NiMotionModbusSDK 使用说明书
NiMotionModbusSDK instruction manual

版 本 号：C
Version No.: C

北京立迈胜控制技术有限责任公司
Beijing Nimotion control Technology Co., Ltd.

目 录 Contents

1 关于手册 About the manual

1.1 概述 Overview

NiMotionModbusSDK 是为了方便用户的控制程序开发而设计的一体化电机二次开发包，该 SDK 内部封装了 Modbus 主站、电机通信协议，并提供了一套简洁易用的函数接口，极大的简化了用户开发控制程序的难度和工作量。本 SDK 适用于立迈胜公司的 Modbus 协议的一体化电机。

NiMotionModbusSDK is an integrated motor secondary development kit designed to facilitate user control program development. The SDK internally encapsulates the Modbus master station and motor communication protocol, and provides a set of simple and easy-to-use function interfaces, which greatly simplifies The difficulty and workload of the user to develop the control program. This SDK is suitable for integrated motors of Modbus protocol of NiMotion.

1.2 SDK 版本信息 SDK version information

Windows SDK 版本：v2.0.0
Windows SDK version: v2.0.0
Linux SDK 版本：v2.0.0
Linux SDK version: v2.0.0
Arm SDK 版本：v2.0.0
Arm SDK version: v2.0.0

1.3 SDK 支持转换器类型 SDK supports converter types

Windows 支持的转换器类型：支持通过串口方式访问的 USB-485 转换器，例如我司的 SCM-USB485 转换器；

Converter types supported by Windows: USB-485 converters that support access via serial port, such as our SCM-USB485 converter;

Linux 支持的转换器类型：支持通过串口方式访问的 USB-485 转换器，例如我司的 SCM-USB485 转换器；

The type of converter supported by Linux: USB-485 converter that supports access via serial port, such as our SCM-USB485 converter;

Arm 支持的转换器类型：支持通过串口方式访问的 USB-485 转换器，例如我司的 SCM-USB485 转换器。

The type of converter supported by Arm: USB-485 converter that supports access via serial port, such as our SCM-USB485 converter.

1.4 SDK 支持编程语言 SDK supports programming languages

C、C++、C#、Java、PHP 等以及其他支持调用 C 语言库的编程语言。

C, C++, C#, Java, PHP, etc. and other programming languages that support calling C language libraries.

1.5 编译环境 Compilation environment

x86 架构 Windows7 及以上 32 位环境
x86 architecture Windows7 and above 32-bit environment
x86 架构 Linux 64 位环境
x86 architecture Linux 64-bit environment
Arm 架构树莓派 32 位环境
Arm architecture Raspberry Pi 32-bit environment

1.6 开发环境 Development environment

Windows7 及以上 64 位操作系统
Windows7 and above 64-bit operating system
Debian 类发行版 Linux64 位操作系统(Ubuntu14.04LTS 以上或其他发行版)

Debian-based distribution Linux64-bit operating system (Ubuntu 14.04LTS or higher or other distributions)
Arch 类发行版 Linux64 位操作系统
Arch distribution version of Linux 64-bit operating system
Raspbian(树莓派)
Raspbian (Raspberry Pi)

1.7 环境配置 Environment configuration

Windows 下 SDK 内包含 Visual Studio 创建的 Demo 工程，无需另行配置；
The SDK under Windows contains the Demo project created by Visual Studio, without additional configuration;
Ubuntu 系统中，为了保证 Qt 的正常运行，必须安装一些必要的工具，包括 g++编译器，以及一些必要的库。可使用以下命令安装：
In Ubuntu system, in order to ensure the normal operation of Qt, some necessary tools must be installed, including g++ compiler, and some necessary libraries. It can be installed using the following command:
sudo apt-get install g++
sudo apt-get install libx11-dev libxext-dev libxtst-dev12
或可直接运行 sudo apt-get install build-essential
Or you can run sudo apt-get install build-essential directly

1.8 SDK 支持电机范围 SDK support motor range

STM28XX-485、BLM28XX-485、STM42XX-485、BLM42XX-485、STM57XX-485、BLM57XX-485、STM86XX-485、BLM86XX-485 系列电机。
STM28XX-485, BLM28XX-485, STM42XX-485, BLM42XX-485, STM57XX-485, BLM57XX-485, STM86XX-485, BLM86XX-485 series motors.

1.9 手册版本信息 Manual version information

手册版本 Manual version	日期 Date	修改记录 Modify record
A	2019/7/16	创建 Establish
B	2020/6/05	新增多串口和广播模式的说明 Description of newly added serial port and broadcast mode
C	2020/7/27	新增无刷电机的相关内容 Added related content about brushless motors
D	2020/10/12	增加英文翻译 Increase English translation

2 接口函数库说明及其使用 Interface function library description and its use

2.1 错误码定义 Error code definition

错误码 (10 进制) Error code (decimal)	含义 Meaning
0	执行成功 Execution succeed
1	不支持的设备类型 Unsupported device type
-1	执行失败 Execution failed
-2	电机地址选择错误 Motor address selection error
-3	参数传入错误 Parameter input error
-4	当前电机运动模式错误 Current motor motion mode error
-5	电机未在使能状态 The motor is not enabled
-6	串口被占用或有通信问题 The serial port is occupied or there is a communication problem
-7	当前电机不支持的操作 Operations not supported by the current motor

2.2 接口库函数结构体 Interface library function structure

2.2.1 MOTOR_INFO

描述：MOTOR_INFO 结构体包含设备信息。结构体将在 NiM_getMotorInfo 函数中被填充。

Description: MOTOR_INFO structure contains device information. The structure will be filled in the NiM_getMotorInfo function.

```
typedef struct _MOTOR_INFO
{
    int nAddr;
    uint8_t szSerialNumber[20];
    uint8_t szHardVersion[20];
    uint8_t szSoftVersion[20];
} MOTOR_INFO,*P_MOTOR_INFO;
```

成员(Member):

nAddr

电机地址(Motor address)。

szSerialNumber

电机序列号(Motor serial number)。

szHardVersion

硬件版本号(Hardware version number)。

szSoftVersion

APP 版本号(APP version number)。

2.2.2 SELFCHECK_RESULT

描述：SELFCHECK_RESULT 结构体是设备自检结构体，即表示设备自检结果。

Description: The SELFCHECK_RESULT structure is the device self-check structure, which means the result of the device self-check.

```
typedef struct _SELFCHECK_RESULT
{
    int nAddr;
    uint8_t nResult[4];
} SELFCHECK_RESULT, *P_SELFCHECK_RESULT;
```

成员(Member)：

nAddr

电机地址(Motor address)

nResult

电机自检结果(Motor self-test result)

数组四个元素分别为四个自检项目，对应分别为：编码器自检、通信自检、EEROM 自检、驱动自检。

The four elements of the array are respectively four self-check items, which correspond to: encoder self-check, communication self-check, EEROM self-check, and drive self-check.

2.2.3 WORK_MODE

描述：WORK_MODE 枚举是运动模式枚举，表示电机的运动模式。

Description: The WORK_MODE enumeration is a motion mode enumeration, which represents the motion mode of the motor.

```
typedef enum _WORK_MODE
{
    POSITION_MODE = 1,
    VELOCITY_MODE = 2,
    GOHOME_MODE = 3
} WORK_MODE;
```

成员(Member)：

POSITION_MODE

位置模式(Position mode)。

VELOCITY_MODE

速度模式(Velocity mode)。

GOHOME_MODE

原点回归模式(Go home mode)。

2.3 08 功能码 08 function code

本公司部分电机支持 08 功能码，用户在开发完成自己的上位机软件后，应将库文件夹内的 08MotorModel.json 置于应用程序当前目录下

Some motors of our company support 08 function codes. After the user has developed his own PC software, he should put 08MotorModel.json in the library folder under the current directory of the application

2.4 电机类型切换函数 Motor type switching function

2.4.1 NiM_specifyMotorType

描述：指定通信电机类型。

Description: Specify the type of communication motor.

int NiM_specifyMotorType(int nMotorType);
参数(Parameter)：
nMotorType
通信电机类型 (0:Modbus 步进电机 1:Modbus 无刷电机)
Communication motor type (0: Modbus stepper motor 1: Modbus brushless motor)

2.5 单串口接口库函数说明

Single serial port interface library function description

通信设备操作函数部分：
Communication equipment operation function part:

2.5.1 NiM_openDevice

描述：打开通信设备。
Description: Turn on the communication device.
int NiM_openDevice(int nType, const char* strConnectString);
参数(Parameter)：
nType
通信设备类型 (0:RTU 1:TCP，暂不支持)
Communication device type (0:RTU 1:TCP, not currently supported)
strConnectString
连接字符串，描述设备连接参数.内容格式为标准 JSON 格式，输入规范为:
The connection string describes the device connection parameters. The content format is standard JSON format, and the input specification is:
{"DeviceName": "COM1", "Baudrate": 115200, "Parity": "None|Odd|Even", "DataBits": 8, "StopBits": 1}
DeviceName
设备名
Baudrate
波特率
Parity
校验位
DataBits
数据位
StopBits
停止位
返回值：0 成功，其它表示错误码。
Return value: 0 success, others indicate error code.
示例(Example)：

```
#include "NiMotionMotorSDK.h"
int rc = NiM_openDevice(0, "{\"DeviceName\": \"COM7\", \"Baudrate\" : 115200, \"Parity\" : \"None\", \"DataBits\" : 8, \"StopBits\" : 1}");
if (rc != 0) {
    //执行失败的处理(Execution failure processing)
}
```

2.5.2 NiM_closeDevice

描述：关闭通信设备。
Description: Turn off the communication device.
int NiM_closeDevice();
返回值：0 执行成功。
Return value: 0 Successful execution.
示例(Example)：

```
#include "NiMotionMotorSDK.h"
int rc = NiM_closeDevice();
if (rc != 0) {
```

```
    //执行失败的处理(Execution failure processing)
}
```

在线电机管理函数(Online motor management functions):

2.5.3 NiM_scanMotors

描述：打开通信设备。
Description: Turn on the communication device.
int NiM_scanMotors(int nFromAddr, int nToAddr);
参数(parameter)：
nFromAddr
起始地址(Initial address)
nToAddr
结束地址(End address)
返回值：0 成功，其它表示错误码。
Return value: 0 success, others indicate error code.
示例(Example)：

```
#include "NiMotionMotorSDK.h"
int nFromAddr = 1;
int nToAddr = 10;
int rc = NiM_scanMotors(nFromAddr, nToAddr);
if (rc != 0) {
    //执行失败的处理(Execution failure processing)
}
```

2.5.4 NiM_getOnlineMotors

描述：获取在线电机列表。
Description: Get a list of online motors.
int NiM_getOnlineMotors(int* pAddrs, int* pCount);
参数(parameter)：
pAddrs
电机地址数组指针，数组大小为 247.
Motor address array pointer, the array size is 247.
pCount
数量指针(Quantity pointer)
返回值：0 成功，其它表示错误码。
Return value: 0 success, others indicate error code.
示例(Example)：

```
#include "NiMotionMotorSDK.h"
int addrs[247] = {0};
int nCount = 0;
int rc = NiM_getOnlineMotors(addrs, &nCount);
if (rc != 0) {
    //执行失败的处理(Execution failure processing)
}
```

2.5.5 NiM_isMotorOnline

描述：此函数用以判断电机是否在线。
Description: This function is used to judge whether the motor is online.
int NiM_isMotorOnline(int nAddr, BOOL* pOnline);
参数(parameter)：
nAddr
电机地址(Motor address)。
pOnline
指针，返回在线状态。0 为电机离线，1 为电机在线
Pointer, return to online status. 0 means the motor is offline, 1 means the motor is online

返回值：0 成功，其它表示错误码。
Return value: 0 success, others indicate error code.
示例(Example)：

```
#include "NiMotionMotorSDK.h"
int nAddr = 1;
BOOL isOnline = FALSE;
int rc = NiM_ isMotorOnline (nAddr, &isOnline);
if (rc != 0) {
    //执行失败的处理(Execution failure processing)
}
printf("Result: %d\r\n", isOnline);
```

2.5.6 NiM_getMotorInfo

描述：此函数用以获取电机基本信息。
Description: This function is used to obtain the basic information of the motor.

```
int NiM_getMotorInfo(int nAddr, MOTOR_INFO* pInfo);
```

参数(parameter)：
nAddr
电机地址(Motor address)。
pInfo
结构体指针，返回电机信息。
Structure pointer, return motor information.
返回值：0 成功，其它表示错误码。
Return value: 0 success, others indicate error code.
示例(Example)：

```
#include "NiMotionMotorSDK.h"
int nAddr = 1;
MOTOR_INFO motorInfo;
int rc = NiM_getMotorInfo(nAddr, &motorInfo);
if (rc != 0) {
    //执行失败的处理(Execution failure processing)
}
printf("StrHardVersion: %s\r\n", motorInfo.szHardVersion);
printf("StrSoftVersion: %s\r\n", motorInfo.szSoftVersion);
printf("StrSN: %s\r\n", motorInfo.szSerialNumber);
```

2.5.7 NiM_selfcheck

描述：此函数用以执行电机自检。无刷电机不支持此函数.
Description: This function is used to perform motor self-test. Brushless motors do not support this function.

```
int NiM_selfcheck(int nAddr, SELFCHECK_RESULT* pResult);
```

参数(parameter)：
nAddr
电机地址(Motor address)。
pResult
结构体指针，返回电机自检结果。结果输出 0 为合格，1 为不合格。
Structure pointer, returns the result of motor self-test. The result output is 0 for qualified and 1 for unqualified.
返回值：0 成功，其它表示错误码。
Return value: 0 success, others indicate error code.
示例(Example)：

```
#include "NiMotionMotorSDK.h"
int nAddr = 1;
SELFCHECK_RESULT selfCheckResult;
int rc = NiM_selfcheck(nAddr,&selfCheckResult);
```

```
if (rc != 0) {
    //执行失败的处理(Execution failure processing)
}
printf("The Motor selfCheck result is %d,%d,%d,%d\r\n",
        selfCheckResult.nResult[0],    //编码器自检(Encoder self-test)
        selfCheckResult.nResult[1],    // 通信自检(Communication self-check)
        selfCheckResult.nResult[2],    // EEROM 自检(EEROM self-check)
        selfCheckResult.nResult[3]);   //驱动自检(Drive self-check)
```

2.5.8 NiM_getLatestAlarm

描述：此函数用以获取电机最近的报警。

Description: This function is used to get the latest alarm of the motor.

int NiM_getLatestAlarm(int nAddr, int* pAlarmCode);

参数(parameter)：

nAddr

电机地址(Motor address)。

pAlarmCode

指针，返回电机报警值，参照附录报警参照表查看对应报警。

Pointer, return the motor alarm value, refer to the appendix alarm reference table to view the corresponding alarm.

返回值：0 成功，其它表示错误码。

Return value: 0 success, others indicate error code.

示例(Example)：

```
#include "NiMotionMotorSDK.h"
int nAddr = 1;
int nAlarmValue = 0;
int rc = NiM_getLatestAlarm(nAddr, &nAlarmValue);
if (rc != 0) {
    //执行失败的处理(Execution failure processing)
}
printf("The Motor LatestAlarm is %d \r\n", nAlarmValue);
```

2.5.9 NiM_getErrorCode

描述：此函数用以获取电机故障码。

Description: This function is used to get the motor fault code.

int NiM_getErrorCode(int nAddr, int* pErrorCode);

参数(parameter)：

nAddr

电机地址（Motor address）。

pErrorCode

指针，返回电机故障码。

Pointer, return the motor fault code.

返回值：0 成功，其它表示错误码。

Return value: 0 success, others indicate error code.

示例(Example)：

```
#include "NiMotionMotorSDK.h"
int nAddr = 1;
int nAlarmValue = 0;
int rc = NiM_getErrorCode (nAddr, &nAlarmValue);
if (rc != 0) {
    //执行失败的处理(Execution failure processing)
}
printf("The Motor ErrorCode is %d \r\n", nAlarmValue);
```

2.5.10 NiM_getHistoryAlarms

描述：此函数用以获取电机历史报警。

Description: This function is used to obtain historical motor alarms.

int NiM_getHistoryAlarms(int nAddr, int * pAlarmCode, int* pCount);

参数(parameter)：

nAddr

电机地址（Motor address）。

pAlarmCode

数组指针，返回电机报警值列表。参照附录报警参照表查看对应报警。

Array pointer, returns a list of motor alarm values. Refer to the appendix alarm reference table to view the corresponding alarms.

pCount

指针，返回报警个数

Pointer, return the number of alarms

返回值：0 成功，其它表示错误码。

Return value: 0 success, others indicate error code.

示例(Example)：

```
#include "NiMotionMotorSDK.h"
int nAddr = 1;
int nHistoryAlarm[16] = {0};
int nCount = 0;
int rc = NiM_getHistoryAlarms(nAddr, nHistoryAlarm, &nCount);
if (rc != 0) {
    //执行失败的处理(Execution failure processing)
}
printf("The Motor HistoryAlarmsCout is %d\r\n", nCount);
for(int i = 0; i< nCount; i++)
{
    printf("The Motor HistoryAlarms is %d \r\n", nHistoryAlarm[i]);
}
```

2.5.11 NiM_clearAlarms

描述：此函数用以清除电机报警。

Description: This function is used to clear the motor alarm.

int NiM_clearAlarms(int nAddr);

参数(parameter)：

nAddr

电机地址（Motor address）。

返回值：0 成功，其它表示错误码。

Return value: 0 success, others indicate error code.

示例(Example)：

```
#include "NiMotionMotorSDK.h"
int nAddr = 1;
int rc = NiM_clearAlarms(nAddr);
if (rc != 0) {
    //执行失败的处理(Execution failure processing)
}
```

2.5.12 NiM_clearErrorState

描述：此函数用以清除电机故障。

Description: This function is used to clear motor faults.

int NiM_clearErrorState(int nAddr);

参数(parameter)：

nAddr

电机地址（Motor address）。

返回值：0 成功，其它表示错误码。
Return value: 0 success, others indicate error code.
示例(Example)：

```
#include "NiMotionMotorSDK.h"
int nAddr = 1;
int rc = NiM_clearErrorState(nAddr);
if (rc != 0) {
    //执行失败的处理(Execution failure processing)
}
```

2.5.13 NiM_rebootMotor

描述：此函数用以重启电机。
Description: This function is used to restart the motor.
int NiM_rebootMotor(int nAddr);
参数(parameter)：
nAddr
电机地址（Motor address）。
返回值：0 成功，其它表示错误码。
Return value: 0 success, others indicate error code.
示例(Example)：

```
#include "NiMotionMotorSDK.h"
int nAddr = 1;
int rc = NiM_rebootMotor(nAddr);
if (rc != 0) {
    //执行失败的处理(Execution failure processing)
}
Sleep(1000)  //重启需要一定的时间，步进需要 1 秒，无刷则是 4 秒（It takes a certain time to restart, 1 second for stepping, 4 seconds for brushless）
```

电机控制函数（Motor control function）：

2.5.14 NiM_readParam

描述：此函数用以获取电机参数值。
Description: This function is used to obtain motor parameter values.
int NiM_readParam(int nAddr, int nParamID, int nBytes, int* pParamValue);
参数(parameter)：
nAddr
电机地址（Motor address）。
NParamID
参数 ID（The parameter ID）。
nBytes
字节数（Number of bytes）
pParamValue
指针，返回参数值。
Pointer, return parameter value.
返回值：0 成功，其它表示错误码。
Return value: 0 success, others indicate error code.
示例(Example)：

```
#include "NiMotionMotorSDK.h"
int nAddr = 1;
int nValue;
int rc = NiM_readParam(nAddr,0x1018,0x04, &nValue);    //获取电机当前位置（Get the current position of the motor）
if (rc != 0) {
    //执行失败的处理(Execution failure processing)
```

}

2.5.15 NiM_writeParam

描述：此函数用以设置电机参数值。
Description: This function is used to set motor parameter values.
int NiM_writeParam(int nAddr, int nParamID, int nBytes, int nParamValue);
参数(parameter)：
nAddr
电机地址（Motor address）。
NParamID
参数 ID（The parameter ID）。
nBytes
字节数（Number of bytes）。
pParamValue
参数值（Parameter value）。
返回值：0 成功，其它表示错误码。
Return value: 0 success, others indicate error code.
示例(Example)：

```
#include "NiMotionMotorSDK.h"
int nAddr = 1;
int nValue = 500;
int rc = NiM_writeParam(nAddr, 0x5B,4,200);     //设置电机最大速度（Set the maximum motor speed）
if (rc != 0) {
    //执行失败的处理(Execution failure processing)
}
```

注：有关于参数的读写，需要用 04 功能码发送的参数，需要在原本 ID 基础上+0x1000 的地址偏移和 03 功能码做出区分，不然会出现读写错误的情况
Note: For parameter reading and writing, parameters that need to be sent with 04 function code need to be distinguished from the original ID +0x1000 address offset and 03 function code, otherwise read and write errors will occur

2.5.16 NiM_saveParams

描述：此函数用以保存电机参数。
Description: This function is used to save motor parameters.
int NiM_saveParams(int nAddr);
参数(parameter)：
nAddr
电机地址（Motor address）。
返回值：0 成功，其它表示错误码。
Return value: 0 success, others indicate error code.
示例(Example)：

```
#include "NiMotionMotorSDK.h"
int nAddr = 1;
int rc = NiM_saveParams(nAddr);
if (rc != 0) {
    //执行失败的处理(Execution failure processing)
}
Sleep(1000)  //保存参数需要一定的时间，步进需要 500 毫秒，无刷则是 3 秒（It takes a certain amount of time to save the parameters, 500 milliseconds for stepping, 3 seconds for brushless.）.
```

2.5.17 NiM_restoreFactorySettings

描述：此函数用以恢复电机出厂参数设置。
Description: This function is used to restore the factory parameter settings of the motor.

int NiM_restoreFactorySettings(int nAddr);
参数(parameter)：
nAddr
电机地址（Motor address）。
返回值：0 成功，其它表示错误码。
Return value: 0 success, others indicate error code.
示例(Example)：

```
#include "NiMotionMotorSDK.h"
int nAddr = 1;
int rc = NiM_restoreFactorySettings(nAddr);
if (rc != 0) {
    //执行失败的处理 (Execution failure processing)
}
```

2.5.18 NiM_changeAddr

描述：此函数用以改变电机地址。
Description: This function is used to change the motor address.
int NiM_changeAddr(int nCurAddr, int nNewAddr);
参数(parameter)：
nCurAddr
当前电机地址（Current motor address）。
nNewAddr
新的电机地址（New motor address）
返回值：0 成功，其它表示错误码。
Return value: 0 success, others indicate error code.
示例(Example)：

```
#include "NiMotionMotorSDK.h"
int nCurAddr = 1;
int nNewAddr = 2;
int rc = NiM_changeAddr(nCurAddr, nNewAddr);
if (rc != 0) {
    //执行失败的处理 (Execution failure processing)
}
```

2.5.19 NiM_setDOState

描述：改变 DO 状态
Description: Change DO status
int NiM_setDOState(int nAddr, int nDOValue);
参数(parameter)：
nAddr
电机地址（Motor address）。
nDOValue
DO 状态值（DO status value）
返回值：0 成功，其它表示错误码。
Return value: 0 success, others indicate error code.
示例(Example)：

```
#include "NiMotionMotorSDK.h"
int nAddr = 1;
int nDOValue = 1;
int rc = NiM_setDOState(nAddr, nDOValue);
if (rc != 0) {
    //执行失败的处理 (Execution failure processing)
}
```

2.5.20 NiM_readDIState

描述：读取 DI 状态
Description: Read DI status
int NiM_readDIState(**int** nAddr, **int*** pDIState);
参数(parameter)：
nAddr
电机地址（Motor address）。
pDIState
指针，返回 DI 状态。
Pointer, return to DI state.
返回值：0 成功，其它表示错误码。
Return value: 0 success, others indicate error code.
示例(Example)：

```
#include "NiMotionMotorSDK.h"
int nAddr = 1;
int nDIValue = 0;
int rc = NiM_readDIState(nAddr, &nDIValue);
if (rc != 0) {
    //执行失败的处理 (Execution failure processing)
}
```

2.5.21 NiM_readDOState

描述：读取 DO 状态
Description: read DO status
int NiM_readDOState(**int** nAddr, **int*** pDOState);
参数(parameter)：
nAddr
电机地址（Motor address）。
pDOState
指针，返回 DO 状态。
Pointer, return to DO status.
返回值：0 成功，其它表示错误码。
Return value: 0 success, others indicate error code.
示例(Example)：

```
#include "NiMotionMotorSDK.h"
int nAddr = 1;
int nDOValue = 0;
int rc = NiM_readDIState(nAddr, &nDOValue,1);
if (rc != 0) {
    //执行失败的处理 (Execution failure processing)
}
```

2.5.22 NiM_changeWorkMode

描述：此函数用以改变电机运行模式。
Description: This function is used to change the motor running mode.
int NiM_changeWorkMode(int nAddr, WORK_MODE nMode);
参数(parameter)：
nAddr
电机地址（Motor address）。
nMode
运行模式（Operating mode）。
返回值：0 成功，其它表示错误码。
Return value: 0 success, others indicate error code.
示例(Example)：

```
#include "NiMotionMotorSDK.h"
```

```
int nAddr = 1;
int rc = NiM_changeWorkMode(nAddr, POSITION_MODE);
if (rc != 0) {
    //执行失败的处理 (Execution failure processing)
}
```

2.5.23 NiM_getCurrentStatus

描述：此函数用以获取当前电机状态字。
Description: This function is used to get the current motor status word.
int NiM_getCurrentStatus(**int** nAddr, **int*** pStatusWord);
参数(parameter)：
nAddr
电机地址（Motor address）。
pStatusWord
指针，返回状态字。
Pointer, return status word.
返回值：0 成功，其它表示错误码。
Return value: 0 success, others indicate error code.
示例(Example)：

```
#include "NiMotionMotorSDK.h"
int nAddr = 1;
int nStatus = 0;
int rc = NiM_getCurrentStatus(nAddr, &nStatus);
if (rc != 0) {
    //执行失败的处理 (Execution failure processing)
}
```

2.5.24 NiM_getCurrentPosition

描述：此函数用以获取当前电机当前位置。
Description: This function is used to get the current position of the current motor.
int NiM_getCurrentPosition(**int** nAddr, **int*** pPosition);
参数(parameter)：
nAddr
电机地址（Motor address）。
pPosition
指针，返回当前位置。
The pointer returns to the current position.
返回值：0 成功，其它表示错误码。
Return value: 0 success, others indicate error code.
示例(Example)：

```
#include "NiMotionMotorSDK.h"
int nAddr = 1;
int nPos = 0;
int rc = NiM_getCurrentPosition (nAddr, &nPos);
if (rc != 0) {
    //执行失败的处理 (Execution failure processing)
}
```

2.5.25 NiM_saveAsHome

描述：此函数用以将当前位置设置为原点。
Description: This function is used to set the current position as the home.
int NiM_saveAsHome(int nAddr);
参数(parameter)：
nAddr
电机地址（Motor address）。

返回值：0 成功，其它表示错误码。
Return value: 0 success, others indicate error code.
示例(Example)：

```
#include "NiMotionMotorSDK.h"
int nAddr = 1;
int rc = NiM_saveAsHome(nAddr);
if (rc != 0) {
    //执行失败的处理(Execution failure processing)
}
```

2.5.26 NiM_saveAsZero

描述：此函数用以将当前位置设置为零点。无刷电机不支持此函数.
Description: This function is used to set the current position as the zero point. Brushless motors do not support this function.

```
int NiM_saveAsZero(int nAddr);
```

参数(parameter)：
nAddr
电机地址（Motor address）。
返回值：0 成功，其它表示错误码。
Return value: 0 success, others indicate error code.
示例(Example)：

```
#include "NiMotionMotorSDK.h"
int nAddr = 1;
int rc = NiM_saveAsZero(nAddr);
if (rc != 0) {
    //执行失败的处理(Execution failure processing)
}
```

2.5.27 NiM_moveAbsolute

描述：此函数用以电机绝对位置运动。注：无刷电机需确保电机处在 PP 模式下。
Description: This function is used to move the absolute position of the motor. Note: The brushless motor needs to ensure that the motor is in PP mode.

```
int NiM_moveAbsolute(int nAddr, int nPosition);
```

参数(parameter)：
nAddr
电机地址（Motor address）。
nPosition
目标位置（target position.）。
返回值：0 成功，其它表示错误码。
Return value: 0 success, others indicate error code.
示例(Example)：

```
#include "NiMotionMotorSDK.h"
int nAddr = 1;
int rc = NiM_powerOn(nAddr);  //使能（Enable）
if (rc != 0) {
    //执行失败的处理(Execution failure processing)
}
rc = NiM_ moveAbsolute(nAddr,1200);
if (rc != 0) {
    //执行失败的处理(Execution failure processing)
}
```

2.5.28 NiM_moveRelative

描述：此函数用以电机相对位置运动。注：无刷电机需确保电机处在 PP 模式下。

Description: This function is used to move the relative position of the motor. Note: The brushless motor needs to ensure that the motor is in PP mode.

int NiM_moveRelative(int nAddr, int nDistance);

参数(parameter)：

nAddr

电机地址（Motor address）。

nDistance

运动距离（Movement distance.）。

返回值：0 成功，其它表示错误码。

Return value: 0 success, others indicate error code.

示例(Example)：

```
#include "NiMotionMotorSDK.h"
int nAddr = 1;
int rc = NiM_powerOn(nAddr);  //使能（Enable）
if (rc != 0) {
    //执行失败的处理(Execution failure processing)
}
rc = NiM_moveRelative(nAddr,1200);
if (rc != 0) {
    //执行失败的处理(Execution failure processing)
}
```

2.5.29 NiM_moveVelocity

描述：此函数用以速度模式下目标速度运行。注：无刷电机需确保电机处在 PV 模式下。

Description: This function is used to run at the target speed in speed mode. Note: The brushless motor needs to ensure that the motor is in PV mode.

int NiM_moveVelocity(int nAddr, int nVelocity);

参数(parameter)：

nAddr

电机地址（Motor address）。

nVelocity

目标速度（Target velocity.）。

返回值：0 成功，其它表示错误码。

Return value: 0 success, others indicate error code.

示例(Example)：

```
#include "NiMotionMotorSDK.h"
int nAddr = 1;
int rc = NiM_writeParam(nAddr,0x448,4,0);  //无刷步骤，使能前设速度为 0，防止电机突然启动（Brushless step, set the speed to 0 before enabling to prevent the motor from starting suddenly）
rc = NiM_powerOn(nAddr);//使能（Enable）
if (rc != 0) {
    //执行失败的处理(Execution failure processing)
}
rc = NiM_moveVelocity(nAddr,500); //若是无刷电机则给定速度单位为用户单位/s（If it is a brushless motor, the given speed unit is user unit/s）
if (rc != 0) {
    //执行失败的处理(Execution failure processing)
}
```

2.5.30 NiM_goHome

描述：此函数用以原点回归。注：无刷电机需确保电机处在 HM 模式下。

Description: This function is used for origin return. Note: For brushless motors, ensure that the motor is in HM mode.

int NiM_goHome(int nAddr, int nType);

参数(parameter)：
nAddr
电机地址（Motor address）。
nType
原点回归方式（Origin return method.）。
返回值：0 成功，其它表示错误码。
Return value: 0 success, others indicate error code.
示例(Example)：

```
/*----------步进 Step----------*/
#include "NiMotionMotorSDK.h"
int nAddr = 1;
int nType = 31;
int rc = NiM_powerOn(nAddr);  //使能（Enable）
if (rc != 0) {
    //执行失败的处理(Execution failure processing)
}
rc = NiM_goHome(nAddr, nType);
if (rc != 0) {
    //执行失败的处理(Execution failure processing)
}
/*----------无刷 Brushless----------*/
//堵转找寻原点方式（The way to find the home by blocking）
#include "NiMotionMotorSDK.h"
int nAddr = 1;
int nType = 37;//原点回归方式（home mode）
int rc = NiM_writeParam(nAddr,0xD5,2,15); //负限位开关（Negative limit switch）
if (rc != 0) {
    //执行失败的处理(Execution failure processing)
}
rc = NiM_writeParam(nAddr,0xD6,2,0); //低电平有效（Active low）
if (rc != 0) {
    //执行失败的处理(Execution failure processing)
}
rc = NiM_writeParam(nAddr,0x419,4,1000); //设置寻找原点信号速度（Set the speed of finding the origin signal）
if (rc != 0) {
    //执行失败的处理(Execution failure processing)
}
rc = NiM_writeParam(nAddr,0x 41B,4,200000); //设置回零加速度（Set zero acceleration）
if (rc != 0) {
    //执行失败的处理(Execution failure processing)
}
rc = NiM_powerOn(nAddr);//使能（Enable）
if (rc != 0) {
    //执行失败的处理(Execution failure processing)
}
rc = NiM_goHome(nAddr, nType);
if (rc != 0) {
    //执行失败的处理(Execution failure processing)
}
```

2.5.31 NiM_powerOn

描述：此函数用以电机使能。
Description: This function is used to enable the motor.

int NiM_powerOn(int nAddr);
参数(parameter)：
nAddr
电机地址（Motor address）。
返回值：0 成功，其它表示错误码。
Return value: 0 success, others indicate error code.
示例(Example)：

```
#include "NiMotionMotorSDK.h"
int nAddr = 1;
int rc = NiM_powerOn(nAddr);
if (rc != 0) {
    //执行失败的处理(Execution failure processing)
}
```

2.5.32 NiM_powerOff

描述：此函数用以电机脱机。
Description: This function is used to offline the motor.
int NiM_powerOff(int nAddr);
参数(parameter)：
nAddr
电机地址（Motor address）。
返回值：0 成功，其它表示错误码。
Return value: 0 success, others indicate error code.
示例(Example)：

```
#include "NiMotionMotorSDK.h"
int nAddr = 1;
int rc = NiM_powerOff(nAddr);
if (rc != 0) {
    //执行失败的处理(Execution failure processing)
}
```

2.5.33 NiM_stop

描述：此函数用以电机停止当前动作。
Description: This function is used to stop the current action of the motor.
int NiM_stop(int nAddr);
参数(parameter)：
nAddr
电机地址（Motor address）。
返回值：0 成功，其它表示错误码。
Return value: 0 success, others indicate error code.
示例(Example)：

```
#include "NiMotionMotorSDK.h"
int nAddr = 1;
int rc = NiM_stop(nAddr);
if (rc != 0) {
    //执行失败的处理(Execution failure processing)
}
```

2.5.34 NiM_fastStop

描述：此函数用以电机急停。
Description: This function is used for emergency stop of the motor.
int NiM_fastStop(int nAddr);
参数(parameter)：
nAddr
电机地址（Motor address）。

返回值：0 成功，其它表示错误码。
Return value: 0 success, others indicate error code.
示例(Example)：

```
#include "NiMotionMotorSDK.h"
int nAddr = 1;
int rc = NiM_fastStop(nAddr);
if (rc != 0) {
    //执行失败的处理(Execution failure processing)
}
```

2.5.35 NiM_sendSync

描述：此函数用以发送同步信号。
Description: This function is used to send synchronization signals.
int NiM_sendSync(void);
该功能暂时缺省（This function is temporarily default.）。

2.5.36 NiM_setDebug

描述：SDK 调试模式，开启后有关函数执行时会输出具体信息
Description: SDK debugging mode, specific information will be output when related functions are executed after opening
int NiM_setDebug(BOOL flag);
参数：TRUE 为开启调试模式，FALSE 为关闭调试模式，默认为关闭。
Parameters: TRUE means to turn on the debugging mode, FALSE to turn off the debugging mode, the default is off.
返回值：0 成功，其它表示错误码。
Return value: 0 success, others indicate error code.
示例(Example)：

```
#include "NiMotionMotorSDK.h"
int rc = NiM_setDebug(TRUE);
if (rc != 0) {
    //执行失败的处理(Execution failure processing)
}
```

2.6 多串口接口库函数说明 Multi-serial interface library function description

通信设备操作函数部分(Communication equipment operation function part)：

2.6.1 NiM_MPopenDevice

描述：打开通信设备（单、多个串口）。
Description: Open communication equipment (single, multiple serial ports).
int NiM_MPopenDevice(int nType, const char* strConnectString);
参数(parameter)：
nType
通信设备类型 (0:RTU 1:TCP，暂不支持)
Communication device type (0:RTU 1:TCP, not currently supported)
strConnectString
连接字符串，描述设备连接参数.内容格式为标准 JSON 格式，输入规范为:
The connection string describes the device connection parameters. The content format is standard JSON format, and the input specification is:
{"DeviceName": " COM1&COM2&COM3", "Baudrate": 115200, "Parity": "None|Odd|Even", "DataBits": 8, "StopBits": 1}

DeviceName
串口名，通过“&”符号分隔
Serial port name, separated by "&" symbol
Baudrate
波特率
Parity
校验位
DataBits
数据位
StopBits
停止位
返回值：0 成功，其它表示错误码。
Return value: 0 success, others indicate error code.
示例(Example)：

```
#include "NiMotionMotorSDK.h"
int rc = NiM_MPopenDevice(0, "{\"DeviceName\": \"COM1&COM2\", \"Baudrate\" : 115200, \"Parity\" : \"None\", \"DataBits\" : 8, \"StopBits\" : 1}");
if (rc != 0) {
    //执行失败的处理(Execution failure processing)
}
```

2.6.2 NiM_MPcloseDevice

描述：关闭所有通信设备。
Description: Turn off all communication equipment.
int NiM_MPcloseDevice();
返回值：0 执行成功。
Return value: 0 Successful execution.
示例(Example)：

```
#include "NiMotionMotorSDK.h"
int rc = NiM_MPcloseDevice();
if (rc != 0) {
    //执行失败的处理(Execution failure processing)
}
```

在线电机管理函数(Online motor management functions:):

2.6.3 NiM_MPscanMotors

描述：打开通信设备。
Description: Turn on the communication device.
int NiM_MPscanMotors(const char* strPort, int nFromAddr, int nToAddr);
参数(parameter)：
strPort
需要操作的串口名(The name of the serial port to be operated)
nFromAddr
起始地址(initial address)
nToAddr
结束地址(End address)
返回值：0 成功，其它表示错误码。
Return value: 0 success, others indicate error code.
示例(Example)：

```
#include "NiMotionMotorSDK.h"
string strPort = "COM1";
int nFromAddr = 1;
int nToAddr = 10;
int rc = NiM_MPscanMotors(strPort, nFromAddr, nToAddr);
if (rc != 0) {
```

```
    //执行失败的处理(Execution failure processing)
}
```

2.6.4 NiM_MPgetOnlineMotors

描述：获取在线电机列表。
Description: Get a list of online motors.
int NiM_MPgetOnlineMotors(const char* strPort, int* pAddrs, int* pCount);
参数(parameter)：
strPort
需要操作的串口名(The name of the serial port to be operated)
pAddrs
电机地址数组指针，数组大小为 247(Motor address array pointer, the array size is 247)

pCount
数量指针(Quantity pointer)
返回值：0 成功，其它表示错误码。
Return value: 0 success, others indicate error code.
示例(Example)：

```
#include "NiMotionMotorSDK.h"
string strPort = "COM1";
int addrs[247] = {0};
int nCount = 0;
int rc = NiM_MPgetOnlineMotors(strPort, addrs, &nCount);
if (rc != 0) {
    //执行失败的处理(Execution failure processing)
}
```

2.6.5 NiM_MPisMotorOnline

描述：此函数用以判断电机是否在线。
Description: This function is used to judge whether the motor is online.
int NiM_MPisMotorOnline(const char* strPort, int nAddr, BOOL* pOnline);
参数(parameter)：
strPort
需要操作的串口名(The name of the serial port to be operated)
nAddr
电机地址（Motor address）。
pOnline
指针，返回在线状态。0 为电机离线，1 为电机在线
Pointer, return to online status. 0 means the motor is offline, 1 means the motor is online
返回值：0 成功，其它表示错误码。
Return value: 0 success, others indicate error code.
示例(Example)：

```
#include "NiMotionMotorSDK.h"
string strPort = "COM1";
int nAddr = 1;
BOOL isOnline = FALSE;
int rc = NiM_MP isMotorOnline (strPort, nAddr, &isOnline);
if (rc != 0) {
    //执行失败的处理(Execution failure processing)
}
printf("Result: %d\r\n", isOnline);
```

2.6.6 NiM_MPgetMotorInfo

描述：此函数用以获取电机基本信息。
Description: This function is used to obtain the basic information of the motor.

int NiM_MPgetMotorInfo(const char* strPort, int nAddr, MOTOR_INFO* pInfo);

参数(parameter)：

strPort

需要操作的串口名(The name of the serial port to be operated)

nAddr

电机地址（Motor address）。

pInfo

结构体指针，返回电机信息(Structure pointer, return motor information.)。

返回值：0 成功，其它表示错误码。

Return value: 0 success, others indicate error code.

示例(Example)：

```
#include "NiMotionMotorSDK.h"
string strPort = "COM1";
int nAddr = 1;
MOTOR_INFO motorInfo;
int rc = NiM_MPgetMotorInfo(strPort, nAddr, &motorInfo);
if (rc != 0) {
    //执行失败的处理(Execution failure processing)
}
printf("StrHardVersion: %s\r\n", motorInfo.szHardVersion);
printf("StrSoftVersion: %s\r\n", motorInfo.szSoftVersion);
printf("StrSN: %s\r\n", motorInfo.szSerialNumber);
```

2.6.7 NiM_MPselfcheck

描述：此函数用以执行电机自检。无刷电机不支持此函数.

Description: This function is used to perform motor self-test. Brushless motors do not support this function.

int NiM_MPselfcheck(const char* strPort, int nAddr, SELFCHECK_RESULT* pResult);

参数(parameter)：

strPort

需要操作的串口名(The name of the serial port to be operated)

nAddr

电机地址（Motor address）。

pResult

结构体指针，返回电机自检结果。结果输出 0 为合格，1 为不合格。

Structure pointer, returns the result of motor self-test. The result output is 0 for qualified and 1 for unqualified.

返回值：0 成功，其它表示错误码。

Return value: 0 success, others indicate error code.

示例(Example)：

```
#include "NiMotionMotorSDK.h"
string strPort = "COM1";
int nAddr = 1;
SELFCHECK_RESULT selfCheckResult;
int rc = NiM_MPselfcheck(strPort, nAddr,&selfCheckResult);
if (rc != 0) {
    //执行失败的处理(Execution failure processing)
}
printf("The Motor selfCheck result is %d,%d,%d,%d\r\n",
        selfCheckResult.nResult[0],     //编码器自检(Encoder self-test)
        selfCheckResult.nResult[1],     // 通信自检(Communication self-check)
        selfCheckResult.nResult[2],     // EEROM 自检(EEROM self-check)
        selfCheckResult.nResult[3]);    //驱动自检(Drive self-check)
```

2.6.8 NiM_MPgetLatestAlarm

描述：此函数用以获取电机最近的报警。

Description: This function is used to get the latest alarm of the motor.

int NiM_MPgetLatestAlarm(const char* strPort, int nAddr, int* pAlarmCode);

参数(parameter)：

strPort

需要操作的串口名(The name of the serial port to be operated)

nAddr

电机地址（Motor address）。

pAlarmCode

指针，返回电机报警值，参照附录报警参照表查看对应报警。

Pointer, return the motor alarm value, refer to the appendix alarm reference table to view the corresponding alarm.

返回值：0 成功，其它表示错误码。

Return value: 0 success, others indicate error code.

示例(Example)：

```
#include "NiMotionMotorSDK.h"
string strPort = "COM1";
int nAddr = 1;
int nAlarmValue = 0;
int rc = NiM_MPgetLatestAlarm(strPort, nAddr, &nAlarmValue);
if (rc != 0) {
    //执行失败的处理(Execution failure processing)
}
printf("The Motor LatestAlarm is %d \r\n", nAlarmValue);
```

2.6.9 NiM_MPgetErrorCode

描述：此函数用以获取电机故障码。

Description: This function is used to get the motor fault code.

int NiM_MPgetErrorCode(const char* strPort, int nAddr, int* pErrorCode);

参数(parameter)：

strPort

需要操作的串口名(The name of the serial port to be operated)

nAddr

电机地址（Motor address）。

pErrorCode

指针，返回电机故障码(Pointer, return the motor fault code)。

返回值：0 成功，其它表示错误码。

Return value: 0 success, others indicate error code.

示例(Example)：

```
#include "NiMotionMotorSDK.h"
string strPort = "COM1";
int nAddr = 1;
int nAlarmValue = 0;
int rc = NiM_MPgetErrorCode (strPort, nAddr, &nAlarmValue);
if (rc != 0) {
    //执行失败的处理(Execution failure processing)
}
printf("The Motor ErrorCode is %d \r\n", nAlarmValue);
```

2.6.10 NiM_MPgetHistoryAlarms

描述：此函数用以获取电机历史报警。

Description: This function is used to obtain historical motor alarms.

int NiM_MPgetHistoryAlarms(const char* strPort, int nAddr, int * pAlarmCode, int* pCount);

参数(parameter)：

strPort
需要操作的串口名(The name of the serial port to be operated)
nAddr
电机地址（Motor address）。
pAlarmCode
数组指针，返回电机报警值列表。参照附录报警参照表查看对应报警。
Array pointer, returns a list of motor alarm values. Refer to the appendix alarm reference table to view the corresponding alarms.
pCount
指针，返回报警个数(Pointer, return the number of alarms)
返回值：0 成功，其它表示错误码。
Return value: 0 success, others indicate error code.
示例(Example)：

```
#include "NiMotionMotorSDK.h"
string strPort = "COM1";
int nAddr = 1;
int nHistoryAlarm[16] = {0};
int nCount = 0;
int rc = NiM_MPgetHistoryAlarms(strPort, nAddr, nHistoryAlarm, &nCount);
if (rc != 0) {
    //执行失败的处理(Execution failure processing)
}
printf("The Motor HistoryAlarmsCout is %d\r\n", nCount);
for(int i = 0; i< nCount; i++)
{
    printf("The Motor HistoryAlarms is %d \r\n", nHistoryAlarm[i]);
}
```

2.6.11 NiM_MPclearAlarms

描述：此函数用以清除电机报警。
Description: This function is used to clear the motor alarm.

```
int NiM_MPclearAlarms(const char* strPort, int nAddr);
```

参数(parameter)：
strPort
需要操作的串口名(The name of the serial port to be operated)
nAddr
电机地址（Motor address）。
返回值：0 成功，其它表示错误码。
Return value: 0 success, others indicate error code.
示例(Example)：

```
#include "NiMotionMotorSDK.h"
string strPort = "COM1";
int nAddr = 1;
int rc = NiM_MPclearAlarms(strPort, nAddr);
if (rc != 0) {
    //执行失败的处理(Execution failure processing)
}
```

2.6.12 NiM_MPclearErrorState

描述：此函数用以清除电机故障。
Description: This function is used to clear motor faults.

```
int NiM_MPclearErrorState(const char* strPort, int nAddr);
```

参数(parameter)：
strPort
需要操作的串口名(The name of the serial port to be operated)

nAddr
电机地址（Motor address）。
返回值：0 成功，其它表示错误码。
Return value: 0 success, others indicate error code.
示例(Example)：

```
#include "NiMotionMotorSDK.h"
string strPort = "COM1";
int nAddr = 1;
int rc = NiM_MPclearErrorState(strPort, nAddr);
if (rc != 0) {
    //执行失败的处理(Execution failure processing)
}
```

2.6.13 NiM_MPrebootMotor

描述：此函数用以重启电机。
Description: This function is used to restart the motor.
参数(parameter)：
strPort
需要操作的串口名(The name of the serial port to be operated)
nAddr
电机地址（Motor address）。
返回值：0 成功，其它表示错误码。
Return value: 0 success, others indicate error code.
示例(Example)：

```
#include "NiMotionMotorSDK.h"
string strPort = "COM1";
int nAddr = 1;
int rc = NiM_MPrebootMotor(strPort, nAddr);
if (rc != 0) {
    //执行失败的处理(Execution failure processing)
}
Sleep(1000)  //重启需要一定的时间，步进需要 1 秒，无刷则是 4 秒(It takes a certain time to restart, 1 second for stepping, 4 seconds for brushless)
```

电机控制函数(Motor control function:):

2.6.14 NiM_MPreadParam

描述：此函数用以获取电机参数值。
Description: This function is used to obtain motor parameter values.
int NiM_MPreadParam(const char* strPort, int nAddr, int nParamID, int nBytes, int* pParamValue);
参数(parameter)：
strPort
需要操作的串口名(The name of the serial port to be operated)
nAddr
电机地址（Motor address）。
NParamID
参数 ID(The parameter ID)。
nBytes
字节数(Number of bytes)
pParamValue
指针，返回参数值(Pointer, return parameter value.)。
返回值：0 成功，其它表示错误码。
Return value: 0 success, others indicate error code.
示例(Example)：

```
#include "NiMotionMotorSDK.h"
string strPort = "COM1";
int nAddr = 1;
int nValue;
int rc = NiM_MPreadParam(strPort, nAddr,0x1018,0x04, &nValue);     //获取电机当前位置(Get the current position of the motor)
if (rc != 0) {
    //执行失败的处理(Execution failure processing)
}
```

2.6.15 NiM_MPwriteParam

描述：此函数用以设置电机参数值。

Description: This function is used to set motor parameter values.

int NiM_MPwriteParam(const char* strPort, int nAddr, int nParamID, int nBytes, int nParamValue);

参数(parameter)：

strPort

需要操作的串口名(The name of the serial port to be operated)

nAddr

电机地址（Motor address）。

NParamID

参数 ID(The parameter ID)。

nBytes

字节数(Number of bytes)。

pParamValue

参数值(Parameter value)。

返回值：0 成功，其它表示错误码。

Return value: 0 success, others indicate error code.

示例(Example)：

```
#include "NiMotionMotorSDK.h"
string strPort = "COM1";
int nAddr = 1;
int nValue = 500;
int rc = NiM_MPwriteParam(strPort, nAddr, 0x5B,4,200);     //设置电机最大速度(Set the maximum motor speed)
if (rc != 0) {
    //执行失败的处理(Execution failure processing)
}
```

注：有关于参数的读写，需要用 04 功能码发送的参数，需要在原本 ID 基础上+0x1000 的地址偏移和 03 功能码做出区分，不然会出现读写错误的情况

Note: For parameter reading and writing, parameters that need to be sent with 04 function code need to be distinguished from the original ID +0x1000 address offset and 03 function code, otherwise read and write errors will occur

2.6.16 NiM_MPsaveParams

描述：此函数用以保存电机参数。

Description: This function is used to save motor parameters.

int NiM_MPsaveParams(const char* strPort, int nAddr);

参数(parameter)：

strPort

需要操作的串口名（The name of the serial port to be operated）

nAddr

电机地址（Motor address）。

返回值：0 成功，其它表示错误码。

Return value: 0 success, others indicate error code.

示例(Example)：

```
#include "NiMotionMotorSDK.h"
string strPort = "COM1";
int nAddr = 1;
int rc = NiM_MPsaveParams(strPort, nAddr);
if (rc != 0) {
    //执行失败的处理(Execution failure processing)
}
Sleep(1000)  //保存参数需要一定的时间，步进需要 500 毫秒，无刷则是 3 秒（It takes a certain amount of time to save the parameters, 500 milliseconds for stepping, 3 seconds for brushless.）.
```

2.6.17 NiM_MPrestoreFactorySettings

描述：此函数用以恢复电机出厂参数设置。

Description: This function is used to restore the factory parameter settings of the motor.

```
int NiM_MPrestoreFactorySettings(const char* strPort, int nAddr);
```

参数(parameter)：

strPort

需要操作的串口名（The name of the serial port to be operated）

nAddr

电机地址（Motor address）。

返回值：0 成功，其它表示错误码。

Return value: 0 success, others indicate error code.

示例(Example)：

```
#include "NiMotionMotorSDK.h"
string strPort = "COM1";
int nAddr = 1;
int rc = NiM_MPrestoreFactorySettings(strPort, nAddr);
if (rc != 0) {
    //执行失败的处理(Execution failure processing)
}
```

2.6.18 NiM_MPchangeAddr

描述：此函数用以改变电机地址。

Description: This function is used to change the motor address.

```
int NiM_MPchangeAddr(int nCurAddr, int nNewAddr);
```

参数(parameter)：

strPort

需要操作的串口名（The name of the serial port to be operated）

nCurAddr

当前电机地址（Current motor address）。

nNewAddr

新的电机地址（New motor address）

返回值：0 成功，其它表示错误码。

Return value: 0 success, others indicate error code.

示例(Example)：

```
#include "NiMotionMotorSDK.h"
string strPort = "COM1";
int nCurAddr = 1;
int nNewAddr = 2;
int rc = NiM_MPchangeAddr(strPort, nCurAddr, nNewAddr);
if (rc != 0) {
    //执行失败的处理(Execution failure processing)
}
```

2.6.19 NiM_MPsetDOState

描述：改变 DO 状态
Description: Change DO status
int NiM_MPsetDOState(const char* strPort, int nAddr, int nDOValue);
参数(parameter)：
strPort
需要操作的串口名（The name of the serial port to be operated）
nAddr
电机地址（Motor address）。
nDOValue
DO 状态值（DO status value）
返回值：0 成功，其它表示错误码。
Return value: 0 success, others indicate error code.
示例(Example)：

```
#include "NiMotionMotorSDK.h"
string strPort = "COM1";
int nAddr = 1;
int nDOValue = 1;
int rc = NiM_MPsetDOState(strPort, nAddr, nDOValue);
if (rc != 0) {
    //执行失败的处理(Execution failure processing)
}
```

2.6.20 NiM_MPreadDIState

描述：读取 DI 状态
Description: Read DI status
int NiM_MPreadDIState(const char* strPort, int nAddr, **int*** pDIState);
参数(parameter)：
strPort
需要操作的串口名（The name of the serial port to be operated）
nAddr
电机地址（Motor address）。
pDIState
指针，返回 DI 状态（Pointer, return to DI state）。
返回值：0 成功，其它表示错误码。
Return value: 0 success, others indicate error code.
示例(Example)：

```
#include "NiMotionMotorSDK.h"
string strPort = "COM1";
int nAddr = 1;
int nDIValue = 0;
int rc = NiM_MPreadDIState(strPort, nAddr, &nDIValue);
if (rc != 0) {
    //执行失败的处理(Execution failure processing)
}
```

2.6.21 NiM_MPreadDOState

描述：读取 DO 状态
Description: read DO status
int NiM_MPreadDOState(const char* strPort, int nAddr, **int*** pDOState);
参数(parameter)：
strPort
需要操作的串口名（The name of the serial port to be operated）
nAddr

电机地址（Motor address）。
pDOState
指针，返回 DO 状态（Pointer, return to DO status）。
返回值：0 成功，其它表示错误码。
Return value: 0 success, others indicate error code.
示例(Example)：

```
#include "NiMotionMotorSDK.h"
string strPort = "COM1";
int nAddr = 1;
int nDOValue = 0;
int rc = NiM_MPreadDIState(strPort, nAddr, &nDOValue,1);
if (rc != 0) {
    //执行失败的处理(Execution failure processing)
}
```

2.6.22 NiM_MPchangeWorkMode

描述：此函数用以改变电机运行模式。
Description: This function is used to change the motor running mode.
int NiM_MPchangeWorkMode(const char* strPort, int nAddr, WORK_MODE nMode);
参数(parameter)：
strPort
需要操作的串口名（The name of the serial port to be operated）
nAddr
电机地址（Motor address）。
nMode
运行模式（Operating mode）。
返回值：0 成功，其它表示错误码。
Return value: 0 success, others indicate error code.
示例(Example)：

```
#include "NiMotionMotorSDK.h"
string strPort = "COM1";
int nAddr = 1;
int rc = NiM_MPchangeWorkMode(strPort, nAddr, POSITION_MODE);
if (rc != 0) {
    //执行失败的处理(Execution failure processing)
}
```

2.6.23 NiM_MPgetCurrentStatus

描述：此函数用以获取当前电机状态字。
Description: This function is used to get the current motor status word.
int NiM_MPgetCurrentStatus(const char* strPort, int nAddr, **int*** pStatusWord);
参数(parameter)：
strPort
需要操作的串口名（The name of the serial port to be operated）
nAddr
电机地址（Motor address）。
pStatusWord
指针，返回状态字（Pointer, return status word）。
返回值：0 成功，其它表示错误码。
Return value: 0 success, others indicate error code.
示例(Example)：

```
#include "NiMotionMotorSDK.h"
string strPort = "COM1";
int nAddr = 1;
int nStatus = 0;
```

```
int rc = NiM_MPgetCurrentStatus(strPort, nAddr, &nStatus);
if (rc != 0) {
    //执行失败的处理(Execution failure processing)
}
```

2.6.24 NiM_MPgetCurrentPosition

描述：此函数用以获取当前电机当前位置。

Description: This function is used to get the current position of the current motor.

int NiM_MPgetCurrentPosition(const char* strPort, int nAddr, **int*** pPosition);

参数(parameter)：

strPort

需要操作的串口名（The name of the serial port to be operated）

nAddr

电机地址（Motor address）。

pPosition

指针，返回当前位置（The pointer returns to the current position）。

返回值：0 成功，其它表示错误码。

Return value: 0 success, others indicate error code.

示例(Example)：

```
#include "NiMotionMotorSDK.h"
string strPort = "COM1";
int nAddr = 1;
int nPos = 0;
int rc = NiM_MPgetCurrentPosition (strPort, nAddr, &nPos);
if (rc != 0) {
    //执行失败的处理(Execution failure processing)
}
```

2.6.25 NiM_MPsaveAsHome

描述：此函数用以将当前位置设置为原点。

Description: This function is used to set the current position as the home.

int NiM_MPsaveAsHome(const char* strPort, int nAddr);

参数(parameter)：

strPort

需要操作的串口名（The name of the serial port to be operated）

nAddr

电机地址（Motor address）。

返回值：0 成功，其它表示错误码。

Return value: 0 success, others indicate error code.

示例(Example)：

```
#include "NiMotionMotorSDK.h"
string strPort = "COM1";
int nAddr = 1;
int rc = NiM_MPsaveAsHome(strPort, nAddr);
if (rc != 0) {
    //执行失败的处理(Execution failure processing)
}
```

2.6.26 NiM_MPsaveAsZero

描述：此函数用以将当前位置设置为零点。无刷电机不支持此函数.

Description: This function is used to set the current position as the zero point. Brushless motors do not support this function.

int NiM_MPsaveAsZero(const char* strPort, int nAddr);

参数(parameter)：

strPort

需要操作的串口名（The name of the serial port to be operated）
nAddr
电机地址（Motor address）。
返回值：0 成功，其它表示错误码。
Return value: 0 success, others indicate error code.
示例(Example)：

```
#include "NiMotionMotorSDK.h"
string strPort = "COM1";
int nAddr = 1;
int rc = NiM_MPsaveAsZero(strPort, nAddr);
if (rc != 0) {
    //执行失败的处理(Execution failure processing)
}
```

2.6.27 NiM_MPmoveAbsolute

描述：此函数用以电机绝对位置运动。注：无刷电机需确保电机处在 PP 模式下。
Description: This function is used to move the absolute position of the motor. Note: The brushless motor needs to ensure that the motor is in PP mode.

```
int NiM_MPmoveAbsolute(const char* strPort, int nAddr, int nPosition);
```

参数(parameter)：
strPort
需要操作的串口名（The name of the serial port to be operated）
nAddr
电机地址（Motor address）。
nPosition
目标位置（target location）。
返回值：0 成功，其它表示错误码。
Return value: 0 success, others indicate error code.
示例(Example)：

```
#include "NiMotionMotorSDK.h"
string strPort = "COM1";
int nAddr = 1;
int rc = NiM_MPpowerOn(strPort, nAddr);     //使能（Enable）
if (rc != 0) {
    //执行失败的处理(Execution failure processing)
}
rc = NiM_MPmoveAbsolute(strPort, nAddr,1200);
if (rc != 0) {
    //执行失败的处理(Execution failure processing)
}
```

2.6.28 NiM_MPmoveRelative

描述：此函数用以电机相对位置运动。注：无刷电机需确保电机处在 PP 模式下。
Description: This function is used to move the relative position of the motor. Note: The brushless motor needs to ensure that the motor is in PP mode.

```
int NiM_MPmoveRelative(const char* strPort, int nAddr, int nDistance);
```

参数(parameter)：
strPort
需要操作的串口名（The name of the serial port to be operated）
nAddr
电机地址（Motor address）。
nDistance
运动距离（Movement distance）。
返回值：0 成功，其它表示错误码。
Return value: 0 success, others indicate error code.

示例(Example)：

```
#include "NiMotionMotorSDK.h"
string strPort = "COM1";
int nAddr = 1;
int rc = NiM_MPpowerOn(strPort, nAddr);     //使能（Enable）
if (rc != 0) {
    //执行失败的处理(Execution failure processing)
}
rc = NiM_MPmoveRelative(strPort, nAddr,1200);
if (rc != 0) {
    //执行失败的处理(Execution failure processing)
}
```

2.6.29 NiM_MPmoveVelocity

描述：此函数用以速度模式下目标速度运行。注：无刷电机需确保电机处在 PV 模式下。

Description: This function is used to run at the target speed in speed mode. Note: The brushless motor needs to ensure that the motor is in PV mode.

```
int NiM_MPmoveVelocity(const char* strPort, int nAddr, int nVelocity);
```

参数(parameter)：

strPort

需要操作的串口名（The name of the serial port to be operated）

nAddr

电机地址（Motor address）。

nVelocity

目标速度(Target speed)。

返回值：0 成功，其它表示错误码。

Return value: 0 success, others indicate error code.

示例(Example)：

```
#include "NiMotionMotorSDK.h"
string strPort = "COM1";
int nAddr = 1;
int rc = NiM_MPpowerOn(strPort, nAddr);     //使能（Enable）
if (rc != 0) {
    //执行失败的处理(Execution failure processing)
}
rc = NiM_MPmoveVelocity(strPort, nAddr,500);
if (rc != 0) {
    //执行失败的处理(Execution failure processing)
}
```

2.6.30 NiM_MPgoHome

描述：此函数用以原点回归。注：无刷电机需确保电机处在 HM 模式下。

Description: This function is used for origin return. Note: For brushless motors, ensure that the motor is in HM mode.

```
int NiM_MPgoHome(const char* strPort, int nAddr, int nType);
```

参数(parameter)：

strPort

需要操作的串口名（The name of the serial port to be operated）

nAddr

电机地址（Motor address）。

nType

原点回归方式(Go home method)。

返回值：0 成功，其它表示错误码。

Return value: 0 success, others indicate error code.

示例(Example)：

```
/*---------------步进 Stepper------------*/
#include "NiMotionMotorSDK.h"
string strPort = "COM1";
int nAddr = 1;
int nType = 31;
int rc = NiM_MPpowerOn(strPort, nAddr);      //使能（Enable）
if (rc != 0) {
    //执行失败的处理(Execution failure processing)
}
rc = NiM_MPgoHome(strPort, nAddr, nType);
if (rc != 0) {
    //执行失败的处理(Execution failure processing)
}
/*---------------无刷 Brushless----------*/
//堵转找寻原点方式(The way to find the home by blocking)
#include "NiMotionMotorSDK.h"
int nAddr = 1;
string strPort = "COM1";
int nType = 37;//原点回归方式(Go home method)
int rc = NiM_MPwriteParam(strPort, nAddr, 0xD5, 2, 15); //负限位开关(Negative limit switch)
if (rc != 0) {
    //执行失败的处理(Execution failure processing)
}
rc = NiM_MPwriteParam(strPort, nAddr, 0xD6, 2, 0); //低电平有效(Active low)
if (rc != 0) {
    //执行失败的处理(Execution failure processing)
}
rc = NiM_MPwriteParam(strPort, nAddr, 0x419, 4, 1000); //设置寻找原点信号速度(Set the
speed of finding the origin signal)
if (rc != 0) {
    //执行失败的处理(Execution failure processing)
}
rc = NiM_MPwriteParam(strPort, nAddr, 0x 41B, 4, 200000); //设置回零加速度(Set zero
acceleration)
if (rc != 0) {
    //执行失败的处理(Execution failure processing)
}
rc = NiM_MPpowerOn(strPort, nAddr);   //使能(Enable)
if (rc != 0) {
    //执行失败的处理(Execution failure processing)
}
rc = NiM_MPgoHome(strPort, nAddr, nType);
if (rc != 0) {
    //执行失败的处理(Execution failure processing)
}
```

2.6.31 NiM_MPpowerOn

描述：此函数用以电机使能。

Description: This function is used to enable the motor.

```
int NiM_MPpowerOn(const char* strPort, int nAddr);
```

参数(parameter)：

strPort

需要操作的串口名（The name of the serial port to be operated）

nAddr

电机地址（Motor address）。

返回值：0 成功，其它表示错误码。
Return value: 0 success, others indicate error code.
示例(Example)：

```
#include "NiMotionMotorSDK.h"
string strPort = "COM1";
int nAddr = 1;
int rc = NiM_MPpowerOn(strPort, nAddr);
if (rc != 0) {
    //执行失败的处理(Execution failure processing)
}
```

2.6.32 NiM_MPpowerOff

描述：此函数用以电机脱机。
Description: This function is used to offline the motor.
int NiM_MPpowerOff(const char* strPort, int nAddr);
参数(parameter)：
strPort
需要操作的串口名（The name of the serial port to be operated）
nAddr
电机地址（Motor address）。
返回值：0 成功，其它表示错误码。
Return value: 0 success, others indicate error code.
示例(Example)：

```
#include "NiMotionMotorSDK.h"
string strPort = "COM1";
int nAddr = 1;
int rc = NiM_MPpowerOff(strPort, nAddr);
if (rc != 0) {
    //执行失败的处理(Execution failure processing)
}
```

2.6.33 NiM_MPstop

描述：此函数用以电机停止当前动作。
Description: This function is used to stop the current action of the motor.
int NiM_MPstop(const char* strPort, int nAddr);
参数(parameter)：
strPort
需要操作的串口名（The name of the serial port to be operated）
nAddr
电机地址（Motor address）。
返回值：0 成功，其它表示错误码。
Return value: 0 success, others indicate error code.
示例(Example)：

```
#include "NiMotionMotorSDK.h"
string strPort = "COM1";
int nAddr = 1;
int rc = NiM_MPstop(strPort, nAddr);
if (rc != 0) {
    //执行失败的处理(Execution failure processing)
}
```

2.6.34 NiM_MPfastStop

描述：此函数用以电机急停。
Description: This function is used for emergency stop of the motor.
int NiM_MPfastStop(const char* strPort, int nAddr);

参数(parameter)：
strPort
需要操作的串口名（The name of the serial port to be operated）
nAddr
电机地址（Motor address）。
返回值：0 成功，其它表示错误码。
Return value: 0 success, others indicate error code.
示例(Example)：

```
#include "NiMotionMotorSDK.h"
string strPort = "COM1";
int nAddr = 1;
int rc = NiM_MPfastStop(strPort, nAddr);
if (rc != 0) {
    //执行失败的处理(Execution failure processing)
}
```

2.7 广播 Broadcast

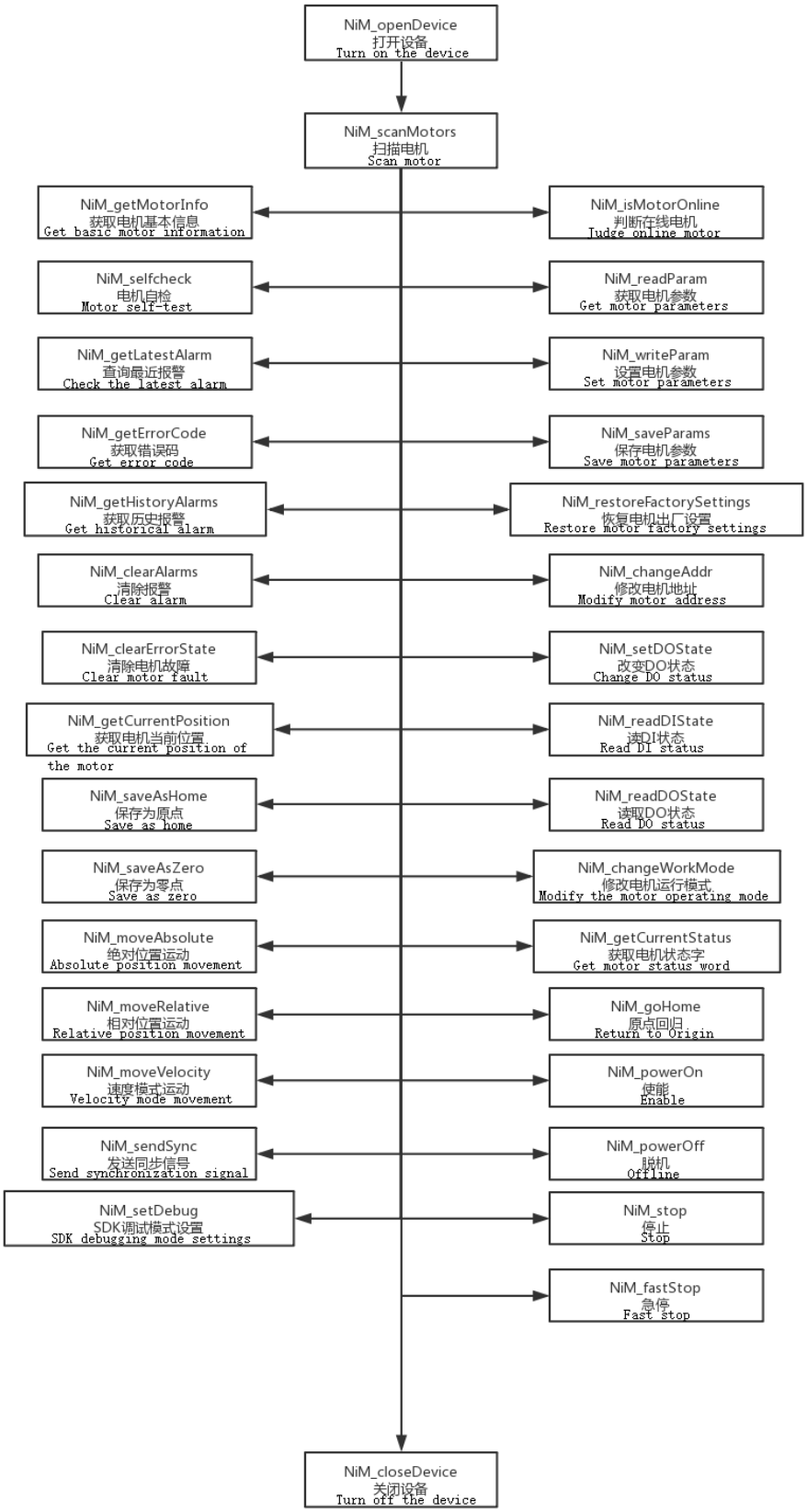
NiMotionModbusSDK 支持 Modbus 广播报文，将电机地址（Motor address）设为 0 即可发送广播报文。
NiMotionModbusSDK supports Modbus broadcast messages. Set the motor address to 0 to send broadcast messages.

示例(Example)：

```
#include "NiMotionMotorSDK.h"
int nAddr = 0;
int rc = NiM_writeParam(nAddr, 0x51, 2, 0x06);    //向所有电机发送 06 控制字(Send 06 control word to all motors)
if (rc != 0) {
    //执行失败的处理(Execution failure processing)
}
```

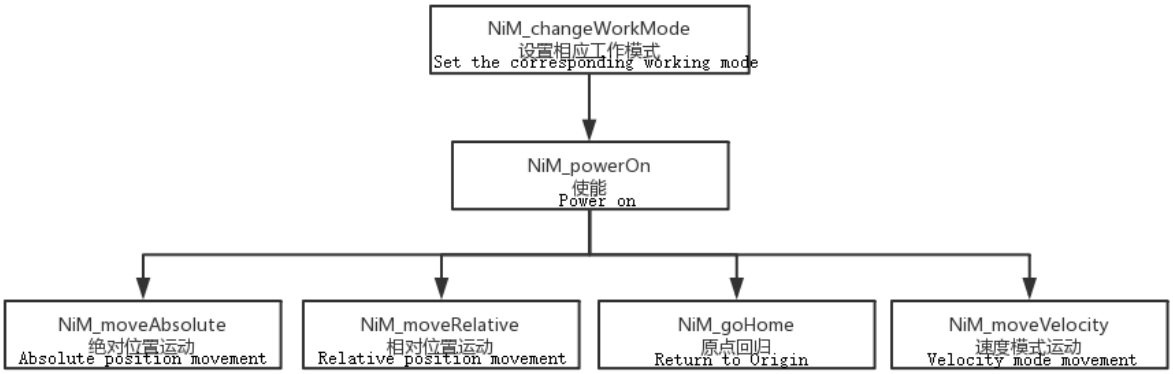
2.8 函数调用方法 Function call method

2.8.1 通信函数与其他函数的关系 The relationship between communication functions and other functions



2.8.2 位置模式、速度模式、原点回归模式下的函数关系

Function relationship in position mode, speed mode, and origin return mode



3 附录 Appendix

表 3.1 报警列表
Table 3.1 Alarm list

步进电机 Stepper motor		无刷电机 Brushless Motor	
0x0000	无报警 No alarm	0x2300	电机过流 Motor overcurrent
0x2300	过流保护 Overcurrent protection	0x4012311	电机过载 Motor overload
0x3110	电源过压 Power supply overvoltage	0x3002312	电机堵转 Motor blocked
0x3120	电源欠压 Power supply undervoltage	0x13210	电源过电压 Power supply overvoltage
0x4310	过热报警 Overheating alarm	0x1013220	电源欠电压 Power supply undervoltage
0x5530	EEPORM 读写故障 EEPORM read and write failure	0x14210	温度过高报警 High temperature alarm
0x7121	堵转报警 Stall alarm	0x14220	温度过低报警 Low temperature alarm
0x7122	失速报警 Stall alarm	0x5080	驱动器故障 Drive failure
0x8210	报文数据长度错误 Packet data length error	0x5540	Flash 操作故障 Flash operation failure
0x8612	限位报警 Limit alarm	0x5541	Flash 初始化故障 Flash initialization failure
0xFF00	过热关机 Thermal shutdown	0x4005542	Flash 校验错误警告 Flash verification error warning
0xFF01	错误的命令 Wrong command	0x5543	Flash 用户区无参数 Flash user area without parameters
0xFF02	不能执行的命令 Unexecuted command	0x6000	硬件初始化故障 Hardware initialization failure
0xFF03	EEPORM 自检故障 EEPORM self-check failure	0x1632	0 参数设置错误 0Parameter setting error
0xFF04	UART 自检故障 UART self-test failure	0x6321	注册故障 Registration failure
0xFF06	编码器自检故障 Encoder self-check failure	0x7305	Z 脉冲故障 Z pulse fault
0xFF07	电机驱动部分自检故障 Motor drive part self-check failure	0x7306	编码器故障 Encoder failure
0xFF08	码器读数据帧错误 Encoder reading data frame error	0x4017307	编码器警告 Encoder warning

0xFF09	编码器校验错误 Encoder check error	0x17310	超速 Overspeed
0xFF0A	编码器指令错误 Encoder instruction error	0x4017501	Modbus 通信中的非法功能码 Illegal function codes in Modbus communication
0xFF0B	编码器帧错误 Encoder frame error	0x4017502	Modbus 通信中的非法地址 Illegal address in Modbus communication
0xFF0C	编码器磁场过高 Encoder magnetic field is too high	0x4017503	Modbus 通信中的非法数据值 Illegal data value in Modbus communication
0xFF0D	编码器磁场过低 Encoder magnetic field is too low	0x4017505	Modbus 通信中的确认 Confirmation in Modbus communication
0xFF0E	超负限位报警 Encoder magnetic field is too low	0x4017506	Modbus 通信中的从设备忙 Slave device in Modbus communication is busy
0xFF0F	超正限位报警 Over-positive limit alarm	0x401750C	Modbus 通信中的同步报文请求数据数据大于映射总数据 The synchronous message request data data in Modbus communication is greater than the total mapped data
0xFF10	通讯波特率自适应失败报警 Communication baud rate adaptive failure alarm	0x401750D	Modbus 通信中的同步报文请求数据个数对应映射数据不相等 The number of synchronous message request data corresponding to the mapping data in Modbus communication is not equal
0xFF0D	编码器磁场过低 Encoder magnetic field is too low	0x401750E	Modbus 通信中的同步功能下单播报文节点地址错误 The unicast message node address is wrong under the synchronization function in Modbus communication
0xFF0E	超负限位报警 Over-negative limit alarm	0x401750F	Modbus 通信中写只读参数 Write read-only parameters in Modbus communication
0xFF0F	超正限位报警 Over-positive limit alarm	0x8610	原点回归超时 Home return timeout
0xFF10	通讯波特率自适应失败报警 Communication baud rate adaptive failure alarm	0x8611	位置超差 Out of tolerance
		0x3018613	软件限位错误 Software limit error
		0x3018614	限位开关错误 Limit switch error
		0x4018615	曲线规划计算错误 Curve planning calculation error
		0x18616	目标位置溢出 Target overflow
		0x5018617	曲线规划参数过小 Curve planning parameters are too small
		0xFF01	电机参数识别故障 Motor parameter identification failure

- 本手册的全部内容或部分内容禁止擅自转载、拷贝。
  All or part of the contents of this manual are forbidden to reprint or copy without authorization.
- 产品性能、规格及外观可能因为改进，会在不经预先通知的情况下发生变化，敬请谅解。
  Product performance, specifications and appearance may change without prior notice due to improvements, please understand.
- 我们力求使手册的内容尽可能正确，如果您发现有什么问题或错误、遗漏之处，请与北京立迈胜控制技术有限责任公司联系。
  We strive to make the contents of the manual as accurate as possible. If you find any problems, errors, or omissions, please contact Beijing NiMotion Control Technology Co., Ltd.

北京立迈胜控制技术有限责任公司
Beijing NiMotion Control Technology Co., Ltd.
北京市大兴区金星路 12 号院 3 号楼
Building 3, Yard 12, Jinxing Road, Daxing District, Beijing
邮编 Postcode：102628
电话 Tel：(010)60213882　　传真 Fax：(010)60213882
邮箱 Email：nimotion@nimotion.com
http://www.nimotion.com