更多嵌入式 Linux 学习资料，请关注：一口 Linux 回复关键字:1024


个人微信

公众号

一、CC2530 的串口资源

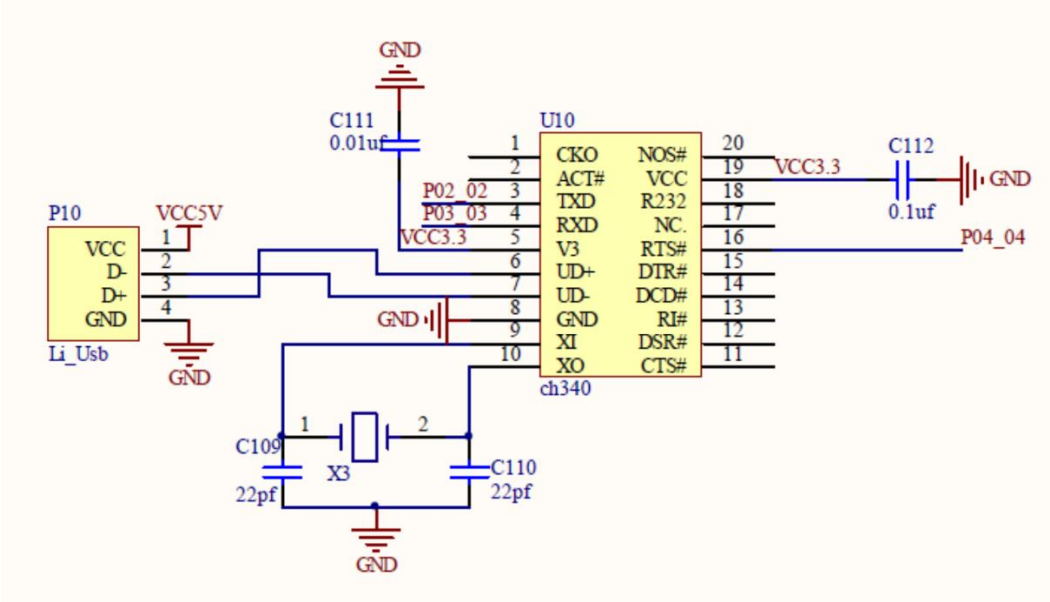
在无线传感网络中， CC2530 需要将采集到的数据发送给上位机（即 PC）处理，同时上位机需要向 CC2530 发送控制信息。这一切都离不开两者之间的信息传递。

CC2530 包括 2 个串行通信接口 USART0 与 USART1， 每个串口包括两个模式： UART（异步）模式、 SPI（同步）模式，两个 USART 具有同样的功能，可以设置在单独的 I/O 引脚。

【本篇仅涉及 UART 模式】

二、CC2530 UART 驱动编写

1. 电路图



串口芯片是 ch340，其中与软件编程密切相关的引脚是 TXD、RXD，分别连接 CC2530 的 P02，P03。

从 SOC CC2530 侧来看：P0_2、 P0_3 配置为外设功能时： P0_2 为 RX， P0_3 为 TX.

2. UART 相关寄存器

对于每个 USART 外设，相关的寄存器（x 时 USART 的编号，0 或 1）

UxCSR： USARTx 控制状态寄存器

UxUCR： USARTx UART 控制寄存器

UxGCR： USARTx 通用控制寄存器

UxBUF： USARTx UART 接受/发送数据缓冲区

UxBAUD： USARTx 波特率控制寄存器

1）U0CSR （0x86） -USART 0 控制和状态

位	名称	复位	R/W	描述
7	MODE	0	R/W	USART模式选择 0: SPI模式 1: UART模式
6	RE	0	R/W	UART接收器使能。注意在UART完全配置之前不使能接收。 0: 禁用接收器 1: 接收器使能
5	SLAVE	0	R/W	SPI主或者从模式选择 0: SPI主模式 1: SPI从模式
4	FE	0	R/W0	UART帧错误状态 0: 无帧错误检测 1: 字节收到不正确停止位级别
3	ERR	0	R/W0	UART奇偶错误状态 0: 无奇偶错误检测 1: 字节收到奇偶错误
2	RX_BYTE	0	R/W0	接收字节状态。URAT模式和SPI从模式。当读U0DBUF该位自动清除，通过写0清除它，这样有效丢弃U0DBUF中的数据。 0: 没有收到字节 1: 准备好接收字节
1	TX_BYTE	0	R/W0	传送字节状态。URAT模式和SPI主模式 0 字节没有被传送 1 写到数据缓存寄存器的最后字节被传送
0	ACTIVE	0	R	USART传送/接收主动状态、在SPI从模式下该位等于从模式选择。 0: USART空闲 1: 在传送或者接收模式USART忙碌

MODE

设置为：1 即 UART 模式

设置 RE 1：接收器使能； 0：禁用接收器

2）U0UCR (0xC4) - USART 0 UART 控制

位	名称	复位	R/W	描述
7	FLUSH	0	R0/W1	清除单元。当设置时，该事件将会立即停止当前操作并且返回单元的空闲状态。
6	FLOW	0	R/W	UART硬件流使能。用RTS和CTS引脚选择硬件流控制的使用。 0: 流控制禁止 1: 流控制使能
5	D9	0	R/W	UART奇偶校验位。当使能奇偶校验，写入D9的值决定发送的第9位的值，如果收到的第9位不匹配收到字节的奇偶校验，接收时报告ERR。 如果奇偶校验使能， 那么该位设置以下奇偶校验级别。 0: 奇校验 1: 偶校验
4	BIT9	0	R/W	UART 9位数据使能。当该位是1时，使能奇偶校验位传输（即第9位）。如果通过PARITY使能奇偶校验，第9位的内容是通过D9给出的。 0: 8位传送 1: 9位传送
3	PARITY	0	R/W	UART奇偶校验使能。除了为奇偶校验设置该位用于计算，必须使能9位模式。 0: 禁用奇偶校验 1: 奇偶校验使能
2	SPB	0	R/W	UART停止位的位数。选择要传送的停止位的位数 0: 1位停止位 1: 2位停止位
1	STOP	1	R/W	UART停止位的电平必须不同于开始位的电平 0: 停止位低电平 1: 停止位高电平
0	START	0	R/W	UART起始位电平。闲置线的极性采用选择的起始位级别的电平的相反的电平。 0: 起始位低电平 1: 起始位高电平

3）U0BUF （0xC1） - USART 0 接收/传送数据缓存

UART 收发时读写数据的缓冲区，当写入数据到该寄存器的时候，控制器会自动写入到内部并传送数据寄存器。
读取该寄存器时，数据同样来自控制器内部的数据寄存器。

位	名称	复位	R/W	描述
7:0	DATA[7:0]	0x00	R/W	USART接收和传送数据。当写这个寄存器的时候数据被写到内部，传送数据寄存器。当读取该寄存器的时候，数据来自内部读取的数据寄存器。

4）U0GCR （0xC5）、 U0BAUD （0xC2） - USART 0 波特率控制

位	名称	复位	R/W	描述
7	CPOL	0	R/W	SPI 的时钟极性 0: 负时钟极性 1: 正时钟极性
6	CPHA	0	R/W	SPI 时钟相位 0: 当 *SCK* 从 CPOL 倒置到 CPOL 时数据输出到 *MOSI*，并且当 SCK 从 CPOL 倒置到 CPOL 时数据输入抽样到 *MISO*。 1: 当 *SCK* 从 CPOL 倒置到 CPOL 时数据输出到 *MOSI*，并且当 *SCK* 从 CPOL 倒置到 CPOL 时数据输入抽样到 *MISO*。
5	ORDER	0	R/W	传送位顺序 0: LSB 先传送 1: MSB 先传送
4:0	BAUD_E[4:0]	0 0000	R/W	波特率指数值。BAUD_E 和 BAUD_M 决定了 UART 波特率 和 SPI 的主 SCK 时钟频率。

位	名称	复位	R/W	描述
7:0	BAUD_M[7:0]	0x00	R/W	波特率小数部分的值。BAUD_E和 BAUD_M决定了UART的波特率和SPI的主SCK时钟频率。

波特率计算

CC2530 的波特率有 BAUD_E 和 BAUD_M 共同决定：

$$\text{Baud Rate} = \frac{(256 + \text{BAUD\_M}) \times 2^{\text{BAUD\_E}}}{2^{28}} \times f$$

选择外部 32MHz 晶振时，具体值可参阅下表：

表 16-1　32 MHz 系统时钟常用的波特率设置

波特率 (bps)	UxBAUD.BAUD_M	UxGCR.BAUD_E	误差(%)
2400	59	6	0.14
4800	59	7	0.14
9600	59	8	0.14
14400	216	8	0.03
19200	59	9	0.14
28800	216	9	0.03
38400	59	10	0.14
57600	216	10	0.03
76800	59	11	0.14
115200	216	11	0.03
230400	216	12	0.03

由计算公式可以计算出 32MHz 系统时钟频率,，波特率 115200 对应的参数值：

```
UxBAUD.BAUD_M: U0BAUD = 216
UxGCR.BAUD_E: U0GCR = 11
```

3. 其他寄存器

除了要配置 UART 相关的 5 个寄存器，要使能 UART 功能，还有其他功能寄存器需要配置：

1）PERCFG（0xF1）- 外设控制

位	名称	复位	R/W	描述
7	-	0	R0	没有使用
6	T1CFG	0	R/W	定时器 1 的 I/O 位置 0: 备用位置 1 1: 备用位置 2
5	T3CFG	0	R/W	定时器 3 的 I/O 位置 0: 备用位置 1 1: 备用位置 2
4	T4CFG	0	R/W	定时器 4 的 I/O 位置 0: 备用位置 1 1: 备用位置 2
3:2	-	00	R0	没有使用
1	U1CFG	0	R/W	USART 1的I/O位置 0 备用位置1 1 备用位置2
0	U0CFG	0	R/W	USART 0的I/O位置 0: 备用位置1 1: 备用位置2

两个 USART 接口具有相同的功能，通过 PERCFG 寄存器可以设置两个 USART 接口对应外部 I/O 引脚的映射关系：

UART0	P0端口 备用位置1								P1端口 备用位置2							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
备用位置1					TX	RX										
备用位置2											TX	RX				
UART1	P0端口 备用位置1								P1端口 备用位置2							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
备用位置1			TX	RX												
备用位置2									TX	RX						

我们选用 p0 端口位置 1，所以设置 bit[0]为 0 即可。

2. P0SEL（0xF3）- 端口 0 功能选择

位	名称	复位	R/W	描述
7：0	SELP0_[7:0]	0x00	R/W	P0.7到 P0.0功能选择 0： 通用I / O 1： 外设功能

P0_2,P0_3 用作串口（外设功能）

设置 bit[2]、bit[3]为 1，即

```
P0SEL = 0x0c;
```

3） P2DIR （0xFF）- 端口 2 方向和端口 0 外设优先级控制

位	名称	复位	R/W	描述
7:6	PRIP0[1:0]	00	R/W	端口0外设优先级控制。当PERCFG分配给一些外设到相同引脚的时候，这些位将确定优先级。 详细优先级列表： 00： 第1优先级：USART 0 第2优先级：USART 1 第3优先级：定时器1 01： 第1优先级：USART 1 第2优先级：USART 0 第3优先级：定时器1 10： 第1优先级：定时器1通道0-1 第2优先级：USART 1 第3优先级：USART 0 第4优先级：定时器1通道2 –3 11： 第1优先级：定时器1通道2-3 第2优先级：USART 0 第3优先级：USART 1 第4优先级：定时器1通道0 –1
5	-	0	R0	不使用
4:0	DIRP2_[4:0]	0 0000	R/W	P2.4到P2.0的I/O方向 0： 输入 1： 输出

P0 优先作为 UART0，所以设置 bite[7:6]为 00，参考代码：

```
P2DIR &= ~0xC0;
```

4. CC2530 配置串口的一般步骤：

- 1、 配置 IO，使用外部设备功能。此处配置 P0_2 和 P0_3 用作串口 UART0
- 2、 配置相应串口的控制和状态寄存器。
- 3、 配置串口工作的波特率。

UART 相关寄存器具体配置如下：

```
PERCFG = 0x00; //位置 1 P0 口
P0SEL = 0x0c; //P0_2,P0_3 用作串口（外部设备功能）
P2DIR &= ~0XC0; //P0 优先作为UART0

U0CSR |= 0x80; //设置为UART 方式
U0GCR |= 11;
U0BAUD |= 216; //波特率设为 115200 根据上面表中获得的数据
UTX0IF = 0; //UART0 TX 中断标志初始置位 0
```

时钟配置如下：

```
    CLKCONCMD &= ~0x40;                     //设置系统时钟源为 32MHZ 晶振
    while(CLKCONSTA & 0x40);                //等待晶振稳定为 32M
    CLKCONCMD &= ~0x47;                     //TICHSPD128 分频， CLKSPD 不分频
```

1. USART 发送

UxBUF 寄存器是双缓冲的。
发送数据写入到 U0DBUF 后，控制器就自动将数据写入到内部，并传输数据寄存器，同时置 UTX0IF 为 1，
如果传输完毕，UTX0IF 会被置 0，
所以每次写入数据到 U0DBUF 后就循环监测 UTX0IF 是否为 0；
UTX0IF 定义在寄存器 IRCON2（0xE8）中。

1	UTX0IF	0	R/W	USART 0 TX中断标志 0: 无中断未决 1: 中断未决

参考发送代码：

```
void UartSendString(char *Data, int len)
{
    uint i;

    for(i=0; i<len; i++)
    {
        U0DBUF = *Data++;
        while(UTX0IF == 0);
        UTX0IF = 0;
```

```
    }
}
```

2. USART 接收

当 USART 检测出有效起始位时，收到的字节就传入到接收寄存器。
每当 USART 收到 1 个数据后，就会产生一个中断，所以必须在中断函数中清除中断标志 URX0IF，同时读走缓冲区 U0DBUF 中数据。
URX0I 定义在寄存器 TCON (0x88):

3	URX0IF	0	R/WH0	USART 0 RX中断标志。当USART0中断发生时设为1且CPU指向中断向量例程时清除。 0: 无中断未决 1: 中断未决

清中断并读走代码:

```
#pragma vector = URX0_VECTOR
__interrupt void UART0_ISR(void)
{
    URX0IF = 0;         // 清中断标志
    RxBuf = U0DBUF;
}
```

5. 代码实现

下面代码主要功能： 通过串口向 CC2530 发送字符串，并以#结尾，然后 CC2530 会将该字符串再回传给 pc。
核心代码如下：

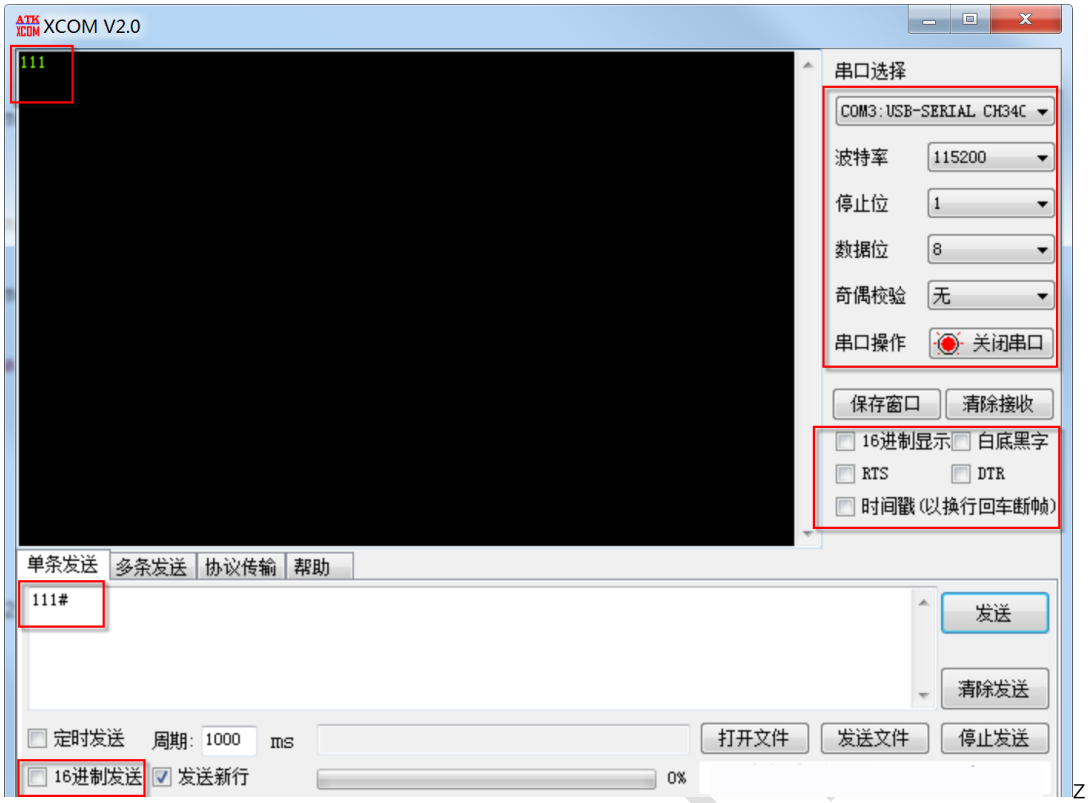
```
#define UART0_RX    1
#define UART0_TX    2
#define SIZE        51

char RxBuf;
char UartState;
uchar count;
char RxData[SIZE];          //存储发送字符串
void InitUart(void)
{
    PERCFG = 0x00;              //外设控制寄存器 USART 0 的IO 位置:0 为P0 口位置1
    P0SEL = 0x0c;               //P0_2,P0_3 用作串口（外设功能）
    P2DIR &= ~0xC0;             //P0 优先作为UART0
```

```
    U0CSR |= 0x80;            //设置为UART 方式
    U0GCR |= 11;
    U0BAUD |= 216;            //波特率设为115200
    UTX0IF = 0;               //UART0 TX 中断标志初始置位0
    U0CSR |= 0x40;            //允许接收
    IEN0 |= 0x84;             //开总中断允许接收中断
}
void UartSendString(char *Data, int len)
{
    uint i;

    for(i=0; i<len; i++)
    {
        U0DBUF = *Data++;
        while(UTX0IF == 0);
        UTX0IF = 0;
    }
}

#pragma vector = URX0_VECTOR
__interrupt void UART0_ISR(void)
{
    URX0IF = 0;       // 清中断标志
    RxBuf = U0DBUF;
}
void main(void)
{
    CLKCONCMD &= ~0x40;                       //设置系统时钟源为32MHZ 晶振
    while(CLKCONSTA & 0x40);                  //等待晶振稳定为32M
    CLKCONCMD &= ~0x47;                       //设置系统主时钟频率为32MHZ

    InitUart();                               //调用串口初始化函数
    UartState = UART0_RX;                     //串口0 默认处于接收模式
    memset(RxData, 0, SIZE);

    while(1)
    {
        if(UartState == UART0_RX)             //接收状态
        {
            if(RxBuf != 0)
            {
                if((RxBuf != '#')&&(count < 50))//以'#'为结束符,一次最多接收50 个字
符
                    RxData[count++] = RxBuf;
```

```
                else
                {
                    if(count >= 50)              //判断数据合法性，防止溢出
                    {
                        count = 0;               //计数清0
                        memset(RxData, 0, SIZE);//清空接收缓冲区
                    }
                    else
                        UartState = UART0_TX;  //进入发送状态
                }
                RxBuf  = 0;
            }
        }

        if(UartState == UART0_TX)               //发送状态
        {
             &= ~0x40;                          //禁止接收
            UartSendString(RxData, count);      //发送已记录的字符串。
            U0CSR |= 0x40;                      //允许接收
            UartState = UART0_RX;               //恢复到接收状态
            count = 0;                          //计数清0
            memset(RxData, 0, SIZE);            //清空接收缓冲区
        }
    }
}
```

实验步骤： 烧入代码后，usb 线插入到 cc2530 的 mini USB，windows 会是被该串口，笔者为 COM3

实际操作结果如下：


XCOM V2.0
111
串口选择
COM3:USB-SERIAL CH34C
波特率
115200
停止位
1
数据位
8
奇偶校验
无
串口操作
关闭串口
保存窗口
清除接收
16进制显示
白底黑字
RTS
DTR
时间戳(以换行回车断帧)
单条发送
多条发送
协议传输
帮助
111#
发送
清除发送
定时发送
周期:
1000
ms
打开文件
发送文件
停止发送
16进制发送
发送新行
0%