Using the Digital I/O interface of STMicroelectronics STM32 Microcontrollers

Corrado Santoro

ARSLAB - Autonomous and Robotic Systems Laboratory
Dipartimento di Matematica e Informatica - Università di Catania, Italy
santoro@dmi.unict.it

L.A.P. 1 Course
What is a “digital I/O interface”? 

It is an interface in which each electrical pin may have two states:

- Logical 0 (it means 0V);
- Logical 1 (it means 5V or 3.3V on the basis of the VDD);

Each line can be programmed as:

- an **output** (it “generates” current and can be used, for example, to light a LED)
- an **input** (it “receives” current and can be used, for example, to read a pushbutton)
The General Purpose I/O (GPIO) Interface of STM32 MCUs of the STM32 family have several digital ports, called GPIOA, GPIOB, GPIOC, ..., Each port has 16 bits and thus 16 electrical pins. Pins are referred as Pxy, where x is the port name (A, B, ..., E) and y is the bit (0, 1, ..., 15). As an example, the pin PC3 is the bit 3 of the port C. Each PIN has also an alternate function, related to a peripheral e.g. Timer, UART, SPI, etc. According to the MCU package, not all bits are mapped to electrical pins. This is a choice “by-design”.
The General Purpose I/O (GPIO) Interface of STM32

Corrado Santoro
Digital I/O in STM32 Family
To use a specific GPIO line (pin), the following operations are needed:

- **Set-up**
  1. Initialize the whole GPIO port (this operation basically enables the clock line to the GPIO port)
  2. Set the direction (input or output) of the pin you intend to use

- **Operate**
  - Read the GPIO pin, if it is programmed as “input”, or
  - Write the GPIO pin, if it is programmed as “output”

These operations are made really simple using the `stm32_unict_lib`

*Corrado Santoro*  
Digital I/O in STM32 Family
Using the GPIO Interface with stm32_unict_lib

- **Example:** Setting **PA5** as output and using it

- **Set-up**
  1. **Initialize the whole GPIO port** (this operation basically enables the clock line to the GPIO port)
     
     ```c
     GPIO_init(GPIOA);
     ```
  2. **Set the direction of the pin you intend to use**
     
     ```c
     GPIO_config_output(GPIOA, 5);
     ```

- **Operate**
  - **Write “0” to PA5:**
    
    ```c
    GPIO_write(GPIOA, 5, 0);
    ```
  - **Write “1” to PA5:**
    
    ```c
    GPIO_write(GPIOA, 5, 1);
    ```
The Nucleo64 Addon Board

Corrado Santoro

Digital I/O in STM32 Family
```c
#include "stm32_unict_lib.h"

int main()
{
    // initialize port B
    GPIO_init(GPIOB);

    // configure pin PA5 as output
    GPIO_config_output(GPIOB, 8);

    // infinite loop
    for (;;)
    {
        GPIO_write(GPIOB, 8, 1); // set PB8 to 1
        delay_ms(500); // wait 0.5 secs
        GPIO_write(GPIOB, 8, 0); // set PB8 to 0
        delay_ms(500); // wait 0.5 secs
    }
}
```
Example: setting **PC3** as input and using it

**Set-up**

1. Initialize the whole GPIO port (this operation basically enables the clock line to the GPIO port)
   ```c
   GPIO_init(GPIOC);
   ```

2. Set the direction of the pin you intend to use
   ```c
   GPIO_config_input(GPIOC, 3);
   ```

**Operate**

- Read PC3 pin:
  ```c
  int pinval = GPIO_read(GPIOC, 3);
  ```

- “pinval” can be “0” or “1”
#include "stm32_unict_lib.h"

int main()
{
    // pushbutton on PA10; LED on PB8

    // initialize ports
    GPIO_init(GPIOA);
    GPIO_init(GPIOB);

    // configure pin PA10 as input
    GPIO_config_input(GPIOA, 10);

    // configure pin PB8 as output
    GPIO_config_output(GPIOB, 8);

    // infinite loop
    for (;;)
    {
        int pinval = GPIO_read(GPIOA, 10);
        GPIO_write(GPIOB, 8, !pinval);
    }
}
What are the GPIOA, GPIOB, ... variables?

What are the prototypes of the GPIO functions?

GPIOA, GPIOB, ... are **global variables** defined in CMSIS libraries as:

```c
GPIO_TypeDef * GPIOA;
GPIO_TypeDef * GPIOB;
...;
```

**GPIO_TypeDef** is a structure whose fields are the special-function-registers (SFRs) of a GPIO port.

Each GPIOA, GPIOB, ... variable is a **pointer** to a **GPIO_TypeDef** and represents the **address** of the memory holding the SFRs of that port.
The GPIO function prototypes

- Initialize a GPIO port:
  ```c
  void GPIO_init(GPIO_TypeDef * port);
  ```

- Configure a GPIO pin as input:
  ```c
  void GPIO_configure_input(GPIO_TypeDef * port,
                             int pin_num);
  ```

- Configure a GPIO pin as output:
  ```c
  void GPIO_configure_output(GPIO_TypeDef * port,
                              int pin_num);
  ```

- Write to an output pin:
  ```c
  void GPIO_write(GPIO_TypeDef * port, int pin_num,
                  int pin_val);
  ```

- Read from an input pin:
  ```c
  int GPIO_read(GPIO_TypeDef * port, int pin_num);
  ```

- Change the state of an output pin:
  ```c
  void GPIO_toggle(GPIO_TypeDef * port, int pin_num);
  ```
Using the Digital I/O interface of STMicroelectronics STM32 Microcontrollers

Corrado Santoro

ARSLAB - Autonomous and Robotic Systems Laboratory
Dipartimento di Matematica e Informatica - Università di Catania, Italy
santoro@dmi.unict.it

L.A.P. 1 Course