California State University, Channel Islands (CSUCI)
Department of Computer Science

COMP-462: Embedded Systems
Lab Report
Fall 2019

Lab Number:	Lab 1
Lab Topic:	Odd Bit Detection System

Student Name:	Keith Skinner
Student ID:	002372111
Student Major:	Computer Science

I. Objectives

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.

II. Introduction

The objective of this system is to implement an odd-bit detection system(If there are an 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: outputting a 1 will turn on the LED, outputting a 0 will turn off theLED. Inputs are negative logic: meaning if the switch not pressed the input is 1, if the switch is pressed the input is 0.

III. 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.

A. Verify Keil Project for Lab1 is present and runs
   1. Download and unzip the starter configuration from http://users.ece.utexas.edu/~valvano/Volume1/EE319K_Install.exeinto this location. Notice the solutions to the labs will be folders that begin with "Lab".
   2. You should rename the Lab1 starter folder to include your EID. Add your name 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.
   3. 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.dllfile must be present in your Keil\ARM\BIN folder (the EE319K DLLs should have been put there by the installer).

B. Draw Flowchart
   1. 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.

C. Write Pseudocode

1. 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).

D. Write Assembly

1. ou 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:
   i. GPIO_PORTE_DATA_R EQU 0x400243
   ii. FCGPIO_PORTE_DIR_R EQU 0x40024400
   iii. GPIO_PORTE_DEN_R EQU 0x4002451
   iv. CSYSCTL_RCGCGPIO_R EQU 0x400FE608
2. The opening comments include: filename, overall objectives, hardware connections, specific functions, author name, and date. The equpseudo-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 power is turned on or after the reset button is pressed.

E. Deliverables

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

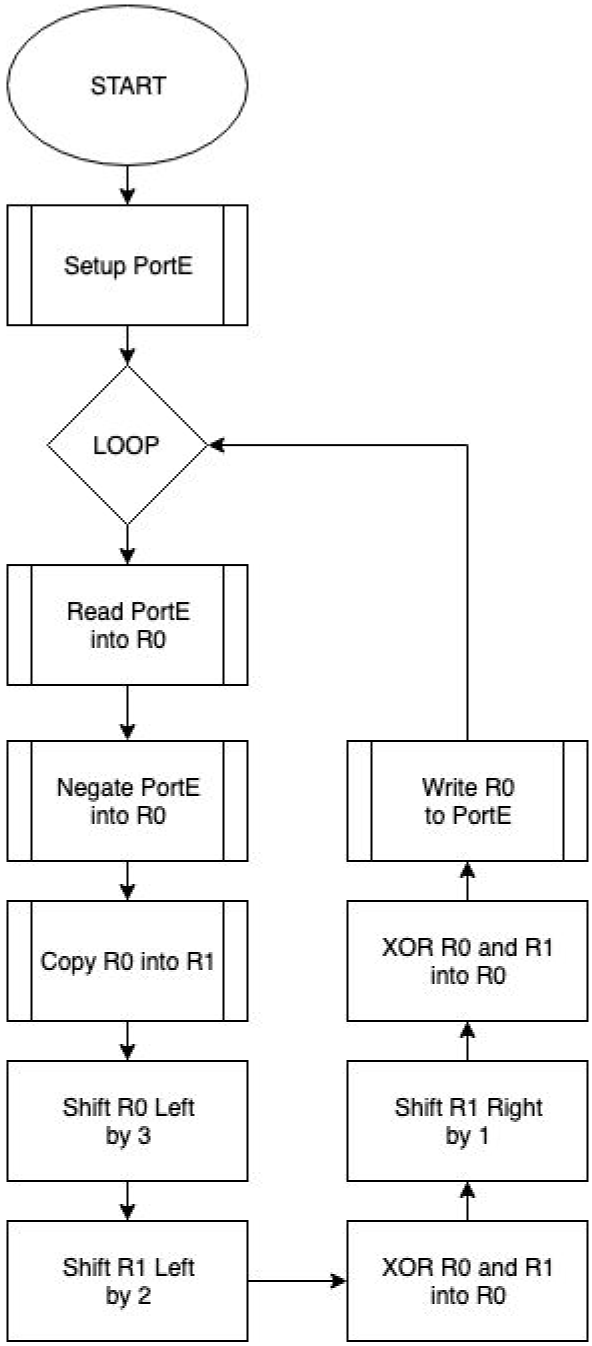
1. Flowchart for the System
2. Pseudocode for the Algorithm
3. Assembly source code of your final main.s program
4. Two screenshots of the Port E window, one showing the LED on and the other showing the LED off.

IV. Problems

There was no problems with the hardware, the simulation, the installation, nor the instructions.

V. Results

1. Flowchart for the System


START
Setup PortE
LOOP
Read PortE into R0
Negate PortE into R0
Copy R0 into R1
Shift R0 Left by 3
Shift R1 Left by 2
XOR R0 and R1 into R0
Shift R1 Right by 1
XOR R0 and R1 into R0
Write R0 to PortE

2. Pseudocode for the Algorithm

```
Odd Bit Detection Algorithm
Setup PortE
Loop forever:
      R0  ← Read(PortE)
      R0' ← Negate(R0)
      R1  ← R0
      R0  ← ShiftLeft(R0, 3)
      R1  ← ShiftLeft(R1, 1)
      R0  ← XOR(R0, R1)
      R1  ← ShiftRight(R1, 1)
      R0  ← XOR(R0, R1)
     Write(PortE, R0)
END[]
```

3. Assembly source code of your final main.s program

```
;****************** main.s ***************
; Program initially written by: Yerraballi and Valvano
; Author: Keith Skinner
; Date Created: 1/15/2018
; Last Modified: 9/12/2019
; 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
      ; Activate clock for port E
```

```
      LDR R1, =SYSCTL_RCGCGPIO_R      ; Grab clock location
      LDR R0, [R1]                    ; Grab clock value
      MOV R0, #0x10                   ; Bit 5 is for Port E
      STR R0, [R1]                    ; Store clock values turning on E
      NOP                             ; Wait part1
      NOP                             ; Wait part2
      ; Set up direction register
      LDR R1, =GPIO_PORTE_DIR_R       ; Grab direction location
      MOV R0, #0x8                    ; Sets pins PE0-2 as input and PE03 as
output
      STR R0, [R1]                    ; Store configuration
      ; Enable Port E digital port
      LDR R1, =GPIO_PORTE_DEN_R       ; Pointer to Digital/Analog Options
      MOV R0, #0xF                    ; Make Ports PE0-3 digital
      STR R0, [R1]                    ; Store configuration

loop
      ; Read PORTE into R0
      LDR R1, =GPIO_PORTE_DATA_R      ; Pointer to Port E data
      LDR R0, [R1]                    ; Read all of Port E into R0
      ;Meat of the work here
      MVN R0, R0                      ; invert R0 to make odd parity
      MOV R1, R0                      ; Copy R1 into R0
      LSL R0, R0, #3                  ; Shift R0 to the left 3 times
(PE0->PE3)
      EOR R0, R0, R1, LSL #2          ; XOR R0 and R1 into R0 after shifting
R1 x2
      EOR R0, R0, R1, LSL #1          ; XOR R0 and R1 into R0 after shifting
R1 x1

      ; Write R0 out to PORTE
      LDR R1, =GPIO_PORTE_DATA_R      ; Pointer to Port E data
      STR R0, [R1]                    ; Write all of R0 out to Port E

      B    loop                                             ; jump to top of
loop
  ALIGN                           ; make sure the end of this section is
aligned
  END                             ; end of file
```

4. Two screenshots of the Port E window, one showing the LED on and the other showing the LED off.

   Keil uVision takes 2 hours to download.
   Next lab forsure.