



**（1）GPIO\_Mode\_AIN模拟输入**

**（2）GPIO\_Mode\_IN\_FLOATING浮空输入**

**（3）GPIO\_Mode\_IPD下拉输入**

**（4）GPIO\_Mode\_IPU上拉输入**

**（5）GPIO\_Mode\_Out\_OD开漏输出**

**（6）GPIO\_Mode\_Out\_PP推挽输出**

**（7）GPIO\_Mode\_AF\_OD复用开漏输出**

**（8）GPIO\_Mode\_AF\_PP复用推挽输出**

**Pwm控制引脚**

**中断线1-4, 单独 5-9一个, 10-15一个**

抢占优先级1个中3个 响应优先级 0-3 越小越优先。

**RCC\_ APB2PeriphClockCmd (RCC\_ APB2Periph AFIO, ENABLE) ;**

**GPIO\_ PinRemapConfig (GPIO\_ PartialRemap1\_TIM2，ENABLE) ;**

**GPIO\_ PinRemapConfig (GPIO\_ Remap\_ SWJ\_ JTAGDisable, ENABLE) ;**

**重新定义引脚解除复用**

Pa0 = tim\_ch1

**与运算赋值（&=）**两位同时为1，结果才为1，否则为0谁小取谁

## **异或运算赋值（^=）**应位为“异”（值不同），则该位结果为1，否则为0。

## **或运算赋值（|=）**两位只要有一个为1，其值为1，其它都为0。谁大去谁

**Exit外部中断**

void CountSensor\_Init(void)

{

**RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_GPIOB, ENABLE);**

**使能gpiob**

**RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_AFIO, ENABLE);**

**使能 afio**

**GPIO\_InitTypeDef GPIO\_InitStructure;**

**GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_IPU;**

**GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_14;**

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

**GPIO\_Init(GPIOB, &GPIO\_InitStructure);**

**把gpio 14与中断相连**

**GPIO\_EXTILineConfig(GPIO\_PortSourceGPIOB, GPIO\_PinSource14);**

EXTI\_InitTypeDef EXTI\_InitStructure;

EXTI\_InitStructure.EXTI\_Line = **EXTI\_Line14**;

EXTI\_InitStructure.EXTI\_LineCmd = ENABLE;

**EXTI\_InitStructure.EXTI\_Mode = EXTI\_Mode\_Interrupt;**

***中断模式：中断模式***

**EXTI\_InitStructure.EXTI\_Trigger = EXTI\_Trigger\_Falling;**

***触发方式：上升沿触发***

EXTI\_Init(&EXTI\_InitStructure);

**NVIC\_PriorityGroupConfig(NVIC\_PriorityGroup\_2);**

**选用中断线2**

NVIC\_InitTypeDef NVIC\_InitStructure;

**NVIC\_InitStructure.NVIC\_IRQChannel = EXTI15\_10\_IRQn;**

**蓝色地方必须相同**

NVIC\_InitStructure.NVIC\_IRQChannelCmd = ENABLE;

NVIC\_InitStructure.NVIC\_IRQChannelPreemptionPriority = 1;

NVIC\_InitStructure.NVIC\_IRQChannelSubPriority = 1;

NVIC\_Init(&NVIC\_InitStructure);

}

**void EXTI15\_10\_IRQHandler(void)**

**蓝色地方必须相同**

{//因为EXTI10--15都能进这个中断函数，所以先判断是不是EXTI14进的中断

if (****EXTI\_GetFlagStatus**(EXTI\_Line14**) == SET)

{

/\*如果出现数据乱跳的现象，可再次判断引脚电平，以避免抖动\*/

if (GPIO\_ReadInputDataBit(GPIOB, GPIO\_Pin\_14) == 0)

{

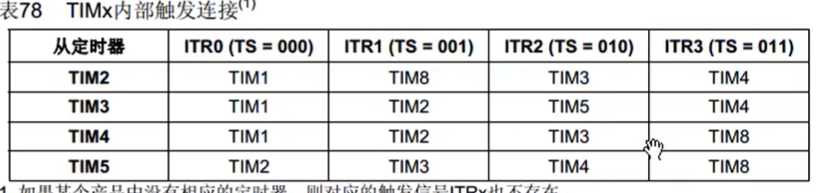
CountSensor\_Count ++;

}

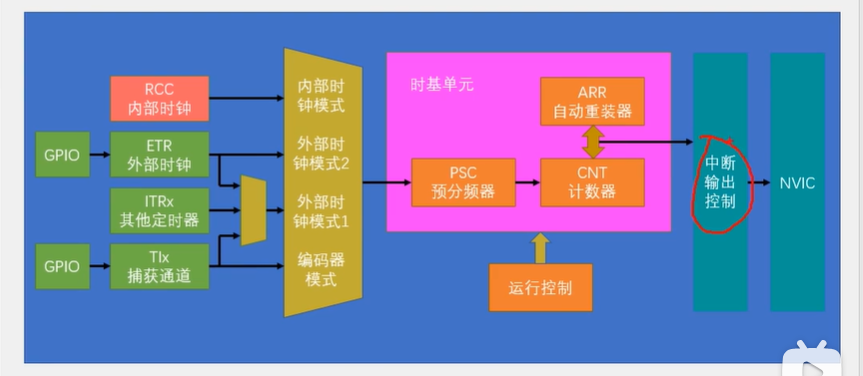
EXTI\_ClearITPendingBit(EXTI\_Line14);

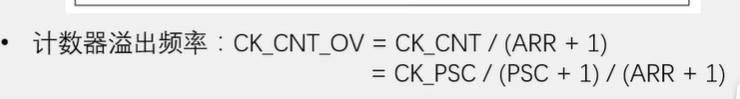
}

}



**定时器**





**时间就是溢出频率倒数**

**定时器**

**void timer\_config(void){**

**RCC\_APB1PeriphClockCmd(RCC\_APB1Periph\_TIM2, ENABLE);**

**1.选择内部时钟**

**1.TIM\_InternalClockConfig(TIM2);**

**2.选择外部时钟**

**2.TIM\_ETRClockMode2Config(TIM2,TIM\_ExtTRGPSC\_OFF, TIM\_ExtTRGPolarity\_NonInverted, 0x00);**

**RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_GPIOA, ENABLE);**

**GPIO\_InitTypeDef GPIO\_StructInit;**

**GPIO\_StructInit.GPIO\_Mode = GPIO\_Mode\_IPU;**

**GPIO\_StructInit.GPIO\_Pin =GPIO\_Pin\_0;**

**GPIO\_StructInit.GPIO\_Speed = GPIO\_Speed\_50MHz;**

**GPIO\_Init(GPIOA, &GPIO\_StructInit);**

**TIM\_TimeBaseInitTypeDef TIM\_TimeBaseInitStructure;**

**TIM\_TimeBaseInitStructure.TIM\_ClockDivision = TIM\_CKD\_DIV1;**

**TIM\_TimeBaseInitStructure.TIM\_CounterMode = TIM\_CounterMode\_Up;**

**TIM\_TimeBaseInitStructure.TIM\_Period = 10000 - 1;**

**TIM\_TimeBaseInitStructure.TIM\_Prescaler = 7200 - 1;**

**TIM\_TimeBaseInitStructure.TIM\_RepetitionCounter = 0;**

**TIM\_TimeBaseInit(TIM2, &TIM\_TimeBaseInitStruct);**

**TIM\_ClearFlag(TIM2, TIM\_FLAG\_Update);**

**启用定时器定时器**

**TIM\_ITConfig(TIM2, TIM\_IT\_Update, ENABLE);**

**NVIC\_PriorityGroupConfig(NVIC\_PriorityGroup\_1);**

**NVIC\_InitTypeDef NVIC\_InitStruct;**

**NVIC\_InitStruct.NVIC\_IRQChanne = TIM2\_IRQn;**

**NVIC\_InitStruct.NVIC\_IRQChannelCmd = ENABLE;**

**NVIC\_InitStruct.NVIC\_IRQChannelPreemptionPriority = 1;**

**NVIC\_InitStruct.NVIC\_IRQChannelSubPriority = 2;**

**NVIC\_Init(&NVIC\_InitStruct);**

**启动定时器外围**

**TIM\_Cmd(TIM2, ENABLE);**

**}**

**void TIM2\_IRQHandler(void){**

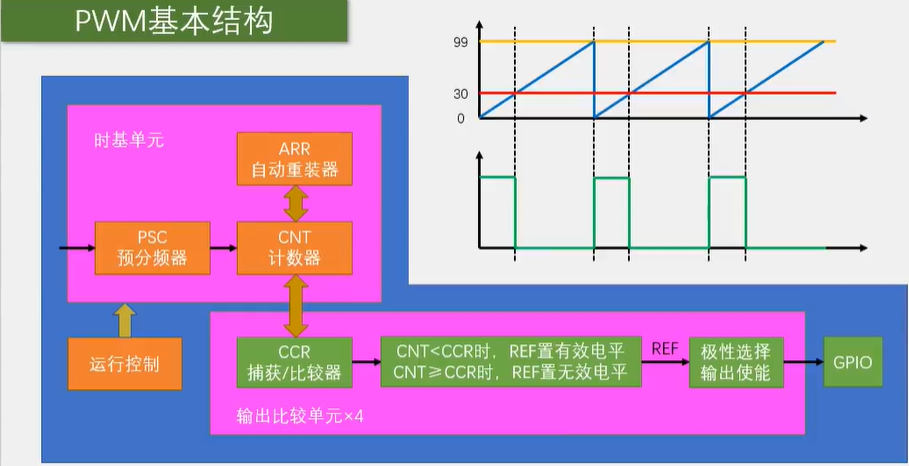
**if (TIM\_GetFlagStatus(TIM2, TIM\_FLAG\_Update) == SET){**

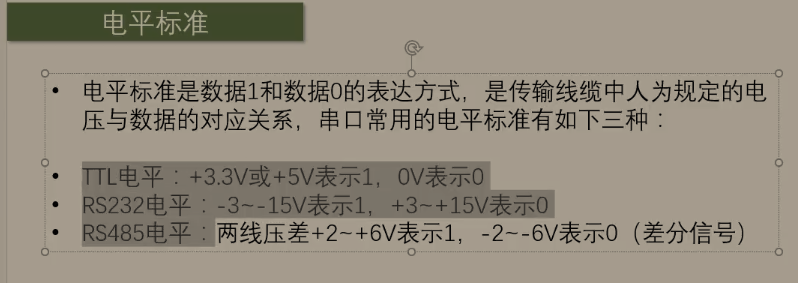
**TIM\_ClearITPendingBit(TIM2, TIM\_IT\_Update);**

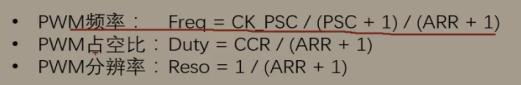
**}**

**}**

**Pwm**





**void Pwm\_Init(){**

**RCC\_APB1PeriphClockCmd(RCC\_APB1Periph\_TIM2, ENABLE);**

**RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_GPIOA, ENABLE);**

**GPIO\_InitTypeDef GPIO\_InitStructure;**

**GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_AF\_PP;**

**GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_0;**

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

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

**TIM\_ETRClockMode2Config(TIM2, TIM\_ExtTRGPSC\_OFF, TIM\_ExtTRGPolarity\_NonInverted, 0x0f);**

**RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_GPIOA, ENABLE);**

**GPIO\_InitTypeDef GPIO\_StructInit;**

**GPIO\_StructInit.GPIO\_Mode = GPIO\_Mode\_IPU;**

**GPIO\_StructInit.GPIO\_Pin = GPIO\_Pin\_0;**

**GPIO\_StructInit.GPIO\_Speed = GPIO\_Speed\_50MHz;**

**GPIO\_Init(GPIOA, &GPIO\_StructInit);**

**TIM\_InternalClockConfig(TIM2);**

**TIM\_TimeBaseInitTypeDef TIM\_TimeBaseInitStructure;**

**TIM\_TimeBaseInitStructure.TIM\_ClockDivision = TIM\_CKD\_DIV1;**

**TIM\_TimeBaseInitStructure.TIM\_CounterMode = TIM\_CounterMode\_Up;**

**TIM\_TimeBaseInitStructure.TIM\_Period = 10000 - 1; //ARR**

**TIM\_TimeBaseInitStructure.TIM\_Prescaler = 7200 - 1; //PSC**

**TIM\_TimeBaseInitStructure.TIM\_RepetitionCounter = 0;**

****TIM\_TimeBaseInit(TIM\_TypeDef\* TIMx, TIM\_TimeBaseInitTypeDef\* TIM\_TimeBaseInitStruct);****

**TIM\_OCInitTypeDef TIM\_OCStructInit;**

**TIM\_OCStructInit(&TIM\_OCInitStruct);**

**TIM\_OCStructInit.TIM\_OCMode = TIM\_OCMode\_PWM1;**

**TIM\_OCStructInit.TIM\_OCPolarity = TIM\_OCPolarity\_High;**

**TIM\_OCStructInit.TIM\_OutputState = TIM\_OutputState\_Enable;**

**TIM\_OCStructInit.TIM\_Pulse= ; //CCR**

**TIM\_OC1Init(TIM2, &TIM\_OCInitStruct);**

**//TIM\_OC1PreloadConfig(TIM2,TIM\_OCPreload\_Enable);**

//TIM\_ITConfig(TIM2,TIM\_IT\_Update,ENABLE);

**IM\_Cmd(TIM2, ENABLE);**

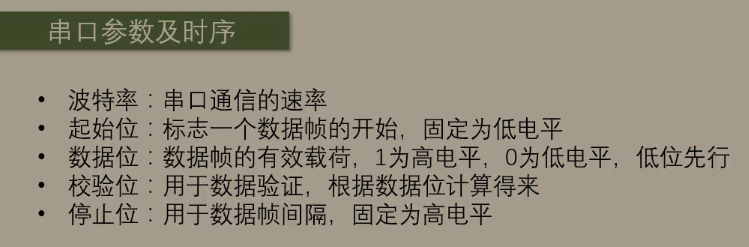
**}**

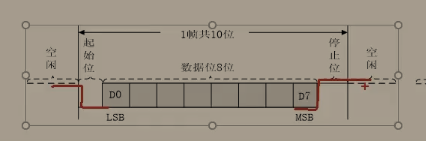
**void PwmSetCompare(uint16\_t compare){**

**TIM\_SetCompare1(TIM2, compare);**

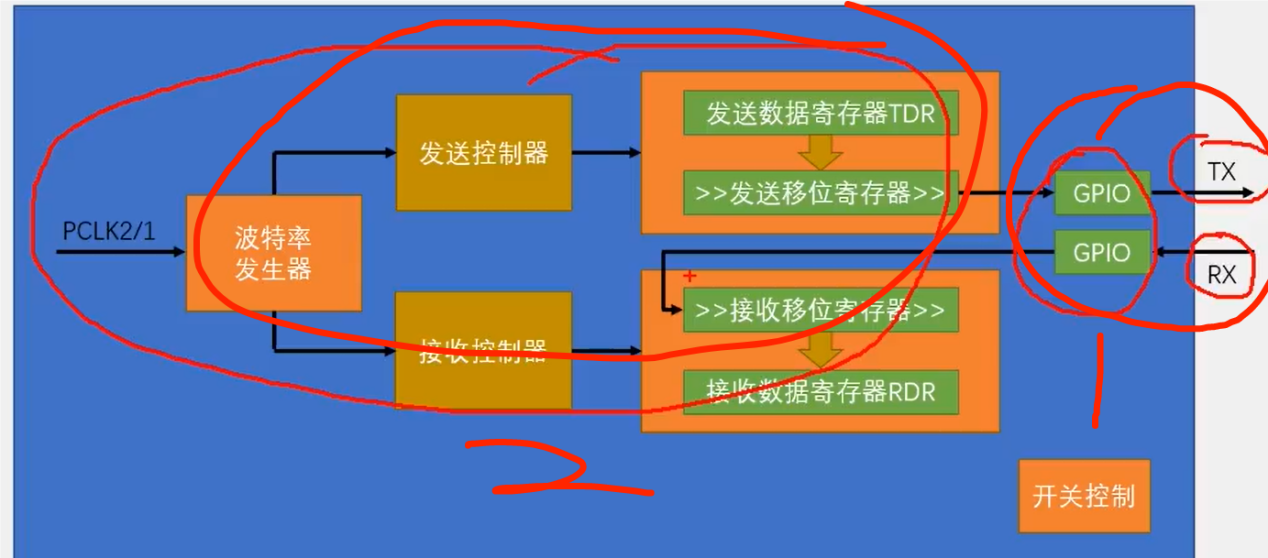
**}**

**Uart**





**起始位为下降沿，停止位为上升沿。**



**void usart\_init(void){**

**RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_GPIOA, ENABLE);**

**RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_USART1, ENABLE);**

**GPIO\_InitTypeDef GPIO\_StructInit;**

**GPIO\_StructInit.GPIO\_Mode = GPIO\_Mode\_AF\_PP;**

**GPIO\_StructInit.GPIO\_Pin = GPIO\_Pin\_9;**

**GPIO\_StructInit.GPIO\_Speed = GPIO\_Speed\_50MHz;**

**GPIO\_Init(GPIOA, &GPIO\_StructInit);**

**USART\_InitTypeDef USART\_StructInit;**

**USART\_StructInit.USART\_BaudRate = 9600;**

**USART\_StructInit.USART\_HardwareFlowControl = USART\_HardwareFlowControl\_None;**

**USART\_StructInit.USART\_Mode = USART\_Mode\_Tx;**

**USART\_StructInit.USART\_Parity = USART\_Parity\_No;**

**USART\_StructInit.USART\_StopBits = USART\_StopBits\_1;**

**停止位1格**

**USART\_StructInit.USART\_WordLength = USART\_WordLength\_8b;**

**8位没有校验位**

**USART\_Init(USART1, &USART\_StructInit);**

**USART\_Cmd(USART1, ENABLE);**

**}**

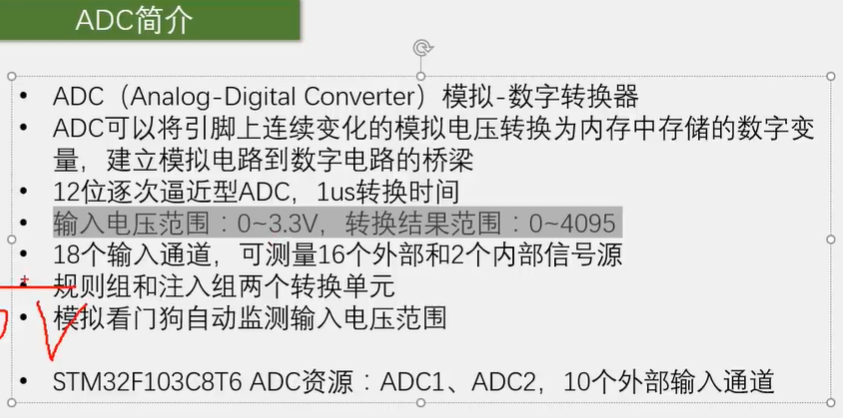
**void send\_byte(uint8\_t byte){**

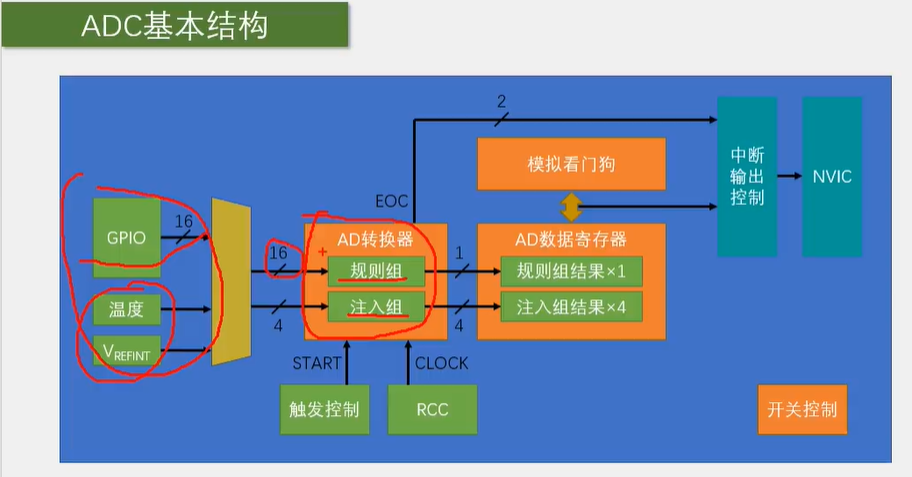
**USART\_SendData(USART1, byte);**

**while(USART\_GetFlagStatus(USART1,USART\_FLAG\_TXE) ==RESET);**

**}**

**ADC**





**多通道adc**

**void AD\_Init(void)**

**{**

**RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_ADC1, ENABLE);**

**RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_GPIOA, ENABLE);**

**RCC\_ADCCLKConfig(RCC\_PCLK2\_Div6);**

**GPIO\_InitTypeDef GPIO\_InitStructure;**

**GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_AIN;**

**GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_0 | GPIO\_Pin\_1 | GPIO\_Pin\_2 | GPIO\_Pin\_3;**

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

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

**ADC\_InitTypeDef ADC\_InitStructure;**

**ADC\_InitStructure.ADC\_Mode = ADC\_Mode\_Independent;**

**ADC\_InitStructure.ADC\_DataAlign = ADC\_DataAlign\_Right;**

**ADC\_InitStructure.ADC\_ExternalTrigConv = ADC\_ExternalTrigConv\_None;**

**ADC\_InitStructure.ADC\_ContinuousConvMode = DISABLE;**

**ADC\_InitStructure.ADC\_ScanConvMode = DISABLE;**

**ADC\_InitStructure.ADC\_NbrOfChannel = 1;**

**ADC\_Init(ADC1, &ADC\_InitStructure);**

**ADC\_Cmd(ADC1, ENABLE);**

**ADC\_ResetCalibration(ADC1);**

**while (ADC\_GetResetCalibrationStatus(ADC1) == SET);**

**ADC\_StartCalibration(ADC1);**

**while (ADC\_GetCalibrationStatus(ADC1) == SET);**

**}**

**uint16\_t AD\_GetValue(uint8\_t ADC\_Channel)**

**{**

**ADC\_RegularChannelConfig(ADC1, ADC\_Channel, 1, ADC\_SampleTime\_55Cycles5);**

**ADC\_SoftwareStartConvCmd(ADC1, ENABLE);**

**while (ADC\_GetFlagStatus(ADC1, ADC\_FLAG\_EOC) == RESET);**

**return ADC\_GetConversionValue(ADC1);**

**}**

**在主函数中定义**

**AD1，AD2，AD3**

**AD1 = AD\_GetValue(ADC\_Channel\_1);**

**AD2 = AD\_GetValue(ADC\_Channel\_2);**

**AD3 = AD\_GetValue(ADC\_Channel\_3);**

**开启三个gpio通道**

**ADC\_InitStructure.ADC\_Mode = ADC\_Mode\_Independent;**

**单通道扫描**

**ADC\_InitStructure.ADC\_DataAlign = ADC\_DataAlign\_Right**

**右对齐**

**ADC\_InitStructure.ADC\_ContinuousConvMode = DISABLE;**

**连续转换模式一直检测**

**ADC\_InitStructure.ADC\_NbrOfChannel = 1**

**指定通道数量**

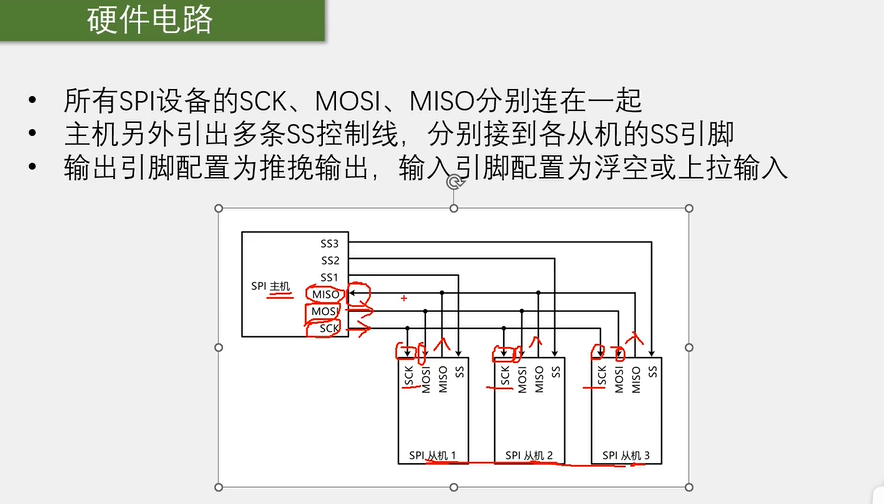
**Spi**

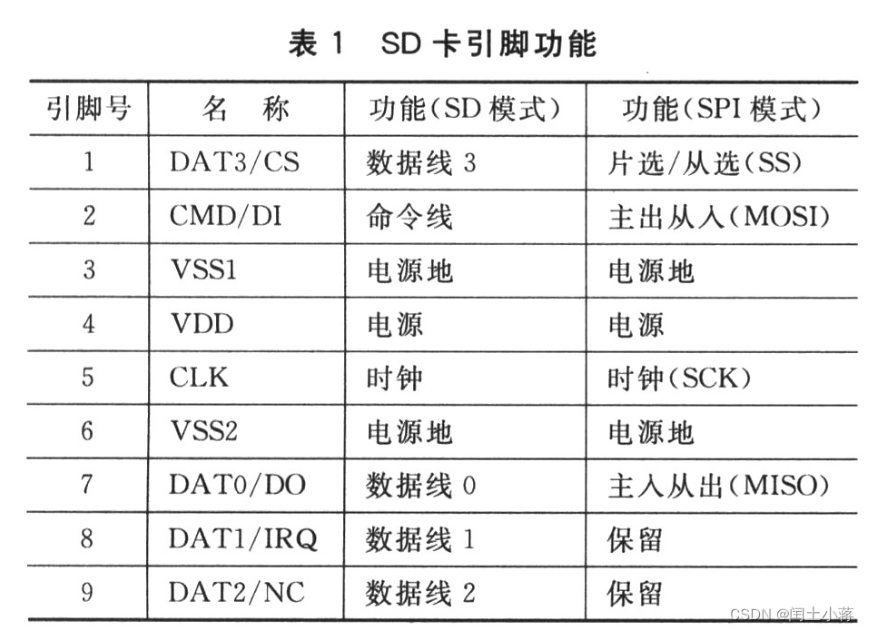


**&可以查询**

**Const a = 0x00 锁定了这个值**







**要在STM32中使用链表来存储USART接收到的数据，可以按照以下步骤进行：**

**1. 首先，定义一个结构体来表示链表节点，每个节点包含数据和指向下一个节点的指针。例如：**

**```c**

**typedef struct Node {**

**uint8\_t data; // 数据**

**struct Node \*next; // 指向下一个节点的指针**

**} Node;**

**```**

**2. 在全局范围内定义一个指向链表头节点的指针和一个指向链表尾节点的指针。例如：**

**```c**

**Node \*head = NULL; // 指向链表头节点的指针**

**Node \*tail = NULL; // 指向链表尾节点的指针**

**```**

**3. 在USART接收中断处理函数中，将接收到的数据插入链表。例如：**

**```c**

**void USART1\_IRQHandler(void) {**

**if (USART\_GetITStatus(USART1, USART\_IT\_RXNE) != RESET) {**

**// 读取接收到的数据**

**uint8\_t rxData = USART\_ReceiveData(USART1);**

**// 创建一个新的节点**

**Node \*newNode = (Node \*)malloc(sizeof(Node));**

**newNode->data = rxData;**

**newNode->next = NULL;**

**// 如果链表为空，则将新节点设置为头节点**

**if (head == NULL) {**

**head = newNode;**

**tail = newNode;**

**} else {**

**// 否则，将新节点插入到链表的尾部**

**tail->next = newNode;**

**tail = newNode;**

**}**

**}**

**}**

**```**

**4. 使用链表中的数据进行后续处理，例如打印链表中的数据：**

**```c**

**void printLinkedList() {**

**Node \*current = head;**

**while (current != NULL) {**

**printf("%d ", current->data);**

**current = current->next;**

**}**

**printf("\n");**

**}**

**```**

**通过以上步骤，你可以在STM32中使用链表来存储USART接收到的数据。注意，在实际应用中，还需要考虑内存管理、错误处理等方面的问题。**