2.5 串口通讯

前言：

无论学习哪款 MUC 串口对于我们进行实验调试都是非常方便实用的，我们可以把程序中涉及的某些中间量或者其他程序状态信息打印出来显示在电脑上进行调试，许多 MUC 和 PC 机通信都是通过串口来进行的。下面一起来学习 zigbee 的串口实验。

实验现象：实验将使用 WeBee 开发板实验 3 个串口功能。发送、收发、控制 LED。

实验讲解：我们先来看看 WeBee 底板的 USB 转串口部分电路原理图：如图 2.9 所示。

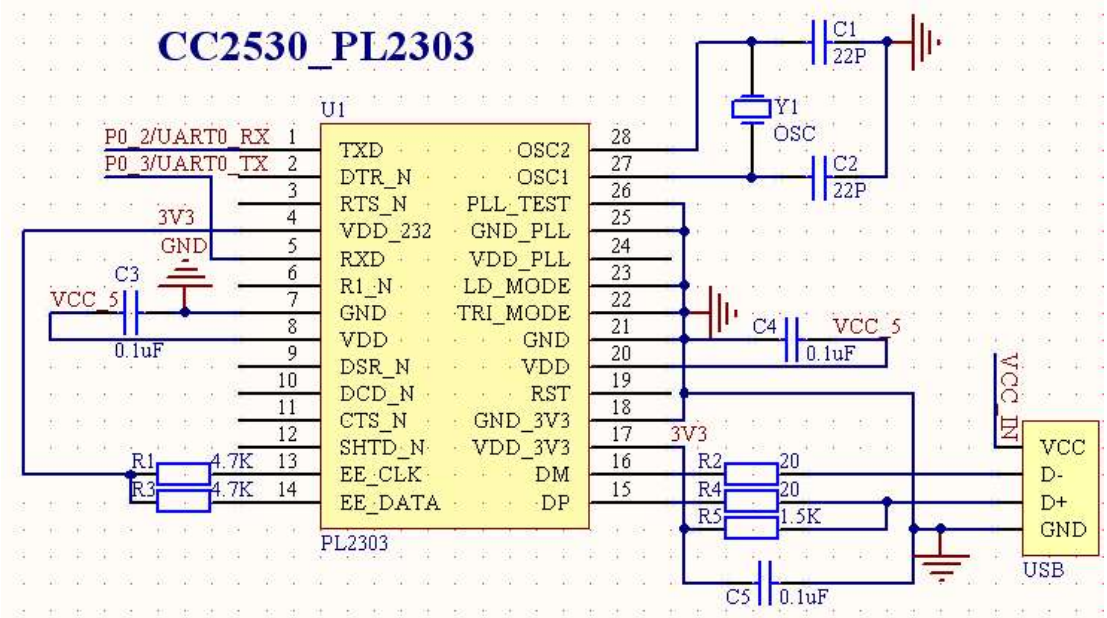

图 2.9 PL2303 USB 转串口电路

2.5.1 串口发送(HELLO WEBEE)

查看 CC2530 的 datasheet 可知：

UART0 对应的外部设备 IO 引脚关系为：P0_2------RX

P0_3------TX

UART1 对应的外部设备 IO 引脚关系为：P0_5------RX

P0_4------TX

在 CC2530 中，USART0 和 USART1 是串行通信接口，它们能够分别运行于异步 USART 模式或者同步 SPI 模式。两个 USART 的功能是一样的，可以通过设置在单独的 IO 引脚上。

USART 模式的操作具有下列特点：

1、8 位或者 9 位负载数据

2、奇校验、偶校验或者无奇偶校验

3、配置起始位和停止位电平

4、配置 LSB 或者 MSB 首先传送

5、独立收发中断

6、独立收发 DMA 触发

注：在本次实验中，我们用到的是 UART0。

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。此处配置 UART0 的工作寄存器

3、 配置串口工作的波特率。此处配置为波特率为 115200

本次实验串口相关的寄存器或者标志位有：U0CSR、U0GCR、U0BAUD、U0DBUF、UTX0IF。各寄存器功能如下表所示：　（详细参考 CC2530 datasheet.pdf）

表 2.6

U0CSR（UART0 控制和状态寄存器）	Bit7： MODE	0：SPI 模式
		1：UART 模式
	Bit6： RE	0：接收器禁止
		1：接收器使能
	Bit5： SLAVE	0：SPI 主模式
		1：SPI 从模式
	Bit4： FE	0：没有检测出帧错误
		1：收到字节停止位电平出错
	Bit3： ERR	0：没有检测出奇偶检验出错
		1：收到字节奇偶检验出错
	Bit2： RX_BYTE	0：没有收到字节
		1：收到字节就绪
	Bit1： TX_BYTE	0：没有发送字节
		1 写到数据缓冲区寄存器的最后字节已经发送
	Bit0： ACTIVE	0：USART 空闲
		1：USART 忙
U0GCR（UART0 通用控制寄存器）	Bit7： CPOL	0：SPI 负时钟极性
		1：SPI 正时钟极性
	Bit6： CPHA	0：当来自 CPOL 的 SCK 反相之后又返回 CPOL 时，数据输出到 MOSI；当来自 CPOL 的 SCK 返回 CPOL 反相时，输入数据采样到 MISO
		1:当来自 CPOL 的 SCK 返回 CPOL 反相时，数据输出到 M OSI ；当来自 CPOL 的 SCK 反相之后又返回 CPOL 时，输入数据采样到 MISO
	Bit5： ORDER	0：LSB 先传送
		1：MSB 先传送

	Bit[4-0]：BAUD_E	波特率指数值 BAUD_E 连同 BAUD_M 一起决定了 UART 的波特率
U0BAUD（UART0 波特率控制寄存器）	Bit[7-0]：BAUD_M	波特率尾数值 BAUD_M 连同 BAUD_E 一起决定了 UART 的波特率
U0DBUF （ UART0 收发数据缓冲区）		串口发送/接收数据缓冲区
UTX0IF（发送中断标志）	中断标志 5 IRCON2 的 Bit1	0：中断未挂起
		1：中断挂起

串口的波特率设置可以从 CC2530 的 datasheet 中查得波特率由下式求得：

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

本次实验设置波特率为 115200bps,具体的参数设置如下：

表 16-1 32MHz 系统时钟的常用波特率设置

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

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
UTX0IF  = 0;            //UART0 TX 中断标志初始置位 0
```

串口发送函数请参考下面源程序：

源程序代码（全）

```
/*************************************
  描述：在串口调试助手上可以看到不停地
        收到 CC2530 发过来的：HELLO WEBEE
        波特率：115200bps
*************************************/
#include <ioCC2530.h>
#include <string.h>

#define   uint   unsigned int
#define   uchar unsigned char

//定义 LED 的端口
#define LED1 P1_0
#define LED2 P1_1

//函数声明
```

```
void Delay_ms(uint);
void initUART(void);
void UartSend_String(char *Data,int len);

char Txdata[14];      //存放”HELLO WEBEE     “共 14 个字符串
/*****************************************************************
延时函数
*****************************************************************/
void Delay_ms(uint n)
{
        uint I,j;
        for(i=0;i<n;i++)
        {
        for(j=0;j<1774;j++);
        }
}

void IO_Init()
{
P1DIR = 0x01;                 //P1_0IO 方向输出
LED1 = 1;                     //关 LED
}

/*****************************************************************
    串口初始化函数
*****************************************************************/
void InitUART(void)
{
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
UTX0IF = 0;                  //UART0 TX 中断标志初始置位 0
}
/************************************************************
串口发送字符串函数
************************************************************/
void UartSend_String(char *Data,int len)
{
  int j;
  for(j=0;j<len;j++)
  {
U0DBUF = *Data++;
while(UTX0IF == 0);
UTX0IF = 0;
  }
}
/************************************************************
主函数
************************************************************/
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
```

```
IO_Init();
InitUART();
strcpy(Txdata,”HELLO WEBEE     “);          //将发送内容 copy 到 Txdata;
while(1)
        {
            UartSend_String(Txdata,sizeof(“HELLO WEBEE     “));   //串口发送
数据
            Delay_ms(500);                                          //延时
            LED1=!LED1;                                             //标志发送状态
        }
}
```

实验 1 图片：

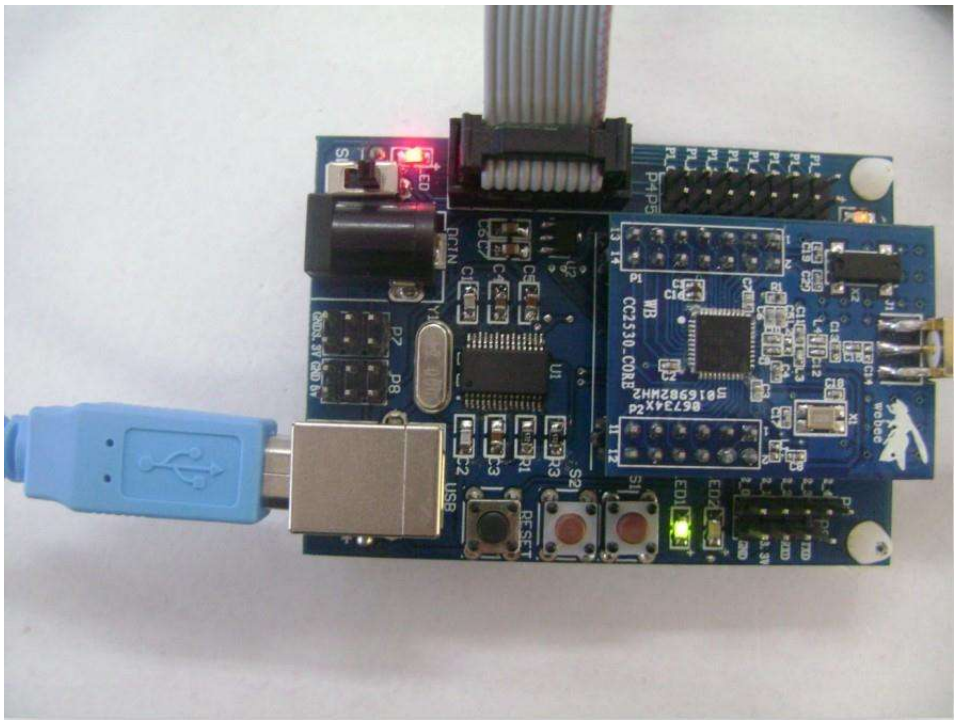

图 2.10　USB 转串口连接方法

图 2.11 上位机接收到发来的“HELLO WEBEE”

2.5.2 串口接收和发送(send & receive)

寄存器配置请参考上方实验 1 的表格。实验 1 较实验 1 增加了串口接收功能，故寄存器配置有所改变，如下。

```
CLKCONCMD &= ~0x40;            // 设置系统时钟源为 32MHZ 晶振
while(CLKCONSTA & 0x40);       // 等待晶振稳定
CLKCONCMD &= ~0x47;            // 设置系统主时钟频率为 32MHZ

PERCFG  = 0x00;              //位置 1 P0 口
POSEL   = 0x3c;              //P0_2,P0_3,P0_4,P0_5 用作串口,第二功能
P2DIR  &= ~0XC0;             //P0 优先作为 UART0 ，优先级

U0CSR  |= 0x80;              //UART 方式
U0GCR  |= 11;                //U0GCR 与 U0BAUD 配合
U0BAUD |= 216;             // 波特率设为 115200
UTX0IF  = 0;               //UART0 TX 中断标志初始置位 1 （收发时候）
U0CSR  |= 0X40;            //允许接收
IEN0   |= 0x84;            // 开总中断，接收中断
```

源程序代码（部分） 请读者自行分析

```
/*************************************
程序描述：例以 abc#方式发送，#为结束符，
          返回 abc。波特率：115200bps
*************************************/
...
...
```

```
/*************************************************************
        串口初始化函数
*************************************************************/
    void InitUart()
    {
    CLKCONCMD &= ~0x40;              // 设置系统时钟源为 32MHZ 晶振
    while(CLKCONSTA & 0x40);         // 等待晶振稳定
    CLKCONCMD &= ~0x47;              // 设置系统主时钟频率为 32MHZ

    PERCFG = 0x00;                  //位置 1 P0 口
    P0SEL = 0x3c;                   //P0_2,P0_3,P0_4,P0_5 用作串口,第二功能
    P2DIR &= ~0XC0;                 //P0 优先作为 UART0 ，优先级

    U0CSR |= 0x80;                  //UART 方式
    U0GCR |= 11;              //U0GCR 与 U0BAUD 配合
    U0BAUD |= 216;            // 波特率设为 115200
    UTX0IF = 0;               //UART0 TX 中断标志初始置位 1   （收发时候）
    U0CSR |= 0X40;            //允许接收
    IEN0 |= 0x84;             // 开总中断，接收中断
    }

/*************************************************************
    串口发送字符串函数
*************************************************************/
  void Uart_Send_String(char *Data,int len)
  {
    {
      int j;
      for(j=0;j<len;j++)
```

```
  {
    U0DBUF = *Data++;
    while(UTX0IF == 0);    //发送完成标志位
    UTX0IF = 0;
   }
  }
}

/*****************************
    主函数
*****************************/
void main(void)
{
  InitLed();                //调用初始化函数
  InitUart();
  while(1)
  {
     if(RXTXflag == 1)      //接收状态
     {
        LED1=1;             //接收状态指示
        if( temp != 0)
        {
       if((temp!=' #' )&&(datanumber<50)) //' #' 被定义为结束字
                                          符，最多能接收 50 个字
                                          符
           Rxdata[datanumber++] = temp;
           else
           {
             RXTXflag = 3;                    //进入发送状态
```

```
                LED1=0;                                    //关指示灯
              }
              temp  = 0;
            }
          }
          if(RXTXflag == 3)      //发送状态
          {
           LED2= 1;
           U0CSR &= ~0x40;       //禁止接收
           Uart_Send_String(Rxdata,datanumber); //发送已记录的字符串。
           U0CSR |= 0x40;        //允许接收
           RXTXflag = 1;         // 恢复到接收状态
           datanumber = 0;       //指针归 0
           LED2 = 0;             //关发送指示
          }
    }
    }
/*****************************************************************
    串口接收一个字符：一旦有数据从串口传至 CC2530，则进入中断，将接收到的数据赋值给变量 temp.
*****************************************************************/
    #pragma vector = URX0_VECTOR
      __interrupt void UART0_ISR(void)
     {
      URX0IF = 0;     // 清中断标志
      temp = U0DBUF;
     }
```

实验 2 图片：

图 2.12　发送：I LOVE WeBee!#　接收到：I LOVE WeBee

2.5.3　UART0-控制 LED

发送和接收函数均和实验 2 相同，这里我们重点分析主函数判断代码：

```
/*****************************************
程序描述：依次发送 L1# L2# 指令分别控制
          LED1、LED2 亮灭，波特率：115200bps
*****************************************/
...
...
/*****************************
//主函数
*****************************/
void main(void)
{
```

```
InitLed();                    //调用初始化函数
InitUart();
while(1)
{
   if(RXTXflag == 1)          //接收状态
   {
      if( temp != 0)
      {
     if((temp!=' #' )&&(datanumber<3)) //' #' 被定义为结束字符,
                                       最//多能接收 50 个字符
         Rxdata[datanumber++] = temp;
         else
         {
           RXTXflag = 3;                  //进入发送状态
         }
          temp  = 0;
      }
   }

   if(RXTXflag == 3)                    //检测接收到的数据
   {
    if(Rxdata[0]==' L' )
  switch(Rxdata[1]-48)              //很重要, ASICC 码转成数字,判
                                      断 L 后面第一个数
     {
   case 1:                     //如果是 L1
     {
       LED1=~LED1;               //低电平点亮
       break;
```

```
        }
      case 2:                    //如果是 L2
        {
          LED2=~LED2;
          break;
        }
       }
       RXTXflag = 1;
       datanumber = 0;           //指针归 0
      }
}
}
```

实验 3 图片：

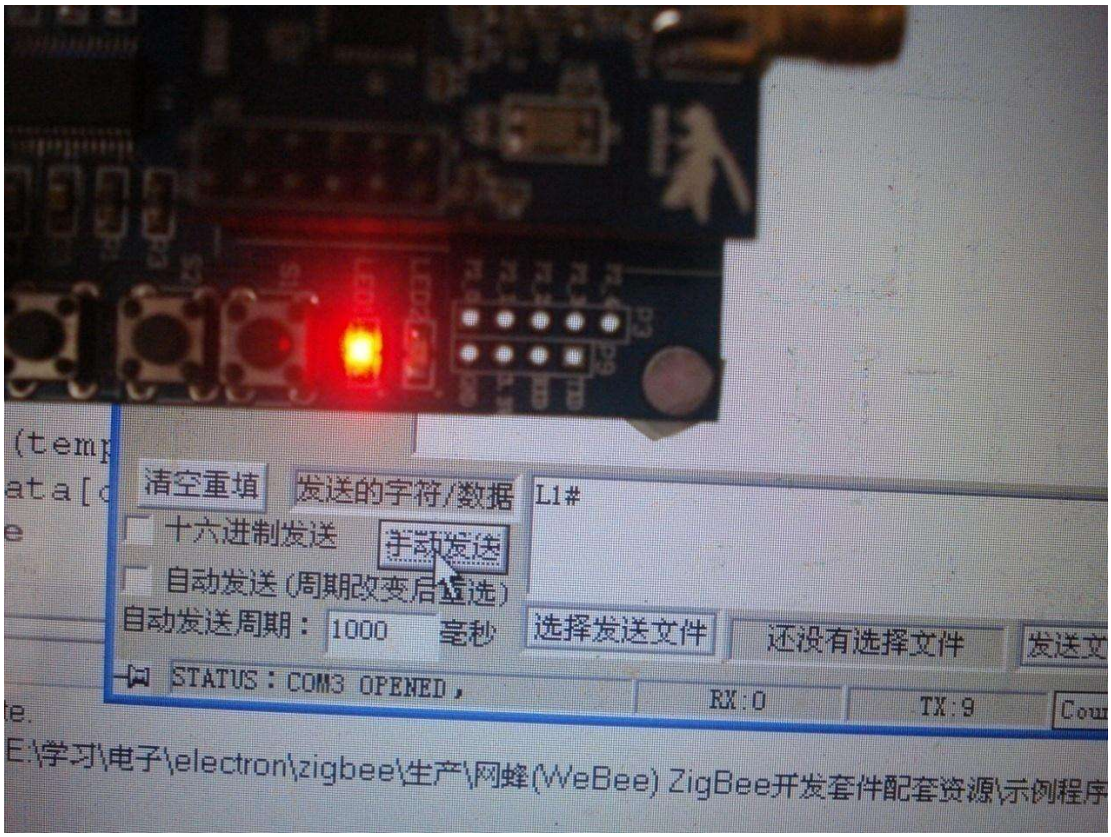

图 2.13 依次发送 L1#和 L2#