Lab 1: Introduction to eZDSP5515/35 and Code Composer Studio v4

EE 352 DSP Laboratory

Lab Goals

- Give a general overview of a digital signal processor
- Overview of eZDSP5515/35, Code Composer Studio(CCS) v4
  - Demonstrate through the tutorial programs
  - Acquaintance with the 5515/35 board and the CCS interface.
  - Creating a new project, compiling and editing.
  - LED, Switch interfacing
  - Debugging tools like breakpoint, watch window etc.

1 Introduction

1.1 Overview of DSPs

DSP stands for digital signal processing and it often involves taking an analog signal, converting it to samples of digital values, processing the digital data, and converting it back to analog for output. We often consider DSP along with the analog front and back-ends since that is what we ultimately hear and see. Why do we process signals digitally rather than working with the original signal in the analog domain? The answer is it depends on the system and its requirements. For some systems, working with the analog signals gives a better performance. For others, digital processing might be better. It is up to you, the designer, to make the decision based on materials you learn through this and other courses.

1.2 DSP Processors vs. General-purpose Microprocessors

How do DSP processors differ from microprocessors? Both types of processors perform computations on digital signal have approximately same level of integration and the same clock frequency. But on the signal processing tasks (like complex arithmetic operations) DSPs overtake them from 2 to 3 times in speed. DSP processors are also designed to consume less power under its target applications. Products that use DSPs include DVD players, camcorders, cell phones, wireless base stations, modems, karaoke players, etc. The gap between DSP processors and microprocessors is narrowing since microprocessors are approaching, if not exceeding, throughput capabilities of the DSP processors. In the personal computer industry, microprocessors are taking over several computational tasks that used to be cost effective only when implemented with DSPs. This results in a narrowing market segment for DSP processors in the PC industry. However, DSP processors remain highly relevant in the embedded system area where high throughput and low power computing are required.

1.3 Introduction to eZDSP5515 and Code Composer Studio v4

For this lab we will use ezDSP5515/35. Block diagram of C5515 is shown in Fig. 1 for reference.

Some features of the ezDSP5515/35 kit are as follows:

- DSP max operating frequency: 120MHz (C5515), 100MHz (C5535).
- On board 32-bit TLV320AIC3204 ultra low power stereo codec with 8 kHz to 192kHz sampling rates support

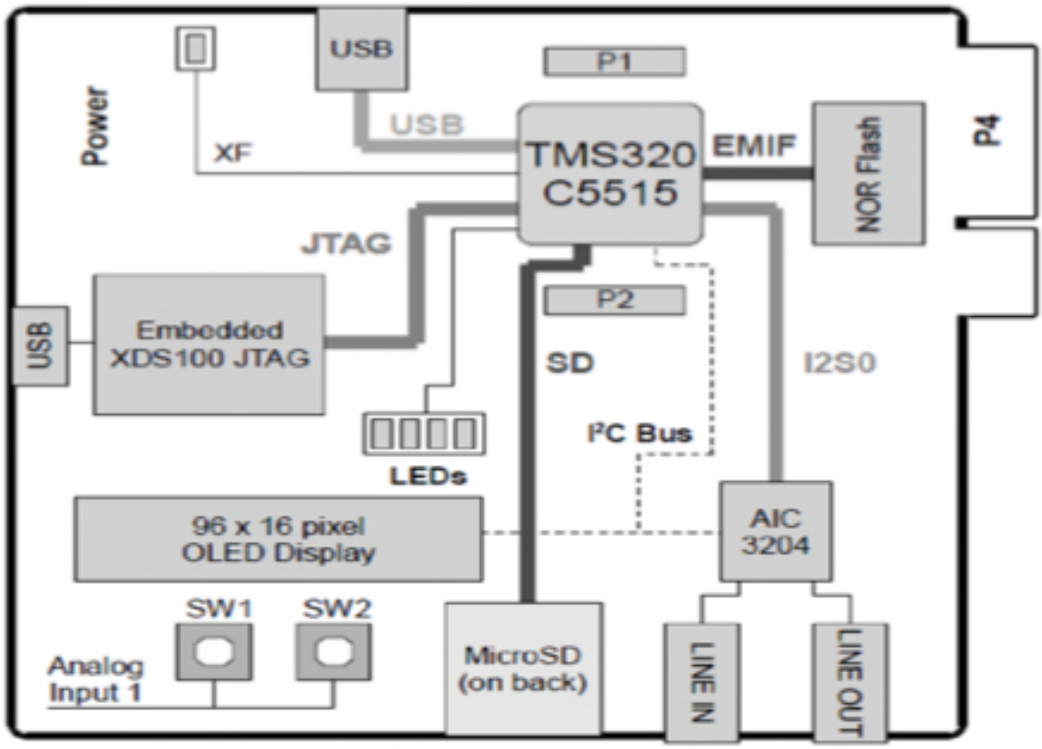

Figure 1: Block diagram of ezDSP5515.

- 2 Push-button Switches and 5 Programmable LEDS (4 color LEDs - 2 of Green, 1 each of Blue, Red, Orange)
- 4 Mbytes (C5515), 8 MBytes (C5535) External Flash Memory
- 128K Bytes on-chip ROM
- 320K Bytes on-chip RAM
- Includes an inbuilt tightly coupled FFT Hardware Accelerator which supports 8 to 1024-point (in power of 2) real and complex-valued FFTs.

The students are suggested to read Texas Instruments Technical Reference Material provided along with this handout to learn more about C5515/35 and its features.

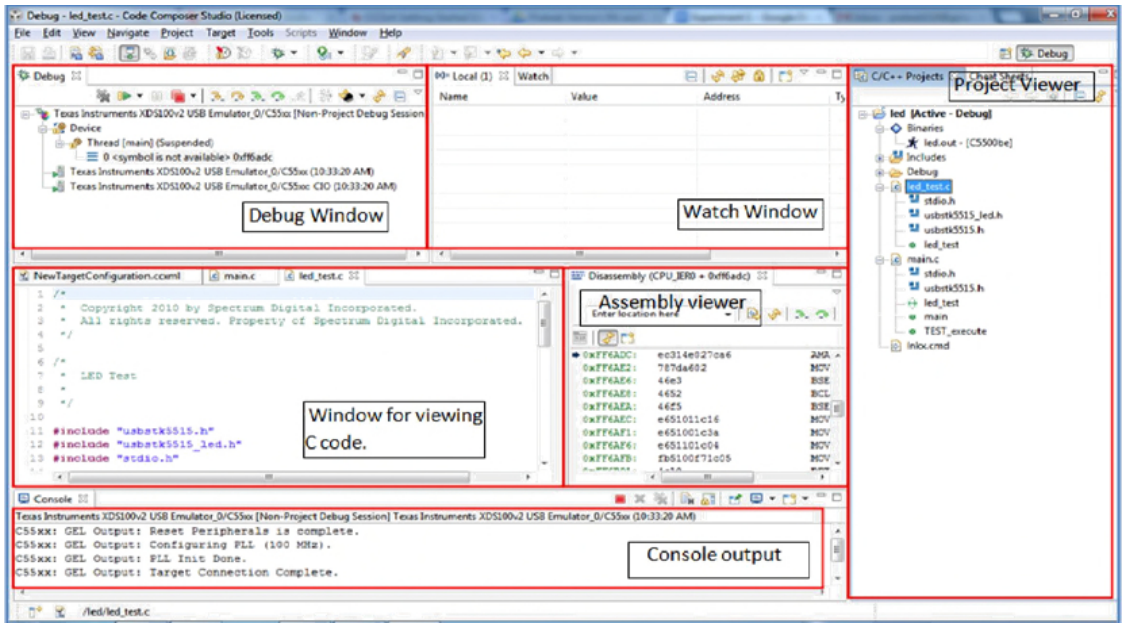

Figure 2: CCS interface or development environment.

Code Composer studio interface will be used to program the eZDSP. It is based on the Eclipse framework. It communicates with the DSP using an onboard JTAG emulator through a USB interface. ICs with embedded JTAG allow access to all pins and test points in package with just 4 pins as an interface to the external world. So, once the processor is housed onto the board along with other peripherals, JTAG enables its user to access its internal registry and therefore, provides with the luxury of debugging the programs by checking the contents of the relevant registers while a program is running on the processor. In addition to this, the IDE supports a lot of developing tools for compiling, editing, profiling, graphing, debugging and optimizing for a variety of embedded applications. The following instructions tell you about the various steps to be followed for setting up of embedded target.

2 Getting Started with CCS

Learning objectives for the section

After reading this section, you must be able to,

- succesfully set-up a JTAG connection between the CCS IDE and the kit.
- copy an existing project from the examples to the workspace, build it and run it on the device.
- create a new project for your device.
- switch between projects
- run a built project on the processor and observe the output.

Getting the board working

This involves the following 5 steps:

1. Connect the device to the CCS.
2. Open an existing project or create a new project.
3. Configure the linker options and file search path.
4. Build the project.
5. Run the program on the processor.

These steps are explained in detail in the following sections.

2.1 Connecting the device to the CCS

- Create a new workspace (preferably on Desktop) and store your projects in it.
- Launch CCS v4 from the shortcut on the desktop. (This was created when CCS v4 was installed).
- The CCS v4 window will appear. Click the `Target` menu, then select `New Target Configuration File`.
- The `New Target Configuration` window will appear. Enter a file name that describes the emulator connection and/or Texas Instruments device being used and click `Finish`. For example: `C5515_USBStick`.
- The `Basic` configuration setup window will open in the CCS v4. Select `Texas Instruments XDS100 USB Emulator` from the `Connection` menu. Type `5515` in the `Device` field and select `USBSTK5515`, from the list.
- Click the `Save` button to save the configuration (see Fig. 3).
- Click the `View` menu and select `Target Configurations` to expose the configuration(s) that have been built or imported. A new tab labeled `Target Configurations` will become available in the CCS window.

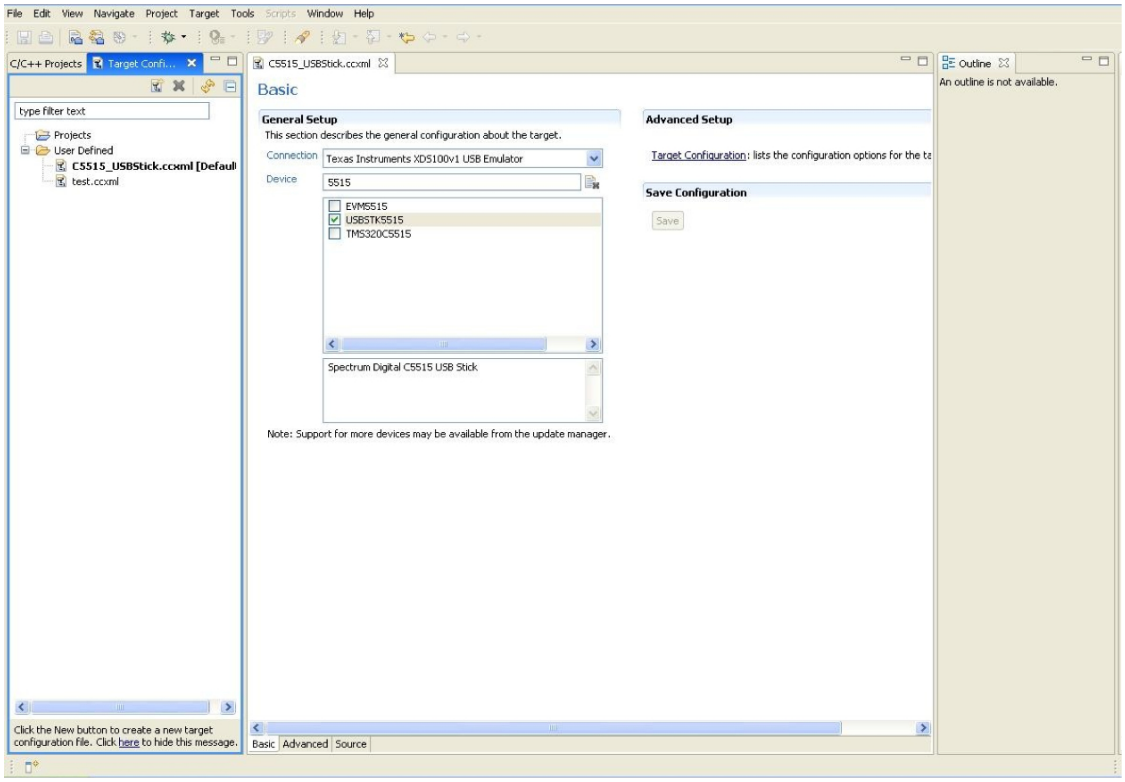

Figure 3: CCS v4 target configuration window.

- Expand the `User Defined` folder. Right-click on the new configuration that has been created and click `Launch Selected Configuration`.
- Connect the device to the pc. Click `View -> Debug` then click `Target -> Connect Target`. CCS will attempt to connect to the C5515 eZdsp USB Stick and run the GEL file. Once the connection is successful and the GEL file has finished, the console window will print a message stating `Target Connection Complete`.
- Even with C5535 board, the same configuration works.

2.2 Project Creation

Make sure that you are in `C/C++` perspective (a pre-defined layout of sub-windows in CCS) before moving ahead. To find out which perspective you are currently in, look at the top-right corner of the CCS window. You will be either in `C/C++` or `debug` perspective. If you are in `debug` perspective, click `Window -> Open Perspective -> C/C++`.

Please follow the workflow in **only** one of the following two subsections. You either use an existing project or create a new one.

2.2.1 Opening an existing project: Sample LED project

This project will focus on blinking the set of LEDs present on the DSP board. Implement the set of instructions given below.

- The C5515 eZdsp USB Stick must be connected in CCS before proceeding. If the board is not connected, press `Alt + C`
- Click `File -> Import`.
- When the new window appears, expand `CCS` and select `Existing CCS/CCE Eclipse Project`. Click `Next`.

- Select `Select Search-Directory` and click the `Browse` button. Browse to directory path `Install_Dir/ccsv4/emulation/boards/usbstk5515_v1/tests/led/`.[1] The default `Install_Dir` is `C:/Program Files (x86)/Texas Instruments`.
- Highlight the `LED` folder. Select the `copy project into workspace` option. Then click `Finish`.
- Click `View -> C/C++ Projects`. A new tab will appear in the CCS v4 window and there should be an LED project visible.

2.2.2 Creating a new project

- Select `File -> New -> other`. Select `C/C++ -> CCS Project` and press `Next`. Give suitable name to the project. The Project will be stored in a workspace which can be changed accordingly. All the files should now be stored in the same workspace.
- Select the `Project type` as `C5500`. Here, `Debug` and `Release` are enabled by default. Click `Next` to continue.
- You will see the names of some projects which are already open in your CCS. This new window is used to create dependencies with other projects in the workspace. Tick on the projects you want to reference or else leave it blank. Click `Next` to continue.
- Select `executable` from the `output type`. Select `TMS320C55XX` followed by `TMS320C5515` from the device variant. Select `lnkx.cmd` from `linker command file`. Click `Finish` to complete the wizard.
- Click on the icon on the bottom left of the CCS window and select `C/C++ projects` to see the created project.
- To create a new source file, right click on the active project and then select `New` followed by `Source file`. You can separately create files for different blocks of your program to give it a structured look. This completes your first project.

2.3 Configuring the linker options and file-search paths

- Right click on the project and select `Build properties`.
- From `C5500 Compiler`, select `Include options` and add the path `Program Files (x86)/Texas Instruments/ccsv4/emulation/boards/usbstk5515_v1/include`. This is shown in Fig. 4.
- Select `Runtime Model options`. Change the `Specify Memory Model` from `Huge` to `Large`.
- Now go to `C5500 linkers` and select `File search path`. From the section named, `Include library file or command file as input(--library, -l)` (Refer to figure 5), remove any lib files already present (For ex. `libc.a` as shown in the figure). Then, Include the following two library files: `C:/Program Files (x86)/Texas Instruments/ccsv4/tools/compiler/c5500/lib/rts55x.lib` and `C:/Program Files (x86)/Texas Instruments/ccsv4/emulation/boards/usbstk5515_v1/lib/usbstk5515bsl.lib`.

2.4 Building the project

- Make sure that you have modified the build properties as mentioned in section 2.3.
- Click `Project -> Build Active Project`.
- When the build is complete, the console window will notify the end of the process.
- Look in the `C/C++ Projects` tab again and expand the `Binaries` folder. There now should be a `led.out` file listed which marks the end of the build process.

[1] Wherever a path to a directory or a file needs to be entered in the IDE, do not copy-paste the path from the manual pdf to the IDE. Browse to that path manually in the IDE and insert it.

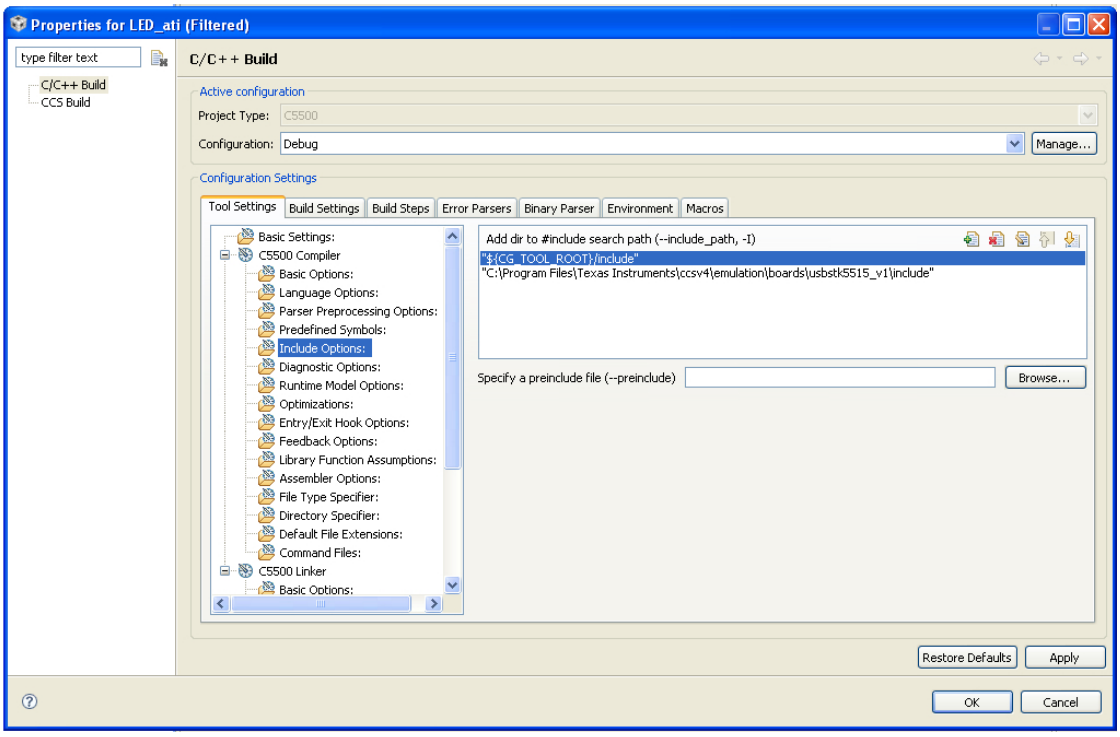

Figure 4: CCS interface demonstration showing include options for the LED project.

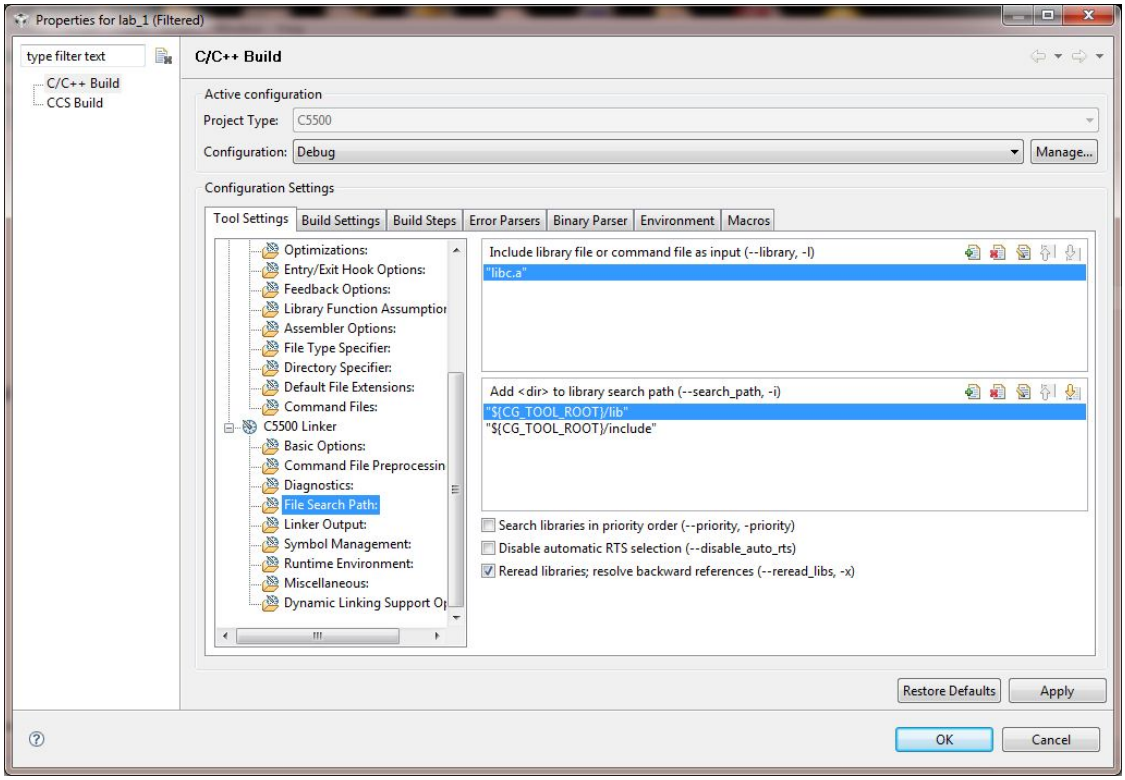

Figure 5: CCS interface demonstration showing linker options for a project.

2.5 Running the program on the processor

- Right click on `led.out` and click `Debug As -> Debug Session`.
- Click `Target -> Run`. The C5515 eZdsp USB Stick `XF` LED should now blink.
- To stop the project use `Halt`. Using `Terminate all` stops the running program and closes the active project.

2.6 Learning Checkpoints

All the following tasks need to be shown to your TA to get full credit for this lab session.

1. Delete the LED project from the workspace and again import it to the CCS. But, this time *show* the compiler errors you get to your TA after setting each of the linker/compiler options. Finally, build and run the project on the kit and get it verified with the TA.
2. Create and run the sample `switch` project from the path `Install_Dir/ccsv4/emulation/boards/usbstk5515_v1/tests/switch/`.
3. Disconnect the device, re-connect it and run the `LED` project again.
4. Set the `switch` project as the active project and run it on the device.

3 Debugging Tools

3.1 Breakpoints for Debugging

One of the main crucial steps in coding is Debugging. CCS supports a wide range of debugging tools. This section will describe adding breakpoints to a particular piece of code.

- First identify where you have to put the break point.
- After this, build the active project by going to the `Project` tab or use the shortcut `Ctrl + Shift + P`.
- Load the program onto the board by going to the `C/C++ Project` window, inside the `Binaries`, notice that a `.out` file is created.
- Load the program as mentioned in section 2.5.
- Now double click on the left hand side in the margin at a point of the code where breakpoint has to be added.
- Notice a blue circle has appeared on the left hand side. Right click on this breakpoint to change various properties associated with this breakpoint.
- Now run the code by pressing `F8`. The code will reach the point till the breakpoint and an arrow will appear. `Step into` the code by pressing `F5`.

3.2 Watch Window and Memory/Register view

In addition to viewing the memory register as well as the watch window, there is a provision to see both the `C code` as well as the `Assembly code` simultaneously in the `Disassembly` window which can be opened from the `View` tab.

In order to view the variables, open the `Watch window` from `View` tab and then `Watch`. Now select a variable or an array and right click on the variable and click `Add to Watch Expression`. Follow the steps as given in break point addition and press `F5` and start observing changes in the `Watch window`. The address of each variable is located along with the value of the variable.

3.3 Learning Checkpoints

1. Set a breakpoint in the LED project and step through the code.
2. Debug the program from the `lab_1` project uploaded on the course-wiki page.