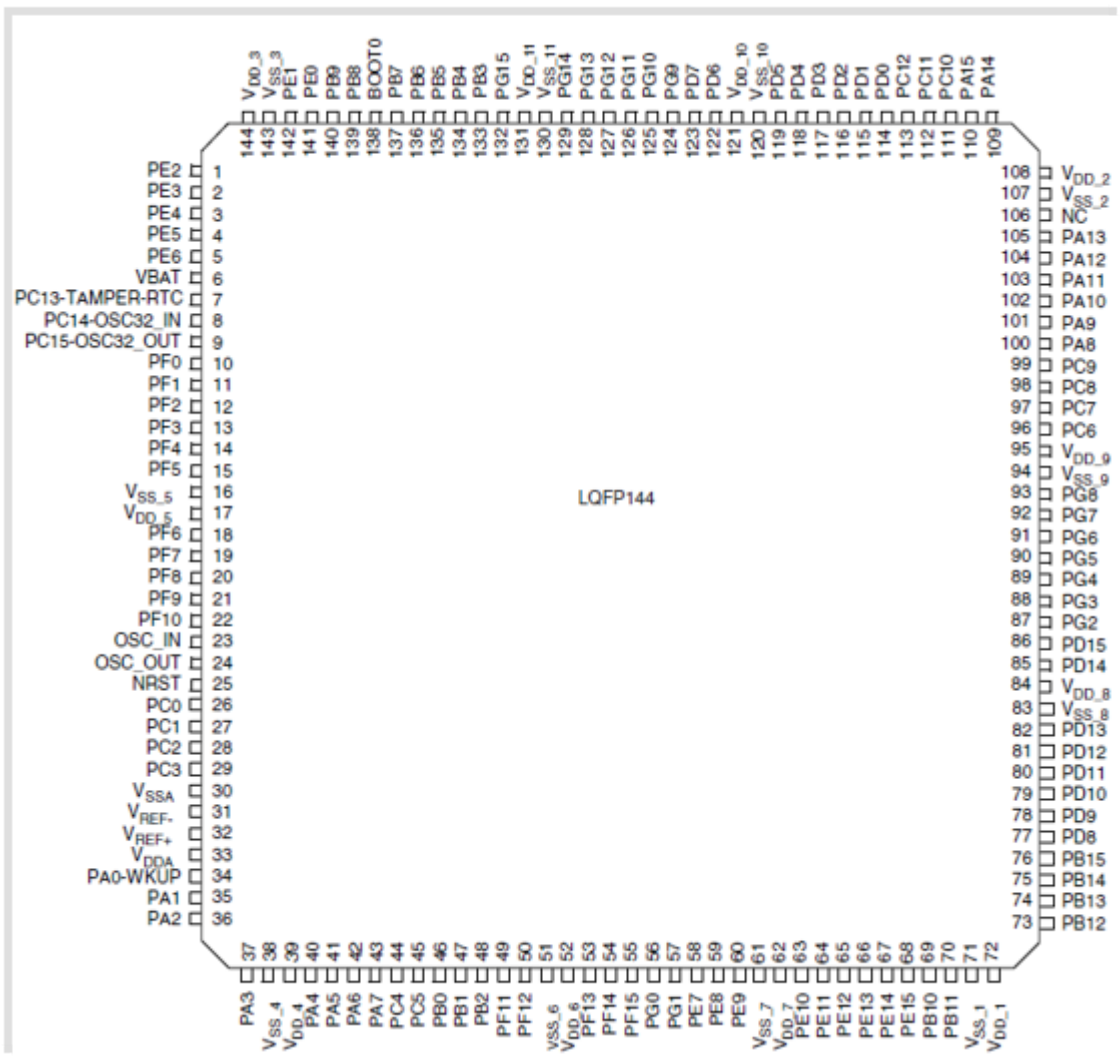
Week4-1 GPIO

一、概念介绍

GPIO（general purpose intput output）是通用输入输出端口的简称，可以通过软件来控制其输入和输出。
STM32 芯片的 GPIO 引脚与外部设备连接起来，从而实现**与外部通讯、控制以及数据采集**的功能。

简化一点！想想**是怎么点亮 LED 灯的！**我们是**通过软件控制改变LED灯的那个引脚的输出高低电平**，对吧？这就是一个最简单的GPIO**输出应用**！其他输出应用比如接继电器或者三极管，通过继电器或三极管来控制外部大功率电路的通断。当然 GPIO 还可以作为**输入控制**，比如在引脚上接入一个按键，通过电平的高低判断按键是否按下，这也是非常实用的！

week2已经把*STM32F1开发指南(精英版)-寄存器版本_V1.3.pdf*文档给大家了，相信你也看到了，STM32有超多引脚！



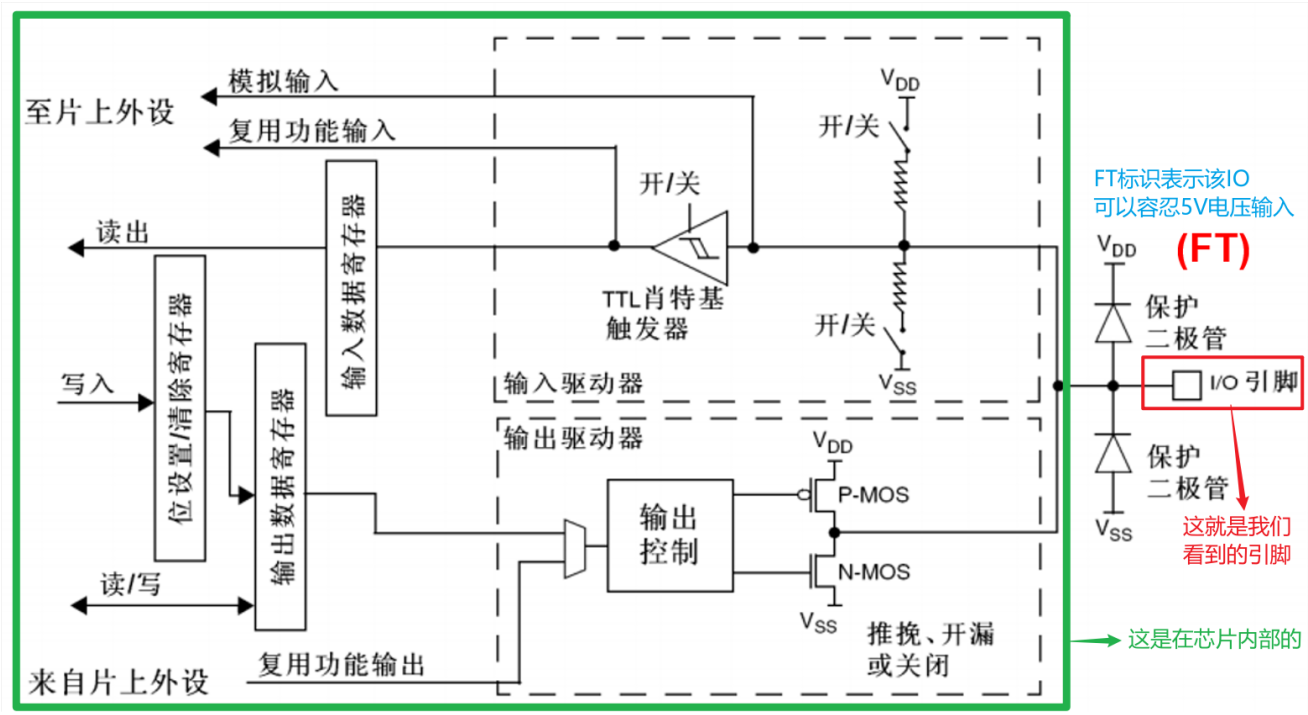
那么我们就把这么些引脚分分类叭！大致可以分为以下几类：

- 电源引脚：引脚图中的 VDD、VSS、VREF+、VREF-、VSSA、VDDA 等都属于电源引脚。
- 晶振引脚：引脚图中的 PC14、PC15 和 OSC_IN、OSC_OUT 都属于晶振引脚，不过它们还可以作为普通引脚使用。

- 复位引脚：引脚图中的 NRST 属于复位引脚，不做其他功能使用。
- 下载引脚：引脚图中的 PA13、PA14、PA15、PB3 和 PB4 属于 JTAG 或SW 下载引脚。不过它们还可以作为普通引脚或者特殊功能使用，具体的功能可以查看芯片数据手册，里面都会有附加功能说明。当然，STM32 的串口功能引脚也是可以作为下载引脚使用。
- BOOT 引脚：引脚图中的 BOOT0 和 PB2(BOOT1)属于 BOOT 引脚，PB2 还可以作为普通管脚使用。在 STM32 启动中会有模式选择，其中就是依靠着BOOT0和 BOOT1 的电平来决定。
- **GPIO 引脚**：STM32F103ZET6芯片为144脚芯片，包括7个通用目的的输入/输出口（GPIO）组，分别为GPIOA、GPIOB、GPIOC、GPIOD、GPIOE、GPIOF、GPIOG，同时每组GPIO口组有16个GPIO口。通常简略称为PAx、PBx、PCx、PDx、PEx、PFx、PGx，其中x为0-15。 大家可以在*原理图*上找一下哦，很容易就找到！

二、工作原理

当当！这是GPIO基本结构系统框图！一个小小引脚背后大有乾坤！



这个图是不是有点眼熟！BriMonzZY已经在week1给大家展示过啦，忘记的可以移步week1/Smatcarer必须要了解的一些软件知识/IO典型结构看哦！在这儿再推荐一个链接叭！

[STM32的GPIO工作原理（附电路图详细分析）]

一些电路知识：

> 保护二极管：IO引脚上下两边两个二极管用于防止引脚外部过高、过低的电压输入。当引脚电压高于VDD时，上方的二极管导通；当引脚电压低于VSS时，下方的二极管导通，防止不正常电压引入芯片导致芯片烧毁。
>
> P-MOS管和N-MOS管：由P-MOS管和N-MOS管组成的单元电路使得GPIO具有“推挽输出”和“开漏输出”的模式。
>
> TTL肖特基触发器：信号经过触发器后，模拟信号转化为0和1的数字信号。但是，当GPIO引脚作为ADC采集电压的输入通道时，用其“模拟输入”功能，此时信号不再经过触发器进行TTL电平转换。ADC外设要采集到的原始的模拟信号。

关于这三个问题该文章也提到了哦：

> 1、什么是推挽结构和推挽电路?
>
> 2、开漏输出和推挽输出的区别?
>
> 3、在STM32中选用怎样选择I/O模式?

三、GPIO相关配置寄存器

每组GPIO端口的寄存器包括：

- 两个32位配置寄存器(GPIOx_CRL和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)，
- 一个32位锁定寄存器(GPIOx_LCKR)。

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)。

每组IO口含下面7个寄存器。也就是7个寄存器，一共可以控制一组GPIO的16个IO口。

- GPIOx_CRL：端口配置低寄存器
- GPIOx_CRH：端口配置高寄存器
- GPIOx_IDR：端口输入寄存器
- GPIOx_ODR：端口输出寄存器
- GPIOx_BSRR：端口位设置/清除寄存器
- GPIOx_BRR：端口位清除寄存器
- GPIOx_LCKR：端口配置锁存寄存器

每一个寄存器的具体配置可参看视频链接：第12讲 STM32F1 GPIO工作原理 p11 22'30''-34'00''

四、引脚说明

人家把STM32F103ZET6 的IO资源分配做了一个总表，以便大家查阅，部分截图如下图所示，完整Excel表格传到本周document/*精英板IO引脚分配表.xlsx*里面啦！

精英板 IO资源分配表						使用提示
引脚编号	GPIO	连接资源		完全独立	连接关系说明	
34	PA0	WK_UP		Y	1，按键KEY_UP 2，可以做待机唤醒脚(WKUP)	只要KEY_UP不按下，该IO完全独立
35	PA1	STM_ADC	TPAD	Y	ADC输入引脚，同时做TPAD检测脚	拔了P7的跳线帽，则该IO完全独立
36	PA2	USART2_TX	485_RX	Y	RS485 RX脚(P5设置)	该IO通过P5选择是否连接RS485_RX,去掉P5的跳线帽，则该IO完全独立
37	PA3	USART2_RX	485_TX	Y	RS485 TX脚(P5设置)	该IO通过P5选择是否连接RS485_TX,去掉P5的跳线帽，则该IO完全独立
40	PA4	STM_DAC	GBC_KEY	Y	1，DAC_OUT1输出脚 2，ATK-MODULE接口的KEY引脚	该IO可做DAC输出，同时也连接在ATK MODULE接口，如不插外设在ATK-MODULE接口，则可以完全独立
41	PA5			Y	未接任何外设	该IO未接任何外设，完全独立
42	PA6			Y	未接任何外设	该IO未接任何外设，完全独立
43	PA7			Y	未接任何外设	该IO未接任何外设，完全独立
100	PA8	OV_VSYNC		Y	OLED/CAMERA接口的VSYNC脚	仅连接OLED/CAMERA接口的VSYNC，当不使用OLED/CAMERA接口时，该IO完全独立
101	PA9	USART1_TX		Y	串口1 TX脚，默认连接CH340的RX(P3设置)	该IO通过P3选择是否连接CH340的RXD，如果不连接，则该IO完全独立
102	PA10	USART1_RX		Y	串口1 RX脚，默认连接CH340的TX(P3设置)	该IO通过P3选择是否连接CH340的TXD，如果不连接，则该IO完全独立
103	PA11	USB_D-	CRX	Y	1，USB D-引脚(P6设置) 2，CRX引脚(P6设置)	该IO通过P6选择连接USB D-还是CAN的RX脚，如果去掉P6的跳线帽，则该IO完全独立
104	PA12	USB_D+	CTX	Y	1，USB D+引脚(P6设置) 2，CTX引脚(P6设置)	该IO通过P6选择连接USB D+还是CAN的TX脚，如果去掉P6的跳线帽，则该IO完全独立
105	PA13	JTMS	SWDIO	N	JTAG/SWD仿真接口，没接任何外设 注意：如要做普通IO，需先禁止JTAG&SWD	JTAG/SWD仿真接口，没连外设。建议仿真器选择SWD调试，这样仅SWDIO和SWDCLK两个信号即可仿真。该IO做普通IO用(有10K上/下拉电阻)，需先禁止JTAG&SWD！此时无法仿真！ 库函数全禁止方法：GPIO_PinRemapConfig(GPIO_Remap_SWJ_Disable) 寄存器全禁止方法：JTAG_Set(JTAG_SWD_DISABLE)
109	PA14	JTCK	SWDCLK	N	JTAG/SWD仿真接口，没接任何外设 注意：如要做普通IO，需先禁止JTAG&SWD	
110	PA15	JTDI	GBC_LED	N	1，JTAG仿真口(JTDI) 2，ATK-MODULE接口的LED引脚(使用时，需先禁止JTAG，才可以当普通IO使用)	JTAG仿真口，也接ATK-MODULE接口的LED脚，如不用JTAG和ATK-MODULE接口，则可做普通IO用(有10K上拉电阻)。做普通IO用，需先禁止JTAG。此时可SWD仿真，但JTAG无法仿真。 库函数禁止JTAG方法： GPIO_PinRemapConfig(GPIO_Remap_SWJ_JTAGDisable)

仔细看一下表格，在*连接关系说明*里面为什么有的引脚如PA5、PA6、PA7写的是*未接任何外设*，而有的引脚如PA9、PA10写的是*串口1*.....呢？下面我们介绍两个概念。

端口复用：

为了**最大限度的利用端口资源**，STM32的大部分端口都具有复用功能。所谓复用，就是一些端口不仅仅可以做为通用IO口，还可以复用为一些外设引脚，比如PA9，PA10可以复用为STM32的串口1引脚。端口的复用情况可以通过查找数据手册（见Ducument/*STM32F103ZET6.pdf*）来获得（上面这个例子就可以在文件的第33页看到呢）。

端口重映射：

为了方便布线，STM32配有端口重映射功能，所谓重映射就是可以把某些功能引脚映射到其他引脚。比如串口1默认引脚是PA9，PA10可以通过配置重映射映射到PB6，PB7。端口的映射情况也可以通过查找数据手册（见Ducument/*STM32F103ZET6.pdf*）来获得。

进一步了解可以参看如下链接：GPIO的复用和重映射

五、GPIO库函数配置

5.1 重要函数

初始化函数：

```
void GPIO_Init(GPIO_TypeDef* GPIOx, GPIO_InitTypeDef* GPIO_InitStruct);
```

2个读取输入电平函数：

```
uint8_t GPIO_ReadInputDataBit(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin);

uint16_t  GPIO_ReadInputData(GPIO_TypeDef* GPIOx);
```

2个读取输出电平函数：

```
uint8_t  GPIO_ReadOutputDataBit(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin);
uint16_t  GPIO_ReadOutputData(GPIO_TypeDef* GPIOx);
```

4个设置输出电平函数：

```
void GPIO_SetBits(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin);
void GPIO_ResetBits(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin);
void GPIO_WriteBit(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin, BitAction BitVal);
void GPIO_Write(GPIO_TypeDef* GPIOx, uint16_t PortVal);
```

5.2 初始化函数

```
void GPIO_Init(GPIO_TypeDef* GPIOx, GPIO_InitTypeDef* GPIO_InitStruct);
```

作用：初始化一个或者多个IO口（同一组）的工作方式和速度。该函数主要是操作GPIO_CRL(CRH)寄存器，在上拉或者下拉的时候有设置BSRR或者BRR寄存器。

GPIO_InitTypeDef结构体（**结构体**，如果不清楚概念的请返回week2进行C语言复习哦）：

```
typedef struct  {
    uint16_t GPIO_Pin;                  //指定要初始化的IO口
    GPIOSpeed_TypeDef GPIO_Speed; //设置IO口输出速度
    GPIOMode_TypeDef GPIO_Mode;  //设置工作模式：8种中的一个
}
```

注意：外设（包括GPIO)在使用之前，几乎都要先使能对应的时钟。

```
RCC_APB2PeriphColckCmd();
```

GPIO输出速度（**枚举**，如果不清楚概念的请返回week2进行C语言复习哦）：

```
typedef enum {
    GPIO_Speed_10MHz,
    GPIO_Speed_2MHz,
    GPIO_Speed_50MHz
}
GPIOSpeed_TypeDef;
```

GPIO模式：

```
typedef enum
{
    GPIO_Mode_AIN = 0x0, //模拟输入
    GPIO_Mode_IN_FLOATING = 0x04, //浮空输入
    GPIO_Mode_IPD = 0x28, //输入下拉
```

```
    GPIO_Mode_IPU = 0x48, //输入上拉
    GPIO_Mode_Out_OD = 0x14, //开漏输出
    GPIO_Mode_Out_PP = 0x10, //推挽输出
    GPIO_Mode_AF_OD = 0x1C, //开漏复用输出
    GPIO_Mode_AF_PP = 0x18 //推挽复用输出
}GPIOMode_TypeDef;
```

GPIO_Init函数初始化样例：

```
GPIO_InitTypeDef  GPIO_InitStructure;
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_5; //LED0-->PB.5 端口配置 GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;  //推挽输出 GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz; //IO口速度为50MHz  GPIO_Init(GPIOB, &GPIO_InitStructure);  //根据设定参数初始化GPIOB.5
```

可以一次初始化一个IO组下的多个IO，前提是这些IO口的配置方式一样。

5.3 两个读取输入电平函数

```
uint8_t GPIO_ReadInputDataBit(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin);
GPIO_ReadInputDataBit(GPIOA, GPIO_Pin_5);//读取GPIOA.5的输入电平
```

作用：读取某组GPIO的输入电平。实际操作的是GPIOx_IDR寄存器。

5.4四个设置输出电平函数

```
void GPIO_SetBits(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin);
```

作用：设置某个IO口输出为高电平（1）。实际操作BSRR寄存器

```
void GPIO_ResetBits(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin);
```

作用：设置某个IO口输出为低电平（0）。实际操作的BRR寄存器。

```
void GPIO_WriteBit(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin, BitAction BitVal);
void GPIO_Write(GPIO_TypeDef* GPIOx, uint16_t PortVal);
```

这两个函数不常用，也是用来设置IO口输出电平。

5.5举例

跑马灯例程：

1. 使能IO口时钟。 调用函数RCC_APB2PeriphColckCmd();（不同IO组，调用时钟使能不一样）
2. 初始化IO口模式。 调用函数：GPIO_Init();
3. 操作IO口，输出高低电平。 GPIO_SetBits(); GPIO_ResetBits();

具体配置请参看视频手把手编写跑马灯实验-库函数h p12 51min

另：请各位仔细阅读document/*STM32不完全手册库函数版本V3.3.pdf* 第六章（p121-p137），加深对整个工程的认识！

六、GPIO寄存器配置

感兴趣的同学可以参看流水灯例程的寄存器版本程序。

```
//led.c源文件
#include "led.h"
#include "stm32f10x.h"

void LED_Init(void){

   RCC->APB2ENR|=1<<3;
   RCC->APB2ENR|=1<<6;

   //GPIOB.5
   GPIOB->CRL&=0xFF0FFFFF;//与运算，使GPIOB的CRL寄存器第20~23位(PB5)清零
   GPIOB->CRL|=0x00300000;//或运算，使GPIOB的CRL寄存器第20~23位置位0011，即设置为推挽输出、速率为50MHz
   GPIOB->ODR|=1<<5;//或运算，使GPIOB的输出数据寄存器ODR第5位置1即PB5输出高电平

   GPIOE->CRL&=0xFF0FFFFF;//与运算，使GPIOE的CRL寄存器第20~23位(PE5)清零
   GPIOE->CRL|=0x00300000;//或运算，使GPIOE的CRL寄存器第20~23位置位0011，即设置为推挽输出、速率为50MHz
   GPIOE->ODR|=1<<5;//或运算，使GPIOE的输出数据寄存器ODR第5位置1即PE5输出高电平
   //GPIOE.5
}
```

```
//main.c主程序
#include "sys.h"
#include "usart.h"
#include "delay.h"
#include "led.h"

int main(void)
{

   while(1){
      GPIOB->ODR|=1<<5;//或运算，使GPIOB的输出数据寄存器ODR第5位置1即PB5输出高电平
      GPIOE->ODR|=1<<5;//或运算，使GPIOE的输出数据寄存器ODR第5位置1即PE5输出高电平
      delay_ms(500);

      GPIOB->ODR&=~(1<<5);//与运算，使GPIOB的输出数据寄存器ODR第5位置0即PB5输出低电平
      GPIOE->ODR&=~(1<<5);//与运算，使GPIOE的输出数据寄存器ODR第5位置0即PE5输出低电平
      delay_ms(500);
   }
}
```

七、HAL库配置GPIO

感兴趣的同学可以参看如下链接，自行学习。

[STM32HAL库 STM32CubeMX教程三----外部中断(HAL库GPIO讲解)]

[STM32 GPIO详细篇（基于HAL库）]

STM32CubeMX应用教程 第一章 GPIO