Preparatory Lab

J.Shankarappa

Agenda

- Install Code Composer Studio
- Install TivaWare™ for C Series
- Create a New CCS Project
- Add file(s) to Project
- Review the CCS GUI
- Add the INCLUDE search path for the header files
- Examine Project Explorer
- Build project and fix any errors
- Debug Configuration
- Load, Debug, Run

CCS v8 - Hardware Requirements

	Minimum	Recommended
Memory	2 GB	6 GB
Disk Space	900MB	2GB average (1 or 2 device families) 3.5GB all features
Processor	1.0GHz x86 compatible processor	Dual core x86 compatible processor

- Disk space listed depends on features selected during installation. 900MB is for a bare minimal installation. Does not include temporary space required by the installer.
- Note that the most important requirement is memory. At least 4GB of memory or more is highly recommended.

CCS v8 - Operating System Requirements

- **Windows**:
  - Windows 7 (SP1 or later), Windows 8.x and Windows 10
- **Ubuntu 18.04 LTS**:
  - update system: *sudo apt-get update*
  - Install dependent libraries:
  - *$ sudo apt-get install libc6-i386 libusb-0.1-4 libgconf-2-4 build-essential*

Types of Installers

- There are two types of installers:
  1. Web installers
  2. Off-line installers
- Web installers will allow you to perform an install using an installer controlled download process that will only download needed software components.
  - An internet connection is mandatory at install time
- Off-line installers are a large archive (~ 800 MB). When you run it you can select the components to be installed.
  - No internet connection is required at install time
- The executable can be used for installing on multiple local systems
- **Note**: Get the "**EmSys_Tools**" CD from the Lab Instructor and copy to "~/**Downloads/"** folder.

Installing CCS - Windows

- De-activate your antivirus software
- **If Your Windows is 32-Bit**
  - Unzip CCS8.3.0.00009_win32.*zip* file
  - Navigate to *CCS8.3.0.00009_win32* folder
  - Run *ccs_setup_8.3.0.00009_win32.exe* file
- **If Your Windows is 64-Bit**
  - Unzip CCS9.2.0.00013_win64.*zip* file
  - Navigate to *CCS9.2.0.00013_win64* folder
  - Run *ccs_setup_9.2.0.00013.exe* file
- Accept the Software License Agreement and click Next.
- Unless you have a specific reason to install CCS in another location, we recommend you install into **drive:\ti\ccs902** folder and click Next.
- In the next dialog, select the processors that your CCS installation will support. Place a checkmark under "**TM4C12x ARM Cortex M4F core based MCUs**" and click Next
- In the Select Debug Probes, leave everything as default and click Next
  - Wait for the Installation to complete!

Installing CCS – Ubuntu 18.04 LTS

- **update system**

  sudo apt-get update

- **Install dependent libraries**:

  sudo apt-get install libc6-i386 libusb-0.1-4 libgconf-2-4 build-essential

- **Open terminal window** either via Ctrl+Alt+T keyboard shortcut or by searching for “terminal” from software launcher
  - cd ~/Downloads/EmSys_Tools
- **Untar ~/Downloads/EmSys_Tools/Linux64/CCS9.2.0.00013_linux-x64.tar.gz** file
  - *tar xvf* ~/Downloads/EmSys_Tools/Linux64/CCS9.2.0.00013_linux-x64.tar.gz
  - cd *CCS9.2.0.00013_linux-x64*
- **Run ccs_setup_9.2.0.00013*.bin***
  - *./ccs_setup_9.2.0.00013.bin*
  - Accept the Software License Agreement and click Next.
  - Unless you have a specific reason to install CCS in another location, we recommend you install into ~/***ti/ccs920*** *folder* and click Next.
  - In the next dialog, select the processors that your CCS installation will support. Place a checkmark under "**TM4C12x ARM Cortex M4F core based MCUs**" and click Next
  - In the Select Debug Probes, leave everything as default and click Next
  - Wait for the Installation to complete!

Install Drivers - Ubuntu

- Once CCS installation has completed, navigate to directory:
  - *<Install-Folder>/ccs920/ccs/install_scripts*
- As root, execute script "install_drivers.sh".
  - sudo ./install_drivers.sh

Installing TivaWare for C Series

- Run *SW-TM4C-2.1.4.178.exe* file (It's self-extracting)
- Please install TivaWare in the following folder
  - cd ti
  - mkdir *TivaWare_C_Series-2.1.4.178*
  - *~/ti/TivaWare_C_Series-2.1.4.178* ***(Ubuntu)***
  - *drive:/ti/TivaWare_C_Series-2.1.4.178* **(Windows)**

  - **Reboot** your System

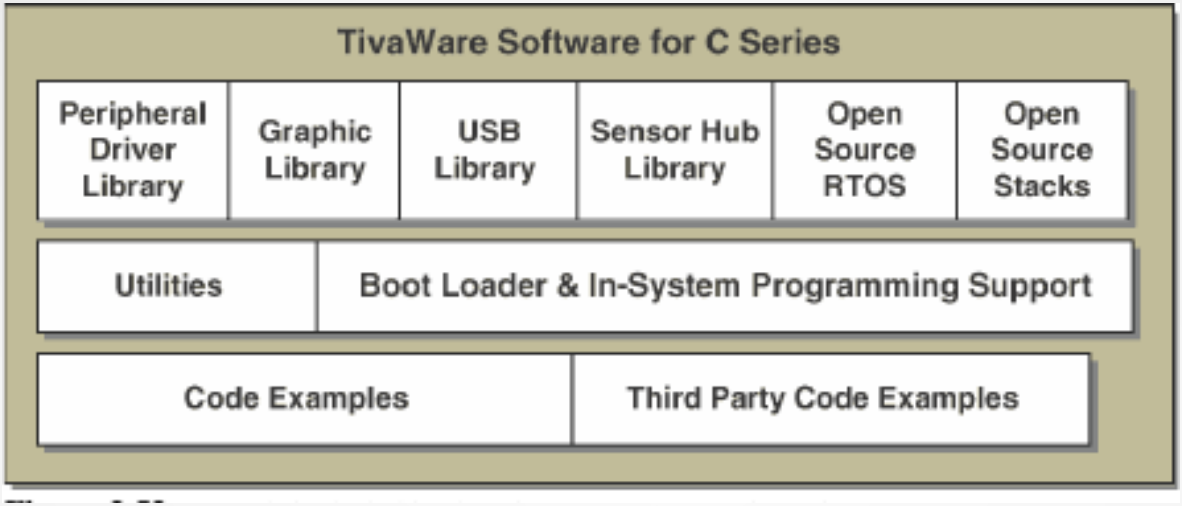
Getting familiarize with CCS

- In this lab, we'll create a project that contains two source files, *main.c*, which contain the code to blink all the LEDs (RGB) on the LaunchPad board and *tm4c123gh6pm_startup_gcc.c file which is generated by CCS*.
- The purpose of this lab is to practice creating projects and getting to know the look and feel of Code Composer Studio.
- In later labs we'll examine the code in more detail.
- So for now, don't worry about the C code we'll be using in this lab.

Programming Models

- Using Tivaware – Library provided by the TI to access Peripherals
- Cortex Microcontroller Software Interface Standard (CMSIS).
- Using Direct Register Addressing (DRA)

Tivaware Library

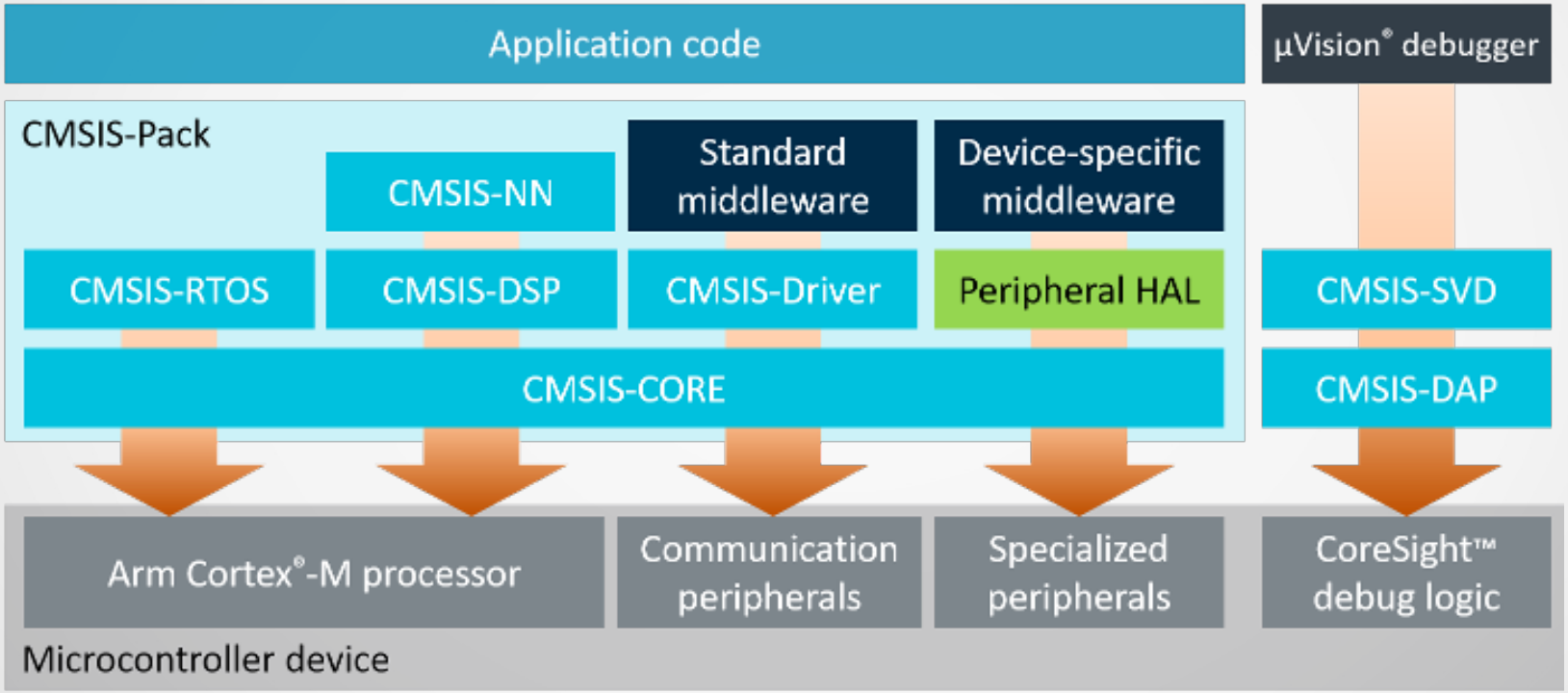


- **Peripheral Driver Library (DriverLib)**
- Graphic Library
- USB Library
- Sensor Hub Library
- Open Source RTOS
- Open Source Stacks
- Utilities
- Boot Loader and In-System Programming Support
- Example Codes
- Third-Party Examples

CMSIS

- CMSIS is a vendor-independent hardware abstraction layer for the Cortex-M processor series and defines generic tool interfaces.
- The CMSIS has generic software libraries and simple software interfaces to the processor and the peripherals.
- CMSIS provides a compiler independent layer and can be compiled by all mainstream compilers (ARMCC, IAR, GCC and TI‟s CCS).
- The CMSIS is defined in close cooperation with various silicon and software vendors and provides a common approach to interface to peripherals, RTOS systems, and middleware components.
- The CMSIS is intended to enable the combination of software components from multiple middleware vendors.

CMSIS ...

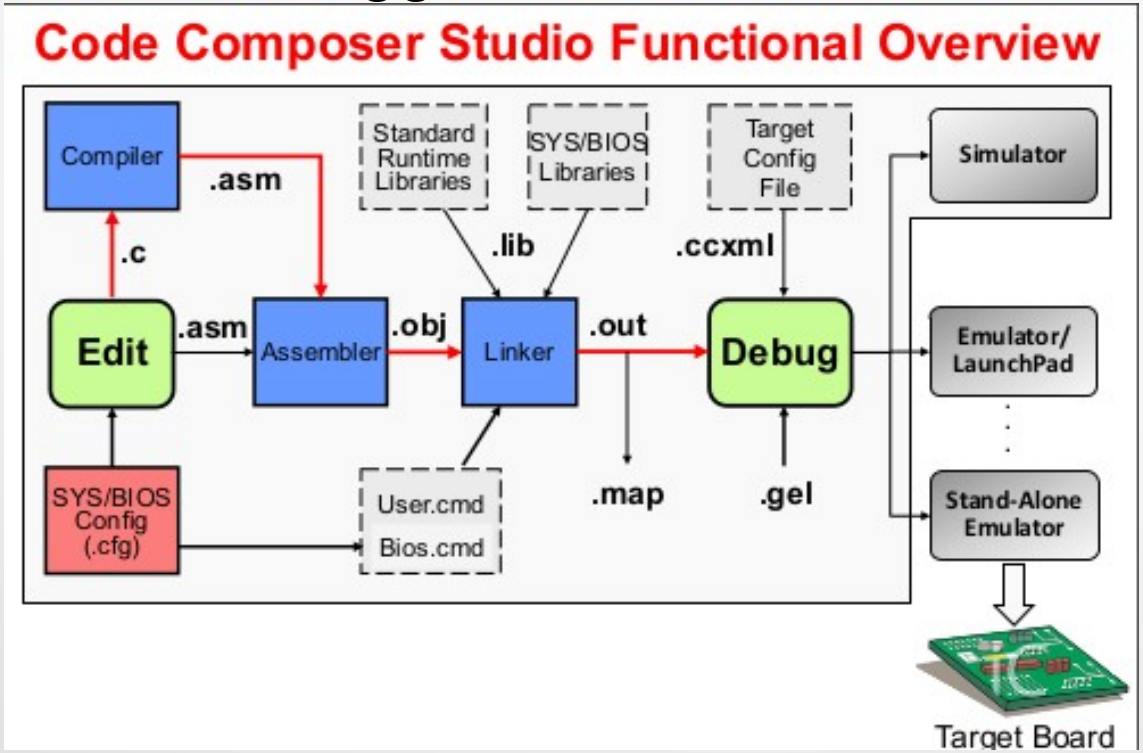

Application code
µVision® debugger
CMSIS-Pack
CMSIS-NN
Standard middleware
Device-specific middleware
CMSIS-RTOS
CMSIS-DSP
CMSIS-Driver
Peripheral HAL
CMSIS-SVD
CMSIS-CORE
CMSIS-DAP
Arm Cortex®-M processor
Communication peripherals
Specialized peripherals
CoreSight™ debug logic
Microcontroller device

Direct Register Addressing

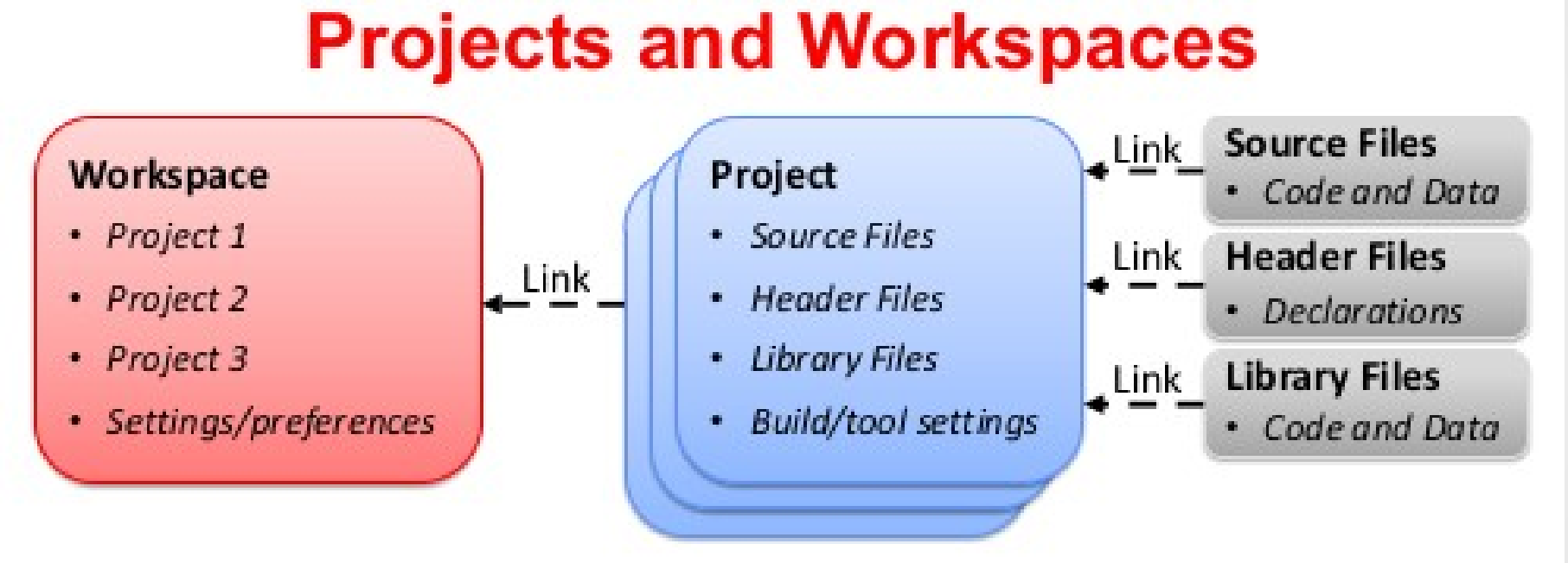
- **Access the Peripheral's Special Function Registers Directly using C Language**.
- Programmers need to use a set of registers/macros or register symbolic definitions provided by the header files to simplify their coding process.
- These register symbolic definitions are stored in MCU-Specific header files contained in the **inc** directory.
- The header file for the **TM4C123GH6PM** MCU is in the *inc/**tm4c123gh6pm.h*** directory under the /ti/TivaWare_C_Series-2.1.4.178 folder.
- Target programs can be developed in smaller size with higher efficiency.
- Developers need to get very detailed knowledge about the processor and peripheral hardware architecture and interfacing configurations/parameters.
- By using the Tivaware, programmers do not need to know too much about the peripheral hardware architecture.

CCS Functional Overview

- Integrated Development Environment (IDE) based on Eclipse
- Contains all development tools – compiler, assembler, linker and debugger.



Workspaces & Projects



Workspaces, Projects ...

- **Workspace**
  - When the Workbench is launched, the first thing you see is a dialog that allows you to select where the workspace should be located.
  - workspace is the directory where your work will be stored.
  - Projects can reside in the workspace folder or be linked from elsewhere
  - IDE settings and Prefernces
  - When importing projects into the workspce, linking is recommended.
  - Deleting a project within the Project Explorer only deletes the link
- **Project**
  - Build and tool settings (used in managed Make projects)
  - Files can be linked to or reside in the project folder
  - Deleting a linked file with in the project explorer only deletes the link

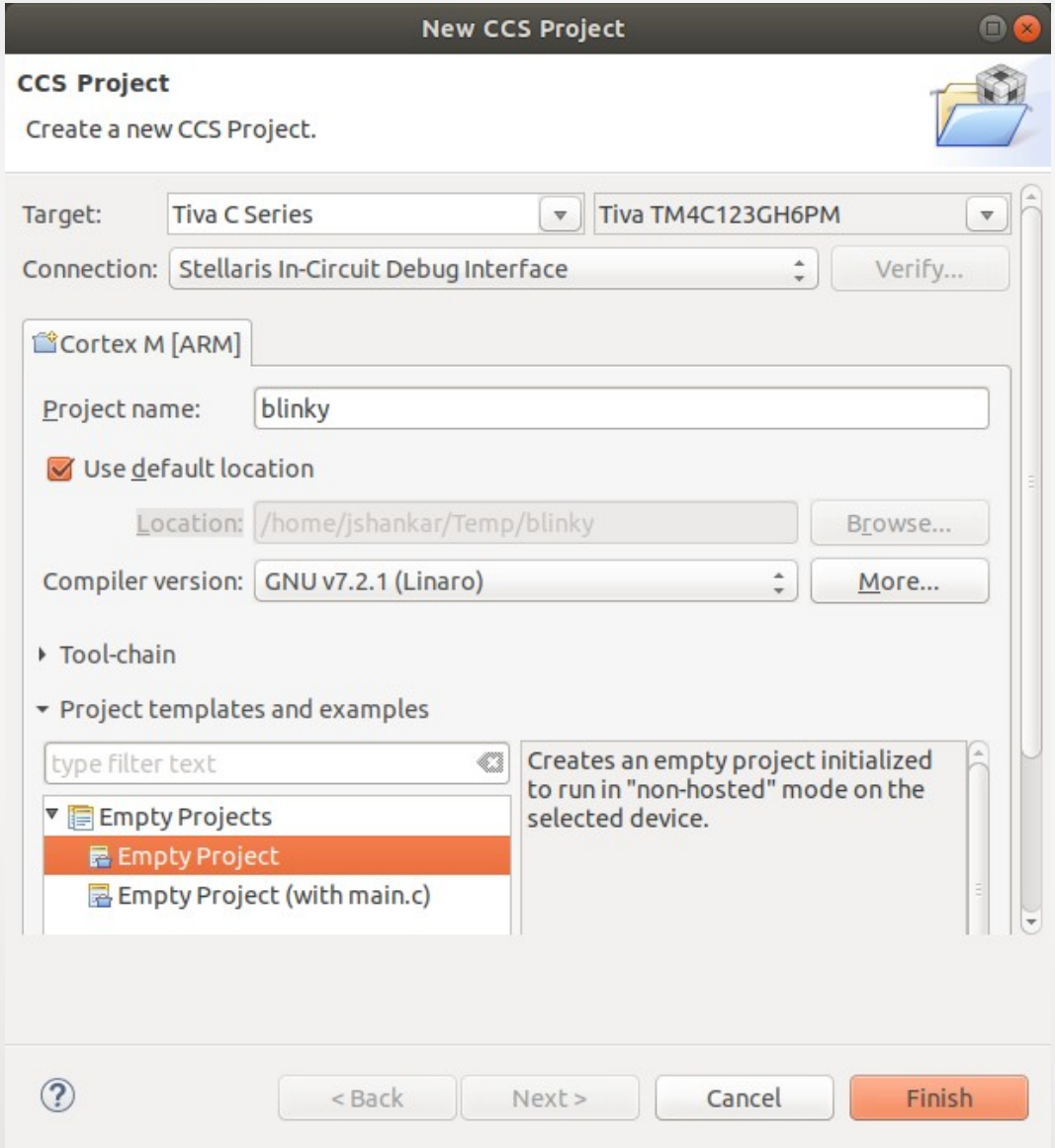
Perspectives

- A perspective defines the initial set and layout of views in the Workbench window.
- A perspective contains editors and views, such as the Project Explorer.
- Perspectives control what appears in certain menus and toolbars.
- A Workbench window offers one or more perspectives.
- A shortcut bar appears in the top right corner of the window. This allows you to open new perspectives and switch between ones already open.
- The name of the active perspective is shown in the title of the window and its item in the shortcut bar is highlighted.

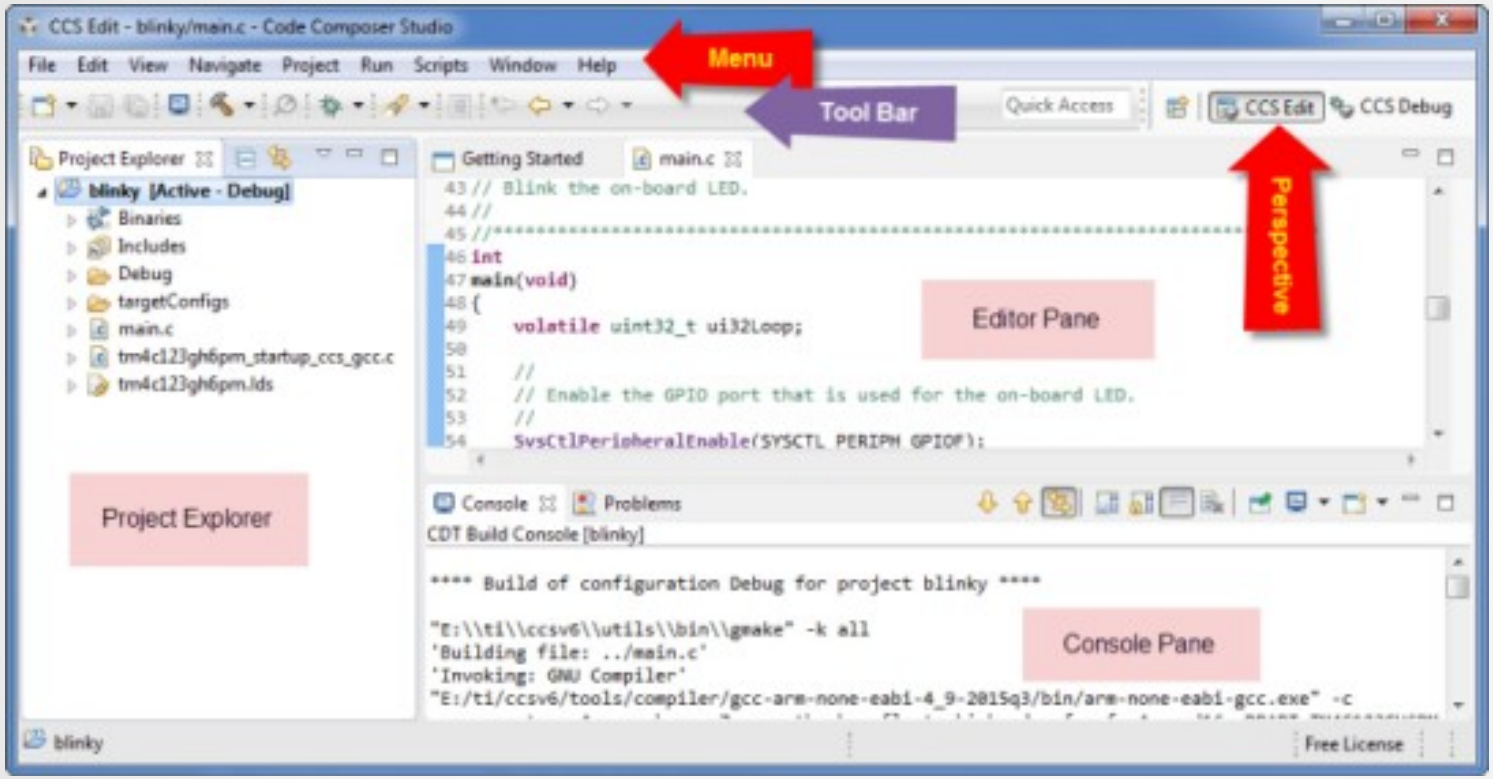
Creating New Project

- Launch CCS
- **File** → **New** → **CCS Project**
- **Project Location**: Donot Check the box “Use default location”
- **Target**: Tiva C Series, **TM4C123GH6PM**
- For **Connection**: choose “Stellaris In-Circuit Debug Interface”. This is the built-in emulator on the LaunchPad board.
- **Project Name**: blinky
- **Compiler Version**: GNU v7.2.1(linaro)
- In the Project templates and examples box, choose Empty Project
- Click Finish.

CCS New Project



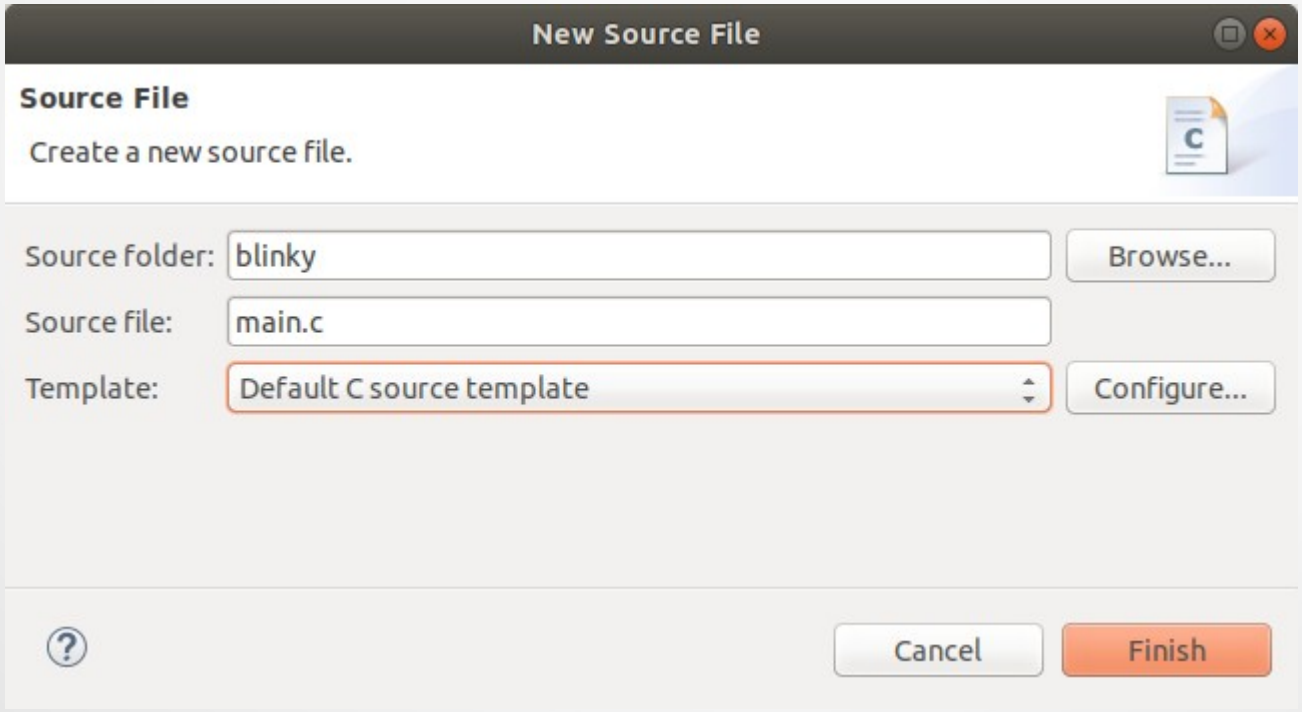
Explore CCS GUI



New Project wizard added a startup file called *tm4c123gh6pm_startup_gcc.c* into the project automatically.

Add New Source File

- We need to add main.c to the project.
- **File → New → Source File**

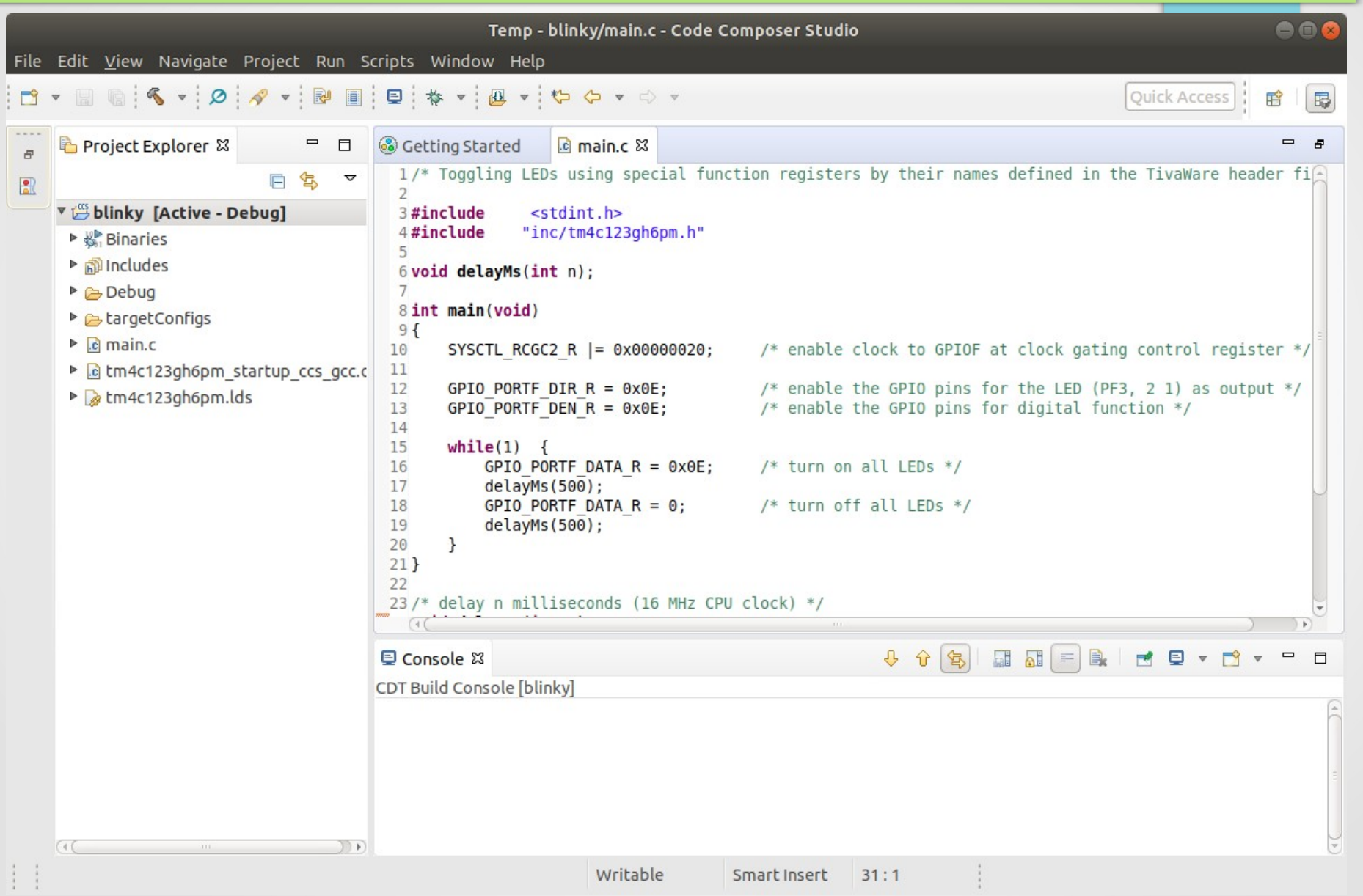


Blinky Program

Copy the following code and Paste it in the main.c file (over-write)

```
/* Toggling LEDs using special function registers by their names defined in the TivaWare header file */

#include     <stdint.h>
#include    "inc/tm4c123gh6pm.h"

void delayMs(int n);

int main(void)
{
    SYSCTL_RCGC2_R |= 0x00000020;     /* enable clock to GPIOF at clock gating control register */

    GPIO_PORTF_DIR_R = 0x0E;          /* enable the GPIO pins for the LED (PF3, 2 1) as output */
    GPIO_PORTF_DEN_R = 0x0E;          /* enable the GPIO pins for digital function */

    while(1)  {
        GPIO_PORTF_DATA_R = 0x0E;     /* turn on all LEDs */
        delayMs(500);
        GPIO_PORTF_DATA_R = 0;        /* turn off all LEDs */
        delayMs(500);
    }
}

/* delay n milliseconds (16 MHz CPU clock) */
void delayMs(int n)
{
    int i, j;

    for(i = 0 ; i < n; i++)
        for(j = 0; j < 3180; j++) {}          /* do nothing for 1 ms */
}
```

Blinky Project Window



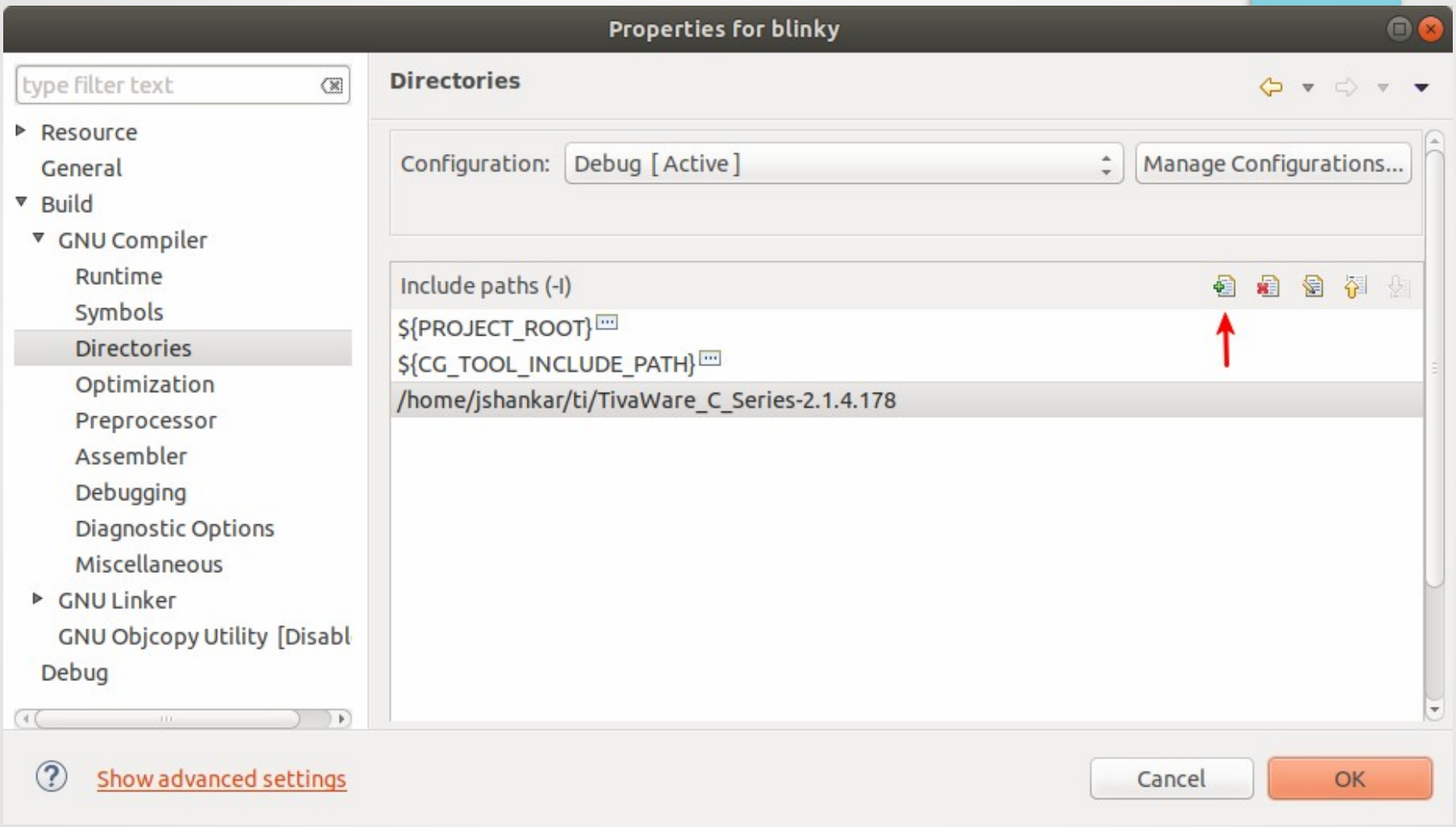
CCS - Build Configuration

- Code Composer has two pre-defined Build Configurations:
- **Debug** (symbols, no optimization) – great for debugging
- **Release** (no symbols, optimization) – great for performance/code size
- Users can create their own custom build configurations

Add the INCLUDE search path

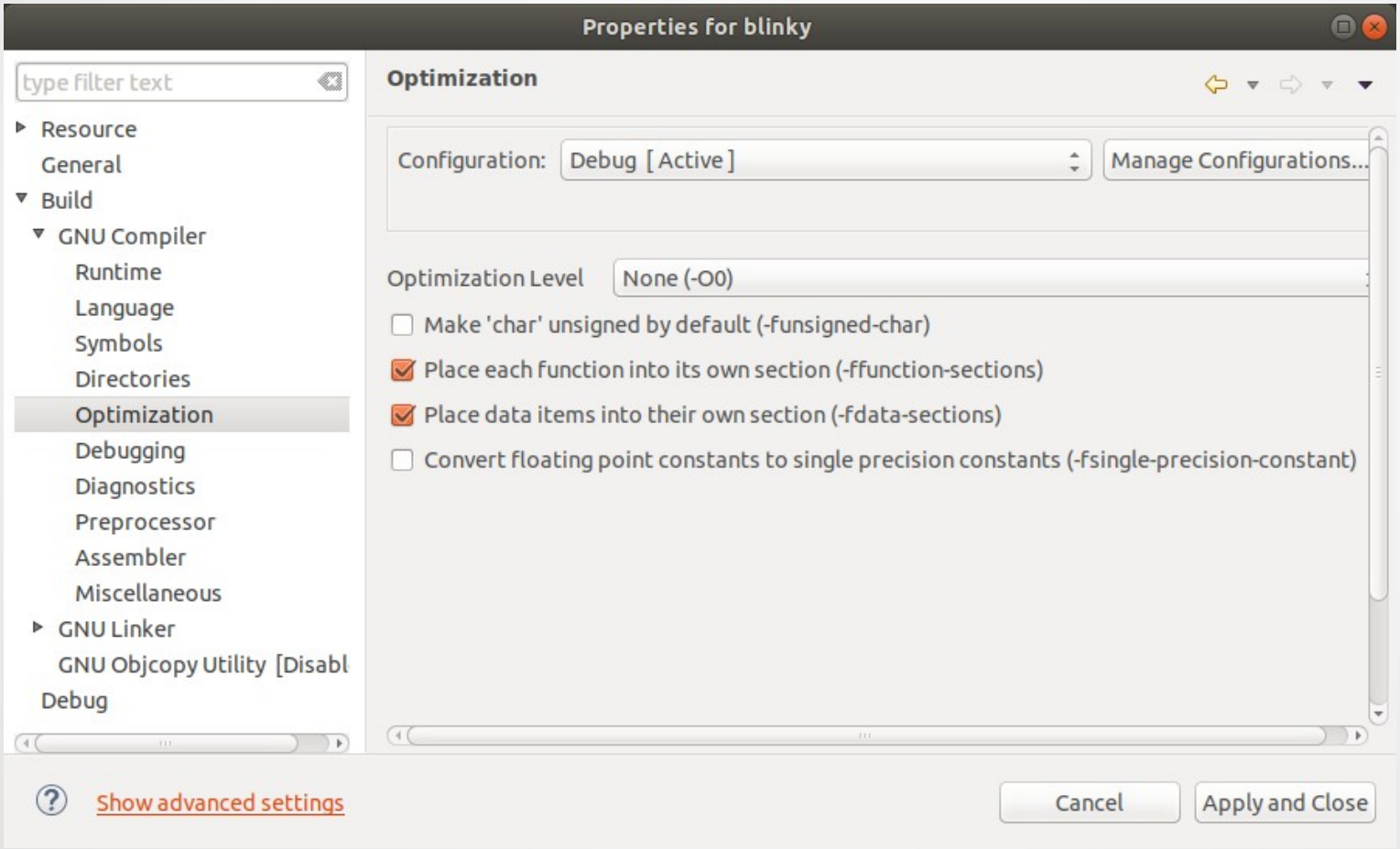
- Right-click on the Project and select Properties
- Then click “Directories” category: In the “Includes Paths” Tab, Press **+**, Browse to Tivaware installation location and Select “${HOME}/*ti/TivaWare_C_Series-2.1.4.178*” folder.
- Click OK

Add the INCLUDE search path



Set Optimization Level - None

Project → Properties
Build → GNU Compiler → Optimization

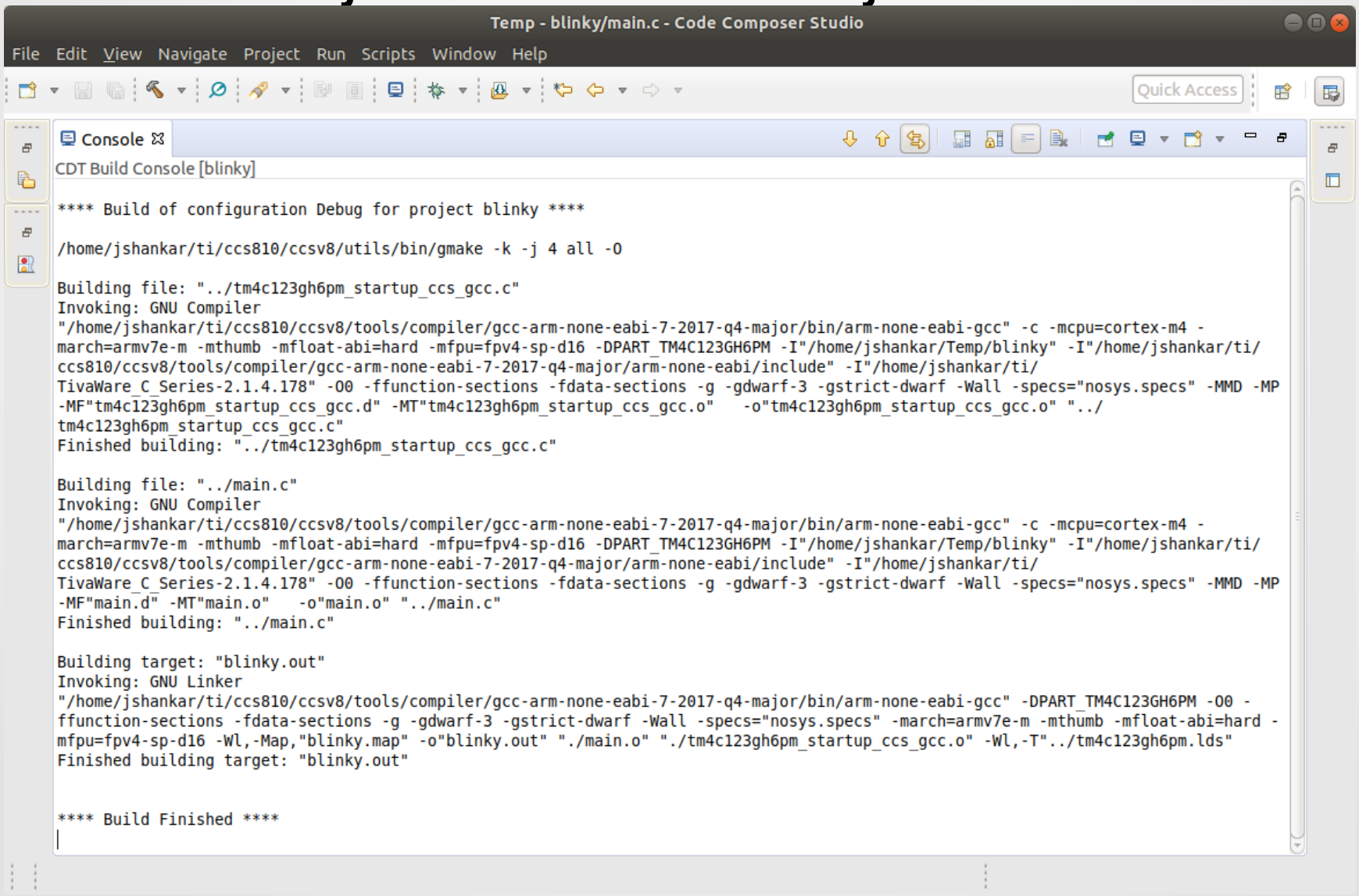


Examine the properties of Project

- In CCS, right-click on the project and select Properties
- Click on each of the sections below:
- **Resource:** This will show you the path of your current project and the resolved path if it is linked into the workspace. Click on “Linked Resources” and both tabs associated with this.
- **General:** shows the main project settings. Notice you can change almost every field here, after the project was created.
- **Build → GNU Compiler:** These are the basic compiler settings along with every compiler setting for your project.
- **Other:** feel free to click on a few more settings, but don’t change any of them.
- Click Cancel.

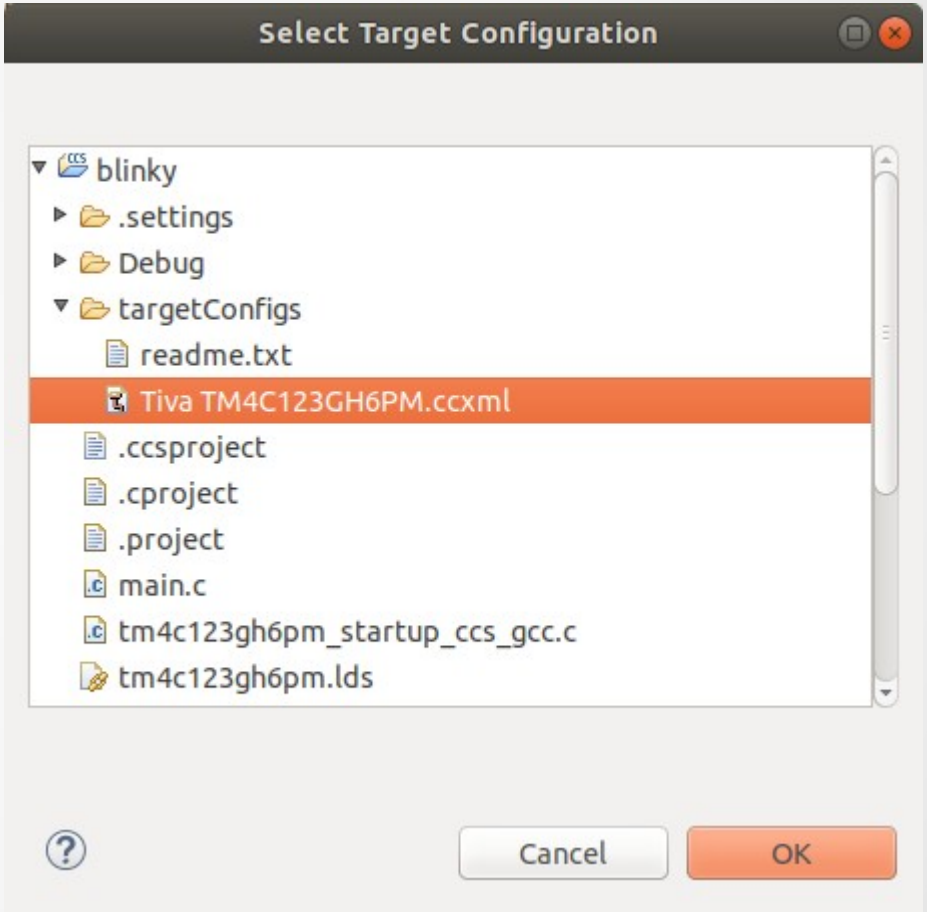
Build Project

Select Project → Build Project



CCS - Debug Configuration

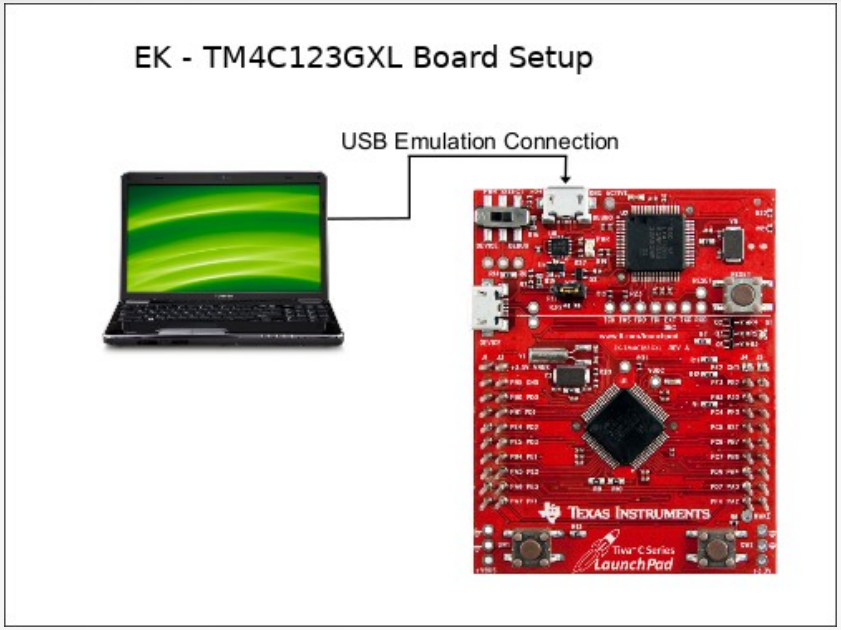
- Select **Run → Debug Configurations...**
- In the next Dialog Box, Double click on **Code Composer Studio - Device Debugging** in the left pane
- **Target configuration** – Browse workspace, select “Tiva TM4C123GH6PM.ccxml file”



CCS - Debug Configuration ...

- Select **Program** tab
- Click on the Workspace button select the project you want to configure.
- Select **Target** tab
- Select “Reset the target on connect” and “Reset the target on a program load or restart”
- Select Common tab
- Check **Debug** in the “Display in favorites menu”
- Click on the Apply button followed by Close button

Board Setup



- Switch the POWER SELECT switch to the right for Debug mode.
- Connect the USB-A cable plug to an available port on the PC and the Micro-B plug to the Debug USB port on the board.
- Verify that the POWER LED D4 on the board is lit.
- More Info:
  - http://shukra.dese.iisc.ernet.in/edwiki/EmSys:EK-TM4C123GXL_LaunchPad_Initial_Board_Setup

Build, Load, Run

- Assure that your LaunchPad is connected to your laptop.
- To Build your Project, Select **Project** → **Build Project**
- Select the Debug Configured project from the Debug Tool Bar Pull-Down Menu 
- This will load the program to the TM4C123GH6PM flash memory and the program counter will run to ***main()*** and stops!
- If you ever want to build the project without loading it, click the HAMMER button.
- Click the Resume button or press the F8 key on your keyboard
- Our program should toggle 3 - RGB LEDs

Thank You

The End