

CH34X 系列芯片串口 Android 程序开发说明

版本: 1.7

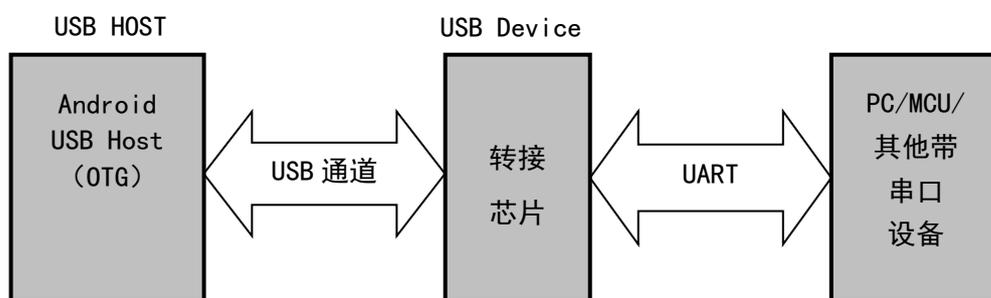
<http://wch.cn>

简介:

本文档是针对 CH340/CH341/CH342/CH343/CH344/CH347/CH9101/CH9102/CH9103/CH9104/CH9143 的 USB 转串口安卓库的开发说明文档。

本文档主要介绍如何使用芯片的 USB 转异步串口功能（以下简称 CH34XUART）以及 GPIO 功能，以及 Android 下如何使用 APK 操作实现串口通讯。该功能基于 Android USB Host 协议完成，用户可调用相关的接口 API 实现与 Android 设备进行通讯。

Android Host、USB Device、串口设备三者关系如下图。



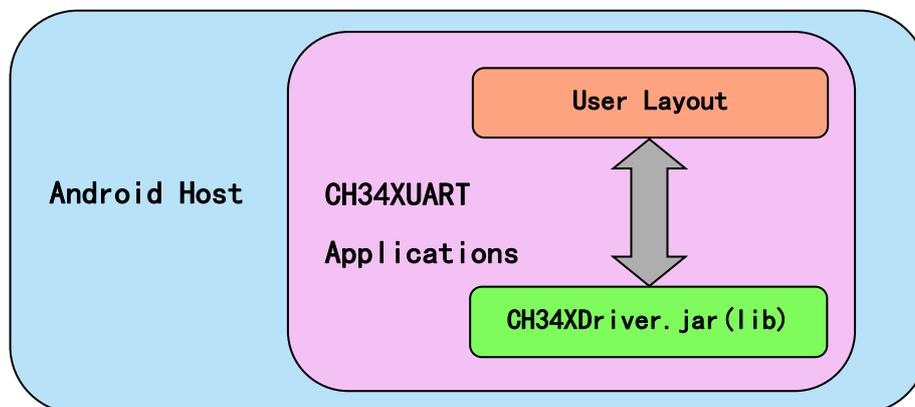
CH34X 串口提供的 Android 接口需要基于 Android 4.4 及以上版本系统,使用 CH34X 串口 Android 驱动条件:

- 1、基于 Android 4.4 及以上版本系统
- 2、Android 设备具有 USB Host 或 OTG 接口

本文档将会重点说明 Android USB Host 与 Device 的通讯接口 API 以及测试程序的操作说明。关于 Android USB Host 协议说明,可以参考 Google 官方文档。

一、Android Host

本文档所描述的例子程序皆是在 Android 4.4 及以上版本系统下编写的。Android 应用程序的启动参数是定义在 `device_filter.xml` 文件中的 `product-id` 和 `vendor-id`。基于 CH34X UART 开发的 Android 应用程序主要分为两个部分,如下图:



二、软件操作说明

用户需要在支持 USB Host 功能的 Android 设备上安装测试软件 (CH34XUARTDemo.apk)。点击扫描设备后, 弹框显示当前安卓设备上所连接的所有设备。点击某个设备打开, 如果没有相应的 USB 访问权限, 系统会自动弹出权限请求窗口, 进入软件后, 首先设置串口参数, 包括波特率、数据位、停止位、奇偶校验位以及硬件流控等。之后就可以执行数据收发操作。

三、函数接口说明

1、getInstance

```
public static WCHUARTManager getInstance()
```

用于获取全局唯一实例

返回	返回全局唯一实例
----	----------

2、init

```
public void init(android.app.Application application)
```

初始化上下文, 注册动态广播监听设备状态变化

参数	application - 全局上下文
----	---------------------

3、enumDevice

```
public java.util.ArrayList<android.hardware.usb.UsbDevice> enumDevice()
```

throws java.lang.Exception

枚举当前所有符合要求的 USB 设备

抛出	java.lang.Exception Exception
----	-------------------------------

4、getChipType

```
public cn.wch.uartlib.chiplmpl.type.ChipType2 getChipType(@NonNull
```

```
android.hardware.usb.UsbDevice usbDevice)
```

获取该 UsbDevice 的芯片类型

参数	usbDevice - USB 设备
返回	芯片类型 如果为 null 则表示无法识别该 USB 设备的芯片类型

5、openDevice

```
public boolean openDevice(@NonNull
                        android.hardware.usb.UsbDevice usbDevice)
throws cn.wch.uartlib.exception.UartLibException,
       cn.wch.uartlib.exception.NoPermissionException,
       cn.wch.uartlib.exception.ChipException
```

打开 USB 设备

参数	usbDevice - USB 设备
返回	true 成功; false 失败
抛出	cn.wch.uartlib.exception.UartLibException cn.wch.uartlib.exception.NoPermissionException cn.wch.uartlib.exception.ChipException

6、requestPermission

```
public void requestPermission(@NonNull
                             android.content.Context context,
                             @NonNull
                             android.hardware.usb.UsbDevice usbDevice)
throws cn.wch.uartlib.exception.UartLibException
```

请求 USB 设备的打开权限

参数	context - 上下文 usbDevice - USB 设备
抛出	cn.wch.uartlib.exception.UartLibException

7、setUsbStateListener

```
public void setUsbStateListener(@NonNull
                                cn.wch.uartlib.callback.IUsbStateChange
                                usbStateListener)
```

监听设备的状态变化

参数	usbStateListener - 设备状态监听回调
----	-----------------------------

8、getSerialCount

```
public int getSerialCount(@NonNull
                          android.hardware.usb.UsbDevice usbDevice)
```

获取设备的串口数目

参数	usbDevice - USB 设备
返回	返回串口数目;如果为负, 说明获取芯片类型失败

9、setSerialParameter

```
public boolean setSerialParameter(@NonNull
    android.hardware.usb.UsbDevice usbDevice,
    int serialNumber,
    int baud,
    int dataBit,
    int stopBit,
    int parityBit,
    boolean flow)
    throws java.lang.Exception,
```

设置串口参数

参数	usbDevice - USB 设备 serialNumber - 串口号 baud - 波特率 dataBit - 数据位 5, 6, 7, 8 stopBit - 停止位 1, 2 parityBit - 校验位 0 NONE; 1 ODD; 2 EVEN; 3 MARK; 4 SPACE flow - true 开启; false 关闭
返回	true 设置成功; false 设置失败
抛出	java.lang.Exception

10、writeData

```
public int writeData(@NonNull
    android.hardware.usb.UsbDevice usbDevice,
    int serialNumber,
    byte[] data,
    int length,
    int timeout)
    throws java.lang.Exception,
```

发送串口数据

参数	usbDevice - USB 设备 serialNumber - 串口号 data - 待发送的数据 length - 待发送的数据的长度 timeout - 超时时间
返回	发送成功的数据的长度
抛出	java.lang.Exception

11、readData

```
public byte[] readData(@NonNull
                        android.hardware.usb.UsbDevice usbDevice,
                        int serialNumber)
    throws java.lang.Exception
```

主动读取数据

参数	usbDevice - USB 设备 serialNumber - 串口号
返回	读取到的数据
抛出	cn.wch.uartlib.exception.ChipException

12、registerDataCallback

```
public void registerDataCallback (@NonNull
                                   android.hardware.usb.UsbDevice usbDevice,
                                   cn.wch.uartlib.callback.IDataCallback dataCallback)
    throws java.lang.Exception
```

注册串口数据回调。此方法可代替 readData 方法。如果注册此回调，数据优先通过该回调方式回传，推荐使用该方法接收数据。解除注册使用 registerDataCallback(device, null) 方法，或者 removeDataCallback(device) 方法。

参数	usbDevice - USB 设备 dataCallback - 数据回调
抛出	java.lang.Exception

13、removeDataCallback

```
public void removeDataCallback (@NonNull
                                 android.hardware.usb.UsbDevice usbDevice)
```

解除注册串口数据回调

参数	usbDevice - USB 设备
----	--------------------

14、isConnected

```
public boolean isConnected(@NonNull
                            android.hardware.usb.UsbDevice usbDevice)
```

判断 USB 设备是否已经被打开

参数	usbDevice - USB 设备
返回	true 已经被打开;false 没有被打开

15、getConnectedDevices

```
public java.util.ArrayList<android.hardware.usb.UsbDevice> getConnectedDevices()
```

获取当前已经被打开的设备

返回	已经被打开的设备列表
----	------------

16、disconnect

```
public void disconnect(@NonNull
                        android.hardware.usb.UsbDevice usbDevice)
```

断开 USB 设备连接

参数	usbDevice - USB 设备
----	--------------------

17、close

```
public void close(@NonNull Context context)
```

释放资源。断开所有连接设备,注销广播

参数	context -上下文
----	--------------

18、isSupportGPIOFeature

```
public boolean isSupportGPIOFeature(UsbDevice device)
                                throws java.lang.Exception
```

检查本库目前是否支持该硬件设备 GPIO 特性的配置,应当于操作 GPIO 前调用

参数	device -USB 设备
抛出	java.lang.Exception
返回	true 支持;false 不支持

19、queryGPIOCount

```
public int queryGPIOCount(UsbDevice device)
                        throws java.lang.Exception
```

查询该 USB 设备的 GPIO 数量

参数	device - USB 设备
抛出	java.lang.Exception
返回	GPIO 数量

20、queryGPIOStatus

```
public GPIO_Status queryGPIOStatus(UsbDevice device, int gpioIndex)
                                throws java.lang.Exception
```

查询该 USB 设备的某个 GPIO 状态

参数	device - USB 设备 gpioIndex - GPIO 序号, 从 0 开始
抛出	java.lang.Exception
返回	GPIO 状态

21、queryAllGPIOStatus

```
public List<GPIO_Status> queryAllGPIOStatus(UsbDevice device)
                                throws java.lang.Exception
```

查询该 USB 设备的所有 GPIO 状态

参数	device - USB 设备
抛出	java.lang.Exception
返回	所有 GPIO 状态

22、enableGPIO

```
public boolean enableGPIO(UsbDevice device, int gpioIndex, boolean enable,
                           GPIO_DIR dir)
                                throws java.lang.Exception
```

使能该硬件设备的某个 GPIO

参数	device - USB 设备 gpioIndex- GPIO 序号 enable- true 打开;false 关闭 dir- GPIO 方向
抛出	java.lang.Exception
返回	true 操作成功;false 操作失败

23、setGPIOVal

```
public boolean setGPIOVal(UsbDevice device, int gpioIndex, GPIO_VALUE value)
```

throws java.lang.Exception

设置该硬件设备的某个 GPIO 的电平值

参数	device - USB 设备 gpioIndex- GPIO 序号 value - GPIO 电平值
抛出	java.lang.Exception
返回	true 操作成功;false 操作失败

24、getGPIOVal

```
public GPIO_VALUE getGPIOVal (UsbDevice device, int gpioIndex)
```

throws java.lang.Exception

获取该硬件设备的某个 GPIO 的电平值

参数	device - USB 设备 gpioIndex- GPIO 序号
抛出	java.lang.Exception
返回	value- GPIO 电平值

25、setDTR

```
public boolean setDTR (@NonNull UsbDevice usbDevice, int serialNumber, boolean valid)
```

throws Exception

设置 DTR 信号

参数	device - USB 设备 serialNumber- 串口号 valid - 是否有效 (低电平有效)
抛出	java.lang.Exception
返回	true 操作成功;false 操作失败

26、setRTS

```
public boolean setRTS (@NonNull UsbDevice usbDevice, int serialNumber, boolean valid)
```

throws Exception

设置 RTS 信号

参数	device - USB 设备 serialNumber- 串口号 valid - 是否有效 (低电平有效)
抛出	java.lang.Exception
返回	true 操作成功;false 操作失败

27、setBreak

```
public boolean setBreak(@NonNull UsbDevice usbDevice, int serialNumber, boolean valid)
throws Exception
```

设置 Break 信号

参数	device - USB 设备 serialNumber- 串口号 valid - 是否有效(低电平有效)
抛出	java.lang.Exception
返回	true 操作成功;false 操作失败

28、registerModemStatusCallback

```
public void registerModemStatusCallback(@NonNull UsbDevice usbDevice, IModemStatus
modemStatus) throws Exception
```

注册 Modem 输入信号状态的回调

参数	device - USB 设备 modemStatus- 状态回调
抛出	java.lang.Exception

29、querySerialErrorCount

```
public int querySerialErrorCount(@NonNull UsbDevice usbDevice, int
serialNumber, @NonNull SerialErrorType errorType) throws Exception
```

查询串口错误状态

参数	device - USB 设备 serialNumber- 串口号 errorType - 错误类型
抛出	java.lang.Exception
返回	出现错误的次数

30、setReadTimeout

```
public static void setReadTimeout(int timeout)
```

设置读超时时间。默认是 0, 读取方式使用 USBRequest 异步传输 ;如果不为 0 读取方式则使用同步传输, 超时时间为 BulkTransfer 同步传输的超时时间。全局有效, 应该在 APP 初始化时调用。

参数	timeout - 超时时间, 单位为 ms
----	------------------------

31、addNewHardware

```
public static void addNewHardware(int vid, int pid)
```

该方法针适用于用户修改了硬件的 VID 和 PID 的情况，需要用户新增修改后的 VID 和 PID。

参数	vid - 硬件 vid pid - 硬件 pid
-----------	------------------------------

32、setDebug

```
public static void setDebug(boolean open)
```

设置调试模式打开或者关闭。打开调试模式会打印日志。默认关闭。应该在 APP 初始化时调用。

参数	open - true 打开;false 关闭
-----------	-------------------------

33、isDebugMode

```
public static boolean isDebugMode()
```

返回当前是否处于调试模式，是否会打印日志。

返回	true 处于调试模式;false 不处于调试模式
-----------	---------------------------