Lab 1 : Helper Document

January 14, 2018

1 Relevant Theory

1. Refer *TM4C123G_LaunchPad_Workshop_Workbook* available on course web page.Texas Instrument "TM4C123G LaunchPad Workshop - Student Guide and Lab Manual".Refer Chapter-3 "Introduction to TivaWare, Initialization and GPIO" of the Manual.

2. Use TivaWare Peripheral Driver Library, an API written by Texas Instrument to access different peripherals and functionality of ARM Cortex-M based micro controller.

2 TivaWare Essentials

For this lab, you need to go through the following Tivaware api calls from the GPIO section of the tivaware manual.

1. *GPIOPinTypeGPIOOutput* : This function call configures a particular pin or port as an output. Useful, when you wish to interface an LED.

2. *GPIOPinTypeGPIOInput* : This function call configures a particular pin or port as an input. Used when an input device like a switch needs to be interfaced.

3. *GPIOPadConfigSet* : This function call is used to enable a pullup when a pin is used an input. Useful, when a switch is to be interfaced. You can go through this short write up about interfacing switches present on this link : Interfacing switches

4. *GPIOPinRead* : This function call returns the value read on a particular pin.

5. *GPIOPinWrite* : This function call is used to write values to a pin or port set as output. Useful, when turning ON/OFF the on board LED.

   Note : The above explanation is not all exhaustive, you need to read the relevant parts of the tivaware manual.

Apart from this you need to know read **TivaWare™, Initialization and GPIO** of the workshop manual to better understand GPIO on Tivalaunchpad.

3 Assembly of Hardware

In this Lab external hardware connections are not required.
The PORT pins which will be required are

1. PF0 - SW1
2. PF1 - R of RGB
3. PF2 - B of RGB
4. PF3 - G of RGB
5. PF4 - SW2

All these connections are present on the board.

4 Procedure

4.1 Procedure for Part 1

1. Include all the required header files. Ensure that the following header file are present:

```
#include <stdio.h>
#include <stdint.h>
#include <stdbool.h>
#include "inc/hw_types.h"
#include "inc/hw_gpio.h"
#include "inc/hw_memmap.h"
#include "driverlib/sysctl.h"
#include "driverlib/gpio.h"
#include "inc/hw_ints.h"
```

2. Declare a variable to monitor the status of switch SW1.
3. Enable all the peripherals which will be used. (PORT F,System Clock)
4. Configure PF4 as input and PF1,PF2 and PF3 as output.
5. When the switch SW1 is pressed the value at PF4 becomes 0. Read the value of PF4.
6. When this value is equal to 0 turn ON R,G or B LED by writing 1 to its corresponding pin

4.2 Procedure for Part 2

1. Include all the required header files
2. Enable PORTF and System Clock.
3. Configure PF0as input pin. PF0 is locked by default. Use the following code to unlock PF0

```
HWREG(GPIO_PORTF_BASE + GPIO_O_LOCK)= GPIO_LOCK_KEY;
HWREG(GPIO_PORTF_BASE + GPIO_O_CR) |= 0x01;
HWREG(GPIO_PORTF_BASE + GPIO_O_LOCK)= 0;
```

4. Use a variable *sw2status* to record the number of times switch SW2 is pressed
5. In debug perspective double click on variable sw2status. Right click on it and select Add Watch Expression. Select continuous refresh option which is at top in Watch Window.

4.3 Procedure for Part 3

1. Include all the required header files.
2. Enable System Clock and PORTF.
3. Configure PF0 and PF4 as input and PF1, PF2 and PF3 as output.
4. Unlock pin PF0.
5. Use SysCtlDelay(6700000) to generate delay of around 0.5 seconds.
6. Monitor the status of SW1 and SW2. Whenever SW1 is pressed the delay should change from 0.5 =>1 =>2 seconds. Whenever SW2 is pressed colour of RGB should change from R =>G =>B.