ECE177: Programming I: From C...

Lab #6 — Keypad

Week of 23 March 2026

1 Objectives

- Complete all of the steps outlined below while working on `keypad.c`
- Program the Pico breadboard with the modified `keypad.c` file and confirm working output

2 Materials

- Assembled and working Pico breadboard
- USB cable
- Laptop and Laptop charger

3 Procedure

1. Make sure to read all of the directions carefully before getting started.
2. Download the `lab06_code.zip` file here:
   `https://web.eece.maine.edu/~vweaver/classes/ece177/labs/lab06_code.zip`
3. Extract the zip file in your `Documents/ece177/labs/` directory that you created in lab01.
4. In VS Code, use the Raspberry Pi Pico Extension to import the `lab06_code` folder as a project.

4 Modify the Provided Code

1. Edit the provided `keypad.c` file
2. This document describes the tasks you need to do. There are also code comments in the `keypad.c` file but they might not be as detailed as the ones in this document.
3. Make sure to read all of the directions carefully before getting started so as to not miss any of the instructions.
4. Be sure to comment your code!
5. Test your code often! Don't try to write it all at once!
6. As with previous labs ideally you can just press the VS Code "Run" button and it will compile and upload your program to the Pico. If it doesn't you might have to find the `keypad.uf2` file in the `build` subdirectory and copy it over to the Pi Pico virtual drive.
7. Note: unlike the last few labs you won't be able to use '*'+'D' to enable BOOTSEL mode. This is because we are writing the keypad code ourself from scratch.

Section A: Update the Code Comments

(a) Update the code comments at the top of the file with your name, the date, and a brief description of the lab.

Section B: Make the code a valid C program

(a) First add a proper `int main()` declaration in Section B near the start of the file.

(b) When it becomes necessary, add any needed variables right after the `main()` function you created

(c) Skip to the end of the file and put do the following:

i. Add a proper `return` statement at the end

ii. Don't forget a closing curly brace.

Section C: Initialize the GPIO Pins

(a) First Initialize the 10 GPIOs for the LED bargraph.

i. Use a for loop, and iterate from GPIO13 to GPIO22 (inclusive).

ii. If you add new variables (for example loop iterators) add them to the top of the file after `main()`

iii. Reminder, you'll want to make the following calls for each of the GPIOs, just like you did in Section Lab 5

```
gpio_init(gpio_number);
gpio_put(gpio_number,0);
gpio_set_dir(gpio_number,GPIO_OUT);
```

(b) Next Initialize the 8 GPIOs for the keypad.

i. Use a for loop, and iterate from GPIO5 to GPIO12 (inclusive).

ii. This is the same as setting things up for the LEDs, but you want to set the GPIO direction to be `GPIO_IN` instead of `GPIO_OUT`

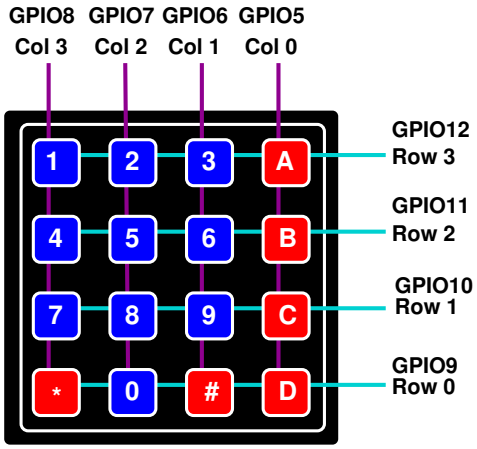
(c) Set the initial values for the keypad GPIOs

i. Have another for loop, and iterate from GPIO5 to GPIO12 (inclusive).

ii. Set the pin to off (0) for GPIO5 to GPIO9 (inclusive)

iii. Otherwise set the pin to on (1) (for GPIO10 to GPIO12)

Section D: Scan the Keypad



The goal of this lab is to "scan" the keypad and print the row and column of the key currently being pressed (nothing should be printed if no key is pressed).

The keypad has 16 buttons arranged in a 4x4 grid. The GPIOs from the Pi-pico are hooked to the rows and columns as shown in the above figure. GPIO9 through GPIO12 are the rows and GPIO5 through GPIO8 are the columns.

In Section C we set all the keypad GPIOs to be inputs. To scan: set one (and only one) row GPIO at a time to an output with an output value of '1'. Then each column (which are set as inputs) is checked in turn to see if the GPIO is *not* zero. When a key is pressed, it shorts the row and column pins together allowing the voltage set on the row to appear on the column. You can read the input from a GPIO by using the `sio_hw->gpio_in` value and checking the proper bit.

By scanning each possible combination (16 in all) you can determine which key is being pressed.

Care needs to be taken to ensure that pressing multiple keypad buttons does not connect a GPIO that is an output at 3.3 volts to another GPIO that is an output at zero volts. This is a short and can damage the Pico. Be sure to *only* set one row GPIO as an output at a time. As long as you do that, and all other pins are inputs, you don't have to worry about a short. When in "input" mode the GPIO pins are in a "high-impedance" state and not actually set to 0 or 1.

For this code we will use a few special memory-mapped GPIO variables:

- `sio_hw->gpio_out` is a 32-bit value used to set all 32 GPIO values with one write. You already used this in Lab 5.
- `sio_hw->gpio_in` is a 32-bit value that can be read to see the current input values on all 32 GPIOs. If a row is active, you can use bit manipulation to check the four column GPIOs (GPIO5/6/7/8) to see if a button is pressed: a `0` represents not pressed and `1` represents being pressed.
- `sio_hw->gpio_oe` is used to pick whether a GPIO is in input our output mode. A '0' represents input and '1' represents an output in each bit position.

Things to Code in Part D

(a) Create an infinite loop; In the body of the never ending loop, do the following steps

(b) Create a for loop that iterates over the four rows

(c) Based on the loop iterator we want to turn on only one of GPIO9, GPIO10, GPIO11, or GPIO12.

i. First clear bits 9, 10, 11, and 12 of `sio_hw->gpio_out` to zero without changing any of the other GPIO pins. As a reminder, you do this with bitwise AND and a proper mask.

ii. Next clear the output enable bits 9, 10, 11, and 12 of `sio_hw->gpio_oe` to 0 as well. This will set GPIO9..12 to be inputs.

iii. Next set the output enable bit of the row you want to be '1' to an output via `sio_hw->gpio_oe`. Remember you can use bitwise OR to set a bit, and you can use something like `(1<<9)` to create an integer with the 9th bit set.

iv. Finally set the output to '1' in `sio_hw->gpio_out` for the proper bit (9 ... 12) using bitwise OR. BE SURE TO ONLY HAVE ONE '1' SET in this range.

(d) Delay 10 microseconds (`sleep_us()`) to allow the pin to become an output.

(e) Create another for loop that iterates over the columns GPIO5, GPIO6, GPIO7, GPIO8 to see if any of the input bits are set

i. Check each column GPIO via `sio_hw->gpio_in` to see if bit 5, 6, 7, or 8 is set.

ii. Remember: to see if a bit is set you can use bitwise AND with a 1 in the proper place, and if it's nonzero it is set, otherwise it isn't.

iii. If the value is non-zero, print the row and columns number of the pressed key using `printf()` and it should appear on the LCD panel.
Something like:

```
Row 3, Col 2
Row 0, Col 1
```

8. If everything went properly, you can press all 16 keys on the keypad and it should show the proper row/column number. If it doesn't you will need to debug things. To make sure you are setting values properly, `printf()` can be used for debugging to make sure values are being set the way you think they are.

5 Grading/Checkoff

1. Upload your `keypad.c` file to BrightSpace.

2. Upload a picture of your board running and showing a successful keypress.

3. TAs will check the output of your final program running on your hardware. They will ask you to demonstrate the lab requirements and will enter grades after the requested information has been uploaded on the assignment rubric.

4. TAs: Enter grades only after verifying full functionality of the code running, the requested information has been uploaded, and the assignment has been submitted.