* + 1. 软件设计：中断的方式接收（寄存器）

USART提供了多个中断事件。



* + - 1. Driver\_USART.c 添加中断相关代码

#include "Driver\_USART.h"

/\*\*

 \* @description: 初始化串口1

 \*/

void Driver\_USART1\_Init(void)

{

    /\* 1. 开启时钟 \*/

    /\* 1.1 串口1外设的时钟 \*/

    RCC->APB2ENR |= RCC\_APB2ENR\_USART1EN;

    /\* 1.2 GPIO时钟 \*/

    RCC->APB2ENR |= RCC\_APB2ENR\_IOPAEN;

    /\* 2. 配置GPIO引脚的工作模式  PA9=Tx(复用推挽 CNF=10 MODE=11)  PA10=Rx(浮空输入 CNF=01 MODE=00)\*/

    GPIOA->CRH |= GPIO\_CRH\_CNF9\_1;

    GPIOA->CRH &= ~GPIO\_CRH\_CNF9\_0;

    GPIOA->CRH |= GPIO\_CRH\_MODE9;

    GPIOA->CRH &= ~GPIO\_CRH\_CNF10\_1;

    GPIOA->CRH |= GPIO\_CRH\_CNF10\_0;

    GPIOA->CRH &= ~GPIO\_CRH\_MODE10;

    /\* 3. 串口的参数配置 \*/

    /\* 3.1 配置波特率 115200 \*/

    USART1->BRR = 0x271;

    /\* 3.2 配置一个字的长度 8位 \*/

    USART1->CR1 &= ~USART\_CR1\_M;

    /\* 3.3 配置不需要校验位 \*/

    USART1->CR1 &= ~USART\_CR1\_PCE;

    /\* 3.4 配置停止位的长度 \*/

    USART1->CR2 &= ~USART\_CR2\_STOP;

    /\* 3.5 使能接收和发送 \*/

    USART1->CR1 |= USART\_CR1\_TE;

    USART1->CR1 |= USART\_CR1\_RE;

    /\* 3.6 使能串口的各种中断 \*/

    USART1->CR1 |= USART\_CR1\_RXNEIE; /\* 接收非空中断 \*/

    USART1->CR1 |= USART\_CR1\_IDLEIE; /\* 空闲中断 \*/

    /\* 4. 配置NVIC \*/

    /\* 4.1 配置优先级组 \*/

    NVIC\_SetPriorityGrouping(3);

    /\* 4.2 设置优先级 \*/

    NVIC\_SetPriority(USART1\_IRQn, 2);

    /\* 4.3 使能串口1的中断 \*/

    NVIC\_EnableIRQ(USART1\_IRQn);

    /\* 4. 使能串口 \*/

    USART1->CR1 |= USART\_CR1\_UE;

}

/\*\*

 \* @description: 发送一个字节

 \* @param {uint8\_t} byte 要发送的字节

 \*/

void Driver\_USART1\_SendChar(uint8\_t byte)

{

    /\* 1. 等待发送寄存器为空 \*/

    while ((USART1->SR & USART\_SR\_TXE) == 0)

        ;

    /\* 2. 数据写出到数据寄存器 \*/

    USART1->DR = byte;

}

/\*\*

 \* @description: 发送一个字符串

 \* @param {uint8\_t} \*str 要发送的字符串

 \* @param {uint16\_t} len 字符串中字节的长度

 \* @return {\*}

 \*/

void Driver\_USART1\_SendString(uint8\_t \*str, uint16\_t len)

{

    for (uint16\_t i = 0; i < len; i++)

    {

        Driver\_USART1\_SendChar(str[i]);

    }

}

/\*\*

 \* @description: 接收一个字节的数据

 \* @return {\*} 接收到的字节

 \*/

uint8\_t Driver\_USART1\_ReceiveChar(void)

{

    /\* 等待数据寄存器非空 \*/

    while ((USART1->SR & USART\_SR\_RXNE) == 0)

        ;

    return USART1->DR;

}

/\*\*

 \* @description: 接收变长数据.接收到的数据存入到buff中

 \* @param {uint8\_t} buff 存放接收到的数据

 \* @param {uint8\_t} \*len 存放收到的数据的字节的长度

 \*/

void Driver\_USART1\_ReceiveString(uint8\_t buff[], uint8\_t \*len)

{

    uint8\_t i = 0;

    while (1)

    {

        // 等待接收非空

        while ((USART1->SR & USART\_SR\_RXNE) == 0)

        {

            // 在等待期间, 判断是否收到空闲帧

            if (USART1->SR & USART\_SR\_IDLE)

            {

                \*len = i;

                return;

            }

        }

        buff[i] = USART1->DR;

        i++;

    }

}

/\* 缓冲接收到的数据 \*/

uint8\_t buff[100] = {0};

/\* 存储接收到的字节的长度 \*/

uint8\_t len = 0;

uint8\_t isToSend = 0;

void USART1\_IRQHandler(void)

{

    /\* 数据接收寄存器非空 \*/

    if (USART1->SR & USART\_SR\_RXNE)

    {

        // 对USART\_DR的读操作可以将接收非空的中断位清零。 所以不用单独清除了.

        //USART1->SR &= ~USART\_SR\_RXNE;

        buff[len] = USART1->DR;

        len++;

    }

    else if (USART1->SR & USART\_SR\_IDLE)

    {

        /\* 清除空闲中断标志位: 先读sr,再读dr.就可以实现清除了 \*/

        USART1->SR;

        USART1->DR;

        /\* 变长数据接收完毕 \*/

        //Driver\_USART1\_SendString(buff, len);

        isToSend = 1;

        /\* 把接收字节的长度清0 \*/

        // len = 0;

    }

}

* + - 1. main.c

#include "Driver\_USART.h"

/\* 缓冲接收到的数据 \*/

extern uint8\_t buff[100];

/\* 存储接收到的字节的长度 \*/

extern uint8\_t len;

extern uint8\_t isToSend;

int main()

{

    Driver\_USART1\_Init();

    Driver\_USART1\_SendString("abc", 3);

    while (1)

    {

        if(isToSend){

            Driver\_USART1\_SendString(buff, len);

            isToSend = 0;

            len = 0;

        }

    }

}