San José State University
Department of Computer Engineering
CMPE 146-01, Real-Time Embedded System Co-Design, Fall 2024

Lab Assignment 1

Due date: 09/08/2024, Sunday

1. Overview

In this assignment, we will be familiarized with the Texas Instruments MSP432P401R LaunchPad Development Kit and the TI IDE (Integrated Development Environment). The development kit features the MSP432P401R, a 48MHz ARM Cortex-M4F MCU. We will also learn how to use the device driver library (DriverLib) functions provided by the TI SDK (Software Development Kit).

2. Resources

Key documents about the hardware and software can be found on Canvas. There are four files under the Labs/Docs folder in the Files section:

LaunchPad User's Guide: TI MSP432P401R SimpleLink Microcontroller LaunchPad User Guide.pdf
MCU datasheet: TI MSP432P401x SimpleLink Microcontrollers datasheet.pdf
MCU technical reference: TI MSP432P4xx SimpleLink Microcontrollers Technical Reference Manual.pdf
DriverLib User's Guide: TI MSP432 DriverLib Users Guide MSP432P4xx-4_30_01_01.pdf

3. Exercise 1 CCS and SDK

In this exercise, we will set up the TI IDE tool, CCS (Code Composer Studio), and the SDK targeted for the specific MCU used on the LaunchPad.

Code Composer Studio User's Guide can be found at https://software-dl.ti.com/ccs/esd/documents/users_guide/index.html. The installation procedure can be found in Chapter 3 of the user's guide, https://software-dl.ti.com/ccs/esd/documents/users_guide/ccs_installation.html.

Go to the "Code Composer Studio Downloads" website: https://www.ti.com/tool/download/CCSTUDIO. Download the "single file (offline)" installer. The current version is 12.8.0. CCS is supported on three different OS platforms: Windows, macOS and Linux.

If you use a Windows machine, for example, click on the "Windows single file (offline) installer for Code Composer Studio IDE (all features, devices)" link. You will download the CCS12.8.0.00012_win64.zip file to your local disk. The file size is about 1.4 GB, so it may take a few minutes to complete the download. After the download is complete, unzip the file. Run the setup program ccs_setup_12.8.0.00012.exe.

If you use a macOS machine, click on the "macOS single file (offline) installer for Code Composer Studio IDE (all features, devices)" link. You will download the ccs_installer_osx_12.8.0.00012.dmg. Open the disk image file. Run the setup program ccs_setup_12.8.0.00012.app.

Follow the on-screen instructions to install the CCS. When you get to the "Select Components" screen, check "SimpleLink™ MSP432™ low power + performance MCUs." Also, when you get to the "Install debug probes" screen, just click on "Next" to continue. Discard the message (if appears) to reboot the machine upon installation; it is not really necessary to do that. It is a large installer, so the entire installation process may take more than half an hour on a slow machine.

To install the SDK, use the installer available in a Google drive. Go to the SDK_installers folder in the Google drive folder:
https://drive.google.com/drive/u/1/folders/1yZ08ZBT5wAmH6w1bjTJDmYINbztM11SO.

If you use a Windows machine, download the SDK installer *Win_simplelink_msp432p4_sdk_3_40_01_02.exe*. Execute the program to install the SDK in your local disk. If the OS does not allow you to run the program due to some security reason, open a Command Prompt window (run the cmd program) with admin privilege, go to the folder (with the cd command) where the executable file is, and type the filename to run the program. When it is done, you should see a new folder created in the TI software folder, for example, *C:\ti\simplelink_msp432p4_sdk_3_40_01_02*.

If you use a macOS machine, download the SDK installer *macOS_simplelink_msp432p4_sdk_3_40_01_02.app.zip*. Open the file to extract the setup program *macOS_simplelink_msp432p4_sdk_3_40_01_02.app*. Run the program. A new folder will be created in the TI software folder, for example, */Applications/ti/simplelink_msp432p4_sdk_3_40_01_02*.

Lab Report Submission

1. List any issues that you may have encountered during the setups. Describe how they were resolved.
2. Include a screenshot of the TI folder showing the CCS and SDK folders.

4. Exercise 2 Hello world

In this exercise, we will create a new project on CCS to have the LaunchPad send a simple message to the CCS debug console.

Quite often, it is much quicker to import an example *empty* project and modify it to suit our needs than to create a new project from scratch. On the CCS menu bar, click on <Project><Import CCS Projects…>. On the pop-up "Import CCS Projects" dialog box, click on the "Browse…" button. Go to the example project folder, for example, on Windows, it would be something like C:\ti\simplelink_msp432p4_sdk_3_40_01_02\examples\nortos\MSP_EXP432P401R\driverlib\empty. Click on the ccs folder (single-click, not double-click), then click on the "Select Folder" button. Finally, click on the "Finish" button to import the project.

Open the Project Explorer by clicking <View><Project Explorer> on the menu bar. Expand the project tree and double-click on *main.c* to bring it to the text editor.

We are going to use the library function *printf*, so insert the following line in the #*include* section.

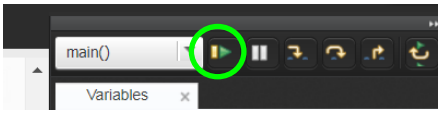
```
#include <stdio.h>
```

Replace the contents of function *main()* with just the following line:

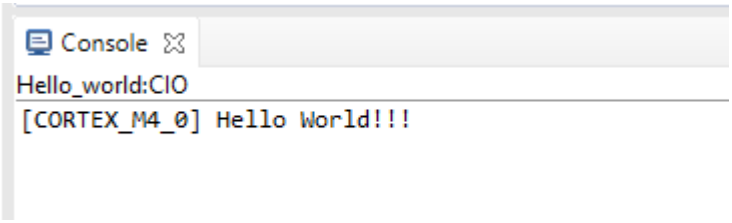
```
printf("Hello World!!!\n");
```

Connect the LaunchPad to your laptop/desktop with the USB cable (that comes with the LaunchPad). To build and run the program, click on the "Debug" button (the bug icon) on the toolbar. CCS will build and download the firmware to the LaunchPad.

After it finishes, click on the green arrow on the right pane to execute the program.



(Section "5.6. Building and Running Your Project" in the CCS user's guide also describes the steps to build and run the program.) We should see the output displayed on the debug console in the lower part of the CCS window.



Lab Report Submission

1. Include a screenshot of the debug console showing the console tab and the outputs.
2. List the entire program in the appendix.

5. Exercise 3 Blink LED

In this exercise, we will import a small example project to blink an LED. Use the same method to import the project in the previous exercise. On the CCS menu bar, click on <Project><Import CCS Projects…>. If you use a Windows machine, you may go to C:\ti\simplelink_msp432p4_sdk_3_40_01_02\examples\nortos\MSP_EXP432P401R\driverlib\gpio_toggle_output. Click on the ccs folder, then click on the "Select Folder" button. Then click on the "Finish" button to import the project. Build and execute the program. You should see LED1 on the board blinking very rapidly.

On the source file *gpio_toggle_output.c*, scroll to the end to find the *while* loop. Change the *for* loop count to 50000. Click on the Stop button (the red square icon) on the toolbar. Click on Debug to rebuild and the green button to run. You should see the blinking is much slower.

You have successfully used the TI MSP432 Peripheral Driver Library (DriverLib) to build a small application!

Take a short video clip of the boarding blinking the LED. Place the video file where it can be accessed remotely.

Lab Report Submission

1. Include a picture showing the LED lit.
2. Include a link to the video clip. Do not submit the video file to Canvas. Make sure the video file can be opened by just clicking on the link.
3. List the entire program in the appendix.

6. Exercise 4 Access MCU Info

In this exercise, we will find out some device information about the MCU.

There is some specific information about the MCU that is stored in the device's non-volatile memory. They are contained in a table called Device Descriptor Table. The information is organized using a tag-length-value (TLV) structure. The table begins with the "TLV checksum" and ends with the "TLV End Word." Between those two values, there are a number of data blocks. Each data block begins with a tag value followed by the number of data items (length) stored in the block. The data items in the block follow the length value. With such data structure, we can traverse the entire table from the beginning to the end.

Create a new project (by importing the example *empty* project) to read and print the information to the debug console. Consult the MCU datasheet for the starting address of the Device Descriptor Table and the table structure format. Print the contents line by line. The first line shall be the TLV checksum. The subsequent lines are the data blocks, one line per block. A line shall begin with the tag value, followed by the length value and the data values, separated by a space character. The last line shall be the TLV End Word. Note that each value in the table is stored as a 32-bit word. Use the library function *printf* to print the values in 8-digit hexadecimal format (just like the way they are shown in the table on the datasheet). Since we are going to use *printf*, make sure to insert the following line in the *#include* section of the main C file.

```
#include <stdio.h>
```

You are not allowed to use any library function, nor any predefined data structure in DriverLib, to access the information. With the physical memory address you found, create a pointer to read different fields in the table. Use this pointer to access all the required data items; you cannot hardcode the addresses for individual data items. Be sure to properly declare the pointer so that it can be used to access a 32-bit word; use data type of specific bit width. You cannot hard-code the number of blocks in the table. Use a *for* loop to move from one block to the next until you encounter the TLV End Word, which is a special tag value.

Note that some values in the table are constants, i.e., they are the same in all devices manufactured. Others may vary from device to device, so they are shown as "Per unit" in the table on the datasheet. So, we can use the constant values to check if the program works properly. Also, the TLV checksum is typically a very large number, so we should expect to see a large number printed for it.

Lab Report Submission

1. List the code snippet that reads and prints all the information. Do not provide a screenshot to show the code; copy and paste the code as text.
2. List the outputs on the debug console. Do not take a screenshot; copy and paste the text outputs.
3. List the entire program in the appendix.

7. Submission Requirements

Write a formal report that contains at least what are required to be submitted in the exercises. Submit the report in PDF to Canvas by the deadline. Include program source code in your report. Do not use screenshots to show your codes. In the report body, if requested, list the relevant codes or screenshots only for the exercise you are discussing. Put the entire program listing in the appendix. Use single-column format for the report.

In your report, show the debug console outputs (when requested) in text format; copy and paste them to your report. In general, do not use screenshot to show any text outputs or data; otherwise, points will be deducted.

When presenting a screenshot in the report, only present the relevant information. You may need to crop them from the initial screenshot. Please use proper scale so that the contents can be read easily.

Your report should have several sections. The first section is the information header that contains at least your name, course name and lab assignment number. Each exercise should have its own section where you discuss the results of the exercise. At the end, if requested, there should be the appendix that contains the complete source code for each exercise. The source code may be used to reproduce your reported results during grading. So, make sure they do produce the results you reported.

8. Grading Policy

The lab's grade is primarily determined by the report. If you miss the submission deadline for the report, you will get zero point for the lab assignment.

If your program (listed in the appendix) fails to compile, you will get zero point for the exercise.

If your program's outputs do not match what you reported, you will get zero point for the exercise.

Points may be deducted for incorrect statements, typos, non-readable texts on screenshots, lack of description on the graphs/pictures shown, etc. So, proofread your report before submission.

Pay attention to the required program behaviors described in the assignment. Points will be deducted for not producing the required outcomes.