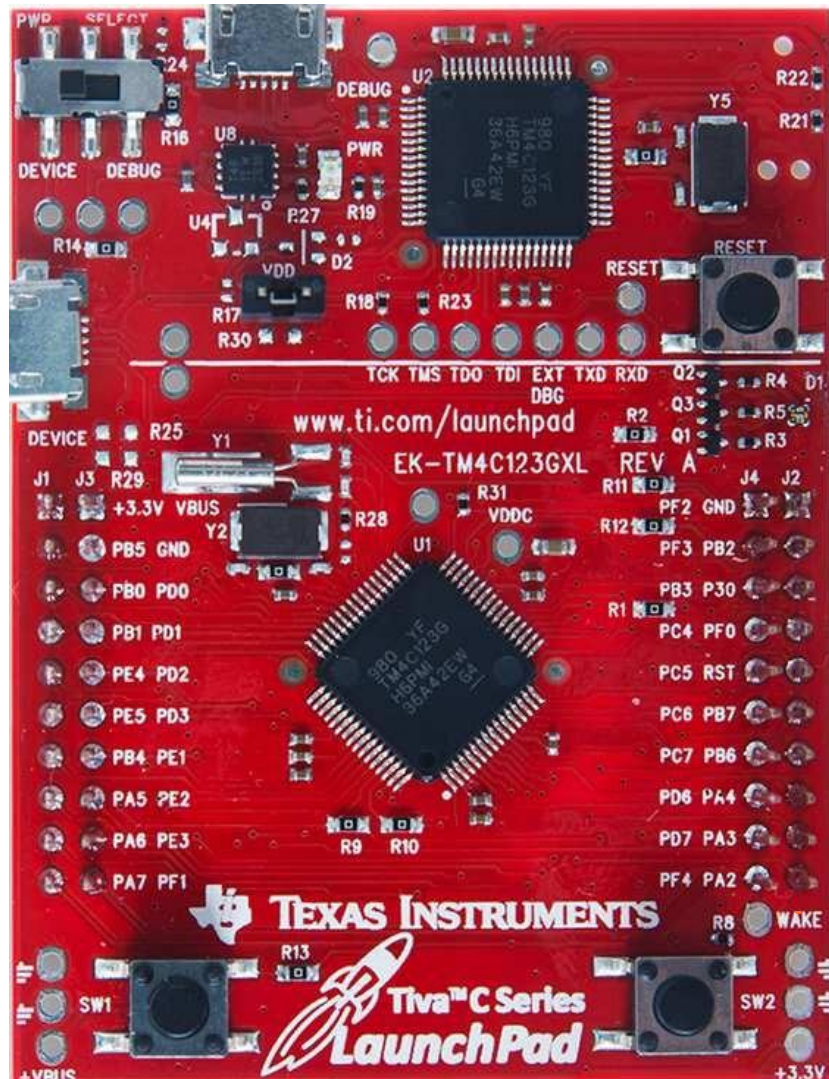



Embedded System Design using TM4C LaunchPad™ Development Kit



Copyright

Copyright © 2007-2014 Texas Instruments Incorporated. Tiva and TivaWare are trademarks of Texas Instruments Incorporated. ARM and Thumb are registered trademarks and Cortex is a trademark of ARM Limited. All other trademarks are the property of others.

 Please be aware that an important notice concerning availability, standard warranty, and use in critical applications of Texas Instruments semiconductor products and disclaimers thereto appears at the end of this data sheet.

Texas Instruments Incorporated
108 Wild Basin, Suite 350
Austin, TX 78746

<http://www.ti.com/tm4c>

<http://www-k.ext.ti.com/sc/technical-support/product-information-centers.htm>



Preface	1
Getting Started	2
Overview.....	3
Tiva C Series LaunchPad.....	7
Code Composer Studio	10
TivaWare Library for Tiva Platform.....	18
Using Tera Term.....	19
Experiment 1 To Blink an Onboard LED	21
1.1 Objective.....	22
1.2 Introduction.....	22
1.2.1 GPIO Module.....	22
1.3 Component Requirements	24
1.3.1 Software Requirements.....	24
1.3.2 Hardware Requirements	24
1.4 Software	24
1.4.1 Flowchart.....	24
1.4.2 C Program Code to Blink an Onboard LED	25
1.5 Procedure.....	26
1.6 Observation	27
1.7 Summary	27
1.8 Exercise.....	27

List of Figures

1	Tiva C Series TM4C12x Microcontroller Architecture	4
2	Pin Diagram of TM4C123GH6PM	5
3	Functional Block Diagram of ARM Cortex-M4F Processor	6
4	EK-TM4C123GXL	7
5	Power Select Switch Position	9
6	CCS Functional Overview	10
7	CCS Edit Perspective	11
8	CCS Debug Perspective	12
9	Workspace and Projects (as viewed in CCS)	12
10	Workspace and Projects	13
11	New CCS Project Creation	14
12	Project Properties Window	15
13	File Search Path	16
14	Debug Environment	17
15	Tera Term New Connection Window	19
16	Tera Term Serial Port Setup Window	20
17	Tera Term Terminal Setup Window	20
1-1	Functional Block Diagram	22
1-2	LaunchPad Schematics for GPIO Connected to LEDs	23
1-3	Flowchart for Blinking an Onboard LED	25

List of Tables

1	API Functions Used in the Application Program	18
1-1	GPIO Pin Functions and USB Device Connected	23
1-2	Components Required for the Experiment.....	24
1-3	API Functions Used in the Application Program	26

The Texas Instruments Tiva C Series microcontrollers (MCUs) are low-power, versatile and smart devices that feature different sets of peripherals and are used for various applications. The Tiva C series microcontrollers feature a powerful 32-bit ARM core, 80-MHz operation, and 100 DMIPS performance. The TM4C123x MCUs of the Tiva C Series provide a balance between the floating-point performance required to create highly responsive mixed-signal applications and the low-power architecture required to enable increasingly aggressive power budgets.

In this lab manual, all the experiments are performed using TM4C123GH6PM microcontrollers to realize widespread applications. The experiments take advantage of the wide range of parts and tools supported by Texas Instruments for design with the Tiva C Series microcontrollers and wireless communication products. The Tiva EK-TM4C123GXL LaunchPad kit is an easily available evaluation kit for users to get started with designing applications using Tiva C Series microcontroller. The TM4C123x MCUs are supported by TivaWare™ for C Series software that helps the user to start writing production-ready code easily and quickly and minimize the overall cost of software ownership.

Each experiment in this manual includes:

- An introduction to the features of the microcontroller used
- A design overview
- The software flowchart and the C program for the application
- The procedure to run the application and note the observations
- An exercise to further enhance the learning in the experiment

Experiments 1 to 7 use typical applications like GPIO, PWM, ADC, Interrupt, Low Power Modes, UART. Experiments 8 and 9 use different interfaces supported by the TM4C123GH6PM such as I2C and USB bulk transfer. Experiment 10 demonstrates the use of mathematical library for exhaustive computation. Experiment 11, 12 and 13 use applications for IoT (Internet of Things).

On completion of all the experiments in this manual, the user will be able to build a variety of production-ready applications with the Tiva C Series microcontrollers.

Getting Started



Topics	Page
Overview.....	3
Tiva C Series LaunchPad	7
Code Composer Studio	10
TivaWare Library for Tiva Platform.....	18
Using Tera Term	19

Overview

Introduction to Tiva C Series

The Texas Instruments Tiva C Series microcontrollers (MCUs) are low-power, versatile and smart devices. The Tiva C Series consists of several devices that feature different sets of peripherals and are targeted for various applications.

The Tiva C Series MCUs provide both high performance for mixed-signal applications and low power architecture for power critical applications. The Tiva C Series MCUs are supported by TivaWare™ for C Series for software development. The [Tiva EK-TM4C123GXL](#) Launchpad kit is an easily available evaluation kit for users to get started with designing applications using Tiva C Series microcontroller.

TM4C12x Microcontrollers

The Tiva C series microcontrollers feature a powerful 32-bit ARM core, 80-MHz operation, and 100 DMIPS¹ performance. The TM4C12x devices provide high performance and advanced integration. The TM4C123GH6PM microcontroller is used on the EK-TM4C123GXL.

The features of TM4C123GH6PM microcontroller are:

- It supports Thumb2 Technology, which means it supports 16/32-bit code. It consumes 26% less memory & is 25% faster when compared to pure 32-bit microcontrollers. This means, it provides the optimum mix of 16-bit and 32-bit microcontroller application requirements.
- It consists of a battery-backed hibernation module that provides logic to switch power off to the main processor and its peripherals while the processor is idle. The features of this module include low-battery detection, signaling and interrupt generation, with optional wake-up on low battery.
- The power consumption is as low as 370µA/MHz. It takes 500µs to wake up from low-power modes.
- The current consumption of Real Time Clock is 1.7µA and it also supports internal and external power control.
- It has six physical GPIO (General Purpose Input Output) blocks, six 16/32-bit and six 32/64-bit general purpose timers, 32 dedicated 32-bit single-precision registers, also addressable as 16 double-word registers.
- TM4C123GH6PM devices support flexible clocking system.
- It has an internal precision oscillator, external main oscillator with PLL(Phase Locked Loop) support, internal low frequency oscillator and real-time-clock through Hibernation module.
- It has saturated math for signal processing.
- It supports atomic bit manipulation and Read-Modify-Write using bit-banding.
- It has single cycle multiply and hardware divider module.
- The device has 256KB of Flash Memory, 32KB single-cycle SRAM and 2KB EEPROM. The memory is used efficiently because of unaligned data access.
- It has two motion control modules each with eight high-resolution PWM outputs.
- The device supports IEEE754 compliant single-precision floating-point unit.
- The debugger access that the device supports are JTAG and Serial Wire Debug - ETM (Embedded Trace Macrocell) available through Keil and IAR emulators.

1. DMIPS - Dhrystone Million Instructions per Second. It is a measure of processor performance.

- The serial connectivity includes peripherals such as USB 2.0 (OTG/Host/Device), 8 - UART (with IrDA, 9-bit and ISO7816 support), 6 - I²C, 4 - SPI (Serial Port Interface), Microwire or TI synchronous serial interfaces and 2 - CAN.

The basic blocks in the Tiva C Series TM4C12x microcontrollers are shown in [Figure 1](#).

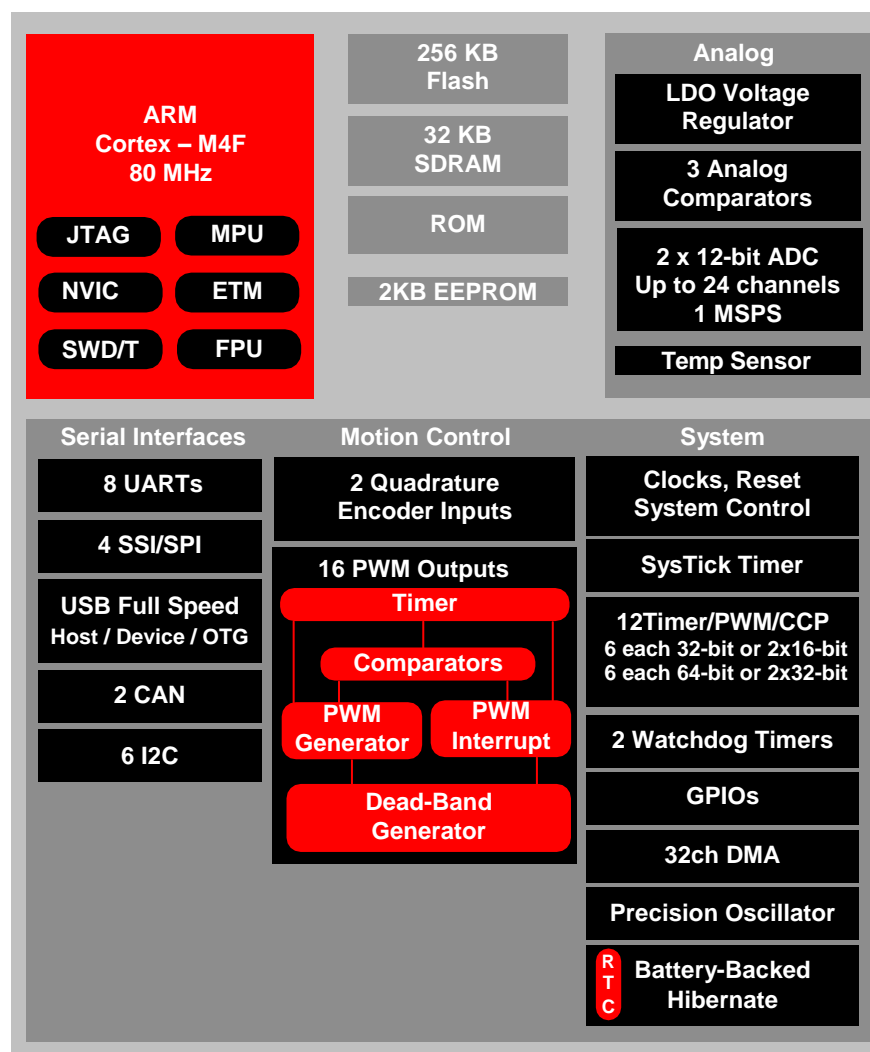


Figure 1 Tiva C Series TM4C12x Microcontroller Architecture

Figure 2 shows the pin diagram for TM4C123GH6PM microcontroller.

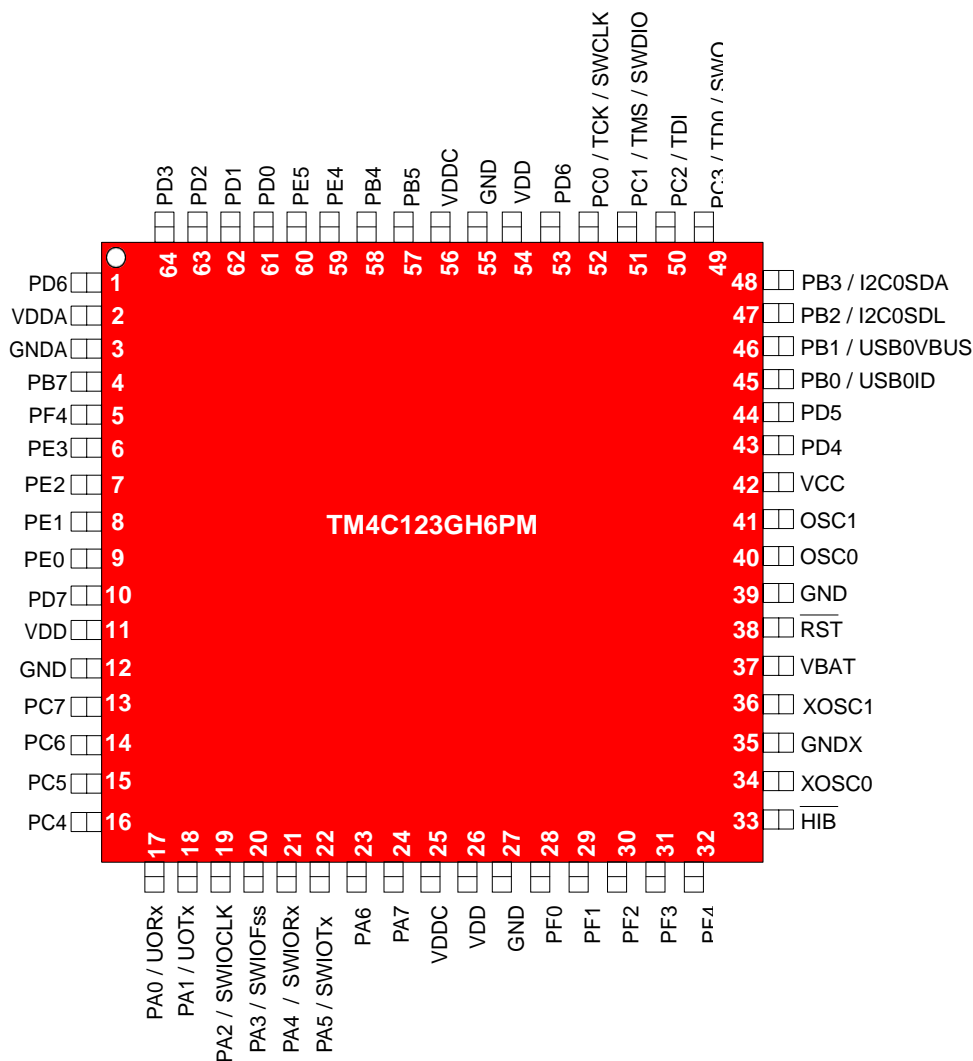


Figure 2 Pin Diagram of TM4C123GH6PM

ARM Cortex-M4F Processor

The ARM Cortex-M4F processor used in the Tiva C Series microcontrollers provides high performance while meeting low-power requirements. Figure 3 shows the functional block diagram of the ARM Cortex-M4F processor.

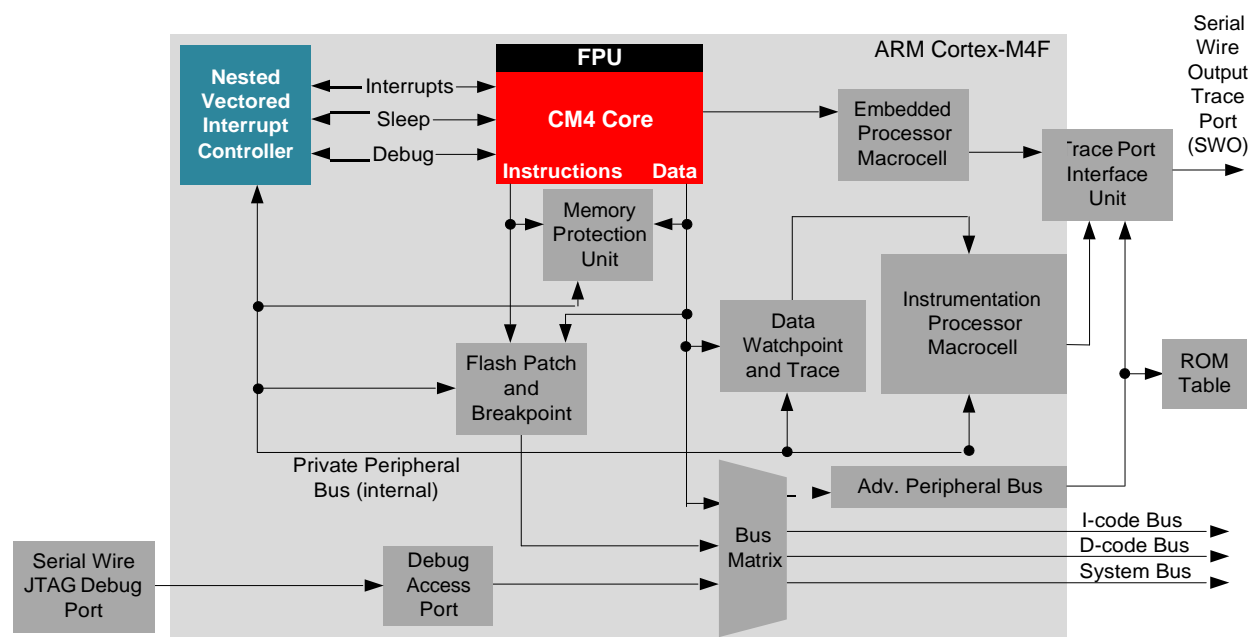


Figure 3 Functional Block Diagram of ARM Cortex-M4F Processor

The Cortex-M4F processor is a 3-stage pipeline Harvard architecture which is most suitable for embedded applications. The processor has a Memory Protection Unit (MPU) for memory control that can separate code, data and stack based on the security levels. The Cortex -M4F processor core interacts with the Nested Interrupt Controller (NVIC) that provides exceptional interrupt performance with eight interrupt priority levels and a Non-Maskable Interrupt (NMI). The interrupt latency is reduced significantly because of tight integration of the processor core with the NVIC.

Enhanced system debug is achieved in the Cortex -M4F processor with several supporting modules. The processor implements a JTAG (Joint Test Actions Group) or a 2-pin Serial Wire Debug (SWD) for hardware debug. For system trace, the processor incorporates an Instrumentation Trace Macrocell (ITM) with Data Watchpoint and Trace unit. The Embedded Trace Macrocell (ETM) enables full instruction trace.

The Flash Patch and Breakpoint Unit (FPB) module provides up to eight hardware breakpoint comparators that debuggers can use. The Trace Port Interface Unit (TPIU) connects the Cortex-M4F trace data to an off-chip Trace Port Analyzer.

Tiva C Series LaunchPad

Introduction

The Tiva C Series TM4C123GXL LaunchPad Evaluation Kit (EK-TM4C123GXL) is a low-cost evaluation platform for ARM Cortex-M4F-based microcontrollers from Texas Instruments. It is an easy to use startup kit that provides everything the user will need to evaluate the Tiva C Series microcontroller. The design of the EK-TM4C123GXL highlights the [TM4C123GH6PM](#) microcontroller with a USB 2.0 device interface and hibernation module. The EK-TM4C123GXL also features programmable user buttons and an RGB LED for custom applications. The EK-TM4C123GXL includes:

- A TM4C123G LaunchPad Evaluation board
- On-board In-Circuit Debug Interface (ICDI)
- USB Micro-B plug to USB-A plug cable
- [ReadMe First](#) quick-start guide

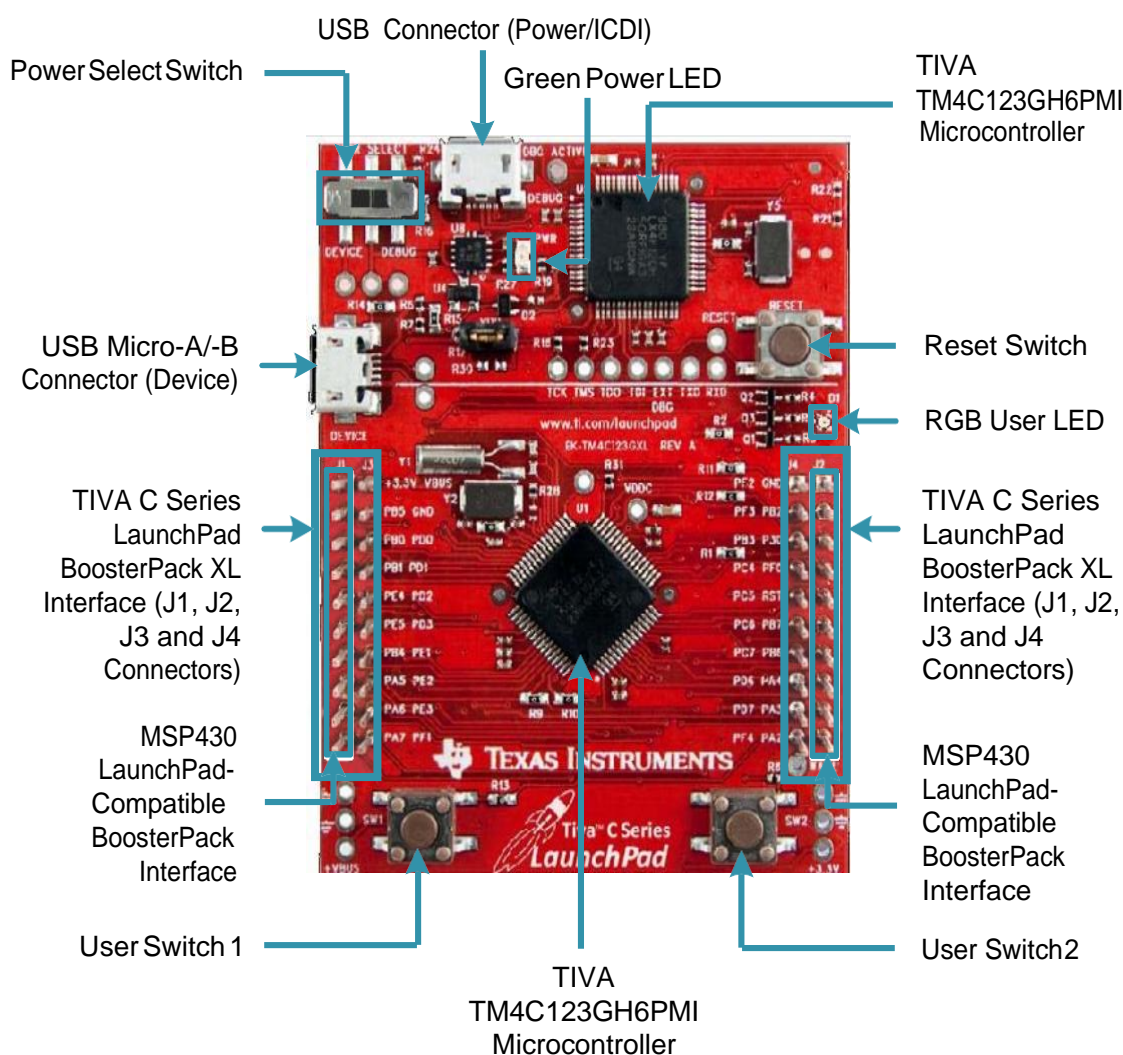


Figure 4 EK-TM4C123GXL

The features of the EK-TM4C123GXL is:

- ARM® Cortex™-M4F, 64-pin 80MHz TM4C123GH6PM processor
- On-board USB ICDI (In-Circuit Debug Interface)
- Micro AB USB port
- Device/ICDI power switch
- BoosterPack XL pinout also supports legacy BoosterPack pinout
- 2 user pushbuttons (SW2 is connected to the WAKE pin)
- Reset button
- 3 user LEDs (1 tri-color device)
- Current measurement test points
- 16MHz Main Oscillator crystal
- 32kHz Real Time Clock crystal
- 3.3V regulator
- Support for multiple IDEs

Hardware Setup

The LaunchPad experimenter board includes a pre-programmed TM4C123H6PM device which comes preinstalled in the target socket. Connect the LaunchPad Board ICDI USB connector (Refer [Figure 4](#)) to a free USB port on your PC using the USB cable.

When the LaunchPad is connected to your PC via USB, the driver installation starts automatically. If prompted for software, allow Windows to install the software automatically.

After driver installation, the LaunchPad board starts a demo application with an LED toggle sequence. The on-board emulator generates the supply voltage and all signals necessary to start the demo.

The Application Demo Program

The Tiva C Series LaunchPad comes pre-programmed with the RGB QuickStart application. This application demonstrates control of the on-board RGB LED, TM4C123GH6PM microcontroller's Hibernate functionality and serial communication with the Tiva C Series LaunchPad. Once the LaunchPad is connected to the PC via the ICDI USB port, the drivers are installed automatically and the demo application starts execution. The steps to be followed for the demo application are:

1. Check if the Power Select Switch in the upper left corner of the board is in the DEBUG position as shown in [Figure 5](#).

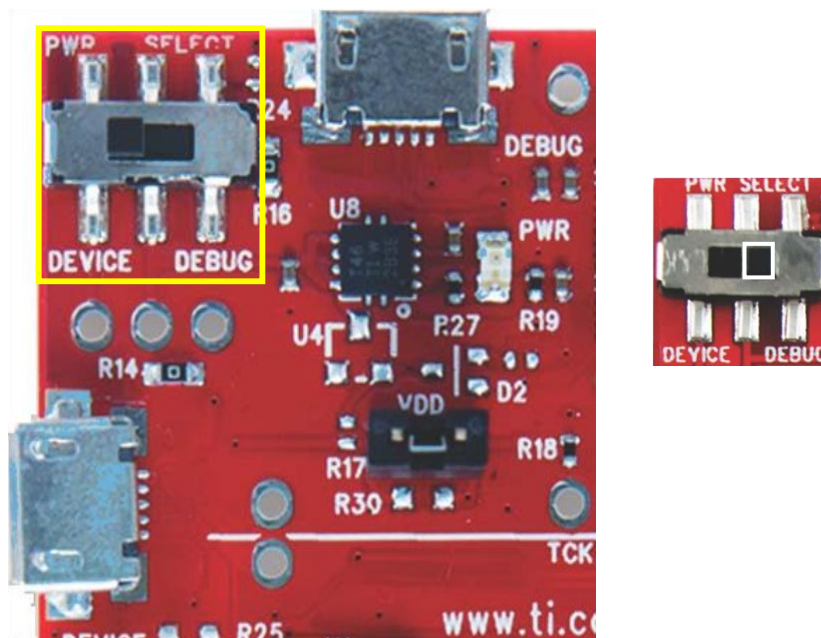


Figure 5 Power Select Switch Position

2. Press the bottom left button (SW1) or the bottom right button (SW2) to scan the ROYGBIV color spectrum of the RGB LED.
3. Leave the LaunchPad idle for five seconds. You will see a random color display of the RGBLED.
4. To enter Hibernation mode, press and hold SW1 and SW2 for three seconds. While in this mode, you can see the LED blink every three seconds.
5. To exit Hibernation mode, press SW2.
6. To control these functions serially using the UART, see the readme file in the qs_rgb project in the Tivaware™ examples for the EK-TM4C123GXL.

Documents for Reference

- [LaunchPadUser Guide](#)
- [ROM User's Guide](#)
- [TM4C123GH6PM Data Sheet](#)
- [Peripheral Driver Library User Guide](#)

Code Composer Studio

Overview

Code Composer Studio (CCS) is the integrated development environment for the whole span of TI Micro-controllers, DSPs and application processors. Code Composer Studio includes a suite of tools used to develop and debug embedded applications. For all device families, CCS includes compilers, source code editor, project build environment, debugger, profiler, simulators and many other features.

CCS Functional Overview

The CCS supports all the phases of the development cycle: design, code and build, debug. The CCS includes all the development tools - compilers, assembler, linker, debugger, BIOS and one target - the Simulator as shown in **Figure 6**.

- The **C compiler** converts C source code into assembly language source code.
- The **assembler** translates assembly language source files into machine language object files.
- The **standard run-time libraries** contain ANSI standard run-time-support functions, compiler-utility functions, floating-point arithmetic functions and I/O functions that are supported by the C compiler.
- The **linker** combines object files, library files and system or BIOS configuration files into a single executable object module. The .out file is the executable program for the target device.
- The **debugger** supports debug activities such as watching variables, graphing signals on the target, viewing the memory and call stack, etc. The .gel files initialize the debugger with address of memory, peripherals and other hardware setup details.
- The program code can be debugged on the **simulator** or **emulator** or the **target device** based on the target config file (.ccxml) that specifies the connection to the target.

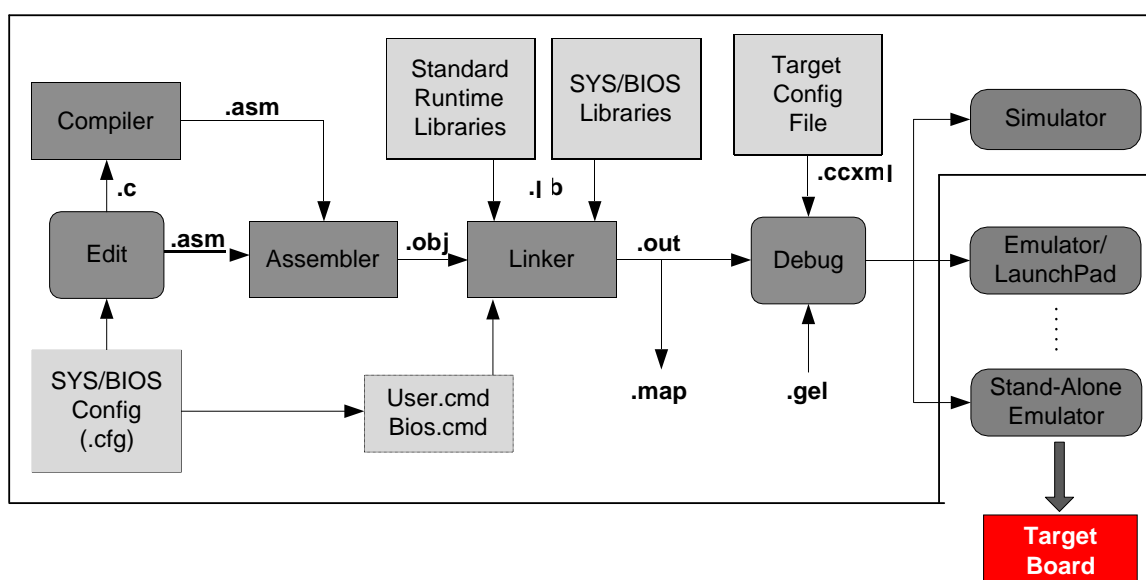


Figure 6 CCS Functional Overview

CCS Perspectives

The Code Composer GUI has two perspectives:

- **The Edit Perspective**
- **The Debug Perspective**

The **Edit perspective** has menus, buttons and editor window to edit the source code as shown in [Figure 7](#).

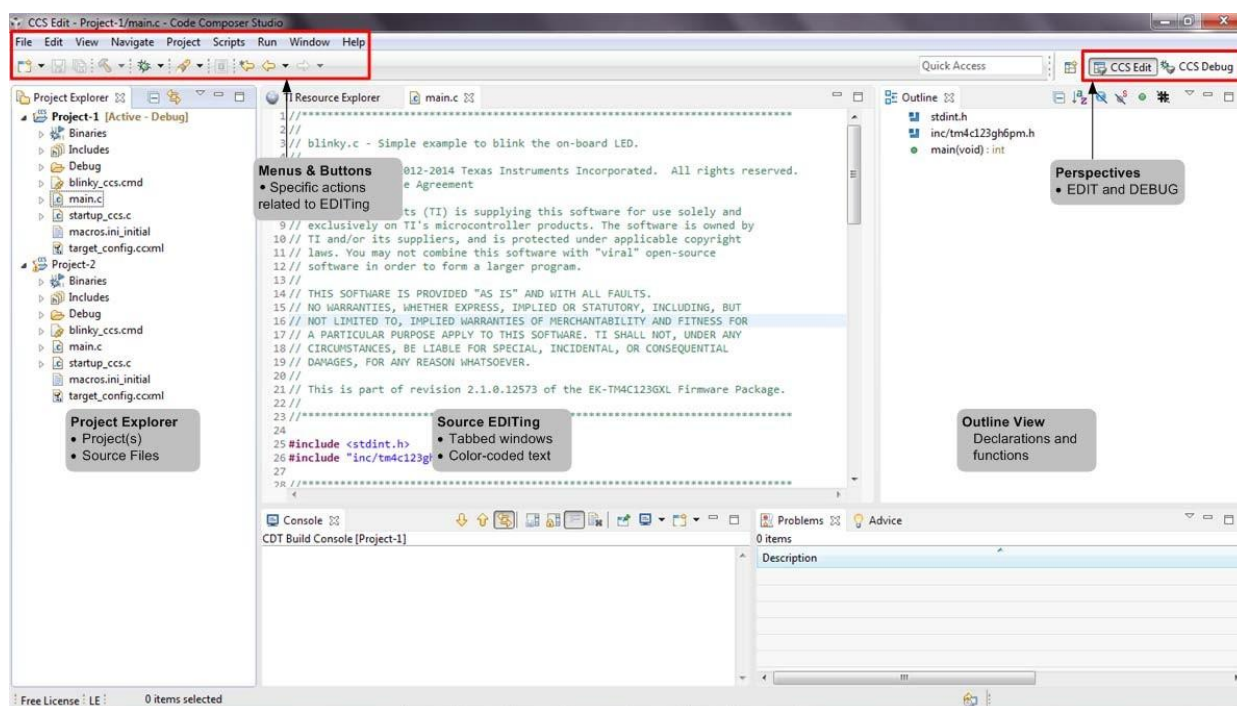


Figure 7 CCS Edit Perspective

The **Debug perspective** has menus, buttons and debug window related to debugging as shown in **Figure 8**.

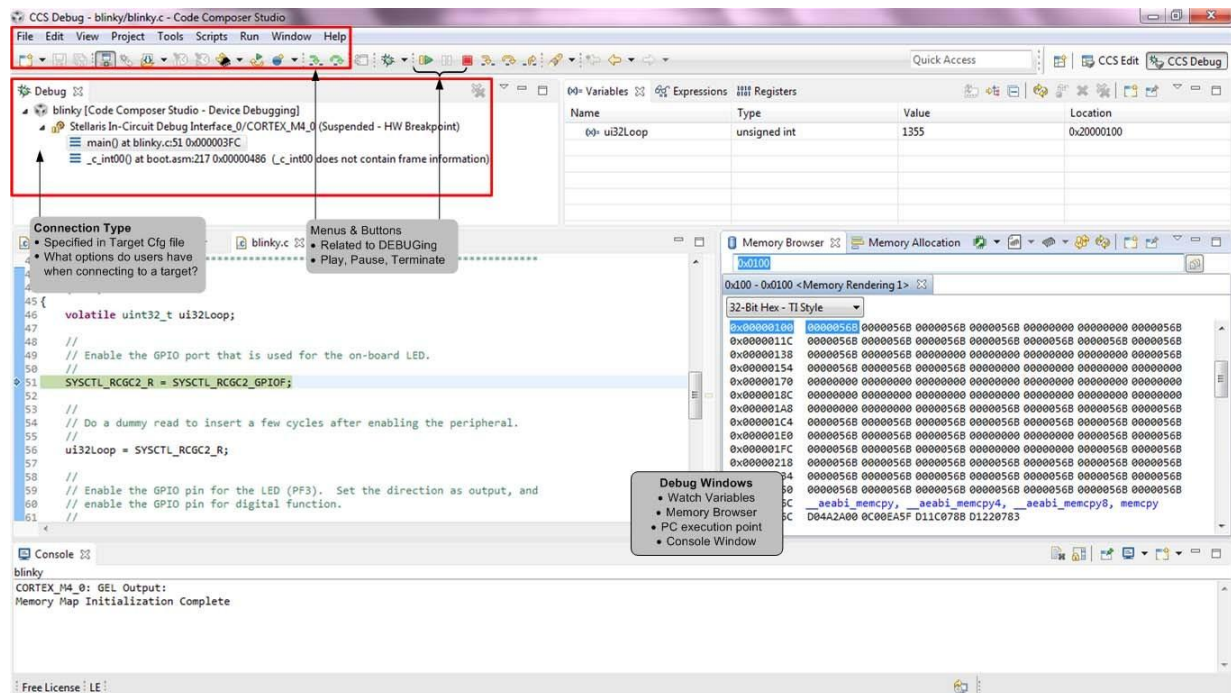


Figure 8 CCS Debug Perspective

Workspaces and Projects

A workspace contains settings and preferences, as well as links to the projects as shown in **Figure 9**.

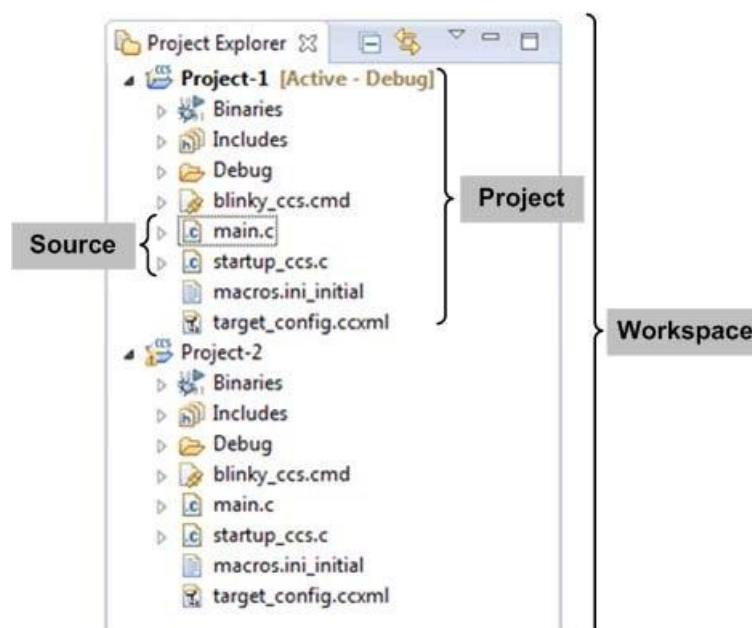


Figure 9 Workspace and Projects (as viewed in CCS)

A project contains build and tool settings, as well as links to the input files - source files, header files and library files as shown in [Figure 10](#).

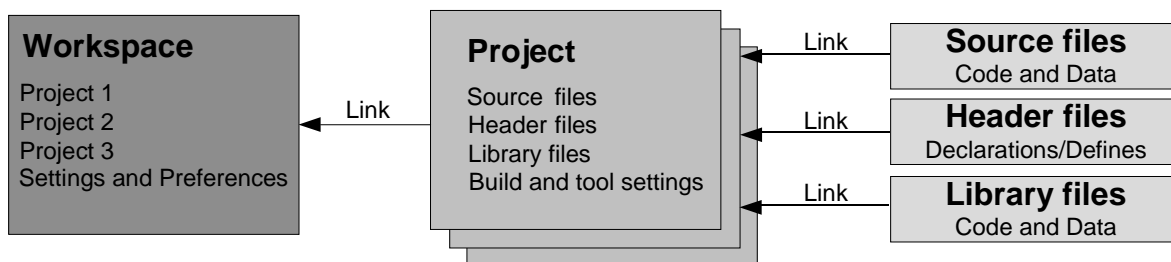


Figure 10 Workspace and Projects

Hence, deleting projects from the workspace deletes the links, not the projects. Similarly, deleting files from the project deletes the links, not the files.

Project Creation and Build

A project contains all the files you will need to develop an executable output file (.out) which can be run on the TM4C123GH6PM hardware. The steps to be followed to create a project are:

1. Create a New Project from **Project** ➔ **New Project**. Refer [Figure 11](#).
 - a. Select the Target as Tiva TM4C123GH6PM from the drop down box.
 - b. Choose the Connection as Stellaris In-Circuit Debug Interface from the drop down box.
 - c. Enter a relevant Project name in the text box provided.
 - d. Click on Finish to create the new project.

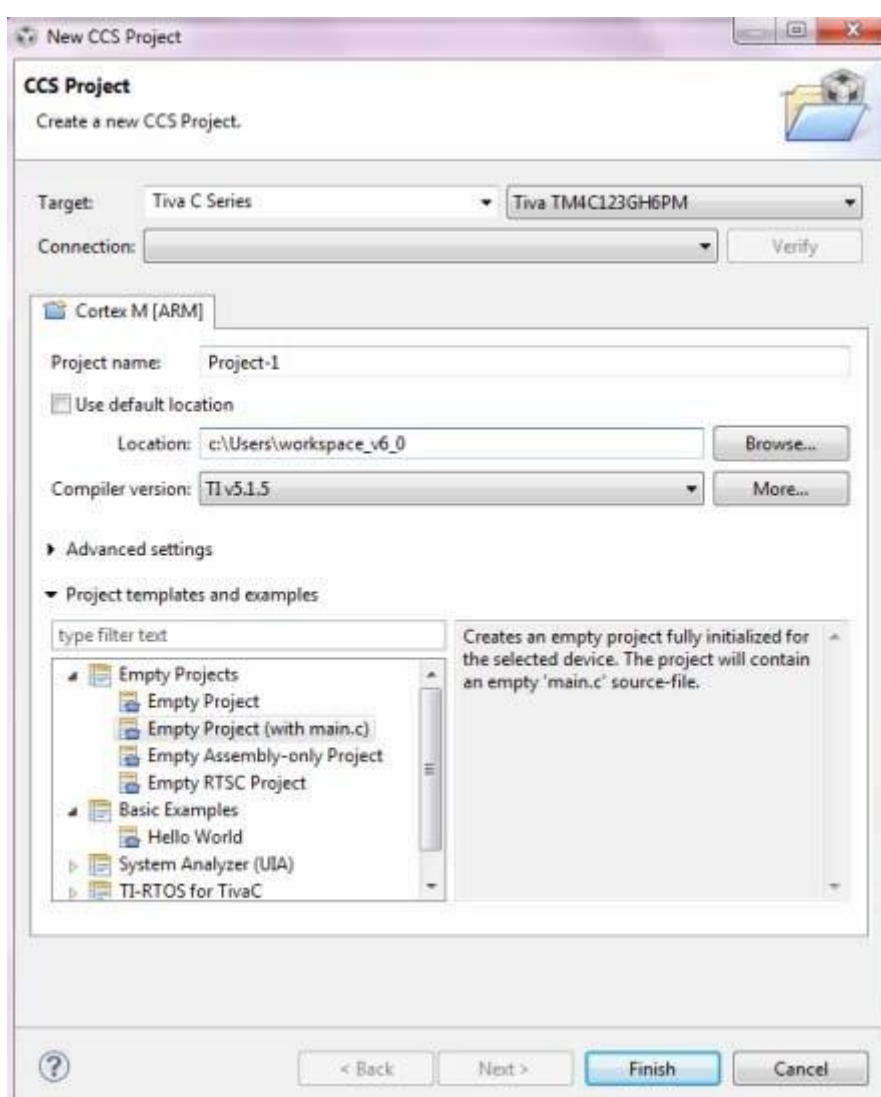


Figure 11 New CCS Project Creation

2. Go to Properties from **Project** ➔ **Properties**.
 - a. Click on **Include Options** under **ARM Compiler** as shown in [Figure 12](#).
 - b. Add the root of "TivaWare_C_Series-2.1.0.12573" to the #include search path.

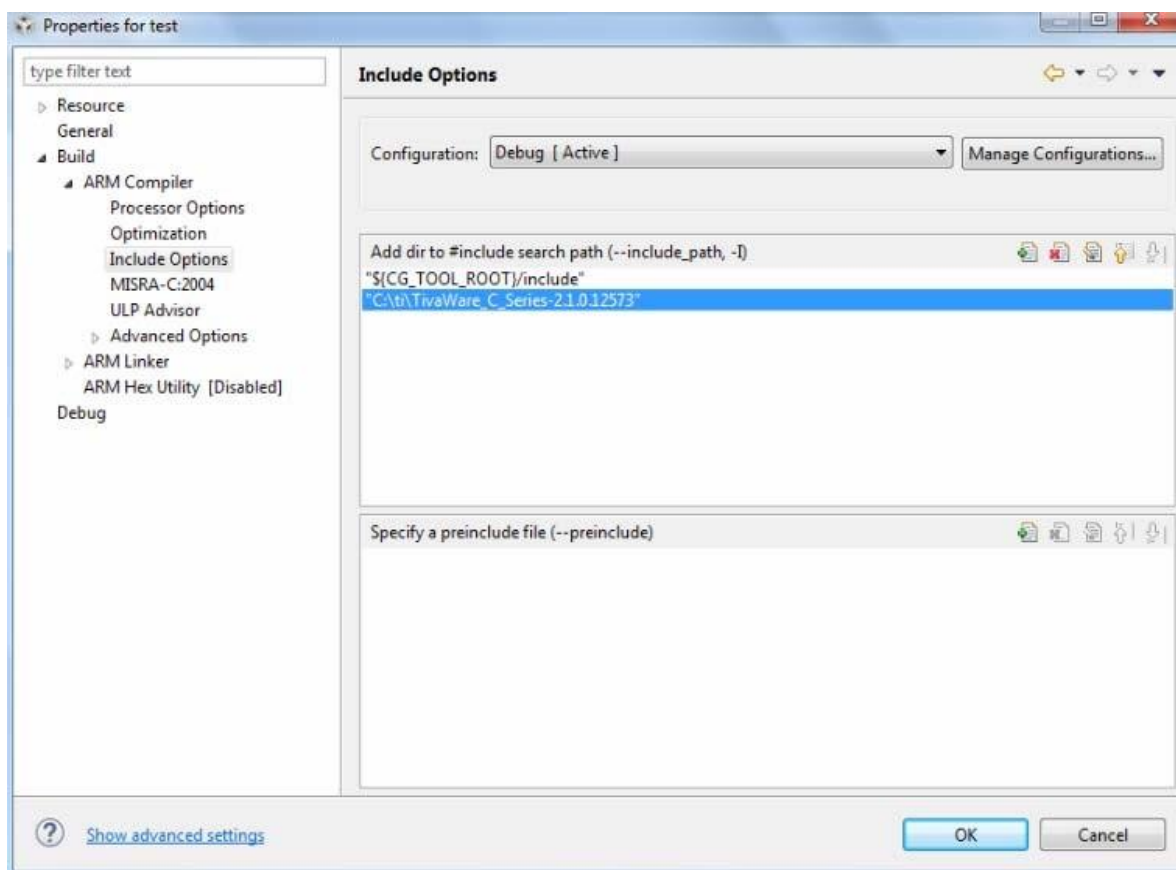


Figure 12 Project Properties Window

- c. Click on **File Search Path** under **ARM Linker** as shown in [Figure 13](#).
 - d. Add "**driverlib.lib**" from driverlib\ccs\Debug inside the TivaWare_C_Series installation directory to the "Include Library" section.

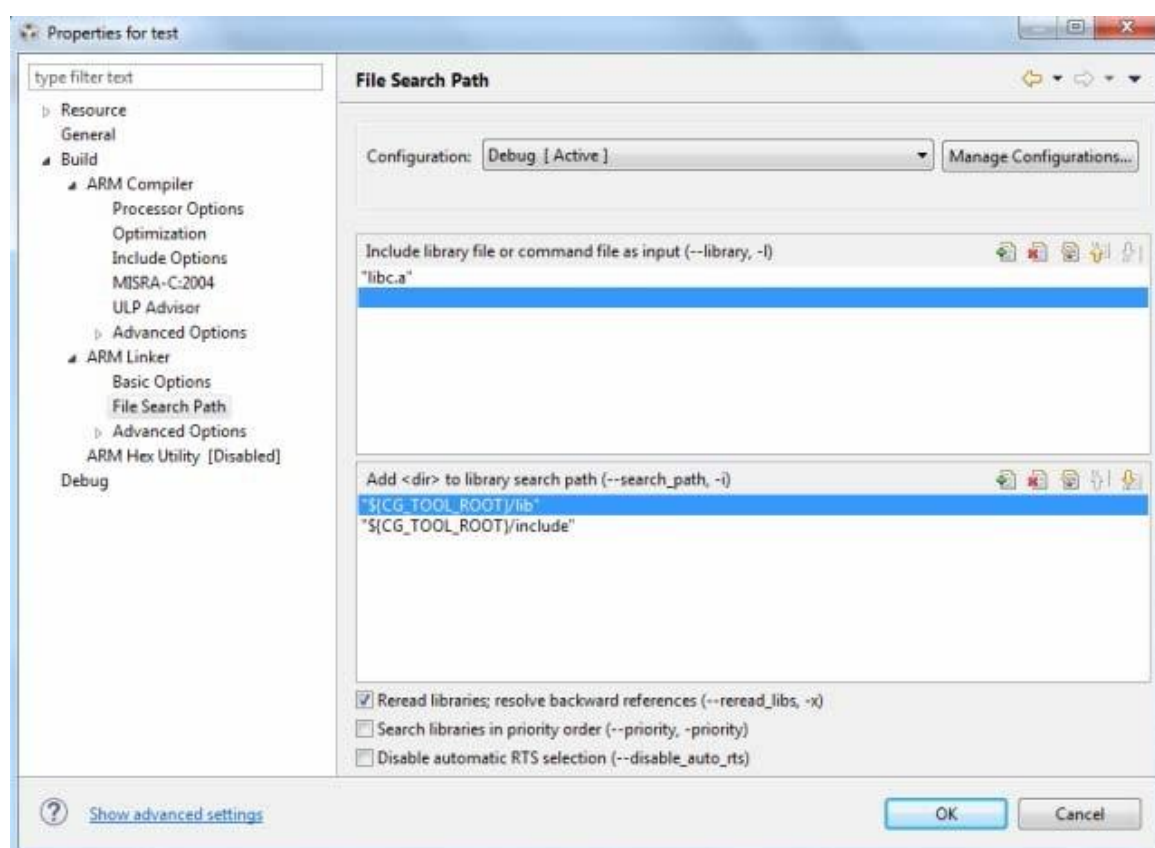



Figure 13 File Search Path

3. Code Composer will add the named project to your workspace and display it in the Project Explorer pane. Based on the template selection, it will also add a file called main.c / main.asm and open it for editing. Type in your program code in the main.c / main.asm file and save it.

Debug Environment

After completion of target download, CCS enters the Debug perspective. Notice the Debug tab in the upper right-hand corner indicating that we are now in the "CCS Debug" view (See [Figure 8](#)). Click and drag the perspective tabs to the left until you can completely see both the tabs.

After modification of source files, CCS can build the program, open the debug perspective view, connect and download it to the target (flash device), and then run the program to the beginning of the main function.

To do this, click on the "Debug" button . When the Ultra-Low-Power Advisor (ULP Advisor) appears, click the **Proceed** button. The program has completed execution through the C-environment initialization routine in the runtime support library and stopped at main() in main.c.


The basic buttons that control the debug environment are located at the top of the Debug pane. If the pane is closed accidentally, the Debug controls will disappear. To bring back the debug controls, click **View**  **Debug** on the menu bar.





Figure 14 Debug Environment

You can explore each button by clicking on it to see its function. At this point, your code should be at the beginning of `main()`. Look for a small blue arrow on the left of the opening brace of `main()` in the middle window as shown in [Figure 14](#). The blue arrow indicates where the Program Counter (PC) is pointing to.

Click **Resume**  button to run the code.

Click **Suspend**  to stop code execution in the middle of the program.

To single-step into the code, click **Step Into**  to help in debugging the program and check if each line of code is producing the desired result.

The **Terminate**  button will terminate the active debug session, close the debugger and return to the "CCS Edit" perspective. It also sends a reset to the LaunchPad board.

Important Notes for CCS

1. Make sure your Launch pad USB DEBUG port is connected to your PC.
2. Check if the project selected for the experiment is active.
3. Build the project by clicking the Debug button on the menu bar.
4. If the Launch pad has gone to Hibernate mode, it can be awakened by pressing SW2.
5. For experiments which are taken from the library, any warning message indicating that the project was created with an earlier compiler version can be safely ignored.
6. When you import the project, it will be automatically copied into your workspace, preserving the original files.
7. All the modifications done are in the local workspace and do not get reflected onto the TivaWare folder. So you can safely modify, save and always retrieve the demo code from TivaWare.
8. If you delete the project accidentally or intentionally in CCS, only the project in the workspace is deleted, while the imported project stays intact in your workspace.
9. In the dialog box, it is also possible to delete files from the disk which will erase the files from the library but that is a general OS file delete operation and should not be attempted.

TivaWare Library for Tiva Platform

The Texas Instruments [TivaWare](#) peripheral driver library is a set of drivers for accessing the peripherals found on the Tiva family of ARM Cortex-M based microcontrollers. They are not drivers in the pure operating system sense, i.e., they do not have a common interface and do not connect into a global device driver infrastructure. TivaWare simplifies the use of the device's peripherals. The capabilities and organization of the drivers are governed by the following:

- Written entirely in C programming language except where absolutely not possible
- Allows the use of peripherals in its common mode of operation
- Efficient usage of memory and processor.
- Where possible, computations that can be performed at compile time are performed in the drivers instead of at run time. They can be built with more than one tool chain.

The peripheral driver library provides support for two programming models namely, the direct register access model and the software driver model. Each model can be used independently or together based on the needs of the application or the programming environment desired by the developer. Use of the direct register access model generally results in smaller and more efficient code than using the software driver model. However, the direct register access model requires detailed knowledge of the operation of each register and bit field, as well as their interactions and any sequencing required for proper operation of the peripheral, the developer is insulated from these details by the software driver model, generally requiring less time to develop applications. The direct register access model and software driver model can be used together in a single application, allowing the most appropriate model to be applied as needed to any particular situation within the application. The general API functions used in this lab manual are given in [Table 1](#).

Table 1: API Functions Used in the Application Program

API Function	Parameters	Description
SysCtlClockSet(uint32_t ui32Config)	<ul style="list-style-type: none"> • ui32Config is the required configuration of the device clocking - logical OR of several different values: <ul style="list-style-type: none"> - System clock divider - Use of the PLL - External crystal frequency - Oscillator source 	This function configures the clocking of the device. The input crystal frequency, oscillator to be used, use of the PLL, and the system clock divider are all configured with this function.
SysCtlPeripheralEnable(uint32_t ui32Peripheral)	<ul style="list-style-type: none"> • ui32Peripheral is the peripheral to enable. 	This function enables a peripheral.
GPIOPinTypeGPIOOutput(uint32_t ui32Port, uint8_t ui8Pins)	<ul style="list-style-type: none"> • ui32Port is the base address of the GPIO port. • ui8Pins is the bit-packed representation of the pin(s). 	Configures pin(s) for use as GPIO outputs.
GPIOPinWrite(uint32_t ui32Port, uint8_t ui8Pins, uint8_t ui8Val)	<ul style="list-style-type: none"> • ui32Port is the base address of the GPIO port. • ui8Pins is the bit-packed representation of the pin(s). • ui8Val is the value to write to the pin(s). 	Writes a value to the specified pin(s).

Using Tera Term

Tera Term is open source freely downloadable terminal emulator software used for testing serial communication. The software can be easily downloaded from the internet. The procedure to use Tera Term as a serial terminal is given below:

1. Open Tera Term program and Select Serial Port. The corresponding port to which the Tiva is connected in the Tera Term is displayed in the **New Connection** Window as shown in [Figure 15](#). The serial port option is enabled only if the Tiva Drivers are properly installed, else it stays at TCP/IP. The Serial Port number will differ based on the USB to Serial Emulation and the connectivity.

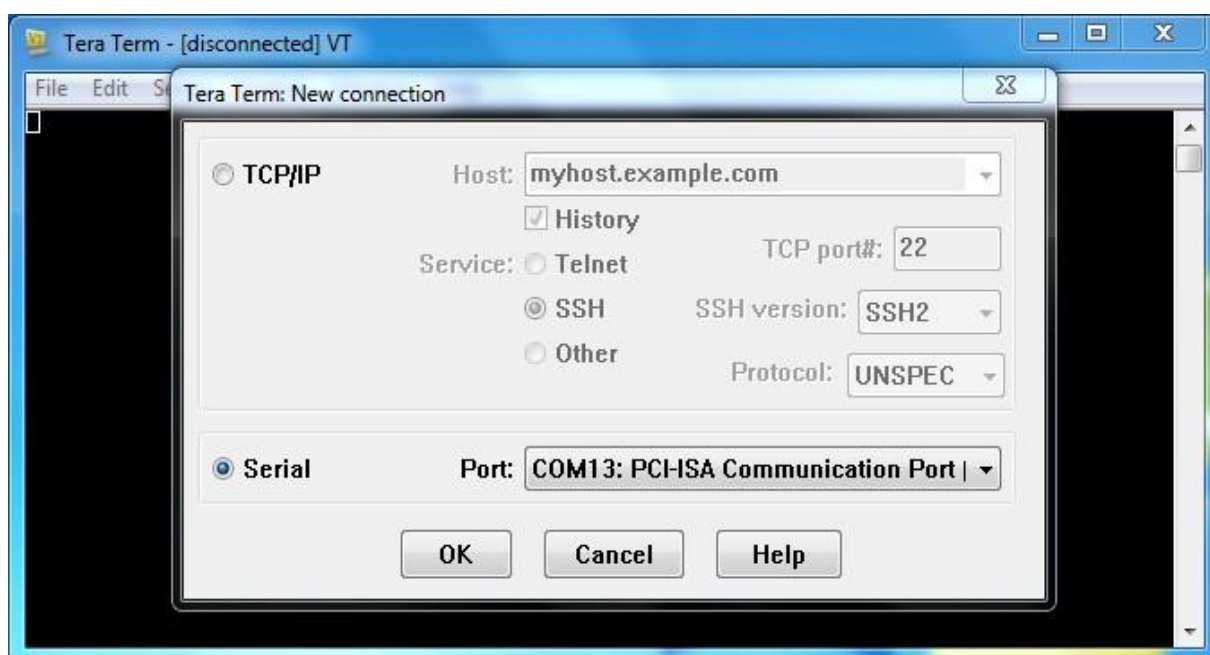


Figure 15 Tera Term New Connection Window

2. The port is opened with a default Baud Rate of 9600. Change the Baud Rate by selecting **Setup** —► **Serial Port**. Change the required Serial Port Setup parameters like Baud Rate, Data bits, Parity, Stop Bit as shown in [Figure 16](#).

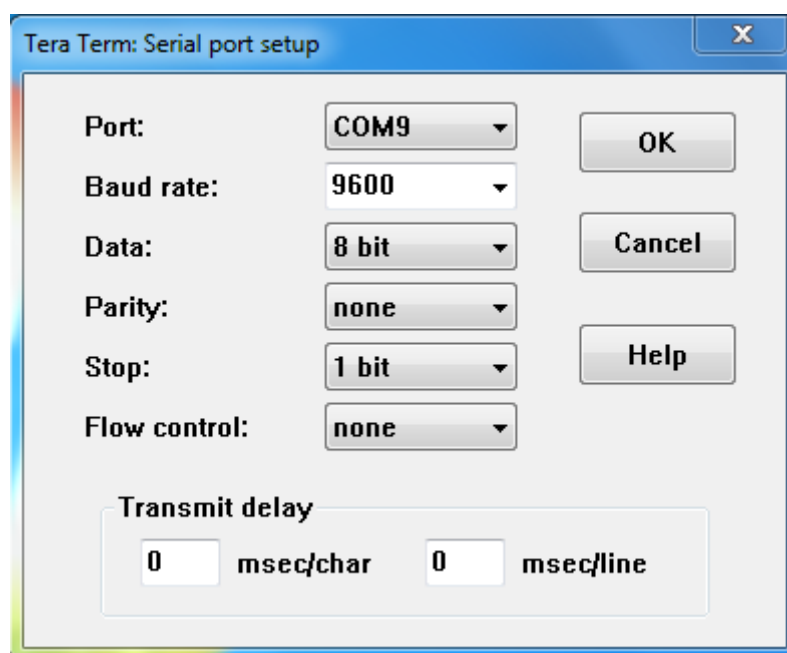


Figure 16 Tera Term Serial Port Setup Window

3. Select **Setup** → **Terminal** → **New Line** → **Receive** → **Auto** to configure the new line character as shown in [Figure 17](#).

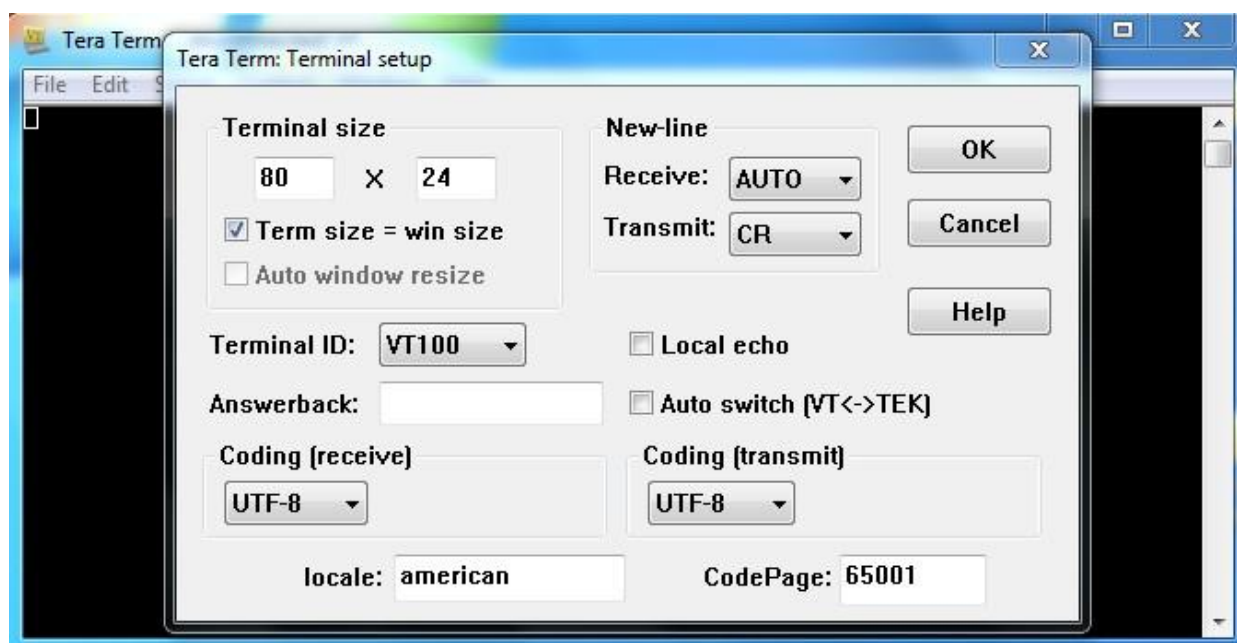


Figure 17 Tera Term Terminal Setup Window

Experiment 1
To Blink an Onboard LED

Topics	Page
1.1 Objective	22
1.2 Introduction	22
1.3 Component Requirements	24
1.4 Software	24
1.5 Procedure.....	26
1.6 Observation	27
1.7 Summary	27
1.8 Exercise.....	27

1.1 Objective

The main objective of this experiment is to configure the Tiva GPIO pins to blink the green on-board LED (connected to PF3) using C program.

1.2 Introduction

The EK-TM4C123GXL has three on-board LEDs which are connected to the GPIO pins PF1, PF2 and PF3 of the TM4C123GH6PM microcontroller. The software code in the TM4C123GH6PM toggles the PF3 output at fixed time intervals computed within the code. A HIGH on PF3 turns the LED On, while a LOW on PF3 turns the LED Off. Thus, a toggling output on the port pin PF3 blinks the LED connected to it. The functional block diagram as shown in [Figure 1-1](#) illustrates the working principle of the experiment.

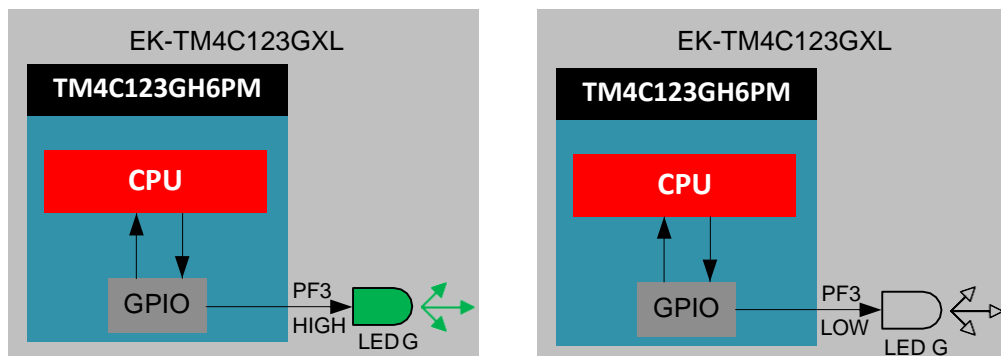


Figure 1-1 Functional Block Diagram

1.2.1 GPIO Module

The TM4C123GH6PM features a GPIO (General Purpose Input Output) module that is composed of six physical GPIO blocks. Each block corresponds to an individual GPIO port (Port A, Port B, Port C, Port D, Port E, Port F). The GPIO pins are connected to the on-board LEDs in the Launch-Pad as shown in the EK-TM4C123GXL schematics in [Figure 1-2](#).

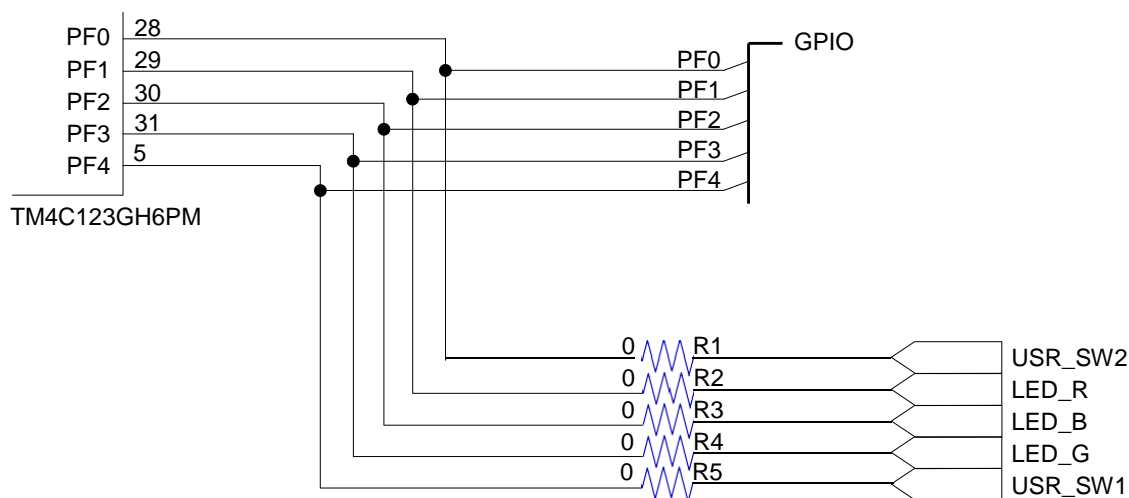


Figure 1-2 LaunchPad Schematics for GPIO Connected to LEDs

The features of GPIO are:

- Availability of up to 43 GPIOs, depending on configuration
- Highly flexible pin multiplexing allows usage of pins as GPIO or one of the several peripheral functions available
- Bit masking in both read and write operations through address lines
- Programmable control for GPIO interrupts:
 - Interrupt generation masking
 - Edge-triggered on rising, falling, or both
 - Level-sensitive on HIGH or LOW values
- Programmable weak pull-up, pull-down and open drain
- Programmable drive strength and slew rate control

Pin multiplexing of the GPIO signals allow alternate hardware functions of the pins. In the EK-TM4C123GXL, the GPIO pins are configured with pin functions as shown in [Table 1-1](#).

Table 1-1: GPIO Pin Functions and USB Device Connected

GPIO Pin	Pin Function	USB Device
PF4	GPIO	SW1
PF0	GPIO	SW2
PF1	GPIO	RGB LED (Red)
PF2	GPIO	RGB LED (Blue)
PF3	GPIO	RGB LED (Green)

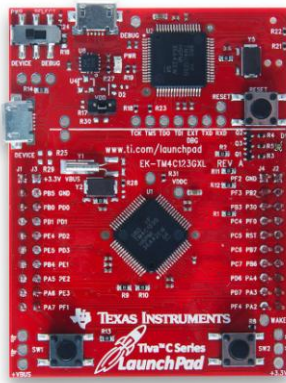
1.3 Component Requirements

1.3.1 Software Requirements

1. [Code Composer Studio](#)
2. [TivaWare_C_Series](#)

1.3.2 Hardware Requirements

Table 1-2: Components Required for the Experiment

S.No	Components	Specification	Image
1.	Tiva LaunchPad	EK-TM4C123GXL LaunchPad	
2.	USB Cable		

1.4 Software

The software for the experiment is written in C and developed using the CCS Integrated Development Environment (IDE). Refer to “[Project Creation and Build](#)” in the Getting Started section of this manual for project build and debug using CCS. The software is programmed into the target device TM4C123GH6PM on the EK-TM4C123GXL using the USB interface.

1.4.1 Flowchart

The flowchart for the code is shown in [Figure 1-3](#). The software written in C configures and enables the system clock to 40MHz. It then enables GPIO Port F of EK-TM4C123GXL and configures pin 3 (PF3) as output. A HIGH output at the GPIO PF3 turns the green LED ON connected to it and a LOW output turns the LED OFF. Hence, to blink the LED, the output is toggled at regular time interval using a while (1) loop. The time interval between the LED turning ON and OFF can be programmed by specifying the delay count value in the SysCtlDelay() function.

Calculation of Delay: The number of counts required to get a time delay is given by

Number of Counts = Time delay required * System Clock Frequency

In the program, for a time delay of 500ms

Number of counts = $500 * 10^{-3} * 40 * 10^6 = 20 * 10^6$

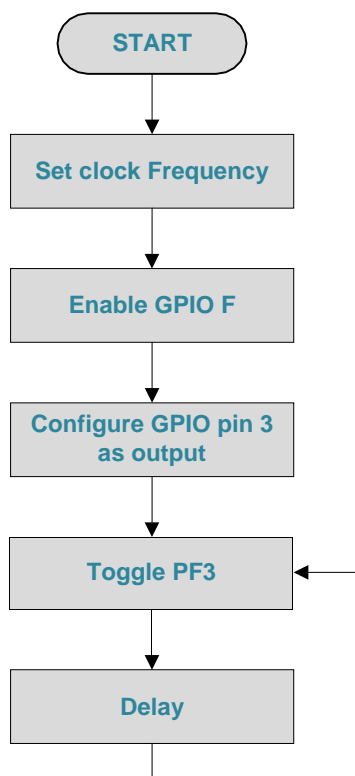


Figure 1-3 Flowchart for Blinking an Onboard LED

1.4.2 C Program Code to Blink an Onboard LED

```

#include <stdint.h>
#include <stdbool.h>
#include "inc/hw_types.h"
#include "inc/hw_memmap.h"
#include "driverlib/sysctl.h"
#include "driverlib/pin_map.h"
#include "driverlib/debug.h"
#include "driverlib/gpio.h"

int main(void)
{
    SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_XTAL_16MHZ|
SYSCTL_OSC_MAIN);
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);
    GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|
GPIO_PIN_3);

```

```

while(1) {

    GPIOPinWrite(GPIO_PORTF_BASE,GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, 0x08);

    SysCtlDelay(20000000);

    GPIOPinWrite(GPIO_PORTF_BASE,GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, 0x00);

    SysCtlDelay(20000000);

}

}

```

Table 1-3: API Functions Used in the Application Program

API Function	Parameters	Description
SysCtlClockSet(uint32_t ui32Config)	<ul style="list-style-type: none"> • ui32Config is the required configuration of the device clocking - logical OR of several different values: <ul style="list-style-type: none"> - System clock divider - Use of the PLL - External crystal frequency - Oscillator source 	This function configures the clocking of the device. The input crystal frequency, oscillator to be used, use of the PLL, and the system clock divider are all configured with this function.
SysCtlPeripheralEnable(uint32_t ui32Peripheral)	<ul style="list-style-type: none"> • ui32Peripheral is the peripheral to enable. 	This function enables a peripheral.
GPIOPinTypeGPIOOutput(uint32_t ui32Port, uint8_t ui8Pins)	<ul style="list-style-type: none"> • ui32Port is the base address of the GPIO port. • ui8Pins is the bit-packed representation of the pin(s). 	Configures pin(s) for use as GPIO outputs.
SysCtlDelay(uint32_t ui32Count)	<ul style="list-style-type: none"> • ui32Count is the number of delay loop iterations to perform. 	Provides a small delay.
GPIOPinWrite(uint32_t ui32Port, uint8_t ui8Pins, uint8_t ui8Val)	<ul style="list-style-type: none"> • ui32Port is the base address of the GPIO port. • ui8Pins is the bit-packed representation of the pin(s). • ui8Val is the value to write to the pin(s). 	Writes a value to the specified pin(s).

1.5 Procedure

1. Connect the EK-TM4C123GXL to the PC using the USB cable supplied.
2. Build, program and debug the code into the LaunchPad using CCS to view the status of the green LED.

1.6 Observation

The green LED blinks on successful execution of the program. The rate of blinking can be altered by changing the delay count value in the program.

1.7 Summary

In this experiment, we have learnt to configure and program the GPIO pins of Tiva and have successfully programmed the GPIO pin PF3 to blink the on-board green LED.

1.8 Exercise

1. Program the TM4C123GH6PM to blink any other two on-board LED's of EK-TM4C123GXL.
2. Program the TM4C123GH6PM to blink all on-board LED's of EK-TM4C123GXL at a time.

