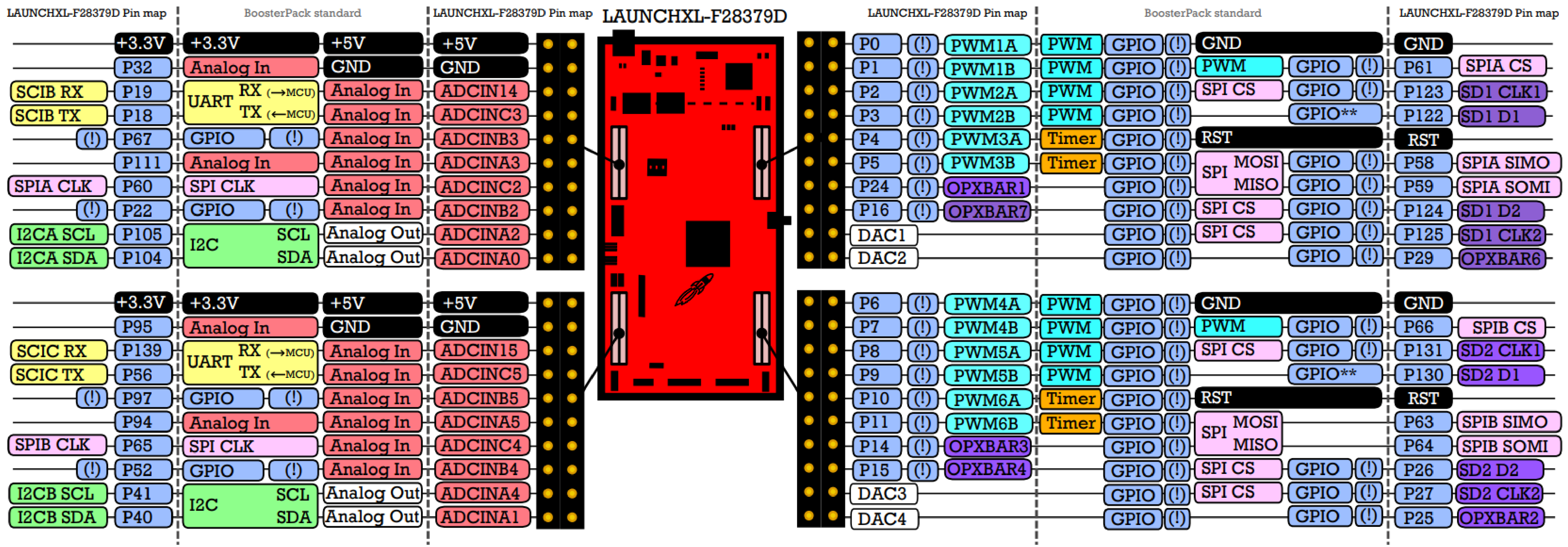
# F28379D ePWM

HCMUTE

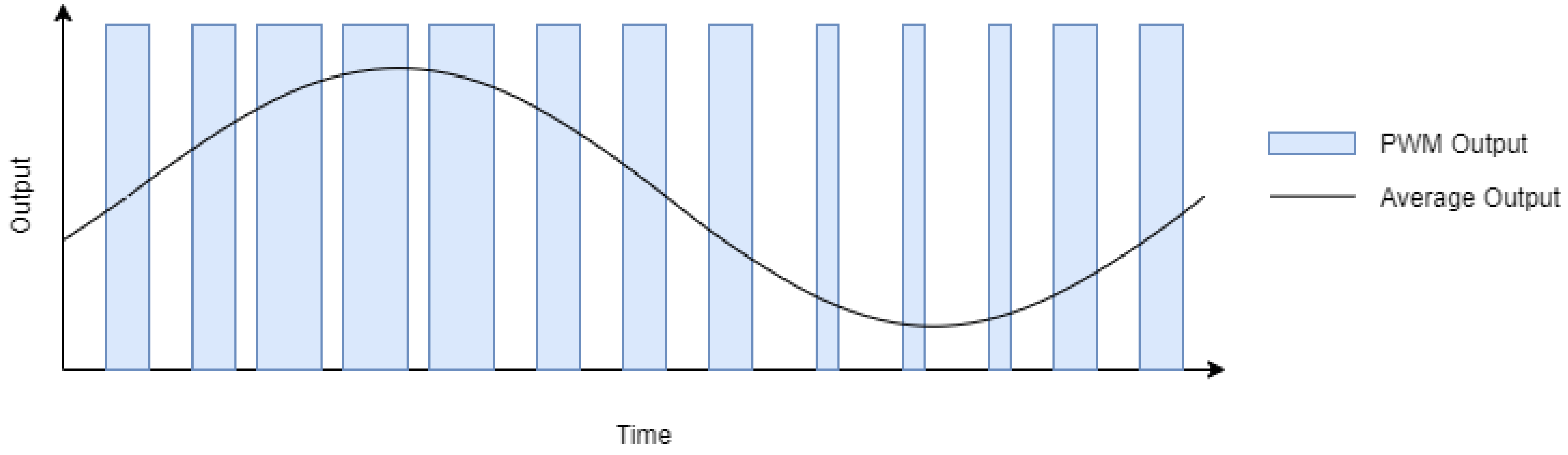
Monna Dang (Dang Hoang Anh Chuong)

02/2024

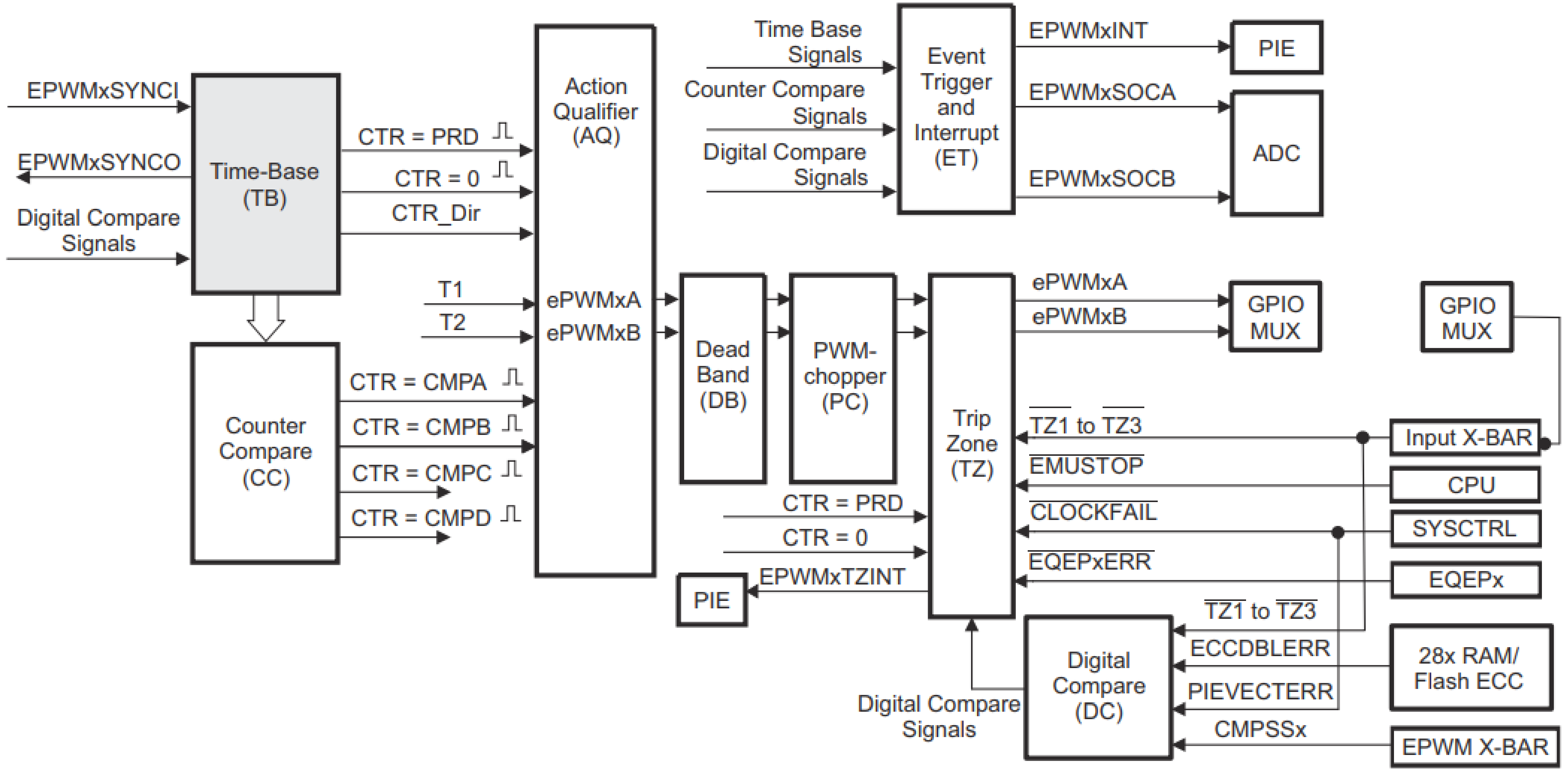

LAUNCHXL-F28379D Pin map
BoosterPack standard
LAUNCHXL-F28379D Pin map
LAUNCHXL-F28379D
LAUNCHXL-F28379D Pin map
BoosterPack standard
LAUNCHXL-F28379D Pin map
+3.3V | +3.3V | +5V | +5V
P32 | Analog In | GND | GND
SCIB RX | P19 | UART RX (→MCU) | Analog In | ADCIN14
SCIB TX | P18 | TX (←MCU) | Analog In | ADCINC3
(!) | P67 | GPIO | (!) | Analog In | ADCINB3
P111 | Analog In | Analog In | ADCINA3
SPIA CLK | P60 | SPI CLK | Analog In | ADCINC2
(!) | P22 | GPIO | (!) | Analog In | ADCINB2
I2CA SCL | P105 | I2C SCL | Analog Out | ADCINA2
I2CA SDA | P104 | SDA | Analog Out | ADCINA0
+3.3V | +3.3V | +5V | +5V
P95 | Analog In | GND | GND
SCIC RX | P139 | UART RX (→MCU) | Analog In | ADCIN15
SCIC TX | P56 | TX (←MCU) | Analog In | ADCINC5
(!) | P97 | GPIO | (!) | Analog In | ADCINB5
P94 | Analog In | Analog In | ADCINA5
SPIB CLK | P65 | SPI CLK | Analog In | ADCINC4
(!) | P52 | GPIO | (!) | Analog In | ADCINB4
I2CB SCL | P41 | I2C SCL | Analog Out | ADCINA4
I2CB SDA | P40 | SDA | Analog Out | ADCINA1
P0 | (!) | PWM1A | PWM | GPIO | (!) | GND | GND
P1 | (!) | PWM1B | PWM | GPIO | (!) | PWM | GPIO | (!) | P61 | SPIA CS
P2 | (!) | PWM2A | PWM | GPIO | (!) | SPI CS | GPIO | (!) | P123 | SD1 CLK1
P3 | (!) | PWM2B | PWM | GPIO | (!) | GPIO** | P122 | SD1 D1
P4 | (!) | PWM3A | Timer | GPIO | (!) | RST | RST
P5 | (!) | PWM3B | Timer | GPIO | (!) | SPI MOSI | GPIO | (!) | P58 | SPIA SIMO
P24 | (!) | OPXBAR1 | GPIO | (!) | MISO | GPIO | (!) | P59 | SPIA SOMI
P16 | (!) | OPXBAR7 | GPIO | (!) | SPI CS | GPIO | (!) | P124 | SD1 D2
DAC1 | GPIO | (!) | SPI CS | GPIO | (!) | P125 | SD1 CLK2
DAC2 | GPIO | (!) | GPIO | (!) | P29 | OPXBAR6
P6 | (!) | PWM4A | PWM | GPIO | (!) | GND | GND
P7 | (!) | PWM4B | PWM | GPIO | (!) | PWM | GPIO | (!) | P66 | SPIB CS
P8 | (!) | PWM5A | PWM | GPIO | (!) | SPI CS | GPIO | (!) | P131 | SD2 CLK1
P9 | (!) | PWM5B | PWM | GPIO | (!) | GPIO** | P130 | SD2 D1
P10 | (!) | PWM6A | Timer | GPIO | (!) | RST | RST
P11 | (!) | PWM6B | Timer | GPIO | (!) | SPI MOSI | P63 | SPIB SIMO
P14 | (!) | OPXBAR3 | GPIO | (!) | MISO | P64 | SPIB SOMI
P15 | (!) | OPXBAR4 | GPIO | (!) | SPI CS | GPIO | (!) | P26 | SD2 D2
DAC3 | GPIO | (!) | SPI CS | GPIO | (!) | P27 | SD2 CLK2
DAC4 | GPIO | (!) | GPIO | (!) | P25 | OPXBAR2

## Outline

## 1. Overview

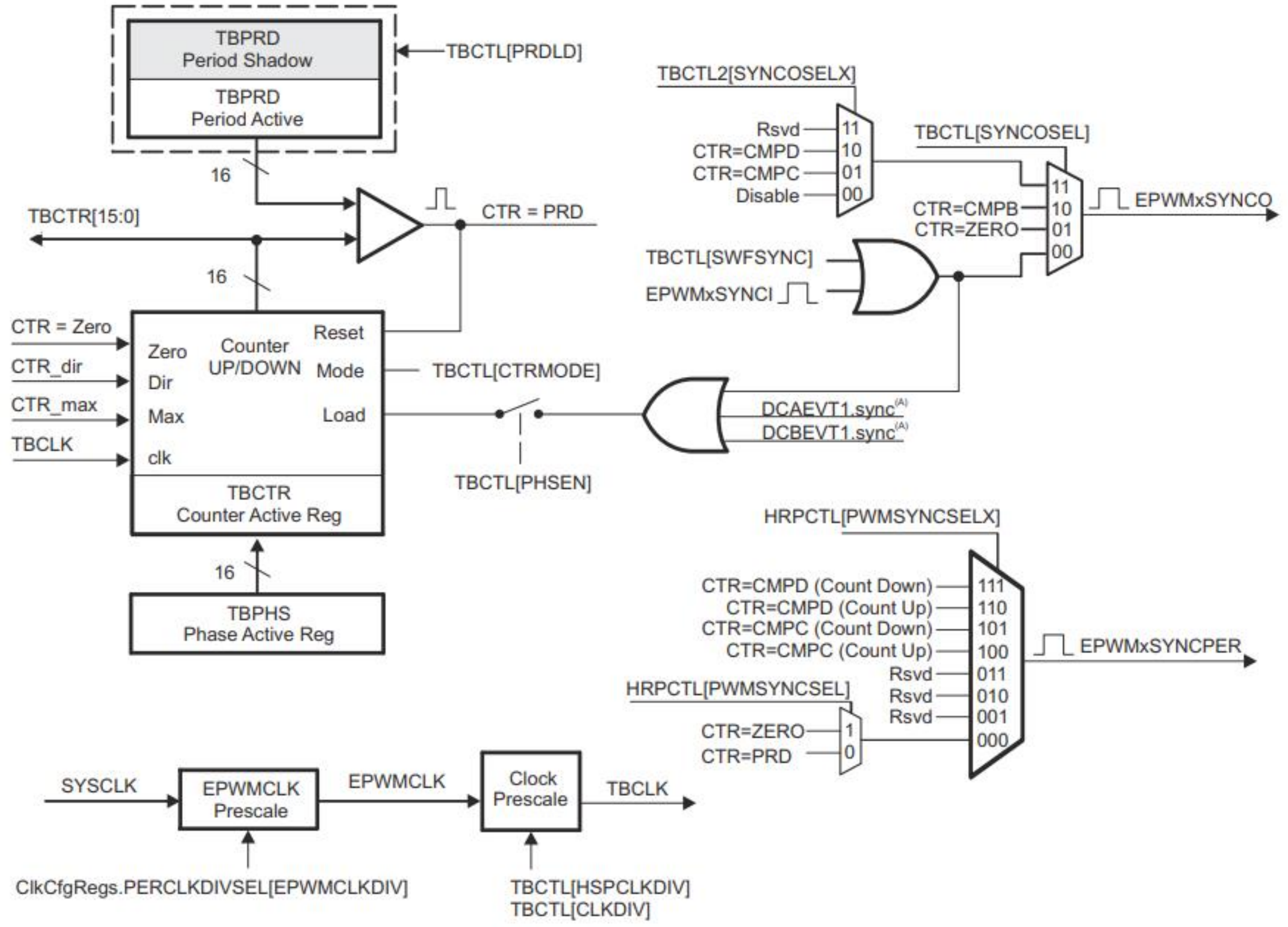


## 2. ePWM


EPWMxSYNCI
EPWMxSYNCO
Digital Compare Signals
Time-Base (TB)
CTR = PRD
CTR = 0
CTR_Dir
Action Qualifier (AQ)
ePWMxA
ePWMxB
T1
T2
Counter Compare (CC)
CTR = CMPA
CTR = CMPB
CTR = CMPC
CTR = CMPD
Time Base Signals
Counter Compare Signals
Digital Compare Signals
Event Trigger and Interrupt (ET)
EPWMxINT
EPWMxSOCA
EPWMxSOCB
PIE
ADC
Dead Band (DB)
PWM-chopper (PC)
Trip Zone (TZ)
ePWMxA
ePWMxB
GPIO MUX
GPIO MUX
TZ1 to TZ3
EMUSTOP
CLOCKFAIL
EQEPxERR
Input X-BAR
CPU
SYSCTRL
EQEPx
CTR = PRD
CTR = 0
EPWMxTZINT
PIE
Digital Compare Signals
Digital Compare (DC)
TZ1 to TZ3
ECCDBLERR
PIEVECTERR
CMPSSx
28x RAM/ Flash ECC
EPWM X-BAR

# 2. ePWM

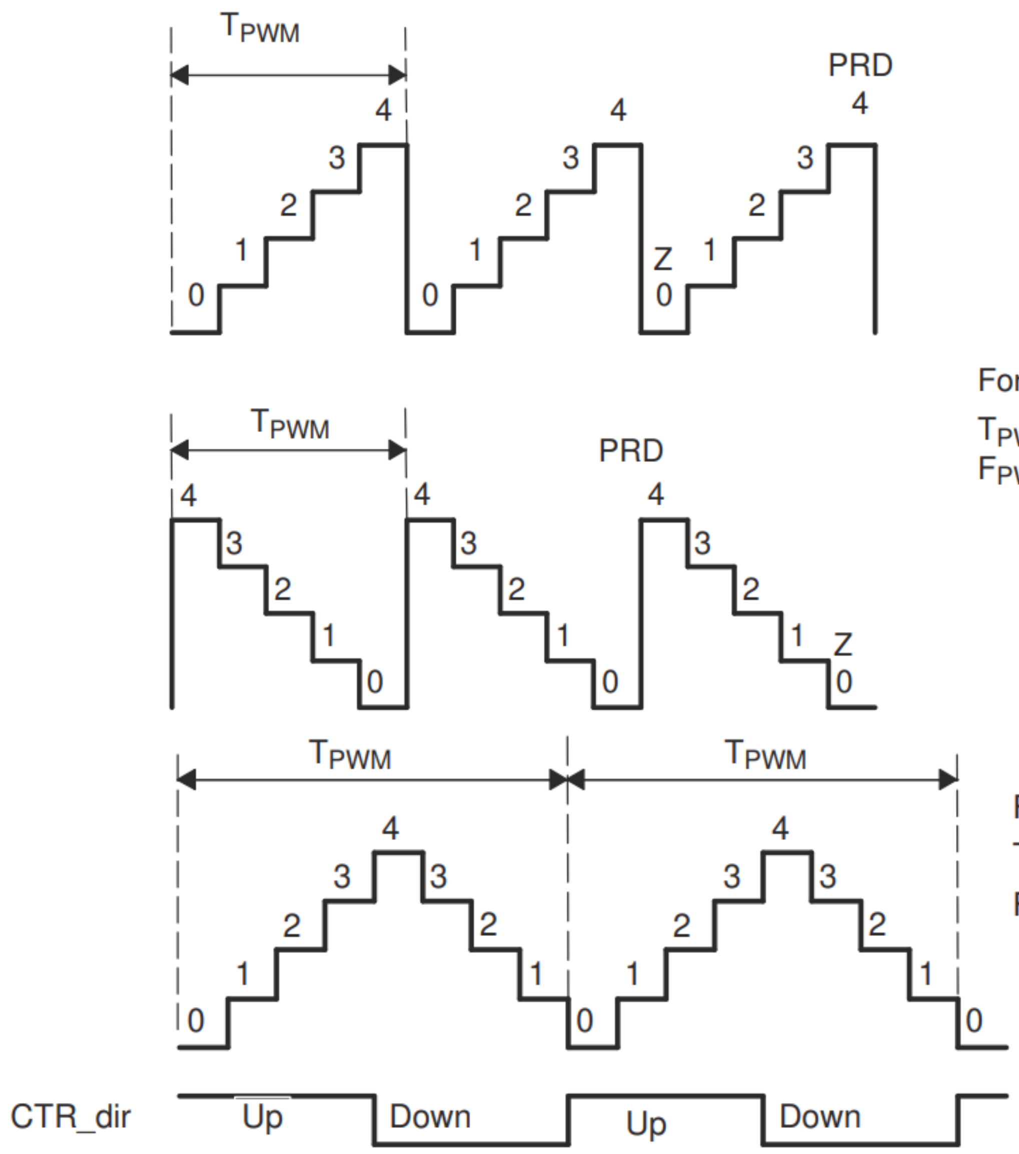
## 2.1 TB block



A. These signals are generated by the digital compare (DC) submodule.

# 2. ePWM

## 2.1 TB block



For Up Count and Down Count

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

$F_{PWM} = 1/(T_{PWM})$

For Up and Down Count

$T_{PWM} = 2 \times TBPRD \times T_{TBCLK}$

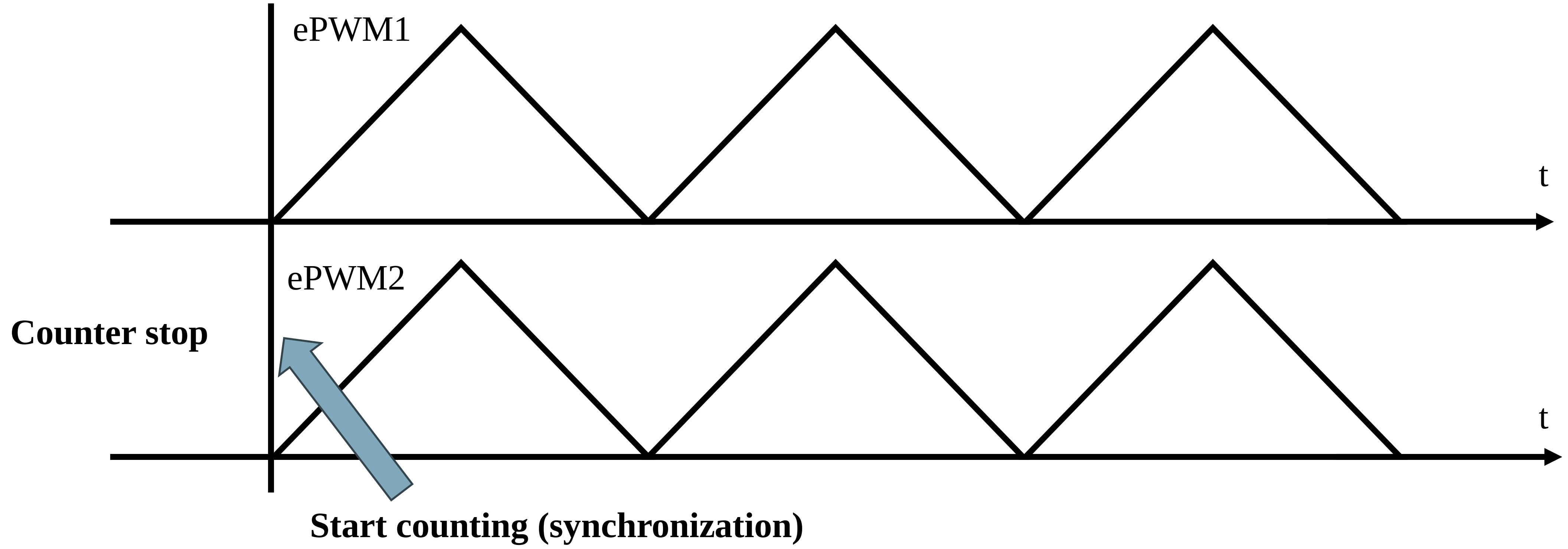
$F_{PWM} = 1/(T_{PWM})$

# 2. ePWM

## 2.1 TB block

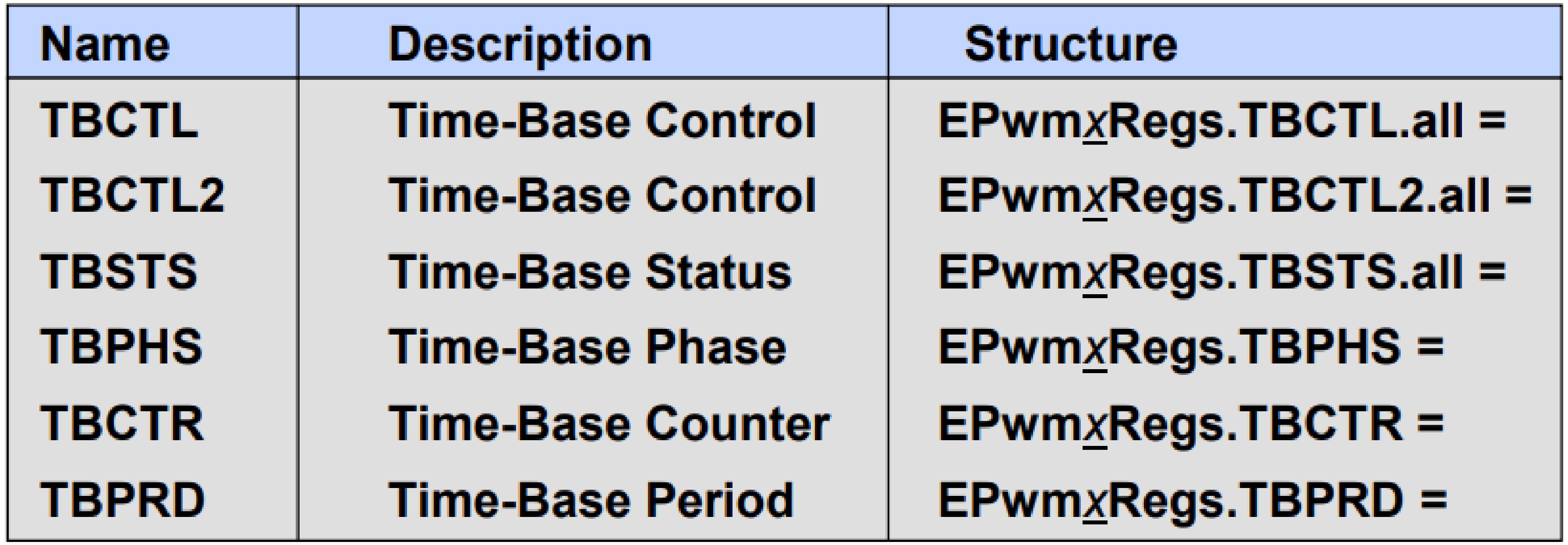
The proper procedure for enabling ePWM clocks is as follows:

1. Enable ePWM module clocks in the PCLKCRx register
2. Set TBCLKSYNC= 0
3. Configure ePWM modules
4. Set TBCLKSYNC= 1



# 2. ePWM

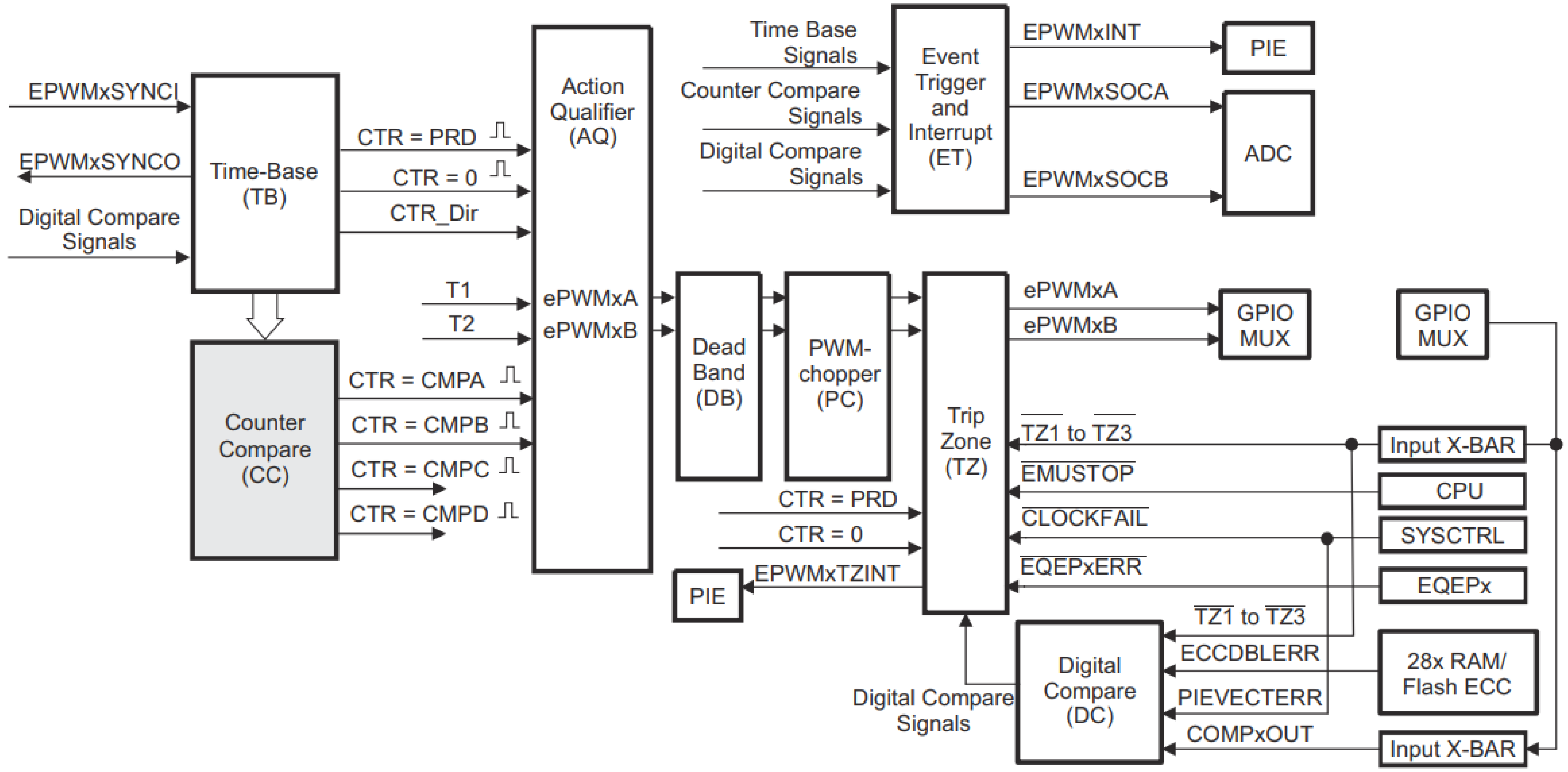
## 2.1 TB block



| Name | Description | Structure |
|---|---|---|
| TBCTL | Time-Base Control | EPwmxRegs.TBCTL.all = |
| TBCTL2 | Time-Base Control | EPwmxRegs.TBCTL2.all = |
| TBSTS | Time-Base Status | EPwmxRegs.TBSTS.all = |
| TBPHS | Time-Base Phase | EPwmxRegs.TBPHS = |
| TBCTR | Time-Base Counter | EPwmxRegs.TBCTR = |
| TBPRD | Time-Base Period | EPwmxRegs.TBPRD = |

# 2. ePWM

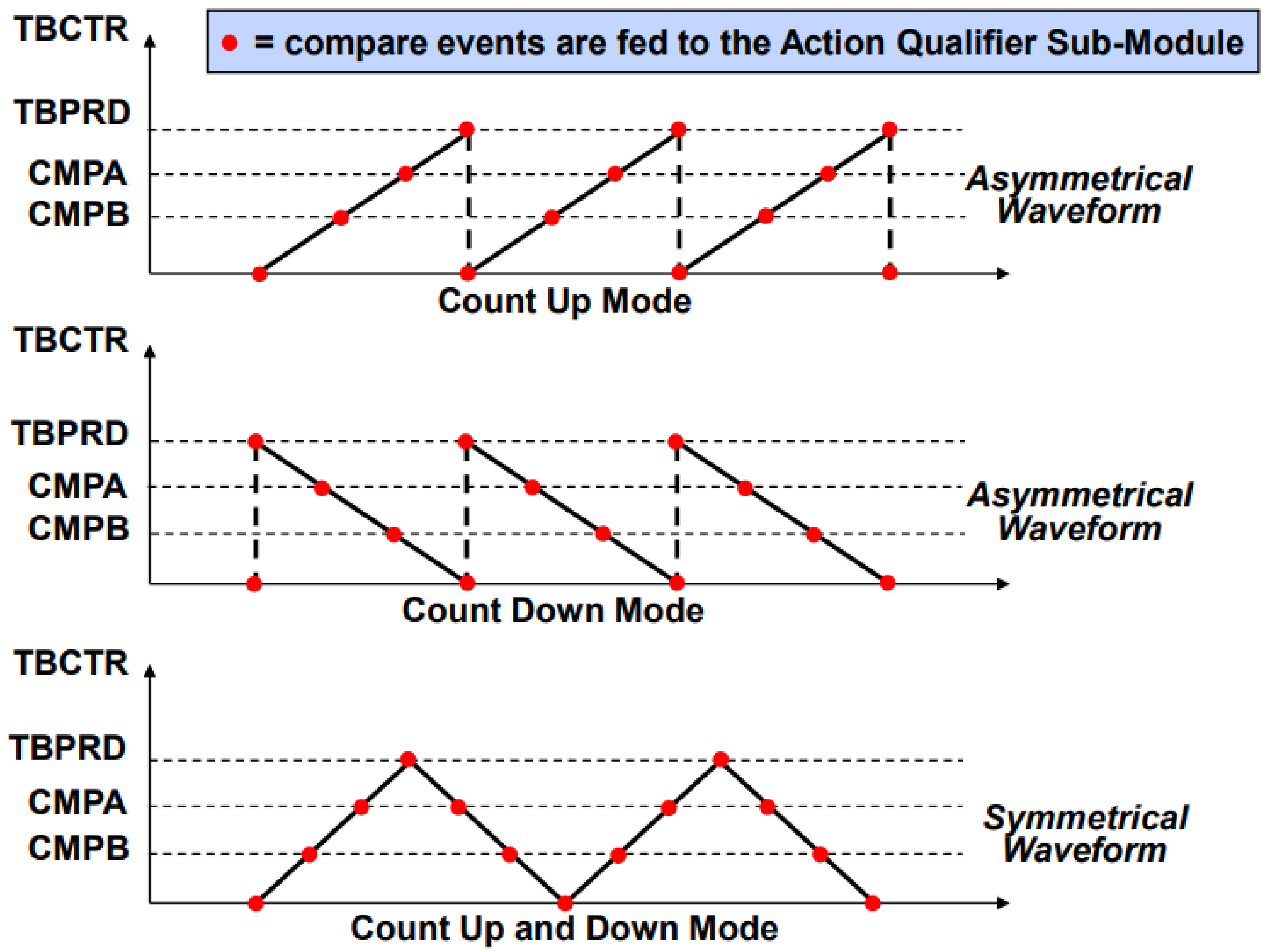
## 2.2 CC block



## 2. ePWM

### 2.2 CC block

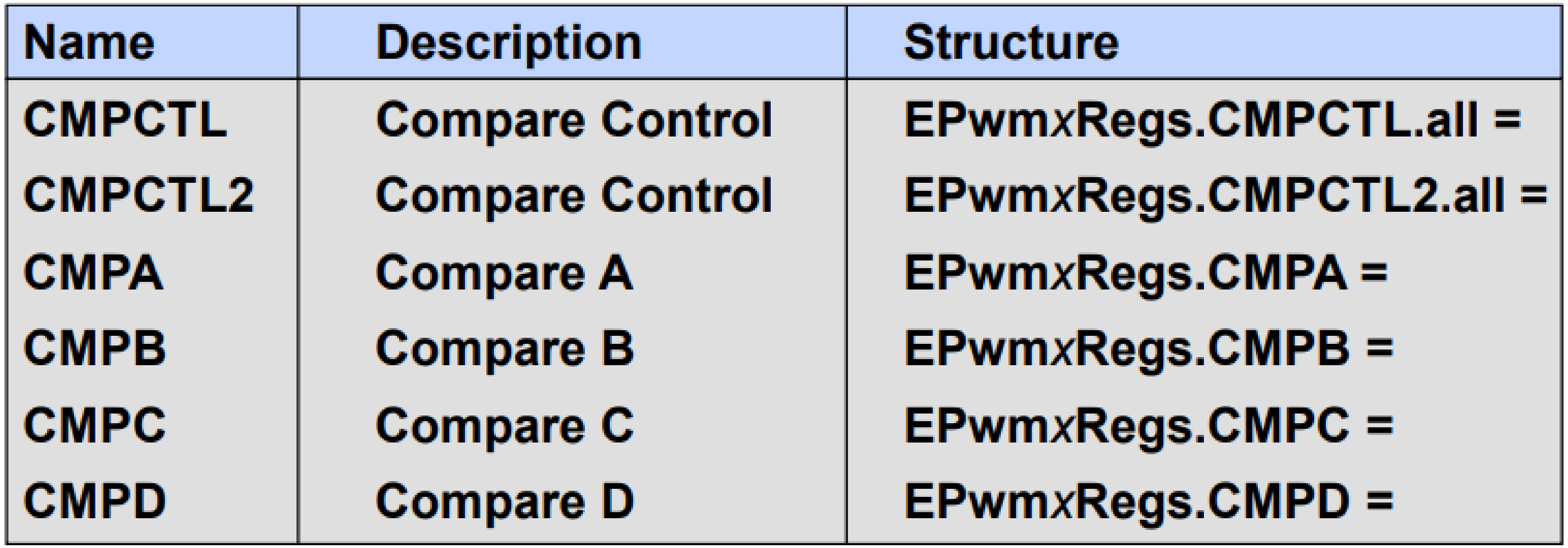
### ePWM Compare Event Waveforms



*CMPC and CMPD available for use as event triggers*

## 2. ePWM

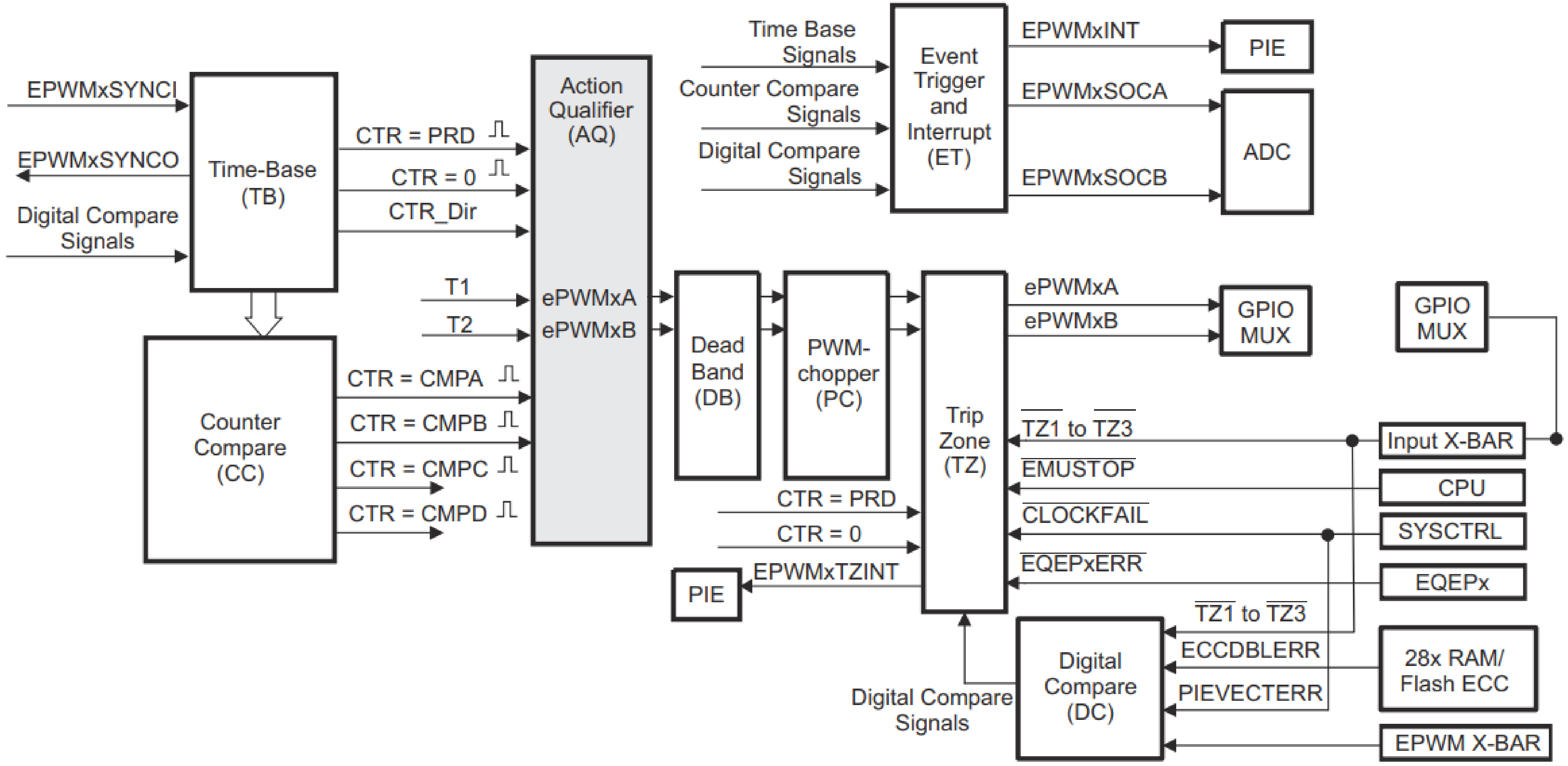
### 2.2 CC block



| Name | Description | Structure |
|---|---|---|
| CMPCTL | Compare Control | EPwm*x*Regs.CMPCTL.all = |
| CMPCTL2 | Compare Control | EPwm*x*Regs.CMPCTL2.all = |
| CMPA | Compare A | EPwm*x*Regs.CMPA = |
| CMPB | Compare B | EPwm*x*Regs.CMPB = |
| CMPC | Compare C | EPwm*x*Regs.CMPC = |
| CMPD | Compare D | EPwm*x*Regs.CMPD = |

# 2. ePWM

## 2.3 AQ block

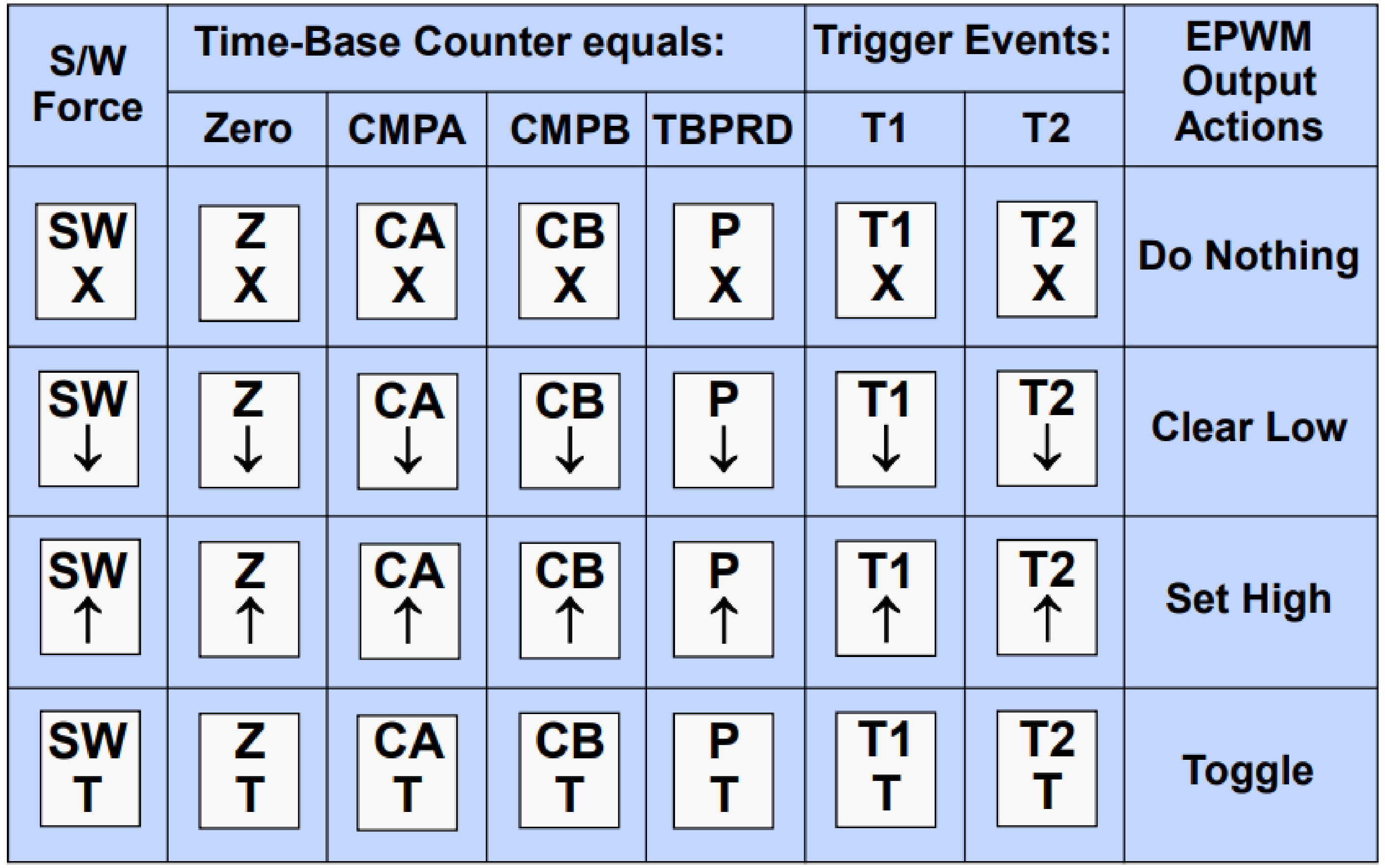


# 2. ePWM

## 2.3 AQ block

### ePWM Action Qualifier Actions

for EPWMA and EPWMB



| S/W Force | Time-Base Counter equals: | | | | Trigger Events: | | EPWM Output Actions |
|---|---|---|---|---|---|---|---|
| | Zero | CMPA | CMPB | TBPRD | T1 | T2 | |
| SW X | Z X | CA X | CB X | P X | T1 X | T2 X | Do Nothing |
| SW ↓ | Z ↓ | CA ↓ | CB ↓ | P ↓ | T1 ↓ | T2 ↓ | Clear Low |
| SW ↑ | Z ↑ | CA ↑ | CB ↑ | P ↑ | T1 ↑ | T2 ↑ | Set High |
| SW T | Z T | CA T | CB T | P T | T1 T | T2 T | Toggle |

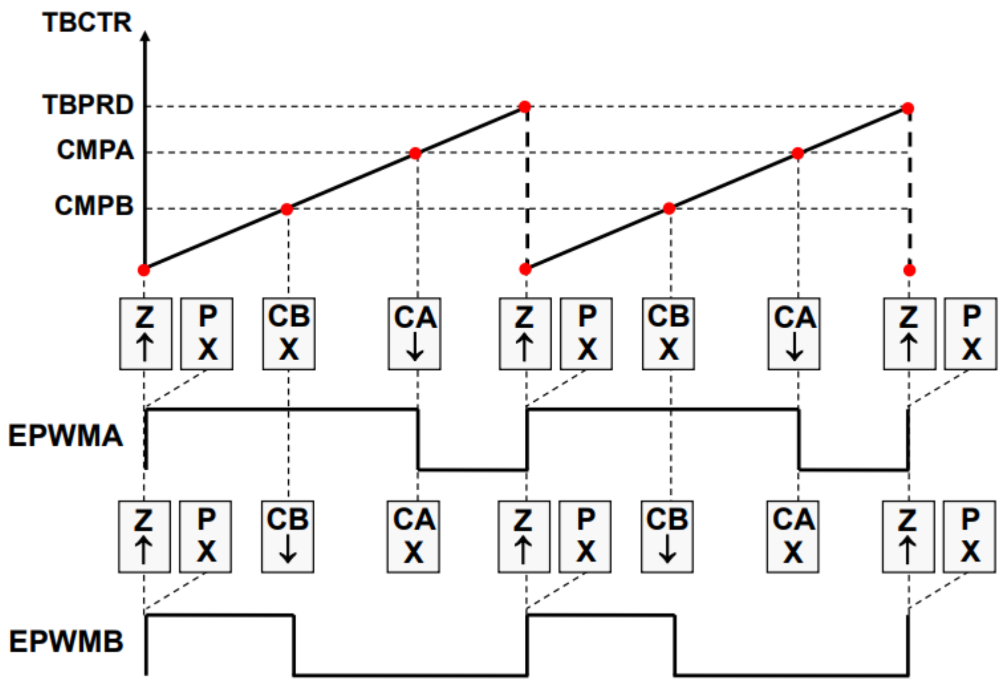
*Tx Event Sources = DCAEVT1, DCAEVT2, DCBEVT1, DCBEVT2, TZ1, TZ2, TZ3, EPWMxSYNCIN*

# 2. ePWM

## 2.3 AQ block

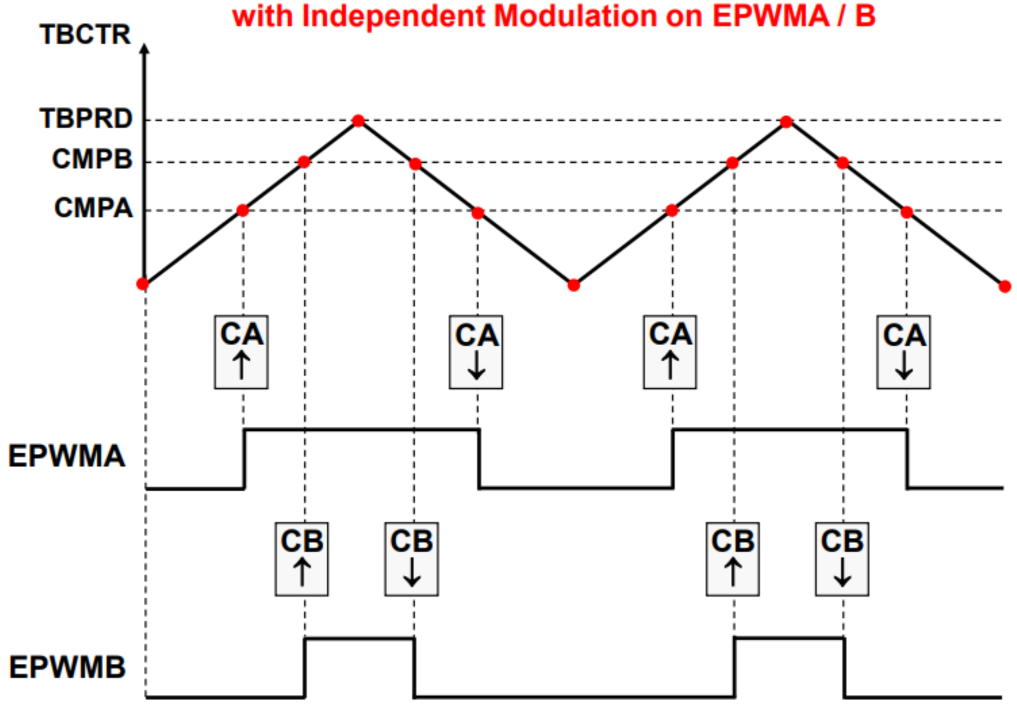
**ePWM Count Up Asymmetric Waveform**

with Independent Modulation on EPWMA / B



**ePWM Count Up-Down Symmetric Waveform**

with Independent Modulation on EPWMA / B



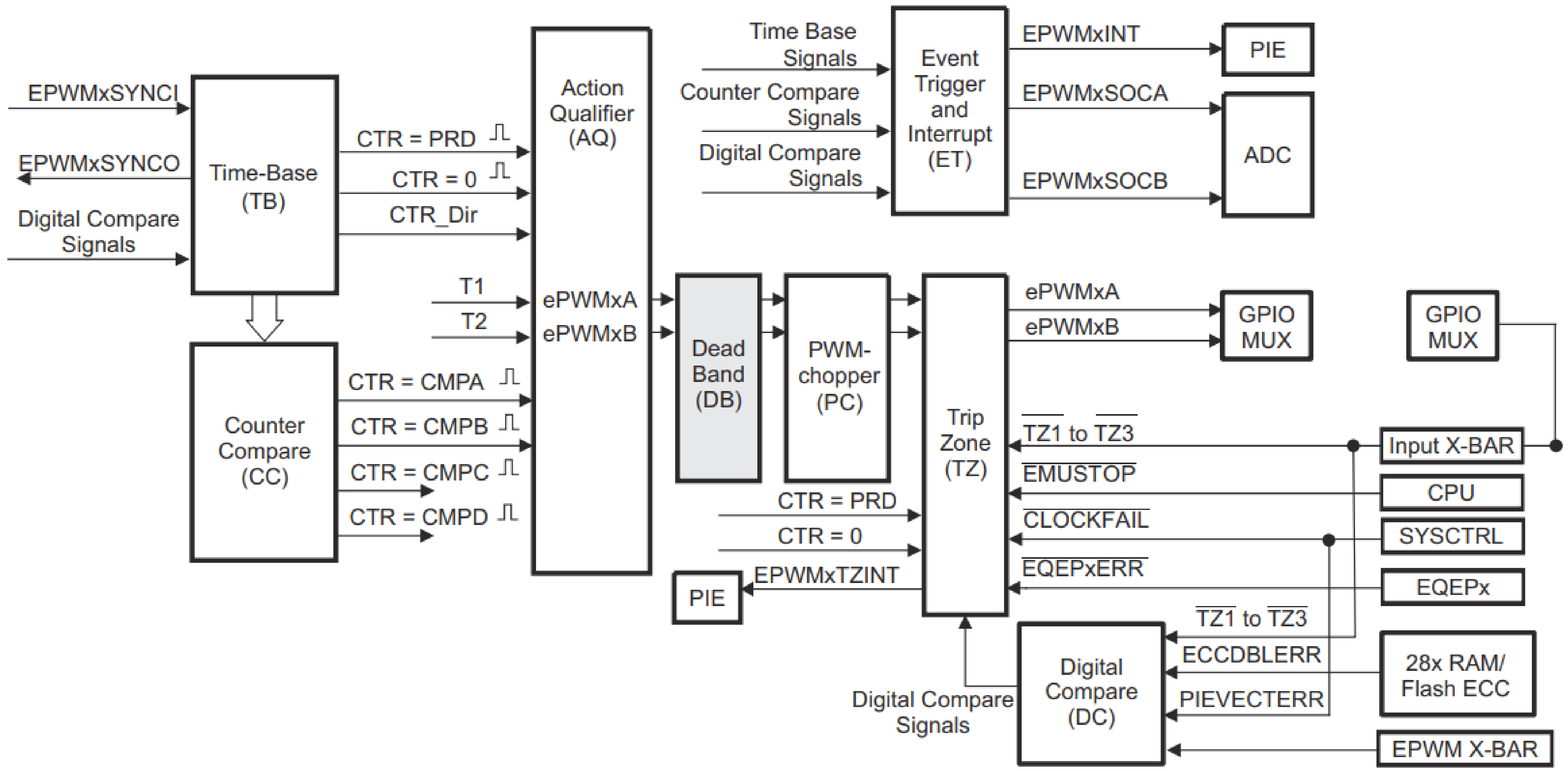
# 2. ePWM

## 2.3 AQ block

| Name | Description | Structure |
|---|---|---|
| AQCTL | AQ Control Register | EPwm*x*Regs.AQCTL.all = |
| AQCTLA | AQ Control Output A | EPwm*x*Regs.AQCTLA.all = |
| AQCTLA2 | AQ Control Output A | EPwm*x*Regs.AQCTLA2.all = |
| AQCTLB | AQ Control Output B | EPwm*x*Regs.AQCTLB.all = |
| AQCTLB2 | AQ Control Output B | EPwm*x*Regs.AQCTLB2.all = |
| AQTSRCSEL | AQ T Source Select | EPwm*x*Regs.AQTSRCSEL = |
| AQSFRC | AQ S/W Force | EPwm*x*Regs.AQSFRC.all = |
| AQCSFRC | AQ Cont. S/W Force | EPwm*x*Regs.AQCSFRC.all = |

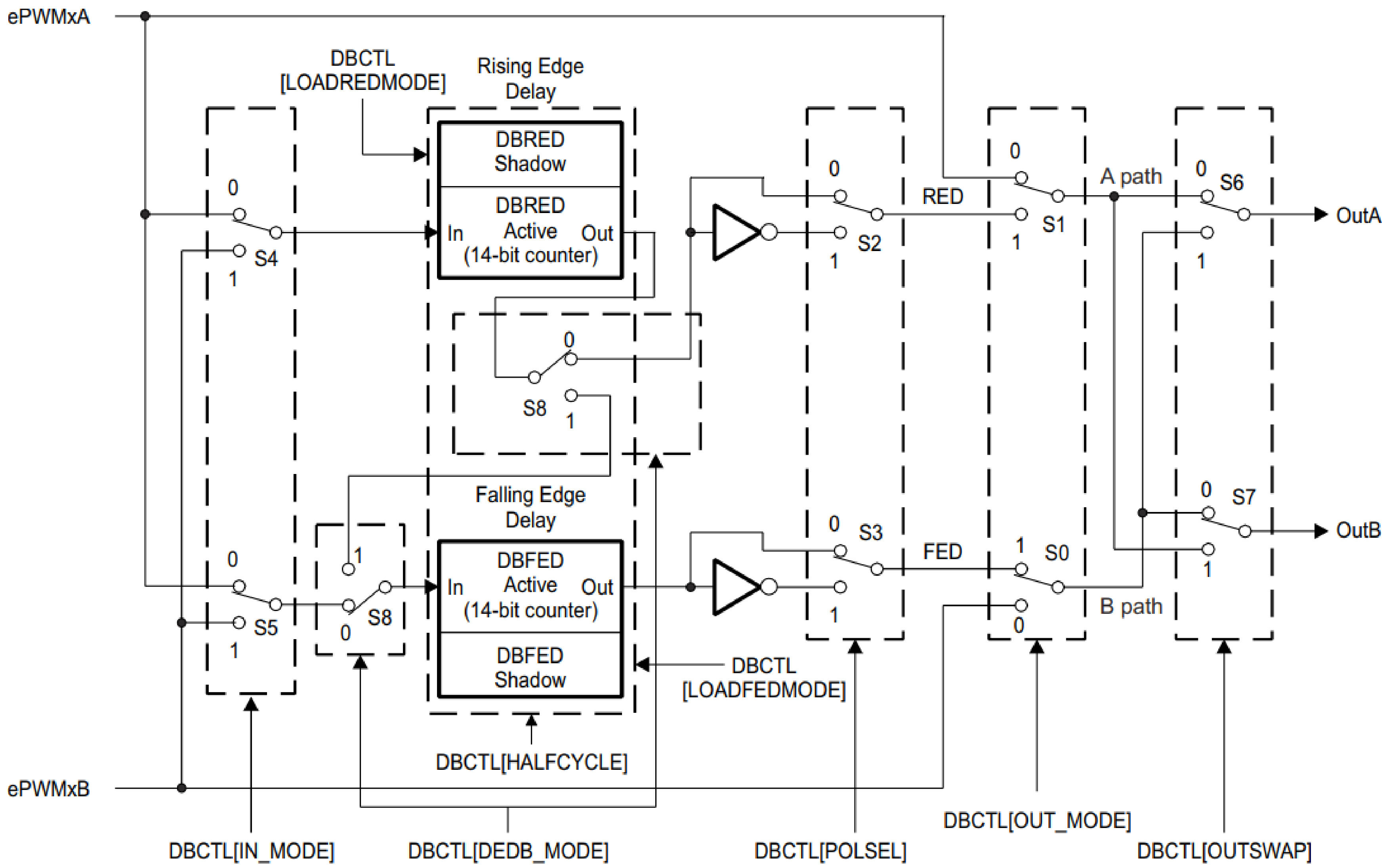
# 2. ePWM

## 2.4 DB block



## 2. ePWM

### 2.4 DB block



## 2. ePWM

### 2.4 DB block

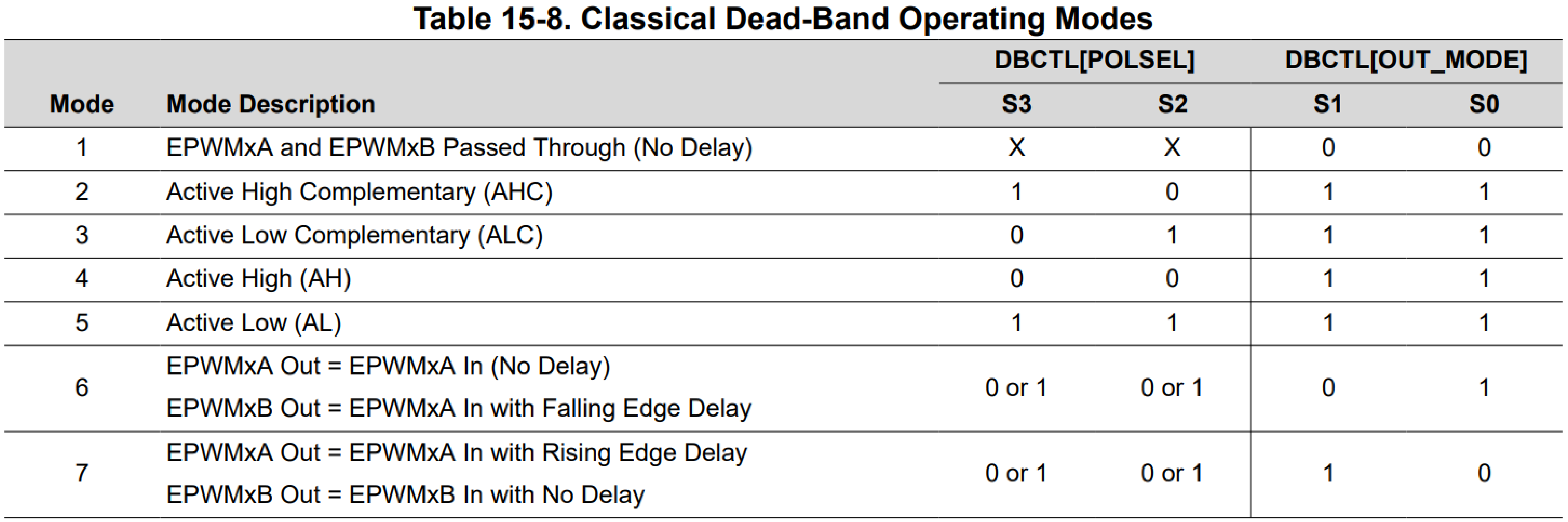


Table 15-8. Classical Dead-Band Operating Modes

| | | DBCTL[POLSEL] | | DBCTL[OUT_MODE] | |
|---|---|---|---|---|---|
| Mode | Mode Description | S3 | S2 | S1 | S0 |
| 1 | EPWMxA and EPWMxB Passed Through (No Delay) | X | X | 0 | 0 |
| 2 | Active High Complementary (AHC) | 1 | 0 | 1 | 1 |
| 3 | Active Low Complementary (ALC) | 0 | 1 | 1 | 1 |
| 4 | Active High (AH) | 0 | 0 | 1 | 1 |
| 5 | Active Low (AL) | 1 | 1 | 1 | 1 |
| 6 | EPWMxA Out = EPWMxA In (No Delay)<br>EPWMxB Out = EPWMxA In with Falling Edge Delay | 0 or 1 | 0 or 1 | 0 | 1 |
| 7 | EPWMxA Out = EPWMxA In with Rising Edge Delay<br>EPWMxB Out = EPWMxB In with No Delay | 0 or 1 | 0 or 1 | 1 | 0 |

# 2. ePWM

## 2.4 DB block

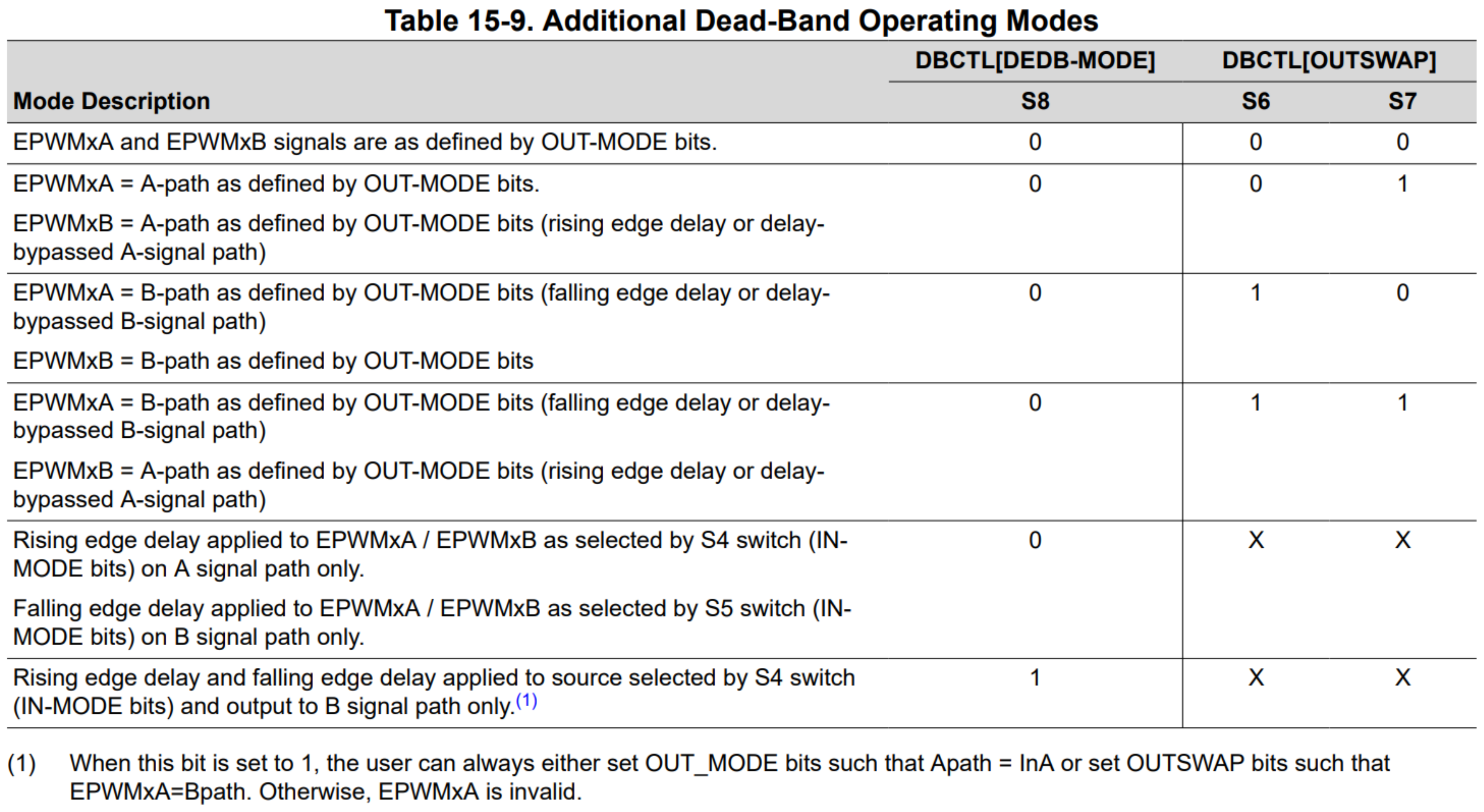


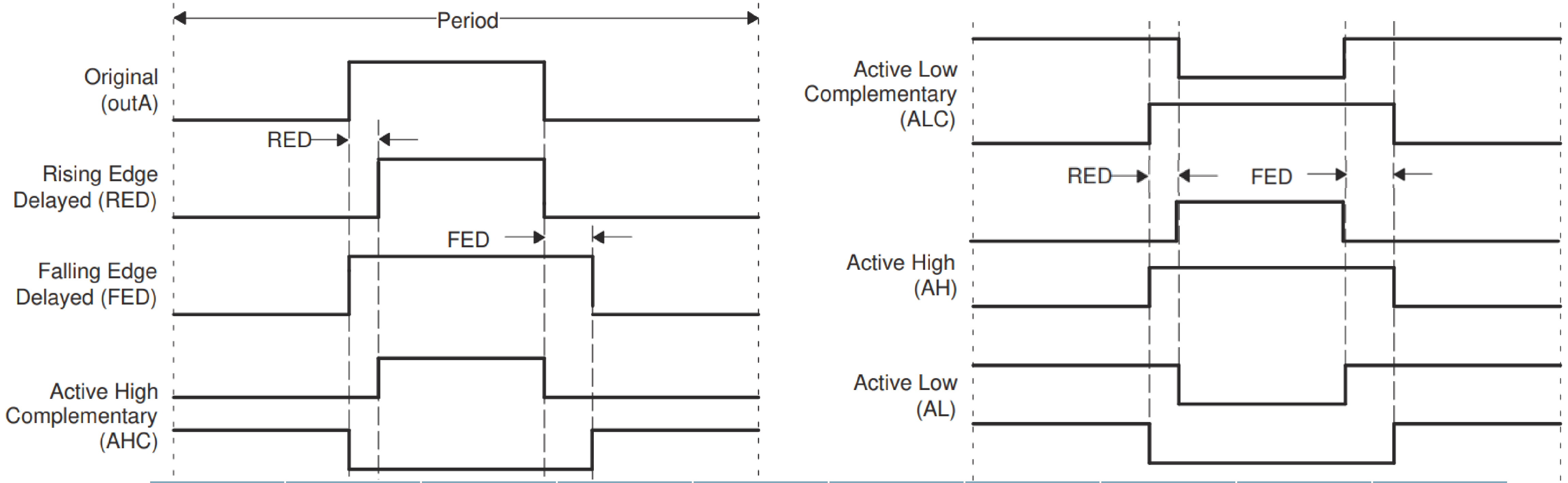
Table 15-9. Additional Dead-Band Operating Modes

| | DBCTL[DEDB-MODE] | DBCTL[OUTSWAP] | |
|---|---|---|---|
| Mode Description | S8 | S6 | S7 |
| EPWMxA and EPWMxB signals are as defined by OUT-MODE bits. | 0 | 0 | 0 |
| EPWMxA = A-path as defined by OUT-MODE bits.<br>EPWMxB = A-path as defined by OUT-MODE bits (rising edge delay or delay-bypassed A-signal path) | 0 | 0 | 1 |
| EPWMxA = B-path as defined by OUT-MODE bits (falling edge delay or delay-bypassed B-signal path)<br>EPWMxB = B-path as defined by OUT-MODE bits | 0 | 1 | 0 |
| EPWMxA = B-path as defined by OUT-MODE bits (falling edge delay or delay-bypassed B-signal path)<br>EPWMxB = A-path as defined by OUT-MODE bits (rising edge delay or delay-bypassed A-signal path) | 0 | 1 | 1 |
| Rising edge delay applied to EPWMxA / EPWMxB as selected by S4 switch (IN-MODE bits) on A signal path only.<br>Falling edge delay applied to EPWMxA / EPWMxB as selected by S5 switch (IN-MODE bits) on B signal path only. | 0 | X | X |
| Rising edge delay and falling edge delay applied to source selected by S4 switch (IN-MODE bits) and output to B signal path only.(1) | 1 | X | X |

(1) When this bit is set to 1, the user can always either set OUT_MODE bits such that Apath = InA or set OUTSWAP bits such that EPWMxA=Bpath. Otherwise, EPWMxA is invalid.

# 2. ePWM

## 2.4 DB block

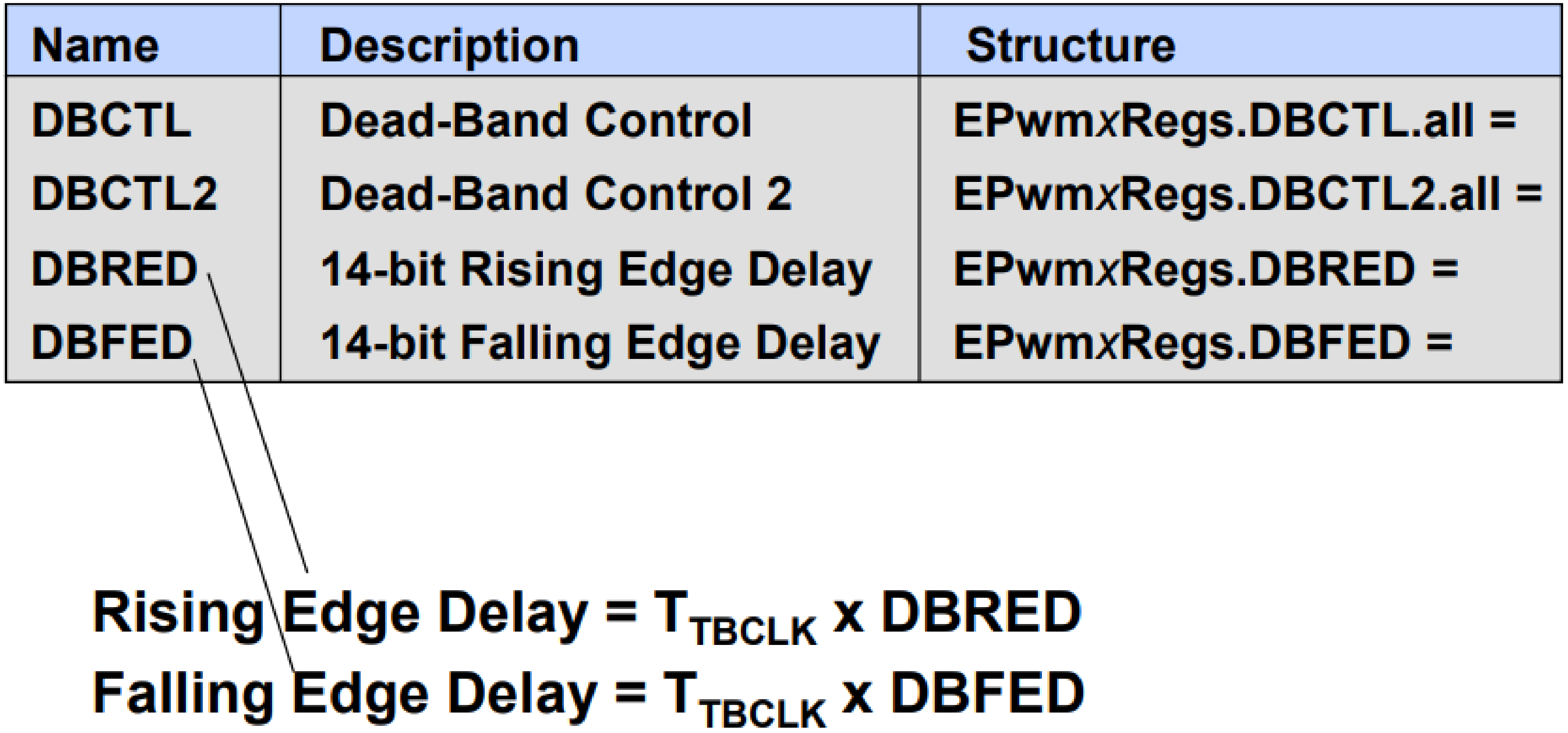


Period
Original (outA)
RED
Rising Edge Delayed (RED)
FED
Falling Edge Delayed (FED)
Active High Complementary (AHC)
Active Low Complementary (ALC)
RED
FED
Active High (AH)
Active Low (AL)

| MODE | S0 | S1 | S2 | S3 | S4 | S5 | S6 | S7 | S8 |
|---|---|---|---|---|---|---|---|---|---|
| AHC | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| ALC | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| AH | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| AL | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |

## 2. ePWM

### 2.4 DB block

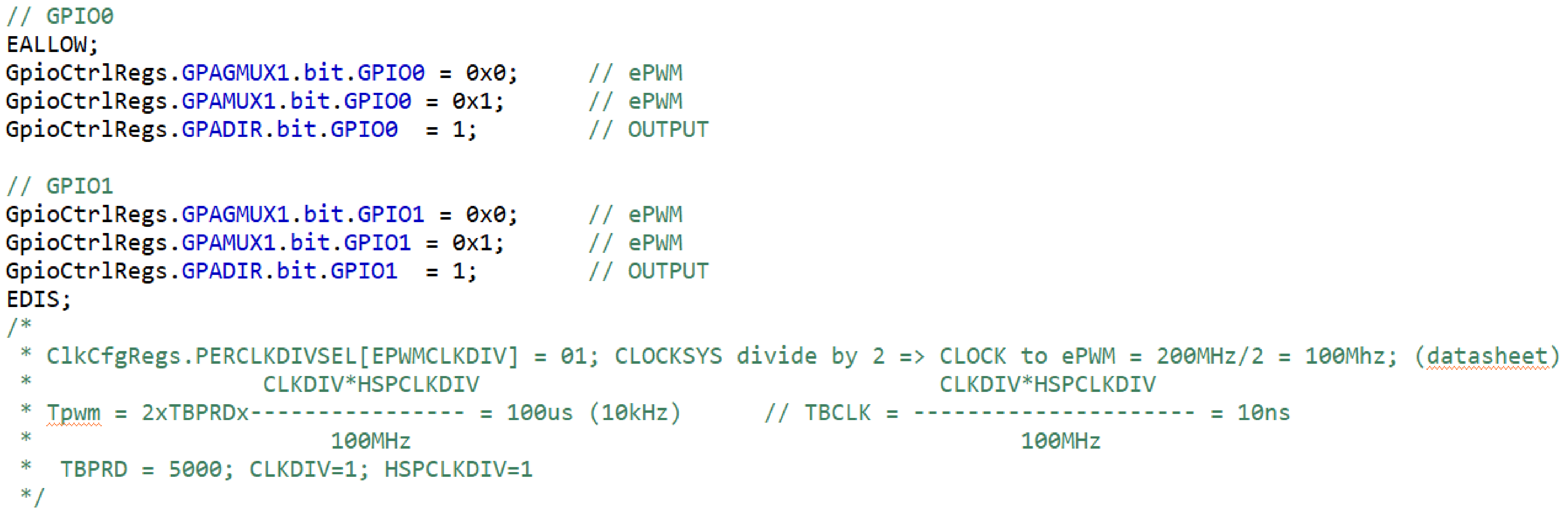


| Name | Description | Structure |
|---|---|---|
| DBCTL | Dead-Band Control | EPwm*x*Regs.DBCTL.all = |
| DBCTL2 | Dead-Band Control 2 | EPwm*x*Regs.DBCTL2.all = |
| DBRED | 14-bit Rising Edge Delay | EPwm*x*Regs.DBRED = |
| DBFED | 14-bit Falling Edge Delay | EPwm*x*Regs.DBFED = |

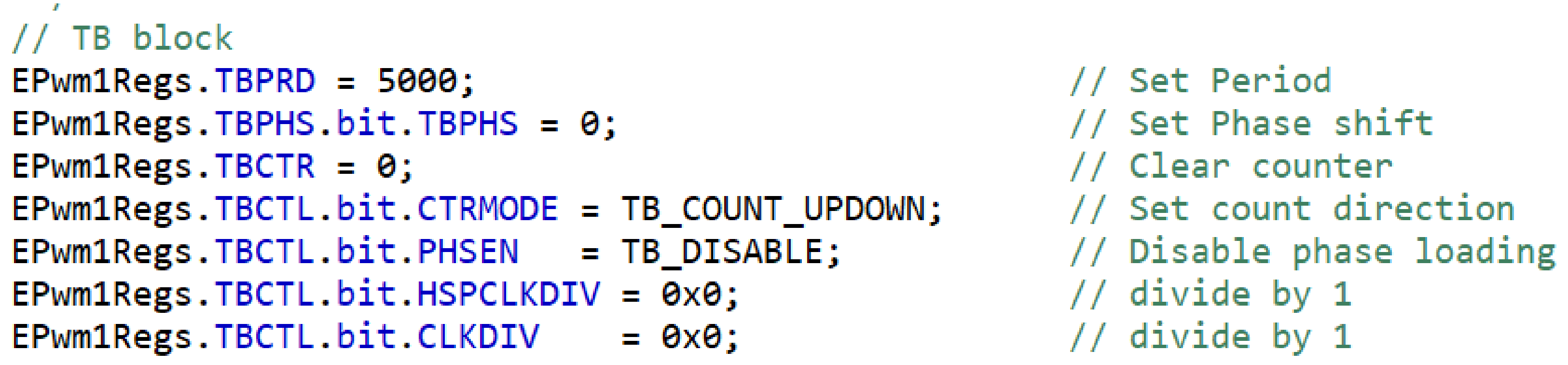
$$\text{Rising Edge Delay} = T_{TBCLK} \times \text{DBRED}$$

$$\text{Falling Edge Delay} = T_{TBCLK} \times \text{DBFED}$$

# 3. ePWM example

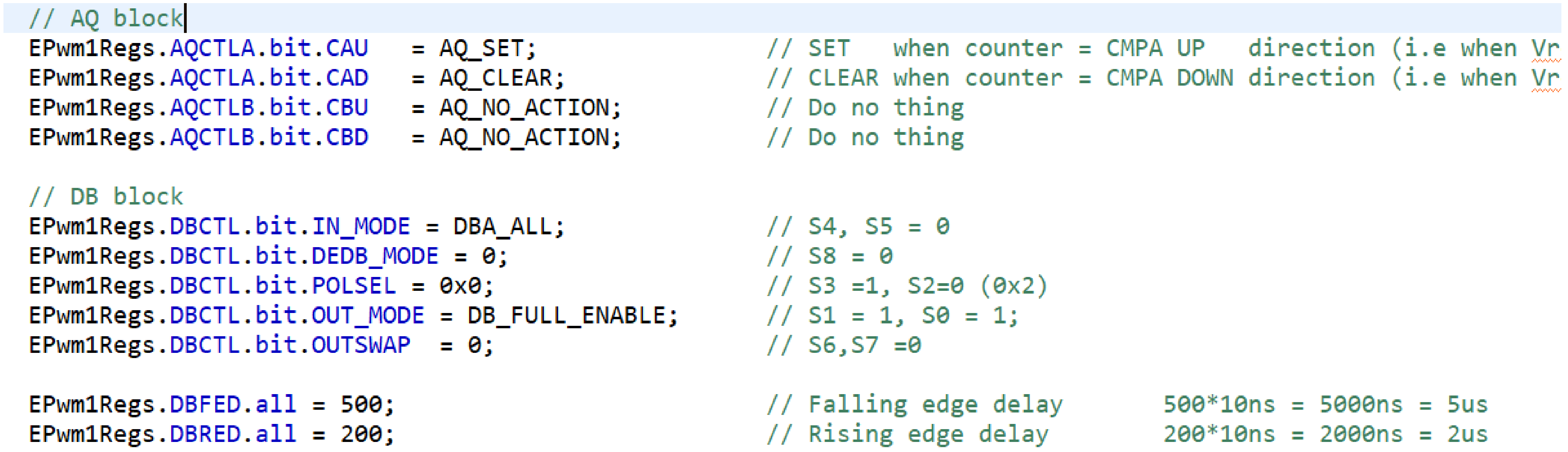


```
// GPIO0
EALLOW;
GpioCtrlRegs.GPAGMUX1.bit.GPIO0 = 0x0;     // ePWM
GpioCtrlRegs.GPAMUX1.bit.GPIO0 = 0x1;      // ePWM
GpioCtrlRegs.GPADIR.bit.GPIO0  = 1;        // OUTPUT

// GPIO1
GpioCtrlRegs.GPAGMUX1.bit.GPIO1 = 0x0;     // ePWM
GpioCtrlRegs.GPAMUX1.bit.GPIO1 = 0x1;      // ePWM
GpioCtrlRegs.GPADIR.bit.GPIO1  = 1;        // OUTPUT
EDIS;
/*
 * ClkCfgRegs.PERCLKDIVSEL[EPWMCLKDIV] = 01; CLOCKSYS divide by 2 => CLOCK to ePWM = 200MHz/2 = 100Mhz; (datasheet)
 *                CLKDIV*HSPCLKDIV                                  CLKDIV*HSPCLKDIV
 * Tpwm = 2xTBPRDx---------------- = 100us (10kHz)      // TBCLK = -------------------- = 10ns
 *                     100MHz                                             100MHz
 *  TBPRD = 5000; CLKDIV=1; HSPCLKDIV=1
 */
```



```
// TB block
EPwm1Regs.TBPRD = 5000;                         // Set Period
EPwm1Regs.TBPHS.bit.TBPHS = 0;                  // Set Phase shift
EPwm1Regs.TBCTR = 0;                            // Clear counter
EPwm1Regs.TBCTL.bit.CTRMODE = TB_COUNT_UPDOWN;  // Set count direction
EPwm1Regs.TBCTL.bit.PHSEN    = TB_DISABLE;      // Disable phase loading
EPwm1Regs.TBCTL.bit.HSPCLKDIV = 0x0;            // divide by 1
EPwm1Regs.TBCTL.bit.CLKDIV    = 0x0;            // divide by 1
```

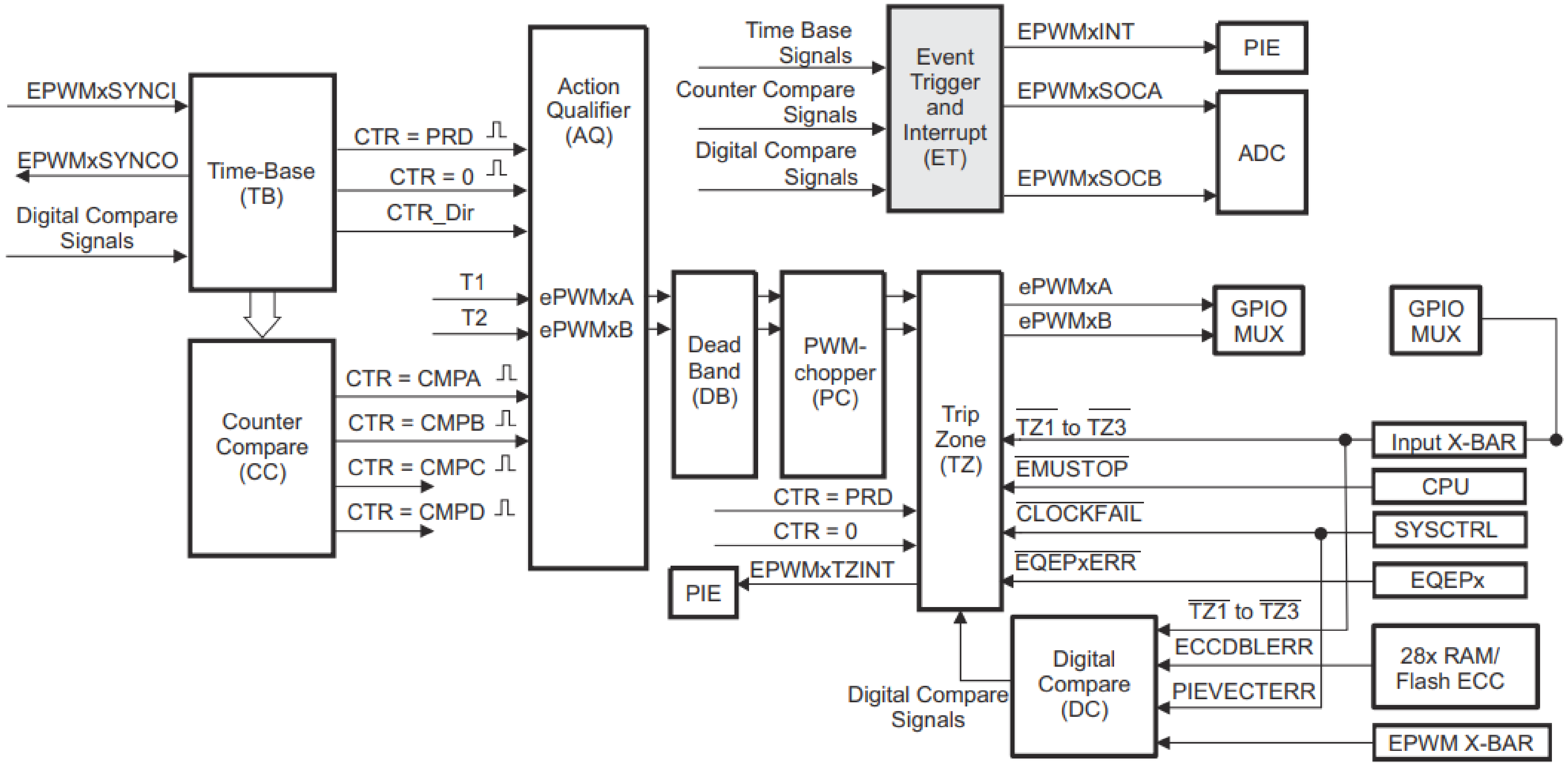
## 3. ePWM example



```
// AQ block
EPwm1Regs.AQCTLA.bit.CAU    = AQ_SET;                  // SET   when counter = CMPA UP   direction (i.e when Vr
EPwm1Regs.AQCTLA.bit.CAD    = AQ_CLEAR;                // CLEAR when counter = CMPA DOWN direction (i.e when Vr
EPwm1Regs.AQCTLB.bit.CBU    = AQ_NO_ACTION;            // Do no thing
EPwm1Regs.AQCTLB.bit.CBD    = AQ_NO_ACTION;            // Do no thing

// DB block
EPwm1Regs.DBCTL.bit.IN_MODE = DBA_ALL;                 // S4, S5 = 0
EPwm1Regs.DBCTL.bit.DEDB_MODE = 0;                     // S8 = 0
EPwm1Regs.DBCTL.bit.POLSEL = 0x0;                      // S3 =1, S2=0 (0x2)
EPwm1Regs.DBCTL.bit.OUT_MODE = DB_FULL_ENABLE;         // S1 = 1, S0 = 1;
EPwm1Regs.DBCTL.bit.OUTSWAP  = 0;                      // S6,S7 =0

EPwm1Regs.DBFED.all = 500;                             // Falling edge delay      500*10ns = 5000ns = 5us
EPwm1Regs.DBRED.all = 200;                             // Rising edge delay       200*10ns = 2000ns = 2us
```

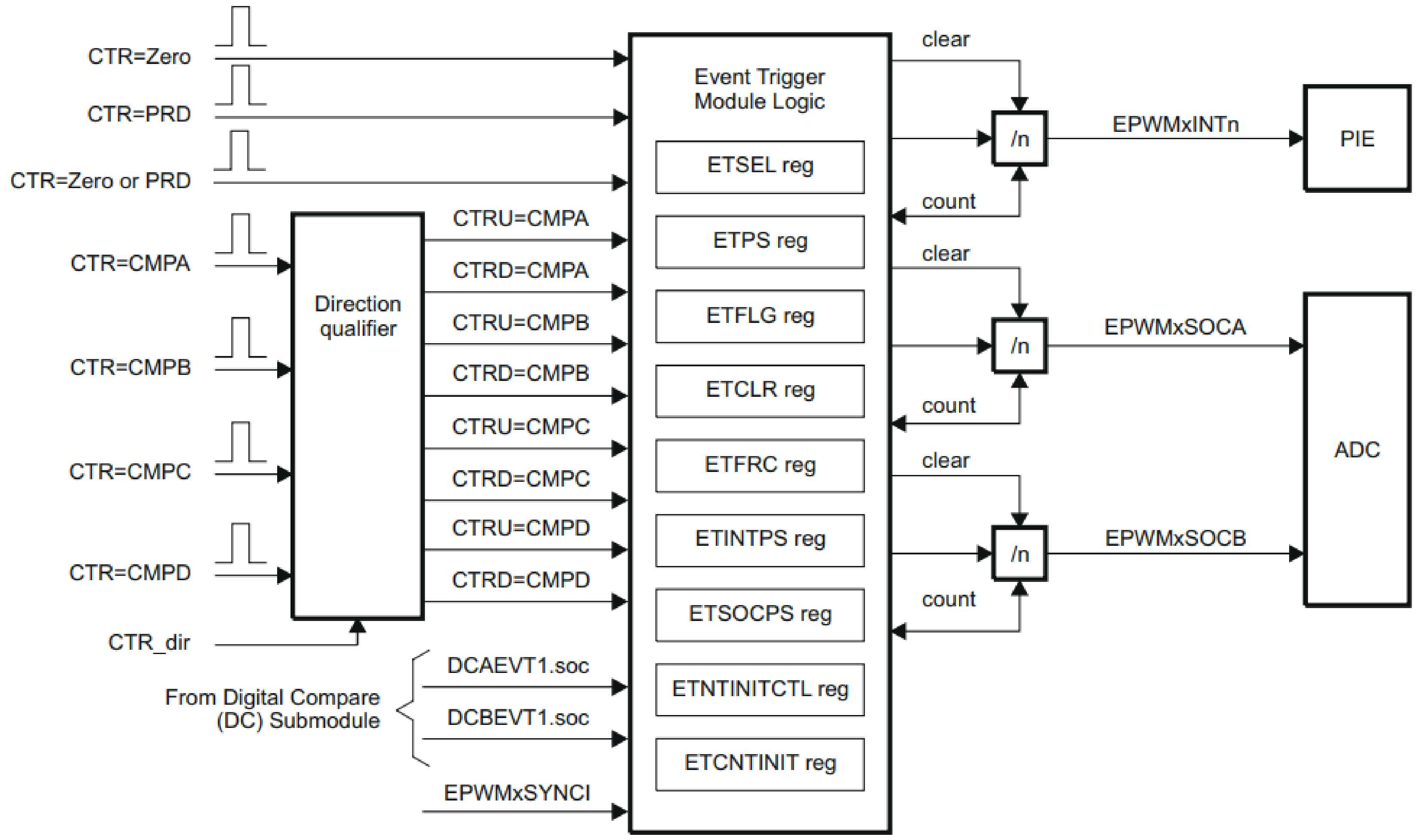
# 4. ePWM Interrupt

## 4.1 ET block



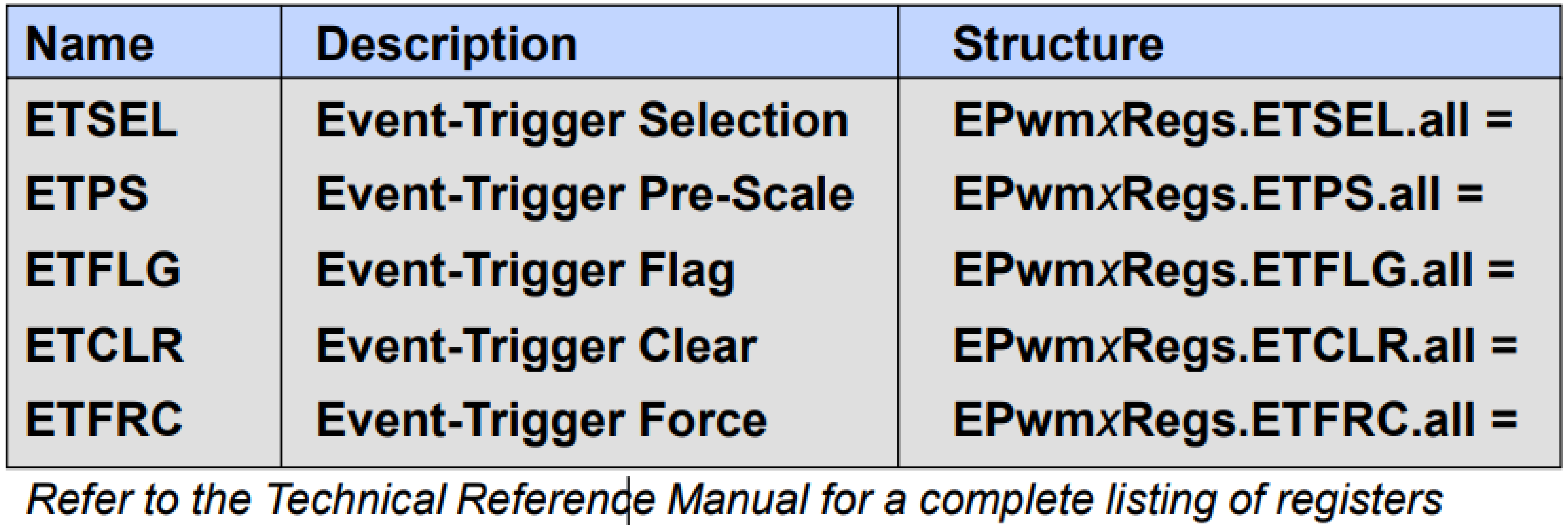
## 4. ePWM Interrupt

### 4.1 ET block



## 4. ePWM Interrupt

### 4.1 ET block

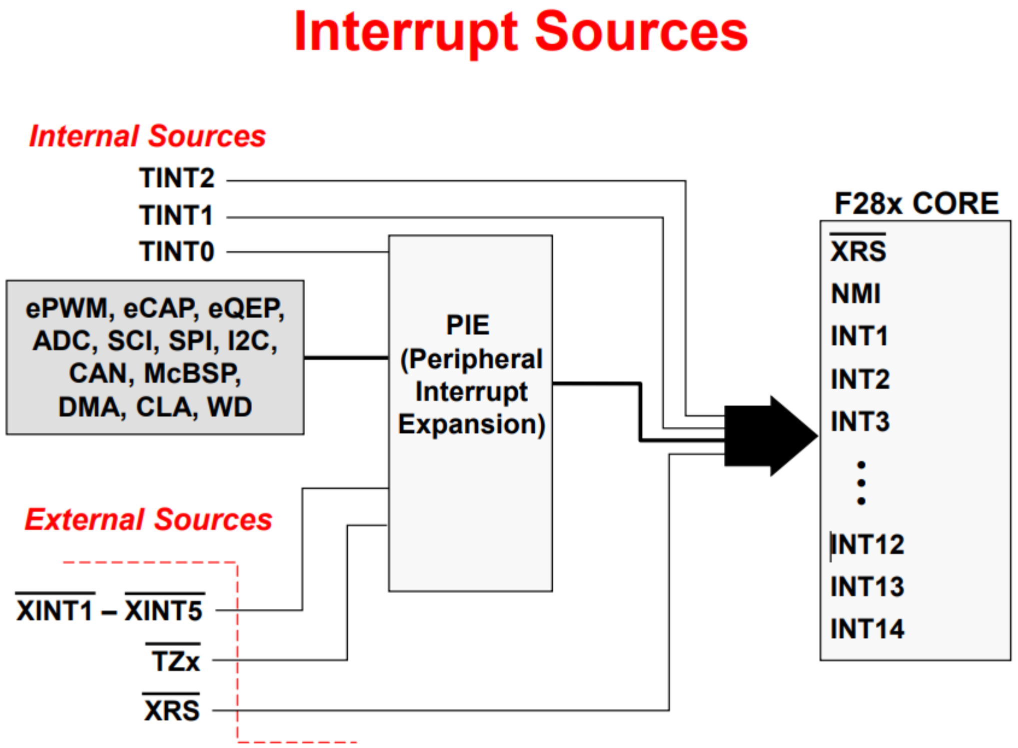


| Name | Description | Structure |
|---|---|---|
| ETSEL | Event-Trigger Selection | EPwmxRegs.ETSEL.all = |
| ETPS | Event-Trigger Pre-Scale | EPwmxRegs.ETPS.all = |
| ETFLG | Event-Trigger Flag | EPwmxRegs.ETFLG.all = |
| ETCLR | Event-Trigger Clear | EPwmxRegs.ETCLR.all = |
| ETFRC | Event-Trigger Force | EPwmxRegs.ETFRC.all = |

*Refer to the Technical Reference Manual for a complete listing of registers*

## 5. PIE

### 5.1 Overview



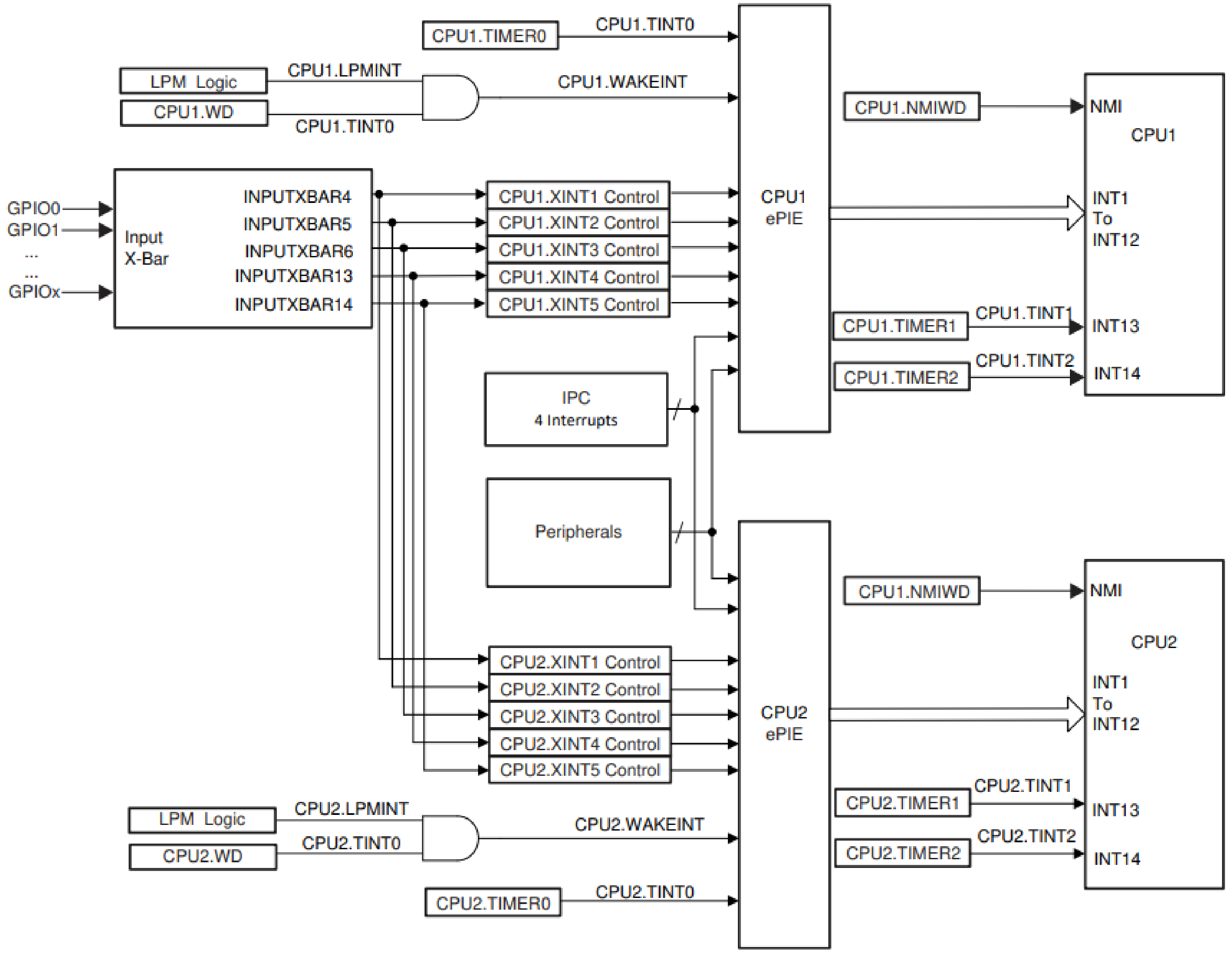


Figure 3-1. Device Interrupt Architecture

# 5. PIE

## 5.1 Overview

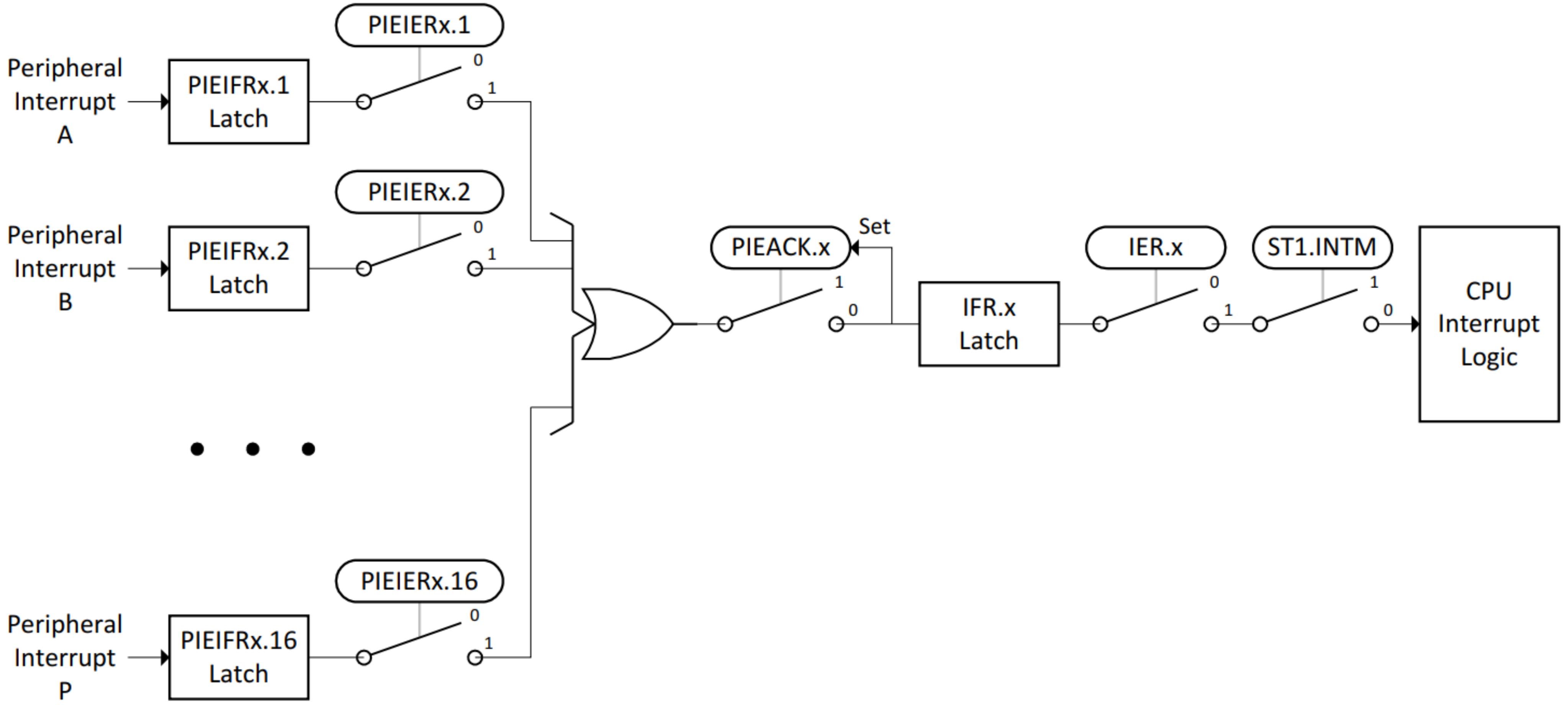
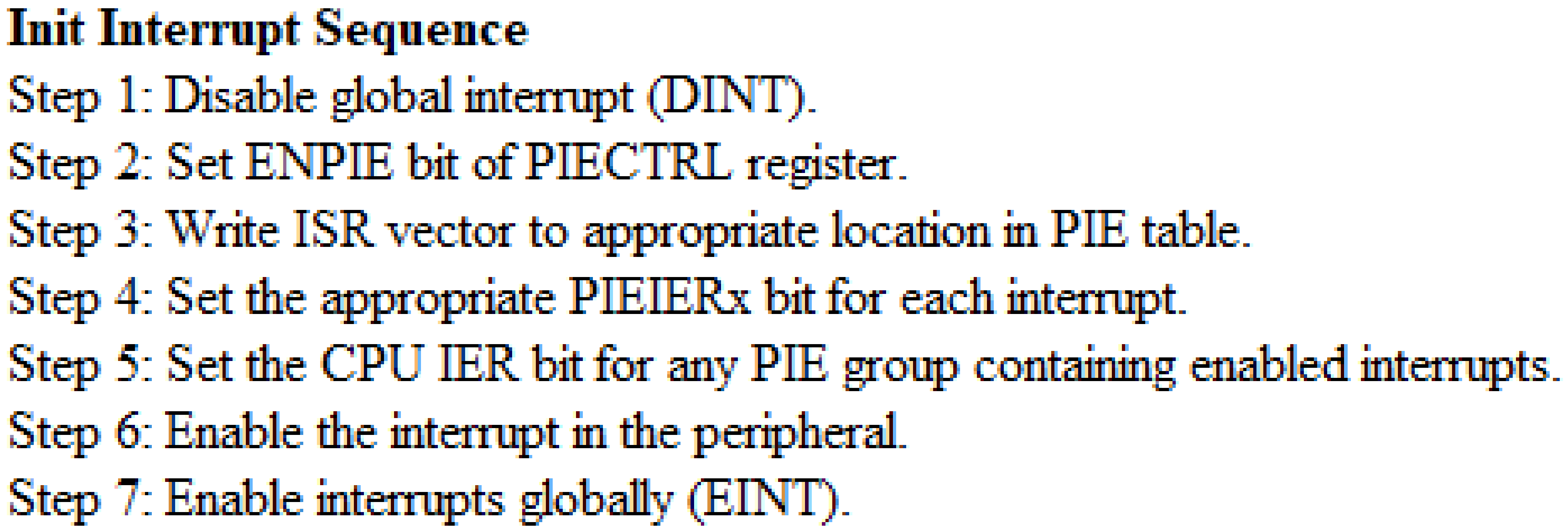


**Figure 3-2. Interrupt Propagation Path**

# 5. PIE

## 5.2 Enable Interrupt



**Init Interrupt Sequence**

Step 1: Disable global interrupt (DINT).

Step 2: Set ENPIE bit of PIECTRL register.

Step 3: Write ISR vector to appropriate location in PIE table.

Step 4: Set the appropriate PIEIERx bit for each interrupt.

Step 5: Set the CPU IER bit for any PIE group containing enabled interrupts.

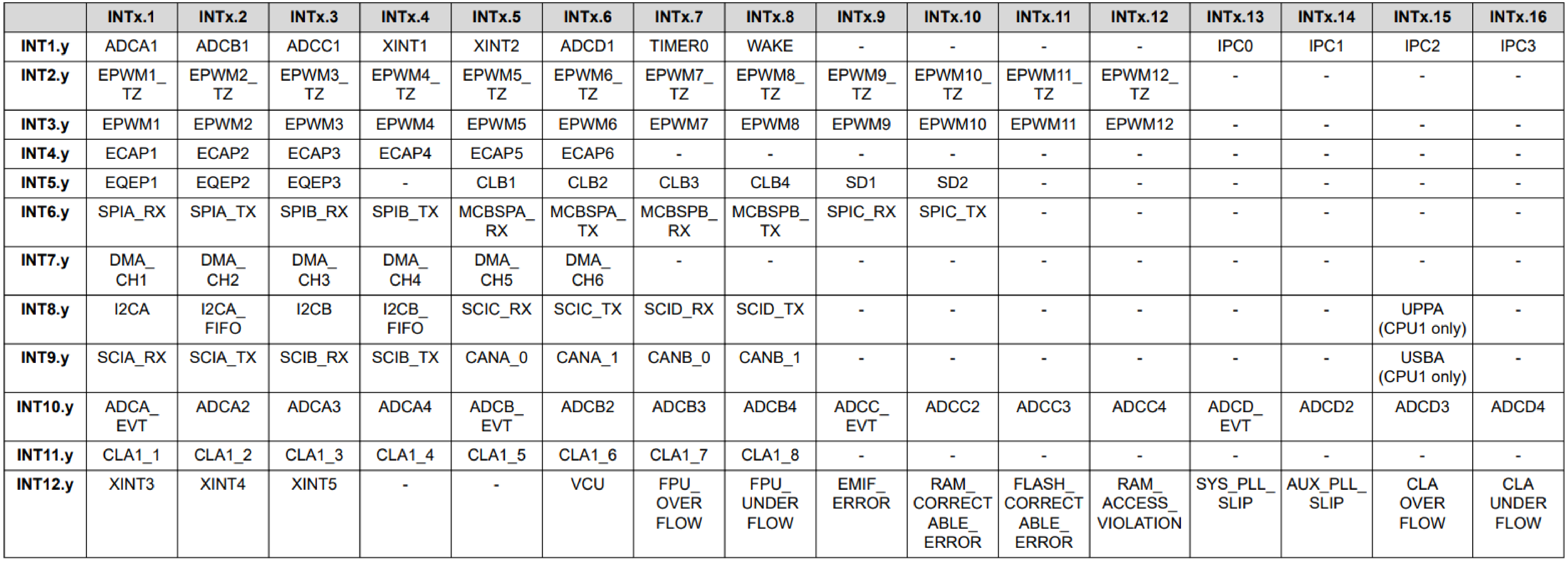
Step 6: Enable the interrupt in the peripheral.

Step 7: Enable interrupts globally (EINT).

# 5. PIE

## 5.3 Channel Mapping

Table 3-2. PIE Channel Mapping



| | INTx.1 | INTx.2 | INTx.3 | INTx.4 | INTx.5 | INTx.6 | INTx.7 | INTx.8 | INTx.9 | INTx.10 | INTx.11 | INTx.12 | INTx.13 | INTx.14 | INTx.15 | INTx.16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **INT1.y** | ADCA1 | ADCB1 | ADCC1 | XINT1 | XINT2 | ADCD1 | TIMER0 | WAKE | - | - | - | - | IPC0 | IPC1 | IPC2 | IPC3 |
| **INT2.y** | EPWM1_TZ | EPWM2_TZ | EPWM3_TZ | EPWM4_TZ | EPWM5_TZ | EPWM6_TZ | EPWM7_TZ | EPWM8_TZ | EPWM9_TZ | EPWM10_TZ | EPWM11_TZ | EPWM12_TZ | - | - | - | - |
| **INT3.y** | EPWM1 | EPWM2 | EPWM3 | EPWM4 | EPWM5 | EPWM6 | EPWM7 | EPWM8 | EPWM9 | EPWM10 | EPWM11 | EPWM12 | - | - | - | - |
| **INT4.y** | ECAP1 | ECAP2 | ECAP3 | ECAP4 | ECAP5 | ECAP6 | - | - | - | - | - | - | - | - | - | - |
| **INT5.y** | EQEP1 | EQEP2 | EQEP3 | - | CLB1 | CLB2 | CLB3 | CLB4 | SD1 | SD2 | - | - | - | - | - | - |
| **INT6.y** | SPIA_RX | SPIA_TX | SPIB_RX | SPIB_TX | MCBSPA_RX | MCBSPA_TX | MCBSPB_RX | MCBSPB_TX | SPIC_RX | SPIC_TX | - | - | - | - | - | - |
| **INT7.y** | DMA_CH1 | DMA_CH2 | DMA_CH3 | DMA_CH4 | DMA_CH5 | DMA_CH6 | - | - | - | - | - | - | - | - | - | - |
| **INT8.y** | I2CA | I2CA_FIFO | I2CB | I2CB_FIFO | SCIC_RX | SCIC_TX | SCID_RX | SCID_TX | - | - | - | - | - | - | UPPA (CPU1 only) | - |
| **INT9.y** | SCIA_RX | SCIA_TX | SCIB_RX | SCIB_TX | CANA_0 | CANA_1 | CANB_0 | CANB_1 | - | - | - | - | - | - | USBA (CPU1 only) | - |
| **INT10.y** | ADCA_EVT | ADCA2 | ADCA3 | ADCA4 | ADCB_EVT | ADCB2 | ADCB3 | ADCB4 | ADCC_EVT | ADCC2 | ADCC3 | ADCC4 | ADCD_EVT | ADCD2 | ADCD3 | ADCD4 |
| **INT11.y** | CLA1_1 | CLA1_2 | CLA1_3 | CLA1_4 | CLA1_5 | CLA1_6 | CLA1_7 | CLA1_8 | - | - | - | - | - | - | - | - |
| **INT12.y** | XINT3 | XINT4 | XINT5 | - | - | VCU | FPU_OVER FLOW | FPU_UNDER FLOW | EMIF_ERROR | RAM_CORRECT ABLE_ERROR | FLASH_CORRECT ABLE_ERROR | RAM_ACCESS_VIOLATION | SYS_PLL_SLIP | AUX_PLL_SLIP | CLA OVER FLOW | CLA UNDER FLOW |

# 5. PIE

## 5.4 Register

### Interrupt Flag Register (IFR)

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RTOSINT | DLOGINT | INT14 | INT13 | INT12 | INT11 | INT10 | INT9 |

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| INT8 | INT7 | INT6 | INT5 | INT4 | INT3 | INT2 | INT1 |

| | |
|---|---|
| Pending : | $IFR_{Bit} = 1$ |
| Absent : | $IFR_{Bit} = 0$ |

### Interrupt Enable Register (IER)

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|
| RTOSINT | DLOGINT | INT14 | INT13 | INT12 | INT11 | INT10 | INT9 |

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| INT8 | INT7 | INT6 | INT5 | INT4 | INT3 | INT2 | INT1 |

| | | |
|---|---|---|
| Enable: | Set | $IER_{Bit} = 1$ |
| Disable: | Clear | $IER_{Bit} = 0$ |

### Interrupt Global Mask Bit

| | | Bit 0 |
|---|---|---|
| ST1 | | $\overline{\text{INTM}}$ |

- INTM used to globally enable/disable interrupts:
  - Enable: $\overline{\text{INTM}} = 0$
  - Disable: $\overline{\text{INTM}} = 1$ (reset value)

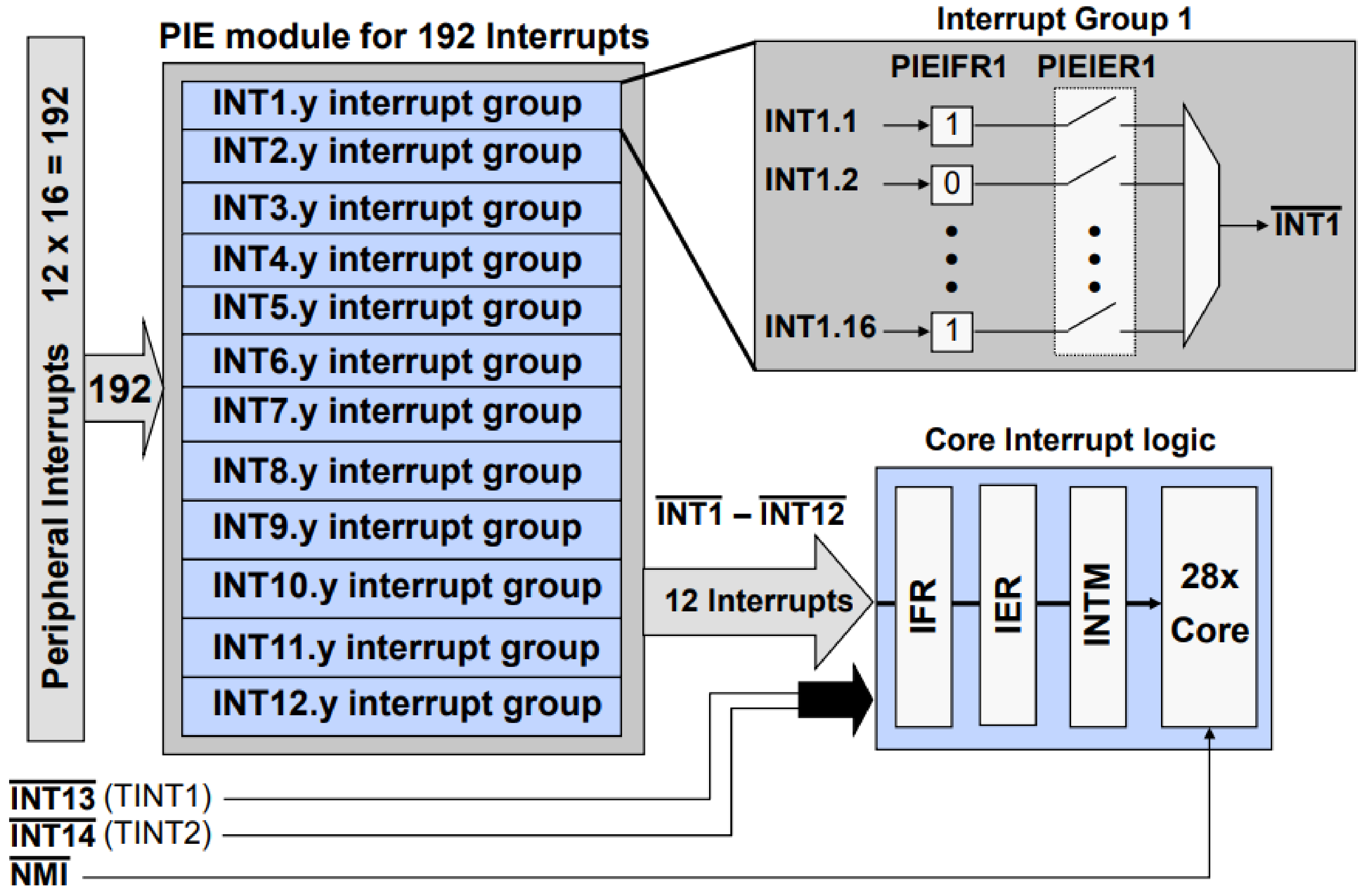
```
/*** Global Interrupts ***/
    asm(“  CLRC  INTM”);      //enable global interrupts
    asm(“  SETC  INTM”);      //disable global interrupts
```

In C code, controlSUITE's DINT and EINT macros can be used for this purpose

# 5. PIE

## 5.4 Register

### Peripheral Interrupt Expansion - PIE



# 5. PIE

## 5.4 Register

### PIE Registers

**PIEIFRx register (x = 1 to 12)**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| INTx.16 | INTx.15 | INTx.14 | INTx.13 | INTx.12 | INTx.11 | INTx.10 | INTx.9 | INTx.8 | INTx.7 | INTx.6 | INTx.5 | INTx.4 | INTx.3 | INTx.2 | INTx.1 |

**PIEIERx register (x = 1 to 12)**

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| INTx.16 | INTx.15 | INTx.14 | INTx.13 | INTx.12 | INTx.11 | INTx.10 | INTx.9 | INTx.8 | INTx.7 | INTx.6 | INTx.5 | INTx.4 | INTx.3 | INTx.2 | INTx.1 |

**PIE Interrupt Acknowledge Register (PIEACK)**

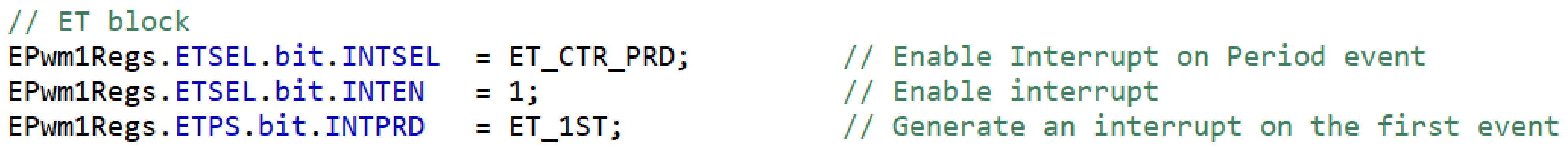
| 15 - 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserved | PIEACKx | | | | | | | | | | | |

**PIECTRL register**

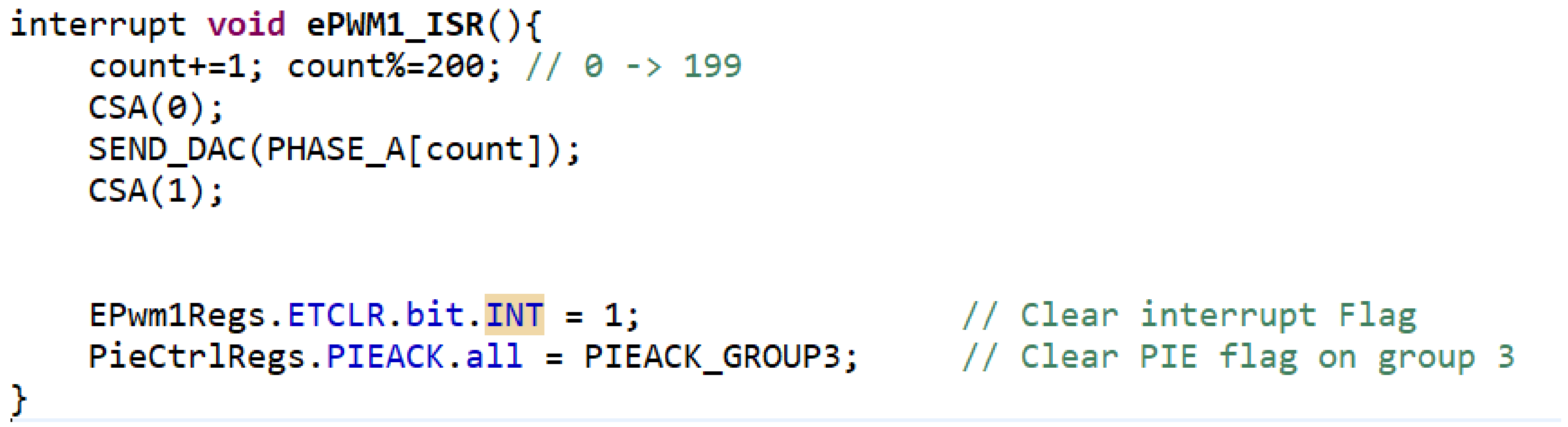
| 15 - 1 | 0 |
|---|---|
| PIEVECT | ENPIE |

```
#include "F2837x_Device.h"
    PieCtrlRegs.PIEIFR1.bit.INTx4 = 1;    //manually set IFR for XINT1 in PIE group 1
    PieCtrlRegs.PIEIER3.bit.INTx2 = 1;    //enable PWM2 interrupt in PIE group 3
    PieCtrlRegs.PIEACK.all = 0x0004;      //acknowledge the PIE group 3
    PieCtrlRegs.PIECTRL.bit.ENPIE = 1;  //enable the PIE
```

## 5.5 Example – Generate 50Hz Sine Waveform Using DAC MCP4921

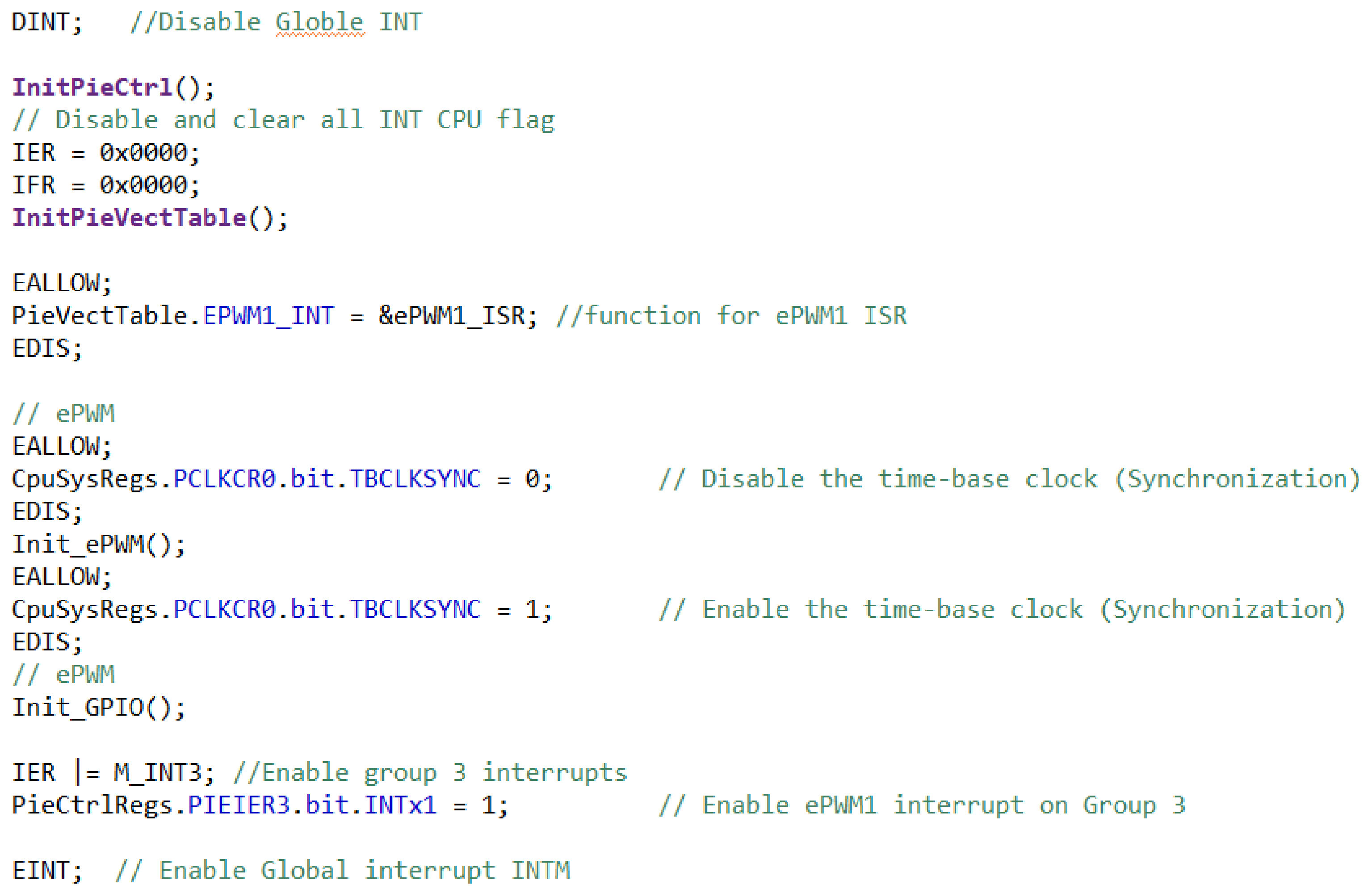


```
// ET block
EPwm1Regs.ETSEL.bit.INTSEL  = ET_CTR_PRD;        // Enable Interrupt on Period event
EPwm1Regs.ETSEL.bit.INTEN   = 1;                 // Enable interrupt
EPwm1Regs.ETPS.bit.INTPRD   = ET_1ST;            // Generate an interrupt on the first event
```



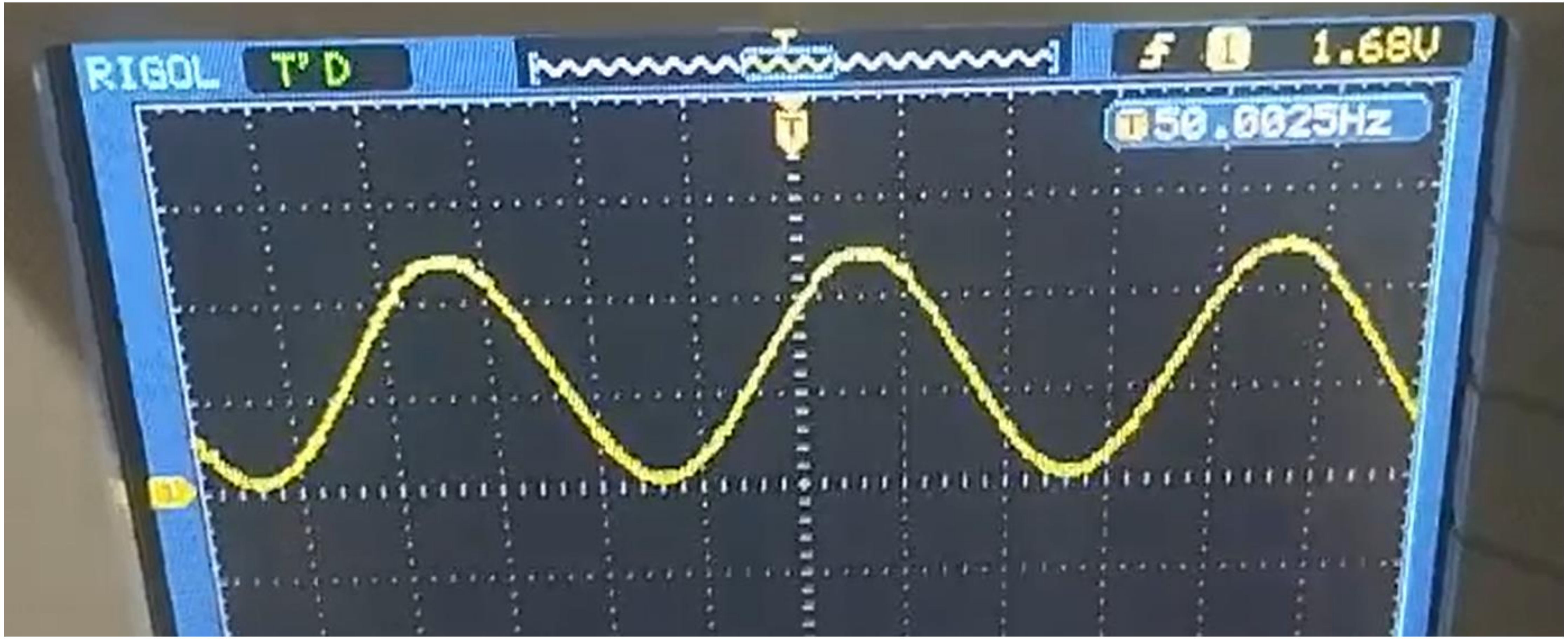
```
interrupt void ePWM1_ISR(){
    count+=1; count%=200; // 0 -> 199
    CSA(0);
    SEND_DAC(PHASE_A[count]);
    CSA(1);


    EPwm1Regs.ETCLR.bit.INT = 1;              // Clear interrupt Flag
    PieCtrlRegs.PIEACK.all = PIEACK_GROUP3;   // Clear PIE flag on group 3
}
```

## 5.5 Example – Generate 50Hz Sine Waveform Using DAC MCP4921



```
DINT;   //Disable Globle INT

InitPieCtrl();
// Disable and clear all INT CPU flag
IER = 0x0000;
IFR = 0x0000;
InitPieVectTable();

EALLOW;
PieVectTable.EPWM1_INT = &ePWM1_ISR; //function for ePWM1 ISR
EDIS;

// ePWM
EALLOW;
CpuSysRegs.PCLKCR0.bit.TBCLKSYNC = 0;       // Disable the time-base clock (Synchronization)
EDIS;
Init_ePWM();
EALLOW;
CpuSysRegs.PCLKCR0.bit.TBCLKSYNC = 1;       // Enable the time-base clock (Synchronization)
EDIS;
// ePWM
Init_GPIO();

IER |= M_INT3; //Enable group 3 interrupts
PieCtrlRegs.PIEIER3.bit.INTx1 = 1;          // Enable ePWM1 interrupt on Group 3

EINT;  // Enable Global interrupt INTM
```

## 5.5 Example – Generate 50Hz Sine Waveform Using DAC MCP4921



## THANKS FOR WATCHING

**HCMUTE**

**Monna Dang (Dang Hoang Anh Chuong)**

**02/2024**

## Some Useful Links

Resource Explorer (Online course)
https://dev.ti.com/tirex/explore/node?node=A__AEd34C6EIRh7UOzlmPh7Fg__c2000ware_software_package__gYkahfz__LATEST

https://software-dl.ti.com/C2000/docs/optimization_guide/intro.html

CCS Guide
https://software-dl.ti.com/ccs/esd/documents/users_guide/index.html

DESIGNDRIVE
https://www.ti.com/tool/DESIGNDRIVE

FAQ
https://e2e.ti.com/support/microcontrollers/c2000-microcontrollers-group/c2000/f/c2000-microcontrollers-forum

SinTable GEN
https://ppelikan.github.io/drlut/