

# CH34X Serial port Android Application Development Manual

Version: 1.7

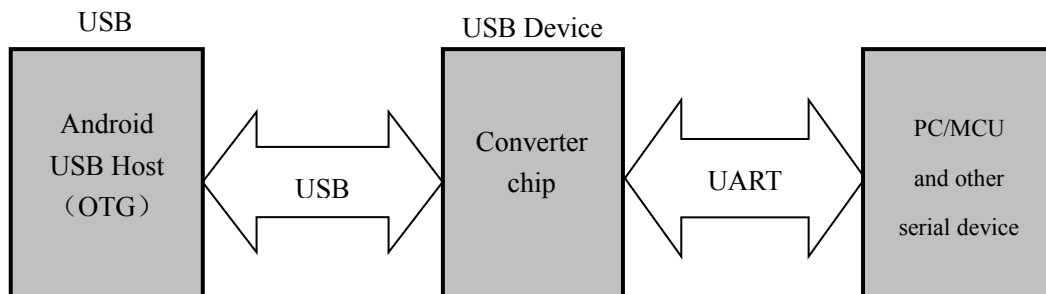
<http://wch.cn>

## Introduction

This document is for the CH340 / CH341 / CH342 / CH343 / CH344 / CH347 / CH9101 / CH9102 / CH9103 / CH9104 / CH9143 USB to serial port android library development instructions document.

This document mainly introduces how to use the chip's USB to asynchronous serial function (hereinafter referred to as CH34XUART) and GPIO function, and how to use APK operation in Android to achieve serial communication. This function is based on Android USB Host protocol, users can call the relevant interface API to achieve communication with Android devices.

The relationship between Android Host, USB Device and serial device is as follows.



The Android interface provided by CH34X serial port needs to be based on Android 4.4 and above, and the conditions for using CH34X serial port Android driver are as follows:

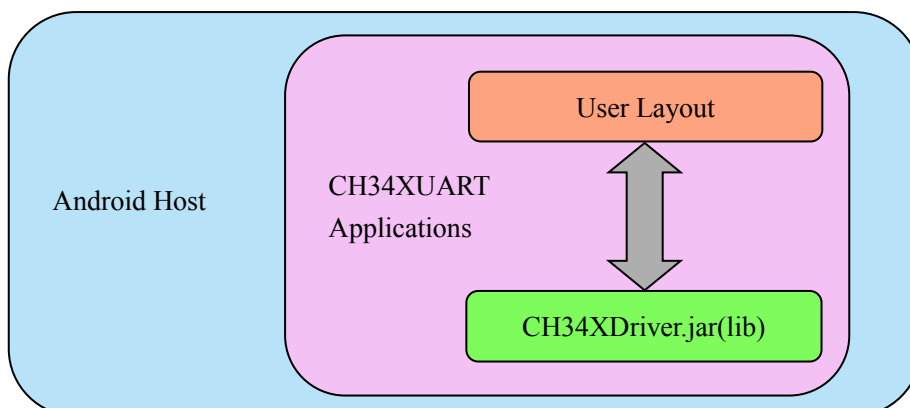
1. Based on Android 4.4 and above system versions
2. Android device with USB Host or OTG interface

This document will focus on the communication interface API between Android USB Host and Device and the operation instructions of the test program.

For Android USB Host protocol description, please refer to the official Google document.

## 1. Android Host

The example programs described in this document are all written in Android 4.4 and above. The start parameters of this Android application are product-id and vendor-id defined in device\_filter.xml file. The Android application developed based on CH34X UART is divided into two parts, as follows:



## 2. Software operation description

Users need to install the test software (i.e. CH34XUARTDemo.apk) provided by our company on the Android devices that support USB Host function. After clicking Scan Device, the pop-up box will show all the devices connected on the current Android device. Click on a device to open it, if there is no corresponding USB access permission, the system will automatically pop up the permission request window.

After entering the software, first set the serial port parameters, including baud rate, data bits, stop bits, parity bits and hardware flow control, etc. After that, you can perform data sending and receiving operations.

## 3. Function interface description

### 3.1. getInstance

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

Use to get the global unique instance.

<b>Return</b>	Return the global unique instance
---------------	-----------------------------------

### 3.2. init

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

Initialize the context and register dynamic broadcasts to listen for device state changes.

<b>Parameter</b>	application - Global contexts
------------------	-------------------------------

### 3.3. enumDevice

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

throws java.lang.Exception

Enumerate all current USB devices that match the requirements.

<b>Throw</b>	java.lang.Exception Exception
--------------	-------------------------------

### 3.4. getChipType

```
public cn.wch.uartlib.chipImpl.type.ChipType2 getChipType(@NonNull  
    android.hardware.usb.UsbDevice usbDevice)
```

Get the chip type of the UsbDevice.

<b>Parameter</b>	usbDevice – USB device
<b>Return</b>	If the chip type is null, it means that the chip type of the USB device cannot be recognized

### 3.5. 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

Open USB device.

<b>Parameter</b>	usbDevice – USB device
<b>Return</b>	true: Successful ; false: Failure
<b>Throw</b>	cn.wch.uartlib.exception.UartLibException cn.wch.uartlib.exception.NoPermissionException cn.wch.uartlib.exception.ChipException

### 3.6. requestPermission

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

Request open permission for USB devices.

<b>Parameter</b>	context –The context usbDevice – USB device
<b>Throw</b>	cn.wch.uartlib.exception.UartLibException

### 3.7. setUsbStateListener

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

Listen to device status changes.

<b>Parameter</b>	usbStateListener –Device status listen callback
------------------	---

### 3.8. getSerialCount

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

Get the number of device serial ports.

<b>Parameter</b>	usbDevice – USB device
<b>Return</b>	Return the number of serial ports; if it is negative, it means the chip type failed to be obtained.

### 3.9. 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,
```

Set the serial port Parameters

<b>Parameter</b>	usbDevice – USB device serialNumber – Serial port number baud – Baud rate dataBit – Data bits 5,6,7,8 stopBit - Stop bits 1,2 parityBit – Parity bits 0 NONE;1 ODD;2 EVEN;3 MARK;4 SPACE flow – true: Open; false: Close
<b>Return</b>	True: Setting successful ; false: Setting Failure
<b>Throw</b>	java.lang.Exception

### 3.10. writeData

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

Send the serial port data

<b>Parameter</b>	usbDevice – USB device serialNumber – Serial port number data - Data to be sent length - Length of the data to be sent timeout - Timeout
<b>Return</b>	The length of the data sent successfully
<b>Throw</b>	java.lang.Exception

### 3.11. readData

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

Active read data

<b>Parameter</b>	usbDevice – USB device serialNumber – Serial port number
<b>Return</b>	Data already read

<b>Throw</b>	cn.wch.uartlib.exception.ChipException
--------------	--

### 3.12. registerDataCallback

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

Register the serial port data callback. This function can be used instead of the readData function. If you register this callback, data will be returned by this callback function first, so it is recommended to use this function to receive data. To unregister, use registerDataCallback(device,null) or removeDataCallback(device).

<b>Parameter</b>	usbDevice – USB device dataCallback –Data callback
<b>Throw</b>	java.lang.Exception

### 3.13. removeDataCallback

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

Unregister serial port data callback.

<b>Parameter</b>	usbDevice – USB device
------------------	------------------------

### 3.14. isConnected

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

Determine if the device is connected or not.

<b>Parameter</b>	usbDevice – USB device
<b>Return</b>	true: Connected; false: Not connected

### 3.15. getConnectedDevices

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

Get the currently connected devices.

<b>Return</b>	List of devices that have been turned on
---------------	--

### 3.16. disconnect

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

Disconnecting USB devices

<b>Parameter</b>	usbDevice – USB device
------------------	------------------------

### 3.17. close

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

Release resources, disconnect all devices, and log off broadcasts.

<b>Parameter</b>	context –The context
------------------	----------------------

### 3.18. isSupportGPIOFeature

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

Check whether this library currently supports the configuration of the GPIO features of this hardware device, which should be called before operating the GPIO.

<b>Parameter</b>	device -USB device
<b>Throw</b>	java.lang.Exception
<b>Return</b>	true: support; false: No support

### 3.19. queryGPIOCount

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

Query the GPIO number of this USB device.

<b>Parameter</b>	device – USB device
<b>Throw</b>	java.lang.Exception
<b>Return</b>	The number of GPIO

### 3.20. queryGPIOStatus

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

Query a GPIO status of this USB device

<b>Parameter</b>	device – USB device gpioIndex – GPIO serial number, start from 0
<b>Throw</b>	java.lang.Exception
<b>Return</b>	GPIO status

### 3.21. queryAllGPIOStatus

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

Query all GPIO status of this USB device.

<b>Parameter</b>	device – USB device
<b>Throw</b>	java.lang.Exception
<b>Return</b>	All GPIO status

### 3.22. enableGPIO

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

Enable a GPIO of this hardware device.

<b>Parameter</b>	device – USB device gpioIndex- GPIO serial number enable- true: Open; false: Close dir- GPIO direction
<b>Throw</b>	java.lang.Exception
<b>Return</b>	true: Successful; false: Failure

### 3.23. setGPIOVal

```
public boolean setGPIOVal(UsbDevice device,int gpioIndex, GPIO_VALUE value)  
                           throws java.lang.Exception
```

Set a GPIO level of this hardware device.

<b>Parameter</b>	device – USB device gpioIndex- GPIO serial number value – GPIO level value
<b>Throw</b>	java.lang.Exception
<b>Return</b>	true: Successful; false: Failure

### 3.24. getGPIOVal

```
public GPIO_VALUE getGPIOVal(UsbDevice device,int gpioIndex)  
                           throws java.lang.Exception
```

Get a GPIO level of this hardware device.

<b>Parameter</b>	device - USB device gpioIndex- GPIO serial number
<b>Throw</b>	java.lang.Exception
<b>Return</b>	value- The GPIO level value

### 3.25. setDTR

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

Set the DTR signal

<b>Parameter</b>	device - USB device serialNumber- Serial port number valid – Valid or not (valid at low)
<b>Throw</b>	java.lang.Exception
<b>Return</b>	true: Successful; false: Failure

### 3.26. setRTS

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

Set the RTS signal

<b>Parameter</b>	device - USB device serialNumber- Serial port number valid - Valid or not (valid at low)
<b>Throw</b>	java.lang.Exception
<b>Return</b>	true: Successful; false: Failure

### 3.27. setBreak

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

Set the Break signal

<b>Parameter</b>	device - USB device serialNumber- Serial port number valid - Valid or not (valid at low)
<b>Throw</b>	java.lang.Exception
<b>Return</b>	true: Successful; false: Failure

### 3.28. registerModemStatusCallback

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

Register callback of Modem input signal status

<b>Parameter</b>	device - USB device modemStatus- Status callback
<b>Throw</b>	java.lang.Exception

### 3.29. querySerialErrorCount

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

Query the error status of serial port.

<b>Parameter</b>	device - USB device serialNumber- Serial port number errorType – Error type
<b>Throw</b>	java.lang.Exception
<b>Return</b>	The number of errors

### 3.30. setReadTimeout

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

Set the read timeout. Default is 0, use the USBRequest asynchronous transfer mode to read; if not 0, use

synchronous transfer mode to read, timeout time is BulkTransfer synchronous transfer timeout time. Valid globally, should be called during APP initialization.

<b>Parameter</b>	timeout –Timeout, in ms
------------------	-------------------------

### 3.31. addNewHardware

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

This function is suitable for the case that the user has modified the VID and PID of the hardware, and requires the user to add the modified VID and PID.

<b>Parameter</b>	vid–Hardware vid pid–Hardware pid
------------------	--------------------------------------

### 3.32. setDebug

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

Set debug mode to on or off. Turning on debug mode will print the log. Turn off by default. Should be called during APP initialization.

<b>Parameter</b>	open–true: Open; false: Close
------------------	-------------------------------

### 3.33. isDebugMode

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

Return the current debug mode status to determine if the log will be printed.

<b>Return</b>	true: Debug mode; false: No debug mode
---------------	--