## 新建工程

Printf，scanf重定向

int fputc(int c, FILE \* stream)

{

USART1->DR=c;

while(!(USART1->SR & 1<<7)){}

return c;

}

int fgetc(FILE \* stream)

{

while(!(USART1->SR & 1<<5)){}

return USART1->DR;

}

## 2、LED控制、蜂鸣器

## 3、按键、封装按键检测函数(消抖)

## 4、串口、数据发送和接收

串口是什么？ 是一个通信协议。数字通信协议。

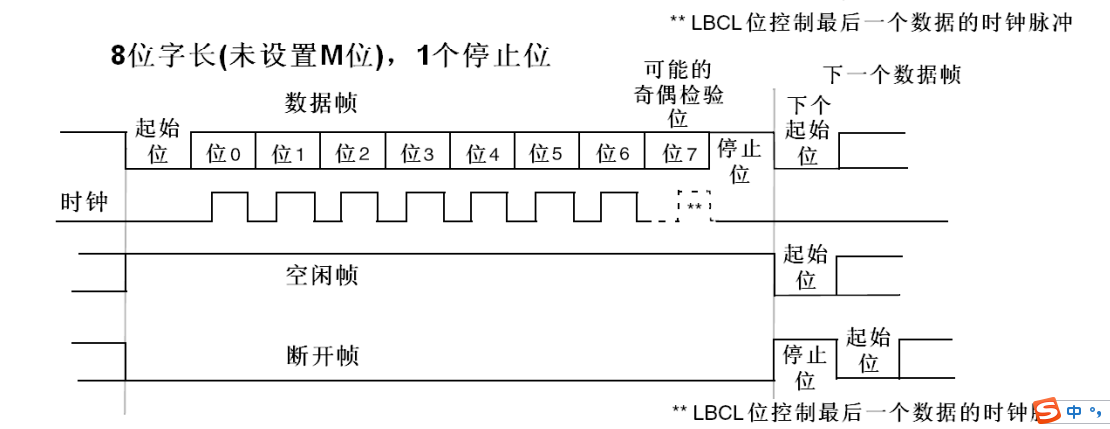
协议: 规定的一种数据传输方法。

串口协议是标准协议，几乎所有设备都支持---最普通的协议。

小板子: CH340-USB信号与串口信号互相转换。

每个单片机都支持串口: **区别就是寄存器名称，配置方法不一样**。

**串口的协议**是一样。



串口的标准配置:

1. **波特率**(不同设备之间通信，波特率得一样)。

只有USART1使用PCLK2(最高72MHz)。其它USART使用PCLK1(最高36MHz)

USART波特比率（usart->BRR）设置,波特比率=Fpclk（usart1为72Mhz，其余最大为36Mhz）/波特率。不用继续除以16，是因为作为小数赋值给BRR时，整数部分要<<4，正好相当于\*16，而小数位也要\*16才能给BRR寄存器低四位赋值。

波特率=Fpclk/（16\*USARTDIV），USARTDIV为BRR高12位，低4位换算为十进制之和。

2. **数据位(8位)**、有9位的，包含了奇偶校验位

一般校验会自己写(异或校验、和校验、CRC校验)

1. **停止位(1)**、**起始位**、**奇偶校验位(无)**

M3串口外设的专用配置:

1. 开启串口的外设时钟，**开启复位时钟(恢复寄存器的默认值)**

2. 使能接收

3. 使能发送

4. 使能串口（总开关） 使能XXX (XXX表示寄存器名称)

M3的GPIO口外设配置:

1. 开启时钟

2. 配置GPIO口的模式。 (模式是复用)

外设属于独立运行的模块，不需要CPU去干预。

STM32F103C8T6芯片资源：

ARM 32-bit Cortex-M3 Microcontroller, **72MHz, 64kB Flash, 20kB SRAM**

**64kB Flash**:存放代码的(hex文件)。

**20kB SRAM**:代码的运行空间。 比如：静态、全局、局部变量。

SysTick Timer, 2 SPI, 2 I2C, **3 USART。**

## STM32的位带操作

#define BITBAND(addr, bitnum) ((addr & 0xF0000000)+0x2000000+((addr &0xFFFFF)<<5)+(bitnum<<2))

#define MEM\_ADDR(addr) \*((volatile unsigned long \*) (addr))

#define GPIOA\_IDR (0x40010800 + 0x08)

#define GPIOA\_ODR (0x40010800 + 0x0C)

#define GPIOB\_IDR (0x40010C00 + 0x08)

#define GPIOB\_ODR (0x40010C00 + 0x0C)

#define pain(x) MEM\_ADDR(BITBAND(GPIOA\_IDR,x))

#define paout(x) MEM\_ADDR(BITBAND(GPIOA\_ODR,x))

#define pbin(x) MEM\_ADDR(BITBAND(GPIOB\_IDR,x))

#define pbout(x) MEM\_ADDR(BITBAND(GPIOB\_ODR,x))

1. 外部中断: 外部中断(很重要)、事件中断、串口中断、定时器、ADC

//外部中断按键中断的初始化。选择中断线（0~7）0~4有对应的中断线和中断响应函数，5——10公用一个中断线和函数。

{

//1.按键初始化。保证按键能用

key\_init();//按键初始化

//2.配置中断线屏蔽位。

EXTI->IMR |= 1<<x;//开启中断线0的中断请求，

//3.配置中断线选择位。选择上升沿下降沿或者双边沿触发方式。一般按键类的都选择双边沿。如鼠标，按钮，参考9章硬件选择章节

EXTI->RTSR |= 1<<x;//允许中断线x上升沿触发的请求

EXTI->FTSR |= 1<<x;//允许中断线x下降沿触发的请求

//4.开启AFIO使能时钟，并选择输入源。参考手册第八章，AFIO节

RCC->APB2ENR |= 1<<0;//开启AFIO时钟，

if(x<=3)

{

AFIO->EXTICR[0] &=~(0XF<<(x\*4));//x为0，1，输入源为PA0,PA1,

switch(x) //5.配置NVIC中断通道，使能够正确响应。

## 定时器:

（定时器定时）1，初始化定时器打开时钟、复位、CR、，设置预分频psc、设置重装载值arr，设置中断；2，计数值清零cnt，开始计时

（定时间输出PWM波）：1，初始化定时器；2，设置输出通道CCMR(x)、设置模式、设置电平有效值、开启输出；3，设置输出比较预装载值CCR(x)；4，使能定时器

8、RTC实时时钟: 配置代码不重要(了解流程)、功能、**时间转换函数。**

9、滴答定时器:

/\*

函数功能:滴答定时器作为外部计算器的初始化函数。不需要配置中断优先级。定时器本身属于内核。

\*/

void SysTickInit(void)

{SysTick->CTRL &= ~(1<<2);//选择作为外部时钟源

SysTick->LOAD = 500\*9000;//9000为1ms，500ms中断一次

SysTick->VAL = 0;//计数值清零

//SysTick->CTRL |= 1<<1;//开启中断

SysTick->CTRL |= 1<<0;//开启滴答定时器}

/\*函数功能:滴答定时器的中断服务函数\*/

void SysTick\_Handler(void)

{LED2=!LED2;}

10、独立看门狗、窗口看门狗:

/\*

函数功能:独立看门狗初始化

函数名称:IWDG\_Init

\*/

void IWDG\_Init(u16 psc,u16 arr)

{

IWDG->KR = 0x5555;//允许对PR，RLR进行写操作。

IWDG->PR = psc;

IWDG->RLR = arr;

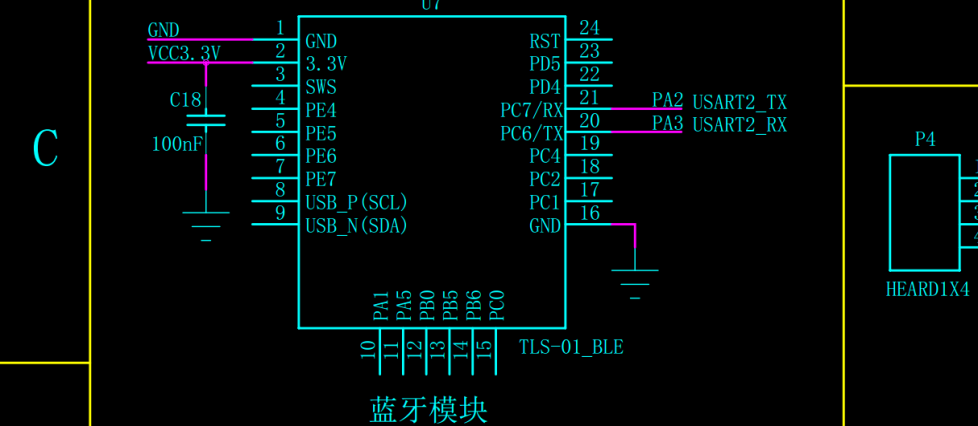
IWDG->KR = 0xAAAA;//刷入重装载值

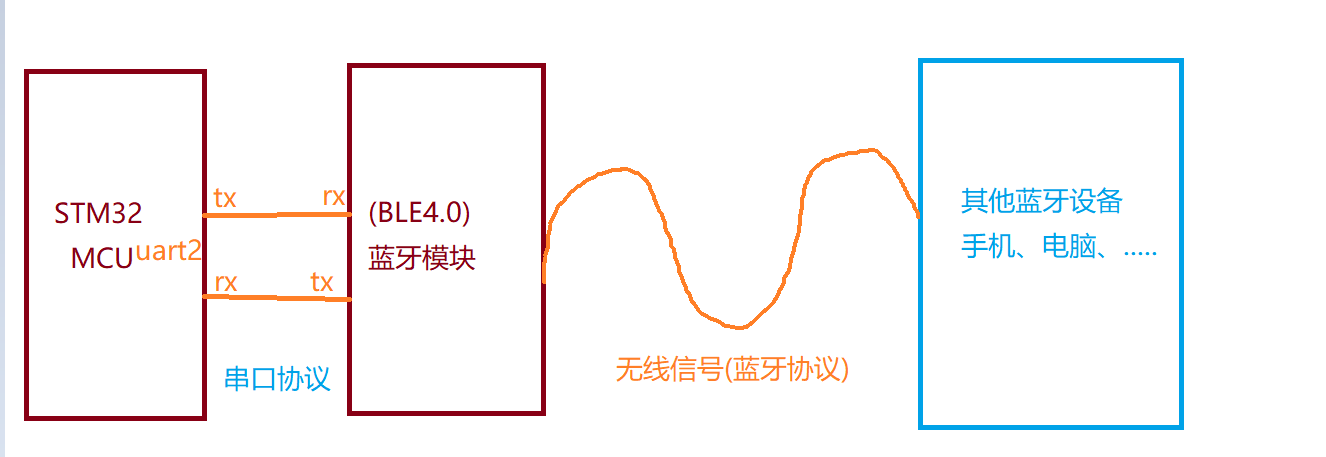
IWDG->KR = 0xCCCC;//启动看门狗

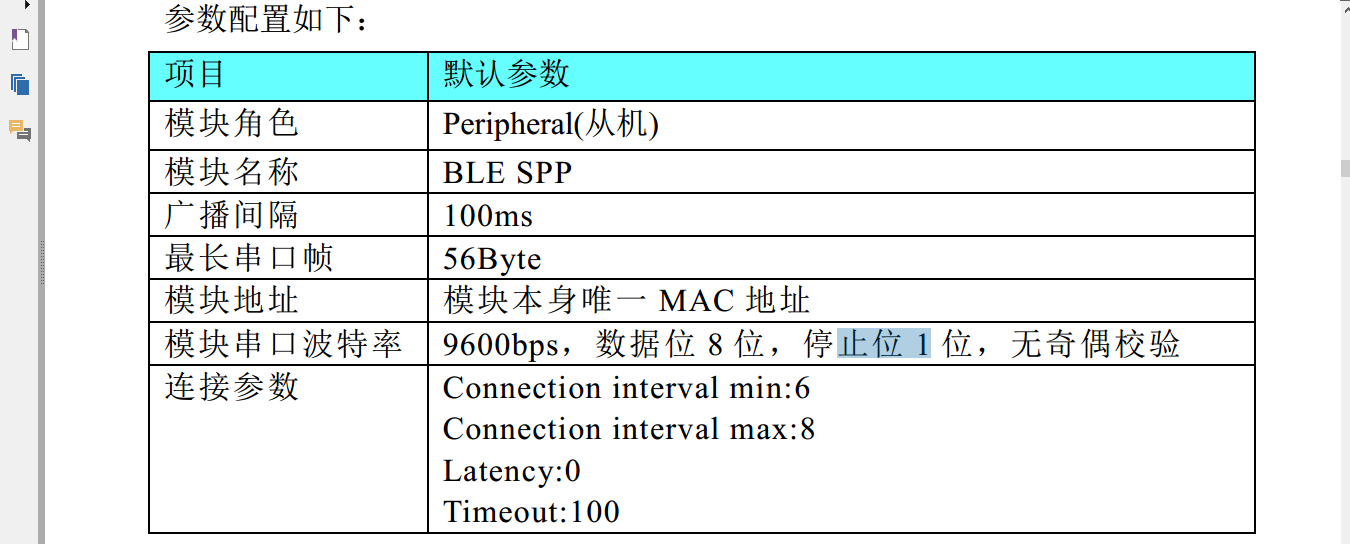
}。

11、蓝牙: 学会**BLE**蓝牙标准函数封装、蓝牙的数据收发(与手机APP)。

# 任务1: 学习串口BLE4.0蓝牙的使用







蓝牙模块有一套指令集: **AT指令集。 指令格式: AT+xxx**

AT指令有两种格式:

1. 获取数据: AT+xxx

2. 设置数据: AT+xxx=xxx

12、WIFI: 学会TCP网络编程(**TCP服务器、客户端**)、学会服务器与客户端的测试方法。

注意：调试助手服务器.ESP8266要先连接你局域网的WiFi才可以连接你的服//

//务器的，连接上WiFi之后，确认你的服务器的ip和端口号是正确的就可以进行连接的了 ,还要关闭防火墙(也就是关闭防火墙的还要保证

学会STA模式、AP模式区别。

学会两种模式的标准函数封装。

两种模式: AP+TCP服务器、STA+TCP客户端。

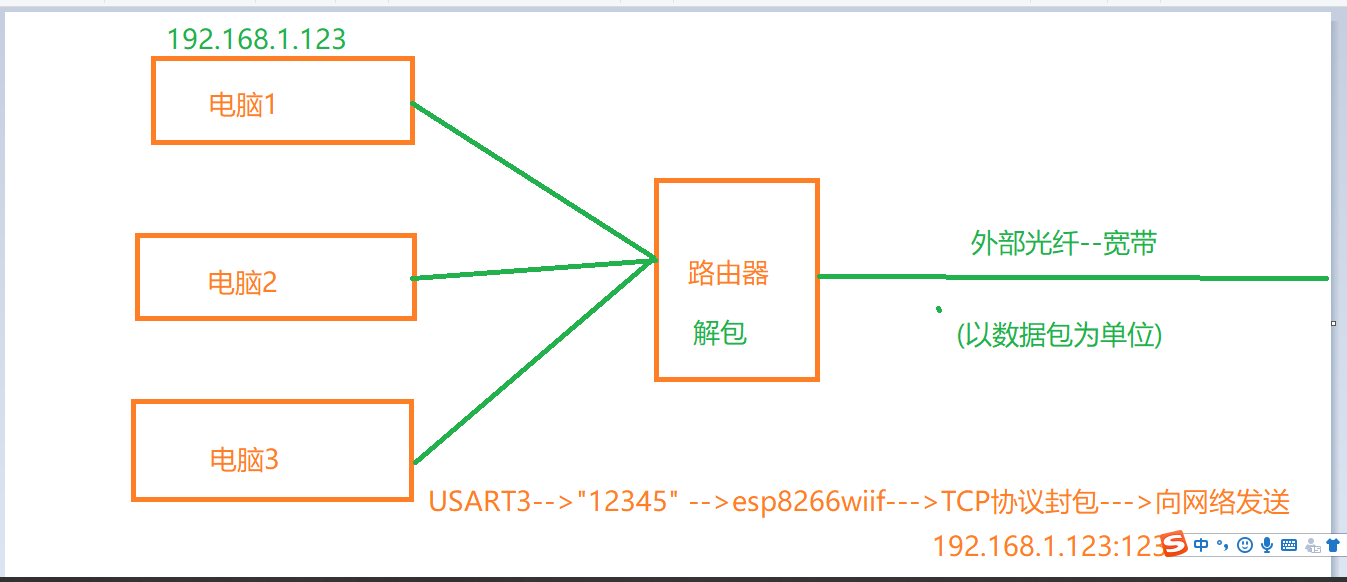
后续的模块: GSM模块、GPS模块。

* WIFI有两种模式:

1. AP模式: 路由模式，将自身当做热点，供其他设备连接
2. STA模式: 从模式，可以去连接其他热点

* WIFI有些必要参数需要设置:

1. AP模式下: 创建的热点名称、加密方式、设置热点密码
2. STA模式下: 设置要连接的热点(WIFI)名称，设置热点密码



* 网络编程的知识

1. 底层网络协议:  **TCP协议**、UDP协议

上层协议: **HTTP**、FTP、RTSP……

1. TCP协议特性: 点对点协议，保证数据的准确性。

UDP协议特性: 广播协议，相对TCP而言无法保证数据准确性。

1. **IP地址**: 一个**局域网**内，每个设备的IP地址都是唯一的。
2. **端口号**: 在一个设备上，每个应用的端口号是唯一的。

* TCP网络编程

1. 在TCP网络通信模型(C/S)里，需要有服务器和客户端两个设备。

电脑上的QQ属于客户端，QQ的服务器在腾讯公司。

**服务器先存在，然后客户端再去连接服务器**。

客户端与客户端之间是无法通信的。

1. 服务器支持多个客户端连接。

* ESP8266 WIFI 4种组合:

1. AP模式+TCP服务器模式
2. AP模式+TCP客户端模式
3. STA模式+TCP服务器模式

模块的基地址:192.168.4.1

AP模式+TCP服务器模式:

AT 测试指令

ATE0 关闭回显

AT+CWMODE=2 设置当前WIFI的模式为AP模式

AT+RST 复位模块,设置的模式必须复位之后才会生效

ATE0 关闭回显

AT+CWSAP="Cortex\_M3","12345678",1,4 设置创建的热点信息

AT+CIPMUX=1 开启多连接,一般在服务器模式下才可开启

AT+CIPSERVER=1,8089 设置端口号(0~65535)

AT+CIFSR 查询当前IP地址

AT+CIPSEND=0,12 //发送数据的指令 0表示给谁发，12表示发送的长度

4. STA模式+TCP客户端模式

AT 测试指令

ATE0 关闭回显

AT+CWMODE=1 设置当前WIFI的模式为STA模式

AT+RST 复位模块,设置的模式必须复位之后才会生效

ATE0 关闭回显

AT+CWJAP="HUAWEI-C4VTTJ","1234567890" 设置将要连接的热点信息

AT+CIFSR 查询当前IP地址

AT+CIPMUX=0 设置单连接

AT+CIPMODE=1 设置透传模式

AT+CIPSTART="TCP","192.168.1.125",8899 手机 server 端的 ip192.168.3.8 和端口 8899

AT+CIPSEND 开始发送数据 (接下来就进入到数据发送模式)

13、DS18B20数字温度测量模块:

(1) 学习看时序图

(2) 单总线时序(收和发)

(3) DS1820组网(多个DS18B20)

14、ADC模拟信号与数字信号的转换:

(1) 注入通道 (2) 规则通道

作业: 根据外部光源强度、调整LED灯的亮度。

void adc1\_init(void)//ADC初始化

{

//打开时钟

RCC->APB2ENR|=1<<3;//GPIOB时钟使能

RCC->APB2ENR|=1<<9;//ADC1时钟使能

//ADC复位

RCC->APB2RSTR|=1<<9;//ADC1复位使能

RCC->APB2RSTR&=~(1<<9);//ADC1复位使能

//GPIOB配置

GPIOB->CRL&=0xFFFFFFF0;//模拟输入模式

//ADC工作频率

RCC->CFGR&=~(0x3<<14);//清空ADC分频

RCC->CFGR|=0x2<<14;//分频系数为6 12MHZ

//配置ADC

//ADC1->CR2|=1<<20;//使用外部事件,规则通道

//ADC1->CR2|=0x7<<17;//选择外部事件为 SWSTART，规则

ADC1->CR2|=1<<15;//使用外部事件,注入通道

ADC1->CR2|=0x7<<12;//选择外部事件为 SWSTART，注入

ADC1->SMPR2|=0x7<<24;//设置通道8的采样时间为 239.5周期

ADC1->CR2|=1<<0;//开启AD转换

ADC1->CR2|=1<<3;//复位校准

while(ADC1->CR2 & 1<<3){}

ADC1->CR2|=1<<2;//校准

while(ADC1->CR2 & 1<<2){}

}

/\*

函数名称:ADC1\_CHx

函数功能:采集规则转换通道内的数据.

\*/

u16 ADC1\_CHx(u8 ch)

{

ADC1->SQR3&=0xFFFFFFE0;

ADC1->SQR3|=ch<<0;

ADC1->CR2|=1<<22;//开启规则通道转换

while(!(ADC1->SR & 1<<1)){}//等待AD转换完成

return ADC1->DR;

}

/\*

函数名称:ADC1\_CHx

函数功能:采集注入转换通道内的数据.

\*/

u16 ADC1\_JCHx(u8 ch)

{

ADC1->JSQR &= 0xFFCFFFE0;

ADC1->JSQR |= 0x00300000;

ADC1->JSQR|=ch<<0;

/\*或者\*/

// ADC1->JSQR &= 0xfff07fff;

// ADC1->JSQR |= ch<<15;

ADC1->CR2|=1<<21;//开启规则通道转换

while(!(ADC1->SR & 1<<1)){}//等待AD转换完成

return ADC1->JDR1;

}

## 15、OLED显示屏

# 任务1: 了解显示屏的特性

分辨率是128\*64:

显示屏类似一个一个方格子，屏幕的横向有128个格子，纵向有64个格子。

现在学习的OLED显示屏是单色屏，只能控制显示屏格子亮和灭。

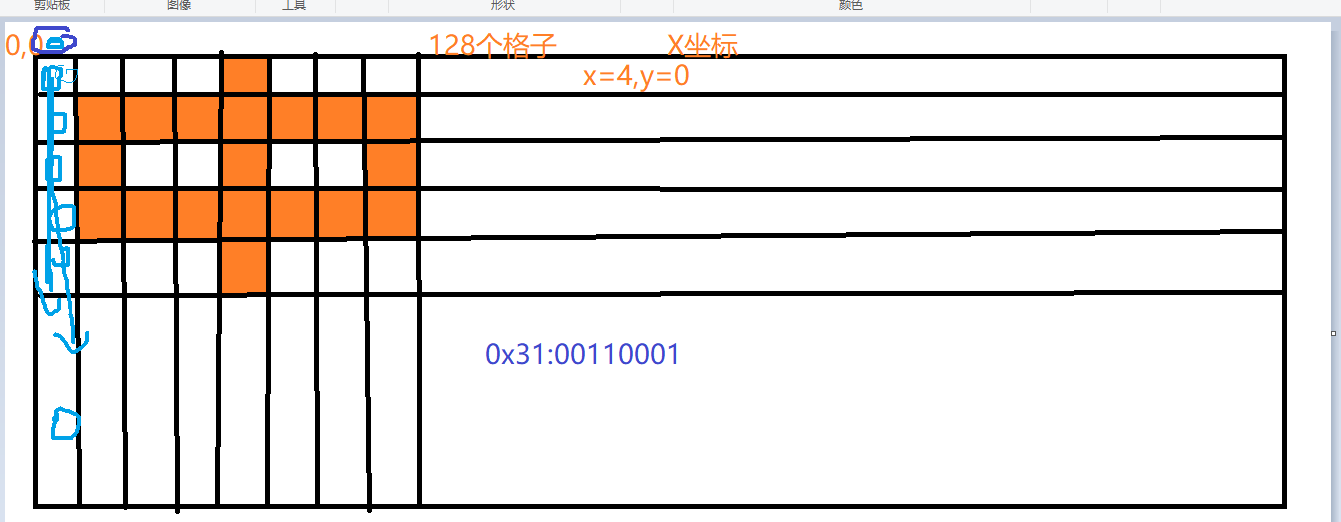
在程序里就是0表示灭、1表示亮。

OLED屏幕通信的时序，使用的是SPI时序，每次发送数据8位(1个字节)。

**OLED屏幕每一个格子表示一个位，每一次操作OLED显示屏最小单位就是8个格子**。

OLED屏幕格子单位划分: 横坐标128个位，纵向是8页(纵向操作的最小单位是页)。

1页==8个位

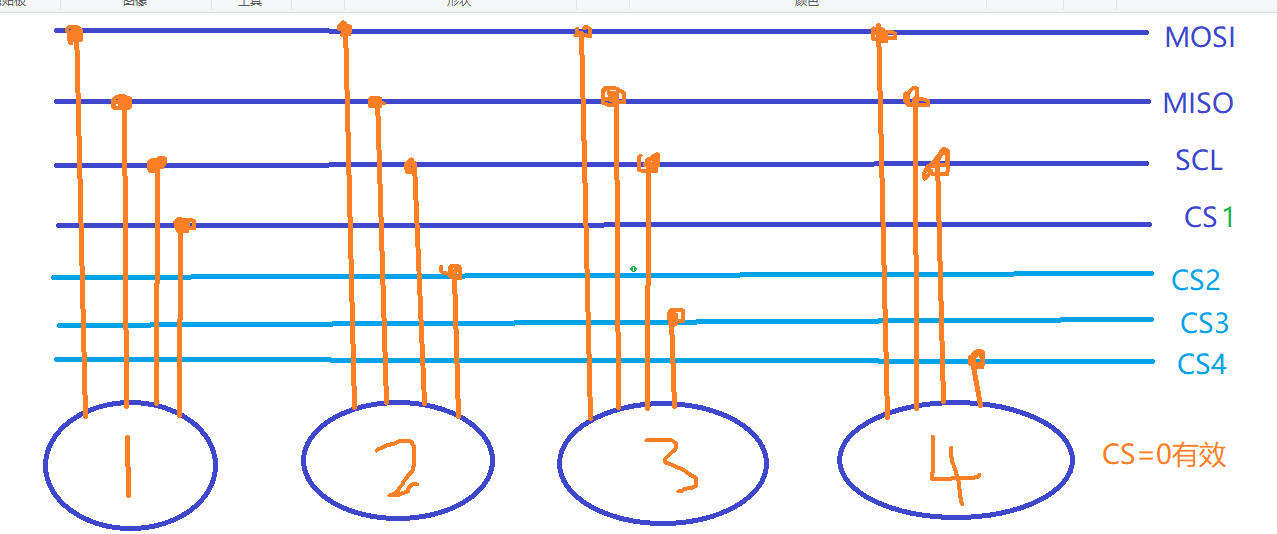


# 任务2: SPI时序

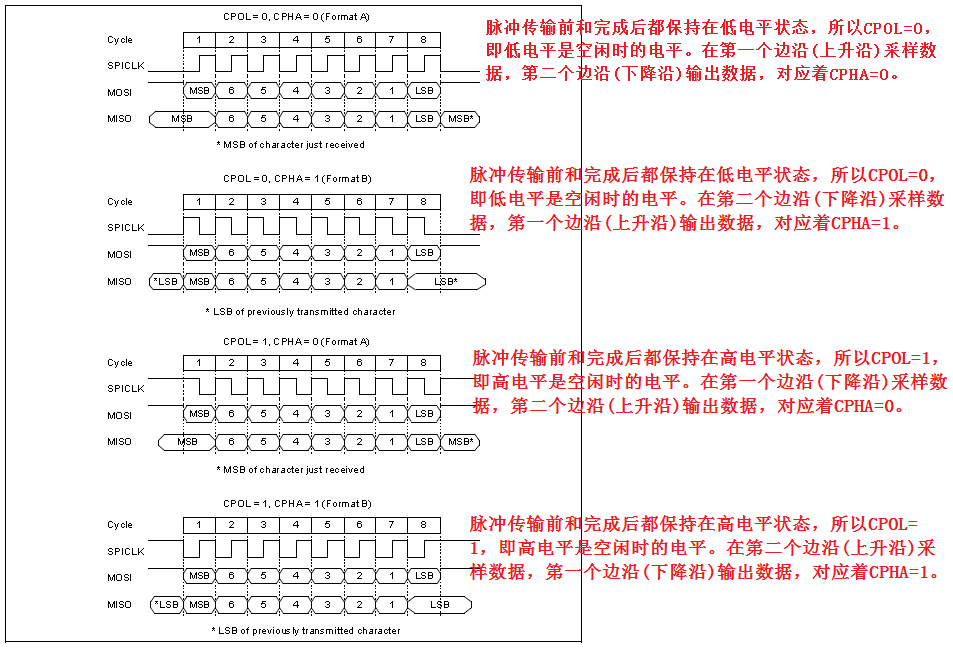
标准**SPI总线**的IO引脚:

1. SCL : 时钟线(用来同步数据的，数据什么时候读或者写都是由时钟线操作)
2. MOSI : 主出从入: 主机输出、从机读取
3. MISO : 主入从出: 主机读取、从机输出
4. CS : 片选（区分设备，每个设备都有一根独立的片选）

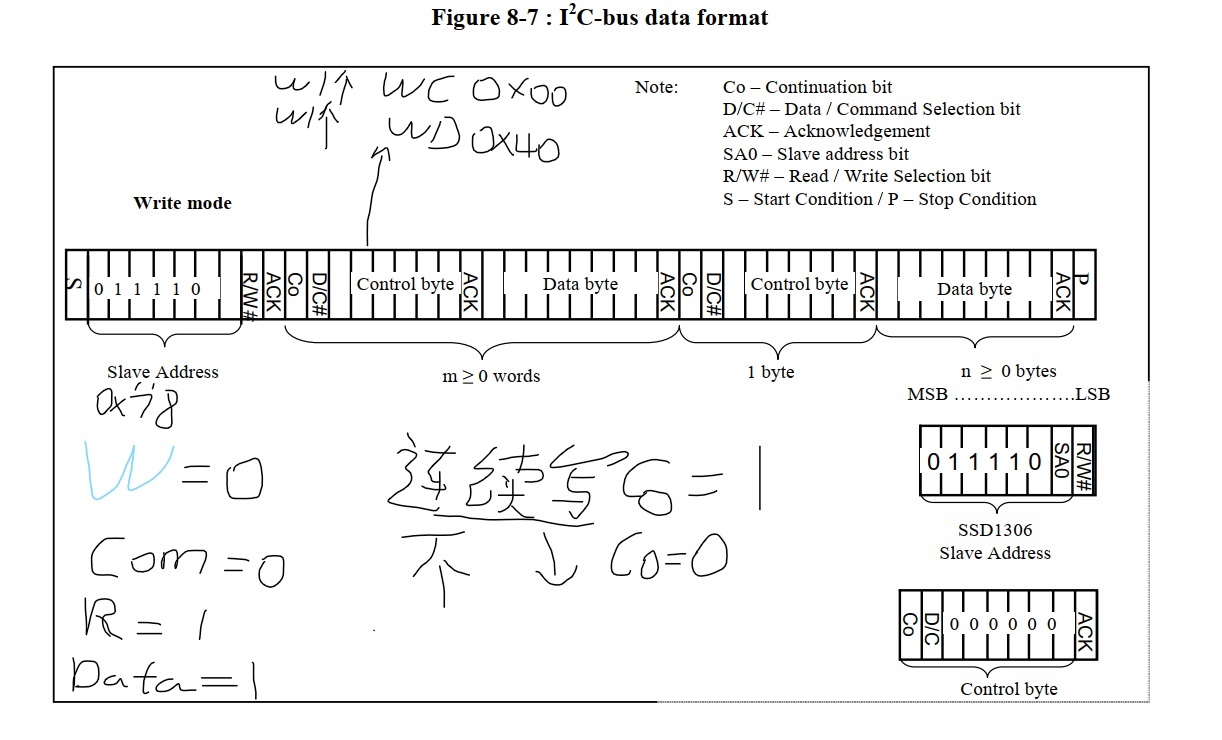
串口: 全双工 SPI: 全双工 DS18B20单总线: 单工



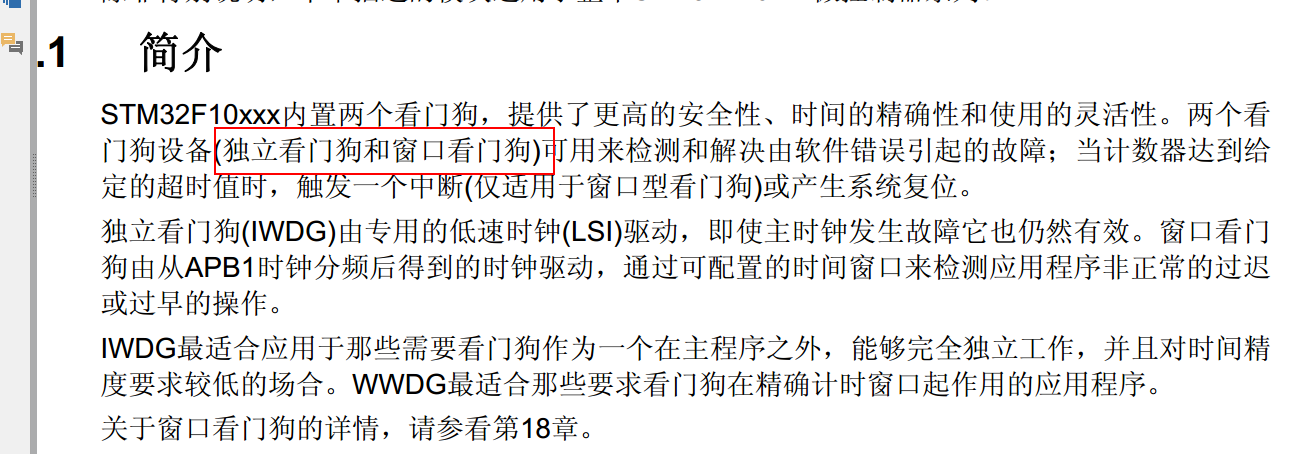
SPI有4种时序：



IIC-OLED



## 16、看门狗

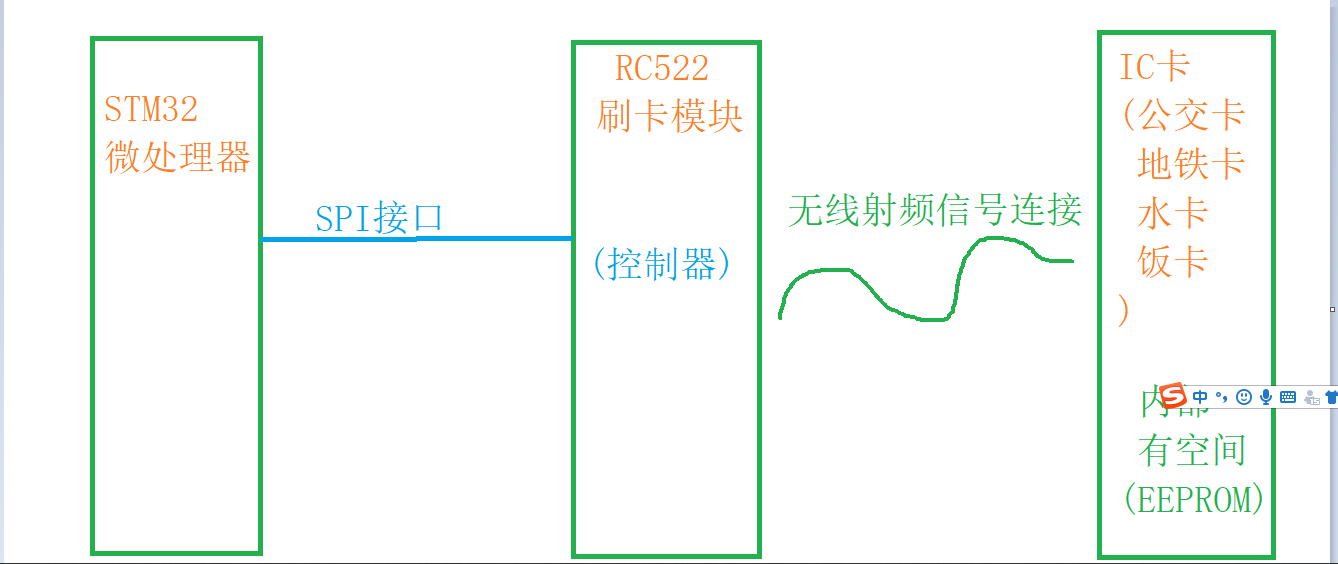


独立看门狗: 只要在计数器到达0之前都可以喂狗。

窗口看门狗: 喂狗时间必须在一个范围(**窗口**)以内

## 17、RCC522

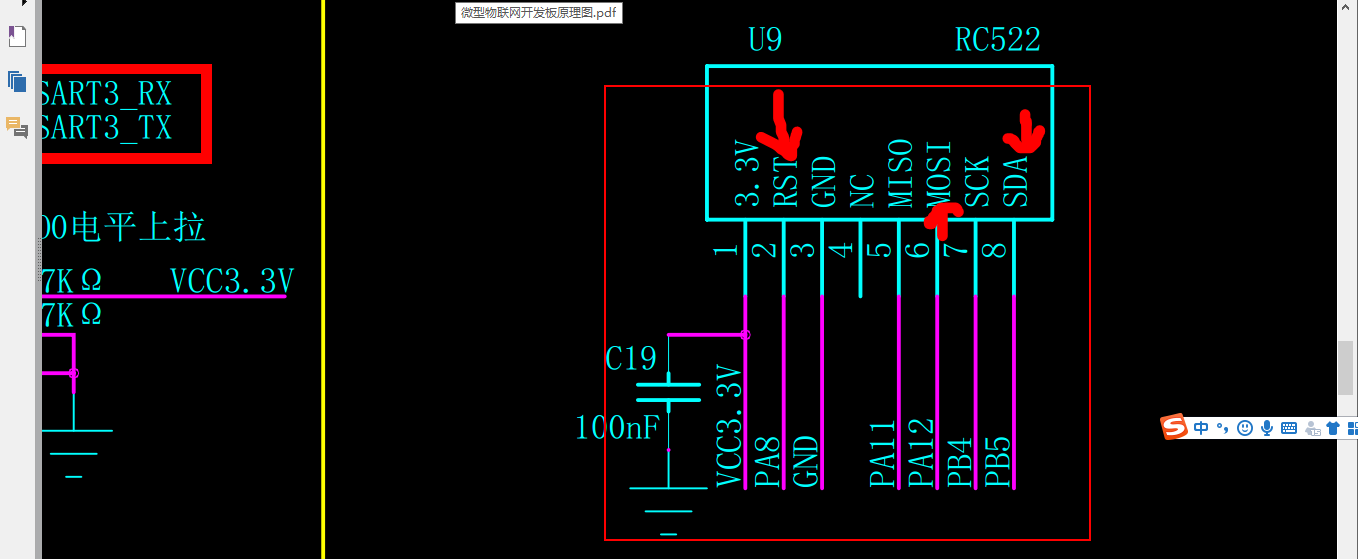
# 任务1: RC522射频刷卡模块



1. 学习RC522控制器的驱动代码
2. 学习IC卡的操作流程，内部的空间布局

S50卡、S70卡。

正常情况下: 白卡密码---**0xFFFFFFFFFFFF 0x000000000000**



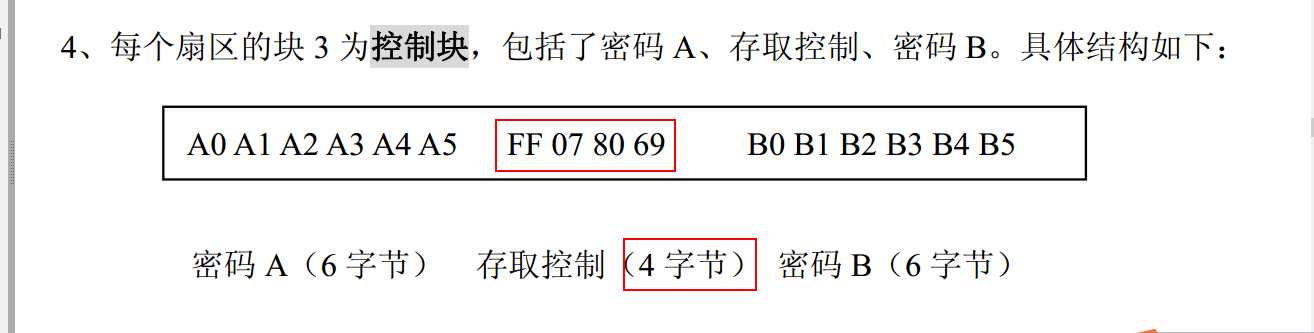
1. RCC522 初始化  
   2. 寻卡  
   3. 防冲撞: 读出卡号  
   4. 选卡: 选中卡片之后才可以去验证卡密码  
   5. 验证卡片: 密码验证成功之后才可以对卡进行读写操作  
   6. 读卡片/写卡片  
   7. 结束卡片操作

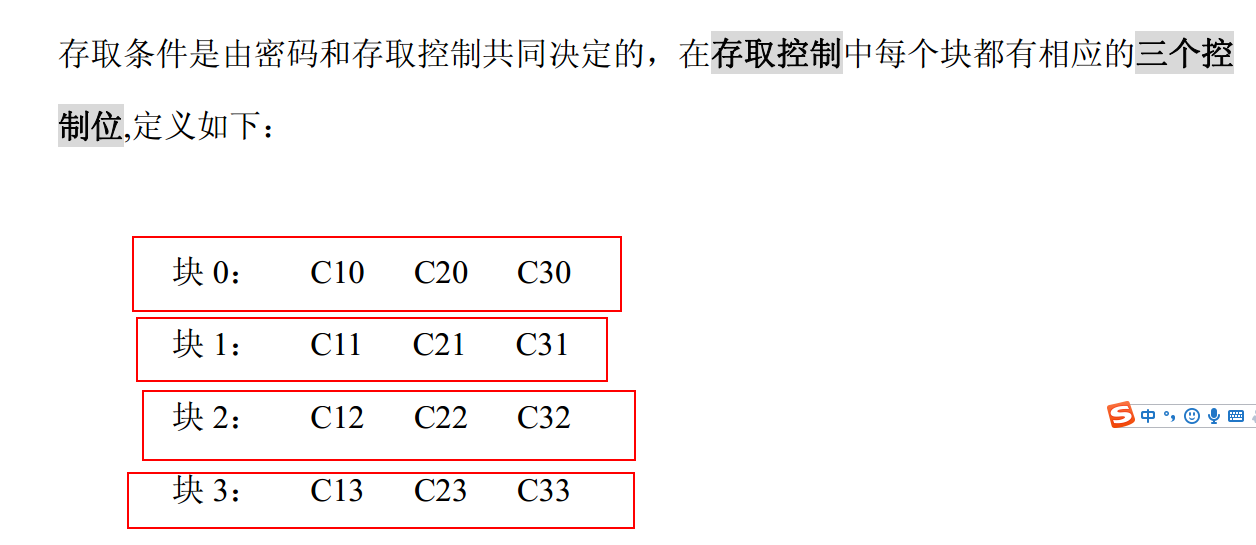
* 对IC卡控制: 写数据、读数据、修改密码。

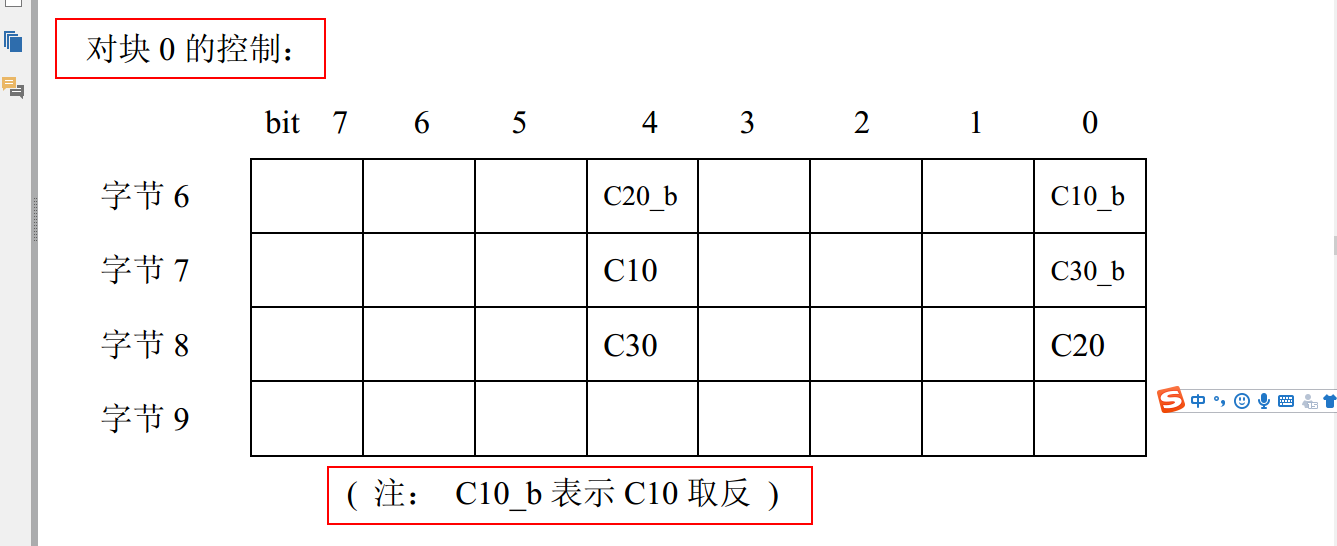
1. 寻卡: 可以寻找天线范围内所有卡，寻卡成功后可以得到卡类型。
2. 防冲撞: 选中寻到的卡片，读出该卡的卡号。
3. 选定卡片: 根据上一步读出的卡号，进行选定卡片，在选定卡片之后才可以进行验证密码操作。
4. 验证密码: 验证密码可以选择A或者B密码进行验证，验证成功之后才可以进行 卡的读写操作。
5. 读写IC卡: 可以连续的对当前扇区之内的**数据块和控制块**进行操作(每个扇区一共是4个块)。

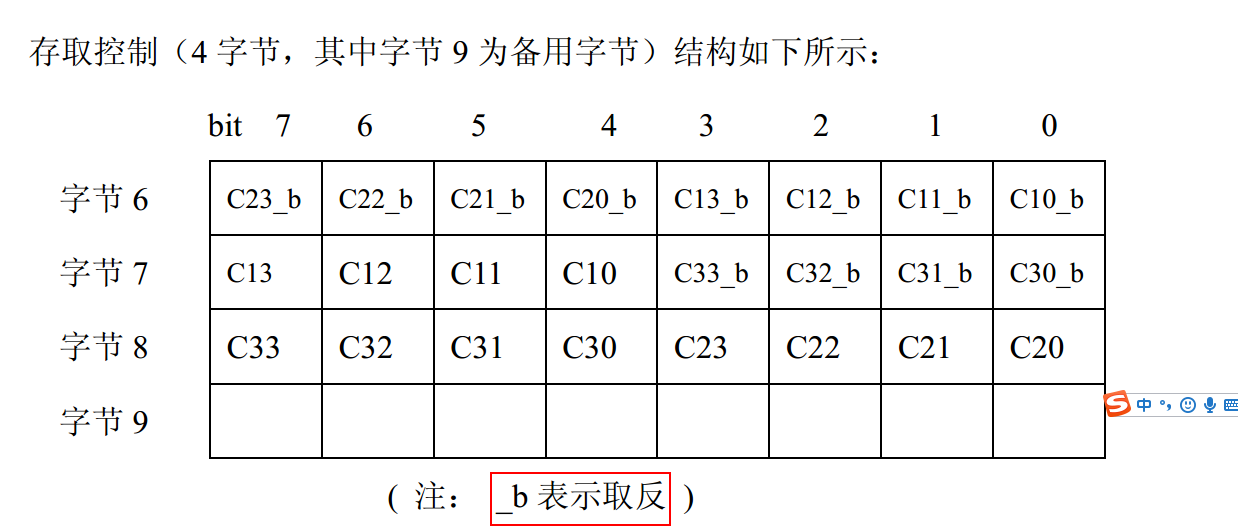
(对当前扇区内的块连续操作时，不需要再次验证密码)

# 任务2: IC卡密码修改与控制块的权限组合









读出白卡控制块的值:

/\*0 0 0 0 0 0 **FF 07 80 69** FF FF FF FF FF FF \*/

组合的权限:



DMA

1）从外设到存储器

以ADC采集为例，DMA外部寄存器地址对应ADC数据寄存器地址，DMA存储器地址是我们自定义的变量的地址。方向设置为源地址。

2）存储器到外设

存储器到外设传输以串口向电脑端发送为例，DMA 外设寄存器的地址对应的就是串口数据寄存器的地址，DMA 存储器的地址就是我们自定义的变量（相当于一个缓冲区，用来存储通过串口发送到电脑的数据）的地址。方向我们设置外设为目标地址。

3）存储器到存储器

当我们使用从存储器到存储器传输时，以内部 FLASH 向内部 SRAM 复制数据为例。

DMA外设寄存器的地址对应的就是内部 FLASH（我们这里把内部 FALSH 当作一个外设来看）的地址，DMA 存储器的地址就是我们自定义的变量（相当于一个缓冲区，用来存储来自内部 FLASH 的数据）的地址。方向我们设置外设（即内部 FLASH）为源地址。跟上面两个不一样的是，这里需要把 DMA\_CCR 位 14：MEM2MEM：存储器到存储器模式配置为 1，启动 M2M 模式。

bxCAN

CAN\_BTR实际上是

设置波特率。tq=tCAN.

波特率=1\*tq+tBS2+tBS1也就是

波特率=[（TS2+1）+(TS1+1)+1]\*tq.

tq=(BRP+1)\*1

如：

tBS1=8,tBS2=7,tBRP=3

波特率=36Mhz/[(9+8+1)\*4]=500bps