

Introduction

Note: This family reference manual section is meant to serve as a complement to device data sheets. Depending on the device variant, this manual section may not apply to all dsPIC33 devices. Please consult the note at the beginning of the chapter in the specific device data sheet to check whether this document supports the device you are using.

Device data sheets and family reference manual sections are available for download from the Microchip Worldwide Website at: www.microchip.com.

This document describes the features and use of the High-Resolution Pulse-Width Modulation (PWM) with Fine Edge Placement. This flexible module provides features to support many types of Motor Control (MC) and Power Control (PC) applications, including:

- AC-to-DC Converters
- DC-to-DC Converters
- AC and DC Motor Control: Brushed DC, BLDC, PMSM, ACIM, SRM, Stepper, etc.
- Inverters
- Battery Chargers
- Digital Lighting
- Power Factor Correction (PFC)

High-Level Features

- Up to Eight Independent PWM Generators, Each With Dual Outputs
- Operating Modes:
 - Independent Edge PWM mode
 - Variable Phase PWM mode
 - Independent Edge PWM mode, Dual Output
 - Center-Aligned PWM mode
 - Double Update Center-Aligned PWM mode
 - Dual Edge Center-Aligned PWM mode
- Output modes:
 - Complementary
 - Independent
 - Push-Pull
- Dead-Time Generator
- Dead-Time Compensation
- Leading-Edge Blanking (LEB)
- Output Override for Fault Handling
- Flexible Period/Duty Cycle Updating Options

- PWM Control Inputs (PCI) for PWM Pin Overrides and External PWM Synchronization
- Advanced Triggering Options
- Combinatorial Logic Output
- PWM Event Outputs

Table of Contents

| | |
|--|-----|
| Introduction..... | 1 |
| High-Level Features..... | 1 |
| 1. Registers..... | 5 |
| 2. Register Maps..... | 6 |
| 2.1. Common Functions Register Map..... | 7 |
| 2.2. PWM Generator Register Map..... | 21 |
| 3. Architecture Overview..... | 52 |
| 4. Operation..... | 55 |
| 4.1. PWM Clocking..... | 55 |
| 4.2. PWM Generator (PG) Features..... | 60 |
| 4.3. Common Features..... | 102 |
| 4.4. Lock and Write Restrictions..... | 108 |
| 5. Application Examples..... | 112 |
| 5.1. Six-Step Commutation of Three-Phase BLDC Motor..... | 112 |
| 5.2. Three-Phase Sinusoidal Control of PMSM/ACIM Motors..... | 121 |
| 5.3. Simple Complementary PWM Output..... | 124 |
| 5.4. Cycle-by-Cycle Current Limit Mode..... | 125 |
| 5.5. External Period Reset Mode..... | 127 |
| 6. Interrupts..... | 130 |
| 7. Operation in Power-Saving Modes..... | 131 |
| 7.1. Operation in Sleep Mode..... | 131 |
| 7.2. Operation in Idle Mode..... | 131 |
| 8. Related Application Notes..... | 132 |
| 9. Revision History..... | 133 |
| 9.1. Revision A (August 2017)..... | 133 |
| 9.2. Revision B (February 2018)..... | 133 |
| 9.3. Revision C (February 2019)..... | 133 |
| 9.4. Revision D (December 2020)..... | 133 |
| 9.5. Revision E (February 2024)..... | 134 |
| The Microchip Website..... | 135 |
| Product Change Notification Service..... | 135 |
| Customer Support..... | 135 |
| Microchip Devices Code Protection Feature..... | 135 |
| Legal Notice..... | 136 |
| Trademarks..... | 136 |

Quality Management System..... 137

Worldwide Sales and Service..... 138

1. Registers

There are two categories of Special Function Registers (SFRs) used to control the operation of the PWM module:

- Common, shared by all PWM Generators
- PWM Generator-specific; Auxiliary PWM generators that operate identically to primary PWM generators but provide a total increase in the number of PWM generators

An 'x' in the register name denotes an instance of a PWM Generator.

A 'y' in the register name denotes an instance of a common function. An optional 'w' in the register name denotes an instance of an Auxiliary PWM Generator.

The LOCK bit in the PCLKCON register may be set in software to block writes to certain registers and bits. See [4.2. PWM Generator \(PG\) Features](#) for more information. Writes to certain data and control registers are not safe at certain times of a PWM cycle or when the module is enabled.

2. Register Maps

Section [2.1. Common Functions Register Map](#) provides a brief summary of the related common High-Resolution PWM with Fine Edge Placement registers. Section [2.2. PWM Generator Register Map](#) provides a brief summary of the PWM Generator registers. The corresponding registers appear after the summaries, followed by a detailed description of each register.

2.1 Common Functions Register Map

Note: The number of LOGCONy and PWMEVTy registers are device-dependent. Refer to the device data sheet for availability.

| Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|----------|----------------|---------|-------------|----------|--------|--------------|--------------|--------|
| PCLKCON | 7:0 | | | DIVSEL[1:0] | | | | MCLKSEL[1:0] | |
| | 15:8 | HRRDY | HRERR | | | | | | LOCK |
| FSCL | 7:0 | FSCL[7:0] | | | | | | | |
| | 15:8 | FSCL[15:8] | | | | | | | |
| FSMINPER | 7:0 | FSMINPER[7:0] | | | | | | | |
| | 15:8 | FSMINPER[15:8] | | | | | | | |
| MPHASE | 7:0 | MPHASE[7:0] | | | | | | | |
| | 15:8 | MPHASE[15:8] | | | | | | | |
| MDC | 7:0 | MDC[7:0] | | | | | | | |
| | 15:8 | MDC[15:8] | | | | | | | |
| MPER | 7:0 | MPER[7:0] | | | | | | | |
| | 15:8 | MPER[15:8] | | | | | | | |
| LFSR | 7:0 | LFSR[7:0] | | | | | | | |
| | 15:8 | LFSR[14:8] | | | | | | | |
| CMBTRIGL | 7:0 | CTA8EN | CTA7EN | CTA6EN | CTA5EN | CTA4EN | CTA3EN | CTA2EN | CTA1EN |
| | 15:8 | | | | | | | | |
| CMBTRIGH | 7:0 | CTB8EN | CTB7EN | CTB6EN | CTB5EN | CTB4EN | CTB3EN | CTB2EN | CTB1EN |
| | 15:8 | | | | | | | | |
| LOGCONy | 7:0 | S1yPOL | S2yPOL | PWMLFy[1:0] | | | PWMLFyD[2:0] | | |
| | 15:8 | PWMS1y[3:0] | | | | | PWMS2y[3:0] | | |
| PWMEVTy | 7:0 | EVTySEL[3:0] | | | | | EVTyPGS[2:0] | | |
| | 15:8 | EVTyOEN | EVTyPOL | EVTySTRD | EVTySYNC | | | | |

2.1.1 PWM Clock Control Register

Name: PCLKCON

Legend: C = Clearable bit

| | | | | | | | | |
|--------|-------|-------|----|----|----|----|---|------|
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| | HRRDY | HRERR | | | | | | LOCK |
| Access | R | R/C | | | | | | R/W |
| Reset | 0 | 0 | | | | | | 0 |

| | | | | | | | | |
|--------|---|---|-------------|-----|---|---|--------------|-----|
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | | | DIVSEL[1:0] | | | | MCLKSEL[1:0] | |
| Access | | | R/W | R/W | | | R/W | R/W |
| Reset | | | 0 | 0 | | | 0 | 0 |

Bit 15 – HRRDY High-Resolution Ready

Note: This bit is not present on all devices. Refer to the device-specific data sheet for availability.

| Value | Description |
|-------|--|
| 1 | The high-resolution circuitry is ready |
| 0 | The high-resolution circuitry is not ready |

Bit 14 – HRERR High-Resolution Error^(1,2)

Notes:

1. This bit is not present on all devices. Refer to the device-specific data sheet for availability.
2. User software may write a '0' to this location to request a reset of the High-Resolution block when HRRDY = 1.

| Value | Description |
|-------|---|
| 1 | An error has occurred; PWM signals will have limited resolution |
| 0 | No error has occurred; PWM signals will have full resolution when HRRDY = 1 |

Bit 8 – LOCK Lock

Note: A device-specific unlock sequence must be performed before this bit can be cleared. Refer to the device data sheet for the unlock sequence.

| Value | Description |
|-------|---|
| 1 | Write-protected registers and bits are locked |
| 0 | Write-protected registers and bits are unlocked |

Bits 5:4 – DIVSEL[1:0] PWM Clock Divider Selection

| Value | Description |
|-------|----------------------|
| 11 | Divide ratio is 1:16 |
| 10 | Divide ratio is 1:8 |
| 01 | Divide ratio is 1:4 |
| 00 | Divide ratio is 1:2 |

Bits 1:0 – MCLKSEL[1:0] PWM Master Clock Selection

Clock sources are device-specific. Refer to the device data sheet for selections.

Note: Do not change the MCLKSEL[1:0] bits while ON (PGxCONL[15]) = 1.

2.1.2 Frequency Scale Register

Name: FSCL

| | | | | | | | | |
|--------|------------|-----|-----|-----|-----|-----|-----|-----|
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| | FSCL[15:8] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | FSCL[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Bits 15:0 – FSCL[15:0] Frequency Scale Register
The value in this register is added to the frequency scaling accumulator at each pwm_master_clk.
When the accumulated value exceeds the value of FSMINPER, a clock pulse is produced.

2.1.3 Frequency Scaling Minimum Period Register

Name: FSMINPER

| | | | | | | | | |
|--------|----------------|-----|-----|-----|-----|-----|-----|-----|
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| | FSMINPER[15:8] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | FSMINPER[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Bits 15:0 – FSMINPER[15:0] Frequency Scaling Minimum Period Register
This register holds the minimum clock period (maximum clock frequency) that can be produced by the frequency scaling circuit.

2.1.4 Master Phase Register

Name: MPHASE

| | | | | | | | | |
|--------|--------------|-----|-----|-----|-----|-----|-----|-----|
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| | MPHASE[15:8] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | MPHASE[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Bits 15:0 – MPHASE[15:0] Master Phase Register
This register holds the phase offset value that can be shared by multiple PWM Generators.

2.1.5 Master Duty Cycle Register

Name: MDC

| | | | | | | | | |
|--------|-----------|-----|-----|-----|-----|-----|-----|-----|
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| | MDC[15:8] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | MDC[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Bits 15:0 – MDC[15:0] Master Duty Cycle Register

This register holds the duty cycle value that can be shared by multiple PWM Generators.

Note: Duty cycle values less than 0x0008 should not be used (0x0020 in High-Resolution mode).

2.1.6 Master Period Register

Name: MPER

| | | | | | | | | |
|--------|------------|-----|-----|-----|-----|-----|-----|-----|
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| | MPER[15:8] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | MPER[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Bits 15:0 – MPER[15:0] Master Period Register

This register holds the period value that can be shared by multiple PWM Generators.

Note: Period values less than 0x0020 should not be used (0x0080 in High-Resolution mode).

2.1.7 Linear Feedback Shift Register

Name: LFSR

| | | | | | | | | |
|--------|------------|-----|-----|-----|-----|-----|-----|-----|
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| | LFSR[14:8] | | | | | | | |
| Access | | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | LFSR[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Bits 14:0 – LFSR[14:0] Linear Feedback Shift Register
A read of this register will provide a 15-bit pseudorandom value.

2.1.8 Combinational Trigger Register Low

Name: CMBTRIGL

| | | | | | | | | |
|--------|----|----|----|----|----|----|---|---|
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Access | | | | | | | | |
| Reset | | | | | | | | |

| | | | | | | | | |
|--------|--------|--------|--------|--------|--------|--------|--------|--------|
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Access | CTA8EN | CTA7EN | CTA6EN | CTA5EN | CTA4EN | CTA3EN | CTA2EN | CTA1EN |
| Reset | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Bit 7 – CTA8EN Enable Trigger Output from PWM Generator #8 as Source for Combinational Trigger A

| Value | Description |
|-------|---|
| 1 | Enables specified trigger signal to be OR'd into the Combinatorial Trigger A signal |
| 0 | Disabled |

Bit 6 – CTA7EN Enable Trigger Output from PWM Generator #7 as Source for Combinational Trigger A

| Value | Description |
|-------|---|
| 1 | Enables specified trigger signal to be OR'd into the Combinatorial Trigger A signal |
| 0 | Disabled |

Bit 5 – CTA6EN Enable Trigger Output from PWM Generator #6 as Source for Combinational Trigger A

| Value | Description |
|-------|---|
| 1 | Enables specified trigger signal to be OR'd into the Combinatorial Trigger A signal |
| 0 | Disabled |

Bit 4 – CTA5EN Enable Trigger Output from PWM Generator #5 as Source for Combinational Trigger A

| Value | Description |
|-------|---|
| 1 | Enables specified trigger signal to be OR'd into the Combinatorial Trigger A signal |
| 0 | Disabled |

Bit 3 – CTA4EN Enable Trigger Output from PWM Generator #4 as Source for Combinational Trigger A

| Value | Description |
|-------|---|
| 1 | Enables specified trigger signal to be OR'd into the Combinatorial Trigger A signal |
| 0 | Disabled |

Bit 2 – CTA3EN Enable Trigger Output from PWM Generator #3 as Source for Combinational Trigger A

| Value | Description |
|-------|---|
| 1 | Enables specified trigger signal to be OR'd into the Combinatorial Trigger A signal |
| 0 | Disabled |

Bit 1 – CTA2EN Enable Trigger Output from PWM Generator #2 as Source for Combinational Trigger A

| Value | Description |
|-------|---|
| 1 | Enables specified trigger signal to be OR'd into the Combinatorial Trigger A signal |
| 0 | Disabled |

Bit 0 – CTA1EN Enable Trigger Output from PWM Generator #1 as Source for Combinational Trigger A

| Value | Description |
|-------|---|
| 1 | Enables specified trigger signal to be OR'd into the Combinatorial Trigger A signal |
| 0 | Disabled |

2.1.9 Combinational Trigger Register High

Name: CMBTRIGH

| | | | | | | | | |
|--------|----|----|----|----|----|----|---|---|
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Access | | | | | | | | |
| Reset | | | | | | | | |

| | | | | | | | | |
|--------|--------|--------|--------|--------|--------|--------|--------|--------|
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Access | CTB8EN | CTB7EN | CTB6EN | CTB5EN | CTB4EN | CTB3EN | CTB2EN | CTB1EN |
| Reset | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Bit 7 – CTB8EN Enable Trigger Output from PWM Generator #8 as Source for Combinational Trigger B

| Value | Description |
|-------|---|
| 1 | Enables specified trigger signal to be OR'd into the Combinatorial Trigger B signal |
| 0 | Disabled |

Bit 6 – CTB7EN Enable Trigger Output from PWM Generator #7 as Source for Combinational Trigger B

| Value | Description |
|-------|---|
| 1 | Enables specified trigger signal to be OR'd into the Combinatorial Trigger B signal |
| 0 | Disabled |

Bit 5 – CTB6EN Enable Trigger Output from PWM Generator #6 as Source for Combinational Trigger B

| Value | Description |
|-------|---|
| 1 | Enables specified trigger signal to be OR'd into the Combinatorial Trigger B signal |
| 0 | Disabled |

Bit 4 – CTB5EN Enable Trigger Output from PWM Generator #5 as Source for Combinational Trigger B

| Value | Description |
|-------|---|
| 1 | Enables specified trigger signal to be OR'd into the Combinatorial Trigger B signal |
| 0 | Disabled |

Bit 3 – CTB4EN Enable Trigger Output from PWM Generator #4 as Source for Combinational Trigger B

| Value | Description |
|-------|---|
| 1 | Enables specified trigger signal to be OR'd into the Combinatorial Trigger B signal |
| 0 | Disabled |

Bit 2 – CTB3EN Enable Trigger Output from PWM Generator #3 as Source for Combinational Trigger B

| Value | Description |
|-------|---|
| 1 | Enables specified trigger signal to be OR'd into the Combinatorial Trigger B signal |
| 0 | Disabled |

Bit 1 – CTB2EN Enable Trigger Output from PWM Generator #2 as Source for Combinational Trigger B

| Value | Description |
|-------|---|
| 1 | Enables specified trigger signal to be OR'd into the Combinatorial Trigger B signal |
| 0 | Disabled |

Bit 0 – CTB1EN Enable Trigger Output from PWM Generator #1 as Source for Combinational Trigger B

| Value | Description |
|-------|---|
| 1 | Enables specified trigger signal to be OR'd into the Combinatorial Trigger B signal |
| 0 | Disabled |

2.1.10 Combinatorial PWM Logic Control Register y

Name: LOGCONy

Note: 'y' denotes a common instance (A-F); the number of the available combinatorial PWM logic is device-dependent. Refer to the device data sheet for availability.

| | | | | | | | | |
|--------|-------------|--------|-------------|-----|-------------|--------------|-----|-----|
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| | PWMS1y[3:0] | | | | PWMS2y[3:0] | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | S1yPOL | S2yPOL | PWMLFy[1:0] | | | PWMLFyD[2:0] | | |
| Access | R/W | R/W | R/W | R/W | | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | | 0 | 0 | 0 |

Bits 15:12 – PWMS1y[3:0] Combinatorial PWM Logic Source #1 Selection

Note: Logic function input will be connected to '0' if the PWM channel is not present.

| Value | Description |
|-------|-------------|
| 1111 | PWM8L |
| 1110 | PWM8H |
| 1101 | PWM7L |
| 1100 | PWM7H |
| 1011 | PWM6L |
| 1010 | PWM6H |
| 1001 | PWM5L |
| 1000 | PWM5H |
| 0111 | PWM4L |
| 0110 | PWM4H |
| 0101 | PWM3L |
| 0100 | PWM3H |
| 0011 | PWM2L |
| 0010 | PWM2H |
| 0001 | PWM1L |
| 0000 | PWM1H |

Bits 11:8 – PWMS2y[3:0] Combinatorial PWM Logic Source #2 Selection

Note: Logic function input will be connected to '0' if the PWM channel is not present.

| Value | Description |
|-------|-------------|
| 1111 | PWM8L |
| 1110 | PWM8H |
| 1101 | PWM7L |
| 1100 | PWM7H |
| 1011 | PWM6L |
| 1010 | PWM6H |
| 1001 | PWM5L |
| 1000 | PWM5H |
| 0111 | PWM4L |
| 0110 | PWM4H |
| 0101 | PWM3L |
| 0100 | PWM3H |
| 0011 | PWM2L |
| 0010 | PWM2H |

| Value | Description |
|-------|-------------|
| 0001 | PWM1L |
| 0000 | PWM1H |

Bit 7 – S1yPOL Combinatorial PWM Logic Source #1 Polarity

| Value | Description |
|-------|-------------------------|
| 1 | Input is inverted |
| 0 | Input is positive logic |

Bit 6 – S2yPOL Combinatorial PWM Logic Source #2 Polarity

| Value | Description |
|-------|-------------------------|
| 1 | Input is inverted |
| 0 | Input is positive logic |

Bits 5:4 – PWMLFy[1:0] Combinatorial PWM Logic Function Selection

| Value | Description |
|-------|-----------------------|
| 11 | Reserved |
| 10 | PWMS1y ^ PWMS2y (XOR) |
| 01 | PWMS1y & PWMS2y (AND) |
| 00 | PWMS1y PWMS2y (OR) |

Bits 2:0 – PWMLFyD[2:0] Combinatorial PWM Logic Destination Selection

Note: Instances of y = A, C, E of LOGCONy assign logic function output to the PWMxH pin. Instances of y = B, D, F of LOGCONy assign logic function to the PWMxL pin.

| Value | Description |
|-------|---|
| 111 | Logic function is assigned to PWM8 |
| 110 | Logic function is assigned to PWM7 |
| 101 | Logic function is assigned to PWM6 |
| 100 | Logic function is assigned to PWM5 |
| 011 | Logic function is assigned to PWM4 |
| 010 | Logic function is assigned to PWM3 |
| 001 | Logic function is assigned to PWM2 |
| 000 | No assignment, combinatorial PWM logic function is disabled |

2.1.11 PWM Event Output Control Register y

Name: PWMEVTy

Note: 'y' denotes a common instance (A-F); the number of the available combinatorial PWM logic is device-dependent. Refer to the device data sheet for availability.

| | | | | | | | | |
|--------|---------|---------|----------|----------|----|----|---|---|
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| | EVTyOEN | EVTyPOL | EVTySTRD | EVTySYNC | | | | |
| Access | R/W | R/W | R/W | R/W | | | | |
| Reset | 0 | 0 | 0 | 0 | | | | |

| | | | | | | | | |
|--------|--------------|-----|-----|-----|---|--------------|-----|-----|
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | EVTySEL[3:0] | | | | | EVTyPGS[2:0] | | |
| Access | R/W | R/W | R/W | R/W | | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | | 0 | 0 | 0 |

Bit 15 – EVTyOEN PWM Event Output Enable

| Value | Description |
|-------|--|
| 1 | Event output signal is output on the PWMEy pin |
| 0 | Event output signal is internal only |

Bit 14 – EVTyPOL PWM Event Output Polarity

| Value | Description |
|-------|------------------------------------|
| 1 | Event output signal is active-low |
| 0 | Event output signal is active-high |

Bit 13 – EVTySTRD PWM Event Output Stretch Disable

Note: The event signal is stretched using peripheral_clk because different PWM Generators may be operating from different clock sources.

| Value | Description |
|-------|--|
| 1 | Event output signal pulse width is not stretched |
| 0 | Event output signal is stretched to eight PWM clock cycles minimum |

Bit 12 – EVTySYNC PWM Event Output Sync

Event output signal pulse will be synchronized to peripheral_clk.

| Value | Description |
|-------|---|
| 1 | Event output signal is synchronized to the system clock |
| 0 | Event output is not synchronized to the system clock |

Bits 7:4 – EVTySEL[3:0] PWM Event Selection

Note: This is the PWM Generator output signal prior to Output mode logic and any output override logic.

| Value | Description |
|-----------|---|
| 1111 | High-resolution error event signal |
| 1110-1010 | Reserved |
| 1001 | ADC Trigger 2 signal |
| 1000 | ADC Trigger 1 signal |
| 0111 | STEER signal (available in Push-Pull Output modes only) |
| 0110 | CAHALF signal (available in Center-Aligned modes only) |
| 0101 | PCI Fault active output signal |
| 0100 | PCI current limit active output signal |
| 0011 | PCI feed-forward active output signal |
| 0010 | PCI Sync active output signal |

| Value | Description |
|-------|---|
| 0001 | PWM Generator output signal ⁽¹⁾ |
| 0000 | Source is selected by the PGTRGSEL [2:0] bits |

Bits 2:0 – EVTyPGS[2:0] PWM Event Source Selection

Note: No event will be produced if the selected PWM Generator is not present.

| Value | Description |
|-------|------------------|
| 111 | PWM Generator #8 |
| 110 | PWM Generator #7 |
| 101 | PWM Generator #6 |
| 100 | PWM Generator #5 |
| 011 | PWM Generator #4 |
| 010 | PWM Generator #3 |
| 001 | PWM Generator #2 |
| 000 | PWM Generator #1 |

2.2 PWM Generator Register Map

Legend: x = PWM Generator #; y = F, CL, FF or S.

| Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|-----------|----------|----------------|-------------|------------|---------------|-------------|---------------|--------------|----------|
| Reserved | | | | | | | | | |
| PGxCONL | 7:0 | HREN | | | CLKSEL[1:0] | | MODSEL[2:0] | | |
| | 15:8 | ON | | | | | TRGCNT[2:0] | | |
| PGxCONH | 7:0 | Reserved | TRGMOD | | | SOCS[3:0] | | | |
| | 15:8 | MDCSEL | MPERSEL | MPHSEL | | MSTEN | UPDMOD[2:0] | | |
| PGxSTAT | 7:0 | TRSET | TRCLR | CAP | UPDATE | UPDREQ | STEER | CAHALF | TRIG |
| | 15:8 | SEVT | FLTEVT | CLEVT | FFEVT | SACT | FLTACT | CLACT | FFACT |
| PGxIOCONL | 7:0 | FLTDAT[1:0] | | CLDAT[1:0] | | FFDAT[1:0] | | DBDAT[1:0] | |
| | 15:8 | CLMOD | SWAP | OVRENH | OVRENL | OVRDAT[1:0] | | OSYNC[1:0] | |
| PGxIOCONH | 7:0 | | | PMOD[1:0] | | PENH | PENL | POLH | POLL |
| | 15:8 | | CAPSRC[2:0] | | | | | | DTCMPSEL |
| PGxEVTL | 7:0 | | | | UPDTRG[1:0] | | PGTRGSEL[2:0] | | |
| | 15:8 | ADTR1PS[4:0] | | | | | ADTR1EN3 | ADTR1EN2 | ADTR1EN1 |
| PGxEVTH | 7:0 | ADTR2EN3 | ADTR2EN2 | ADTR2EN1 | ADTR1OFS[4:0] | | | | |
| | 15:8 | FLTEN | CLIEN | FFIEN | SIEN | | | IEVTSEL[1:0] | |
| PGxPCIL | 7:0 | SWTERM | PSYNC | PPS | PSS[4:0] | | | | |
| | 15:8 | TSYNCDIS | TERM[2:0] | | | AQPS | | AQSS[2:0] | |
| PGxPCIH | 7:0 | SWPCI | SWPCIM[1:0] | | LATMOD | TQPS | TQSS[2:0] | | |
| | 15:8 | BPEN | BPSEL[2:0] | | | | | ACP[2:0] | |
| Reserved | | | | | | | | | |
| PGxLEBL | 7:0 | LEB[10:6] | | | | | | [2:0] | |
| | 15:8 | LEB[18:11] | | | | | | | |
| PGxLEBH | 7:0 | | | | | PHR | PHF | PLR | PLF |
| | 15:8 | | | | | | PWMPCI[2:0] | | |
| PGxPHASE | 7:0 | PGxPHASE[7:0] | | | | | | | |
| | 15:8 | PGxPHASE[15:8] | | | | | | | |
| Reserved | | | | | | | | | |
| PGxDC | 7:0 | PGxDC[7:0] | | | | | | | |
| | 15:8 | PGxDC[15:8] | | | | | | | |
| PGxDCA | 7:0 | PGxDCA[7:0] | | | | | | | |
| | 15:8 | | | | | | | | |
| PGxPER | 7:0 | PGxPER[7:0] | | | | | | | |
| | 15:8 | PGxPER[15:8] | | | | | | | |
| PGxTRIGA | 7:0 | PGxTRIGA[7:0] | | | | | | | |
| | 15:8 | PGxTRIGA[15:8] | | | | | | | |
| PGxTRIGB | 7:0 | PGxTRIGB[7:0] | | | | | | | |
| | 15:8 | PGxTRIGB[15:8] | | | | | | | |
| PGxTRIGC | 7:0 | PGxTRIGC[7:0] | | | | | | | |
| | 15:8 | PGxTRIGC[15:8] | | | | | | | |
| PGxDTL | 7:0 | DTL[7:0] | | | | | | | |
| | 15:8 | | | | DTL[13:8] | | | | |
| PGxDTH | 7:0 | DTH[7:0] | | | | | | | |
| | 15:8 | | | | DTH[13:8] | | | | |
| PGxCAP | 7:0 | PGxCAP[6:0] | | | | | | | |
| | 15:8 | PGxCAP[14:7] | | | | | | | |

2.2.1 PWM Generator x Control Register Low

Name: PGxCONL

| | | | | | | | | |
|--------|-----|----|----|----|----|-------------|-----|-----|
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| | ON | | | | | TRGCNT[2:0] | | |
| Access | R/W | | | | | R/W | R/W | R/W |
| Reset | 0 | | | | | 0 | 0 | 0 |

| | | | | | | | | |
|--------|------|---|---|-------------|-----|-------------|-----|-----|
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | HREN | | | CLKSEL[1:0] | | MODSEL[2:0] | | |
| Access | R/W | | | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | | | 0 | 0 | 0 | 0 | 0 |

Bit 15 – ON PWM Generator x Enable

| Value | Description |
|-------|------------------------------|
| 1 | PWM Generator is enabled |
| 0 | PWM Generator is not enabled |

Bits 10:8 – TRGCNT[2:0] PWM Generator x Trigger Count Select

| Value | Description |
|-------|---|
| 111 | PWM Generator produces 8 PWM cycles after triggered |
| 110 | PWM Generator produces 7 PWM cycles after triggered |
| 101 | PWM Generator produces 6 PWM cycles after triggered |
| 100 | PWM Generator produces 5 PWM cycles after triggered |
| 011 | PWM Generator produces 4 PWM cycles after triggered |
| 010 | PWM Generator produces 3 PWM cycles after triggered |
| 001 | PWM Generator produces 2 PWM cycles after triggered |
| 000 | PWM Generator produces 1 PWM cycle after triggered |

Bit 7 – HREN PWM Generator x High-Resolution Enable

Note: This bit is not present on all devices. Refer to the device-specific data sheet for availability. When High-Resolution mode is not available, this bit will read as '0'.

| Value | Description |
|-------|--|
| 1 | PWM Generator x operates in High-Resolution mode |
| 0 | PWM Generator x operates in Standard Resolution mode |

Bits 4:3 – CLKSEL[1:0] Clock Selection⁽¹⁾

Notes:

- Do not change the CLKSEL[1:0] bits while ON (PGxCONL[15]) = 1.
- The PWM Generator time base operates from the frequency scaling circuit clock, effectively scaling the duty cycle and period of the PWM Generator output.
- This clock source should not be used when HREN (PGxCONL[7]) = 1.

| Value | Description |
|-------|--|
| 11 | PWM Generator uses the master clock scaled by the frequency scaling circuit ^(2,3) |
| 10 | PWM Generator uses the master clock divided by the clock divider circuit ⁽²⁾ |
| 01 | PWM Generator uses the master clock selected by the MCLKSEL[1:0] (PCLKCON[1:0]) control bits |
| 00 | No clock selected, PWM Generator is in the lowest power state (default) |

Bits 2:0 – MODSEL[2:0] PWM Generator x Mode Selection

| Value | Description |
|-------|---|
| 111 | Dual Edge Center-Aligned PWM mode (interrupt/register update twice per cycle) |

| Value | Description |
|-------|--|
| 110 | Dual Edge Center-Aligned PWM mode (interrupt/register update once per cycle) |
| 101 | Double Update Center-Aligned PWM mode |
| 100 | Center-Aligned PWM mode |
| 011 | Reserved |
| 010 | Independent Edge PWM mode, dual output |
| 001 | Variable Phase PWM mode |
| 000 | Independent Edge PWM mode |

2.2.2 PWM Generator x Control Register High

Name: PGxCONH

| | | | | | | | | |
|--------|--------|---------|--------|----|-------|-------------|-----|-----|
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| | MDCSEL | MPERSEL | MPHSEL | | MSTEN | UPDMOD[2:0] | | |
| Access | R/W | R/W | R/W | | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | | 0 | 0 | 0 | 0 |

| | | | | | | | | |
|--------|----------|--------|---|---|-----------|-----|-----|-----|
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | Reserved | TRGMOD | | | SOCS[3:0] | | | |
| Access | R/W | R/W | | | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | | | 0 | 0 | 0 | 0 |

Bit 15 – MDCSEL Master Duty Cycle Register Select

| Value | Description |
|-------|-----------------------------------|
| 1 | PWM Generator uses MDC register |
| 0 | PWM Generator uses PGxDC register |

Bit 14 – MPERSEL Master Period Register Select

| Value | Description |
|-------|------------------------------------|
| 1 | PWM Generator uses MPER register |
| 0 | PWM Generator uses PGxPER register |

Bit 13 – MPHSEL Master Phase Register Select

| Value | Description |
|-------|--------------------------------------|
| 1 | PWM Generator uses MPHASE register |
| 0 | PWM Generator uses PGxPHASE register |

Bit 11 – MSTEN Master Update Enable

| Value | Description |
|-------|--|
| 1 | PWM Generator broadcasts software set of UPDREQ control bit and EOC signal to other PWM Generators |
| 0 | PWM Generator does not broadcast UPDREQ status bit state or EOC signal |

Bits 10:8 – UPDMOD[2:0] PWM Buffer Update Mode Selection

See [Table 4-5](#) for details.

Bit 7 – Reserved Maintain as '0'

Bit 6 – TRGMOD PWM Generator x Trigger Mode Selection

| Value | Description |
|-------|---|
| 1 | PWM Generator operates in Retriggerable mode |
| 0 | PWM Generator operates in Single Trigger mode |

Bits 3:0 – SOCS[3:0] Start-of-Cycle Selection bits^(1,2,3)
Notes:

1. The PCI selected Sync signal is always available to be OR'd with the selected SOC signal per the SOCS[3:0] bits if the PCI Sync function is enabled.
2. The source selected by the SOCS[3:0] bits MUST operate from the same clock source as the local PWM Generator. If not, the source must be routed through the PCI Sync logic so the trigger signal may be synchronized to the PWM Generator clock domain.
3. PWM Generators are grouped into groups of four: PG1-PG4 and PG5-PG8, if available. Any generator within a group of four may be used to trigger another generator within the same group.

| Value | Description |
|-------------|--|
| 1111 | TRIG bit or PCI Sync function only (no hardware trigger source is selected) |
| 1110 – 0101 | Reserved |
| 0100 | Trigger output selected by PG4 or PG8 PGTRGSEL [2:0] bits (PGxEVTL[2:0]) |
| 0011 | Trigger output selected by PG3 or PG7 PGTRGSEL [2:0] bits (PGxEVTL[2:0]) |
| 0010 | Trigger output selected by PG2 or PG6 PGTRGSEL [2:0] bits (PGxEVTL[2:0]) |
| 0001 | Trigger output selected by PG1 or PG5 PGTRGSEL [2:0] bits (PGxEVTL[2:0]) |
| 0000 | Local EOC – PWM Generator is self-triggered |

2.2.3 PWM Generator x Status Register

Name: PGxSTAT

Legend: C = Clearable bit; HS = Hardware Settable bit

| | | | | | | | | |
|--------|------|--------|-------|-------|------|--------|-------|-------|
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| | SEVT | FLTEVT | CLEVT | FFEVT | SACT | FLTACT | CLACT | FFACT |
| Access | HS/C | HS/C | HS/C | HS/C | R | R | R | R |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | | | | | | | | |
|--------|-------|-------|------|--------|--------|-------|--------|------|
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | TRSET | TRCLR | CAP | UPDATE | UPDREQ | STEER | CAHALF | TRIG |
| Access | W | W | R/HS | R | W | R | R | R |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Bit 15 – SEVT PCI Sync Event

| Value | Description |
|-------|--|
| 1 | A PCI Sync event has occurred (rising edge on PCI Sync output or PCI Sync output is high when module is enabled) |
| 0 | No PCI Sync event has occurred |

Bit 14 – FLTEVT PCI Fault Active Status

| Value | Description |
|-------|---|
| 1 | A Fault event has occurred (rising edge on PCI Fault output or PCI Fault output is high when module is enabled) |
| 0 | No Fault event has occurred |

Bit 13 – CLEVT PCI Current Limit Status

| Value | Description |
|-------|---|
| 1 | A PCI current limit event has occurred (rising edge on PCI current limit output or PCI current limit output is high when module is enabled) |
| 0 | No PCI current limit event has occurred |

Bit 12 – FFEVT PCI Feed-Forward Active Status

| Value | Description |
|-------|--|
| 1 | A PCI feed-forward event has occurred (the rising edge on the PCI feed-forward output or PCI feed-forward output is high when module is enabled) |
| 0 | No PCI feed-forward event has occurred |

Bit 11 – SACT PCI Sync Status

| Value | Description |
|-------|-----------------------------|
| 1 | PCI Sync output is active |
| 0 | PCI Sync output is inactive |

Bit 10 – FLTACT PCI Fault Active Status

| Value | Description |
|-------|------------------------------|
| 1 | PCI Fault output is active |
| 0 | PCI Fault output is inactive |

Bit 9 – CLACT PCI Current Limit Status

| Value | Description |
|-------|--------------------------------------|
| 1 | PCI current limit output is active |
| 0 | PCI current limit output is inactive |

Bit 8 – FFACT PCI Feed-Forward Active Status

| Value | Description |
|-------|-------------------------------------|
| 1 | PCI feed-forward output is active |
| 0 | PCI feed-forward output is inactive |

Bit 7 – TRSET PWM Generator Software Trigger Set

User software writes a '1' to this bit location to trigger a PWM Generator cycle. The bit location always reads as '0'. The TRIG bit will indicate '1' when the PWM Generator is triggered.

Bit 6 – TRCLR PWM Generator Software Trigger Clear

User software writes a '1' to this bit location to stop a PWM Generator cycle. The bit location always reads as '0'. The TRIG bit will indicate '0' when the PWM Generator is not triggered.

Bit 5 – CAP Capture Status

| Value | Description |
|-------|---|
| 1 | PWM Generator time base value has been captured in PGxCAP |
| 0 | No capture has occurred |

Bit 4 – UPDATE PWM Data Register Update Status/Control

| Value | Description |
|-------|--|
| 1 | PWM Data register update is pending – user Data registers are not writable |
| 0 | No PWM Data register update is pending |

Bit 3 – UPDREQ PWM Data Register Update Request

User software writes a '1' to this bit location to request a PWM Data register update. The bit location always reads as '0'. The UPDATE status bit will indicate a '1' when an update is pending.

Bit 2 – STEER Output Steering Status (Push-Pull Output mode only)

| Value | Description |
|-------|---|
| 1 | PWM Generator is in 2nd cycle of Push-Pull mode |
| 0 | PWM Generator is in 1st cycle of Push-Pull mode |

Bit 1 – CAHALF Half Cycle Status (Center-Aligned modes only)

| Value | Description |
|-------|---|
| 1 | PWM Generator is in 2nd half of time base cycle |
| 0 | PWM Generator is in 1st half of time base cycle |

Bit 0 – TRIG Trigger Status

| Value | Description |
|-------|---|
| 1 | PWM Generator is triggered and PWM cycle is in progress |
| 0 | No PWM cycle is in progress |

2.2.4 PWM Generator x I/O Control Register Low

Name: PGxIOCONL

| | | | | | | | | |
|--------|-------|------|--------|--------|-------------|-----|------------|-----|
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| | CLMOD | SWAP | OVRENH | OVRENL | OVRDAT[1:0] | | OSYNC[1:0] | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | | | | | | | | |
|--------|-------------|-----|------------|-----|------------|-----|------------|-----|
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | FLTDAT[1:0] | | CLDAT[1:0] | | FFDAT[1:0] | | DBDAT[1:0] | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Bit 15 – CLMOD Current Limit Mode Select

| Value | Description |
|-------|---|
| 1 | If PCI current limit is active, then the PWMxH and PWMxL output signals are inverted (bit flipping), and the CLDAT[1:0] bits are not used |
| 0 | If PCI current limit is active, then the CLDAT[1:0] bits define the PWM output levels |

Bit 14 – SWAP Swap PWM Signals to PWMxH and PWMxL Device Pins

| Value | Description |
|-------|---|
| 1 | The PWMxH signal is connected to the PWMxL pin and the PWMxL signal is connected to the PWMxH pin |
| 0 | PWMxH/L signals are mapped to their respective pins |

Bit 13 – OVRENH User Override Enable for PWMxH Pin

| Value | Description |
|-------|---|
| 1 | OVRDAT[1] provides data for output on the PWMxH pin |
| 0 | PWM Generator provides data for the PWMxH pin |

Bit 12 – OVRENL User Override Enable for PWMxL Pin

| Value | Description |
|-------|---|
| 1 | OVRDAT[0] provides data for output on the PWMxL pin |
| 0 | PWM Generator provides data for the PWMxL pin |

Bits 11:10 – OVRDAT[1:0] Data for PWMxH/PWMxL Pins if Override is Enabled

| Description |
|--|
| If OVRENH = 1, then OVRDAT[1] provides data for PWMxH. |
| If OVRENL = 1, then OVRDAT[0] provides data for PWMxL. |

Bits 9:8 – OSYNC[1:0] User Output Override Synchronization Control

| Value | Description |
|-------|--|
| 11 | Reserved |
| 10 | User output overrides via the SWAP, OVRENH/L and OVRDAT[1:0] bits occur when specified by the UPDMOD[2:0] bits in the PGxCONH register |
| 01 | User output overrides via the SWAP, OVRENH/L and OVRDAT[1:0] bits occur immediately (as soon as possible) |
| 00 | User output overrides via the SWAP, OVRENH/L and OVRDAT[1:0] bits are synchronized to the local PWM time base (next start of cycle) |

Bits 7:6 – FLTDAT[1:0] Data for PWMxH/PWMxL Pins if FLT Event is Active

| Description |
|---|
| If Fault is active, then FLTDAT[1] provides data for PWMxH. |
| If Fault is active, then FLTDAT[0] provides data for PWMxL. |

Bits 5:4 – CLDAT[1:0] Data for PWMxH/PWMxL Pins if CLMT Event is Active

| Description |
|--|
| If current limit is active, then CLDAT[1] provides data for PWMxH. |
| If current limit is active, then CLDAT[0] provides data for PWMxL. |

Bits 3:2 – FFDAT[1:0] Data for PWMxH/PWMxL Pins if Feed-Forward Event is Active

| Description |
|---|
| If feed-forward is active, then FFDAT[1] provides data for PWMxH. |
| If feed-forward is active, then FFDAT[0] provides data for PWMxL. |

Bits 1:0 – DBDAT[1:0] Data for PWMxH/PWMxL Pins if Debug Mode is Active

| Description |
|---|
| If Debug mode is active and PTFRZ = 1, then DBDAT[1] provides data for PWMxH. |
| If Debug mode is active and PTFRZ = 1, then DBDAT[0] provides data for PWMxL. |

2.2.5 PWM Generator x I/O Control Register High

Name: PGxIOCONH

| | | | | | | | | |
|--------|----|-------------|-----|-----|----|----|---|----------|
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| | | CAPSRC[2:0] | | | | | | DTCMPSEL |
| Access | | R/W | R/W | R/W | | | | R/W |
| Reset | | 0 | 0 | 0 | | | | 0 |

| | | | | | | | | |
|--------|---|---|-----------|-----|------|------|------|------|
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | | | PMOD[1:0] | | PENH | PENL | POLH | POLL |
| Access | | | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | | | 0 | 0 | 0 | 0 | 0 | 0 |

Bits 14:12 – CAPSRC[2:0] Time Base Capture Source Selection

Note: A capture may be initiated in software at any time by writing a '1' to PGxCAP[0].

| Value | Description |
|-------|---|
| 111 | Reserved |
| 110 | Reserved |
| 101 | Reserved |
| 100 | Capture time base value at assertion of selected PCI Fault signal |
| 011 | Capture time base value at assertion of selected PCI current limit signal |
| 010 | Capture time base value at assertion of selected PCI feed-forward signal |
| 001 | Capture time base value at assertion of selected PCI Sync signal |
| 000 | No hardware source selected for time base capture – software only |

Bit 8 – DTCMPSEL Dead-Time Compensation Select

| Value | Description |
|-------|--|
| 1 | Dead-time compensation is controlled by PCI feed-forward limit logic |
| 0 | Dead-time compensation is controlled by PCI Sync logic |

Bits 5:4 – PMOD[1:0] PWM Generator Output Mode Selection

| Value | Description |
|-------|---|
| 11 | Reserved |
| 10 | PWM Generator outputs operate in Push-Pull mode |
| 01 | PWM Generator outputs operate in Independent mode |
| 00 | PWM Generator outputs operate in Complementary mode |

Bit 3 – PENH PWMxH Output Port Enable

| Value | Description |
|-------|---|
| 1 | PWM Generator controls the PWMxH output pin |
| 0 | PWM Generator does not control the PWMxH output pin |

Bit 2 – PENL PWMxL Output Port Enable

| Value | Description |
|-------|---|
| 1 | PWM Generator controls the PWMxL output pin |
| 0 | PWM Generator does not control the PWMxL output pin |

Bit 1 – POLH PWMxH Output Polarity

| Value | Description |
|-------|---------------------------|
| 1 | Output pin is active-low |
| 0 | Output pin is active-high |

Bit 0 – POLL PWMxL Output Polarity

| Value | Description |
|-------|---------------------------|
| 1 | Output pin is active-low |
| 0 | Output pin is active-high |

2.2.6 PWM Generator x Event Register Low

Name: PGxEVTL

| | | | | | | | | |
|--------|--------------|-----|-----|-------------|-----|---------------|----------|----------|
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| | ADTR1PS[4:0] | | | | | ADTR1EN3 | ADTR1EN2 | ADTR1EN1 |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | | | | UPDTRG[1:0] | | PGTRGSEL[2:0] | | |
| Access | | | | R/W | R/W | R/W | R/W | R/W |
| Reset | | | | 0 | 0 | 0 | 0 | 0 |

Bits 15:11 – ADTR1PS[4:0] ADC Trigger 1 Postscaler Selection

| Value | Description |
|-------|-------------|
| 11111 | 1:32 |
| . . . | . . . |
| 00010 | 1:3 |
| 00001 | 1:2 |
| 00000 | 1:1 |

Bit 10 – ADTR1EN3 ADC Trigger 1 Source is PGxTRIGC Compare Event Enable

| Value | Description |
|-------|---|
| 1 | PGxTRIGC register compare event is enabled as trigger source for ADC Trigger 1 |
| 0 | PGxTRIGC register compare event is disabled as trigger source for ADC Trigger 1 |

Bit 9 – ADTR1EN2 ADC Trigger 1 Source is PGxTRIGB Compare Event Enable

| Value | Description |
|-------|---|
| 1 | PGxTRIGB register compare event is enabled as trigger source for ADC Trigger 1 |
| 0 | PGxTRIGB register compare event is disabled as trigger source for ADC Trigger 1 |

Bit 8 – ADTR1EN1 ADC Trigger 1 Source is PGxTRIGA Compare Event Enable

| Value | Description |
|-------|---|
| 1 | PGxTRIGA register compare event is enabled as trigger source for ADC Trigger 1 |
| 0 | PGxTRIGA register compare event is disabled as trigger source for ADC Trigger 1 |

Bits 4:3 – UPDTRG[1:0] Update Trigger Select

| Value | Description |
|-------|--|
| 11 | A write of the PGxTRIGA register automatically sets the UPDREQ bit |
| 10 | A write of the PGxPHASE register automatically sets the UPDREQ bit |
| 01 | A write of the PGxDC register automatically sets the UPDREQ bit |
| 00 | User must set the UPDREQ bit (PGxSTAT[3]) manually |

Bits 2:0 – PGTRGSEL[2:0] PWM Generator Trigger Output Selection

Note: These events are derived from the internal PWM Generator time base comparison events.

| Value | Description |
|-------|---|
| 111 | Reserved |
| 110 | Reserved |
| 101 | Reserved |
| 100 | Reserved |
| 011 | PGxTRIGC compare event is the PWM Generator trigger |
| 010 | PGxTRIGB compare event is the PWM Generator trigger |
| 001 | PGxTRIGA compare event is the PWM Generator trigger |

| Value | Description |
|-------|--|
| 000 | EOC event is the PWM Generator trigger |

2.2.7 PWM Generator x Event Register High

Name: PGxEVTH

| | | | | | | | | |
|--------|-------|-------|-------|------|----|----|--------------|-----|
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| | FLTEN | CLIEN | FFIEN | SIEN | | | IEVTSEL[1:0] | |
| Access | R/W | R/W | R/W | R/W | | | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | | | 0 | 0 |

| | | | | | | | | |
|--------|----------|----------|----------|-----|-----|---------------|-----|-----|
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | ADTR2EN3 | ADTR2EN2 | ADTR2EN1 | | | ADTR1OFS[4:0] | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Bit 15 – FLTEN PCI Fault Interrupt Enable

Note: An interrupt is only generated on the rising edge of the PCI Fault active signal.

| Value | Description |
|-------|-----------------------------|
| 1 | Fault interrupt is enabled |
| 0 | Fault interrupt is disabled |

Bit 14 – CLIEN PCI Current Limit Interrupt Enable

Note: An interrupt is only generated on the rising edge of the PCI current limit active signal.

| Value | Description |
|-------|-------------------------------------|
| 1 | Current limit interrupt is enabled |
| 0 | Current limit interrupt is disabled |

Bit 13 – FFIEN PCI Feed-Forward Interrupt Enable

Note: An interrupt is only generated on the rising edge of the PCI feed-forward active signal.

| Value | Description |
|-------|------------------------------------|
| 1 | Feed-forward interrupt is enabled |
| 0 | Feed-forward interrupt is disabled |

Bit 12 – SIEN PCI Sync Interrupt Enable

Note: An interrupt is only generated on the rising edge of the PCI Sync active signal.

| Value | Description |
|-------|----------------------------|
| 1 | Sync interrupt is enabled |
| 0 | Sync interrupt is disabled |

Bits 9:8 – IEVTSEL[1:0] Interrupt Event Selection

| Value | Description |
|-------|---|
| 11 | Time base interrupts are disabled (Sync, Fault, current limit and feed-forward events can be independently enabled) |
| 10 | Interrupts CPU at ADC Trigger 1 event |
| 01 | Interrupts CPU at TRIGA compare event |
| 00 | Interrupts CPU at EOC |

Bit 7 – ADTR2EN3 ADC Trigger 2 Source is PGxTRIGC Compare Event Enable

| Value | Description |
|-------|---|
| 1 | PGxTRIGC register compare event is enabled as trigger source for ADC Trigger 2 |
| 0 | PGxTRIGC register compare event is disabled as trigger source for ADC Trigger 2 |

Bit 6 – ADTR2EN2 ADC Trigger 2 Source is PGxTRIGB Compare Event Enable

| Value | Description |
|-------|---|
| 1 | PGxTRIGB register compare event is enabled as trigger source for ADC Trigger 2 |
| 0 | PGxTRIGB register compare event is disabled as trigger source for ADC Trigger 2 |

Bit 5 – ADTR2EN1 ADC Trigger 2 Source is PGxTRIGA Compare Event Enable

| Value | Description |
|-------|---|
| 1 | PGxTRIGA register compare event is enabled as trigger source for ADC Trigger 2 |
| 0 | PGxTRIGA register compare event is disabled as trigger source for ADC Trigger 2 |

Bits 4:0 – ADTR1OFS[4:0] ADC Trigger 1 Offset Selection

| Value | Description |
|-------|-----------------------------|
| 11111 | Offset by 31 trigger events |
| . . . | . . . |
| 00010 | Offset by 2 trigger events |
| 00001 | Offset by 1 trigger event |
| 00000 | No offset |

2.2.8 PWM Generator xy PCI Register Low (x = PWM Generator #; y = F, CL, FF or S)

Name: PGxyPCIL

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|--------|----------|-----|-----------|-----|-----|------|-----------|-----|
| | TSYNCDIS | | TERM[2:0] | | | AQPS | AQSS[2:0] | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|--------|--------|-------|-----|----------|-----|-----|-----|-----|
| | SWTERM | PSYNC | PPS | PSS[4:0] | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Bit 15 – TSYNCDIS Termination Synchronization Disable

| Value | Description |
|-------|---|
| 1 | Termination of latched PCI occurs immediately |
| 0 | Termination of latched PCI occurs at PWM EOC |

Bits 14:12 – TERM[2:0] Termination Event Selection

Notes:

1. PCI sources are device-dependent; refer to the device data sheet for availability.
2. Do not use this selection when the ACP[2:0] bits (PGxyPCIH[10:8]) are set for latched on any edge.

| Value | Description |
|-------|--|
| 111 | Selects PCI Source #9 ⁽¹⁾ |
| 110 | Selects PCI Source #8 ⁽¹⁾ |
| 101 | Selects PCI Source #1 (PWM Generator output selected by the PWMPCI[2:0] bits) |
| 100 | PGxTRIGC trigger event |
| 011 | PGxTRIGB trigger event |
| 010 | PGxTRIGA trigger event |
| 001 | Auto-Terminate: Terminate when PCI source transitions from active to inactive ⁽²⁾ |
| 000 | Manual Terminate: Terminate on a write of '1' to the SWTERM bit location |

Bit 11 – AQPS Acceptance Qualifier Polarity Select

| Value | Description |
|-------|--------------|
| 1 | Inverted |
| 0 | Not inverted |

Bits 10:8 – AQSS[2:0] Acceptance Qualifier Source Selection

| Value | Description |
|-------|---|
| 111 | SWPCI control bit only (qualifier forced to '0') |
| 110 | Selects PCI Source #9 |
| 101 | Selects PCI Source #8 |
| 100 | Selects PCI Source #1 (PWM Generator output selected by the PWMPCI[2:0] bits) |
| 011 | PWM Generator is triggered |
| 010 | LEB is active |
| 001 | Duty cycle is active (base PWM Generator signal) |
| 000 | No acceptance qualifier is used (qualifier forced to '1') |

Bit 7 – SWTERM PCI Software Termination

A write of '1' to this location will produce a termination event. This bit location always reads as '0'.

Bit 6 – PSYNC PCI Synchronization Control

| Value | Description |
|-------|---|
| 1 | PCI source is synchronized to PWM EOC |
| 0 | PCI source is not synchronized to PWM EOC |

Bit 5 – PPS PCI Polarity Select

| Value | Description |
|-------|--------------|
| 1 | Inverted |
| 0 | Not inverted |

Bits 4:0 – PSS[4:0] PCI Source Selection

Note: PCI sources are device-dependent; refer to the device data sheet for availability.

| Value | Description |
|-------|---|
| 11111 | PCI Source #31 (reserved) |
| . . . | . . . |
| 00101 | PCI Source #5 (reserved) |
| 00100 | PCI Source #4 (reserved) |
| 00011 | PCI Source #3 (internally connected to Combinatorial Trigger B) |
| 00010 | PCI Source #2 (internally connected to Combinatorial Trigger A) |
| 00001 | PCI Source #1 (internally connected to PWMPCI[2:0] output MUX) |
| 00000 | Software PCI control bit (SWPCI) only |

2.2.9 PWM Generator xy PCI Register High (x = PWM Generator #; y = F, CL, FF or S)

Name: PGxyPCIH

| | | | | | | | | |
|--------|------|------------|-----|-----|----|----------|-----|-----|
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| | BPEN | BPSEL[2:0] | | | | ACP[2:0] | | |
| Access | R/W | R/W | R/W | R/W | | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | | 0 | 0 | 0 |

| | | | | | | | | |
|--------|-------|-------------|-----|--------|------|-----------|-----|-----|
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | SWPCI | SWPCIM[1:0] | | LATMOD | TQPS | TQSS[2:0] | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Bit 15 – BPEN PCI Bypass Enable

| Value | Description |
|-------|--|
| 1 | PCI function is enabled and local PCI logic is bypassed; PWM Generator will be controlled by PCI function in the PWM Generator selected by the BPSEL[2:0] bits |
| 0 | PCI function is not bypassed |

Bits 14:12 – BPSEL[2:0] PCI Bypass Source Selection

Note: Selects '0' if the selected PWM Generator is not present.

| Value | Description |
|-------|---|
| 111 | PCI control is sourced from PWM Generator 8 PCI logic when BPEN = 1 |
| 110 | PCI control is sourced from PWM Generator 7 PCI logic when BPEN = 1 |
| 101 | PCI control is sourced from PWM Generator 6 PCI logic when BPEN = 1 |
| 100 | PCI control is sourced from PWM Generator 5 PCI logic when BPEN = 1 |
| 011 | PCI control is sourced from PWM Generator 4 PCI logic when BPEN = 1 |
| 010 | PCI control is sourced from PWM Generator 3 PCI logic when BPEN = 1 |
| 001 | PCI control is sourced from PWM Generator 2 PCI logic when BPEN = 1 |
| 000 | PCI control is sourced from PWM Generator 1 PCI logic when BPEN = 1 |

Bits 10:8 – ACP[2:0] PCI Acceptance Criteria Selection

Note:

- Don't use this selection when the TERM[2:0] bits (PGxyPCIL[14:12]) are set to auto-termination.

| Value | Description |
|-------|---------------------------------|
| 111 | Reserved |
| 110 | Reserved |
| 101 | Latched any edge ⁽¹⁾ |
| 100 | Latched rising edge |
| 011 | Latched |
| 010 | Any edge |
| 001 | Rising edge |
| 000 | Level-sensitive |

Bit 7 – SWPCI Software PCI Control

| Value | Description |
|-------|---|
| 1 | Drives a '1' to PCI logic assigned to by the SWPCIM[1:0] control bits |
| 0 | Drives a '0' to PCI logic assigned to by the SWPCIM[1:0] control bits |

Bits 6:5 – SWPCIM[1:0] Software PCI Control Mode

| Value | Description |
|-------|-------------|
| 11 | Reserved |

| Value | Description |
|-------|--|
| 10 | SWPCI bit is assigned to termination qualifier logic |
| 01 | SWPCI bit is assigned to acceptance qualifier logic |
| 00 | SWPCI bit is assigned to PCI acceptance logic |

Bit 4 – LATMOD PCI SR Latch Mode

| Value | Description |
|-------|--|
| 1 | SR latch is Reset-dominant in Latched Acceptance modes |
| 0 | SR latch is set-dominant in Latched Acceptance modes |

Bit 3 – TQPS Termination Qualifier Polarity Select

| Value | Description |
|-------|--------------|
| 1 | Inverted |
| 0 | Not inverted |

Bits 2:0 – TQSS[2:0] Termination Qualifier Source Selection

Note:

1. Polarity control bit, TQPS, has no effect on these selections.

| Value | Description |
|-------|---|
| 111 | SWPCI control bit only (qualifier forced to '1') ⁽¹⁾ |
| 110 | Selects PCI Source #9 |
| 101 | Selects PCI Source #8 |
| 100 | Selects PCI Source #1 (PWM Generator output selected by the PWMPCI[2:0] bits) |
| 011 | PWM Generator is triggered |
| 010 | LEB is active |
| 001 | Duty cycle is active (base PWM Generator signal) |
| 000 | No termination qualifier used (qualifier forced to '1') ⁽¹⁾ |

2.2.10 PWM Generator x Leading-Edge Blanking Register Low

Name: PGxLEBL

| | | | | | | | | |
|--------|------------|-----|-----|-----|-----|-------|-----|-----|
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| | LEB[18:11] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | LEB[10:6] | | | | | [2:0] | | |
| Access | R/W | R/W | R/W | R/W | R/W | R | R | R |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Bits 15:3 – LEB[15:3] Leading-Edge Blanking Period

Leading-Edge Blanking period. The three LSBs of the blanking time are not used, providing a blanking resolution of eight PGx_clks. The minimum blanking period is eight PGx_clks, which occurs when LEB[15:3] = 0.

Bits 2:0 – [2:0] Read-Only

2.2.11 PWM Generator x Leading-Edge Blanking Register High

Name: PGxLEBH

| | | | | | | | | |
|--------|----|----|----|----|----|-------------|-----|-----|
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| | | | | | | PWMPCI[2:0] | | |
| Access | | | | | | R/W | R/W | R/W |
| Reset | | | | | | 0 | 0 | 0 |

| | | | | | | | | |
|--------|---|---|---|---|-----|-----|-----|-----|
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | | | | | PHR | PHF | PLR | PLF |
| Access | | | | | R/W | R/W | R/W | R/W |
| Reset | | | | | 0 | 0 | 0 | 0 |

Bits 10:8 – PWMPCI[2:0] PWM Source for PCI Selection

Note: The selected PWM Generator source does not affect the LEB counter. This source can be optionally used as a PCI input, PCI qualifier, PCI terminator or PCI terminator qualifier (see the description in the PGxyPCIL and PGxyPCIH registers for more information).

| Value | Description |
|-------|--|
| 111 | PWM Generator #8 output is made available to PCI logic |
| 110 | PWM Generator #7 output is made available to PCI logic |
| 101 | PWM Generator #6 output is made available to PCI logic |
| 100 | PWM Generator #5 output is made available to PCI logic |
| 011 | PWM Generator #4 output is made available to PCI logic |
| 010 | PWM Generator #3 output is made available to PCI logic |
| 001 | PWM Generator #2 output is made available to PCI logic |
| 000 | PWM Generator #1 output is made available to PCI logic |

Bit 3 – PHR PWMxH Rising Edge Trigger Enable

| Value | Description |
|-------|--|
| 1 | Rising edge of PWMxH will trigger the LEB duration counter |
| 0 | LEB ignores the rising edge of PWMxH |

Bit 2 – PHF PWMxH Falling Edge Trigger Enable

| Value | Description |
|-------|---|
| 1 | Falling edge of PWMxH will trigger the LEB duration counter |
| 0 | LEB ignores the falling edge of PWMxH |

Bit 1 – PLR PWMxL Rising Edge Trigger Enable

| Value | Description |
|-------|--|
| 1 | Rising edge of PWMxL will trigger the LEB duration counter |
| 0 | LEB ignores the rising edge of PWMxL |

Bit 0 – PLF PWMxL Falling Edge Trigger Enable

| Value | Description |
|-------|---|
| 1 | Falling edge of PWMxL will trigger the LEB duration counter |
| 0 | LEB ignores the falling edge of PWMxL |

2.2.12 PWM Generator x Phase Register

Name: PGxPHASE

| | | | | | | | | |
|--------|----------------|-----|-----|-----|-----|-----|-----|-----|
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| | PGxPHASE[15:8] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | PGxPHASE[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Bits 15:0 – PGxPHASE[15:0] PWM Generator x Phase Register

2.2.13 PWM Generator x Duty Cycle Register

Name: PGxDC

| | | | | | | | | |
|--------|-------------|-----|-----|-----|-----|-----|-----|-----|
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| | PGxDC[15:8] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | PGxDC[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Bits 15:0 – PGxDC[15:0] PWM Generator x Duty Cycle Register

Note: Duty cycle values less than 0x0008 should not be used (0x0020 in High-Resolution mode).

2.2.14 PWM Generator x Duty Cycle Adjustment Register

Name: PGxDCA

| | | | | | | | | |
|--------|----|----|----|----|----|----|---|---|
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| | | | | | | | | |
| Access | | | | | | | | |
| Reset | | | | | | | | |

| | | | | | | | | |
|--------|-------------|-----|-----|-----|-----|-----|-----|-----|
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | PGxDCA[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Bits 7:0 – PGxDCA[7:0] PWM Generator x Duty Cycle Adjustment Value
Depending on the state of the selected PCI source, the PGxDCA value will be added to the value in the PGxDC register to create the effective duty cycle. When the PCI source is active, PGxDCA is added. In High-Resolution mode, bits[2:0] are forced to '0'.

2.2.15 PWM Generator x Period Register

Name: PGxPER

| | | | | | | | | |
|--------|--------------|-----|-----|-----|-----|-----|-----|-----|
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| | PGxPER[15:8] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | PGxPER[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Bits 15:0 – PGxPER[15:0] PWM Generator x Period Register
Note: Period values less than 0x0010 should not be used (0x0080 in High-Resolution mode).

2.2.16 PWM Generator x Trigger A Register

Name: PGxTRIGA

| | | | | | | | | |
|--------|----------------|-----|-----|-----|-----|-----|-----|-----|
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| | PGxTRIGA[15:8] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | PGxTRIGA[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Bits 15:0 – PGxTRIGA[15:0] PWM Generator x Trigger A Register
In High-Resolution mode, bits[2:0] are forced to '0'.

2.2.17 PWM Generator x Trigger B Register

Name: PGxTRIGB

| | | | | | | | | |
|--------|----------------|-----|-----|-----|-----|-----|-----|-----|
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| | PGxTRIGB[15:8] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | PGxTRIGB[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Bits 15:0 – PGxTRIGB[15:0] PWM Generator x Trigger B Register
In High-Resolution mode, bits[2:0] are forced to '0'.

2.2.18 PWM Generator x Trigger C Register

Name: PGxTRIGC

| | | | | | | | | |
|--------|----------------|-----|-----|-----|-----|-----|-----|-----|
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| | PGxTRIGC[15:8] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | PGxTRIGC[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Bits 15:0 – PGxTRIGC[15:0] PWM Generator x Trigger C Register
In High-Resolution mode, bits[2:0] are forced to '0'.

2.2.19 PWM Generator x Dead-Time Register Low

Name: PGxDTL

| | | | | | | | | |
|--------|----------|-----|-----------|-----|-----|-----|-----|-----|
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| | | | DTL[13:8] | | | | | |
| Access | | | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | | | 0 | 0 | 0 | 0 | 0 | 0 |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | DTL[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Bits 13:0 – DTL[13:0] PWMxL Dead-Time Delay
Note: The DTL[13:11] bits are not available when HREN (PGxCONL[7]) = 0.

2.2.20 PWM Generator x Dead-Time Register High

Name: PGxDTH

| | | | | | | | | |
|--------|----------|-----|-----------|-----|-----|-----|-----|-----|
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| | | | DTH[13:8] | | | | | |
| Access | | | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | | | 0 | 0 | 0 | 0 | 0 | 0 |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | DTH[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Bits 13:0 – DTH[13:0] PWMxH Dead-Time Delay
Note: The DTH[13:11] bits are not available when HREN (PGxCONL[7]) = 0.

2.2.21 PWM Generator x Capture Register

Name: PGxCAP

| | | | | | | | | |
|--------|--------------|----|----|----|----|----|---|-----|
| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| | PGxCAP[14:7] | | | | | | | |
| Access | R | R | R | R | R | R | R | R |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | PGxCAP[6:0] | | | | | | | |
| Access | R | R | R | R | R | R | R | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Bits 15:1 – PGxCAP[14:0] PGx Time Base Capture bits

Note: A capture event can be manually initiated in software by writing a '1' to PGxCAP[0].

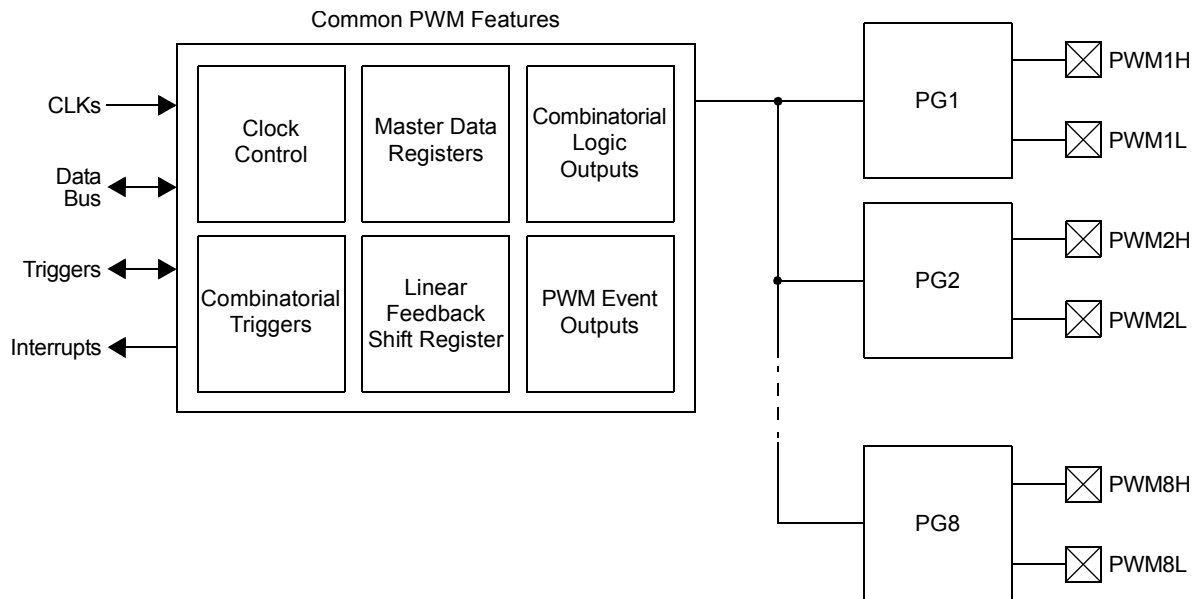
The CAP bit (PGxSTAT[5]) will indicate when a new capture value is available. A read of PGxCAP will automatically clear the CAP bit and allow a new capture event to occur. PGxCAP[1:0] will always read as '0'. In High-Resolution mode, PGxCAP[4:0] will always read as '0'.

Bit 0 – Read/Write

3. Architecture Overview

The PWM module consists of a common set of controls and features and multiple instantiations of PWM Generators (PGx). Each PWM Generator can be independently configured or multiple PWM Generators can be used to achieve complex multiphase systems. PWM Generators can also be used to implement sophisticated triggering, protection and logic functions. A high-level block diagram is shown in [Figure 3-1](#).

Figure 3-1. PWM High-Level Block Diagram



Each PWM Generator behaves as a separate peripheral that can be independently enabled from the other PWM Generators. Each PWM Generator consists of a signal generator and an Output Control block.

The PWM Generators use 'events' to trigger other PWM Generators, ADC conversions and external operations. Each PWM Generator accepts a trigger input and produces a trigger output. The trigger input signals the PWM Generator when to start a new PWM period. The trigger output is generated when the trigger time value is equal to the PWM Generator timer value.

Output Control blocks provide the capability to alter the base PWM signal sent to the output pins and incorporate several functions, including:

- Output mode selection (Complementary, Push-Pull, Independent)
- Dead-time generator
- PWM Control Input (PCI) block
- Leading-Edge Blanking (LEB)
- Override

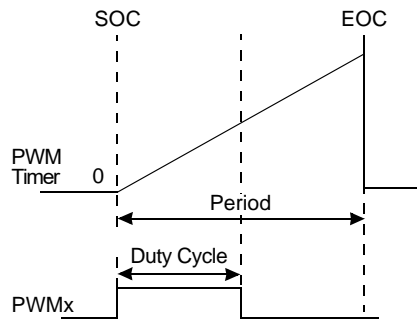
Each PWM Generator Output block is associated with the control of two PWM output pins. Output blocks contain a PWM Control Input (PCI) that can be used for many purposes, including Fault detection, external triggering and interfacing with other peripherals. The LEB block works in conjunction with the PCI block and allows PCI inputs to be ignored during certain periods of the PWM cycle. The Override block determines the PWM output pin states during various types of events, including Faults, current limit and feed-forward control. A block diagram of a single PWM Generator is shown in [Figure 3-2](#).

Functional Reference Manuals (FRM)
© 2017-2024 Microchip Technology Inc. and its
subsidiaries



Some operating modes and output modes use multiple counter cycles to produce a single PWM cycle. Refer to [4.2.2. PWM Modes](#) and [4.2.3. Output Modes](#) for more information.

Figure 3-3. Basic PWM Waveform



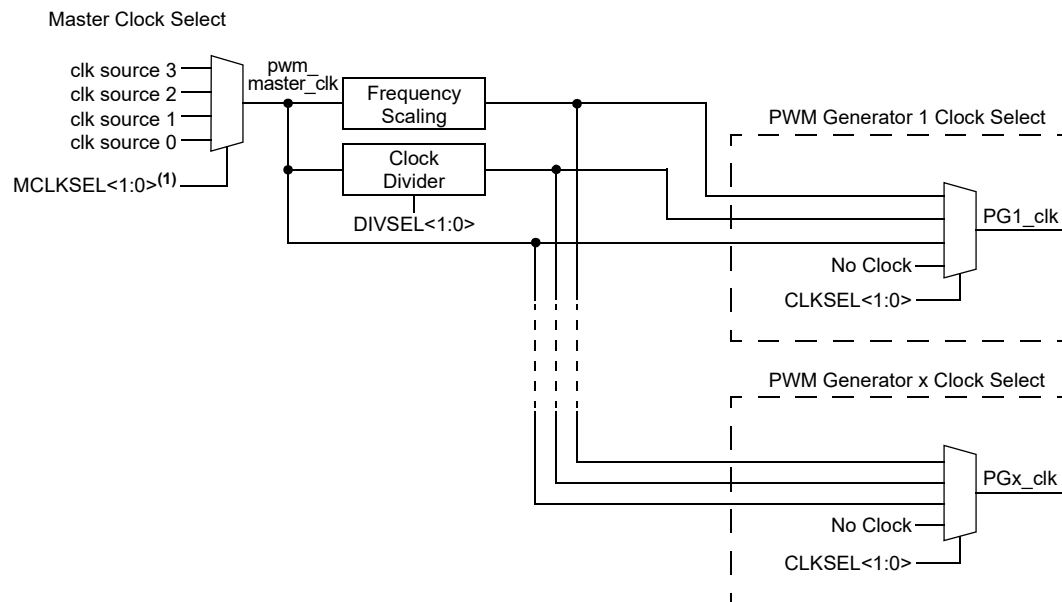
4. Operation

4.1 PWM Clocking

4.1.1 Master Clocking

The PWM module provides several clocking features at the top level of the module. Each PWM Generator can then independently select one of the clock sources, as shown in Figure 4-1. The clock input into the PWM module is selected with the MCLKSEL[1:0] control bits (PCLKCON[1:0]). The available clock inputs are device-dependent; refer to the device data sheet for availability. The CLKSEL[1:0] control bits (PGxCONL[4:3]) are used to select the clock for each PWM Generator instance; see 4.2.1. PWM Generator Clocking for details. Frequency scaling and the clock divider are discussed in 4.3.3. Shared Clocking. The CLKSELx bits need to be changed from the default selection to allow the PWM Generator to function.

Figure 4-1. PWM Generator Clocking



Note:

1. Clock inputs are device-specific. Refer to the device data sheet for availability.

Note: Writing MCLKSEL[1:0] to a non-zero value will request and enable the chosen clock source, whether any PWM Generators are enabled or not. This allows a PLL, for example, to be requested and warmed up before using it as a PWM clock source. For the lowest device power consumption, the MCLKSEL[1:0] bits should be set to the value, '00', if all PWM Generators have been disabled. Changing the MCLKSEL[1:0] or CLKSEL[1:0] bits while ON (PGxCONL[15] = 1) is not recommended.

Note: The CPU and PWM typically run at different clock speeds depending on the application requirements. If the PWM clock speed is equal or slower than the CPU, writes to registers may have delayed behavior. For example, if SWTERM is used to clear a Fault, the instruction may need to be stretched with a REPEAT instruction to ensure the PWM can detect the edge within its clock cycle.

4.1.2 Clocking Equations in Standard Resolution

Some modes of operation utilize multiple period matches to complete one PWM 'cycle'. The following equation provides timing equations for the various operating modes.

Equation: PWM Period Calculation, Standard Resolution

Edge-Aligned, Variable Phase Operating Modes

$$F_{PWM} = \frac{F_{PGx_clk}}{PGxPER + 1}$$

$$PGxPER = \left(\frac{F_{PGx_clk}}{F_{PWM}} \right) - 1$$

Where:

F_{PWM} = Switching Frequency

PWM Period = $1/F_{PWM}$

Center-Aligned Modes, Edge-Aligned and Variable Phase Modes with Push-Pull Output Mode

$$F_{PWM} = \frac{F_{PGx_clk}}{2 \cdot (PGxPER + 1)}$$

$$PGxPER = \left(\frac{F_{PGx_clk}}{2 \cdot F_{PWM}} \right) - 1$$

Center-Aligned Modes with Push-Pull Output Mode

$$F_{PWM} = \frac{F_{PGx_clk}}{4 \cdot (PGxPER + 1)}$$

$$PGxPER = \left(\frac{F_{PGx_clk}}{4 \cdot F_{PWM}} \right) - 1$$

Equation: PWM Duty Cycle, Phase, Trigger and Dead-Time Calculations, Standard Resolution

$$MDC \text{ or } PGxDC(A) = (PGxPER + 1) \cdot \text{Duty Cycle}$$

Where:

Duty Cycle is % between 0 and 100

$$MPHASE \text{ or } PGxPHASE = F_{PGx_clk} \cdot \text{Phase}$$

$$PGxTRIGy = F_{PGx_clk} \cdot \text{Trigger Offset}$$

(y = A, B or C)

$$PGxDTy = F_{PGx_clk} \cdot \text{Dead Time}$$

(y = H or L)

Where:

Phase, Trigger Offset and Dead Time are specified in time units (ms, μ s or ns)

4.1.3 High-Resolution Mode

High-Resolution mode is not available on all devices. Refer to the device-specific data sheet for availability.

The PWM Generators may operate in High-Resolution mode to enhance phase, duty cycle and dead-time resolution up to 250 ps. High-Resolution mode cannot be used with frequency scaling or the clock divider. To enable High-Resolution mode for a given PWM Generator, set the HREN control bit (PGxCONL[7]). The HRRDY status bit (PCLKCON[15]) indicates when the high-resolution circuitry is ready and the HRERR bit (PCLKCON[14]) indicates a clocking error has occurred. When operating in high resolution, Dual PWM mode cannot be used in conjunction with Complementary Output mode.

Note: When using High-Resolution mode, the CLKSEL[1:0] bits (PGxCONL[4:3]) must be set to '01' to select pwm_master_clk directly, and the pwm_master_clk must be configured for the correct frequency. Refer to the PWMx Module Timing Requirements within the **"Electrical Characteristics"** section of the device data sheet for this value. For dsPIC33C devices, the pwm_master_clk frequency must be 500 MHz in High-Resolution mode.

4.1.3.1 Data Registers in High Resolution

When High-Resolution mode is selected, some of the PWM Data registers have limited resolution. For some registers, the Least Significant bits (LSBs) of the data value are forced to '0', regardless of the value written to the register. When configuring the PWM in High-Resolution mode, first set the HREN bit before writing data registers whose function is dependent on High-Resolution mode. High-resolution operational differences are summarized in [Table 4-1](#).

Table 4-1. PWM Data Registers, High-Resolution Mode

| Register | Bits | | | |
|----------------|--------------|---|---|---|
| | 15:3 | 2 | 1 | 0 |
| PGxLEB | | 0 | 0 | 0 |
| PGxPHASE | | | | |
| PGxDC | | | | |
| PGxDCA | | 0 | 0 | 0 |
| PGxPER | | | | |
| PGxTRIGA(B)(C) | (Notes 2, 5) | 0 | 0 | 0 |
| PGxDT | (Note 1) | | | |
| PGxCAP | (Note 3) | | | |
| FSCL | (Note 4) | | | |
| FSMINPER | (Note 4) | | | |
| MPHASE | | | | |
| MDC | | | | |
| MPER | | | | |

Notes:

1. The DTH and DTL register sizes are retained in High-Resolution mode. See the [PGxDTL](#) and [PGxDTH](#) registers for details.
2. Bit 15 of the PGxTRIGy registers selects the counter phase that produces the trigger when operating in Center-Aligned modes.
3. Bits 1 and 0 will read as '0' in Standard Resolution mode. In High-Resolution mode, bits[4:0] will read as '0'.
4. Not used in High-Resolution mode.
5. In Dual PWM mode, the PGxTRIGA and PGxTRIGB registers will be used to set the rising and falling edge of the 2nd PWM signal, and the three LSBs will be utilized.

4.1.3.2 Clocking Equations in High Resolution

Period calculations, when using High-Resolution mode, are shown in the following equations.

Equation: PWM Period Calculation, High-Resolution Mode

Edge-Aligned, Variable Phase Operating Modes

$$F_{PWM} = \frac{8 \cdot F_{PGx_clk}}{(PGxPER + 8)}$$

$$PGxPER = \left(\frac{8 \cdot F_{PGx_clk}}{F_{PWM}} \right) - 8$$

Center-Aligned Modes, Edge-Aligned and Variable Phase Modes with Push-Pull Output Mode

$$F_{PWM} = \frac{4 \cdot F_{PGx_clk}}{(PGxPER + 8)}$$

$$PGxPER = \left(\frac{4 \cdot F_{PGx_clk}}{F_{PWM}} \right) - 8$$

Center-Aligned Modes with Push-Pull Output Mode

$$F_{PWM} = \frac{2 \cdot F_{PGx_clk}}{(PGxPER + 8)}$$

$$PGxPER = \left(\frac{2 \cdot F_{PGx_clk}}{F_{PWM}} \right) - 8$$

Where:

F_{PWM} = Switching Frequency

PWM Period = $1/F_{PWM}$

Equation: PWM Duty Cycle, Phase, Trigger and Dead-Time Calculations, High Resolution

$$MDC \text{ or } PGxDC(A) = (PGxPER + 8) \cdot \text{Duty Cycle}$$

Where:

Duty Cycle is % between 0 and 100

$$MPHASE \text{ or } PGxPHASE = 8 \cdot F_{PGx_clk} \cdot \text{Phase}$$

$$PGxTRIGy = 8 \cdot F_{PGx_clk} \cdot \text{Trigger Offset}$$

(y = A, B or C)

$$PGxDTy = 8 \cdot F_{PGx_clk} \cdot \text{Dead Time}$$

(y = H or L)

Where:

Phase, *Trigger Offset* and *Dead Time* are specified in time units (ms, µs or ns)

4.1.3.3 High-Resolution Period Synchronization

When operating in High-Resolution mode, it is possible for PWM output edges to not be aligned with PGx_clk that the rest of the PWM module operates at. When PGxPER (or MPER) values are not divisible by eight, the period contains a fractional value of PGx_clk. This fractional clock difference can cause other events, including End-of-Cycle (EOC), triggers, etc., to not align with the output edges. The module contains an accumulator circuit to calculate and minimize the offset over long time periods.

If synchronous behavior is desired, it is recommended to use PGxPER values with bits[2:0] equal to '0'.

The fine edge placement circuit itself adds delay to the PWM outputs when compared to the base PWM signal. Using the base PWM signal for gating and synchronization in High-Resolution mode may cause unexpected results for a few fine edge clock cycles in some cases. For example, using the PCI's auto-terminate feature will remove an override condition at EOC and places the PWM outputs back to their existing state. Override is applied after the fine edge placement circuit, as shown in [Figure 4-11](#) and [Figure 4-15](#). Since the EOC event is based on the base PWM signal, the delay through the fine edge circuit may be observed before the next PWM cycle is started. This behavior can be mitigated by using a PHASE offset equal to PGxPER – 8.

In addition to EOC events, using timers operating on the base PWM signal (such as LEB and PGxTRIGy) or other PWM Generators as a source may also be susceptible in some conditions.

4.1.4 Clocking Synchronization

Due to the separate clocking domains of the PWM module and the CPU's system clock, there are inherent synchronization delays associated with SFR reads. This delay is dependent on the relative speeds of the CPU (sys_clk) and the PWM Generator clock (PGx_clk). Typically, the CPU clock will be slower and SFR data can be delayed up to one sys_clk cycle. It is also important to note that each PWM Generator can run at a different speed and this can have an effect on interactions between PWM Generators.

4.1.5 Minimum PWM Period and Pulse Width

The PWM module has some restrictions regarding minimum PWM periods and pulse widths. Depending on the operating mode, the pulse width can also be dependent on PHASE and TRIGy, in addition to duty cycle. The minimum pulse width applies to both active and inactive states; 0% and 100% duty cycles are supported.

[Table 4-2](#) below lists restrictions to period and pulse width.

Table 4-2. Minimum Period and Pulse Width

| Mode | Minimum Period (PGxPER or MPER) | Minimum Active Pulse Width in Counts | Minimum Inactive Pulse Width in Counts |
|---------------------|---------------------------------|--------------------------------------|--|
| Standard Resolution | 0x0020 | 0x0008 | Period – 0x0008 |
| High Resolution | 0x0080 | 0x0020 | Period – 0x0020 |

4.2 PWM Generator (PG) Features

Most of the features and controls of the PWM module are at the PWM Generator level and are controlled using each PWM Generator's SFRs. PWM Generator operation is based on triggers. The PWM Generator must receive a Start-of-Cycle (SOC) trigger signal to generate each PWM cycle. The trigger signal may be generated outside of the PWM Generator or the PWM Generator may be self-triggered. When a PWM Generator reaches the end of a PWM cycle, it produces an End-of-Cycle (EOC) trigger that can be used by other PWM Generators.

If multiple PWM Generators run at different frequencies, the triggers can be synchronized using the PCI Sync block.

4.2.1 PWM Generator Clocking

Each PWM Generator can be clocked independently of one another for maximum flexibility. There are four clock options available selected by the CLKSEL[1:0] bits (PGxCONL[4:3]):

1. No clock (lowest power state).
2. Output of MCLKSEL[1:0].
3. Output of clock divider.
4. Output frequency scaler.

This configuration flexibility allows, for example, one group of PWM Generators to operate at a higher frequency, while another group of PWM Generators operates at a lower frequency. For additional information on clock inputs, see [4.1.1. Master Clocking](#).

Note: Do not change the MCLKSEL[1:0] or CLKSEL[1:0] bits while the PWM Generator is in operation (ON (PGxCONL[15]) = 1).

4.2.2 PWM Modes

The PWM module supports a wide range of PWM modes for both motor control and power supply designs. The following PWM modes are supported:

- Independent Edge PWM mode (default)
- Variable Phase PWM mode
- Independent Edge PWM mode, Dual Output
- Center-Aligned PWM mode
- Double Update Center-Aligned PWM mode
- Dual Edge Center-Aligned PWM mode

The PWM modes are selected by setting the MODSEL[2:0] bits (PGxCONL[2:0]). Some modes utilize multiple time base cycles to complete a single PWM cycle. Refer to the previous equation for specifics on timing.

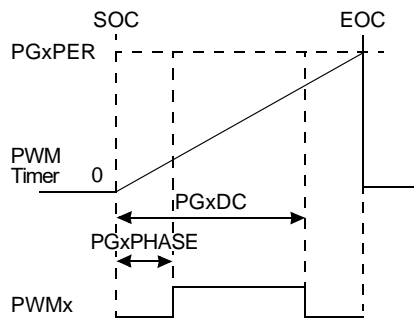
4.2.2.1 Independent Edge PWM Mode

Independent Edge PWM mode is used for many applications and can be used to create edge-aligned PWM signals, as well as PWM signals with arbitrary phase offsets. This mode is the default operating mode of the PWM Generator and is selected when MODSEL[2:0] = 000 (PGxCONL[2:0]). Two Data registers must be written to define the rising and falling edges. The characteristics of the PWM signal are determined by these three SFRs:

- PGxPHASE: Determines the position of the PWM signal rising edge from the start of the timer count cycle
- PGxDC: Determines the position of the PWM signal falling edge from the start of the timer count cycle
- PGxPER: Determines the end of the PWM timer count cycle

A basic Edge-Aligned PWM mode signal is created by setting PGxPHASE = 0. Alternatively, multiple PWM Generators can be synchronized to one another by using the same PGxPHASE value. A constant value equivalent to the PGxPHASE value of other PWM Generators can be used to synchronize multiple PGs. The duty cycle is varied by writing to the PGxDC register. Arbitrary phase PWM signals may be generated by writing to PGxPHASE and PGxDC with the appropriate values. If PGxPHASE = PGxDC, no PWM pulse will be produced. If PGxDC ≥ PGxPER, a 100% duty cycle pulse is produced. If PGxPHASE > PGxDC, the first falling edge of PGxDC compare will be missed as the state is already inactive (low). The PGxPHASE comparison will then generate a rising edge and state will stay active across EOC/SOC boundary, until the next PGxDC is reached. Effective DC is (PGxPER - PGxPHASE) + PGxDC. [Figure 4-2](#) shows the relationship between the control SFRs and the output waveform.

Figure 4-2. Independent Edge PWM Mode



4.2.2.1.1 Multiphase Systems in Independent Mode

Multiple PWM generators can be synchronized to make a multiphase system. There are multiple methods that can be used in different applications and PWM modes. SOC triggers can be shared across groups of four PWM generators (PG1-PG4 or PG5-PG8).

In this example, three PWM generators are set to start and run synchronized:

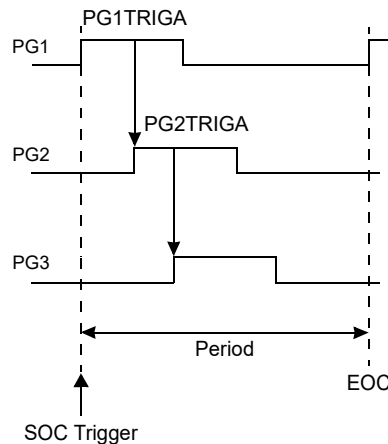
1. Configure all PWM generators for application (ckl, PER, DC, etc).
2. PG1 is self-triggered SOCS = 0b0000.
3. PG2 SOCS trigger is PG1, SOCS = 0b0001.
4. PG3 SOCS trigger is PG2, SOCS = 0b0010.
5. PG2 and PG3 ON bits are set. Cycles will not start until trigger from PWM1.
6. PG1 ON bit is set. All PWM generators start simultaneously.

In this example, a daisy-chain triggering scheme is used to synchronize and create an offset between each PWM generators SOC (see [Figure 4-3](#) for example):

1. Configure all PWM generators for application.
2. PG1 is self-triggered SOCS = 0b0000.
3. PG1TRIGA is set to provide phase offset for PG1 to PG2.
4. PG1 PTGRGSEL = 0001 to select TRIGA as trigger.
5. PG2 SOCS trigger is PG1, SOCS = 0b0001.

6. PG2 TRIGA is set to provide phase offset for PG2 to PG3.
7. PG2 PTGRGSEL = 0001 to select TRIGA as trigger.
8. PG3 SOCS trigger is PG2, SOCS = 0b0010.
9. PG2 and PG3 ON bits are set. Cycles will not start until trigger from PWM1.
10. PG1 ON bit is set.

Figure 4-3. Multiphase System Using TRIGx to Offset PGs



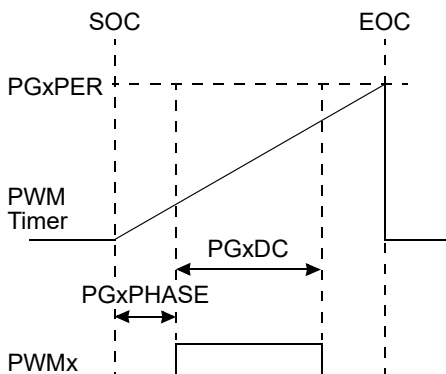
4.2.2.2 Variable Phase PWM Mode

The Variable Phase PWM mode differs from Independent Edge mode in that one register is used to select the phase offset from the Start-of-Cycle and a second register is used to select the width of the pulse. The Variable Phase PWM mode is useful as the PGxDC register is programmed to a constant value, while the PGxPHASE value is modulated. The PWM logic will automatically calculate rising edge and falling edge times to maintain a constant pulse width. Similarly, the user can leave the PGxPHASE register programmed to a constant value to create signals with a constant phase offset and variable duty cycle. The Variable Phase PWM mode is selected when MODSEL[2:0] (PGxCONL[2:0]) = 001. The characteristics of the PWM signal are determined by these three SFRs:

- PGxPHASE: Determines the offset of the PWM signal rising edge from the start of the timer cycle
- PGxDC: Determines the width of the PWM pulse and location of the PWM signal falling edge
- PGxPER: Determines the end of the PWM timer count cycle

Figure 4-4 shows the relationship between the control SFRs and the output waveform.

Figure 4-4. Variable Phase PWM Mode



The Master Duty Cycle SFR (MDC) can also be used to change the duty cycle of all phases with a single register write. An example of a multiphase system is shown in [Figure 4-5](#). Variable Phase mode cannot support active duty cycles across EOC boundaries. Phase + DC ≤ Period to allow for completion of the duty cycle for EOC.

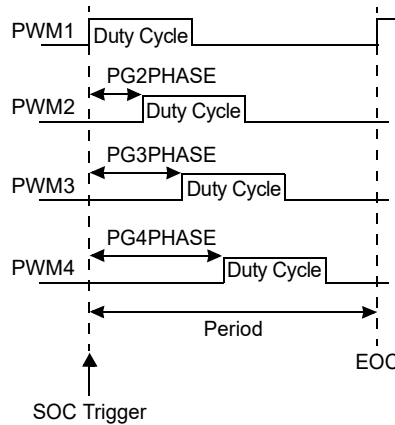
When updating PER in Variable Phase mode, ensure that client PWM generator PER > host PER. The recommended method is client PER = host PER + max PHASE to ensure client PWM generator does not generate its own SOC.

The host will always provide the SOC trigger for the client PWM generators. If host PER is lengthened such that client PER > host PER, a client PWM cycle may be missed as the host SOC trigger arrives when the client cycle is in progress and the trigger is ignored.

4 phase example setup:

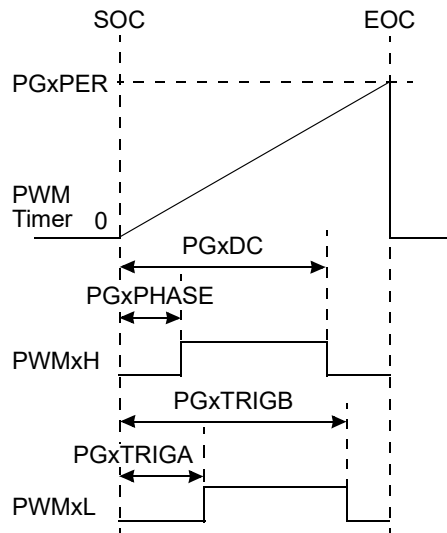
PG1 provides SOC triggers to PG2-PG4

1. Set all PGs to same clock source.
2. Set all PGs in Variable Phase mode, MODSEL = 0b001.
3. Set PG1 for self-trigger, SOCS = 0b0000.
4. Set PG2-PG4 to use PG1's SOC trigger, SOCS = 0b0001.
5. Write initial PER, DC and PHASE to all PGs.
6. Enable outputs, PENx = '1'
7. Set ON = '1' of PG2-PG4. No cycles will start until PG1 sends trigger.
8. Set ON = '1' of PG1. This will start all PG synchronously.

Figure 4-5. Multiphase PWM Example


4.2.2.3 Dual PWM Mode

The Dual PWM mode allows a single PWM Generator to produce two independent pulse widths on the PWMxH and PWMxL output pins. This mode is the equivalent of Independent Edge mode, except that it allows a second PWM pulse to be produced if the Independent Output mode is used. The Dual PWM modes are selected when $\text{MODSEL}[2:0]$ ($\text{PGxCONL}[2:0]$) = 010. The PGxTRIGA and PGxTRIGB registers function as a second set of PGxPHASE and PGxDC registers to allow control of a second duty cycle generator. [Figure 4-6](#) shows the relationship between the control SFRs and the output waveform. Dual PWM mode cannot be used in conjunction with Complementary Output mode when operating in high resolution ($\text{HREN} = 1$).

Figure 4-6. Dual PWM Mode


The PGxTRIGA and PGxTRIGB event output signals continue to operate normally in this mode, and can still be used as phase offset triggers for other PWM Generators, ADC triggers, etc. If an independent trigger is needed, the PGxTRIGC register can be used. For additional information on ADC triggers, see [4.2.10. ADC Triggers](#).

Since the PWM signals produced on the PWMxH and PWMxL pins are produced from the same PWM generator, they will be equally affected by any PWM Control Input (PCI) signals that are enabled. The PWMxH and PWMxL pins will be driven to the states defined in the PGxIOCON register. Therefore,

it is important that the two PWM outputs be used for related application functions if the PCI signals are to be used.

4.2.2.4 Center-Aligned PWM Mode

Center-Aligned PWM mode signals avoid coincident rising or falling edges between PWM Generators when the duty cycles are different, reducing excessive current ripple and filtering requirements in power inverter applications.

The PWM pulse maintains symmetry around the end of the first timer cycle and the beginning of the second cycle. If the duty cycle of the PWM signal is increased, then the position of the rising edge and the falling edge will change to maintain this symmetry. Center-Aligned PWM mode is selected when $\text{MODSEL}[2:0]$ ($\text{PGxCONL}[2:0]$) = 100. Center-Aligned PWM operating mode uses two timer cycles to produce a single pulse. The characteristics of the PWM signal are defined by two SFRs:

- PGxDC: Determines the width of the PWM pulse from the center of the two timer cycles
- PGxPER: Determines the end of the PWM timer count cycle

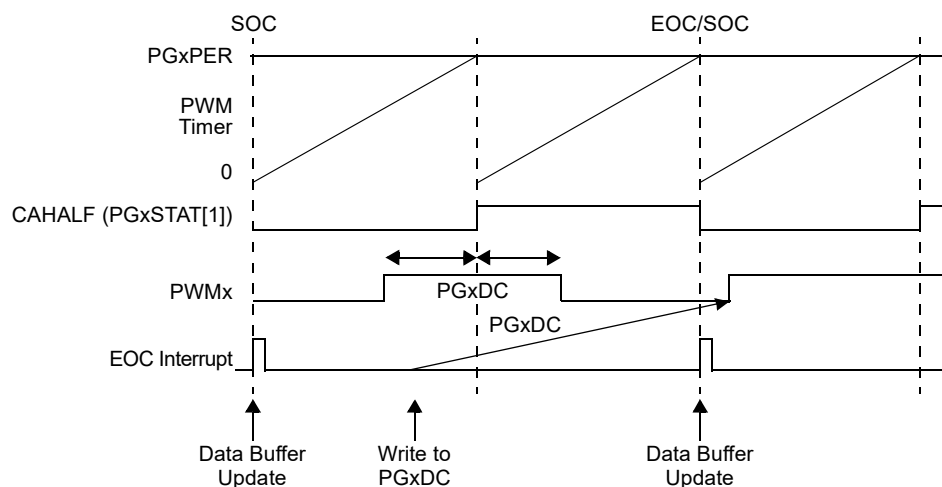
The falling edge occurs when the PWM Generator Timer = PGxDC, and the rising edge occurs when the PG Timer = PGxPER - PGxDC + 1. An offset of 1 is added to the rising edge calculation to produce symmetry between the two timer count cycles. A PGxDC value of '1', for example, will produce a pulse that is two cycles in duration. When PGxDC = 0, there are restrictions on the maximum PER value. In standard resolution, the max value is 0xFFFE. In high resolution, the max value is 0xFFF0. If PER value is above this, a short pulse at EOC may occur.

Center-aligned mode data restrictions and behavior:

- PHASE must be > 0. If not, rising edge will not occur.
- PHASE must be < PER. If not, falling edge will not occur.
- DC must be > 0. If not, falling edge will not occur.
- DC must be < PER. If not, falling edge will not occur, 0% DC.
- IF PER=DC, DC is 100%.

The timer cycle is tracked using the CAHALF status bit (PGxSTAT[1]), and is read as '0' on the first half of cycle and '1' on the second half. Buffer updates to the duty cycle or period are allowed only at the beginning of the first timer cycle. The End-of-Cycle (EOC) interrupt is generated only after the completion of both period cycles. [Figure 4-7](#) shows the relationship between the control SFRs and the output waveform. See [4.2.12. Data Buffering](#) for additional information on data buffering.

Figure 4-7. Center-Aligned PWM Mode

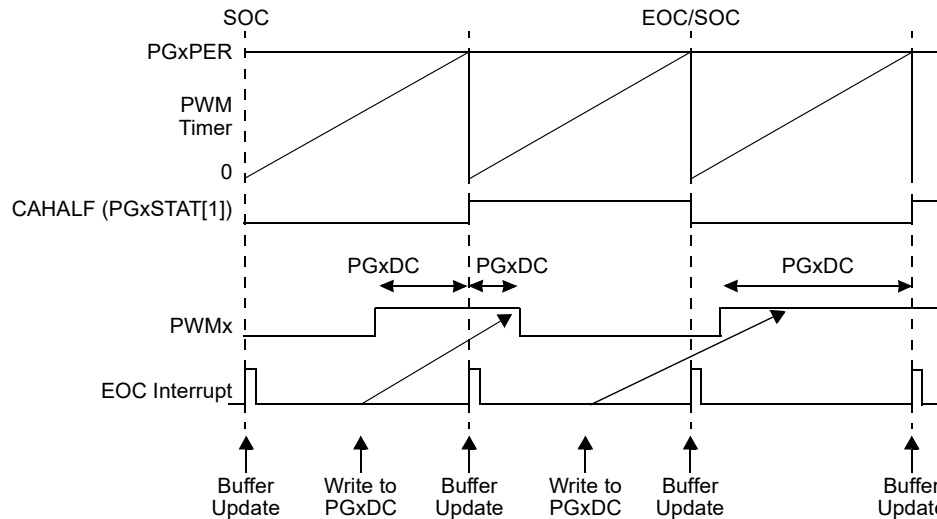


4.2.2.5 Double Update Center-Aligned PWM Mode

Double Update Center-Aligned PWM mode works identically to Center-Aligned PWM mode, except that two interrupts and two data buffer updates occur per PWM cycle. This mode is useful when

the user wants to decrease the latency of a control loop response. Note that this will eliminate the symmetrical nature of the Center-Aligned PWM mode pulse, since the rising edge and falling edge of the pulse can be controlled independently. Double Update Center-Aligned PWM mode is selected when $\text{MODSEL}[2:0]$ ($\text{PGxCONL}[2:0]$) = 101. Figure 4-8 shows the relationship between the control SFRs and the output waveform.

Figure 4-8. Double Update Center-Aligned Mode

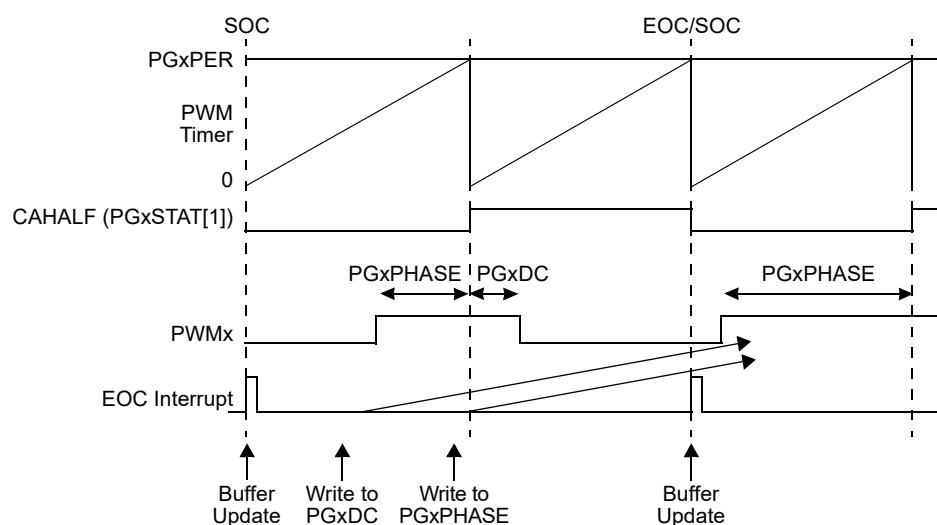
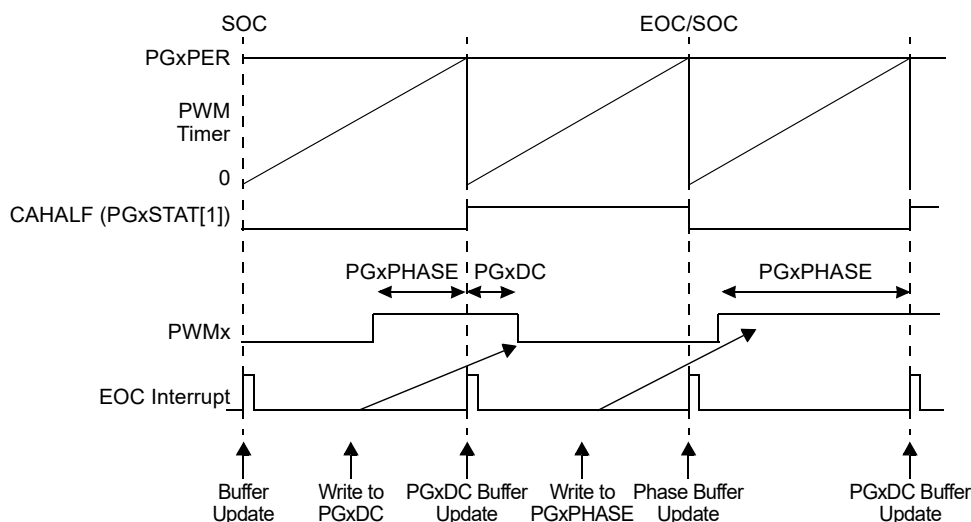


4.2.2.6 Dual Edge Center-Aligned PWM Mode

Dual Edge Center-Aligned PWM mode works identically to Double Update Center-Aligned PWM mode, but it allows the rising edge time and the falling edge time to be controlled via separate Data registers. This mode gives the user the most flexibility in the adjustment of the center-aligned pulse, yet it offers a lower frequency of interrupt events. Note that this will eliminate the symmetrical nature of the center-aligned PWM pulse unless the $\text{PGxPHASE} = \text{PGxDC}$.

- **PGxPHASE:** Determines the rising edge time pulse from the center of the two timer cycles (MPHASE is not supported)
- **PGxDC:** Determines the falling edge time pulse from the center of the two timer cycles

Both Single and Double Data Buffer Update modes are available within the Dual Edge Center-Aligned PWM mode. Single Update mode is selected when $\text{MODSEL}[2:0] = 110$ and Double Update mode is selected when $\text{MODSEL}[2:0] = 111$. In Single Update mode, the user may write a new PGxPHASE and PGxDC value at any time during the cycle to be used on the next center-aligned cycle. In Double Update mode, an interrupt event and a Data register update occur every timer cycle. This provides user software the opportunity to modify the PGxDC value for the falling edge event and PGxPHASE for the rising edge event. User software must check the state of the CAHALF bit ($\text{PGxSTAT}[1]$) to determine the appropriate register to update. If $\text{CAHALF} = 0$ (first half of the center-aligned cycle), the user software should write to the PGxDC register. If $\text{CAHALF} = 1$ (second half of cycle), the user software should write to the PGxPHASE register. Figure 4-9 and Figure 4-10 show the relationship between the control SFRs and the output waveform.

Figure 4-9. Dual Edge Center-Aligned PWM Mode (MODSEL[2:0] = 110)

Figure 4-10. Dual Edge Center-Aligned PWM Mode (MODSEL[2:0] = 111)


4.2.3 Output Modes

Each PWM Generator can be programmed to one of three output modes to control the behavior of the PWMxH and PWMxL pins. The output mode selection is independent of the PWM mode. The output modes are:

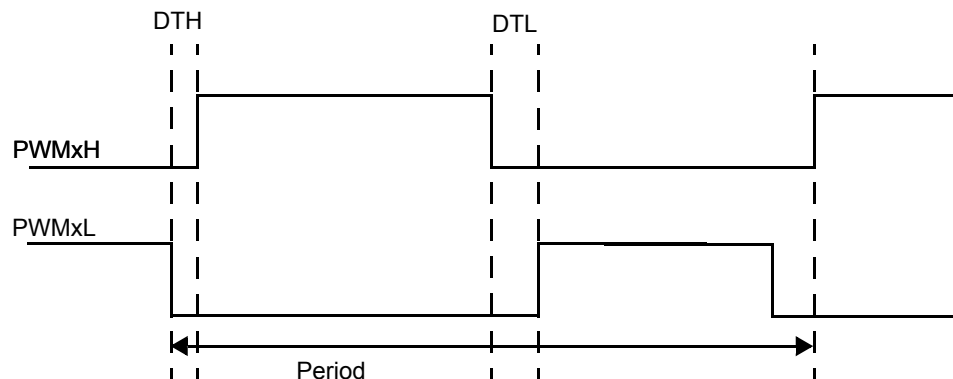
- Complementary Output mode (default)
- Independent Output mode
- Push-Pull Output mode

4.2.3.1 Complementary Output Mode

In Complementary Output mode, both the PWMxH and PWMxL signals are never active at the same time. A dead-time switching delay may be inserted between the two signals and is controlled by the

PGxDT register. Complementary Output mode is selected when PMOD[1:0] (PGxIOCONH[5:4]) = 00. For more information on dead time, see [4.2.6. Dead Time](#).

Figure 4-11. PWMxH/PWMxL Rising and Falling Edges Due to Dead Time



OUTPUT OVERRIDE BEHAVIOR IN COMPLEMENTARY OUTPUT MODE

The PWMxH and PWMxL outputs may be controlled by external hardware signals or by software overrides. The output pins are restricted from being placed in a state which violates the complementary output relationship or in a state which violates dead-time insertion delays. An output pin may be driven inactive immediately as a result of a hardware event. However, a pin will not be driven active until the programmed dead-time delay has expired. The following hardware and software override states are programmed using the following:

- PCI Fault event, FLTDAT[1:0] (PGxIOCONL[7:6])
- PCI current limit event, CLDAT[1:0] (PGxIOCONL[5:4])
- PCI feed-forward event, FFDAT[1:0] (PGxIOCONL[3:2])
- Debugger Halt, DBDAT[1:0] (PGxIOCONL[1:0])
- Software override, OVRENH (PGxIOCONL[13]) and OVRENL (PGxIOCONL[12])
- Swap of PWMxH and PWMxL pins, SWAP (PGxIOCONL[14])

[Figure 4-12](#) shows the signal chain for override behavior in Complementary mode. The SWAP control is applied first and is therefore overridden by all other controls. Next, the request to drive a pin active is applied before dead time, so dead time is still applied to the output; after which, the dead-time generator is requested to drive a pin inactive. This arrangement allows the inactive state to take precedence over SWAP and an active request. Finally, the polarity control is applied to the pin.

The PCI overrides operate on a priority scheme; see [4.2.5.2. Output Control PCI Blocks](#) for more information.

Figure 4-12. Override and SWAP Signal Flow, Complementary Mode

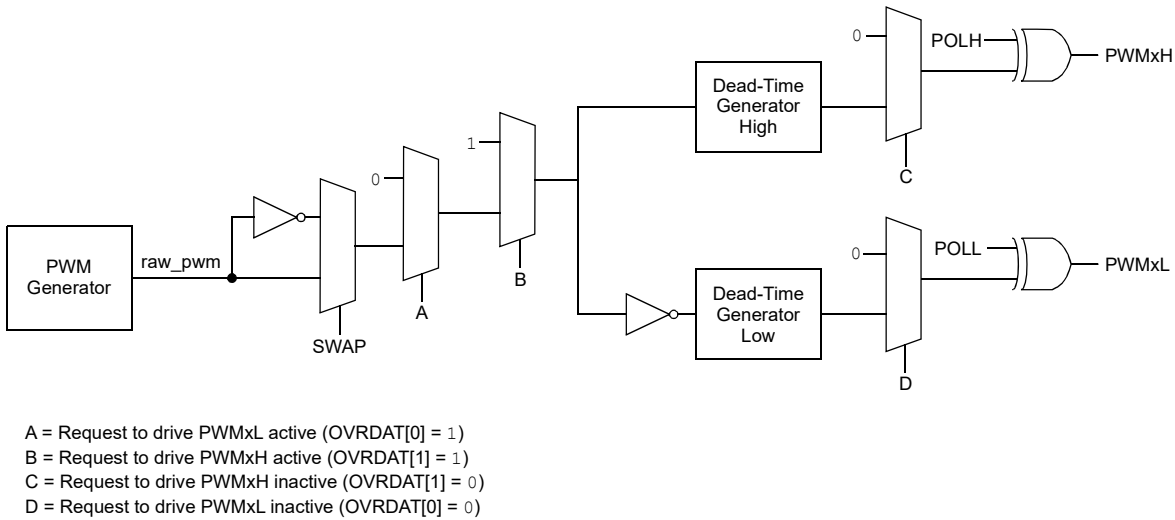


Table 4-3 shows the rules for pin override conditions. The active state is a '1' on the output pin and the inactive state is a '0'. An 'x' denotes a 'don't care' input; ~PWM indicates the complementary output of the PWM Generator's output.

Table 4-3. Override Behavior in Complementary Output Mode

| Source | SWAP | OVRENH | OVRENL | OVRDAT[1:0] | FFDAT[1:0] | CLDAT[1:0] | FLTDAT[1:0] | DBGDAT[1:0] | PWMxH Signal | PWMxL Signal |
|---|------|--------|--------|-------------|------------|------------|-------------|-------------|--------------|--------------|
| Debug Override | | | | | | | | | | |
| DEBUG | x | x | x | xx | xx | xx | xx | 00 | Inactive | Inactive |
| | | | | | | | | 01 | Inactive | Active |
| | | | | | | | | 1x | Active | Inactive |
| Fault Override – Debug Override must be Inactive | | | | | | | | | | |
| PCI FLT | x | x | x | xx | xx | xx | 00 | xx | Inactive | Inactive |
| | | | | | | | 01 | | Inactive | Active |
| | | | | | | | 1x | | Active | Inactive |
| Current Limit Override – Fault and Debug Overrides must be Inactive | | | | | | | | | | |
| PCI CL | x | x | x | xx | xx | 00 | xx | xx | Inactive | Inactive |
| | | | | | | 01 | | Inactive | Active | |
| | | | | | | 1x | | Active | Inactive | |
| Feed-Forward Override – Software, Current Limit, Fault and Debug Overrides must be Inactive | | | | | | | | | | |
| PCI FF | x | 0 | 0 | xx | 00 | xx | xx | xx | Inactive | Inactive |
| | | | | | 01 | | Inactive | Active | | |
| | | | | | 1x | | Active | Inactive | | |
| Software Override – Current Limit, Fault and Debug Overrides must be Inactive | | | | | | | | | | |
| Software Override | 0 | 0 | 1 | x0 | xx | xx | xx | xx | PWM | Inactive |
| | | | | | | | | | Inactive | ~PWM |
| | 1 | 0 | 0 | 00 | | | | | ~PWM | PWM |
| | | | | | | | | | ~PWM | Inactive |
| | x | 0 | 1 | x0 | | | | | ~PWM | Inactive |
| | | | | | | | | | Inactive | Active |
| | | | | | | | | | Inactive | Inactive |
| | | | | | | | | | Inactive | Active |

OUTPUT BEHAVIOR AT START-UP IN COMPLEMENTARY MODE

When the PWM is initialized and the ON bit is set, the outputs immediately go to a Complementary state. There is an output delay as the signals propagate through the PWM logic. This causes the start of the active duty cycle to appear delayed, with the PWMxL output transitioning to an Inactive state (pin high) for four master_pwm_clk cycles (eight cycles in high resolution). Once active duty cycle starts, the PWMx pins will behave as stated in [Table 4-3](#).

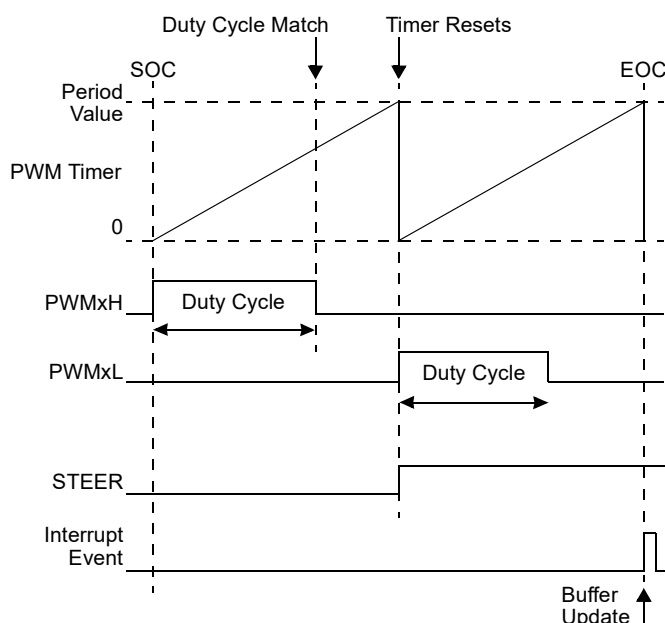
4.2.3.2 Independent Output Mode

In Independent Output mode, the output of the PWM Generator is connected to both the PWMxH and PWMxL pins. In most application scenarios, only the PWMxH or PWMxL pin would be enabled. The other pin remains available for GPIO or other peripheral functions. If the Dual PWM mode is selected, the PWM Generator will produce independent pulse widths on PWMxH and PWMxL, as described in [4.2.2.3. Dual PWM Mode](#). No dead-time switching delay is used in Independent Output mode. No restrictions exist for the states of the PWMxH and PWMxL pins; they can be controlled by external hardware signals or by software overrides. Independent Output mode is selected when $PMOD[1:0] (PGxIOCONH[5:4]) = 01$.

4.2.3.3 Push-Pull Output Mode

The Push-Pull Output mode is similar to Independent Edge mode; however, the PWM cycle, as defined by the MODSEL[2:0] bits, is repeated twice each time a SOC trigger is received. The EOC trigger event and updates from Data registers are held off until the end of the second PWM cycle. [Figure 4-13](#) shows the 2nd cycle that is invoked when using Push Pull Output mode.

Figure 4-13. Push-Pull PWM



Note: Operating the PWM in Push-Pull mode will double the period for a complete cycle, as there are two timer matches per cycle. If PGxTRIGy timers are used for event timing, the STEER signal can be used to gate application timing.

Push-Pull PWM mode is typically used in transformer coupled circuits to ensure that no net DC currents flow through the transformer. Push-Pull mode ensures that the same duty cycle PWM pulse is applied to the transformer windings in alternate directions. The phase of the push-pull count period can be determined by reading the STEER status bit (PGxSTAT[2]). If STEER = 0, the PWM Generator is generating the first PWM pulse. If STEER = 1, the PWM Generator is generating the second PWM pulse.

Since dead time is not available in Push-Pull mode, delays can be emulated in the Push-Pull Output mode by introducing a small phase offset with the PGxPHASE register. Similarly, the maximum duty cycle may be limited in software to avoid a pulse that ends too close to the start of the next PWM cycle.

PUSH-PULL OPERATION WITH CENTER-ALIGNED MODES

When the PWM Generator is operated in one of the two Center-Aligned modes, and the Push-Pull mode is selected, a complete PWM cycle will comprise four time base cycles.

Note: High-Resolution mode should not be used in Push-Pull operation with Center-Aligned modes.

Figure 4-14 shows the operation of the module with Push-Pull Output mode and Center-Aligned PWM mode. This combination of modes limits PWM buffer updates and interrupt events to every 4th time base cycle. Therefore, the same pulse is produced on the PWMxH and PWMxL pins before any changes to the duty cycle are allowed. Similar interrupt behavior also occurs when Dual Edge Center-Aligned mode (one update per cycle) is selected (MODSEL[2:0] = 110).

Figure 4-14. Push-Pull PWM: Center-Aligned Mode, Dual Edge Center-Aligned Mode with One Update per Cycle (MODSEL[2:0] = 110)

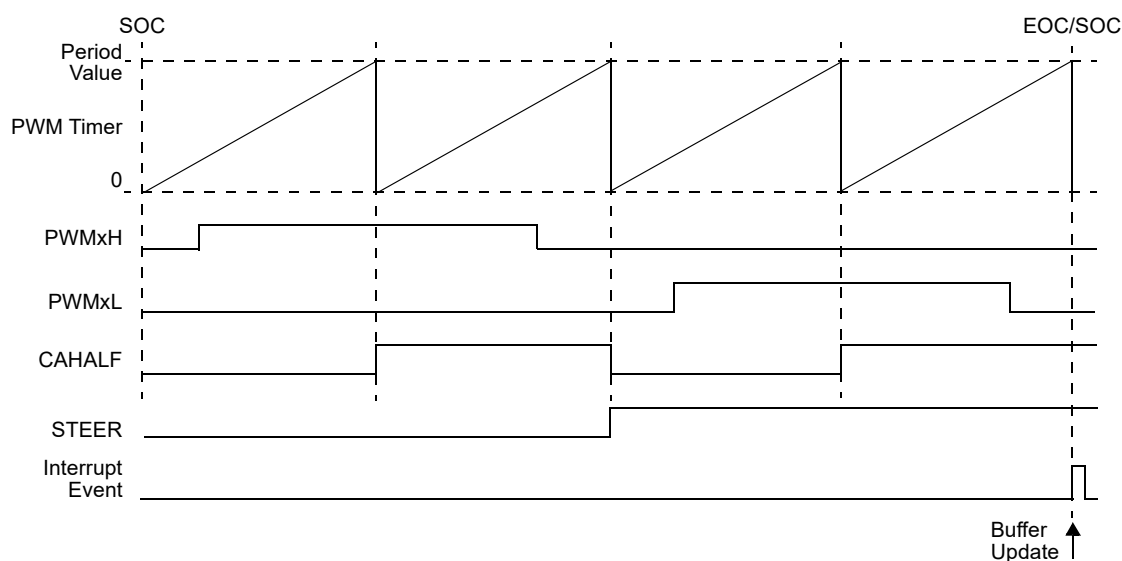
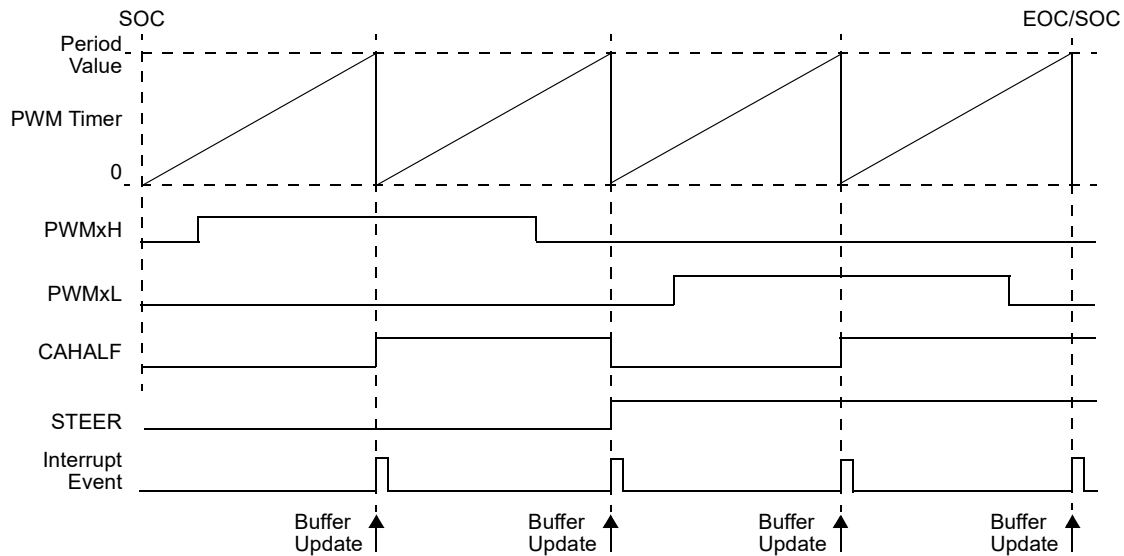


Figure 4-15 shows the operation of the module with Push-Pull Output mode and Dual Edge Center-Aligned PWM mode (two updates per cycle, MODSEL[2:0] = 111) or Double Update Center-Aligned mode. This combination of modes allows a buffer update and interrupt event on every time base cycle. This operating configuration does not attempt to maintain symmetrical pulses on the PWMxH and PWMxL outputs, which is a requirement for many push-pull applications. User software can change the edge times of the center-aligned pulses after every edge event, which minimizes control loop latency.

Figure 4-15. Push-Pull PWM: Double Update Center-Aligned Mode, Dual Edge Center-Aligned Mode with Four Updates per Cycle (MODESEL[2:0] = 111)



4.2.3.4 Output Override in Push-Pull and Independent Modes

When operating in Push-Pull or Independent Output modes, there is no logic that enforces a complementary relationship between the PWMxH and PWMxL signals. It is possible to drive both pins to an active state with a software or hardware (PCI) override. This output state may or may not be desirable, depending on the external circuit that is controlled by the PWM Generator. Therefore, care must be taken when selecting the pin override values. Many push-pull applications require an equal pulse on both the PWMxH and PWMxL outputs to avoid a DC component. If the application is sensitive to this, perform software overrides after two complete timer cycles have taken place. Hardware PCI overrides should be configured to take effect after both timer cycles in the push-pull sequence have occurred. This can be accomplished by using the STEER signal, routed through the Event logic to a pin, which can then be selected as an input to the PCI block.

Figure 4-16. Override and SWAP Signal Flow, Push-Pull Output Mode

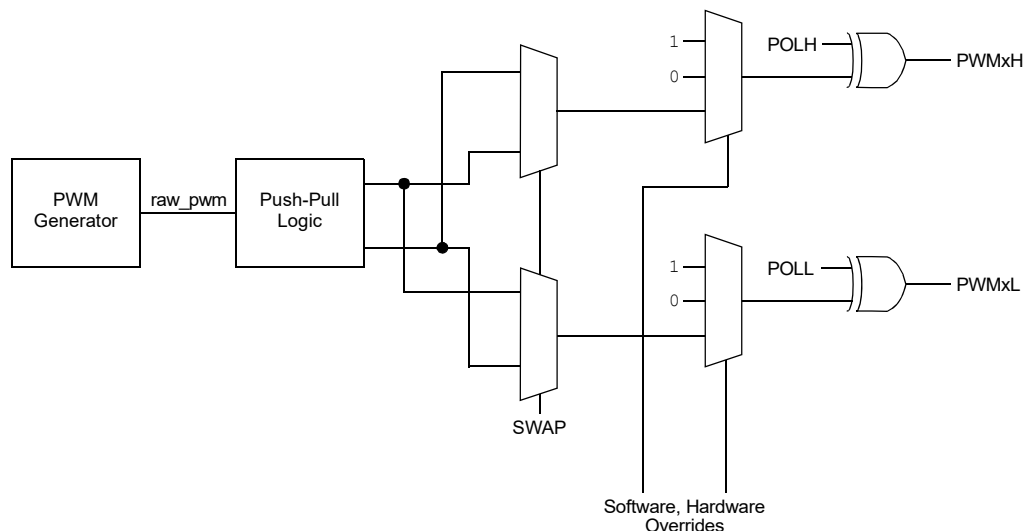


Table 4-4 shows the rules for pin override conditions. The active state is a '1' on the output pin and the inactive state is a '0'. An 'x' denotes a 'don't care' input; ~PWM indicates the complementary output of the PWM Generator's output.

Table 4-4. Override and SWAP Behavior in Push-Pull, Independent Output Modes

| Source | SWAP | OVRENH | OVRENH | OVRDAT[1:0] | FFDAT[1:0] | CLDAT[1:0] | FLTDAT[1:0] | DBGDAT[1:0] | PWMxH Pin State | PWMxL Pin State |
|---|------|--------|--------|-------------|------------|------------|-------------|-------------|--------------------|--------------------|
| Debug Override | | | | | | | | | | |
| DEBUG | x | x | x | xx | xx | xx | xx | 00 | Inactive | Inactive |
| | | | | | | | | 01 | Inactive | Active |
| | | | | | | | | 10 | Active | Inactive |
| | | | | | | | | 11 | Active | Active |
| Fault Override – Debug Override Must be Inactive | | | | | | | | | | |
| PCI FLT | x | x | x | xx | xx | xx | 00 | xx | Inactive | Inactive |
| | | | | | | | 01 | | Inactive | Active |
| | | | | | | | 10 | | Active | Inactive |
| | | | | | | | 11 | | Active | Active |
| Current Limit Override – Fault and Debug Overrides Must be Inactive | | | | | | | | | | |
| PCI CL | x | x | x | xx | xx | 00 | xx | xx | Inactive | Inactive |
| | | | | | | 01 | | | Inactive | Active |
| | | | | | | 10 | | | Active | Inactive |
| | | | | | | 11 | | | Active | Active |
| Feed-Forward Override – Software, Current Limit, Fault and Debug Overrides Must be Inactive | | | | | | | | | | |
| PCI FF | x | 0 | 0 | xx | 00 | xx | xx | xx | Inactive | Inactive |
| | | | | | 01 | | | | Inactive | Active |
| | | | | | 10 | | | | Active | Inactive |
| | | | | | 11 | | | | Active | Active |
| Software Override – Current Limit, Fault and Debug Overrides Must be Inactive | | | | | | | | | | |

|continued | | | | | | | | | | | |
|-------------------|------|--------|--------|--------|-------------|------------|------------|-------------|-------------|-----------------|-----------------|
| Source | SWAP | OVRENH | OVRENH | OVRENH | OVRDAT[1:0] | FFDAT[1:0] | CLDAT[1:0] | FLTDAT[1:0] | DBGDAT[1:0] | PWMxH Pin State | PWMxL Pin State |
| Software Override | 0 | 0 | 0 | 1 | x0 | xx | xx | xx | xx | PWMH | Inactive |
| | | 0 | 0 | 1 | x1 | | | | | PWMH | Active |
| | | 1 | 1 | 0 | 0x | | | | | Inactive | PWML |
| | | 1 | 1 | 0 | 1x | | | | | Active | PWML |
| | 1 | 0 | 0 | 1 | x0 | | | | | PWML | Inactive |
| | | 0 | 0 | 1 | x1 | | | | | PWML | Active |
| | | 1 | 1 | 0 | 0x | | | | | Inactive | PWMH |
| | | 1 | 1 | 0 | 1x | | | | | Active | PWMH |
| | x | 1 | 1 | 1 | 00 | | | | | Inactive | Inactive |
| | | | | | 01 | | | | | Inactive | Active |
| | | | | | 10 | | | | | Active | Inactive |
| | | | | | 11 | | | | | Active | Active |

4.2.4 PWM Generator Triggers

Each PWM Generator must receive a Start-of-Cycle (SOC) trigger to begin a PWM cycle. The trigger signal can be supplied by the PWM Generator itself (self-triggered) or another external trigger source. The SOC trigger can be generated from three sources:

- An internal source operating from the same clock source selected by the [SOCS\[3:0\]](#) bits (PGxCONH[3:0])
- An external source selected by the PWM Control Input (PCI) Sync block
- A software trigger request, write to TRSET (PGxSTAT[7])

Any of the PWM Generators may act as a 'host' by providing the trigger for other PWM Generators. Many trigger configurations may be achieved, including:

- Multiple PWM outputs with independent periods (no synchronization between PWM Generators)
- Multiple PWM outputs with synchronized periods (synchronized operation)
- Multiple PWM outputs with offset phase relationships (triggered operation)

Synchronized operation is achieved by setting a (client) PWM Generator's SOCSx bits to that of another (host) PWM Generator, with the host's PGTRGSEL[2:0] bits (PGxEVT[2:0]) set to '000'. This selects the host's EOC to be used as the client's SOC trigger. When using PGxTRIGy to phase offset PWM generators from one another, a synchronization delay of up to five pwm_master_clk cycles are present. If the TRIG value is '0', there will still be an offset. The TRIGy value may need to be compensated for the application.

Triggered operation is achieved in a similar way, but with the host's [PGTRGSEL\[2:0\]](#) bits (PGxEVTL[2:0]) set to select one of the PGxTRIGy counters (y = A, B or C). The value specified in the host's TRIGy register defines the client's trigger offset from that of the host's SOC.

The SOCS[3:0] control bits have two special selections. When SOCS[3:0] = 0000, the PWM Generator is internally triggered. When SOCS[3:0] = 1111, no trigger source is selected. This selection is useful when the PWM Generator will be triggered only by software using the TRSET bit or from a source connected to the PCI Sync block. In this mode, the next PWM cycle will not start until another trigger is received. The sources available to the PCI Sync block include external signals, such as comparator events, device I/O pins, etc. One of the important functions of the PCI Sync block is to synchronize external input signals into the clock domain of the PWM Generator. See [4.2.5. PWM Control Input \(PCI\) Logic Blocks](#) for more information on the PCI block. The PCI Sync can be OR'd into any of the other Start-of-Cycle inputs (SOCS[3:0]) as long as the block is enabled. The trigger output of another PWM Generator can also be used as a SOC event. See [4.2.11. Event Selection Block](#) for configuration options.

4.2.4.1 Trigger Operation

A PWM cycle starts only when it receives a SOC trigger. When the time base reaches its end, the PWM cycle completes and the PWM Generator exits a triggered state. The PWM Generator must be retriggered to continue operation, which can be done in several ways.

- The PWM Generator is self-triggered (SOCS[3:0] = 0000, default)
- A new trigger pulse is received which is coincident with the end of the PWM cycle event. This can be achieved by multiple PWM Generators having matching PGxPER values, PWM modes and PWM Output modes

The TRIG status bit (PGxSTAT[0]) indicates whether the PWM Generator is in a triggered state.

The EOC signal is the default input to the SOC trigger selection multiplexer, which allows self-triggering. The EOC trigger is generated when the PWM Generator has finished a PWM cycle. It is also generated when the ON bit (PGxCONL[15]) associated with the PWM Generator has been set. This allows all PWM Generators receiving the EOC signal to start in unison when the ON bit of the

host PWM Generator has been set. The ON bit of the other client PWM Generators needs to be set previously to achieve a synchronous start.

4.2.4.2 Triggering Modes

The PWM Generator provides two types of Triggering modes that determine how the SOC trigger is used. The Triggering modes are:

- Single Trigger mode (default)
- Retriggerable mode

The Trigger mode is selected using the TRGMOD control bits (PGxCONH[7]).

SINGLE TRIGGER MODE

Single Trigger (Single Shot) mode is used when a PWM Generator timer is started at the same time as (or a time that is offset from) another PWM Generator timer. This mode is also useful for creating a single PWM pulse or creating a single delay based on an external event. If a timer cycle is currently in progress, any incoming SOC trigger pulses will be ignored. The entire timer cycle must complete before another SOC trigger can restart the timer. Single Trigger mode is selected when TRGMOD = 0.

RETRIGGERABLE MODE

Retriggerable mode is different from Single Trigger mode in that a PWM cycle may be restarted before the end of a cycle that is already in progress. If this is done, the count will be reset when a new trigger is received and the current PWM cycle has stopped. This mode can be especially useful when a PWM Generator is synchronized to an external, off-chip source that operates from a different clock source. The Retriggerable mode is selected when TRGMOD[1:0] = 01. The TRGCNT[2:0] bits (PGxCONL[10:8]) can be written to produce a multiple cycle PWM event. If using software trigger TRSET, subsequent writes of TRSET should wait for the TRIG bit to clear, indicating that the requested number of cycle is complete.

4.2.4.3 Burst Mode

In some applications, it is desirable to have the PWM cycle repeat a certain number of times after the PWM Generator is triggered. The TRGCNT[2:0] control bits (PGxCON[10:8]) select the number of times the PWM cycle will be repeated after a trigger event. Any incoming triggers will be ignored until all PWM cycles are completed. Use of Retriggerable mode in conjunction with Burst mode is not recommended. For very small period values, extra cycles beyond the number defined in TRGCNT may result.

4.2.4.4 Trigger Registers in Center-Aligned Mode

When using any of the Center-Aligned modes, bit 15 of the PGxTRIGA, PGxTRIGB and PGxTRIGC Trigger registers specifies whether the trigger compare time occurs in the first phase (CAHALF = 0) or the second phase (CAHALF = 1). User software should limit the maximum time base count period to 0x7FFF (15 bits) in Center-Aligned PWM mode to ensure proper operation of the Trigger registers in all Center-Aligned modes. Otherwise, two trigger events may be generated in the center-aligned count phase depending on the values of the programmed period and Trigger register value. In some situations, it is desirable to have a trigger event occur in both count phases; this can be accomplished by programming two Trigger registers.

4.2.4.5 Behavior of PWM Generator Output Signal Across PWM Cycle Boundaries

During normal operation, the PWM Data registers will be programmed to create a PWM pulse which begins and terminates within a single PWM cycle. It is possible to write values to the PWM Data registers which will result in a 100% duty cycle output or produce an active output that spans across PWM cycles. The PWM Generator must remain in a continuously triggered state in order for the PWM output to remain active across PWM cycles. To remain triggered, the PWM Generator trigger input signal must be coincident with the EOC output signal. This will happen automatically when:

- The PWM Generator is self-triggered (SOCS[3:0] = 0000)

- The local PGxPER value is set to the same value as the external PWM Generator that is providing the trigger signal

If the PWM Generator trigger input signal does not occur at or before the EOC output signal, then the PWM Generator will exit the triggered state and the PWM Generator output will be driven inactive.

4.2.5 PWM Control Input (PCI) Logic Blocks

The PWM Control Input (PCI) Logic blocks are flexible state machines that can be used for a wide variety of purposes. The PCI blocks condition input signals and provide output signals used to trigger, gate and override the PWM outputs. The PCI also allows interfacing between PWM Generators, both with each other and with external input signals. The PCI blocks can be used to implement output control and triggering algorithms in hardware instead of using software resources. There are four identical PCI blocks available for each PWM Generator. The PCI blocks are:

- Fault
- Current Limit
- Feed-Forward
- Sync

The names of the PCI block do not limit their usage; they are given unique names to designate the priority levels. The Sync PCI block is intended for triggering, specifically from external events, including other PWM Generators. The Fault, Current Limit and Feed-Forward PCI blocks are used to control the PWM output from external signals and other peripherals. The output state of the PWM pins can be independently configured to a predefined state for each PCI block and operates in a priority scheme if more than one PCI block requests control over the PWM outputs. Each PCI block has its own control register, PGxyPCI (with y = F, CL, FF or S), that contains the control bit associated with its operation. The PCI logic has three major components used to create logic functions:

- Inputs:
 - PCI source
 - PCI source qualifier, used to gate the PCI source signal
 - Terminator event, used to stop the 'PCI_active' output signal
 - Terminator qualifier event, used to gate the terminator event
- Acceptance logic
- Output and bypass function

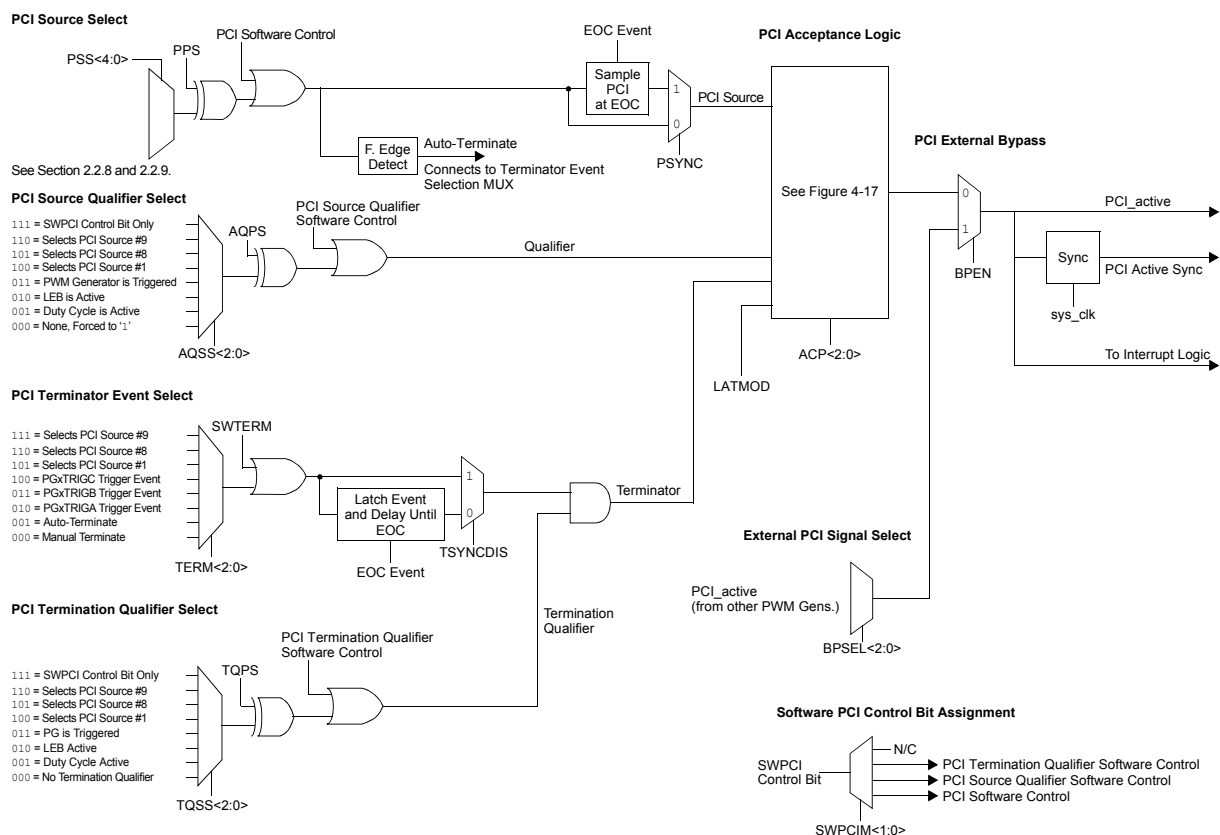
Available PCI source signals and mappings are device-dependent; refer to the specific device data sheet for availability. Typical signals may include:

- Outputs to other PWM Generators
- Combo triggers (see [4.3.5. Combinatorial Triggers](#))
- Analog-to-Digital Converter (ADC)
- Analog comparator
- Input capture
- Configurable Logic Cell (CLC)
- External input (device pin)

The output of a PCI block (PCI_active signal) is made available to the PWM output logic and other PWM Generators. The status of the output signal of each PCI block is made available in the PGxSTAT register ([2.2.3. PGxSTAT](#)) in both current and latched states. The PCI blocks can also generate

interrupts; see [4.2.11.2. Event Interrupts](#) for more information. The block diagrams of the PCI function are shown in [Figure 4-17](#) and [Figure 4-18](#).

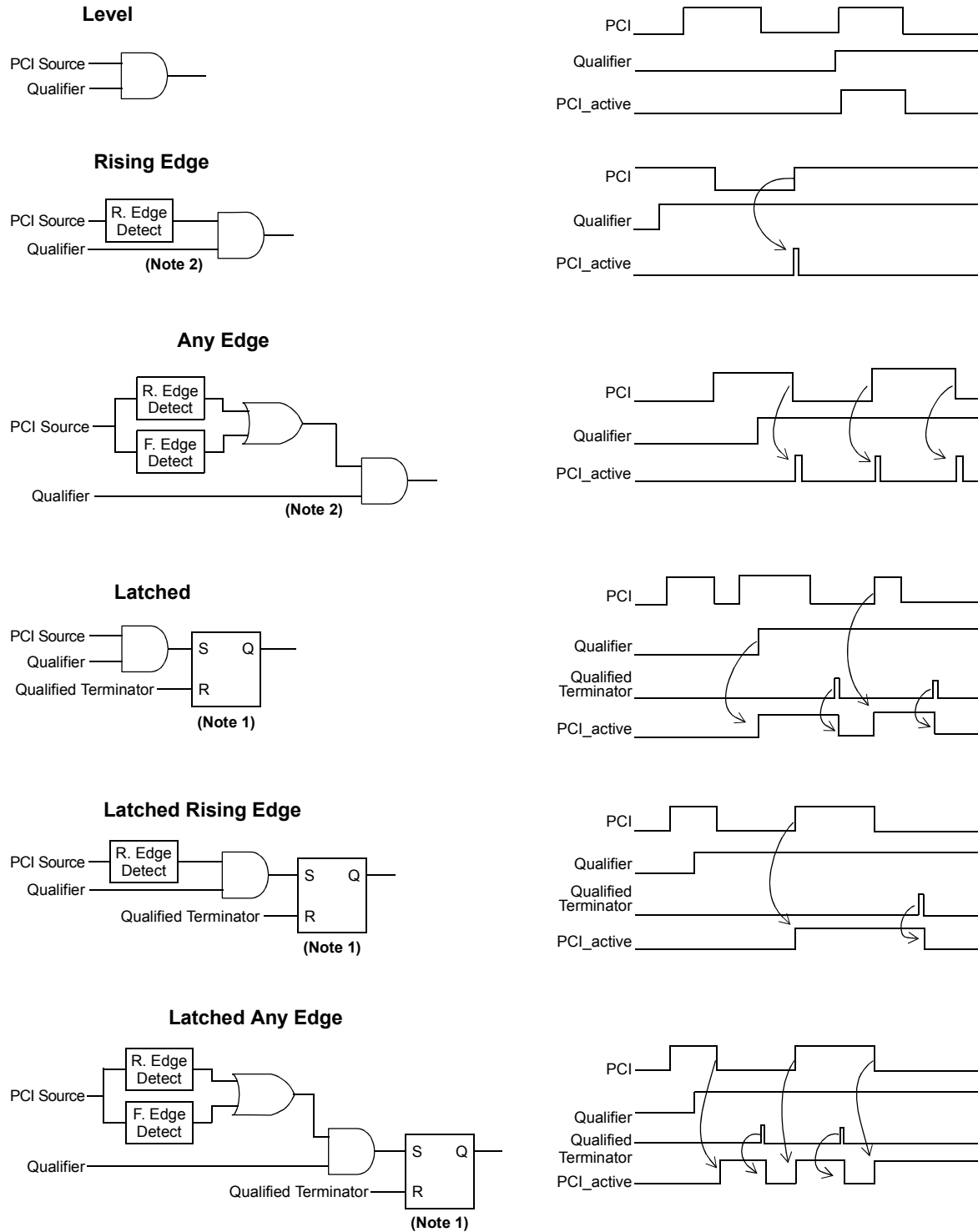
Figure 4-17. PCI Function Block Diagram



Note:

1. See [Section “2.2.8. PGxyPCIL”](#) and [Section “2.2.9. PGxyPCIH”](#).

Figure 4-18. PCI Acceptance Modes



Note 1: SR latch is Set-dominant when LATMOD = 0 and Reset-dominant when LATMOD = 1.

Note 2: Qualifier signal and edge detection is synchronized to PGx_clk.

4.2.5.1 Sync PCI

The main purpose of the Sync PCI block is to trigger and synchronize external events to the PWM clock domain. The synchronization can induce up to a one PWM clock delay. The Sync block is the only PCI block that can initiate a Start-of-Cycle and is available as an input to the SOCS[3:0] (PGxCONH[3:0]) MUX. The Sync and Feed-Forward are the only PCI blocks that can trigger alternate dead time (duty cycle adjustment); see [4.2.6.1. Dead-Time Compensation](#) for additional details.

4.2.5.2 Output Control PCI Blocks

The three output control type PCI blocks are provided to place the PWM outputs in a predetermined state. The blocks are prioritized in the following order, along with further details shown in [Table 4-3](#) and [Table 4-4](#):

1. Fault
2. Current Limit
3. Feed-Forward

The Fault PCI block has the highest priority of the PCI blocks and will dictate the state of the outputs when the block is used. A Fault condition is generally considered catastrophic, and is typically cleared with software.

The Current Limit PCI block was intended for use with current limit sensing circuitry, for either protection or use as a control loop. Leading-Edge Blanking (LEB) is typically used to ignore switching transients in current sensing applications. See [4.2.7. Leading-Edge Blanking](#) for details on LEB.

The Feed-Forward PCI block is intended for use as a control loop for power supply applications. If the sensing circuitry detects a rapid change in load conditions, the system can be configured to take immediate action without having to wait until the next PWM cycle to react.

Once a 'PCI active' signal is asserted, the value stored in the xDAT[1:0] bits (x = FLT, CL or FF) will be applied immediately to the pins. xDAT bits are located in the PGxIOCON register.

4.2.5.3 PCI Logic Description

The PCI block contains three major blocks to support a wide range of applications. First are the inputs, with logic for selecting and conditioning the input signals. Second, the PCI acceptance logic, which is the selectable logic functions applied to the inputs and finally, output logic including bypass.

PCI SOURCE

The PCI source input is the main input into the PCI block and has the following features:

- Input selection multiplexer
- Polarity control
- Software control (SW override)
- Edge detect circuit for auto-terminate
- End-of-Cycle (EOC) synchronization

The PCI source is selected using the PSS[4:0] control bits (PGxyPCIL[4:0]). The polarity of the PCI input source can be selected using the PPS control bit. The chosen PCI input source may be optionally synchronized to the end of a PWM cycle using the PSYNC control bit. This synchronization is useful when a PCI signal is used to gate PWM pulses, as the PCI signal can be delayed to the next PWM boundary, ensuring that a partial pulse is not produced at the output. A falling edge detect circuit is present and can be used to automatically terminate the PCI active signal when selected by the terminator event selection multiplexer.

PCI SOURCE QUALIFIER

The PCI source qualifier is a 2nd input signal used to 'qualify' the PCI source. The PCI source qualifier is ANDed with the PCI source within the PCI acceptance logic. Inputs into the PCI source qualifier multiplexer include:

- Duty cycle active
- LEB active
- PWM Generator triggered
- PWMx output selected by PWMPCI[2:0] (PGxLEBH[10:8])
- External input (another peripheral or device pin)

Like the PCI source input, the PCI source qualifier input has polarity and software control. The PCI source qualifier is used in all PCU acceptance logic types, however, if it is unneeded, AQSS[2:0] (PGxyPCIL[10:8]) can be set to '000' to effectively disable the qualifier.

PCI TERMINATOR EVENT AND QUALIFIER

The PCI termination event sources are used only in the Latched modes of the PCI acceptance logic functions and are used to reset the latch. Inputs to the terminator event inputs include:

- SWTERM bit
- Trigger events (Trigger A, B, C)
- Auto-termination (falling edge detect on PCI source)
- PWMx output selected by PWMPCI[2:0] (PGxLEBH[10:8])
- External input (another peripheral or device pin)

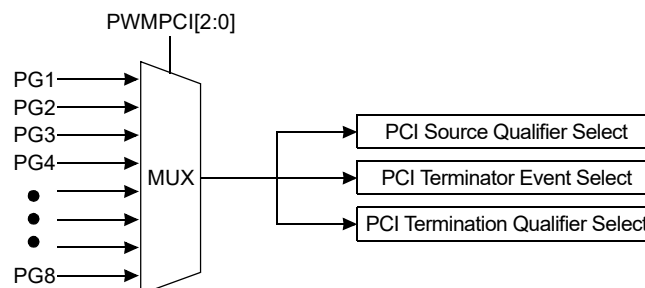
The default option for the PCI terminator is SWTERM. The SWTERM bit must be written at least two PGx_clk cycles prior to the EOC. Otherwise, the PCI termination event will be delayed until the following EOC. The PCI trigger option (Trigger A, B or C) allows the PCI logic to be reset at a particular time in the PWM cycle. User software must select the appropriate PGxTRIG to be used as the PCI trigger source. When using Automatic Termination mode, it is recommended to select 'none' as the PCI termination qualifier. An EOC synchronization is provided by default and can be disabled by setting the TSYNCDIS bit (PGxyPCIL[15]).

The inputs and features of the termination qualifier are similar to the PCI source qualifier. The termination qualifier is used to create more advanced termination events.

USING A PWMx OUTPUT FOR PCI FUNCTION INPUT

The PWMPCI[2:0] control bits (PGxLEBH[10:8]) are used to select which one of the eight PWM outputs can be used by the PCI block. In some control loops, it is desirable to use the output of one PWM Generator to control another generator. The selected PWMx output is made available as a selection on the PCI source qualifier select, PCI terminator event select and PCI termination qualifier select MUXes, as shown in [Figure 4-19](#).

Figure 4-19. PWM Source Selection for PCI



4.2.5.4 PCI Acceptance Logic

PCI acceptance logic is the selectable logic function that is applied to the PCI inputs. The six types of available logic functions shown in [Figure 4-18](#) are:

- Level mode: The PCI signal is passed directly through for use by the PWM Generator. The PCI signal may be optionally qualified (ANDed) with an acceptance qualifier signal.
- Rising Edge modes: The PCI signal is passed through a rising edge detection circuit that generates a pulse event. The PCI signal may be optionally qualified (ANDed) with an acceptance qualifier signal.
- Any Edge mode: The PCI signal is passed through both rising and falling edge detection circuits that generate a pulse event on either edge transition. The PCI signal may be optionally qualified (ANDed) with an acceptance qualifier signal.
- Latched mode: The PCI signal is used to set a SR latch. In this mode, a terminator signal and optional terminator qualifier are used to reset the latch. The entry into the PCI active state is asynchronously latched and possibly gated by a qualifier signal. The exit from the PCI active state is determined by a terminator signal and possibly a terminator qualifier signal. The exit from the PCI active state can also be qualified by the absence of the PCI signal itself. (This is particularly important when the Latched mode is used for Fault control applications.)
- Latched Rising Edge mode: The PCI signal is passed through a rising edge detect circuit and optionally qualified to create a pulse event. This pulse event is used to set a SR latch. The SR latch can be reset in a similar fashion to the Latched mode. The Latched Edge Detect mode allows the PCI to become active on a PCI edge event after a qualifier signal is present.
- Latched Any Edge mode: This mode is similar to Latched Rising Edge mode except that either a rising or falling edge is used to create the pulse event to set the latch.

Each mode of the PCI logic is intended to target a particular kind of power control function, although the functions can be applied to a wide variety of applications. The Level mode is useful when the PCI signals are used to affect the state of the PWM outputs asynchronously. For example, the Level mode could be used to allow an external blanking signal to force the PWM output pins to a specific state for a period of time.

The Edge Event mode is useful when a PCI signal is used to synchronize a PWM Generator time base to an external source. When the PCI logic is used as a synchronization function, the rising edge event of the PCI signal is of primary interest. The edge event causes the PCI logic to generate an internal pulse which triggers a PWM Generator.

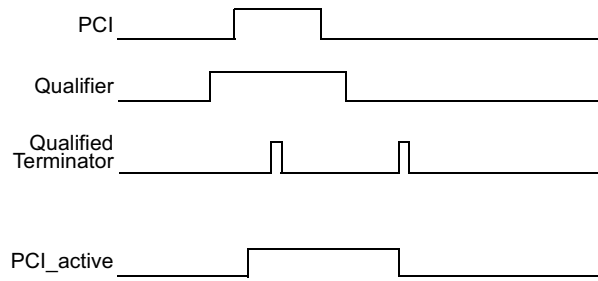
The Latched mode is useful for Fault and current-limiting applications. In these applications, it is important for the PCI logic to enter the active state asynchronously when qualified. The PCI logic will remain active until a selected terminating event occurs. Usually, the terminating event is a software action (manual) or the end of a PWM cycle (automatic). The Latched Edge Detect mode is useful for some types of current control applications. The PCI output cannot become active until a transition of the PCI input occurs after a qualifying condition.

LATCHING MODE CONTROL

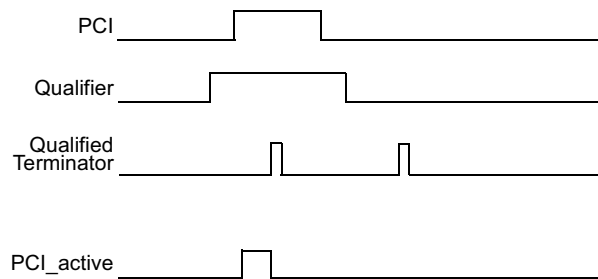
By default, the SR latch used in Latched Acceptance modes is Set-dominant. This prevents a reset of the SR latch if the PCI signal is active when the termination event signal is asserted. The LATMOD control bit (PGxyPCIH[4]) can be used to create a Reset-dominant SR latch for certain PWM control functions. It is not recommended to use a Reset-dominant SR latch when the PCI logic is used to handle Fault conditions as this could allow the active state of the PCI logic to be reset while the PCI input signal is still active. Examples of Latched modes are shown in [Figure 4-20](#).

Figure 4-20. Latch Mode Control

Set-Dominant



Reset-Dominant



4.2.5.5 PCI External Bypass

An option to use the PCI output of another PWM Generator is possible using the PCI external bypass feature. The PCI bypass function is useful when auxiliary, client or combinatorial PWM Generators require PCI functions based on the host PWM Generator's timing. The local PCI logic can be bypassed, using instead the output of the PCI block from another PWM Generator. Only the same type of PCI (Fault, Overcurrent, Sync or Feed-Forward) block can be utilized from another PWM Generator. When $BPEN = 1$, the PWM Generator specified by the $BPSEL[2:0]$ bits ($PGxyPCIH[14:12]$) provides the PCI control. The override states of the $FLTDAT[1:0]$, $CLDAT[1:0]$ and $FFDAT[1:0]$ control bits are not affected when $BPEN = 1$. PWM pin override states are always determined by local control bits.

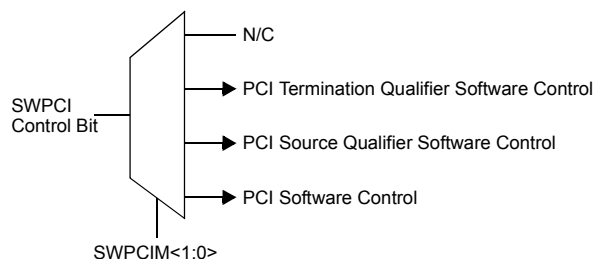
4.2.5.6 Software PCI Control

All PCI blocks have provisions for software control to force and clear events or for development debugging. There are three controls that can be used to manually control the PCI inputs:

- SWPCI control bit
- SWPCIM demultiplexer (for SWPCI control bit)
- SWTERM to generate a termination event

The SWPCI control bit ($PGxyPCIH[7]$) can have its programmed state of '0' or '1' routed to one of three destinations, specified by the SWPCIM[1:0] bits ($PGxyPCIH[6:5]$), as shown in [Figure 4-21](#).

Figure 4-21. Software PCI Control Bit Assignment

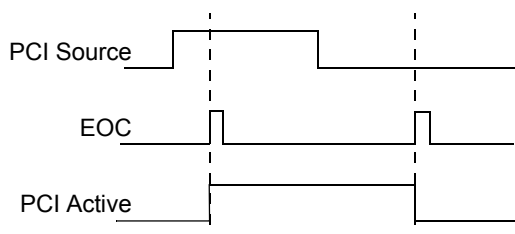


The SWTERM bit is tied directly to the terminator event input logic, and can be used to manually terminate PCI events by writing a '1' to SWTERM and having the TERM[2:0] bits selection set to '000'. Additionally, the acceptance and terminator qualifier input multiplexers have an option to output a fixed state of '1', or when used with their respective polarity control, a fixed state of '0'. These fixed states can be used for debugging or when the acceptance function is not needed.

PCI SOURCE EOC, LEVEL MODE

When the PCI acceptance logic is operated in Level mode and the PCI source is synchronized to the EOC event, there is no logic that retains the state of the prior PCI source signal. Therefore, the resultant PCI output is simply the PCI source signal synchronized to the EOC event. This configuration is useful for PWM chopping applications where the PCI source signal is used as a gating signal. The gating signal is automatically aligned to the PWM cycle boundaries, as shown in [Figure 4-22](#).

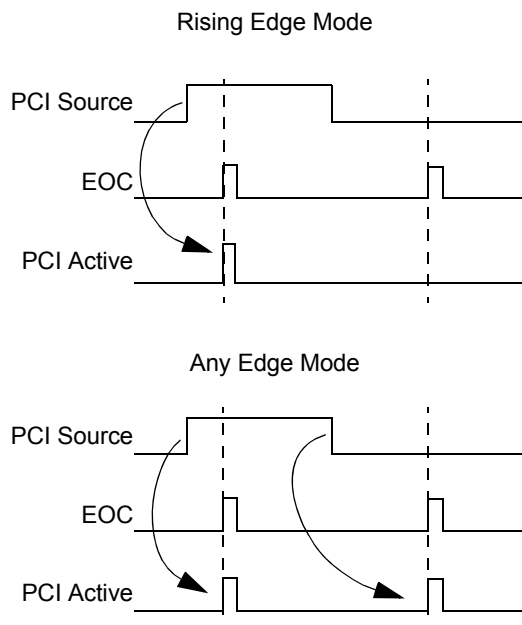
Figure 4-22. PCI Source EOC Sync, Level Acceptance Mode



PCI SOURCE EOC, EDGE MODES

When the PCI acceptance logic is operated in the Rising Edge or Any Edge modes and PSYNC = 1, the PCI source is synchronized to the EOC event, as shown in [Figure 4-23](#). If an edge event is detected, the pulse is delayed until the next EOC event. In the case that a PCI source signal becomes active, and then inactive, within a single PWM cycle, the PCI active signal will not assert.

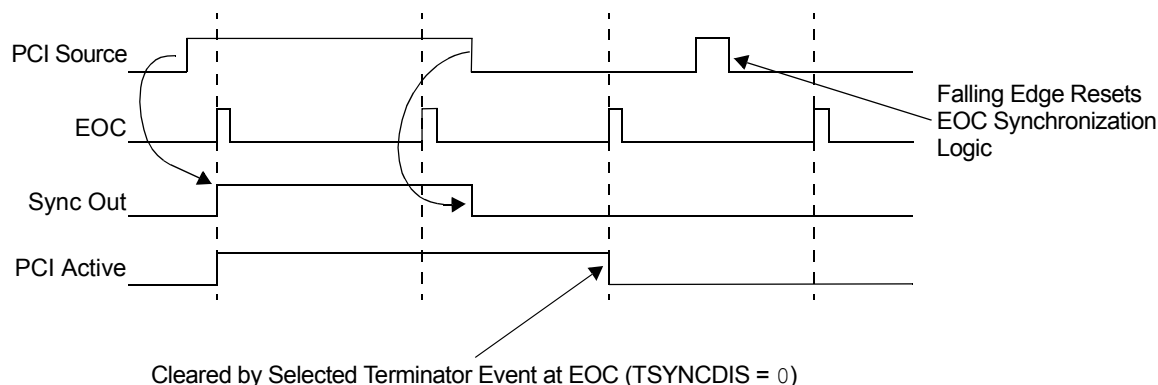
Figure 4-23. PCI Source EOC Sync, Edge Acceptance Modes



PCI SOURCE EOC, LATCHED MODES

When the PCI acceptance logic is operated in the Latched mode and $PSYNC = 1$, the PCI source is synchronized to the EOC event, as shown in Figure 4-24. The synchronization logic delays the rising edge of the PCI source signal until the next occurrence of the EOC signal. The output of the synchronization logic is deasserted on the falling edge of the PCI source signal. The output of the synchronization logic is then used to set the SR latch. A PCI input pulse that operates entirely within one EOC period will not assert the PCI active signal. This is because the falling edge of the PCI input signal resets the EOC synchronization logic before an event can be produced.

Figure 4-24. PCI Source EOC Sync, Latched Acceptance Mode

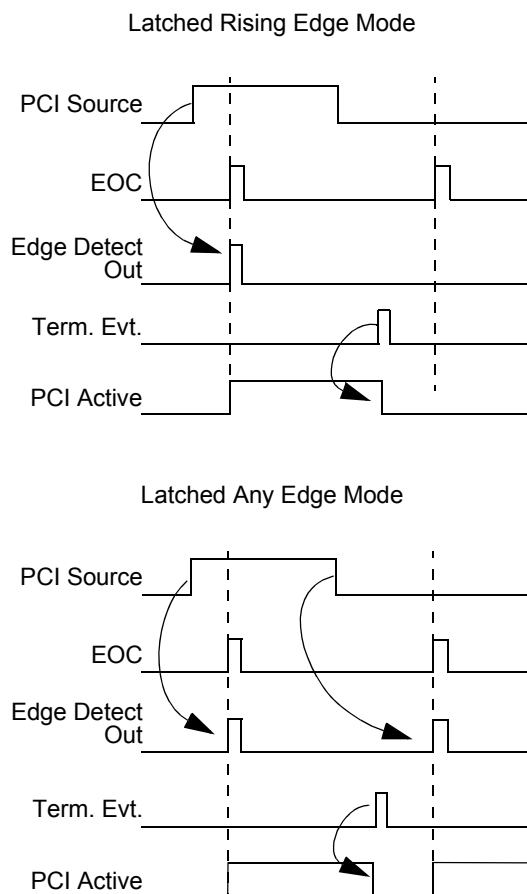


PCI SOURCE EOC, LATCHED EDGE MODES

When the PCI acceptance logic is operated in the Latched Edge modes and $PSYNC = 1$, the PCI source is synchronized to the EOC event, as shown in Figure 4-25. This configuration operates similar to the Rising Edge and Any Edge modes, except that the event output of the synchronization logic is

latched. A PCI source input pulse that operates entirely within a PWM cycle will assert the PCI active signal.

Figure 4-25. PCI Source EOC Sync, Latched Edge Acceptance Modes

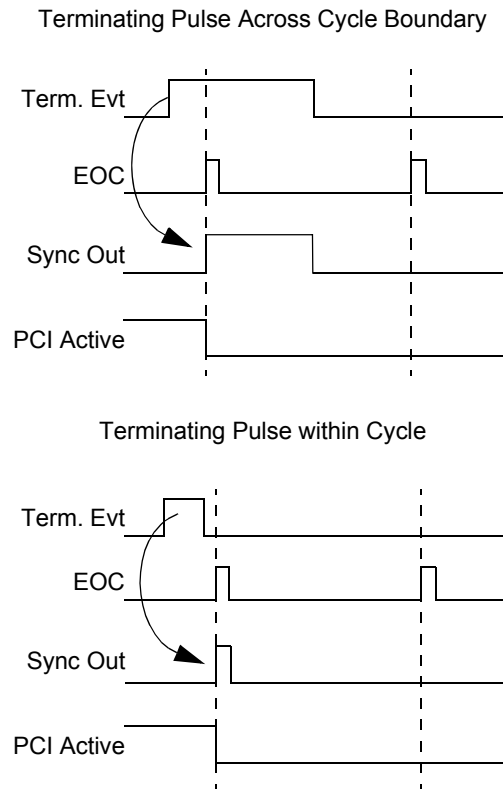


Note: These timing diagrams assume $TSYNCDIS = 1$; therefore, the termination event takes effect immediately.

PCI TERMINATOR EOC

By default, the PCI logic synchronizes a terminator event to the PWM EOC. This allows the PWM to resume cleanly at the start of a new cycle. The rising edge of the terminating signal is held off until an occurrence of the EOC event. The terminator signal is usually a pulse event used to reset the latched state of the PCI logic. If a short pulse is received prior to the occurrence of an EOC event, a Reset pulse is produced at the next EOC event. If the terminator signal is a longer pulse, the synchronized output is held active for as long as the terminator signal is present. This behavior can be used to force the PCI logic to a Reset state, if desired. Terminator event synchronization timing is shown in [Figure 4-26](#).

Figure 4-26. PCI Terminator EOC Synchronization



4.2.6 Dead Time

The dead-time feature is used to provide a time period where neither complementary outputs are active at the same time. Dead time is used to prevent both output driver devices (switches) in a bridge from conducting at the same time, causing excessive current flow. Since output switch turn-on and turn-off times are non-instantaneous, dead time is set to ensure that only one device is active. Dead time is implemented by holding off the assertion of the active state. For the PWMxH and PWMxL outputs, this will delay the rising edge, as shown in [Figure 4-27](#).

Dead-time duration is configured using the PGxDT register. The PGxDT register holds a pair of up to 14-bit dead-time values, DTH and DTL, that are applied independently to the PWMxH and PWMxL outputs, respectively. The effective bit width of the DTH and DTL registers is dependent on High-Resolution mode. When in standard resolution, the upper three bits (13:11) are not used. This yields a maximum value of 0x07FF in standard resolution and 0x3FFF in high resolution. Dead time is typically only used in Complementary Output mode.

Example: Dead-Time Calculation in Standard Resolution

$$F_{PGX_clk} = 500 \text{ MHz}$$

$$\text{Desired Dead Time} = 2 \mu\text{s}$$

$$PGxDTy = F_{PGX_clk} * \text{Dead Time}$$

$$(y = H \text{ or } L)$$

$$PGxDTy = 500 \text{ MHz} * 2 \mu\text{s} = 1000 = 0x03E8$$

Example: Dead-Time Calculation in High Resolution

$$F_{PGx_clk} = 500 \text{ MHz}$$

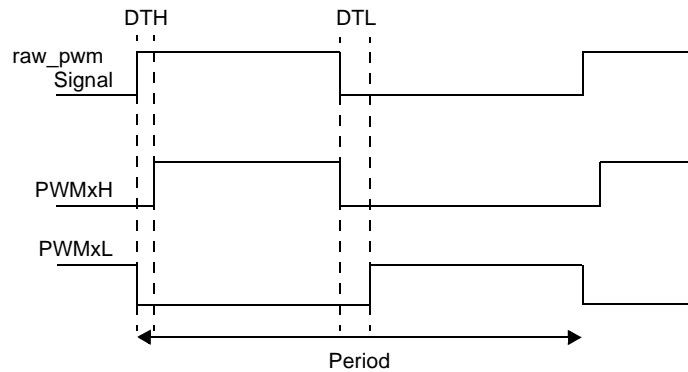
$$\text{Desired Dead Time} = 200 \text{ ns}$$

$$PGxDTy = 8 * F_{PGx_clk} * \text{Dead Time}$$

$$(y = H \text{ or } L)$$

$$PGxDTy = 8 * 500 \text{ MHz} * 200 \text{ ns} = 800 = 0x0320$$

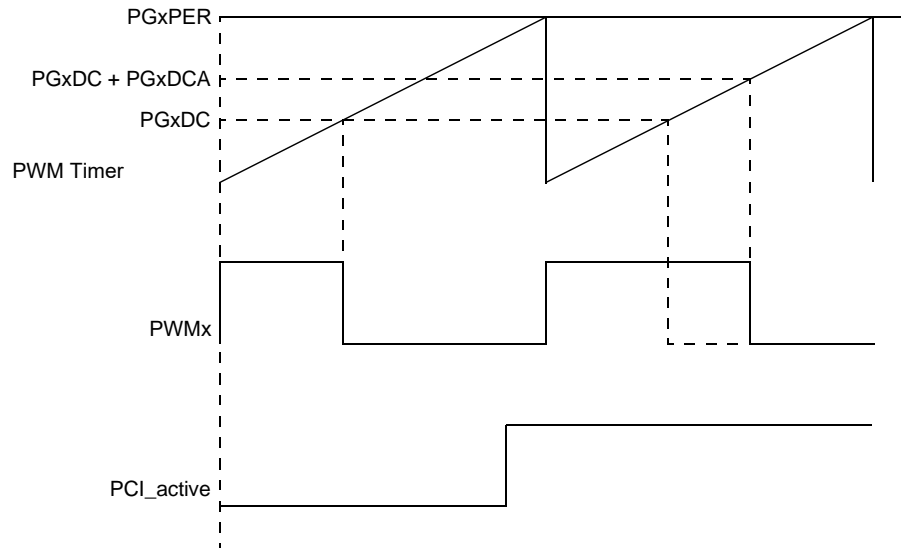
Figure 4-27. PWMxH/PWMxL Rising and Falling Edges Due to Dead Time



4.2.6.1 Dead-Time Compensation

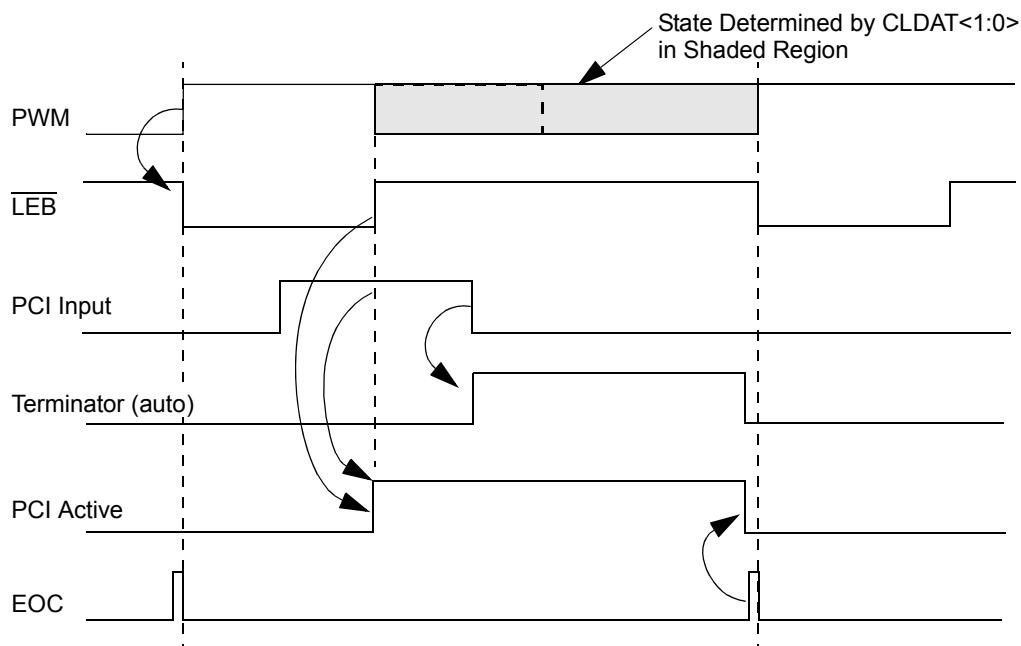
The dead-time compensation feature allows the duty cycle to be selectively controlled by a PCI input. Dead-time compensation is enabled by writing a non-zero value to the PGxDCA (PWM Generator x Duty Cycle Adjustment) register and setting up PCI logic to control the compensation adjustment. When active, the PGxDCA value will be added to the value in the PGxDC register to create the effective duty cycle, as shown in [Figure 4-28](#). In High-Resolution mode, bits[2:0] of PGxDCA are forced to '0'.

The DTCMPSEL control bit (PGxIOCONH[8]) selects the PCI Logic block to be used for dead-time compensation, which can either be the Feed-Forward or Sync PCI blocks. If the PGxDCA register is '0', the dead-time compensation function is disabled regardless of the DTCMPSEL value. The dead-time compensation input signal from the PCI logic is sampled at the end of a PWM cycle for use in the next PWM cycle. The modification of the duty cycle duration via the PGxDCA registers occurs during the end (trailing edge) of the duty cycle.

Figure 4-28. Adding PGxDCA Value to the PGxDC Register Value


4.2.7 Leading-Edge Blanking

The Leading-Edge Blanking (LEB) feature is used to mask transients that could otherwise cause an erroneous Fault condition. Leading-Edge Blanking can be implemented using any of the PCI blocks and basically 'ignores' an input signal for a specified time following a PWM edge event.

Figure 4-29. Leading-Edge Blanking (LEB)


Both the rising and falling edges of both PWMxH and PWMxL signals can be selected to start the LEB timer. The LEB time duration is set by writing a value to the LEB bits (PGxLEBL[15:3]). More than one edge (PHR, PHF, PLR or PLF; refer to the [PGxLEBL](#) register) may be used; however, if timing overlaps,

the counter will be reset on each valid edge. In most applications, only one edge of the PWM signal needs to be selected to trigger the LEB timer.

The LEB counter is commonly used to avoid a false trip when the PCI logic is used for current limiting. In this scenario, the LEB counter can be triggered on both edges of the PWM signal. The PCI logic is operated in Latched Acceptance mode with the LEB active signal used as a disqualifier to the PCI input signal. Figure 4-29 shows a PWM cycle where a PCI input goes active during the LEB timer. There is no PCI active event or output override until the LEB timer has expired.

4.2.7.1 Leading-Edge Blanking Counter Period Calculation

The LEB counter value is stored in the PGxLEBL[15:3] bits and defines the period of the counter (T_{LEB}). The lower three bits are read-only and always read as '0', yielding a minimum LEB resolution of eight PGx_clks. The minimum blanking period is eight PGx_clks, which occurs when LEB[15:0] = 0. Equations for both standard resolution and high resolution are provided in the equations below.

Equation: Leading-Edge Blanking Period

$$T_{LEB} = \frac{8 * (LEB[15:3] + 1)}{F_{PGx_clk}}$$

$$LEB[15:3] = \left(\frac{F_{PGx_clk} * T_{LEB}}{8} \right) - 1$$

Where:

T_{LEB} is specified in time units (ms, μ s or ns)

Equation: Leading-Edge Blanking in Standard and High Resolution

$$F_{PGx_clk} = 500 \text{ MHz}$$

$$LEB \text{ Resolution} = 8/500 \text{ MHz} = 16 \text{ ns Steps}$$

$$\text{Desired LEB Time} = T_{LEB} = 1 \mu\text{s}$$

$$LEB[15:3] = \left(\frac{F_{PGx_clk} * T_{LEB}}{8} \right) - 1$$

$$LEB[15:3] = \left(\frac{500 \text{ MHz} * 1 \mu\text{s}}{8} \right) - 1 = 61 = 0x3D$$

LEB[15:3] is shifted left 3 to yield LEB[15:0] written to PGxLEBL.

LEB[15:0] = 0x01E8.

4.2.7.2 Leading-Edge Blanking PCI Configuration

The LEB counter produces an "LEB active" signal that is supplied to the acceptance qualifier and/or termination qualifier selection multiplexers of the PCI blocks. This allows the LEB timer to be used as a gating signal for the selected PCI or PCI terminator signal. The polarity of the acceptance qualifier and termination qualifier signals can be inverted using the PCI control bits so that the "LEB active" signal is changed to "LEB inactive". It is recommended that the Latched PCI Acceptance mode be used when using LEB, so that the LEB active signal can only affect entry or exit to/from the PCI active state. Auto-termination can also be used to 'reset' the system after the Fault condition has cleared.

Example LEB initialization sequence:

1. Select the type PCI to use for LEB (y = CL, FF, Fault).

2. Select the PCI input using the PSS[4:0] (PGxyPCIL[4:0]) bits. This is typically connected to a comparator to sense overcurrent.
3. Select LEB as the acceptance qualifier using the AQSS[2:0] (PGxyPCIL[10:8]) bits.
4. Invert the acceptance qualifier using the AQPS (PGxyPCIL[11]) bit.
5. Configure logic for Latched mode using the ACP[2:0] (PGxyPCIH[10:8]) bits.
6. Select auto-terminate using the TREM[2:0] (PGxyPCIL[14:12]) bits.

4.2.8 Override

The override feature can be used to take control of the PWM outputs and force certain conditions onto the pins. User software can override the output states of the pins by writing a '1' to the OVRENH and OVRENL control bits located in the PGxIOCONL register. When overridden, the state of the pins will be that of the value written to OVRDAT[1:0], unless it conflicts with restrictions imposed by the given output mode. Most constraints are in Complementary mode and are discussed in [4.2.3.1. Complementary Output Mode](#).

The OVRDAT[1:0] and OVRENH/L control bits are double-buffered for flexibility. The OSYNC[1:0] control bits in the PGxIOCONL register specify when the user override values are applied to the PWM outputs. Manual software overrides can be applied at the following times:

- At the start of a new PWM cycle
- Immediately (or as soon as possible)
- As configured by the UPDMOD[2:0] control bits

Override update process:

1. Configure OSYNC and UPDMOD as needed for application.
2. Configure PWM PG interrupt for SOC (default)
When interrupt event occurs:
3. Read UPDATE bit and verify is '0'.
4. Write new values to OVRENx or OVRDAT.
5. Set UPDREQ bit to a '1' to commit data to PWM buffer.

Details of the UPDMOD[2:0] bits are discussed in [4.2.12. Data Buffering](#).

4.2.9 Synchronizing Overrides on Multiple PWM Generators

Synchronizing overrides on multiple PGs is done by enabling one of the host/client modes as defined by UPDMOD[2:0] bits. Like data updates, overrides are applied by a write of the UPDREQ. When UPDATE = 0, the user software may write new values to OVRENx or OVRDAT to set the UPDREQ bit. Setting the UPDREQ bit 'commits' the new values to the PWM Generator. The UPDATE bit will set and writes are prohibited. When the process is complete, the UPDATE bit is cleared by hardware. SOC update modes are recommended as it allows scheduling of override updates with an interrupt, which provides the maximum time to complete the next write. Immediate updates should poll the UPDATE bit to write at a safe time.

Synchronizing Multiple PWM Generator override updates:

1. Configure host PG for SOC updates by writing OSYNC = '0' and MSTEN = '1'.
2. Configure client(s) PG for 'Client SOC' mode by writing OSYNC = 0b10 and UPDMOD = 0b0101.
3. Configure host PWM interrupt for SOC (default).
When interrupt even occurs:
4. Read host UPDATE bit and verify is '0'.
5. Host write new data values to OVRENx or OVRDAT for all PWM generators (PGx).

6. Set host UPDREQ bit to a '1' to commit data to PWM buffer.

On next SOC, the host and all the client PWM generators will operate with new data values.

4.2.10 ADC Triggers

Each PWM Generator has the capability to trigger multiple ADCs, either internal or external to the device. ADC triggers are based on the TRIGA, TRIGB and TRIGC compare events for timing within the PWM cycle. The ADC triggers are also used as triggers for other peripherals and functions, such as the PTG, DAC and interrupts. The ADC triggers are also made available externally through the Event Output block (see 4.3.6. [PWM Event Outputs](#)) or internally in conjunction with the CPU interrupt controller.

Multiple TRIGx sources may be enabled to create the ADC trigger output, and when enabled, are logically OR'd together. If the multiple TRIGx registers are enabled to produce ADC trigger events, they must be configured to allow unique trigger events to the ADC.

Each PWM Generator can generate two ADC triggers: ADC Trigger 1 and ADC Trigger 2. The two trigger outputs are useful for SMPS applications, where it is often desirable to measure two quantities in a single cycle. Each trigger is connected to a separate ADC, or possibly, a separate ADC trigger input. The ADC Trigger 1 output has an additional offset and postscaler function to allow these functions:

- Postscaler, to reduce the frequency of ADC trigger events.
- Offset, a one-time offset may be applied to ADC trigger events. This allows postscaled trigger events to be interleaved with trigger events from other PWM Generators.

Trigger events from ADC Trigger 2 will be produced every PWM cycle. ADC Trigger 1 output may be postscaled using the ADTR1PS[4:0] control bits (PGxEVTL[15:11]) to reduce the frequency of ADC conversions. In addition, the ADC Trigger 1 output can be offset by a certain number of trigger events using the ADTR1OFS[4:0] control bits (PGxEVTH[4:0]). Together, these two sets of control bits allow the user to establish an interleaved set of ADC triggers from multiple PWM Generators. In addition, ADC trigger events may be simply postscaled to reduce the frequency of ADC measurements. If the ADTR1PS[4:0] control bits are set to '00000', an ADC trigger event will be produced on every PWM cycle. When these control bits are set to a non-zero value, an ADC trigger will be produced during the PWM cycle after the ON bit is set and every N cycle thereafter. The ADTR1OFS[4:0] bits value establishes a one-time offset of 0 to 15 trigger events after the ON bit has been set. After this offset has been established, the trigger postscaler will begin to count the number of trigger events determined by the ADTR1PS[4:0] bits value. When interleaving ADC triggers from multiple PWM Generators, all PWM Generators should be programmed to have the same period to ensure consistent spacing between the trigger events.

4.2.11 Event Selection Block

Each PWM Generator has a logic block for events, triggers and interrupts. These signals are then used by the Event Output block (see 4.3.6. [PWM Event Outputs](#)) or as the trigger source to start a PWM cycle (SOCS[3:0]). The Event Selection block has three main functions:

- ADC trigger configuration
- PWM Generator trigger
- Interrupts

4.2.11.1 PWM Generator Trigger Output

One of the PWM Generator internal events may be selected to drive the PWM Generator trigger output. The PWM Generator trigger output signal is selected using the PGTRGSEL[2:0] bits (PGxEVTL[2:0]) with the selection being either EOC or one of the three TRIGx compare events. Using one of the TRIGx compare events as a SOC trigger for another PWM Generator is useful for implementing a variable phase PWM. The phase relationship between two different PWM Generators can be controlled by the value written to the TRIGx registers.

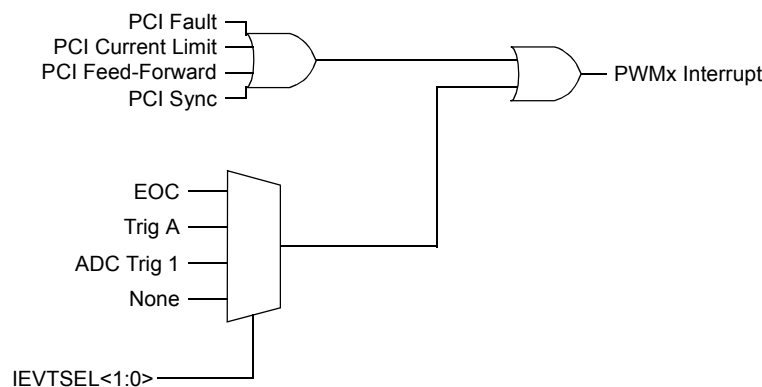
4.2.11.2 Event Interrupts

The PWM event that causes a CPU interrupt is programmable for flexibility. The IEVTSEL[1:0] control bits (PGxEVTH[9:8]) allow the user to select one of the following:

- EOC (default)
- TRIGA compare event
- ADC Trigger 1 event
- None (disabled)

The Event Selection block also contains interrupt enables for each of the four PCI blocks. The SIEN, FFIEN, CLIEN and FLTIEN bits in the PGxEVTH register are used to independently enable interrupts for their respective PCI block. When IEVTSEL[1:0] are set to disabled, the PCI interrupts can still be used independently.

Figure 4-30. Event Selection Block



4.2.12 Data Buffering

The PWM module allows for certain SFR data values to be buffered and applied to the PWM output at later events. The following user registers and/or bits are buffered, allowing the user to modify data while the PWM Generator operates on the previous set of data values:

- PGxPER
- PGxPHASE
- PGxDC
- PGxTRIGA
- PGxTRIGB
- PGxTRIGC
- PGxDT
- SWAP
- OVRDAT[1:0] (software output override values)
- OVRENL/H (software output override enables)

Data are transferred from the SFR registers to the internal PWM registers at the start of a PWM cycle. This can be every one, two or four timer cycles, depending on the PWM Generator mode and the Output mode. It may be required that a register be updated immediately to produce an immediate change in a power converter operation. In other cases, it may be desirable to hold off the buffer update until some external event occurs where data coherency between multiple PWM Generators

is of concern. The module supports the user to specify when the contents of the SFRs associated with a PWM Generator are transferred into the “active” internal registers. Available options are:

- Immediately
- At the beginning of the next PWM cycle
- As part of a larger group

The UPDMOD[2:0] control bits in the PGxCONH register determine the operating mode for register updates. The UPDATE status bit in the PGxSTAT register allows visibility to when register updates are complete and changes may be applied. When UPDATE = 0, the user software may write new values to the PWM Data registers and set the UPDREQ bit when done. Setting the UPDREQ bit ‘commits’ the new values to the PWM Generator and user software can not modify PWM data values until the bit is cleared by hardware.

In order to avoid extra CPU cycles, the data updates can be configured to be automatically performed on a write to one of the PWM Data registers. The register is selected using the UPDTRG[1:0] bits in the PGxEVTL control register. The default selection is that the UPDREQ bit must be manually set in software. A write to the PGxDC register can trigger an update, since many applications frequently change the duty cycle of the PWM. The PGxPHASE and PGxTRIGA registers may also be chosen as update triggers. These registers may be modified on a frequent basis in variable phase applications. The register that is selected as the update trigger must be the last one to be written if several PWM Data registers are to be updated. The PWM Data registers should not be modified once the UPDATE bit becomes set. User software must wait for the PWM hardware to clear the UPDATE bit before the Data registers can be modified again.

In general, SOC update modes are recommend as they allow scheduling of override updates with an interrupt, which provides the maximum time to complete the next write. Otherwise, the UPDATE bit should be polled to write at a safe time.

Data update process:

1. Configure UPDMOD as needed for application.
2. Configure PWM interrupt for SOC (default).
When interrupt even occurs:
3. Read UPDATE bit and verify is '0'.
4. Write new data value(s).
5. Set UPDREQ bit to a '1' to commit data to PWM buffer.

On next SOC, the PWM generator will operate with new data values.

4.2.12.1 Synchronizing Multiple PWM Generator Buffer Updates

The MSTEN control bit (PGxCONH[11]) allows the PWM Generator to control register updates in other PWM Generators. The UPDREQ control and UPDATE status bits can be effectively broadcast to other PWM Generators to allow coherent register updates among a set of PWM Generators that controls a common function. When MSTEN is set and user software (or the PWM Generator hardware) sets the UPDREQ control bit, this event will be broadcast to all other PWM Generators. If UPDMOD[2:0] = 01x in a PWM Generator that receives the request, the receiving module will set its local UPDREQ bit. The local UPDATE status bit will then be cleared when the local registers have been updated. The user software may set a local UPDREQ bit manually.

Table 4-5. PWM Data Register Update Modes

| UPDMOD[2:0] | Mode | Description |
|-------------|-----------|---|
| 000 | SOC | Update Data registers at start of next PWM cycle if UPDREQ = 1. The UPDATE status bit will be cleared automatically after the update occurs. ⁽¹⁾ |
| 001 | Immediate | Update Data registers immediately, or as soon as possible, if UPDREQ = 1. The UPDATE status bit will be cleared automatically after the update occurs. |

.....continued

| UPDMOD[2:0] | Mode | Description |
|-------------|------------------|--|
| 010 | Client SOC | Update Data registers at start of next cycle if a host update request is received. A host update request will be transmitted if MSTEN = 1 and UPDREQ = 1 for the requesting PWM Generator. |
| 011 | Client Immediate | Update Data registers immediately, or as soon as possible, when a host update request is received. A host update request will be transmitted if MSTEN = 1 and UPDREQ = 1 for the requesting PWM Generator. |

Note:

1. The UPDREQ bit must be set at least three sys_clk cycles, followed by three PGx_clk cycles, followed by another three sys_clk cycles, before the next PWM cycle boundary in order to take effect. Otherwise, the data update will be delayed until the following PWM cycle.

Synchronizing Multiple PWM Generator data updates:

1. Configure host PG for SOC updates and write MSTEN = '1'.
2. Configure client(s) PG for 'Client SOC' mode by writing UPDMOD = 0b0101.
3. Configure host PWM interrupt for SOC (default).
When interrupt even occurs:
4. Read host UPDATE bit and verify is '0'.
5. Write new data values to for all PWM generators (PGx).
6. Set host UPDREQ bit to a '1' to commit data to PWM buffer.

On next SOC, the host all PWM generator configured as a clients will operate with new data values.

For the purpose of Data register updates, a PWM cycle length is variable. A PWM cycle may comprise one, two or four timer cycles, depending on the PWM operating mode and the Output mode that is selected. The PWM Data registers may be updated on the next, second or fourth timer cycle when a SOC update has been requested. [Table 4-6](#) summarizes the number of timer cycles between each SOC update vs. the PWM Generator operating mode and the Output mode. For additional information on the timing of update events, refer to the chapter pertaining to the selected PWM mode.

Table 4-6. Timer Cycles per Data Register Update

| PWM Mode | Output Mode | Timer Cycles per PWM Cycle | Timer Cycles per Interrupt and Data Register Update |
|--|-----------------------------------|----------------------------|---|
| Independent Edge, Dual PWM or Variable Phase | Independent Output, Complementary | 1 | 1 |
| Independent Edge, Dual PWM or Variable Phase | Push-Pull | 2 | 2 |
| Center-Aligned | Independent Output, Complementary | 2 | 2 |
| Center-Aligned | Push-Pull | 4 | 4 |
| Double Update Center-Aligned or Dual Edge Center-Aligned | Independent Output, Complementary | 2 | 1 |
| Double Update Center-Aligned or Dual Edge Center-Aligned | Push-Pull | 4 | 1 |

4.2.12.2 Immediate Updates

When using Immediate Update mode, there may be latency from the time of commanding a change to it getting applied. This mode applies changes as soon as possible to prevent unexpected results.

Immediate update of period value updates to the PGxPER value become effective instantaneously. Care should be taken when the PWM period is shortened. If the PWM time base has already counted beyond the new (shorter) PWM period value, a long period will result as the counter must now count to 0xFFFF and then roll over.

Writing a smaller period value after both rising and falling edges have occurred will result in a counter roll over and no EOC event. The subsequent cycle will have no rising or falling edges. Changes made to the period value within 16 clock periods of an existing period match value will not take place until the next PWM cycle due to the pipelined architecture of the PWM timebase.

If immediate updates are required, the best practice is to capture the time base value prior to the period update so a safe minimum period value may be calculated and written.

IMMEDIATE UPDATES TO DUTY CYCLE, PHASE OFFSET

Immediate updates to the duty cycle will be delayed until the next cycle if the PWM pulse is already complete. A second rising edge cannot be created in the same cycle as a result of an immediate update. If the new duty cycle value is greater than the existing phase offset value, and the duty cycle pulse is still in progress, the new value becomes active immediately.

Care should be taken when the phase offset of the PWM pulse is reduced or the length of the PWM pulse is extended. Under these circumstances, a hardware correction may take place depending on the PWM timer and new data value. The best practice is to capture the time base value prior to the duty cycle or phase update so a safe value can be calculated and written.

Hardware correction is implemented using a comparison of PWM time base and new data value. The rising edge and falling edge circuits of each PWM generator perform a greater-than or equal comparison on the 12 MSBs of the time base value (11 MSBs in center-aligned modes).

Figure 4-31 shows two examples of a correction during immediate updates. The PWM period is relatively short (0x1FF) in these examples and a large duty cycle adjustment is made to emphasize how the correction works. In both examples, the duty cycle is decreased from 75% (0xBF) to 25% (0x7F) at approximately the mid-point of the PWM cycle. In both cases, the time base has already elapsed beyond the compare time for 25% duty.

In the first case, the immediate update write occurs at approximately 55% duty cycle (0x011x). The PWM pulse is truncated immediately because the PWM time base is already greater than 0x008x. The programmed duty cycle is 0x007x, so the value of 0x011x provides a true 'greater than' comparison.

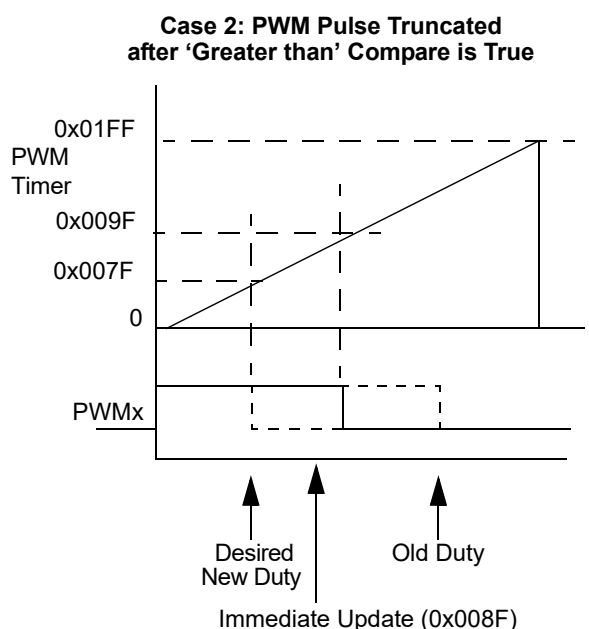
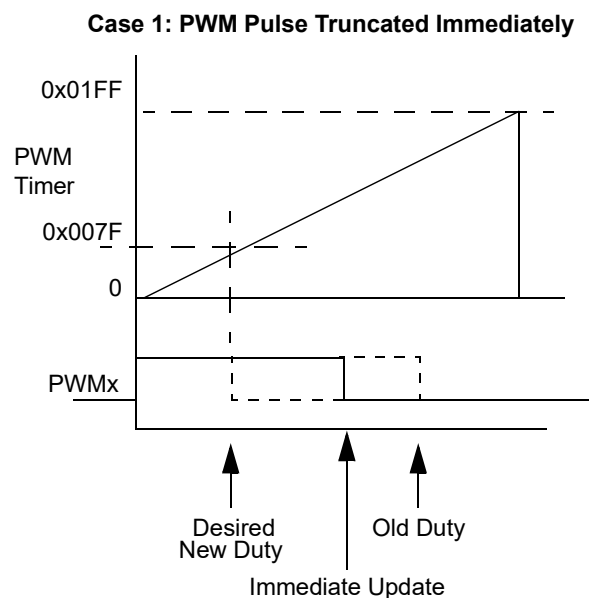
In the second case, the immediate update write occurs just beyond the time of the newly programmed duty cycle, at 0x008F. The PWM pulse is not truncated until the time base reaches a value of 0x009x and the 'greater than' comparison becomes true.

The accuracy of the immediate update correction is a function of the chosen PWM period. The impact of duty cycle corrections could be very noticeable when the programmed period is small. For larger periods, the correction circuit will have much finer granularity, and the effects of the correction circuit will be almost immediate.

An immediate update to the PWM registers takes 8 PWM clock cycles. During that time, no time base matches with the PGxPHASE or PGxDC register may occur. Therefore, a slightly displaced PWM edge may be observed when the PGxPHASE or PGxDC register is updated with a new value that is the same as the old value and the immediate update is coincident with the old programmed edge time.

At EOC/SOC time, the PWM state machine is handling EOC/SOC events. If an update occurs during this time, the edge will be missed and may result in 0% or 100% duty cycle. The best practice is to capture the time base value prior to the duty cycle or phase update so a safe value can be calculated and written.

Figure 4-31. Immediate Update Correction Examples



IMMEDIATE UPDATE TO PHASE

The immediate update correction logic keeps track of whether a PWM pulse has already been created in the current PWM cycle and will not restart a second pulse within the same cycle. For example, if an immediate update changes the PGxPHASE value to a later time in the cycle but a rising edge and falling edge have already been generated, the new value of PGxPHASE will not take effect until the next cycle. An immediate update of PGxPHASE can only be used to delay a PWM pulse which has not occurred yet or cause the pulse to start sooner.

IMMEDIATE UPDATE TO DEAD TIME

If a DT blanking is in progress and an immediate update to the DT occurs, the actual dead time after the update will be extended. This extension is due to the DT counter being reloaded before it has expired. Future dead-time delays after the immediate update will be the new time, as expected.

4.2.13 Time Base Capture

A time base capture feature is provided as the PWM timer itself is not directly readable. When the timer value is needed, it may be captured and read via the PGxCAP register. There are two methods to capture a value: either manually with software or with hardware on a PCI event. The CAPSRC[2:0] control bits (PGxIOCONH[14:12]) are used to select either a manual capture or one of the four PCI blocks as the trigger for a time base capture.

To manually capture the timer value, write a '1' to PGxCAP[0]. The CAP status bit (PGxSTAT[5]) will set to indicate the capture is complete and then the user may read the PGxCAP register to determine the time base value at the time of the hardware event. A read operation of PGxCAP will clear the CAP status bit. No further captures are allowed until user software reads the PGxCAP register. Similarly, when a PCI block is used to capture a time base event, a read operation is needed to reset the logic to allow a subsequent capture event. It is recommended to read the CAP status bit to verify it is set before reading PGxCAP. This is to avoid a read of the PGxCAP register at the same time the PWM hardware is writing it. An alternative method is to schedule reads with an interrupt to avoid concurrent access.

There will be up to four time base clock cycles of latency between the time of the actual event that caused the capture and the actual time base value that is captured. This delay is due to synchronization and sampling delays.

Time base capture example:

1. Read the CAP status bit and verify CAP is '0' (no pending capture).
2. Initiate capture event (SW or PCI).
3. Poll the CAP status bit and wait for it to set to indicate data are ready.

4.2.14 Operation in Debug Mode

When halting program flow using the debugger, the PWMx output pins can be left in a state that may be harmful to the hardware. To avoid this, logic is included to force the pins to a predetermined state, defined by the DBDAT[1:0] bits (PGxIOCONL[1:0]). The pin states are still subject to the priority of overrides, as shown in [Table 4-3](#) and [Table 4-4](#).

4.3 Common Features

4.3.1 Master Data Registers

The PWM module has a set of common Data registers that can be optionally assigned to multiple PWM Generators:

- MDC: Master Duty Cycle register
- MPER: Master Period register
- MPHASE: Master Phase register

These master registers allow user software to affect the operation of multiple PWM Generators by writing one Data register. The MDCSEL, MPERSEL and MPHSEL control bits in each PGxCONH register determine whether the PWM Generator will use the local Data registers or the Master Data registers.

4.3.2 LFSR – Linear Feedback Shift Register

The Linear Feedback Shift register (LFSR) is a pseudorandom number generator that provides 15-bit values. It can be used in applications to modify either the duty cycle and/or period by a small amount to dither the corresponding switching edges of the application circuit's power transistors. This dithering can be useful in reducing peak EMI (Electromagnetic Interference) emissions.

Each read of the LFSR register will result in a new update of the LFSR value. The LFSR initializes at a Power-on Reset to 0x0000, and for successive reads, follows the deterministic sequence shown in Table 4-7. It has the equivalent circuit, as shown in Figure 4-32, which implements a Fibonacci form LFSR based on the primitive polynomial: $x^{15} + x^{14} + 1$ over GF(2). The circuit is modified by a Zero-Detect circuit that causes the 0x0000 value to be followed by 0x0001. Subsequent reads of the LFSR will cycle through all 15-bit values, other than 0x0000, before repeating. The high bit of the LFSR output is always '0'.

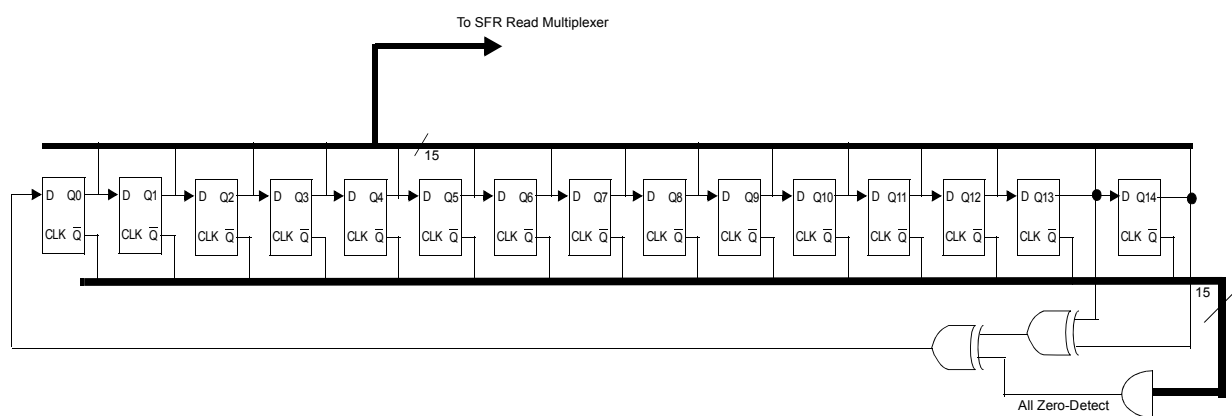
If the same LFSR value is to be used for multiple calculations, then the value read from the LFSR register should be saved in a temporary location for this purpose.

Successive readings of one particular bit of the LFSR forms a Pseudonoise (PN) sequence with an impulse auto-correlation function (shifted versions of the PN sequence are uncorrelated) and may be useful for dithering applications. The entire 15-bit LFSR value has auto-correlation properties that may be undesirable in some applications as a source of pseudorandom noise; its use should be validated in the end application.

Table 4-7. LFSR Successive Read Sequence

| |
|--|
| 0x0000, 0x0001, 0x0002, 0x0004, 0x0008, 0x0010, 0x0020, 0x0040, 0x0080, 0x0100, 0x0200, 0x0400, 0x0800, 0x1000, 0x2000, 0x4001, 0x0003, 0x0006, 0x000c, 0x0018, 0x0030, 0x0060, 0x00c0, 0x0180, 0x0300, 0x0600, 0x0c00, 0x1800, 0x3000, 0x6001, 0x4002, 0x0005, 0x000a, 0x0014, 0x0028, 0x0050, 0x00a0, 0x0140, 0x0280, ..., |
| 0x5557, 0x2AAF, 0x555F, 0x2ABF, 0x557F, 0x2AFF, 0x55FF, 0x2BFF, 0x57FF, 0x2FFF, 0x5FFF, 0x3FFF, 0x7FFF, 0x7FFE, 0x7FFC, 0x7FF8, 0x7FF0, 0x7FE0, 0x7FC0, 0x7F80, 0x7F00, 0x7E00, 0x7C00, 0x7800, 0x7000, 0x6000, 0x4000, 0x0001, 0x0002, 0x0004, 0x0008, |

Figure 4-32. LFSR Block Diagram



4.3.3 Shared Clocking

The PWM clocking system has two additional clock features to support a wide variety of applications. Each PWM generator's clock selection (CLKSEL[1:0]) can independently select the pwm_master_clk, or a divided or scaled version of it.

4.3.3.1 Clock Divider

A common clock divider circuit is available for use by all PWM Generators and allows a PWM Generator to be operated at a low frequency. Four different divider ratios may be selected using the DIVSEL[1:0] control bits (PCLKCON[5:4]). The clock divider circuit remains in a low-power state if none of the PWM Generators have requested it.

4.3.3.2 Frequency Scaling

Frequency scaling provides the ability to drop clocks to effectively stretch the period or duty cycle and is useful for resonant power control applications that require a variable frequency control input. The clock input for the frequency scaling circuit is chosen using the MCLKSEL[1:0] bits (PCLKCON[1:0]). The frequency scaling clock output is available to each PWM Generator and can be selected using the CLKSEL[1:0] control bits (PGxCONL[4:3]).

The FSCL (Frequency Scale) and FSMINPER (Frequency Scaling Minimum Period) registers specify the amount of frequency scaling and are read/writable at all times. The frequency scaling circuit performs modulo arithmetic where the FSCL value is constantly accumulated until the sum is larger than the FSMINPER register value. When the sum becomes larger than the FSMINPER register value, a clock pulse is produced and the accumulated value is reduced by the value in the FSMINPER register, as shown in [Figure 4-33](#).

Note that the frequency scaling signal is applied only to the PWM time base counter, and does not affect the operation of the dead time or the LEB counters. The frequency scaling circuit remains in a low-power state if not selected by any of the PWM Generators. When in High-Resolution mode, frequency scaling cannot be used.

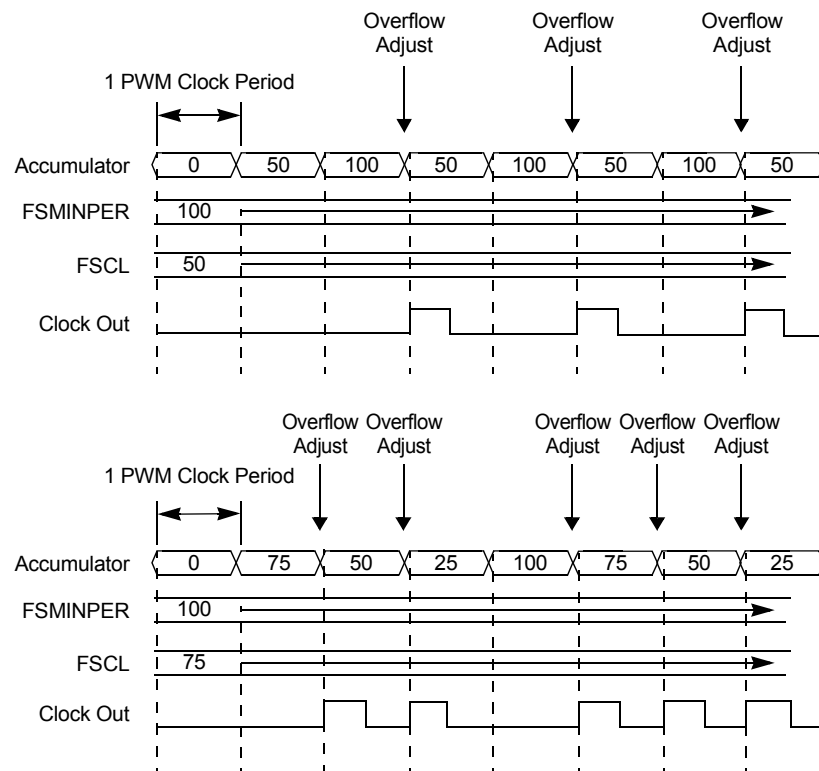
Example: Frequency Scaling Calculation

$$F_{FSCL} = (FSCL / FSMINPER) * F_{PWM}$$

Where:

$$FSCL \leq FSMINPER$$

Figure 4-33. Frequency Scaling Examples



Note: When the frequency scaling circuit is selected as a PWM Generator clock source, the PWM Generator receives two clocks. One clock is the raw clock used to operate the frequency scaling circuit itself. This clock is also used to operate the dead-time counter and LEB counter within the PWM Generator. The second clock is the output of the frequency scaling circuit. This clock is used to operate the PWM time base counter.

4.3.4 Combinatorial Logic Output

The combinatorial logic output feature can be used to generate control signals for synchronous rectification or other applications. One or more PWM Generators can be used to output a logic function, with programmable input selections and logic functions. When assigned to a PWM output, the combinatorial logic function replaces the PWM signal that would normally be connected to that pin. The number of Combinatorial Logic Output blocks is device-dependent. Refer to the device-specific data sheet for availability. The controls include:

- Input sources (PWMSxy)
- Input polarity (SxyPOL)
- Logic AND, OR, XOR function (PWMLFy)
- Output destination (PWMLFyD)

Note: An 'x' in a bit name denotes Input Source '1' or '2'. A 'y' denotes a function instance (A-F).

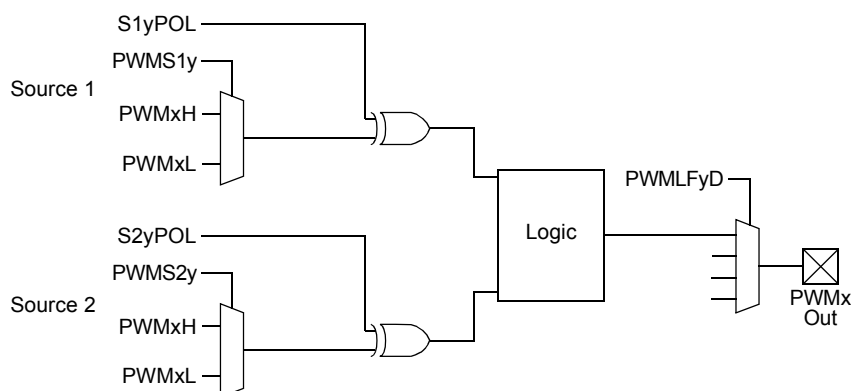
An example of a device with six LOGCONy registers and combinatorial logic output functions, A-F, is shown in Table 4-8.

Table 4-8. Combinatorial Logic Instance Mapping

| Register | Combinatorial Logic Instance | Available Output Pin Selection |
|----------|------------------------------|--------------------------------|
| LOGCONA | A | PWM2H-PWM8H |
| LOGCONB | B | PWM2L-PWM8L |
| LOGCONC | C | PWM2H-PWM8H |
| LOGCOND | D | PWM2L-PWM8L |
| LOGCONE | E | PWM2H-PWM8H |
| LOGCONF | F | PWM2L-PWM8L |

Figure 4-34 shows the combinatorial logic function block diagram.

Figure 4-34. Combinatorial Logic Function Block Diagram



Note: When using combinatorial logic, the two inputs from the PWM Generators must operate from the same clock source; otherwise, the outputs may not be valid. The minimum pulse width of the combinatorial output is device-specific and may be limited by the device pins.

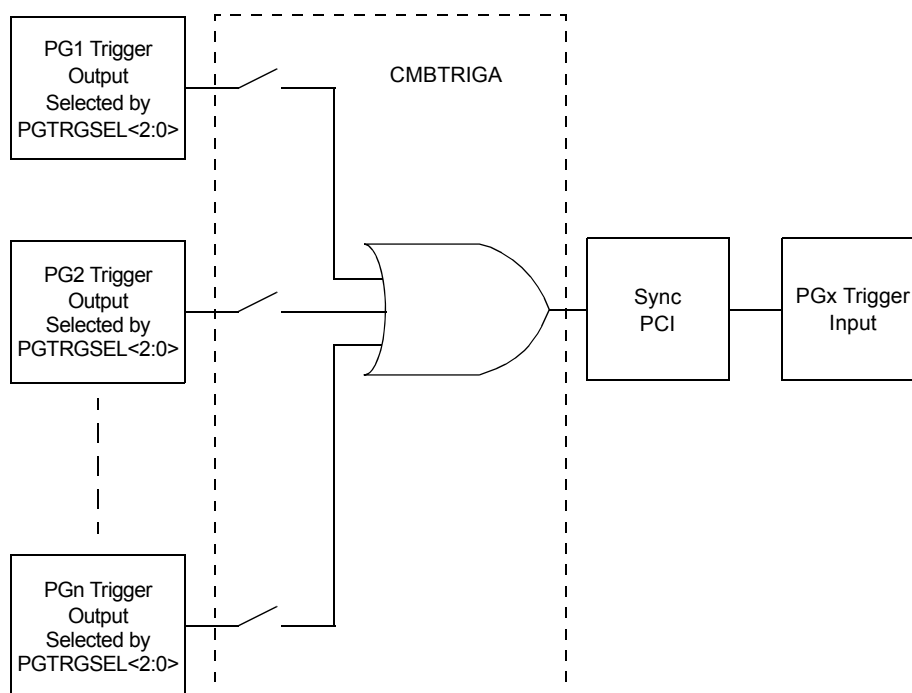
The PWM Generator outputs selected as the source inputs are taken before the PWMxH and PWMxL output polarity control, POLH/POLL (PGxIOCONH[1:0]). If no destination is selected, the combinatorial logic is disabled. The output destination is grouped into pairs, where the odd LOGCONy registers (Instances A, C and E) can be assigned only to the PWMxH output pins, and even LOGCONy registers (Instances B, D and F) assigned only to the PWMxL pins. Only PWM2-PWM8 can use combinatorial logic output; PWM1 is not available. More than one instance (A-F) of a combinatorial logic output can be assigned to a single PWM output if desired. In the case that multiple combinatorial logic functions have been enabled and assigned to the same PWM output, the function with the lowest letter value will take priority.

4.3.5 Combinatorial Triggers

Complex triggering algorithms can be created using the combinatorial trigger feature. There are two independent combinatorial trigger circuits, A and B. This feature allows trigger outputs from multiple PWM Generators to be combined into a single trigger signal, to be used as the trigger source for another PWM Generator.

The input signals used as sources for the combinatorial trigger logic are the trigger outputs selected by the PGTRGSEL[2:0] control bits in each PGxEVTL control register. This trigger output can either be End-of-Cycle (EOC) or one of the three PGxTRIGy (y = A, B or C) compare events. These trigger output signals can be enabled and logically OR'd together by setting the appropriate bits in the CMBTRIGH/CMBTRIGL registers. The Combinatorial Trigger A and Combinatorial Trigger B outputs are then made available on the PWM Control Input (PCI) logic input multiplexers, and routed through the PCI logic for synchronization. Finally, the signal can then be selected as a PWM Generator's input trigger. A block diagram showing an example is shown in Figure 4-35. See 4.2.4. PWM Generator Triggers for details on triggering.

Figure 4-35. Combinatorial Triggers Block Diagram, Example of Instance A



4.3.6 PWM Event Outputs

The PWM event output feature provides a mechanism to interface various PWM signals and events to other peripherals and external devices. The PWM event output logic provides a way to select

and condition an event from any of the PWM Generators. Each PWM Event Output block has the following configuration options:

- PWM Generator instance (PG1...PG8)
- Choice of signal from PWM Generator
- Pulse stretching
- Output signal polarity
- System clock synchronization
- Output enable for the signal

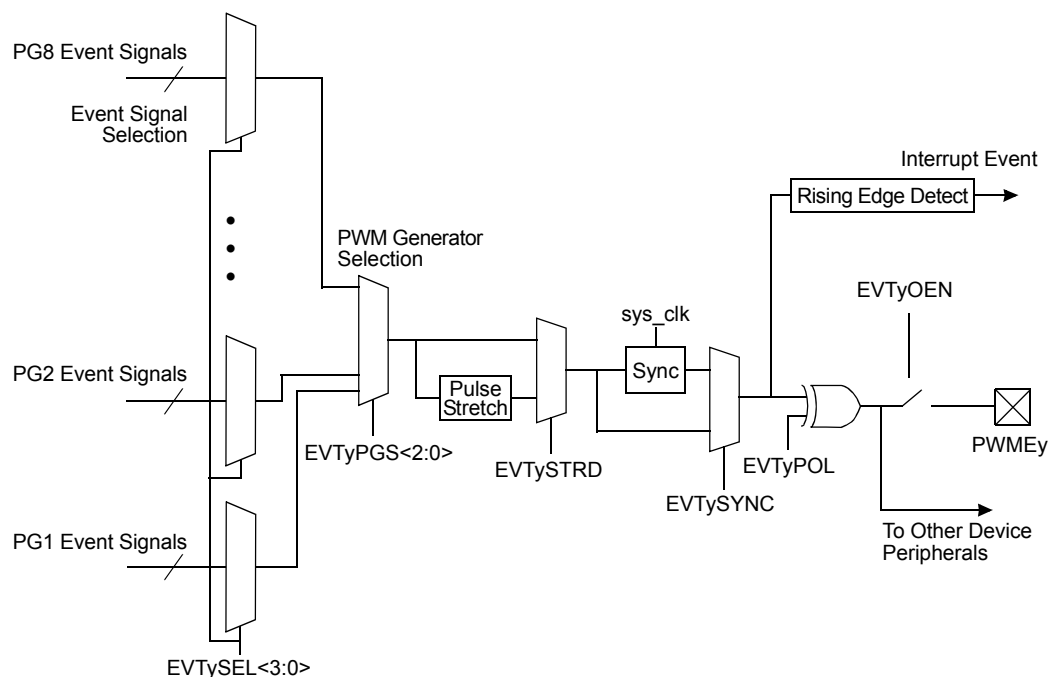
Each **PWMEV_{Ty}** register contains controls for a PWM event output. A device may have multiple instances (A-F) of the PWMEV_{Ty} registers, resulting in four or more total PWM event outputs. Refer to the device-specific data sheet for availability.

The EV_{Ty}SEL[3:0] (PWMEV_{Ty}[7:4]) bits select the signal to be used by the Output block. The default source is the selection determined by PGTRGSEL[2:0] (PGxEVTL[2:0]). For additional information on these signals and configuring the ADC triggers, see [4.2.11. Event Selection Block](#). The EV_{Ty}PGS[2:0] bits (PWMEV_{Ty}[2:0]) are then used to select which of the eight PWM Generator's event signal is to be used.

Some of the event signals running at high speed have short pulses that may not be detected by other circuits and would make it impossible, for example, to connect a PWM event signal to an off-chip destination. A pulse stretching circuit can be used to extend the duration of the pulse by setting the EV_{Ty}STRD bit (PWMEV_{Ty}[13]). If synchronization is desired to the main PWM clock domain, the EV_{Ty}SYNC bit (PWMEV_{Ty}[12]) can be set. The EV_{Ty}POL (PWMEV_{Ty}[14]) control is provided to invert the polarity of the event signal. Finally, an output enable bit, EV_{Ty}OEN (PWMEV_{Ty}[15]), is provided for control over the output pin PWME_{Ey}.

The PWM event output can also generate a system interrupt. An interrupt can be generated from any of the various triggers and events that are input into the Event Output block. A block diagram of the event output function is shown in [Figure 4-36](#).

Figure 4-36. PWM Event Output Function



4.4 Lock and Write Restrictions

The LOCK bit (PCLKCON[8]) may be set in software to block writes to certain registers. A special system-dependent lock/unlock sequence is required to set or clear the LOCK bit. Refer to the device-specific data sheet for the unlock sequence. The following [Table 4-9](#) details write access and register modification restrictions. In general, modifications to configuration controls should not be done while the module is running, as indicated by the ON bit (PGxCONL[15]) being set.

Caution should be used when modifying data registers (Period, Duty Cycle and Phase) as behavior may be dependent on the specific mode of operation used. Refer to the applicable PWM mode within [4.2.2. PWM Modes](#). Also see [4.2.12. Data Buffering](#) regarding data buffering for further details, including the UPDATE bit (PGxSTAT[4]).

Table 4-9. PWM Register Map (Lock and Write Restrictions)

| Name | Bit Range | Bit Names | | | | | | | |
|-------------|-----------|------------------|------------|----------------|----------------|--------------|----------------|-----------------|-----------|
| PCLKCON | 15:8 | HRRDY | HRERR | — | — | — | — | — | LOCK |
| | 7:0 | — | — | DIVSEL[1:0](2) | | — | — | MCLKSEL[1:0](2) | |
| FSCL(5) | 15:8 | FSCL[15:8] | | | | | | | |
| | 7:0 | FSCL[7:0] | | | | | | | |
| FSMINPER(5) | 15:8 | FSMINPER[15:8] | | | | | | | |
| | 7:0 | FSMINPER[7:0] | | | | | | | |
| MPHASE(5) | 15:8 | MPHASE[15:8] | | | | | | | |
| | 7:0 | MPHASE[7:0] | | | | | | | |
| MDC(5) | 15:8 | MDC[15:8] | | | | | | | |
| | 7:0 | MDC[7:0] | | | | | | | |
| MPER(5) | 15:8 | MPER[15:8] | | | | | | | |
| | 7:0 | MPER[7:0] | | | | | | | |
| LFSR(6) | 15:8 | — | LFSR[14:8] | | | | | | |
| | 7:0 | LFSR[7:0] | | | | | | | |
| CMBTRIGL(4) | 15:8 | — | — | — | — | — | — | — | — |
| | 7:0 | CTA8EN(4) | CTA7EN(4) | CTA6EN(4) | CTA5EN(4) | CTA4EN(4) | CTA3EN(4) | CTA2EN(4) | CTA1EN(4) |
| CMBTRIGH(4) | 15:8 | — | — | — | — | — | — | — | — |
| | 7:0 | CTB8EN(4) | CTB7EN(4) | CTB6EN(4) | CTB5EN(4) | CTB4EN(4) | CTB3EN(4) | CTB2EN(4) | CTB1EN(4) |
| LOGCONyL(4) | 15:8 | PWMS1A[3:0] | | | | PWMS2A[3:0] | | | |
| | 7:0 | S1APOL | S2APOL | PWMLFA[1:0] | | — | PWMLFAD[2:0] | | |
| LOGCONyH(4) | 15:8 | PWMS1B[3:0] | | | | PWMS2B[3:0] | | | |
| | 7:0 | S1BPOL | S2BPOL | PWMLFB[1:0] | | — | PWMLFBD[2:0] | | |
| PWMEVtyL | 15:8 | EVT1OEN | EVT1POL | EVT1STRD | EVT1SYNC | — | — | — | — |
| | 7:0 | EVT1SEL[3:0] | | | | — | EVT1PGS[2:0] | | |
| PWMEVtyH | 15:8 | EVT2OEN | EVT2POL | EVT2STRD | EVT2SYNC | — | — | — | — |
| | 7:0 | EVT2SEL[3:0] | | | | — | EVT2PGS[2:0] | | |
| PGxCONL | 15:8 | ON | r | — | — | — | TRGCNT[2:0](1) | | |
| | 7:0 | HREN(2) | — | — | CLKSEL[1:0](4) | | MODSEL[2:0](1) | | |
| PGxCONH | 15:8 | MDCSEL(4) | MPERSEL(4) | MPHSEL(4) | — | MSTEN(4) | UPDMOD[2:0] | | |
| | 7:0 | TRIGMOD[1:0](4)(| | — | — | SOCS[3:0](4) | | | |

.....continued

| Name | Bit Range | Bit Names | | | | | | | |
|------------------------|-----------|-----------------------------|----------------------------|--------------------------|------------------------------|---------------------|------------------------------|-----------------------------|-------------------------|
| PGxSTAT | 15:8 | SEVT ⁽⁵⁾ | FLTEVT ⁽⁵⁾ | CLEVT ⁽⁵⁾ | FFEVT ⁽⁵⁾ | SAVT ⁽⁶⁾ | FLTACT ⁽⁶⁾ | CLACT ⁽⁶⁾ | FFACT ⁽⁶⁾ |
| | 7:0 | TRSET ⁽⁵⁾ | TRCLR ⁽⁵⁾ | CAP ⁽⁵⁾ | UPDATE ⁽⁵⁾ | — | STEER ⁽⁶⁾ | CAHALF ⁽⁶⁾ | TRIG ⁽⁵⁾ |
| PGxIOCONL | 15:8 | CLMOD | SWAP | OVRENH | OVRENL | OVRDAT[1:0] | | OSYNC[1:0] | |
| | 7:0 | FLTDAT[1:0] | | CLDAT[1:0] | | FFDAT[1:0] | | DBDAT[1:0] | |
| PGxIOCONH | 15:8 | — | CAPSRC[2:0] ⁽⁵⁾ | | | — | — | — | DTCMPSEL ⁽⁴⁾ |
| | 7:0 | — | — | PMOD[1:0] ⁽²⁾ | | PENH ⁽²⁾ | PENL ⁽²⁾ | POLH ⁽²⁾ | POLL ⁽²⁾ |
| PGxEVTL | 15:8 | ADTR1PS[4:0] ⁽⁴⁾ | | | | | ADTR1EN3 ⁽⁴⁾ | ADTR1EN2 ⁽⁴⁾ | ADTR1EN1 ⁽⁴⁾ |
| | 7:0 | — | — | — | UPDTRG[1:0] ⁽⁴⁾ | | PGTRGSEL[2:0] ⁽⁴⁾ | | |
| PGxEVTH | 15:8 | FLTIE ⁽⁵⁾ | CLIE ⁽⁵⁾ | FFIE ⁽⁵⁾ | SIEN ⁽⁵⁾ | — | — | IEVTSEL[1:0] ⁽⁴⁾ | |
| | 7:0 | ADTR2EN3 ⁽⁴⁾ | ADTR2EN2 ⁽⁴⁾ | ADTR2EN1 ⁽⁴⁾ | ADTR1OFS[4:0] ⁽⁴⁾ | | | | |
| PGxFPCIL | 15:8 | TSYNCDIS | TERM[2:0] | | | AQPS | AQSS[2:0] | | |
| | 7:0 | SWTERM | PSYNC | PPS | PSS[4:0] | | | | |
| PGxFPCIH | 15:8 | BPEN | BPSEL[2:0] | | | — | ACP[2:0] | | |
| | 7:0 | SWPCI | SWPCIM[1:0] | | LATMOD | TQPS | TQSS[2:0] | | |
| PGxCLPCIL | 15:8 | TSYNCDIS | TERM[2:0] | | | AQPS | AQSS[2:0] | | |
| | 7:0 | SWTERM | PSYNC | PPS | PSS[4:0] | | | | |
| PGxCLPCIH | 15:8 | BPEN | BPSEL[2:0] | | | — | ACP[2:0] | | |
| | 7:0 | SWPCI | SWPCIM[1:0] | | LATMOD | TQPS | TQSS[2:0] | | |
| PGxFFPCIL | 15:8 | TSYNCDIS | TERM[2:0] | | | AQPS | AQSS[2:0] | | |
| | 7:0 | SWTERM | PSYNC | PPS | PSS[4:0] | | | | |
| PGxFFPCIH | 15:8 | BPEN | BPSEL[2:0] | | | — | ACP[2:0] | | |
| | 7:0 | SWPCI | SWPCIM[1:0] | | LATMOD | TQPS | TQSS[2:0] | | |
| PGxSPCIL | 15:8 | TSYNCDIS | TERM[2:0] | | | AQPS | AQSS[2:0] | | |
| | 7:0 | SWTERM | PSYNC | PPS | PSS[4:0] | | | | |
| PGxSPCIH | 15:8 | BPEN | BPSEL[2:0] | | | — | ACP[2:0] | | |
| | 7:0 | SWPCI | SWPCIM[1:0] | | LATMOD | TQPS | TQSS[2:0] | | |
| PGxLEBL ⁽⁴⁾ | 15:8 | LEB[15:8] | | | | | | | |
| | 7:0 | LEB[7:0] | | | | | | | |
| PGxLEBH ⁽⁴⁾ | 15:8 | — | — | — | — | — | PWMPCI[2:0] | | |
| | 7:0 | — | — | — | — | PHR | PHF | PLR | PLF |

|continued | | | | | | | | | |
|-------------------------|-----------|----------------|---|-----------|---|---|---|---|---|
| Name | Bit Range | Bit Names | | | | | | | |
| PGxPHASE ⁽³⁾ | 15:8 | PGxPHASE[15:8] | | | | | | | |
| | 7:0 | PGxPHASE[7:0] | | | | | | | |
| PGxDC ⁽³⁾ | 15:8 | PGxDC[15:8] | | | | | | | |
| | 7:0 | PGxDC[7:0] | | | | | | | |
| PGxDCA ⁽³⁾ | 15:8 | — | — | — | — | — | — | — | — |
| | 7:0 | PGxDCA[7:0] | | | | | | | |
| PGxPER ⁽³⁾ | 15:8 | PGxPER[15:8] | | | | | | | |
| | 7:0 | PGxPER[7:0] | | | | | | | |
| PGxTRIGA ⁽³⁾ | 15:8 | PGxTRIGA[15:8] | | | | | | | |
| | 7:0 | PGxTRIGA[7:0] | | | | | | | |
| PGxTRIGB ⁽³⁾ | 15:8 | PGxTRIGB[15:8] | | | | | | | |
| | 7:0 | PGxTRIGB[7:0] | | | | | | | |
| PGxTRIGC ⁽³⁾ | 15:8 | PGxTRIGC[15:8] | | | | | | | |
| | 7:0 | PGxTRIGC[7:0] | | | | | | | |
| PGxDTL ⁽³⁾ | 15:8 | — | — | DTL[13:8] | | | | | |
| | 7:0 | DTL[7:0] | | | | | | | |
| PGxDTH ⁽³⁾ | 15:8 | — | — | DTH[13:8] | | | | | |
| | 7:0 | DTH[7:0] | | | | | | | |
| PGxCAP ⁽⁶⁾ | 15:8 | PGxCAP[15:8] | | | | | | | |
| | 7:0 | PGxCAP[7:2] | | | | | | 0 | 0 |

Legend: — = unimplemented, read as '0'; r = reserved bit.

Notes:

1. This bit(s) cannot be modified while on ON (PGxCONL[15]) = 1.
2. This bit(s) or register(s) cannot be modified while LOCK (PCLKCON[8]) = 1. Avoid modifying this bit(s) when ON (PGxCONL[15]) = 1.
3. This bit(s) or register(s) cannot be modified while UPDATE = 1.
4. Avoid modifying this bit(s) or register(s) while ON (PGxCONL[15]) = 1.
5. This bit(s) or register(s) can be freely modified while ON (PGxCONL[15]) = 1.
6. Location is read-only. No write restrictions exist.

5. Application Examples

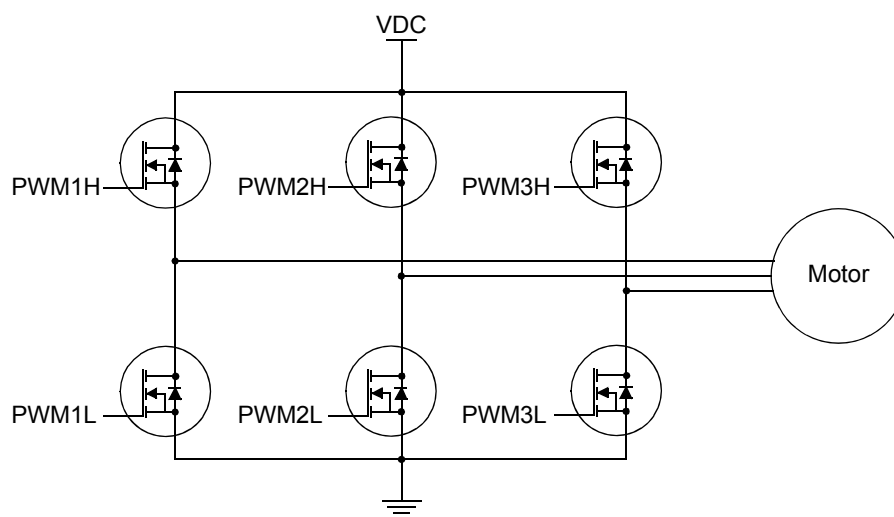
5.1 Six-Step Commutation of Three-Phase BLDC Motor

One method for controlling a three-phase Brushless DC (BLDC) is six-step commutation. Six-step commutation is also called 120° commutation or trapezoidal control and uses six steps or 'sectors' over one electrical cycle to energize a BLDC motor. Each sector is equivalent to 60 electrical degrees, with the six sectors resulting in 360 electrical degrees or one electrical cycle.

Sequencing through these steps moves the motor through one electrical cycle, which in mechanical terms, corresponds to one pair of rotor magnet poles moving past stator windings. A given BLDC motor has a certain number of pole pairs, defined by N_p as a positive integer. If the motor is rotated one mechanical revolution, this corresponds to N_p electrical cycles. This yields one electrical cycle for a 2-pole motor, two electrical cycles for a 4-pole motor, three electrical cycles for a 6-pole motor and so on.

A six-step commutation drive is typically implemented using a three-phase bridge circuit, as shown in [Figure 5-1](#). Each phase of the motor is connected to a half-bridge driver and controlled with a complementary PWM pair output. At any given time in the six-step commutation scheme, only two of the three motor windings are energized. Current in the motor winding flows from one phase to another, in either direction.

Figure 5-1. Three-Phase Bridge and Motor



Various PWM switching techniques can be employed for six-step commutation. The following [Table 5-1](#) summarizes the three schemes presented in this manual.

Table 5-1. PWM Switching Schemes for 6-Step Commutation

| Scheme | Technique Overview | Advantage | Disadvantage |
|---------|--|--|---------------------|
| Scheme1 | High side of one active phase and low side of the other active phase driven at any given time | Simplest scheme; no dead time needed; low switching loss | High-current ripple |
| Scheme2 | One active phase driven complementary and the other active phase's low side is driven to 100% duty cycle | Low switching loss | Requires dead time |

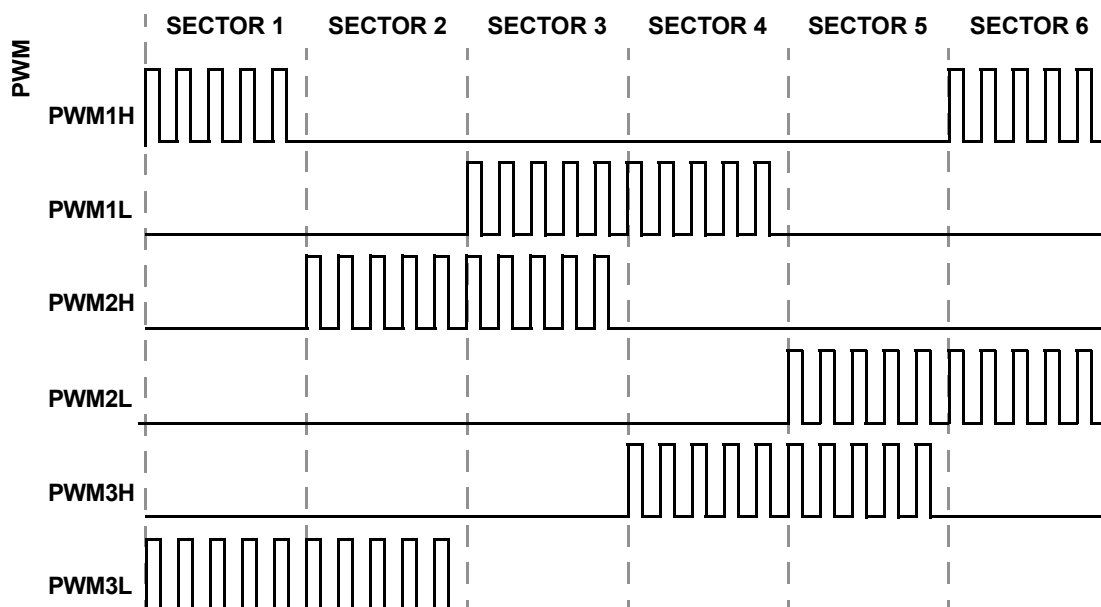
.....continued

| Scheme | Technique Overview | Advantage | Disadvantage |
|----------|---|-----------------------|--|
| Scheme 3 | Both active phases are driven complementary | Lowest current ripple | Higher switching loss and requires dead time |

5.1.1 Six-Step Commutation – PWM Scheme 1

In this PWM scheme, only two switches are active at any given time. Of the two active phases, one high-side and one low-side switch is controlled with its phase's corresponding PWM waveform, as shown in Figure 5-2.

Figure 5-2. Six-Step PWM Scheme 1 Waveform



Since only one switch needs to be driven at a time on a given phase, Independent PWM Output mode is used. The output override feature is then used to suppress the unused output. A three-phase scheme is implemented using PWM Generator 1 (PG1) configured as host and the other two PWM Generators (PG2 and PG3) configured as clients. PG1 is self-triggered, whereas PG2 and PG3 are triggered from PG1's Start-of-Cycle (SOC). Enabling PG1 will start the system in a synchronized fashion.

Configuration Summary:

- Independent Edge PWM mode
- Independent Output mode
- Master Period and Duty Cycle Used
- Override State is drive low

Example 5-1. Six-Step PWM Scheme 1 Code

```
#include<stdint.h>
uint16_t state = 0;
uint16_t PWMState1[6] = {0x1000, 0x1000, 0x3000, 0x2000, 0x2000, 0x3000};
uint16_t PWMState2[6] = {0x2000, 0x3000, 0x1000, 0x1000, 0x3000, 0x2000};
uint16_t PWMState3[6] = {0x3000, 0x2000, 0x2000, 0x3000, 0x1000, 0x1000};

int main()
{
    .....
```

```

.....
.....
PWMInitialization();
while(1)
{

for(state = 0;state < 6;state++)
{
    Delay();
    PG1IOCONL = PWMState1[state];
    PG2IOCONL = PWMState2[state];
    PG3IOCONL = PWMState3[state];

    }
}
}
void    PWMInitialization(void)
{
/* Set PWM MASTER Period */

MPER = 10000;

/* Set Duty Cycle - 25% */

MDC = 2500;

/* Set Phase shift - No phase shift */

MPHASE = 0;

/* Select PWM Generator duty cycle register as MDC */
/* Select PWM Generator period register as MPER */
/* Select PWM Generator phase register as MPHASE */
/* PWM Generator broadcasts software set of UPDREQ */
/* control bit and EOC signal to other PWM Generators. */
/* PWM buffer update mode is at start of next PWM cycle if UPDREQ = 1 */
/* PWM generator operates in single trigger mode */
/* Start of cycle is local EOC */

PG1CONH = 0xE800;

/* PWM Generator is disabled */
/* PWM Generator uses Master Clock selected by
 * the PCLKCONbits.MCLKSEL bits */
/* PWM Generator operates in Independent Edge PWM mode*/

PG1CONL = 0x0008;

/* PWM Generator Output Mode is Independent Mode */
/* PWM Generator controls the PWMxH output pin */
/* PWM Generator controls the PWMxL output pin */

PG1IOCONH = 0x001C;

/* Override is enabled on PWMxH/L with OVRDAT = 0b00,
 * turning OFF PWM outputs */
/* User output overrides are synchronized to next start of cycle */

PG1IOCONL = 0x3000;

/* PGxTRIGA register compare event is enabled as trigger source
 * for ADC Trigger 1 */
/* A write of the PGxDC register automatically sets the UPDREQ bit */
/* PWM generator trigger output is EOC*/

PG1EVTL = 0x0108

/* Select PWM Generator Duty Cycle Register as MDC */
/* Select PWM Generator Period Register as MPER */
/* Select PWM Generator Phase Register as MPHASE */
/* PWM generator does not broadcast UPDATE status bit or
 * EOC signal to other PWM generators */
/* PWM Buffer Update Mode is client immediate*/
/* PWM generator operates in Single Trigger Mode */
/* Start of Cycle is PG1 trigger output selected
 * by PG1EVTbits.PGTRGSEL<2:0> bits */

PG2CONH = 0xE301;

```

```

/* PWM Generator is enabled */
/* PWM Generator uses Master Clock selected by
 * the PCLKCONbits.MCLKSEL bits */
/* PWM Generator operates in Independent Edge PWM mode*/

PG2CONL = 0x8008;

/* PWM Generator Output Mode is Independent Mode */
/* PWM Generator controls the PWMxH output pin */
/* PWM Generator controls the PWMxL output pin */

PG2IOCONH = 0x001C;

/* Override is enabled on PWMxH/L with OVRDAT = 0b00,
 * turning OFF PWM outputs */
/* User output overrides are synchronized to next start of cycle */

PG2IOCONL = 0x3000;

/* Select PWM Generator Duty Cycle Register as MDC */
/* Select PWM Generator Period Register as MPER */
/* Select PWM Generator Phase Register as MPM */
/* PWM generator does not broadcasts UPDATE status bit or EOC signal
 * to other PWM generators */
/* PWM Buffer Update Mode is client immediate*/
/* PWM generator operates in Single Trigger Mode */
/* Start of Cycle is PG1 trigger output selected by
 * PG1EVTbits.PGTRGSEL<2:0> bits */

PG3CONH = 0xE301;

/* PWM Generator is enabled */
/* PWM Generator uses Master Clock selected by the PCLKCONbits.MCLKSEL bits */
/* PWM Generator operates in Independent Edge PWM mode*/

PG3CONL = 0x8008;

/* PWM Generator Output Mode is Independent Mode */
/* PWM Generator controls the PWMxH output pin */
/* PWM Generator controls the PWMxL output pin */

PG3IOCONH = 0x001C;

/* Override is enabled on PWMxH/L with OVRDAT = 0b00, turning OFF PWM
 * outputs */
/* User output overrides are synchronized to next start of cycle */

PG3IOCONL = 0x3000;

/* Enable PWM generator 1, starting all PWM generators together */

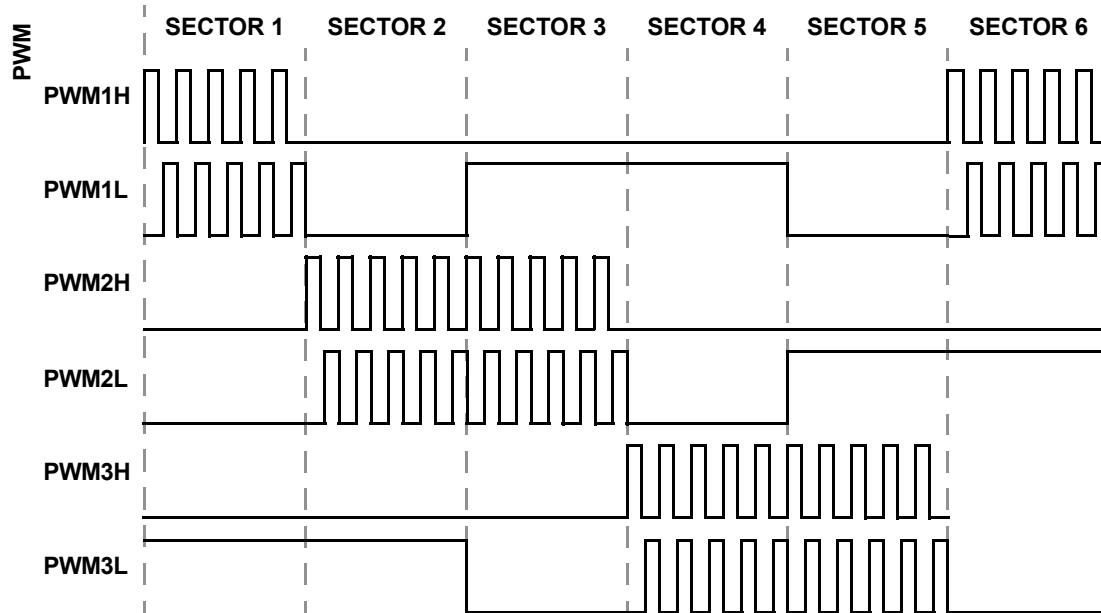
PG1CONLbits.ON = 1;
}

```

5.1.2 Six-Step Commutation – PWM Scheme 2

In this PWM scheme, three switches are used to control the two active phases. In a given sector, one active phase is driven with a complementary PWM waveform and the other active phase has only its low side driven low at 100% duty cycle, as shown in [Figure 5-3](#). Like Scheme 1, overrides are used to control the outputs in each sector.

Figure 5-3. Six-Step PWM Scheme 2 Waveform



In this scheme, Complementary Output mode is used and overridden as needed in each sector. The same three-phase host/client synchronization technique is used as in Scheme 1.

Configuration Summary:

- Independent Edge PWM mode
- Complementary Output mode
- Master Period and Duty Cycle Used
- Override State is Dependent on Sector State
- Dead time is applied to the Complementary PWM Signal

Example 5-2. Six-Step PWM Scheme 2 Code

```
#include<stdint.h>
uint16_t state = 0;
uint16_t PWMState1[6] = {0x1000, 0x1000, 0x3000, 0x2000, 0x2000, 0x3000};
uint16_t PWMState2[6] = {0x2000, 0x3000, 0x1000, 0x1000, 0x3000, 0x2000};
uint16_t PWMState3[6] = {0x3000, 0x2000, 0x2000, 0x3000, 0x1000, 0x1000};

int main()
{
    .....
    .....
    .....
    /* Function call to initialize PWM module */
    PWMInitialization();

    while(1)
    {
        for(state = 0; state < 6; state++)
        {
            /* Delay is used to simulate BLDC commutation; In practical
            application, commutation state transition will be based on feedback
            from Motor */
            Delay();
            PG1IOCONL = PWMState1[state];
            PG2IOCONL = PWMState2[state];
        }
    }
}
```

```

        PG3IOCONL = PWMState3[state];
    }
}

void      PWMInitialization(void)
{
    /* MASTER clock source is selected as High Speed PLL clock */
    PCLKCONbits.MCLKSEL = 0b00;

    /* Set PWM Period */

    MPER = 10000;

    /* Set Duty Cycle - 25% */

    MDC = 2500;

    /* Set Phase shift - No phase shift */

    MPHASE = 0;

    /* Select PWM Generator Duty Cycle Register as MDC */
    /* Select PWM Generator Duty Cycle Register as MPER */
    /* Select PWM Generator Duty Cycle Register as MPHASE */
    /* PWM Generator broadcasts software set of UPDREQ control */
    /* bit and EOC signal to other PWM Generators. */

    /* PWM Buffer Update Mode is at start of next PWM cycle if UPDREQ = 1 */
    /* PWM generator operates in Single Trigger Mode */
    /* Start of Cycle is local EOC */

    PG1CONH = 0xE800;

    /* PWM Generator is disabled */
    /* PWM Generator uses Master Clock selected by the PCLKCONbits.MCLKSEL bits */
    /* PWM Generator operates in Independent Edge PWM mode */

    PG1CONL = 0x0008;

    /* PWM Generator Output Mode is Complementary Mode */
    /* PWM Generator controls the PWMxH output pin */
    /* PWM Generator controls the PWMxL output pin */

    PG1IOCONH = 0x000C;

    /* Override is enabled on PWMxH/L with OVRDAT = 0b00, turning OFF PWM
       outputs */
    /* User output overrides are synchronized to next start of cycle */

    PG1IOCONL = 0x3000;

    /* PGxTRIGA register compare event is enabled as trigger source for ADC
       Trigger 1 */
    /* A write of the PGxDC register automatically sets the UPDREQ bit */
    /* PWM generator trigger output is EOC*/

    PG1EVTL = 0x0108

    /* Select PWM Generator Duty Cycle Register as MDC */
    /* Select PWM Generator Period Register as MPER */
    /* Select PWM Generator Phase Register as MPHASE */
    /* PWM generator does not broadcast UPDATE status bit or EOC signal
       * to other PWM generators */
    /* PWM Buffer Update Mode is client immediate*/
    /* PWM generator operates in Single Trigger Mode */
    /* Start of Cycle is PG1 trigger output selected by
       * PG1EVTbits.PGTRGSEL<2:0> bits */

    PG2CONH = 0xE301;

    /* PWM Generator is enabled */
    /* PWM Generator uses Master Clock selected by the PCLKCONbits.MCLKSEL bits */
    /* PWM Generator operates in Independent Edge PWM mode*/

    PG2CONL = 0x8008;

```

```

/* PWM Generator output operates in Complementary Mode */
/* PWM Generator controls the PWMxH output pin */
/* PWM Generator controls the PWMxL output pin */

PG2IOCONH = 0x000C;

/* Override is enabled on PWMxH/L with OVRDAT = 0b00, turning OFF PWM
   outputs */
/* User output overrides are synchronized to next start of cycle */

PG2IOCONL = 0x0300;

/* Select PWM Generator Duty Cycle Register as MDC */
/* Select PWM Generator Period Register as MPER */
/* Select PWM Generator Phase Register as MPHASE */
/* PWM generator does not broadcast UPDATE status bit or EOC signal
   * to other PWM generators */
/* PWM Buffer Update Mode is client immediate*/
/* PWM generator operates in Single Trigger Mode */
/* Start of Cycle is PGL trigger output selected by
   * PGL1EVTbits.PGTRGSEL<2:0> bits */

PG3CONH = 0xE301;

/* PWM Generator is enabled */
/* PWM Generator uses Master Clock selected by the PCLKCONbits.MCLKSEL bits */
/* PWM Generator operates in Independent Edge PWM mode*/

PG3CONL = 0x8008;

/* PWM Generator Output Mode is Complementary Mode */
/* PWM Generator controls the PWMxH output pin */
/* PWM Generator controls the PWMxL output pin */

PG3IOCONH = 0x000C;

/* Override is enabled on PWMxH/L with OVRDAT = 0b00, turning OFF PWM
   outputs */
/* User output overrides are synchronized to next start of cycle */

PG3IOCONL = 0x0300;

/* Enable PWM generator 1 ; starting all PWM generators together */

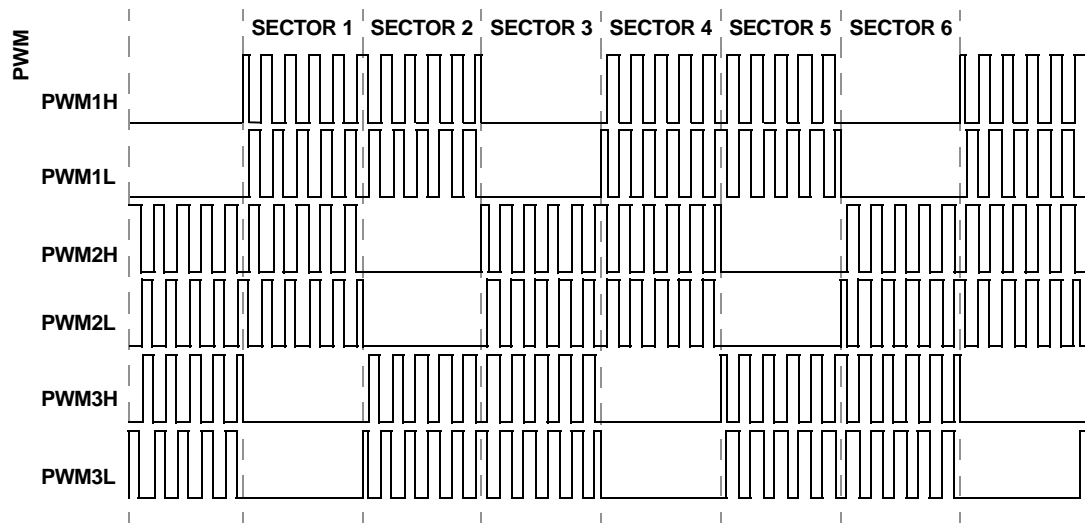
PG1CONLbits.ON = 1;
}

```

5.1.3 Six-Step Commutation – PWM Scheme 3

In this PWM scheme, four switches are driven in a given sector. Two pairs of complementary PWM outputs are applied to the two active phases. The inactive phase is overridden low as needed, as shown in [Figure 5-4](#).

Figure 5-4. Six-Step PWM Scheme 3 Waveform



In this scheme, Center-Aligned PWM mode is used with dead time to prevent high current during switching transitions. The two active phases are driven 180 degrees out of phase to one another using the SWAP feature.

Configuration Summary:

- Center-Aligned PWM mode
- Complementary Output mode
- Master Period and Duty Cycle Used
- Override and SWAP State are Dependent on Sector State
- Dead time is applied to the Complementary PWM Signal

Example 5-3. Six-Step PWM Scheme 3 Code

```
unsigned int state = 0, cycleCount = 0;
unsigned int PWMState1[6] = {0x0000, 0x3000, 0x3400, 0x3400, 0x3000, 0x0000};
unsigned int PWMState2[6] = {0x3000, 0x0000, 0x0000, 0x3000, 0x3400, 0x3400};
unsigned int PWMState3[6] = {0x3400, 0x3400, 0x3000, 0x0000, 0x0000, 0x3000};

int main(void)
{
    .....
    .....
    .....

    /* Function call to initialize PWM(PG1- PG3) Generators */
    PWMInitialization();

    /* To Update Duty cycle values to PG1-PG3, add two lines of code written
       below(In this example setting 50% Duty Cycle):

       After writing MDC register set Update request bit PG1STATbits.UPDREQ.
       This will transfer MDC value to all the PWM generators PG1-PG3.
       Note that Update Mode(UPDMOD) of PG2, PG3 is client EOC and
       PG1 MSTEN bits is '1' */
    MDC = 2500;
    PG1STATbits.UPDREQ = 1;

    /* Clear variables used in the _PWM1Interrupt() */
    state = 0;
    cycleCount = 0;
}
```

```

/* Enable PWM Generator 1 Interrupt */
_PWM1IE = 1;

while (1)
{
}

void PWMInitialization(void)
{
    PCLKCON      = 0x0000;
    /* PWM Clock Divider Selection bits
       0b11 = 1:16 ; 0b10 = 1:8 ; 0b01 = 1:4 ; 0b00 = 1:2*/
    PCLKCONbits.DIVSEL = 0;
    /* PWM Master Clock is clock source 0(Refer device data sheet for
    details)*/
    PCLKCONbits.MCLKSEL = 0;
    /* Lock bit: 0 = Write-protected registers and bits are unlocked */
    PCLKCONbits.LOCK = 0;

    /* Initialize Master Phase Register */
    MPHASE      = 0x0000;
    /* Initialize Master Duty Cycle */
    MDC         = 0x0000;
    /* Initialize Master Period Register */
    MPER        = 5000;

    /* Initialize PWM GENERATOR 1 CONTROL REGISTER LOW */
    /* Ensuring PWM Generator is disabled prior to configuring module */
    /* Macro uses Master clock selected by the PCLKCON.MCLKSEL bits*/
    /* Center-Aligned PWM mode(interrupt/register update once per cycle)*/
    PG1CONL     = 0x000C;

    /* Initialize PWM GENERATOR 1 CONTROL REGISTER HIGH */
    /* Macro uses the MDC register instead of PG1DC */
    /* Macro uses the MPER register instead of PG1PER */
    /* Macro uses the MPHASE register instead of PG1PHASE */
    /* PWM Generator broadcasts software set of UPDREQ */
    /* control bit and EOC signal to other PWM generators */
    /* Update Data registers at start of next PWM cycle if UPDREQ = 1. */
    /* PWM Generator operates in Single Trigger mode */
    /* Start of Cycle Selection is Local EOC*/
    PG1CONH     = 0xE800;

    /* Initialize PWM GENERATOR 1 I/O CONTROL REGISTER LOW */
    /* PWM1H/L signals are mapped to their respective pins */
    /* Override is enabled on PWMxH/L with OVRDAT = 0b00,
    turning OFF PWM outputs */
    /* User output overrides via the OVRENH/L and OVRDAT<1:0> bits are
    synchronized to the local PWM time base (next start of cycle)*/
    PG1IOCONL   = 0x3000;

    /* Initialize PWM GENERATOR 1 I/O CONTROL REGISTER HIGH */
    /* PWM Generator outputs operate in Complementary mode*/
    /* PWM Generator controls the PWM1H output pin */
    /* PWM Generator controls the PWM1L output pin */
    /* PWM1H Output Polarity is active-high*/
    /* PWM1L Output Polarity is active-high*/
    PG1IOCONH   = 0x000C;

    /* Initialize PWM GENERATOR 1 EVENT REGISTER LOW*/
    /* PG1TRIGA register compare event is enabled as trigger source for
    ADC Trigger 1 */
    /* User must set the PG1STATbits.UPDREQ bit (PGxSTAT<3>) manually */
    /* EOC event is the PWM Generator trigger*/
    PG1EVTL     = 0x0100;

    /* Initialize PWM GENERATOR 1 EVENT REGISTER HIGH */
    /* Interrupts CPU at EOC*/
    PG1EVTH     = 0x0000;

    /* Initialize PWM GENERATOR 1 STATUS REGISTER */
    PG1STAT     = 0x0000;

    /* Initialize PWM GENERATOR 1 DEAD-TIME REGISTER LOW */
    PG1DTL      = 0x64;

```



```

/* Initialize PWM GENERATOR 1 DEAD-TIME REGISTER HIGH */
PG1DTH      = 0x64;

/* Initialize PWM GENERATOR 1 TRIGGER A REGISTER */
PG1TRIGA     = 0;

/* Initialize PWM GENERATOR 2 CONTROL REGISTER LOW */
/* Ensuring PWM Generator is disabled prior to configuring module */
/* Macro uses Master clock selected by the PCLKCON.MCLKSEL bits*/
/* Center-Aligned PWM mode(interrupt/register update once per cycle)*/
PG2CONL      = 0x000C;

/* Initialize PWM GENERATOR 2 CONTROL REGISTER HIGH */
/* Macro uses the MDC register instead of PG2DC */
/* Macro uses the MPER register instead of PG2PER */
/* Macro uses the MPHASE register instead of PG2PHASE */
/* PWM Generator does not broadcast UPDATE status bit or EOC signal */
/* Client SOC :Update Data registers at start of next cycle if a host
update request is received

```

5.2 Three-Phase Sinusoidal Control of PMSM/ACIM Motors

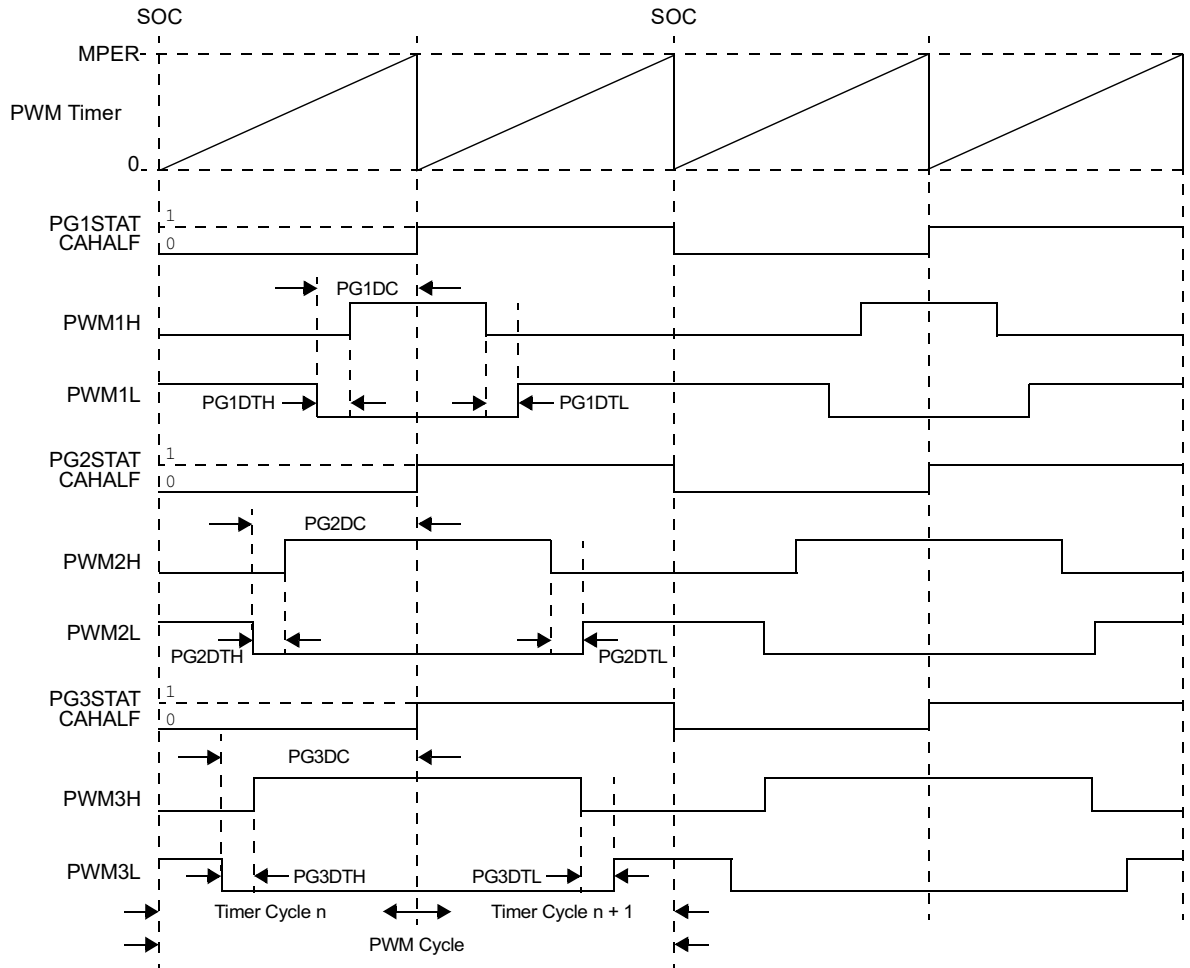
Three-phase sinusoidal control applies voltages to the three-phase motor windings, which are Pulse-Width Modulated to produce sinusoidal currents as the motor spins. This eliminates the torque ripple and commutation spikes associated with trapezoidal commutation. Typically, Center-Aligned Complementary PWMs are used for sinusoidal control of a Permanent Magnet Synchronous Motor (PMSM) or three-phase AC Induction Motor (ACIM). Center-aligned PWM signals reduce the level of harmonics in output voltages and currents as compared to edge-aligned PWMs. Three PWM Generators are connected to the three-phase power bridge driving the motor, as shown in [Figure 5-1](#).

PWM Generator 1 is configured as host, and PWM Generator 2 and PWM Generator 3 are configured as client PWMs. PWM configuration used in three-phase sinusoidal control is summarized below:

- PG1-PG3: Uses master period and independent duty cycles
- PG1-PG3: PWM Operating mode is selected as Center-Aligned mode
- PG1-PG3: Configured to operate in Single Trigger mode
- PG1-PG3: PWM Output mode is configured as Complementary Output mode
- PG2-PG3: Uses PG1 trigger output as Start-of-Cycle, whereas PG1 is self-triggered
- PG2-PG3: Enabled during initialization
- PG1 is enabled only after configuring all the control registers; whenever PG1 is enabled, all the generators will start in unison

[Figure 5-5](#) shows the PWM waveforms for a given point in time. Center-Aligned mode uses two timer cycles to produce a PWM cycle and maintains symmetry at the center of each PWM cycle. Each timer cycle can be tracked using the status bit, CAHALF (PGxSTAT[1]), of the respective PWM Generator. The leading edge is produced when CAHALF = 0 and the falling edge is produced when CAHALF = 1. Note that with Center-Aligned mode, as long as the duty cycles are different for each phase, the switching instants occur at different times. (In Edge-Aligned mode, the turn-on times are coincident.) This generally reduces electromagnetic interference.

Figure 5-5. Three-Phase Center-Aligned Waveforms



Example 5-4. Three-Phase Sinusoidal PMSM/ACIM Motor Control Code

```
/* Master clock source is selected as High speed PLL clock */
PCLKCONbits.MCLKSEL = 0b00;

/* Set PWM Phase Register - No phase shift */
MPHASE = 0;

/* Set PWM Period */
MPER = 5000;

/* Set PWM Duty Cycles */
PG1DC = 1250;
PG2DC = 2500;
PG3DC = 3750;

/* Set Dead Time Registers */
PG1DTL = 200;
PG1DTH = 200;
PG2DTL = 200;
PG2DTH = 200;
PG3DTL = 200;
PG3DTH = 200;
```

```

/* Select PWM Generator Duty Cycle Register as PG1DC */
/* Select PWM Generator Period Register as MPER */
/* Select PWM Generator Phase Register as MPHASE */
/* PWM Generator broadcasts software set of UPDREQ */
/* control bit and EOC signal to other PWM generators. */
/* PWM Buffer Update Mode is at start of next PWM cycle if UPDREQ = 1 */
/* PWM generator operates in Single Trigger Mode */
/* Start of Cycle is Local EOC */

PG1CONH = 0x6800;

/* PWM Generator is disabled */
/* PWM Generator uses Master Clock selected by
   the PCLKCONbits.MCLKSEL bits */
/* PWM Generator operates in Center-Aligned mode*/

PG1CONL = 0x000C;

/* PWM Generator Output operates in Complementary Mode */
/* PWM Generator controls the PWMxH output pin */
/* PWM Generator controls the PWMxL output pin */

PG1IOCONH = 0x000C;

/* PGxTRIGA register compare event is enabled as trigger source for
   ADC Trigger 1 */
/* A write of the PGxDC register automatically sets the UPDREQ bit */
/* PWM generator trigger output is EOC*/

PG1EVTL = 0x0108;

/* Select PWM Generator Duty Cycle Register as PG2DC */
/* Select PWM Generator Period Register as MPER */
/* Select PWM Generator Phase Register as MPHASE */
/* PWM generator does not broadcast UPDATE status bit or EOC signal
   to other PWM generators */
/* PWM Buffer Update Mode is client immediate*/
/* PWM generator operates in Single Trigger Mode */
/* Start of Cycle is PG1 trigger output selected by
   PG1EVThbits.PGTRGSEL<2:0> bits */

PG2CONH = 0x6301;

/* PWM Generator is enabled */
/* PWM Generator uses Master Clock selected by
   the PCLKCONbits.MCLKSEL bits */
/* PWM Generator operates in Center-Aligned mode */

PG2CONL = 0x800C;

/* PWM Generator output operates in Complementary Mode */
/* PWM Generator controls the PWMxH output pin */
/* PWM Generator controls the PWMxL output pin */

PG2IOCONH = 0x000C;

/* Select PWM Generator Duty Cycle Register as PG3DC */
/* Select PWM Generator Period Register as MPER */
/* Select PWM Generator Phase Register as MPHASE */
/* PWM generator does not broadcast UPDATE status bit or EOC signal to
   other PWM generators */
/* PWM Buffer Update Mode is client immediate*/
/* PWM generator operates in Single Trigger Mode */
/* Start of Cycle is PG1 trigger output selected by
   PG1EVThbits.PGTRGSEL<2:0> bits */

PG3CONH = 0x6301;

/* PWM Generator is enabled */
/* PWM Generator uses Master Clock selected by the PCLKCONbits.MCLKSEL bits */
/* PWM Generator operates in Center-Aligned mode */

PG3CONL = 0x800C;

/* PWM Generator output operates in Complementary Mode */
/* PWM Generator controls the PWMxH output pin */
/* PWM Generator controls the PWMxL output pin */

```

```
PG3IOCONH = 0x000C;

/* Turning ON the PWM Generator 1;
   Thus starting all the PWM modules in unison */

PG1CONLbits.ON = 1;
```

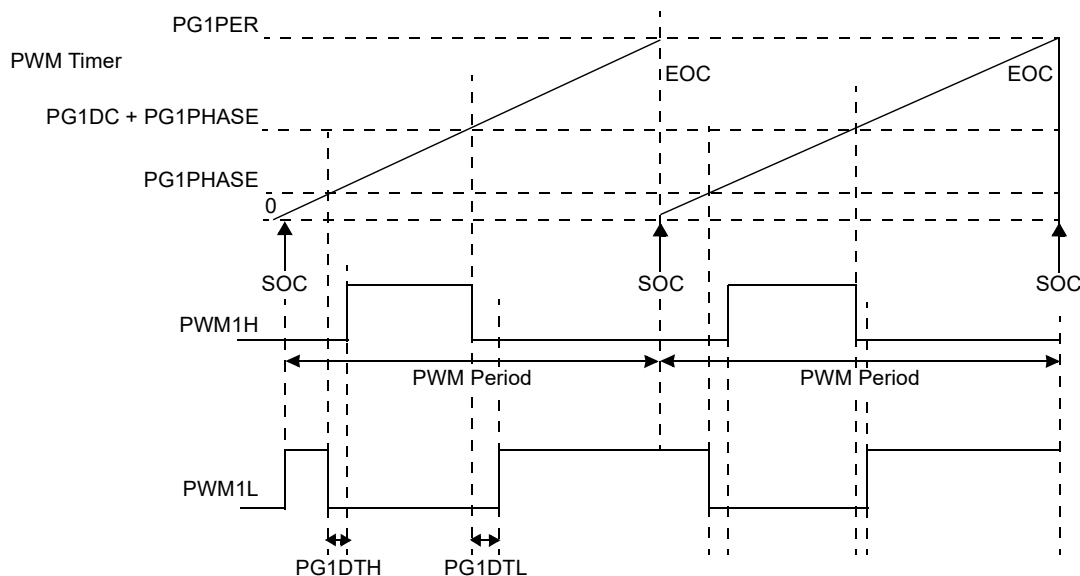
5.3 Simple Complementary PWM Output

This complementary PWM example uses a single PWM Generator and can be used for half-bridge applications. The PWM is configured as follows:

- Independent Edge PWM mode
- Complementary Output mode
- Self-Triggered mode

Figure 5-6 shows the timing relations of the PWM signals. In this example, continuous triggering (local EOC) is used in addition to a phase offset (PG1PHASE). Dead time is implemented to prevent simultaneous switch conduction.

Figure 5-6. Timing Diagram for Complementary and Local EOC Triggered PWM Output



Example 5-5. Complementary PWM Output Mode

```
/*PWM control register configuration*/
PCLKCONbits.MCLKSEL = 3;
PG1CONLbits.CLKSEL = 1;
PG1CONLbits.MODSEL = 0b0000; /*Independent edge triggered mode */
PG1CONH = 0x0000;
/*PWM Generator outputs operate in Complementary mode*/
/*PWM Generator controls the PWM1H and PWM1L output pins*/
/*PWM1H & PWM1L output pins are active high*/
PG1IOCONH = 0x000C;
/*PWM uses PG1DC, PG1PER, PG1PHASE registers*/
/*PWM Generator does not broadcast UPDATE status bit state or EOC signal*/
/*Update the data registers at start of next PWM cycle (SOC) */
/*PWM Generator operates in Single Trigger mode*/
/*Start of cycle (SOC) = local EOC*/
/*Write to DATA REGISTERS*/
PG1PER = 5000; /*PWM frequency is 100kHz*/
PG1DC = 1250; /* 25% duty*/
PG1PHASE = 500; /*Phase offset in rising edge of PWM*/
```

```

PG1DTH      = 100;          /*Dead time on PWMH */
PG1DTL      = 100;          /*Dead time on PWML*/
/*Enable PWM*/
PG1CONLbits.ON      = 1;          /*PWM module is enabled*/

```

5.4 Cycle-by-Cycle Current Limit Mode

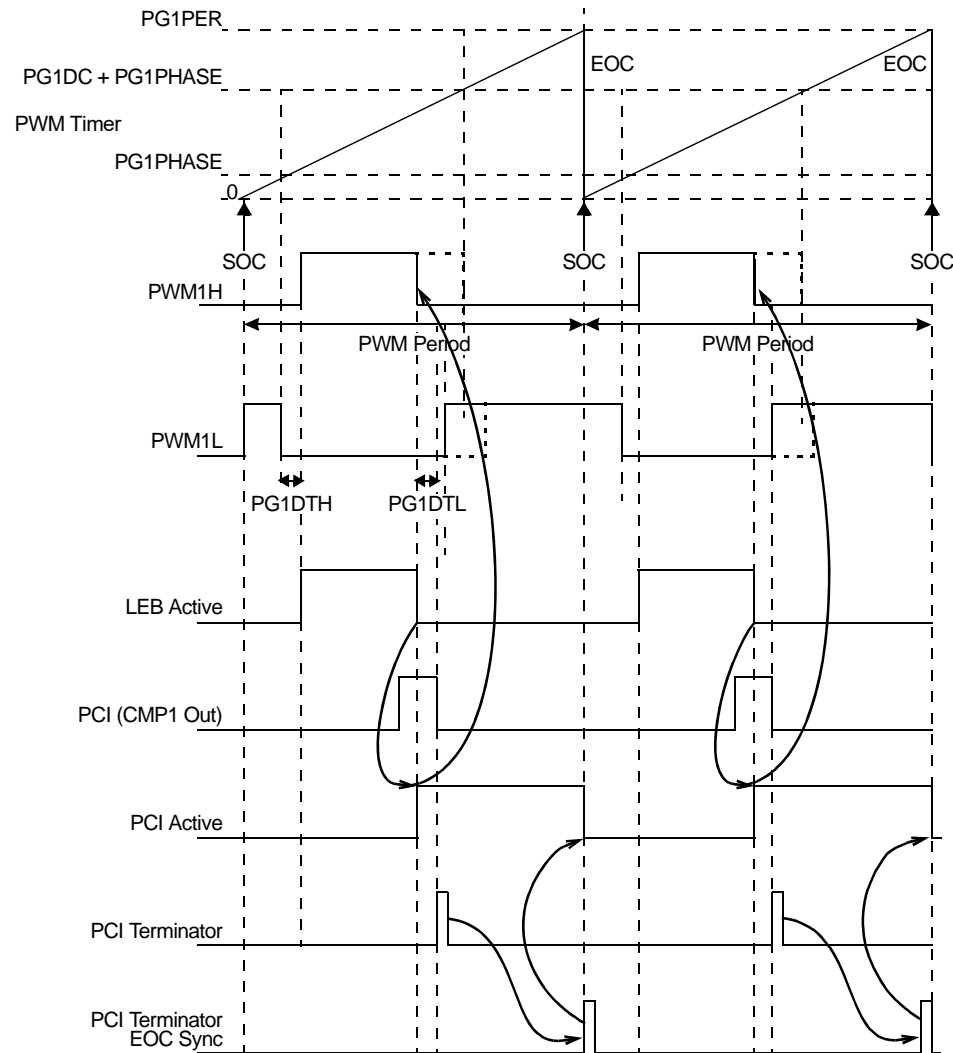
Cycle-by-Cycle Current Limit mode is a widely adopted control strategy for power applications and motor control. Current is measured and limited to a predetermined level using the internal comparator module. Cycle-by-Cycle Current Limit mode control for BLDC motors can automatically limit motor phase currents to a predetermined maximum value, using a comparator to provide an input to the PCI block. This has the advantage of allowing operation to continue when the limit is reached, rather than triggering a Fault. The PWM is configured as follows:

- Independent Edge PWM mode
- Complementary Output mode
- Self-Triggered mode

The current limit PCI block logic is used to control the cycle truncation. The Leading-Edge Blanking feature is used to filter out switching transients and slightly delays cycle truncation, as shown with the upper arrows in [Figure 5-7](#). The duty cycle is truncated when the PCI active signal goes high.

To reset the PCI block for the next cycle, the PCI terminator is configured to detect the falling edge of the comparator (CMP1 Out). The terminator signal is then synchronized to the End-of-Cycle (EOC) and the PCI active signal is reset, as shown with the lower arrows in [Figure 5-7](#).

Figure 5-7. Timing Diagram for Self-Triggered, Complementary Output and Current Limit Cycle-by-Cycle PWM Modes



Example 5-6. Cycle-by-Cycle Current Limit Mode

```
/*PWM control register configuration*/
PCLKCONbits.MCLKSEL = 3;
PG1CONLbits.CLKSEL = 1;
PG1CONLbits.MODSEL = 0b000; /*Independent edge triggered mode */
PG1CONH = 0x0000;
/*PWM Generator outputs operate in Complementary mode*/
/*PWM Generator controls the PWM1H and PWM1L output pins*/
/*PWM1H & PWM1L output pins are active high*/
PG1IOCONH = 0x000C;
/*PWM uses PG1DC, PG1PER, PG1PHASE registers*/
/*PWM Generator does not broadcast UPDATE status bit state or EOC signal*/
/*Update the data registers at start of next PWM cycle (SOC) */
/*PWM Generator operates in Single Trigger mode*/
/*Start of cycle (SOC) = local EOC*/
/*Write to DATA REGISTERS */
PG1PER = 5000; /*PWM frequency is 100kHz */
PG1DC = 2500; /* 50% duty cycle*/
PG1PHASE = 500; /*Phase offset in rising edge of PWM*/
PG1DTH = 100; /*Dead time on PWMH */
PG1DTL = 100; /*Dead time on PWML*/
PG1LEBH = 0x0008; /* PHR=1, Rising edge of PWM1H will trigger
```

```

                                the LEB counter*/
PG1LEBL                        = 200;    /*LEB=200*/

/*PCI logic configuration for current limit cycle by cycle mode, comparator 1
output as PCI source*/
PG1IOCONL                      = 0x0010;
/*CLDAT=0b01, 1 on PWM1L and 0 on PWM1H*/
PG1CLPCIL                      = 0x1A1B;
/* TERM=0b001, Terminate when PCI source transitions from active to inactive*/
/* TSYNCDIS=0, Termination of latched PCI delays till PWM EOC (for Cycle by
cycle mode) */
/*AQSS=0b010, LEB active is selected as acceptance qualifier */
/*AQPS=1, LEB active is inverted to accept PCI signal when LEB duration is
over*/
/*PSYNC=0, PCI source is not synchronized to PWM EOC so that current limit
resets PWM immediately*/
/*PSS=0bxxxx, ACMP1 out is selected as PCI source signal */
/*PPS=0; PCI source signal is not inverted*/
PG1CLPCIH                      = 0x0300;
/*ACP=0b011, latched PCI is selected as acceptance criteria to work when compl
out is active*/
/*TQSS=0b000, No termination qualifier used so terminator will work straight
away without any qualifier*/
/*Enable PWM*/
PG1CONLbits.ON                 = 1;      /*PWM module is enabled*/
    
```

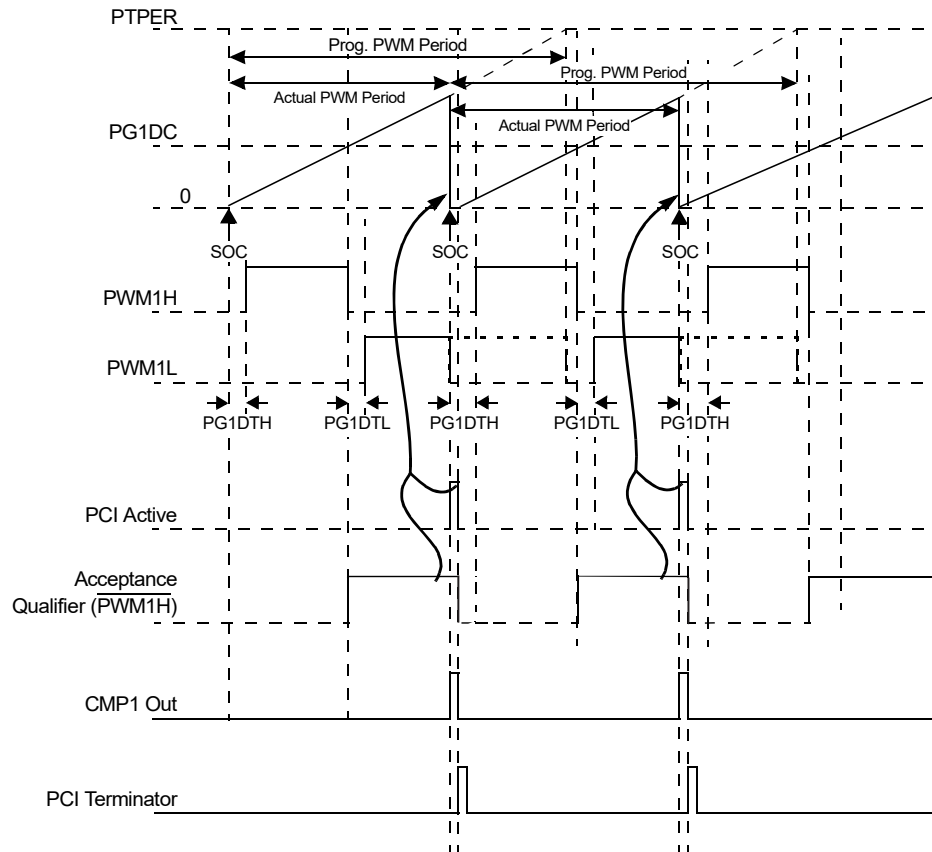
5.5 External Period Reset Mode

External Period Reset mode for power control monitors inductor current and varies the PWM period to control power delivery. When the inductor current returns to '0', the PWM cycle will restart. The PWM is configured as follows:

- Independent Edge PWM mode
- Complementary Output mode
- Self-Triggered mode

The initial programmed PWM period may be shortened depending on the inductor current compared to a predetermined trip point. The Sync PCI block is used to control cycle truncation. The comparator (CMP1 Out) output is used as the input to the Sync PCI block and an inverted PWM1H signal is used as a qualifier to allow truncation only on the duty cycle inactive time of a cycle, as shown by the arrows in [Figure 5-8](#). The PCI block is reset for the next cycle using the auto-terminate function.

Figure 5-8. Timing Diagram for Self-Triggered, Complementary Output and Current Reset PWM Modes



Example 5-7. External Period Reset Mode

```
/*PWM control register configuration*/
PCLKCONbits.MCLKSEL    = 3;
PG1CONLbits.CLKSEL     = 1;
PG1CONLbits.MODSEL     = 0b000;          /*Independent edge triggered mode */
PG1CONH                = 0x0040
/*PWM Generator outputs operate in Complementary mode*/
/*PWM Generator controls the PWM1H and PWM1L output pins*/
/*PWM1H & PWM1L output pins are active high*/
PG1IOCONH              = 0x000C;
/*PWM uses PG1DC, PG1PER, PG1PHASE registers*/
/*PWM Generator does not broadcast UPDATE status bit state or EOC signal*/
/*Update the data registers at start of next PWM cycle (SOC) */
/*PWM Generator operates in Re-Triggerable mode*/
/*Start of cycle (SOC) = local EOC is OR'd with PCI sync*/
/*Write to DATA REGISTERS */
PG1PER                 = 5000;           /*PWM frequency is 100kHz */
PG1DC                  = 1250;           /*25%duty*/
PG1PHASE               = 0;             /*No Phase offset in rising edge of
PWM*/
PG1DTH                 = 100;           /*dead time on PWMH */
PG1DTL                 = 100;           /*dead time on PWML*/
PG1LEBH                = 0x0008;        /* PHR=1, Rising edge of PWM1H will
trigger the LEB counter*/
PG1LEBL                = 200;           /*LEB=200*/

/*PCI logic configuration for current reset (PCI sync mode), comparator 1
output (current reset signal) as PCI source and PWM1H falling edge as
acceptance qualifier*/
LOGCON1L               = 0x0000;
/*PWM1out
PG1SPC1L               = 0x942C;
```



```

/* TERM=0b001, Terminate when PCI source transitions from active to inactive*/
/* TSYNCDIS=1, Termination of latched PCI occurs immediately*/
/*AQSS=0b100, PWM1H is selected as acceptance qualifier because PWM should be
reset in OFF time */
/*AQPS=0, PWM1H is inverted to accept PCI signal when PWM1H on time is over*/
/*PSYNC=0, PCI source is not synchronized to PWM EOC so that current limit
resets PWM immediately*/
/*PSS=0b01100, ACMP1 out is selected as PCI source signal */
/*PPS=1; PCI source signal is inverted*/
PG1SPCIH = 0x0300;
/*ACP=0b011, Latched PCI is selected as acceptance criteria to work when compl
out is active*/
/*TQSS=0b000, No termination qualifier used so terminator will work straight
away without any qualifier*/
/*Enable PWM*/
PG1CONLbits.ON = 1; /*PWM module is enabled*/

```

6. Interrupts

The interrupt sources within the PWM system are routed to the CPU in one of two ways:

1. Through a shared signal for each PWM Generator (see [4.2.11.2. Event Interrupts](#)). The signals include:
 - a. EOC event
 - b. TRIGA compare event
 - c. ADC Trigger 1 event
 - d. PCI_active event
2. Through the PWM Event Output block (see [4.3.6. PWM Event Outputs](#)) are various sources of interrupts that can be generated by the PWM module.

For example, a device which has eight PWM Generators and six PWM event outputs would have a total of fourteen independent interrupt sources.

Since the PWM module can operate at a much higher frequency than the CPU system clock, care should be taken with the interrupt event configuration. Successive interrupt events on the same interrupt vector, that occur at a rate greater than the CPU can detect and service them, may result in missed processing and unexpected results. This limitation is dependent on the relationship of the system clock frequency, PWM operating frequency and the execution time of the Interrupt Service Routine (# of instructions is irrelevant, what matters is how long the ISR takes to execute before it yields control back to the interrupted thread).

Interrupts from different sources that occur in close proximity to each other will also not be detected by the CPU as separate interrupt events. Therefore, it is good software practice to check the PWM status flags to differentiate a PCI interrupt event from a PWM time base interrupt. In some cases, it is desirable to separate different types of interrupt events that could be produced by the PWM Generator. When multiple PWM Generators have been synchronized together, it is possible to enable the time base interrupt on a separate PWM Generator that is synchronized to the host PWM Generator. Using this method, the time base interrupt can be serviced by a separate interrupt vector. It is also possible to bring various PWM event signals outside of the PWM module via the PWM Event blocks to an external off-chip destination (see [4.3.6. PWM Event Outputs](#)).

7. Operation in Power-Saving Modes

This section discusses the operation of the High-Speed PWM module in Sleep mode and Idle mode.

7.1 Operation in Sleep Mode

When the device enters Sleep mode, the clocks available to the PWM are disabled. All enabled PWM output pins that were in operation prior to entering Sleep mode will be frozen in their current output states. If the application circuit can be damaged by this condition, the outputs must be placed into a safe state before executing the `PWRSV` instruction to enter Sleep mode.

7.2 Operation in Idle Mode

When the device enters Idle mode, the CPU is no longer clocked; however, the PWM remains clocked and operational. If the PWM module is controlling a power conversion/motor control application, the action of putting the device into Idle mode will cause any control loops to be disabled, and most applications will likely experience issues unless they are explicitly designed to operate in an Open-Loop mode. It is recommended that the outputs be placed into a safe state before executing the `PWRSV` instruction to enter Idle mode.

8. Related Application Notes

This section lists application notes that are related to this section of the manual. These application notes may not be written specifically for the dsPIC33 device families, but the concepts are pertinent and could be used with modification and possible limitations. The current application notes related to the High-Resolution PWM with Fine Edge Placement module are:

No related application notes at this time.

Note: Please visit the Microchip website (www.microchip.com) for additional application notes and code examples for the dsPIC33 families of devices.

9. Revision History

9.1 Revision A (August 2017)

This is the initial release of this document.

9.2 Revision B (February 2018)

Changed document header from dsPIC33CH_PWM to dsPIC33/PIC24 FRM.

Updated the Master Period Register and PGCxCONH Registers.

Updated Figure 4-10: PWMxH/PWMxL Rising and Falling Edges Due to Dead Time

Updated Section 4.3.10 Synchronizing Multiple PWM Generator Buffer

Updated the Six-Step PWM Scheme 1 Code example, Six-Step PWM Scheme 3 Code example and the Three-Phase Sinusoidal PMSM/ACIM Motor Control Code example.

9.3 Revision C (February 2019)

Removed references to dsPIC33CH devices.

Changed bit brackets ([]).

Updated LOGCONy and PGxIOCONL registers

Updated Master Clocking, Clocking Synchronization, Shared Clocking, Push-Pull Output Mode, High-Resolution Mode, PWM Event Outputs, Leading-Edge Blanking Period, Leading-Edge Blanking Counter Period Calculation and High-Resolution Mode sections.

Added Clocking equations in Standard Resolution section. Added PWM Duty Cycle, Phase, Trigger and Dead-Time Calculations, High Resolution equation in the High-Resolution Mode section.

Updated Leading-Edge Blanking Period equation.

Updated PWM Event Output Function figure.

Added Dead-Time Calculation in Standard Resolution example in the Dead Time section. Added Leading-Edge Blanking in Standard and High Resolution. Added example in the Leading-Edge Blanking Counter Period Calculation section. Added the High-Resolution Calculation example in the High-Resolution Mode section.

9.4 Revision D (December 2020)

Updated Figure 3-2: Single PWM Generator, Figure 3-3: Basic PWM Waveform, Figure 4-2: Independent Edge PWM Mode, Figure 4-3: Variable Phase PWM Mode, Figure 4-5: Dual PWM Mode, Figure 4-6: Center-Aligned PWM Mode, Figure 4-7: Double Update Center-Aligned Mode, Figure 4-8: Dual Edge Center-Aligned PWM Mode (MODSEL[2:0] = 110), Figure 4-9: Dual Edge Center-Aligned PWM Mode (MODSEL[2:0] = 111), Figure 4-11: Override and SWAP Signal Flow, Complementary Mode, Figure 4-12: Push-Pull PWM, Figure 4-13: Push-Pull PWM: Center-Aligned Mode, Dual Edge Center-Aligned Mode with One Update per Cycle (MODSEL[2:0] = 110), Figure 4-14: Push-Pull PWM: Double Update Center-Aligned Mode, Dual Edge Center-Aligned Mode with Four Updates per Cycle (MODSEL[2:0] = 111), Figure 4-15: Override and SWAP Signal Flow, Push-Pull Output Mode, Figure 4-16: PCI Function Block Diagram, Figure 4-18: PWM Source Selection for PCI, Figure 4-19: Latch Mode Control, Figure 4-26: PWMxH/PWMxL Rising and Falling Edges Due to Dead Time, Figure 4-27: Adding PGxDCA Value to the PGxDC Register Value, Figure 4-30: Immediate Update Correction Examples, Figure 5-5: Three-Phase Center-Aligned Waveforms, Figure 5-6: Timing Diagram for Complementary and Local EOC Triggered PWM Output and Figure 5-7: Timing Diagram for Self-Triggered, Complementary Output and Current Limit Cycle-by-Cycle PWM Modes.

Added subsection: Output Behavior at Start-up in Complementary Mode.

Updated Section 3. Architecture Overview, Section 4.1.1: Master Clocking, Section 4.1.3.1: Data Registers in High Resolution, Section 4.2.2.2: Variable Phase PWM Mode, Section 4.2.2.3: Dual PWM Mode, Section 4.2.2.4: Center-Aligned PWM Mode, Section 4.2.2.5: Double Update Center-Aligned PWM Mode, Section 4.2.3.1: Complementary Output Mode, Section 4.2.3.3: Push-Pull Output Mode, Section 4.2.4: PWM Generator Triggers, Section 4.2.10: ADC Triggers, Section 4.2.11: Event Selection Block, Section 4.2.11.2: Event Interrupts, Section 4.2.11.2: Immediate Updates and Section 4.2.12: Time Base Capture.

Updated Table 4-1: PWM Data Registers, High-Resolution Mode.

Updated Example 5-1: Six-Step PWM Scheme 1 Code, Example 5-2: Six-Step PWM Scheme 2 Code and Example 5-4: Three-Phase Sinusoidal PMSM/ACIM Motor Control Code.

9.5 Revision E (February 2024)

Updated Figure 3-2. Single PWM Generator and Figure 4-31. Immediate Update Correction Examples.

Added Figure 4-3. Multiphase System Using TRIGx to Offset PGs

Updated Sections 4.2.2.1 Independent Edge PWM Mode, 4.2.2.2 Variable Phase PWM Mode, 4.2.2.4 Center-Aligned PWM Mode, 4.2.2.6 Dual Edge Center-Aligned PWM Mode, 4.2.4.2 Triggering Modes, 4.2.4.3 Burst Mode, 4.2.6 Dead Time, 4.2.8 Override, 4.2.11 Event Selection Block and 4.2.11.1 PWM Generator Trigger Output.

Added Sections 4.2.2.1.1 Multiphase Systems in Independent Mode, 4.2.8.1 Synchronizing overrides on multiple PWM Generators and 4.2.12.2 Immediate Updates.

Added subsection IMMEDIATE UPDATE TO PHASE.

Updated Section 3.

Updated Table 4-9. PWM Register Map (Lock and Write Restrictions).

Updated Example 5-3. Six-Step PWM Scheme 3 Code

The Microchip Website

Microchip provides online support via our website at www.microchip.com/. This website is used to make files and information easily available to customers. Some of the content available includes:

- **Product Support** – Data sheets and errata, application notes and sample programs, design resources, user's guides and hardware support documents, latest software releases and archived software
- **General Technical Support** – Frequently Asked Questions (FAQs), technical support requests, online discussion groups, Microchip design partner program member listing
- **Business of Microchip** – Product selector and ordering guides, latest Microchip press releases, listing of seminars and events, listings of Microchip sales offices, distributors and factory representatives

Product Change Notification Service

Microchip's product change notification service helps keep customers current on Microchip products. Subscribers will receive email notification whenever there are changes, updates, revisions or errata related to a specified product family or development tool of interest.

To register, go to www.microchip.com/pcn and follow the registration instructions.

Customer Support

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Embedded Solutions Engineer (ESE)
- Technical Support

Customers should contact their distributor, representative or ESE for support. Local sales offices are also available to help customers. A listing of sales offices and locations is included in this document.

Technical support is available through the website at: www.microchip.com/support

Microchip Devices Code Protection Feature

Note the following details of the code protection feature on Microchip devices:

- Microchip products meet the specifications contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is secure when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods being used in attempts to breach the code protection features of the Microchip devices. We believe that these methods require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Attempts to breach these code protection features, most likely, cannot be accomplished without violating Microchip's intellectual property rights.
- Microchip is willing to work with any customer who is concerned about the integrity of its code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of its code. Code protection does not mean that we are guaranteeing the product is "unbreakable." Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip's code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized

access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

Legal Notice

Information contained in this publication is provided for the sole purpose of designing with and using Microchip products. Information regarding device applications and the like is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications.

THIS INFORMATION IS PROVIDED BY MICROCHIP "AS IS". MICROCHIP MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WHETHER EXPRESS OR IMPLIED, WRITTEN OR ORAL, STATUTORY OR OTHERWISE, RELATED TO THE INFORMATION INCLUDING BUT NOT LIMITED TO ANY IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE OR WARRANTIES RELATED TO ITS CONDITION, QUALITY, OR PERFORMANCE.

IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE, INCIDENTAL OR CONSEQUENTIAL LOSS, DAMAGE, COST OR EXPENSE OF ANY KIND WHATSOEVER RELATED TO THE INFORMATION OR ITS USE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS IN ANY WAY RELATED TO THE INFORMATION OR ITS USE WILL NOT EXCEED THE AMOUNT OF FEES, IF ANY, THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THE INFORMATION. Use of Microchip devices in life support and/or safety applications is entirely at the buyer's risk, and the buyer agrees to defend, indemnify and hold harmless Microchip from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any Microchip intellectual property rights unless otherwise stated.

Trademarks

The Microchip name and logo, the Microchip logo, Adaptec, AnyRate, AVR, AVR logo, AVR Freaks, BesTime, BitCloud, chipKIT, chipKIT logo, CryptoMemory, CryptoRF, dsPIC, FlashFlex, flexPWR, HELDO, IGLOO, JukeBlox, KeeLoq, Kleer, LANCheck, LinkMD, maxStylus, maxTouch, MediaLB, megaAVR, Microsemi, Microsemi logo, MOST, MOST logo, MPLAB, OptoLyzer, PackeTime, PIC, picoPower, PICSTART, PIC32 logo, PolarFire, Prochip Designer, QTouch, SAM-BA, SenGenuity, SpyNIC, SST, SST Logo, SuperFlash, Symmetricom, SyncServer, Tachyon, TimeSource, tinyAVR, UNI/O, Vectron, and XMEGA are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

AgileSwitch, APT, ClockWorks, The Embedded Control Solutions Company, EtherSynch, FlashTec, Hyper Speed Control, HyperLight Load, IntelliMOS, Libero, motorBench, mTouch, Powermite 3, Precision Edge, ProASIC, ProASIC Plus, ProASIC Plus logo, Quiet-Wire, SmartFusion, SyncWorld, Temux, TimeCesium, TimeHub, TimePictra, TimeProvider, WinPath, and ZL are registered trademarks of Microchip Technology Incorporated in the U.S.A.

Adjacent Key Suppression, AKS, Analog-for-the-Digital Age, Any Capacitor, AnyIn, AnyOut, Augmented Switching, BlueSky, BodyCom, CodeGuard, CryptoAuthentication, CryptoAutomotive, CryptoCompanion, CryptoController, dsPICDEM, dsPICDEM.net, Dynamic Average Matching, DAM, ECAN, Espresso T1S, EtherGREEN, IdealBridge, In-Circuit Serial Programming, ICSP, INICnet, Intelligent Paralleling, Inter-Chip Connectivity, JitterBlocker, maxCrypto, maxView, memBrain, Mindi, MiWi, MPASM, MPF, MPLAB Certified logo, MPLIB, MPLINK, MultiTRAK, NetDetach, Omniscient Code Generation, PICDEM, PICDEM.net, PICkit, PICtail, PowerSmart, PureSilicon, QMatrix, REAL ICE, Ripple Blocker, RTAX, RTG4, SAM-ICE, Serial Quad I/O, simpleMAP, SimpliPHY, SmartBuffer, SMART-I.S., storClad, SQL, SuperSwitcher, SuperSwitcher II, Switchtec, SynchroPHY, Total Endurance, TSHARC, USBCheck, VariSense, VectorBlox, VeriPHY, ViewSpan, WiperLock, XpressConnect, and ZENA are trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

SQTP is a service mark of Microchip Technology Incorporated in the U.S.A.

The Adaptec logo, Frequency on Demand, Silicon Storage Technology, and Symmcom are registered trademarks of Microchip Technology Inc. in other countries.

GestlC is a registered trademark of Microchip Technology Germany II GmbH & Co. KG, a subsidiary of Microchip Technology Inc., in other countries.

All other trademarks mentioned herein are property of their respective companies.

© 2017-2024, Microchip Technology Incorporated, Printed in the U.S.A., All Rights Reserved.

ISBN: 978-1-6683-3744-8

Quality Management System

For information regarding Microchip's Quality Management Systems, please visit www.microchip.com/quality.

Worldwide Sales and Service

| AMERICAS | ASIA/PACIFIC | ASIA/PACIFIC | EUROPE |
|--|--|---|---|
| Corporate Office 2355 West Chandler Blvd. Chandler, AZ 85224-6199 Tel: 480-792-7200 Fax: 480-792-7277 Technical Support: www.microchip.com/support Web Address: www.microchip.com | Australia - Sydney Tel: 61-2-9868-6733 China - Beijing Tel: 86-10-8569-7000 China - Chengdu Tel: 86-28-8665-5511 China - Chongqing Tel: 86-23-8980-9588 China - Dongguan Tel: 86-769-8702-9880 China - Guangzhou Tel: 86-20-8755-8029 China - Hangzhou Tel: 86-571-8792-8115 China - Hong Kong SAR Tel: 852-2943-5100 China - Nanjing Tel: 86-25-8473-2460 China - Qingdao Tel: 86-532-8502-7355 China - Shanghai Tel: 86-21-3326-8000 China - Shenyang Tel: 86-24-2334-2829 China - Shenzhen Tel: 86-755-8864-2200 China - Suzhou Tel: 86-186-6233-1526 China - Wuhan Tel: 86-27-5980-5300 China - Xian Tel: 86-29-8833-7252 China - Xiamen Tel: 86-592-2388138 China - Zhuhai Tel: 86-756-3210040 | India - Bangalore Tel: 91-80-3090-4444 India - New Delhi Tel: 91-11-4160-8631 India - Pune Tel: 91-20-4121-0141 Japan - Osaka Tel: 81-6-6152-7160 Japan - Tokyo Tel: 81-3-6880-3770 Korea - Daegu Tel: 82-53-744-4301 Korea - Seoul Tel: 82-2-554-7200 Malaysia - Kuala Lumpur Tel: 60-3-7651-7906 Malaysia - Penang Tel: 60-4-227-8870 Philippines - Manila Tel: 63-2-634-9065 Singapore Tel: 65-6334-8870 Taiwan - Hsin Chu Tel: 886-3-577-8366 Taiwan - Kaohsiung Tel: 886-7-213-7830 Taiwan - Taipei Tel: 886-2-2508-8600 Thailand - Bangkok Tel: 66-2-694-1351 Vietnam - Ho Chi Minh Tel: 84-28-5448-2100 | Austria - Wels Tel: 43-7242-2244-39 Fax: 43-7242-2244-393 Denmark - Copenhagen Tel: 45-4485-5910 Fax: 45-4485-2829 Finland - Espoo Tel: 358-9-4520-820 France - Paris Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79 Germany - Garching Tel: 49-8931-9700 Germany - Haan Tel: 49-2129-3766400 Germany - Heilbronn Tel: 49-7131-72400 Germany - Karlsruhe Tel: 49-721-625370 Germany - Munich Tel: 49-89-627-144-0 Fax: 49-89-627-144-44 Germany - Rosenheim Tel: 49-8031-354-560 Israel - Ra'anana Tel: 972-9-744-7705 Italy - Milan Tel: 39-0331-742611 Fax: 39-0331-466781 Italy - Padova Tel: 39-049-7625286 Netherlands - Drunen Tel: 31-416-690399 Fax: 31-416-690340 Norway - Trondheim Tel: 47-72884388 Poland - Warsaw Tel: 48-22-3325737 Romania - Bucharest Tel: 40-21-407-87-50 Spain - Madrid Tel: 34-91-708-08-90 Fax: 34-91-708-08-91 Sweden - Gothenberg Tel: 46-31-704-60-40 Sweden - Stockholm Tel: 46-8-5090-4654 UK - Wokingham Tel: 44-118-921-5800 Fax: 44-118-921-5820 |