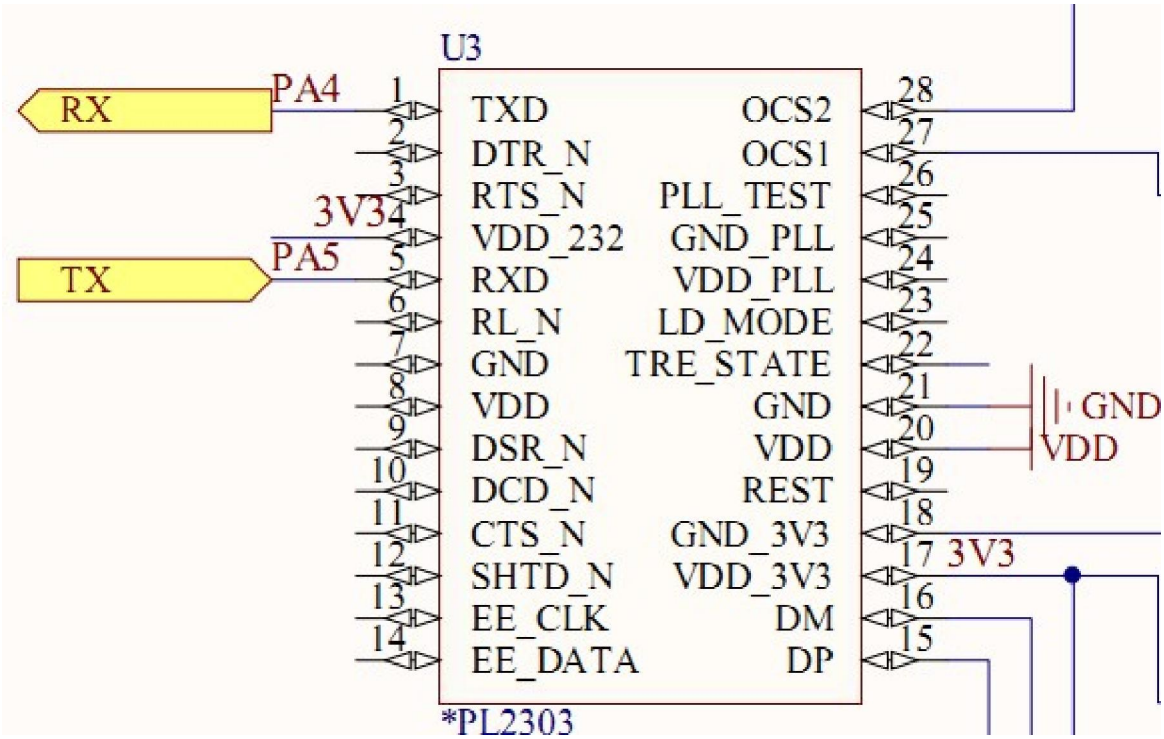
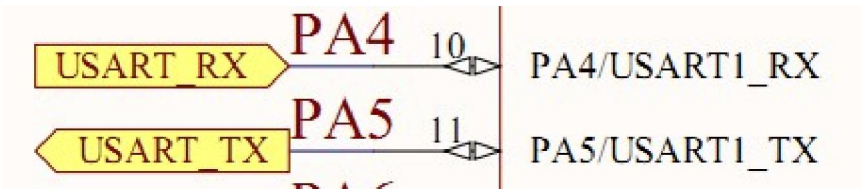
例程三 串口通信

学玩前面两个例程之后，是否觉得 STM8S 操作起来也很简单吧，其实是的。在这个例程我要讲下串口通信的设置和用法。但我们学一款新的单片机，必须要做到这样的步骤，第一会点亮 LED，第二会按键扫描，第三就是要会串口通信。为什么呢要这样的安排呢？会 LED 和按键扫描的就是学会的 IO 口的输出与输入，这是最基本的，但学会这两个的话，就要学串口通信，因为串口通信对其他模块的调式是会有很大的帮助，可以这样说必须的。这些当你接触到你就会知道的了。好的，下面看下我们风驰电子 STM8 开发板的串口通信的原理图吧。





我们的板子上以利用 PL2303 来下载程序和串口通信，只要大家按照上面的链接的话就可以了。这里我只是把重要的截图给大家看而已，大家可以参考我们的风驰电子 STM8S 开发板原理图.pdf 文件的

要用到内部资源：

"stm8s_clk.h"

"stm8s_uart1.h"

"stm8s_clk.c"

"stm8s_uart1.c"

好的，我们先看我们的主程序

```
int main(void)
{
   u8 len ;
  /* Infinite loop */

  /*设置内部时钟16M为主时钟*/

  CLK_HSIPrescalerConfig(CLK_PRESCALER_HSIDIV1);
  /*!<Set High speed internal clock  */

   Uart_Init();
   __enable_interrupt();
   UART1_SendString("Serial Communication ---STM8 Development Board of FengChi Electron ",\
             sizeof("Serial Communication ---STM8 Development Board of FengChi Electron"));
   Delay(0xffff);
   UART1_SendByte('\n');
   Delay(0xffff);
 while (1)
 {


    if(UART_RX_NUM&0x80)
    {
      len=UART_RX_NUM&0x3f;/*得到此次接收到的数据长度*/
      UART1_SendString("You sent the messages is:",sizeof("You sent the messages is"));
      UART1_SendString(RxBuffer,len);
      UART1_SendByte('\n');
      UART_RX_NUM=0;
    }
  }
```

时钟的初始化在前面都有说了，这里就不说了，下面主要来看一下串口的初始化 Uart_Init();。
该函数的原型

```
/* ********************************************
UART1  configured as follow:
  - BaudRate = 115200 baud
  - Word Length = 8 Bits
  - One Stop Bit
  - No parity
  - Receive and transmit enabled
  -  Receive interrupt
  - UART1 Clock disabled
**********************************************/
void Uart_Init(void)
{
    UART1_DeInit();
    UART1_Init((u32)115200, UART1_WORDLENGTH_8D, UART1_STOPBITS_1, \
    UART1_PARITY_NO , UART1_SYNCMODE_CLOCK_DISABLE , UART1_MODE_TXRX_ENABLE);
    UART1_ITConfig(UART1_IT_RXNE_OR,ENABLE  );
    UART1_Cmd(ENABLE );

}
```

在这个例程是设置波特率为 115200，8 位数据位，1 位停止位，没有奇偶校验，查询方式发送，中断方式接收。Uart1 的时钟是关闭的。
下面，在看看几个重要的函数：

```
void UART1_SendByte(u8 data)
{
    UART1_SendData8((unsigned char)data);
  /* Loop until the end of transmission */
  while (UART1_GetFlagStatus(UART1_FLAG_TXE) == RESET);
}
```

这个函数是发送一个字节，查询发送。当发送寄存器是空的就发送。

```
/**
  * @brief  Transmits 8 bit data through the UART1 peripheral.
  * @param  Data: The data to transmit.
  * @retval None
  */
void UART1_SendData8(uint8_t Data)
{
    /* Transmit Data */
    UART1->DR = Data;
}
```

这个函数是把一个字节写进发送寄存器。

```
/**
  * @brief  Checks whether the specified UART1 flag is set or not.
  * @param  UART1_FLAG specifies the flag to check.
  *         This parameter can be any of the @ref UART1_Flag_TypeDe
  * @retval FlagStatus (SET or RESET)
  */
FlagStatus UART1_GetFlagStatus(UART1_Flag_TypeDef UART1_FLAG)
{
```

这个函数是获得标志的一个函数，在这里的参数是选择是发送寄存器标志。
这两个函数是库自带的，直接调用过来的。所以说基于库开发 STM8 是挺快的，
不用直接操作寄存器。直接调用库里面的函数。

```
typedef enum { UART1_FLAG_TXE   = (uint16_t)0x0080, /*!< Transmit Data Register Empty flag */
               UART1_FLAG_TC    = (uint16_t)0x0040, /*!< Transmission Complete flag */
               UART1_FLAG_RXNE  = (uint16_t)0x0020, /*!< Read Data Register Not Empty flag */
               UART1_FLAG_IDLE  = (uint16_t)0x0010, /*!< Idle line detected flag */
               UART1_FLAG_OR    = (uint16_t)0x0008, /*!< OverRun error flag */
               UART1_FLAG_NF    = (uint16_t)0x0004, /*!< Noise error flag */
               UART1_FLAG_FE    = (uint16_t)0x0002, /*!< Framing Error flag */
               UART1_FLAG_PE    = (uint16_t)0x0001, /*!< Parity Error flag */
               UART1_FLAG_LBDF  = (uint16_t)0x0210, /*!< Line Break Detection Flag */
               UART1_FLAG_SBK   = (uint16_t)0x0101  /*!< Send Break characters Flag */
             } UART1_Flag_TypeDef;
```

这个就是各种标志位参数，如果大家想查询哪个标志位的是什么状态的话就可以
调用这个函数和这些标志参数，在库里面会有各种子函数，都是官方封装好的，
用起来挺方便的，是不是学起来比 51 单片机和 AVR 方便多了。

```
void UART1_SendString(u8* Data,u16 len)
{
  u16 i=0;
  for(;i<len;i++)
    UART1_SendByte(Data[i]);

}
```

这个函数是发送字符串的，包括英文和中文的字符串。

```
u8 UART1_ReceiveByte(void)
{
     u8 USART1_RX_BUF;
     while (UART1_GetFlagStatus(UART1_FLAG_RXNE) == RESET);
     USART1_RX_BUF=UART1_ReceiveData8();
     return  USART1_RX_BUF;

}
```

这个是函数是利用查询方法来接收一个字节，用查询方法来接收一个字节的话，会限制 CPU 的利用率，因为 CPU 要等到接收到了数据才释放，这样对于开发利用不好，所以还是提倡中断接收，这样会有效的利益 CPU。查询的话，也是查询接收寄存器，如果是 RXNE==RESET 的话就表明可以接受这个数据了。
下面重点介绍一下中断服务函数的写法。在我们例程中我把所有中断函数都列出来，只要大家添加“stm8s_interrupt.c”这个文件就可以，大家可以在相应的位置添加自己的代码就可以了。
下面介绍一下接收中断服务函数：

```
#pragma vector=0x14
__interrupt void UART1_RX_IRQHandler(void)
{
   u8 Res;
    if(UART1_GetITStatus(UART1_IT_RXNE )!= RESET)
    {/*接收中断(接收到的数据必须是0x0d 0x0a结尾)*/
        Res =UART1_ReceiveData8();
        /*(USART1->DR);读取接收到的数据,当读完数据后自动取消RXNE的中断标志位*/
        if(( UART_RX_NUM&0x80)==0)/*接收未完成*/
        {
            if( UART_RX_NUM&0x40)/*接收到了0x0d*/
                {
                  if(Res!=0x0a) UART_RX_NUM=0;/*接收错误,重新开始*/
                  else  UART_RX_NUM|=0x80;       /*接收完成了 */
                }
            else /*还没收到0X0D*/
              {
                if(Res==0x0d) UART_RX_NUM|=0x40;
                else
                  {
                    RxBuffer[ UART_RX_NUM&0X3F]=Res ;
```

```
                UART_RX_NUM++;
                if( UART_RX_NUM>63) UART_RX_NUM=0;/*接收数据错误,重新开始接收*/
            }
        }
    }
        }
```

接收中断函数的中断号是 0x14。所以设置`#pragma vector=0x14`。中断服务函数里面是用来接收一次最多能接收 64Byte 的数据，并且是以回车键结束的数据，具体的意思看上面的注释，注释很清楚的说明的，在这里就不多说了。最后跟大家说过最最重要的东西。怎么样开中断呢？只有总中断打开，才可以触发中断，在 IAR 编译器自带了一个开总中断的函数。所以我们要添加相应的头文件才能调用它的函数，所以要在主函数的文件中加上`#include "intrinsics.h"`，否则，编译不成功。开总中断`__enable_interrupt();`关中断`disable_interrupt();`
看下他们的函数原型

```
__intrinsic void __enable_interrupt(void);      /* RIM */
__intrinsic void __disable_interrupt(void);     /* SIM */
```

里面是用汇编写的，所以我们不必去理会它，只要我们直接调用就可以了。

实验现象：


串口调试助手 （C■精装版 V3.7.1）
串口设置
串口号 COM4
波特率 115200
校验位 NONE
数据位 8
停止位 1
断开
接收区设置
接收转向文件...
自动换行显示
十六进制显示
暂停接收显示
保存数据 清除显示
发送区设置
启用文件数据源...
自动发送附加位
发送完自动清空
按十六进制发送
数据流循环发送
发送间隔 20 毫秒
文件载入 清除输入
串口数据接收
Serial Communication ---STM8 Development Board of FengChi Electron
You sent the messages is:串口通信测试成功--风驰电子STM8开发板
串口通信测试成功--风驰电子STM8开发板
发送
就绪!
发送：114
接收：322
复位计数

风驰电子祝您学习愉快～～～！！！！！