

---

## Section 32. Peripheral Trigger Generator (PTG)

---

### HIGHLIGHTS

This section of the manual contains the following major topics:

32.1	Introduction .....	32-2
32.2	Status and Control Registers .....	32-4
32.3	Step Commands and Format .....	32-14
32.4	Module Applications .....	32-17
32.5	Application Examples .....	32-29
32.6	Power-Saving modes .....	32-36
32.7	Register Map .....	32-37
32.8	Related Application Notes .....	32-38
32.9	Revision History .....	32-39



**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 dsPIC33E/PIC24E devices.

Please consult the note at the beginning of the “**Peripheral Trigger Generator (PTG)**” chapter in the current 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 Web site at: <http://www.microchip.com>

## 32.1 INTRODUCTION

The Peripheral Trigger Generator (PTG) module is a user-programmable sequencer, which is capable of generating complex trigger signal sequences to coordinate the operation of other peripherals.

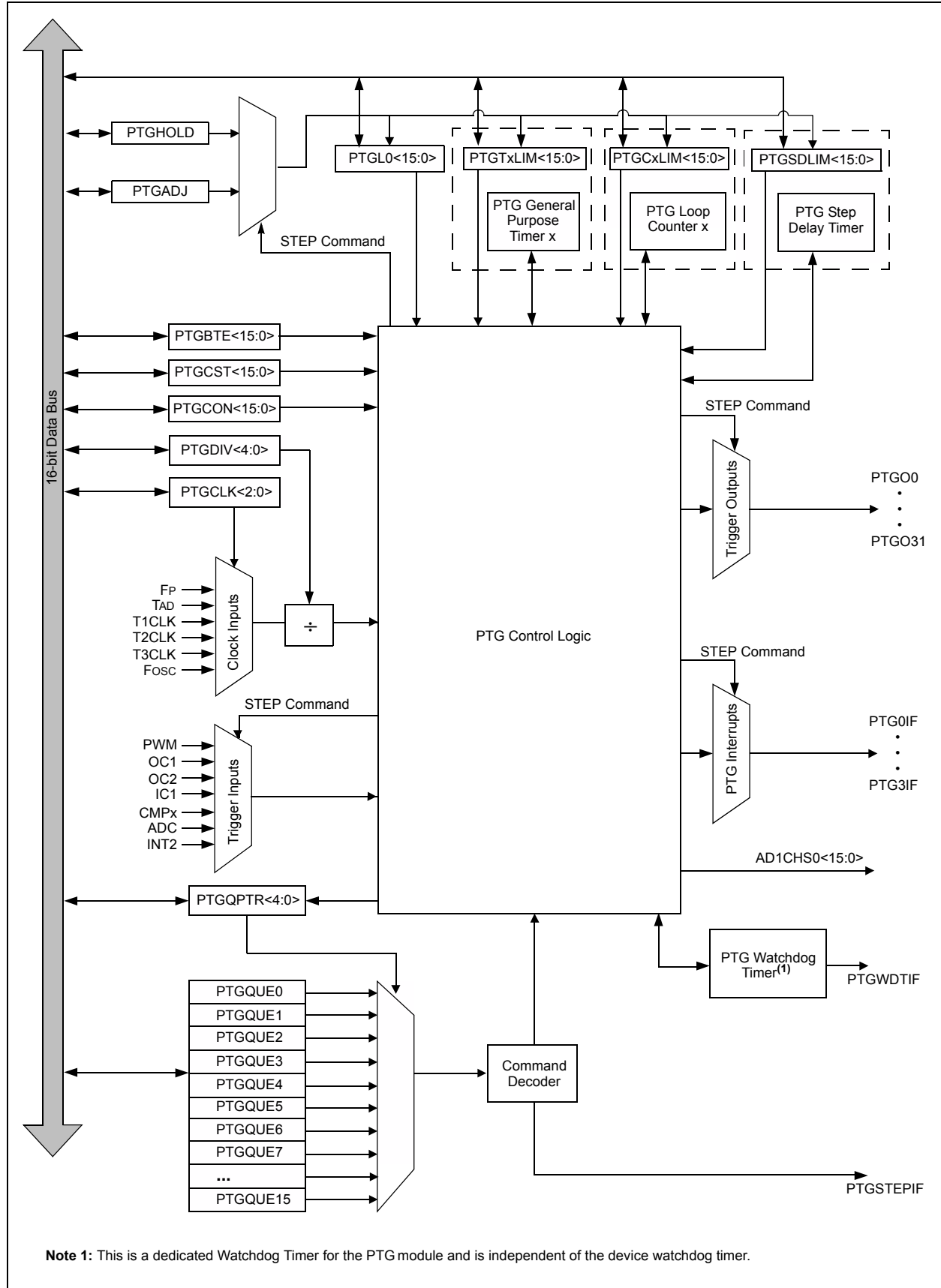
The PTG module is designed to interface with the modules such as an Analog-to-Digital Converter (ADC), Output Compare and PWM modules, Timers, and Interrupt Controllers.

### 32.1.1 FEATURES

- Behavior is Step command-driven
  - Step commands are 8 bits wide
- Commands are stored in a Step queue
  - Queue depth is parameterized (8-32 entries)
  - Programmable Step execution time (Step delay)
- Supports the command sequence loop
  - Can be nested one level deep
  - Conditional or unconditional loop
  - Two 16-bit loop counters
- 16 hardware input triggers
  - Sensitive to either positive or negative edges, or a high or low level
- One software input trigger
- Generates up to 32 unique output trigger signals
- Generates two types of trigger outputs:
  - Individual
  - Broadcast
- Strobed output port for literal data values
  - 5-bit literal write (literal part of a command)
  - 16-bit literal write (literal held in the PTGL0 register)
- Generates up to 16 unique interrupt signals
- Two 16-bit general purpose timers
- Flexible self-contained Watchdog Timer (WDT) to set an upper limit to trigger wait time
- Single Step command capability in Debug mode
- Selectable clock (system, Pulse-width Modulator (PWM), or ADC)
- Programmable clock divider



Figure 32-1: PTG Block Diagram





## 32.2 STATUS AND CONTROL REGISTERS

The following registers are used to control the operation of the PTG module:

- **PTGCST: PTG Control/Status Register**

This register is used to control the following features of the PTG module:

- Enable/disable the PTG module
- Idle mode control
- Trigger output mode control
- Software trigger bit
- Enable/disable Single-Step mode
- Counter/timer visibility control bit
- Start/stop the PTG sequencer
- Input trigger command operating mode control

This register is also used to obtain the status of PTG watchdog timer.

- **PTGCON: PTG Control Register**

This register is used to control the following features of the PTG module:

- PTG clock selection
- PTG clock prescaler
- PTG trigger output pulse width
- PTG watchdog time-out count value

- **PTGBTE: PTG Broadcast Trigger Enable Register<sup>(1,2)</sup>**

This register is used to enable the broadcast trigger for the PTG module.

- **PTGT0LIM: PTG Timer0 Limit Register<sup>(1)</sup>**

This register determines the limit value for Timer0.

- **PTGT1LIM: PTG Timer1 Limit Register<sup>(1)</sup>**

This register determines the limit value for Timer1.

- **PTGSDLIM: PTG Step Delay Limit Register<sup>(1,2)</sup>**

This register determines the limit value for the Step delay.

- **PTGC0LIM: PTG Counter 0 Limit Register<sup>(1)</sup>**

This register determines the limit value for Counter 0.

- **PTGC1LIM: PTG Counter 1 Limit Register<sup>(1)</sup>**

This register determines the limit value for Counter 1.

- **PTGHOLD: PTG Hold Register<sup>(1)</sup>**

This register holds the value that can be copied to the PTGTxLIM, PTGCxLIM, PTDSDLIM or PTGL0 register using the `PTGCOPY` command.

- **PTGADJ: PTG Adjust Register<sup>(1)</sup>**

This register holds the value that can be added to the PTGTxLIM, PTGCxLIM, PTDSDLIM or PTGL0 register using the `PTGADD` command.

- **PTGL0: PTG Literal 0 Register<sup>(1)</sup>**

This register holds the value that can be written to the AD1CHS0 register using the `PTGCTRL` command.

- **PTGQPTR: PTG Step Queue Pointer Register<sup>(1)</sup>**

This register points to the currently active Step command in the Step queue.

- **PTGQEn: PTG Step Queue Pointer Register (n = 0-15)<sup>(1)</sup>**

This register is the queue register for the PTG module.



## Section 32. Peripheral Trigger Generator (PTG)

**Register 32-1: PTGCST: PTG Control/Status Register**

R/W-0	U-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0
PTGEN	—	PTGSIDL	PTGTOGL	—	PTGSWT <sup>(2)</sup>	PTGSSEN	PTGIVIS
bit 15							bit 8

R/W-0	HS-0	U-0	U-0	U-0	U-0	R/W-0
PTGSTRT	PTGWDTO	—	—	—	—	PTGITM<1:0> <sup>(1)</sup>
bit 7						bit 0

<b>Legend:</b>		HS = Set by Hardware	
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 15	<b>PTGEN:</b> Module Enable bit 1 = PTG module is enabled 0 = PTG module is disabled
bit 14	<b>Unimplemented:</b> Read as '0'
bit 13	<b>PTGSIDL:</b> Stop in Idle Mode bit 1 = Discontinue the module operation when the device enters Idle mode 0 = Continue the module operation in Idle mode
bit 12	<b>PTGTOGL:</b> TRIG Output Toggle Mode bit 1 = Toggle state of the PTGOx for each execution of the PTGTRIG command 0 = Each execution of the PTGTRIG command will generate a single PTGOx pulse
bit 11	<b>Unimplemented:</b> Read as '0'
bit 10	<b>PTGSWT:</b> Software Trigger bit <sup>(2)</sup> 1 = Trigger the PTG module 0 = No action (clearing this bit will have no effect)
bit 9	<b>PTGSSEN:</b> Enable Single-Step bit 1 = Enable Single-Step mode 0 = Disable Single-Step mode
bit 8	<b>PTGIVIS:</b> Counter/Timer Visibility Control bit 1 = Reading PTGSDLIM, PTGCxLIM or PTGTxLIM registers return the current values of their corresponding counter/timer registers (PTGSD, PTGCx, and PTGTx) 0 = Reading PTGSDLIM, PTGCxLIM or PTGTxLIM registers return the value of these limit registers
bit 7	<b>PTGSTRT:</b> Start PTG Sequencer bit 1 = Start to sequentially execute the commands (Continuous mode) 0 = Stop executing the commands
bit 6	<b>PTGWDTO:</b> PTG Watchdog Timer Time-out Status bit 1 = PTG Watchdog Timer has timed out 0 = PTG Watchdog Timer has not timed out
bit 5-2	<b>Unimplemented:</b> Read as '0'
bit 1-0	<b>PTGITM&lt;1:0&gt;:</b> PTG Input Trigger Command Operating Mode bits <sup>(1)</sup> 11 = Single level detect with Step delay not executed on exit of command (regardless of the PTGCTRL command) (Mode 3) 10 = Single level detect with Step delay executed on exit of command (Mode 2) 01 = Continuous edge detect with Step delay not executed on exit of command (regardless of the PTGCTRL command) (Mode 1) 00 = Continuous edge detect with Step delay executed on exit of command (Mode 0)

**Note 1:** These bits apply to the PTGWHI and PTGWLO commands only.

**Note 2:** This bit is only used with the PTGCTRL Step command software trigger option.

32

Peripheral Trigger  
Generator (PTG)



# dsPIC33E/PIC24E Family Reference Manual

**Register 32-2: PTGCON: PTG Control Register**

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PTGCLK<2:0>			PTGDIV<4:0>				
bit 15			bit 8				

R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0
PTGPWD<3:0>				—	PTGWDT<2:0>		
bit 7					bit 0		

**Legend:**

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 15-13 **PTGCLK<2:0>**: Select PTG Module Clock Source bits

111 = Reserved

110 = Reserved

101 = PTG module clock source will be T3CLK

100 = PTG module clock source will be T2CLK

011 = PTG module clock source will be T1CLK

010 = PTG module clock source will be ADC FRC clock

001 = PTG module clock source will be Fosc

000 = PTG module clock source will be Fp

bit 12-8 **PTGDIV<4:0>**: PTG Module Clock Prescaler (Divider) bits

11111 = Divide by 32

11110 = Divide by 31

•

•

•

00001 = Divide by 2

00000 = Divide by 1

bit 7-4 **PTGPWD<3:0>**: PTG Trigger Output Pulse Width bits

1111 = All trigger outputs are 16 PTG clock cycles wide

1110 = All trigger outputs are 15 PTG clock cycles wide

•

•

•

0001 = All trigger outputs are 2 PTG clock cycles wide

0000 = All trigger outputs are 1 PTG clock cycle wide

bit 3 **Unimplemented**: Read as '0'

bit 2-0 **PTGWDT<2:0>**: Select PTG Watchdog Time-out Count Value bits

111 = Watchdog will time out after 512 PTG clocks

110 = Watchdog will time out after 256 PTG clocks

101 = Watchdog will time out after 128 PTG clocks

100 = Watchdog will time out after 64 PTG clocks

011 = Watchdog will time out after 32 PTG clocks

010 = Watchdog will time out after 16 PTG clocks

001 = Watchdog will time out after 8 PTG clocks

000 = Watchdog is disabled



## Section 32. Peripheral Trigger Generator (PTG)

**Register 32-3: PTGBTE: PTG Broadcast Trigger Enable Register<sup>(1,2)</sup>**

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
ADCTS4	ADCTS3	ADCTS2	ADCTS1	IC4TSS	IC3TSS	IC2TSS	IC1TSS
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
OC4CS	OC3CS	OC2CS	OC1CS	OC4TSS	OC3TSS	OC2TSS	OC1TSS
bit 7							bit 0

**Legend:**

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

- bit 15 **ADCTS4:** Sample Trigger PTGO15 for ADC bit  
1 = Generate trigger when the broadcast command is executed  
0 = Do not generate trigger when the broadcast command is executed
- bit 14 **ADCTS3:** Sample Trigger PTGO14 for ADC bit  
1 = Generate trigger when the broadcast command is executed  
0 = Do not generate trigger when the broadcast command is executed
- bit 13 **ADCTS2:** Sample Trigger PTGO13 for ADC bit  
1 = Generate trigger when the broadcast command is executed  
0 = Do not generate trigger when the broadcast command is executed
- bit 12 **ADCTS1:** Sample Trigger PTGO12 for ADC bit  
1 = Generate trigger when the broadcast command is executed  
0 = Do not generate trigger when the broadcast command is executed
- bit 11 **IC4TSS:** Trigger/Synchronization Source for IC4 bit  
1 = Generate trigger/synchronization when the broadcast command is executed  
0 = Do not generate trigger/synchronization when the broadcast command is executed
- bit 10 **IC3TSS:** Trigger/Synchronization Source for IC3 bit  
1 = Generate trigger/synchronization when the broadcast command is executed  
0 = Do not generate trigger/synchronization when the broadcast command is executed
- bit 9 **IC2TSS:** Trigger/Synchronization Source for IC2 bit  
1 = Generate trigger/synchronization when the broadcast command is executed  
0 = Do not generate trigger/synchronization when the broadcast command is executed
- bit 8 **IC1TSS:** Trigger/Synchronization Source for IC1 bit  
1 = Generate trigger/synchronization when the broadcast command is executed  
0 = Do not generate trigger/synchronization when the broadcast command is executed
- bit 7 **OC4CS:** Clock Source for OC4 bit  
1 = Generate clock pulse when the broadcast command is executed  
0 = Do not generate clock pulse when the broadcast command is executed
- bit 6 **OC3CS:** Clock Source for OC3 bit  
1 = Generate clock pulse when the broadcast command is executed  
0 = Do not generate clock pulse when the broadcast command is executed
- bit 5 **OC2CS:** Clock Source for OC2 bit  
1 = Generate clock pulse when the broadcast command is executed  
0 = Do not generate clock pulse when the broadcast command is executed

**Note 1:** This register is read-only when the PTG module is executing the Step commands (PTGEN = 1 and PTGSTRT = 1).

**2:** This register is only used with the PTGCTRL OPTION = 1111 Step command.

**32**

**Peripheral Trigger  
Generator (PTG)**



## Register 32-3: PTGBTE: PTG Broadcast Trigger Enable Register<sup>(1,2)</sup> (Continued)

bit 4	<b>OC1CS:</b> Clock Source for OC1 bit 1 = Generate clock pulse when the broadcast command is executed 0 = Do not generate clock pulse when the broadcast command is executed
bit 3	<b>OC4TSS:</b> Trigger/Synchronization Source for OC4 bit 1 = Generate trigger/synchronization when the broadcast command is executed 0 = Do not generate trigger/synchronization when the broadcast command is executed
bit 2	<b>OC3TSS:</b> Trigger/Synchronization Source for OC3 bit 1 = Generate trigger/synchronization when the broadcast command is executed 0 = Do not generate trigger/synchronization when the broadcast command is executed
bit 1	<b>OC2TSS:</b> Trigger/Synchronization Source for OC2 bit 1 = Generate trigger/synchronization when the broadcast command is executed 0 = Do not generate trigger/synchronization when the broadcast command is executed
bit 0	<b>OC1TSS:</b> Trigger/Synchronization Source for OC1 bit 1 = Generate trigger/synchronization when the broadcast command is executed 0 = Do not generate trigger/synchronization when the broadcast command is executed

**Note 1:** This register is read-only when the PTG module is executing the Step commands (PTGEN = 1 and PTGSTRT = 1).

**2:** This register is only used with the PTGCTRL OPTION = 1111 Step command.



## Section 32. Peripheral Trigger Generator (PTG)

**Register 32-4: PTGT0LIM: PTG Timer0 Limit Register<sup>(1)</sup>**

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PTGT0LIM<15:8>							
bit 15				bit 8			

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PTGT0LIM<7:0>							
bit 7				bit 0			

**Legend:**

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 15-0 **PTGT0LIM<15:0>**: PTG Timer0 Limit Register bits  
General purpose Timer0 limit register.

**Note 1:** This register is read-only when the PTG module is executing the Step commands (PTGEN = 1 and PTGSTRT = 1).

**Register 32-5: PTGT1LIM: PTG Timer1 Limit Register<sup>(1)</sup>**

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PTGT1LIM<15:8>							
bit 15				bit 8			

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PTGT1LIM<7:0>							
bit 7				bit 0			

**Legend:**

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 15-0 **PTGT1LIM<15:0>**: PTG Timer1 Limit Register bits  
General purpose Timer1 limit register.

**Note 1:** This register is read-only when the PTG module is executing the Step commands (PTGEN = 1 and PTGSTRT = 1).



# dsPIC33E/PIC24E Family Reference Manual

**Register 32-6: PTGSDLIM: PTG Step Delay Limit Register<sup>(1,2)</sup>**

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PTGSDLIM<15:8>							
bit 15				bit 8			

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PTGSDLIM<7:0>							
bit 7				bit 0			

**Legend:**

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 15-0 **PTGSDLIM<15:0>**: PTG Step Delay Limit Register bits

This register holds a PTG Step delay value representing the number of additional PTG clocks between the start of a Step command and the completion of a Step command.

**Note 1:** A base Step delay of one PTG clock is added to any value written to the PTGSDLIM register (Step Delay = (PTGSDLIM) + 1).

**2:** This register is read-only when the PTG module is executing the Step commands (PTGEN = 1 and PTGSTRT = 1).

**Register 32-7: PTGC0LIM: PTG Counter 0 Limit Register<sup>(1)</sup>**

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PTGC0LIM<15:8>							
bit 15				bit 8			

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PTGC0LIM<7:0>							
bit 7				bit 0			

**Legend:**

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 15-0 **PTGC0LIM<15:0>**: PTG Counter 0 Limit Register bits

This register is used to specify the loop count for the PTGJMPC0 Step command, or as a limit register for the general purpose counter 0.

**Note 1:** This register is read-only when the PTG module is executing the Step commands (PTGEN = 1 and PTGSTRT = 1).



## Section 32. Peripheral Trigger Generator (PTG)

**Register 32-8: PTGC1LIM: PTG Counter 1 Limit Register<sup>(1)</sup>**

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PTGC1LIM<15:8>							
bit 15				bit 8			

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PTGC1LIM<7:0>							
bit 7				bit 0			

**Legend:**

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 15-0 **PTGC1LIM<15:0>**: PTG Counter 1 Limit Register bits

This register is used to specify the loop count for the PTGJMPC1 Step command, or as a limit register for the general purpose counter 1.

**Note 1:** This register is read-only when the PTG module is executing the Step commands (PTGEN = 1 and PTGSTRT = 1).

**Register 32-9: PTGHOLD: PTG Hold Register<sup>(1)</sup>**

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PTGHOLD<15:8>							
bit 15				bit 8			

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PTGHOLD<7:0>							
bit 7				bit 0			

**Legend:**

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 15-0 **PTGHOLD<15:0>**: PTG General Purpose Hold Register bits

This register holds the user supplied data to be copied to the PTGTxLIM, PTGCxLIM, PTGSDLIM, or PTGL0 register using the PTGCOPY command.

**Note 1:** This register is read-only when the PTG module is executing the Step commands (PTGEN = 1 and PTGSTRT = 1).



# dsPIC33E/PIC24E Family Reference Manual

## Register 32-10: PTGADJ: PTG Adjust Register<sup>(1)</sup>

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PTGADJ<15:8>							
bit 15				bit 8			

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PTGADJ<7:0>							
bit 7				bit 0			

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 15-0

**PTGADJ<15:0>:** PTG Adjust Register bits

This register holds the user-supplied data to be added to the PTGTxLIM, PTGCxLIM, PTGSDLIM, or PTGL0 register using the PTGADD command.

**Note 1:** This register is read-only when the PTG module is executing the Step commands (PTGEN = 1 and PTGSTRT = 1).

## Register 32-11: PTGL0: PTG Literal 0 Register<sup>(1)</sup>

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PTGL0<15:8>							
bit 15				bit 8			

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PTGL0<7:0>							
bit 7				bit 0			

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 15-0

**PTGL0<15:0>:** PTG Literal 0 Register bits

This register holds a 16-bit value to be written to the AD1CHS0 register using the PTGCTRL command.

**Note 1:** This register is read-only when the PTG module is executing the Step commands (PTGEN = 1 and PTGSTRT = 1).



## Section 32. Peripheral Trigger Generator (PTG)

**Register 32-12: PTGQPTR: PTG Step Queue Pointer Register<sup>(1)</sup>**

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 15				bit 8			

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	PTGQPTR<4:0>				
bit 7				bit 0			

**Legend:**

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 15-5 **Unimplemented:** Read as '0'

bit 4-0 **PTGQPTR<4:0>:** PTG Step Queue Pointer Register bits

This register points to the currently active Step command in the Step queue.

**Note 1:** This register is read-only when the PTG module is executing the Step commands (PTGEN = 1 and PTGSTRT = 1).

**Register 32-13: PTGQUEn: PTG Step Queue Pointer Register (n = 0-15)<sup>(1)</sup>**

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
STEPx<7:0> <sup>(2)</sup>							
bit 15				bit 8			

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
STEPy<7:0> <sup>(2)</sup>							
bit 7				bit 0			

**Legend:**

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 15-8 **STEPx<7:0>:** PTG Step Queue Pointer Register bits<sup>(2)</sup>

A queue location for storage of the STEP(2x + 1) command byte, where 'x' are the odd-numbered step queue pointers.

bit 7-0 **STEPy<7:0>:** PTG Step Queue Pointer Register bits<sup>(2)</sup>

A queue location for storage of the STEP(2x) command byte, where 'y' are the even-numbered step queue pointers.

**Note 1:** This register is read-only when the PTG module is executing the Step commands (PTGEN = 1 and PTGSTRT = 1).

**2:** Refer to [Table 32-1](#) for the Step command encoding.



# dsPIC33E/PIC24E Family Reference Manual

## 32.3 STEP COMMANDS AND FORMAT

**Table 32-1: PTG Step Command Format and Description**

Step Command Byte			
STEPx<7:0>			
CMD<3:0>		OPTION<3:0>	
bit 7	bit 4	bit 3	bit 0

bit 7-4	Step Command	CMD<3:0>	Command Description
	PTGCTRL	0000	Execute the control command as described by the OPTION<3:0> bits.
	PTGADD	0001	Add contents of the PTGADJ register to the target register as described by the OPTION<3:0> bits.
	PTGCOPY		Copy contents of the PTGHOLD register to the target register as described by the OPTION<3:0> bits.
	PTGSTRB	001x	Copy the values contained in the bits, CMD<0>:OPTION<3:0> to the CH0SA<4:0> bits (AD1CHS0<4:0>).
	PTGWHI	0100	Wait for a low-to-high edge input from a selected PTG trigger input as described by the OPTION<3:0> bits.
	PTGWLO	0101	Wait for a high-to-low edge input from a selected PTG trigger input as described by the OPTION<3:0> bits.
	—	0110	Reserved; do not use.
	PTGIRQ	0111	Generate individual interrupt request as described by the OPTION<3:0> bits.
	PTGTRIG	100x	Generate individual trigger output as described by the bits, CMD<0>:OPTION<3:0>.
	PTGJMP	101x	Copy the values contained in the bits, CMD<0>:OPTION<3:0> to the PTGQPTR register, and jump to that Step queue.
	PTGJMPC0	110x	PTGC0 = PTGC0LIM: Increment the PTGQPTR register. PTGC0 ≠ PTGC0LIM: Increment Counter 0 (PTGC0) and copy the values contained in the bits, CMD<0>:OPTION<3:0> to the PTGQPTR register, and jump to that Step queue.
	PTGJMPC1	111x	PTGC1 = PTGC1LIM: Increment the PTGQPTR register. PTGC1 ≠ PTGC1LIM: Increment Counter 1 (PTGC1) and copy the values contained in the bits, CMD<0>:OPTION<3:0> to the PTGQPTR register, and jump to that Step queue.

**Note 1:** All reserved commands or options will execute, but they do not have any affect (i.e., execute as a NOP instruction).

**2:** This feature is only available on dsPIC33EPXXXMC20X/50X and PIC24EPXXXMC20X devices.



## Section 32. Peripheral Trigger Generator (PTG)

**TABLE 32-1: PTG STEP COMMAND FORMAT (CONTINUED)**

bit 3-0	Step Command	OPTION<3:0>	Command Description
	PTGCTRL <sup>(1)</sup>	0000	Reserved; do not use.
		0001	Reserved; do not use.
		0010	Disable Step delay timer (PTGSD).
		0011	Reserved; do not use.
		0100	Reserved; do not use.
		0101	Reserved; do not use.
		0110	Enable Step delay timer (PTGSD).
		0111	Reserved; do not use.
		1000	Start and wait for the PTG Timer0 to match the PTGT0LIM register.
		1001	Start and wait for the PTG Timer1 to match the PTGT1LIM register.
		1010	Wait for the software trigger (level, PTGSWT = 1).
		1011	Wait for the software trigger (positive edge, PTGSWT = 0 to 1).
		1100	Copy the PTGC0LIM register contents to the AD1CHS0 register.
		1101	Copy the PTGC1LIM register contents to the AD1CHS0 register.
		1110	Copy the PTGL0 register contents to the AD1CHS0 register.
		1111	Generate the triggers indicated in the PTGBTE register.
	PTGADD <sup>(1)</sup>	0000	Add the PTGADJ register contents to the PTGC0LIM register.
		0001	Add the PTGADJ register contents to the PTGC1LIM register.
		0010	Add the PTGADJ register contents to the PTGT0LIM register.
		0011	Add the PTGADJ register contents to the PTGT1LIM register.
		0100	Add the PTGADJ register contents to the PTGSDLIM register.
		0101	Add the PTGADJ register contents to the PTGL0 register.
		0110	Reserved; do not use.
		0111	Reserved; do not use.
	PTGCOPY <sup>(1)</sup>	1000	Copy the PTGHOLD register contents to the PTGC0LIM register.
		1001	Copy the PTGHOLD register contents to the PTGC1LIM register.
		1010	Copy the PTGHOLD register contents to the PTGT0LIM register.
		1011	Copy the PTGHOLD register contents to the PTGT1LIM register.
		1100	Copy the PTGHOLD register contents to the PTGSDLIM register.
		1101	Copy the PTGHOLD register contents to the PTGL0 register.
		1110	Reserved; do not use.
		1111	Reserved; do not use.

**Note 1:** All reserved commands or options will execute, but they do not have any affect (i.e., execute as a NOP instruction).

**2:** This feature is only available on dsPIC33EPXXXMC20X/50X and PIC24EPXXXMC20X devices.



# dsPIC33E/PIC24E Family Reference Manual

**TABLE 32-1: PTG STEP COMMAND FORMAT (CONTINUED)**

bit 3-0	Step Command	OPTION<3:0>	Option Description
	PTGWHI <sup>(1)</sup> or PTGWLO <sup>(1)</sup>	0000	PWM Special Event Trigger <sup>(2)</sup> .
		0001	PWM Master Timebase Synchronization Output <sup>(2)</sup> .
		0010	PWM1 Interrupt.
		0011	PWM2 Interrupt.
		0100	PWM3 Interrupt.
		0101	Reserved; do not use.
		0110	Reserved; do not use.
		0111	OC1 Trigger Event.
		1000	OC2 Trigger Event.
		1001	IC1 Trigger Event.
		1010	CMP1 Trigger Event.
		1011	CMP2 Trigger Event.
		1100	CMP3 Trigger Event.
		1101	CMP4 Trigger Event.
		1110	ADC Conversion Done Interrupt.
		1111	INT2 External Interrupt.
	PTGIRO <sup>(1)</sup>	0000	Generate PTG Interrupt 0.
		0001	Generate PTG Interrupt 1.
		0010	Generate PTG Interrupt 2.
		0011	Generate PTG Interrupt 3.
		0100	Reserved; do not use.
		.	.
		.	.
		.	.
	PTGTRIG	1111	Reserved; do not use.
		00000	PTGO0.
		00001	PTGO1.
		.	.
		.	.
		.	.
		11110	PTGO30.
		11111	PTGO31.

**Note 1:** All reserved commands or options will execute, but they do not have any affect (i.e., execute as a NOP instruction).

**2:** This feature is only available on dsPIC33EPXXXMC20X/50X and PIC24EPXXXMC20X devices.



### 32.4 MODULE APPLICATIONS

#### 32.4.1 Module Description

The PTG module is a user-programmable sequencer for generating complex peripheral trigger sequences. The PTG module provides the ability to schedule complex peripheral operations, which would be difficult or impossible to achieve via a software solution.

The user writes 8-bit commands, called Step commands, to the PTG queue registers (PTGQUE0-PTGQUE15). Each 8-bit Step command is made up of a 4-bit command code and a 4-bit option field. [Table 32-1](#) shows the format and encoding of a Step command. Based on the commands, the PTG can interact with other modules such as the PWM, ADC, Input Capture, and Output Compare.

#### 32.4.2 Basic Operation

The user loads the Step commands (8-bit values) into the PTG queue registers. The commands define a sequence of events for generating the trigger output signals to peripherals. The Step commands can also be used to generate the interrupt requests to the processor.

- The PGT module is disabled when the PTGEN bit (PTGCST<15>) = 0
- The PTG module is enabled and clocked when the PTGEN bit (PTGCST<15>) = 1. While the PTGSTRT bit (PTGCST<7>) = 0, the PTG module is in the Halt state.

**Note 1:** The control registers cannot be modified when the PTG module is in the Halt state.  
**2:** The PTG module must be enabled (PTGEN = 1) prior to attempting to set the PTGSTRT bit.  
**3:** The user should not attempt to set the PTGEN and PTGSTRT bits within the same data write cycle.

Subsequently setting the PTGSTRT = 1, will enable the module for Continuous mode execution of the Step command queue. The PTG sequencer will start to read the Step queue at the address held in the queue pointer (PTGQPTR). Each command byte is read, decoded, and executed sequentially. The minimum duration of any Step command is one PTG clock as explained in [32.4.4 “PTG Clock Selection”](#).

Step commands will execute sequentially until any of the following occurs:

- A PTGJMP, PTGJMPC0 or PTGJMPC1 (flow change) Step command is executed
- The user clears the PTGSTRT bit stopping the PTG sequencer. No further Step commands are read/decoded and execution halts. For more information, refer to [32.4.3 “Stopping the Sequencer”](#).
- The internal Watchdog Timer overflows, clearing the PTGSTRT bit, and stopping the PTG sequencer. No further Step commands are read/decoded and execution halts. For more information, refer to [32.4.3 “Stopping the Sequencer”](#).
- The PTG module is disabled (PTGEN = 0). For more information, refer to [32.4.3 “Stopping the Sequencer”](#).

The Step commands can also be made to wait on a condition, such as an input trigger edge, a software trigger, or a timer match, before continuing execution.



## 32.4.3 Stopping the Sequencer

When the PTG module is disabled (PTGEN = 0), the PTG clocks are disabled (except the trigger pulse counter), the sequencer stops execution, and the module enters its lowest power state. The PTGSTRT, PTGSWT, PTGWDTO and PTGQPTR bits are all reset. All other bits and registers are not modified. All of the control registers can be read or written when the PTGEN = 0.

When the PTGEN bit is cleared, a command that is underway is immediately aborted if the command is waiting for any of the following actions:

- An input from another source
- A timer match
- The Step delay to expire (For more information, refer to [32.4.9.5 “Step Command Delay”](#))

All other commands are allowed to complete before the PTG module is disabled.

When the PTG module is halted, all of the control registers remain in their present state. The PTG module can be halted by the user by clearing the PTGSTRT bit, or in the event of a Watchdog time-out, which also clears the PTGSTRT bit. Refer to [32.4.10 “PTG Input Trigger Watchdog Timer”](#) for more information.

## 32.4.4 PTG Clock Selection

The PTG module has multiple clock options and has a selectable prescaler, which divides the PTG clock input from 1 to 32.

### 32.4.4.1 CLOCK SOURCE SELECTION

The PTGCLK<2:0> bits (PTGCON<15:13>) specify the clock source for the PTG clock generation logic. The PTG module can be operated with a system clock, an ADC clock, a PWM clock, or an external clock.

### 32.4.4.2 CLOCK PRESCALER SELECTION

The PTGDIV<4:0> bits (PTGCON<12:8>) specify the Prescaler value for the PTG clock generation logic. These bits can be written only when the PTG module is disabled (PTGEN = 0).

**Note:** Any attempt to write to the PTGDIV<4:0> bits or the PTGCLK<2:0> bits while PTGEN = 1 will have no effect.

### 32.4.4.3 MODULE ENABLE DELAY

Once the PTG module is enabled (PTGEN = 1), there is a delay before the PTG starts to execute commands. This delay is expressed in [Equation 32-2](#).

#### Equation 32-2:

$$TDLYEN = 4 \bullet \text{PTG clock period (Maximum)}$$

The user must ensure that no control bits are modified during the delay. Also, no external triggers are asserted prior to the PTG state machine commencing execution, otherwise, the triggers will be missed.



### 32.4.5 Control Register Access

When the PTG module is enabled ( $PTGEN = 1$ ), writes are inhibited to all control registers with the exception of the PTGCST register, which may be read and written to as normal.

When the PTG module is enabled ( $PTGEN = 1$ ), reads can be performed from any control register at any time; however, the data read (control register or associated timer/counter value) will depend upon the state of the PTGIVIS bit ( $PTGCST<8>$ ).

**Note:** Only some registers are affected by the state of the PTGIVIS bit. Refer to PTGIVIS bit description.

When the PTG module is disabled ( $PTGEN = 0$ ), all control registers can be read and written to as normal. The PTGIVIS bit ( $PTGCST<8>$ ) has no effect when  $PTGEN = 0$ , because all timers/counters will be cleared to zero; however, the values in the limit register will stay same.

### 32.4.6 Step Queue Pointer

The PTG Step Queue Pointer register (PTGQPTR) holds the pointer to the PTG queue. The pointer is loaded from the PTGQPTR register when  $PTGEN = 1$  and the PTGSTRT bit is set, and addresses the currently active Step command in the Step queue. The PTGQPTR register is cleared when the module is disabled ( $PTGEN = 0$ ), or is in the Reset state. This is a one-time event that occurs on transition of the PTGEN bit from '1' to '0'.

**Note 1:** The PTGQPTR register is not cleared when the PTG module is halted ( $PTGSTRT = 0$ ).

**2:** The PTGQPTR register is cleared when the PTGEN bit is cleared, but any value can be written to the register irrespective of the state of the PTGEN bit.

The user can read the PTGQPTR register at any time. In the Disabled state ( $PTGEN = 0$ ), a read returns the contents of the PTGQPTR register. In the Idle or Active state ( $PTGEN = 1$ ), a read returns the contents of the pointer.

The PTGQPTR register is typically incremented during the first cycle of each command. The exceptions to this rule are:

- If the  $PTGJMP$  command is executed: the Step queue pointer is loaded with the target queue address
- If the  $PTGJMPCx$  command is executed, and  $PTGCx$  is not zero: the Step queue pointer is loaded with the target queue address

### 32.4.7 Command Looping Control

Two 16-bit loop counters are provided ( $PTGC0$  and  $PTGC1$ ) that can be used by the  $PTGJMPCx$  command as a block loop counter or delay generator. All internal counters are cleared when the module is in the Reset state, or when the PTG module is disabled ( $PTGEN = 0$ ).

Each loop counter consists of an incrementing counter ( $PTGCx$ ) and a limit register ( $PTGCxLIM$ ). The limit register value can be changed by writing directly to the register (when the module is disabled), or by the PTG sequencer (when the module is enabled). The data read from the limit register depends upon the state of the internal visibility bit ( $PTGIVIS$ ).



## 32.4.7.1 USING THE LOOP COUNTER AS A BLOCK LOOP COUNTER

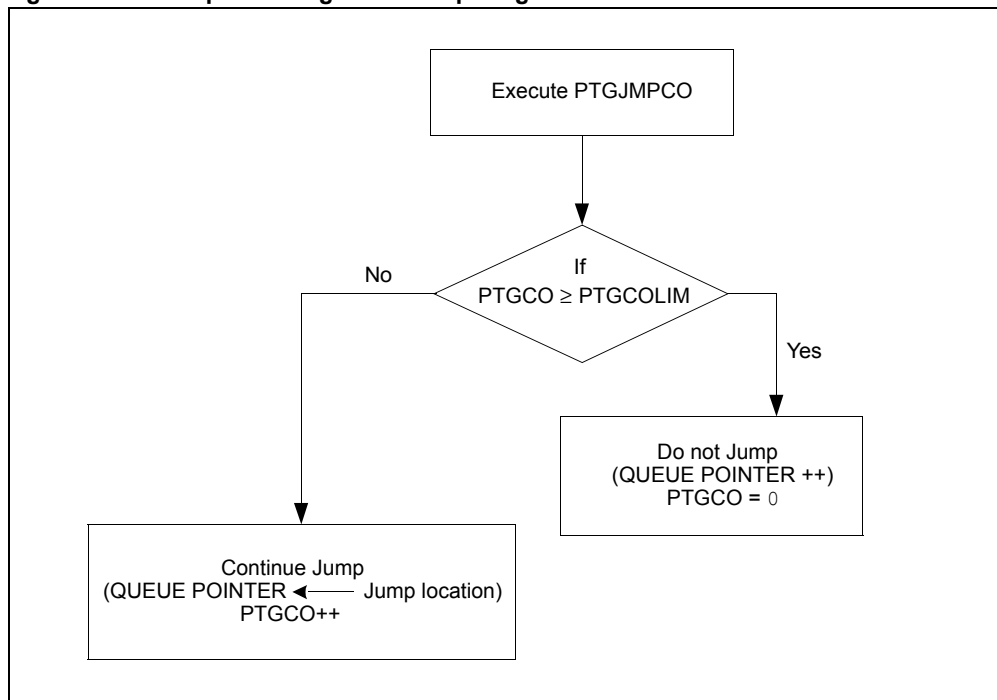
The `PTGJMPCx` (Jump Conditional) command uses one of the loop counters to keep track of the number of times the `PTGJMPCx` command is executed, and can therefore be used to create code block loops. This is useful in applications where a sequence of peripheral events needs to be repeated several times. The `PTGJMPCx` command allows the user to create code loops and use fewer step commands.

Each time the `PTGJMPCx` command is executed, the corresponding internal loop counter is compared to its limit value. If the loop counter has not reached the limit value, the jump location is loaded into the `PTGQPTR` register, and the loop counter is incremented by 1. The next command will be fetched from the new queue location. If the counter has reached the limit value, the sequencer proceeds to the next command (i.e., increment the queue pointer). While preparing for the next `PTGJMPCx` command loop execution, the corresponding loop counter is cleared. See [Figure 32-2](#).

**Note:** The loop counter value can be modified (via the `PTGADD` or `PTGCOPY` command) prior to execution of the first iteration of the command loop.

The provision for two separate loop counters and associated `PTGJMPCx` commands, allows for the nested loops to be supported (one level deep). There are no restrictions with regard to which `PTGJMPCx` command resides in the inner or outer loops.

**Figure 32-2: Implementing Block Loop Diagram**



## 32.4.8 Step Commands

As shown previously, [Table 32-1](#) provides the format and encoding of the Step commands. Each 8-bit Step command consists of a 4-bit command field (`CMD<3:0>`) and a 4-bit option field (`OPTION<3:0>`).



### 32.4.9 Sequencer Operation

All commands are executed in a single cycle, except for the flow change commands, and the commands that are waiting for an external input.

#### 32.4.9.1 STEP COMMAND DURATION

By default, each Step command executes in one PTG clock period. There are several methods to slow the execution of the Step commands:

- Wait for a trigger input
- Wait for a GP timer (PTGTxLIM)
- Insert a delay loop using the PTGJMPCx and PTGCx commands
- Enable and insert a Step delay (PTGSDLIM) after execution of each command

#### 32.4.9.2 WAIT FOR TRIGGER INPUT

The PTG module can support up to 16 independent trigger inputs. The user can specify a Step command that waits for a positive or negative edge, or a high or low level, of the selected input signal to occur. The operating mode is selected by the PTGITM<1:0> bits in the PTGCST register.

The PTGWHI command looks for a positive edge or high state to occur on the selected trigger input. The PTGWLO command looks for a negative edge or low state to occur on the selected trigger input. The PTG repeats the trigger input command (i.e., effectively wait) until the selected signal becomes valid before continuing the Step command execution.

The minimum execution time to wait for a trigger is one PTG clock. There is no limit for the PTG wait for a trigger input other than that enforced by the Watchdog Timer. Refer to [32.4.10 “PTG Input Trigger Watchdog Timer”](#) for more information.

The PTG module supports four input trigger command operating modes (Mode 0 - Mode 3), which are selected by the PTGITM<1:0> bits in the PTGCST register.

**Note:** If the Step delay is disabled, Mode 0 and Mode 1 are equivalent in operation, and Mode 2 and Mode 3 are equivalent in operation.

##### 32.4.9.2.1 Mode 0: PTGITM<1:0> = 0x00 (Continuous Edge Detect with Step Delay at Exit)

In this mode, the selected trigger input is continuously tested starting immediately after the PTGWHI or PTGWLO command is executed. When the trigger edge is detected, the command execution completes.

If the Step delay counter is enabled, the Step delay will be inserted (once) after the valid edge is detected, and after the command execution.

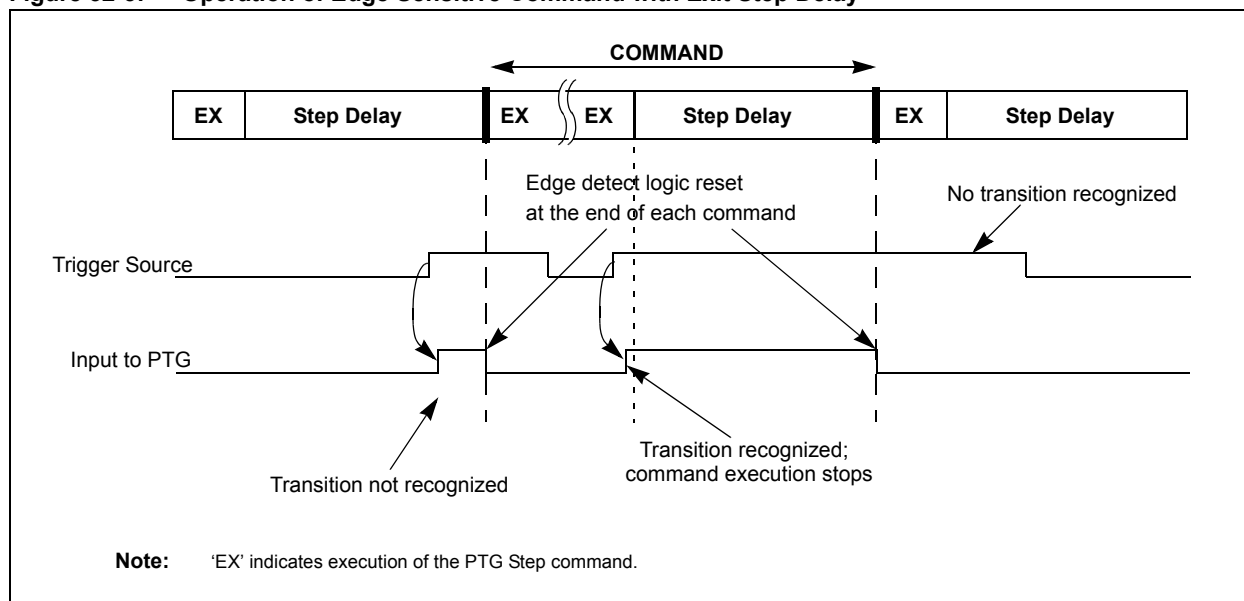
If the Step delay counter is not enabled, the command will complete after the valid edge is detected, and execution of the subsequent command will commence immediately.

**Note:** The edge detect logic is reset after the command execution is complete (i.e., prior to any Step delay associated with the command). For the edge to be detected, the edge should occur during the PTGWHI or PTGWLO command execution, or during the Step delay of the prior command.



Figure 32-3 shows an example timing diagram of Mode 0 operation.

**Figure 32-3: Operation of Edge Sensitive Command with Exit Step Delay**



## 32.4.9.2.2 Mode 1: PTGITM<1:0> = 0x01

(Continuous Edge Detect without Step Delay at Exit)

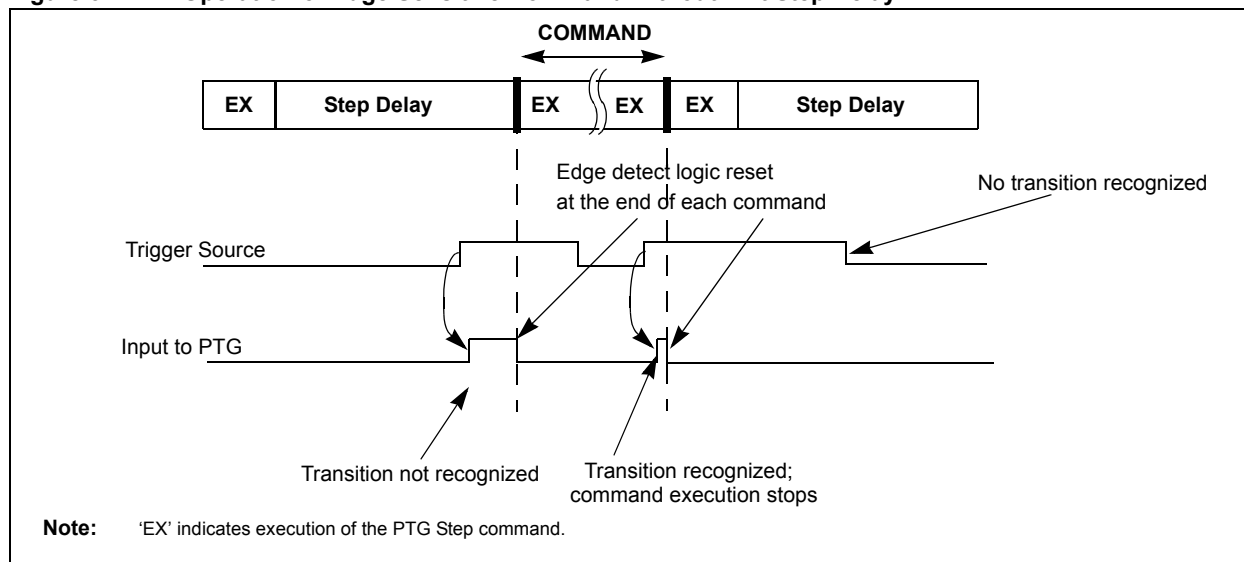
In this mode, the selected trigger input is continuously tested starting immediately after the PTGWHI or PTGWLO command is executed. When the trigger edge is detected, the command execution completes.

Regardless of whether the Step delay counter is enabled or disabled, the Step delay will not be inserted after the command execution has completed.

**Note:** The edge detect logic is reset after the command execution completes. To be detected, the edge may therefore occur during the PTGWHI or PTGWLO command execution, or during the Step delay of the prior command.

Figure 32-4 shows an example timing diagram of Mode 1 operation.

**Figure 32-4: Operation of Edge Sensitive Command without Exit Step Delay**





## 32.4.9.2.3 Mode 2: PTGITM<1:0> = 0x10 (Sampled Level Detect with Step Delay at Exit)

In this mode, the selected trigger input is sample tested for a valid level immediately after the PTGWHI or PTGWLO command is executed, the trigger input is tested (once per PTG clock).

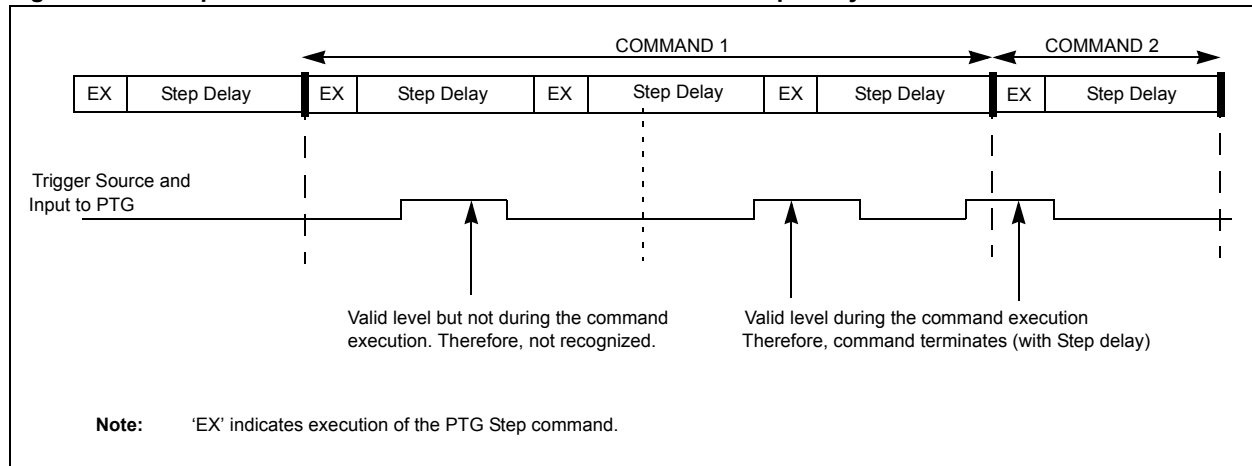
If the trigger does not occur, and the Step delay is enabled, the command waits for the Step delay to expire before testing the trigger input again. When the trigger occurs, the command execution completes, and the Step delay is reinserted.

If the trigger does not occur, and the Step delay is disabled, the command immediately tests the trigger input again during the next PTG clock cycle. When the trigger occurs, the command execution completes, and execution of the subsequent command will commence immediately.

- Note 1:** As this operating mode is level sensitive, if the input trigger level is true at the start of execution of the PTGWHI or PTGWLO command, the input test will be instantly satisfied.
- 2:** The input is not latched, therefore, it must be valid when the command executes in order to be recognized.

Figure 32-5 shows an example timing diagram of Mode 2 operation.

**Figure 32-5: Operation of Level Sensitive Command with Exit Step Delay**





## 32.4.9.2.4 Mode 3: PTGITM<1:0> = 0x11

(Sampled Level Detect without Step Delay at Exit)

In this mode, the selected trigger input is sample tested for a valid level immediately after the PTGWHI or PTGWLO command is executed, the trigger input is tested (once per PTG clock).

If the trigger does not occur, and the Step delay is enabled, the command waits for the Step delay to expire before testing the trigger input again. When the trigger is found to be true, the command execution completes, and execution of the subsequent command will commence immediately. The Step delay is not inserted.

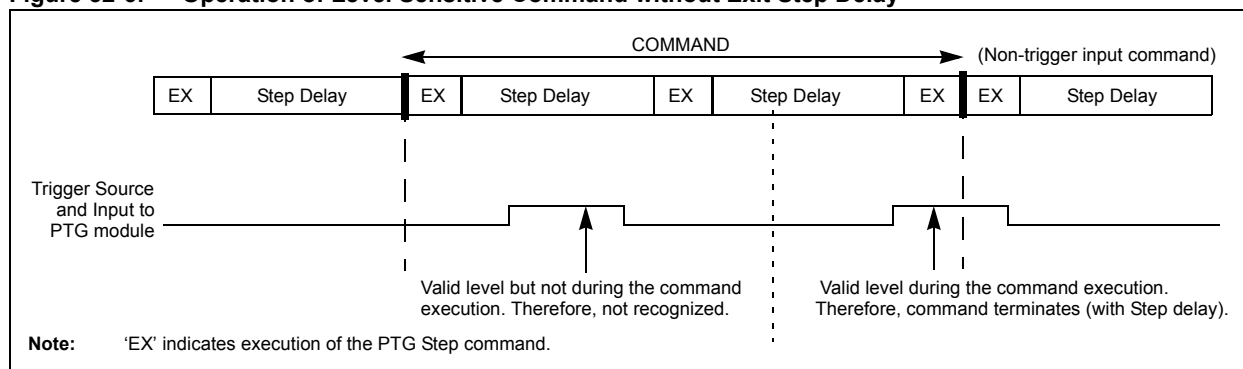
If the trigger does not occur, and the Step delay is disabled, the command immediately tests the trigger input again during the next PTG clock cycle. When the trigger occurs, the command execution completes, and execution of the subsequent command will commence immediately.

**Note 1:** As this operating mode is level sensitive, if the input trigger level is true at the start of execution of the PTGWHI or PTGWLO command, the input test will be instantly satisfied.

**2:** The input is not latched, therefore, it must be valid when the command executes in order to be recognized.

Figure 32-6 shows an example timing diagram of Mode 3 operation.

**Figure 32-6: Operation of Level Sensitive Command without Exit Step Delay**

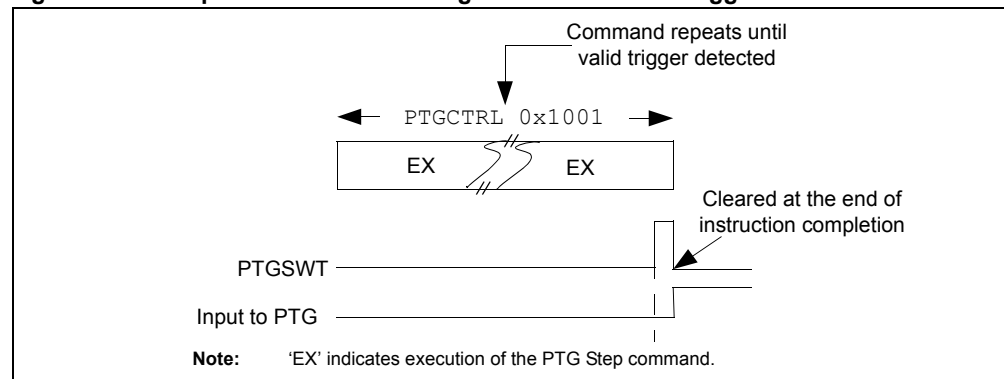


## 32.4.9.3 WAIT FOR SOFTWARE TRIGGER

The user can set either a PTGCTRL 0x1011 (edge triggered) or PTGCTRL 0x1010 (level triggered) command to wait for a software generated trigger. This trigger is generated by setting the PTGSWT bit (PTGCST<10>).

The PTGCTRL 0x1011 command is sensitive only to the PTGSWT bit transition from '0' to '1'. This transition must occur during the command execution, otherwise the command will continue to wait. The PTGSWT bit is automatically cleared by hardware on completion of the PTGCTRL 0x1011 command execution, initializing the bit for the next software trigger command. Figure 32-7 explains the operation of the wait for edge-based software trigger.

**Figure 32-7: Operation of Wait for Edge-Based Software Trigger**

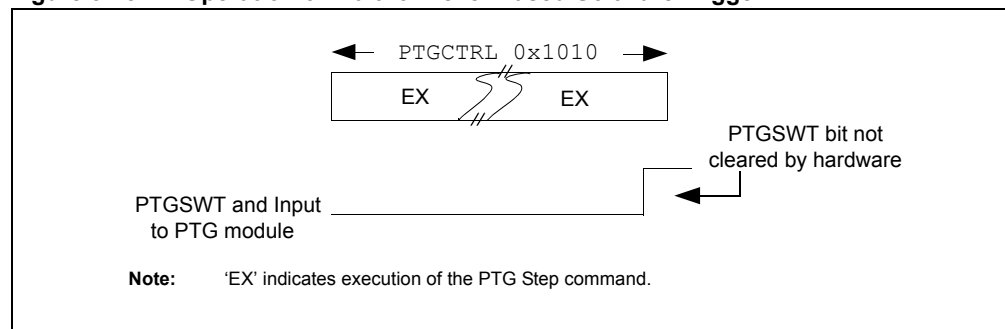




## Section 32. Peripheral Trigger Generator (PTG)

The `PTGCTRL 0x1010` command is sensitive to the level of the `PTGSWT` bit. This command waits until the `PTGSWT = 1`. It will complete immediately if `PTGSWT = 1` upon entry to the command. The `PTGSWT` bit is not automatically cleared by the `PTGCTRL 0x1010` command. If desired, the `PTGSWT` bit can be cleared by the user application on completion of the `PTGCTRL 0x1010` command execution. Figure 32-8 explains the operation of the wait for level-based software trigger.

**Figure 32-8: Operation of Wait for Level-Based Software Trigger**



The `PTGSWT` bit along with a PTG Step command generates interrupt that allows the user to coordinate activity between the PTG module and the application software.

**Note:** The level sensitive software trigger (`PTGCTRL 0x1010`) is not available on all devices. For details, refer to the specific device data sheet.

### 32.4.9.4 WAIT FOR GP TIMER

There are two 16-bit General Purpose (GP) timers (`PTGT0` and `PTGT1`), which can be used by the sequencer to wait for a specified period. All timers are cleared when the device is in the Reset state, or when the PTG module is disabled (`PTGEN = 0`). The Step commands are available for loading, modifying or initializing the GP timers.

Each GP timer consists of an incrementing timer (`PTGTx`) and a limit register (`PTGTxLIM`). The limit register value can be changed by a CPU write (when the module is disabled), or by the PTG sequencer (when the module is enabled). Data read from the limit register depends upon the state of the Counter/Timer Visibility Control bit (`PTGIVIS`).

When running, the timers increment on the rising edge of the PTG clock, which is defined in the `PTGCST` register. The user can specify a wait operation using a GP timer by executing the appropriate `PTGCTRL PTGTx` command (wait for selected GP timer).

When waiting for the GP timer, the command will wait until the value of the timer (Timer0 or Timer1) reaches its respective limit value (`PTGT0LIM` or `PTGT1LIM`). On reaching the limit value, the step command execution completes and the next command will start. The timer is also cleared for its next use.

### 32.4.9.5 STEP COMMAND DELAY

The Step Delay Timer (`SDLY`) is a convenient method to make each Step command consume a specific amount of time. Normally, the user specifies a Step delay equal to the duration of a peripheral function, such as the ADC conversion time. The Step delay enables the user to generate the trigger output signals at a controlled rate, thereby avoiding overload on the target peripheral.

The `PTGSDLIM` register defines the additional time duration of each Step command in terms of PTG clocks.

By default, the `SDLY` is disabled. The user can enable and disable the `SDLY` via the `PTGCTRL 0x0110` or `PTGCTRL 0x0010` command, which can be placed in the Step queue.

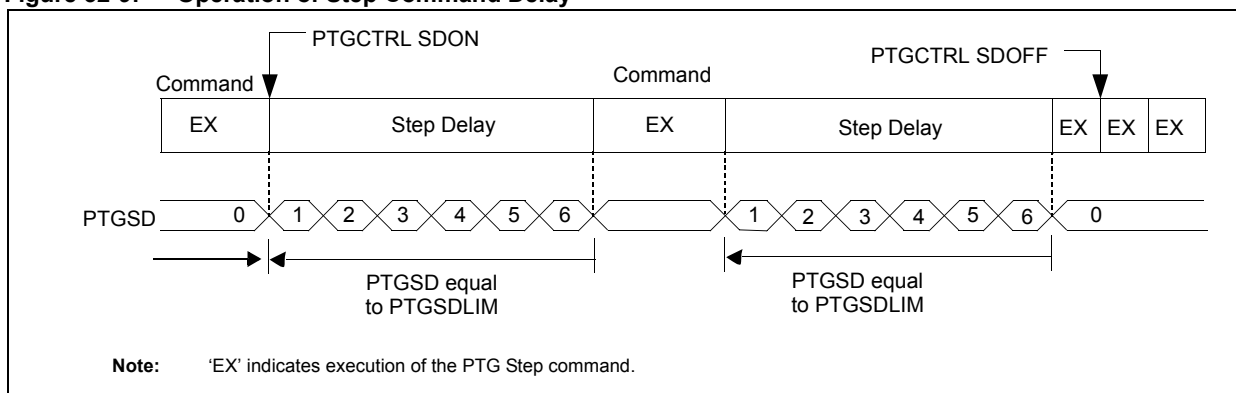


When operating, the SDLY increments at the PTG clock rate defined in the PTGCST register. The PTGSDLIM register value is referred to as the Step delay timer limit value. The Step delay is inserted after each command is executed, so that all Step commands are stalled until the PTGSD timer reaches its limit value. On reaching the limit value, the command execution completes, and the next command starts execution. The timer is also cleared during execution of each command, so that it is ready for the next command execution.

**Note:** The PTGSDLIM register value of 0x0000 does not insert the additional PTG clocks when the Step delay timer is enabled. The PTGSDLIM register value of 0x0001 inserts a single PTG Step delay (1 PTG clock) into every subsequent instruction after the Step delay timer is enabled.

The trigger sources for the edge sensitive commands (PTGCTRL 0x1011 and PTGWHI/PTGWLO when operated in Edge Sensitive mode) have an additional hardware, external to the sequencer to recognize the appropriate edge transition. The hardware is reset at the end of each command to maintain the edge sensitive nature of these input triggers. If an additional valid edge occur during a Step delay that has been inserted after the Step command has executed, it will not be recognized by any subsequent Step command. Figure 32-9 explains the operation of the Step command delay.

**Figure 32-9: Operation of Step Command Delay**



## 32.4.10 PTG Input Trigger Watchdog Timer

A Watchdog Timer (WDT) is required as the PTG can wait indefinitely for an external event when executing the following commands:

- Wait for hardware trigger positive edge or high state (PTGWHI)
- Wait for hardware trigger negative edge or low state (PTGWLO)

The WDT is enabled and it starts counting when the command starts to execute. It is disabled when the command completes execution, and prior to any Step delay insertion. All other commands execute with the predefined cycle counts.

**Note:** The PTG Watchdog Timer is not required during execution of the PTGCTRL 0x1011 or PTGCTRL 0x1010 command. It is assumed that correct operation of the device will be monitored through other means.

### 32.4.10.1 OPERATION OVERVIEW

If an expected event fails to arrive before the WDT time-out period expires, the PTG module:

1. Aborts the (failing) command underway.
2. Halts the sequencer (PTGSTRT = 0).
3. Sets PTGWDTO = 1.
4. Issues a Watchdog Error interrupt to the processor.

The user can either use the Watchdog Error interrupt, or periodically poll the PTGWDTO bit (PTGCST<6>) to determine that a WDT event has occurred.



### 32.4.10.2 CONFIGURATION

The WDT is configured by setting the PTGWDT<2:0> bits (PTGCON<2:0>). The WDT counts the PTG clocks as defined by the PTGCLK<2:0> and PTGDIV<4:0> bits in the PTGCON register. For more information, refer to [32.4.4 “PTG Clock Selection”](#). The WDT time-out count value is selected by using the PTGWDT<2:0> bits, and is disabled when the PTGWDT = 0x000.

- Note 1:** The WDT is disabled prior to insertion of any Step delay; therefore, the user does not need to account for the Step delay when calculating a suitable WDT time-out value.
- 2:** Some bits within the PTGCON register are read-only when PTGSTRT = 1 (Sequencer Executing commands). Refer to [Register 32-2](#).

### 32.4.10.3 WATCHDOG EVENT RECOVERY

If a WDT event occurs, the user has the option to take the necessary action to identify and fix the problem, and then continue the Step command sequence, or can simply restart the sequence.

To clear the PTGWDTO bit and to restart the PTG Sequencer from the start of the Step queue, disable (PTGEN = 0) and re-enable (PTGEN = 1) the PTG module, and then restart execution (PTGSTRT = 1).

Alternatively, as the sequencer is only halted (not Reset), the user has the option to examine the PTGQPTR register to identify which Step command was the source of problem, and can then take a corrective action. The offending command is aborted prior to the PTGQPTR update. Therefore, it will still address the failing command after the WDT event. After the PTGWDTO bit is cleared, the Step queue can be restarted at the same command by setting PTGSTRT = 1.

- Note:** The user should clear the PTGWDTO bit after a WDT event. Failing to clear the bit will not interfere with the subsequent module operation, but the bit will not be possible to poll any future WDT events.

### 32.4.11 PTG Module Outputs

The PTG module can generate trigger, interrupt, and strobed data outputs by execution of specific Step commands.

#### 32.4.11.1 TRIGGER OUTPUTS

The PTG module can generate up to 32 unique trigger output signals. There are two types of trigger output functions:

- Individual
- Broadcast

The PTG module can generate an individual output trigger on any one of the 32 trigger outputs using the PTGTRIG command.

The individual trigger outputs are typically used to trigger individual ADC input conversion operations, but can be assigned to any function, including general purpose I/O ports. When the PTG module is used with a compatible peripheral, such as the ADC module, the individual trigger output signals of the PTG are individually assigned to specific analog input conversion controllers within the ADC module.

The broadcast trigger output feature is specified by the PTGBTE register. Each bit in the PTGBTE register corresponds to an associated individual trigger output. If a bit is set in the PTGBTE register, and a broadcast trigger Step command (PTGCTRL 0x1111) is executed, the corresponding individual trigger output is asserted. The broadcast trigger output enables the user to simultaneously generate large number of trigger outputs with a single Step command.

#### 32.4.11.2 INTERRUPT OUTPUTS

The PTG module can generate up to 16 unique interrupt request signals. These signals are useful for interacting with an application software to create more complex functions.

The PTG module can generate an individual interrupt pulse by using the PTGIRQ command.



## 32.4.12 Strobe Output

The strobe output of the PTG module can be used to write to the AD1CHS0 register.

The `PTGCTRL 0x1110` command writes the PTGL0 register contents to the AD1CHS0 register. The PTGL0 register can be modified by using the `PTGADD` and `PTGCOPY` commands.

The `PTGCTRL 0x1100` command writes the PTGC0 register contents to the AD1CHS0 register. The `PTGCTRL 0x1101` command writes the PTGC1 register contents to the AD1CHS0 register.

### 32.4.12.1 OUTPUT TIMING

All triggers, interrupts and data strobe outputs are internally asserted by the PTG state machine, when the corresponding Step command execution starts (i.e., before any additional time specified by the Step delay timer) on the rising edge of the PTG execution clock.

**Note:** If a command has triggered the pulse width delay counter, the counter is synchronously reset with respect to the PTG clock, terminating the pulse (subject to a minimum pulse width of 1 PTG clock cycle).

In Pulse mode (`PTGTOGL = 0`), the width of the trigger output signals is determined by the `PTGPWD<3:0>` bits (`PTGCON<7:4>`), and can be any value between 1 and 16 PTG clock cycles. The default value is 1 PTG clock cycle.

Refer to [32.4.12.1.2 “TRIG Negation, when PTGTOGL = 1”](#) when operating in Toggle mode (`PTGTOGL = 1`).

When globally controlled by the `PTGCTRL BTRIG` broadcast trigger command, the TRIG output pulse width is determined by the `PTGPWD<3:0>` bits (`PTGCON<7:4>`), and can be any value between 1 and 16 PTG clock cycles. The default value is 1 PTG clock cycle.

**Note:** The trigger generated by using the `PTGCTRL BTRIG` broadcast trigger command can only operate in Pulse mode (i.e., `PTGTOGL = 'don't care'`).

#### 32.4.12.1.1 TRIG Negation, when PTGTOGL = 0

If generating an individual trigger output, and when the `PTGTOGL` bit (`PTGCST<12>`) = 0, or if generating a broadcast trigger output, the TRIG output(s) pulse width is determined by the `PTGPWD<3:0>` bits.

#### 32.4.12.1.2 TRIG Negation, when PTGTOGL = 1

If generating an individual trigger output and when the `PTGTOGL` bit (`PTGCST<12>`) = 1, the TRIG outputs will remain set until the `PTGTRIG` command is executed again. On start of the `PTGTRIG` command execution, the TRIG outputs are toggled at the beginning of the PTG execution clock.

**Note:** The `PTGTOGL` bit has no effect on the operation of the `PTGCTRL BTRIG` multiple trigger (broadcast) generation command. The exception cases are as follows:

- The pulse width of all broadcast triggers is always determined by the `PTGPWD<3:0>` bits.
- If a target trigger output is already in the logic 1 state (because `PTGTOGL` is active), the `PTGCTRL BTRIG` command will have no effect, and the trigger output will remain at logic 1.

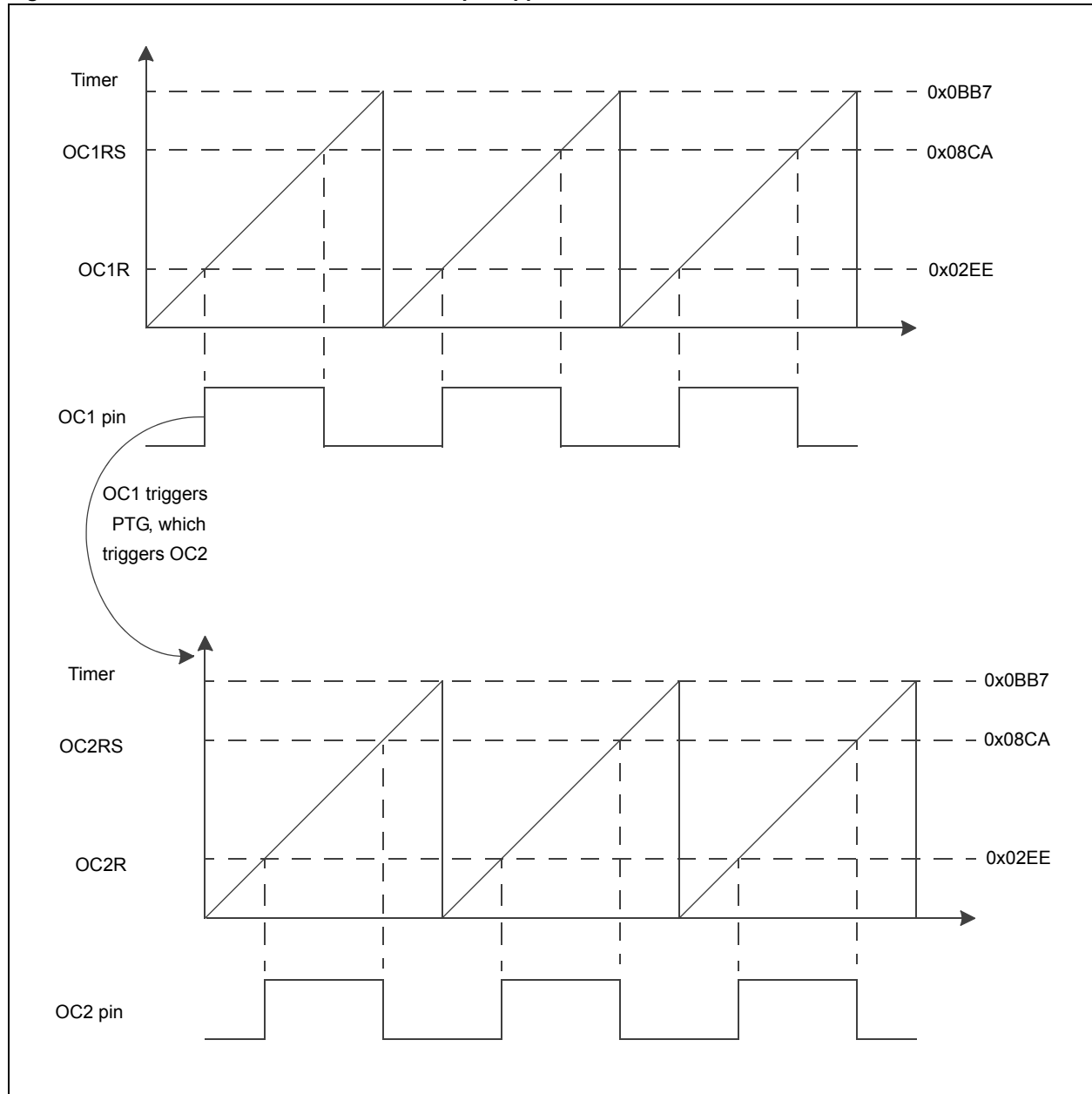


## 32.5 APPLICATION EXAMPLES

### 32.5.1 Generating Phase-shifted Waveforms

Figure 32-10 shows an application example where the user needs to generate phase-shifted waveforms. In this example, Output Compare 1 uses Timer2 as the synchronization source to generate a pulse. The rising edge of the pulse is the trigger input to the PTG module. When the trigger is sensed, the PTG module triggers Output Compare 2, which uses the PTG module as the synchronization source to generate a phase-shifted waveform.

Figure 32-10: Phase-shifted Waveform Example Application





Example 32-1 shows code for generating a phase-shifted waveform.

## Example 32-1: Generating Phase-shifted Waveforms

```
#include "p33Exxxx.h"

/* PTG Commands */
#define PTGWHI    (0x4<<4)
#define PTGTRIG   (0x8<<4)
#define PTGJMP    (0xA<<4)

_FOSCSEL(FNOSC_FRC);
_FOSC(FCKSM_CSECMD & POSCMD_XT & OSCIOFNC_OFF & IOL1WAY_OFF);
_FWDT(FWDTEN_OFF);
_FPOR(ALT12C1_ON & ALT12C2_ON);
_FICD(ICS_PGD2 & JTAGEN_OFF);

void Init_Timer(void);
void Init_Ptg(void);
void Init_PPS(void);
void Init_OC1(void);
void Init_OC2(void);

int main(void)
{
    // Configure the device PLL to obtain 60 MIPS operation. The crystal frequency is 8 MHz.
    // Divide 8 MHz by 2, multiply by 60 and divide by 2. This results in Fosc of 120 MHz.
    // The CPU clock frequency is Fcy = Fosc/2 = 60 MHz.
    PLLFBD = 58;                /* M = 30 */
    CLKDIVbits.PLLPOST = 0;     /* N1 = 2 */
    CLKDIVbits.PLLPRE = 0;     /* N2 = 2 */
    OSCTUN = 0;

    // Initiate Clock Switch to Primary
    // Oscillator with PLL (NOSC= 0x3)
    __builtin_write_OSCCONH(0x03);
    __builtin_write_OSCCONL(0x01);
    while (OSCCONbits.COSC != 0x3);
    while (_LOCK == 0);        /* Wait for PLL lock at 60 MIPS */

    Init_Timer();
    Init_Ptg();
    Init_PPS();
    Init_OC1();
    Init_OC2();

    PTGCSTbits.PTGEN = 1;      // Enable the PTG
    PTGCSTbits.PTGSTRT = 1;    // Start the PTG
    T2CONbits.TON = 1;         // Start the timer

    while(1);
}

void Init_Timer( void )
{
    // Initialize and enable Timer2
    T2CON = 0x0000; // Timer reset
    TMR2 = 0x0000; // Clear timer register
    PR2 = 0x0BB7; // Load the period value
}

void Init_Ptg( void )
{
    PTGCST = 0; // Clear the control/status register
    PTGCON = 0; // Clear the control register

    /* Program the command sequence */
    _STEP0 = PTGWHI | (0x7); // Wait for OC1 input trigger event
    _STEP1 = PTGTRIG | (0x1); // Trigger PTG02 (trig/sync for OC2)
    _STEP2 = PTGJMP | (0x0); // Jump to _STEP0
}
```



### Example 32-1: Generating Phase-shifted Waveforms (Continued)

```
void Init_PPS(void)
{
    _RP39R = 0x10; // Set up the PPS for OC1
    _RP40R = 0x11; // Set up the PPS for OC2
}

void Init_OC1(void)
{
    OC1R = 0x02EE; // Initialize the compare register
    OC1RS = 0x08CA; // Initialize the secondary compare register

    // Initialize Output Compare Module
    OC1CON1 = 0x0; // Clear all control bits
    OC1CON2 = 0x0; // Clear all control bits
    OC1CON1bits.OCTSEL = 0x7; // Select peripheral clock as clock source
    OC1CON2bits.SYNCSEL = 0xC; // Select Timer2 as sync source
    OC1CON1bits.OCM = 0x5; // Double compare continuous pulse mode
}

void Init_OC2(void)
{
    OC2R = 0x02EE; // Initialize the compare register
    OC2RS = 0x08CA; // Initialize the secondary compare register

    // Initialize Output Compare Module
    OC2CON1 = 0x0; // Clear all control bits
    OC2CON2 = 0x0; // Clear all control bits
    OC2CON1bits.OCTSEL = 0x7; // Select peripheral clock as clock source
    OC2CON2bits.SYNCSEL = 0xA; // Select PTG as sync source
    OC2CON1bits.OCM = 0x5; // Double compare continuous pulse mode
}
```



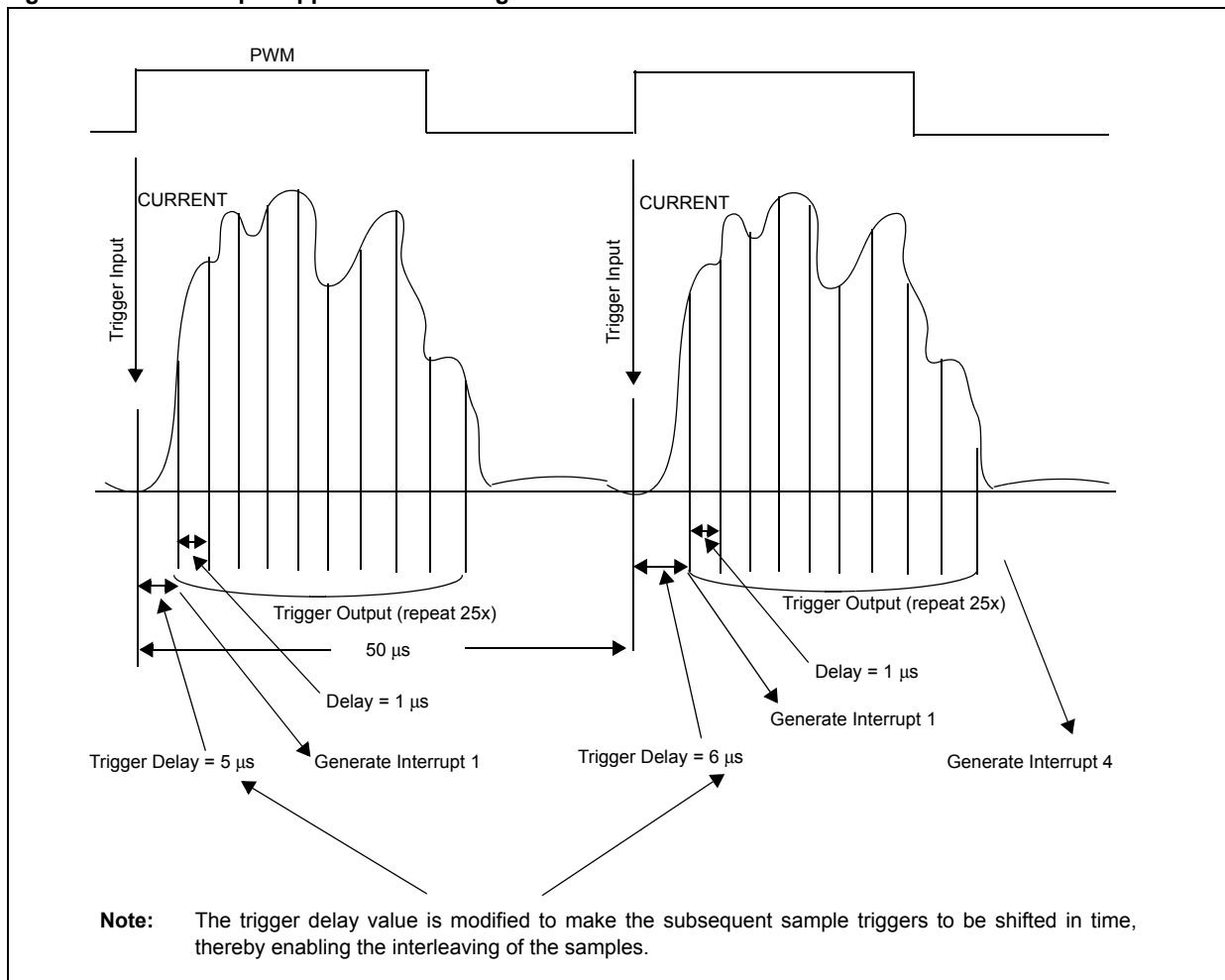
## 32.5.2 Interleaving Samples Over Multiple Cycles

Figure 32-11 shows the waveforms of an application where the user needs to accurately measure the power in a system where the current load is highly dependent on temperature, voltage, and user application. The current waveforms vary widely per user usage, but over a few PWM cycles, the waveforms are relatively stable.

The aim of this example is to collect many current and/or voltage readings over several PWM cycles in an interleaved manner. The data is stored in the memory during acquisition and is later processed (integrated) to yield an accurate power value.

This example shows a situation where it would not be practical or possible for software to accurately schedule the ADC samples.

Figure 32-11: Example Application - Average Power Calculation





## 32.5.3 Interleaved Sampling Step Command Program

This section describes the Step command programming for implementing the timing sequence shown in Figure 32-11.

The following assumptions are made:

1. Trigger Input 1 is connected to the PWM signal. The rising edge of the PWM signal starts the sequence.
2. Output Trigger 3 is connected to the ADC module. This signal gives command to the ADC module to begin a sample and conversion process.
3. Interrupt 1 is used to indicate the processor that a sub sequence has started (provides status).
4. Interrupt 4 is used to indicate the processor that the complete sequence has completed.
5. The ADC clock is selected as the PTG clock source.
6. The ADC clock is 14 MHz.
7. The initial trigger delay is 5  $\mu$ s.
8. The second trigger delay is 6  $\mu$ s.
9. In each PWM cycle, the ADC will be triggered 25 times.
10. The basic sequence is executed twice.

Initialize the following control registers:

- PTGT0LIM = 0x0046 (5  $\mu$ s x 14 clocks/ $\mu$ s)
- PTGT1LIM = 0x000B (1  $\mu$ s x 14 clocks/ $\mu$ s) - three Step clocks)
- PTGC0LIM = 0x0018 (total of 25 inner loop iterations)
- PTGC1LIM = 0x0001 (total of two outer loop iterations)
- PTGHOLD = 0x0046 (5  $\mu$ s x 14 clocks/ $\mu$ s)
- PTGADJ = 0x000E (1  $\mu$ s x 14 clocks/ $\mu$ s)
- PTGSDLIM = 0x0000 (no Step delay)
- PTGBTE = 0x0000 (no broadcast triggers)
- PTGQPTR = 0x0000 (start of Step queue)
- PTGCST = 0x8200 (after PTGQPTR is initialized)

### Example 32-2: Step Commands in PTGQUE

```
/* PTG Commands */
#define PTGCTRL (0x0<<0)
#define PTGADD (0x1<<4)
#define PTGCOPY (0x1<<4)
#define PTGWHI (0x4<<4)
#define PTGIRQ (0x7<<4)
#define PTGTRIG (0x8<<4)
#define PTGJMP (0xA<<4)
#define PTGJMPC0 (0xC<<4)
#define PTGJMPC1 (0xE<<4)

// Outer loop
_STEP0 = PTGWHI | 0x1; // Wait for positive edge trigger 1
_STEP1 = PTGCTRL | 0x8; // Start PTGT0, wait for time out
_STEP2 = PTGIRQ | 0x1; // Generate IRQ 1

// Inner loop
_STEP3 = PTGTRIG | 0x3; // Generate output trigger 3
_STEP4 = PTGCTRL | 0x9; // Start PTGT1, wait for time out
_STEP5 = PTGJMPC0 | 0x3; // Jump to STEP3 if PTGC0! = PTGC0LIM, increment PTGC0

// End inner loop
_STEP6 = PTGADD | 0x1; // Add PTGADJ to PTGT0LIM
_STEP7 = PTGJMPC1 | 0x0; // Jump to STEP0 if PTGC1! = PTGC1LIM, increment PTGC1

// End outer loop
_STEP8 = PTGIRQ | 0x4; // Generate IRQ 4
_STEP9 = PTGCOPY | 0x8; // Copy PTGHOLD to PTGT0LIM (restore original value)
_STEP10 = PTGJMP | 0x0; // Jump to start of queue
```

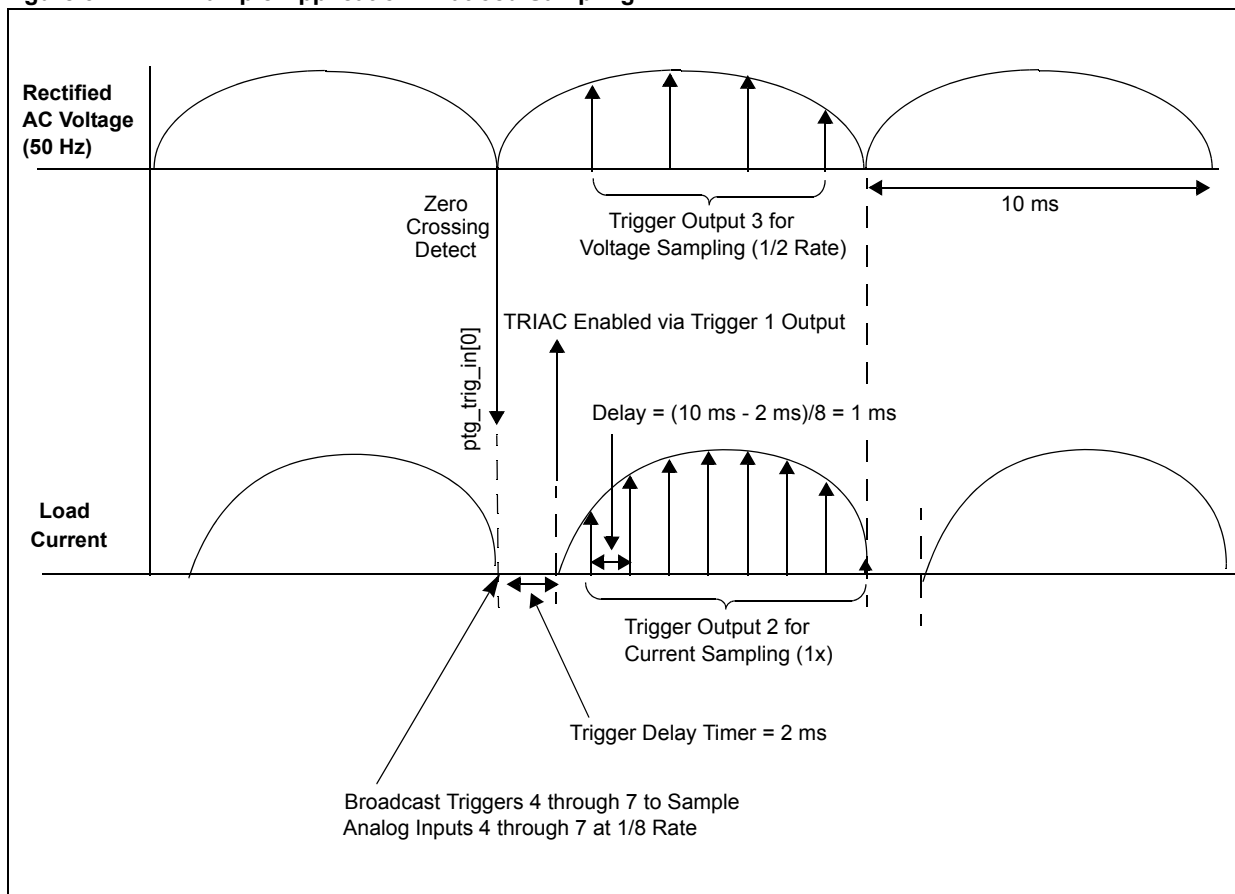


## 32.5.4 Sampling at Multiple Rates

Figure 32-12 shows an application example wherein the aim is to sample an ADC input at a fast rate (1x rate), a second analog input at a slower rate (1/2 rate), and Analog Inputs 4 through 7 at an 1/8 rate. The example is of a motor control application using a Silicon Controlled Rectifier (SCR) that triggers at a specified time after the AC line zero crossing.

While this example uses the simple binary sampling ratios, the PTG module can generate a very wide range of sample ratios to meet the requirements of an application.

Figure 32-12: Example Application - Ratioed Sampling





## 32.5.5 Ratioed Sampling Step Command Program

This section describes the Step command programming for implementing the timing sequence shown in [Figure 32-12](#).

The following assumptions are made:

- Trigger Input 0 is connected to the zero crossing detect. The rising edge of the zero crossing detect signal starts the sequence
- The trigger delay from Trigger Input 0 to the generation of Trigger Output 1 is 2 ms
- Trigger Output 1 enables the SCR in the application circuit
- Trigger Output 2 is connected to the ADC to trigger sampling of the current measurement at 1 ms intervals
- Trigger Output 3 is connected to the ADC to trigger sampling of the supply voltage measurement at 2 ms intervals
- Trigger Outputs 4, 5, 6 and 7 are connected to the ADC to sample other data values once per cycle
- The ADC clock is selected as the PTG clock source
- The ADC clock is 14 MHz

Initialize the following control registers:

- PTGT0LIM = 0x6D60 (2 ms x 14 clocks/ $\mu$ s)
- PTGT1LIM = 0x36B0 (1 ms x 14 clocks/ $\mu$ s)
- PTGC0LIM = 0x0018 (total of 25 inner loop iterations)
- PTGC1LIM = 0x0001 (total of two outer loop iterations)
- PTGHOLD = 0x0000 (not used)
- PTGADJ = 0x0000 (not used)
- PTGSDLIM = 0x0000 (no Step delay)
- PTGBTE = 0x00F0 (enable broadcast triggers 4-7)
- PTGQPTR = 0x0000 (start of Step queue)
- PTGCST = 0x8200 (after PTGQPTR is initialized)

### 32.5.5.1 CALCULATING TIME WITH MORE ACCURACY

As each Step command takes at least two clocks, for more accurate timing, the PTGTDLY register should be programmed with a value that compensates for the delay of the wait for trigger command and the generate triggers 4-7 command, and the wait for trigger delay command. Therefore, the PTGTDLY initialization value should be  $28,000 - 6 = 27,994$ .

Similarly, the PTGTMR register value should be a slightly smaller value of  $14,000 - 4 = 13,996$ .

### Example 32-3: Step Commands in PTGQUE Register

```
/* PTG Commands */
#define PTGCTRL (0x0<<0)
#define PTGWHI (0x4<<4)
#define PTGTRIG (0x8<<4)
#define PTGJMP (0xA<<4)
#define PTGJMPC0 (0xC<<4)

_STEP0 = PTGWHI | 0x0; // Wait for positive edge trigger 0
_STEP1 = PTGCTRL | 0xF; // Generate output triggers 7, 6, 5 and 4 (broadcast)
_STEP2 = PTGCTRL | 0x8; // Start PTGT0, wait for time-out
_STEP3 = PTGTRIG | 0x1; // Generate output trigger 1

// Start main loop
_STEP4 = PTGCTRL | 0x9; // Start PTGT1, wait for time-out
_STEP5 = PTGTRIG | 0x2; // Generate output trigger 2
_STEP6 = PTGTRIG | 0x3; // Generate output trigger 3
_STEP7 = PTGCTRL | 0x9; // Start PTGT1, wait for time-out
_STEP8 = PTGTRIG | 0x2; // Generate output trigger 2
_STEP9 = PTGJMPC0 | 0x4; // Jump to STEP4 if PTGC0! = PTGC0LIM, increment PTGC0

// End main loop
_STEP10 = PTGJMP | 0x0; // Jump to start of queue
```



## 32.6 POWER-SAVING MODES

The PTG module supports three power-saving modes:

- Disabled: The PTG module is not clocked in this mode
- Idle: The processor core and selected peripherals are shut down
- Sleep: The entire device is shut down

### 32.6.1 Disabled Mode

When PTGEN = 1, the module is considered in an active mode and is fully powered and functional. When PTGEN = 0, the module is turned off. The PTG clock portions of the circuit are disabled for maximum current savings. Only the control registers remain functional for reading and writing to allow the software to change the module's operational mode. The module sequencer is reset.

### 32.6.2 Idle Mode

To continue full module operation while the PTG module is in Idle mode, the PTGSIDL bit must be cleared prior to entry into Idle mode. If PTGSIDL = 1, the module will behave the same way in Idle mode as it does in Sleep mode.

### 32.6.3 Sleep Mode

If the PTG module enters Sleep mode while the module is enabled (PTGEN = 1), the module will be suspended in its current state until clock execution resumes. This situation should be avoided as it might result in unexpected operation. It is recommended that all peripherals be shut down in an orderly manner prior to entering Sleep mode.



## 32.7 REGISTER MAP

**Table 32-2: PTG Register Map**

File Name	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	All Resets
PTGCST	PTGEN	—	PTGSIDL	PTGTOGL	—	PTGSWT	PTGSSEN	PTGIVIS	PTGSTRT	PTGWDTO	—	—	—	—	PTGITM<1:0>		0000
PTGCON	PTGCLK<2:0>			PTGDIV<4:0>					PTGPWD<3:0>				—	PTGWDT<2:0>			0000
PTGBTE	ADCTS4	ADCTS3	ADCTS2	ADCTS1	IC4TSS	IC3TSS	IC2TSS	IC1TSS	OC4CS	OC3CS	OC2CS	OC1CS	OC4TSS	OC3TSS	OC2TSS	OC1TSS	0000
PTGT0LIM	PTGT0LIM<15:0>																0000
PTGT1LIM	PTGT1LIM<15:0>																0000
PTGSDLIM	PTGSDLIM<15:0>																0000
PTGC0LIM	PTGC0LIM<15:0>																0000
PTGC1LIM	PTGC1LIM<15:0>																0000
PTGHOLD	PTGHOLD<15:0>																0000
PTGADJ	PTGADJ<15:0>																0000
PTGL0	PTGL0<15:0>																0000
PTGQPTR	—	—	—	—	—	—	—	—	—	—	—	PTGQPTR<4:0>				0000	
PTGQUEn	STEPx<7:0> <sup>(2)</sup>								STEPy<7:0> <sup>(3)</sup>								xxxx

**Legend:** — = unimplemented, read as '0'; 'x' = unknown value at Reset. Reset values are shown in hexadecimal.

- Note**
- 1: The actual number of PTG queue registers is parameterized. The number shown in this table corresponds to `PTG_QUEUE_SIZE = 16` (equivalent to 32 Step commands).
  - 2: An 'x' in the bit name refers to the odd-numbered step queue pointers.
  - 3: A 'y' in the bit name refers to the even-numbered step queue pointers.



## 32.8 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 dsPIC33E/PIC24E product family, but the concepts are pertinent and could be used with modification and possible limitations. The current application notes related to the Peripheral Trigger Generator (PTG) module include the following:

Title	Application Note #
No application notes at this point of time.	N/A

**Note:** Please visit the Microchip web site ([www.microchip.com](http://www.microchip.com)) for additional Application Notes and code examples for the dsPIC33E/PIC24E family of devices.



### 32.9 REVISION HISTORY

#### Revision A (September 2011)

This is the initial released version of this document.



NOTES:



---

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

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as "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.

---

Information contained in this publication 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. 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 ITS CONDITION, QUALITY, PERFORMANCE, MERCHANTABILITY OR FITNESS FOR PURPOSE. Microchip disclaims all liability arising from this information and its use. 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.

#### **Trademarks**

The Microchip name and logo, the Microchip logo, dsPIC, KEELOQ, KEELOQ logo, MPLAB, PIC, PICmicro, PICSTART, PIC<sup>32</sup> logo, rfPIC and UNI/O are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.


FilterLab, Hampshire, HI-TECH C, Linear Active Thermistor, MXDEV, MXLAB, SEEVAL and The Embedded Control Solutions Company are registered trademarks of Microchip Technology Incorporated in the U.S.A.

Analog-for-the-Digital Age, Application Maestro, chipKIT, chipKIT logo, CodeGuard, dsPICDEM, dsPICDEM.net, dsPICworks, dsSPEAK, ECAN, ECONOMONITOR, FanSense, HI-TIDE, In-Circuit Serial Programming, ICSP, Mindi, MiWi, MPASM, MPLAB Certified logo, MPLIB, MPLINK, mTouch, Omniscent Code Generation, PICC, PICC-18, PICDEM, PICDEM.net, PICKit, PICtail, REAL ICE, rfLAB, Select Mode, Total Endurance, TSHARC, UniWinDriver, WiperLock 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.

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

© 2011, Microchip Technology Incorporated, Printed in the U.S.A., All Rights Reserved.

 Printed on recycled paper.

ISBN: 978-1-61341-609-9

*Microchip received ISO/TS-16949:2009 certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona; Gresham, Oregon and design centers in California and India. The Company's quality system processes and procedures are for its PIC® MCUs and dsPIC® DSCs, KEELOQ® code hopping devices, Serial EEPROMs, microperipherals, nonvolatile memory and analog products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001:2000 certified.*

**QUALITY MANAGEMENT SYSTEM**  
**CERTIFIED BY DNV**  
**== ISO/TS 16949:2009 ==**



## Worldwide Sales and Service

### AMERICAS

**Corporate Office**  
2355 West Chandler Blvd.  
Chandler, AZ 85224-6199  
Tel: 480-792-7200  
Fax: 480-792-7277  
Technical Support:  
<http://www.microchip.com/support>  
Web Address:  
[www.microchip.com](http://www.microchip.com)

**Atlanta**  
Duluth, GA  
Tel: 678-957-9614  
Fax: 678-957-1455

**Boston**  
Westborough, MA  
Tel: 774-760-0087  
Fax: 774-760-0088

**Chicago**  
Itasca, IL  
Tel: 630-285-0071  
Fax: 630-285-0075

**Cleveland**  
Independence, OH  
Tel: 216-447-0464  
Fax: 216-447-0643

**Dallas**  
Addison, TX  
Tel: 972-818-7423  
Fax: 972-818-2924

**Detroit**  
Farmington Hills, MI  
Tel: 248-538-2250  
Fax: 248-538-2260

**Indianapolis**  
Noblesville, IN  
Tel: 317-773-8323  
Fax: 317-773-5453

**Los Angeles**  
Mission Viejo, CA  
Tel: 949-462-9523  
Fax: 949-462-9608

**Santa Clara**  
Santa Clara, CA  
Tel: 408-961-6444  
Fax: 408-961-6445

**Toronto**  
Mississauga, Ontario,  
Canada  
Tel: 905-673-0699  
Fax: 905-673-6509

### ASIA/PACIFIC

**Asia Pacific Office**  
Suites 3707-14, 37th Floor  
Tower 6, The Gateway  
Harbour City, Kowloon  
Hong Kong  
Tel: 852-2401-1200  
Fax: 852-2401-3431

**Australia - Sydney**  
Tel: 61-2-9868-6733  
Fax: 61-2-9868-6755

**China - Beijing**  
Tel: 86-10-8569-7000  
Fax: 86-10-8528-2104

**China - Chengdu**  
Tel: 86-28-8665-5511  
Fax: 86-28-8665-7889

**China - Chongqing**  
Tel: 86-23-8980-9588  
Fax: 86-23-8980-9500

**China - Hangzhou**  
Tel: 86-571-2819-3187  
Fax: 86-571-2819-3189

**China - Hong Kong SAR**  
Tel: 852-2401-1200  
Fax: 852-2401-3431

**China - Nanjing**  
Tel: 86-25-8473-2460  
Fax: 86-25-8473-2470

**China - Qingdao**  
Tel: 86-532-8502-7355  
Fax: 86-532-8502-7205

**China - Shanghai**  
Tel: 86-21-5407-5533  
Fax: 86-21-5407-5066

**China - Shenyang**  
Tel: 86-24-2334-2829  
Fax: 86-24-2334-2393

**China - Shenzhen**  
Tel: 86-755-8203-2660  
Fax: 86-755-8203-1760

**China - Wuhan**  
Tel: 86-27-5980-5300  
Fax: 86-27-5980-5118

**China - Xian**  
Tel: 86-29-8833-7252  
Fax: 86-29-8833-7256

**China - Xiamen**  
Tel: 86-592-2388138  
Fax: 86-592-2388130

**China - Zhuhai**  
Tel: 86-756-3210040  
Fax: 86-756-3210049

### ASIA/PACIFIC

**India - Bangalore**  
Tel: 91-80-3090-4444  
Fax: 91-80-3090-4123

**India - New Delhi**  
Tel: 91-11-4160-8631  
Fax: 91-11-4160-8632

**India - Pune**  
Tel: 91-20-2566-1512  
Fax: 91-20-2566-1513

**Japan - Yokohama**  
Tel: 81-45-471- 6166  
Fax: 81-45-471-6122

**Korea - Daegu**  
Tel: 82-53-744-4301  
Fax: 82-53-744-4302

**Korea - Seoul**  
Tel: 82-2-554-7200  
Fax: 82-2-558-5932 or  
82-2-558-5934

**Malaysia - Kuala Lumpur**  
Tel: 60-3-6201-9857  
Fax: 60-3-6201-9859

**Malaysia - Penang**  
Tel: 60-4-227-8870  
Fax: 60-4-227-4068

**Philippines - Manila**  
Tel: 63-2-634-9065  
Fax: 63-2-634-9069

**Singapore**  
Tel: 65-6334-8870  
Fax: 65-6334-8850

**Taiwan - Hsin Chu**  
Tel: 886-3-5778-366  
Fax: 886-3-5770-955

**Taiwan - Kaohsiung**  
Tel: 886-7-536-4818  
Fax: 886-7-330-9305

**Taiwan - Taipei**  
Tel: 886-2-2500-6610  
Fax: 886-2-2508-0102

**Thailand - Bangkok**  
Tel: 66-2-694-1351  
Fax: 66-2-694-1350

### EUROPE

**Austria - Wels**  
Tel: 43-7242-2244-39  
Fax: 43-7242-2244-393

**Denmark - Copenhagen**  
Tel: 45-4450-2828  
Fax: 45-4485-2829

**France - Paris**  
Tel: 33-1-69-53-63-20  
Fax: 33-1-69-30-90-79

**Germany - Munich**  
Tel: 49-89-627-144-0  
Fax: 49-89-627-144-44

**Italy - Milan**  
Tel: 39-0331-742611  
Fax: 39-0331-466781

**Netherlands - Druenen**  
Tel: 31-416-690399  
Fax: 31-416-690340

**Spain - Madrid**  
Tel: 34-91-708-08-90  
Fax: 34-91-708-08-91

**UK - Wokingham**  
Tel: 44-118-921-5869  
Fax: 44-118-921-5820