



DWIN Android LCD Screen Development Guide

(31 Series & 32 Series)

Contents

Chapter 1 Product Introduction	1
1.1 OS Version	1
1.2 Development Method	1
1.3 Shipping List	1
1.4 Other Optional Accessories	1
1.5 Accessories Recommended for Self-Preparation	1
Chapter 2 Tool Introduction	2
2.1 The Tools Provided	2
2.2 Tools Which Might Require Self-Downloading	2
Chapter 3 Hardware Wiring	3
3.1 Serial Port Wiring Schematic Diagram	3
3.2 Serial Port Parameter Configuration	4
3.3 Usage Precautions	4
Chapter 4 Firmware Burning	5
4.1 Firmware Burning via SD Card	5
4.2 Firmware Burning via USB	6
Chapter 5 Development Examples	8
5.1 Common Interface Invocation Examples	8
5.1.1 Boot LOGO	8
5.1.2 Boot Animation (Multiple Images)	8
5.1.3 Boot Video	10
5.1.4 Receiving and Monitoring of Boot Broadcast	11
5.1.5 Top Status Bar	12
5.1.6 Bottom Navigation Bar	13
5.1.7 Top Dropdown Box	14
5.1.8 Silent Installation of APK	14
5.1.9 Reading External Storage Path	15
5.1.10 System Language Settings	15
5.1.11 System Time Settings	15
5.1.12 System Date Settings	16
5.1.13 Time Zone and Time Settings	16
5.1.14 Ethernet MATIC Network Parameters and DHCP Settings	17

5.1.15 Wi-Fi STATIC Network Parameters and DHCP Settings	17
5.1.16 Retrieval of IP, Gateway, DNS, and Subnet Mask	18
5.1.17 System Reboot	19
5.1.18 Interface Invocation Return Value	19
5.2 SerialPort	22
5.2.1 Serial Port Initialization	22
5.2.2 Serial Port Opening	22
5.2.3 Send Serial Port Data	22
5.2.4 Serial Port Closure	23
5.2.5 Resources Release	23
5.2.6 Obtain Supported Serial Ports	23
5.2.7 Serial Callback	24
5.3 Gpio	24
5.3.1 Gpio Initialization	24
5.3.2 Value of Gpio Settings	25
5.3.3 Gpio Direction Settings	25
5.3.4 Read Gpio Direction	25
5.3.5 Read Gpio Value	25
5.3.6 Initialization CallBack	26
5.4 FAQs	27
Chapter 6 Firmware Version Inquiry	28
Chapter 7 Revision Records	29

Chapter 1 Product Introduction

1.1 OS Version

linux 4.19 kernel, Android 11 version.

1.2 Development Method

Java development or installing and using existing Android applications (APKs)

1.3 Shipping List

- Standard screen*1
- Antenna*1

1.4 Optional Accessories

You can contact the DWIN sales to purchase the following accessories:

- Camera: support DWIN-defined mipi interface suitable for horizontal screen display, 5mp
support DWIN-defined mipi interface suitable for vertical display, 5mp
- Speaker: 8Ω2W, cable length 320mm, front-facing sound output
8Ω0.8W, cable length 180mm, side-facing sound output
- 4G module: China/India region version
European region version
Australian region version
- Power adapter: T050-32ZJB
For use with DMG80480T050_32/40 only

1.5 Accessories Recommended for Self-Preparation

- 12V 2A power supply
- USB TYPE-A to TYPE-C or Micro-USB adapter cable (according to the type of USB debugging interface indicated in the device specification)
- USB flash drive
- SD card (Max 64GB)
- Network cable
- The adapter board and related wires for connecting your serial port or power supply device
- Microphone (with socket 2PIN_1.25)
- USB driverless camera

Chapter 2 Tool Introduction

2.1 The Tools Provided

- For development, refer to the DEMO source code DWAndroidLibraryDemo: Examples of calling common interfaces such as serial ports, GPIO, and CAN.
- Source code of the hardware testing tool APP: DWINTEST
- Burning tools: SD card burning tool: SDDiskTool_V1.69
USB firmware burning tool: RKDevTool_v2.93 (There is a pre-installed ADB tool in the bin folder.)
- USB driver tool: DriverAssitant_v5.1.1
- Tool for modifying the boot animation and LOGO: FWFactoryTool_RK3566_v2.4

2.2 Tools Which Might Require Self-Downloading

- Android Studio (You can download it at <http://developer.android.google.cn/studio/archive>)

Chapter 3 Hardware Wiring

For the definition of the serial port interface, please refer to the product specification sheet, as shown in the following figure:

Peripherals and Interfaces

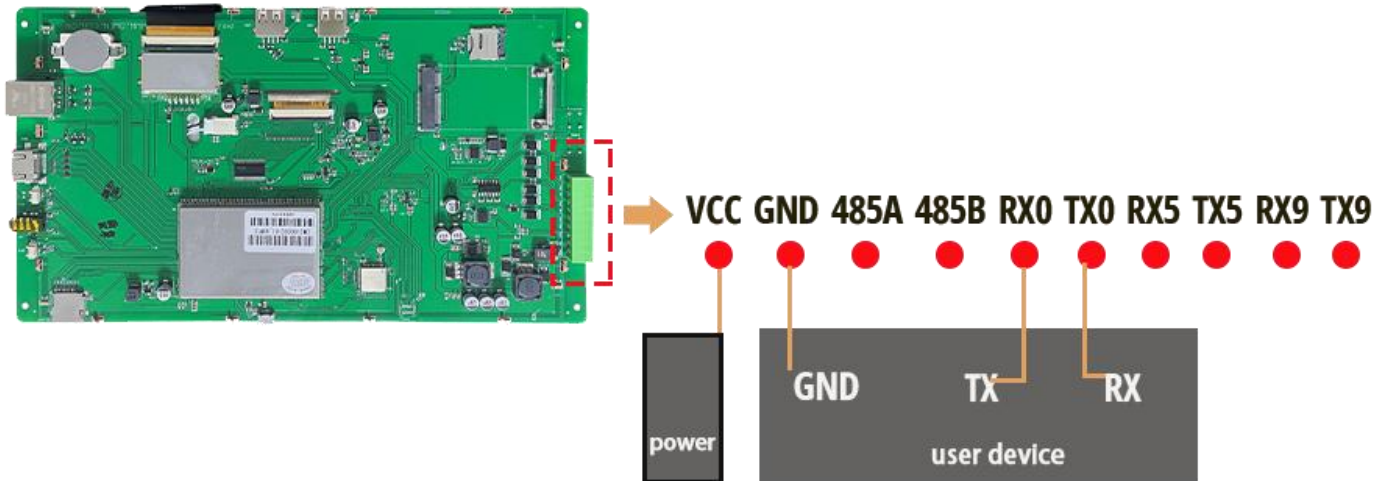
Properties	Parameters	Description
COM	2-way RS232	UART5 & UART9
	1-way RS485	UART8
	1-way TTL/COMS	UART0

3.1 Serial Port Parameter Configuration

GND: Ground, connect to GND pin of the user device.

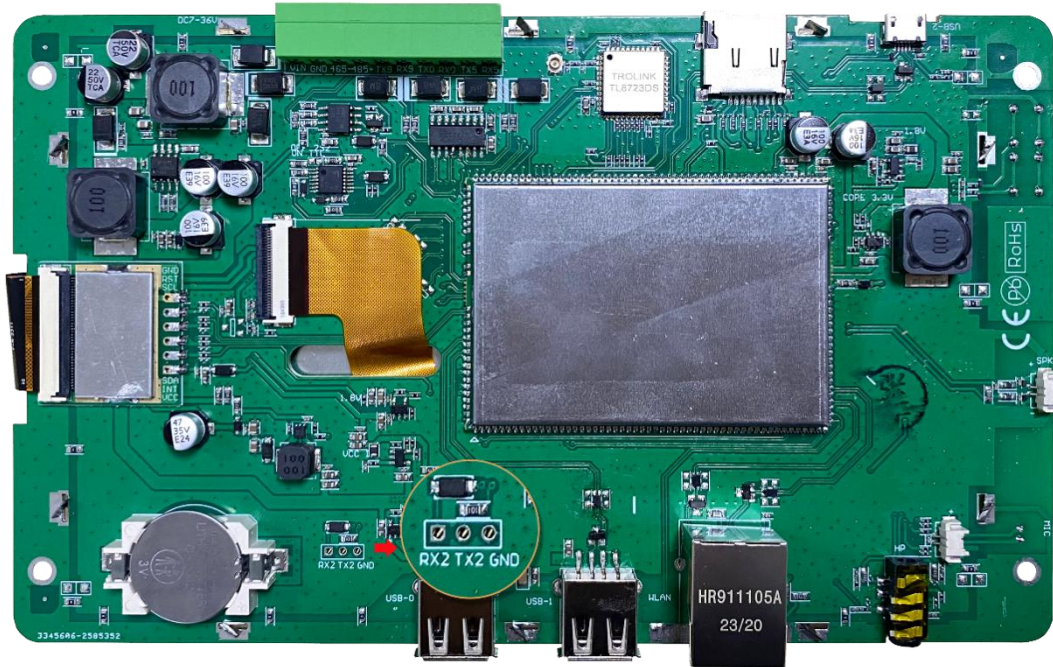
TXD: Transmit, connect to RX pin of the user device.

RXD: Receive, connect to TX pin of the user device.



3.2 Serial Parameter Settings

The baud rate of Serial Port 2 is 1500000, and the baud rate of the remaining serial ports is 115200.



3.3 Usage Precautions

- The standard firmware does not support the simultaneous use of keyboard input and barcode scanners when the android screen is connected to barcode scanner via USB. To use them simultaneously, please contact our sales for a customized firmware.
- To set a static IP address, please connect the device directly to the router using an Ethernet cable. Connecting the device to a computer for setup might lead to configuration failures.
- The Ethernet-related pages follow the official settings of RK3566 and are set to English by default. They will not change with the system language. If you need them to be displayed in Chinese or other languages, you can contact the sales for customized firmware.

Chapter 4 Firmware Burning

4.1 Firmware Burning via SD Card

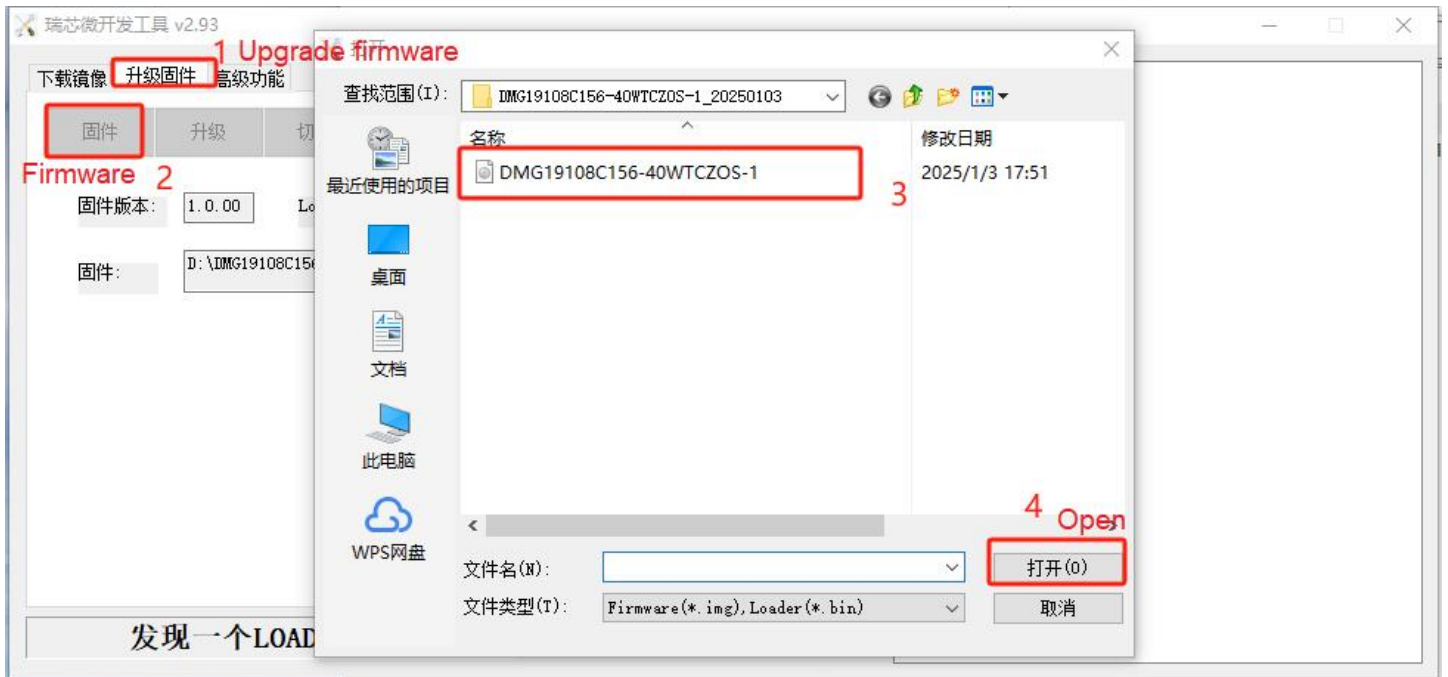
- When upgrading the firmware using an SD card, you need to write the firmware to the SD card using a tool on a computer. Currently, this operation can only be completed on a Windows operating system.
- Insert the card reader with the loaded SD card into the USB port of your computer. Open SDDiskTool_v1.69, and first select this card reader. Check the "Firmware Upgrade" box, then click "Select Firmware" to find the firmware to be programmed.



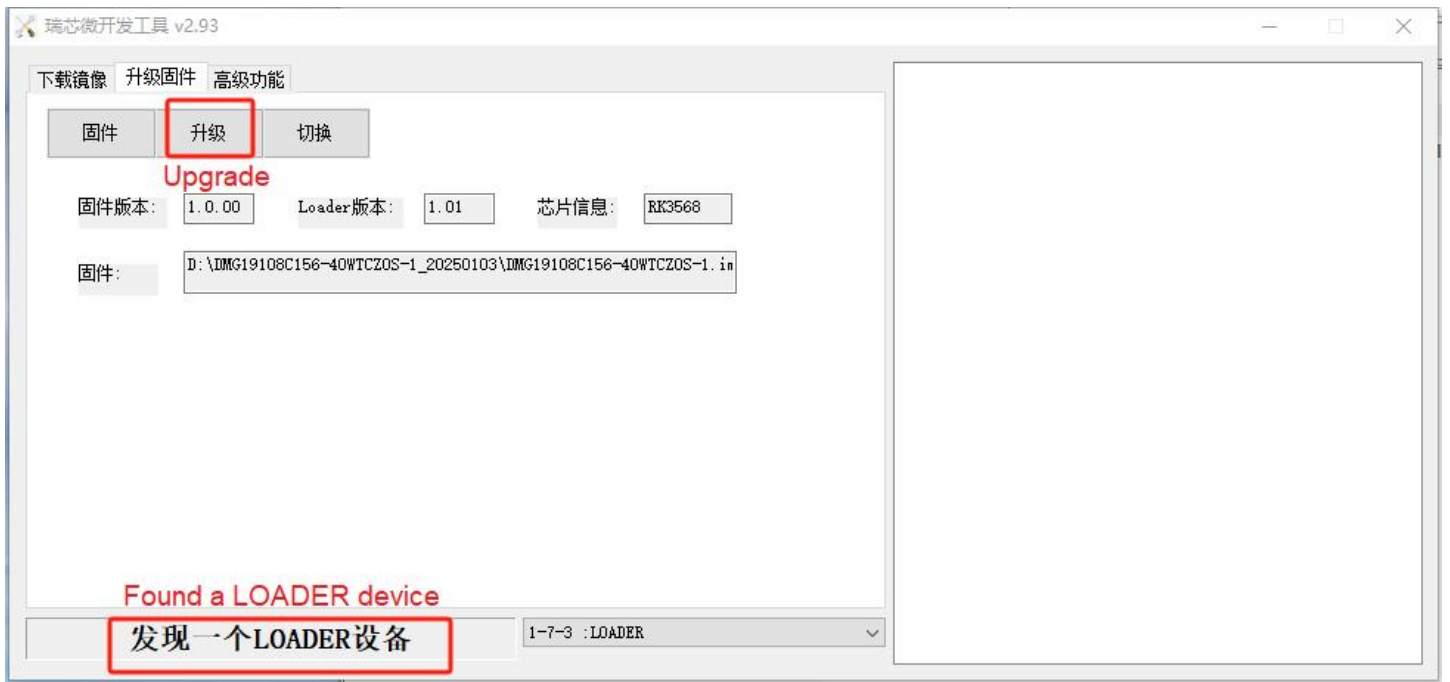
- After clicking "Create", just wait until the progress is completed.
- Remove the SD card and insert it into the SD card slot of the device. Power on the device with 12V DC. The screen will automatically start the upgrade. After the upgrade is completed, remove the SD card. The screen will automatically restart, indicating that the process is finished.

4.2 Firmware Burning via USB

- If it's the first time for the computer used for programming to perform the programming operation, you need to install the driver. Open the "DriverAssitant_v5.1.1" folder of the USB programming driver tool we provided, click on "DriverInstall.exe", and then complete the installation according to the prompts.
- Open RKDevTool_v2.93, click on the second tab "Upgrade Firmware", then click on "Firmware" and upload the img firmware package that needs to be programmed.



- Next, first use a USB cable to connect the device to the computer, and then power on the device with 12V DC. At this time, the RKDevTool will display the status "Found an ADB device".
- Click the switching function of the tool and wait until the update status of the tool shows "Found a LOADER device". At this time, you can click "Upgrade" and wait for the progress to be completed. After the burning process is finished, the device will restart.



Chapter 5 Development Examples

5.1 Common Interface Invocation Examples

The following development can refer to the DEMO named DWAndroidLibraryDemo. Please contact the sales or technical staff to obtain it.

5.1.1 Boot LOGO

The boot logos are divided into ubootlogo and kernellogo. Generally, they are replaced simultaneously. The image format is BMP. Please note the following two points:

- When replacing and compiling the boot logo in the system source code or replacing it using API interface, the image resolution should not exceed the screen resolution, and it should be controlled within 2M as much as possible.
- When using FWFactoryTool to modify the firmware package to replace the boot LOGO, the image resolution should not exceed 600*600 pixels, and the quality should be less than 1MB.

```
* @param path Logo path
* @return Return value of interface calling. Refer to DWErroCode
*/
int setBootLogo (String path);
```

Note: Re-burning firmware or restoring factory settings will not delete the logo. If you need to delete the logo during burning, please use the "erase" function of the burning tool first.

```
/**
 * Delete boot logo
 *
 * @return Return value of interface calling. Refer to DWErroCode
 */
int deleteBootLogo ();
```

5.1.2 Boot Animation (Multiple Images)

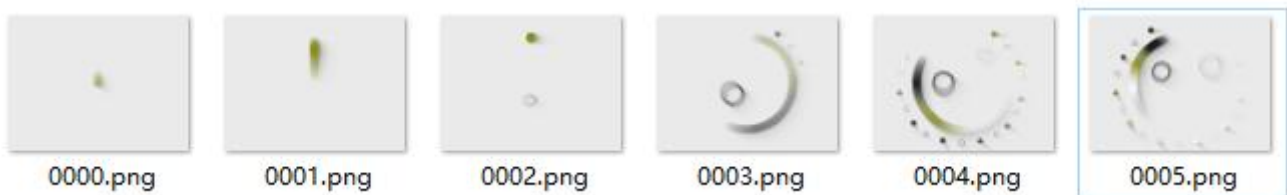
```
* @param path Boot animation path
* @return Return value of interface calling. Refer to DWErroCode
*/
int setBootAnimation (String path);
```

5.1.2.1 Create Bootanimation.zip

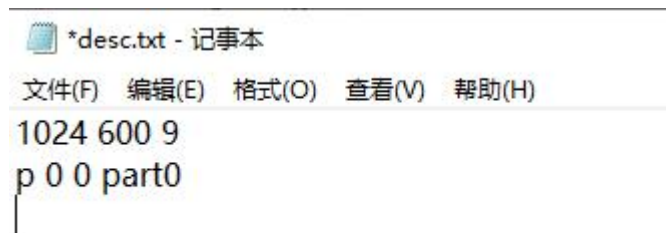
Bootanimation.zip contains part0 folder and desc.txt, or also can contain 2 folders as part0, part1, here we only take part0 as example.



Create a new folder and name it as part0, put the animation pictures in it (.png format), naming pictures with 000.png, 0001.png, 0002.png and so on. Please note that the image resolution should not exceed the screen resolution, and the memory for a single image should be controlled within 2M. If the images are too large or too many, it may cause the startup animation to be unsmoothly or frozen.



Create a new txt document and name it as desc.txt, Manually input the following content. **(Please note that after the last line, enter a blank line and save the file. Otherwise, the device will not be able to parse it.)**

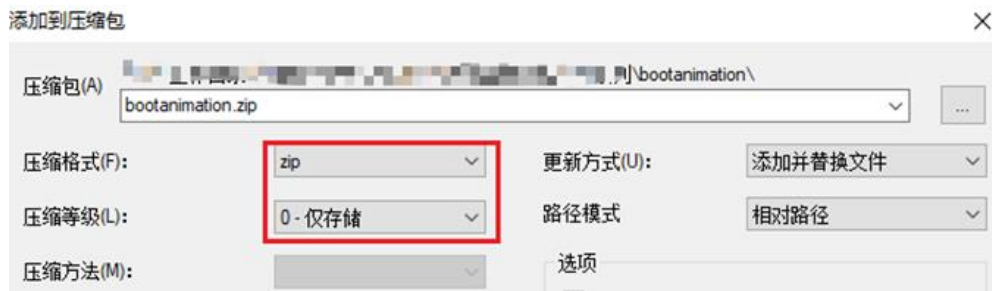


First line: 1024 600 9: 1024 and 600 means resolution is 1024*600p, 9 represent display 9 pictures every second;

Second line: pis a fixed starting point, The first number represent repeated times, 0 is infinite playback (If it is 3, it means it will play three times); The second 0 represents the interval time/frame rate between the two frames; The last part0 represents the animation folder.

Select both part0 and desc.txt simultaneously. Right-click the mouse and choose "Add to compressed file". Select "zip" as the compressed file format and "Store" as the compression method. After the

compression is completed, check that the compressed file does not contain the bootanimation folder.



5.1.2.2 Delete Bootanimation

Note: Re-burning firmware or restoring factory settings will not delete the boot animation. If you need to delete the bootanimaton during burning, please use the "erase" function of the burning tool first.

```
/**
 * Delete bootanimation (multiple pictures)
 *
 * @return Return value of interface calling. Refer to DWErrorCode
 */
int deleteBootAnimation ();
```

5.1.3 Boot Video

Boot video format: Modify the suffix of the video file to ts, and the duration can vary from a dozen seconds.

If both the boot animation (multiple images) and the boot video exist simultaneously, the system will display the boot video by default.

```
* @param path Startup video path
 * @return Return value of interface calling. Refer to DWErrorCode
 */
int setBootVideo (String path);
```

Delete the boot video as follows:

```
* Delete the startup video
 *
 * @return Return value of interface calling. Refer to DWErrorCode
 */
int deleteBootVideo ();
```

5.1.4 Receiving and Monitoring of Boot Broadcast

Regarding boot monitoring, the broadcast monitoring will make APP start automatically.

```
public class SelfStartReceiver extends BroadcastReceiver {

    @Override

    public void onReceive(Context context, Intent intent) {

        //Android A boot broadcast will be sent when the device is powered on: "android.intent.action.BOOT_COMPLETED"

        if (Intent.ACTION_BOOT_COMPLETED.equals(intent.getAction())) {

            Log.d("SelfStartReceiver", "The boot broadcast has been detected");

            Intent i = new Intent(context, MainActivity.class);

            i.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);

            context.startActivity(i);

        }

    }

}
```

Please refer to the following code for broadcast settings and permission configurations:

```
MainActivity.java x AndroidManifest.xml x SelfStartReceiver.java x
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.dwin.selfstartdemo">

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

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.SelfStartDemo">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

        <receiver
            android:name=".SelfStartReceiver"
            android:enabled="true"
            android:exported="true">
            <intent-filter android:priority="100">
                <action android:name="android.intent.action.BOOT_COMPLETED" />
            </intent-filter>
        </receiver>
    </application>
</manifest>
```

5.1.5 Top Status Bar

It can display and hide the top status bar, and read the status of the top status bar.

```
/**
 * Display the top status bar
 *
 * @return Return value of interface calling. Refer to DWErrorCode
 */
int showStatusBar ();

/**
```

```
* Hide the top status bar
*
* @return Return value of interface calling. Refer to DWErroRCode
*/
int hideStatusBar ();

/**
 * Read the status of the top status bar, display or hide
 *
 * @return Return value of interface calling.1: display, 0: hide
 */
int isStatusBarShow ();
```

5.1.6 Bottom Navigation Bar

It can display and hide the bottom navigation bar, read the status of the bottom navigation bar, display or hide the screenshot button within it, and read the status of the screenshot button.

```
/**
 * Display bottom navigation bar
 *
 * @return Return value of interface calling. Refer to DWErroRCode
*/
int showNavigationBar ();

/**
 * Hide bottom navigation bar
 *
 * @return Return value of interface calling. Refer to DWErroRCode
*/
int hideNavigationBar ();

/**
 * read the status of the bottom navigation bar, display or hide
 *
 * @return Return value of interface calling.1: display, 0: hide
 */
int isNavigationBarShow ();

/**
 * Display the screenshot button in the bottom navigation bar
 *
 * @return Return value of interface calling. Refer to DWErroRCode
*/
int showScreenShotButton ();
```



```
/**
 * Hide the screenshot button in the bottom navigation bar
 *
 * @return Return value of interface calling. Refer to DWErrorCode
 */
int hideScreenShotButton ();

/**
 * Read the status of the bottom navigation bar, display or hide
 *
 * @return Return value of interface calling.1: display, 0: hide
 */
int isScreenShotShow ();
```

5.1.7 Top Dropdown Box

It can display and hide the top drop-down menu, and read the status of the top drop-down menu.

```
/**
 * Display the top dropdown box
 *
 * @return Return value of interface calling. Refer to DWErrorCode
 */
int showDropDownMenu ();

/**
 * Hide the top dropdown box
 *
 * @return Return value of interface calling. Refer to DWErrorCode
 */
int hideDropDownMenu ();

/**
 * Read the status of top dropdown box, display or hide
 *
 * @return Return value of interface calling. 1: display, 0: hide
 */
int isDropDownMenuShow ();
```

5.1.8 Silent Installation of APK

```
/**
 * Install APK silently
 *
```

```
* @param path APK path
* @return Return value of interface calling. Refer to DWErrorCode
*/
int installApp (String path);
```

5.1.9 Reading External Storage Path

```
/**
 * Read external storage path
 *
 * @return Return path collection
 */
ArrayList<StorageBean> readExternalStoragePath ();
```

5.1.10 System Language Settings

```
/**
 * Set system language
 *
 * @param locale Language format Locale.SIMPLIFIED_CHINESE, Locale.ENGLISH etc.
 * @return Return value of interface calling. Refer to DWErrorCode
 */
int setLanguage (Locale locale);
```

5.1.11 System Time Settings

```
/**
 * Set system time
 *
 * @param hour Hour
 * @param minute Minute
 * @return Return value of interface calling. Refer to DWErrorCode
 */
int setTime (int hour, int minute);

/**
 * Set system time
 *
 * @param year Year
 * @param month Month
 * @param day Day
 * @param hour Hour
 * @param minute Minute
 * @return Return value of interface calling. Refer to DWErrorCode
```

```
*/
int setTime (int year, int month, int day, int hour, int minute);

/**
 * Enable automatic synchronization of time
 *
 * @param auto 1: Automatically synchronizing time, 0: Not automatically synchronizing time
 * @return Return value of interface calling. Refer to DWEErrorCode
 */
int setTimeAuto (int auto);
```

5.1.12 System Date Settings

```
/**
 * Set system date
 *
 * @param year Year
 * @param month Month
 * @param day Day
 * @return Return value of interface calling. Refer to DWEErrorCode
 */
int setDate (int year, int month, int day);
```

5.1.13 Time Zone and Time Settings

Set time zone, enable automatic time zone synchronization, check if automatic synchronization function is enabled, set 12 or 24-hour clock.

```
/**
 * Set time zone
 *
 * @param timeZone Time Zone Name, such as Asia/Shanghai
 */
void setTimeZone (String timeZone);

/**
 * Enable automatic time zone synchronization
 *
 * @param auto 1: Automatically synchronize time zones, 0: Not automatically synchronize time zones
 * @return Return value of interface calling. Refer to DWEErrorCode
 */
int setTimeZoneAuto (int auto);
```

```
/**
 * Check if automatic time zone synchronization is enabled
 *
 * @return Return value of interface calling. Refer to DWEErrorCode
 */
int checkTimeZoneAuto ();

/**
 * Set "12" or "24" hour clock
 *
 * @param value "12" or "24"
 * @return Return value of interface calling. Refer to DWEErrorCode
 */
int setTime_12_24 (String value);
```

5.1.14 Ethernet MATIC Network Parameters and DHCP Settings

```
/**
 * Set ethernet MATIC network parameters
 *
 * @param address IP
 * @param mask Subnet mask code
 * @param gateway Gateway
 * @param dns1 DNS1
 * @param dns2 DNS2
 * @return Return value of interface calling. Refer to DWEErrorCode
 */
int setEthernetStaticConfig (String address, String mask, String gateway, String dns1, String dns2);

/**
 * Set Ethernet DHCP
 *
 * @return Return value of interface calling. Refer to DWEErrorCode
 */
int setEthernetDynamicConfig ();
```

5.1.15 Wi-Fi STATIC Network Parameters and DHCP Settings

```
/**
 * Set WiFi STATIC parameters
 *
 * @param address IP
 * @param mask Mask
 * @param gateway Gateway
```

```
* @param dns1 DNS1
* @param dns2 DNS2
* @return Return value of interface calling. Refer to DWErrorCode
*/
int setWiFiStaticConfig (String address, String mask, String gateway, String dns1, String dns2);

/**
 * Set WiFi DHCP
 *
 * @return Return value of interface calling. Refer to DWErrorCode
 */
int setWiFiDynamicConfig ();
```

5.1.16 Retrieval of Get IP, Gateway, DNS, Subnet Mask

```
/**
 * Get IP
 *
 * @return Return to IP collection
 */
ArrayList<String> getNetWorkIP ();

/**
 * Get gateway
 *
 * @return Return to gateway collection
 */
ArrayList<String> getNetWorkGateway ();

/**
 * Get DNS
 *
 * @return Return to DNScollection
 */
ArrayList<String> getNetWorkDns ();

/**
 * Get subnet Mask
 *
 * @return Return to subnet Mask collection
 */
ArrayList<String> getNetworkMask ();
```

5.1.17 System Reboot

```
/**
 * Reboot System
 *
 * @return Return value of interface calling. Refer to DWErrorCode
 */
int rebootSystem ();
```

5.1.18 Interface Invocation Return Value

Interface invocation return value reference: DWErrorCode, only for Android SDK.

```
/**
 * Interface calling return code
 */
public class DWErrorCode {

    /**
     * Succeed
     */
    public final static int DW_COMMON_SUCCESS = 0;

    /**
     * unknown error
     */
    public final static int DW_COMMON_ERROR_UNKNOUN = -1;

    /**
     * Android SDK do not support it (perhaps due to firmware mismatch)
     */
    public final static int DW_COMMON_ERROR_SDK_NOT_SUPPORT = -2;

    /**
     * Settings.Global.putXXX Writing value failed
     */
    public final static int DW_COMMON_ERROR_SET_SETTINGS_GLOBAL = -3;

    /**
     * Settings.Global.getXXX Reading value failed
     */
    public final static int DW_COMMON_ERROR_GET_SETTINGS_GLOBAL = -4;

    /**
     * Settings.System.putXXX Writing value failed
     */
```

```
public final static int DW_COMMON_ERROR_SET_SETTINGS_SYSTEM = -5;

/**
 * Settings.System.getXXX Reading value failed
 */
public final static int DW_COMMON_ERROR_GET_SETTINGS_SYSTEM = -6;

/**
 * Path is empty or null
 */
public final static int DW_COMMON_ERROR_PATH_INVALID = -7;

/**
 * Path is not exist
 */
public final static int DW_COMMON_ERROR_PATH_NOT_EXIST = -8;

/**
 * Copy file error
 */
public final static int DW_COMMON_ERROR_COPY_FILE = -9;

/**
 * Deleting file failed
 */
public final static int DW_COMMON_ERROR_DELETE_FILE = -10;

/**
 * bootlogo is not a .BMP file
 */
public final static int DW_BOOT_LOGO_ERROR_NOT_BMP_FILE = -100;

/**
 * bootanimation is not a .zip or .ts file
 */
public final static int DW_BOOT_ANIMATION_ERROR_NOT_ZIP_OR_TS_FILE = -101;

/**
 * the installed file is not an APK file
 */
public final static int DW_INSTALL_APP_ERROR_NOT_APK_FILE = -200;

/**
 * installation failed
 */
```

```
public final static int DW_INSTALL_APP_ERROR = -201;

/**
 * Failed to read external file
 */
public final static int DW_EXTERNAL_STORAGE_READ_ERROR = -300;

/**
 * Failed to set system language
 */
public final static int DW_LANGUAGE_SET_ERROR = -400;

/**
 * Failed to set system time or date
 */
public final static int DW_TIME_OR_DATE_SET_ERROR = -500;

/**
 * Failed to configure Ethernet STATIC network
 */
public final static int DW_ETHERNET_STATIC_SET_ERROR = -600;

/**
 * Failed to configure Ethernet DHCP network
 */
public final static int DW_ETHERNET_DYNAMIC_SET_ERROR = -601;

/**
 * Failed to configure WiFi STATIC network
 */
public final static int DW_WIFI_STATIC_SET_ERROR = -602;

/**
 * Failed to configure WiFi DHCP network
 */
public final static int DW_WIFI_DYNAMIC_SET_ERROR = -603;

/**
 * System restart failed
 */
public final static int DW_REBOOT_SYSTEM_ERROR = -600;
}
```


5.2 SerialPort

Serial port invocation method: SerialManage.getInstance().xxx, such as SerialManage.getInstance().init

5.2.1 Serial Port Initialization

```
/**
 *Serial port initialization
 *
 * @param serialInter Serial callback
 */
public void init (SerialInter serialInter)
```

5.2.2 Serial Port Opening

```
/**
 * Open serial port
 *
 * @param devicesPath Serial port address
 * @param isReadLoop Whether to continuously monitor the data returned by the serial port
 * @param baudrate Baud-rate
 * @param dataBits Data Bits
 * @param stopBits Stop Bits
 * @param parity Parity
 * @param readTimePeriod ReadTimePeriod
 * @return Open successfully or not
 */
public void open (String devicesPath, boolean isReadLoop, int baudrate, int dataBits, int stopBits, int parity, long readTimePeriod)
```

5.2.3 Send Serial Port Data

```
/**
 * Send serial port data
 *
 * @param msg Serial port data
 */
public void send (byte[] msg)
```

5.2.4 Serial Port Closure

```
/**
 * Close serial port
 */
public void close()
```

5.2.5 Resources Release

```
/**
 * Release resources
 */
public void release ()
```

5.2.6 Obtain Supported Serial Ports

```
/**
 * Obtain supported serial ports
 *
 * @return Return to serial port collection
 */
public String[] getDriverList ()
```

5.2.7 Serial Callback

```
/**
 * Serial callback
 */
public interface SerialInter {

/**
 * Connection result callback
 * @param path Serial port address ( (when multiple serial ports need to be processed uniformly, addresses can be used
to distinguish them)
 * @param isSuccess Connection successful or not
 */
void connectMsg (String path,boolean isSuccess);

/**
 * Loop read, callback the data read
 * @param path Serial port address ( (when multiple serial ports need to be processed uniformly, addresses can be used
to distinguish them)
 * @param bytes Read data
 * @param size Data Length
 */
void readData (String path, byte[] bytes, int size);
```

5.3 Gpio

Gpio interface invocation method: Gpio.xxxx, such as Gpio.initGpio

5.3.1 Gpio Initialization

```
/**
 * Initialize GPIO
 *
 * @param gpioArr GPIO port that needs to be configured, just a numerical number is enough
 * @param value 0:low, 1:high
 * @param direction In:input, out:output
 * @param context Above and follow context
 * @param gpioInitCallBack Initialize callback interface
 */
public static void initGpio (int[] gpioArr, int value, String direction, Context context, GpioInitCallBack
gpioInitCallBack)
```

5.3.2 Value of GPIO Settings

```
/**
 * Set value of GPIO
 *
 * @param gpio GPIO port, such as gpio100, value is 100
 * @param value Set the voltage level, 0: low, 1: high
 * @return Return value, 0: success, <0:fail
 */
public static native int setGpioValue (int gpio, int value);
```

5.3.3 GPIO Direction Settings

```
/**
 * Set GPIO direction
 *
 * @param gpio GPIO port, such as gpio100, value is 100
 * @param direction Set gpiodirection, 0:in; 1:out
 * @return Return value, 0: success, <0:fail
 */
private static native int setGpioDirection (int gpio, int direction);
```

5.3.4 Read GPIO Direction

```
/**
 * Read GPIO directio
 *
 * @param gpio GPIO port, such as gpio100, value is 100
 * @return Return value GPIO direction (in/out)
 */
private static native int getGpioDirections (int gpio);
```

5.3.5 Read GPIO Value

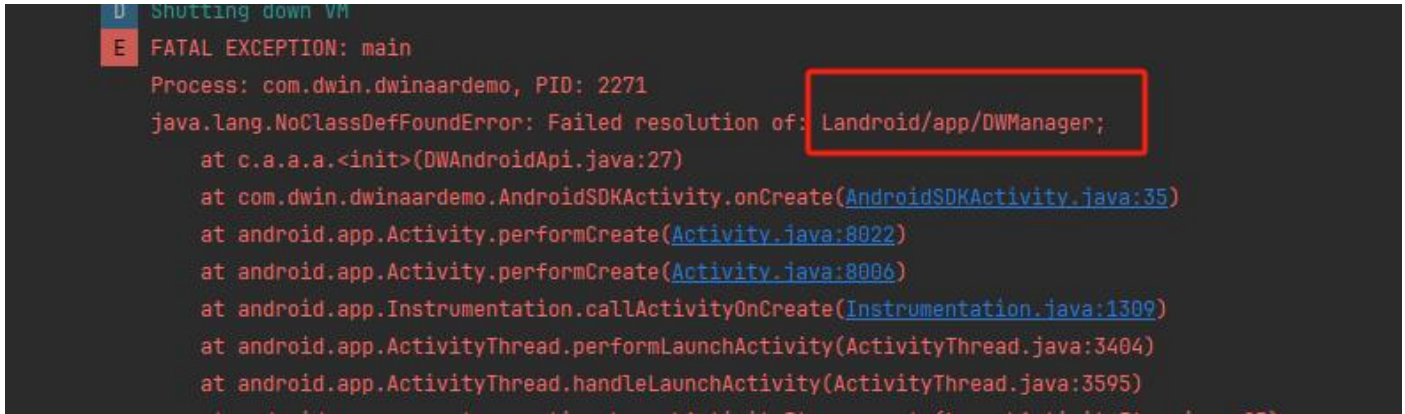
```
/**
 * Read GPIO value
 *
 * @param gpio GPIO port, such as gpio100, value is 100
 * @return Return value
 */
public static native int getGpioValue (int gpio);
```

5.3.6 Initialization CallBack

```
public interface GpioInitCallBack {  
    /**  
     * Initialization callBack method  
     *  
     * @param success Initialization succeed  
     * @param gpio Collection of successfully initialized GPIO ports  
     */  
    void initState(boolean success, List<Integer> gpio);  
}
```

5.4 FAQs

- There is an error when running the DWINAndroidLibraryDemo:



```

D Shutting down VM
E FATAL EXCEPTION: main
Process: com.dwin.dwinaardemo, PID: 2271
java.lang.NoClassDefFoundError: Failed resolution of: Landroid/app/DWManager;
    at c.a.a.a.<init>(DWAndroidApi.java:27)
    at com.dwin.dwinaardemo.AndroidSDKActivity.onCreate(AndroidSDKActivity.java:35)
    at android.app.Activity.performCreate(Activity.java:8022)
    at android.app.Activity.performCreate(Activity.java:8006)
    at android.app.Instrumentation.callActivityOnCreate(Instrumentation.java:1309)
    at android.app.ActivityThread.performLaunchActivity(ActivityThread.java:3404)
    at android.app.ActivityThread.handleLaunchActivity(ActivityThread.java:3595)

```

Reason: The latest version of the firmware has not been burned.

Solution: Contact the sales to obtain the new version of the firmware and burn it.

- After the APP is downloaded to the screen, its size does not match that in the code.

Reason: The UI has not been designed according to the screen resolution.

Solution: Design the UI with a 1:1 ratio between px and dp. If the screen density (density) is 160, UI designers and Android engineers can calculate based on the ratio of px:dp = 1:1. If the screen density is not 160, Android engineers need to convert the lengths proportionally according to the actual design drawings.

- Error

Unable to find method "void org.gradle.api.internal.DefaultDomainObjectSet.<init>(java.lang.Class)"

'void org.gradle.api.internal.DefaultDomainObjectSet.<init>(java.lang.Class)'

Gradle's dependency cache may be corrupt (this sometimes occurs after a network connection timeout.)

Reason: The development tool and the Gradle version are inconsistent.

Solution: Users can create a new project using their own tools, and then copy the code (such as the source code, lib library, and dependent libraries) to it.

Chapter 6 Firmware Version Inquiry

The firmware of the 32 series products has been uniformly upgraded since the batch of 20240904, optimizing some inconveniences that may occur in the old firmware.

Since the firmware of the new and old versions is not fully compatible, you can query the firmware version in the following ways:

1. Check the product model label. The product with a production date after 20240904 is the one with the new version firmware.

2. Check the firmware compilation date in the software. The firmware with a compilation date on or after 20240902 is the new version.

The new firmware provides a series of SDKs and interfaces that are convenient for secondary development. If you have received a product with the old firmware, you can contact our salesperson to obtain the new firmware for burning

Firmware Version Upgrade Records		
Firmware Name	Release Time	Contents
Update.img	2023.8.8	It only contains a simple example of hiding the status bar.
Update.img	2024.9.4	It includes general functions, such as APP examples and the usage methods of interface APIs.
Update.img	2024.11.3-2024.11.13	Update DWIN TEST and solve the problem of the boot animation stuttering.

Chapter 7 Revision Records

Revision	Date	Description	Editor
00	2023.02.20	First release	Xu Jiayang
01	2024.08.15	Add Chapter 1	Chen Yan
02	2024.09.05	Add Chapter 7	Chen Yan
03	2024.10.31	Add a new speaker B01279 with a higher volume	Chen Yan
04	2024.11.07	Add boot broadcast receiving and monitoring.	Chen Yan
05	2024.11.15	Add the precautions for updating the boot logo in the new firmware.	Chen Yan
06	2024.11.20	Add the usage precautions for 1.3, as well as the firmware version upgrade records.	Chen Yan
07	2025.4.23	Optimize the overall logic and expression of the development guide, and remove the descriptions of the old firmware versions.	Chen Xian