Lab 0. Digital Logic Implemented on a Microcontroller

All students do Lab0 by themselves (no partner for Lab 0)

Preparation

Read Chapters 1 and 2 of the book
Read Sections 3.1, 3.2, 3.3.1-3.3.5, 3.3.9, 3.3.10, 4.1.2 and 4.2 of the book
Step0: Install and run Keil uVision on your personal computer
Step1: Download and run the installer for EE319K
http://users.ece.utexas.edu/~valvano/Volume1/EE319K_Install.exe

Purpose

The general purpose of this laboratory is to familiarize you with the software development steps using the **uVision** simulator. Starting with Lab 2, we will use uVision for both simulation and debugging on the real board, but for this lab, we will use just the simulator. You will learn how to perform digital input/output on parallel ports of the LM4F120/TM4C123. Software skills you will learn include port initialization, logic operations, and unconditional branching.

System Requirements

The objective of this system is to implement a car door signal system.
Hardware connections: Inputs are negative logic; output is positive logic

- PF0 is left-door input sensor (1 means door is open, 0 means door is closed)
- PF4 is right-door input sensor (1 means door is open, 0 means door is closed)
- PF3 is Safe (Green) LED signal - ON when both doors are closed, otherwise OFF
- PF1 is Unsafe (Red) LED signal - ON when either (or both) doors are open, otherwise OFF

The specific operation of this system

- Turn Unsafe LED signal ON if any or both doors are open, otherwise turn the Safe LED signal ON
- Only one of the LEDs should be on at any given time.

Procedure

The basic approach to this lab will be to develop and debug your system using the simulator.

Part a - Verify Keil Project for Lab0 is present and runs

To work on Lab 0, perform these tasks. Find a place on your hard drive to save all your LM4F120/TM4C123 software. In Figure 0.1 it is called **EE319Kware**, created when you install the .exe.

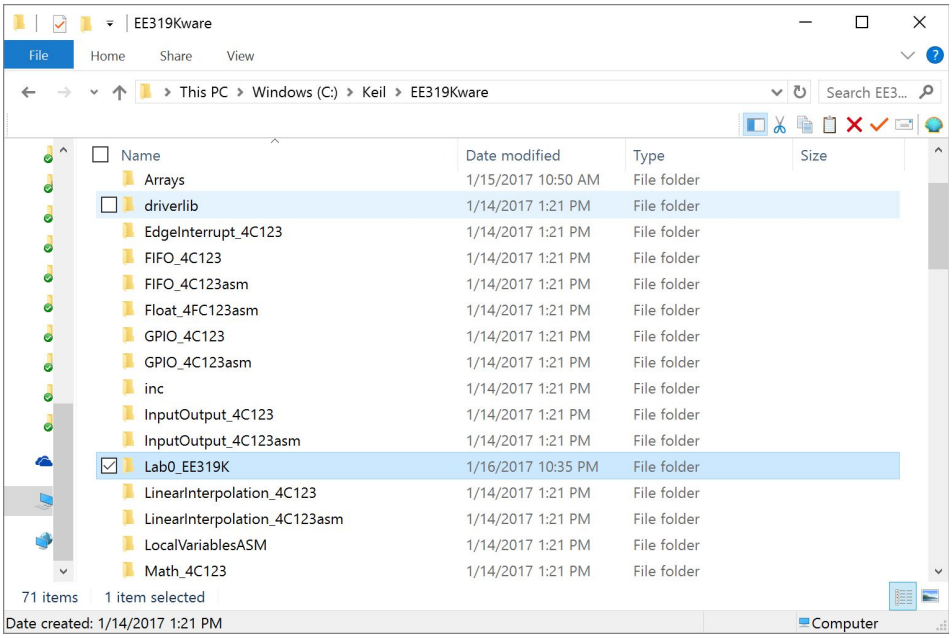

Figure 0.1. Directory structure with your Lab0.

It is important for the directory structure to look like Figure 0.1. Download and unzip the starter configuration from
http://users.ece.utexas.edu/~valvano/Volume1/EE319K_Install.exe
into this location. We will place our projects in folders like my Lab 0_EE319K.

Begin with the **Lab0_EE319K** project in the folder **EE319Kware**. Either double click the **uvproj** file or open the project from within uVision. Make sure you can compile it and run on the simulator. Please contact your TA if the starter project does not compile or run on the simulator. **Startup.s** contains assembly source code to define the stack, reset vector, and interrupt vectors. All projects in this class will include this file, and you should not need to make any changes to the **Startup.s** file. **main.s** will contain your assembly source code for this lab. You will edit the **main.s** file.

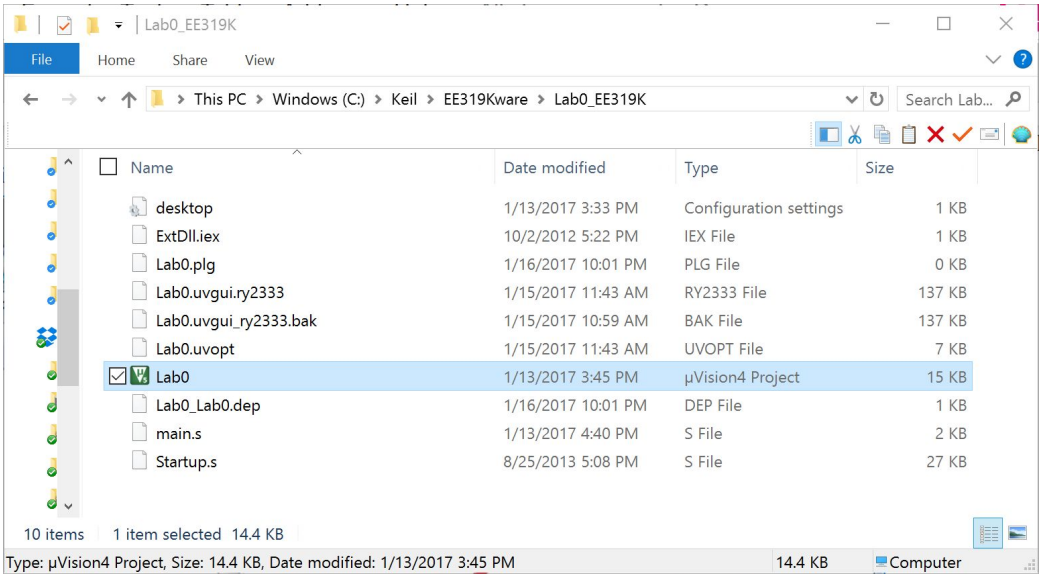

*Figure 0.2. Start Keil by opening the **Lab0.uvproj** file.*

You should rename the Lab1 starter folder to include your EIDs. Add your names and the most recent date to the comments at the top of main.s. This code shows the basic template for the first few labs. You will not need global variables in this lab.

```
      THUMB
      AREA  DATA, ALIGN=2
;global variables go here
      ALIGN
      AREA  |.text|, CODE, READONLY, ALIGN=2
      EXPORT  Start
Start
;initialization code goes here
loop
;the body of the code goes here
      B   loop
      ALIGN
      END
```

Program 0 Assembly language template.

To run the Lab 0 simulator, you must check two things. First, execute Project->Options and select the Debug tab. The debug parameter field must include **-dLauchPadDLL**. Second, the LauchPadDLL.dll file must be present in your Keil\ARM\BIN folder (put there by the installer).

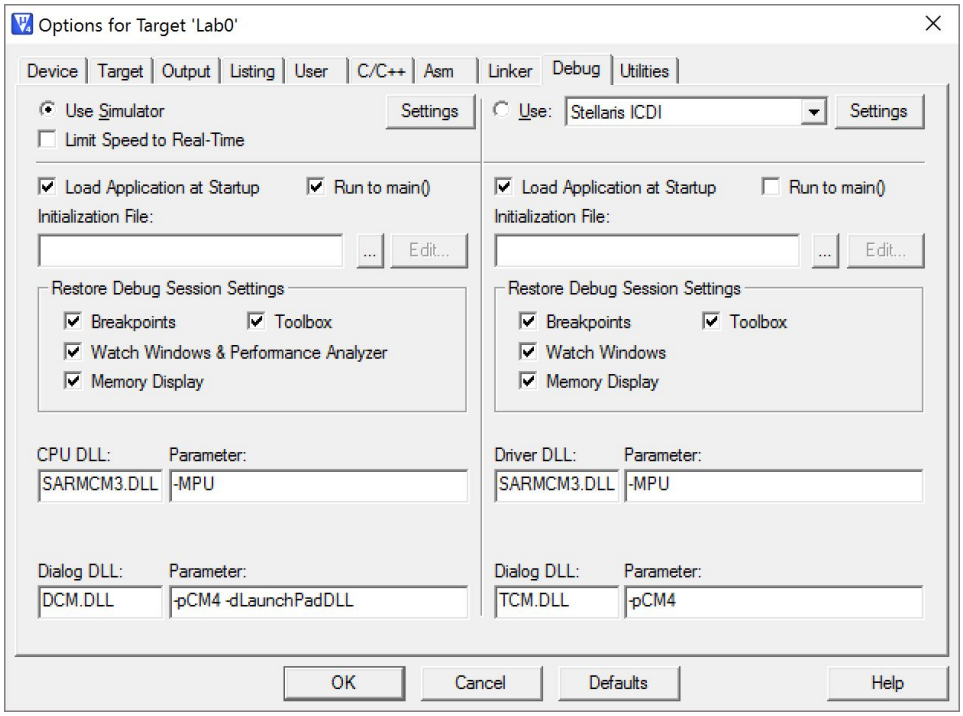

Figure 0.3. Debug the software using the simulator (DCM.DLL -pCM4 -dLauchPadDLL).

Part b - Draw Flow Chart

Write a flowchart for this program. We expect 5 to 15 symbols in the flow chart. A flowchart describes the algorithm used to solve the problem and is a visual equivalent of pseudocode. See Example 1.7.1 in book for a sample flowchart.

Part c - Write Pseudocode

Write pseudocode for this program. We expect 5 to 10 steps in the pseudo code. You may use any syntax you wish, but the algorithm should be clear. See Example 1.17.1 in book for an instance of what pseudocode ought to look like. Note, pseudocode ought to embody the algorithm and therefore be language blind. The same pseudocode can serve as an aid to writing the solution out in either assembly or C (or any other language).

Part d - Write Assembly

You will write assembly code that inputs from PF0, PF4 and outputs to PF1 and PF3. The address definitions for Port F are available in **lm4f120.s** or **tm4c123gh6pm.s** file (listed below) and already placed in your starter file main.s:

```
GPIO_PORTF_DATA_R  EQU 0x400253FC
GPIO_PORTF_DIR_R   EQU 0x40025400
GPIO_PORTF_AFSEL_R EQU 0x40025420
GPIO_PORTF_PUR_R   EQU 0x40025510
GPIO_PORTF_DEN_R   EQU 0x4002551C
GPIO_PORTF_LOCK_R  EQU 0x40025520
```

```
GPIO_PORTF_CR_REQU 0x40025524
GPIO_PORTF_AMSEL_R EQU 0x40025528
GPIO_PORTF_PCTL_R  EQU 0x4002552C
GPIO_LOCK_KEY  EQU 0x4C4F434B  ; Unlocks the GPIO_CR register
SYSCTL_RCGCGPIO_R  EQU   0x400FE608
```

The opening comments include: file name, overall objectives, hardware connections, specific functions, author name, TA, and date. The **equ** pseudo-op is used to define port addresses. Global variables are declared in RAM, and the main program is placed in EEPROM. The 32-bit contents at ROM address 0x00000004 define where the computer will begin execution after a power is turned on or after the reset button is pressed.

```
;****************** main.s ***************
; Program written by: Your Name
; Date Created: 1/15/2017
; Last Modified: 1/15/2017
; Brief description of the program
; The objective of this system is to implement a Car door signal system
; Hardware connections: Inputs are negative logic; output is positive logic
;  PF0 is right-door input sensor (1 means door is open, 0 means door is closed)
;  PF4 is left-door input sensor (1 means door is open, 0 means door is closed)
;  PF3 is Safe (Green) LED signal - ON when both doors are closed, otherwise OFF
;  PF1 is Unsafe (Red) LED signal - ON when either (or both) doors are open, otherwise OFF
; The specific operation of this system
;   Turn Unsafe LED signal ON if any or both doors are open, otherwise turn the Safe LED signal
ON
;   Only one of the two LEDs must be ON at any time.
; NOTE: Do not use any conditional branches in your solution.
;       We want you to think of the solution in terms of logical and shift operations
```

Program 0.2. Required comments at the top of every file.

To interact with the I/O during simulation, **make sure that View->Periodic Window Update is checked or the simulator will not update!** Then, execute the **Peripherals->TExaS Port F** command. When running the simulator, we check and uncheck bits in the I/O Port box to change input pins (1) (2) and (3). We observe output pin in the window (4). You can also see the other DIR AFSEL DEN and PUR registers.

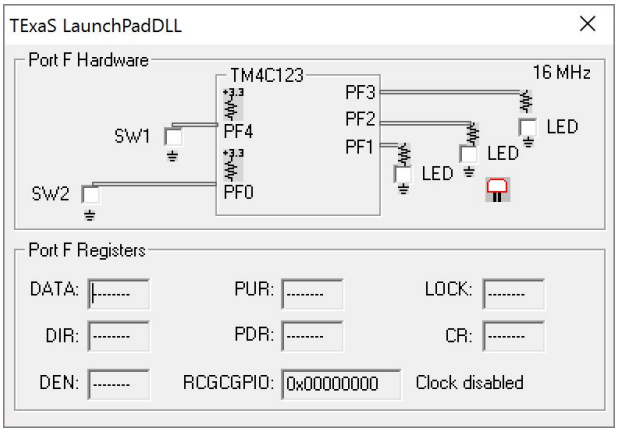

Figure 0.4. In simulation mode, we interact with virtual hardware.

Demonstration

During the demonstration, you will be asked to run your program to verify proper operation. You should be able to single step your program and explain what your program is doing and why. You need to know how to set and clear breakpoints. You should know how to visualize Port F input/output in the simulator.

Deliverables

Items 1-4 are one pdf file uploaded to Canvas, have this file open during demo.

0. Lab 0 grading sheet. You can print it yourself or pick up a copy in lab. You fill out the information at the top.
1. Flowchart of the system
2. Pseudo code for the algorithm
3. Assembly source code of your final **main.s** program
4. Two screenshots of the Port F window, one showing the green LED on (and red LED off) and the other showing the red LED on (and green LED off).
5. Optional Feedback : http://goo.gl/forms/rBsP9NTxSy

FAQ

The list of FAQ below are populated from Piazza over the semesters (thanks to the contributions of all past TAs and students). More questions may be posted so please check back regularly.

1. Should the program keep checking for inputs (doors) and update the LEDs continuously?
   Your program should loop, so yes.
2. Our program works as expected when stepping through but when it is run through it does not. What could be causing this? What is an effective way to debug when our debugging method says that the program is working fine but the actual running of the program says otherwise?
   First check if the “Periodic Window Update” under the “View" tab is on when you are in debugging mode. Also, some run-time errors can occur when setting up the clock register which don't appear when single stepping. Look over and ensure you are writing to the correct register and only affecting those bits required to activate Port E, as well as give enough time for it to start up.
3. What is the best way to include the source code for the main.s into the PDF?
   One way to include the source code for the main.s into the PDF is to copy / paste the code into a Word document and then combine that with all of your other deliverables, depending on the lab, and then converting that file to a PDF. Please do not only include a screenshot.
4. For the flowchart and pseudocode, do we need to start at the very beginning with all the initialization tasks, or just at the actual logic for locking and unlocking the lock?
   Your flowchart should cover the entire program that you write. Therefore, initialization should be included. How you represent initialization is up to you.
5. Are we allowed to branch?
   Only unconditional branches are allowed in this lab. You are required to implement this lab using only Boolean logic such as AND, OR, and shifts etc.
6. Do both members of our lab group need to turn in a pdf, or do we just turn in one for the two of us?
   Please both submit you pdf deliverables on Canvas.
7. When I try to run the debugger, I get: Error: Could not load file 'C:\Keil\EE319Kware\Lab0_EE319K\Lab0.axf'. Debugger aborted! What should I do?

You most likely forgot to build your project before running the debugger. If that doesn't solve the problem, try running Keil as an administrator.

8. When I try to build, it gives the errors: Build target 'Lab0', error - cannot create command input file '.\startup._ia', error - cannot create command input file '.\main._ia', Target not created! What should I do?
   Try running Keil as an administrator
9. Where can I find the addresses for the different ports?
   Register definitions for the microcontroller may be found here:
   http://users.ece.utexas.edu/~valvano/arm/tm4c123gh6pm.s
10. I edited my code but nothing changes when I re-run it in the debugger!
    If you made changes to your code "in debug mode", you most likely have not re-built your project and therefore your debugger is running the old version of your code. Exit out of debug mode and rebuild your code for the changes to take effect.
11. I cannot find the interactive UI simulator with switches and LEDs when I am in debugging mode!
    Enable "TExaS Port x" under the "Peripherals" tab. If you do not see the Ports that you are expecting, you most likely do not have the correct DLL file listed in "Debug" tab in "Options for Target" under "Project" tab.
12. Keil tells me I have tons of build errors but nothing seems to be wrong with my assembly code.
    Make sure you instructions are indented. Only the labels are aligned all the way to the left.