CH34x series chip serial port function Android program development manual

Version: 1.3

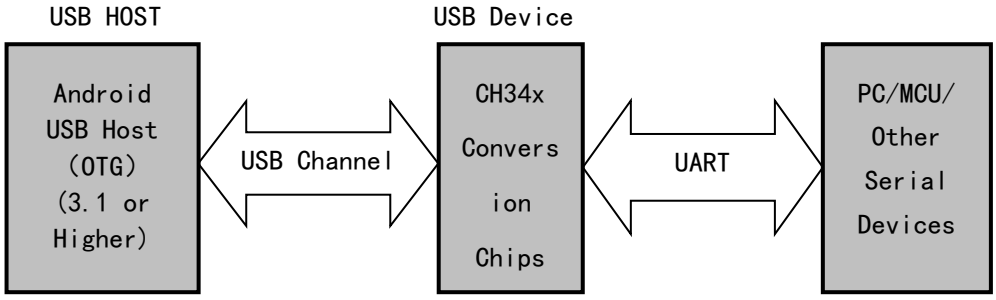
Introduction

CH340 is a USB bus conversion chip, it can realize USB to UART interface or USB to printer interface.

In serial UART mode, CH340 provides common MODEM liaison signal, used to expand UART interface of computer or upgrade the common serial device to USB bus directly. For more information on USB conversion to printer interface please refer to the manual CH340DS2.

This document describes the CH340/CH341 asynchronous serial port function(referred to as CH34x UART),and how to operate CH34x to achieve serial communication with APK. This function is based on the Android USB Host protocol,users can call the relevant API to achieve communication with the Android devices.

Android Host, USB Device, Serial Device, the relationship between the three as shown below:



The Android API interfaces which operate CH34x serial ports are based on Android 3.1 and above system, the conditions of using CH34x serial port Android driver:

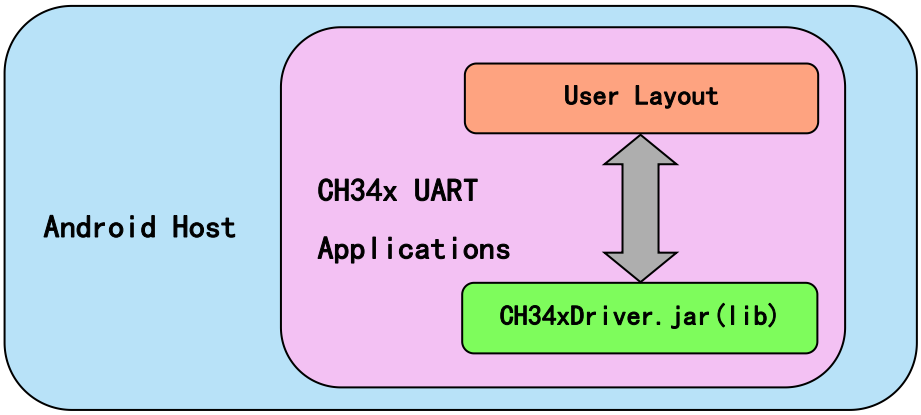
1. Need to be based on Android 3.1 and above system
2. Android devices have a USB Host or OTG interface

This document will focus on the communication API interface between Android USB Host and Device and operating instructions for the DEMO.

For Android USB Host protocol, you can refer to the Google documentation.

1. Android Host

The examples described in this document are written under Android 3.1 and above. The startup parameters of Android application are the product-id and vendor-id defined in the device_filter.xml file. The Android application is divided into two parts, as shown below:



2. Android USB To Uart Demo

2.1 UART

The operation of the CH34x UART provides EnumerateDevice, OpenDevice, UartInit, SetConfig, WriteData and ReadData methods, and the WriteTimeOutMillis and ReadTimeOutMillis properties to communicate with the CH34x UART modules. While providing CloseDevice interface to close the UART Device, isConnected interface to determine whether the device is connected.

Programming should pay attention to the following points:

- Create CH34xUARTDriver object under the Application class to ensure that the application has multiple Activity can transmit and receive data when switch Activity.
- Operation process: ResumeUsbList (or OpenDevice after EnumerateDevice), UartInit, SetConfig, then data could be transmitted or received after the implementation of process above.

Please refer to the implementation of the Demo provided by our company.

2.2 UART User-Layout

EnumerateDevice: Enum the CH34x devices
Prototype: public UsbDevice EnumerateDevice()
Returns the device of the CH34x enumerated, or null if there is no device

OpenDevice: Open CH34x device
Prototype: public void OpenDevice(UsbDevice mDevice)
mDevice: the CH34x device to open

ResumeUsbList: Enum and open the CH34x device, this function contains EnumerateDevice, OpenDevice operation
Prototype: public int ResumeUsbList()
Return 0 if succeed, otherwise fail.

UartInit： CH34x device initialization

Prototype: public boolean UartInit()

Return false if the initialization fails, otherwise return true.

SetConfig： Set the baud rate, data bits, stop bits, parity bits, and flow control of the UART

Prototype: public boolean SetConfig(int baudRate, byte dataBit, byte stopBit, byte parity, byte flowControl)

baudRate: 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200, 230400, 460800, 921600. default：9600

dataBits: 5, 6, 7, or 8 data bits. Default: 8 data bits.

stopBits: 0 -> 1 stop bit, 1 -> 2 stop bits. default: 1.

Parity: 0 -> none, 1 -> odd, 2 -> even, 3 -> mark, 4 -> space. default： none

flowControl: 0 -> none, 1 -> CTS/RTS. default: none

Return false if setting fails, otherwise return true.

WriteData: Send data

Prototype: public int WriteData(byte[] buf, int length)

buf: send buffer

length: send length

Return the number of bytes written successfully

ReadData： Read data

Prototype: public int ReadData(char[] data, int length)

data ： receive buffer

length ： read length

Return the number of bytes read successfully

Prototype: public int ReadData(byte[] data, int length)

data ： receive buffer

length ： read length

Return the number of bytes read successfully

CloseDevice： Close device

Prototype: public void CloseDevice()

isConnected： Determine if the device is connected to the Android device

Prototype: public boolean isConnected()

Return false if the device is not connected to the system, and true indicates that the device is connected

In addition to the interface API provided above, users can also set the read and write timeout based on their own devices:

Prototype: public boolean SetTimeOut(int WriteTimeOut, int ReadTimeOut)

WriteTimeOut： Set write timeout, default 10000ms

ReadTimeOut ： Set read timeout, default 10000ms

3. Software operating instructions

Install the Demo provided by our company on the Android device with OTG interface (CH34xUARTDemo.apk). If the software is installed for the first time, after the CH34x UART function module is inserted, the system will automatically pop up the permission request dialog. Click "Use by default for this USB device". After selecting the operation, the dialog will not pop up again unless the Demo is re-installed; if you do not select "Use by default for this USB device" and directly determine the operation, the software will pop up no permission dialog and request to exit.

When the Demo is using, the process of opening the device will perform ResumeUsbList to complete the USB device enumeration, open the device, access to device resource information and other steps (or use OpenDevice to open device after EnumerateDevice), then the user needs to set the baud rate, data bits , stop bits, parity and flow control and other parameters, click the “Config” button, the UART configuration will be completed, then you can perform read and write operations.