*Application Note*

# *MSPM0 - Advanced Control Timer Helps for Better Control and Better Digital Output*

*Yuhao Zhao*

**ABSTRACT**

The timer module (TIMx) is a timer counting module with multiple compare/capture blocks. Based on the device, two types of timers are available: general-purpose timers (TIMG) and advanced control timers (TIMA). Both timers use a common timer architecture, which allows for easy migration between timer instances with common functions. This minimizes the need to write extra software for timer-based applications and allows for easy porting and maintenance between TIMx instances.

The TIMA module consists of a 16-bit auto reload counter driven by a programmable prescaler with up to four capture/compare (CC) blocks for multiple capture/compares, PWM outputs with deadband insertion, and interval timing. TIMA has extensive event generation capabilities from different counter events such as overflow, reload, and from each of the capture/compare registers. It also has the hardware design to handle the fault signal generated by internal or external circuitry to indicate a fault in the system.

Based on TIMA, you can configure richer timer module behaviors to design and implement more complex applications, such as motor control, LED modules, power modules, etc. Advanced control timers help you achieve better control and better digital outputs.

## Table of Contents

## Trademarks

All trademarks are the property of their respective owners.

# 1 Introduction

This application note describes how to configure TIMA to achieve unique application designs and introduces how to use SysConfig to achieve quick configuration. If you have questions about the specific configuration of TIMA (such as registers), please refer to the TRM of the specific device.

In this application note:

- TIMx needs to be understood as a common feature available on TIMG and TIMA
- TIMA needs to be understood as a feature available only on TIMA
- TIMG needs to be understood as a feature available only on TIMG

# 2 Comparison Between TIMA and TIMG

Specific features for each TIMA instance include:

- 16-bit up, down, or up-down counter, with repeat-reload mode
- Selectable and configurable clock source
- 8-bit programmable prescaler to divide the counter clock frequency
- Repeat counter to generate an interrupt or event only after a given number of cycles of the counter
- Up to four independent channels for:
  - Output compare
  - Input capture
  - PWM output (Edge-Aligned and Center-Aligned)
  - One-shot mode
- Two additional capture/compare channels for internal events (CC4/CC5)
- Shadow register for load and compare values
- Complementary PWM output with programmable deadband insertion
- Asymmetric PWM output

The TIMG module consists of 16-bit and 32-bit auto reload counters driven by a programmable prescaler with two capture/compare (CC) blocks for multiple capture/compares, PWM outputs, and interval timing. TIMG also has extensive event generation capabilities, including counter overflow, reload, and capture/compare actions for a variety of use cases.

Specific features for TIMG include:

- 16-bit up, down, or up-down counter, with repeat-reload mode
- 8-bit programmable prescaler to divide the counter clock frequency
- Up to two independent channels for
  - Output compare
  - Input capture
  - PWM output (Edge-Aligned and Center-Aligned)
  - One-shot mode
- Shadow register mode for load and compare values
- Support for quadrature encoder interface (QEI)
- 3-input Hall sensor mode for position sensing and speed computation
- Support synchronization and cross trigger among different TIMx instances in the same power domain
- Support CPU interrupt generation and cross peripherals (such as ADC, DAC, and so on) using the Event

Table 2-1 shows the configuration for TIMA and TIMG instances.

### Table 2-1. TIMx Instance Configuration

| Instance | Power Domain | Counter Resolutio n | Prescaler | Repeat Counter | CCP Channels (External / Internal) | External PWM Channels | Phase Load | Shadow Load | Shadow CCs | Deadband | Fault Handler | QEI/Hall Input Mode |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TIMG0 | PD0 | 16-bit | 8-bit | - | 2 | 2 | - | - | - | - | - | - |
| TIMG1 | PD0 | 16-bit | 8-bit | - | 2 | 2 | - | - | - | - | - | - |
| TIMG2 | PD0 | 16-bit | 8-bit | - | 2 | 2 | - | - | - | - | - | - |
| TIMG3 | PD0 | 16-bit | 8-bit | - | 2 | 2 | - | - | - | - | - | - |
| TIMG4 | PD0 | 16-bit | 8-bit | - | 2 | 2 | - | Yes | Yes | - | - | - |
| TIMG5 | PD0 | 16-bit | 8-bit | - | 2 | 2 | - | Yes | Yes | - | - | - |
| TIMG6 | PD1 | 16-bit | 8-bit | - | 2 | 2 | - | Yes | Yes | - | - | - |
| TIMG7 | PD1 | 16-bit | 8-bit | - | 2 | 2 | - | Yes | Yes | - | - | - |
| TIMG8 | PD0 | 16-bit | 8-bit | - | 2 | 2 | - | - | - | - | - | Yes |
| TIMG9 | PD0 | 16-bit | 8-bit | - | 2 | 2 | - | - | - | - | - | Yes |
| TIMG10 | PD1 | 16-bit | 8-bit | - | 2 | 2 | - | - | - | - | - | Yes |
| TIMG11 | PD1 | 16-bit | 8-bit | - | 2 | 2 | - | - | - | - | - | Yes |
| TIMG12 | PD1 | 32-bit | - | - | 2 | 2 | - | - | Yes | - | - | - |
| TIMG13 | PD0 | 32-bit | - | - | 2 | 2 | - | - | Yes | - | - | - |
| TIMG14 | PD1 | 16-bit | 8-bit | - | 4 | 4 | - | - | - | - | - | - |
| TIMA0 | PD1 | 16-bit | 8-bit | Yes | 4/2 | 8 | Yes | Yes | Yes | Yes | Yes | - |
| TIMA1 | PD1 | 16-bit | 8-bit | Yes | 2/2 | 4 | Yes | Yes | Yes | Yes | Yes | - |

# 3 Use Case - 3 Pairs of Complementary PWM with Deadband Insertion

## 3.1 Principle

Complimentary outputs with deadband insertion are a common use case for inverter-based applications with half-bridge topologies. Deadband is useful for applications with half-bridge control to avoid shoot-through conditions, such as motor driver or inverter-based applications.

To generate 3 pairs of complementary PWM with deadband insertion, TIMA0 is required for 3 CCP channels. The following is the key feature for the use case.

- Generate PWM with 3 CCP channels (TIMA0 only)
- Enable complimentary CCP output channels (TIMA only)
- Configure deadband insertion (TIMA only)

## 3.2 Implement

TIMx can be used to generate desired pulse-width modulation (PWM) output waveforms with the counter and capture/compare modules (compare match). The waveform generation of a CCP output depends on the counting mode and counter compare actions

For example, with different counting mode, edge-aligned PWM or center-aligned PWM can be generated. A typical 2-channel PWM is shown in Figure 3-1. The user can set LOAD value to configure the PWM period, or set CC value to configure the duty cycle. The user can also configure polarity of the signal, CCP output action and so on. Please refer to the TRM for specific configuration.

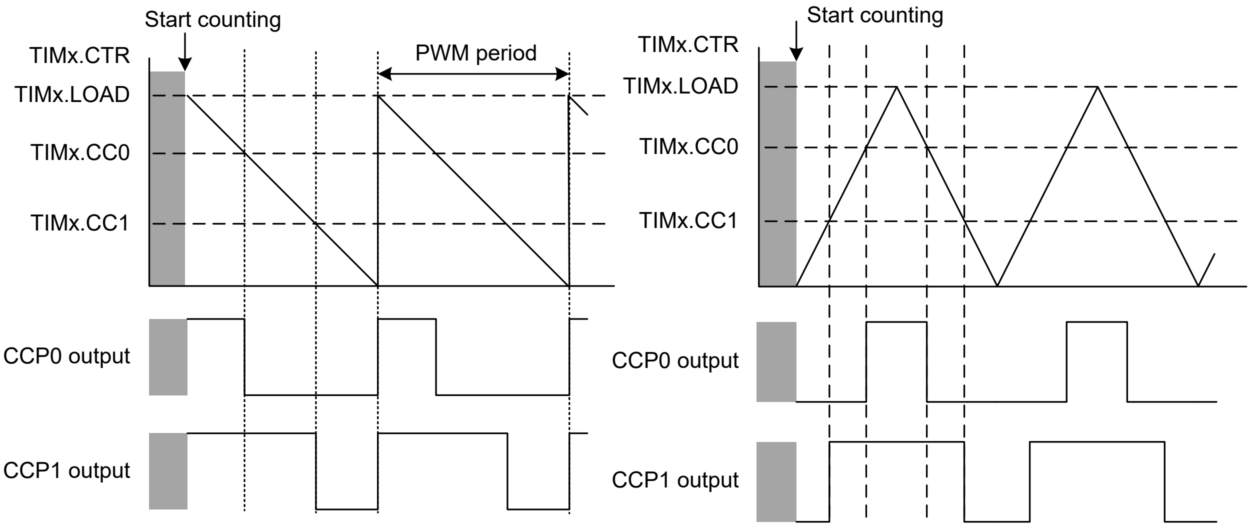

**Figure 3-1. Edge-Aligned PWM and center-aligned PWM**

Both TIMA and TIMG can generate PWM. However, TIMA provides the optional feature on complimentary CCP output channels, such as TIMA0_C1 and TIMA0_C1N for a reference PWM signal on TIMA0 CCP output channel 1. Besides, TIMA0 has 4 CCP channels while TIMA1 and TIMG have 2 CCP channels. It means that TIMA0 can output 8 PWMs (4 pairs) and TIMA1 can output 4 PWMs (2 pairs), while TIMG can only output 2 PWMs.

- TIMA0:
  - TIMA0_C0, TIMA0_C0N
  - TIMA0_C1, TIMA0_C1N
  - TIMA0_C2, TIMA0_C2N
  - TIMA0_C3, TIMA0_C3N
- TIMA1:
  - TIMA1_C0, TIMA1_C0N

  - TIMA1_C1, TIMA1_C1N
- TIMGx:
  - TIMGx_C0
  - TIMGx_C1

TIMA also provides the option on deadband insertion(non-overlapping transitions in complimentary PWM signals). Multiple modes of configurable deadband can be inserted in edge-aligned or center-aligned complimentary PWMs, as shown in Figure 3-2 and Figure 3-3.

The user can configure deadband mode and timing information with the deadband control register (TIMA.DBCTL) programmed. With mode 0, rise delay and fall delay are applied with respect to the output generator signal's rising and falling edges to generate CCP and CCP complimentary signal. With mode 1, rise delay and fall delay are both applied to CCP complimentary signal only. Figure 3-2 shows the edge-aligned PWM with deadband (mode 0 only). Figure 3-3 shows the center-aligned PWM with deadband (mode 0 and mode 1).

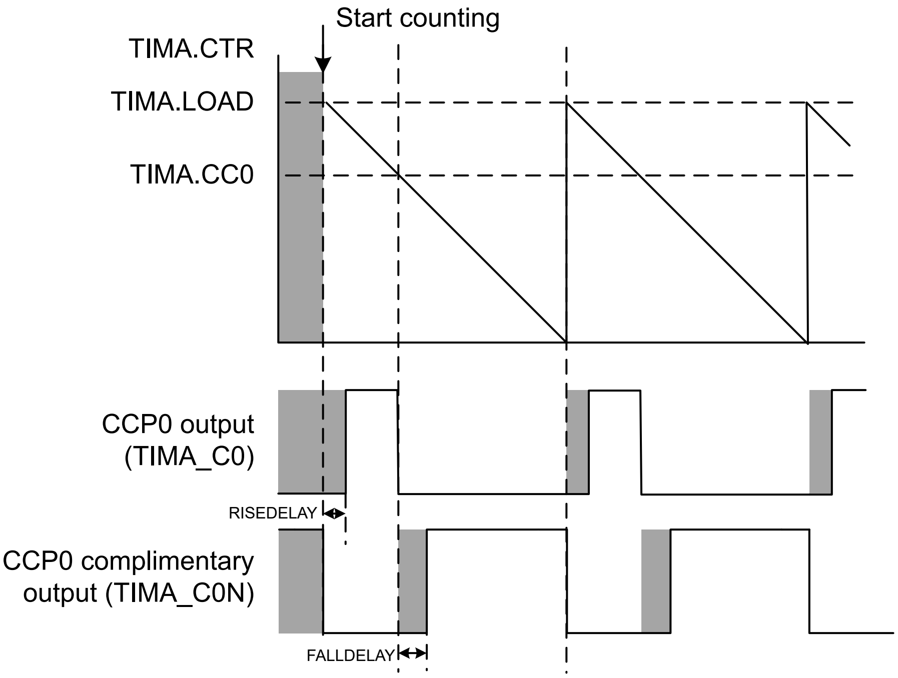

**Figure 3-2. Edge-Aligned PWM with Deadband**

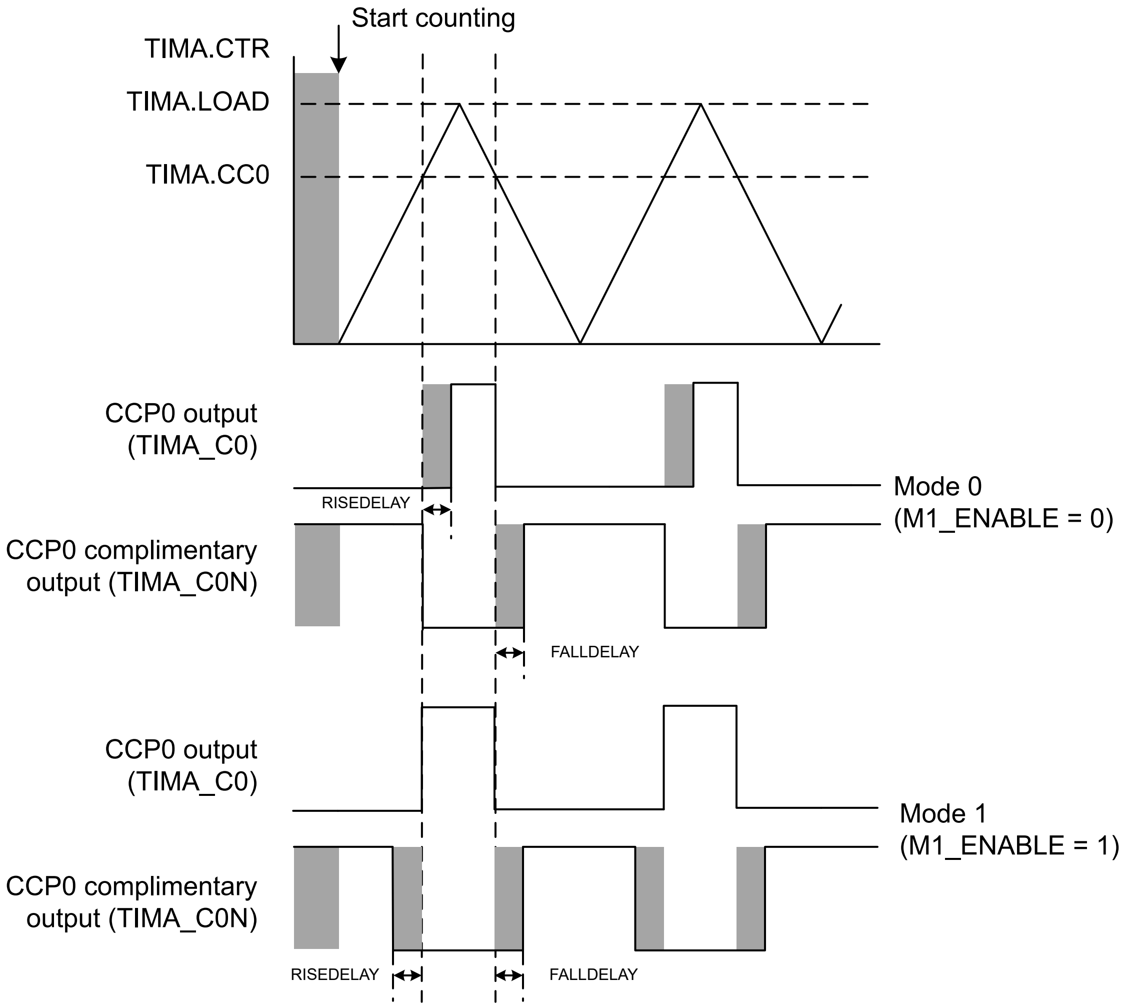

**Figure 3-3. Center-Aligned PWM with Deadband**

To accelerate development, please refer to the following resources.

- Examples from SDK: tima_timer_mode_pwm_dead_band

# 4 Use Case - Timing-Critical PWM Control with Shadow Load and Compare

## 4.1 Principle

For MSPM0, some timer modules have a shadow load and shadow compare register feature which gives the user the flexibility of holding the update of load and CC values until a certain event occurs. This is useful in timing-critical applications where PWM control signals need to be updated with correct timings, such as duty cycle updates.

To control PWM with shadow load / compare, TIMA or specific TIMG are required. The following is the key feature for the use case.

- Generate center-aligned / edge-aligned PWMs
- Enable shadow load
- Configure CC update method

## 4.2 Implement

The shadow load feature allows holding the update of load values until a zero event occurs. If the TIMx module has a shadow load feature, there is an internal shadow register for the load value (TIMx.LOAD). The shadow register can update the load value at a zero event.

For up-counting mode and up or down counting mode, note that a shadow load is necessary. A shadow load can make sure that TIMx counts up to the load value before the zero event, or else the load value can update immediately and cause incorrect timings.

When shadow compare is enabled for updating the capture or compare register (TIMx.CC), the value written to the respective compare register is first stored in a shadow compare register and then transferred to the compare register at different events. User can also configure to update the CC action at different events.

Figure 4-1 shows an example of how shadow load and shadow compare takes effect at the zero event for both the TIMx.LOAD and TIMx.CC value in up/down counting mode.

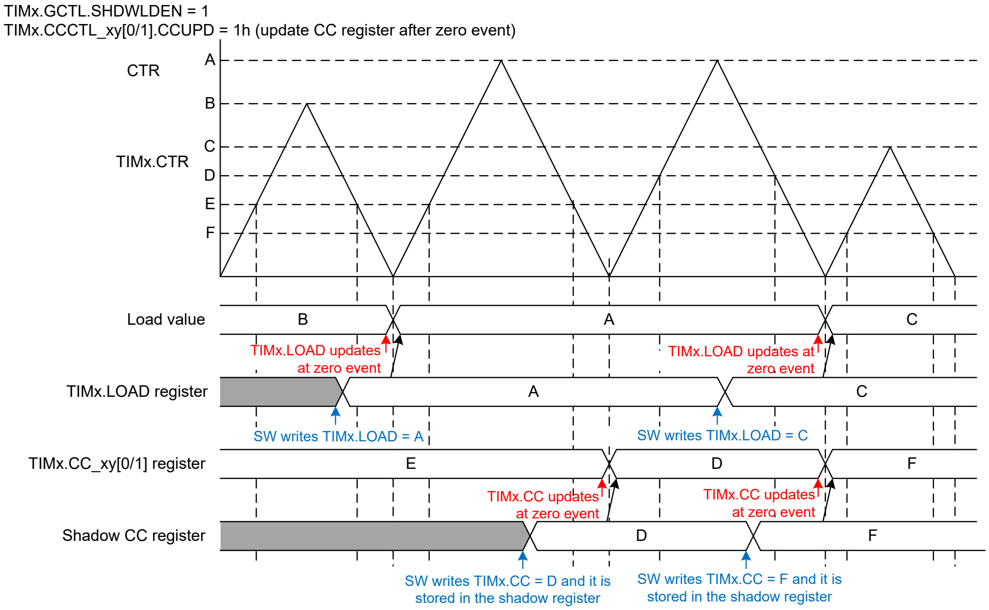
TIMx.GCTL.SHDWLDEN = 1
TIMx.CCCTL_xy[0/1].CCUPD = 1h (update CC register after zero event)
CTR
A
B
TIMx.CTR
C
D
E
F
Load value
B
A
C
TIMx.LOAD updates at zero event
TIMx.LOAD updates at zero event
TIMx.LOAD register
A
C
SW writes TIMx.LOAD = A
SW writes TIMx.LOAD = C
TIMx.CC_xy[0/1] register
E
D
F
TIMx.CC updates at zero event
TIMx.CC updates at zero event
Shadow CC register
D
F
SW writes TIMx.CC = D and it is stored in the shadow register
SW writes TIMx.CC = F and it is stored in the shadow register

**Figure 4-1. Shadow Load and Shadow Compare Taking Effect at Zero Event in Up or Down Mode**

To accelerate development, please refer to the following resources.

- Examples from SDK: timx_timer_mode_pwm_edge_sleep_shadow

# 5 Use Case - Fault Handler

## 5.1 Principle

Fault handler is usually required in real-time control to handle abnormal conditions, for example overcurrent or overvoltage conditions. Hardware fault behavior enables faster protection than software code. In addition, more complex PWM control can be implemented based on the fault handler.

In TIMA only, there are internal and external fault inputs which can be used to control the generation of PWM signals. The intended use of these inputs is as a mechanism for internal or external circuitry to indicate a fault in the system. This allows the hardware to react quickly to the external fault while optionally signaling an interrupt for software correction and leaving the output signals in a safe state.

To enable fault handler, TIMA is required. The following is the key feature for the use case.

- Generate center-aligned or edge-aligned PWMs
- Enable fault handler (TIMA only)

## 5.2 Implement

Before configuring fault handler, it is important to consider the following basic properties of faults in a system, such as:

- Fault input selection (fault signal from external IC, internal signal, and so on)
- How long a fault condition lasts, or the fault condition duration
- How the counter reacts to the entry and exit of a fault condition
- How the output signal reacts to the entry and exit of a fault condition

For fault input source, the user can select the polarity and enable various fault input sources, such as comparator (COMP) output, external fault pin, system clock fault or a trigger. The user can also controls the input filtering for the fault input.

- The comparator output is useful for fault detection when COMPs are used for detecting overcurrent or overvoltage events.
- There are 3 fault external signal pins can be used to detect the fault condition outside the MCU.
- Any system clock fault can be used to trigger the PWM output to a Hi-Z state.
- A trigger can be configured to generate a fault condition is detected. This is useful for performing diagnostics or creating fault dependencies from other peripherals in the event fabric.

Fault handler can have effects on counter and output. For counter behavior with fault conditions, counter can be set to stop immediately, stop when reaches zero or continue to count. Upon exit of fault mode, the counter can be set to restart from Load value, 0 value, or continue counting from where it left off. Figure 5-1shows the counter behavior in fault condition with different configurations.

For output behavior with fault conditions, the CCP output value can be set high, set low, toggled, tristated (Hi-Z), or unaffected by the fault event.

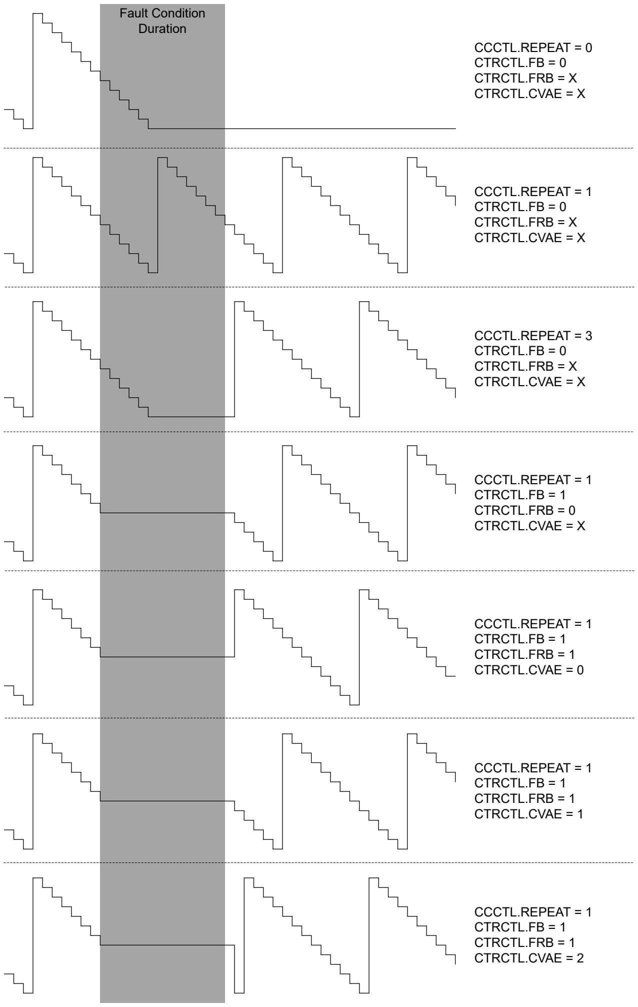

**Figure 5-1. Counter Behavior in Fault Condition**

To accelerate development, please refer to the following resources.

- Examples from SDK: tima_trigger_fail_mechanism

# 6 Use Case – PWM Disable with Software Force Output

## 6.1 Principle

In addition to the fault handler, users can also force the PWM output in software. Each output channel signal can be forced to a high or low level directly by software, independently of any comparison between the compare register and the counter.

The following is the key feature for the use case. All timer modules support the feature.

- Generate center-aligned or edge-aligned PWMs
- Set the SWFRCACT bit in the TIMx.CCACT_xy[0/1] register

## 6.2 Implement

The output of the CCP channel can be forced to high or low. Additionally, in TIMA only, the complimentary output channel can also be forced to high or low. Table 6-1 shows the software force output action configuration options.

**Table 6-1. Force Output Action Configuration**

| Bit Field | Value | Description or Comment |
|---|---|---|
| SWFRCACT / SWFRCACT_CMPL | 0 | No forced output. Output is directly from the signal generation block. |
| | 1 | Force output high |
| | 2 | Force output low |

The user can set the SWFRCACT bit in register to force or release the PWM output. Besides, the user can call the API to produce the same effect as shown in the following code.

```
/**
 *  @brief Overrides the timer CCP output
 *
 *  @param[in]  gptimer       Pointer to the register overlay for the
 *                            peripheral
 *  @param[in]  out           Specifies the CCP output state.
 *                            @ref DL_TIMER_FORCE_OUT
 *  @param[in]  outComp       If timer insatance supports complementary output,
 *                            it allows to override complementary out also.
 *                            If timer instance doesn't support complementary
 *                            output, this parameter is ignored.
 *                            @ref DL_TIMER_FORCE_CMPL_OUT
 *  @param[in]  ccIndex       Index associated to capture compare register
 *                            @ref DL_TIMER_CC_INDEX.
 */
void DL_Timer_overrideCCPOut(GPTIMER_Regs *gptimer, DL_TIMER_FORCE_OUT out,
    DL_TIMER_FORCE_CMPL_OUT outComp, DL_TIMER_CC_INDEX ccIndex);
```

# 7 Use Case - Asymmetric PWM

## 7.1 Principle

Phase shift control is widely used in applications such as motor control, power supplies, and so on. In TIMA only, asymmetric PWMs can be generated by generating two synchronized center-aligned PWM signals with a controlled phase shift.

To generate Asymmetric PWM, TIMA is required for phase load. Besides, cross trigger is used to synchronize timers. The following is the key feature for the use case.

- Generate two center-aligned PWMs
- Use a cross trigger to synchronize two TIMx.
- Configure phase load (TIMA only)

## 7.2 Implement

Cross trigger can be used to synchronize different timers. When using a main-secondary timer configuration by connecting multiple timers together, the cross-trigger feature can instruct multiple timer modules in the same power domain or across different power domains using the event fabric to start counting simultaneously.

Cross-triggers can be enabled using software, compare events from other timer instances, zero or load events, or generic subscriber events. Some applications can require more than one counter block that can be simultaneously started across the same power domain (such as TIMA0 and TIMA1) or different power domains (such as TIMA0 and TIMG0). As shown in Figure 7-1, TIMGx is the main timer and TIMAx is the secondary timer that can be cross triggered in the configuration example.

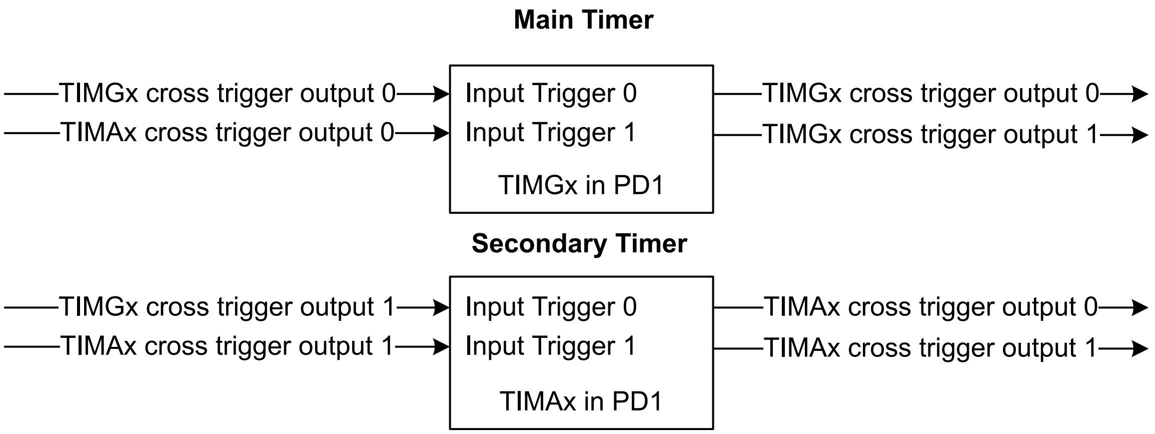
Main Timer
TIMGx cross trigger output 0
TIMAx cross trigger output 0
Input Trigger 0
Input Trigger 1
TIMGx in PD1
TIMGx cross trigger output 0
TIMGx cross trigger output 1
Secondary Timer
TIMGx cross trigger output 1
TIMAx cross trigger output 1
Input Trigger 0
Input Trigger 1
TIMAx in PD1
TIMAx cross trigger output 0
TIMAx cross trigger output 1

**Figure 7-1. Cross Trigger Connections for Main Timer (TIMGx) and Secondary Timer (TIMAx) in Power Domain 1**

By using cross trigger, different timers can start counting simultaneously. To produce a controlled phase shift, phase load is required. In TIMA only, the phase load register provides the capability for counter to count from a value other than zero or load value in Up/Down counting mode. Phase load is used to generate asymmetric center-aligned PWM output signals with a controlled phase shift between different timer instances.

When phase load is triggered, the timer counts from the phase load value. Phase load value is latched when the timer starts. The phase load is synchronized every time when the counter reaches the previously latched phase load value. Figure 7-2 shows how the phase load register works when the timer is counting in the up-down direction and the phase load value changes to a new value.

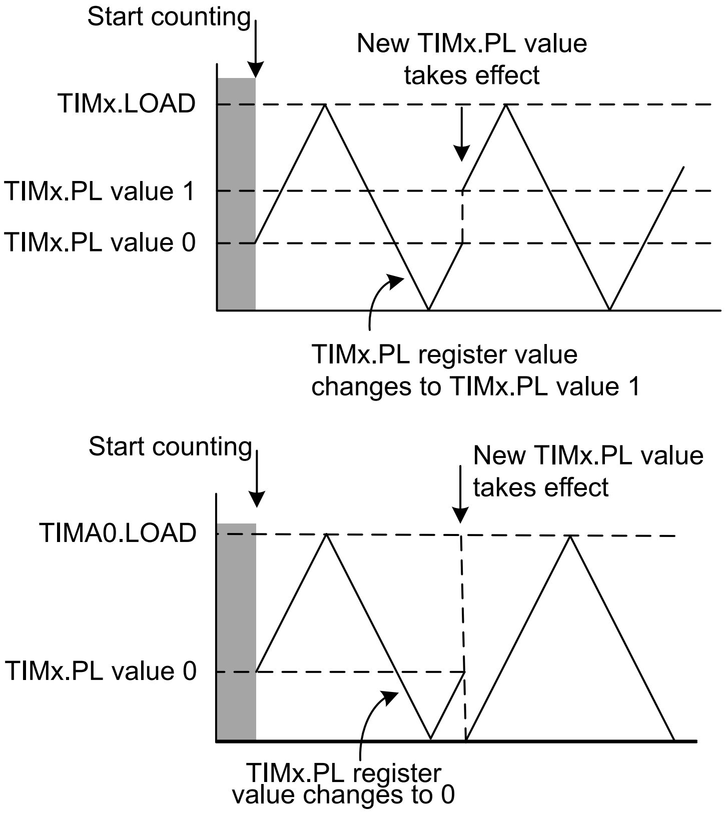

**Figure 7-2. Phase Load Register Synchronization in Up-Down Mode**

As an example of asymmetric PWM configuration using CCP channel 0 of TIMA0 and TIMA1, firstly synchronize TIMA0 and TIMA1 using a cross trigger. Configure two center-aligned using TIMA0 and TIMA1 with the same load value and compare value to generate the same PWM frequency and duty cycle. Add a phase shift value for TIMA0 or TIMA1 by configuring the phase load value. The Asymmetric PWM is shown in Figure 7-3.

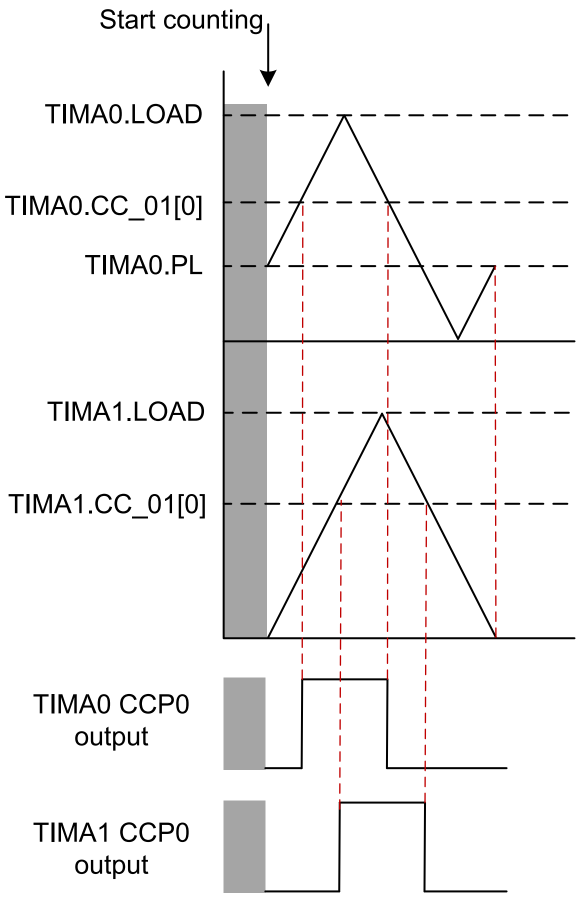

**Figure 7-3. Asymmetric PWM Configuration with Phase Load for CCP Channel 0 of TIMA0 and TIMA1**

To accelerate development, please refer to the following resources.

- Examples from SDK: timx_timer_mode_pwm_cross_trigger_stop_restore

# 8 Use Case – Optimal Interrupt Generation with Repeat Counter

## 8.1 Principle

In TIMA only, the repeat counter is an 8-bit counter that provides the mechanism to suppress unnecessary events and generate real events for optimal interrupt generation. Specifically, the repeat counter can suppress Load, Compare, and Zero events in the case where the timer is generating events that repeat for a known number of cycles, such a periodic PWM output waveform. This prevents generating excessive and unnecessary interrupts every timer period.

To generate optimal Interrupt with repeat counter, TIMA is required. The following is the key feature for the use case.

- Generate center-aligned or edge-aligned PWMs
- Configure repeat counter(TIMA only)

## 8.2 Implement

When the timer counter is advancing, the repeat counter advances once the counter reloads. The user can set the how many timer counter reloads occur until generating the interrupts and events. Once repeat counter equals the setting number, the repeat counter is reset back to zero and a Repeat Counter Zero event occurs (REPC) in the Interrupt and Event Status registers.

Additionally, the repeat counter provides the ability to suppress generation of Zero, Load, and Compare events when TIMA.RC does not equal zero. Based on this, MCU can avoid generating excessive and unnecessary interrupts.

As shown in Figure 8-1, the timer counter is configured for down-counting mode and zero events are generated once timer counter = 0. Repeat counter is used to suppress zero and load events until 4 timer reloads occur. The user can configure repeat counter to generate Optimal Interrupt as needed.

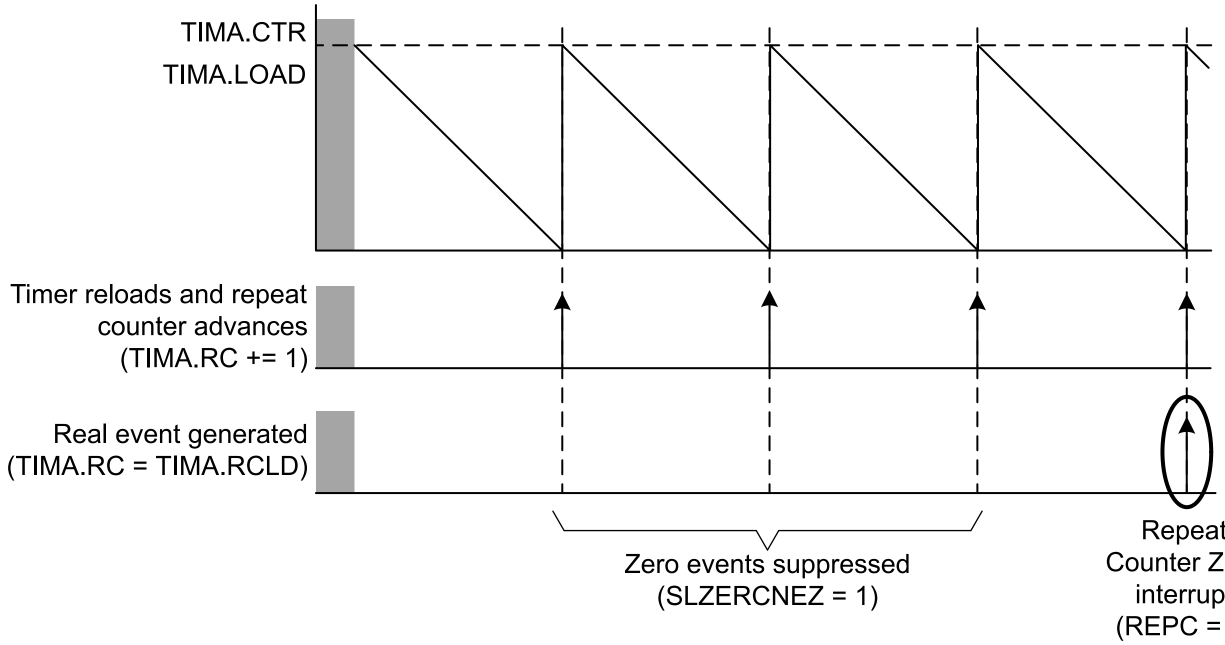
TIMA.CTR
TIMA.LOAD
Timer reloads and repeat counter advances (TIMA.RC += 1)
Real event generated (TIMA.RC = TIMA.RCLD)
Zero events suppressed (SLZERCNEZ = 1)
Repeat Counter Zero interrupt (REPC = 1)

**Figure 8-1. Event Suppressed by Repeat Counter**

To accelerate development, please refer to the following resources.

- Examples from SDK: tima_timer_mode_periodic_repeat_count

## 9 Summary

This application note mainly introduces some typical application scenarios of advanced control timer. Based on advanced control timer, users can implement richer configurations to adapt to more diverse usage scenarios. Users can combine many of the unique features of a single timer or use multiple timers together to achieve better control and better digital output.

For more detailed information, please refer to the Technical Reference Manual for the specific device.

## 10 References

- Texas Instruments, *Download the MSPM0 SDK*.
- Texas Instruments, *System Configuration Tool*, product page.
- Texas Instruments, *MSPM0 C-Series 24-MHz Microcontrollers*, technical reference manual.
- Texas Instruments, *MSPMC1104 LaunchPad™ development kit for 24-MHz Arm® Cortex®-M0+ MCU*, LaunchPad development kit.
- Texas Instruments, *MSPM0 Timer academy*.

# IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATA SHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, regulatory or other requirements.

These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products.

TI objects to and rejects any additional or different terms you may have proposed.

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