ARC

GPIO

支持	www.armrunc.com http://armrunc.taobao.com

Change History

版本号	更改描述	作者	修改时间
1.0.0	第一版	armrunc	2012-03-06

目录

第3章 GPIO

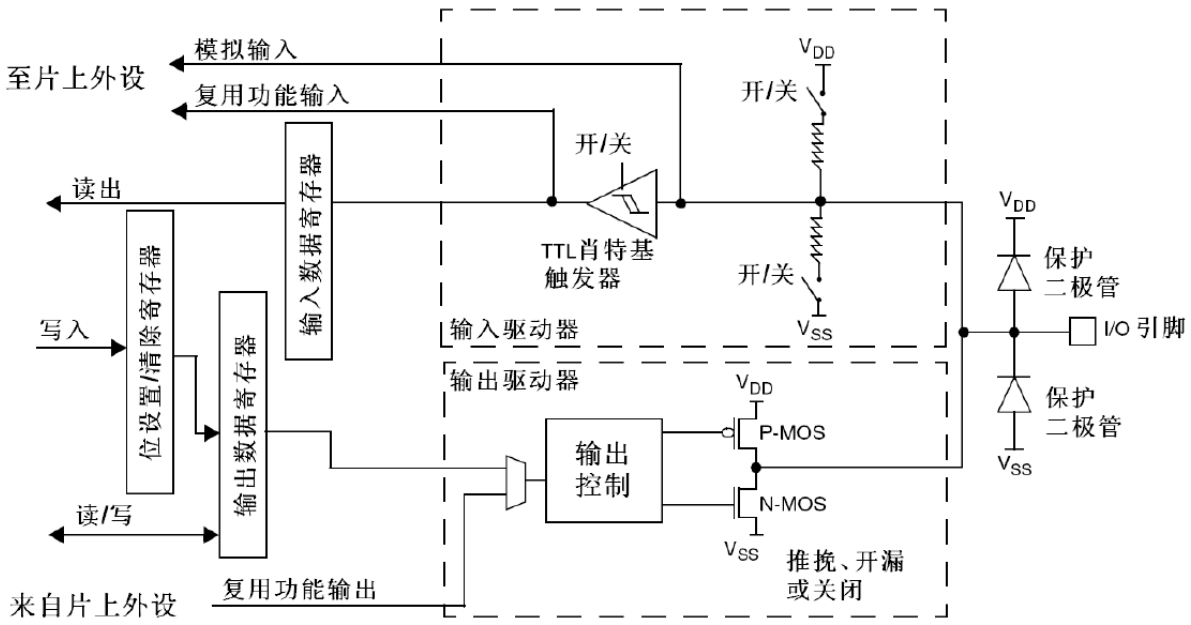
3.1 GPIO 简介

General Purpose Input/Output (通用输入/输出)简称为 GPIO，而 GPIO 作为 STM32 的一个最基本的外设，可以通过软件配置为以下 8 种模式：

1. 浮空输入
2. 上拉输入
3. 下拉输入
4. 模拟输入
5. 开漏输出
6. 推挽输出
7. 复用功能推挽输出
8. 复用功能开漏输出

STM32 GPIO 端口位的基本结构如下：

图 2-1 I/O 端口基本结构



STM32 端口位配置表如下：

表 2-1 端口位配置表

配置模式	CNF1	CNF0	MODE1	MODE0	PxODR

						Register
通用输出	推挽	0	0	01 10 11 见表 1-1		0 或 1
	开漏		1			0 或 1
复用功能输出	推挽	1	0			不使用
	开漏		1			不使用
输入	模拟输入	0	0	00		不使用
	浮空输入		1			不使用
	下拉输入	1	0			0
	上拉输入					1

表 2-2 输出模式位

MODE[1:0]	意义
00	保留
01	最大输出频率 10MHz
10	最大输出频率 2MHz
11	最大输出频率 50MHz

复位期间和刚复位后，复用功能未开启，I/O端口被配置成浮空输入模式(CNFx[1:0]=01b，MODEx[1:0]=00b)。 复位后，JTAG引脚被置于输入上拉或下拉模式：

- PA15：JTDI置于上拉模式
- PA14：JTCK置于下拉模式
- PA13：JTMS置于上拉模式
- PB4： JNTRST置于上拉模式

当作为输出配置时，写到输出数据寄存器上的值(GPIOx_ODR)输出到相应的I/O引脚。可以以推挽模式或开漏模式(当输出0时，只有N-MOS被打开)使用输出驱动器。 输入数据寄存器(GPIOx_IDR)在每个APB2时钟周期捕捉I/O引脚上的数据。 所有GPIO引脚有一个内部弱上拉和弱下拉，当配置为输入时，它们可以被激活也可以被断开。

3.2 GPIO 应用实例 ----- 驱动发光二极管

3.2.1 实例描述

本实例通过配置 PA1 和 PA2 为推挽输出，来分别驱动 LED0 和 LED1，注意，为了简单，本实验的延迟使用空指令循环。

将次程序烧录到 ARC 开发板上，你可以看到 LED0 和 LED1 不停的轮流闪烁。

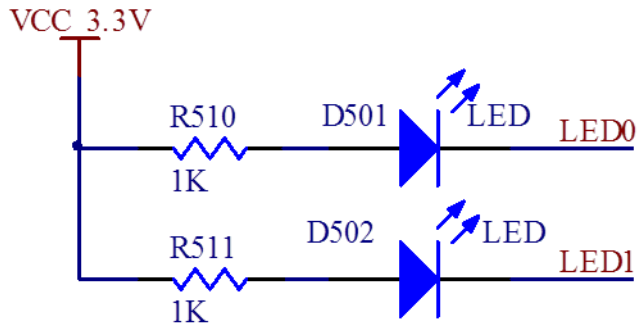
3.2.2 硬件设计

PA1 连接到 LED0, PA2 连接到 LED1，这两个 I/O 口都被配置成推挽输出，最大输出频率为 50MHz，注意，输出为低电平时，发光二极管点亮。

表 1-3 GPIO LED 连接

PA1	LED0
PA2	LED1

图 2-2 LED 硬件电路图



3.2.3 软件设计

本实例首先调用 ARC_LED_Init() 来初始化 PA1 (LED0) 和 PA2 (LED1)，该初始化函数首先使能 GPIO A 端口的时钟，然后配置 PA1 和 PA2 为最大时钟为 50Mhz 的推挽输出。

然后调用 ARC_LED_Set() 函数，设置 PA1 和 PA2 输出电平。

注意，由上面的电路图可得出，当输出高电平时，熄灭 LED，输出为低电平时，点亮 LED。

该实例的主要代码如下：

文件 LED_main.c

```
/**
  * @brief   Main program, flash LED0 and LED1.
  * @param   None
  * @retval None
  */
int main(void)
{
    uint32_t i;
    ARC_LED_Init();
    while (1)
    {
```

```
            ARC_LED_Set(0, 1);
            ARC_LED_Set(1, 0);
            for( i = 0; i < 5000000; i++);
            ARC_LED_Set(0, 0);
            ARC_LED_Set(1, 1);
            for( i = 0; i < 5000000; i++);
        }
}
```

文件 ARC_LED.c

```
/**
   * @brief    Initialize LED.
   * @param    None
   * @retval None
   */
void ARC_LED_Init()
{
      ARC_LED_RCC_Init();
      ARC_LED_GPIO_Init();
}

/**
   * @brief    set/reset LED0/LED1.
   * @param    LED: Indicate the LED number, either 0 or 1.
   * @param    value: the output of LED0/LED1, either 0 or 1.
   * @retval None
   */
void ARC_LED_Set(uint8_t LED, uint8_t value)
{
      if (LED == 0)
      {
            if (value == 0)
                  GPIO_ResetBits(GPIOA, GPIO_Pin_1);
            else if (value == 1)
                  GPIO_SetBits(GPIOA, GPIO_Pin_1);
      }
      else if (LED == 1)
      {
            if (value == 0)
                  GPIO_ResetBits(GPIOA, GPIO_Pin_2);
            else if (value == 1)
                  GPIO_SetBits(GPIOA, GPIO_Pin_2);
      }
}
```

文件 ARC_RCC.c

```
/**
  * @brief  Configures LED clocks.
  * @param  None
  * @retval None
  */
void ARC_LED_RCC_Init(void)
{
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA, ENABLE);
}
```

文件 ARC_GPIO.c

```
/**
  * @brief  Configures LED GPIO ports.
  * @param  None
  * @retval None
  */

/*
        --------------
        | LED0 | PA1 |
        --------------
        | LED1 | PA2 |
        --------------
*/
void ARC_LED_GPIO_Init()
{
    GPIO_InitTypeDef GPIO_InitStructure;
    /* Configure the LED0 pin */
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_1;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_Init(GPIOA, &GPIO_InitStructure);

    /* Configure the LED0 pin */
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_2;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_Init(GPIOA, &GPIO_InitStructure);
}
```