Application Report
Lit. Number – Month Year

Determine Source of Multiple ePWM Trip-Zone Events

Jeff Stafford *AEC Field Applications – Semiconductor Group*

ABSTRACT

Applications that require PWM outputs to respond to multiple Trip-Zones (TZ) have the challenge of determining the specific source of the TZ event.

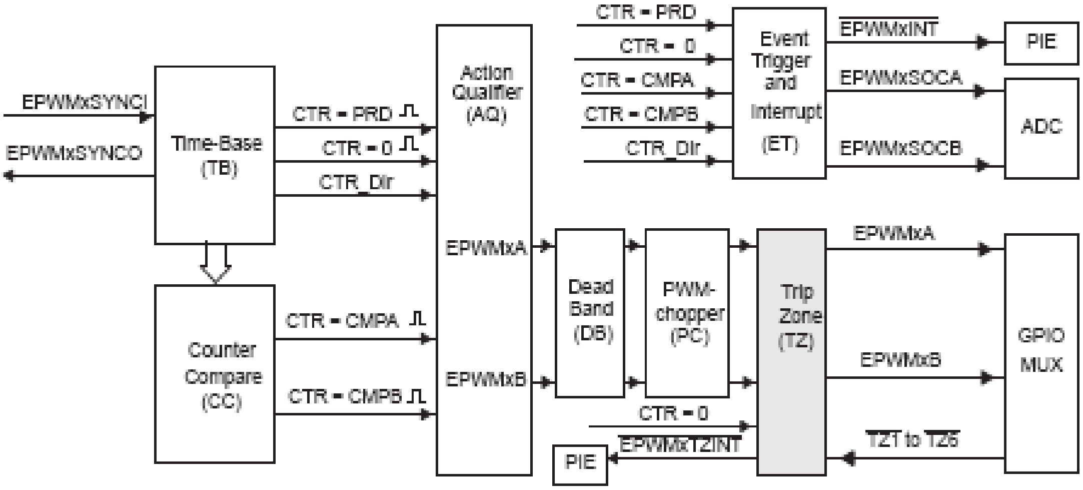
This report describes how to configure a ePWM module to identify the specific source of a TZ event when multiple TZ events are enabled to act on a single PWM channel.

Contents

Figures

1 Trip-Zone Introduction

Figure 2-34. Trip-Zone Submodule



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

2.7.1 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 (6) TZ inputs (TZ1 to TZ6). Since there is only (1) TZ flag for a selected TZ type (One-Shot or Cycle-By-Cycle) the source of TZ event can not be determined if multiple TZs are enabled for a single ePWM output.

One possible solution to this problem would be to enable all the TZ interrupts. Since all TZs share PIE group 2, upon entry into the interrupt service routine the code could read PIEIFR2 to determine which TZ's flag is set. The problem is that the hardware automatically clears the PIEIFRx.y bit when the interrupt vector is fetched from the PIE and before entry into the ISR. So the TZ source identifier is lost. A flowchart of the hardware interrupt response is flowcharted in Figure 6-2 in *TMS320x280x DSP System Control and Interrupts Reference Guide* (SPRU712).

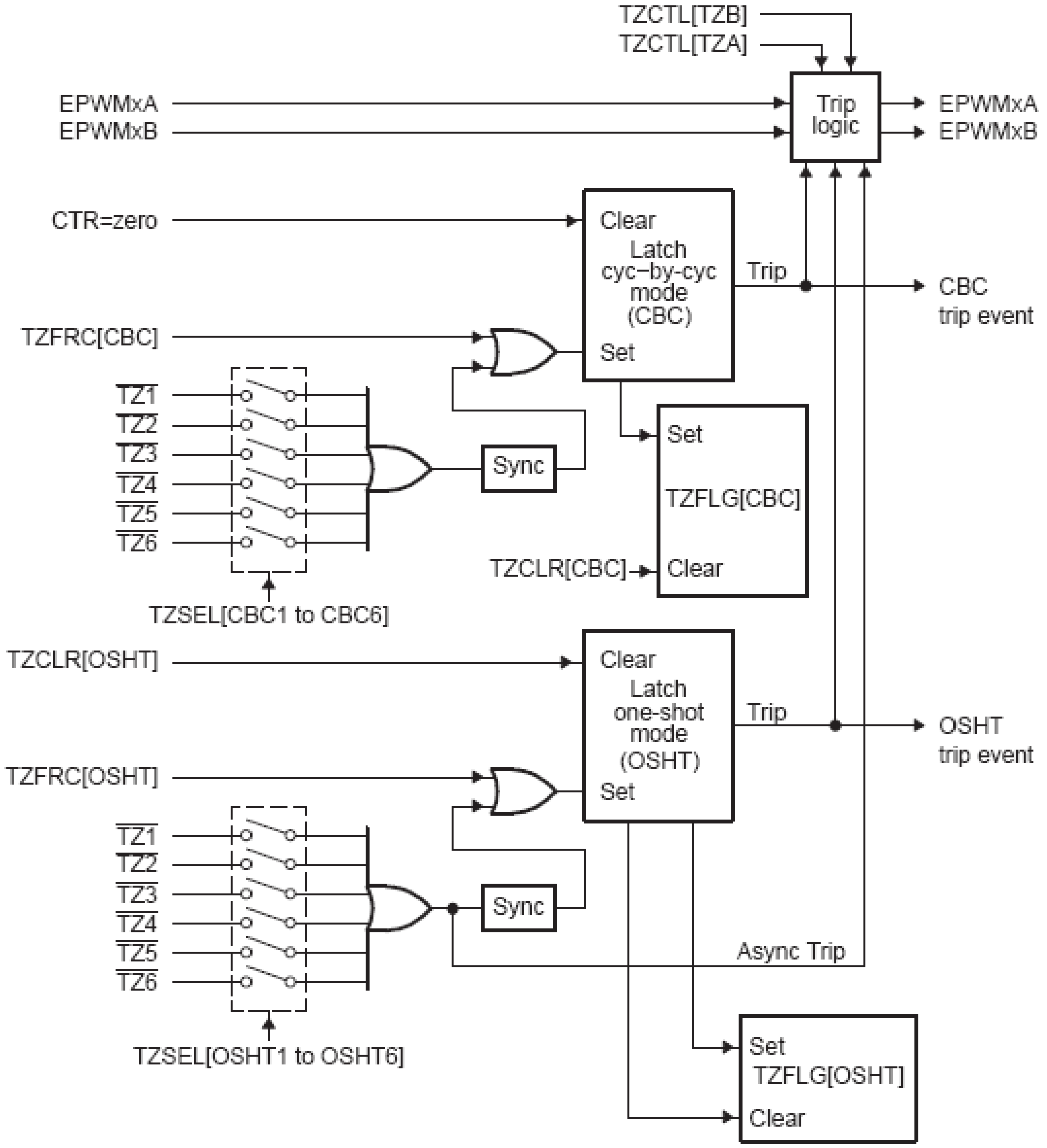
3 Determining Trip-Zone Source

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 PWM channel. This would allow 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 having the TZ events latched with external circuits.

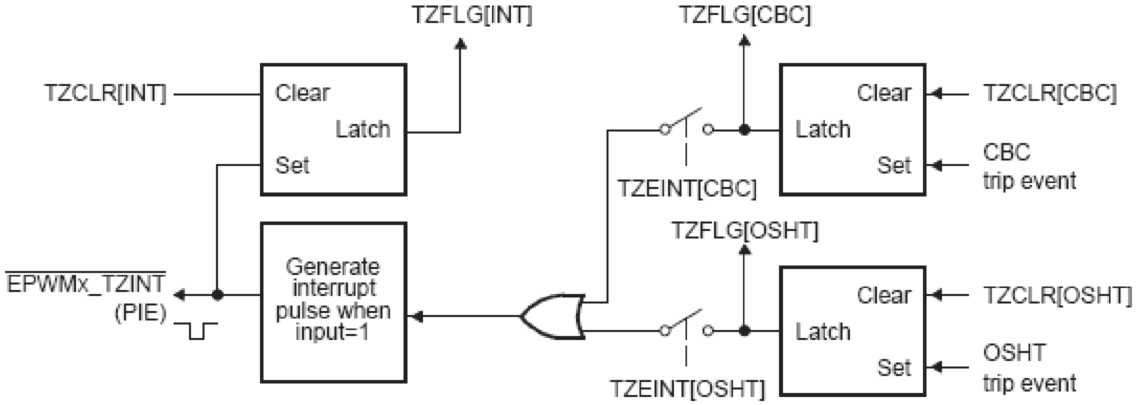
If external latching of the TZ event is already available, then the GPIO inputs that are shared with the TZ pins can be read to determine the logic level. For instance with the TMS320F2808 device TZ1 shares the same pin with GPIO12, TZ2 with GPIO13, etc. This is shown in the GPIO MUX Block Diagram Figure 4-16 of *TMS320F280xx Data Manual* (SPRS230).

3.1 Enable Multiple Trip-Zones for a Single ePWM Output

The figure below is an illustration of the TZ control logic that is designed into every ePWM module. Note that all (6) of the TZs are available to the (2) ePWM channels. Each TZ can be configured for either One-Shot or Cycle-By-Cycle modes, or both.

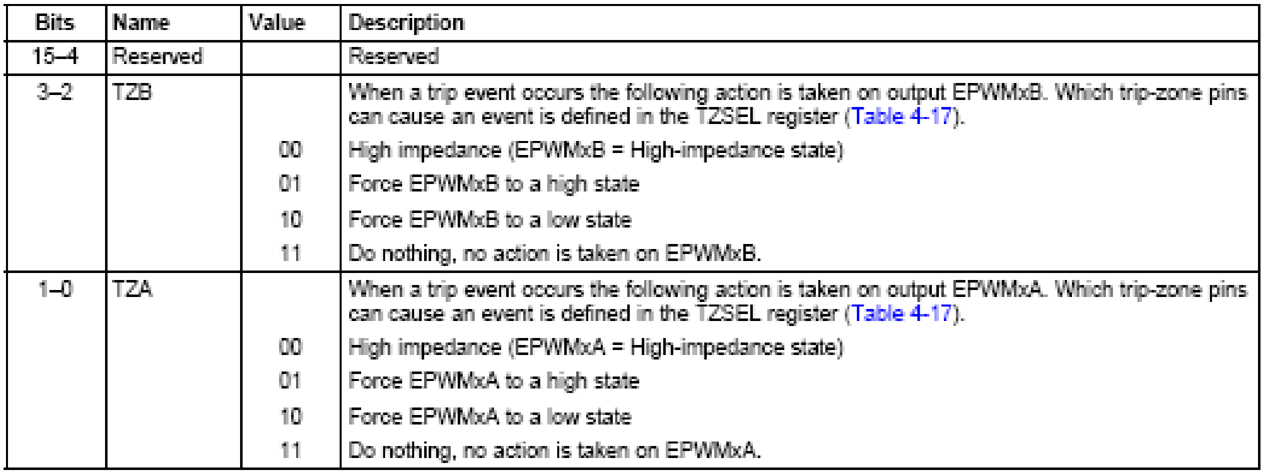


The TZSEL register allows each TZ to be enabled for a single ePWM. With multiple TZs enabled, a single CBC or OSHT trip event occurs. The trip event will set the corresponding TS interrupt flag, CBC-TZFLG or OSHT-TZFLG. These flags can be used to generate an interrupt to respond to the event. In the ISR code one of the 1st tasks will be to determine which TZ event caused the interrupt.



Also, this trip event will result in a selected action on the selected ePWM channel, see figure below. Note that EPWMxA and EPWMxB have separate control over their respective action in response to a trip event.

Table 4-18. 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 (Table 4-17).
		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 (Table 4-17).
		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 (6) TZ 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 aPWM channel. This would allow 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 report. Example software for TZs in general is provided with the peripheral example code package for each 28x family or parts. For this 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 (6) TZs for ePWM1, each being configured to set the EPWM1A output high on a TZ event. TZs 2-56were 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

Here's the configuration for a causing EPWM1A to be forced high when any of the (6) TZ 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 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 ground located on P8-39. Connect the other end of the jumper to TZ1 (shared with GPIO12) on P8-37. Note that EPWM1A's output is now forced high. Also note that EPwm1TZIntCount in the Watch Window is increasing in count indicating that the ePWM1 interrupt is being serviced in response to TZ1 being active.

Now let's repeat this test for the rest of the TZs.

1) For TZ2 (shared with GPIO13) connect the jumper from P8-9 to P8-17
2) For TZ3 (shared with GPIO14) connect the jumper from P8-9 to P8-5
3) For TZ4 (shared with GPIO15) connect the jumper from P8-9 to P8-22
4) For TZ5 (shared with GPIO16) connect the jumper from P8-9 to P8-23
5) For TZ6 (shared with GPIO17) connect the jumper from P8-9 to P8-24

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

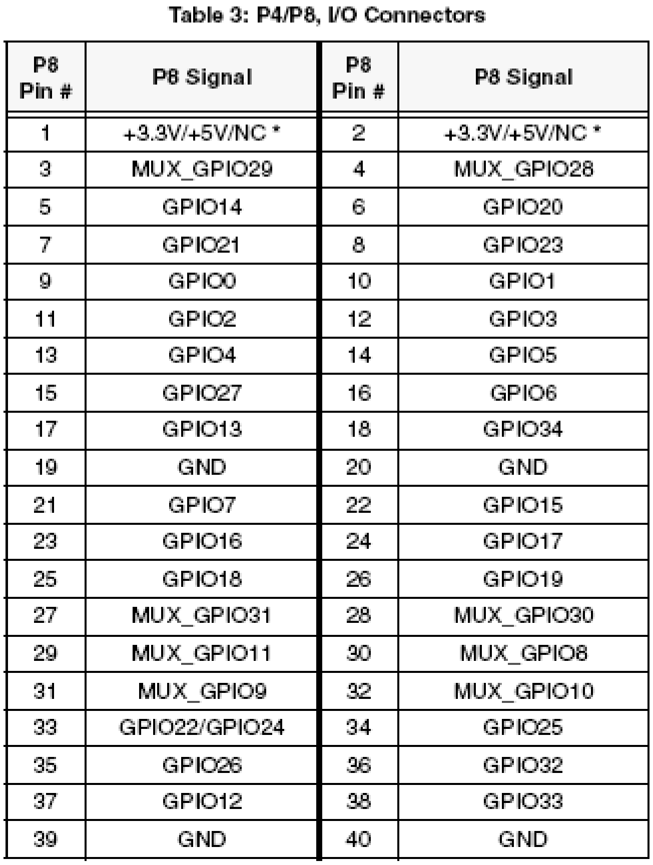


Table 3: P4/P8, I/O Connectors

P8 Pin #	P8 Signal	P8 Pin #	P8 Signal
1	+3.3V/+5V/NC *	2	+3.3V/+5V/NC *
3	MUX_GPIO29	4	MUX_GPIO28
5	GPIO14	6	GPIO20
7	GPIO21	8	GPIO23
9	GPIO0	10	GPIO1
11	GPIO2	12	GPIO3
13	GPIO4	14	GPIO5
15	GPIO27	16	GPIO6
17	GPIO13	18	GPIO34
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	MUX_GPIO9	32	MUX_GPIO10
33	GPIO22/GPIO24	34	GPIO25
35	GPIO26	36	GPIO32
37	GPIO12	38	GPIO33
39	GND	40	GND

Figure 1. F2808 eZdsp P8 Connector

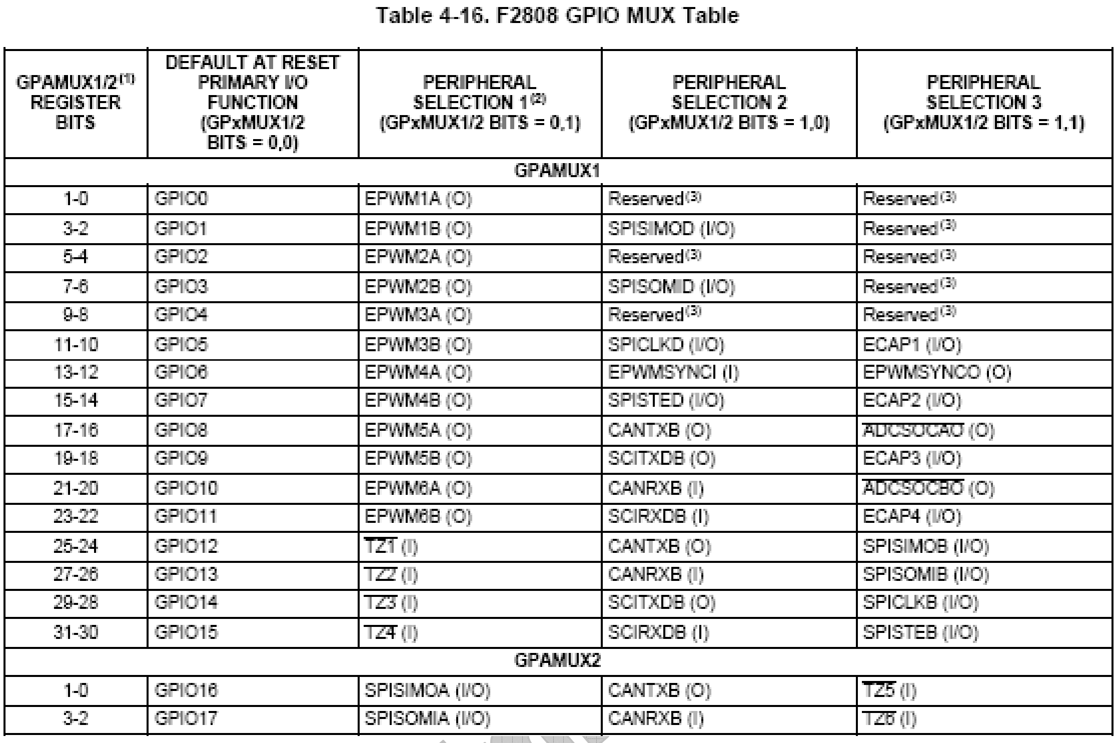


Table 4-16. F2808 GPIO MUX Table

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)

Figure 2. GPIO Pins Shared with TZs

4 Conclusion

This 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.

References

1. TMS320x28xx, 28xxx Enhanced Pulse Width Modulator (ePWM) Module Reference Guide (SPRU791)
2. TMS320x280x DSP System Control and Interrupts Reference Guide (SPRU712)
3. TMS320F280xx Data Manual (SPRS230)
4. C280x, C2801x C/C++ Header Files and Peripheral Examples (SPRC191) (http://focus.ti.com/docs/toolsw/folders/print/sprc191.html)
5. eZdsp 2808 Technical Reference (http://c2000.spectrumdigital.com/ezf2808/docs/2808_ezdspusb_techref_c.pdf)