Lab 1 – Introduction (Continued) to TI Code Composer Studio v7 IDE and TI TMS320F28027 Piccolo Microcontroller

NOTE: THIS DOC IS WRITTEN ASSUMING A LAUNCHPAD IS BEING USED; IF YOU HAVE ANOTHER PLATFORM, THERE SHOULD BE ONLY MINOR DIFFERENCES TO WATCH OUT FOR

NOTE: SOME OF THE PICS IN THIS DOC WERE TAKEN WITH CCS VERSION 5; VERSION 7 SHOULD BE SIMILAR

In this lab you will learn how to:

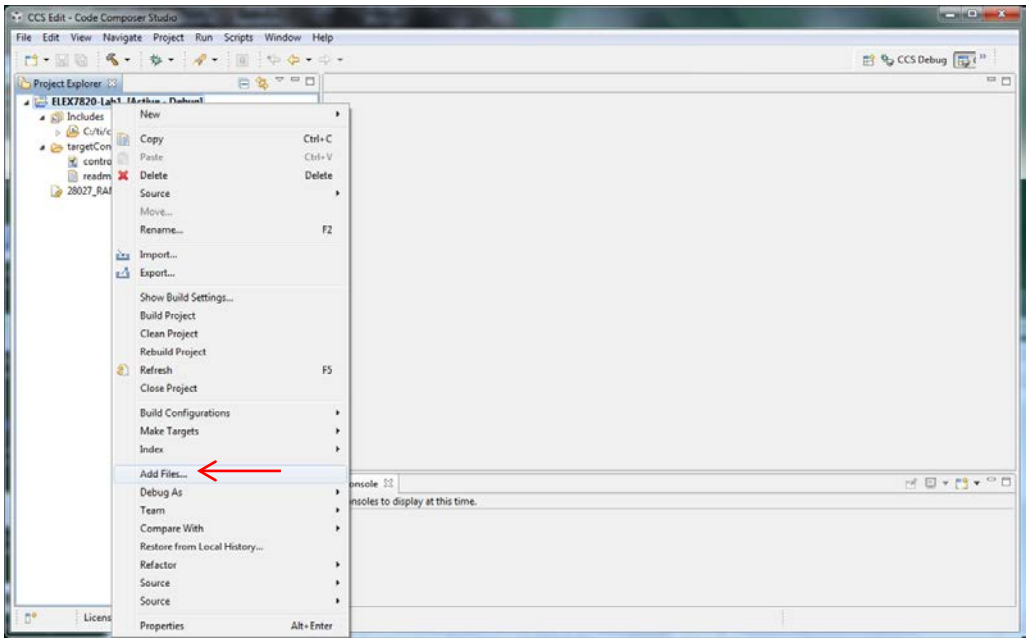
- Add existing files to your new Project, including TI's header files
- Add an existing main program that flashes an LED on the LaunchPad
- Observe the hierarchical structure of the Peripheral information for the CPU Timers and GPIO pins in their associated header files
- Observe and modify variables in the Variables window
- Observe the hierarchical structure of the Peripheral information for the CPU Timers and GPIO pins in the Registers window
- Observe the" look-ahead" context feature when entering a Peripheral in the C code
- Modify the main code to configure an additional GPIO pin as an output (not GPIO0) and use this line to indicate on a scope the utilization of a simple FIR dot product that you shall add to the C code

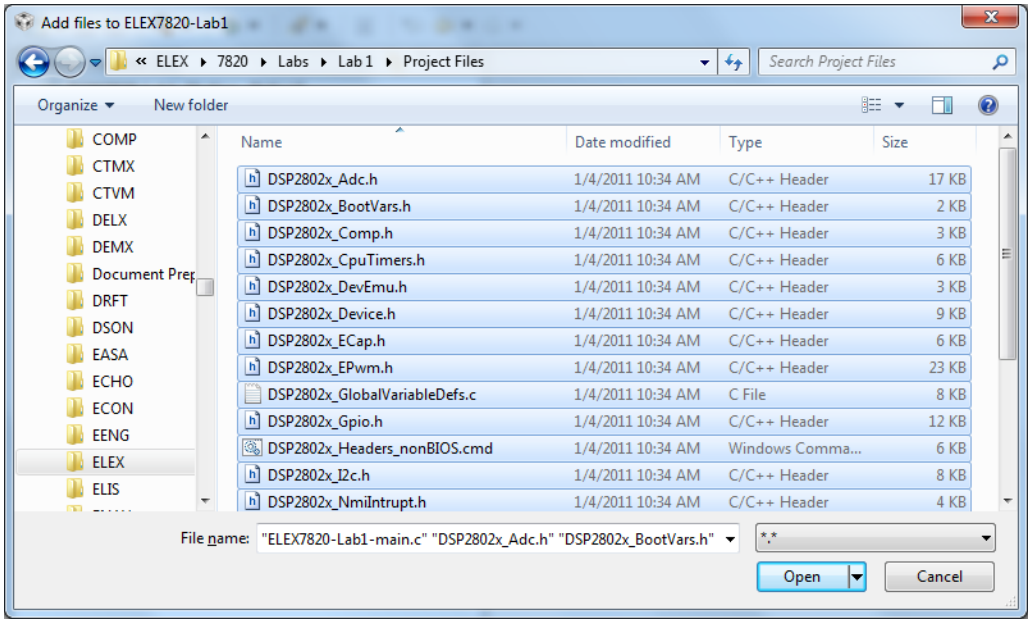
See last page for details on what to submit.

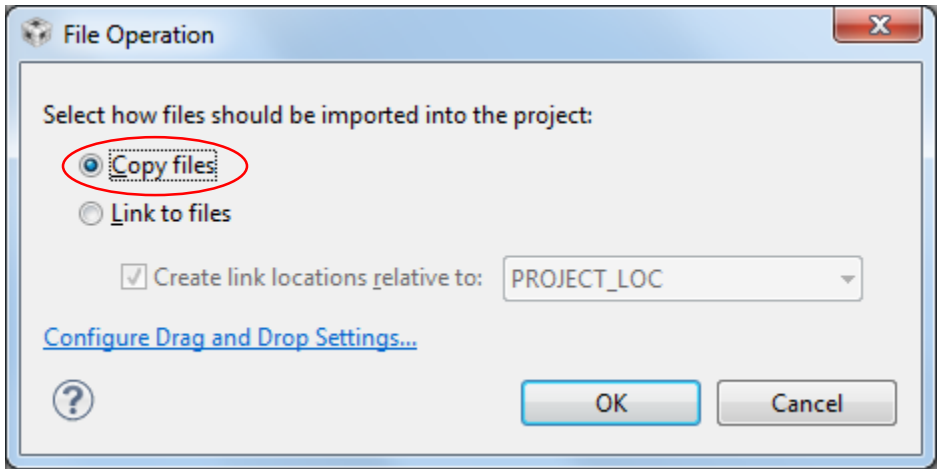
Create a New Project

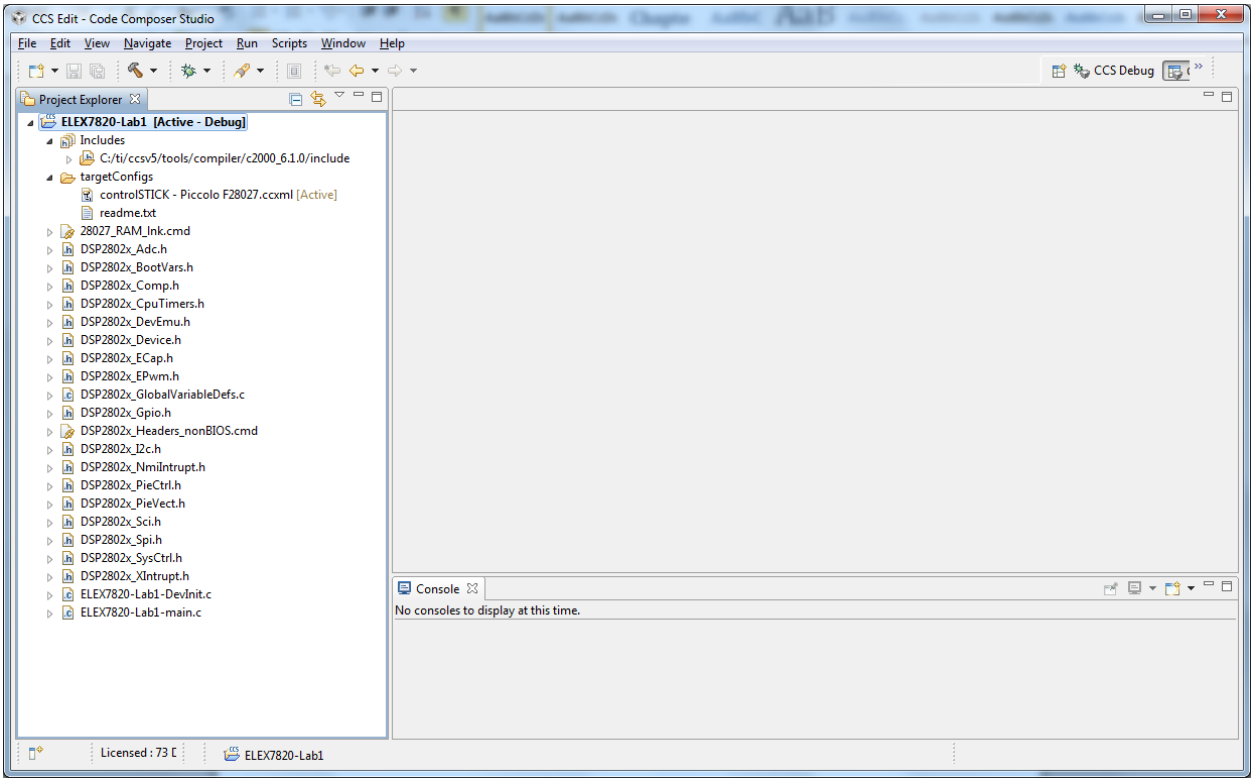
Open CCS and create a new empty CCS project called “ELEX7820-Lab1” in the workspace and configure it appropriately for the Piccolo target. The project should be configured for Debug (not Release) and set as Active.

Add Existing Files (from D2L) to New Project (right-click name of project to access menu)



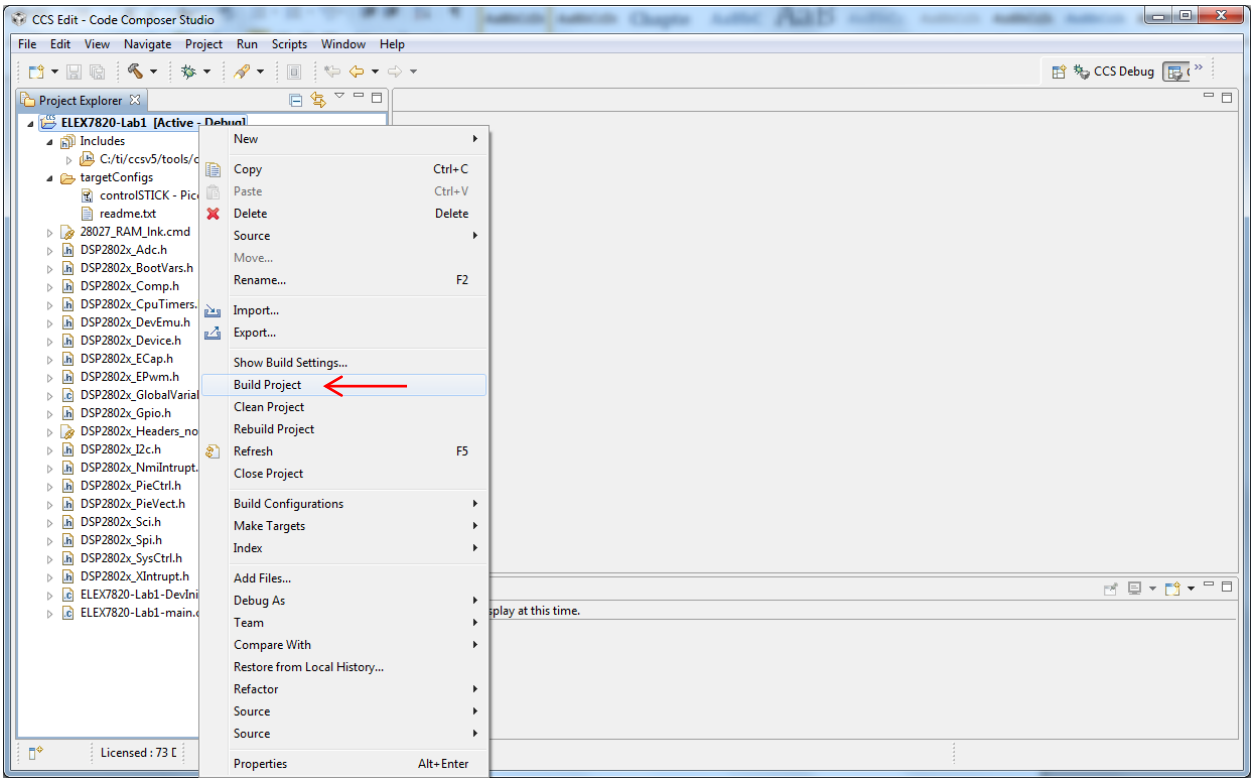


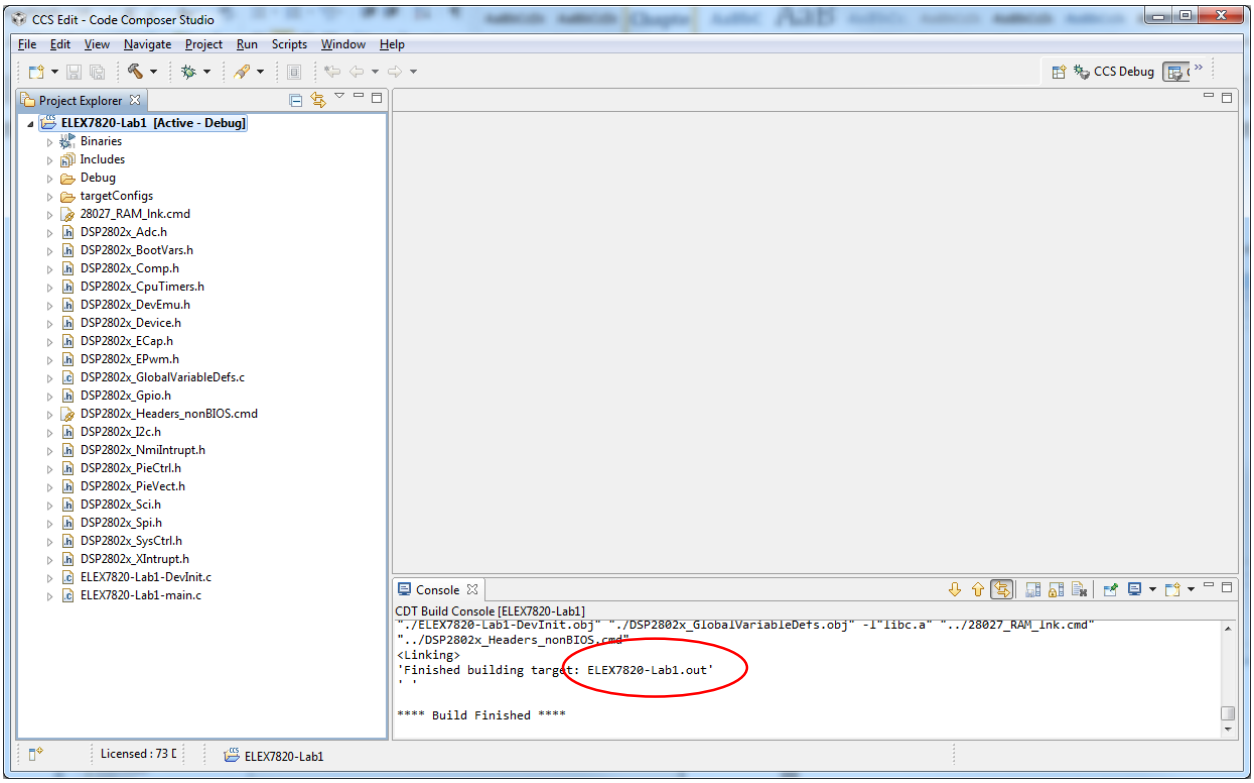




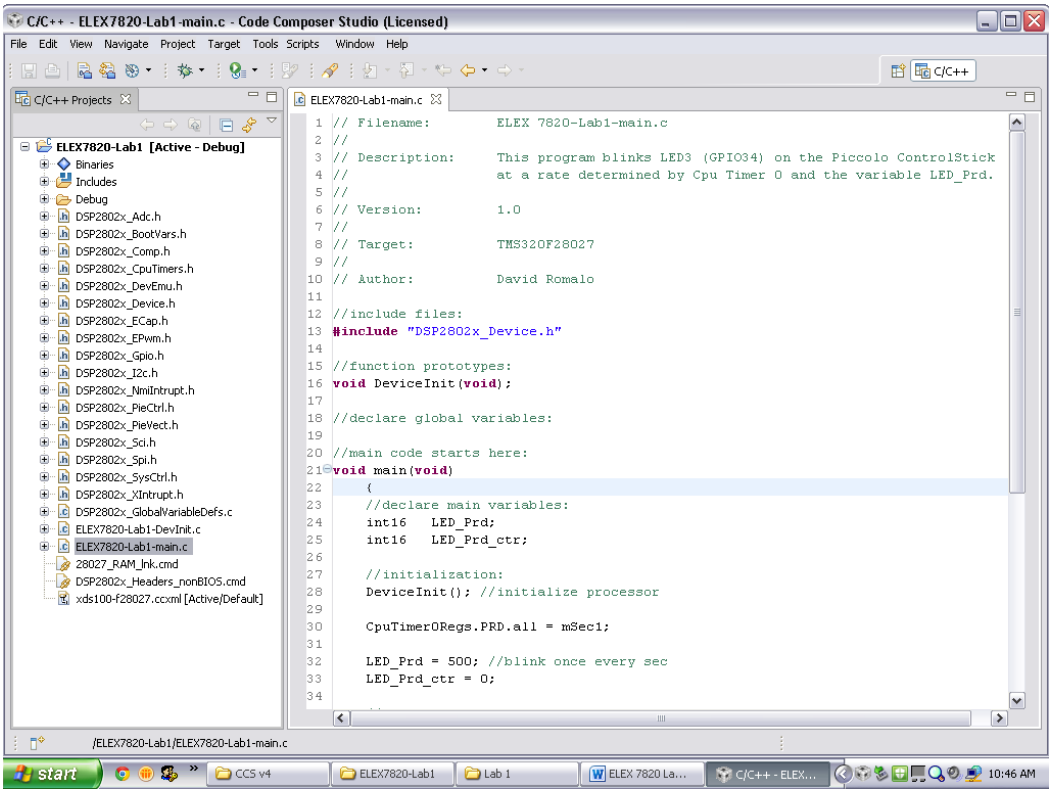
These files comprise all the files needed to develop a program on the c2000 target (with the exception that there is no RTOS yet). For example, there is a main program file and many of the .h files contain structures to support use of the c2000's peripherals.

Build (i.e., Compile & Link) Project (right-click name of project to access menu)

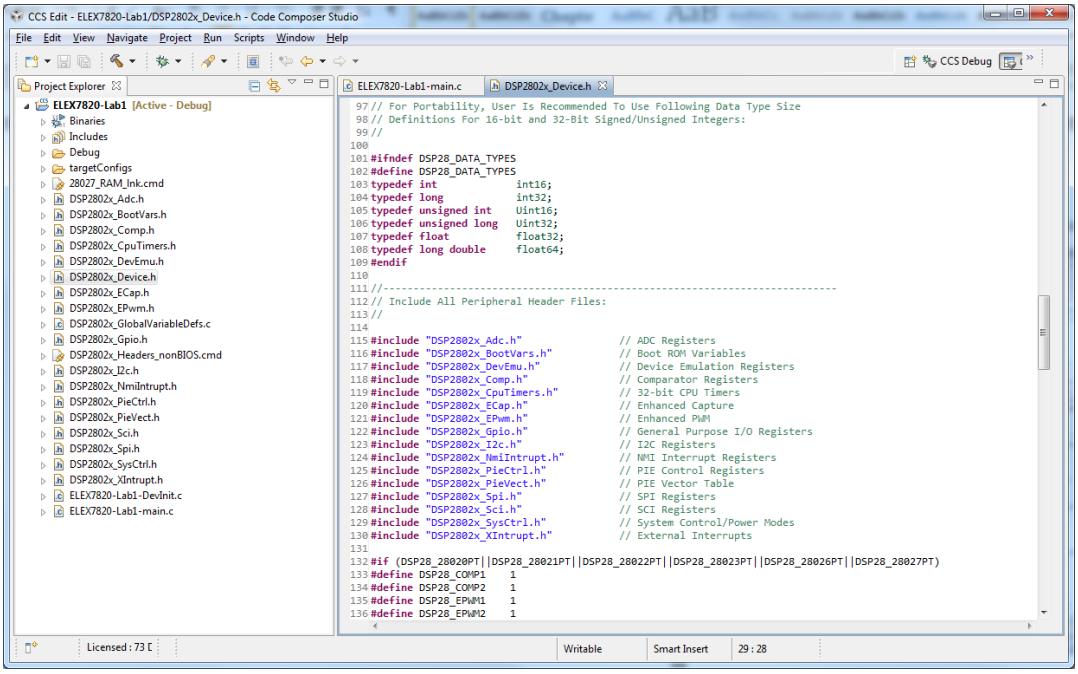




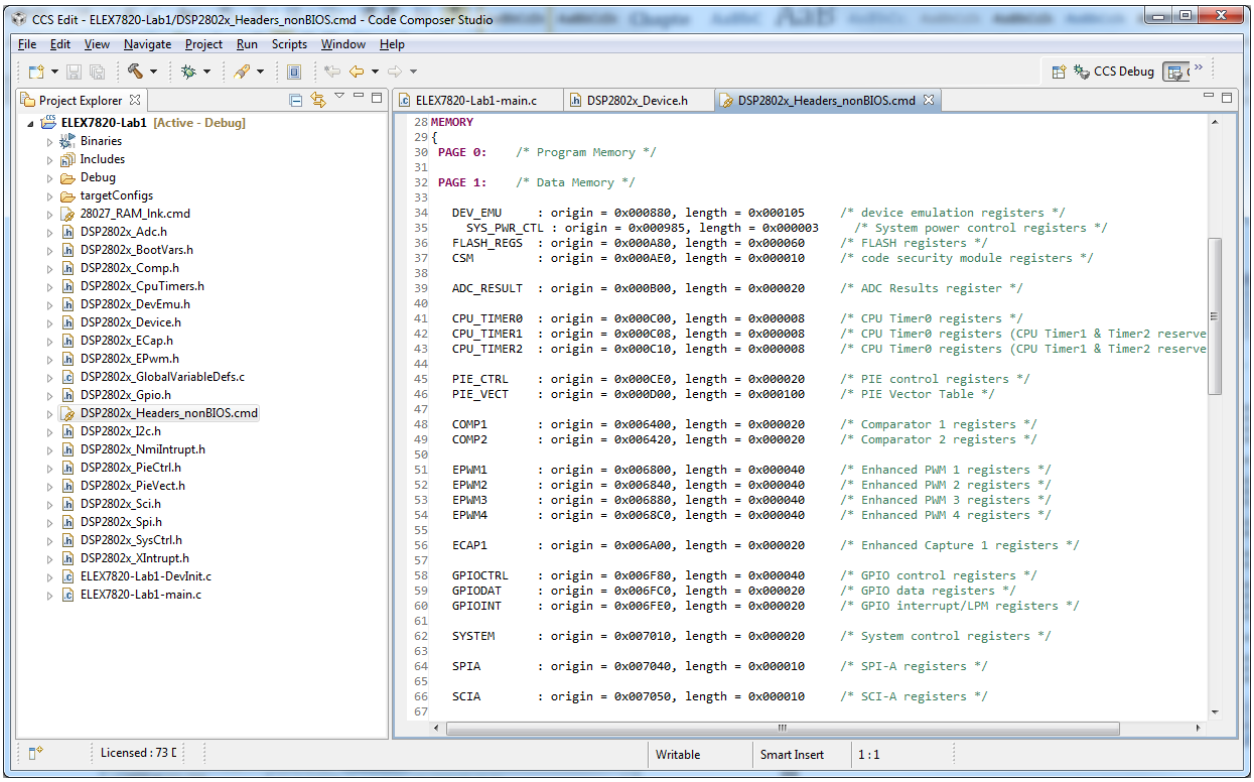
Look at main.c file (double-click on file name)



Look at DSP2802x_Device.h file



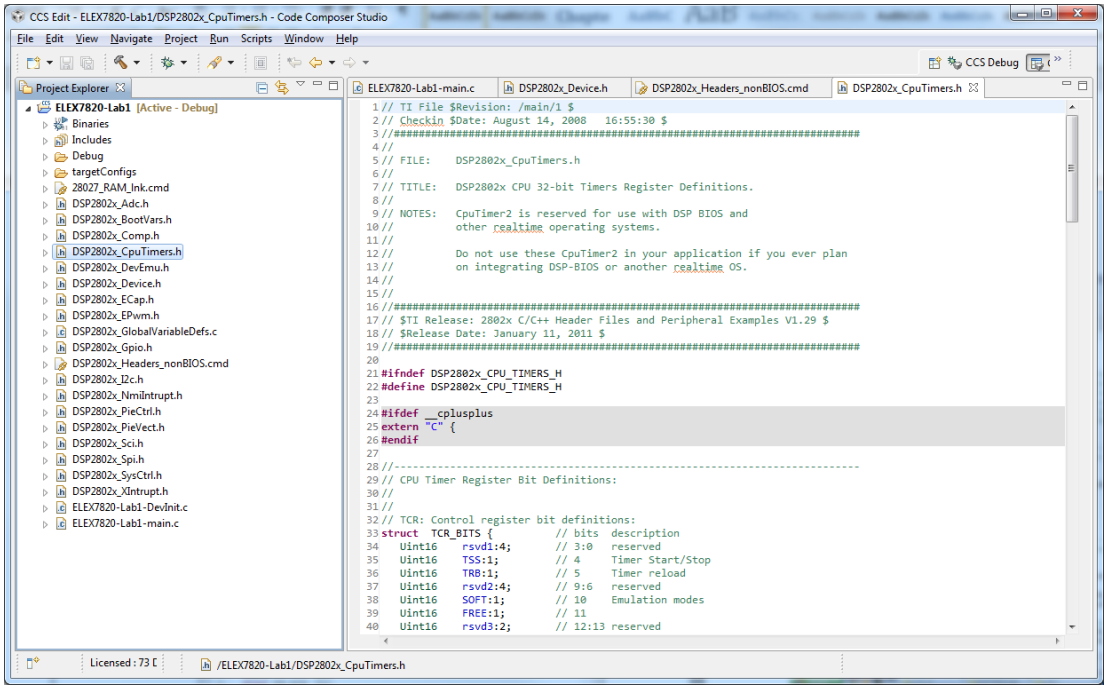
Look at DSP2802x_Headers_nonBIOS.cmd file



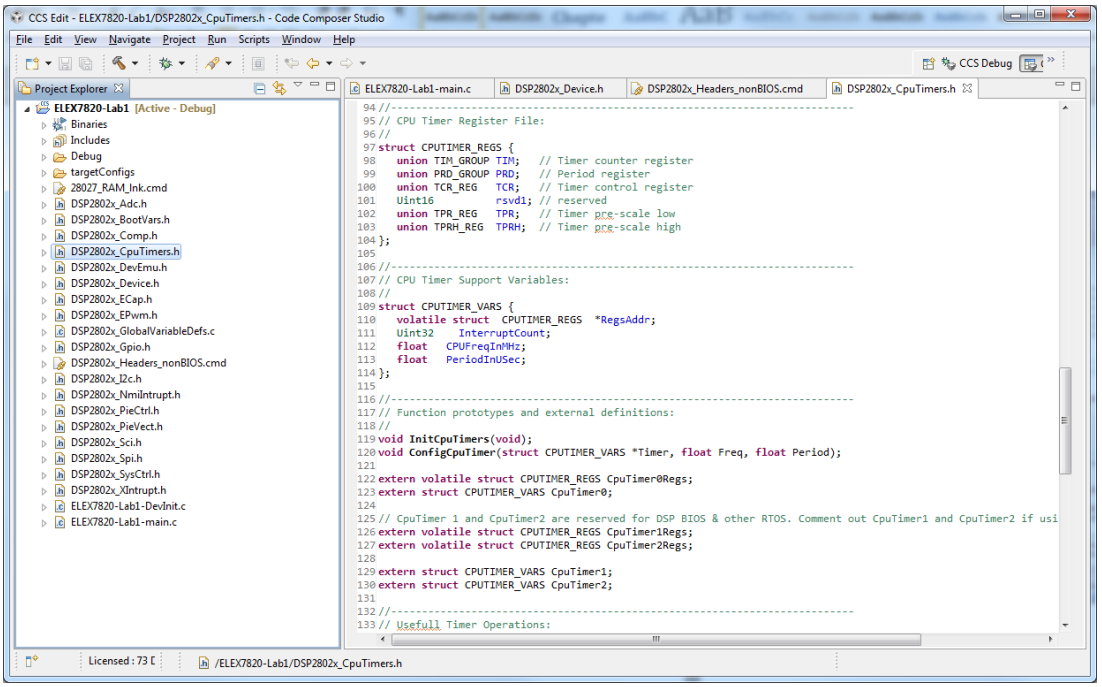
What is the basic function of the "DSP2802x_Headers_nonBIOS.cmd" file?

Note the Structure Hierarchy in the CPU Timer and GPIO Header Files

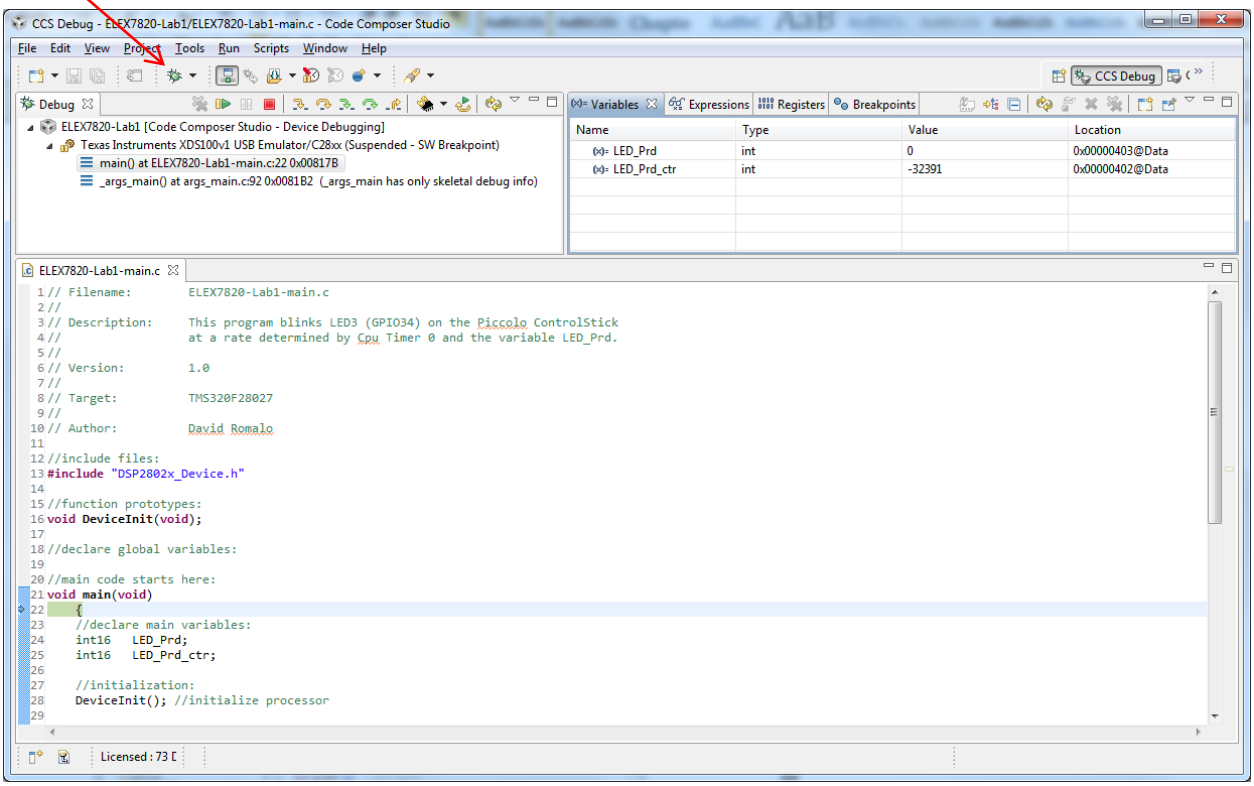
DSP2802x_CpuTimers.h file



Observe the structure of **`struct`**s and **`union`**s in the “DSP2802x_CpuTimers.h” file.



Launch the Debugger…

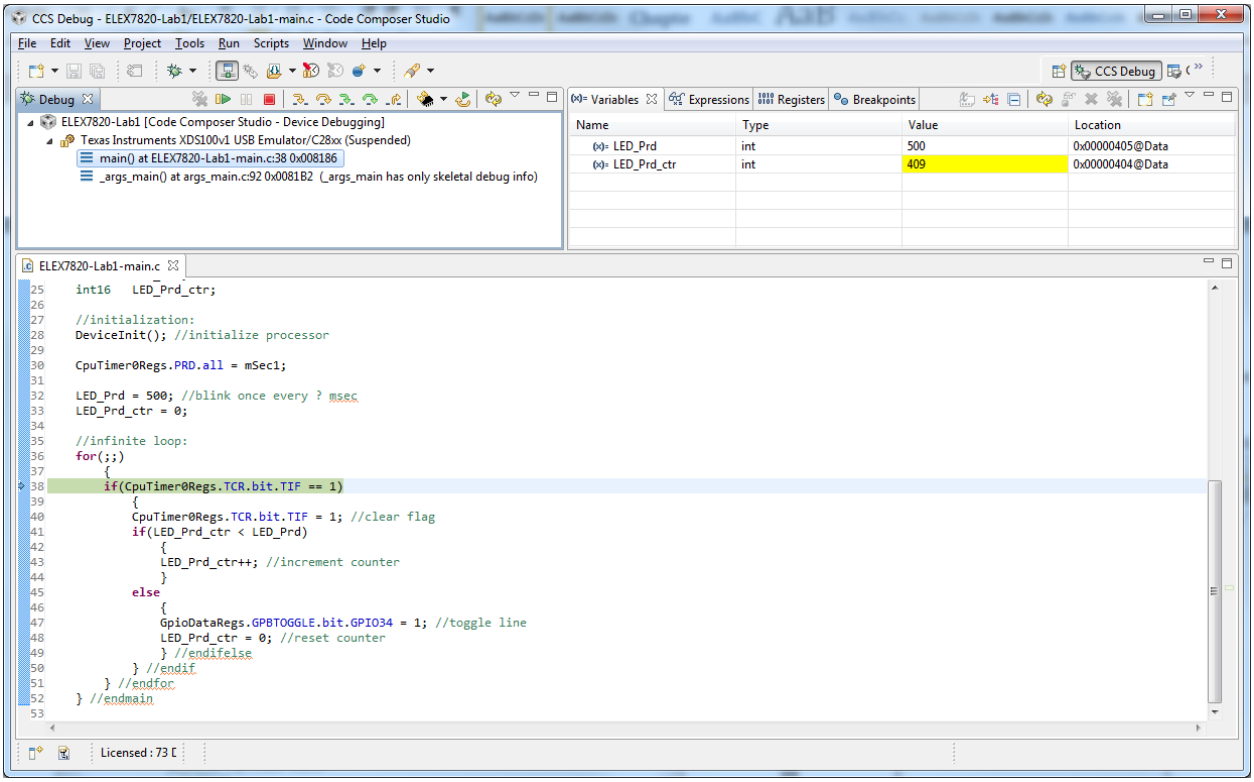


The Program Counter has been set to point to the beginning of main.

Start the program executing with the Resume button. A blue LED on the LaunchPad should flash continuously.

At what rate does the LED turn on and off? What is the duty cycle?

Does setting GPIO0 to a logical "1" output cause the LED to turn on or off? Why?

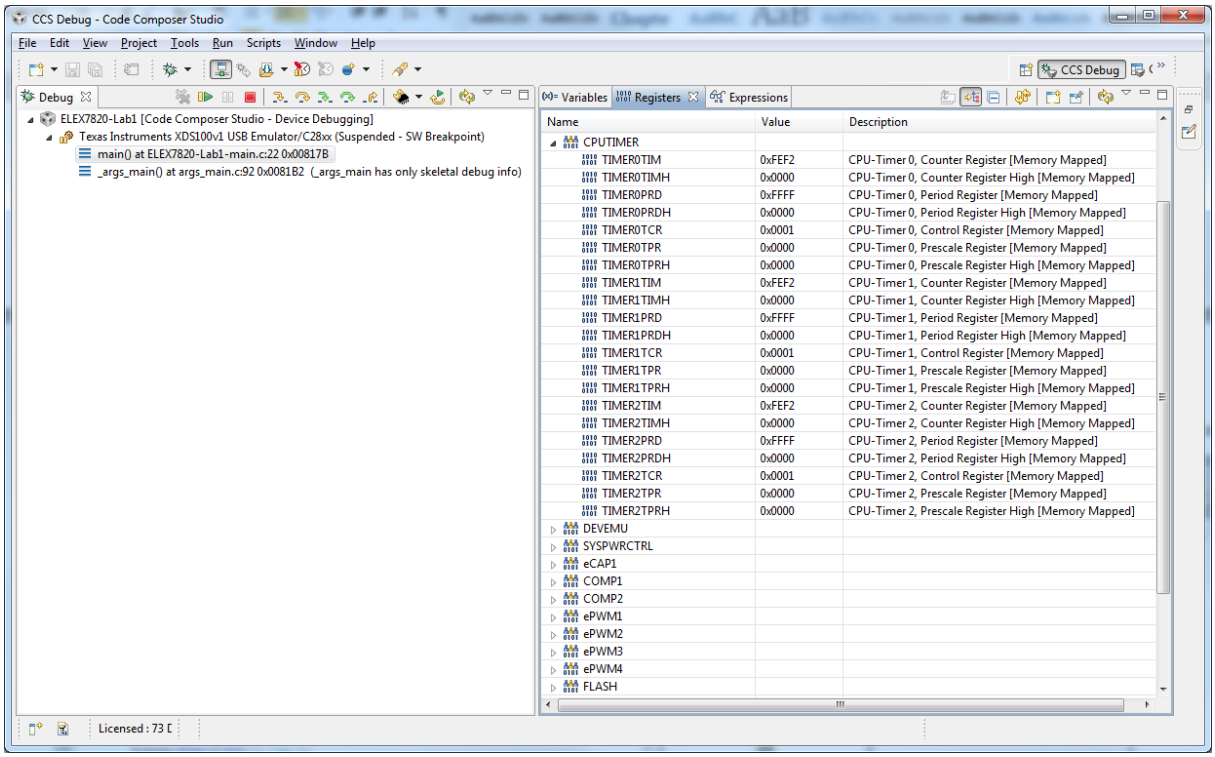


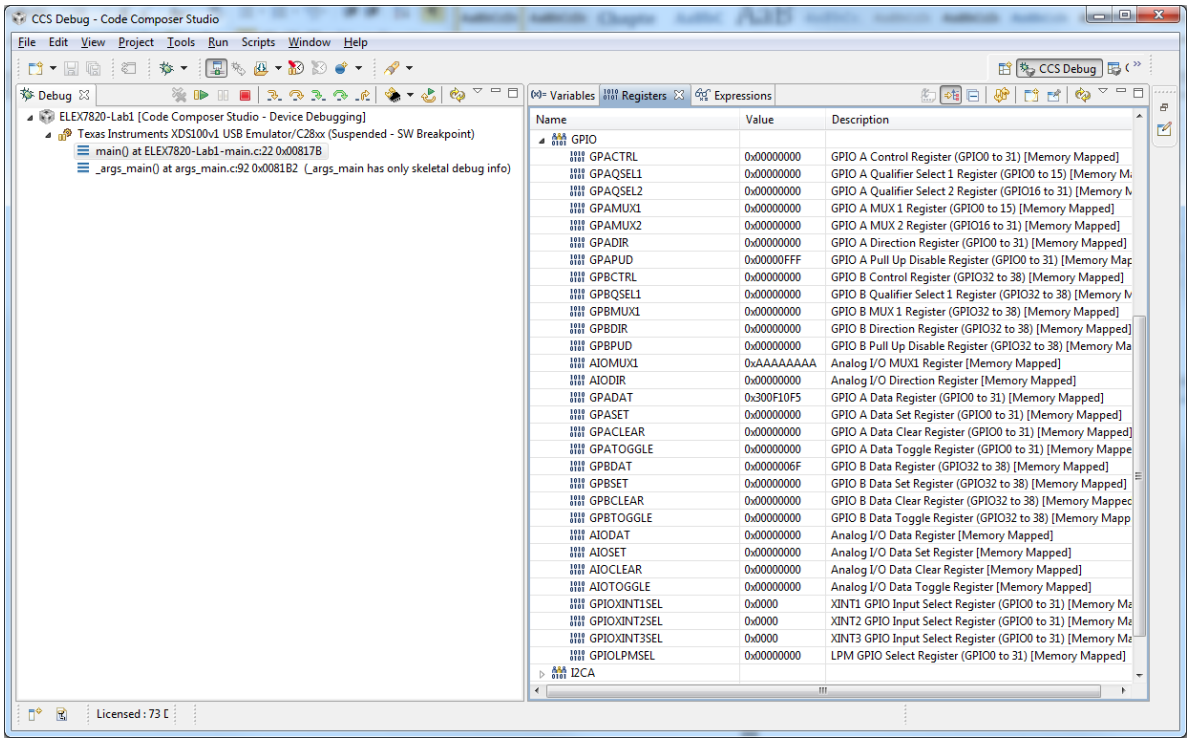
Try modifying the values of variables by editing them in the Variables window.

Example:

1. Suspend
2. Modify value of “LED_Prd”
3. Resume

CPU Timer and GPIO Registers in Registers Window





Understand how the timer works by perusing "Operation of CpuTimer0.pdf" and "sprufn3c excerpt pp64-68 - CPU Timers.pdf".

Study "Peripheral Structure Hierarchy - e.g. CpuTimer0.pdf" and follow (trace through) the hierarchy of the **`struct`**s and **`union`**s that define the registers of Timer 0.

Modify the main program by adding code to do the following (retain the flashing LED code):

1. Configure a GPIO pin of your choice (other than the LED one) as an output and initialize it to 0. (Choose one that is accessible on the LaunchPad board headers.)
2. Write C code for an N-tap FIR filter[1] that processes non-zero[2] dummy data once every LED period.
3. Set your GPIO pin before the FIR filter code and clear it afterwards.
   When you type in the struct/union name of the new GPIO line, use the "look-ahead" context feature.
4. Observe your GPIO pin on a scope to determine a rough upper bound on the processing time for an N-tap FIR filter and hence a lower bound on the corresponding sampling rate for three different values of N = 10, 50, 100. (It may be handy to trigger on the (slower) LED waveform.)
5. Take a scope snapshot of the pulse for N = 50 for both array and pointer code.

How many core clock cycles are needed to do N taps? Is this what you expected? Explain.

Submit to D2L a zip file named "LastnameFirstnameELEX7820Lab1.zip" that contains:

- A Word doc that includes your observations and answers to any questions that are posed in this handout or on the whiteboard, plus scope snapshot(s). (In your document that you submit, repeat the question above each answer and list them in the same order as presented in this handout.)
- Your C code file(s) for any code that you modified (*do not submit files that you did not modify*).

[1] Just do an N-point dot product, no need to do circular buffering of the data.
[2] Your dummy data need to be non-zero: If the data are zero, the compiler will optimize out the calculations and you will not measure a valid processing time.