Renergy

IO 模拟 UART 程序应用笔记

Date: 2014-05-05

Rev: 1.0

深圳市锐能微科技有限公司

文档修订控制

序号	版本号	修订日期	修订概述	修订人	备注
1	V1.0	2014-05-05	初稿	林小满	

目录

1. 适用范围

本文适合使用单片机 IO 口模拟 UART 接口和锐能微 UART 接口的 RN820X 系列芯片通信。
本文所使用的单片机为锐能微基于 ARM-M0 核的 SOC RN821X。

2. 原理概述

本应用用于扩展UART端口，在单片机自带的UART口不够用的情况下，使用GPIO、定时器和外部中断实现模拟UART通信。

3. 实现方案

使用定时器周期产生中断，作为波特率的基准，定时器的定时时间是波特率周期的一半。在中断里面进行 GPIO 的读写，和判断起始和结束，实现 UART 的发送和接收。结构如：

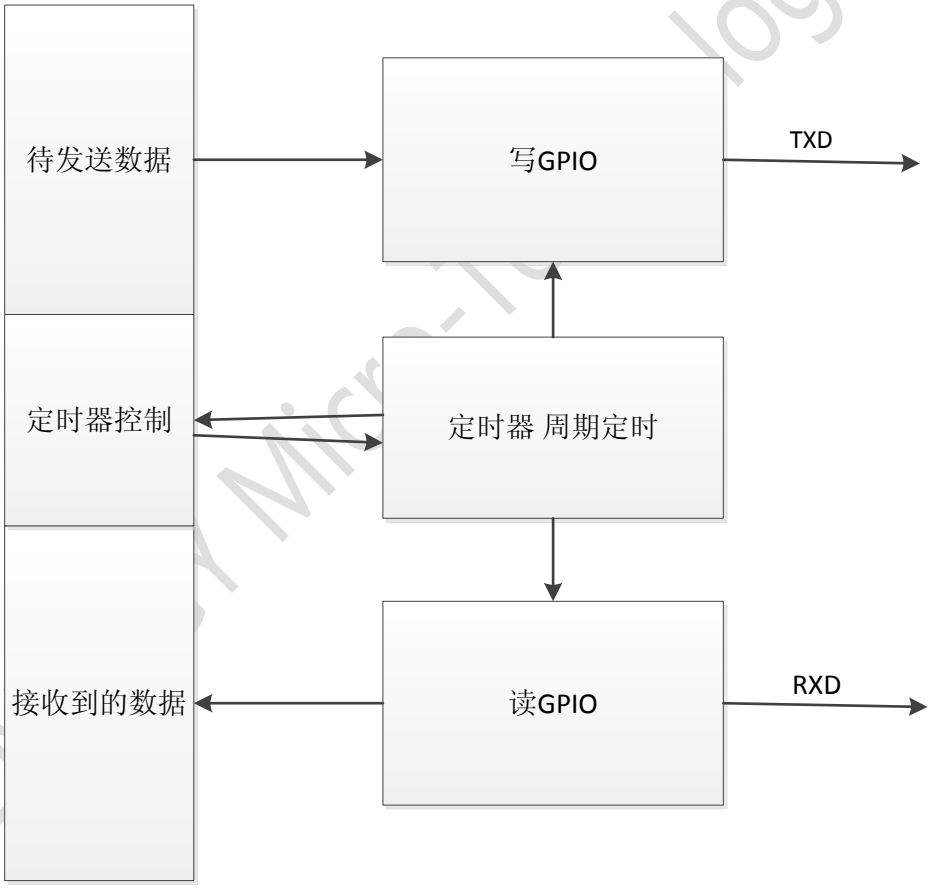

图 3.1 系统结构图

3.1. IO 模拟 UART 发送流程

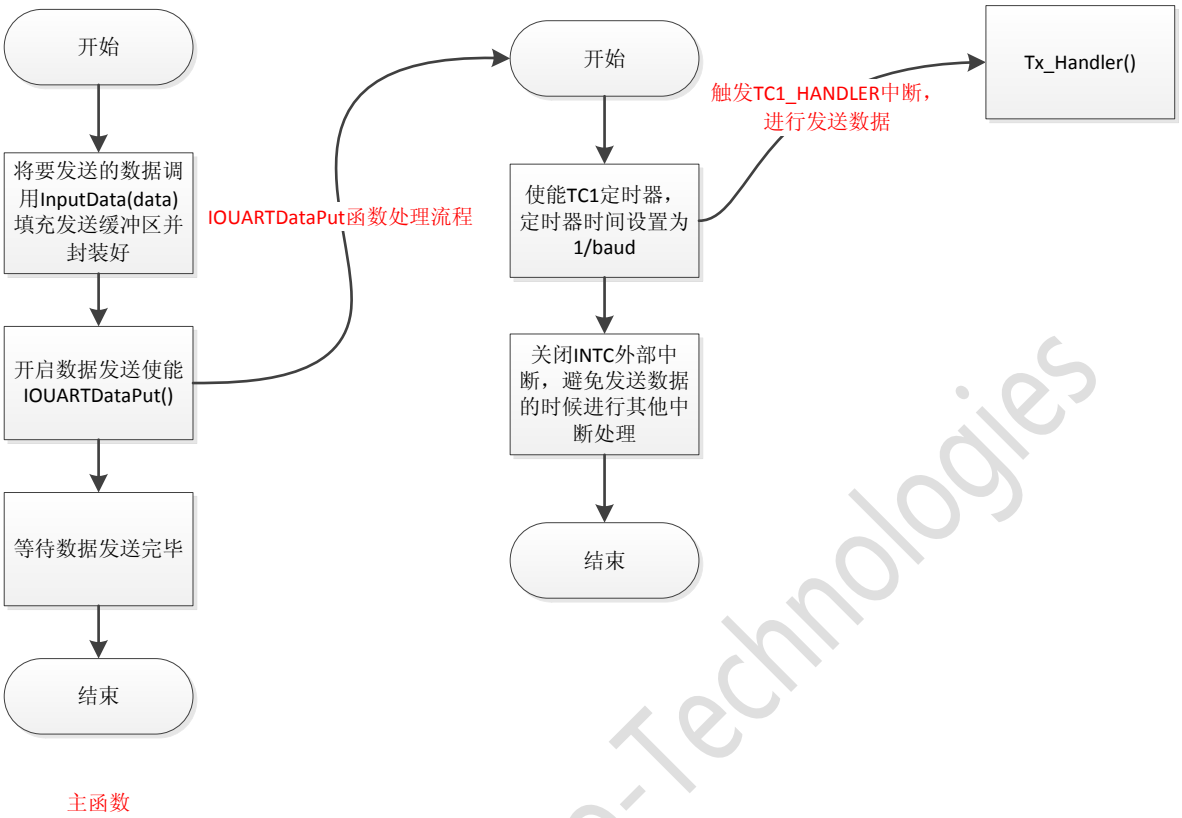

图 3.1-1IO 模拟 UART 发送流程图

图 3.1-2Tx_Handler 处理流程图

【简单说明】

- 主函数流程中将要发送的数据调用 InputData()接口将数据填充发送缓冲区并封装好
- 填充好发送缓冲区数据后，再调用开启数据发送使能接口 IOUARTDataPut（）即可
- TC0 定时器时间设定为 1/baud 的时间，在中断发生时进行逐个发送封装好的数据（该数据封装格式：(2bit 停止位(3) + 1bit 偶校验位+8bit 数据位+1bit 起始位(0)）），发送缓冲区在 tx_buf 全局变量中定义
- 数据发送从起始位开始发送，逐个发送至停止位，为确保停止位发送成功，此处采用双停止位
- 当发到最后一位时进行判断发送缓冲队列中数据是否发送完成，若发送完成再进行相应的数据资源初始化，同时开启接收使能即外部中断使能开关。

3.2. IO 模拟 UART 接收流程

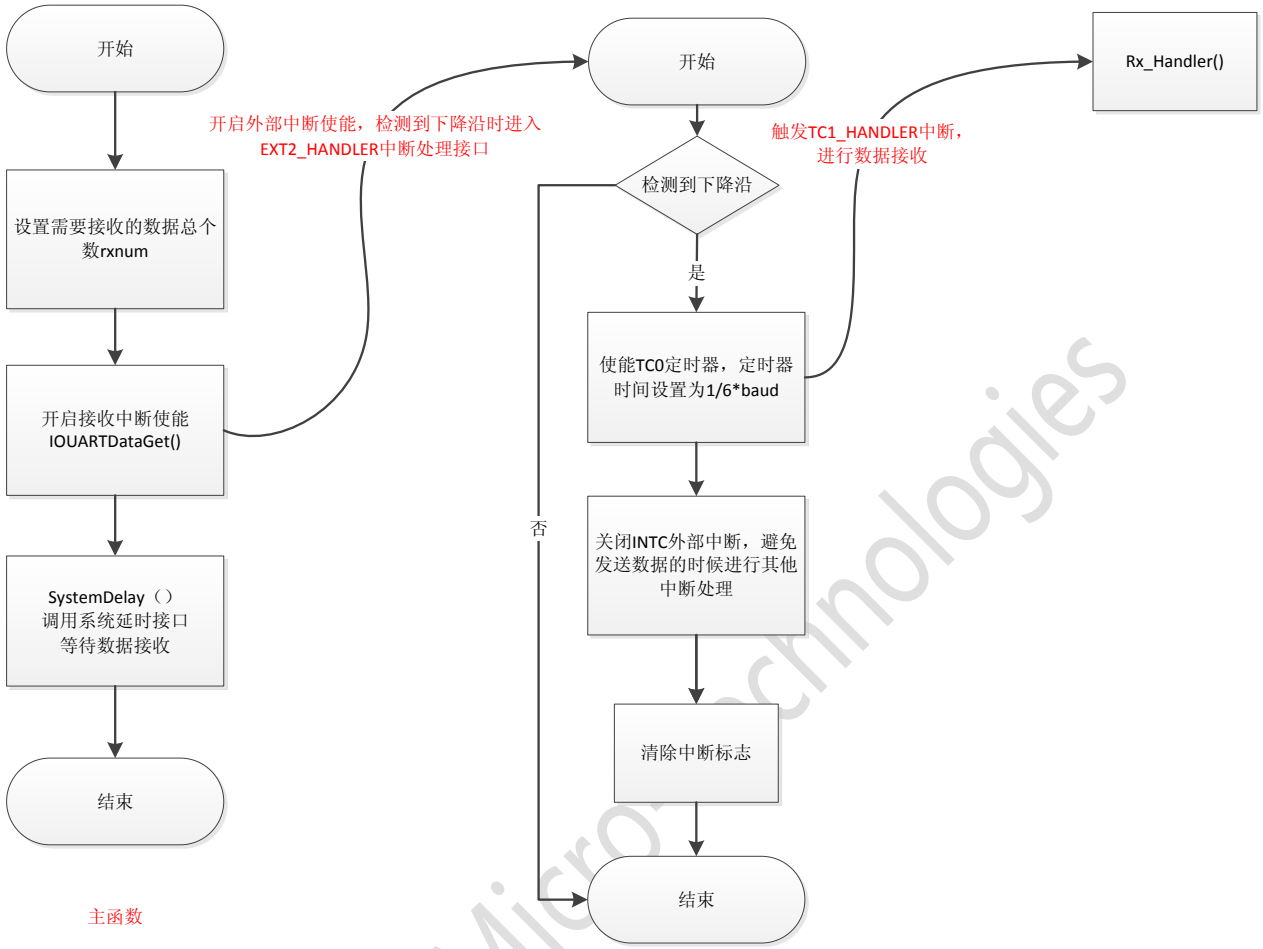

图 3.2-1IO 模拟 UART 接收流程图

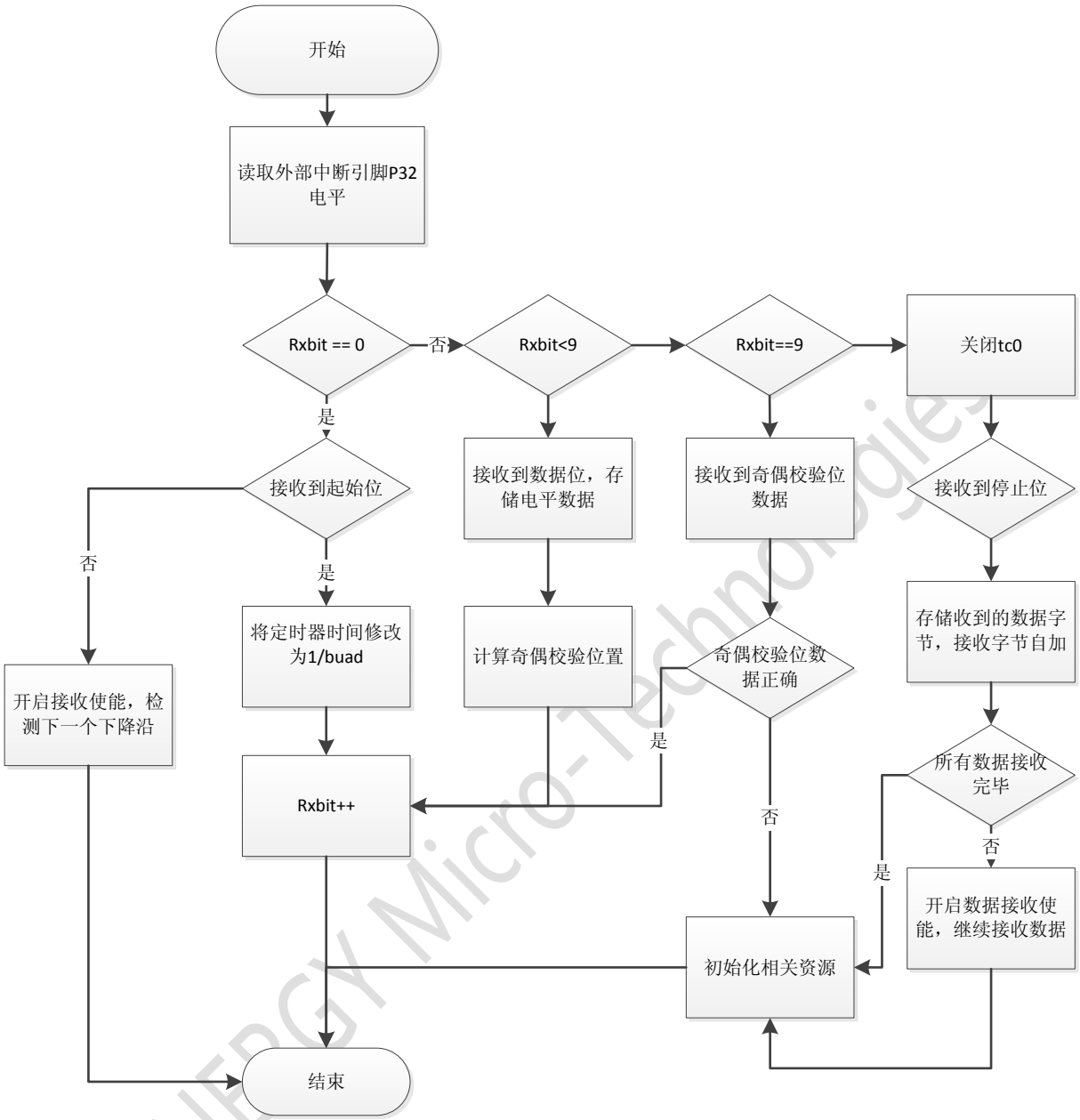

图 3.2-2Rx_Handler 接口处理流程图

【简单说明】

- 主函数流程中设置好接收数据总数后，再调用开启接收数据使能接口 IOUARTDataGet（）即可等待数据接收
- 接收到的数据将存储在 rx_buf 全局变量中
- 接收使能开关主要用于触发外部中断使能，用于检测下降沿的到来
- 检测到下降沿再开启 TC0 定时器，定时器时间理论上设置为 1/2*buad，由于 TC0 定时器开启时间有所延迟，在此将定时器时间设置为 1/6*buad，定时器的调整主要为确保第一个进入第一个时间中断时刚好检查到数据的中间位置才可确保后面的时间正确，此步骤非常重要。
- 检查到停止位后，再将定时器调整为 1/buad，进行检测后面的数据。

4. IOUART 函数应用参考

4.1. IOUART 数据类型及结构定义

```
#define TX_MODE 1
#define RX_MODE 0
typedef struct{
 uint16_t  data[10];//发送数据缓冲队列
   uint8_t  num;        //缓冲队列有效数据个数
   uint8_t  txbit;      //记录当前数据发送数据位
   uint8_t  txcnt;      //记录当前已发送数据个数
}TX_BUF;//发送缓冲区数据结构

typedef struct{
   uint16_t data[10];  //接收数据缓冲队列
   uint8_t num;        //记录当前接收到数据的个数
   uint8_t rxbit;      //记录当前接收数据的数据位
   uint8_t SBUF;       //记录最近一次接收到的有效字节
}RX_BUF;//接收缓冲区数据结构
```

4.2. IOUART 主要函数定义

4.2.1 MainClock_Init ()

```
void MainClock_Init( void );
```

函数功能：用于初始化系统时钟，设置成 PLL，开启 TC0 模块功能

函数入参：无

函数返回值：无

4.2.2SystemDataInit ()

```
void SystemDataInit(uint32_t baud);
```

函数功能：用于系统数据初始化，主要进行全局变量的初始化

函数入参：

- **Baud：**比特率

函数返回值：无

4.2.3SystemIoInit ()

```
void SystemIoInit( void );
```

函数功能：io 模拟 uart 口相关资源初始化

1、 开启 GPIO CLK

2、 开启 INTC CLK

3、 配置接收 GPIO 口，此处使用 P32, 此接口必需有外部中断功能

4、 配置发送 GPIO 口，此处使用 P27，只要是 IO 口即可

5、 配置 INTC 模块功能，配置为检测下降沿模式，使能该模块

函数入参：无

函数返回值：无

4.2.4 InputData()

```
void InputData(uint16_t ucData);
```

函数功能：发送缓冲区数据封装接口，主要将停止位、数据位、奇偶校验位、起始位集成到同一个变量中并放入发送缓冲队列中，同时缓冲队列长度自加

函数入参：

- **ucData：**需要进行封装的数据

函数返回值：无

4.2.5IOUARTDataPut ()

```
void IOUARTDataPut ( void );
```

函数功能：数据发送使能开关，主要关闭外部中断使能，开启 TC0 定时器

函数入参：无

函数返回值：无

4.2.6IOUARTDataGet ()

```
void IOUARTDataGet ( void );
```

函数功能：数据接收使能开关，主要开启外部中断使能，关闭 TC0 定时器

函数入参：无

函数返回值：无

4.2.7Rx_Handler ()

```
void Rx_Handler ( void );
```

函数功能：数据接收处理接口

函数入参：无

函数返回值：无

4.2.8 Tx_Handler ()

```
void Tx_Handler ( void );
```

函数功能：数据发送处理接口

函数入参：无

函数返回值：无

4.2.9TC0_HANDLER()

```
void TC0_HANDLER( void );
```

函数功能：TC0 定时器中断处理接口

函数入参：无

函数返回值：无

4.2.10EXT2_HANDLER()

```
void EXT2_HANDLER( void );
```

函数功能：外部中断处理接口，用于检测接收到下降沿

函数入参：无

函数返回值：无

4.2.10iouart_rx_frame_09()

```
void iouart_rx_frame_09(uint8_t num);
```

函数功能：接收数据帧（读 RN8209C 寄存器）例子，此为测试程序部分代码

函数入参：

- **num：**需等待接收数据的个数，不含校验位

函数返回值：无

4.2.11iouart_tx_frame_09()

```
void iouart_tx_frame_09 (uint8_t num);
```

函数功能：发送数据帧（写 RN8209C 寄存器）例子，此为测试程序部分代码

函数入参：

- **num：**入参为需要发送数据的个数，不含校验位

函数返回值：无

4.2.12iouart_test()

```
void iouart_tx_frame_09 (uint8_t num);
```

函数功能：读取 8209C 寄存器列表中寄存器的值 test 程序，此为测试程序部分代码

函数入参：无

函数返回值：无

4.3. 接收发送应用实例

```
#include "RN8213.h"
#include "io2uart.h"
#include "string.h"

uint8_t rxnum = 0;       //接收数据总数全局变量
uint32_t BaudRate = 0;  //波特率全局变量
TX_BUF txbuf;           //接收缓冲区全局变量
RX_BUF rxbuf;          //发送缓冲区全局变量

/********** 接收数据帧（读 RN8209C 寄存器）例子 ********/
void iouart_rx_frame_09(uint8_t num)// 入参为需等待接收数据的个数，不含校验位
{
    SystemDelay(1000);
    rxnum = num + 1;              //记录等待接收的数据总个数，含校验位
    InputData(txbuf.data[0]);       //将需要发送的数据封装好放入发送缓冲队列中
    IOUARTDataPut();              //开启数据发送使能
    SystemDelay(1000);             //等待发送完成
    return;
}
/********** 发送数据帧（写 RN8209C 寄存器）例子 ********/
void iouart_tx_frame_09(uint8_t num) // 入参为需要发送数据的个数，不含校验位
{
    uint8_t  CHSUM = 0,i=1;

    SystemDelay(1000);
    for(i = 0;i< num;i++)
    {
        CHSUM += txbuf.data[i];  //计算 CHECKSUM
        InputData(txbuf.data[i]);//将需要发送的数据封装好放入发送缓冲队列中
    }
    CHSUM=(~CHSUM);       //取反
    InputData(CHSUM);     //将需要发送的数据封装好放入发送缓冲队列中
    IOUARTDataPut();      //开启数据发送使能
    SystemDelay(1000);
```

```
        return;
    }
    int32_t  main ( void )
    {
        MainClock_Init();    //初始化系统时钟
        SystemDataInit(4800);//系统上电初始化，8209C 使用 4800 比特率
        SystemIoInit();      //系统接口初始化
        WdtFeed();           //清除看门狗计数器

        while(1)
{
            //写 RN8209C 寄存器:
            txbuf.data[0] = 0xEA;  //修改写使能寄存器，开启 8209C 写使能功能
            txbuf.data[1] = 0xE5;
            iouart_tx_frame_09(2);

            txbuf.data[0] = 0x82;  //填写写寄存器命令（0x80+ADDR）
            txbuf.data[1] = 0x5A;  //填写修改数据
            txbuf.data[2] = 0xA5;
            iouart_tx_frame_09(3);

            //读 RN8209C 寄存器:
            txbuf.data[0]=0x02;     //填写需读取的寄存器（ADDR）
            iouart_rx_frame_09(2);
        }
    }
```