Lab 1. odd-bit detection system

Preparation

Read Chapter 1 of the book (second edition), and lecture 3 slides.
All students do Lab 1 and 2 by themselves to understand the software and the modeling better (no partner for this lab). For Lab 3 and later, you will work with a partner. However, Lab Reports will be always individually prepared and submitted.

Purpose

The general purpose of this laboratory is to familiarize you with the software development steps using the **uVision** simulator. Starting with Lab 3, we will use uVision for both simulation and debugging on the real board, but for Labs 1 and 2, we will just use just the simulator. You will learn how to perform digital input/output on parallel ports of the TM4C123. Software skills you will learn include port initialization, logic operations, and unconditional branching. Do not use any conditional branches in your solution. We want you to think of the solution in terms of logical and shift operations.

System Requirements

The objective of this system is to implement an odd-bit detection system (If there are odd number of input “1”, the output will be 1). There are three bits of inputs and one bit of output. The output is in positive logic: outputing a 1 will turn on the LED, outputing a 0 will turn off the LED. Inputs are negative logic: meaning if the switch not pressed the input is 1, if the switch is pressed the input is 0.

- PE0 is an input
- PE1 is an input
- PE2 is an input
- PE3 is the output

The specific operation of this system

- Initialize Port E to make PE0,PE1,PE2 inputs and PE3 an output
- Make the output 1 if there is an odd number of 1's at the inputs, otherwise make the output 0.
- Over and over, read the inputs, calculate the result and set the output

The input/output specification refers to the input, not the switch. The following table illustrates the expected behavior relative to output PE3 as a function of inputs PE0,PE1,PE2

PE2	PE1	PE0	PE3	
0	0	0	0	even number of 1’s
0	0	1	1	odd number of 1’s
0	1	0	1	odd number of 1’s
0	1	1	0	even number of 1’s
1	0	0	1	odd number of 1’s

1	0	1	0	even number of 1's
1	1	0	0	even number of 1's
1	1	1	1	odd number of 1's

You could also consider the specification the LED output as a function of the switch input. The following table illustrates the expected behavior relative to LED output as a function of switch presses

Sw2	Sw1	Sw0	LED	
press	press	press	Off	odd number switches pressed
press	press	not	On	even number switches pressed
press	not	press	On	evan number switches pressed
press	not	not	Off	odd number switches pressed
not	press	press	On	even number switches pressed
not	press	not	Off	odd number switches pressed
not	not	press	Off	odd number switches pressed
not	not	not	On	even number switches pressed

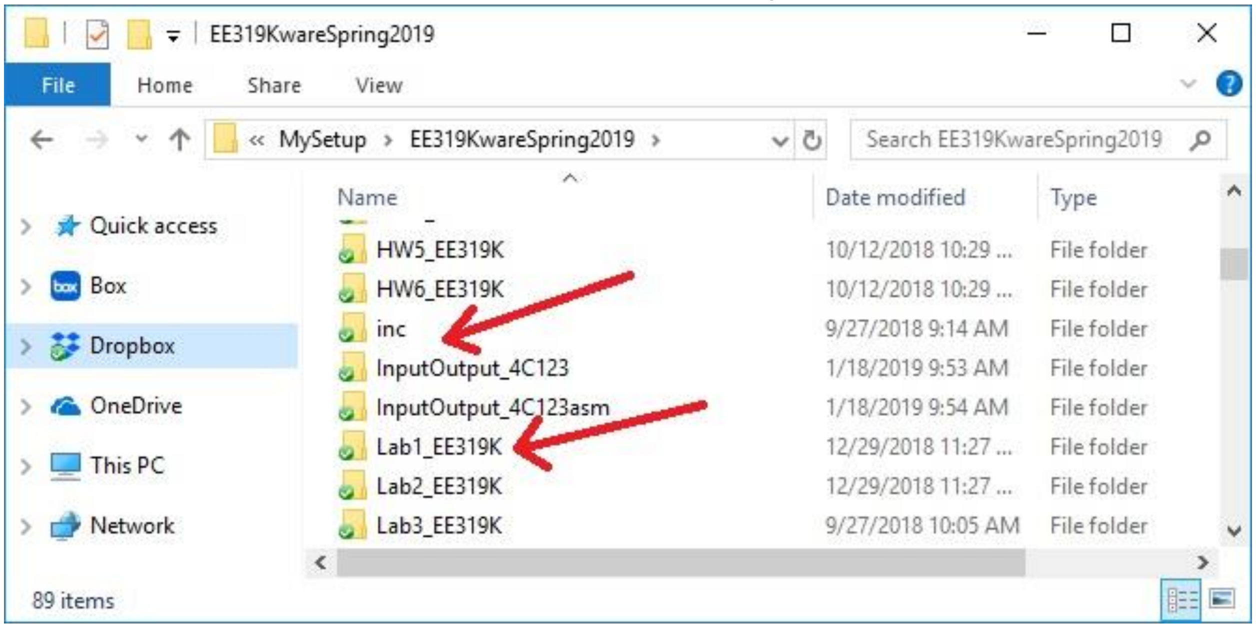
where press means the switch is pressed and not means the switch is not pressed.

Procedure

The basic approach to this lab will be to develop and debug your system using the simulator. There is no hardware required for Lab 1.

Part a - Verify Keil Project for Lab1 is present and runs

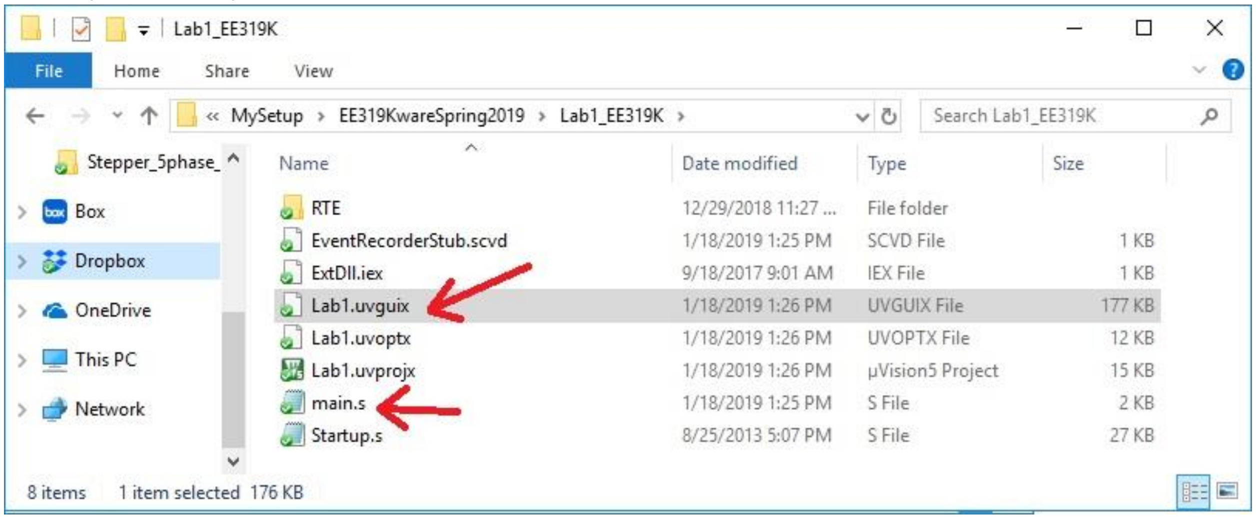
To work on Lab 1, perform these tasks. Find a place on your hard drive to save all your TM4C123 software. In Figure 1 it is called **EE319KwareSpring2019**, created when you install the .exe. Download and unzip the starter configuration from http://users.ece.utexas.edu/~valvano/Volume1/EE319K_Install.exe into this location. Notice the solutions to the labs will be folders that begin with "Lab".



Name	Date modified	Type
HW5_EE319K	10/12/2018 10:29 ...	File folder
HW6_EE319K	10/12/2018 10:29 ...	File folder
inc	9/27/2018 9:14 AM	File folder
InputOutput_4C123	1/18/2019 9:53 AM	File folder
InputOutput_4C123asm	1/18/2019 9:54 AM	File folder
Lab1_EE319K	12/29/2018 11:27 ...	File folder
Lab2_EE319K	12/29/2018 11:27 ...	File folder
Lab3_EE319K	9/27/2018 10:05 AM	File folder

Figure 1. Directory structure with your Lab1.

It is important for the directory structure to look like Figure 1. Notice the directory relationship between the lab folders and the **inc** (include) folder. Begin with the **Lab1_EE319K** project in the folder **EE319KwareSpring2019**. Either double click the **uvprojx** file or open the project from within uVision. Make sure you can compile it and 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.



Name	Date modified	Type	Size
RTE	12/29/2018 11:27 ...	File folder	
EventRecorderStub.scvd	1/18/2019 1:25 PM	SCVD File	1 KB
ExtDll.iex	9/18/2017 9:01 AM	IEX File	1 KB
Lab1.uvguix	1/18/2019 1:26 PM	UVGUIX File	177 KB
Lab1.uvoptx	1/18/2019 1:26 PM	UVOPTX File	12 KB
Lab1.uvprojx	1/18/2019 1:26 PM	µVision5 Project	15 KB
main.s	1/18/2019 1:25 PM	S File	2 KB
Startup.s	8/25/2013 5:07 PM	S File	27 KB

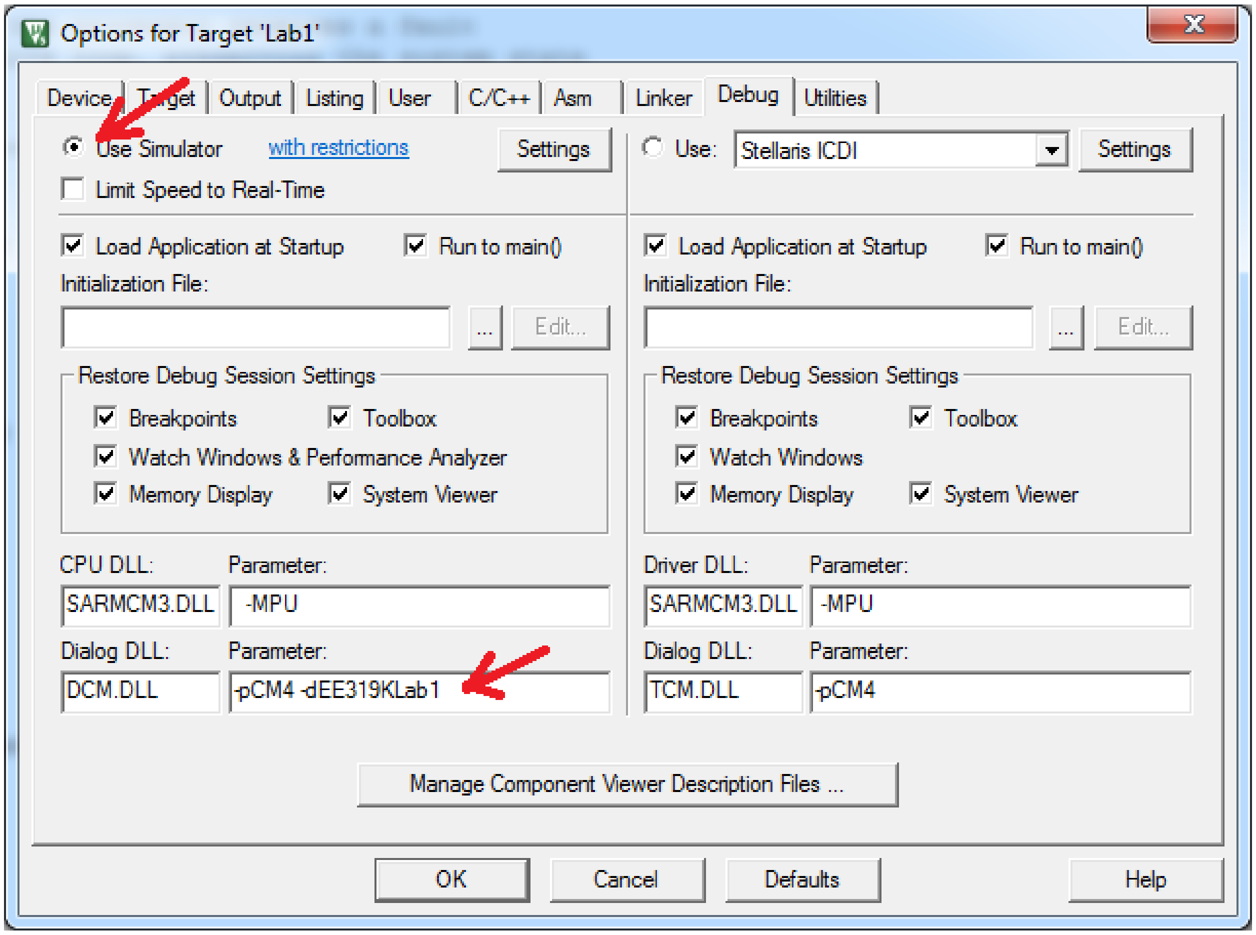
*Figure 2. Start Keil by opening the **Lab1.uvprojx** file.*

You should rename the Lab1 starter folder to include your EID. 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 lab 1.

```
      THUMB
      AREA    DATA, ALIGN=2
;global variables go here
      ALIGN
      AREA    |.text|, CODE, READONLY, ALIGN=2
      EXPORT  Start
Start
   ; initialization code goes here
loop
   ; put your main engine here
    B   loop
; put any subroutines here
      ALIGN        ; make sure the end of this section is aligned
      END          ; end of file
```

Program 1 Assembly language template.

To run the Lab 1 simulator, you must check two things. First, execute Project->Options and select the Debug tab. The debug parameter field must include **-dEE319KLab1**. Second, the **EE319KLab1.dll** file must be present in your Keil\ARM\BIN folder (the EE319K DLLs should have been put there by the installer).


Options for Target 'Lab1'

Device | Target | Output | Listing | User | C/C++ | Asm | Linker | Debug | Utilities

(●) Use Simulator with restrictions Settings
[ ] Limit Speed to Real-Time

[✓] Load Application at Startup [✓] Run to main()
Initialization File:

Restore Debug Session Settings
[✓] Breakpoints [✓] Toolbox
[✓] Watch Windows & Performance Analyzer
[✓] Memory Display [✓] System Viewer

CPU DLL:	Parameter:
SARMCM3.DLL	-MPU

Dialog DLL:	Parameter:
DCM.DLL	-pCM4 -dEE319KLab1

( ) Use: Stellaris ICDI Settings

[✓] Load Application at Startup [✓] Run to main()
Initialization File:

Restore Debug Session Settings
[✓] Breakpoints [✓] Toolbox
[✓] Watch Windows
[✓] Memory Display [✓] System Viewer

Driver DLL:	Parameter:
SARMCM3.DLL	-MPU

Dialog DLL:	Parameter:
TCM.DLL	-pCM4

Figure 3. Debug the software using the simulator (DCM.DLL -pCM4 -dEE319KLab1).

Part b - Draw Flowchart

Write a flowchart for this program. We expect 5 to 15 symbols in the flowchart. A flowchart describes the algorithm used to solve the problem and is a visual equivalent of pseudocode. See Section 1.7 in the book for example flowcharts.

Part c - Write Pseudocode

Write pseudocode for this program. We expect 5 to 10 steps in the pseudocode. You may use any syntax you wish, but the algorithm should be clear. See Example 1.17.1 in the book (Section 1.17) 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 PE2, PE1, PE0 and outputs to PE3. The address definitions for Port E are listed below, and these are placed in the starter file main.s:

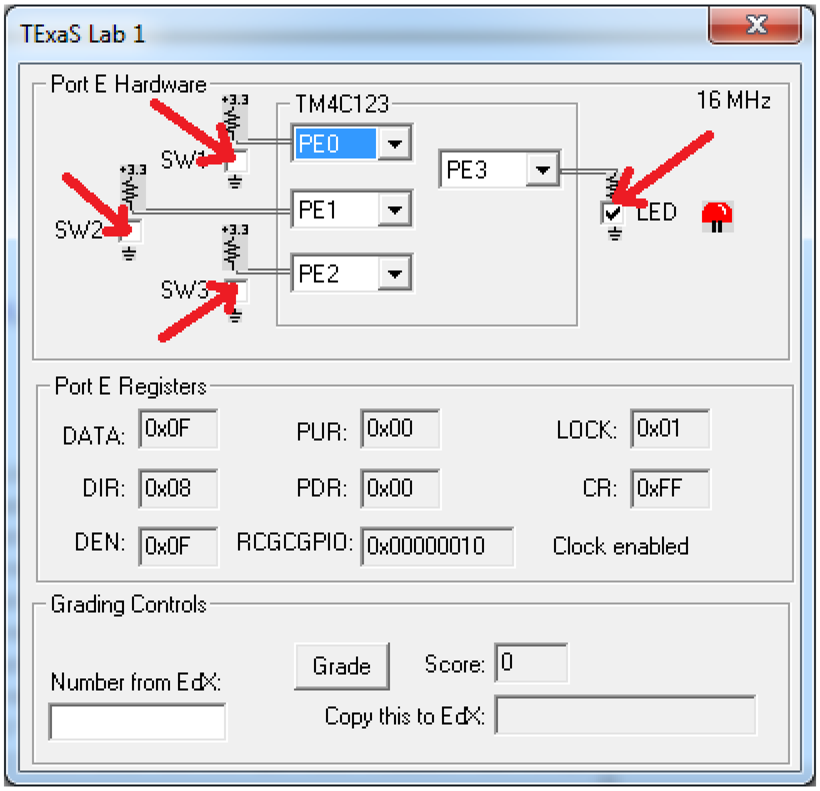
```
GPIO_PORTE_DATA_R  EQU 0x400243FC
GPIO_PORTE_DIR_R   EQU 0x40024400
```

```
GPIO_PORTE_DEN_R   EQU 0x4002451C
SYSCTL_RCGCGPIO_R  EQU 0x400FE608
```

The opening comments include: filename, overall objectives, hardware connections, specific functions, author name, 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 ***************
; Author: Place your name here
; Date Created: 1/15/2018
; Last Modified: 1/18/2019
; Brief description of the program: Spring 2019 Lab1
; The objective of this system is to implement odd-bit counting system
; Hardware connections:
;  Output is positive logic, 1 turns on the LED, 0 turns off the LED
;  Inputs are negative logic, meaning switch not pressed is 1, pressed is 0
;       PE0 is an input
;       PE1 is an input
;       PE2 is an input
;       PE3 is the output
; Overall goal:
;   Make the output 1 if there is an odd number of 1's at the inputs,
;       otherwise make the output 0
; The specific operation of this system
;   Initialize Port E to make PE0,PE1,PE2 inputs and PE3 an output
;   Over and over, read the inputs, calculate the result and set the output
; 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
GPIO_PORTE_DATA_R  EQU 0x400243FC
GPIO_PORTE_DIR_R   EQU 0x40024400
GPIO_PORTE_DEN_R   EQU 0x4002451C
SYSCTL_RCGCGPIO_R  EQU 0x400FE608
       THUMB
       AREA   DATA, ALIGN=2
;global variables go here
       ALIGN
       AREA   |.text|, CODE, READONLY, ALIGN=2
       EXPORT  Start
Start
; code to execute once at start goes here
loop
; code that runs over and over goes here
       B      loop
       ALIGN          ; make sure the end of this section is aligned
       END            ; end of file
```
Program 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 E** command. When running the simulator, we check and uncheck bits in the I/O Port box to change input pins. We observe output pin in the window. You can also see the other registers, such as DIR DEN and RCGCGPIO.



Port E Registers

DATA: 0x0F PUR: 0x00 LOCK: 0x01
DIR: 0x08 PDR: 0x00 CR: 0xFF
DEN: 0x0F RCGCGPIO: 0x00000010 Clock enabled

Grading Controls

Number from EdX:
Score: 0
Copy this to EdX:

Figure 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 E input/output in the simulator.

Deliverables

Include items 1-4 into your lab report. Follow the lab report format explained for Lab-0.

1. Flowchart of the system
2. Pseudocode for the algorithm
3. Assembly source code of your final **main.s** program
4. Two screenshots of the Port E window, one showing the LED on and the other showing the LED off.

FAQ

Below you can find answers to the list of questions you might have during this experiment:

1. Should the program keep checking for inputs 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 a pdf file?
   One way to include the source code for the main.s into a pdf file 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, EOR, shift left and shift right.
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?
   Labs 1 and 2 are done individually. For Labs 3-10 both partners should submit the same pdf on Canvas.
7. When I try to run the debugger, I get: Error: Could not load file 'C:\Keil_5\EE319Kware\Lab1_EE319K\Lab1.axf'. Debugger aborted! What should I do?
   You most likely forgot to build your project before running the debugger. The other possibility is the code has a syntax error. 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 'Lab1', 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 can be found from tm4c123gh6pm.pdf or from the textbook.

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 E” 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.