第八讲 增强型脉宽调制模块ePWM

主讲：叶剑
电话：13728639620
Email：yejian@hit.edu.cn

第八讲 增强型脉宽调制模块ePWM

1、PWM基本原理

2、概述

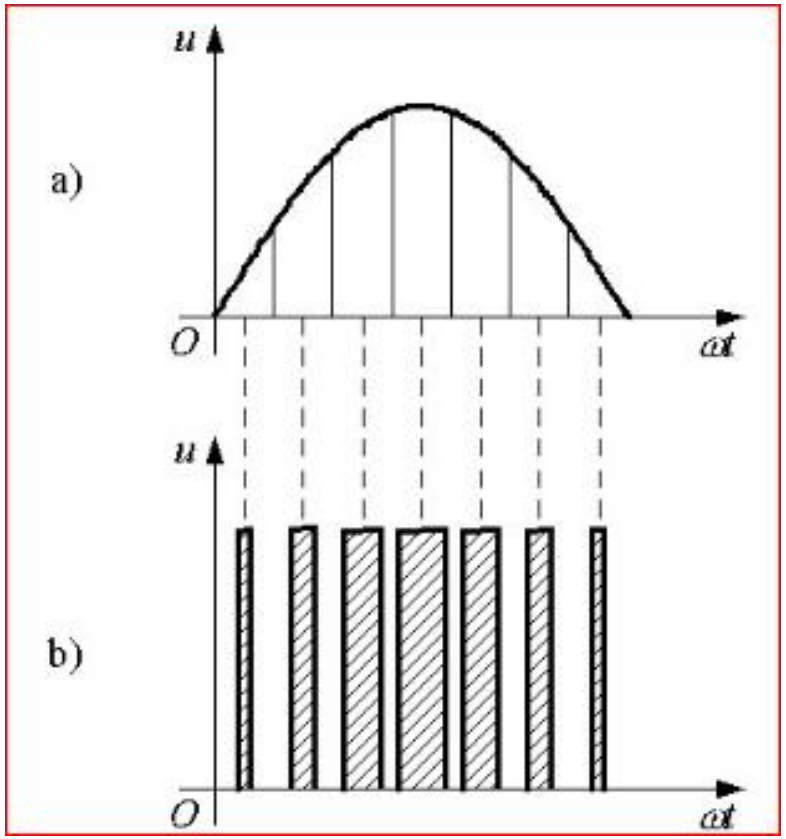
3、ePWM子模块

4、ePWM寄存器

5、例子

PWM技术简介

PWM (Pulse Width Modulation) 就是脉宽调制技术：即通过对一系列脉冲的宽度进行调制，来等效的获得所需要的波形。



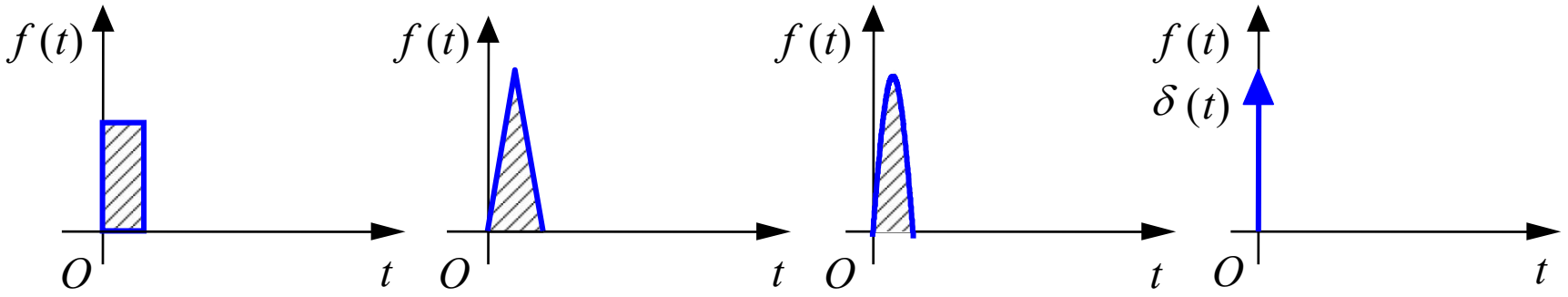
PWM的基本原理

重要理论基础——面积等效原理

冲量相等而形状不同的窄脉冲加在具有惯性的环节上时，其**效果基本相同**。

冲量 → 窄脉冲的面积

效果基本相同 → 环节的输出响应波形基本相同



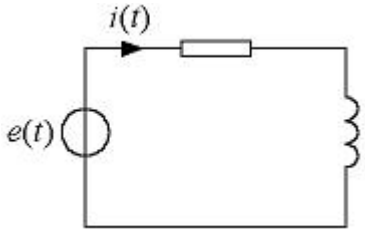
a)矩形脉冲　b)三角形脉冲　c)正弦半波脉冲　d)单位脉冲函数

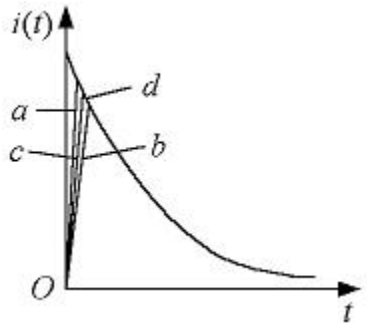
形状不同而冲量相同的各种窄脉冲

PWM的基本原理

具体的实例说明

“面积等效原理”

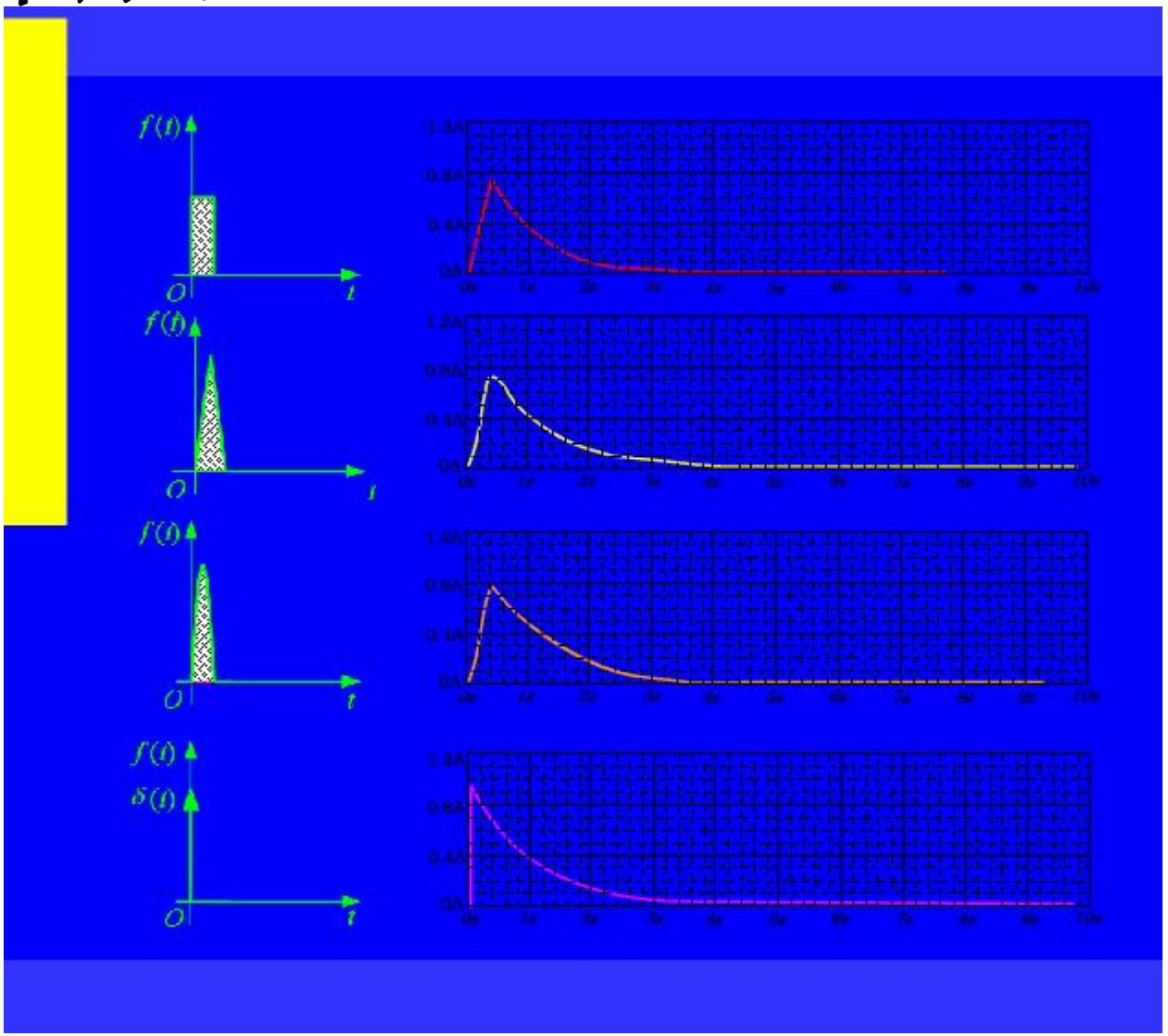




冲量相等的各种窄脉冲的响应波形

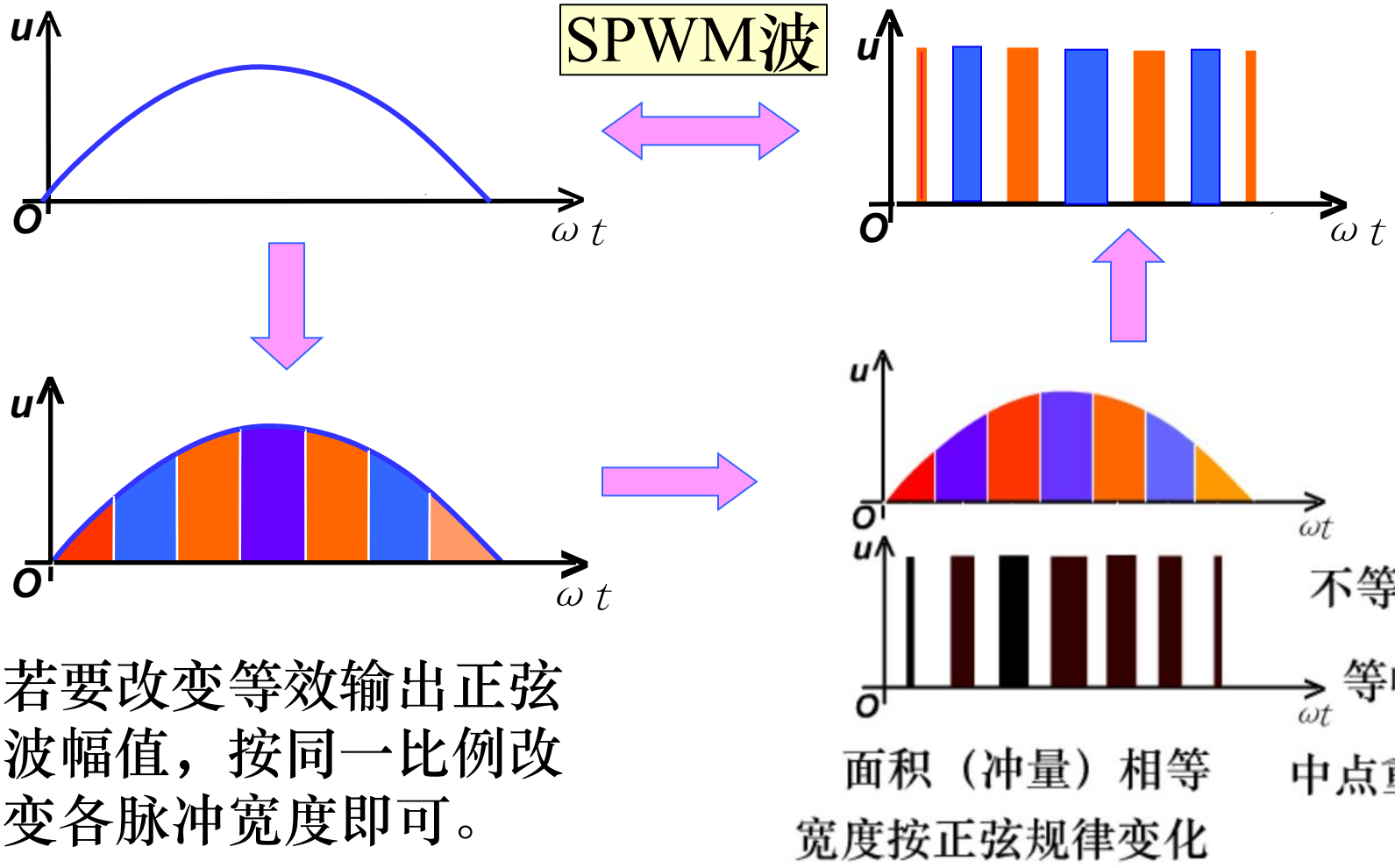
u (t)－电压窄脉冲，是电路的输入 。

i (t)－输出电流，是电路的响应。



PWM的基本原理

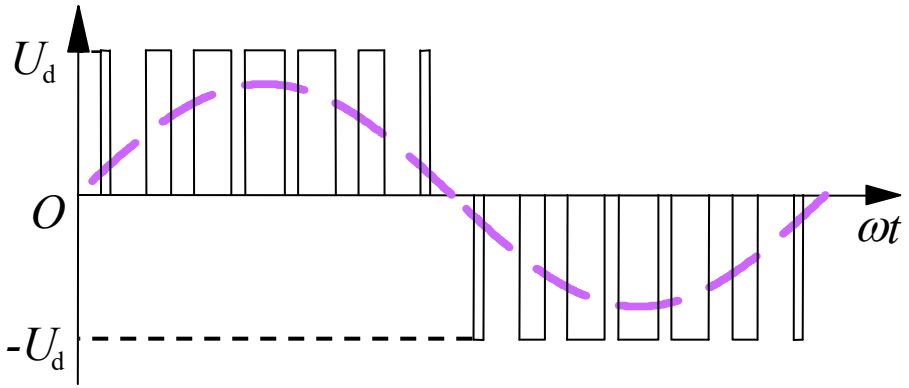
如何用一系列**等幅不等宽的脉冲**来代替一个正弦半波



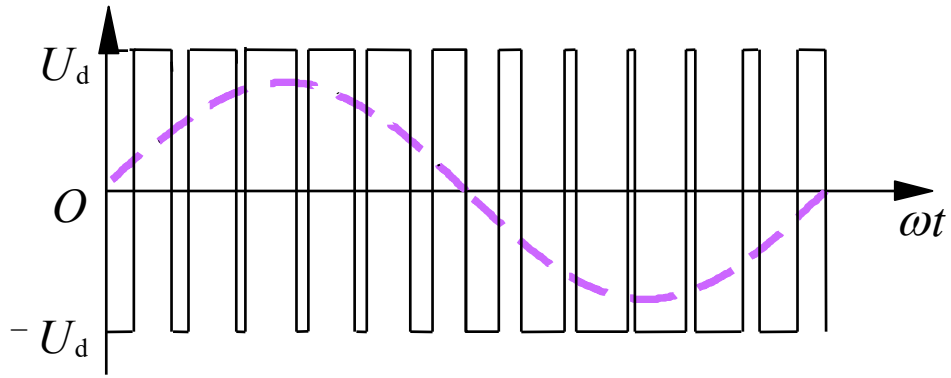
若要改变等效输出正弦波幅值，按同一比例改变各脉冲宽度即可。

PWM的基本原理

对于正弦波的负半周，采取同样的方法，得到PWM波形，因此正弦波一个完整周期的等效PWM波为：



根据面积等效原理，正弦波还可等效为下图中的PWM波，而且这种方式在实际应用中更为广泛。



第八讲 增强型脉宽调制模块ePWM

1、PWM基本原理

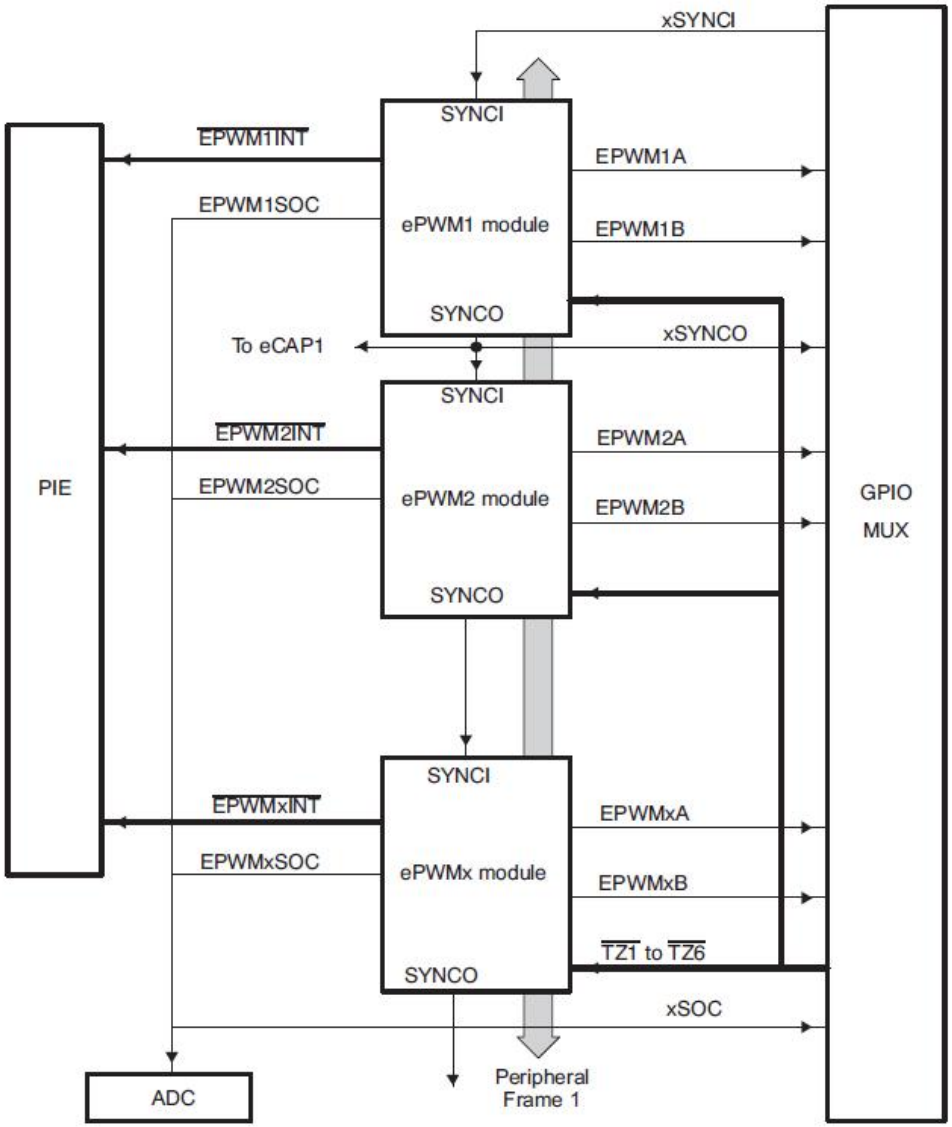
2、概述

3、ePWM子模块

4、ePWM寄存器

5、例子

ePWM模块级联结构



每组PWM模块主要的输入输出信号如下

（1）PWM输出信号（ePWMxA和ePWMxB）。

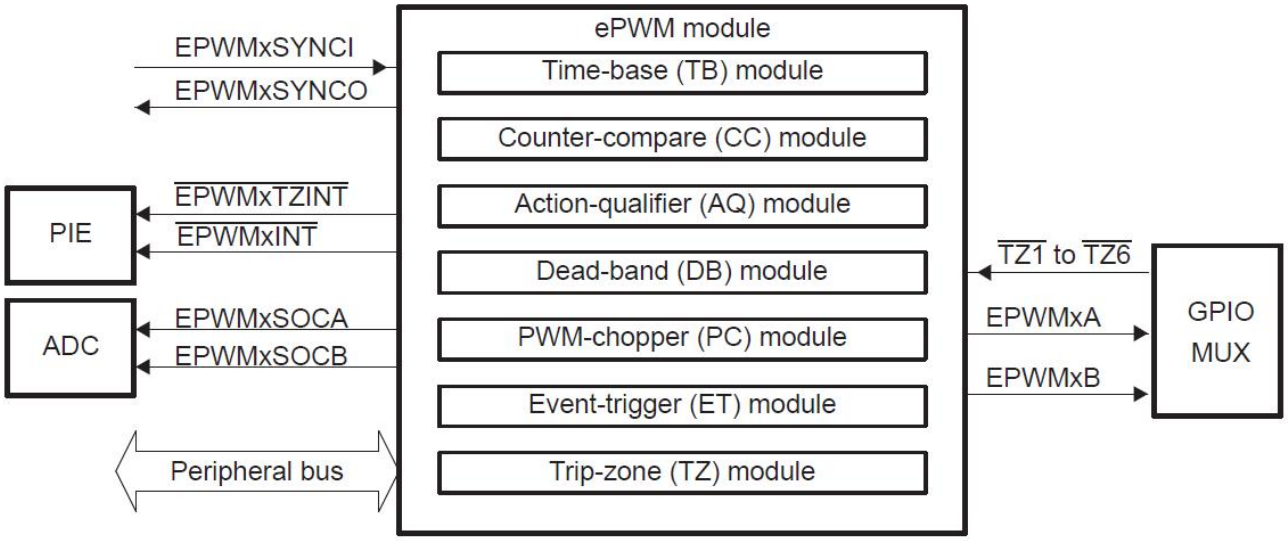
（2）时间基础同步输入（ePWMXSYNCI）和输出(ePWMxSYNCO)信号。

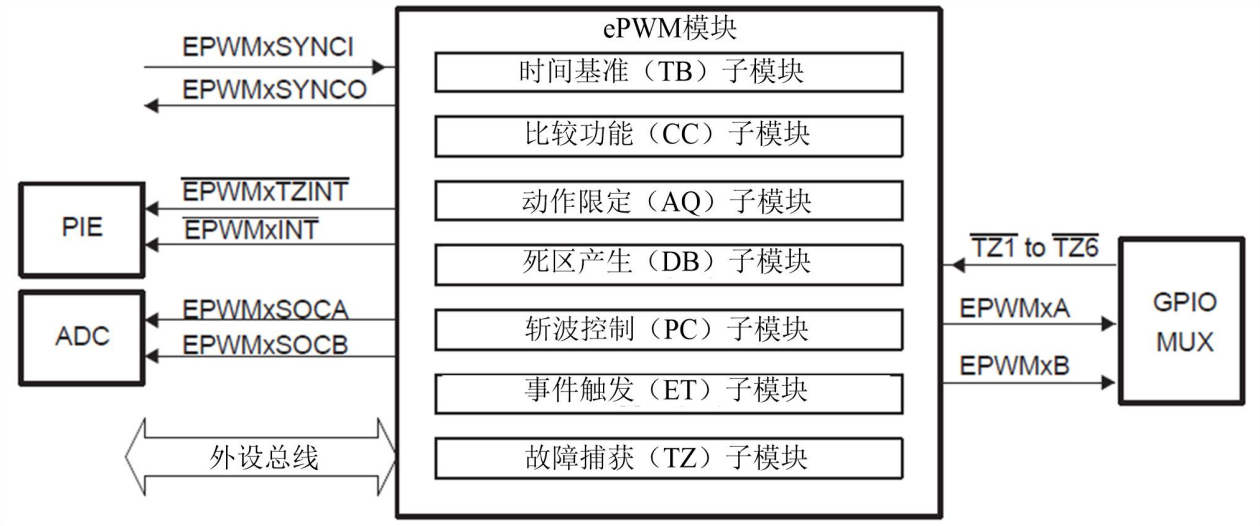
（3）外部触发信号（ TZ1-TZ6）。当外部被控单元符合错误条件时，诸如IGBT等功率器件模块过电压、过电流或过热时，这些输入信号为ePWM模块发出错误警告。

（4）ADC启动信号（ePWMSOCA和ePWMSOCB）。每个ePWM模块都有两个ADC转换启动信号。

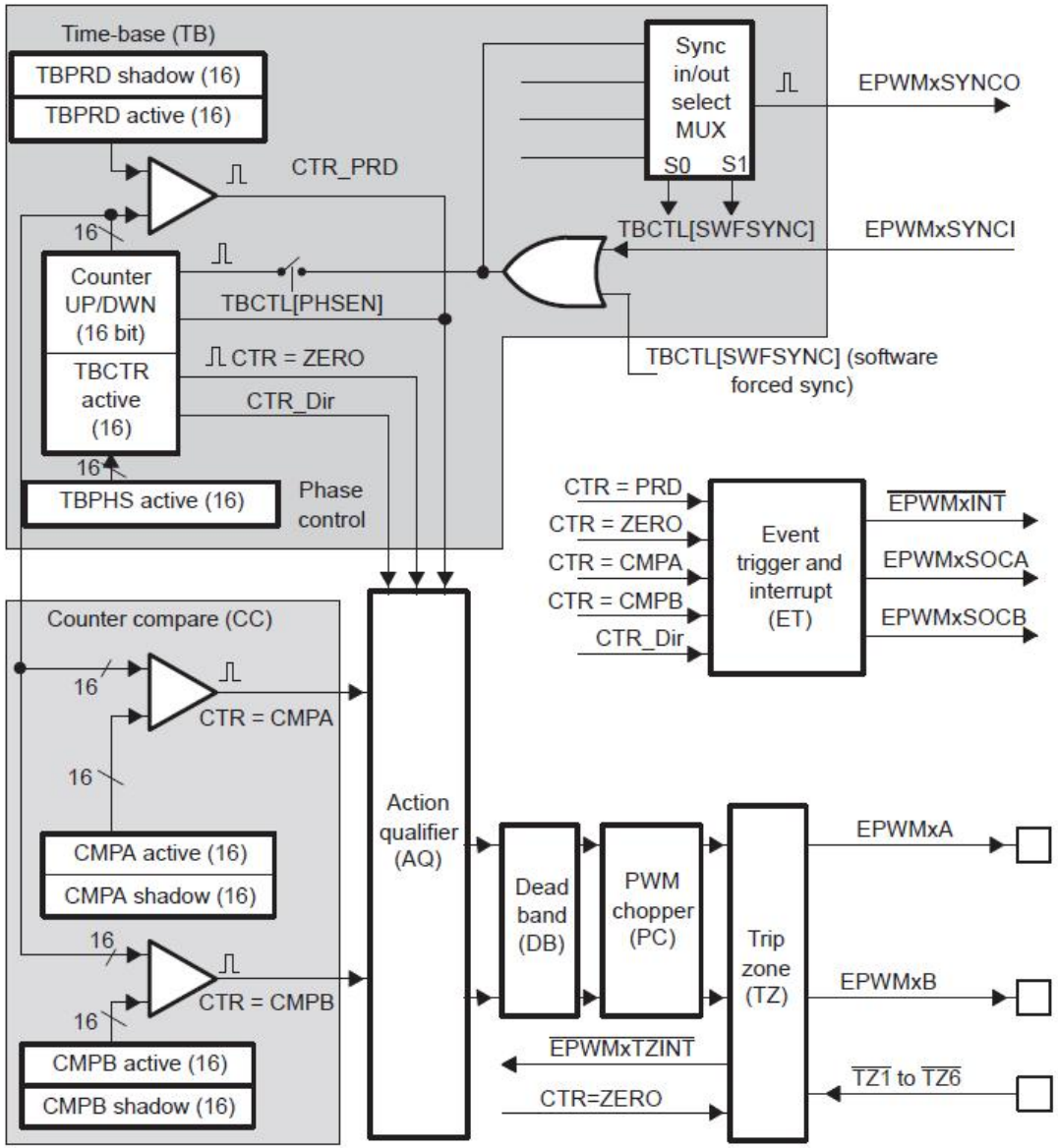
（5）外设总线。外设总线宽度为32位，允许16位和32位数据通过外设总线写入ePWM模块寄存器。

ePWM模块的子模块


ePWM module
EPWMxSYNCI
EPWMxSYNCO
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
PIE
EPWMxTZINT
EPWMxINT
ADC
EPWMxSOCA
EPWMxSOCB
Peripheral bus
TZ1 to TZ6
EPWMxA
EPWMxB
GPIO MUX


ePWM模块
EPWMxSYNCI
EPWMxSYNCO
时间基准（TB）子模块
比较功能（CC）子模块
动作限定（AQ）子模块
死区产生（DB）子模块
斩波控制（PC）子模块
事件触发（ET）子模块
故障捕获（TZ）子模块
PIE
EPWMxTZINT
EPWMxINT
ADC
EPWMxSOCA
EPWMxSOCB
外设总线
TZ1 to TZ6
EPWMxA
EPWMxB
GPIO MUX

ePWM模块内部结构框图



第八讲 增强型脉宽调制模块ePWM

1、PWM基本原理

2、概述

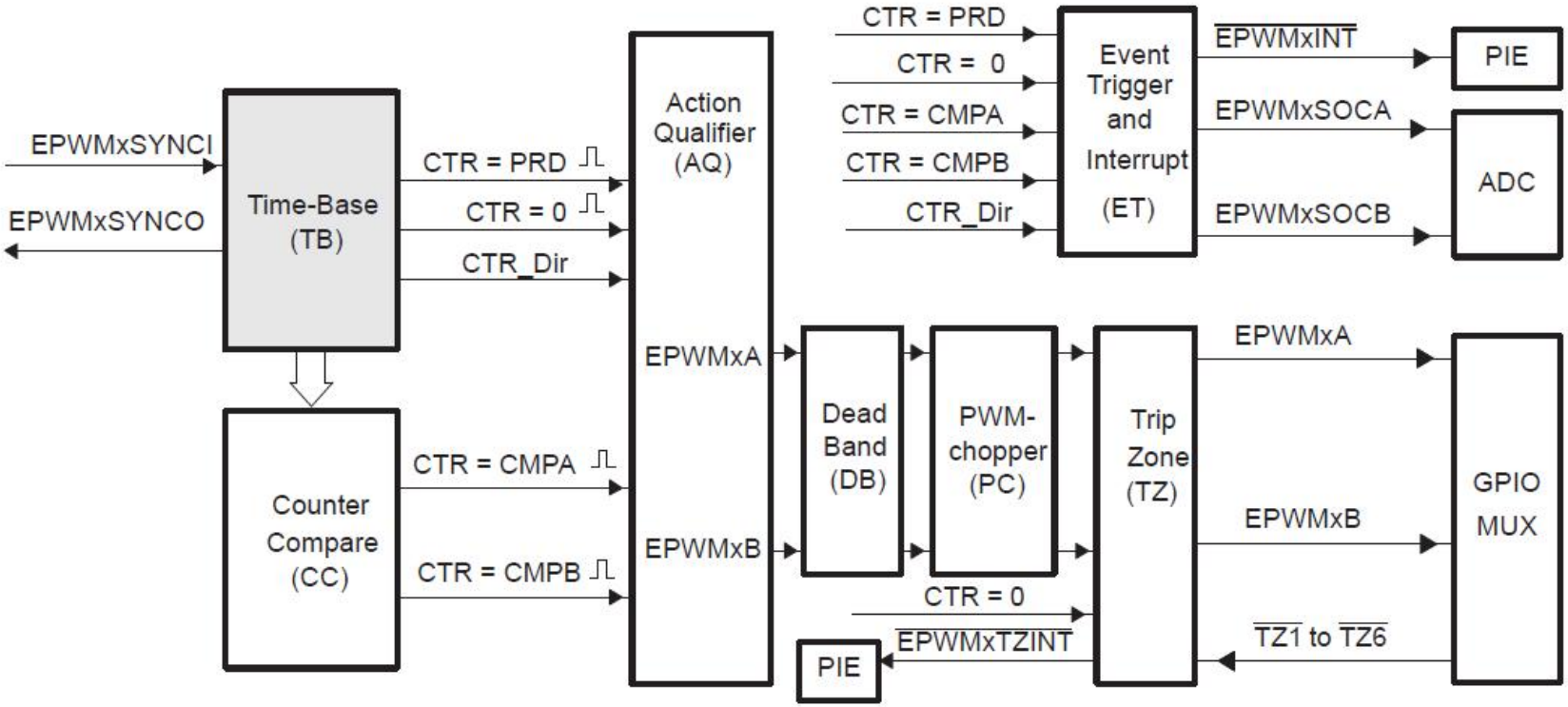
3、ePWM子模块

4、ePWM寄存器

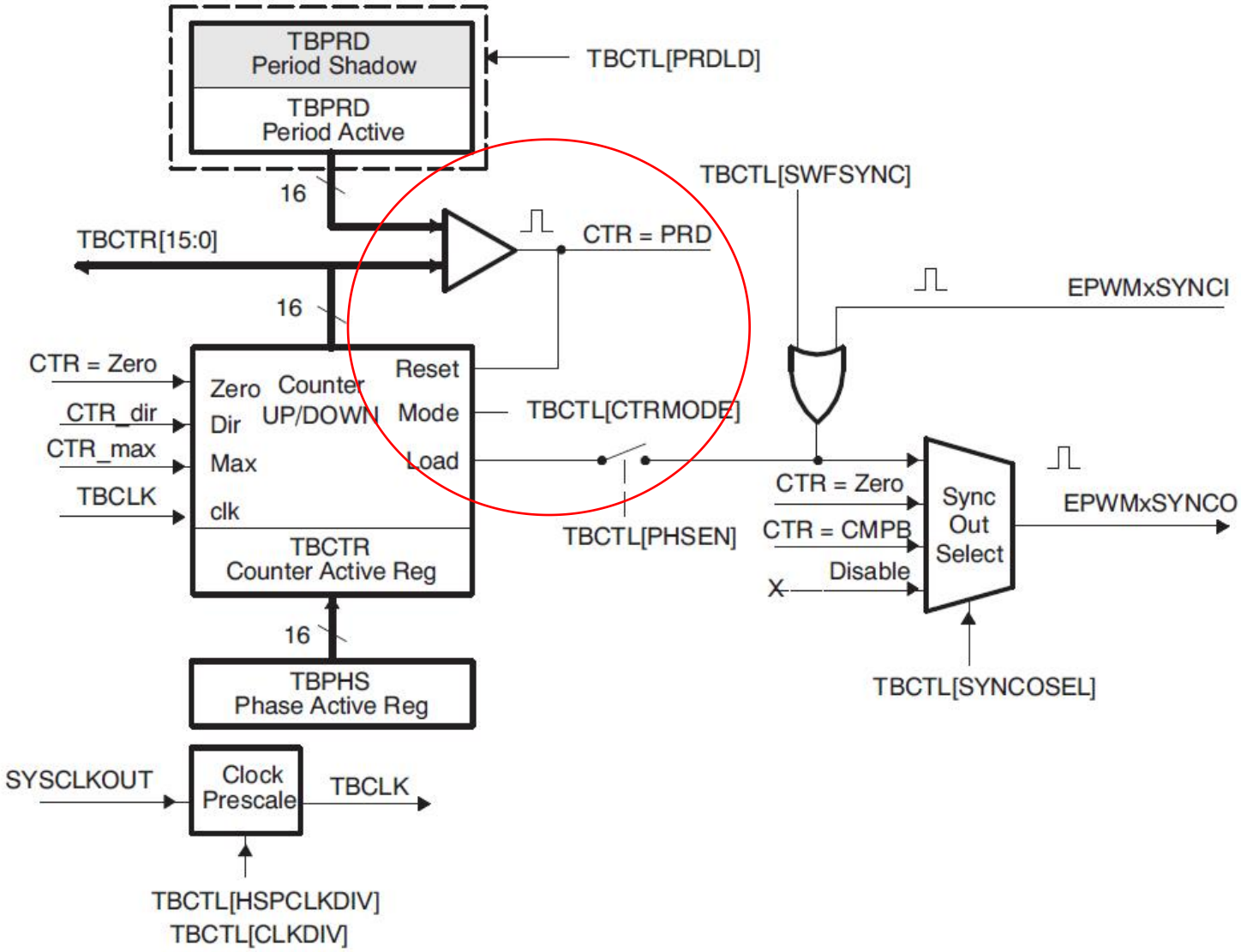
5、例子

时间基准子模块

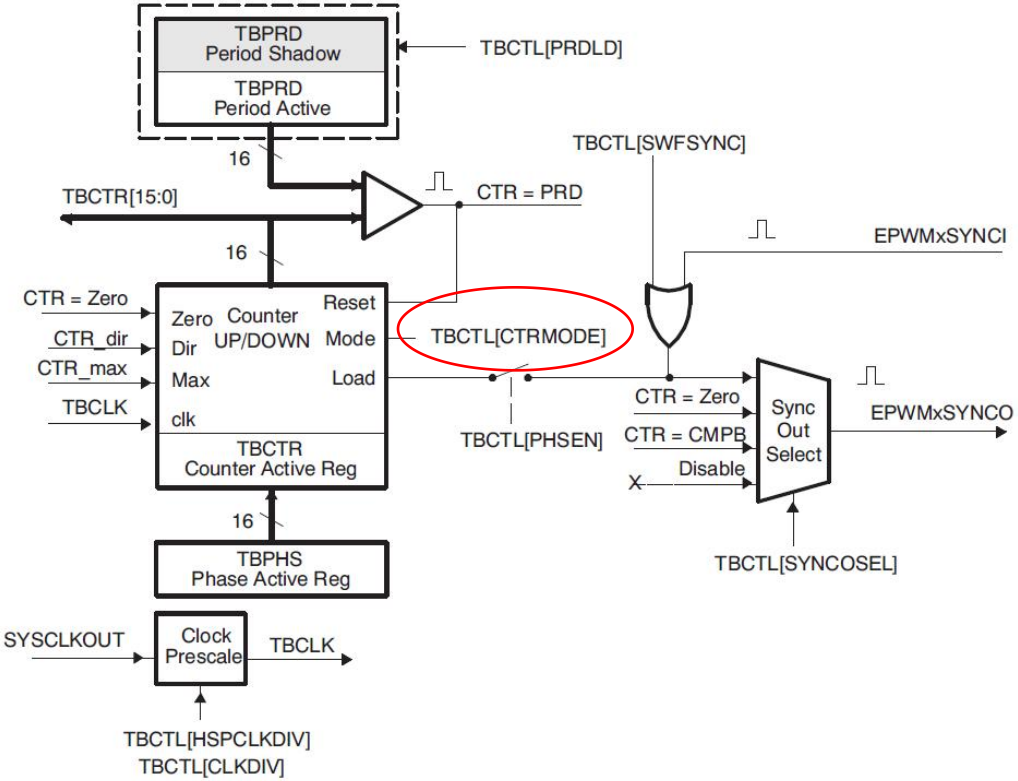
每个ePWM模块都有自己的时间基准子模块，用以决定ePWM模块工作时序



1. TB子模块内部结构

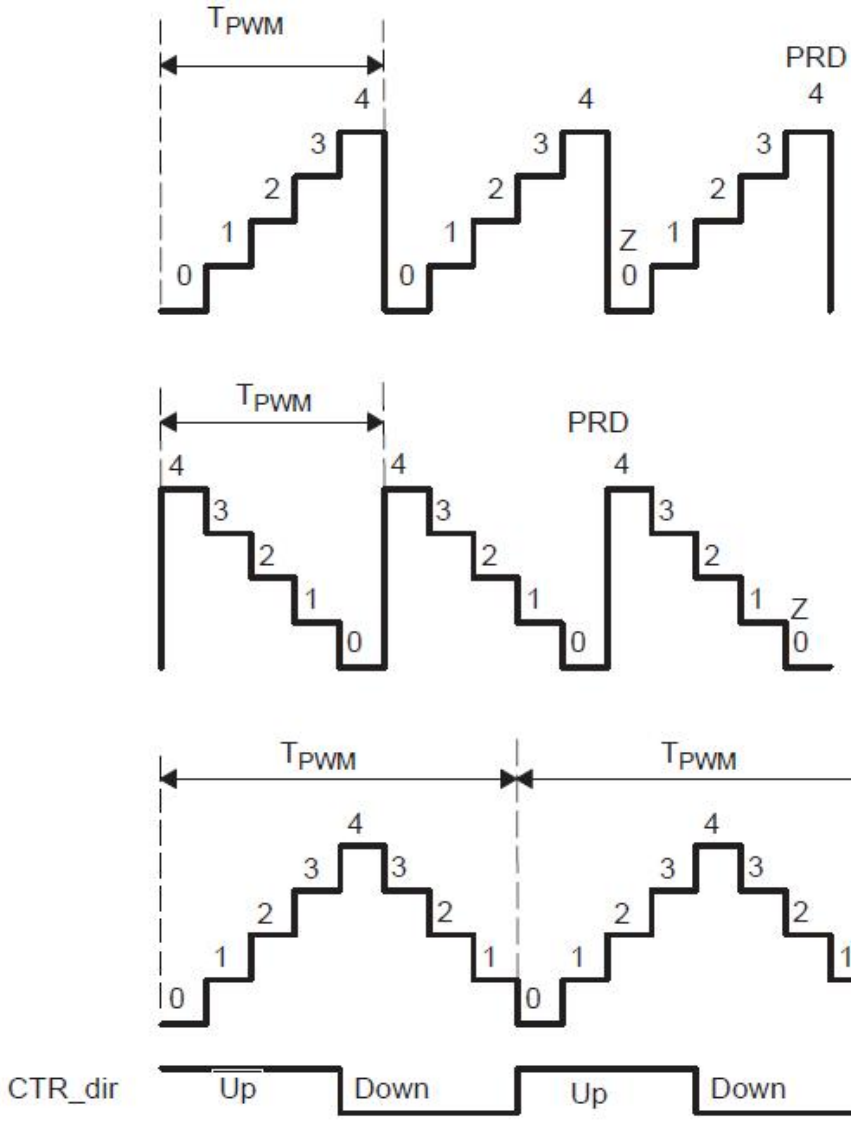

TBPRD
Period Shadow
TBPRD
Period Active
TBCTL[PRDLD]
16
TBCTR[15:0]
CTR = PRD
TBCTL[SWFSYNC]
EPWMxSYNCI
16
CTR = Zero
CTR_dir
CTR_max
TBCLK
Zero
Dir
Max
clk
Counter
UP/DOWN
Reset
Mode
Load
TBCTR
Counter Active Reg
TBCTL[CTRMODE]
TBCTL[PHSEN]
CTR = Zero
CTR = CMPB
Disable
X
Sync
Out
Select
EPWMxSYNCO
TBCTL[SYNCOSEL]
16
TBPHS
Phase Active Reg
SYSCLKOUT
Clock
Prescale
TBCLK
TBCTL[HSPCLKDIV]
TBCTL[CLKDIV]

2. 计算PWM周期和频率



1:0	CTRMODE		Counter Mode The time-base counter mode is normally configured once and not changed during normal operation. If you change the mode of the counter, the change will take effect at the next TBCLK edge and the current counter value shall increment or decrement from the value before the mode change. These bits set the time-base counter mode of operation as follows:
		00	Up-count mode
		01	Down-count mode
		10	Up-down-count mode
		11	Stop-freeze counter operation (default on reset)

2. 计算PWM周期和频率



For Up Count and Down Count

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

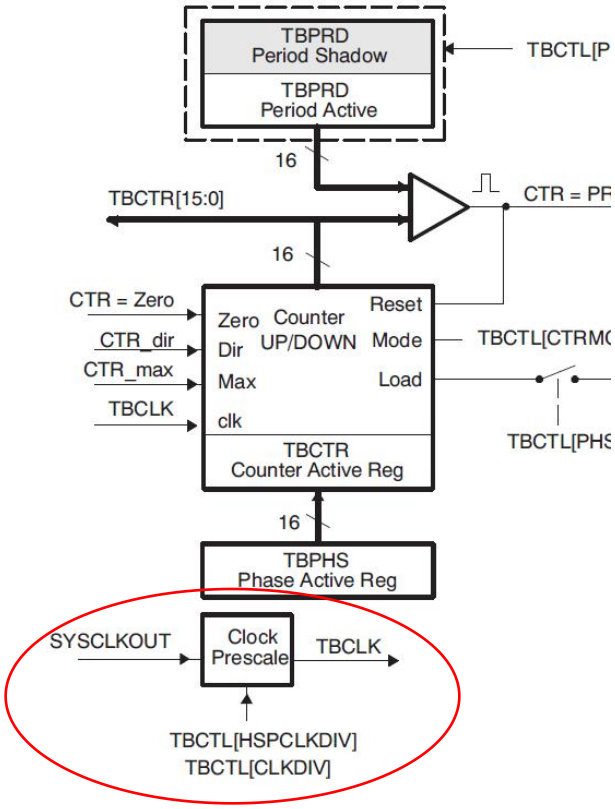
$F_{PWM} = 1/(T_{PWM})$

For Up and Down Count

$T_{PWM} = 2 \times TBPRD \times T_{TBCLK}$

$F_{PWM} = 1/(T_{PWM})$

2. 计算PWM周期和频率



12:10	CLKDIV		Time-base Clock Prescale Bits These bits determine part of the time-base clock prescale value. TBCLK = SYSCLKOUT / (HSPCLKDIV × CLKDIV)
		000	/1 (default on reset)
		001	/2
		010	/4
		011	/8
		100	/16
		101	/32
		110	/64
		111	/128
9:7	HSPCLKDIV		High Speed Time-base Clock Prescale Bits These bits determine part of the time-base clock prescale value. TBCLK = SYSCLKOUT / (HSPCLKDIV × CLKDIV) This divisor emulates the HSPCLK in the TMS320x281x system as used on the (EV) peripheral.
		000	/1
		001	/2 (default on reset)
		010	/4
		011	/6
		100	/8
		101	/10
		110	/12
		111	/14

3. TB周期寄存器及其影子寄存器

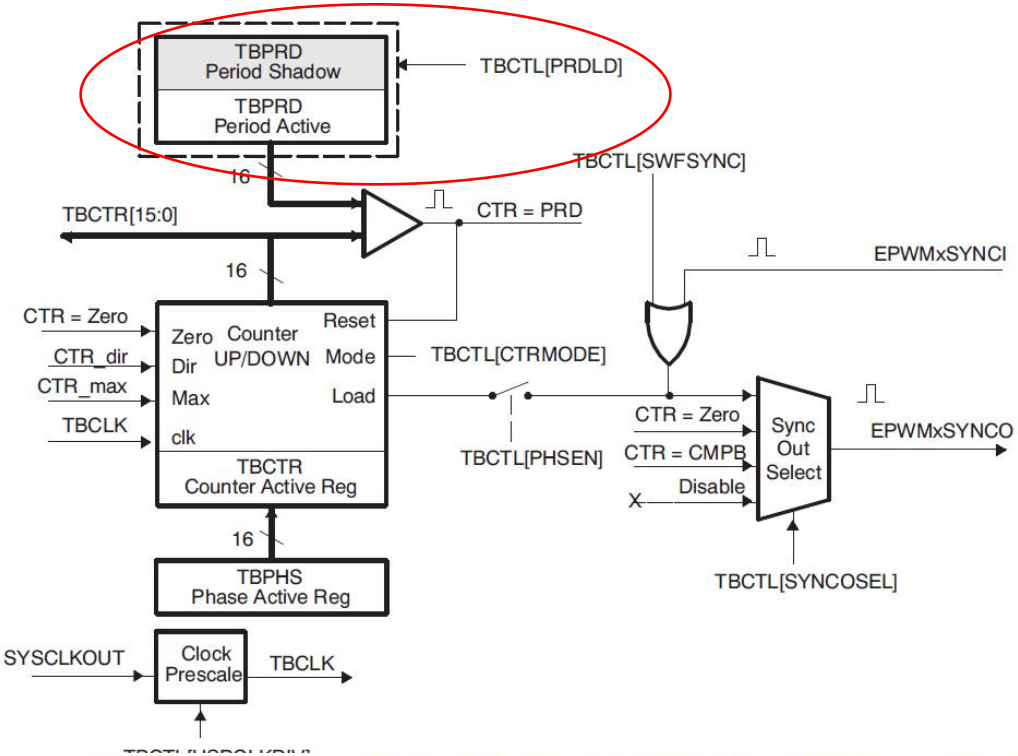
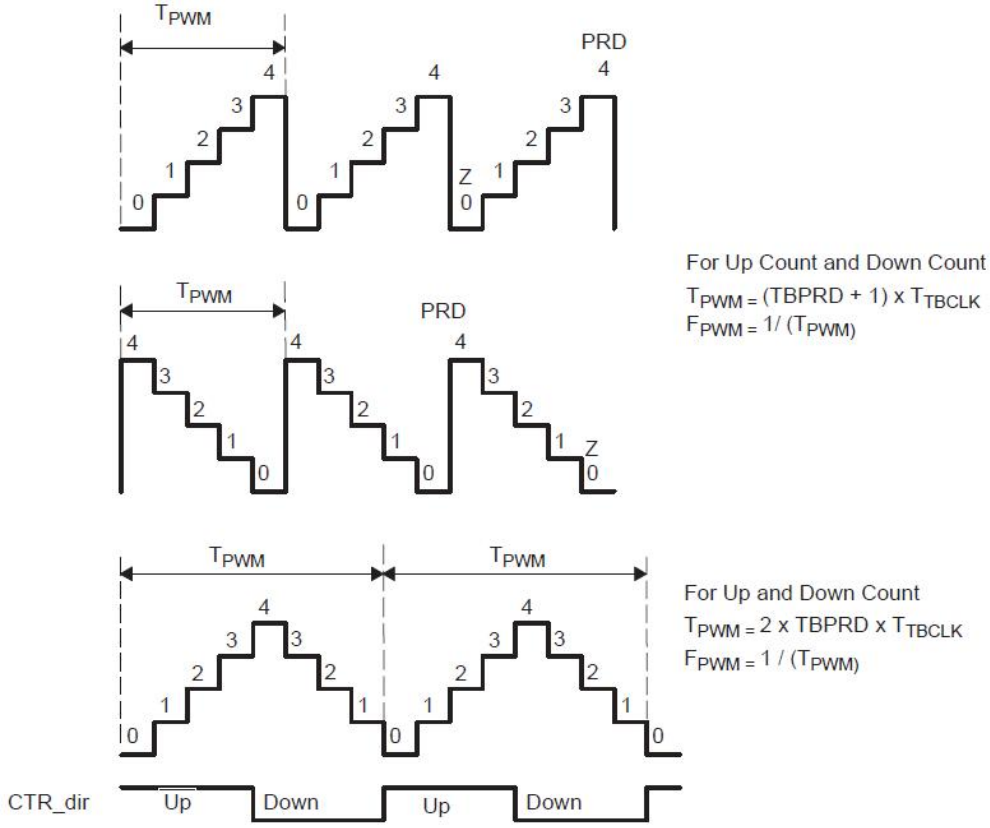


Table 21. Time-Base Period Register (TBPRD) Field Descriptions

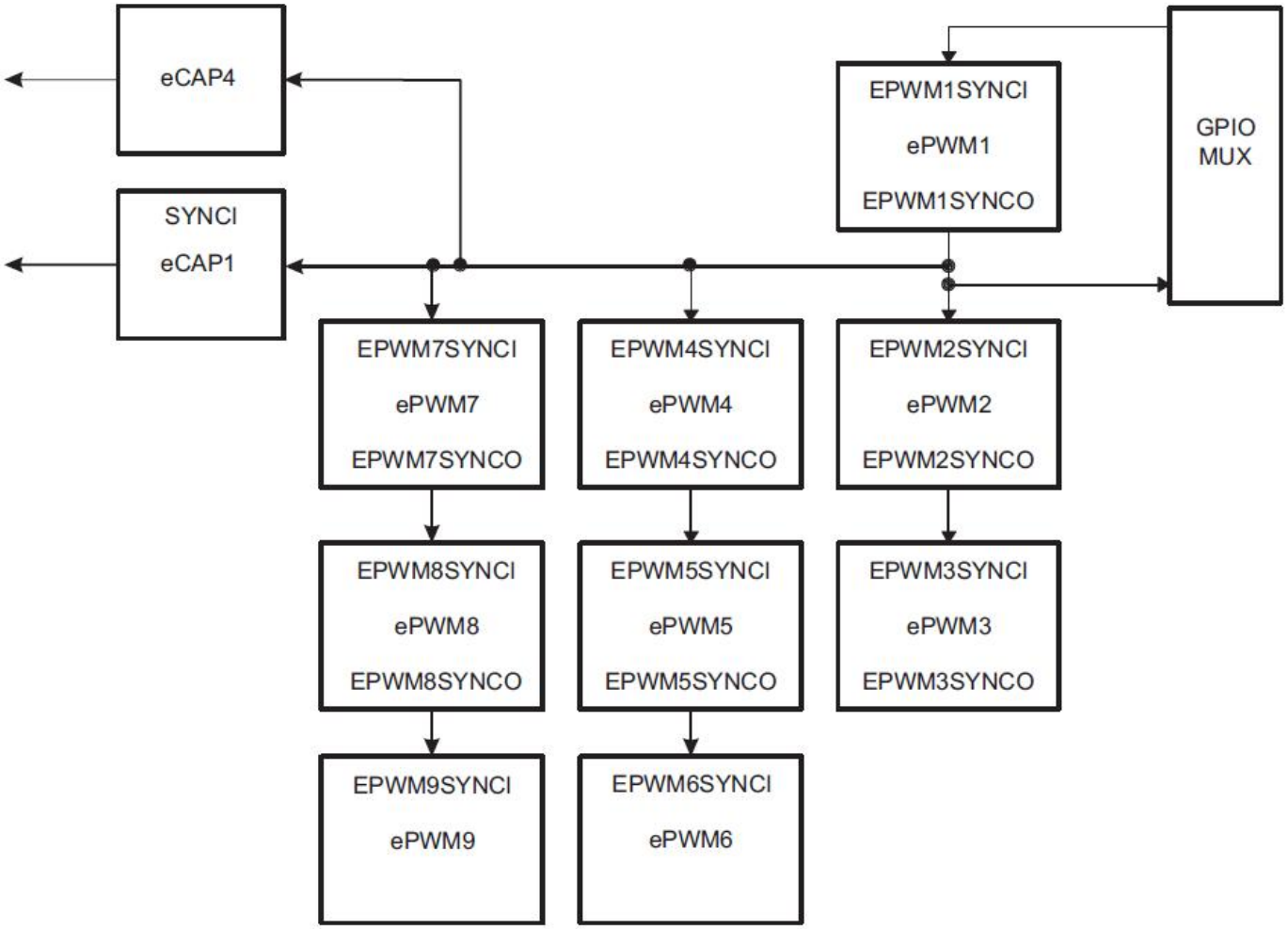
Bits	Name	Value	Description
15-0	TBPRD	0000-FFFFh	These bits determine the period of the time-base counter. This sets the PWM frequency. Shadowing of this register is enabled and disabled by the TBCTL[PRDLD] bit. By default this register is shadowed. • If TBCTL[PRDLD] = 0, then the shadow is enabled and any write or read will automatically go to the shadow register. In this case, the active register will be loaded from the shadow register when the time-base counter equals zero. • If TBCTL[PRDLD] = 1, then the shadow is disabled and any write or read will go directly to the active register, that is the register actively controlling the hardware. • The active and shadow registers share the same memory map address.

3. TB周期寄存器及其影子寄存器



3	PRDLD		Active Period Register Load From Shadow Register Select
		0	The period register (TBPRD) is loaded from its shadow register when the time-base counter, TBCTR, is equal to zero. A write or read to the TBPRD register accesses the shadow register.
		1	Load the TBPRD register immediately without using a shadow register. A write or read to the TBPRD register directly accesses the active register.

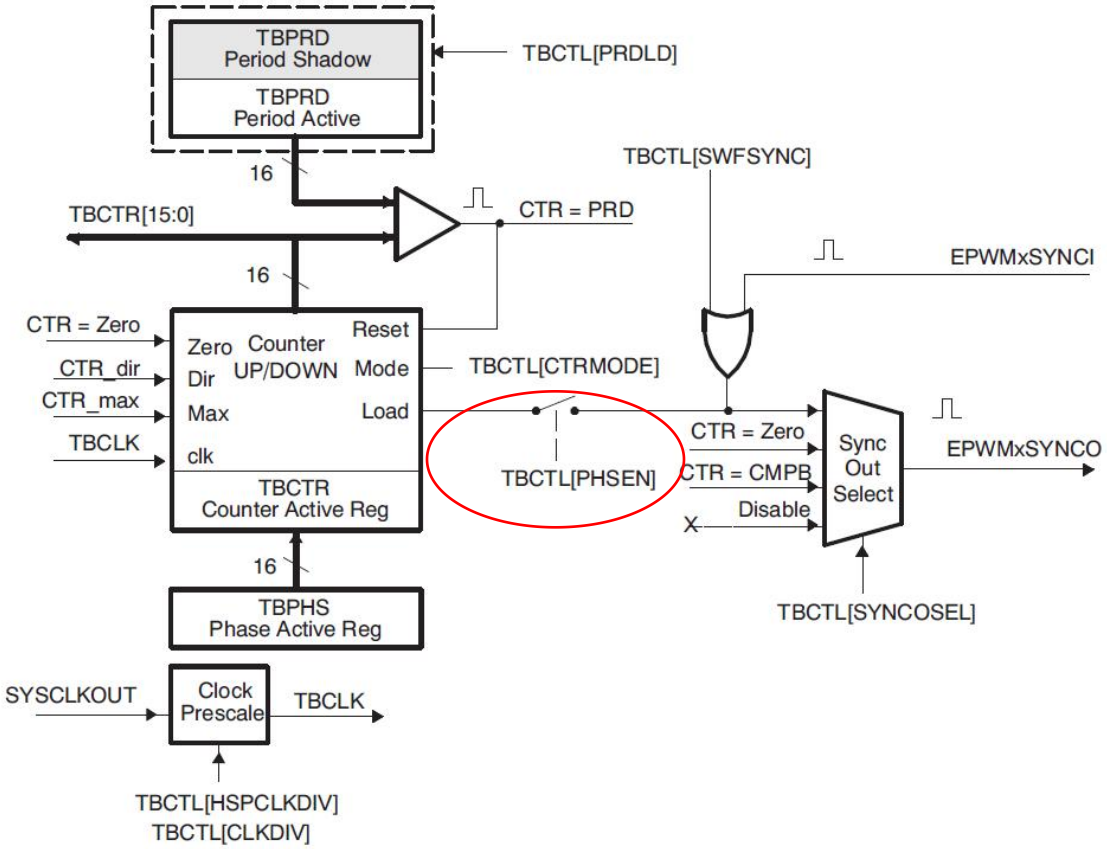
4. TB计数器的同步



NOTE: All modules shown in the synchronization schemes may not be available on all devices. Please refer to the device specific data manual to determine which modules are available on a particular device.

4. TB计数器的同步（有相移）

第一步：TBCTL[PHSEN]选择是否装载TBPHS



2	PHSEN		Counter Register Load From Phase Register Enable
		0	Do not load the time-base counter (TBCTR) from the time-base phase register (TBPHS)
		1	Load the time-base counter with the phase register when an EPWMxSYNCI input signal occurs or when a software synchronization is forced by the SWFSYNC bit

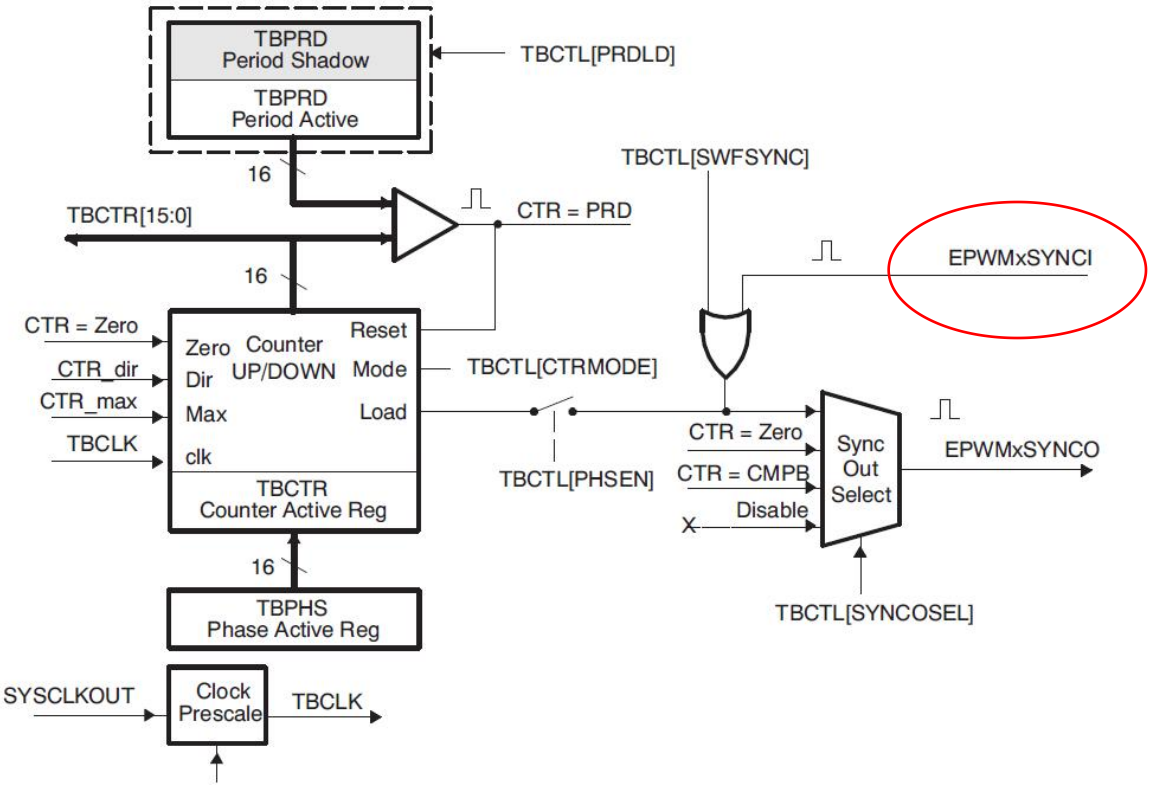
4. TB计数器的同步（有相移）

第二步：以下两种情况可以装载TBPHS

（1）同步脉冲EPWMxSYNCI输入时

The delay from internal master module to slave modules is given by:

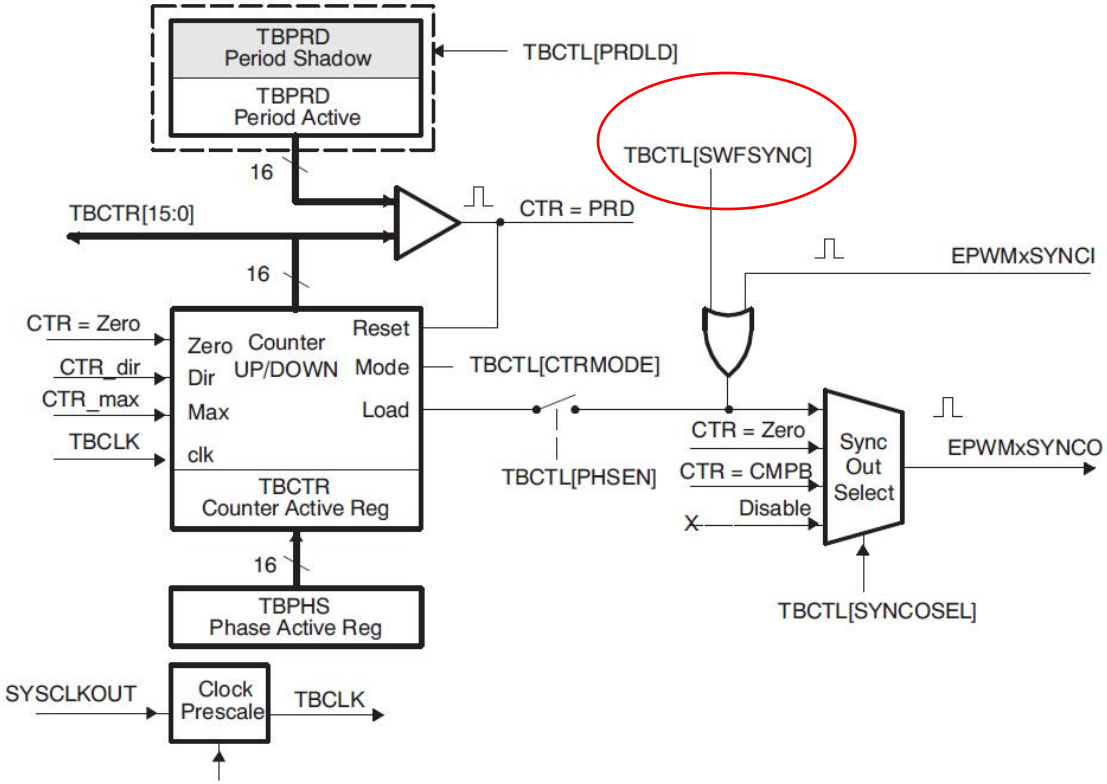
- if (TBCLK = SYSCLKOUT): 2 x SYSCLKOUT
- if (TBCLK != SYSCLKOUT):1 TBCLK



4. TB计数器的同步（有相移）

第二步：以下两种情况可以装载TBPHS

（2）软件强制同步脉冲产生时



4. TB计数器的同步（有相移）

1.从模块PWM脉冲的相位可超前、滞后或同步主模块PWM脉冲。

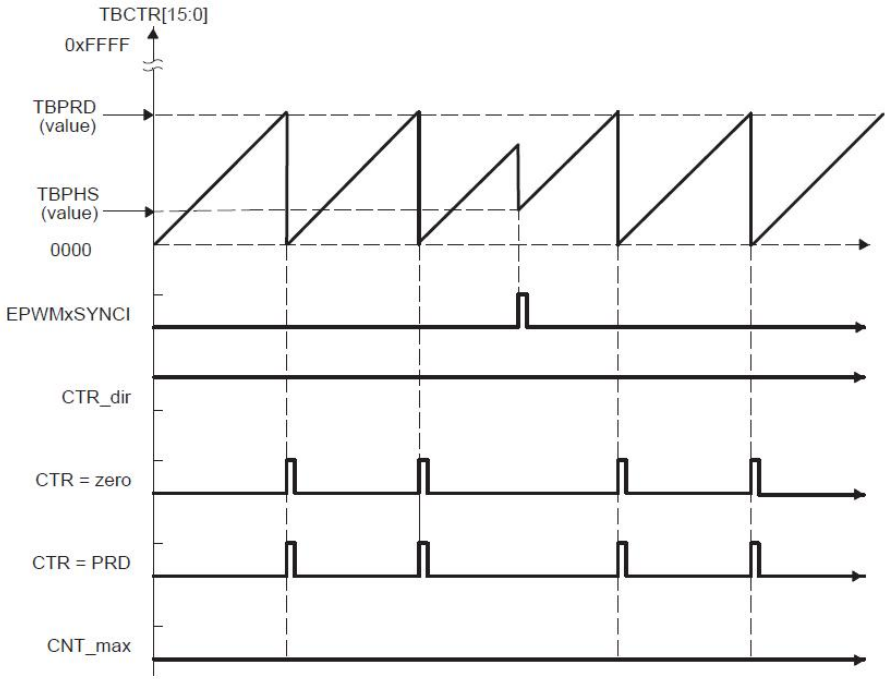
2.在增减计数模式下，TBCTL[PSHDIR]控制TB计数器方向。

3.在增计数或减计数模式下， TBCTL[PSHDIR]被忽略。

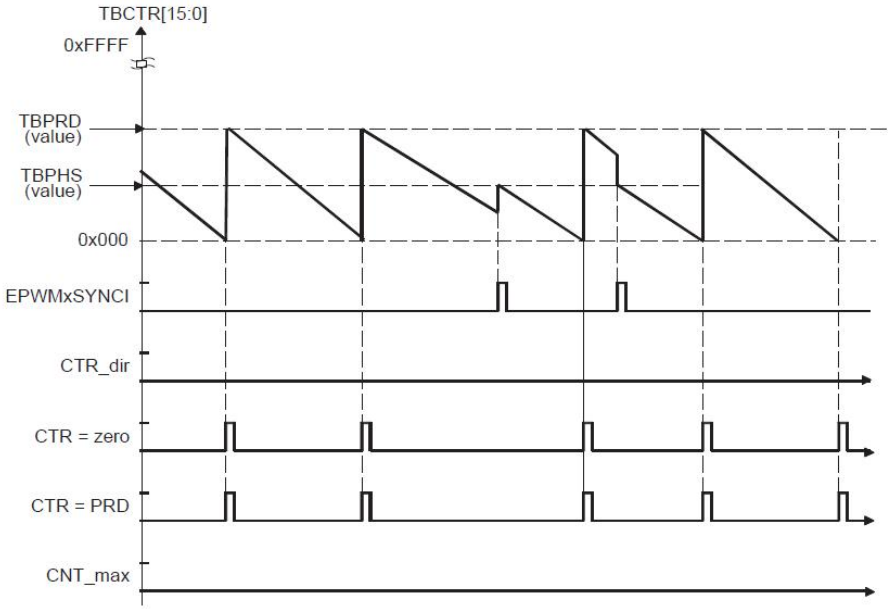
13	PHSDIR		Phase Direction Bit. This bit is only used when the time-base counter is configured in the up-down-count mode. The PHSDIR bit indicates the direction the time-base counter (TBCTR) will count after a synchronization event occurs and a new phase value is loaded from the phase (TBPHS) register. This is irrespective of the direction of the counter before the synchronization event.. In the up-count and down-count modes this bit is ignored.
		0	Count down after the synchronization event.
		1	Count up after the synchronization event.

4. TB计数器的同步（有相移）

Time-Base Up-Count Mode Waveforms



Time-Base Down-Count Mode Waveforms

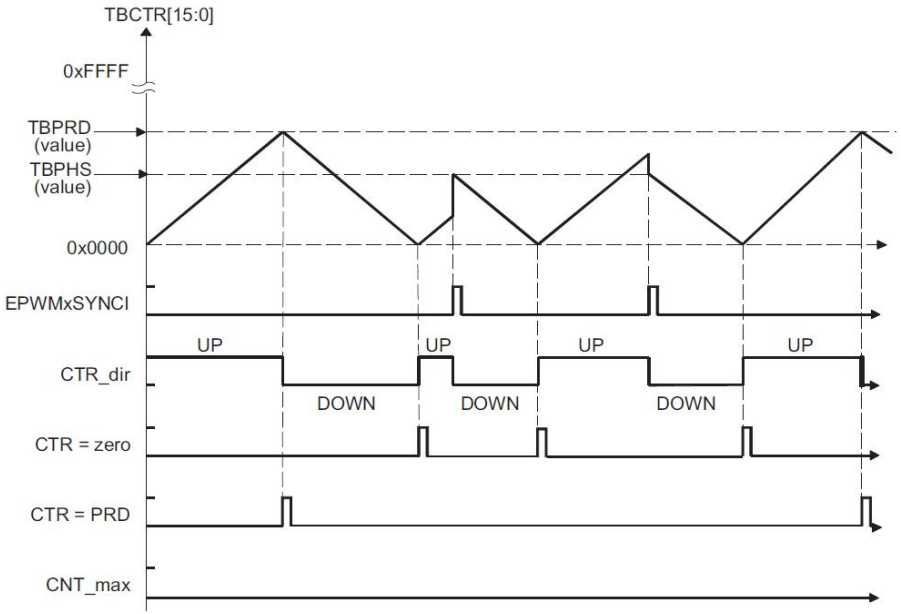


4. TB计数器的同步（有相移）

Time-Base Up-Down-Count Waveforms,

TBCTL[PHSDIR = 0] Count Down

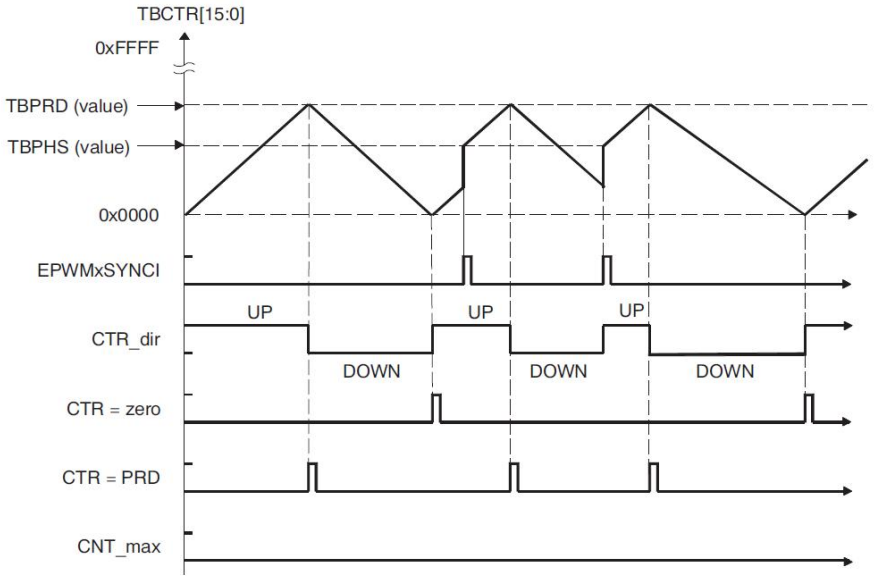
On Synchronization Event



Time-Base Up-Down Count Waveforms,

TBCTL[PHSDIR = 1] Count Up

On Synchronization Event



4. TB时钟的同步

TBCLKSYNC用于同步多个ePWM模块的基准时钟

2	TBCLKSYNC		ePWM Module Time Base Clock (TBCLK) Sync: Allows the user to globally synchronize all enabled ePWM modules to the time base clock (TBCLK):
		0	The TBCLK (Time Base Clock) within each enabled ePWM module is stopped. (default). If, however, the ePWM clock enable bit is set in the PCLKCR1 register, then the ePWM module will still be clocked by SYSCLKOUT even if TBCLKSYNC is 0.
		1	All enabled ePWM module clocks are started with the first rising edge of TBCLK aligned. For perfectly synchronized TBCLKs, the prescaler bits in the TBCTL register of each ePWM module must be set identically. The proper procedure for enabling ePWM clocks is as follows: • Enable ePWM module clocks in the PCLKCR1 register. • Set TBCLKSYNC to 0. • Configure prescaler values and ePWM modes. • Set TBCLKSYNC to 1.

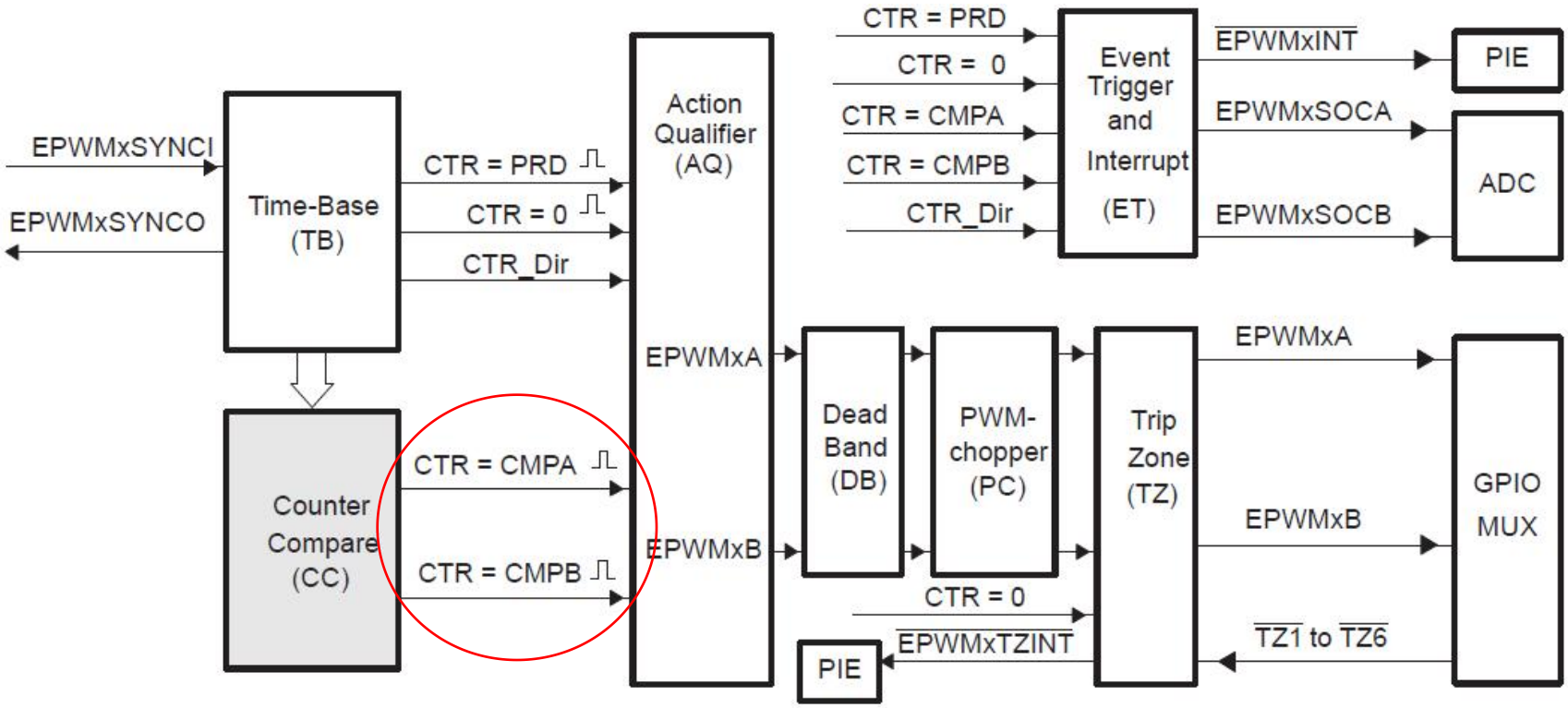
——TI“TMS320x2833x, 2823x System Control and Interrupts：Reference Guide”

比较功能子模块

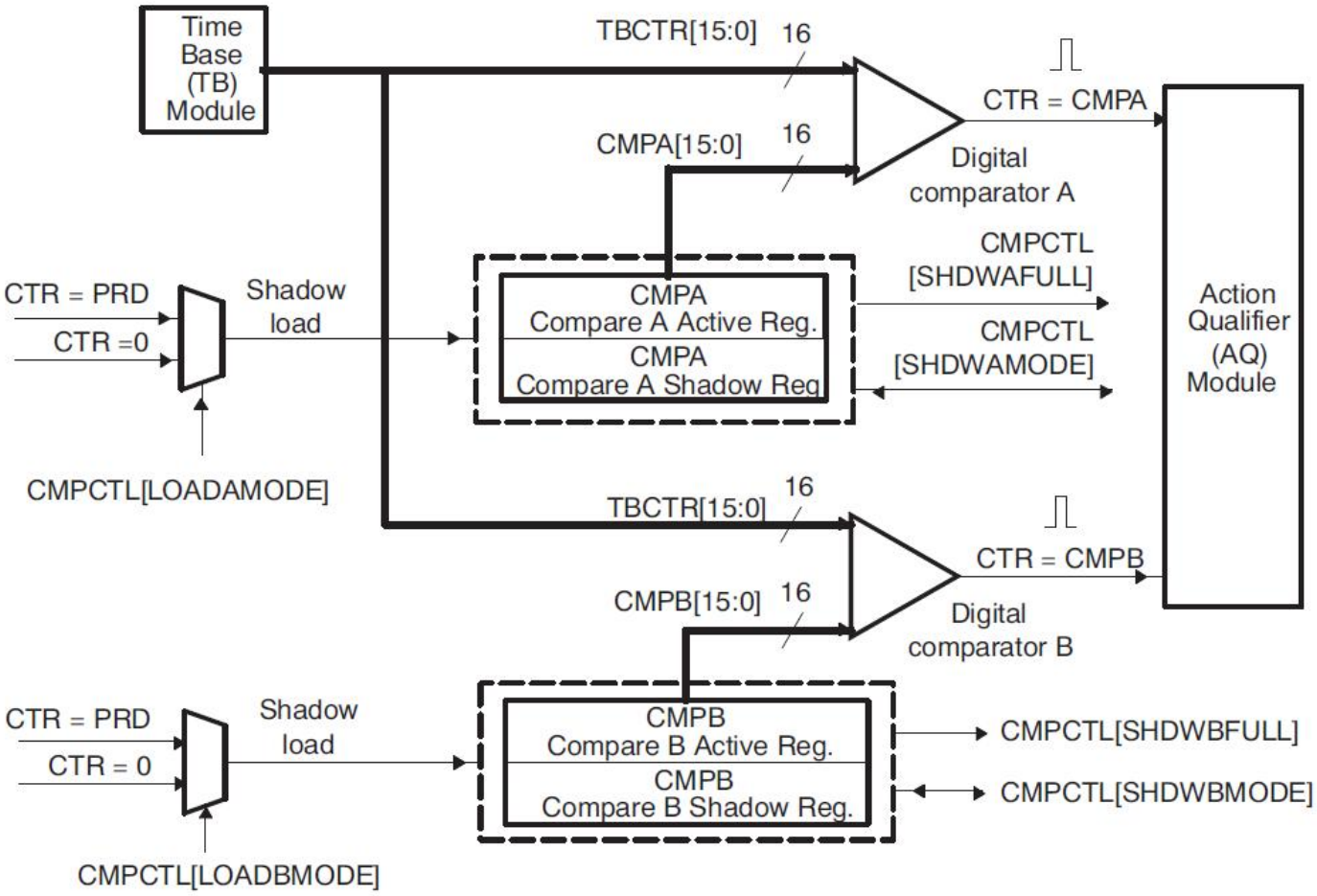
比较功能子模块通过两个寄存器产生两路独立比较事件

(1) CTR = CMPA

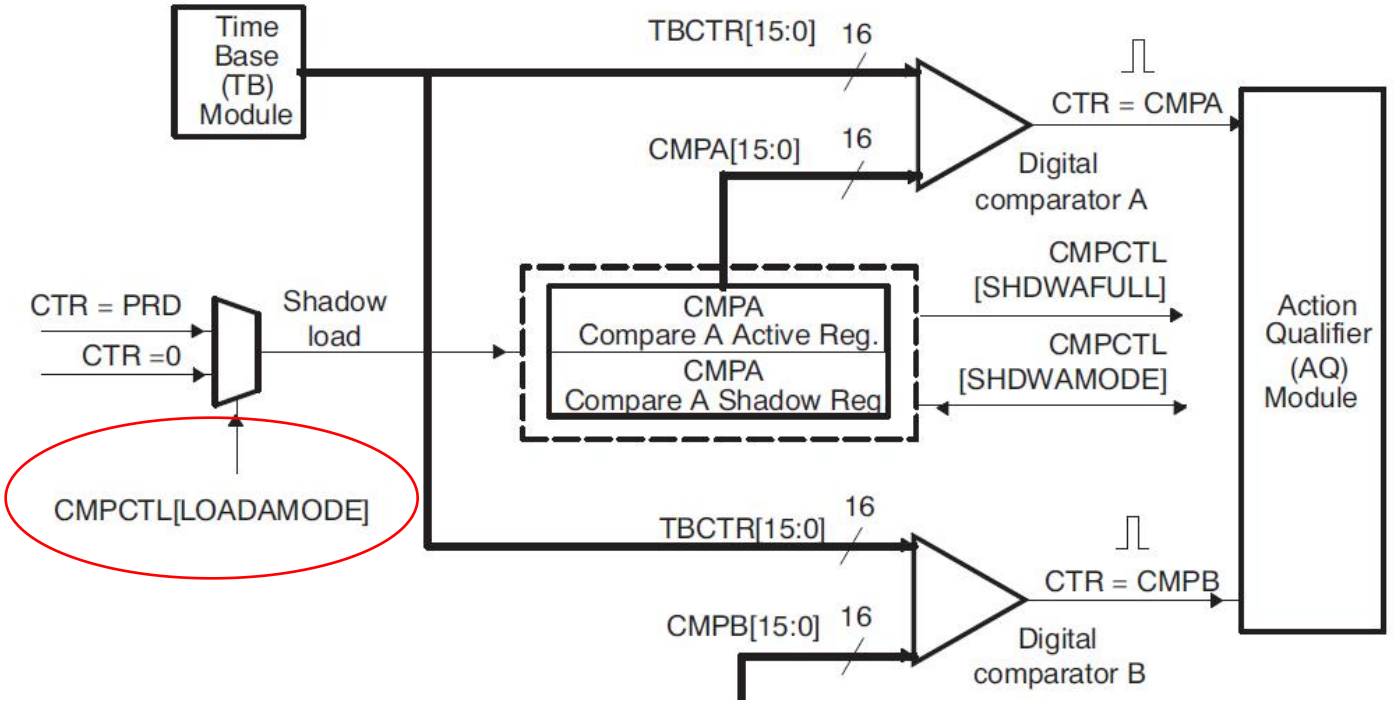
(2) CTR = CMPB



1. CC子模块内部结构


Time Base (TB) Module
TBCTR[15:0]
16
CMPA[15:0]
16
CTR = CMPA
Digital comparator A
CMPCTL [SHDWAFULL]
CMPCTL [SHDWAMODE]
Action Qualifier (AQ) Module
CTR = PRD
CTR =0
Shadow load
CMPA Compare A Active Reg.
CMPA Compare A Shadow Reg
CMPCTL[LOADAMODE]
TBCTR[15:0]
16
CTR = CMPB
CMPB[15:0]
16
Digital comparator B
CTR = PRD
CTR = 0
Shadow load
CMPB Compare B Active Reg.
CMPB Compare B Shadow Reg.
CMPCTL[SHDWBFULL]
CMPCTL[SHDWBMODE]
CMPCTL[LOADBMODE]

1. CC子模块内部结构

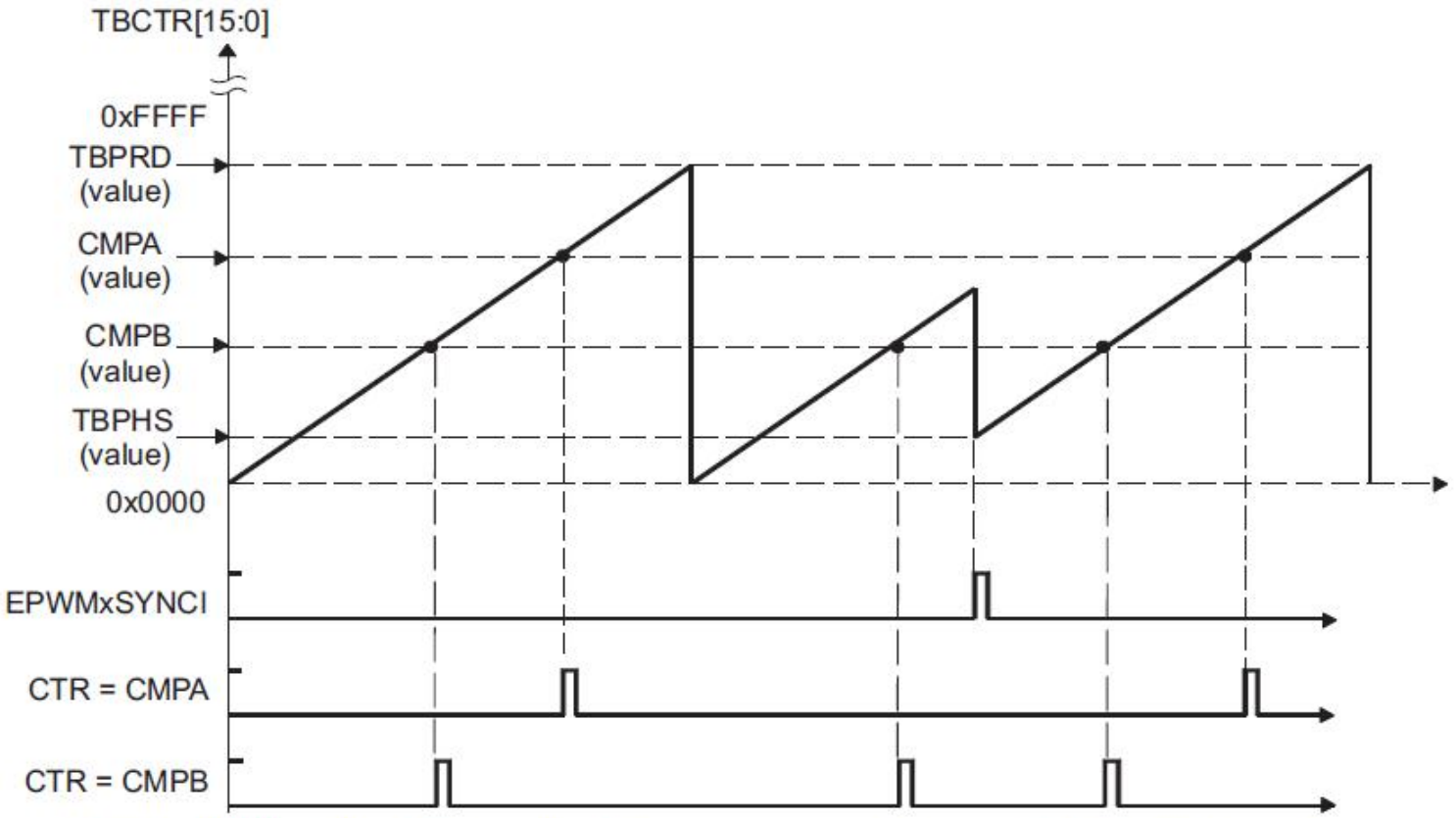


1-0	LOADAMODE		Active Counter-Compare A (CMPA) Load From Shadow Select Mode. This bit has no effect in immediate mode (CMPCTL[SHDWAMODE] = 1).
		00	Load on CTR = Zero: Time-base counter equal to zero (TBCTR = 0x0000)
		01	Load on CTR = PRD: Time-base counter equal to period (TBCTR = TBPRD)
		10	Load on either CTR = Zero or CTR = PRD
		11	Freeze (no loads possible)

2. CC子模块工作时序

比较功能子模块在下列3种计数模式下都可产生比较事件

(1) 增计数模式：不对称PWM脉冲

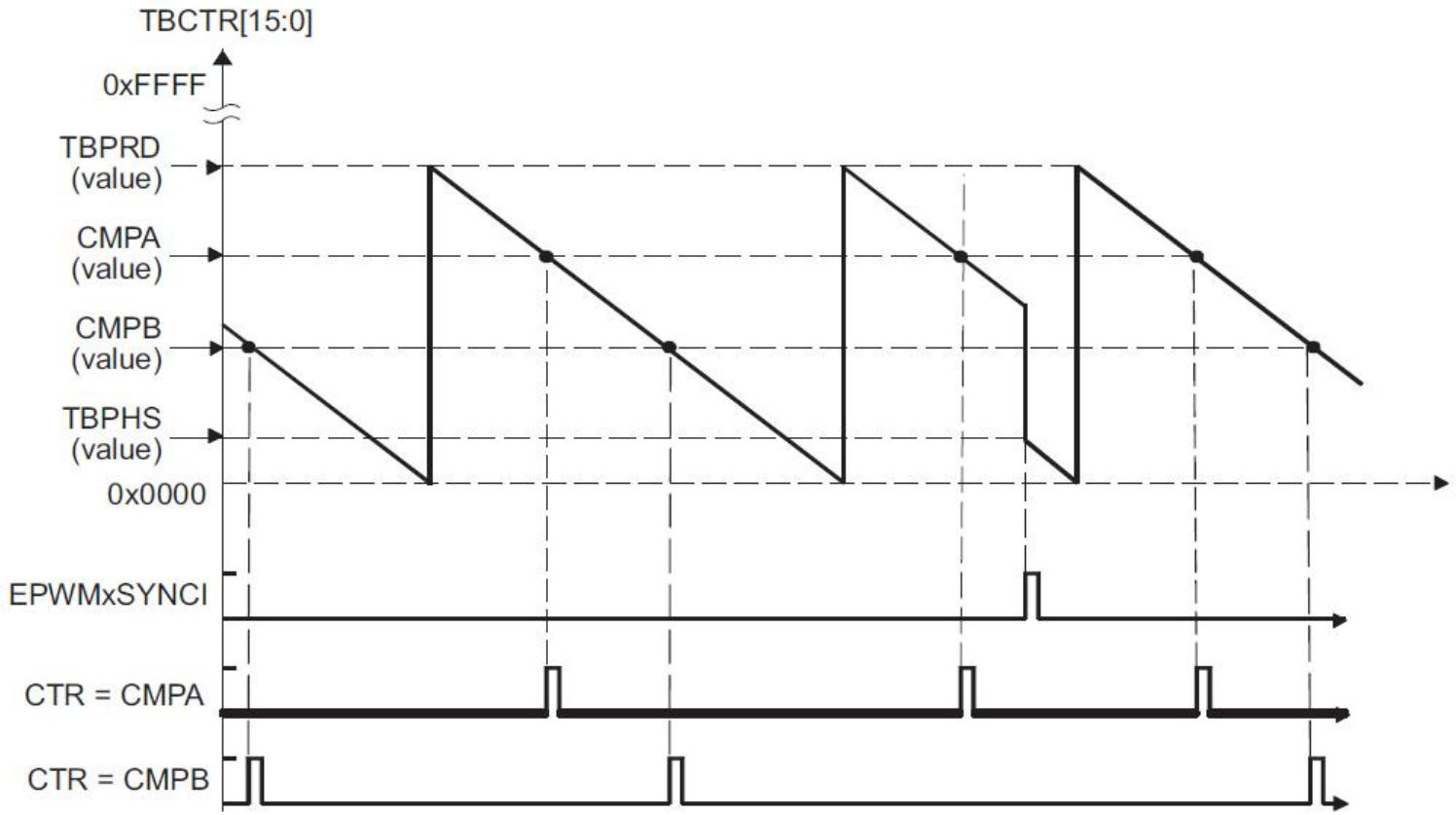


NOTE: An EPWMxSYNCI external synchronization event can cause a discontinuity in the TBCTR count sequence. This can lead to a compare event being skipped. This skipping is considered normal operation and must be taken into account.

2. CC子模块工作时序

比较功能子模块在下列3种计数模式下都可产生比较事件

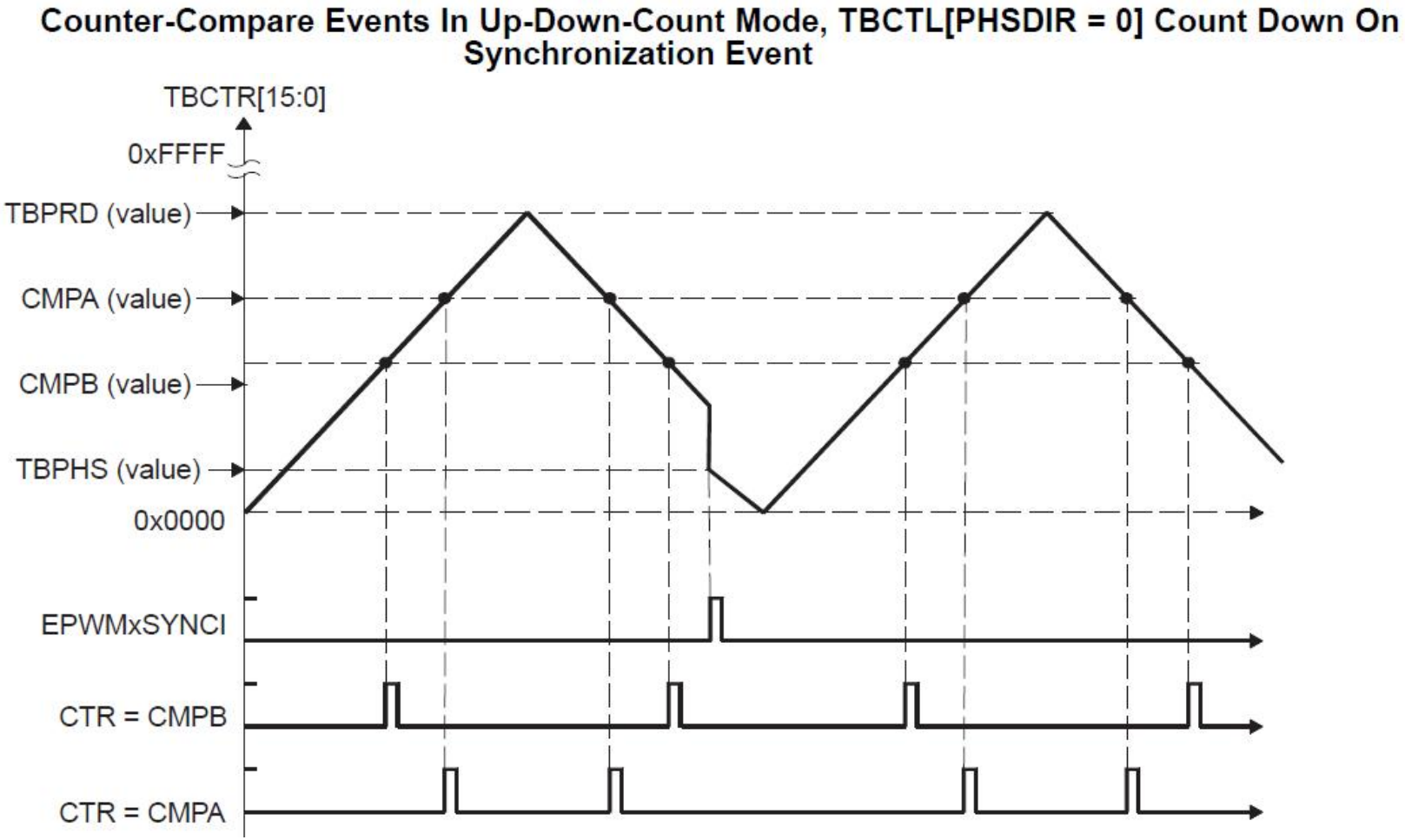
(2) 减计数模式：不对称PWM脉冲



2. CC子模块工作时序

比较功能子模块在下列3种计数模式下都可产生比较事件

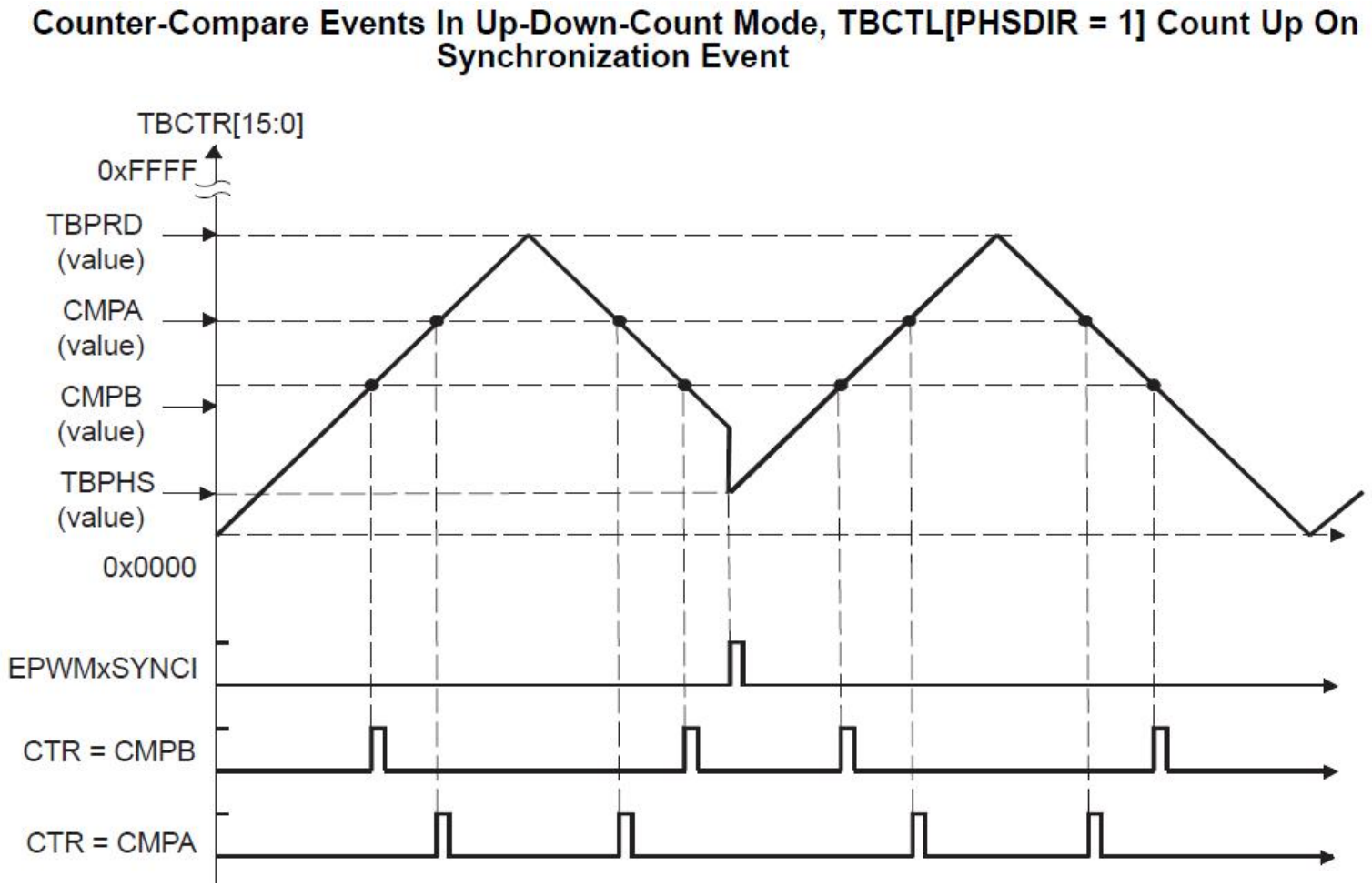
(3) 增减计数模式：对称PWM脉冲



2. CC子模块工作时序

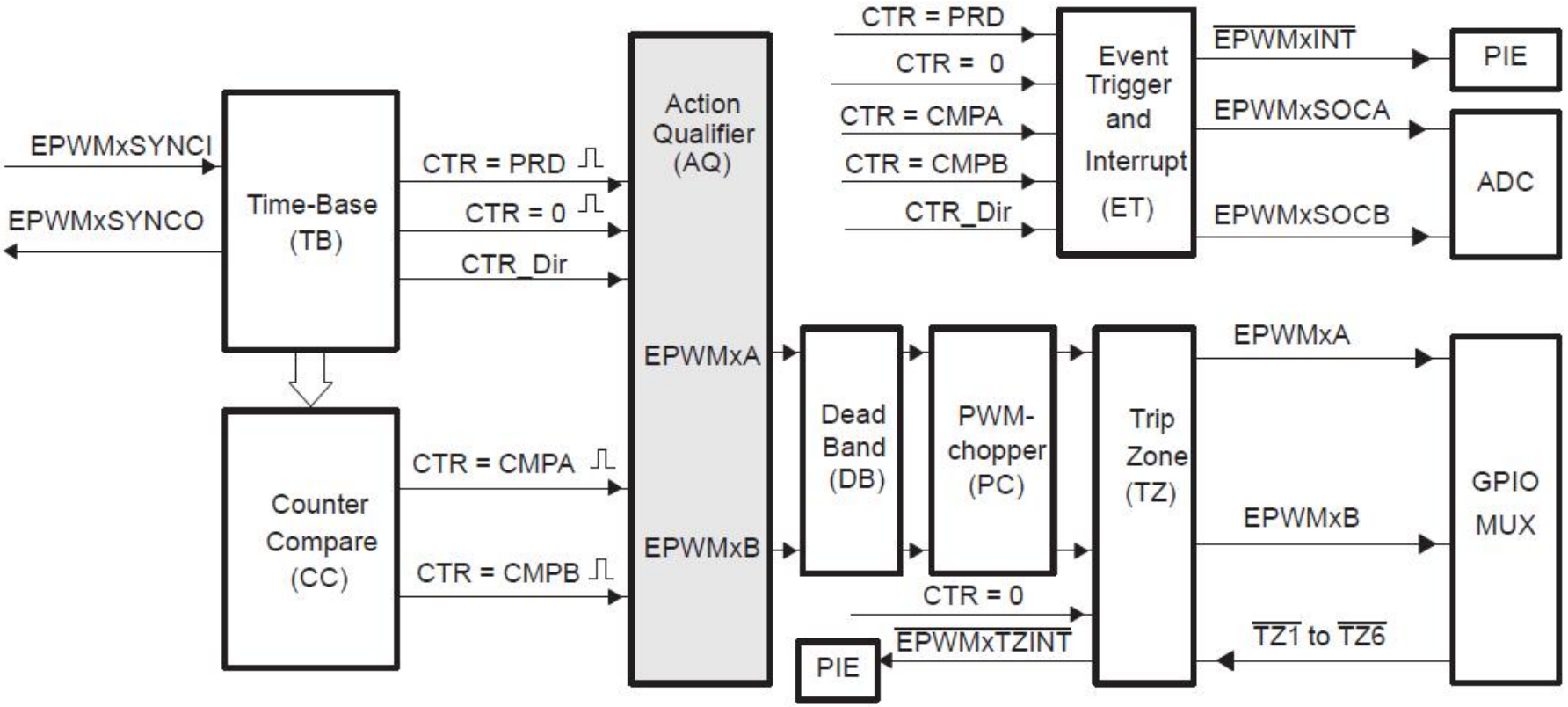
比较功能子模块在下列3种计数模式下都可产生比较事件

(3) 增减计数模式：对称PWM脉冲

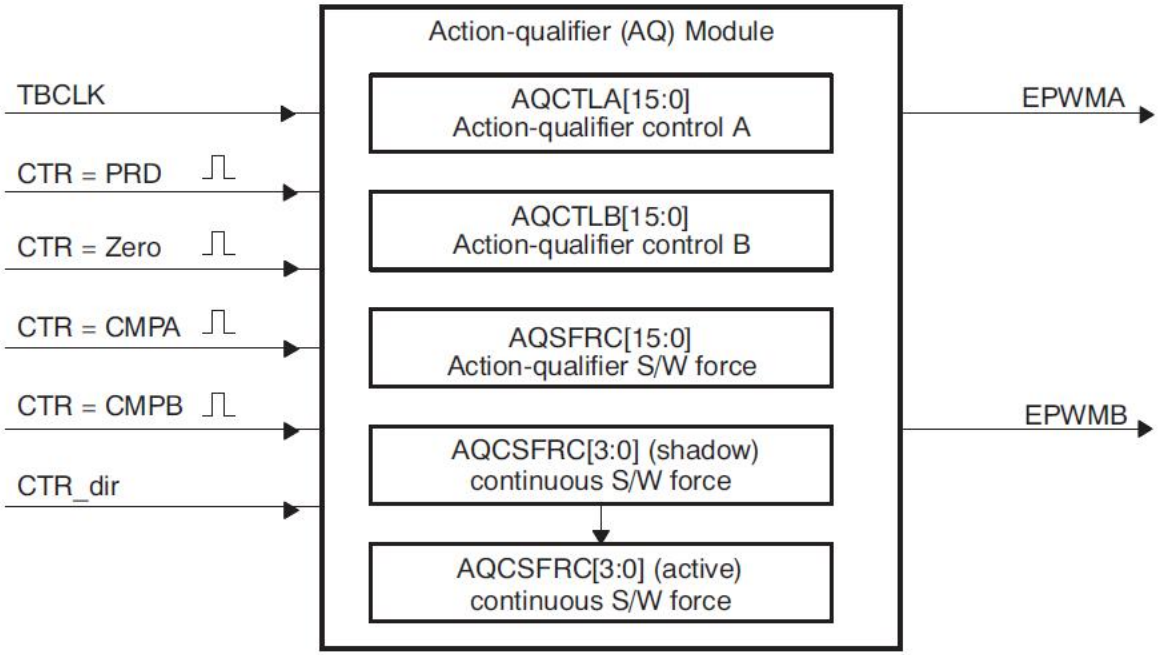


动作限定子模块

AQ子模块是ePWM最重要的模块，用来决定在特定事件发生时产生何种动作，形成需要的PWM脉冲



1. AQ子模块内部结构及功能



特定事件：

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

1. AQ子模块内部结构及功能

基于5种特定事件对EPWMxA和EPWMxB允许的操作如下：

（1）置高：将EPWMxA或EPWMxB的输出设定为高电平；

（2）置低：将EPWMxA或EPWMxB的输出设定为低电平；

（3）翻转：将EPWMxA或EPWMxB的输出状态翻转；

（4）无动作：保持EPWMxA或EPWMxB的输出状态不变。

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

1. AQ子模块内部结构及功能

对EPWMxA和EPWMxB的动作设定是完全独立的，
任何事件对EPWMxA或EPWMxB的任何一个都可以产生任何动作。

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

2. AQ子模块事件优先级

Action-Qualifier Event Priority for Up-Down-Count Mode

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

Action-Qualifier Event Priority for Up-Count Mode

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

2. AQ子模块事件优先级

Action-Qualifier Event Priority for Down-Count Mode

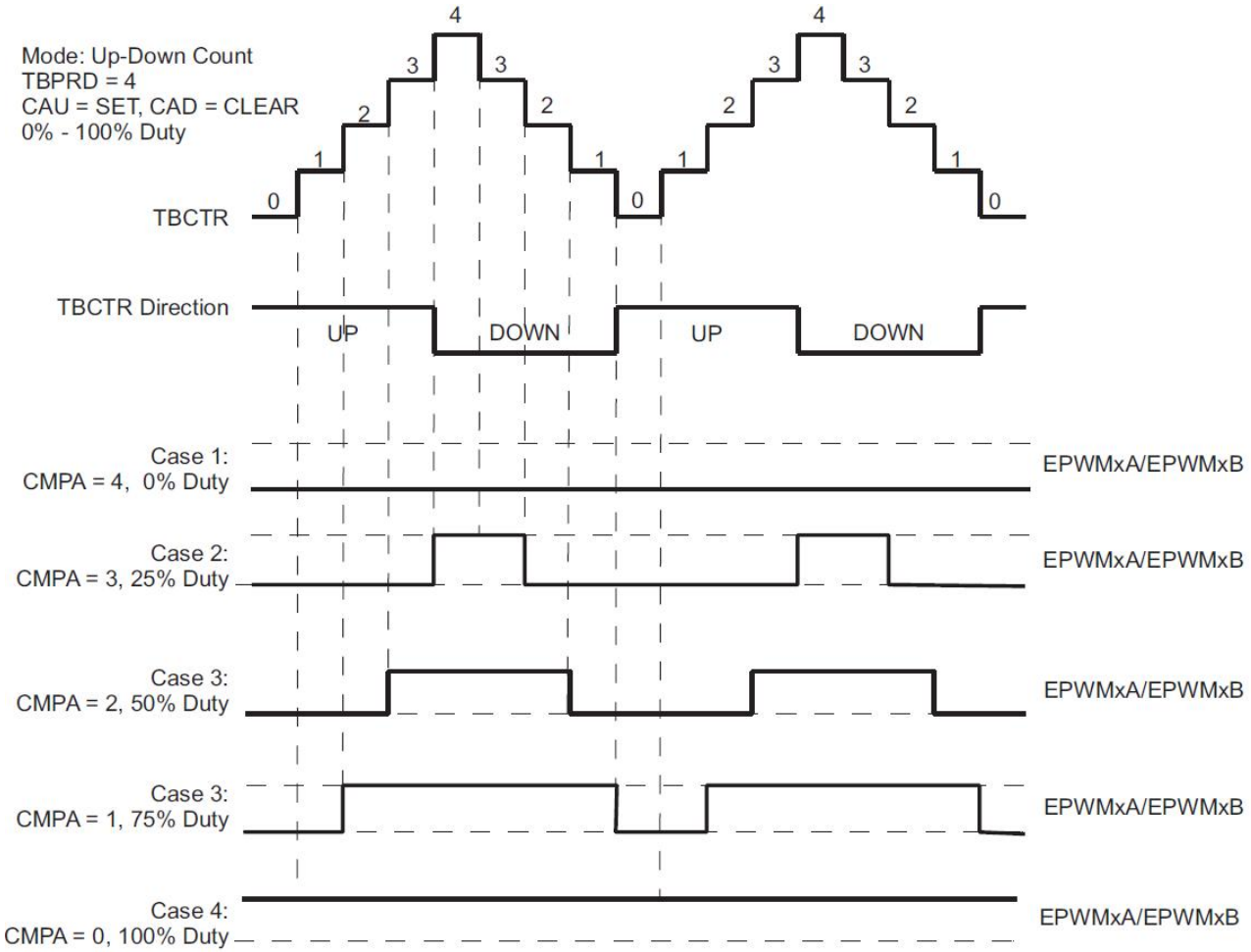
Priority Level	Event
1 (Highest)	Software forced event
2	Counter equal to Zero
3	Counter equal to CMPB on down-count (CBD)
4	Counter equal to CMPA on down-count (CAD)
5 (Lowest)	Counter equal to period (TBPRD)

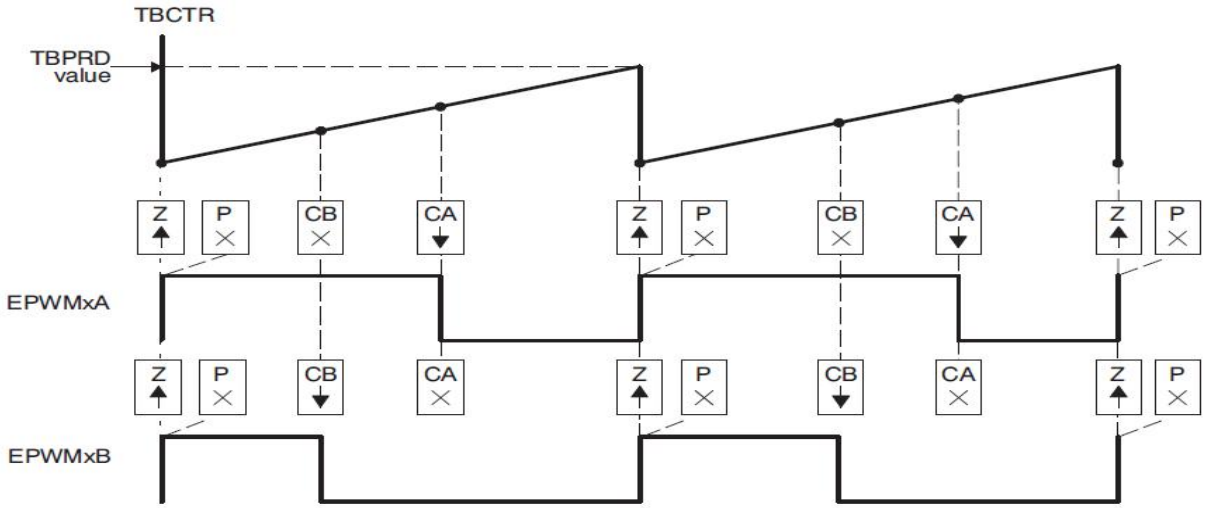
3. 常用波形配置

（1）使用增减计数模式产生对称PWM脉冲；

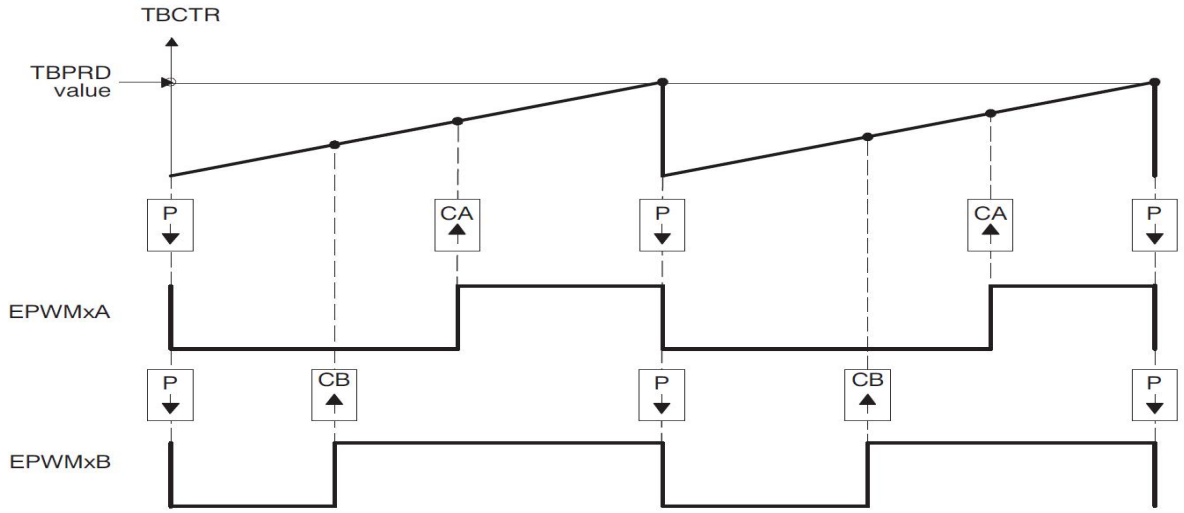
（2）使用增计数模式产生非对称PWM脉冲；

（3）使用减计数模式产生非对称PWM脉冲。

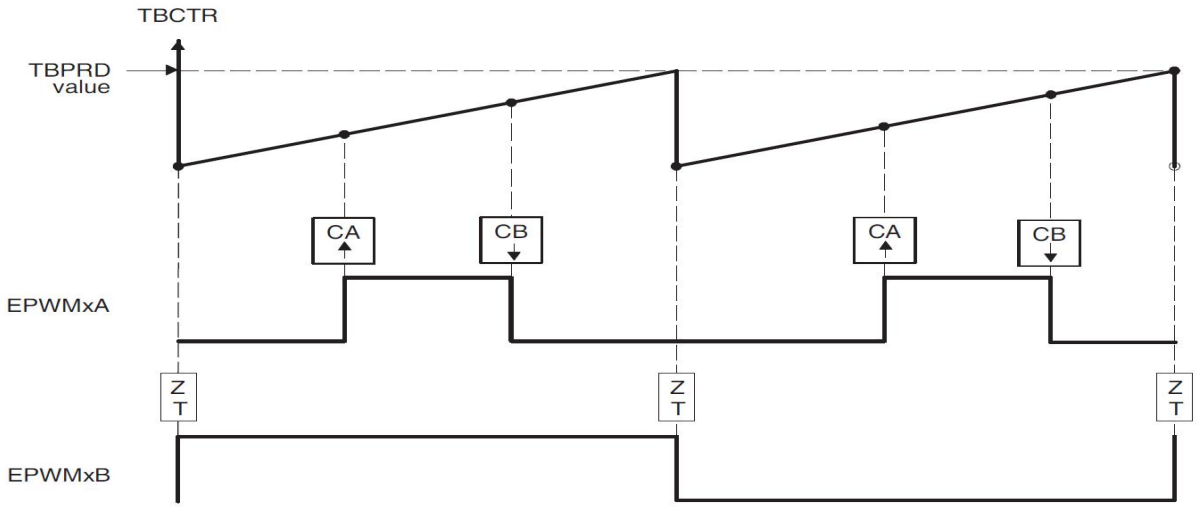




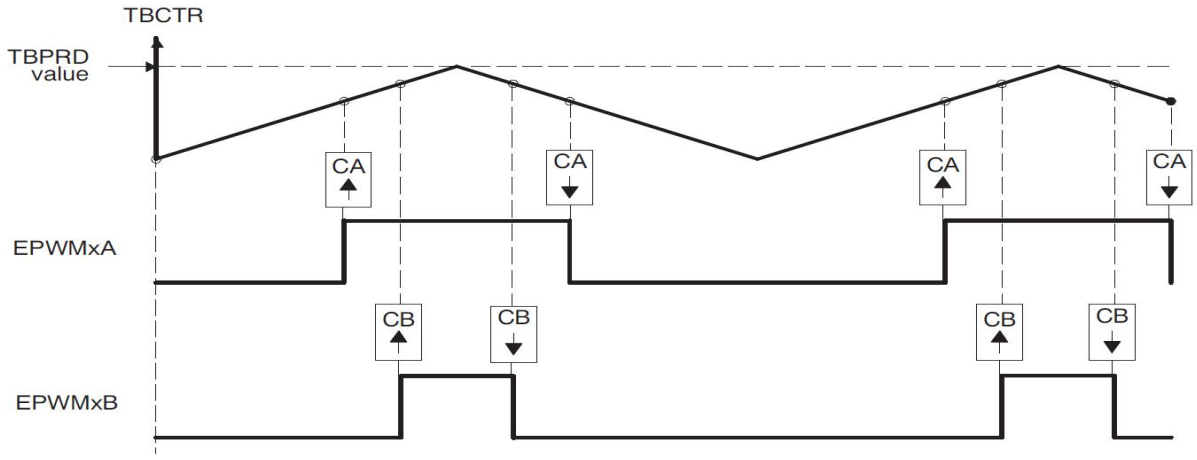

```
// Initialization Time
// = = = = = = = = = = = = = = = = = = = = = = = = = = = =
EPwm1Regs.TBPRD = 600;                          // Period = 601 TBCLK counts
EPwm1Regs.CMPA.half.CMPA = 350;                 // Compare A = 350 TBCLK counts
EPwm1Regs.CMPB = 200;                           // Compare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




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

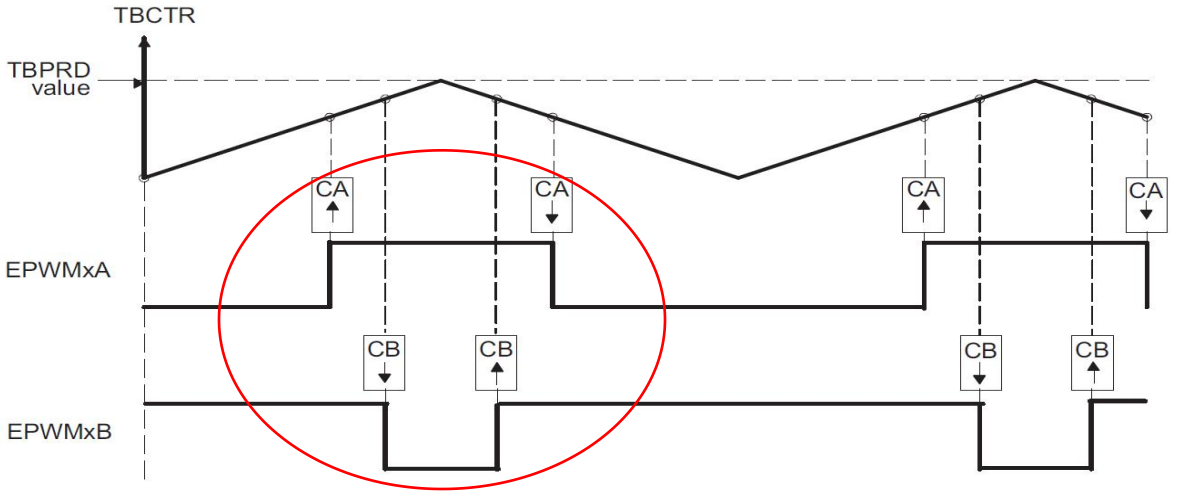



```
// Initialization Time
// = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
EPwm1Regs.TBPRD = 600;                          // Period = 601 TBCLK counts
EPwm1Regs.CMPA.half.CMPA = 200;                 // Compare A = 200 TBCLK counts
EPwm1Regs.CMPB = 400;                           // Compare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

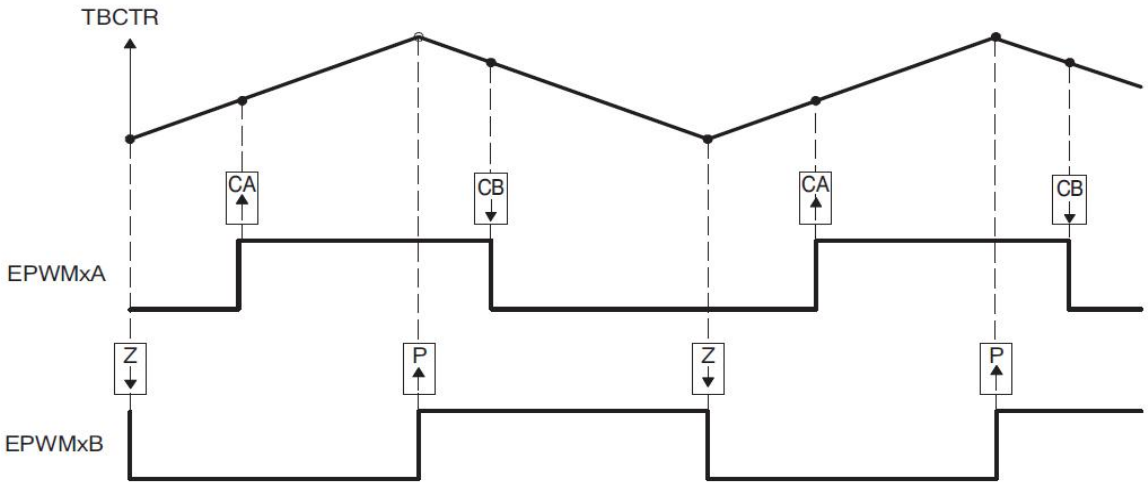



```
// Initialization Time
// = = = = = = = = = = = = = = = = = = = = = = = = = = = =
EPwm1Regs.TBPRD = 600;                          // Period = 2*600 TBCLK counts
EPwm1Regs.CMPA.half.CMPA = 400;                 // Compare A = 400 TBCLK counts
EPwm1Regs.CMPB = 500;                           // Compare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

互补，
可用于产生死区




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

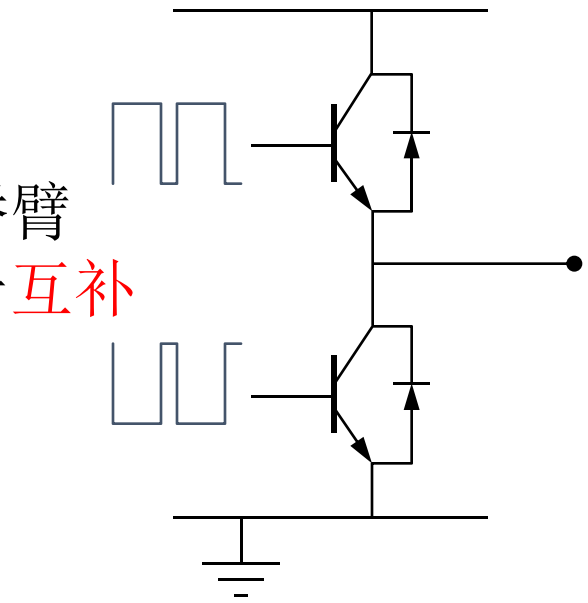


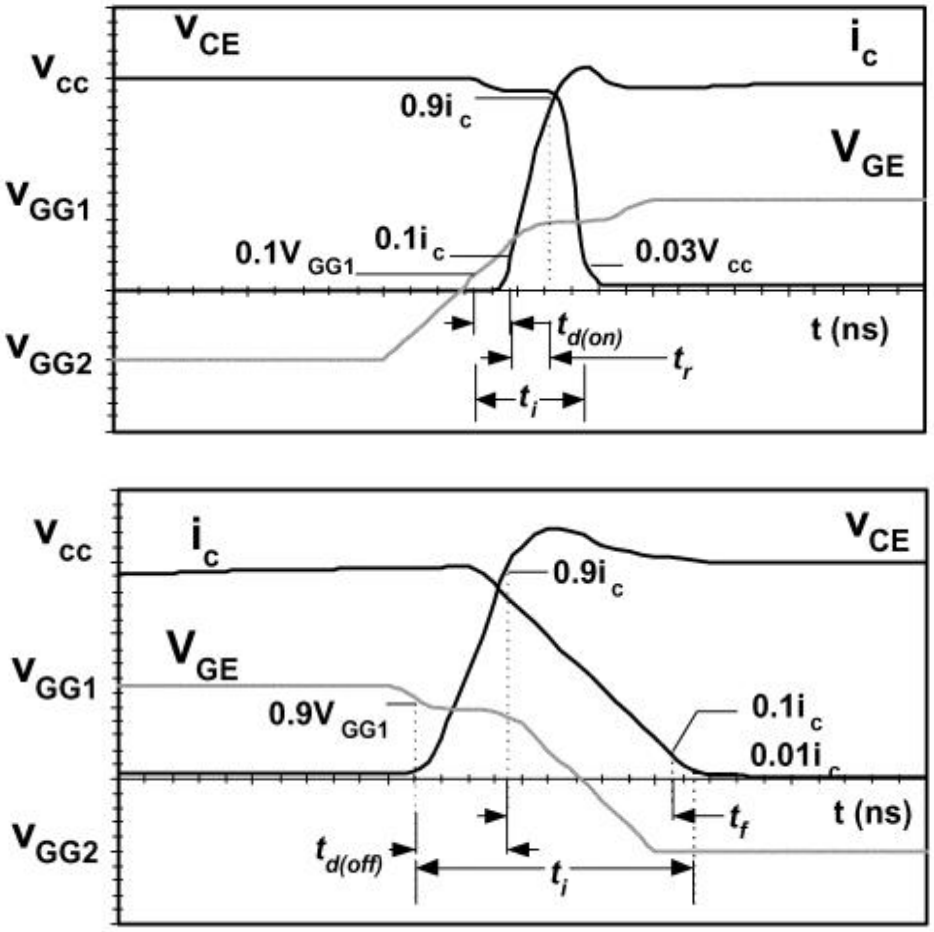

```
// Initialization Time
// = = = = = = = = = = = = = = = = = = = = = = = = = = = =
EPwm1Regs.TBPRD = 600;                          //  Period = 2 ´ 600 TBCLK counts
EPwm1Regs.CMPA.half.CMPA = 250;                 //  Compare A = 250 TBCLK counts
EPwm1Regs.CMPB = 450;                           // 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

死区产生子模块

为什么要使用PWM死区？

上下桥臂
开关信号互补

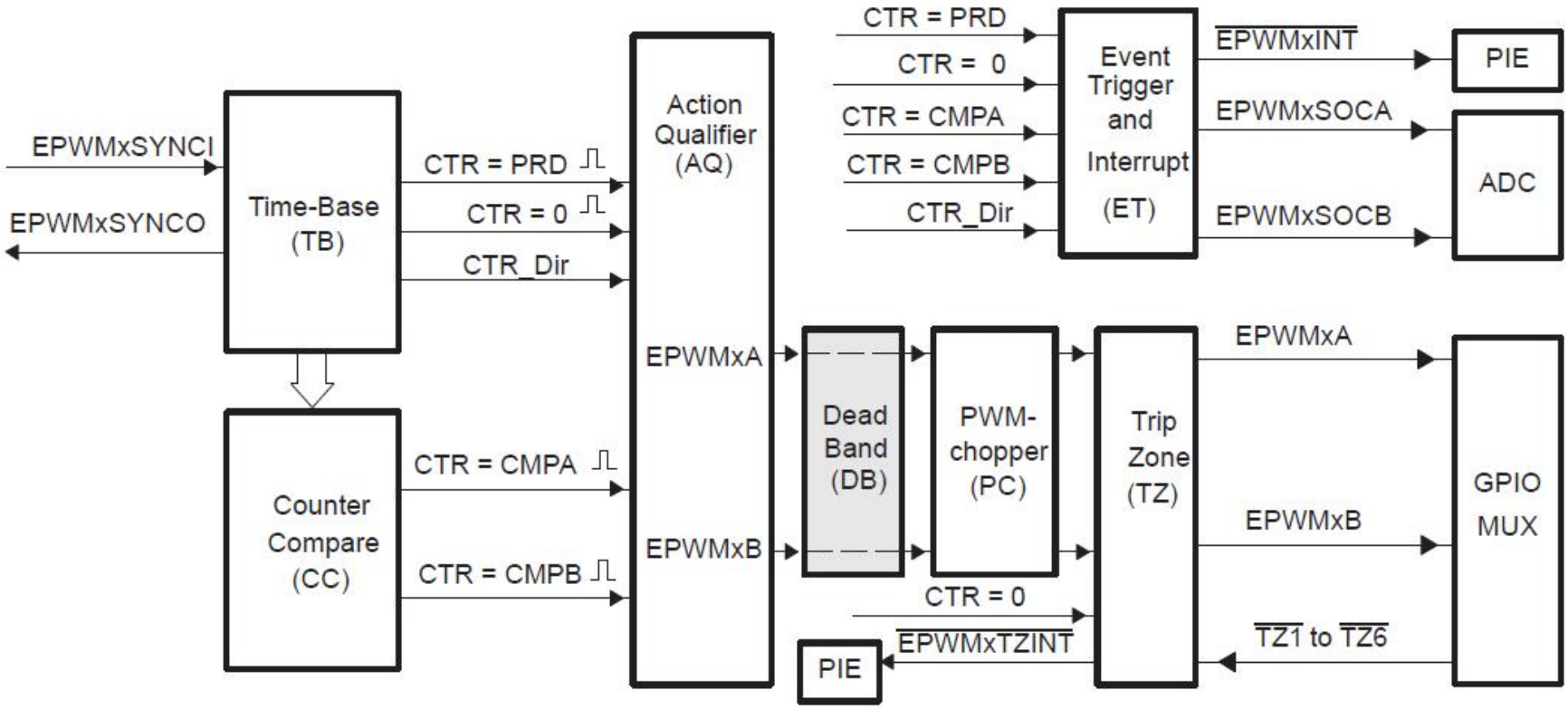




IGBT开通与关断曲线

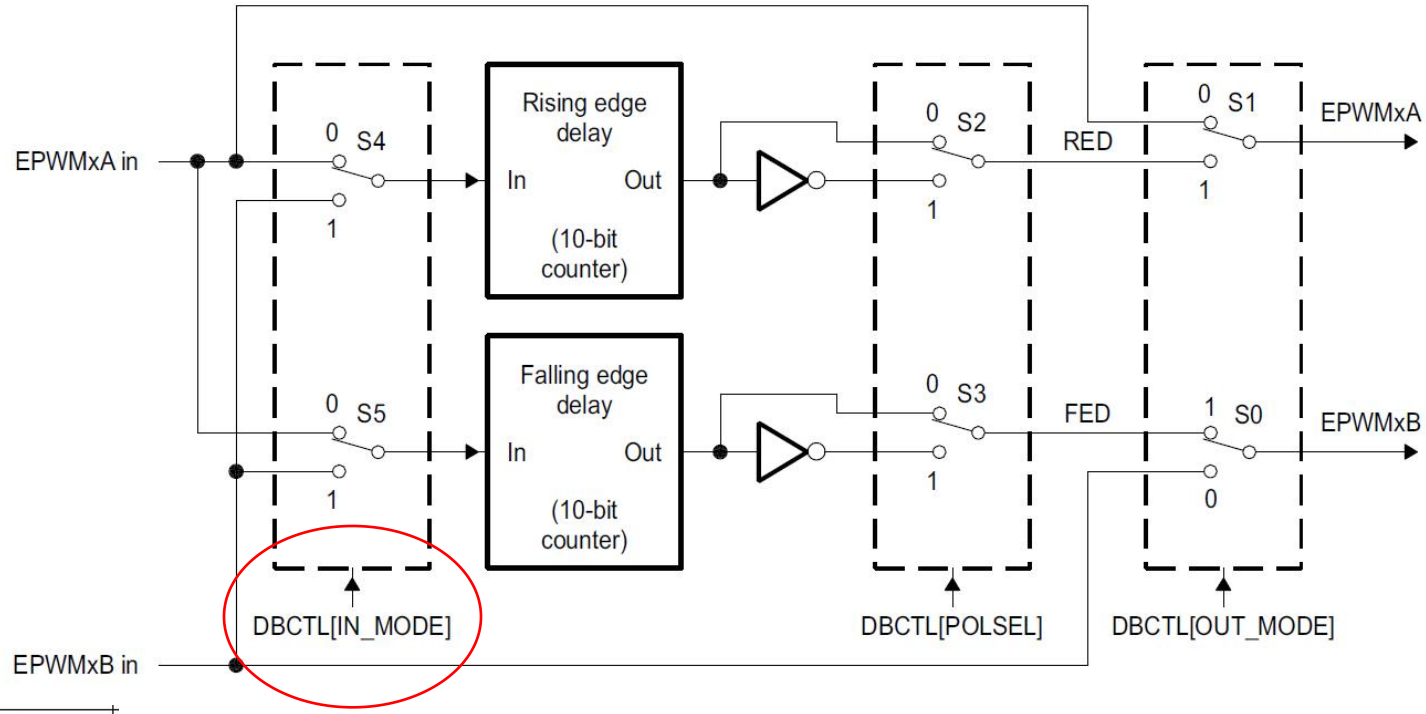
死区产生子模块

AQ子模块中通过设定COMPA和COPMB可产生简单死区。如需严格控制死区的边沿和极性，则需要DB子模块。



1. DB子模块工作过程

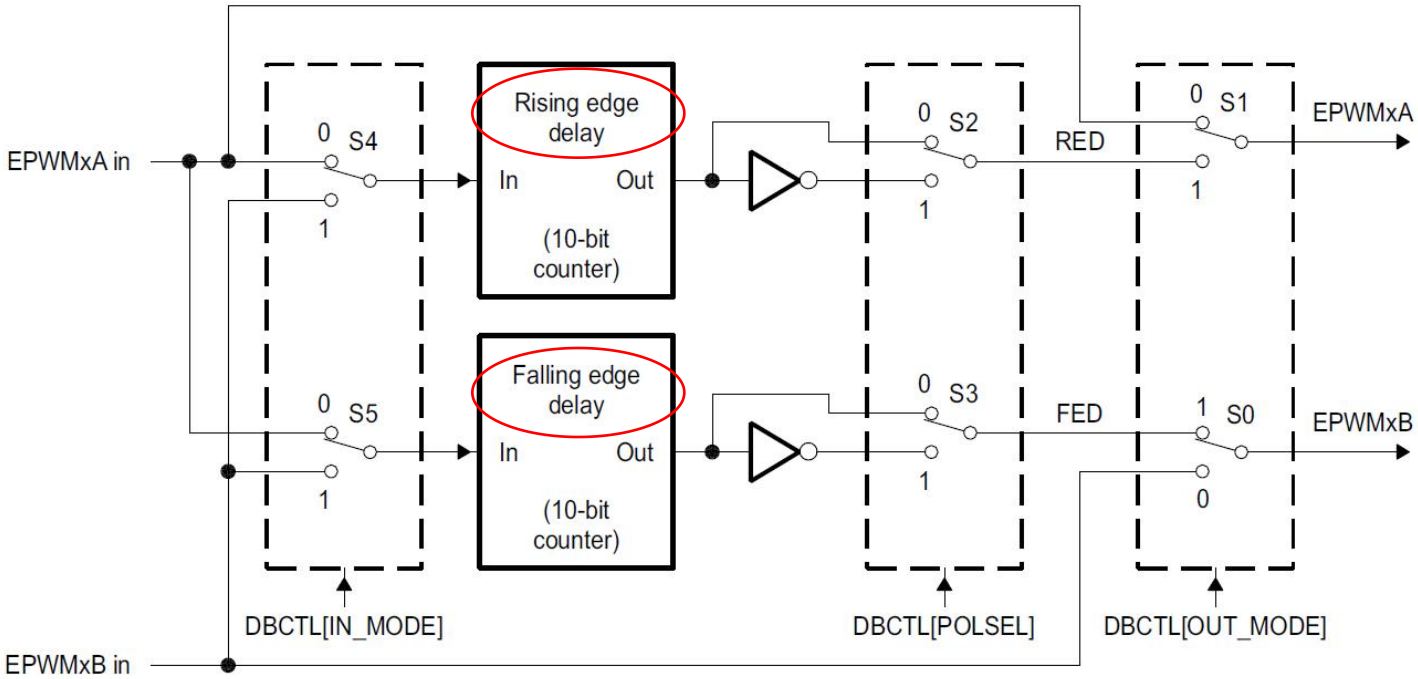
（1）输入源选择

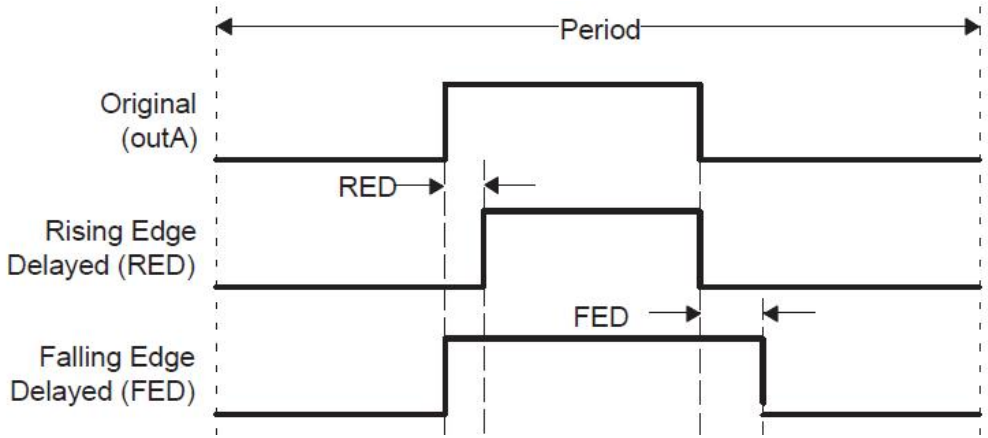


5-4	IN_MODE	00	EPWMxA In (from the action-qualifier) is the source for both falling-edge and rising-edge delay.
		01	EPWMxB In (from the action-qualifier) is the source for rising-edge delayed signal. EPWMxA In (from the action-qualifier) is the source for falling-edge delayed signal.
		10	EPWMxA In (from the action-qualifier) is the source for rising-edge delayed signal. EPWMxB In (from the action-qualifier) is the source for falling-edge delayed signal.
		11	EPWMxB In (from the action-qualifier) is the source for both rising-edge delay and falling-edge delayed signal.

1. DB子模块工作过程

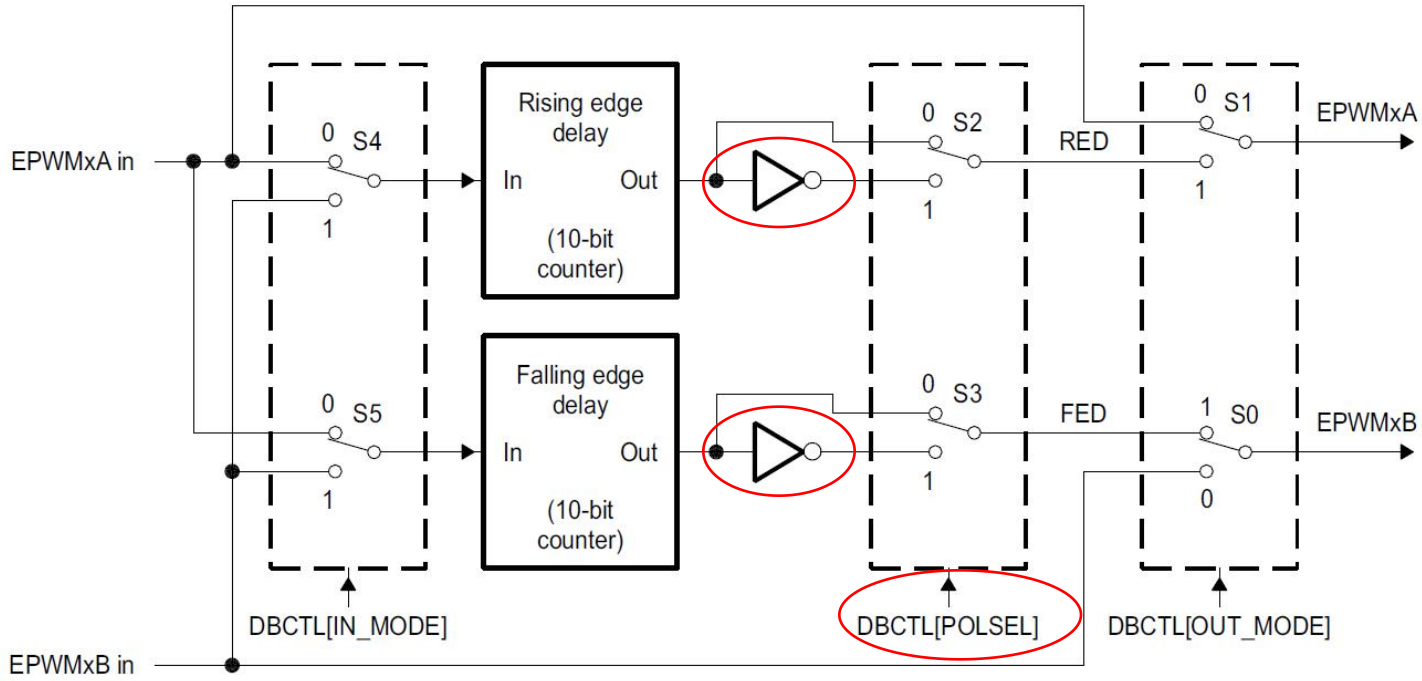
（2）上升沿/下降沿延时





1. DB子模块工作过程

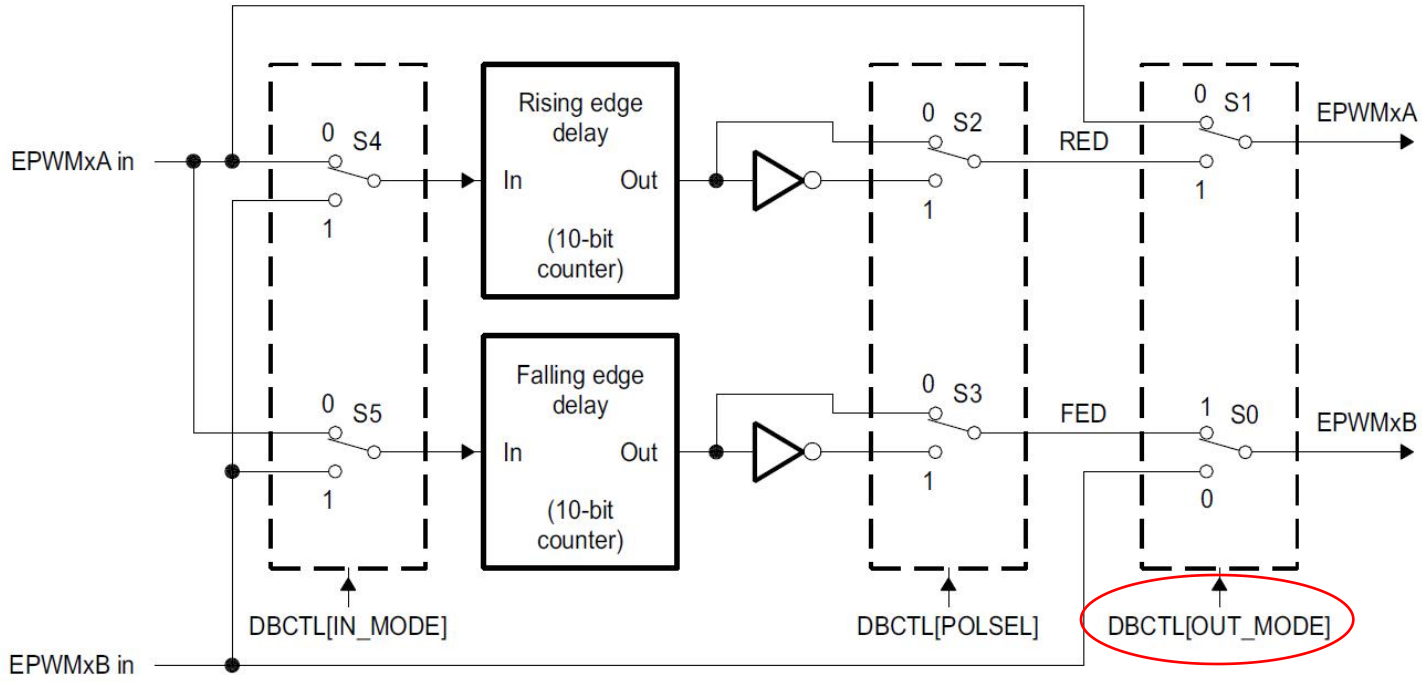
（3）极性选择



3-2	POLSEL		These assume that DBCTL[OUT_MODE] = 1,1 and DBCTL[IN_MODE] = 0,0. Other enhanced modes are also possible, but not regarded as typical usage modes.
		00	Active high (AH) mode. Neither EPWMxA nor EPWMxB is inverted (default).
		01	Active low complementary (ALC) mode. EPWMxA is inverted.
		10	Active high complementary (AHC). EPWMxB is inverted.
		11	Active low (AL) mode. Both EPWMxA and EPWMxB are inverted.

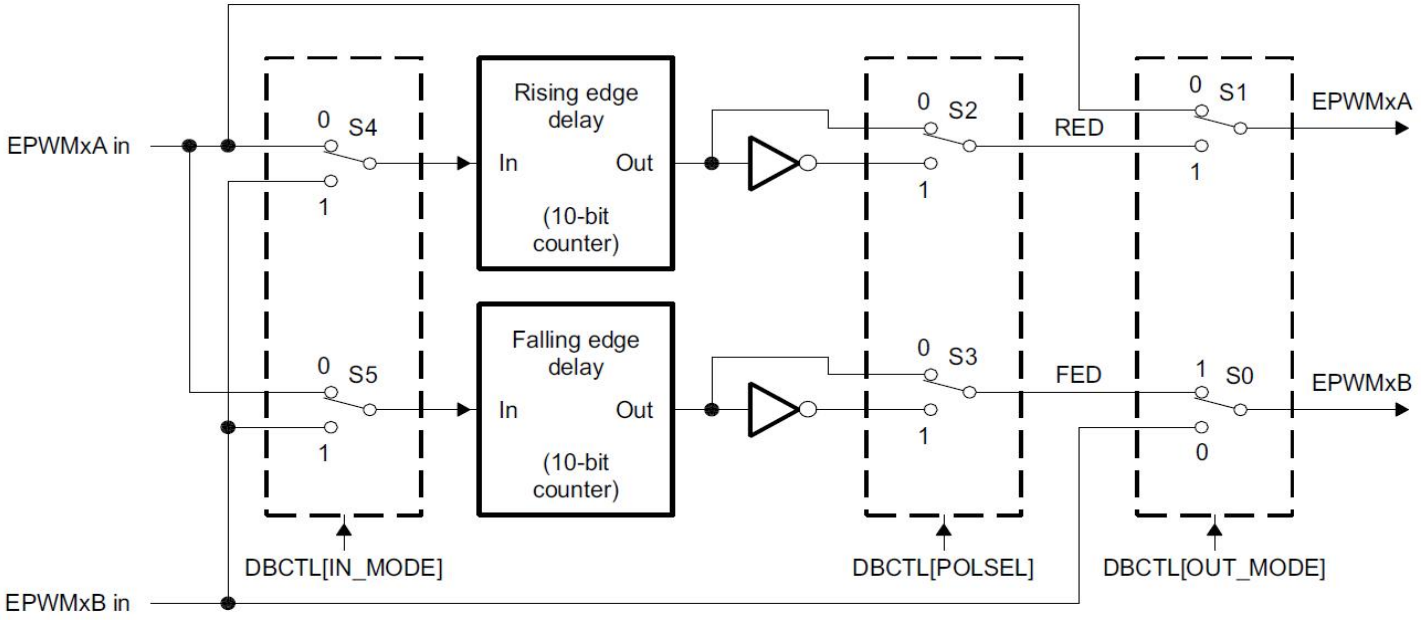
1. DB子模块工作过程

（4）输出模式选择



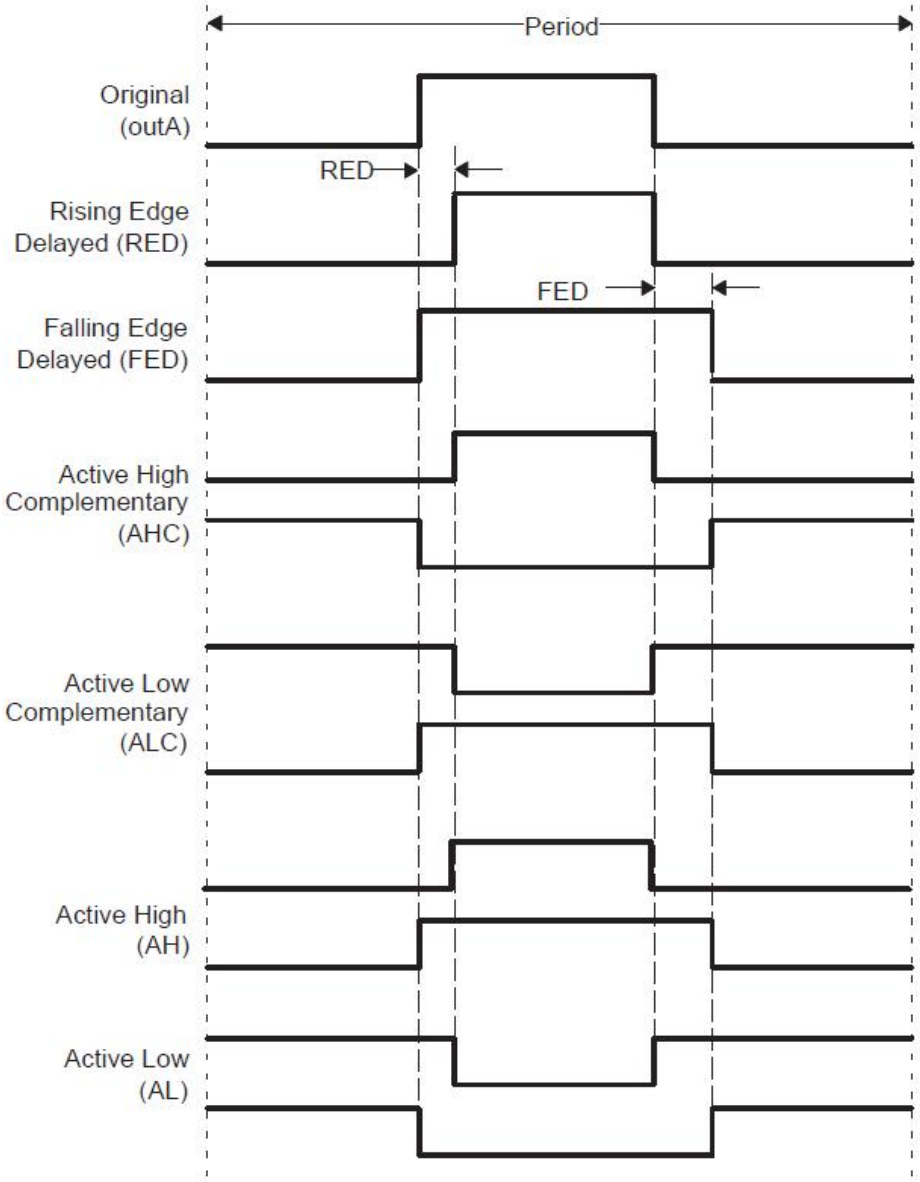
1-0	OUT_MODE		Dead-band Output Mode Control Bit 1 controls the S1 switch and bit 0 controls the S0 switch shown in Figure This allows you to selectively enable or bypass the dead-band generation for the falling-edge and rising-edge delay.

2. 典型死区方案的实现



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

2. 典型死区方案的实现


Period
Original
(outA)
RED
Rising Edge
Delayed (RED)
FED
Falling Edge
Delayed (FED)
Active High
Complementary
(AHC)
Active Low
Complementary
(ALC)
Active High
(AH)
Active Low
(AL)

3. 死区时间的计算

DB子模块通过DBRED和DBFED寄存器对上升沿和下降沿配置延时时间。

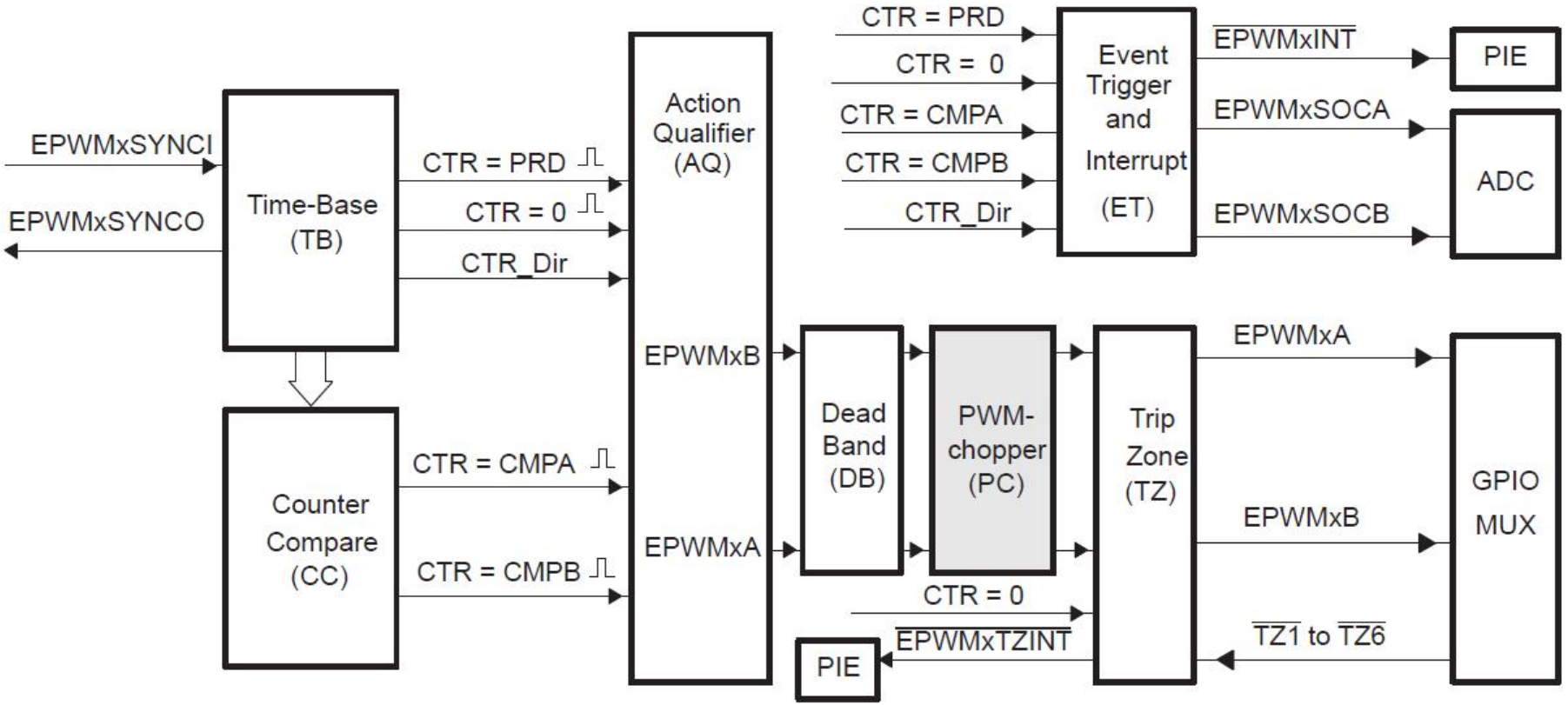
上升沿延迟(RED) = T_{TBCLK} x DBRED

下降沿延迟(FED) = T_{TBCLK} x DBFED

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

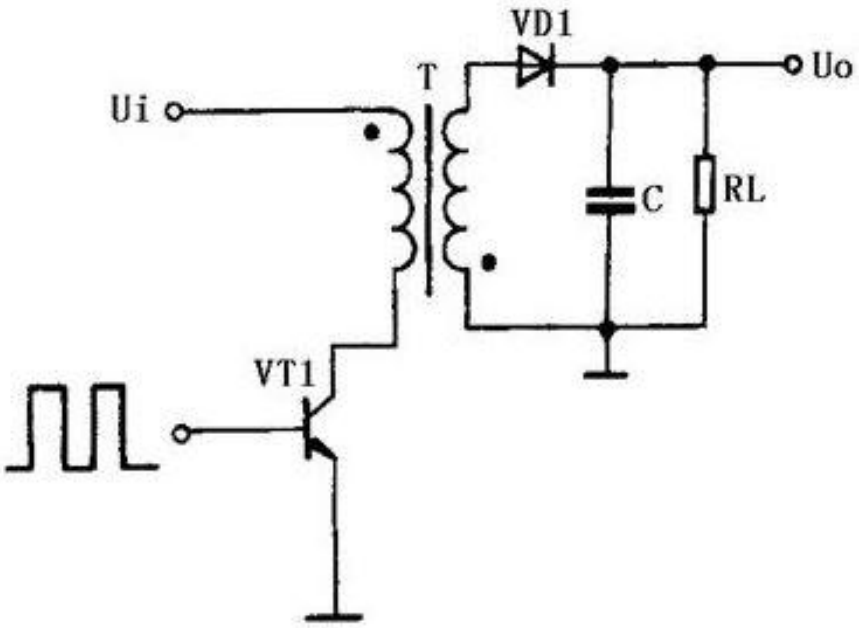
斩波控制子模块

1. 需要MHz以上的调制波来驱动开关电源电路中的高频变压器；
2. 基于高频变压器的驱动电路。



斩波控制子模块

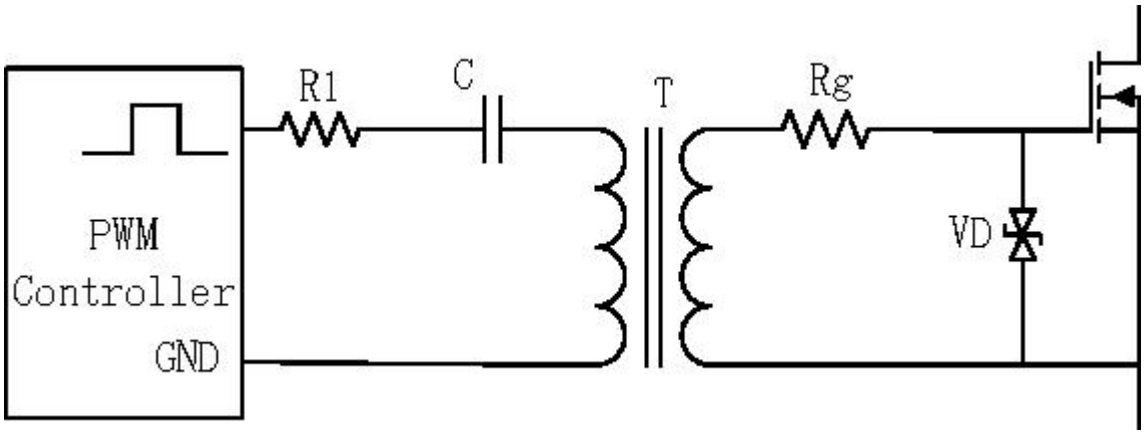
1. 需要MHz以上的调制波来驱动开关电源电路中的高频变压器；



单端反激式开关电源

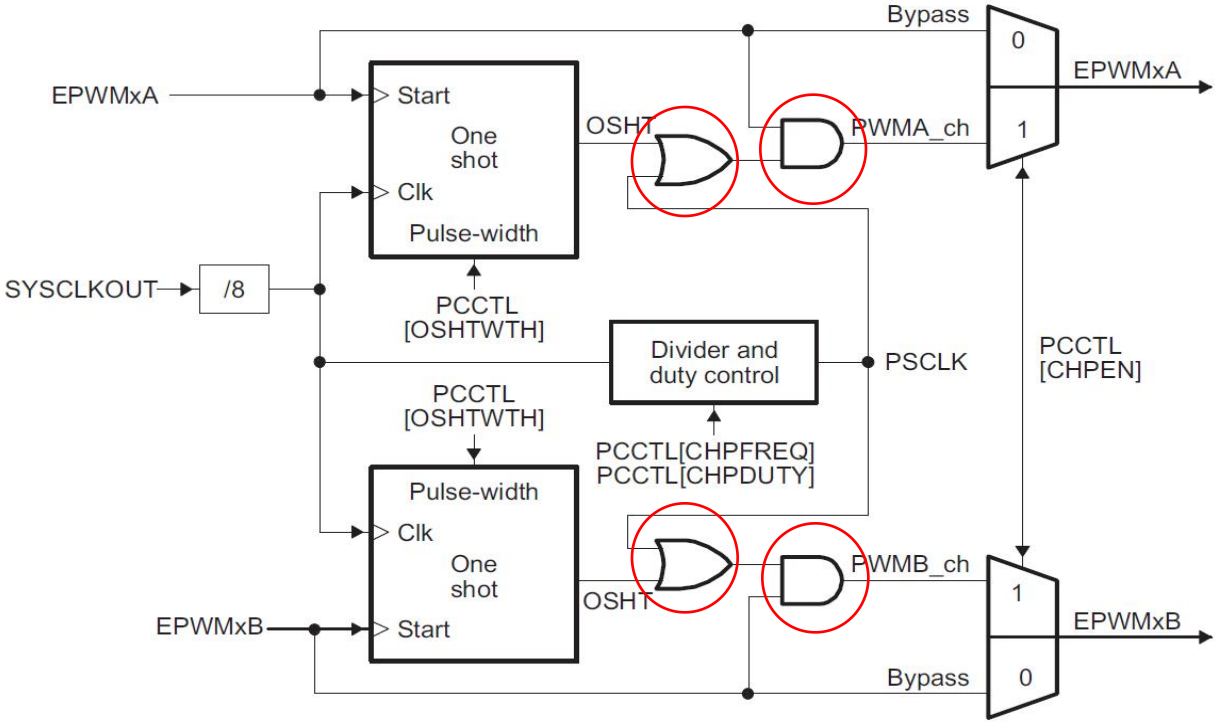
斩波控制子模块

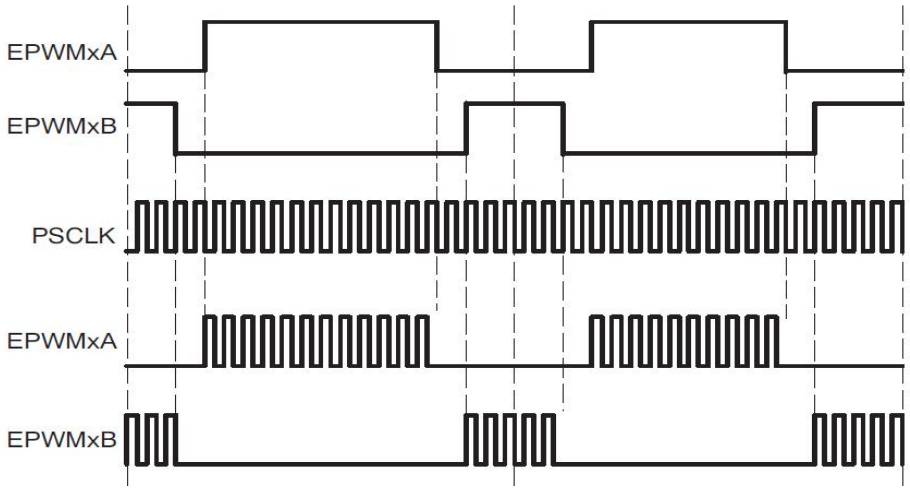
2. 基于高频变压器的驱动电路。



基于高频变压器的驱动电路

1. PC子模块工作过程

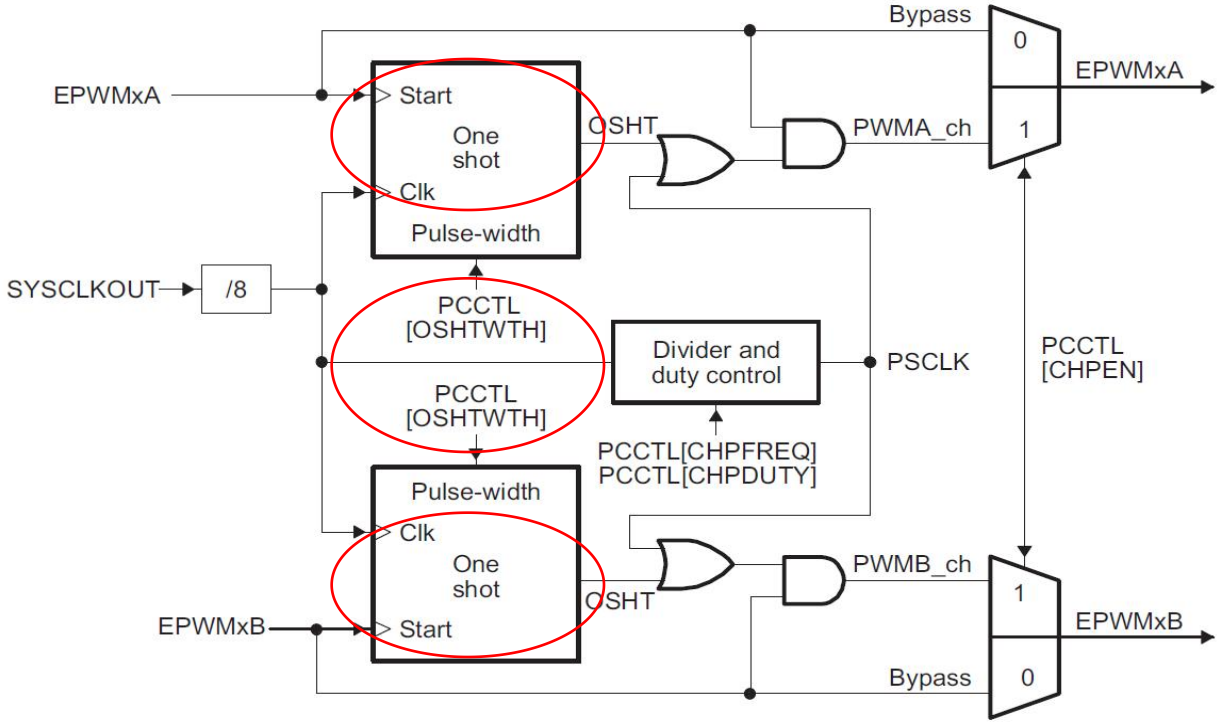

Bypass
0
EPWMxA
EPWMxA
Start
One
shot
Clk
Pulse-width
OSHT
PWMA_ch
1
SYSCLKOUT
/8
PCCTL
[OSHTWTH]
Divider and
duty control
PSCLK
PCCTL
[CHPEN]
PCCTL
[OSHTWTH]
PCCTL[CHPFREQ]
PCCTL[CHPDUTY]
Pulse-width
Clk
One
shot
Start
EPWMxB
OSHT
PWMB_ch
1
EPWMxB
Bypass
0


EPWMxA
EPWMxB
PSCLK
EPWMxA
EPWMxB

2. 首次脉冲宽度配置

首次脉冲单元产生携带较大能量的第一个脉冲，确保功率器件可靠开通。

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



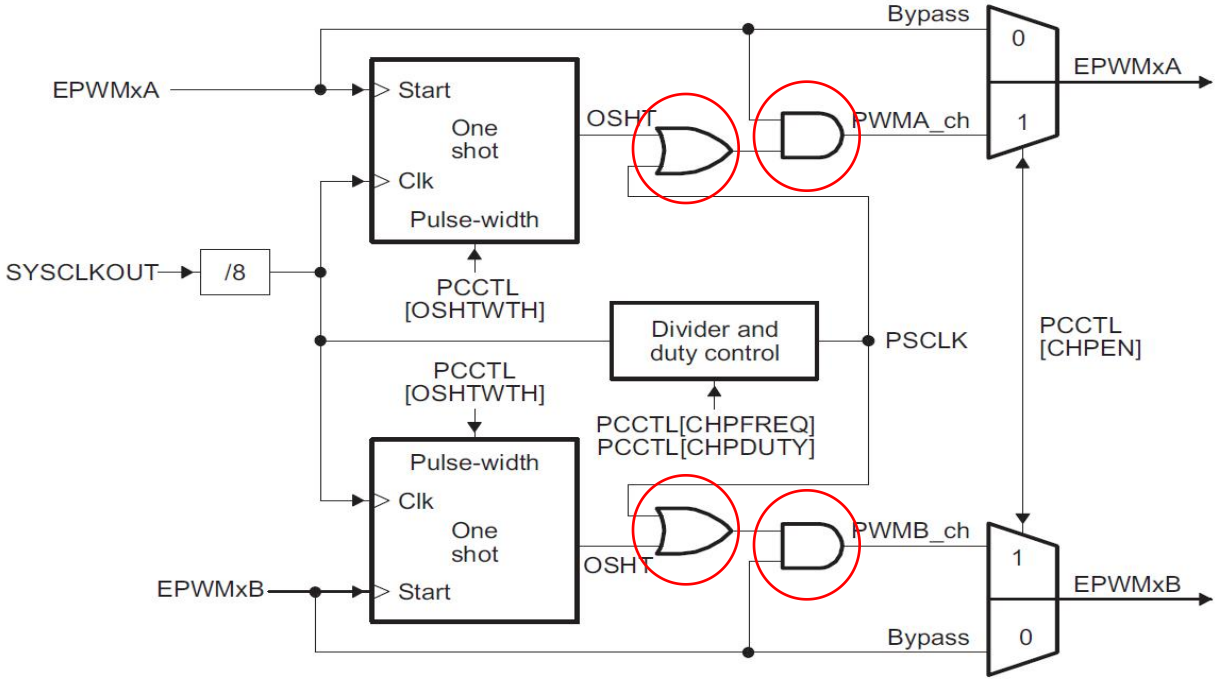
2. 首次脉冲宽度配置

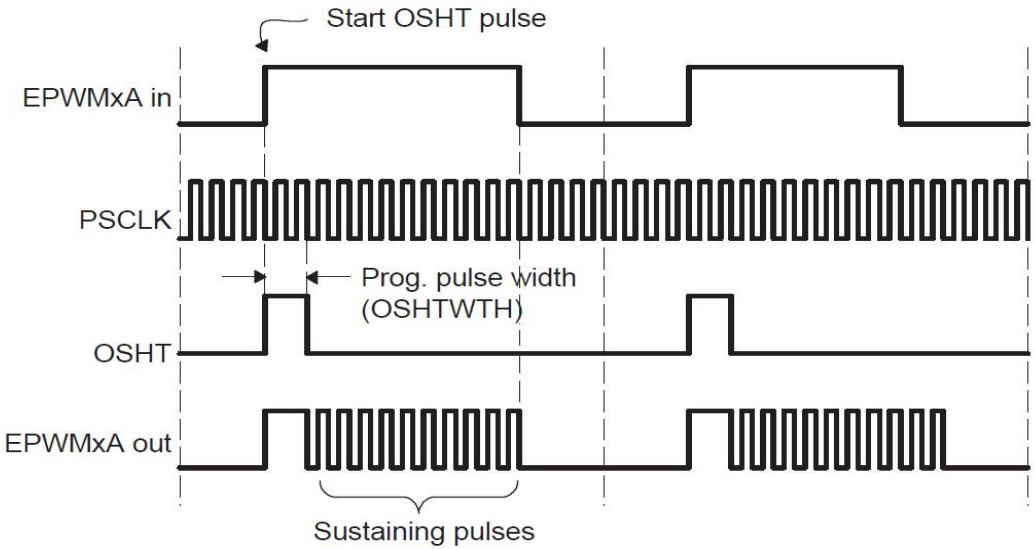
首次脉冲单元产生携带较大能量的第一个脉冲，确保功率器件可靠开通。

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

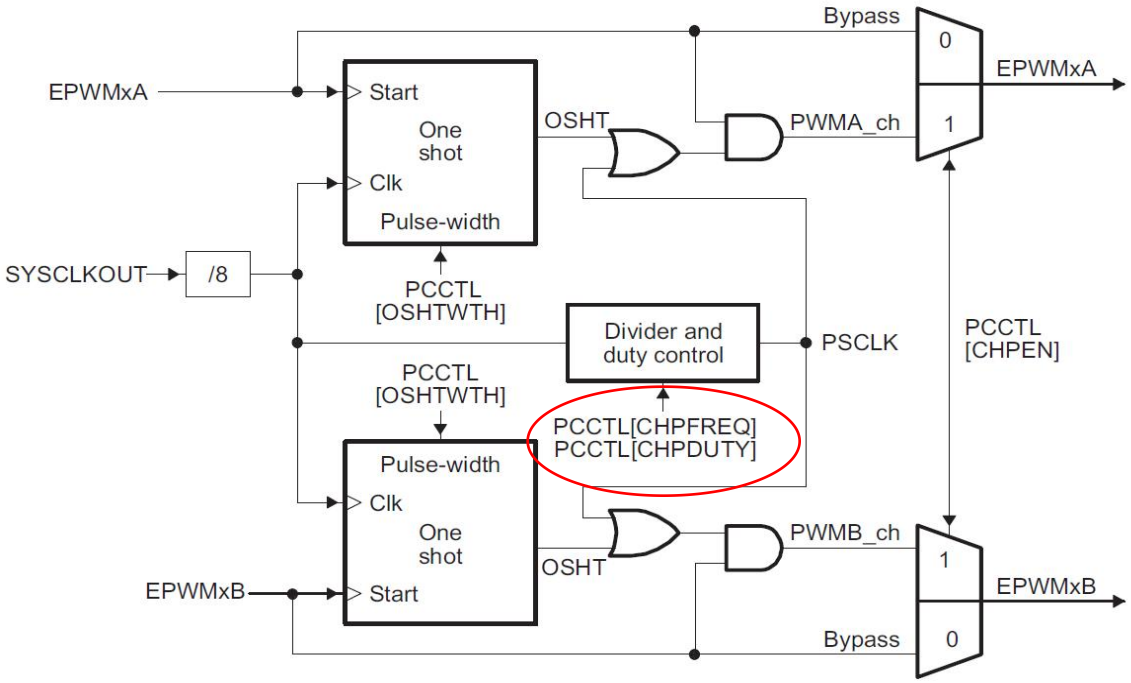
4:1	OSHTWTH		One-Shot Pulse Width
		0000	1 x SYSCLKOUT / 8 wide (= 80 nS at 100 MHz SYSCLKOUT)
		0001	2 x SYSCLKOUT / 8 wide (= 160 nS at 100 MHz SYSCLKOUT)
		0010	3 x SYSCLKOUT / 8 wide (= 240 nS at 100 MHz SYSCLKOUT)
		0011	4 x SYSCLKOUT / 8 wide (= 320 nS at 100 MHz SYSCLKOUT)
		0100	5 x SYSCLKOUT / 8 wide (= 400 nS at 100 MHz SYSCLKOUT)
		0101	6 x SYSCLKOUT / 8 wide (= 480 nS at 100 MHz SYSCLKOUT)
		0110	7 x SYSCLKOUT / 8 wide (= 560 nS at 100 MHz SYSCLKOUT)
		0111	8 x SYSCLKOUT / 8 wide (= 640 nS at 100 MHz SYSCLKOUT)
		1000	9 x SYSCLKOUT / 8 wide (= 720 nS at 100 MHz SYSCLKOUT)
		1001	10 x SYSCLKOUT / 8 wide (= 800 nS at 100 MHz SYSCLKOUT)
		1010	11 x SYSCLKOUT / 8 wide (= 880 nS at 100 MHz SYSCLKOUT)
		1011	12 x SYSCLKOUT / 8 wide (= 960 nS at 100 MHz SYSCLKOUT)
		1100	13 x SYSCLKOUT / 8 wide (= 1040 nS at 100 MHz SYSCLKOUT)
		1101	14 x SYSCLKOUT / 8 wide (= 1120 nS at 100 MHz SYSCLKOUT)
		1110	15 x SYSCLKOUT / 8 wide (= 1200 nS at 100 MHz SYSCLKOUT)
		1111	16 x SYSCLKOUT / 8 wide (= 1280 nS at 100 MHz SYSCLKOUT)

2. 首次脉冲宽度配置


Bypass
0
EPWMxA
EPWMxA
Start
One
shot
Clk
Pulse-width
OSHT
PWMA_ch
1
SYSCLKOUT
/8
PCCTL
[OSHTWTH]
Divider and
duty control
PSCLK
PCCTL
[CHPEN]
PCCTL
[OSHTWTH]
PCCTL[CHPFREQ]
PCCTL[CHPDUTY]
Pulse-width
Clk
One
shot
PWMB_ch
1
OSHT
EPWMxB
Start
EPWMxB
Bypass
0


Start OSHT pulse
EPWMxA in
PSCLK
Prog. pulse width
(OSHTWTH)
OSHT
EPWMxA out
Sustaining pulses

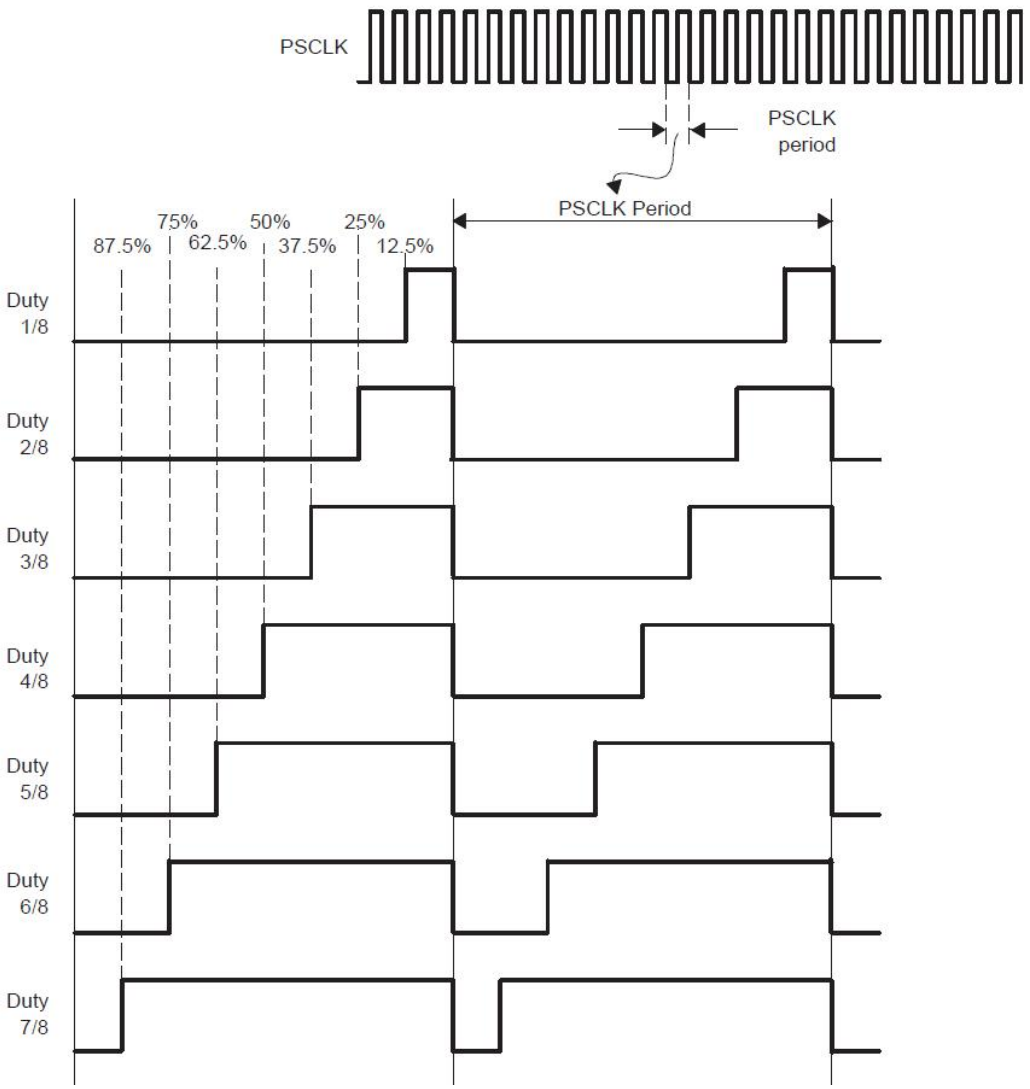
3. 占空比控制



10-8	CHPDUTY		Chopping Clock Duty Cycle
		000	Duty = 1/8 (12.5%)
		001	Duty = 2/8 (25.0%)
		010	Duty = 3/8 (37.5%)
		011	Duty = 4/8 (50.0%)
		100	Duty = 5/8 (62.5%)
		101	Duty = 6/8 (75.0%)
		110	Duty = 7/8 (87.5%)
		111	Reserved

7:5	CHPFREQ		Chopping Clock Frequency
		000	Divide by 1 (no prescale, = 12.5 MHz at 100 MHz SYSCLKOUT)
		001	Divide by 2 (6.25 MHz at 100 MHz SYSCLKOUT)
		010	Divide by 3 (4.16 MHz at 100 MHz SYSCLKOUT)
		011	Divide by 4 (3.12 MHz at 100 MHz SYSCLKOUT)
		100	Divide by 5 (2.50 MHz at 100 MHz SYSCLKOUT)
		101	Divide by 6 (2.08 MHz at 100 MHz SYSCLKOUT)
		110	Divide by 7 (1.78 MHz at 100 MHz SYSCLKOUT)
		111	Divide by 8 (1.56 MHz at 100 MHz SYSCLKOUT)

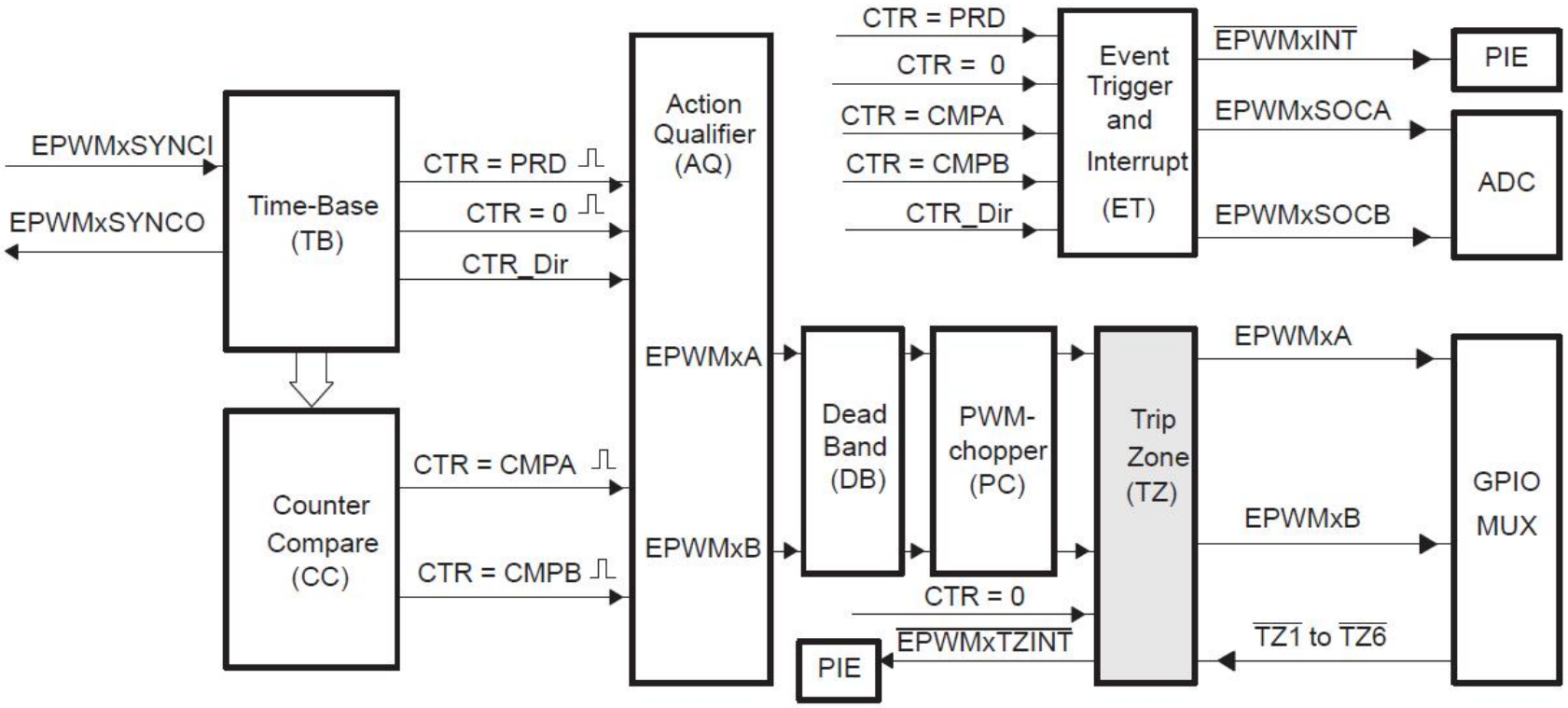
3. 占空比控制



10-8	CHPDUTY		Chopping Clock Duty Cycle
		000	Duty = 1/8 (12.5%)
		001	Duty = 2/8 (25.0%)
		010	Duty = 3/8 (37.5%)
		011	Duty = 4/8 (50.0%)
		100	Duty = 5/8 (62.5%)
		101	Duty = 6/8 (75.0%)
		110	Duty = 7/8 (87.5%)
		111	Reserved

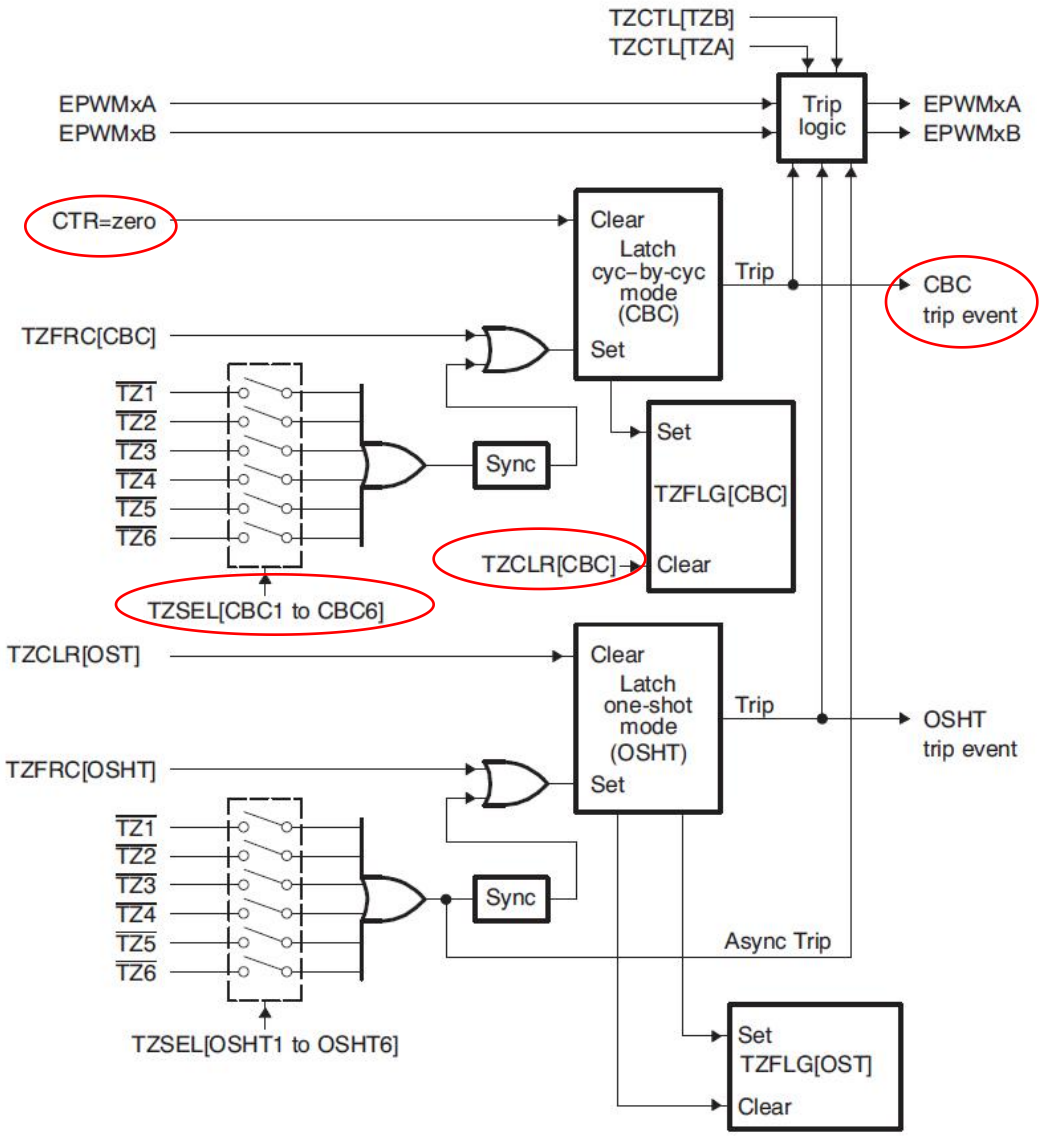
故障捕获子模块

1. 每个ePWM模块都与通过GPIO的6路触发信号 $\overline{TZn}$ 相连接；
2. 外部触发信号$\overline{TZn}$是低电平有效的触发信号。



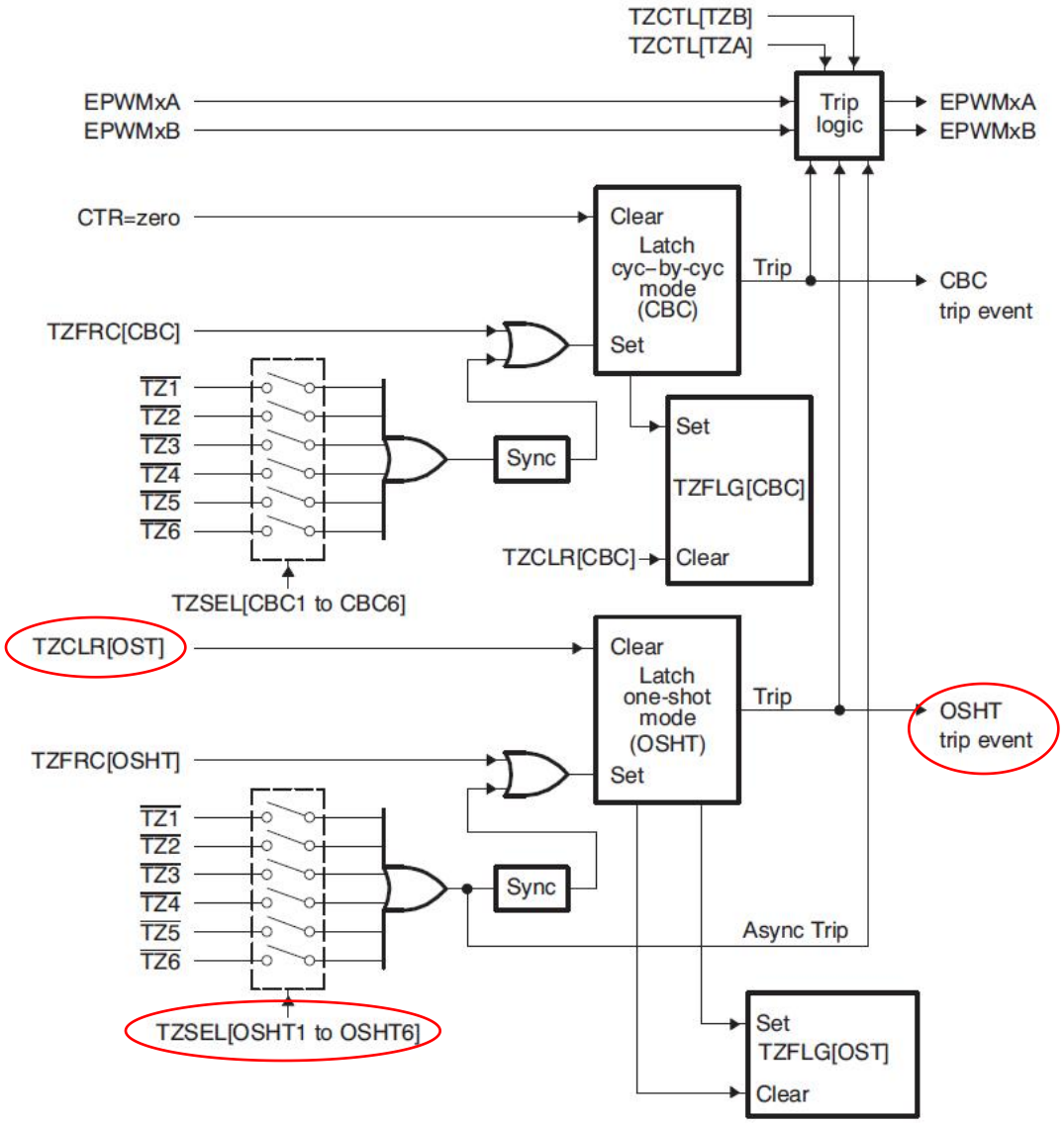
1. TZ子模块工作过程

（1）周期触发事件

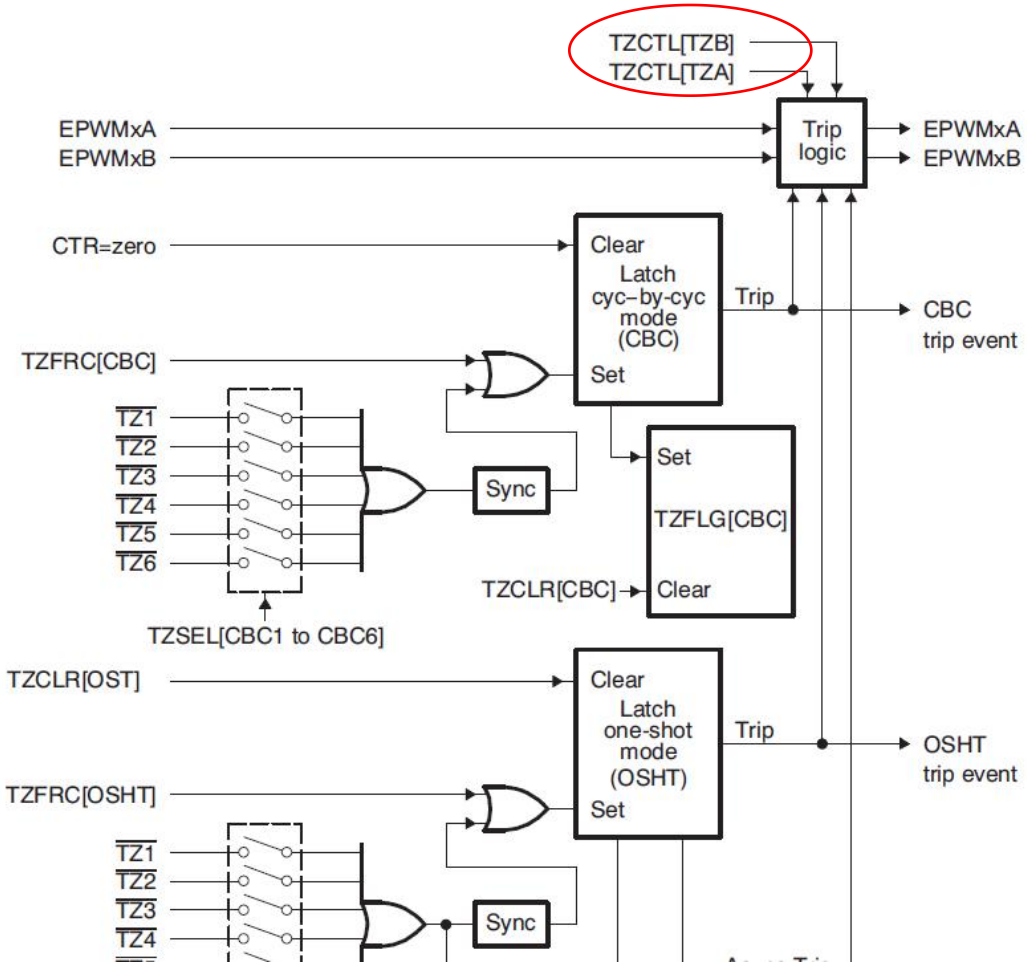


1. TZ子模块工作过程

（2）单次触发事件

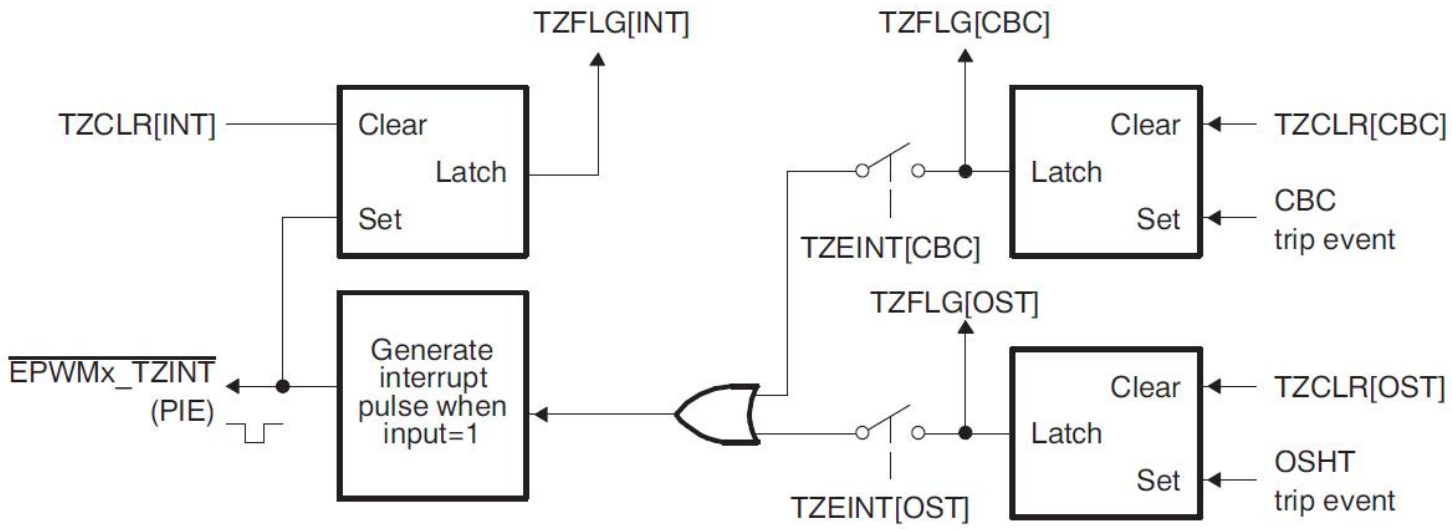


2. 故障捕获后输出



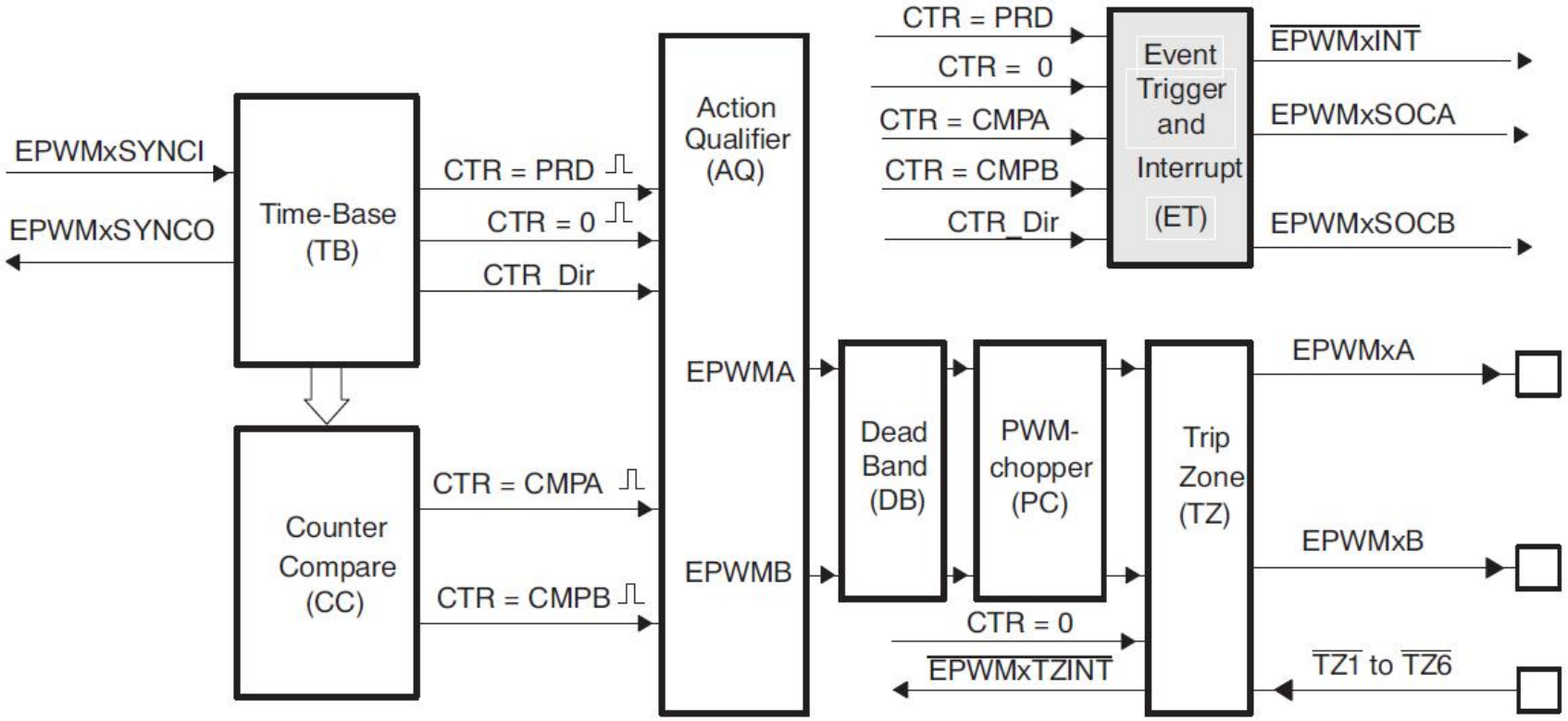
3–2	TZB		When a trip event occurs the following action is taken on output EPWMxB. Which trip-zone pins can cause an event is defined in the TZSEL register.
		00	High impedance (EPWMxB = High-impedance state)
		01	Force EPWMxB to a high state
		10	Force EPWMxB to a low state
		11	Do nothing, no action is taken on EPWMxB.

3. 中断功能


TZFLG[INT]
TZFLG[CBC]
TZCLR[INT]
Clear
Latch
Set
Clear
Latch
Set
TZCLR[CBC]
CBC
trip event
TZEINT[CBC]
TZFLG[OST]
EPWMx_TZINT
(PIE)
Generate
interrupt
pulse when
input=1
Clear
Latch
Set
TZCLR[OST]
OSHT
trip event
TZEINT[OST]

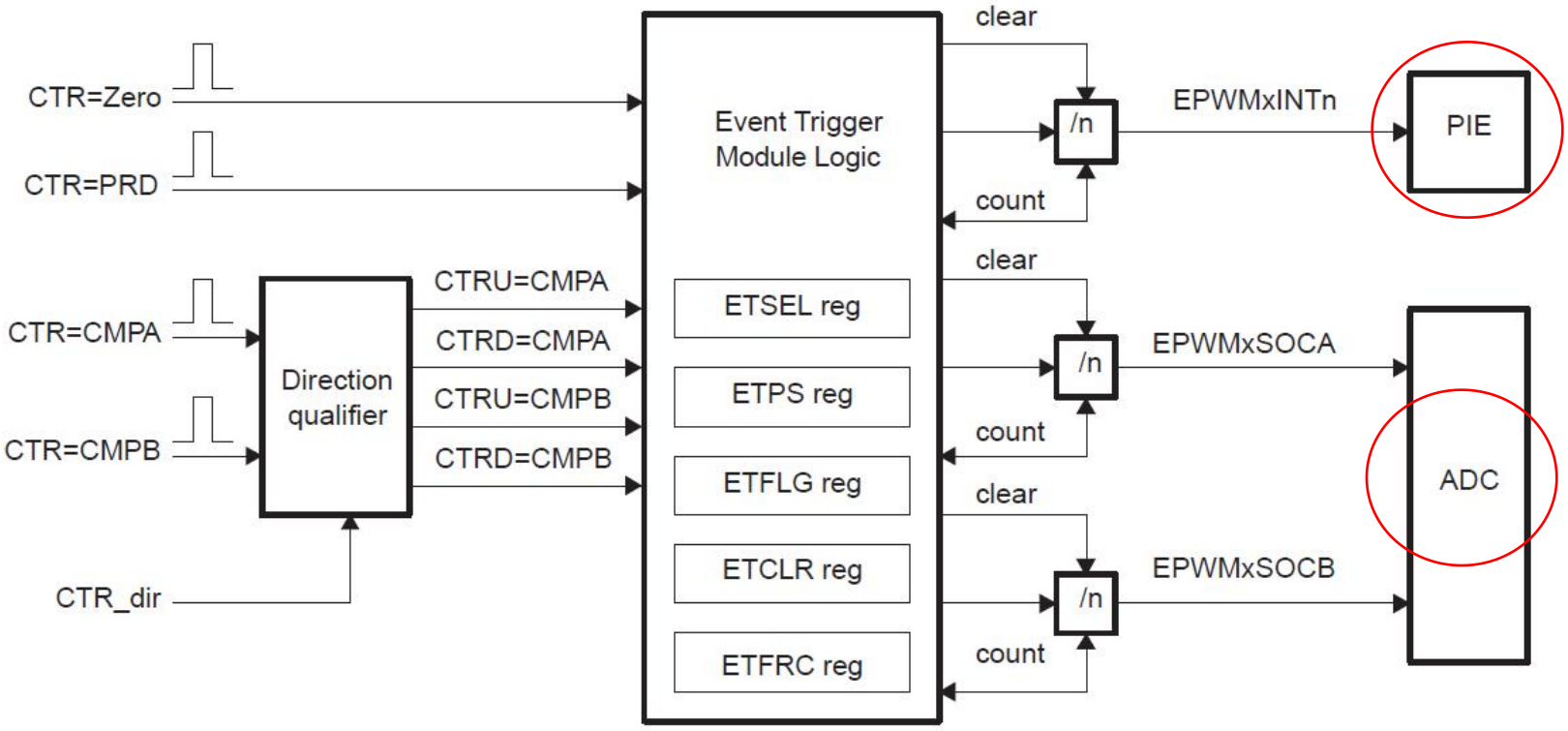
事件触发子模块

1. ET模块用来产生中断请求
2. 或产生ADC启动信号SOCA/SOCB

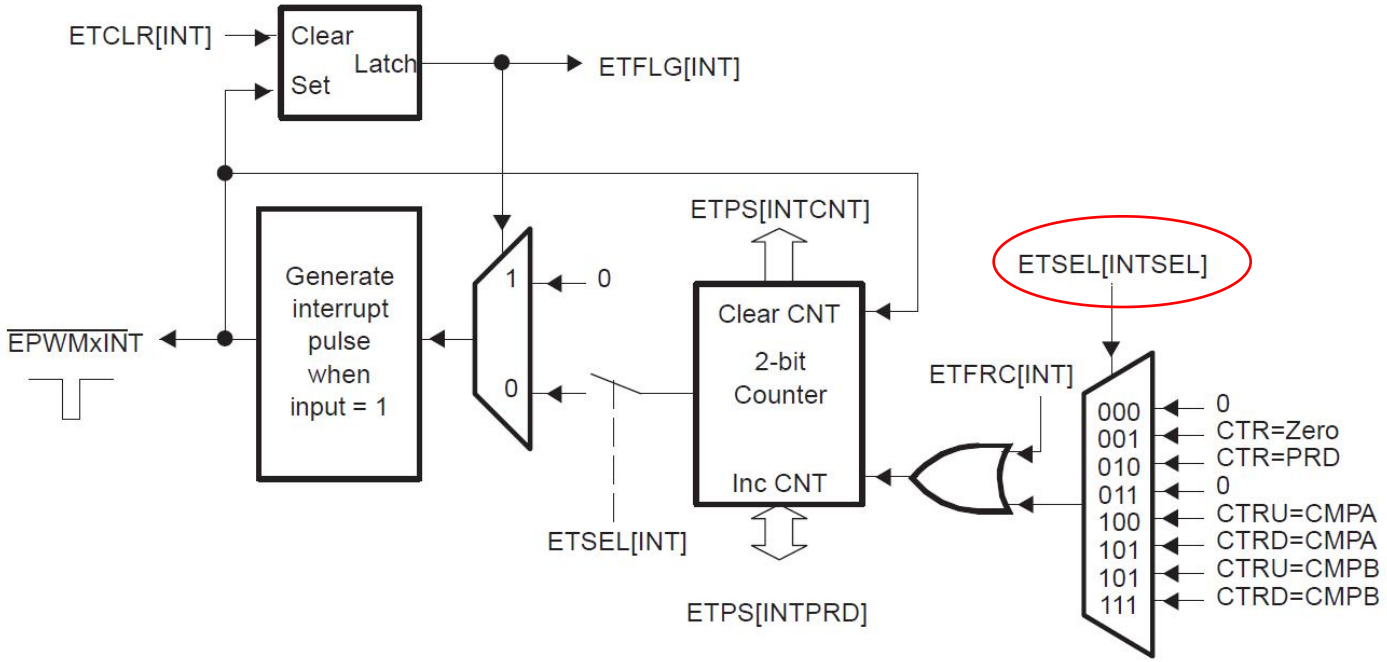


事件触发子模块

1. ET模块用来产生中断请求
2. 或产生ADC启动信号SOCA/SOCB

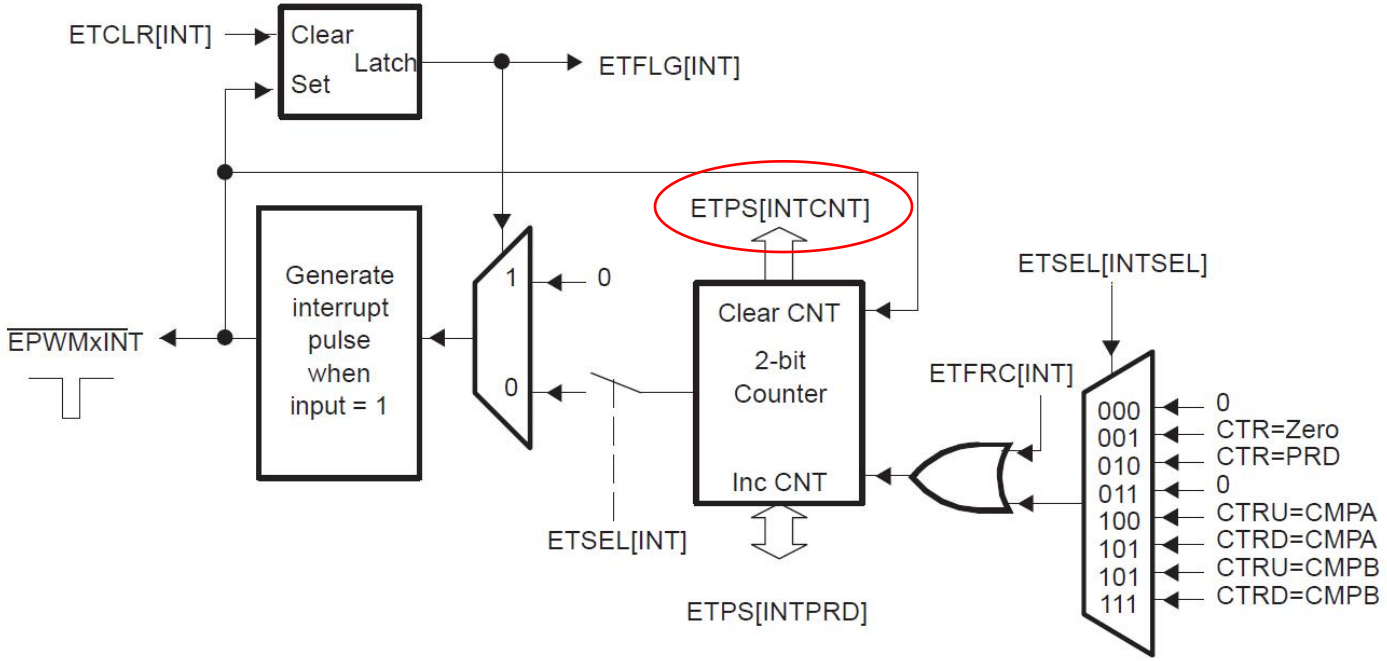


1. 中断控制功能



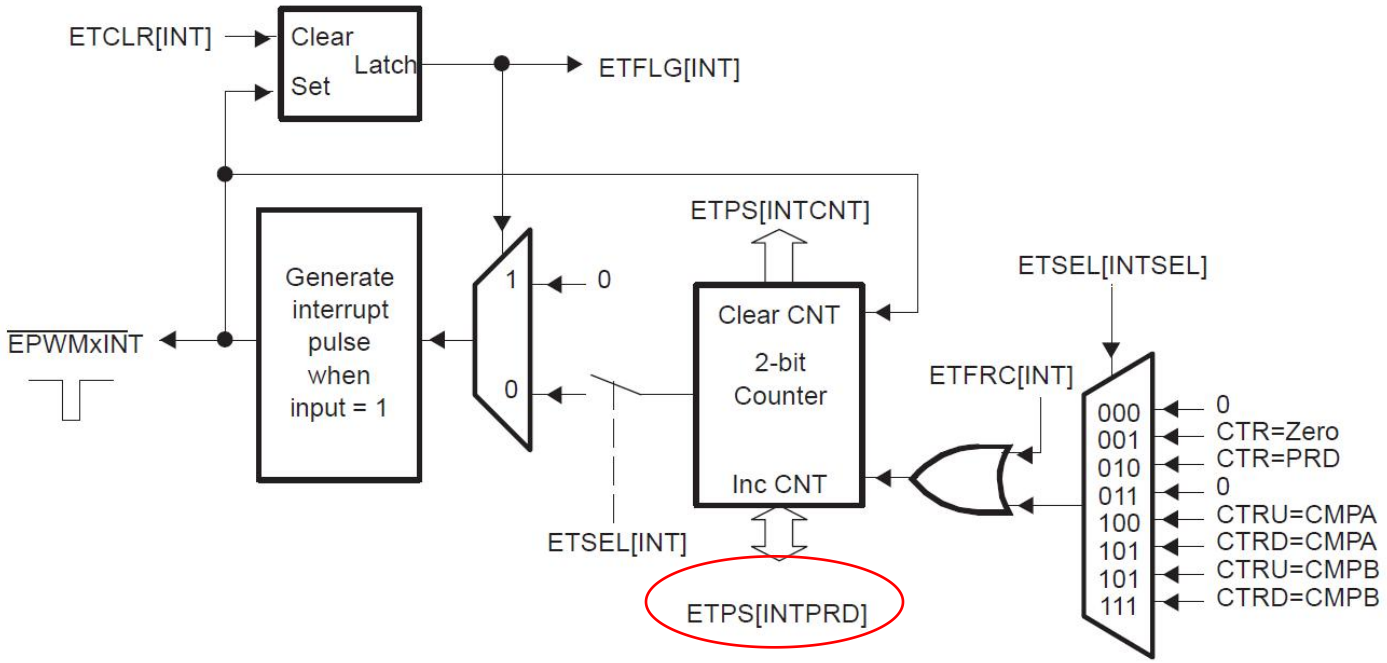
2-0	INTSEL		ePWM Interrupt (EPWMx_INT) Selection Options
		000	Reserved
		001	Enable event time-base counter equal to zero. (TBCTR = 0x0000)
		010	Enable event time-base counter equal to period (TBCTR = TBPRD)
		011	Reserved
		100	Enable event time-base counter equal to CMPA when the timer is incrementing.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

1. 中断控制功能



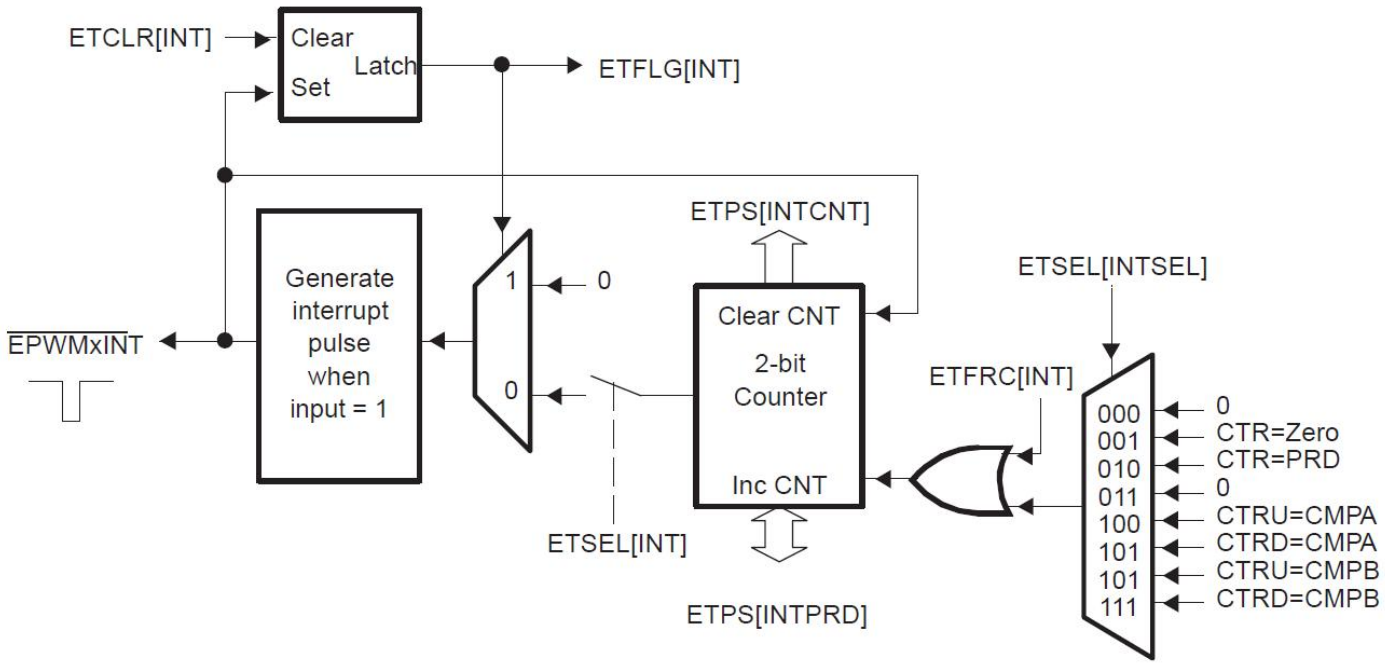
3-2	INTCNT		ePWM Interrupt Event (EPWMx_INT) Counter Register These bits indicate how many selected ETSEL[INTSEL] events have occurred. These bits are automatically cleared when an interrupt pulse is generated. If interrupts are disabled, ETSEL[INT] = 0 or the interrupt flag is set, ETFLG[INT] = 1, the counter will stop counting events when it reaches the period value ETPS[INTCNT] = ETPS[INTPRD].
		00	No events have occurred.
		01	1 event has occurred.
		10	2 events have occurred.
		11	3 events have occurred.

1. 中断控制功能



Bits	Name		ePWM Interrupt (EPWMx_INT) Period Select
1-0	INTPRD	00	Disable the interrupt event counter. No interrupt will be generated and ETFRC[INT] is ignored.
		01	Generate an interrupt on the first event INTCNT = 01 (first event)
		10	Generate interrupt on ETPS[INTCNT] = 1,0 (second event)
		11	Generate interrupt on ETPS[INTCNT] = 1,1 (third event)

1. 中断控制功能



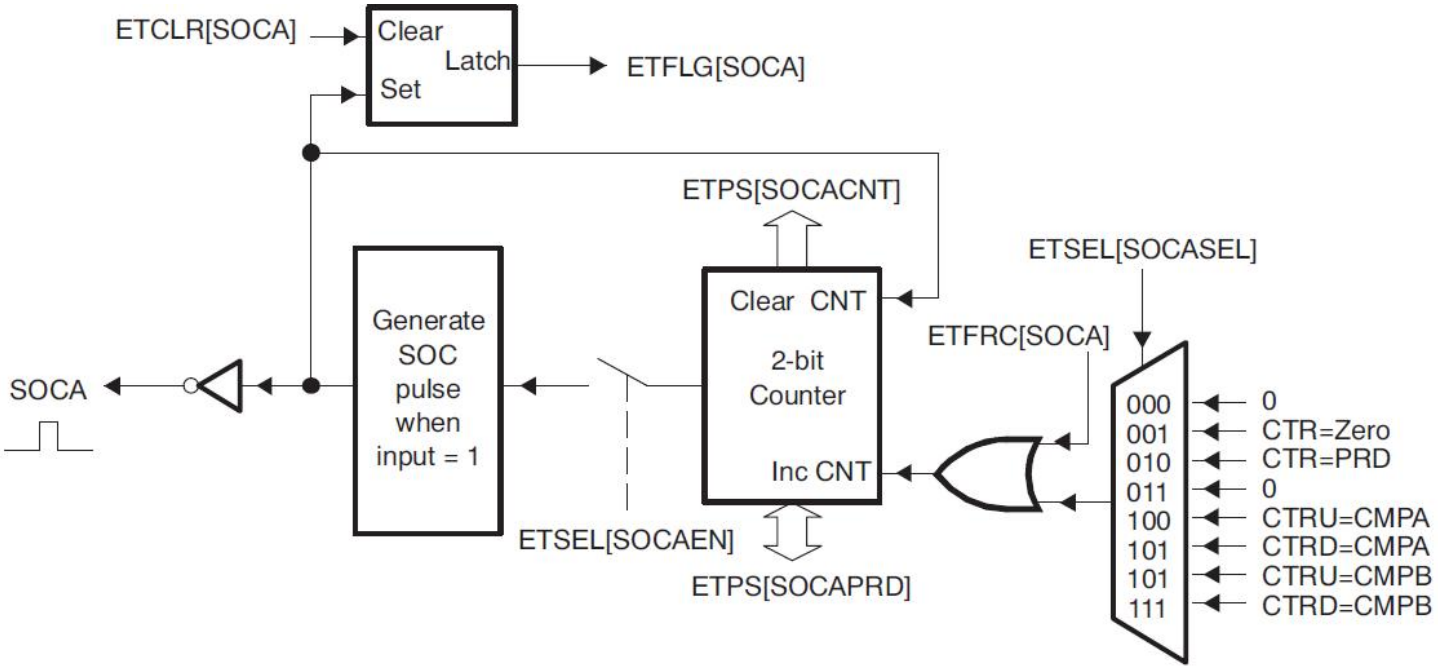
① 如果中断使能位ETSEL[INTEN]=1且中断标志位ETFLG[INT]=0,则产生中断并将ETFLG[INT]置1，中断事件计数器ETPS[INTCNT]清零，并重新计数;

② 如果ETSEL[INTEN]=0，或ETFLG[INT]=1，则不产生中断;

③ 如果ETSEL[INTEN]=1且ETFLG[INT]=1，中断事件计数器输出保持高直到ETFLG[INT]被清零，即当一个中断正在执行时允许另一个中断挂起。

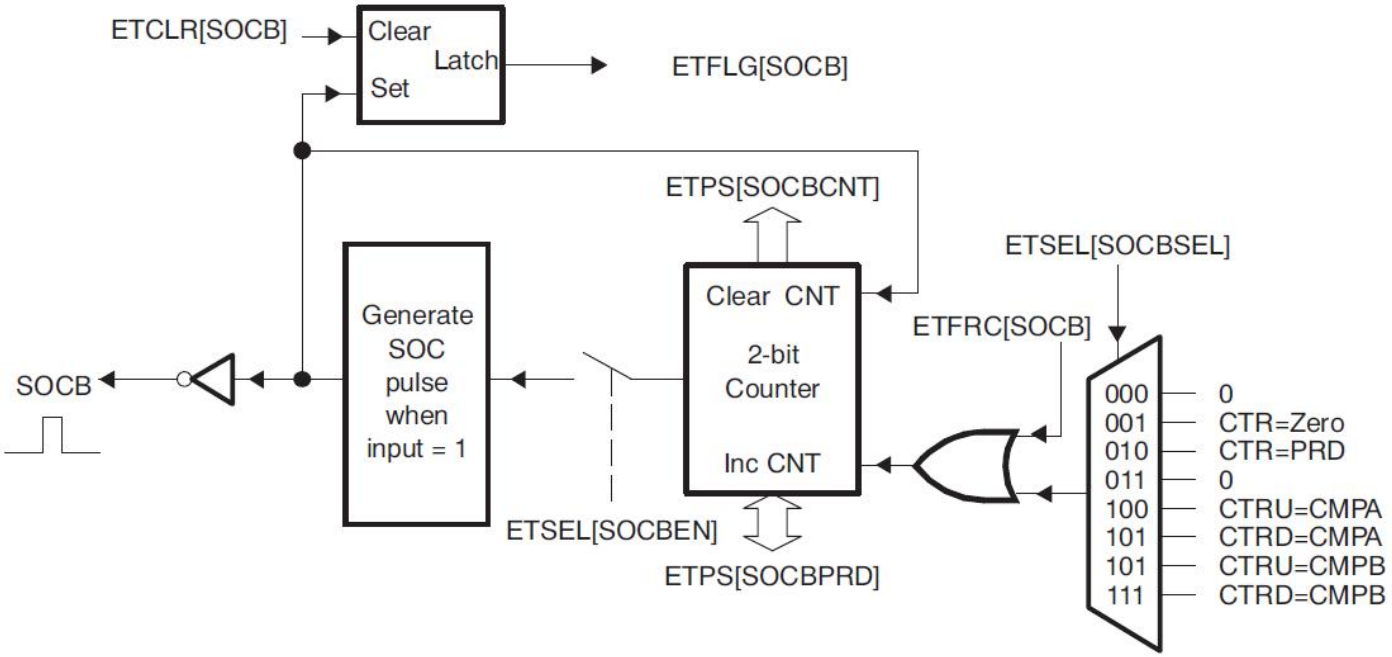
2. 启动信号SOC

与中断功能类似，不同点是当ETFLG[SOCA]被置位后仍产生连续脉冲。



2. 启动信号SOC

SOCB的产生与SOCA相同



第八讲 增强型脉宽调制模块ePWM

1、PWM基本原理

2、概述

3、ePWM子模块

4、ePWM寄存器

5、例子

已结合ePWM子模块介绍

第八讲 增强型脉宽调制模块ePWM

1、PWM基本原理

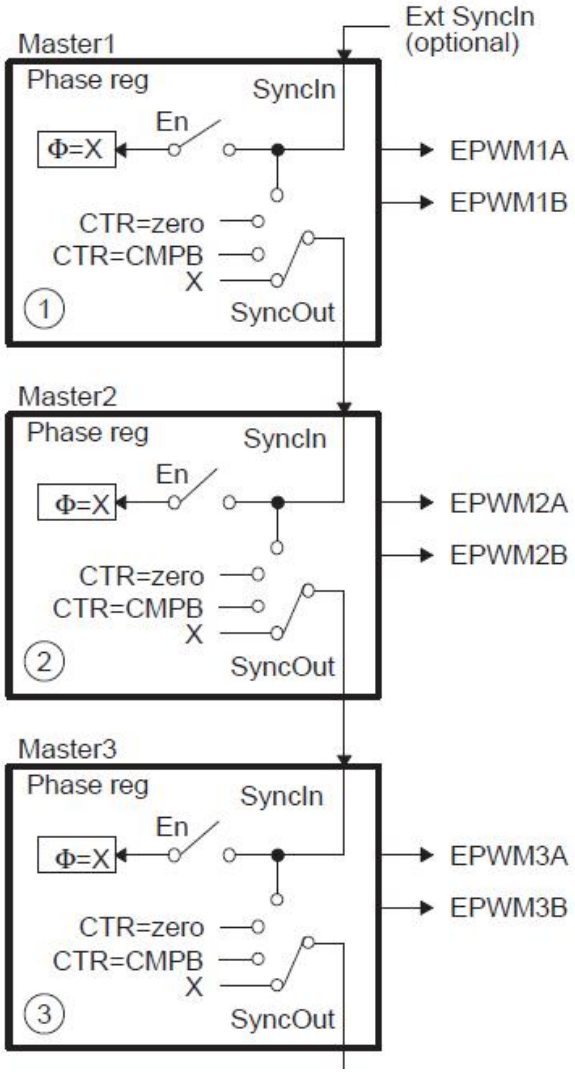
2、概述

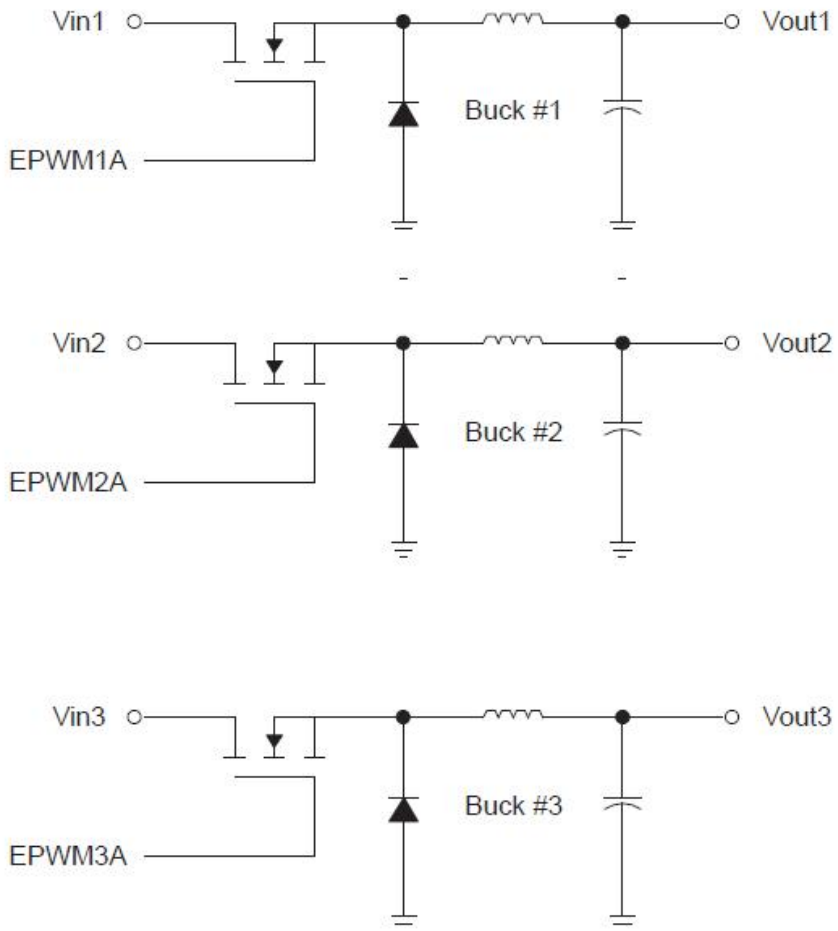
3、ePWM子模块

4、ePWM寄存器

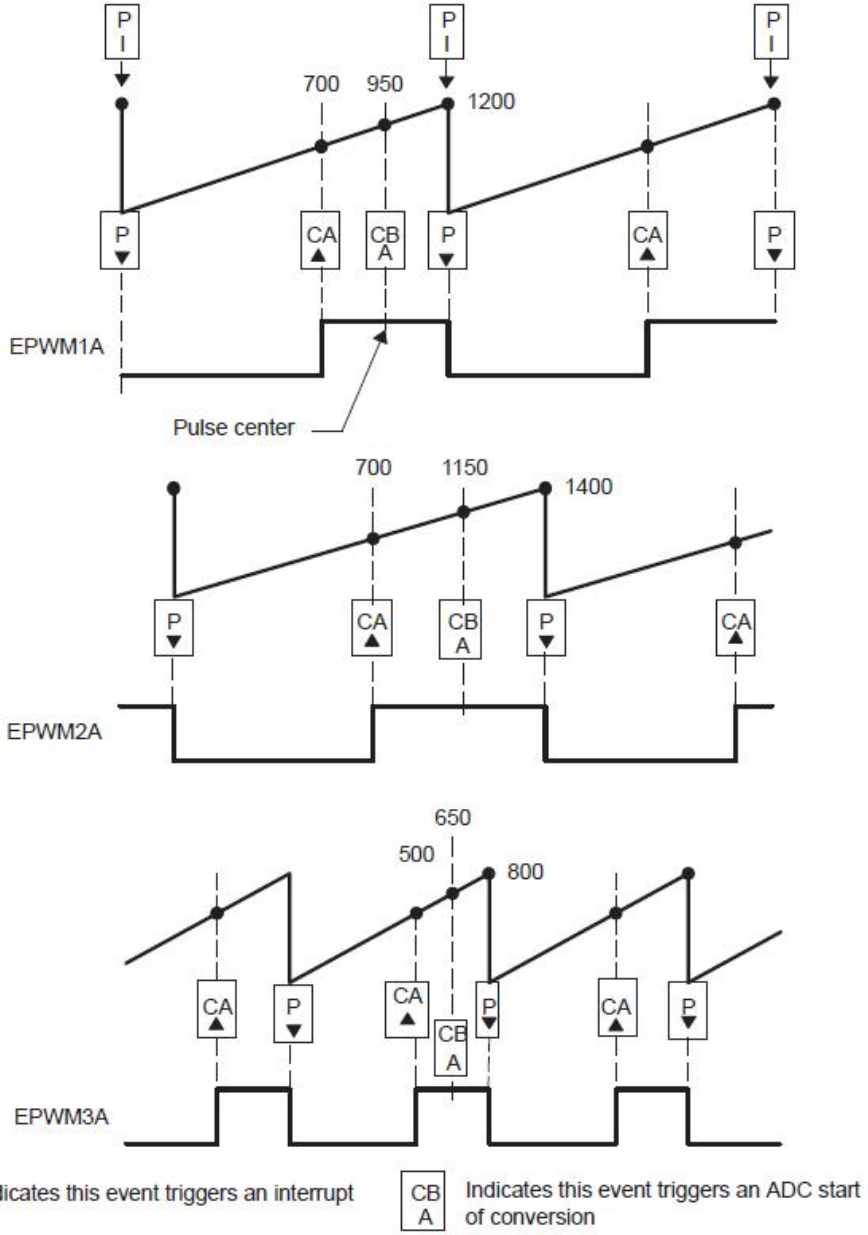
5、例子

1. 不同频率下多个BUCK电路的控制





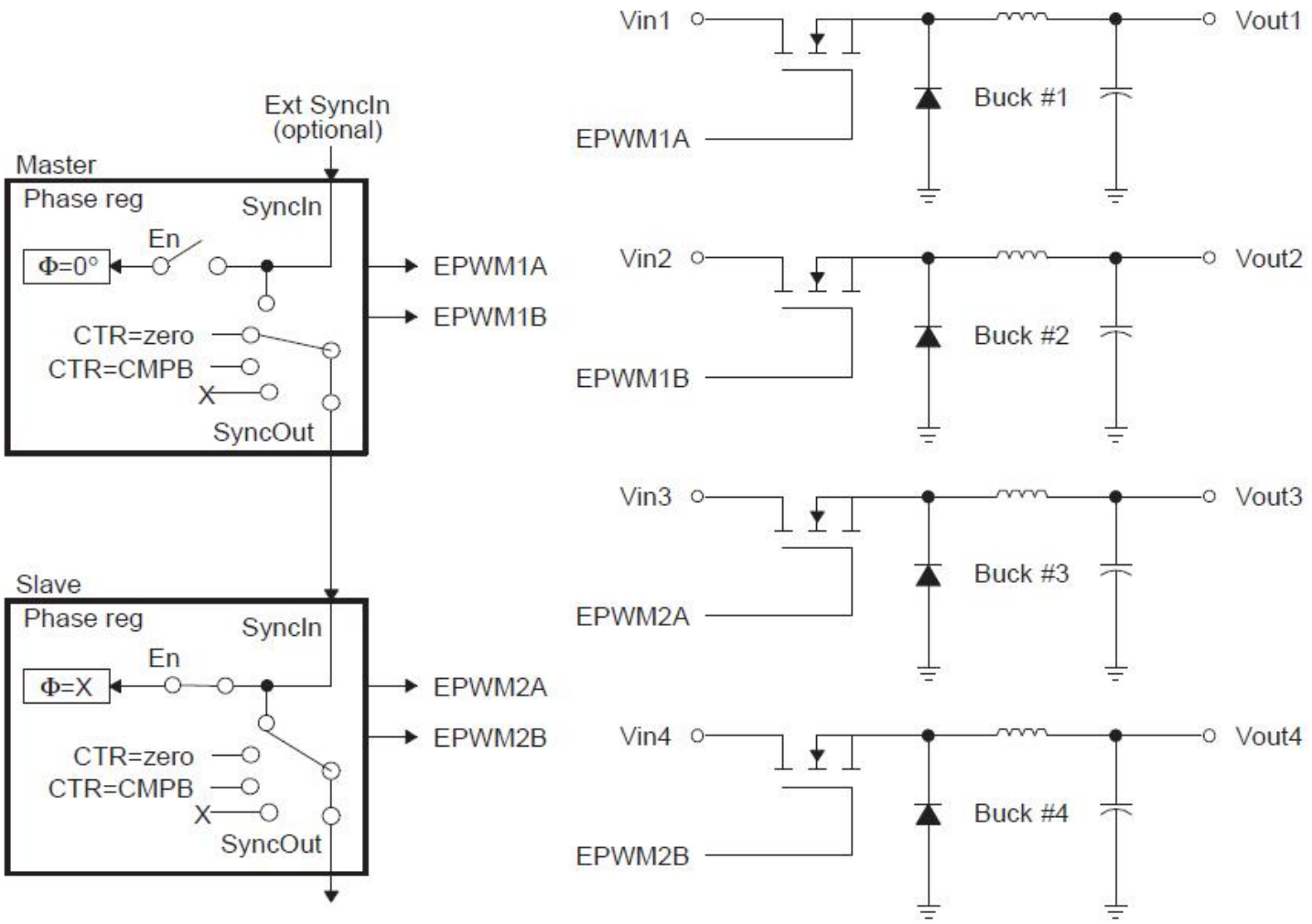
1. 不同频率下多个BUCK电路的控制



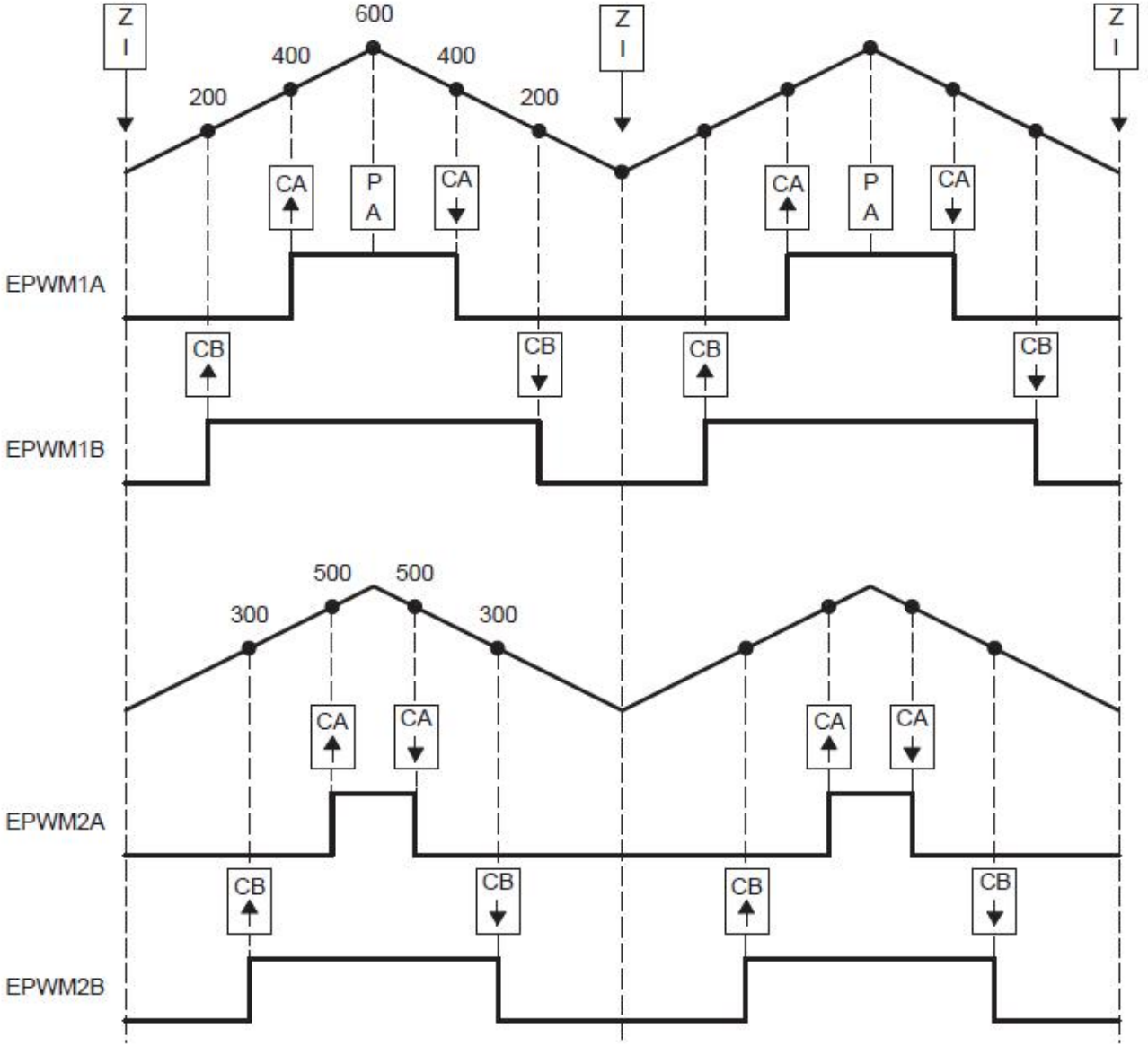
1. 不同频率下多个BUCK电路的控制

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

2. 同频率下多个BUCK电路的控制



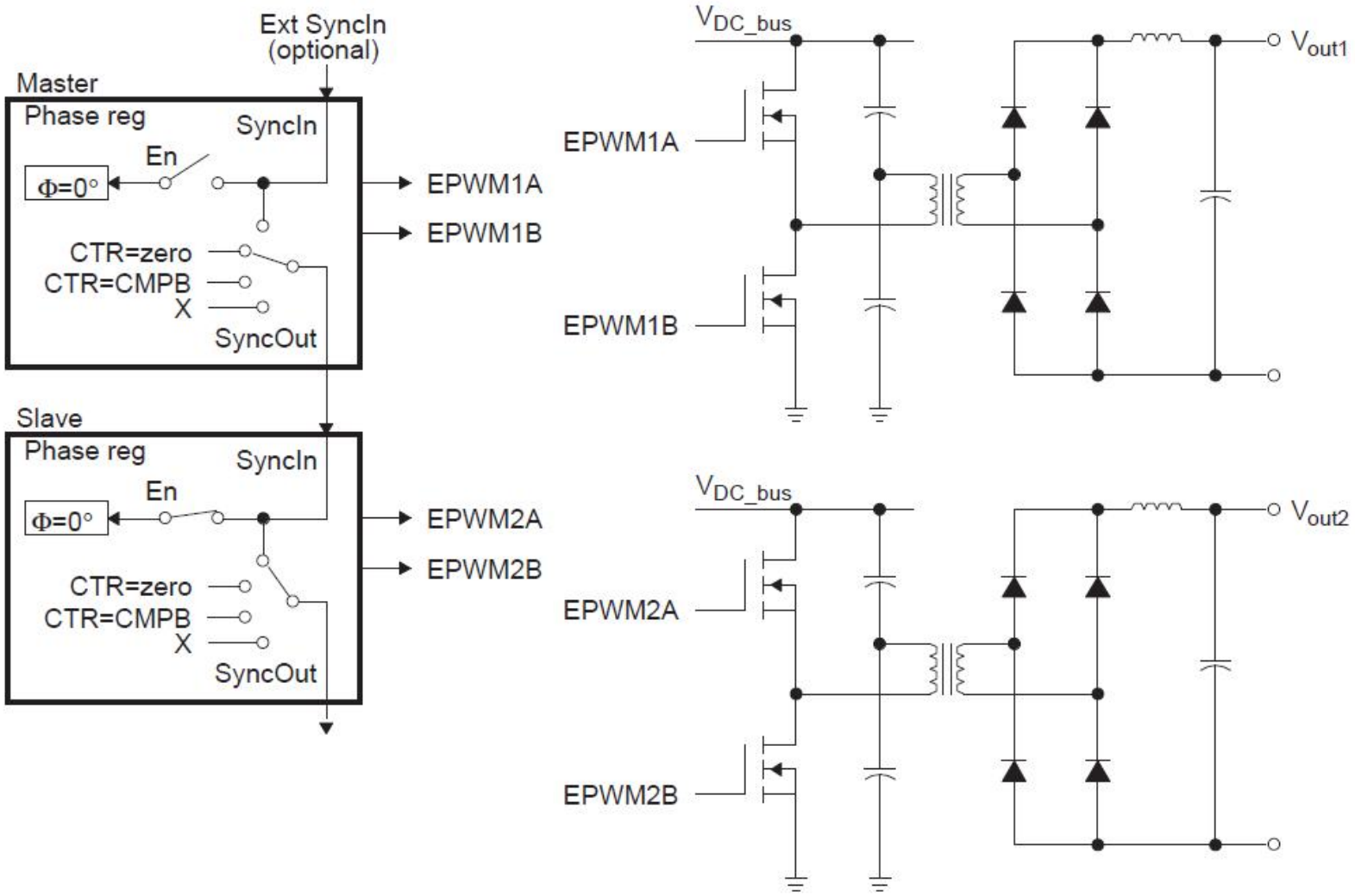
2. 同频率下多个BUCK电路的控制



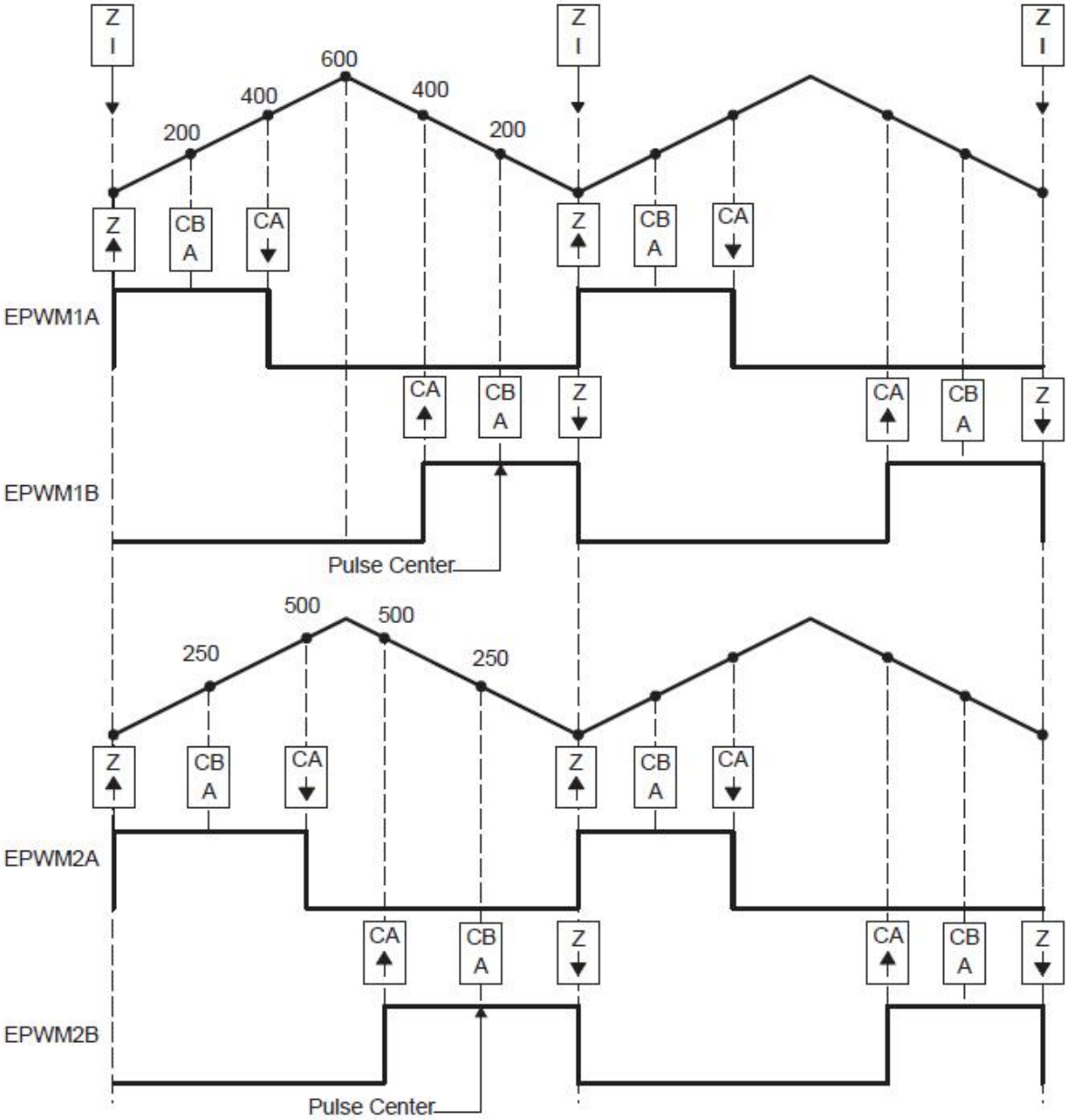
2. 同频率下多个BUCK电路的控制

```
//=======================
// EPWM Module 1 config
     EPwm1Regs.TBPRD = 600;                           // Period = 1200 TBCLK counts
     EPwm1Regs.TBPHS.half.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.half.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 instance)
//=====================================================
    EPwm1Regs.CMPA.half.CMPA = 400;                  // adjust duty for output EPWM1A
    EPwm1Regs.CMPB = 200;                            // adjust duty for output EPWM1B
    EPwm2Regs.CMPA.half.CMPA = 500;                  // adjust duty for output EPWM2A
    EPwm2Regs.CMPB = 300;                            // adjust duty for output EPWM2B
```

3. 半H桥逆变器的控制


Ext SyncIn
(optional)
Master
Phase reg
SyncIn
En
Φ=0°
CTR=zero
CTR=CMPB
X
SyncOut
EPWM1A
EPWM1B
Slave
Phase reg
SyncIn
En
Φ=0°
CTR=zero
CTR=CMPB
X
SyncOut
EPWM2A
EPWM2B
VDC_bus
EPWM1A
EPWM1B
Vout1
VDC_bus
EPWM2A
EPWM2B
Vout2

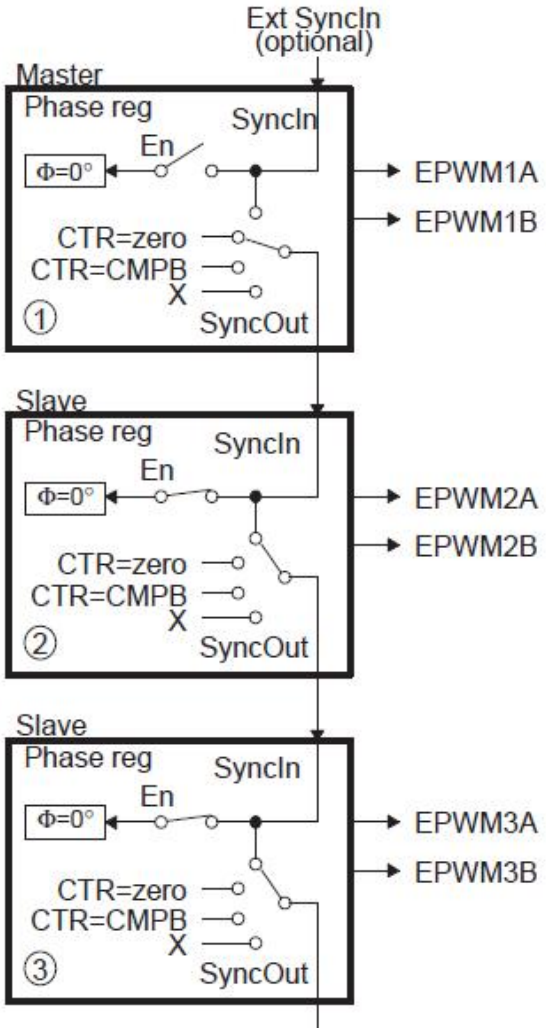
3. 半H桥逆变器的控制

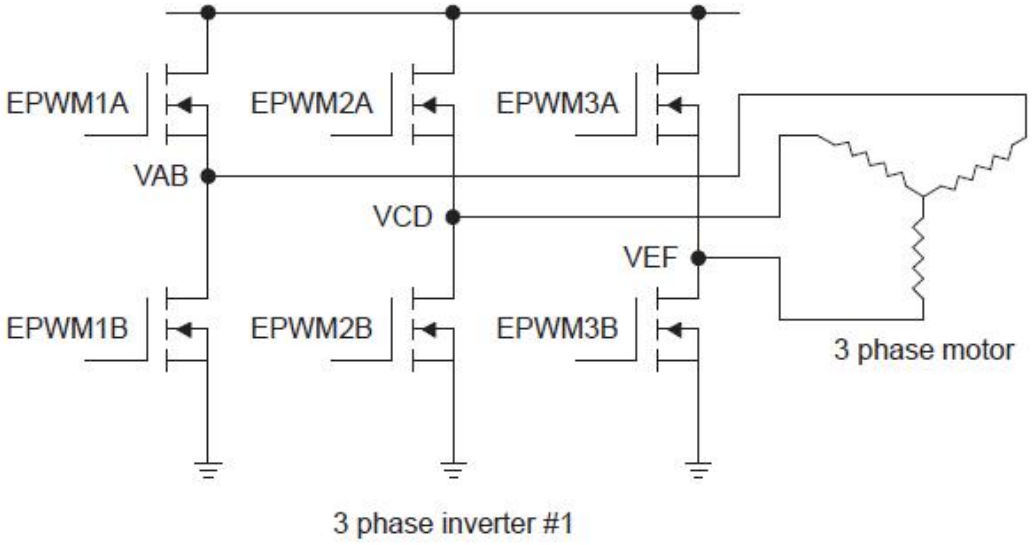

ZI
600
400
400
200
200
Z
CB
A
CA
EPWM1A
EPWM1B
Pulse Center
500
500
250
250
EPWM2A
EPWM2B
Pulse Center

3. 半H桥逆变器的控制

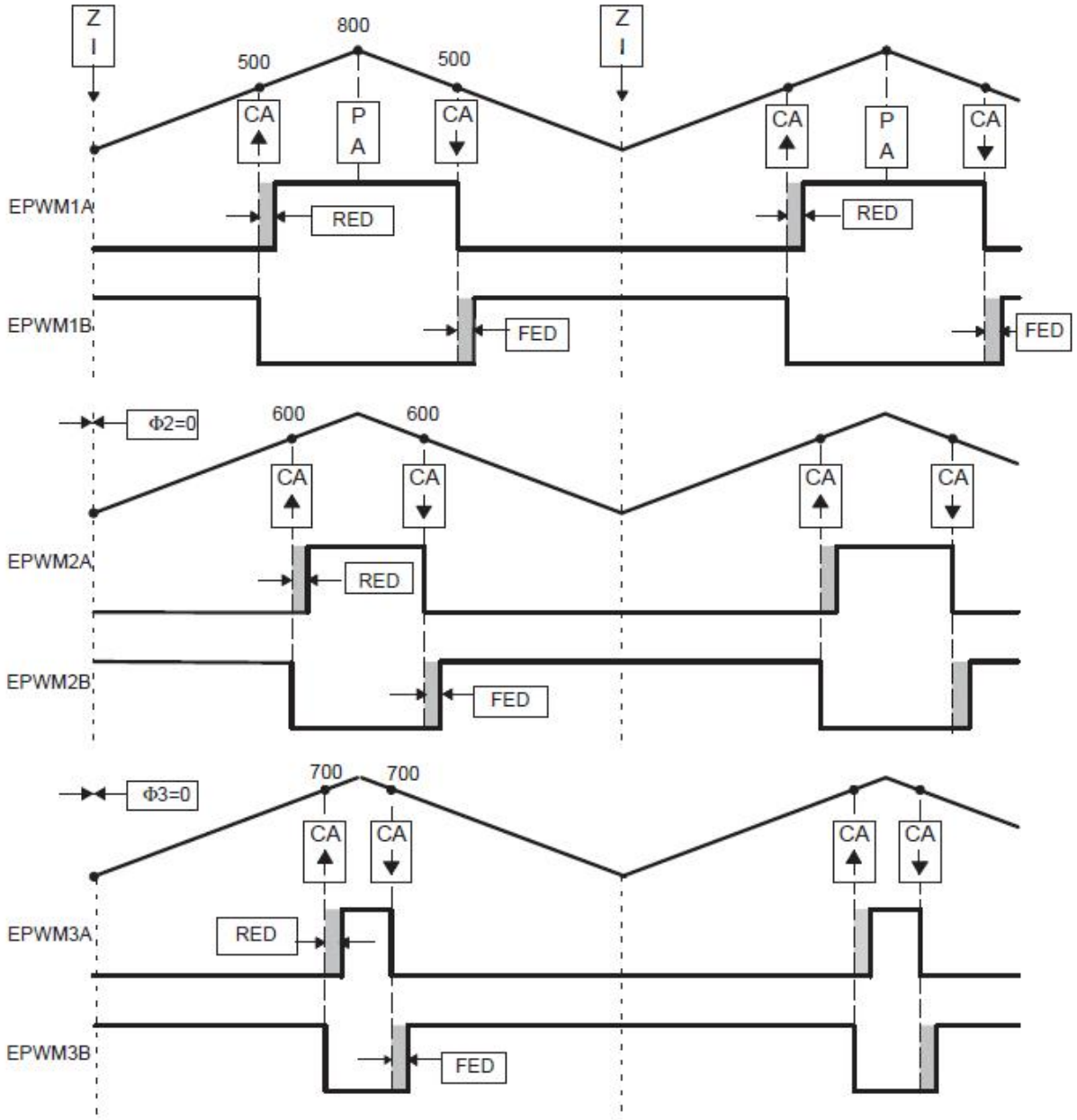
```
//===========================================================================
// Config
//===========================================================================
// Initialization Time
//=========================
// EPWM Module 1 config
     EPwm1Regs.TBPRD = 600;                           // Period = 1200 TBCLK counts
     EPwm1Regs.TBPHS.half.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.half.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;
//===========================================================
     EPwm1Regs.CMPA.half.CMPA = 400;                  // adjust duty for output EPWM1A & EPWM1B

     EPwm1Regs.CMPB = 200;                            // adjust point-in-time for ADCSOC trigger
     EPwm2Regs.CMPA.half.CMPA = 500;                  // adjust duty for output EPWM2A & EPWM2B
     EPwm2Regs.CMPB = 250;                            // adjust point-in-time for ADCSOC trigger
```

4. 三相逆变器的控制


Ext SyncIn
(optional)
Master
Phase reg
SyncIn
En
Φ=0°
EPWM1A
EPWM1B
CTR=zero
CTR=CMPB
X
SyncOut
①
Slave
Phase reg
SyncIn
En
Φ=0°
EPWM2A
EPWM2B
CTR=zero
CTR=CMPB
X
SyncOut
②
Slave
Phase reg
SyncIn
En
Φ=0°
EPWM3A
EPWM3B
CTR=zero
CTR=CMPB
X
SyncOut
③


EPWM1A
EPWM2A
EPWM3A
VAB
VCD
VEF
EPWM1B
EPWM2B
EPWM3B
3 phase motor
3 phase inverter #1

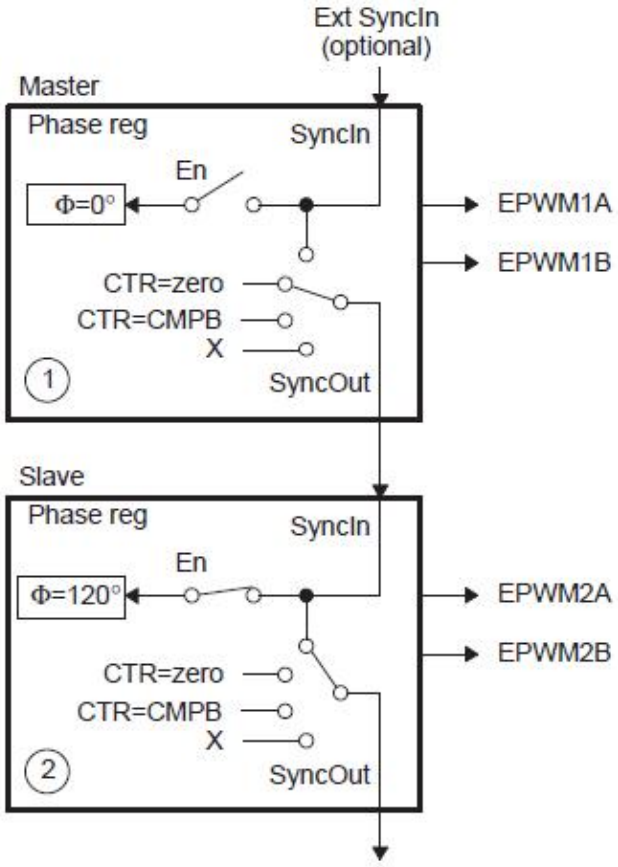
4. 三相逆变器的控制

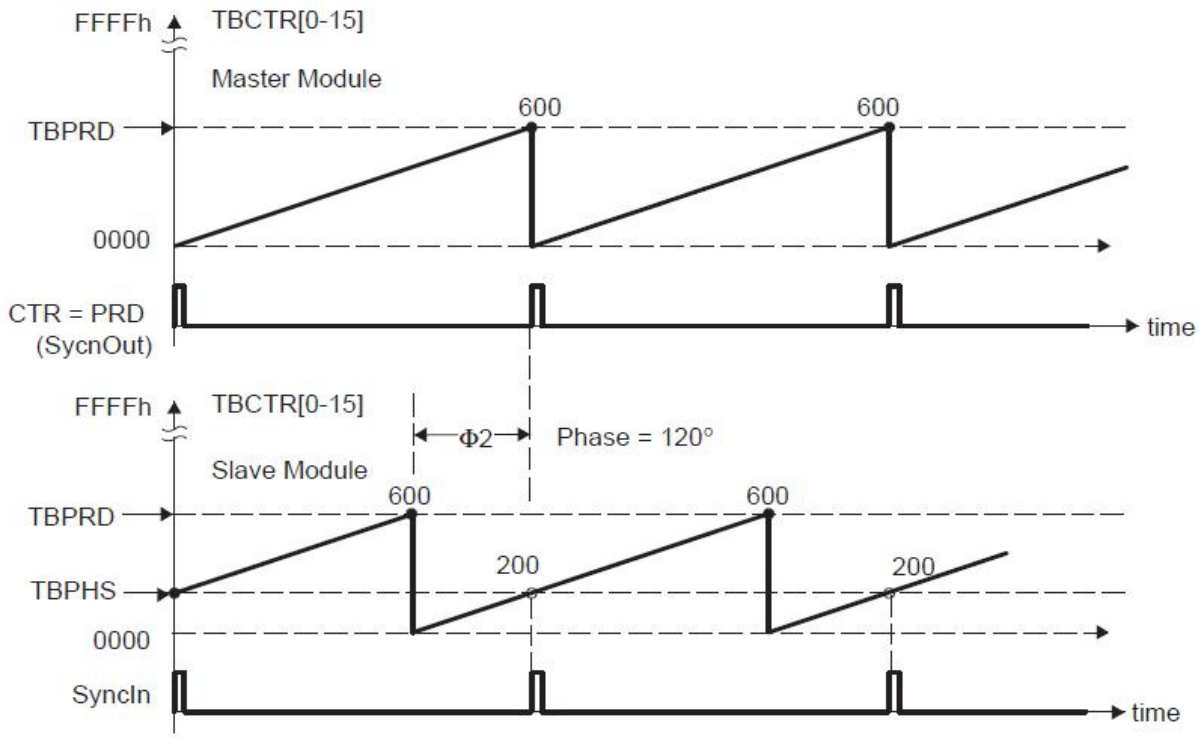

ZI
800
500
500
CA
PA
CA
ZI
CA
PA
CA
EPWM1A
RED
RED
EPWM1B
FED
FED
Φ2=0
600
600
CA
CA
CA
CA
EPWM2A
RED
EPWM2B
FED
Φ3=0
700
700
CA
CA
CA
CA
EPWM3A
RED
EPWM3B
FED

4. 三相逆变器的控制

```
//===========================================================================
// Configuration
//===========================================================================
// Initialization Time
//=======================// EPWM Module 1 config
     EPwm1Regs.TBPRD = 800;                           // Period = 1600 TBCLK counts
     EPwm1Regs.TBPHS.half.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.OUT_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.half.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.OUT_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.half.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.OUT_MODE = DB_FULL_ENABLE;       // enable Dead-band module
     EPwm3Regs.DBCTL.bit.POLSEL = DB_ACTV_HIC;        // Active Hi complementary
     EPwm3Regs.DBFED = 50;                            // FED = 50 TBCLKs
     EPwm3Regs.DBRED = 50;                            // RED = 50 TBCLKs
// Run Time (Note: Example execution of one run-time instant)
//===========================================================
     EPwm1Regs.CMPA.half.CMPA = 500;                  // adjust duty for output EPWM1A
     EPwm2Regs.CMPA.half.CMPA = 600;                  // adjust duty for output EPWM2A
     EPwm3Regs.CMPA.half.CMPA = 700;                  // adjust duty for output EPWM3A
```

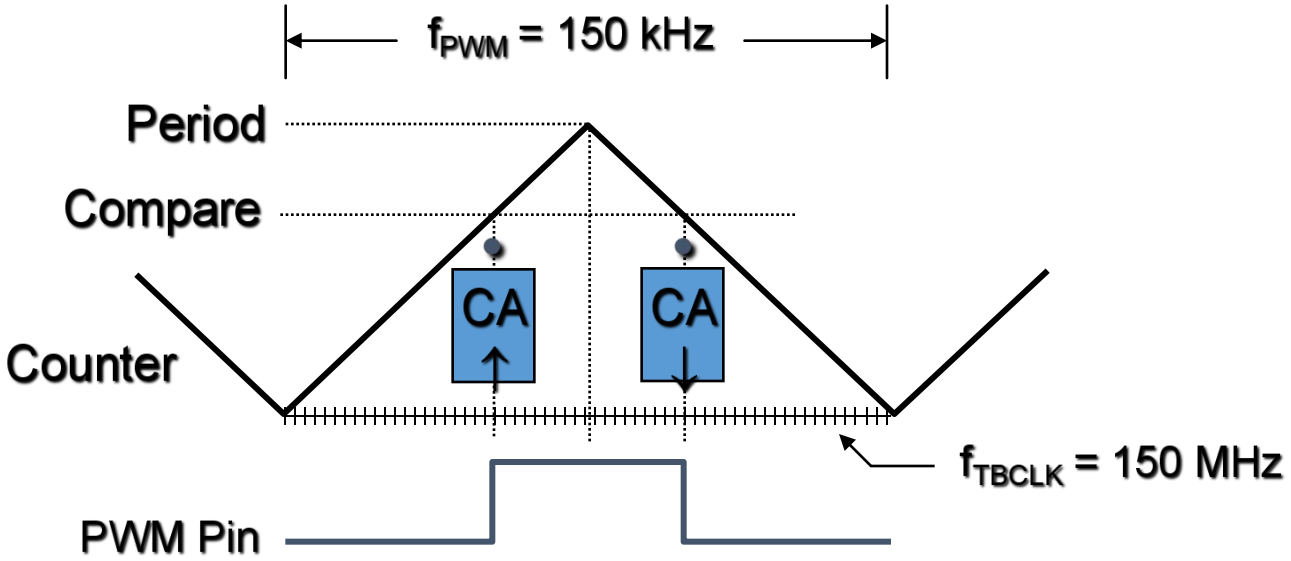
5. 移相控制


Ext SyncIn
(optional)
Master
Phase reg
SyncIn
En
Φ=0°
EPWM1A
EPWM1B
CTR=zero
CTR=CMPB
X
SyncOut
1
Slave
Phase reg
SyncIn
En
Φ=120°
EPWM2A
EPWM2B
CTR=zero
CTR=CMPB
X
SyncOut
2


FFFFh
TBCTR[0-15]
Master Module
600
600
TBPRD
0000
CTR = PRD
(SycnOut)
time
FFFFh
TBCTR[0-15]
Φ2
Phase = 120°
Slave Module
600
600
TBPRD
200
200
TBPHS
0000
SyncIn
time

例： 对称PWM的计算

- 求150MHz CPU 频率下， 150 kHz, 25%的对称PWM输出所对应的TBPRD和CMPA的值。

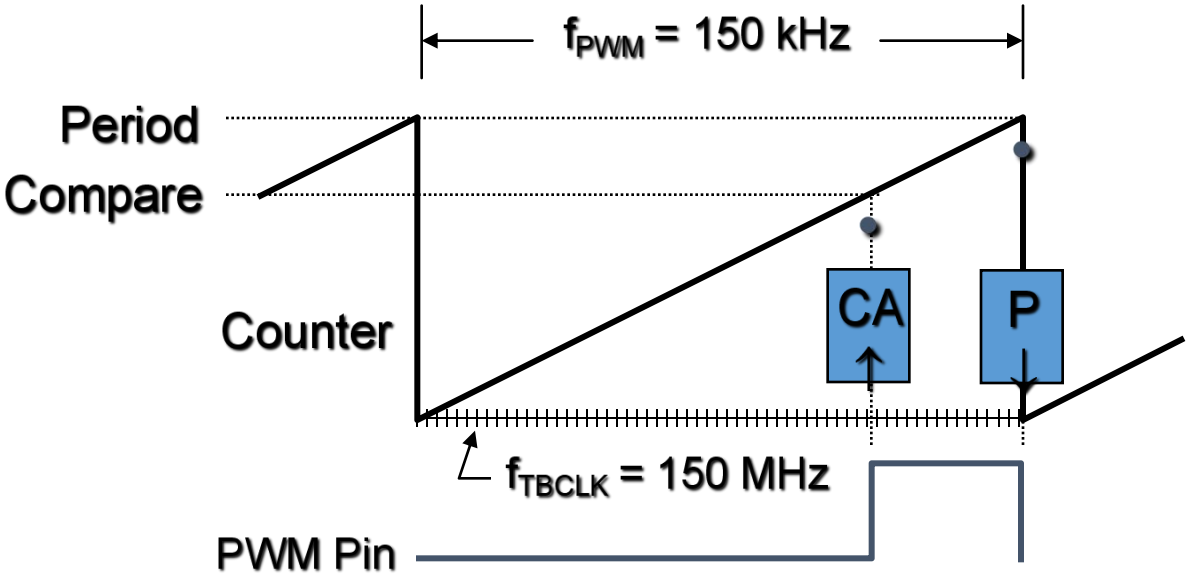


$$TBPRD = \frac{1}{2} \cdot \frac{f_{TBCLK}}{f_{PWM}} = \frac{1}{2} \cdot \frac{150\ MHz}{150\ kHz} = 500$$

CMPA = (100% - 占空比)*TBPRD = 0.75*500 = 375

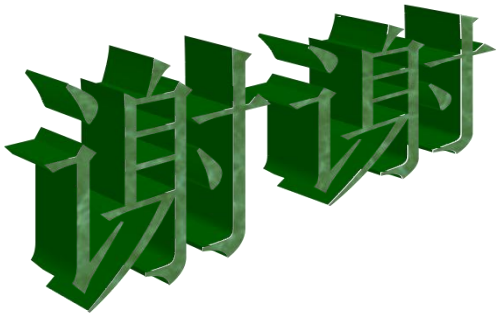
例： 不对称PWM的计算

- 求150MHz CPU 频率下， 150 kHz, 25%的不对称PWM输出所对应的TBPRD和CMPA的值。



$$TBPRD = \frac{f_{TBCLK}}{f_{PWM}} - 1 = \frac{150\ MHz}{150\ kHz} - 1 = 999$$

CMPA = (100% - 占空比)*(TBPRD+1) - 1 = 0.75*(999+1) - 1 = 749


谢谢