**EGR 226: Microcontroller Programming and Applications**

**Winter 2021**

Instructor: Professor Trevor Ekins

Lab 6: Keypad User Input

Sam Wieneke

March 3, 2021

**Contents**

1. **Objectives**
2. **Equipment**
3. **Introduction**
4. **Procedure**
5. **Results/ Discussion**
6. **Conclusion and Future Work**

**Appendices**

1. **Source Code: Part 1**

**Source Code: Part 2**

1. **Objectives**

The objectives of this lab were to develop a C program that detects and decodes which key is pressed on the keypad interfaced with the MSP432, and to develop a C program that collects a PIN from a user through the keypad and confirms its entry.

1. **Equipment**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Part | Description | Model | Measured Value | Notes |
| Code Composer Studios | Texas Instruments programming environment | Version 9.3.0 | N/A | N/A |
| GitLab | Remote Repository for code maintenance | N/A | N/A | Makes collaboration on team projects and code very convenient. |
|  |  |  |  |  |

1. **Introduction**

**Part 1: Displaying the key pressed on the CCS console**

Part 1 involved sending the keypress information collected by the program and displaying the key pressed on the CCS console window. Switch bounce will be implemented so the program will not sense multiple keys being pressed. The program will also not return the key pressed until it is released, when a keypad button is held down, only 1 character will be displayed.

**Part 2: Collecting and storing input from the keypad.**

Part 2 involved prompting the user to enter a 4 digit PIN. It will rejected the “\*” and “#” keys as part of the pin. The “#” was used to signify the termination of entry for the PIN. The program stored this code as a temporary variable and print the PIN to the screen after “#” was pressed and correct entries were entered.

1. **Procedure**

**Part 1**

This code was created by having a large looping function where depending what number the keypad reads due to the Read\_Keypad custom function, the given character will be displayed. There are three special cases where the number returned from the Read\_Keypad function has to be different and there are if and else if statements that print a different character and not the number it returns from the Read\_Keypad function.

**Part 2**

Part 2 was similar to part 1, but an array was added to save the numbers and when the “#” symbol was pressed, it would display the last 4 numbers that were pressed. The code would still display each key pressed, but would check to see if 4 correct numbers were displayed before and redisplay the last 4 numbers pressed.

1. **Results/ Discussion**

**Lab Results**

To demonstrate this lab, part 1 had to be shown to the instructor as well as part 2. For part 1, each time a character on the keypad was pressed, it had to be displayed on the console window. There had to be slight modifications to the reading the keypad function, to make the last row work correctly. For part 2, each number was also displayed to the console window, and saved into an array that would redisplay the last 4 numbers entered when “#” was pushed.

**Prelab:**

The prelab involved writing the Keypad\_Read() function to scan the keypad and save the key press as a variable. It also involved writing code that would print the character pressed on the keypad to the console window.

1. **Conclusion/ Future Work**

This lab was the first use of the keypad. Code was created to print the keys and redisplay them as a “pin”. Some challenges that were faced in the lab was figuring out a way to not accept the “\*” as a part of the pin and when pressing the pound symbol, to have it redisplay the last 4 numbers that were pressed. Am improvement to the code could’ve been to have “\*” act as a backspace instead of nothing for the pin part of the lab so it could erase a number pressed so all 4 numbers would not have to be entered to get the desired pin.

**Appendix A**

Part 1 Source Code:

**#include** "msp.h"

**#include** <stdio.h>

**void** **SysTick\_Init** (**void**);

**void** **SysTick\_delay** (uint16\_t delay);

uint8\_t **Read\_Keypad**(**void**);

**void** **keypad\_init** (**void**);

uint8\_t pressed, num;

**void** **main**(**void**)

{

WDT\_A->CTL = WDT\_A\_CTL\_PW | WDT\_A\_CTL\_HOLD; // stop watchdog timer

keypad\_init ();

SysTick\_Init();

**fflush**(stdin);

**printf**("Print a Key\n");

**while** (1)

{

pressed = Read\_Keypad ();

**if** (pressed)

{

**if**(num == 10)

{

**printf**("\*\n");

}

**else** **if**(num == 11)

{

**printf**("0\n");

}

**else** **if**(num == 12)

{

**printf**("#\n");

}

**else**

{

**printf**("%d\n", num);

}

SysTick\_delay (1000);

}

}

}

uint8\_t **Read\_Keypad**(**void**)

{

uint8\_t col, row;

**for** ( col = 0; col < 3; col++ )

{

P4->DIR = 0x00; // Set Columns to inputs

P4->DIR |= BIT ( 4+col ); // Set column 3 to output

P4->OUT &= ~BIT ( 4+col ); // Set column 3 to LOW

SysTick\_delay (10); // Delay the while loop

row = P4->IN & 0x0F; // read all rows

**while** ( !(P4IN & BIT0) | !(P4IN & BIT1) | !( P4IN & BIT2) | !( P4IN & BIT3) );

**if** (row != 0x0F) **break**; // if one of the input is low, some key is pressed.

}

P4->DIR = 0x00; // Set Columns to inputs

**if** ( col == 3)

**return** 0;

**if** (row == 0x0E) num = col + 1; // key in row 0

**if** (row == 0x0D) num = 3 + col +1; // key in row 1

**if** (row == 0x0B) num = 6 + col +1; // key in row 2

**if** (row == 0x07) num = 9 + col+1; // key in row 3

**return** 1;

}

**void** **keypad\_init** (**void**)

{

P4->SEL1 &= ~BIT0; // configure P4.0 as simple I/O

P4->SEL0 &= ~BIT0;

P4->DIR &= ~BIT0; // P4.0 set as output pin \*/

P4->REN |= BIT0;

P4->OUT |= BIT0;

P4->SEL1 &= ~BIT1; // configure P4.1 as simple I/O

P4->SEL0 &= ~BIT1;

P4->DIR &= ~BIT1; // P4.1 set as output pin \*/

P4->REN |= BIT1;

P4->OUT |= BIT1;

P4->SEL1 &= ~BIT2; // configure P4.2 as simple I/O

P4->SEL0 &= ~BIT2;

P4->DIR &= ~BIT2; // P4.2 set as output pin \*/

P4->REN |= BIT2;

P4->OUT |= BIT2;

P4->SEL1 &= ~BIT3; // configure P4.3 as simple I/O

P4->SEL0 &= ~BIT3;

P4->DIR &= ~BIT3; // P4.3 set as output pin \*/

P4->REN |= BIT3;

P4->OUT |= BIT3;

P4->SEL1 &= ~BIT4; // configure P4.4 as simple I/O

P4->SEL0 &= ~BIT4;

P4->DIR &= ~BIT4; // P4.4 set as output pin \*/

P4->SEL1 &= ~BIT5; // configure P4.5 as simple I/O

P4->SEL0 &= ~BIT5;

P4->DIR &= ~BIT5; // P4.5 set as output pin \*/

P4->SEL1 &= ~BIT6; // configure P4.6 as simple I/O

P4->SEL0 &= ~BIT6;

P4->DIR &= ~BIT6; // P4.6 set as output pin \*/

}

**void** **SysTick\_Init** (**void**)

{ //initialization of systic timer

SysTick->CTRL = 0; // disable SysTick During step

SysTick->LOAD = 0x00FFFFFF; // max reload value

SysTick->VAL = 0; // any write to current clears it

SysTick->CTRL = 0x00000005; // enable systic, 3MHz, No Interrupts

}

**void** **SysTick\_delay** (uint16\_t delay)

{ // Systick delay function

SysTick->LOAD = ((delay \* 3000) - 1); //delay for 1 msecond per delay value

SysTick->VAL = 0; // any write to CVR clears it

**while** ( (SysTick->CTRL & 0x00010000) == 0); // wait for flag to be SET

}

Part 2 Source Code:

**#include** "msp.h"

**#include** <stdio.h>

**void** **SysTick\_Init** (**void**);

**void** **SysTick\_delay** (uint16\_t delay);

uint8\_t **Read\_Keypad**(**void**);

**void** **keypad\_init** (**void**);

uint8\_t pressed, num;

**void** **main**(**void**)

{

WDT\_A->CTL = WDT\_A\_CTL\_PW | WDT\_A\_CTL\_HOLD; // stop watchdog timer

keypad\_init ();

SysTick\_Init();

**fflush**(stdin);

uint8\_t pinArray[4];

**int** i=0;

**printf**("Enter a 4 digit pin, then press # to confirm\n");

**while** (1)

{

pressed = Read\_Keypad ();

**if** (pressed)

{

**if**(num == 10)

{

**printf**("\*\n");

}

**else** **if**(num == 11)

{

num = 0;

**printf**("%d\n",num);

}

**else** **if**(num == 12)

{

**printf**("#\n");

}

**else**

{

**printf**("%d\n", num);

}

SysTick\_delay (100);

**if**(num != 10 && num !=12)

{

**if** (i==4)

{

pinArray[0] = pinArray[1];

pinArray[1] = pinArray[2];

pinArray[2] = pinArray[3];

pinArray[3] = num;

}

**if** (i<4)

{

pinArray[i] = num;

i++;

}

}

**if**(num == 12 && i==4)

{

**printf**("Pin Entered: %d %d %d %d\n", pinArray[0], pinArray[1], pinArray[2], pinArray[3]);

}

**if**(num == 12 && i<4)

{

**printf**("Please enter more digits, then press #\n");

4-i;

}

}

}

}

uint8\_t **Read\_Keypad**(**void**)

{

uint8\_t col, row;

**for** ( col = 0; col < 3; col++ )

{

P4->DIR = 0x00; // Set Columns to inputs

P4->DIR |= BIT ( 4+col ); // Set column 3 to output

P4->OUT &= ~BIT ( 4+col ); // Set column 3 to LOW

SysTick\_delay (10); // Delay the while loop

row = P4->IN & 0x0F; // read all rows

**while** ( !(P4IN & BIT0) | !(P4IN & BIT1) | !( P4IN & BIT2) | !( P4IN & BIT3) );

**if** (row != 0x0F) **break**; // if one of the input is low, some key is pressed.

}

P4->DIR = 0x00; // Set Columns to inputs

**if** ( col == 3)

**return** 0;

**if** (row == 0x0E) num = col + 1; // key in row 0

**if** (row == 0x0D) num = 3 + col +1; // key in row 1

**if** (row == 0x0B) num = 6 + col +1; // key in row 2

**if** (row == 0x07) num = 9 + col+1; // key in row 3

**return** 1;

}

**void** **keypad\_init** (**void**)

{

P4->SEL1 &= ~BIT0; // configure P4.0 as simple I/O

P4->SEL0 &= ~BIT0;

P4->DIR &= ~BIT0; // P4.0 set as output pin \*/

P4->REN |= BIT0;

P4->OUT |= BIT0;

P4->SEL1 &= ~BIT1; // configure P4.1 as simple I/O

P4->SEL0 &= ~BIT1;

P4->DIR &= ~BIT1; // P4.1 set as output pin \*/

P4->REN |= BIT1;

P4->OUT |= BIT1;

P4->SEL1 &= ~BIT2; // configure P4.2 as simple I/O

P4->SEL0 &= ~BIT2;

P4->DIR &= ~BIT2; // P4.2 set as output pin \*/

P4->REN |= BIT2;

P4->OUT |= BIT2;

P4->SEL1 &= ~BIT3; // configure P4.3 as simple I/O

P4->SEL0 &= ~BIT3;

P4->DIR &= ~BIT3; // P4.3 set as output pin \*/

P4->REN |= BIT3;

P4->OUT |= BIT3;

P4->SEL1 &= ~BIT4; // configure P4.4 as simple I/O

P4->SEL0 &= ~BIT4;

P4->DIR &= ~BIT4; // P4.4 set as output pin \*/

P4->SEL1 &= ~BIT5; // configure P4.5 as simple I/O

P4->SEL0 &= ~BIT5;

P4->DIR &= ~BIT5; // P4.5 set as output pin \*/

P4->SEL1 &= ~BIT6; // configure P4.6 as simple I/O

P4->SEL0 &= ~BIT6;

P4->DIR &= ~BIT6; // P4.6 set as output pin \*/

}

**void** **SysTick\_Init** (**void**)

{ //initialization of systic timer

SysTick->CTRL = 0; // disable SysTick During step

SysTick->LOAD = 0x00FFFFFF; // max reload value

SysTick->VAL = 0; // any write to current clears it

SysTick->CTRL = 0x00000005; // enable systic, 3MHz, No Interrupts

}

**void** **SysTick\_delay** (uint16\_t delay)

{ // Systick delay function

SysTick->LOAD = ((delay \* 3000) - 1); //delay for 1 msecond per delay value

SysTick->VAL = 0; // any write to CVR clears it

**while** ( (SysTick->CTRL & 0x00010000) == 0); // wait for flag to be SET

}