UniWin 系统 API 文档 V1.1.3

版本号	日期	制/修订人	制/修订记录
V1.0	2017-03-12		初始版本
V1.1	2017-08-02		1.更换休眠唤醒接口； 2.新增：定时开关机接口，通过时间设置星期接口、关闭所有外部接口设置的定时开关机； 3.新增：开关 HDMI、开关 LCD 屏；
V1.1.1	2017-09-28		1.增加设置系统时间接口； 2.增加设置系统时区接口；
V1.1.2	2018-03-19		修改“APP 中使用 GPIO 说明”
V1.1.3	2018-05-21		静默升级增加 intent.putExtra("autostart", true)
V1.1.4	2019-06-11		导航栏描述错误修改

1. 关机：3
2. 重启:3
3. 屏幕关闭进入休眠:3
4. 屏幕唤醒显示:3
5. 截屏:3
6. 隐藏导航栏:3
7. 显示导航栏:3
8. 获取系统亮度:3
9. 设置系统亮度:3
10. APP 中使用 GPIO 说明：4
11. 静默升级使用说明:4
12. 获取设备序列号:5
13. 新版定时开关机接口:5
13.1 通过日期时间设置接口（单次）5
13.2 通过时间设置星期接口（重复）5
13.3 关闭所有外部接口设置的定时开关机6
14. 获取设备型号:6
15. 获取 android 版本:6
16. 获取内核版本:6
17. 获取固件版本:7
18. 获取主副屏分辨率:7
19. 获取内存大小:8
20. 获取存储空间大小:9
21. 获取横竖屏状态:9
22. 开关 HDMI:9
23. 开关 LCD 屏:9
24. 设置系统时间:10
25. 设置系统时区:10

1. 关机：

```
Intent intent = new Intent("com.allwinner.poweroff");
sendBroadcast(intent);
```

2. 重启：

```
Intent intent = new Intent("com.allwinner.reboot");
sendBroadcast(intent);
```

3. 屏幕关闭进入休眠：

```
Intent intent = new Intent("com.allwinner.screen_off");
sendBroadcast(intent);
```

4. 屏幕唤醒显示：

```
Intent intent = new Intent("com.allwinner.screen_on");
sendBroadcast(intent);
```

5. 截屏：

```
Intent intent = new Intent("com.allwinner.screencap");
sendBroadcast(intent);
```

图片保存路径：String path = "/sdcard/Pictures/Screenshots/Screenshots_" + System.*currentTimeMillis*() + ".png";

6. 隐藏导航栏：

```
Intent intent = new Intent("com.allwinner.hide_nav_bar");
sendBroadcast(intent);
```

7. 显示导航栏：

```
Intent intent = new Intent("com.allwinner.show_nav_bar");
sendBroadcast(intent);
```

8. 获取系统亮度：

```
Settings.System.getInt(getContentResolver(),
                       Settings.System.SCREEN_BRIGHTNESS);
```

9. 设置系统亮度：

其中，需要注意的是，返回的亮度值是处于 0-255 之间的整型数值。

```
Settings.System.putInt(this.getContentResolver(),
                       Settings.System.SCREEN_BRIGHTNESS,
                       value);      //value  表示需要设置的亮度值
```

10. APP 中使用 GPIO 说明：

1）把 libs 里的 gpio.jar、arm64-v8a、armeabi、armeabi-v7a、x86 放到应用里 libs 目录；

2）引入 import com.uniwin.Gpio;

3）具体读写 IO 说明：

a.读取 IO 口状态示例：

```
int value = Gpio.readGpio('g', 13);  //gpio 读 pg13 的值，低电平返回值为 0，高电平返回值为 1
```

说明：PG13 是 M183 板子上 JP21 上标注为 IO 的 PIN

b.写 IO 口状态示例：

```
Gpio.writeGpio('h', 20, 0);  //gpio h20 写 0
```

4)libgpio_jni.so 说明：

为了更好的兼容不同平台，libs 目录下提供有多个版本的 so，如 armeabi、arm64-v8a 等；

可以针对不同的 cpu 架构选择不同的 so 版本放到 app 或者系统里；

如只想放一个版本的 so，则放 armeabi 即可；

5）预装 app 到 system 目录下：

需要预置 so 文件到系统指定目录下；

①android4.4 系统，so 预置到/system/lib/目录下；

②android5.0 或以上，so 预置到 app 安装目录下；

例如 UniwinGpioDemo，libgpio_jni.so 文件放到 system/app/UniwinGpioDemo/lib/arm/目录下；

特别说明：预置到 system 只需放一个版本的 so 文件，建议放 armeabi；

具体可参考《GPIO 使用说明 2.0》Demo

11. 静默升级使用说明：

1)首先定义 apk 的路径；

2)发送广播，名字为 android.intent.action.SILENT_INSTALL_PACKAGE，putExtra apkFilePath 为 apk 的绝对路径；

3)发送该广播后，该 apk 在后台自动安装，安装成功后界面会弹出 Toast 提示；

```
private static String apkFilePath = "/mnt/usbhost/ApiDemos.apk";
Intent intent = new Intent("android.intent.action.SILENT_INSTALL_PACKAGE");
intent.putExtra("apkFilePath",apkFilePath);
intent.putExtra("autostart", true); //安装成功后是否立刻启动，true 为启动、false 为不启动，默认为 false；
context.sendBroadcast(intent);
```

12. 获取设备序列号:

说明：每一台机器此序列号唯一，跟 cpu id 一致

```
import android.os.SystemProperties;
String serial = SystemProperties.get("ro.serialno", "1234567890ABCDEF");
```

如没法引用 SystemProperties 可以通过反射方法获取，如下：

```
public static String uwGetSerialNumber(){
    String serial = null;
    try {
        Class<?> c =Class.forName("android.os.SystemProperties");
        Method get =c.getMethod("get", String.class);
        serial = (String)get.invoke(c, "ro.serialno");
    } catch (Exception e) {
        e.printStackTrace();
    }
    return serial;
}
```

13. 新版定时开关机接口:

13.1 通过日期时间设置接口（单次）

timeonArray 及 timeoffArray 分表代表了定时开关机的年、月、日、时（24 小时制）、分;

```
Intent intent = new Intent("android.allwinner.intent.action.setpoweronoff");
int[] timeonArray = {2016,10,17,06,30};
int[] timeoffArray = {2016,10,17,23,30};
intent.putExtra("timeon", timeonArray);
intent.putExtra("timeoff", timeoffArray);
intent.putExtra("enable", true); //定时开关机
//或 intent.putExtra("enable", false); //取消定时开关机
sendBroadcast(intent);
```

13.2 通过时间设置星期接口（重复）

```
Intent intent = new Intent("android.allwinner.intent.action.setpoweronoff");
String timeon = "06:30"; //开机时间，格式为"00:00"，分别为小时和分钟。
String timeoff = "23:30"; //关机时间，格式为"00:00"，分别为小时和分钟。
int[] weekArray = {1,2,3,4,5,6,7};  //设置星期，1 为星期一，以此类推。如只设置星期一就改为{1}，星期一和星期三就改成{1,3}。此处必须填写至少 1 个。
boolean enable = true;  //enable true 为打开、false 为关闭
intent.putExtra("timeon", timeon);  //设置开机时间
intent.putExtra("timeoff", timeoff); //设置关机时间
intent.putExtra("week",weekArray);  //设置星期
intent.putExtra("enable", enable); //开关控制
sendBroadcast(intent);
```

注意：此接口只在定时开关机 v1.1.1 版本或以上支持

13.3 关闭所有外部接口设置的定时开关机

```
Intent intent = new Intent("android.allwinner.intent.action.setpoweronoff");
intent.putExtra("delete_all", true);  //关闭并删除所有外部设置的定时开关机
sendBroadcast(intent);
```

注意：此接口只在定时开关机 v1.1.0 版本或以上支持

14. 获取设备型号:

```
public String UniwinGetDeviceModel(){
    return android.os.Build.MODEL;
}
```

15. 获取 android 版本:

```
public String UniwinGetAndroidVersion(){
    return android.os.Build.VERSION.RELEASE;
}
```

16. 获取内核版本:

```
public static String UniwinGetKernelVersion() {
    String kernelVersion = "";
    InputStream inputStream = null;
    try {
        inputStream = new FileInputStream("/proc/version");
    } catch (FileNotFoundException e) {
        e.printStackTrace();
        return kernelVersion;
    }
    BufferedReader bufferedReader = new BufferedReader(new
InputStreamReader(inputStream), 8 * 1024);
    String info = "";
    String line = "";
    try {
        while ((line = bufferedReader.readLine()) != null) {
            info += line;
        }
    } catch (IOException e) {
        e.printStackTrace();
    } finally {
        try {
            bufferedReader.close();
            inputStream.close();
        } catch (IOException e) {
            e.printStackTrace();
```

```
            }
        }
        try {
            if (info != "") {
                final String keyword = "version ";
                int index = info.indexOf(keyword);
                line = info.substring(index + keyword.length());
                index = line.indexOf(" ");
                kernelVersion = line.substring(0, index);
            }
        } catch (IndexOutOfBoundsException e) {
            e.printStackTrace();
        }

        return kernelVersion;
    }
```

17. 获取固件版本:

```
    public String UniwinGetFirmwareVersion(){
        return SystemProperties.get("ro.product.firmware", "Unknown");
    }
```

如没法引用 SystemProperties 可以通过反射方法获取，如下：

```
    public static String uniwinGetFirmwareVersion(){
            String serial = null;
               try {
                   Class<?> c =Class.forName("android.os.SystemProperties");
                   Method get =c.getMethod("get", String.class);
                   serial = (String)get.invoke(c, "ro.product.firmware");
               } catch (Exception e) {
                   e.printStackTrace();
               }
            return serial;
    }
```

18. 获取主副屏分辨率:

```
    //主屏分辨率
    public static String getMainScreenSize(Context context){
        String screenSize = "";
        DisplayManager mDisplayManager = (DisplayManager) context.getSystemService(
                Context.DISPLAY_SERVICE);
        Display[] displays = mDisplayManager.getDisplays();
        if(displays.length>0 && null!=displays[0]){
```

```
            int screenWidth = screenWidth = displays[0].getWidth();
            int screenHeight = screenHeight = displays[0].getHeight();
            screenSize = screenWidth+"x"+screenHeight;
        }
        return screenSize;
    }

    //副屏分辨率
    public static String getSecondScreenSize(Context context){
        String screenSize = "";
        DisplayManager mDisplayManager = (DisplayManager) context.getSystemService(
                Context.DISPLAY_SERVICE);
        Display[] displays = mDisplayManager.getDisplays();
        if(displays.length>1 && null!=displays[1]){
        int screenWidth = screenWidth = displays[1].getWidth();
        int screenHeight = screenHeight = displays[1].getHeight();
        screenSize = screenWidth+"x"+screenHeight;
        }
        return screenSize;
    }
```

19. 获取内存大小：

```
    public long UniwinGetTotalMemory() {
        String str1 = "/proc/meminfo";
        String str2;
        String[] arrayOfString;
        long initial_memory = 0;
        try {
            FileReader localFileReader = new FileReader(str1);
            BufferedReader localBufferedReader = new BufferedReader(
                    localFileReader, 8192);
            str2 = localBufferedReader.readLine();
            arrayOfString = str2.split("\\s+");
            for (String num : arrayOfString) {
                Log.i(str2, num + "\t");
            }
            initial_memory = Integer.valueOf(arrayOfString[1]).intValue() * 1024;
            localBufferedReader.close();

        } catch (IOException e) {
        }
        return initial_memory;
    }
```

20. 获取存储空间大小:

```
public static long UniwinGetTotalInternalMemorySize() {
    File path = Environment.getDataDirectory();
    StatFs stat = new StatFs(path.getPath());
    long blockSize = stat.getBlockSize();
    long totalBlocks = stat.getBlockCount();
    return totalBlocks * blockSize;
}
```

21. 获取横竖屏状态:

```
public static String getOrientation(Activity context) {
    switch (context.getWindowManager().getDefaultDisplay().getRotation()) {
    case Surface.ROTATION_0:
        return "portrait";
    case Surface.ROTATION_90:
        return "landscape";
    case Surface.ROTATION_180:
        return "reverse portrait";
    default:
        return "reverse landscape";
        }
}
```

22. 开关 HDMI:

```
Intent intent = new Intent("com.uniwin.hdmi_hotplug");
intent.putExtra("hotplug", value);        //value =1, 开启 HDMI; value=0,关闭 HDMI; 需要保持 HDMI 显示器与板子连接的情况下才可以控制
sendBroadcast(intent);
```

23. 开关 LCD 屏:

```
Intent intent = new Intent("com.uniwin.lcdbacklight_ctrl");
intent.putExtra("lcdbacklight_on", value);    //value=1 , 开启屏幕, 并开启触摸等输入功能; value=0, 关闭屏幕, 并关闭触摸等输入功能;
sendBroadcast(intent);
```

24. 设置系统时间:

```
Intent intent = new Intent("com.uniwin.settime");
intent.putExtra("year", 1987);  //年，int 整形
intent.putExtra("month", 3);    //月，int 整形
intent.putExtra("day", 10);     //日，int 整形
intent.putExtra("hour", 18);    //小时，int 整形
intent.putExtra("minute", 30);  //分钟，int 整形
this.sendBroadcast(intent);
```

注：可以单独设置年月日，也可以单独设置小时和分钟；

25. 设置系统时区:

```
Intent intent = new Intent("com.uniwin.settimezone");
intent.putExtra("timezone", "GMT+05:00"); //格式必须为"GMT+XX:XX"
this.sendBroadcast(intent);
```