Lab 3: TM4C123 Microcontroller I/O Ports
Switches and LED Interfaces

Nima Karimian

Preparation

- ➢ Review the content of lecture 5, 6
- ➢ Before you start the lab, make sure you have completed Lab 1 and installed Keil uVision, TI's TivaWare Development Kit, and windows drivers for the LaunchPad on the computer. You will need the actual physical LaunchPad to complete this lab.

References

- ➢ Getting Started with the Tiva TM4C123G LaunchPad Workshop Student Guide and Lab Manual (Canvas -> Reference Materials -> TM4C123G_LaunchPad_Workshop_Workbook.pdf)
- ➢ TivaWare Peripheral Driver Library User's Guide (Canvas-> Reference Materials -> SW-TM4C-DRL-UG-2.1.0.12573.pdf)
- ➢ Tiva TM4C123GH6PM Microcontroller Data Sheet (Canvas-> Reference Materials)

Purpose

The general purpose of this lab is to familiarize you with the TM4C123 microcontroller and develop C programs to configure and access the parallel ports on the microcontroller by using the TivaWare Peripheral Driver Library. Software skills you will learn include flow chart, bit-masking operations, toggling, if-then, looping, delay, and writing friendly software.

Demonstration and Submission

You will have one week to complete the lab. You can discuss with your group members and complete the lab work together. Every group will need to write and submit a lab report to Can->Labs->Lab3 report submission. The lab report should include

- The students' names, emails and IDs
- The answers of all the questions asked in the lab handout
- Discussion and suggestions: Through your lab experiments, what have you learned? Do you have any suggestions for future labs or improvement on the learning experiences?

If you finish the lab experiments during the lab time, please demonstrate your results to the instructor. The instructor may ask questions regarding your program. After the demonstration, you can leave. **The latest demonstration time will be the beginning of next lab. The lab report is due at 6pm on the day of your next lab.**

Again, you can work with your group members on all the lab activities, but make sure you understand all the materials.

Adaptive from Dr. Xiaorong Zhang

Procedure

This lab provides a step-by-step tutorial which will familiarize you with the Keil μVision integrated development environment (IDE) and teach you how to create a new project, compile, download, and run the project on the launchpad. The tutorial provides two examples on how to initialize and configure the GPIO ports on the microcontroller to interface with the switches and LEDs on the launchpad. You will also learn how to use the functions provided in the TivaWare Peripheral Driver Library to simplify the development process.

After you go through the tutorial, please answer all the questions listed at the end of the handout.

Tutorial: Getting Started with Keil µVision; Two Examples

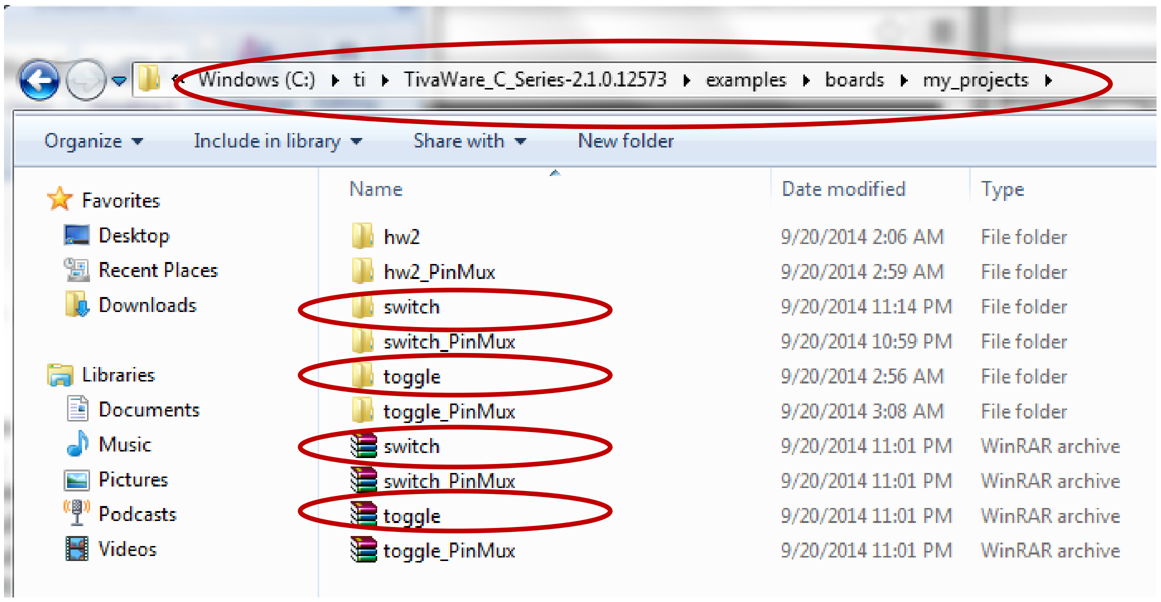
Procedure

Plug in the LaunchPad into a USB port of your computer. Turn the LaunchPad power on. Make sure you have installed all the software tools and drivers listed in the "Preparation" section of this document.

A. Start the Keil µVision IDE and Run the Sample Projects

1. Download the sample projects from Canvas (Canvas->Labs->sample_projects). There are two sample projects: *switch* and *toggle*. *switch* develops a system which interfaces with the blue LED and the switch SW2 on the launchPad. When SW2 is pressed, the LED is turned on. When SW2 is released, the LED is turned off. *toggle* implements a system that toggles the on-board red LED repeatedly.

2. The downloaded sample projects must be saved in a specific directory in order to run correctly. It is assumed you already install the TivaWare library in C:\ti\TivaWare_C_Series-2.x.x.xxxxx. First create a new directory "my_projects" (you may have already created it in Lab 2) under C:\ti\TivaWare_C_Series-2.x.x.xxxxx \examples\boards and then save and decompress the sample projects under "C:\ti\TivaWare_C_Series-2.x.x.xxxxx\examples\boards\my_projects" as shown in Figure 1.

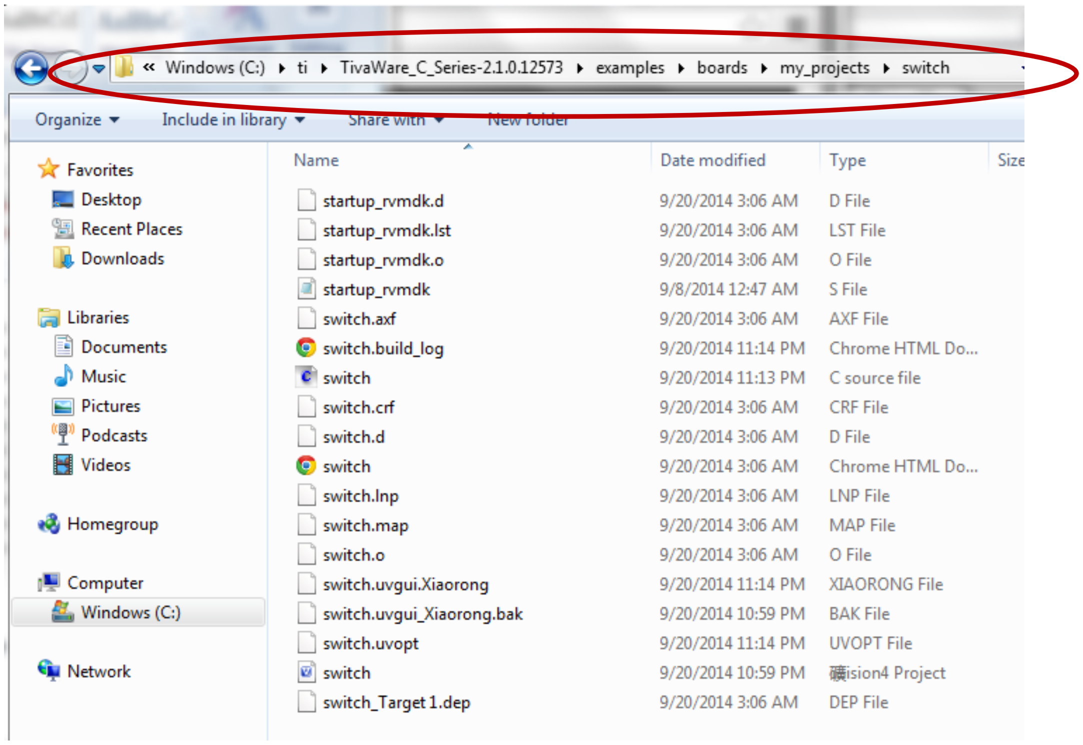


Windows (C:) ▸ ti ▸ TivaWare_C_Series-2.1.0.12573 ▸ examples ▸ boards ▸ my_projects ▸

Name	Date modified	Type
hw2	9/20/2014 2:06 AM	File folder
hw2_PinMux	9/20/2014 2:59 AM	File folder
switch	9/20/2014 11:14 PM	File folder
switch_PinMux	9/20/2014 10:59 PM	File folder
toggle	9/20/2014 2:56 AM	File folder
toggle_PinMux	9/20/2014 3:08 AM	File folder
switch	9/20/2014 11:01 PM	WinRAR archive
switch_PinMux	9/20/2014 11:01 PM	WinRAR archive
toggle	9/20/2014 11:01 PM	WinRAR archive
toggle_PinMux	9/20/2014 11:01 PM	WinRAR archive

Figure 1

Again, make sure you save them in the correct directory. If you double click the *switch* folder, it will look something like this as shown in Figure 2. Make sure the directory is C:\ti\TivaWare_C_Series-2.x.x.xxxxx\examples\boards\my_projects\switch\, not C:\ti\TivaWare_C_Series-2.x.x.xxxxx\examples\boards\my_projects\switch\switch\.



« Windows (C:) ▸ ti ▸ TivaWare_C_Series-2.1.0.12573 ▸ examples ▸ boards ▸ my_projects ▸ switch

Name	Date modified	Type	Size
startup_rvmdk.d	9/20/2014 3:06 AM	D File	
startup_rvmdk.lst	9/20/2014 3:06 AM	LST File	
startup_rvmdk.o	9/20/2014 3:06 AM	O File	
startup_rvmdk	9/8/2014 12:47 AM	S File	
switch.axf	9/20/2014 3:06 AM	AXF File	
switch.build_log	9/20/2014 11:14 PM	Chrome HTML Do...	
switch	9/20/2014 11:13 PM	C source file	
switch.crf	9/20/2014 3:06 AM	CRF File	
switch.d	9/20/2014 3:06 AM	D File	
switch	9/20/2014 3:06 AM	Chrome HTML Do...	
switch.lnp	9/20/2014 3:06 AM	LNP File	
switch.map	9/20/2014 3:06 AM	MAP File	
switch.o	9/20/2014 3:06 AM	O File	
switch.uvgui.Xiaorong	9/20/2014 11:14 PM	XIAORONG File	
switch.uvgui_Xiaorong.bak	9/20/2014 10:59 PM	BAK File	
switch.uvopt	9/20/2014 11:14 PM	UVOPT File	
switch	9/20/2014 10:59 PM	礦ision4 Project	
switch_Target 1.dep	9/20/2014 3:06 AM	DEP File	

Figure 2

3. Start the Keil µVision IDE by double-clicking the icon on your desktop or by selecting it from the Windows Start Menu. When the IDE loads, it was a blank screen (see Figure 3).

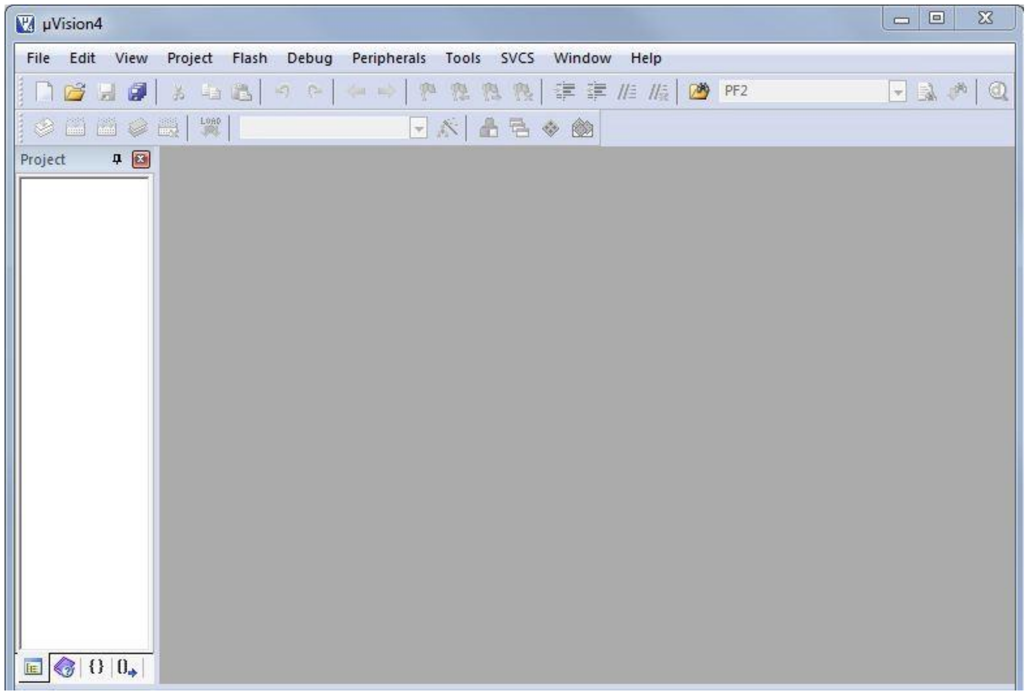


Figure 3

4. From the Project menu, select Open Project… (See Figure 4)

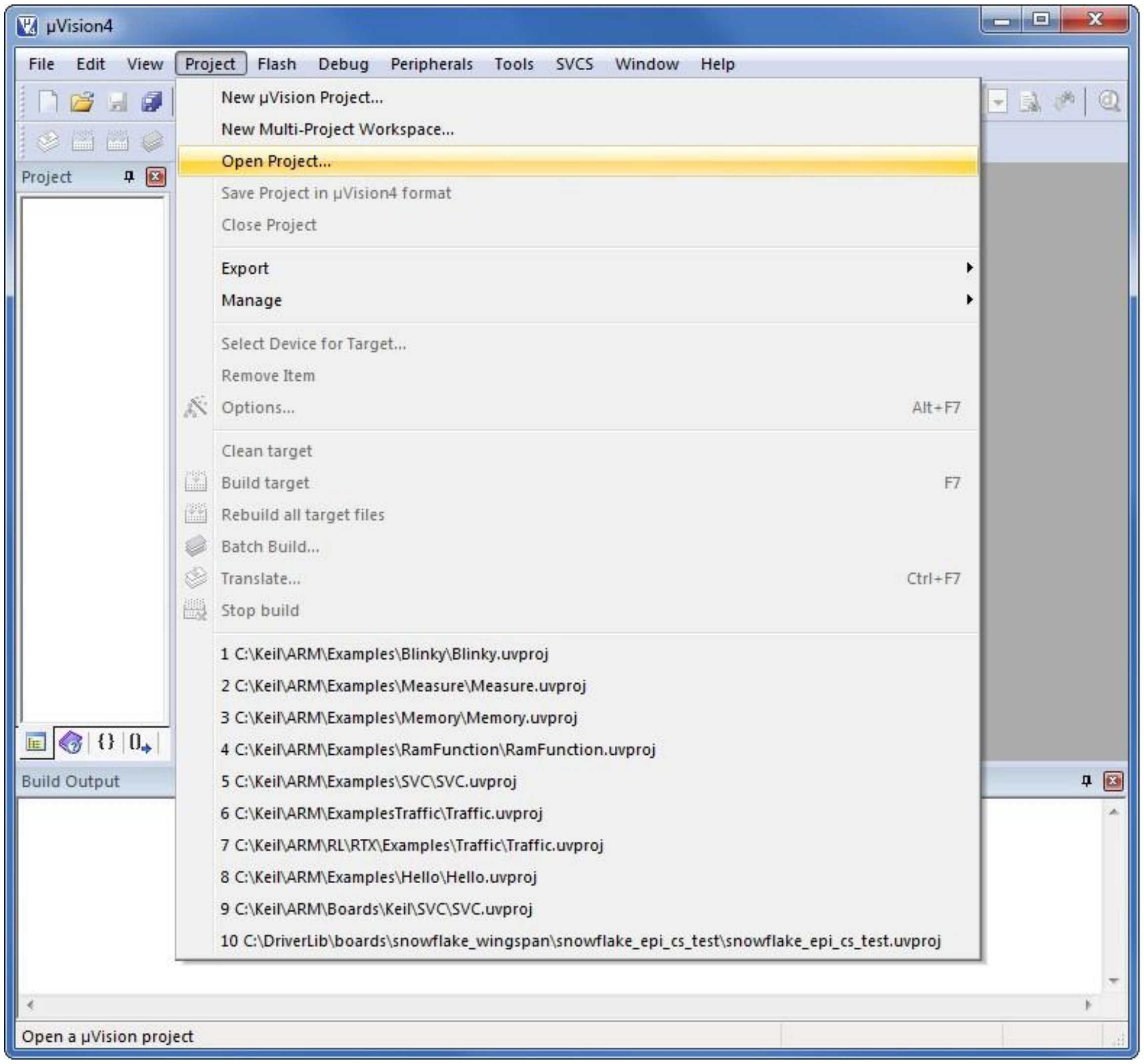
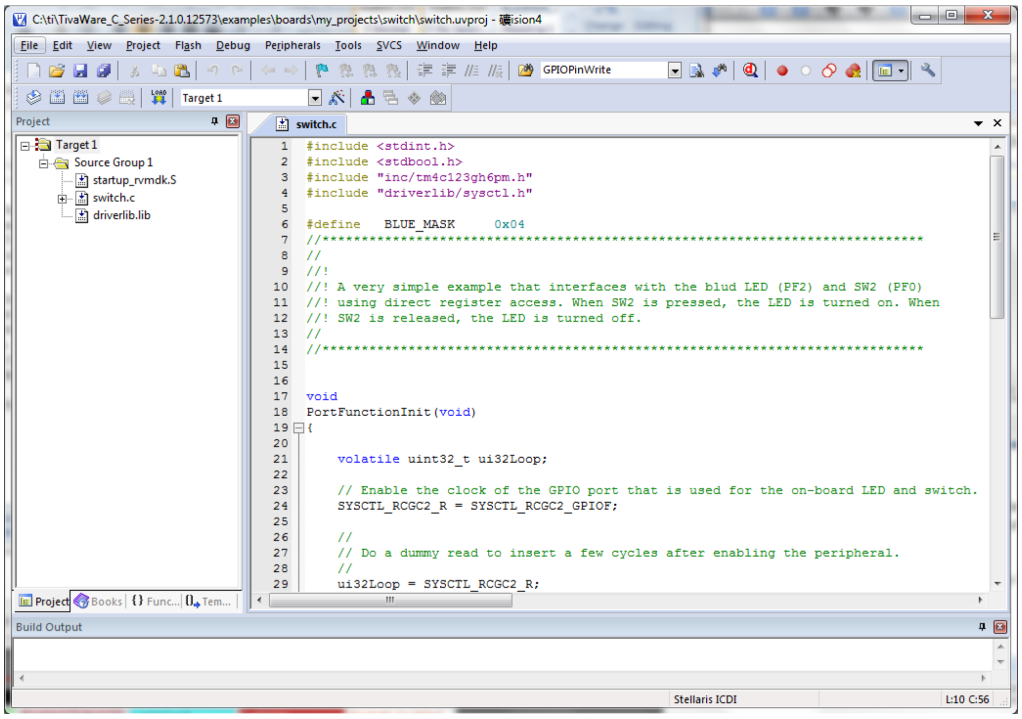

µVision4
File
Edit
View
Project
Flash
Debug
Peripherals
Tools
SVCS
Window
Help
New µVision Project...
New Multi-Project Workspace...
Open Project...
Save Project in µVision4 format
Close Project
Export
Manage
Select Device for Target...
Remove Item
Options... Alt+F7
Clean target
Build target F7
Rebuild all target files
Batch Build...
Translate... Ctrl+F7
Stop build
1 C:\Keil\ARM\Examples\Blinky\Blinky.uvproj
2 C:\Keil\ARM\Examples\Measure\Measure.uvproj
3 C:\Keil\ARM\Examples\Memory\Memory.uvproj
4 C:\Keil\ARM\Examples\RamFunction\RamFunction.uvproj
5 C:\Keil\ARM\Examples\SVC\SVC.uvproj
6 C:\Keil\ARM\ExamplesTraffic\Traffic.uvproj
7 C:\Keil\ARM\RL\RTX\Examples\Traffic\Traffic.uvproj
8 C:\Keil\ARM\Examples\Hello\Hello.uvproj
9 C:\Keil\ARM\Boards\Keil\SVC\SVC.uvproj
10 C:\DriverLib\boards\snowflake_wingspan\snowflake_epi_cs_test\snowflake_epi_cs_test.uvproj
Project
Build Output
Open a µVision project

Figure 4

5. Use the dialog box to navigate to the *switch* project. From the location where you installed TivaWare, the *switch* project should be located in: *C:\ti\TivaWare_C_Series-2.x.x.xxxxx\examples\boards\my_projects\switch*

6. Select the switch.uvproj project file and click Open. The project opens in the IDE (see Figure 5).


C:\ti\TivaWare_C_Series-2.1.0.12573\examples\boards\my_projects\switch\switch.uvproj - 礦ision4
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
GPIOPinWrite
Target 1
Project
Target 1
Source Group 1
startup_rvmdk.S
switch.c
driverlib.lib
switch.c
Project Books Func... Tem...
Build Output
Stellaris ICDI
L:10 C:56

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

#define   BLUE_MASK      0x04
//*****************************************************************************
//
//!
//! A very simple example that interfaces with the blud LED (PF2) and SW2 (PF0)
//! using direct register access. When SW2 is pressed, the LED is turned on. When
//! SW2 is released, the LED is turned off.
//
//*****************************************************************************


void
PortFunctionInit(void)
{

    volatile uint32_t ui32Loop;

    // Enable the clock of the GPIO port that is used for the on-board LED and switch.
    SYSCTL_RCGC2_R = SYSCTL_RCGC2_GPIOF;

    //
    // Do a dummy read to insert a few cycles after enabling the peripheral.
    //
    ui32Loop = SYSCTL_RCGC2_R;
```

Figure 5

7. Build the project. Select Project → Rebuild all target files, or click the Rebuild all button (icon) (see Figure 6).

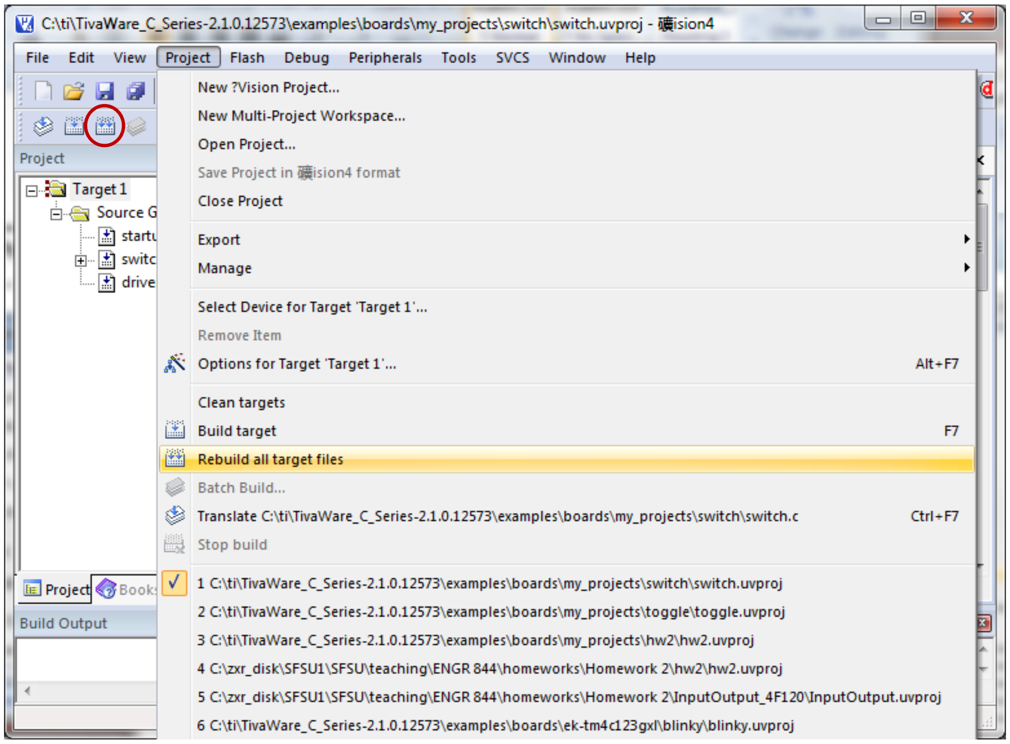
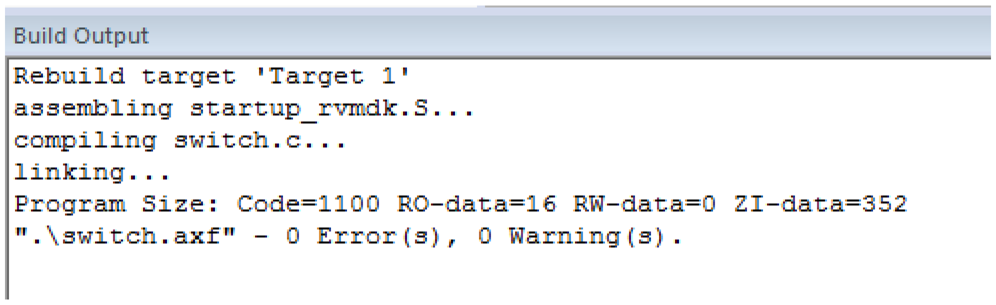

C:\ti\TivaWare_C_Series-2.1.0.12573\examples\boards\my_projects\switch\switch.uvproj - 礦ision4
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
New ?Vision Project...
New Multi-Project Workspace...
Open Project...
Save Project in 礦ision4 format
Close Project
Export
Manage
Select Device for Target 'Target 1'...
Remove Item
Options for Target 'Target 1'... Alt+F7
Clean targets
Build target F7
Rebuild all target files
Batch Build...
Translate C:\ti\TivaWare_C_Series-2.1.0.12573\examples\boards\my_projects\switch\switch.c Ctrl+F7
Stop build
1 C:\ti\TivaWare_C_Series-2.1.0.12573\examples\boards\my_projects\switch\switch.uvproj
2 C:\ti\TivaWare_C_Series-2.1.0.12573\examples\boards\my_projects\toggle\toggle.uvproj
3 C:\ti\TivaWare_C_Series-2.1.0.12573\examples\boards\my_projects\hw2\hw2.uvproj
4 C:\zxr_disk\SFSU1\SFSU\teaching\ENGR 844\homeworks\Homework 2\hw2\hw2.uvproj
5 C:\zxr_disk\SFSU1\SFSU\teaching\ENGR 844\homeworks\Homework 2\InputOutput_4F120\InputOutput.uvproj
6 C:\ti\TivaWare_C_Series-2.1.0.12573\examples\boards\ek-tm4c123gxl\blinky\blinky.uvproj
Project
Target 1
Source G
startu
switc
drive
Project Books
Build Output

Figure 6

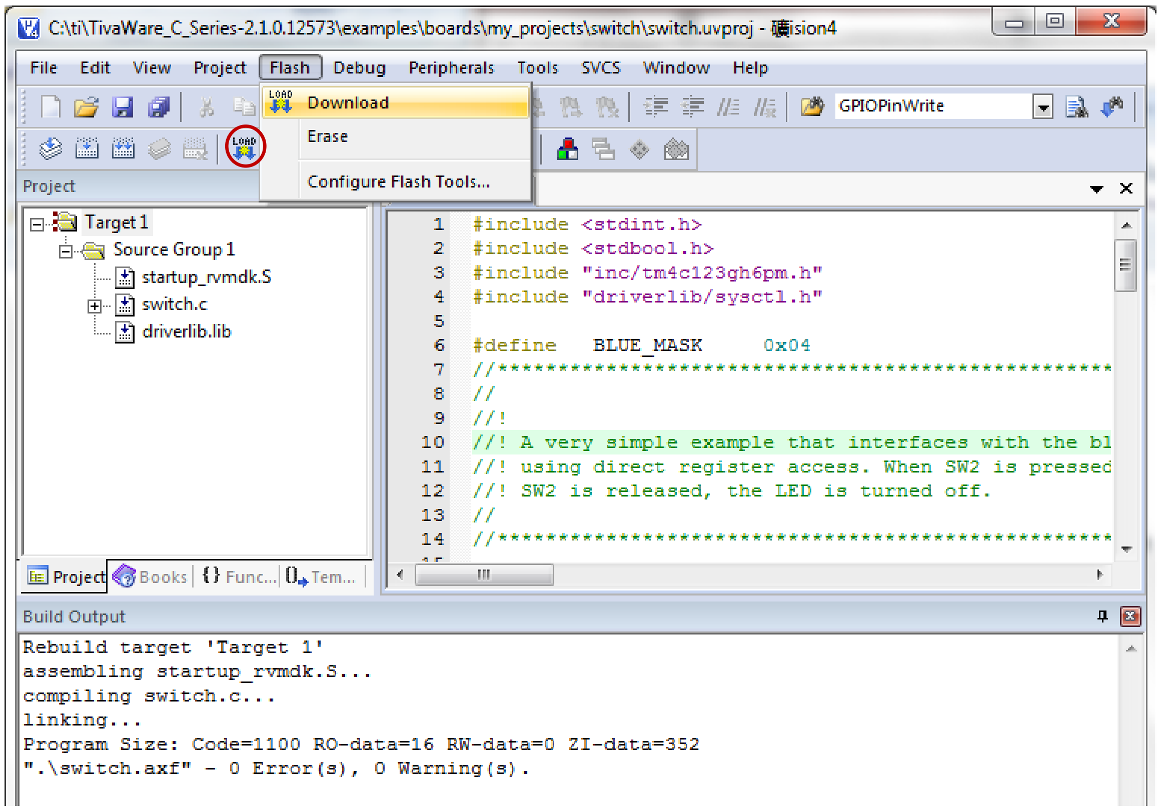
All of the source files are compiled and linked. The activity can be seen in the Build window at the bottom of the µVision IDE. The process completes with an application named *switch.axf* built with no errors and no warnings (see Figure 7).



```
Rebuild target 'Target 1'
assembling startup_rvmdk.S...
compiling switch.c...
linking...
Program Size: Code=1100 RO-data=16 RW-data=0 ZI-data=352
".\switch.axf" - 0 Error(s), 0 Warning(s).
```

Figure 7

8. Load the program into the flash memory (make sure you have plugged in the LaunchPad). Select Download from the Flash menu, or click the Download button (icon) (see Figure 8).



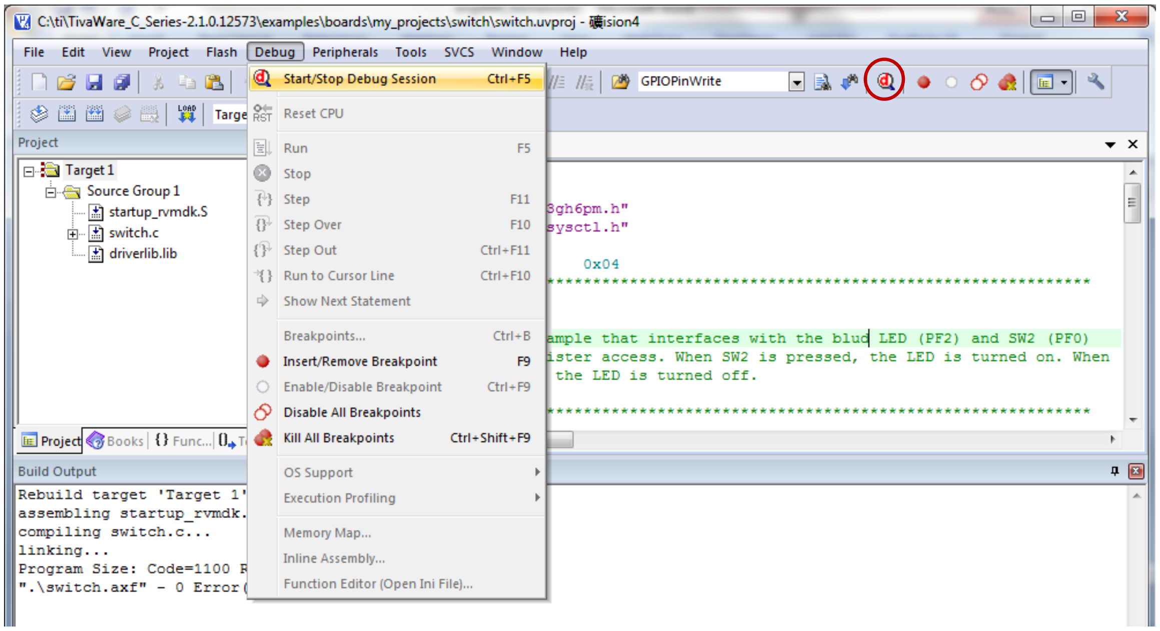
```
#include <stdint.h>
#include <stdbool.h>
#include "inc/tm4c123gh6pm.h"
#include "driverlib/sysctl.h"

#define    BLUE_MASK      0x04
//*****************************************************
//
//!
//! A very simple example that interfaces with the bl
//! using direct register access. When SW2 is pressed
//! SW2 is released, the LED is turned off.
//
//*****************************************************
```

```
Rebuild target 'Target 1'
assembling startup_rvmdk.S...
compiling switch.c...
linking...
Program Size: Code=1100 RO-data=16 RW-data=0 ZI-data=352
".\switch.axf" - 0 Error(s), 0 Warning(s).
```

Figure 8

The process takes a few seconds. A progress bar will show at the bottom of the IDE window as the device is programmed. When it is finished, the Build window will show that the device was erased, programmed, and verified OK. The project is now programmed into the flash memory of the microcontroller on the LaunchPad.

9. Debug and run the program. Select Start/Stop Debug Session from the Debug menu or click the Debug button (icon) (see Figure 9).

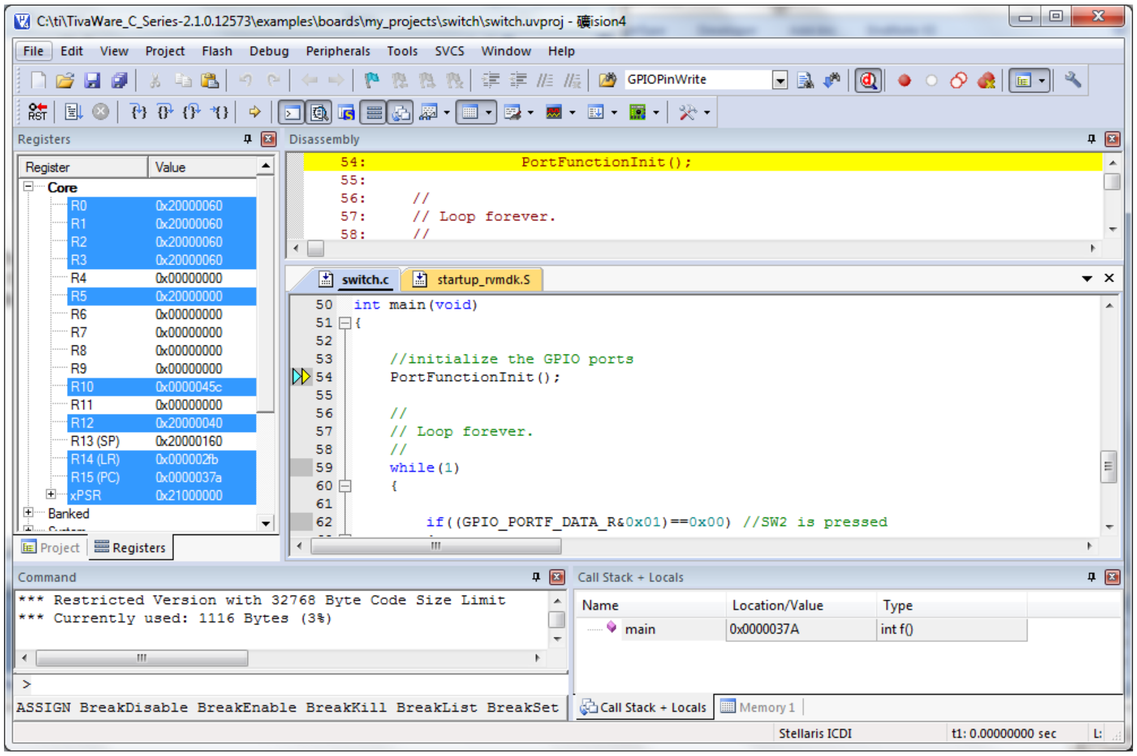

C:\ti\TivaWare_C_Series-2.1.0.12573\examples\boards\my_projects\switch\switch.uvproj - µVision4
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
GPIOPinWrite
Project
Target 1
Source Group 1
startup_rvmdk.S
switch.c
driverlib.lib
Project Books Func...
Build Output
Start/Stop Debug Session Ctrl+F5
Reset CPU
Run F5
Stop
Step F11
Step Over F10
Step Out Ctrl+F11
Run to Cursor Line Ctrl+F10
Show Next Statement
Breakpoints... Ctrl+B
Insert/Remove Breakpoint F9
Enable/Disable Breakpoint Ctrl+F9
Disable All Breakpoints
Kill All Breakpoints Ctrl+Shift+F9
OS Support
Execution Profiling
Memory Map...
Inline Assembly...
Function Editor (Open Ini File)...

```
3gh6pm.h"
sysctl.h"

     0x04
*******************************************************************

ample that interfaces with the blue LED (PF2) and SW2 (PF0)
ister access. When SW2 is pressed, the LED is turned on. When
 the LED is turned off.

*******************************************************************
```

```
Rebuild target 'Target 1'
assembling startup_rvmdk.
compiling switch.c...
linking...
Program Size: Code=1100 R
".\switch.axf" - 0 Error(
```

Figure 9

The IDE switches to debugging mode. The processor registers show in a window on the left, the debugger command window is visible at the bottom, and the main window shows the source code being debugged. The debugger automatically stops at main (see Figure 10).


C:\ti\TivaWare_C_Series-2.1.0.12573\examples\boards\my_projects\switch\switch.uvproj - µVision4
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
GPIOPinWrite
Registers
Disassembly
switch.c
startup_rvmdk.S
Project Registers
Command
Call Stack + Locals
Memory 1
Stellaris ICDI
t1: 0.00000000 sec

Register	Value
Core	
R0	0x20000060
R1	0x20000060
R2	0x20000060
R3	0x20000060
R4	0x00000000
R5	0x20000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x0000045c
R11	0x00000000
R12	0x20000040
R13 (SP)	0x20000160
R14 (LR)	0x000002fb
R15 (PC)	0x0000037a
xPSR	0x21000000
Banked	

```
    54:                 PortFunctionInit();
    55:
    56:     //
    57:     // Loop forever.
    58:     //
```

```
int main(void)
{

    //initialize the GPIO ports
    PortFunctionInit();

    //
    // Loop forever.
    //
    while(1)
    {

        if((GPIO_PORTF_DATA_R&0x01)==0x00) //SW2 is pressed
```

```
*** Restricted Version with 32768 Byte Code Size Limit
*** Currently used: 1116 Bytes (3%)

>
ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet
```

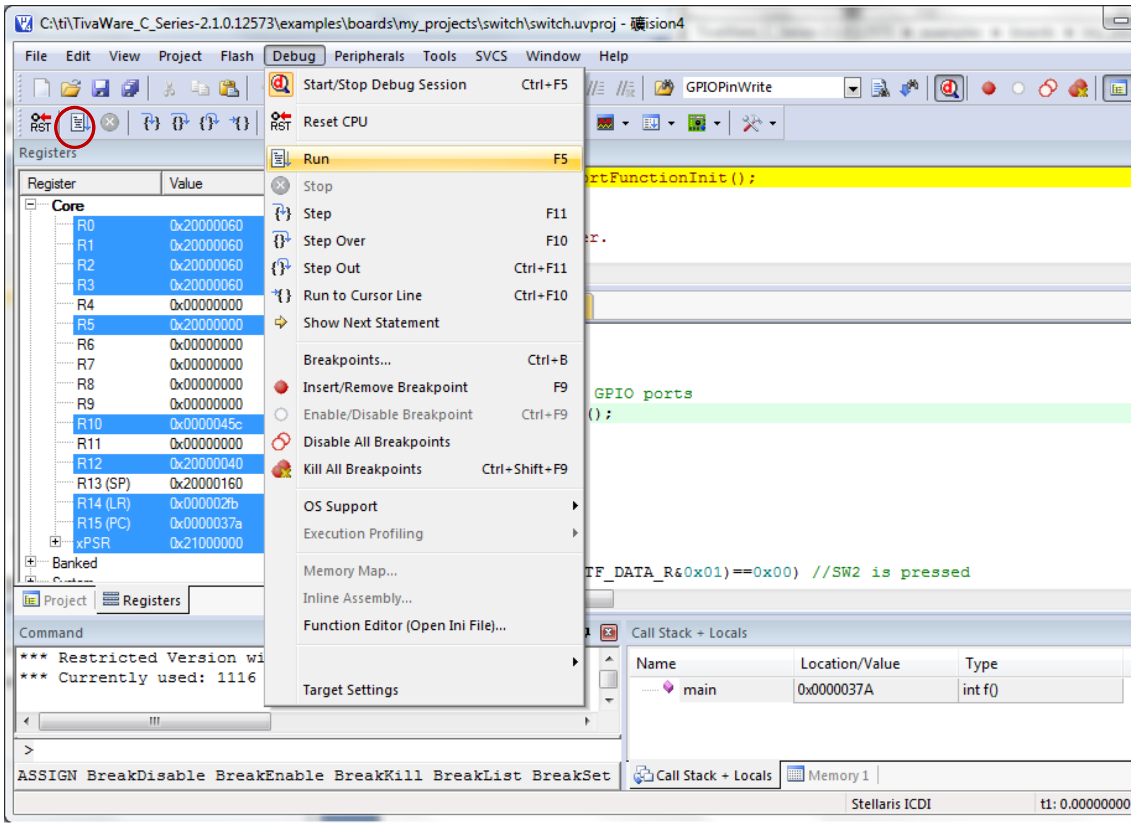
Name	Location/Value	Type
main	0x0000037A	int f()

Figure 10

From here, you can:

- Examine and modify memory.
- Program variables and processor registers.
- Set breakpoints.
- Single step through a program.
- Perform other typical debugging activities.

To run the program, select Run from the Debug menu, or click the Run button (icon) (see Figure 11).



Register	Value
Core	
R0	0x20000060
R1	0x20000060
R2	0x20000060
R3	0x20000060
R4	0x00000000
R5	0x20000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x0000045c
R11	0x00000000
R12	0x20000040
R13 (SP)	0x20000160
R14 (LR)	0x000002fb
R15 (PC)	0x0000037a
xPSR	0x21000000
Banked	

```
rtFunctionInit();
r.
 GPIO ports
();
TF_DATA_R&0x01)==0x00) //SW2 is pressed
```

```
*** Restricted Version wi
*** Currently used: 1116
>
ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet
```

Name	Location/Value	Type
main	0x0000037A	int f()

Figure 11

The application runs. When you press SW2, the blue LED is turned on, otherwise the LED is off.

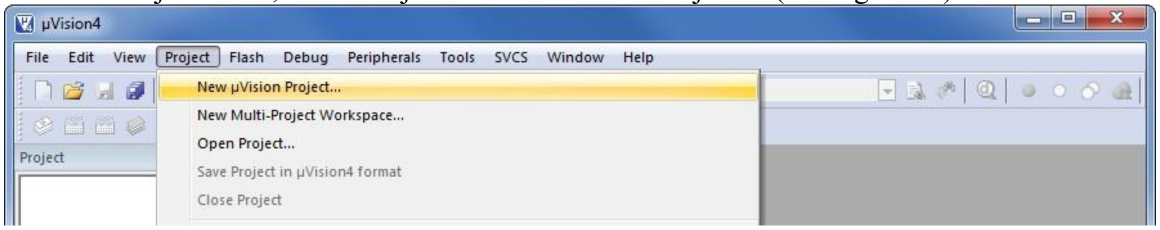
B. Build and Run Sample Project *toggle*

Follow the instructions in Section A, load, and run the other sample project *toggle*, and see the results.

C. Create a New Project

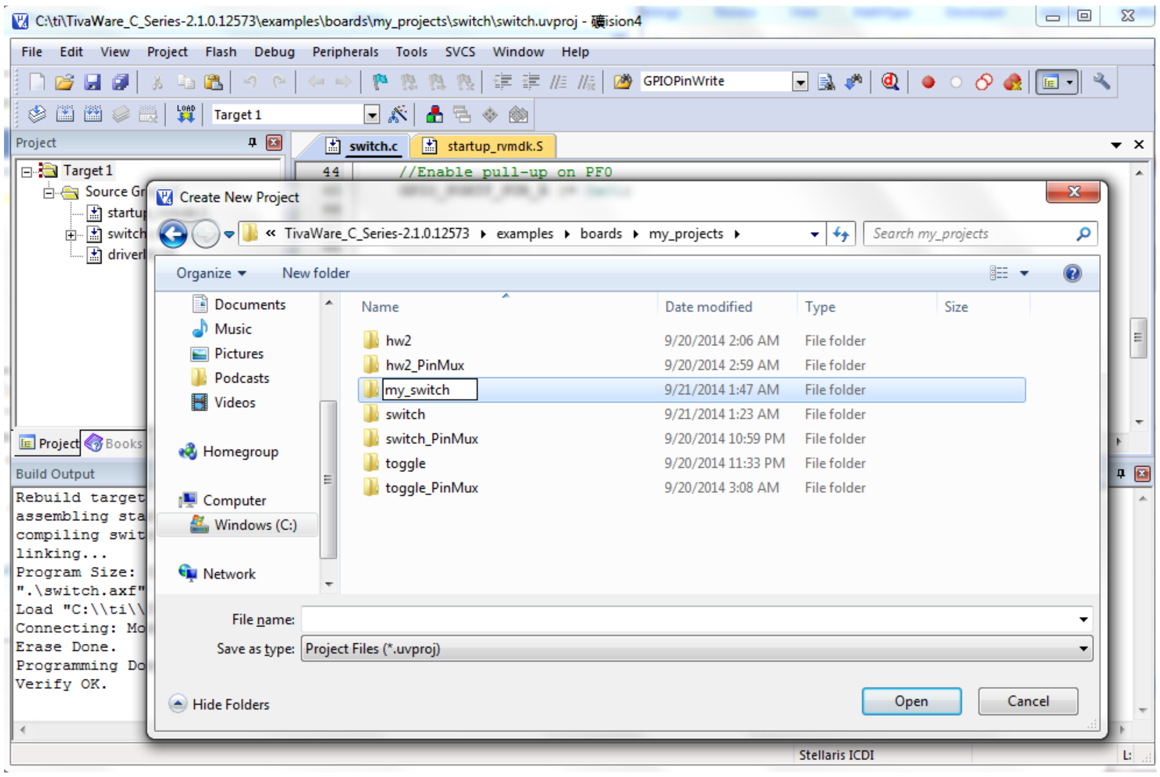
Once you have gone through the sample projects, you may want to create your own project to start development. While you can always start with an existing, simple project, sometimes you may want to create a new project. In this tutorial, we will duplicate the process of implementing the *switch* project step by step.

1. In the Project menu, select Project → New uVision Project… (see Figure 12)


µVision4
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
New µVision Project...
New Multi-Project Workspace...
Open Project...
Save Project in µVision4 format
Close Project
Project

. **Figure 12**

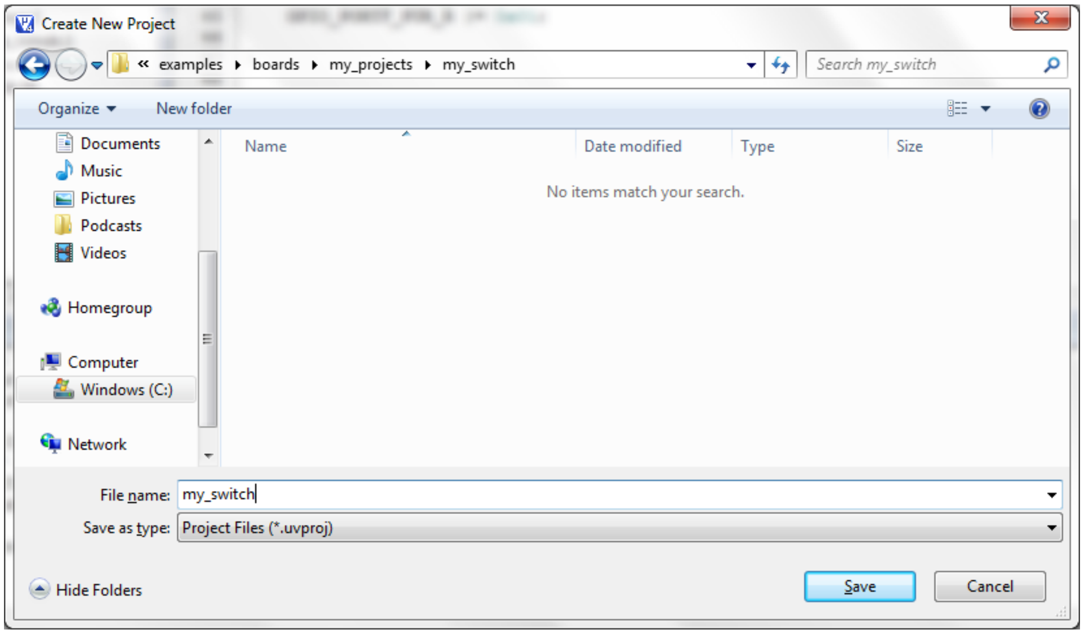
2. The Create New Project dialog box appears (see Figure 13). To use the functions in the TivaWare library, we need to do some configuration. In this example, we create the project within the TivaWare tree by saving it in the existing *C:\ti\TivaWare_C_Series-2.x.x.xxxxx\examples\boards\my_projects* directory. We first create a new folder named as *my_switch*, and then create the project file *my_switch.uvproj* in the *C:\ti\TivaWare_C_Series-2.x.x.xxxxx\examples\boards\my_projects\my_switch directory* (see Figure 14) and click "save".


C:\ti\TivaWare_C_Series-2.1.0.12573\examples\boards\my_projects\switch\switch.uvproj - µVision4
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
GPIOPinWrite
Target 1
Project
switch.c
startup_rvmdk.S
//Enable pull-up on PF0
Create New Project
« TivaWare_C_Series-2.1.0.12573 ▸ examples ▸ boards ▸ my_projects ▸
Search my_projects
Organize ▾
New folder
Documents
Music
Pictures
Podcasts
Videos
Homegroup
Computer
Windows (C:)
Network
File name:
Save as type: Project Files (*.uvproj)
Hide Folders
Open
Cancel
Build Output
Stellaris ICDI

Name	Date modified	Type	Size
hw2	9/20/2014 2:06 AM	File folder	
hw2_PinMux	9/20/2014 2:59 AM	File folder	
my_switch	9/21/2014 1:47 AM	File folder	
switch	9/21/2014 1:23 AM	File folder	
switch_PinMux	9/20/2014 10:59 PM	File folder	
toggle	9/20/2014 11:33 PM	File folder	
toggle_PinMux	9/20/2014 3:08 AM	File folder	

```
Rebuild target
assembling sta
compiling swit
linking...
Program Size:
".\switch.axf"
Load "C:\\ti\\
Connecting: Mo
Erase Done.
Programming Do
Verify OK.
```

Figure 13



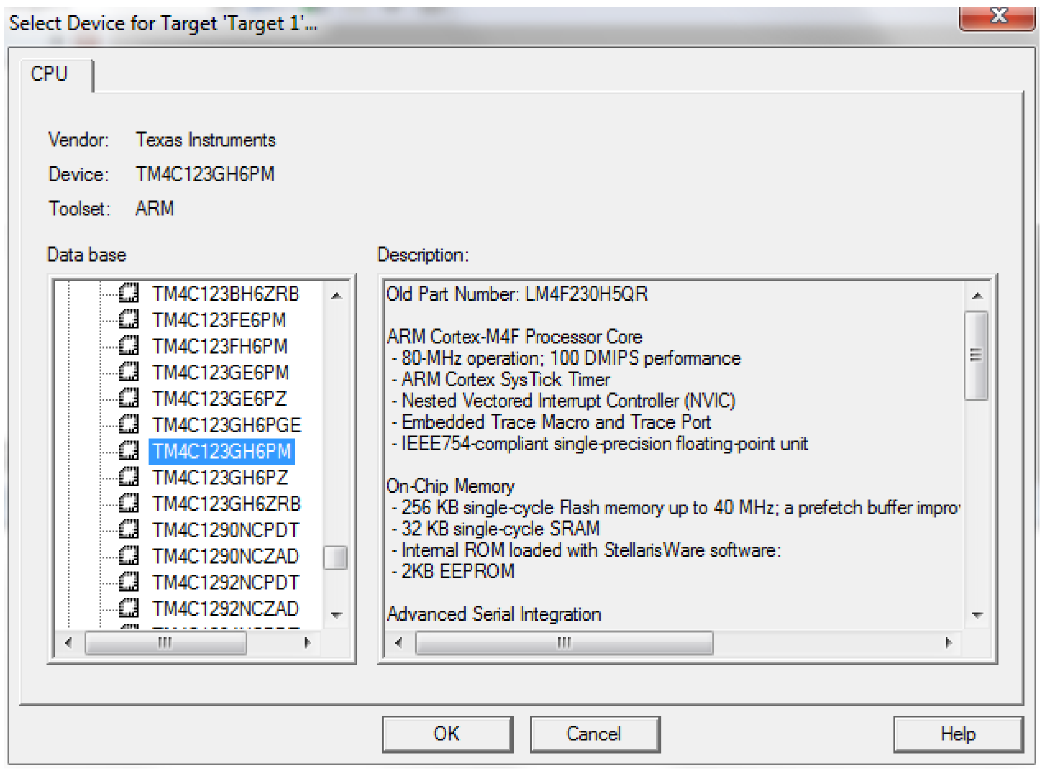
No items match your search.

File name: my_switch

Save as type: Project Files (*.uvproj)

Figure 14

3. Once the project file (.uvproj) is saved, a dialog window appears asking you to select the device that you are using (see Figure 15). Select the appropriate device under the “Texas Instruments” list. The device we need to select is TM4C123GH6PM.



Vendor: Texas Instruments

Device: TM4C123GH6PM

Toolset: ARM

Data base

TM4C123BH6ZRB
TM4C123FE6PM
TM4C123FH6PM
TM4C123GE6PM
TM4C123GE6PZ
TM4C123GH6PGE
TM4C123GH6PM
TM4C123GH6PZ
TM4C123GH6ZRB
TM4C1290NCPDT
TM4C1290NCZAD
TM4C1292NCPDT
TM4C1292NCZAD

Description:

Old Part Number: LM4F230H5QR

ARM Cortex-M4F Processor Core
- 80-MHz operation; 100 DMIPS performance
- ARM Cortex SysTick Timer
- Nested Vectored Interrupt Controller (NVIC)
- Embedded Trace Macro and Trace Port
- IEEE754-compliant single-precision floating-point unit

On-Chip Memory
- 256 KB single-cycle Flash memory up to 40 MHz; a prefetch buffer impro
- 32 KB single-cycle SRAM
- Internal ROM loaded with StellarisWare software:
- 2KB EEPROM

Advanced Serial Integration

Figure 15

4. The tool asks whether you want to add startup code to the project (see Figure 16). Select “No” because we will add the startup code provided by the TivaWare library.

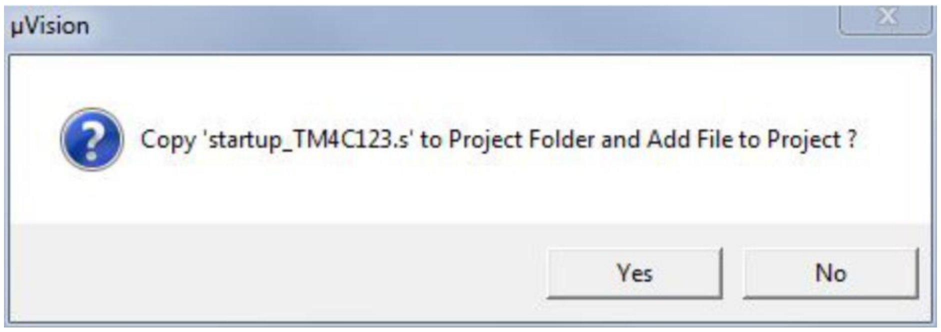
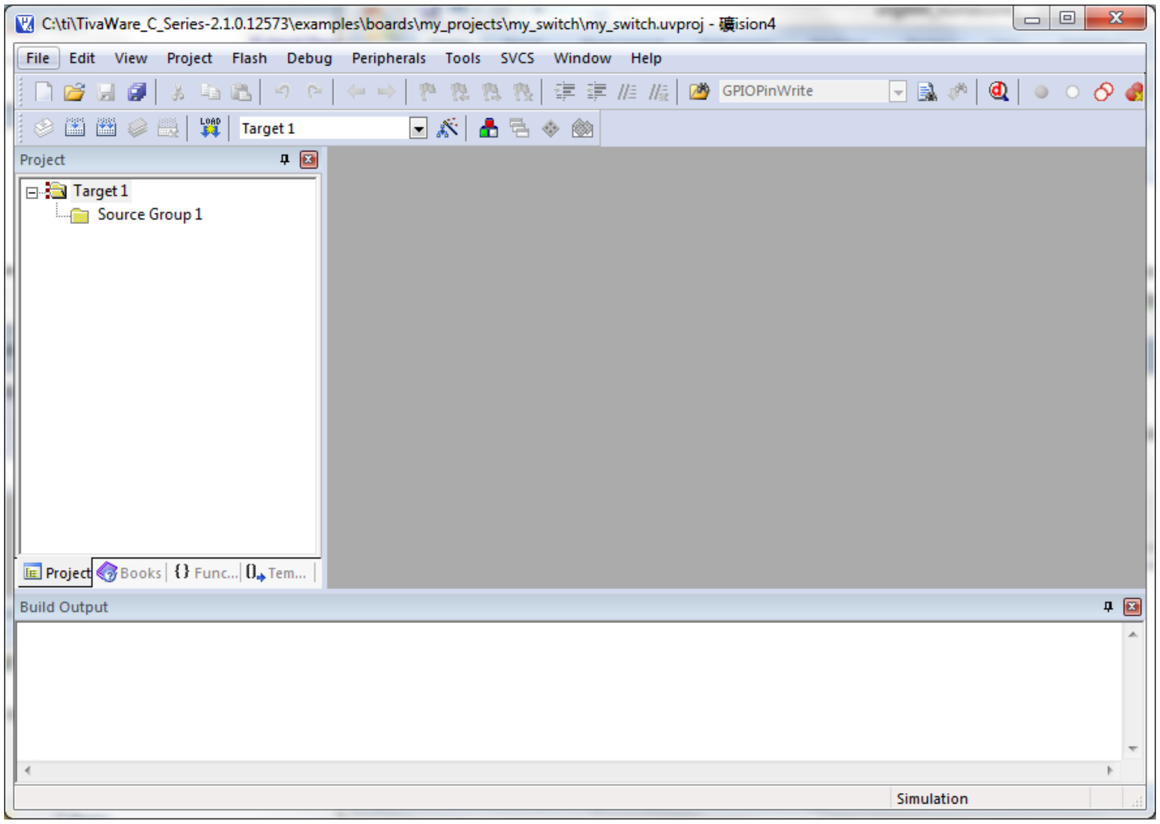

µVision
Copy 'startup_TM4C123.s' to Project Folder and Add File to Project ?
Yes
No

Figure 16

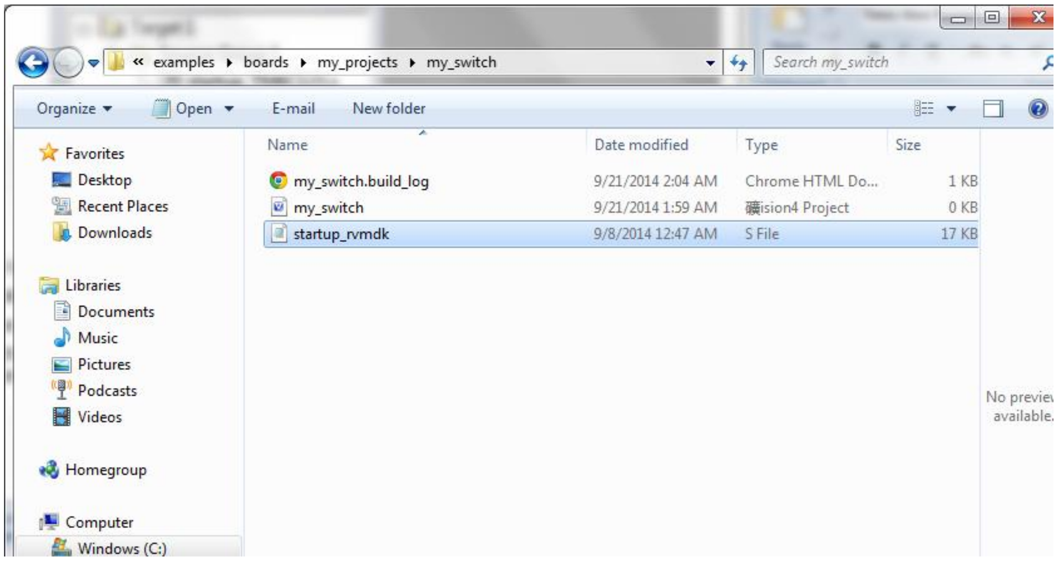
The empty project appears as shown in Figure 17.


C:\ti\TivaWare_C_Series-2.1.0.12573\examples\boards\my_projects\my_switch\my_switch.uvproj - µVision4
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
GPIOPinWrite
Target 1
Project
Target 1
Source Group 1
Project Books Func... Tem...
Build Output
Simulation

. **Figure 17**

5. Now we will add the startup code to the project. Go to the directory where stores the sample project *switch* and copy the startup file **startup_rvmdk.s** and paste it in the *my_switch* directory (see Figure 18).

Note that for all the future labs and your project, we will use the same fixed startup file.



Name	Date modified	Type	Size
my_switch.build_log	9/21/2014 2:04 AM	Chrome HTML Do...	1 KB
my_switch	9/21/2014 1:59 AM	礦ision4 Project	0 KB
startup_rvmdk	9/8/2014 12:47 AM	S File	17 KB

Figure 18

6. Next we will add the startup code to our project. Right-click the *Source Group 1* folder in the µVision IDE and select *Add Existing Files to Group 'Source Group 1'…* (see Figure 19). When the dialog box pops up to find the file, browse to your project directory, select **startup_rvmdk.s,** and add it to *Source Group 1* (see Figure 20).

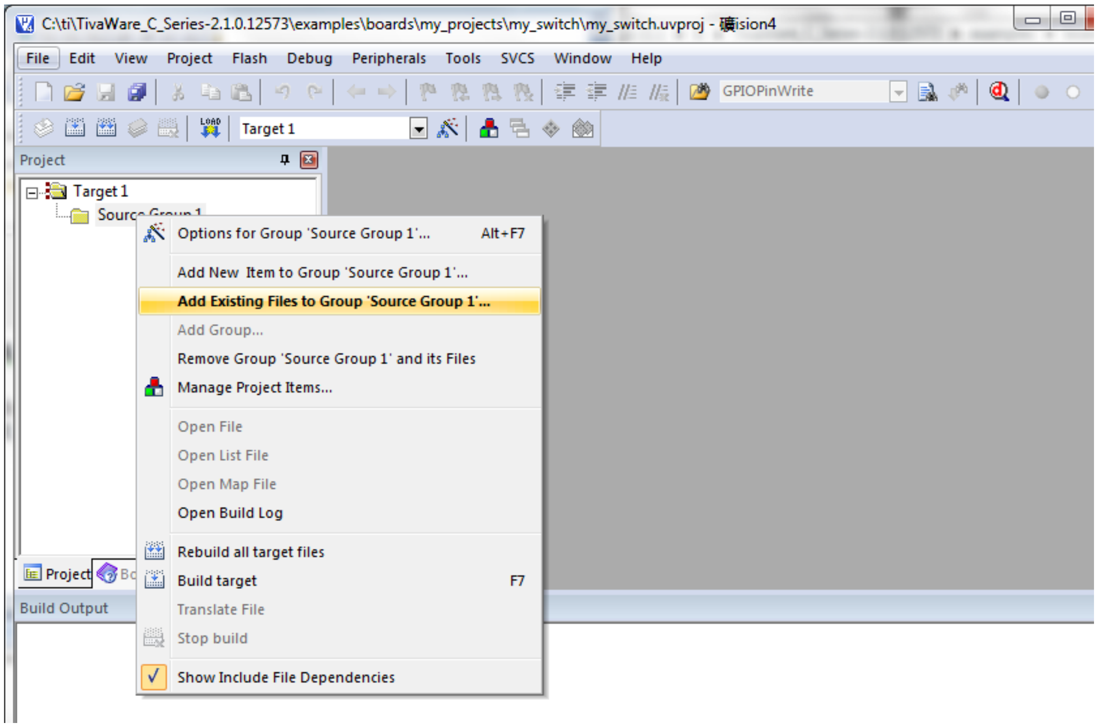

C:\ti\TivaWare_C_Series-2.1.0.12573\examples\boards\my_projects\my_switch\my_switch.uvproj - 礦ision4
File
Edit
View
Project
Flash
Debug
Peripherals
Tools
SVCS
Window
Help
GPIOPinWrite
Target 1
Project
Target 1
Source Group 1
Options for Group 'Source Group 1'... Alt+F7
Add New Item to Group 'Source Group 1'...
Add Existing Files to Group 'Source Group 1'...
Add Group...
Remove Group 'Source Group 1' and its Files
Manage Project Items...
Open File
Open List File
Open Map File
Open Build Log
Rebuild all target files
Build target F7
Translate File
Stop build
Show Include File Dependencies
Project
Build Output

Figure 19

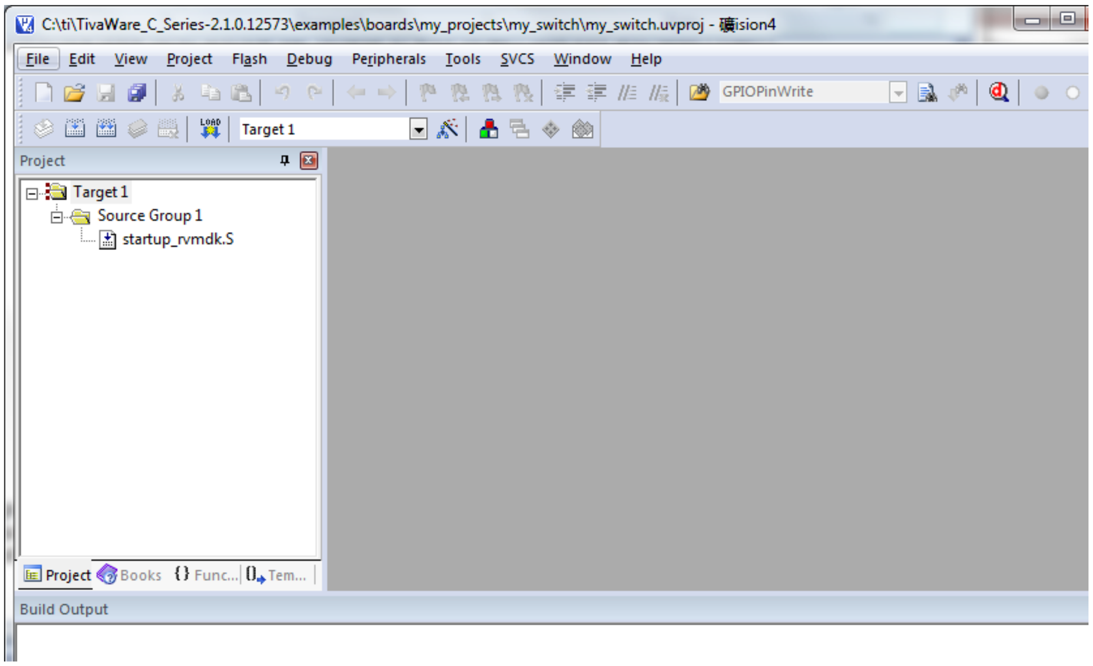

C:\ti\TivaWare_C_Series-2.1.0.12573\examples\boards\my_projects\my_switch\my_switch.uvproj - µVision4
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
GPIOPinWrite
Target 1
Project
Target 1
Source Group 1
startup_rvmdk.S
Project Books Func... Tem...
Build Output

Figure 20

7. Add TivaWare Hooks. To use the functions in the TivaWare library, we need to add the **driverlib.lib** file to the project. Right-click the *Source Group 1* folder in the µVision IDE and select *Add Existing Files to Group 'Source Group 1'…* When the dialog box pops up to find the file, browse to the directory *C:\ti\TivaWare_C_Series-2.x.x.xxxxx\driverlib\rvmdk*, select **driverlib.lib,** and add it to *Source Group 1* (see Figure 21).

Note: You must tell the file browser to look for *.lib file types, so change the Files of type drop-down from C Source file (*.c) to Library file (*.lib) or All files.

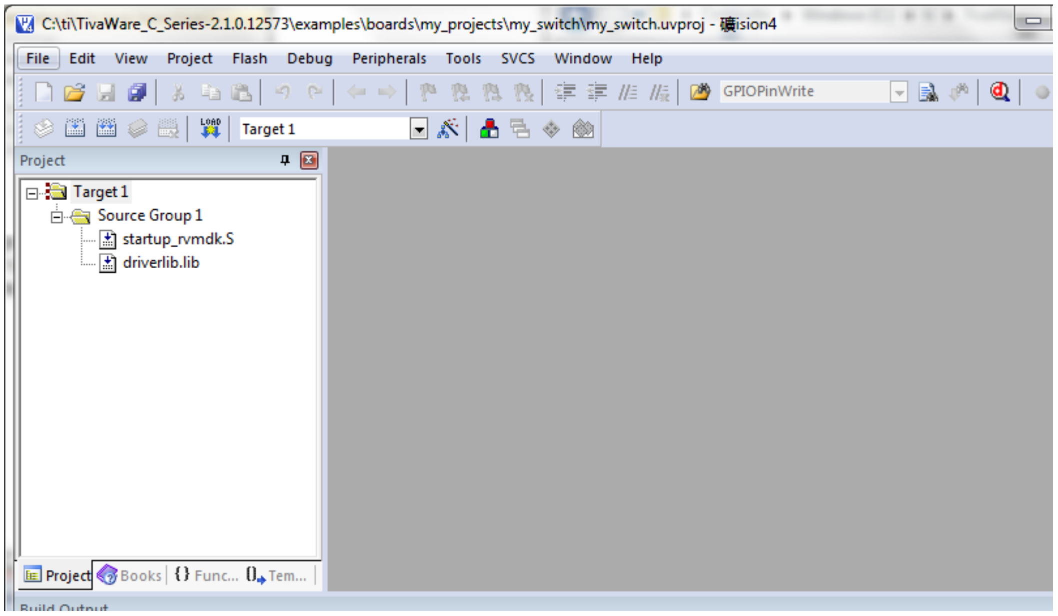

C:\ti\TivaWare_C_Series-2.1.0.12573\examples\boards\my_projects\my_switch\my_switch.uvproj - µVision4
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
GPIOPinWrite
Target 1
Project
Target 1
Source Group 1
startup_rvmdk.S
driverlib.lib
Project Books Func... Tem...
Build Output

Figure 21

8. Create our main C program file by right-clicking the *Source Group 1* folder in the µVision IDE and select *Add New Item to Group 'Source Group 1'*… (see Figure 22). When the dialog box pops up to add a new item, choose the file type as C File (.c), and name the new file as *my_switch* and click Add (see Figure 23).

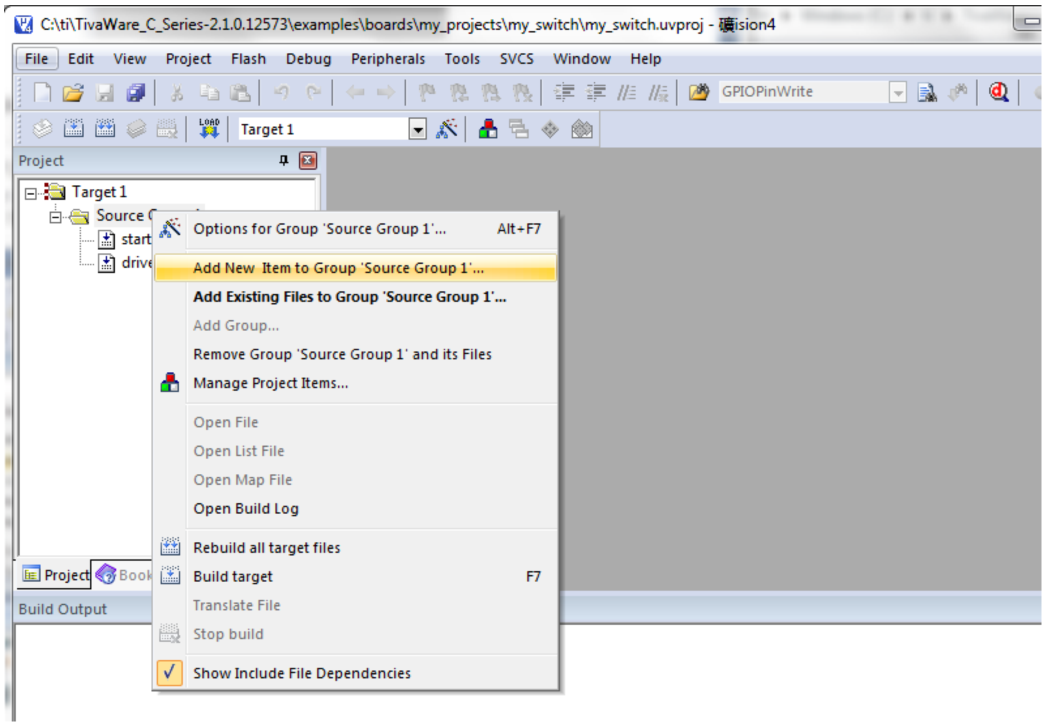
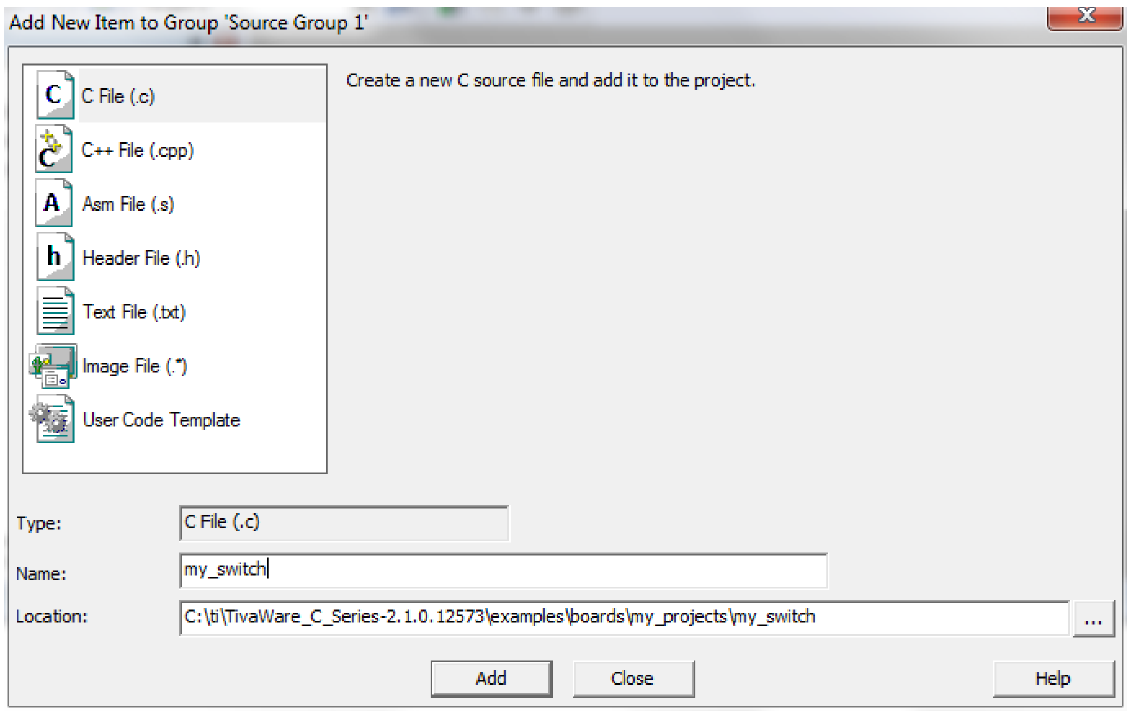

C:\ti\TivaWare_C_Series-2.1.0.12573\examples\boards\my_projects\my_switch\my_switch.uvproj - µVision4
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
GPIOPinWrite
Target 1
Project
Target 1
Source Group 1
start
drive
Options for Group 'Source Group 1'... Alt+F7
Add New Item to Group 'Source Group 1'...
Add Existing Files to Group 'Source Group 1'...
Add Group...
Remove Group 'Source Group 1' and its Files
Manage Project Items...
Open File
Open List File
Open Map File
Open Build Log
Rebuild all target files
Build target F7
Translate File
Stop build
Show Include File Dependencies
Project
Book
Build Output

Figure 22


Add New Item to Group 'Source Group 1'

C File (.c)
C++ File (.cpp)
Asm File (.s)
Header File (.h)
Text File (.txt)
Image File (.*)
User Code Template

Create a new C source file and add it to the project.

Type: C File (.c)

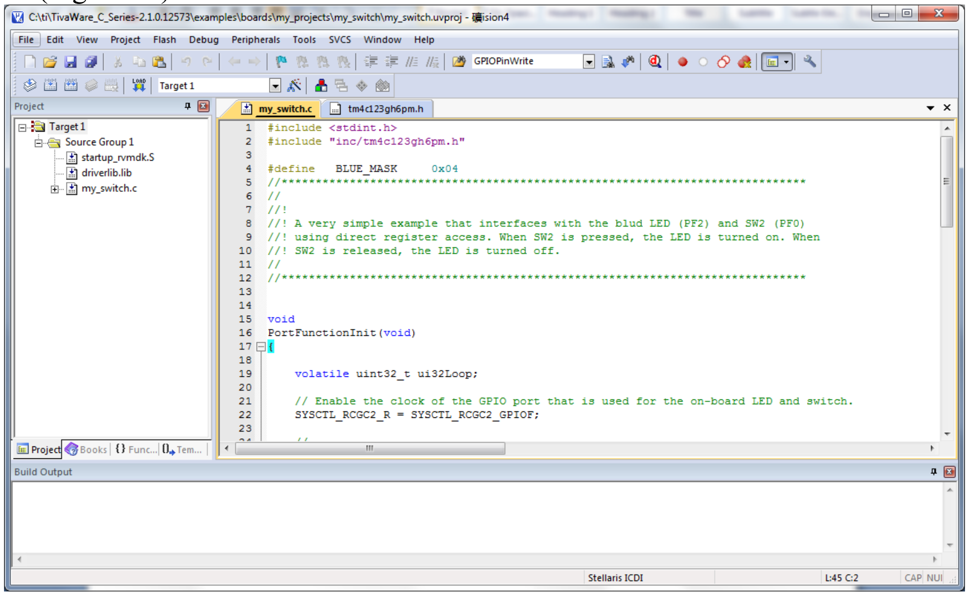
Name: my_switch

Location: C:\ti\TivaWare_C_Series-2.1.0.12573\examples\boards\my_projects\my_switch

Add Close Help

Figure 23

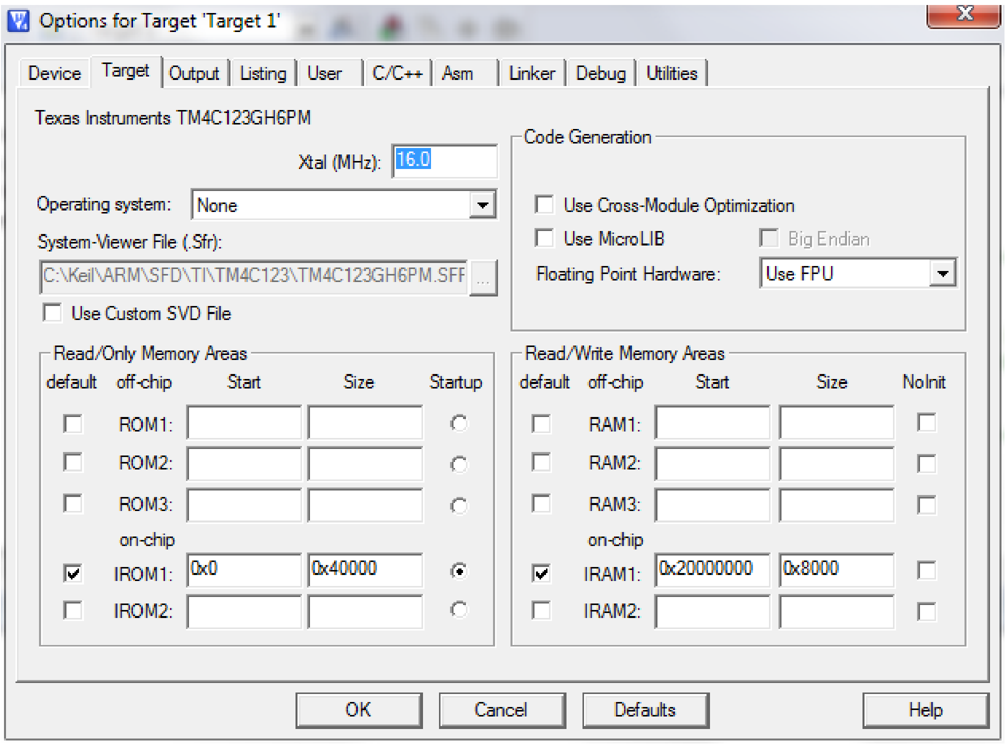
9. Now the file *my_switch.c* is empty. We just copy the code in the file *switch.c* in the sample project *switch* and paste it in our *my_switch.c* file. So far we have included all necessary files for the project (Figure 24).



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

#define   BLUE_MASK     0x04
//*****************************************************************************
//
//!
//! A very simple example that interfaces with the blud LED (PF2) and SW2 (PF0)
//! using direct register access. When SW2 is pressed, the LED is turned on. When
//! SW2 is released, the LED is turned off.
//
//*****************************************************************************


void
PortFunctionInit(void)
{

    volatile uint32_t ui32Loop;

    // Enable the clock of the GPIO port that is used for the on-board LED and switch.
    SYSCTL_RCGC2_R = SYSCTL_RCGC2_GPIOF;

    //
```

Figure 24

10. Before we can build the project correctly, we need to do some project configurations. Go to Project → Options for Target 'Target 1.' The Options for Target 'Target 1' dialog box appears (see Figure 25).



Texas Instruments TM4C123GH6PM

Xtal (MHz): 16.0

Operating system: None

System-Viewer File (.Sfr): C:\Keil\ARM\SFD\TI\TM4C123\TM4C123GH6PM.SFR

☐ Use Custom SVD File

Code Generation

☐ Use Cross-Module Optimization

☐ Use MicroLIB ☐ Big Endian

Floating Point Hardware: Use FPU

Read/Only Memory Areas

default	off-chip	Start	Size	Startup
☐	ROM1:			○
☐	ROM2:			○
☐	ROM3:			○
	on-chip			
✓	IROM1:	0x0	0x40000	●
☐	IROM2:			○

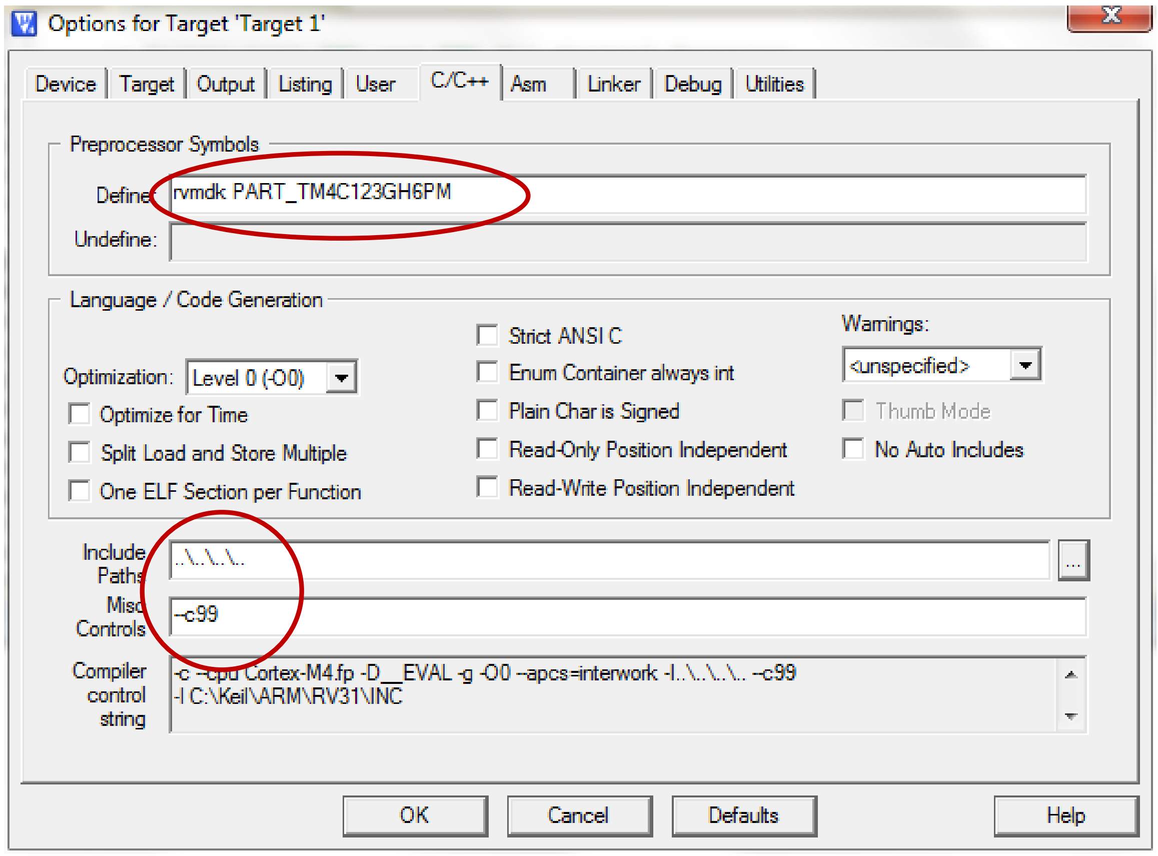
Read/Write Memory Areas

default	off-chip	Start	Size	NoInit
☐	RAM1:			☐
☐	RAM2:			☐
☐	RAM3:			☐
	on-chip			
✓	IRAM1:	0x20000000	0x8000	☐
☐	IRAM2:			☐

Figure 25

11. **Configure C/C++ Tab.** Select the C/C++ tab. Here you need to tell the project that you are using the ARM compiler, so define rvmdk. This definition is used in TivaWare to compile the Keil specific sections correctly. Define the part you are using as well as add the top-level TivaWare directory to your include path (see Figure 26).

Note: The easiest way to see how to configure your project is to look at the sample projects. You can launch another Keil uVision process and open the sample project *switch*. You can compare each tab in the project options, and do the configurations in your project by exactly copying the configurations from the sample project.



Preprocessor Symbols

Define: rvmdk PART_TM4C123GH6PM

Undefine:

Language / Code Generation

Optimization: Level 0 (-O0)

[ ] Optimize for Time

[ ] Split Load and Store Multiple

[ ] One ELF Section per Function

[ ] Strict ANSI C

[ ] Enum Container always int

[ ] Plain Char is Signed

[ ] Read-Only Position Independent

[ ] Read-Write Position Independent

Warnings: <unspecified>

[ ] Thumb Mode

[ ] No Auto Includes

Include Paths: ..\..\..\..

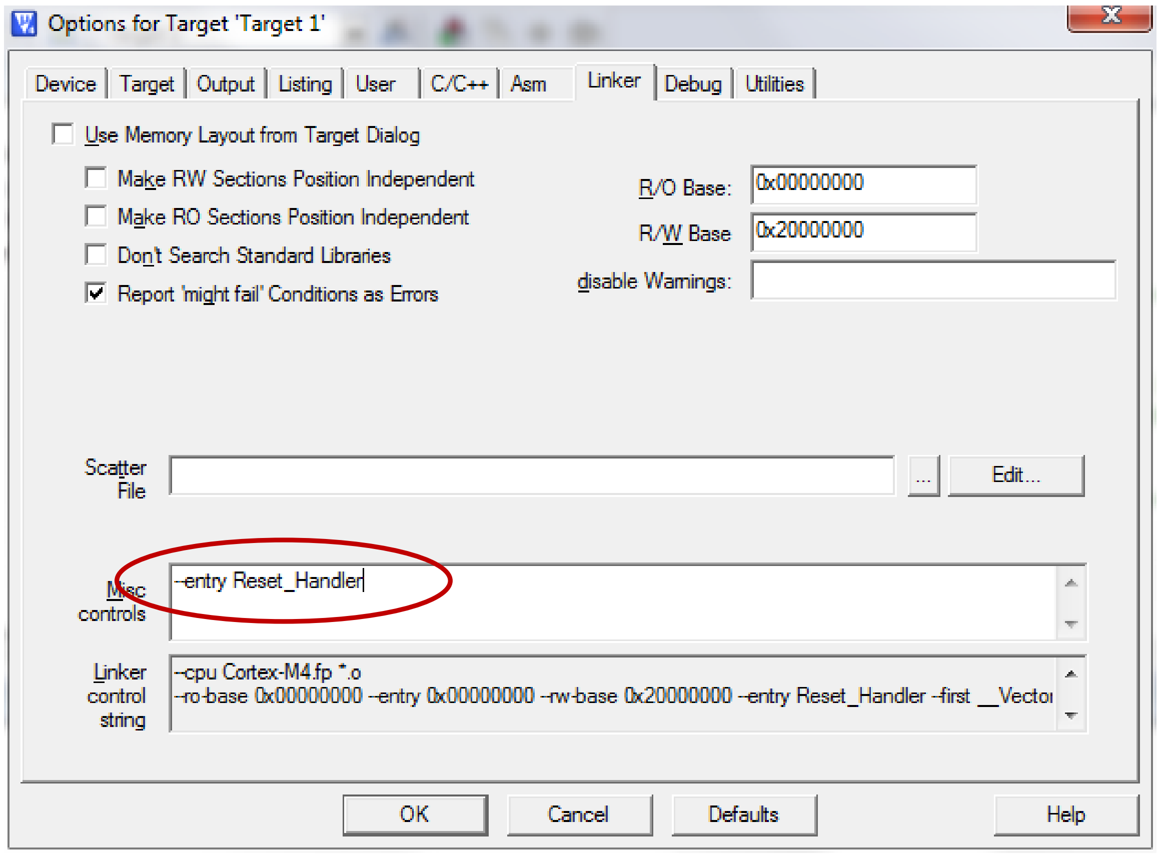
Misc Controls: --c99

Compiler control string:

```
-c --cpu Cortex-M4.fp -D__EVAL -g -O0 --apcs=interwork -I..\..\..\.. --c99
-I C:\Keil\ARM\RV31\INC
```

Figure 26

12. **Configure Linker Tab.** In the linker section, you must add **—entry Reset_Handler** to the Miscellaneous controls box (see Figure 27).



☐ Use Memory Layout from Target Dialog
☐ Make RW Sections Position Independent
☐ Make RO Sections Position Independent
☐ Don't Search Standard Libraries
☑ Report 'might fail' Conditions as Errors

R/O Base: 0x00000000
R/W Base 0x20000000
disable Warnings:

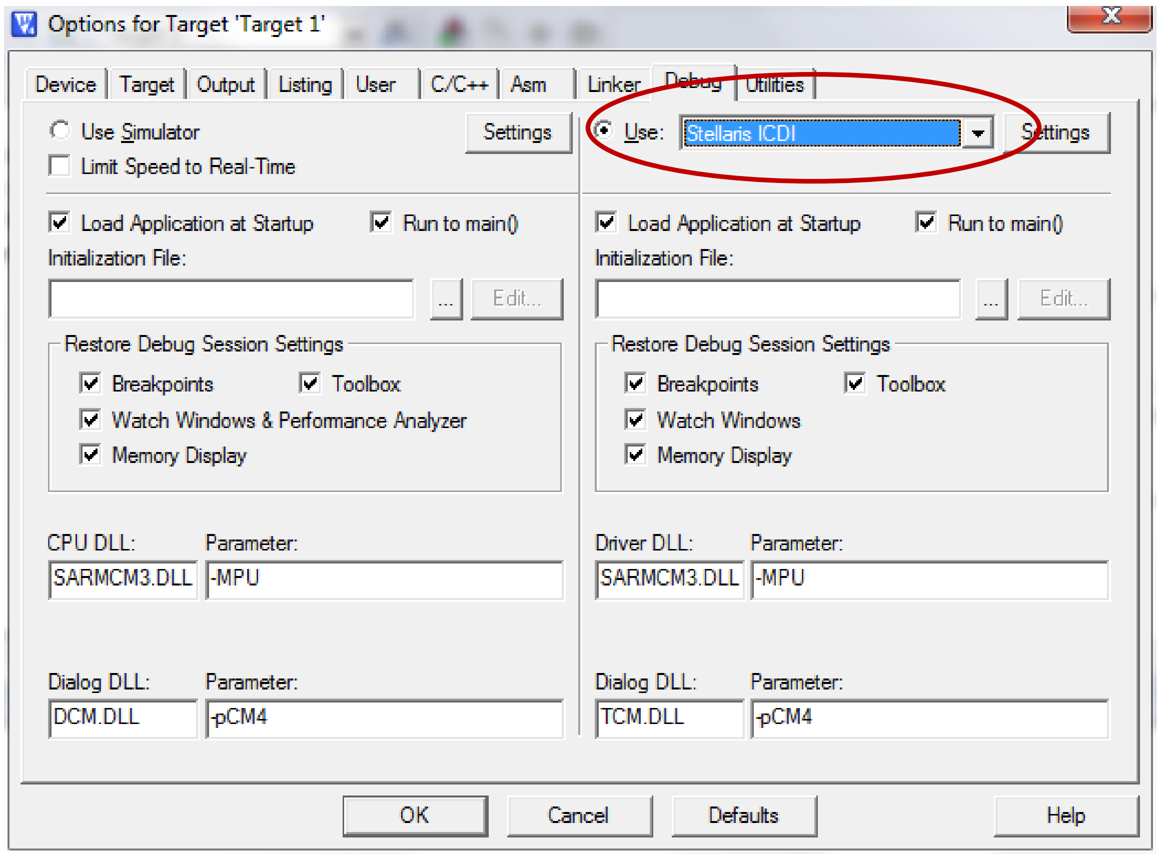
Scatter File:

Misc controls: --entry Reset_Handler

Linker control string:
```
--cpu Cortex-M4.fp *.o
--ro-base 0x00000000 --entry 0x00000000 --rw-base 0x20000000 --entry Reset_Handler --first __Vecto
```

Figure 27

13. **Configure Debug Tab.** The default configuration for a new project is to use the simulator, which is not going to allow you to debug real hardware. Choose the Use radio button on the right side with the ULINK Cortex™ Debugger next to it (this selection is the default in the drop-down box).Choose the Stellaris ICDI option from the drop-down box (see Figure 28).



( ) Use Simulator
[ ] Limit Speed to Real-Time
[✓] Load Application at Startup [✓] Run to main()
Initialization File:
Restore Debug Session Settings
[✓] Breakpoints [✓] Toolbox
[✓] Watch Windows & Performance Analyzer
[✓] Memory Display
CPU DLL: SARMCM3.DLL Parameter: -MPU
Dialog DLL: DCM.DLL Parameter: -pCM4

(●) Use: Stellaris ICDI
[✓] Load Application at Startup [✓] Run to main()
Initialization File:
Restore Debug Session Settings
[✓] Breakpoints [✓] Toolbox
[✓] Watch Windows
[✓] Memory Display
Driver DLL: SARMCM3.DLL Parameter: -MPU
Dialog DLL: TCM.DLL Parameter: -pCM4

Figure 28

You have now done all necessary configurations for the created new project. You can now build, download, and run the project on your launchpad and see if it works correctly.

D. Understand the Sample Project *switch*

The sample project *switch* implements a system that allows the microcontroller to interface with the on-board switch SW2 (connected with port PF0) and the blue LED (connected with port PF2). When SW2 is pressed, the LED is turned on, otherwise the LED is off.

1. The pseudo code and the flowchart (Figure 29) of the system design are shown below, illustrating the basic steps for the system.

```
Start     Initialize Port F, configure PF2 as output and PF0 as switch input
loop      Read the switch and test if the switch is pressed
          If PF0==0 (SW2 is pressed)
```

```
        Set PF2, so LED is on
If PF0==1 (SW2 is released)
        Clear PF2, so LED is off
Go to loop
```

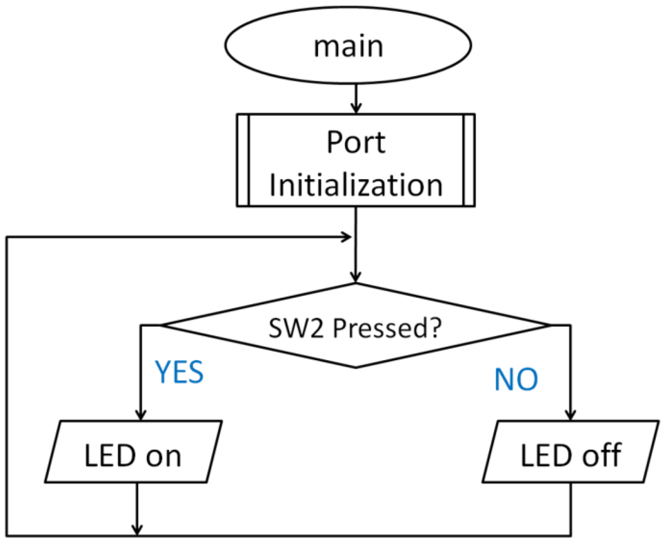


Figure 29

2. The header files included in switch.c are listed below:

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

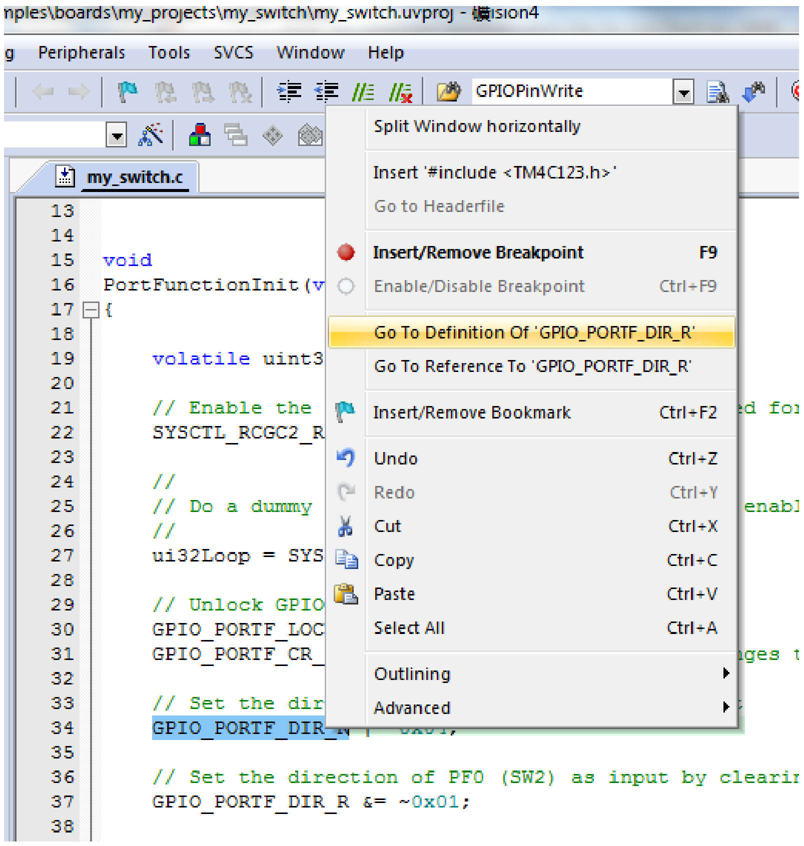
The use of the < > restricts the search path to only the specified path. Using the " " causes the search to start in the project directory. For includes like the two standard ones, you want to assure that you're accessing the original, standard files … not one's that may have been modified.

stdint.h: Variable definitions for the C99 standard

tm4c123gh6pm.h: Definitions for the interrupt and register assignments on the Tiva C Series device on the LaunchPad board. For detailed information of the register descriptions, please refer to the **Tiva TM4C123GH6PM Microcontroller Data Sheet** (Canvas->Reference Materials).

3. In this example, we initialize the IO ports by configure the register values manually. We first need to figure out the registers need to be configured, and then determine the values to be written to the registers. The register initialization is defined in the function *PortFunctionInit()* (see the code below). The registers such as SYSCTL_RCGC2_R, GPIO_PORTF_LOCK_R, GPIO_PORTF_CR_R, GPIO_PORTF_DIR_R are all defined in **tm4c123gh6pm.h**. To find out the definition of a specific register, you can right-click the register name, and choose Go to Definition of 'register name'. Then the definition of the variable in **tm4c123gh6pm.h** will be shown to you (see Figure 30 and Figure 31).

```
void
PortFunctionInit(void)
{

    volatile uint32_t ui32Loop;

    // Enable the clock of the GPIO port that is used for the on-board LED and switch.
    SYSCTL_RCGC2_R = SYSCTL_RCGC2_GPIOF;

    //
    // Do a dummy read to insert a few cycles after enabling the peripheral.
    //
    ui32Loop = SYSCTL_RCGC2_R;

    // Unlock GPIO Port F
    GPIO_PORTF_LOCK_R = 0x4C4F434B;
    GPIO_PORTF_CR_R |= 0x01;           // allow changes to PF0

    // Set the direction of PF2 (blue LED) as output
    GPIO_PORTF_DIR_R |= 0x04;

    // Set the direction of PF0 (SW2) as input by clearing the bit
    GPIO_PORTF_DIR_R &= ~0x01;

    // Enable both PF2 and PF0 for digital function.
    GPIO_PORTF_DEN_R |= 0x05;

    //Enable pull-up on PF0
    GPIO_PORTF_PUR_R |= 0x01;

}
```



nples\boards\my_projects\my_switch\my_switch.uvproj - µVision4

g Peripherals Tools SVCS Window Help

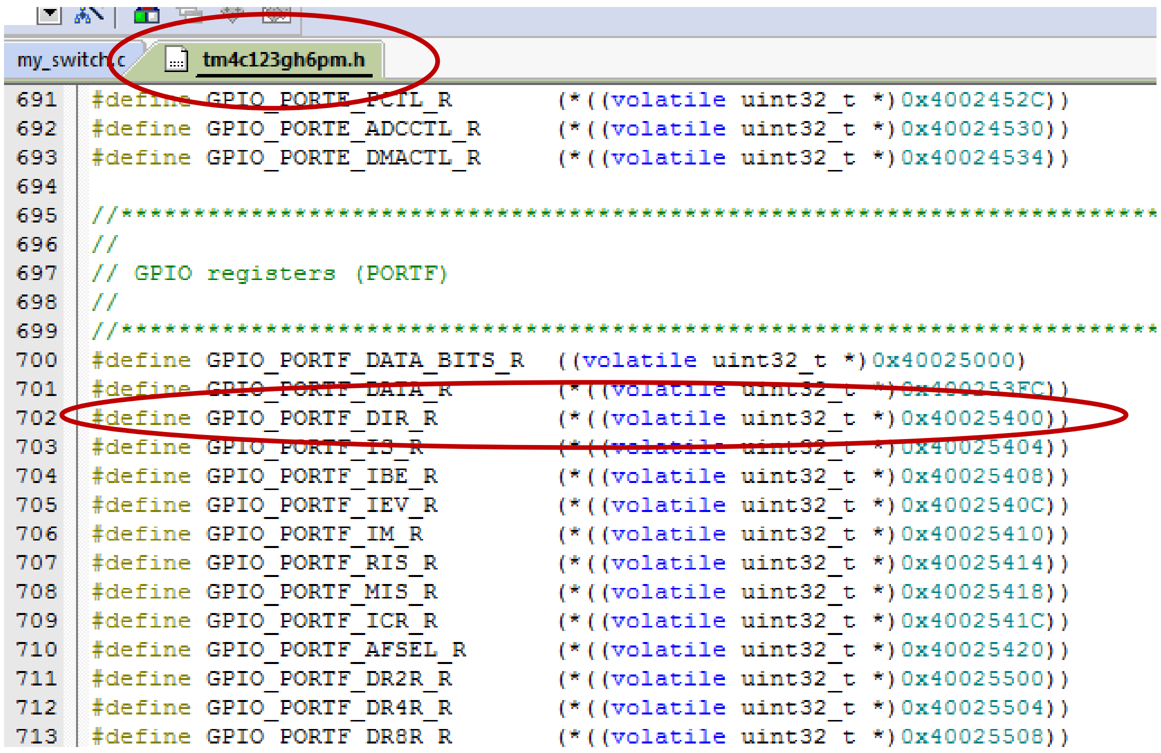
GPIOPinWrite

my_switch.c

```
void
PortFunctionInit(v
{

    volatile uint3

    // Enable the                                            ed fo
    SYSCTL_RCGC2_R

    //
    // Do a dummy                                            enabl
    //
    ui32Loop = SYS

    // Unlock GPIO
    GPIO_PORTF_LOC
    GPIO_PORTF_CR_                                           ges t

    // Set the dir
    GPIO_PORTF_DIR_R |= 0x04;

    // Set the direction of PF0 (SW2) as input by clearin
    GPIO_PORTF_DIR_R &= ~0x01;
```

Split Window horizontally
Insert '#include <TM4C123.h>'
Go to Headerfile
Insert/Remove Breakpoint F9
Enable/Disable Breakpoint Ctrl+F9
Go To Definition Of 'GPIO_PORTF_DIR_R'
Go To Reference To 'GPIO_PORTF_DIR_R'
Insert/Remove Bookmark Ctrl+F2
Undo Ctrl+Z
Redo Ctrl+Y
Cut Ctrl+X
Copy Ctrl+C
Paste Ctrl+V
Select All Ctrl+A
Outlining
Advanced

Figure 30



```
#define GPIO_PORTE_PCTL_R       (*((volatile uint32_t *)0x4002452C))
#define GPIO_PORTE_ADCCTL_R     (*((volatile uint32_t *)0x40024530))
#define GPIO_PORTE_DMACTL_R     (*((volatile uint32_t *)0x40024534))

//*****************************************************************************
//
// GPIO registers (PORTF)
//
//*****************************************************************************
#define GPIO_PORTF_DATA_BITS_R  ((volatile uint32_t *)0x40025000)
#define GPIO_PORTF_DATA_R       (*((volatile uint32_t *)0x400253FC))
#define GPIO_PORTF_DIR_R        (*((volatile uint32_t *)0x40025400))
#define GPIO_PORTF_IS_R         (*((volatile uint32_t *)0x40025404))
#define GPIO_PORTF_IBE_R        (*((volatile uint32_t *)0x40025408))
#define GPIO_PORTF_IEV_R        (*((volatile uint32_t *)0x4002540C))
#define GPIO_PORTF_IM_R         (*((volatile uint32_t *)0x40025410))
#define GPIO_PORTF_RIS_R        (*((volatile uint32_t *)0x40025414))
#define GPIO_PORTF_MIS_R        (*((volatile uint32_t *)0x40025418))
#define GPIO_PORTF_ICR_R        (*((volatile uint32_t *)0x4002541C))
#define GPIO_PORTF_AFSEL_R      (*((volatile uint32_t *)0x40025420))
#define GPIO_PORTF_DR2R_R       (*((volatile uint32_t *)0x40025500))
#define GPIO_PORTF_DR4R_R       (*((volatile uint32_t *)0x40025504))
#define GPIO_PORTF_DR8R_R       (*((volatile uint32_t *)0x40025508))
```

Figure 31

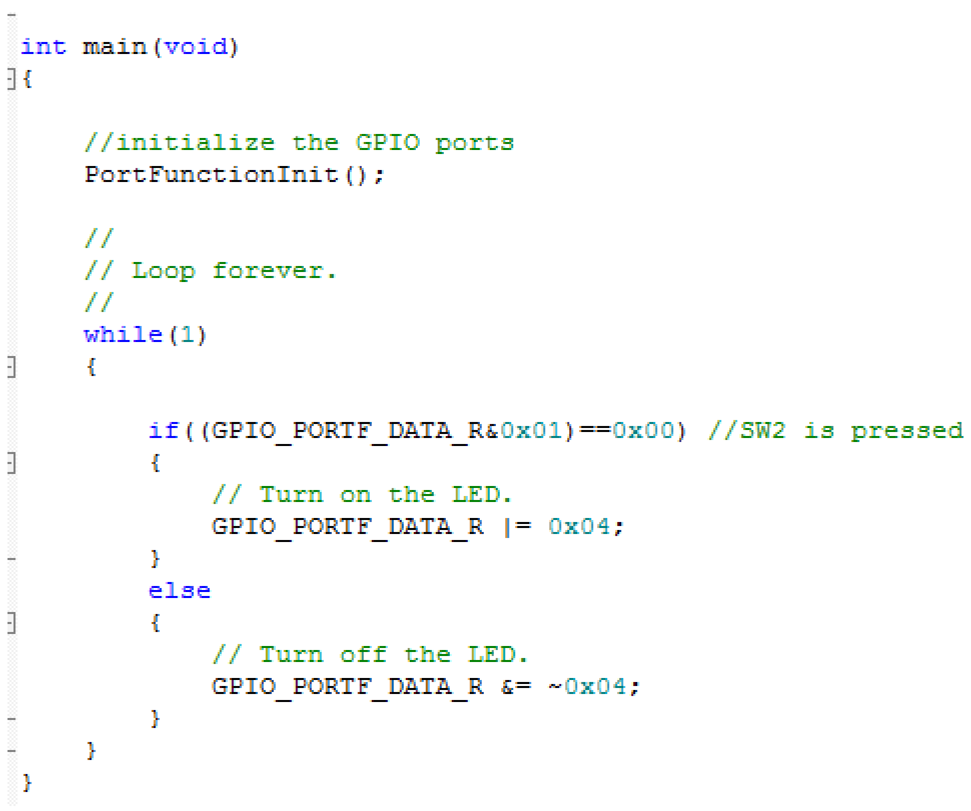
4. In this example, we need to configure PF0 as an output port and PF2 as an input port. To access PF0, we need to unlock it first. Here we use two lines of code to unlock PF0:

```
// Unlock GPIO Port F
GPIO_PORTF_LOCK_R = 0x4C4F434B;
GPIO_PORTF_CR_R |= 0x01;            // allow changes to PF0
```

In addition, PF0 is connected with the on-board switch SW2. The switches on the LaunchPad are using negative logic. We need to enable the pull-up register on PF0 in order for SW2 to function correctly.

```
//Enable pull-up on PF0
GPIO_PORTF_PUR_R |= 0x01;
```

5. In the main function *main()*, the system is implemented by converting the flow chart shown in Figure 29 into C program. The main function starts with the port initialization function *PortFunctionInit()*, followed by an infinite while loop which checks the status of SW2 repeatedly.



```
int main(void)
{

    //initialize the GPIO ports
    PortFunctionInit();

    //
    // Loop forever.
    //
    while(1)
    {

        if((GPIO_PORTF_DATA_R&0x01)==0x00) //SW2 is pressed
        {
            // Turn on the LED.
            GPIO_PORTF_DATA_R |= 0x04;
        }
        else
        {
            // Turn off the LED.
            GPIO_PORTF_DATA_R &= ~0x04;
        }
    }
}
```

E. Understand the Sample Project *toggle*

The sample project *toggle* implements a system that toggles the on-board red LED (connected with port PF1) repeatedly.

1. The pseudo code and the flowchart (Figure 32) of the system design are shown below, illustrating the basic steps for the system.

```
Start    Initialize Port F, configure PF1 as output
         Initialize the value on PF1 as 1 (red LED on)
loop     Delay for a number of cycles
         toggle PF1
         Go to loop
```

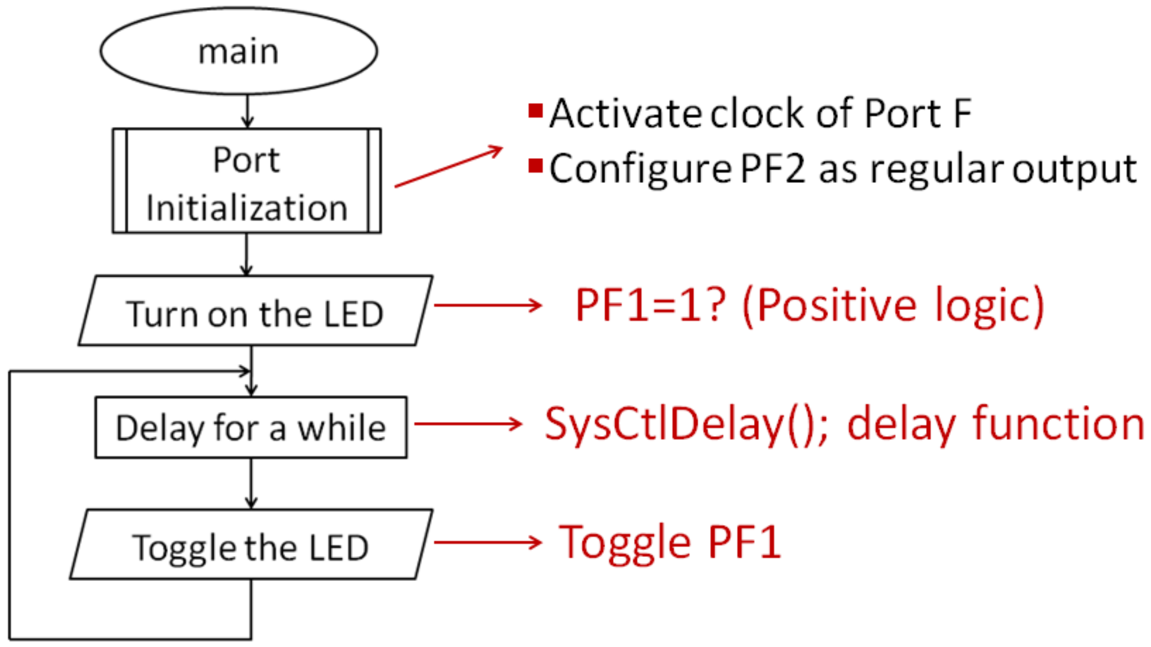


Figure 32

2. **SysCtlDelay**(unit32_t ui32Count) is a delay function provided in the TivaWare library. It implements a loop timer in which the parameter ui32Count is the number of delay loop iterations to perform. Please search for "SysCtlDelay" in the *TivaWare Peripheral Driver Library User Guide* and find the detailed description of this function.

Questions

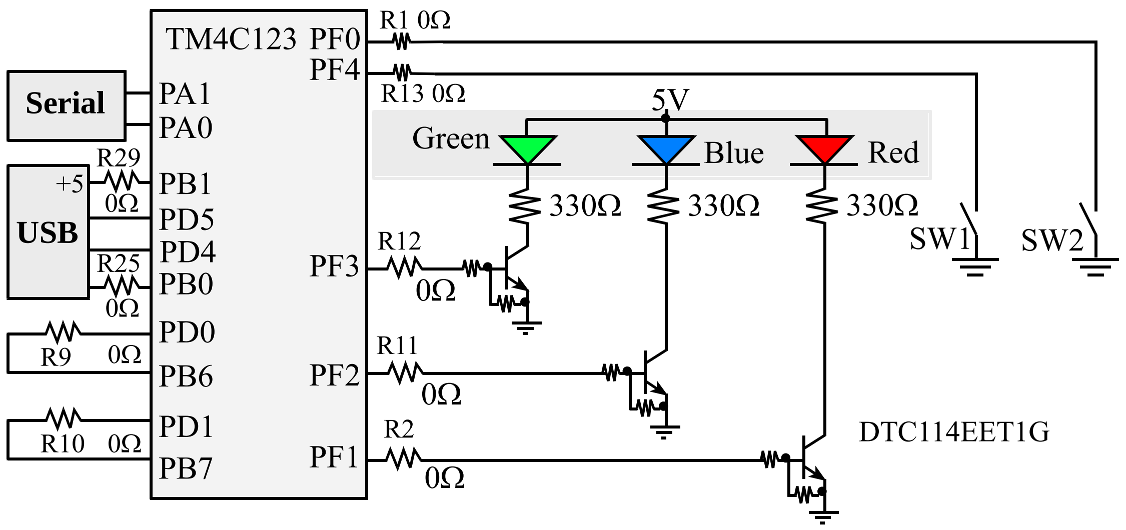
1. Modify the C program in the project *my_switch* created by yourself in Section C. In the modified system, when **SW2** (connected with **PF0**) is pressed, the **red LED** (connected with **PF1**) is turned **off**, otherwise the **red LED** is **on**. You need to modify the configurations in both the *PortFunctionInit()* function and the *main()* function. In your lab report, please include the following:

- The flow chart of the system
- Explanation of your modifications
- The C program listing with detailed comments for each line of code.
- The execution results of your program (your observations).

2. Create a new project *my_toggle* by following the instructions of creating a new project in Section C. Copy the code in the file *toggle.c* in the sample project *toggle* and paste it in your created project. Modify the code to make the **green LED** (connected with **PF3**) flash. Also try to make the LED flash **5 times slower**. You need to modify the configurations in both the *PortFunctionInit()* function and the *main()* function. In your lab report, please include the following:

- The flow chart of the system
- Explanation of your modification
- The C program listing with detailed comments for each line of code.
- The execution results of your program (your observations).

Reference: The figure below shows the LED and switch interfaces on the LaunchPad.


TM4C123
PF0
PF4
R1 0Ω
R13 0Ω
Serial
PA1
PA0
R29
+5
0Ω
USB
PB1
PD5
PD4
R25
PB0
0Ω
PD0
R9
0Ω
PB6
PD1
R10
0Ω
PB7
5V
Green
Blue
Red
330Ω
330Ω
330Ω
SW1
SW2
PF3
R12
0Ω
PF2
R11
0Ω
PF1
R2
0Ω
DTC114EET1G