

## 第二周任务

**鼓励新生在大群里解答问题，有态度加分！**

### 一、任务要求

1. 基础任务必须完成，进阶任务可选择性完成。

2. 任务实现要求：

- 程序上的实现
- 拍摄任务实现的效果
- 针对项目所涉及到的内容完成技术报告

3. 任务提交要求：

- 请将关键代码贴到技术报告中，并将视频和代码的压缩包均上传到文档中(请一定要使用语雀进行编写)
- 请在截止时间前将自己的技术文档的语雀链接在智海思源平台上提交(提交前一定要先共享文档!!!)

**注：技术报告格式可自己安排，重点在于体现自己完成这个任务的过程，可以将自己所用到的知识点和遇到的问题等记录在其中。**

### 二、任务内容

#### (一) 任务：GPIO输入输出

1. 任务介绍

说到GPIO，它最基本的功能就是输入与输出，这为单片机与外界交互提供了桥梁。无论是什么高级外设，基本都离不开GPIO，所以GPIO才会有复杂的复用和重定向功能，以链接到众多其他外设上去。

## 2. 任务要求

设置两个GPIO，同一时间一个输出，一个输入，输出的GPIO将信号传入输入的GPIO，控制灯的亮灭

- 通过一个GPIO输出高电平信号，输入另一个GPIO，接收高电平信号的同时点亮LED。
- 通过一个GPIO输出低电平信号，输入另一个GPIO，接收低电平信号的同时熄灭LED。
- 周期性重复1和2的操作，使得LED周期性亮灭。

## (二) 任务：PWM控制舵机

### 1. 任务介绍

对于单片机来说，很难在软件上来控制引脚输出不同的固定电平，而PWM解决了这个问题。PWM是一种常用的控制外设的信号，通过PWM信号，可以使得只能输出高/低电平的单片机输出从低电平到高电平的所有电压值。

### 2. 任务要求

- 通过定时器配置PWM使得舵机能够稳定维持在一定角度。（备注：由于舵机的特性，输入一定占空比的PWM就会转动到一定角度）

### 3. 技术点介绍

- 定时器的比较输出（比较输出的工作原理，其中的ccr寄存器）
- PWM 是什么？输出不同固定电平的原理是什么？
- PWM 的实现

### (三) 任务：阅读参考手册

#### 1. 有些内容可能教程里没有涉及，希望同学们学会在网上查找答案

#### 2. 任务介绍

我们现阶段学习的资料大多是来自 718 自编、野火教程以及如博客园 cnblog 等网络论坛社区。

你有没有想过，他们是从谁那里学习如此复杂的单片机知识，并加以运用的呢？

答案是 ST 官方的参考手册，数据手册，CM3 内核手册等众多手册。这些手册对外设功能，原理，使用，引脚分配等众多方面进行了简明扼要的阐述，懂得如何使用参考手册是成为一名优秀的电子工程师所必不可少的技能。

- 准确来说，这只是以任务的形式来向大家介绍参考手册和数据手册（这两个是现阶段接触最多的手册）
- 参考手册的内容很多，懂得如何使用参考手册（也就是迅速找到自己用得到的内容）是一个循序渐进的过程

#### 3. 任务要求

- 了解 GPIO 的功能框图，知道哪部分是 GPIO 输出、GPIO 输入、外设复用功能、ADC 功能，并将 GPIO 的各种模式和它们联系起来
- 根据功能框图，说明三种 TIM（高级，通用，基本）的联系，不必细化到都有什么功能/每个部分都是干什么的（提示：你总能在高级和通用里找到和基本定时器拓扑结构相像的部分 ^\_^）
- （这项不做要求）学有余力的同学可以看看通用定时器的各个寄存器，看不懂很正常（

## 8.1 GPIO功能描述

每个GPIO端口有两个32位配置寄存器(GPIOx\_CRL, GPIOx\_CRH), 两个32位数据寄存器(GPIOx\_IDR和GPIOx\_ODR), 一个32位置位/复位寄存器(GPIOx\_BSRR), 一个16位复位寄存器(GPIOx\_BRR)和一个32位锁定寄存器(GPIOx\_LCKR)。

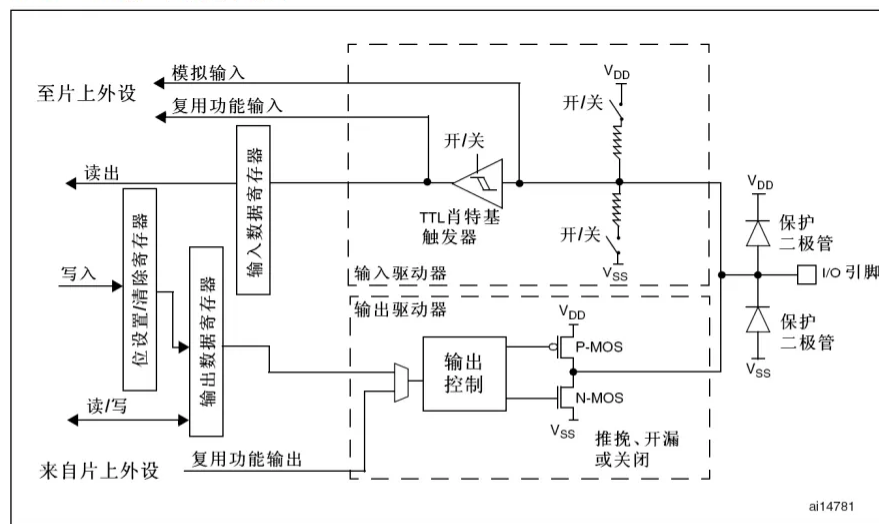
根据数据手册中列出的每个I/O端口的特定硬件特征, GPIO端口的每个位可以由软件分别配置成多种模式。

- 输入浮空
- 输入上拉
- 输入下拉
- 模拟输入
- 开漏输出
- 推挽式输出
- 推挽式复用功能
- 开漏复用功能

每个I/O端口位可以自由编程, 然而I/O端口寄存器必须按32位字被访问(不允许半字或字节访问)。GPIOx\_BSRR和GPIOx\_BRR寄存器允许对任何GPIO寄存器的读/更改的独立访问; 这样, 在读和更改访问之间产生IRQ时不会发生危险。

下图给出了一个I/O端口位的基本结构。

图13 I/O端口位的基本结构



105/754

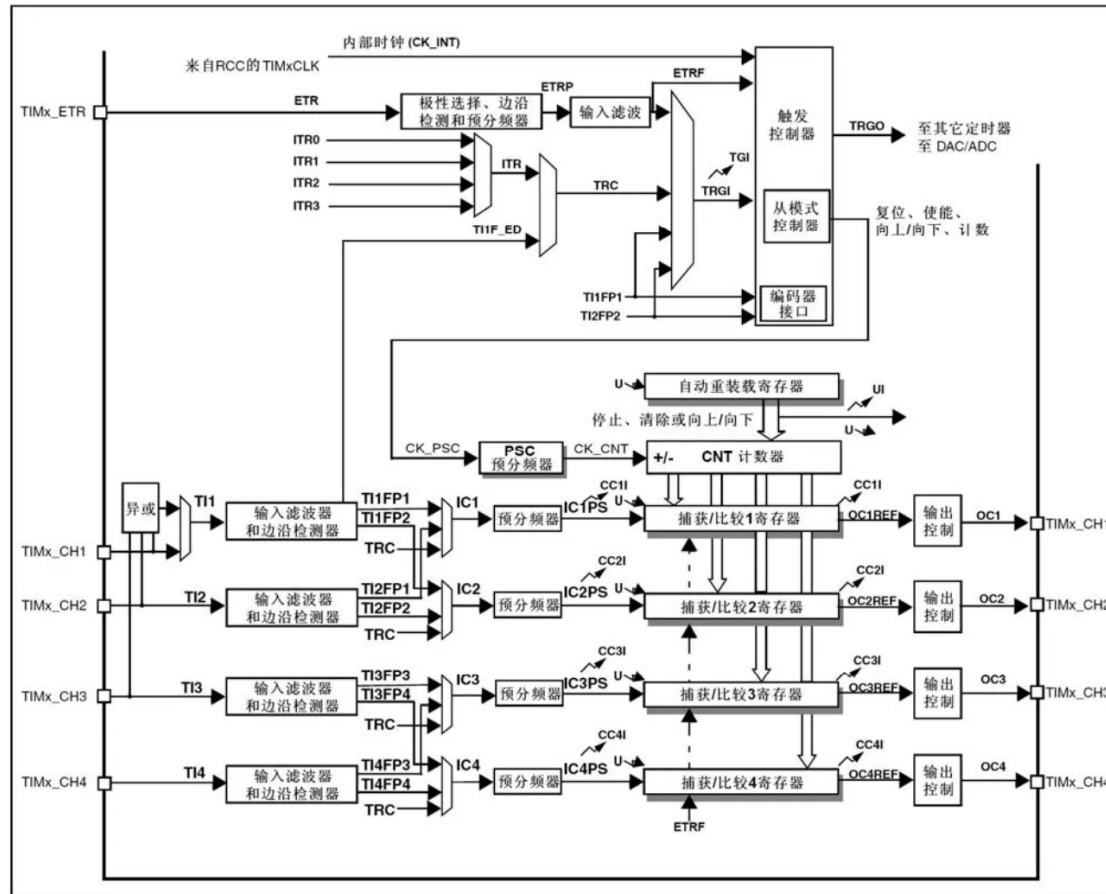





参照2009年12月 RM0008 Reference Manual 英文第10版  
本译文仅供参考, 如有翻译错误, 请以英文原稿为准。请读者随时注意在ST网站下载最新版本

718\_LAB

GPIO

图98 通用定时器框图



注：  根据控制位的设定，在U事件时传送预加载寄存器的内容至工作寄存器  
 事件  
 中断和DMA输出

通用定时器，看中间 CNT 计数器是不是很眼熟？

## 4. 技术点介绍

- 参考手册的使用
- 外设功能，功能框图，寄存器的查看

## (四) 进阶任务：代码模块化思想

### 1. 任务介绍

你是否在做以上任务的时候，对各种代码看得懵懵懂懂，要配新的外设时不知道要改哪里？每次都是从头开始配置？

试试代码封装吧！把一切细节舍去，只保留填入必要参数的接口，就此和烦恼告别！

## 2. 任务要求

- 实现一个函数，要求仅调用这个函数 就能实现指定 LED 的流水灯及相关外设的初始化（流水灯的延时要用 TIM 而不是 SysTick 那个 Delay() 函数实现），函数原型如下：

```
led_waterfall()

1 // 定义一个如下的枚举类型
2 typedef enum{
3     led1 = (0x01 << 0),
4     led2 = (0x01 << 1),
5     led3 = (0x01 << 2),
6     led4 = (0x01 << 3),
7     led5 = (0x01 << 4),
8     led6 = (0x01 << 5)
9 }leds;
10
11 // 函数原型，也就是说你最终调用的函数得长这样
12 void led_waterfall(uint8_t leds);
13
14 // 函数调用示例
15 // 调用这个函数后，可以实现 LED1 LED2 LED6 三灯的流水灯
16 led_waterfall(led1 | led2 | led6);
17
```

- 模仿 led.c 和 led.h 文件，将这个函数放入一个 waterfall.c 文件，将函数声明放入同名 waterfall.h 文件，通过在 main.c 里包含这个 .h 文件实现这个函数的调用。（你可以自己检验这个任务是否实现：把这两个文件直接放到我们提供的例程里的“空白工程”里，在 main() 函数里包含头文件后，仅调用这个函数，也实现同样效果）

## 3. 技术点介绍

- 跨文件调用

- 封装的意义
- 条件编译的使用

## (五) 进阶任务：共地和上拉\下拉电阻的作用

### 1. 任务介绍

有的时候，电路中某些模块会与整个电路断开，使这些模块的电位处于一种不确定的“悬浮”状态，称为电路浮空。为了把不确定的状态变成确定的状态，硬件工程师们通常会用什么手段呢？

### 2. 任务要求

- 了解并简述电路的共地，以及不共地可能导致的后果
- 了解并简述上拉电阻和下拉电阻的作用
- 结合你对上述知识的理解，分析按键按下和断开时NRST的电势以及电阻R4的作用（提示：电容C16由于硬件消抖，分析时直接看成断路即可）

