

Mechatronics Laboratory Assignment #1

Programming a Digital Signal Processor and the TI 6713 DSP

Recommended Due Date: By your lab time the week of February 2nd

Possible Points: If checked off before your lab time the week of Feb. 9th ... 10 points

If checked off after your lab time the week of Feb. 9th and before your lab time the week of Feb. 16th ... 7 points

If checked off after your lab time the week of Feb. 16th and before your lab time the week of Feb. 23rd ... 5 points

If work not finished before your lab time the week of Feb. 23rd ... 0 points

Goals for this Lab Assignment:

1. Learn about the rules and responsibilities of access to the lab room and its PCs
2. Discuss the use of the PCs, locations to your data, etc.
3. Begin the semester with a quick introduction to the DSP controller and its development software called Code Composer Studio. Specifically:
 - a. Present a quick introduction to “GE423_Project_Creator”
 - b. Learn about the code that GE423_Project_Creator is auto-generating for you.
 - c. Introduce the “DSP/BIOS” kernel and its Configuration Manager.
 - d. Learn how to manage compiler errors and to debug your source code with Breakpoints and the Watch Window.
 - e. Learn some of the quirks of CCS with the C6713DSK.

DSP/BIOS Objects Used:

PRD object

Daughter Card Library Functions Used:

get_switchstate, set_LEDstate

Matlab Functions Used:

None

Prelab: If you have a PC at home you may wish to install the demonstration version of Code Composer Studio (CCS) provided by TI. It is a 90-day trial version that will allow you to practice at home with CCS if you desire. You can download the trial version from the class website, or for those that do not have a fast Internet connection, a CD copy of the software can be checked out from Dan Block.

Laboratory Exercise 1 Read the handout on rules of the lab.

Laboratory Exercise 2 Disk space and file management.

Recommended Drive Usage:

- C: Drive Use a temporary directory on this drive to develop and debug code. Save the results at the end of lab to the W: drive.
- V: Drive This drive is not writable, except for the scratch directory. This directory should only be used to transfer files from one account to another. It is flushed periodically and work should NOT be saved here.
- W: Drive When you login on the stations in the Control Systems Lab, the computer will automatically map to your own personal drive labeled “W:”. This directory is only accessible by your login name, and is your

personal space to store your DSP projects and other files related to the class. Every individual's W: drive will be backed up regularly, so make sure to copy your code off your C drive at the end of class to this drive because we cannot insure the safety of your data on the lab PCs.

Laboratory Exercise 3

- GE423_Project_Creator.exe is found on the V: drive at V:\c6713dsk\project_creator\ge423\GE423_Project_Creator.exe.
- Brief library function documentation is found in the file V:\mechlab\docs\DaughterCard_LibFuncs.pdf.
- Library function source code is found in the directory V:\c6713dsk\include. Because this code is already a part of your default project, it can be viewed in Code Composer Studio once the project is created.

Part (a): Building a Project with GE423_Project_Creator.

For the first part of the Lab Exercise, your TA is going to walk you through the creation of your first DSP project for the semester. This includes creating a project, building the DSP code, and finally running the code on the DSP.

Part (b): What GE423_Project_Creator is doing for you.

Throughout the semester we will use a "GE423_Project_Creator" application to assist you in creating a DSP/BIOS project. It is a small program, but it automatically sets up a number of Code Composer Studio (CCS) initializations that would normally be required if you started a project from scratch. For the second exercise this week, we are going to show you how to perform these steps manually. The following steps illustrate what the "GE423_Project_Creator" application is doing:

1. First we create a new project. In CCS, a new project is created by selecting the **Project→New...** menu option. Give the project any name you wish, select the directory location you wish, select project type as "Executable", and select the TMS320C67XX processor as the target.
2. We now load some C-code for a sample project. Use the Menu item **Project→Add Files to Project...** to add the following files to your project (note the last two files can be added using a multiple selection):
 - a. V:\mechlab\lab1\lab2run1.c
 - b. V:\mechlab\lab1\user_UARTFuncs.c
 - c. V:\C6713DSK\include\c6xdskdigio.c
 - d. V:\C6713DSK\include\switch_led.c
3. Now we create a new DSP/BIOS configuration file. To do this, select the menu option:
File→New→DSP/BIOS Configuration...
 - a. Click the "C6XXX" tab.
 - b. Select **ti.platforms.dsk6713** as the starter configuration, uncheck all of the "Enable DSP/BIOS Features" and click OK.
 - c. Save the TCF file with a new name to your project directory using the **File→Save As...** menu.
 - d. When you save the configuration files, the Configuration Tool automatically generates initialization code for your project. To add this DSP/BIOS configuration to your project, select

- Project→Add Files to Project...** and change the **Files of Type...** option to *.TCF. You should now see the file you just saved. Select this file and add it to your project... you might also notice that the tabs in the file view window (left side of screen) now have files in the 'DSP/BIOS Config' and in the 'Generated Files' tabs.
- e. Also add the *.CMD file that the DSP/BIOS configuration tool creates when you saved the created *.TCF file.
4. Next, we add two C-library function files. These library files have specialized functions for the C6713DSK board. One is found in the directory c:\CCStudio_v3.3\c6000\dsk6713\lib**dsk6713bsl.lib**. and the second is found in the directory c:\CCStudio_v3.3\c6000\csl\lib**csl6713.lib**.
 5. Finally you need to modify the compiler options for your project. Select **Project→Build Options...** to open the compiler options. *Note: If an option is not listed leave it as the default.*
 - a. Compiler Tab, Category Basic
 - i. Target Version = 671x
 - ii. Opt. Level = File (-o3)
 - b. Compiler Tab, Category Files
 - i. Asm Directory = Your project directory, Note: NOT "Asm File Ext"
 - ii. Obj Directory = Your project directory, Note: NOT "Obj File Ext"
 - c. Compiler Tab, Category Preprocessor
 - i. Include Search Path =
c:\CCStudio_v3.3\c6000\dsk6x11\include;c:\CCStudio_v3.3\c6000\dsk6713\include;
V:\c6713dsk\include
 - ii. Pre-Define Symbols should say `_DEBUG;CHIP_6713`
 - d. Linker Tab, Category Basic
 - i. Stack Size = 0x800
 6. Save your project. Do this by selecting **Project→Save**. You can now exit CCS without losing your project, and you can reopen everything using the **Project→Open...** option.
 7. Build your project. Do this by selecting the **Project→Build**. The project should build correctly with 0 Errors, 0 warnings; 0 remarks.
 8. You may close the project, using **Project→Close**.
 9. Show your TA that you were able to build this project. You do not have to download and run this project. This was just an exercise to show you what Project Creator is doing for you. Be thankful that you will not have to do this again this semester.

Note: Get in the habit of saving your project often. As you design controllers in the future, the two most common mistakes are (1) to accidentally access memory space that is used by the program code and (2) run a complex DSP code from a project on the network. Both can unexpectedly cause the DSP and Code Composer Studio to crash, and if your project is not saved, you will permanently lose any changes you have made.

Part (c): DSP Bios Kernel and Configuration Manager Example: The Period Object

In this section of the exercise, the student will create a new application that uses the PRD (Period) object in DSP/BIOS. First use **GE423_Project_Creator** to generate a new project (**do NOT go through the steps of part (b)**). Give the project a descriptive name so that you can find it easily. You will also want to organize your projects in some kind of directory structure first on the C:\ drive and then backed up on your W:\ drive. I recommend something like `c:\your_login_name\ge423\lab1\switchchecker`.

10. Open your project inside CCS and find the configuration (*.tcf) file for your **switchchecker** project. Double click on the TCF file and the Configuration Tool will open. The Configuration Tool shows the DSP/BIOS kernel setup for your project. Modify your DSP/BIOS configuration to add a period object by selecting the tab: **Scheduling→PRD**. Right click over the PRD tab, and select **Insert PRD**. You should see a new PRD object created. Give the period a name other than PRD0. While the name can be anything, the industry convention is that period objects should begin with 'PRD_'. Thus, one good name for the period object might be PRD_checkswitches.
11. Set the period object to call a function every second. To set the PRD object properties, right click over the name and select **Properties**. A menu will appear with the following options:
 - i. Comments: can be anything descriptive
 - ii. Period (ticks): The number of clock ticks to wait before executing the function. The default ticks per unit time is 1 tick per millisecond.
 - iii. Mode: Select 'Continuous' so that the function will be repeatedly called.
 - iv. Function: The name of your c-function (see Step 12) that will be executed each period, with the function name prefaced with an underscore. For example, if your function is called "CheckMe", then the entry should be "_CheckMe" (this is how assembly code accesses C-functions).
 - v. Arg1/Arg2: Leave these set to 0x00000000. These are used to pass arguments to the function if needed. We don't use this option.
12. Write a function in your main program to read the status of the four dip-switches and echo their state to the four LEDs. A switch that is UP should turn the corresponding LED ON, and a switch that is DOWN should turn the corresponding LED OFF. Compile and download your code to the DSP, and when finished demonstrate your application to your TA.

If you are working with a partner on this exercise, trade places and give the other person a chance to do the coding for the next several steps.

13. Create another new project with Project Creator. Again add a PRD object to your DSP/BIOS configuration, and have the period object call a function (see step 14) every second.
14. Within your period function, increment a global integer variable by one. However, if switch 1 is down you should not increment the count. Display the last four bits of the count to the four LEDs. Compile and download to the DSP, and when finished demonstrate your application to your TA.

Part (d):: Breakpoints and Watch Windows. Using the code you just finished, we want you to experiment with adding breakpoints to your code and using the “Watch Window” to edit the values of your variables.

15. In your previous code and with the DSP halted, put your cursor over the integer variable that you are incrementing. You should see a value appear corresponding to the value of the variable. Run your code, halt it again, and again put your cursor over the same variable to confirm that it changes.
16. An easier method than using the cursor repeatedly is to add the variable to a watch window. When the DSP is halted, the watch window displays the current values of each variable in the watch window. To add your counting integer variable to the watch window, highlight the variable and then right-click, then select **Add to Watch Window**. The variable will appear in a window to the lower right, with the current value of the variable listed alongside. The Watch Window dialog is also found under the View menu.
17. **IMPORTANT! BREAKPOINTS! IMPORTANT!** *Breakpoints (along with the Watch Window) make debugging source code so much easier. In fact as a general rule for your lab assignments, you will need to set a few breakpoints and try to figure out your source code errors before asking an instructor for help. A large part of this mechatronics course is teaching you how to debug an embedded control system.*

Add a breakpoint to your code by double clicking on the left gray margin of your source file. A breakpoint is a location where the program will literally halt during execution. This allows you to check the values of your variables midway during operation. After a breakpoint, you can single step through your code using the (F10) key and watch the variables update as different calculations are performed. You remove breakpoints by again clicking in the left gray margin.

Note: The DSP compiler is very advanced and is optimized to produce assembly code that has a speed, on average, of approximately 70% of that of hand-coded assembly, i.e. the theoretical maximum usage of the DSP. If your code contains simple loop instructions or if statements, then the C compiler may optimize your code to a point that eliminates portions of your code, thus making it impossible to set breakpoints at each line. To get around this, you can turn off optimizations in the compiler options or you can declare the variables of interest as “volatile.” This will allow you to single step through your code in the sections you are debugging. You will learn more about this “volatile” variable type in upcoming labs.

18. Use also the menu item **View→Mixed Source/ASM** to see how the compiler is optimizing your code. This shows a listing of the c-code together with the resulting assembly code. We will not be programming in assembly, but some students have learned it and can use this view for additional debugging.

As a final set of exercises for this lab perform the following:

19. See what happens to your connection to CCS when you lose power to the DSP. To restore the connection you will need to:
 - i. Close CCS.

- ii. Power On and Off the DSK.
 - iii. Re-launch CCS.
20. Also, if you are lucky and have not yet received any compiler errors during any of the above exercises, you should intentionally add some errors to your code so that you will see how CCS will alert you during the build process.

Lab Check Off:

1. Demonstrate you have completed Parts A and B.
2. Explain to your TA the given source code in the main C file. The DaughterCard_LibFuncs.pdf file will help you explain the different function calls. We are not asking for a lot of detail here, we just mainly want you to find and read the function documentation. It also may be helpful to look at the actual source code in the include directory.
3. Demonstrate your first application, the one that continually checks the status of the four dip-switches and displays their current state on the four LEDs. (First half of Part C)
4. Demonstrate your second application, the one that updates a counter every second and outputs the last four bits of the count to the four LEDs. The count should stop if switch 1 is in the down position and resume when it is in the up position. (Second half of Part C)
5. Demonstrate that you know how to use Breakpoints and the Watch Window to debug your source code. (Part D)
6. Demonstrate what happens when you power off the DSK with CCS still running. How do you recover from that situation? (Part D)