

Marlyn Rosales and Nima Eskandari

摘要

增强型脉宽调制器 (ePWM) 外设是控制工业和汽车应用中的很多电力电子系统的关键元件。C2000™ 实时微控制器在 ePWM 外设中提供了许多差异化功能，支持高级控制技术。本应用手册重点介绍简介部分描述的应用用例，演示了 ePWM 的每项特性以及如何使用 SysConfig 系统配置工具设置所需的输出并进行编程。SysConfig 工具可集成于 Code Composer Studio™ (CCS) 中，或作为独立程序使用，支持您使用图形用户界面 (GUI) 生成 C 头文件和代码文件。本应用手册是根据 TMS320F28388D 器件编写的。但是，本应用手册中的内容适用于具有 4 类 ePWM 模块的所有器件。更多有关外设类型的信息，请参阅 [C2000 实时控制 MCU 外设指南](#)。

内容

1 引言	3
2 SysConfig	4
3 时基 (TB) 子模块	4
3.1 设置频率.....	4
3.2 应用相移.....	6
3.3 设置同步 (Sync) 方案.....	6
4 计数器比较 (CC) 和动作限定器 (AQ) 子模块	8
4.1 计算占空比.....	8
5 死区 (DB) 子模块	10
5.1 设置信号对.....	10
6 验证输出	12
6.1 检查占空比和死区时间插入.....	12
6.2 检查应用的相移.....	13
7 跳闸区域 (TZ) 和数字比较 (DC) 子模块	14
7.1 通过 CMPSS 设置跳闸条件时，在 ePWM 周期内将输出驱动为低电平.....	14
7.2 通过 GPIO 设置跳闸条件时，通过软件将输出驱动为低电平，直到清除.....	19
8 事件触发 (ET) 子模块	22
8.1 设置时基中断.....	22
9 全局加载	22
9.1 应用全局加载和一次性加载功能.....	22
9.2 链接 ePWM 模块.....	23
9.3 通过全局加载更新动作限定器设置和计数器比较值.....	23
10 总结	25
11 参考文献	25
12 修订历史记录	25

插图清单

图 1-1. ePWM 模块的方框图.....	3
图 2-1. SysConfig - ePWM 模块：全局参数.....	4
图 3-1. ePWM 时基：设置 TBPRD 和计数模式.....	5
图 3-2. 同步方案.....	6
图 3-3. EPWM 时基：EPWM1 作为同步源的配置.....	7
图 3-4. ePWM 时基：EPWM2 同步接收器设置的配置.....	7
图 4-1. ePWM 动作限定器：设置全局加载和输出事件.....	8
图 4-2. ePWM 计数器比较：设置计数器比较值.....	9

图 5-1. 高电平有效互补信号对.....	10
图 5-2. 死区子模块的配置选项.....	10
图 5-3. ePWM 死区：高电平有效互补.....	11
图 6-1. ePWM 输出正占空比的示波器捕获.....	12
图 6-2. 插入死区时间的 ePWM 输出示波器捕获.....	13
图 6-3. 具有相移的 EPWM 输出的示波器捕获.....	13
图 7-1. 比较器块图.....	14
图 7-2. 比较器引脚.....	15
图 7-3. CMPSS 配置.....	15
图 7-4. ePWM X-BAR 多路复用器配置表的部分示例.....	16
图 7-5. 输入 X-BAR 方框图.....	17
图 7-6. EPWMXBAR 的 SysConfig 配置.....	18
图 7-7. ePWM 数字比较：设置 DCAEVT2.....	18
图 7-8. ePWM 跳闸区域：基于 DCAEVT2 的 CBC 跳闸配置.....	19
图 7-9. SysConfig 中的 GPIO 设置.....	20
图 7-10. SysConfig 中的输入 X-BAR.....	20
图 7-11. 跳闸区域配置，包括一次性配置.....	21
图 8-1. ePWM 事件触发器：设置 TBCTR=ZRO 时中断.....	22
图 9-1. 全局加载 SysConfig 设置.....	23
图 9-2. AQ 和 CC 更新后 EPWM 输出正占空比的示波器捕获.....	25

商标

C2000™ and Code Composer Studio™ are trademarks of Texas Instruments.

所有商标均为其各自所有者的财产。

1 引言

ePWM 模块分为多个子模块，每个子模块都有自己的功能。图 1-1 展示了这些子模块如何相互连接。本应用报告详细介绍了每个不同的子模块。

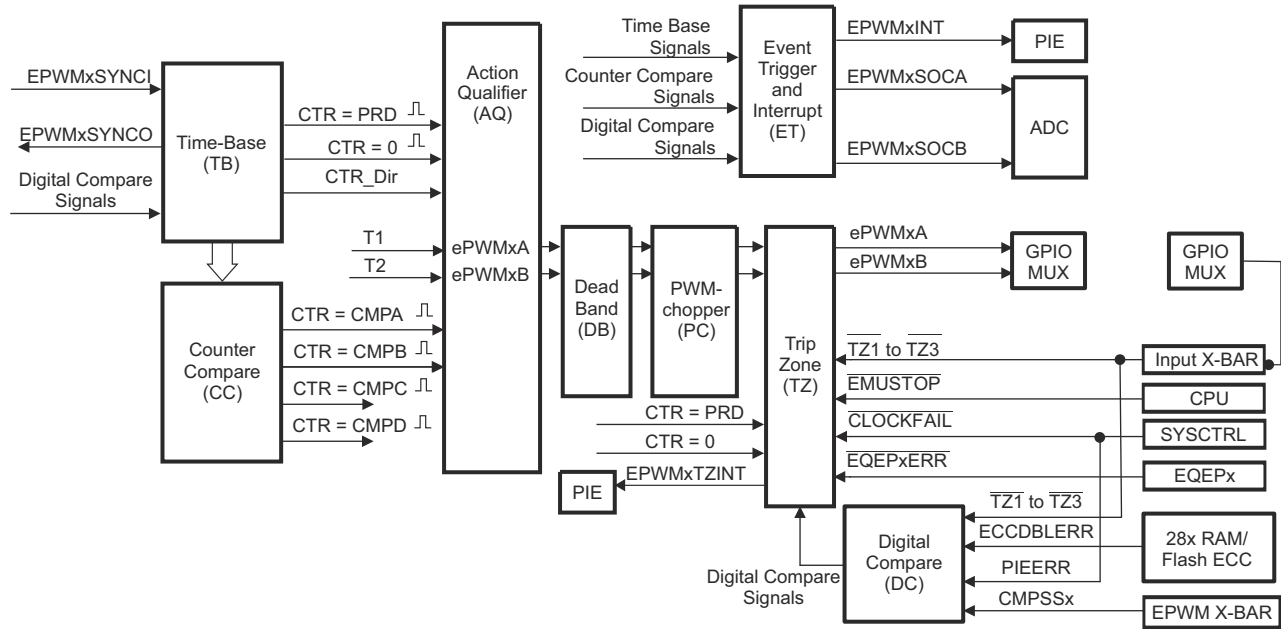


图 1-1. ePWM 模块的方框图

本应用报告中讨论的用例具有以下标准：

用例

- EPWM1/2/3 的输出频率为 400kHz
- EPWM2 相对于 EPWM1 的相移为 120°
- EPWM3 相对于 EPWM1 的相移为 240°
- EPWM1/2/3 的占空比为 45%
- EPWM1/2/3 的高电平有效互补信号对具有 200ns 的上升/下降沿延迟
- 通过 EPWM2 上的比较器信号实现逐周期跳闸区域保护
- 通过 EPWM3 上的通用输入/输出 (GPIO) 实现一次性跳闸保护
- 每次 EPWM1 上的时基计数器等于 0 时生成中断
- 全局加载，支持动作限定器设置的异步更新
- 将 EPWM1 的 CMPA/CMPB 链接到 EPWM2 和 EPWM3

2 SysConfig

如摘要中所述，本应用报告利用 SysConfig 来配置 ePWM 模块以及其他必要元件。在 SysConfig 中，ePWM 模块的每个子模块都有各自的分组，包括全局加载和 HRPWM 分组。如需有关 SysConfig 的更多常规信息，请参阅 [节 11](#) 中列出的项目。

除了能够配置每个子模块之外，ePWM 模块的顶部还有“Global Parameters”部分。这些设置用于启用信息/警告，还可影响如何从 ePWM 模块生成代码。

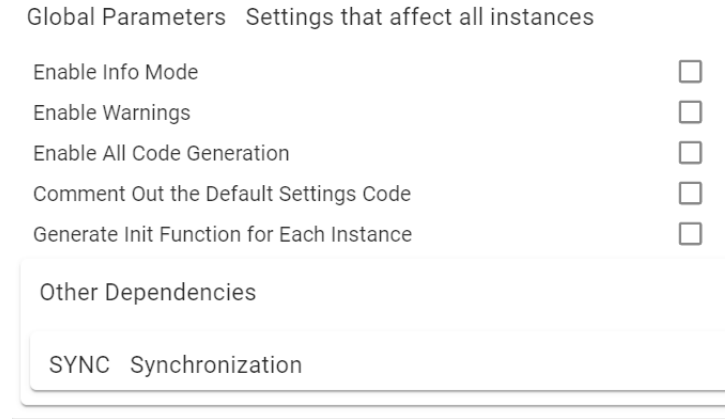


图 2-1. SysConfig - ePWM 模块：全局参数

备注

默认情况下，SysConfig 仅生成一组预定义代码，以及 EPWM 模块的任何非默认状态设置的代码。但是，可以在“Global Parameters”部分中更改这些设置。

3 时基 (TB) 子模块

3.1 设置频率

时基 (TB) 子模块用于将 ePWM 模块的频率设置为 400kHz。每个 ePWM 模块都有一个时基计数器 (TBCTR)。时基计数器有三种模式：向上计数、向下计数和向上向下双向计数。PWM 事件的频率 (F_{PWM}) 由时基周期 (TBPRD) 寄存器和时基计数器的模式控制。

对于向上计数和向下计数模式：

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

$$F_{PWM} = \frac{1}{T_{PWM}} \quad (2)$$

对于向上向下双向计数模式：

$$T_{PWM} = 2 * TBPRD * T_{TBCLK} \quad (3)$$

$$F_{PWM} = \frac{1}{T_{PWM}} \quad (4)$$

其中， T_{PWM} 是 PWM 事件的周期， T_{TBCLK} 是时基时钟的周期。时基时钟 (TBCLK) 是 ePWM 时钟 (EPWMCLK) 的预分频版本。该时钟决定时基计数器递增和递减的速率。

本应用报告中讨论的用例使用了向上向下双向计数模式，由于计数模式的对称性提供了更多可配置选项。首先确定 T_{PWM} ：

$$F_{PWM} = \frac{1}{T_{PWM}} \rightarrow T_{PWM} = \frac{1}{F_{PWM}} \quad (5)$$

$$T_{PWM} = \frac{1}{400k} \rightarrow 2.5 \mu\text{sec} \quad (6)$$

在知道 T_{PWM} 值之后，可以开始计算 TBPRD 的值：

$$TBPRD = \frac{T_{PWM}}{2 * T_{TBCLK}} \quad (7)$$

时基时钟通过以下公式定义：

$$TBCLK = \frac{EPWMCLK}{HSPCLKDIV * CLKDIV} \quad (8)$$

器件特定的数据表中定义了最大 EPWMCLK。EPWMCLK 通常与 SYSCLK 相同，但某些器件可能有时钟分频器 EPWMCLKDIV，这样会降低 EPWM 时钟的频率。高速时钟分频器 (HSPCLKDIV) 和时钟分频器 (CLKDIV) 是可编程分频器，有助于实现所需的 ePWM 时基时钟。对于本用例，两个时钟分频器都设置为除以 1，EPWMCLK 为 100MHz。

$$TBCLK = \frac{100M}{1 * 1} \rightarrow 100 \text{ MHz} \quad (9)$$

$$T_{TBCLK} = \frac{1}{TBCLK} = \frac{1}{100M} = 10 \text{ nsec} \quad (10)$$

现在您已经有了所有需要的参数，可以计算 TBPRD 值了：

$$TBPRD = \frac{T_{PWM}}{2 * T_{TBCLK}} \rightarrow \frac{2.5 \mu\text{sec}}{2 * 10 \text{ nsec}} = 125 \quad (11)$$

这就意味着时基计数器从零开始计数到 125，然后再返回零。这计为 ePWM 输出的一个周期，为 EPWM1、EPWM2 和 EPWM3 产生 400kHz 的输出频率。

EPWM Time Base	
Emulation Mode	Stop after next Time Base counter increment or decrement
Time Base Clock Divider	Divide clock by 1
High Speed Clock Divider	Divide clock by 1
Time Base Period	125
Time Base Period Link	Disable Linking
Enable Time Base Period Global Load	<input type="checkbox"/>
Time Base Period Load Mode	PWM Period register access is through shadow register
Time Base Period Load Event	Shadow to active load occurs when time base counter reaches 0
Initial Counter Value	0
Counter Mode	Up - down - count mode
Counter Mode After Sync	Count down after sync event
Enable Phase Shift Load	<input type="checkbox"/>
Sync In Pulse Source	Disable Sync-in
Sync Out Pulse	None
One-Shot Sync Out Trigger	Trigger is OSHT sync
Force a Sync Pulse	<input type="checkbox"/>

图 3-1. ePWM 时基：设置 TBPRD 和计数模式

以下代码由 SysConfig 生成：

```
// EPWM1 (the code is the same for EPWM2 and EPWM3, except for the base address)
EPWM_setClockPrescaler(myEPWM1_BASE, EPWM_CLOCK_DIVIDER_1, EPWM_HSCLOCK_DIVIDER_1);
EPWM_setTimeBasePeriod(myEPWM1_BASE, 125);
EPWM_setTimeBaseCounterMode(myEPWM1_BASE, EPWM_COUNTER_MODE_UP_DOWN);
```

3.2 应用相移

另一个关注领域是在 EPWM1、EPWM2 和 EPWM3 之间应用相移。这也是通过 TB 子模块实现的。

要计算时基相移 (TBPHS) 值，请使用 [方程式 12](#) 中所示的公式：

$$TBPHS = \frac{TBPRD * \text{Desired Phase Degree}}{360^\circ} \quad (12)$$

同步源为 EPWM1，这意味着它会将同步信号驱动到 EPWM2 和 EPWM3。EPWM2 与 EPWM1 之间的相移为 120° 。

$$TBPHS = \frac{125 * 120^\circ}{360^\circ} = \sim 42 \quad (13)$$

EPWM3 与 EPWM1 之间的相移为 240° ：

$$TBPHS = \frac{125 * 240^\circ}{360^\circ} = \sim 83 \quad (14)$$

3.3 设置同步 (Sync) 方案

[节 3.3](#) 展示了如何在 SysConfig 中设置相移值。

现在所需的时基相移值是已知的，您可以在三个 ePWM 模块之间设置同步方案。ePWM 4 类模块有两种不同的同步方案。我们将重点介绍 4 类模块的最新同步方案。对于这种同步方案，每个 ePWM 模块都有一个同步输入 (SYNCIN)、一个同步输出 (SYNCO) 和一个外设同步输出 (SYNCPER)。

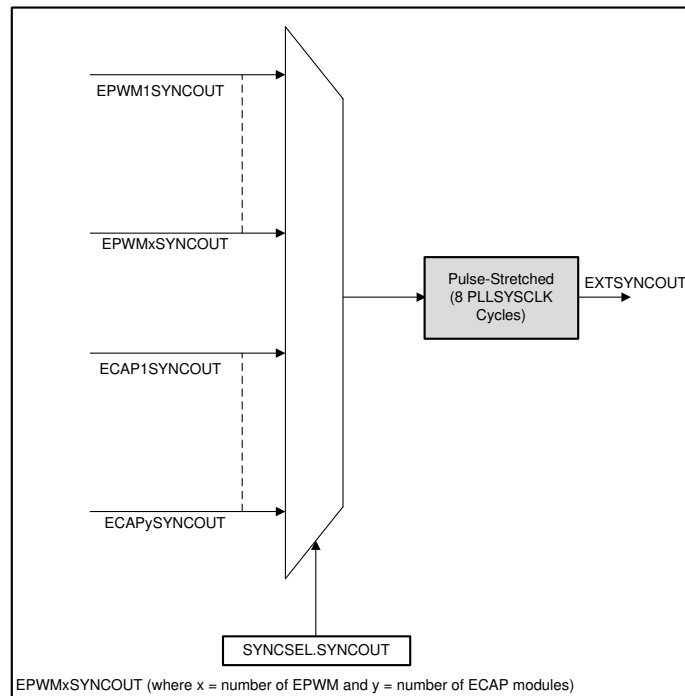


图 3-2. 同步方案

由于 EPWM1 是同步源，因此只需设置 SYNCO，它可驱动同步链。EPWM1 的 SYNCO 有很多备选择，例如时基计数器等于零或周期时。在此用例中，当 EPWM1 的时基计数器等于零时，可以生成 SYNCO。

EPWM Time Base	
Emulation Mode	Stop after next Time Base counter increment or decrement
Time Base Clock Divider	Divide clock by 1
High Speed Clock Divider	Divide clock by 1
Time Base Period	125
Time Base Period Link	Disable Linking
Enable Time Base Period Global Load	<input type="checkbox"/>
Time Base Period Load Mode	PWM Period register access is through shadow register
Time Base Period Load Event	Shadow to active load occurs when time base counter reaches 0
Initial Counter Value	0
Counter Mode	Up - down - count mode
Counter Mode After Sync	Count down after sync event
Enable Phase Shift Load	<input type="checkbox"/>
Sync In Pulse Source	Disable Sync-in
Sync Out Pulse	Counter zero event generates EPWM sync-out pulse
One-Shot Sync Out Trigger	Trigger is OSHT sync
Force a Sync Pulse	<input type="checkbox"/>

图 3-3. EPWM 时基 : EPWM1 作为同步源的配置

以下代码由 SysConfig 为 EPWM1 生成 :

```
EPWM_enableSyncOutPulseSource(myEPWM1_BASE, EPWM_SYNC_OUT_PULSE_ON_CNTR_ZERO);
```

对于 EPWM2 和 EPWM3, SYNCIN 来自 EPWM1 的 SYNCO。只要时基计数器等于零,就会出现 SYNCO。这样 EPWM2 和 EPWM3 都会从 EPWM1 接收到 SYNCO 信号。

对于 EPWM2 和 EPWM3 等同步接收器,启用相移并提供相移值是非常重要的。此配置的示例如图 3-4 所示。

EPWM Time Base	
Emulation Mode	Stop after next Time Base counter increment or decrement
Time Base Clock Divider	Divide clock by 1
High Speed Clock Divider	Divide clock by 1
Time Base Period	125
Time Base Period Link	Disable Linking
Enable Time Base Period Global Load	<input type="checkbox"/>
Time Base Period Load Mode	PWM Period register access is through shadow register
Time Base Period Load Event	Shadow to active load occurs when time base counter reaches 0
Initial Counter Value	0
Counter Mode	Up - down - count mode
Counter Mode After Sync	Count down after sync event
Enable Phase Shift Load	<input checked="" type="checkbox"/>
Phase Shift Value	42
Sync In Pulse Source	Sync-in source is EPWM1 sync-out signal
Sync Out Pulse	Counter zero event generates EPWM sync-out pulse
One-Shot Sync Out Trigger	Trigger is OSHT sync
Force a Sync Pulse	<input type="checkbox"/>

图 3-4. ePWM 时基 : EPWM2 同步接收器设置的配置

以下代码由 SysConfig 为 EPWM2 和 EPWM3 生成 :

```
// EPWM 2
EPWM_enablePhaseShiftLoad(myEPWM2_BASE);
EPWM_setPhaseShift(myEPWM2_BASE, 42);
```

```

EPWM_setSyncInPulseSource(myEPWM2_BASE, EPWM_SYNC_IN_PULSE_SRC SYNCOUT_EPWM1);
EPWM_enableSyncOutPulseSource(myEPWM2_BASE, EPWM_SYNC_OUT_PULSE_ON_CNTR_ZERO);
// EPWM 3
EPWM_enablePhaseShiftLoad(myEPWM3_BASE);
EPWM_setPhaseShift(myEPWM3_BASE, 83);
EPWM_setSyncInPulseSource(myEPWM3_BASE, EPWM_SYNC_IN_PULSE_SRC SYNCOUT_EPWM1);
EPWM_enableSyncOutPulseSource(myEPWM3_BASE, EPWM_SYNC_OUT_PULSE_ON_CNTR_ZERO);
    
```

4 计数器比较 (CC) 和动作限定器 (AQ) 子模块

4.1 计算占空比

此用例的要求之一是具有 45% 的初始占空比。每个 ePWM 的占空比基于计数器比较值计算，以及出现计数器比较匹配时采取的措施。当时基计数器递增或递减时，它会不断检查零、计数器比较或周期匹配。如果出现匹配，则可以将 ePWM 输出编程为变为低电平（清除）、变为高电平（设置）、切换或不执行任何操作。

在本例中，动作限定器事件的定义方式如下：

对于 EPWMxA：

- CMPA 向上 -> 设置
- CMPA 向下 -> 清除

对于 EPWMxB：

- CMPB 向上 -> 设置
- CMPB 向下 -> 清除

节 4.1 介绍了如何配置 AQ 子模块。节 9 介绍了全局加载特性。

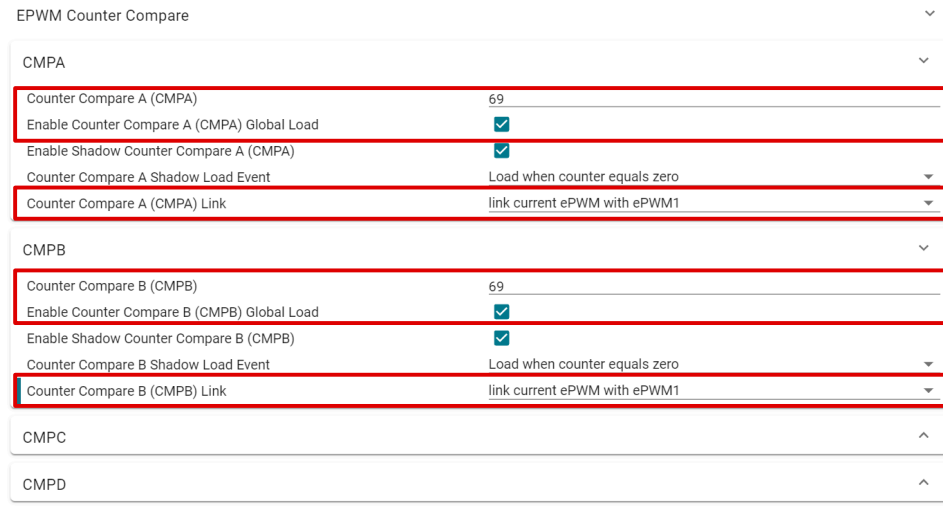
EPWM Action Qualifier	
Enable Continuous SW Force Global Load	<input type="checkbox"/>
Continuous SW Force Shadow Mode	Shadow mode load when counter equals zero
ePWMxA Output Configuration	
ePWMxA Global Load Enable	<input checked="" type="checkbox"/>
ePWMxA Shadow Mode Enable	<input checked="" type="checkbox"/>
ePWMxA Shadow Load Event	Load when counter equals zero
ePWMxA T1 Trigger Source	Digital compare event A 1
ePWMxA T2 Trigger Source	Digital compare event A 1
ePWMxA One-Time SW Force Action	No change in the output pins
ePWMxA Continuous SW Force Action	Software forcing disabled
Events to Configure for ePWMxA output	None
ePWMxA Event Output Configuration	
ePWMxA Time base counter equals zero	No change in the output pins
ePWMxA Time base counter equals period	No change in the output pins
ePWMxA Time base counter up equals COMPA	Set output pins to High
ePWMxA Time base counter down equals COMPA	Set output pins to low
ePWMxA Time base counter up equals COMPB	No change in the output pins
ePWMxA Time base counter down equals COMPB	No change in the output pins
ePWMxA T1 event on count up	No change in the output pins

图 4-1. ePWM 动作限定器：设置全局加载和输出事件

为了获得 45% 的初始占空比，您需要确定为 CMPA 和 CMPB 设置的值。

鉴于在向上和向下双向计数，我们的时基计数器在一个周期内总共计数 250 (2*125) 次。由于所选的设置，对于输出 A，在 2*(TBPRD-CMPA) 持续时间内，波形为“ON”，对于输出 B，在 2*(TBPRD-CMPB) 持续时间内，波形为“ON”。根据这些信息，您可以计算出 CMPA 和 CMPB 值，缩写为 CMPX。

为了实现 45% 的初始占空比，需要根据所需的动作限定器事件将 CMPA 和 CMPB 设置为 69。



EPWM Counter Compare	
CMPA	
Counter Compare A (CMPA)	69
Enable Counter Compare A (CMPA) Global Load	<input checked="" type="checkbox"/>
Enable Shadow Counter Compare A (CMPA)	<input checked="" type="checkbox"/>
Counter Compare A Shadow Load Event	Load when counter equals zero
Counter Compare A (CMPA) Link	link current ePWM with ePWM1
CMPB	
Counter Compare B (CMPB)	69
Enable Counter Compare B (CMPB) Global Load	<input checked="" type="checkbox"/>
Enable Shadow Counter Compare B (CMPB)	<input checked="" type="checkbox"/>
Counter Compare B Shadow Load Event	Load when counter equals zero
Counter Compare B (CMPB) Link	link current ePWM with ePWM1
CMPC	
CMPD	

图 4-2. ePWM 计数器比较：设置计数器比较值

以下代码由 SysConfig 生成：

```

// EPWM 1
EPWM_setCounterCompareValue(myEPWM1_BASE, EPWM_COUNTER_COMPARE_A, 69);
EPWM_enableGlobalLoadRegisters(myEPWM1_BASE, EPWM_GL_REGISTER_CMPA_CMPAHR);
EPWM_setCounterCompareValue(myEPWM1_BASE, EPWM_COUNTER_COMPARE_B, 69);
EPWM_enableGlobalLoadRegisters(myEPWM1_BASE, EPWM_GL_REGISTER_CMPB_CMPBHR);
// EPWM 2 (the code is the same for EPWM3, except for the base address)
EPWM_setCounterCompareValue(myEPWM2_BASE, EPWM_COUNTER_COMPARE_A, 69);
EPWM_enableGlobalLoadRegisters(myEPWM2_BASE, EPWM_GL_REGISTER_CMPA_CMPAHR);
EPWM_setupEPWMLinks(myEPWM2_BASE, EPWM_LINK_WITH_EPWM_1, EPWM_LINK_COMP_A);
EPWM_setCounterCompareValue(myEPWM2_BASE, EPWM_COUNTER_COMPARE_B, 69);
EPWM_enableGlobalLoadRegisters(myEPWM2_BASE, EPWM_GL_REGISTER_CMPB_CMPBHR);
EPWM_setupEPWMLinks(myEPWM2_BASE, EPWM_LINK_WITH_EPWM_1, EPWM_LINK_COMP_B);
  
```

5 死区 (DB) 子模块

5.1 设置信号对

对于此用例，三个 ePWM 模块均设置为高电平有效互补信号。这就意味着 ePWMxA 在上升沿延迟 (RED) 时保持不变，而 ePWMB 具有与 ePWMA 相同的源信号，但它反转为下降沿延迟 (FED)，而不是上升沿延迟。图 5-1 展示了这两个信号。

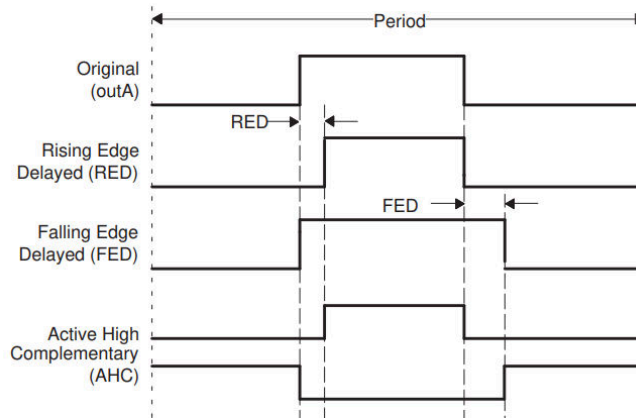


图 5-1. 高电平有效互补信号对

为了设置这些信号对，请利用死区子模块，因为它支持您应用此输出方案，以及许多其他信号对。

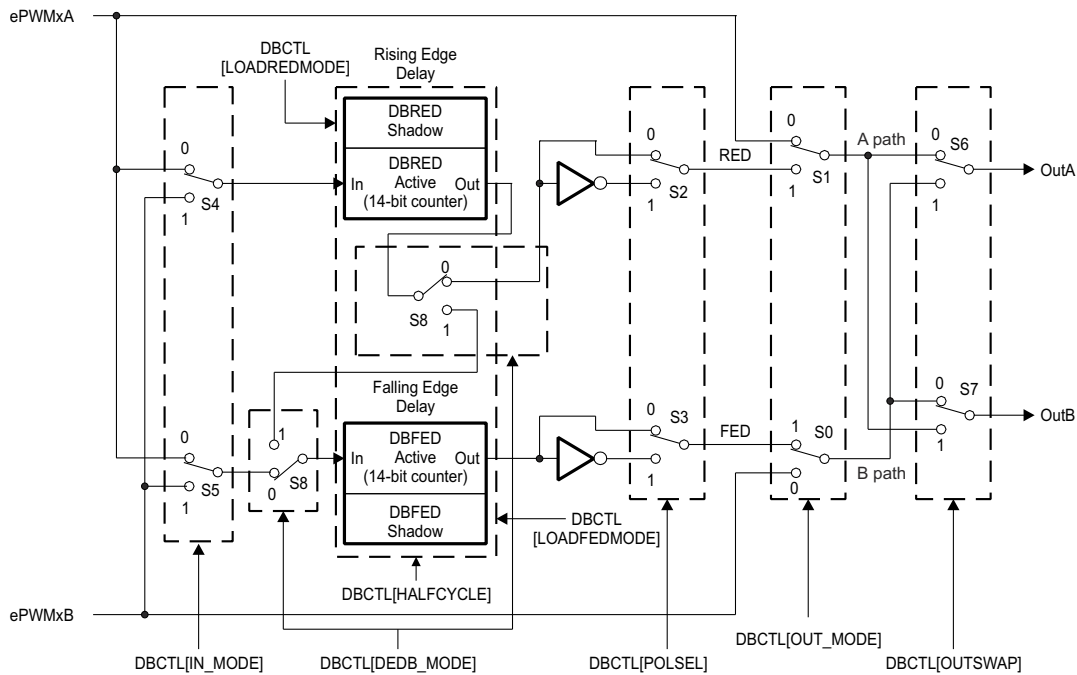


图 5-2. 死区子模块的配置选项

首先来看一下死区子模块，它本质上是一组开关，可协同工作以实现所需的输出。S4 和 S5 决定该子模块其余部分的源信号：ePWMA 或 ePWMB。由于您希望生成的 ePWMA 和 ePWMB 输出具有相同的源信号，请将两个开关都设置为 ePWMA 输入。接下来是 S8，它负责决定哪个输出将具有 RED 和 FED，或同时应用二者。只有一个通道 (A 或 B) 可同时具有 RED 和 FED。

在本例中，ePWMA 将具有 RED，ePWMB 将具有 FED，因此开关设置为零位置。由于 ePWMB 是反转的，S3 处于 1 位置，S2 仍处于 0 位置。由于使用了死区模块，因此 S0 和 S1 都设置为 1。最后一步是决定是否交换电流输出。对于此用例，S7 和 S6 将保持原样，这意味着当前设置即是应用的设置，而不是交换后的输出版本。

图 5-3 展示了 SysConfig 中的此配置。按下“Active High Complementary”选项旁的“SETUP THE DEAD-BAND MODULE”按钮，SysConfig 工具会以正确的方式自动设置此模块，以实现高电平有效互补对。

要计算所需的死区上升沿延迟 (DBRED) 和死区下降沿延迟 (DBFED)，请执行以下计算：

$$RED = DBRED * T_{TBCLK} \quad (15)$$

整理公式，得出 DBRED 的值，这就是所需的值：

$$DBRED = \frac{RED}{T_{TBCLK}} \quad (16)$$

时基时钟的周期可通过本应用报告中之前介绍的公式得出：

$$DBRED = \frac{200 \text{ nsec}}{10 \text{ nsec}} = 20 \quad (17)$$

备注

以上方法与得出 DBFED 值的方法相同。

EPWM Dead-Band

Common Dead-Band Modes Mode for the Dead-Band Submodule

Active High	SETUP THE DEAD-BAND MODULE
Active Low	SETUP THE DEAD-BAND MODULE
Active High Complementary	SETUP THE DEAD-BAND MODULE
Active Low Complementary	SETUP THE DEAD-BAND MODULE
Dual Edge Delay Mode	SETUP THE DEAD-BAND MODULE

Rising Edge Delay Input	Input signal is ePWMA
Falling Edge Delay Input	Input signal is ePWMA
Rising Edge Delay Polarity	DB polarity is not inverted
Falling Edge Delay Polarity	DB polarity is inverted
Enable Rising Edge Delay	<input checked="" type="checkbox"/>
Rising Edge Delay Value	20
Enable Falling Edge Delay	<input checked="" type="checkbox"/>
Falling Edge Delay Value	20
Swap Output for EPWMxA	<input type="checkbox"/>
Swap Output for EPWMxB	<input type="checkbox"/>

图 5-3. ePWM 死区：高电平有效互补

```
// EPWM 1 (the code is the same for EPWM2 and EPWM3, except for the base address)
EPWM_setDeadBandDelayPolarity(myEPWM1_BASE, EPWM_DB_FED, EPWM_DB_POLARITY_ACTIVE_LOW);
EPWM_setDeadBandDelayMode(myEPWM1_BASE, EPWM_DB_RED, true);
EPWM_setRisingEdgeDelayCount(myEPWM1_BASE, 20);
EPWM_setDeadBandDelayMode(myEPWM1_BASE, EPWM_DB_FED, true);
EPWM_setFallingEdgeDelayCount(myEPWM1_BASE, 20);
```

6 验证输出

在应用的这个阶段，应验证 ePWM 的初始设置，特别是占空比、死区时间和相移。

6.1 检查占空比和死区时间插入

首先要验证死区的正占空比。EPWM1、EPWM2 和 EPWM3 所需的频率为 400kHz。如方程式 6 所示，ePWM 输出的周期应为 $2.5 \mu\text{s}$ 。根据方程式 11 可知，TBPRD 值为 125，这意味着 ePWM 模块的时基计数器会从 0 计数到 125，然后再减至 0，因为计数器模式设置为向上向下双向。这样在一个周期中可产生总计 250 个计数。因此，根据方程式 18 可知，时基计数器的每个计数均为 10ns。

$$\text{Time for one count of the time base counter} = \frac{T_{\text{PWM}}}{\text{TBPRD} * 2} = \frac{2.5 \mu\text{sec}}{250} = 10 \text{ nsec} \quad (18)$$

计数器比较值设置为 69。对于 A 输出，当时基计数器计数递增至与计数器比较值相等时，输出设置为高电平。当时基计数器递减计数并且与计数器比较值匹配时，输出应变为低电平。但是，死区子模块也已设置为在输出中包含死区时间。通过方程式 17 可知，为了使死区时间为 200ns，DBRED 和 DBFED 可设置为 20，如图 6-2 所示。因此，对于 EPWMxA，当时基计数器等于 69 (CMPA 值) 时，输出不会变为高电平，而是变为 89，因为上升沿插入了 20 个死区时间计数。方程式 20 展示了如何根据此信息计算占空比。

$$\text{Duty Cycle} = \frac{\text{ON Time}}{\text{ON Time} + \text{OFF Time}} \quad (19)$$

$$\text{Duty Cycle} = \frac{(\text{TBPRD} - (\text{CMPA} + \text{DBRED})) + (\text{TBPRD} - \text{CMPA})}{\text{TBPRD} * 2} \quad (20)$$

$$\text{Duty Cycle} = \frac{(125 - 89) + (125 - 69)}{\text{TBPRD} * 2} = \frac{92}{250} = 36.8 \% \quad (21)$$

按时间计算， $92 * 10\text{ns}$ 等于 920ns，因此正脉冲宽度应为 920ns，如图 6-1 所示。

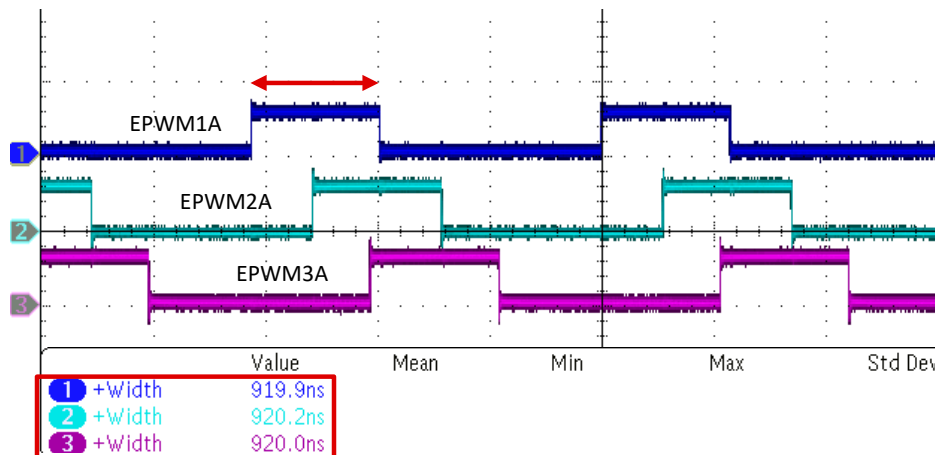


图 6-1. ePWM 输出正占空比的示波器捕获

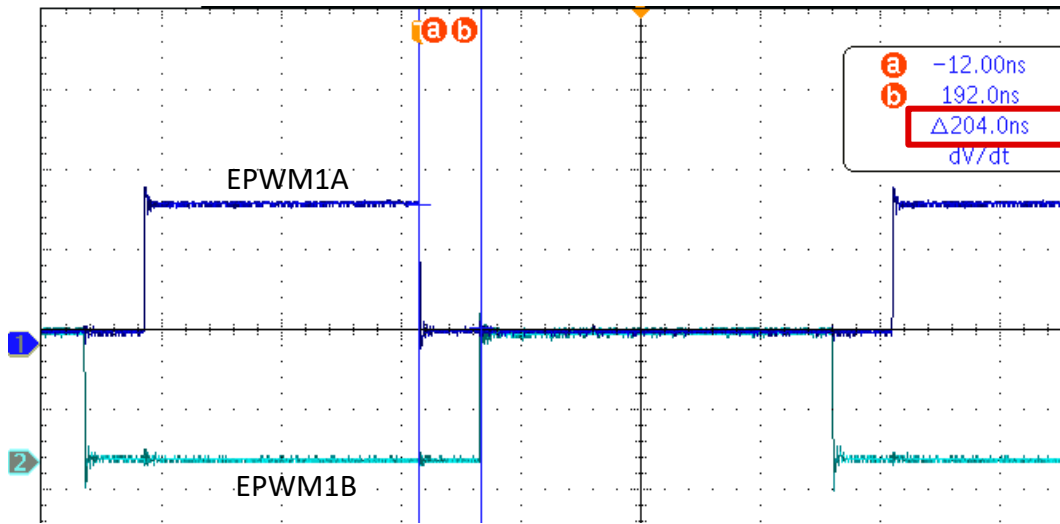


图 6-2. 插入死区时间的 ePWM 输出示波器捕获

6.2 检查应用的相移

三个 ePWM 输出会发生相移。EPWM2 相对于 EPWM1 具有 120° 相移，而 EPWM3 相对于 EPWM1 具有 240° 相移。从方程式 13 和方程式 14 中可以看出，TBPHS 的 EPWM2 设为 42、EPWM3 设为 83。根据设置的同步方案，每次 EPWM1 的时基计数器等于 0 时，都会产生一个同步脉冲，其中 EPWM2 和 EPWM3 的时基计数器设置为 TBPHS 寄存器中的值。

备注

如果 $TBCLK = EPWMCLK$ ，则内部同步源模块到同步接收模块的延迟为 EPWMCLK 的 2 倍。

根据注释，EPWM2 和 EPWM3 的时基计数器未设置为 TBPHS 值，而是 $TBPHS + 2$ 。例如，对于 EPWM2，TBPHS 为 42 个计数，这意味着 $42 * 10ns = 420ns$ ，但考虑到额外的周期为 $2 * 10ns = 20ns$ ，则总共为 440ns。图 6-3 展示了 EPWM1A 和 EPWM2A 上升沿之间的延时时间，用于展示相位差。

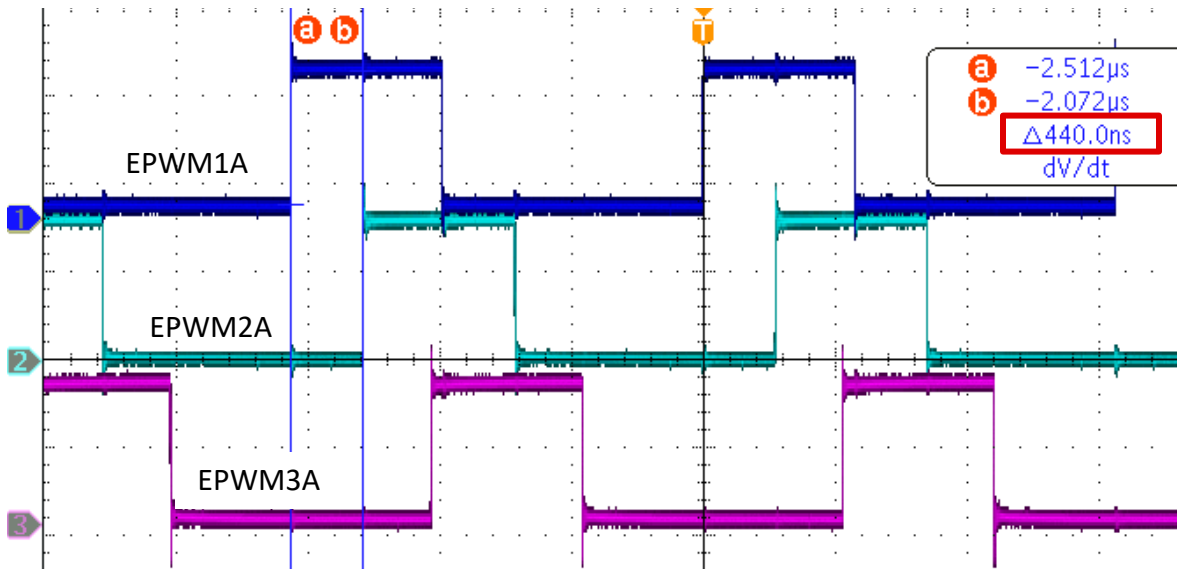


图 6-3. 具有相移的 EPWM 输出的示波器捕获

7 跳闸区域 (TZ) 和数字比较 (DC) 子模块

7.1 通过 CMPSS 设置跳闸条件时，在 ePWM 周期内将输出驱动为低电平

电流限制操作使用逐周期 (CBC) 跳闸。检测到逐周期跳闸时，跳闸区域子模块会将 EPWMxA 和 EPWMxB 驱动为某种指定状态。在发生下一次 ZRO、PRD 或 ZRO/PRD 事件时，输出会恢复跳闸前的状态。

比较器子系统 (CMPSS) 中的一个比较器设置为引发跳闸事件，然后在 EPWM2 上生成 CBC 跳闸。

CMPSS 包含模拟比较器和支持电路，这些电路对于电源应用非常有用。比较器具有正负输入。当正输入上的电压大于负输入上的电压时，会产生高数字输出；当正输入上的电压小于负输入上的电压时，会产生低数字输出。

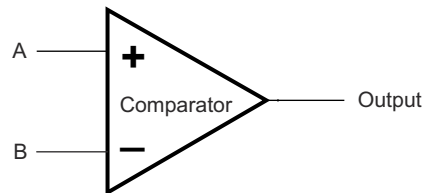


图 7-1. 比较器块图

为了产生高数字输出，当正输入（电压 A）大于负输入（电压 B）时需要跳闸。因此，您需要对正输入连接的外部电压源进行监控，以确保其不会超出特定状态。负输入可以来自器件上的基准数模转换器 (DAC)。对于此应用，将值 2500 写入 DACVALA，该值驱动至负端子。

基准 12 位 DAC 的理想输出可通过以下公式进行计算：

$$\text{DACOUT} = \frac{\text{DACVALA} * \text{DACREF}}{4096} \quad (22)$$

如果应用已知值，会得到 2V 的 DACOUT 值。因此，只要施加到比较器正输入的电压超过 2V，就会触发跳闸信号。

$$\text{DACOUT} = \frac{2500 * 3.3}{4096} = 2.01 \text{ V} \quad (23)$$

在此用例中使用比较器 1。图 7-3 显示了 CMPSS1 模块根据上述计算进行的设置。

备注

比较器引脚通常与其他模拟功能共用。要确定比较器 1 正输入的模拟引脚，请参阅特定器件 TRM 中的模拟子系统部分。以 F2838x 器件为例，CMPIN1P 与 ADCINA2 共用，所以电压将被施加到器件的 A2 引脚。

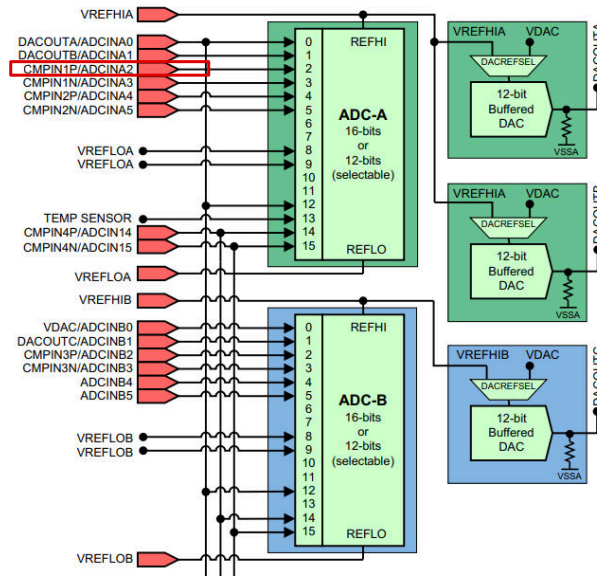


图 7-2. 比较器引脚

Name	myCMPSS1
CMPSS Instance	CMPSS1
Enable module	<input checked="" type="checkbox"/>
High Comparator Configuration	
Negative input source	Input driven by internal DAC
Output is inverted	<input type="checkbox"/>
Asynch OR Latch	<input type="checkbox"/>
Signal driving CTRIPOUTH	Asynchronous comparator output drives CTRIPOUTH
Signal driving CTRIPH	Asynchronous comparator output drives CTRIPH
Set high comparator DAC value	2500
Digital Filter Configuration	
Ramp Generator Configuration	
Low Comparator Configuration	
DAC Configuration	
Hysteresis	None
Blanking Signal	EPWM1BLANK
Enable Blanking Signal	<input type="checkbox"/>

图 7-3. CMPSS 配置

以下代码是从 SysConfig 为 CMPSS 配置生成的：

```

void CMPSS_init(){
// myCMPSS1 initialization
// Sets the configuration for the high comparator.
CMPSS_configHighComparator(myCMPSS1_BASE, (CMPSS_INSRC_DAC));
// Sets the configuration for the high comparator.
CMPSS_configLowComparator(myCMPSS1_BASE, (CMPSS_INSRC_DAC));
// Sets the configuration for the internal comparator DACs.
CMPSS_configDAC(myCMPSS1_BASE, (CMPSS_DACVAL_SYCLK | CMPSS_DACREF_VDDA | CMPSS_DACSRC_SHDW));
// Sets the value of the internal DAC of the high comparator.
CMPSS_setDACValueHigh(myCMPSS1_BASE, 2500U);
// Sets the value of the internal DAC of the low comparator.
CMPSS_setDACValueLow(myCMPSS1_BASE, 0U);
// Configures the digital filter of the high comparator.
CMPSS_configFilterHigh(myCMPSS1_BASE, 0U, 1U, 1U);
// Configures the digital filter of the low comparator.
CMPSS_configFilterLow(myCMPSS1_BASE, 0U, 1U, 1U);
// Sets the output signal configuration for the high comparator.
CMPSS_configOutputsHigh(myCMPSS1_BASE, (CMPSS_TRIPOUT_ASYNC_COMP | CMPSS_TRIP_ASYNC_COMP));
// Sets the output signal configuration for the low comparator.
CMPSS_configOutputsLow(myCMPSS1_BASE, (CMPSS_TRIPOUT_ASYNC_COMP | CMPSS_TRIP_ASYNC_COMP));
// Sets the comparator hysteresis settings.
CMPSS_setHysteresis(myCMPSS1_BASE, 0U);
// Configures the comparator subsystem's ramp generator.
CMPSS_configRamp(myCMPSS1_BASE, 0U, 0U, 0U, 1U, true);
// Disables reset of HIGH comparator digital filter output latch on PWMSYNC
CMPSS_disableLatchResetOnPWMSYNCHigh(myCMPSS1_BASE);
// Disables reset of LOW comparator digital filter output latch on PWMSYNC
CMPSS_disableLatchResetOnPWMSYNCLow(myCMPSS1_BASE);
// Sets the ePWM module blanking signal that holds trip in reset.
CMPSS_configBlanking(myCMPSS1_BASE, 1U);
// Disables an ePWM blanking signal from holding trip in reset.
CMPSS_disableBlanking(myCMPSS1_BASE);
// Configures whether or not the digital filter latches are reset by PWMSYNC
CMPSS_configLatchOnPWMSYNC(myCMPSS1_BASE, false, false);
// Enables the CMPSS module.
CMPSS_enableModule(myCMPSS1_BASE);
// Delay for CMPSS DAC to power up.
DEVICE_DELAY_US(500);
}
    
```

请务必注意，施加于比较器正输入的外部信号需要保持至少 3 倍于 TBCLK 的时间，才能正确检测到该信号并导致跳闸。

从比较器子系统到 ePWM 子模块的跳闸区域子模块之间没有直接路径，因此必须使用数字比较子模块来路由信号。要搞清楚如何通过数字比较子模块正确路由 CMPSS 信号，您必须从 ePWM X-BAR 入手。每个器件的 TRM 中都有一个表，用于介绍 ePWM X-BAR 的多路复用器定位，称为“ePWM X-BAR 多路复用器配置表”。节 7.1 展示了 ePWM X-BAR 配置表的一个示例。从图 7-4 中，您可以看到“CMPSS1.CTRIPH”位于多路复用器 0 的位置 0。在输入 X-BAR 配置中，您可以将其路由到 4 到 12 之间的任何跳闸信号；此应用选择了跳闸 4。

Mux	0	1	2	3
0	CMPSS1.CTRIPH	CMPSS1.CTRIPH_OR_CTRIPL	ADCAEVT1	ECAP1.OUT
1	CMPSS1.CTRIPL	INPUTXBAR1	CLB1_4	ADCCEVT1
2	CMPSS2.CTRIPH	CMPSS2.CTRIPH_OR_CTRIPL	ADCAEVT2	ECAP2.OUT
3	CMPSS2.CTRIPL	INPUTXBAR2	CLB1_5	ADCCEVT2
4	CMPSS3.CTRIPH	CMPSS3.CTRIPH_OR_CTRIPL	ADCAEVT3	ECAP3.OUT
5	CMPSS3.CTRIPL	INPUTXBAR3	CLB2_4	ADCCEVT3
6	CMPSS4.CTRIPH	CMPSS4.CTRIPH_OR_CTRIPL	ADCAEVT4	ECAP4.OUT
7	CMPSS4.CTRIPL	INPUTXBAR4	CLB2_5	ADCCEVT4
8	CMPSS5.CTRIPH	CMPSS5.CTRIPH_OR_CTRIPL	ADCBEVT1	ECAP5.OUT
9	CMPSS5.CTRIPL	INPUTXBAR5	CLB3_4	ADCDEVT1
10	CMPSS6.CTRIPH	CMPSS6.CTRIPH_OR_CTRIPL	ADCBEVT2	ECAP6.OUT
11	CMPSS6.CTRIPL	INPUTXBAR6	CLB3_5	ADCDEVT2

图 7-4. ePWM X-BAR 多路复用器配置表的部分示例

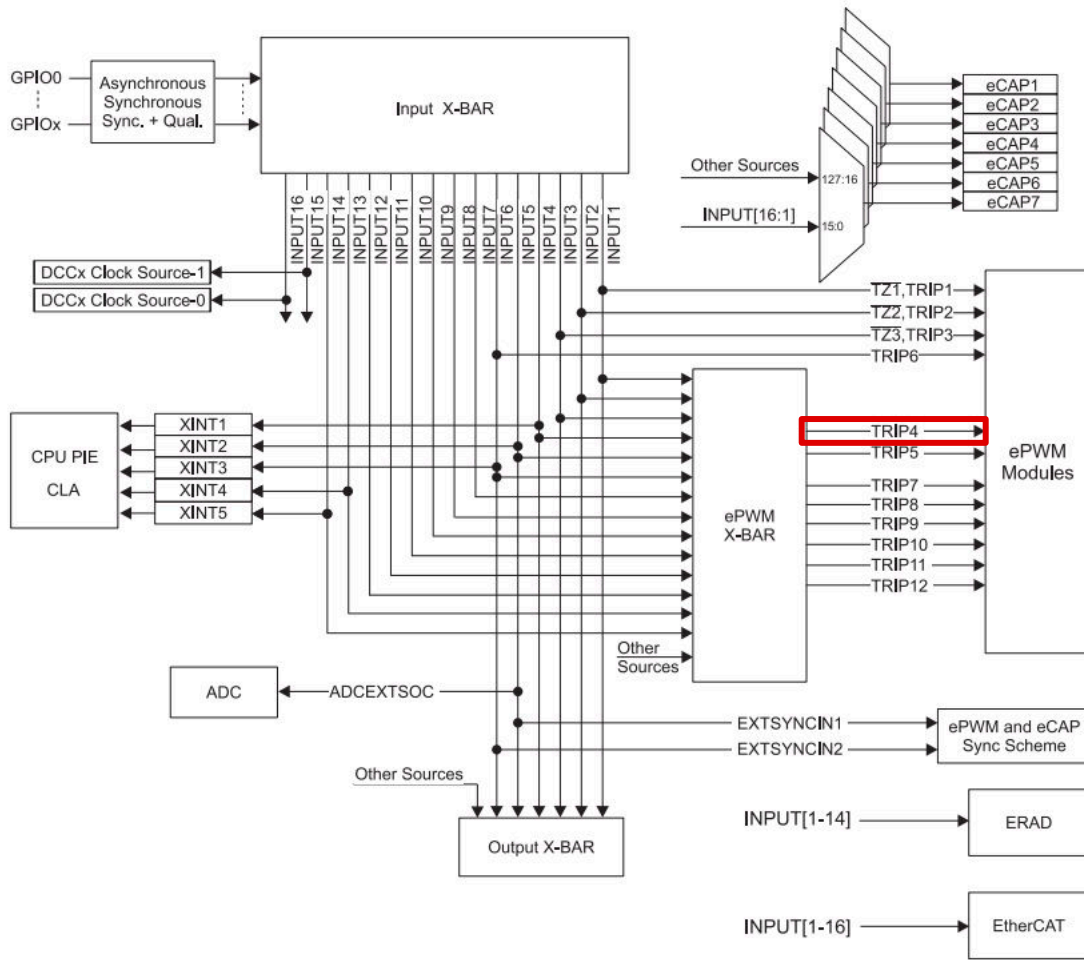


图 7-5. 输入 X-BAR 方框图

图 7-6 展示了如何在 SysConfig 的系统选项中配置“EPWMXBAR”。

Name	myEPWMXBAR4
Trip Input	TRIP4 of the ePWM X-BAR
Invert Mode	<input type="checkbox"/>
MUXes to be used	MUX 00
MUX 0 Config	CMPSS1 CTRIPH

图 7-6. EPWMXBAR 的 SysConfig 配置

以下代码由 SysConfig 生成：

```
void EPWMXBAR_init(){
//myEPWMXBAR4 initialization
XBAR_setEPWMmuxConfig(XBAR_TRIP4, XBAR_EPWM_MUX00_CMPSS1_CTRIPH);
XBAR_enableEPWMmux(XBAR_TRIP4, XBAR_MUX00);
}
```

直流子模块最多可生成四个事件：DCAEVT1、DCAEVT2、DCBEVT1 和 DCBEVT2。只有事件 2 能引发逐周期跳闸，对于此应用用例，您需要关注 DCAEVT2。数字比较事件有高电平 (DCAH) 和低电平 (DCAL) 信号。这两个信号可用于引发 DCAEVT2。在本例中，由于只需要一个信号，因此可以将 CMPSS1 输出路由到两个信号中的任何一个。我们选择 DCAH。产生事件的条件是此信号变为高电平，这意味着施加到比较器正输入的电压高于负输入的电压。

EPWM Digital Compare		▼
DCAEVT1 and DCAEVT2		▼
Digital Compare A High	Trip 4	▼
Combination Input Sources (Digital Compare A High)	None	▼
Digital Compare A Low	Trip 1	▼
Combination Input Sources (Digital Compare A Low)	None	▼
Condition For Digital Compare output 1 A	Event is disabled	▼
Condition For Digital Compare output 2 A	Event when DCxH high	▼
Generate ADC SOC (DCAEVT1)	<input type="checkbox"/>	
Generate SYNCOUT (DCAEVT1)	<input type="checkbox"/>	
Synch Mode (DCAEVT1)	DC input signal is synced with TBCLK	▼
Signal Source (DCAEVT1)	Signal source is unfiltered (DCAEVT1/2)	▼
CBC Latch Mode (DCAEVT1)	DC cycle-by-cycle(CBC) latch is disabled	▼
CBC Latch Clear Event (DCAEVT1)	Clear CBC latch when counter equals zero	▼
Synch Mode (DCAEVT2)	DC input signal is synced with TBCLK	▼
Signal Source (DCAEVT2)	Signal source is unfiltered (DCAEVT1/2)	▼
CBC Latch Mode (DCAEVT2)	DC cycle-by-cycle(CBC) latch is disabled	▼
CBC Latch Clear Event (DCAEVT2)	Clear CBC latch when counter equals zero	▼

图 7-7. ePWM 数字比较：设置 DCAEVT2

以下代码由 SysConfig 生成：

```
EPWM_selectDigitalCompareTripInput(myEPWM2_BASE, EPWM_DC_TRIP_TRIPIN4, EPWM_DC_TYPE_DCAH);
EPWM_setTripZoneDigitalCompareEventCondition(myEPWM2_BASE, EPWM_TZ_DC_OUTPUT_A2,
EPWM_TZ_EVENT_DCXH_HIGH);
```

现在，您可以设置跳闸区域设置，以便在发生 DCAEVT2 事件时通过逐周期跳闸将 A 和 B 输出驱动至低电平状态。每当发生跳闸时，还会生成一个中断，可用于清除标志。

EPWM Trip Zone ▼

Use Advanced EPWM Trip Zone Actions

TZA Event	Low voltage state	▼
TZB Event	Low voltage state	▼
DCAEVT1 Event	High impedance output	▼
DCAEVT2 Event	High impedance output	▼
DCBEVT1 Event	High impedance output	▼
DCBEVT2 Event	High impedance output	▼
One-Shot Source	None	▼
CBC Source	DCAEVT2 Cycle By Cycle	▼
CBC Latch Clear Signal	Clear CBC pulse when counter equals zero	▼
TZ Interrupt Source (ORed)	Trip Zones Cycle By Cycle interrupt	▼
Register Interrupt Handler	<input checked="" type="checkbox"/>	

图 7-8. ePWM 跳闸区域：基于 DCAEVT2 的 CBC 跳闸配置

以下代码由 SysConfig 生成：

```
EPWM_setTripZoneAction(myEPWM2_BASE, EPWM_TZ_ACTION_EVENT_TZA, EPWM_TZ_ACTION_LOW);
EPWM_setTripZoneAction(myEPWM2_BASE, EPWM_TZ_ACTION_EVENT_TZB, EPWM_TZ_ACTION_LOW);
EPWM_enableTripZoneSignals(myEPWM2_BASE, EPWM_TZ_SIGNAL_DCBEVT2);
```

备注

要查找正确的中断组，请参阅特定器件 TRM 中的 *PIE 通道映射表*。

```
void epwm2TZISR(void){
    epwm2TZIntCount++;

    // Clear the flags
    EPWM_clearTripZoneFlag(myEPWM2_BASE, (EPWM_TZ_INTERRUPT | EPWM_TZ_FLAG_CBC));

    // Acknowledge this interrupt to receive more interrupts from group 2
    Interrupt_clearACKGroup (INTERRUPT_ACK_GROUP2);
}
```

将 EPWM2 配置为具有跳闸条件后，当大于 2V 的正电压施加到比较器 1 的正输入时，EPWM2 应驱动为低电平。

7.2 通过 GPIO 设置跳闸条件时，通过软件将输出驱动为低电平，直到清除

另一种常见的跳闸方式是一次性跳闸 (OSHT)。一次性跳闸用于严重短路或过流情况。检测到一次性跳闸时，跳闸区域子模块会将 EPWMxA 和 EPWMxB 驱动为某种指定状态。输出将保持该状态，直到手动清除跳闸。

在我们的应用中，使用一个 GPIO 指示 EPWM3 上的跳闸条件。为了将 GPIO 连接到 ePWM 模块的跳闸源，您需要通过输入 X-BAR 对其进行路由。输入 X-BAR 可用于将任何 GPIO 映射到 ePWM X-BAR，或 ePWM 模块的跳闸源本身，具体取决于所需的信号。

对于此用例，请使用 GPIO 12 并将其路由到输入 X-BAR 的输入 1，它将直接馈送到 TZ1 (跳闸区域输入 1)。

Name	myGPIO12
Analog Mode	Pin is in digital mode
GPIO Direction	Pin is a GPIO input
Pin Type	Push-pull output/floating input
Qualification Mode	Synchronization to SYSCLKOUT
Master Core	CPU1 selected as master core
Write Initial Value	<input type="checkbox"/>
PinMux Peripheral and Pin Configuration	
GPIO	GPIO12

图 7-9. SysConfig 中的 GPIO 设置

Name	myINPUTXBAR1
INPUTs to be used	INPUTXBAR1
INPUTXBAR1	GPIO12
INPUTXBAR1 Lock	<input checked="" type="checkbox"/>

图 7-10. SysConfig 中的输入 X-BAR

以下代码由 SysConfig 生成：

```

Void GPIO_init(){
//myGPIO12 initialization
GPIO_setDirectionMode(myGPIO12, GPIO_DIR_MODE_IN);
GPIO_setPadConfig(myGPIO12, GPIO_PIN_TYPE_STD);
GPIO_setMasterCore(myGPIO12, GPIO_CORE_CPU1);
GPIO_setQualificationMode(myGPIO12, GPIO_QUAL_SYNC);
GPIO_writePin(myGPIO12, 1);
}
void INPUTXBAR_init(){
//myINPUTXBAR1 initialization
XBAR_setInputPin(INPUTXBAR_BASE, XBAR_INPUT1, 12);
XBAR_lockInput(INPUTXBAR_BASE, XBAR_INPUT1);
}
    
```

路由 GPIO 后，您就可以设置事件。第一步是选择跳闸 1 作为一次性跳闸条件。每次 GPIO 12 上发生跳闸（GPIO 的状态由高电平变为低电平）时，ePWM 输出都会变为低电平。所执行的清除操作是通过 TZCTL 寄存器的 TZA（用于 ePWMXA）和 TZB（用于 ePWMXB）位指定的。

EPWM Trip Zone ▼

Use Advanced EPWM Trip Zone Actions

TZA Event	Low voltage state	▼
TZB Event	Low voltage state	▼
DCAEVT1 Event	High impedance output	▼
DCAEVT2 Event	High impedance output	▼
DCBEVT1 Event	High impedance output	▼
DCBEVT2 Event	High impedance output	▼
One-Shot Source	One-shot TZ1	▼
CBC Source	None	▼
CBC Latch Clear Signal	Clear CBC pulse when counter equals zero	▼
TZ Interrupt Source (ORed)	Trip Zones One Shot interrupt	▼
Register Interrupt Handler	<input checked="" type="checkbox"/>	

图 7-11. 跳闸区域配置，包括一次性配置

以下代码由 SysConfig 生成：

```
EPWM_setTripZoneAction(myEPWM3_BASE, EPWM_TZ_ACTION_EVENT_TZA, EPWM_TZ_ACTION_LOW);
EPWM_setTripZoneAction(myEPWM3_BASE, EPWM_TZ_ACTION_EVENT_TZB, EPWM_TZ_ACTION_LOW);
EPWM_enableTripZoneSignals(myEPWM3_BASE, EPWM_TZ_SIGNAL_OSHT1);
EPWM_enableTripZoneInterrupt(myEPWM3_BASE, EPWM_TZ_INTERRUPT_OST);
```

与逐周期跳闸情况类似，一次性跳闸配置也设置为在跳闸发生时生成中断，以便清除中断标志。

```
void epwm3TZISR(void) {
    epwm3TZIntCount++
    //
    // Re-enable the OST Interrupt
    //
    EPWM_clearTripZoneFlag(myEPWM3_BASE, (EPWM_TZ_INTERRUPT | EPWM_TZ_FLAG_OST));
    //
    // Acknowledge this interrupt to receive more interrupts from group 2
    //
    Interrupt_clearACKGroup(INTERRUPT_ACK_GROUP2);
}
```

现在 GPIO12 已设置为在 EPWM3 上执行一次性跳闸，将 GPIO12 驱动为低电平，EPWM3 也应该变为低电平，直到该 GPIO 不再为低电平，并且一次性跳闸标志被清除。

8 事件触发 (ET) 子模块

8.1 设置时基中断

事件触发器子模块管理由 TB、CC 和 DC 子模块生成的事件，能够在发生所选事件时产生 CPU 中断，或开始向模数转换器 (ADC) 转换的脉冲。对于本示例，只要发生一次跳闸事件，就会生成中断（在节 7.1 和节 7.2 中进行了阐释）。要在 EPWM1 的时基计数器达到 0 时设置中断，请参阅节 8.1。

EPWM Event-Trigger ▼

Enable EPWM Interrupt	<input checked="" type="checkbox"/>
Register Interrupt Handler	<input checked="" type="checkbox"/>
Interrupt Event Sources	Time-base counter equal to zero ▼
Interrupt Event Count	1 Event Generates Interrupt ▼
Interrupt Event Count Initial Value Load Enable	<input type="checkbox"/>
Force Interrupt Event Count Initial Value	<input type="checkbox"/>

ADC SOC Trigger ▲

图 8-1. ePWM 事件触发器：设置 TBCTR=ZRO 时中断

以下代码由 SysConfig 生成：

```
// EPWM1
EPWM_enableInterrupt(myEPWM1_BASE);
EPWM_setInterruptSource(myEPWM1_BASE, EPWM_INT_TBCTR_ZERO);
EPWM_setInterruptEventCount(myEPWM1_BASE, 1);
```

9 全局加载

9.1 应用全局加载和一次性加载功能

全局加载是 ePWM 模块内的一项功能，支持将单一事件作为源，更新多个关键参数，例如 TBPRD、CMPA、CMPB、CMPC、CMPD、DBRED、DBFED、DBCTL、AQCTLA、AQCTLB 和 AQCSFRC。这些寄存器均有影子寄存器，允许将写入寄存器的内容保留在“影子”寄存器中，直到发生指定事件（例如 TBCTR=ZRO、PRD 或 ZRO|PRD 时），此时影子寄存器中的内容将被传输到活动寄存器。

一次性加载功能可与全局加载功能结合使用。在正常全局加载的情况下，导致所有活动寄存器全部更新的单一事件会定期发生。但在很多情况下，在生成全局加载信号的下一个周期性事件发生时，更新关键 ePWM 参数所需的新值并不能总是准备就绪。因此，只有所有内容都准备好进行更新时，才会使用一次性功能来更新活动寄存器。对于控制环路发生在异步 ISR 中的应用，此功能非常有用，例如不以 EPWM 的倍数频率运行的计时器或 ADC ISR。

在本应用报告的用例中，全局加载/一次性加载功能用于更新动作限定器设置以及计数器比较值。动作限定器和计数器比较子模块已配置全局加载设置。如需了解更多详细信息，请参阅节 4。

但是，仍需要配置 SysConfig 中的全局加载部分，以设置全面的全局加载和一次性加载功能。

EPWM Global Load	
Enable Global Shadow to Active Load	<input checked="" type="checkbox"/>
Global Load Pulse Selection	Load when counter is equal to zero
Global Load Strobe Period	Counter is disabled
One Shot Mode	<input checked="" type="checkbox"/>
Enable Reload Event in One Shot Mode	<input checked="" type="checkbox"/>
Force Load Event	<input type="checkbox"/>
Global PWM Load Link	link current ePWM with ePWM1

图 9-1. 全局加载 SysConfig 设置

以下代码由 SysConfig 生成。有关动作限定器和计数器比较子模块的详细信息，请参阅节 4。

```
// EPWM1
EPWM_enableGlobalLoad(myEPWM1_BASE); EPWM_enableGlobalLoadOneShotMode(myEPWM1_BASE);
EPWM_setGlobalLoadOneShotLatch(myEPWM1_BASE);
// EPWM2 (the code is the same for EPWM3, except for the base address)
EPWM_enableGlobalLoad(myEPWM2_BASE); EPWM_enableGlobalLoadOneShotMode(myEPWM2_BASE);
EPWM_setGlobalLoadOneShotEvent(myEPWM2_BASE);
EPWM_setupEPWMLinks(myEPWM2_BASE, EPWM_LINK_WITH_EPWM_1, EPWM_LINK_GLDCTL2);
```

9.2 链接 ePWM 模块

在许多应用（例如三相交错式 LLC 实现）中，能够针对不同模块同时更新关键 ePWM 参数非常有用。C2000 的 ePWM 4 类可通过链接功能实现此目标。多个 ePWM 模块可以链接到一起，以便同时更新 TBPRD、CMPA、CMPB、CMPC、CMPD 和 GLDCTL2（全局负载控制 2）寄存器。

在本例中，将 ePWM2 和 ePWM3 的 CMPA 和 CMPB 链接到 ePWM1，这意味着每次更新 ePWM1 的 CMPA 或 CMPB 值时，ePWM2/3 的 CMPA/CMPB 也会更新为相同的值。

图 4-2 和图 9-1 展示了这些参数的链接设置。

在主应用程序代码中，您可以使用链接功能，这样只需更新 EPWM1 的 CMPA 和 CMPB 即可。EPWM2 和 EPWM3 的 CMPA/CMPB 值将同时更新为相同的值。

动作限定器设置没有链接功能，因此三个 ePWM 模块的动作限定器设置都需要分别更新。但是，由于所有三个模块之间的一次性加载都已链接，并且这些设置的影子寄存器已由一次性加载信号锁存，因此可使用影子寄存器同时更新全部三个 ePWM 模块的 AQCTLA 设置。

9.3 通过全局加载更新动作限定器设置和计数器比较值

现在已设置全局加载和链接，下一步是编写应用程序代码，以修改动作限定器设置和计数器比较值。在此应用报告中，计数器比较值从 69 更改为 100，动作限定器设置为反转。对于 ePWMxA，CMPA 向上的设置已清除，并设为 CMPA 向下。以下代码显示了这些新设置，以及在设置更新后设置的全局加载一次性锁存。通常代码应集成到与 ePWM 周期异步的控制环路中。但是，为了更清楚地展示功能，它被放置在本应用报告程序的主“for”循环内。要启用新设置，请通过 CCS 的表达式窗口将变量“perform_one_shot_load”设置为 1。

```

if(perform_one_shot_load==1)
{
    //
    // EPWM1, EPWM2, EPWM3 are linked so only EPWM1 needs to be updated
    //
    EPWM_setCounterCompareValue(myEPWM1_BASE, EPWM_COUNTER_COMPARE_A, 100);
    EPWM_setCounterCompareValue(myEPWM1_BASE, EPWM_COUNTER_COMPARE_B, 100);
    //
    // Change the Action Qualifier Settings for EPWM1, EPWM2, and EPWM3
    //
    EPWM_setActionQualifierAction(myEPWM1_BASE, EPWM_AQ_OUTPUT_A, EPWM_AQ_OUTPUT_LOW,
        EPWM_AQ_OUTPUT_ON_TIMEBASE_UP_CMPA);
    EPWM_setActionQualifierAction(myEPWM1_BASE, EPWM_AQ_OUTPUT_A, EPWM_AQ_OUTPUT_HIGH,
        EPWM_AQ_OUTPUT_ON_TIMEBASE_DOWN_CMPA);
    EPWM_setActionQualifierAction(myEPWM2_BASE, EPWM_AQ_OUTPUT_A, EPWM_AQ_OUTPUT_LOW,
        EPWM_AQ_OUTPUT_ON_TIMEBASE_UP_CMPA);
    EPWM_setActionQualifierAction(myEPWM2_BASE, EPWM_AQ_OUTPUT_A, EPWM_AQ_OUTPUT_HIGH,
        EPWM_AQ_OUTPUT_ON_TIMEBASE_DOWN_CMPA);
    EPWM_setActionQualifierAction(myEPWM3_BASE, EPWM_AQ_OUTPUT_A, EPWM_AQ_OUTPUT_LOW,
        EPWM_AQ_OUTPUT_ON_TIMEBASE_UP_CMPA);
    EPWM_setActionQualifierAction(myEPWM3_BASE, EPWM_AQ_OUTPUT_A, EPWM_AQ_OUTPUT_HIGH,
        EPWM_AQ_OUTPUT_ON_TIMEBASE_DOWN_CMPA);
    //
    // EPWM1, EPWM2, EPWM3 are linked
    //
    EPWM_setGlobalLoadOneShotLatch(myEPWM1_BASE);
    //
    // Clear the setting
    //
    perform_one_shot_load = 0;
}

```

备注

请记住将“perform_one_shot_load”作为程序全局变量的一部分声明。

备注

对于使用 C2000Ware 版本 4.01 或更低版本的项目，需要使用以下权变措施，以确保在 SysConfig 中启用全局加载功能后不会清除动作限定器设置。在主函数的“board_init()”函数调用后，插入以下代码。

```

EPWM_setActionQualifierAction(myEPWM1_BASE, EPWM_AQ_OUTPUT_A, EPWM_AQ_OUTPUT_HIGH,
EPWM_AQ_OUTPUT_ON_TIMEBASE_UP_CMPA);
EPWM_setActionQualifierAction(myEPWM1_BASE, EPWM_AQ_OUTPUT_A, EPWM_AQ_OUTPUT_LOW,
EPWM_AQ_OUTPUT_ON_TIMEBASE_DOWN_CMPA);
EPWM_setActionQualifierAction(myEPWM1_BASE, EPWM_AQ_OUTPUT_B, EPWM_AQ_OUTPUT_LOW,
EPWM_AQ_OUTPUT_ON_TIMEBASE_UP_CMPB);
EPWM_setActionQualifierAction(myEPWM1_BASE, EPWM_AQ_OUTPUT_B, EPWM_AQ_OUTPUT_HIGH,
EPWM_AQ_OUTPUT_ON_TIMEBASE_DOWN_CMPB);

EPWM_setActionQualifierAction(myEPWM2_BASE, EPWM_AQ_OUTPUT_A, EPWM_AQ_OUTPUT_HIGH,
EPWM_AQ_OUTPUT_ON_TIMEBASE_UP_CMPA);
EPWM_setActionQualifierAction(myEPWM2_BASE, EPWM_AQ_OUTPUT_A, EPWM_AQ_OUTPUT_LOW,
EPWM_AQ_OUTPUT_ON_TIMEBASE_DOWN_CMPA);
EPWM_setActionQualifierAction(myEPWM2_BASE, EPWM_AQ_OUTPUT_B, EPWM_AQ_OUTPUT_LOW,
EPWM_AQ_OUTPUT_ON_TIMEBASE_UP_CMPB);
EPWM_setActionQualifierAction(myEPWM2_BASE, EPWM_AQ_OUTPUT_B, EPWM_AQ_OUTPUT_HIGH,
EPWM_AQ_OUTPUT_ON_TIMEBASE_DOWN_CMPB);

EPWM_setActionQualifierAction(myEPWM3_BASE, EPWM_AQ_OUTPUT_A, EPWM_AQ_OUTPUT_HIGH,
EPWM_AQ_OUTPUT_ON_TIMEBASE_UP_CMPA);
EPWM_setActionQualifierAction(myEPWM3_BASE, EPWM_AQ_OUTPUT_A, EPWM_AQ_OUTPUT_LOW,
EPWM_AQ_OUTPUT_ON_TIMEBASE_DOWN_CMPA);
EPWM_setActionQualifierAction(myEPWM3_BASE, EPWM_AQ_OUTPUT_B, EPWM_AQ_OUTPUT_LOW,
EPWM_AQ_OUTPUT_ON_TIMEBASE_UP_CMPB);
EPWM_setActionQualifierAction(myEPWM3_BASE, EPWM_AQ_OUTPUT_B, EPWM_AQ_OUTPUT_HIGH,
EPWM_AQ_OUTPUT_ON_TIMEBASE_DOWN_CMPB);

// EPWM1, EPWM2, EPWM3 are linked so this will update both
EPWM_setGlobalLoadOneShotLatch(myEPWM1_BASE);

```


重新进行计算，新的占空比应如方程式 24 中所示：

$$\text{Duty Cycle} = \frac{\text{CMPA} + (\text{CMPA} - \text{DBRED})}{\text{TBPRD} * 2} = \frac{(100 + 80)}{250} = \frac{180}{250} = 72 \% \quad (24)$$

按时间计算， $180 * 10\text{ns}$ 等于 $1.8\mu\text{s}$ ，因此正脉冲宽度应为 $1.8\mu\text{s}$ ，如节 9.3 所示。

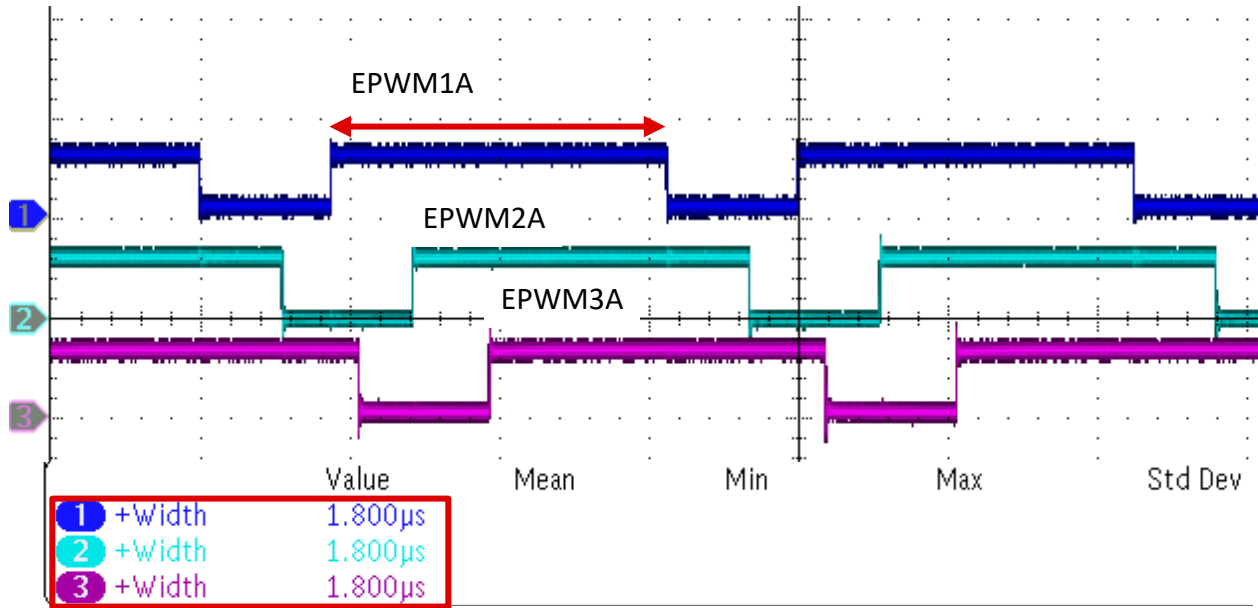


图 9-2. AQ 和 CC 更新后 EPWM 输出正占空比的示波器捕获

10 总结

本应用手册重点介绍了 ePWM 4 类模块的所有主要功能，以及如何在应用中利用这些功能。此外，还介绍了所有必要的计算、如何使用 SysConfig 配置模块以及 SysConfig 生成的代码。此实现的软件解决方案可在最新的 C2000Ware SDK 中的 epwm driverlib 示例 (epwm_ex_global_load_use_case) 中找到。有关寄存器或 ePWM 模块的更多具体指导，请参阅特定器件的 TRM。

11 参考文献

- EPWM 视频系列
- C2000 Academy 的 EPWM 部分
- 德州仪器 (TI)：利用新型 ePWM 特性进行多相控制
- 德州仪器 (TI)：CRM/ZVS PFC 实施基于 C2000™ 4 类 PWM 模块
- 德州仪器 (TI)：在 C2000 4 类 PWM 上实现三相交错型 LLC
- SysConfig 视频系列
- 德州仪器 (TI)：C2000 实时控制 MCU 外设指南

12 修订历史记录

注：以前版本的页码可能与当前版本的页码不同

Changes from Revision * (July 2022) to Revision A (February 2023)	Page
• 更新了整个文档中的表、图和交叉参考的编号格式.....	3
• 更新了 节 10。.....	25

重要声明和免责声明

TI“按原样”提供技术和可靠性数据（包括数据表）、设计资源（包括参考设计）、应用或其他设计建议、网络工具、安全信息和其他资源，不保证没有瑕疵且不做任何明示或暗示的担保，包括但不限于对适销性、某特定用途方面的适用性或不侵犯任何第三方知识产权的暗示担保。

这些资源可供使用 TI 产品进行设计的熟练开发人员使用。您将自行承担以下全部责任：(1) 针对您的应用选择合适的 TI 产品，(2) 设计、验证并测试您的应用，(3) 确保您的应用满足相应标准以及任何其他功能安全、信息安全、监管或其他要求。

这些资源如有变更，恕不另行通知。TI 授权您仅可将这些资源用于研发本资源所述的 TI 产品的应用。严禁对这些资源进行其他复制或展示。您无权使用任何其他 TI 知识产权或任何第三方知识产权。您应全额赔偿因在这些资源的使用中对 TI 及其代表造成的任何索赔、损害、成本、损失和债务，TI 对此概不负责。

TI 提供的产品受 [TI 的销售条款](#) 或 [ti.com](#) 上其他适用条款/TI 产品随附的其他适用条款的约束。TI 提供这些资源并不会扩展或以其他方式更改 TI 针对 TI 产品发布的适用的担保或担保免责声明。

TI 反对并拒绝您可能提出的任何其他或不同的条款。

邮寄地址：Texas Instruments, Post Office Box 655303, Dallas, Texas 75265

Copyright © 2023，德州仪器 (TI) 公司