Getting Started with TivaWare Launchpad - Basics

Program a 32-bit Tivaware Board (TM4C123x series) from Texas Instruments using Embedded C from scratch. Contains TM4C123G ARM-cortex MCU.

Difficulty **Easy** | Duration **1 hour(s)** | Categories **Electronics, Machines & Tools, Robotics**

Cost **25 USD ($)**

Contents

Introduction

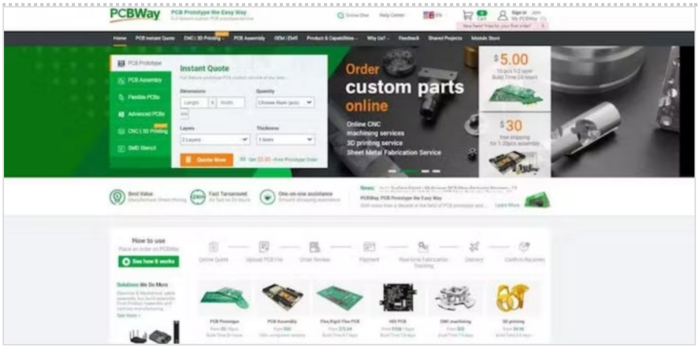
Texas Instruments, also known as TI, is an embedded design company, that designs, manufactures and sells embedded systems. These systems include semiconductors, integrated circuits, development boards, etc, for industrial, commercial, automotive, and personal electronics. Tivaware is an initiative by TI to manufacture a series of MCUs that support a common SDK.

Materials

Tools

Step 1 - Get PCBs For Your Projects Manufactured

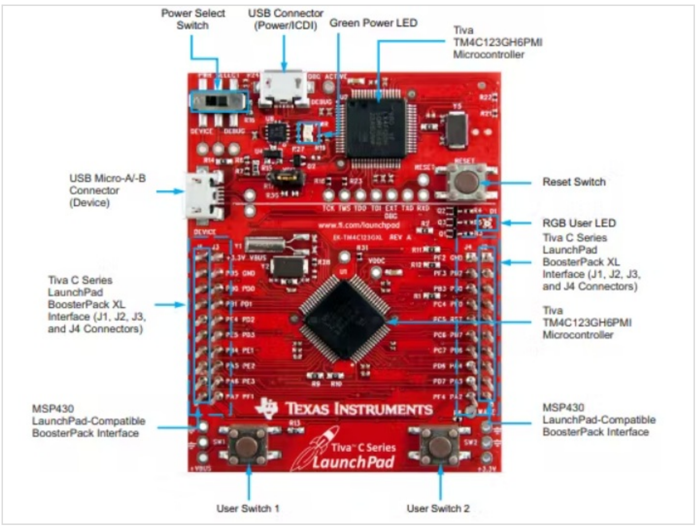
You must check out PCBWAY to order PCBs online for cheap!
You get 10 good-quality PCBs manufactured and shipped to your doorstep for cheap. You will also get a discount on shipping on your first order. Upload your Gerber files onto PCBWAY to get them manufactured with good quality and quick turnaround time. PCBWay now could provide a complete product solution, from design to enclosure production. Check out their online Gerber viewer function. With reward points, you can get free stuff from their gift shop.



Step 2 - Getting Started ⍰

The TivaWare for software development kit (SDK) provides all of the components necessary for engineers to evaluate and develop applications for the Texas Instruments TM4C Arm Cortex - M4F device family. Based on Embedded C language, it allows developers to get started quickly, uses already-tested development methods and drivers that take less time to build and create.
In our case, we'll be using a basic microcontroller board TM4C123GXL, which has the same properties as an STM Nucleo Board. TI Boards for this series are termed as 'Launchpad' due to its ability to support other booster drivers and boards that'll give extra support to provide more features.

Step 3 - Hardware Section ⍰


Power Select Switch
USB Connector (Power/ICDI)
Green Power LED
Tiva TM4C123GH6PMI Microcontroller
USB Micro-A/-B Connector (Device)
Reset Switch
RGB User LED
Tiva C Series LaunchPad BoosterPack XL Interface (J1, J2, J3, and J4 Connectors)
Tiva C Series LaunchPad BoosterPack XL Interface (J1, J2, J3, and J4 Connectors)
Tiva TM4C123GH6PMI Microcontroller
MSP430 LaunchPad-Compatible BoosterPack Interface
MSP430 LaunchPad-Compatible BoosterPack Interface
Texas Instruments
Tiva C Series LaunchPad
User Switch 1
User Switch 2

This **Tivaware Launchpad (TM4C123GXL)** has the following features :

- Tiva TM4C123GH6PMI microcontroller
- Motion control PWM
- USB micro-A and micro-B connectors for USB device, host, and on-the-go (OTG) connectivity
- RGB user LED
- Two user switches (application/wake)
- Available I/O brought out to headers on a 0.1-in (2.54-mm) grid
- On-board ICDI
- Switch-selectable power sources: – ICDI – USB device
- Reset switch
- Preloaded RGB quickstart application

Let us check more about the TM4C123GH6PMMCU :

- 80MHz 32-bit ARM Cortex-M4-based microcontrollers CPU
- 256KB Flash, 32KB SRAM, 2KB EEPROM
- Two Controller Area Network (CAN) modules (requires CAN transceivers)
- USB 2.0 Host/Device/OTG + PHY
- Dual 12-bit 2MSPS ADCs, motion control PWMs
- 8 UART, 6 I2C, 4 SPI

This board provides more features than any other Arduino Board. What's special about it, is the 32-bit MCU taking control of the board.

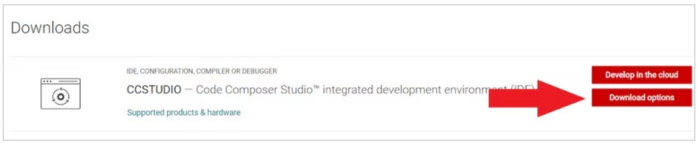
Step 4 - Software Requirements ??

To be able to upload and run codes on the Tiva launchpad, we need to download the IDE and library, then make a couple of configurations. Below are the software to be downloaded -

- Download and Install Code Composer Studio from here. Then click on '**Download Options**' > select '**Windows single file (offline)**'
- Now, Download the library for TM4Cxx family - **SW-TM4C** from here.Then click on '**Download Options**' > select '**TivaWare for TM4C Series**'
- Move the downloaded library file to a separate folder. It can also be kept inside the Code Composer Studio folder after its installation. ***Note the location of this file.***
- There are other libraries for the particular board. Even though I used the EK-TM4C123GXL board in this article. You can use any other board, and hence download the whole library, instead of a particular board series.

Installing Code Composer Studio -

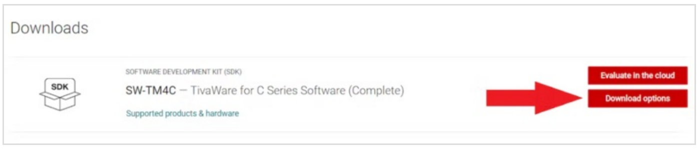
- Run the CCS (Code Composer Studio) setup file.
- Choose the Location to install the CCS. Tip - All the code files are saved in that location.
- Choose '**Custom Installation**', since we will only be using the TM4C123GXL board, select '**TM4C12x ARM Cortex M4F core-based MCU**'


Downloads

IDE, CONFIGURATION, COMPILER OR DEBUGGER

CCSTUDIO — Code Composer Studio™ integrated development environment (IDE)

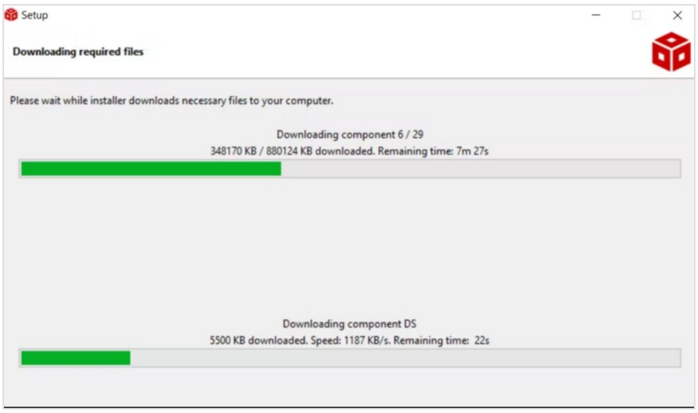
Supported products & hardware


Downloads

SOFTWARE DEVELOPMENT KIT (SDK)

SW-TM4C — TivaWare for C Series Software (Complete)

Supported products & hardware


Setup

Downloading required files

Please wait while installer downloads necessary files to your computer.

Downloading component 6 / 29
348170 KB / 880124 KB downloaded. Remaining time: 7m 27s

Downloading component DS
5500 KB downloaded. Speed: 1187 KB/s. Remaining time: 22s

Step 5 - Running CCS forTM4C Library -

- Open CCS and create a new project > select Device - **TM4C123GH6PM**, connection - **Stellaris ICD** > Create project with main.c file
- Now, right-click on the **project** > and select **properties** > go to **Build** > go to the **ARM Compiler** section > click on the **Include Options** > click the **icon with** a **green plus sign** to add file > browse to the **downloaded library file** (previously noted location)
- Now, let us use the **ARM Linker** > go to **File Search Path** > add the **driverlib.lib** file here (**found at CCS installed location** - ..\..\TivaWare_C_Series-v.er.sion\driverlib\ccs\Debug\driverlib.lib).
- Once done, click 'Apply and Close' to continue with the build.

Alright! We can now begin with Write - Build - Upload to the Launchpad !!

?? We can also follow the Documentation for guidance on the functions used for the peripherals of the board. Download the documentation - **TivaWare™ Peripheral Driver Library for C Series User's Guide(Rev. E).** ??

Step 6 - PORTs - How to use them? ⍰

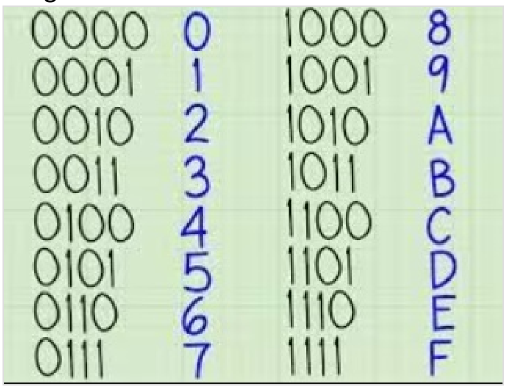
Ports are the peripheral pins on an embedded system, or SoC to communicate with the external environment. Whenever a port receives a HIGH or LOW signal, it communicates throughout the system.

- A PORT is usually bit-packed, i.e. each port is represented with an 8-bit binary number. The number count starts from right to left. So when we mention a pin's address as **0x0E** - it means we are pointing at the **pins which are O** according to the below representation -**X X X X O O O X**
- If we **convert 0x0E to** a **binary number, it becomes - 00001110**. Here pins 1, 2, and 3 are pointed at to be controlled. Because the pin counting goes reverse - 7 6 5 4 3 2 1 0 (8 pins) And therefore it looks similar to the above representation.
- When we mention **GPIO_PIN_1,** we mean **0x02,** in binary - **0000 0010.** This turns **pin 1 on port F** to emit a **HIGH signal**. You can access the Github Repository of the project from here.
- For example, if we have enabled port A, which has 8 pins. And we require the use of only 2 of the pins on that port. To enable those 2 pins according to their arrangement, we can use an 8-bit number to assign them. **PortA - X X X X O X O X,** which means - **0x0AW**e have enabled **PIN 1** and **PIN 3**.
- Above we can see that **O** represents the pins we require to enable. We can use hexadecimal format to assign. Therefore we shall be using the function as**GPIOPinTypeGPIOOutput(GPIO_PORTA_BASE, 0x0A);**
- **Similarly,** if we require to turn ON the LEDs on this port alternatively, we can use the below command -

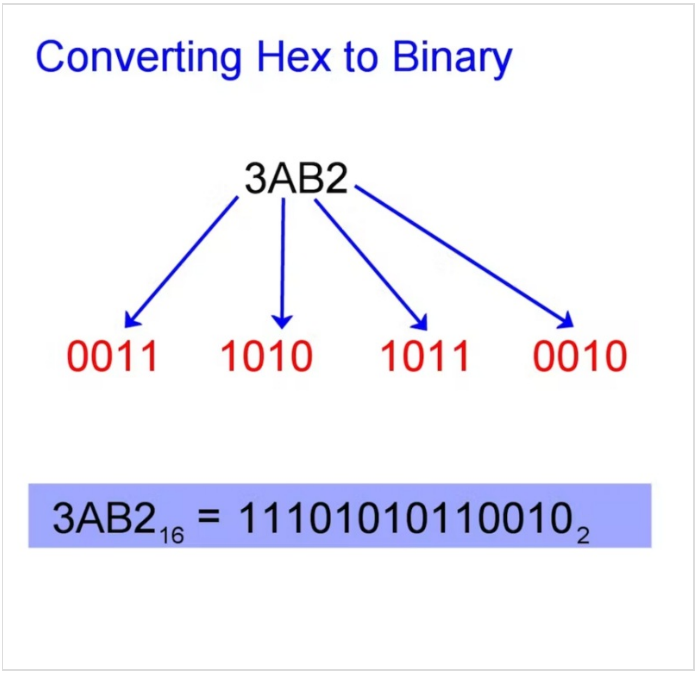
```
while(1)
{
GPIOPinWrite(GPIO_PORTA_BASE,0x0A, 0x02);
SysCtlDelay(1333333);
GPIOPinWrite(GPIO_PORTA_BASE,0x0A, 0x08);
SysCtlDelay(1333333);
}
```

- As you can see above, when the command with value 0x02 is run, only PIN 1 will be giving a HIGH signal - **0000 0010**
- When the command with value 0x08 is run, only PIN 3 will be giving a HIGH signal - **0000 1000**
- In both commands, the other **LED will turn off since it has 0 on its pin**.

Congratulations! You can now understand other functions that use the ports, directly from the documentation by TI.



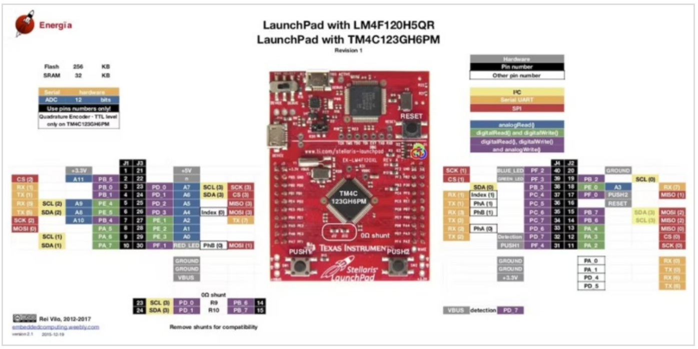
0000	0	1000	8
0001	1	1001	9
0010	2	1010	A
0011	3	1011	B
0100	4	1100	C
0101	5	1101	D
0110	6	1110	E
0111	7	1111	F



Step 7 - First Project - LED Blinking ?

We shall be using the inbuilt RGB LED present on the board itself. The LEDs are internally connected to **GPIO 1 on Port F**. You can access the Github Repository of the project from here.

- We can see from the above diagram, the pin we are trying to control has an **inbuilt RED LED**. If you look carefully, the pin name is **PF_1**. Rest **PF_2** and **PF_3** are **Blue** and **Green LED** respectively.
- Also, we require to set the System Clock Frequency to 40 MHz. Therefore, we'll use the below parameters (in sequence) -- **SYSCTL_XTAL_16MHZ** - Precise Internal OSC- **SYSCTL_USE_OSC** - Forwarding the OSC- **SYSCTL_USE_PLL** - Generates 400 MHz, then divided by 2 = 200 MHz- **SYSCTL_SYSDIV_5** - OSC divided by 5 = **40MHz**
- Using the **SysCtlClockSet()** function, we can set the clock configuration required - **Final Clock frequency = 40 MHz**



```
SysCtlClockSet(SYSCTL_SYSDIV_5
```