项目5

串口控制

5.1 项目任务和指标

本项目将完成串口收发数据和串口控制 LED 灯等任务。

通过本项目的实施，读者应掌握串行通信接口的概念、串行通信接口寄存器的相关概念和方法，设置串行通信接口寄存器波特率的方法，重点是掌握 UART 接收的具体应用。

5.2 项目的预备知识

5.2.1 串行通信接口

CC2530 有两个串行通信接口 USART0 和 USART1，它们能够分别运行于异步模式(UART)或者同步模式(SPI)。当寄存器位 UxCSR. MODE 设置为 1 时，就选择了 UART 模式，这里 x 是 USART 的编号，数值为 0 或者 1。当两个 USART 具有同样的功能，可以设置单独的 I/O 引脚，一旦硬件电路确定下来，再进行程序设计时，需要按照硬件电路来设置 USART 的 I/O 引脚。寄存器位 PERCFGU0CFG 选择是否使用备用位置 1 或备用位置 2。在 UART 模式中，可以使用双线连接方式(含有引脚 RXD、TXD)或者四线连接方式(含有引脚 PXD、TXD、RTS 和 CTS)，其中 RTS 和 CTS 引脚用于硬件流量控制。UART 模式的操作具有以下特点：

(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。

（7）奇偶校验和帧校验出错状态。

UART 模式提供全双工传送，接收器中的位同步不影响发送功能。传送一个 UART 字节包含 1 个起始位、8 个数据位、1 个作为可选项的第 9 位数据或者奇偶校验位再加上 1 个或 2 个停止位。注意，虽然真实的数据包含 8 位或者 9 位，但是，数据传送只涉及 1 字节。

5.2.2 串行通信接口寄存器

UART 操作由 USART 控制和状态寄存器 UxCSR 以及 UART 控制寄存器 UxUCR 来控制。寄存器 UxBAUD 用于设置波特率，寄存器 UxBUF 是 USART 接收/传送数据缓存，这里的 x 是 USART 的编号，其数值为 0 或者 1。表 5.1～表 5.5 为 USART0 的相关寄存器。

表 5.1 U0CSR——USART0 控制和状态寄存器

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	USART 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。UART 模式和 SPI 从模式。当读 U0DBUF 该位自动清除，通过写 0 清除它，这样能有效丢弃 U0DBUF 中的数据 0：没有收到字节 1：准备好接收字节

续表

位	名称	复位	R/W	描　　述
1	TX_BYTE	0	R/W0	传送字节状态。UART 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

表 5.2　U0UCR——USART0 UART 控制寄存器

位	名称	复位	R/W	描　　述
7	FLUSH	0	RO/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。当使能奇偶检验，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	UART9 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

续表

位	名称	复位	R/W	描　　述
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

表 5.3　U0GCR(0xC5)——USART0 通用控制寄存器

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 的时钟极性 0：CPOL 颠倒后传送 CPOL 1：CPOL 到 CPOL 后颠倒
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

表 5.4　U0BUF——USART0 接收/传送数据缓存寄存器

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

表 5.5　U0BAUD——USART0 波特率控制寄存器

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

5.2.3　设置串行通信接口寄存器波特率

当运行状态 UART 模式时，内部的波特率发生器设置 UART 波特率，由寄存器

UxBAUD. BAUD_M[7：0]和 UxGCR. BAUD_E[4：0]定义波特率，如表 5.6 所示。

表 5.6　32MHz 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
14 400	216	8	0.03
19 200	59	9	0.14
28 800	216	9	0.03
38 400	59	10	0.14
57 600	216	10	0.03
76 800	59	11	0.14
115 200	216	11	0.03
230 400	216	12	0.03

5.2.4　UART 接收

当 1 写入 UxCSR. RE 位时，在 UART 上数据接收就开始了。然后 UART 会在输入引脚 RXDx 中寻找有效起始位，并且设置 UxCSR. ACTIVE 位为 1。当检测出有效起始位时，收到的字节就传入到接收寄存器，通过寄存器 UxBUF 提供收到的数据字节。当 UxBUF 读出时，xCSR. RX_BYTE 位由硬件清 0。

5.3　项目实施

5.3.1　任务 1：串口收发数据

1. 项目环境

(1) 硬件：ZigBee(CC2530)模块、ZigBee 下载调试板、USB 仿真器和 PC。

(2) 软件：IAR Embedded Workbench for MCS-51。

2. 项目原理

1) 硬件接口原理

ZigBee(CC2530)模块 LED 硬件接口如图 5.1 所示。

ZigBee(CC2530)模块硬件上设计有两个 LED 灯，用来编程调试使用。分别连接 CC2530 的 P1_0、P1_1 两个 I/O 引脚。从原理图上可以看出，两个 LED 灯共阳极，当 P1_0、P1_1 引脚为低电平时，LED 灯点亮。

2) CC2530 I/O 相关寄存器

P1 寄存器的相关信息如表 5.7 所示，P1DIR 寄存器的相关信息如表 5.8 所示。

图 5.1　LED 硬件接口

表 5.7　P1 寄存器

位	名称	复位	R/W	描　　述
7：0	P1_[7：0]	0xFF	R/W	端口 1 是通用 I/O 端口。位寻址从 SFR 开始。CPU 内部寄存器是可读的，但是不可写，从 XDATA(0x7090)开始

表 5.8　寄存器 P1DIR

位	名称	复位	R/W	描　　述
7：0	DIRP1_[7：0]	0x00	R/W	P1_7～P1_0 的 I/O 方向 0：输入 1：输出

表 5.7 和表 5.8 中列出了关于 CC2530 处理器的 P1 I/O 相关寄存器，其中只用到了 P1 和 P1DIR 两个寄存器的设置，P1 寄存器为可读写的数据寄存器，P1DIR 为 I/O 选择寄存器，其他 I/O 寄存器的功能，使用默认配置。

CLKCONCMD 和 CLKCONSTA 寄存器的相关信息如表 5.9 和表 5.10 所示。

表 5.9　CLKCONCMD 时钟控制寄存器

位	名称	复位	R/W	描　　述
7	OSC32K	1	W	32kHz 时钟源选择 0：32kHz 晶振 1：32kHz RC 振荡
6	OSC	1	W	主时钟源选择 0：32MHz 晶振 1：16MHz RC 振荡
5：3	TICKSPD[2：0]	001	W	定时器计数时钟分频(该时钟频不大于 OSC 决定频率) 000：32MHz 001：16MHz 010：8MHz 011：4MHz 100：2MHz 101：1MHz 110：0.5MHz 111：0.25MHz

续表

位	名称	复位	R/W	描　　述
2：0	CLKSPD	001	W	时钟速率，不能高于系统时钟 000：32MHz 001：16MHz 010：8MHz 011：4MHz 100：2MHz 101：1MHz 110：500kHz 111：250kHz

表 5.10　CLKCONSTA 时钟状态寄存器

位	名称	复位	R/W	描　　述
7	OSC32K	1	R	32kHz 时钟源选择 0：32kHz 晶振 1：32kHz RC 振荡
6	OSC	1	R	主时钟源选择 0：32MHz 晶振 1：16MHz RC 振荡
5：3	TICKSPD[2：0]	001	R	定时器计数时钟分频（该时钟频不大于 OSC 决定频率） 000：32MHz 001：16MHz 010：8MHz 011：4MHz 100：2MHz 101：1MHz 110：0.5MHz 111：0.25MHz
2：0	CLKSPD	001	R	时钟速率，不能高于系统时钟 000：32MHz 001：16MHz 010：8MHz 011：4MHz 100：2MHz 101：1MHz 110：500kHz 111：250kHz

SLEEPCMD 控制寄存器的相关信息如表 5.11 所示，PERCFG 寄存器的相关信息如表 5.12 所示。

表 5.11 SLEEPCMD 睡眠模式控制寄存器

位	名称	复位	R/W	描　　述
7	—	0	R	预留
6	XOSC_STB	0	W	低速时钟状态 0：没有打开或者不稳定 1：打开且稳定
5	HFRC_STB	0	W	主时钟状态 0：没有打开或者不稳定 1：打开且稳定
4：3	RST[1：0]	xx	W	最后一次复位指示 00：上电复位 01：外部复位 10：看门狗复位
2	OSC_PD	0	W	节能控制，OSC 状态改变的时候硬件清 0 0：不关闭无用时钟 1：关闭无用时钟
1：0	MODE[1：0]	0	W	功能模式选择 00：PM0 01：PM1 10：PM2 11：PM3

表 5.12 PERCFG 外设控制寄存器

位	名称	复位	R/W	描　　述
7	—	0	R	预留
6	T1CFG	0	R/W	T1 I/O 位置选择 0：位置 1 1：位置 2
5	T3CFG	0	R/W	T3 I/O 位置选择 0：位置 1 1：位置 2
4	T4CFG	0	R/W	T4 I/O 位置选择 0：位置 1 1：位置 2
3：2	—	00	RO	预留
1	U1CFG	0	R/W	串口 1 位置选择 0：位置 1 1：位置 2
0	U0CFG	0	R/W	串口 0 位置选择 0：位置 1 1：位置 2

U0CSR 寄存器的相关信息如表 5.13 所示，U0GCR 寄存器的相关信息如表 5.14 所示，U0BUF 寄存器的相关信息如表 5.15 所示，U0BAUD 寄存器的相关信息如表 5.16 所示。

表 5.13　U0CSR（串口 0 控制和状态寄存器）

位	名称	复位	R/W	描　　述
7	MODE	0	R/W	串口模式选择 0：SPI 模式 1：UART 模式
6	RE	0	R/W	接收使能 0：关闭接收 1：允许接收
5	SLAVE	0	R/W	SPI 主从选择 0：SPI 主 1：SPI 从
4	FE	0	R/W	串口帧错误状态 0：没有帧错误 1：出现帧错误
3	ERR	0	R/W	串口校验结果 0：没有校验错误 1：字节校验出错
2	RX_BYTE	0	R/W	接收状态 0：没有接收到数据 1：接收到 1 字节数据
1	TX_BYTE	0	R/W	发送状态 0：没有发送 1：最后一次写入 U0BUF 的数据已经发送
0	ACTIVE	0	R	串口忙标志 0：串口闲 1：串口忙

表 5.14　U0GCR（串口 0 常规控制寄存器）

位	名称	复位	R/W	描　　述
7	CPOL	0	R/W	SPI 时钟极性 0：低电平空闲 1：高电平空闲
6	CPHA	0	R/W	SPI 时钟相位 0：由 CPOL 跳向非 CPOL 时采样，由非 CPOL 跳向 CPOL 时输出 1：由非 CPOL 跳向 CPOL 时采样，由 CPOL 跳向非 CPOL 时输出

续表

位	名称	复位	R/W	描　　述
5	ORDER	0	R/W	传输位序 0：低位在先 1：高位在先
4：0	BAUD_E[4：0]	0x00	R/W	波特率指数值，BAUD_M 决定波特率

表 5.15　U0BUF（串口 0 收发缓冲器）

位	名称	复位	R/W	描　　述
7：0	DATA[7：0]	0x00	R/W	UART0 收发寄存器

表 5.16　U0BAUD（串口 0 波特率控制器）

位	名称	复位	R/W	描　　述
7：0	BAUD_M[7：0]	0x00	R/W	波特率尾数，与 BAUD_E 决定波特率

表 5.9～表 5.16 中列举了和 CC2530 处理器串口操作相关的寄存器，其中包括：CLKCONCMD 控制寄存器，用来控制系统时钟源；SLEEPCMD 和 SLEEPSTA 寄存器，用来控制各种时钟源的开关和状态；PERCFG 寄存器为外设功能控制寄存器，用来控制外设功能模式；U0CSR、U0GCR、U0BUF、U0BAUD 等为串口相关寄存器。

3. 软件设计

```
#include <iocc2530.h>
#include <stdio.h>
#include "./uart/hal_uart.h"
#define uchar unsigned char
#define uint unsigned int
#define uint8 uchar
#define uint16 uint
#define TRUE 1
#define FALSE 0
//定义控制 LED 灯的端口
#define LED1 P1_0                          //定义 LED1 为 P1_0 端口控制
#define LED2 P1_1                          //定义 LED2 为 P1_1 端口控制
uchar temp;
/*****************************
//延时函数
*****************************/
void Delay(uint n)
{
    uint i,t;
    for(i = 0;i<5;i++)
    for(t = 0;t<n;t++);
}
```

```
void InitLed(void)
{
    P1DIR | = 0x03;                          //P1_0、P1_1 定义为输出
    LED1 = 1;                                //LED1 灯熄灭
    LED2 = 1;                                //LED2 灯熄灭
}
void main(void)
{
    char receive_buf[30];
    uchar counter = 0;
    uchar RT_flag = 1;
    InitUart();                              // 波特率为 57600bps
    InitLed();
    while(1)
    {
        if(RT_flag == 1)                     //接收
        {
          LED2 = 0;                          //接收状态指示
          if( temp != 0)
          {
           if((temp!= '\r')&&(counter < 30))  //'\r'回车键为结束字符
                                              //最多能接收 30 个字符
              {
                receive_buf[counter++] = temp;
              }
              else
              {
                RT_flag = 3;                  //进入发送状态
              }
              if(counter == 30) RT_flag = 3;
              temp = 0;
          }
        }
        if(RT_flag == 3)                      //发送
        {
            LED2 = 1;                         //关 LED2
            LED1 = 0;                         //发送状态指示
            U0CSR &= ~0x40;                   //禁止接收
            receive_buf[counter] = '\0';
            prints(receive_buf);
            prints("\r\n");
            U0CSR | = 0x40;                   //允许接收
            RT_flag = 1;                      //恢复到接收状态
            counter = 0;                      //指针归 0
            LED1 = 1;                         //关发送指示
        }
    }
}
```

```
/*********************************************************************
* 函数功能 : 串口接收一个字符
* 入口参数 : 无
* 返 回 值 : 无
* 说 明 : 接收完成后打开接收
*********************************************************************/
#pragma vector = URX0_VECTOR
__interrupt void UART0_ISR(void)
{
    URX0IF = 0;                                    //清中断标志
    temp = U0DBUF;
}
```

程序通过配置 CC2530 处理器的串口相关控制寄存器来设置串口 0 的工作模式为串口模式，波特率为 57600bps，使用中断方式接收串口数据并向串口输出。

4. 实施步骤

（1）使用 ZigBee Debuger USB 仿真器连接 PC 和 ZigBee（CC2530）模块，打开 ZigBee 模块开关供电。将系统配套串口线一端连接 PC，另一端连接 ZigBee 调试板的串口上。

（2）启动 IAR 开发环境，新建工程。

（3）在 IAR 开发环境中编译、运行、调试程序。

（4）使用 PC 自带的超级终端（注意：字符串必须以回车键结束或输入字符串长度超过 30 个字符，才会显示）。连接串口，将超级终端设置为串口波特率 57600bps、8 位、无奇偶校验，无硬件流模式，当向串口终端输入数据并按回车结束时，将在超级终端看到串口输入的数据。

5.3.2 任务 2：串口控制 LED

1. 项目环境

（1）硬件：ZigBee（CC2530）模块、ZigBee 下载调试板、USB 仿真器和 PC。

（2）软件：IAR Embedded Workbench for MCS-51。

2. 项目原理

1）硬件接口原理

ZigBee（CC2530）模块 LED 硬件接口如图 5.2 所示。

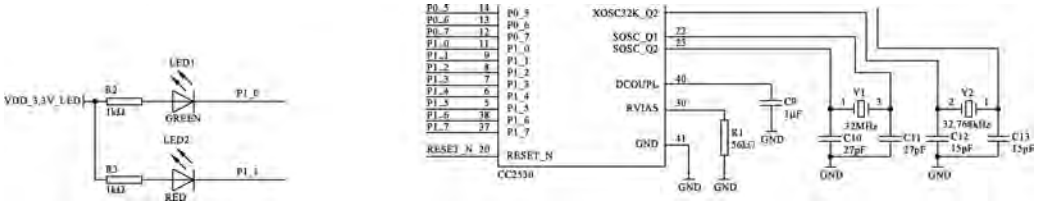

图 5.2 LED 硬件接口

ZigBee(CC2530)模块硬件上设计有两个 LED 灯，用来编程调试使用。分别连接 CC2530 的 P1_0、P1_1 两个 I/O 引脚。从原理图上可以看出，两个 LED 灯共阳极，当 P1_0、P1_1 引脚为低电平时，LED 灯点亮。

2) CC2530 I/O 相关寄存器

P1 寄存器的相关信息如表 5.17 所示，P1DIR 寄存器的相关信息如表 5.18 所示。

表 5.17　P1 寄存器

位	名称	复位	R/W	描　　述
7：0	P1_[7：0]	0xFF	R/W	端口 1 是通用 I/O 端口。位寻址从 SFR 开始。CPU 内部寄存器是可读的，但是不可写，从 XDATA(0x7090)开始

表 5.18　寄存器 P1DIR

位	名称	复位	R/W	描　　述
7：0	DIRP1_[7：0]	0x00	R/W	P1_7～P1_0 的 I/O 方向 0：输入 1：输出

表 5.17 和表 5.18 中列出了关于 CC2530 处理器的 P1 I/O 相关寄存器，其中只用到了 P1 和 P1DIR 两个寄存器的设置，P1 寄存器为可读写的数据寄存器，P1DIR 为 I/O 选择寄存器，其他 I/O 寄存器的功能使用默认配置。

CLKCONCMD 和 CLKCONSTA 寄存器的相关信息如表 5.19 和表 5.20 所示。

表 5.19　CLKCONCMD 时钟控制寄存器

位	名称	复位	R/W	描　　述
7	OSC32K	1	W	32kHz 时钟源选择 0：32kHz 晶振 1：32kHz RC 振荡
6	OSC	1	W	主时钟源选择 0：32MHz 晶振 1：16MHz RC 振荡
5：3	TICKSPD[2：0]	001	W	定时器计数时钟分频(该时钟频不大于 OSC 决定频率) 000：32MHz 001：16MHz 010：8MHz 011：4MHz 100：2MHz 101：1MHz 110：0.5MHz 111：0.25MHz

续表

位	名称	复位	R/W	描　　述
2：0	CLKSPD	001	W	时钟速率，不能高于系统时钟 000：32MHz 001：16MHz 010：8MHz 011：4MHz 100：2MHz 101：1MHz 110：500kHz 111：250kHz

表 5.20　CLKCONSTA 时钟状态寄存器

位	名称	复位	R/W	描　　述
7	OSC32K	1	R	32kHz 时钟源选择 0：32kHz 晶振 1：32kHz RC 振荡
6	OSC	1	R	主时钟源选择 0：32MHz 晶振 1：16MHz RC 振荡
5：3	TICKSPD[2：0]	001	R	定时器计数时钟速率（该时钟频不大于 OSC 决定频率） 000：32MHz 001：16MHz 010：8MHz 011：4MHz 100：2MHz 101：1MHz 110：0.5MHz 111：0.25MHz
2：0	CLKSPD	001	R	时钟速率，不能高于系统时钟 000：32MHz 001：16MHz 010：8MHz 011：4MHz 100：2MHz 101：1MHz 110：500kHz 111：250kHz

SLEEPCMD 控制寄存器的相关信息如表 5.21 所示，PERCFG 寄存器的相关信息如表 5.22 所示。

表 5.21　SLEEPCMD 睡眠模式控制寄存器

位	名称	复位	R/W	描　　述
7	—	0	R	预留
6	XOSC_STB	0	W	低速时钟状态 0：没有打开或者不稳定 1：打开且稳定
5	HFRC_STB	0	W	主时钟状态 0：没有打开或者不稳定 1：打开且稳定
4：3	RST[1：0]	xx	W	最后一次复位指示 00：上电复位 01：外部复位 10：看门狗复位
2	OSC_PD	0	W	节能控制，OSC 状态改变的时候硬件清 0 0：不关闭无用时钟 1：关闭无用时钟
1：0	MODE[1：0]	0	W	功能模式选择 00：PM0 01：PM1 10：PM2 11：PM3

表 5.22　PERCFG 外设控制寄存器

位	名称	复位	R/W	描　　述
7	—	0	R	预留
6	T1CFG	0	R/W	T1 I/O 位置选择 0：位置 1 1：位置 2
5	T3CFG	0	R/W	T3 I/O 位置选择 0：位置 1 1：位置 2
4	T4CFG	0	R/W	T4 I/O 位置选择 0：位置 1 1：位置 2
3：2	—	00	RO	预留
1	U1CFG	0	R/W	串口 1 位置选择 0：位置 1 1：位置 2
0	U0CFG	0	R/W	串口 0 位置选择 0：位置 1 1：位置 2

U0CSR 寄存器的相关信息如表 5.23 所示，U0GCR 寄存器的相关信息如表 5.24 所示，U0BUF 寄存器的相关信息如表 5.25 所示，U0BAUD 寄存器的相关信息如表 5.26 所示。

表 5.23　U0CSR（串口 0 控制和状态寄存器）

位	名称	复位	R/W	描　　述
7	MODE	0	R/W	串口模式选择 0：SPI 模式 1：UART 模式
6	RE	0	R/W	接收使能 0：关闭接收 1：允许接收
5	SLAVE	0	R/W	SPI 主从选择 0：SPI 主 1：SPI 从
4	FE	0	R/W	串口帧错误状态 0：没有帧错误 1：出现帧错误
3	ERR	0	R/W	串口校验结果 0：没有校验错误 1：字节校验出错
2	RX_BYTE	0	R/W	接收状态 0：没有接收到数据 1：接收到 1 字节数据
1	TX_BYTE	0	R/W	发送状态 0：没有发送 1：最后一次写入 U0BUF 的数据已经发送
0	ACTIVE	0	R	串口忙标志 0：串口闲 1：串口忙

表 5.24　U0GCR（串口 0 常规控制寄存器）

位	名称	复位	R/W	描　　述
7	CPOL	0	R/W	SPI 时钟极性 0：低电平空闲 1：高电平空闲
6	CPHA	0	R/W	SPI 时钟相位 0：由 CPOL 跳向非 CPOL 时采样，由非 CPOL 跳向 CPOL 时输出 1：由非 CPOL 跳向 CPOL 时采样，由 CPOL 跳向非 CPOL 时输出

续表

位	名称	复位	R/W	描　　述
5	ORDER	0	R/W	传输位序 0：低位在先 1：高位在先
4：0	BAUD_E[4：0]	0x00	R/W	波特率指数值，BAUD_M 决定波特率

表 5.25　U0BUF（串口 0 收发缓冲器）

位	名称	复位	R/W	描　　述
7：0	DATA[7：0]	0x00	R/W	UART0 收发寄存器

表 5.26　U0BAUD（串口 0 波特率控制器）

位	名称	复位	R/W	描　　述
7：0	BAUD_M[7：0]	0x00	R/W	波特率尾数，与 BAUD_E 决定波特率

表 5.19～表 5.26 中列举了和 CC2530 处理器串口操作相关的寄存器，其中包括：CLKCONCMD 和 CLKCONSTA 控制寄存器，用来控制系统时钟源和状态；SLEEPCMD 和 SLEEPSTA 寄存器，用来控制各种时钟源的开关和状态；PERCFG 寄存器为外设功能控制寄存器，用来控制外设功能模式；U0CSR、U0GCR、U0BUF、U0BAUD 等为串口相关寄存器。

3. 软件设计

```
#include <iocc2530.h>
#include <stdio.h>
#include "./uart/hal_uart.h"
#define uchar unsigned char
#define uint unsigned int
#define uint8 uchar
#define uint16 uint
#define TRUE 1
#define FALSE 0
//定义控制 LED 灯的端口
#define LED1 P1_0                          //定义 LED1 为 P1_0 端口控制
#define LED2 P1_1                          //定义 LED2 为 P1_1 端口控制
uchar temp;
/*****************************
//延时函数
*****************************/
void Delay(uint n)
{
    uint i,t;
    for(i = 0;i<5;i++)
```

```
    for(t = 0;t<n;t++);
}
void InitLed(void)
{
    P1DIR | = 0x03;                          //P1_0、P1_1 定义为输出
    LED1 = 1;                                //LED1 灯熄灭
    LED2 = 1;                                //LED2 灯熄灭
}
void main(void)
{
    char receive_buf[3];
    uchar counter = 0;
    uchar RT_flag = 1;

    InitUart();                              // 波特率为 57600bps
    InitLed();
    prints("input: 11 -----> LED1 on     10 -----> LED1 off     21 -----> LED2 on
20 -----> LED2 off\r\n");
    while(1)
    {
        if(RT_flag == 1)                     //接收
        {
          if( temp != 0)
          {
             if((temp!= '\r')&&(counter<3))  //'\r'回车键为结束字符
                                             //最多能接收 3 个字符
              {
                receive_buf[counter++] = temp;
              }
              else
              {
                RT_flag = 3;                 //进入 LED 设置状态
              }
              if(counter == 3) RT_flag = 3;
              temp = 0;
          }
        }
        if(RT_flag == 3)                     //LED 状态设置
        {
            U0CSR &= ~0x40;                  //禁止接收
            receive_buf[2] = '\0';
           // prints(receive_buf); prints("\r\n");
            if(receive_buf[0] == '1')
            {
            if(receive_buf[1] == '1') { LED1 = 0; prints("led1 on\r\n"); }
           else if(receive_buf[1] == '0') { LED1 = 1; prints("led1 off\r\n"); }
            }
            else if(receive_buf[0] == '2')
```

```
            {
              if(receive_buf[1] == '1') { LED2 = 0; prints("led2 on\r\n"); }
              else if(receive_buf[1] == '0') { LED2 = 1; prints("led2 off\r\n"); }
            }
            U0CSR | = 0x40;                    //允许接收
            RT_flag = 1;                       //恢复到接收状态
            counter = 0;                       //指针归 0
        }
    }
}
/*********************************************************************
* 函数功能 : 串口接收一个字符
* 入口参数 : 无
* 返 回 值 : 无
* 说 明 : 接收完成后打开接收
*********************************************************************/
#pragma vector = URX0_VECTOR
__interrupt void UART0_ISR(void)
{
    URX0IF = 0;                                //清中断标志
    temp = U0DBUF;
}
```

程序通过配置 CC2530 处理器的串口相关控制寄存器来设置串口 0 的工作模式为串口模式，波特率为 57600bps，通过判断串口的输入来控制 LED 灯的状态。

4. 实施步骤

(1) 使用 ZigBee Debuger USB 仿真器连接 PC 和 ZigBee(CC2530)模块，打开 ZigBee 模块开关供电。将系统配套串口线一端连接 PC，另一端连接在 ZigBee 调试板的串口上。

(2) 启动 IAR 开发环境，新建工程。

(3) 在 IAR 开发环境中编译、运行、调试程序。

(4) 使用 PC 自带的超级终端连接串口，将超级终端设置为串口波特率 57600bps、8 位、无奇偶校验，无硬件流模式，当向串口输入相应数据格式的数据时，即可控制 LED 灯的开关。

LED1 开：　11 回车

LED1 关：　10 回车

LED2 开：　21 回车

LED2 关：　20 回车