# 终端软件通信机制

本系统采用了三种通信方式与其他设备通信，分别是：NET、CAN、SCI。为了屏蔽底层驱动细节，便于应用层使用，这里对三种不同的通信方式定制统一通信接口和格式。本文主要就通信接口的定制做详细介绍，暂不涉及底层驱动细节。

## 1 通信接口及使用方法

接口文件：CommunicationInterface.java

接口示意图如下：



**图1-1 通信接口示意图**

### 1.1 打开通信线路

void open();

调用此函数用于建立终端与各模块之间的通信连接。在此之后，设备与终端才能开始交换数据。

### 1.2 发送数据

void send(final byte[] data, int cmd);

调用此函数向设备发送数据。其中data是待发送的数据，用字节数组保存；cmd是根据业务逻辑自定义的操作码。

**注意：调用send之前需要先调用open并成功打开通信连接，否则会发送失败。**

### 1.3 关闭通信线路

void close();

调用此函数用来关闭已打开的通信线路，通常在长时间不用以及关闭系统之前调用。

## 2 通信子线程管理以及数据的接收

由于主界面需要完成图片、工作状态的显示以及参数设置等工作，为保证主界面操作流畅，这里将通信任务放到后台进行，对每一个通信方式创建一个独立子线程用以处理相应通信线路的所有数据传递。

在以上接口之中，并没有定义接收数据的函数。因为数据的接收状态通常是不可预知的，所以这里在通信子线程中使用消息队列的来完成数据的接收。

数据接收主要步骤如下：

1. 当主线程调用open()函数的时候，为相应通信线路创建独立子线程，并传入主线程类的context（上下文对象）以及handler（线程处理对象）；
2. 在新线程中轮询通信端口，检查是否有数据输入；
3. 当检查到有数据输入时，子线程创建一个消息队列，并将接收到的数据放入该消息队列，通过handler对象通知主线程接收数据；
4. 主线程接收到handler的消息通知，按顺序从消息队列中取出数据进行解析，完成数据接收。

## 3 各个通信线程类的实现

这里需要为3个通信模块（NET、SCI、CAN）分别创建各自的线程类。在Java中，线程类需要继承自“Thread”，然后由于API封装的需要，还要实现CommunicationInterface接口。下面分别是这3个模块的实现机制。

## 3.1 网络通信（NET）

网络通信协议暂定为TCP/IP协议，数据传输使用字节数组。

### 3.1.1 打开网络设备

在用户需要连接网络设备的时候，首先需要调用open()函数和网络设备建立连接。那么在open()接口函数中我们需要启动子线程，并在线程中创建socket对象，建立TCP/IP连接。请求连接的过程如下所示：



**图3-1 打开NET过程示意图**

### 3.1.2 通过网络发送数据

在网络连接建立好了之后，网络通信子线程NetThread就进入等待状态，此时线程会不断检查是否有数据输入，一旦捕捉到数据输入，则对接收包的帧头进行解析，解析过程如下：

1. 首先获取Socket输入流；
2. 从输入流中读取四个字节的枕头标识（如：0xAAAD），如果错误则舍弃；
3. 如果帧头标识正确，则按照协议对整个包的数据进行校验，校验错误则舍弃；
4. 校验成功之后，获取一个整形数据作为操作码；
5. 最后建立一个字节数组存储包中余下的所有数据。

### 3.1.3 网络数据接收

应用程序可以通过send()函数发送数据。为了避免与接收线程冲突，Send()函数会首先建立一个发送子线程，然后从Socket中获取输出流。

发送之前需要打包好数据包，打包过程如下：

1. 首先将整形操作符cmd写入数据输出流。
2. 然后将字符数据data[]数组写入数据输出流。
3. 最后用ByteArrayOutputStream类对所有数据进行包装，统一转换成字节数组。

这样数据包就组装完毕，接着按照通信协议加上帧头，最后将整个字节数据数据包发送出去。

### 3.1.4 关闭网络设备

当长时间不需要数据传输时，和关闭系统之前，建议调用close()函数关闭网络连接。另外，在网络连接过程中，如果出现任何故障，系统也会自动关闭网络连接。

## 3.2 串口通信（SCI）

串口可调参数：波特率、校验位。

其他参数：8位数据位，1位停止位

### 3.2.1 打开串口设备

当用户需要连接串口设备的时候，可以调用串口设备的open()函数。此时系统会首先检查串口设备是否存在，并且是否有使用权限。然后启动串口子线程SciThread，在线程中按照用户设定的波特率和校验位打开串口设备。由于在整个系统中只存在一个串口，故这里串口类使用的单例模式。当open()调用之后，系统会启动串口线程，同时返回单例串口类。详细过程如下：



**图3-2 串口设备打开过程**

### 3.2.2 通过串口发送数据

串口发送数据是通过调用串口类中的send()函数完成的，函数完成签名如下：

send(final byte[] data, int cmd);

byte[]表示待发送的字节数组，cmd是附带的操作码。为了不与接收串口数据的接收产生冲突，当应用程序请求发送数据的时候，串口类会单独开启一个子线程，并在子线程里调用底层的write函数对SCI设备文件写入比特流。在发送完毕后，子线程会自动关闭。

### 3.2.3 串口数据接收

当串口open()函数成功返回时，标志着串口设备已经打开。此时串口子线程进入对SCI设备的轮询状态，通过select()函数判断SCI设备输入流是否有可用数据产生，一旦有数据可用，则进入接收程序。整个过程如下：

1. 系统轮询SCI输入端口。
2. 有数据可用则调用read()函数从SCI设备读取字节数组到接收缓存revBuf[]中。
3. 接收完毕后通过系统的消息队列发送给主线程。
4. 主线程从消息队列中取出数据完成串口数据接收。



**图3-3 串口接收流程**

### 3.2.4 关闭串口设备

当不需要使用串口设备的时候，应用程序可以通过SCI类中的close()函数关闭串口。close()函数首先会修改串口状态标志，结束串口接收的轮询操作。随后调用串口设备底层的close()函数关闭串口设备。至此，串口设备关闭完成。