妈的

复用功能重映射示例

将USART2的Tx和Rx从默认引脚PA2和PA3重新映射到PD5和PD6

RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_GPIOD,ENABLE); //①

RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_AFIO,ENABLE); //②

//以下为③

GPIO\_InitStructure.GPIO\_Pin=GPIO\_Pin\_5;

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

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

GPIO\_InitStructure.GPIO\_Pin=GPIO\_Pin\_6;

GPIO\_InitStructure.GPIO\_Mode=GPIO\_Mode\_IN\_FLOATING;

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

RCC\_APB1PeriphClockCmd(RCC\_APB1Periph\_USART2,ENABLE); //④

//⑤进行USART2的I/O引脚重映射

GPIO\_PinRemapConfig(GPIO\_Remap\_USART2,ENABLE);

①使能被重新映射的I/O引脚时钟

②使能APB2总线上的AFIO时钟

③按复用功能的方式配置I/O引脚

④使能被重新映射的外设时钟

⑤对外设进行I/O引脚重映射

①使能APB2总线上该引脚所属GPIO端口时钟 ②通过GPIO\_InitTypeDef结构体变量配置GPIO引脚 ③调用函数初始化 ④操作该引脚

时钟

RCC\_APB1PeriphClockCmd(RCC\_APB1Periph\_TIM3,ENABLE); ///使能 TIM3 时钟

使能 TIM3 的更新中断，格式为：

TIM\_ITConfig(TIM3,TIM\_IT\_Update, ENABLE );

TIM\_Cmd(TIM3, ENABLE); //使能 TIMx 外设

void TIM3\_Init(u16 arr, u16 psc)

{

TIM\_TimeBaseInitTypeDef TIM\_TimeBaseInitStructure;

NVIC\_InitTypeDef NVIC\_InitStructure;

RCC\_APB1PeriphClockCmd(RCC\_APB1Periph\_TIM3,ENABLE);

//①使能 TIM3 时钟

TIM\_TimeBaseInitStructure.TIM\_Period = arr;

//自动重装载值

TIM\_TimeBaseInitStructure.TIM\_Prescaler=psc;

//定时器分频

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

//向上计数模式

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

TIM\_TimeBaseInit(TIM3,&TIM\_TimeBaseInitStructure);

// ②初始化定时器

TIM3 TIM\_ITConfig(TIM3,TIM\_IT\_Update,ENABLE);

//③允许定时器 3 更新中断

NVIC\_InitStructure.NVIC\_IRQChannel=TIM3\_IRQn;

//定时器 3 中断

NVIC\_InitStructure.NVIC\_IRQChannelPreemptionPriority=0x01;

//抢占优先级 1

NVIC\_InitStructure.NVIC\_IRQChannelSubPriority=0x03;

//子优先级 3

NVIC\_InitStructure.NVIC\_IRQChannelCmd=ENABLE;

NVIC\_Init(&NVIC\_InitStructure);

// ④初始化 NVIC

TIM\_Cmd(TIM3,ENABLE);//⑤使能定时器 3

}

void TIM3\_IRQHandler(void)

{

if(TIM\_GetITStatus(TIM3,TIM\_IT\_Update)!=RESET;) //TIM3 溢出中断

{

LED1=!LED1;

}

TIM\_ClearITPendingBit(TIM3,TIM\_IT\_Update); //清除中断标志位

}

delay\_init(168); //初始化延时函数

NVIC\_PriorityGroupConfig(NVIC\_PriorityGroup\_2);//设置系统中断优先级分组 2

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

//使能 USART1 时钟

GPIO 时钟使能，因为使用的是串口 1，串口 1 对应着芯片引脚 PA9、PA10，所以只需

要使能 GPIOA 时钟即可。

RCC\_AHB1PeriphClockCmd(RCC\_AHB1Periph\_GPIOA,ENABLE); //使能 GPIOA 时钟

GPIO 端口模式设置：PA9 和 PA10 要设置为复用功能。

GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_9 ;//GPIOA9

GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_AF;//复用功能

GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_50MHz; //速度 50MHz

GPIO\_InitStructure.GPIO\_OType = GPIO\_OType\_PP; //推挽复用输出

GPIO\_Init(GPIOA,&GPIO\_InitStructure); //初始化 PA9

串口参数初始化：设置波特率、字长、奇偶校验等参数

串口初始化是调用函数 USART\_Init 来实现的，具体设置方法如下：

USART\_InitStructure.USART\_BaudRate = bound;

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;

USART\_InitStructure.USART\_Mode = USART\_Mode\_Rx | USART\_Mode\_Tx;

//收发模式

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

（5）使能串口

使能串口调用函数 USART\_Cmd 来实现，使能串口 1 方法如下：

USART\_Cmd(USART1, ENABLE); //使能串口 1

开启中断并且初始化 NVIC，使能相应中断

要开启串口中断必须要配置 NVIC 中断优先级分组，通过调用函数 NVIC\_Init 来设置。NVIC\_InitStructure.NVIC\_IRQChannel = USART1\_IRQn;

NVIC\_InitStructure.NVIC\_IRQChannelPreemptionPriority=1;//抢占优先级 1

NVIC\_InitStructure.NVIC\_IRQChannelSubPriority =1;

//响应优先级 1

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、

同时，我们还需要使能相应中断,使能串口中断的函数是：

void USART\_ITConfig(USART\_TypeDef\* USARTx, uint16\_t USART\_IT,

FunctionalState NewState)

while(USART\_GetFlagStatus(USART1,USART\_FLAG\_TXE)!=SET);

串口设置的一般步骤可以总结为如下几个步骤：

（1）串口时钟使能，GPIO 时钟使能。

（2）设置引脚复用器映射：调用 GPIO\_PinAFConfig 函数。

（3）GPIO 初始化设置：要设置模式为复用功能。

（4）串口参数初始化：设置波特率，字长，奇偶校验等参数。

（5）开启中断并且初始化 NVIC，使能中断（如果需要开启中断才需要这个步骤）。

（6）使能串口。

（7）编写中断处理函数：函数名格式为 USARTx\_IRQHandler(x 对应串口号)。

（1）串口时钟和 GPIO 时钟使能。RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_USART1,ENABLE);

//使能 USART1 时钟

RCC\_AHB1PeriphClockCmd(RCC\_AHB1Periph\_GPIOA,ENABLE); //使能 GPIOA 时钟

（2）设置引脚复用器映射

调用函数为：

GPIO\_PinAFConfig(GPIOA,GPIO\_PinSource9,GPIO\_AF\_USART1);

串口 1 使用 PA9 做为 TX，PA10 为 RX，所以要把 PA9 和 PA10 都映射到串口 1，

需要调用两次函数 GPIO\_PinAFConfig(XXX)。

（3）GPIO 端口模式设置：PA9 和 PA10 要设置为复用功能（GPIO\_Mode\_AF）。

GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_9 ;//GPIOA9

GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_AF;//复用功能

GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_50MHz; //速度 50MHz

GPIO\_InitStructure.GPIO\_OType = GPIO\_OType\_PP; //推挽复用输出

GPIO\_Init(GPIOA,&GPIO\_InitStructure); //初始化 PA9

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

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

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

GPIO\_InitStructure.GPIO\_PuPd = GPIO\_PuPd\_UP; //上拉输入

GPIO\_Init(GPIOA,&GPIO\_InitStructure); //初始化 PA10

（4）串口参数初始化：设置波特率、字长、奇偶校验等参数

串口初始化是调用函数 USART\_Init 来实现的，具体设置方法如下：

USART\_InitStructure.USART\_BaudRate = bound;

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;

USART\_InitStructure.USART\_Mode = USART\_Mode\_Rx | USART\_Mode\_Tx;

//收发模式

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

（5）使能串口

使能串口调用函数 USART\_Cmd 来实现，使能串口 1 方法如下：

USART\_Cmd(USART1, ENABLE); //使能串口 1

（6）串口数据发送与接收。

STM32F4 的发送与接收都是通过数据寄存器 USART\_DR 来实现的，这是一个双寄存

器，包含了 TDR 和 RDR。当向该寄存器写数据的时候，串口就会自动发送，当收到数据

的时候，也是存在该寄存器内。

STM32 库函数操作 USART\_DR 寄存器发送数据的函数是：

void USART\_SendData(USART\_TypeDef\* USARTx, uint16\_t Data);

通过该函数向串口寄存器 USART\_DR 写入一个数据。

STM32 库函数操作 USART\_DR 寄存器读取串口接收到的数据的函数是：

uint16\_t USART\_ReceiveData(USART\_TypeDef\* USARTx);

通过该函数可以读取串口接受到的数据。

（7）串口状态

串口的状态可以通过状态寄存器 USART\_SR 读取。主要关注两个位，RXNE 和 TXE。

RXNE（读数据寄存器非空），当该位被置 1 的时候，就是提示已经有数据被接收到

了，并且可以读出来了。这时要读取 USART\_DR 送入内存变量保存，通过读 USART\_DR

可以将该位清零，也可以向该位写 0，直接清除。

TXE（发送寄存器空），当 TDR 寄存器的内容已传输到移位寄存器时，该位由硬件

置 1。如果 USART\_CR1 寄存器中 TXEIE 位 = 1，则会生成中断。通过对 USART\_DR

寄存器执行写入操作将该位清零。0：数据未传输到移位寄存器，1：数据传输到移位寄存

器。

（8） 开启中断并且初始化 NVIC，使能相应中断

要开启串口中断必须要配置 NVIC 中断优先级分组，通过调用函数 NVIC\_Init 来设置。NVIC\_InitStructure.NVIC\_IRQChannel = USART1\_IRQn;

NVIC\_InitStructure.NVIC\_IRQChannelPreemptionPriority=1;//抢占优先级 1

NVIC\_InitStructure.NVIC\_IRQChannelSubPriority =1;

//响应优先级 1

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、

同时，我们还需要使能相应中断,使能串口中断的函数是：

void USART\_ITConfig(USART\_TypeDef\* USARTx, uint16\_t USART\_IT,

FunctionalState NewState)

这个函数的第二个入口参数是标示使能串口的类型，也就是使能哪种中断，因为串口

的中断类型有很多种。比如在接收到数据的时候（RXNE 读数据寄存器非空），我们要产

生中断，那么开启中断的方法是：

USART\_ITConfig(USART1, USART\_IT\_RXNE, ENABLE);//开启中断，接收到数据中断

我们在发送数据结束的时候（TC，发送完成）要产生中断，那么方法是：

USART\_ITConfig(USART1，USART\_IT\_TC，ENABLE);

因为实验开启了串口中断，所以我们在系统初始化的时候需要先设置系统的中断优先

级分组，是在 main 函数开头设置的，代码如下：

NVIC\_PriorityGroupConfig(NVIC\_PriorityGroup\_2);//设置系统中断优先级分组 2

设置为 2 位抢占优先级，2 位响应优先级。

（9）获取相应中断状态

当使能了某个中断的时候，当该中断发生时，就会设置状态寄存器中的某个标志位。

经常在中断处理函数中，要判断该中断是哪种中断，使用的函数是：

ITStatus USART\_GetITStatus(USART\_TypeDef\* USARTx, uint16\_t USART\_IT)

比如使能了串口接收完成中断，那么当中断发生了，可以在中断处理函数中调用这个

函数来判断到底是否是串口完成中断，方法是：

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

（10）中断服务函数

串口 1 中断服务函数为：

void USART1\_IRQHandler(void) ;

当发生中断的时候，程序就会执行中断服务函数。然后在中断服务函数中添加相应的

逻辑代码即可。