

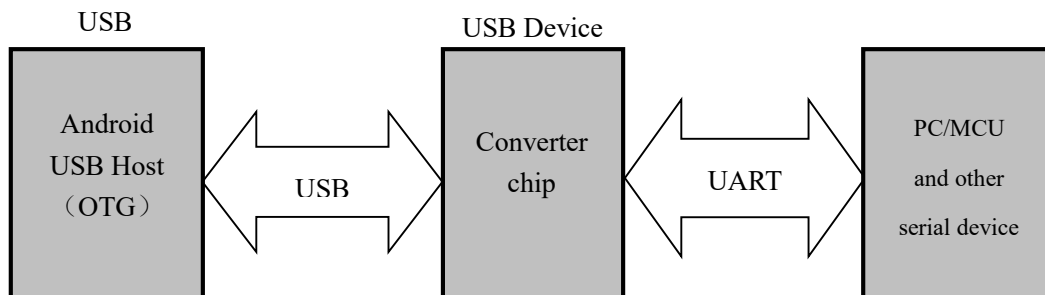
CH34X Serial port Android Application Development Manual

1. Introduction

This document is for the CH339 / CH340 / CH341 / CH342 / CH343 / CH344 / CH347 / CH9101 / CH9102 / CH9103 / CH9104 / CH9143/ CH9111 / CH9114 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:

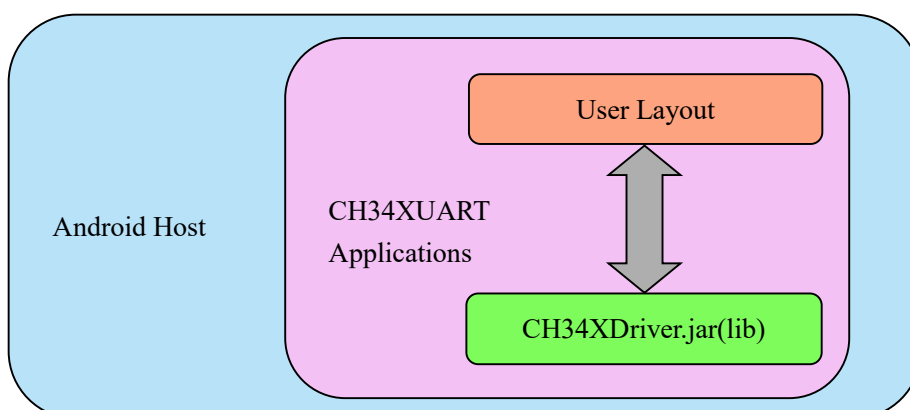
1. Based on Android 4.4 and above system versions
2. Android device with USB Host or OTG interface

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.

2. 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:



3. 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.

4. Function interface description

4.1. getInstance

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

Use to get the global unique instance.

Return	Return the global unique instance
---------------	-----------------------------------

4.2. init

```
public void init(Application application)
```

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

Parameter	application - Global contexts
------------------	-------------------------------

4.3. enumDevice

```
public ArrayList<UsbDevice> enumDevice() throws Exception
```

Enumerate all current USB devices that match the requirements.

Return	UsbDevice Device List
Throw	Exception

4.4. getChipType

```
public ChipType2 getChipType(@NonNull UsbDevice usbDevice)
```

Get the chip type of the UsbDevice.

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

4.5. openDevice

```
public boolean openDevice(@NonNull UsbDevice usbDevice)
```

throws UartLibException,
NoPermissionException,
ChipException

Open USB device.

Parameter	usbDevice – USB device
Return	true: Successful ; false: Failure
Throw	UartLibException NoPermissionException ChipException

4.6. requestPermission

```
public void requestPermission(@NonNull Context context,@NonNull UsbDevice usbDevice)  
    throws UartLibException
```

Request open permission for USB devices.

Parameter	context –The context usbDevice – USB device
Throw	UartLibException

4.7. setUsbStateListener

```
public void setUsbStateListener(@NonNull IUsbStateChange usbStateListener)
```

Listen to device status changes.

Parameter	usbStateListener –Device status listen callback
------------------	---

4.8. getSerialCount

```
public int getSerialCount(@NonNull UsbDevice usbDevice)
```

Get the number of device serial ports.

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

4.9. enableSerial

```
public boolean enableSerial(@NonNull UsbDevice usbDevice,int serialNumber,  
    boolean enable) throws Exception
```

Open and close the serial port (actually only valid for CH9114 series, no need to call other types of devices).

Parameter	usbDevice – USB device serialNumber– Serial port number enable – false close; true open
Return	True: Setting successful; False: Setting failed.
Throw	Exception

4.10. setSerialParameter

```
public boolean setSerialParameter(@NonNull UsbDevice usbDevice,
    int serialNumber,
    int baud,
    int dataBit,
    int stopBit,
    int parityBit,
    boolean flow)
    throws Exception,
```

Set the serial port Parameters

Parameter	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
Return	True: Setting successful ; false: Setting failed
Throw	Exception

4.11. getSerialBaud

```
public int getSerialBaud(@NonNull UsbDevice usbDevice,int serialNumber) throws Exception
```

Get the serial port baud rate (actually only valid for CH9114 series, no need to call other types of devices)

Parameter	usbDevice – USB device serialNumber – Serial port number
Return	When the return value is greater than 0, it indicates the serial port baud rate; when it is less than 0, an error occurs.
Throw	Exception

4.12. getChipMasterFrequency

```
public ChipMasterFrequency getChipMasterFrequency(@NonNull UsbDevice usbDevice) throws Exception
```

Get the chip master frequency (actually only valid for CH9114 series, no need to call other types of devices)

Parameter	usbDevice – USB device serialNumber – Serial port number
Return	ChipMasterFrequency : int frequency - master Frequency boolean switchEnable - Whether to allow switching int CoStatus - Crystal status
Throw	Exception

4.13. syncWriteData

```
public int syncWriteData(@NonNull UsbDevice usbDevice, int serialNumber, byte[] data, int length,  
                        int timeout) throws Exception
```

Send serial port data (synchronous sending)

Parameter	usbDevice – USB device serialNumber – Serial port number data - Data to be sent length - Length of the data to be sent timeout - Timeout
Return	The length of the data sent successfully
Throw	Exception

4.14. asyncWriteData

```
public void asyncWriteData(@NonNull UsbDevice usbDevice, int serialNumber,  
                           byte[] data) throws Exception
```

Sending serial data (asynchronous sending, adding data to cache and continuously sending, unable to return status and results)

Parameter	usbDevice – USB device serialNumber – Serial port number data - Data to be sent
Throw	Exception

4.15. readData

```
public byte[] readData(@NonNull  
                      UsbDevice usbDevice,  
                      int serialNumber)  
    throws Exception
```

Active read data

Parameter	usbDevice – USB device serialNumber – Serial port number
Return	Data already read
Throw	ChipException

4.16. readData

```
public byte[] readData(@NonNull  
                      UsbDevice usbDevice,  
                      int serialNumber,  
                      int vTime,
```

int vMin)

throws Exception

Active read data

1. When vTime>0, vMin>0. The read call will remain blocked until the first character is read, and the timing will start after the first character is read. After that, if the time reaches vTime or the time has not expired but vMin characters have been read, it will return; if the time has not expired, If another character is read before (but the total number read at this time is still not enough for vMin), the timing starts again.
2. When vTime>0, vMin=0. The read call returns immediately after reading the data, otherwise it will wait up to vTime for each character.
3. When vTime=0, vMin>0. The read call blocks until vMin characters are read and returns immediately.

Parameter	usbDevice - USB device serialNumber - Serial port number vTime - Waiting time vMin - Minimum number of reads
Return	Data already read
Throw	ChipException

4.17. registerDataCallback

public void registerDataCallback (@NonNull UsbDevice usbDevice, IDataCallback dataCallback)
throws 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).

Parameter	usbDevice – USB device dataCallback –Data callback
Throw	java.lang.Exception

4.18. removeDataCallback

public void removeDataCallback (@NonNull UsbDevice usbDevice)

Unregister serial port data callback.

Parameter	usbDevice – USB device
------------------	------------------------

4.19. isConnected

public boolean isConnected(@NonNull UsbDevice usbDevice)

Determine if the device is connected or not.

Parameter	usbDevice – USB device
Return	true: Connected; false: Not connected

4.20. getConnectedDevices

```
public ArrayList<UsbDevice> getConnectedDevices()
```

Get the currently connected devices.

Return	List of devices that have been turned on
---------------	--

4.21. disconnect

```
public void disconnect(@NonNull UsbDevice usbDevice)
```

Disconnecting USB devices

Parameter	usbDevice – USB device
------------------	------------------------

4.22. close

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

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

Parameter	context –The context
------------------	----------------------

4.23. isSupportGPIOFeature

```
public boolean isSupportGPIOFeature(UsbDevice device) throws 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.

Parameter	device -USB device
Throw	Exception
Return	true: support; false: No support

4.24. queryGPIOCount

```
public int queryGPIOCount(UsbDevice device) throws Exception
```

Query the GPIO number of this USB device.

Parameter	device – USB device
Throw	Exception
Return	The number of GPIO

4.25. queryGPIOStatus

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

Query a GPIO status of this USB device

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

4.26. queryAllGPIOStatus

public List<GPIO_Status> queryAllGPIOStatus(UsbDevice device) throws Exception
Query all GPIO status of this USB device.

Parameter	device – USB device
Throw	Exception
Return	All GPIO status

4.27. enableGPIO

public boolean enableGPIO(UsbDevice device,int gpioIndex, boolean enable,GPIO_DIR dir)
throws Exception

Enable a GPIO of this hardware device.

Parameter	device – USB device gpioIndex- GPIO serial number enable- true: Open; false: Close dir- GPIO direction
Throw	Exception
Return	true: Successful; false: Failure

4.28. setGPIOVal

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

Set a GPIO level of this hardware device.

Parameter	device – USB device gpioIndex- GPIO serial number value – GPIO level value
Throw	.Exception
Return	true: Successful; false: Failure

4.29. getGPIOVal

public GPIO_VALUE getGPIOVal(UsbDevice device,int gpioIndex) throws Exception
Get a GPIO level of this hardware device.

Parameter	device - USB device gpioIndex- GPIO serial number
Throw	Exception
Return	value- The GPIO level value

4.30. setDTR

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

Set the DTR signal

Parameter	device - USB device serialNumber- Serial port number valid – Valid or not (valid at low)
Throw	Exception
Return	true: Successful; false: Failure

4.31. setRTS

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

throws Exception

Set the RTS signal

Parameter	device - USB device serialNumber- Serial port number valid - Valid or not (valid at low)
Throw	Exception
Return	true: Successful; false: Failure

4.32. setBreak

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

throws Exception

Set the Break signal

Parameter	device - USB device serialNumber- Serial port number valid - Valid or not (valid at low)
Throw	Exception
Return	true: Successful; false: Failure

4.33. registerModemStatusCallback

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

throws Exception

Register callback of Modem input signal status

Parameter	device - USB device modemStatus- Status callback
Throw	Exception

4.34. querySerialErrorCount

```
public int querySerialErrorCount(@NonNull UsbDevice usbDevice,int serialNumber,
```

@NonNull SerialErrorType errorType) throws Exception

Query the error status of serial port.

Parameter	device - USB device serialNumber- Serial port number errorType – Error type
Throw	Exception
Return	The number of errors

4.35. 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.

Parameter	timeout –Timeout, in ms
------------------	-------------------------

4.36. 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.

Parameter	vid–Hardware vid pid–Hardware pid
------------------	--------------------------------------

4.37. 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.

Parameter	open–true: Open; false: Close
------------------	-------------------------------

4.38. isDebugMode

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

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

Return	true: Debug mode; false: No debug mode
---------------	--