



API process of sending program via LAN

1. Binding callback function and then settle DLL return message data in this function.

CallBack(ProcessCallBackMessage);

int WINAPI ProcessCallBackMessage (int Param1, char * Param2);

parameter: Param1 is the type of message

Param2 is the pointer of message data buffer structure

```
#define MAX_CALLBACK_NET_PACKET_LENGTH 2000//alahover 20120719
```

```
typedef struct _STRUCT_CALLBACK_NET_PORT_DATA_PACKET
```

```
{
```

```
    char cIp[20];
```

```
    int iPort;
```

```
    unsigned char pDataBuffer[MAX_CALLBACK_NET_PACKET_LENGTH];
```

```
}STRUCT_CALLBACK_NET_PORT_DATA_PACKET;
```

2. call DLL_Net_Init initialization function

DLL_Net_Init();

3. call DLL_Net_GetCardNumberInLan to get the card in the LAN

```
extern "C" __declspec(dllexport) int __stdcall DLL_Net_GetCardNumberInLan(char *cLocalIp);
```

parameter : cLocalIp: PC's LAN IP; if this PC has multi network card or IP addresses in different network segment, could call this DLL repeated to get all controllers in different network segment.

Return value: check controller numbers

4. call DLL_Net_GetCardIpFromList to get the serial number and IP address of the controller that has been detected already

```
extern "C" __declspec(dllexport) void __stdcall DLL_Net_GetCardIpFromList(int index, char *ip, char *id);
```

parameter: index: serial number of the controller that has been detected already, ip: to return the controller IP address, id: to return controller's serial number

NOTE: example: int itemNumber=DLL_Net_GetCardNumberInLan(tempLocalIp);// get controller numbers in advance

```
//get IP and ID information per each controller according to the numbers
```

```
for(int i=0;i<itemNumber;i++)
```

```
{
```

```
    char ip[20]={0};
```

```
    char id[20]={0};
```

```
    MACRO_IS_ENABLE_QIUT_FUNC(m_CanQuitFlag, return, ienableFlag)
```

```
    DLL_Net_GetCardIpFromList(i, ip, id); //ip, id need to distribute the memory firstly, for saving the controller IP
```

and ID information.

```
}
```

- 5 call DLL_Make_Contab_File to produce negotiation table file

```
extern "C" __declspec(dllexport) int __stdcall DLL_Make_Contab_File(char *cPanthName, char *cFileName)
```

parameter: cPanthName: storage directory for program material ,

for example, if project name is "d:\\project\\test.xml", then directory of program material will be "d:\\project\\test"

cFileName: should be "contab.txt";



6 call DLL_Net_GetSystemStatus Get controller status can received program or not.

```
extern "C" __declspec(dllexport) int __stdcall DLL_Net_GetSystemStatus(char *ip,int iPort,unsigned char ucCtrl2,MY_SOCKET mySocket)
```

parameter: ip: controller IP address; iPort:31299 ucCtrl2: 0x80

Callback function gets Param1=1062,this means controller enable receive program, then could do 7th operation.

7 call DLL_Net_ReadyToSendProgram controller enters the status of receiving programs

```
extern "C" __declspec(dllexport) int __stdcall DLL_Net_ReadyToSendProgram(char *ip/*IP address*/,int iPort,char *cPathName/*directory and name of program*/,int iMainVer/*version number*/,int iChildVer/*date version */,MY_SOCKET mySocket);
```

parameter: ip: controller IP address; iPort:31299, cPathName: path file name of the project file; iMainVer: 9; iChildVer: 0x20100915; mySocket:NULL;

callback function gets Param1=1028,this means controller is in the status of receiving programs, then could do 8th operation.

8 Call DLL_Net_SendProgramToPanel controller starting to receiving programs

```
extern "C" __declspec(dllexport) void __stdcall DLL_Net_SendProgramToPanel(char *ip/*IP address */,int iPort,char *cPathName/*project file path and name */,MY_SOCKET mySocket);
```

parameter: ip: controller IP address; iPort:31299; cPathName: path file name of project file; mySocket:NULL;

callback function gets Param1=1032, this means the receiving programs speed;

Param2 is the structure pointer of STRUCT_CALLBACK_NET_PORT_DATA_PACKET

```
STRUCT_CALLBACK_NET_PORT_DATA_PACKET *pSt=(STRUCT_GPRS_CALLBACK_DATA_PACKET *)Param2;
```

```
int iPrecentValue=((int *)&pSt->pDataBuffer[0]); //process value, 0~100
```

callback function gets Param1=1031, this means receiving program finished;

NOTE: advices: start threading to send program, send message to the main software interface in callback function; after main software gets the message will draw the process; to avoid the block of main software interface.