POST 1: What are device Drivers

The four important nodes required to start device driver programming are:

1. Memory addressing of the interface to be configured.
2. Register definitions of the interface.
3. Configuration definitions of the interface.
4. API’s for performing specific function of the interface.

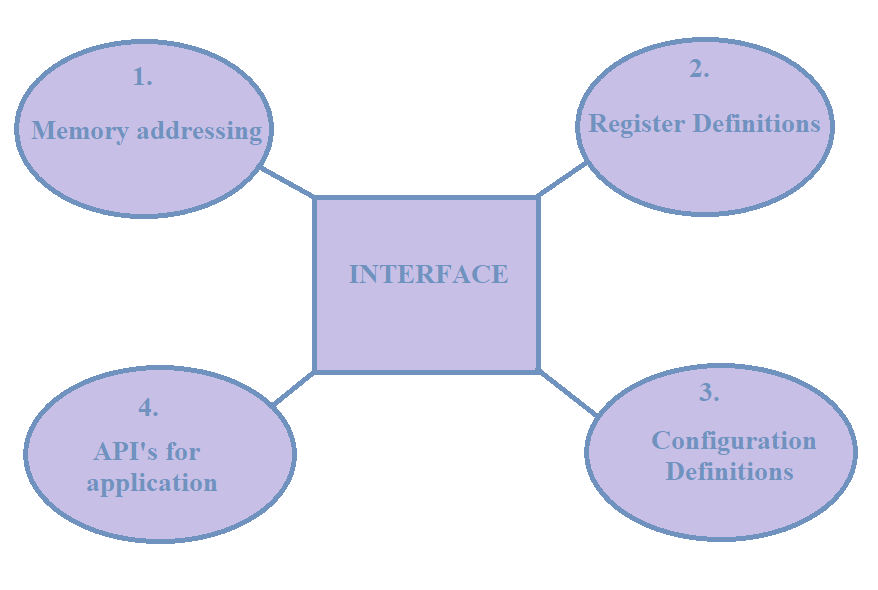


Fig.

1. Memory addressing:

2. Register Definitions:

3. Configuration definitions:

4. API’s for Application:

Let us start with the most basic interface to begin with device driver programming, the GPIO (General purpose input/output).

POST 2: What are GPIOs

In this post, you can also explain why GPIOs are needed and how they are useful. You can take examples of 5/6 LEDs and some switches to explain and then go on to explain how GPIOs work

**GPIO:**

Before diving in, into the coding let us check out some basics of GPIO. GPIO, General Purpose Input Output as the name says, functions as a bidirectional pin i.e. either serves as an input or output pin. Many a times, the GPIO pins are can also serve as an alternate functionality pin (They are multiplexed). The alternate functionality can include for the pin being used as either a USART transmit pin or a SPI MOSI pin etc. This can be selected from GPIOx\_AFR register. The popular examples include Blinking LED’s and use of PUSH buttons. It is also used to issue interrupts, reading digital signals, waking up the processor etc.

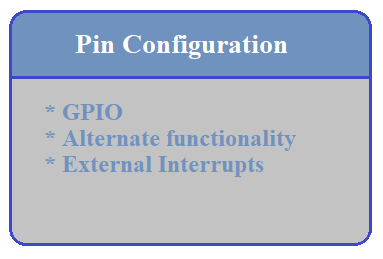


Fig 1: Pin configuration

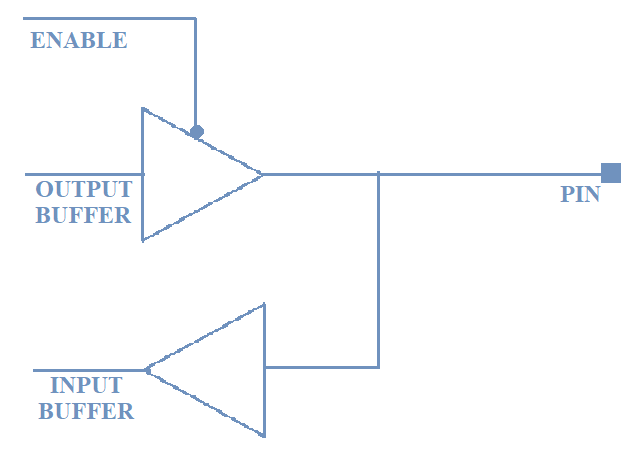
Generally a set of GPIO pins form a GPIO port represented as GPIOx, for eg: GPIOA. In STM32F4 board, 16 of these I/O pins form a GPIO port, hence it can be said that a GPIO port in STM32F4xx is 16-bit wide and each pin refers to a particular bit in the GPIO port registers. In the STM32F4xx board there are 11 such ports from GPIOA to GPIOK with each having its own register set. These ports are configured for specific modes which are discussed later.

**GPIO features**:

* Output states: push-pull or open drain + pull-up/down.
* Output data from output data register (GPIOx\_ODR) or peripheral (alternate function output).
* Speed selection for each I/O.
* Input states: floating, pull-up/down, analog.
* Input data to input data register (GPIOx\_IDR) or peripheral (alternate function input).
* Bit set and reset register (GPIOx\_ BSRR) for bitwise write access to GPIOx\_ODR
* Locking mechanism (GPIOx\_LCKR) provided to freeze the port A or B I/O port configuration.
* Analog function
* Alternate function selection registers(at most 16 AFs possible per I/O)
* Fast toggle capable of changing every two clock cycles
* Highly flexible pin multiplexing allows the use of I/O pins as GPIOs or as one of several peripheral functions

**How does a GPIO work internally?**

Basically a GPIO pin consists of an input buffer, an output buffer and an ENABLE pin. The value provided at the ENABLE pin decides whether the GPIO would work either as an input or an output.



The internal circuitry of the buffer is a simple CMOS logic circuit. It has a PMOS transistor connected to the +Vcc and a NMOS transistor connected to the ground as shown in fig. x. When the ENABLE pin set to 0, the output buffer is enabled and when it is set to 1, the input buffer is enabled.

ENABLE.[0] = 0 🡪 Pin functions as **OUTPUT** pin.

ENABLE.[0] = 1 🡪 Pin functions as **INPUT** pin.

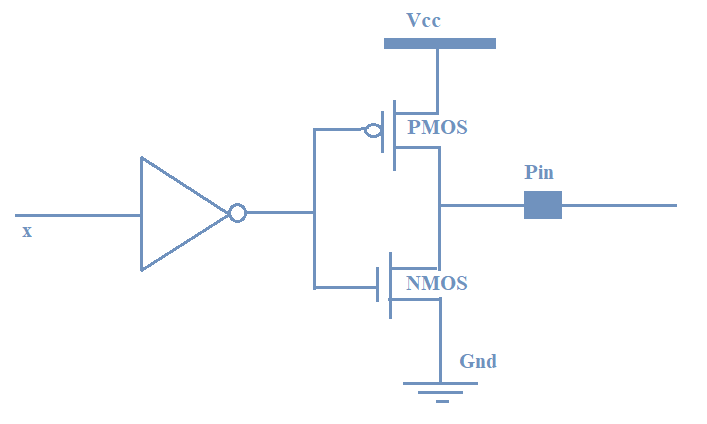


Fig. 3

Working:

For **OUTPUT** function: The ENABLE pin is set to 0, after passing the inverter logic it is set to 1, due to which the PMOS transistor is disabled and the NMOS transistor is enabled pulling the pin to GND (state LOW).

For **INPUT** function: The ENABLE pin is set to 1, after passing the inverter logic it is set to 0, due to which the NMOS transistor is disabled and the PMOS transistor is enabled pulling the pin to +Vcc (state HIGH).

Whenever the board is powered ON the GPIO pins, are by default in INPUT state i.e. they are in a HIGH impedance state. This is also referred to as Floating state.

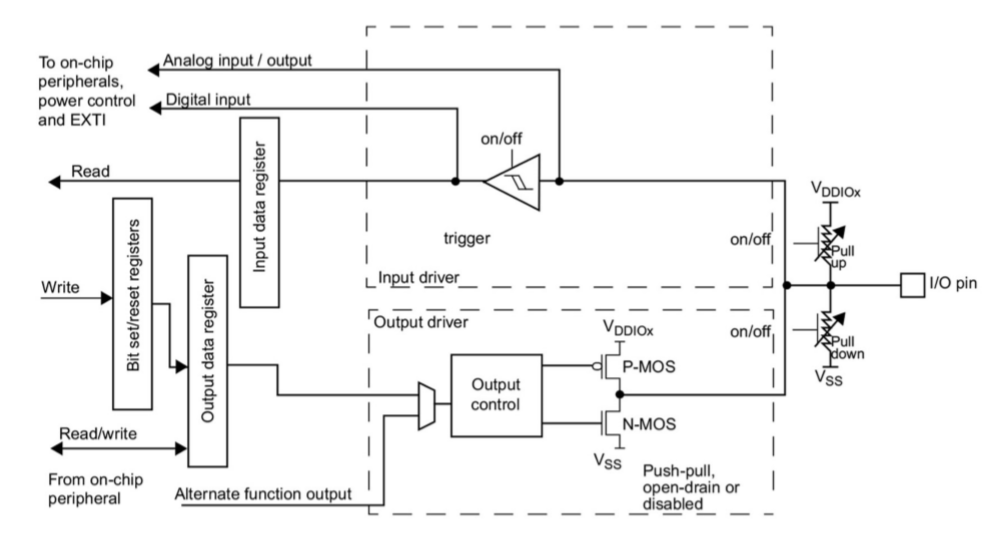


Fig. x

**GPIO modes:**

The GPIO pins can be individually configured to several modes:

1. Input Mode:
2. Output Mode
3. Analog Mode
4. Alternate function Mode

Input Mode: The input mode can be of three configurations:

* Pull-Up: If the input is configured for internal pull-up, then the state will be HIGH unless an external ground is connected.
* Pull-down: If the input is configured for pull-down, then the state will be LOW unless an external HIGH is connected.
* Floating point: This state is called as HIGH impedance state or indeterminate state wherein the pin toggles between HIGH and LOW induced by the external noise.

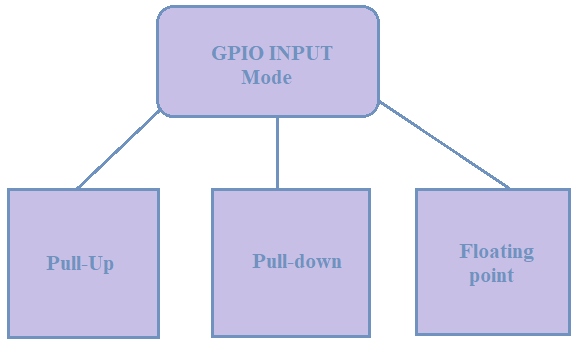


Fig.

Whenever a pin is configured as input the following characteristics are exhibited:

* The output buffer is disabled.
* The Schmitt trigger input is activated.
* The pull-up or pull-down resistors are activated depending on the value in the GPIOx\_PUPDR register.
* The data present on the I/O pin is sampled into the input data register at each AHB clock cycle.
* The I/O state is obtained by reading the GPIOx\_IDR input data register.

Output Mode: The output mode configuration can be of two types:

* Pull-up/Pull-down:

GPIO registers:

POST 3: Writing the DRIVERS for GPIO, the link to your device drivers(hosted on github) for download, and the API explanations with examples.