

实验 7 - UART 控制指示灯

1. 实验目的

掌握 NRF52832 串口的配置和使用。

学习简单命令的解析。

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。

通过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，熄灭其他指示灯，发送字符“D2#”、“D3#”、“D4#”和发送“D1#”的功能一样，分别点亮相应的指示灯，同时，熄灭其他指示灯。

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

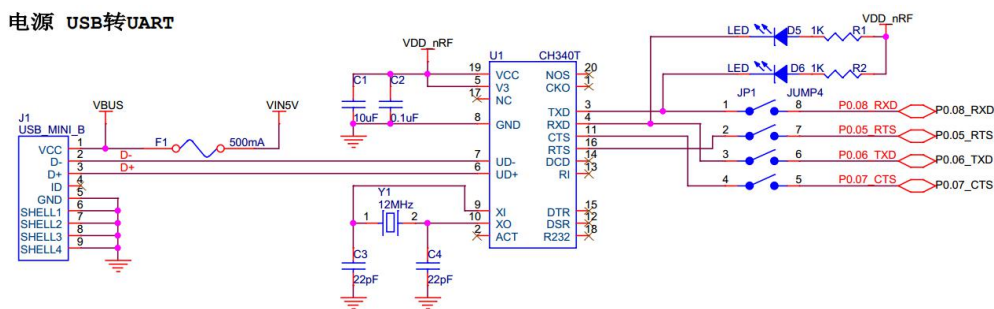


图 1: USB 转串口电路

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，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

4. 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。

5. 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。

6. 流控

CTS 和 RTS 用于控制流量，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串口：P0.06 和 P0.08，LED：P0.17~P0.20 管脚，如下图红框所示：

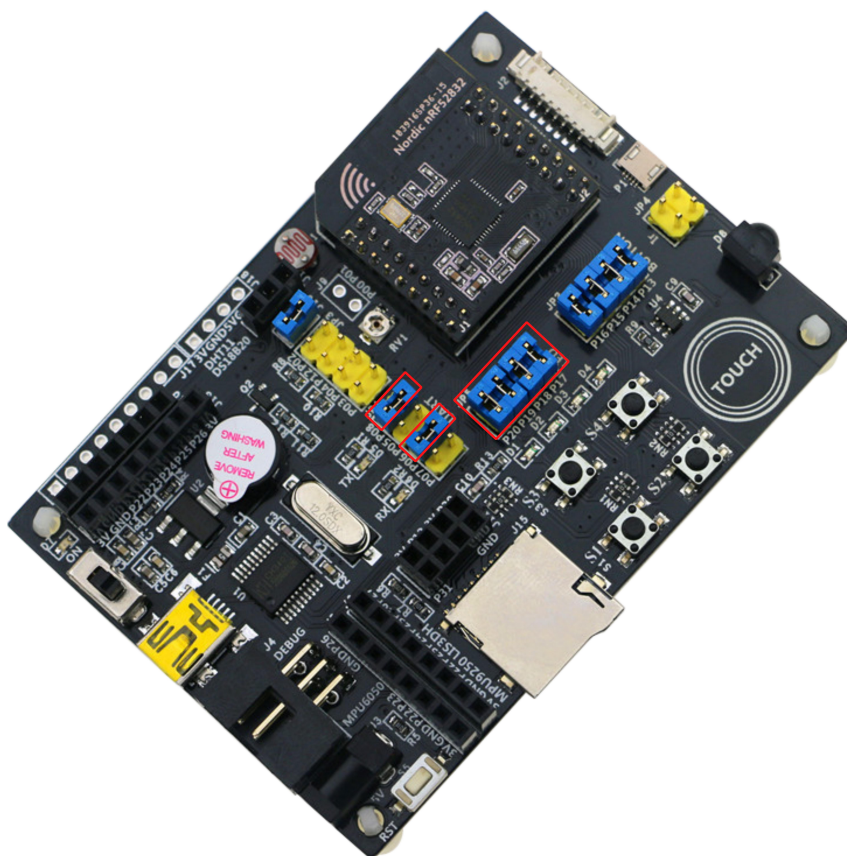




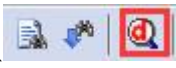
图 2：开发板跳线连接

6. 实验步骤

- 拷贝出 “...\6 - 开发板应用\3 - 基础实验\实验 7 - UART 控制指示灯” 目录下的 uart_led 文件夹，存放到合适的目录，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_led\project\” 目录下的工程 “uart_led.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_led.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，按照下图所示设置串口调试助手(波特率设置为 115200)。

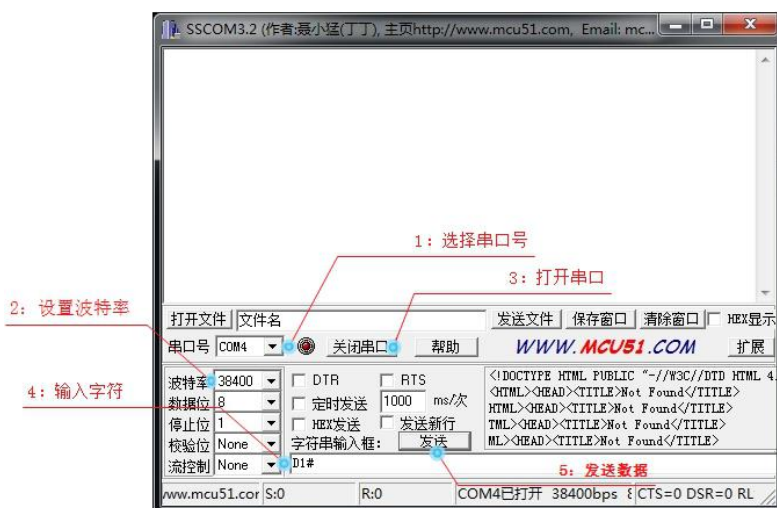


图 2: 串口调试助手设置

- 输入 “D1#”、“D2#”、“D3#” 或 “D4#”，点击发送按钮发送数据，观察开发板上的指示灯的亮灭，应和发送的指示灯编号一致。

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 */
#define RXBUF_LEN 3           //UART 接收缓存字节数

/*****
```

* 描 述 : 串口初始化。波特率 115200bps, 流控关闭

* 参 数 : 无

* 返回值 : 无

*****/

```
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)
{
    uint8_t RxCnt = 0;           //UART 接收字节数
    uint8_t UartRxBuf[RXBUF_LEN]; //UART 接收缓存

    LEDS_CONFIGURE(LED_MASK);    //配置驱动 LED 指示灯的管脚
    LEDS_OFF(LED_MASK);          //关闭所有指示灯
    uart_config();
}
```

```
while (true)
{
    uint8_t cr;
    while(app_uart_get(&cr) != NRF_SUCCESS);

    if((cr != '#') && (RxCnt < 3))
    {
        UartRxBuf[RxCnt++] = cr;
    }
    else
    {
        if(RxCnt >= 3)
        {
            RxCnt = 0;
        }
        else
        {
            if((UartRxBuf[0] == 'D') || (UartRxBuf[0] == 'd'))
            {
                switch(UartRxBuf[1]-48)
                {
                    case 1:
                        LEDS_OFF(LED_MASK); //关闭所有指示灯
                        nrf_gpio_pin_clear(LED_1); //点亮 D1 指示灯
                        RxCnt = 0;
                        break;

                    case 2:
                        LEDS_OFF(LED_MASK); //关闭所有指示灯
                        nrf_gpio_pin_clear(LED_2); //点亮 D2 指示灯
                        RxCnt = 0;
                        break;

                    case 3:
                        LEDS_OFF(LED_MASK); //关闭所有指示灯
                        nrf_gpio_pin_clear(LED_3); //点亮 D3 指示灯
                        RxCnt = 0;
                        break;

                    case 4:
                        LEDS_OFF(LED_MASK); //关闭所有指示灯
                        nrf_gpio_pin_clear(LED_4); //点亮 D4 指示灯
                        RxCnt = 0;
                        break;
                }
            }
        }
    }
}
```

```
        default:  
            break;  
    }  
    }  
    }  
    }  
    }  
}
```