O Supremo STM32|Periférico GPIO

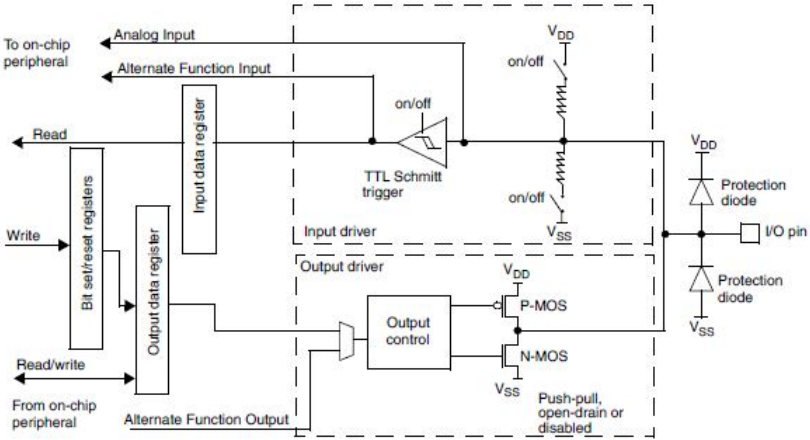
Luiz Fernando Pinto de Oliveira

7 de julho de 2018

GPIO - *General Purpouse Input/Output*

1. Características:
   - Dependendo da alimentação 2 a 3,6V (tolerantes a 5 V);
   - Correntes de saída e entrada até 25 mA (total de 150 mA nas linhas de alimentação);
   - *Pull-up* e *pull-downs* internos com aproximadamente 40 kΩ;
   - Velocidade de comutação até 50 MHz (configurável por *software*);
   - Todos podem ser configurados como interrupções externas;
   - Cada GPIO pode ser alterado bit a bit usando a instrução apenas BSR e BRR. Desta forma garante-se que o *set* ou o *clear* de um bit é realizado numa instrução apenas.

GPIO - Estrutura Básica



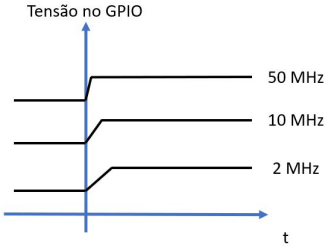
Modos de Configuração

- *General purpose output*
  - *Push-pull*
  - *Open-drain*
- *Alternate Function output*
  - *Push-pull*
  - *Open-drain*
- *Input*
  - *Analog*
  - *Floating*
  - *Pull-down*
  - *Pull-up*

Definição

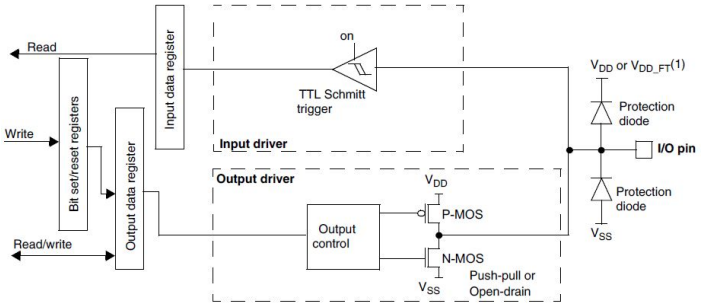
É o tempo em que o pino do microcontrolador demora para mudar seu estado lógico. Para altos valores pode ocorrer o aparecimento de *overshoot*, ocasionando interferências eletromagnéticas, além de aumentar o consumo no pino.

Modo	*Output speed*
00	Reservado
01	Máx. 10 MHz
10	Máx. 2 MHz
11	Máx. 50 MHz



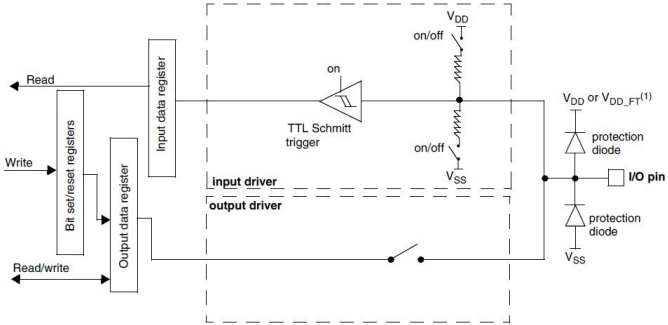
GPIO - Configurado como *Output*

- No modo *Open Drain* ”0” ativa o N-MOS e ”1” fica em alta impedância (O P-MOS nunca é ativado);
- No modo *Push-Pull* ”0” ativa o N-MOS e ”1” ativa o P-MOS;
- O *Schmitt Trigger* é ativado;
- As resistências de *pull-up* e *pull-down* são desativadas;
- A leitura do *Input Data Register* obtém o estado do I/O no modo *Open Drain*;
- A leitura do *Output Data Register* obtém o último valor escrito no modo *Push-Pull*.



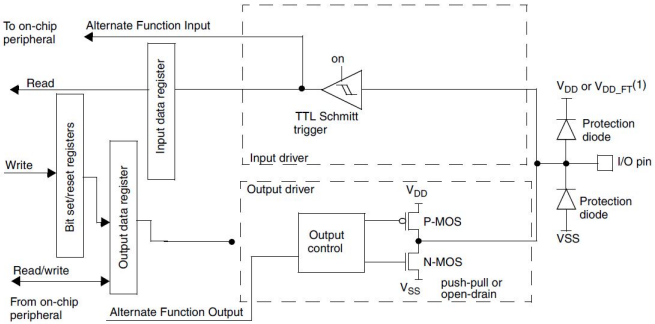
GPIO - Configurado como *Input*

- O *buffer* de *output* é desligado;
- O *Schmitt Trigger* é ativado;
- As resistência de *pull-up* e *pull-down* são ativadas ou não mediante a configuração escolhida: *pull-up*, *pull-down* ou *floating*;
- A leitura do *Input Data Register* obtém o estado do I/O.



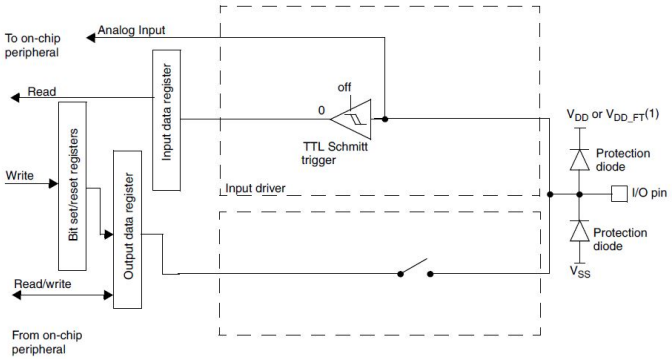
GPIO - Configurado como *Alternation Function*

- O *buffer* de *output* é controlado pelo sinal Alternate Function *Output*, ficando em *Open Drain* ou *Push-pull* conforme a funcionalidade;
- O *Schmitt Trigger* é ativado;
- As resistências de *pull-up* e *pull-down* são desativadas;
- A leitura do *Input Data Register* obtém o estado do I/O no modo *Open Drain*;
- A leitura do *Output Data Register* obtém o último valor escrito no modo *Push-Pull*.



GPIO - Configurado como *Analog Input*

- O *buffer* de *output* é desligado;
- O *Schmitt Trigger* é desligado sendo forçado a um valor constante "0";
- As resistências de *pull-up* e *pull-down* são desativadas;
- A leitura do *Input Data Register* obtém o valor "0";
- A leitura do *Output Data Register* obtém o último valor escrito no modo *Push-Pull*.



Declaração da variável do tipo:
GPIO_InitTypeDef GPIO_InitStructure;
Estrutura de um GPIO:

```
typedef struct{
  u16 Pin;
  GPIO_TydeDef GPIO_Mode;
  GPIO_TydeDef GPIO_Speed;
}GPIO_InitTypeDef;
```

Exemplo de inicialização do port GPIOA:
GPIO_Init(GPIOC, GPIO_InitStructure);

Exemplo da configuração de um GPIO:

```
RCC_APB1PeriphClockCmd(RCC_APB1Periph_GPIOC, ENABLE); //Ver Cap. 2

GPIO_InitTypeDef GPIO_InitStructure;

GPIO_InitStructure.Pin = GPIO_Pin_12;
GPIO_InitStructure.Mode = GPIO_MODE_IN_FLOATING;
GPIO_InitStructure.Speed = GPIO_Speed_50MHz;
GPIO_Init(GPIOC, &GPIO_InitStructure);
```

- Por omissão, o perififério GPIO permanece configurado como:
  - GPIO_Pin_ALL;
  - GPIO_Mode_IN_FLOATING;
  - GPIO_Speed_2MHz.

GPIO - Funções para escrever no GPIO

- GPIO_SetBits()
  - GPIO_SetBits(GPIOx, GPIO_Pin_x);
- GPIO_ResetBits()
  - GPIO_ResetBits(GPIOx, GPIO_Pin_x);
- GPIO_WriteBit()
  - GPIO_WriteBit(GPIOx, GPIO_Pin_x, Bit_SET ou Bit_RESET);
- GPIO_Write()
  - GPIO_Write(GPIOx, 0xffff);

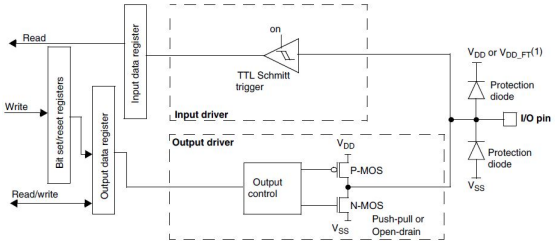
Leitura da entrada:

GPIO_ReadInputDataBit

```
uint8_t ReadValue;
ReadValue = GPIO_ReadInputDataBit(GPIOB, GPIO_Pin_7);
```

GPIO_ReadInputData

```
uint16_t ReadValue;
ReadValue = GPIO_ReadInputData(GPIOB);
```



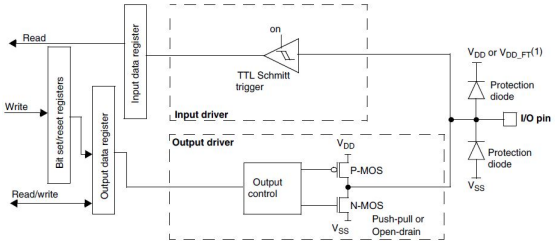
Leitura da saída:

GPIO_ReadOutputDataBit

```
uint8_t ReadValue;
ReadValue = GPIO_ReadOutputDataBit(GPIOB, GPIO_Pin_7);
```

GPIO_ReadOutputData

```
uint16_t ReadValue;
ReadValue = GPIO_ReadOutputData(GPIOB);
```



Configuração e leitura de um GPIO

```
RCC_APB1PeriphClockCmd(RCC_APB1Periph_GPIOC, ENABLE); //Ver Cap. 2

GPIO_InitTypeDef GPIO_InitStructure;

GPIO_InitStructure.Pin = GPIO_Pin_12;
GPIO_InitStructure.Mode = GPIO_MODE_IN_FLOATING;
GPIO_InitStructure.Speed = GPIO_Speed_50MHz;
GPIO_Init(GPIOC, &GPIO_InitStructure);

uint32_t ReadValue;
ReadValue = GPIO_ReadInputDataBit(GPIOC, GPIO_Pin_12);  //Le o pino
    12 do GPIOC
ReadValue = GPIO_ReadInputData(GPIOC);  //Le o PORT GPIOC inteiro
```

GPIO - Exemplo do uso do Periférico GPIO

```c
#include "stm32f4xx_hal.h"

void Config_GPIO(void);
void msDelay(uint32_t time);
uint32_t read;

int main(){
        Config_GPIO();
        while(1){
                read = GPIO_ReadInputDataBit(GPIOA, GPIO_Pin_0);
                if(read == 0){
                        GPIO_WriteBit(GPIOD,GPIO_Pin_12|GPIO_Pin_13|GPIO_Pin_14|GPIO_Pin_15, Bit_SET);
                }
                else{
                        GPIO_WriteBit(GPIOD,GPIO_Pin_12|GPIO_Pin_13|GPIO_Pin_14|GPIO_Pin_15, Bit_RESET);
                }
        }
} ...
```

GPIO - Exemplo do uso do Periférico GPIO

```
void Config_GPIO(void){
        RCC_APB1PeriphClockCmd(RCC_APB1Periph_GPIOA|RCC_APB1Periph_GPIOC, ENABLE);

        GPIO_InitTypeDef GPIO_InitStructure;

        GPIO_InitStructure.Pin = GPIO_PIN_12|GPIO_PIN_13|GPIO_PIN_14|GPIO_PIN_15;     //LEDs em PD12, PD13, PD14 e
            PD15
        GPIO_InitStructure.Mode = GPIO_MODE_OUTPUT_PP;
        GPIO_Init(GPIOD, &GPIO_InitStructure);

        GPIO_InitStructure.Pin = GPIO_PIN_0;     //Botao em PA0
        GPIO_InitStructure.Mode = GPIO_MODE_INPUT;
        GPIO_InitStructure.Pull = GPIO_PULLDOWN;
        GPIO_Init(GPIOA, &GPIO_InitStructure);
}

void msDelay(uint32_t time){
        for(uint32_t i = 0;i<time*4000;i++);
}
```

LSERF
Laboratório de Soluções em Eletrônica e RF

Seja 100 % Motivado!