

实验 6 - UART 数据收发

1. 实验目的

掌握 NRF52832 串口的配置和使用。

2. 实验内容

配置 NRF52832 的串口。

- 波特率：115200。
- 数据位：8 位。
- 停止位：1 位。
- 无校验。
- 硬件流控：关闭。**

程序运行后，通过串口调试助手发送数据，NRF52832 会将接收到的数据返回。

3. 实验设备

硬件	
1.	IK-52832DK 开发板
2.	USB MINI 数据线
3.	JLINK 仿真器
4.	JTAG-SWD 转接板、排线
软件	
1.	win7/win8.1 系统
2.	MDK5.18A 集成开发环境

4. 实验原理

4.1. 电路原理

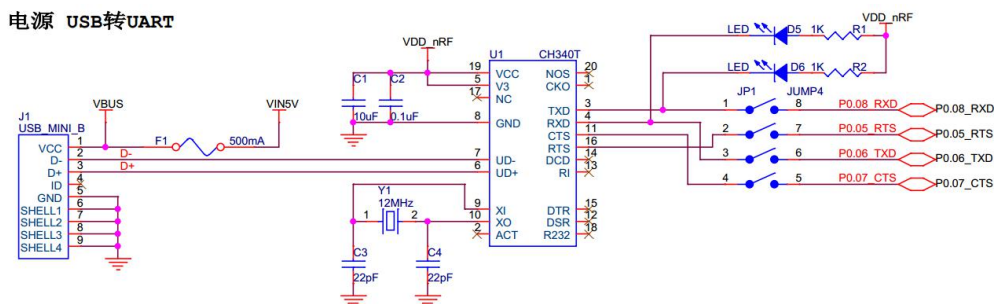


图 1: USB 转串口电路

NRF52832 的串口支持如下特性：

- 全双工。
- 自动的流控。
- 奇偶校验并自动产生校验位。

开发板上的串口接收和发送的管脚上连接了指示灯，收发数据时指示灯会闪烁，这样，更方便我们从硬件的角度观察串口有没有在进行数据收发。

4.2. UART

1. 管脚配置

NRF52832 的任何一个 GPIO 都可以用作 UART。这样极大的提高了布线的灵活性，有效的降低了 PCB 的尺寸（或者层数）。当然，同时只能配置一个 UART。

在 IK-52832DK 开发板中，UART 管脚配置如下

UART 管脚配置	
UART	管脚号
UART_RX	P0.08
UART_TX	P0.06
UART_CTS	未使用
UART_RTS	未使用

2. 奇偶校验

当使能自动奇偶校验后，发送和接收的 TXD 和 RXD 会分别自动生成奇偶校验。

3. Error

下列两种情形会导致错误事件的产生：

- 结束位没有被正确识别。
- RXD 一直被拉低，并且被拉低的时间超过一帧数据的长度。

4. 流控

CTS 和 RTS 用于流控，可以通过 [CONFIG](#) 寄存器使能和关闭流控，一般情况下用不到流控(本实验中禁止了流控)。如果使能了流控，串口调试软件中也要根据代码配置开启或关闭 RTS/CTS 的选项。

4.3. 配置并使用 UART

使用 UART 时，需要先配置 UART 的各个参数。本实例中，UART 的配置过程很简单，先使用 `app_uart_comm_params_t` 定义并初始化一个 UART 配置用的结构体 `comm_params`，之后调用 UART 初始化函数 `APP_UART_FIFO_INIT` 对 UART 进行初始化。

comm_params 的定义和初始化：

```
void uart_config(void)
{
```

```

uint32_t err_code;

const app_uart_comm_params_t comm_params =
{
    RX_PIN_NUMBER,
    TX_PIN_NUMBER,
    RTS_PIN_NUMBER,
    CTS_PIN_NUMBER,
    APP_UART_FLOW_CONTROL_DISABLED,    //关闭流控
    false,
    UART_BAUDRATE_BAUDRATE_Baud115200 //波特率设置为 115200bps
};

APP_UART_FIFO_INIT(&comm_params,
                   UART_RX_BUF_SIZE,
                   UART_TX_BUF_SIZE,
                   uart_error_handle,
                   APP_IRQ_PRIORITY_LOW,
                   err_code);

APP_ERROR_CHECK(err_code);
}

```

app_uart_comm_params_t 的声明如下:

```

typedef struct
{
    uint8_t          rx_pin_no;    //RX 管脚号
    uint8_t          tx_pin_no;    //TX 管脚号
    uint8_t          rts_pin_no;   //RTS 管脚号, 仅用于流控使能的情况下
    uint8_t          cts_pin_no;   //CTS 管脚号, 仅用于流控使能的情况下
    app_uart_flow_control_t flow_control; //流控配置
    bool             use_parity;    //是否使用校验
    uint32_t         baud_rate;    //波特率
} app_uart_comm_params_t;

```

调用 APP_UART_FIFO_INIT 初始化 UART:

调用此函数, 按照 comm_params 中的参数配置 UART, 同时设置 UART 的接收和发送缓存、事件处理以及中断优先级。

```

APP_UART_FIFO_INIT(&comm_params,
                   UART_RX_BUF_SIZE,
                   UART_TX_BUF_SIZE,
                   uart_error_handle,
                   APP_IRQ_PRIORITY_LOW,
                   err_code);

```

5. 开发板电路连接

本实验需要用跳线帽短接 P.06 和 P0.08 管脚，如下图红框所示：

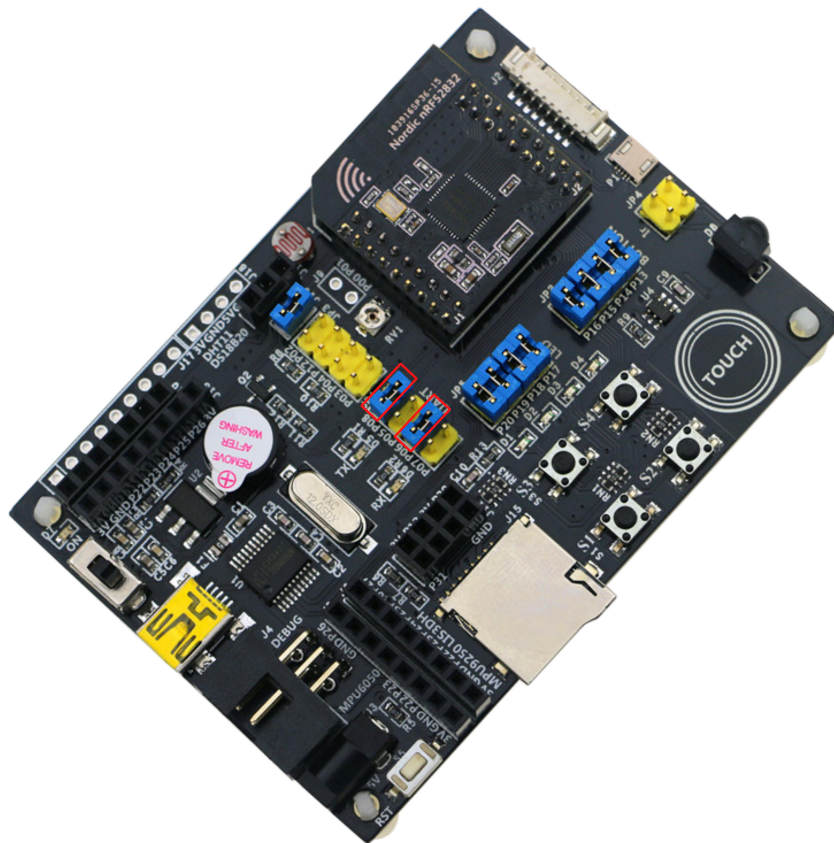




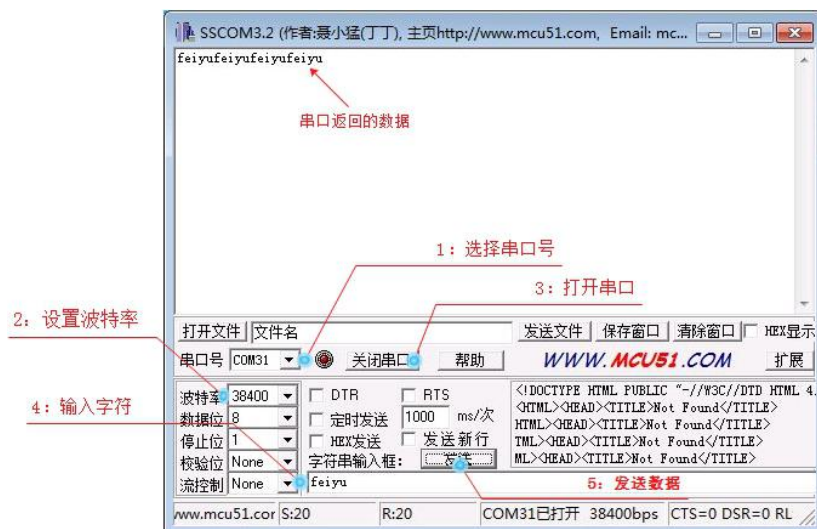
图 2：开发板跳线连接

6. 实验步骤

- 拷贝出“...\实验 6 - UART 数据收发”目录下的 uart 文件夹，存放至合适的目录，如“D:\NRF52832”。**强烈建议不要在资料包中直接打开工程，因为包含了中文路径且工程路径较深，可能会出现问题。**
- 启动 MDK5.18A。
- 在 MDK5 中执行“Project→Open Project”打开“...\uart\project\”目录下的工程“uart.uvproj”。
- 点击编译按钮编译工程 。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。编译后生成的 HEX 文件“uart.hex”位于工程目录下的“Objects”文件夹中。

```
linking...
Program Size: Code=408 RO-data=224 RW-data=4 ZI-data=2052
FromELF: creating hex file...
".\_build\led.axf" - 0 Error(s), 0 Warning(s). 错误: 0, 警告: 0表示编译通过
Build Time Elapsed: 00:00:04
```

- 点击下载按钮下载程序。如果需要对程序进行仿真，点击 Debug 按钮即可将程序下载到 NRF52832 进行仿真。
- 用 USB MIN 数据线连接开发板到计算机，打开串口调试助手，设置好参数(波特率设置为 115200)，如下图。



- 输入数据“feiyu”，点击发送按钮发送数据，观察串口调试助手接收窗口，接收到的数据应和发送的数据一样。

7. 实验程序

```
#define UART_TX_BUF_SIZE 256    /**< UART TX 缓存大小 */
#define UART_RX_BUF_SIZE 1     /**< UART RX 缓存大小*/

void uart_config(void)
{
    uint32_t err_code;

    const app_uart_comm_params_t comm_params =
    {
        RX_PIN_NUMBER,
        TX_PIN_NUMBER,
```

```
    RTS_PIN_NUMBER,
    CTS_PIN_NUMBER,
    APP_UART_FLOW_CONTROL_DISABLED,    //关闭流控
    false,
    UART_BAUDRATE_BAUDRATE_Baud115200 //波特率设置为 115200bps
};

APP_UART_FIFO_INIT(&comm_params,
                  UART_RX_BUF_SIZE,
                  UART_TX_BUF_SIZE,
                  uart_error_handle,
                  APP_IRQ_PRIORITY_LOW,
                  err_code);

APP_ERROR_CHECK(err_code);
}

/*****
* 描 述 : main函数
* 入 参 : 无
* 返回值 : 无
*****/
int main(void)
{
    LEDS_CONFIGURE(LED_MASK);
    LEDS_OFF(LED_MASK);

    uart_config();
    while (true)
    {
        uint8_t cr;
        while(app_uart_get(&cr) != NRF_SUCCESS); //等待接收串口数据
        while(app_uart_put(cr) != NRF_SUCCESS);  //返回接收到的串口数据
    }
}
```