

# CH934X Serial ports Android Application Development Manual

## 1. Introduction

This document is for the CH934X / CH348 USB to serial port android library development instructions document.

The Android interface provided by CH934X serial port needs to be based on Android 4.4 and above, and the conditions for using CH934X 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

## 2. interface description

### 2.1. getInstance

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

Use to get the global unique instance.

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

### 2.2. init

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

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

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

### 2.3. enumDevice

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

Enumerate the current CH934X devices.

<b>Return</b>	UsbDevice Device List
<b>Throw</b>	UartLibException

### 2.4. getChipType

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

Get the chip type of the UsbDevice.

<b>Parameter</b>	usbDevice – USB device
<b>Return</b>	If it is null, it means the parameter USB device is not CH934X

### 2.5. openDevice

```
public boolean openDevice(@NonNull UsbDevice usbDevice)  
    throws UartLibException,  
    NoPermissionException,  
    ChipException
```

Open USB device.

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

## 2.6. requestPermission

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

Request open permission for USB devices.

<b>Parameter</b>	context –The context usbDevice – USB device
<b>Throw</b>	UartLibException

## 2.7. setUsbStateListener

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

Listen to device status changes.

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

## 2.8. getSerialCount

```
public int getSerialCount(@NonNull 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.

## 2.9. setSerialParameter

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

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 failed
<b>Throw</b>	UartLibException ChipException

## 2.10. writeData

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

throws UartLibException,

ChipException

Send 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>	UartLibException ChipException

## 2.11. readData

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

Read data

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

## 2.12. registerDataCallback

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

throws ChipException

Register serial port data callback, unregister using registerDataCallback(device,null) method, or removeDataCallback(device) method.

<b>Parameter</b>	usbDevice – USB device dataCallback –Data callback
<b>Throw</b>	ChipException

### 2.13. removeDataCallback

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

Unregister serial port data callback.

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

### 2.14. setBreak

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

throws Exception

Set the Break signal

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

### 2.15. setDTR

```
public boolean setDTR(UsbDevice usbDevice,int serialNumber,boolean dtr)
```

throws UartLibException, ChipException

Set the DTR signal

<b>Parameter</b>	usbDevice- USB device serialNumber- Serial port number dtr– True :valid False: invalid
<b>Return</b>	true: Successful; false: Failure
<b>Throw</b>	UartLibException ChipException

### 2.16. setRTS

```
public boolean setRTS(UsbDevice usbDevice,int serialNumber,boolean rts)
```

throws UartLibException, ChipException

Set the RTS signal

<b>Parameter</b>	usbDevice- USB device serialNumber- Serial port number rts– True :valid False: invalid
------------------	--

<b>Throw</b>	UartLibException ChipException
<b>Return</b>	true: Successful; false: Failure

## 2.17 registerModemStatusCallback

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

Register callback of Modem input signal status

<b>Parameter</b>	usbDevice- USB device modemStatus- Status callback
<b>Throw</b>	Exception

## 2.18. querySerialErrorCount

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

Query the error status of serial port.

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

## 2.19. getCurrentMode

```
public Mode getCurrentMode(UsbDevice usbDevice, int serialNumber) throws
                           UartLibException
```

Get the current serial port mode. The initial default state is normal mode (only for CH934X model devices, CH348 is invalid).

<b>Parameter</b>	usbDevice- USB device serialNumber- Serial port number
<b>Return</b>	Mode.NORMAL : Normal mode; Mode.HARDFLOW : Hardware flow control mode; Mode.GPIO : GPIO mode;
<b>Throw</b>	UartLibException

## 2.20. getSpecificType

```
public SpecificChipType getSpecificType(UsbDevice usbDevice) throws UartLibException
```

Get the specific chip model.

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

<b>Return</b>	Chip specific model
<b>Throw</b>	UartLibException

## 2.21. getGPIOCount

public int getGPIOCount(UsbDevice usbDevice) throws UartLibException

Get the number of GPIO.

<b>Parameter</b>	usbDevice- USB device
<b>Return</b>	Specific number of GPIO (starting from 0)
<b>Throw</b>	UartLibException

## 2.22. getGPIOGroup

public int getGPIOGroup(UsbDevice usbDevice) throws UartLibException

Get GPIO group.

<b>Parameter</b>	usbDevice- USB device
<b>Return</b>	GPIO group number (starting from 0)
<b>Throw</b>	UartLibException

## 2.23. enableGPIO

public boolean enableGPIO(@NonNull UsbDevice usbDevice, ChipType chipType, int  
gpioGroup,  
int enable) throws UartLibException

Enable the GPIO function of the serial port. If the current mode is hardware flow control mode, you need to exit first.

<b>Parameter</b>	usbDevice- USB device chipType- Chip Type gpioGroup- GPIO group number enable- Enable status CH9344: 1 - enables all GPIOs in the group; 0 - disables all GPIOs in the group CH348: bits 0-7 correspond to GPIO [0*N-7*N], 1 - enables; 0 - disables
<b>Return</b>	true: Successful; false: Failure
<b>Throw</b>	UartLibException

## 2.24. setGPIONDir

public boolean setGPIONDir(@NonNull UsbDevice usbDevice,int gpioGroup, int  
gpioNumber,  
@NonNull GPIO\_DIR dir) throws UartLibException

Set the direction of the GPIO port. If successful, it will be synchronized to the cache.

<b>Parameter</b>	usbDevice– USB device gpioGroup - GPIO group number gpioNumber – GPIO number dir - direction
<b>Return</b>	true: Successful; false: Failure
<b>Throw</b>	UartLibException

## 2.25. queryGPIODirFromCache

```
public GPIO_DIR queryGPIODirFromCache(@NonNull UsbDevice usbDevice, int
                                     gpioGroup,
                                     int gpioNumber) throws UartLibException
```

Get the direction of a GPIO from the cache. The initial default direction of each GPIO is GPIO\_DIR.IN

<b>Parameter</b>	usbDevice– USB device gpioGroup - GPIO group number gpioNumber – GPIO number
<b>Return</b>	GPIO_DIR.IN : IN direction; GPIO_DIR.OUT : OUT direction
<b>Throw</b>	UartLibException

## 2.26. setGPIOValue

```
public boolean setGPIOValue(@NonNull UsbDevice usbDevice, int gpioGroup, int
                             gpioNumber,
                             @NonNull GPIO_VALUE value) throws UartLibException
```

Set the value of the GPIO port, which will be synchronized to the cache if successful

<b>Parameter</b>	usbDevice– USB device gpioGroup - GPIO group number gpioNumber – GPIO number value – GPIO value
<b>Return</b>	true: Successful; false: Failure
<b>Throw</b>	UartLibException

## 2.27. getGPIOValue

```
public boolean getGPIOValue(@NonNull UsbDevice usbDevice) throws UartLibException
```

Get GPIO value from hardware, and refresh cache if successful.

<b>Parameter</b>	usbDevice- USB device
<b>Return</b>	true: Successful; false: Failure
<b>Throw</b>	UartLibException

## 2.28. queryGPIOValueFromCache

```
public GPIO_VALUE queryGPIOValueFromCache(@NonNull UsbDevice usbDevice,int
gpioGroup,
int gpioNumber) throws
UartLibException
```

Get a GPIO value from the cache. Before using this method, you need to successfully use the getGPIOValue() method to refresh the cache. The initial default value of each GPIO is GPIO\_VALUE.LOW

<b>Parameter</b>	usbDevice- USB device gpioGroup - GPIO group number gpioNumber – GPIO number
<b>Return</b>	GPIO_VALUE.HIGH : high level; GPIO_VALUE.LOW : low level
<b>Throw</b>	UartLibException

## 2.29. isConnected

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

Determine if the device is connected or not.

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

## 2.30. getConnectedDevices

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

Get the currently connected devices.

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

## 2.31. disconnect

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

Disconnecting USB devices

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

## 2.32. close

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

Release resources, disconnect all devices.

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