EGR: 226 Microcontroller Programming and Applications

Winter 2021

Instructor Prof. Trevor Ekin

**Lab 6: Interfacing a keypad with the MSP432**

Gabriel Gasbarre

March 3rd, 2021

Table of Contents

[1. Objectives 3](#_Toc65677682)

[2. Equipment 3](#_Toc65677683)

[3. Introduction 4](#_Toc65677684)

[3.1: Pre-Lab: 4](#_Toc65677685)

[3.2: Part 1- Displaying the key pressed on the CCS console 4](#_Toc65677686)

[3.3 Part 2 - Collecting and storing input from the keypad 4](#_Toc65677687)

[3.4 Part 3 – EXTRA CREDIT Collecting a PIN and comparing with key entry 5](#_Toc65677688)

[4. Procedure 5](#_Toc65677689)

[4.1: Part 1 - Displaying the key pressed on the CCS console 5](#_Toc65677690)

[4.2: Part 2- Collecting and storing input from the keypad 5](#_Toc65677691)

[4.3 Part 3 - EXTRA CREDIT Collecting a PIN and comparing with key entry 6](#_Toc65677692)

[5. Results 6](#_Toc65677693)

[5.1 Part 1 Results 6](#_Toc65677694)

[5.2: Part 2 Results 6](#_Toc65677695)

[5.3: Part 3 Results 6](#_Toc65677696)

[Conclusions. 7](#_Toc65677697)

1. Objectives

The objectives for lab 6 are as follows:

To develop a C program that detects and decodes which key is pressed on a standard 12 key keypad interfaced to the MSP432.

To develop a C program that collects a PIN from a user through the keypad and confirm its entry.

1. Equipment

|  |  |  |
| --- | --- | --- |
| **Part** | **Description** | **Model** |
| CCS (Code Composer Studio) | Integrated development environment to develop applications for Texas Instruments embedded processors. | 10.0.00010 |
| MSP432 | Mixed-signal microcontroller family from Texas Instruments. | MSP432P401x |
| EGR:226 Lab 6 Exercise | Interfacing a keypad with the MSP432 | N/A |
| Keypad | EGR 226 Lab Kit component | N/A |

1. Introduction

3.1: Pre-Lab:

For the pre-lab deliverables, students are to make a drawing of the keypad switch layout, along with what ports the I/O pins will be connected to on the MSP. Pre-Lab datasheet included below. A picture containing text, whiteboard

Description automatically generated

3.2: Part 1- Displaying the key pressed on the CCS console

For part 1, students are to use the MSP432 to send the key press information collected by the program to the CCS console window for display. Switch bounce accommodation is necessary, along with input buffers to keep the program from reporting multiple responses from a single key press.

## 3.3 Part 2 - Collecting and storing input from the keypad

For part 2, students will modify their code from part one to prompt the user and collect a 4-digit PIN code that will be stored and displayed. The \* and # key presses must be detected and prevented from being entered into the numerical code. This program should also continue to receive key presses and store the last 4 entries, displaying them after the # key is pressed. This function should also detect if less than 4 entries were collected when the # key is pressed and prompt the user to enter a 4 digit PIN code.

## 3.4 Part 3 – EXTRA CREDIT Collecting a PIN and comparing with key entry

For part 3, students will convert their PIN entry program into a subroutine that is called from main. The program will collect input from the keypad to create a valid 4 digit code as in part 2. Upon successful entry, the program will prompt the user to enter the PIN again to validate consistent entry. This program should then compare the codes and confirm if or if not they are identical.

1. Procedure

4.1: Part 1 - Displaying the key pressed on the CCS console

First, students are to connect their keypad to the MSP432 I/O ports designated in the pre-lab exercise. After writing the keypad initialization function for all of these I/O ports, students are to write the complete code for printing the key pressed on the CCS console using the Keypad\_Read function. The Keypad\_Read function operates by setting one column to a low output while reading each row. Thus, if there is a reading in row 1 while column 1 is active, the program reports a key press of 1. Keypad\_Read then returns a value of 1 which can be printed through .main.

4.2: Part 2- Collecting and storing input from the keypad

For part 2, Keypad\_Read is called multiple times until the pressing of the # key. The values read from the Keypad\_Read function are stored in a PIN array that cycles out the oldest entry for the newest, allowing for unlimited entries while still only printing the 4 most recently entered keystrokes. When the # key is pressed, each value in the PIN array is printed in order to the CCS console.

## 4.3 Part 3 - EXTRA CREDIT Collecting a PIN and comparing with key entry

For part 3, students are to convert the program developed in part 2 so that it can be called as a subroutine from main. Comparing the 2 PINs inputted as discussed in the introduction section is done basically by duplicating the code in part 2 and storing each pin individually. After both pins are read in, the data values are sent to a compareArray function that returns 1 if the PINs are unequal, and 0 if the pins are equal. This value is then used to display print statements corresponding to if or if not the pins are equal.

1. Results
   1. Part 1 Results

As with many labs, part 1 was the most difficult as it is the foundation for the rest of the lab. Upon running the code, however there were very few debugging issues that were required which lead to a much easier implementation of the code in parts 2 and 3. This function also has complete debouncing and input buffering included, so calling it in other functions resulted in very simplified part 2 and 3 functions.

5.2: Part 2 Results

For part 2, all requirements were met and very few issues persist. Very infrequently the function receives a null (404) value when the keypad read function fails to receive input, but for the most part reading and printing pins is flawless.

## 5.3: Part 3 Results

Part 3 was almost identical in execution to part 2, as the code was basically duplicated

and each input compared. All requirements have been successfully met for each part of the lab.

# Conclusions.

I’m very proud of the results of my lab, aside from the occasional missed input error. I would say my program is about 98% accurate. I am also happy with the amount of comments in my code, as I included in line comments in almost every line, and function comments for each function as well as library comments.