# EEC 134AB

*Application Note*

Implementing Tiva on Radar System

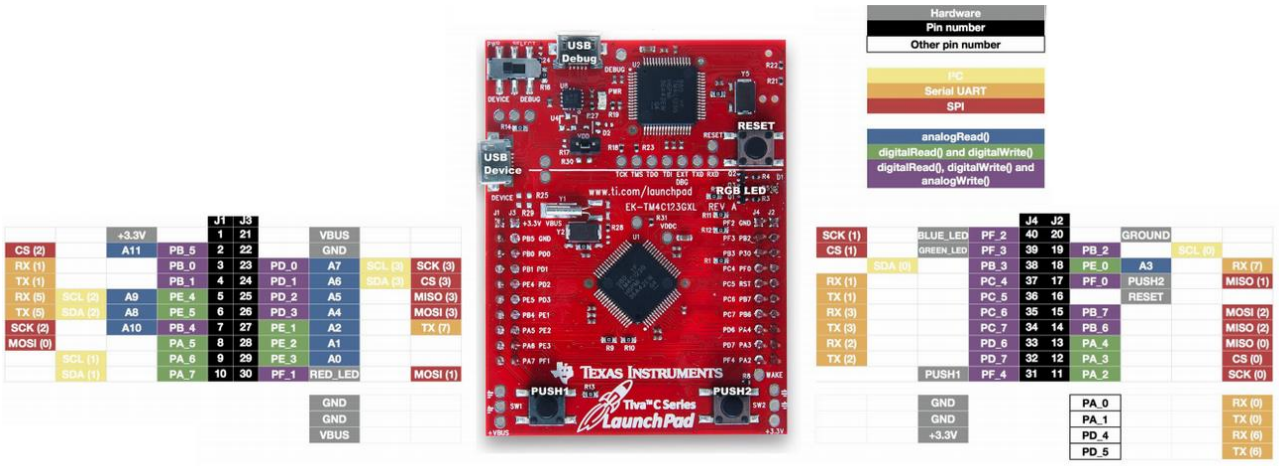
By: Genevieve Kam

## Introduction:

This application note will discuss implementing Tiva C Series on the radar system. The paper will explain what a Tiva C Series TM4C123GXL is and the functionalities of the Tiva. It will also discuss the implementation of the Tiva with the radar system.

## Tiva C Series:

The Tiva C Series LaunchPad is a microcontroller that is capable of controlling peripheral devices. The Tiva LaunchPad has an ARM Cortex 80MHz CPU so it is ideal for being a microprocessor to have lower heat due to having a smaller amount of transistors vs an x86 processor. The 256 kB of flash memory, 32kB of RAM, and 2 kB EEPROM enables us save data onto the microcontroller as well as give us some temporary space for running projects. The 12-bit ADC allows us to have a way to convert analog signals to digital signals without having to connect an external ADC. We may want to consider an external ADC if we need more than 12-bits of resolution. For digital signal processing, we will need to use an ADC to measure, filter or compress the analog signals and convert them to usable digital signals. The theory behind using DSP for a radar system is to take the analog signals that we receive from pinging certain locations and then quantifying that location by converting that analog signal to a digital signal to determine said location. The image below the functions that map with the pinout standard.




## DSP:

After the RF signal goes thru the PCB portion of the project, the output signal is fed into an ADC. The group decided to use Tiva C Series TM4C123GXL to handle all the signal processing and to display the results on the Kentec Electronic EB-LM4F120-L35. That way everything is processed within the microcontroller, and not use a laptop like in quarter one.

Digital signal processing (DSP) with the mathematical manipulation representation, and transformation of the information of signals to modify to improve. The representation of discrete-time signal processing is based on processing sequences of numbers or symbols indexed on integer variables rather than functions of continuous independent variable. Processing is implemented using digital computation.

## Getting Started:

The first thing is to do is to install Code Composer Studio. The link provided, Quick Start Guide, show you how the install the development tool and how to use it to build and run an example application on the Tiva LaunchPad.

What I learned from this project is to be able to read the pinouts of the Tiva LaunchPad and decide what pins I am able to use to control the radar system. There are many configurations

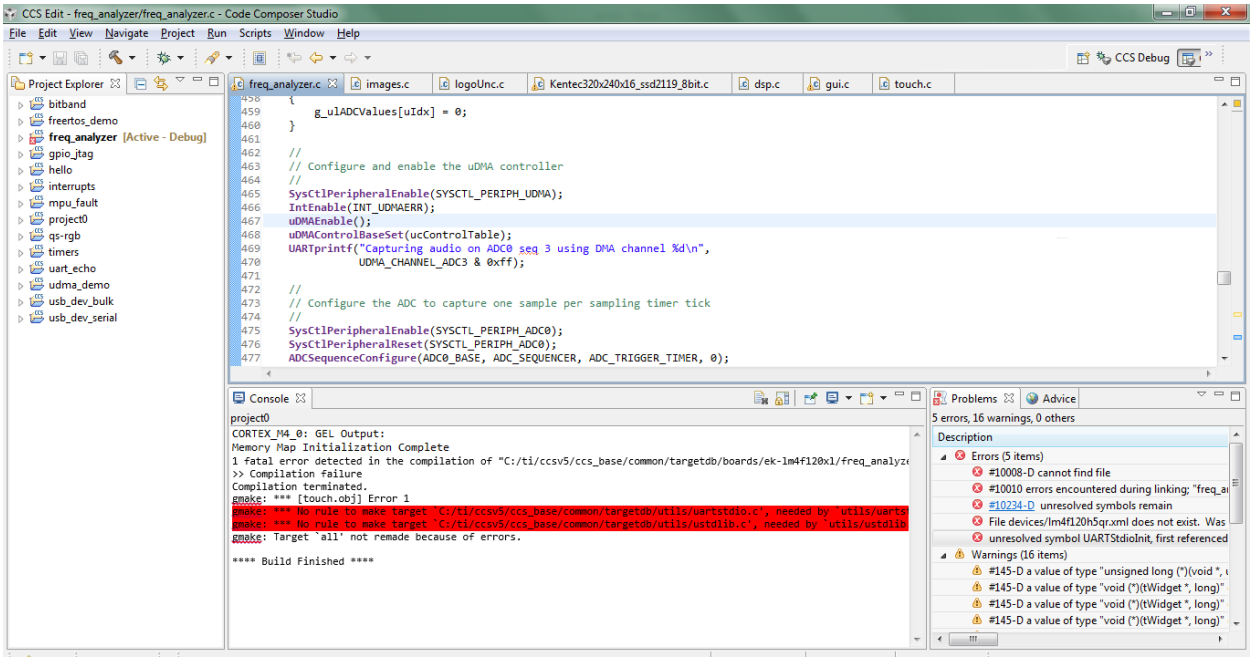
that the LaunchPad could take based on the type of communication that we need. For example we can set up CAN, SSI, PWM, SPI, UART, I2C and GPIO. If we need more of one and less of another, all we need to do is go into the code and in the setup definitions, we can specify the type of communication we need and the pin that we are putting the peripheral device on. I learned how to do a simple "hello world" type program by downloading the "Project 0" and putting the code into "Code Composer" and running the demo onto the Tiva LaunchPad. Using this demo, I am able to control the onboard LEDs and toggle them with the push of the onboard user switches. After getting a function starting code to work with, I started thinking about the type of communications that I would need to use to communicate with the radar system.

In addition, following the workshop, LaunchPad Workshop, gave me a better understanding and a model to help me see how the Tiva works.

**Frequency Analyzer:**

A great example I followed is EuphonistiHack Blog. He has already created a frequency analysis system based on the Launchpad with all the hardware and software design open sourced. The frequency analyzer helps capture the RF signal using Fourier transform. Here the link that provides an overview of the project, Euphonistihack. Following the instructions, I download his open source code on gethub. Upon downloading the source code, I had went through the main file, dsp file, and gui file, and the rest of the files I left alone. Because he used StellarisWare instead of TivaWare, I had to go through the code changing prefixes such as from `sRect.sYMin` to `sRect.i16YMin`. If you see table 3, in the link below, they both do the same function, but different for TivaWare and StellarisWare. This link, StellarisWare® to TivaWare™, is very helpful in converting from StellarWare to TivaWare.

Although I manage to change the code, there were more bugs that popped up.




One of the issues is not able to detect the target configuration anymore. I believe that might the reason why the touch screen stopped working as well. It seemed to be functioning at first, but the screen wouldn't work anymore.

## Conclusions:

Unfortunately, I was having trouble configuring the code. In addition, the gui file was having technical issues and it wasn't able to display anything. In the end, we decided to go back to the MATLAB program and work from there. Some lessons are I learn is being unfamiliar with the Tiva was much harder than I thought. I should read more on the material.

## Suggestions:

Some suggestion I recommend using the Code Composer rather than Keil Real View MDK or the Energia Development Tool because the Code Composer is easier to work with and the other two development tool. Also, for my computer, there was a very long and lagging process starting up the Keil and Energia development tool.

I also recommend to future students is to get help and work with others, especially if you are not similar with embedded systems. Asking for help from other groups, and/or working with a group partner will ensure you have a better understanding the material.

In addition, be patient. Take the time to read each line of code and understand it. Overall, since there were several files of code, I had trouble distinguishing what each section of the code did as well as where one file correlated with the other files. Debugging code is a long process, and change one line can potentially terminate the program.