概述

当计算机或者其他 USB 主机上使用多个 USB 转串口设备时，会遇到多个串口无法与具体的串口设备对应起来的问题，包括更换不同 USB 端口串口序号发生改变，多个设备 USB 插拔顺序不同导致串口序号改变等问题。

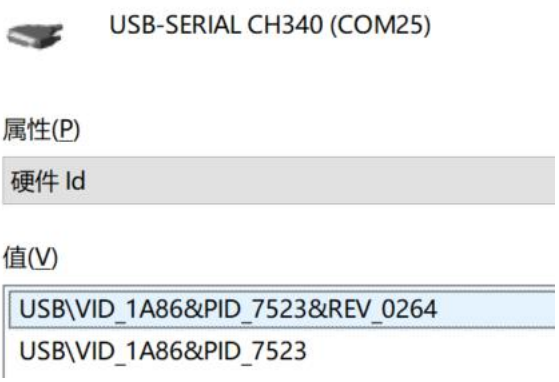
本文提出几种常见解决方式，用于解决开发及使用过程中遇到的问题。部分方法同样适用于单个 USB 串口设备连接的场景。

一、USB 设备硬件信息不同

当使用不同厂家的 USB 转串口芯片或同一厂家不同系列的 USB 转串口芯片，可使用以下几种常用方法用于区分多个设备。

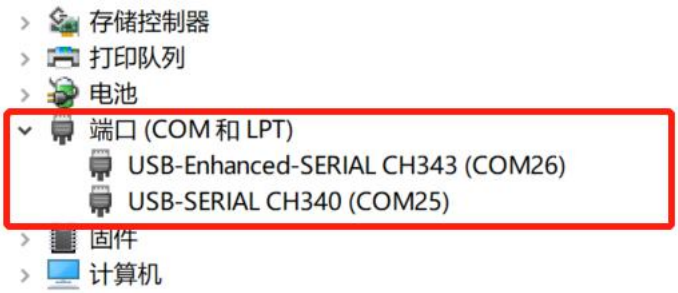
- **通过 USB 设备描述符中设备信息**

  USB 厂商 ID、USB 产品 PID、USB 厂商字符串、USB 产品字符串、串口序列号等



- **通过虚拟串口驱动生成的设备名称**

  当使用不同的 VCP 厂商驱动或类驱动时，串口设备名称会有区别。如：
  Windows 操作系统使用 USB 转串口芯片 CH340 和 CH343 时，设备管理器信息：



  Linux 操作系统下分别生成 ttyUSB 设备和 ttyACM 设备（使用系统自带 CDC 串口驱动时）。

```
[ 2860.684988] usb 3-1.2: new full-speed USB device number 9 using xhci_hcd
[ 2860.930717] usb 3-1.2: New USB device found, idVendor=1a86, idProduct=7523
[ 2860.930731] usb 3-1.2: New USB device strings: Mfr=0, Product=2, SerialNumber=0
[ 2860.930732] usb 3-1.2: Product: USB Serial
[ 2860.931854] ch341 3-1.2:1.0: ch341-uart converter detected
[ 2860.935188] usb 3-1.2: ch341-uart converter now attached to ttyUSB0
[ 2861.949636] usb 3-1.3: new full-speed USB device number 10 using xhci_hcd
[ 2862.195113] usb 3-1.3: New USB device found, idVendor=1a86, idProduct=55d3
[ 2862.195116] usb 3-1.3: New USB device strings: Mfr=0, Product=2, SerialNumber=0
[ 2862.195117] usb 3-1.3: Product: USB Single Serial
[ 2862.196174] cdc_acm 3-1.3:1.0: ttyACM0: USB ACM device
```

Android 系统下进行免驱 USB 应用开发，可直接调用 UsbDevice 类提供的接口函数。

String	getManufacturerName() Returns the manufacturer name of the device.
int	getProductId() Returns a product ID for the device.
String	getProductName() Returns the product name of the device.
String	getSerialNumber() Returns the serial number of the device.
int	getVendorId() Returns a vendor ID for the device.

- **通过 VCP 厂商驱动提供的应用层接口**

如使用 CH340 芯片及驱动时，可通过沁恒提供的动态库及 API 接口函数来判断当前操作的串口是否为 CH340 或 CH341。

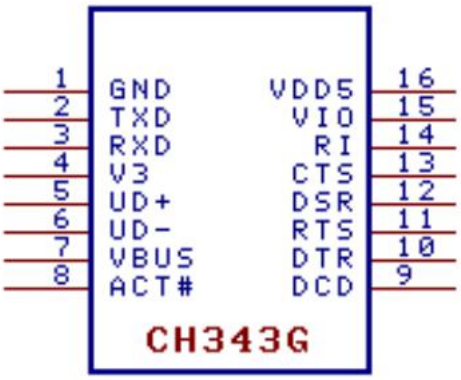
Windows 系统下动态库 CH341PT.DLL，接口函数：CH341PtNameIsCH341、CH341PtHandleIsCH341。除此之外，还提供 USB 热插拔检测等功能。

二、USB 设备硬件信息相同时

当使用 USB 硬件信息完全相同的 USB 转串口芯片，可使用以下几种常用方法用于区分多个设备。

- **通过串口 MODEM 信号线**

通常 USB 转串口芯片，除提供 TXD 和 RXD 异步串口信号，还支持 MODEM 联络信号 RTS、DTR、DCD、RI、DSR、CTS 等。其中 RTS 与 DTR 信号可用作通用输出信号，CTS、DSR、RI、DCD 信号可用作通用输入信号。可利用输入信号或输入与输出信号的组合来区分不同串口设备。



使用输入信号

在设备上电工作前根据需要将相应输入信号置高或拉低。操作串口时，主动获取 MODEM 输入信号的电平状态，根据状态区分串口设备。如：使用 4 路输入信号的组合，最多可以区分 $2^4=16$ 种串口外设。

使用输出与输入信号的组合

在设备上电工作前根据需要将输出信号与输入信号短接（可 1 对 1，也可以 1 对多）。操作串口时，置高或拉低 MODEM 输出信号，然后读取与其短接的输入信号的电平状态，根据状态是否同步区分硬件。如：短接 DTR 和 DSR 信号，先置低 DTR，读取 DSR 是否为低。再置高 DTR，读取 DSR 是否同步为高。

- **通过串口应用协议**

串口通信双方可事先约定好通信协议，通过串口数据内容来确认串口设备的“身份”。比如同一台计算机下需要接入多台串口设备，约定身份确认通信格式为：

计算机：包头+获取身份命令码+包尾+校验

设备：包头+身份 ID+包尾+校验

当需要与其中某个具体串口设备通讯前，先通过如上命令发送数据包，等待设备回复后，提取“身份 ID”用于计算机确认设备“身份”。

- **通过 USB 物理端口位置**

部分操作系统可以获取到 USB 设备接入的 USB 物理端口的绝对路径，如 Linux 系统可使用 sysfs 文件系统确定路径，然后使用 udev 工具为 USB 串口设备创建匹配规则，生成固定名称的串口名称。如通过 udevadm 命令（udevadm -a -n /dev/ttyUSB0）或者 sysfs（ls -la /sys/class/tty）查看 tty 串口详细信息：

```
looking at parent device '/devices/pci0000:00/0000:00:15.0/0000:03:00.0/usb3/3-1/3-1.3':
  KERNELS=="3-1.3"
  SUBSYSTEMS=="usb"
  DRIVERS=="usb"
  ATTRS{authorized}=="1"
  ATTRS{avoid_reset_quirk}=="0"
  ATTRS{bConfigurationValue}=="1"
  ATTRS{bDeviceClass}=="ff"
  ATTRS{bDeviceProtocol}=="00"
  ATTRS{bDeviceSubClass}=="00"
  ATTRS{bMaxPacketSize0}=="8"
  ATTRS{bMaxPower}=="98mA"
  ATTRS{bNumConfigurations}=="1"
  ATTRS{bNumInterfaces}==" 1"
  ATTRS{bcdDevice}=="0263"
  ATTRS{bmAttributes}=="80"
  ATTRS{busnum}=="3"
  ATTRS{configuration}==""
  ATTRS{devnum}=="15"
  ATTRS{devpath}=="1.3"
  ATTRS{idProduct}=="7523"
  ATTRS{idVendor}=="1a86"
  ATTRS{ltm_capable}=="no"
  ATTRS{maxchild}=="0"
  ATTRS{product}=="USB2.0-Serial"
  ATTRS{quirks}=="0x0"
  ATTRS{removable}=="unknown"
  ATTRS{speed}=="12"
  ATTRS{urbnum}=="20"
  ATTRS{version}==" 1.10"
```

udev 运行在用户态，脱离驱动层的关联，基于这种设计实现，用户可以通过编写规则来动态删除和修改/dev 下的设备文件，任意命名设备。

每当 udevd 收到 uevent 事件时就会去匹配规则，匹配成功后执行规则对应的操作。用户自定义规则放到/etc/udev/rules.d/目录下，以 rules 为扩展名。规则匹配主要基于几个字段：

KERNELS：kernel 对设备的命名。此处为 USB 设备路径名。

ATTR / ATTRS：设备的属性，如 idProduct/idVendor

SUBSYSTEMS：设备类型

ACTION：设备触发的操作，如 add/change/remove

实例：

接入两个相同的 CH340 设备，查看信息如下：

```
ttyUSB0 -> ../../devices/pci0000:00/0000:00:15.0/0000:03:00.0/usb3/3-1/3-1.3/3-1.3:1.0/ttyUSB0/tty/ttyUSB0
ttyUSB1 -> ../../devices/pci0000:00/0000:00:15.0/0000:03:00.0/usb3/3-1/3-1.4/3-1.4:1.0/ttyUSB1/tty/ttyUSB1
```

在/etc/udev/rules.d/创建自定义规则 70-usb.rules，内容如下：

```
KERNELS=="3-1.3:1.0", SUBSYSTEMS=="usb", MODE:="0666", SYMLINK+="my_serial0"
KERNELS=="3-1.4:1.0", SUBSYSTEMS=="usb", MODE:="0666", SYMLINK+="my_serial1"
```

通过命令：udevadm control --reload-rules 让规则生效或重启后自动生效。

设备接入后即可在/dev 下查看到 SYMLINK 中重命名的串口设备。