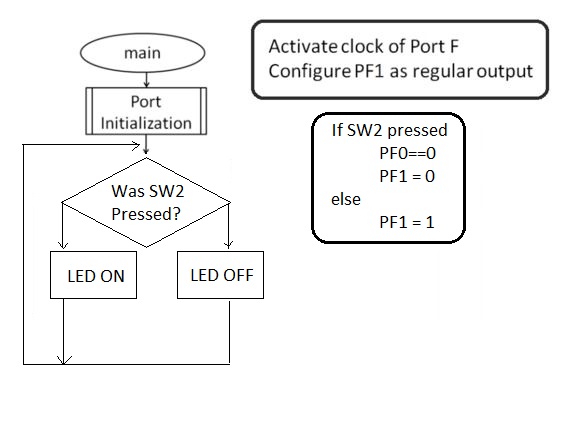
TM4C123 Microcontroller I/O Ports Switches and LED Interfaces

Farnam Adelkhani

***Questions:***

***Question #1:***

The goal is to modify the C program in the project my\_switch. 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. Our group modified the configurations in both the PortFunctionInit() function and the main() function.



Explanation of modifications:

The original program, my\_switch, was setup to turn the blue led (PF2) located on the Ti launchpad on when the switch SW2 (PF0) is pressed. This program was modified to instead turn the red LED (PF1) off when SW2 is pressed and otherwise just allow the red led to remain on.

Changes made to the syntax began by defining the reserved cluster for the mask. Next modification is to change the route of the output to PF1 which is the red led, previously the output was set to PF2 which triggers the Blue led. Lastly our group modified the GPIO\_PORTF\_DATA\_R &=(); line to turn the red led off when SW2 is pressed. All new programming is marked with "!".

Q1: my\_switch code

#include <stdint.h>

#include "inc/tm4c123gh6pm.h"

#define RED\_MASK 0x02

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//

//!

//! A very simple example that interfaces with the blue 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.

//

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//!This program has been modified to SW2 Press = Red LED Off

//!When SW2 is not being pressed the Red LED will remain turned on

//

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 PF1 (red LED) as output

GPIO\_PORTF\_DIR\_R |= 0x02;

// Set the direction of PF0 (SW2) as input by clearing the bit

GPIO\_PORTF\_DIR\_R &= ~0x01;

//!Enable both PF1 and PF0 for digital function.

GPIO\_PORTF\_DEN\_R |= 0x03;

//Enable pull-up on PF0

GPIO\_PORTF\_PUR\_R |= 0x01;

}

int main(void)

{

//initialize the GPIO ports

PortFunctionInit();

//

// Loop forever.

//

while(1)

{

if((GPIO\_PORTF\_DATA\_R&0x01)==0x00) //SW2 is pressed

{

//Turn off the LED

GPIO\_PORTF\_DATA\_R &= ~0x02;

}

else

{

//Turn on the LED

GPIO\_PORTF\_DATA\_R |= 0x02;

}

}

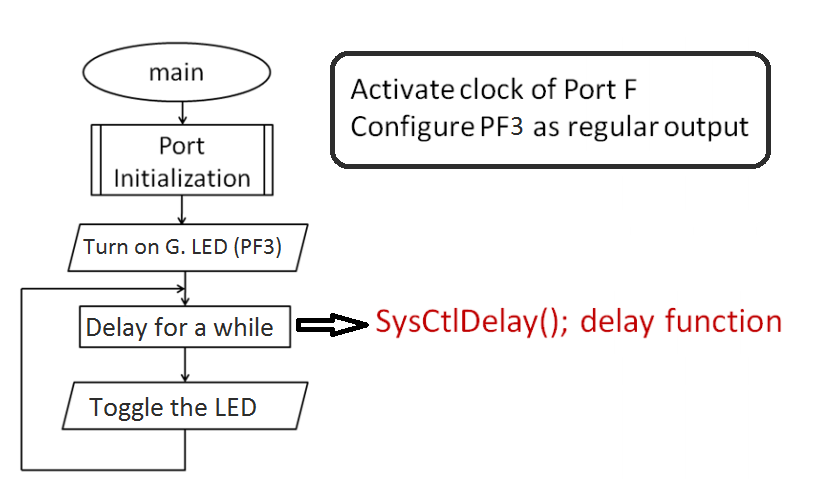
}

Execution results:

As a result when the program is compiled onto the launchpad the red led remains on until the switch is pressed which causes the led to turn off until the switch is again released.

***Question #2:***

The objective is to create a new project my\_toggle and modify the code to make the green LED (connected with PF3) flash. While also trying to make the LED flash 5 times slower. We modified the configurations in both the PortFunctionInit() function and the main() function.



Explanation of modifications:

The program was setup to simply toggle the red led (PF1) on the launchpad after the preset delay. The program was modified to instead toggle the green led (PF3) after a delay five times longer than the original delay.

As in the previous section our group was required to define the green mask in the syntax. Next mod is to set the green led as the output (PF3), as well as enabling the pin. Finally changes in the main section are made to accommodate the green LED turning on as well as the system delay taking five times longer than previously programmed. All new programming is marked with "!".

Q2: my\_toggle code

#include <stdint.h>

#include <stdbool.h>

#include "inc/tm4c123gh6pm.h"

#include "driverlib/sysctl.h"

#define GREEN\_MASK 0x08

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//

//!

//! A very simple example that toggles the on-board red LED using direct register

//! access

//

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void

PortFunctionInit(void)

{

//

volatile uint32\_t ui32Loop;

// Enable the GPIO port that is used for the on-board LED.

//

SYSCTL\_RCGC2\_R = SYSCTL\_RCGC2\_GPIOF;

//

// Do a dummy read to insert a few cycles after enabling the peripheral.

//

ui32Loop = SYSCTL\_RCGC2\_R;

//

//Enable the GPIO pin for the green LED (PF3). Set the direction as output, and

//enable the GPIO pin for digital function.

GPIO\_PORTF\_DIR\_R |= 0x08;

GPIO\_PORTF\_DEN\_R |= 0x08;

}

int main(void)

{

//initialize the GPIO ports

PortFunctionInit();

// Turn on the LED.

GPIO\_PORTF\_DATA\_R |= 0x08;

//

// Loop forever.

//

while(1)

{

// Delay for a bit.

SysCtlDelay(2000000\*5);

// Toggle the LED.

GPIO\_PORTF\_DATA\_R ^=GREEN\_MASK;

}

}

Execution results:

As a result the ti launchpad is toggling the green led on and off at an interval ~5 times longer than originally set in the default program.

Discussion and Suggestions:

Through the course of this experiment our lab group has learned how to create a new project, rather than just opening an existing project. When creating a new project the user must pay close attention to properly allow the new .uvproj to access both the launchpad and the required libraries.

We will include a brief overview of the steps taken for our own benefit in regard to quickly setting up new projects in the future. After creating the new project file it is necessary to select the appropriate device you are intending the project to be used on; in this case the TM4C123GH6PM. Second important step is to add the startup code to the project. In this case and for all future labs we will be using the file **startup\_rvmdk.s.** Third important step is to include the **driverlib.lib** file to the project in order to allow the user to implement the functions in the TivaWare library. Tivaware of course being an extensive suite of software tools designed to simplify and speed up development. The last important step is to add a new C program file.

Next it is necessary to edit project options and the lab recommends opening another instance of keil to match up the settings with an existing project.

Having learned how to create and setup a new project, the goal is now to modify the C programming of my\_switch and my\_toggle to perform the actions specified to us in the lab assignment.

Lab assignment overlaps with the lecture and is generally pretty straightforward. Only recommendation is providing more details about how and what to modify in the program to get the desired settings, but that was the challenge of this lab.