# LABORATORIO: INTRODUZIONE AI MICROCONTROLLORI STM32 NUCLEO

## GPIO

antonino.raucea@dieei.unict.it

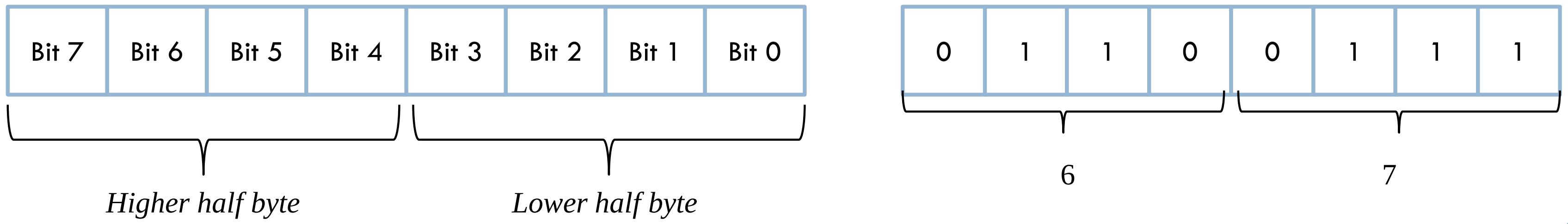
Ing. Antonino Raucea

# Configuration Registers

- In order to use a peripheral, its configuration register must be set
- Registers are memory location (usually 1, 2 or 4 bytes long) where each single bit has a specific meaning
- Each peripheral has its own configuration registers.
- Each register has a reserved name. They are listed and detailed in datasheets

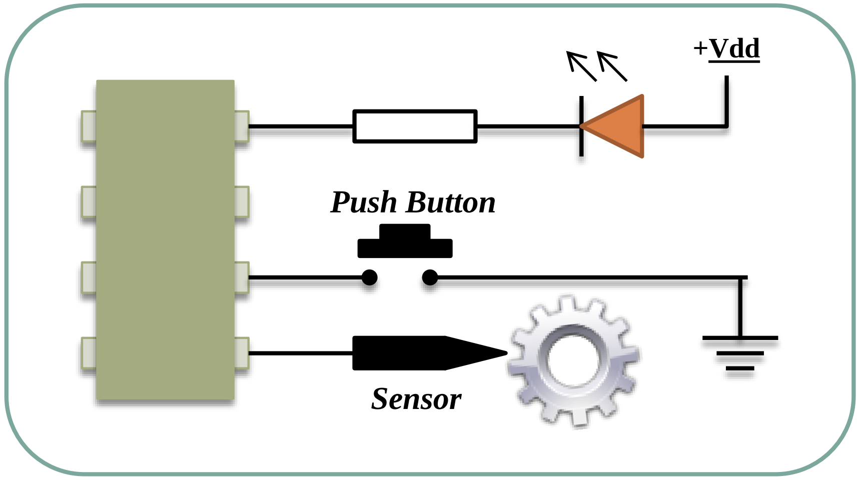
Hexadecimal numeral system is usually used

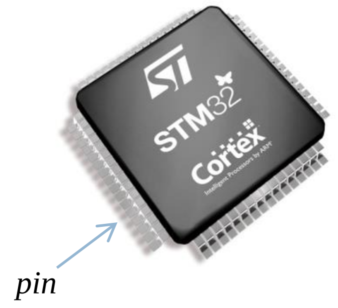
- Example (1 byte register):
  - CR1 = 01100111b (binary)= 103 (decimal) = 0x67 (hexadecimal)

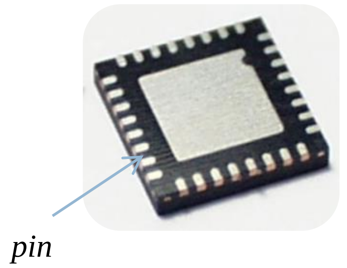


# Ports

- General Purpose I/O Ports are standard peripherals for communication from/to outside .
- They can be configure as input or output
- Several GPIO pins are divided into PORTS (usually 8 or 16 pins): PortA, PortB, etc.
- Example:
  - Port A pin 0, Port A pin1, … Port A pin 15 (some pin may be missing)

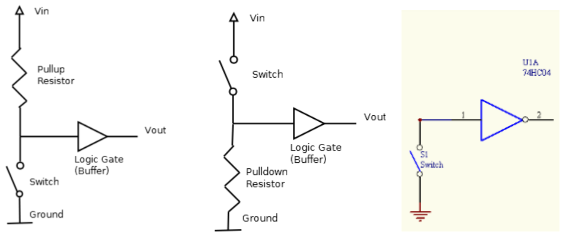






# GPIO Functional Description

- Each port bit of the general-purpose I/O (GPIO) ports can be individually configured by software in several modes:
  - Input floating
  - Input pull-up
  - Input-pull-down
  - Analog
  - Output open-drain with pull-up or pull-down capability
  - Output push-pull with pull-up or pull-down capability
  - Alternate function push-pull with pull-up or pull-down capability
  - Alternate function open-drain with pull-up or pull-down capability



- By means of configuration registers atomic read/modify/accesses to any of the GPIO registers is allowed.

# GPIO Registers

- **I/O port control registers**
  - GPIOx_MODER, I/O mode (input, output, AF, analog)
  - GPIOx_OTYPER, output type (pushpull or open-drain)
  - GPIOx_OSPEEDR, speed
  - GPIOx_PUPDR,  the pullup/pull-down whatever the I/O direction
- **I/O port data registers**
  - GPIOx_IDR

    The data input through the I/O are stored into the input data register, a read-only register
  - GPIOx_ODR

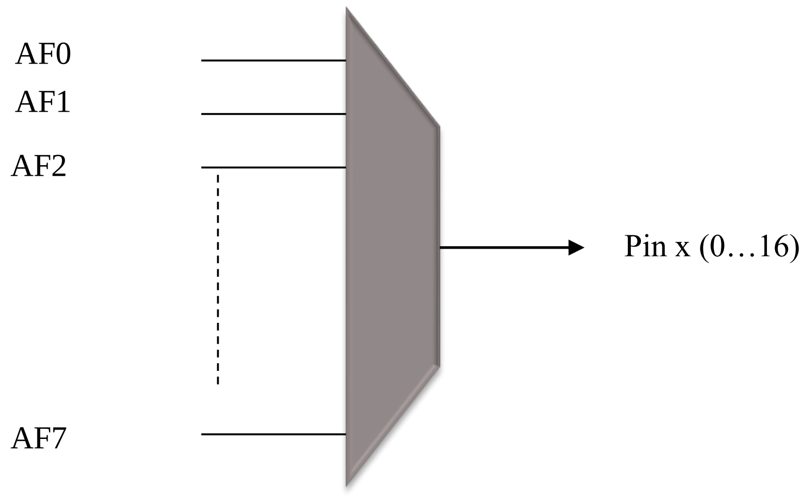
    stores the data to be output, it is read/write accessible
- **I/O data bitwise handling**
  - GPIOx_BSRR

    To each bit in GPIOx_ODR, correspond two control bits in GPIOx_BSRR: BS(i) and BR(i).

    When written to 1, bit BS(i) **sets** the corresponding ODR(i) bit. When written to 1, bit BR(i) **resets** the ODR(i) corresponding bit.

# Alternate Functions features

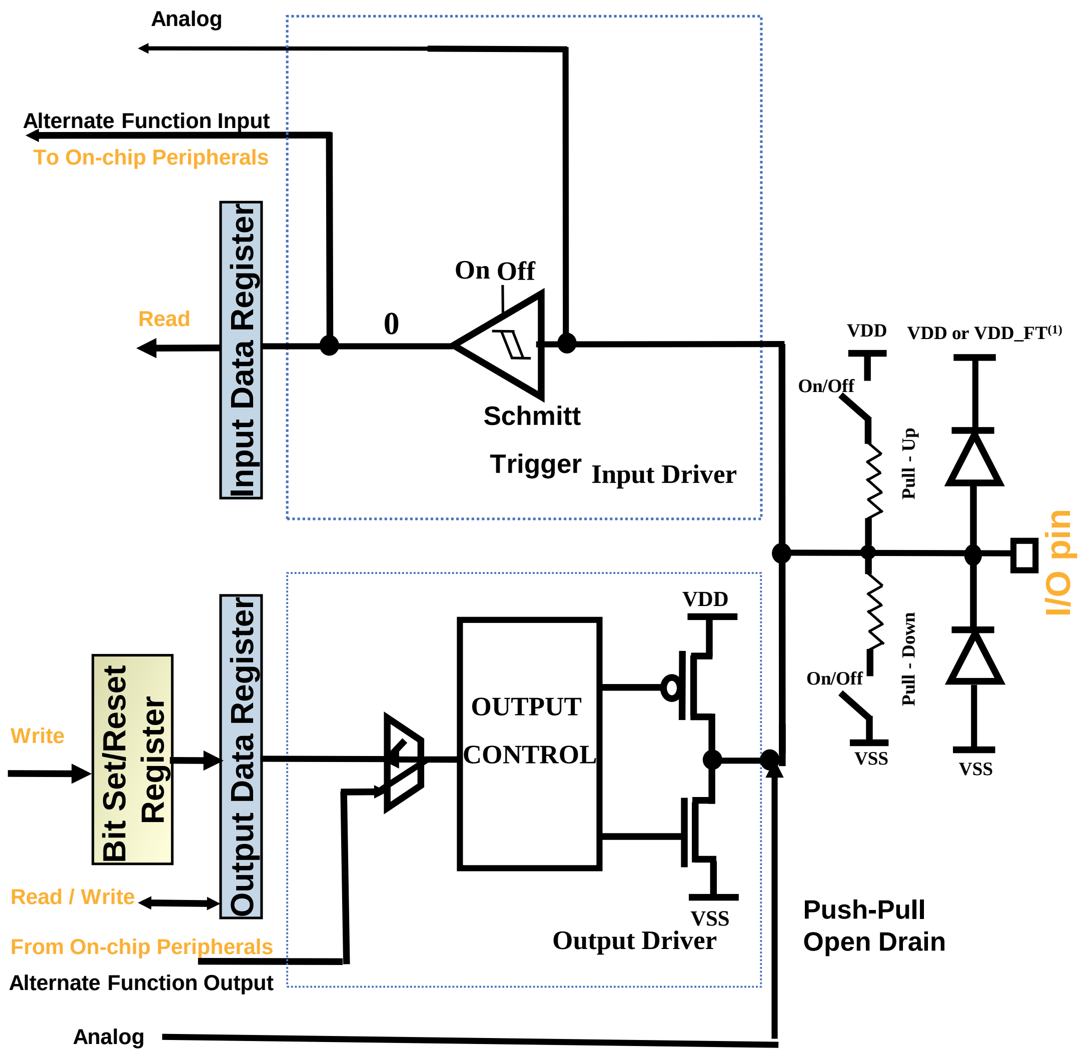
- Most of the peripherals shares the same pin (like USARTx_TX, TIMx_CH2, I2Cx_SCL, SPIx_MISO, EVENTOUT…)
- Alternate functions multiplexers prevent to have several peripheral's function pin to be connected to a specific I/O at a time.



# GPIO Configuration Modes

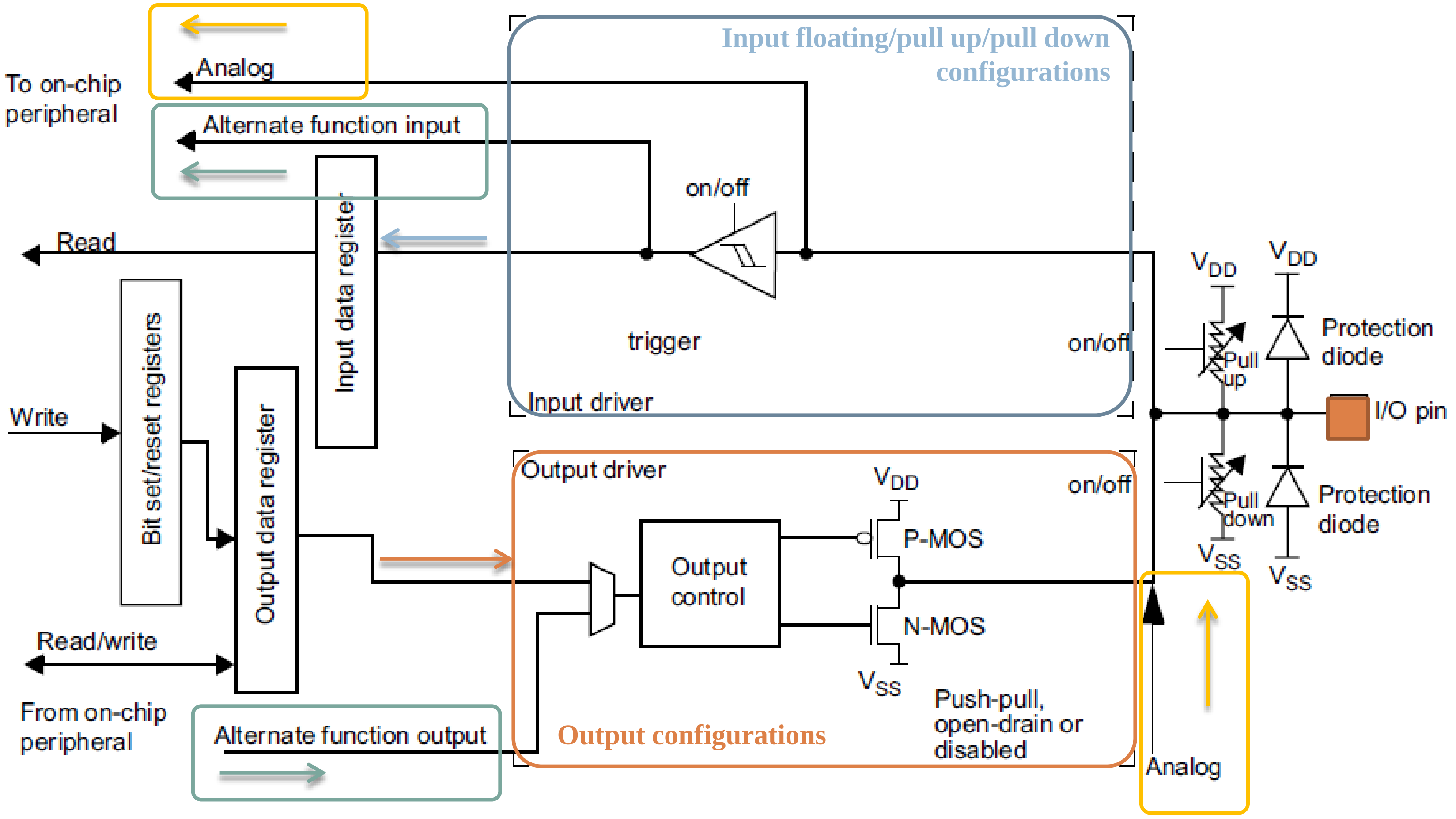
| MODER(i) [1:0] | OTYPER(i) | OSPEEDR(i) [B:A] | | PUPDR(i) [1:0] | | I/O configuration | |
|---|---|---|---|---|---|---|---|
| 01 | 0 | SPEED [B:A] | | 0 | 0 | GP output | PP |
| | 0 | | | 0 | 1 | GP output | PP + PU |
| | 0 | | | 1 | 0 | GP output | PP + PD |
| | 0 | | | 1 | 1 | Reserved | |
| | 1 | | | 0 | 0 | GP output | OD |
| | 1 | | | 0 | 1 | GP output | OD + PU |
| | 1 | | | 1 | 0 | GP output | OD + PD |
| | 1 | | | 1 | 1 | Reserved (GP output OD) | |
| 10 | 0 | SPEED [B:A] | | 0 | 0 | AF | PP |
| | 0 | | | 0 | 1 | AF | PP + PU |
| | 0 | | | 1 | 0 | AF | PP + PD |
| | 0 | | | 1 | 1 | Reserved | |
| | 1 | | | 0 | 0 | AF | OD |
| | 1 | | | 0 | 1 | AF | OD + PU |
| | 1 | | | 1 | 0 | AF | OD + PD |
| | 1 | | | 1 | 1 | Reserved | |
| 00 | x | x | x | 0 | 0 | Input | Floating |
| | x | x | x | 0 | 1 | Input | PU |
| | x | x | x | 1 | 0 | Input | PD |
| | x | x | x | 1 | 1 | Reserved (input floating) | |
| 11 | x | x | x | 0 | 0 | Input/output | Analog |
| | x | x | x | 0 | 1 | Reserved | |
| | x | x | x | 1 | 0 | | |
| | x | x | x | 1 | 1 | | |

* In output mode, the I/O speed is configurable through OSPEEDR register: 2MHz, 10MHz or 50MHz



(1) VDD_FT is a potential specific to five-volt tolerant I/Os and different from VDD.

# Basic Structure of a Standard I/O Port Bit



# STM32 Configuration Example

STM32 libraries allows to configure easily peripherals.

- **Configure GPIO PC11 & PC12 as Output Push-Pull**

```
GPIO_InitTypeDef GPIO_InitStructure;        /* Pointer to a GPIO_InitTypeDef structure that
                                               contains the configuration information for the
                                               specified GPIO peripheral */
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_11 | GPIO_Pin_12;        /* Specifies the GPIO pins to be
                                                                   configured → Two GPIO pins selected*/
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;       /* Specifies the operating mode for the selected
                                                          pins →  Output push-pull */
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;      /* Specifies the speed for the selected pins */
GPIO_Init(GPIOC, &GPIO_InitStructure);                 /* Send command, configure GPIOs*/
```

- where ***GPIO_InitTypeDef*** is defined as:

```
typedef struct {
  uint16_t GPIO_Pin;                        /* Specifies the GPIO pins to be configured. */
  GPIOSpeed_TypeDef GPIO_Speed;             /* Specifies the speed for the selected pins.*/
  GPIOMode_TypeDef GPIO_Mode;               /* Specifies the operating mode for the selected pins.*/
} GPIO_InitTypeDef;
```