Getting Starting with Code Composer Studio

James O. Barnes (james.barnes@colostate.edu)

January 20, 2014

Contents

Note: Red font indicates a click-able ink.

1 Introduction

In this class, we will be using Code Composer Studio version 5.5 (CCS) from Texas Instruments. There are tutorials – both video and document form – on running this tool (accessible from the help menu and explorer window) which you are encouraged to work through. You should know basics such as how to access the Edit and Debug environments and what the project explorer pane is. This document contains only information specific to how we will be using the tool in this class.

First an observation regarding the labs: there are no exams or homework assignments in this class. Instead, you should expect to spend about as much time in the laboratory performing the lab assignments and outside the laboratory writing reports as you would for a similarly-credited class preparing for tests and doing homework assignments. The best advice I can give you in managing this time wisely is to carefully read the lab writeup before coming to the lab and do as much pre-work as possible. This could even involve writing and compiling some of the code you will be running in the lab. To do this requires access to CCS outside of the lab; at present,

this requires it to be installed on your laptop. If you are interested in installing CCS, see the instructor.

2 Overview of the Development Environment

2.1 Definitions

Here are some definitions of terms we will be using in this class.

TMS320C6713 or DSK6713 A DSP development board containing chips from Texas Instruments. The board is designed and manufactured by Spectrum Digital, Inc.

C6713 The DSP chip from Texas Instruments which performs the DSP operations. The chip is optimized to perform functions extensively used in DSP programs, such as MAC (multiply-accumulate).

AIC23 The Codec (Coder/Decoder) chip on the DSK. Basically a two channel Analog-to-digital converter and Digital-to-analog converter.

Emulator A chip on the DSK board used in downloading the program. The "emulator" chip converts USB to whatever serial link is used for configuring the chips on the development board. This may be SPI or (in the case of the DSK6713) JTAG.

Eclipse A generic development environment which can be customized via plug-ins for code development for a wide range of languages and embedded processors. Eclipse-based environments are extensively used in industry and academia.

CCS Code Composer Studio, TI's customization of eclipse for TI embedded processors. Third party development boards, such as the one used in this class, are covered by separate libraries and plug-ins supplied by the vendors.

workspace A CCS directory owned and writable by you which will contain one or (typically) multiple projects. Workspaces can only be created by CCS, which will add hidden files containing CCS settings and information. There is essentially no limit to the number of workspaces that can be created.

project A directory within a CCS workspace containing a collection of files, including user code which can be compiled and downloaded run on the DSK. A project can have only one file with a `main()` function, which is the starting point for any c program. This will impact how you will organize your files, as discussed below.

cross-compilation compiling a program on one machine architecture (e.g. x86 machine running windows or linux) to be run on another architecture (e.g. C6713 DSP chip). The core CCS program contains libraries to link in standard C functions such as printf and math functions on the C6713. This happens more-or-less automatically.

chip support library A library which implements specialized functions on the C6713 such IRQ processing. For the C6713, the chip support library, supplied by Texas Instruments, is installed under `C:/ti/DSK6713/`.

board support library A library providing low-level board functions such as USB-to-JTAG translation, read-write access to control registers, and so forth. For the DSK, the board

support library, supplied by the manufacturer Spectrum Digital, Inc, is installed under `C:/ti/DSK6713/`.

Support files A small number of initialization and configuration C code files written specifically for the DSK. You will copy these files into your project directories and compile them with the code you write (more later).

2.2 CCS GUI

Code Composer Studio (CCS) is an advanced development environment which we will use to write C programs, compile them ("Build" in CCS terminology) and run them ("Debug" in CCS terminology).

Figure 1 shows the "Edit Perspective" in the CCS GUI, which is used to write and compile programs. The `Project Explorer` pane, shown on the left, is used to access projects within a single workspace. The open project "sine_gen_intr" is the first exercise in Lab1.

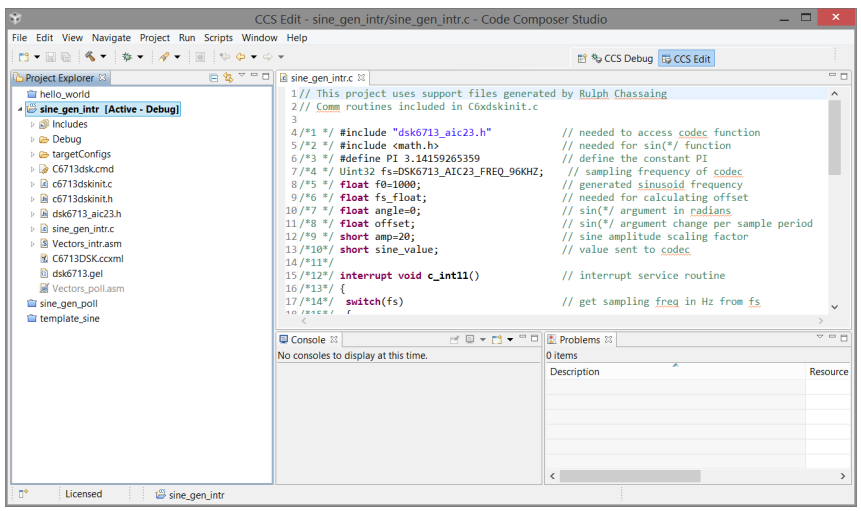

Figure 1: CCS Edit Environment

Figure 2 shows the "Debug Perspective" in the CCS GUI. In this environment, the program has been downloaded to the board and can be run, halted, single-stepped, and run to breakpoints. The various panes in the window show information on the state of the program, such as the current program execution point when halted or being single-stepped, both in the C program and the assembly code version, values of variables, and so forth programs.

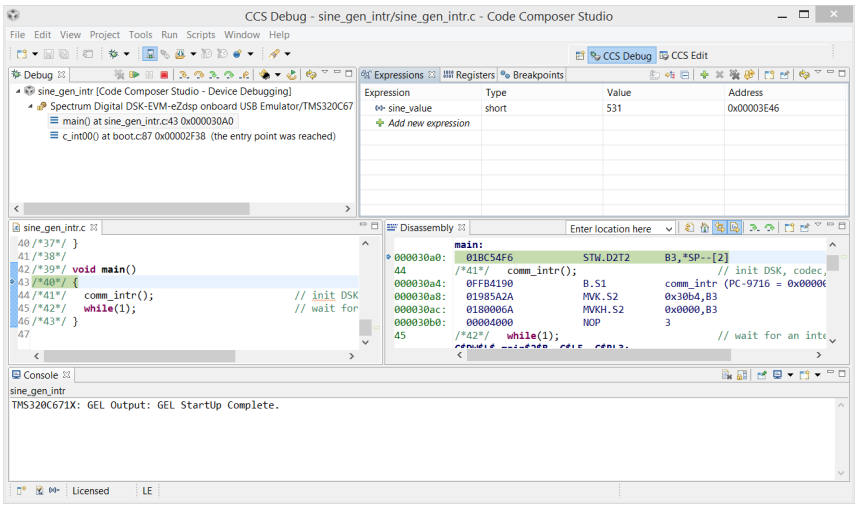

Figure 2: CCS Debug Environment

2.3 Organizing Your Files

This section concerns how the workspace(s) and projects are should be organized. Following this will simplify creating and compiling additional projects once you have created the first project.

CCS does not specify an explicit arrangement, but there are a couple of limitations in CCS which lead to the recommendation given here. The first limitation is that CCS does not allow a hierarchy of projects within a workspace. Secondly, as mentioned above, a project can have only one top-level c source file, i.e. the file contain the `main()` function. This leads to the following recommendation:

1. Create a new workspace for *each* Lab.
2. Make each required exercise for the lab a separate project within the workspace.
3. You may want additional directories to hold non-CCS files, such as MATLAB code, report drafts, data, and so forth. If they are project-specific, you can create them with the windows file explorer under each project, or you can create them at a higher level. CCS will ignore these additional directories and files.

Following these guidelines will lead to the file organization shown in Figure 3, where the directories shown in blue are separate workspaces:

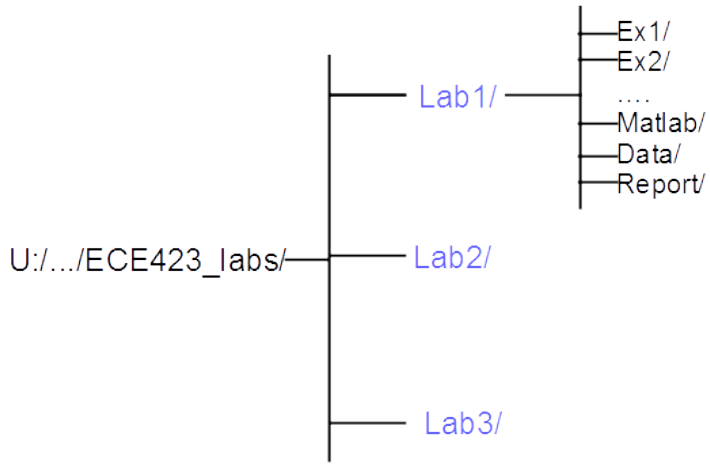

Figure 3: Recommended File Structure

3 Steps for Creating and Running Programs on the DSK

Note:

- CCS uses a lot of icons to launch commands. You can generally find out what the icon does by hovering the cursor.
- In this document `->` = "left click", `[RC]->` = "right click".
- I created this document using CCS installed on my laptop and the directory paths in screenshots below reflect that. Your project directory paths in the lab will start with `U:`. The paths to the TI software and libraries, described in Section 3.2 should be as given below.

3.1 Creating a Workspace

Every time you start CCS, a `Workspace Launcher` window will pop up asking you if you want to use the last-used workspace. The first time, since there is no previous workspace, it will suggest a default name `C:/Users/<username>/workspace_v5_5`. Edit the name to, for example, `U:/Lab1`, where the U drive belongs to whoever logged into the lab computer. CCS will create the directory or you could have created it yourself beforehand using the windows file explorer. Putting your workspaces and projects on the `U:` drive will allow you to access them remotely, which could be useful when you are writing up a report.

Where you create this workspace under your `U:` directory is not important. For example, you can move the workspace down several levels by changing the workspace name to, for example, `U:/ECE423/Labs/Lab1`. However, once you make this first choice, you should stay with it on subsequent labs. Note also that if you're working as a team, you might want to alternate whose `U:/` drive is used for different labs.

Once you have clicked `OK` to create the workspace, CCS will start up in the "Edit Perspective". It may show the "TI Resource Explorer" window. If so, click the red X to dismiss (it can be brought back if needed). The next step is to create a project. The required settings will be described in the next section. Section 3.3 will discuss methods for cloning a project so that you

will not need to re-enter the settings.

3.2 Creating, Compiling, and Debugging a Project

1. Create a project within the workspace you just created by doing the following:

   `->File->New->CCS Project`

   A window titled `New CCS Project` will pop up. See Figure 4. Give the project a descriptive name, for example ``sine_gen_intr'' to signify the first exercise in Lab 1, generation of sine wave using interrupts. Use drop-down arrows to fill in all the boxes as in Figure 4. Leave the `Linker command file` box blank because you will be copying in a custom file later. Note that you may need to click the drop-down triangle next to `Advanced Settings` to reveal the boxes.

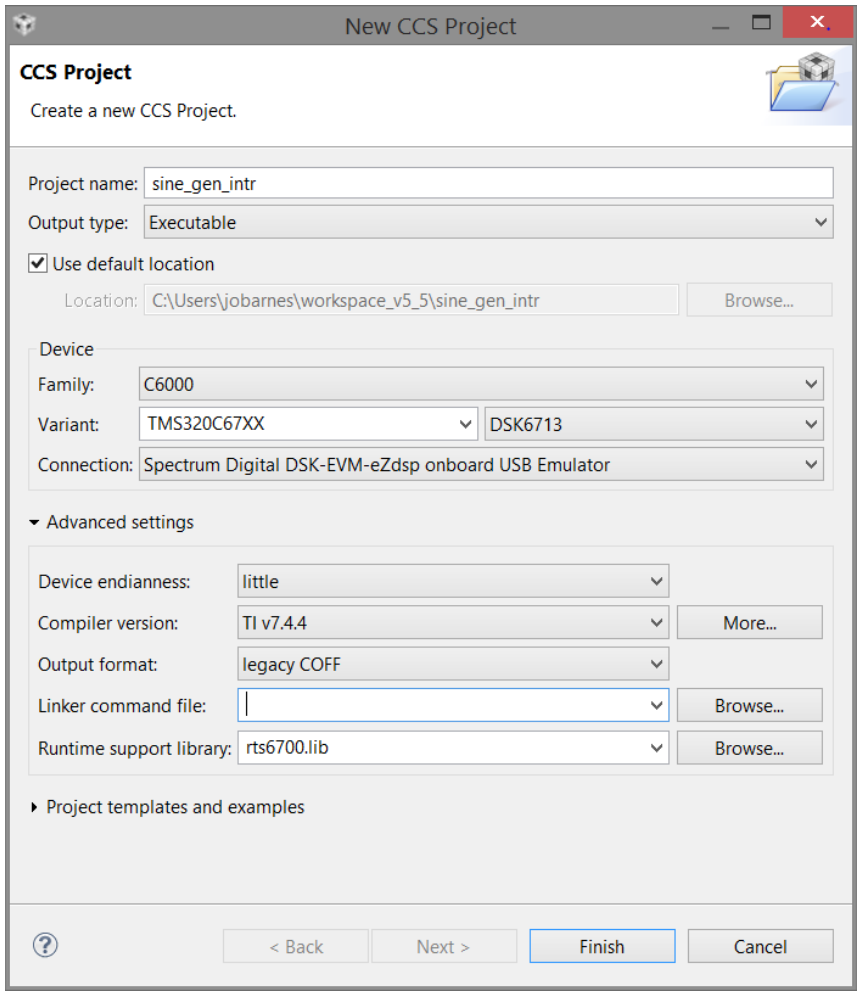

Figure 4: New CCS Project Dialog

When you click `FINISH`, CCS will create a project folder icon in the CCS `Project Explorer` pane. This was shown in Figure 1; this workspace contains three projects, only one of which is open. Projects can be in a state of `OPEN` or `CLOSED`. To open a project,

right-click on the folder and click `Open Project`. Use the same procedure to close a project. In general, only the project you are working on will be open, an exception being when you need to copy files from one project to another. This can be done in the `Project Explorer` pane.

Figure 1 shows the project `sine_gen_intr` after all of the files to create the program have been added, as will be described below. Your view will only show a stub file `main.c`.

When you click on the triangle to the left of an `OPEN` project, the folder icon will open and the contents will be shown below the icon. When you click on the project folder icon or project name, the text will become bold and CCS will append `[Active - Debug]` to the name, signifying that the project is "in focus". Focus is necessary because CCS allows multiple projects to be simultaneously open and modifiable, but only one project can be loaded on the board. Click to make the project you created in focus, then click on build icon above (the hammer). This will build (compile) the project and create an executable that does absolutely nothing. You of course will edit `main.c` and put in your own code or delete it and replace it with a file of your own. Note that the top-level C source file need not be named `main.c` but it must contain the `main()` function.

Once you have created the project, copy the file `Support.zip` to a temporary location and expand it. Copy the files in the expanded `Support/` directory into the project directory (not the directory, just the contents). You can do this using the windows file explorer or within CCS clicking

```
->Project->Add Files...
```

and navigating to the expanded directory.

2. Now you need to tell CCS the locations of the chip and board support libraries, and set a few options. This is needed for both the compiling and linking stages and is done be modifying properties of the project. Note that you should only need to do these steps on your first project. From that point you can clone the project for subsequent projects, which will copy the settings, and just modify your code.

   If necessary, click on the project name to bring it in focus and do

   ```
   ->Project->Properties
   ```

   This will bring up the `Project Properties` window - see Figure 5. Fill in the following by navigating through the drop-down settings as follows:

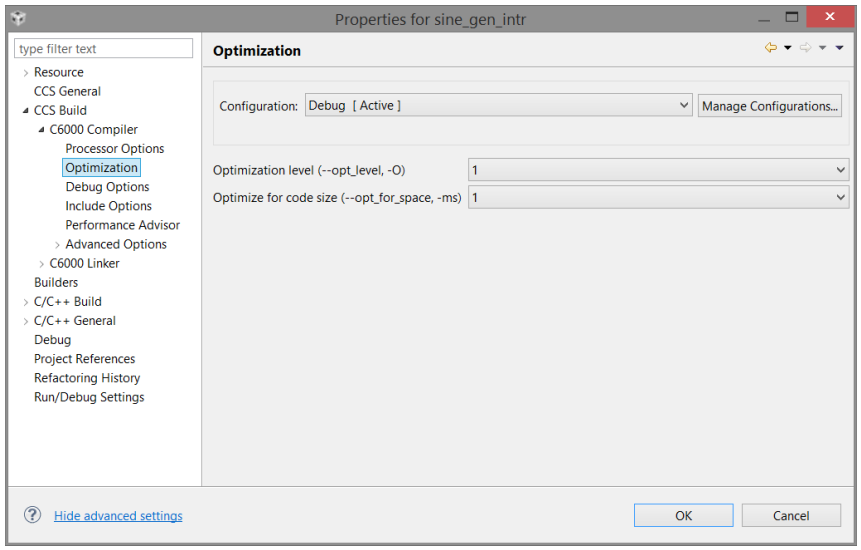

Figure 5: Project Properties Window

`CCS Build->C6000 Compiler->Processor Options` Fill in the `Target processor version` box with the entry `6700`.

`CCS Build->C6000 Compiler->Optimization` Set both Optimization levels to 1 using the drop-down arrows.

`CCS Build->C6000 Compiler->Include Options` This will bring up a window with two panels (Figure 6). The top panel is where you add the paths clicking on the green icon on the right side of the upper border, clicking on the Filesystem button, and filling in the paths. Note that you can find what the other icons do by hovering the cursor.

In the top panel, the first path was added by CCS and points to the core libraries. The next two, which you add, are paths to the chip and board support libraries.

```
``C:\ti\DSK6713\c6000\bios\include''
``C:\ti\DSK6713\c6000\dsk6713\include''
```

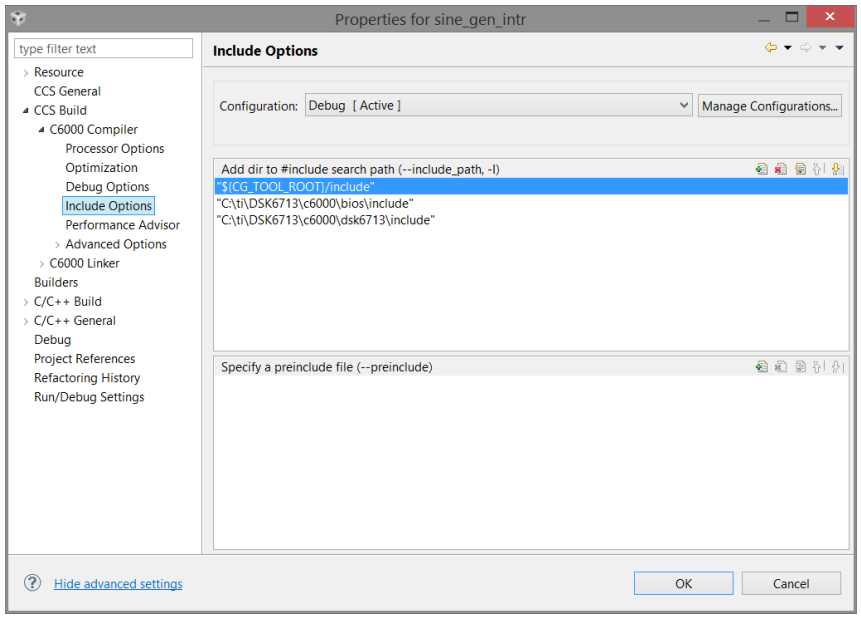

Figure 6: Includes Settings

`CCS Build->C6000 Compiler->Advanced Options->Predefined Symbols` Add `CHIP_6713` to the `Pre-define NAME` panel by clicking on the green plus on the upper right frame.

`CCS Build->C6000 Compiler->Advanced Options->Runtime Model Options` In the box to the right of the text `Data access model (--mem_model:data)`, use the drop-down arrow to make the box read `far`

Linker Paths Click on

`->Properties->Build->C6000 Linker->File Search Path`.

Add the following libraries to the top panel (`libc.a` is only needed if you will be using certain c library functions such as `printf()`):

```
``libc.a''
``csl6713.lib''
``dsk6713bsl.lib
```

Add the following search paths to the bottom panel (the first entry was added by CCS):

```
``C:\ti\DSK6713\c6000\bios\lib''
``C:\ti\DSK6713\c6000\dsk6713\lib''
```

3. The actual file for creating the sine wave can be downloaded from this link: `sine_gen_intr.c`. Lab1 will use this file as a starting point and will explain the function of the program. Download this file and copy it into the project directory. Afterwards, delete the `main.c` file created by CCS.

   This program uses interrupts to co-ordinate between the codec (ADC/DAC chip) and the DSP chip. As will be discussed in the Lab writeup and later in the course, the support file `Vectors_intr.asm` is needed and the file `Vectors_poll.asm` will not be used. In order

to prevent conflict between these two asm files, the `Vectors_poll.asm` must be prevented from being compiled. Rather than deleting it, it can be "hidden" from the compiler by right clicking on the file and checking `Exclude from Build`.

After excluding the polling asm file, you can run the compile by clicking on the hammer symbol at the top of the screen or clicking

`->Project->Build All`

If this is the first time you have compiled, there may be a compilation of the runtime library `rts6700.lib`, supplied by TI, and there will be two errors in compiling two trigonometric functions. You can ignore these errors. Subsequent compiles of only the user code should be clean.

4. Power the board and plug in the USB communication cable (if you haven't already done so). **Make sure the "computer" end of the USB cable is plugged into the USB port labeled "2.0"**. This is the port near the top of the computer. The DSK6713 is incompatible with USB3.0.

5. Within CCS, run the debugger, which will load the `<project>.out` file onto the board and allow you to run or single-step the program. The debugger is invoked by clicking on the bug icon next to the hammer or clicking

   `->Run->Debug`

6. Connect a set of headphones (or ear buds) to the headphone jack on the board and you should hear a high frequency tone.

3.3 Cloning Workspaces and Projects

Cloning a Workspace

Once the first workspace is created, you can easily create another workspace in one of two ways.

- If you are starting up CCS, just edit the workspace name in the `Workspace Launcher` window. The new workspace will be created and CCS will re-launch into it.
- If you are already in CCS in an existing workspace, just do

  `->File->Switch Workspace->Other`.

  This will bring up the `Workspace Launcher` window and you can change the current workspace name to the new workspace name.
  This is also how you switch between existing workspaces, except you do not select `Other` but from the list of existing workspaces. Only workspaces at that directory level will be listed (which should be all if you are following the guidelines).

Either of these methods will copy settings from the previous workspace, saving you some time.

Cloning a Project

Once you have created your first project, creating additional projects can be easy as a copy and paste operation; all of the project settings including paths and options will be copied.

The procedure is slightly different for cloning within a workspace and from one workspace to a newly created empty workspace.

Cloning within a workspace Most of the windows file explorer commands work within the CCS Project Explorer window. Simply do a copy-paste-rename operation. Then you replace or modify the *.cc files within the cloned project. It is suggested to use the sine-gen project as a template, since it does use the project settings for most of the lab exercises.

Cloning from one workspace to a new workspace For this, you will need to start CCS in the new workspace and use

```
->File->Import->Code Composer Studio->Existing CCS Eclipse Projects
```

and browse to the template project in the other workspace. It is suggested to use the first project cloned this way as a template and then use copy-paste-rename to make the new projects.

4 Hints, Troubleshooting

4.1 Compile/Linker Errors

- Incorrect configuration ("Properties" of project in CCS)
  - Check that the include paths are complete and correct. Expanding any of the entries under the includes directory in the project explorer should list all the header files. Clicking on these should reveal all the defines in the header file. You can also try the search capabilities available from

    `->Properties->Index`
  - make sure `CHIP_6713` is defined.

4.2 Cannot download program to board

Possible causes and fixes are:

- USB3.0 port on computer is being used. The DSK USB interface seems not to work with USB3.0. Switch to the port marked USB2.0, and try downloading again. If that doesn't work, to the power-down/power-up procedure.
- Corruption of the USB link. Often this problem will be accompanied by a pop-up with the text "Error initializing emulator". A typical cause is rebooting the computer with the USB cable connected to a powered board; this can leave the USB link or the "emulator" chip in a corrupted state. The simplest fix is to "reboot" the DSK: unplug the USB cable, unplug the power cable, wait 10 seconds, then power the board, wait for the self-test to complete, then re-connect the USB cable.