

Motherboard API

Programming Manual

Version	V1.7.2
Date	2023-05-22

Notice: Copyright in this document belongs to the original content owner. All rights are reserved. Content may be revised without notice; contact the provider for the latest version.

Revision History

Version	Date	Change
1.0.0	2017-05-12	First version of this document.
1.1.0	2017-06-15	Added the serial-port close interface.
1.2.0	2017-07-04	Added instructions for importing the library in Android Studio.
1.3.0	2017-08-18	Added HDMI-IN interfaces and APK signing instructions.
1.3.1	2017-12-26	Updated APK signing address: http://120.78.220.29:18080/
1.3.2	2018-03-05	Corrected wording in the display-density section.
1.4.0	2018-04-09	Added permission requirements for screen rotation.
1.5.0	2018-07-30	Added Android Studio 3.0 integration settings.
1.6.0	2018-10-30	Added permission requirements for <code>setSystemTime</code> .
1.6.1	2018-11-12	Updated the signing-server IP address.
1.6.2	2019-02-18	Added the signing URL for Android 7.1 and later.
1.6.3	2020-09-28	Added Android Studio code-obfuscation settings.
1.7.0	2021-12-15	V1.2.0.20211022 supports operation without root; fixed the DNS string format for the static-IP interface on Android 7.1+; added timed power and application startup-management broadcasts.
1.7.1	2022-03-18	Corrected text in the daily timed power-on/off example.
1.7.2	2023-05-22	Added Android 8.0+ invocation notes for timed power and application startup-management broadcasts.

Contents

1 Programming Instructions	7
1.1 Eclipse IDE Integration	7
1.2 Android Studio Integration	8
1.3 Calling the API	11
1.4 APK System Signing	11
2 Hardware Management	13
2.1 Wireless and Network	13
2.1.1 Read Ethernet State	13
2.1.2 Enable / Disable Ethernet	13
2.1.3 Set a Static Ethernet Address	13
2.1.4 Set Ethernet DHCP Mode	14
2.1.5 Read Ethernet Settings	14
2.1.6 Read Ethernet MAC Address	14
2.1.7 Read Current Network Type	15
2.1.8 Wi-Fi Network Operations	15
2.1.9 Read Mobile-Network Settings	15
2.2 Storage	15
2.2.1 Read Internal-Storage Path	15
2.2.2 Read External SD-Card Path	16

2.2.3 Read External USB-Drive Path	16
2.2.4 Read Storage Capacity	16
2.3 Runtime Memory	17
2.3.1 Read System Memory Size	17
2.4 Backlight Control	17
2.4.1 Turn the LCD Backlight On / Off	17
2.5 System Time	17
2.5.1 Set the Android System Clock	17
2.6 GPIO	18
2.6.1 Enable an I/O Port	18
2.6.2 Set I/O Direction to Input	18
2.6.3 Set I/O Direction to Output	18
2.6.4 Read I/O Level	19
2.6.5 Set I/O Level	19
2.7 Serial Port	19
2.7.1 Open a Serial Port	20
2.7.2 Read Serial Data	20
2.7.3 Write Serial Data	20
2.7.4 Close a Serial Port	21
2.8 Hardware Watchdog	21
2.8.1 Enable the Watchdog	21
2.8.2 Feed the Watchdog	21

2.8.3 Disable the Watchdog	21
2.9 Power Control	22
2.9.1 Hardware Shutdown	22
2.9.2 Software Restart	22
2.9.3 Hardware Restart	22
2.9.4 Timed Power-On	23
2.10 HDMI-IN	23
2.10.1 Read HDMI Input State	23
2.10.2 Read HDMI Input Resolution	23
3 Android Management	25
3.1 Display Management	25
3.1.1 Read Resolution	25
3.1.2 Capture the Screen	25
3.1.3 Rotate the Screen	26
3.1.4 Read Display Density	26
3.1.5 Set Display Density	26
3.2 Navigation Bar	26
3.2.1 Show the Navigation Bar	27
3.2.2 Hide the Navigation Bar	27
3.2.3 Enable / Disable Swipe	27
3.2.4 Automatically Hide the Navigation Bar	27
3.3 Status Bar	28

3.4 Application Management	28
3.4.1 Read Application List	28
3.4.2 Silent Application Installation	28
3.4.3 Automatic Application Installation	28
3.4.4 Silent Application Uninstallation	29
3.4.5 Automatic Application Uninstallation	29
3.4.6 Automatic Application Startup	29
3.5 System Information	29
3.5.1 Read Android Version	29
3.5.2 Read Android Serial Number	30
3.5.3 Read System Version	30
3.5.4 Read Kernel Version	30
3.5.5 Read API Version	30
3.5.6 Read Wi-Fi Hardware Address	30
3.5.7 Read Ethernet Hardware Address	31
3.6 System Upgrade	31
3.6.1 System OTA Upgrade	31
3.7 Shell Command Execution	31
3.7.1 Execute a Shell Command	31
4 Timed Power-On/Off Broadcast Interface	34
5 Application Startup-Management Broadcast Interface	36

1 Programming Instructions

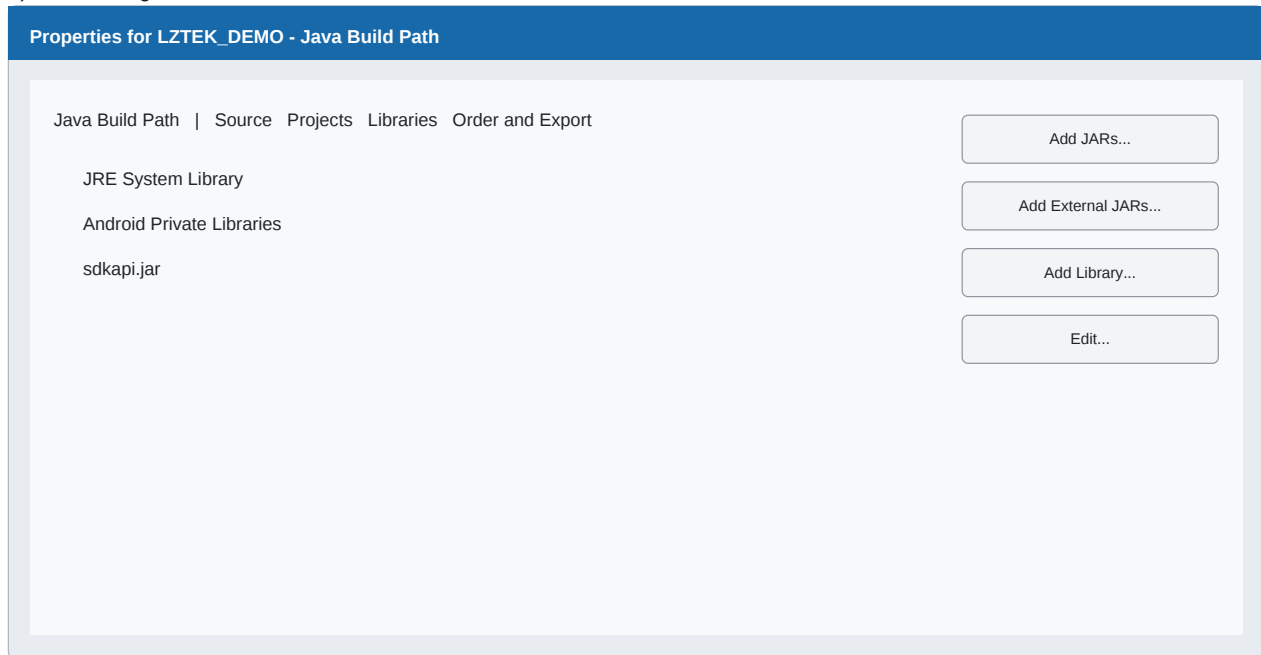
Integrate the supplied library files into the project and call them as described in this manual. Some APIs require root privileges; consult the motherboard vendor for the Android root method. Each API description states whether root is required.

The APIs in this document require the company's Android system version 20170518 or later.

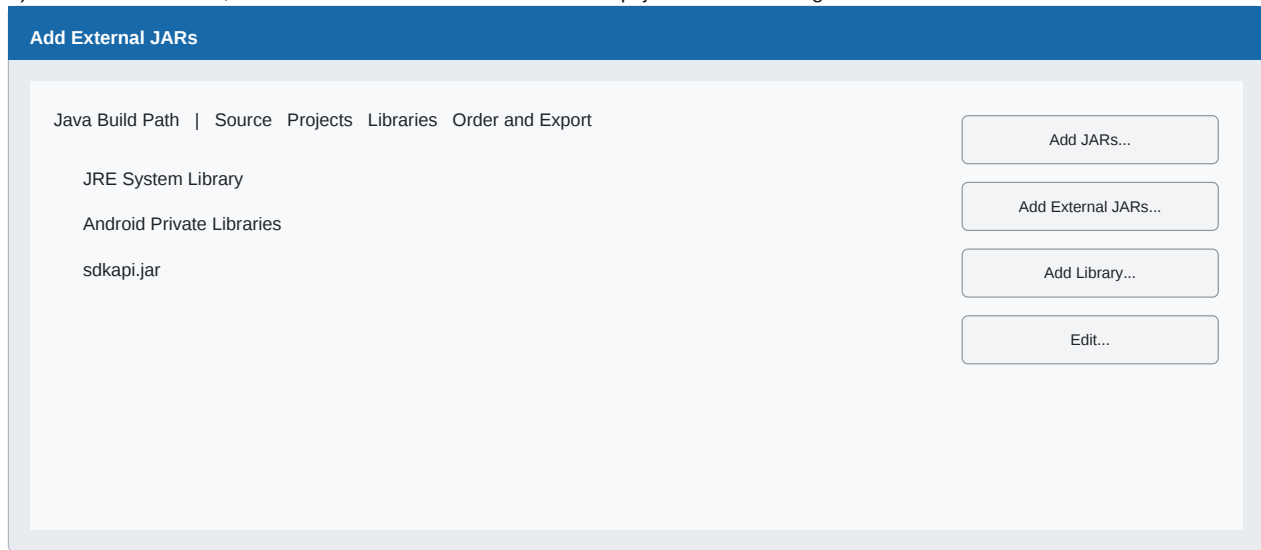
1.1 Eclipse IDE Integration

Use the API in Eclipse as follows:

- 1) Open Eclipse IDE, select the project, right-click it, and choose Properties.
- 2) In the dialog, select Java Build Path and then the Libraries tab.



3) On the Libraries tab, click Add External JARs and select sdkapi.jar in the file dialog.

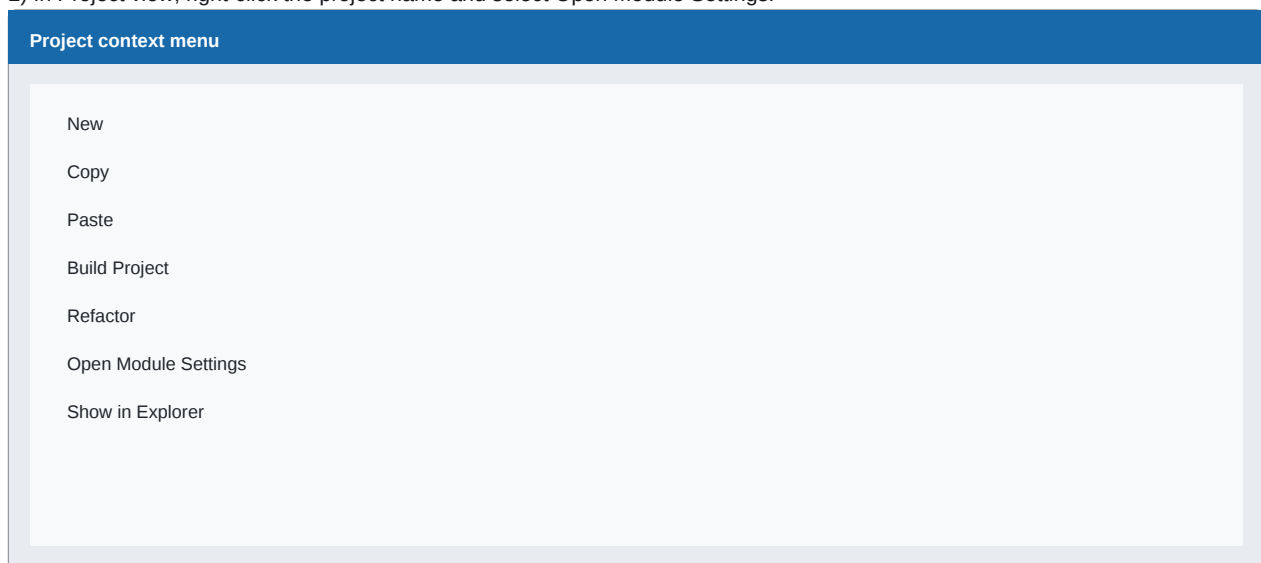


4) Confirm the selection to add sdkapi.jar.

1.2 Android Studio Integration

Use the API in Android Studio as follows:

- 1) Place sdkapi.jar in the module's third-party library folder, such as app\libs.
- 2) In Project view, right-click the project name and select Open Module Settings.



3) Select the Dependencies tab.

Module Settings - Dependencies

PropertiesSigningFlavorsBuild TypesDependencies

+ sdkapi.jar

Scope: Provided

4) Click the plus sign on the right and select Jar Dependency.

Dependency Scope

libs/sdkapi.jar

Compile

Provided

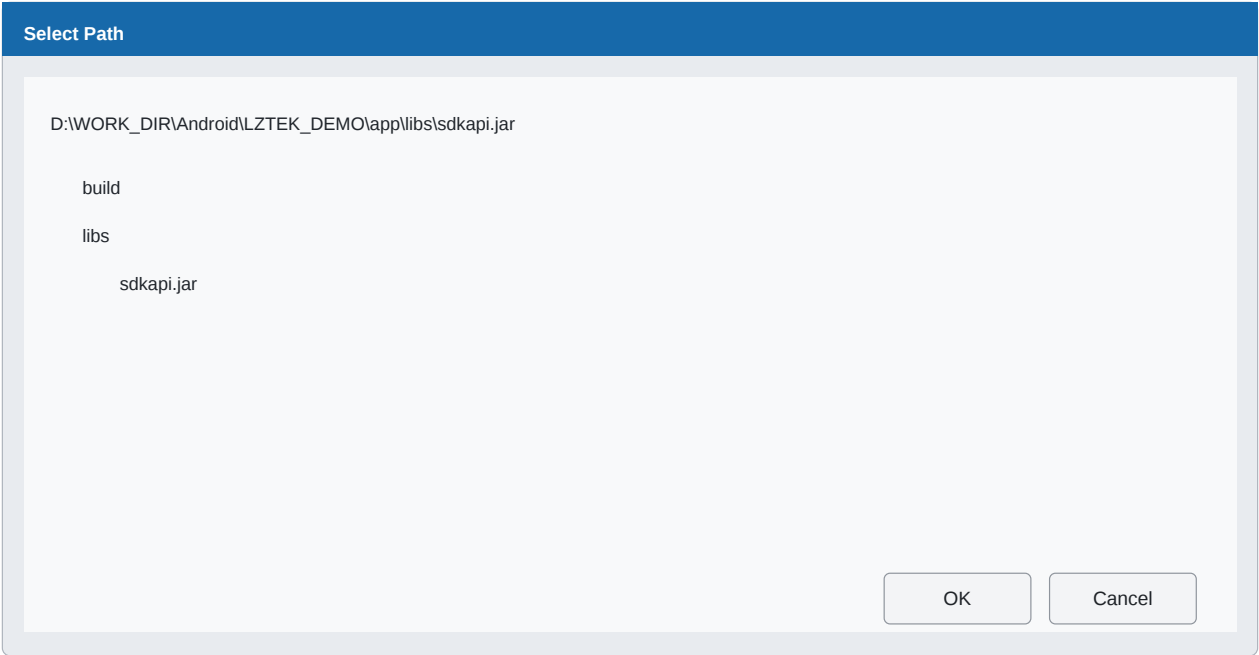
APK

Test compile

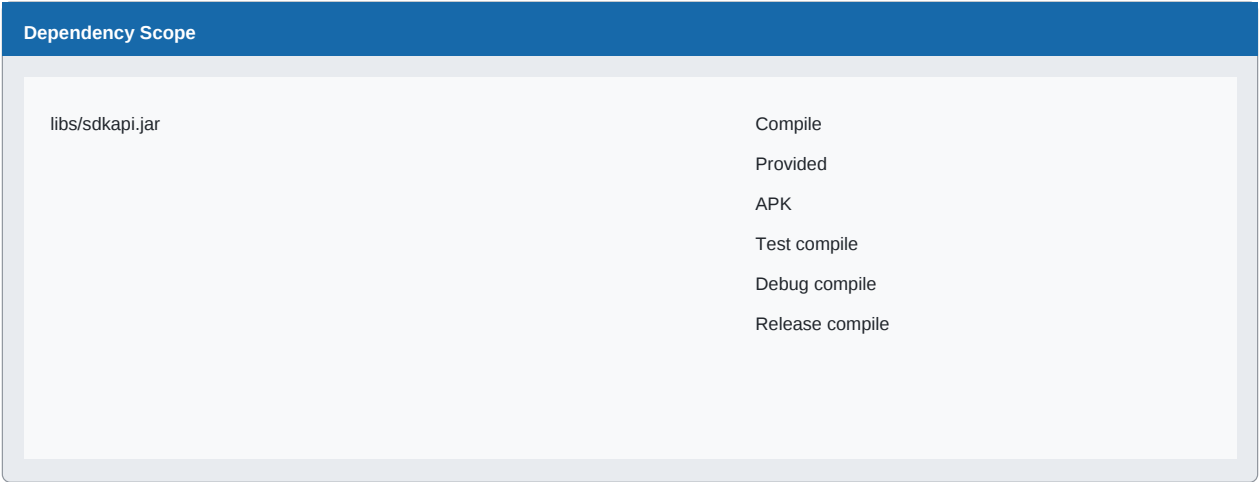
Debug compile

Release compile

5) Select sdkapi.jar in the file chooser and click OK.



6) Select the sdkapi.jar row, choose Provided in the Scope column, and click OK.



Note 1: For Android Studio versions earlier than 3.0, edit the module build.gradle file and add:

```
dependencies {  
    provided files('libs/sdkapi.jar')  
}
```

Note 2: For Android Studio 3.0 or later, add:

```
dependencies {  
    compileOnly files('libs/sdkapi.jar')  
}
```

For code obfuscation, add these lines to proguard-rules.pro:

```
-dontwarn com.lztek.toolkit.**  
-keep class com.lztek.toolkit.** { *; }
```

1.3 Calling the API

Add sdkapi.jar to the project. All APIs are called through an Lztek object, for example:

```
Lztek lztek = Lztek.create(context);  
boolean enable = lztek.getEthEnable();
```

1.4 APK System Signing

For system signing, upload the APK to <http://47.107.162.209:18080> (below Android 7.1) or <http://47.107.162.209:19090> (Android 7.1 and later), then download the signed APK.

APK Signing Upload

Choose APK

Drop APK here to upload

Signed and original files remain on the server for one hour. Download promptly.

The signing server automatically signs uploaded APK files. The original and signed files are retained for only one hour; download them promptly. Use a browser with HTML5 upload support.
End of the programming-instructions section.

2 Hardware Management

This chapter describes software interfaces for motherboard hardware resources, including wireless and network connectivity, storage, runtime memory, backlight control, system time, GPIO, serial ports, the hardware watchdog, shutdown/restart, and timed power control.

2.1 Wireless and Network

Functions: configure and read Ethernet IP parameters, enable or disable Ethernet, and read the current network connection.

Add this permission to the manifest when using Ethernet configuration:

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

2.1.1 Read Ethernet State

boolean getEthEnable()

Parameter / Return	Type	Description	Example
Return value	boolean	true: Ethernet enabled; false: Ethernet disabled	

2.1.2 Enable / Disable Ethernet

void setEthEnable(boolean enable)

Parameter / Return	Type	Description	Example
enable	boolean	true: enable Ethernet; false: disable Ethernet	

2.1.3 Set a Static Ethernet Address

void setEthIpAddress(String ip, String mask, String gateway, String dns)

Sets a static Ethernet address. If Ethernet was previously in DHCP mode, this call disables DHCP.

Parameter / Return	Type	Description	Example
ip	String	IP address	192.168.1.200
mask	String	Subnet mask	255.255.255.0
gateway	String	Gateway address	192.168.1.1
dns	String	DNS server address	8.8.8.8

2.1.4 Set Ethernet DHCP Mode
void setEthDhcpMode()

Configures Ethernet to obtain an IP address automatically. To leave DHCP mode, call setEthIpAddress with a static configuration.

2.1.5 Read Ethernet Settings
AddrInfo getEthAddrInfo()

Returns a network-address information object defined as public class AddrInfo { }. It includes DHCP mode, IP address, subnet mask, gateway, and related values.

Parameter / Return	Type	Description	Example
Return value	AddrInfo	AddrInfo object	

2.1.6 Read Ethernet MAC Address
String getEthMac()

Reads the Ethernet hardware-address string. By default, this address is generated from the CPU ID or motherboard serial number. Contact the original manufacturer's technical support if a specific MAC address must be assigned.

Parameter / Return	Type	Description	Example
Return value	String	Ethernet MAC address	00:01:02:03:04:05

2.1.7 Read Current Network Type

Use Android `ConnectivityManager.getActiveNetworkInfo()` to obtain a `NetworkInfo` object, then evaluate its `getType()` result.

```
ConnectivityManager cm = (ConnectivityManager) getSystemService(Context.CONNECTIVITY_SERVICE);
NetworkInfo networkInfo = cm.getActiveNetworkInfo();
int netType = networkInfo.getType();
// Compare with ConnectivityManager.TYPE_ETHERNET, TYPE_WIFI, or TYPE_MOBILE.
```

2.1.8 Wi-Fi Network Operations

Use the standard Android API directly.

2.1.9 Read Mobile-Network Settings

`AddrInfo getMobileAddrInfo()`

Returns a network-address information object defined as public class `AddrInfo { }`. At present, only the IP address and subnet mask are supported.

Parameter / Return	Type	Description	Example
Return value	<code>AddrInfo</code>	<code>AddrInfo</code> object	

2.2 Storage

Functions: read internal-storage, external SD-card, and external USB-drive paths, and determine storage capacity.

2.2.1 Read Internal-Storage Path

`String getInternalStoragePath()`

Reads the built-in storage path, such as /mnt/embsd or /mnt/internal_sd. For better compatibility, prefer android.os.Environment.getExternalStorageDirectory().

Parameter / Return	Type	Description	Example
Return value	String	Built-in storage path	/mnt/internal_sd

2.2.2 Read External SD-Card Path
String getStorageCardPath()

Reads the external SD-card path, such as /mnt/extsd or /mnt/external_sd.

Parameter / Return	Type	Description	Example
Return value	String	External SD-card path	/mnt/external_sd

2.2.3 Read External USB-Drive Path
String getUsbStoragePath()

Reads the USB-storage path, such as /mnt/udisk or /mnt/usb_storage.

Parameter / Return	Type	Description	Example
Return value	String	External USB-storage path	/mnt/usb_storage

2.2.4 Read Storage Capacity

Use the system API to read storage capacity. Example for an external SD card:

```
android.os.StatFs statfs = new android.os.StatFs("/mnt/extsd");
long blockSize = statfs.getBlockSizeLong();
long availableSize = statfs.getAvailableBlocksLong() * blockSize;
long totalSize = statfs.getBlockCountLong() * blockSize;
```

2.3 Runtime Memory

Function: read the system's runtime-memory capacity.

2.3.1 Read System Memory Size

long getSystemMemory()

Returns motherboard RAM capacity in bytes. For example, 1 GB returns 1024*1024*1024.

Parameter / Return	Type	Description	Example
Return value	long	Memory size in bytes	1073741824

2.4 Backlight Control

Function: turn the LCD backlight on or off.

2.4.1 Turn the LCD Backlight On / Off

void setLcdBackLight(boolean on)

Turns the LCD backlight power and display-output signal on or off. It does not control the backlight of an external HDMI monitor or television.

Parameter / Return	Type	Description	Example
on	boolean	true: backlight on; false: backlight off	

2.5 System Time

Function: set the Android system clock.

2.5.1 Set the Android System Clock

void setSystemTime(long milliseconds1970)

Sets the Android real-time clock. The parameter is the elapsed time from 1970 to the target time. Declare the following permission and system-sign the application:

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

Parameter / Return	Type	Description	Example
milliseconds1970	long	Elapsed time from 1970 to the target time	

2.6 GPIO

Functions: configure I/O input/output and read or set low/high state.

2.6.1 Enable an I/O Port

boolean gpioEnable(int port)

Enables the specified I/O port. An I/O must be enabled before its direction or value can be configured. Use the motherboard manual to determine the port number.

This API requires root privileges.

Parameter / Return	Type	Description	Example
port	int	Motherboard GPIO port number	180
Return value	boolean	true: enabled; false: failed	

2.6.2 Set I/O Direction to Input

void setGpioInputMode(int port)

Sets the enabled port to input mode. Determine the port number from the hardware design.

This API requires root privileges.

Parameter / Return	Type	Description	Example
port	int	Motherboard GPIO port number	180

2.6.3 Set I/O Direction to Output

void setGpioOutputMode(int port)

Sets the enabled port to output mode. Determine the port number from the hardware design.

This API requires root privileges.

Parameter / Return	Type	Description	Example
port	int	Motherboard GPIO port number	180

2.6.4 Read I/O Level

int getGpioValue(int port)

Reads the level of the specified I/O. Return 0 means low and return 1 means high. Enable the port first. If no direction was set, the operation automatically selects input mode.

This API requires root privileges.

Parameter / Return	Type	Description	Example
port	int	Motherboard GPIO port number	180
Return value	int	0: low level; 1: high level	

2.6.5 Set I/O Level

void setGpioValue(int port, int value)

Sets the I/O level. value=0 outputs low; value=1 outputs high. Enable the port and set output mode first.

This API requires root privileges.

Parameter / Return	Type	Description	Example
port	int	Motherboard GPIO port number	180
value	int	0: low level; 1: high level	

2.7 Serial Port

Functions: open and close a serial port, read serial data, and write serial data.

2.7.1 Open a Serial Port

SerialPort openSerialPort(String path, int baudrate, int dataBit, int parity, int stopBits, int dataFlow)

Opens the specified serial device and returns a SerialPort object, as defined in the JAR. Returns null on failure. This interface currently applies path and baudrate; other values use defaults. The simplified overload below is recommended.

SerialPort openSerialPort(String path, int baudrate)

Opens the specified device using path and baudrate; other values use system defaults: data bits=8, parity=none, stop bits=1, flow control=none.

Parameter / Return	Type	Description	Example
path	String	Serial device path	/dev/ttyS1
baudrate	int	Baud rate	9600, 115200
dataBit	int	Data bits; currently only 8 is supported	8
parity	int	Parity; only none is supported	0
stopBits	int	Stop bits; only 1 is supported	1
dataFlow	int	Flow control; only none is supported	0
Return value	SerialPort	Custom SerialPort object	

2.7.2 Read Serial Data

Use SerialPort.getInputStream() to obtain an InputStream, then read data using the standard system API.

2.7.3 Write Serial Data

Use SerialPort.getOutputStream() to obtain an OutputStream, then write data using the standard system API.

2.7.4 Close a Serial Port

void close(SerialPort port)

Closes the specified serial-port object. Closing the port also closes any open stream interfaces.

2.8 Hardware Watchdog

Functions: enable, disable, and feed the hardware watchdog.

2.8.1 Enable the Watchdog

boolean watchDogEnable()

Enables the hardware watchdog. Once enabled, it must be fed periodically. If software stops feeding it, the watchdog timer expires and the system restarts automatically.

This API requires root privileges.

Parameter / Return	Type	Description	Example
Return value	boolean	true: enabled successfully; false: enable failed	

2.8.2 Feed the Watchdog

boolean watchDogFeed()

Feeds the watchdog. Ignore the return value unless feeding is attempted before the watchdog device is open, in which case false is returned. The timeout is typically 10-20 seconds; feed at least once every five seconds.

This API requires root privileges.

Parameter / Return	Type	Description	Example
Return value	boolean	Return value can normally be ignored	

2.8.3 Disable the Watchdog

boolean watchDogDisable()

Disables the hardware watchdog. Ignore the return value. Periodic feeding is no longer required after it is disabled.

This API requires root privileges.

Parameter / Return	Type	Description	Example
Return value	boolean	Return value can normally be ignored	

2.9 Power Control

Functions: hardware shutdown, software restart, hardware restart, and timed power-on.

2.9.1 Hardware Shutdown

`void hardShutdown()`

Performs shutdown through the hardware power-control circuit. Power-on is then possible only by remote control, reconnecting the supply, or using the power button.

This API requires root privileges.

2.9.2 Software Restart

`void softReboot()`

Performs a warm restart through the system reboot command. The CPU restarts from its initial instruction, but not every device or interface is guaranteed to be fully reset.

This API requires root privileges.

2.9.3 Hardware Restart

`void hardReboot()`

Uses the hardware power-control circuit to power-cycle the system. All supplies except 12 V input and 5 V standby are removed. Because it triggers power-off, restart is not immediate: the interval is tens of seconds and never exceeds 60 seconds.

This API requires root privileges.

2.9.4 Timed Power-On

`void alarmPoweron(int onSeconds)`

Immediately shuts down and powers on again after the specified number of seconds. The minimum is 60 seconds. Example: at 18:00, use 50400 to power on at 08:00 the next day.

This API requires root privileges.

Parameter / Return	Type	Description	Example
onSeconds	int	Seconds from the current shutdown to automatic power-on	

2.10 HDMI-IN

Functions: read HDMI input state and input resolution. HDMI-IN must be supported by the motherboard hardware.

2.10.1 Read HDMI Input State

`int getHdmiinStatus()`

Returns 1 when an HDMI input signal is present and 0 when no signal is present.

Parameter / Return	Type	Description	Example
Return value	int	1: HDMI signal present; 0: no HDMI signal	

Requires SDK 1.1.0.20170818 or later. Call only when the HDMI-IN camera preview is not open. After entering the preview flow, query Android property `getprop sys.hdmiin.status` instead.

2.10.2 Read HDMI Input Resolution

`int getHdmiinResolution()`

Returns 0 for unknown, 1 for 1080p (1920x1080), or 2 for 720p (1280x720). Only 1080p and 720p input resolutions are supported.

Parameter / Return	Type	Description	Example
Return value	int	0: unknown; 1: HDMI-IN 1080p; 2: HDMI-IN 720p	

Requires SDK 1.1.0.20170818 or later. Call only while the HDMI-IN camera preview is closed. Once the preview flow is active, query `getprop sys.hdmiin.resolution`; it returns 1 for 1080p and 2 for 720p.

End of the hardware-management section.

3 Android Management

This chapter describes software interfaces for Android-platform resources on the motherboard, including display and navigation-bar management, the status bar, applications, system information, system upgrade, and shell commands.

3.1 Display Management

Functions: read resolution, capture the screen, rotate the display, read density, and set density (restart required).

3.1.1 Read Resolution

Use the Android system API. `WindowManager.getDefaultDisplay().getRealMetrics(dm)` returns the physical screen resolution; `WindowManager.getDefaultDisplay().getMetrics(dm)` returns the resolution excluding navigation-bar height.

3.1.2 Capture the Screen

Bitmap `screenCapture()`

Captures the screen at its original dimensions and returns a Bitmap. Returns null on failure.

Parameter / Return	Type	Description	Example
Return value	Bitmap	Captured Bitmap object	

void `screenCapture(String path)`

Captures the screen at its original dimensions and saves it to the specified path.

Parameter / Return	Type	Description	Example
path	String	Output-file path	/mnt/external_sd/1.png

3.1.3 Rotate the Screen

Use `Settings.System.getInt/putInt` with `Settings.System.USER_ROTATION`. Values include `Surface.ROTATION_0`, `ROTATION_90`, `ROTATION_180`, and `ROTATION_270`.

```
Settings.System.putInt(contentResolver,
    Settings.System.USER_ROTATION, Surface.ROTATION_270);
<uses-permission android:name="android.permission.WRITE_SETTINGS" />
```

3.1.4 Read Display Density

`int getDisplayDensity()`

Reads the Android display-density value `ro.sf.lcd_density`. Typical values: 120 ldpi, 160 mdpi, 240 hdpi, 320 xhdpi, 480 xxhdpi.

Parameter / Return	Type	Description	Example
Return value	int	System display density	160

3.1.5 Set Display Density

`void setDisplayDensity(int density)`

~~Sets the display density. The system restarts automatically, and the new value becomes effective after restart.~~

This API requires root privileges.

Parameter / Return	Type	Description	Example
density	int	System display density (80-640)	160

3.2 Navigation Bar

Functions: show or hide the navigation bar, enable or disable swipe-to-show, and configure automatic hiding.

3.2.1 Show the Navigation Bar

void showNavigationBar()

Shows the Android bottom navigation bar. If swipe-to-show and timeout hiding are enabled, the bar still hides automatically after the idle timeout.

3.2.2 Hide the Navigation Bar

void hideNavigationBar()

Hides the Android bottom navigation bar.

3.2.3 Enable / Disable Swipe

void navigationBarSlideShow(boolean enable)

Enables or disables swipe-to-show. When enabled, the hidden navigation bar can be shown by swiping up from the bottom edge.

Parameter / Return	Type	Description	Example
enable	boolean	true: enable swipe; false: disable swipe	

3.2.4 Automatically Hide the Navigation Bar

void navigationBarMaxIdle(int seconds)

Sets the idle time before the navigation bar hides. If seconds > 0, the bar hides after that many idle seconds and can later be revealed by swiping. If seconds <= 0, swipe is disabled and a currently visible bar no longer hides automatically.

3.3 Status Bar

The Android status bar displays dynamic information such as network state, time, settings, messages, and notifications. This API does not dynamically hide or show it. Dedicated system builds either omit the status bar entirely or show the standard status bar. If a build shows the status bar, the navigation bar must also be visible; otherwise pull-down interaction may fail.

3.4 Application Management

Functions: read application lists (system/user/all), silently install or uninstall, and configure automatic startup.

3.4.1 Read Application List

Use the standard system API.

3.4.2 Silent Application Installation

void installApplication(String apkPath)

Installs the APK at the specified path without user interaction.

This API requires root privileges.

Parameter / Return	Type	Description	Example
apkPath	String	Path of the APK to install	

3.4.3 Automatic Application Installation

Use the following method. A progress dialog appears, but no confirmation is required.

```
Intent intent = new Intent(Intent.ACTION_VIEW);
intent.setDataAndType(Uri.parse("file:"),
    "application/vnd.android.package-archive");
intent.putExtra("IMPLUS_INSTALL", "SILENT_INSTALL");
startActivity(intent);
```

3.4.4 Silent Application Uninstallation

void uninstallApplication(String packageName)

Silently uninstalls the application identified by its actual package name, not its APK filename. For example, the package name for WeChat is `com.tencent.mm`.

This API requires root privileges.

Parameter / Return	Type	Description	Example
packageName	String	Package name of the application to uninstall	com.tencent.mm

3.4.5 Automatic Application Uninstallation

Use the following method; no confirmation is required during uninstallation.

```
Intent intent = new Intent(Intent.ACTION_DELETE);
intent.setData(Uri.parse("package:" + packageName));
intent.putExtra("IMPLUS_UNINSTALL", "SILENT_UNINSTALL");
startActivity(intent);
```

3.4.6 Automatic Application Startup

Implement a Receiver that listens for `android.intent.action.BOOT_COMPLETED` and handles startup. Add `android.permission.RECEIVE_BOOT_COMPLETED`. The default motherboard package also includes a startup utility in which third-party applications can be selected.

3.5 System Information

Functions: read the Android version, serial number, system version, kernel version, API version, Wi-Fi hardware address, and Ethernet hardware address.

3.5.1 Read Android Version

Use `android.os.Build.VERSION.RELEASE`, for example "4.4.4".

3.5.2 Read Android Serial Number

Use `android.os.Build.SERIAL`, for example "QK2141IQAK". The system generates this hardware serial number internally from the Wi-Fi MAC address. It is normally unique and can identify the hardware.

3.5.3 Read System Version

String `getSystemVersion()`

Reads the motherboard OEM Android release version, for example "2.0.0-170514-OEM". Prefer `android.os.Build.DISPLAY`.

Parameter / Return	Type	Description	Example
Return value	String	Motherboard system version	2.0.0-170514-OEM

3.5.4 Read Kernel Version

String `getKernelVersion()`

Reads the motherboard Linux kernel version, for example "3.10.96".

Parameter / Return	Type	Description	Example
Return value	String	Linux kernel version	3.10.96

3.5.5 Read API Version

String `getApiVersion()`

Reads the unified software version of the APIs described in this manual, for example "1.0.0.20170512".

Parameter / Return	Type	Description	Example
Return value	String	API version in this manual	1.0.0.20170512

3.5.6 Read Wi-Fi Hardware Address

Use `WifiManager.getConnectionInfo()` to obtain `WifiInfo`, then call `WifiInfo.getMacAddress()`.

3.5.7 Read Ethernet Hardware Address

See section 2.1.6, Read Ethernet MAC Address.

3.6 System Upgrade

Function: specify an OTA upgrade package and complete the system upgrade.

3.6.1 System OTA Upgrade

void updateSystem(String updateFilePath)

Specifies an Android OTA package, normally update.zip supplied by the motherboard vendor, then restarts into OTA upgrade mode. An Android robot and progress bar are shown. After completion, the system restarts into Android.

This API requires root privileges. Verify the upgrade file's size and contents after copying it to the target path, ensuring that the data has reached physical storage. Delete the original package after a successful upgrade if it is no longer needed.

Parameter / Return	Type	Description	Example
updateFilePath	String	Path of the Android upgrade package	

3.7 Shell Command Execution

Function: execute a specified shell command.

3.7.1 Execute a Shell Command

void suExec(String command)

Executes a Linux shell command with root privileges. Android root access is relayed through a background service, so this call is not fully blocking or synchronous.

This API requires root privileges.

For example, a file-copy command may return before copying finishes. For strictly synchronous execution, implement a method such as the following and avoid deadlocks while waiting between processes.

Parameter / Return	Type	Description	Example
command	String	Shell command to execute	chmod 0666 /dev/video0

Strictly synchronous shell-command example (not supplied as an API):

```
private static String suExecWait(String command, File workingDirectory) {
    if (command == null || (command = command.trim()).length() == 0)
        return null;
    if (workingDirectory == null)
        workingDirectory = new File("/");
    java.io.OutputStream out = null;
    java.io.InputStream in = null;
    java.io.InputStream err = null;
    try {
        Runtime runtime = Runtime.getRuntime();
        Process process = runtime.exec("su", null, workingDirectory);
        StringBuffer inString = new StringBuffer();
        StringBuffer errString = new StringBuffer();
        out = process.getOutputStream();
        out.write(command.endsWith("\n") ? command.getBytes() :
            (command + "\n").getBytes());
        out.write(new byte[]{'e','x','i','t','\n'});
        in = process.getInputStream();
        err = process.getErrorStream();
        process.waitFor(); // Blocks until the command returns
        while (in.available() > 0) inString.append((char)in.read());
        while (err.available() > 0) errString.append((char)err.read());
        return inString.toString();
    } catch (Exception ioex) {
        return null;
    } finally {
        closeStream(out);
        closeStream(in);
        closeStream(err);
    }
}
```

```
    }  
}
```

End of the Android-management section.

4 Timed Power-On/Off Broadcast Interface

Function: broadcast interface for timed power-on/off configuration. The timed power control APK must be installed.

Because Android 8.0 and later restrict static broadcasts, specify the target package by adding `intent.setPackage("com.lztek.bootmaster.poweralarm7")`.

Daily schedule - `com.lztek.tools.action.ALARM_DAILY`

onTime - Power-on time, String in HH:mm 24-hour format (for example 08:05). An empty string means no power-on time.

offTime - Power-off time, String in HH:mm 24-hour format (for example 20:30). An empty string means no power-off time.

Example - power on daily at 08:05 and power off at 20:30:

```
Intent intent = new Intent("com.lztek.tools.action.ALARM_DAILY");
intent.putExtra("onTime", "08:05");
intent.putExtra("offTime", "20:30");
intent.setPackage("com.lztek.bootmaster.poweralarm7"); // Android 8.0+
context.sendBroadcast(intent);
```

Weekly schedule - `com.lztek.tools.action.ALARM_WEEKLY`

onTime - Power-on times, String[7] ordered Sunday through Saturday. Each element uses HH:mm; an empty element means no power-on time that day.

offTime - Power-off times, String[7] ordered Sunday through Saturday. Each element uses HH:mm; an empty element means no power-off time that day.

Example - power on at specified times Monday through Thursday; power off at specified times Monday through Wednesday and Friday:

```
Intent intent = new Intent("com.lztek.tools.action.ALARM_WEEKLY");
intent.putExtra("onTime", new String[]{"", "08:05", "09:15",
    "10:00", "10:00", "", ""});
intent.putExtra("offTime", new String[]{"", "21:45", "21:05",
    "21:00", "", "22:00", ""});
intent.setPackage("com.lztek.bootmaster.poweralarm7"); // Android 8.0+
context.sendBroadcast(intent);
```

Clear timed power settings - com.lztek.tools.action.ALARM_UNSET

Example:

```
Intent intent = new Intent("com.lztek.tools.action.ALARM_UNSET");
intent.setPackage("com.lztek.bootmaster.poweralarm7"); // Android 8.0+
context.sendBroadcast(intent);
```

5 Application Startup-Management Broadcast Interface

Function: application startup management and direct-launch/keep-alive broadcasts. The application startup-management APK must be installed.

Because Android 8.0 and later restrict static broadcasts, specify the package by adding `intent.setPackage("com.lztek.bootmaster.autoboot7")`.

Configure application keep-alive - `com.lztek.tools.action.KEEPALIVE_SETUP`

`packageName` - Package name of the protected application, String.

`delaySeconds` - Seconds to wait before restarting after exit, int; default 0 starts immediately.

`foreground` - Keep the application in the foreground and reopen it if needed, boolean; default true.

Example:

```
Intent intent = new Intent("com.lztek.tools.action.KEEPALIVE_SETUP");
intent.putExtra("packageName", "xxx.xxx.xxx");
// intent.putExtra("delaySeconds", 5); // Restart 5 seconds after exit
// intent.putExtra("foreground", false); // Background allowed; reopen after process exits
intent.setPackage("com.lztek.bootmaster.autoboot7"); // Android 8.0+
context.sendBroadcast(intent);
```

Cancel application keep-alive - `com.lztek.tools.action.KEEPALIVE_UNSET`

`packageName` - Package name of the protected application, String.

Example:

```
Intent intent = new Intent("com.lztek.tools.action.KEEPALIVE_UNSET");
```

```
intent.putExtra("packageName", "xxx.xxx.xxx");
intent.setPackage("com.lztek.bootmaster.autoboot7"); // Android 8.0+
context.sendBroadcast(intent);
```

Cancel all application keep-alive rules - com.lztek.tools.action.KEEPALIVE_UNSET_ALL

Example:

```
Intent intent = new Intent("com.lztek.tools.action.KEEPALIVE_UNSET_ALL");
intent.setPackage("com.lztek.bootmaster.autoboot7"); // Android 8.0+
context.sendBroadcast(intent);
```

Configure direct launch at boot - com.lztek.tools.action.BOOT_SETUP

packageName - Package name of the application to launch directly at boot, String.

delaySeconds - Delay before direct launch, int; default 0 launches immediately after boot.

Example:

```
Intent intent = new Intent("com.lztek.tools.action.BOOT_SETUP");
intent.putExtra("packageName", "xxx.xxx.xxx");
// intent.putExtra("delaySeconds", 5); // Launch 5 seconds after boot completes
intent.setPackage("com.lztek.bootmaster.autoboot7"); // Android 8.0+
context.sendBroadcast(intent);
```

Cancel direct launch - com.lztek.tools.action.BOOT_UNSET

packageName - Package name of the application configured for direct launch, String.

Example:

```
Intent intent = new Intent("com.lztek.tools.action.BOOT_UNSET");
```

```
intent.putExtra("packageName", "xxx.xxx.xxx");  
intent.setPackage("com.lztek.bootmaster.autoboot7"); // Android 8.0+  
context.sendBroadcast(intent);
```

Cancel all direct-launch rules - com.lztek.tools.action.BOOT_UNSET_ALL

Example:

```
Intent intent = new Intent("com.lztek.tools.action.BOOT_UNSET_ALL");  
intent.setPackage("com.lztek.bootmaster.autoboot7"); // Android 8.0+  
context.sendBroadcast(intent);
```

End of the Motherboard API Programming Manual.