协议宏：

typedef enum

{

SOCKET\_DATA\_DATA\_LEN\_BIT\_START = 0,

SOCKET\_DATA\_DATA\_LEN\_BIT\_DATA\_LEN = 4,

SOCKET\_DATA\_DATA\_LEN\_BIT\_SOURCE\_LEN = 8,

SOCKET\_DATA\_DATA\_LEN\_BIT\_DEST\_LEN = 12,

SOCKET\_DATA\_DATA\_LEN\_BIT\_VALUE1\_LEN = 16,

SOCKET\_DATA\_DATA\_LEN\_BIT\_VALUE2\_LEN = 20,

} SOCKET\_DATA\_DATA\_LEN\_BIT;

typedef enum

{

EN\_SYSTEM\_INFORMATION\_TYPE\_NONE = 0x0,

EN\_SYSTEM\_INFORMATION\_TYPE\_CONFIG,

EN\_SYSTEM\_INFORMATION\_TYPE\_WIRTE\_CHANNELS,

EN\_SYSTEM\_INFORMATION\_TYPE\_REQUEST\_APPLY,

EN\_SYSTEM\_INFORMATION\_TYPE\_KEY\_CODE,

EN\_SYSTEM\_INFORMATION\_TYPE\_AREA\_CODE,

EN\_SYSTEM\_INFORMATION\_TYPE\_RFID\_CODE,

EN\_SYSTEM\_INFORMATION\_TYPE\_DISPLAYS\_DEV,

EN\_SYSTEM\_INFORMATION\_TYPE\_VIDEO\_H264 = 0x10,

EN\_SYSTEM\_INFORMATION\_TYPE\_VIDEO\_YV12 = 0x11,

EN\_SYSTEM\_INFORMATION\_TYPE\_VIDEO\_RGB32 = 0x12,

EN\_SYSTEM\_INFORMATION\_TYPE\_RFID\_COMM\_EPC\_AUTO = 0x97,

EN\_SYSTEM\_INFORMATION\_TYPE\_RFID\_COMM\_EPC\_AUTO\_STOP = 0x98,

EN\_SYSTEM\_INFORMATION\_TYPE\_MAX,

} EN\_SYSTEM\_INFORMATION\_TYPE;

#define SOCKET\_DATA\_CONNECT\_FLAG 24

1. TCP 连接，建立基本的TCP SOCK连接到服务器.

TCP协议格式：

|  |
| --- |
| unsigned char \* pvData = (unsigned char\* )malloc(sBuffLen+SOCKET\_DATA\_CONNECT\_FLAG\*2);  if(NULL == pvData)  {  return wmFALSE;  }  memset(pvData,0x0,sizeof(pvData));  \*reinterpret\_cast< unsigned long \*>(pvData + SOCKET\_DATA\_DATA\_LEN\_BIT\_START) = htonl (sBuffLen);  \*reinterpret\_cast< unsigned long \*>(pvData + SOCKET\_DATA\_DATA\_LEN\_BIT\_DATA\_LEN) = htonl (iType);  \*reinterpret\_cast< unsigned long \*>(pvData + SOCKET\_DATA\_DATA\_LEN\_BIT\_SOURCE\_LEN) = htonl (iSourceID);  \*reinterpret\_cast< unsigned long \*>(pvData + SOCKET\_DATA\_DATA\_LEN\_BIT\_DEST\_LEN) = htonl (iDest);  \*reinterpret\_cast< unsigned long \*>(pvData + SOCKET\_DATA\_DATA\_LEN\_BIT\_VALUE1\_LEN) = htonl (iValue1);  \*reinterpret\_cast< unsigned long \*>(pvData + SOCKET\_DATA\_DATA\_LEN\_BIT\_VALUE2\_LEN) = htonl (iValue2);  memcpy (pvData+SOCKET\_DATA\_CONNECT\_FLAG, pBuff, sBuffLen);  // TCP Send…..  free(pvData);  pvData = NULL;  // sBuffLen 是 pBuff的长度.  // iType是 发生命令类型  // iSourceID非必须，可以直接输入0  // iDest 输入对应RFID的ID  // iValue1 备用参数  // iValue2 备用参数  // pBuff对应需要传输的具体数据  EN\_SYSTEM\_INFORMATION\_TYPE\_NONE类型服务器数没处理的，  可以使用这个类型来做定时发送活动的信息来判断网络连接和网速. |

1. 注册到服务器

|  |
| --- |
| iType = EN\_SYSTEM\_INFORMATION\_TYPE\_REQUEST\_APPLY  sBuffLen = strlen("{\"type\":\"login\",\"loginname\":\"sony520\",\"password\":\"sony520\",\"status\":null}");  pBuff = "{\"type\":\"login\",\"loginname\":\"sony520\",\"password\":\"sony520\",\"status\":null}";  其他位都设置为0  其中：  loginname和password 是你自定义的用户名和密码  type 为login为注册，loginout为注销 |

1. 发送指令

|  |
| --- |
| 开始扫描：  iType = EN\_SYSTEM\_INFORMATION\_TYPE\_RFID\_COMM\_EPC\_AUTO  iDest 输入对应RFID的ID, 现有对应的RFID数据ID为：55660  iDest = 55660  sBuffLen = strlen("EPC\_AUTO");  pBuff = "EPC\_AUTO"  其他位都设置为0  结束扫描和清楚结果：  iType = EN\_SYSTEM\_INFORMATION\_TYPE\_RFID\_COMM\_EPC\_AUTO\_STOP  iDest 输入对应RFID的ID, 现有对应的RFID数据ID为：55660  iDest = 55660  sBuffLen = strlen("EPC\_AUTO\_STOP");  pBuff = " EPC\_AUTO\_STOP"  其他位都设置为0 |

1. 发送图片数据

|  |
| --- |
| iType = EN\_SYSTEM\_INFORMATION\_TYPE\_VIDEO\_H264  发送单帧H264或者X264数据  iType = EN\_SYSTEM\_INFORMATION\_TYPE\_VIDEO\_YV12  发送YV12数据  iType = EN\_SYSTEM\_INFORMATION\_TYPE\_VIDEO\_RGB32  发送RGB24/32数据  iValue1 = 图片宽度  iValue2 = 图片长度  iDest 输入对应RFID的ID, 现有对应的RFID数据ID为：55660  iDest = 55660  sBuffLen = 数据长度;  pBuff = 数据 |

1. 运行方法：
2. 打开Linux服务器，

cd /home/C++/rfidServer/Linux

./ServerInterface

1. Android显示器打开APP，RFID
2. 打开RFID硬件设备
3. 客户的连接到TCP服务器
4. 客户的注册到TCP服务器
5. 客户端发生扫描指令和传输头像到TCP服务器
6. 客户端停止扫描

因为有X64、i386和win/linux/androiud/Iphone的各种类型需要实装，所以最好能直接自己封装TCP协议.

假如不方便，我可以提供生成各种类型的DLL和SO文件，来代替TCP协议进行通信.