# 控制器接口文档说明(标签和灯带)Java

## 简介

系统软件接口采用SDK 包的方式提供给客户进行二次业务层的开发，目前提供支持C#和Java的SDK DLL 接口(SysControllerNewApi.dll)和调用DEMO，用户二次开发的终端软件在PC/平板终端上运行。

本文档为Java版本。

## 使用环境

### 编译环境

JDK1.8

### 控制器支持类型

TCP/IP 接口的网络通讯控制器.

每个控制器建立三个连接：

4001 端口是通道一的标签；

4002 端口是通道二的标签

4003 端口是控制控制器屏幕，声音和灯带的连接。

### 调用方式

建议创建maven项目引入sdk

mvn install:install-file -Dfile=device-sdk-1.0.jar -DgroupId=com.nbicc.liansheng -DartifactId=device-sdk -Dversion=1.0 -Dpackaging=jar

需要初始化，设置回调

setCallback

示例：

public class Test {

//实现回调接口SDKCallback，具体功能见四、标签反馈事件

static class SDKCallbackImpl implements SDKCallback {

@Override

public void connectCallback(String channelId, String ip, int port) {

log.info("connectCallback：{} {} {}", channelId, ip, port);

}

@Override

public void disconnectCallback(String channelId, String ip, int port) {

log.info("disconnectCallback：{} {} {}", channelId, ip, port);

}

@Override

public void pushButtonCallback(String channelId, byte labelAddr, LightMode lightMode, short num) {

log.info("pushButtonCallback：{} {} {} {}", channelId, labelAddr, lightMode, num);

}

@Override

public void pushButtonTAL600Callback(String channelId, byte labelAddr, LightMode lightMode, int num, LEDLightMode ledLightModel\_Special1, byte charIndex\_SpecialLED1,

LEDLightMode ledLightModel\_Special2, byte charIndex\_SpecialLED2,LEDLightMode ledLightModel\_Special3, byte charIndex\_SpecialLED3) {

log.info("pushButtonTAL600Callback：{} {} {} {} {} {} {} {} {} {}", channelId, labelAddr, lightMode, num,

ledLightModel\_Special1, charIndex\_SpecialLED1, ledLightModel\_Special2, charIndex\_SpecialLED2,

ledLightModel\_Special3, charIndex\_SpecialLED3);

}

@Override

public void batteryLowCallback(String channelId, byte labelAddr, float voltage) {

log.info("batteryLowCallback：{} {} {}", channelId, labelAddr, voltage);

}

@Override

public void getVoltageCallback(String channelId, float voltage) {

log.info("getVoltageCallback：{} {}", channelId, voltage);

}

@Override

public void setupVTHCallback(String channelId, byte code) {

log.info("setupVTHCallback：{} {}", channelId, code);

}

}

public static void main(String[] args) throws InterruptedException {

//初始化回调setCallback

LSSdkApi.setCallback(new SDKCallbackImpl());

//接口调用

LSSdkApi.startServer(8888);

String channelId = LSSdkApi.startClientSync("192.168.224.53", 4001, 3000L);

}

## 接口函数列表

### 公共接口函数

#### 开启监听（startServer）\_客户端

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **接口名** | void startServer(int port); | | | |
| **功能说明** | 监听开启运行的控制器，需控制器配置为客户端模式，可监听多台控制器 | | | |
| **参数** | **参数名** | **数据类型** | **是否必填** | **备注** |
| port | int | 是 | 监听端口 |
| **返回结果** |  | void | 否 |  |
| **Java调用** | | | | |
| **示例** | int port = 8888;  LSSdkApi.startServer(port); | | | |

#### 关闭监听（stopServer）\_客户端

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **接口名** | void stopServer(int port); | | | |
| **功能说明** | 关闭监听 | | | |
| **参数** | **参数名** | **数据类型** | **是否必填** | **备注** |
| port | int | 是 | 监听端口 |
| **返回结果** |  | void | 否 |  |
| **Java调用** | | | | |
| **示例** | int port = 8888;  LSSdkApi.stopServer(port); | | | |

#### 控制器连接（startClientSync）\_服务端

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **接口名** | String startClientSync(String host, Integer port, Long timeOut) | | | |
| **功能说明** | 连接控制器，需控制器配置为服务端模式，调用此函数需控制器先开启 | | | |
| **参数** | **参数名** | **数据类型** | **是否必填** | **备注** |
| host | String | 是 | 控制器IP地址 |
| port | Integer | 是 | 监听端口 |
| timeOut | Long | 否 | 超时，默认3000毫秒 |
| **返回结果** | channelId | String | 是 | 返回控制器ID |
| **Java调用** | | | | |
| **示例** | String channelId = LSSdkApi.startClientSync("192.168.224.53", 4001, 3000L); | | | |

#### 控制器批量连接（startClient）\_服务端

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **接口名** | void startClient(String host, int... ports) | | | |
| **功能说明** | 连接控制器，需控制器配置为服务端模式，调用此函数需控制器先开启 | | | |
| **参数** | **参数名** | **数据类型** | **是否必填** | **备注** |
| host | String | 是 | 控制器IP地址 |
| ports | int | 是 | 可变参数，监听端口数组 |
| **返回结果** |  | void | 否 |  |
| **Java调用** | | | | |
| **示例** | LSSdkApi.startClient("192.168.224.53", 4001, 4002);  //连接结果需等待回调，见“连接回调” | | | |

#### 断开连接（disconnectClient）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **接口名** | boolean disconnectClient(String channelId) | | | |
| **功能说明** | 断开与控制器的连接 | | | |
| **参数** | **参数名** | **数据类型** | **是否必填** | **备注** |
| channelId | String | 是 | 连接id |
| **返回结果** | result | boolean | 是 | 返回是否成功 |
| **Java调用** | | | | |
| **示例** | String channelId = LSSdkApi.startClientSync("192.168.224.53", 4001, 3000L);  boolean result = LSSdkApi.disconnectClient(channelId); | | | |

### 标签接口函数

#### 标签亮灭灯（4位）updateTagScreen

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **接口名** | boolean updateTagScreen(String channelId, byte labelAddr, LightMode lightMode, int num, Long timeout) | | | |
| **功能说明** | 控制指定的标签地址亮灭灯和显示数量 | | | |
| **参数** | **参数名** | **数据类型** | **是否必填** | **备注** |
| channelId | String | 是 | 连接获取的channelId |
| labelAddr | byte | 是 | 发送命令的标签的地址（1-255） |
| lightMode | LightMode | 是 | 标签显示的颜色(见标签和塔灯显示颜色列表) |
| num | int | 是 | 标签显示的数据  (调用此方法时，6位标签只能显示右边3位数据，如需显示左边3位，需调用UpdateTagScreen\_TAL600方法) |
| timeout | Long | 否 | 超时，默认3000毫秒 |
| **返回结果** | bResult | boolean | 是 | 结果（true-成功，false-失败） |
| **Java调用** | | | | |
| **示例** | String channelId = LSSdkApi.startClientSync("192.168.224.53", 4001, 3000L);  LSSdkApi.updateTagScreen(channelId, (byte) 2, LightMode.RED, 999, 3000L); | | | |

#### 标签亮灭灯（包括高位显示）updateTagScreen\_TAL600

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **接口名** | boolean updateTagScreen\_TAL600(String channelId, byte labelAddr, LightMode lightMode, int num, LEDLightMode ledLightModel\_Special1, byte charIndex\_SpecialLED1,  LEDLightMode ledLightModel\_Special2, byte charIndex\_SpecialLED2,  LEDLightMode ledLightModel\_Special3, byte charIndex\_SpecialLED3,  int timeout ) | | | |
| **功能说明** | 控制指定的标签亮灭灯(包括高3位显示字符内容、颜色) | | | |
| **参数** | **参数名** | **数据类型** | **是否必填** | **备注** |
| channelId | String | 是 | 连接获取的channelId |
| labelAddr | byte | 是 | 发送命令的标签的地址（1-255） |
| lightMode | LightMode | 是 | 标签显示的颜色(见标签和塔灯显示颜色列表) |
| num | int | 是 | 标签右边3位显示的数据 |
| ledLightModel\_Special1 | LEDLightMode | 是 | 标签显示的颜色(第4位)  (见标签和塔灯显示颜色列表) |
| charIndex\_SpecialLED1 | byte | 是 | 标签显示的字符(第4位) |
| ledLightModel\_Special2 | LEDLightMode | 是 | 标签显示的颜色(第5位)  (见标签和塔灯显示颜色列表 |
| charIndex\_SpecialLED2 | byte | 是 | 标签显示的字符(第5位) |
| ledLightModel\_Special3 | LEDLightMode | 是 | 标签显示的颜色(第6位)  (见标签和塔灯显示颜色列表 |
| charIndex\_SpecialLED3 | byte | 是 | 标签显示的字符(第6位) |
| timeout | int | 否 | 超时，默认3000毫秒 |
| **返回结果** | bResult | boolean | 是 | 结果（true-成功，false-失败） |
| **Java调用** | | | | |
| **示例** | String channelId = LSSdkApi.startClientSync("192.168.224.53", 4001, 3000L);  boolean b1 = LSSdkApi.updateTagScreen\_TAL600(channelId, (byte) 1,  LightMode.RED, 999, LEDLightMode.YELLOW,  (byte) 7, LEDLightMode.RED, (byte) 5, LEDLightMode.RED, (byte) 7, null); | | | |

#### 分配地址（getTagPhysicalAddr）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **接口名** | void getTagPhysicalAddr(string channelId, byte start) | | | |
| **功能说明** | 分配标签地址**[慎用]**  当调用此接口后，标签上会显示一排下划线”\_”,代表进入分配地址状态，通过依次手动拍标签灯来分配地址，标签上会显示分配好的地址 | | | |
| **参数** | **参数名** | **数据类型** | **是否必填** | **备注** |
| channelId | string | 是 | 连接获取的channelId |
| start | byte | 是 | 分配起始地址 |
| **返回结果** | bResult | boolean | 是 | 结果（true-成功，false-失败） |
| **Java调用** | | | | |
| **示例** | String channelId = LSSdkApi.startClientSync("192.168.224.53", 4001, 3000L);  LSSdkApi.getTagPhysicalAddr(channelId,(byte)1); | | | |

#### 标签自检(setSelfCheck)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **接口名** | byte[] setSelfCheck(String channelId,Long timeout) | | | |
| **功能说明** | 标签自检 | | | |
| **参数** | **参数名** | **数据类型** | **是否必填** | **备注** |
| channelId | String | 是 | 连接获取的channelId |
| timeout | Long | 否 | 超时，默认3000毫秒 |
| **返回结果** | tags | byte[] | 是 | 返回标签地址数组  数组中下标地址和标签地址一一对应 |
| **Java调用** | | | | |
| **示例** | logText.append("自检结果"+LSSdkApi.setSelfCheck(channelId,timeOut).length+"\n");  //页面打印自检标签个数 | | | |

## 标签反馈事件

### 连接回调（connectCallback）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **触发事件** | public void channelActive(ChannelHandlerContext ctx) | | | | |
| **功能说明** | 连接成功时触发 | | | | |
| **返回参数** | **参数名** | | **数据类型** | **是否必填** | **备注** |
| channelId | | String | 是 | 控制器连接ID |
| ip | | String | 是 | 连接上的控制器IP |
| port | | int | 是 | 连接上的控制器端口 |
| **Java调用** | | | | | |
| **示例** | **调用** | @Override  public void connectCallback(String channelId, String ip, int port) {  log.info("connectCallback：{} {} {}", channelId, ip, port);  } | | | |

### 离线回调（disconnectCallback）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **触发事件** | public void channelInactive(ChannelHandlerContext ctx) | | | | |
| **功能说明** | 连接断开时触发 | | | | |
| **返回参数** | **参数名** | | **数据类型** | **是否必填** | **备注** |
| channelId | | String | 是 | 离线的控制器连接ID |
| ip | | String | 是 | 离线的控制器IP |
| port | | int | 是 | 离线的控制器端口 |
| **Java调用** | | | | | |
| **示例** | **调用** | @Override  public void disconnectCallback(String channelId, String ip, int port) {  log.info("disconnectCallback：{} {} {}", channelId, ip, port);  } | | | |

### 拍灯4位标签（pushButtonCallback）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **触发事件** | void solvePushButtonCommand(byte[] bytes, ChannelHandlerContext ctx) | | | | |
| **功能说明** | 拍灯事件，手动短拍标签灯触发的回调函数(4位标签和不高位显示的6位标签) | | | | |
| **返回参数** | **参数名** | | **数据类型** | **是否必填** | **备注** |
| channelId | | String | 是 | 控制器连接ID |
| labelAddr | | byte | 是 | 标签编号 |
| lightMode | | LightMode | 是 | 所拍灯色(见标签和塔灯显示颜色列表) |
| num | | short | 是 | 所拍标签显示数据 |
| **Java调用** | | | | | |
| **示例** | **调用** | @Override  public void pushButtonCallback(String channelId, byte labelAddr, LightMode lightMode, short num) {  log.info("pushButtonCallback：{} {} {} {}", channelId, labelAddr, lightMode, num);  } | | | |

### 拍灯6位标签(pushButtonTAL600Callback)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **触发事件** | void solvePushButtonTAL600Command(byte[] bytes, ChannelHandlerContext ctx) | | | | |
| **功能说明** | 拍灯事件，手动短拍标签灯，触发的回调函数(包含高位显示的6位标签) | | | | |
| **返回参数** | **参数名** | | **数据类型** | **是否必填** | **备注** |
| channelId | | String | 是 | 控制器连接ID |
| labelAddr | | byte | 是 | 所拍标签地址 |
| lightMode | | LightMode | 是 | 所拍灯色(见标签和塔灯显示颜色列表) |
| num | | int | 是 | 所拍标签右边3位显示数据 |
| ledLightModel\_Special1 | | LEDLightMode | 是 | 标签显示的颜色(第4位) |
| charIndex\_SpecialLED1 | | byte | 是 | 标签显示的字符(第4位) |
| ledLightModel\_Special2 | | LEDLightMode | 是 | 标签显示的颜色(第5位) |
| charIndex\_SpecialLED2 | | byte | 是 | 标签显示的字符(第5位) |
| ledLightModel\_Special3 | | LEDLightMode | 是 | 标签显示的颜色(第6位) |
| charIndex\_SpecialLED3 | | byte | 是 | 标签显示的字符(第6位) |
| **Java调用** | | | | | |
| **示例** | **调用** | @Override  public void pushButtonTAL600Callback(String channelId, byte labelAddr, LightMode lightMode, int num,LEDLightMode ledLightModel\_Special1, byte charIndex\_SpecialLED1,LEDLightMode ledLightModel\_Special2, byte charIndex\_SpecialLED2,LEDLightMode ledLightModel\_Special3, byte charIndex\_SpecialLED3) {  log.info("pushButtonTAL600Callback：{} {} {} {} {} {} {} {} {} {}", channelId, labelAddr, lightMode, num,  ledLightModel\_Special1, charIndex\_SpecialLED1, ledLightModel\_Special2, charIndex\_SpecialLED2,  ledLightModel\_Special3, charIndex\_SpecialLED3);  } | | | |

## 高3位可显示字符列表

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **码** | **字符** | **码** | **字符** | **码** | **字符** | **码** | **字符** |
| 0 | 0 | 8 | 8 | 16 | U | 24 | S |
| 1 | 1 | 9 | 9 | 17 | P | 25 | n |
| 2 | 2 | 10 | A | 18 | - | 26 |  |
| 3 | 3 | 11 | b | 19 | null | 27 |  |
| 4 | 4 | 12 | C | 20 | \_. | 28 |  |
| 5 | 5 | 13 | d | 21 | \_ | 29 |  |
| 6 | 6 | 14 | E | 22 | L | 30 |  |
| 7 | 7 | 15 | F | 23 | H | 31 |  |

## 标签和塔灯显示颜色列表

|  |  |  |
| --- | --- | --- |
| **内容** | **值** | **枚举定义** |
| 标签亮灯颜色 | 白灯：0x01 白灯闪烁：0x11  红灯：0x02 红灯闪烁：0x12  黄灯：0x03 黄灯闪烁：0x13  绿灯：0x04 绿灯闪烁：0x14  灭灯：0x05 亮塔灯: 0x24  其中数量：0(不显示) | public enum LightMode {  WHITE("白色", 0x01),  RED("红色", 0x02),  YELLOW("黄色", 0x03),  GREEN("绿色", 0x04),  TUNEOFF("关闭", 0x05),  WHITEFLICKER("白色闪烁", 0x11),  REDFLICKER("红色闪烁", 0x12),  YELLOWFLICKER("黄色闪烁", 0x13),  GREENFLICKER("绿色闪烁", 0x14),  LIGHTBELT("灯带", 0x24);} |
| 特殊字符颜色 | 红色：64  黄色：192  绿色：128  不显示：0x00 | public enum LEDLightMode {  RED("红色", 64),  YELLOW("黄色", 192),  GREEN("绿色", 128),  TRUNOFF("不显示", 0x00);} |