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
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
Section 32. Peripheral Trigger Generator (PTG)

Figure 32-1: PTG Block Diagram

Note 1: This is a dedicated Watchdog Timer for the PTG module and is independent of the device watchdog timer.
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\(^{(1,2)}\)
  This register is used to enable the broadcast trigger for the PTG module.

- **PTG0LIM**: PTG Timer0 Limit Register\(^{(1)}\)
  This register determines the limit value for Timer0.

- **PTG1LIM**: PTG Timer1 Limit Register\(^{(1)}\)
  This register determines the limit value for Timer1.

- **PTGSDLIM**: PTG Step Delay Limit Register\(^{(1,2)}\)
  This register determines the limit value for the Step delay.

- **PTGC0LIM**: PTG Counter 0 Limit Register\(^{(1)}\)
  This register determines the limit value for Counter 0.

- **PTGC1LIM**: PTG Counter 1 Limit Register\(^{(1)}\)
  This register determines the limit value for Counter 1.

- **PTGHOLD**: PTG Hold Register\(^{(1)}\)
  This register holds the value that can be copied to the PTGTxLIM, PTGCxLIM, PTDSDELIM or PTGL0 register using the **PTGCOPY** command.

- **PTGADD**: PTG Adjust Register\(^{(1)}\)
  This register holds the value that can be added to the PTGTxLIM, PTGCxLIM, PTDSDELIM or PTGL0 register using the **PTGADD** command.

- **PTGL0**: PTG Literal 0 Register\(^{(1)}\)
  This register holds the value that can be written to the AD1CHS0 register using the **PTGCTRL** command.

- **PTGQPTR**: PTG Step Queue Pointer Register\(^{(1)}\)
  This register points to the currently active Step command in the Step queue.

- **PTGQUEn**: PTG Step Queue Pointer Register (n = 0-15)\(^{(1)}\)
  This register is the queue register for the PTG module.
Section 32. Peripheral Trigger Generator (PTG)

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

<table>
<thead>
<tr>
<th>Bit 15-0</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>R/W-0</td>
<td>PTGEN</td>
<td>Module Enable bit: 1 = PTG module is enabled, 0 = PTG module is disabled</td>
</tr>
<tr>
<td>U-0</td>
<td>PTGSIDL</td>
<td>Stop in Idle Mode bit: 1 = Discontinue the module operation when the device enters Idle mode, 0 = Continue the module operation in Idle mode</td>
</tr>
<tr>
<td>R/W-0</td>
<td>PTGTOGL</td>
<td>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</td>
</tr>
<tr>
<td>U-0</td>
<td>PTGSWT</td>
<td>Software Trigger bit: 1 = Trigger the PTG module, 0 = No action (clearing this bit will have no effect)</td>
</tr>
<tr>
<td>U-0</td>
<td>PTGSSEN</td>
<td>Enable Single-Step bit: 1 = Enable Single-Step mode, 0 = Disable Single-Step mode</td>
</tr>
<tr>
<td>U-0</td>
<td>PTGIVIS</td>
<td>Counter/Timer Visibility Control bit: 1 = Reading PTGSDLIM, PTGCxLIM or PTGTXLIM registers return the current values of their corresponding counter/timer registers (PTGS, PTGCx, and PTGTX), 0 = Reading PTGSDLIM, PTGCxLIM or PTGTXLIM registers return the value of these limit registers</td>
</tr>
<tr>
<td>R/W-0</td>
<td>PTGSTRT</td>
<td>Start PTG Sequencer bit: 1 = Start to sequentially execute the commands (Continuous mode), 0 = Stop executing the commands</td>
</tr>
<tr>
<td>U-0</td>
<td>PTGWDTO</td>
<td>PTG Watchdog Timer Time-out Status bit: 1 = PTG Watchdog Timer has timed out, 0 = PTG Watchdog Timer has not timed out</td>
</tr>
<tr>
<td>U-0</td>
<td>PTGITM&lt;1:0&gt;</td>
<td>PTG Input Trigger Command Operating Mode bits: 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)</td>
</tr>
</tbody>
</table>

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

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.
Register 32-2:  PTGCON: PTG Control Register

<table>
<thead>
<tr>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>PTGCLK&lt;2:0&gt;</td>
<td>PTGDIV&lt;4:0&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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
- 11101 = Divide by 30
- 11100 = Divide by 29
- 11011 = Divide by 28
- 11010 = Divide by 27
- 11001 = Divide by 26
- 11000 = Divide by 25
- 10111 = Divide by 24
- 10110 = Divide by 23
- 10101 = Divide by 22
- 10100 = Divide by 21
- 10011 = Divide by 20
- 10010 = Divide by 19
- 10001 = Divide by 18
- 10000 = Divide by 17

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
- 1101 = All trigger outputs are 14 PTG clock cycles wide
- 1100 = All trigger outputs are 13 PTG clock cycles wide
- 1011 = All trigger outputs are 12 PTG clock cycles wide
- 1010 = All trigger outputs are 11 PTG clock cycles wide
- 1001 = All trigger outputs are 10 PTG clock cycles wide
- 1000 = All trigger outputs are 9 PTG clock cycles 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

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
### Register 32-3: PTGBTE: PTG Broadcast Trigger Enable Register

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

**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

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

**Note 2:** This register is only used with the PTGCTRL OPTION = 1111 Step command.
Register 32-3: PTGBTE: PTG Broadcast Trigger Enable Register\(^{(1,2)}\) (Continued)

bit 4 **OC1CS**: 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 **OC4TSS**: 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 **OC3TSS**: 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 **OC2TSS**: 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 **OC1TSS**: 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(1)

<table>
<thead>
<tr>
<th>R/W</th>
<th>R/W</th>
<th>R/W</th>
<th>R/W</th>
<th>R/W</th>
<th>R/W</th>
<th>R/W</th>
<th>R/W</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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(1)

<table>
<thead>
<tr>
<th>R/W</th>
<th>R/W</th>
<th>R/W</th>
<th>R/W</th>
<th>R/W</th>
<th>R/W</th>
<th>R/W</th>
<th>R/W</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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).
Register 32-6:  PTGSDLIM: PTG Step Delay Limit Register\(^{(1,2)}\)

<table>
<thead>
<tr>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PTGSDLIM&lt;15:8&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bit 15</td>
<td>bit 8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PTGSDLIM&lt;7:0&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bit 7</td>
<td>bit 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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).

Note 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\(^{(1)}\)

<table>
<thead>
<tr>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PTGC0LIM&lt;15:8&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bit 15</td>
<td>bit 8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PTGC0LIM&lt;7:0&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bit 7</td>
<td>bit 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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\(^{(1)}\)

<table>
<thead>
<tr>
<th></th>
<th>R/W</th>
<th>R/W</th>
<th>R/W</th>
<th>R/W</th>
<th>R/W</th>
<th>R/W</th>
<th>R/W</th>
</tr>
</thead>
<tbody>
<tr>
<td>PTGC1LIM&lt;15:8&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bit 15</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>bit 8</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Legend:
- R = Readable bit
- W =Writable bit
- U = Unimplemented bit, read as ‘0’
- ‘1’ = Bit is set
- ‘0’ = Bit is cleared
- x = Bit is unknown

bit 15-0  \(\text{PTGC1LIM}<15:0>: \) 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\(^{(1)}\)

<table>
<thead>
<tr>
<th></th>
<th>R/W</th>
<th>R/W</th>
<th>R/W</th>
<th>R/W</th>
<th>R/W</th>
<th>R/W</th>
<th>R/W</th>
</tr>
</thead>
<tbody>
<tr>
<td>PTGHOLD&lt;15:8&gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bit 15</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>bit 8</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Legend:
- R = Readable bit
- W = Writable bit
- U = Unimplemented bit, read as ‘0’
- ‘1’ = Bit is set
- ‘0’ = Bit is cleared
- x = Bit is unknown

bit 15-0  \(\text{PTGHOLD}<15:0>: \) 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).
Register 32-10: PTGADJ: PTG Adjust Register\(^{(1)}\)

<table>
<thead>
<tr>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>bit 15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

PTGADJ<15:8>

<table>
<thead>
<tr>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>bit 7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

PTGADJ<7:0>

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\(^{(1)}\)

<table>
<thead>
<tr>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>bit 15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

PTGL0<15:8>

<table>
<thead>
<tr>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>bit 7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

PTGL0<7:0>

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(1)

<table>
<thead>
<tr>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
<th>U-0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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)(1)

<table>
<thead>
<tr>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
<th>R/W-0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>STEPx&lt;7:0&gt;(2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>bit 15-8  STEPx&lt;7:0&gt;: PTG Step Queue Pointer Register bits(2)</td>
</tr>
</tbody>
</table>

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(2)

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.
## 32.3  STEP COMMANDS AND FORMAT

Table 32-1:  PTG Step Command Format and Description

<table>
<thead>
<tr>
<th>Step Command Byte</th>
<th>CMD&lt;3:0&gt;</th>
<th>OPTION&lt;3:0&gt;</th>
<th>Command Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bit 7</td>
<td>bit 4</td>
<td>bit 3</td>
<td>bit 0</td>
</tr>
</tbody>
</table>

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

### Note 2:
This feature is only available on dsPIC33EPXXXMC20X/50X and PIC24EPXXXMC20X devices.
### TABLE 32-1: PTG STEP COMMAND FORMAT (CONTINUED)

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

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

**Note 2:** This feature is only available on dsPIC33EPXXXMC20X/50X and PIC24EPXXXMC20X devices.
### TABLE 32-1: PTG STEP COMMAND FORMAT (CONTINUED)

<table>
<thead>
<tr>
<th>bit 3-0 Step Command</th>
<th>OPTION&lt;3:0&gt;</th>
<th>Option Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PTGWHI (1) or PTGWLO (1)</td>
<td>0000</td>
<td>PWM Special Event Trigger (2).</td>
</tr>
<tr>
<td></td>
<td>0001</td>
<td>PWM Master Timebase Synchronization Output (2).</td>
</tr>
<tr>
<td></td>
<td>0010</td>
<td>PWM1 Interrupt.</td>
</tr>
<tr>
<td></td>
<td>0011</td>
<td>PWM2 Interrupt.</td>
</tr>
<tr>
<td></td>
<td>0100</td>
<td>PWM3 Interrupt.</td>
</tr>
<tr>
<td></td>
<td>0101</td>
<td>Reserved; do not use.</td>
</tr>
<tr>
<td></td>
<td>0110</td>
<td>Reserved; do not use.</td>
</tr>
<tr>
<td></td>
<td>0111</td>
<td>OC1 Trigger Event.</td>
</tr>
<tr>
<td></td>
<td>1000</td>
<td>OC2 Trigger Event.</td>
</tr>
<tr>
<td></td>
<td>1001</td>
<td>IC1 Trigger Event.</td>
</tr>
<tr>
<td></td>
<td>1010</td>
<td>CMP1 Trigger Event.</td>
</tr>
<tr>
<td></td>
<td>1011</td>
<td>CMP2 Trigger Event.</td>
</tr>
<tr>
<td></td>
<td>1100</td>
<td>CMP3 Trigger Event.</td>
</tr>
<tr>
<td></td>
<td>1101</td>
<td>CMP4 Trigger Event.</td>
</tr>
<tr>
<td></td>
<td>1110</td>
<td>ADC Conversion Done Interrupt.</td>
</tr>
<tr>
<td></td>
<td>1111</td>
<td>INT2 External Interrupt.</td>
</tr>
<tr>
<td>PTGIRO (1)</td>
<td>0000</td>
<td>Generate PTG Interrupt 0.</td>
</tr>
<tr>
<td></td>
<td>0001</td>
<td>Generate PTG Interrupt 1.</td>
</tr>
<tr>
<td></td>
<td>0010</td>
<td>Generate PTG Interrupt 2.</td>
</tr>
<tr>
<td></td>
<td>0011</td>
<td>Generate PTG Interrupt 3.</td>
</tr>
<tr>
<td></td>
<td>0100</td>
<td>Reserved; do not use.</td>
</tr>
<tr>
<td></td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td></td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td></td>
<td>1111</td>
<td>Reserved; do not use.</td>
</tr>
<tr>
<td>PTGTRIG</td>
<td>000000</td>
<td>PTGO0.</td>
</tr>
<tr>
<td></td>
<td>000001</td>
<td>PTGO1.</td>
</tr>
<tr>
<td></td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td></td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td></td>
<td>11110</td>
<td>PTGO30.</td>
</tr>
<tr>
<td></td>
<td>11111</td>
<td>PTGO31.</td>
</tr>
</tbody>
</table>

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

**Note 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 PTG 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.

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.

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.
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:**

\[ T_{DLYEN} = 4 \times \text{PTG clock period (Maximum)} \]

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).

Note 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.

![Figure 32-2: Implementing Block Loop Diagram](image)

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>).

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.
Section 32. Peripheral Trigger Generator (PTG)

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

Note: ‘EX’ indicates execution of the PTG Step command.
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

<table>
<thead>
<tr>
<th>COMMAND 1</th>
<th>COMMAND 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>EX Step Delay</td>
<td>EX Step Delay</td>
</tr>
<tr>
<td>EX Step Delay</td>
<td>EX Step Delay</td>
</tr>
<tr>
<td>EX Step Delay</td>
<td>EX Step Delay</td>
</tr>
<tr>
<td>EX Step Delay</td>
<td>EX Step Delay</td>
</tr>
</tbody>
</table>

**Note:** 'EX' indicates execution of the PTG Step command.
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

<table>
<thead>
<tr>
<th>Trigger Source and Input to PTG module</th>
<th>COMMAND</th>
<th>(Non-trigger input command)</th>
</tr>
</thead>
<tbody>
<tr>
<td>EX Step Delay</td>
<td>EX Step Delay</td>
<td>EX Step Delay</td>
</tr>
</tbody>
</table>

Valid level but not during the command execution. Therefore, not recognized.

Valid level during the command execution. Therefore, command terminates (with Step delay).

Note: ‘EX’ indicates execution of the PTG Step command.

---

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

Command repeats until valid trigger detected

PTGCTRL 0x1011

EX EX

Cleared at the end of instruction completion

PTGSWT

Input to PTG

Note: ‘EX’ indicates execution of the PTG Step command.
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](image)

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 OPERATING 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).
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.

Note 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 PTG0 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

```c
#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(ALTI2C1_ON & ALTI2C2_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.
    PLLFBF = 58;                    /* M  = 30 */
    CLKDIVbits.PLLPOST = 0;         /* N1 = 2 */
    CLKDIVbits.PLLPRE = 0;          /* N2 = 2 */
    OSCON = 0;

    // Initiate Clock Switch to Primary
    // Oscillator with PLL (NOSC= 0x3)
    __builtin_write_OSCONH(0x03);
    __builtin_write_OSCONL(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)

```c
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
    OC1ICON1 = 0x0; // Clear all control bits
    OC1ICON2 = 0x0; // Clear all control bits
    OC1ICON1bits.OCTSEL = 0x7; // Select peripheral clock as clock source
    OC1ICON2bits.SYNCSEL = 0xC; // Select Timer2 as sync source
    OC1ICON1bits.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
    OC2ICON1 = 0x0; // Clear all control bits
    OC2ICON2 = 0x0; // Clear all control bits
    OC2ICON1bits.OCTSEL = 0x7; // Select peripheral clock as clock source
    OC2ICON2bits.SYNCSEL = 0xA; // Select PTG as sync source
    OC2ICON1bits.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**

- **Trigger Delay = 5 \(\mu\)s**
- **Generate Interrupt 1**
- **Delay = 1 \(\mu\)s**
- **Generate Interrupt 4**
- **Trigger Output (repeat 25x)**
- **Trigger Delay = 6 \(\mu\)s**
- **Trigger Delay = 5 \(\mu\)s**
- **50 \(\mu\)s**
- **Generate Interrupt 1**
- **Trigger Delay = 6 \(\mu\)s**
- **Generate Interrupt 4**
- **Trigger Output (repeat 25x)**
- **Delay = 1 \(\mu\)s**

**Note:** The trigger delay value is modified to make the subsequent sample triggers to be shifted in time, thereby enabling the interleaving of the samples.
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 μs.
8. The second trigger delay is 6 μs.
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 μs x 14 clocks/μs)
• PTGT1LIM = 0x000B ([1 μs x 14 clocks/μs] - three Step clocks)
• PTGC0LIM = 0x0018 (total of 25 inner loop iterations)
• PTGC1LIM = 0x0001 (total of two outer loop iterations)
• PTGHOLD = 0x0046 (5 μs x 14 clocks/μs)
• PTGADJ = 0x000E (1 μs x 14 clocks/μs)
• PTGSDLIM = 0x0000 (no Step delay)
• PTGBTM = 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 = PTG JMP | 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/μs)
- PTGT1LIM = 0x36B0 (1 ms x 14 clocks/μs)
- 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)
- 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

```c
/* 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 = PTGJMPC0 | 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

<table>
<thead>
<tr>
<th>File Name</th>
<th>Bit 15</th>
<th>Bit 14</th>
<th>Bit 13</th>
<th>Bit 12</th>
<th>Bit 11</th>
<th>Bit 10</th>
<th>Bit 9</th>
<th>Bit 8</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
<th>All Resets</th>
</tr>
</thead>
<tbody>
<tr>
<td>PTGCST</td>
<td></td>
<td></td>
<td>PTGEN</td>
<td>PTGSIDL</td>
<td>PTGTOGL</td>
<td>PTGSWT</td>
<td>PTGSSEN</td>
<td>PTGIVIS</td>
<td>PTGSTRT</td>
<td>PTGWDTO</td>
<td>PTGSTRT</td>
<td>PTGWDTO</td>
<td>PTGTOGL</td>
<td>PTGSIDL</td>
<td>PTGEN</td>
<td>PTGCST 0000</td>
<td></td>
</tr>
<tr>
<td>PTGCON</td>
<td></td>
<td>PTGCLK&lt;2:0&gt;</td>
<td></td>
<td>PTGDIV&lt;4:0&gt;</td>
<td></td>
<td></td>
<td>PTGPWD&lt;3:0&gt;</td>
<td></td>
<td></td>
<td>PTGWDT&lt;2:0&gt;</td>
<td></td>
<td></td>
<td>PTGCON 0000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PTGBTE</td>
<td>ADCTS4</td>
<td>ADCTS3</td>
<td>ADCTS2</td>
<td>ADCTS1</td>
<td>IC4TSS</td>
<td>IC3TSS</td>
<td>IC2TSS</td>
<td>IC1TSS</td>
<td>OC4CS</td>
<td>OC3CS</td>
<td>OC2CS</td>
<td>OC1CS</td>
<td>OC4TSS</td>
<td>OC3TSS</td>
<td>OC2TSS</td>
<td>OC1TSS</td>
<td>PTGBTE 0000</td>
</tr>
<tr>
<td>PTGTOLim</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>PTGTOLim&lt;15:0&gt;</td>
</tr>
<tr>
<td>PTGT1Lim</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>PTGT1Lim&lt;15:0&gt;</td>
</tr>
<tr>
<td>PTGSDLim</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>PTGSDLim&lt;15:0&gt;</td>
</tr>
<tr>
<td>PTGC0Lim</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>PTGC0Lim&lt;15:0&gt;</td>
</tr>
<tr>
<td>PTGC1Lim</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>PTGC1Lim&lt;15:0&gt;</td>
</tr>
<tr>
<td>PTGHold</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>PTGHold&lt;15:0&gt;</td>
</tr>
<tr>
<td>PTGAdj</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>PTGAdj&lt;15:0&gt;</td>
</tr>
<tr>
<td>PTGL0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>PTGL0&lt;15:0&gt;</td>
</tr>
<tr>
<td>PTGQptr</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>PTGQptr&lt;4:0&gt;</td>
</tr>
<tr>
<td>PTGQueEn</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>PTGQueEn xxxx</td>
</tr>
</tbody>
</table>

**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 \texttt{PTG\_QUEUE\_SIZE} = 16 (equivalent to 32 Step commands).
**Note 2:** An 'x' in the bit name refers to the odd-numbered step queue pointers.
**Note 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:

<table>
<thead>
<tr>
<th>Title</th>
<th>Application Note #</th>
</tr>
</thead>
<tbody>
<tr>
<td>No application notes at this point of time.</td>
<td>N/A</td>
</tr>
</tbody>
</table>

**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.
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(r) logo, rPIC 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, Omniscient 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.

ISO/TS 16949:2009

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(r) MCUs and dsPIC(r) DSCs, KlixLoc(r) 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.
# 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](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: 774-447-0087
Fax: 774-447-0088

**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: 317-773-9323
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: 86-352-2388138
Fax: 86-352-2388130

**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 - Drunen**
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

08/02/11