SE101—Lab Introduction

Rollen D'Souza, Patrick Lam, Mahesh Tripunitara

Fall 2016
Last Updated September 16, 2016

This document explains how to setup and use the Tiva C Series (TM4C123G) board you have been provided using the Energia Integrated Development Environment (IDE) [1]. Consider this an introduction to help familiarize you with the microcontroller and programming environment, not a comprehensive guide. For more information about the board, see the documents referenced by this guide as well as those cited on Learn.

Tiva C Series—TM4C123G

The Tiva C Series microcontroller board is a versatile, yet reasonably powerful, board designed and produced by Texas Instruments. Your specific board is generally used to evaluate the Tiva C Series Architecture, which they describe as:

> Targeting performance and flexibility, the Tiva C Series architecture offers a 80 MHz Cortex-M with FPU, a variety of integrated memories and multiple programmable GPIO. Tiva C Series devices offer consumers compelling cost-effective solutions by integrating application-specific peripherals and providing a comprehensive library of software tools which minimize board costs and design-cycle time. Offering quicker time-to-market and cost savings, the Tiva C Series microcontrollers are the leading choice in high-performance 32-bit applications. [2]

Energia

Energia is the Integrated Development Environment (IDE) you will use to compile and deploy code onto your microcontroller board [3]. If you are not using the lab computers, you can download Energia from `http://energia.nu`. Energia is already pre-installed on the lab computers and can be started by:

1. [Windows] Click the Start Button
2. Click on Run
3. Paste: `Q:\eng\ece\energia-0101E0016`
4. Run

Once Energia starts, you will see something that looks like Figure 1. For now, ignore the code generated by Energia. Instead, first ensure that your computer can communicate with your board. To do this, check to see that Energia is looking for the correct board model by:

1. Select *Tools*
2. Select *Board*
3. Now scroll to find and select: LaunchPad (Tiva C) w/tmc4c123g (80Mhz)

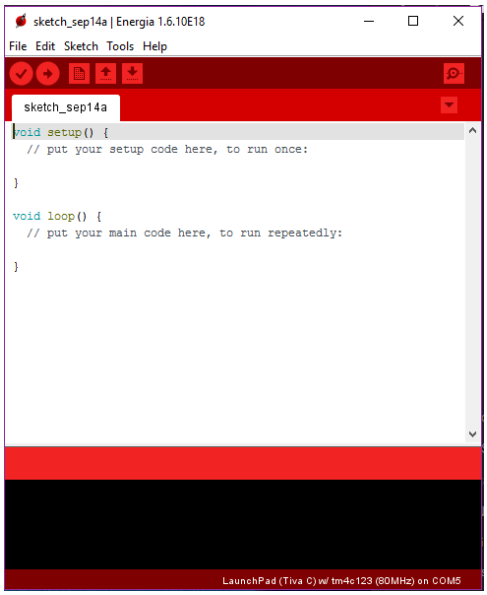

Figure 1: Energia on startup.

Then you need to verify that Energia communicates over the correct port. To do this:

1. Select *Tools*
2. Select *Serial Port*
3. You should find two COM options. Make sure to select COM3 (a checkmark appears besides the selected option). *Note that this choice is only guaranteed to work on lab computers. The correct COM port may be different on your personal machine.*

With that, you are all set to communicate with the board! To test that everything works, press the red right arrow button (second button from the left) and read the black console at the bottom. After a slew of messages, you should see "Success!" printed.

Your First Tiva Program

You may have noticed that Energia generates a bit of code for you in the editor, namely the two functions `setup` and `loop`. The `setup` function is executed exactly once, immediately after the code is loaded onto the board. Normally, this function is used to initialize variables and configure pins to be inputs or outputs. The `loop` function is where most of your code will reside. It is executed repeatedly until you turn off the board or load a new program.

Let us begin coding! Listing 1 shows an example program that flashes a light emitting diode (LED). By the way, we absolutely expect your code to have fewer comments than that in Listing 1; it is excessive and for explanatory purposes only. Use good variable names and explain anything that's unexpected (or better yet, rewrite that code.)

This program will turn the red LED on and then off, alternating every 5 seconds. The sequence will repeat forever. Let us inspect the code closely:

- Line 2: `#define` allows us to to give another name for some expression. In this case, we have redefined `LED` so that it will mean `RED_LED`. The compiler implements this by simply replacing all instances of `LED` with `RED_LED`. You might be wondering: where is `RED_LED` defined? It can be found in the hardware header for this board. You might also be wondering: why are we doing this? Well, if we wanted to refer to a hypothetical `GREEN_LED` instead of `RED_LED`, a single change to the `#define` would suffice.

```
1 // Define LED to refer to the RED LED.
2 #define LED RED_LED
3
4 void setup() {
5   // Configure the LED pin to be an output.
6   pinMode(LED, OUTPUT);
7 }
8
9 void loop() {
10  // Raise the voltage across the LED pin.
11  digitalWrite(LED, HIGH);
12
13  // Wait for 5 seconds (LED stays on).
14  delay(5000);
15
16  // Drop the voltage across the LED pin.
17  digitalWrite(LED, LOW);
18
19  // Wait another 5 seconds (LED stays off).
20  delay(5000);
21 }
```

Listing 1: Flashing the Red LED.

- Line 6: `pinMode(LED, OUTPUT)` configures the board to use the LED pin as an output pin. You *must* do this to permit modification of the LED state by changing voltage between `HIGH` and `LOW`. You are only required to do this once for your program, before modifying LED state, and this is why it is placed within the `setup` function.
- Line 11: `digitalWrite(LED, HIGH)` raises the voltage across the pin connected to the LED. The function takes two parameters, a pin and the new state: `HIGH` or `LOW`. Although in this circumstance `HIGH` illuminates the LED, not all pins behave in this intuitive manner! Read the documentation of the board to determine which state results in which behaviour.
- Line 14 - `delay(5000)` waits for 5 seconds. `delay`'s first parameter is the amount of milliseconds to wait.

After entering your code, you need to *compile* and *upload* your code to the board. To compile your code, simply click the check mark icon (first button from the top left). You can check for successful compilation in the black console at the bottom. Once you have compiled your code, you can upload the code using the right arrow button beside the compile button. *You must compile and then upload your code everytime you modify it. Any experienced person will have been bitten by the "running the wrong code" problem more than once.*

Now you are ready to create more complicated programs. Obviously you may need to use more devices than just the `RED_LED`! Pin defines can be found in the Energia guide on Launchpad TM4C123GXL [4].

Aside from that guide, Energia has a wealth of examples that you can learn from, including a demo of how the LCD works on the Orbit Booster Pack. Demos can be found under *File/Examples* in the Energia main menu. The LCD example is found under the *OrbitBoosterPack* sublevel.

References

[1] Texas Instruments. (2014). *ARM®Cortex®-M4F Based MCU TM4C123G Launchpad^{TM}Evaluation Kit* [Online]. Available: http://www.ti.com/tool/ek-tm4c123gxl

[2] Texas Instruments. (2014). *Tiva^{TM} TM4C123GH6PM Microcontroller Data Sheet* (Rev. E) [pdf]. Available: http://www.ti.com/lit/pdf/spms376

[3] Energia. (2016). *Energia* [Online]. Available: http://energia.nu/

[4] Energia. (2016). *Guide_TM4C123Launchpad — Energia* [Online]. Available: http://energia.nu/pin-maps/guide_tm4c123launchpad/