# TMS320x280x Enhanced Pulse Width Modulator (ePWM) Module Reference Guide

Literature Number: SPRU791
November 2004

Texas Instruments

# IMPORTANT NOTICE

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

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

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

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

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

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

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

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

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

## Preface

# Read This First

### About This Manual

This guide describes the Enhanced Pulse Width Modulator 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:

***TMS320C28x DSP CPU and Instruction Set Reference Guide*** (literature number SPRU430) 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.

***TMS320F2801, TMS320F2806, TMS320F2808 Digital Signal Processors*** (literature number SPRS230) data sheet contains the pinout, signal descriptions, as well as electrical and timing specifications for the F280x devices.

***TMS320x280x Analog-to-Digital Converter (ADC) Reference Guide*** (literature number SPRU716) describes the ADC module. The module is a 12−bit pipelined ADC. The analog circuits of this converter, referred to as the core in this document, include the front-end analog multiplexers

(MUXs), sample−and−hold (S/H) circuits, the conversion core, voltage regulators, and other analog supporting circuits. Digital circuits, referred to as the wrapper in this document, include programmable conversion sequencer, result registers, interface to analog circuits, interface to device peripheral bus, and interface to other on-chip modules.

***TMS320x280x Boot ROM Reference Guide*** (literature number SPRU722) 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.

***TMS320x208x Enhanced Capture (eCAP) Module Reference Guide*** (literature number SPRU807) describes the enhanced Capture Module. It includes the module description and registers.

***TMS320x280x Enhanced Quadrature Encoder Pulse (eQEP) Reference Guide*** (literature number SPRU790) 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.

***TMS320x280x System Control and Interrupts Reference Guide*** (literature number SPRU712) describes the various interrupts and system control features of the 280x digital signal processors (DSPs).

***TMS320x281x, 280x Enhanced Controller Area Network (eCAN) Reference Guide*** (literature number SPRU074) describes the eCAN that uses established protocol to communicate serially with other controllers in electrically noisy environments. With 32 fully configurable mailboxes and time-stamping feature, the eCAN module provides a versatile and robust serial communication interface. The eCAN module implemented in the C28x DSP is compatible with the CAN 2.0B standard (active).

***TMS320x281x, 280x Peripheral Reference Guide*** (literature number SPRU566) describes the peripheral reference guides of the 28x digital signal processors (DSPs).

***TMS320x281x, 280x Serial Communication Interface (SCI) Reference Guide*** (literature number SPRU051) describes the SCI that 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.

***TMS320x281x, 280x Serial Peripheral Interface (SPI) Reference Guide*** (literature number SPRU059) 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. The SPI is used for communications between the DSP controller and external peripherals or another controller.

***TMS320x280x System Control and Interrupts Reference Guide*** (literature number SPRU712) describes the various interrupts and system control features of the 280x digital signal processors (DSPs).

***The TMS320C28x Instruction Set Simulator Technical Overview*** (literature number SPRU608) describes the simulator, available within the Code Composer Studio for TMS320C2000 IDE, that simulates the instruction set of the C28x core.

***TMS320C28x DSP/BIOS Application Programming Interface (API) Reference Guide*** (literature number SPRU625) describes development using DSP/BIOS.

***3.3 V DSP for Digital Motor Control Application Report*** (literature number SPRA550). New generations of motor control digital signal processors (DSPs) lower their supply voltages from 5 V to 3.3 V to offer higher performance at lower cost. Replacing traditional 5-V digital control circuitry by 3.3-V designs introduce no additional system cost and no significant complication in interfacing with TTL and CMOS compatible components, as well as with mixed voltage ICs such as power transistor gate drivers. Just like 5-V based designs, good engineering practice should be exercised to minimize noise and EMI effects by proper component layout and PCB design when 3.3-V DSP, ADC, and digital circuitry are used in a mixed signal environment, with high and low voltage analog and switching signals, such as a motor control system. In addition, software techniques such as Random PWM method can be used by special features of the Texas Instruments (TI) TMS320x24xx DSP controllers to significantly reduce noise effects caused by EMI radiation.

This application report reviews designs of 3.3-V DSP versus 5-V DSP for low HP motor control applications.The application report first describes a scenario of a 3.3-V-only motor controller indicating that for most applications, no significant issue of interfacing between 3.3 V and 5 V exists. Cost-effective 3.3-V − 5-V interfacing techniques are then discussed for the situations where such interfacing is needed. On-chip 3.3-V ADC versus 5-V ADC is also discussed. Sensitivity and noise effects in 3.3-V and 5-V ADC conversions are addressed. Guidelines for component layout and printed circuit board (PCB) design that can reduce system's noise and EMI effects are summarized in the last section.

## *Trademarks*

Code Composer Studio and C28x are trademarks of Texas Instruments.

# Contents

*This chapter provides information about applications of the EPWM module.*

*Provides the register layouts and bit descriptions.*

# Figures

# Tables

# Examples

Chapter 1

# 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, uninterruptable power supplies (UPS), and other forms of power conversion. The PWM peripheral performs a 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 family of processors. This includes all Flash-based, ROM-based, and RAM-based devices within the 280x family.

**Topic** **Page**

## 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, TI has opted for a PWM generation unit that is built up from smaller single channel modules with separate local 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.

## 1.2 Sub-module Overview

The ePWM module represents one complete PWM channel. One channel is defined as having two PWM outputs (described later in detail). Multiple identical ePWM modules are contained in a 280x system as shown in Figure 1−1; they are chained together via a 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) if required. The number of modules is device-dependent and based on target application needs. Modules can also operate stand-alone.

Each ePWM module (shown in Figure 2) supports the following features:

❑ Dedicated 16-bit time base with period/frequency control

❑ Two independent PWM outputs with single-edge operation, or
two independent PWM outputs with dual-edge symmetric operation, or
one independent PWM output with dual-edge asymmetric operation

❑ Asynchronous override (forcing) control of PWM signals via 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.

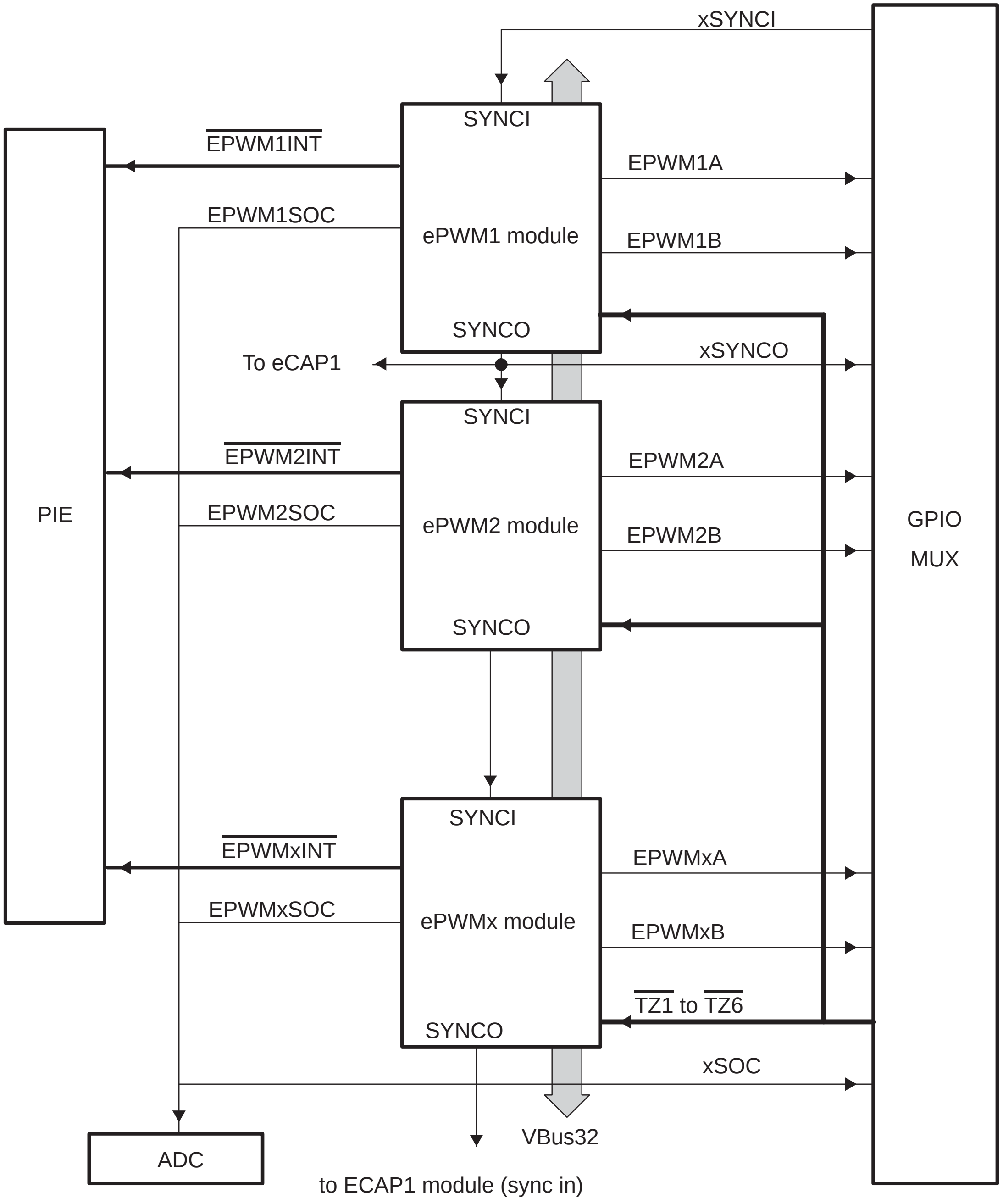
❑ All events can trigger both CPU interrupts and ADC start of conversion (SOC)

❑ 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 1−2. The signals are described in detail in subsequent sections.

*Figure 1−1. Multiple ePWM Modules in a 280x System*



Each ePWM module consists of seven submodules and is connected within a 280x system via the signals shown in Figure 1−2.

*Figure 1−2. Sub-Modules and Signal Connections for an ePWM Module*

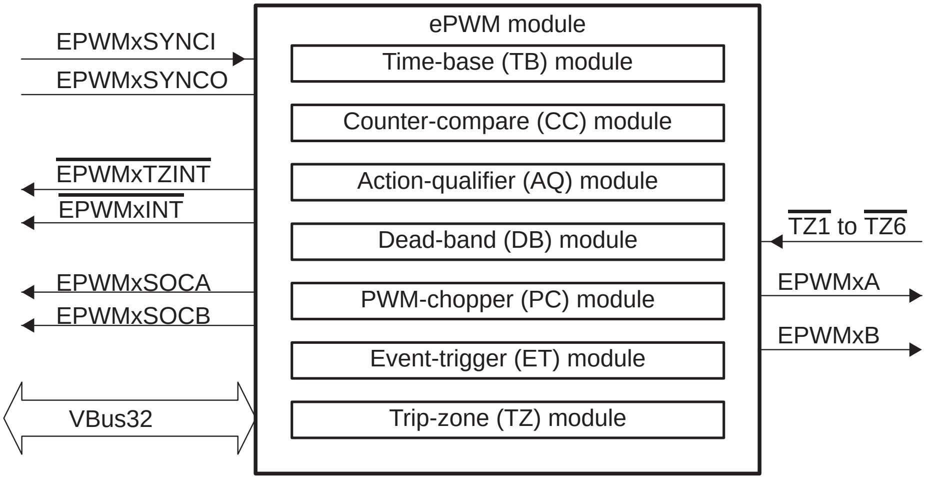

ePWM module
Time-base (TB) module
Counter-compare (CC) module
Action-qualifier (AQ) module
Dead-band (DB) module
PWM-chopper (PC) module
Event-trigger (ET) module
Trip-zone (TZ) module
EPWMxSYNCI
EPWMxSYNCO
$\overline{\text{EPWMxTZINT}}$
$\overline{\text{EPWMxINT}}$
EPWMxSOCA
EPWMxSOCB
VBus32
$\overline{\text{TZ1}}$ to $\overline{\text{TZ6}}$
EPWMxA
EPWMxB

Figure 1−3 shows more internal details of a single ePWM module. For all ePWM modules, signals EPWMxA, EPWMxB, and TZ1 to TZ6 are made available external to the 280x device via the GPIO peripheral. In general, EPWMSYNCI and EPWMSYNCO are brought out to pins (via GPIO) only for ePWM module #1. Also ePWM1SYNCO is connected to SYNCI of eCAP1. See Figure 1−1 and the device-specific data sheet.

*Figure 1−3. ePWM Sub-Modules Showing Critical Internal Signal Interconnects*

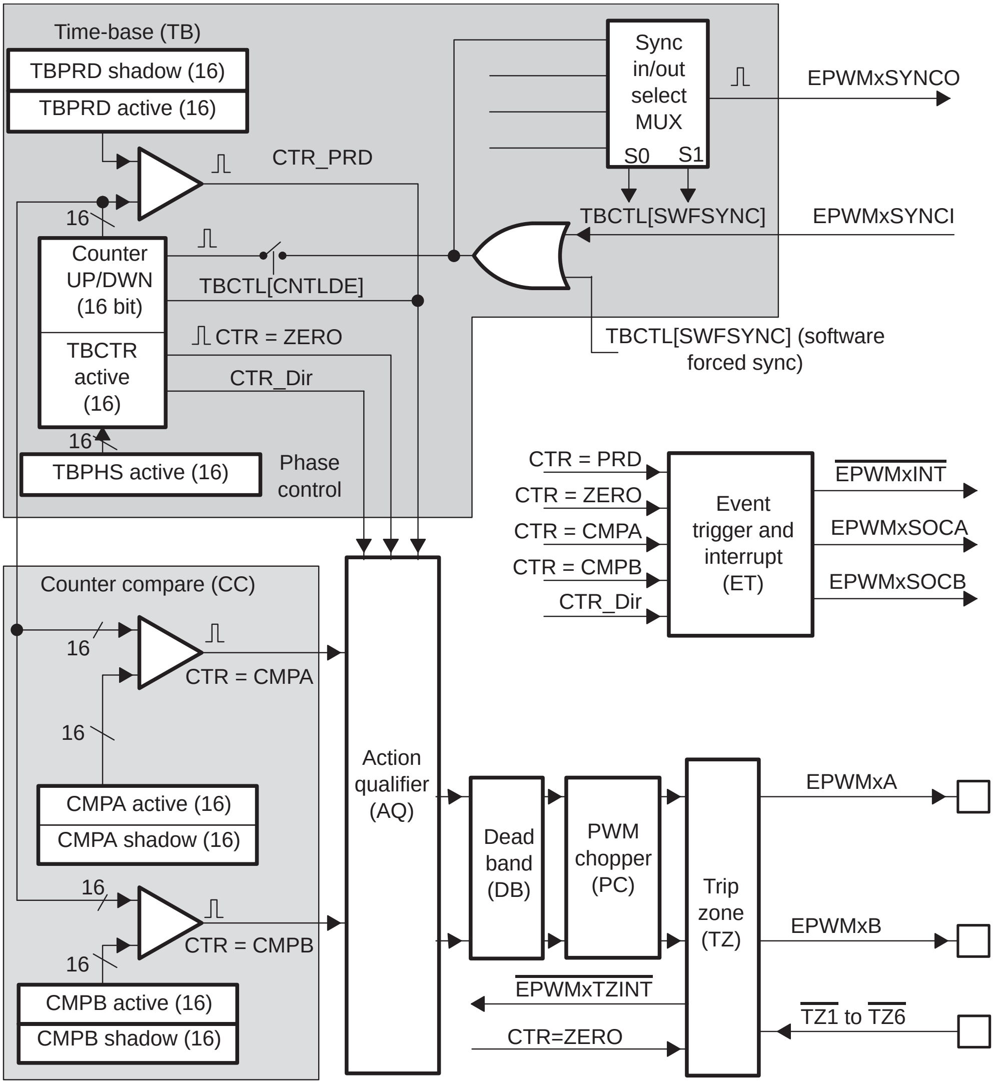


Figure 1−3 also shows the key internal sub-module interconnects. Each sub-module is described in detail in its respective section.

## 1.3 Register Mapping

The complete ePWM module control and status register set is grouped by sub-module as shown in Table 1−1.

*Table 1−1. ePWM Module Control and Status Register Set Grouped by Sub-Module*

| Name | Offset | Size (x16) / # Shadow | Description |
|---|---|---|---|
| | | | **Time-base Module Registers** |
| TBCTL | 0x0000 | 1 / 0 | Time-base Control Register |
| TBSTS | 0x0001 | 1 / 0 | Time-base Status Register |
| Reserved | 0x0002 | | |
| TBPHS | 0x0003 | 1 / 0 | Time-base Phase Register |
| TBCTR | 0x0004 | 1 / 0 | Time-base Counter Register |
| TBPRD | 0x0005 | 1 / **1** | Time-base Period Register Set |
| Reserved | 0x0006 | 1 / 0 | |
| | | | **Compare Module Registers** |
| CMPCTL | 0x0007 | 1 / 0 | Counter Compare Control Register |
| Reserved | 0x0008 | | |
| CMPA | 0x0009 | 1 / **1** | Counter Compare A Register Set |
| CMPB | 0x000A | 1 / **1** | Counter Compare B Register Set |
| | | | **Action-qualifier Module Registers** |
| AQCTLA | 0x000B | 1 / 0 | Action-qualifier Control Register For Output A |
| AQCTLB | 0x000C | 1 / 0 | Action-qualifier Control Register For Output B |
| AQSFRC | 0x000D | 1 / 0 | Action-qualifier Software Force Register |
| AQCSFRC | 0x000E | 1 / **1** | Action-qualifier Continuous S/W Force Register Set |

**Notes:** 1) These registers are EALLOW protected.

*Table 1−1. ePWM Module Control and Status Register Set Grouped by Sub-Module (Continued)*

| Name | Offset | Size (x16) / # Shadow | Description |
|---|---|---|---|
| | | | **Dead-Band Generator Module Registers** |
| DBCTL | 0x000F | 1 / 0 | Dead−Band Generator Control Register |
| DBRED | 0x0010 | 1 / 0 | Dead−Band Generator Rising Edge Delay Count Register |
| DBFED | 0x0011 | 1 / 0 | Dead−Band Generator Falling Edge Delay Count Register |
| | | | **PWM-chopper Module Registers** |
| PCCTL | 0x001E | 1 / 0 | PWM-chopper Control Register |
| Reserved | 0x001F | | |
| | | | **Trip-zone Module Registers** |
| TZSEL | 0x0012 | 1 / 0 | Trip-zone Select Register (1) |
| TZDCSEL | 0x0013 | 1 / 0 | Trip-zone Digital Comparator Select Register (1) |
| TZCTL | 0x0014 | 1 / 0 | Trip-zone Control Register (1) |
| TZEINT | 0x0015 | 1 / 0 | Trip-zone Enable Interrupt Register (1) |
| TZFLG | 0x0016 | 1 / 0 | Trip-zone Flag Register (1) |
| TZCLR | 0x0017 | 1 / 0 | Trip-zone Clear Register (1) |
| TZFRC | 0x0018 | 1 / 0 | Trip-zone Force Register (1) |
| | | | **Event-trigger Module Registers** |
| ETSEL | 0x0019 | 1 / 0 | Event-trigger Selection Register |
| ETPS | 0x001A | 1 / 0 | Event-trigger Pre−Scale Register |
| ETFLG | 0x001B | 1 / 0 | Event-trigger Flag Register |
| ETCLR | 0x001C | 1 / 0 | Event-trigger Clear Register |
| ETFRC | 0x001D | 1 / 0 | Event-trigger Force Register |
| | | | **Reserved** |
| Reserved | 0x0030<br>to<br>0x003F | 16 / 0 | |

**Notes:** 1) These registers are EALLOW protected.

Chapter 2

# Sub-Modules

Seven sub-modules comprise the ePWM peripheral. Each one performs specific tasks that can be programmed or modified.

**Topic** **Page**

## 2.1 Overview

Table 2−1 lists the seven key sub-modules 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 refer to the Counter Compare (CC) Module section 2.3 (page 2-15) for relevant details.

*Table 2−1. Sub-Modules and Their Configuration Parameters*

| Sub-Module | Configuration Parameter or Option |
|---|---|
| Time-base (TB) Module | PWM frequency (or period) |
| | Count mode (up or down for asymmetric, or up and down for symmetric) |
| | Time-base phase relative to another ePWM module |
| | Time-base synchronization options in hardware |
| | Time-base synchronization forcing via software |
| Counter-compare (CC) Module | PWM duty cycle for either EPWMxA or EPWMxB outputs |
| | Time at which switching events occur on either EPWMxA or EPWMxB outputs |
| | Time at which interrupt or ADC SOC events occur |
| Action-qualifier (AQ) Module | The type (high, low, etc.) of switching action when an event occurs from the TB or CC modules |
| | Force a switching action via software control |
| | Full software control and flexibility over dead-band |
| Dead-band (DB) Module | Control of traditional complementary dead-band relationship between upper and lower switches |
| | Rising edge delay value |
| | Falling edge delay value |
| | Bypass this sub-module entirely |
| PWM-chopper (PC) Module | Chopping (carrier) frequency |
| | Pulse width of the first pulse in the chopped pulse train |
| | Duty cycle of the second and subsequent pulses |
| | Bypass this sub-module entirely |

*Table 2−1. (Continued) Sub-Modules and Their Configuration Parameters*

| Sub-Module | Configuration Parameter or Option |
|---|---|
| Trip-zone (TZ) Module | Tripping action taken when a fault occurs (Hi, Low, Hi-Z,…etc) |
| | Trip modes: Cycle-by-cycle, one-shot, or both |
| | Allocation or mapping of trip-zone fault pins to an ePWM module |
| | Bypass this sub-module entirely |
| Event-trigger (ET) Module | Mapping (enable) events that will trigger a CPU interrupt |
| | Mapping (enable) events that will trigger an ADC SOC |
| | Rate (prescale) at which events cause triggers, i.e., every occurrence or every second or third occurrence, and so on |
| | 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 all make use of the constant definitions shown in Example 2−1 as well as the F280x peripheral header file structures.

*Example 2−1. Constant Definitions Used in the Code Examples*

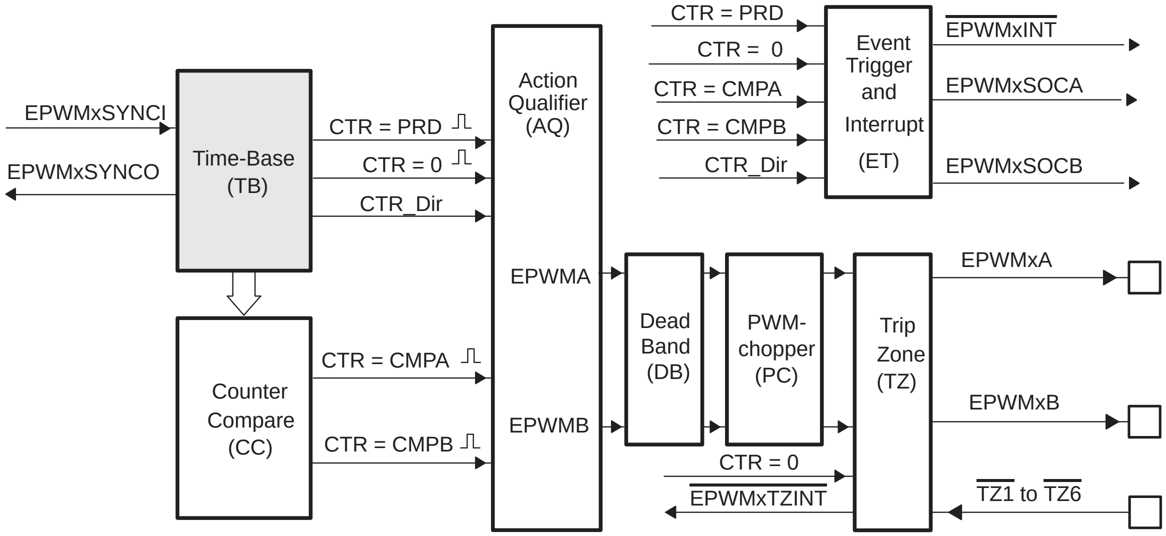
```
// TBCTL (Time-Base Control)
// = = = = = = = = = = = = = = = = = = = = = = = = = = =  =
// CTRMODE bits
#define      TB_COUNT_UP         0x0
#define      TB_COUNT_DOWN       0x1
#define      TB_COUNT_UPDOWN     0x2
#define      TB_FREEZE           0x3
// PHSEN bit
#define      TB_DISABLE          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
#define       DB_ACTV_LO           0x3
// CHPCTL (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) Module

The time-base (TB) module provides all event timing for the ePWM. All event timing is derived from this block. Additionally, the built-in synchronization logic allows multiple time-bases to work together as a single system. Figure 2−1 illustrates the TB module's place in the ePWM.

*Figure 2−1. TB Module Block Diagram*



### 2.2.1 Purpose of the TB Module

You can use the TB module for the following tasks:

- ❑ Controlling the frequency or period of events
- ❑ Managing synchronization with up-stream and down-stream modules
- ❑ Maintaining phase relationship (value) with other ePWM modules
- ❑ Setting the count mode (up, down, or up-down)
- ❑ Generating events:
  - ■ CTR = PRD
  - ■ CTR = Zero
- ❑ Prescaling the incoming CPU system clock (SYSCLKOUT) to allow the ePWM module to run at lower clock speeds if required.

### 2.2.2 Controlling and Monitoring the TB Module

Table 2−2 shows how to control and monitor the TB module operation via the registers.

*Table 2−2. TB Module Registers*

| mnemonic | Address offset | Shadowed | Description |
|---|---|---|---|
| TBCTL | 0x0000 | No | TB Control Register |
| TBSTS | 0x0001 | No | TB Status Register |
| reserved | | | |
| TBPHS | 0x0003 | No | TB Phase Register |
| TBCTR | 0x0004 | No | TB Counter Register |
| TBPRD | 0x0005 | Yes | TB Period |

The block diagram in Figure 2−2 shows the critical signals and registers of the TB module.

*Figure 2−2. TB Module Signals and Registers*

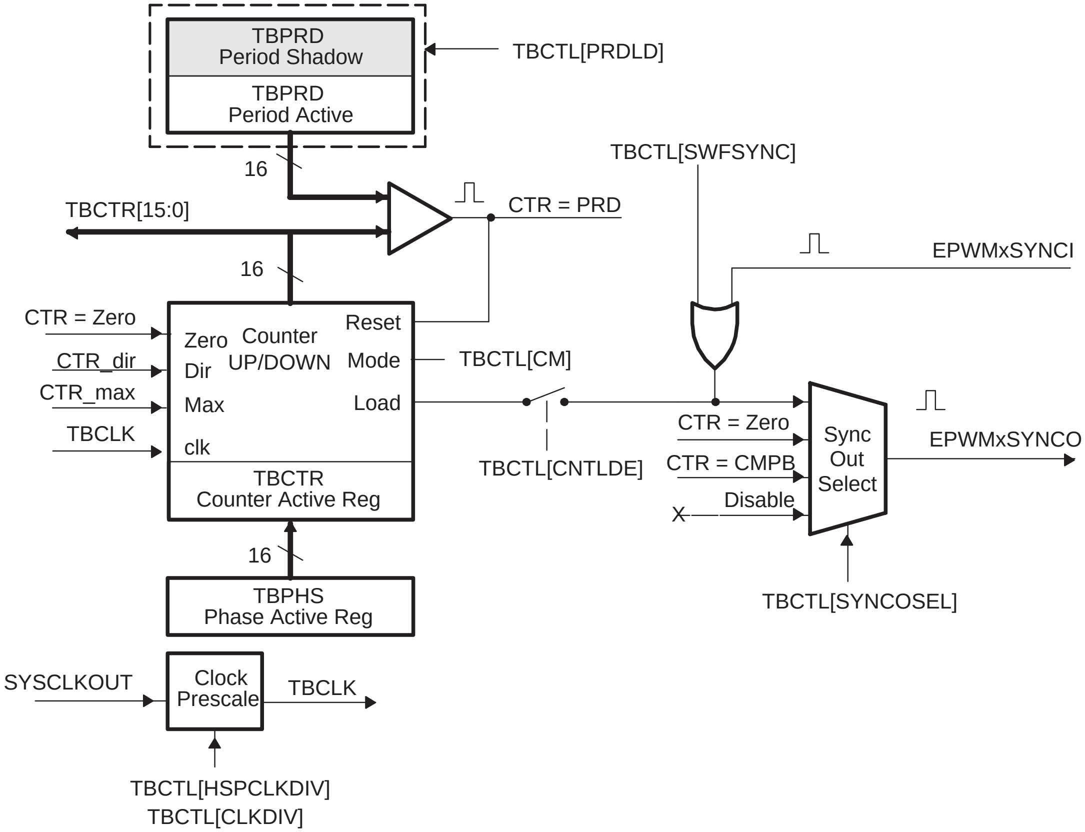


Table 2−3 provides descriptions of the key signals associated with the TB Module.

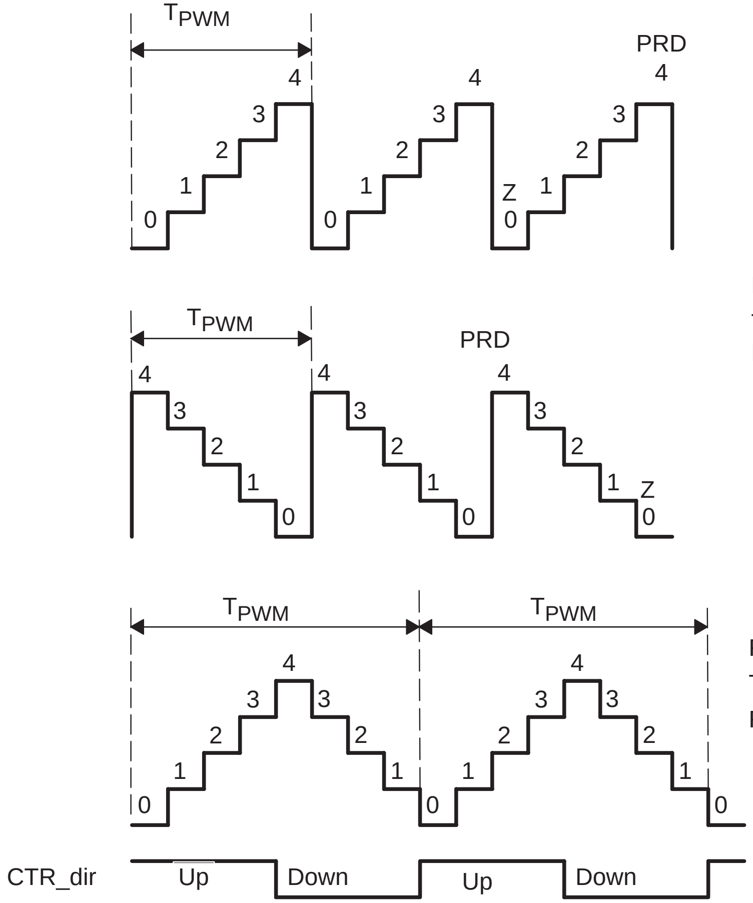
*Table 2−3. Descriptions of Key Signals for a TB Module*

| Signal | Description |
|---|---|
| EPWMxSYNCI | Input pulse that is used to synchronize its own counter to an up-stream ePWM module. |
| EPWMxSYNCO | Output pulse that is used to synchronize the counter of a down-stream ePWM module. |
| CTR = PRD | Generated event whenever counter value is equal to the active period register value, i.e., (TBCTR[15:0] = = TBPRD[15:0]) |
| CTR_dir | Signal indicating current status of counter direction. High = up counting, Low = down counting. |
| CTR = ZERO | Generated event whenever counter value is zero (TBCTR[15:0] = = 0x0000) |
| CTR_max | Generated event whenever counter value reaches max (TBCTR[15:0] = = 0xFFFF) used only as a status bit |
| TBCLK | Prescaled (divided) version of the system clock. Goes to all other sub-modules in the ePWM. |
| CTR = CMPB | Event (TBCTR[15:0] = = CMPB[15:0] ) generated by the CC module and used by the SyncOut logic |

### 2.2.3 Calculating Period and Frequency

The frequency of generated events (i.e., within a PWM cycle) is controlled by the value written in the TBPRD register. Figure 2−3 shows the period and frequency relationships for up, down, and up and down count modes with a simple case of TBPRD = 4. The minimum time step increment is defined by TBCLK, which is a prescaled version of the system (CPU) clock SYSCLK.

*Figure 2−3. TB Module Operational Highlights*



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 Period Register Loading From Shadow

The period register TBPRD has a shadow (buffering) loading option. If shadow mode is selected (TBCTL[PRDLD] = 0), then when event CTR = 0 occurs the contents of shadow register are transferred to the active register (i.e., TBPRD (Shadow) ← TBPRD (Active)).

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

### 2.2.3.2 Counter Register Load From Phase Register Option

This feature enables the ePWM module to be automatically synchronized to the time base of another ePWM module. This allows adding lead or lag phase control to the waveforms generated by different ePWM modules to synchronize them.

By setting the TBCTL[PHSEN] bit, the counter register (TBCNT) is configured to be reloaded from the phase register (TBPHS) when either of the folowing two conditions occurs:

❑ EPWMxSYNCI (sync input pulse): When a sync pulse is detected, the value in the phase register is loaded into the counter register (TBPHS → TBCNT). This operation occurs on the next valid TBCLK edge.

❑ Software forced sync pulse: This S/W forcing option, invoked via TBCTL[SWFSYNC] control bit, allows you to override or generate a pseudo sync-in pulse. This S/W forcing function is ORed with the EPWMxSYNCI signal, and has the same effect as the sync-in pulse.

The ePWM module can be configured to ignore (disable) the sync-In action locally by clearing the TBCTL [PHSEN] bit, but still allow the sync pulse to flow-through and be used to synchronize down-stream modules. In this way, you can set up a master time-base (for example, EPWM1) and downstream modules may elect to run in synchronization with the master only. See the Application to Power Topologies chapter 3 (page 3-1) for more details on synchronization strategies.

In up-down mode, the TBPHS bit in the TBCTL register configures the direction of the counter on a sync event regardless of the current direction of the counter. This bit has no effect in up or down modes. See Figure 2−4 through Figure 2−7 for examples.

## 2.2.4 Count Mode Timing Wave Forms

The TB module can operate in four distinct modes:

❑ Count up—commonly referred to as asymmetrical
❑ Count Down—commonly referred to as asymmetrical
❑ Count Up and Down—commonly referred to as symmetrical
❑ Frozen—counter is held constant at current value

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

*Figure 2−4. TB Module Up-Count Mode Waveforms*

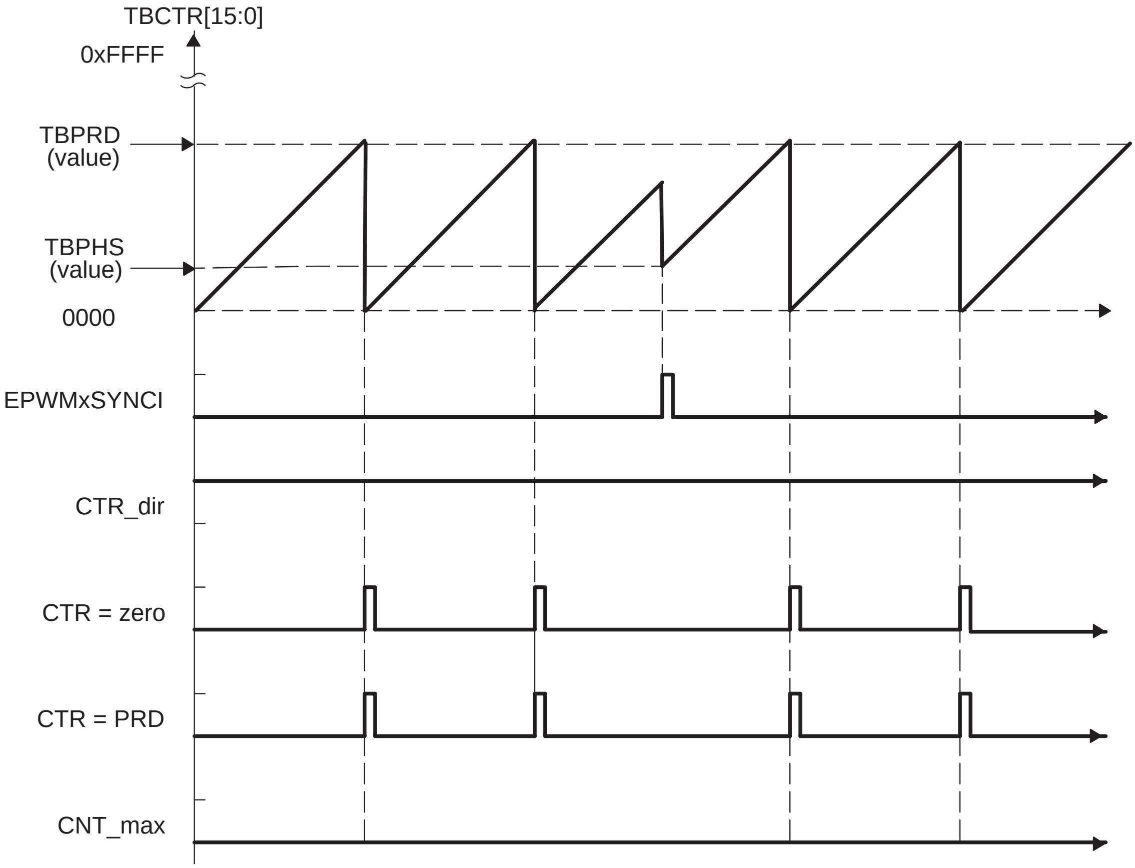


*Figure 2−5. TB Module Down-Count Mode Waveforms*

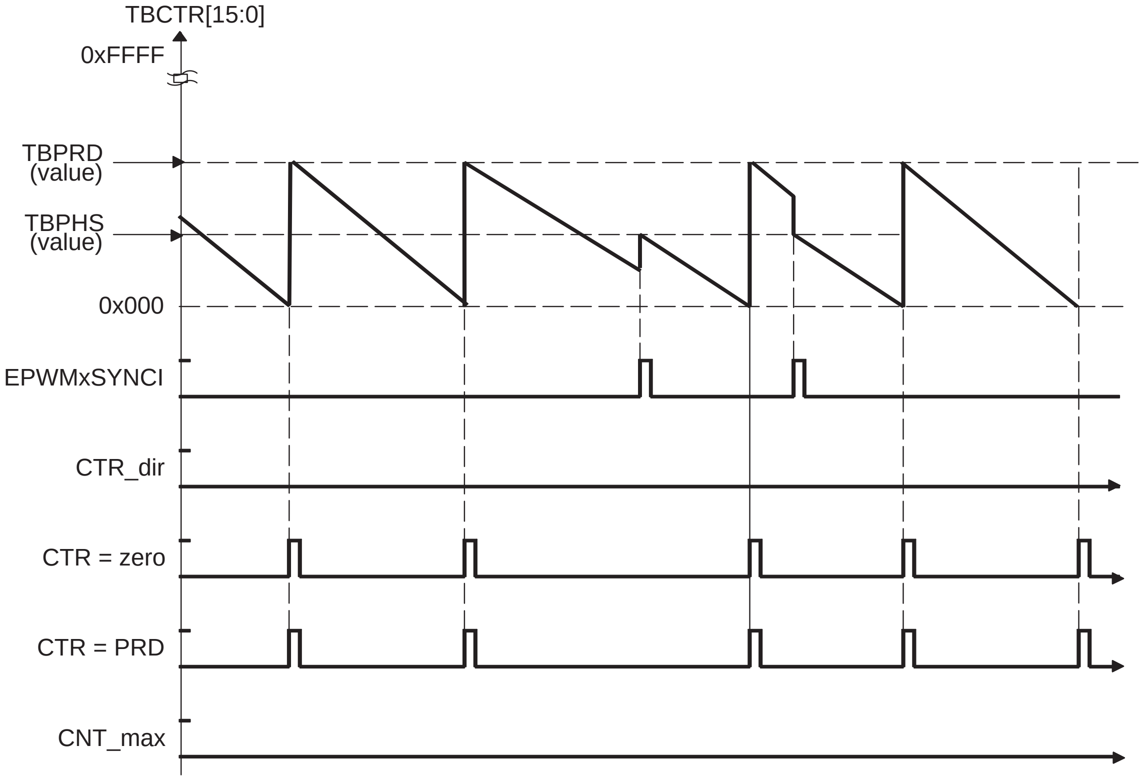


*Figure 2−6. TB Module Up and Down-Count Waveforms, TBCTL[TBPHS = 0] Count Down On Sync Event*

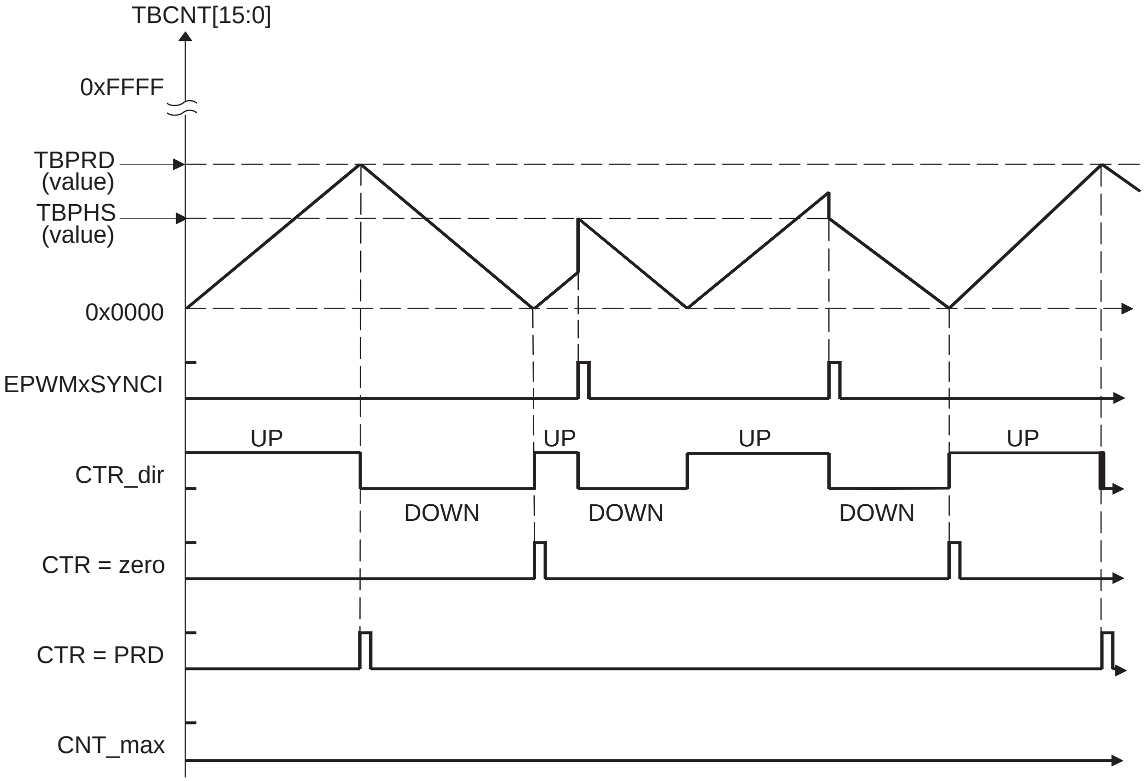
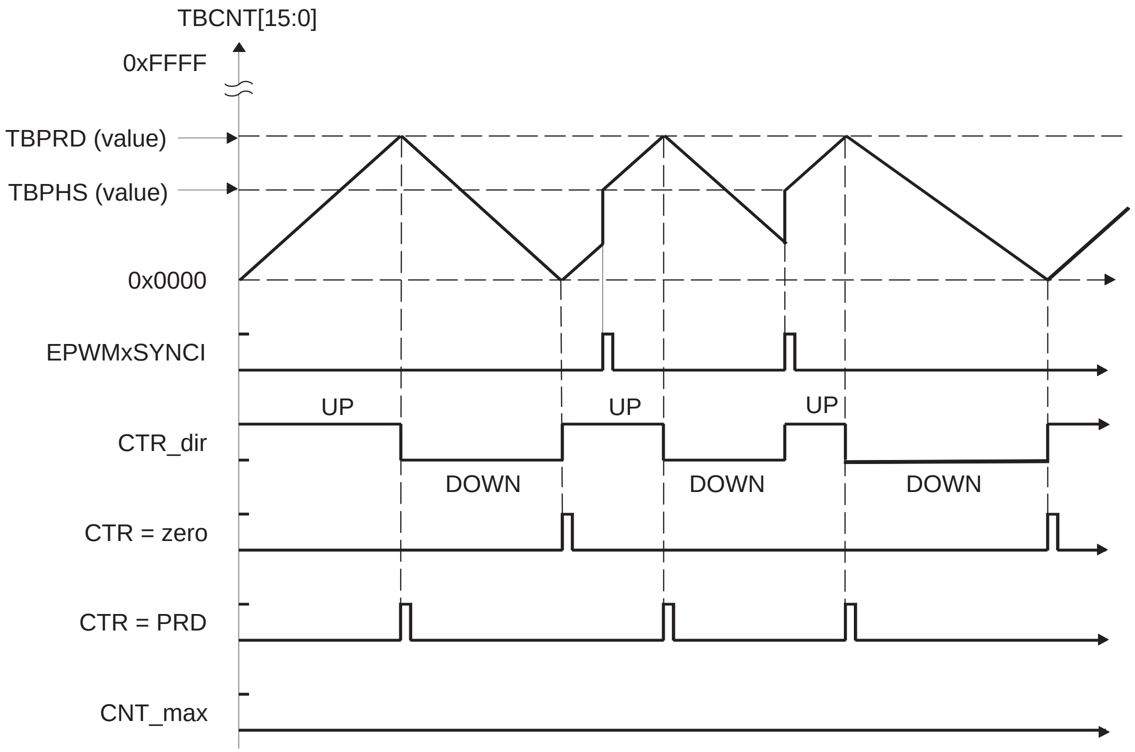


*Figure 2−7. TB Module Up-Down Count Waveforms, TBCTL[TBPHS = 1] Count Up On Sync Event*



## 2.3 Counter-compare (CC) Module

Figure 2−8 illustrates the CC sub-module within the ePWM.

*Figure 2−8. CC Module*

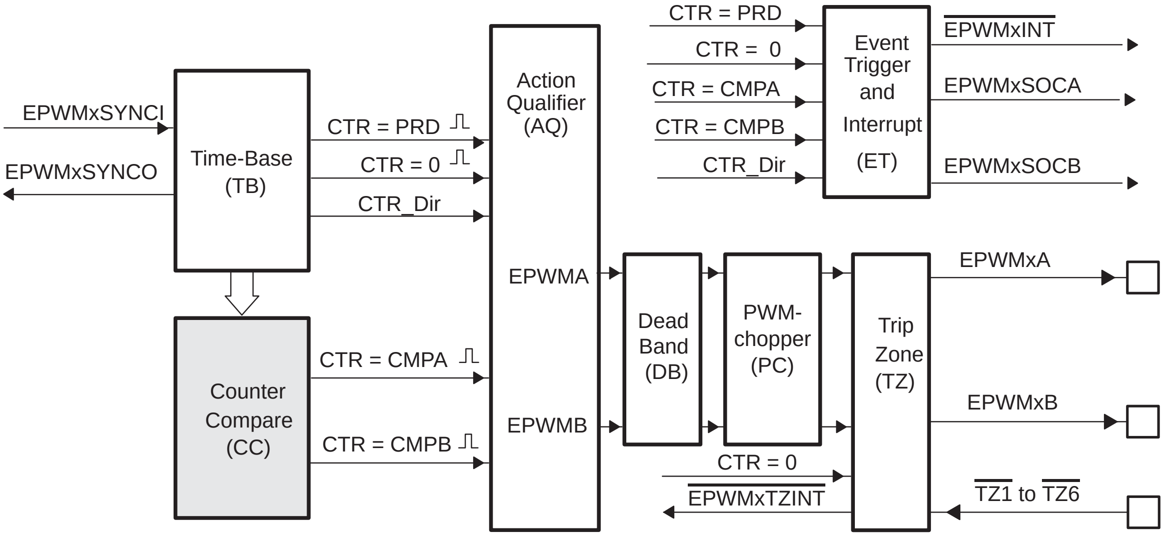


Figure 2−9 shows the basic structure of the CC module.

### 2.3.1 Purpose of the CC Module

The CC takes as input the TBCTR[15:0] value via a 16-bit bus. This value is continuously compared to the 16-bit compare registers A and B and when equal, generates appropriate compare events.

The key purposes for the CC module are:

- ❑ Generating events based on programmable time-stamps using the CMPA and CMPB registers
  - ■ CTR = CMPA
  - ■ CTR = CMPB
- ❑ Controlling the PWM duty cycle, if the action-qualifier (AQ) module is configured appropriately
- ❑ Shadowing (buffering) new compare values to prevent corruption or glitches during the active PWM cycle

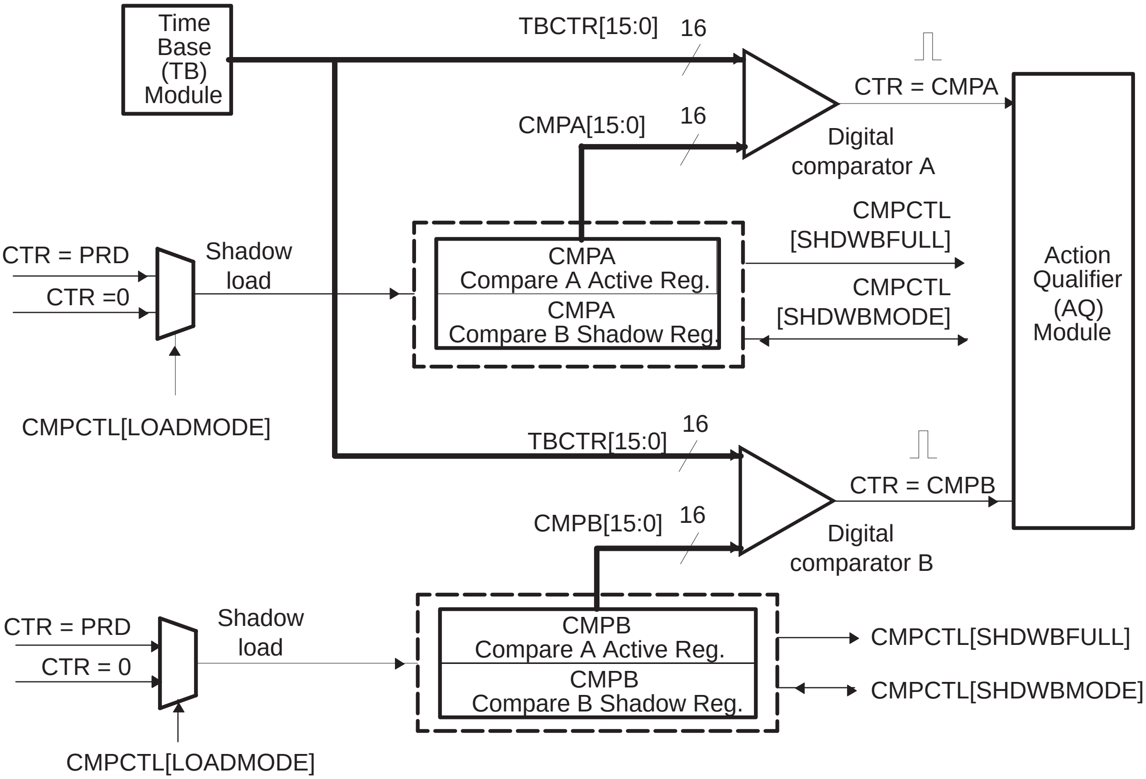
### 2.3.2 Controlling and Monitoring the CC Module

The CC module operation is controlled and monitored by the following registers:

*Table 2−4. CC Module Registers*

| mnemonic | Address offset | Shadowed | Description |
|---|---|---|---|
| CMPCTL | 0x0007 | No | CC Control Register. |
| reserved | | | |
| CMPA | 0x0009 | Yes | Counter-compare A |
| CMPB | 0x000A | Yes | Counter-compare B |

*Figure 2−9. Detailed View of the CC Module*



The key signals associated with the CC module are given in Table 2−5.

*Table 2−5. CC Module Key Signals*

| Signal | Description |
|---|---|
| CTR = CMPA | Generated event whenever counter value is equal to the active compare A register value, i.e., (TBCTR[15:0] = CMPA[15:0]) |
| CTR = CMPB | Generated event whenever counter value is equal to the active compare B register value, i.e., (TBCTR[15:0] = CMPB[15:0]) |
| CTR = PRD | Used as load strobe option (shadow register → active register) |
| CTR = ZERO | Used as load strobe option (shadow register → active register) |
| TBCTR[15:0] | 16 bit TB counter bus used for comparison to CMPA and CMPB active registers |

### 2.3.3 Operational Highlights for the CC Module

The CC module is responsible for generating two independent compare events:

1) CTR = CMPA
2) CTR = CMPB

These events feed into the AQ module, where they are qualified and converted into actions if enabled (see “Action-qualifier (AQ) Module ” section for more details).

Compare events are generated in the three possible count modes according to the following conditions:

If CMPA and CMPB are the values written into Compare A and B registers respectively:

CTR = CMPA events occur if $0000 \leq CMPA \leq TBPRD$
CTR = CMPB events occur if $0000 \leq CMPB \leq TBPRD$

For count up or count down, each event occurs once per cycle.

For count up and down each event occurs twice per cycle if $0000 < CMPx < TBPRD$.

For count up and down, each event occurs once per cycle if CMPx = 0000 or CMPx = TBPRD.

The compare registers CMPA and CMPB have a shadow (buffering) loading option.

If shadow load mode is selected, CMPCTL[SHDWxMODE] = 0, the contents of the shadow register are transferred to the active register, i.e., CMPx (Shadow) → CMPx (Active) on the following events/conditions:

- ❑ CTR = PRD
- ❑ CTR = Zero
- ❑ CTR = PRD or CTR = Zero

The choices above  are selected by CMPCTL[LOADxMODE] bits.

If immediate load mode is selected, CMPCTL[SHDWxMODE] = 1, then a CPU write to CMPx registers will go directly to the active register.

In shadow mode, a CPU read or write to the CMPx addresses always accesses the shadow registers.

In Immediate mode, a CPU read or write to the CMPx addresses always accesses the active registers.

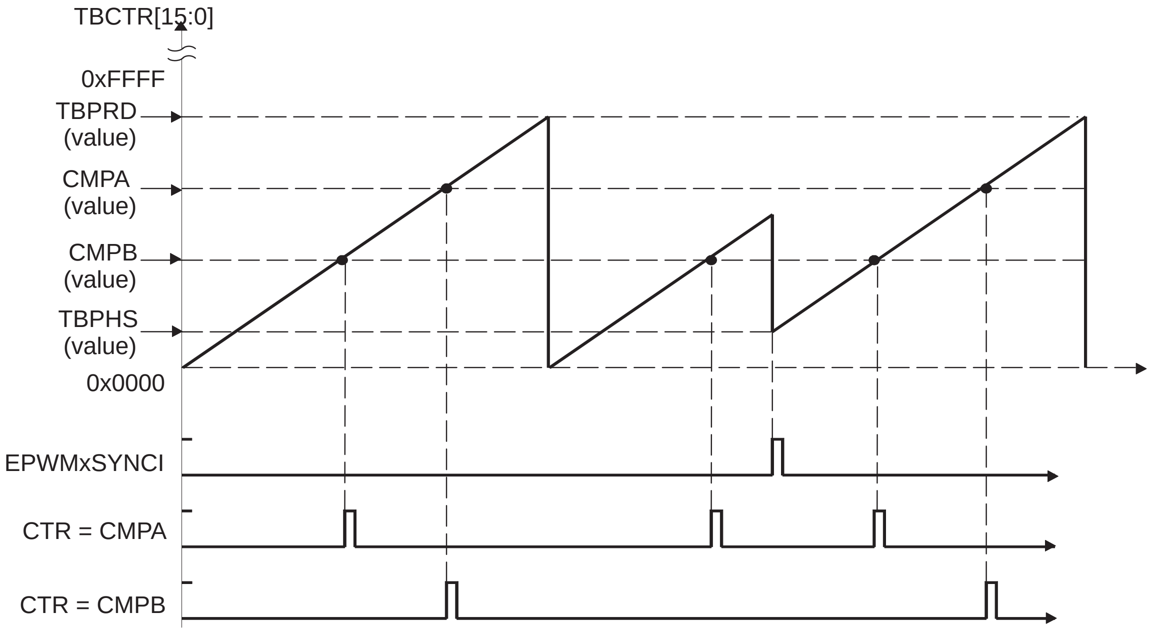
### 2.3.4 Count Mode Timing Waveforms

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

- ❑ Count up — used to generate asymmetrical PWM
- ❑ Count down — used to generate asymmetrical PWM
- ❑ Count up-down — used to generate symmetrical PWM

To best illustrate the operation of the first three modes, the timing diagrams in Figure 2−10 through Figure 2−13 show when events are generated and how the EPWMxSYNCI signal interacts.

*Figure 2−10. CC Event Waveforms in Up-Count Mode*



**Note:** An EPWMxSYNCI external sync event can cause a discontinuity in the TBCTR count sequence, which can lead to a compare event being skipped. This skipping is normal operation and must be taken into account.

*Figure 2−11.CC Event Waveforms in Down-Count Mode*

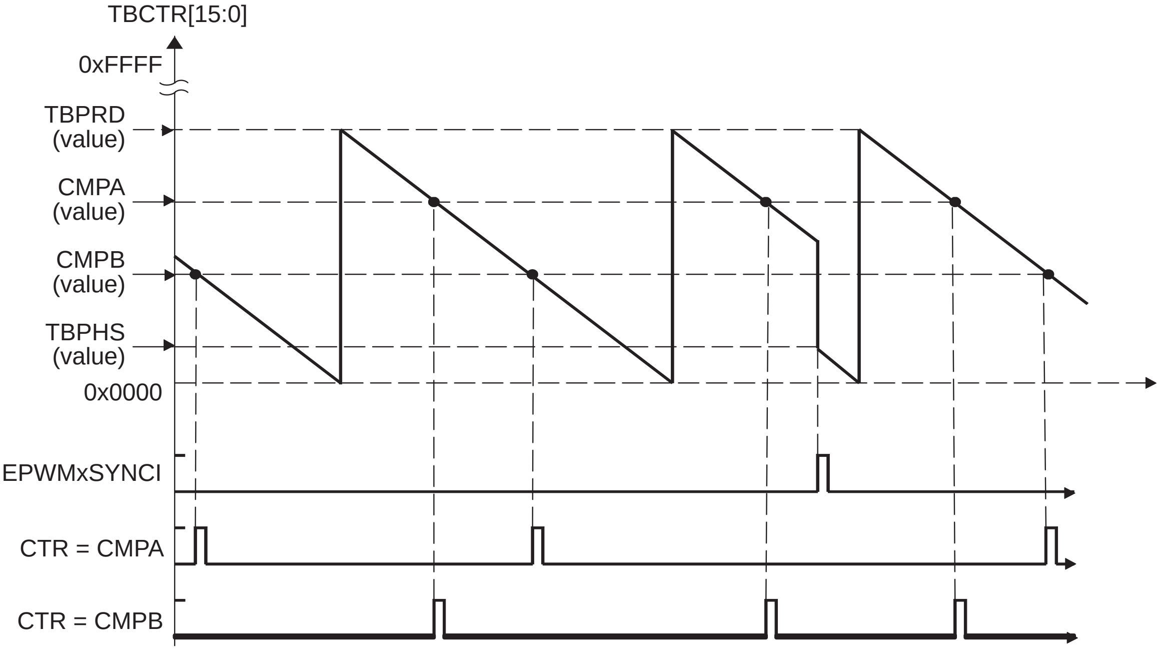


*Figure 2−12. CC Event Waveforms In Up/Down-Count Mode, TBCTL[TBPHS = 0] Count Down On Sync Event*

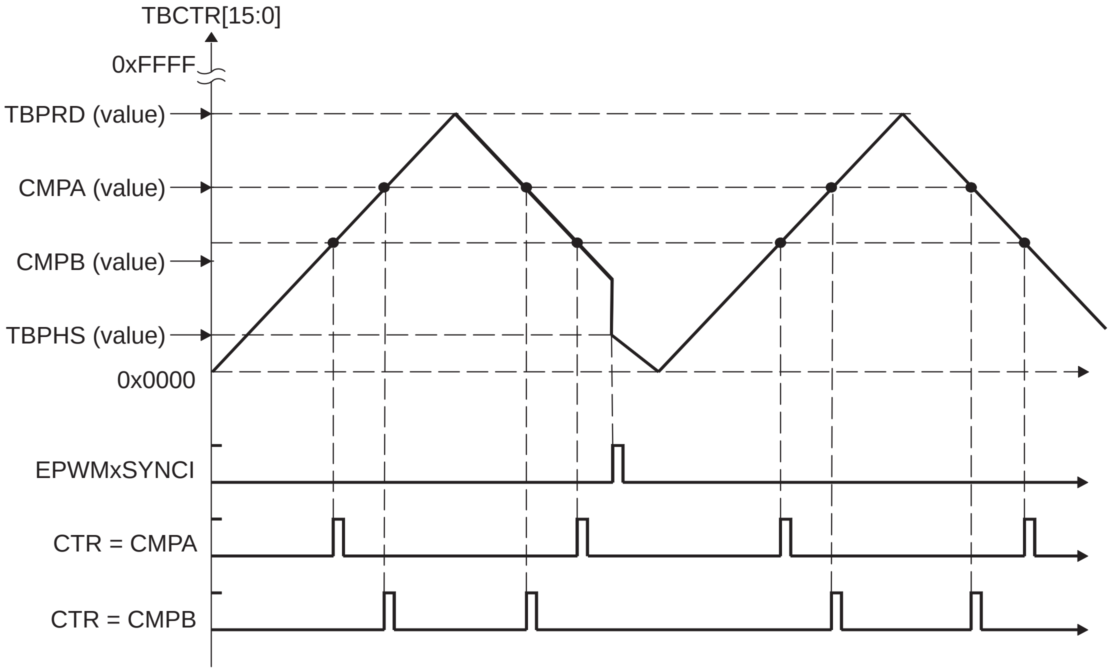
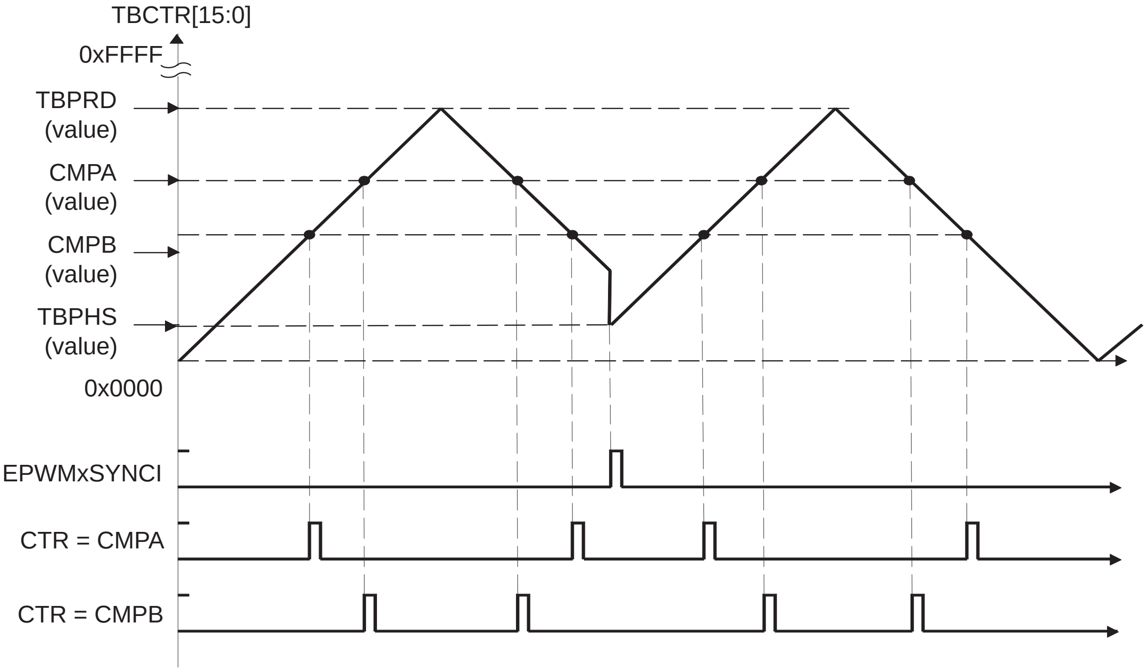


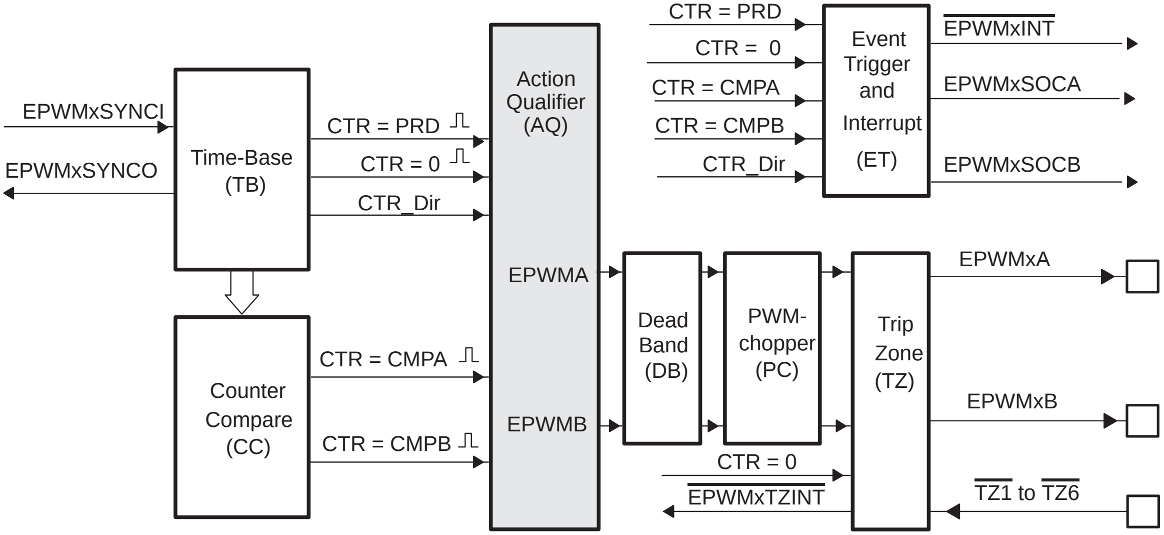
*Figure 2−13. CC Event Waveforms In Up/Down-Count Mode, TBCTL[TBPHS = 1] Count Up On Sync Event*



## 2.4 Action-qualifier (AQ) Module

Figure 2−14 shows the action-qualifier (AQ) module (see shaded block) in the ePWM system.

*Figure 2−14. AQ Module*



The AQ module 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 AQ Module

The AQ module is responsible for the following:

- Qualifying and generating actions (set, clear, toggle) based on the following input events:
  - CTR = PRD
  - CTR = Zero
  - CTR = CMPA
  - CTR = CMPB
- Managing priority when these events occur concurrently
- Using the time-base direction information (CTR_dir) as a further qualifier, to allow for independent control of up events and down events

### 2.4.2 AQ Module Control and Status Register Definitions

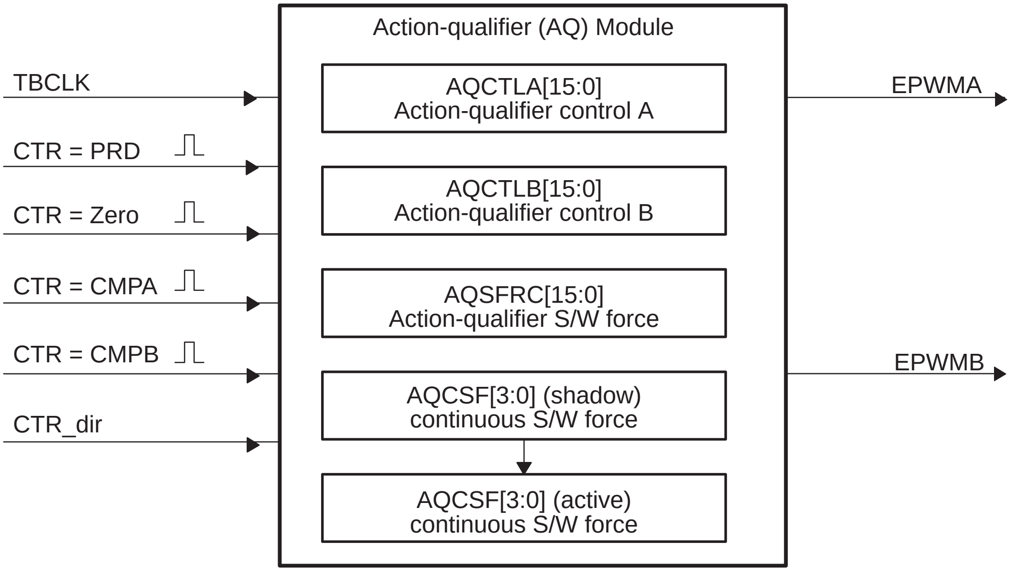
The AQ module operation is controlled and monitored via the registers in Table 2−6.

*Table 2−6. AQ Module Registers*

| mnemonic | Address offset | Shadowed | Description |
|---|---|---|---|
| AQCTLA | 0x000B | No | Action-qualifier Control Register For Output A |
| AQCTLB | 0x000C | No | Action-qualifier Control Register For Output B |
| AQSFRC | 0x000D | No | Action-qualifier Software Force Register |
| AQCSFRC | 0x000E | Yes | Action-qualifier Continuous S/W Force |

The AQ module 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 Figure 2−15.

*Figure 2−15. AQ Module Inputs and Outputs*



For convenience, the possible input events are summarized again in Table 2−7.

*Table 2−7. AQ Module Possible Input Events*

| Event Condition | Description |
|---|---|
| CTR = PRD | Counter equal to period value, i.e., CTR[0–15] = PRD[0–15] |
| CTR = Zero | Counter equal to zero value, i.e., CTR[0–15] = 0000h |
| CTR = CMPA | Counter equal to compareA value, i.e., CTR[0–15] = CMPA[0–15] |
| CTR = CMPB | Counter equal to compareA value, i.e., CTR[0–15] = CMPB[0–15] |
| S/W force | Asynchronous event initiated by software (CPU) via control register bits |

Although not an external event to the AQ module, the software (S/W) force is a useful asynchronous event that can be initiated. This control is handled by registers AQSFRC and AQCSF. The AQ module controls how the two outputs A and B behave when a particular event occurs. The event inputs to the AQ module are further qualified by the counter direction status, i.e., up or down. This allows for independent action on outputs A and B on both the count-up and count-down phases. The possible actions imposed on outputs A and B are given in Table 2−8.

*Table 2−8. Possible Actions Imposed on Outputs A and B*

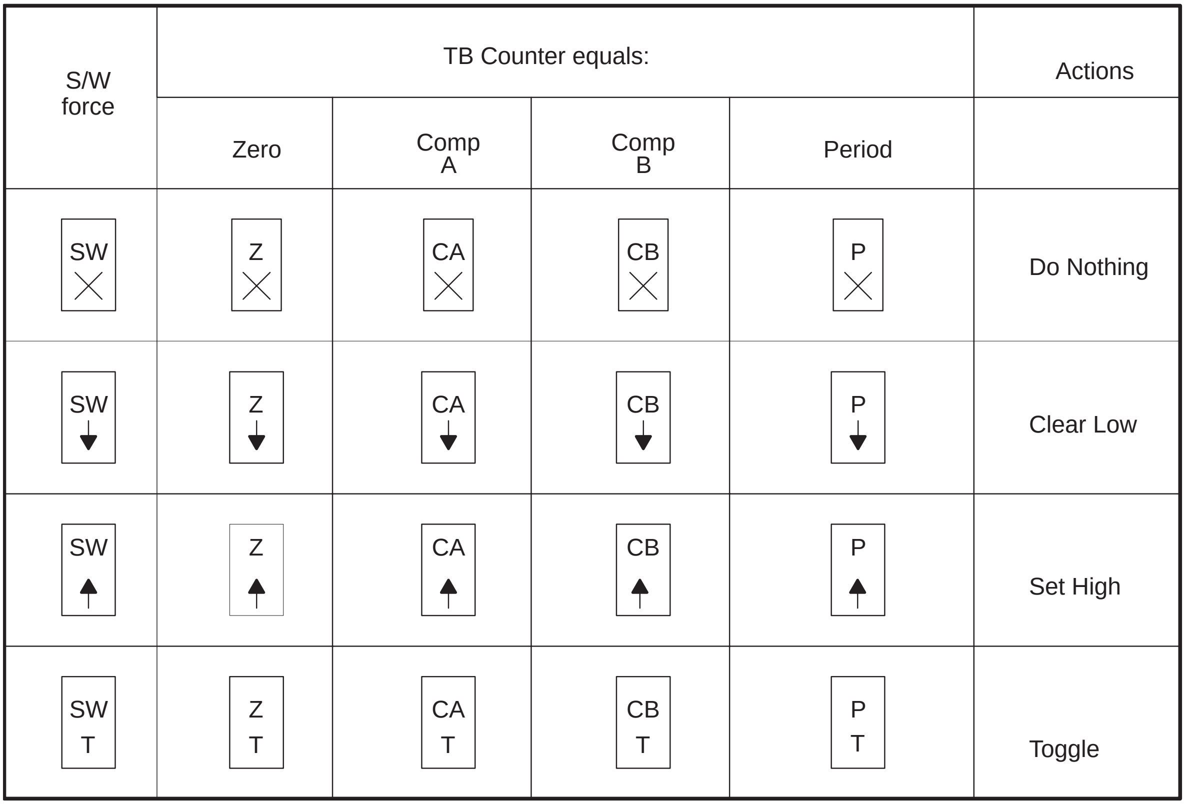
| Action Type | Description |
|---|---|
| Set High | Set output A or B to a high level |
| Clear Low | Set output A or B to a low level |
| Toggle | Set output A or B to a low if currently set to a high<br>Set output A or B to a high if currently set to a low |
| Do Nothing | Keep outputs A or B at same level as currently set. |

These action types are independently possible on either output (A or B). Moreover, any or all events can be made to generate actions on a given output EPWMxA or EPWMxB. For example, both CTR = CMPA and CTR = CMPB can be made to operate on output EPWMxA.

## 2.4.3 Waveform Construction

The AQ module can drive or control two independent outputs: EPWMA and EPWMB. All qualifier actions that are available to you are programmable via the control registers found at the end of this section. However, for clarity, a set of symbolic actions has been summarized in Figure 2−16. Each symbol represents an action as a marker in time. Some actions are fixed in time; for example, zero and period, while the CMPA and CMPB actions are moveable and their time positions are programmed via the compare A and B registers, respectively. To turn off or disable an action, you can use the Do Nothing option; it is the default at reset.

*Figure 2−16. Set of AQ Qualifier Actions Possible for EPWMA and EPWMB*



| 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 |

Although the Do Nothing option prevents an event from causing an action on the EPWMxA/B outputs, this event can still be used to trigger interrupts and ADC start of conversion. See the “Event-trigger (ET) Module” section for details.

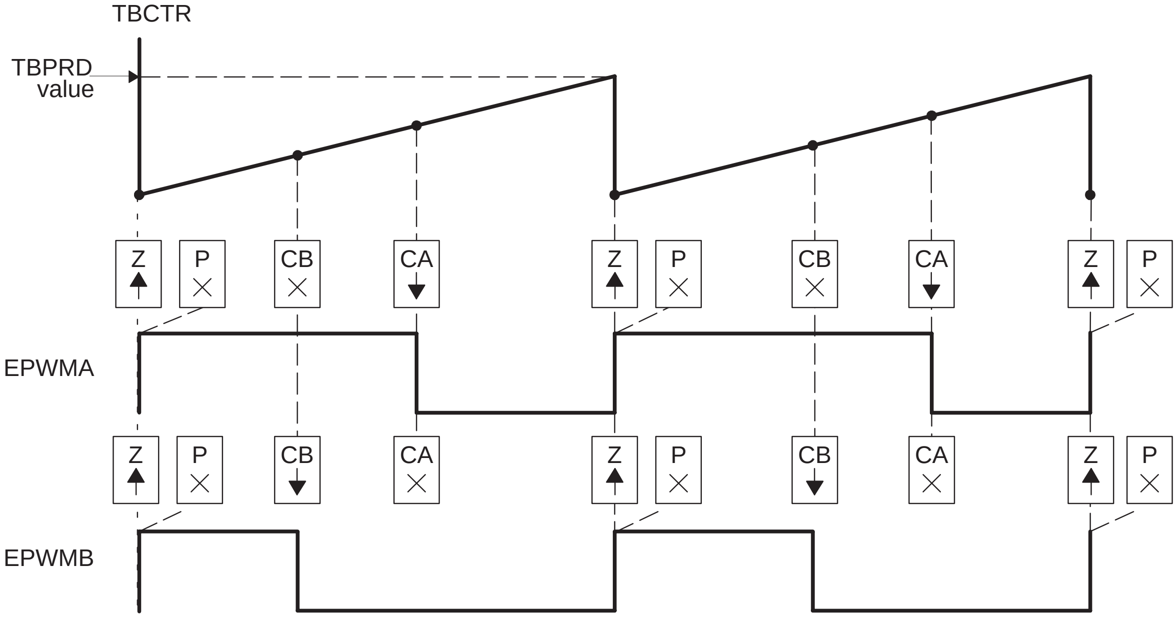
### 2.4.4 Waveforms for Common Configurations

The PWM waveforms in Figure 2−17 through Figure 2−22 show some common configurations. The C code samples in Example 2−2 through Example 2−7 shows you how to configure an EPWMx 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.

- ❑ If x is used in a name, for example, CMPx, then it refers to either CMPA or CMPB.
- ❑ Up-Down = Up and Down Count mode, Up = Up count mode, Dwn = Down count mode
- ❑ Sym = Symmetric, Asym = Asymmetric

*Figure 2−17. Up, Single Edge Asymmetric Waveform, With Independent Modulation on EPWMA and B—Active High*



**Notes:** 1) PWM period = (TBPRD + 1 ) × $T_{TBCLK}$

2) Duty modulation for EPWMA is set by CMPA, and is Active High, i.e., High Time Duty proportional to CMPA

3) Duty modulation for EPWMB is set by CMPB and is Active High, i.e., High Time Duty proportional to CMPB

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

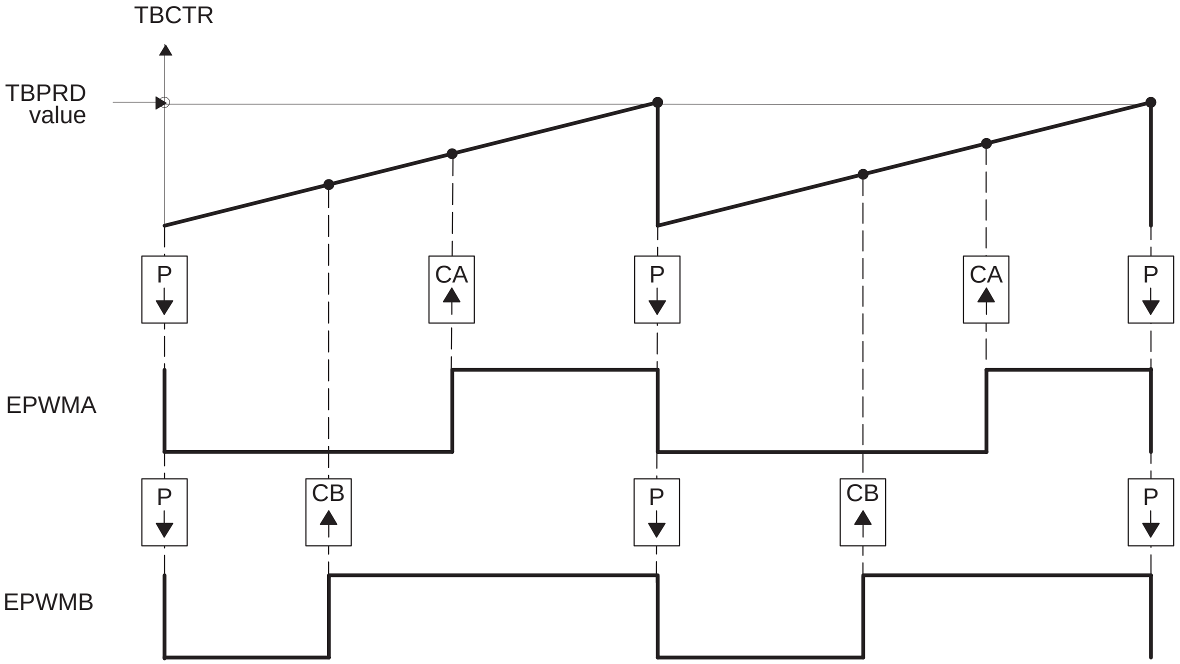
5) Actions at zero and period, although appearing to occur concurrently, are actually separated by one TBCLK period, i.e., TBCTR wraps from period value 0000.

Example 2−2 contains a code sample showing initialization and run time for the waveforms in Figure 2−17.

## *Example 2−2. Code Sample for Figure 2−17*

```
// Initialization Time
// = = = = = = = = = = = = = = = = = = = = = = = =  =
    EPwm1Regs.TBPRD = 600;                   // Period = 601 TBCLK counts
    EPwm1Regs.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 = Duty1A;             // adjust duty for output EPWM1A
    EPwm1Regs.CMPB = Duty1B;             // adjust duty for output EPWM1B
```

*Figure 2−18. Up, Single Edge Asymmetric Waveform With Independent Modulation on EPWMA and B—Active Low*



**Notes:** 1) PWM period = (TBPRD + 1 ) × $T_{TBCLK}$

2) Duty modulation for EPWMA is set by CMPA, and is Active Low, i.e., Low Time Duty proportional to CMPA

3) Duty modulation for EPWMB is set by CMPB and is Active Low, i.e., Low Time Duty proportional to CMPB

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

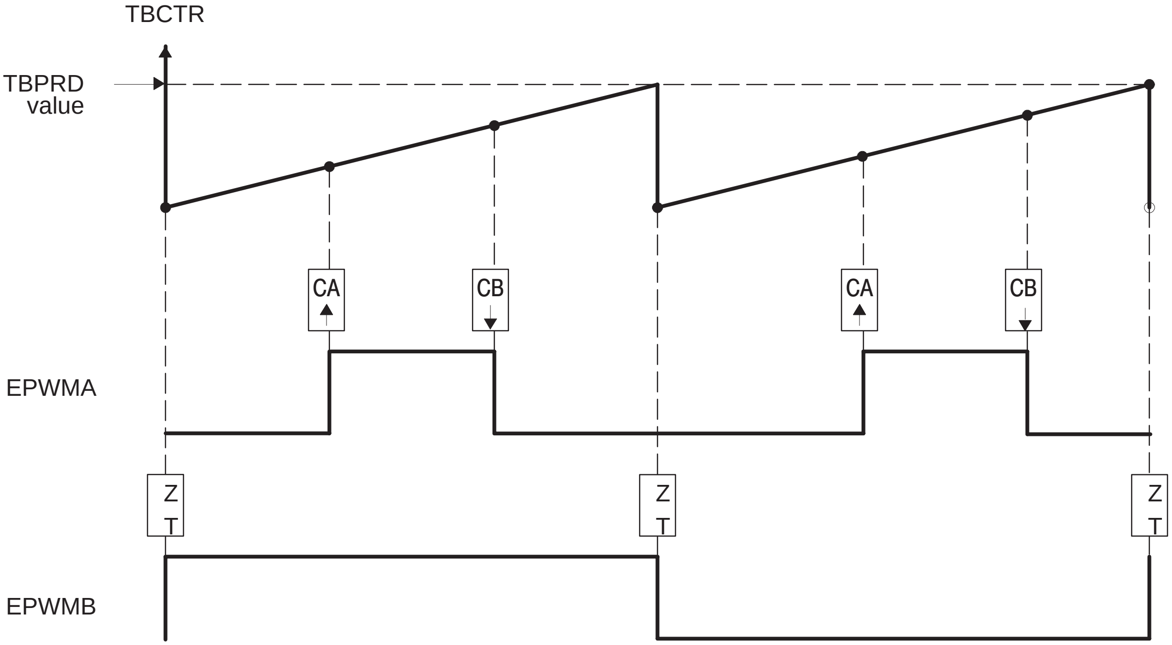
5) Actions at zero and period, although appearing to occur concurrently, are actually separated by one TBCLK period, i.e., TBCTR wraps from period value à 0000.

Example 2−3 contains a code sample showing initialization and run time for the waveforms in Figure 2−18.

### *Example 2−3. Code Sample for Figure 2−18*

```
// Initialization Time
// = = = = = = = = = = = = = = = = = = = = = = = =  =
    EPwm1Regs.TBPRD = 600;                      // Period = 601 TBCLK counts
    EPwm1Regs.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.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 = Duty1A;                    // adjust duty for output EPWM1A
    EPwm1Regs.CMPB = Duty1B;                    // adjust duty for output EPWM1B
```

*Figure 2−19. Up, Pulse Placement Asymmetric Waveform With Independent Modulation on EPWMA*



**Notes:**

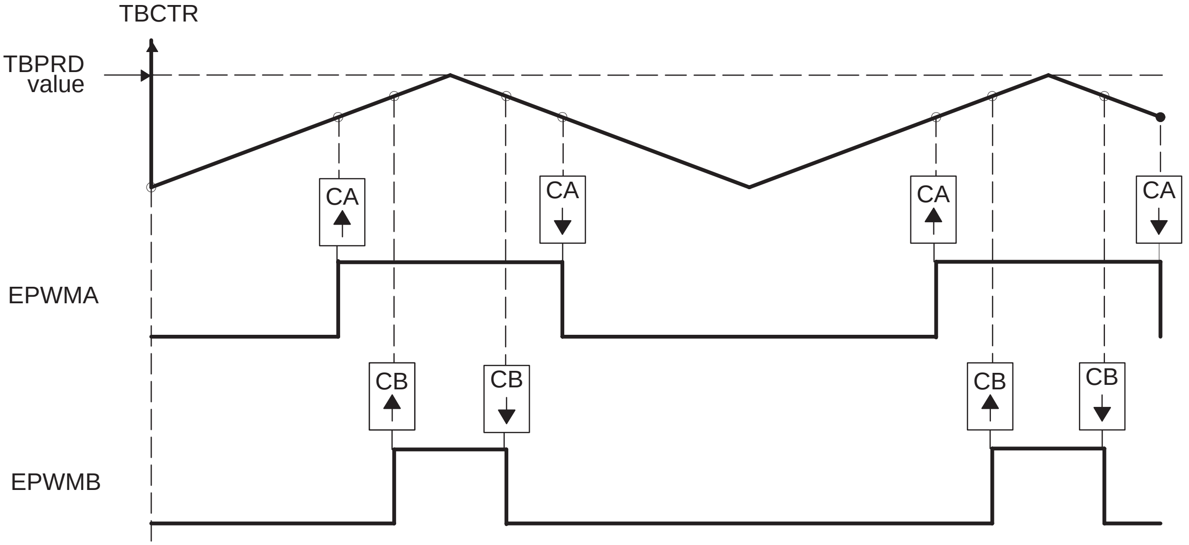
1) PWM frequency = 1/( (TBPRD + 1 ) × $T_{TBCLK}$ )
2) Pulse can be placed anywhere within the PWM cycle (0000 à TBPRD)
3) High Time Duty proportional to (CMPB – CMPA)
4) EPWMB can be used to generate a 50% duty square wave with frequency = 1/2 × ( (TBPRD + 1 ) × TBCLK )

Example 2−4 contains a code sample showing initialization and run time for the waveforms Figure 2−19. Use the code in Example 2−1 to define the headers.

## Example 2−4. Code Sample for Figure 2−19

```
// Initialization Time
// = = = = = = = = = = = = = = = = = = = = = = =   =
    EPwm1Regs.TBPRD = 600;                      // Period = 601 TBCLK counts
    EPwm1Regs.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 = 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.CAU = AQ_SET;
    EPwm1Regs.AQCTLA.bit.CBU = AQ_CLEAR;
    EPwm1Regs.AQCTLB.bit.ZRO = AQ_TOGGLE;

// Run Time
// = = = = = = = = = = = = = = = = = = = = = = =   =
    EPwm1Regs.CMPA = EdgePosA;       // adjust duty for output EPWM1A only
    EPwm1Regs.CMPB = EdgePosB;
```

*Figure 2−20. Up-Down, Dual Edge Symmetric Waveform, With Independent Modulation on EPWMA and B—Active Low*



**Notes:** 1) PWM period = 2 x TBPRD × $T_{TBCLK}$

2) Duty modulation for EPWMA is set by CMPA, and is active low, i.e., Low Time Duty proportional to CMPA

3) Duty modulation for EPWMB is set by CMPB and is active low, i.e., Low Time Duty proportional to CMPB

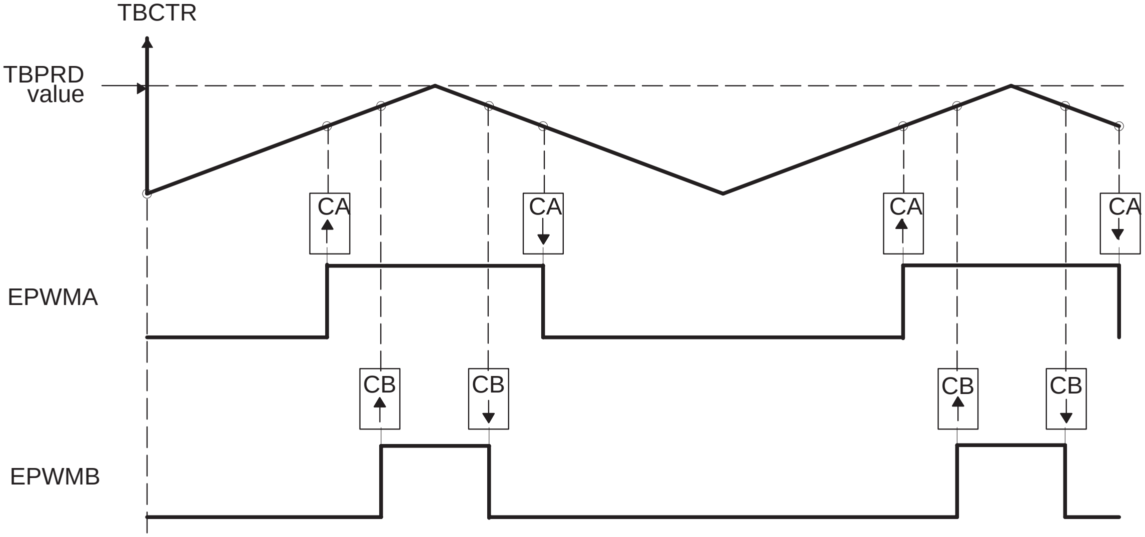
4) Outputs EPWMx can drive independent power switches

Example 2−5 contains a code sample showing initialization and run time for the waveforms in Figure 2−20. Use the code in Example 2−1 to define the headers.

### *Example 2−5. Code Sample for Figure 2−20*

```
// Initialization Time
// = = = = = = = = = = = = = = = = = = = = = = = =  =
    EPwm1Regs.TBPRD = 600;                    // Period = 2 x 600 TBCLK counts
    EPwm1Regs.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
    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.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 = Duty1A;          // adjust duty for output EPWM1A
    EPwm1Regs.CMPB = Duty1B;          // adjust duty for output EPWM1B
```

*Figure 2−21. Up-Down, Dual Edge Symmetric Waveform, With Independent Modulation on EPWMA and B—Complementary*



**Notes:**

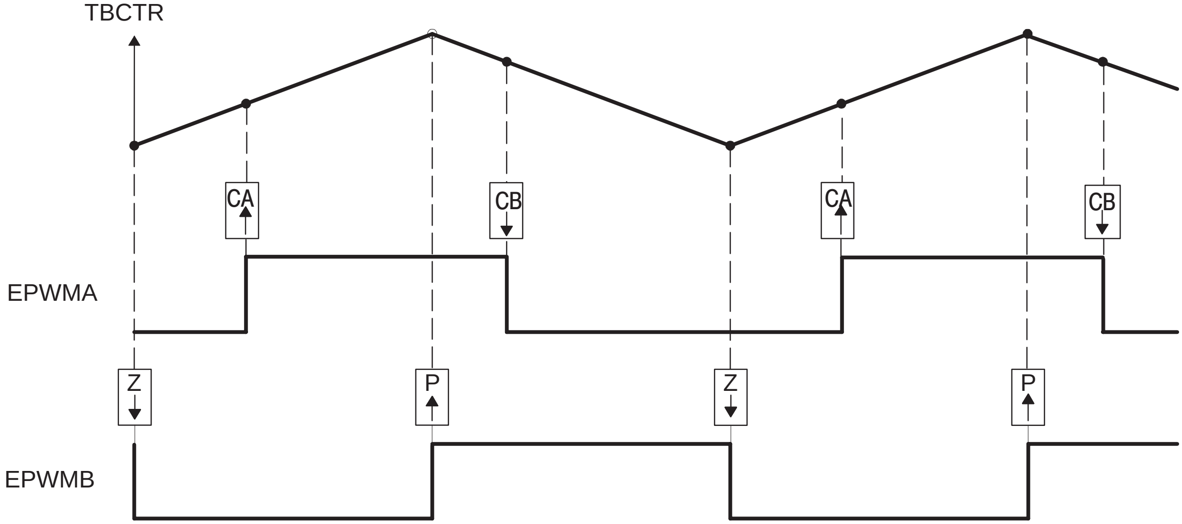
1) PWM period = 2 × TBPRD × $T_{TBCLK}$
2) Duty modulation for EPWMA is set by CMPA, and is active low, i.e., Low Time Duty proportional to CMPA
3) Duty modulation for EPWMB is set by CMPB and is active high, i.e., High Time Duty proportional to CMPB
4) Outputs EPWMx can drive upper/lower (complementary) power switches
5) 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 2−21. Use the code in Example 2−1 to define the headers.

### Example 2−6. Code Sample for Figure 2−21

```
// Initialization Time
// = = = = = = = = = = = = = = = = = = = = = = = =  =
    EPwm1Regs.TBPRD = 600;                        // Period = 2 × 600 TBCLK counts
    EPwm1Regs.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 = 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.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 = Duty1A;                      // adjust duty for output EPWM1A
    EPwm1Regs.CMPB = Duty1B;                      // adjust duty for output EPWM1B
```

*Figure 2−22. Up-Down, Dual Edge Asymmetric Waveform, With Independent Modulation on EPWMA—Active Low*



**Notes:**

1) PWM period = 2 × TBPRD × TBCLK
2) Rising edge and falling edge can be asymmetrically positioned within a PWM cycle. This allows for pulse placement techniques.
3) Duty modulation for EPWMA is set by CMPA and CMPB
4) Low time duty for EPWMA proportional to (CMPA + CMPB)
5) To change this example to active high, CMPA and CMPB actions need to be inverted, i.e., Set → Clear and Clear Set.
6) Duty modulation for EPWMB is fixed at 50% (utilizes spare action resources for EPWMB)

Example 2−7 contains a code sample showing initialization and run time for the waveforms in Figure 2−22. Use the code in Example 2−1 to define the headers.

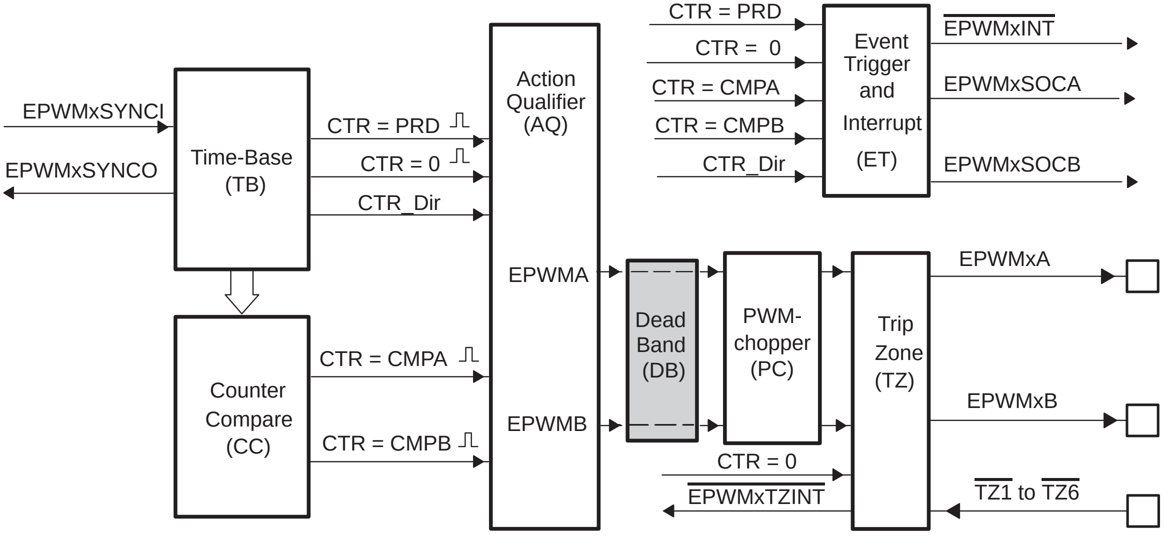
### Example 2−7. Code Sample for Figure 2−22

```
// Initialization Time
// = = = = = = = = = = = = = = = = = = = = = = = =  =
    EPwm1Regs.TBPRD = 600;                        // Period = 2 × 600 TBCLK counts
    EPwm1Regs.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 = 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.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 = EdgePosA;                    // adjust duty for output EPWM1A only
    EPwm1Regs.CMPB = EdgePosB;
```

## 2.5 Dead-Band Generator (DB) Module

Figure 2−23 illustrates the DB sub-module within the ePWM module.

*Figure 2−23. DB Module*



### 2.5.1 Purpose of the DB Module

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 DB module described here should be used.

The key functions of the DB module are:

- ❑ Generating appropriate signal pairs (EPWMxA and EPWMxB) with dead-band relationship from a single EPWMA 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 DB Module

The DB module operation is controlled and monitored via the following registers:

*Table 2−9. Dead-Band Generator Module Registers*

| mnemonic | Address offset | Shadowed | Description |
|---|---|---|---|
| DBCTL | 0x000F | No | DBM Control Register |
| DBRED | 0x0010 | No | DBM Rising Edge Delay Count Register |
| DBFED | 0x0011 | No | DBM Falling Edge Delay Count Register |

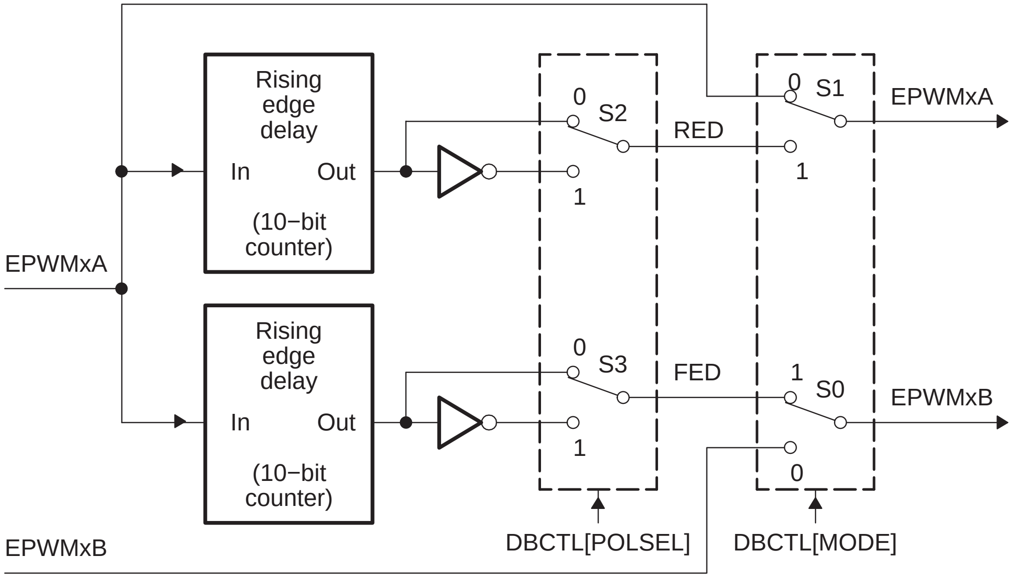
### 2.5.3 Operational Highlights for the DB Module

The following sections provide the operational highlights.

The DB module has two groups of independent selection options as shown in Figure 2−24.

1) Mode control (MODE bits)
2) Polarity control (POLSEL bits)

*Figure 2−24. Configuration Options for the DB Module*



Although all combinations are valid (supported), not all are typical usage modes. However if you find these useful in certain applications, Table 2−10 documents them.

The first entry (choice) allows you to fully disable the DB module from the PWM signal path.

The next four entries 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 2−25. Note that to generate equivalent waveforms to Figure 2−25, configure the AQ module to generate the signal as shown for EPWMxA.

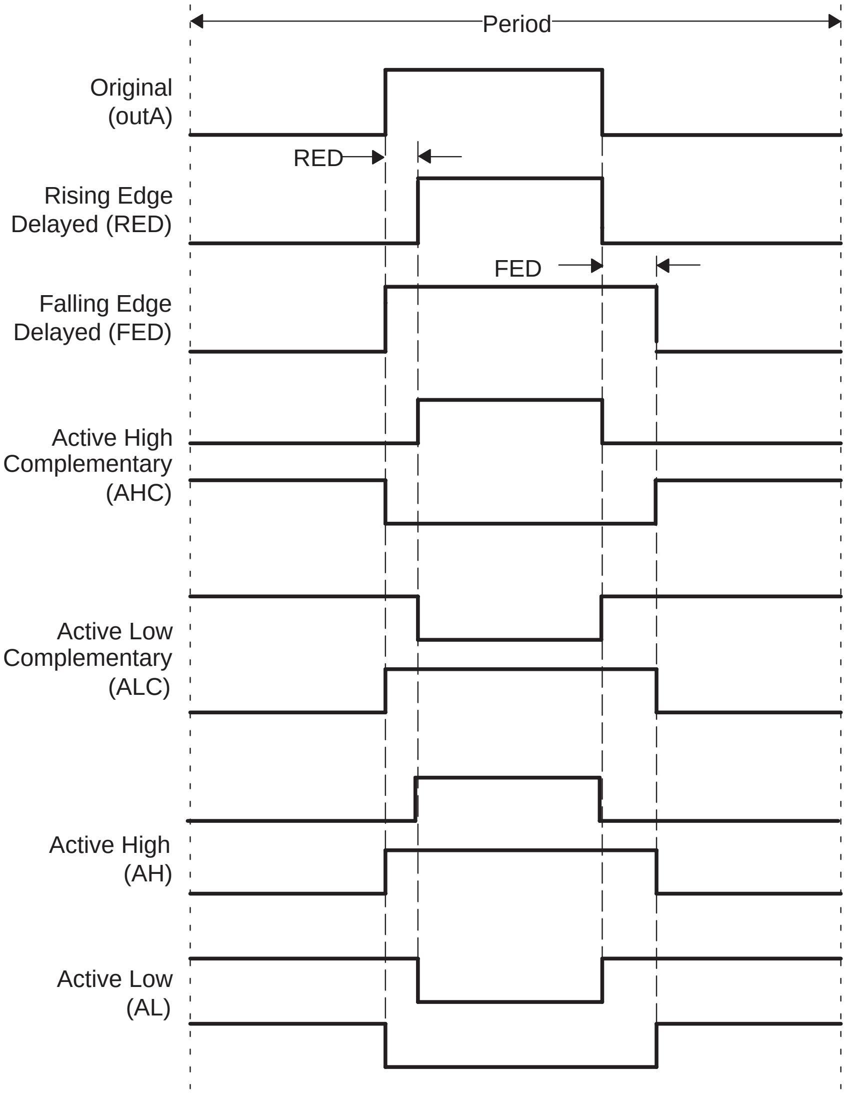
Finally the last two entries in Table 2−10 show combinations where either the FED or RED blocks are bypassed.

*Table 2−10. Operating Modes Supported by the DB Module*

| | | POLSEL | | MODE | |
|---|---|---|---|---|---|
| **Use** | **Deadband Operating Modes** | **S3** | **S2** | **S1** | **S0** |
| Disable DB from PWM | Disabled (by-pass mode) | X | X | 0 | 0 |
| Typical polarity settings | Active High Complementary (AHC) | 1 | 0 | 1 | 1 |
| | Active Low Complementary (ALC) | 0 | 1 | 1 | 1 |
| | Active High (AH) | 0 | 0 | 1 | 1 |
| | Active Low (AL) | 1 | 1 | 1 | 1 |
| Bypass either FED or RED blocks | iPWMxA = OutA (i.e., no delay)<br>iPWMxB = FED (delay falling edge) | 0 or 1 | 0 or 1 | 0 | 1 |
| Bypass either FED or RED blocks | iPWMxA = RED (delay rising edge)<br>PWMxB = OutB (i.e., no delay) | 0 or 1 | 0 or 1 | 1 | 0 |

Figure 2−25 shows waveforms for typical cases where 0% < duty < 100% .

*Figure 2−25. DB Module Waveforms for Typical Cases (0% < Duty < 100%)*



The DB module supports independent values for rising edge (RED) and falling edge (FED) delays. The amount of delay is programmed using the DBRED and DBFED memory-mapped registers. These are 10-bit registers and their value represents the number of TBCLK periods a signal edge is delayed by. The formula to calculate FED for example, (RED is calculated the same way), is:

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

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 2−11.

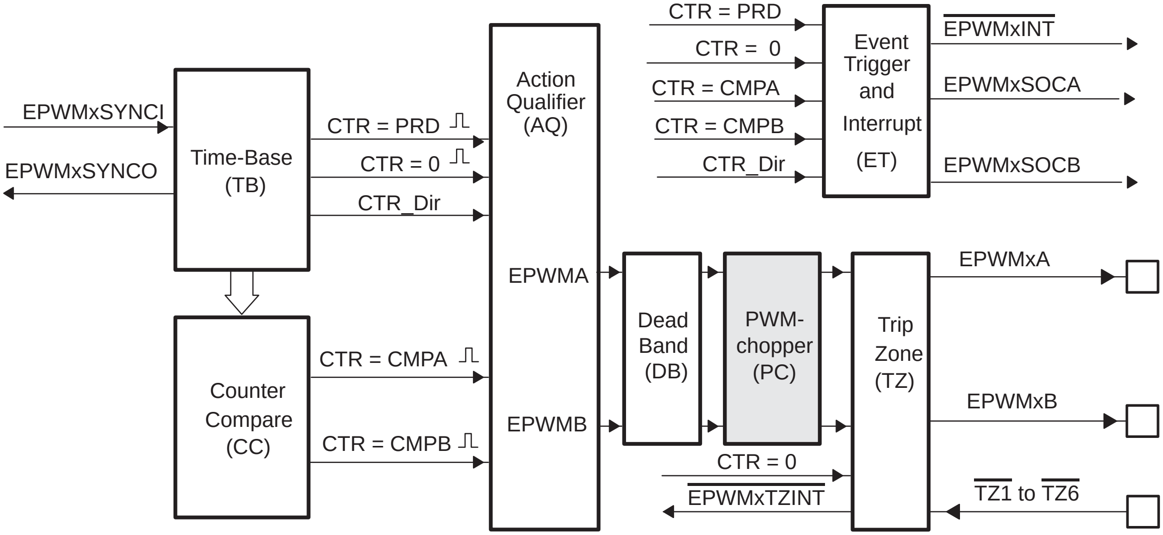
*Table 2−11. DBM Delay Values in μS as a Function of DBFED and DBRED (Table Values Calculated Based on SYSCLKOUT = 100 MHz)*

| DB value | TBCLK = SYSCLKOUT/PS | | |
|---|---|---|---|
| **DBFED, DBRED** | **SYSCLKOUT/1** | **SYSCLKOUT /2** | **SYSCLKOUT/4** |
| 1 | 0.01 | 0.02 | 0.04 |
| 5 | 0.05 | 0.10 | 0.20 |
| 10 | 0.10 | 0.20 | 0.40 |
| 100 | 1.00 | 2.00 | 4.00 |
| 200 | 2.00 | 4.00 | 8.00 |
| 300 | 3.00 | 6.00 | 12.00 |
| 400 | 4.00 | 8.00 | 16.00 |
| 500 | 5.00 | 10.00 | 20.00 |
| 600 | 6.00 | 12.00 | 24.00 |
| 700 | 7.00 | 14.00 | 28.00 |
| 800 | 8.00 | 16.00 | 32.00 |
| 900 | 9.00 | 18.00 | 36.00 |
| 1000 | 10.00 | 20.00 | 40.00 |

## 2.6 PWM-chopper (PC) Module

Figure 2−26 illustrates the PWM-chopper (PC) sub-module within the ePWM module.

*Figure 2−26. PC Module*



The PC module allows a high-frequency carrier signal to modulate the PWM waveform generated by the AQ and DB modules. This capability is important if you need pulse transformer-based gate drivers to control the power switching elements.

### 2.6.1 Purpose of the PC Module

The key functions of the PC module 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 PC Module

The PC module operation is controlled via the registers in Table 2−12.

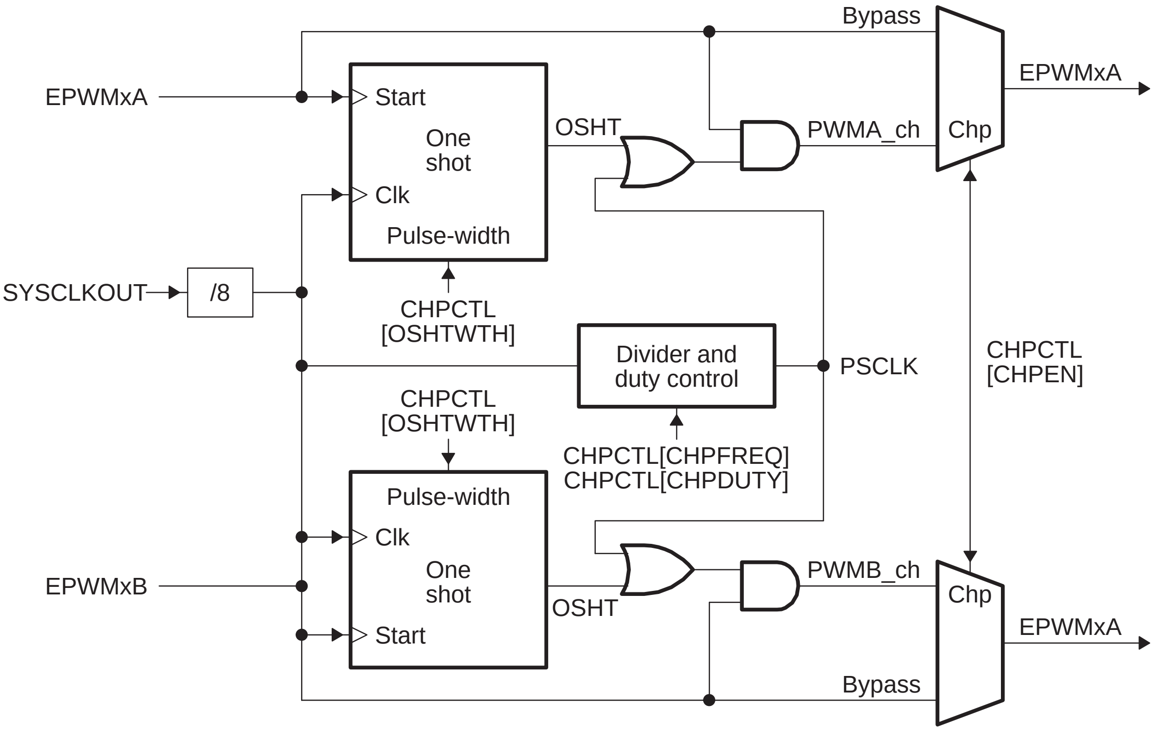
*Table 2−12. PWM-chopper Module Registers*

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

## 2.6.3 Operational Highlights for the PC Module

Figure 2−27 shows the operational details of the PCM. The carrier clock is derived from SYSCLK. Its frequency and duty cycle are controlled via the CHPFREQ and CHPDUTY bits in the CHPCTL 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 PC module can be fully disabled (bypassed) via the CHPEN bit.

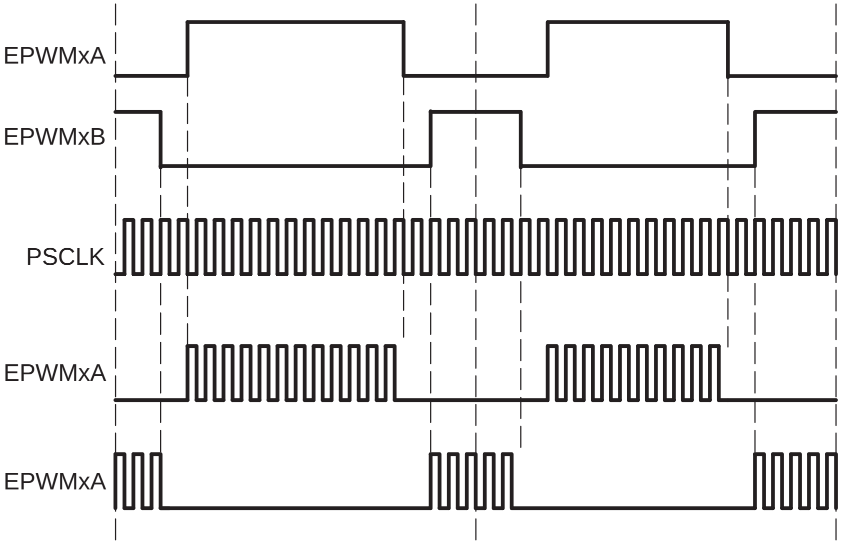
*Figure 2−27. PC Module Operational Details*



## 2.6.4 Waveforms

Figure 2−28 shows simplified waveforms of the chopping action only; one-shot and duty control are not shown. Details of the one-shot and duty-cycle control are discussed in the following sections.

*Figure 2−28. Simple PC Module 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 system clock period and OSHTWTH is the four control bits (value from 1 to 16)

Figure 2−29 shows the first and subsequent sustaining pulses and Table 7.3 gives the possible pulse width values for a SYSCLKOUT = 100 MHz.

*Figure 2−29. PC Module Waveforms Showing the First Pulse and Subsequent Sustaining Pulses*

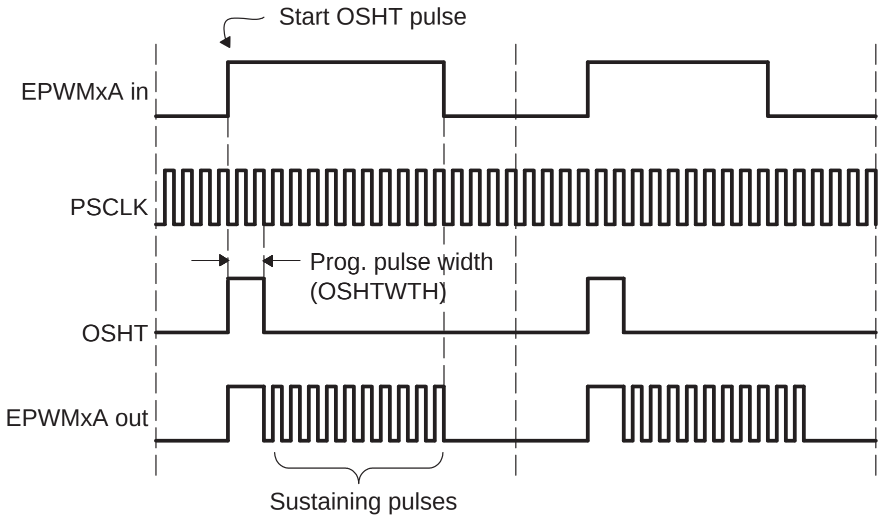


*Table 2−13. Possible Pulse Width Values for SYSCLKOUT = 100 MHz*

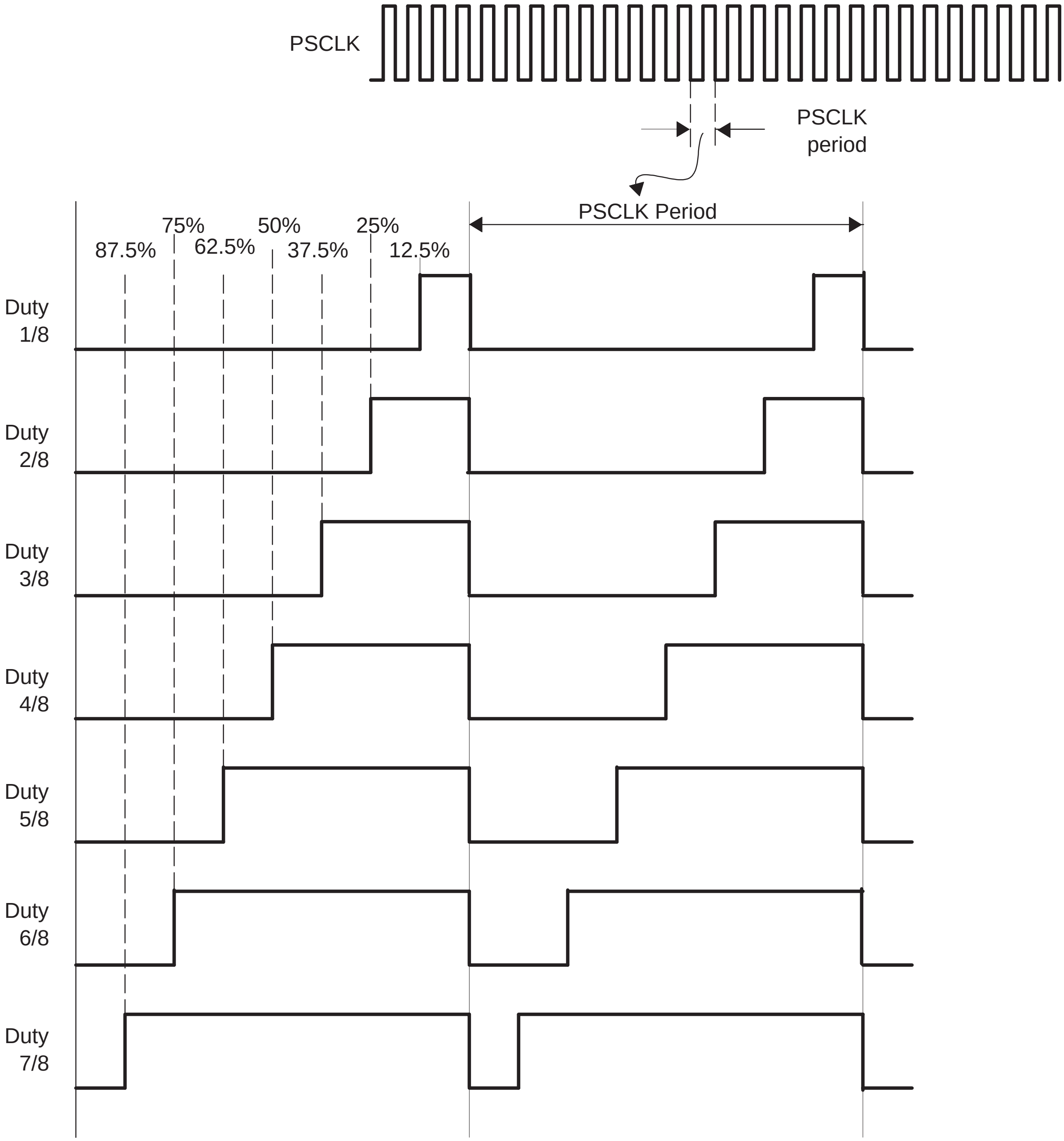
| OSHTWTHz (hex) | Pulse width (nS) |
|---|---|
| 0 | 80 |
| 1 | 160 |
| 2 | 240 |
| 3 | 320 |
| 4 | 400 |
| 5 | 480 |
| 6 | 560 |
| 7 | 640 |
| 8 | 720 |
| 9 | 800 |
| A | 880 |
| B | 960 |
| C | 1040 |
| D | 1120 |
| E | 1200 |
| F | 1280 |

#### 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 2−30 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%.

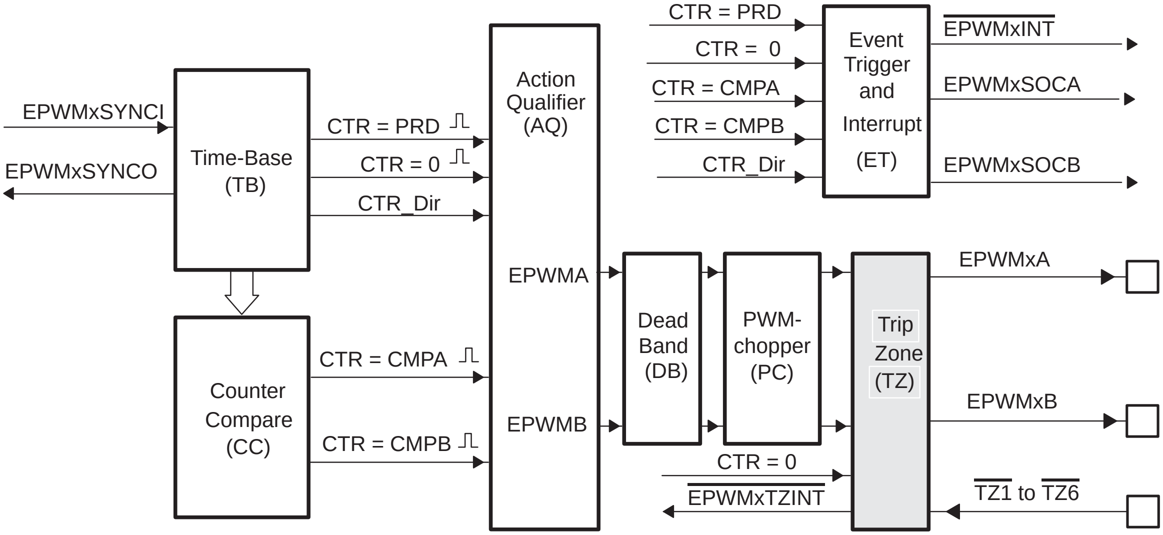
*Figure 2−30. PC Module Waveforms Showing the Pulse Width (Duty Cycle) Control of Sustaining Pulses*



## 2.7 Trip-zone (TZ) Module

Figure 2−31 shows how the trip-zone (TZ) sub-module fits within the ePWM module.

*Figure 2−31. TZ Module*



Each ePWM module is connected to six TZ signals ($\overline{\text{TZ1}}$ to $\overline{\text{TZ6}}$) 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 TZ Module

The key functions of the TZ module are:

- Trip inputs $\overline{\text{TZn}}$ (i.e., $\overline{\text{TZ1}}$ to $\overline{\text{TZ6}}$ ) 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
  - Hi-Z (high impedance)
  - No action taken
- It supports one-shot trip (OSHT) for major short circuits or over-current conditions.

- ❑ It supports cycle-by-cycle tripping (CBC) for current limiting operation.
- ❑ TZs can be allocated to either OSHT or CBC operation.
- ❑ Interrupt generation is possible on any trip-zone.
- ❑ Software-forced tripping is also supported.
- ❑ The TZ module can be fully bypassed if not required.

### 2.7.2 Controlling and Monitoring the TZ Module

The TZ module operation is controlled and monitored via the following registers:

*Table 2−14. Trip-zone Module Registers*

| mnemonic | Address offset | Shadowed | Description |
|---|---|---|---|
| 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 |

**Note:** All TZ module registers are EALLOW protected. They can by modified only after executing the EALLOW instruction.

### 2.7.3 Operational Highlights for the TZ Module

The following sections describe the operational highlights and configuation options for the TZ module.

The trip-zone signals at pins $\overline{\text{TZn}}$ are active low signals. The 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 $\overline{\text{TZn}}$ inputs is sufficient to trigger a fault condition in the EPWMx module.

You can configure two events for each output and based on these events, you can specify a trip action on the respective outputs. The trip event is forced on the respective outputs on the next valid TBCLK edge.

The trip-zone signals ($\overline{TZn}$) support the following two operating trip modes:

Cycle-by-Cycle Mode (CBC): In this mode, whenever a selected trip event is active (TZn inputs) the respective action on the EPWMxA/B output is carried out immediately and latched. The condition is automatically cleared when the PWM Counter reaches zero, i.e., on a CTR = Zero event. Therefore, in this mode, the trip event is cleared or reset every PWM cycle.

One-Shot Mode (OSHT): In this mode, whenever a selected trip event is active (TZn inputs) the respective action on the EPWMxA/B output is carried out immediately and is latched. The condition remains latched and can only be cleared by the user under software control.

The actions on the EPWMxA/B output signals of the module when a trip condition occurs are:

| No. | State of EPWMxA or B | Comment |
|---|---|---|
| 1 | Hi-Z | Tripped |
| 2 | Force Hi | Tripped |
| 3 | Force Lo | Tripped |
| 4 | No Change | Do Nothing (signal passes through) |

Figure 2−32 shows the key operational blocks for the TZ module. Each TZn input can be mapped (allocated) to provide a trip event for either CBC or OSHT operation. The mapping for this is selected via the TZSEL[CBCn] and TZSEL[OSHTn] control bits respectively.

The action or forced state that occurs on outputs EPWMA0 and EPWMB0 once a trip occurs, is controlled by the trip logic block and is programmed via the TZCTL[TZA] and TZCTL[TZB] control bits respectively.

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 TZn inputs, providing the GPIO is appropriately configured.

### 2.7.4 Generating Interrupts

Figure 2−32 and Figure 2−33 illustrate the TZ module control and interrupt logic, respectively.

*Figure 2−32. TZ Module Mode Control Logic*

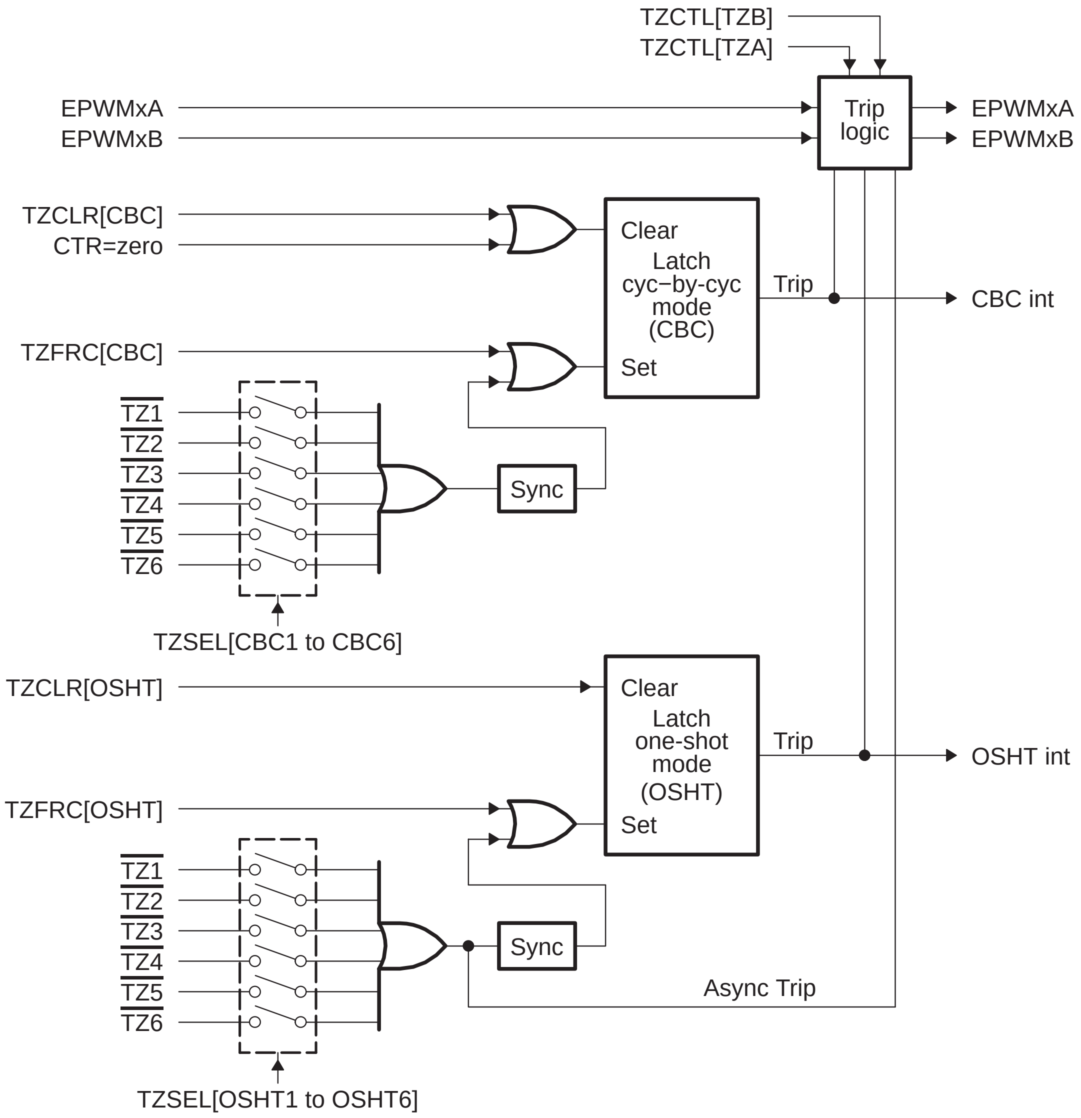
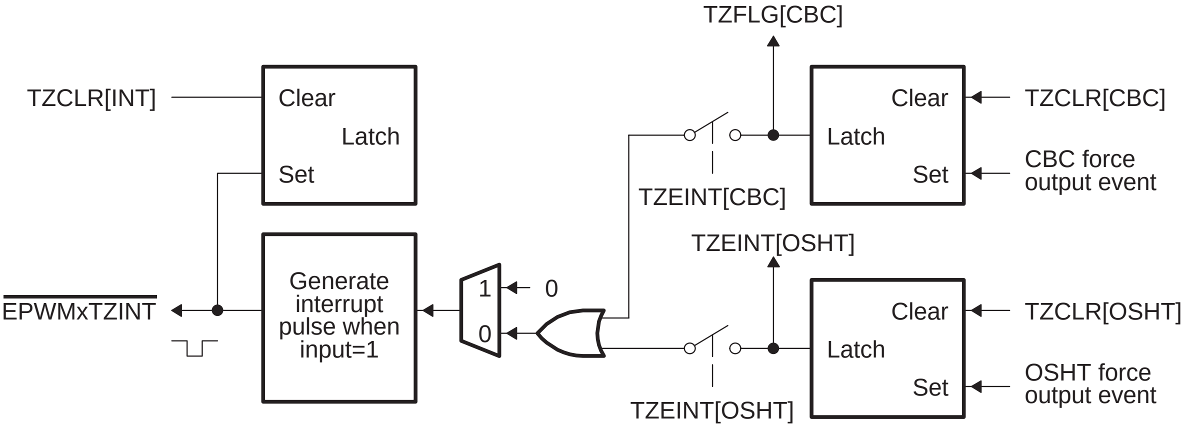


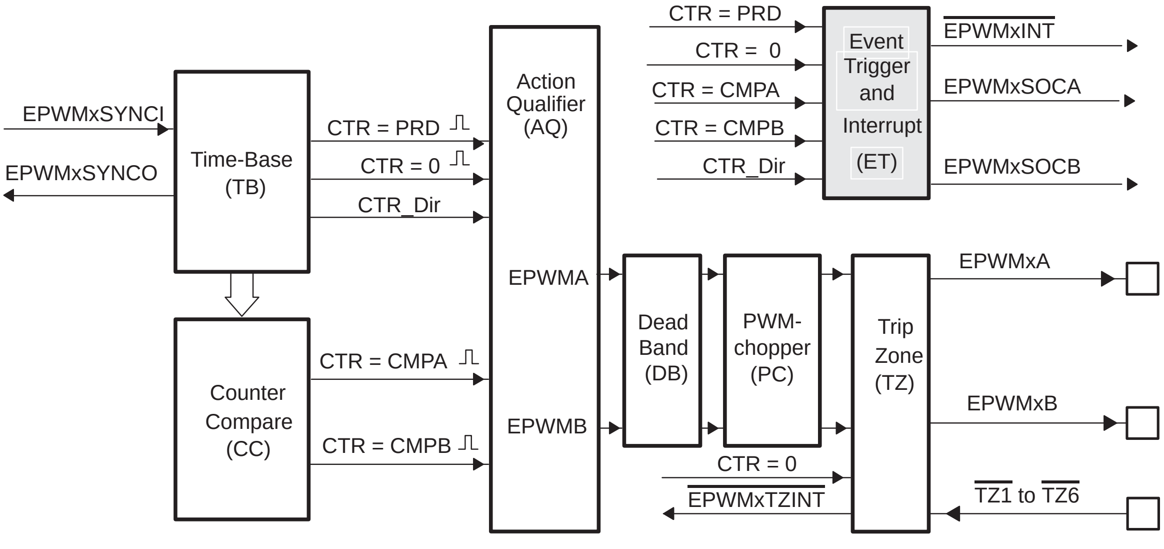
*Figure 2−33. TZ Module Interrupt Logic*



## 2.8 Event-trigger (ET) Module

Figure 2−34 illustrates the event-trigger (ET) module within the ePWM system.

*Figure 2−34. ET Module*



The ET module manages the events generated by the TB module and the CC module to generate an interrupt to the CPU and/or a start of conversion pulse to the ADC when a selected event occurs.

### 2.8.1 Purpose of the ET Module

The key functions of the ET module are:

- Receives event inputs generated by the TB and CC modules
- Uses the TB direction information for up/down event qualification
- Uses prescaling logic to issue interrupt requests and ADC SOC 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 SOC

### 2.8.2 Controlling and Monitoring the ET Module

The ET module operation is controlled and monitored via the following registers:

*Table 2−15. ET Module Registers*

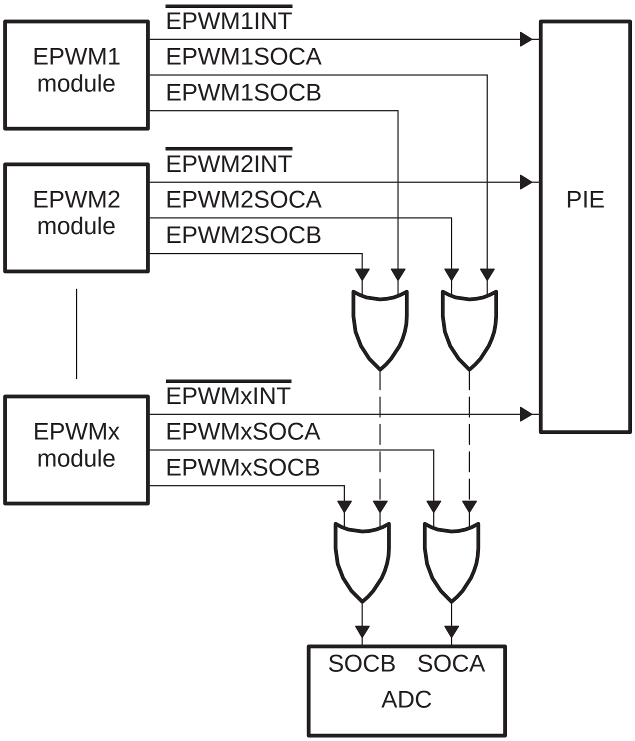
| mnemonic | Address offset | Shadowed | Description |
|---|---|---|---|
| ETSEL | 0x0019 | No | Event-trigger Selection Register |
| ETPS | 0x001A | No | Event-trigger Prescale Register |
| ETFLG | 0x001B | No | Event-trigger Flag Register |
| ETCLR | 0x001C | No | Event-trigger Clear Register |
| ETFRC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

### 2.8.3 Operational Highlights for the ET Module

The following sections describe the ET module's operational highlights.

Each ePWM module has one interrupt request line connected to the PIE and two SOC signals (one for each sequencer) connected to the ADC module. As shown in Figure 2−35, ADC SOCs for all ePWM modules are ORed together and hence multiple modules can initiate an ADC SOC.

*Figure 2−35. ET Module Inter-Module Connectivity of SOC and Interrupt Signals*

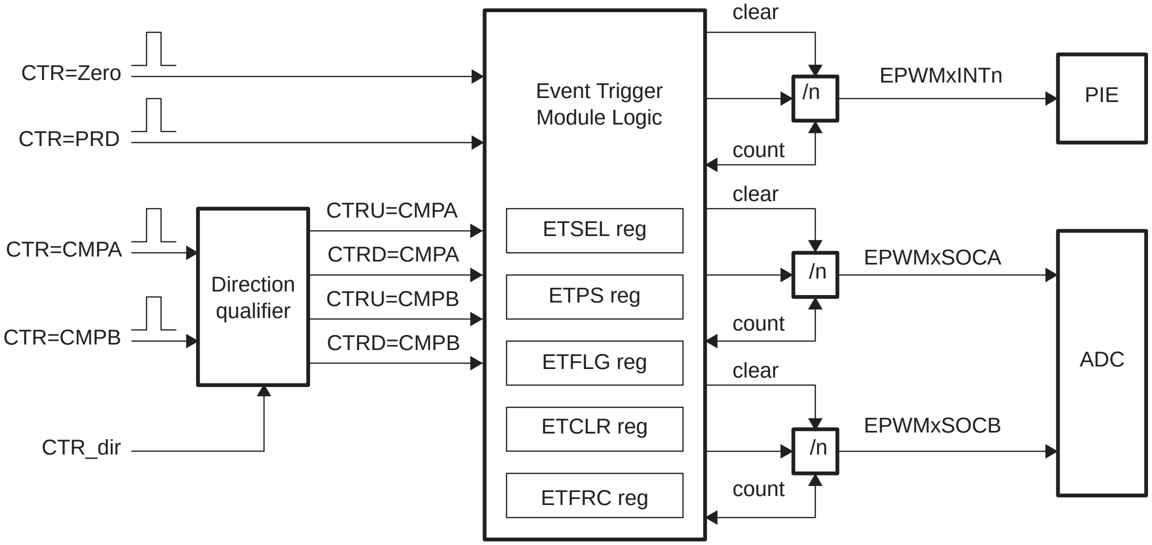


The ET module monitors various event conditions (the left side inputs to ET module shown in Figure 2−36) and can be configured to prescale these events before issuing an Interrupt request or an ADC SOC.

The ET prescaling logic can issue Interrupt requests and ADC SOC at:

❑ Every event
❑ Every second event
❑ Every third event

*Figure 2−36. ET Module Showing Event Inputs and Prescaled Outputs*



The key registers used to configure the ET module are:

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 SOC logic are shown in Figure 2−37, Figure 2−38, and Figure 2−39.

*Figure 2−37. ET Register Bits Interacting with ET Interrupt Generator*

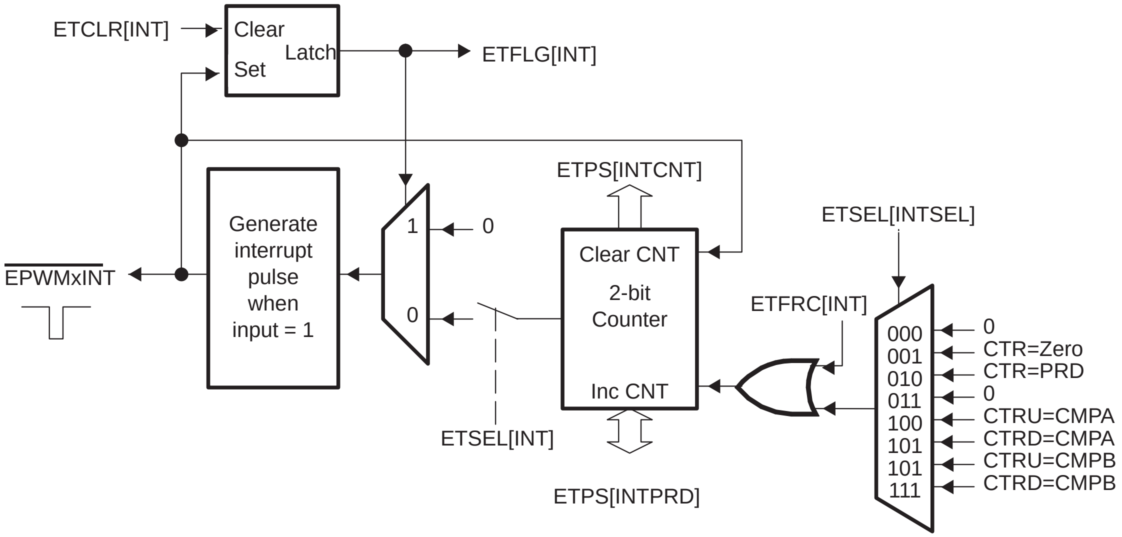


*Figure 2−38. ET Register Bits Interacting with ET SOCA Pulse Generator*

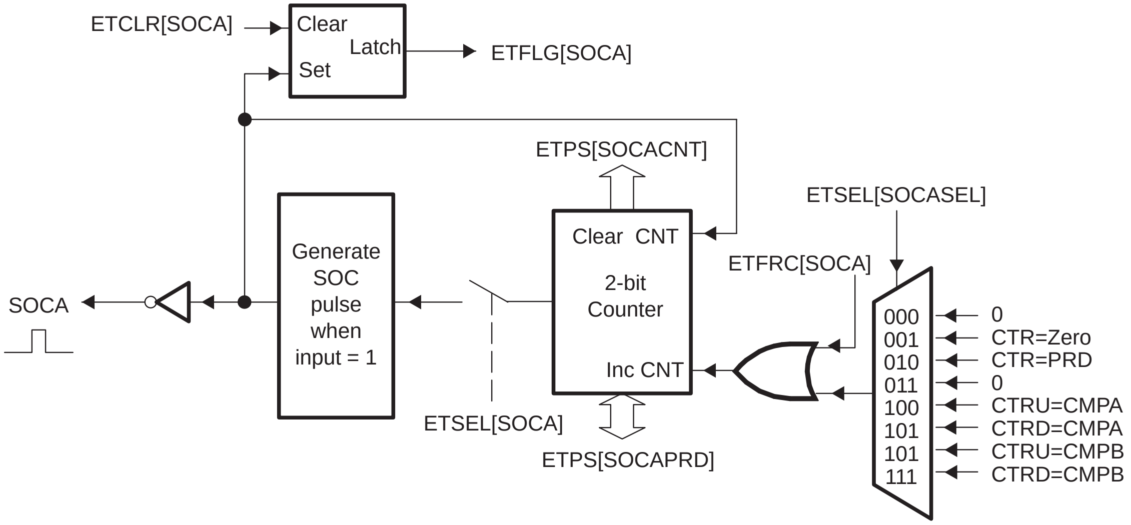


*Figure 2−39. ET Register Bits Interacting with ET SOCB Pulse Generator*

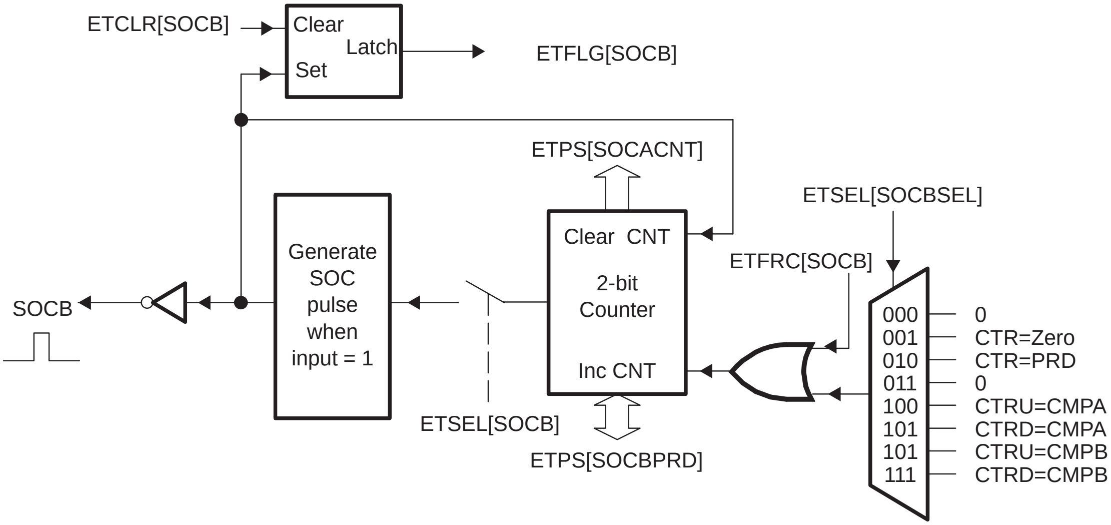


Figure 2−37 shows the INTPRD/INTCNT operation. The ETPS[INTPRD] bits specify the number of input events, as indicated by ETPS[INTCNT], which will cause an interrupt pulse to be generated.

❑ When INTCNT = INTPRD, then the counter stops counting and its output is set.

❑ If interrupts are enabled, ETSEL[INT = 1] and the interrupt flag is clear ETFLG[INT = 0], then an interrupt pulse is generated and the interrupt flag is set to ETFLG[INT = 1] and the event counter is automatically cleared INTCNT = 0.

❑ If interrupts are disabled, ETSEL[EINT = 0] or the interrupt flag is set ETFLG[INT = 1], the counter stops counting events when it reaches the period value INTCNT = INTPRD.

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 2−38 shows the operation of the ET 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. The pulse flag ETFLG[SOCA] is latched when a pulse is generated, but does not stop further pulse generation. The enable/disable bit ETPS[ESOCA] stops pulse generation, but input events can still be counted until the period value is reached (the same as way the interrupt generator operates).

Figure 2−39 shows the operation of the ET SOCB pulse generator. The ET SOCB pulse generator operates the same way as does the SOCA.

Chapter 3

# Applications to Power Topologies

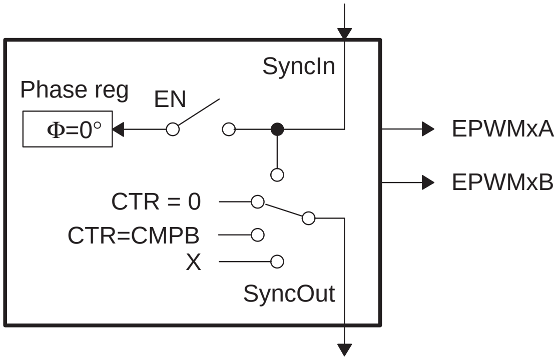
An ePWM module has all the local resources necessary to operate completely as a standalone module or to operate in synchronization with other identical ePWM modules.

**Topic** **Page**

## 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 Figure 1−3 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

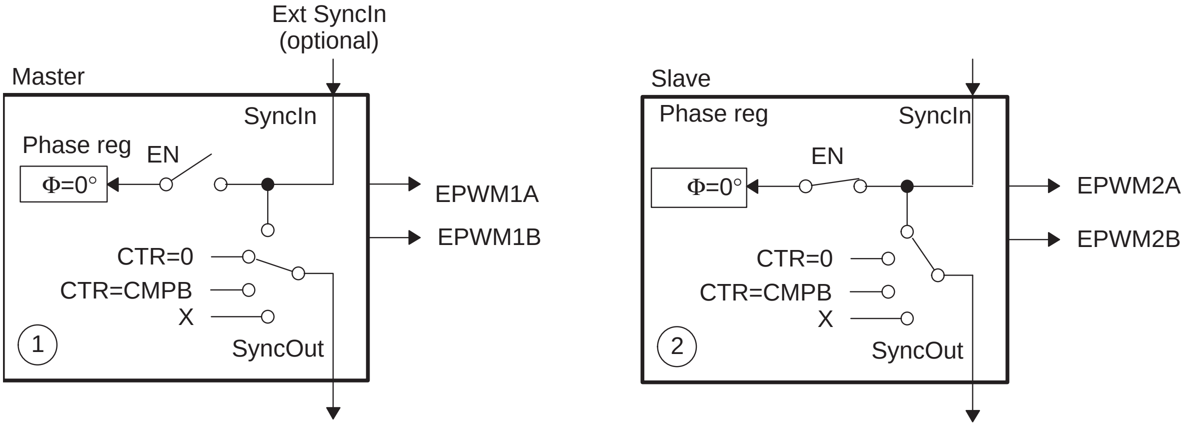
- Load own counter with phase register on an incoming sync strobe—enable (EN) switch closed
- Do nothing or ignore incoming sync strobe—enable switch open

❑ 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.

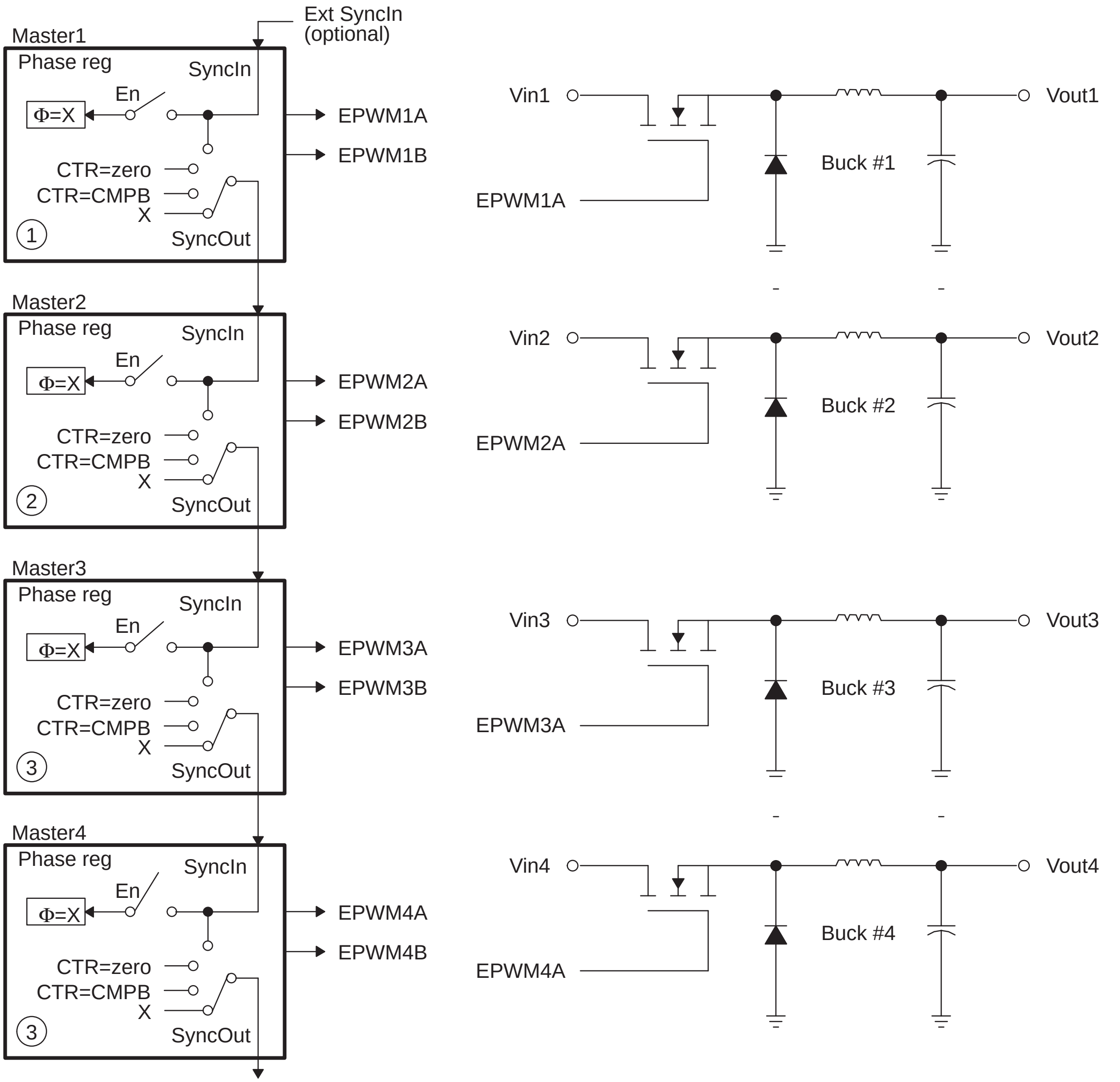
*Figure 3−2. EPWM1 Configured as a Typical Master, EPWM2 Configured as a Slave*



## 3.3 Controlling Multiple Buck Converters With Independent Frequencies

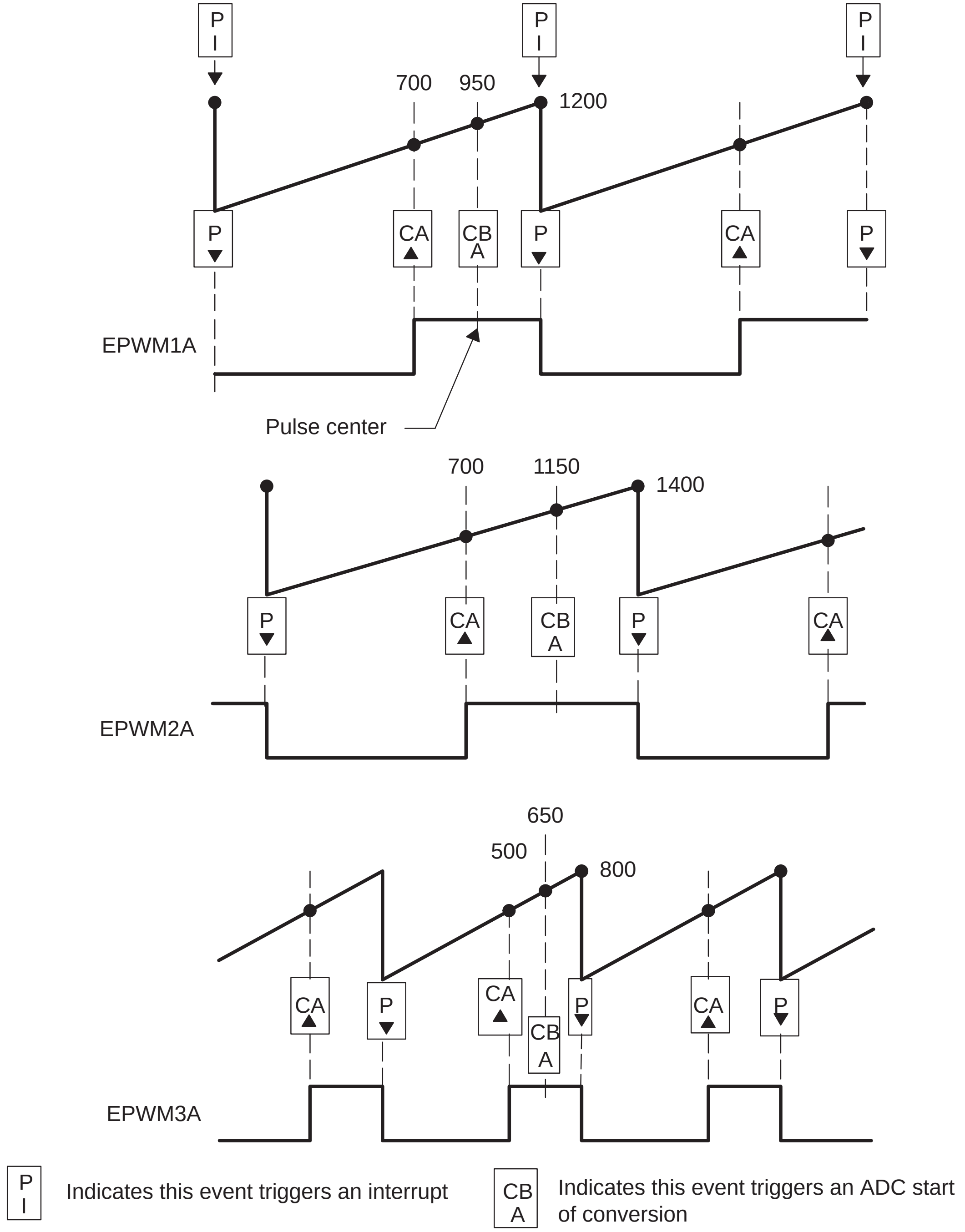
One of the simplest power converter topologies is the buck. A single ePWM module configured as a master can control two buck stages with the same PWM frequency. If independent frequency control is required for each buck converter, then one ePWM module must be allocated for each converter stage. Figure 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:** Φ = 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 = 700;                // adjust duty for output EPWM1A
    EPwm2Regs.CMPA = 700;                // adjust duty for output EPWM2A
    EPwm3Regs.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: $F_{PWM2} = N \times F_{PWM1}$)*

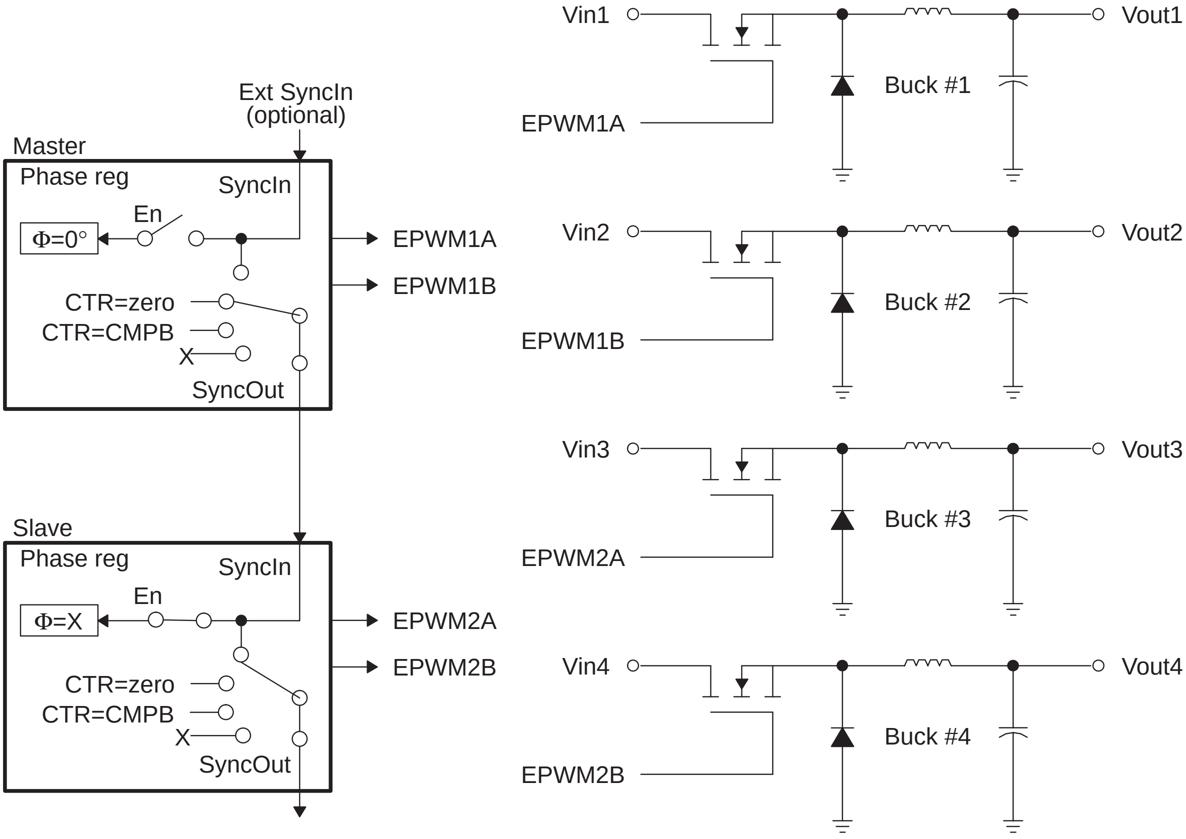
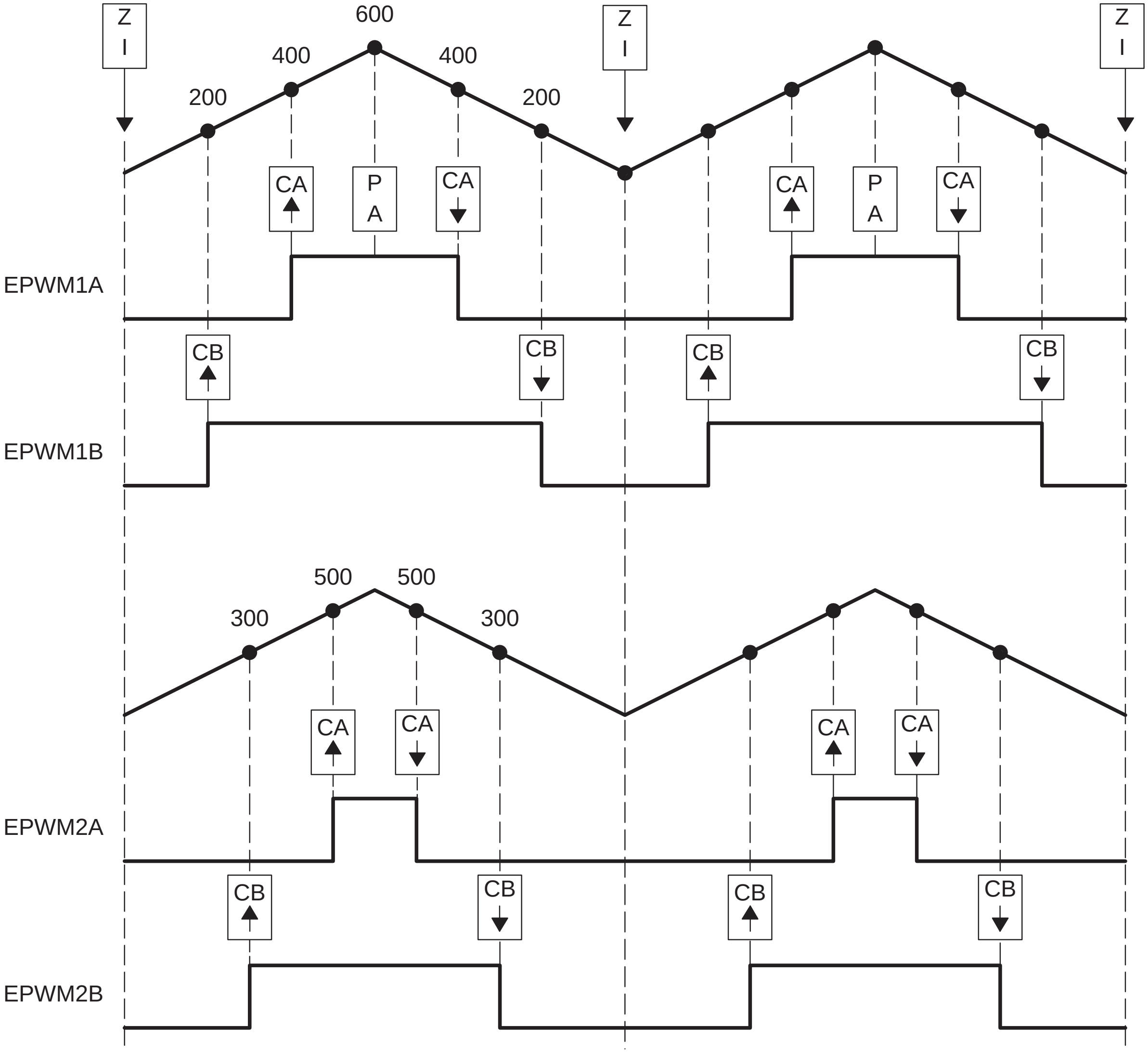


*Figure 3−6. Buck Waveforms for Figure 3−5 (Note: $F_{PWM2} = F_{PWM1})$)*



## *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.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.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.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.AQCTLB.bit.CBU = AQ_SET;               // set actions for EPWM2B
    EPwm2Regs.AQCTLB.bit.CBD = AQ_CLEAR;

// Run Time (Note: Example execution of one run-time instant)
//==========================================================
    EPwm1Regs.CMPA = 400;                            // adjust duty for output EPWM1A
    EPwm1Regs.CMPB = 200;                            // adjust duty for output EPWM1B
    EPwm2Regs.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 x $F_{PWM1}$)*

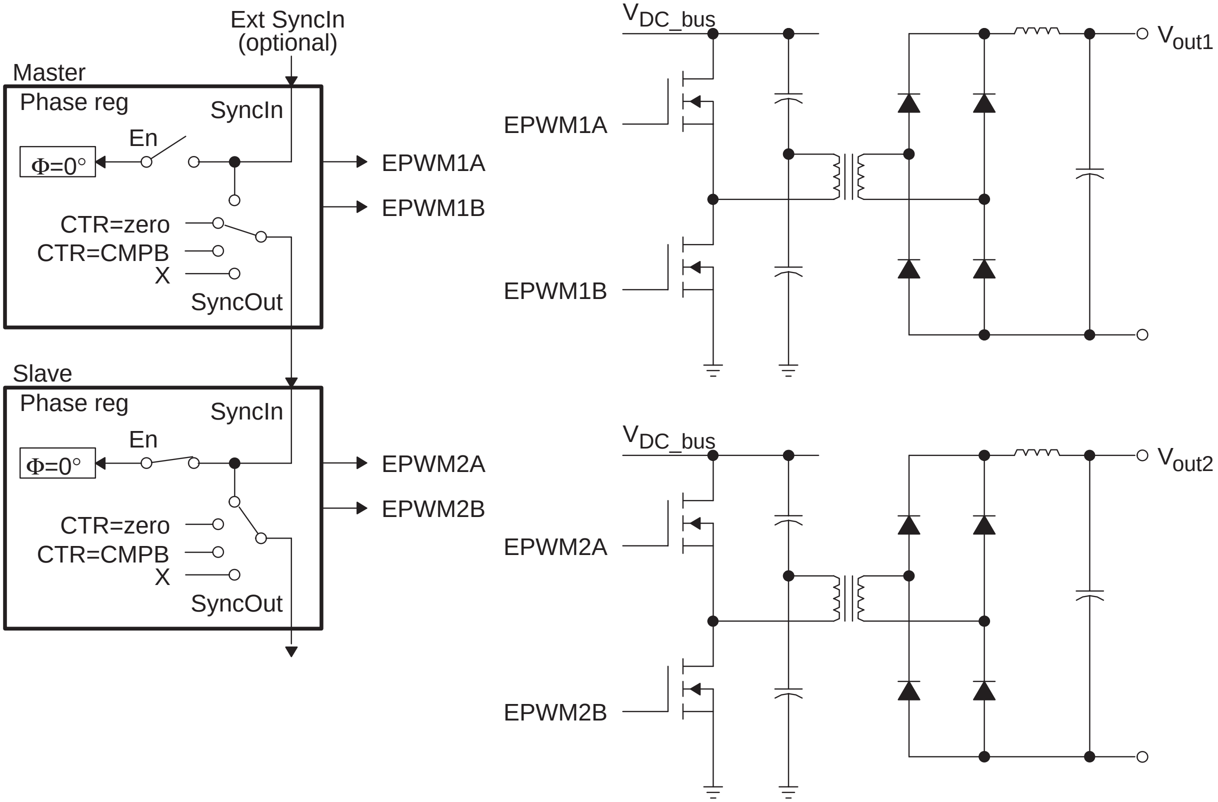
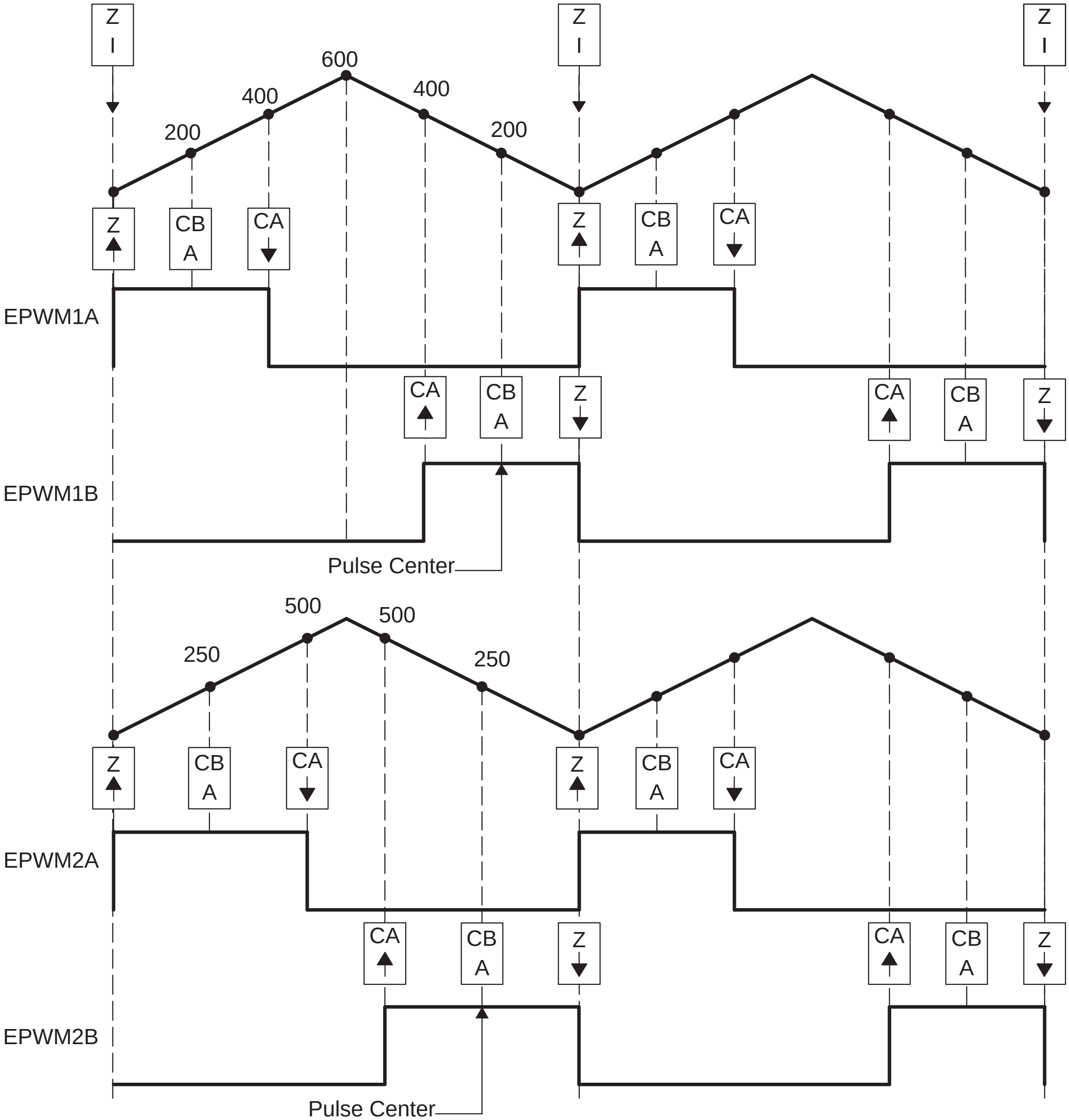


*Figure 3−8. Half-H Bridge Waveforms for Figure 3−7 (Note: Here $F_{PWM2} = F_{PWM1}$ )*



*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.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.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.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 EPWM1A
    EPwm2Regs.AQCTLA.bit.CAU = AQ_CLEAR;
    EPwm2Regs.AQCTLB.bit.ZRO = AQ_CLEAR;             // set actions for EPWM1B
    EPwm2Regs.AQCTLB.bit.CAD = AQ_SET;

// Run Time (Note: Example execution of one run-time instant)
//============================================================
    EPwm1Regs.CMPA = 400;             // adjust duty for output EPWM1A & EPWM1B
    EPwm1Regs.CMPB = 200;             // adjust point-in-time for ADCSOC trigger
    EPwm2Regs.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*

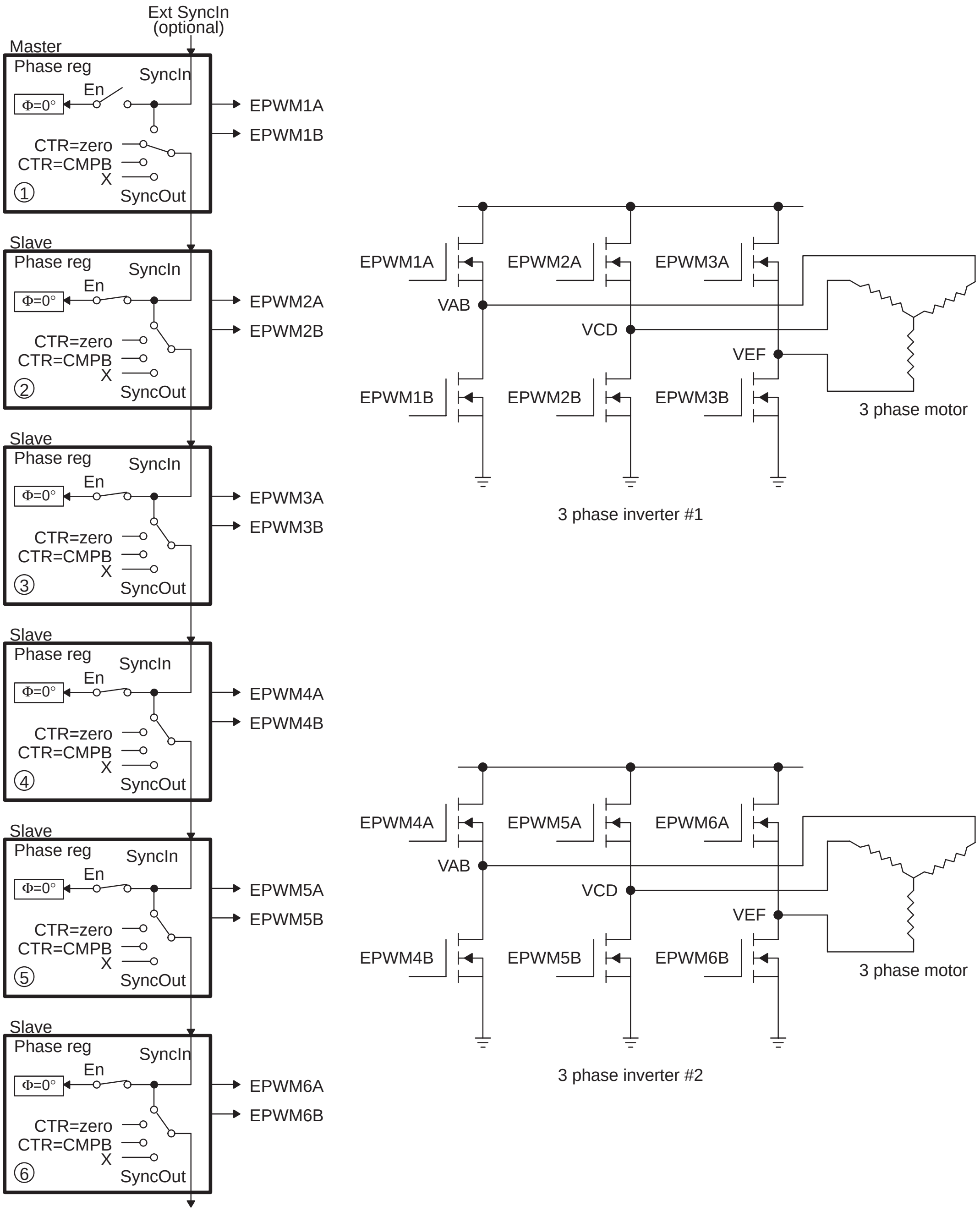
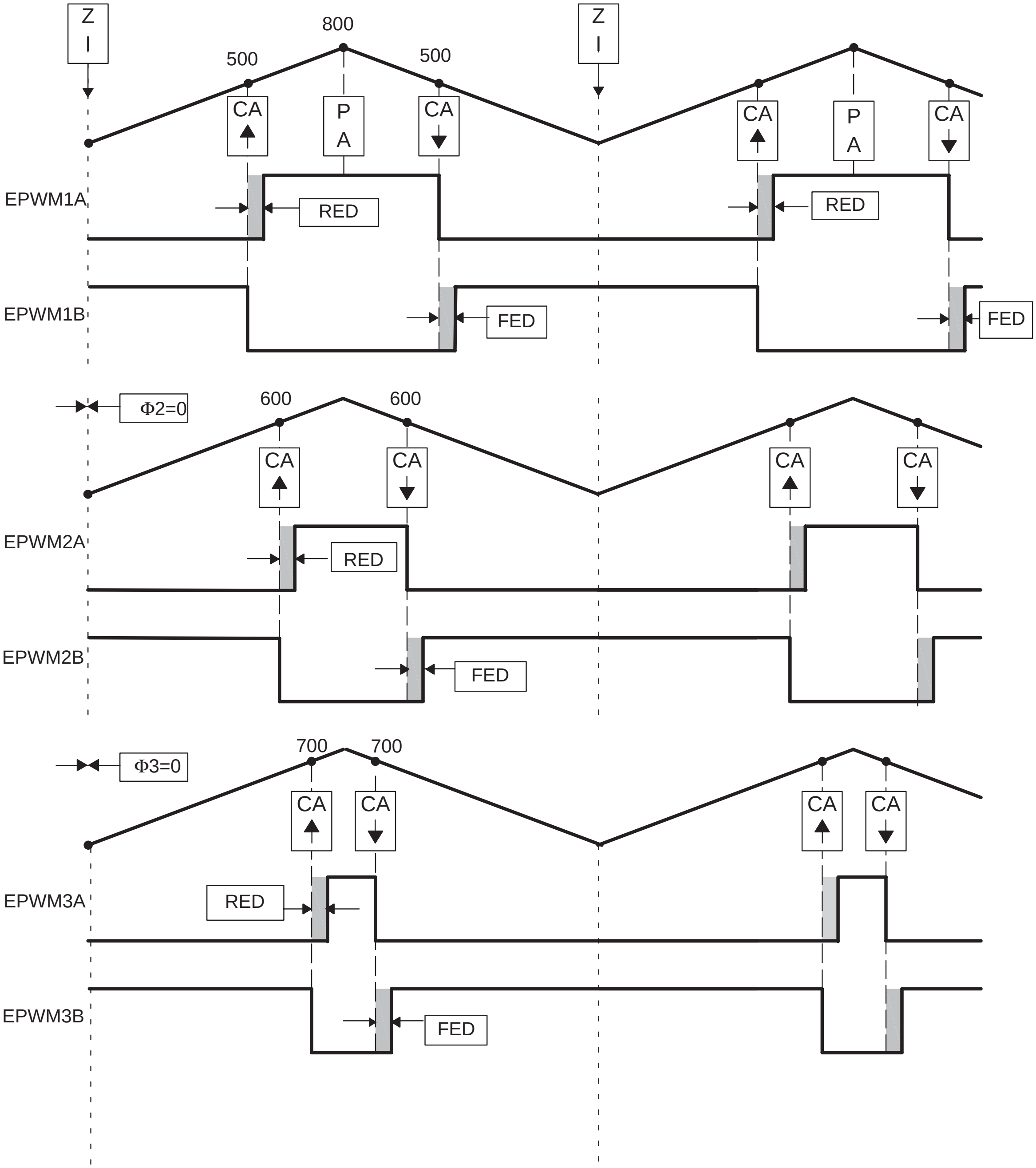


*Figure 3−10. 3-Phase Inverter Waveforms for Figure 3−11 (Note: Only one inverter shown.)*



*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 = 500;                           // adjust duty for output EPWM1A
    EPwm2Regs.CMPA = 600;                           // adjust duty for output EPWM2A
    EPwm3Regs.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 3−12 shows a master and slave module with a phase relationship of 120°, i.e., the slave leads the master.

*Figure 3−12. Configuring Two PWM Modules for Phase Control*

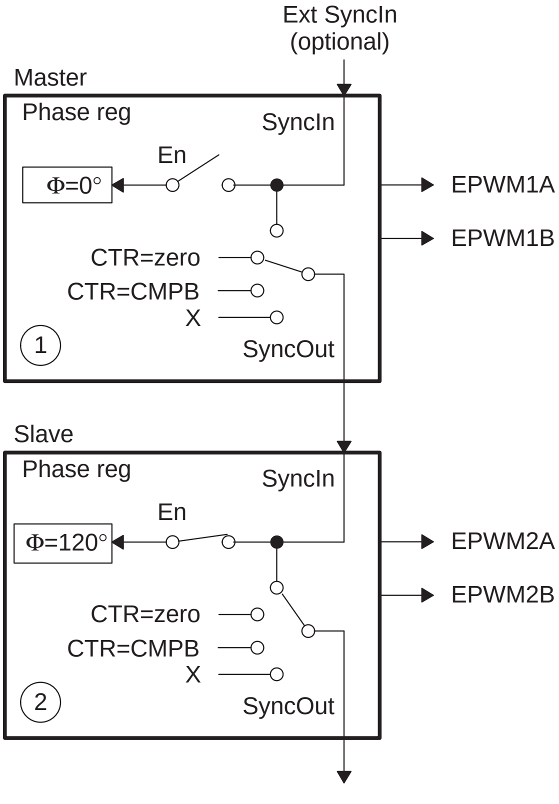
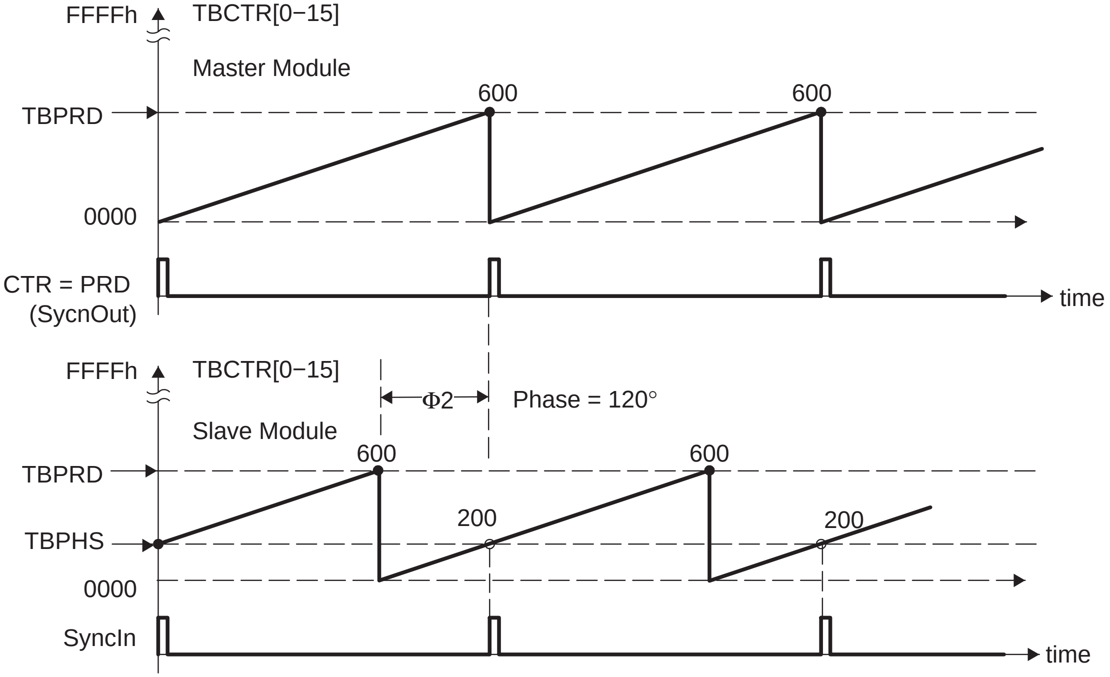


Figure 3−13 shows the associated timing waveforms for this configuration. Here, TBPRD = 600 for both master and slave. For the slave, TBPHS = 200 (i.e., 200/600 × 360° = 120°). Whenever the master generates a SyncIn pulse (CTR = PRD), the value of TBPHS = 200 is loaded into the slave TBCTR register so the slave time-base is always leading the master's time-base by 120° degrees.

*Figure 3−13. 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 3−14. This system uses three PWM modules, with module 1 configured as the master. To work, the phase relationship between adjacent modules must be $\Phi = 120^\circ$. 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 (M−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 3−15 shows the waveforms for the configuration in Figure 3−14.

*Figure 3−14. Control of a 3-Phase Interleaved DC/DC Converter*

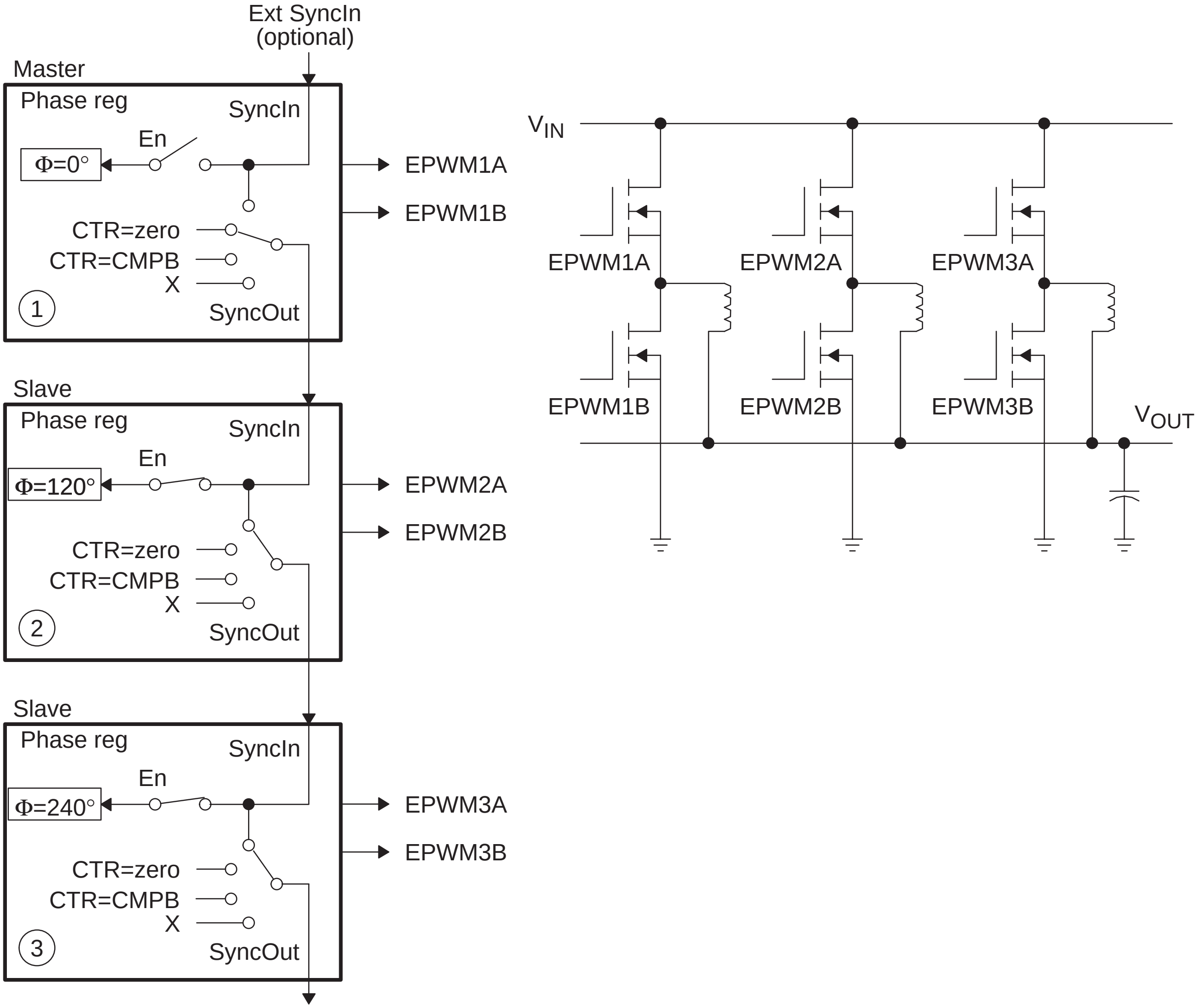
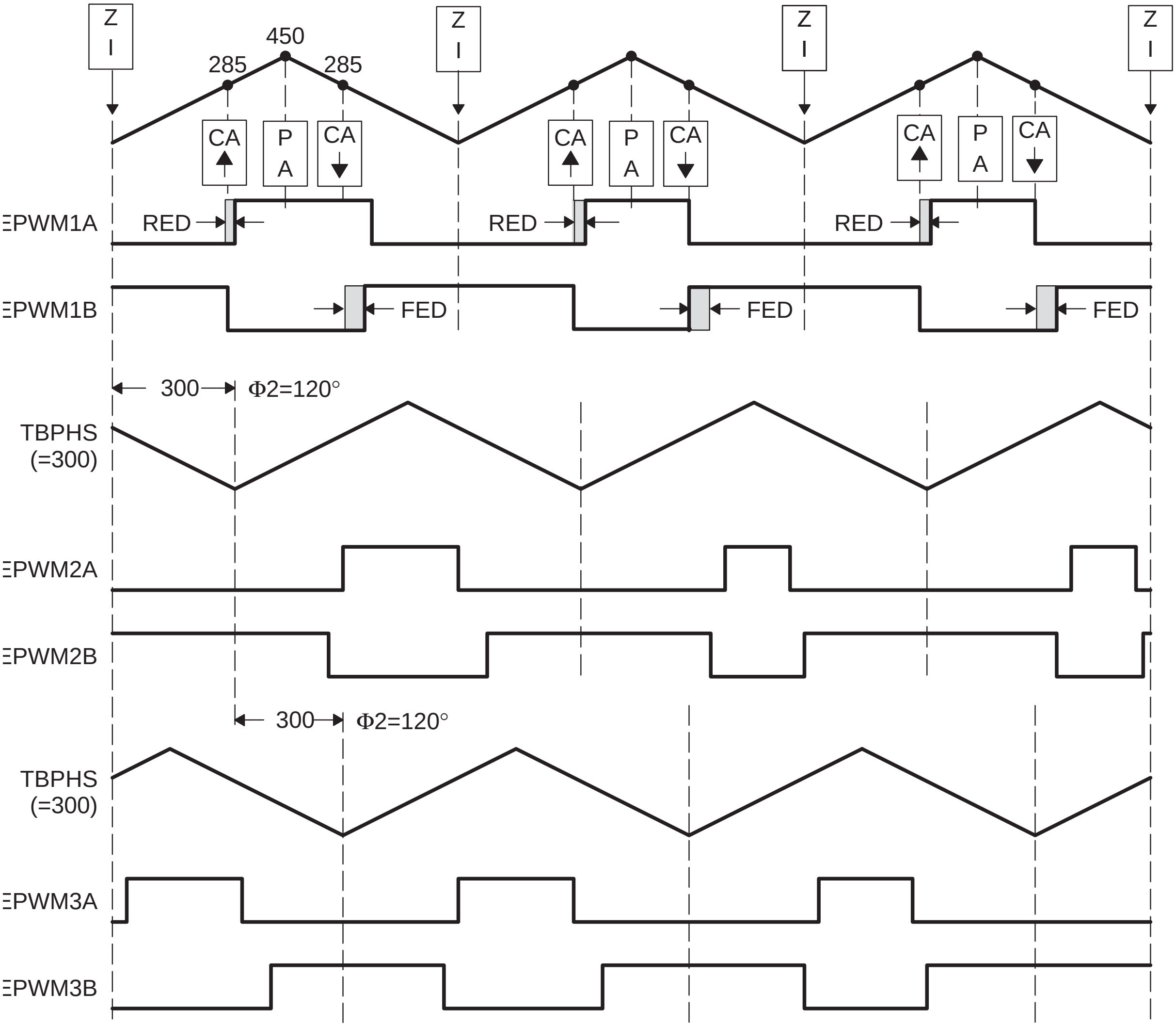


*Figure 3−15. 3-Phase Interleaved DC/DC Converter Waveforms for Figure 3−14*



*Example 3−5. Code Snippet for Configuration in Figure 3−14*

```
//=====================================================================
// 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 EPWM1Ai
    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 EPWM2Ai
    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 = 285;                               // adjust duty for output EPWM1Ai
    EPwm2Regs.CMPA = 285;                               // adjust duty for output EPWM2Ai
    EPwm3Regs.CMPA = 285;                               // adjust duty for output EPWM3Ai
```

## 3.9 Controlling Zero Voltage Switched Full Bridge (ZVSFB) Converter

The example given in Figure 3−16 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 3−17 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 3−16. Controlling a Full-H Bridge Stage ($F_{PWM2}$ = $F_{PWM1}$)*

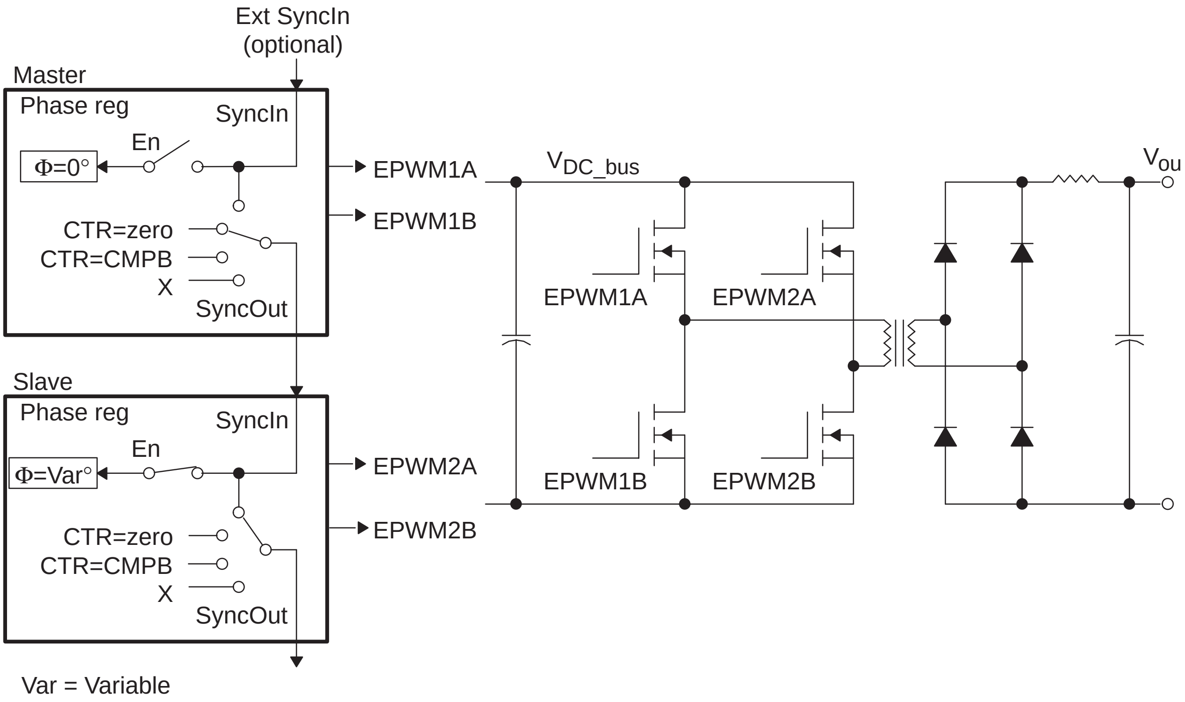
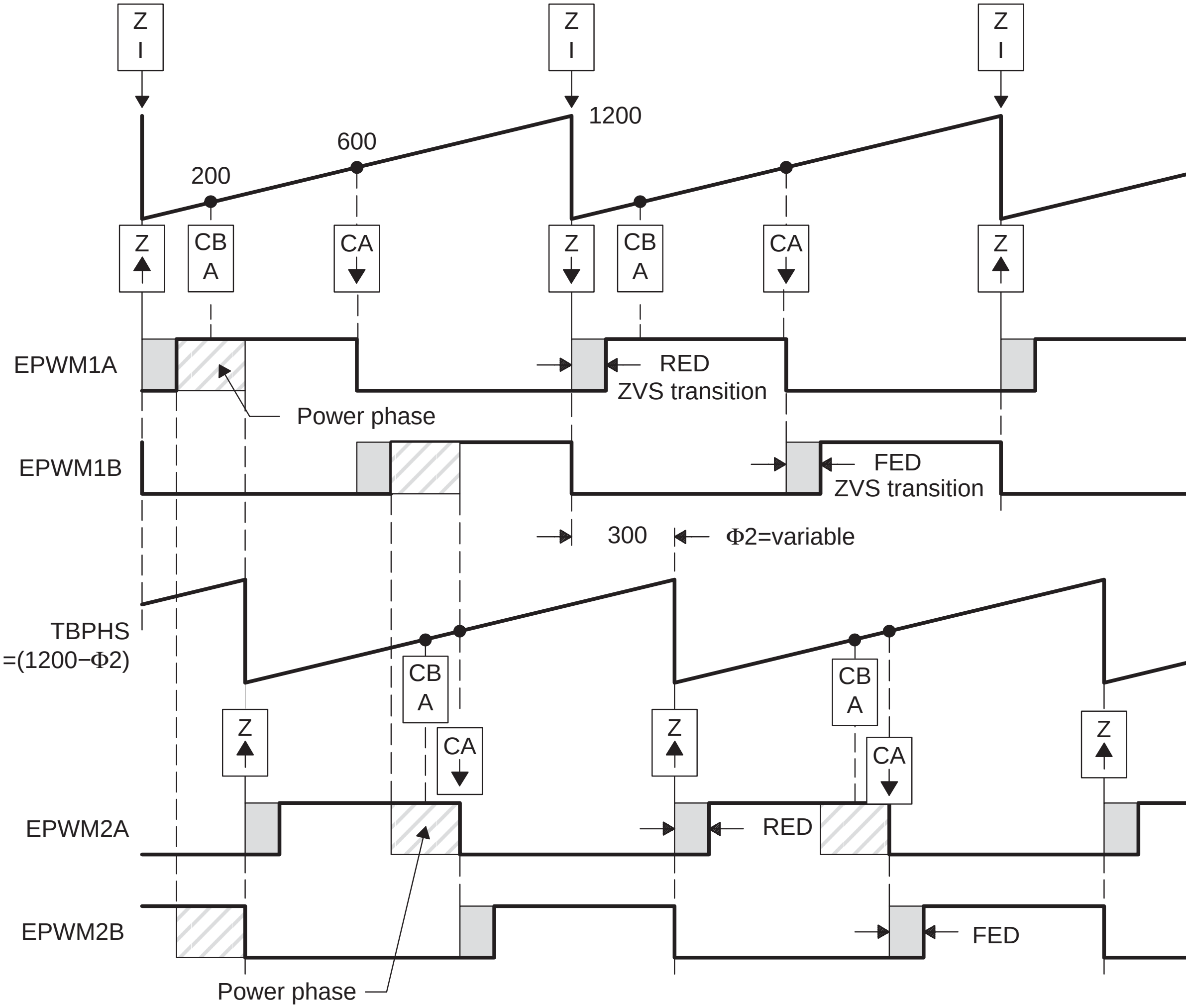


*Figure 3−17. ZVS Full-H Bridge Waveforms*



### Example 3−6. Code Snippet for Configuration in Figure 3−16

```
//=====================================================================
// Config
//=====================================================================
// Initialization Time
//========================
// EPWM Module 1 config
   EPwm1Regs.TBPRD = 1200;                     // Period = 1201 TBCLK counts
   EPwm1Regs.CMPA = 600;                       // Set 50% fixed duty for EPWM1Ai
   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 EPWM1Ai
   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 = 600;                       // Set 50% fixed duty EPWM2Ai
   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 EPWM2Ai
   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
```

## Chapter 4

# Registers

This chapter includes the register layouts and bit description for the sub-modules.

Topic Page

## 4.1 TB Module Registers

Figure 4−1 through Figure 4−5 and Table 4−1 through Table 4−5 provide the TBPRD register definitions.

*Figure 4−1. TB Module Control and Status Register (TBPRD)*

| 15 | 0 |
|---|---|
| TBPRD | |
| R/W | |

**Note:** R—Read, W—Write

*Table 4−1. Time-Base Module Control and Status Register (TBPRD) Field Descriptions*

| Bits | Name | Value | Description |
|---|---|---|---|
| 15–0 | TBPRD | | Period Register Set. These bits set the period value for the counter, i.e., the PWM frequency. This register is shadowed.<br>The active and shadow registers share the same memory map address and name. |

*Figure 4−2. TB Phase Register (TBPHS)*

| 15 | 0 |
|---|---|
| TBPHS | |
| R/W | |

**Note:** R − Read, W − Write

*Table 4−2. TB Phase Register (TBPHS) Field Descriptions*

| Bits | Name | Value | Description |
|---|---|---|---|
| 15–0 | TBPHS | | Phase Offset Register. These bits set the counter phase offset value relative to the time-base that is supplying the sync -in signal. |

*Figure 4−3. TB Counter Register (TBCTR)*

| 15 … 0 |
|---|
| TBCTR |
| R/W |

**Note:** R − Read, W − Write

*Table 4−3. TB Counter Register (TBCTR) Field Descriptions*

| Bits | Name | Value | Description |
|---|---|---|---|
| 15–0 | TBPRD | | Counter Register. Reading these bits gives the current counter value. Writing to these bits sets the current counter value.<br>Writing to these bits updates the current counter value as soon as the write occurs to the register. It is NOT synchronized to the TB clock (TBCLK). |

*Figure 4−4. TB Control Register (TBCTL)*

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

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

**Note:** R − Read, W − Write

*Table 4−4. TB Control Register (TBCTL) Field Descriptions*

| Bits | Name | Description |
|---|---|---|
| 15–14 | FREE,SOFT | Emulation Mode Bits.<br>These bits select the behavior of the ePWM module during emulation events:<br>00 Stop after the next counter increments/decrements<br>01 Stop when counter completes a whole cycle:<br>Up-Mode. Stop when Counter = Period<br>Down-Mode. Stop when Counter = 0x0000<br>Up- and Down-Mode. Stop when Counter = 0x0000<br>1X Free run |

### *Table 4−4. TB Control Register (TBCTL) Field Descriptions (Continued)*

| Bits | Name | Description |
|---|---|---|
| 13 | PHSDIR | Phase Direction Bit.<br>If in up-down mode, PHSDIR indicates what direction the counter counts when a sync event occurs and a new phase value (TBPHS reg) is loaded into the counter (TBCNT reg) irrespective of the current direction of the counter:<br>0 Count down after sync<br>1 Count up after sync<br>In count up or count down mode, this bit is ignored. |
| 12–10 | CLKDIV | TB Clock Prescale Bits<br>These bits select the TB clock prescale value (TBCLK = SYSCLKOUT/(HSPCLKDIV * CLKDIV):<br>000 /1 (default on reset)<br>001 /2<br>010 /4<br>111 /8<br>100 /16<br>101 /32<br>110 /64<br>111 /128 |
| 9–7 | HSPCLKDIV | High Speed Time-base Clock Prescale Bits<br>These bits select the TB clock prescale value (TBCLK = SYSCLKOUT/(HSPCLKDIV * CLKDIV):<br>000 /1<br>001 /2 (default on reset)<br>010 /4<br>111 /6<br>100 /8<br>101 /10<br>110 /12<br>111 /14<br>This divisor emulates the HSPCLK in the F2810/12 system as used on the older Event Manager (EV). |
| 6 | SWFSYNC | Software Forced Sync Pulse<br>0 Writing a 0 has no effect. Read always returns a 0.<br>1 Writing a 1 forces a one-time sync pulse to be generated.<br>This event is ORed with the EPWMxSYNCI input, and hence SWFSYNC is valid (operates) only when EPWMxSYNCI is selected by SYNCOSEL = 00. |

*Table 4−4. TB Control Register (TBCTL) Field Descriptions (Continued)*

| Bits | Name | Description |
|---|---|---|
| 5–4 | SYNCOSEL | Sync Output Select<br>These bits select the source that generate the EPWMxSYNCO signal. The available sources are:<br>00 EPWMxSYNCI<br>01 CTR = ZERO<br>10 CTR = CMPB<br>11 Disable SyncOut |
| 3 | PRDLD | Active Period Reg Load From Shadow Select<br>0 TBPRD is loaded from its shadow register on Counter = 0 event (CTR_zero) A CPU write/read to the TBPRD register accesses the shadow register.<br>1 Load immediately (shadow loading is inhibited)<br>A CPU write/read to the TBPRD register directly accesses the active register. |
| 2 | PHSEN | Counter Reg Load From Phase Reg Enable<br>0 Does NOT allow counter (TBCTR) to be loaded from the phase register (TBPHS)<br>1 Allow counter to be loaded from the phase register (TBPHS) when an EPWMxSYNCI input signal occurs or a S/W forced sync signal, see bit 6. |
| 1–0 | CTRMODE | Counter Mode<br>These bits set the counter mode of operation as follows:<br>These bits are normally configured once and not changed during the normal operation of the module. If you change the mode of the counter, the change will take effect at the next valid TBCLK edge and the current counter value shall increment/decrement from the value before the mode change.<br>00 Count up<br>01 Count down<br>10 Count up and down<br>11 Stop-freeze counter operation (default on reset) |

*Figure 4−5. TB Status Register (TBSTS)*

| 15 – 8 |
|---|
| Reserved |
| R = 0 |

| 7 – 3 | 2 | 1 | 0 |
|---|---|---|---|
| Reserved | CTRMAX | SYNCI | CTRDIR |
| R = 0 | R/W = 1 | R/W = 1 | R |

**Note:** R − Read, W − Write

*Table 4−5. TB Status Register (TBSTS) Field Descriptions*

| Bits | Name | Description |
|---|---|---|
| 15–3 | Reserved | Reserved |
| 2 | CTRMAX | Counter Max Latched Status Bit<br>0 Reading a 0 indicates the counter never reached this value. Writing a 0 will have no effect.<br>1 Reading a 1 on this bit indicates that the counter reached the value 0xFFFF (CNT_max). Writing a 1 to this bit will clear the latched event. |
| 1 | SYNCI | External Input Sync Latched Status Bit<br>0 Writing a 0 will have no effect. Reading a 0 indicates no external sync event has occurred.<br>1 Writing a 1 to this bit will clear the latched event. Reading a 1 on this bit indicates that an external sync event has occurred (EPWMxSYNCI). |
| 0 | CTRDIR | Counter Direction Status Bit<br>0 Counter currently counting down.<br>1 Counter currently counting up.<br>At reset, the counter is disabled (TBCTL[CTRMODE] = 1,1); therefore, the value of this bit has no meaning. This bit will have meaning only after the user selects the appropriate count mode via the TBCTL[CTRMODE] bits. |

## 4.2 CC Module Registers

Figure 4−6 through Figure 4−8 and Table 4−6 through Table 4−8 illustrate the CC module control and status registers.

*Figure 4−6. Compare A Register (CMPA) Field Descriptions*

| 15 | 0 |
|---|---|
| CMPA | |
| R/W | |

**Note:** R − Read, W − Write

*Table 4−6. CC A Register (CMPA) Field Descriptions*

| Bits | Name | Description |
|---|---|---|
| 15–0 | CMPA | Compare A Register Set.<br>This address provides access to the 16 bits of the active/shadow register A.<br>**Note:** The active and shadow registers share the same memory map address and name. |

*Figure 4−7. Compare B Register (CMPB)*

| 15 | 0 |
|---|---|
| CMPB | |
| R/W | |

**Note:** R − Read, W − Write

*Table 4−7. Compare B Register (CMPB) Field Descriptions*

| Bits | Name | Description |
|---|---|---|
| 15–0 | CMPB | Compare B Register Set.<br>This address provides access to the 16 bits of the active/shadow register B.<br>The active and shadow registers share the same memory map address and name. |

*Figure 4−8. Compare Control Register (CMPCTL)*

| 15 | 10 | 9 | 8 |
|---|---|---|---|
| Reserved | | SHDWBFULL | SHDWAFULL |

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| Reserved | SHDWBMODE | Reserved | SHDWAMODE | LOADBMODE | | LOADAMODE | |

**Note:** R − Read, W − Write

*Table 4−8. Compare Control Register (CMPCTL) Field Descriptions*

| Bits | Name | Description |
|---|---|---|
| 15–10 | Reserved | Reserved |
| 9 | SHDWBFULL | Compare B Shadow Registers Full Status Flag<br>This bit self clears once a load-strobe occurs.<br>0 Shadow FIFO not full yet<br>1 Indicates the shadow FIFO is full; a CPU write will overwrite current shadow value. |
| 8 | SHDWAFULL | Compare A Shadow Registers Full Status Flag<br>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.<br>This bit self clears once a load-strobe occurs.<br>0 Indicates shadow FIFO not full yet<br>1 Indicates the shadow FIFO is full, a CPU write will over-write current shadow value. |
| 7 | Reserved | Reserved |
| 6 | SHDWBMODE | Compare B Register Block Operating Mode<br>0 Shadow mode. Operates as a double buffer. All writes via the CPU access the shadow register.<br>1 Immediate mode. Only the active compare register is used. All writes/reads via the CPU directly access the active register for immediate compare action. |
| 5 | Reserved | Reserved |

*Table 4−8. Compare Control Register (CMPCTL) Field Descriptions (Continued)*

| Bits | Name | Description |
|---|---|---|
| 4 | SHDWAMODE | Compare A Register Block Operating Mode<br>0 Shadow mode. Operates as a double buffer. All writes via the CPU access the shadow register.<br>1 Immediate mode. Only the active compare register is used. All writes/reads via the CPU directly access the active register for immediate compare action |
| 3–2 | LOADBMODE | Active Compare B Load From Shadow Select Mode<br>This bit has no effect in immediate mode.<br>00 Load on CTR = Zero<br>01 Load on CTR = PRD<br>10 Load on either CTR = Zero or CTR = PRD<br>11 Freeze (no loads possible) |
| 1–0 | LOADAMODE | Active Compare A Load From Shadow Select Mode. This bit has no effect in Immediate mode.<br>00 Load on CTR = Zero<br>01 Load on CTR = PRD<br>10 Load on either CTR = Zero or CTR = PRD<br>11 Freeze (no loads possible) |

## 4.3 AQ Module Registers

Figure 4−9 through Figure 4−12 and Table 4−9 through Table 4−12 provide the AQ module register definitions.

*Figure 4−9. AQ 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 | |

*Table 4−9. Action-qualifier Output A Control Register (AQCTLA) Field Descriptions*

| Bits | Name | Description | |
|---|---|---|---|
| 15–12 | Reserved | Reserved | |
| 11–10 | CBD | Action when CTR = CMPB on DOWN-count | |
| | | 00 | Do nothing (action disabled) |
| | | 01 | Clear (low) |
| | | 10 | Set (high) |
| | | 11 | Toggle (Low → High, High → Low) |
| 9–8 | CBU | Action when CTR = CMPB on UP-count | |
| | | 00 | Do nothing (action disabled) |
| | | 01 | Clear (low) |
| | | 10 | Set (high) |
| | | 11 | Toggle (Low → High, High → Low) |
| 7–6 | CAD | Action when CTR = CMPA on DOWN-count | |
| | | 00 | Do nothing (action disabled) |
| | | 01 | Clear (low) |
| | | 10 | Set (high) |
| | | 11 | Toggle (Low → High, High → Low) |

*Table 4−9. Action-qualifier Output A Control Register (AQCTLA) Field Descriptions (Continued)*

| Bits | Name | Description |
|---|---|---|
| 5–4 | CAU | Action when CTR = CMPA on UP-count |
| | | 00 Do nothing (action disabled) |
| | | 01 Clear (low) |
| | | 10 Set (high) |
| | | 11 Toggle (Low → High, High → Low) |
| 3–2 | PRD | Action when CTR = PRD |
| | | 00 Do nothing (action disabled) |
| | | 01 Clear (low) |
| | | 10 Set (high) |
| | | 11 Toggle (Low →High, High → Low)<br>Note: By definition, in up/down count mode when CNT = Period, Direction = 0, i.e., count DOWN |
| 1–0 | ZRO | Action when CTR = Zero |
| | | 00 Disabled—does nothing |
| | | 01 Clear (low) |
| | | 10 Set (high) |
| | | 11 Toggle (Low → High, High → Low)<br>Note: By definition, in Up/Down count mode when CNT = 0, Direction = 1 i.e. count UP |

*Figure 4−10. AQ Output B Control Register (AQCTLB)*

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

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

**Note:** R − Read, W − Write

*Table 4−10. Action-qualifier Output B Control Register (AQCTLB) Field Descriptions*

| Bits | Name | Description | |
|---|---|---|---|
| 15–12 | reserved | | |
| 11–10 | CBD | Action when CTR = CMPB on DOWN-count | |
| | | 00 | Do nothing (action disabled) |
| | | 01 | Clear (low) |
| | | 10 | Set (high) |
| | | 11 | Toggle (Low → High, High → Low) |
| 9–8 | CBU | Action when CTR = CMPB on UP-count | |
| | | 00 | Do nothing (action disabled) |
| | | 01 | Clear (low) |
| | | 10 | Set (high) |
| | | 11 | Toggle (Low → High, High → Low) |
| 7–6 | CAD | Action when CTR = CMPA on DOWN-count | |
| | | 00 | Do nothing (action disabled) |
| | | 01 | Clear (low) |
| | | 10 | Set (high) |
| | | 11 | Toggle (Low → High, High → Low) |

*Table 4−10. Action-qualifier Output B Control Register (AQCTLB) Field Descriptions (Continued)*

| Bits | Name | Description |
|---|---|---|
| 5–4 | CAU | Action when CTR = CMPA on UP-count<br>00 Do nothing (action disabled)<br>01 Clear (low)<br>10 Set (high)<br>11 Toggle (Low → High, High → Low) |
| 3–2 | PRD | Action when CTR = PRD. By definition, in Up/Down count mode when CNT = Period, Direction = 0, (i.e., count DOWN)<br>00 Do nothing (action disabled)<br>01 Clear (low)<br>10 Set (high)<br>11 Toggle (Low → High, High → Low) |
| 1–0 | ZRO | Action when CTR = Zero. By definition, in Up/Down count mode when CNT = 0, Direction = 1, ( i.e., count UP)<br>00 Do nothing (action disabled)<br>01 Clear (low)<br>10 Set (high)<br>11 Toggle (Low →High, High →Low) |

*Figure 4−11.AQ S/W Force Register (AQSFRC)*

| 15 | 8 |
|---|---|
| Reserved | |
| R = 0 | |

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| RLDCSF | | OTSFB | ACTSFB | | OTSFA | ACTSFA | |
| R/W | | R = 0/W = 1 | R/W | | R = 0/W = 1 | R/W | |

**Note:** R − Read, W − Write

*Table 4−11. Action-qualifier S/W Force Register (AQSFRC) Field Descriptions*

| Bits | Name | Description | |
|---|---|---|---|
| 15–8 | Reserved | Reserved | |
| 7–6 | RLDCSF | AQCSF Active Register Reload From Shadow Options | |
| | | 00 | Load on event CTR = Zero |
| | | 01 | Load on event CTR = PRD |
| | | 10 | Load on event CTR = Zero or CTR = PRD |
| | | 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.<br>**Notes:** 1) This bit is auto cleared once a write to this register is complete, i.e., a forced event is initiated.<br>2) 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) |

*Table 4−11. Action-qualifier S/W Force Register (AQSFRC) Field Descriptions (Continued)*

| Bits | Name | Description |
|---|---|---|
| 2 | OTSFA | One-Time Software Forced Event on Output A |
| | | 0 Writing a 0 (zero) has no effect. Always reads back a 0. |
| | | 1 Initiates a single s/w forced event. |
| | | **Notes:** 1) This bit is auto cleared once a write to this register is complete, i.e., a forced event is initiated.<br>2) This is a one-shot forced event. It can be overridden by another subsequent event on output A. |
| 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. AQ Continuous S/W Force Register (AQCSFRC)*

| 15 | 8 |
|---|---|
| Reserved | |
| R = 0 | |

| 7 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|
| Reserved | | CSFB | | CSFA | |
| R = 0 | | R/W | | R/W | |

**Note:** R − Read, W − Write

*Table 4−12. Action-qualifier Continuous S/W Force Register (AQCSFRC) Field Descriptions*

| Bits | Name | Description |
|---|---|---|
| 15–4 | Reserved | Reserved |
| 3–2 | CSFB | Continuous Software Force on Output B<br>In immediate mode, a continuous force takes effect on the next TBCLK edge.<br>In shadow mode, a continuous force takes effect on the next TBCLK edge after a shadow load into the active register.<br>00 Forcing disabled, i.e., has no effect<br>01 Forces a continuous low on output B<br>10 Forces a continuous high on output B<br>11 Forcing disabled, i.e., has no effect |
| 1–0 | CSFA | Continuous Software Force on Output A<br>In immediate mode, a continuous force takes effect on the next TBCLK edge.<br>In shadow mode, a continuous force takes effect on the next TBCLK edge after a shadow load into the active register.<br>00 Forcing disabled, i.e., has no effect<br>01 Forces a continuous low on output A<br>10 Forces a continuous high on output A<br>11 Forcing disabled, i.e., has no effect |

## 4.4 DB Module Registers

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

*Figure 4−13. DB Generator Control Register (DBCTL)*

| 15 … 8 |
|---|
| Reserved |
| R = 0 |

| 7 … 4 | 3 … 2 | 1 … 0 |
|---|---|---|
| Reserved | POLSEL | MODE |
| R = 0 | R/W | R/W |

**Note:** R − Read, W − Write

*Table 4−13. DB Generator Control Register (DBCTL) Field Descriptions*

| Bits | Name | Description |
|---|---|---|
| 15–4 | Reserved | Reserved |
| 3–2 | POLSEL | Polarity Select Control.<br>The following modes are valid only when MODE bits = 11<br>00 Active high (AH) mode<br>01 Active low complementary (ALC) mode<br>10 Active high complementary (AHC) mode<br>11 Active low (AL) mode<br>Note: Other modes, MODE = 01 and 10 are also possible, but are not regarded as typical usage modes. However, you may find these useful in some applications. |
| 1–0 | MODE | Dead Band Mode Control<br>00 DBM is fully disabled or by-passed. In this mode, the POLSEL bits have no effect.<br>01 DBA = AQA (delay is by-passed for A)<br>DBB = FED (Falling Edge Delay version of A)<br>10 DBA = RED (Rising Edge Delay version of A)<br>DBB = AQB (delay is by-passed for B)<br>11 DBM is fully enabled (i.e. both RED and FED active) |

*Figure 4−14. DB Generator Rising Edge Delay Register (DBRED)*

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

| 7 0 |
|---|
| DEL |
| R/W |

**Note:** R − Read, W − Write

*Table 4−14. DB Generator Rising Edge Delay Register (DBRED) Field Descriptions*

| Bits | Name | Description |
|---|---|---|
| 15–4 | Reserved | Reserved |
| 9–0 | DEL | Rising Edge Delay Count. 10-bit counter. |

*Figure 4−15. DB Generator Falling Edge Delay Register (DBFED)*

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

| 7 0 |
|---|
| DEL |
| R/W |

**Note:** R − Read, W − Write

*Table 4−15. DB Generator Falling Edge Delay Register (DBFED) Field Descriptions*

| Bits | Name | Description |
|---|---|---|
| 15–4 | Reserved | Reserved |
| 9–0 | DEL | Falling Edge Delay Count. 10-bit counter. |

## 4.5 PC Module Control Register

Figure 4−16 and Table 4−16 provide the definitions for the PC module control register.

*Figure 4−16. PC Control Register (PCCTL)*

|  |  |
|---|---|
| 15 11 | 10 8 |
| Reserved | CHPDUTY |
| R = 0 | R/W |

|  |  |  |
|---|---|---|
| 7 5 | 4 3 2 1 | 0 |
| CHPFREQ | OSHTWTH | CHPEN |
| R/W | R/W | R/W |

*Table 4−16. PC Control Register (PCCTL) Bit Descriptions*

| Bits | Name | Description |
|---|---|---|
| 15–11 | Reserved | Reserved |
| 10–8 | CHPDUTY | Chopping Clock Duty Cycle<br>000 Duty = 1/8 (12.5%)<br>001 Duty = 2/8 (25.0%)<br>010 Duty = 3/8 (37.5%)<br>011 Duty = 4/8 (50.0%)<br>100 Duty = 5/8 (62.5%)<br>101 Duty = 6/8 (75.0%)<br>110 Duty = 7/8 (87.5%)<br>111 Reserved |
| 7:5 | CHPFREQ | Chopping Clock Frequency<br>000 Divide by 1 (i.e. no prescale, = 12.5 MHz with 100 MHz sysclk)<br>001 Divide by 2 (6.25 MHz )<br>010 Divide by 3 (4.16 MHz )<br>011 Divide by 4 (3.12 MHz )<br>100 Divide by 5 (2.50 MHz )<br>101 Divide by 6 (2.08 MHz )<br>110 Divide by 7 (1.78 MHz)<br>111 Divide by 8 (1.56 MHz) |

*Table 4−16. PC Control Register (PCCTL) Bit Descriptions (Continued)*

| Bits | Name | Description | |
|---|---|---|---|
| 4:1 | OSHTWTH | | One-Shot Pulse Width |
| | | 0000 | 1 x sysclk/8 wide ( = 80 nS @ 100 MHz) |
| | | 0001 | 2 x sysclk/8 wide ( = 160 nS @ 100 MHz) |
| | | 0010 | 3 x sysclk/8 wide ( = 240 nS @ 100 MHz) |
| | | 0011 | 4 x sysclk/8 wide ( = 320 nS @ 100 MHz) |
| | | 0100 | 5 x sysclk/8 wide ( = 400 nS @ 100 MHz) |
| | | 0101 | 6 x sysclk/8 wide ( = 480 nS @ 100 MHz) |
| | | 0110 | 7 x sysclk/8 wide ( = 560 nS @ 100 MHz) |
| | | 0111 | 8 x sysclk/8 wide ( = 640 nS @ 100 MHz) |
| | | 1000 | 9 x sysclk/8 wide ( = 720 nS @ 100 MHz) |
| | | 1001 | 10 x sysclk/8 wide ( = 800 nS @ 100 MHz) |
| | | 1010 | 11 x sysclk/8 wide ( = 880 nS @ 100 MHz) |
| | | 1011 | 12 x sysclk/8 wide ( = 960 nS @ 100 MHz) |
| | | 1100 | 13 x sysclk/8 wide ( = 1040 nS @ 100 MHz) |
| | | 1101 | 14 x sysclk/8 wide ( = 1120 nS @ 100 MHz) |
| | | 1110 | 15 x sysclk/8 wide ( = 1200 nS @ 100 MHz) |
| | | 1111 | 16 x sysclk/8 wide ( = 1280 nS @ 100 MHz) |
| 0 | CHPEN | PWM-chopping Enable | |
| | | 0 | Disable (by-pass) chopping function |
| | | 1 | Enable chopping function |

## 4.6 TZ Module Control and Status Registers

Figure 4−17 and Table 4−17 provide the TZ control and status register definitions.

*Figure 4−17. TZ Select Register (TZSEL)*

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

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| Reserved | | CBC6 | CBC5 | CBC4 | CBC3 | CBC2 | CBC1 |
| R = 0 | | R/W | R/W | R/W | R/W | R/W | R/W |

**Note:** R − Read, W − Write

*Table 4−17. TZ Module Select Register (TZSEL) Field Descriptions*

| Bits | Name | Description |
|---|---|---|
| **Period-by-Period (CBC) Trip-zone Mapping of Outputs EPWMxA and B** | | |
| 15:14 | Reserved | Reserved |
| 13 | OSHT6 | Trip-zone 6 ($\overline{\text{TZ6}}$) Select |
| | | 0 Disable $\overline{\text{TZ6}}$ as a one-shot trip source |
| | | 1 Enable $\overline{\text{TZ6}}$ as a one-shot trip source |
| 12 | OSHT5 | Trip-zone 5 ($\overline{\text{TZ5}}$) Select |
| | | 0 Disable $\overline{\text{TZ5}}$ as a one-shot trip source |
| | | 1 Enable $\overline{\text{TZ5}}$ as a one-shot trip source |
| 11 | OSHT4 | Trip-zone 4 ($\overline{\text{TZ4}}$) Select |
| | | 0 Disable $\overline{\text{TZ4}}$ as a one-shot trip source |
| | | 1 Enable $\overline{\text{TZ4}}$ as a one-shot trip source |
| 10 | OSHT3 | Trip-zone 3 ($\overline{\text{TZ3}}$) Select |
| | | 0 Disable $\overline{\text{TZ3}}$ as a one-shot trip source |
| | | 1 Enable $\overline{\text{TZ3}}$ as a one-shot trip source |
| 9 | OSHT2 | Trip-zone 2 ($\overline{\text{TZ2}}$) Select |
| | | 0 Disable $\overline{\text{TZ2}}$ as a one-shot trip source |
| | | 1 Enable $\overline{\text{TZ2}}$ as a one-shot trip source |

*Table 4−17. TZ Module Select Register (TZSEL) Field Descriptions (Continued)*

| Bits | Name | Description |
|---|---|---|
| 8 | OSHT1 | Trip-zone 1 ($\overline{\text{TZ1}}$) Select |
| | | 0 Disable $\overline{\text{TZ1}}$ as a one-shot trip source |
| | | 1 Enable $\overline{\text{TZ1}}$ as a one-shot trip source |
| **One-Shot (OSHT) Trip-zone Mapping of Outputs EPWMxA and B** | | |
| 7:6 | Reserved | Reserved |
| 5 | CBC6 | Trip-zone 6 ($\overline{\text{TZ6}}$) Select |
| | | 0 Disable $\overline{\text{TZ6}}$ as a CBC trip source |
| | | 1 Enable $\overline{\text{TZ6}}$ as a CBC trip source |
| 4 | CBC5 | Trip-zone 5 ($\overline{\text{TZ5}}$) Select |
| | | 0 Disable $\overline{\text{TZ5}}$ as a CBC trip source |
| | | 1 Enable $\overline{\text{TZ5}}$ as a CBC trip source |
| 3 | CBC4 | Trip-zone 4 ($\overline{\text{TZ4}}$) Select |
| | | 0 Disable $\overline{\text{TZ4}}$ as a CBC trip source |
| | | 1 Enable $\overline{\text{TZ4}}$ as a CBC trip source |
| 2 | CBC3 | Trip-zone 3 ($\overline{\text{TZ3}}$) Select |
| | | 0 Disable $\overline{\text{TZ3}}$ as a CBC trip source |
| | | 1 Enable $\overline{\text{TZ3}}$ as a CBC trip source |
| 1 | CBC2 | Trip-zone 2 ($\overline{\text{TZ2}}$) Select |
| | | 0 Disable $\overline{\text{TZ2}}$ as a CBC trip source |
| | | 1 Enable $\overline{\text{TZ2}}$ as a CBC trip source |
| 0 | CBC1 | Trip-zone 1 ($\overline{\text{TZ1}}$) Select |
| | | 0 Disable $\overline{\text{TZ1}}$ as a CBC trip source |
| | | 1 Enable $\overline{\text{TZ1}}$ as a CBC trip source |

*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 | | R/W | |

**Note:** R − Read, W − Write

*Table 4−18. TZ Control Register (TZCTL) Field Descriptions*

| Bits | Name | Description | |
|---|---|---|---|
| 15–4 | Reserved | Reserved | |
| 3–2 | TZB | $\overline{\text{TZ1}}$ to $\overline{\text{TZ6}}$ Action On EPWMxB | |
| | | 00 | High impedance (EPWMxB = High-impedance state) |
| | | 01 | Forced Hi (EPWMxB = High state) |
| | | 10 | Forced Lo (EPWMxB = Lo state) |
| | | 11 | Do Nothing, trip action is disabled |
| 1–0 | TZA | $\overline{\text{TZ1}}$ to $\overline{\text{TZ6}}$ Trip Action On EPWMxA | |
| | | 00 | High impedance (EPWMxA = High-impedance state) |
| | | 01 | Forced Hi (EPWMxA = High state) |
| | | 10 | Forced Lo (EPWMxA = Lo state) |
| | | 11 | Do Nothing, trip action is disabled |

## Figure 4−19. TZ Enable Interrupt Register (TZEINT)

| 15–8 |
|---|
| Reserved |
| R = 0 |

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

**Note:** R − Read, W − Write

## Table 4−19. Trip-zone Enable Interrupt Register (TZEINT) Field Descriptions

| Bits | Name | Description |
|---|---|---|
| 15–3 | Reserved | Reserved |
| 2 | OST | Trip-zones One Shot Interrupt Enable<br>0 Disabled<br>1 Enabled |
| 1 | CBC | Trip-zones Cycle by Cycle Interrupt Enable<br>0 Disabled<br>1 Enabled |
| 0 | Reserved | Reserved |

*Figure 4−20. TZ Flag Register (TZFLG)*

| 15–8 |
|---|
| Reserved |
| R = 0 |

| 7–3 | 2 | 1 | 0 |
|---|---|---|---|
| Reserved | OST | CBC | INT |
| R = 0 | R | R | R |

**Note:** R − Read, W − Write

*Table 4−20. TZ Flag Register (TZFLG) Field Descriptions*

| Bits | Name | Description | |
|---|---|---|---|
| 15–3 | Reserved | Reserved | |
| 2 | OST | Latched Status Flag for One-Shot Trip (OST) Latch | |
| | | 0 | No trip has occurred on OST |
| | | 1 | Indicates a trip has occurred on the OST selected zone |
| 1 | CBC | Latched Status Flag for Cycle-By-Cycle (CBC) Trip Latch | |
| | | 0 | No trip has occurred on CBC |
| | | 1 | A trip has occurred on the CBC selected zone |
| 0 | INT | Latched Global Interrupt Status Flag | |
| | | 0 | Indicates no interrupt was generated. |
| | | 1 | Indicates an interrupt was generated on $\overline{\text{EPWMxTZINT}}$ because of a trip condition.<br>No further interrupts are generated on $\overline{\text{EPWMxTZINT}}$ until this flag is cleared. |

*Figure 4−21. TZ Clear Register (TZCLR)*

| 15 – 8 |
|---|
| Reserved |
| R = 0 |

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

**Note:** R − Read, W − Write

*Table 4−21. TZ Clear Register (TZCLR) Field Descriptions*

| Bits | Name | 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 this Trip (set) condition.<br>No further interrupts will be generated on $\overline{\text{EPWMxTZINT}}$ until INT flag is cleared. |

**Note:** If the interrupt flag (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. TZ Force Register (TZFRC)*

| 15 | 8 |
|---|---|
| Reserved | |
| R = 0 | |

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

**Note:** R − Read, W − Write

*Table 4−22. TZ Force Register (TZFRC) Field Descriptions*

| Bits | Name | Description | |
|---|---|---|---|
| 15–3 | Reserved | Reserved | |
| 2 | OST | Force a Fault on OST Trip Latch. | |
| | | 0 | Writing of 0 is ignored. Always reads back a 0. |
| | | 1 | Forces a fault on the OST latch and sets the OSTFLG bit. |
| 1 | CBC | Force a Fault on CBC Trip Latch | |
| | | 0 | Writing of 0 is ignored. Always reads back a 0. |
| | | 1 | Forces a fault on the CBC latch and sets the CBCFLG bit. |
| 0 | Reserved | Reserved | |

## 4.7 ET Module Registers

Figure 4−23 through Figure 4−27 and Table 4−23 through Table 4−27 describe the registers for the TB module.

*Figure 4−23. ET Selection Register (ETSEL)*

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

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

**Note:** R − Read, W − Write

*Table 4−23. ET Selection Register (ETSEL) Field Descriptions*

| Bits | Name | Value | Description |
|---|---|---|---|
| 15 | SOCBEN | Enable SOCB Pulse | |
| | | 0 | Does not enable SOCB pulse |
| | | 1 | Enables SOCB Pulse |
| 14−12 | SOCBSEL | EPWMxSOCB Selection Options | |
| | | 000 | Spare |
| | | 001 | Enable CNTR = zero event |
| | | 010 | Enable CTR = PRD event |
| | | 011 | Spare |
| | | 100 | Enable CTRU = CMPA event |
| | | 101 | Enable CTRD = CMPA event |
| | | 110 | Enable CTRU = CMPB event |
| | | 111 | Enable CTRD = CMPB event |
| 11 | SOCAEN | Enable SOCA Pulse | |
| | | 0 | Does not enable SOCA pulse |
| | | 1 | Enables SOCA pulse |

*Table 4−23. ET Selection Register (ETSEL) Field Descriptions (Continued)*

| Bits | Name | Value | Description |
|---|---|---|---|
| 10−8 | SOCASEL | | EPWMxSOCA Selection Options |
| | | 000 | Spare |
| | | 001 | Enable CNTR = zero event |
| | | 010 | Enable CTR = PRD event |
| | | 011 | Spare |
| | | 100 | Enable CTRU = CMPA event |
| | | 101 | Enable CTRD = CMPA event |
| | | 110 | Enable CTRU = CMPB event |
| | | 111 | Enable CTRD = CMPB event |
| 7−4 | Reserved | | Reserved |
| 3 | INTEN | | Enable EPWMxINT Generation |
| | | 0 | Does not enable EPWMxINT generation |
| | | 1 | Enables EPWMxINT generation |
| 2−0 | INTSEL | | $\overline{\text{EPWMxINT}}$ Selection Options |
| | | 000 | Spare |
| | | 001 | Enable CNTR = zero event |
| | | 010 | Enable CTR = PRD event |
| | | 011 | Spare |
| | | 100 | Enable CTRU = CMPA event |
| | | 101 | Enable CTRD = CMPA event |
| | | 110 | Enable CTRU = CMPB event |
| | | 111 | Enable CTRD = CMPB event |

*Figure 4−24. ET Prescale Register (ETPS)*

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

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

**Note:** R − Read, W − Write

*Table 4−24. ET Prescale Register (ETPS) Field Descriptions*

| Bits | Name | Description |
|---|---|---|
| 15−14 | SOCBCNT | EPWMxSOCB Counter Register<br>These bits indicate how many selected events have occurred:<br>00 No events<br>01 2 events<br>10 3 events<br>11 |
| 13−12 | SOCBPRD | EPWMxSOCB Period Select<br>These bits select how many selected event need to occur before an SOCB pulse is generated:<br>00 Disable counter<br>01 Generate pulse on SOCBCNT = 01 (first event)<br>10 Generate pulse on SOCBCNT = 10 (second event)<br>11 Generate pulse on SOCBCNT = 11 (third event) |
| 11−10 | SOCACNT | EPWMxSOCA Counter Register<br>These bits indicate how many selected events have occurred:<br>00 No events<br>01 1 event<br>10 2 events<br>11 3 events |
| 9−8 | SOCAPRD | EPWMxSOCA Period Select<br>These bits select how many selected event need to occur before an SOCA pulse is generated:<br>00 Disable counter<br>01 Generate pulse on SOCACNT = 01 (first event)<br>10 Generate pulse on SOCACNT = 10 (second event)<br>11 Generate pulse on SOCACNT = 11 (third event) |
| 7−4 | Reserved | Reserved |

*Table 4−24. ET Prescale Register (ETPS) Field Descriptions (Continued)*

| Bits | Name | Description |
|---|---|---|
| 3−2 | INTCNT | $\overline{\text{EPWMxINT}}$ Counter Register<br>These bits indicate how many selected events have occurred:<br>00 No events<br>01 1 event<br>10 2 events<br>11 3 events |
| 1−0 | INTPRD | $\overline{\text{EPWMxINT}}$ Period Select<br>These bits select how many selected events need to occur before an interrupt is generated:<br>00 Disable counter<br>01 Generate interrupt on INTCNT = 01 (first event)<br>10 Generate interrupt on INTCNT = 10 (second event)<br>11 Generate interrupt on INTCNT = 11 (third event) |

*Figure 4−25. ET Flag Register (ETFLG)*

| 15 | 8 |
|---|---|
| Reserved | |
| R = 0 | |

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

**Note:** R − Read

*Table 4−25. ET Flag Register (ETFLG) Field Descriptions*

| Bits | Name | Value | Description |
|---|---|---|---|
| 15−4 | Reserved | | Reserved |
| 3 | SOCB | | Latched SOCB Flag Bit Status<br>Unlike the INT flag bit, the EPWMxSOCB 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 EPWMxSOCB |

| Bits | Name | Value | Description |
|---|---|---|---|
| 2 | SOCA | | Latched SOCA Flag Bit Status<br>Unlike the INT flag bit, 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 |
| 1 | Reserved | | Reserved |
| 0 | INT | | Latched INT Flag Bit Status |
| | | 0 | Indicates no event occurred |
| | | 1 | Indicates that an interrupt pulse event was generated on $\overline{\text{EPWMxINT}}$. No further interrupt pulses will be generated until the flag bit is cleared. |

*Figure 4−26. ET Clear Register (ETCLR)*

| 15–8 |
|---|
| Reserved |
| R = 0 |

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

**Note:** R − Read, W − Write

*Table 4−26. ET Clear Register (ETCLR) Field Descriptions*

| Bits | Name | Value | Description |
|---|---|---|---|
| 15−4 | Reserved | | Reserved |
| 3 | SOCB | | SOCB Flag Clear Bit |
| | | 0 | Has no effect. Always reads back a 0 |
| | | 1 | Clears the SOCB flag bit |
| 2 | SOCA | | SOCA Flag Clear Bit |
| | | 0 | Has no effect. Always reads back a 0 |
| | | 1 | Clears the SOCA flag bit |
| 1 | Reserved | | Reserved |

| Bits | Name | Value | Description |
|---|---|---|---|
| 0 | INT | | INT Flag Clear Bit |
| | | 0 | Has no effect. Always reads back a 0 |
| | | 1 | Clears the INT flag bit and enable further interrupts pulses to be generated |

### *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. ET Force Register (ETFRC)*

**Note:** R − Read, W − Write

| 15 8 |
|---|
| Reserved |
| R = 0 |

| 7 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|
| Reserved | SOCB | SOCA | Reserved | INT |
| R = 0 | R = 0<br>W = 1 | R = 0<br>W = 1 | R = 0 | R = 0<br>W = 1 |

*Table 4−27. ET 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 SOCBFLG 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. |

*Table 4−27. ET Force Register (ETFRC) Field Descriptions (Continued)*

| Bits | Name | Value | Description |
|---|---|---|---|
| 2 | SOCA | | SOCA Force Bit. The SOCA pulse will only be generated if the event is enabled in the ETSEL register. The SOCAFLG 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. |