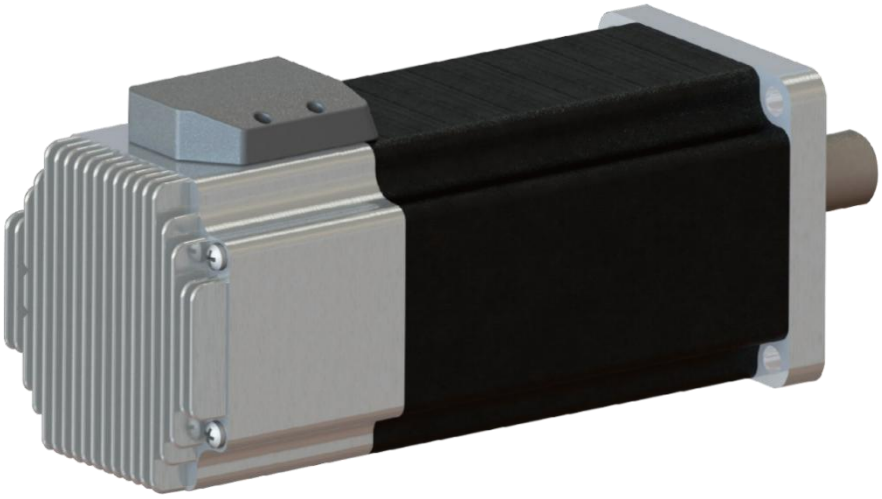
NiMServoSDK_MM 使用说明书

版 本 号：D



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

目 录

1 关于手册

1.1 概述

本手册用于说明北京立迈胜控制技术有限公司所生产的 NiMServoSDK_MM 的使用说明。

1.2 版本信息

手册版本	日期	修改记录
A	2022/8/8	创建
B	2023/3/22	修改手册部分函数描述问题
C	2023/10/10	新增若干函数，修改部分函数使用说明，新增 SocketCAN 注意事项
D	2024/12/5	修改部分函数文字描述

2 SDK 适用场景

2.1 通信协议支持

表 2-1

序号	通信协议
0	CANopen
1	EtherCAT
2	Modbus

2.2 电机支持

表 2-2

序号	电机类型
1	PMM-XXXXX-CANopen
2	BLM-XXXXX-CANopen
3	STM-XXXXX-CANopen-M
4	STM-XXXXX-CANopen
5	PMM-XXXXX-EtherCAT
6	BLM-XXXXX-Modbus
7	STM-XXXXX-Modbus-M

2.3 平台支持

表 2-3

序号	硬件平台	软件平台
1	X86_32	Windows
2	X86_64	Windows
3	X86_64	Linux
4	ARMv7	Linux
5	ARMv8	Linux

注：Linux 系统二进制文件所需动态库：
libm.so.6
libpthread.so.0
libdl.so.2
libc.so.6
libgcc_s.so.1
libstdc++.so.6
libQt5Core.so.5

2.4 CAN 设备支持

表 2-4

序号	CAN 设备	Windows X86_32/64	Linux X86_64	ARMv7	ARMv8
1	NiMotion USBCAN	支持	支持	支持	支持
2	NiMotion TCPCAN	支持	支持	支持	支持
3	Ixxat USBCAN	支持			
4	周立功 USBCAN	支持			
5	Linux Socket CAN			支持	支持

3 函数接口

为便于在不同编程语言中调用，函数接口定义为标准 C 函数接口。返回值定义为 int 数据类型，成功返回 0，失败返回错误码，具体返回参数见附录 5.1.1 ServoSDK_Error 枚举内容。函数接口详细定义如下：

3.1 SDK 初始化

3.1.1 Nim_init

函数原型: int Nim_init(const char* strSdkPath);

描述：SDK 初始化，
参数：strSdkPath 库路径，与设置的系统环境一致

返回值：0 成功；其它失败

3.1.2 Nim_create_master

函数原型：int Nim_create_master(int nCommType, unsigned int* handle);

描述：创建主站对象。注：SDK 支持多主站运行，最多可以同时添加 16 个主站。

参数：int nCommType 通信方式：0 CANopen；1 EtherCAT；2 Modbus

handle 输出参数，成功时返回创建的主站对象句柄

返回值：0 成功；其他 失败

3.1.3 Nim_destroy_master

函数原型：int Nim_destroy_master(unsigned int handle);

描述：销毁主站对象。

参数：handle 由 Nim_create_master 函数创建的主站对象句柄

返回值：0 成功；其他 失败

3.1.4 Nim_clean

函数原型: int Nim_clean();

描述：SDK 反初始化

参数：无

返回值：0 成功；其它失败

3.2 主站操作

3.2.1 Nim_master_run

函数原型: int Nim_master_run(unsigned int hMaster, const char* conn_str);

描述：启动通信主站

参数：hMaster 主站对象句柄

conn_str 连接字符串，json 格式

具体内容与通信方式有关，相关参数见附录 5.2 运行通信主站注意事项的内容

返回值：0 成功；其它失败

3.2.2 Nim_master_stop

函数原型: int Nim_master_stop(unsigned int hMaster);

描述：停止通信主站

参数：hMaster 主站对象句柄

返回值：0 成功；其它失败

3.2.3 Nim_master_changeToPreOP

函数原型: int Nim_master_changeToPreOP(unsigned int hMaster);

描述：切换通信主站到 PreOP 状态，PreOP 状态下不进行 PDO 传输

参数：hMaster 主站对象句柄

3.2.4 Nim_master_changeToOP

函数原型: int Nim_master_changeToOP(unsigned int hMaster);

描述：切换通信主站到 OP 状态，并开始 PDO 传输

参数：hMaster 主站对象句柄

返回值：0 成功；其它失败

3.3 执行日志

3.3.1 Nim_setLogFlags

函数原型: void Nim_setLogFlags(int nFlags);

描述：设置日志输出标志

参数： nFlags bit0=1 输出到控制台；bit1=1 输出到文件

3.3.2 Nim_getLogFlags

函数原型: int Nim_getLogFlags ();

描述：获取日志输出标志

返回值：bit0=1 输出到控制台；bit1=1 输出到文件

3.4 电机扫描和初始化

3.4.1 Nim_scan_nodes

函数原型: int Nim_scan_nodes(unsigned int hMaster,int from, int to);

描述：扫描在线的从站

参数：hMaster 主站对象句柄

from 起始地址；

to 结束地址；

返回值：0 成功；其它失败

3.4.2 Nim_is_online

函数原型: int Nim_is_online(unsigned int hMaster,int nodeId);

描述：判断从站是否在线

参数：hMaster 主站对象句柄

nodeId 从站地址

返回值：1 在线；0 不在线

3.4.3 Nim_load_params

函数原型: int Nim_load_params(unsigned int hMaster,int nodeId, const char* db_name);

描述：加载从站参数表

参数：hMaster 主站对象句柄

nodeId 从站地址

db_name 参数表文件名

不同型号电机对应不同数据库，详细关系见附录 5.3 数据库选择内容。

返回值：0 成功；其它失败

3.4.4 Nim_read_PDOConfig

函数原型: int Nim_read_PDOConfig(unsigned int hMaster,int nodeId);

描述：读取从站的 PDO 配置

参数：hMaster 主站对象句柄

nodeId 从站地址

返回值：0 成功；其它失败

3.5 电机通用控制

3.5.1 Nim_set_controlWord

函数原型: int Nim_set_controlWord(unsigned int hMaster,int nodeId, unsigned short cw, int bSDO);

描述：设置控制字

参数：hMaster 主站对象句柄

nodeId 从站地址

cw 控制字

bSDO 是否通过 SDO 设置：1 通过 SDO；0 通过 PDO

返回值：0 成功；其它失败

3.5.2 Nim_get_statusWord

函数原型: int Nim_get_statusWord(unsigned int hMaster,int nodeId, unsigned short *sw, int bSDO);

描述：读取状态字

参数：hMaster 主站对象句柄

nodeId 从站地址

sw 输出参数，成功时返回状态字

bSDO 是否通过 SDO 读取：1 通过 SDO；0 通过 PDO

返回值：0 成功；其它失败

3.5.3 Nim_power_on

函数原型: int Nim_power_on(unsigned int hMaster,int nodeId, int bSDO);

描述：电机使能

参数：hMaster 主站对象句柄

nodeId 从站地址

bSDO 是否通过 SDO 控制：1 通过 SDO；0 通过 PDO

返回值：0 成功；其它失败

3.5.4 Nim_power_off

函数原型: int Nim_power_off(unsigned int hMaster,int nodeId, int bSDO);

描述：电机脱机

参数：hMaster 主站对象句柄

nodeId 从站地址

bSDO 是否通过 SDO 控制：1 通过 SDO；0 通过 PDO

返回值：0 成功；其它失败

3.5.5 Nim_set_workMode

函数原型: int Nim_set_workMode(unsigned int hMaster,int nodeId, int mode, int bSDO);

描述：设置电机工作模式

参数：hMaster 主站对象句柄

nodeId 从站地址

mode 工作模式，见附录 5.1.3 ServoWorkMode 的枚举内容

bSDO 是否通过 SDO 设置：1 通过 SDO；0 通过 PDO

返回值：0 成功；其它失败

3.5.6 Nim_get_workModeDisplay

函数原型: int Nim_get_workModeDisplay(unsigned int hMaster,int nodeId, int *mode, int bSDO);

描述：读取电机工作模式显示值

参数：hMaster 主站对象句柄

nodeId 从站地址

mode 输出参数，成功时返回工作模式显示值

bSDO 是否通过 SDO 读取：1 通过 SDO；0 通过 PDO

返回值：0 成功；其它失败

3.6 轮廓速度模式控制（PV）

3.6.1 Nim_forward

函数原型: int Nim_forward(unsigned int hMaster,int nodeId, double fVelocity, int bSDO);

描述：控制电机在轮廓速度模式下正转

参数：hMaster 主站对象句柄

nodeId 从站地址

fVelocity 目标速度（用户单位/秒）

bSDO 是否通过 SDO 控制：1 通过 SDO；0 通过 PDO

返回值：0 成功；其它失败

3.6.2 Nim_backward

函数原型: int Nim_backward(unsigned int hMaster,int nodeId, double fVelocity, int bSDO);

描述：控制电机在轮廓速度模式下反转

参数：hMaster 主站对象句柄

nodeId 从站地址

fVelocity 目标速度（用户单位/秒）

bSDO 是否通过 SDO 控制：1 通过 SDO；0 通过 PDO

返回值：0 成功；其它失败

3.6.3 Nim_set_targetVelocity

函数原型: int Nim_set_targetVelocity(unsigned int hMaster,int nodeId, double fVelocity, int bSDO);

描述：设置电机目标速度

参数：hMaster 主站对象句柄

nodeId 从站地址

fVelocity 目标速度（用户单位/秒）

bSDO 是否通过 SDO 设置：1 通过 SDO；0 通过 PDO

返回值：0 成功；其它失败

3.6.4 Nim_get_currentVelocity

函数原型: int Nim_get_currentVelocity(unsigned int hMaster,int nodeId, double *fVelocity, int bSDO);

描述：读取电机当前速度（读取 6069 对象）

参数：hMaster 主站对象句柄

nodeId 从站地址

fVelocity 输出参数，成功时返回当前速度（用户单位/秒）

bSDO 是否通过 SDO 读取：1 通过 SDO；0 通过 PDO

返回值：0 成功；其它失败

3.6.5 Nim_get_currentVelocity2

函数原型: int Nim_get_currentVelocity2(unsigned int hMaster,int nodeId, double *fVelocity, int bSDO);

描述：读取电机当前速度（读取 606C 对象）

参数：hMaster 主站对象句柄

nodeId 从站地址

fVelocity 输出参数，成功时返回当前速度（用户单位/秒）

bSDO 是否通过 SDO 读取：1 通过 SDO；0 通过 PDO

返回值：0 成功；其它失败

3.6.6 Nim_get_currentMotorSpeed

函数原型: int Nim_get_currentMotorSpeed(unsigned int hMaster,int nodeId, int *speed, int bSDO);

描述：读取当前电机速度

参数：hMaster 主站对象句柄

nodeId 从站地址

speed 输出参数，成功时返回当前电机速度（rpm）

bSDO 是否通过 SDO 读取：1 通过 SDO；0 通过 PDO

返回值：0 成功；其它失败

3.6.7 Nim_set_profileAccel

函数原型: int Nim_set_profileAccel(unsigned int hMaster,int nodeId, double accel);

描述：设置电机轮廓加速度

参数：hMaster 主站对象句柄

nodeId 从站地址

accel 轮廓加速度（用户单位/秒 ^{2}）

返回值：0 成功；其它失败

3.6.8 Nim_get_profileAccel

函数原型: int Nim_get_profileAccel(unsigned int hMaster,int nodeId, double *accel);

描述：获取电机轮廓加速度

参数：hMaster 主站对象句柄

nodeId 从站地址

accel 输出参数，成功时返回轮廓加速度（用户单位/秒 ^{2}）

返回值：0 成功；其它失败

3.6.9 Nim_set_profileDecel

函数原型: int Nim_set_profileDecel(unsigned int hMaster,int nodeId, double decel);

描述：设置电机轮廓减速度

参数：hMaster 主站对象句柄

nodeId 从站地址

decel 轮廓减速度（用户单位/秒 ^{2}）

返回值：0 成功；其它失败

3.6.10 Nim_get_profileDecel

函数原型: int Nim_get_profileDecel(unsigned int hMaster,int nodeId, double *decel);

描述：获取电机轮廓减速度

参数：hMaster 主站对象句柄

nodeId 从站地址

decel 输出参数，成功时返回轮廓减速度（用户单位/秒 ^{2}）

返回值：0 成功；其它失败

3.7 速度模式控制（VM）

3.7.1 Nim_set_vmTargetSpeed

函数原型: int Nim_set_vmTargetSpeed(unsigned int hMaster,int nodeId, int speed, int bSDO);

描述：设置电机 VM 模式下的目标速度

参数：hMaster 主站对象句柄

nodeId 从站地址

speed 目标速度（rpm）

bSDO 是否通过 SDO 设置：1 通过 SDO；0 通过 PDO

返回值：0 成功；其它失败

3.7.2 Nim_get_vmCurrentSpeed

函数原型: int Nim_get_vmCurrentSpeed(unsigned int hMaster,int nodeId, int *speed, int bSDO);

描述：获取电机 VM 模式下的当前速度

参数：hMaster 主站对象句柄

nodeId 从站地址

speed 输出参数，成功时返回 VM 模式下的当前速度（rpm）

bSDO 是否通过 SDO 设置：1 通过 SDO；0 通过 PDO

返回值：0 成功；其它失败

3.7.3 Nim_set_vmAccel

函数原型: int Nim_set_vmAccel(unsigned int hMaster,int nodeId, unsigned int deltaV, unsigned int deltaT);

描述：设置 VM 模式下的加速度

参数：hMaster 主站对象句柄

nodeId 从站地址

deltaV 速度变化量（rpm）

deltaT 时间变化量（秒）

加速度 = deltaV/ deltaT

返回值：0 成功；其它失败

3.7.4 Nim_get_vmAccel

函数原型: int Nim_get_vmAccel(unsigned int hMaster,int nodeId, double *accel);

描述：获取 VM 模式下的加速度

参数：hMaster 主站对象句柄

nodeId 从站地址

accel 输出参数，成功时返回加速度（rpm/秒）

返回值：0 成功；其它失败

3.7.5 Nim_set_vmDecel

函数原型: int Nim_set_vmDecel(unsigned int hMaster,int nodeId, unsigned int deltaV, unsigned int deltaT);

描述：设置 VM 模式下的减速度

参数：hMaster 主站对象句柄

nodeId 从站地址

deltaV 速度变化量（rpm）

deltaT 时间变化量（秒）

减速度 = deltaV/ deltaT

返回值：0 成功；其它失败

3.7.6 Nim_get_vmDecel

函数原型: int Nim_get_vmDecel(unsigned int hMaster,int nodeId, double *decel);

描述：获取 VM 模式下的减速度

参数：hMaster 主站对象句柄

nodeId 从站地址

decel 输出参数，成功时返回减速度（rpm/秒）

返回值：0 成功；其它失败

3.7.7 Nim_set_vmSpeedLimit

函数原型: int Nim_set_vmSpeedLimit(unsigned int hMaster,int nodeId, unsigned int minSpeed, unsigned int maxSpeed);

描述：设置 VM 模式下的速度限制值

参数：hMaster 主站对象句柄

nodeId 从站地址

minSpeed 最小速度（rpm）

maxSpeed 最大速度（rpm）

返回值：0 成功；其它失败

3.7.8 Nim_get_vmSpeedLimit

函数原型: int Nim_get_vmSpeedLimit(unsigned int hMaster,int nodeId, unsigned int *minSpeed, unsigned int *maxSpeed);

描述：获取 VM 模式下的速度限制值

参数：hMaster 主站对象句柄

nodeId 从站地址

minSpeed 输出参数，成功时返回最小速度（rpm）

maxSpeed 输出参数，成功时返回最大速度（rpm）

返回值：0 成功；其它失败

3.8 轮廓位置控制（PP）

3.8.1 Nim_moveAbsolute

函数原型: int Nim_moveAbsolute(unsigned int hMaster,int nodeId, double position, int bChangeImmediatly, int bSDO);

描述：控制电机在轮廓位置模式下执行绝对位置运动

参数：hMaster 主站对象句柄

nodeId 从站地址

position 目标位置（用户单位）

bChangeImmediatly 是否立即更新：1 立即更新；0 非立即更新

bSDO 是否通过 SDO 控制：1 通过 SDO；0 通过 PDO

返回值：0 成功；其它失败

3.8.2 Nim_moveRelative

函数原型: int Nim_moveRelative(unsigned int hMaster,int nodeId, double distance, int bChangeImmediatly, int bSDO);

描述：控制电机在轮廓位置模式下执行相对位置运动

参数：hMaster 主站对象句柄

nodeId 从站地址

distance 目标位置（用户单位）

bChangeImmediatly 是否立即更新：1 立即更新；0 非立即更新

bSDO 是否通过 SDO 控制：1 通过 SDO；0 通过 PDO

返回值：0 成功；其它失败

3.8.3 Nim_set_targetPosition

函数原型: int Nim_set_targetPosition(unsigned int hMaster,int nodeId, double position, int bSDO);

描述：设置目标位置(607A)

参数：hMaster 主站对象句柄

nodeId 从站地址

position 目标位置（用户单位）

bSDO 是否通过 SDO 设置：1 通过 SDO；0 通过 PDO

返回值：0 成功；其它失败

3.8.4 Nim_get_currentPosition

函数原型: int Nim_get_currentPosition(unsigned int hMaster,int nodeId, double *position, int bSDO);

描述：通过 6064 获取当前位置

参数：hMaster 主站对象句柄

nodeId 从站地址

position 输出参数，成功时返回目标位置（用户单位）

bSDO 是否通过 SDO 设置：1 通过 SDO；0 通过 PDO

返回值：0 成功；其它失败

3.8.5 Nim_set_profileVelocity

函数原型: int Nim_set_profileVelocity(unsigned int hMaster,int nodeId, double fVelocity);

描述：设置轮廓速度(6081)

参数：hMaster 主站对象句柄

nodeId 从站地址

fVelocity 轮廓速度（用户单位/秒）

返回值：0 成功；其它失败

3.8.6 Nim_get_profileVelocity

函数原型: int Nim_get_profileVelocity(unsigned int hMaster,int nodeId, double * fVelocity);

描述：获取轮廓速度(6081)

参数：hMaster 主站对象句柄

nodeId 从站地址

fVelocity 输出参数，成功时返回轮廓速度（用户单位/秒）

返回值：0 成功；其它失败

3.9 转矩模式控制（PT）

3.9.1 Nim_set_targetTorque

函数原型: int Nim_set_targetTorque(unsigned int hMaster,int nodeId, int torque, int bSDO);

描述：设置目标转矩

参数：hMaster 主站对象句柄

nodeId 从站地址

torque 目标转矩（0.001 倍额定转矩）

bSDO 是否通过 SDO 设置：1 通过 SDO；0 通过 PDO

返回值：0 成功；其它失败

3.9.2 Nim_get_currentTorque

函数原型: int Nim_get_currentTorque(unsigned int hMaster,int nodeId, int *torque, int bSDO);

描述：获取当前转矩

参数：hMaster 主站对象句柄

nodeId 从站地址

torque 输出参数，成功时返回当前转矩（0.001 倍额定转矩）

bSDO 是否通过 SDO 读取：1 通过 SDO；0 通过 PDO

返回值：0 成功；其它失败

3.9.3 Nim_set_PT_TorqueRamp

函数原型: int Nim_set_PT_TorqueRamp(unsigned int hMaster, int nodeId, unsigned int torqueRamp);

描述：设置转矩斜坡

参数：hMaster 主站对象句柄

nodeId 从站地址

torque torqueRamp 转矩斜坡：0 无斜坡；>0 每秒钟增加的转矩值（单位：0.001 倍额定转矩）

返回值：0 成功；其它失败

3.9.4 Nim_get_PT_TorqueRamp

函数原型: int Nim_get_PT_TorqueRamp(unsigned int hMaster, int nodeId, unsigned int* torqueRamp);

描述：获取转矩斜坡

参数：hMaster 主站对象句柄

nodeId 从站地址

torque torqueRamp 转矩斜坡：0 无斜坡；>0 每秒钟增加的转矩值（单位：0.001 倍额定转矩）

返回值：0 成功；其它失败

3.9.5 Nim_set_PT_SpeedLimit

函数原型: int Nim_set_PT_SpeedLimit(unsigned int hMaster, int nodeId, unsigned short FwrSpeedLimit, unsigned short BwrSpeedLimit);

描述：设置轮廓转矩模式下速度限制(2007:10h、2007:11h)

参数：hMaster 主站对象句柄

nodeId 从站地址

FwrSpeedLimit 正向速度限制（单位 rpm）

BwrSpeedLimit 反向速度限制（单位 rpm）

返回值：0 成功；其它失败

3.9.6 Nim_get_PT_SpeedLimit

函数原型: int Nim_get_PT_SpeedLimit(unsigned int hMaster, int nodeId, unsigned short* FwrSpeedLimit, unsigned short* BwrSpeedLimit);

描述：获取轮廓转矩模式下速度限制(2007:10h、2007:11h)

参数：hMaster 主站对象句柄

nodeId 从站地址

FwrSpeedLimit 正向速度限制（单位 rpm）

BwrSpeedLimit 反向速度限制（单位 rpm）

返回值：0 成功；其它失败

3.10 原点回归控制（HM）

3.10.1 Nim_goHome

函数原型: int Nim_goHome(unsigned int hMaster,int nodeId, int bSDO);

描述：原点回归

参数：hMaster 主站对象句柄

nodeId 从站地址

bSDO 是否通过 SDO 控制：1 通过 SDO；0 通过 PDO

返回值：0 成功；其它失败

3.10.2 Nim_set_homeType

函数原型: int Nim_set_homeType(unsigned int hMaster, int nodeId, unsigned char type);

描述：设置原点回归方式

参数：hMaster 主站对象句柄

nodeId 从站地址

Type 原点回归方式，参照电机通信手册设置

返回值：0 成功；其它失败

3.10.3 Nim_get_homeType

函数原型: int Nim_get_homeType(unsigned int hMaster, int nodeId, unsigned char* type);

描述：获取原点回归方式

参数：hMaster 主站对象句柄

nodeId 从站地址

Type 原点回归方式

返回值：0 成功；其它失败

3.10.4 Nim_set_homeOffset

函数原型: int Nim_set_homeOffset(unsigned int hMaster,int nodeId, double offset);

描述：设置电机原点偏移

参数：hMaster 主站对象句柄

nodeId 从站地址

offset 原点偏移（用户单位）

返回值：0 成功；其它失败

3.10.5 Nim_get_homeOffset

函数原型: int Nim_get_homeOffset(unsigned int hMaster,int nodeId, double *offset);

描述：获取电机原点偏移

参数：hMaster 主站对象句柄

nodeId 从站地址

offset 输出参数，成功时返回原点偏移（用户单位）

返回值：0 成功；其它失败

3.10.6 Nim_set_goHome_velocity

函数原型: int Nim_set_goHome_velocity(unsigned int hMaster,int nodeId, double velocity1, double velocity2);

描述：设置电机原点回归速度

参数：hMaster 主站对象句柄

nodeId 从站地址

velocity1 寻找开关速度（用户单位/秒）

velocity2 寻找原点速度（用户单位/秒）

返回值：0 成功；其它失败

3.10.7 Nim_get_goHome_velocity

函数原型: int Nim_get_goHome_velocity(unsigned int hMaster,int nodeId, double *velocity1, double *velocity2);

描述：获取电机原点回归速度

参数：hMaster 主站对象句柄

nodeId 从站地址

velocity1 输出参数，成功时返回寻找开关速度（用户单位/秒）

velocity2 输出参数，成功时返回寻找原点速度（用户单位/秒）

返回值：0 成功；其它失败

3.10.8 Nim_set_goHome_accel

函数原型: int Nim_set_goHome_accel(unsigned int hMaster,int nodeId, double accel);

描述：设置原点回归加速度

参数：hMaster 主站对象句柄

nodeId 从站地址

accel 原点回归加速度（用户单位/秒 ^{2}）

返回值：0 成功；其它失败

3.10.9 Nim_get_goHome_accel

函数原型: int Nim_get_goHome_accel(unsigned int hMaster,int nodeId, double *accel);

描述：获取原点回归加速度

参数：hMaster 主站对象句柄

nodeId 从站地址

accel 输出参数，成功时返回原点回归加速度（用户单位/秒 ^{2}）

返回值：0 成功；其它失败

3.11 限制参数

3.11.1 Nim_set_posLimit

函数原型: int Nim_set_posLimit(unsigned int hMaster,int nodeId, double minPos, double maxPos);

描述：设置电机位置限制

参数：hMaster 主站对象句柄

nodeId 从站地址

minPos 最小极限位置（用户单位）

maxPos 最大极限位置（用户单位）

返回值：0 成功；其它失败

3.11.2 Nim_get_posLimit

函数原型: int Nim_get_posLimit(unsigned int hMaster,int nodeId, double *minPos, double *maxPos);

描述：获取电机位置限制

参数：hMaster 主站对象句柄

nodeId 从站地址

minPos 输出参数，成功时返回最小极限位置（用户单位）

maxPos 输出参数，成功时返回最大极限位置（用户单位）

返回值：0 成功；其它失败

3.11.3 Nim_set_maxVelocity

函数原型: int Nim_set_maxVelocity (unsigned int hMaster,int nodeId, double velocity);

描述：设置轴最大速度

参数：hMaster 主站对象句柄

nodeId 从站地址

velocity 最大速度（用户单位/秒）

返回值：0 成功；其它失败

3.11.4 Nim_get_maxVelocity

函数原型: int Nim_get_maxVelocity (unsigned int hMaster,int nodeId, double *velocity);

描述：获取轴最大速度

参数：hMaster 主站对象句柄

nodeId 从站地址

velocity 输出参数，成功时返回最大速度（用户单位/秒）

返回值：0 成功；其它失败

3.11.5 Nim_set_maxMotorSpeed

函数原型: int Nim_set_maxMotorSpeed(unsigned int hMaster,int nodeId, unsigned int speed);

描述：设置电机最大转速

参数：hMaster 主站对象句柄

nodeId 从站地址

speed 最大转速（rpm）

返回值：0 成功；其它失败

3.11.6 Nim_get_maxMotorSpeed

函数原型: int Nim_get_maxMotorSpeed(unsigned int hMaster,int nodeId, unsigned int *speed);

描述：获取电机最大转速

参数：hMaster 主站对象句柄

nodeId 从站地址

speed 输出参数，成功时返回最大转速（rpm）

返回值：0 成功；其它失败

3.11.7 Nim_set_maxTorque

函数原型: int Nim_set_maxTorque(unsigned int hMaster,int nodeId, unsigned int torque);

描述：设置电机最大转矩

参数：hMaster 主站对象句柄

nodeId 从站地址

torque 最大转矩（0.001 倍额定转矩）

返回值：0 成功；其它失败

3.11.8 Nim_get_maxTorque

函数原型: int Nim_get_maxTorque(unsigned int hMaster,int nodeId, unsigned int *torque);

描述：获取电机最大转矩

参数：hMaster 主站对象句柄

nodeId 从站地址

torque 输出参数，成功时返回最大转矩（0.001 倍额定转矩）

返回值：0 成功；其它失败

3.12 快速停机控制

3.12.1 Nim_fastStop

函数原型: int Nim_fastStop(unsigned int hMaster,int nodeId, int bSDO);

描述：控制电机快速停机

参数：hMaster 主站对象句柄

nodeId 从站地址

bSDO 是否通过 SDO 控制：1 通过 SDO；0 通过 PDO

返回值：0 成功；其它失败

3.12.2 Nim_set_quickStopDecel

函数原型: int Nim_set_quickStopDecel(unsigned int hMaster,int nodeId, double decel);

描述：设置电机快速停机减速度

参数：hMaster 主站对象句柄

nodeId 从站地址

decel 快速停机减速度（用户单位/秒 ^{2}）

返回值：0 成功；其它失败

3.12.3 Nim_get_quickStopDecel

函数原型: int Nim_get_quickStopDecel(unsigned int hMaster,int nodeId, double *decel);

描述：获取电机快速停机减速度

参数：hMaster 主站对象句柄

nodeId 从站地址

decel 输出参数，成功时返回快速停机减速度（用户单位/秒 ^{2}）

返回值：0 成功；其它失败

3.13 IO 控制

3.13.1 Nim_get_DIs

函数原型: int Nim_get_DIs(unsigned int hMaster,int nodeId, unsigned int * nDIs, int bSDO);

描述：获取电机的 DI 输入状态

参数：hMaster 主站对象句柄

nodeId 从站地址

nDIs 输出参数，成功时返回 DI 值

每位表示一路 DI：bit0 表示 DI1；bit1 表示 DI2，以此类推

1 输入为高电平；0 输入为低电平

bSDO 是否通过 SDO 读取：1 通过 SDO；0 通过 PDO

返回值：0 成功；其它失败

3.13.2 Nim_set_DOs

函数原型: int Nim_set_DOs(unsigned int hMaster,int nodeId, unsigned int nDOs, int bSDO);

描述：设置电机的 DO 输出状态

参数：hMaster 主站对象句柄

nodeId 从站地址

nDOs DO 值,每位表示一路 DO：bit0 表示 DO1；bit1 表示 DO2，以此类推
1 输出高电平；0 输出低电平

bSDO 是否通过 SDO 读取：1 通过 SDO；0 通过 PDO

返回值：0 成功；其它失败

3.13.3 Nim_set_VDIs

函数原型: int Nim_set_VDIs(unsigned int hMaster,int nodeId, unsigned int nVDIs);

描述：设置电机的 VDI 状态

参数：hMaster 主站对象句柄

nodeId 从站地址

nVDIs VDI 值,每位表示一路 VDI：bit0 表示 VDI1；bit1 表示 VDI2，以此类推
1 输出高电平；0 输出低电平

返回值：0 成功；其它失败

3.14 用户单位转换

3.14.1 Nim_set_unitsFactor

函数原型: int Nim_set_unitsFactor (unsigned int hMaster,int nodeId, double factor);

描述：设置单位换算系数

参数：hMaster 主站对象句柄

nodeId 从站地址

factor 单位换算系数（电机编码器单位/用户单位）

返回值：0 成功；其它失败

3.14.2 Nim_get_unitsFactor

函数原型: int Nim_get_unitsFactor (unsigned int hMaster,int nodeId, double *factor);

描述：获取单位换算系数

参数：hMaster 主站对象句柄

nodeId 从站地址

factor 输出参数，成功时返回单位换算系数（电机编码器单位/用户单位）

返回值：0 成功；其它失败

3.15 其它接口函数

3.15.1 Nim_set_ipPosition

函数原型: int Nim_set_ipPosition(unsigned int hMaster,int nodeId, double position, int bSDO);

描述：设置插补位置

参数：hMaster 主站对象句柄

nodeId 从站地址

position 插补位置（用户单位）

bSDO 是否通过 SDO 设置：1 通过 SDO；0 通过 PDO

返回值：0 成功；其它失败

3.15.2 Nim_set_ipPeriod

函数原型: int Nim_set_ipPeriod(unsigned int hMaster, int nodeId, unsigned int nPeriodMS);

描述：设置插补周期

参数：hMaster 主站对象句柄

nodeId 从站地址

position nPeriodMS 插补周期（单位：ms）

返回值：0 成功；其它失败

3.15.3 Nim_get_ipPeriod

函数原型: int Nim_get_ipPeriod(unsigned int hMaster, int nodeId, unsigned int* nPeriodMS);

描述：获取插补周期(60C2:01, 60C2:02)

参数：hMaster 主站对象句柄

nodeId 从站地址

nPeriodMS 插补周期（单位：ms）

返回值：0 成功；其它失败

3.15.4 Nim_set_param_value

函数原型: int Nim_set_param_value(unsigned int hMaster,int nodeId, const char* paramNO, unsigned int uiValue, int bSDO);

描述：设置电机参数值

参数：hMaster 主站对象句柄

nodeId 从站地址

paramNO 参数编号（CANopen/EtherCAT 使用 H 的对象字典，Modbus 使用寄存器）

uiValue 参数值

bSDO 是否通过 SDO 设置：1 通过 SDO；0 通过 PDO

返回值：0 成功；其它失败

3.15.5 Nim_get_param_value

函数原型: int Nim_get_param_value(unsigned int hMaster,int nodeId, const char* paramNO, unsigned int *uiValue, int bSDO);

描述：获取电机参数值

参数：hMaster 主站对象句柄

nodeId 从站地址

paramNO 参数编号（CANopen/EtherCAT 使用 H 的对象字典，Modbus 使用寄存器）

uiValue 输出参数，成功时返回 参数值

bSDO 是否通过 SDO 设置：1 通过 SDO；0 通过 PDO

返回值：0 成功；其它失败

3.15.6 Nim_save_AllParams

函数原型: int Nim_save_AllParams(unsigned int hMaster,int nodeId, int timeoutMS);

描述：保存电机所有参数

参数：hMaster 主站对象句柄

nodeId 从站地址

timeoutMS 超时时间（毫秒）

返回值：0 成功；其它失败

3.16 错误和报警

3.16.1 Nim_get_newestAlarm

函数原型: int Nim_get_newestAlarm(unsigned int hMaster,int nodeId, unsigned int *alarmCode, int bSDO);

描述：获取电机最新报警

参数：hMaster 主站对象句柄

nodeId 从站地址

alarmCode 输出参数，成功时返回最新报警

bSDO 是否通过 SDO 读取：1 通过 SDO；0 通过 PDO

返回值：0 成功；其它失败

3.16.2 Nim_get_alarmCount

函数原型: int Nim_get_alarmCount (unsigned int hMaster,int nodeId, int *count);

描述：获取历史报警数量

参数：hMaster 主站对象句柄

nodeId 从站地址

count 输出参数，成功时返回历史报警数量

返回值：0 成功；其它失败

3.16.3 Nim_get_alarm

函数原型: int Nim_get_alarm (unsigned int hMaster,int nodeId, int index, unsigned int *alarmCode);

描述：获取历史报警

参数：hMaster 主站对象句柄

nodeId　从站地址

index　历史报警序号（1~16）

alarmCode　输出参数，成功时返回指定的历史报警

返回值：0 成功；其它失败

3.16.4 Nim_clearError

函数原型: int Nim_clearError(unsigned int hMaster,int nodeId, int bSDO);

描述：清除电机故障状态

参数：hMaster 主站对象句柄

nodeId　从站地址

bSDO　是否通过 SDO 控制：1 通过 SDO；0 通过 PDO

返回值：0 成功；其它失败

4 函数调用流程

4.1 函数使用主流程图

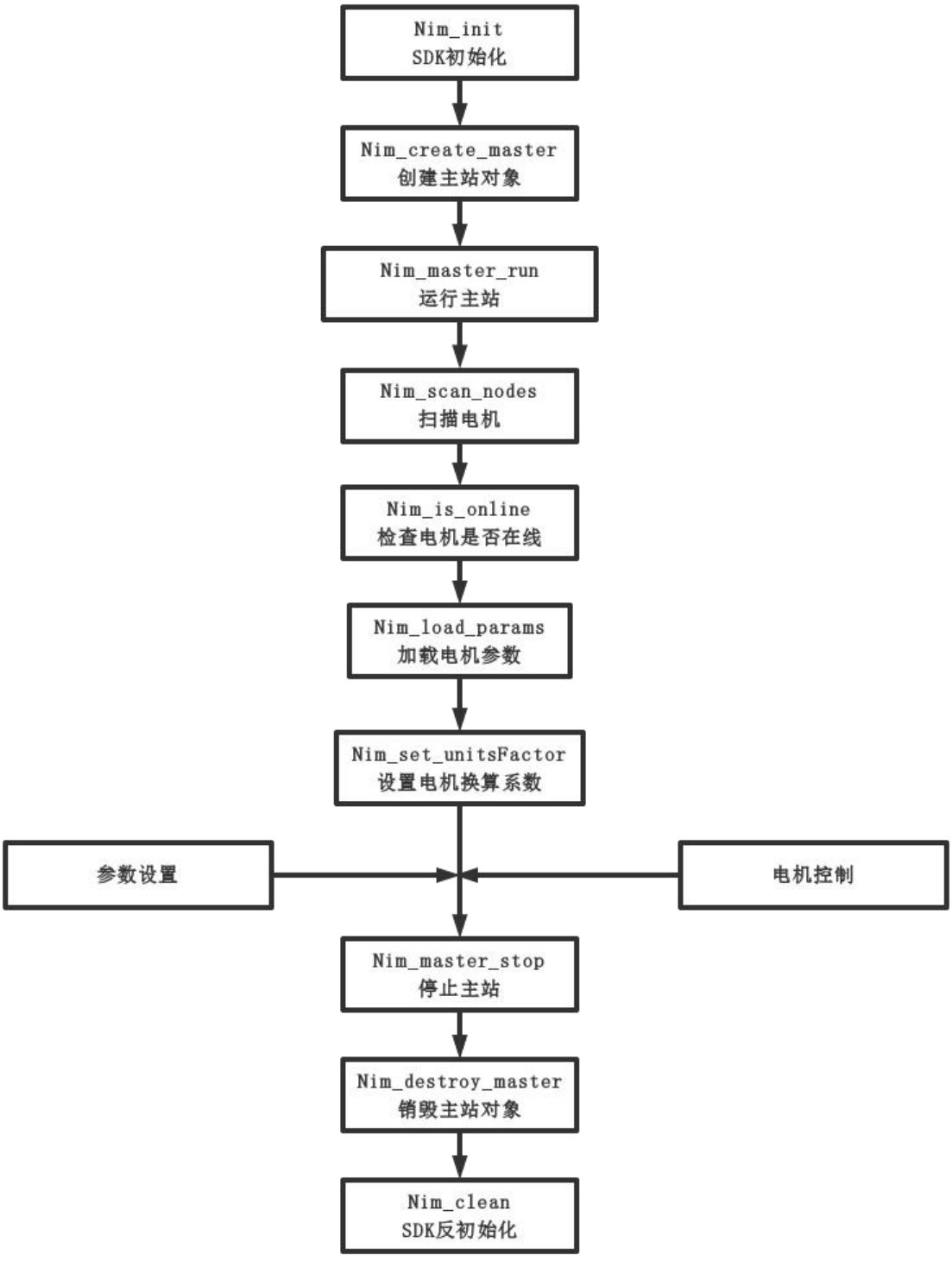


图 4-1 函数使用主流程图

4.2 电机控制流程及其常用模式

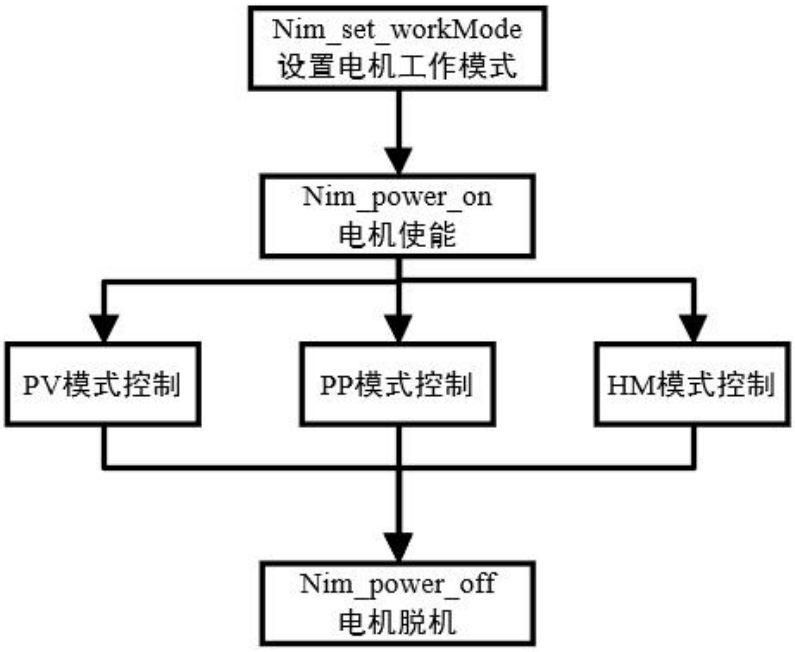


图 4-2 电机控制流程图

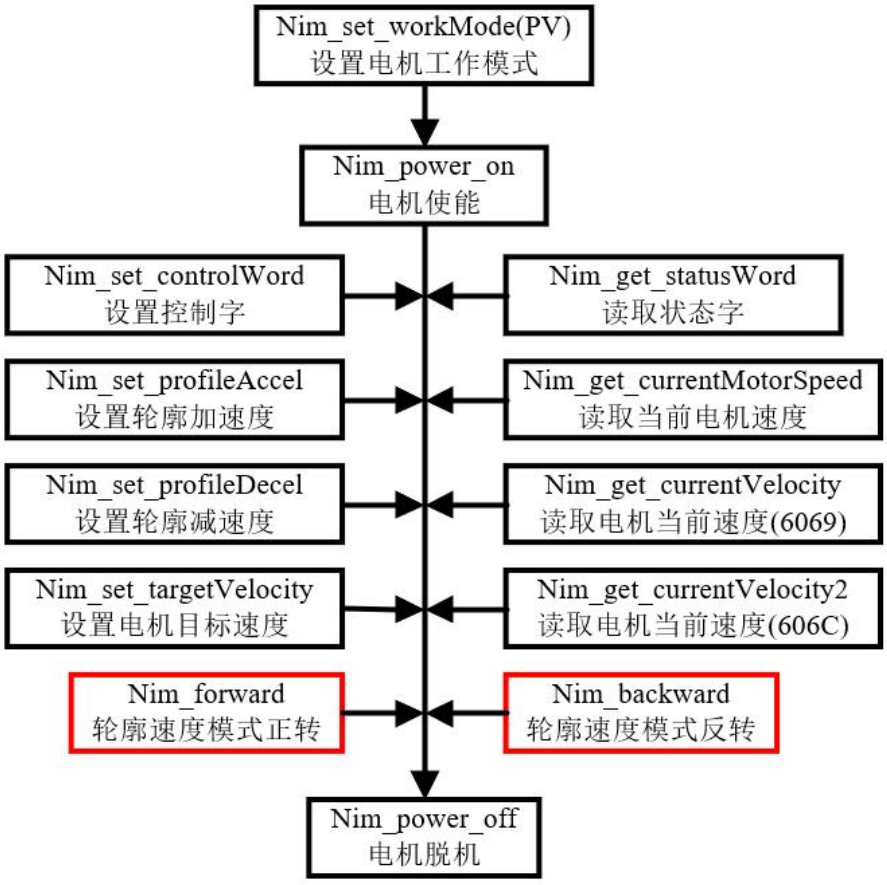


图 4-3 轮廓速度模式(PV)电机控制流程图

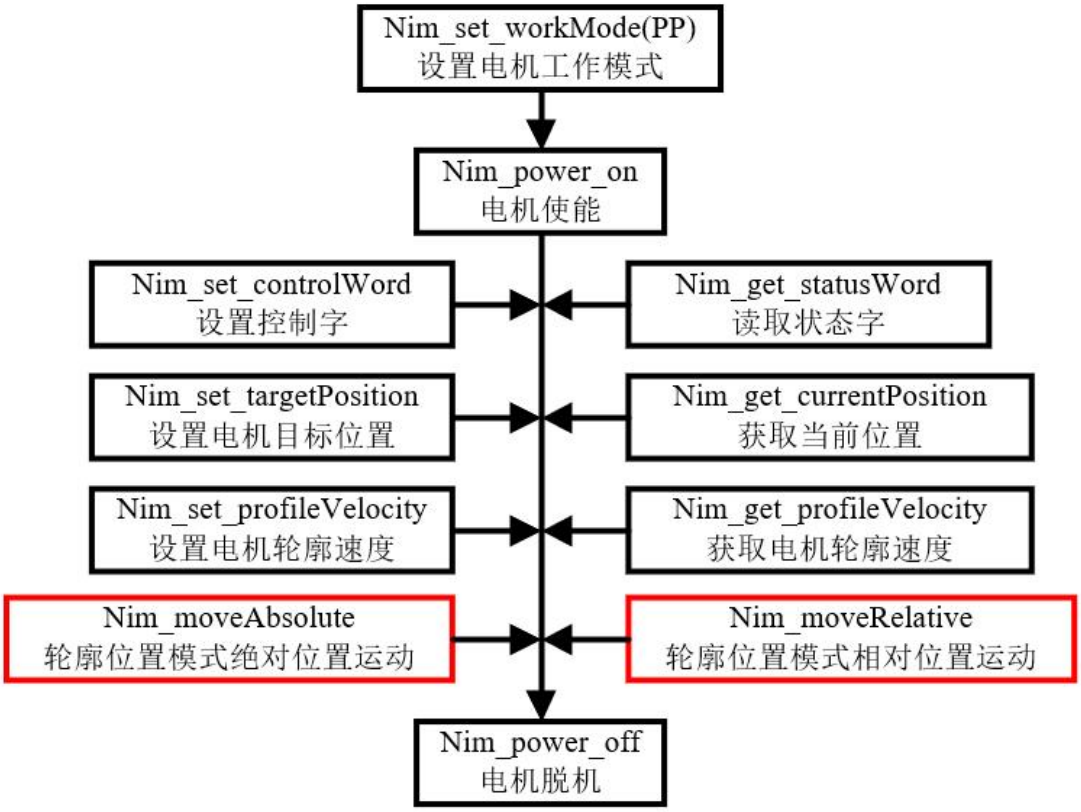


图 4-4 轮廓位置模式(PP)电机控制流程图

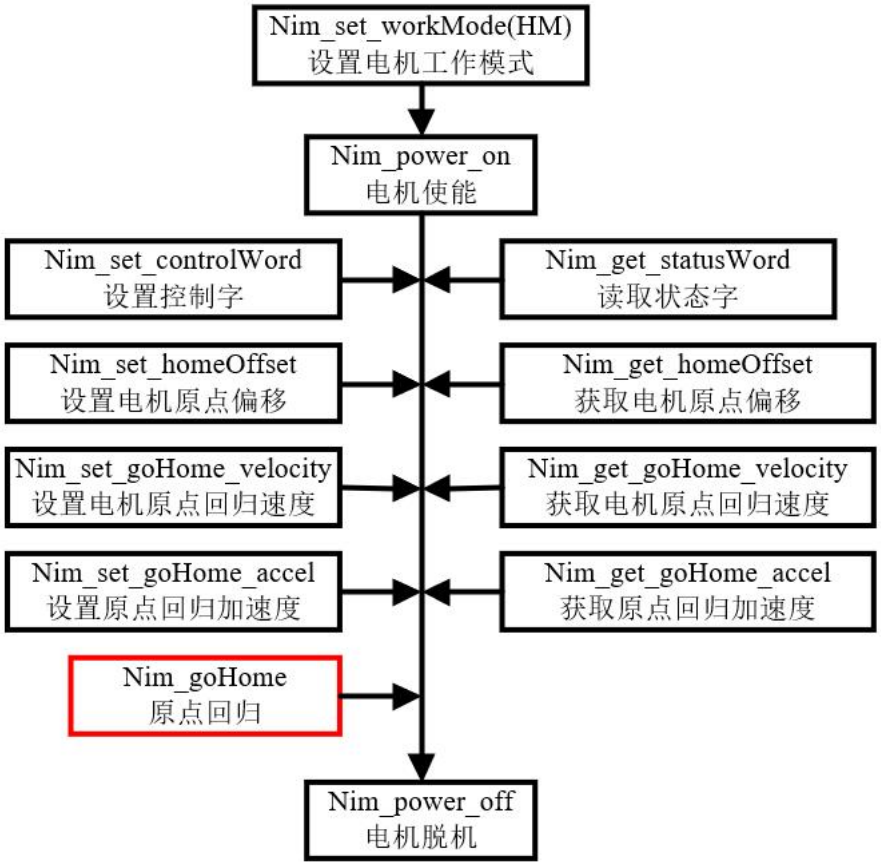


图 4-5 原点回归模式(HM)电机控制流程图

4.3 PDO 控制流程图

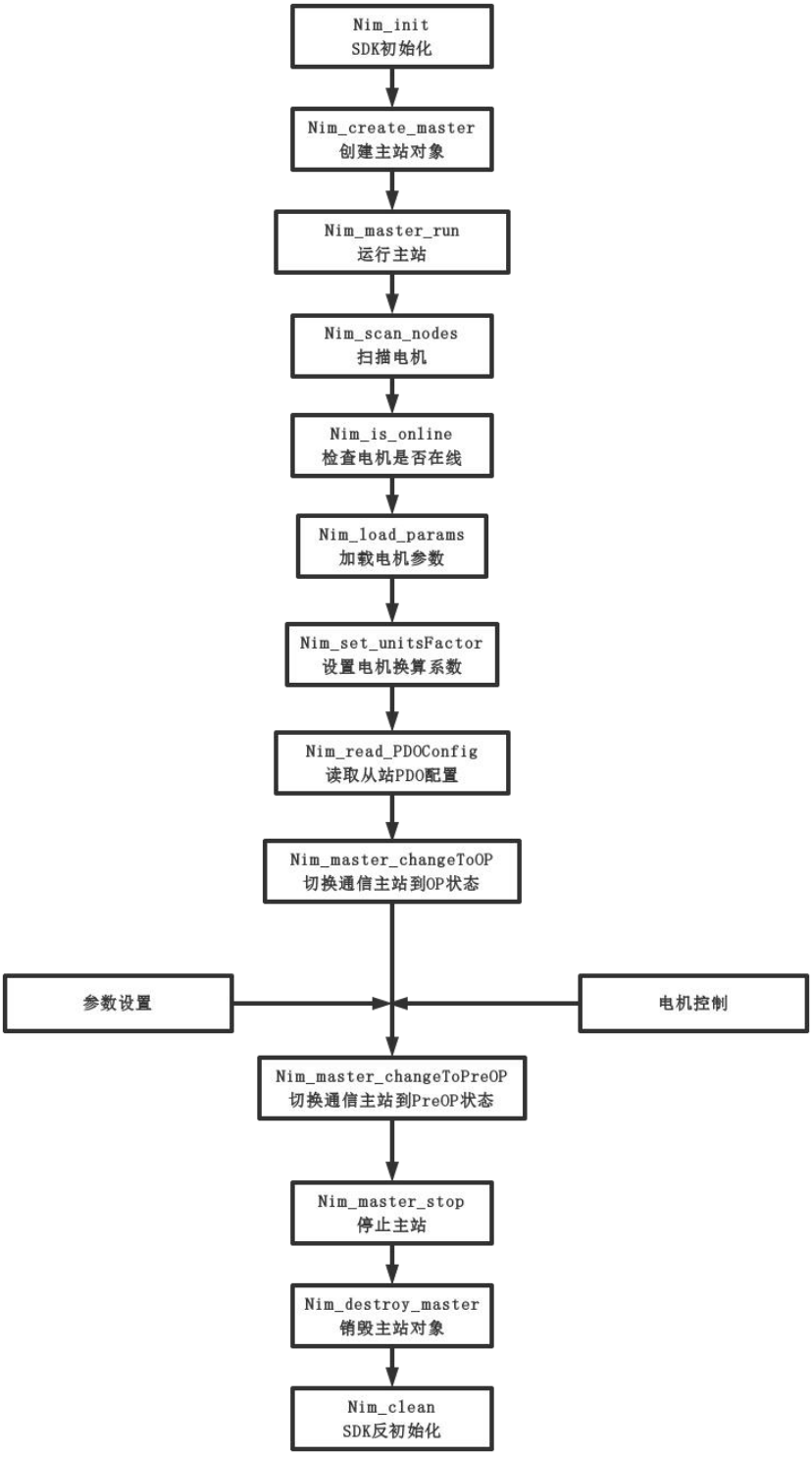


图 4-6 PDO 控制流程图

5 附录

5.1 枚举的相关内容

5.1.1 ServoSDK_Error

描述： 定义 SDK 接口函数除 Nim_is_online 外执行成功时返回 0；失败时返回错误码的枚举值。

```
enum ServoSDK_Error
{
    ServoSDK_NoError = 0,              // 0没有错误
    ServoSDK_NotRegisted,              // 1SDK没有注册              (当前无需注册)
    ServoSDK_NotInitialized,           // 2SDK没有初始化            (未执行初始化函数)
    ServoSDK_UnsupportedCommType,      // 3不支持的通信方式         (缺少bin路径下相关文件)
    ServoSDK_ParamError,               // 4输入参数错误             (通信连接字符串填写错误)
    ServoSDK_CreateMasterFailed,       // 5创建主站失败             (缺少对应主站)
    ServoSDK_MasterNotExist,           // 6主站不存在
    ServoSDK_MasterStartFailed,        // 7主站启动失败
    ServoSDK_MasterNotRunning,         // 8主站未运行
    ServoSDK_SlaveNotOnline,           // 9从站不在线               (对象字典H200C-02)
    ServoSDK_LoadParamSheetFailed,     //10加载参数表错误           (缺少数据库文件)
    ServoSDK_ParamNotExist,            //11请求的参数不存在         (对象字典填写错误)
    ServoSDK_ReadSDOFailed,            //12读SDO失败
    ServoSDK_WriteSDOFailed,           //13写SDO失败
    ServoSDK_OperationNotAllowed,      //14操作不允许               (电机使能不能进行相关操作)
    ServoSDK_MasterInternalError,      //15主站内部错误             (无法切换操作、预操作模式)
    ServoSDK_SlaveInternalError,       //16从站内部错误             (H6041处于故障状态)
    ServoSDK_Cia402ModeError,          //17从站402模式错误          (H6060与实际运动不符)
    ServoSDK_ReadWorkModeFailed,       //18读取工作模式失败         (对象字典H6060)
    ServoSDK_ReadStatusWordFailed,     //19读取状态字失败           (对象字典H6041)
    ServoSDK_ReadCurrentPosFailed,     //20读取当前位置失败         (对象字典H6064)
    ServoSDK_ReadRPDOConfigFailed,     //21读取PDO配置失败
    ServoSDK_ReadTPDOConfigFailed,     //22读取PDO配置失败
    ServoSDK_WriteControlWordFailed,//23写控制字失败               (对象字典H6040)
    ServoSDK_WriteTargetPosFailed,     //24写目标位置失败           (对象字典H607A)
    ServoSDK_WriteTargetVelFailed,     //25写目标速度失败           (对象字典H60FF)
    ServoSDK_WriteGoHomeTypeFailed, //26写原点回归方式失败          (对象字典H6098)
    ServoSDK_GetHostInfoFailed,        //27获取主机信息失败         (无需注册不再使用)
    ServoSDK_SaveParamsFailed,         //28保存参数失败
    ServoSDK_NoAvailableDevice,        //29没有可用的设备
    ServoSDK_Unknown = 255             //255未知错误
};
```

图 1 ServoSDK_Error 枚举

5.1.2 CanBaudrate

描述：定义可用波特率的枚举值。

```
enum CanBaudRate
{
    CAN_BT_10K = 0,
    CAN_BT_20K = 1,
    CAN_BT_50K = 2,
    CAN_BT_100K= 3,
    CAN_BT_125K= 4,
    CAN_BT_250K= 5,
    CAN_BT_500K= 6,
    CAN_BT_800K= 7,
    CAN_BT_1000K=8
};
```

图 2 CanBaudrate 枚举

5.1.3 ServoWorkMode

描述： 定义电机常用的工作模式枚举值。

```
enum ServoWorkMode
{
    SERVO_PP_MODE = 1,  //轮廓位置模式      PP
    SERVO_VM_MODE = 2,  //速度模式          VM
    SERVO_PV_MODE = 3,  //轮廓速度模式      PV
    SERVO_PT_MODE = 4,  //轮廓转矩模式      PT
    SERVO_HM_MODE = 6,  //原点回归模式      HM
    SERVO_IP_MODE = 7,  //位置插补模式      IP
    SERVO_CSP_MODE= 8,  //循环同步位置模式  CSP
    SERVO_CSV_MODE= 9,  //循环同步速度模式  CSV
    SERVO_CST_MODE= 10  //循环同步转矩模式  CST
};
```

图 3 ServoWorkMode 枚举

5.2 运行通信主站注意事项

5.2.1 CANopen

格式： {"DevType": "1001", "Baudrate": 8, "PDOIntervalMS": 10,SyncIntervalMS": 0, "……. "}
"DevType":"1001" : 表示设备类型，"1001"代表 NiMotion USBCAN
"Baudrate": 8 : 表示波特率参数序号，"8"对应定义的波特率参数枚举值，具体见附录 5.1.2
"PDOIntervalMS": 10: 表示 PDO 通信周期，"10" 通信周期参数
"SyncIntervalMS": 0: 设置同步参数，"0"不启动同步；
启动同步时必须为 PDOIntervalMS 的整倍数
"……. ": 不同 CAN 通信设备相关配置，具体内容如下

1）CANopen--NiMotoin USBCAN
格式： {"DevType": "1001", "Baudrate": 8, "PDOIntervalMS": 10,
"SyncIntervalMS": 0, "DevIndex": 0}
"DevType":"1001"： 表示设备类型，"1001"代表 NiMotion USBCAN
"DevIndex":0 ： 表示转换器参数序号，"0"代表 第一个设备

2）CANopen--周立功 USB-CAN
格式： {"DevType": "1002", "Baudrate": 8, "PDOIntervalMS": 10, "SyncIntervalMS": 0,
"ChannelIndex": 0,"DevSubType": 3, "DevIndex": 0, " ChannelNum": 0}
"DevType": "1002"： 表示设备类型，"1002"代表 周立功 USB-CAN
"ChannelNum": 1： 分两种单通只能为 0，双通可以选择 0 或 1
"DevSubType": 3： 子设备类型号，详细见附录 5.4 周立功设备对应子设备类型号。
"DevIndex":0 ： 表示转换器参数序号，"0"代表 第一个设备
"ChannelIndex": 0： 表示通道号序号，"0"代表 第一个通道

3）CANopen--Ixxat USB-CAN
格式：{"DevType": "1003", "Baudrate": 8,"PDOIntervalMS": 10,
"SyncIntervalMS": 0, "DevIndex": 0}
"DevType": "1003"： 表示设备类型，"1002"代表 Ixxat USB-CAN
"DevIndex":0 ： 表示转换器参数序号，"0"代表 第一个设备

4）CANopen--NiMotion TCP-CAN
格式：{"DevType": "1004","Baudrate": 8, "PDOIntervalMS": 10,
"SyncIntervalMS": 0, "IP": "192.168.0.100","Port": 40001}
"DevType": "1004"： 表示设备类型，"1002"代表 NiMotion TCP-CAN
"IP": "192.168.0.100"： IP 地址
"Port": 40001 ： 设置连接的端口号

5）CANopen--Linux SocketCAN
格式：{"DevType": "1005","Baudrate": 8, "PDOIntervalMS": 10,
"SyncIntervalMS": 0, "DeviceName": "can0"}
"DevType": "1005"： 表示设备类型，"1002"代表 Linux SocketCAN
"DeviceName": "can0"：设备名称

5.2.2 EtherCAT

格式：{"NetworkAdapter": "eth0","OverlappingPDO": true,"PDOIntervalMS": 10}
"NetworkAdapter": "eth0"： 网络适配器
"OverlappingPDO": true： true 使用重叠 PDO；false 不使用重叠 PDO
"PDOIntervalMS": 10： 表示 PDO 通信周期，"10" 通信周期参数

5.2.3 Modbus

格式：{"SerialPort": "COM1", "Baudrate": 115200, "Parity": "N", "DataBits": 8,
"StopBits": 1, "PDOIntervalMS": 10, "SyncIntervalMS": 0 }
"SerialPort": "COM1"：选择设备类型
"Baudrate": 115200 ：设置波特率
"Parity": "N"： N 无校验； O 奇校验； E 偶校验
"DataBits": 8： 数据位
"StopBits": 1： 停止位
"PDOIntervalMS": 10：表示 PDO 通信周期，"10" 通信周期参数
"SyncIntervalMS": 0 ：设置同步参数，"0"不启动同步；
启动同步时必须为 PDOIntervalMS 的整倍数

5.3 数据库选择

表 5-1

电机型号	数据库
PMM-XXXXX-CANopen	CANopen.db
BLM-XXXXX-CANopen	
STM-XXXXX-CANopen-M	
STM-XXXX-CANopen	CANopenSTM.db
PMM-XXXX-EtherCAT	EtherCAT.db
STM-XXXX-Modbus	Modbus.db
BLM-XXXX-Modbus-M	

备注：在 CANopen 和 EtherCAT 协议下的电机选择数据库如果选择了 Modbus 的数据库，再对电机进行操作时，所有函数都会报错，反之也是如此。

5.4 周立功设备对应子设备类型号

表 5-2

产品型号	设备名称	类型号
PCI-9810I	PCI9810	2
USBCAN-I/I+	USBCAN1	3
USBCAN-II/II+	USBCAN2	4
PCI-9820	PCI9820	5
CANET 系列的 TCP 工作方式	CANET-UDP	12
PCI-9840I	PCI9840	14
PC104-CAN2I	PC104-CAN2	15
PCI-9820I	PCI9820I	16
CANET 系列的 TCP 工作方式	CANET-TCP	17
PCI-5010-U	PCI-5010-U	19
USBCAN-E-U	USBCAN-E-U	20

USBCAN-2E-U	USBCAN-2E-U	21
PCI-5020-U	PCI-5020-U	22
PCIe-9221	PCIE-9221	24
CANWiFi-200T 的 TCP 工作方式	CANWIFI_TCP	25
CANWiFi-200T 的 UDP 工作方式	CANWIFI_UDP	26
PCIe-9120I	PCIe-9120I	27
PCIe-9110I	PCIe-9110I	28
PCIe-9140I	PCIe-9140I	29
USBCAN-4E-U	USBCAN-4E-U	31
CANDTU	CANDTU	32
USBCAN-8E-U	USBCAN-8E-U	34

5.5 SocketCAN 注意事项

ARM 平台下 SDK 内有一个 scripts 文件夹，如果使用 SocketCAN 连接电机，需要使用管理员权限，在/opt 路径下创建 NimServoSDK 文件夹将 scripts 文件夹移动到此处（SDK 调用脚本路径名称、不可修改/opt/NimServoSDK/scripts/rts_set_baud.sh），将脚本设置可执行权限。如下图 4 所示。

```
debian@NIMC2000:/opt/NimServoSDK/scripts$ ls -all
total 12
drwxr-xr-x 2 root root 4096 Jan  1 02:16 .
drwxr-xr-x 3 root root 4096 Jan  1 02:05 ..
-rwxr-xr-x 1 root root  174 Jan  1 02:06 rts_set_baud.sh
debian@NIMC2000:/opt/NimServoSDK/scripts$
```

图 4 设置脚本

SDK 执行成功，如图 5 所示。

```
root@NIMC2000:/home/debian/MMSDK2023/NimServoSDK-MM-bin-linux-arm_v7a/bin# ./test_pp 0 1005 10000
exec Nim_getLogFlags :1
"[2000-01-01 02:14:38.026] SDK init"
"[2000-01-01 02:14:38.066] 0 : CANopen Master"
"[2000-01-01 02:14:38.068] 1 : EtherCAT Master"
"[2000-01-01 02:14:38.071] 2 : Modbus Master"
"[2000-01-01 02:14:38.107] 1001 : NiMotion USB-CAN Device"
"[2000-01-01 02:14:38.110] 1004 : NiMotion TCP-CAN Device"
"[2000-01-01 02:14:38.112] 1005 : SocketCan Device"
{"DevType": "1005", "DeviceName": "can0", "Baudrate": 8, "PDOIntervalMS": 10, "SyncIntervalMS": 10}
ip link set can0 type can bitrate 1000000
Open socketcan successed
motor 1 is online
UnitFactor = 10000.000000
********************move relative**********************
statusword = 1637        currentpos = 9.998300
********************move absolute**********************
statusword = 1637        currentpos = 0.000700
root@NIMC2000:/home/debian/MMSDK2023/NimServoSDK-MM-bin-linux-arm_v7a/bin# ./test_csp 0 1005 10000
```

图 5 Socket 执行成功

SDK 执行失败，有可能是找不到脚本，或当前板子不支持对应脚本命令，需参考板子手册修改脚本内容。执行失败如图 6 所示。

```
root@NIMC2000:/home/debian/MMSDK2023/NimServoSDK-MM-bin-linux-arm_v7a/bin# ./test_csp 0 1005 10000
exec Nim_getLogFlags :1
"[2000-01-01 02:16:14.852] SDK init"
"[2000-01-01 02:16:14.892] 0 : CANopen Master"
"[2000-01-01 02:16:14.895] 1 : EtherCAT Master"
"[2000-01-01 02:16:14.898] 2 : Modbus Master"
"[2000-01-01 02:16:14.933] 1001 : NiMotion USB-CAN Device"
"[2000-01-01 02:16:14.936] 1004 : NiMotion TCP-CAN Device"
"[2000-01-01 02:16:14.939] 1005 : SocketCan Device"
{"DevType": "1005", "DeviceName": "can0", "Baudrate": 8, "PDOIntervalMS": 10, "SyncIntervalMS": 10}
sudo: /opt/NimServoSDK/scripts/rts_set_baud.sh: command not found
Open socketcan successed
Send Error frame[0]!
Send Error frame[0]!
Send Error frame[0]!
Send Error frame[0]!
Send Error frame[0]!
```

图 6 Socket 执行失败

- 本手册的全部内容或部分内容禁止擅自转载、拷贝。
- 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。
- 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。

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
邮编：102628
电话：(010)60213882　　传真：(010)60213882
邮箱：nimotion@nimotion.com
http://www.nimotion.com