BÀI 7. GIAO TIẾP BỘ TIMER VÀ XỬ LÝ NGẮT

1. Mục tiêu

Thông qua bài thực hành này, sinh viên sẽ biết:

- Cấu trúc bộ Timer của Intel (các thanh ghi, chức năng…).
- Cách xây dựng hệ thống phần cứng bằng Platform Designer để giao tiếp với bộ Timer.
- Cách xây dựng chương trình C trên công cụ Nios II – EDS để giao tiếp với bộ Timer, và sử dụng ngắt của bộ Timer.

2. Phần lý thuyết

2.1. Lý thuyết về bộ timer

2.1.1. Giới thiệu

Mô đun Timer của Altera có những tính năng cơ bản sau:

- Bộ đếm 32bit, hỗ trợ điều khiển Start, Stop.
- Hỗ trợ cấu hình chu kì đếm, hỗ trợ phát sinh ngắt khi kết thúc chu kì đếm.

2.1.2. Thanh ghi của Timer

Mô đun Timer bao gồm 6 thanh ghi cở bản như hình bên dưới.

Bảng 1. Các thanh ghi của mô đun Timer.

Offset	Name	R/W	Description of Bits						
			15	...	4	3	2	1	0
0	status	RW	(1)					RUN	TO
1	control	RW	(1)			STOP	START	CONT	ITO
2	periodl	RW	Timeout Period – 1 (bits [15:0])						
3	periodh	RW	Timeout Period – 1 (bits [31:16])						
4	snapl	RW	Counter Snapshot (bits [15:0])						
5	snaph	RW	Counter Snapshot (bits [31:16])						
Notes : 1. Reserved. Read values are undefined. Write zero.									

Dù mỗi thanh ghi của bộ Timer là 16 bit, địa chỉ của tất cả các thanh ghi này được sắp hàng theo 32bit. Nghĩa là nếu địa chỉ của thanh ghi "status" là BASE, thì địa chỉ của thanh ghi control là BASE + 4.

a. Thanh ghi Status

Bảng 2. Các bit của thanh ghi Status.

Bit	Name	R/W/C	Description
0	TO	R/WC	The TO (timeout) bit is set to 1 when the internal counter reaches zero. Once set by a timeout event, the TO bit stays set until explicitly cleared by a master peripheral. Write 0 or 1 to the status register to clear the TO bit.
1	RUN	R	The RUN bit reads as 1 when the internal counter is running; otherwise this bit reads as 0. The RUN bit is not changed by a write operation to the status register.

b. Thanh ghi Control

Bảng 3. Các bit của thanh ghi Control.

Bit	Name	R/W/C	Description
0	ITO	RW	If the ITO bit is 1, the interval timer core generates an IRQ when the status register's TO bit is 1. When the ITO bit is 0, the timer does not generate IRQs.
1	CONT	RW	The CONT (continuous) bit determines how the internal counter behaves when it reaches zero. If the CONT bit is 1, the counter runs continuously until it is stopped by the STOP bit. If CONT is 0, the counter stops after it reaches zero. When the counter reaches zero, it reloads with the value stored in the period registers, regardless of the CONT bit.
2	START (1)	W	Writing a 1 to the START bit starts the internal counter running (counting down). The START bit is an event bit that enables the counter when a write operation is performed. If the timer is stopped, writing a 1 to the START bit causes the timer to restart counting from the number currently stored in its counter. If the timer is already running, writing a 1 to START has no effect. Writing 0 to the START bit has no effect.
3	STOP (1)	W	Writing a 1 to the STOP bit stops the internal counter. The STOP bit is an event bit that causes the counter to stop when a write operation is performed. If the timer is already stopped, writing a 1 to STOP has no effect. Writing a 0 to the stop bit has no effect. If the timer hardware is configured with **Start/Stop control bits** off, writing the STOP bit has no effect.
Notes : 1. Writing 1 to both START and STOP bits simultaneously produces an undefined result.			

c. Thanh ghi periodh và periodl

Hai thanh ghi này quy định chu kì đếm của Timer. "periodh" chứa 16 bit cao của chu kì đếm và "periodl" chứa 16 bit thấp của chu kì đếm.

d. Thanh ghi snaph và snapl

Hai thanh ghi này quy định chu kì đếm của Timer. "periodh" chứa 16 bit cao của chu kì đếm và "periodl" chứa 16 bit thấp của chu kì đếm.

2.1.3. Hoạt động của bộ Timer

Khi bắt đầu hoạt động, bộ đếm của Timer sẽ lấy giá trị khởi tạo từ thanh ghi "periodh" và "periodl". Sau đó, bộ đếm sẽ bắt đầu đếm xuống tại mỗi chu kì xung clock.

Khi giá trị của bộ đếm bằng 0 thì:

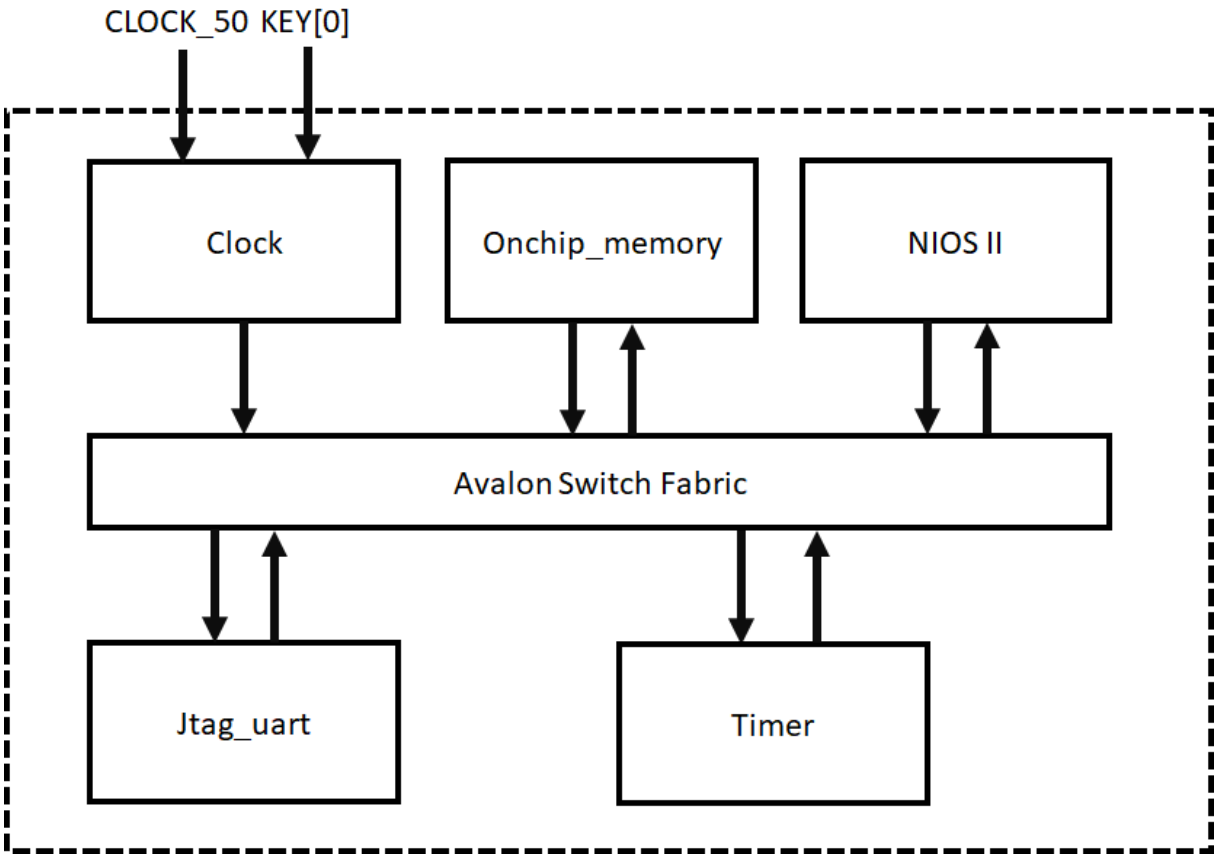
- Bộ đếm sẽ được khởi tạo lại giá trị ban đầu từ thanh ghi “periodh” và “periodl”.
- Bit “TO” của thanh ghi “status” sẽ bằng 1.
- Ngắt sẽ được sinh ra nếu bit “ITO” của thanh ghi “control” bằng 1.

Trong hàm xử lý ngắt Timer, xóa ngắt bằng cách xóa bit “TO” của thanh ghi “status”.

3. Phần thực hành

3.1. Tổng quan hệ thống

Hệ thống được thiết kế như hình 4 bên dưới:



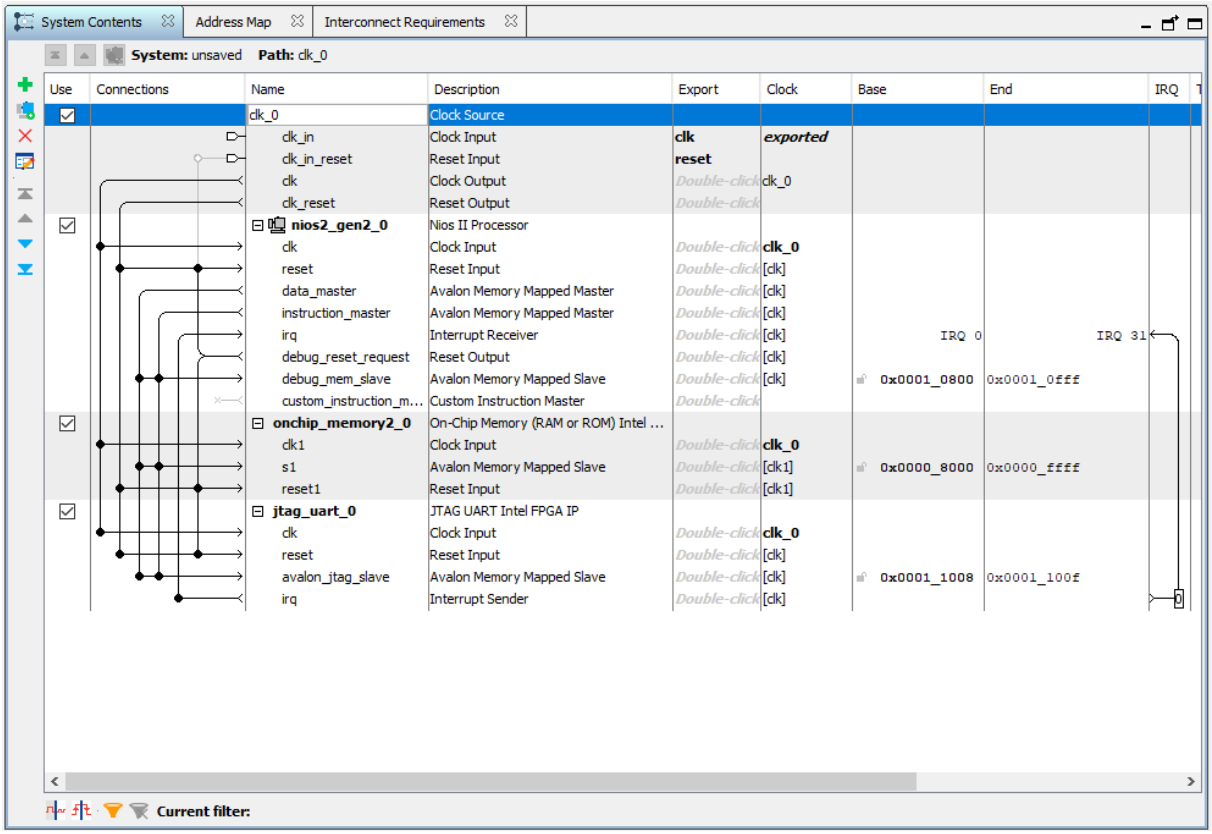
Hình 1. Tổng quan hệ thống.

3.2. Tạo project trên Quartus Prime

- Tạo project Quartus tên là “**Bai7**”. Lưu ý đường dẫn thư mục project không được có khoảng trắng. Chọn board DE2-115, chọn Family là **Cyclone IV E**, device là **EP4CE115F29C7**.

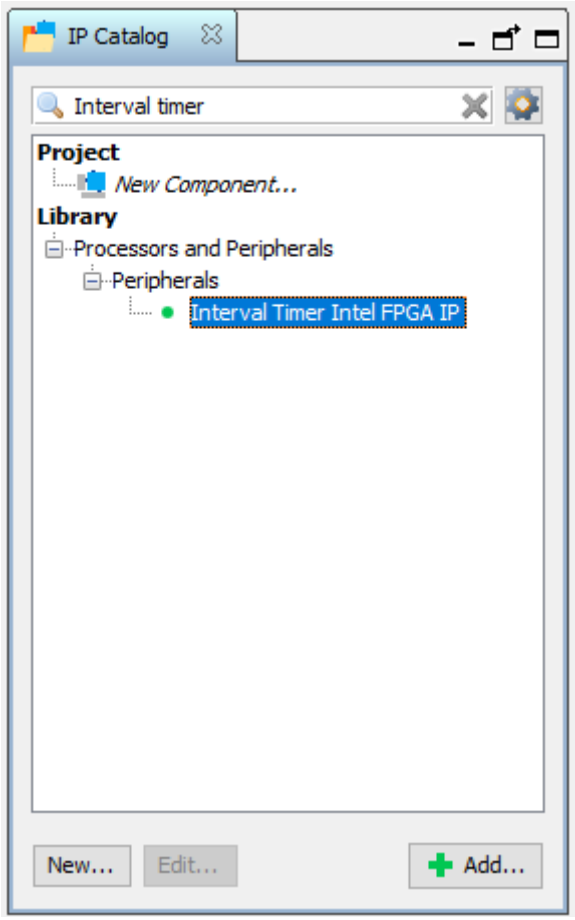
3.3. Xây dựng phần cứng trên Platform Designer

- Xây dựng hệ thống phần cứng như hình 2 bên dưới.



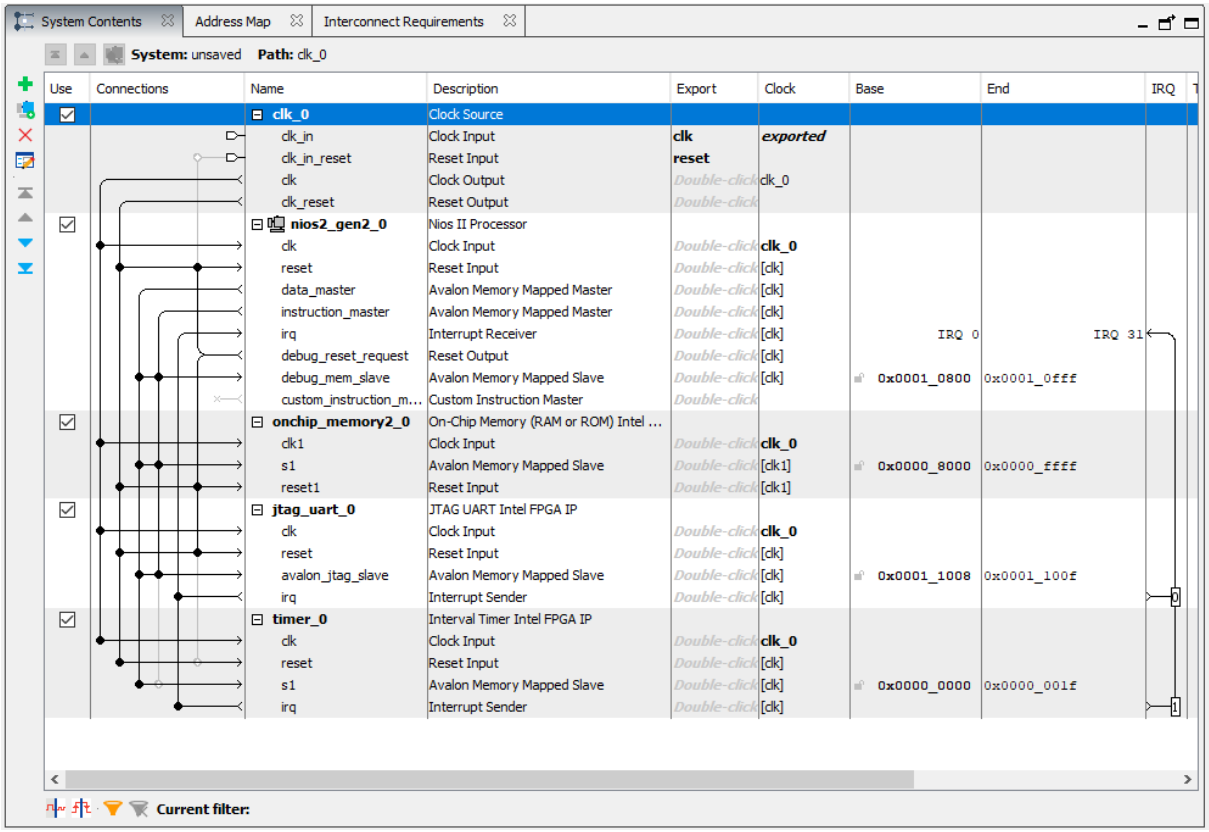
Hình 2. Hệ thống ban đầu.

- Thêm mô đun Interval Timer bằng cách gõ Interval Timer vào ô tìm kiếm, và nhấn đúp chuột như hình 3.



Hình 3. Mô đun Interval Timer trong thư viện.

- Giữ nguyên cấu hình, kết nối mô đun Interval Timer vào hệ thống như hình 4.



Hình 4. Hệ thống hoàn chỉnh.

- Gán lại địa chỉ: **System → Assign Base Addresses**.
- Lưu hệ thống dưới tên là **system** và **generate** hệ thống.

3.4. Tích hợp hệ thống

Thêm file **system.qip.**

- Tạo file top-level là **Bai7.v** được mô tả như đoạn code bên dưới để tổng hợp hệ thống.

```
module Bai7(
    input               CLOCK_50,
    input       [0:0]   KEY
);
    system Nios_system(
        .clk_clk        (CLOCK_50),
        .reset_reset_n  (KEY[0])
    );
endmodule
```

- Gán chân cho thiết bị: **Assignments → Import Assignments.**

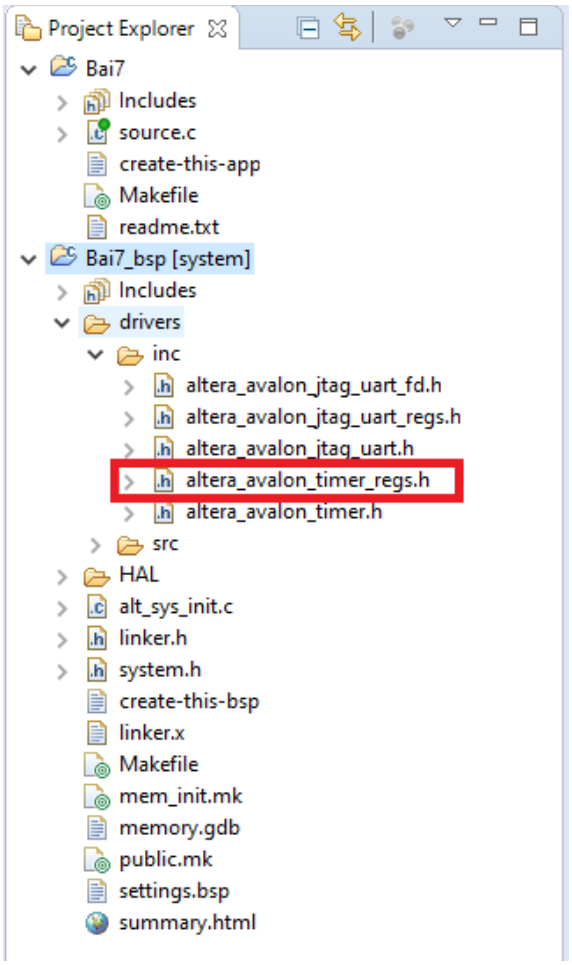
- Tiến hành biên dịch project và cuối cùng là nạp xuống board.

3.5. Xây dựng phần mềm

- Tạo và đặt tên project là "**Bai7**".
- Thêm file "**source.c**" vào project "**Bai7". File "source.c**" có nội dung như đoạn code bên dưới.

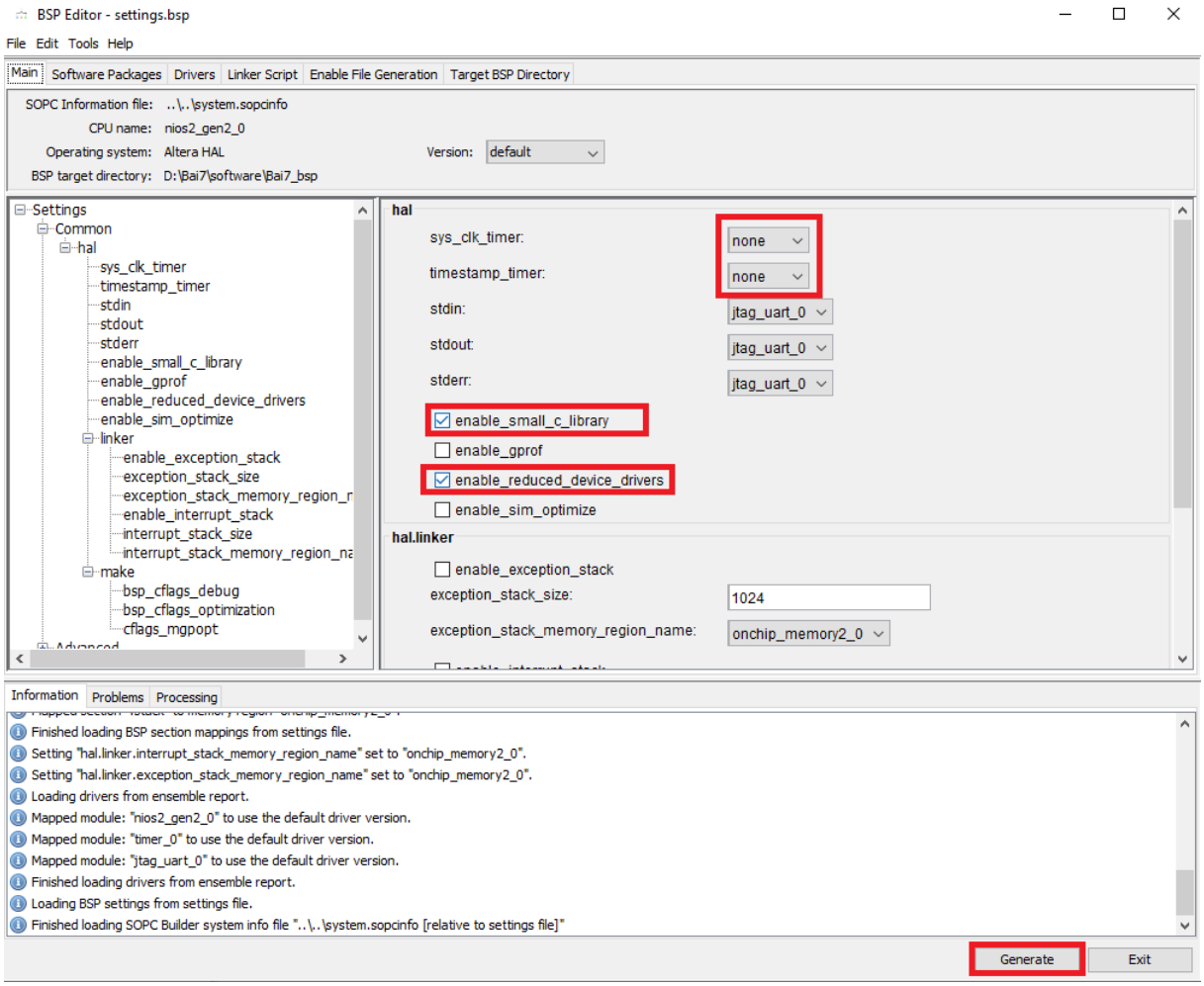
```
#include <stdio.h>
#include "io.h"
#include "system.h"
#include "altera_avalon_timer_regs.h"
#include "sys/alt_irq.h"

unsigned int counter = 0;

void timer_Init(){
	unsigned int period = 0;
	// Stop Timer
	IOWR_ALTERA_AVALON_TIMER_CONTROL(TIMER_0_BASE,
ALTERA_AVALON_TIMER_CONTROL_STOP_MSK);
	//Configure period
	period = 50000000 - 1;
	IOWR_ALTERA_AVALON_TIMER_PERIODL(TIMER_0_BASE, period);
	IOWR_ALTERA_AVALON_TIMER_PERIODH(TIMER_0_BASE, (period >> 16));

	IOWR_ALTERA_AVALON_TIMER_CONTROL(TIMER_0_BASE,
	ALTERA_AVALON_TIMER_CONTROL_CONT_MSK | // Continue counting mode

	ALTERA_AVALON_TIMER_CONTROL_ITO_MSK  | // Interrupt enable

	ALTERA_AVALON_TIMER_CONTROL_START_MSK);// Start Timer
}

void Timer_IRQ_Handler(void* isr_context){
	counter ++;
	printf("%d seconds\n", counter);
	// Clear Timer interrupt bit
	IOWR_ALTERA_AVALON_TIMER_STATUS(TIMER_0_BASE,
ALTERA_AVALON_TIMER_STATUS_TO_MSK);
}

void main(){
	timer_Init();
	alt_ic_isr_register(0,  TIMER_0_IRQ,  Timer_IRQ_Handler,  (void*)0,
(void*)0);
	while(1);
}
```

- Thư viện phục vụ cho mô đun Interval Timer ở vị trí như hình 5 bên dưới.



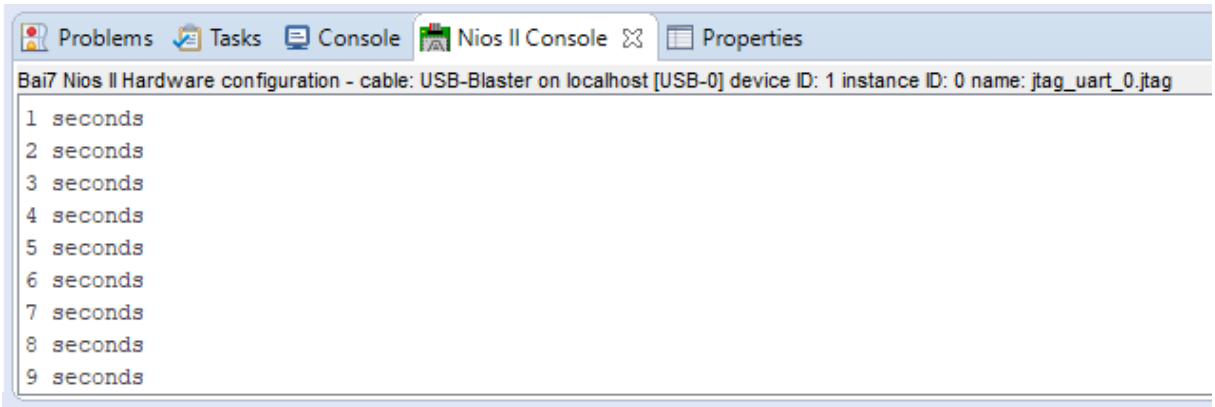
Hình 5. Vị trí của thư viện phục vụ mô đun Timer.

- Cấu hình **BSP** như hình 6 bên dưới.



Hình 6. Cấu hình BSP.

- Build project và dowload phần mềm xuống board. Xem kết quả trên console.



Hình 7. Kết quả trên console.

BÀI TẬP CHUẨN BỊ Ở NHÀ

Bài 1. Giải thích các bit trong các thanh ghi của mô đun Interval Timer.

Bài 2. Giải thích code C trong phần hướng dẫn thực hành.

BÁO CÁO THỰC HÀNH

Bài 1. Xây dựng mô đun hệ thống SoC dùng Timer, SW, HEX tạo thành đồng hồ.

Cách tùy chỉnh đồng hồ cho sinh viên tự quy định.

TÀI LIỆU THAM KHẢO

[1] DE2_115_User_Manual.

[2] Embedded Peripherals IP User Guide.

[3] Nios II Processor Reference.

[4] Avalon Interface Specification.