*Application Note*

# *Achieve Delayed Protection for Three-Level Inverter With Type 4 EPWM*

*Aki Li and Ricky Zhang*

## ABSTRACT

The three-level inverter is commonly used in high-power applications, while a special protection control scheme is required, and many users try to implement with costly circuits externally. Since *Achieve Delayed Protection for Three-Level Inverter With CLB* could only apply to the C2000 devices with configurable logic block (CLB), this application report discusses how to design the EPWM configurations to achieve customize delayed protection with Type 4 EPWM, for most of the C2000 devices.

## Table of Contents

## Trademarks

All trademarks are the property of their respective owners.

# 1 Introduction

Figure 1-1 shows a typical single phase three-level I-Type inverter, named neutral point clamped (NPC) inverter. The single phase NPC inverter includes 4 FETs, like IGBT, in series, where S1 and S4 are called outer switches, with S2 and S3 called inner switches.

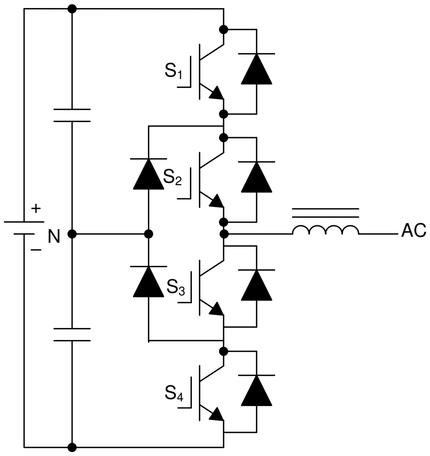


**Figure 1-1. Single Phase Three-Level I-Type Inverter**

Considering the difference between positive cycle and negative cycle when tied to the grid, the general switching states of four FETs in normal operation are shown in Table 1-1.

**Table 1-1. General Switching States in Normal Operation**

| | **Switching states** | | | |
|---|---|---|---|---|
| **Symbol** | **S1** | **S2** | **S3** | **S4** |
| Positive | Alternate switch | Remaining ON | Alternate switch | Remaining OFF |
| Negative | Remaining OFF | Alternate switch | Remaining ON | Alternate switch |

There are several events which lead to quick shut-down to protect the semiconductors and the system, like over current, thermal overload, etc. Unlike immediately switching off all the FETs simultaneously in two level inverter, for three-level inverter, it must be made sure that the correct switch-off sequence is maintained: outer switches (S1 or S4) off first, inner switches (S2 or S3) off after a specific delay, while the inner one must be switched on firstly during the recover process. This delayed protection requirement has been a challenge for lots of UPS or solar inverter customers for a long time. Since using software algorithms will cause too much delay to provide in-time protection, some customers have to use external hardware circuits, like FPGA or CPLD, to achieve such protection logic, which increases the system cost and also the development effort.

This application report demonstrates how to implement delayed protection with existed Type 4 ePWM features that are available in all new C2000 devices, including F2838xD/S, F2837xD/S, F2807x and F28004x. In the Type 4 EPWM module, the Action Qualifier sub-module includes two additional trigger events (T1 and T2), which can be sourced from comparator, trip or sync events, together with Dead-Band sub-module, make it possible to add customized delay time to tripped/fault signals.

# 2 Implementation Guidance

## 2.1 Expected Results Overview

Figure 2-1 shows the expected EPWM protection results during positive and negative cycle operation conditions. In the given example, EPWM1A/2B/1B/2A are used for S1/2/3/4 control, respectively. And an extra EPWM output, EPWM7A in the example, is required to support the delayed protection logic. The trip signal is simulated and generated with an EPWM output, EPWM8A here, for the quick validation purpose.

Take the positive cycle operation as example, whenever the trip event occurs, EPWM1A and EPWM1B, which are configured with cycle-by-cycle tripping (CBC) action, are shut down immediately, while EPWM2B will be forced down after a Trip delay. Besides, when trip event disappears, EPWM2B will turn high immediately, which is also aligned with the basic recover requirement.

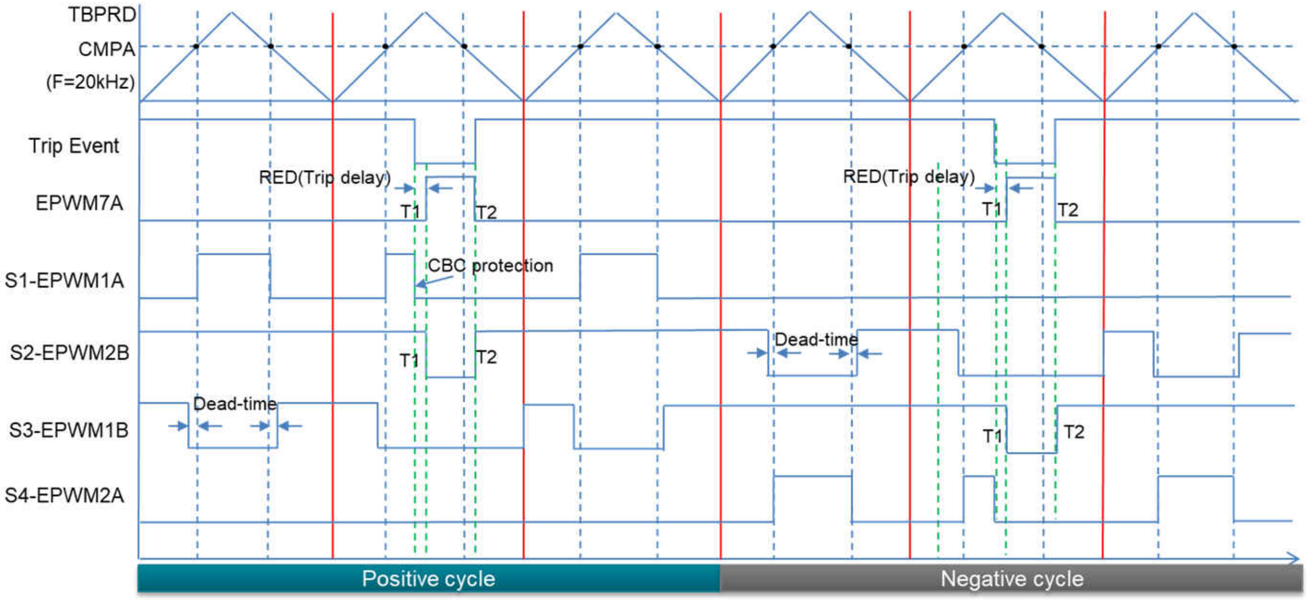


**Figure 2-1. Expected EPWM Protection Results During Positive and Negative Cycle Operation Conditions**

Note that the given example code is based on F28004x, and it can be easily migrated to any other C2000 devices with Type 4 EPWM. And the code, built with the Driverlib API format, makes it more flexible to change or add the selected EPWM modules for different FET controls. Though the example is designed for single phase three-level inverters, the same configuration method can be used for three phase inverter topology, with six EPWM modules for main FETs control and one auxiliary EPWM channel needed. The example project discussed in this application report is available in the latest C2000WARE-DIGITALPOWER-SDK, navigating to the directory C:\ti\c2000\C2000Ware_DigitalPower_SDK_X_XX_XX_XX\libraries\3_level_inv_delayed_protection_scheme\examples.

## 2.2 System and Auxiliary Trip Source Signal Configuration

In the given example code, it provides a showcase to use a GPIO signal, to simulate the external comparator output state, as the system trip source signal. In F28004x, the on-chip signal crossbars (X-bars), including Input X-BAR and the EPWM X-BAR mechanism, make it possible to route any GPIO signal to work as the any of the eight dedicated EPWM TRIPx signals.

The following codes, found in InitINPUTXBAR() and InitEPWMXBAR(), show that GPIO14, referring to EPWM8A, is set as the source for XBAR_INPUT_system (INPUT X-BAR 1), and then fed to

XBAR_TRIP_system (EPWM X-BAR TRIP4). For more details, see the *EPWM X-BAR Mux Configuration* table in the *TMS320F28004x Piccolo Microcontrollers Technical Reference Manual*.

```
    // INPUT X-BAR 1□ GPIO14 □ EPWM8A
    XBAR_setInputPin(XBAR_INPUT_system, simulate_PWM_GPIO);
    XBAR_setEPWMMuxConfig(XBAR_TRIP_system, XBAR_EPWM_MUX01_INPUTXBAR1);
    XBAR_enableEPWMMux(XBAR_TRIP_system, XBAR_MUX01);
```

Note that the trip source can also be flexibly selected from the internal comparator output. For example, using the output CTRIPH of CMPSS1 as the X-BAR_TRIP_system is shown as below:

```
XBAR_setEPWMMuxConfig(XBAR_INPUT_system, XBAR_EPWM_MUX00_CMPSS1_CTRIPH);
XBAR_enableEPWMMux(XBAR_TRIP4, XBAR_MUX00);
```

Besides, in order to implement the protection logic, an extra EPWM channel (EPWM7A used in the example) is also required to set as the auxiliary trip source signal, where EPWM7A is configured as XBAR_TRIP_auxiliary (EPWM X-BAR TRIP7).

```
    // INPUT X-BAR 3->GPIO12->EPWM7A
    XBAR_setInputPin(XBAR_INPUT_auxiliary, AUX_PWM_GPIO);
    XBAR_setEPWMMuxConfig(XBAR_TRIP_auxiliary, XBAR_EPWM_MUX05_INPUTXBAR3);
    XBAR_enableEPWMMux(XBAR_TRIP_auxiliary, XBAR_MUX05);
```

## 2.3 CBC Protection Configuration

Take a positive cycle operation as shown in the example, the trip signal (active low) is required to set as CBC trip event, which forces EPWM1A and EPWM1B low and applies to the EPWM2 during negative cycle.

Firstly, the trip signal (EPWM8A, GPIO14) is configured as Digital Compare A High (DCAH) in both the EPWM1 and EPWM2 modules, and the DCAEVT2 event takes effect when DCAH becomes low.

```
  // DCAH = TRIPIN4 = INPUT X-BAR 1 = EPWM8A
  EPWM_selectDigitalCompareTripInput(base1, DC_TRIP_system, EPWM_DC_TYPE_DCAH);
  // DCAH = Low and DCAL = Don't care, Trigger DCAEVT2 when EPWM8A goes low
  EPWM_setTripZoneDigitalCompareEventCondition(base1,EPWM_TZ_DC_OUTPUT_A2,EPWM_TZ_EVENT_DCXH_LOW);
  // DCAH = TRIPIN4 = INPUT X-BAR 1 = EPWM8A
  EPWM_selectDigitalCompareTripInput(base2, DC_TRIP_system, EPWM_DC_TYPE_DCAH);
  // DCAH = Low and DCAL = Don't care, Trigger DCAEVT2 when EPWM8A goes low
  EPWM_setTripZoneDigitalCompareEventCondition(base2,EPWM_TZ_DC_OUTPUT_A2,EPWM_TZ_EVENT_DCXH_LOW);
```

Then, the force to low action is enabled for both EPWM1A and EPWM2A with TZCTL[TZA], while the CBC trip action is ignored for EPWMxB during the initialization. And it will be adjusted according to the positive or negative cycle conditions, see Section 2.6.

```
 EPWM_setTripZoneAction(base1,EPWM_TZ_ACTION_EVENT_TZA,EPWM_TZ_ACTION_LOW);
 EPWM_setTripZoneAction(base1,EPWM_TZ_ACTION_EVENT_TZB, EPWM_TZ_ACTION_DISABLE);
 EPWM_setTripZoneAction(base2,EPWM_TZ_ACTION_EVENT_TZA,EPWM_TZ_ACTION_LOW);
 EPWM_setTripZoneAction(base2,EPWM_TZ_ACTION_EVENT_TZB, EPWM_TZ_ACTION_DISABLE);
```

## 2.4 Auxiliary EPWM Output Configuration

The main challenge is how to insert a customized delay for the tripped action at the specific EPWM output, which is EPWM2B during positive cycle and EPWM1B during negative cycle in this example. As shown in Figure 2-2 with the latest Type 4 EPWM, events T1 and T2, sourcing from comparator, trip or sync events, can also generate actions through Action Qualifier submodule, which mean that dead-band insertion for trip events is possible. Considering the ease of configuration, an extra EPWM7A will be used as auxiliary trip signal for EPWM1B or EPWM2B.

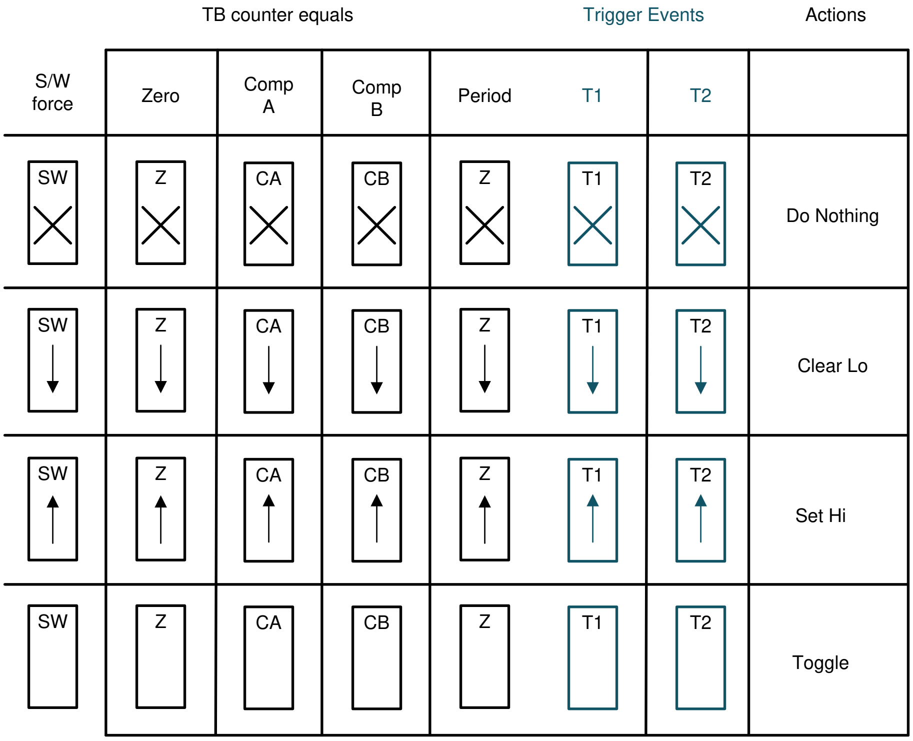


**Figure 2-2. Action-Qualifier Actions for EPWMxA and EPWMxB Outputs**

The detailed configurations can refer to the library function initEPWM_aux_trip(uint32_t aux_pwm_base, EPWM_DigitalCompareTripInput system_trip_source, uint16_t trip_delay). By default, EPWM7A keeps low during normal operation, while it outputs high when the trip signal occurs. The customized delay is defined for the rising edge using the dead-band module with the below settings, and the initial delay is set with trip_delay=1 μs, which could be adjusted in actual applications. Besides, after the trip signal disappears, EPWM7A is expected to turn low immediately, and then wait for the next trip event. The dead-band settings are as below.

```
// S1 = 1
EPWM_setDeadBandDelayMode(aux_pwm_base,EPWM_DB_RED, true);
// S2 = 0
EPWM_setDeadBandDelayPolarity(aux_pwm_base, EPWM_DB_RED, EPWM_DB_POLARITY_ACTIVE_HIGH);
// S4 = 0
EPWM_setRisingEdgeDeadBandDelayInput(aux_pwm_base, EPWM_DB_INPUT_EPWMA);
EPWM_setRisingEdgeDelayCount(aux_pwm_base, trip_delay);
```

Before the action-qualifier settings, it is required to configure the trip signal (active low) and the recover signal (active high) as DCBEVT1 and DCBEVT2, which are set as T1 and T2 events for EPWM7A, respectively. With the system trip source signal defined in Section 2.2, the related codes are as shown below:

```
    // DCBH = TRIPIN4 = INPUT X-BAR 1 = EPWM8A in this example
    EPWM_selectDigitalCompareTripInput(aux_pwm_base, system_trip_source, EPWM_DC_TYPE_DCBH);
     //Trigger DCBEVT1 when system_trip_source signal goes low(fault occurs)
    //Trigger DCBEVT2 when system_trip_source signal goes high(recover from fault)
    EPWM_setTripZoneDigitalCompareEventCondition(aux_pwm_base,EPWM_TZ_DC_OUTPUT_B1,
EPWM_TZ_EVENT_DCXH_LOW);
    EPWM_setTripZoneDigitalCompareEventCondition(aux_pwm_base,EPWM_TZ_DC_OUTPUT_B2,
EPWM_TZ_EVENT_DCXH_HIGH);
    //
    // DCBEVT1/2 event as AQ T1/2
    // T1 = DCBEVT1
    EPWM_setActionQualifierT1TriggerSource(aux_pwm_base, EPWM_AQ_TRIGGER_EVENT_TRIG_DCB_1);
    // T2 = DCBEVT2
    EPWM_setActionQualifierT2TriggerSource(aux_pwm_base, EPWM_AQ_TRIGGER_EVENT_TRIG_DCB_2);
```

Then, the auxiliary EPWM output EPWM7A is controlled to set high at T1 event and clear low at T2 event through the action-qualifier submodule.

```
EPWM_setActionQualifierAction(aux_pwm_base,
                              EPWM_AQ_OUTPUT_A, EPWM_AQ_OUTPUT_HIGH,
                              EPWM_AQ_OUTPUT_ON_T1_COUNT_UP);
EPWM_setActionQualifierAction(aux_pwm_base,
                              EPWM_AQ_OUTPUT_A, EPWM_AQ_OUTPUT_HIGH,
                              EPWM_AQ_OUTPUT_ON_T1_COUNT_DOWN);
EPWM_setActionQualifierAction(aux_pwm_base,
                              EPWM_AQ_OUTPUT_A, EPWM_AQ_OUTPUT_LOW,
                              EPWM_AQ_OUTPUT_ON_T2_COUNT_UP);
EPWM_setActionQualifierAction(aux_pwm_base,
                              EPWM_AQ_OUTPUT_A, EPWM_AQ_OUTPUT_LOW,
                              EPWM_AQ_OUTPUT_ON_T2_COUNT_DOWN);
```

## 2.5 Delayed Protection Configuration

Based on the above configuration, EPWM1B or EPWM2B is able to achieve expected protection logic now. The detailed configurations can refer to the library function configEPWMDelayTrip(uint32_t base1, EPWM_DigitalCompareTripInput aux_trip_source).

As shown in Figure 4, since the force to low action triggered by CBC trip will not be changed by the actions for T1 and T2 events, both EPWM1B and EPWM2B can be configured to clear low at the auxiliary trip source signal (EPWM7A) high, and set high at the auxiliary trip source signal low, no matter during positive or negative cycles. Similar to the previous section, it is required to configure the T1 and T2 events for EPWM1B and EPWM2B with the auxiliary trip source signal through the action-qualifier submodule. The below code snippet shows the action settings for EPWMxB.

```
// DCBL = TRIPIN7 = INPUT X-BAR 3 = EPWM7A in this example
EPWM_selectDigitalCompareTripInput(base1, aux_trip_source, EPWM_DC_TYPE_DCBL);

 //
 // DCBL = High, Trigger DCBEVT1 when auxiliary PWM goes high
 //
 EPWM_setTripZoneDigitalCompareEventCondition(base1, EPWM_TZ_DC_OUTPUT_B1,
                                              EPWM_TZ_EVENT_DCXL_HIGH);

 //
 // DCBL = low, Trigger DCBEVT2 when auxiliary PWM goes low
 //
 EPWM_setTripZoneDigitalCompareEventCondition(base1, EPWM_TZ_DC_OUTPUT_B2,
                                              EPWM_TZ_EVENT_DCXL_LOW);

 // DCBEVT1/2 event as AQ T1/2
 // T1 = DCBEVT1
 // T2 = DCBEVT2
 //
 EPWM_setActionQualifierT1TriggerSource(base1,
                                        EPWM_AQ_TRIGGER_EVENT_TRIG_DCB_1);
 EPWM_setActionQualifierT2TriggerSource(base1,
                                        EPWM_AQ_TRIGGER_EVENT_TRIG_DCB_2);

 //
 // set T1/T2 action setting for EPWMxB
 //
 EPWM_setActionQualifierAction(base1, EPWM_AQ_OUTPUT_B, EPWM_AQ_OUTPUT_LOW,
                               EPWM_AQ_OUTPUT_ON_T1_COUNT_UP);
 EPWM_setActionQualifierAction(base1, EPWM_AQ_OUTPUT_B, EPWM_AQ_OUTPUT_LOW,
                               EPWM_AQ_OUTPUT_ON_T1_COUNT_DOWN);
 EPWM_setActionQualifierAction(base1, EPWM_AQ_OUTPUT_B, EPWM_AQ_OUTPUT_HIGH,
                               EPWM_AQ_OUTPUT_ON_T2_COUNT_UP);
 EPWM_setActionQualifierAction(base1,EPWM_AQ_OUTPUT_B, EPWM_AQ_OUTPUT_HIGH,
                               EPWM_AQ_OUTPUT_ON_T2_COUNT_DOWN);
```

## 2.6 Swapping EPWM Configurations During Zero Crossing Point

Since grid-tied inverter needs to take care of the control during both positive and negative cycles, the EPWM1 and EPWM2 control signals should be swapped, together with related protection logic, at the zero crossing point.

The main challenge is how to minimize the software overhead during zero cross point. For example, as for EPWM1B, it is complementary to EPWM1A during the positive cycle, with CBC action at the trip event, while during the negative cycle, EPWM1B is required to delay the trip action, independently with EPWM1A. It is required to change lots of EPWM settings, which will induce software overhead and be not reliable for the system.

Fortunately, with the DBCTL[OUT_MODE] register in the dead-band submodule, as shown in Figure 2-3, it is possible to select the different sources for EPWMxB through a single bit change during the zero cross point. As shown in Figure 2-4, during the positive cycle, EPWM1B is sourced form EPWM1A together with both falling-edge and rising-edge delay enabled; during the negative cycle, EPWM1B bypasses the dead-band submodule and uses the original source with the normally high signal.

Finally, combined with the all the previous settings, the completed block diagram of the EPWM configurations can be found in Figure 2-5.

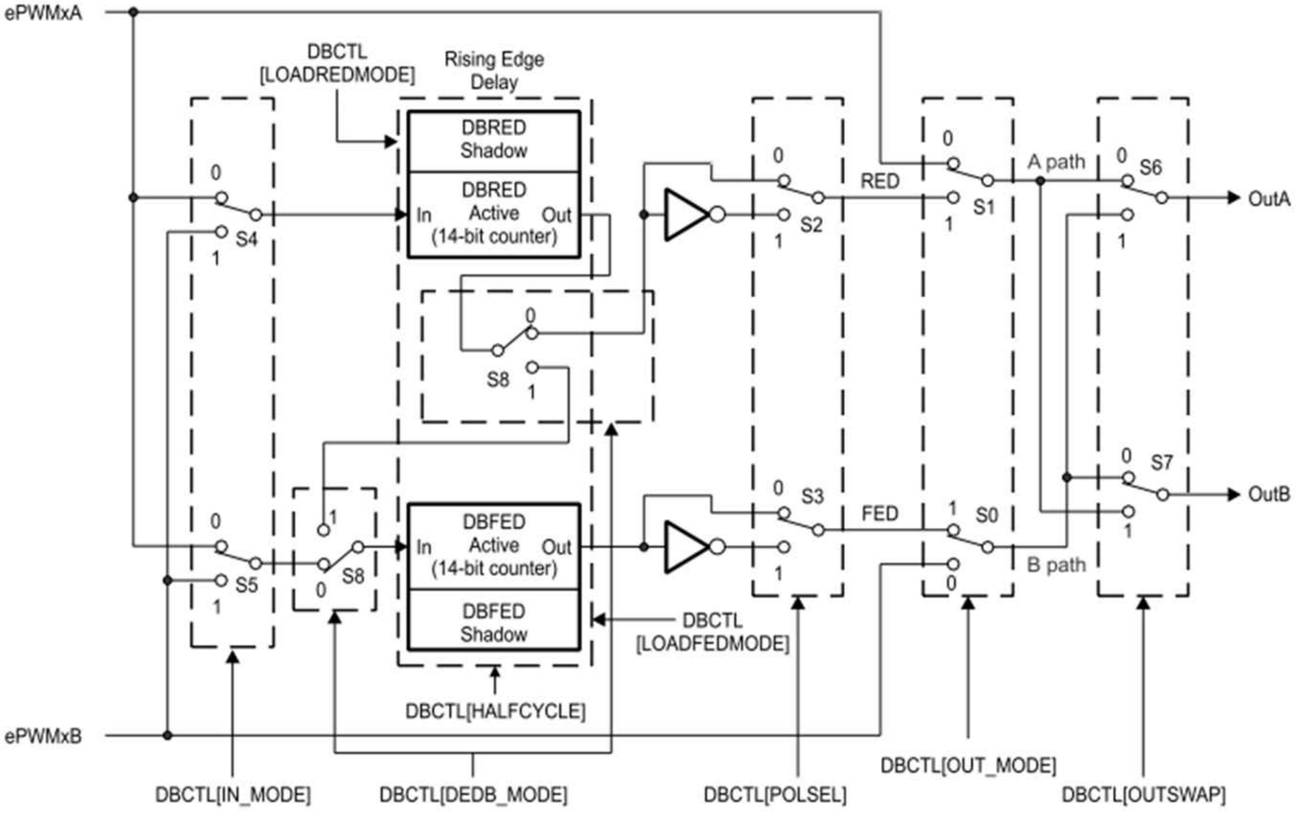


**Figure 2-3. Configuration Options for the Dead-Band Submodule**

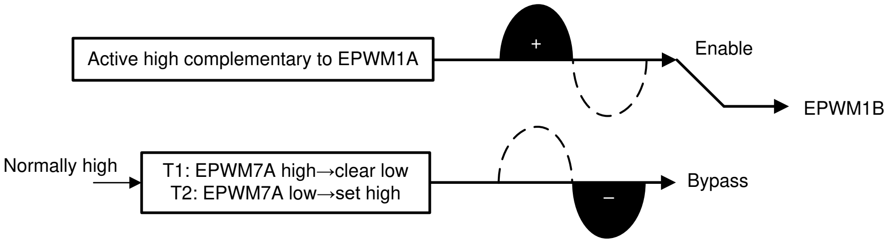

Active high complementary to EPWM1A
+
Enable
EPWM1B
Normally high
T1: EPWM7A high→clear low
T2: EPWM7A low→set high
−
Bypass

**Figure 2-4. Different Sources Selection for EPWM1B**

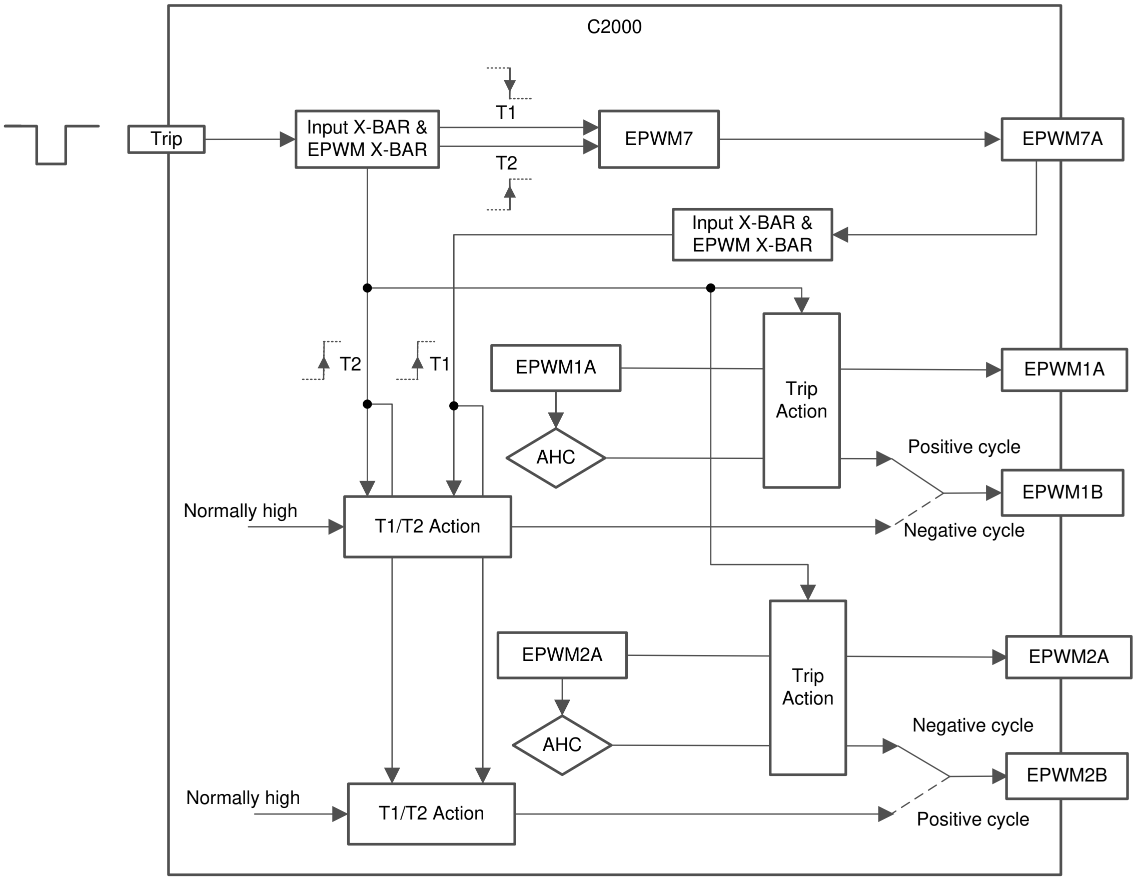


**Figure 2-5. Block Diagram of the EPWM Configurations**

The below codes, found in ISR1(), show the basic settings required during the zero-crossing point. The negative_flag and positive_flag are used to make sure the EPWM reconfiguration is implemented once.

```
if(positive_cycle)
{
    EPWM_setCounterCompareValue(S1_S3_PWM_BASE, EPWM_COUNTER_COMPARE_A,
                                Test_Cmpa);
    if(negative_flag)
    {
        EPWM_setCounterCompareValue(S2_S4_PWM_BASE, EPWM_COUNTER_COMPARE_A,
                                    PERIOD_TICKS);
        EPWM_setDeadBandDelayMode(S1_S3_PWM_BASE, EPWM_DB_FED, true);
        EPWM_setDeadBandDelayMode(S2_S4_PWM_BASE, EPWM_DB_FED, false);

        EPWM_setTripZoneAction(S1_S3_PWM_BASE, EPWM_TZ_ACTION_EVENT_TZB,
                               EPWM_TZ_ACTION_LOW);
        EPWM_setTripZoneAction(S2_S4_PWM_BASE, EPWM_TZ_ACTION_EVENT_TZB,
                               EPWM_TZ_ACTION_DISABLE);

        negative_flag=0;
        positive_flag=1;
    }
}

else
{
    EPWM_setCounterCompareValue(S2_S4_PWM_BASE, EPWM_COUNTER_COMPARE_A,
                                Test_Cmpa);
    if(positive_flag)
    {
        EPWM_setCounterCompareValue(S1_S3_PWM_BASE, EPWM_COUNTER_COMPARE_A,
                                    PERIOD_TICKS);
        EPWM_setDeadBandDelayMode(S1_S3_PWM_BASE, EPWM_DB_FED, false);
        EPWM_setDeadBandDelayMode(S2_S4_PWM_BASE, EPWM_DB_FED, true);

        EPWM_setTripZoneAction(S1_S3_PWM_BASE, EPWM_TZ_ACTION_EVENT_TZB,
                               EPWM_TZ_ACTION_DISABLE );
        EPWM_setTripZoneAction(S2_S4_PWM_BASE, EPWM_TZ_ACTION_EVENT_TZB,
                               EPWM_TZ_ACTION_LOW);

        positive_flag =0;
        negative_flag = 1;
    }
}
```

Table 2-1 has also summarized the different EPWM settings for positive and negative cycles. During the zero crossing point, CBC protection and normal operation actions should be swapped for EPWM1 and EPWM2. For example, at the transition from negative cycle to positive cycle, EPWM1B should change to be complementary to EPWM1A, by enabling the dead-band mode for EPWM1B, while EPWM2B needs to bypass the dead-band mode. Beside, CBC trip action should be disabled for EPWM2B and enabled for EPWM1B, so that EPWM2B state will be only influenced by T1 and T2 events to achieve delayed protection. Also, the switching for Test_Cmpa ↔ PERIOD_TICKS for the CMPA register should be taken care, where CMPA is directly related to the duty cycle provided by the control loop, that is CMPA=(1- duty cycle)* PERIOD_TICKS.

**Table 2-1. Different EPWM Settings for Positive and Negative Cycles**

| PWM Signals | Basic Settings | Positive Cycle | Negative Cycle |
|---|---|---|---|
| **EPWM1A-S1** | ↑ CAU, PRD<br>↓ ZRO, CAD<br>Enable CBC, forced low action | CMPA = Test_Cmpa | CMPA = PERIOD_TICKS |
| **EPWM2B-S2** | One time Software Force high; active high complementary to EPWM2A;<br>↓T1<br>↑T2<br>T1 = DCBEVT1(EPWM7A high)<br>T2 = DCBEVT2(EPWM7A low)<br>Enable CBC | Bypass dead-band mode;<br>Disable CBC action | Enable dead-band mode;Enable CBC forced low action |
| **EPWM1B-S3** | One time Software Force high;active high complementary to EPWM1A;<br>↓T1<br>↑T2<br>T1 = DCBEVT1(EPWM7A high)<br>T2 = DCBEVT2(EPWM7A low)<br>Enable CBC | Enable dead-band mode;<br>Enable CBC forced low action | Bypass dead-band mode;<br>Disable CBC trip action |
| **EPWM2A-S4** | ↑ CAU, PRD<br>↓ ZRO, CAD | CMPA = PERIOD_TICKS | CMPA = Test_Cmpa |
| **EPWM7A** | ↑ T1<br>↓ T2<br>RED = trip_delay (1us), T1 = DCBEVT1(Trip low), T2 = DCBEVT2(Trip high ) | | |

# 3 Test Results

The delayed protection scheme has been validated with the LaunchPad LAUNCHXL-F280049C. In the given example codes, changing positive_cycle to 1 or 0 to decide positive cycle or negative cycle operation; changing simulate_EPWM_Cmpa value to provide different time scale for the trip signal (EPWM8A). The test results are measured with the Kingst Logic Analyzer, as shown in Figure 3-1.

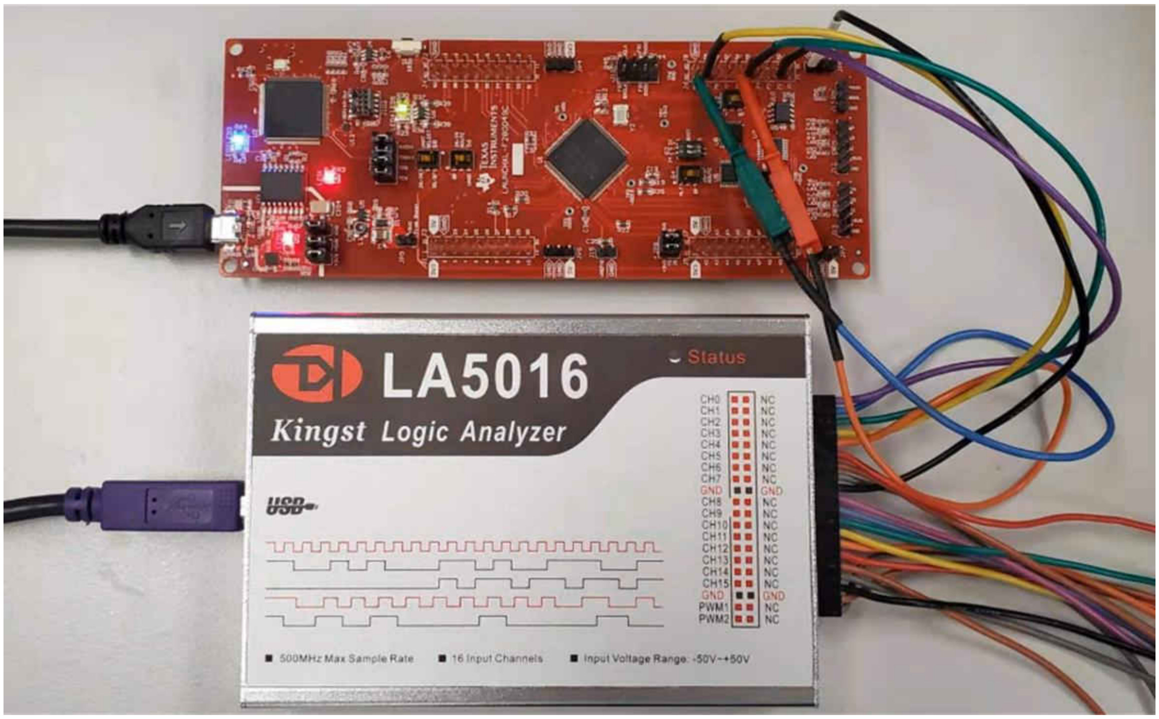
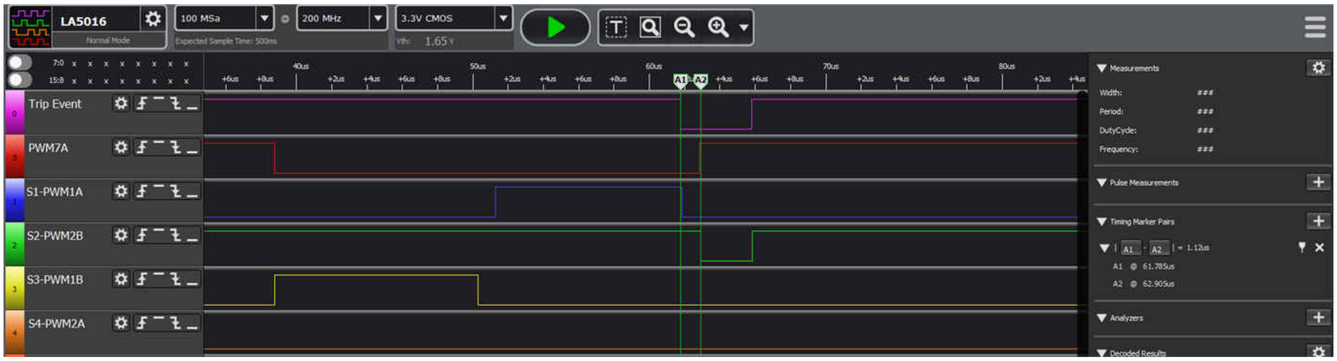


**Figure 3-1. Test platform Setup**

Figure 3-2 shows the condition of trip signal during positive cycle, where EPWM1A starts CBC protection right at the trip low event, while EPWM2B turns low after a delay of 1.12 µs measured. Since EPWM2B needs to follow the action of EPWM7A, the actual delay would be a bit longer than the defined value of 1 µs. Figure 3-3 shows the trip signal during negative cycle, where the delayed protection logic also works as expected.

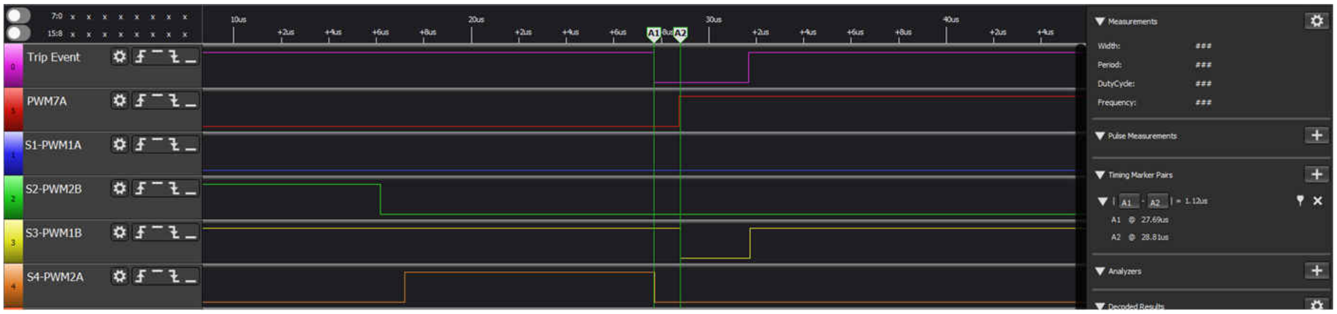


100 MSa
200 MHz
Expected Sample Time: 500ms
3.3V CMOS
Vth: 1.65 V

Width: ###
Period: ###
DutyCycle: ###
Frequency: ###

| A1 - A2 | = 1.12us
A1 @ 61.785us
A2 @ 62.905us

**Figure 3-2. Trip Signal during Positive Cycle**



Width: ###
Period: ###
DutyCycle: ###
Frequency: ###

| A1 - A2 | = 1.12us
A1 @ 27.69us
A2 @ 28.81us

**Figure 3-3. Trip Signal during Negative Cycle**

## 4 References

- Texas Instruments: *TMS320F28004x Piccolo Microcontrollers Technical Reference Manual*
- Texas Instruments: *TMS320F28004x Microcontrollers Data Manual*
- Texas Instruments: *Achieve Delayed Protection for Three-Level Inverter With CLB*
- Rodriguez, Jose, et al. "A survey on neutral-point-clamped inverters." IEEE transactions on Industrial Electronics 57.7 (2009): 2219-2230.
- Kim, Youngroc, et al. "Design and control of a grid-connected three-phase 3-level NPC inverter for building integrated photovoltaic systems." 2012 IEEE PES Innovative Smart Grid Technologies (ISGT). IEEE, 2012.

# 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