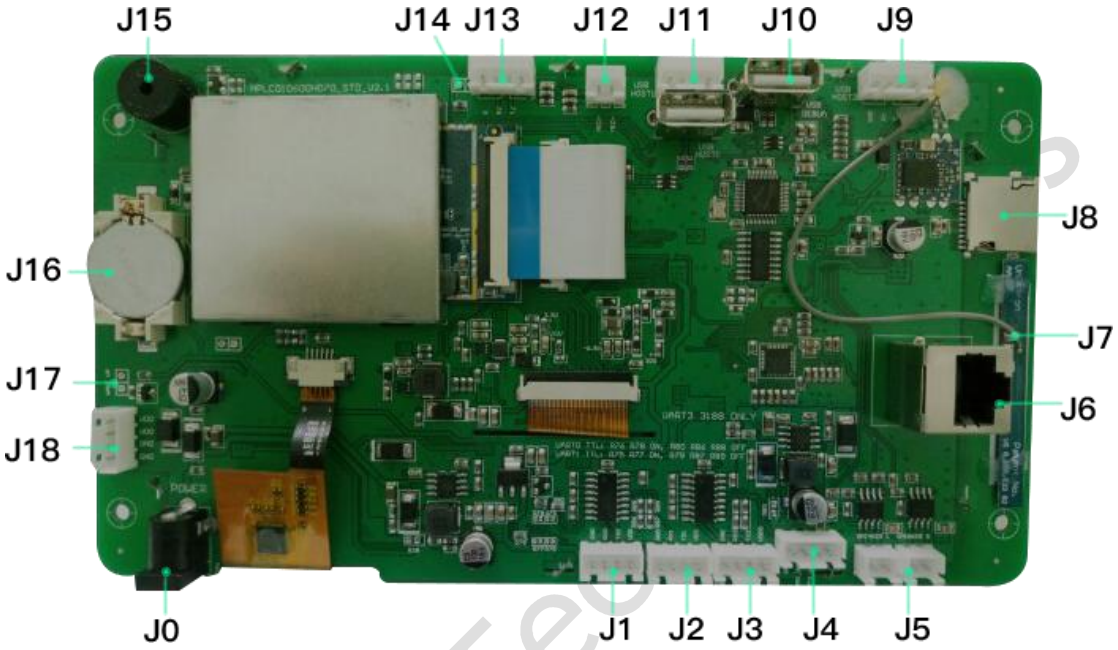
Android LCM Development Guide

Index

1. Interface Introduction

1.1 Serial Port Introduction

Take 7'' Android LCM with RK3188 chip for example. It has 3-level serial ports. TtyS0, ttyS1,ttyS2 are normal serial ports. They can be used for serial communication and support baud rate up to 115200. Besides, they support extension by USB-to-serial.



Standard Interface Description of 7'' Android LCM

No.	Interface	Description
J0	Power Input	12V3A Power Input
J1	Serial Port 0	Device name: ttyS0. Support RS232 and TTL/CMOS. Pin definition: GND, RX, TX, VCC.
J2	Serial Port 1	Device name: ttyS1. Support RS232. Pin definition: GND, RX, TX, VCC.
J3	Serial Port 3	Device name: ttyS3. Support RS232. Pin definition: GND, RX, TX, VCC (Not available for RK3128).
J4	USB to Serial Port	Device name: ttyCOM0. Support RS232. Pin definition: GND, RX, TX, VCC.
J5	Loudspeaker Port	4W Loudspeaker Output
J6	RJ45 Network Port	Support 10M/100M Network
J7	Wireless Port	Support IEEE802.11b/g/n. Wireless & Bluetooth 2in1 available.
J8	TF Card	
J9	USB 2	Support USB peripheral devices.
J10	USB DEBUG	USB port for App debugging and firmware upgrading.
J11	USB1	Support USB peripheral devices.
J12	MIC Port	Audio input port

J13	Serial Port for Debugging	Reserved for Debugging
J14	RECOVERY	Short-circuit to enter firmware burning mode
J15	Buzzer	
J16	RTC Port	Provide system RTC
J17	Sleep Wake-up Port	
J18	Power Input	Same as J0

1.2 USB Port Introduction

Also take the example of 7'' Android LCM with RK3188 chip. It has both USB HOST and USB DEVICE. USB HOST supports USB mouse, USB flash drive, etc. USB DEVICE is used for android applications development by android engineers.

2. Software Development

2.1 Java Environment Construction and Configuration

2.1.1 Tools needed to install java on Windows:

JDK: JDK is short for Java Development Kit. It includes java running environment, tools set, basic libraries, etc.

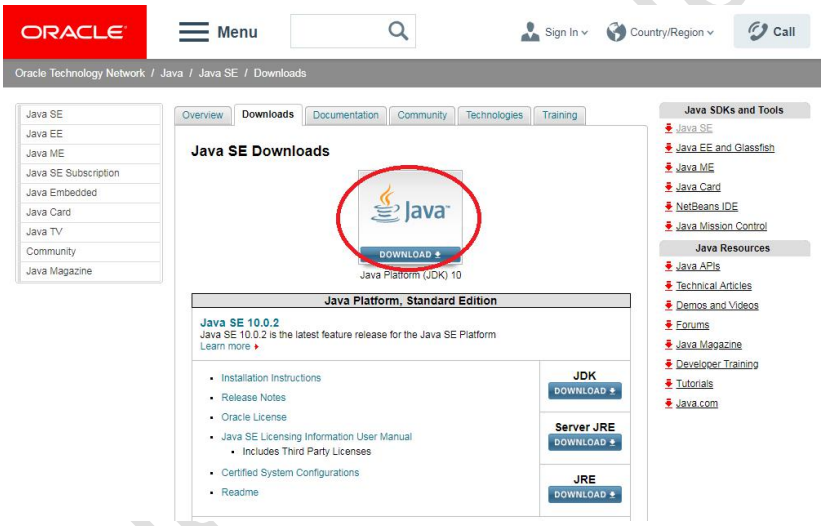
Android JDK: Android JDK is Android Java Development Kit provided by Google. During the development, it needs to be drawn in to use relevant API.

Android Studio: Google official IDE tool produced in 2003. As android studio no longer exists as a widget, it is much more convenient and stronger than Eclipse.

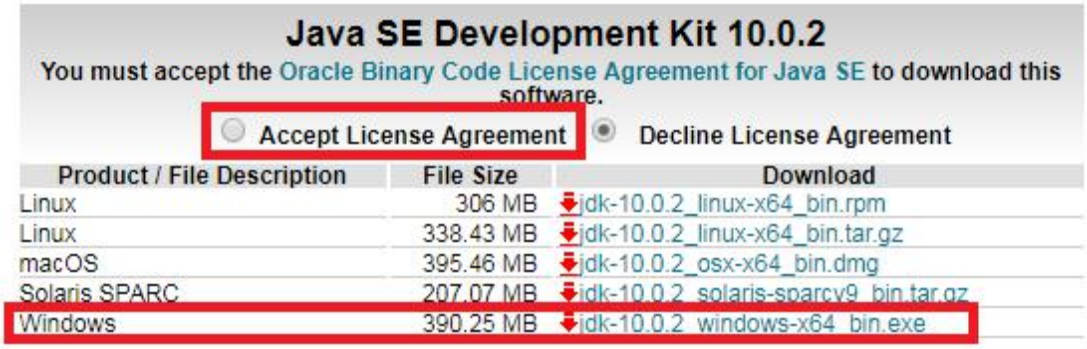
2.1.2 Download JDK

http://www.oracle.com/technetwork/java/javase/downloads/index.html

Click Download button as below:



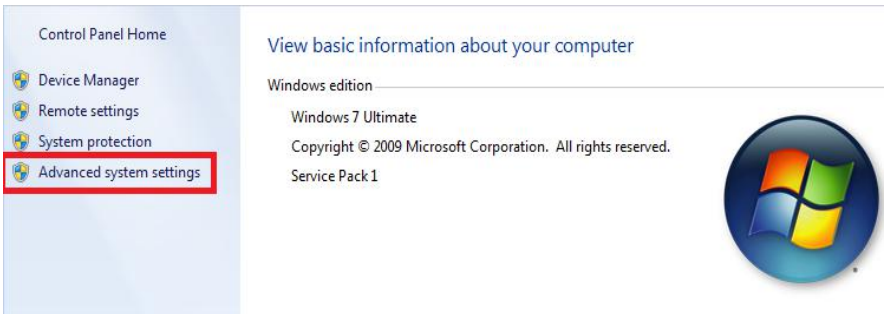
You need to accept license agreement and choose the version that fits your OS. Take Windows(x64) for example.



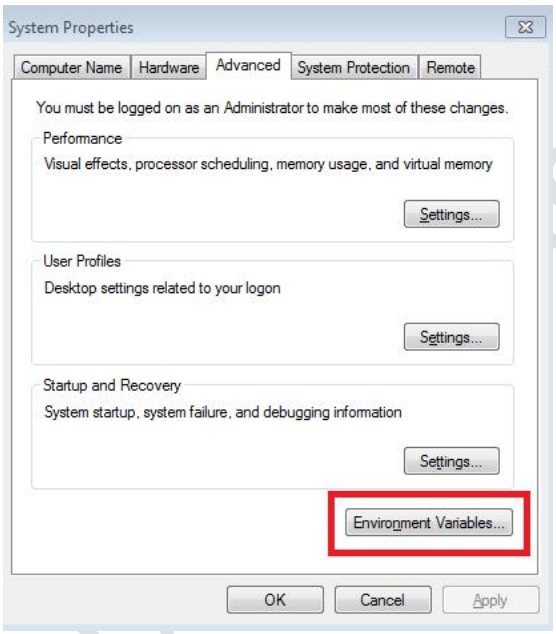
Install JDK after downloading. JRE can be installed together with JDK. Installation information can be customized such as the install path and etc..

2.1.3 Deploy Environment Variables

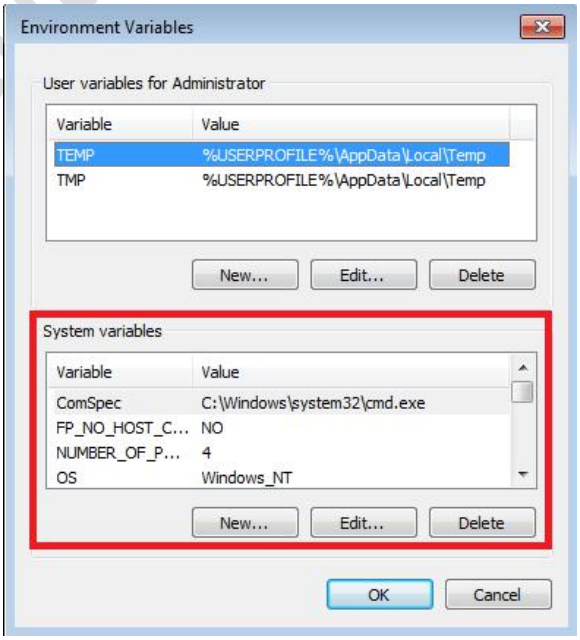
1. After installation of JDK, right click on “My Computer”, Click “Properties” and choose.



2. Click “Advanced” tab and click on “Environment Variables”



3. Configure 3 values in “System variables”: “JAVA_HOME”, “PATH” and “CLASSPATH”. Click “Edit” if they exist. Otherwise click “New”.



Parameters of Variables configuration as below:

Variable Name	Value
JAVA_HOME	C:\Program Files (x86)\Java\jdk1.8.0_91 //Configure to your own path
PATH	%JAVA_HOME%\bin;%JAVA_HOME%\jre\bin;
CLASSPATH	.;%JAVA_HOME%\lib\dt.jar;%JAVA_HOME%\lib\tools.jar; //

Note: PATH needs to be re-configured as %JAVA_HOME%\bin;, otherwise operations such as command lines and Eclipse will fail.

After all above java environment configuration, you can run Eclipse to write codes. Eclipse will finish the configuration automatically.

2.1.4 Test if JDK is successfully installed

1. "Start" -> "Run" -> Input "cmd";
2. Input commands: "java-version", "java" and "javac". Environment variable configuration is successful if below information shows:

```
C:\Users\prado>java -version
java version "1.8.0_91"
Java(TM) SE Runtime Environment (build 1.8.0_91-b14)
Java HotSpot(TM) 64-Bit Server VM (build 25.91-b14, mixed mode)
```

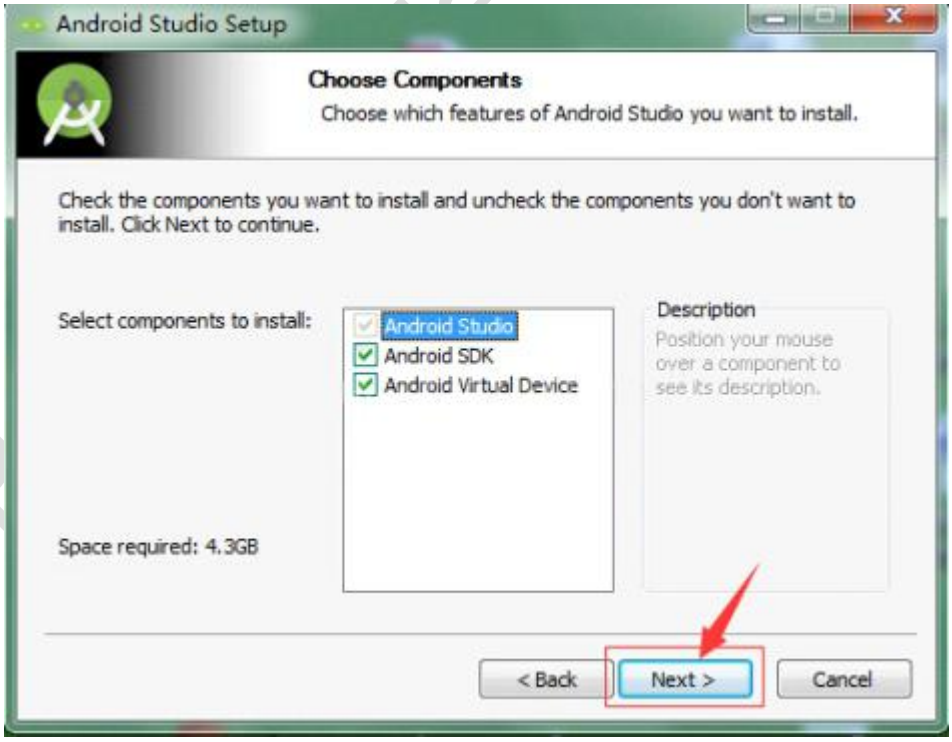
2.2 Installation of Android Studio and Environment Building

2.2.1 Download and Install Android Studio

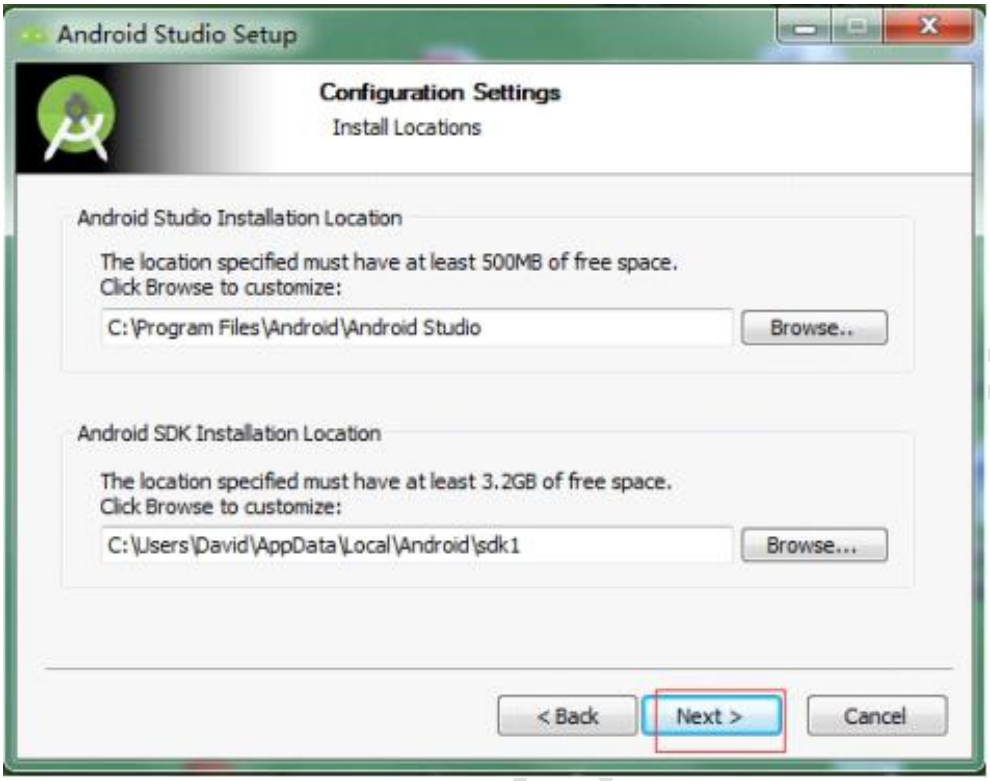
Open link: http://www.android-studio.org/ and click Download Android Studio.

1. Open the installation package and start installation process:

2. Select “Android Virtual Device” :

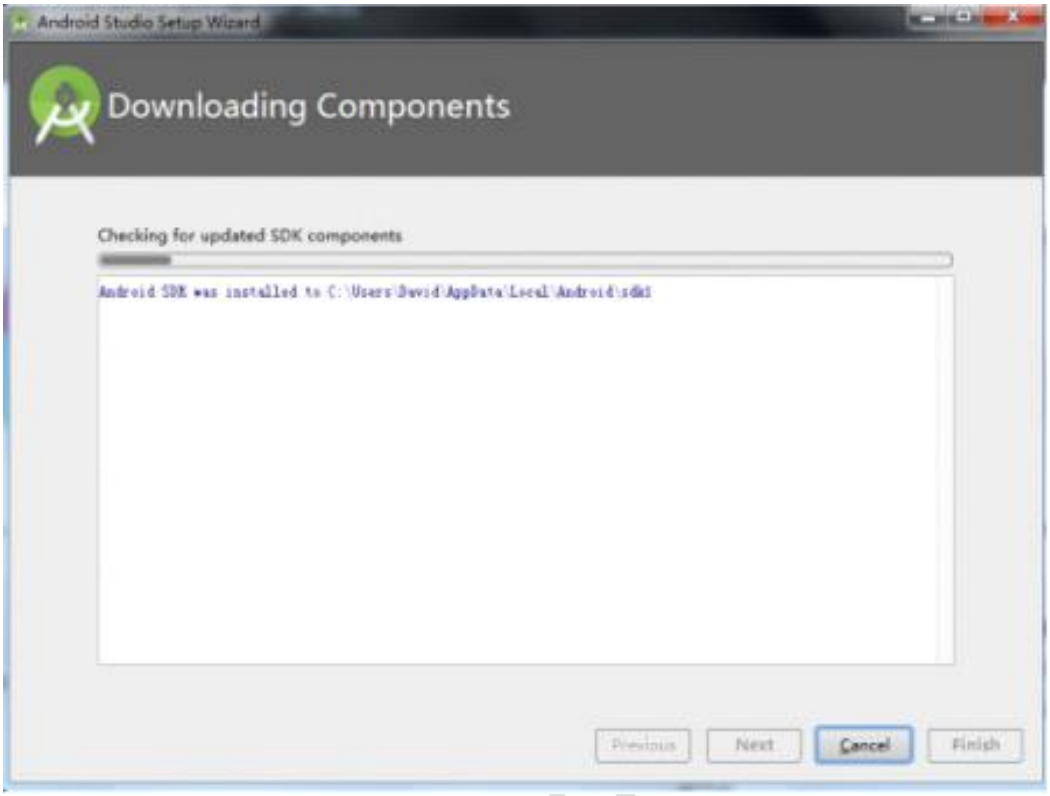


3. Choose “Android Studio Installation Location” and “Android SDK Installation Location” and continue:

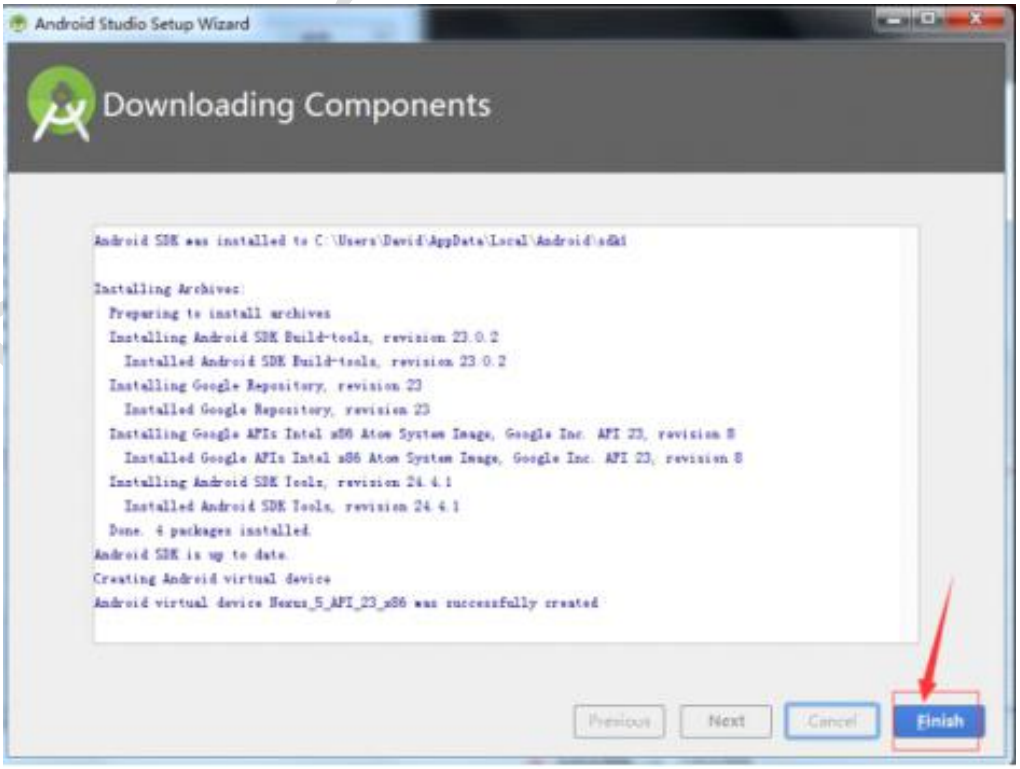


4. Finish the installation:

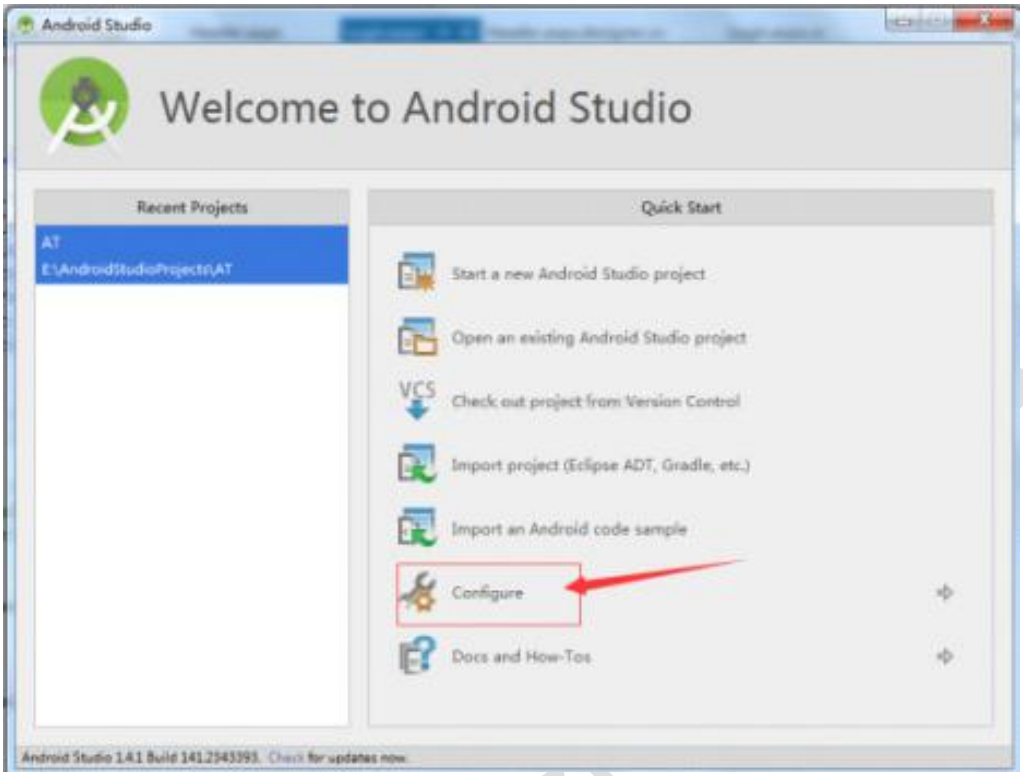
5. Make sure your computer can get access to Google and download necessary files:



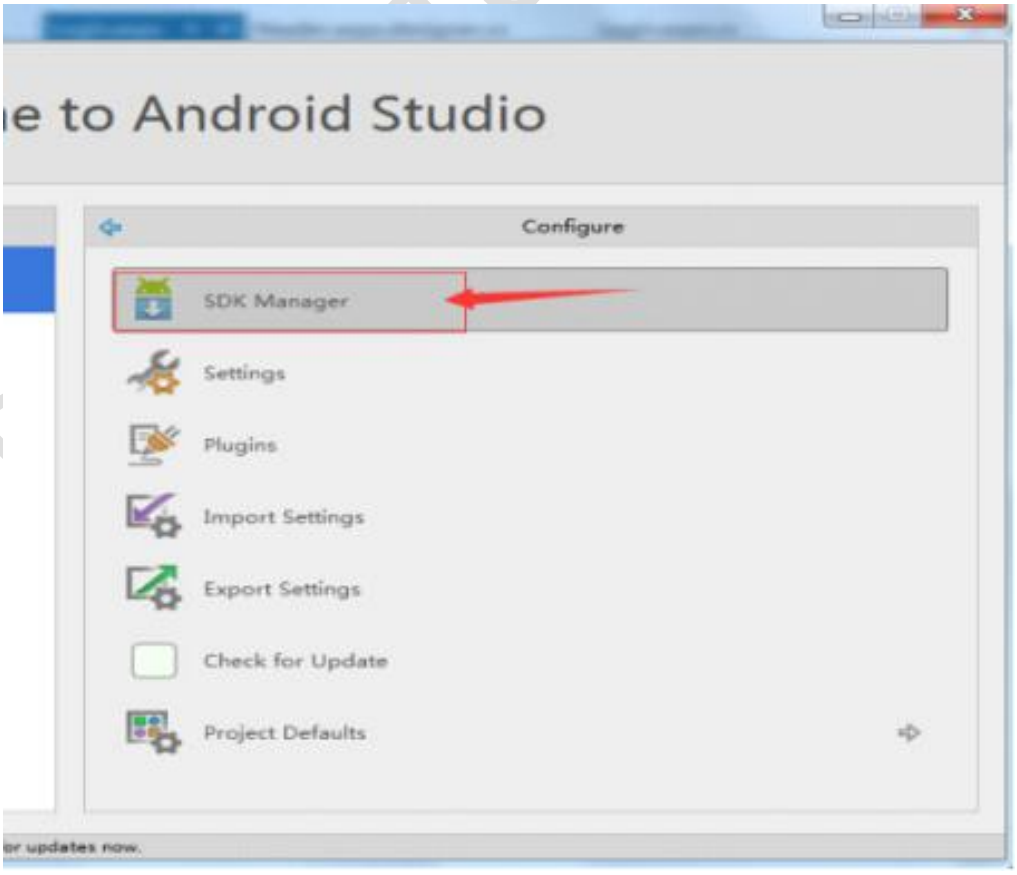
6. Click “Finish”



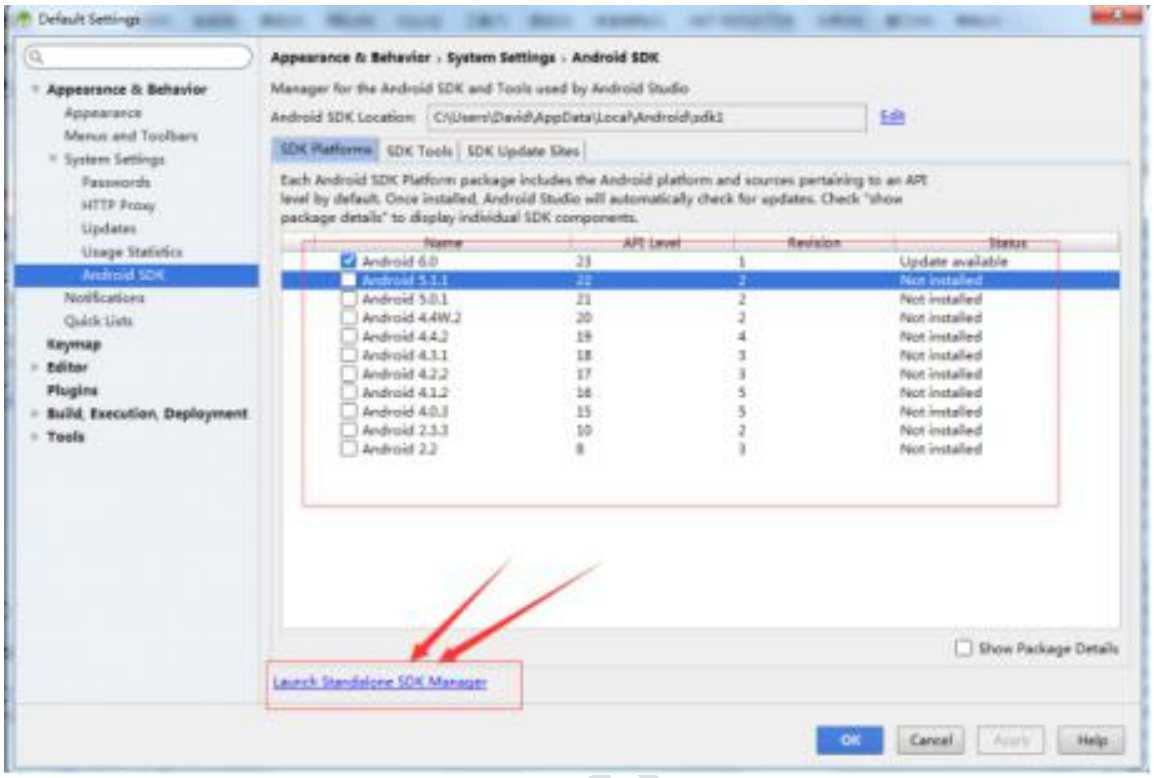
7. If you still need to install other version of SDK, open Android Studio and Click “Configure”:



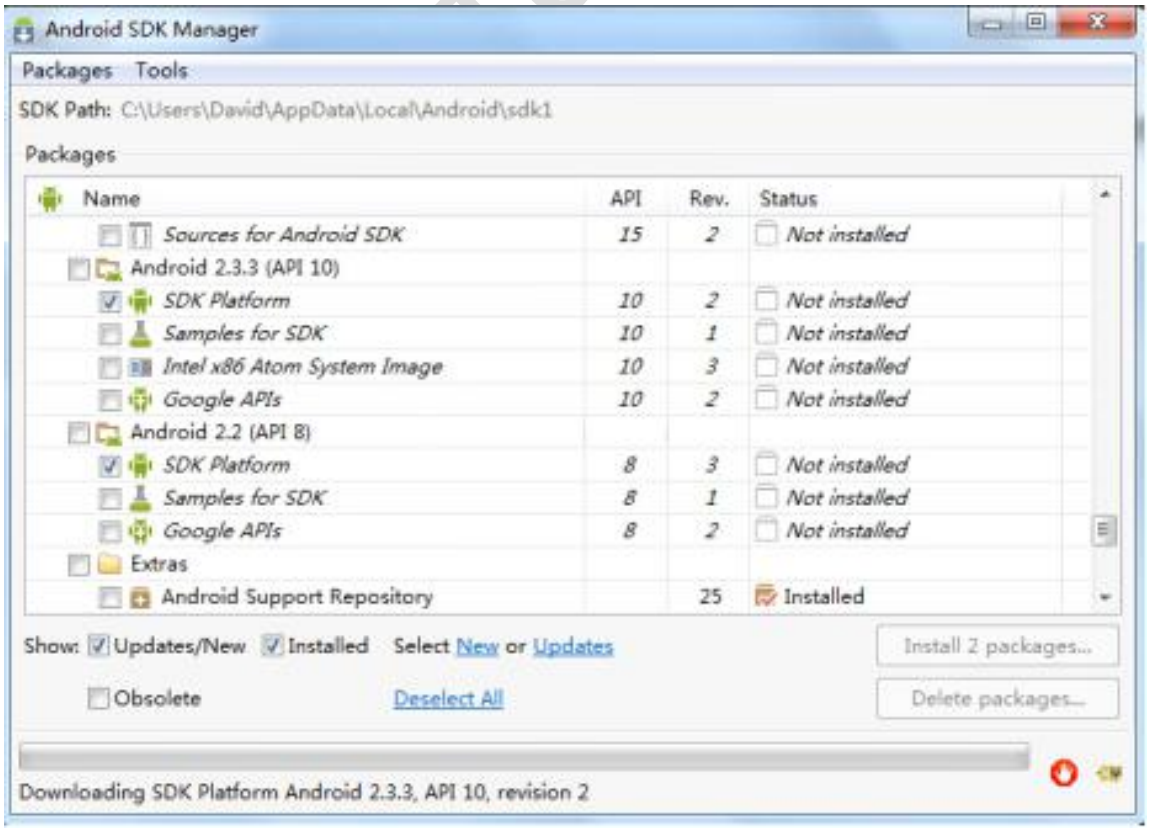
8. Click “SDK Manager”:



9. Click “Launch Standalone SDK Manager” in “Default Settings”:



10. Choose the version you need and download:

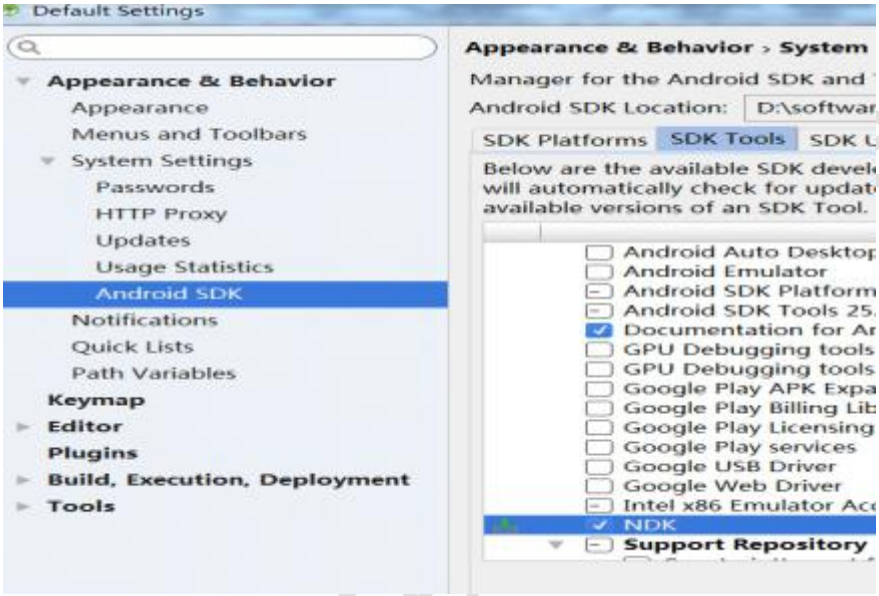


2.2.2 Configure NDK Development Environment in Android Studio

It is more complicated to realize NDK development in Android Studio than in Eclipse as there is no way to add native support with one button in Android Studio. Below shows how to realize such functions in Android Studio.

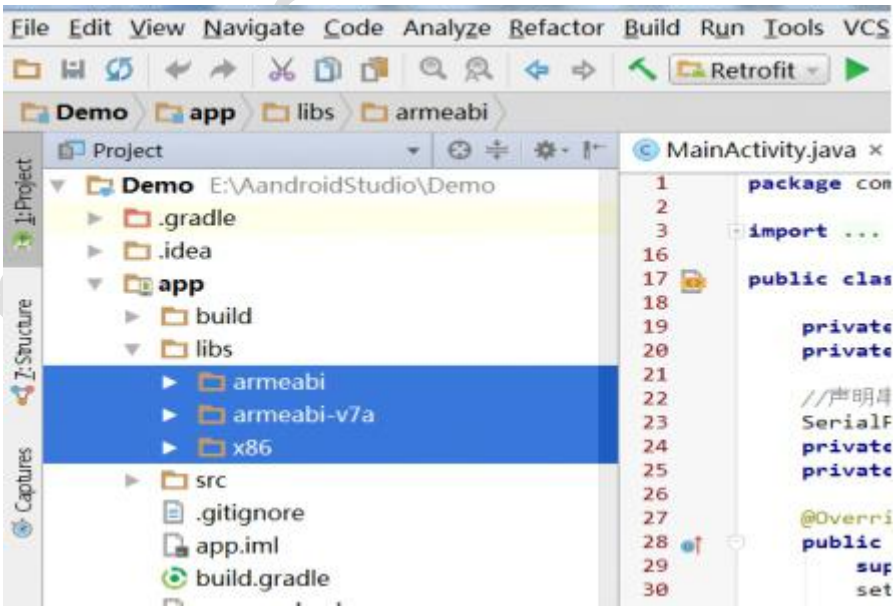
Step 1: Install NDK

Open “Tools” -> “Android” -> “SDK Manager” -> “SDK Tools”, select LLDB and NDK, click OK to start the installation.



Step 2: Import So Library to “libs”

So Library Download link: http://www.proculustech.com/SoftwareResources/libs.rar



Step 3: Synchronize resource library

2.3 Serial Port Development & Demo Briefing

Serial Port Demo Download Link:
http://www.proculustech.com/SoftwareResources/Serial%20Port%20Demo.rar

2.3.1 Serial Communications

It is unavoidable to discuss serial port development without serial communications.
The concept of Serial Communications is quite simple. It is one kind of communications that happens between peripheral and PC via signal wire, earth wire or control wire. The serial port sends and receives bytes by bit. Although it is slower than communicate by byte, serial port can send data with one wire while receive data with another. It is simple and can realize long-distance communications.
The most important parameters of serial communications are baud rate, data bit, stop bit and parity check. For the terminals to communicate, these parameters must match.

2.3.2 Serial Communication between Android LCM and Android Applications

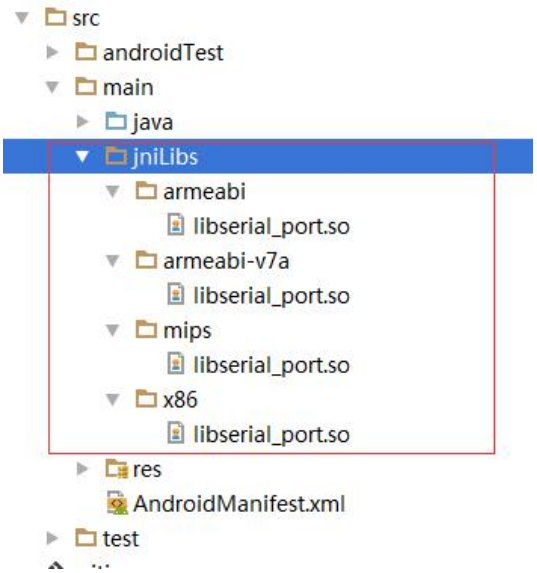
Below are the steps to realize the communication:

1. Serial port initialization: Create serial port and set serial parameters (defined path and baud rate);
2. Access input stream: Read serial data;
3. Access output stream: Send data to serial port;
4. Data processing and display;
5. Close serial port.

2.3.3 Instructions of codes in serial demo

Step 1: Import "So" Library

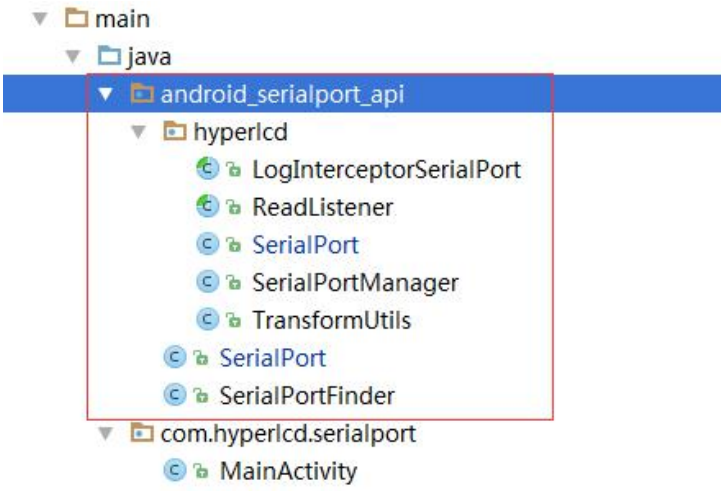
1. Copy "so" library to folder "jinLibs"



2. Add configuration in “android” tag in “build.gradle”

```
sourceSets {
main {
jni.srcDirs = []
jniLibs.srcDirs = ['src/main/jniLibs', 'libs']
    }
}
```

Step 2: Copy all the files under “android_sericalport_api” to folder “java”. (Do not change the path)



Step 3: Use the serial port

1 Initialize SerialPortManager

```
SerialPortManager.getInstances().initSerialPort();
```

2 Open Serial Port

```
// Access the example of serial tool object(The example is used to open/close serial port and send data).
SerialPortserialPort = SerialPortManager.getInstances().getSerialPort();
```

// Initialize read listener

```
ReadListenerreadListener= new ReadListener() {
@Override
public void onRead(final String port, final booleanisAscii, final String read) {
Log.d("SerialPortRead", new StringBuffer()
.append("serial port number：").append(port)
.append("\n data format:").append(isAscii ? "ascii" :"hexString")
.append("\n read:").append(read).toString());
    }
};
```

// Open serial port with serial tool object example to transfer to read listener.

serialPort.startSerialPort(SerialPortManager.ttyCOM0, **false**, **readListener**);// open serial port /dev/ttyCOM0. Data format is HexString.

3 Modify read data format

serialPort.setReadCode(SerialPortManager.ttyCOM0, true);// Modify serial port /dev/ttyCOM0 Read data format is Ascii

4 Send data (Serial port should be open)

```
serialPort.writeSerialService(SerialPortManager.ttyCOM0, false, "data to be sent");// write data to
/dev/ttyCOM0. Data format is HexString. Data content: "data to be sent”
```

5 Close serial port

```
serialPort.stopSerialPort(SerialPortManager.ttyCOM0);// Close serial port /dev/ttyCOM0
```

6 Log block (configure log block to read the log)

```
// Configure the log block
SerialPortManager.getInstances().setLogInterceptor(new LogInterceptorSerialPort() {
@Override
public void log(@SerialPortManager.Typefinal String type, final String port, final booleanisAscii,
final String log) {
Log.d("SerialPortLog", new StringBuffer()
.append("serial port number:").append(port)
.append("\n data format:").append(isAscii ? "ascii" : "hexString")
.append("\n operation type:").append(type)
.append("operation report:").append(log).toString());
}
});
```

7 Destroy SerialPortManager. It requires re-initialize when using serial port after destroying.

```
SerialPortManager.getInstances().destroySerialPort();
```

2.3.4 Serial Demo Code

```
packagecom.hyperlcd.serialport;
importandroid.os.Bundle;
importandroid.support.annotation.IdRes;
import android.support.v7.app.AppCompatActivity;
importandroid.text.TextUtils;
importandroid.util.Log;
importandroid.view.Gravity;
importandroid.view.View;
importandroid.widget.CompoundButton;
importandroid.widget.EditText;
importandroid.widget.RadioButton;
importandroid.widget.RadioGroup;
```

```
importandroid.widget.TextView;
importandroid.widget.Toast;

importandroid_serialport_api.hyperlcd.LogInterceptorSerialPort;
importandroid_serialport_api.hyperlcd.ReadListener;
importandroid_serialport_api.hyperlcd.SerialPort;
importandroid_serialport_api.hyperlcd.SerialPortManager;

public class MainActivity extends AppCompatActivity implements View.OnClickListener {

privateRadioGroupcodeRG;
privateRadioGroupserialRG;
privateEditTextserialET;
privateEditTextsendET;
privateTextViewreadTV;
privateTextViewlogTV;
privateTextViewserialTitle;
privateTextViewcodeTitle;

private String currentPort;
privateSerialPortserialPort;
privateReadListenerreadListener;
privatebooleanisAscii;

    @Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);

initView();

initData();
    }

private void initData() {
SerialPortManager.getInstances().initSerialPort();
SerialPortManager.getInstances().setLogInterceptor(new LogInterceptorSerialPort() {
    @Override
public void log(@SerialPortManager.Type final String type, final String port, final booleanisAscii, final String log)
{
Log.d("SerialPortLog", new StringBuffer()
    .append("serial port number: ").append(port)
.append("\n data format:").append(isAscii ? "ascii" : "hexString")
.append("\n operation type:").append(type)
```

```
.append("operation report: ").append(log).toString());
runOnUiThread(new Runnable() {
@Override
public void run() {
logTV.append(new StringBuffer()
        .append(" ").append(port)
        .append(" ").append(isAscii ? "ascii" : "hexString")
.append(" ").append(type)
.append("：  ").append(log)
.append("\n").toString());
            }
         });
     }
});

serialPort = SerialPortManager.getInstances().getSerialPort();
readListener = new ReadListener() {
@Override
public void onRead(final String port, final booleanisAscii, final String read) {
Log.d("SerialPortRead", new StringBuffer()
.append(port).append("/").append(isAscii ? "ascii" : "hex")
.append(" read：  ").append(read).append("\n").toString());
runOnUiThread(new Runnable() {
@Override
public void run() {
readTV.append(new StringBuffer()
.append(port).append("/").append(isAscii ? "ascii" : "hex")
.append(" read：  ").append(read).append("\n").toString());
           }
       });
     }
   };
 }


private void initView() {
codeRG = (RadioGroup) findViewById(R.id.rg_code);
serialRG = (RadioGroup) findViewById(R.id.rg_serial);

serialET = (EditText) findViewById(R.id.et_serial);

sendET = (EditText) findViewById(R.id.et_send);
serialTitle = (TextView) findViewById(R.id.title_serial);
codeTitle = (TextView) findViewById(R.id.title_code);
```

```
readTV = (TextView) findViewById(R.id.tv_read);
logTV = (TextView) findViewById(R.id.tv_log);

codeRG.setOnCheckedChangeListener(new RadioGroup.OnCheckedChangeListener() {
    @Override
public void onCheckedChanged(RadioGroup group, @IdResintcheckedId) {
changeCode(checkedId == R.id.rb_ascii);
                }
            });

serialRG.setOnCheckedChangeListener(new RadioGroup.OnCheckedChangeListener() {
                @Override
public void onCheckedChanged(RadioGroup group, @IdResintcheckedId) {
if (checkedId == R.id.rb_other) {
serialET.requestFocus();
                    } else {
sendET.requestFocus();
                    }
                }
            });

serialET.setOnFocusChangeListener(new View.OnFocusChangeListener() {
                @Override
public void onFocusChange(View v, booleanhasFocus) {
if (hasFocus) {
serialRG.check(R.id.rb_other);
                    }
                }
            });

findViewById(R.id.btn_open).setOnClickListener(this);
findViewById(R.id.btn_close).setOnClickListener(this);
findViewById(R.id.clear_send).setOnClickListener(this);
findViewById(R.id.btn_send).setOnClickListener(this);
findViewById(R.id.clear_read).setOnClickListener(this);
findViewById(R.id.clear_log).setOnClickListener(this);

codeRG.check(R.id.rb_ascii);
serialRG.check(R.id.rb_com0);
sendET.requestFocus();
    }

    @Override
protected void onDestroy() {
```

```
SerialPortManager.getInstances().destroySerialPort();
super.onDestroy();
    }

    @Override
public void onClick(View v) {
switch (v.getId()) {
caseR.id.btn_open:
open();
break;
caseR.id.btn_close:
close();
break;
caseR.id.btn_send:
send();
break;
caseR.id.clear_send:
sendET.setText("");
break;
caseR.id.clear_read:
readTV.setText("");
break;
caseR.id.clear_log:
logTV.setText("");
break;
default:
        }
    }

private void send() {
if (serialPort == null) {
            // Serial Port Uninitialized
            T("Serial Port Uninitialized");
return;
        }

if (TextUtils.isEmpty(currentPort)) {
            // Serial Port Not Open
            T("Serial Port Not Open");
return;
        }

        String send = sendET.getText().toString().trim();
if (TextUtils.isEmpty(send)) {
```

```
                //Data Empty
                T("Data Empty");
return;
            }
            // Send Data
try {
serialPort.writeSerialService(currentPort, isAscii, send);
            } catch (Exception e) {
e.printStackTrace();
            }
        }

        /**
         * Open Serial Port
         */
private void open() {
if (serialPort == null) {
return;
            }
            String checkPort = getCurrentPort();
if (TextUtils.isEmpty(checkPort)) {
return;
            } else if (TextUtils.equals(checkPort, SerialPortManager.other)) {
checkPort = serialET.getText().toString().trim();
if (TextUtils.isEmpty(checkPort)) {
                    T("Enter Serial Port Num.");
return;
                }
            }

if (TextUtils.equals(currentPort, checkPort)) {
return;
            }

if (!TextUtils.isEmpty(currentPort)) {
                // Close currentPort
serialPort.stopSerialPort(currentPort);
            }

isAscii = codeRG.getCheckedRadioButtonId() == R.id.rb_ascii;
currentPort = checkPort;
            // Open checkPort
serialPort.startSerialPort(checkPort, isAscii, readListener);
```

```
serialTitle.setText("Serial Port:");
serialTitle.append(currentPort);
codeTitle.setText("Data Format:");
codeTitle.append(isAscii ? "ASCII" : "HexString");
    }

    /**
     * Close Serial Port
     */
private void close() {
if (!TextUtils.isEmpty(currentPort)) {
                // Close currentPort
serialPort.stopSerialPort(currentPort);
currentPort = "";
serialTitle.setText("Serial Port");
codeTitle.setText("Data Format");
        }
    }

    /**
     * Modify Data Format
     *
     * @paramisAsciitrue:asciifalse:HexString
     */
private void changeCode(booleanisAscii) {
if (TextUtils.isEmpty(currentPort)) {
return;
        }
serialPort.setReadCode(currentPort, isAscii);
codeTitle.setText("Data Format:");
codeTitle.append(isAscii ? "ASCII" : "HexString");
    }

    /**
     * Access chosen serial port
     *
     * @return
     */
private String getCurrentPort() {
        String checkPort;
switch (serialRG.getCheckedRadioButtonId()) {
case R.id.rb_com0:
checkPort = SerialPortManager.ttyCOM0;
break;
```

```
case R.id.rb_com1:
checkPort = SerialPortManager.ttyCOM1;
break;
case R.id.rb_com2:
checkPort = SerialPortManager.ttyCOM2;
break;
case R.id.rb_com3:
checkPort = SerialPortManager.ttyCOM3;
break;
case R.id.rb_s0:
checkPort = SerialPortManager.ttyS0;
break;
case R.id.rb_s1:
checkPort = SerialPortManager.ttyS1;
break;
case R.id.rb_s2:
checkPort = SerialPortManager.ttyS2;
break;
case R.id.rb_s3:
checkPort = SerialPortManager.ttyS3;
break;
caseR.id.rb_other:
checkPort = SerialPortManager.other;
break;
default:
checkPort = "";
            }
returncheckPort;
      }

private Toast toast;
privateTextViewtextView;

private void T(String message) {
if (toast == null) {
toast = Toast.makeText(this, "", Toast.LENGTH_SHORT);
textView = new TextView(this);
textView.setTextColor(0xffffffff);
textView.setTextSize(30);
textView.setPadding(10, 5, 10, 5);
textView.setBackgroundResource(R.drawable.shape_toast_bg);
toast.setView(textView);
toast.setGravity(Gravity.CENTER, 0, 0);
            }
```

```
textView.setText(message);
toast.show();
    }
}
```

2.4 Introduction of Self-starting

There are two methods of realizing self-starting. One is to self-start when receiving booting broadcasting. It starts APK after entering system desk. The other is to set its own APK as Launcher. It will skip system desk and start APK directly.

2.4.1 Self-start after receiving booting broadcasting

When Android start-up, it will send out a system broadcast "ACTION_BOOT_COMPLETED". Usually, its string constant is "android.intent.action.BOOT_COMPLETED". The program just needs to capture this message and self-start. So the way of realizing this function is to make a BroadcastReceiver.

The most important part of self-starting is to give the application the authority of "launcher".

Step 1: Customize Broadcast Class "BootReceiver"

```
public class BootReceiver extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent) {
        if(intent.getAction().equals("android.intent.action.BOOT_COMPLETED")) {        // boot
            Intent intent2 = new Intent(context, MainActivity.class);
//              intent2.setAction("android.intent.action.MAIN");
//              intent2.addCategory("android.intent.category.LAUNCHER");
            intent2.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
            context.startActivity(intent2);
        }
    }
}
```

Step 2: Manifest Files Configuration

Add customized broadcast class in "Application" of "AndroidManifest.xml":

```
<receiver android:name="BootReceiver" >
    <intent-filter>
        <action android:name="android.intent.action.BOOT_COMPLETED" />
            <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</receiver>
```

Step 3: Add auto-start authority in "Manifest" of "AndroidManifest.xml"

```
<uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED" />
```

2.4.2 Set its own APK as Launcher

Replace <intent-file> of the first activity in configuration file.

```
<activity
    android:name="com.imstlife.mydoor.MainActivity"
    android:configChanges="screenSize"
    android:label="@string/app_name"
    android:launchMode="singleTop">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />

            <category android:name="android.intent.category.LAUNCHER" />
            <category android:name="android.intent.category.HOME" />
            <category android:name="android.intent.category.DEFAULT" />
            <category android:name="android.intent.category.MONKEY" />

        </intent-filter>
</activity>
```

Start program and click “Home”. Select the APK in popup select box, choose “Always” and confirm.

2.5 Change System Signature

Step 1:

Add android:sharedUserId="android.uid.system" in configuration file “Manifest” of “AndroidManifest.xml”. Compile source code with Eclipse to generate APK file (Please refer to the code in the demo)

Step 2:

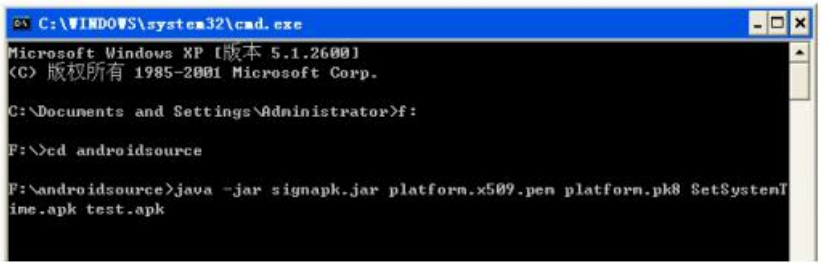
Change the apk file’s format into *.zip. Open it and delete “CERT.SF” and “CERT.RSA” in “META-INF”.

Step 3:

Change the file back into *.apk. Put the file under the path F:/androidsource (Can be modified). Make sure “platform.pk8”, “platform.x509.pem”, “signapk.jar” are in the same folder.

Step 4:

Run “cmd.exe” and input “cd androidsource” and run command: java -jar signapk.jar platform.x509.pem platform.pk8 SetSystemTime.apk test.apk. As below image:



2.6 APK Encryption

2.6.1 Obfuscated Code

1. Global obfuscation options. Modify build/core/package.mk

```
ifndef LOCAL_PROGUARD_ENABLED
ifneq ($(filter user userdebug, $(TARGET_BUILD_VARIANT)),)
    # turn on Proguard by default for user & userdebug build
    LOCAL_PROGUARD_ENABLED :=full
Endif
```

2. Global flag files modification. Shield –dontobfuscate and modify build/core/proguard.flags

```
# Don't obfuscate. We only need dead code striping.
#-dontobfuscate
```

3. Create proguard.cfg under its own module to configure obfuscation options.
4. Add below commands under LOCAL_MODULE -> LOCAL_PACKAGE_NAME in Android.mk

```
LOCAL_PROGUARD_ENABLED := full #Designate current application to open obfuscation
LOCAL_PROGUARD_FLAG_FILES := proguard.cfg #Designate obfuscation configuration file
```

5. User /setenv.sh -bv user to set environment variable when compiling
6. Modify proguard.cfg when reporting error.

2.6.2 Avoid Repeated Packing

1. Add signature verification in Java code (modify smali file directly)
2. Add signature verification in NDK (check with IDA and modify in HEX)
3. Take advantage of its own defect to prevent packing (Manifest cheat, change suffix of image file, etc.)

2.6.3 Use Third-party Tools to Encrypt DEX, RES and SO Library

Step 1: Export APK

Step 2: Reinforce with third-party platform (generate multi-channel package)

Step 3: Re-sign APK

There is no signature in the reinforced APK. And it cannot be installed to LCM without a signature. Using built-in signature tool to resign APK is very complicated. Therefore we suggest using third-party signature tool.

2.7 Usage of Buzzer (Only available for certain series of LCMs)

2.7.1 Principle

Buzzer is integrated electronic equipment used for sounding device in electronic products.

2.7.2 Procedure

Buzzer is controlled by MCU. MCU and Main chip communicate with serial port and Android application open serial port to send [BEEP] to realize buzzer.

2.7.3 Android Code

Step 1: Create Class RootShellCmd

```
public class RootShellCmd {
    private OutputStream os;
    /**
     * Execute shell command
     * @param cmd
     */
    public final void exec(String cmd) {
        try {
            if (os == null) {
                os = Runtime.getRuntime().exec("su").getOutputStream();
            }
            os.write(cmd.getBytes());
            os.flush();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
    /**
     * Open buzzer
     */
    public void openBuzzer() {
        exec("echo 168 > /sys/class/gpio/export\n");
        exec("echo"+" out"+" > /sys/class/gpio/gpio168/direction\n");
        exec("echo out > /sys/class/gpio/gpio168/direction\n");
        exec("echo 1 > /sys/class/gpio/gpio168/value\n");
    }
    /**
     * Close buzzer
     */
    public void closeBuzzer() {
        exec("echo 0 > /sys/class/gpio/gpio168/value\n");
    }
}
```

Step 2: Call the method of controlling buzzer in MainActivity

```
RootShellCmd rootShellCmd = new RootShellCmd();
rootShellCmd.openBuzzer();//
rootShellCmd.closeBuzzer();//
```

2.8 Show & Hide Navigation Bar

Create Class “RootShellCmd.Android” in Android application to send adb command to LCM to realize the function.

Step 1: Create Class “RootShellCmd”

```
public class RootShellCmd {
    private OutputStream os;
    /**
     * Execute shell command
     * @param cmd
     */
    public final void exec(String cmd) {
        try {
            if (os == null) {
                os = Runtime.getRuntime().exec("su").getOutputStream();
            }
            os.write(cmd.getBytes());
            os.flush();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
    /**
     * Backstage simulation global key
     * @param keyCode
     */
    public final void simulateKey(int keyCode) {
        exec("input keyevent " + keyCode + "\n");
    }
    /**
     * Hide Navigation Bar
     */
    public static void closeBar() {
        try {
            // REQUIRES ROOT
            Build.VERSION_CODES vc = new Build.VERSION_CODES();
            Build.VERSION vr = new Build.VERSION();
            String ProcID = "79"; // HONEYCOMB AND OLDER
            // v.RELEASE //4.0.3
            if (vr.SDK_INT >= vc.ICE_CREAM_SANDWICH) {
                ProcID = "42"; // ICS AND NEWER
            }
            Process proc = Runtime.getRuntime().exec(
```

```
                        new String[] {"su","-c","service call activity " +
    ProcID+ " s16 com.android.systemui" }); // WAS 79
                proc.waitFor();

            } catch (Exception ex) {
                ex.printStackTrace();
            }
        }
        /**
         * Show Navigation Bar
         */
        public static void showBar(){
            try {
                // REQUIRES ROOT
                Build.VERSION_CODES vc = new Build.VERSION_CODES();
                Build.VERSION vr = new Build.VERSION();
                String ProcID = "79"; // HONEYCOMB AND OLDER
                // v.RELEASE //4.0.3
                if (vr.SDK_INT >= vc.ICE_CREAM_SANDWICH) {
                    ProcID = "42"; // ICS AND NEWER
                }
                //Execute command
                Process proc = Runtime.getRuntime().exec(
                        new String[] {"su","-c","am startservice -n
    com.android.systemui/.SystemUIService " }); // WAS 79
                proc.waitFor();
            } catch (Exception ex) {
                ex.printStackTrace();
            }
        }
}
```

Step 2: Hide/Show Navigation Bar in MainActivity

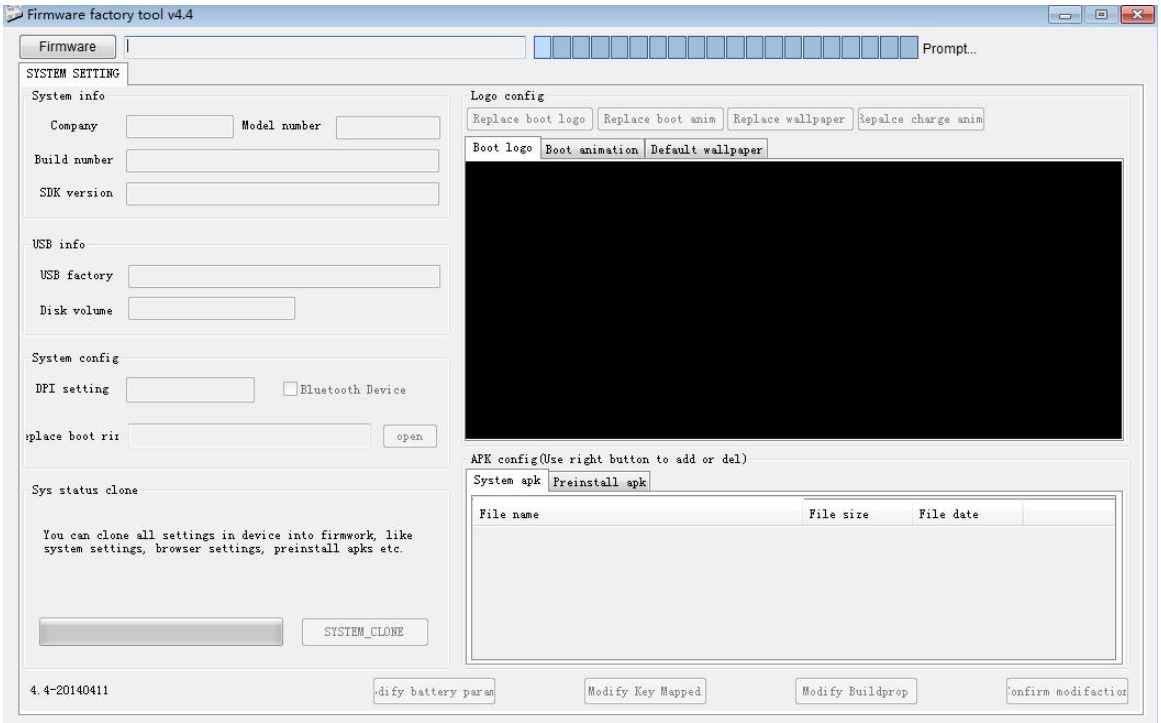
```
RootShellCmd.closeBar();//Hide Navigation Bar
//RootShellCmd.showBar();//Show Navigation Bar
```

2.9 Change Boot Animation & Boot Image

Download firmware factory tool
Link: http://www.proculustech.com/SoftwareResources/Firmware%20Factory%20Tool.zip

2.9.1 Image Configuration

"Firmware Factory Tool" can be used to check & replace kernel picture, Android boot animation, default wallpaper & charging animation.



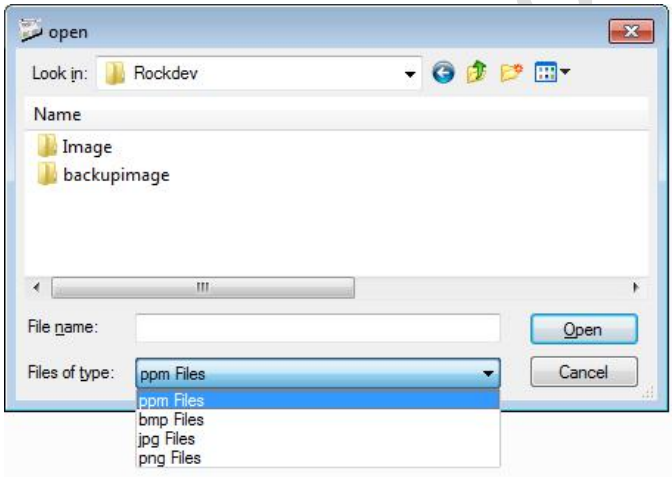
2.9.2 Modify Boot Logo

2.9.2.1 Generation of Kernel Logo

Use "FirmwareToPPM.exe" in the folder "Firmware Factory Tool\bin" to transform 224ppm picture (support *.ppm, *.bmp, *.jpg, *.png). Put the transformed picture into kernel to compile.

2.9.2.2 Modify Boot Logo

Open “Firmware Factory Tool” and click “Replace boot logo”. Choose the boot logo that needs to be changed. The tool will filter compatible pictures according to the logo file in your firmware (support *.ppm, *.bmp, *.jpg, *.png).



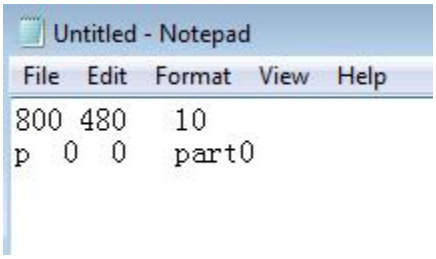
Note: The firmware with bmp logo file configuration supports multi-resolution. The maximum resolution and displayed logo resolution are shown below:

2.9.3 Modify Boot Animation

2.9.3.1 Create Boot Animation

The file of boot animation is bootanimation.zip. It contains two parts: Part0 and desc.txt.

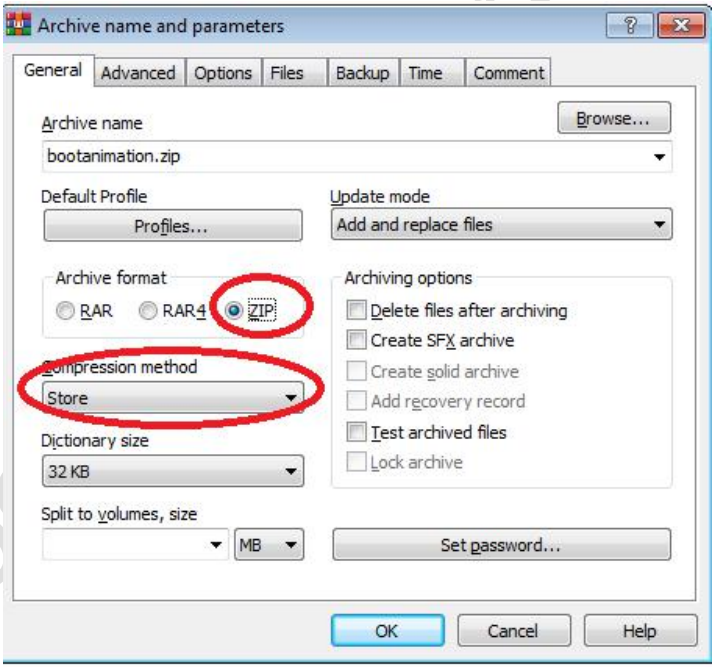
1. Part0: Pictures of boot animation is stored in this folder. The resolution of the picture must be the same as the resolution of the device.
2. Desc.tex: This file is generated in Linux. The parameters are shown below (give an example with resolution 800*480):



Descriptions of parameters in file “desc”

Parameter	Description
800 480 10	800 480 – Screen Resolution 10 -- No. of pictures played per second
P 0 0 part0	P – Play 0 – 0: Loop Play; 1: Single Play 0 – Delay time Part0 – Folder to store animation files

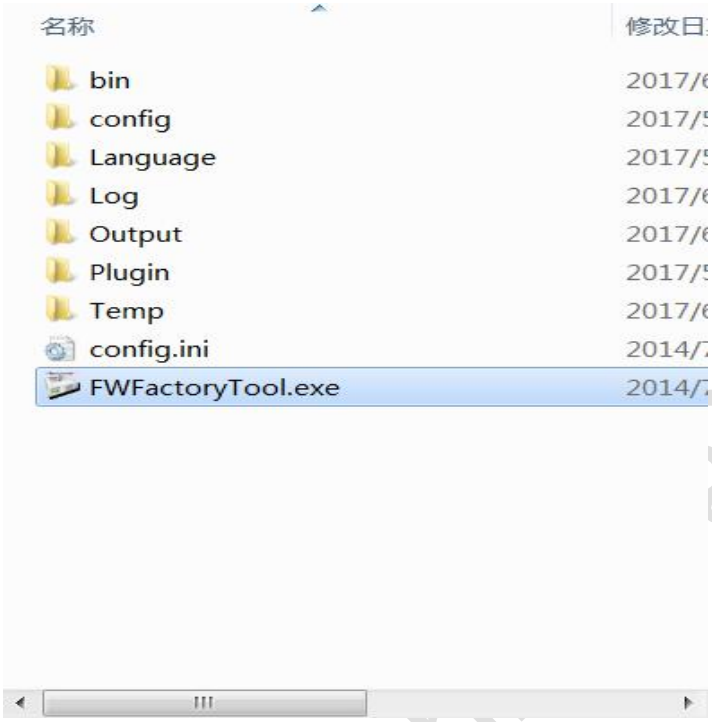
After replacing the animation files in part0 folder, compress file as below (must be compressed as *.zip file):

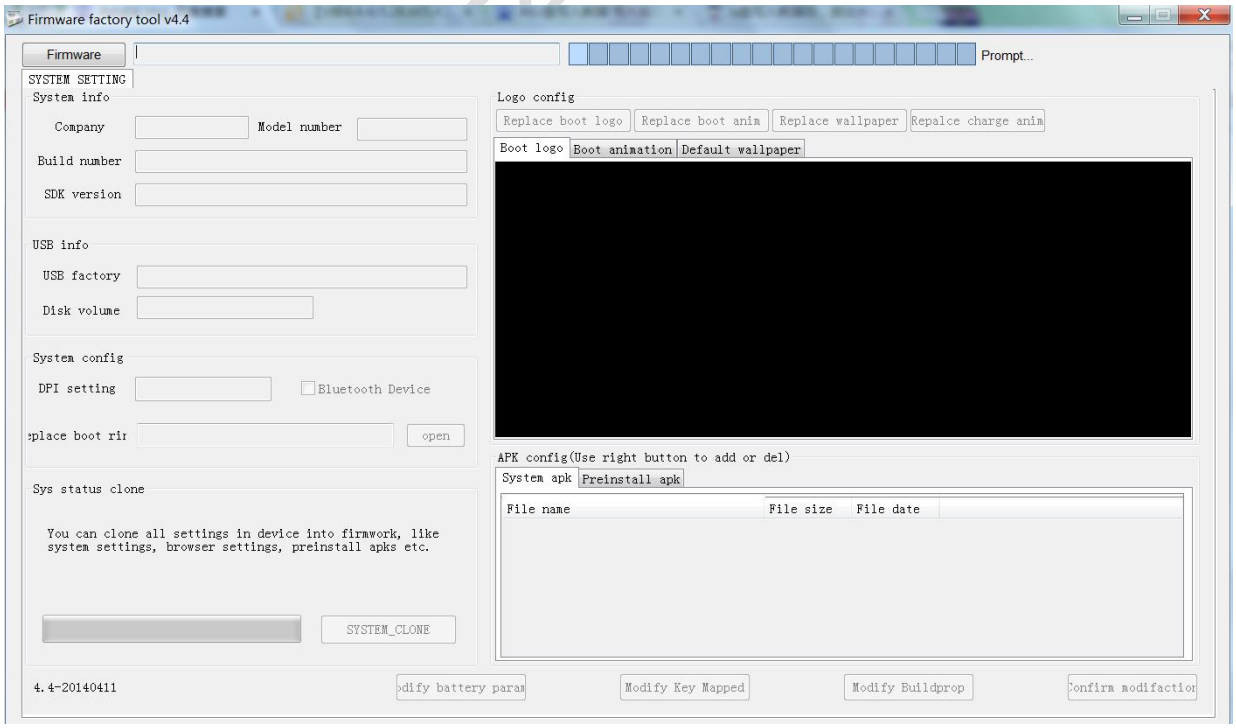


Note: Open compressed file to see if there are any idle files. If yes, delete them. Otherwise animation will fail when booting.

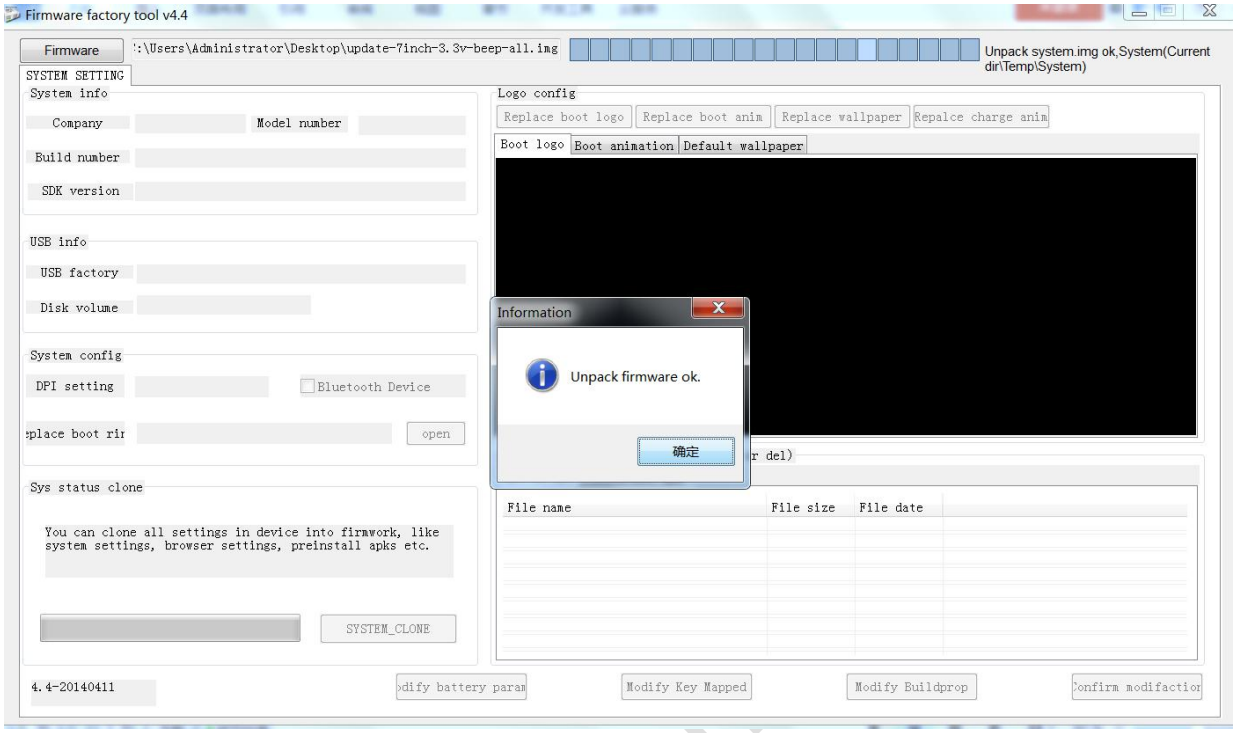
2.9.3.2 Replace boot animation:

Step 1: Open the firmware factory tool

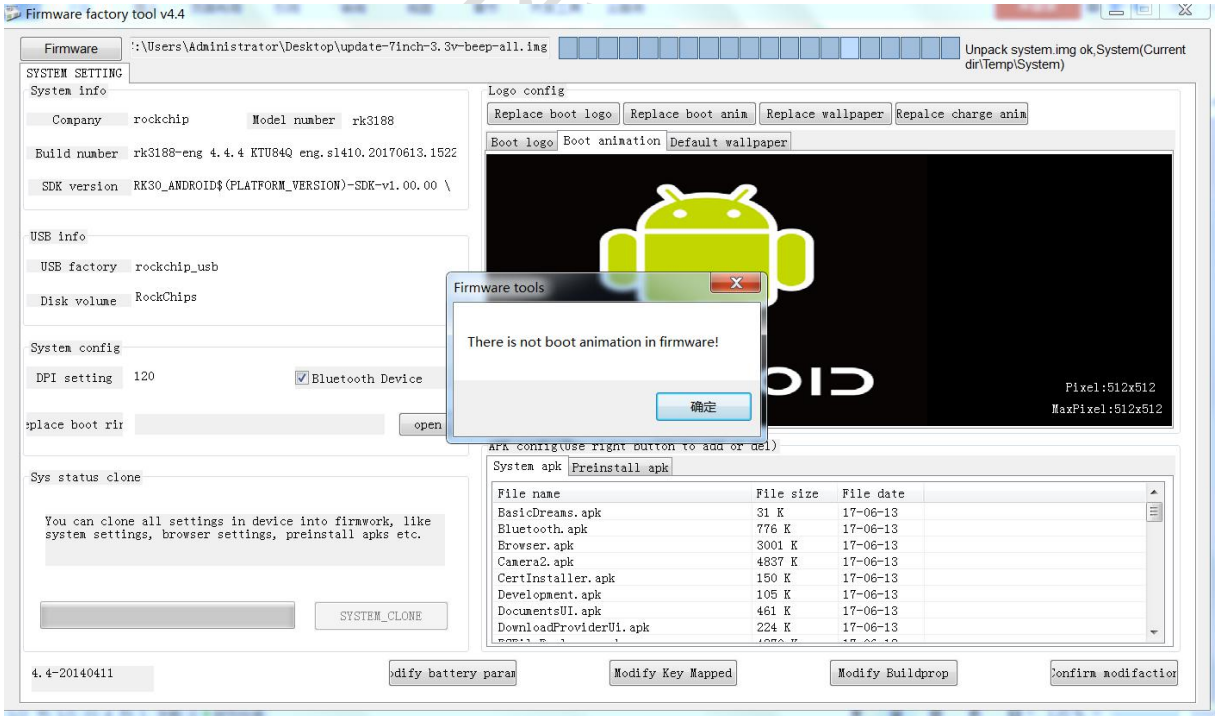




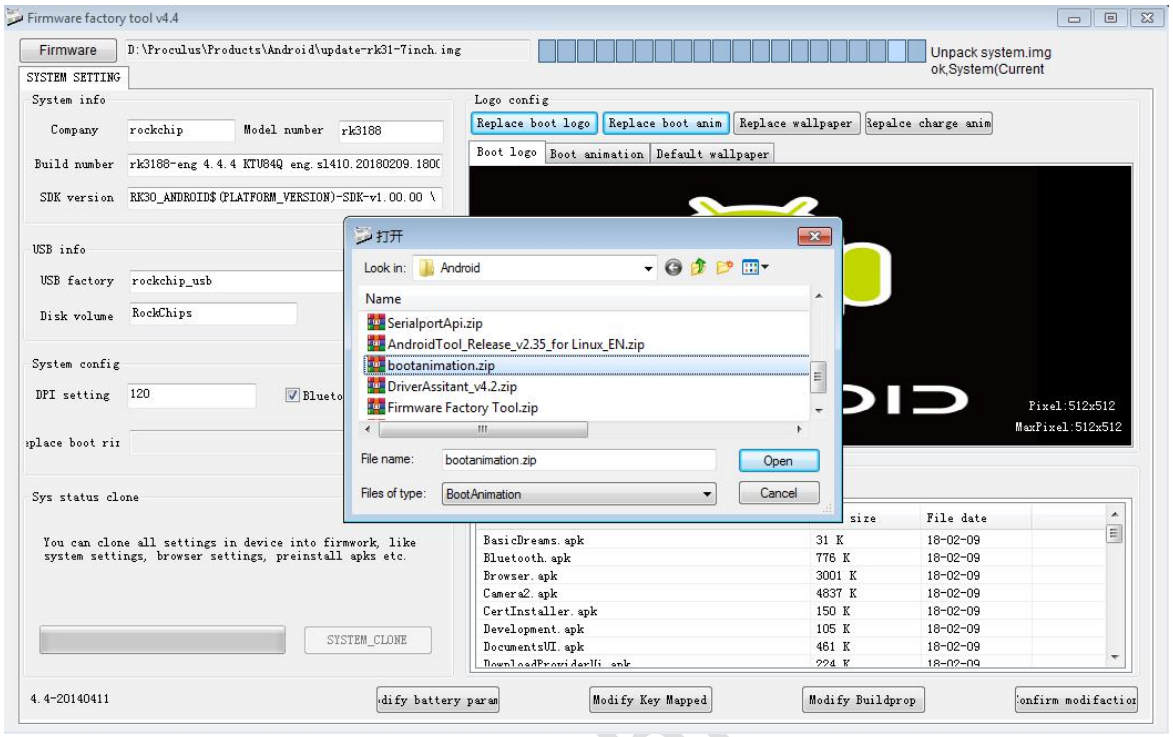
Step 2: Click “Firmware” and open the firmware



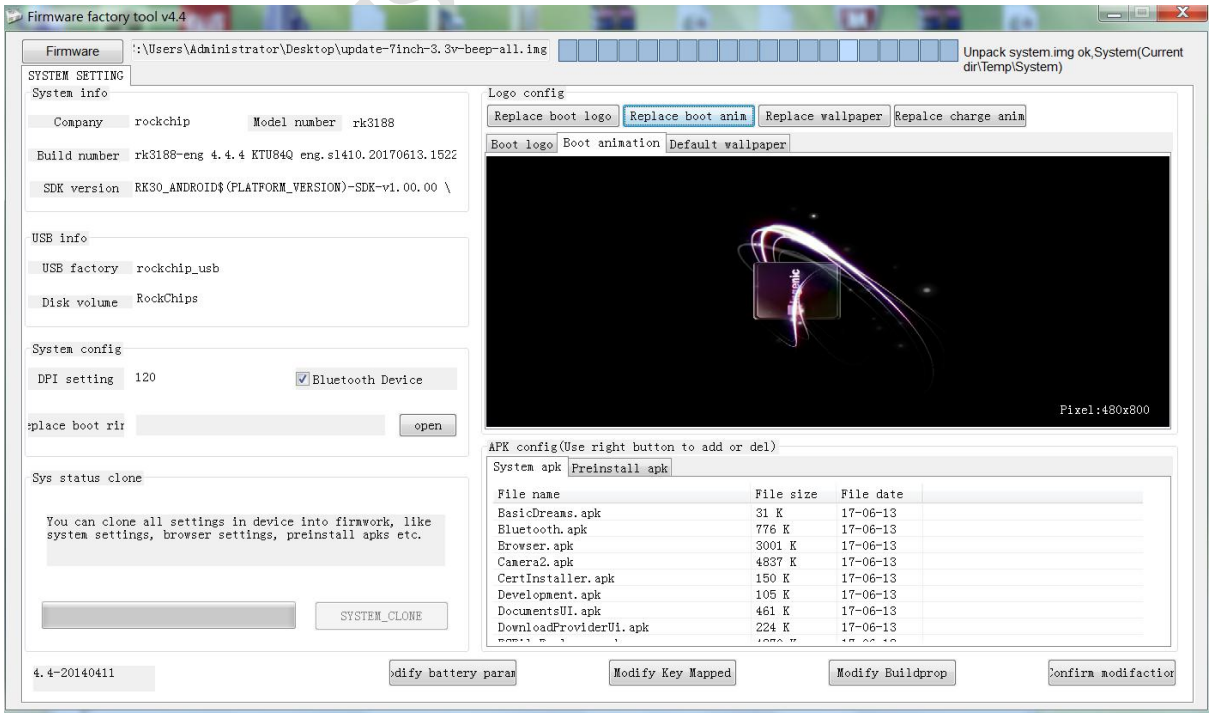
Step 3: Boot animation is not in the firmware. Click “OK” to continue.



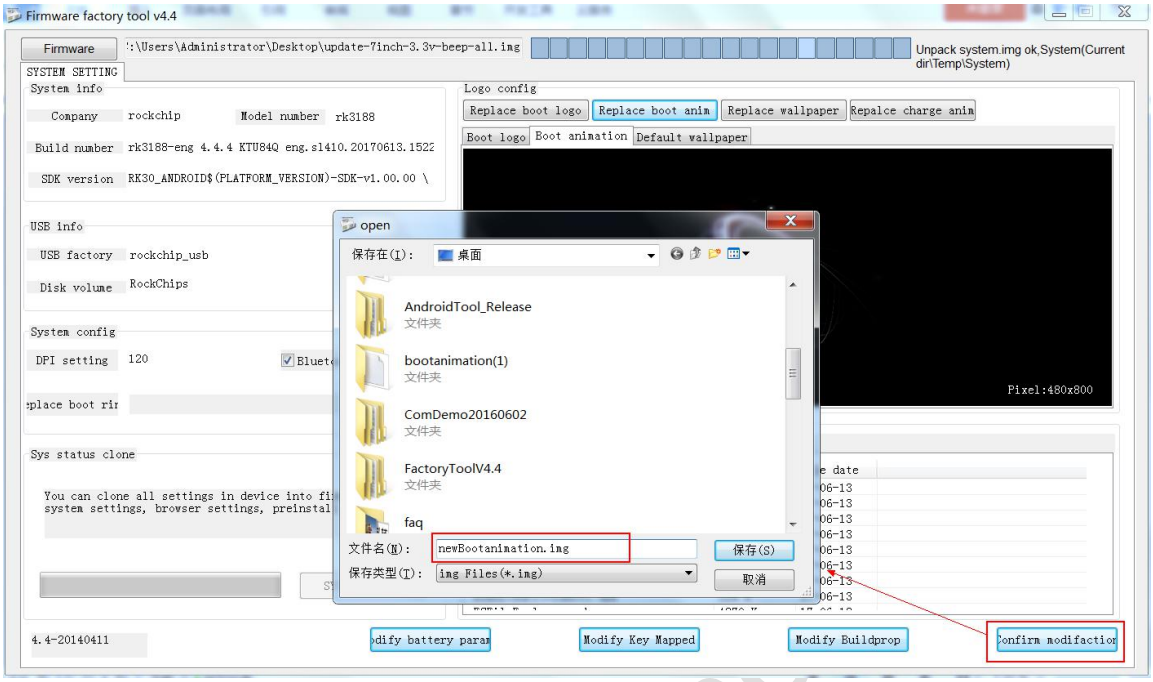
Step 4: Click “Replace Boot Animation” button and import the animation created in Step 1.



Step 5: Preview boot animation:



Step 5: Save boot animation

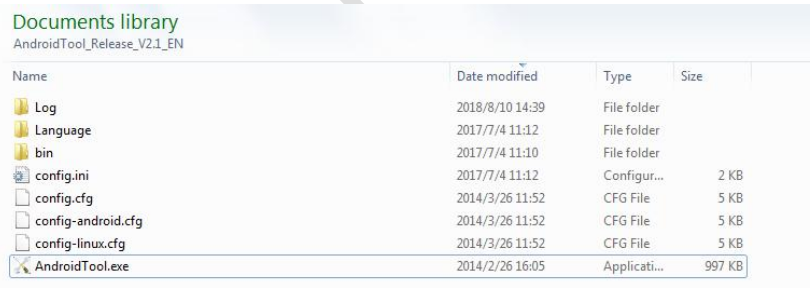


*Note: The file must be *.img format.*

2.10 Firmware Burning Process

Step1: Open AndroidTool.exe

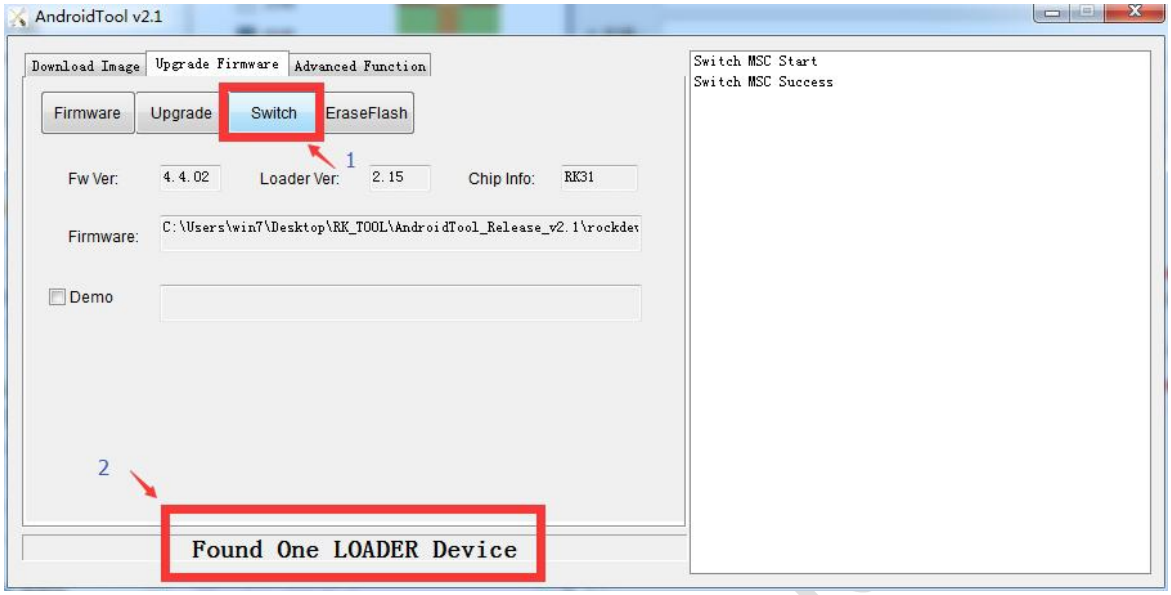
Download Link: www.proculustech.com/SoftwareResources/AndroidTool_Release.zip



Step2: Connect the device and select the firmware you want to burn

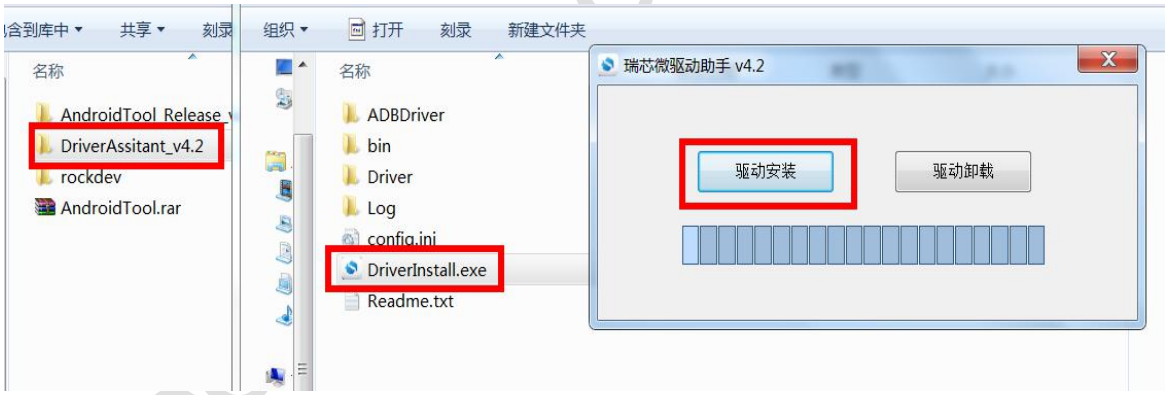
First it will show 1 ”Found One MSC Device”, second click 2 ”Upgrade Firmware”, then click 3 “Firmware”, and select the firmware you want to burn.

Click 1 “Switch”, and it will show 2 “Found One LOADER Device” after success.

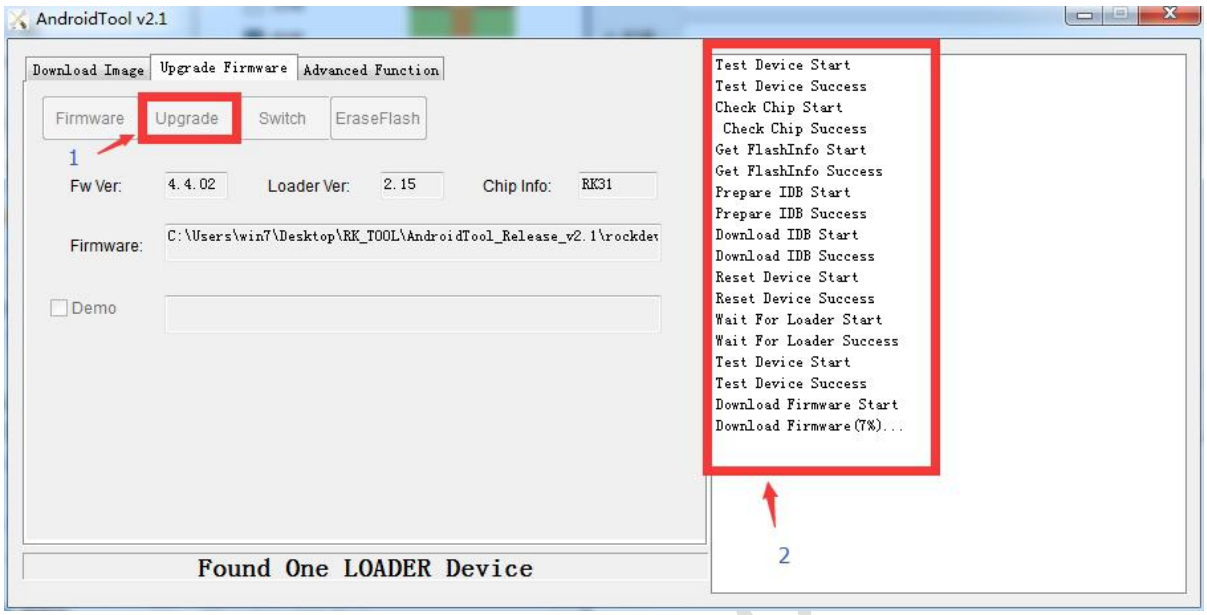


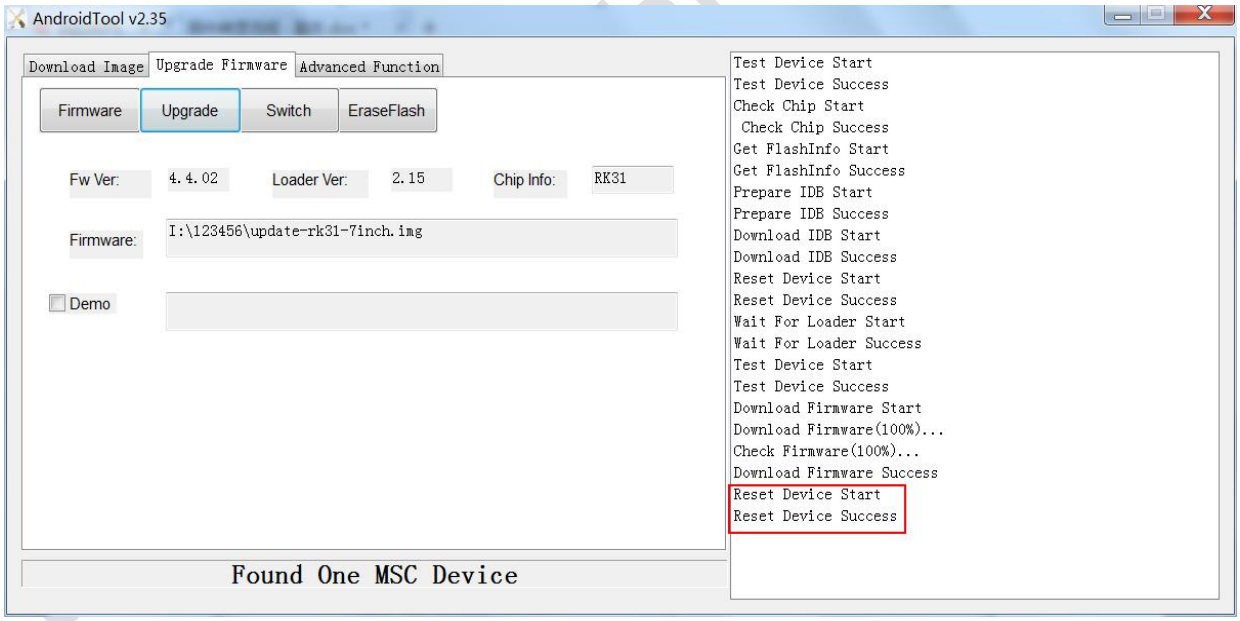
If you cannot load the firmware, please install the driver

Download Link:http://www.proculustech.com/SoftwareResources/DriverAssitant.zip



Step3: Upgrade and show “Reset Device Success” after burning succeeded.





2.11 Others

Demo Downloading Link:
http://www.proculustech.com/SoftwareResources/zq_demo.rar

Step 1: Import Class RootShellCmd.java

Download link: http://www.proculustech.com/SoftwareResources/RootShellCmd.java

Step 2: Invoke in Main class

2.11.1 Adjust brightness

```
RootShellCmd.getInstance().setBrightness(0);
```

2.11.2 Home button

```
RootShellCmd.getInstance().home();
```

2.11.3 Back button

```
RootShellCmd.getInstance().back();
```

2.11.4 Dynamic shutdown

```
RootShellCmd.getInstance().shutDown();
```

2.11.5 Dynamic reboot

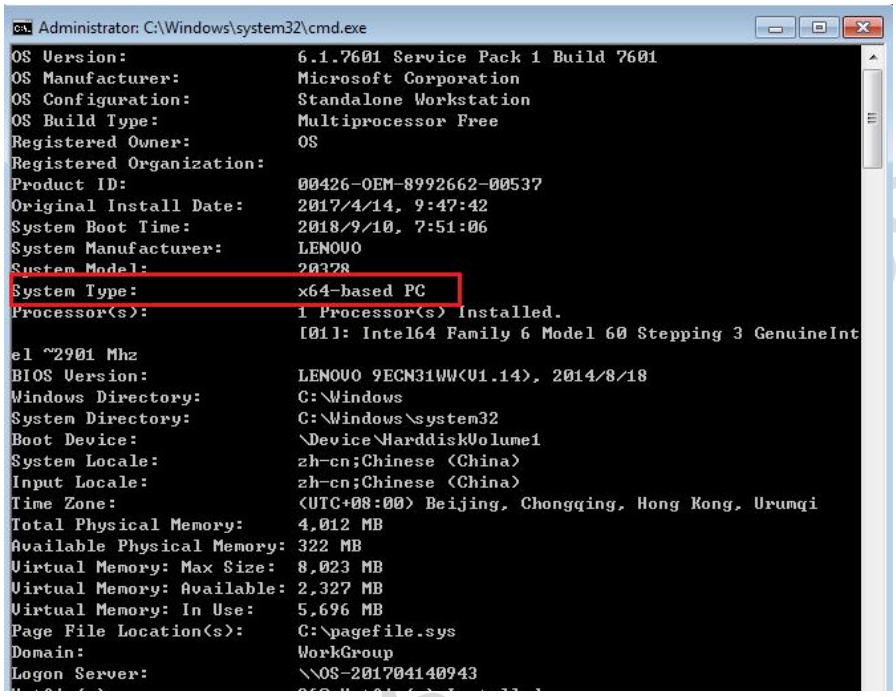
```
RootShellCmd.getInstance().reboot();
```

3. Matters Needing Attention

3.1 Possible Problems In JAVA Installation

1. How to check if my OS is 32-bit or 64-bit?

A: “Start” -> “Run”, input “CMD” and enter. Input “systeminfo” and enter. You can find it in “System Type” (x86 means 32-bit. X64 means 64-bit).



3.2 Possible Problems In Android Studio Installation

Android Studio must work with JDK8 or above version.

3.3 Serial Port

1. PATH

There are many serial ports on Android LCM, therefore it must be clearly aware of the path of the serial port (Do NOT invoke serial port 2 as it is system serial port).

2. Baud rate

Baud rate is the transmission speed which is crucial. If LCM can receive information but cannot work, it’s quite possible that baud rate is not right.

3. Gibberish

Gibberish is usually caused by the inconsistent of sending and receiving. Check if send & receive

baud rates, check bits, stop bits are consistent, or if the formats of data are consistent (HEX or ASCII).

4. Abnormal in data receiving & sending

First, check the cable connection. Most problems are caused by bad connections of the cables;

Second, Android LCM comes with a serial testing file. Short circuit DB9 transceiver and run the apk file to see if it's working fine;

3.4 USB Device

USB Device is recognized automatically when connecting to Internet. It will needs confirmation when offline. Android LCM will recognize automatically when USB connect to DEBUG interface.

If PC is connecting too many USB drive, it could possibly cause connection exception.

USB drive should be inserted into USB0. Device Manager will refresh and install new device. If not, please check the connection.

If it popup to install driver, go on with the auto-search.

After installation, there will be an "Android Tablet" in Device Manager. Check if there is an exclamatory mark on it. If yes, remove this driver and plug USB again.

3.5 Android LCM can't Connect to PC

1. Reinstall the driver;
2. Unplug and plug the USB again;
3. Reboot the LCM;
4. PC can only communicate with one Android device once.