Keypad Project

Project 2

Prepared for

CSE 410 Advanced Topics in Embedded System Design

By

Steven Collins (sac9)

September 23, 2024

Table of Contents

Section 1: Division of Work

I, Steven Collins, wrote the entirety of the keypad project. Some subroutines were used from CSE 379, on which my partner Frank Peretti and I worked together to implement.

Section 2: Program Overview

➔ Program Overview

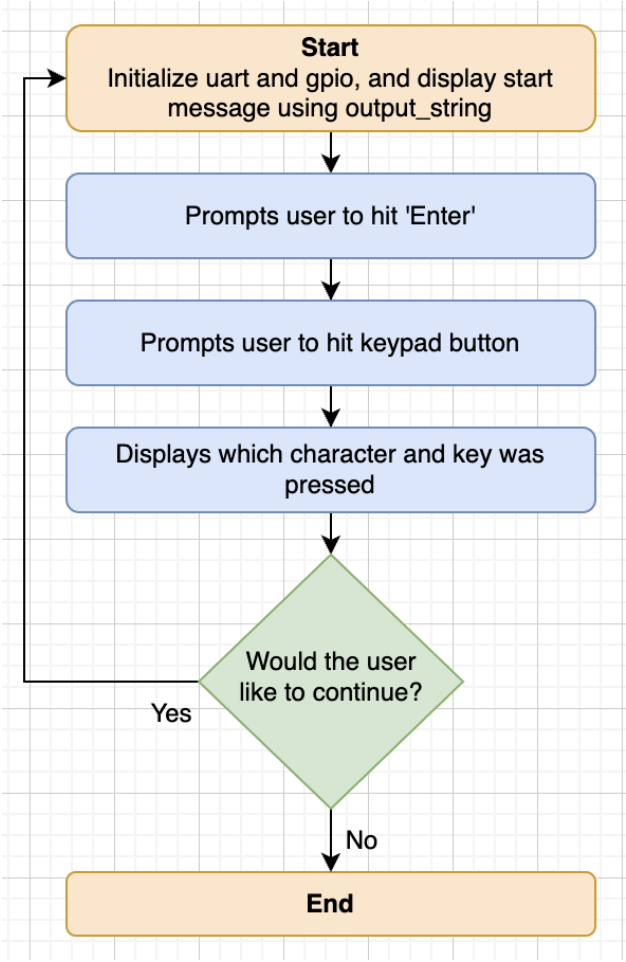
- Hit ‘Enter’ to press a keypad key
- Press one keypad button
- The screen will display the character corresponding to that key, and the actual key; for example: '1' was pressed (K0)
- Press ‘c’ to enter another keypad button; ‘q’ to quit the program

➔ Program Summary

This keyboard button routine allows a user to enter a state to press a keypad button, and whichever keypad button they press will be displayed to the screen. Afterwards, the user can essentially “rerun” the program by entering ‘c’ to continue, and press another keypad button, or they can quit by entering ‘q.’

➔ High Level Flowchart

- *An overview of the entire program*



Section 3: Subroutine Descriptions

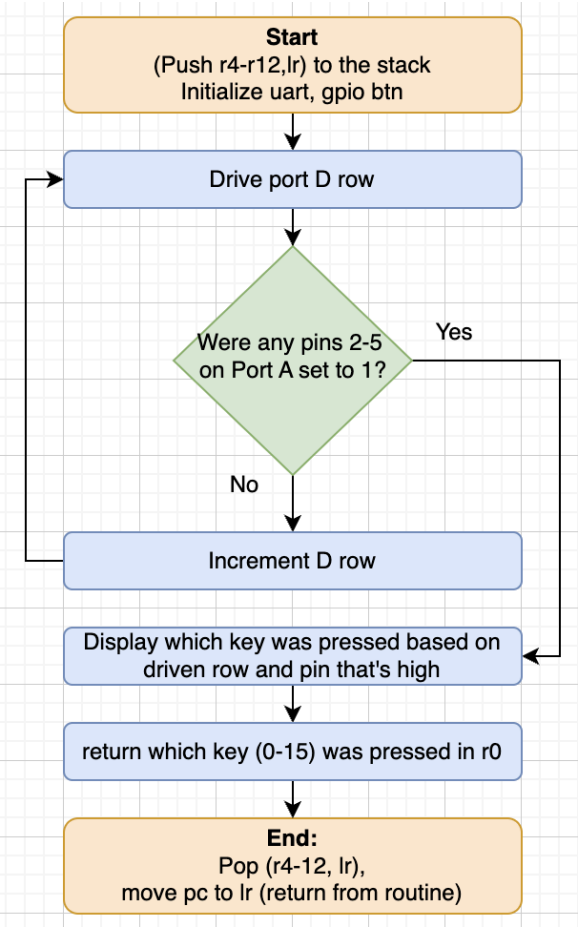
Description of each subroutine's functionality, the arguments it passes through, and what the return values in each register are.

➔ determine_keypad_button

- Functionality:
  - This subroutine determines which keypad button was pressed by displaying it to the screen, and well as returning which key was pressed in r0.
- Arguments:
  - There are no argument registers for the functionality of this subroutine.
- Return registers:
  - r0 - returns decimal value (0-15) of which keypad button was pressed.

Section 4: Subroutine Flowcharts

➔ determine_keypad_button Flowchart



Section 5: Hypothesis

The purpose of this project was to configure gpio ports to be able to determine which keypad button was being pressed. In the process of doing this, a situation from the past was to be kept in mind; that is, be careful to note if and when the Tiva boards were bricked.

The code written for this project works on all three versions of the board, and no Tivas were bricked in the process. A few possible reasons why they may have been bricked in the past are static electricity, GPIO port misconfiguration, and memory and peripheral conflicts. Like Dr. Schindler previously mentioned, the static electricity of the room where the keypads were previously being used is not the same as the UB Commons 113. With tile floors, the static electricity could have been causing extraneous voltages/currents, messing up the boards. It's also possible that between versions on the board, there is an additional area that needs to be configured. Lastly, the Tiva boards may have been getting bricked by students who were mapping peripheral control registers to memory addresses for one version of the board, but is different for the other. It's possible that under incorrect assumptions about register addresses, important control registers may have been overwritten, disabling key functionality for CCS and the Tiva board.