Lab 9: TM4C Microcontroller GPIO Ports

Name: **Roll Number:**

Learning Objectives:

- Understand the binary number system, hexadecimal representation and the representation of signed and unsigned integers.
- Know what is a parallel port
- Know how a pin can be either input or output as specified by the direction register
- Know the steps required to initialize a parallel port
- Know how to access I/O registers
- Know how to read data from an input port
- Know how to write data to an output port

Lab Objectives:

- To reflect the understanding of the Microcontroller Ports
- To write an Embedded C program to meet the requirement specified in the flowchart.
- To use the logic analyzer in the simulator

PART 1 – Try out an existing program

Purpose

When first learning a new programming language it is tradition to begin by running a program that outputs the message "Hello World". Later you will write your own programs, but in this lab you will run simply a program that we have written for you. The input and output on the microcontroller comes from physical devices like switches and LEDs. Consequently, our "Hello World" will ask you to push a switch and observe an LED. The purpose of this lab is to work through the process of configuring the development system for the microcontroller board, and to learn how we will be grading labs in this course.

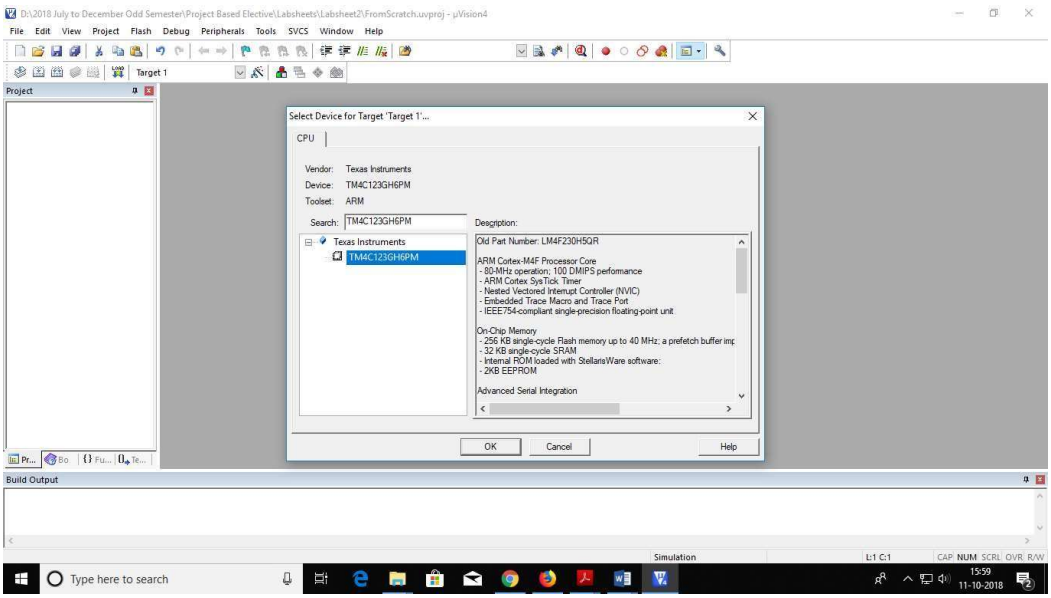
System Requirements

The system will have two inputs and three outputs. The inputs are switches called SW1 and SW2, which are connected to PF4 and PF0 respectively. Three outputs (PF3, PF2, PF1) are connected to one multi-color LED. The color of the LED is determined by the 3-bit value written to the outputs. Refer Appendix for further details.

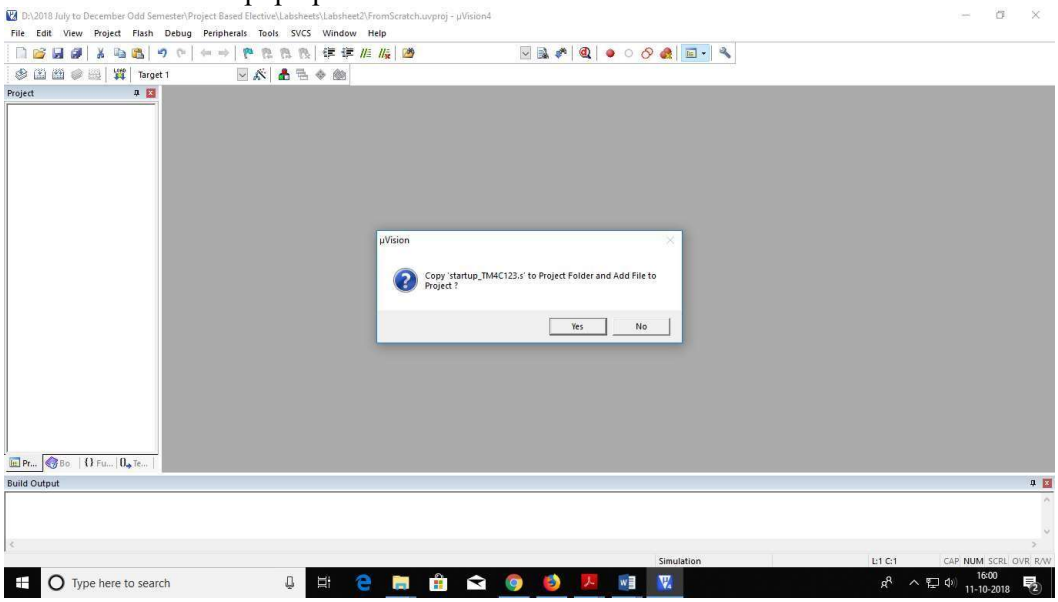
Phase 1: Build an Embedded C program

Steps to start the program from scratch in Keil IDE:

1. Create a new project with a suitable name.
2. Select Device for the target TM4C123GH6PM and select 'OK' button

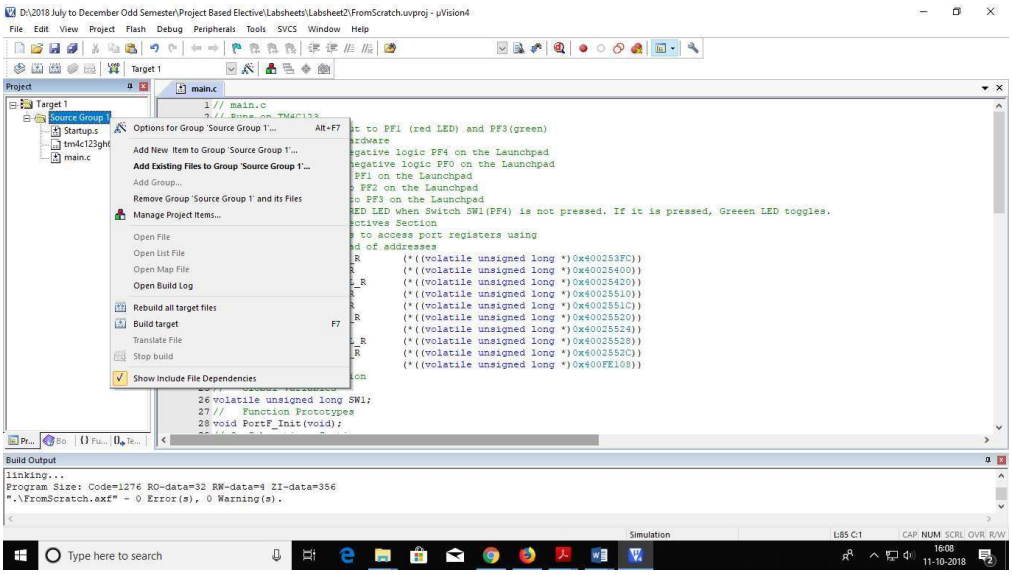


3. Give 'No' to the pop up window that comes



4. Add the Existing files Startup.s and tm4c123gh6pm.h to the Source Group1

5. Create a new file main.c and add it to the Source Group1



6. Now copy your written program in the main.c (attached document – sample program)
7. Build the program and check for 0 errors

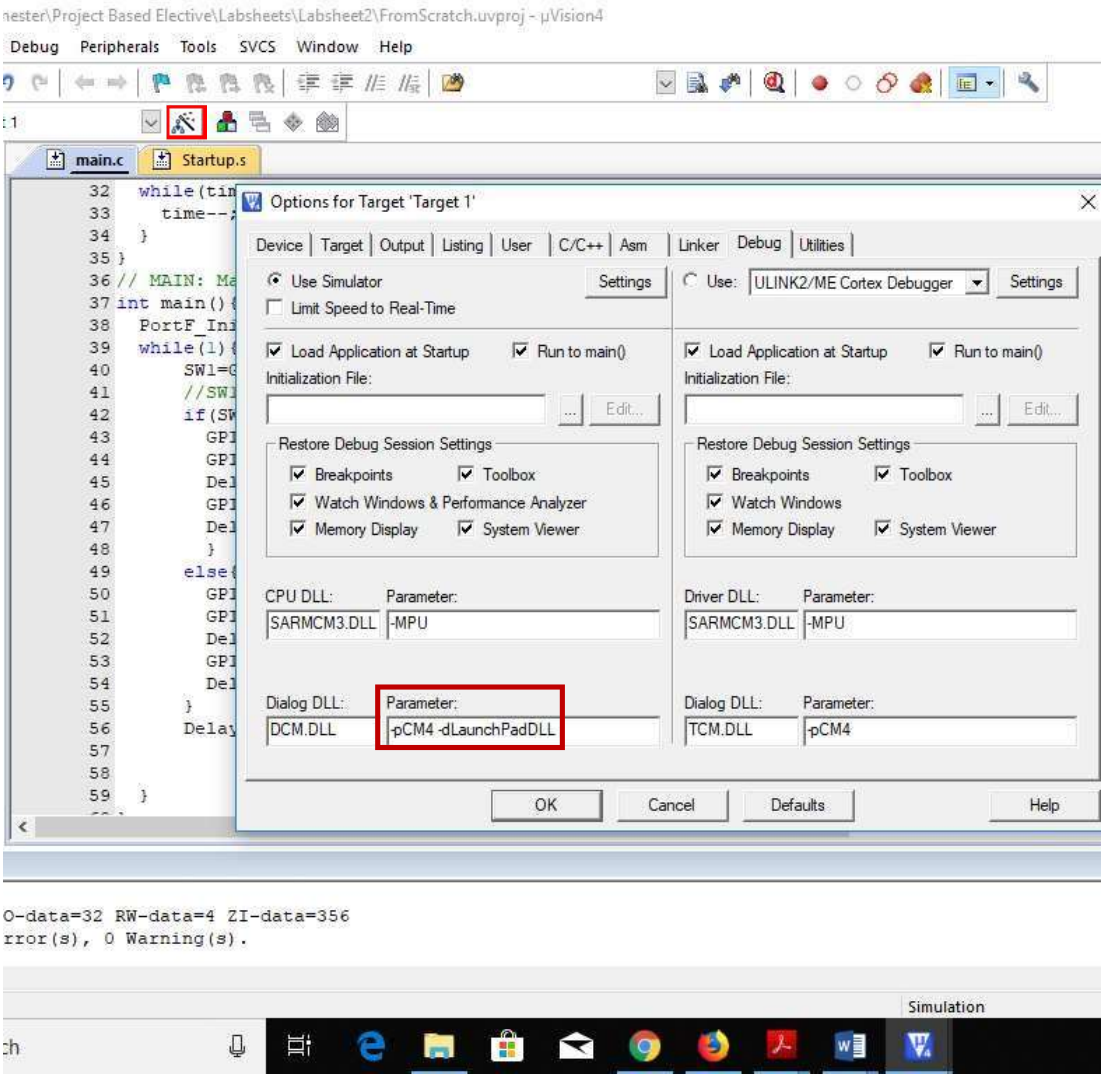
Phase 2: Check the output in the simulator

//Ensure that necessary DLL file are placed in C://keil/ARM/bin

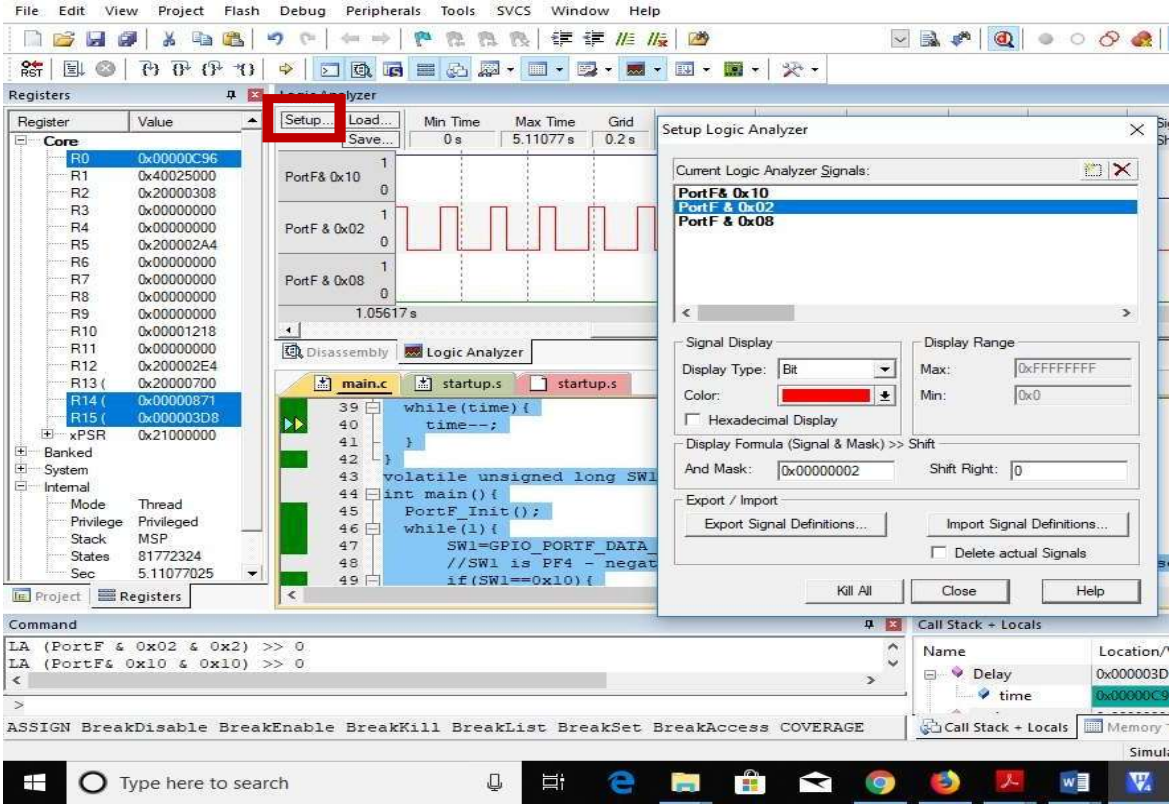
//also add this specifications in the window that comes after clicking the magic wand icon

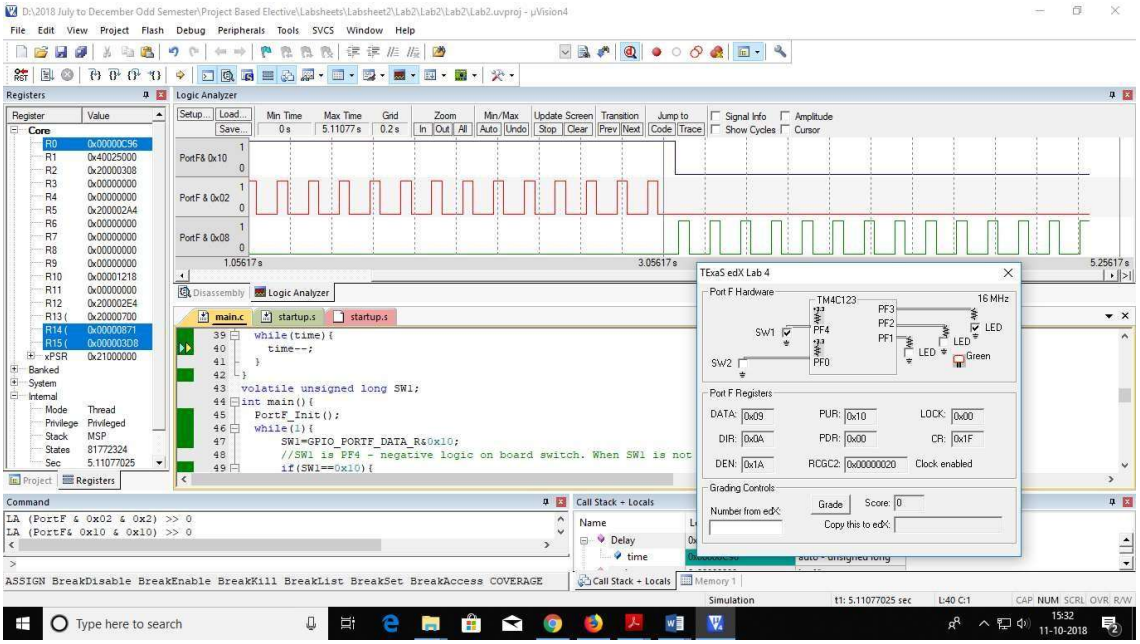
The parameter was earlier just –pCM4. Now add –d followed by correct DLL file name that is LaunchPadDLL. So –dLaunchPadDLL was added.

NOTE: If this DLL is not referred, you may get errors like DLL file missing, when you click Debug icon. Or you may not get the TexaS PortF window. Or you may get the following error like 'No Signals' during the setup of Logic Analyzer.

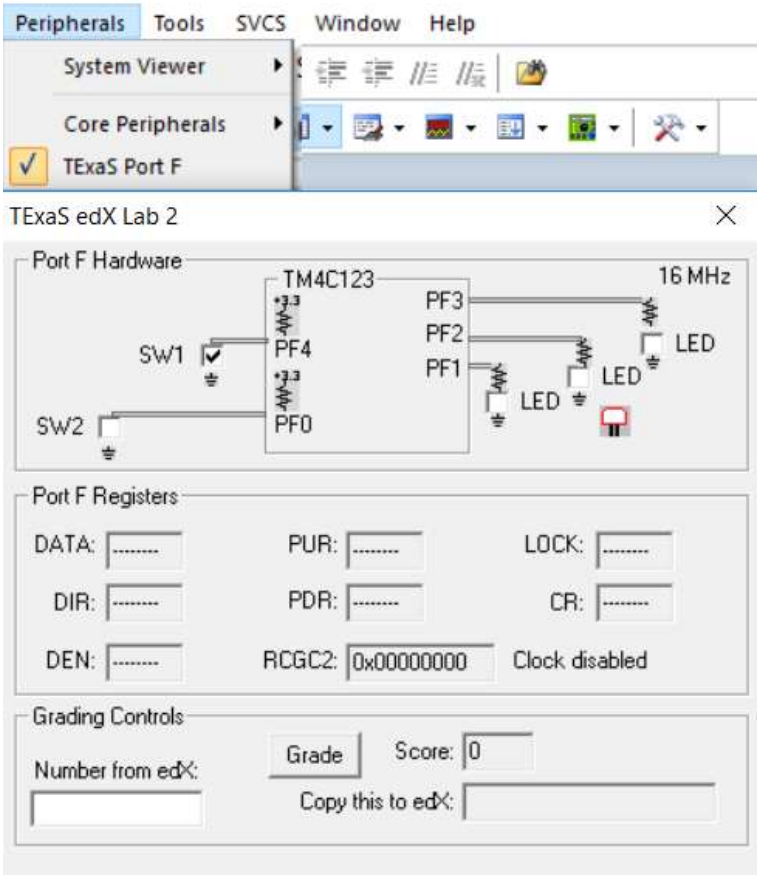


Ensure that Logic Analyzer is properly setup to digital





Build the program and start debugging it. Now select TExaS port F and select the switch to make it off and unselect the switch to turn it on.



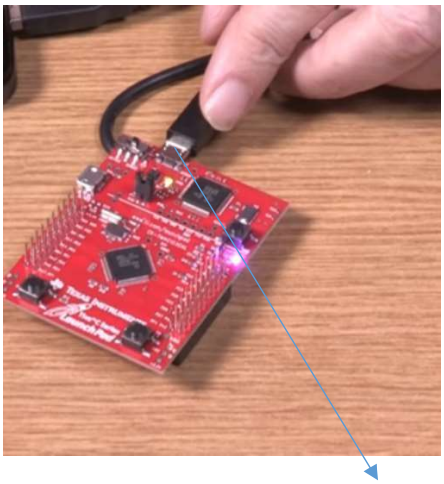
Note: The system has two input switches and three output LEDs. Algorithm shows the specifications of the system as the switch is pressed in the below table. A negative logic switch means the PF4 signal will be 1 (high, 3.3V) if the switch is not pressed, and the PF4 signal will be 0 (low, +0V) if the switch is pressed. A positive logic blue LED interface means if the software outputs a 1 to PF2 (high, +3.3V) the LED will turn ON, and if the software outputs a 0 to PF2 (low, 0V) the blue LED will be OFF. SW1 is on PF4 and SW2 is on PF0.

Switch Input	LED Output
Both switches SW1 and SW2 are pressed	The LED should be green
Just SW1 switch is pressed	The LED should be blue
Just SW2 switch is pressed	The LED should be red
Neither SW1 or SW2 is pressed	The LED should be off

Phase 3: Use Launchpad

//Follow the instructions to use the Launchpad effectively

Things not to do:

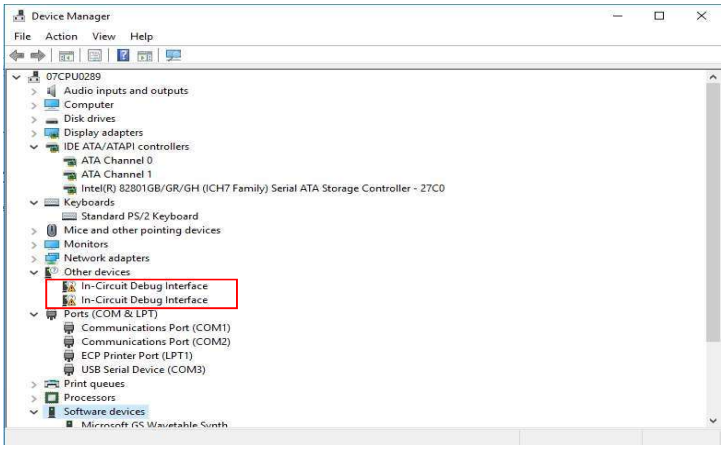


1. Do not unplug/rotate the other end of USB(USB ICDI) as pointed in the above figure.

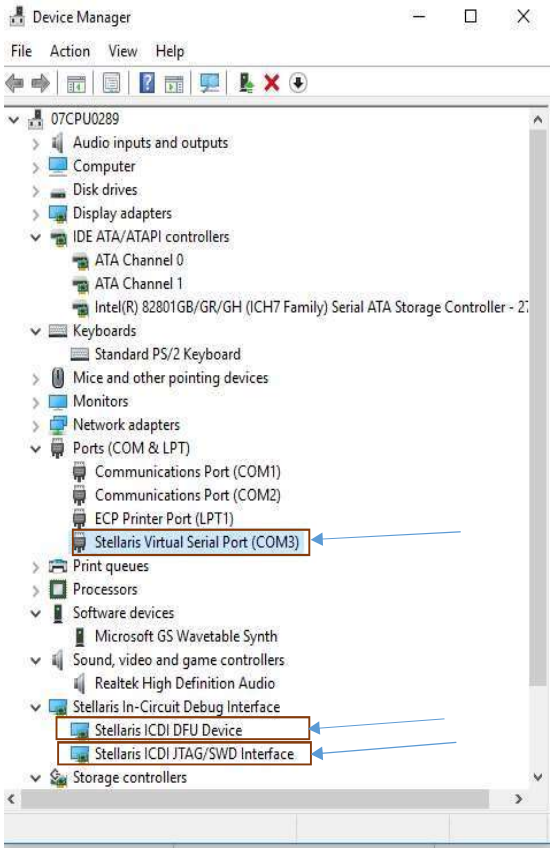
2. Do not touch with wet hands (avoid yourself getting hurted by electric shock)
3. Never disconnect/connect the wires in the Launchpad, when the power is on

Things to ensure:

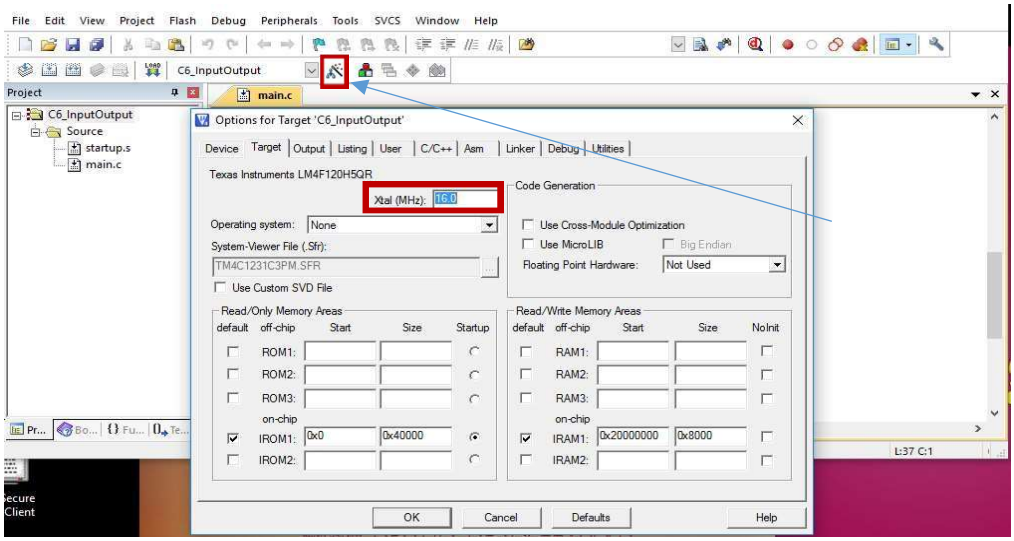
1. Check Device Drivers are installed successfully in your PC. The below screenshot shows that the device drivers are not installed.



Refer Labsheet6 to install the device drivers in your laptop successfully. The below screenshot shows that the device drivers are installed successfully.

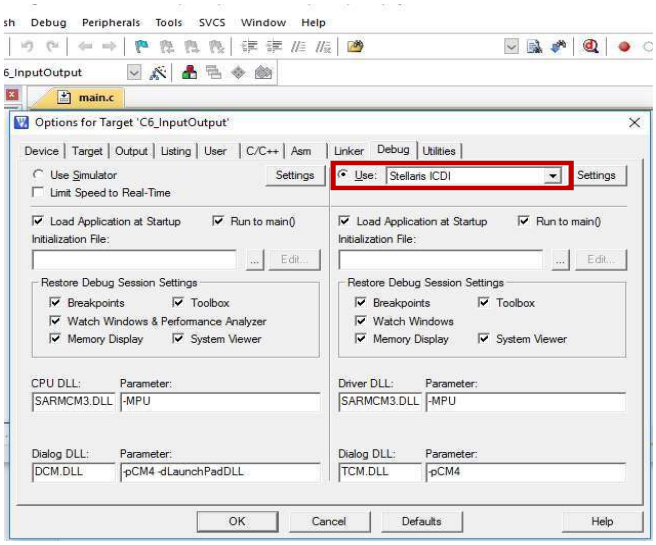


2. Ensure the following Options window – **Very IMPORTANT INSTRUCTION**

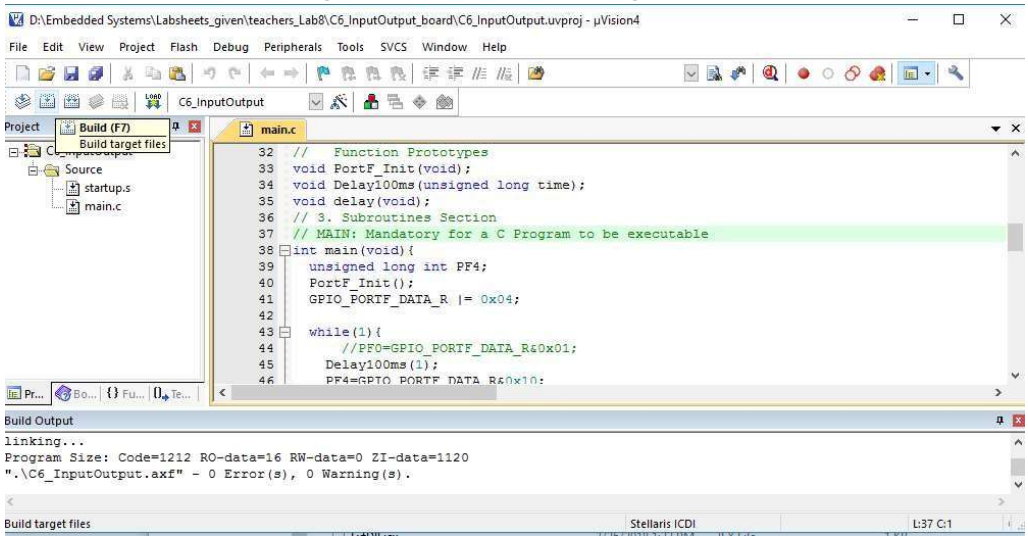


Click the icon pointed in the above screenshot. Ensure that Xtal value is 16 MHz as specified.

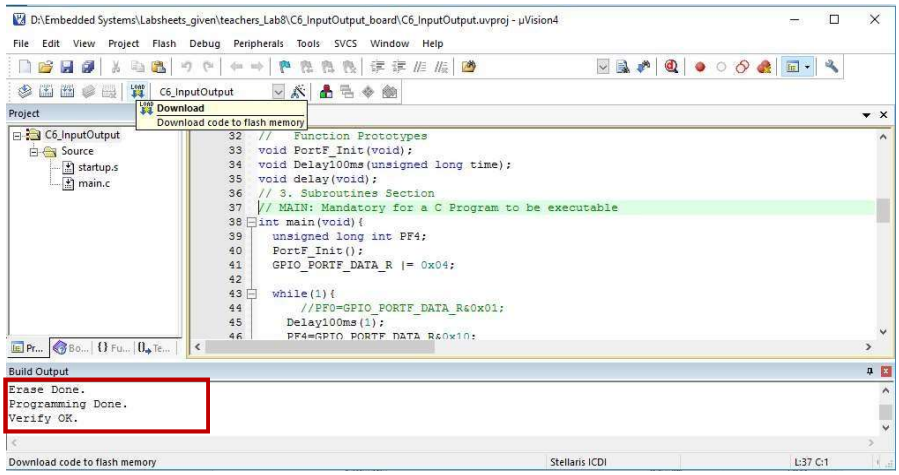
3. Steps in executing the successfully simulated code in the Launchpad
    a. Change the Debug option from simulator to stellaris ICDI as highlighted below



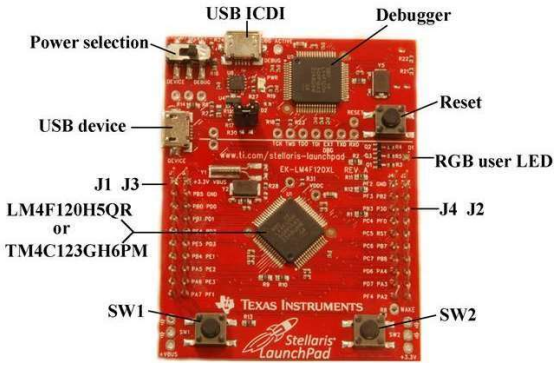
b. Build the code as usual to get zero errors and warnings



c. Press the Load icon to Download the code into ROM. Ensure that Build Output window displays all three highlighted lines.



d. Press Reset button for your processor to start executing from the first



PART 2 – Create your own program

1. Create a new project and write a delay function and main program in C to toggle Red LED. See the output in logic analyzer and in the Launchpad also.

```
void Delay(void){unsigned long volatile time;
  time = 727240*200;  // counter value for delay
  while(time){
          time--;
  }
}
```

2. Create a new project and write a main program in C that implements the input/output system mentioned. Overall functionality of this system is described in the following rules.

1) Make PF2 an output and make PF4 an input (enable PUR for PF4).

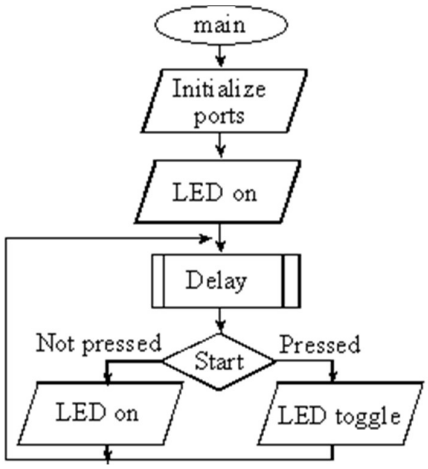
2) The system starts with the LED ON (make PF2 =1).

3) Delay for some random time using while loop

4) If the switch is pressed (PF4 is 0), then toggle the LED once, else turn the LED ON.

5) Repeat steps 3 and 4 over and over.

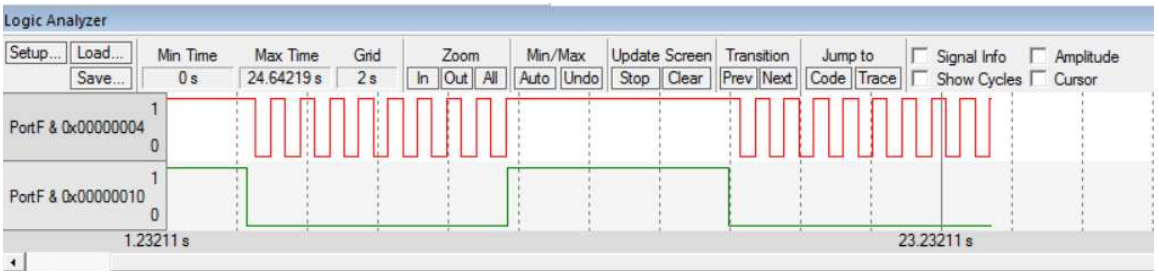
Pseudo code and flowchart are shown below, illustrating the basic steps for the system. For this you access the entire I/O port using GPIO_PORTF_DATA_R.



main **Turn on the clock for Port F**
Go to GPIO_Init
loop **Delay about 100 ms**
Read the switch and test if the switch is pressed
If PF4=0 (the switch is pressed),
toggle PF2 (flip bit from 0 to 1, or from 1 to 0)
If PF4=1 (the switch is not pressed),
set PF2, so LED is ON
Go to loop

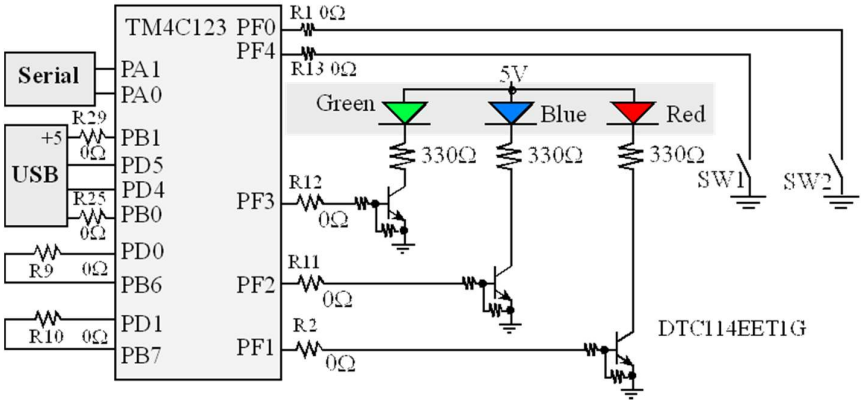
GPIO_Init **Clear the PF4 and PF2 bits in Port F AMSEL to disable analog**
Clear the PF4 and PF2 bit fields in Port F PCTL to configure as GPIO
Set the Port F direction register so
PF4 is an input and
PF2 is an output
Clear the PF4 and PF2 bits in Port F AFSEL to disable alternate functions
Set the PF4 and PF2 bits in Port F DEN to enable digital
Set the PF4 bit in Port F PUR to activate an internal pullup resistor
Set the PF2 bit in Port F DATA so the LED is initially ON

Explore Logic Analyzer tool available in simulator mode for the PF4 and PF2 signals as shown below. Logic Analyzer tools helps in debugging digital signals. More than 128 digital signals can be used in the logic analyzer.

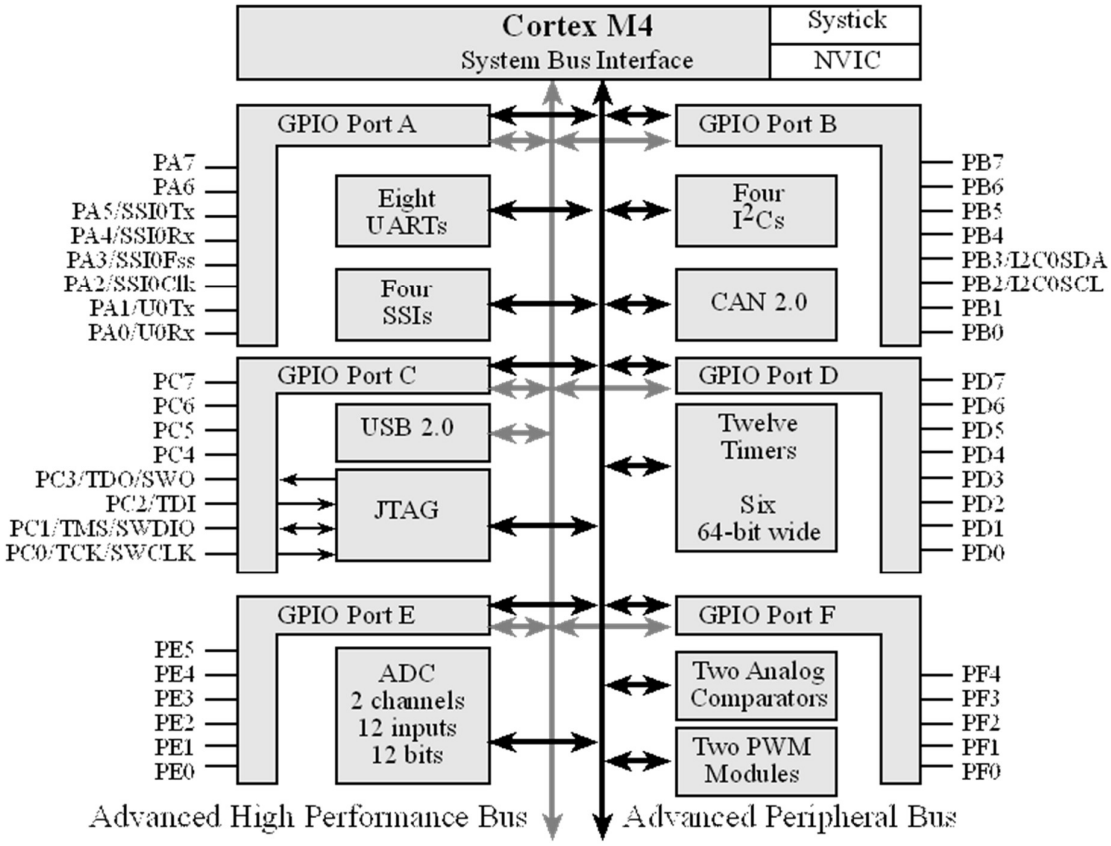


APPENDIX

The following circuit is used for all the above problems. SW1 is PF4 and SW2 is PF0. Red LED is PF1. Blue LED is PF2. Green LED is PF3.



IO Architecture in TM4C123 Microcontroller



You can refer to this datasheet for your Lab exercises.

Address	7	6	5	4	3	2	1	0	Name
$400F.E108	--	--	GPIOF	GPIOE	GPIOD	GPIOC	GPIOB	GPIOA	SYSCTL_RCGC2_R
$4000.43FC	DATA	DATA	DATA	DATA	DATA	DATA	DATA	DATA	GPIO_PORTA_DATA_R
$4000.4400	DIR	DIR	DIR	DIR	DIR	DIR	DIR	DIR	GPIO_PORTA_DIR_R
$4000.4420	SEL	SEL	SEL	SEL	SEL	SEL	SEL	SEL	GPIO_PORTA_AFSEL_R
$4000.4510	PUE	PUE	PUE	PUE	PUE	PUE	PUE	PUE	GPIO_PORTA_PUR_R
$4000.451C	DEN	DEN	DEN	DEN	DEN	DEN	DEN	DEN	GPIO_PORTA_DEN_R
$4000.4524	1	1	1	1	1	1	1	1	GPIO_PORTA_CR_R
$4000.4528	0	0	0	0	0	0	0	0	GPIO_PORTA_AMSEL_R
$4000.53FC	DATA	DATA	DATA	DATA	DATA	DATA	DATA	DATA	GPIO_PORTB_DATA_R
$4000.5400	DIR	DIR	DIR	DIR	DIR	DIR	DIR	DIR	GPIO_PORTB_DIR_R
$4000.5420	SEL	SEL	SEL	SEL	SEL	SEL	SEL	SEL	GPIO_PORTB_AFSEL_R
$4000.5510	PUE	PUE	PUE	PUE	PUE	PUE	PUE	PUE	GPIO_PORTB_PUR_R
$4000.551C	DEN	DEN	DEN	DEN	DEN	DEN	DEN	DEN	GPIO_PORTB_DEN_R
$4000.5524	1	1	1	1	1	1	1	1	GPIO_PORTB_CR_R
$4000.5528	0	0	AMSEL	AMSEL	0	0	0	0	GPIO_PORTB_AMSEL_R
$4000.63FC	DATA	DATA	DATA	DATA	JTAG	JTAG	JTAG	JTAG	GPIO_PORTC_DATA_R
$4000.6400	DIR	DIR	DIR	DIR	JTAG	JTAG	JTAG	JTAG	GPIO_PORTC_DIR_R
$4000.6420	SEL	SEL	SEL	SEL	JTAG	JTAG	JTAG	JTAG	GPIO_PORTC_AFSEL_R
$4000.6510	PUE	PUE	PUE	PUE	JTAG	JTAG	JTAG	JTAG	GPIO_PORTC_PUR_R
$4000.651C	DEN	DEN	DEN	DEN	JTAG	JTAG	JTAG	JTAG	GPIO_PORTC_DEN_R
$4000.6524	1	1	1	1	JTAG	JTAG	JTAG	JTAG	GPIO_PORTC_CR_R
$4000.6528	AMSEL	AMSEL	AMSEL	AMSEL	JTAG	JTAG	JTAG	JTAG	GPIO_PORTC_AMSEL_R
$4000.73FC	DATA	DATA	DATA	DATA	DATA	DATA	DATA	DATA	GPIO_PORTD_DATA_R
$4000.7400	DIR	DIR	DIR	DIR	DIR	DIR	DIR	DIR	GPIO_PORTD_DIR_R
$4000.7420	SEL	SEL	SEL	SEL	SEL	SEL	SEL	SEL	GPIO_PORTD_AFSEL_R
$4000.7510	PUE	PUE	PUE	PUE	PUE	PUE	PUE	PUE	GPIO_PORTD_PUR_R
$4000.751C	DEN	DEN	DEN	DEN	DEN	DEN	DEN	DEN	GPIO_PORTD_DEN_R
$4000.7524	CR	1	1	1	1	1	1	1	GPIO_PORTD_CR_R
$4000.7528	0	0	AMSEL	AMSEL	AMSEL	AMSEL	AMSEL	AMSEL	GPIO_PORTD_AMSEL_R
$4002.43FC			DATA	DATA	DATA	DATA	DATA	DATA	GPIO_PORTE_DATA_R
$4002.4400			DIR	DIR	DIR	DIR	DIR	DIR	GPIO_PORTE_DIR_R
$4002.4420			SEL	SEL	SEL	SEL	SEL	SEL	GPIO_PORTE_AFSEL_R
$4002.4510			PUE	PUE	PUE	PUE	PUE	PUE	GPIO_PORTE_PUR_R
$4002.451C			DEN	DEN	DEN	DEN	DEN	DEN	GPIO_PORTE_DEN_R
$4002.4524			1	1	1	1	1	1	GPIO_PORTE_CR_R
$4002.4528			AMSEL	AMSEL	AMSEL	AMSEL	AMSEL	AMSEL	GPIO_PORTE_AMSEL_R
$4002.53FC				DATA	DATA	DATA	DATA	DATA	GPIO_PORTF_DATA_R
$4002.5400				DIR	DIR	DIR	DIR	DIR	GPIO_PORTF_DIR_R
$4002.5420				SEL	SEL	SEL	SEL	SEL	GPIO_PORTF_AFSEL_R
$4002.5510				PUE	PUE	PUE	PUE	PUE	GPIO_PORTF_PUR_R
$4002.551C				DEN	DEN	DEN	DEN	DEN	GPIO_PORTF_DEN_R
$4002.5524				1	1	1	1	CR	GPIO_PORTF_CR_R
$4002.5528				0	0	0	0	0	GPIO_PORTF_AMSEL_R

	31-28	27-24	23-20	19-16	15-12	11-8	7-4	3-0	
$4000.452C	PMC7	PMC6	PMC5	PMC4	PMC3	PMC2	PMC1	PMC0	GPIO_PORTA_PCTL_R
$4000.552C	PMC7	PMC6	PMC5	PMC4	PMC3	PMC2	PMC1	PMC0	GPIO_PORTB_PCTL_R
$4000.652C	PMC7	PMC6	PMC5	PMC4	0x1	0x1	0x1	0x1	GPIO_PORTC_PCTL_R
$4000.752C	PMC7	PMC6	PMC5	PMC4	PMC3	PMC2	PMC1	PMC0	GPIO_PORTD_PCTL_R
$4002.452C			PMC5	PMC4	PMC3	PMC2	PMC1	PMC0	GPIO_PORTE_PCTL_R
$4002.552C				PMC4	PMC3	PMC2	PMC1	PMC0	GPIO_PORTF_PCTL_R
$4000.6520	LOCK (write 0x4C4F434B to unlock, other locks) (reads 1 if locked, 0 if unlocked)								GPIO_PORTC_LOCK_R
$4000.7520	LOCK (write 0x4C4F434B to unlock, other locks) (reads 1 if locked, 0 if unlocked)								GPIO_PORTD_LOCK_R
$4002.5520	LOCK (write 0x4C4F434B to unlock, other locks) (reads 1 if locked, 0 if unlocked)								GPIO_PORTF_LOCK_R