实验 8、9 - 串口数据收发实验

1. 实验目的

掌握 NRF24LE1 的串口的配置和使用。

对单片机软件开发来说，掌握串口的使用至关重要，串口是软件开发重要的调试手段，在开发过程中，可以通过串口输出程序中涉及的某些中间量、程序运行状态等信息，从而方便开发者直观地观察单片机的运行。

某些情况下，串口调试是仿真器无法替代的，可以这么说：熟练的使用串口，必定会加快开发进度。

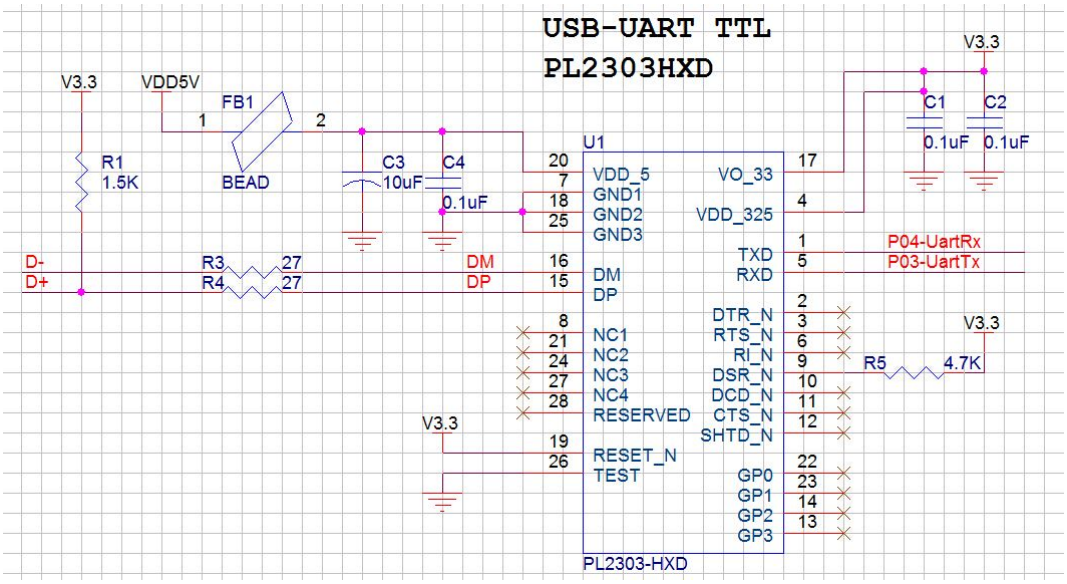
2. 实验内容

配置 NRF24LE1 的串口。

通过串口收发数据。

3. 实验原理

3.1. 电路原理



NRF24LE1 集成了一个串口，管脚分配如下：

P0.3：UART TXD

P0.4：UART RXD

NRF24LE1 的 UART 接口特性如下：

- 同步模式，固定速率；
- 8 位 UART 模式，可变速率；

- 9 位 UART 模式，可变速率；
- 9 位 UART 模式，固定速率；
- 附加波特率发生器。

注意：不推荐使用定时器 1 溢出作为波特率发生器。

开发板上使用了 USB 转 UART TTL 芯片 PL2303-HXD（注意：该芯片是最新的 HXD 版本，不需要外接晶振，早期 HX 版本的需要外接晶振）将串口信号转换为 USB 信号，方便和计算机连接。

NRF24LE1 配置串口的一般步骤：

- 配置 IO，使 IO 连接到串口。
- 配置串口的控制和状态寄存器。
- 配置串口工作的波特率。 此处配置为波特率为 57600。

3.2. 寄存器配置

NRF24LE1 的串口由寄存器 S0CON 控制，数据传送通过读/写 S0BUF 寄存器实现，波特率由 S0RELH、S0RELL 和 ADCON 寄存器来选择**(更详细的内容请查阅** NRF24LE1 数据手册**)**。

1. S0CON 寄存器

表 1：S0CON 寄存器

地址	复位值	位	名称	说明
0x98	0x00	7~6	sm0:sm1	串口 0 模式选择 00：模式 0，移位寄存器波特率为 ckCpu / 12。 01：模式 1,8 位 UART。 10：模式 2,9 位 UART 波特率为 ckCpu /32 or ckCpu/64[a]。 11：模式 3，9 位 UART。
		5	sm20	多处理器通信使能。
		4	ren0	串口接收使能：置 1 允许接收。
		3	tb80	发送位 8，在串口模式 2、3 中，tb8 发送机要发送的第 9 位数据（如奇偶校验或多机通信），由软件进行控制。
		2	rb80	接收位 8，在串口模式 2、3 中，rb8 为接收机收到的第 9 位数据。
		1	ti0	发送中断标志。置位时指示串口 0 完成一次发送。模式 0 下第 8 位完成后或其他模式下在停止位开始时，由硬件置位。 如果再发送，必须由软件清零。
		0	ri0	接收中断标志。置位时指示串口 0 完成一次接收。模式 0 下第 8 位完成后或其他模式下在停止位中间时，由硬件置位。 如果再接收，必须由软件清零。

2. 串口 0 数据缓冲器 S0BUF

写数据到寄存器 S0BUF 将使数据移入串口输出缓冲器并开始通过串口 0 发送；读寄存

器 S0BUF 将会读出串行接收缓冲器所接收的数据。

表 2：S0BUF 寄存器

地址	复位值	名称
0x99	0x00	S0BUF

3. 串口 0 重载寄存器 S0RELH、S0RELL

串口 0 重载寄存器用作串口 0 的波特率发生器，只使用了 10 位，8 位在 S0RELL 寄存器，2 位在 S0RELH 寄存器。

表 3：S0RELH/S0RELL 寄存器

地址	复位值	名称
0xAA	0xD9	S0RELL
0xBA	0x03	S0RELH

4. 串口 0 波特率选择寄存器 ADCON

此寄存器的最高位用来设置串口 0 的波特率发生器。

表 4：ADCON 寄存器

地址	复位值	位	名称	说明
0xD8	0x00	7	bd	串口 0 波特率选择（模式 1 和模式 3）
		6~0		未使用

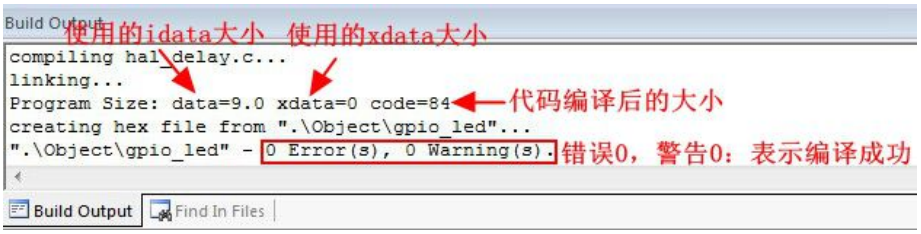
根据上述内容，我们来学习串口的配置，对串口配置如下。

```
S0CON = 0x50;        // 8 位 UART，使能接收
PCON |= 0x80;        // SMOD = 1
ADCON |= 0x80;       //  使用内部波特率发生器

S0RELL = (uint8_t)baud;   //配置波特率
S0RELH = (uint8_t)(baud >> 8);
```

4. 串口实验：数据收发

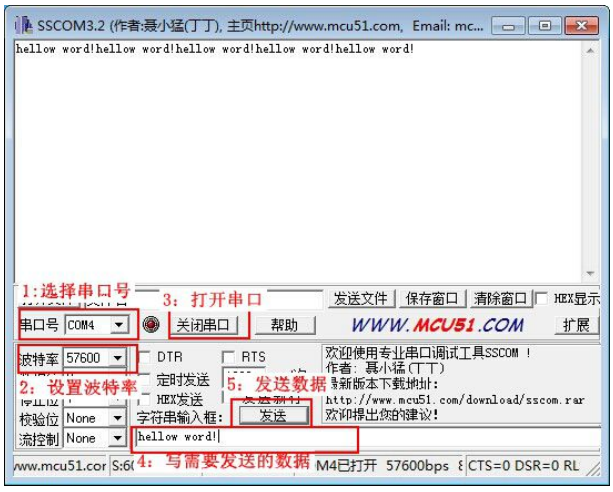
4.1. 实验步骤

- 在 Keil uVision4 中打开工程“uart_echo.uvproj”工程；
- 编译工程，注意查看编译输出栏，观察编译的结果，如果有错误，修改程序，直到编译成功为止；



- 将编译生成的 HEX 文件“uart_echo.hex” (该文件位于工程目录下的“Object”文件夹中)通过编程器下载到开发板中运行。

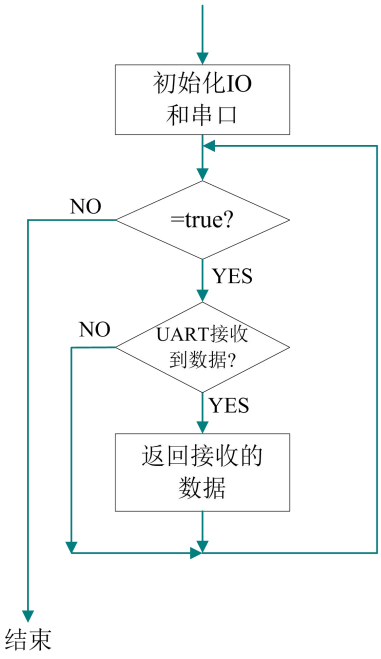
- 程序运行后，D1 闪烁指示程序已经正常运行。
- 打开串口调试助手，选择串口号，设置波特率为 57600，打开串口，写好需要发送的数据，点击发送按钮发送数据。



- 观察接收窗口，应能正确接收到发送的数据(NRF24LE1 接收到数据后会将接收到的数据通过串口直接返回)。

4.2. 实验程序

4.2.1. 程序流程



4.2.2. 程序清单

```
#define   D1      P00   //开发板上的指示灯 D1

#define BAUD_57K6     1015   // = Round(1024 - (2*16e6)/(64*57600))
```

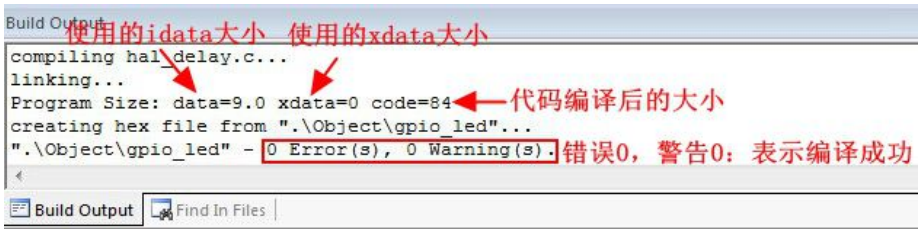
```
#define BAUD_38K4     1011    // = Round(1024 - (2*16e6)/(64*38400))
#define BAUD_19K2      998    // = Round(1024 - (2*16e6)/(64*19200))
#define BAUD_9K6       972    // = Round(1024 - (2*16e6)/(64*9600))
/*****************************************************************************
 *描   述 : 配置 IO P0.0 和 P0.1 为输出，驱动 LED。P03 输出: UART TXD，P04:输入 UART
            RXD
 *入   参 : 无
 *返回值 : 无
*****************************************************************************/
void IO_Init(void)
{
    P0DIR &= ~0x01;     //配置 P0.0 和 P0.1 为输出
    P0DIR &= ~0x08;     //P03:输出 UART TXD
    P0DIR |= 0x10;      //P04:输入 UART RXD
    D1 = 1;     //设置 D1 初始状态为熄灭
}
/*****************************************************************************
 *描   述 : 串口发送一个字符
 *入   参 : 无
 *返回值 : 无
*****************************************************************************/
void uart_sendchar(uint8_t dat)
{
  S0BUF = dat;
  while(!TI0);
  TI0 = 0;
}

/*****************************************************************************
 *描   述 : 串口打印字符串
 *入   参 : 无
 *返回值 : 无
*****************************************************************************/
void PutString(char *s)
{
  while(*s != 0)
    uart_sendchar (*s++);
}
/*****************************************************************************
 *描   述 : 主函数
 *入   参 : 无
 *返回值 : 无
*****************************************************************************/
```

```
void main(void)
{
  uint8_t   temp_char;

  IO_Init();   //初始化 IO
  uart_init(BAUD_57K6); //  初始化 UART，波特率 57600

  while(hal_clk_get_16m_source() != HAL_CLK_XOSC16M) ;//等待时钟稳定

  PutString("Welcome to FiYu!\n"); //启动时，打印字符串

  while(1)
  {
      if(RI0==1)//串口接收到数据?
      {
        D1 = ~D1;//指示灯状态取反
        temp_char = S0BUF; //读数据
        RI0 = 0;
        S0BUF = temp_char; //写入发送的数据
        while(!TI0);              //等待发送完成
        TI0=0;
      }
   }
}
```

5. 串口实验：串口控制指示灯

5.1. 实验步骤

- 在 Keil uVision4 中打开工程“uart_led.uvproj”工程；
- 编译工程，注意查看编译输出栏，观察编译的结果，如果有错误，修改程序，直到编译成功为止；

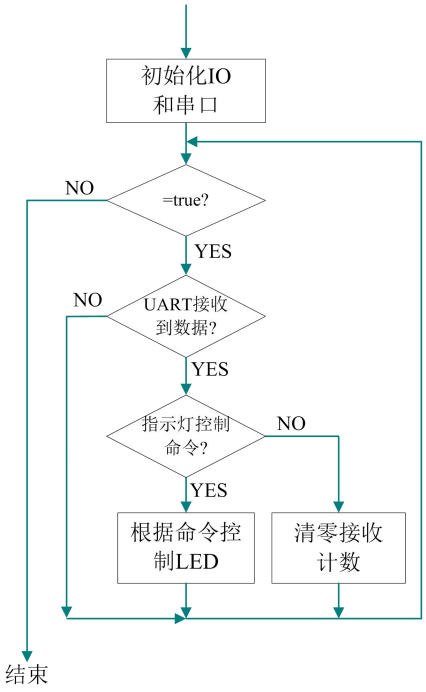


- 将编译生成的 HEX 文件“uart_led.hex” (该文件位于工程目录下的“Object”文件夹中)通过编程器下载到开发板中运行。
- 观察指示灯 D1，应以 500ms 的间隔闪烁。
- 打开串口调试助手，选择串口号，设置波特率为 57600，打开串口，发送命令点亮相应的指示灯。发送字符“D1#”点亮指示灯 D1，熄灭指示灯 D2。发送字符“D2#”

点亮指示灯 D2，熄灭指示灯 D1。

5.2. 实验程序

5.2.1. 程序流程



5.2.2. 程序清单

```
#define   D1      P00   //开发板上的指示灯 D1
#define   D2      P01   //开发板上的指示灯 D2

#define   LED_ON        0 //点亮指示灯
#define   LED_OFF       1 //熄灭指示灯
#define   RXBUF_LEN     3 //UART 接收缓存字节数

/*******************************************************************************
 *描   述 : 主函数
 *入   参 : 无
 *返回值 : 无
*******************************************************************************/
void main(void)
{
   uint8_t UartRxBuf[RXBUF_LEN];   //UART 接收缓存
   uint8_t RxCnt = 0;              //UART 接收字节数
   uint8_t RxByte;                 //UART 接收的字节
```

```
IO_Init();   //初始化 IO
hal_uart_init(UART_BAUD_57K6); // 初始化 UART，波特率 57600

// Wait for XOSC to start to ensure proper UART baudrate
while(hal_clk_get_16m_source() != HAL_CLK_XOSC16M) ;//

EA = 1;   // 开启全局中断
PutString("Welcome to FiYu!\n");

while(1)
{
  while(hal_uart_chars_available())     //UART 接收到数据
  {
    RxByte = hal_uart_getchar();        // UART 接收到数据中读取一个字节
    if((RxByte != '#')&&(RxCnt < 3)) //#是指示灯操作命令的结束符
    {
      UartRxBuf[RxCnt++] = RxByte; //UART 接收的输入放入缓存
    }
    else
    {
      if(RxCnt >= 3) //数据非法
      {
        RxCnt = 0;   //清零 UART 接收计数
      }
      else
      {
        if((UartRxBuf[0] = 'D') || (UartRxBuf[0] = 'd'))//是指示灯控制命令?
        {
          switch(UartRxBuf[1]-48) //ASICC 码转成数字，得到指示灯编号
          {
            case 1:
              D1 = 0;      //点亮 D1
              D2 = 1;      //熄灭 D2
              RxCnt = 0; //清零 UART 接收计数
              break;

            case 2:
              D1 = 1;
              D2 = 0;
              RxCnt = 0;
              break;

            default:
```

```
                    break;
                }
            }
        }
    }
  }
}
```