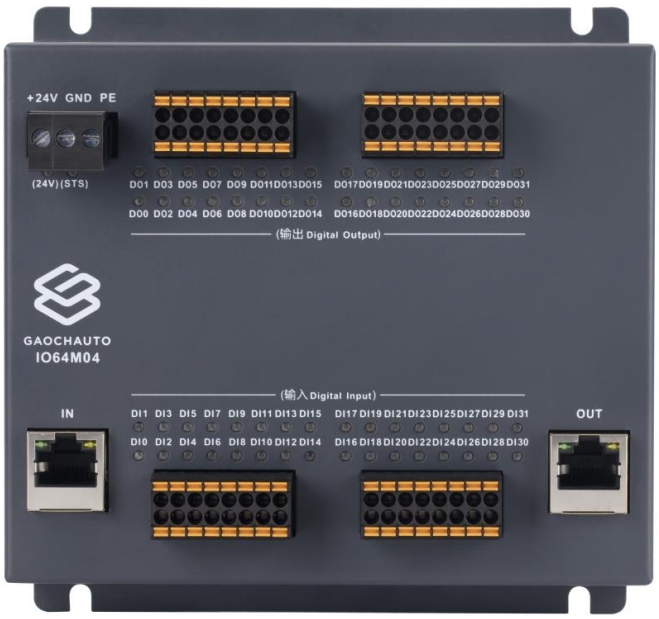
EtherCAT 扩展模块用户手册



2024

Version 1.01

目录

版权声明

本手册中的信息资料仅供参考。由于改进设计和功能等原因，高川自动化保留对本资料的最终解释权，内容如有更改，不另行通知。

调试、运动中的机器有危险！用户有责任在机器中设计有效的出错处理和安全保护机制，高川自动化没有义务或责任对由此造成的附带的或相应产生的损失负责。

联系我们

深圳市高川自动化技术有限公司

电话：0755-23502680

邮箱：sales@gcauto.com.cn

网址：www.gcauto.com.cn

Shenzhen Gaochuan Industrial Automation Co., Ltd.

Tel: +86 0755-23502680

Email: sales@gcauto.com.cn

Website: www.gcauto.com.cn

文档版本

版本号	修订日期	内容
V1.00	2023 年 6 月 1 日	-
V1.01	2024 年 8 月 20 日	增加部分指令说明，图和其他

前言

为了给用户提供更快捷，更方便的服务，提高用户的工作效率，本手册主要针对 EtherCAT 扩展模块硬件使用上的讲解，包括控制器的产品概述，配线指导，模块测试和软件编程，方便用户更好的使用我们的产品。

1 产品概述

IO64M02/M04，IO32M02/M04 和 IO24M02 提供一种低成本、简单易用、功能可靠的 IO 扩展方式，使用 EtherCAT 协议与外部通讯，可搭配我司 GC/GCS/GCE 系列控制器使用，也可与其他 EtherCAT 主站连接,我司将提供对应的 xml 文件,具体使用方法查看《EtherCAT 工具使用说明》。

1.1 IO64M02/M04（数字量）

1.1.1 尺寸图

如图 1.1.1 为 IO64M02/M04 模块的外观及尺寸：

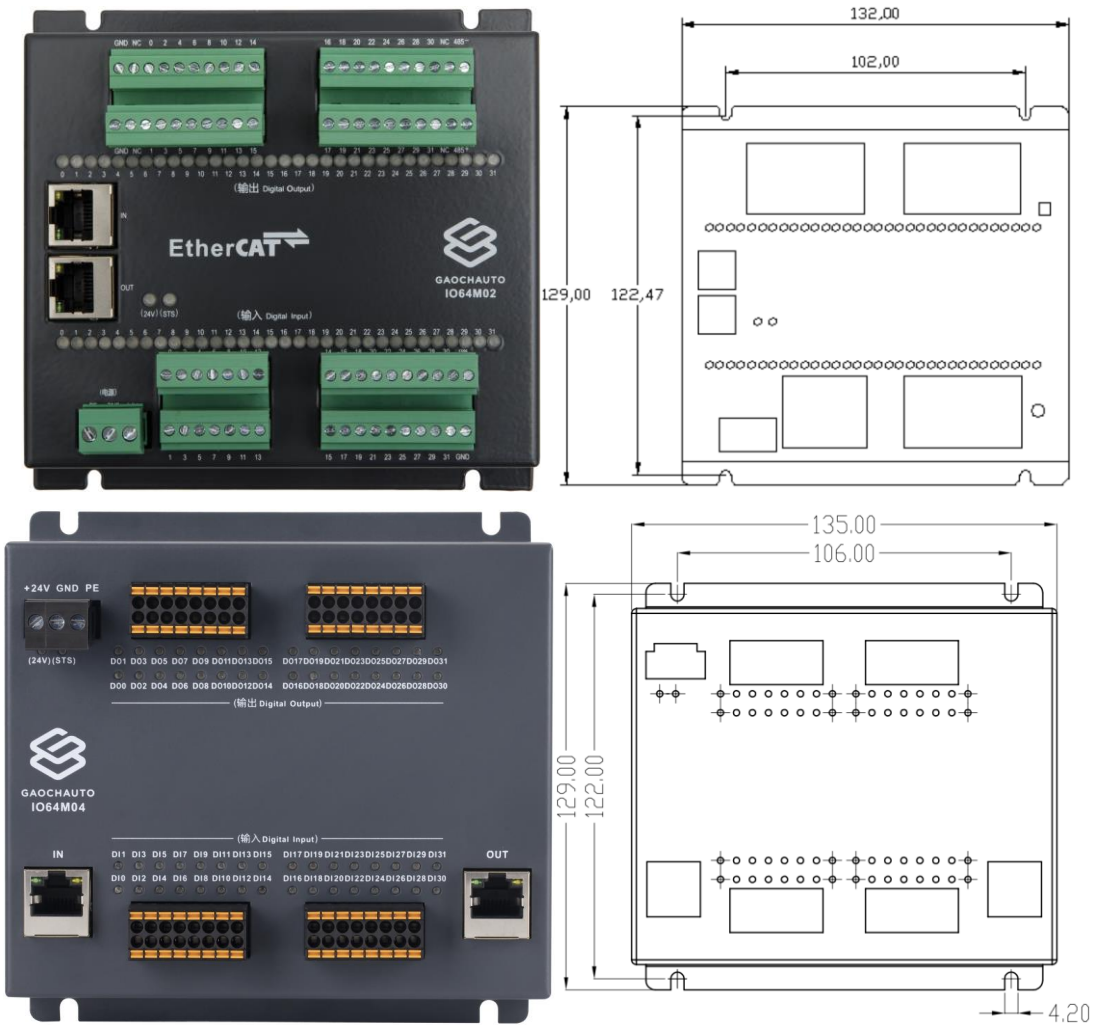


图 1.1.1 IO64M02/M04 外观及尺寸

1.1.2 电气规格

数字量输入	
通道	32

输入类型	NPN 型/低电平有效
输入阻抗	5KΩ
过压保护	70 VDC
ESD	2000 VDC
输入电流	3.5mA@0 VDC
输入电压	Logic 0：4Vmax.　　Logic 1：5V min(50V max)
数字量输出	
通道	32
输出类型	Sink（NPN）,短路保护
输出电压	Logic 0：0.5Vmax. Logic 1：开路(50V max)
Sink 电流	100mA
通讯	
通讯接口	EtherCAT 接口
常规规格	
尺寸	132x129mm(M02) / 135x129mm(M04)
系统供电	24VDC +/- 20%
湿度	5 ~ 95% RH, non-condensing (IEC 68-2-3)
工作温度	0 ~ 60° C (32 ~ 140° F)
存储温度	-20 ~ 85° C (-4 ~ 185° F)
软件支持	
操作系统	Windows XP/7/8/10/11 Linux WinCE
软件兼容性	VB/VC++/BCB/C#
演示软件	GCS.exe

1.2 IO32M02/M04（数字量）

1.2.1 尺寸图

如图 1.2.1 为 IO32M02/M04 模块的外观及尺寸：

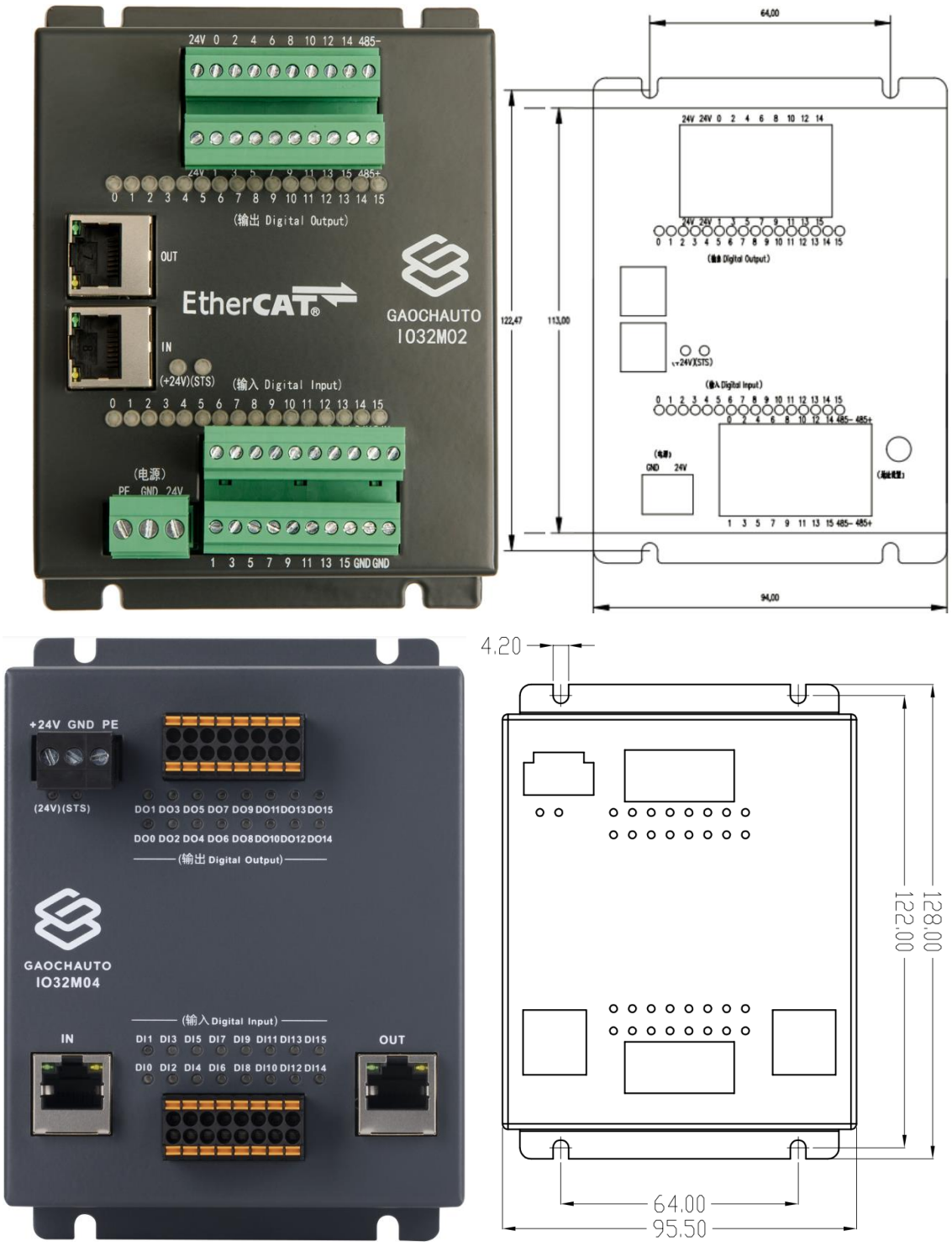


图 1.2.1 IO32M02M/04 外观及尺寸

1.2.2 电气规格

数字量输入	
通道	16

输入类型	NPN 型/低电平有效
输入阻抗	5KΩ
过压保护	70 VDC
ESD	2000 VDC
输入电流	3.5mA@0 VDC
输入电压	Logic 0：4Vmax. Logic 1：5V min(50V max)
数字量输出	
通道	16
输出类型	Sink（NPN）,短路保护
输出电压	Logic 0：0.5Vmax. Logic 1：开路(50V max)
Sink 电流	100mA
通讯	
通讯接口	EtherCAT 接口
常规规格	
尺寸	94x122mm(M02) / 95X128mm(M04)
系统供电	24VDC +/- 20%
湿度	5 ~ 95% RH, non-condensing (IEC 68-2-3)
工作温度	0 ~ 60° C (32 ~ 140° F)
存储温度	-20 ~ 85° C (-4 ~ 185° F)
软件支持	
操作系统	Windows XP/7/8/10/11 Linux WinCE
软件兼容性	VB/VC++/BCB/C#
演示软件	GCS.EXE

1.3 IO24M02（模拟量）

1.3.1 尺寸图

如图 1.3.1 为 IO24M02 模块的外观及尺寸：

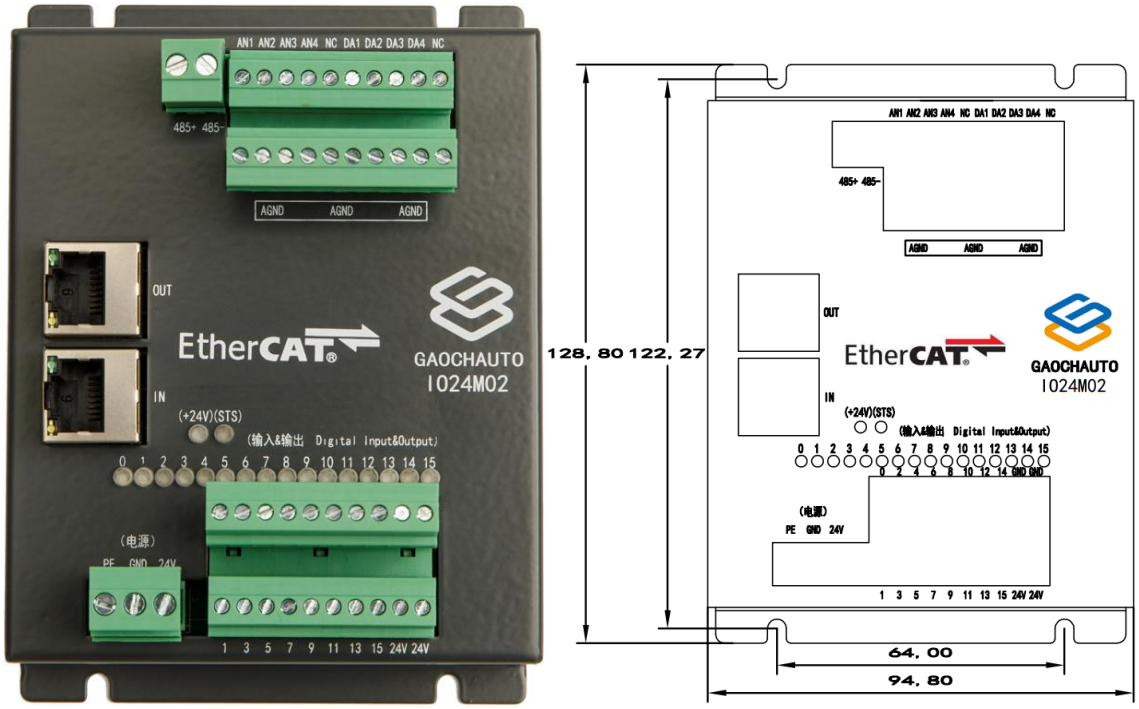


图 1.3.1 IO24M02 外观及尺寸图

1.3.2 电气规格

数字量输入	
通道	16 路(输入&输出)共用
输入类型	NPN 型/低电平有效
输入阻抗	5KΩ
过压保护	70 VDC
ESD	2000 VDC
输入电流	3.5mA@0 VDC
输入电压	Logic 0：4Vmax. Logic 1：5V min(50V max)
数字量输出	
通道	16 路(输入&输出)共用
输出类型	Sink（NPN）,短路保护
输出电压	Logic 0：0.5Vmax. Logic 1：开路(50V max)
Sink 电流	100mA
模拟量输入	

通道	4
输入阻抗	5KΩ
过压保护	70 VDC
ESD	2000 VDC
输入电流	3.5mA@0 VDC
输入电压范围	-10V ~ +10V
精度	12 位精度
模拟量输出	
通道	4
输出电压范围	-10V ~ +10V
精度	12 位精度
通讯	
通讯接口	EtherCAT 接口
常规规格	
尺寸	95x128mm
系统供电	24VDC +/- 20%
湿度	5 ~ 95% RH, non-condensing (IEC 68-2-3)
工作温度	0 ~ 60° C (32 ~ 140° F)
存储温度	-20 ~ 85° C (-4 ~ 185° F)
软件支持	
操作系统	Windows XP/7/8/10/11 Linux WinCE
软件兼容性	VB/VC++/BCB/C#
演示软件	GCS.exe

2 配线指导

IO64M02/M04，IO32M02/M04和IO24M02在配线和测试使用上相同，下面对三种模块统一说明；IO模块搭配端子板或控制卡接线示意图2.1.1和图2.1.2所示，黄色连接线为网线通讯线：

图 2.1.1 IO 模块与端子板接线示意图

图 2.1.2 IO 模块与 GCE 控制卡接线示意图

IO64M02/M04，IO32M02/M04 和 IO24M02 模块的上下两排的可拔插式接线端子分别为：输入、输出、模拟量输入（IO24M02）、模拟量输出（IO24M02），电源和网络通讯，输入输出部分请参考图 2.1.3 接线；

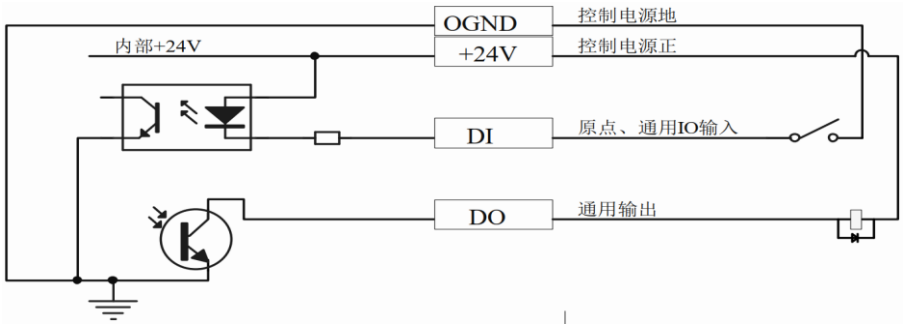


图 2.1.3 数字输入输出配线示意图

注意：当 GC/GCS 卡与两个或两个以上的扩展模块连接时，需要断电操作；IO64M02 扩展模块上标注“ COM” 的引脚， 需要与电源+24V 连接；

3 模块测试

3.1 搭配GC/GCS控制卡

（1）根据图 2.1.1 正确连接后上电；

（2）打开 GCS 工具，在菜单栏“功能”->“IO 相关”->通用 IO 扩展模块，如图 3.1.1；

（3）点击“使能”，保持常绿时，表示通讯成功；

（4）模块类型匹配型号：IO24M02->IO24_*；IO32M02/M04->IO32_*；IO64M02/M04->IO64_*；

（5）扩展 IO 地址：按照接入模块顺序排列，范围 2~9；

扩展 IO 地址	IO 起始位	IO 结束位	说明
2	64	95	注：每增加一个模块，则多增加 32； 如拨码为 2 的模块输入/输出第 1 位索引为 64； 软件编写请移步到 4.1；
3	96	127	
4	128	159	
5	160	191	
……	……	……	

图 3.1.1 扩展模块测试界面

3.2 搭配GCE控制卡

查看《EtherCAT 工具使用说明》，完成配置后，测试方式与 GC/GCS 控制卡相同；

4 软件编程

4.1 挂接GC/GCS控制卡

函数原形	函数说明
NMC_SetDOGroup	设置扩展 IO 输出(按通道,支持超过 32 位),带默认 group
NMC_GetDOGroup	读取扩展 IO 输出(按通道,支持超过 32 位) ,带默认 group
NMC_SetDOBit	按位设置扩展 IO 输出
NMC_GetDOBit	按位读取扩展 IO 输出
NMC_GetDIGroup	读扩展 IO 输入(按通道,支持超过 32 位) ,带默认 group
NMC_GetDIBit	按位读取扩展 IO 输入
NMC_SetDIBitRevs	按位设置扩展 IO 输入信号取反（不会改变灯的状态）
NMC_GetDIBitRevs	按位读取扩展 IO 输入信号取反的设置值
NMC_SetDOBitRevs	按位设置扩展 IO 输出信号取反（会改变灯的状态）
NMC_GetDOBitRevs	按位读取扩展 IO 输出信号取反的设置值
NMC_GetIOModuleSts	读取扩展 IO 模块的状态
NMC_IOModuleSetEn	设置扩展 IO 模块有效(带模块类型)
NMC_IOModuleGetType	读取扩展 IO 模块类型

（1）设置扩展 IO 输出(按通道,支持超过 32 位)，与 NMC_SetDO 功能相同

```
NMC_SetDOGroup( HAND devHandle, long value, short groupID);
```

参数	输入/输出	描述
devHandle	输入	控制器句柄
groupID	输入	DO 组，取值范围[0,n],0:本地 DO0~DO31， 1: 本地 DO32~DO63。其他指扩展 IO 模块，无拨码：第 1 个默认 groupID=2，以此类推；有拨码：根据拨码设置；(2≤groupID≤9)第 1 个模块开始位为 64，每增加 1 个模块，则多增加 32 位；
value	输入	默认情况下，1 表示高电平，0 表示低电平

（2）按位设置扩展 IO 输出

NMC_SetDOBit(HAND devHandle, short bitIndex, unsigned char value);

参数	输入/输出	描述
devHandle	输入	控制器句柄
bitIndex	输入	取值范围[0,n],位序号,前 64 位为本地通用输出。其他指扩展 IO 模块，第 1 个模块开始位为 64，每多 1 个模块，则多增加 32 位；
value	输入	默认情况下，1 表示高电平，0 表示低电平

（3）按位读取扩展 IO 输出

NMC_GetDOBit(HAND devHandle,short bitIndex, short *bitValue);

名称	输入/输出	描述
devHandle	输入	控制器句柄
bitIndex	输入	取值范围[0,n],位序号,前 64 位为本地通用输出。其他指扩展 IO 模块，第 1 个模块开始位为 64，每多 1 个模块，则多增加 32 位；
bitValue	输出	返回输出电平，默认情况下，1 表示高电平，0 表示低电平；

（4）读取扩展 IO 输出(按通道,支持超过 32 位)，此函数与 NMC_GetDO 功能相同

NMC_GetDOGroup(HAND devHandle, long *pDoValue, short groupID);

参数	输入/输出	描述
devHandle	输入	控制器句柄
groupID	输入	DO 组，取值范围[0,n],0:本地 DO0~DO31， 1: 本地 DO32~DO63。其他指扩展 IO 模块，无拨码：第 1 个默认 groupID=2，以此类推；有拨码：根据拨码设置；(2≤groupID≤9)第 1 个模块开始位为 64，每增加 1 个模块，则多增加 32 位；
value	输出	按组返回输出电平的状态 默认情况下 1 表示高电平，0 表示低电平

（5）读取扩展 IO 输入(按通道,支持超过 32 位)，此函数与 NMC_GetDI 功能相同

NMC_GetDIGroup(HAND devHandle, long *pInValue, short groupID);

参数	输入/输出	描述
devHandle	输入	控制器句柄

groupID	输入	DI 组，取值范围[0,n],0:本地 DOI~DI31， 1: 本地 DI32~DI63。其他指扩展 IO 模块，无拨码：第 1 个默认 groupID=2，以此类推；有拨码：根据拨码设置；(2≤groupID≤9)第 1 个模块开始位为 64，每增加 1 个模块，则多增加 32 位；
pInValue	输出	按位返回输入电平的状态 1：表示高电平/开路 0：表示低电平/闭合

（6）按位读取扩展 IO 输入

NMC_GetDIBit(HAND devHandle, short bitIndex, short *bitValue);

参数	输入/输出	描述
devHandle	输入	控制器句柄
bitIndex	输入	取值范围[0,n],位序号,前 64 位为本地通用输入。其他指扩展 IO 模块，第 1 个模块开始位为 64，每多 1 个模块，则多增加 32 位；
bitValue	输出	按位返回电平输入的状态 1：表示高电平/开路 0：表示低电平/闭合

（7）按位设置扩展 IO 输入信号取反（不会改变灯的状态）

NMC_SetDIBitRevs (HAND devHandle,short bitIndex,short revs);

参数	输入/输出	描述
devHandle	输入	控制器句柄
bitIndex	输入	取值范围[0,n],位序号,前 64 位为本地通用输入。其他指扩展 IO 模块，第 1 个模块开始位为 64，每多 1 个模块，则多增加 32 位；
revs	输入	是否取反 1：取反， 0：不取反

（8）按位读取扩展 IO 输入信号取反的设置值

NMC_GetDIBitRevs (HAND devHandle,short bitIndex,short *pRevs);

参数	输入/输出	描述
devHandle	输入	控制器句柄
bitIndex	输入	取值范围[0,n],位序号,前 64 位为本地通用输入。其他指扩展 IO 模块，第 1 个模块开始位为 64，每多 1 个模块，则多增加 32 位；
pRevs	输出	是否取反 1：取反，0：不取反

（9）按位设置扩展 IO 输出信号取反（会改变灯的状态）

NMC_SetDOBitRevs(HAND devHandle,short bitIndex,short revs);

参数	输入/输出	描述
devHandle	输入	控制器句柄
bitIndex	输入	取值范围[0,n],位序号,前 64 位为本地通用输出。其他指扩展 IO 模块，第 1 个模块开始位为 64，每多 1 个模块，则多增加 32 位；
revs	输入	是否取反　1：取反,0：不取反

（10）按位读取扩展 IO 输出信号取反的设置值

NMC_GetDOBitRevs(HAND devHandle,short bitIndex,short *pRevs);

参数	输入/输出	描述
devHandle	输入	控制器句柄
bitIndex	输入	取值范围[0,n]位序号,前 64 位为本地的 IO 输出，大于 64 为扩展 IO 输出
pRevs	输出	是否取反　1：取反,0：不取反

（11）读取扩展 IO 模块的状态

NMC_GetIOModuleSts(HAND devHandle, unsigned long *sts);

参数	输入/输出	描述
devHandle	输入	控制器句柄
sts	输出	在线：比特位为 1；离线：比特位为 0(最大支持 32 位比特位)

（12）设置扩展 IO 模块有效(带模块类型)

NMC_IOModuleSetEn(HAND devHandle, unsigned char chDevId,short chDevType);

参数	输入/输出	描述
devHandle	输入	控制器句柄
chDevId	输入	设备 ID
chDevType	输入	模块类型，见宏定义 // 扩展模块类型定义 #define IOMODULE_TYPE_IO64　1// 32DI32DO 模块（包括 16DI16DO 模块） #define IOMODULE_TYPE_IO32_DA 2// 4AD4DA 模块

（13）读取扩展 IO 模块类型

NMC_IOModuleGetType(HAND devHandle, unsigned char chDevId,short *pChDevType);

参数	输入/输出	描述
devHandle	输入	控制器句柄
chDevId	输入	设备 ID
pchDevType	输出	返回的模块类型，见宏定义 // 扩展模块类型定义 #define IOMODULE_TYPE_IO64 1// 32DI32DO 模块（包括 16DI16DO 模块） #define IOMODULE_TYPE_IO32_DA 2 // 4AD4DA 模块

4.2 应用案例

本地通用 IO 与扩展 IO 模块操作

```
/*********此处省略控制器初始化部分***********/
    HAND devHandle = 0;
    short rtn = 0;
    long divaule = 0;
    long divalueex = 0;
    long dovalue = 0;
    long dovalueex = 0;
    short di0 = 0;
    short di64 = 0;
    //读取扩展模块，设备 ID 从 2 开始，拨码开关也要拨码成对应 ID
    rtn = NMC_IOModuleSetEn(devHandle ,2,1);
    //读取一组扩展模块 IO 的数字量输入
    rtn = NMC_GetDIGroup(devHandle ,&divalueex,2);
    //读取一组本地 IO 数字量输入
    rtn = NMC_GetDIGroup(devHandle ,&divaule,0);
    //按位读取（单个读取）,这里示范读取第 0 个输入
    rtn = NMC_GetDIBit(devHandle ,0,&di0);
    //按位读取第一个扩展模块第一个 IO 输入值
    rtn = NMC_GetDIBit(devHandle ,64,&di64);
    //读取一组扩展模块 IO 的数字量输出（地址为 2 的扩展模块 DO）
    rtn = NMC_GetDOGroup(devHandle ,&dovalueex,2);
    //读取本地数字量输出
    rtn = NMC_GetDOGroup(devHandle ,&dovalue,0);
```

```
//设置本地数字量输出
rtn = NMC_SetDOGroup(devHandle ,0,0);
//设置本地第 1 个输出
rtn = NMC_SetDOGroup(devHandle ,0xFFFD,0);
//设置本地第 0 个输出,按位输出
rtn = NMC_SetDOBit(devHandle ,0,0);
return rtn;
```