**ECE0202: Embedded Systems and Interfacing**

**Lab 4: Keypad Scanning (in assembly)**

**Due: 3/21/21 at 11:59pm**

**Objectives**

* **Understand I/O multiplexing technique**
* **Be familiar with keypad scanning algorithms**
* **Implement software debouncing**

**Pre-Lab Reading**

* **Chapter 14.9 Keypad Scanning**

**Deliverables – total 100 points**

* **(40 points) – demonstration of keypad scanning that displays the pressed character on the Tera Term.**
* **(35 points) – Code submission. Code should use software debouncing and be well-written and commented.**
* **(25 points) Submission of the pre-lab register tables and post-lab questions. Indicate how each group member participates and contributes to the lab at the end of the lab report.**
* **(10 points extra credit) Display the last pressed 6 characters, in order, on the Tera Term.**
* **Indicate each group member’s Participation and Contribution.**

**Please submit your code as \*.s files and your schematic as a pdf**

**Keyboard Interface**

The 4x4 keypad used in this lab requires 8 pins (4 row pins and 4 column pins). In this lab, the connection between the keypad and the discovery kit is shown in the following table.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Row | R1 → PC0 | R2 →PC1 | R3 →PC2 | R4 →PC3 |
| Column | C1 → PB1 | C2 → PB2 | C3 → PB3 | C4 → PB5 |

All pins of the input port (C1, C2, C3, and C4) are pulled up to 3V via 2.2k resistors already placed on the Discovery board; however, the output port pins (R1, R2, R3, and R4) will require us to configure pull-up resistors. Within the processor, each GPIO pin can be pulled up via an internal resistor (between 20 and 55k), but the internal pull-up current capability is too weak, and therefore an external pull-up resistor is required, as drawn in Fig. 1.

When looking at the front side of the keypad, the pins on the connector from left to right are:

R1 – R2 – R3 – R4 – C1 – C2 – C3 – C4

Diagram

Description automatically generated

Figure - Picture and schematic of the keypad

The maximum current a GPIO pin can source or sink is 20 mA. When calculating the value of external pull-up resistors, make sure that the current does not exceed 20 mA. For example:

On the Nucleo board, all pins in the input port (PB1, PB2, PB3, and PB5) are connected to ground via a 100 nF capacitor. A very short delay should be added before reading the input port, as seen later in the software flowchart.

**ASCII**

In order to write to the Tera Term, you must store the code for an ASCII character into a memory location. In lab 1, the string to be displayed on the Tera Term is stored at a memory location called “str”. This can also be used as the memory location that stores the character you display in this lab.

To display a character on the Tera Term, you must store the associated ASCII code in the memory location “str”, load the memory address of “str” into r0, and then run the instruction *BL USART2\_Write*.

The following table gives ASCII codes for many characters. Note that these are in decimal!

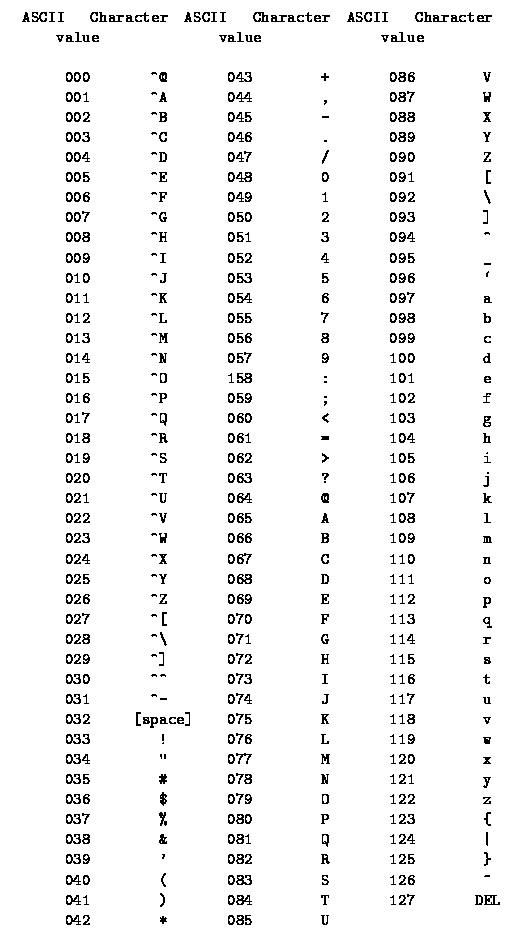


Figure - ASCII character codes. From https://ee.hawaii.edu/~tep/EE160/Book/chap4/subsection2.1.1.1.html

**Software Flowchart**

The following software flowchart is a modified version of that shown in textbook chapter 14.9, and should be used as a general guide for writing the program used in this lab.

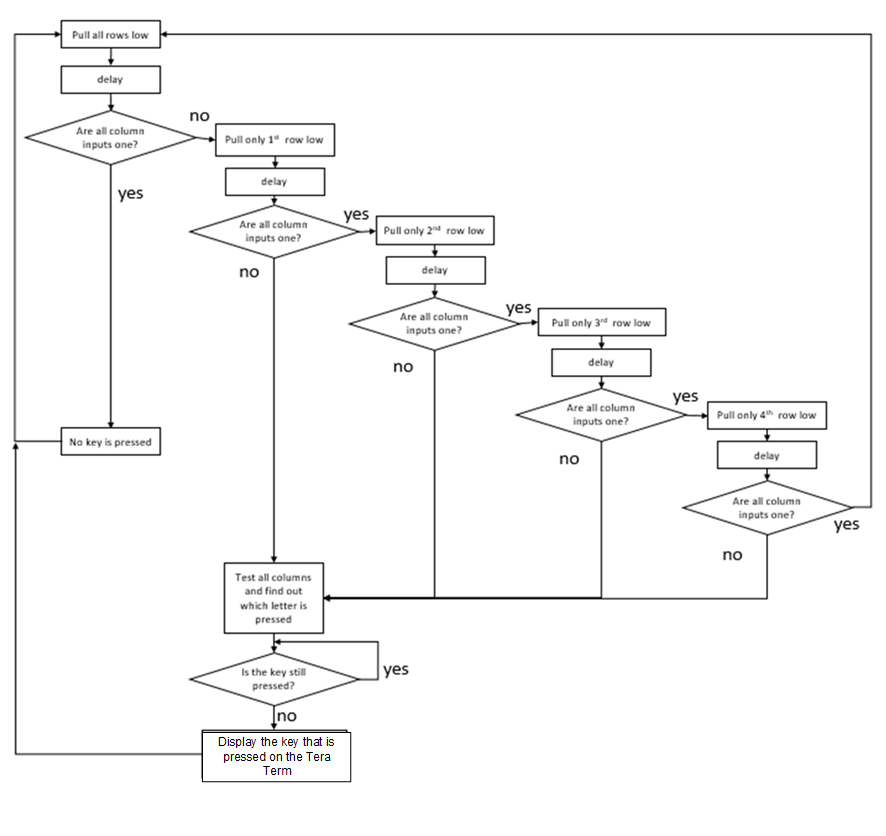
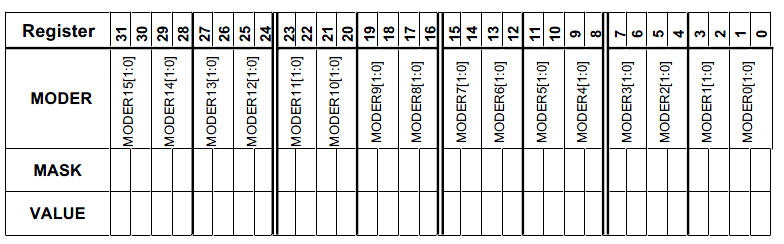


Figure - Software Flowchart for the keypad scanning algorithm

**Pre-Lab Register Tables (5 points)**

**Configure Port C: Pin 0, 1, 2, and 3 as Digital Output**

GPIO Mode: Digital Input (00), Digital Output (01), Alternative Function (10), Analog (11)

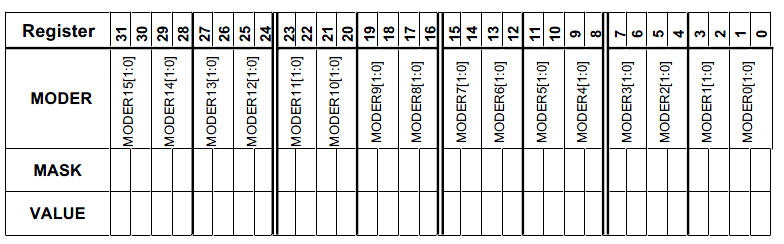


GPIOC Mode Register MASK Value = 0x\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (in HEX)

GPIOC Mode Register Value = 0x\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (in HEX)

**Configure Port B: Pin 1, 2, 3, and 5 as Digital Input**

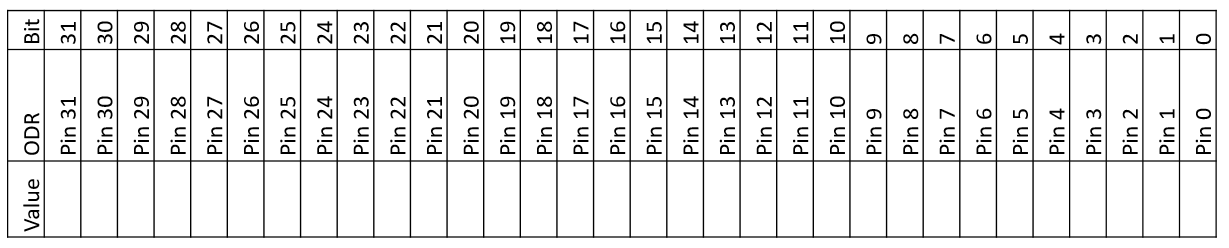
GPIO Mode: Digital Input (00), Digital Output (01), Alternative Function (10), Analog (11)



GPIOB Mode Register MASK Value = 0x\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (in HEX)

GPIOB Mode Register Value = 0x\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (in HEX)

**Write to Port C: Pins 0, 1, 2, and 3 connect to the rows of the keypad**



**Value written to PORTC ODR in order to pull down all rows:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(in HEX)**

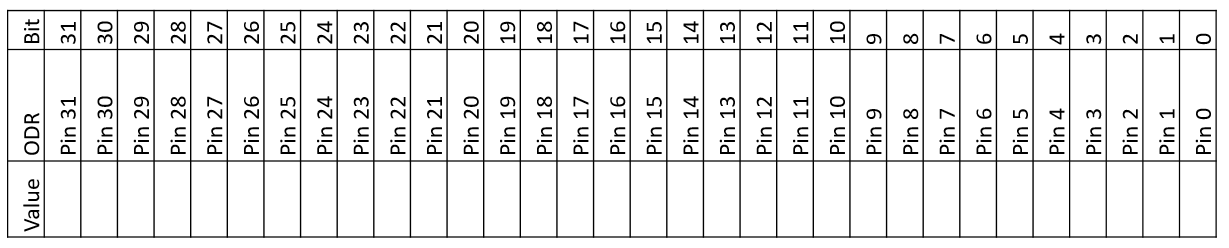
**Value written to PORTC ODR in order to pull down row 1:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(in HEX)**

**Value written to PORTC ODR in order to pull down row 2:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(in HEX)**

**Value written to PORTC ODR in order to pull down row 3:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(in HEX)**

**Value written to PORTC ODR in order to pull down row 4:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(in HEX)**

**Read from Port B: Pins 1, 2, 3, and 5 connect to the columns of the keypad**



**Mask to check if a button from column 1 has been pressed:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(in HEX)**

**Mask to check if a button from column 2 has been pressed:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(in HEX)**

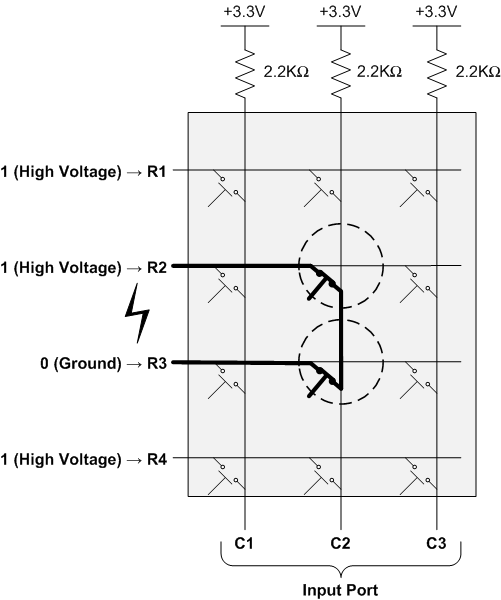
**Mask to check if a button from column 3 has been pressed:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(in HEX)**

**Mask to check if a button from column 4 has been pressed:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(in HEX)**

**Post-Lab Questions (20 points)**

Please include answers to the following questions with your submission of the pre-lab register contents:

1. How is software debouncing implemented in your program? (3 points)
2. What do we mean when we say that the STM32L4’s internal pull-up resistors are too weak for this application? (5 points)
3. When multiple keys are pressed, there could be a short circuit (as shown in the figure). How to configure the output GPIO to avoid this scenario? (7 points)



1. In the following 2 cases, can the scan algorithm correctly detect all keys pressed? If so, how to modify the flowchart (figure 4) of the scan algorithm. If not, explain the reason. (5 points)

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Case1 Case2

Means the keys that are simultaneously pressed.

**Participation and Contribution**

Please indicate the participation and contribution for each group member using the following table.

|  |  |
| --- | --- |
| Name | Participation and Contribution |
|  |  |
|  |  |