Bern University
of Applied Sciences

Lab - 01

GPIO Three-State Input/Output

Andreas Habegger | Adrian Steiner
BTS3230 | Version 1.0.0 of 17.03.2025

Please be aware that the content is subject to change at any time. For the latest version, please check the website.

In this lab you will focus on the GPIO I/O configuration. So far, you have used binary logic levels on your output and input pins. However, some logic elements use three-state logic as an interface. You will implement a simple transmitter and receiver application that combines three-state logic input and output handling.

Objectives

- Three-State “GPIO” operation
- Electrical characteristics of the “GPIO” pins

Outcomes

- Gain a better understanding of the GPIO peripherals
- Understand the GPIO output driver
- Understand the difference between open-drain and push/pull.
- Understand what three-state means and how it can be used to transfer data
- How to find the necessary information from the schematic to control elements such as RGB LEDs

Description

In the previous GPIO and Edge Detection exercises you used binary logic states where the zero represented a low voltage equal to “GND” and the one represented a high voltage equal to “VDD”. However, this is a rather inaccurate view of the connection between the abstract digital domain and the real physical domain. Therefore, in this exercise we will further investigate the conversion from physical to digital and vice versa.

A three-state transmitter and receiver application combines the previously mentioned topics. The joystick on the “mbed application shield” serves as user input, which acts as either a “high”, “low” or “high-Z” state on a “GPIO” output pin. The detected input level is then displayed via another GPIO’s using the “RGB LED” on the “mbed application shield”. You can proof the implementation on the board with a single jumper wire or with a classmate by wiring two boards together.

> **Note**
>
> Only the **NULCEO-64** board and **MBED Application Shield** are used in this exercise, but the temperature logger board without an SD-Card inserted has no effect on this exercise and can remain connected.

Three state logic

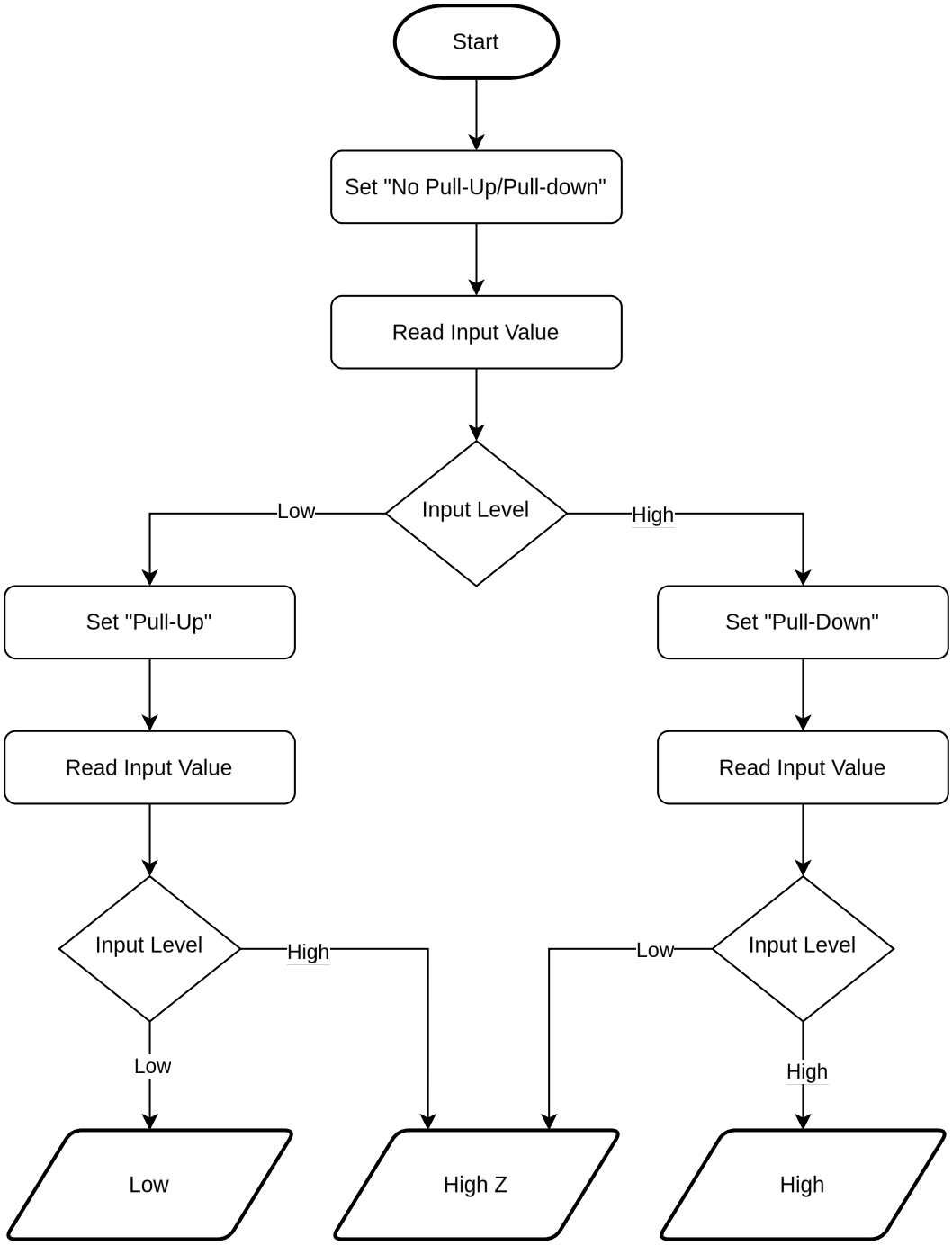
You may be wondering what the third state “High-Z” is. The low and the high state are pretty obvious, are they not? Take a look at the first LED implementation. To turn the LED on, you have to apply a sufficient forward voltage, and to turn it off, you have to apply a voltage lower than the minimum forward voltage. You have done this by either setting the pin to high level “VDD” or low level “VSS”. As both configurations are low impedance, the pin was electrically connected in both configurations. It was therefore tied to an electrical level. We call this behaviour “push-pull”. In other words, you can connect either the internal low voltage rail “VSS” or the high voltage rail “VDD” with low impedance to the output pin. There is another configuration called “Open-Drain”. This configuration allows the pin to be either connected to “VSS” or “disconnected” from the internal rails. A high impedance to “VSS” remains because there is no galvanic isolation. We call this “High-Z” or high impedance. See section **7.3.10 Output Configuration** of reference manual RM0390.

Now that you know how to handle three states on an output, we will discuss the input side next. The input driver consists of a “TTL” Schmitt trigger. As you know, a Schmitt trigger is similar to a comparator with a hysteresis instead of a single threshold. You can look up the hysteresis values in the **6.3.17 I/O Port Characteristics** section of the datasheet STM32F446re. You’re more likely to find the electrical characteristics in the datasheet STM32F446re than in the reference manual RM0390. The translation from the applied voltage at the input pin to a binary value in the input data register is given by the Schmitt trigger. However, we are still left with a binary result. How can you distinguish between three states at the input? In a nutshell, we can’t. At least not with the elements we have discussed so far.

So let us have a look at the pull-up/pull-down resistors of a GPIO port. Using either a pull-up or pull-down configuration, you can apply a bias current to either “VDD” or “VSS” via R_{PX}. The values of R_{PU}

and R_{PD} are described in the same section as the hysteresis values. The input impedance of a GPIO pin configured as an input is in the range of a few megaohms without using a pull-up/pull-down resistor. However, the input impedance is not defined by a resistor, as the input is a Schmitt trigger which consists of transistors. The input impedance is defined by a leakage current at a given supply voltage, described in the same section as the characteristics above. However, when using a pull-up/pull-down resistor, the input impedance is mainly defined by this resistor, $Z_{\text{Input Driver}} \gg Z_{\text{RPx}}$. If we put everything from this section together and connect an output pin to an input pin, we can discuss a three-state measurement. Remember that the “push-pull” configuration is a low impedance output regardless of the rail (“VDD” or “VSS”) connected. The open-drain configuration represents a high impedance connection to ground on a logical one in the output data register. On a logical zero, the impedance is the same as in a “pull” configuration. If you connect the “No Pull-Up/Pull-Down” input pin to a “High-Z” output pin, the voltage seen at the input can be any value between “VDD” and “VSS”. However, using either “Pull-Up” or “Pull-Down” on the input pin will pull the voltage to the appropriate rail. In summary, if the drive pin has a low output impedance, using a pull-up/pull-down resistor will not affect the result of your measurement. The result of your measurement will follow your pull-up/pull-down configuration if the drive pin has a high output impedance.

Three-state sensing algorithm



Flow chart of a possible three-state sensing algorithm.

Hint

Wait a moment until the level at the input of the pull-up/down resistor is in a stable state. Otherwise, the state detection will fail.

Tasks

Task description

The aim is to implement an algorithm to detect the current state of an input. This state is displayed using the RGB LED with different colours. The current state is controlled with the joystick, more precisely with the **left** and **right** inputs. To make it easier to share with your colleagues, we will use the same input pin and control pin:

- Sense Pin “PC12”
- Source Pin “PC10”

Preparation

On the **MBED Application Shield** you will find the interface required for this exercise (joystick and RGB LED). Use the NULCEO-64 MB1136 schematic and mbed application shield schematic to determine the connection to the microcontroller GPIOs.

Exercise

1. Find all used GPIO pins for the hardware and their location on the Morpho connectors.

Tip

You can take notes or draw on the schematics. All outputs from the Nucleo board are wired to the Morpho connectors. You can also look at the PCB Description to find the pins you need.

Solution

Lab-1.01 Used Pins

Hardware	MCU Pin	Morpho Pin
Switch left	PC1	CN7 Pin 36
Switch right	PC0	CN7 Pin 38
Sense pin	PC12	CN7 Pin 3
Source pin	PC10	CN7 Pin 1
LED red	PB4	CN10 Pin 27

Hardware	MCU Pin	Morpho Pin
LED green	PC7	CN10 Pin 19
LED blue	PA9	CN10 Pin 21

Direct manipulation

Before starting the implementation, check some basic configurations by manipulating the registers in a debug session, as you did in EX2 Direct Manipulation.

The first thing to do is start with the hardware on the mbed Application Shield.

Exercise

2. Configure all channels from the **RGB-LED** as output and toggle them.
3. Configure **SW_LEFT** and **SW_RIGHT** as inputs and observe the input values when pressing the joystick in that direction.

Hint

Remember to switch on the clock for each peripheral used.

The next step is to check the behaviour of different configurations of an output and an input with direct manipulations. The Sense and Source pins are used for this.

Exercise

4. Connect the pins together with a jumper wire. If you do not have a jumper wire, use a jumper from the back of your “Nucleo-64” board or the “nucleo-analog-frontend”.

With the connected pins we check the different types of a general purpose output of the Source pin with the Sense pin. Our MCU provides two different types of general purpose output:

Push-Pull

This type can be compared to a switch that connects the output to either **VDD** or **GND**.

Exercise

5. Configure the source pin as a push-pull output
6. Configure the sense pin as No Pull-Up/Pull-Down
7. Toggle the source pin to “High” and “Low”, what is the corresponding value of the sense pin?
8. Do the “Pull-Up/Pull-Down” resistors have any effect?

Hint

Remember to Pause `F6` and Continue `F5` the debugger to reload the input data register.

Open-Drain

The behaviour of this type can be compared to a button that connects to **GND** when the output is “pressed”, otherwise it is an open wire.

Exercise

9. Configure the source pin as an open-drain” output
10. Configure the sense pin as No Pull-Up/Pull-Down
11. Set the output of the source pin to “High-Z” (1)
12. Configure the sense pin to “Pull-Up”, what is its value?
13. Set the sense pin to “Pull-Down”, what is its value now?
14. Repeat the procedure with the output set to “Pull” (0). What do you expect?

Implementation

After reading and understanding the concepts described earlier, you have all the basics you need. You can start with this exercise.

The transmitter

The transmitter role is the one you should implement first. It requires less logic and is easier to implement than the receiver role. Configure an output pin to be either “VDD”, “VSS” or “High-Z” depending on the position of the joystick.

- Joystick in pos. right → “VDD”

- Joystick in pos. left → “VSS”
- Joystick none → “High-Z”

Exercise

15. Implement the logic to read the position of the joystick and set the transmitter output pin accordingly.

The receiver

The receiver reads the sense pin level and impedance. This pin is driven by the pin in the transmitter role. Refer to the Three-state sensing algorithm for more information on how to determine the logic level and its impedance. You will need to make at least two measurements to get all the information you need to make a decision. Depending on the current state, the RGB LED must show a different colour, according to the “state to colour mapping” given below.

- “VDD” → red
- “VSS” → blue
- “High-Z” → yellow

Exercise

16. Recognize the current state of the Sense pin and display it with the corresponding color on the RGB LED.