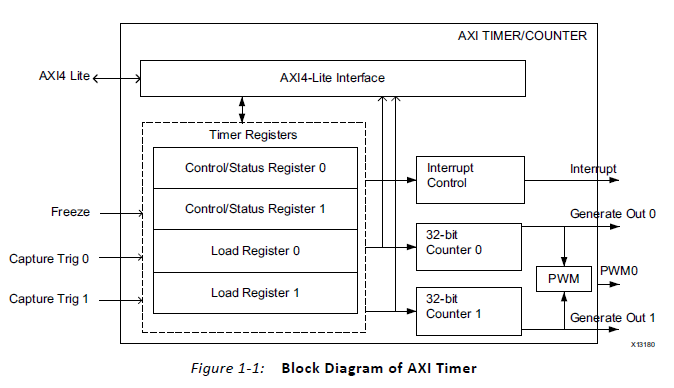
# Using Timer with MicroBlaze

## Overview

AXI timer is a 32 or 64-bit timer module. It has two programmable interval timers with interrupt capability, one pulse width modulation output(PWM), and freeze input for halting counters during software debug.

The AXI Timer is organized as two identical timer modules. Each timer module has an associated load register that is used to hold either the initial value for the counter for event generation or a capture value, depending on the mode of the timer.



The 32-bit counter modules can be configuring for up/down counts and can be loaded with a value from the load register.

The PWM block generates a pulse signal, PWM0, with a specified frequency and duty factor. It uses Timer 0 for PWM0 period, and Timer 1 for PWM0 output width.

### Modes of Operation

There are 4 modes of operations of the timer module. They are Generate mode, capture mode PWM mode and cascade mode. I used only the generate mode.

Generate Mode: On the startup generate value in the load register is loaded into the counter. Timer/Counter starts ticking when the module is enabled. When the ARHT bit (Auto Reload/Hold) is set to 1 and the counter rolls over from all 1s to all 0s when counting, or conversely from all 0s to all 1s when counting down, the generate value in the load register is automatically reloaded into the counter and the counter continues to count. When the counter is set to count down, TIMING\_INTERVAL = (TLRx + 2) \* AXI\_CLOCK\_PERIOD • When the counter is set to count, TIMING\_INTERVAL = (MAX\_COUNT - TLRx + 2) \* AXI\_CLOCK\_PERIOD where MAX\_COUNT is the maximum count value of the counter, such as 0xFFFFFFFF for a 32-bit counter.

Hardware Part

To add an AXI timer IP to the project, click on the “Add IP” button and select “AXI Timer” then click on “Run Connection Automation”. After that only AXI port with clock and reset will be connected. Interrupt connection must be done manually to “Concat” IP which is connected to the “Interrupt Controller”. “freeze” pin can be connected to “MB\_Halted” pin of the MicroBlaze. Before that “Enable Discrete Ports” option of the MicroBlaze must be selected, otherwise MB\_Halted pin cannot be used.

Software Part

|  |
| --- |
| **#include** "xparameters.h"  **#include** "xtmrctr.h"  **#include** "xil\_exception.h"  **#include** "xintc.h"  **#include** <stdio.h>  **#define** TMRCTR\_DEVICE\_ID XPAR\_TMRCTR\_0\_DEVICE\_ID  **#define** TMRCTR\_INTERRUPT\_ID XPAR\_INTC\_0\_TMRCTR\_0\_VEC\_ID  **#define** INTC\_DEVICE\_ID XPAR\_INTC\_0\_DEVICE\_ID  **#define** TIMER\_CNTR\_0 0  **#define** RESET\_VALUE 0xF0000000  XIntc InterruptController; /\* The instance of the Interrupt Controller \*/  XTmrCtr TimerCounterInst; /\* The instance of the Timer Counter \*/  // Function Prototypes dor Interrupt handler  **void** **TimerCounterHandler**(**void** \*CallBackRef, u8 TmrCtrNumber);  **int** **main**(){  u8 TmrCtrNumber = TIMER\_CNTR\_0;  XTmrCtr\_Initialize(&TimerCounterInst, TMRCTR\_DEVICE\_ID);  XIntc\_Initialize(&InterruptController, INTC\_DEVICE\_ID);  XIntc\_Connect(&InterruptController, TMRCTR\_INTERRUPT\_ID, (XInterruptHandler)XTmrCtr\_InterruptHandler, (**void** \*)&TimerCounterInst);  XIntc\_Start(&InterruptController, XIN\_REAL\_MODE);  XIntc\_Enable(&InterruptController, TMRCTR\_INTERRUPT\_ID);  Xil\_ExceptionInit();  Xil\_ExceptionRegisterHandler(XIL\_EXCEPTION\_ID\_INT, (Xil\_ExceptionHandler) XIntc\_InterruptHandler,&InterruptController);  Xil\_ExceptionEnable();  XTmrCtr\_SetHandler(&TimerCounterInst, TimerCounterHandler, &TimerCounterInst);  XTmrCtr\_SetOptions(&TimerCounterInst, TmrCtrNumber,XTC\_INT\_MODE\_OPTION | XTC\_AUTO\_RELOAD\_OPTION);  XTmrCtr\_SetResetValue(&TimerCounterInst, TmrCtrNumber, RESET\_VALUE);  XTmrCtr\_Start(&TimerCounterInst, TmrCtrNumber);  **while** (1) {  }  }  **void** **TimerCounterHandler**(**void** \*CallBackRef, u8 TmrCtrNumber)  {  /\*  \* USER INTERRUPT CODES  \*/  } |