Embedded Systems with ARM Cortex-M3 Microcontrollers in Assembly Language and C

Chapter 14
GPIO

Dr. Yifeng Zhu
Electrical and Computer Engineering
University of Maine

Spring 2015

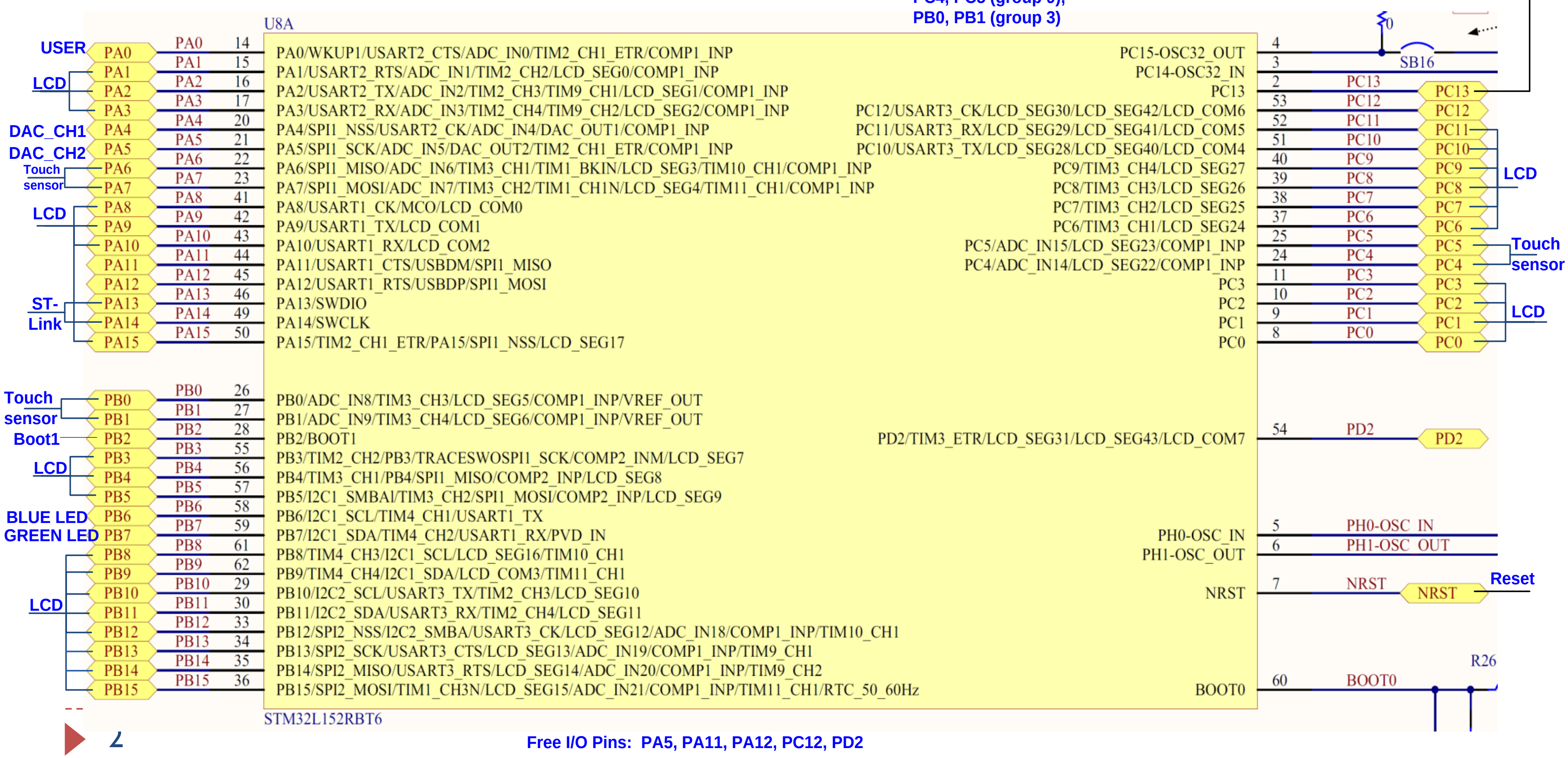
General-Purpose Input and Output (GPIO)

- Each pin can be configured as digital input and digital output to interface external devices or on-chip peripherals

USER Pushbutton: PA0
GREEN LED: PB7
BLUE LED: PB6
Touch sensor:
PA6, PA7 (group 2),
PC4, PC5 (group 9),
PB0, PB1 (group 3)

If JP1 is on, PC13 is used for DD_CNT_EN

All PINs for touch sensors are not extended.



Free I/O Pins: PA5, PA11, PA12, PC12, PD2

General-Purpose Input and Output (GPIO)

- This chapter focus on digital input and digital output
  - Interfacing with on-chip peripherals will be discussed later.
- Each GPIO port has
  - Four 32-bit control registers.
    - GPIO_MODER (digital input, digital output, alternative function, analog input/output)
    - GPIO_OTYPER (output type: push-pull or open-drain)
    - GPIO_OSPEEDR(speed, i.e., slew rate)
    - GPIO_PUPDR (pull-up/pull-down)
  - One 32-bit input data register (GPIO_IDR) and one 32-bit ouput data register (GPIO_ODR)
    - Each bit holds the input/ouput value of one GPIO pin
  - Two 32-bit alternative function selection registers (GPIO_AFRH, GPIO_AFRL)
- Clock to GPIO are turned off by default to save power
  - Software program needs to turn on the clock

GPIO: Logic Voltage Level

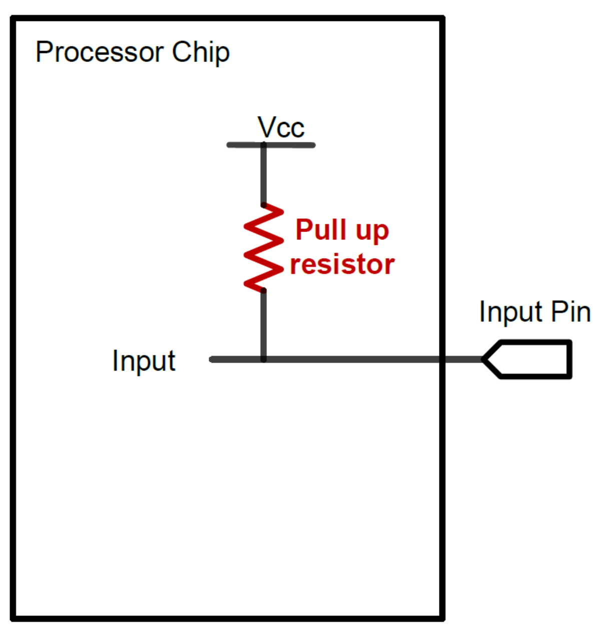
- Thresholds

Technology	Low voltage	High voltage	
CMOS	0 to $\frac{1}{3}V_{dd}$	$\frac{2}{3}V_{dd}$ to V_{dd}	V_{dd} = supply voltage
TTL	0 V to 0.8 V	2 V to V_{cc}	$V_{cc} = 5V \pm 10\%$

- Logic Level
  - Active High
    - Logic 1 = High voltage
    - Logic 0 = Low voltage
  - Active Low
    - Logic 1 = Low voltage
    - Logic 0 = High voltage

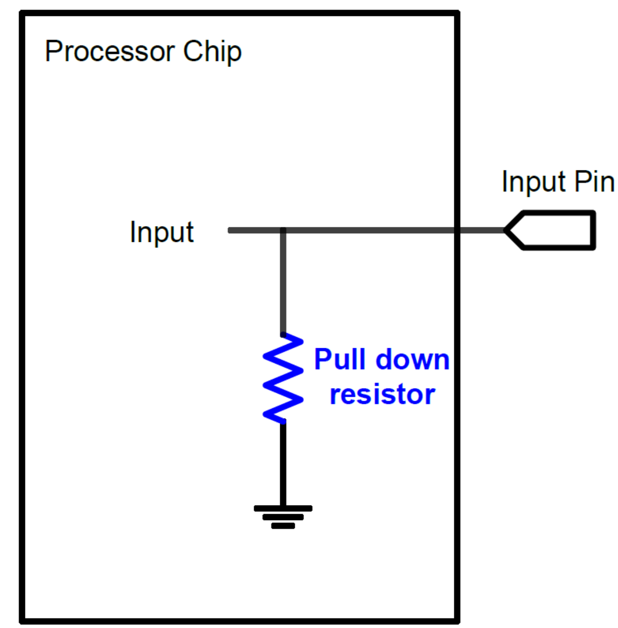
GPIO Input:
Pull Up and Pull Down

- A digital input can have three states: High, Low, and High-Impedance (also called floating, tri-stated, HiZ)



Pull-Up

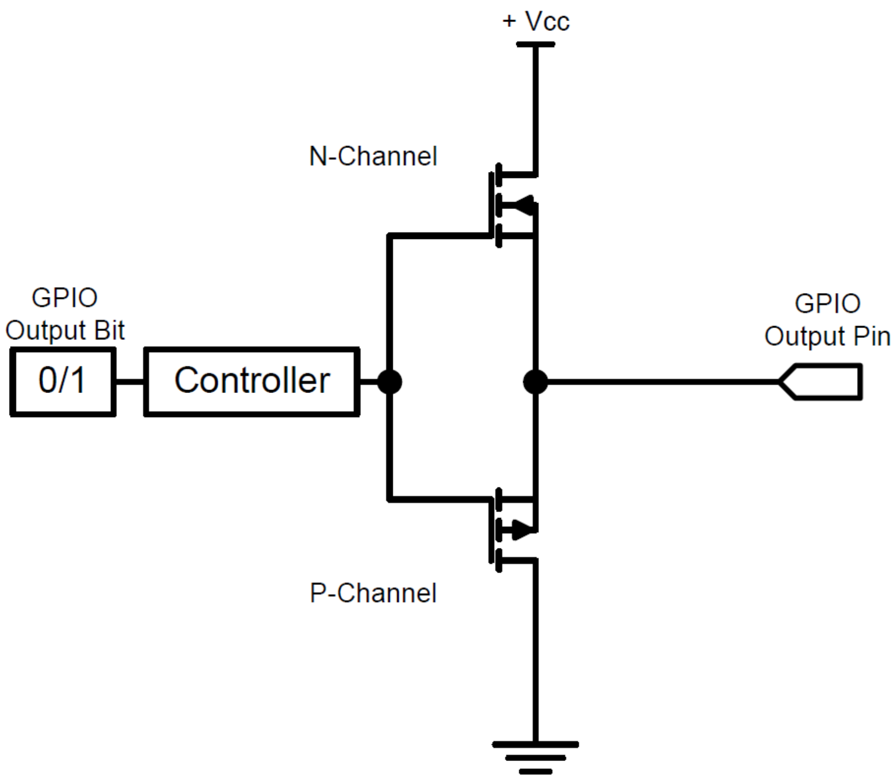
If external input is HiZ, the input is read as a valid HIGH.

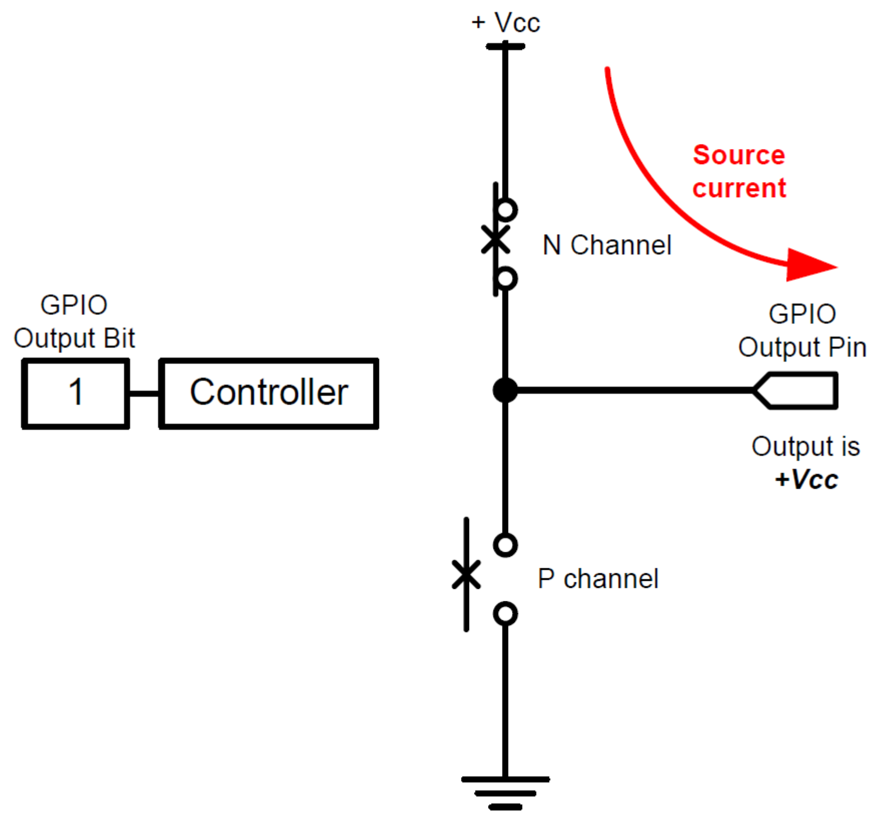


Pull-Down

If external input is HiZ, the input is read as a valid LOW.

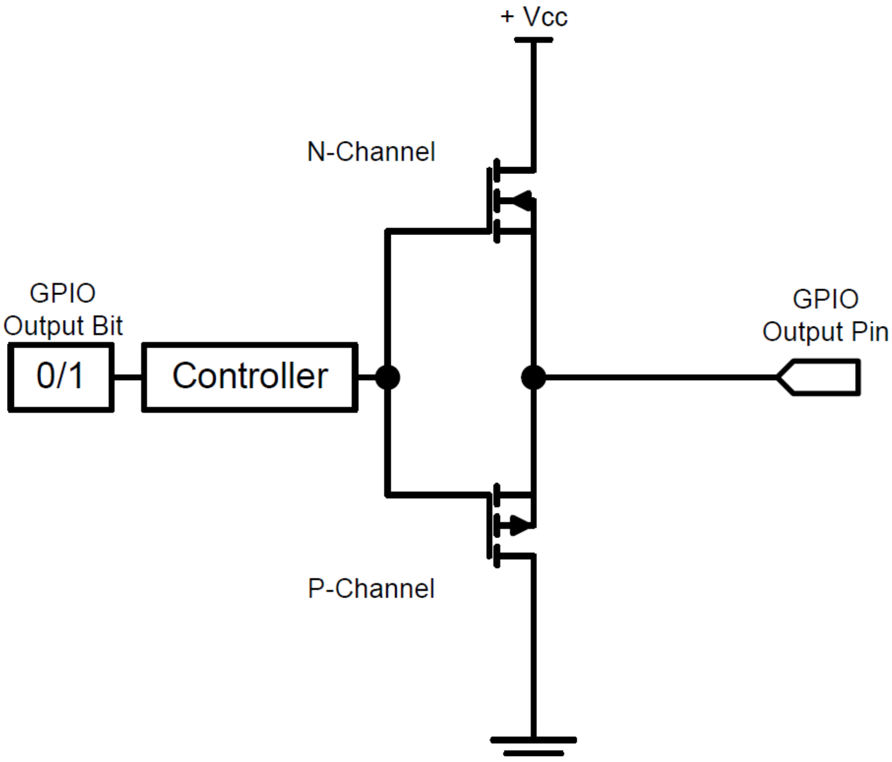
GPIO Output: Push-Pull

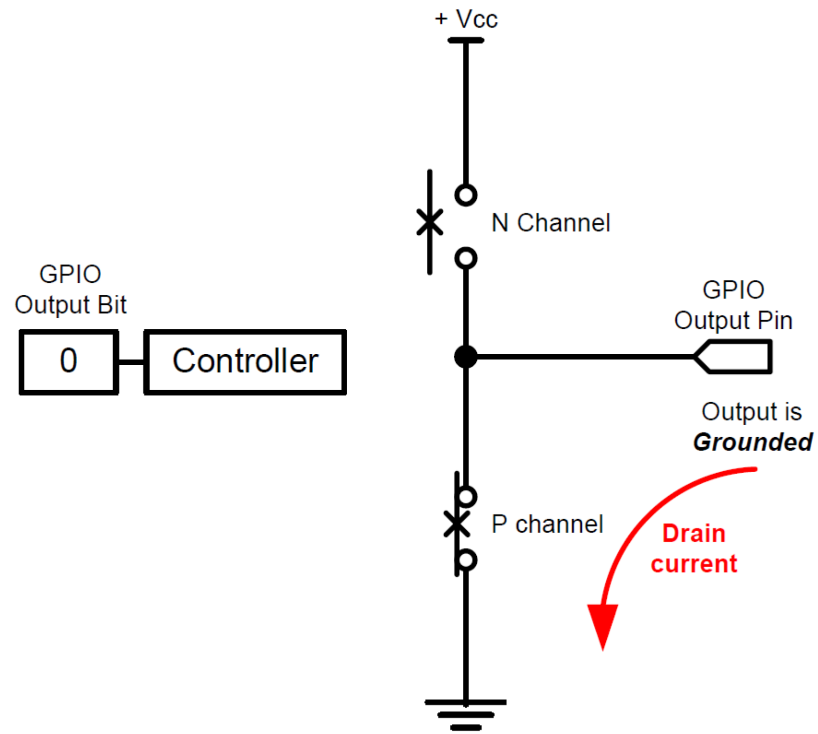




GPIO Output = 1
Source current to external circuit

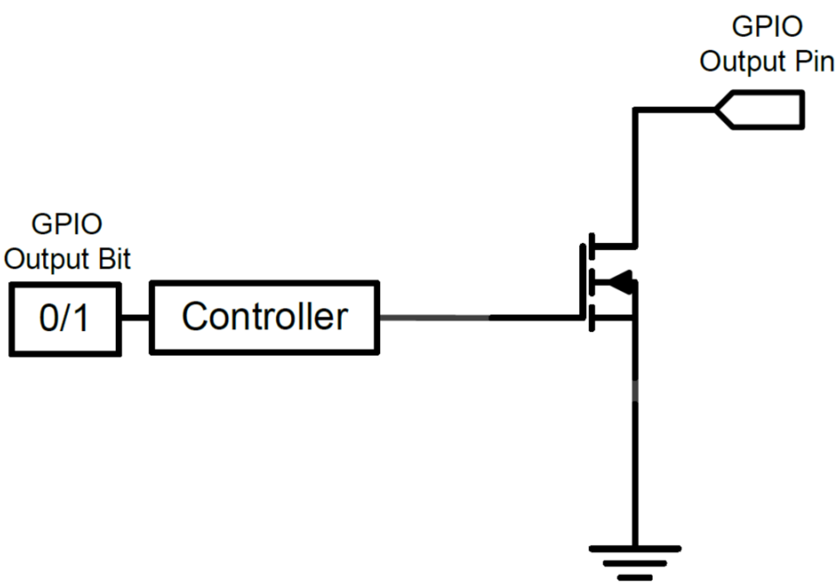
GPIO Output: Push-Pull

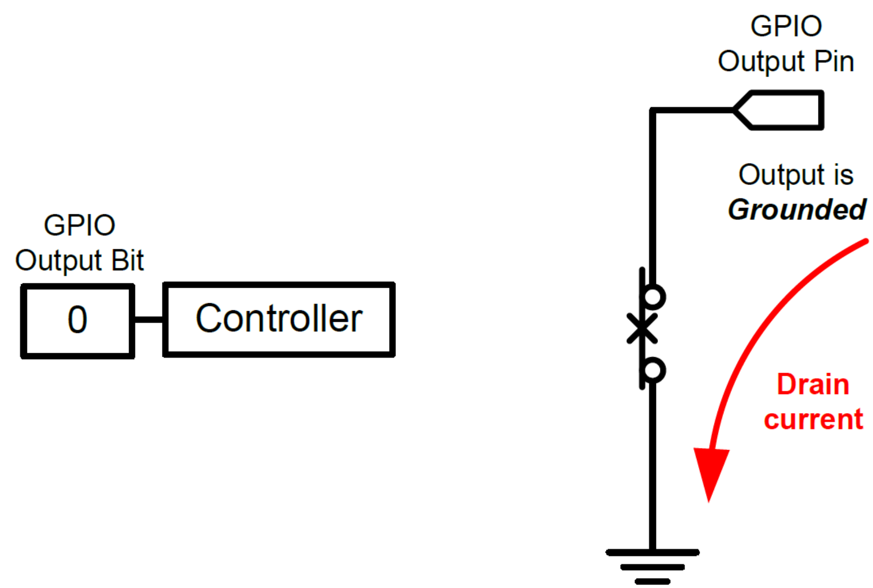




GPIO Output = 0
Drain current from external circuit

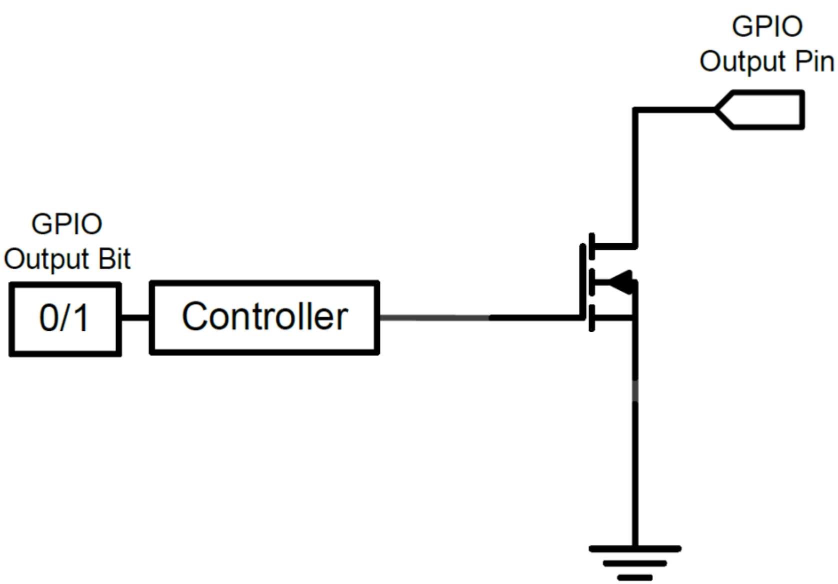
GPIO Output: Open-Drain

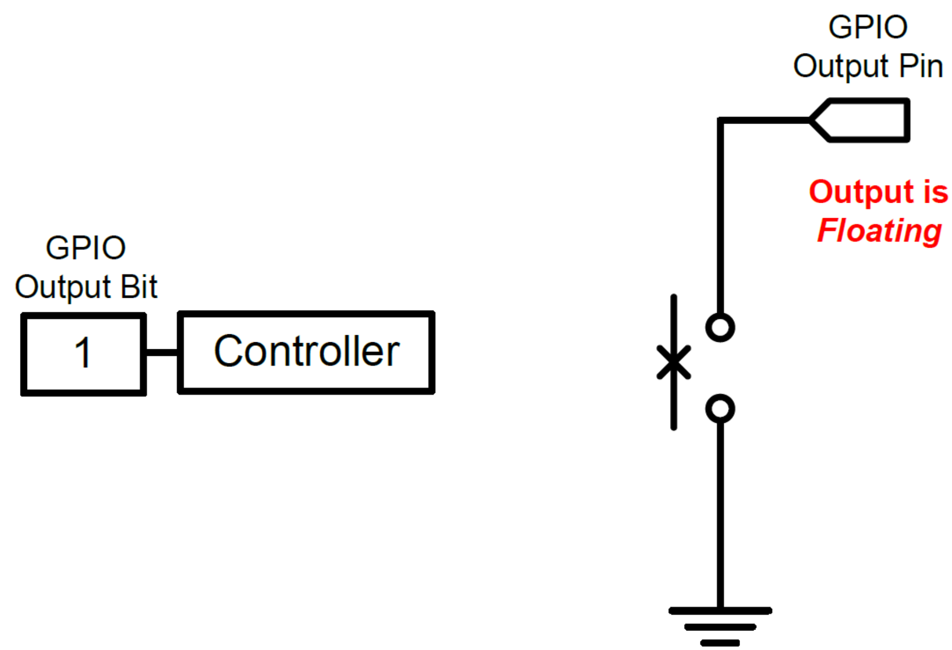




GPIO Output = 0
Drain current from external circuit

GPIO Output: Open-Drain





Output = 1

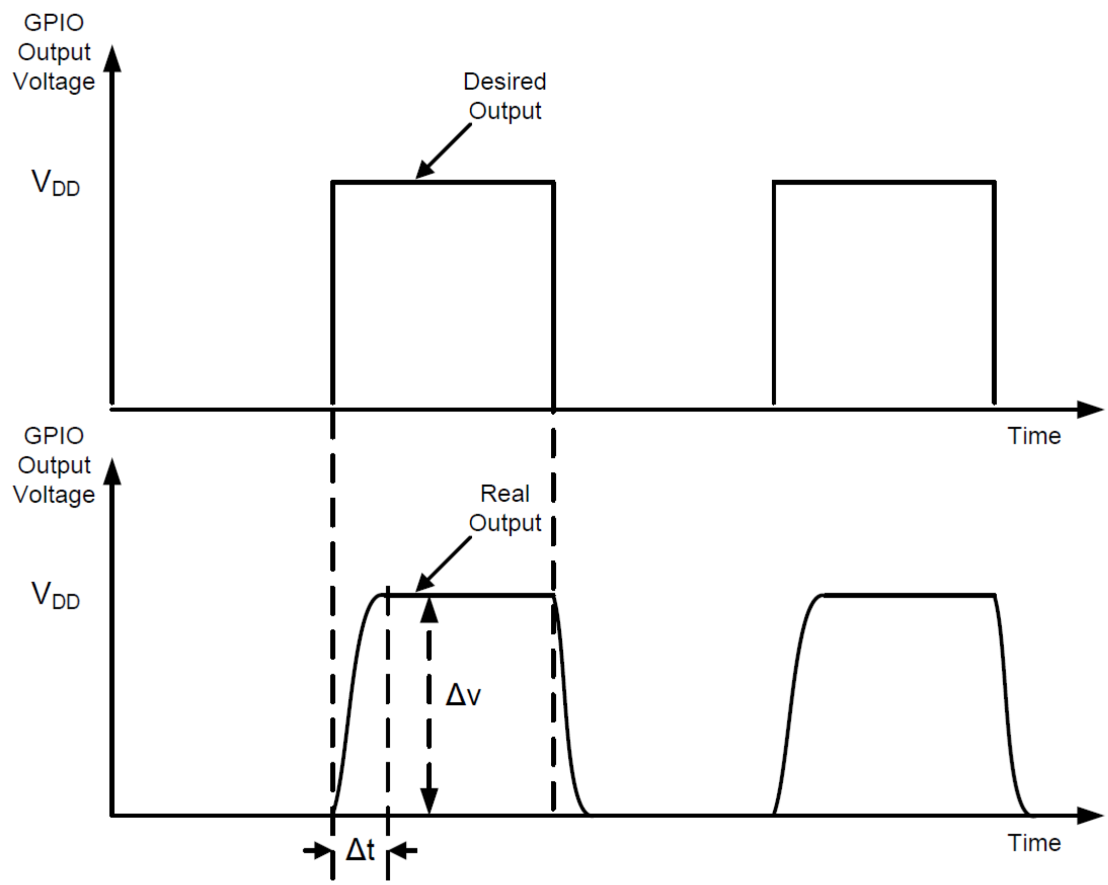
GPIO Pin has high-impedance to external circuit

Slew Rate

Slew Rate:
Maximum rate of change of the output voltage

$$Slew\ Rate = max\left(\frac{\Delta V}{\Delta t}\right)$$

A high slew rate allows the output to be toggled at a fast speed.



GPIO Initialization

- Turn on the clock to the GPIO Port (e.g. Port B)

```
RCC->AHBENR |= RCC_AHBENR_GPIOBEN; Reset and Clock Control (RCC)
```

- Configure GPIO mode, output type, speed, pull-up/pull-down

```
typedef struct
{
  __IO uint32_t MODER;
  __IO uint16_t OTYPER;
  uint16_t RESERVED0;
  __IO uint32_t OSPEEDR;
  __IO uint32_t PUPDR;
  __IO uint16_t IDR;
  uint16_t RESERVED1;
  __IO uint16_t ODR;
  uint16_t RESERVED2;
  __IO uint16_t BSRRL; /* BSRR register is split to 2 * 16-bit fields BSRRL */
  __IO uint16_t BSRRH; /* BSRR register is split to 2 * 16-bit fields BSRRH */
  __IO uint32_t LCKR;
  __IO uint32_t AFR[2];
} GPIO_TypeDef;
#define PERIPH_BASE           ((uint32_t)0x40000000)
#define AHBPERIPH_BASE        (PERIPH_BASE + 0x20000)
#define GPIOB_BASE            (AHBPERIPH_BASE + 0x0400)
#define GPIOB                 ((GPIO_TypeDef *) GPIOB_BASE)
```

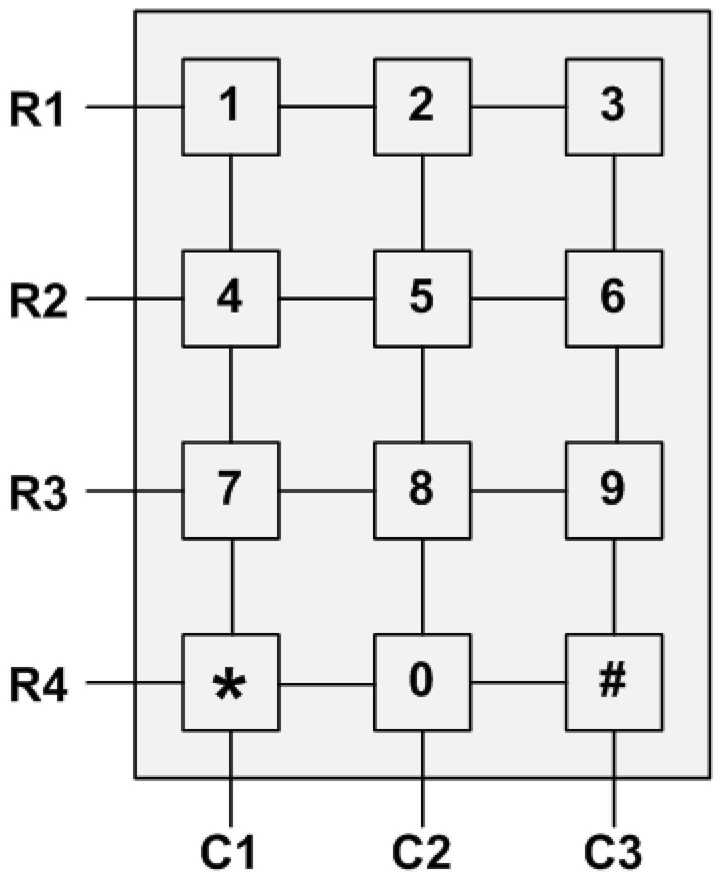
GPIO Digital Input/Output

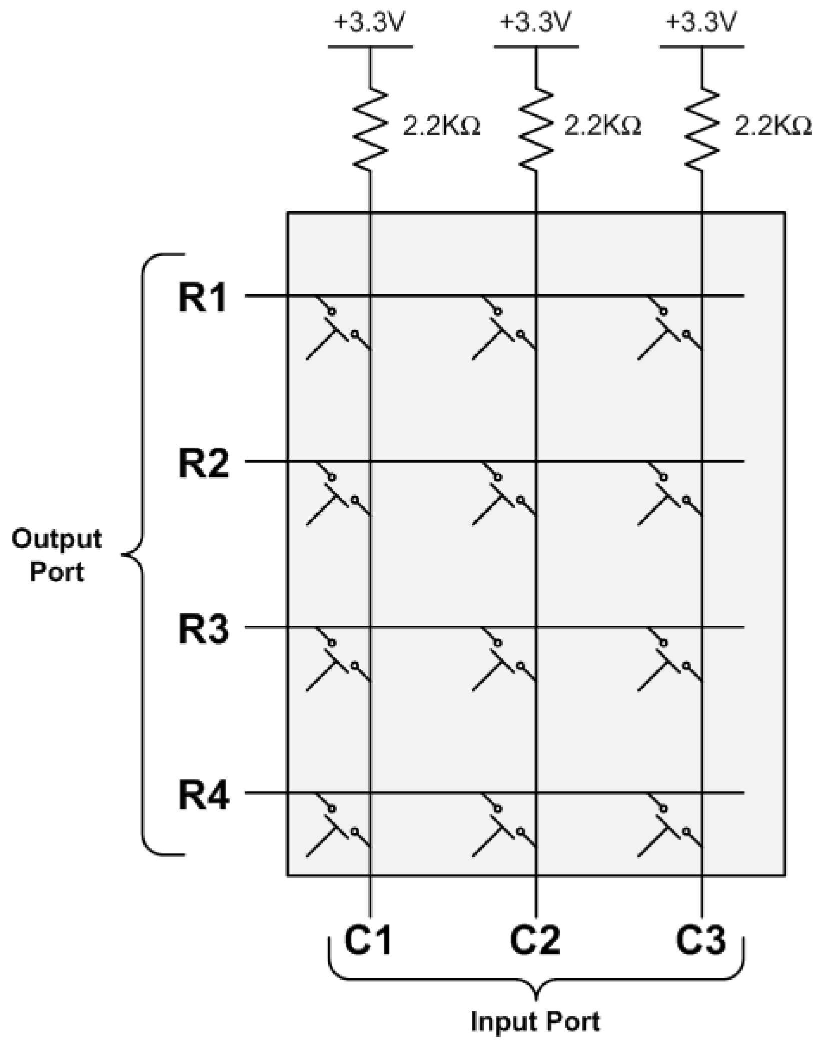
- Pin number starts with 0
- Set Pin 7 of Port B:
  - `GPIOB->ODR |= (1<<7);`
- Clear Pin 7 of Port B :
  - `GPIOB->ODR ^= (1<<7);`
- Read Pin 7 input
  - bit = `GPIOB->IDR` & (1<<k);

Using mask:

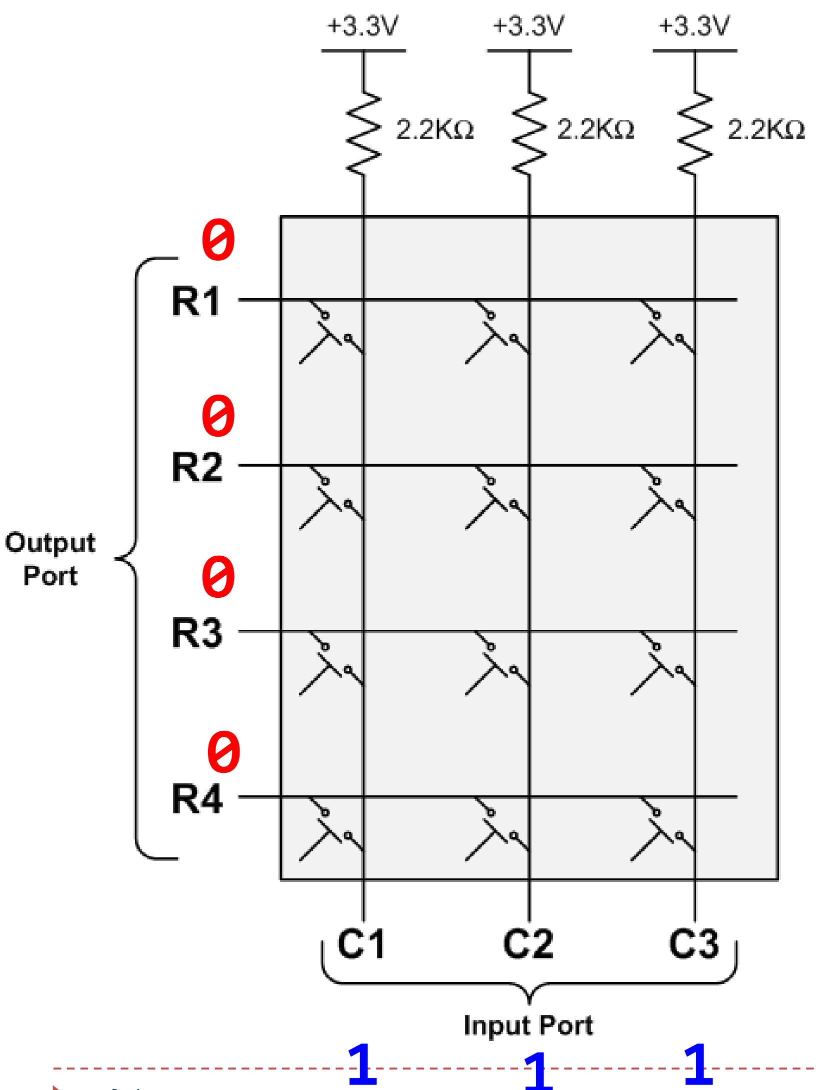
```
Mask = 1<< 7;
GPIOB->ODR |= Mask;

GPIOB->ODR ^= Mask;

bit = GPIOB->IDR & Mask;
```

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

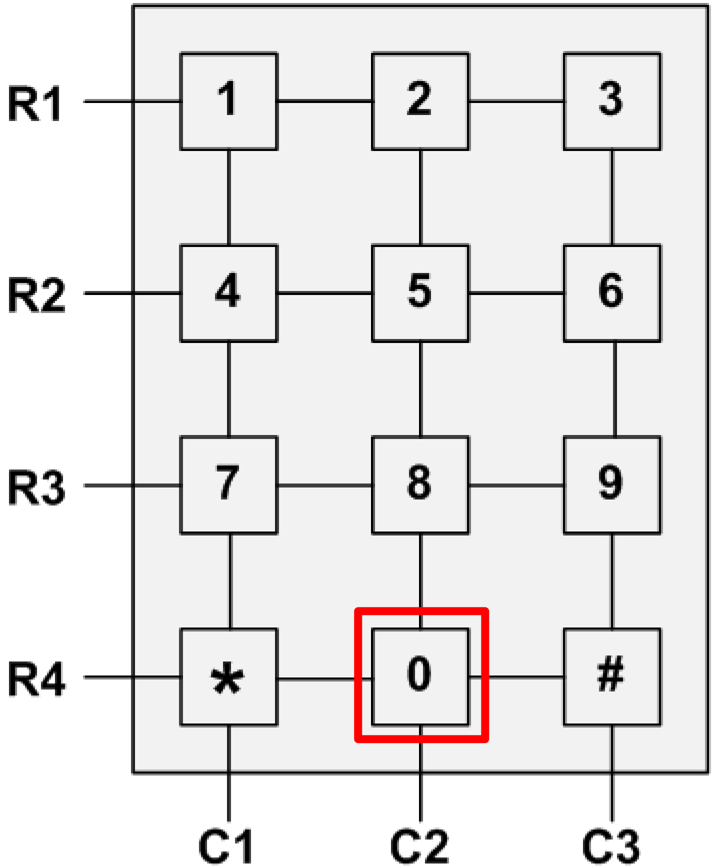


Step 1: Set Output
`R1,R2,R3,R4 = 0000`

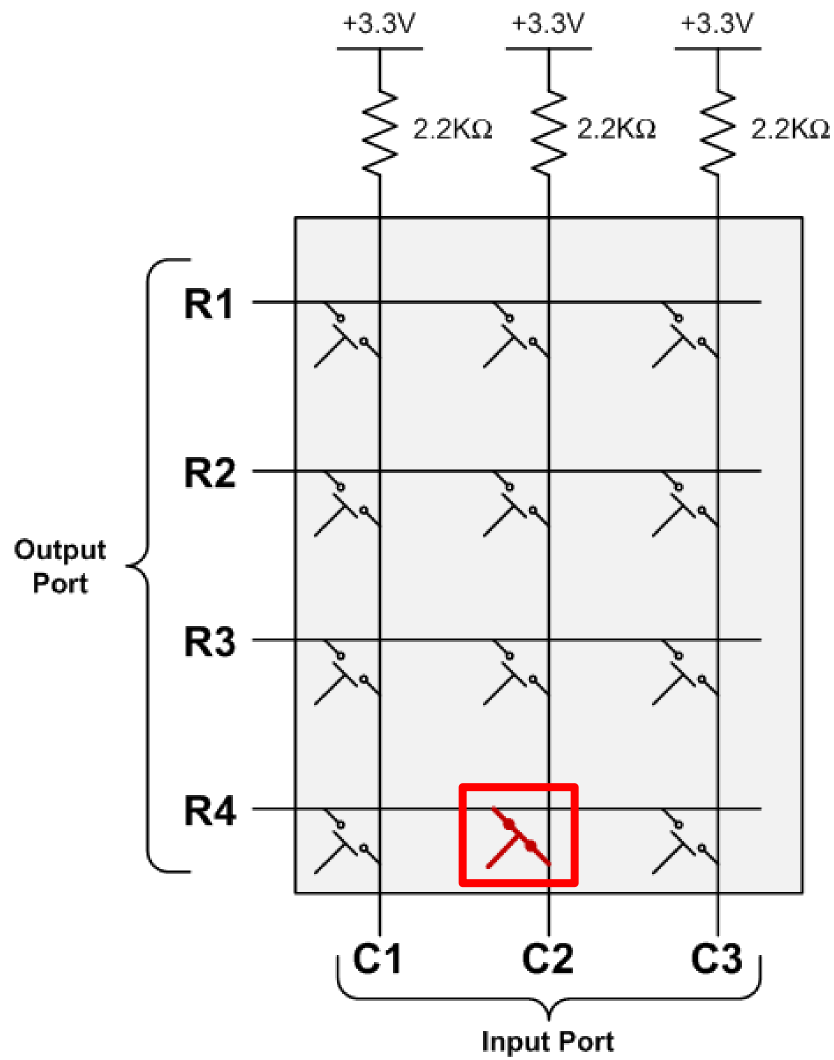
Step 2: Read Input
`C1,C2,C3 = 111`

⟹ No key pressed

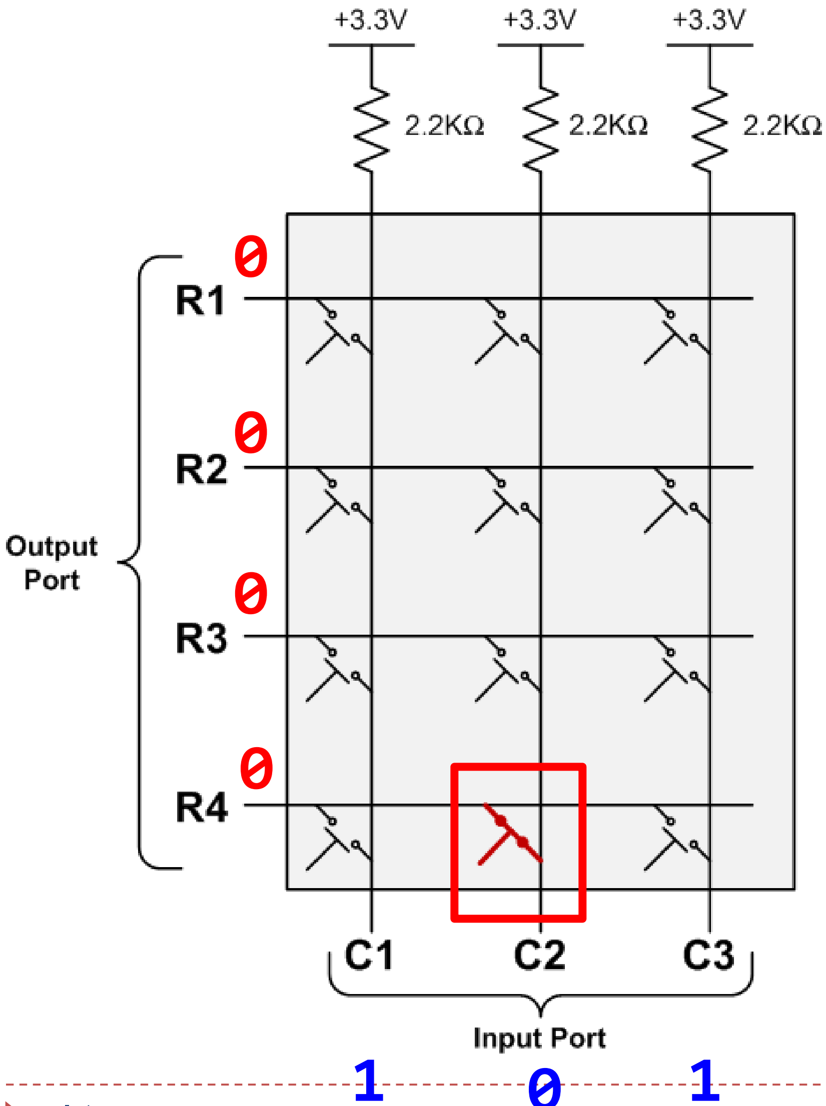
Keypad Scan



Key “0” is pressed



Keypad Scan



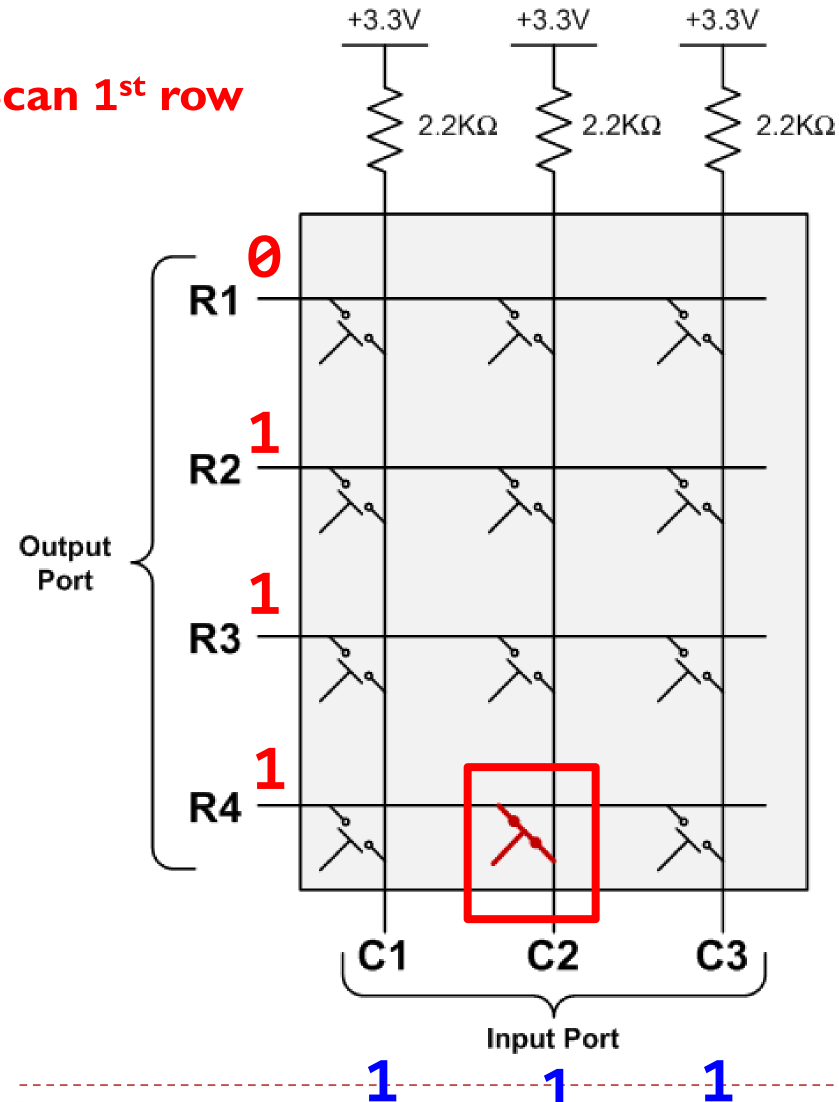
Step 1: Set Output
R1,R2,R3,R4 = 0000

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

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

Keypad Scan

Scan 1st row



Step 1: Set Output

R1,R2,R3,R4 = 0000

Step 2: Read Input

C1,C2,C3 = 101

→ Step 3a: Scan 1st row

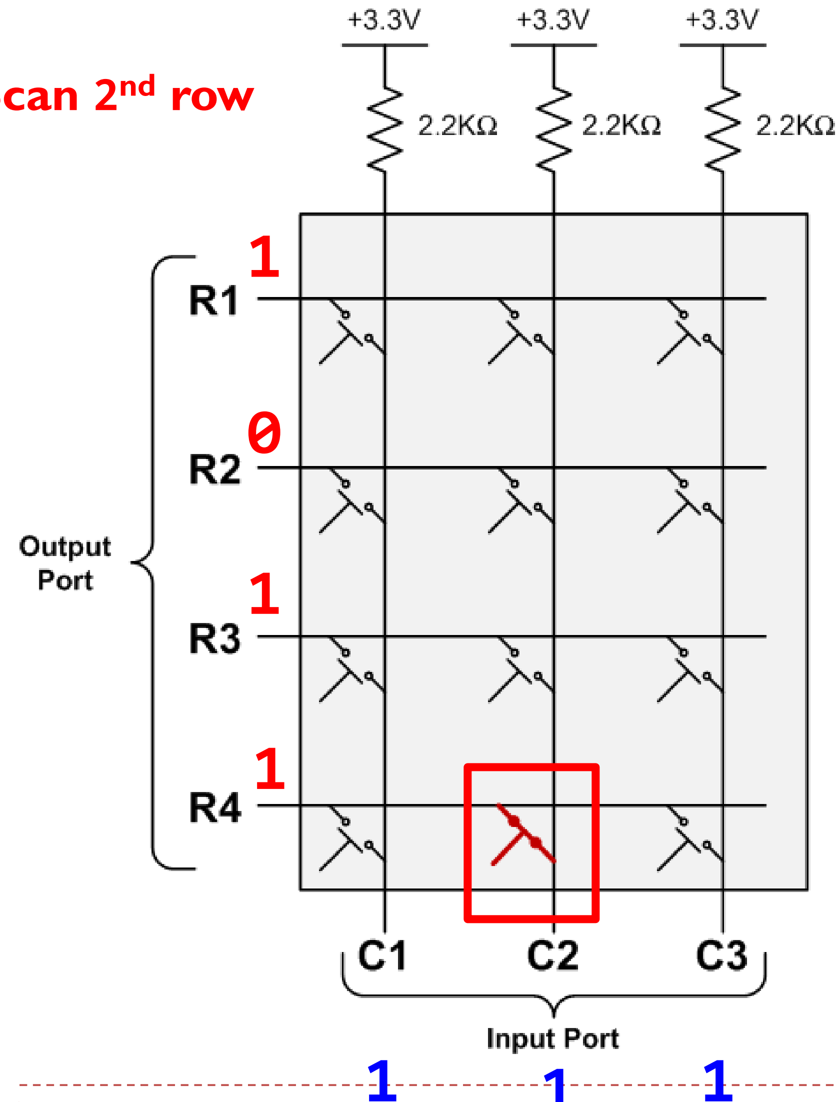
R1,R2,R3,R4 = 0111

C1,C2,C3 = 111

⇒ No key in 1st row is pressed down

Keypad Scan

Scan 2nd row



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

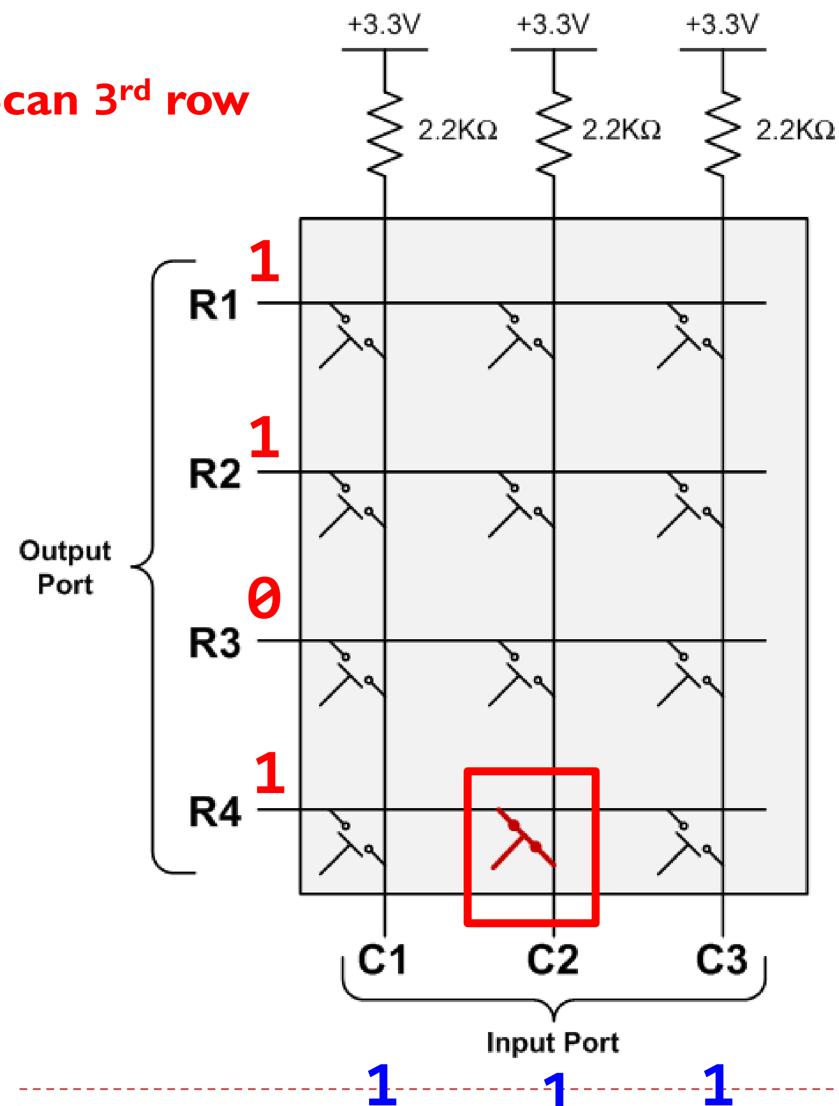
Step 2: Read Input
C1,C2,C3 = 101

→ 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

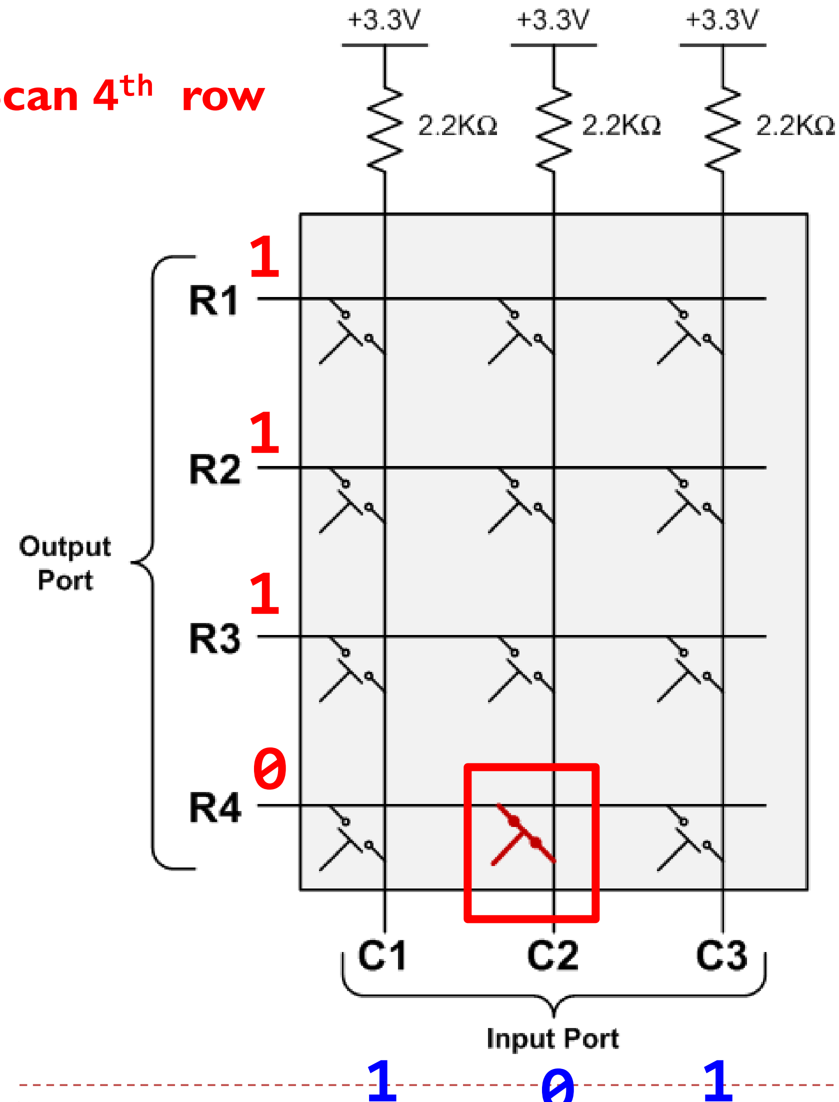
R1,R2,R3,R4 = 1101

C1,C2,C3 = 111

⟹ No key in 3rd row is pressed down

Keypad Scan

Scan 4th row



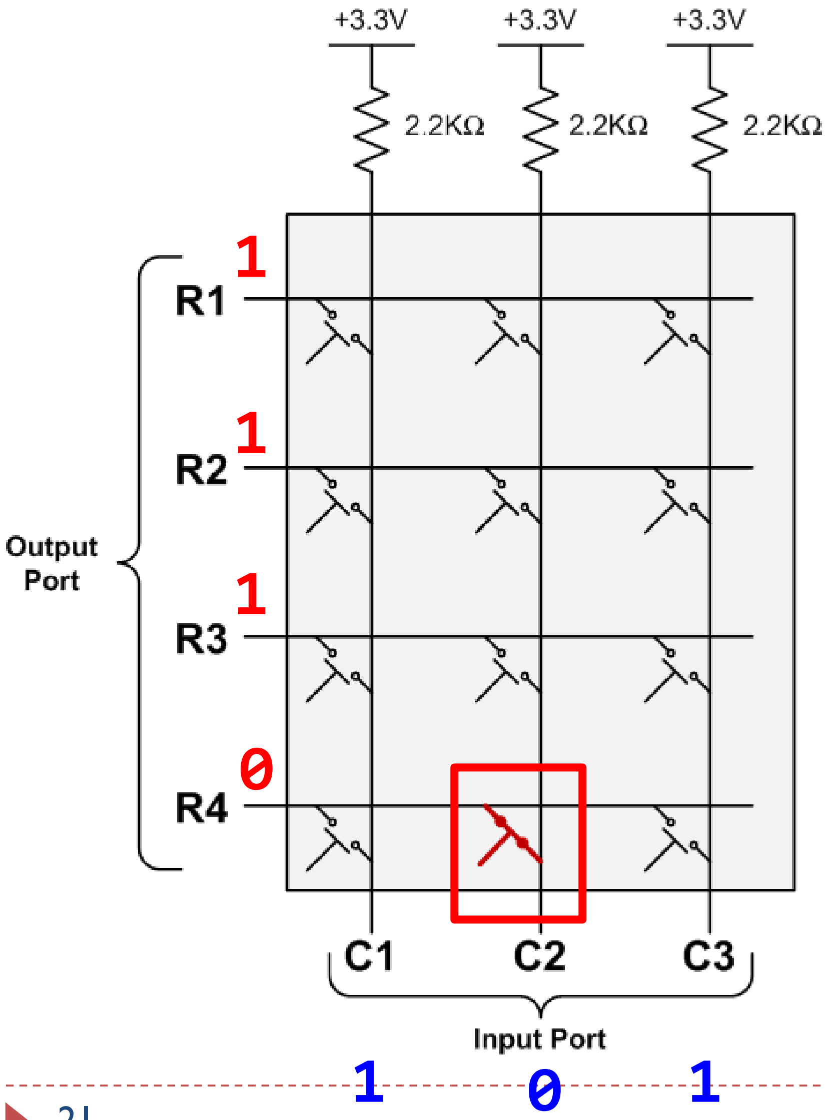
Step 1: Set Output
R1,R2,R3,R4 = 0000

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

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

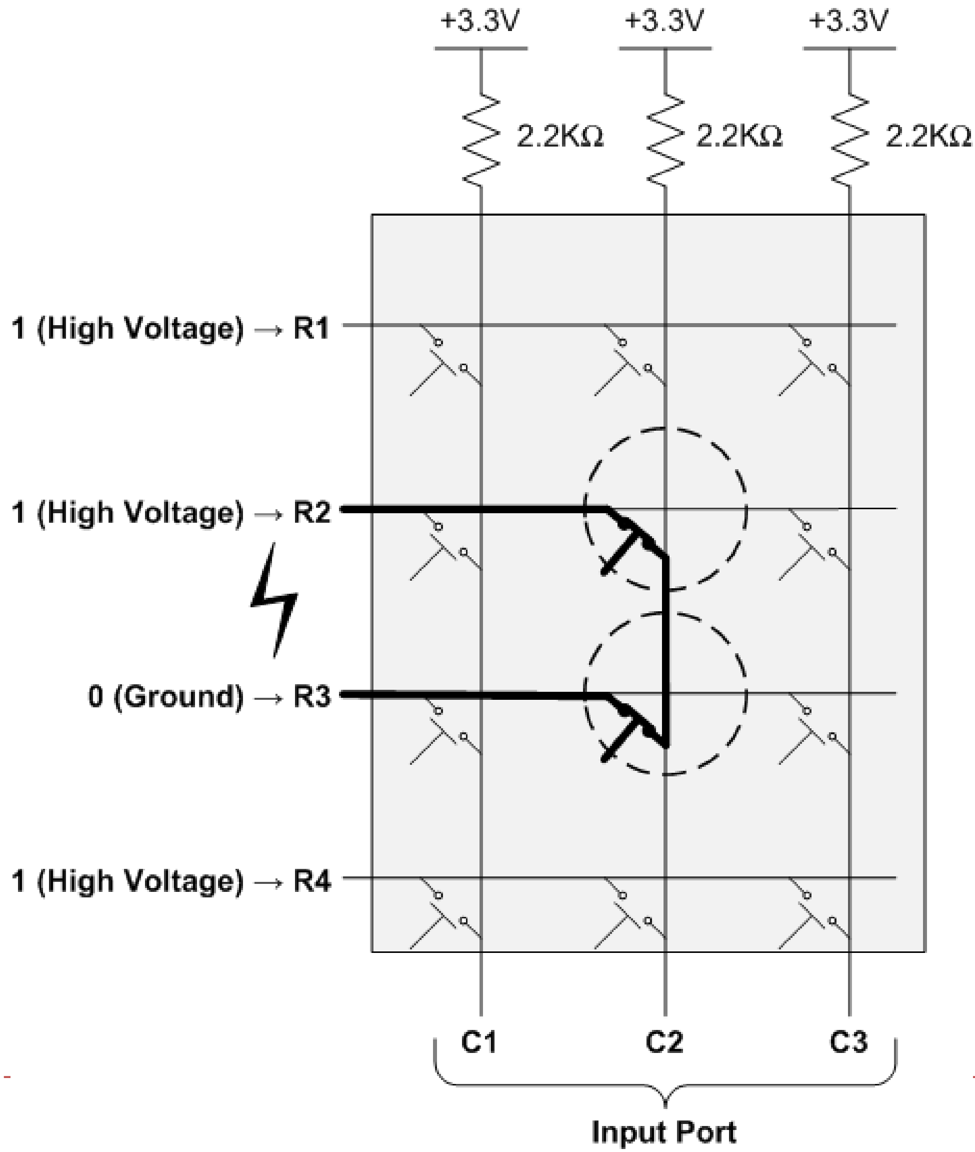
⇒ key in 4th row
is pressed down

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