CSE 379

University at Buffalo

Keming Kuang (kemingku)

Eric Li (eli9)

April 3, 2018

Contents

1 Division of work

There are two main components to this lab, learning how to use timers and implementing strobing. As a team we did not split the lab into its components instead, we worked on the lab together. All the subroutines and interrupt handling were thought through and written together as a collaboration between Keming and Eric.

2 Description

2.1 Objective:

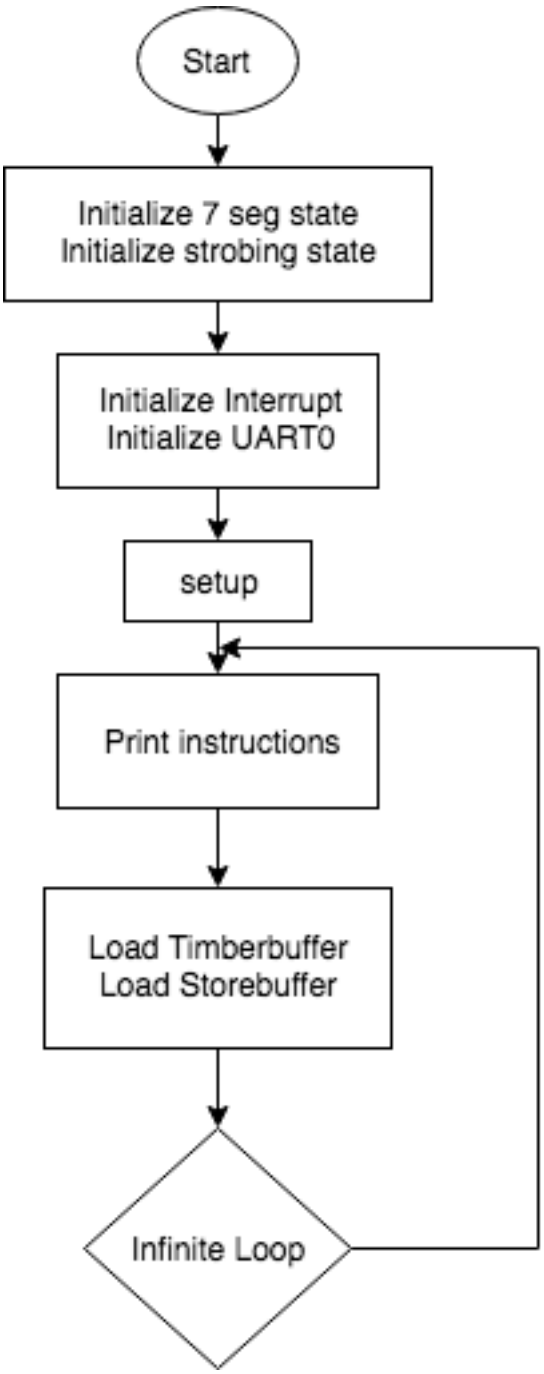
The objective of this lab assignment is mainly to learn how to use timers and timer interrupts. The application in this case, is strobing which allowed a four-digit hexadecimal number to be displayed on the four seven-segment displays. The main challenge of the lab was learning how to properly enable timer interrupts and properly configure them. The second challenge was the implementation of strobing. The culmination of these two aspects is a program which will display a four-digit hexadecimal number read from putty on the four seven-segment displays.

2.2 Debugging steps

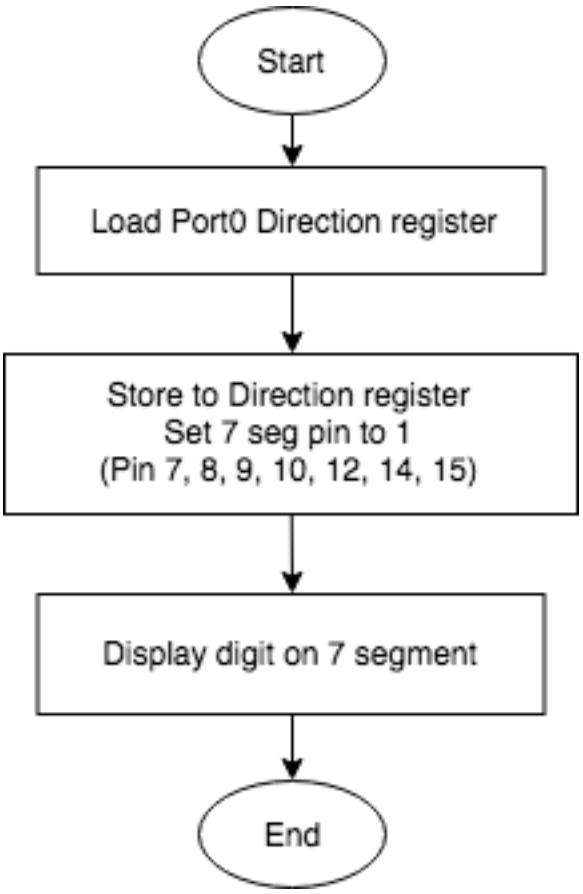
To debug the program, simply load it onto the arm board and instructions will be displayed. The function of the program is that it will read a 4-digit hexadecimal number entered via putty and display it on the seven-segment display. The number will only be displayed on the seven-segment after enter is pressed. The display should initially be off. It will turn on once the first number is entered. The external push button will turn the display on/off. Valid inputs are 0-9, A-F, and q. Once q is entered the program will terminate, if the display was originally on it will be turned off once program exits.

3 Flow Chart

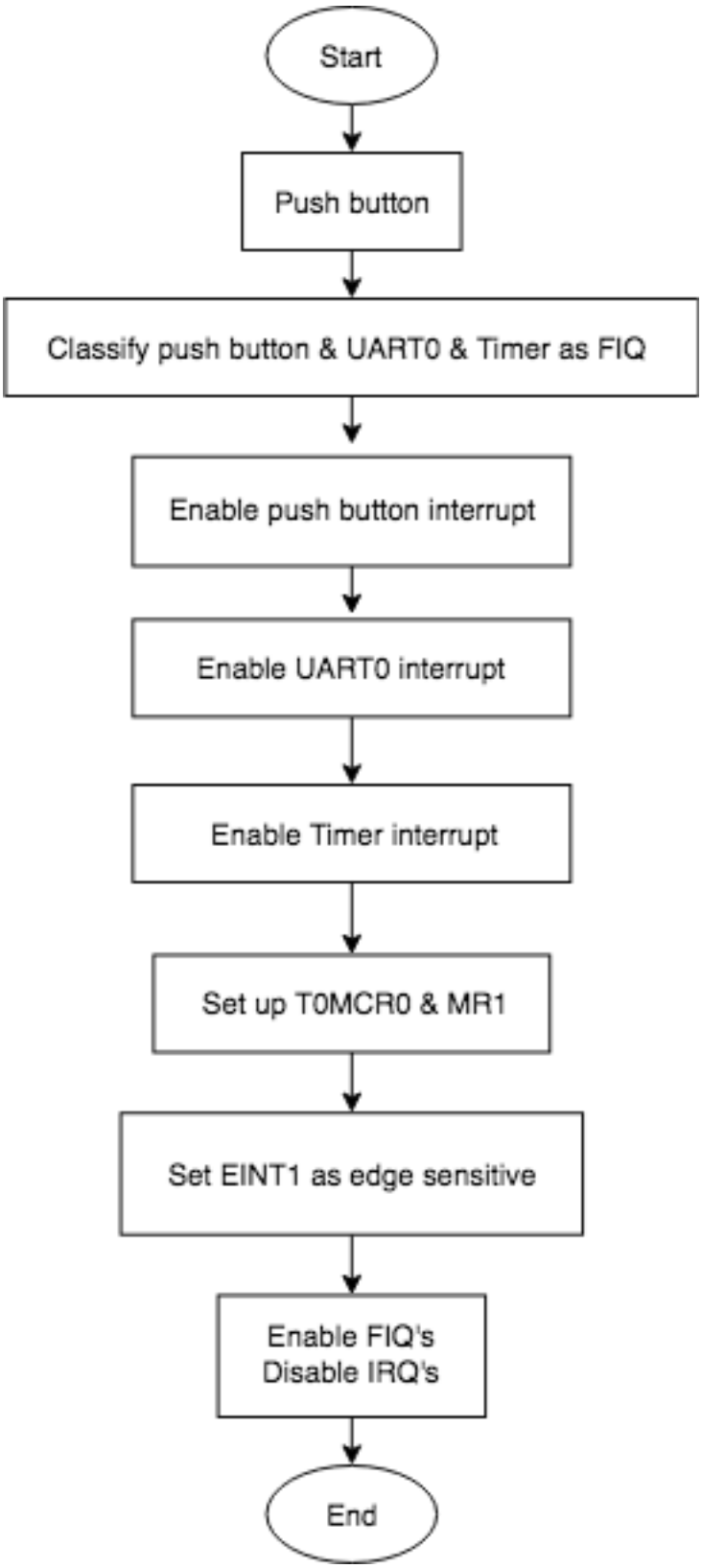
3.1 Main code



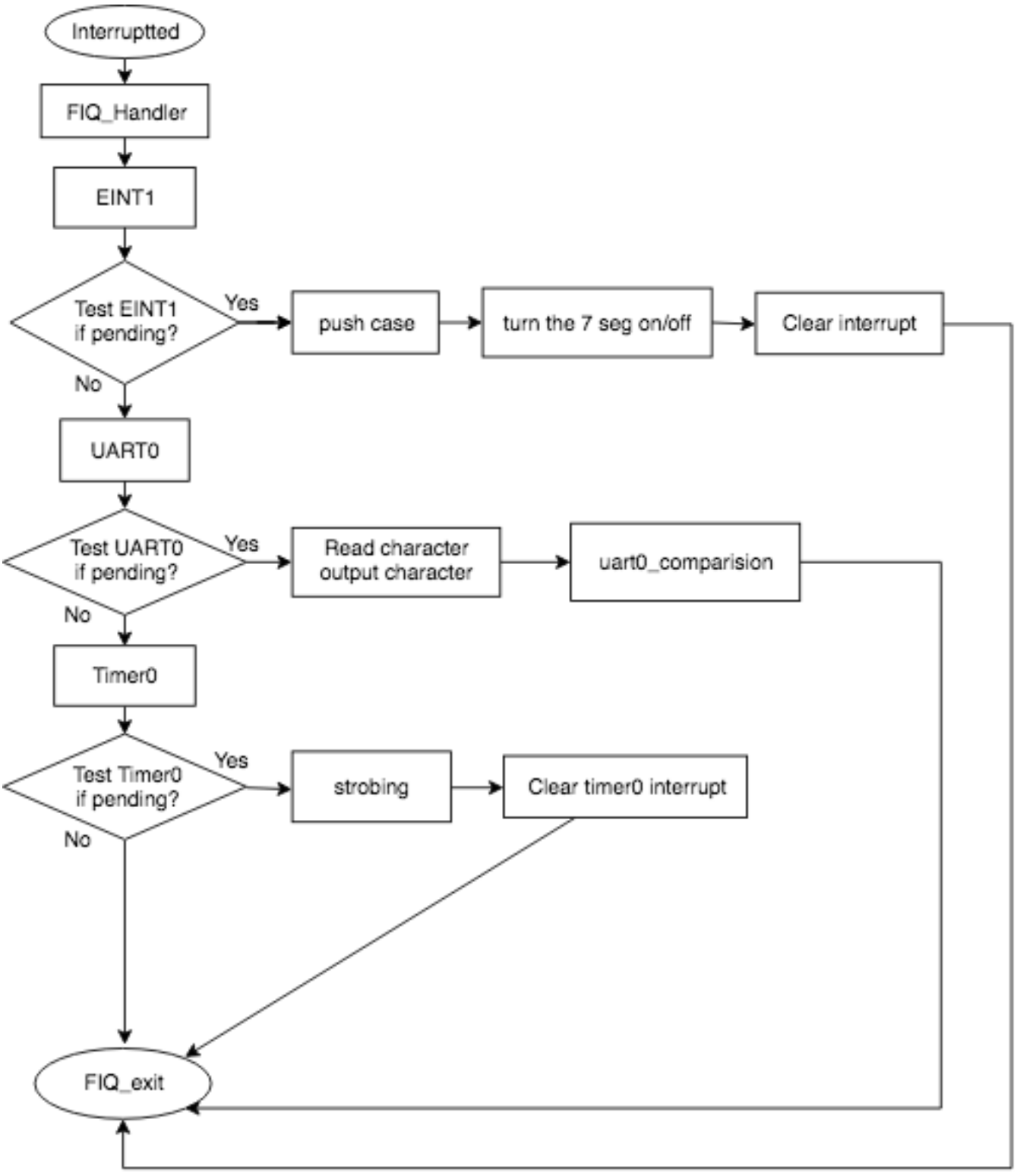
3.2 setup


Start
Load Port0 Direction register
Store to Direction register
Set 7 seg pin to 1
(Pin 7, 8, 9, 10, 12, 14, 15)
Display digit on 7 segment
End

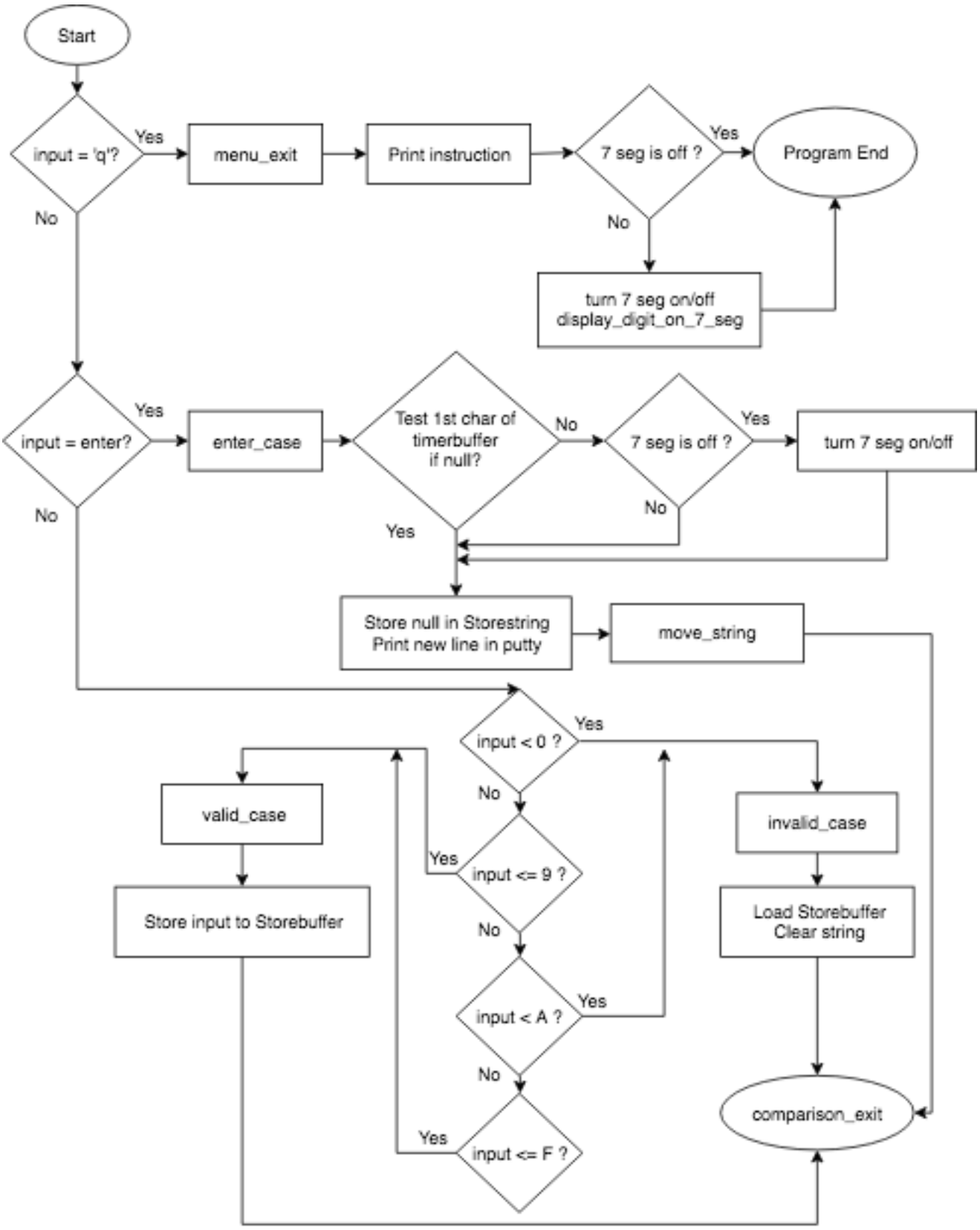
3.3 initialize interrupt



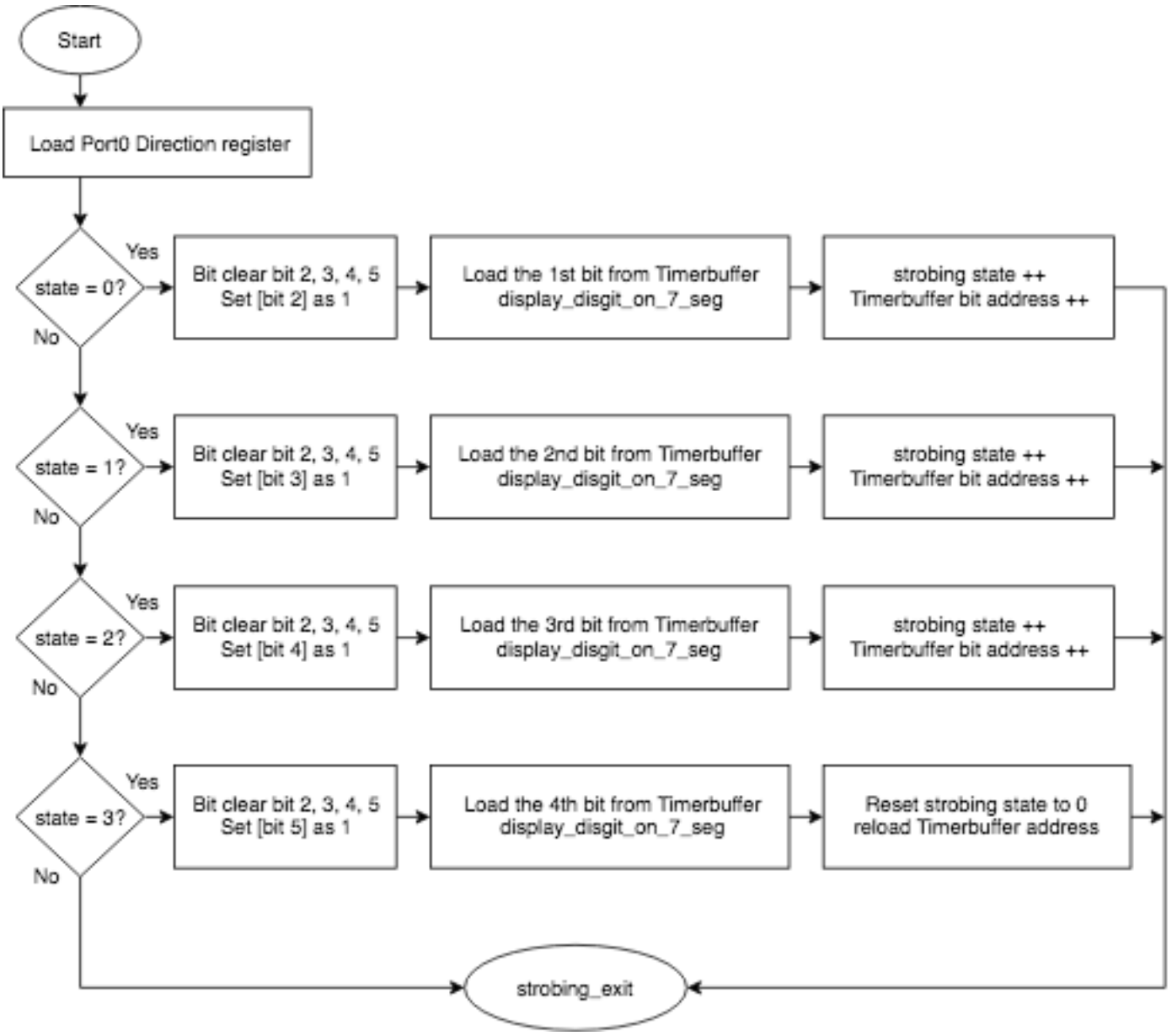
3.4 FIQ_Handler



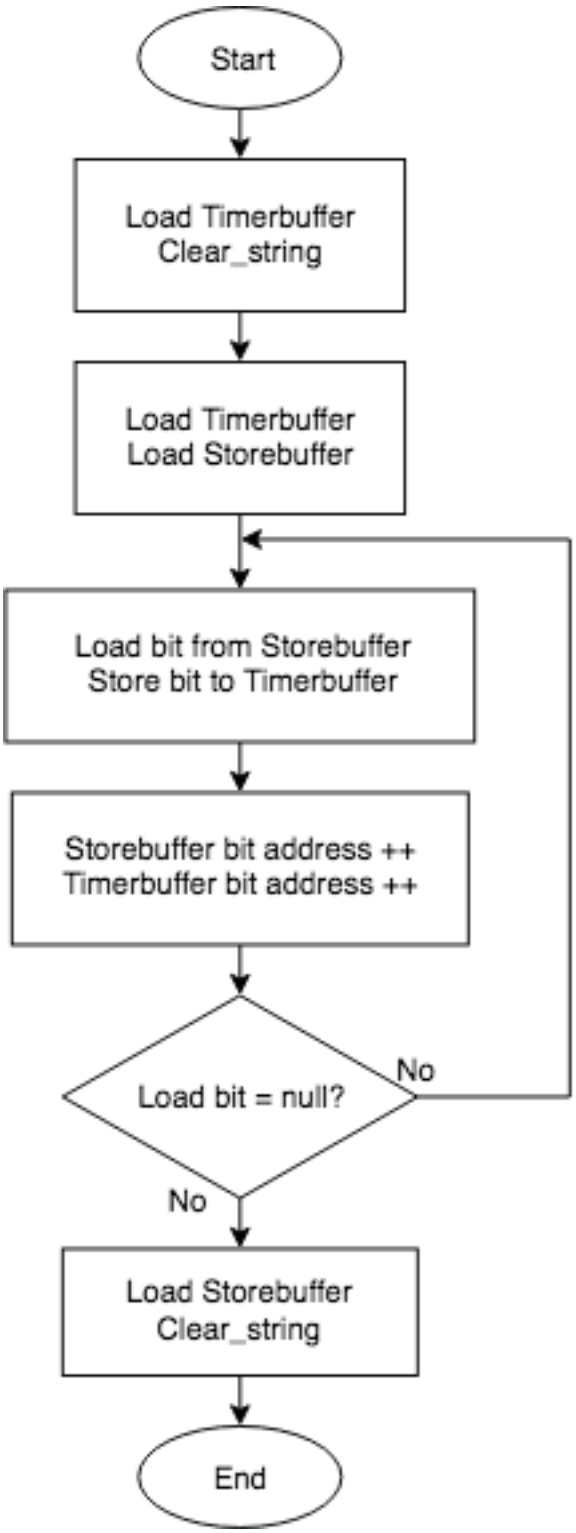
3.5 uart0_comparisons



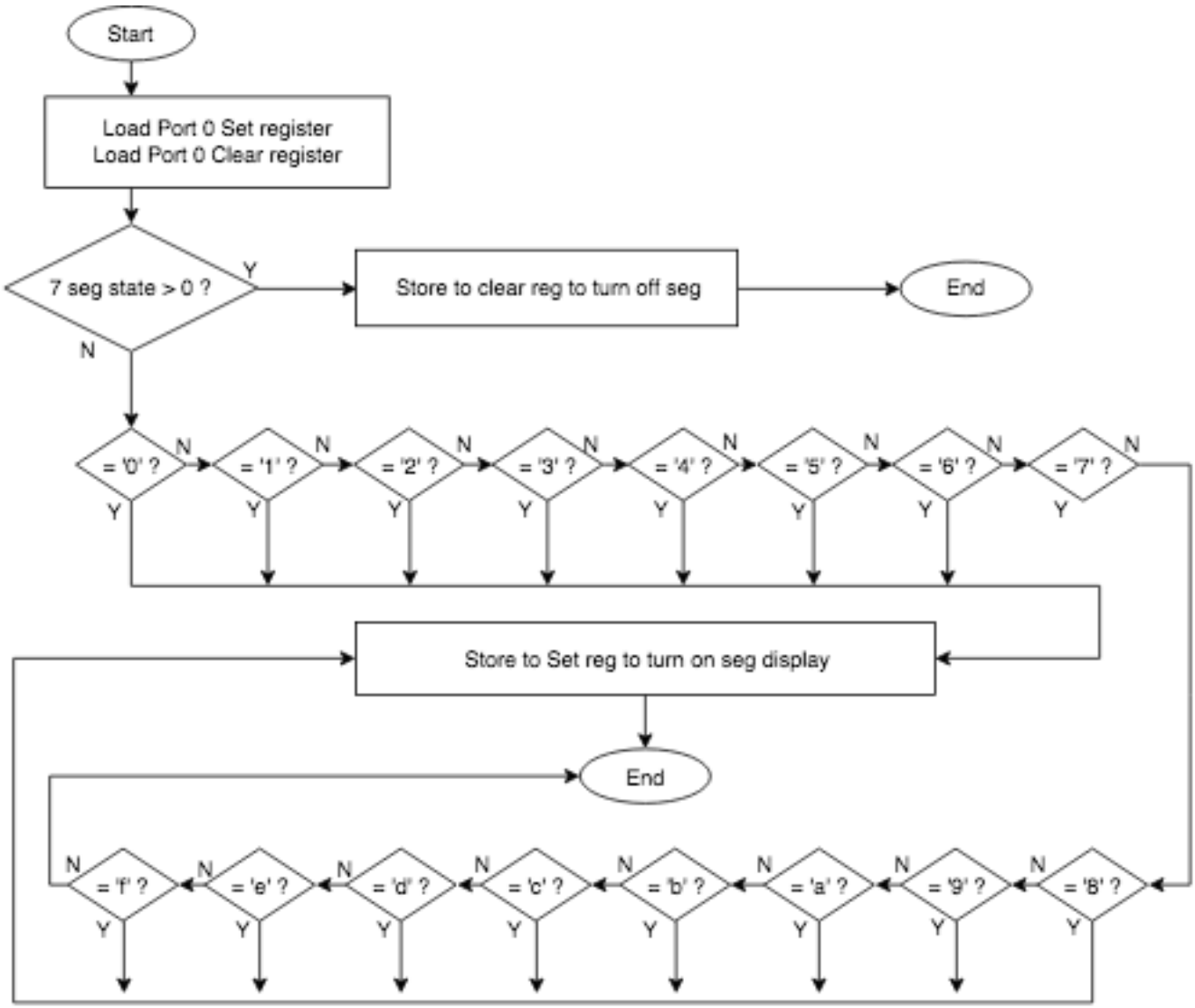
3.6 strobing


Start
Load Port0 Direction register
state = 0?
Yes
No
Bit clear bit 2, 3, 4, 5
Set [bit 2] as 1
Load the 1st bit from Timerbuffer
display_disgit_on_7_seg
strobing state ++
Timerbuffer bit address ++
state = 1?
Yes
No
Bit clear bit 2, 3, 4, 5
Set [bit 3] as 1
Load the 2nd bit from Timerbuffer
display_disgit_on_7_seg
strobing state ++
Timerbuffer bit address ++
state = 2?
Yes
No
Bit clear bit 2, 3, 4, 5
Set [bit 4] as 1
Load the 3rd bit from Timerbuffer
display_disgit_on_7_seg
strobing state ++
Timerbuffer bit address ++
state = 3?
Yes
No
Bit clear bit 2, 3, 4, 5
Set [bit 5] as 1
Load the 4th bit from Timerbuffer
display_disgit_on_7_seg
Reset strobing state to 0
reload Timerbuffer address
strobing_exit

3.7 move_string



3.8 display_digit_on_7_seg



4 Summary

This program has three components, the Cwrapper, library, and lab6. The Cwrapper first initializes pin select, then it calls lab6. The function of lab6 is to create an environment where the program can be interrupted this is done by putting the program in an infinite loop. The first thing the program does is to initialize UART0 then initializes all of the interrupts used. The interrupts used are UART0 interrupt, external interrupt 1, and timer0 interrupt. Then, the program prints out the instructions onto putty before going into an infinite loop to wait for interrupts. Once an interrupt occurs, the program will first identify whether the interrupt is external interrupt 1, UART0, or TIMER0 interrupt. After determining the interrupt, the program will proceed to handle the interrupts. External interrupt 1 is very simple, it just changes the on/off state of the seven-segment display then clearing the interrupt. UART0 interrupt is handled by first reading a character, then it proceeds to check whether the character read is q or enter before checking whether character is a valid input. If it is a valid input store the value into storebuffer1. If q is read terminate program, if enter is read add NULL to storebuffer1 then move the content of storebuffer1 to timerbuffer before reloading storebuffer1. If an invalid input is read, clear storebuffer1 and print an error message. TIMER0 interrupt is handled simply with strobing and clearing the interrupt. Every time a TIMER0 interrupt occurs strobing will occur. After each interrupt is handled it will branch back to infinite loop to continue waiting for interrupts.

4.1 Interrupt_init:

Initializes TIMER0 interrupts, UART0 interrupts, and external interrupt 1. Also sets up the match control register and match register of TIMER0.

4.2 FIQ_Handler:

This subroutine determines the type of interrupt, TIMER0, external interrupt 1, and UART0 interrupt then handles the interrupt appropriately. UART0 interrupt reads character then tests character using uart0_coparisons. External interrupt 1 turns seven-segment display on/off. Timer interrupt simply calls strobing.

4.3 Uart0_comparisons

This subroutine is a set of comparisons to determine firstly whether read character is q, enter, valid input, or invalid input. If the entered character is valid input store into storebuffer1. If character is invalid input, clear storebuffer1 and print an error message. If read character is q terminate program. If read character is enter move content of storebuffer1 to timerbuffer and reload storebuffer1.

4.4 Strobing:

Strobing displays the content of timerbuffer on the seven-segment display. Register r0 holds the current display to be turned on with 0 corresponding to leftmost display and 3 corresponding to rightmost display. The logic for strobing first determines which display to turn on using r0, turns on the corresponding display via IO0DIR then displaying the character in the corresponding position in timerbuffer. After the character is displayed r0 will be incremented by 1 and timerbuffer?s address will also be incremented by 1. Once r0 hits becomes 3 it will first display the digit before changing r0 back to 0 and reloading timerbuffer to repeat this process starting from the leftmost display.

4.5 Move_string:

Move string moves content of storebuffer1 to timerbuffer before clearing content of storebuffer1.