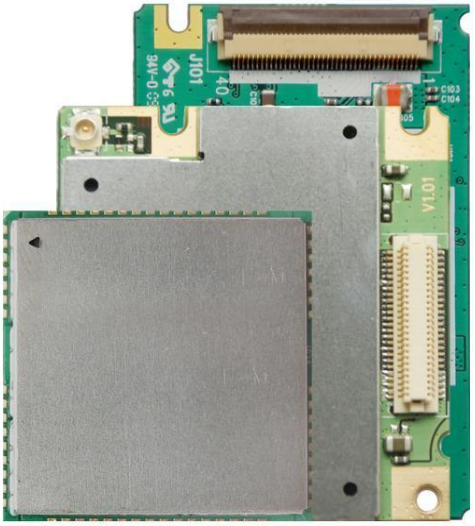
GSM无线通信模块

GSM模块串口软件流控应用指导



文档名	GSM 模块串口软件流控应用指导
版本	1.1
日期	2015-04-02
状态	正式发布

目录内容

0. 修改记录

版本	日期	作者	修改内容记录
1.0	2010-10-8	辛健	初始版本
1.1	2015-04-02	张涛	增加适用模块说明

1. 概要

在没有启用串口硬件流控功能的情况下，使用模块的一些数据传输功能，在传输的过程中数据可能会丢失。基于这种无硬件流控的传输需求，Quectel 提供了软件流控方案。该文档主要描述 Quectel 模块软件流控应用的方法。

本文档适用于所有Quectel GSM模块。

1.1. 参考文档

表 1: 参考文档表

序号	文档名	备注
[1]	Mxx_ATC	AT 命令集简介
[2]	GSM_UART_AN	串口应用说明

2. 应用环境的假设

该文档中，客户端的 CPU 我们定义为 MCU，而 Quectel 模块统称为模块。同时假设 MCU 上的 UART 控制器没有软件流控功能和转义功能，所以需要 MCU 去编写代码来仿真实现软件流控功能。

其他常用开发平台中，通讯串口一般都支持软件流控功能，但当数据流中存在一些数据和 **XON**、**XOFF** 信号值相等时，流控过程就被打乱了。对于这些平台，要求开发者在发送数据前先进行转义，然后把转义后的数据包进行发送。同样的方式，在接收流程中数据需要进行转义，来恢复数据。

3. 启动 Quectel 模块软件流控功能

使用Quectel模块软件流控功能，首先需要运行**AT+IFC=1,1**命令来通知模块启动软件流控功能，用户可以使用**AT&W**来保存这个配置。

同时建议用户使用**AT+IPR**命令在串口上设置成固定波特率进行工作。

4. Quectel 软件流控方案

Quectel模块软件流控方案中需要使用以下三个特殊字符：**XON(0x11)**，**XOFF(0x13)**，**ESCAPE(0x77)**。下面解释这些字符功能的各自作用。

当MCU通过串口接收大量数据，而MCU来不及处理这些数据时，可以在串口上直接发送**XOFF**字符给模块，通知模块暂停数据的发送。

当MCU部分或全部处理完这些数据后，需要在串口上直接发送**XON**字符给模块，及时地通知模块继续后续数据的发送。

在用户将要发送的数据中，本身可能包含**0x11**，**0x13**， **0x77**这三个个字符，那么要求MCU把这些数据进行转义后才可以发送。否则模块会误解这些数据为**XON**，**XOFF**等控制字符；**0x11**转义为**0x77 0xEE**两个字符，**0x13**转义为**0x77 0xEC**两个字符，**0x77**转义为**0x77 0x88**两个字符；当模块接收到这些带**ESCAPE(0x77)**前导字符的数据后，会“自动”地恢复成原始数据进行处理。

同样，当MCU通过串口发送大量数据给模块，模块来不及处理这些数据时，模块将在串口上发送**XOFF**字符给MCU，通知MCU暂停数据的发送；所以，需要注意的是：当MCU在串口上不停地发送大量数据给模块的时候，要有能力或办法“监视”串口上模块发给MCU的数据，用来判断是否含有**XOFF**，**XON**等控制字符。

当模块处理完部分或全部数据后，模块将在串口上发送**XON**字符给MCU，通知MCU进行后续数据的发送。

模块发送数据给MCU时，数据本身可能含有**0x11**，**0x13**，**0x77**等三个字符，那么模块将把这些数据进行转义后再发给MCU，**0x11**转义为**0x77 0xEE**两个字符，**0x13**转义为**0x77 0xEC**两个字符，**0x77**转义为**0x77 0x88**两个字符；当MCU接收到这些带**ESCAPE(0x77)**前导字符后，要把这些数据恢复后再提交给MCU的上层应用进行处理。

5. MCU 软件流控处理流程图

下面的MCU软件流控处理流程图仅仅是一个参考建议，MCU可以根据自己的平台特色进行修改。

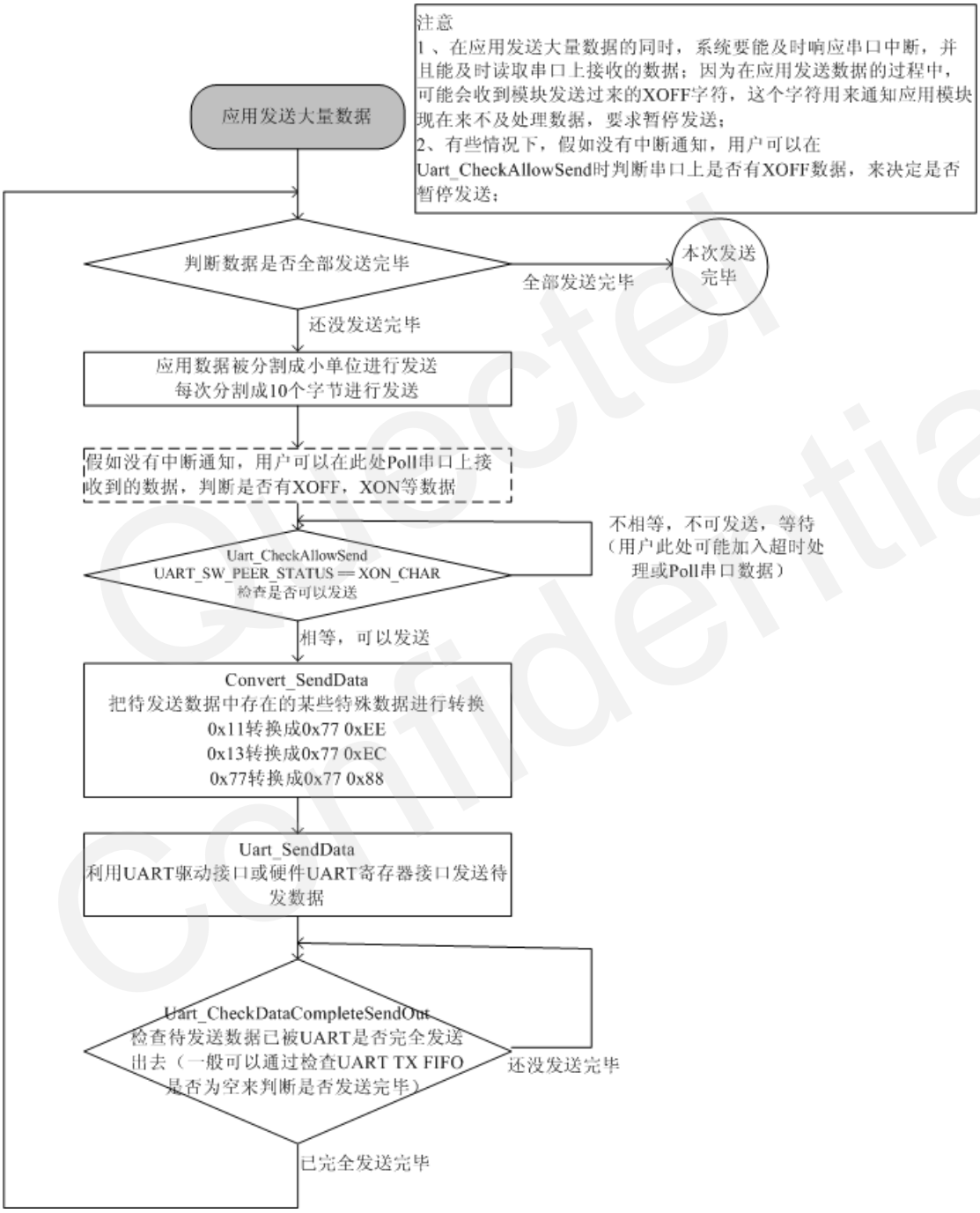


图 1: 软件流控MCU发送流程

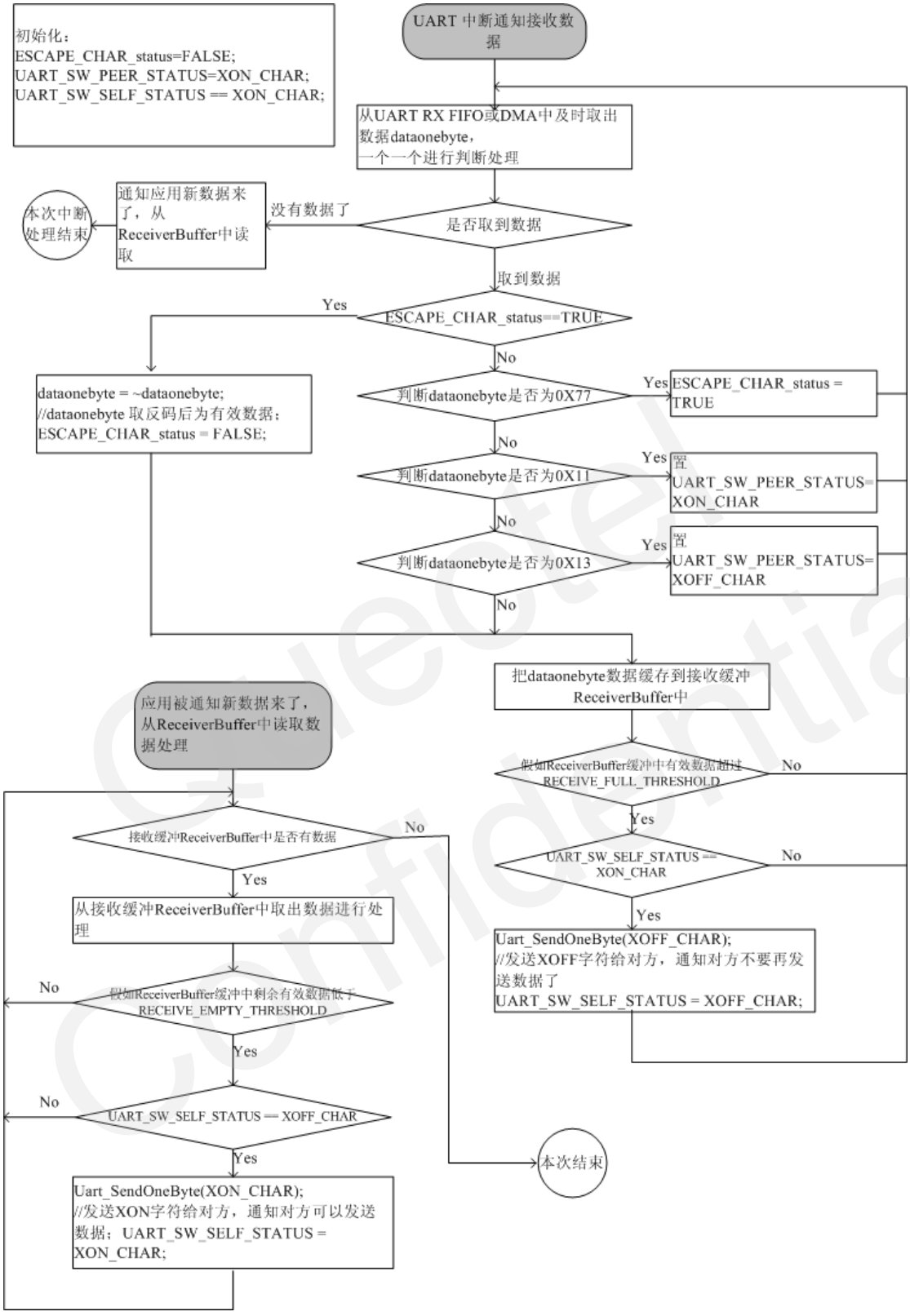


图 2: 软件流控MCU接收流程

6. MCU 软件流控处理流程伪代码

```
#define XON_CHAR          0x11
#define XOFF_CHAR         0x13
#define ESCAPE_CHAR           0x77
#define SEGMENTATION_LENGTH  10

#define RECEIVE_FULL_THRESHOLD    10
#define RECEIVE_EMPTY_THRESHOLD 10

static char UART_SW_PEER_STATUS      = XON_CHAR;
static char UART_SW_SELF_STATUS      = XON_CHAR;
static bool ESCAPE_CHAR_status = FALSE;

Application_SendData(Application_Buffer[],Application_Buffer_DataLength)
{
    Application_Buffer_pos = 0;

    //split Application_Buffer to small segmentation to send
    for(;Application_Buffer_DataLength>0;)
    {
        //check allow send
        while(!Uart_CheckAllowSend());

        //split small data to send
        if(Application_Buffer_DataLength > SEGMENTATION_LENGTH)
        {
            Data_Buffer_DataLength = SEGMENTATION_LENGTH;
            Application_Buffer_DataLength -= SEGMENTATION_LENGTH;
        }
        else
        {
            Data_Buffer_DataLength = Application_Buffer_DataLength;
            Application_Buffer_DataLength = 0;
        }

        //convert data
        Convert_Buffer_DataLength          =          Convert_Buffer_DataLength          =
Convert_SendData(Application_Buffer[Application_Buffer_pos],Data_Buffer_DataLength,
Convert_Buffer);
        Application_Buffer_pos += Data_Buffer_DataLength;
```

```
            //uart send
            Uart_SendData(Convert_Buffer, Convert_Buffer_DataLength);

            //must check uart send complete
            while(!Uart_CheckDataCompleteSendOut());
        }
}


int Convert_SendData(Data_Buffer[],Data_Buffer_DataLength, Convert_Buffer[])
{
        i=0;
        Convert_pos = 0;
        for(i=0;i<Data_Buffer_DataLength;i++)
        {
            if(Data_Buffer[i] == XON_CHAR)
            {
                Convert_Buffer[Convert_pos] = ESCAPE_CHAR;
                Convert_pos++;
                Convert_Buffer[Convert_pos] = ~XON_CHAR;
                Convert_pos++;
            }
            else if(Data_Buffer[i] == XOFF_CHAR)
            {
                Convert_Buffer[Convert_pos] = ESCAPE_CHAR;
                Convert_pos++;
                Convert_Buffer[Convert_pos] = ~XOFF_CHAR;
                Convert_pos++;
            }
            else if(Data_Buffer[i] == ESCAPE_CHAR)
            {
                Convert_Buffer[Convert_pos] = ESCAPE_CHAR;
                Convert_pos++;
                Convert_Buffer[Convert_pos] = ~ESCAPE_CHAR;
                Convert_pos++;
            }
            else
            {
                Convert_Buffer[Convert_pos] = Data_Buffer[i];
                Convert_pos++;
            }
        }
        return Convert_pos;
```

```
}



void Uart_SendData(Convert_Buffer[],Convert_Buffer_DataLength)
{
    //Send Data to UART Control
}

bool Uart_CheckDataCompleteSendOut(void)
{
    //Check all datas are sended out in UART Control FIFO
    //if UART Control FIFO is empty, return TRUE, otherwise return FALSE.
}

bool Uart_CheckAllowSend(void)
{
    return (UART_SW_PEER_STATUS == XON_CHAR);
}

//UART ISR notify received DATA, or poll data

void ISRorPollDataFromUart(void)
{
    char uartdata;


    //
    //read data from UART rx FIFO, put these datas to ReceiverBuffer[]

    while(Uart_CheckExistData())
    {
        uartdata = Uart_ReadByte();
        if(ESCAPE_CHAR_status)
        {
            ReceiverBuffer[ReceiverBuffer_Write_pos] = ~uartdata;
            ReceiverBuffer_Write_pos++;
            ReceiverBuffer_DataLength++;
            ESCAPE_CHAR_status = FALSE;
        }
        else if(uartdata == XON_CHAR)
        {
            UART_SW_PEER_STATUS = XON_CHAR;
        }
```

```
            else if(uartdata == XOFF_CHAR)
            {
                UART_SW_PEER_STATUS = XOFF_CHAR;
            }
            else if(uartdata == ESCAPE_CHAR))
            {
                ESCAPE_CHAR_status = TRUE;
            }
            else
            {
                ReceiverBuffer[ReceiverBuffer_Write_pos] = uartdata;
                ReceiverBuffer_Write_pos++;
                ReceiverBuffer_DataLength++;
                if((ReceiverBuffer_DataLength >= RECEIVE_FULL_THRESHOLD) &&
(UART_SW_SELF_STATUS == XON_CHAR))
                {
                    Uart_SendOneByte(XOFF_CHAR);
                    UART_SW_SELF_STATUS = XOFF_CHAR;
                }
            }
        }


        //notify application receive data, application copy data to application buffer
        Application_NotifyDataReceive();

}

bool Uart_CheckExistData(void)
{
        //Check uart rx fifo exist data
}

char Uart_ReadByte(void)
{
        //read one data from uart fifo
}

void Uart_SendOneByte(char sendchar)
{
        //Send one data
}

Application_MoveDataFromReceiveBuffer()
```

```
{
    //
    //move data from ReceiverBuffer[], and reduce ReceiverBuffer_DataLength
    //.....

    //
    if((ReceiverBuffer_DataLength <= RECEIVE_EMPTY_THRESHOLD) &&
(UART_SW_SELF_STATUS == XOFF_CHAR))
    {
        Uart_SendOneByte(XON_CHAR);
        UART_SW_SELF_STATUS = XON_CHAR;
    }
}
```

上海移远通信技术有限公司

上海市田州路 99 号 9 幢 501 室 200233

电话: +86 21 5108 2965

电子邮箱: info@quectel.com