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 2.2kΩ 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 55kΩ), 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

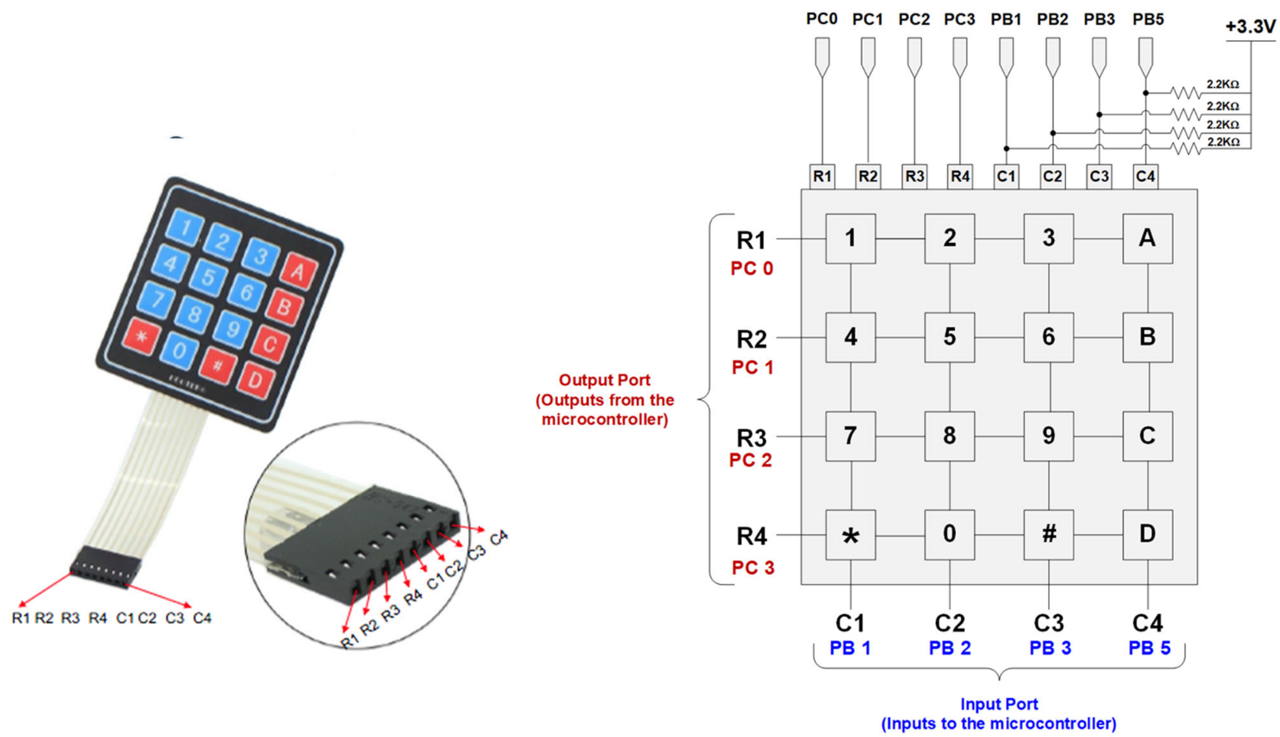


Figure 1- Picture and schematic of the keypad

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:

$$\frac{3\,V}{2.2\,k\Omega} = 1.4\,mA$$

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!

ASCII value	Character	ASCII value	Character	ASCII value	Character
000	^@	043	+	086	V
001	^A	044	,	087	W
002	^B	045	-	088	X
003	^C	046	.	089	Y
004	^D	047	/	090	Z
005	^E	048	0	091	[
006	^F	049	1	092	\
007	^G	050	2	093	]
008	^H	051	3	094	^
009	^I	052	4	095	_
010	^J	053	5	096	`
011	^K	054	6	097	a
012	^L	055	7	098	b
013	^M	056	8	099	c
014	^N	057	9	100	d
015	^O	158	:	101	e
016	^P	059	;	102	f
017	^Q	060	<	103	g
018	^R	061	=	104	h
019	^S	062	>	105	i
020	^T	063	?	106	j
021	^U	064	@	107	k
022	^V	065	A	108	l
023	^W	066	B	109	m
024	^X	067	C	110	n
025	^Y	068	D	111	o
026	^Z	069	E	112	p
027	^[	070	F	113	q
028	^\	071	G	114	r
029	^]	072	H	115	s
030	^^	073	I	116	t
031	^-	074	J	117	u
032	[space]	075	K	118	v
033	!	076	L	119	w
034	"	077	M	120	x
035	#	078	N	121	y
036	$	079	O	122	z
037	%	080	P	123	{
038	&	081	Q	124	|
039	'	082	R	125	}
040	(	083	S	126	~
041	)	084	T	127	DEL
042	*	085	U		

Figure 2 - 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.

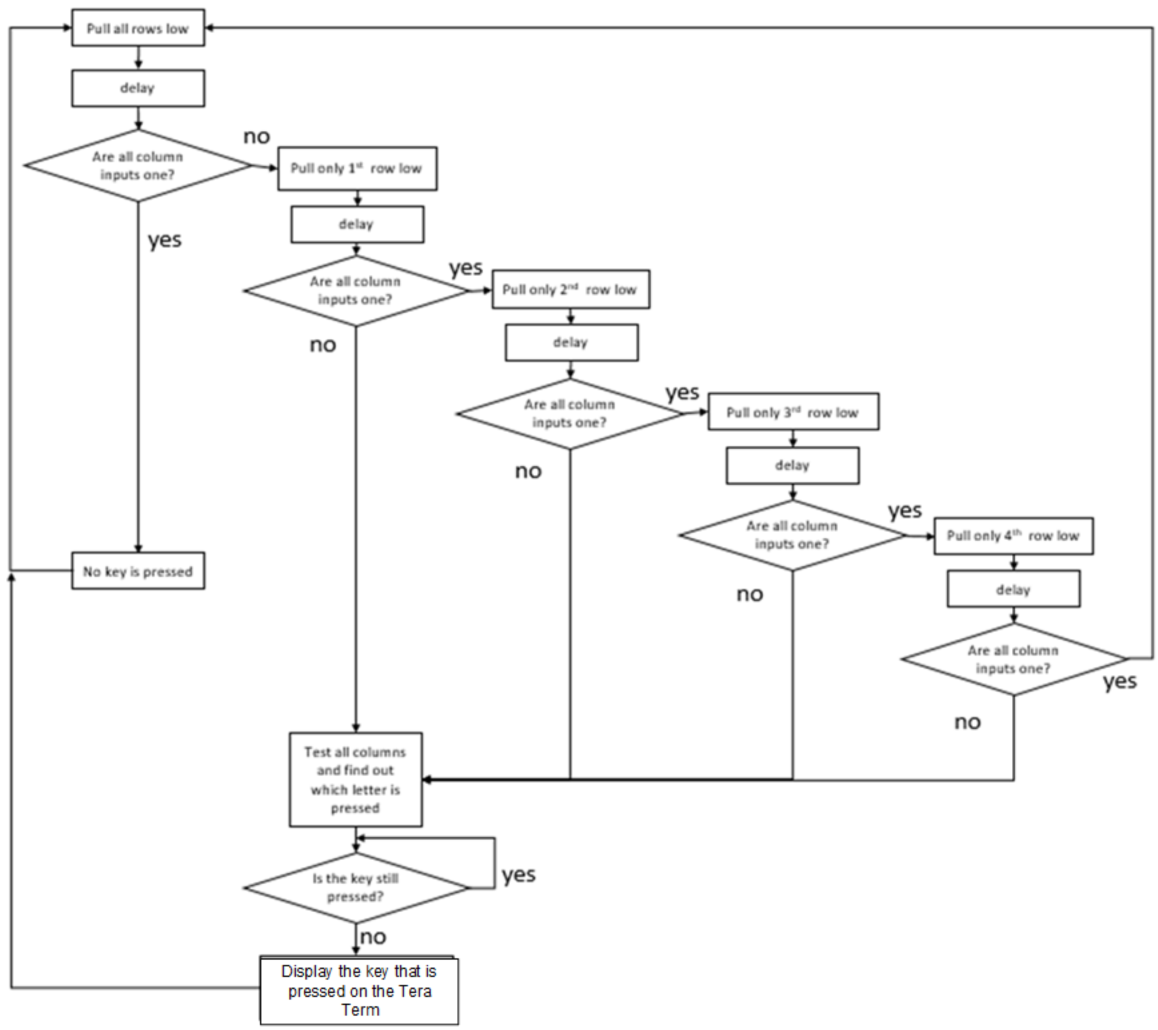


Figure 3 - Software Flowchart for the keypad scanning algorithm

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)

Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MODER	MODER15[1:0]		MODER14[1:0]		MODER13[1:0]		MODER12[1:0]		MODER11[1:0]		MODER10[1:0]		MODER9[1:0]		MODER8[1:0]		MODER7[1:0]		MODER6[1:0]		MODER5[1:0]		MODER4[1:0]		MODER3[1:0]		MODER2[1:0]		MODER1[1:0]		MODER0[1:0]	
MASK																									1	1	1	1	1	1	1	1
VALUE																									0	1	0	1	0	1	0	1

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

GPIOC Mode Register Value = 0x 55 (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)

Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MODER	MODER15[1:0]		MODER14[1:0]		MODER13[1:0]		MODER12[1:0]		MODER11[1:0]		MODER10[1:0]		MODER9[1:0]		MODER8[1:0]		MODER7[1:0]		MODER6[1:0]		MODER5[1:0]		MODER4[1:0]		MODER3[1:0]		MODER2[1:0]		MODER1[1:0]		MODER0[1:0]	
MASK																					1	1			1	1	1	1	1	1		
VALUE																					0	0			0	0	0	0	0	0		

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

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

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

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ODR	Pin 31	Pin 30	Pin 29	Pin 28	Pin 27	Pin 26	Pin 25	Pin 24	Pin 23	Pin 22	Pin 21	Pin 20	Pin 19	Pin 18	Pin 17	Pin 16	Pin 15	Pin 14	Pin 13	Pin 12	Pin 11	Pin 10	Pin 9	Pin 8	Pin 7	Pin 6	Pin 5	Pin 4	Pin 3	Pin 2	Pin 1	Pin 0
Value																																

Value written to PORTC ODR in order to pull down all rows: 0x0 [0000] (in HEX)

Value written to PORTC ODR in order to pull down row 1: 0xE [1110] (in HEX)

Value written to PORTC ODR in order to pull down row 2: 0xD [1101] (in HEX)

Value written to PORTC ODR in order to pull down row 3: 0xB [1011] (in HEX)

Value written to PORTC ODR in order to pull down row 4: 0x7 [0111] (in HEX)

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

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ODR	Pin 31	Pin 30	Pin 29	Pin 28	Pin 27	Pin 26	Pin 25	Pin 24	Pin 23	Pin 22	Pin 21	Pin 20	Pin 19	Pin 18	Pin 17	Pin 16	Pin 15	Pin 14	Pin 13	Pin 12	Pin 11	Pin 10	Pin 9	Pin 8	Pin 7	Pin 6	Pin 5	Pin 4	Pin 3	Pin 2	Pin 1	Pin 0
Value																																

Mask to check if a button from column 1 has been pressed: 0x2C [101100] (in HEX)

Mask to check if a button from column 2 has been pressed: 0x2A [101010] (in HEX)

Mask to check if a button from column 3 has been pressed: 0x26 [100110] (in HEX)

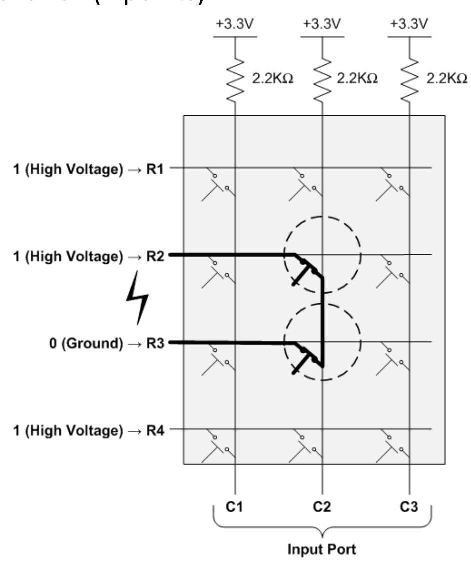
Mask to check if a button from column 4 has been pressed: 0xE [001110] (in HEX)

All columns 1: 0x2E [101110]

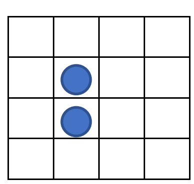
Post-Lab Questions (20 points) RESPONSES ON NEXT PAGE

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

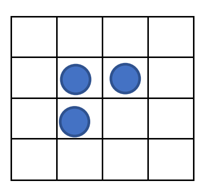
1. How is software debouncing implemented in your program? (3 points)
2. What do we mean when we say that the STM32L4's internal pull-up resistors are too weak for this application? (5 points)
3. When multiple keys are pressed, there could be a short circuit (as shown in the figure). How to configure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



4. 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.

202 Lab 4 Post-Lab Questions

1) Software debouncing is used in the form of a delay subrouting to allow the capacitors on the board to discharge.

2) The internal pull-up capability is too weak to pull-up enough voltage for the needed application. External pull-up is used to fix this.

3) There could be more logic checks in the software to ensure that nothing is written to the TeraTerm when multiple buttons are pressed.

4) Case 1: Yes.
   Case 2: No.

   The current scan algorithm cannot detect when multiple buttons are pressed in the same row AND in the same column.

   Case 1 has 2 only in the same columns, but in different rows, so it can be checked.

Participation and Contribution

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

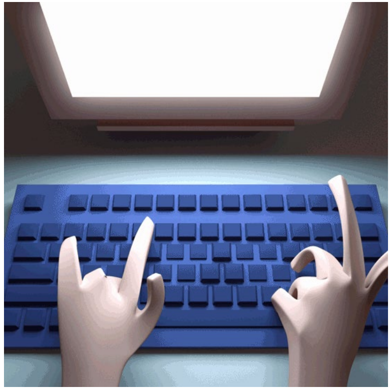
Name	Participation and Contribution
Jeremy	Built Board, Wrote & Debugged Code, Pre- & Post-Lab Questions
Peter	Wrote & Debugged Code

Lab 4: Keypad Scanning

ECE 0202

Lab Outline

1. Download assembly template project from canvas
2. Enable peripheral GPIO pins (4 output, 4 input)
3. Write logic to determine which button is pressed
4. Write that character to TeraTerm (through UART)



- ➢ Show us the working keypad (checkoff)
- ➢ Show us that debouncing is implemented (checkoff)
- ➢ Submit code (e.g. ‘main.s’) on canvas
- ➢ Submit pre-lab and post-lab questions on canvas

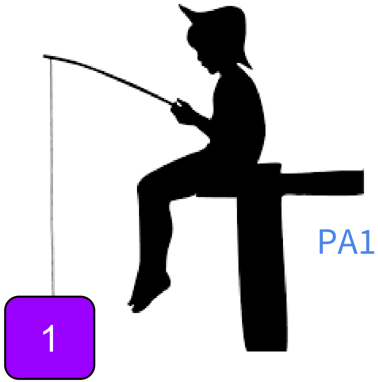
The Circuit

Our output pins drag down the voltage of our input pins when a button is pressed

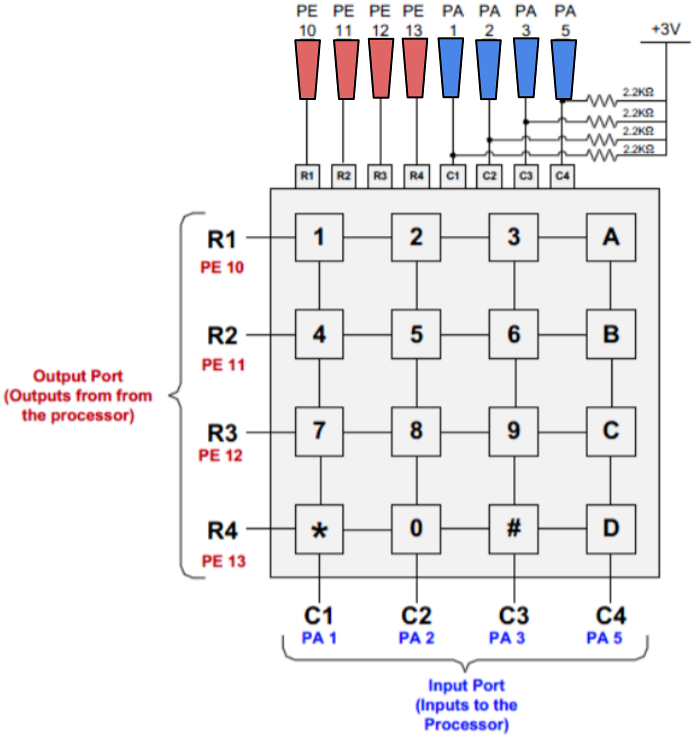
We need external pull-up resistors for our input pins

Not mandatory, but can configure the output pins to open-drain instead of push pull to prevent short circuit

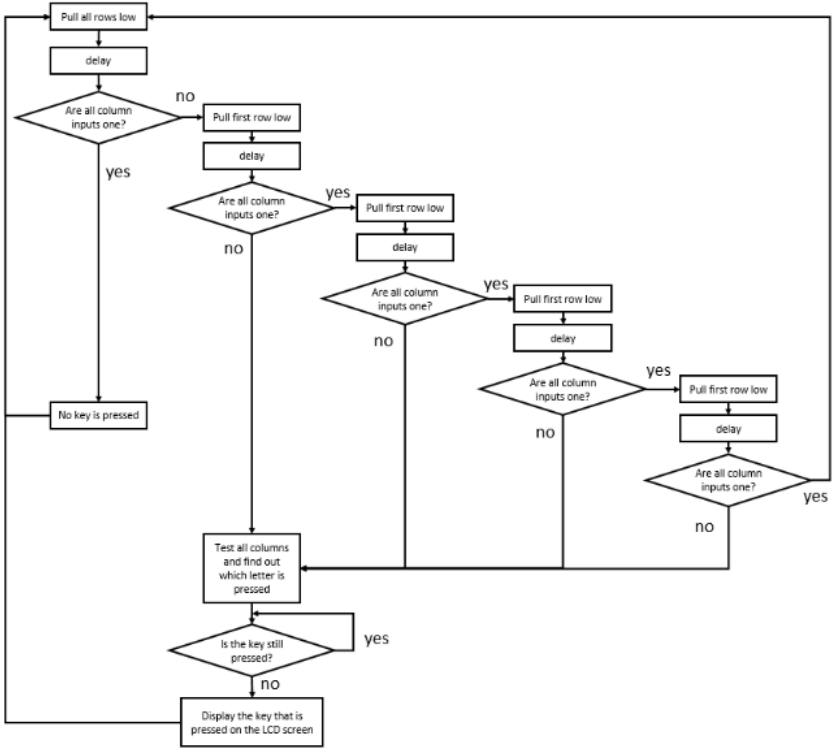
PE10



PA1



Writing the Logic



- Every “split” of the flowchart should be a conditional statement
- Idea is to find the row of the pressed button first, then the column

Delays/Software Debouncing

Delay 1: needed in between each change of the output pins (primarily due to capacitors on the input pins)

...also provide debouncing (trace flowchart)

Delay 2: once the button is pressed, wait until it is not pressed anymore

Writing to Tera Term

Using the help of a function in UART.c, which makes our life easy!

void USART2_Write(uint8_t* buffer, uint32_t nBytes)

1. Argument 1 is a pointer to a buffer of bytes - Load this into R0
2. Argument 2 is how many bytes are in the buffer - Load this into R1
3. Branch and link to USART2_Write

```
    AREA myData, DATA, READWRITE
    ALIGN
str DCB    "ECE0202\r\n", 0
    END
```

Example of how to declare a string of bytes called “str”

```
      AREA myData, DATA, READWRITE
      ALIGN
char1    DCD 43
      END
```

Example of how to declare a single word called “char1”

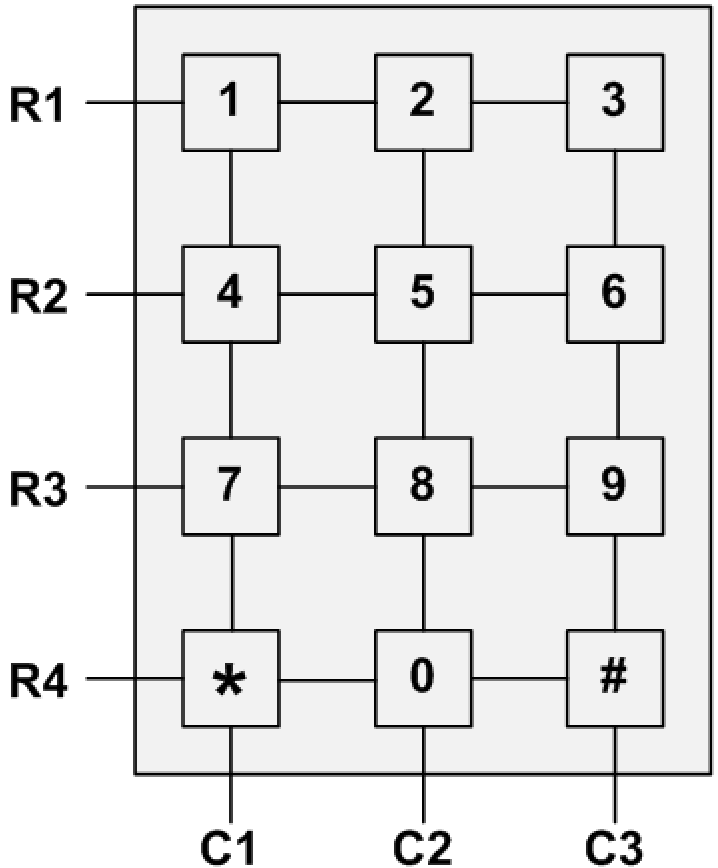
University of Pittsburgh

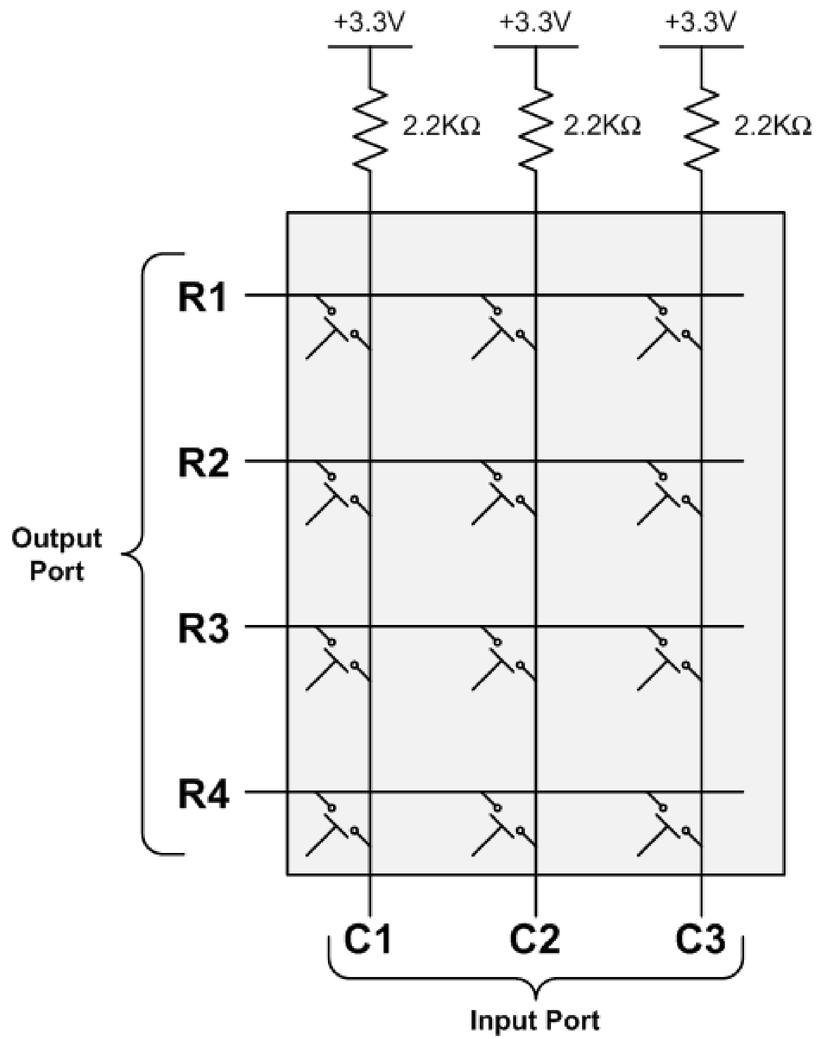
Lab 4 Keypad

Dr. Jingtong Hu
Department of Electrical and Computer Engineering
Swanson School of Engineering
jthu@pitt.edu

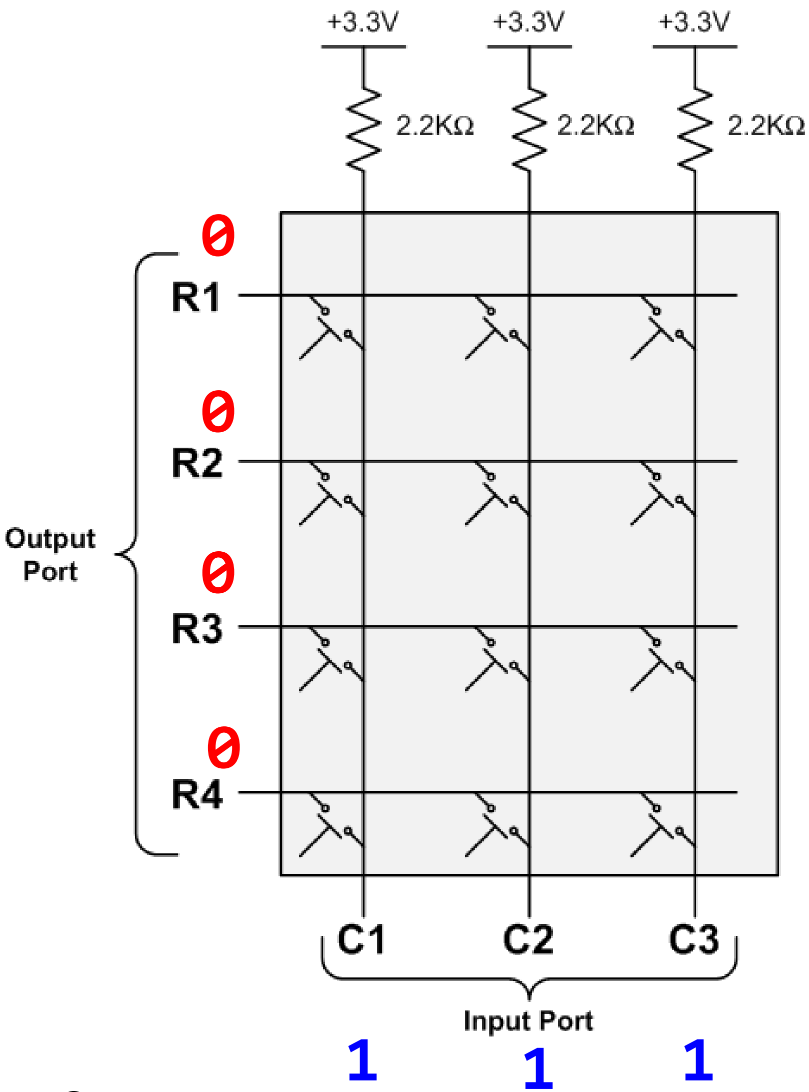
22 November 2022

Keypad Scan


R1
R2
R3
R4
1
2
3
4
5
6
7
8
9
*
0
#
C1
C2
C3


+3.3V
+3.3V
+3.3V
2.2KΩ
2.2KΩ
2.2KΩ
Output Port
R1
R2
R3
R4
C1
C2
C3
Input Port

Keypad Scan

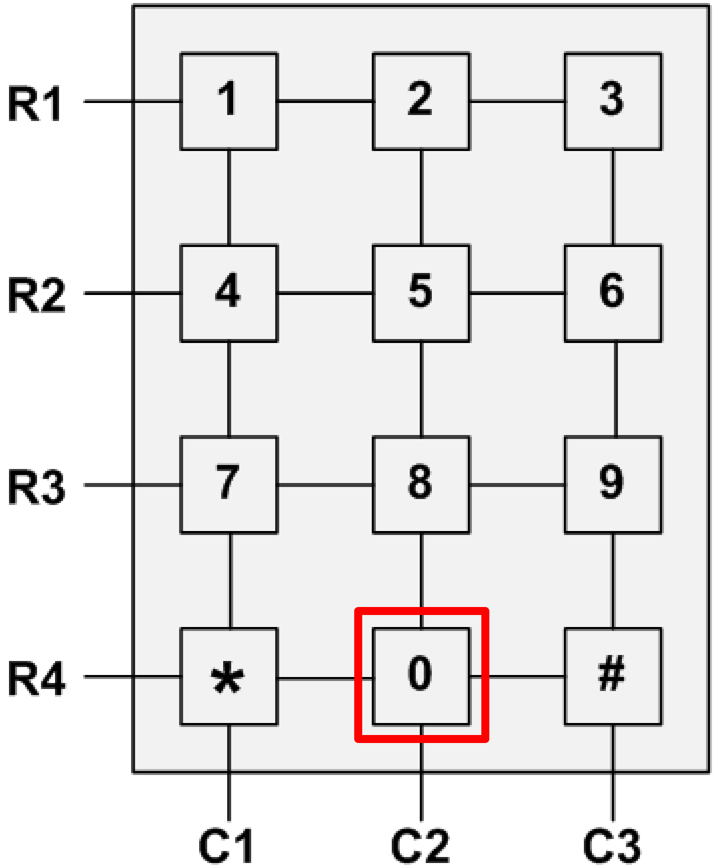


Step 1: Set Output
R1,R2,R3,R4 = 0000

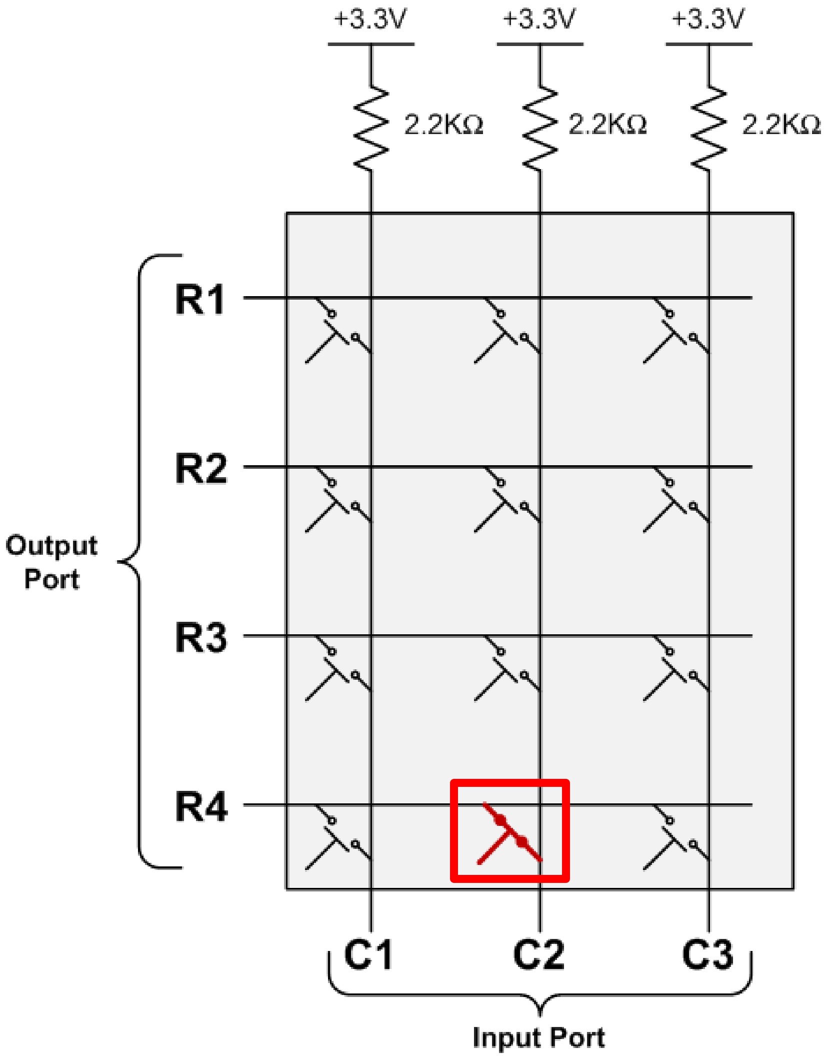
Step 2: Read Input
C1,C2,C3 = 111

⟹ No key pressed

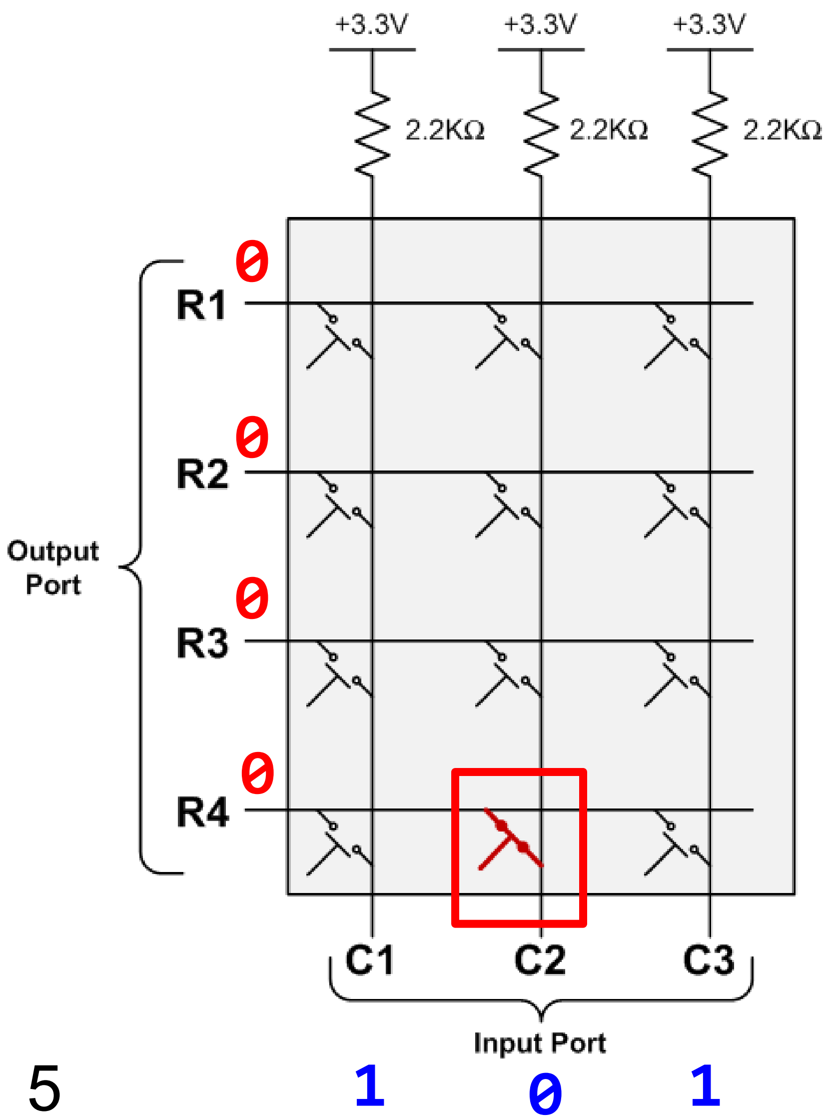
Keypad Scan



Key "0" is pressed



Keypad Scan



Step 1: Set Output

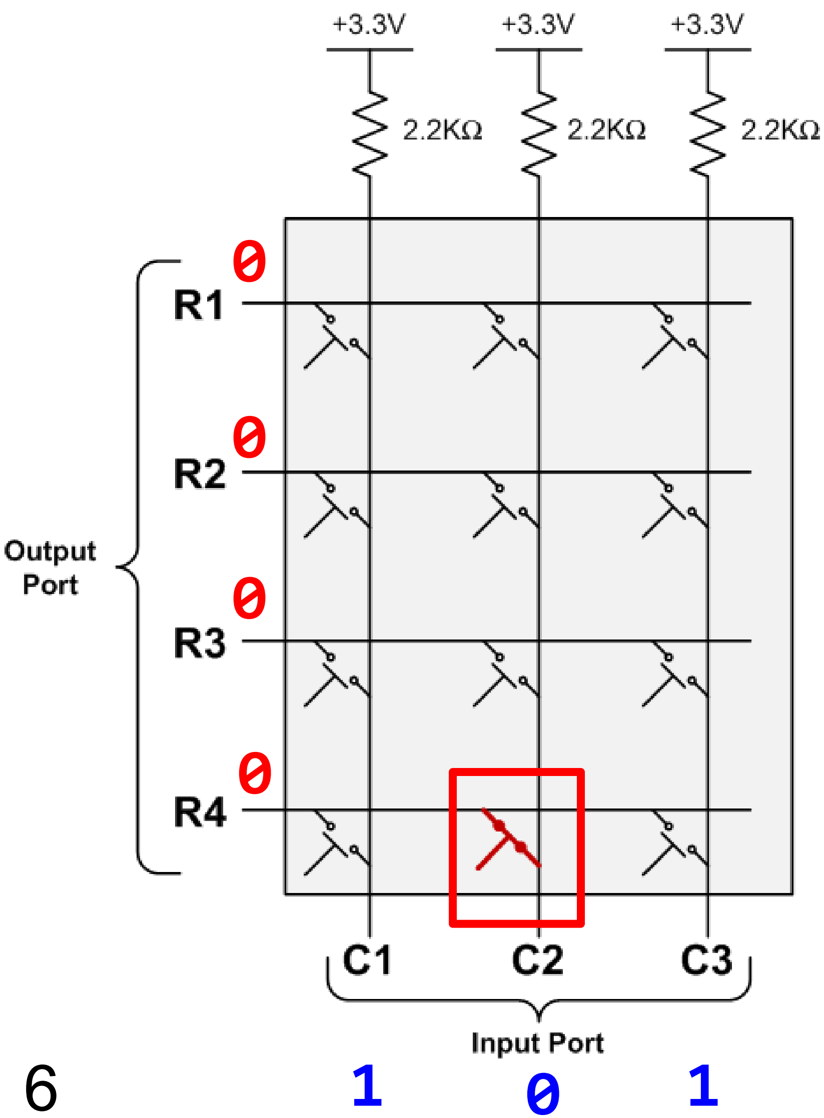
R1,R2,R3,R4 = 0000

Step 2: Read Input

C1,C2,C3 = 101

$\Longrightarrow$ Some key in 2nd column is pressed down

Keypad Scan



Step 1: Set Output
`R1,R2,R3,R4 = 0000`

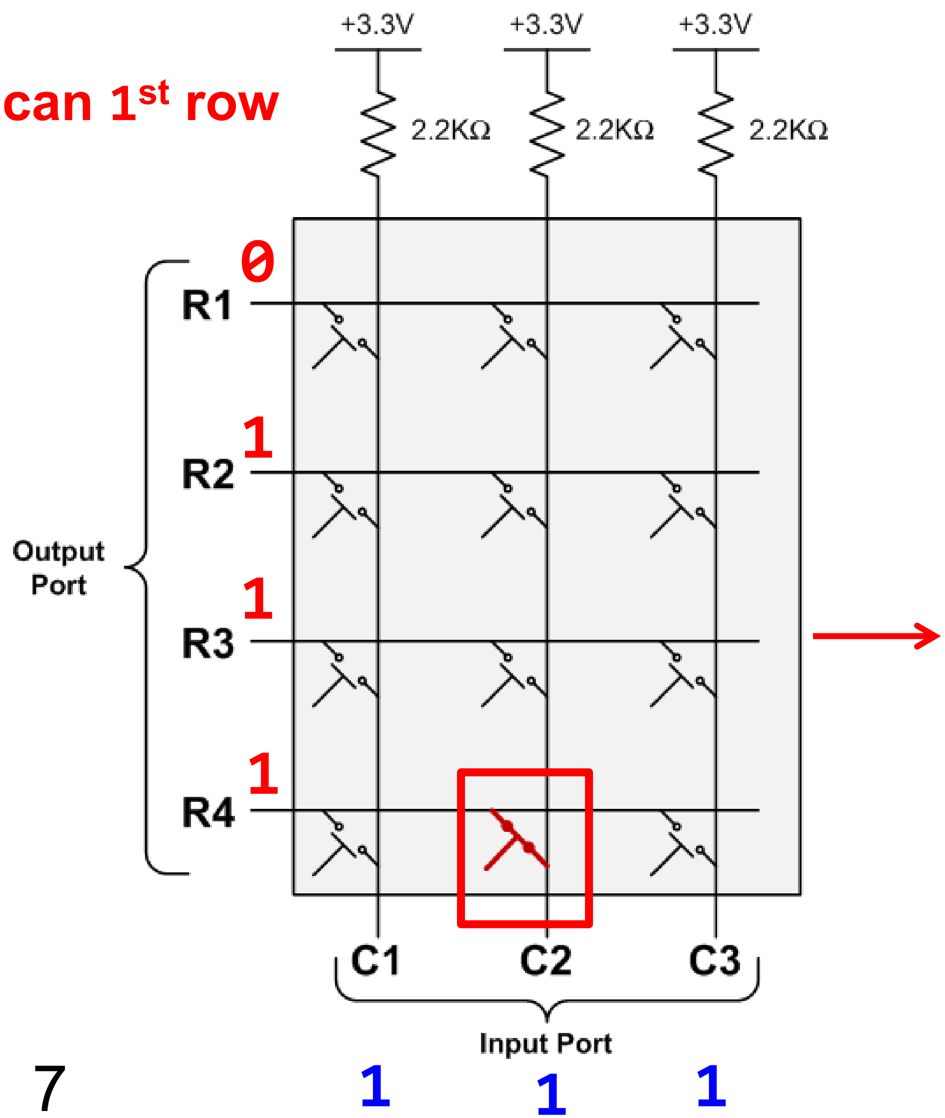
Step 2: Read Input
`C1,C2,C3 = 101`

$\Longrightarrow$ Some key in 2nd column is pressed down

Which one?

Keypad Scan

Scan 1^{st} row



Step 1: Set Output
R1,R2,R3,R4 = 0000

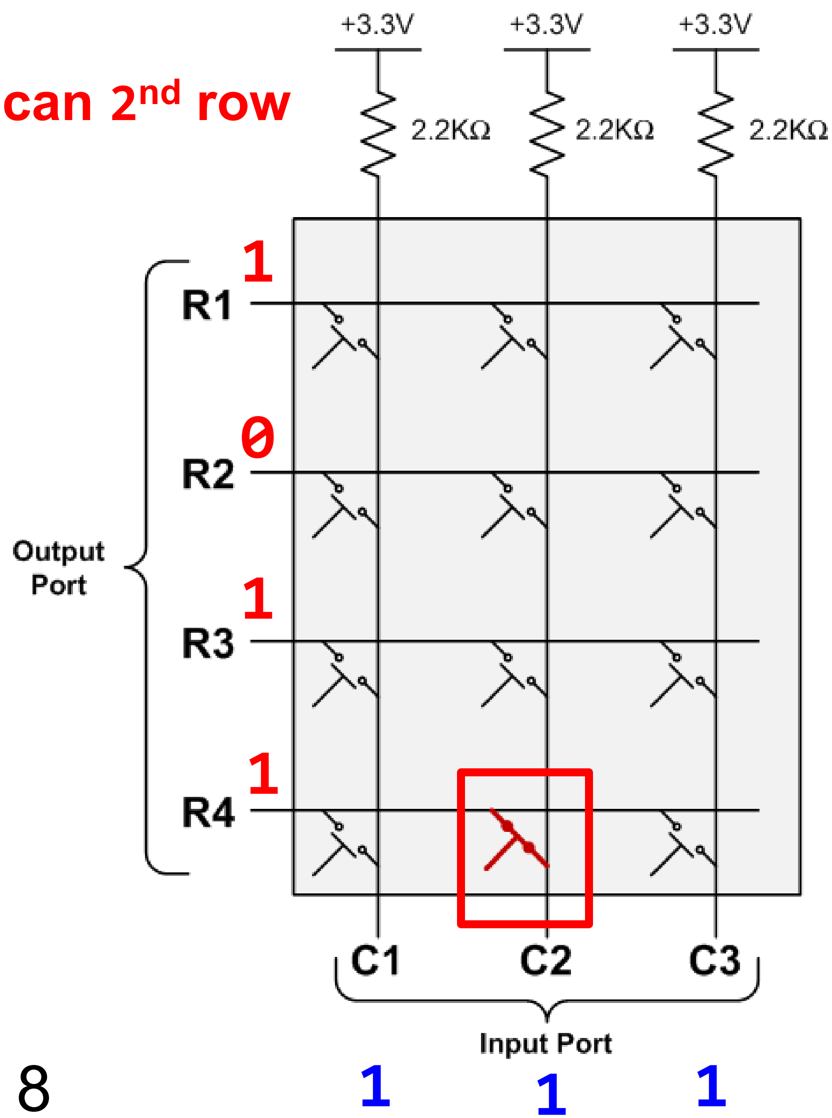
Step 2: Read Input
C1,C2,C3 = 101

Step 3a: Scan 1^{st} row
R1,R2,R3,R4 = 0111
C1,C2,C3 = 111

$\Rightarrow$ No key in 1^{st} row is pressed down

Keypad Scan

Scan 2nd row



Step 1: Set Output

`R1,R2,R3,R4 = 0000`

Step 2: Read Input

`C1,C2,C3 = 101`

→ Step 3b: Scan 2nd row

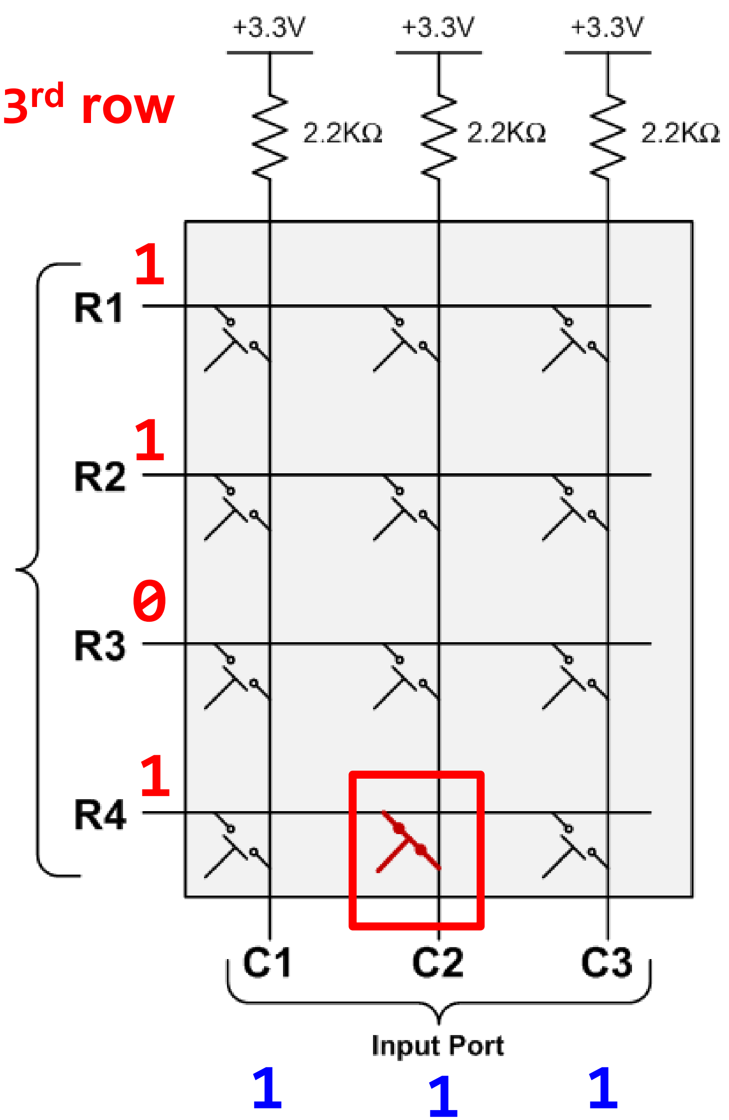
`R1,R2,R3,R4 = 1011`

`C1,C2,C3 = 111`

⟹ `No key in 2nd row is pressed down`

Keypad Scan

Scan 3rd row



Step 1: Set Output
R1,R2,R3,R4 = 0000

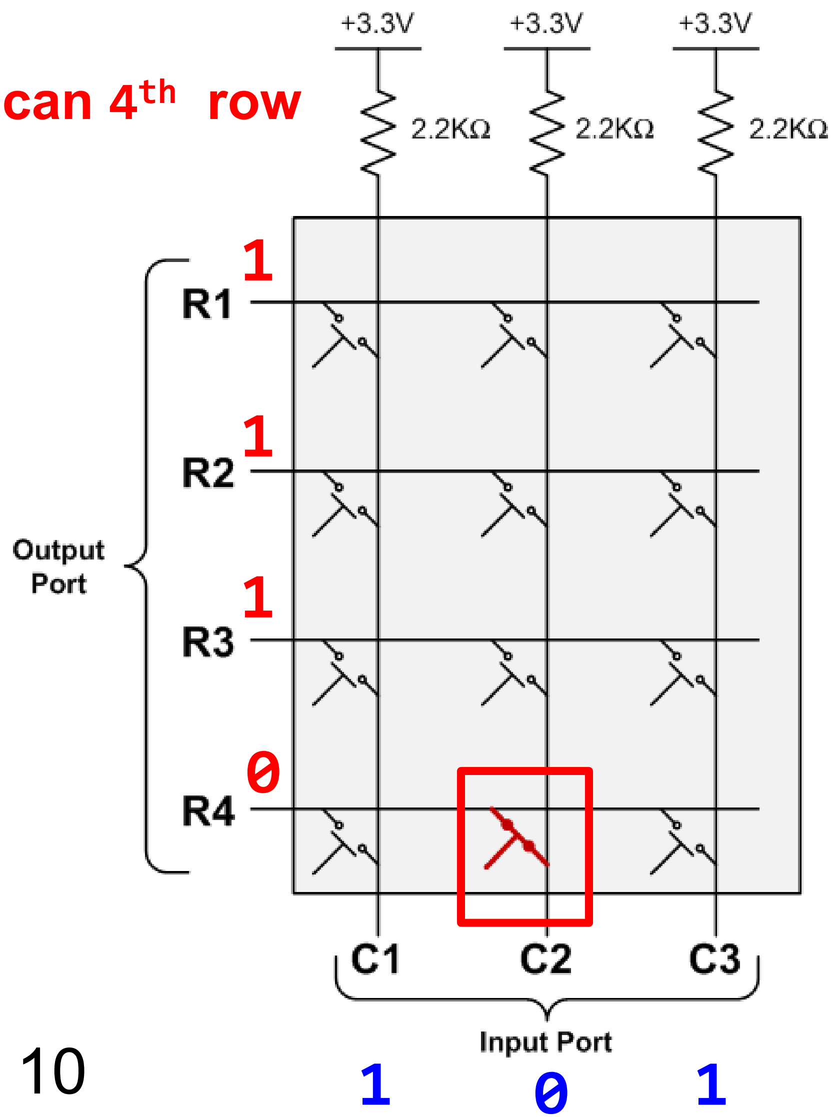
Step 2: Read Input
C1,C2,C3 = 101

→ Step 3c: Scan 3rd row
R1,R2,R3,R4 = 1101
C1,C2,C3 = 111

⇒ No key in 3rd row
is pressed down

Keypad Scan

Scan 4^{th} row



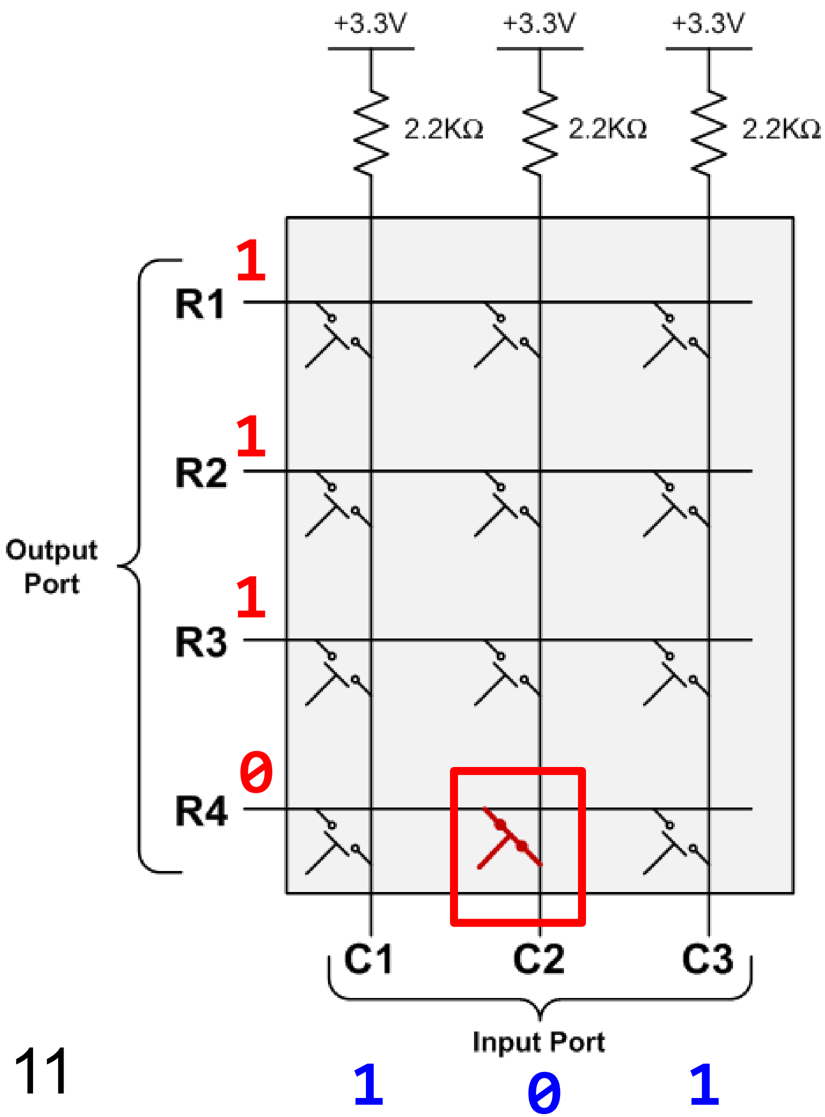
Step 1: Set Output
`R1,R2,R3,R4 = 0000`

Step 2: Read Input
`C1,C2,C3 = 101`

→ Step 3d: Scan 4^{th} row
`R1,R2,R3,R4 = 1110`
`C1,C2,C3 = 101`

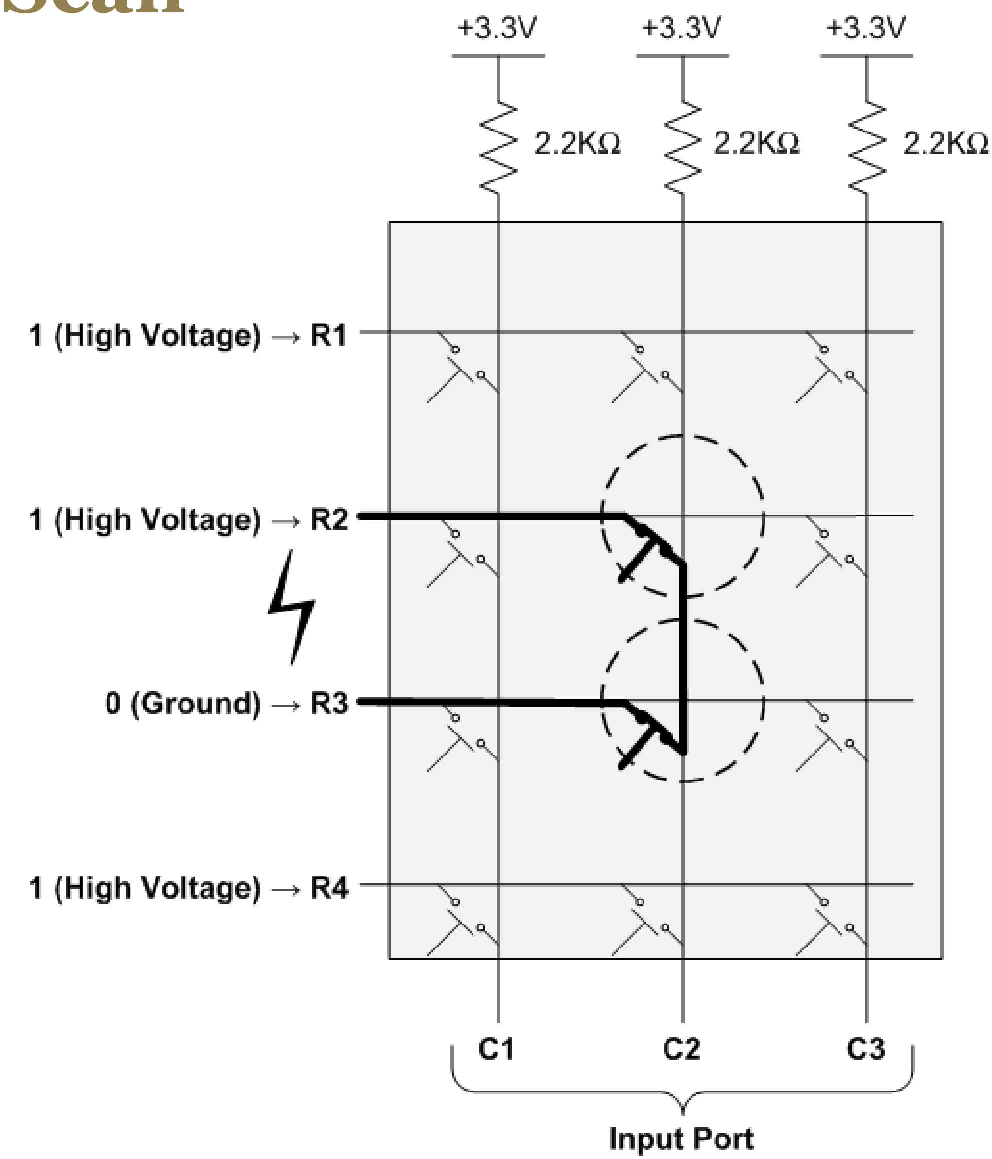
⟹ key in 4^{th} row is pressed down

Keypad Scan



⟹ Key pressed is located at the second column and the fourth row.

Keypad Scan


+3.3V
+3.3V
+3.3V
2.2KΩ
2.2KΩ
2.2KΩ
1 (High Voltage) → R1
1 (High Voltage) → R2
0 (Ground) → R3
1 (High Voltage) → R4
C1
C2
C3
Input Port

I/O Debouncing

- Example signal when a button is pressed