

**ĐẠI HỌC QUỐC GIA  
TRƯỜNG ĐH BÁCH KHOA TP.HCM**



# **BÁO CÁO THỰC TẬP TỐT NGHIỆP**

**GVHD: THẦY TRẦN VĂN HOÀNG**

**NHÓM 7**

**SVTH:**

1. TRẦN VĂN TRỌNG
2. VÕ VĂN LONG
3. NGUYỄN TẤN TÍN

## MỤC LỤC

CHƯƠNG 1: LAB 1 HARDWARE AND SOFTWARE SET UP.....

CHƯƠNG 2: LAB 2 CODE COMPOSER STUDIO.....

CHƯƠNG 3 : LAB 3 INITIALIZATION AND GPIO.....

CHƯƠNG 4: LAB 4 INTERRUPTS AND THE GP TIMER.....

CHƯƠNG 5: LAB 5 ADC12.....

CHƯƠNG 6: LAB 6 LOW POWER MODES.....

CHƯƠNG 7 : LAB 7 USB.....

CHƯƠNG 8 : LAB 8 BỘ NHỚ VÀ MPU.....

CHƯƠNG 9 : LAB 9 FPU.....

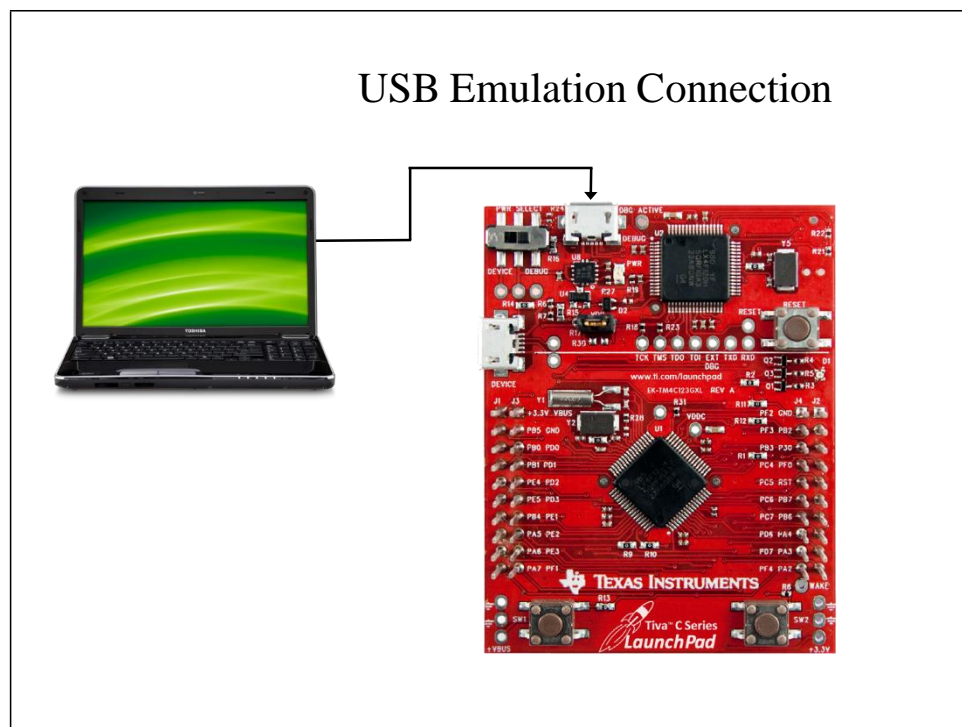
CHƯƠNG 10 : LAB 12 UART.....

# CHƯƠNG 1

## Lab1: Hardware and Software Set Up

### 1.Mục tiêu thí nghiệm.

+ Mục tiêu của bài luyện tập lab này là tải về và cài đặt Code Composer Studio, cũng như tải các tài liệu hỗ trợ và phần mềm khác nhau được sử dụng với bài tập này. Sau đó chúng ta sẽ xem xét các nội dung của bộ kit và xác minh hoạt động với các chương trình quickstart bản demo được nạp sẵn. Các công cụ phát triển sẽ được sử dụng trong suốt các bài lab còn lại trong các bài tập này.



### 2.Tiến hành thí nghiệm.

#### Hardware

Yêu cầu phần cứng:

- Laptop 32 hoặc 64 bit Window XP, Window 7 hoặc 8 với 2G hoặc hơn bộ nhớ ổ cứng trống. 1G ram tối thiểu hoặc hơn. Laptop Apple có thể chạy

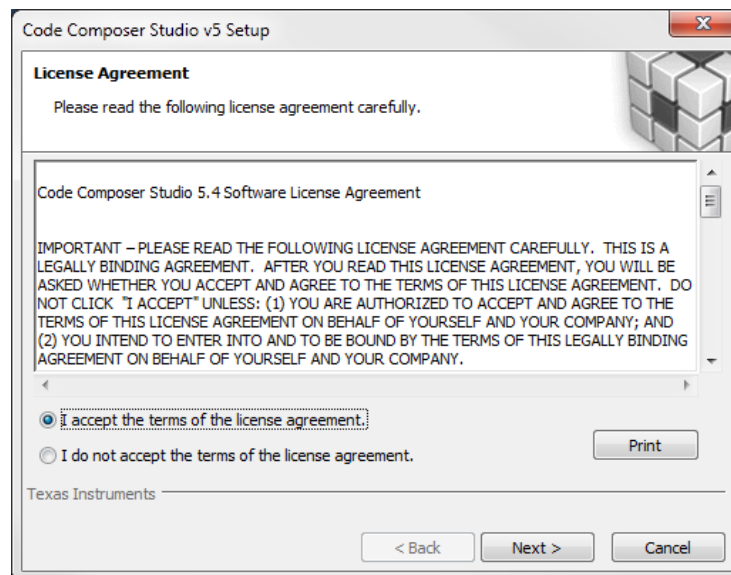
với bất kỳ hệ điều hành nào. Laptop chạy hệ điều hành linux không được khuyến khích sử dụng.

- Có kết nối mạng là cần thiết.
- Nếu thực hiện các bài lab dùng 2 màn hình hiển thị thì sẽ giúp quá trình dễ dàng hơn. Do đó, khuyến khích các bạn mang theo labtop cá nhân trong các giờ thực hành.
- Nếu bạn làm bài lab ở phòng thí nghiệm. bạn sẽ nhận được một board ứng dụng. nếu không có bạn cần phải mua.
- Thực hiện các bài lab trên kit **Tiva™ TM4C123G LaunchPad**

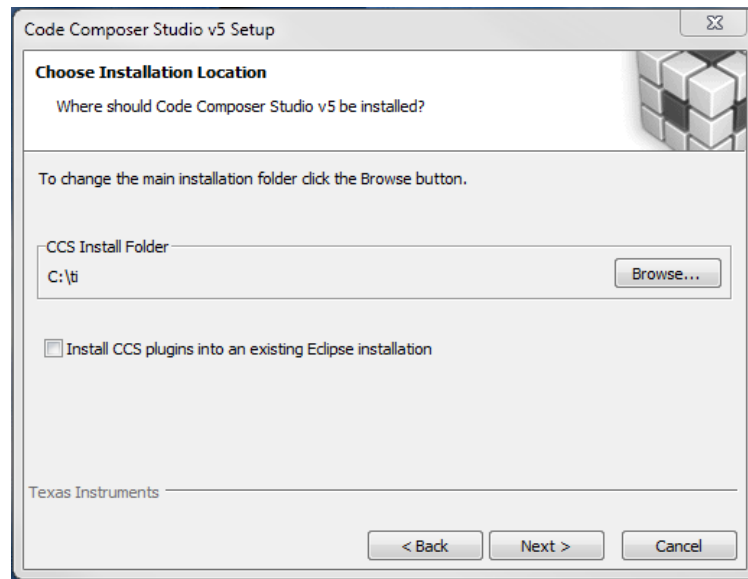
Một multi – meter.

### *Download và cài đặt Code Composer Studio*

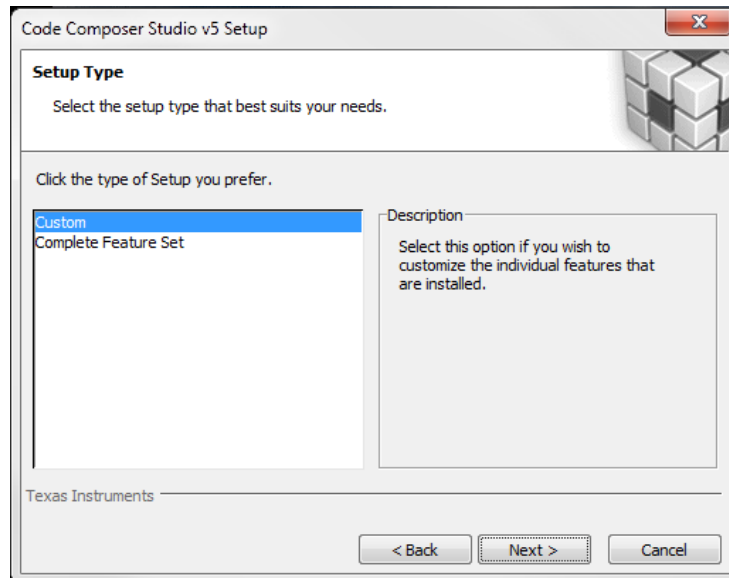
2. Download phần mềm *Code Composer Studio (CCS) 5.x web installer* từ địa chỉ [http://processors.wiki.ti.com/index.php/Download\\_CCS](http://processors.wiki.ti.com/index.php/Download_CCS) (không download bất kì bản dùng thử nào). Yêu cầu kết nối mạng cho đến khi quá trình hoàn tất. Nếu không có kết nối mạng thì có thể sử dụng offline version để cài đặt.
3. Nếu sử dụng file offline, chạy file *ccs\_setup\_5.xxxx.exe* trong folder sau khi giải nén.
4. Đồng ý Software License Agreement và click Next.



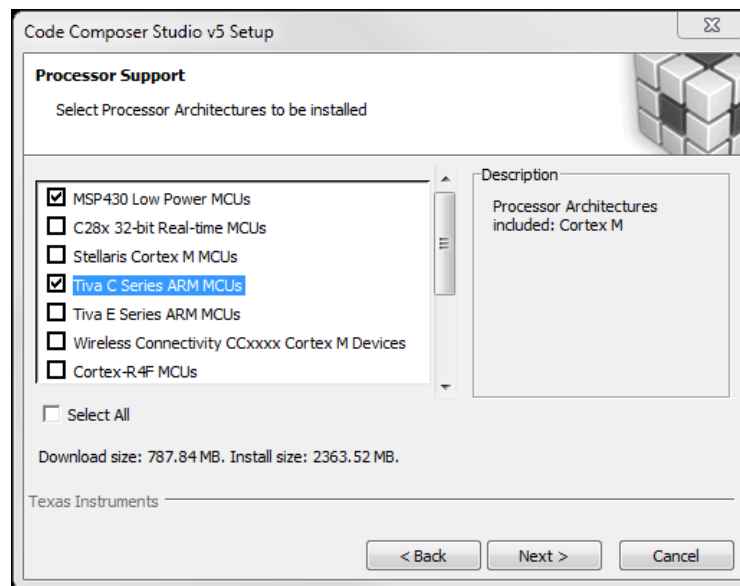
5. Ngoại trừ việc muốn cài đặt CCS ở một vị trí khác, thì tốt nhất nên chọn folder mặc định theo chương trình cài đặt và Click Next.



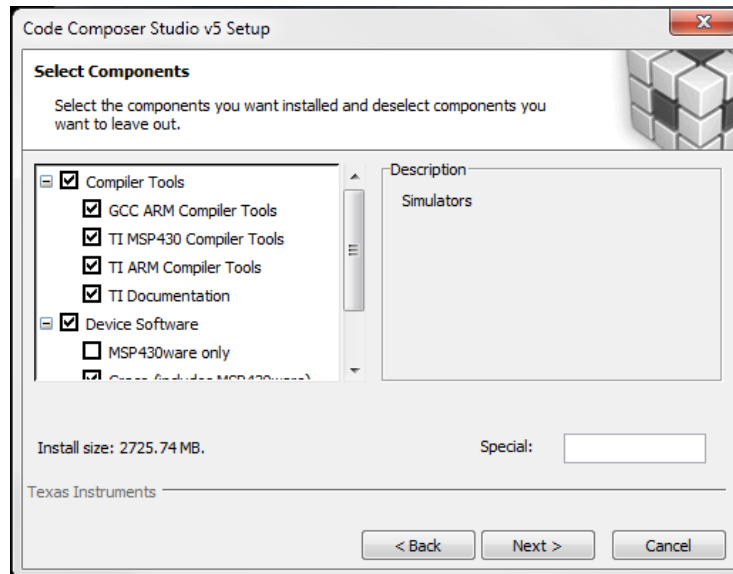
6. Chọn “Custom” cho phần Setup type và Click Next.



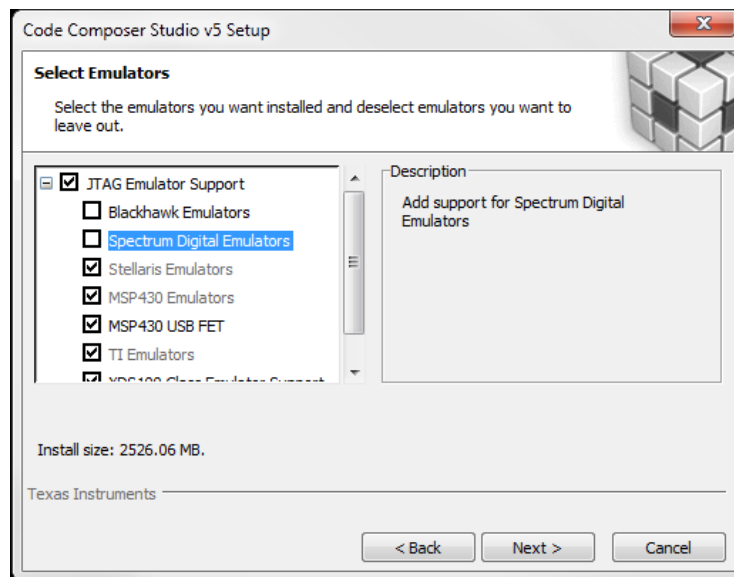
7. Trong hộp thoại tiếp theo, chọn processors mà CCS sẽ hỗ trợ. Chọn “Tiva C Series ARM MCUs” để thực hiện các bài lab. Có thể chọn thêm các kiến trúc khác, tuy nhiên sẽ mất thời gian cài đặt cũng như kích thước cài đặt sẽ lớn. Click Next.



8. Trong hộp thoại Component, giữ mặc định các lựa chọn và Click Next.



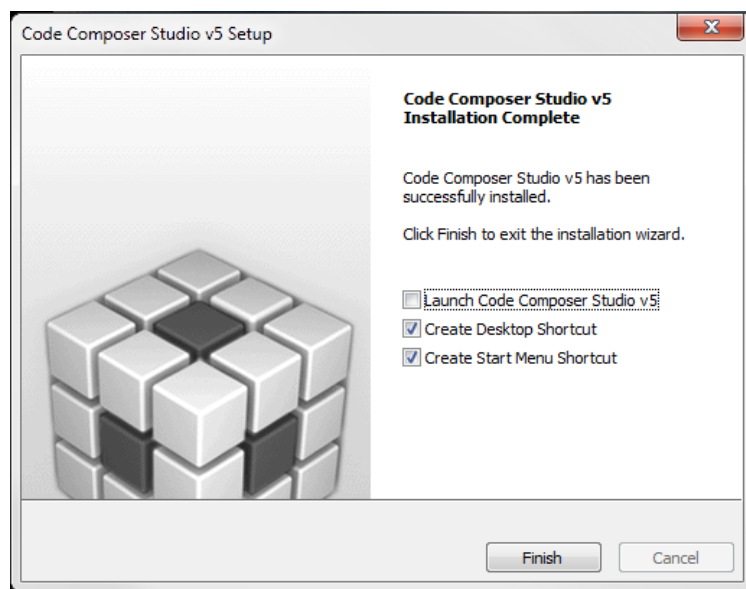
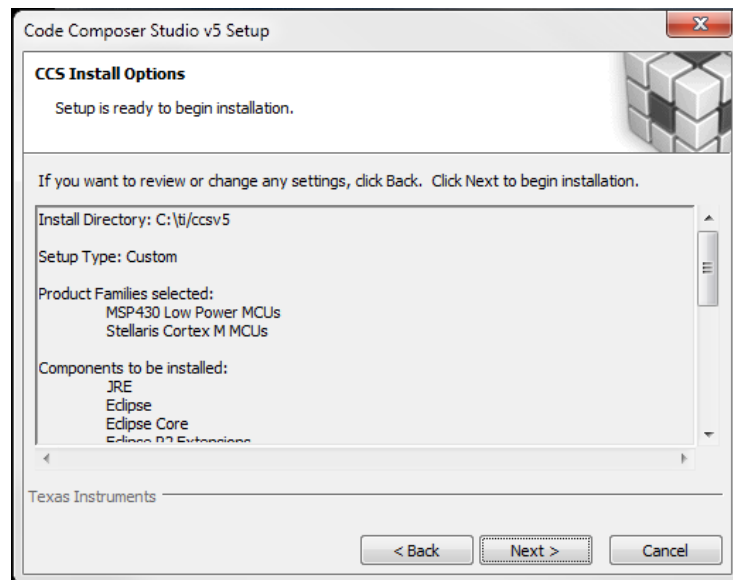
9. Trong hộp thoại Emulator, bỏ chọn Blackhawk and Spectrum Digital emulators, trừ khi có ý định sử dụng chúng.



10. Khi tới đến hộp thoại cài đặt cuối cùng, Click Next. Quá trình cài đặt sẽ bắt đầu.

Khi hoàn tất quá trình cài đặt, **không chạy CCS**.





11. có một vài công cụ mở rộng sẽ được yêu cầu cài đặt trong suốt quá trình cài đặt CCS. Click yes hoặc ok cho đến khi hoàn tất.

### ***Cài đặt TivaWare for C series***

12. Download và cài đặt version cuối cùng của TivaWare từ địa chỉ:  
<http://www.ti.com/tool/sw-tm4c> . tên file là SW-TM4C-x.x.exe.

Nếu có thể, nên cài đặt StellarisWare vào folder mặc định  
C:\TI\TivaWare\_C\_Series-x.x.

### ***Cài đặt LM Flash Programmer***

13. Download, giải nén, và cài đặt mới nhất LM Flash Programmer (LMFLASHPROGRAMMER) từ  
<http://www.ti.com/tool/lmflashprogrammer> .

### ***Download và cài đặt Workshop Lab Files***

14. Download the lab installation file từ địa chỉ Wiki site bên dưới.  
Các file lab sẽ được cài đặt trong  
C:\Tiva\_TM4C123G\_LaunchPad.. Do đó, phải chắc chắn rằng  
StellarisWare phải được cài đặt trước đó.  
<http://www.ti.com/TM4C123G-Launchpad-Workshop>

### ***Download Workshop Workbook***

15. Có thể download file Tiếng Anh hướng dẫn các bài lab này với  
nhiều bài lab hơn theo địa chỉ sau:  
[www.ti.com/StellarisLaunchPadWorkshop](http://www.ti.com/StellarisLaunchPadWorkshop)

### ***Terminal Program***

16. Nếu sử dụng Window XP, có thể sử dụng HyperTerminal.  
Window 7 không có chương trình terminal, nên phải sử dụng một  
phần mềm khác. Các câu lệnh trong các bài labs sử dụng  
HyperTerminal và PuTTY. Có thể download PuTTY từ địa chỉ  
sau:

<http://the.earth.li/~sgtatham/putty/latest/x86/putty.exe>

### ***Window-side USB Examples***

17. Download và cài đặt *StellarisWare Windows-side USB examples* từ địa chỉ:

[www.ti.com/sw-usb-win](http://www.ti.com/sw-usb-win)

### ***Download và cài đặt GIMP***

18. Chúng ta sẽ cần một công cụ thao tác đồ họa có khả năng xử lý các ảnh định dạng PNM. GIMP có thể làm điều đó. Download và cài đặt GIMP từ địa chỉ:

[www.gimp.org](http://www.gimp.org)

**L** phần cuối của tài liệu này.

**au**

**nc**

**h**

**P**

**ad**

**B**

**oa**

**rd**

**Sc**

**he**

**m**

**ati**

**c**

19. Để  
tha  
m  
kh  
ảo,  
sc  
he  
ma  
tic  
sẽ  
có  
ở

### ***Các tài liệu và trang web tham khảo hữu dụng***

20. Có rất nhiều tài liệu hữu dụng, tuy nhiên ít nhất bạn nên có các tài liệu sau.

Tìm trong C:\TI\TivaWare\_C\_Series-1.1\docs sẽ thấy:

**Peripheral Driver User's**

**Guide (SW-DRL-UGx.x.pdf)**

**USB Library User's Guide**

**(SW-USBL-UG-x.x.pdf)**

**Graphics Library User's**

**Guide (SW-GRL-UGx.x.pdf)**

**LaunchPad Firmware User's Guide (SW-EK-  
TM4C123GXL-UG-x.x.pdf )**

21. Vào địa chỉ: <http://www.ti.com/lit/gpn/tm4c123gh6pm> và

Sheet thực sự là một hướng dẫn sử dụng đầy đủ cho các device.

22. Download the ARM Optimizing C/C++ Compilers User Guide từ địa chỉ

<http://www.ti.com/lit/pdf/spnu151> (SPNU151).

*Có thể tìm thêm thông tin ở các websites sau:*

**Main page:** [www.ti.com/launchpad](http://www.ti.com/launchpad)

**Tiva C Series TM4C123G LaunchPad:**

<http://www.ti.com/tool/ek-tm4c123gxl>

**TM4C123GH6PM folder:**

<http://www.ti.com/product/tm4c123gh6pm> **BoosterPack**

**webpage:** [www.ti.com/boosterpack](http://www.ti.com/boosterpack)

**LaunchPad Wiki:** [www.ti.com/launchpadwiki](http://www.ti.com/launchpadwiki)

23. Mở hộp kit ra

Bạn sẽ tìm thấy trong đó có :

- **The TM4C123GXL LaunchPad Board**
- **USB cable (A-male to micro-B-male)**
- **README First card**
- **If you are in a live workshop, you should find a 2<sup>nd</sup> USB cable**

## *Cài đặt Board ban đầu*

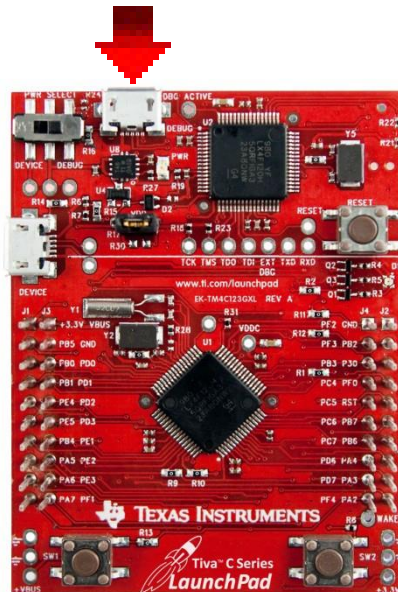
### 24. Kết nối board với máy tính và cài đặt drivers

The TM4C123GXL LaunchPad Board ICDI USB port (marked DEBUG and shown in the picture below) là cổng USB và bao gồm 3 kết nối:

**Stellaris ICDI JTAG/SWD Interface** - debugger connection

**Stellaris ICDI DFU Device** - firmware update connection

**Stellaris Virtual Serial Port** - a serial data connection

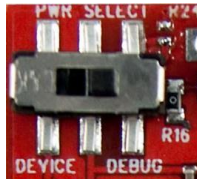


Sử dụng cab USB. Kết nối cổng debug của kit với cổng USB của laptop hoặc PC. Drivers sẽ được cài đặt tự động.

## *QuickStart Application*

LaunchPad Board đã được lập trình trước với một ứng dụng QuickStart. Chỉ cần cung cấp nguồn cho board, ứng dụng này sẽ chạy một cách tự động.

25. Phải chắc chắn rằng Power switch ở phía trên góc trái của board được gạt qua vị trí bên phải DEBUG như trong hình:



26. Phần mềm trong TM4C123GH6PM sử dụng timers như là pulse-width modulators (PWMs) để thay đổi cường độ của tất cả 3 màu trên Led đơn RGB (red, green, and blue). Bởi vậy, mắt sẽ cảm nhận được nhiều màu sắc khác nhau được tạo ra thông qua việc kết hợp các màu cơ bản.

Hai pushbuttons ở phía dưới của board được đánh nhãn **SW1** (bên trái) và **SW2** (ở bên phải). Nhấn hoặc nhấn và giữ **SW1** để di chuyển về phía phổ màu đỏ ở cuối. Nhấn hoặc nhấn và giữ **SW2** để di chuyển về phía phổ màu tím ở cuối.

Nếu không nhấn nút nào trong vòng 5 giây, phần mềm sẽ tự động quay về thay đổi màu sắc như mặc định.

27. Nhấn và giữ cả 2 nút nhấn **SW1** và **SW2** trong vòng 3 giây sẽ đi vào hibernate mode (chế độ ngủ). Trong chế độ này màu sắc cuối cùng sẽ nhấp nháy 1/2 giây sau mỗi 3 giây. Giữa các khoảng nhấp nháy, thiết bị ở chế độ ngủ VDD3ON với realtime-clock (RTC) đang chạy. Nhấn **SW2** ở bất kì lúc nào sẽ đánh thức thiết bị và quay về chương trình hiển thị màu sắc một cách tự động.

28. Ta có thể giao tiếp với board thông qua UART. UART được kết nối như là cổng nối tiếp ảo thông qua kết nối USB giả lập.

Các bước sau đây sẽ hướng dẫn làm cách nào mở kết nối với board sử dụng HyperTerminal (trong Window XP) và PuTTY (trong Window 7 hoặc 8).

29. Ta cần tìm COM port number của Stellaris virtual Serial Port trong Device Manager.

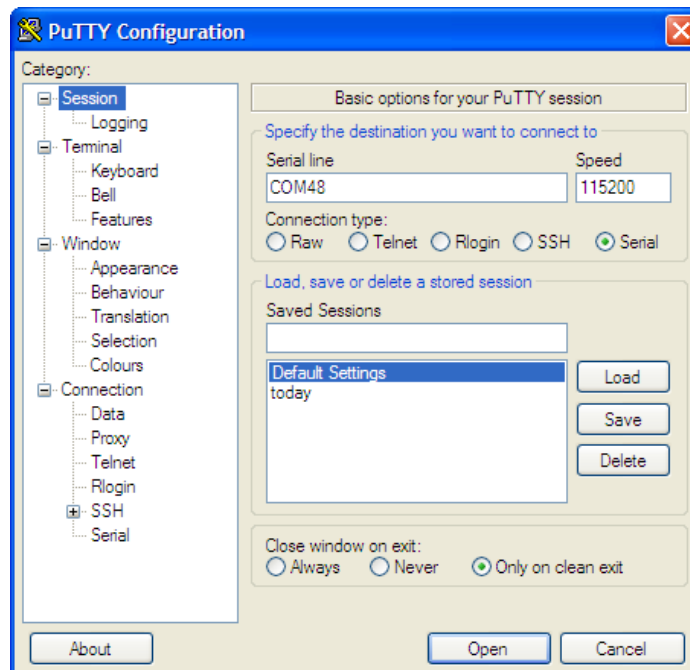
### **Window 7:**

- A. Click chuột phải ở *My Computer* và chọn *manger*.

B. Trong cửa sổ computer mangement, chọn tab device manager .

Mở *Ports heading* và viết số của Stellaris Virtual Serial Port ở đây: **COM**\_\_\_\_\_

30. Trong **Win 7 hoặc 8**, double click vào *putty.exe*. Thiết lập các cài đặt như hình dưới và sau đó click Open. COM Port number sẽ là số mà ta đã ghi chú lại trước đây.



Khi cửa sổ terminal được mở, nhấn ENTER một lần và LaunchPad board sẽ phản hồi là xác nhận giao tiếp đã mở. Bỏ qua bước 31.

31. Bạn có thể giao tiếp bằng cách gõ các dòng lệnh sau và nhấn ENTER:

**help:** sẽ tạo ra một danh sách các lệnh và



thông tin.

**hib:** sẽ đưa thiết bị vào chế độ ngủ. Nhấn SW2 để đánh thức thiết bị.

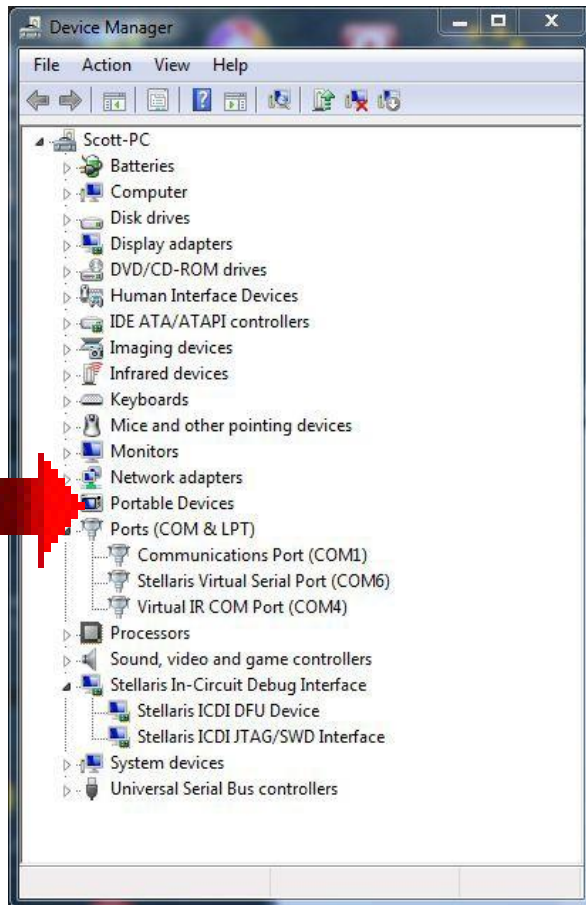
**rand:** sẽ bắt đầu một chuỗi pseudo-random màu sắc.

**intensity:** điều chỉnh độ sáng của LED từ 0 tới 100%. Với 100 sẽ làm LED hiển thị sáng nhất.

**rgb:** theo sau là một “6 hex character value” để thiết lập cường độ của tất cả 3 LEDs. Ví dụ: rgb FF0000 LED sáng màu đỏ, rgb 00FF00 LED sáng màu xanh dương và rgb 0000FF LED sáng màu xanh lá.

32. Đóng chương trình Terminal.

Hoàn thành!

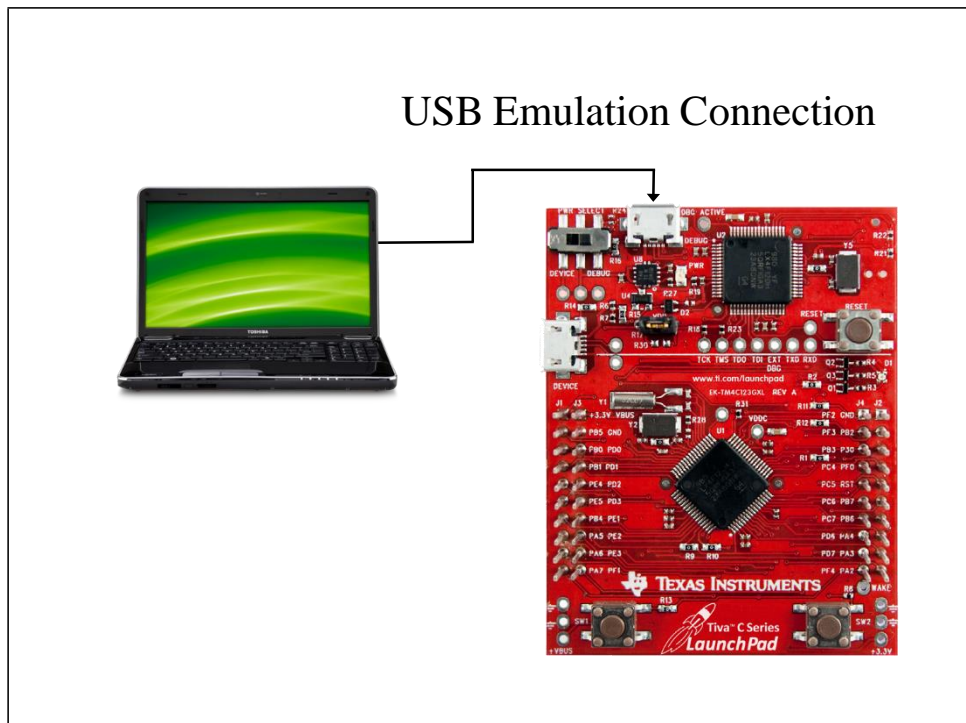


# CHƯƠNG 2

## Lab2: Code Composer Studio

### 1.Mục tiêu thí nghiệm.

+ Trong bài lab này, chúng tôi sẽ tạo ra một dự án có chứa hai tập tin nguồn, main.c và tm4c123gh6pm\_startup\_ccs.c, trong đó có chứa đoạn mã để chớp LED trên tàu LaunchPad của bạn. Mục đích của thí nghiệm này là để thực hành tạo ra các dự án và tìm hiểu cái nhìn và cảm nhận của Code Composer Studio. Trong các bài lab sau đó chúng tôi sẽ kiểm tra mã chi tiết hơn. Cho đến nay bây giờ, không phải lo lắng về các mã C chúng ta sẽ sử dụng trong bài lab này.



## **2.Tiến hành thí nghiệm.**

Thư mục chứa các bài lab

### **1. Chọn đường dẫn chứa các bài lab**

► sử dụng Windows Explorer, địa điểm chứa thư mục sau:

C:\TM4C123G\_LaunchPad\_Workshop

Trong thư mục này, bạn sẽ tìm thấy tất cả các bài lab trong các buổi thí nghiệm. nếu không nhìn thấy trong c:\drive,kiểm tra lại chắc chắn rằng bạn đã cài workshop lab hay chưa. Mở rộng \lab2folder và chúng ta sẽ thấy có hai thư mục \files và \project.

Mục \filesfolder sẽ chứa các file phụ liên quan. Mục \project sẽ chứa các file cài đặt và các file quan trọng mà bạn tạo ra trong project. nó cũng chứa các file biên dịch và file code. Bạn cũng nhìn thấy những file này trong cửa sổ project explorer và dễ dàng cắt/dán nội dung của nó khi cần thiết.

---

**chú ý:** khi tạo một project, bạn phải chọn một thư mục mặc định trong CCS workspace or hoặc một đường dẫn khác. Trong các bài lab này, trong các bài lab này chúng ta sẽ sử dụng đường dẫn đã cài là, C:\TM4C123G\_LaunchPad\_Workshop.

---

Tạo một project mới trong CCS

### **1. tạo một project mới**

► chạy CCS. khi hộp thoại “Select a workspace” xuất hiện, ► browse

to your My Documents folder:

(trong WinXP) C:\Documents and Settings\<user>\My Documents

(trong Win7 or 8)

C:\Users\<user>\MyDocuments

có thể thấy rằng, thay thế <user> bằng tên máy tính của bạn. tên và địa chỉ đường dẫn có thể thay đổi, nhưng chúng tôi đề nghị bạn hãy sử dụng **MyWorkspaceTM4C123G**. không chọn ô “*Use this as the default and do not ask again*” . Nếu lúc nào đó bạn vô tình chọn hộp này, nó có thể được thay đổi trong CCS.

► Click OK.

2. cài đặt license CCS: Nếu không có licensed Code Composer, bạn sẽ được hỏi trong những bước cài đặt tiếp theo. Khi đó, chọn “Evaluation”. Ngay khi máy tính được kết nối với LaunchPad board, Code Composer sẽ có đầy đủ chức năng, miễn phí.

3. tạo một project mới:

để tạo một project mới, ► chọn *Project* → *New CCS Project*:

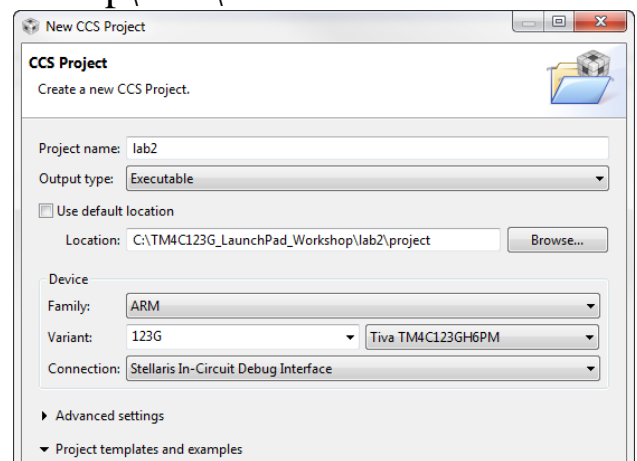
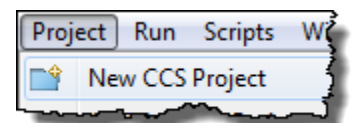
► chọn tên cho project, như *lab2*

► bỏ chọn hộp “*Use default location*” và click *Browse...* button. chọn đường dẫn sau:

C:\TM4C123G\_LaunchPad\_Workshop\lab2\project và click OK

► chọn Device family:

*ARM*, for Variant, type *123G* trong ô tiếp theo, sau đó

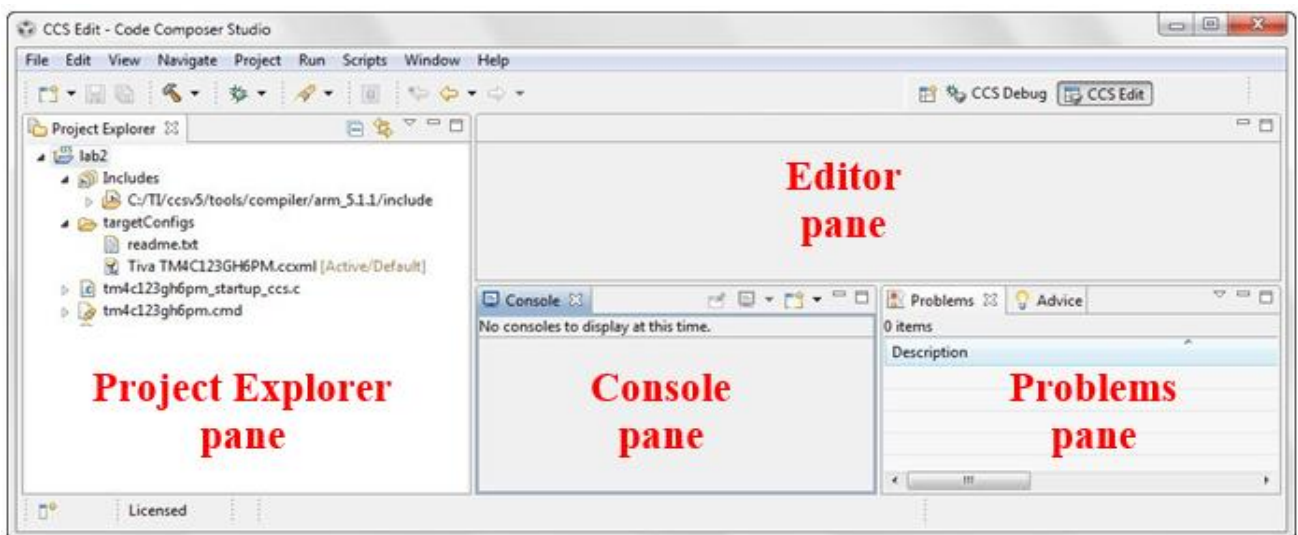


chọn *Tiva TM4C123GH6PM*  
trong ô kế bên

▶ trong ô Connection:  
chọn *Stellaris In- Circuit  
Debug Interface* . đây là giả  
lập cài sẵn trên LaunchPad  
board.


▶ ở mục Project templates  
và examples box, chọn  
*Empty Project* và sau đó  
click *Finish*.

#### 4. Xem trước hướng dẫn CCS Edit



Chú ý tên của pane hướng dẫn Code Composer ở trên.

► Ở trong Project Explorer pane trên màn hình desktop, click ký

tự  cạnh *lab2*, *Includes* and *targetConfigs* để mở rộng project. project của bạn sẽ trông giống ở trên hình.

5. Bạn có lẽ nhận ra rằng New Project wizard thêm vào một tập tin khởi động là tm4c123gh6pm\_startup\_ccs.c vào trong project một cách tự động. Chúng tôi sẽ xem xét kỹ hơn các tập tin này sau.

### **Thêm Path và xây dựng biến**

Nếu bạn nhớ lại trong bài trình bày, các path và xây dựng các biến được sử dụng cho:

- Path variable – khi bạn ADD (link) một tập tin đến project của bạn, bạn có thể chỉ rõ một “relative to” path. Trường hợp khác là *PROJECT\_LOC* có nghĩa là linked resource của bạn (like a .lib file) sẽ được liên kết đến project của bạn.
- Build variable – được sử dụng cho các nhóm như là tìm kiếm path bao gồm các tập tin liên kết với một thư viện, tức là nó được sử dụng khi bạn xây dựng project của bạn.

Các biến có thể hoặc có một *PROJECT* scope (chúng chỉ làm việc cho project này) hoặc một *WORKSPACE* scope (chúng làm việc với tất cả project trong workspace).

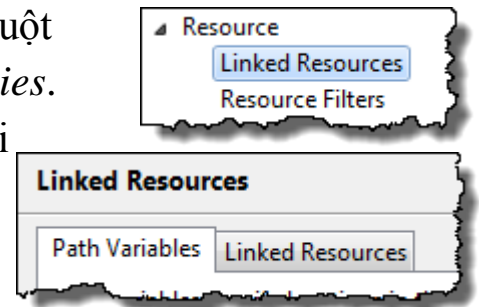
Trong bước tiếp theo, chúng ta cần add (link) một tập tin thư viện và sau đó add một đường dẫn tìm kiếm cho bao gồm các tập tin. Đầu tiên, chúng ta sẽ add những biến này MANUALLY as PROJECT variables. Sau đó, chúng tôi sẽ chỉ cho bạn một cách nhanh chóng và dễ dàng để thêm các biến vào WORKSPACE của bạn để cho bất kỳ project trong vùng làm việc của bạn có thể sử dụng các biến.

### **6. Thêm một biến đường dẫn**



Để thêm một biến đường dẫn, ► click chuột phải trên project của bạn và chọn *Properties*.

► Mở rộng danh sách *Resource* ở góc trái phía trên như hình và click vào *Linked Resources*:



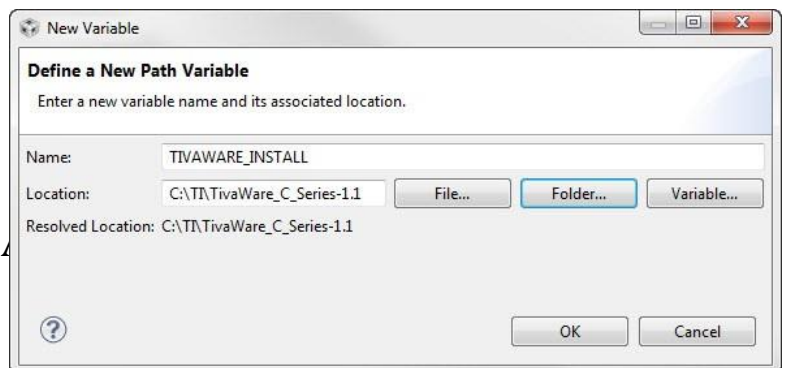
Bạn sẽ thấy hai tab cạnh nhau - *Path Variables* và *Linked Resources*:

Ở trong tab *Path Variables*, chú ý rằng *PROJECT\_LOC* được liệt kê và sẽ hiển thị như các biến đường dẫn mặc định cho các nguồn tài nguyên liên quan trong dự án của bạn.

Chúng ta muốn thêm biến *New* để xác định chính xác nơi bạn cài đặt TivaWare.

► Click *New*

► Khi *New Variable* xuất hiện, gõ *TIVAWARE\_INSTALL* cho mục *name*.



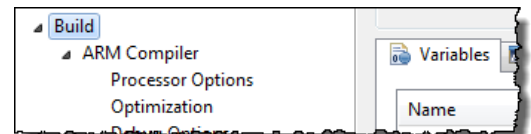
► ỏa mục  
*Location*, click  
button *Folder...* và  
điều hướng để cài  
đặt TivaWare của  
bạn. Click trên  
folder name và sau  
đó click OK.

► Click OK. Bạn sẽ thấy biến đường dẫn new được ghi vào trong danh sách Path Variables.

## 7. Thêm vào một biến xây dựng

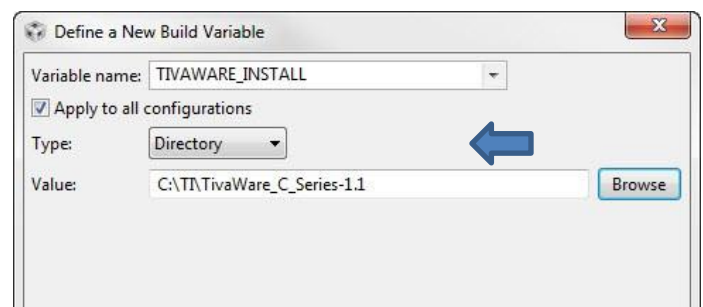
Bây giờ chúng ta hãy thêm một build variable mà chúng ta sẽ sử dụng trong đường dẫn bao gồm tìm kiếm các file INCLUDE kết hợp với các thư viện điều khiển TivaWare.

► Click vào *Build* và sau đó tab *Variables*:



► Click vào button *Add*. Khi *Define a New Build Variable* xuất hiện, chèn TIVAWARE\_INSTALL vào mục Variables name.

► Check vào ô “Apply to all configurations”



► Thay đổi Type là Directory và browse đến Tivaware installation folder của bạn.

► Click OK.

► Click OK lần nữa để lưu và đóng cửa sổ Build Properties.

### **Thêm các tập tin vào project của bạn**

Chúng tôi cần thêm main.c vào project. Chúng tôi cần thêm TivaWare driverlib.lib object library. Các tập tin C nên được sao chép vào dự án, tập tin driverlib nên được liên kết.

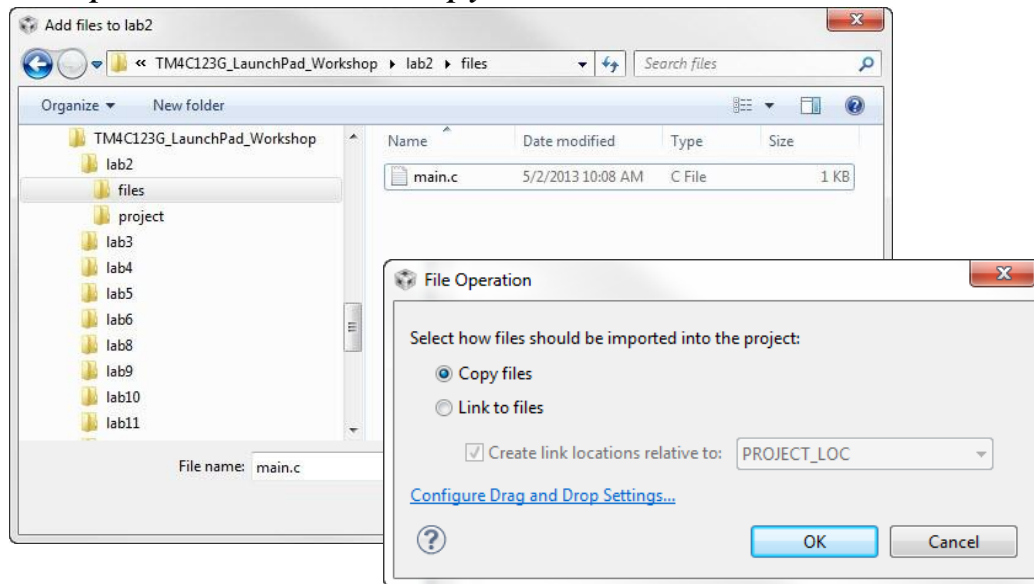
#### **8. Thêm (copy) tập tin C**

► Chọn *Project* → *Add Files...* ► Điều hướng đến thư mục:

C:\TM4C123G\_LaunchPad\_Workshop\lab2\files

Chọn main.c và click

*Open.* Sau đó chọn *Copy*



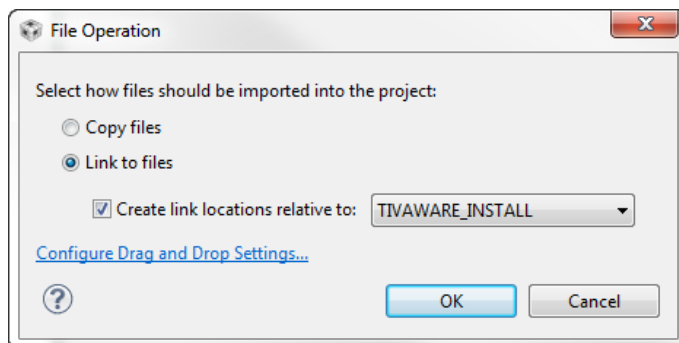
## 9. Liên kết tập tin TivaWare driverlib.lib đến project của bạn.

► Chọn *Project-Add Files...* điều hướng đến:

C:\TI\TivaWare\_C\_Series-1.1\driverlib\ccs\Debug\driverlib.lib

... và ► click Open. The File Operation sẽ mở ra ...

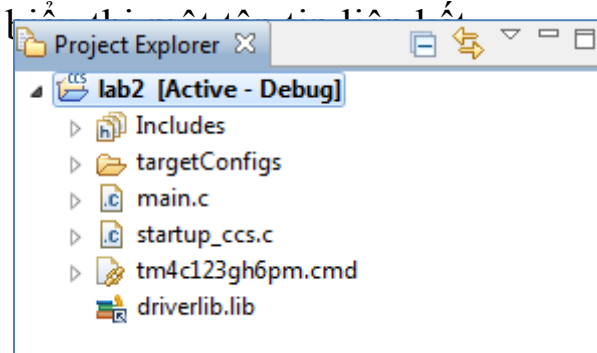
Sử dụng biến con đường TIVAWARE\_INSTALL bạn tạo ra trước đó. Điều này có nghĩa rằng LINK (hoặc tham chiếu đến thư viện) tập tin sẽ được RELATIVE đến vị trí của các cài đặt TivaWare. Nếu bạn giao dự án này cho người khác, họ có thể cài đặt dự án bất cứ nơi nào trong hệ thống tập tin và liên kết này sẽ vẫn làm việc. Nếu bạn chọn PROJECT\_LOC, bạn sẽ có được một con đường đó là tương đối so với vị trí của dự án của bạn và nó sẽ yêu cầu các dự án phải được cài đặt ở "cấp độ" tương tự trong cấu trúc thư mục. Một ưu điểm khác của phương pháp này là nếu bạn muốn liên kết đến một phiên bản mới, nói TivaWare\_C\_Series-1.2, tất cả các bạn phải làm là sửa đổi các biến với tên thư mục mới.



► Làm cho phần hiển thị và click OK.

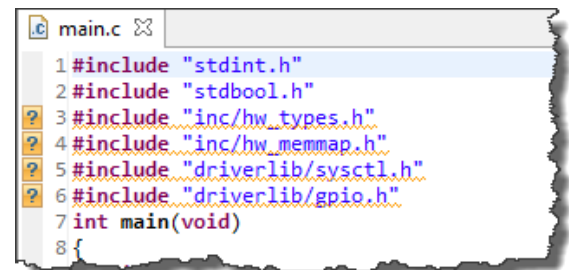


Dự án của bạn bây giờ trông giống như ảnh chụp màn hình dưới đây. Lưu ý các biểu tượng cho driverlib.lib



## 10. Thêm những đường dẫn tìm kiếm INCLUDE cho những tập tin tiềm kiếm.

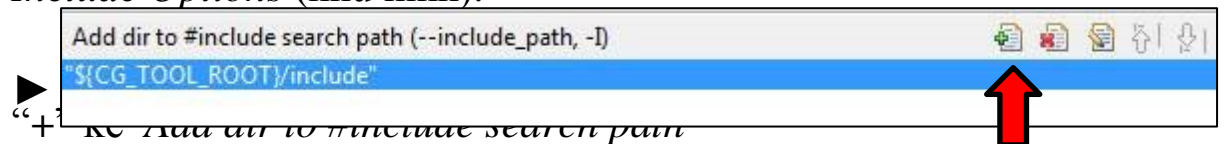
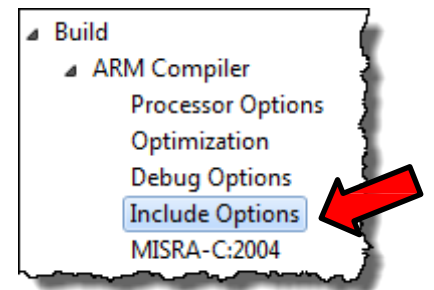
► Mở main.c bằng click đôi trên filename trong Project Explorer pane của CCS. Bạn sẽ thấy “?” cảnh báo ở lề trái mà chỉ "hòa nhập chưa được giải quyết". Di chuột qua các dấu chấm hỏi để xem các thông điệp hữu ích.



Cho đến bây giờ, bạn đã không nói với dự án nơi để tìm các tập tin tiêu đề.

► click phải trên lab2 project của bạn trong Project Explorer pane và chọn *Properties*.

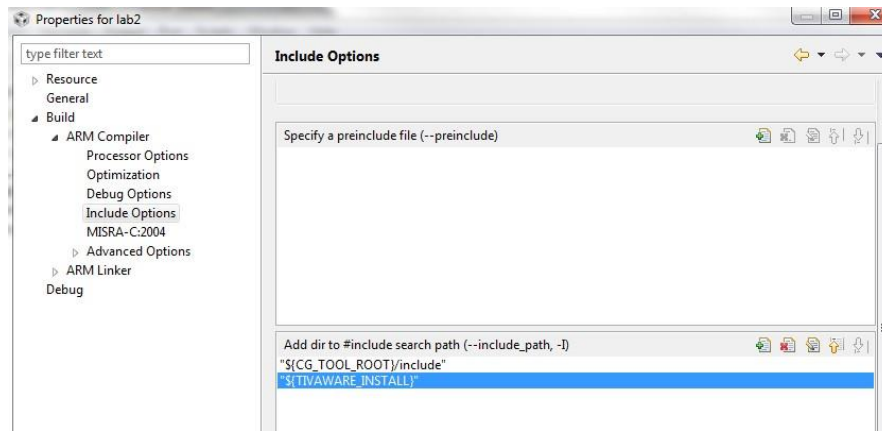
► Click vào *Build* → *ARM Compiler* → *Include Options* (như hình):



và thêm theo sau đường dẫn sử dụng build variable bạn đã tạo ra sớm hơn. Nơi variable name bên trong **braces**, sau \$ như hình:

`${TIVAWARE_INSTALL}`

► Click OK.



► Click OK lần nữa và bây giờ bạn sẽ thấy cái này “?” trong main.c biến mất sau một lúc. Vấn đề được giải quyết.

## 11. Kiểm tra các tập tin project của bạn sử dụng Windows

### Explorer

► Sử dụng Windows Explorer, dẫn đến thư mục lab2 project của bạn:

C:\TM4C123G\_LaunchPad\_Workshop\lab2\project

Bạn có thấy main.c? Nó sẽ được ở đó bởi vì bạn đã sao chép nó ở đó. Bạn có thấy tập tin driverlib.lib? Tập tin này không nên có mặt ở đó vì nó chỉ được liên kết trong project của bạn. Chú ý các thư mục khác trong thư mục \project- những chứa các cài đặt cụ thể CCS project của bạn. Đóng Windows Explorer.

## 12. Kiểm tra các tính chất của project mới của bạn



► Trong CCS, click phải vào project của bạn và chọn *Properties*. Click trên mỗi phần dưới đây:

**Resource:** Điều này sẽ cho bạn thấy con đường của project hiện tại của bạn và con đường giải quyết nếu nó được liên kết vào workspace. Click trên “*Linked Resources*” và cả hai tab kết hợp với điều này.

Đường dẫn PROJECT\_LOC là gì? \_\_\_\_\_

Có bất kỳ nguồn tài nguyên liên quan? Nếu vậy, tập tin gì(s)? \_\_\_\_\_

**General:** hiện các thiết lập dự án chính. Chú ý là bạn có thể thay đổi gần như mọi lĩnh vực ở đây SAU KHI dự án đã được tạo ra.

**Build → ARM Compiler:** Các thiết lập của trình biên dịch cơ bản cùng với tất cả các trình biên dịch thiết lập cho dự án của bạn.

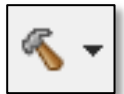
**Other:** click vào một vài thiết lập nhiều hơn, nhưng không thay đổi bất kỳ của chúng.

► Click *Cancel*.

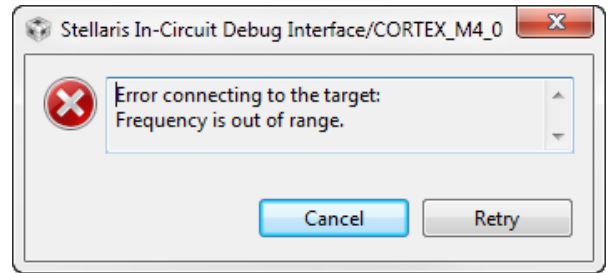
## Buid, Load, Run

### 13. Xây dựng project của bạn và sửa lỗi

► Đảm bảo rằng LaunchPad của bạn được kết nối với máy tính xách tay của bạn. Xây dựng và tải dự án của bạn vào bộ nhớ flash TM4C123GH6PM bằng cách nhấn vào nút Debug. nếu bạn bao giờ muốn xây dựng dự án mà không cần tải nó, nhấp vào nút HAMMER (Xây dựng).



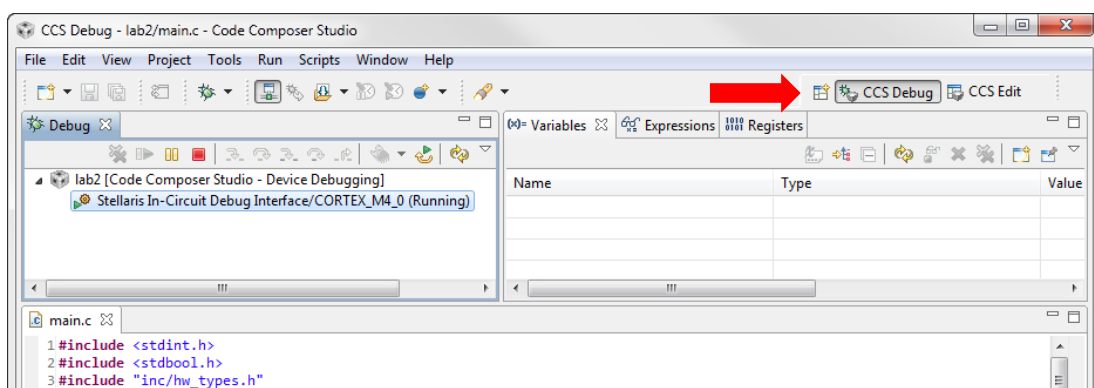
► Sửa chữa bất kỳ lỗi nào xảy ra. Đối với hiện tại, bạn có thể bỏ qua bất kỳ cảnh báo. Nếu bạn gặp phải các lỗi hiển thị, hội đồng quản trị của bạn bị ngắt kết nối, chuyển đổi năng lượng của bạn là sai vị trí hoặc trình điều khiển được không đúng cách cài đặt.



Bộ đếm chương trình sẽ chạy đến main() và thiết lập như hình:

```
36 void main(void)
37 {
38
```

## 14. Làm quen với giao diện CCS Debug



**Debug Pane**

**Watch & Expressions Panes**

**Code/Editor Pane**

**Console and Problems Panes**

Lưu ý tên của các Code Composer pane trên. Có hai quan điểm được xác định trước trong Code Composer; CCS Chỉnh Edit và CCS Debug. ► Click và kéo các tab (tại mũi tên ở trên) để bên trái, do đó bạn có thể thấy cả hai. Quan điểm chỉ là một "cái nhìn" của dữ liệu có sẵn ... bạn có thể chỉnh sửa mã của bạn ở đây mà không thay đổi quan điểm. Và bạn có thể sửa đổi các hoặc tạo ra nhiều quan điểm khác như bạn muốn. Thêm vào đó trong một thời điểm.

### **15. Chạy chương trình của bạn**

► Click nút Resume hoặc press là key F  n bàn phím:



Các tri-màu LED trên tàu mục tiêu của bạn nên chớp hiển thị ba màu trong chuỗi. Nếu không được, cố gắng giải quyết các vấn đề chính mình trong một vài phút, và sau đó yêu cầu hướng dẫn của bạn để được giúp đỡ.

Để ngăn chặn chương trình đang chạy, ► click the Suspend button:



Nếu mã dừng lại với một "Không có nguồn sẵn có ..." chỉ thị, nhấp chuột vào tab main.c. Hầu hết thời gian trong while () vòng lặp được chỉ bên trong chức năng chậm trễ. Đó là tập tin nguồn không liên quan vào dự án này.

## 16. Đặt một Breakpoint

Trong cửa sổ code ở giữa màn hình của bạn, bấm đúp chuột vào trong vùng màu xanh bên trái của số dòng của các GPIOPinWrite() sự giới thiệu. Đây sẽ đặt một breakpoint (nó sẽ giống hình: ). Click nút Resume  để restart code. Chương trình sẽ dừng tại breakpoint và bạn sẽ thấy một mũi tên ở bên trái của số dòng, chỉ ra rằng chương trình truy cập đã ngừng trên dòng code này. **Lưu ý rằng các lái xe ICDI hiện tại không hỗ trợ thêm hoặc loại bỏ các điểm dừng trong khi bộ xử lý đang chạy.** Nhấp vào nút Resume một vài lần hoặc bấm phím F8 để chạy mã. Quan sát các đèn LED trên bảng LaunchPad khi bạn làm điều này.

## 17. View/Watch memory and variables.

- Click vào tab Expressions trong Watch và Expressions pane.
- Double-click vào biến ui8LED bất kỳ đâu trong main().

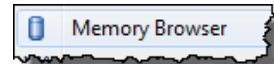
► click phải vào *ui8LED* và chọn:



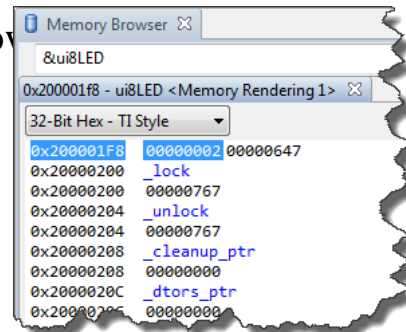
► Click OK. click phải vào *ui8LED* trong Expressions pane, và chọn Number Format ☐ Hex. lưu ý giá trị của *ui8LED*.

Tuy nhiên, biến *ui8LED* được đặt trong SRAM. Bạn có thể thấy địa chỉ trong expressions view. Nhưng hãy lấy nó trong memory.

► Chọn *View* → *Memory Browser*:

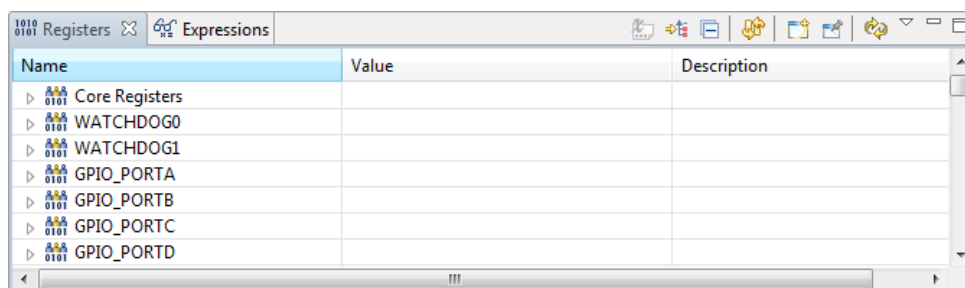


► Gõ *&ui8LED* vào memory window *ui8LED* trong memory:



## 18. Xem các Thanh ghi

► Chọn *View* → *Registers* và nhận thấy rằng bạn có thể xem nội dung của tất cả các đăng ký trong kiến trúc của đối tượng. Điều này rất tiện dụng cho mục đích gỡ lỗi.



► Bấm vào mũi tên ở bên trái để mở rộng các điểm đăng ký. Lưu ý rằng thiết bị ngoại vi không thuộc hệ thống chưa được bật, không

thể đọc. Trong dự án này, bạn có thể xem Lỗi các thanh ghi, GPIO\_PORTA (nơi các chân UART đang có), GPIO\_PORTF (nơi các đèn LED và nút bấm được đặt), HIB, FLASH\_CTRL, SYSCTL và NVIC.

## Perspectives

CCS perspectives là khá linh hoạt. Bạn có thể tùy chỉnh các quan điểm (s) và lưu chúng như là quan điểm riêng của bạn nếu bạn muốn. Thật dễ dàng để thay đổi kích cỡ, phóng to, mở quan điểm khác nhau, quan điểm gắn gủi, và thỉnh thoảng, bạn có thể tự hỏi "Làm thế nào để có được những thứ trở lại bình thường?"

### 19. Hãy di chuyển một số cửa sổ xung quanh và sau đó thiết lập lại perspective.

► click phải vào tab cửa sổ *Console* và chọn “*Detached*”. Bây giờ bạn có thể di chuyển này cửa sổ xung quanh bất cứ nơi nào bạn muốn. ► Nhấp chuột phải và chọn "tách rời" để lại dính kèm nó.


Trong cửa sổ chỉnh sửa, ► nhấp đúp vào tab hiển thị `main.c`:



Chú ý rằng cửa sổ soạn thảo nhằm tối đa hóa toàn màn hình. Double-click vào các tab một lần nữa để khôi phục lại nó.

► Di chuyển một số cửa sổ xung quanh trên máy tính để bàn của bạn bằng cách click và giữ trên các tab. Bất cứ khi nào bạn bị mất hoặc một số cửa sổ dường như đã biến mất trong hoặc CCS Edit, CCS

Debug hoặc quan điểm riêng của bạn, bạn có thể khôi phục lại việc bố trí cửa sổ trở lại mặc định.

► Tìm và click vào nút Restore  bên trái hoặc bên phải của hiển thị. Nếu bạn muốn reset view về mặc định factory bạn cũng có thể chọn *Window* → *Reset Perspective*:

**LƯU Ý:** Không sử dụng các tab quan điểm để di chuyển qua lại giữa các quan điểm.

Nhấp vào tab CCS gỡ lỗi chỉ làm thay đổi quan điểm; nó không kết nối với điện thoại, tải về mã hoặc bắt đầu một phiên debug. Tương tự như vậy, nhấn vào CCS thẻ Edit không chấm dứt một phiên debug.

Chỉ sử dụng Debug và Chấm dứt nút để di chuyển giữa các quan điểm trong hội thảo này.

## 20. Gỡ tất cả breakpoints

► Click *Run* ☐ *Remove All Breakpoints* từ thanh menu hoặc double-click vào breakpoint symbol trong editor pane. mặc khác, breakpoints chỉ có thể được loại bỏ khi bộ xử lý không chạy.

### Terminate the debug session.



► Click nút red Terminate để chấm dứt các phiên gỡ lỗi và trở về góc độ CCS Edit.

## VAR.S.INI – Một cách dễ dàng hơn để Thêm Biến

Nhớ lại rằng trước đó trong phòng thí nghiệm bạn tạo ra hai biến - một biến path và một biến build. Chúng là những bộ biến giống nhau đặt CÙNG đường dẫn, nhưng được sử dụng trong hai cách khác nhau - một là

cho liên kết tập tin vào dự án của bạn và các khác đã được sử dụng để bao gồm đường dẫn tìm kiếm trong quá trình xây dựng.

Các biến mà bạn đã tạo trước đó đã có sẵn trên một cấp độ dự án. Vì vậy, nếu bạn đã có hai dự án mở trong vùng làm việc của bạn, các dự án khác sẽ không thể sử dụng các biến mà bạn đã tạo.

Bây giờ, chúng tôi sẽ chỉ cho bạn làm thế nào để thêm các biến gần như tự động cho WORKSPACE của bạn để bất kỳ dự án trong vùng làm việc có thể sử dụng chúng.

## **21. Sử dụng vars.ini để thiết lập đường dẫn không gian làm việc và xây dựng các biến.**

Đầu tiên hãy nhìn vào tập tin mới được gọi là vars.ini. ► Chọn *File* ☐ *Open File* và browse đến:

C:\TM4C123G\_LaunchPad\_Workshop\  
vars.ini

► Click Open

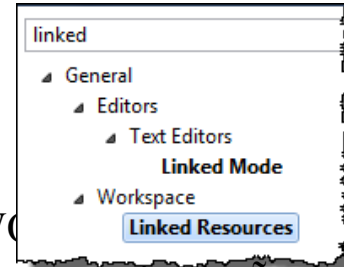
bạn sẽ tìm biến đơn TIVAWARE\_INSTALL được ghi vào bên trong file:





Trước khi chúng ta import tập tin này vào workspace, chúng ta hãy xem vị trí các biến được lưu trữ.

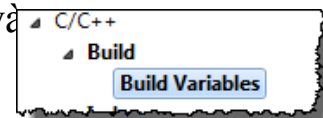
► Chọn *Window* > *Preferences*. khi dialogue xuất hiện, ► gõ “*linked*” vào filter field như hình – sau đó click vào *Linked Resources*:



Điều này sẽ hiển thị tất cả biến đường dẫn Workspace. Chúng tôi thiết lập các biến ở cấp PROJECT trước. Chúng tôi đã sẵn sàng để cài đặt chúng ở cấp không gian làm việc để tất cả các dự án trong vùng làm việc của chúng tôi có thể sử dụng các biến tương tự.

Bạn chỉ có thể thêm các biến ở đây bằng tay, nhưng nhập khẩu chúng từ vars.ini là đơn giản và sẽ thiết lập cả hai biến cùng một lúc.

► Gõ “*build*” vào filter area và click vào *Build Variables* như hình:



Đây là nơi bạn có thể thiết lập mức độ không gian làm việc xây dựng

chỉ định. Một lần nữa, bạn chỉ có thể thêm các biến bây giờ thủ công, nhưng vars.ini sẽ làm điều này cho chúng tôi.

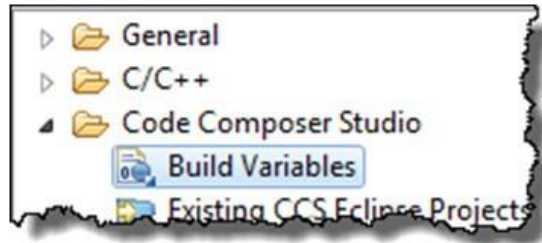
Cả hai *Linked Resources* và *Build Variables* khu vực biến cho không gian làm việc của bạn là BLANK - không chứa biến không gian làm việc ở tất cả. Đó là về để thay đổi ...

► Click *Cancel*.



Hãy import file vars.ini và thấy điều gì xảy ra....

- ▶ Chọn *File* ☐ *Import*, sau khi mở rộng CCS category, click vào *Build Variables* (như hình):



- ▶ Click *Next* và browse đến nơi vars.ini:  
C:\TM4C123G\_LaunchPad\_Workshop\vars.ini

- ▶ Click *Open*, sau đó click *Finish*. ▶ Sau khi chọn *Window Preferences* và đặt *WORKSPACE* path variable và build variable của bạn. Chúng hiển thị? Nó cần phải có import các biến được liệt kê vào cả hai con đường, xây dựng khu vực biên (như hình):

Name	Value
TIVAWARE_INSTALL	C:\TI\TivaWare_C_Series-1.0

- ▶ Click OK. Minimize Code Composer.

## Using VARS.INI – Conclusion

Bây giờ, bất kỳ dự án trong vùng làm việc của bạn (giống như tất cả các phòng thí nghiệm trong tương lai tại hội thảo này) có thể sử dụng các biến mà không cần bất kỳ nhập khẩu nhiều hơn. Họ là một phần của không gian làm việc của bạn. Ngoài ra, nếu bạn xuất một dự án và trao nó cho một

người bạn, các biến không gian làm việc sẽ không được bao gồm trong dự án. Đó là khá tiện dụng. Tại sao? bạn của bạn có thể có một vị trí cài đặt KHÁC với các công cụ. Vì vậy, nếu họ sử dụng các tên Workspace BIẾN cùng, nhưng con đường khác nhau, họ xây dựng sẽ chỉ làm việc tốt. Bây giờ bạn có một dự án hoàn thành và hoàn toàn PORTABLE PROJECT.

---

Lưu ý: Nếu bạn thay đổi không gian làm việc, bạn sẽ phải tải nhập vars.ini để thiết lập các biến một lần nữa. Nếu thay đổi công cụ của bạn cài đặt, bạn sẽ phải chỉnh sửa vars.ini và tải nhập. Vì vậy, hãy cẩn thận.

---

## LM Flash Programmer

LM flash Lập trình là một giao diện lập trình mà cho phép bạn để chương trình flash của thiết bị Tiva C Series thông qua nhiều cổng. Tạo các tập tin cần thiết cho điều này là một bước xây dựng riêng biệt trong Code Composer rằng nó được hiển thị trên trang tiếp theo. Nếu bạn đã không làm như vậy, cài đặt các LM flash Lập trình trên máy tính của bạn.

Hãy chắc chắn rằng Code Composer Studio không tích cực chạy mã trong quan điểm CCS gỡ lỗi ... nếu không CCS và Flash Lập trình có thể xung đột để kiểm soát các cổng USB.

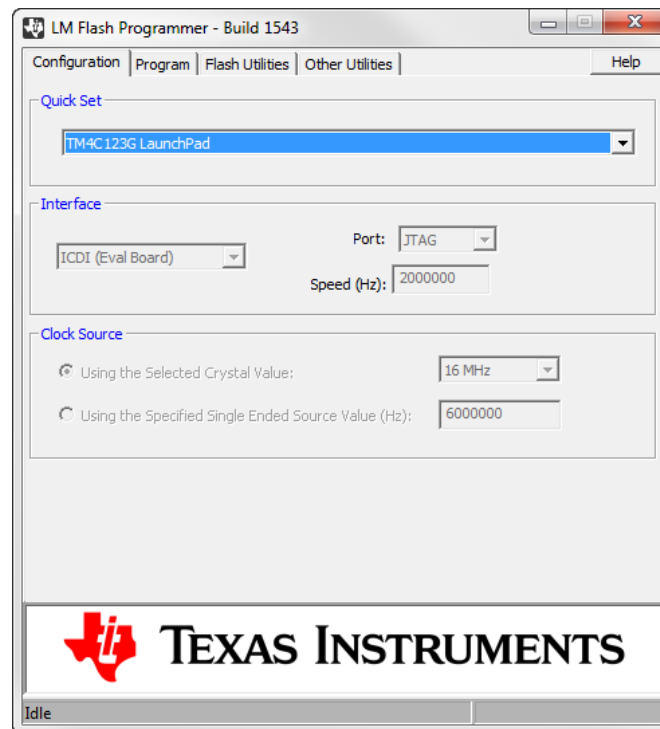
### 22. Mở LM Flash Programmer

Nên có một phím tắt để các LM flash Lập trình trên máy tính để bàn của bạn, kích đúp vào nó để mở công cụ. Nếu các phím tắt không xuất hiện, đến *Start*  
☐ *All Programs* ☐ *Texas Instruments* ☐ *Stellaris*  
☐ *LM Flash Programmer* và click vào *LM Flash Programmer*.

evaluation board của bạn hiện nay nên được lập trình với các ứng dụng lab2 và nó cần được chạy. Nếu đèn LED dùng không nhấp nháy, nhấn nút RESET trên điện đàn. Chúng tôi đang đi để chương trình ứng dụng ban đầu trở lại vào bộ nhớ flash TM4C123GH6PM.

► Click vào tab Configuration. Chọn *TM4C123G LaunchPad* từ Quick Set pull-down menu dưới tab Configuration. **Nếu *TM4C123G LaunchPad* không xuất hiện, chọn *LM4F120 LaunchPad* từ danh sách.**

Xem hướng dẫn của người sử dụng thông tin về làm thế nào để tự cấu hình các công cụ cho các mục tiêu mà không phải là evaluation boards .



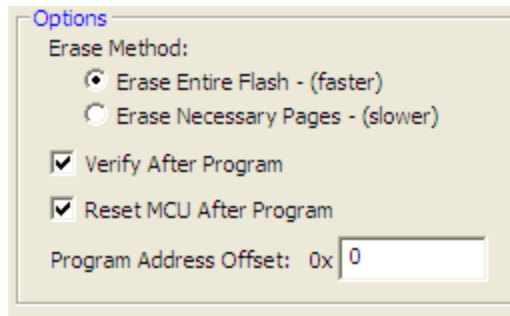
**23. Click vào Program Tab, sau đó click vào nút Browse và điều hướng đến:**

c:\TI\TivaWare\_C\_Series-1.1\examples\boards\ek-tm4c123gx1\qs-rgb\ccs\Debug\qs-rgb.bin

và ► click Open. Bạn có thể thấy rằng cách nhấp vào biểu tượng ► chứ không phải là tên tập tin dễ dàng để điều hướng. qs-rgb là ứng dụng đã được lập trình trong bộ nhớ flash của TM4C123GH6PM khi bạn gỡ bỏ nó từ hộp.

Lưu ý rằng có những ứng dụng ở đây đã được xây dựng với mỗi IDE hỗ trợ.

► Hãy chắc chắn rằng các hộp kiểm được chọn:



## 24. Chương trình

► Click vào nút Program. Bạn sẽ thấy các lập trình và trạng thái xác minh ở dưới cùng của cửa sổ. Sau khi các bước này hoàn tất, khởi động ứng dụng nhanh chóng nên được chạy trên LaunchPad của bạn.

## 25. Đóng LM Flash Programmer

### Optional: Creating a bin File for the Flash Programmer

Nếu bạn muốn tạo một file .bin để sử dụng bởi các lập trình viên độc lập trong bất kỳ phòng thí nghiệm tại hội thảo này hoặc trong dự án riêng của bạn, sử dụng các bước dưới đây.

Hãy nhớ rằng dự án sẽ phải được mở trước khi bạn có thể thay đổi thuộc tính của nó.

## 26. Set Post-Build step to call “tiobj2bin” utility



► Trong CCS Project Explorer, right-click vào project của bạn và chọn *Properties*. On the left, click Build và sau đó *Steps* tab. Dán các lệnh sau đây vào *Post-build steps Command box*.

---

**Note:** Bốn "dòng" sau đây sẽ được nhập là một dòng trong *Command box*. Để làm cho nó dễ dàng hơn, chúng tôi bao gồm một tập tin văn bản mà bạn có thể sao chép-dán. Hướng đến  
C:\TM4C123G\_LaunchPad\_Workshop\postbuild.txt để tìm dòng lệnh hoàn thành.

---

```
"${CCS_INSTALL_ROOT}/utils/tiobj2bin/tiobj2bin"  
"${BuildArtifactFileName}"  
"${BuildArtifactFileName}.bin"  
"${CG_TOOL_ROOT}/bin/armofd"  
"${CG_TOOL_ROOT}/bin/armhex"  
"${CCS_INSTALL_ROOT}/utils/tiobj2bin/mkhex4bin"
```

## 27. Rebuild project của bạn

Bước post-build sẽ chạy sau khi project builds của bạn và tập tin .bin sẽ trong thư mục

C:\TM4C123G\_LaunchPad\_Workshop\labx\project\debug. Bạn có thể vào file .bin trong CCS Project Explorer ở project của bạn bằng cách mở rộng thư mục Debug.

Nếu bạn thử re-build và bạn nhận được tin “gmake: Nothing to be done for ‘all’.”, điều này chỉ ra rằng không có tập tin đã thay đổi trong dự án của bạn kể từ lần cuối cùng bạn xây dựng nó. Bạn có thể buộc các dự án xây dựng bằng cách kích chuột

phải vào dự án đầu tiên và sau đó chọn *Clean Project*. Bây giờ bạn có thể re-build project của bạn cái mà sẽ chạy trong post-build step để tạo file .bin.

28. Nếu bạn mở lab2 để thực hiện những bước này, đóng nó bây giờ.



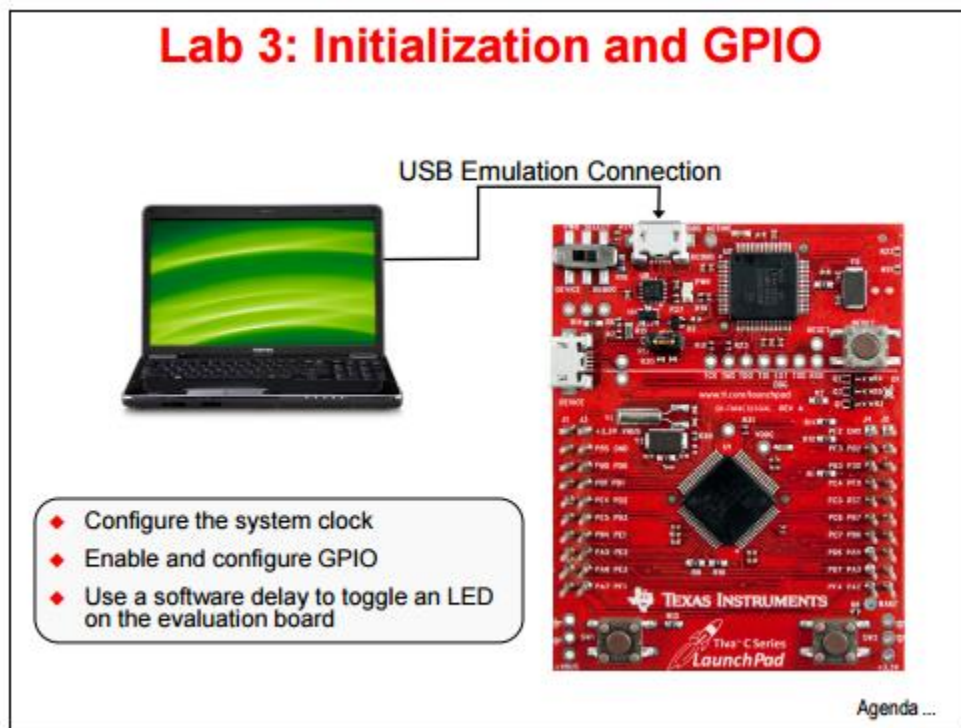
Bạn hoàn thành xong.

# CHƯƠNG 3

## Lab 3: Initialization and GPIO

### 1.Mục đích thí nghiệm

- + Trong bài thí nghiệm này, chúng ta sẽ tìm hiểu làm thế nào để khởi tạo các hệ thống clock và sử dụng các thiết bị ngoại vi GPIO của kit TivaWare.
- + Sau đó chúng ta sẽ sử dụng đầu ra GPIO nhấp nháy một đèn LED trên board.



### 2.Các bước tiến hành thí nghiệm

- + Tạo dự án lab3
- + Trên thanh công cụ menu chọn File => New => CCS project.
- + Lựa chọn như hình dưới đây. Hãy chắc chắn để bỏ chọn "Sử dụng vị trí mặc định" và chọn đúng đường dẫn đến thư mục project như được hiển thị.
- + Khi khởi tạo hoàn tất, nhấp vào bên cạnh lab3 trong cửa sổ

Project Explorer để mở rộng dự án.

- + Xóa các nội dung hiện tại của main.c.
- + Thêm các dòng sau vào hàm main.c bao gồm các tập tin cần thiết để truy cập vào các API TivaWare và định nghĩa biến:

```
#include <stdint.h>
#include <stdbool.h>
#include "inc / hw_memmap.h"
#include "inc / hw_types.h"
#include "driverlib / sysctl.h"
#include "driverlib / gpio.h"
uint8_t ui8PinData = 2;
```

- + Việc sử dụng dấu <> hạn chế đường dẫn tìm kiếm, để chỉ đường dẫn cụ thể. Sử dụng " " để bắt đầu trong thư mục.

- + Ý nghĩa của một số file header:

**stdint.h:** Các định nghĩa biến cho các tiêu chuẩn C99.

**stdbool.int:** Định nghĩa Boolean cho các tiêu chuẩn C99.

**hw\_memmap.h:** Macros xác định bản đồ bộ nhớ của thiết bị Tiva C Series. Điều này bao gồm định nghĩa như ngoại vi ,địa điểm, địa chỉ cơ sở như GPIO\_PORTF\_BASE.

**hw\_types.h:** Xác định loại phổ biến và macro.

**sysctl.h:** Định nghĩa và macro cho hệ thống điều khiển API của

**DriverLib:** Nó bao gồm hàm API như SysCtlClockSet và SysCtlClockGet.

**gpio.h:** Định nghĩa và macro cho GPIO API của DriverLib bao gồm các chức năng API như GPIOPinTypePWM và GPIOPinWrite.

**uint8\_t ui8PinData = 2:** Tạo ra một biến số nguyên được gọi là ui8PinData và khởi tạo. Điều này sẽ được sử dụng như chu kỳ gi ữa ba đèn LED.

Lưu ý rằng các loại C99 là một số nguyên không dấu 8-bit và tên biến phản ánh điều này.

- + Hàm chính:

Cấu trúc của một hàm chính

```
int main (void)
```

```
{
}
```

Nếu bạn gõ dấu “{”, trình soạn thảo sẽ tự động thêm ngoặc “}”

phía sau.

- + Thiết lập xung clock

- + Cấu hình xung clock của hệ thống sử dụng một tinh thể có tần số 16MHz trên dao động chính, điều khiển các PLL 400MHz . Các PLL 400MHz dao động ở tần số đó, nhưng có thể được điều khiển bởi các tinh thể hoặc dao động từ 5 đến 25MHz. Có một default / 2 chia trong clock và chúng ta đang xác định khác / 5, tổng số 10. hệ thống có các clock sẽ là 40MHz.

- + Nhập dòng mã này bên trong main ():

```
SysCtlClockSet (SYSCTL_SYSDIV_5 |  
SYSCTL_USE_PLL | SYSCTL_XTAL_16MHZ |  
SYSCTL_OSC_MAIN);
```

- +Tinh thể gắn vào các đầu vào dao động chính là 16MHz, trong khi các tinh thể gắn vào RTC đầu vào là 32,768Hz.

- + Trước khi gọi bất kỳ chức năng ngoại vi driverLib, chúng ta phải cung cấp clock cho ngoại vi. Nếu bạn không làm điều này, nó sẽ gây ra một lỗi ISR (lỗi địa chỉ) .Đây một sai lầm phổ biến cho người dùng Tiva C Series mới. Tiếp theo ta cần chỉnh cấu hình, ba chân GPIO kết nối với các đèn LED là kết quả đầu ra. Hình dưới đây cho thấy GPIO chân PF1, PF2 và PF3 được kết nối với các đèn LED.

- + Để lại một dòng cho khoảng cách, sau đó nhập vào hai dòng mã bên trong main () sau các dòng trong bước trước.

```
SysCtlPeripheralEnable (SYSCTL_PERIPH_GPIOF);  
GPIOPinTypeGPIOOutput (GPIO_PORTF_BASE,  
GPIO_PIN_1 | GPIO_PIN_2 | GPIO_PIN_3);
```

- +APB đề cập đến các thiết bị ngoại vi Bus nâng cao, trong khi AHB đề cập đến sự trình diễn nâng cao. Trong phòng thí nghiệm của chúng ta, GPIO\_PORTF\_BASE là 0x40025000.

```
GPIO Port A (APB): 0x4000.4000
```

```
GPIO Port A (AHB): 0x4005.8000
```

```
GPIO Port B (APB): 0x4000.5000
```

```
GPIO Port B (AHB): 0x4005.9000
```

```
GPIO Cổng C (APB): 0x4000.6000
```

```
GPIO Cổng C (AHB): 0x4005.A000
```

```
GPIO Port D (APB): 0x4000.7000
```

```
GPIO Port D (AHB): 0x4005.B000
```

**GPIO Cổng E (APB): 0x4002.4000**

**GPIO Cổng E (AHB): 0x4005.C000**

**GPIO Cổng F (APB): 0x4002.5000**

**GPIO Cổng F (AHB): 0x4005.D000**

**while () Loop**

+ Cuối cùng, tạo một thời gian (1) vòng lặp chậm trễ.

SysCtlDelay () là một bộ đếm thời gian loop cung cấp trong TivaWare. Các tham số tính là vòng lặp đếm, không chậm trễ thực tế trong chu kỳ đồng hồ. Mỗi vòng lặp là 3 chu kỳ CPU.

+ Để viết với các chân GPIO, sử dụng các GPIO API chức năng gọi GPIOPinWrite. Bảo đảm để đọc và hiểu cách các chức năng GPIOPinWrite được sử dụng trong datasheet. Các tranh luận dữ liệu thứ ba không chỉ đơn giản là 1 hoặc 0, nhưng đại diện cho toàn bộ cổng dữ liệu 8-bit. Các số thứ hai là một mặt nạ bit-đóng gói của các dữ liệu được ghi.

+ Trong ví dụ của chúng tôi dưới đây, chúng tôi đang viết các giá trị trong biến ui8PinData cho cả ba chân GPIO được kết nối với các đèn LED. Được viết để dựa trên mặt nạ bit quy định. Các chu kỳ chỉ dẫn cuối cùng thông qua các đèn LED bằng cách làm ui8PinData bằng 2, 4, 8, 2, 4, 8 và như vậy. Lưu ý rằng các giá trị gửi đến các chân phù hợp vị trí của họ; "một" trong vị trí bit hai chỉ có thể đạt được pin chút hai trên cổng.

Bây giờ có thể là một thời điểm tốt để nhìn vào các thông số kỹ thuật cho thiết bị Tiva C Series của bạn. kiểm tra ra chương GPIO để hiểu được cách duy nhất các dữ liệu đăng ký GPIO được thiết kế và những ưu điểm của phương pháp này.

**while (1)**

```
{  
    GPIOPinWrite (GPIO_PORTF_BASE, GPIO_PIN_1 |  
    GPIO_PIN_2 | GPIO_PIN_3, ui8PinData);  
    SysCtlDelay (2000000);  
    GPIOPinWrite (GPIO_PORTF_BASE, GPIO_PIN_1 |  
    GPIO_PIN_2 | GPIO_PIN_3, 0x00);  
    SysCtlDelay (2000000);  
    if (ui8PinData == 8) {ui8PinData = 2;} else {ui8PinData =  
    ui8PinData * 2;}  
}
```

+ Ta có chương trình hoàn chỉnh như sau :

```
#include <stdint.h>
#include <stdbool.h>
#include "inc/hw_memmap.h"
#include "inc/hw_types.h"
#include "driverlib/sysctl.h"
#include "driverlib/gpio.h"
uint8_t ui8PinData=2;
int main(void)
{
    SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_XTAL_16MHZ|SYSCTL_OSC_MAIN);
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);
    GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3);
    while(1)
    {
        GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, ui8PinData);
        SysCtlDelay(2000000);
        GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, 0x00);
        SysCtlDelay(2000000);
        if(ui8PinData==8) {ui8PinData=2;} else {ui8PinData=ui8PinData*2;}
    }
}
```

+ Ta có một sự chậm trễ trong 3-6 chu kỳ của thiết bị ngoại vi.

+ Kích hoạt tính năng GPIO

### **Config ADC**

### **Config GPIO**

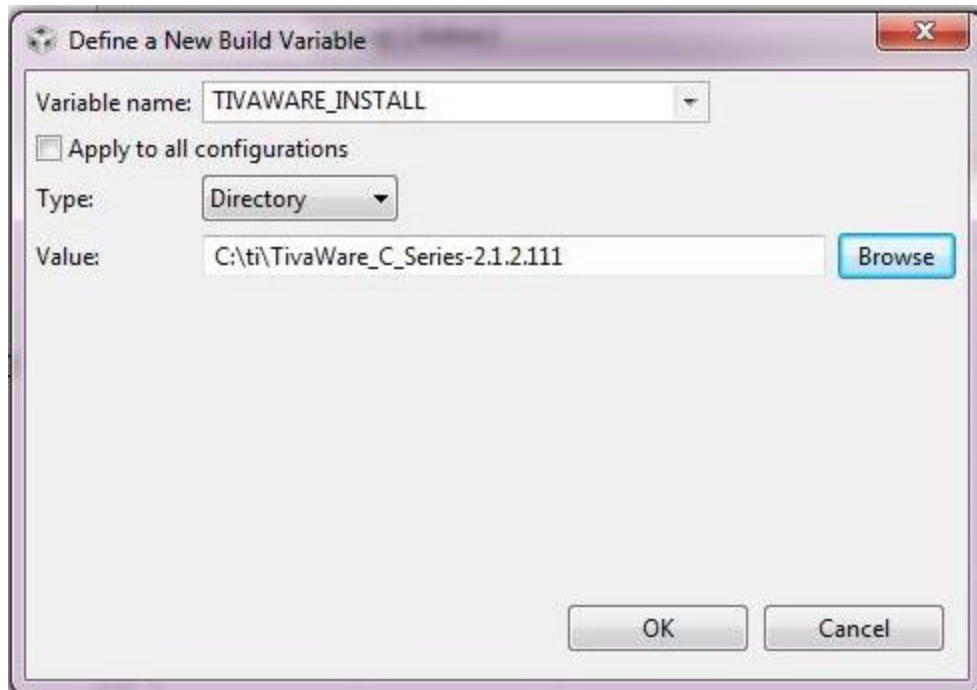
+ Mã khởi động ngoài các tập tin chính bạn đã tạo ra, bạn cũng cần có một tập tin khởi động cụ thể tập tin này có chứa các bảng vector, ta cần khởi động để sao chép dữ liệu khởi tạo vào RAM và xóa phần bss, và ISRs lỗi mặc định. Các project mới Wizard sẽ tự động thêm một bản sao của tập tin

+ Click đôi vào file tm4c123gh6pm\_startup\_ccs.c trong cửa sổ làm việc và xem xét. Không thực hiện bất kỳ thay đổi vào thời điểm này. Đóng tập tin.

+ Thiết lập các tùy chọn xây dựng

Nhấp chuột phải vào Lab3 trong cửa sổ Project Explorer và chọn Properties. Nhấp vào include Options trong ARM Compiler. Ở phía dưới, nhấp vào nút Add và thêm đường dẫn tìm kiếm sau đây:

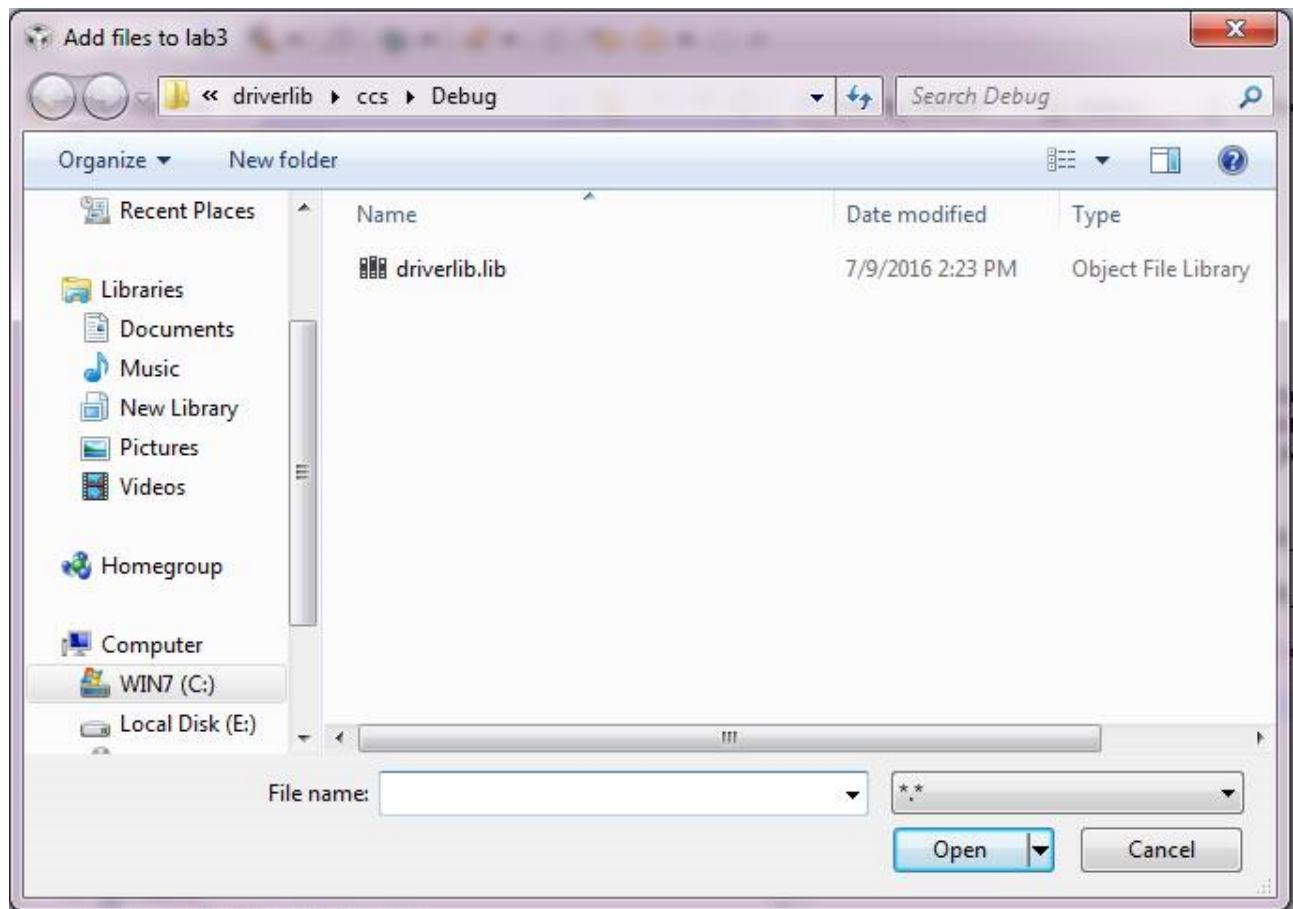
**\$ {} TIVAWARE\_INSTALL**



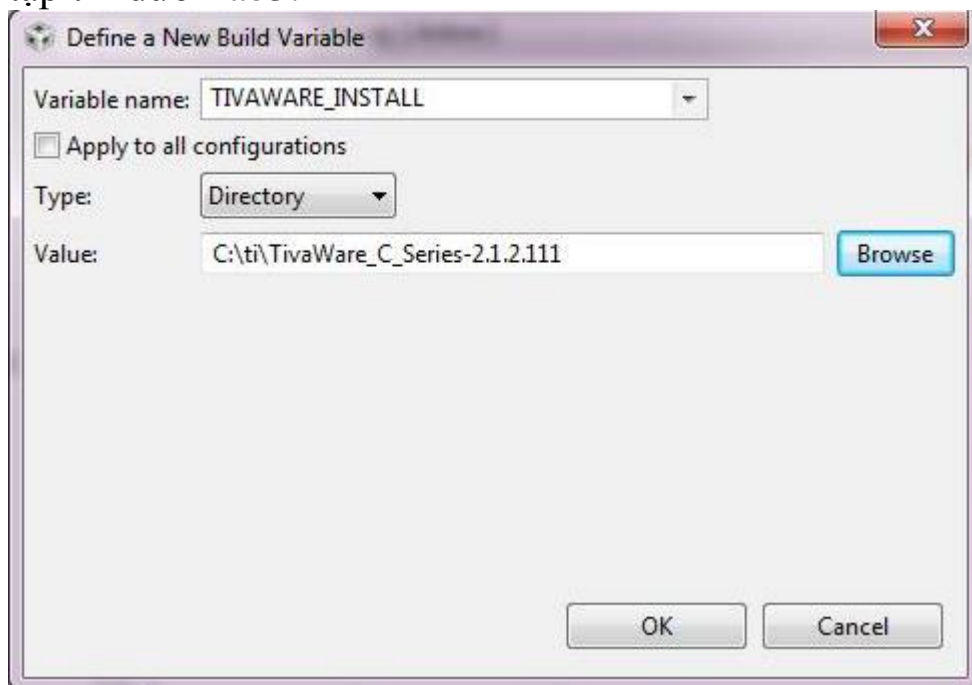
+ Đây là đường dẫn chúng ta đã tạo trước đó bằng cách sử dụng các tập tin vars.ini trong dự án lab2. Kể từ khi những đường dẫn được quy định tại các mức độ không gian làm việc, chúng tôi chỉ đơn giản là có thể sử dụng nó một lần nữa ở đây.

+ Thêm Driver :Thư viện Tập tin các tập tin driverlib.lib cần phải được thêm vào trong dự án lab3. Trong lab2 chúng tôi đã thêm một liên kết đến tập tin. Ta có thể kéo nó qua, hãy thử xem nào.





+ Nhấp và giữ driverlib.lib thuộc dự án lab2 trong Project Explorer của sổ. Kéo nó vào dự án lab3. Bây giờ bạn sẽ thấy các tập tin dưới lab3.



+ Các tập tin đã được kết nối với lab2 được liên kết tới lab3.  
Nếu ta thấy dòng [Active - Debug]. Điều này sẽ cho bạn biết rằng các dự án đang hoạt động và lab3 đang xây dựng cấu hình là debug.

+ Hãy thay đổi mã trở lại ban đầu thiết lập: Thực hiện những thay đổi sau đây:

+ Tìm dòng có chứa `uint8_t ui8PinData = 14;` và thay đổi nó trở lại:

**`uint8_t ui8PinData = 2;`**

+ Tìm dòng có chứa `if (ui8PinData ...` và bỏ nó.

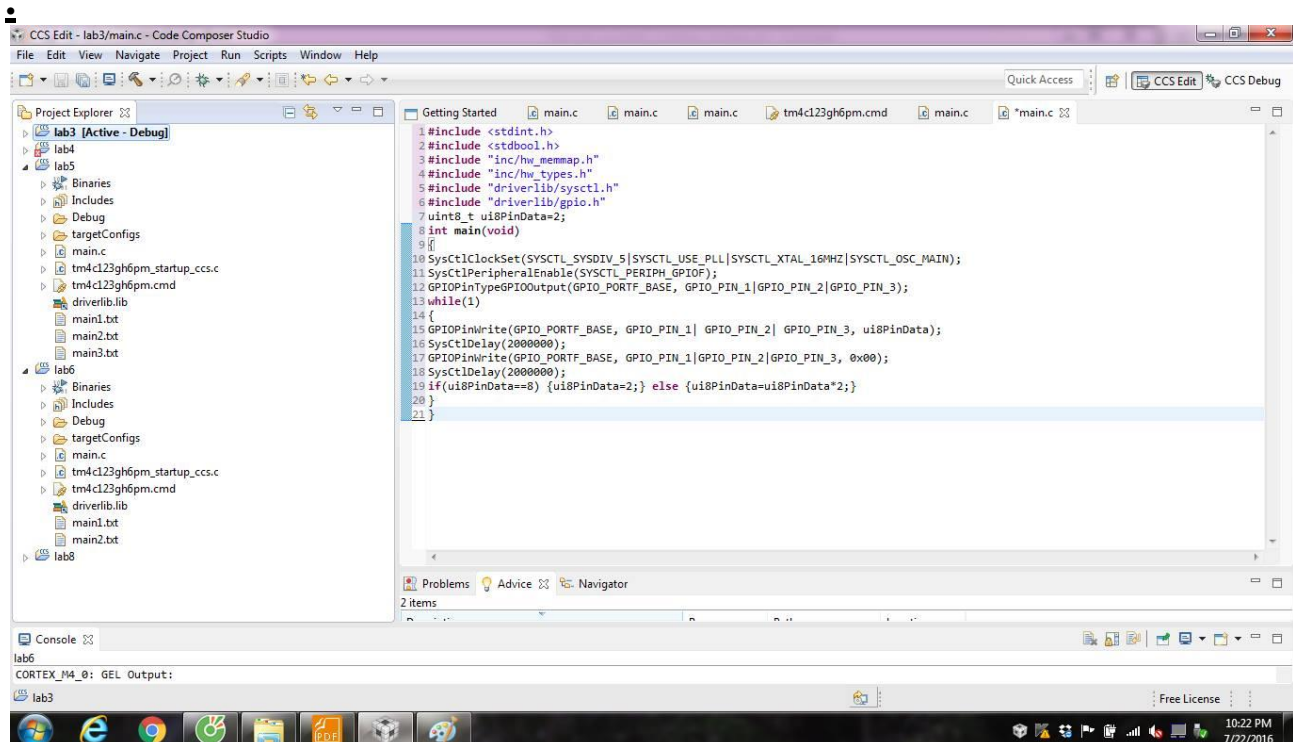
+ Tìm dòng có chứa các `GPIO_PinWrite` đầu tiên () và thay đổi nó trở lại:

**`GPIO_PinWrite (GPIO_PORTF_BASE, GPIO_PIN_1 |  
GPIO_PIN_2 | GPIO_PIN_3, ui8PinData);`**

+ Biên dịch và chạy.

+ Bấm vào nút Terminate để trở về CCS Edit.

+ Giảm thiểu Code Composer Studio.



### 3.Kết quả

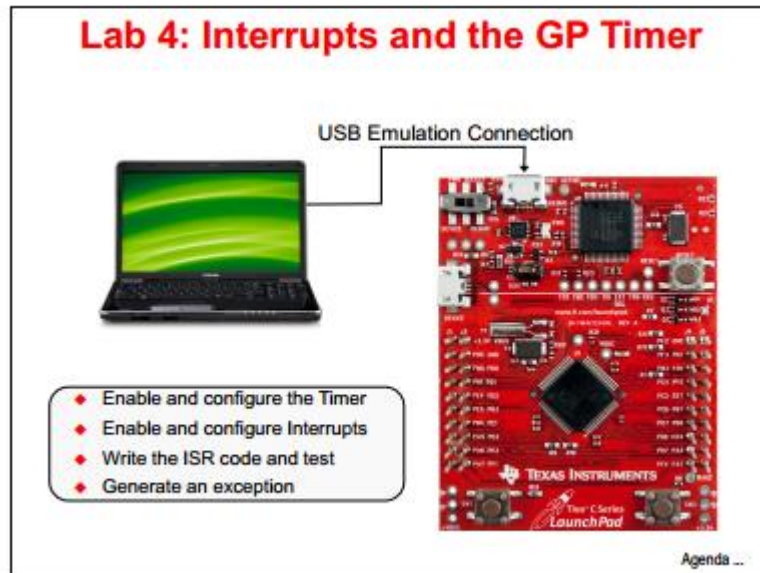


# CHƯƠNG 4

## Lab 4: Interrupts and the Timer

### 1.Mục đích thí nghiệm.

+ Trong bài thí nghiệm này, chúng ta sẽ thiết lập bộ đếm thời gian để tạo ra ngắt, và sau đó viết code mà đáp ứng ngắt nhấp nháy đèn LED. Chúng tôi cũng sẽ thử nghiệm với việc tạo ra một hệ thống, bằng cách cố gắng để cấu hình một thiết bị ngoại vi trước khi kích hoạt nó.



### 2.Quy trình thí nghiệm.

+ Ta tiến hành mở project bằng cách chọn Project => Import Existing CCS Eclipse Project => chọn đường dẫn đến lab4 => chọn finish.

+ Click chuột vào tên lab4 để mở rộng dự án.

+ Chọn vào file main.c mở file main.c ta thêm các dòng khai báo file header dưới đây :

```
#include <stdint.h>
#include <stdbool.h>
#include "inc/tm4c123gh6pm.h"
```

```

#include "inc/hw_memmap.h"
#include "inc/hw_types.h"
#include "driverlib/sysctl.h"
#include "driverlib/interrupt.h"
#include "driverlib/gpio.h"
#include "driverlib/timer.h"

```

+ Ý nghĩa của một số file header:

+ tm4c123gh6pm.h: Các định nghĩa cho các gián đoạn và đăng ký thiết bị Tiva C Series trên bảng LaunchPad.

+ interrupt.h: Định nghĩa và mở rộng cho NVIC Controller (Interrupt) API

+ driverLib. Điều này bao gồm các chức năng API như IntEnable và IntPrioritySet.

+ timer.h: Định nghĩa và macro cho hẹn giờ API của driverLib. Điều này bao gồm API các chức năng như TimerConfigure và TimerLoadSet.

+ Hàm chính

```

int main(void)
{
    uint32_t ui32Period ;
}

```

+ Thiết lập xung clock

```

SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_
XTAL_16MHZ|SYSCTL_OSC_MAIN);

```

+ Cấu hình GPIO

```

SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);
GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE,
GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3);

```

+ Cấu hình timer

```

SysCtlPeripheralEnable(SYSCTL_PERIPH_TIMER0);
TimerConfigure(TIMER0_BASE, TIMER_CFG_PERIODIC);

```

+ Tính toán delay

```

ui32Period = (SysCtlClockGet() / 10) / 2;
TimerLoadSet(TIMER0_BASE, TIMER_A, ui32Period -1);
Cho phép ngắt
intEnable(INT_TIMER0A);
TimerIntEnable(TIMER0_BASE, TIMER_TIMA_TIMEOUT);
IntMasterEnable();

```

+ Bật timer

```

TimerEnable(TIMER0_BASE, TIMER_A);
While(1) loop
While
{
}

```

+ Giải quyết các vấn đề ngắt timer

```

void Timer0IntHandler(void)
{
// Clear the timer interrupt
TimerIntClear(TIMER0_BASE, TIMER_TIMA_TIMEOUT);
// Read the current state of the GPIO pin and
// write back the opposite state
if(GPIOPinRead(GPIO_PORTF_BASE, GPIO_PIN_2))
{
GPIOPinWrite(GPIO_PORTF_BASE,
GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, 0);
}
else
{
GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_2, 4);
}
}

```

+ Ta được chương trình hoàn chỉnh như sau :

```

#include <stdint.h>
#include <stdbool.h>
#include "inc/tm4c123gh6pm.h"
#include "inc/hw_memmap.h"
#include "inc/hw_types.h"
#include "driverlib/sysctl.h"
#include "driverlib/interrupt.h"
#include "driverlib/gpio.h"
#include "driverlib/timer.h"
int main(void)
{
    uint32_t ui32Period;
    SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_X
    TAL_16MHZ|SYSCTL_OSC_MAIN);
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);
    GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE,
    GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3);
    SysCtlPeripheralEnable(SYSCTL_PERIPH_TIMER0);
    TimerConfigure(TIMER0_BASE, TIMER_CFG_PERIODIC);
    ui32Period = (SysCtlClockGet() / 10) / 2;

```

```

96     IntDefaultHandler,           // PWM Generator 0
97     IntDefaultHandler,           // PWM Generator 1
98     IntDefaultHandler,           // PWM Generator 2
99     IntDefaultHandler,           // Quadrature Encoder 0
100    IntDefaultHandler,           // ADC Sequence 0
101    IntDefaultHandler,           // ADC Sequence 1
102    IntDefaultHandler,           // ADC Sequence 2
103    IntDefaultHandler,           // ADC Sequence 3
104    IntDefaultHandler,           // Watchdog timer
105    Timer0IntHandler,             // Timer 0 subtimer A
106    IntDefaultHandler,           // Timer 0 subtimer B
107    IntDefaultHandler,           // Timer 1 subtimer A
108    IntDefaultHandler,           // Timer 1 subtimer B
109    IntDefaultHandler,           // Timer 2 subtimer A
110    IntDefaultHandler,           // Timer 2 subtimer B
111    IntDefaultHandler,           // Analog Comparator 0
112    IntDefaultHandler,           // Analog Comparator 1
113    IntDefaultHandler,           // Analog Comparator 2
114    IntDefaultHandler,           // System Control (PLL, OSC, BO)
115    IntDefaultHandler,           // FLASH Control
116    IntDefaultHandler,           // GPIO Port F
117    IntDefaultHandler,           // GPIO Port G
118    IntDefaultHandler,           // GPIO Port H
119    IntDefaultHandler,           // UART2 Rx and Tx

```



```

32 void ResetISR(void);
33 static void NmiISR(void);
34 static void FaultISR(void);
35 static void IntDefaultHandler(void);
36
37 //*****
38 //
39 // External declaration for the reset handler that is to be called when the
40 // processor is started
41 //
42 //*****
43 extern void _c_int00(void);
44 extern void Timer0IntHandler(void);
45 //*****
46 //
47 // Linker variable that marks the top of the stack.
48 //
49 //*****
50 extern uint32_t __STACK_TOP;
51
52 //*****

```

```

TimerLoadSet(TIMER0_BASE, TIMER_A, ui32Period -1);
IntEnable(INT_TIMER0A);
TimerIntEnable(TIMER0_BASE, TIMER_TIMA_TIMEOUT);
IntMasterEnable();
TimerEnable(TIMER0_BASE, TIMER_A);
while(1)
{
}
void Timer0IntHandler(void)
{
    // Clear the timer interrupt
    TimerIntClear(TIMER0_BASE, TIMER_TIMA_TIMEOUT);
    // Read the current state of the GPIO pin and
    // write back the opposite state
    if(GPIOPinRead(GPIO_PORTF_BASE, GPIO_PIN_2))
    {
        GPIOPinWrite(GPIO_PORTF_BASE,
        GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, 0);
    }
    else
    {
        GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_2, 4);
    }
}

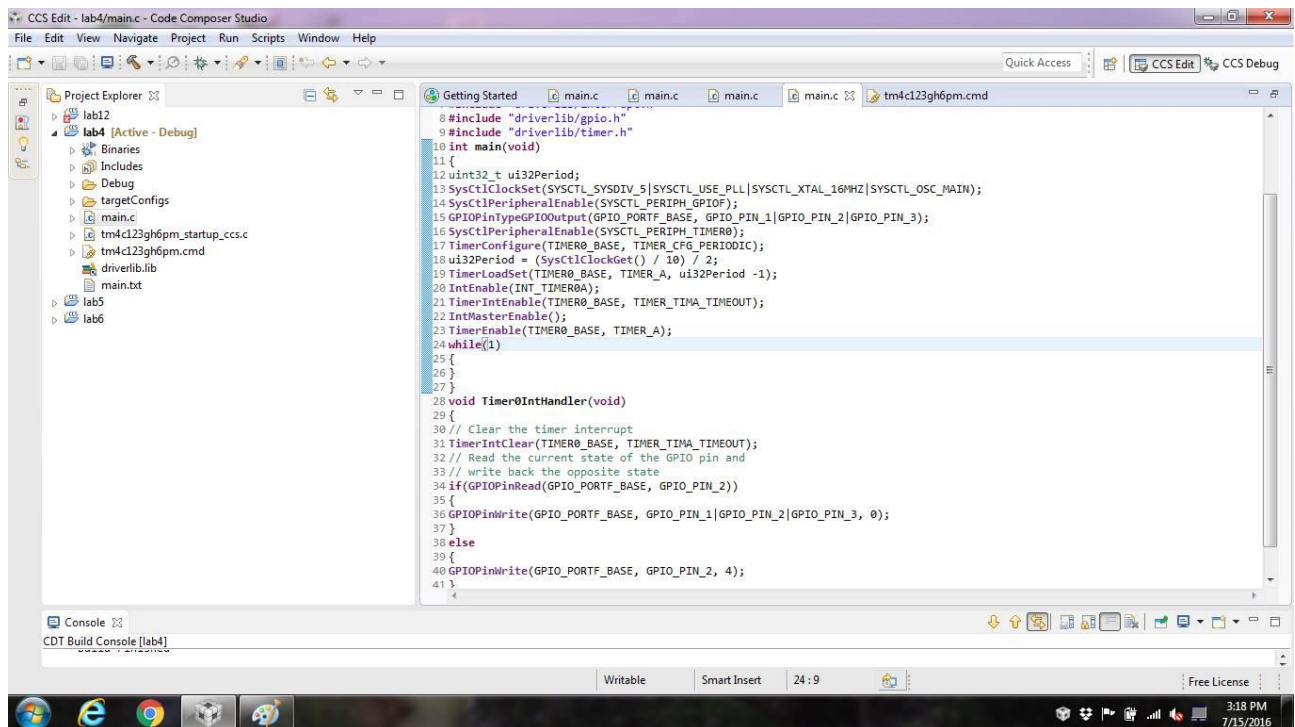
```



```
}  
}
```

### Khởi động code

Mở file `tm4c123gh6pm_startup_ccs.c` thay thế dòng `IntDefaultHandler` bằng dòng `Timer0Intandler` và tìm dòng `extern void _c_int00(void);` ta thêm vào dòng `extern void Timer0IntHandler(void)` sau nó.



+ Sau đó biên dịch load và chạy chương trình.

### 3.Kết quả.



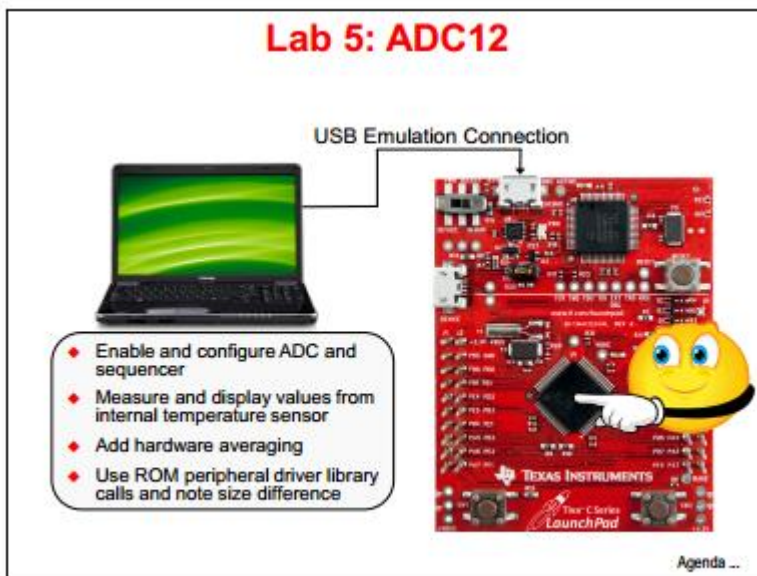


# CHƯƠNG 5

## Lab 5: ADC12

### 1. Mục đích thí nghiệm.

+ Trong bài thí nghiệm này chúng ta sẽ sử dụng ADC12 và mẫu trình tự để đo lường các dữ liệu từ trên chip như cảm biến nhiệt độ. Chúng tôi sẽ sử dụng Code Composer để hiển thị các giá trị thay đổi.



### 2. Các bước tiến hành thí nghiệm.

+ Ta tiến hành mở project bằng cách chọn Project => Import Existing CCS Eclipse Project => chọn đường dẫn đến lab4 => chọn finish.

+ Click chuột vào tên lab4 để mở rộng dự án.

+ Chọn vào file main.c mở file main.c ta thêm các dòng khai báo file header dưới đây:

```
#include <stdint.h>
#include <stdbool.h>
#include "inc/hw_memmap.h"
#include "inc/hw_types.h"
#include "driverlib/debug.h"
#include "driverlib/sysctl.h"
#include "driverlib/adc.h"
```

+ Thiết lập hàm chính main bằng cách thêm ba dòng dưới đây:

```
int main (void)
```

```
{  
  
}
```

+ Thêm dòng mã sau đây vào bên trong hàm main ():

```
uint32_t ui32ADC0Value [4];
```

+ Chúng ta sẽ cần một số biến để tính nhiệt độ từ các dữ liệu cảm biến. Đầu tiên biến là để lưu trữ trung bình của nhiệt độ. Các biến còn lại được sử dụng để lưu trữ các giá trị nhiệt độ Celsius và Fahrenheit.

+ Thêm các dòng sau vào cuối cùng:

```
volatile uint32_t ui32TempAvg;
```

```
volatile uint32_t ui32TempValueC;
```

```
volatile uint32_t ui32TempValueF;
```

+ Thiết lập hệ thống clock một lần nữa để chạy ở 40MHz. Thêm các dòng sau:

```
SysCtlClockSet (SYSCTL_SYSDIV_5 | SYSCTL_USE_PLL |  
SYSCTL_OSC_MAIN | SYSCTL_XTAL_16MHZ);
```

+ Và thêm dòng này sau:

```
SysCtlPeripheralEnable (SYSCTL_PERIPH_ADC0);
```

+ Đối với các phòng thí nghiệm này, chúng ta sẽ chỉ cho phép các ADC12 để chạy ở tốc độ mặc định của 1Msps.

+ Thêm một dòng cho khoảng cách và thêm dòng mã này:

```
ADCSequenceConfigure (ADC0_BASE, 1,  
ADC_TRIGGER_PROCESSOR, 0);
```

+Thêm ba dòng sau đây sau các dòng trên:

```
ADCSequenceStepConfigure (ADC0_BASE, 1, 0, ADC_CTL_TS);
```

```
ADCSequenceStepConfigure (ADC0_BASE, 1, 1, ADC_CTL_TS);
```

```
ADCSequenceStepConfigure (ADC0_BASE, 1, 2, ADC_CTL_TS);
```

+Thêm một dòng cho khoảng cách và nhập ba dòng mã:

```
while (1)  
{  
  
}
```

+Ta có một chương trình hoàn chỉnh như sau :

```
include <stdint.h>
```

```
#include <stdbool.h>
```

```
#include "inc/hw_memmap.h"
```

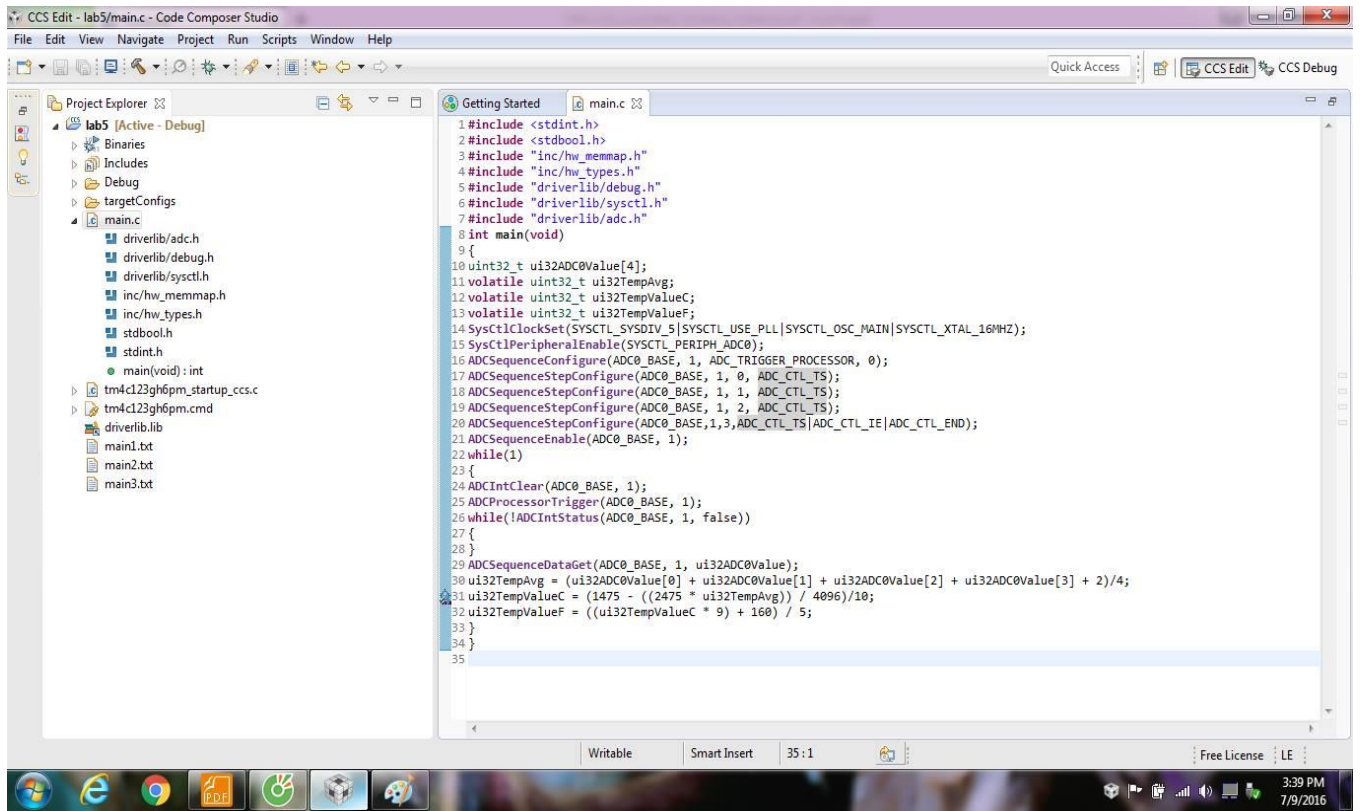
```
#include "inc/hw_types.h"
```

```

#include "driverlib/debug.h"
#include "driverlib/sysctl.h"
#include "driverlib/adc.h"
int main(void)
{
    uint32_t ui32ADC0Value[4];
    volatile uint32_t ui32TempAvg;
    volatile uint32_t ui32TempValueC;
    volatile uint32_t ui32TempValueF;
    SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_OSC_MAIN|SYSCTL_XTAL_16MHZ);
    SysCtlPeripheralEnable(SYSCTL_PERIPH_ADC0);
    ADCSequenceConfigure(ADC0_BASE, 1,
        ADC_TRIGGER_PROCESSOR, 0);
    ADCSequenceStepConfigure(ADC0_BASE, 1, 0,
        ADC_CTL_TS);
    ADCSequenceStepConfigure(ADC0_BASE, 1, 1,
        ADC_CTL_TS);
    ADCSequenceStepConfigure(ADC0_BASE, 1, 2,
        ADC_CTL_TS);
    ADCSequenceStepConfigure(ADC0_BASE, 1, 3, ADC_CTL_TS |
        ADC_CTL_IE | ADC_CTL_END);
    ADCSequenceEnable(ADC0_BASE, 1);
    while(1)
    {
        ADCIntClear(ADC0_BASE, 1);
        ADCProcessorTrigger(ADC0_BASE, 1);
        while(!ADCIntStatus(ADC0_BASE, 1, false))
        {}
        ADCSequenceDataGet(ADC0_BASE, 1, ui32ADC0Value);
        ui32TempAvg = (ui32ADC0Value[0] +
            ui32ADC0Value[1] + ui32ADC0Value[2] +
            ui32ADC0Value[3] + 2)/4;
        ui32TempValueC = (1475 - ((2475 * ui32TempAvg))
            / 4096)/10;
        ui32TempValueF = ((ui32TempValueC * 9) + 160) /

```

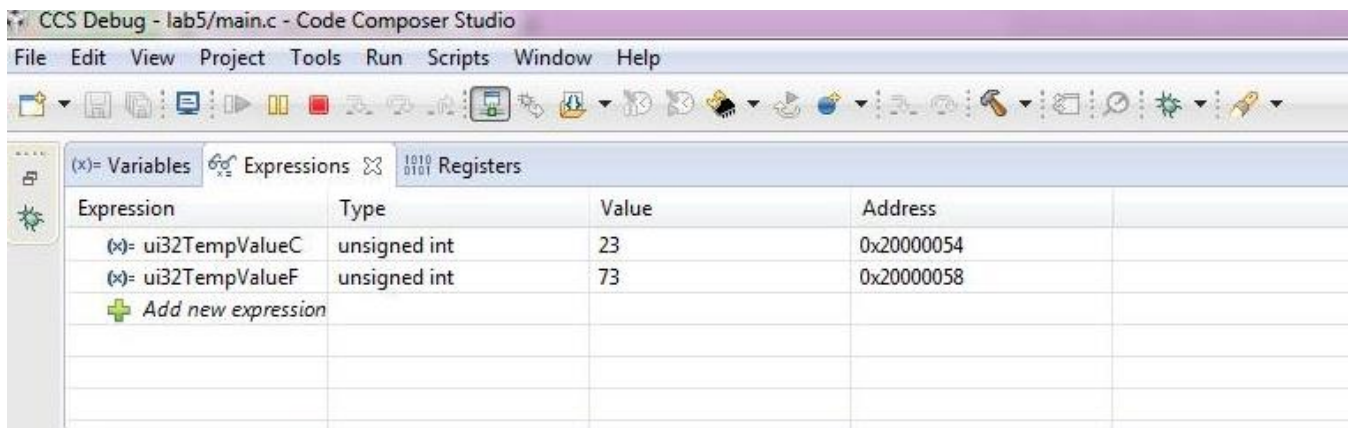
```
5;
}
```



```
}
```

+Lưu công việc của bạn.

### 3. Kết quả thí nghiệm.



<div> <div> </div> <div>Quick Access</div> <div> </div> </div>			
(x)= Variables	Expressions	Registers	
Expression	Type	Value	Address
ui32TempValueC	unsigned int	23	0x20000054
ui32TempValueF	unsigned int	73	0x20000058
ui32ADC0Value	unsigned int[4]	0x20000040	0x20000040
ui32TempAvg	unsigned int	2053	0x20000050
Add new expression			

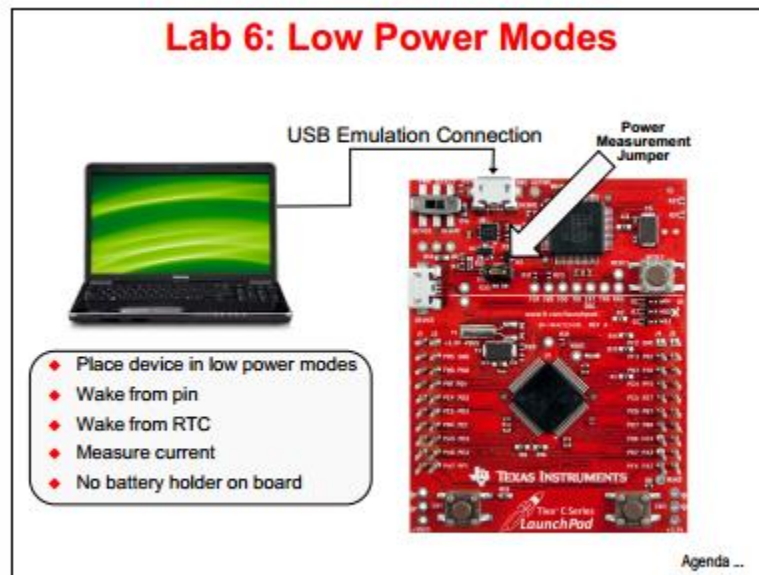


# CHƯƠNG 6

## Lab 6: Low Power Modes

### 1.Mục đích thí nghiệm.

+Trong bài thí nghiệm này, chúng ta sẽ sử dụng các module ngủ đông để đặt các thiết bị trong một trạng thái năng lượng thấp. Sau đó chúng ta sẽ thiết lập chúng hoạt động lại từ cả pin đánh thức và Real-Time Clock (RTC). Chúng ta cũng sẽ đo và vẽ để xem những ảnh hưởng của chế độ năng lượng khác nhau.



### 2.Các bước thí nghiệm.

+ Ta tiến hành mở project bằng cách chọn Project => Import Existing CCS Eclipse Project => chọn đường dẫn đến lab6 => chọn finish.

+ Click chuột vào tên lab6 để mở rộng dự án.

+ Chọn vào file main.c mở file main.c ta thêm các dòng khai báo file header dưới đây:



```

#include <stdint.h>
#include <stdbool.h>
#include "utils/ustdlib.h"
#include "inc/hw_types.h"
#include "inc/hw_memmap.h"
#include "driverlib/sysctl.h"
#include "driverlib/pin_map.h"
#include "driverlib/debug.h"
#include "driverlib/hibernate.h"
#include "driverlib/gpio.h"

```

+ Ý nghĩa của một số file header:

**tm4c123gh6pm.h:** Các định nghĩa cho các gián đoạn và đăng ký thiết bị Tiva C Series trên bảng LaunchPad.

**interrupt.h:** Định nghĩa và mở rộng cho NVIC Controller (Interrupt) API

**driverLib.** Điều này bao gồm các chức năng API như IntEnable và IntPrioritySet.

**timer.h:** Định nghĩa và macro cho hẹn giờ API của driverLib. Điều này bao gồm API các chức năng như TimerConfigure và TimerLoadSet.

+ Hàm chính

```

Int main(void)
{
    uint32_t ui32Period ;
}

```

+ Thiết lập xung clock

```

SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_XTAL_16MHZ|SYSCTL_OSC_MAIN);

```

+ Cấu hình GPIO

```

SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);
GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE,
GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3);

```

**GPIOPinWrite(GPIO\_PORTF\_BASE,GPIO\_PIN\_1|GPIO\_PIN\_2|GPIO\_PIN\_3, 0x08);**

+ Cấu hình hibernate.

Hãy nhìn vào sơ đồ bảng và xem như thế nào pin WAKE được kết nối để sử dụng nút nhấn 2 (SW2) trên bảng LaunchPad.

Đoạn code dưới đây có các chức năng sau:

Dòng 1: Cho phép các module ngủ đông

Dòng 2: Xác định clock cung cấp cho các module ngủ đông

Dòng 3: Gọi chức năng này cho phép pin bang GPIO được duy trì trong thời gian ngủ đông và vẫn hoạt động ngay cả khi thức dậy từ chế độ ngủ đông.

Dòng 4: Trì hoãn 4 giây để bạn có thể quan sát các LED

Dòng 5: Thiết lập các điều kiện.

Dòng 6: Tắt đèn LED màu xanh trước khi thiết bị đi ngủ

+ Thêm một dòng cho khoảng cách và thêm những dòng sau cuối cùng trong main ():

**SysCtlPeripheralEnable (SYSCTL\_PERIPH\_HIBERNATE);**

**HibernateEnableExpClk (SysCtlClockGet ());**

**HibernateGPIORetentionEnable ();**

**SysCtlDelay (64000000);**

**HibernateWakeSet (HIBERNATE\_WAKE\_PIN);**

**GPIOPinWrite (GPIO\_PORTF\_BASE, GPIO\_PIN\_3, 0x00);**

+ Yêu cầu Hibernate

+ Cuối cùng, chúng ta cần phải đi vào chế độ ngủ đông. Các HibernateRequest () chức năng yêu cầu các mô-đun Hibernation để vô hiệu hóa các điều bên ngoài, loại bỏ điện từ xử lý và tắt cả các thiết bị ngoại vi. Nếu điện áp pin là thấp (hoặc tắt) hoặc nếu ngắt đang được phục vụ, chuyển sang chế độ ngủ đông có thể bị trì hoãn. Nếu điện áp pin không có mặt, việc chuyển đổi sẽ không bao giờ xảy ra.

+ Thêm những dòng sau cuối cùng trong hàm main ():

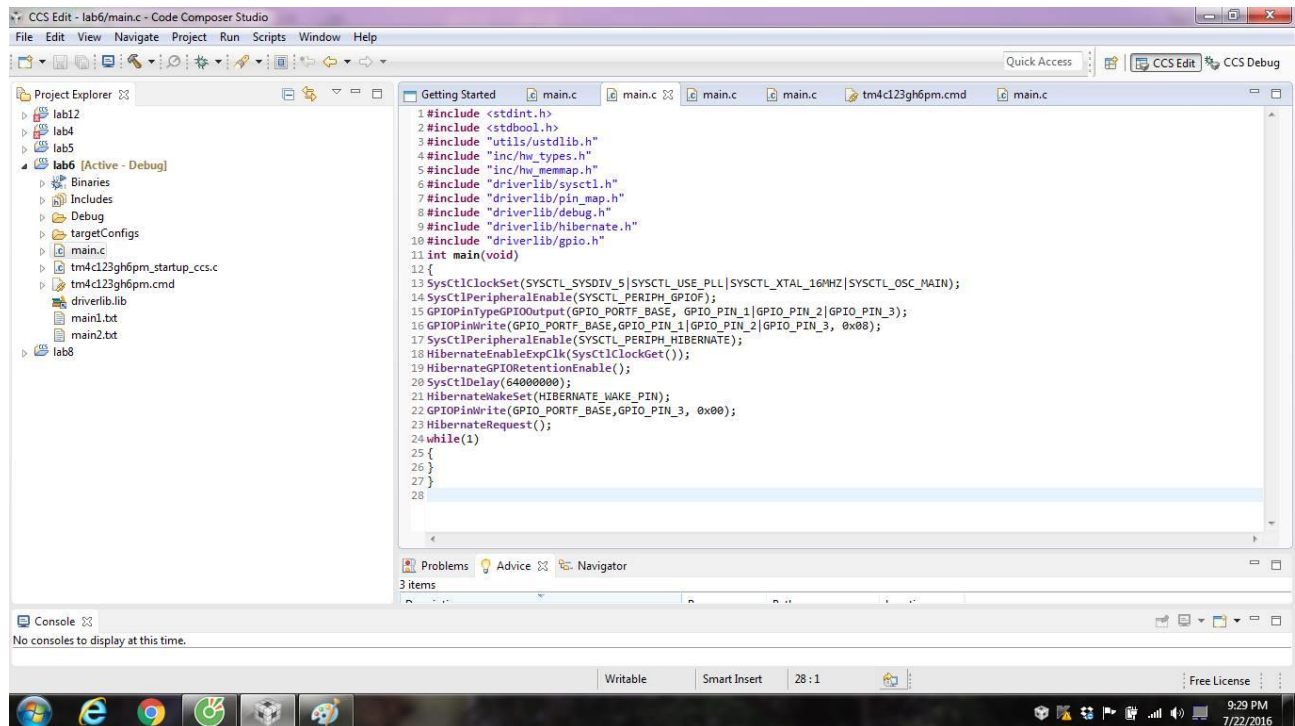
```
HibernateRequest ();  
  
while (1)  
  
{  
  
}
```

+ Ta có chương trình hoàn chỉnh :

```
#include <stdint.h>  
#include <stdbool.h>  
#include "utils/ustdlib.h"  
#include "inc/hw_types.h"  
#include "inc/hw_memmap.h"  
#include "driverlib/sysctl.h"  
#include "driverlib/pin_map.h"  
#include "driverlib/debug.h"  
#include "driverlib/hibernate.h"  
#include "driverlib/gpio.h"  
int main(void)  
{  
    SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYS  
CTL_XTAL_16MHZ|SYSCTL_OSC_MAIN);  
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);  
    GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE,  
GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3);  
    GPIOPinWrite(GPIO_PORTF_BASE,GPIO_PIN_1|GPIO_PIN_  
2|GPIO_PIN_3, 0x08);  
    SysCtlPeripheralEnable(SYSCTL_PERIPH_HIBERNATE);  
    HibernateEnableExpClk(SysCtlClockGet());  
    HibernateGPIORetentionEnable();  
    SysCtlDelay(64000000);  
    HibernateWakeSet(HIBERNATE_WAKE_PIN);  
    GPIOPinWrite(GPIO_PORTF_BASE,GPIO_PIN_3, 0x00);  
    HibernateRequest();  
    while(1)
```

```
}  
}
```

+ Nhấp vào nút Save để lưu công việc của bạn.



### 3.kết quả





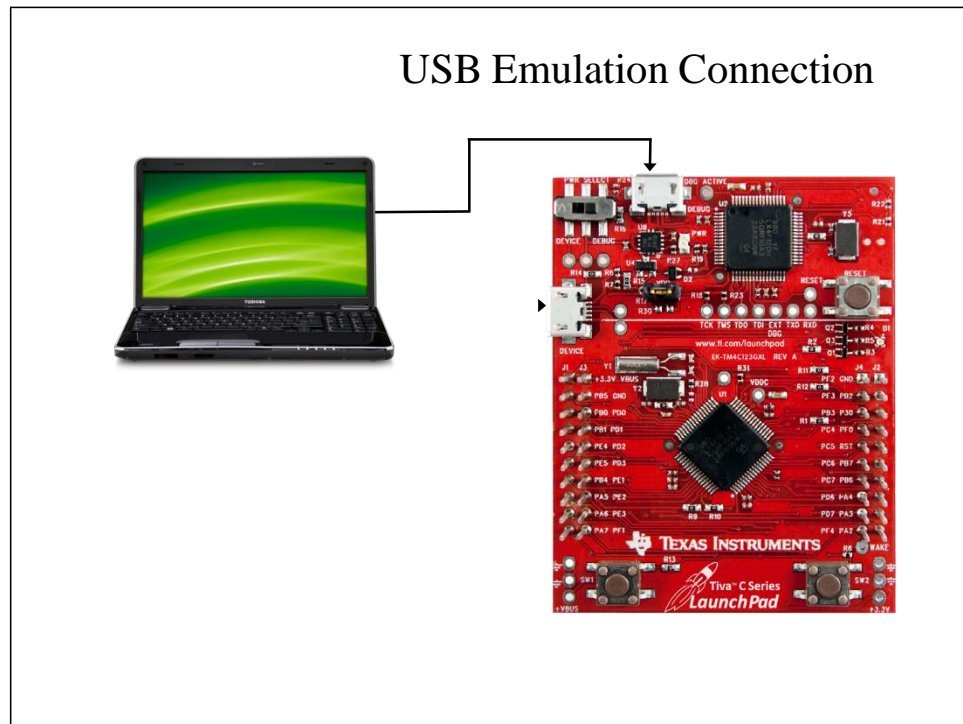


# CHƯƠNG 7

## Lab 7: USB

### 1. Mục tiêu thí nghiệm

Trong bài thí nghiệm này, bạn sẽ thử nghiệm với việc gửi dữ liệu qua lại trên một kết nối với cổng USB.



## 2.Các bước tiến hành

### Code mẫu

+ Có bốn loại chuyển dữ liệu / thiết bị đầu cuối trong kết nối USB: chuyển điều khiển (với lệnh và trạng thái), ngắt chuyển (để nhanh chóng nhận được sự chú ý của host), truyền đẳng thời (chuyển liên tục và định kỳ các dữ liệu) và chuyển số lượng lớn (chuyển lớn, dữ liệu bùng phát).

Trước khi chúng ta bắt đầu với đoạn code này, chúng ta hãy cài usb\_bulk\_example cho quá trình thí nghiệm. Chúng tôi sẽ sử dụng một ứng dụng dòng lệnh máy chủ Windows để chuyển dây qua kết nối USB cho Ban LaunchPad. Chương trình này sẽ thay đổi từ chữ thường thành hoa và ngược lại, sau đó chuyển dữ liệu lại cho host.

### Lấy Project mẫu

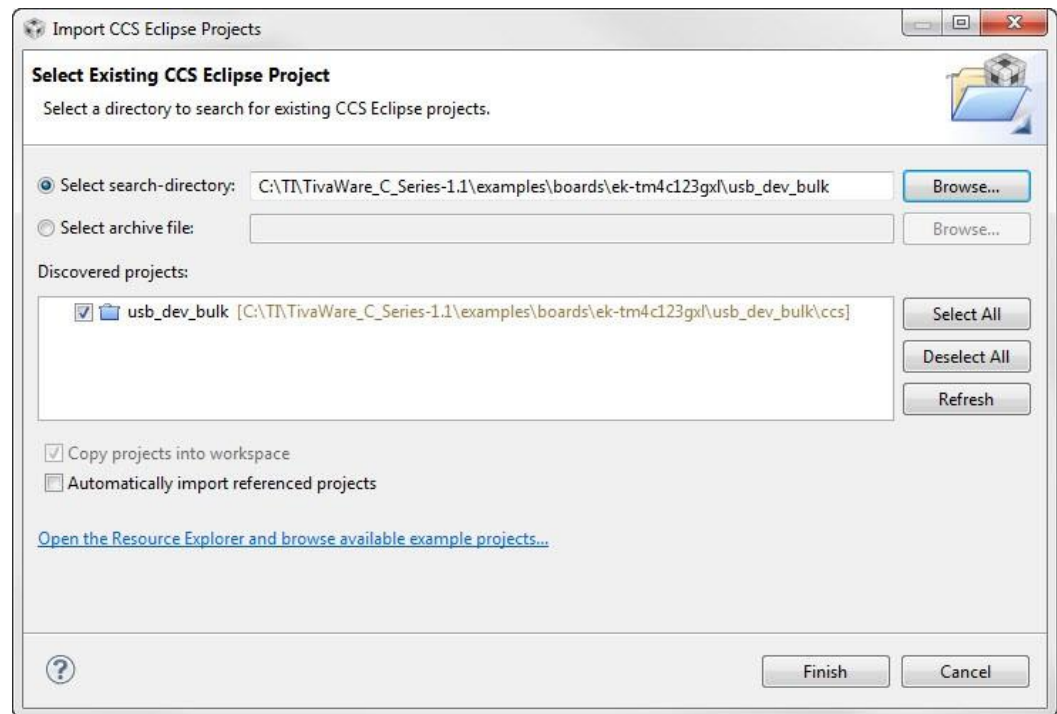
1. usb\_bulk\_exampleproject là một ví dụ mẫu của tivaware. Khi bạn nhập project, lưu ý rằng nó sẽ được tự động sao chép vào không gian làm việc của bạn, bảo quản các tập tin ban đầu. Nếu bạn muốn truy cập các tập tin của project thông qua Windows Explorer, file bạn đang làm việc trên trong thư mục không gian làm việc của bạn, không phải là thư mục TivaWare. Nếu bạn xóa các project trong CCS, các project nhập khẩu vẫn sẽ ở trong không gian làm việc của bạn trừ khi bạn nói với các hộp thoại để xóa các tập tin từ ổ đĩa.

► Chọn *Project* ☐ *Import Existing CCS*

*Eclipse Project.*

Tiến hành cài đặt hư hình dưới, sau đó

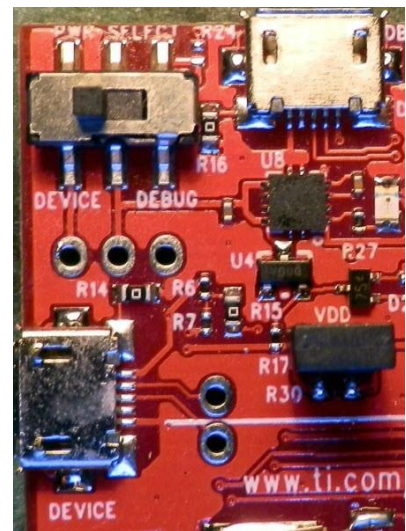
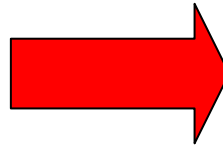
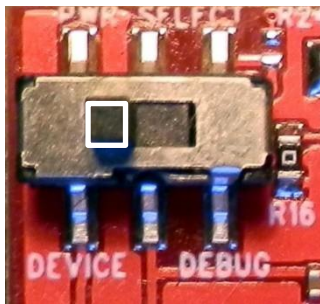
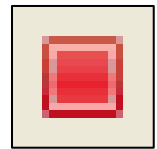
click ► *Finish*





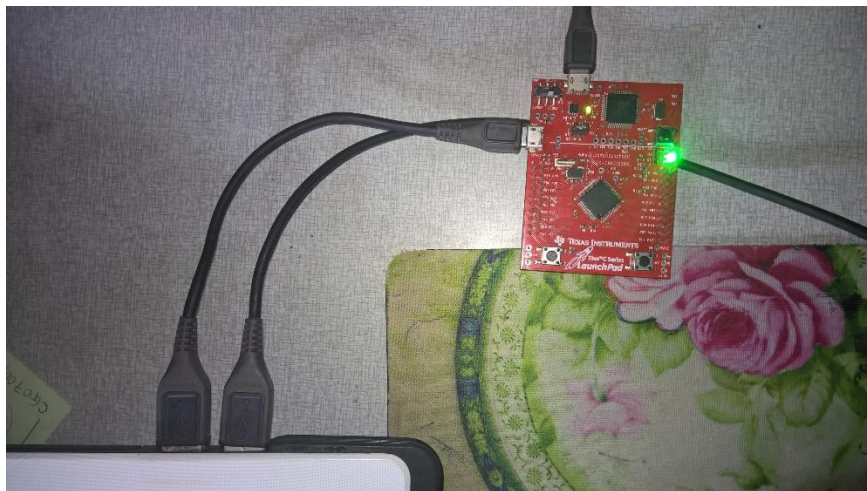
## Build, Download and Run The Code

2. hãy chắc chắn rằng kết nối của kit đang ở chế độ debug để kết nối với PC và usb\_dev\_bulk đã được cài đặt. Build và download chương trình bằng cách click nút debug ở thanh menu bar (hãy bấm nút SW2 để thoát khỏi chế độ hibernate từ bài lab trước). nếu bạn thấy cảnh báo việc chạy ứng dụng do phiên bản trước thì hãy bỏ qua cảnh báo.
3. ► hãy để kit ở chế độ debug. Gắn đồng thời 2 cổng trên kit với 2 cổng USB của máy tính bằng cáp USB



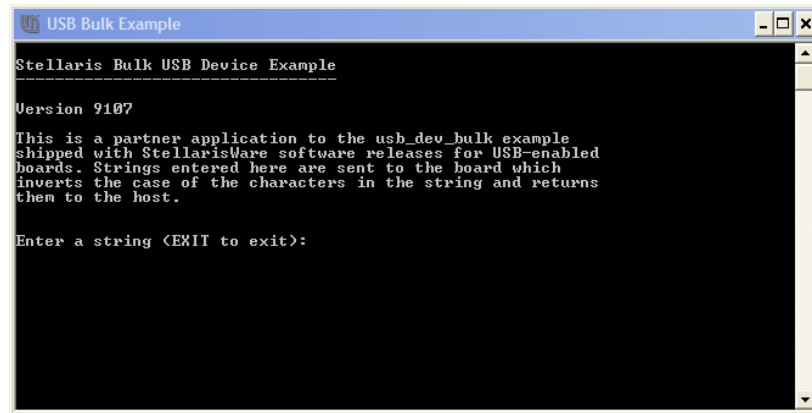
4. Đợi một lúc để máy tính nhận **generic bulk device** đã được cài qua cổng USB. ► nếu không nhận có thể cài driver từ đường dẫn sau:

C:\TI\TivaWare\_C\_Series-1.1\windows\_drivers



5. Hãy chắc chắn đã cài StellarisWare Windows-side USB examples  
Từ trang [www.ti.com/sw-usb-win](http://www.ti.com/sw-usb-win) . trong Windows, ► click  
*Start* □ *All Programs* □ *Texas Instruments* □ *Stellaris*  
□ *USB Examples* □ *USB Bulk Example*.

Trong window sẽ xuất hiện hình như sau:



6. ► trong Code Composer Studio, nếu usb\_dev\_bulk.c không có sẵn, expand usb\_dev\_bulkproject trong Project Explorer pane và double-click vào usb\_dev\_bulk.c để mở.

Chương trình có 5 lựa chọn:

**SysTickIntHandler** – an ISR that handles interrupts from the SysTick timer to

keep track of “time”.

**EchoNewDataToHost**– a routine that takes the received data from a buffer, flips the

case and sends it to the USB port for transmission.

**TxHandler**– an ISR that will report when the USB transmit

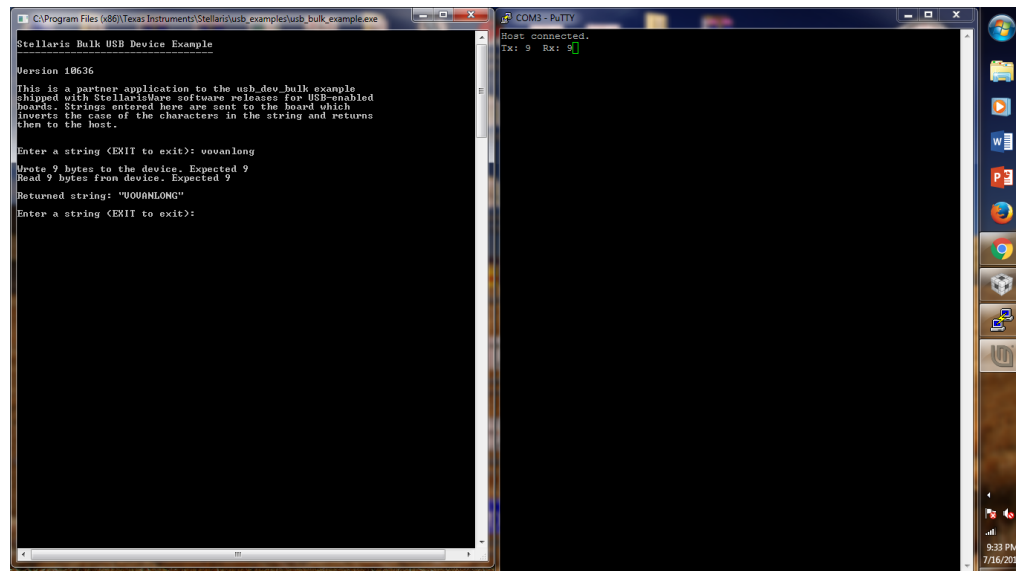
process is complete.

**RxHandler**— an ISR that handles the interaction with the incoming data, then calls the  
EchoNewDataHostroutine.

**main()**— chủ yếu là khởi tạo, nhưng một vòng lặp trong khi giữ một mắt trên các số byte chuyên chú ý rằng UARTprintf() APIs sprinkled throughout the code. This technique “instruments” the code, allowing us to monitor its status via a serial port.

## Watching the Instrumentation

7. ► chạy chương trình để kết nối với Stellaris Virtual Serial Port. Sắp xếp các cửa sổ thiết bị đầu cuối để nó chiếm không quá một phần tư của màn hình của bạn và vị trí của nó ở phía trên bên trái của màn hình của bạn.
8. ► Thay đổi kích thước CCS để nó chiếm nửa dưới của màn hình của bạn. ► Nhấp vào nút Debug để xây dựng và tải về mã và kết nối lại với LaunchPad của bạn. ► Khởi mã bằng cách nhấn vào nút Resume.
9. ► hãy bắt đầu USB Bulk Example Windows như bước 5. Đặt cửa sổ ở góc trên bên phải màn hình của bạn.
10. ► Lưu ý các trạng thái trên màn hình thiết bị đầu cuối của bạn và gõ một cái gì đó, giống như vovanlong vào USB Bulk và nhấn enter. Lưu ý rằng chương trình thiết bị đầu cuối sẽ hiển thị





# CHƯƠNG 8

## LAB 8: Bộ nhớ và MPU

### 1. Mục đích thí nghiệm :

- Ghi dữ liệu vào bộ nhớ FLASH in-system.
- Đọc/ ghi EEPROM
- Sử dụng MPU
- Bit-banding

### 2. Trình tự thực hiện

#### FLASH và EEPROM

- 2.1. Import project lab8
- 2.2. Mở file main.c để biên tập
- 2.3. Chèn đoạn code sau vào file main.c

```
#include <stdint.h>
#include <stdbool.h>
#include "inc/hw_types.h"
#include "inc/hw_memmap.h"
#include "driverlib/sysctl.h"
#include "driverlib/pin_map.h"
#include "driverlib/debug.h"
#include "driverlib/gpio.h"
int main(void)
{
    SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|
SYSCTL_XTAL_16MHZ|SYSCTL_OSC_MAIN);
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);
    GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE,
GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3);
    GPIOPinWrite(GPIO_PORTF_BASE,GPIO_PIN_1|GPIO_
PIN_2|GPIO_PIN_3, 0x00);
```

```

    SysCtlDelay(20000000);
    while(1)
    {
    }
}

```

Bên trong hàm main(), xung nhịp hệ thống được cấu hình ở 40MHz, các pin kết nối với 3 LED ở chế độ xuất, và các LED này đều ở trạng thái tắt, chờ khoảng 2s. Sau đó bước vào vòng lặp vô tận.

2.4. Build project: chỉ build project mà không download chương trình vào bộ nhớ của Tiva TM4C.

2.5. Mở file lab8.map

2.6. Section MEMORY CONFIGURATION và SEGMENT ALLOCATION MAP:

MEMORY CONFIGURATION						
name	origin	length	used	unused	attr	fill
FLASH	00000000	00040000	000007a6	0003f85a	R X	
SRAM	20000000	00008000	00000214	00007dec	RW X	

SEGMENT ALLOCATION MAP						
run origin	load origin	length	init length	attrs	members	
00000000	00000000	000007a8	000007a8	r-x		
00000000	00000000	0000026c	0000026c	r--	.intvecs	
0000026c	0000026c	0000051a	0000051a	r-x	.text	
00000788	00000788	00000020	00000020	r--	.cinit	
20000000	20000000	00000200	00000000	rw-		
20000000	20000000	00000200	00000000	rw-	.stack	
20000200	20000200	00000014	00000014	rw-		
20000200	20000200	00000014	00000014	rw-	.data	

Dung lượng bộ nhớ flash sử dụng là 0x07a8 bắt đầu ở 0x0.

2.7. Thêm dòng code sau vào phần *include header*:

```
#include "driverlib/flash.h"
```

2.8. Thêm phần code sau vào phần đầu của hàm main():



```
uint32_t pui32Data[2];
uint32_t pui32Read[2];
pui32Data[0] = 0x12345678;
pui32Data[1] = 0x56789abc;
```

Hai biến buffer để đọc và ghi chứa dữ liệu khởi tạo.

2.9. Thêm các dòng code sau vào vòng lặp while (1):

```
FlashErase(0x10000);
FlashProgram(pui32Data, 0x10000, sizeof(pui32Data));
GPIOPinWrite(GPIO_PORTF_BASE,GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, 0x02);
SysCtlDelay(20000000);
```

Dòng 1: Xóa dữ liệu bộ nhớ flash ở địa chỉ 0x10000

Dòng 2: Ghi dữ liệu vào bộ nhớ flash

Dòng 3: Bật LED đỏ

Dòng 4: Chờ khoảng 2s

2.10. Kiểm tra lại đoạn code

```
#include <stdint.h>
#include <stdbool.h>
#include "inc/hw_types.h"
#include "inc/hw_memmap.h"
#include "driverlib/sysctl.h"
#include "driverlib/pin_map.h"
#include "driverlib/debug.h"
#include "driverlib/gpio.h"
#include "driverlib/flash.h"
int main(void)
{
    uint32_t pui32Data[2];
    uint32_t pui32Read[2];
    pui32Data[0] = 0x12345678;
    pui32Data[1] = 0x56789abc;
    SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYS
```

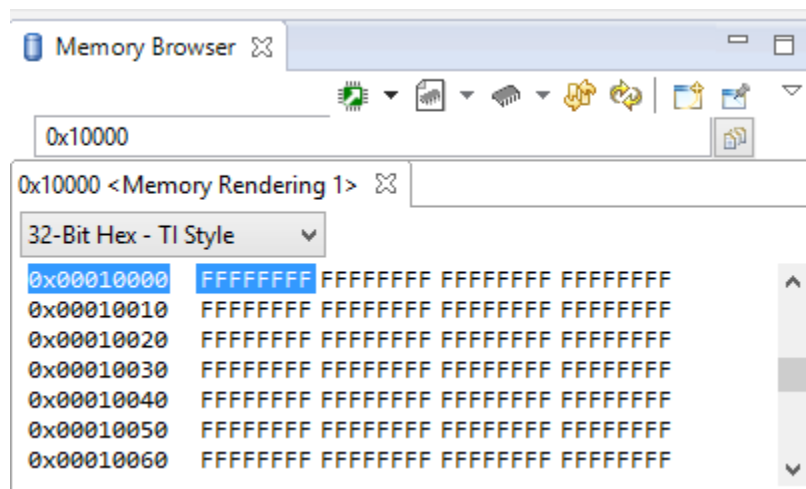
```

CTL_XTAL_16MHZ|SYSCTL_OSC_MAIN);
SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);
GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE,
GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3);
GPIOPinWrite(GPIO_PORTF_BASE,GPIO_PIN_1|GPIO_PIN_
2|GPIO_PIN_3, 0x00);
SysCtlDelay(20000000);
FlashErase(0x10000);
FlashProgram(pui32Data, 0x10000, sizeof(pui32Data));
GPIOPinWrite(GPIO_PORTF_BASE,GPIO_PIN_1|GPIO_PIN_
2|GPIO_PIN_3, 0x02);
SysCtlDelay(20000000);
while(1)
{
}
}

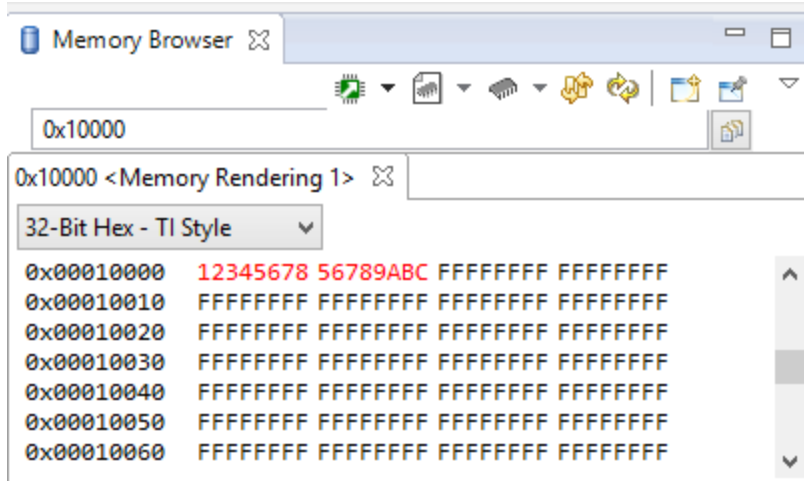
```

2.11. Build và nạp chương trình, đặt breakpoint tại  
FlashProgram().

2.12. Resume và kết quả trong Memory Browser tại địa chỉ  
0x10000:



2.13. Resume để tiếp tục chạy:



Dữ liệu trong các biến được ghi vào ở địa chỉ bắt đầu 0x10000, LED đỏ bật sáng.

2.14. Bỏ breakpoint

2.15. Terminate (Dừng)

2.16. Thêm dòng code sau vào *include header*:

```
#include "driverlib/eeprom.h"
```

2.17. Bên trong vòng lặp while (1), thêm vào các dòng code sau:

```
SysCtlPeripheralEnable(SYSCTL_PERIPH_EEPROM0);
EEPROMInit();
EEPROMMassErase();
EEPROMRead(pui32Read, 0x0, sizeof(pui32Read));
EEPROMProgram(pui32Data, 0x0, sizeof(pui32Data));
EEPROMRead(pui32Read, 0x0, sizeof(pui32Read));
GPIOPinWrite(GPIO_PORTF_BASE,GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, 0x04);
```

Dòng 1: Enable EEPROM

Dòng 3: Xóa toàn bộ EEPROM

Dòng 5: Ghi dữ liệu vào EEPROM

Dòng 7: Tắt LED đỏ, bật LED xanh dương

2.18. Lưu project.

2.19. Build và nạp chương trình.

2.20. Mở Variables tab ở CCS Debug

2.21. Đặt breakpoint tại EEPROMProgram()

2.22. Sau khi hàm `EEPROMMassErase()` được gọi, giá trị trong bộ nhớ sẽ chứa F-s:

(x)= Variables Expressions Breakpoints			
Name	Type	Value	Location
▲  pui32Data	unsigned int[2]	0x200001E8	0x200001E8
(x)= [0]	unsigned int	0x12345678 (Hex)	0x200001E8
(x)= [1]	unsigned int	0x56789ABC (Hex)	0x200001EC
▲  pui32Read	unsigned int[2]	0x200001F0	0x200001F0
(x)= [0]	unsigned int	0xFFFFFFFF (Hex)	0x200001F0
(x)= [1]	unsigned int	0xFFFFFFFF (Hex)	0x200001F4

2.23. Sau khi Resume, LED xanh dương bật sáng, và dữ liệu được cập nhật:

(x)= Variables Expressions Breakpoints			
Name	Type	Value	Location
▲  pui32Data	unsigned int[2]	0x200001E8	0x200001E8
(x)= [0]	unsigned int	0x12345678 (Hex)	0x200001E8
(x)= [1]	unsigned int	0x56789ABC (Hex)	0x200001EC
▲  pui32Read	unsigned int[2]	0x200001F0	0x200001F0
(x)= [0]	unsigned int	0x12345678 (Hex)	0x200001F0
(x)= [1]	unsigned int	0x56789ABC (Hex)	0x200001F4

2.24. Xóa breakpoint và dừng (terminate).

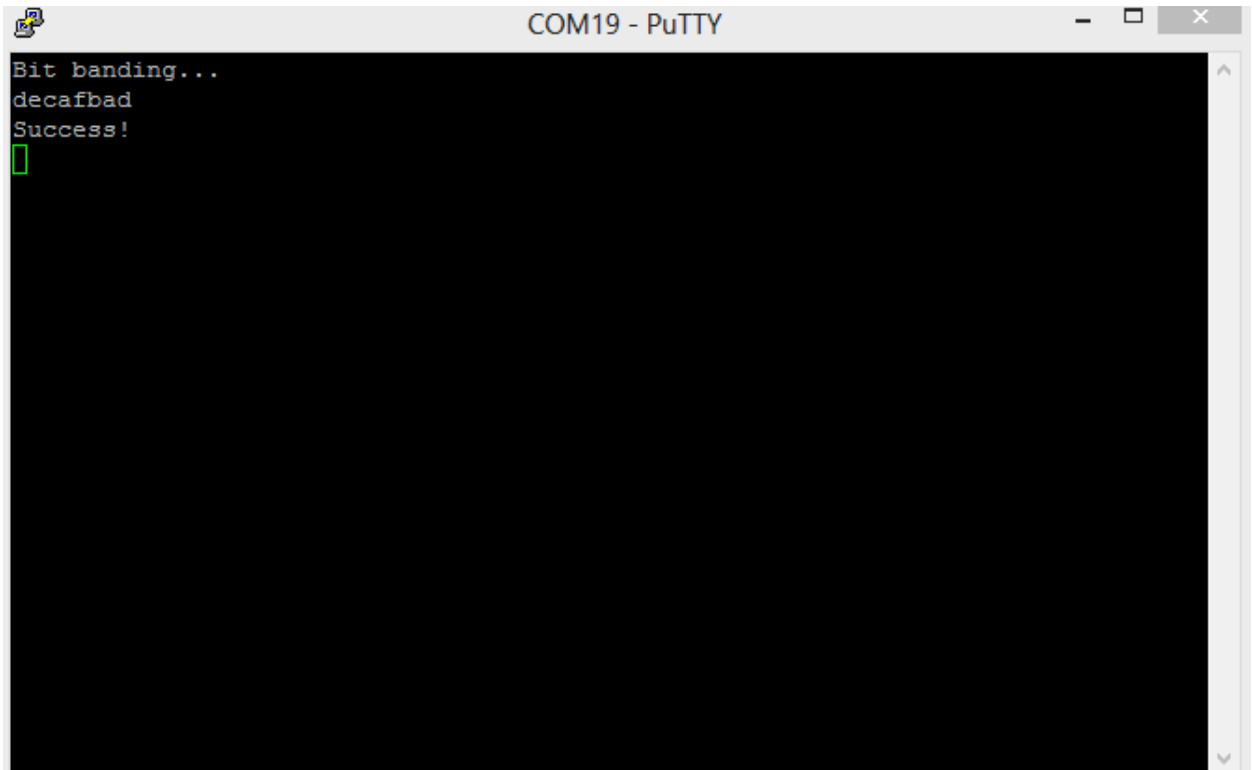
## BIT-BANDING

2.25. Import project bit-banding.

2.26. Mở file `bitband.c` : UART được sử dụng để xuất kết quả.

2.27. Build và nạp chương trình.

2.28. Click Resume, và kết quả:



```
COM19 - PuTTY
Bit banding...
decafbad
Success!
█
```

2.29. Terminate (Dừng).

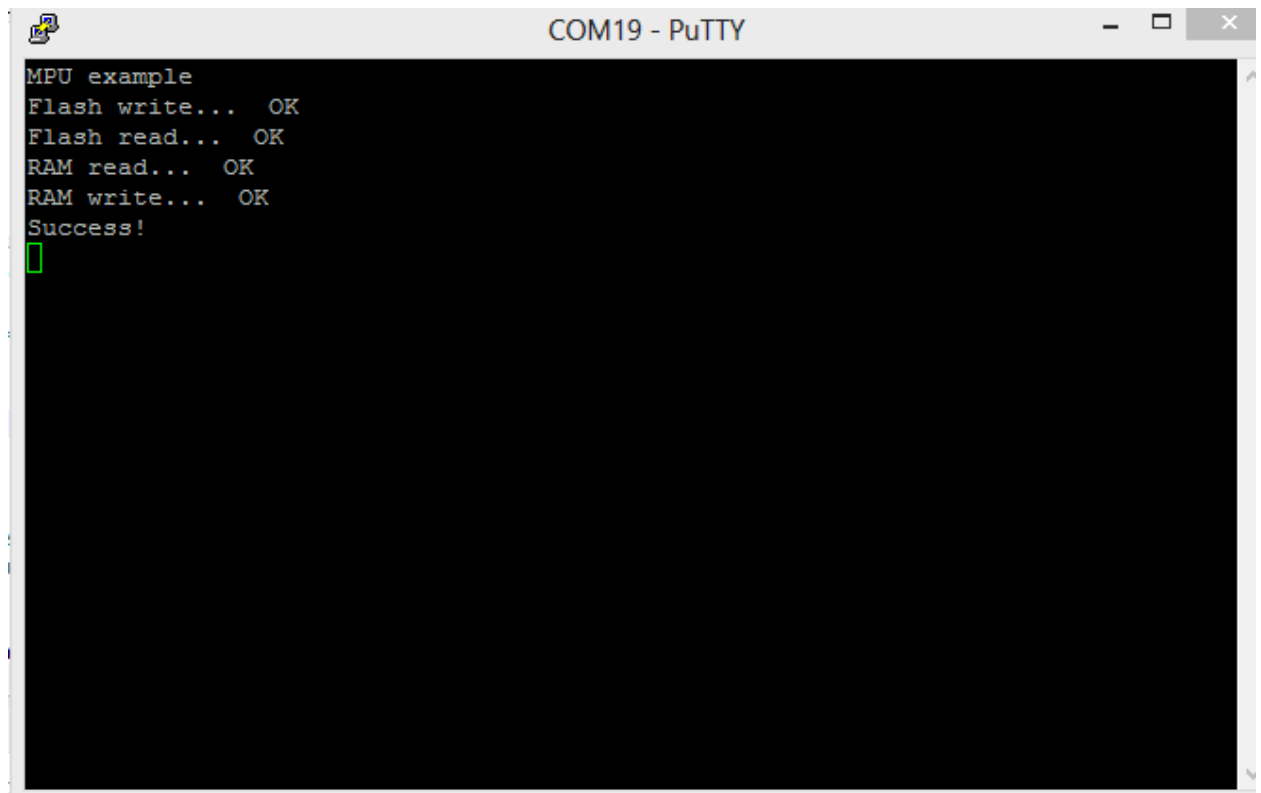
## **MEMORY PROTECTION UNIT (MPU)**

2.30. Import project mpu\_fault

2.31. Mở file mpu\_fault.c

2.32. Build và nạp chương trình.

2.33. Kết quả:



A screenshot of a PuTTY terminal window. The title bar at the top reads "COM19 - PuTTY". The terminal area has a black background with white text. The text displayed is as follows:

```
MPU example  
Flash write... OK  
Flash read... OK  
RAM read... OK  
RAM write... OK  
Success!  
█
```

The text "Success!" is followed by a green cursor symbol (a small rectangle) on the next line.

2.34. Terminate (Dừng).

# CHƯƠNG 9

## LAB 9: FPU

### 1. Mục đích

Tìm hiểu về FPU trên TM4C123G.

### 2. Trình tự thực hiện

2.1. Import project lab9

2.2. Mở file main.c, thêm vào đoạn code sau:

```
#include <stdint.h>
#include <stdbool.h>
#include <math.h>
#include "inc/hw_memmap.h"
#include "inc/hw_types.h"
#include "driverlib/fpu.h"
#include "driverlib/sysctl.h"
#include "driverlib/rom.h"
#ifndef M_PI
#define M_PI 3.14159265358979323846
#endif
#define SERIES_LENGTH 100
float gSeriesData[SERIES_LENGTH];
int32_t i32DataCount = 0;
int main(void)
{
    float fRadians;
    ROM_FPULazyStackingEnable();
    ROM_FPUEnable();
    ROM_SysCtlClockSet(SYSCTL_SYSDIV_4 |
        SYSCTL_USE_PLL | SYSCTL_XTAL_16MHZ |
        SYSCTL_OSC_MAIN);
    fRadians = ((2 * M_PI) / SERIES_LENGTH);
    while(i32DataCount < SERIES_LENGTH)
```

```

{
gSeriesData[i32DataCount] = sinf(fRadians * i32DataCount);
i32DataCount++;
}
while(1)
{
}
}

```

2.3. Bên trong hàm main ():

Khai báo một biến có kiểu *float* để tính giá trị *sin*.

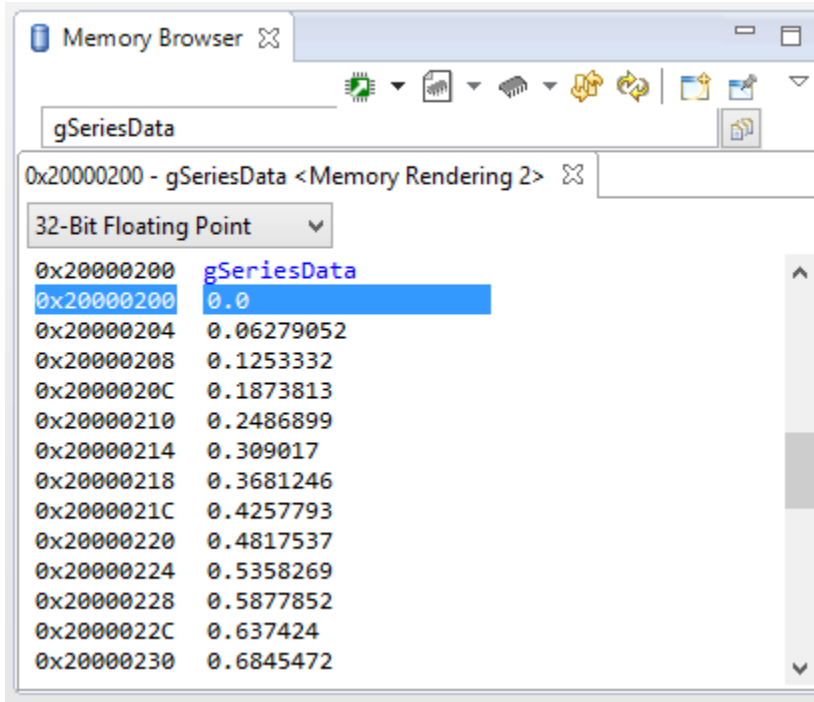
Enable Lazy Stacking.

Cấu hình xung nhịp hệ thống 50MHz.

Vòng lặp while tính giá trị của sin và lưu vào mảng.

2.4. Build và nạp chương trình.

2.5. Dữ liệu của biến gSeriesData trong cửa sổ Memory Browser:



2.6. Dùng công cụ Graph:





# CHƯƠNG 10

## LAB 12: UART

### 1. Mục đích thí nghiệm.

Thiết lập và gửi nhận dữ liệu UART dùng polling và interrupt.

### 2. Trình tự thực hiện.

#### **Polling**

2.1. Import project lab12

2.2. Mở file main.c

```
#include <stdint.h>
#include <stdbool.h>
#include "inc/hw_memmap.h"
#include "inc/hw_types.h"
#include "driverlib/gpio.h"
#include "driverlib/pin_map.h"
#include "driverlib/sysctl.h"
#include "driverlib/uart.h"
int main(void) {
    SysCtlClockSet(SYSCTL_SYSDIV_4 | SYSCTL_USE_PLL |
SYSCTL_OSC_MAIN | SYSCTL_XTAL_16MHZ);
    SysCtlPeripheralEnable(SYSCTL_PERIPH_UART0);
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOA);
    GPIOPinConfigure(GPIO_PA0_U0RX);
    GPIOPinConfigure(GPIO_PA1_U0TX);
    GPIOPinTypeUART(GPIO_PORTA_BASE, GPIO_PIN_0 |
GPIO_PIN_1);
    UARTConfigSetExpClk(UART0_BASE, SysCtlClockGet(),
115200,
(UART_CONFIG_WLEN_8 | UART_CONFIG_STOP_ONE |
UART_CONFIG_PAR_NONE));
    UARTCharPut(UART0_BASE, 'E');
    UARTCharPut(UART0_BASE, 'n');
```

```
UARTCharPut(UART0_BASE, 't');
UARTCharPut(UART0_BASE, 'e');
UARTCharPut(UART0_BASE, 'r');
UARTCharPut(UART0_BASE, ' ');
UARTCharPut(UART0_BASE, 'T');
UARTCharPut(UART0_BASE, 'e');
UARTCharPut(UART0_BASE, 'x');
UARTCharPut(UART0_BASE, 't');
UARTCharPut(UART0_BASE, ':');
UARTCharPut(UART0_BASE, ' ');
while (1)
{
if (UARTCharsAvail(UART0_BASE))
UARTCharPut(UART0_BASE,
UARTCharGet(UART0_BASE));
}
}
```

2.3. Trong hàm main():

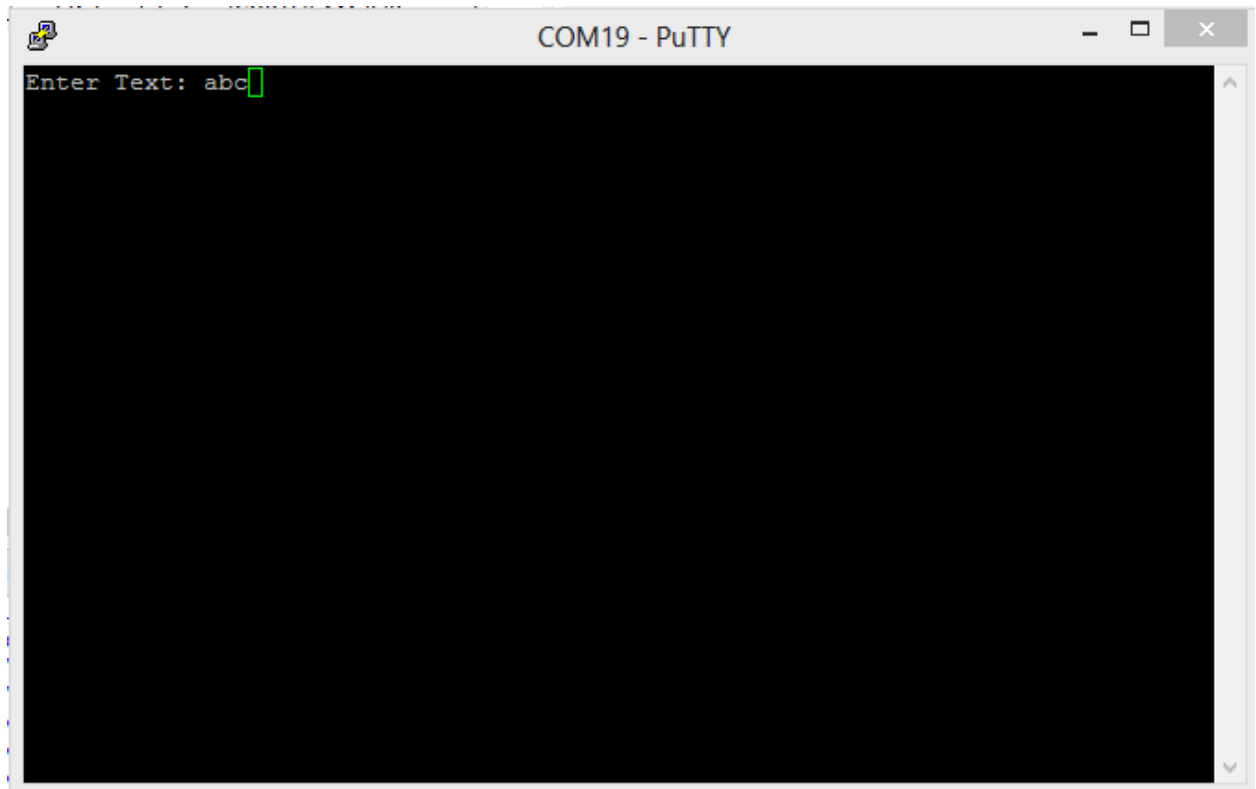
Cấu hình xung nhịp hệ thống.

Cấu hình các chân truyền và nhận UART.

Cấu hình tham số UART: 115200, 8-1-N.

2.4. Build và nạp chương trình.

2.5. Kết quả:



## Interrupt

Thêm đoạn code sau vào phần *include header*:

```
#include "inc/hw_ints.h"  
#include "driverlib/interrupt.h"
```

2.6. Thêm vào đoạn code sau ngay sau hàm

**UARTConfigSetExpClk():**

```
IntMasterEnable();  
IntEnable(INT_UART0);  
UARTIntEnable(UART0_BASE, UART_INT_RX |  
UART_INT_RT);
```

2.7. Cấu hình GPIO pin cho LED

```
SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);  
GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE,
```

```
GPIO_PIN_2);
```

2.8. Bên trong vòng lặp while, xóa bỏ dòng:

```
if  
(UARTCharsAvail(UART0_BASE))UARTCharPut(UART0_B  
ASE,UARTCharGet(UART0_BASE));
```

2.9. Lưu lại file main.c.

2.10. Thêm vào đoạn code sau để xử lý ngắt UART:

```
void UARTIntHandler(void)  
{  
uint32_t ui32Status;  
ui32Status = UARTIntStatus(UART0_BASE, true); //get  
interrupt status  
UARTIntClear(UART0_BASE, ui32Status); //clear the  
asserted interrupts  
while(UARTCharsAvail(UART0_BASE)) //loop while there  
are chars  
{  
UARTCharPutNonBlocking(UART0_BASE,  
UARTCharGetNonBlocking(UART0_BASE));  
//echo character  
GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_2,  
GPIO_PIN_2); //blink LED  
SysCtlDelay(SysCtlClockGet() / (1000 * 3)); //delay ~1 msec  
GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_2, 0);  
//turn off LED  
}  
}
```

2.11. Mở file startup\_ccs.c để thêm vào khai báo trình xử lý  
UART:

```
extern void UARTIntHandler(void);
```

2.12. Thay đổi trình xử lý ngắt mặc định bằng trình xử lý ngắt  
UART ở UART0 interrupt vector:

## UARTIntHandler, // UART0 Rx and Tx

2.13. Lưu lại và file main.c có nội dung như sau:

```
#include <stdint.h>
#include <stdbool.h>
#include "inc/hw_ints.h"
#include "inc/hw_memmap.h"
#include "inc/hw_types.h"
#include "driverlib/gpio.h"
#include "driverlib/interrupt.h"
#include "driverlib/pin_map.h"
#include "driverlib/sysctl.h"
#include "driverlib/uart.h"

void UARTIntHandler(void)
{
    uint32_t ui32Status;
    ui32Status = UARTIntStatus(UART0_BASE, true); //get
    interrupt status
    UARTIntClear(UART0_BASE, ui32Status); //clear the asserted
    interrupts
    while(UARTCharsAvail(UART0_BASE)) //loop while there are
    chars
    {
        UARTCharPutNonBlocking(UART0_BASE,
        UARTCharGetNonBlocking(UART0_BASE)); //echo character
        GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_2,
        GPIO_PIN_2); //blink LED
        SysCtlDelay(SysCtlClockGet() / (1000 * 3)); //delay ~1 msec
        GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_2, 0); //turn
        off LED
    }
}

int main(void) {
    SysCtlClockSet(SYSCTL_SYSDIV_4 | SYSCTL_USE_PLL |
    SYSCTL_OSC_MAIN | SYSCTL_XTAL_16MHZ);
```

```

SysCtlPeripheralEnable(SYSCTL_PERIPH_UART0);
SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOA);
GPIOPinConfigure(GPIO_PA0_U0RX);
GPIOPinConfigure(GPIO_PA1_U0TX);
GPIOPinTypeUART(GPIO_PORTA_BASE, GPIO_PIN_0 |
GPIO_PIN_1);
SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF); //enable
GPIO port for LED
GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE,
GPIO_PIN_2); //enable pin for LED PF2
UARTConfigSetExpClk(UART0_BASE, SysCtlClockGet(),
115200,
(UART_CONFIG_WLEN_8 | UART_CONFIG_STOP_ONE |
UART_CONFIG_PAR_NONE));
IntMasterEnable(); //enable processor interrupts
IntEnable(INT_UART0); //enable the UART interrupt
UARTIntEnable(UART0_BASE, UART_INT_RX |
UART_INT_RT); //only enable RX and TX interrupts
UARTCharPut(UART0_BASE, 'E');
UARTCharPut(UART0_BASE, '\n');
UARTCharPut(UART0_BASE, 't');
UARTCharPut(UART0_BASE, 'e');
UARTCharPut(UART0_BASE, 'r');
UARTCharPut(UART0_BASE, ' ');
UARTCharPut(UART0_BASE, 'T');
UARTCharPut(UART0_BASE, 'e');
UARTCharPut(UART0_BASE, 'x');
UARTCharPut(UART0_BASE, 't');
UARTCharPut(UART0_BASE, ':');
UARTCharPut(UART0_BASE, ' ');
while (1) //let interrupt handler do the UART echo function
{
// if (UARTCharsAvail(UART0_BASE))
UARTCharPut(UART0_BASE, UARTCharGet(UART0_BASE));
}

```

}

2.14. Build và nạp chương trình.

2.15. Kết quả: ký tự nhập vào hiển thị trên terminal và LED nhấp khi nhận được dữ liệu.

