Configuring Source of Multiple ePWM Trip-Zone Events

Jeff Stafford *AEC C2000*

ABSTRACT

Applications that require PWM outputs to respond to multiple trip-zones (TZ) are challenged to determine the specific source of the TZ event.

This application report describes how to configure an ePWM module to identify the specific source of a TZ event when multiple TZ events are enabled to act on a single PWM channel.

Project collateral and source code discussed in this application report can be downloaded from the following URL: http://www.ti.com/lit/zip/SPRAAR4.

Contents

List of Figures

List of Tables

1 Trip-Zone Introduction

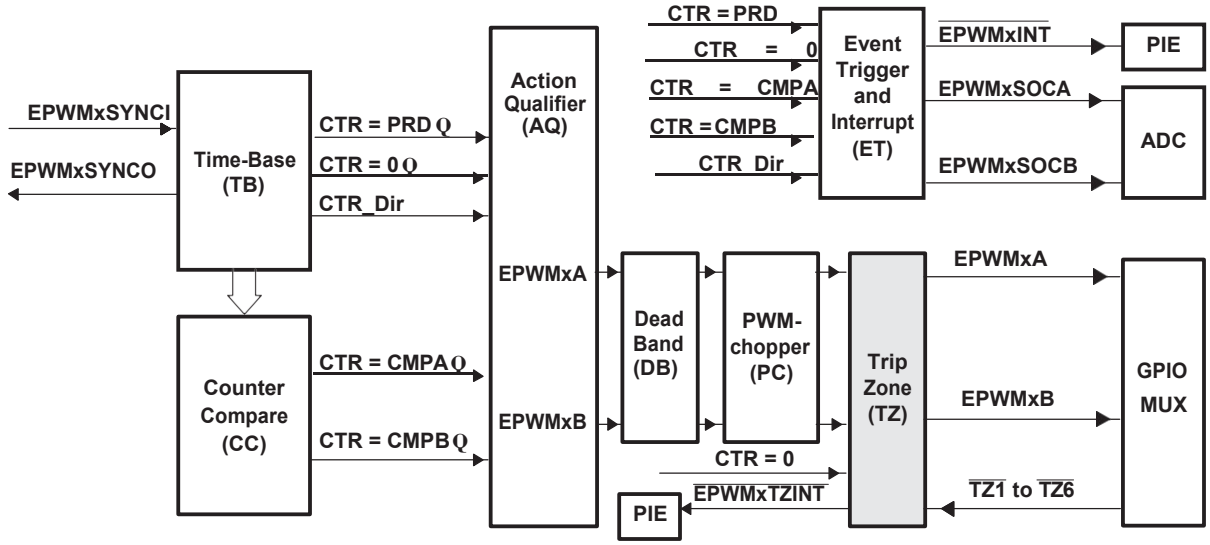

Figure 1. Trip-Zone Submodule

Each ePWM module is connected to six $\overline{TZn}$ signals ($\overline{TZ1}$ to $\overline{TZ6}$) that are sourced from the general-purpose input/output (GPIO) MUX. These signals indicate external fault or trip conditions, and the ePWM outputs can be programmed to respond accordingly when faults occur.

1.1 Purpose of the Trip-Zone Submodule

The key functions of the trip-zone submodule are:

- Trip inputs $\overline{TZ1}$ to $\overline{TZ6}$ can be flexibly mapped to any ePWM module.
- Upon a fault condition, outputs EPWMxA and EPWMxB can be forced to one of the following:
  - High
  - Low
  - High-impedance
  - No action taken
- Support for one-shot trip (OSHT) for major short circuits or over-current conditions
- Support for cycle-by-cycle tripping (CBC) for current limiting operation
- Each trip-zone input pin can be allocated to either one-shot or cycle-by-cycle operation.
- Interrupt generation is possible on any trip-zone pin.
- Software-forced tripping is also supported.
- The trip-zone submodule can be fully bypassed if it is not required.

2 Challenge of Multiple Trip-Zones

Each ePWM output (EPWMxA and EPWMxB) can be sourced by any or all of the six $\overline{TZn}$ inputs ($\overline{TZ1}$ to $\overline{TZ6}$). Since there is only one TZ flag for a selected TZ type (one-shot or cycle-by-cycle), the source of the TZ event can not be determined if multiple TZs are enabled for a single ePWM output. One solution is to enable all the TZ interrupts.

Since all TZs share PIE group 2, the code could read PIEIFR2 to determine which TZ's flag is set upon entry into the interrupt service routine. The problem is that the hardware automatically clears the PIEIFRx.y bit when fetching the interrupt vector from the PIE, and before entry into the ISR; so the TZ source identifier is lost. A flowchart of the hardware interrupt response is shown in the *TMS320x280x, 2801x, 2804x System Control and Interrupts Reference Guide* (SPRU712).

3 Determining Trip-Zone Source

In addition to configuring a PWM channel to trip on multiple TZs, enable the same TZs for other PWM channels assigning a single TZ to the PWM channel. This allows polling the TZFLG bits for the other PWM channels to determine the TZ source. You can configure the TZ event to have no action on the PWM channel in the TZCTL register. This solution avoids the requirement of latching the TZ events with external circuits.

If external latching of the TZ event is already available, then the GPIO inputs that are shared with the TZ pins can be read to determine the logic level. For instance, with the TMS320F2808 device, $\overline{TZ1}$ shares the same pin with GPIO12, $\overline{TZ2}$ with GPIO13, etc. An example of this is shown in the GPIO MUX Block Diagram in the *TMS320F2809, TMS320F2808, TMS320F2806, TMS320F2802, TMS320F2801, TMS320C2802, TMS320C2801, and TMS320F2801x DSPs Data Manual* (SPRS230).

3.1 Enable Multiple Trip-Zones for a Single ePWM Output

Figure 2 is an illustration of the TZ control logic that is designed into every ePWM module. Note that all six of the TZs are available to the two ePWM channels. Each TZ can be configured for either one-shot or cycle-by-cycle modes, or both.

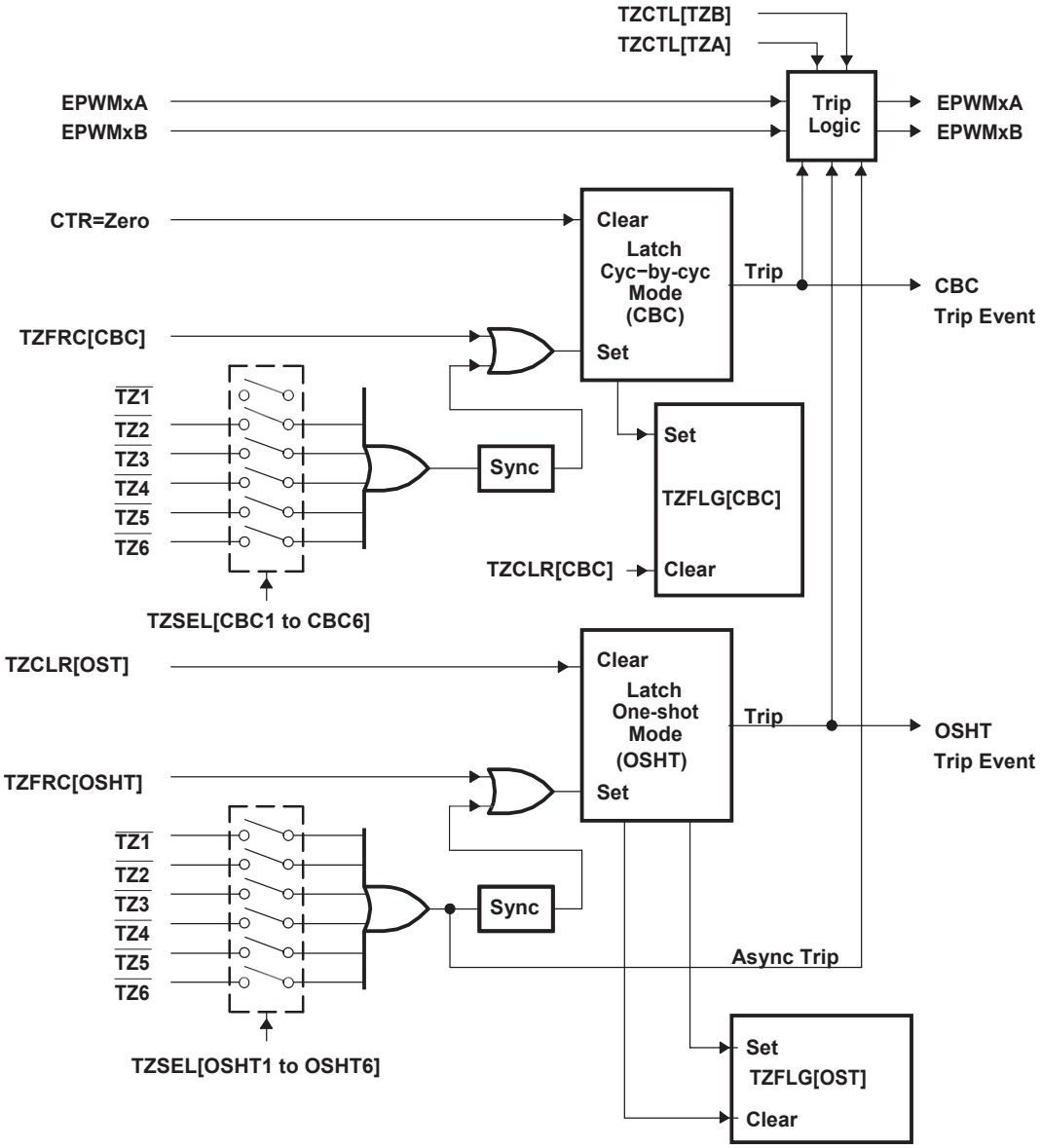

Figure 2. Trip-Zone Submodule Mode Control Logic

The TZSEL register allows each TZ to be enabled for a single ePWM. With multiple TZs enabled, a single CBC or OSHT trip event occurs, see Figure 3. The trip event sets the corresponding TS interrupt flag, CBC-TZFLG or OSHT-TZFLG. These flags can be used to generate an interrupt to respond to the event. In the ISR code, one of the first tasks is to determine which TZ event caused the interrupt.

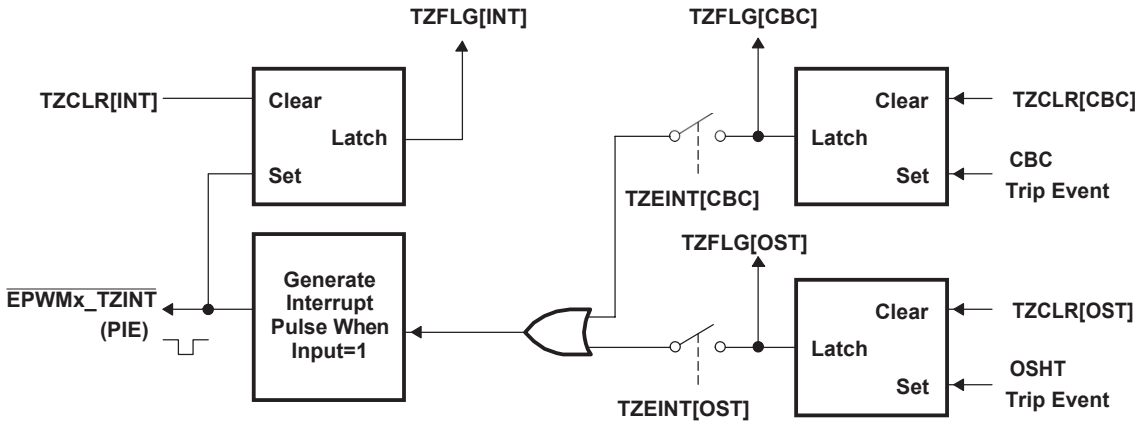

Figure 3. Trip-Zone Submodule Interrupt Logic

Also, this trip event results in a particular action on the selected ePWM channel, see Table 1. Note that EPWMxA and EPWMxB have separate control over their respective action in response to a trip event.

Table 1. Trip-Zone Control Register (TZCTL) Field Descriptions

Bits	Name	Value	Description
15–4	Reserved		Reserved
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 in the *TMS320x280x, 2801x, 2804x System Control and Interrupts Reference Guide* (SPRU712).
		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.
1–0	TZA		When a trip event occurs the following action is taken on output EPWMxA. Which trip-zone pins can cause an event is defined in the TZSEL register in the *TMS320x280x, 2801x, 2804x System Control and Interrupts Reference Guide* (SPRU712).
		00	High impedance (EPWMxA = High-impedance state)
		01	Force EPWMxA to a high state
		10	Force EPWMxA to a low state
		11	Do nothing, no action is taken on EPWMxA.

3.2 Determining Source of TZ Event

The challenge is to determine the specific TZ event that causes a TZ related interrupt when multiple TZ events are enabled. There are no interrupts directly associated with a specific TZ event. Instead, TZ events are associated with a specific ePWM module. For example, any of the enabled TZ events for ePWM1 could cause the bit PIEIFR2.1 to be set. If PIEIER2.1 is also set, then the interrupt EPWM1_TZINT would be generated. But the source of the interrupt could have been any of the six $\overline{TZn}$ events.

The solution is to enable a single TZ event for additional ePWM modules.

In addition to configuring a PWM channel to trip on multiple TZs, enable the same TZs for other PWM channels assigning a single TZ to a PWM channel. This allows polling the TZFLG bits for the other PWM channels to determine the TZ source. Configure the additional TZ events to have no action on the PWM channel in the TZCTL register.

3.3 Example

This example shows how to implement the technique presented in this application report. In general, example software for TZs is provided with the peripheral example code package for each 28x generation or parts. For this application report, the *C280x, C2801x C/C++ Header Files and Peripheral Examples* (SPRC191) was used on an F2808 eZdsp™.

This code was modified to enable all six $\overline{\text{TZn}}$s for ePWM1, each being configured to set the EPWM1A output high on a TZ event. TZs 2-56 were enabled for ePWMs 2-6, with one TZ enabled per ePWM module and with each TZ configured for no action on a TZ event. The ePWM1 TZ interrupt was modified to determine the source of the TZ event.

3.3.1 Software Configuration

This section discusses the configuration that causes the EPWM1A to be forced high when any of the six $\overline{\text{TZn}}$ events occur. When the EPWM_TZINT occurs, poll the TZFLG bits for ePWMs1-5 to determine the source of the TZ event. If none are set, then $\overline{\text{TZ6}}$ is the source. This example is to the maximum number of TZ events that could be polled, but a configuration of less enabled TZ events could also be used.

```
void InitEPwm1Example()
{
   // Enable all TZs as one shot trip sources
   EALLOW;
   EPwm1Regs.TZSEL.bit.OSHT1 = 1;
   EPwm1Regs.TZSEL.bit.OSHT2 = 1;
   EPwm1Regs.TZSEL.bit.OSHT3 = 1;
   EPwm1Regs.TZSEL.bit.OSHT4 = 1;
   EPwm1Regs.TZSEL.bit.OSHT5 = 1;
   EPwm1Regs.TZSEL.bit.OSHT6 = 1;

   // What do we want any of the TZs to do?
   EPwm1Regs.TZCTL.bit.TZA = TZ_FORCE_HI;
   EPwm1Regs.TZCTL.bit.TZB = TZ_NO_CHANGE;

   // Enable TZ interrupt on any TZ event
   EPwm1Regs.TZEINT.bit.OST = 1;
   EDIS;

void InitEPwm2Example()
{

   // Enable TZ1 as one shot trip source
   EALLOW;
   EPwm2Regs.TZSEL.bit.OSHT2 = 1;

   // Do not impact PWM outputs on TZ event
   EPwm2Regs.TZCTL.bit.TZA = TZ_NO_CHANGE;
   EPwm2Regs.TZCTL.bit.TZB = TZ_NO_CHANGE;
   EDIS;
}
```

3.3.2 Procedure

With the example software loaded and running in real-time on the F2808 eZdsp, add the variable EPwm1TZIntCount to the Watch Window and continuously refresh this window.

Connect an oscilloscope to EPWM1A (shared with GPIO0) found on the F2808 eZdsp's P8 Connector pin #9 (P8-9). Note that the EPWM1A output is active. You should see PWM with 50% duty cycle.

Connect a jumper to the ground located on P8-39. Connect the other end of the jumper to $\overline{TZ1}$ (shared with GPIO12) on P8-37. Note that EPWM1A's output is now forced high and that EPwm1TZIntCount in the Watch Window is increasing in count indicating that the ePWM1 interrupt is being serviced in response to $\overline{TZ1}$ being active.

Now repeat this test for the rest of the TZs.

1. Connect the jumper from P8-9 to P8-17 for $\overline{TZ2}$ (shared with GPIO13)
2. Connect the jumper from P8-9 to P8-5 for $\overline{TZ3}$ (shared with GPIO14)
3. Connect the jumper from P8-9 to P8-22 3) for $\overline{TZ4}$ (shared with GPIO15)
4. Connect the jumper from P8-9 to P8-23 for $\overline{TZ5}$ (shared with GPIO16)
5. Connect the jumper from P8-9 to P8-24 for $\overline{TZ6}$ (shared with GPIO17)

For each of these tests you will see EPWM1A's output forced high and EPwm1TZIntCount continue to increase its count.

Table 2. F2808 eZdsp P8 Connector

P8 Pin #	P8 Signal	P8 Pin #	
1	+ 3.3 V/+5 V /NC	2	+ 3.3 V/+5 V /NC
3	MUX_GPIO29	4	MUX_GPIO28
5	GPIO14	6	GPIO20
7	GPIO21	8	GPIO23
9	GPIO0	10	GPIO1
11	GPIO2	12	GPIO13
13	GPIO4	14	GPIO15
15	GPIO27	16	GPIO16
17	GPIO13	18	GPIO134
19	GND	20	GND
21	GPIO7	22	GPIO15
23	GPIO16	24	GPIO17
25	GPIO18	26	GPIO19
27	MUX_GPIO31	28	MUX_GPIO30
29	MUX_GPIO11	30	MUX_GPIO8
31	GPIO22/GPIO24	32	MUX_GPIO10
33	GPIO25	34	GPIO25
35	GPIO26	36	GPIO32
37	GPIO12	38	GPIO33
39	GND	40	GND

Table 3. GPIO Pins Shared With TZs

GPAMUX1/2[1] Register Bits	Default at Reset Primary I/O Function (GPxMUX1/2 bits = 0,0)	Peripheral Selection 1[2] (GPxMUX1/2 bits = 0,1)	Peripheral Selection 2 (GPxMUX1/2 bits = 1,0)	Peripheral Selection 3 (GPxMUX1/2 bits = 1,1)
		GPAMUX1		
1-0	GPIO0	EPWM1A (O)	Reserved[3]	Reserved[3]
3-2	GPIO1	EPWM1B (O)	SPISIMOD (I/O)	Reserved[3]
5-4	GPIO2	EPWM2A (O)	Reserved[3]	Reserved[3]
7-6	GPIO3	EPWM2B (O)	SPISOMID (I/O)	Reserved[3]
9-8	GPIO4	EPWM3A (O)	Reserved[3]	Reserved[3]
11-10	GPIO5	EPWM3B (O)	SPICLKD (I/O)	ECAP1 (I/O)
13-12	GPIO6	EPWM4A (O)	EPWMSYNCI (I)	EPWMSYNCO (O)
15-14	GPIO7	EPWM4B (O)	SPISTED (I/O)	ECAP2 (I/O)
17-16	GPIO8	EPWM5A (O)	CANTXB (O)	$\overline{\text{ADCSOCAO}}$ (O)
19-18	GPIO9	EPWM5B (O)	SCITXDB (O)	ECAP3 (I/O)
21-20	GPIO10	EPWM6A (O)	CANRXB (I)	$\overline{\text{ADCSOCBO}}$ (O)
23-22	GPIO11	EPWM6B (O)	SCIRXDB (I)	ECAP4 (I/O)
25-24	GPIO12	$\overline{\text{TZ1}}$ (I)	CANTXB (O)	SPISIMOB (I/O)
27-26	GPIO13	$\overline{\text{TZ2}}$ (I)	CANRXB (I)	SPISOMIB (I/O)
29-28	GPIO14	$\overline{\text{TZ3}}$ (I)	SCITXDB (O)	SPICLKB (I/O)
31-30	GPIO15	$\overline{\text{TZ4}}$ (I)	SCIRXDB (I)	SPISTEB (I/O)
		GPAMUX2		
1-0	GPIO16	SPISIMOA (I/O)	CANTXB (O)	$\overline{\text{TZ5}}$ (I)
3-2	GPIO17	SPISOMIA (I/O)	CANRXB (I)	$\overline{\text{TZ6}}$ (I)

(1) The word *Reserved* means that there is no peripheral assigned to this GPxMUX1/2 register setting. Should it be selected, the state of the pin is undefined and the pin may be driven. This selection is a reserved configuration for future expansion.
(2) GPxMUX1/2 refers to the appropriate MUX register for the pin, GPAMUX1, GPAMUX2 or GPBMUX1.
(3) This table pertains to the F2808 device. Some peripherals may not be available in the 2809, 2806, 2802, or 2801 devices. For more details, see the pin descriptions.

4 Conclusion

This application report has shown that it is possible to use multiple TZs for a single PWM channel with the ability to detect the specific TZ source.

5 References

- *TMS320x28xx, 28xxx Enhanced Pulse Width Modulator (ePWM) Module Reference Guide* (SPRU791)
- *TMS320x280x, 2801x, 2804x System Control and Interrupts Reference Guide* (SPRU712)
- *TMS320F2809, TMS320F2808, TMS320F2806, TMS320F2802, TMS320F2801, TMS320C2802, TMS320C2801, and TMS320F2801x DSPs Data Manual* (SPRS230)
- *C280x, C2801x C/C++ Header Files and Peripheral Examples* (SPRC191)
- eZdsp 2808 Technical Reference (http://c2000.spectrumdigital.com/ezf2808/docs/2808_ezdspusb_techref_c.pdf)

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATASHEETS), 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, 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 (www.ti.com/legal/termsofsale.html) 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.

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