**智能终端通信协议**

马超

2015-08-26

# 1 网络通信机制

网络通信协议暂定为TCP/IP协议，数据传输使用字节数组。SC280采用小端格式，而本系统采用大端格式，所以在通信过程中需要注意数据存储格式。

## 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1 打开NET过程示意图**

## 1.2 数据包的产生

数据包里包含以下几个部分：

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| magic | version | type | block | length | offset | minid | data[1] |

其中前7个部分是数据帧头，而后面的是用户数据。这里主要需要控制改变的是type、block、minid以及用户数据这几个部分。所以在生成数据包的过程中，仅对这几项做控制。

首先我们根据协议针对不同的命令编写相应的数据包；然后通过工厂方法模式根据用户需求生成相应数据包。

## 1.3 数据发送

应用程序可以通过调用数据包的send()函数发送数据。这里将send()函数封装在了数据包对象当中。调用send()之后，程序从Socket中获取输出流。然后向网络输出流中写入我们要发送的数据。

发送数据过程如下：

1. 通过用户操作判断需要执行的命令；
2. 根据协议找到需要生成的数据包；
3. 将数据包的帧头写入输出缓冲区；
4. 如果此数据包含有数据区，则在帧头后面紧跟用户数据；
5. 最后将输出缓冲区的数据统一转换成字节数组，并通过网络发送。

这样数据包的发送就完成了。

## 1.4 网络数据接收

和数据发送一样，这里将接收函数recvDataPack封装到数据包对象当中。当我们需要去接收一个数据包的时候，直接调用数据包的recvDataPack()函数。数据的接收会以阻塞的方式进行，设置网络超时时间为5秒。

数据的接收及解析过程如下：

1. 调用recvDataPack()函数后，获取Socket输入流；
2. 设置一个2M的接收缓存，用于接收网络上的数据；
3. 从接收缓存中查找4字节的帧头标识（0x695A695A），如果未找到则舍弃；
4. 如果找到帧头，则记录下位置，并判断数据帧头是否接收完整。如完整，则对帧头进行解析，反之则舍弃此包。
5. 在数据帧头解析完毕后，可获得数据长度字段，按照此字段将整包数据打包存入接收数据包对象中完成接收。
6. 将数据包的用户数据部分传给上层，供应用层使用。

**图2 网络接收流程图**

## 1.5 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。

# 2 网络通信连接

1）采用TCP/IP通信方式，SC280作为客户机，ARM作为服务器。

2） SC280以UDP广播帧通过6019端口发送请求ARM服务器IP地址帧，发送内容为字符串”Get Server IP”，并在监听6018端口等待IP地址数据。

3）ARM在6019端口接收到请求数据后，通过6018端口以字符串形式将IP发送给广播给SC280。如，”115.156.211.147”。

4）默认工作在6020端口上。

# 3 网络通信数据帧

## 3.1 命令说明

本协议规定了ARM与SC280的网络通信协议，该协议只规定了传输数据内容及意义，而不关心具体的通信方式。无论是SC280作为TCP/IP客户机、服务器、或是以UDP进行通信，该协议均适用。

## 3.2 命令帧格式

本协议规定了通信帧头格式，如下表：

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| magic | version | type | block | length | offset | minid | data[1] |
| Uint32 | Uint32 | Uint32 | Uint32 | Uint32 | Uint32 | Uint32 | Uint8 |

magic：帧头标识，用于判断一个命令帧的起点。值为：0x695a695a。

version：版本标识，用于区分不同版本的通信协议，因为目前没有第二版的协议，此值未使用，但为了方便以后拓展，默认写入0。

type：用于区分不同的帧处理方式，**只对SC280有效**，ARM端没有这些约束

值为0，该帧无需返回值；

值为1，该帧需要返回值，但超过block ms仍没能返回则不返回；

值为2，该帧必须返回值，处理直到将值返回。

block：设置等待返回值超时，当type为1时有效，单位ms。

length：该帧总长，单位为Byte。

offset：帧头长度，不包括data[1]部分，即该值为28。

minid：命令号，区分不同的命令。

data[1]：有效数据区，相当于一个指针，后面可填充任意长度有效数据。

## 3.3 命令集

|  |  |  |  |
| --- | --- | --- | --- |
| 命令标识  minid | 子命令枚举量 | 功能 | 命令方向 |
| 1 | MSG\_NET\_GET\_VIDEO | 获取图像 | ARM🡨🡪DSP |
| 2 | MSG\_NET\_GENERAL | 设置工作参数 | ARM🡪DSP |
| 9 | MSG\_NET\_TEXTINFO | 显示信息 | DSP🡪ARM |
| 10 | MSG\_NET\_LINKINFO | 连接状态 | DSP🡪ARM |
| 11 | MSG\_NET\_NORMAL | 选择算法 | ARM🡪DSP |
| 12 | MSG\_NET\_FLASH | NAND操作 | ARM🡨🡪DSP |
| 14 | MSG\_NET\_STATE | 获取温度 | ARM🡨🡪DSP |
| 15 | MSG\_NET\_SEND\_IMAGE | 发送图像 | ARM🡪DSP |
| 16 | MSG\_NET\_GET\_RAW | 获取raw图像 | ARM🡨🡪DSP |
| 18 | MSG\_NET\_CONFSAVE | 保存配置参数 | ARM🡪DSP |
| 19 | MSG\_NET\_SETNET | 设置网络参数 | ARM🡪DSP |
| 20 | MSG\_NET\_FACTRESET | 恢复出厂设置 | ARM🡪DSP |
| 21 | MSG\_NET\_GET\_PARAM | 获取工作参数 | ARM🡨🡪DSP |
| 22 | MSG\_NET\_SAVE\_VIDEO | 保存图像 | DSP🡪ARM |
| 25 | MSG\_NET\_HELP\_ALG\_CMD | 获取算法命令 | ARM🡪DSP |
| 26 | MSG\_NET\_GET\_JSON | 获取工作参数2 | ARM🡪DSP |
| 27 | MSG\_NET\_SET\_JSON | 设置工作参数 | DSP🡪ARM |
| 28 | MSG\_NET\_RESULT | 返回检测结果 | DSP🡪ARM |
| 29 | MSG\_NET\_GET\_THUMB | 获取缩略图 | ARM🡨🡪DSP |
| 200 | MSG\_NET\_ALGRESULT | 返回算法结果 | DSP🡪ARM |

# 4 网络通信示例

## 4.1 通过网络获取图片

### 1）发送数据包

|  |  |  |
| --- | --- | --- |
| 字段 | 值 | 备注 |
| type | 1 | 需要返回 |
| block | 5000 | 超时5000ms |
| data | 无 | 请求帧，不需要数据 |

无数据段，长度为28字节。

### 2）接收数据包

|  |  |  |
| --- | --- | --- |
| 字段 | 值 | 备注 |
| data | len (4B) | 图像大小，单位：字节 |
| width (4B) | 图像宽度 |
| height (4B) | 图像高度 |
| bpb (4B) | 每个像素占字节数 |
| reserve (84B) | 保留 |
| image | 图像数据 |

## 4.2 通过网络获取温度

### 1）发送数据包

|  |  |  |
| --- | --- | --- |
| 字段 | 值 | 备注 |
| type | 1 | 需要返回 |
| block | 50000 |  |
| data | 1 (4B) | ADT75\_GET\_TEMPVALUE，获取温度子命令 |

数据段长度为4个字节，表示温度子命令。整个包长度为32字节。

### 2）接收数据包

|  |  |  |
| --- | --- | --- |
| 字段 | 值 | 备注 |
| data1 | 1 (4B) | ADT75\_GET\_TEMPVALUE子命令 |
| interger (4B) | 温度的整数部分，若35.6度，则为35 |
| float (4B) | 温度的小数部分，若为35.6度，则为6 |
| data2 | 0 (4B) | 读取温度失败 |

## 4.3 通过网络获取相机配置参数

### 1）通信格式

由于相机参数项比较多，为了方便通信双方解析以及数据的维护，这里采用Json数据格式传输。此命令通常在开机的时候使用，用来获取相机设备参数。

相机配置参数总共有八项内容，针对这八个内容定义了如下Json对象：

net、hecc、uart、sensor、mode、ad9849、version、trigger。

Json文件内容如下：

**{**http://tool.oschina.net/tools/json_format/Expanded.gif

**"net"**: **{**http://tool.oschina.net/tools/json_format/Expanded.gif

**"port"**: 6020**,**

**"work\_mode"**: 2**,**

**"ip\_address"**: **[**http://tool.oschina.net/tools/json_format/Expanded.gif

115**,**

156**,**

211**,**

2

**],**

**"remote\_ip"**: **[**http://tool.oschina.net/tools/json_format/Expanded.gif

115**,**

156**,**

211**,**

1

**],**

**"mac\_address"**: **[**http://tool.oschina.net/tools/json_format/Expanded.gif

0**,**

30**,**

140**,**

128**,**

0**,**

0

**],**

**"gateway"**: **[**http://tool.oschina.net/tools/json_format/Expanded.gif

115**,**

156**,**

211**,**

254

**],**

**"ip\_mask"**: **[**http://tool.oschina.net/tools/json_format/Expanded.gif

255**,**

255**,**

255**,**

0

**],**

**"dns"**: **[**http://tool.oschina.net/tools/json_format/Expanded.gif

0**,**

0**,**

0**,**

0

**]**

**},**

**"hecc"**: **{**http://tool.oschina.net/tools/json_format/Expanded.gif

**"baudRate"**: 1**,**

**"work\_mode"**: 0

**},**

**"uart"**: **{**http://tool.oschina.net/tools/json_format/Expanded.gif

**"baudRate"**: 1**,**

**"work\_mode"**: 129

**},**

**"sensor"**: **{**http://tool.oschina.net/tools/json_format/Expanded.gif

**"width\_max"**: 659**,**

**"height\_max"**: 494**,**

**"width\_input"**: 640**,**

**"height\_input"**: 480**,**

**"startPixel\_width"**: 0**,**

**"startPixel\_height"**: 0**,**

**"sensor\_number"**: 2**,**

**"isColour"**: 0**,**

**"bitPixel"**: 12

**},**

**"mode"**: **{**http://tool.oschina.net/tools/json_format/Expanded.gif

**"expoTime"**: 860**,**

**"trigger"**: 0**,**

**"algorithm"**: 0**,**

**"width\_crt"**: 752**,**

**"height\_crt"**: 560**,**

**"work\_mode"**: 2**,**

**"bitType"**: 16

**},**

**"ad9849"**: **{**http://tool.oschina.net/tools/json_format/Expanded.gif

**"vga"**: **[**http://tool.oschina.net/tools/json_format/Expanded.gif

0**,**

1

**],**

**"pxga"**: **[**http://tool.oschina.net/tools/json_format/Expanded.gif

0**,**

0**,**

0**,**

0

**],**

**"hxdrv"**: **[**http://tool.oschina.net/tools/json_format/Expanded.gif

3**,**

3**,**

3**,**

3

**],**

**"rgdrv"**: 3**,**

**"shp"**: 36**,**

**"shd"**: 0**,**

**"hpl"**: 0**,**

**"hnl"**: 32**,**

**"rgpl"**: 0**,**

**"rgnl"**: 84

**},**

**"version"**: **{**http://tool.oschina.net/tools/json_format/Expanded.gif

**"id"**: **[**http://tool.oschina.net/tools/json_format/Expanded.gif

0**,**

0**,**

0**,**

4

**],**

**"version"**: 566**,**

**"write\_time"**: **[**http://tool.oschina.net/tools/json_format/Expanded.gif

20**,**

15**,**

11**,**

30

**]**

**},**

**"trigger"**: **{**http://tool.oschina.net/tools/json_format/Expanded.gif

**"trigDelay"**: 2000**,**

**"partDelay"**: 4000**,**

**"velocity"**: 400**,**

**"departWide"**: 100**,**

**"expLead"**: 100

**}**

**}**

### 2）发送数据包

|  |  |  |
| --- | --- | --- |
| 字段 | 值 | 备注 |
| type | 1 | 需要返回 |
| block | 5000 | 超时5000ms |
| data | 无 | 请求帧，不需要数据 |

数据包一共28字节，无需数据位。

### 3）接收数据包

|  |  |  |
| --- | --- | --- |
| 字段 | 值 | 备注 |
| type | 0 | 不需要返回 |
| block | 0 | 无超时 |
| data | n | 包含所有参数的Json文件 |

数据包一共28+n字节。其中n为ARM终端接收到的Json格式的参数，由相机发送的参数多少决定。

## 4.4 通过网络向相机发送配置参数

### 1）通信格式

此命令用来配置相机设备参数，由ARM终端主动发起并且不需要返回值。

为了支持任意局部参数的发送，同时参数数量比较大，这里同样采用Json文件的传输格式。Json文件示意如下：

**"trigger"**: **{**http://tool.oschina.net/tools/json_format/Expanded.gif

**"trigDelay"**: 2000**,**

**"partDelay"**: 4000**,**

**"velocity"**: 400**,**

**"departWide"**: 100**,**

**"expLead"**: 100

}

以上文件是完整的用于发送触发参数的，它支持任意数量的参数缺省配置，即需要配置那些就发送哪些，可以大大增强可维护性。

### 2）发送数据包

|  |  |  |
| --- | --- | --- |
| 字段 | 值 | 备注 |
| type | 0 | 需要返回 |
| block | 0 | 5000ms延时 |
| data | n | 需要发送的设备参数 |

数据包一共28+n个字节长度。其中n表示用户要发送的以Json文件表示的设备参数。