ESD Project Report

Generate PWM Signals with Tiva C Launchpad

A report submitted in part fulfilment ESD course **3nd Year (6th Semester) in ECE**

**Supervisor:** Dr. Deepak Nair



Designer: Apoorv Singh Negi (roll no:18UEC002)

Garv Tambi (roll no: 18UEC118)

**Department of Electronics and Communication Engineering The LNMIIT, Jaipur, Rajasthan, India**

April 18, 2021

**Declaration**

This report has been prepared on the basis of my own work and designs developed under Micro Lab Course. Where other published and unpublished source materials have been used, these have been acknowledged.

Student Name:

1. APOORV SINGH NEGI Roll No: 18UEC002

2) GARV TAMBI Roll No. 18UEC118

Date of Submission: 18/04/2021

Signature of Supervisor/ Supervisors: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Name of Supervisor/ Supervisors: Dr. Deepak Nair

[**CHAPTER 1: Introduction**](#_78vl9vx1wjsf)5

[1.1 Synopsis:](#_c1v9f18gcle9) 5

[**CHAPTER 2: Project Principle**](#_qfeozqwo2ybb)6

[**CHAPTER 3: Block Diagram and Schematic**](#_lms8d0qa60oz)7

[**Chapter 4: Components Required**](#_2gz6b9pmsxa3)9

[**CHAPTER 5: Code Explanation**](#_ed1dkx1bzbco)10

[5.1 Software Programming TM4C123 PWM Modules](#_cp8usws8hltf) 10

[5.1.1 Enabling Clock to PWM Modules](#_dreawykyth45) 10

[5.1.2 PWM Clock Configuration](#_fe91ian7h2pg) 10

[5.1.3 Enable TM4C123 PWM Generator](#_yxlckp5mln1j) 11

[5.1.4 Setting TM4C123 PWM Frequency](#_stothjm7xgl2) 12

[5.1.5 Setting PWM Duty Cycle](#_2jvvu2135f6r) 13

[5.1.5.1 Compare Registers](#_kmdia0w3dcvn) 13

[5.1.5.2 Generator register](#_igkx46wnmmua) 13

[5.2 software programming of GPIO Port and PWM pins](#_hurwkr8rxvp1) 14

[5.2.1 Enable clocks for PWM pin and GPIO Port](#_dzujw2s4si28) 14

[5.2.2 Configuration of PF2 pin of GPIO for M1PWM6 Channel output pin](#_c6xmtimn2wa2) 15

[5.2.2.1 alternate functionality](#_ht8y98darfhm) 15

[5.2.2.2 port control](#_xxqtzn2c538s) 15

[**CHAPTER 6: CODE**](#_v1jb2zagk350)17

[**CHAPTER 7: Observations and Results**](#_s2mqbv4rxw2g)19

[**CHAPTER 8: Future Goals and Applications**](#_cit9s080hkkc)21

[**Chapter 9: Bibliography**](#_whtcgsavwuwq)22

**Acknowledgment**

Firstly, I would like to thank Prof. Deepak Nair for his unconditional support and thorough guidance without which this project would not have seen the light of the day. He not only guided us through the project but also played a vital role in helping us relate the theoretical concepts with their practical concepts and instilled in us enthusiasm to LEARN BY DOING. This project was definitely a milestone in our career and I learned a lot more than just electronics, values of hard work.

# CHAPTER 1: Introduction

The main aim of this project is to implement Pulse-Width Modulation (PWM) on TIVA C launchpad. The code generates 50 Hz frequency and variable duty cycle PWM signal on PF2 pin which is connected to the onboard BLUE Led. So, the intensity of BLUE LED varies according to the varying duty cycle of PWM output signals.

This project uses basic fundamentals of PWM.

## 1.1 Synopsis

Pulse width modulation (PWM) is a powerful technique for digitally encoding analog signal levels. High-resolution counters are used to generate a square wave, and the duty cycle of the square. Wave is modulated to encode an analog signal. Typical applications include switching power supplied and motor control. The TM4C123GH6PM microcontroller contains two PWM modules, each with four PWM generator blocks and a control block, for a total of 16 PWM outputs. The control block determines the polarity of the PWM signals, and which signals are passed through to the pins.

# 

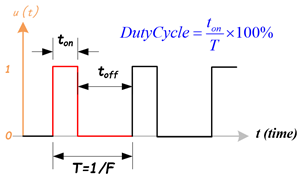
# 

# 

# 

# CHAPTER 2: Project Principle

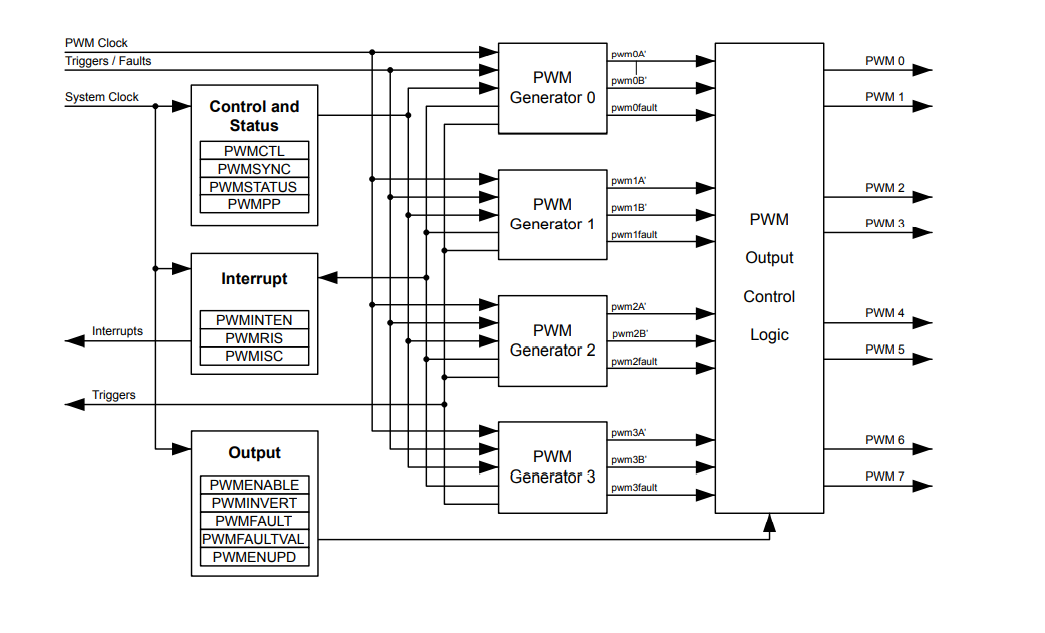
A PWM signal consists of two main components that define its behaviour: a duty cycle and a frequency.



* Duty Cycle  
  The duty cycle describes the amount of time the signal is in a high (**ON**) state as a percentage of the total time it takes to complete one cycle. A lower duty cycle corresponds to lower power, because the power is **OFF** for most of the time. Duty cycle is expressed in percent, 100% duty cycle would be fully **ON** as same as setting the signal to Vcc (high); 0% duty cycle would be the same as grounding the signal.
* Frequency  
  The frequency determines how fast the PWM completes a cycle, ie. 100Hz would be 100 cycles per second. In another words, it shows how fast the PWM switches between high and low states. In the digital system, PWM is the method to produce variable voltage using digital means. Typically, digital system only has two output voltages, the high (5V, 3.3V … etc.) or low (0V).

Hence, by varying the value of duty cycle we could interface any peripheral that we want.

# CHAPTER 3: Block Diagram and Schematic

Fig 3.1: PWM Module Diagram

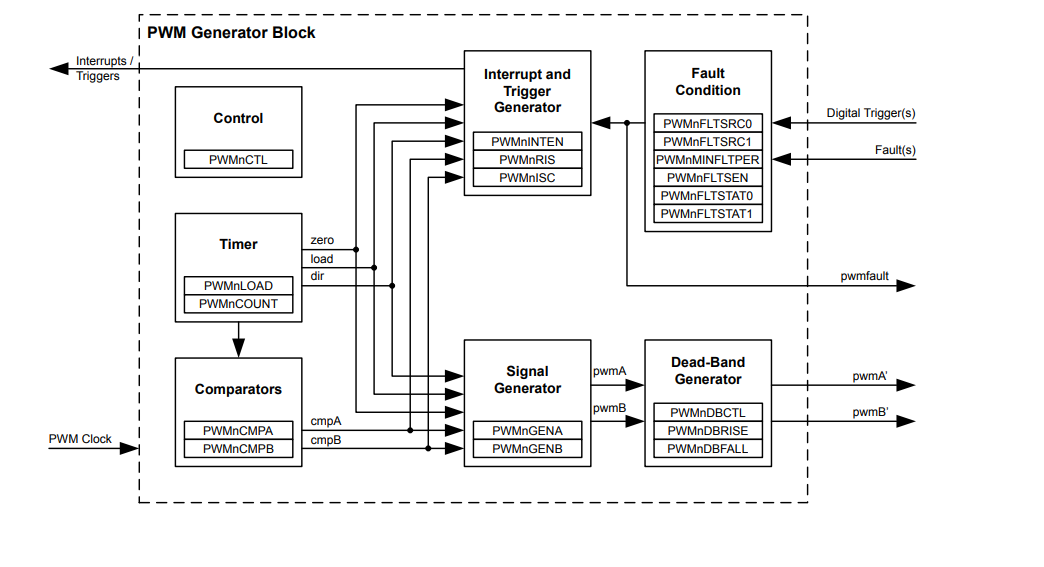


Fig 3.2: PWM Generator Block Diagram

# 

# Chapter 4: Components Required

1. TIVA C Launchpad Board

A microcontroller is an integrated circuit (IC) device used for controlling other portions of an electronic system, usually via a microprocessor unit (MPU), memory, and some peripherals.

We use microcontrollers when we need more controlling in the program rather than processing. For processing we would use a microprocessor.

It is a low-cost evaluation platform for Arm cortex-M4F based microcontrollers. It consists of 80-MHz Arm Cortex-M4F CPU, 256kB of flash memory, and 32kB of SRAM. Providing integrated USB 2.0 support for USB and two 12-bit ADC modules.

1. LED

# CHAPTER 5: Code Explanation and working

## 5.1 Software Programming TM4C123 PWM Modules

We have integrated a well-functioning code in TIVA - C microcontroller. The code was written in IAR IDE for Arm using Embedded C language. TM4C123GH6PM header file was used.

By default, or on reset, system clock is disabled to all peripherals of TM4C123 microcontroller and so is to PWM modules. The clock is disabled to save power. Therefore, first we enable the clock source to PWM modules.

### 5.1.1 Enabling Clock to PWM Modules

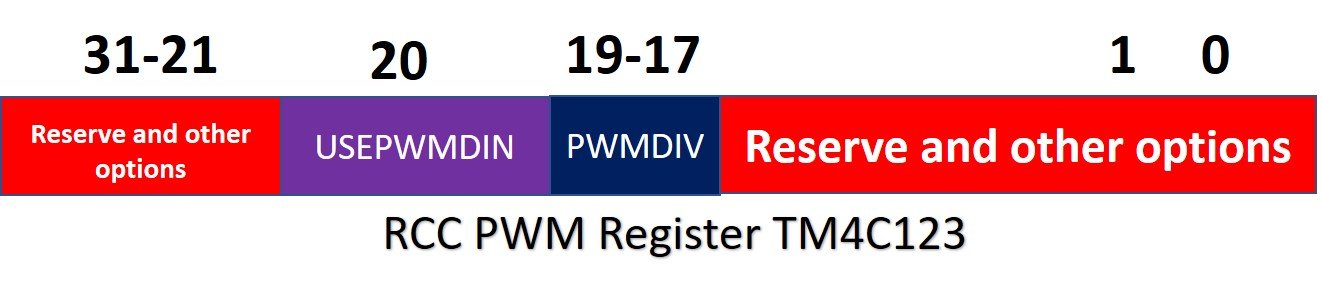
RCGCPWM register is used to enable clock source to both PWM modules. Setting Bit0 and Bit1 of RCGCPWM register enables the clock and clearing these bits disable the clock to both modules.



SYSCTL->RCGCPWM |= (1<<1); /\* To enable clock source to PWM1 module \*/

### 5.1.2 PWM Clock Configuration

RCGCPWM register selects operating frequency of TM4C123 microcontroller directly as a clock for PWM modules. Run-Mode clock configuration (RCC) register is used to scale down frequency for PWM modules. We lower down the clock frequency using the RCC PWM register.

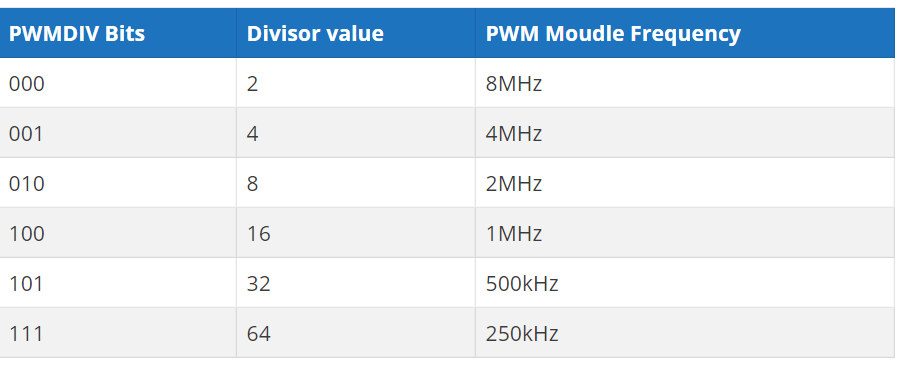


RCC register acts as a divider between system clock frequency and input frequency to PWM modules.

SYSCTL->RCC |= (1<<20); /\* Enable System Clock Divisor function \*/

SYSCTL->RCC |= 0x000E0000; /\* Using pre-divider value of 64 and after that feeding that clock to PWM1 module to make PWM module frequency as 250 KHz \*/

We can choose any value of PWM module frequency according to our need from frequency divider table:



### 5.1.3 Enable TM4C123 PWM Generator

Each PWM block contains four generators which are also known as counters. They are of 16-bit width.



We use PWMnCTL registers to enable each generator for PWM modules.

* Bit0 of the control register is used to enable and disable the generator. Setting this bit enables and clearing disables the generator.
* Bit1 is a mode select bit of counter. Each generator has a 32-bit counter registers. It can be configured either in up-count (Mode bit = 1) or down-count mode (Mode bit = 0).

We use this PWM1->\_3\_CTL as we are using PWM module 1 and in that module, we are using PWM generator 3.

Here we are using down count mode and disabling the generator 3 counter.

In down count mode, the counter starts counting from load value and keeps decrementing. When the counter reaches zero, it resets to load value again.

We first disable generator 3 counter and after configuring all the PWM registers then we again enable this register.

PWM1->\_3\_CTL &= ~(1<<0); /\* Disabling the Generator 3 counter \*/

PWM1->\_3\_CTL &= ~(1<<1); /\* select down count mode for counter 3\*/

### 5.1.4 Setting TM4C123 PWM Frequency

The load value must be loaded to the PWMnLOAD register. Each generator has its own load register. This load register defines the frequency of PWM signals. For our case:

Clock frequency = 16MHz

Divider frequency value used = 64

Final clock frequency supplied =

16Mhz/64 = 250KHz

PWM Module Clock Period = 1 / 250KHz = 0.004ms

PWM Signal Period = 1 / 50Hz = 20ms

Load register value = 20ms/0.004ms = 5,000

PWM1->\_3\_LOAD = 5000; /\* set load value for 50Hz frequency \*/

### 5.1.5 Setting PWM Duty Cycle

In TM4C123 microcontroller we use two registers to set duty cycle of PWM signal. These registers are compared and generator output enables registers. Each generator block provides two PWM output signals such as pwmA and pwmB. Also, for each PWM output, there is one compare register such as PWMxCMPA and PWMxCMPB. These compare registers are used to set the duty cycle PWM signals.

#### 5.1.5.1 Compare Registers

The value of the compare register (PWMxCMPA and PWMxCMPB) is compared with the value of the counter. As soon as the value counter matches with the value of counter, we can configure the PWM pin to do one of the following things:

1. Toggle
2. Go active high
3. Make transition to active low

As for our code the duty cycle that we want to set is variable so we can’t directly put the value of duty cycle in our PWMxCMPx register. So, initially we set the value of duty cycle to be minimum which can be calculated using the formula:

PWMxCMPx = (100% - required duty cycle in %) of PWMxLOAD for down counter

As, initially we will set the minimum duty cycle in order to get least intensity value for LED:

Required duty cycle in % = 0%

PWMxLOAD = 5000

PWMxCMPx = (100% - 0%) \* 5000

= 5000

But we take the value to be 4999.

PWM1->\_3\_CMPA = 4999; /\* setting duty cycle to minimum value\*/

#### 5.1.5.2 Generator register

We set the value of generator register as:

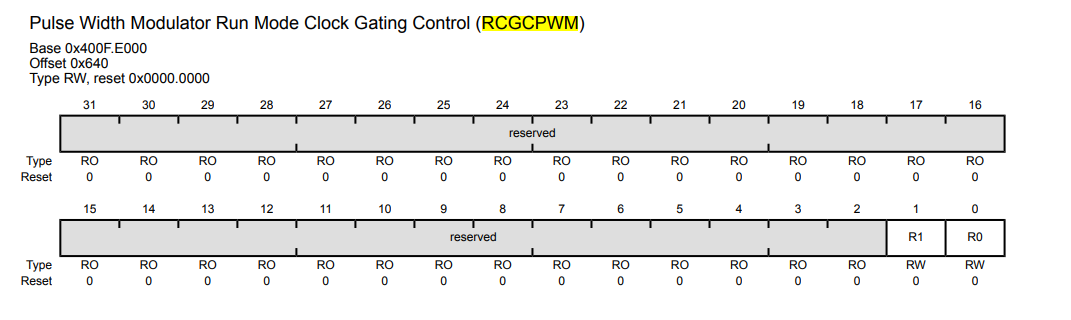
PWM1->\_3\_GENA = 0x0000008C; /\* Set PWM output when counter reloaded and clear when matches PWMCMPA \*/

We set the output when counter is reloaded and value is cleared when it matches the value present in compare register.

## 5.2 software programming of GPIO Port and PWM pins

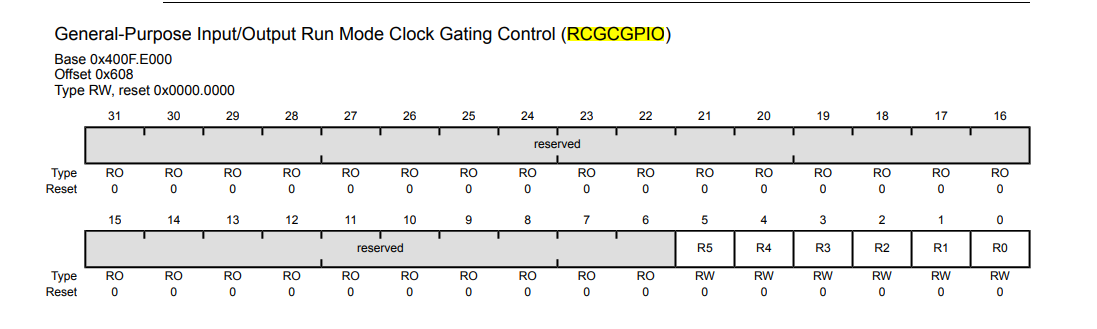
## 5.2.1 Enable clocks for PWM pin and GPIO Port

Enable clock for PWM1 module using Pulse Width Modulator Run Mode Clock Gating Control (RCGCPWM) Register using R1 or bit 1 of the register:



SYSCTL->RCGCPWM |= 2; /\* Enable clock to PWM1 module \*/

Enable clock for GPIO PORT F using: General-Purpose Input/Output Run Mode Clock Gating Control (RCGCGPIO) Register using R5 or bit 5 of the register:

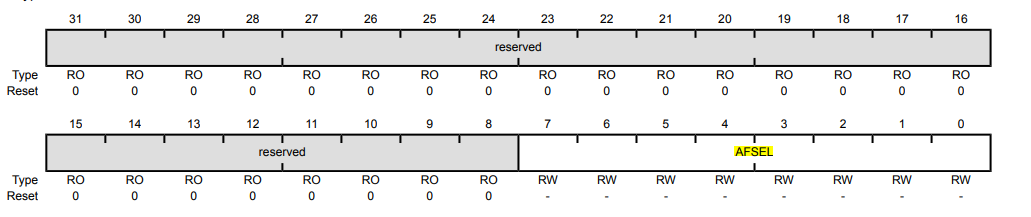


SYSCTL->RCGCGPIO |= 0x20; /\* Enable system clock to PORTF \*

## 5.2.2 Configuration of PF2 pin of GPIO for M1PWM6 Channel output pin

### 5.2.2.1 alternate functionality

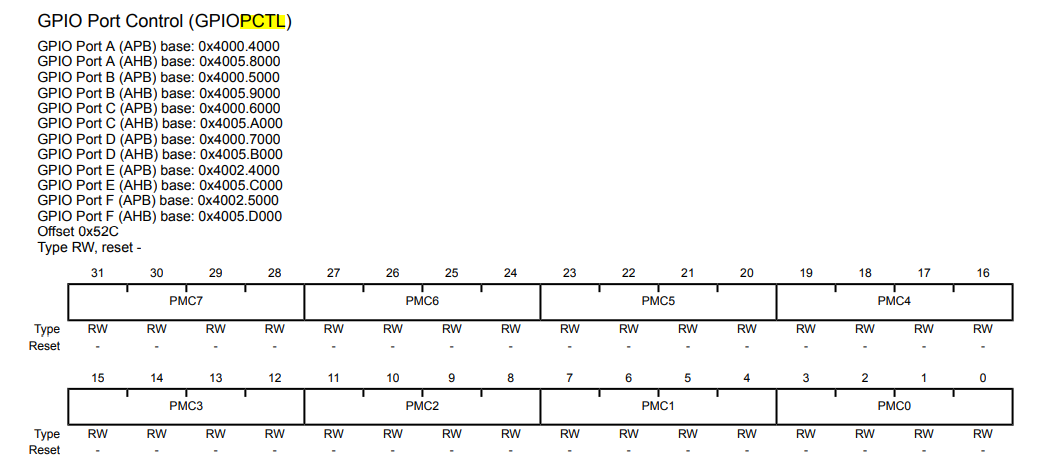
As PWM is considered to be alternate functionality of GPIO PIN so we have to configure this pin as its alternate function for pwm module using GPIO Alternate Function Select (GPIOAFSEL) register of PORTF:



GPIOF->AFSEL |= (1<<2); /\* PF2 pin sets as alternate function \*/

### 5.2.2.2 port control

We are using GPIOPCTL register in conjunction with the GPIOAFSEL register. This register selects the specific peripheral signal for each GPIO pin when using the alternate function mode. We are setting PF2 as an output PWM pin as Blue Led is connected to it in our hardware.





Here, we can see alternate functionality of PF2 as M1PWM6 pin.

GPIOF->PCTL &= ~0x00000F00; /\*set PF2 as output pin \*/  
GPIOF->PCTL |= 0x00000500; /\* make PF2 PWM output pin \*/

Digitally enabling the pin to glow LED as by default all the LEDs are in reset mode:

GPIOF->DEN |= (1<<2); /\* set PF2 as a digital pin as we want to glow LED \*/

**While loop:**

In the while loop, we are decreasing the value of duty cycle in each iteration and when the value of duty cycle becomes negative, we again reload the value of duty cycle with 5000 and this process keeps on repeating for infinite time. Between reloading the value of duty cycle again to CMPA register we provide a delay so that we could notice the beginning of the LED fading process again in our board.

Delay is provided using the header file.

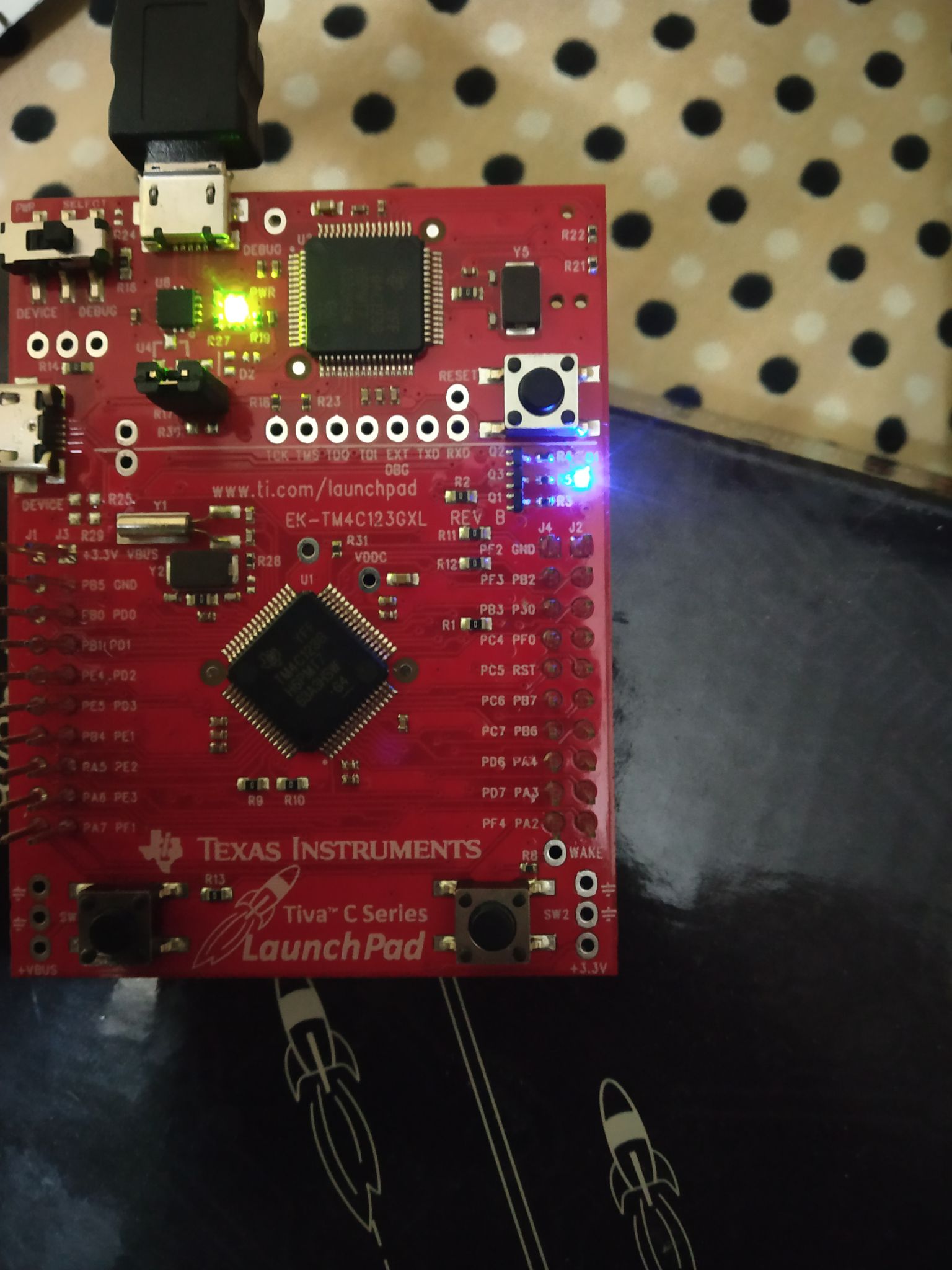
# CHAPTER 6: CODE

|  |
| --- |
| **Main code:**  /\* Generating 50HZ and variable duty cycle on PF2 pin\*/ /\* PWM1 module and PWM generator 3 of PWM1 module is used\*/ #include "TM4C123GH6PM.h" #include "delay.h" int main(void) {  int duty\_cycle = 4999;   /\* Clock setting for PWM and GPIO PORT \*/ SYSCTL->RCGCPWM |= 2; /\* To enable Clock in PWM1 Module\*/ SYSCTL->RCGCGPIO |= 0x20; /\* Enable system clock to PORTF as all LEDs are connected to PORTF\*/ SYSCTL->RCC |= (1<<20); /\* Enable System Clock Divisor function for PWM modules \*/ SYSCTL->RCC |= 0x000E0000; /\* Use pre-divider value of 64 and after that feed clock to PWM1 module\*/   /\* Setting of PF2 pin for M1PWM6 channel output pin \*/ GPIOF->AFSEL |= (1<<2); /\* PF2 sets an alternate function as blue LED is connected to PF2 \*/ GPIOF->PCTL &= ~0x00000F00; /\*set PF2 as output pin \*/ GPIOF->PCTL |= 0x00000500; /\* make PF2 PWM output pin \*/ GPIOF->DEN |= (1<<2); /\* set PF2 as a digital pin as we want to glow LED \*/   PWM1->\_3\_CTL &= ~(1<<0); /\* Disable Generator 3 counter \*/ PWM1->\_3\_CTL &= ~(1<<1); /\* select down count mode of counter 3\*/ PWM1->\_3\_GENA = 0x0000008C; /\* Set PWM output when counter reloaded and clear when matches PWMCMPA \*/ PWM1->\_3\_LOAD = 5000; /\* set load value for 50Hz 16MHz/65 = 250kHz and (250KHz/5000) \*/ PWM1->\_3\_CMPA = 4999; /\* set duty cyle to to miniumum value\*/ PWM1->\_3\_CTL = 1; /\* Enable Generator 3 counter \*/ PWM1->ENABLE = 0x40; /\* Enable PWM1 channel 6 output \*/    while(1)  {  duty\_cycle = duty\_cycle - 10;  if (duty\_cycle <= 0)   duty\_cycle = 5000;  PWM1->\_3\_CMPA = duty\_cycle;  delay(100000);  } }  Header file Delay.c #include "delay.h" void delay( int iteration) { for(unsigned j =0; j<iteration;j++); } Delay.h #ifndef \_\_DELAY\_H\_\_ #define \_\_DELAY\_H\_\_  void delay(int iteration); #endif |

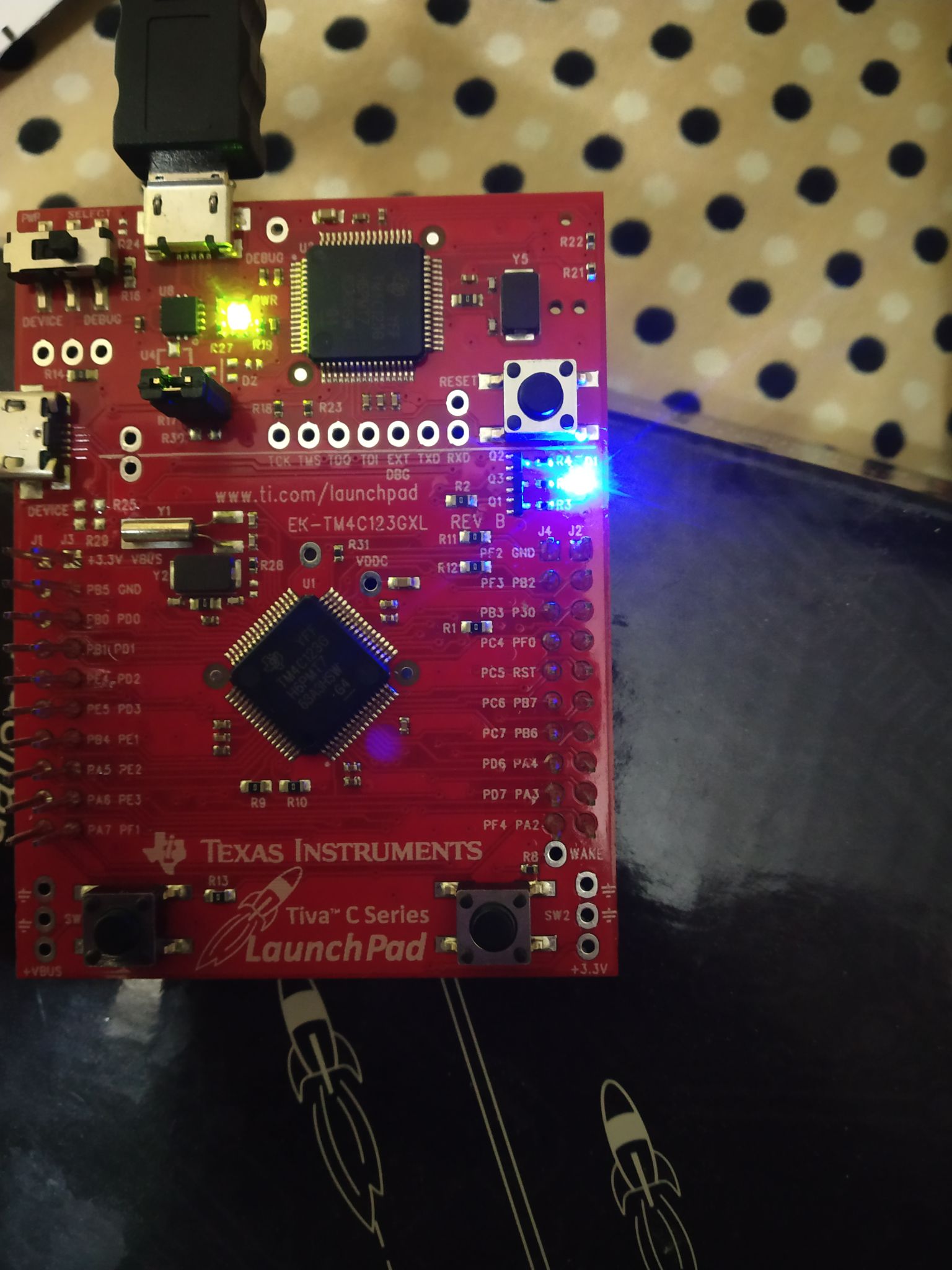
# CHAPTER 7: Observations and Results

When program is uploaded in the board, we could see that:

Initially the intensity of light from the LED is low or we can say LED is off as the value of duty cycle is minimum:



As the value of the duty cycle of PWM signal increases the intensity of blue light also increases:



# Similarly, as the value of duty cycle changes, we could see changes in the intensity of the blue LED.

# CHAPTER 8: Future Goals and Applications

Here we have implemented a basic project of PWM using LED interfacing but PWM in real life has a number of applications including music generation using PWM, fan controller or motor interfacing.

We could add a motor at the output of the PF2 pin to interface with the motor at different operating speeds.

We could also add an oscilloscope at PF2 pin to see the PWM signal generated by the TIVA-C board of different duty cycles.

We could also add a FAN at PF2 pin to control the speed of the fan at variable speeds. This can be used in electrical vehicles or in automation in those applications where we didn’t need that thing to run continuously and constantly.

This concept helps us to give speed control and power control to a specific device.

# 

# Chapter 9: Bibliography

❖ <https://www.youtube.com/channel/UCEvrlcJ63LONZiXh4yVNB5Q>

❖ David E. Simon, An Embedded Software Primer, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, 1999

❖ Jack G. Ganssle, The Art of Programming Embedded Systems,

Academic Press, San Diego, 1992 Publication.