STM32查询方式的USART串口简单使用

STM32 的库实在强大 ~！函数长的像句子.....

好了开始了：

使用查询方式的USART:

设置时钟：

RCC_APB2Periph_AFIO 功能复用IO时钟
RCC_APB2Periph_GPIOA GPIOA时钟

RCC_APB2Periph_USART1 USART1时钟

你可以用
// 使能串口1，PA，AFIO总线 RCC_APB2PeriphClockCmd (RCC_APB2Periph_GPIOA|RCC_APB2Periph_AFIO|RCC_APB2Periph_USART1 ,ENABLE);

或直接

RCC_APB2PeriphClockCmd（RCC_APB2Periph_ALL,ENABLE); //全部APB2外设时钟开启

注意USART2的你开启为 RCC_APB1PeriphClockCmd(RCC_APB1Periph_USART2,ENABLE);

设置GPIO:

```
GPIO_InitTypeDef GPIO_InitStructure;


GPIO_InitStructure.GPIO_Pin = GPIO_Pin_9;
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP; //推挽输出-TX
GPIO_Init(GPIOA, &GPIO_InitStructure);


GPIO_InitStructure.GPIO_Pin = GPIO_Pin_10;
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN_FLOATING; //浮空输入-RX
GPIO_Init(GPIOA, &GPIO_InitStructure);
```

设置USART:

这里我用的是3.0的库相对于2.0的库来说多了一步，先说2.0

```
USART_InitTypeDef USART_InitStructure;

USART_StructInit(&USART_InitStructure); //装填默认值

USART_Init(USART1, &USART_InitStructure);//根据 USART_InitStruct 中指定的参数初始化外设 USARTx 寄存器
USART_Cmd(USART1, ENABLE); //启用
```

就好了~!

而 3.0 的库需要

```
USART_InitTypeDef USART_InitStructure;

USART_ClockInitTypeDef USART_ClockInitStructure;

USART_StructInit(&USART_InitStructure);
USART_ClockStructInit (&USART_ClockInitStructure);
USART_ClockInit(USART1, &USART_ClockInitStructure);
USART_Init(USART1, &USART_InitStructure);
USART_Cmd(USART1, ENABLE);
```

//只是多分出了 1 个 USART_ClockInitStructure 我也不知为啥要这样？？为了同步异步模式？
USART_InitStruct 中指定的参数内容为：（2.0 的）

```
typedef struct
{
u32 USART_BaudRate; //USART 传输的波特率
u16 USART_WordLength; //一个帧中传输或者接收到的数据位数通常是 8
u16 USART_StopBits; //停止位
u16 USART_Parity; //奇偶校验
u16 USART_HardwareFlowControl; //硬件流控制模式使能还是失能
u16 USART_Mode; //指定了使能或者失能发送和接收模式
u16 USART_Clock; //提示了 USART 时钟使能还是失能
u16 USART_CPOL; //指定了下 SLCK 引脚上时钟输出的极性
u16 USART_CPHA; //指定了下 SLCK 引脚上时钟输出的相位
u16 USART_LastBit;

//来控制是否在同步模式下，在 SCLK 引脚上输出最后发送的那个数据字通常用 USART_LastBit_Disable
} USART_InitTypeDef;
```

我靠~！太细了~！我只知道（9600，8，n，1）这就够了 其他的统统默认~！

USART_StructInit(&USART_InitStructure);
USART_ClockStructInit (&USART_ClockInitStructure); //2.0 不用这句，这样就设好了好了~！自动为您装填了默认参数。默认的参数如下（3.0 的库）：

```
void USART_StructInit(USART_InitTypeDef* USART_InitStruct)
{

USART_InitStruct->USART_BaudRate = 9600;
USART_InitStruct->USART_WordLength = USART_WordLength_8b;
USART_InitStruct->USART_StopBits = USART_StopBits_1;
USART_InitStruct->USART_Parity = USART_Parity_No ;
USART_InitStruct->USART_Mode = USART_Mode_Rx | USART_Mode_Tx;
USART_InitStruct->USART_HardwareFlowControl = USART_HardwareFlowControl_None;
}




void USART_ClockStructInit(USART_ClockInitTypeDef* USART_ClockInitStruct)
{

USART_ClockInitStruct->USART_Clock = USART_Clock_Disable;
USART_ClockInitStruct->USART_CPOL = USART_CPOL_Low;
USART_ClockInitStruct->USART_CPHA = USART_CPHA_1Edge;
USART_ClockInitStruct->USART_LastBit = USART_LastBit_Disable;
}

/*******************************************************************************
*****/
```

当然了你也可以自己设参数，比如这样。

```
void USART_Configuration(void)

{

USART_InitTypeDef USART_InitStructure;
USART_ClockInitTypeDef USART_ClockInitStructure;
```

```
USART_InitStructure.USART_BaudRate = 9600;
USART_InitStructure.USART_WordLength = USART_WordLength_8b;
USART_InitStructure.USART_StopBits = USART_StopBits_1;
USART_InitStructure.USART_Parity = USART_Parity_No;
USART_InitStructure.USART_HardwareFlowControl = USART_HardwareFlowControl_None;
USART_InitStructure.USART_Mode = USART_Mode_Rx | USART_Mode_Tx;

USART_ClockInitStructure.USART_Clock = USART_Clock_Disable;
USART_ClockInitStructure.USART_CPOL = USART_CPOL_Low;
USART_ClockInitStructure.USART_CPHA = USART_CPHA_2Edge;
USART_ClockInitStructure.USART_LastBit = USART_LastBit_Disable;

USART_ClockInit(USART1, &USART_ClockInitStructure);
USART_Init(USART1, &USART_InitStructure);

USART_Init(USART1, &USART_InitStructure);

USART_ClockInit(USART1, &USART_ClockInitStructure);
USART_Cmd(USART1, ENABLE);

} ////USART_ClockInitStructure.USART_CPHA= USART_CPHA_2Edge;除了这句以外其他的都和默认的参数一样，二者有啥区别我至今也不太清楚但就一般的应用来说两个都可以正常工作。
```

收发的方法：

1.发送

```
void USART1_Puts(char * str)
{
while(*str)
{
USART_SendData(USART1, *str++);

while(USART_GetFlagStatus(USART1, USART_FLAG_TXE) == RESET);
}
}
```

USART1_Puts("hello-java~!\r\n"); //这样就发送了 hello-java~!

跟 C 语言的 printf 不太一样在于\n 并没有另起一行要用个\r 这样在终端上好看。

2.接收

u8 uart1_get_data; //存放接受的内容

while (1)

{

if(USART_GetFlagStatus(USART1,USART_IT_RXNE)==SET)
{
uart1_get_data = USART_ReceiveData(USART1);
USART1_Puts("\r\n 获取到串口 1 数据:");
USART1_Putc(uart1_get_data);
USART1_Puts("\r\n");
}

}