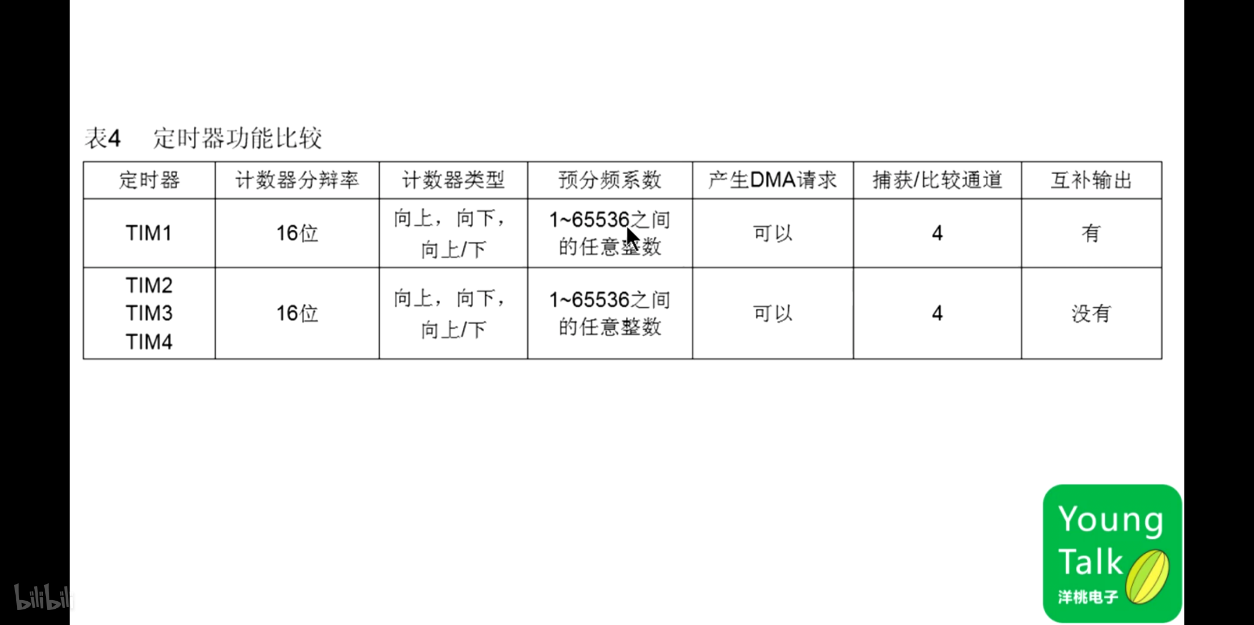
1. 低功耗和ADC
2. 低功耗包含睡眠、停机、待机模式；Vbat为RTC和后备寄存器供电。
3. 2个12位的ADC转换器，1us转换时间（16个输入通道），转换范围0-3.6V，最小分辨率12位ADC最大值4095，3.3V/4095=0.805mV,双采样和保持功能，温度传感器。扫描模式自动刷新，可以采用DMA。



1. 看门狗和滴答定时器
2. 独立看门狗基于12位的递减计数器和一个8位的预分频器，内部独立的40kHz的RC震荡器提供时钟；因为独立时钟，所以可以运行在停机和待机模式。可以被当成看门狗在系统出问题时复位系统使用或者当作一个自由的定时器提供超时处理。
3. 窗口看门狗是一个7位的递减计数器，可以设置为自由运行，可以被当作看门狗在系统发生问题时复位整个系统，由主时钟驱动。可以给出中断或者复位信号。
4. 滴答定时器专用于实时操作系统，也可以当作标准递减计数器。具有以下特性：24位递减计数器；自动重加载；当计数器为0时会产生一个可屏蔽的系统中断；可编程时钟源。
5. 普通定时器和高级定时器
6. 普通定时器：TIM2,TIM3,TIM4三个16位可同步运行的普通定时器，每个定时器有4个用于输入捕获/输出比较/pwm或者脉冲计数的通道和增量编码输出，一个16位的预分频器，一个16位的自动加载递减/递加计数器。
7. 高级定时器：TIM1 16位的的高级定时器，可以被看成是分配到6个通道的三相pwm发生器。带死区控制和紧急刹车，用于电机控制和pwm高级控制定时器。

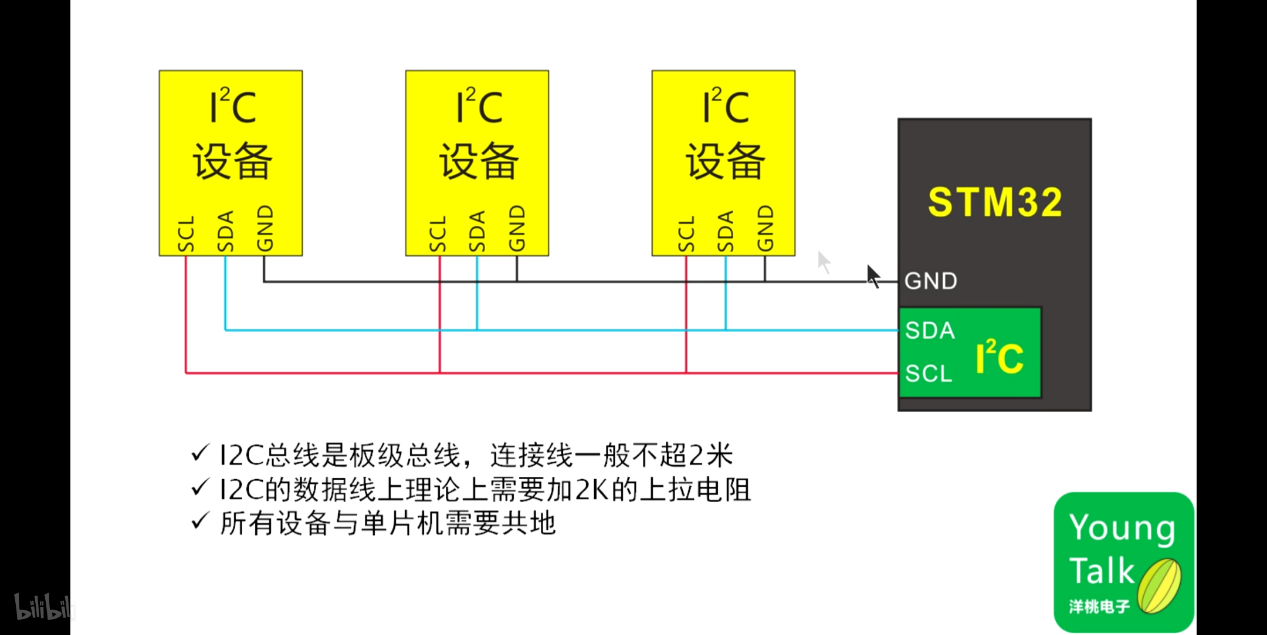


1. 串行接口

通信接口资源，2个iic接口，3个usart接口，2个spi,can,usb2.0

1. IIC总线

多达2个iic接口，能够工作于多主模式或从模式，支持标准和快速模式。支持7位或者10位地址寻址，7位从模式支持双从地址寻址，内置硬件crc发生器/校验器。可以使用DMA操作，支持SMBUS（简化的iic）和PMBUS。

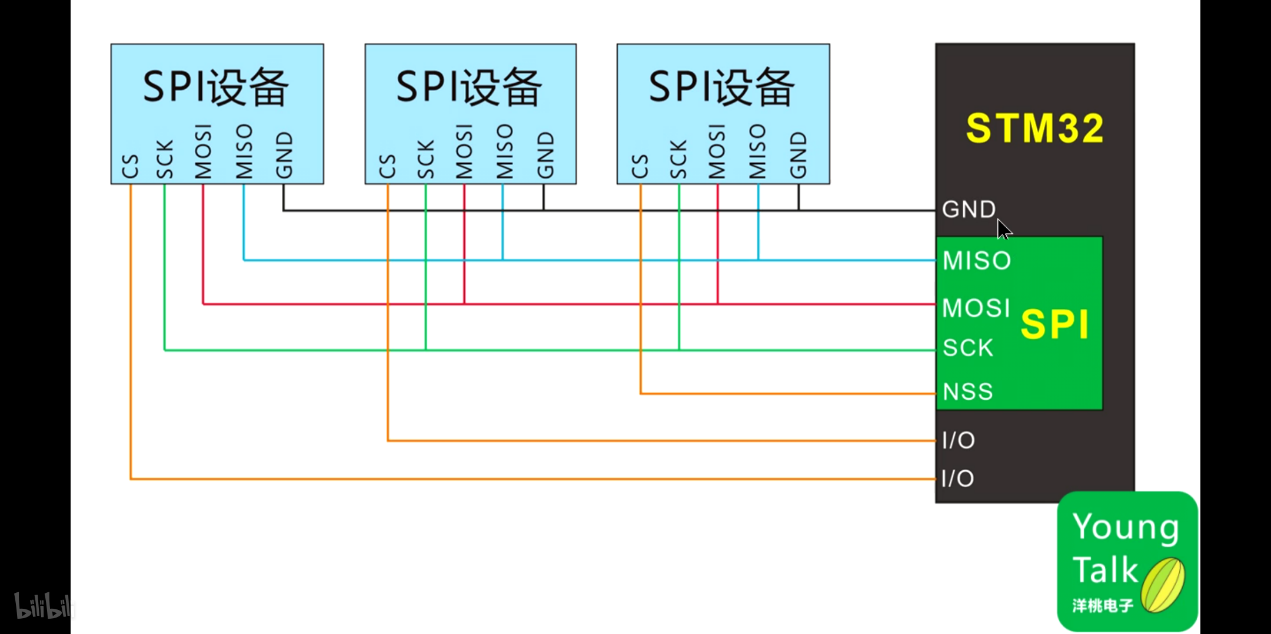


1. 通用同步/异步收发器（USART）

USART1速度可达4.5Mbit/s,其他的2.25Mbit/s,具有硬件的CTS和RTS信号管理。USART是通用同步/异步收发器，带同步时钟线的。UART是通用异步收发器，无时钟线。USART只是一种协议的方式，根据不同电平方式分为RS232（RS232接口任何一条信号线的电压均为负逻辑关系。即：逻辑“1”为-3—-15V；逻辑“0”：+3—+15V，噪声容限为2V。）和RS485（逻辑“1”以两线间的电压差+2V~+6V表示，逻辑“0”以两线间的电压差-6V~-2V表示。）等

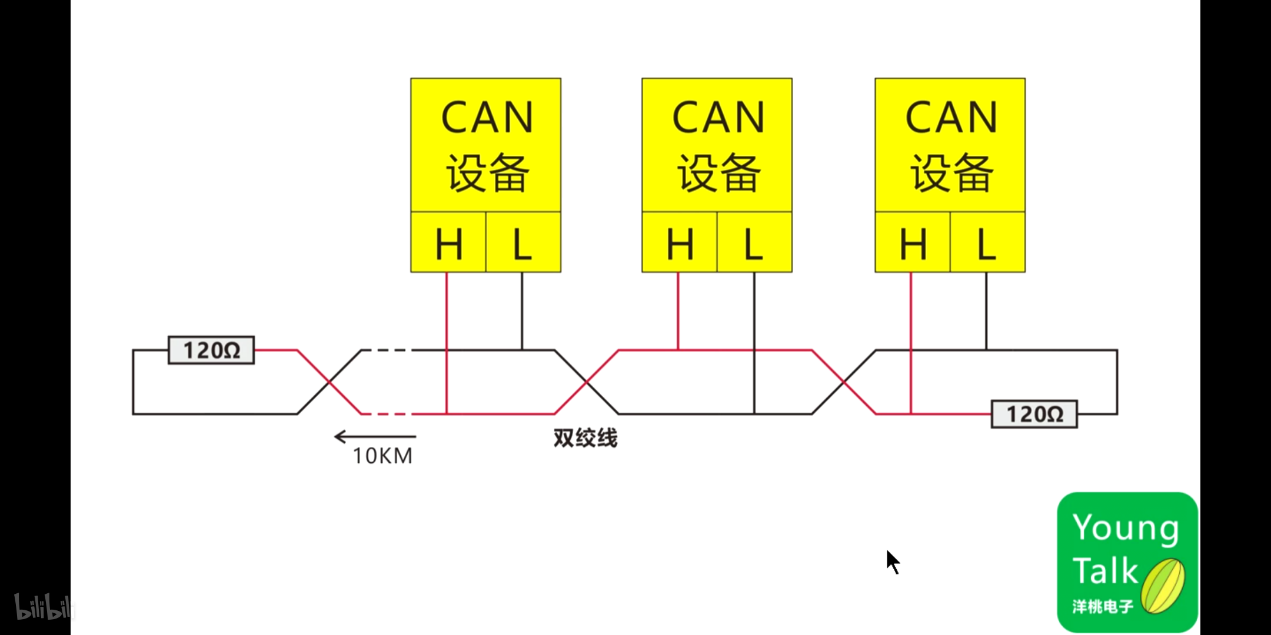
1. 串行外设接口（SPI）

2个spi接口，在从或者主模式，全双工和半双工的通信速度18Mbit/s，3位预分频器，8种主模式频率，可配置每帧8位或者16位。硬件CRC，SD卡和MMC模式，可用DMA操作。 特点：协议简单稳定；速度快。片选线独立，从设备多时需要较多的io口。



1. 控制器区域网络（CAN）

CAN速率1Mbit/s,接收和发送11位标识的标准帧，也可以接收和发送29位标识符的扩展帧。具有3个发送邮箱和2个接收FIFO,3级14个可调节的滤波器。特点：通信速度快、距离远、稳定、自动查错。



1. 通用串行总线(usb)
2. 1个USB接口
3. 设备控制器
4. 支持全速2.0，12Mbit/s
5. 有待机和唤醒功能
6. 内部PLL倍频器提供时钟
7. 时钟必须由外部高速晶振产生
8. CRC校验和芯片ID
9. CRC计算单元
10. CRC是用于数据正确性校验
11. 由一个32位的数据字产生
12. 可应用在Flash检测
13. 可用于软件签名及对比
14. 芯片ID

96位编码，唯一，用来作产品序列号，作为密码，提高安全性。用来保护程序不可复制。

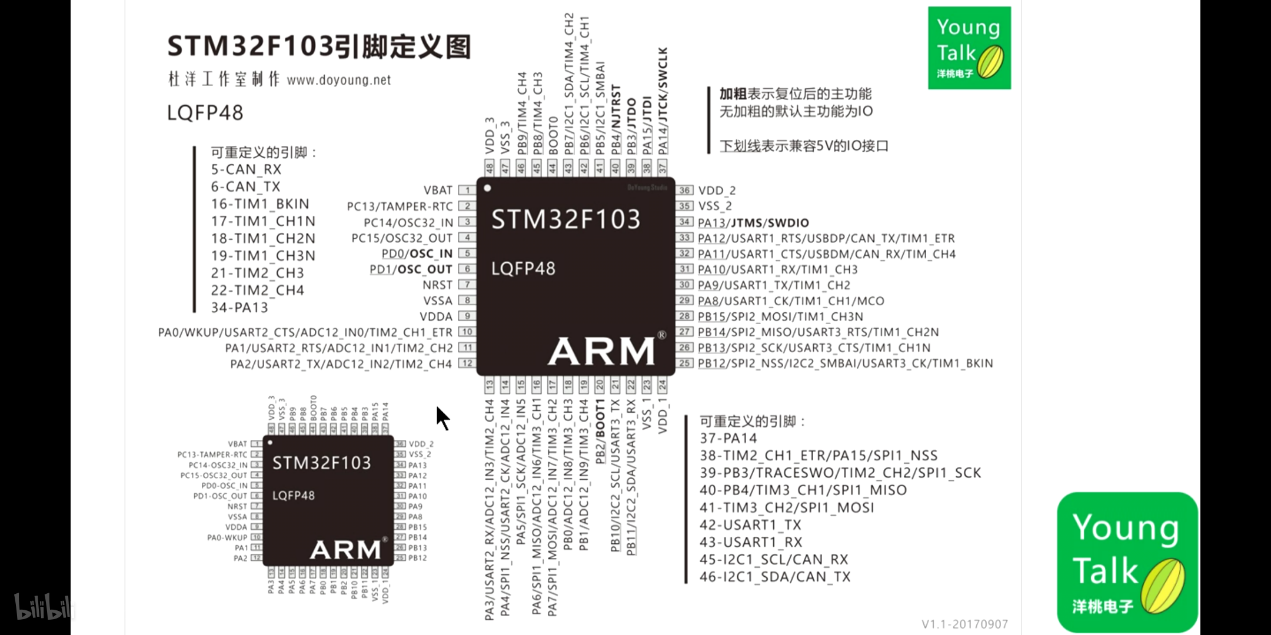
1. 嵌套的向量式中断控制器（NVIC）

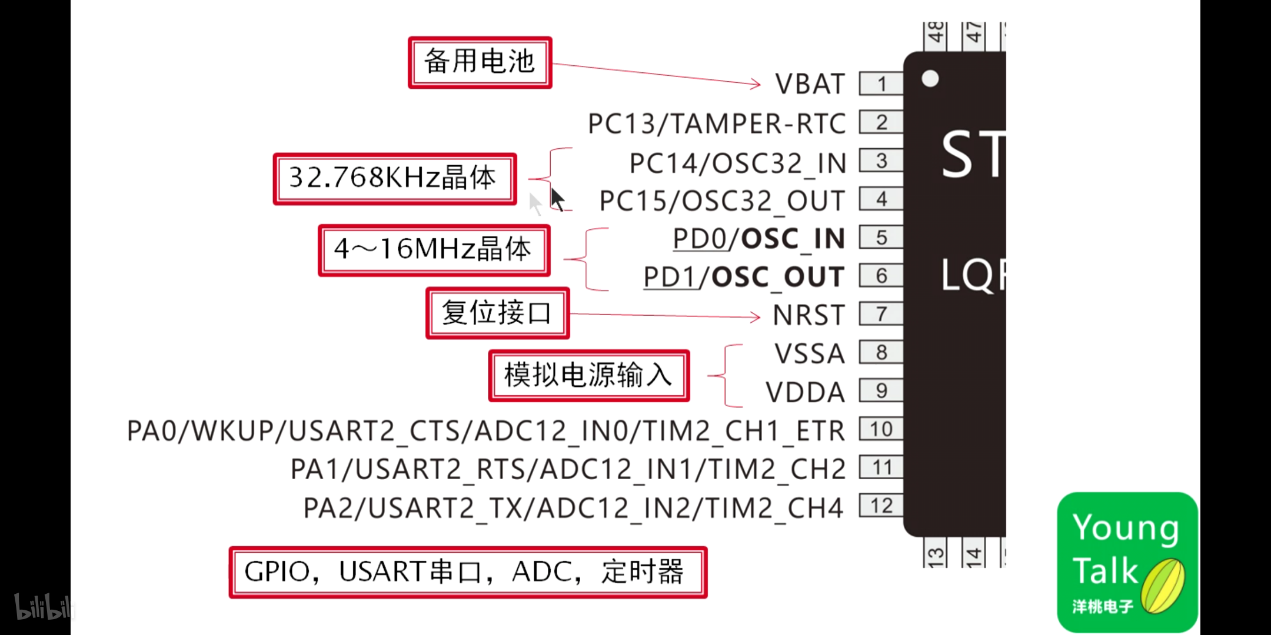
能够处理多达43个可屏蔽中断通道和16个优先级。

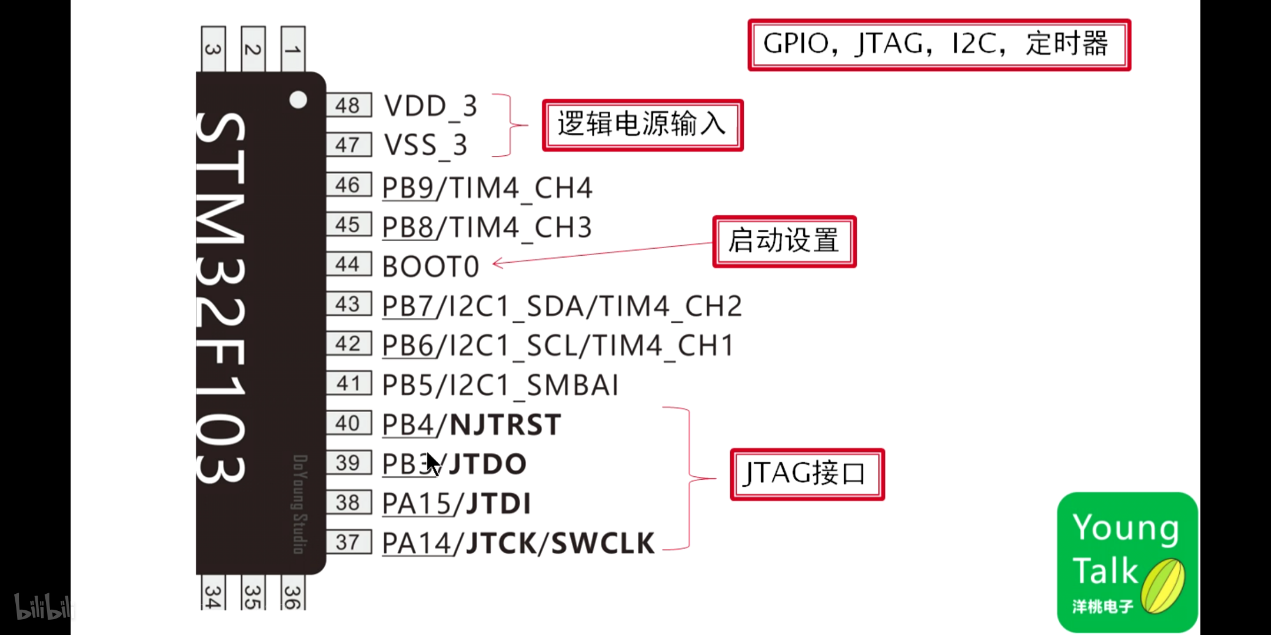
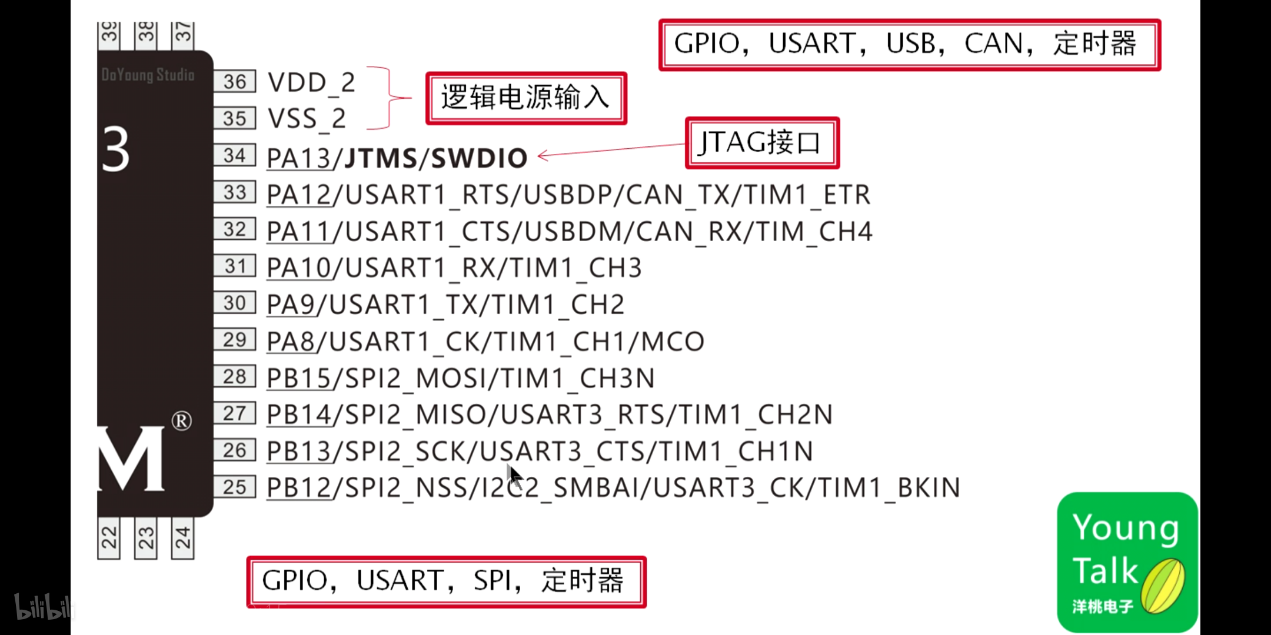
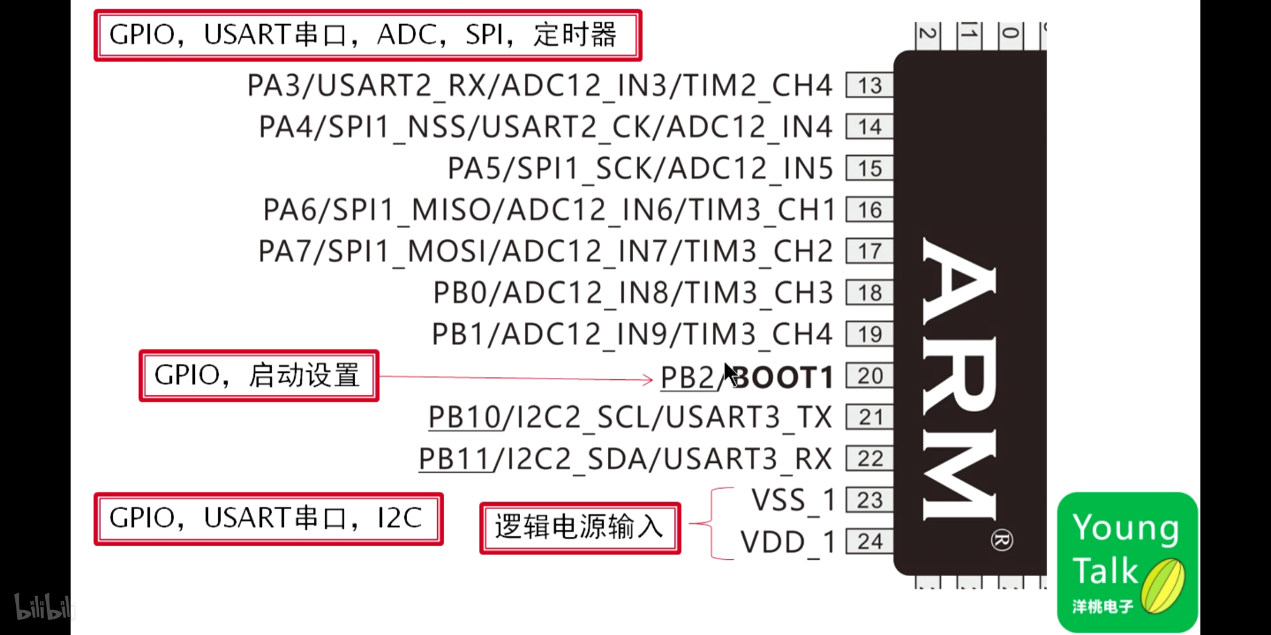
1. 外部中断/事件控制器（EXTI）

包含19个边沿检测器，用于产生中断/事件请求。每个中断都能独立配置触发事件，并能单独地被屏蔽。

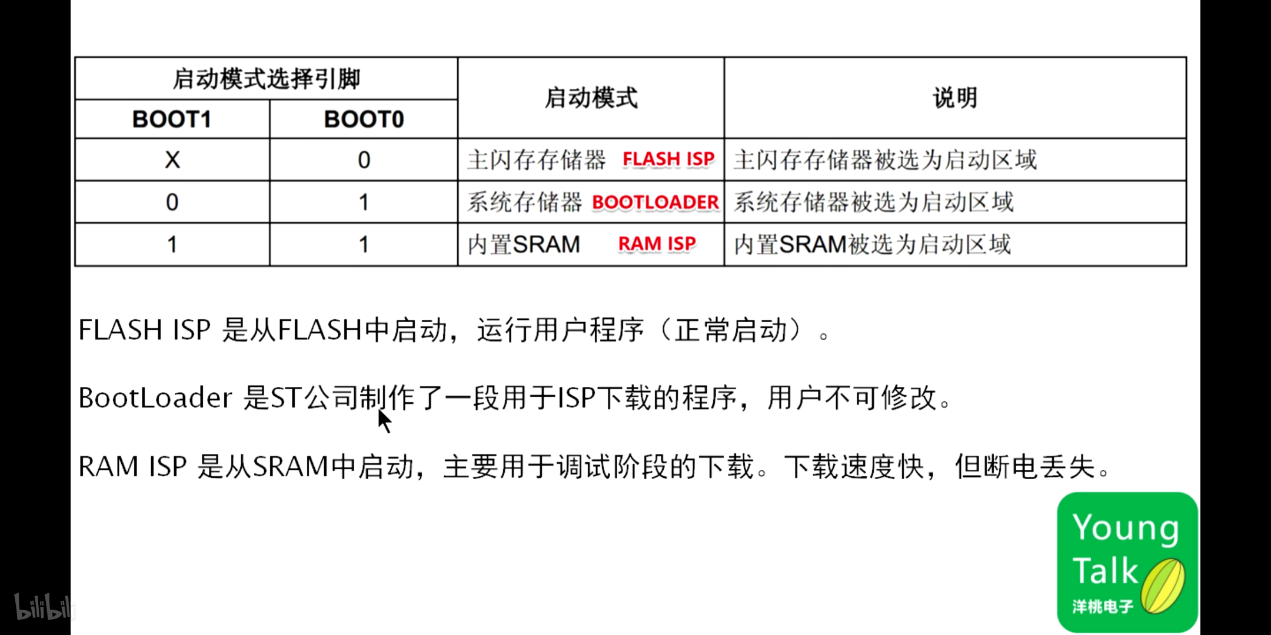
1. 接口定义







1. 启动模式



1. 工程中文件的介绍

CMSIS:内核驱动程序

Lib:内部功能的基本函数库

Startup:单片机启动程序

User:用户程序（包括main）

Basic:内部功能的驱动程序

Hardware:外部硬件的驱动程序

十一、GPIO 读写的三种操作

1、void GPIO\_WriteBit(GPIO\_TypeDef\* GPIOx, u16 GPIO\_Pin, BitAction BitVal)；

2、GPIO\_SetBits(GPIO\_TypeDef\* GPIOx, u16 GPIO\_Pin) 和

GPIO\_ResetBits(GPIO\_TypeDef\* GPIOx, u16 GPIO\_Pin)

3、void GPIO\_Write(GPIO\_TypeDef\* GPIOx, u16 PortVal)

十二、FLASH 操作



1. 操作步骤
2. 打开高速时钟（HIS/HSE）:RCC\_HSICmd(ENABLE);
3. 解锁FLASH编程擦除控制器 FLASH\_Unlock();
4. 清除标志位（忙标志位，操作结束标志位，编写错误标志位，页面写保护错误标志位）FLASH\_ClearFlag(FLASH\_FLAG\_BSY|FLASH\_FLAG\_EOP|FLASH\_FLAG\_PGERR|FLASH\_FLAG\_WRPRTERR);
5. 擦出指定页：FLASH\_ErasePage(Address);
6. 从指定页的addr地址开始写：FLASH\_ProgramHalfWord(Address,Data);
7. 擦出标志位
8. 锁定FLASH编程擦除控制器 FLASH\_lock();
9. 注意事项
10. 操作一定要先擦后写
11. 每页是1024个地址，起始地址是0x8000000
12. 擦除操作以页为单位，写操作则必须以16位宽度为单位，允许跨页写入。
13. STM32内置Flash操作时，必须打开高速时钟振荡器。
14. Flash可多次擦写10万次，不可死循环擦写。
15. 擦写时要避开用户程序储存区的区域，否则会擦掉用户程序导致错误。
16. 擦除一页要10ms(1k一页)，不能单个字节擦写。

十三、USART 操作

1. USART 初始化
2. 使能串口，GPIOA时钟：

RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_USART1|RCC\_APB2Periph\_GPIOA, ENABLE);

1. 初始化GPIO口：

GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_9; //PA.9

GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_50MHz;

GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_AF\_PP; //复用推挽输出

GPIO\_Init(GPIOA, &GPIO\_InitStructure);

//USART1\_RX PA.10

GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_10;

GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_IN\_FLOATING;//浮空输入

GPIO\_Init(GPIOA, &GPIO\_InitStructure);

1. NVIC配置

NVIC\_InitStructure.NVIC\_IRQChannel = USART1\_IRQn;

NVIC\_InitStructure.NVIC\_IRQChannelPreemptionPriority=3 ;//抢占优先级3

NVIC\_InitStructure.NVIC\_IRQChannelSubPriority = 3;//子优先级3

NVIC\_InitStructure.NVIC\_IRQChannelCmd = ENABLE;//IRQ通道使能

NVIC\_Init(&NVIC\_InitStructure); //根据指定的参数初始化VIC寄存器

1. 串口配置

USART\_InitStructure.USART\_BaudRate = bound;//一般设置为9600;

USART\_InitStructure.USART\_WordLength= USART\_WordLength\_8b;//字长为8位数据格式

USART\_InitStructure.USART\_StopBits = USART\_StopBits\_1;//一个停止位

USART\_InitStructure.USART\_Parity = USART\_Parity\_No;//无奇偶校验位

USART\_InitStructure.USART\_HardwareFlowControl=USART\_HardwareFlowControl\_None;//无硬件数据流控制

USART\_InitStructure.USART\_Mode=USART\_Mode\_Rx|USART\_Mode\_Tx; //收发模式

USART\_Init(USART1, &USART\_InitStructure); //初始化串口

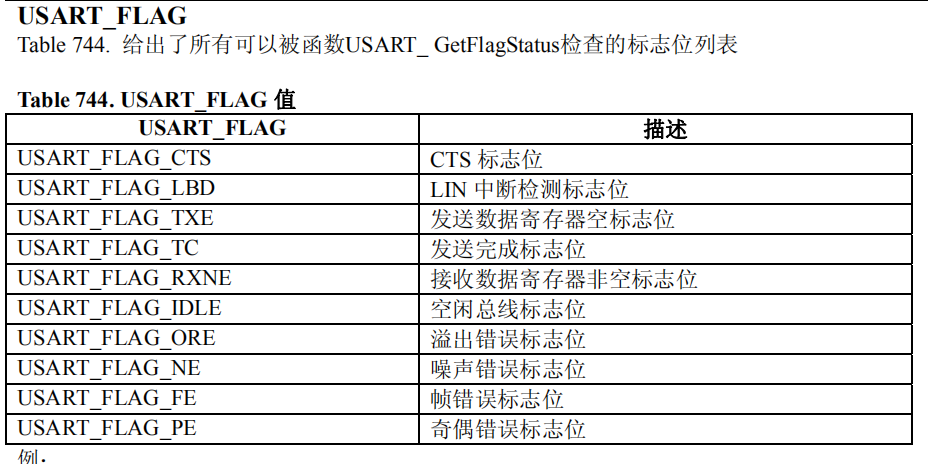
USART\_ITConfig(USART1, USART\_IT\_RXNE, ENABLE);//开启ENABLE/关闭DISABLE中断

USART\_Cmd(USART1, ENABLE); //使能串口

1. 串口发送
2. 使用库函数

USART\_SendData(USART1 , 0x55); //发送单个数值

while(USART\_GetFlagStatus(USART1, USART\_FLAG\_TC)==RESET); //检查发送中断标志位



1. 串口接收
2. 查询接收方式

使用USART\_GetFlagStatus检查USART\_FLAG\_RXEN是否为1（收到数据）

if(USART\_GetFlagStatus(USART1,USART\_FLAG\_RXNE) != RESET){ //查询串口待处理标志位

a =USART\_ReceiveData(USART1);//读取接收到的数据

1. 中断接收方式

if(USART\_GetITStatus(USART1, USART\_IT\_RXNE) != RESET){//接收中断

a =USART\_ReceiveData(USART1);//读取接收到的数据。

十四、RTC（实时时钟）和后备寄存器

后备寄存器是10个16位的寄存器，由Vbat引脚供电，保存20个字节的用户数据。RTC和后备寄存器不会被系统或电源复位；从待机模式唤醒，也不会被唤醒。RTC具有一个32位的可编程计数器，使用比较寄存器可以进行长时间的测量。有一个20位的预分频器用于时基时钟，默认情况下时钟为32.768KHZ时，它将产生一个1秒长度时间基准。

void RTC\_First\_Config(void){ //首次启用RTC的设置

RCC\_APB1PeriphClockCmd(RCC\_APB1Periph\_PWR | RCC\_APB1Periph\_BKP, ENABLE);//启用PWR和BKP的时钟（from APB1）

PWR\_BackupAccessCmd(ENABLE);//后备域解锁

BKP\_DeInit();//备份寄存器模块复位

RCC\_LSEConfig(RCC\_LSE\_ON);//外部32.768KHZ晶振开启

while (RCC\_GetFlagStatus(RCC\_FLAG\_LSERDY) == RESET);//等待稳定

RCC\_RTCCLKConfig(RCC\_RTCCLKSource\_LSE);//RTC时钟源配置成LSE（外部低速晶振32.768KHZ）

RCC\_RTCCLKCmd(ENABLE);//RTC开启

RTC\_WaitForSynchro();//开启后需要等待APB1时钟与RTC时钟同步，才能读写寄存器

RTC\_WaitForLastTask();//读写寄存器前，要确定上一个操作已经结束

RTC\_SetPrescaler(32767);//设置RTC分频器，使RTC时钟为1Hz,RTC period = RTCCLK/RTC\_PR = (32.768 KHz)/(32767+1)

RTC\_WaitForLastTask();//等待寄存器写入完成

//当不使用RTC秒中断，可以屏蔽下面2条

NVIC\_InitStructure.NVIC\_IRQChannel = RTC\_IRQn;

NVIC\_InitStructure.NVIC\_IRQChannelPreemptionPriority=3 ;//抢占优先级3

NVIC\_InitStructure.NVIC\_IRQChannelSubPriority = 2; //子优先级3

NVIC\_InitStructure.NVIC\_IRQChannelCmd = ENABLE; //IRQ通道使能

NVIC\_Init(&NVIC\_InitStructure); //根据指定的参数初始化VIC寄存器

RTC\_ITConfig(RTC\_IT\_SEC, ENABLE);//使能秒中断

RTC\_WaitForLastTask();//等待写入完成

}

void RTC\_Config(void){ //实时时钟初始化

//在BKP的后备寄存器1中，存了一个特殊字符0xA5A5

//第一次上电或后备电源掉电后，该寄存器数据丢失，表明RTC数据丢失，需要重新配置

if (BKP\_ReadBackupRegister(BKP\_DR1) != 0xA5A5){//判断寄存数据是否丢失

RTC\_First\_Config();//重新配置RTC

BKP\_WriteBackupRegister(BKP\_DR1, 0xA5A5);//配置完成后，向后备寄存器中写特殊字符0xA5A5

}else{

//若后备寄存器没有掉电，则无需重新配置RTC

//这里我们可以利用RCC\_GetFlagStatus()函数查看本次复位类型

if (RCC\_GetFlagStatus(RCC\_FLAG\_PORRST) != RESET){

//这是上电复位

}

else if (RCC\_GetFlagStatus(RCC\_FLAG\_PINRST) != RESET){

//这是外部RST管脚复位

}

RCC\_ClearFlag();//清除RCC中复位标志

//虽然RTC模块不需要重新配置，且掉电后依靠后备电池依然运行

//但是每次上电后，还是要使能RTCCLK

RCC\_RTCCLKCmd(ENABLE);//使能RTCCLK

RTC\_WaitForSynchro();//等待RTC时钟与APB1时钟同步

//当不使用RTC秒中断，可以屏蔽下面2条

NVIC\_InitStructure.NVIC\_IRQChannel = RTC\_IRQn;

NVIC\_InitStructure.NVIC\_IRQChannelPreemptionPriority=3 ;//抢占优先级3

NVIC\_InitStructure.NVIC\_IRQChannelSubPriority = 2; //子优先级3

NVIC\_InitStructure.NVIC\_IRQChannelCmd = ENABLE; //IRQ通道使能

NVIC\_Init(&NVIC\_InitStructure); //根据指定的参数初始化VIC寄存器

RTC\_ITConfig(RTC\_IT\_SEC, ENABLE);//使能秒中断

RTC\_WaitForLastTask();//等待操作完成

}

十五、