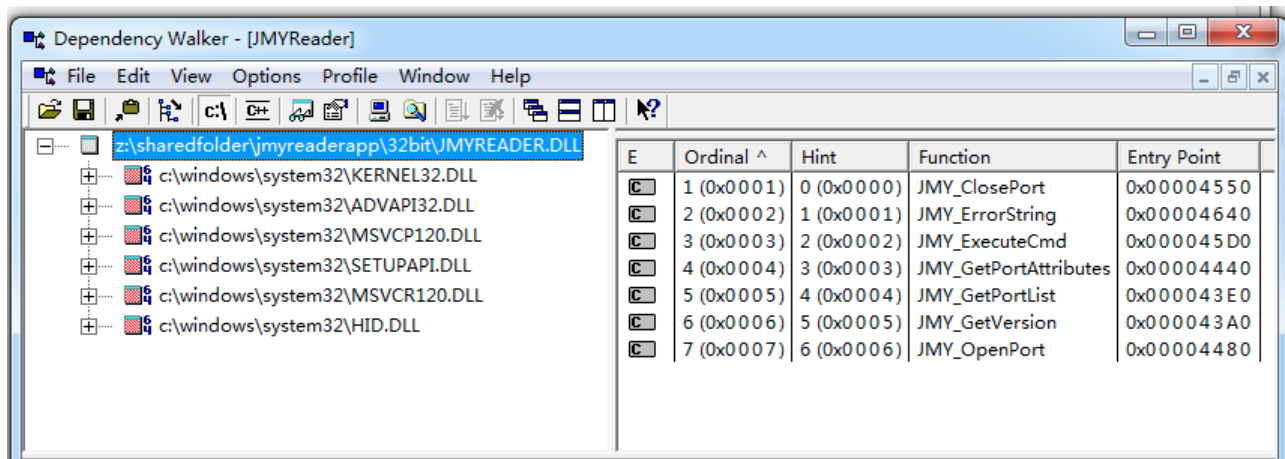


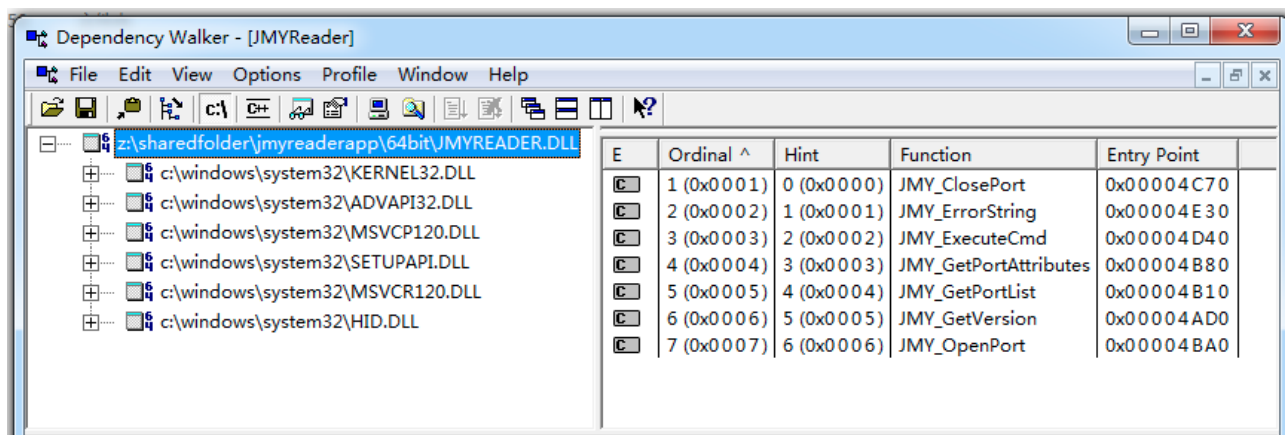
## JMYReader.DLL Instructions

导出函数共 7 个（64 位版本与 32 位版本相同）：

There are 7 exported functions (the 64-bit version is the same as the 32-bit version):



The same as the 32-bit version



函数说明(Function description):

### 1, 获取动态库版本(Get dynamic library version)

```
int __stdcall JMY_GetVersion(unsigned char *verInfo);
```

verInfo 需要足够长的空间，目前直接返回字符串 "JmyReaderDLL(01.24)"

(verInfo needs enough space, and currently returns the string "JmyReaderDLL(01.24)" directly)

### 2, 获取可用端口列表(Get a list of available ports)

```
int __stdcall JMY_GetPortList(OUT int* numOfCom, OUT int* numOfUsbHid);
```

如果要使用 UsbHid 接口类型的读卡器，必须先调用此函数，每次重新插拔都需要调用此函数，此函数内部枚举可用的串口列表，以及 UsbHid 列表；动态库，需要通过枚举 UsbHid 设备保存设备的路径，然后才能通过 JMY\_OpenPort 函数打开 UsbHid 类型的读卡器。

(If you want to use the card reader of the UsbHid interface type, you must call this function first, and you need to call this function every time you re-plug and unplug. This function enumerates the list of available serial ports and the list of UsbHid; dynamic library, you need to enumerate the

UsbHid list. The device saves the path of the device before the UsbHid type card reader can be opened through the JMY\_OpenPort function.)

### 3, 获取端口属性, 端口号和友好名称(Get port properties, port number and friendly name)

```
int __stdcall JMY_GetPortAttributes(int idx, int* portNum, char* friendName);
```

该函数一般用于具有 UI 的程序, 需要显示可用端口列表时使用, 不需要显示则不需要使用该函数;

(This function is generally used for programs with UI. It is used when the list of available ports needs to be displayed, and it is not necessary to use this function if it does not need to be displayed);

idx 参数为获取所需端口的索引号, 该值如果是串口, 则是从 0 到 numOfCom-1, 如果是 UsbHid 则是 100 到 100+numOfUsbHid-1;

(The idx parameter is the index number of the required port. If the value is a serial port, it is from 0 to numOfCom-1, and if it is UsbHid, it is 100 to 100+numOfUsbHid-1);

### 4 获取错误代码的字符串(Get a string of error codes)

```
void __stdcall JMY_ErrorString(int err, char*errmsg);
```

该函数也是用于具有 UI 程序的显示, 可以不需要使用该函数;

(This function is also used for the display with UI program, you don't need to use this function);

```
void __stdcall JMY_ErrorString(int err, char*errmsg){  
  
    static std::map<int, std::string> errs = {  
        { 0, "ERR_SUCCEED" },  
        { -1, "ERR_PARAM" },  
        { -2, "ERR_WTIMEOUT" },  
        { -3, "ERR_RTIMEOUT" },  
        { -4, "ERR_OPENREADER" },  
        { -5, "ERR_DATAOUT" },  
        { -6, "ERR_DATAIN" },  
        { -7, "ERR_CRCERR" },  
        { -8, "ERR_CMDFAIL" }  
    };  
    if (errs.count(err) > 0){  
        strcpy(errmsg, errs[err].c_str());  
    }  
    else strcpy(errmsg, "ERR_UNKNOWN");  
}
```

### 5, 打开端口(open port)

```
int __stdcall JMY_OpenPort(int nPort, int nBaud);
```

nPort 为端口号，如果是串口则是实际的端口号，如果是 UsbHid 则，首先需要调用 JMY\_GetPortList 函数，根据获得的 numOfUsbHid 的数量，第一个，端口号为 100，第二个端口号为 101，依次类推；

(nPort is the port number. If it is a serial port, it is the actual port number. If it is UsbHid, you need to call the JMY\_GetPortList function first. According to the number of numOfUsbHid obtained, the first port number is 100, and the second port number is 101. And so on);

nBaud, 如果为串口则为下列值之一 { 9600, 19200, 38400, 57600, 115200 };

(nBaud, one of the following values if serial port { 9600, 19200, 38400, 57600, 115200 });)

如果为 UsbHid, 把 nBaud 设置为 MagicNumber 表示要求内部使用

SetOutputReport/GetInputReport API, 否则使用 ReadFile/WriteFile Api, 两者对读卡器的数据传输实现方式有不同的要求; MagicNumber = 0x20220529;

(f it is UsbHid, setting nBaud to MagicNumber means that SetOutputReport/GetInputReport API is required to be used internally, otherwise ReadFile/WriteFile Api is used, and the two have different requirements for the data transmission implementation of the card reader; MagicNumber = 0x20220529;)

#### 6, 关闭端口(close port)

int \_\_stdcall JMY\_ClosePort(int nPort);

关闭端口, nPort 应为打开端口所使用的端口号;

(Close the port, nPort should be the port number used to open the port)

#### 7, 执行指令(execute instruction)

int \_\_stdcall JMY\_ExecuteCmd(int nPort, unsigned short Addr, unsigned char Cmd, unsigned char \*Data, int\* DataLen, int Timeout);

nPort 为打开的端口号(nPort is the open port number)

Addr 将该值设置为大于 0xFF 的值, 例如 0x100, 函数内部使用 JCP04 协议封装数据包, 该值的范围如果为 0x00 ~ 0xFF, 则函数内部使用 JCP05 协议封装数据包。

(Addr sets the value to a value greater than 0xFF, such as 0x100, the function uses the JCP04 protocol to encapsulate the data packet internally. If the value is in the range of 0x00 to 0xFF, the function uses the JCP05 protocol to encapsulate the data packet internally.)

Cmd 为命令码, 根据金木雨的读卡器手册设置(Cmd is the command code, set according to the card reader manual of Jinmuyu)

Data 为数据指针, 是输入数据, 也是输出数据的 Buffer, 所以, 需要足够的空间, 一般为 512 字节(Data is the data pointer, which is the input data and the Buffer of the output data.

Therefore, enough space is required, generally 512 bytes.)

DataLen 表示 Data 的长度, 读卡器返回的数据长度也将赋值给该变量;

(DataLen represents the length of Data, and the data length returned by the card reader will also be assigned to this variable);

Timeout 为需要等待读卡器数据的最长等待时长, 毫秒。

(Timeout is the longest waiting time for card reader data, in milliseconds.)

关于返回值, 错误值 ERR\_CRCERR 和 ERR\_CMDFAIL 表示 IO 正确, 但是数据检查错误, CRCERR 表示校验字节不符合计算要求(除 CRC 字节外的所有字节的 Xor 值), CMDFAIL 表示读卡器执行

命令出错！ OPENREADER 表示打开设备句柄发生错误。 DATAOUT 表示写数据到设备句柄发生超时以外的其他错误， DATAIN 表示从设备读数据时，发生了超时以外的其他错误。 TIMEOUT 表示读写数据超时！

(Regarding the return value, the error values ERR\_CRCERR and ERR\_CMDFAIL indicate that the IO is correct, but the data check is wrong, CRCERR indicates that the check byte does not meet the calculation requirements (the Xor value of all bytes except the CRC byte), and CMDFAIL indicates that the card reader executes the command Error! OPENREADER indicates that there was an error opening the device handle. DATAOUT indicates that other errors other than timeout occurred when writing data to the device handle, and DATAIN indicates that other errors other than timeout occurred when reading data from the device. TIMEOUT means read and write data timeout)!

建议发送读卡器有可能很长时间执行的命令之前，先执行一个执行较短时长的指令（0015000001 读一字节 Flash 内容），用来明确通信是正常，以避免不必要的等待。

(It is recommended to execute a command with a short duration (0015000001 to read one byte of Flash content) before sending a command that may be executed by the card reader for a long time to make sure that the communication is normal to avoid unnecessary waiting).

使用 JMYReader.dll 不需要头文件，不需要 lib 文件，使用以下代码（需 VS2013 以上，支持 C++11 标准）即可。

(Using JMYReader.dll does not require header files or lib files, just use the following code (requires VS2013 or above, supports C++11 standard)).

```
#include <Windows.h>
#include <functional>
#include <string>

//JMYReader Dll Declare
/*
int __stdcall *JMY_GetVersion(OUT unsigned char * verInfo);
int __stdcall *JMY_GetPortList(OUT int* numOfCom, OUT int* numOfUsbHid);
int __stdcall *JMY_GetPortAttributes(int idx, int* portNum, char* friendName);
int __stdcall *JMY_OpenPort(int nPort, int nBaud);
int __stdcall *JMY_ClosePort(int nPort);
int __stdcall *JMY_ExecuteCmd(int nport, unsigned short Addr, unsigned char Cmd, unsigned char* Data,
int* DataLen, int timeout);
void __stdcall *JMY_ErrorString(int err, char*errmsg);*/

const int ERR_SUCCEED = 0;
const int ERR_PARAM = -1;
const int ERR_WTIMEOUT = -2;
const int ERR_RTIMEOUT = -3;
const int ERR_OPENREADER = -4;
const int ERR_DATAOUT = -5;
const int ERR_DATAIN = -6;
const int ERR_CRCERR = -7;
const int ERR_CMDFAIL = -8;
```

```
static HMODULE hJMYReader = LoadLibrary(L"JMYReader.dll");
```

```
template <typename T>
std::function<T> GetFunc(const std::string &funcName)
{
    FARPROC funAddress = GetProcAddress(hJMYReader, funcName.c_str());
    if (funAddress == NULL)
    {
        CString szMsg;
        CString szfunc = CString(funcName.c_str());
        szMsg.Format(L"Load JMYReader.DLL failed! func<%s> cann't be load!", szfunc);
        AfxMessageBox(szMsg);
        exit(-1);
    }
    return std::function<T>((T*)(funAddress));
}
```

```
#define F(funcName,typedefunc) static auto funcName = GetFunc<typedefunc>(#funcName)
```

```
F(JMY_GetVersion, int __stdcall (unsigned char* verInfo));
F(JMY_GetPortList, int __stdcall (int* numOfCom, int* numOfUsbHid));
F(JMY_GetPortAttributes, int __stdcall (int idx, int* portNum, char* friendName));
F(JMY_OpenPort, int __stdcall (int nport, int nBaud));
F(JMY_ClosePort, int __stdcall (int nport));
F(JMY_ExecuteCmd, int __stdcall (int nport, USHORT Addr, BYTE Cmd, BYTE* Data, int* DataLen, int
timeout));
F(JMY_ErrorString, void __stdcall (int err, char* errMsg));
```

就可以在程序中调用相应的函数(You can call the corresponding function in the program).