TI ARM Lab 4
Digilent Orbit Board Introduction and LEDs

Acknowledgements

Lab Summary

This lab is very similar to the third lab but it is accomplished using the Digilent Corporation's Orbit daughter card. This card was built for academic purposes for learning how to do small projects (with a pathway to more complex projects) and can be found as a combination with the Tiva board on the Digilent website.

Lab Goal

The goal of this lab is to continue to build upon the skills learned from previous labs. This lab helps the student to continue to gain new skills and insight on the C code syntax and how it is used in the TI implementation of the ARM processor with the addition of a daughter board. Each of these labs add upon the previous labs and it is the intention of the authors that they will build with each lab a better understanding of the ARM processor and basic C code. Even though these tutorials assume the student has not entered with a knowledge of C code, it is the desire that by the time the student completes the entire series of tutorials that they will have a sufficient knowledge of C code so as to be able to accomplish useful projects on the ARM processor.

Learning Objectives

The student should continue to become familiar with the complier and understand the use and modification of a "main.c" file. The student should also begin to be exposed to C "functions."   In addition, the Digilent Orbit board is introduced in this tutorial.   This board provides expanded capabilities to the Tiva platform for doing more and varied projects during the workshop.   At the simplest level, it provides additional LEDs, switches and buttons.

Grading Criteria

N/A

Time Required

Approximately one hour

Lab Preparation

It is highly recommended that the student read through this procedure once before actually using it as a tutorial. By understanding the final goal it will be easier to use this as a tutorial and as a learning guide. This lab creates the entire project from scratch.

Equipment and Materials

Access to Tiva TM4C123G LaunchPad software and evaluation kit (EK-RM4C123GXL) as well as the Digilent Orbit board. It is assumed that the student has already completed Lab 3 and the software is installed properly. Ensure that the boards are plugged in before start of the tutorial.

Software needed	Quantity
The software requirements from Lab 3 are still applicable here.	1

Hardware needed	Quantity
The hardware required is the TI Tiva LaunchPad Kit and the Digilent Corporation's Orbit Daughter card	1

Additional References

The Evaluation Board user's manual is on this web site: http://datasheet.octopart.com/EK-TM4C123GXL-Texas-Instruments-datasheet-15542121.pdf and the manuals for the Digilent orbit board are located at http://digilentinc.com/Products/Detail.cfm?NavPath=2,396,1181&Prod=ORBIT-BOOSTER.
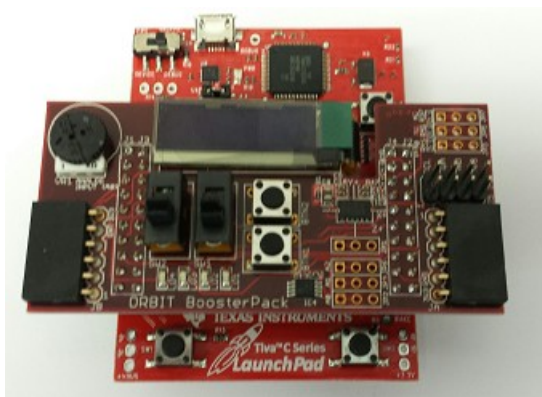
Lab Procedure

Figure 1. ARM and ORBIT Combination

This picture of the correct way to mate the Tiva LaunchPad and the Digilent Orbit boards together. Please do so at this point and connect them as shown in Figure 1.

Figure 2. Code Composer Icon

Launch Code Composer and where prompted, chose the workspaces location to store your project (as shown in Figure 3).
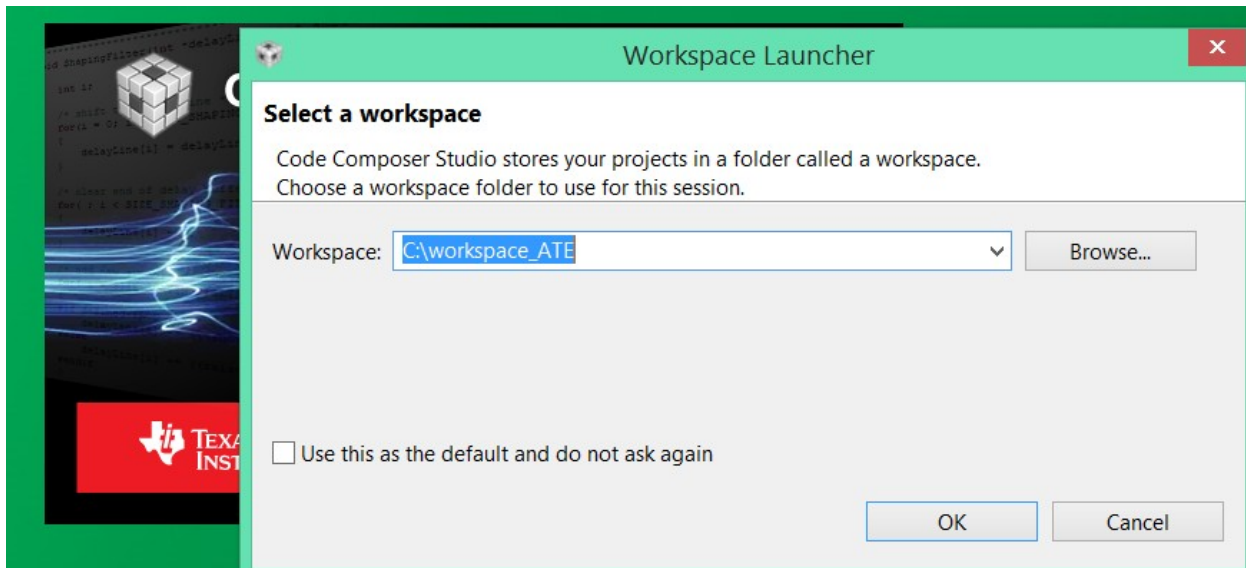
Figure 3. Workspace Selection

Since the installer for the workshop has been run prior to this, the user will be presented with the following view (Figure 4) where all lab projects exist.
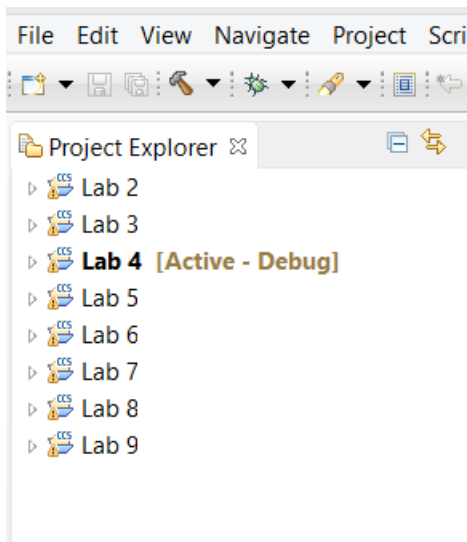

Figure 4. CCS Starting Point

The laboratory material is created to have the students type in a lot of the code.   Only by typing the code and then debugging the errors will a user ever really understand how to do projects.   For the sake of this activity, the source code is provided at the end of the tutorial.   In Lab 4, open main.c. Then either type in all the code from attachment 2 or copy and paste in the code from Attachement 2 into main.c.

Figure 5. Header Files

Shown in Figure 5 is a collection of header files.   Header files contain all the functions that you will need to have available to do any project with the device.   From here until the end of time, you should include these nine header files in every project that you do with this device.   Remember that inclusion of all these lines of code doesn't necessarily mean that all the code will be part of the final project. The header files contain all the "possible" functions that will be necessary and when the compiler actually accomplishes the work, then it will only include the functions that are actually required.

Now it is time to introduce come new concepts in C code syntax. Here are three rules that the authors always follow. If they are memorized, it will make it easier to quickly work through new code when it is presented.

- Any time you see something where all letters are capitalized the authors will use this to identify a "define" statement for a preprocessor.
- If the first letter only is capitalized, it is what is known as a function cell.
- If all of the letters are lower case, it is known as a variable.

What is beginning to be accomplished in this tutorial is the additional use of cleaner and more elegant code functions in the projects.   Lab 3 accomplished much of what needed to be accomplished through a very brute force methodology.

The ARM chip has thousands of functions built into it for making code cleaner and more efficient.   The key to remember when working with the ARM processor is that it is ultimately designed to be highly efficient and to be able to operate for months at a time on batteries.   To accomplish this, the device is virtually powered down when it is turned on.   The device has a series of ports that then connect to external devices such as UARTs and I2C busses.   EVERY TIME that a new project is started, the first thing than must be done is to first enable the ports and then set up the pins.

```
// Enable Peripheral ports for input/ output
SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOC);  //PORTC
GPIOPinTypeGPIOOutput(GPIO_PORTC_BASE, GPIO_PIN_6|GPIO_PIN_7); // LED 1 LED 2

SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOB);  //PORTB
GPIOPinTypeGPIOOutput(GPIO_PORTB_BASE, GPIO_PIN_5); // LED 4

SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOD);  //PORT D
GPIOPinTypeGPIOOutput(GPIO_PORTD_BASE, GPIO_PIN_6); // LED 3
GPIOPinTypeGPIOInput(GPIO_PORTD_BASE, GPIO_PIN_2); // BTN 1

SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);  //PORT F
GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3); // RGB LED on Launchpad

SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOA);  //PORT A
GPIOPinTypeGPIOInput(GPIO_PORTA_BASE, GPIO_PIN_6); // Switch 2
```

Figure 6. Enabling Ports and Configuring Pins

As shown in Figure 6, the first port to be enabled is port C (GPIOC). Once the port is enabled, the next step is to configure the pins. In this case, pins 6 and 7 are enabled as output pins. As can be shown by the bottom line, Port A, Pin 6 is configured as an input pin. If you review the picture in attachment 2, you can see pictorially where all the ports and pins run to.

```
57
58          GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, 4); // LED Blue on Laun
59
60          // Cycle through the LEDs on the Orbit board
61          GPIOPinWrite(GPIO_PORTC_BASE, GPIO_PIN_6|GPIO_PIN_7, 0x40); // LED 1 on LED 2 Off
62          GPIOPinWrite(GPIO_PORTD_BASE, GPIO_PIN_6, 0x00); // LED 3 off, Note different PORT
63          GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_5, 0x00); // LED 4 off
64
65          SysCtlDelay(2000000); // Delay, Replaces for LOOP
66
67          GPIOPinWrite(GPIO_PORTC_BASE, GPIO_PIN_6|GPIO_PIN_7, 0x80); // LED 1 OFF, LED 2 on
68
69          SysCtlDelay(2000000); // Delay, Replaces for LOOP
70
71          GPIOPinWrite(GPIO_PORTC_BASE, GPIO_PIN_6|GPIO_PIN_7, 0x00); //LED 1 off LED 2 off
72          GPIOPinWrite(GPIO_PORTD_BASE, GPIO_PIN_6, 0x40); // LED 3 on
73
74          SysCtlDelay(2000000); // Delay, Replaces for LOOP
75
76          GPIOPinWrite(GPIO_PORTD_BASE, GPIO_PIN_6, 0x00); // LED 3 off
77          GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_5, 0x20); // LED 4 on
78
79          SysCtlDelay(2000000); // Delay, Replaces for LOOP
80
81
```

Figure 7. "IF" Work

In Figure 7, it is possible to see how much more elegant and clean the code is that what was used before. What can be seen is, if the switch two (left hand on orbit board) is thrown, write to the LEDs on port F. However, not all the LEDs are being written to. As can be seen in line 58, the three LEDs on the Tiva board are OR'd together. This is very similar to what is accomplished on line 61. Realize that "4" is the same as 0x4 in hex. If you hover over GPIO_PIN_1, you can see that the register value is 2 (0010). Green is 4 and blue is 8. Thus, to turn the light white (R, G, and B on) it is necessary to change the 4 to an 11 or 0x0E.

Another item to notice is the use of the SysCtlDelay function. This is much cleaner than the massive for loop used in previous labs.

```
111     while(GPIOPinRead(GPIO_PORTD_BASE, GPIO_PIN_2)){      // Listen for BTN 1
112
113         // All LED Blink
114         GPIOPinWrite(GPIO_PORTC_BASE, GPIO_PIN_6|GPIO_PIN_7, 0xFF);
115         GPIOPinWrite(GPIO_PORTD_BASE, GPIO_PIN_6, 0xFF);
116         GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_5, 0x20);
117         GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, 0xFF); // White Output
118
119         SysCtlDelay(2000000);
120
121         GPIOPinWrite(GPIO_PORTC_BASE, GPIO_PIN_6|GPIO_PIN_7, 0x00);
122         GPIOPinWrite(GPIO_PORTD_BASE, GPIO_PIN_6, 0x00);
123         GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_5, 0x00);
124         GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, 0x00); // White Output
125
126         SysCtlDelay(2000000);
```

Figure 8. Blinky Lights

As shown in Figure 8, if the button is pushed, we write outputs to all the LEDs   First we write 1s and then after a little delay time we write 0s. This is repeated until the button is released.

Now it is possible to compile the program, create the executable and run the program.
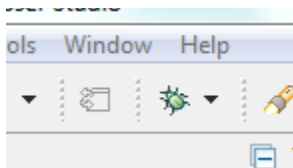

Figure 9. Debug Icon

Click on the debug icon shown in Figure 9. Save changes to main.c if prompted. The program will then compile the code, create and executable file and place it into the microcontroller.
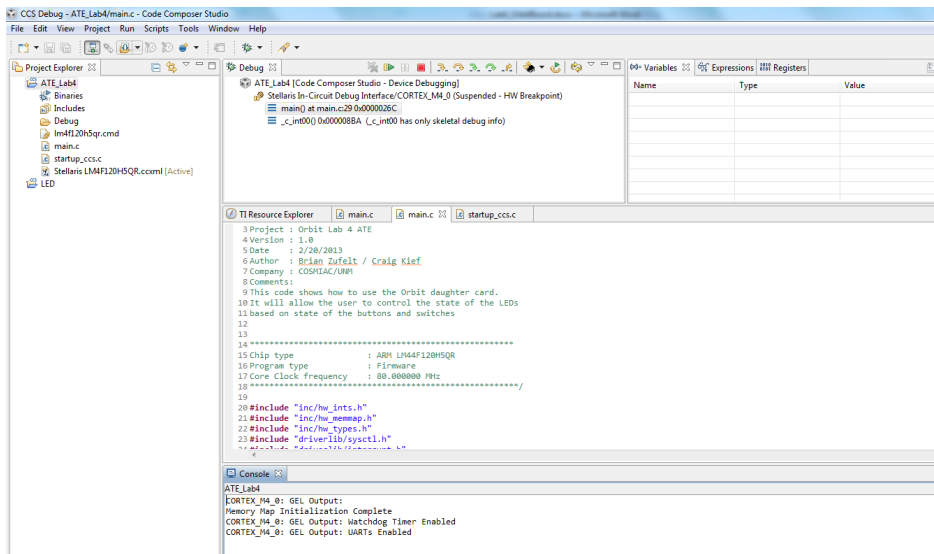
Figure 10. Program Compiled

What has now happened is that the executable file has been loaded into the microcontroller and has halted the program at void main (void) which is the start of your program. Using the mouse, rearrange the desktop to look like Figure 10. This will make it easier to see all the components as the program executes.
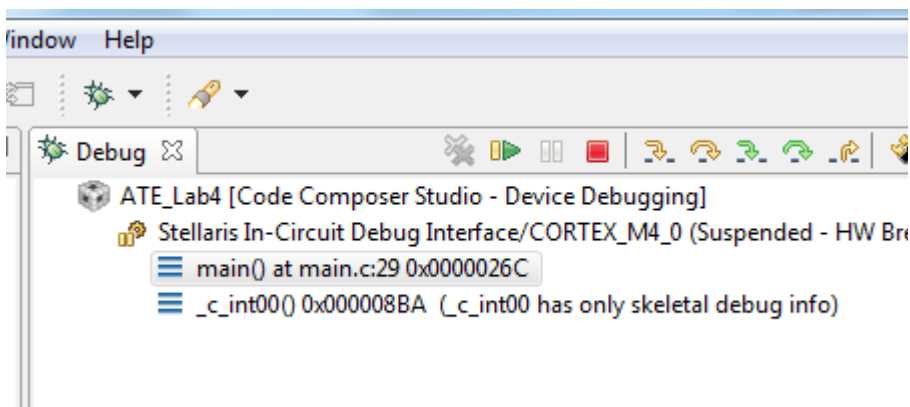
Figure 11. Run Executable

There are now several options. The designer can step line by line through the program by clicking on the yellow "step over" icon. When ready to continuously run the program, click on the resume icon (or hit F8) in the Debug pane to allow the program to continuously run.

Challenge: Remove the if/else statement, see results
Challenge: Change from button 1 to button 2

To accomplish this, you would need to refer to the LaunchPad Evaluation Board and the Orbit Schematics. There is a quick cheat sheet at the end of this tutorial that will provide ports and pins.

Attachment 1: Main.c file

```c
/*******************************************************

Project : Orbit Lab 4 ATE (Blink)
Version : 1.0
Date    : 2/20/2013
Author  : Brian Zufelt / Craig Kief
Company : COSMIAC/UNM
Comments:
This code shows how to use the Orbit daughter card.
It will allow the user to control the state of the LEDs
based on state of the buttons and switches


****************************************************
Chip type           : ARM TM4C123GH6PM
Program type        : Firmware
Core Clock frequency        : 80.000000 MHz
****************************************************/

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


void main(void) {

        // Setting the internal clock
        SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_XTAL_16MHZ|SYSCTL_OSC_MAIN);

        // Enable Peripheral ports for input/ output
        SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOC);  //PORTC
        GPIOPinTypeGPIOOutput(GPIO_PORTC_BASE, GPIO_PIN_6|GPIO_PIN_7); // LED 1 LED 2

        SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOB);  //PORTB
        GPIOPinTypeGPIOOutput(GPIO_PORTB_BASE, GPIO_PIN_5); // LED 4

        SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOD);  //PORT D
        GPIOPinTypeGPIOOutput(GPIO_PORTD_BASE, GPIO_PIN_6); // LED 3
        GPIOPinTypeGPIOInput(GPIO_PORTD_BASE, GPIO_PIN_2); // BTN 1

        SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);  //PORT F
        GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3); // RGB LED on Launchpad

        SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOA);  //PORT A
        GPIOPinTypeGPIOInput(GPIO_PORTA_BASE, GPIO_PIN_6); // Switch 2

        while(1)
        {
                if(GPIOPinRead(GPIO_PORTA_BASE, GPIO_PIN_6)){    // Listen for the switch

                        GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, 4); // LED Blue on Launchpad

                        // Cycle through the LEDs on the Orbit board
                        GPIOPinWrite(GPIO_PORTC_BASE, GPIO_PIN_6|GPIO_PIN_7, 0x40); // LED 1 on LED 2 Off
                        GPIOPinWrite(GPIO_PORTD_BASE, GPIO_PIN_6, 0x00); // LED 3 off, Note different PORT
                        GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_5, 0x00); // LED 4 off

                        SysCtlDelay(2000000); // Delay, Replaces for LOOP
```

```
                    GPIOPinWrite(GPIO_PORTC_BASE, GPIO_PIN_6|GPIO_PIN_7, 0x80); // LED 1 OFF, LED 2 on

                    SysCtlDelay(2000000); // Delay, Replaces for LOOP

                    GPIOPinWrite(GPIO_PORTC_BASE, GPIO_PIN_6|GPIO_PIN_7, 0x00); //LED 1 off LED 2 off
                    GPIOPinWrite(GPIO_PORTD_BASE, GPIO_PIN_6, 0x40); // LED 3 on

                    SysCtlDelay(2000000); // Delay, Replaces for LOOP

                    GPIOPinWrite(GPIO_PORTD_BASE, GPIO_PIN_6, 0x00); // LED 3 off
                    GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_5, 0x20); // LED 4 on

                    SysCtlDelay(2000000); // Delay, Replaces for LOOP


            }

            else {

                    GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, 8); //  LED Green on
Launchpad


                    // Cycle through the LEDs on the Orbit board
                    GPIOPinWrite(GPIO_PORTC_BASE, GPIO_PIN_6|GPIO_PIN_7, 0x00); // LED 1 off LED 2 Off
                    GPIOPinWrite(GPIO_PORTD_BASE, GPIO_PIN_6, 0x00); // LED 3 off, Note different PORT
                    GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_5, 0x20); // LED 4 on

                    SysCtlDelay(2000000); // Delay, Replaces for LOOP

                    GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_5, 0x00); // LED 4 off
                    GPIOPinWrite(GPIO_PORTD_BASE, GPIO_PIN_6, 0x40); // LED 3 on

                    SysCtlDelay(2000000); // Delay, Replaces for LOOP

                    GPIOPinWrite(GPIO_PORTC_BASE, GPIO_PIN_6|GPIO_PIN_7, 0x80); // LED 1 OFF, LED 2 on
                    GPIOPinWrite(GPIO_PORTD_BASE, GPIO_PIN_6, 0x00); // LED 3 off

                    SysCtlDelay(2000000); // Delay, Replaces for LOOP

                    GPIOPinWrite(GPIO_PORTC_BASE, GPIO_PIN_6|GPIO_PIN_7, 0x40); // LED 1 on LED 2 Off

                    SysCtlDelay(2000000); // Delay, Replaces for LOOP

            }

            while(GPIOPinRead(GPIO_PORTD_BASE, GPIO_PIN_2)){     // Listen for BTN 1

                    // All LED Blink
                    GPIOPinWrite(GPIO_PORTC_BASE, GPIO_PIN_6|GPIO_PIN_7, 0xFF);
                    GPIOPinWrite(GPIO_PORTD_BASE, GPIO_PIN_6, 0xFF);
                    GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_5, 0x20);
                    GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, 0xFF); // White Output

                    SysCtlDelay(2000000);

                    GPIOPinWrite(GPIO_PORTC_BASE, GPIO_PIN_6|GPIO_PIN_7, 0x00);
                    GPIOPinWrite(GPIO_PORTD_BASE, GPIO_PIN_6, 0x00);
                    GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_5, 0x00);
                    GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, 0x00); // White Output

                    SysCtlDelay(2000000);
            }

    }
```

9

}

Attachment 2: Block Diagram of the Pins Used in Projects