15.6 通信设备类程序设计

本文档依据 STC 官方提供的源码对程序设计进行讲解，本文档仅涉及 CDC 设备中 POTS 模型中的抽象模型，其他模型的源码需要读者自行了解。本文档将使用 USB CDC 协议相关的知识，实现一个 USB 设备和 UART 设备的通信桥。

15.6.1 CDC 程序设计原理（缺少 LINECODING 数据结构的说明）

1. 设计分层

此次程序设计由于涉及不同的协议规范，因此需要对设计进行分层，以便日后扩展和修改。此次程序设计依据不同协议规范分为以下几层，由于厂商没有定义特定的请求故厂商请求层部分将略去。

（1）USB 层

（2）USB 标准请求层

（3）CDC 协议子类请求层

（4）串口层

1）USB 层

该层对应程序设计中的 usb.c 文件，该层存放 USB 初始化函数以及 USB 中断处理函数，USB 中断处理函数中提供了 USB 所有相关中断的的处理函数的入口，其中代码相同部分的分析详见 USB2.0 程序设计。

不同处，EPOUT1 的中断响应函数需要对主机发出的 OUT 进行响应；其需要进行将主机下行的数据包存入 256 字节的 RxBuffer 缓冲区中。

2）USB 标准请求层

该层对应程序设计中的 usb_req_std.c 文件，该层存放设备对 USB 主机 11 个标准请求的响应函数，具体详见第 3 章。

不同处，在控制传输阶段，若主机发送 OUT 包，需要设备对串口进行设置，串口设置函数处于 CDC 协议子类请求层。

3）CDC 协议子类请求层

该层对应程序设计中的 usb_req_class.c 文件，为了标准起见，本书 CDC 协议子类请求函数同样分为三个步骤：1. 特殊情况处理 2. 依据状态发包 3. 结束状态，即进行数据操作（如发送数据，或者接收数据，或者不进行数据操作）。

（1）usb_set_line_coding，格式见表 15.124。

表 15.124

bmRequestType	bRequest	wValue	wIndex	wLength	Data
00100001B	SET_LINE_CODING	0	接口号	结构体大小	线路编码结构体

第 1 步：特殊情况处理，若请求的 bmRequestType 字段不满足 usb_set_line_coding 请求的要求则挂起。

第 2 步：端点 0 数据区指向线路编码结构体的数据区，端点 0 的大小定义为请求数据结构的大小。

第 3 步:进入请求结束状态,端点 0 设置成 DATAOUT 状态表示设备接收来自主机的数据,寄存器 CSR0 的 SPORDY 位置位,表示处理完成从端点 0 接收到的数据包。

(2) usb_get_line_coding,格式见表 15.125。

表 15.125

bmRequestType	bRequest	wValue	wIndex	wLength	Data
10100001B	GET_LINE_CODING	0	接口号	结构体大小	线路编码结构

第 1 步:特殊情况处理,若请求的 bmRequestType 字段不满足 usb_get_line_coding 请求的要求则挂起。

第 2 步:端点 0 数据区指向线路编码结构体的数据区,端点 0 的大小定义为请求数据结构的大小。

第 3 步:请求结束状态,软件调用 usb_setup_in(),完成数据包的发送。

注意:usb_setup_in()的内容是:先将控制端点 0 设置为 DATAIN 状态,控制寄存器 CSR0 的 SPORDY 写 1 以清零 OPRDY,随后调用 usb_ctrl_in()进行数据发送,即使用控制 IN 类型向主机发送数据包。

(3) usb_set_ctrl_line_state,格式见表 15.126。

表 15.126

bmRequestType	bRequest	wValue	wIndex	wLength	Data
00100001B	SET_CONTROL_LINE_STATE	控制信号位图	接口号	0	无

第 1 步:特殊情况处理,若请求的 bmRequestType 字段不满足 usb_set_ctrl_line_state 请求的要求则挂起。

第 2 步:无该步骤。

第 3 步:进入请求结束状态,端点 0 设置成空闲状态,寄存器 CSR0 的 SPORDY 位以及 DATEND 位置位,表示处理完成从端点 0 接收到的数据包,且数据结束。

4) 串口层

该层对应程序设计中的 uart.c 文件,串口层中存放了串口初始化函数、串口中断处理函数,串口设置奇偶校验函数、串口设置波特率函数,还有串口轮询函数。由于轮询函数是本次程序设计中 STC32 最主要的工作负载,因此本节最为重要的是讲述轮询函数的写法。

(1) 串口设置

在前面 1.1.2 节中说过在控制传输阶段,若主机发送 OUT 包,需要设备对串口进行设置,尽管 USB 串口设置函数(sb_uart_settings)处于协议子类请求层,但是 USB 串口设置函数调用了串口设置奇偶校验函数、串口设置波特率函数。

其原理是,若在控制传输阶段,若主机发送 OUT 包,且请求类型为 SET_LINE_CODING,则需要依据接收到的线路编码结构体的奇偶校验字段以及波特率字段,对 STC32 上串口的奇偶校验以及波特率进行设置。

以上设置的好处是,通过次程序设计创建的工程不需要依据串口设备的波特率和奇偶校验进行额外的设置。

(2) 串口轮询

DATAIN:若串口设备通过发送数据给 STC32 触发了 STC32 的串口中断,STC32 会将串口

发送的数据缓存进缓存区 TxBuffer 中，此时指向 TxBuffer 数据的头指针与尾指针不相等。

若在串口轮询时发现 TxBuffer 数据的头指针与尾指针不相等，则启动 STC32 发送数据给 USB 主机的通信过程，具体步骤如下：

第一步：关闭 USB 中断；

第二步：设置端点为忙碌状态，将端点定位至端点 1；

第三步：将 TxBuffer 中的数据以字节为单位写入 FIFO1 寄存器中，若写入的字节数超过端点的大小或者数据的头指针与尾指针相等，则退出循环。

最后设置 INCSR1 寄存器的 INIPRDY 位（硬件要求），再打开 USB 中断。

注意：由于以上过程为轮询执行，故当 TxBuffer 中存储的数据大小超过端点大小时，在下一次轮询时 STC32 会继续发送 TxBuffer 缓冲存储区的数据，直至 TxBuffer 中的数据为空。

TX：在 1.1.1 节 USB 层中若 USB 主机通过下行数据给端点 1，从而触发了 EP1OUT 的中断，则 STC32 会将 USB 主机下行的数据缓存至 RxBuffer 中等待轮询发送，与 TxBuffer 一样，RxBuffer 也定义了数据的头指针与尾指针。

若在串口轮询时发现 RxBuffer 数据的头指针与尾指针不相等，则启动 STC32 发送数据给串口设备的通信过程，具体为：

第一步：设置串口忙

第二步：依据线路编码结构体中的校验类型字段设置 STC2 串口的校验类型。

第三步：以字节为单位将 RxBuffer 中的数据写入 S2BUF 寄存器中，并等待发送完成。

缓存区即将溢出处理：RxBuffer 和 TxBuffer 缓冲区的大小是有限的，其固定为 256 字节。如果下一次缓冲区中接收的数据要超过缓冲区的大小，则缓冲区应停止接收数据，并设置相应的状态字，如：UsbOutBusy，表示若接收下一次数据，则缓冲区数据将溢出，拒绝接收下一次数据。

2. 描述符数据结构

描述符数据结构存储于工程目录下 usb_desc.c 文件中，数据结构依据《通用串行总线通讯类设备定义》中的数据进行定义。读者可配合本节讲解对 usb_desc.c 文件进行阅读。

（1）设备描述符

通信类设备描述符 bDeviceClass 字段值固定为 02H，bDeviceSubClass 以及 bDeviceProtocol 字段值固定为 00H，其余字段值依据 USB 规范进行定义。

（2）配置描述符

依据 USB 规范，USB 主机发送 Get Descriptor 请求获取设备的描述符数据结构，其中配置描述符、接口描述符、类特定描述符、端点描述符是合并为一个数据结构发送给主机的。

① 配置描述符

配置描述符依据 USB 规范定义方式进行定义。需要注意的是本次实验中接口定义了两个，因此配置描述符中 bNumInterfaces 字段值定义为 0x02H，后面的接口描述符也将有两个，其接口号分别为 0 还有 1。

② 接口描述符

接口描述符 0

该接口描述符用于定义设备的接口类型为通信类模型，接口子类为抽象控制模型，协议类型为 V.25ter（通用 AT 命令），故 bInterfaceClass 字段值定义为 02H，bInterfaceSubClass 字段值定义为 02H，bInterfaceProtocol 字段值定义 01H。其余字段依据 USB 规范定义方式进行定义。

由于该描述符定义了抽象模型,故在该描述符后应当接续类特定描述符。该接口使用了定义为中断端点的端点 2。

接口描述符 1

该接口描述符由于定义设备的接口类型为数据类接口,该接口使用了定义为批量的端点 1。

③ 类特定描述符

各字段值请参照通用串行总线通信类设备(CDC)文档。

④ 端点描述符

端点 2

由于端点 2 仅被定义为中断 IN 端点,故 bEndpointAddress 字段值定义为 82H,bmAttributes 字段值定义为 03H,其余字段依据 USB 规范定义方式进行定义。

端点 1

由于端点 1 既要使用 IN 端点又要使用 OUT 端点,因此需要用两个数据结构分别定义端点 1。端点 1 的数据传输模式依据协议需要定义为批量类型的端点。因此端点 1IN 的 bEndpointAddress 字段值定义为 81H,端点 1OUT 的 bEndpointAddress 字段值定义为 01H,两个端点描述符结构中 bmAttributes 字段值都为 02H,其余字段依据 USB 规范定义方式进行定义。

(3) 厂商描述符

此处依据厂商标准对该描述符进行定义。

3. 通信数据流图

图 15.36 为本次程序设计的设备通信数据流简化图。

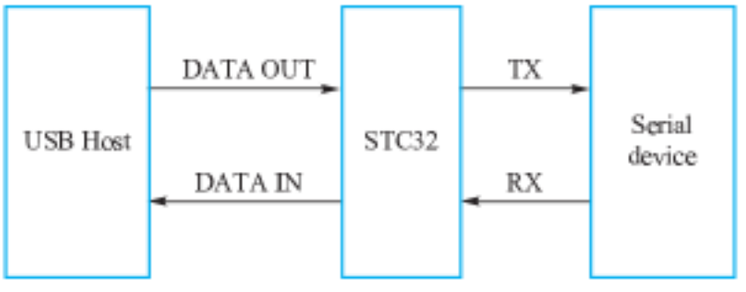

图 15.36　设备通信数据流图

更为详细的描述为:

USB 主机发送数据到串口设备:USB 主机触发 STC32 的 USB 中断,STC32 使用 EP1OUT 端点的 FIFO1 寄存器负责接收来自 USB 主机的数据,并将接收到的数据放入 RxBuffer 缓冲区内。在 RxBuffer 缓冲区内的数据需要等待 STC32 的轮询才能通过 S2BUF 寄存器被发送给串口设备。

串口设备发送数据给 USB 主机:串口设备发送数据触发 STC32 的串口中断,STC32 使用 TxBuffer 缓存来自 S2BUF 的数据,TxBuffer 缓冲区内的数据需要等待 STC32 的轮询才能通过 EP1IN 端点的 FIFO1 寄存器被发送给 USB 主机。

注意:在 STC32 的 RX 中断触发时,STC32 是从串口设备接收数据,但是其数据存入的缓冲区是 TxBuffer 缓冲区。

为了更详细地说明以上数据过程,上述通信模型可以描述为如图 15.37 所示。

图 15.37 中,黑框的 TxBuffer 与 RxBuffer 为软件定义地 256 字节数据缓冲区,灰框为 STC32 本身有的寄存器,白框为 USB 端点,箭头指向为数据流流向。注意 S2BUF 与 FIFO1 都

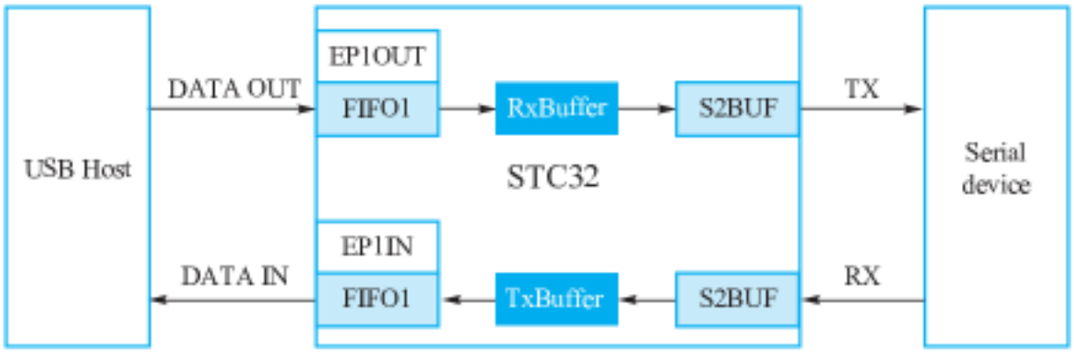

图 15.37　设备通信模型

是单字节的，从两个寄存器中读取数据至软件定义的数据的缓冲区是以入队的方式读取的，而从软件定义的数据缓冲区中发送数据给这两个寄存器是以出队的方式进行的。

4. 例程使用说明

STC32 实验箱上的 DB9 接口连接串口设备，STC32 实验箱上的 USB 接口连接 USB 主机，如图 15.38 所示，为了便捷起见我们使用 PC 兼做 USB 主机以及串口设备，需要注意的是，使用该例程时不能将串口与 USB 连接于同一 USB Hub，这样会造成设备冲突导致系统蓝屏。

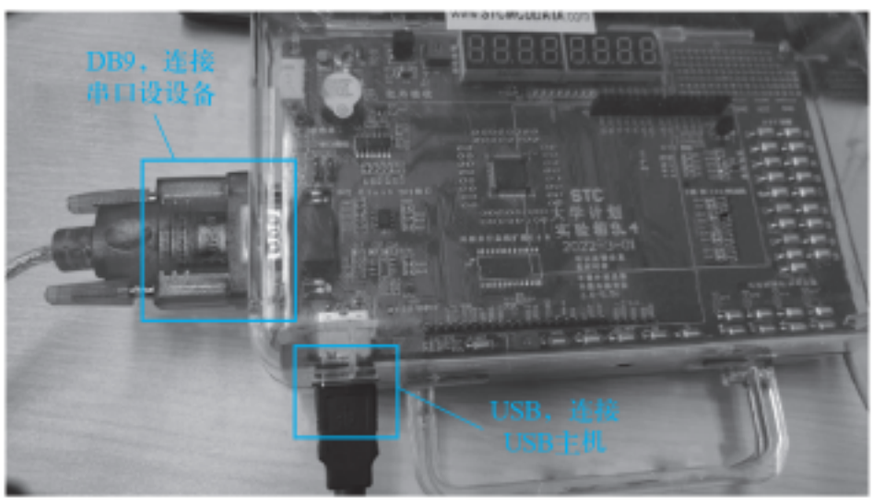

图 15.38　实验箱连接图

将连接好设备，右键开始打开 PC 的设备管理器，我们可以读取到连接 STC32 实验箱的串口号，如图 15.38 所示，串口号为 17。

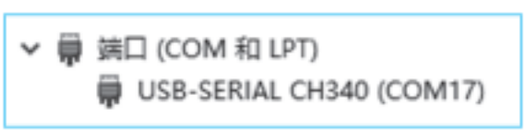

图 15.39　串口号

打开 XCOM 软件或者其他类型的串口管理软件，将串口号定位至设备管理器所读取的串口号，设置波特率为 115200，停止位为 1，数据位为 8，无奇偶校验，打开串口。如图 15.40 所示。

打开 STC-ISP 烧录程序，此时设备管理器会新识别一个串口设备，如图 15.41 所示，将 ISP 切换至 USB-CDC 串口将串口定位至 PC 新识别的那个串口号，其余串口参数按之前的设置情况进行设置。打开串口，如图 15.42 所示。

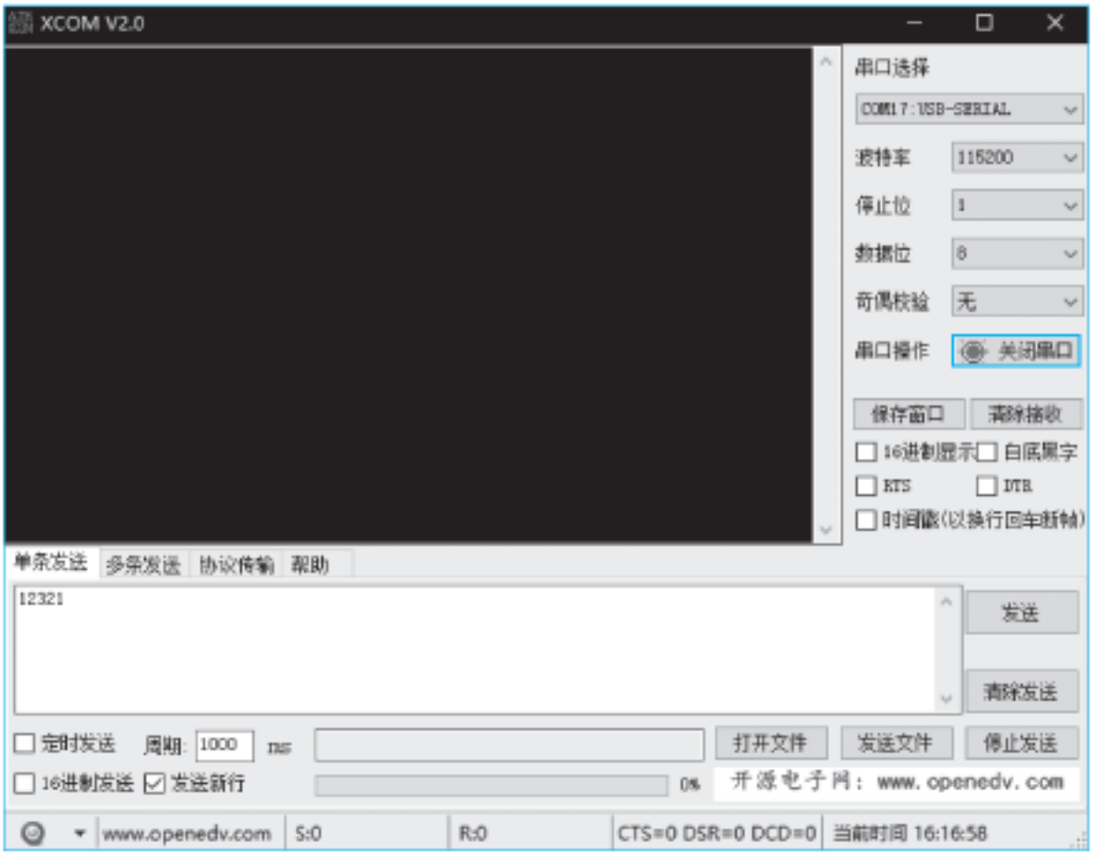

图 15.40　XCOM 设置界面

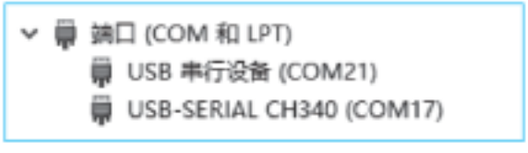

图 15.41　新识别出的串口号

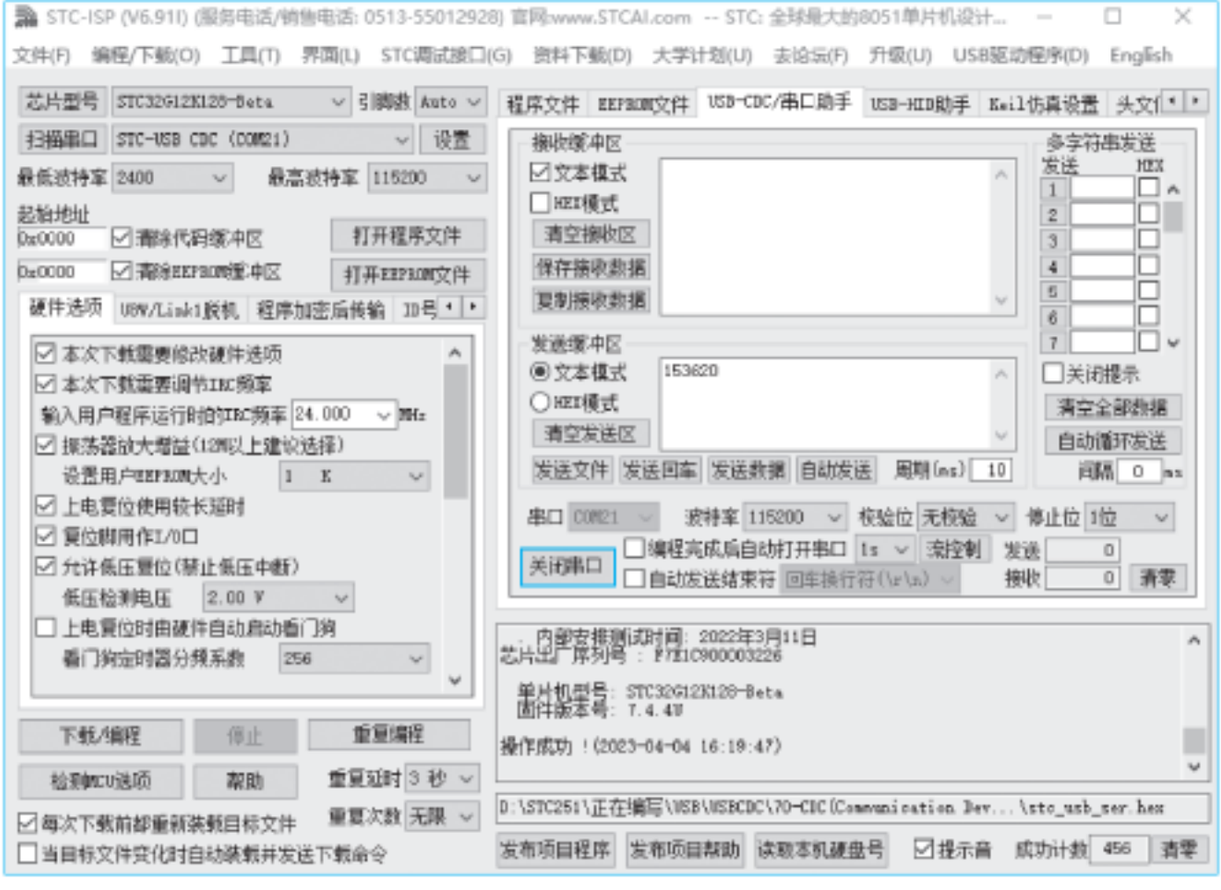

图 15.42　STC-ISP USB-CDC/串口助手界面

在 STC-ISP 的 USB-CDC/串口助手界面中的发送缓冲区输入数据，单击发送数据。之后切换回 XCOM 软件界面，发现正确接收数据，如图 15.43 所示。

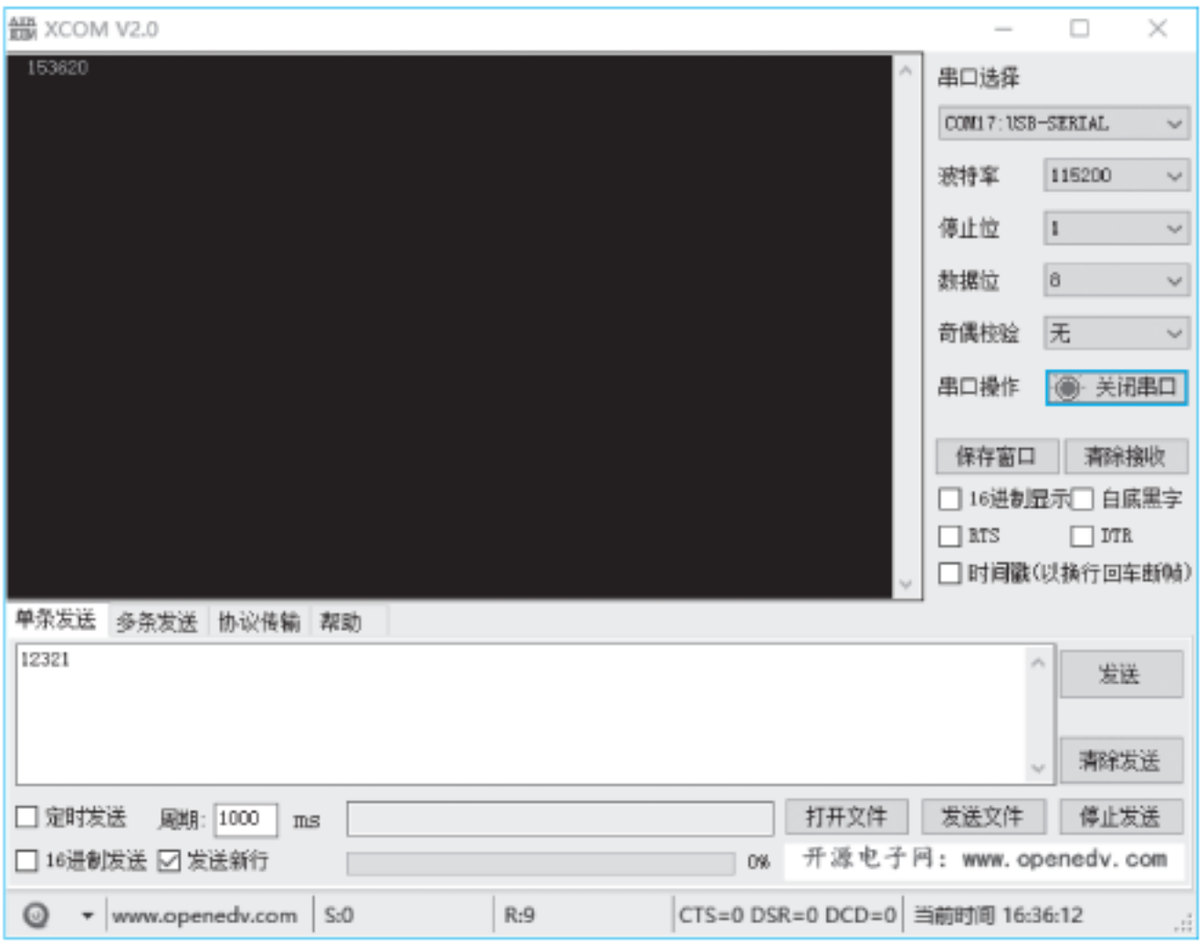

图 15.43　串口设备接收到的数据

在 XCOM 中输入数据并单击发送，且换回 STC-ISP，发现正确接收数据，如图 15.44 所示。

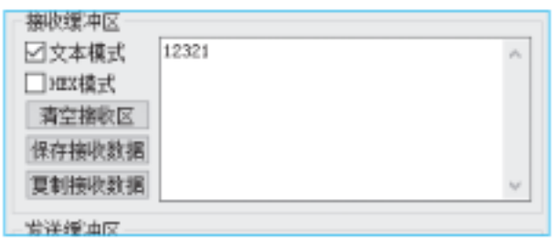

图 15.44　USB 主机接收到的数据