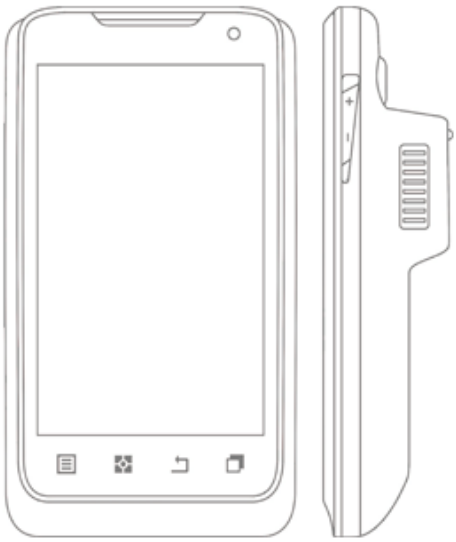

FS789 一维影像手持终端

开发手册

FS789 Barcode Scan Mobile Terminal Development Manual Android 4.0

专为无线移动查询、采购、补货、收货、批发、盘点设计

硬件参数

联想 A789 设备参数
手机类型：3G 手机，智能手机
主屏尺寸：4 英寸
触摸屏：电容屏，多点触控
主屏材质：TFT
主屏分辨率：800x480 像素
主屏色彩：1600 万色
网络类型：双卡双模
网络模式：GSM，WCDMA
数据业务：GPRS，EDGE，HSDPA
支持频段：2G：GSM 850/900/1800/1900
3G：WCDMA 900/2100MHz
操作系统：Android OS 4.0
核心数：双核
CPU 型号：联发科 MT6577
CPU 频率：1024MHz
GPU 型号：Imagination PowerVR SGX531
RAM 容量：512MB
ROM 容量：4GB
存储卡：MicroSD 卡，支持 App2SD 功能
扩展容量：32GB
电池容量：2000mAh
键盘类型：虚拟键盘
机身颜色：黑色

联想 A789 数据功能
WLAN 功能：WIFI
数据接口：Micro USB v2.0
耳机插孔：3.5mm
蓝牙传输 WAP 浏览器 WWW 浏览器

联想 A789 商务功能
办公工具：TXT，Quick Office，
Adobe PDF，电子邮件 飞行模式
世界时间

条码扫描参数
每秒扫描次数：200 次/秒
解析精度：标准 4Mil 条码(0.1mm)
读取范围：35mm 至 300mm
读取方式：一维红光影像

联想 A789 基本功能
输入法：手写，笔画，拼音输入法
输入方式：手写
通话记录：已接+已拨+未接电话
通讯录：名片式存储 短信(SMS) 彩信(MMS)
免提通话 录音功能 情景模式 待机图片 主题菜单 来电铃声识别 来电图片识别 日历功能 闹钟功能 计算器

联想 A789 产品特性
GPS 导航：内置 GPS，支持 A-GPS
重力感应器 光线传感器 距离传感器

联想 A789 拍照功能
摄像头：内置
摄像头类型：双摄像头（前后）
摄像头像素：前 30 万像素，后 500 万像素
传感器类型：CMOS
自动对焦：支持
图像尺寸：最大支持 2592×1944
视频拍摄：支持

联想 A789 娱乐功能
视频播放：支持 3GP/MP4 等格式
音频播放：支持 MIDI/MP3/AAC 等格式
铃音描述：和弦，支持 MP3/MIDI 等格式

开发语言

Eclipse
Java

条码读取原理

FS789 采用蓝牙传输方式将条码发送至手机，用户开发程序主要是蓝牙编程。

1. 搜索
2. 连接
3. 配对
4. 建立通讯流
5. 读取条码

蓝牙开发

打开蓝牙

Android 蓝牙编程需要引入库和在 AndroidManifest.xml 中加入操作权限，有了这两个前提，开发人员就可以通过 enable()函数打开手机上的蓝牙适配器。

库:

```
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.bluetooth.BluetoothSocket;
```

权限:

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

代码:

```
myBluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
if (myBluetoothAdapter != null) {
    if (!myBluetoothAdapter.isEnabled()) {
        myBluetoothAdapter.enable(); //打开蓝牙适配器
     }
}
```

搜索和配对

FS789 条码读头有一个蓝牙 MAC 地址，搜索就是为了得到这个 MAC 地址，只有得到这个 MAC 地址，今后就可以通过程序直接连接这个 MAC 地址读取条码。

1. 建立一个事件信息处理

```
private BroadcastReceiver mReceiver = new BroadcastReceiver() {
   @Override
   public void onReceive(Context context, Intent intent) {
      String action = intent.getAction();
      // 获得已经搜索到的蓝牙设备
      if (action.equals(BluetoothDevice.ACTION_FOUND))
      {
         BluetoothDevice deviceone = intent.getParcelableExtra(BluetoothDevice.EXTRA_DEVICE);
         //将搜索到的蓝牙设备信息存放入列表中
         HashMap<String, Object> listItemmap = new HashMap<String, Object>();
         //蓝牙条码名称,FSScan
         listItemmap.put("macname", Deviceone.getName().toString());
         //蓝牙地址,如 1D:C4:FE:D0:F7:16
         listItemmap.put("macaddr", deviceone.getAddress().toString());
         //加入 BlueScanlistItem 列表
         BlueScanlistItem.add(listItemmap);
      } else if (action.equals(BluetoothAdapter.ACTION_DISCOVERY_FINISHED)) {
         //搜索完成,告诉 handler 让系统弹出一个选择窗口
         setProgressBarIndeterminateVisibility(false);
         Message message = new Message();
         message.what = 1;
         handler.sendMessage(message);
      }
   }
};
```

2. 在 public void onCreate(Bundle saveInstanceState)事件中注册事件

```
BlueScanlistItem = new ArrayList<HashMap<String, Object>>();
//注册搜索蓝牙条码设备时的事件
IntentFilter mFilter = new IntentFilter(BluetoothDevice.ACTION_FOUND);
registerReceiver(mReceiver, mFilter);
//注册搜索完时的 receiver
mFilter = new IntentFilter(BluetoothAdapter.ACTION_DISCOVERY_FINISHED);
registerReceiver(mReceiver, mFilter);
```

3. 开始搜索

```
if(myBluetoothAdapter!=null)
{
    if (myBluetoothAdapter.isDiscovering())
    {
        myBluetoothAdapter.cancelDiscovery();
    }
    //弹出一个等待窗口,预计花 20 秒钟搜索
    mProgressDlg = ProgressDialog.show(FS789DemoActivity.this,"FS789 条码扫描演示", "扫描
    蓝牙设备中,请稍候...", false, false);
    BlueScanlistItem.clear();
    myBluetoothAdapter.startDiscovery(); //开始搜索蓝牙
}
```

4. 在弹出的窗口中选择蓝牙条码设备，发现未配对的话就弹出配对密码输入窗口，一般蓝牙条码名称为 FSScan，配对密码为 1234

```
BluetoothDevice btDev =
myBluetoothAdapter.getRemoteDevice('蓝牙地址,如 1D:C4:FE:D0:F7:16');
int connectState = btDev.getBondState();
if(connectState == BluetoothDevice.BOND_NONE){
    //配对 1234
    try {
        Method createBondMethod = BluetoothDevice.class.getMethod("createBond");
        createBondMethod.invoke(btDev);
    } catch (Exception e) {
    }
}
```

通过 4 个步骤完成了蓝牙条码的搜索和配对，用户应写这个 MAC 地址记录保存至数据库，下次连接使用蓝牙条码时，可以直接用这个 MAC 地址而无需再次搜索和配对。

连接蓝牙条码设备和读取

蓝牙通讯一般情况下是非常稳定的，但无线通讯必定会出现断开连接的情况(如其中一方断电)，而导致蓝牙条码扫描读取不到，这时就需要重新连接蓝牙，故我们可以专门写一个函数连接或重覆连接。

```
public void BlueScanConnect()
{
  // P_FScanMac  全局变量保存了蓝牙条码 MAC 地址
  if(P_FScanMac.length()==0){
    MyInfo.setText("蓝牙条码未设置,请先设置!");
    return;
```

```
}
if (myBluetoothAdapter == null)return;

BluetoothDevice btDev;
try{
   btDev = myBluetoothAdapter.getRemoteDevice(P_FScanMac);
}catch(Exception ec)
{
   Toast.makeText(getApplicationContext(),"错误:" + ec.getMessage(),
   Toast.LENGTH_LONG).show();
   MyInfo.setText("状态:[" + ec.getMessage() + "]");
   return;
}

try {
   if(socket!=null){
        mainThreadFlag = false;
        inStream.close();
        socket.close();
        SystemClock.sleep(500);
   }
} catch (Exception e1x) {
}

try{
   socket = btDev.createRfcommSocketToServiceRecord(uuid);
}catch(Exception e)
{
   Toast.makeText(getApplicationContext(),
   "建立蓝牙模块连接操作失败!" +
   e.getMessage().toString(),Toast.LENGTH_LONG).show();
   MyInfo.setText("建立蓝牙模块连接操作失败!");
   return;
}
try {
   socket.connect();
} catch (IOException e1) {
   Toast.makeText(getApplicationContext(),
   "建立连接蓝牙模块操作失败!" +
   e1.getMessage().toString(),Toast.LENGTH_LONG).show();
   MyInfo.setText("建立蓝牙模块连接操作失败!");
   return;
}
Toast.makeText(getApplicationContext(),
```

```
        "连接蓝牙条码成功!",Toast.LENGTH_LONG).show();
        MyInfo.setText("连接蓝牙条码成功!");
        try {
           inStream = socket.getInputStream(); //建立一个读取蓝牙信息的流
        } catch (Exception e) {
        try{
           socket.close();
        }catch (Exception e13) {
        }
        Toast.makeText(getApplicationContext(),
        "建立接收缓冲池失败!" + e.getMessage().toString(),Toast.LENGTH_LONG).show();
        return;
        }
        //连接蓝牙成功后,开启一个线程,轮循读取蓝牙传过来的信息
        mainThreadFlag = true;
        new BlueProgressThread().start();
}

private class BlueProgressThread extends Thread {
        public void run() {
                byte[] buffer = new byte[2048];
                int bytes;
                String a;
                String b;
                a = "";
                long Sc,u;
                Sc = SystemClock.uptimeMillis();

                while (mainThreadFlag) {
                    SystemClock.sleep(10);
                    if(socket!=null)
                    {
                        try {
                            bytes = 0;
                            if (inStream.available() != 0)
                            {
                               bytes = inStream.read(buffer);
                            }
                            if(bytes==0)continue;
                            if(P_BarcodeUse==0) //条码不接收状态
                            {
                                a = "";
                                continue;
                            }
```

```
                        //b = new String(buffer,0,bytes,"GB2312");
                        b = new String(buffer,0,bytes);
                        a = a + b;
                        if(a.indexOf("\r\n")>=0){
                            u = SystemClock.uptimeMillis();

                            if((u - Sc) < 500)
                            {
                                //保证半秒钟只读取一次
                                a = "";
                                continue;
                            }
                            Sc = SystemClock.uptimeMillis();
                            P_TxmS = a.substring(0,a.indexOf("\r\n"));
                            a = "";
                            Message message = new Message();
                            message.what = 999; //告诉 handler，条码读到了
                            handler.sendMessage(message);
                        }

                    } catch (Exception fe) {
                        Message message = new Message();
                        message.what = 4000;//告诉 handler，有错误
                        handler.sendMessage(message);
                        return;

                    }

                }// if(socket!=null)

            }//while


        } //run()

}

Handler handler = new Handler() {
    public void handleMessage(Message msg) {
        switch (msg.what) {
            case 1:
            //蓝牙搜索完毕,弹出一个窗口让用户选择
            mProgressDlg.dismiss();
            BlueScanlistItemAdapter.notifyDataSetChanged();
```

```
            bluedlg.show();
            break;
        case 999:
            //接收到蓝牙传过来的条码字符串了 P_TxmS
            P_TxmString = P_TxmS;
            //处理条码
            break;
        case 4000:
            P_BarcodeUse=1; //可以接收蓝牙条码
            Toast.makeText(getApplicationContext(),
            "蓝牙条码断开!",Toast.LENGTH_LONG).show();
            MyInfo.setText("状态:蓝牙条码断开!");
            break;
    }
    super.handleMessage(msg);
  }
};
```

断开连接

```
    try {
      if(socket!=null){
          mainThreadFlag = false;
          inStream.close();
          socket.close();
          SystemClock.sleep(500);
      }
    } catch (Exception e1x) {
    }
```

上例的代码是关闭蓝牙流和断开蓝牙连接，在退出系统时则需要做得更多。

```
protected void onDestroy() {
            super.onDestroy();
            //退出系统时,要撤消注册
            unregisterReceiver(mReceiver);
            // 关闭蓝牙连接和线程
            mainThreadFlag = false;
            if(socket!=null)
            {
                try {
                    socket.close();
                } catch (IOException e)
```

```
                {
                }
                SystemClock.sleep(500);
            }
            System.exit(0);
}
```

开发技巧

FS789 手持终端条码读取基本就是蓝牙的编程操作，读出条码时会向 handler 发送消息，开发人员要处理好这个消息非常关键，输入光标在哪个控件要判断好才赋值 setText()，开发人员可以建一个全局变量，如 public int P_InputNo = 0，这个值记录当前条码应赋值处，在处理 handler 消息时就可以明确的向对应控件赋值条码信息或处理其它操作了，如

0- 不作任何处理，如在等待界面或异步查询时
1- 主界面条码输入框
2- 价格输入框
3- 数量输入框
4- 弹出的查找窗口输入框
5- 等等

用户需要用好两个关键事件

1. 焦点处理
   public void setOnFocusChangeListener(View v, Boolean hasFocus){};
2. 弹出窗口用户按返回键关闭时对 P_InputNo 变量的赋值，用于 AlertDialog
   setOnCancelListener(new OnCancelListener()

源码例子

系统提供了一个例子，用户可以下载细看，源码中例举了蓝牙条码的几个重要操作案例。
Http://www.Fs-789.com/download/fs789demo.zip