding any applications-related information or support that may be provided by TI. Further, Buyers must fully indemnify TI and its representatives against any damages arising out of the use of TI products in such safety-critical applications.

TI products are neither designed nor intended for use in military/aerospace applications or environments unless the TI products are specifically designated by TI as military-grade or "enhanced plastic." Only products designated by TI as military-grade meet military specifications. Buyers acknowledge and agree that any such use of TI products which TI has not designated as military-grade is solely at the Buyer's risk, and that they are solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI products are neither designed nor intended for use in automotive applications or environments unless the specific TI products are designated by TI as compliant with ISO/TS 16949 requirements. Buyers acknowledge and agree that, if they use any non-designated products in automotive applications, TI will not be responsible for any failure to meet such requirements.

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
RFID	www.ti-rfid.com	Telephony	www.ti.com/telephony
Low Power Wireless	www.ti.com/lpw	Video & Imaging	www.ti.com/video
		Wireless	www.ti.com/wireless