恒昱控制运动控制器

MCX00A 系列

用户手册

Version 1.0

说明书版本号	修改日期	修改内容
V1.0	2026/1/5	

版 权 说 明

本手册版权归深圳市恒昱控制技术有限公司所有，未经公司书面许可，任何人不得翻印、翻译和抄袭本手册中的任何内容。

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，恒昱控制公司保留对本资料的最终解释权，内容如有更改，恕不另行通知。

调试机器要注意安全！用户必须在机器中设计有效的安全保护装置，在调试过程中人员保持足够安全的距离，在软件中必须加入出错处理程序，否则所造成的损失，恒昱控制技术有限公司没有义务或责任对此负责。

目 录

0 关于本手册

本手册旨在帮助你学习 MCX00A 系列控制器的使用，包括软件函数的调用、参数的设置、硬件接口的接线以及应用例程软件的编写等。

本手册总共分为 5 个部分：

1．引言：关于本产品的大致描述和关于本产品的相关申明。

2．软件系统概述：关于本产品的软件相关介绍，包括本产品所支持的系统驱动程序、运动控制函数库详细说明、演示程序和示例程序的说明以及多卡运行部分。

3．驱动程序安装：包括驱动程序详细的安装步骤。

4．演示软件及应用：包括演示软件及应用详细的说明和使用。

5．附录：运动控制函数说明、常见问题库。

备注：本说明书适用 MC400A、MC800A 运动控制器。

1 引言

MCX00A 是一款自主研发基于 ARM 和 FPGA 技术的高性能，高可靠的网络型运动控制器，4MHz 脉冲频率，可以控制步进或者数字伺服电机，支持非对称梯形，非对称 S 型加减速度曲线，支持梯形和 S 型速度曲线，在运动过程中可改变运动速度和目标位置。该卡同时最多可接受 4 路编码器信号和位置锁存信号，最多可以控制十六轴总线型步进电机或伺服电机。

同时，恒昱控制技术有限公司为 MCX00A 系列运动控制器设计了一套易学易用、功能丰富的运动函数库，大大缩短了用户应用软件开发、调试时间。随控制器免费提供的调试软件，不但可以演示和测试 MCX00A 系列运动控制器的绝大部分控制功能，而且还可以方便客户去测试控制器及电机控制系统硬件。

1.1 MCX00A 系列控制器对应的轴号范围及轴数

控制器对应轴号范围及轴数如图所示：

控制器名称	轴号范围	轴数
MC400A	（0-3）	4
MC800A	（0-7）	8

2 软件系统概述

恒昱控制 MCX00A 系列运动控制器软件系统包括：电脑 IP 设置、运动控制函数库、演示程序、示例程序。

2.1 硬件驱动程序

恒昱控制 MCX00A 系列配套提供 Windows7（32 位或 64 位）/XP NT/2000 等操作系统环境下的驱动程序。客户可以根据自己的需要选择相应的系统平台来开发适合自己的应用软件。硬件驱动程序具体安装方法请参考：驱动程序安装。

2.2 运动控制函数库

恒昱控制 MCX00A 系列为使客户能够方便地开发适合自己的应用控制系统，提供了丰富功能的函数库，客户可以根据自己应用系统的需要灵活调用不同的功能函数。

注：各函数具体功能介绍参考附录：运动控制函数库。

2.2.1 初始化、关闭运动控制器

在操作恒昱控制 MCX00A 系列运动控制器之前，必须调用控制器初始化函数为运动控制器分配资源。同样，当结束对运动控制器的操作时，必须调用控制器关闭函数释放运动控制器所占用的系统资源。本控制器支持打开多个进程对控制器同时进行操作。具体相关函数和功能如表 2-1 所示：

表 2-1 初始化/关闭控制器函数说明

	名称	功能	参考
1	MC_OpenEth	使用网络链接控制器	4.1.2.1
2	MC_Close	关闭控制器并释放系统资源	4.1.2.1
3	MC_OpenCom	使用串口链接控制器	4.1.2.1
4	MC_get_total_axes	读取控制器总轴数	4.1.2.1
注意：程序结束时，必须调用 MC_Close()函数释放系统资源。			

例程：初始化和关闭控制器 (以 C++语言为例说明，下同)

……

```
int iret = MC_OpenEth("192.168.1.221",&g_handle);
if(iret==0)
{
```

```
        MessageBox("链接成功");
}
else
{
        MessageBox("链接失败");
}
```

2.2.2 设置脉冲输出模式

恒昱控制 MCX00A 系列运动控制器使用指令脉冲方式控制步进/伺服电机。市面上的众多电机驱动器厂家信号接口要求各有不同（常用的有六种类型），所以在使用控制器控制具体的电机驱动器时，必须对脉冲输出方式进行正确的设定，电机才能正常工作。具体相关函数和功能如表 2-2 所示：

表 2-2 脉冲设置函数说明

	名称	功能	参考
1	MC_set_pulse_outmode	设置指定轴的脉冲输出模式	4.1.2.2
2	MC_get_pulse_outmode	读取指定轴的脉冲输出模式	4.1.2.2
注意：在调用运动控制函数之前应先调用来 MC_set_pulse_outmode 设置指令脉冲模式。			

指令脉冲包括两项基本信息：电机运转距离即脉冲数和电机转动方向。有两种基本指令模式：两种基本模式如表 2-3 所示：

表 2-3 两种基本的指令脉冲输出方式

模式	PULn-脚输出	DIRn-脚输出
脉冲/方向（PULSE/DIR）	脉冲信号	方向信号（电平）
双脉冲（CW/CCW）	正向（CW）脉冲	反向（CCW）脉冲
注意：具体怎样设置请参考 MC_set_pulse_outmode 函数具体说明。		

2.2.2.1 脉冲 / 方向模式

在此模式下，PULn-输出指令脉冲串，脉冲数对应电机运行的相应“距离”，而脉冲频率对应电机运行“速度”；DIRn-输出方向信号，该信号的不同电平对应电机不同的转动方向。此种模式在驱动器中最多。

脉冲信号可以设置为上升沿有效（即脉冲信号常态为低电平，变化为高时电机走一步）；也可设置为下降沿有效（即脉冲信号常态为高电平，变化为低时电机走一步）。方向信号可设置为高电平对应正向或低电平对应正向两种选择。所以实际上此种模式下有四种指令类型，如图 2-1 所示：

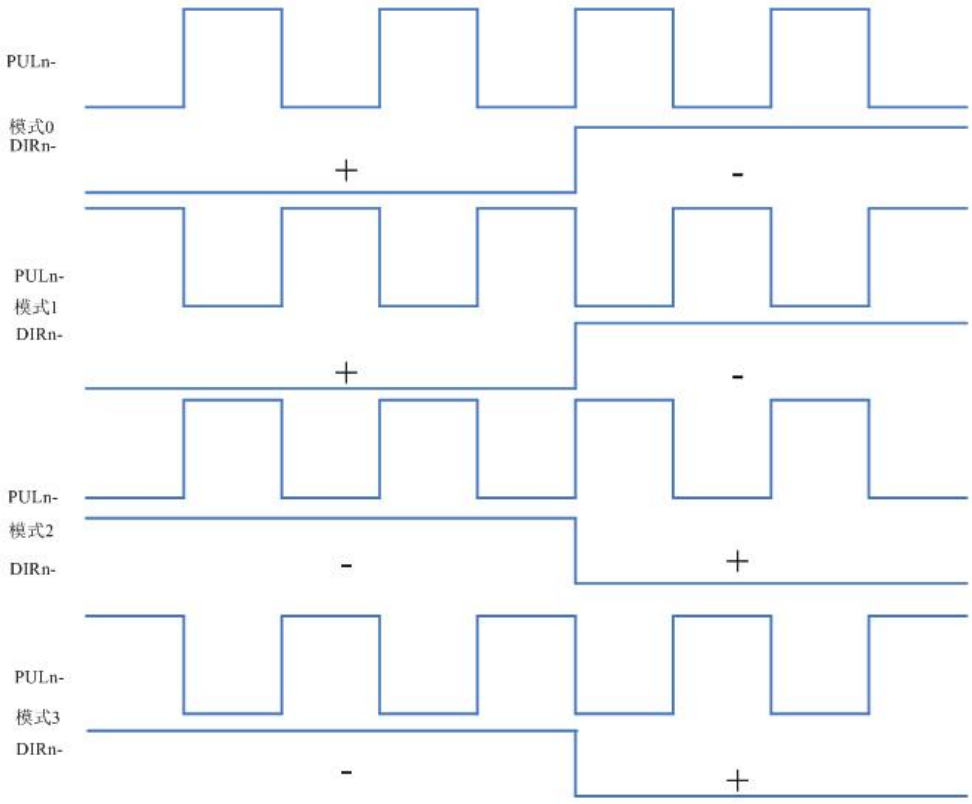


图 2-1 脉冲方向输出模式

2.2.2.2 双脉冲模式

在此模式下，PULn-和 DIRn-引脚分别表示正向（CW）和反向（CCW）脉冲输出。从 PULn-引脚输出的脉冲使电机正转，而 DIRn-引脚输出的脉冲使电机反转。脉冲信号有上升沿有效或下降沿有效的选择，所以此模式下共有两种指令类型，如图 2-2 所示：

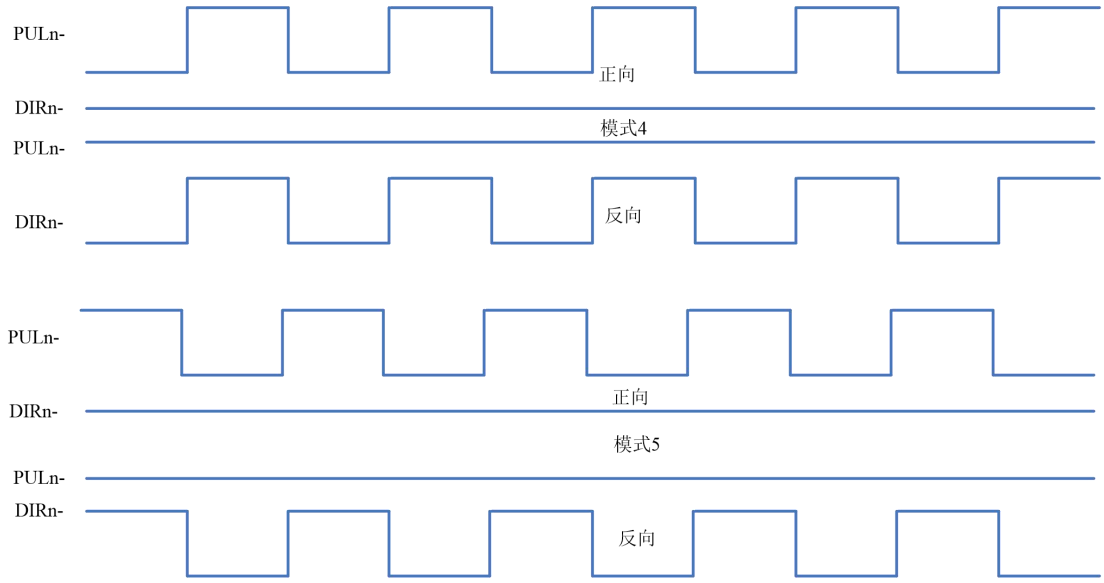


图 2-2 双脉冲输出模式

例程：设置脉冲输出方式

……

MC_set_pulse_outmode (g_handle,0,0)；//设置控制器第 0 轴脉冲输出模式为单脉冲模式，PULn-信号上升沿有效，DIRn-正向为低电平。

MC_set_pulse_outmode (g_handle,1,4)；//设置控制器第 1 轴脉冲输出模式为双脉冲模式，上升沿有效。

……

2.2.3 回原点运动

2.2.3.1 回原点模式

在进行精确的运动控制之前，需要设定运动坐标系的原点。运动系统动作时通常会执行寻找该原点的动作，然后将系统坐标位置归零，即回原点运动。恒昱控制 MCX00A 系列运动控制器提供多种回原点运动的方式，下面介绍 12 种常用的回原点的方式：

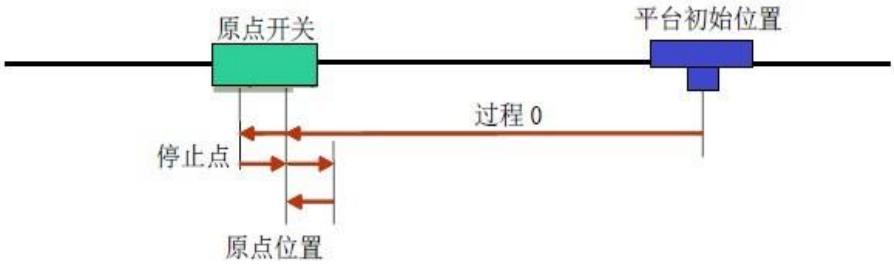
方式 0：一次回零

该方式以低速回原点，适合于行程短、安全性要求高的场合。动作过程为：电机从初始位置以恒定低速度向原点方向运动，当到达原点开关位置，原点信号被触发，电机立即停止（过程 0）；将停止位置设为原点位置，如图 2-3 所示：

图2-3 回原点方式0

方式 1：两次回零

该方式是方式 0 和方式 2 的结合，先以回零速度找到原点后，减速停止；然后以低速反向退出原点信号，检测不到原点信号后立即停止；再以回原点方向低速第二次找原点，检测到原点后，立即停止。如下图所示：



方式 2：一次回零加反找

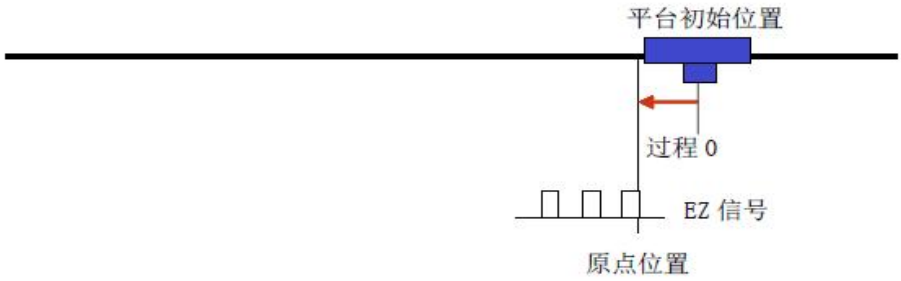
该方式先以回零速度找到原点后，减速停止；然后以低速反向退出原点信号，检测不到原点信号后立即停止。如下图所示：

方式 3：先一次回零加同向 EZ 回零运动

先一次回零，然后以一次回零相同的运动方向进行 EZ 回零运动，类似于方式 0 和方式 4 的组合。如下图所示：

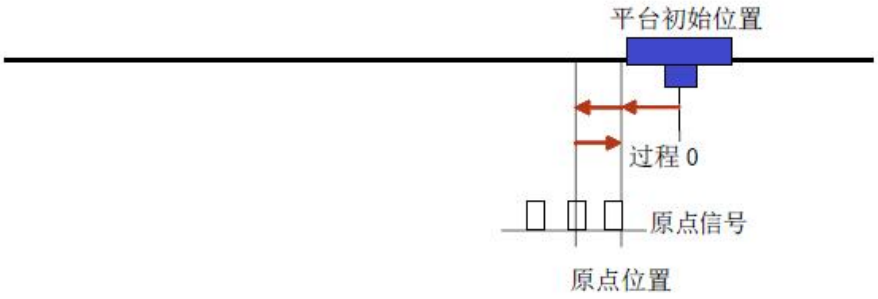
方式 4：以 EZ 作为原点进行一次回零

回零之前需要清除 EZ 状态，当 EZ 信号到来时，回零立即停止。如下图所示：



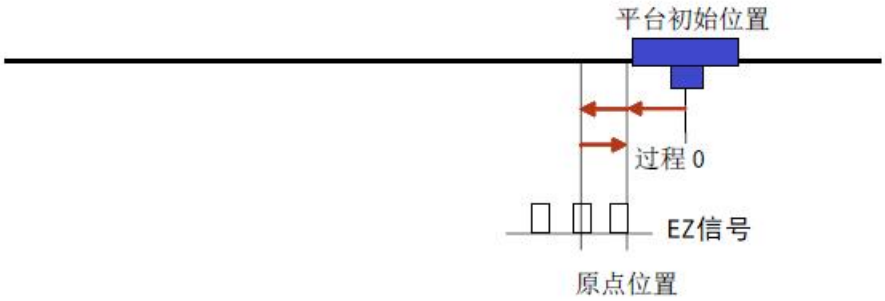
方式 16：原点信号锁存回零

原点信号设置为锁存信号，当原点信号被触发时，控制器会锁存当前轴的编码器位置并让当前轴减速停止，电机停止后再移动当前轴到锁存位置，将锁存位置作为原点。如下图所示：



方式 18：EZ 信号锁存回零

EZ 信号设置为锁存信号，EZ 点信号被触发时，控制器会锁存当前轴的编码器位置并让当前轴减速停止，电机停止后再移动当前轴到锁存位置，将锁存位置作为原点。如下图所示：



方式 20：一次限位回零

该方式以低速回原点，适合于行程短、安全性要求高的场合。动作过程为：电机从初始位置以恒定低速度向原点方向运动，当电机到达当前运动方向对应的限位开关位置，限位信号被触发，电机立即停止（过程 0）；将停止位置设为原点位置，如下图所示：

方式 21：一次限位回零加反找

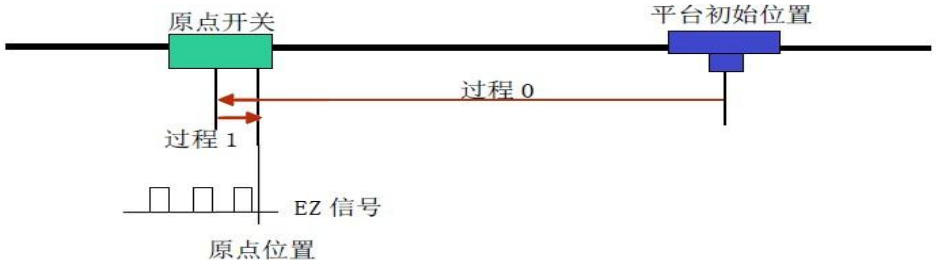
该方式先以回零速度找到限位后，减速停止；然后以低速反向退出限位信号，检测不到限位信号后立即停止。如下图所示：

方式 22：二次限位回零

该方式是方式 20 和方式 21 的结合，先以回零速度找到限位后，减速停止；然后以低速反向退出限位信号，检测不到限位信号后立即停止；再以回原点方向低速第二次找限位，检测到限位信号后，立即停止。如下图所示：

方式 40：先一次回零加反向 EZ 回零运动

先一次回零，然后以一次回零相反的运动方向进行 EZ 回零运动，类似于方式 0 和方式 4 的组合。如下图所示：



方式 41：一次限位回零加反向 EZ 回零运动

该方式先以回零速度找到限位后，减速停止；然后以一次回零相反的运动方

向进行 EZ 回零运动。如下图所示：

恒昱控制 MCX00A 系列运动控制器回原点的相关函数如下表 2-4 所示：

表 2-4 回原点相关函数说明

	名称	功能	参考
1	MC_config_HOME_PIN_logic	设置原点信号的电平和滤波器使能	5.1.2.9
2	MC_config_home_mode	设置指定轴的回原点模式	4.1.2.3.1
3	MC_get_config_home_mode	读取指定轴的回原地模式	4.1.2.3.1
4	MC_home_move	按指定的方向和速度方式开始回原点	4.1.2.3.1
注意：执行完 MC_home_move 函数后，指令脉冲计数器不会自动清零；如需清零可以在回零运动完成之后，调用 MC_set_position 函数进行清零。			

例程：一次回零

……

MC_config_HOME_PIN_logic(g_handle,0,0,0); //表示控制器的第 0 号轴的原点信号低电平有效

MC_set_profile_ex(g_handle,0,500,6000,300,0.02,0.01)；//设置控制器 0 号轴起始速度为 500 脉冲/秒、运行速度为 6000 脉冲/秒、结束速度为 300 脉冲/秒、加速时间为 0.02 秒、减速时间为 0.01 秒。

MC_config_home_mode(g_handle,0,2,0,1,0); //设置控制器，0 号轴，2 表示负方向，0 为低速(即该轴起始速度 500 脉冲/秒速度)回原点，1 表示二次回零，最后一个参数保留。

MC_home_move(g_handle,0); //表示控制器的第 0 轴开始回零运动

while (MC_check_done(g_handle,0) == 0) //等待回原点动作完成

{

}

MC_set_position(g_handle,0,0); //设置控制器 0 号轴的指令脉冲计数器绝对位置为 0

//设置编码器位置为 0

```
MC_set_encoder(g_handle,0,0);
……
```

2.2.4 限位开关和急停开关的设置

在设备进行运动功能调试之前，必须确保安全机制的有效。

限位开关在运动平台出现超出行程的运动时，起到限制作用，使电机减速或紧急停止，提高设备运行时的安全性能。在使用运动控制器进行运动控制之前，必须保证限位开关的有效性。

急停开关在运动过程中出现意外的运动时，能起到紧急停止运动的功能，提高设备运行时的安全性能。在使用运动控制器进行运动控制之前，必须保证急停开关的有效性。

2.2.4.1 限位开关的设置

恒昱控制 MCX00A 系列运动控制器提供了限位开关设置功能函数。用户必须根据设备的限位开关硬件接线，来设置限位开关工作的有效电平。

相关函数如下表所示：

	名称	功能	参考
1	MC_config_EL_PIN	设置 EL 信号的有效电平及制动方式	5.1.2.9
2	MC_axis_io_status	读取指定轴有关运动信号的状态	5.1.2.9

例程：限位开关设置

```
//设置控制器 0 号轴的限位信号使能，低电平有效，停止方式为立即停止
MC_config_EL_PIN(g_handle,0,1,0,0);

//读取控制器 0 号轴的轴信号状态
WORD iret = MC_axis_io_status(g_handle,0);
// EL+（正限位信号）
if (iret & 0x02)
{
    printf("正限位信号触发\n");
}
else
{
    printf("正限位信号未触发\n");
}
```

```
// EL-（负限位信号）
if (iret & 0x04)
{
    printf("负限位信号触发\n");
}
else
{
    printf("负限位信号未触发\n");
}
```

2.2.4.2 急停开关的设置

恒昱控制 MCX00A 系列运动控制器提供了急停开关设置功能函数。用户必须根据设备的急停开关硬件接线，来设置急停开关工作的有效电平。

相关函数如下表所示：

	名称	功能	参考
1	MC_config_EMG_PIN	设置 EMG 信号的有效电平及制动方式	5.1.2.9
2	MC_axis_io_status	读取指定轴有关运动信号的状态	5.1.2.9

例程：急停开关设置

```
//设置控制器的急停信号使能，低电平有效，停止方式为立即停止
MC_config_EMG_PIN(g_handle,1,0,0);

//读取控制器 0 号轴的轴信号状态
WORD iret = MC_axis_io_status(g_handle,0);
// EMG（急停信号）
if (iret & 0x08)
{
    printf("急停信号没有触发\n");
}
else
{
    printf("急停信号触发\n");
}
```

2.2.5 位置计数锁存比较

恒昱控制 MCX00A 系列运动控制器每个轴均有命令位置计数器和反馈位置计数器，命令位置计数器用于监测指令位置，反馈位置计数器用于监测机械位置，同时提供位置锁存和位置比较输出功能。

位置比较的一般步骤是：

1、配置比较器；

2、清除比较点；

3、添加/更新位置比较点；

4、开始运动并查看比较状态。

2.2.5.1 命令位置计数器

指令位置计数器是一个 32 位正负计数器，对控制器输出的指令脉冲进行计数。当输出一个正向脉冲后，计数器加 1；当输出一个负向脉冲后，计数器减 1。

相关函数如表 2-5 所示：

表 2-5 指令脉冲位置相关函数说明

	名称	功能	参考
1	MC_get_position	读取指定轴的指令脉冲计数器	4.1.2.4.1
2	MC_set_position	设置指定轴的指令脉冲计数器	4.1.2.4.1
3	MC_get_position_all	读取所有轴的指令脉冲计数器	4.1.2.4.1

例程：位置操作

MC_set_postion(g_handle,0,100); //设置控制器 0 号轴的脉冲位置为 100

position = MC_get_position(g_handle,0); //读取控制器 0 号轴的当前位置值至变量 position

2.2.5.2 反馈位置计数器

位置反馈计数器是一个 32 位正负计数器，对通过控制器编码器接口 EA，EB 输入的脉冲（如编码器、光栅尺反馈信号等）进行计数。

相关函数如表 2-6 所示：

表 2-6 编码器相关函数说明

	名称	功能	参考
1	MC_counter_config	设置指定轴编码器输入口的计数方式	4.1.2.4.2

2	MC_get_encoder	读取指定轴编码器反馈的脉冲计数值	4.1.2.4.2
3	MC_set_encoder	设置指定轴编码器的脉冲计数值	4.1.2.4.2
4	MC_get_encoder_all	读取所有轴编码器反馈的脉冲计数值	4.1.2.4.2
5	MC_config_encoder_dir	设置指定轴编码器反馈脉冲计数值方向	4.1.2.4.2

例程：编码器反馈计数的操作

MC_counter_config (g_handle,0,3); //设置控制器 0 号轴为 4 倍计数，默认的 EA、EB 计数方向

MC_set_encoder(g_handle,0,0); //设置控制器 0 号轴的计数初始值为 0

X_Position = MC_get_encoder(g_handle,0); //读取控制器 0 号轴的计数器的数值至变量 X_Position

2.2.5.3 位置锁存

恒昱控制 MCX00A 系列运动控制器提供编码器计数值锁存功能，该功能广泛应用于各种测量行业。位置锁存方式可以选择对每个编码器信号独立锁存，也可以通过任一个锁存端口对全部编码器计数值同时锁存，触发指令 LTC 接口一般接测量探头的触发信号。该功能用于位置测量时十分准确和方便。在复位锁存标志位后，锁存信号被触发，当前编码器计数值立即被捕获至位置锁存器中，并将触发标志位置位，保护当前锁存器内的数值，直到触发标志位被再次复位。注意：锁存信号接口为每一个轴的 EZ 信号或 ORG 信号，EZ 信号为 5V 信号，ORG 信号为 24V 信号。如果是使用 ORG 信号锁存，可直接把光纤的 24V 输入信号接入 ORG 信号。配置锁存方式通过 MC_config_LTC_PIN 函数进行设置。

相关函数如下表 2-7 所示：

表 2-7 位置锁存相关函数说明

	名称	功能	参考
1	MC_config_latch_mode	设置 LTC 端口的锁存方式	4.1.2.4.3
2	MC_get _latch_value	读取被触发锁存到锁存器内的编码器计数值	4.1.2.4.3
3	MC_get_latch_flag	读取指定控制器的锁存器的标志位	4.1.2.4.3
4	MC_reset_latch_flag	复位指定控制器的锁存器的标志位	4.1.2.4.3
5	MC_reset_latch_flag_ex	复位指定控制器指定轴的锁存器的标志位	4.1.2.4.3

位置锁存功能使用方法：

1. 开始前默认清除一次锁存标志-> MC_reset_latch_flag

2. 设置锁存模式为原点信号锁存-> MC_config_LTC_PIN(0,0,0,1)

3. 一直循环读取锁存标志直到锁存标志有效-> MC_get_latch_flag

4. 读取锁存器中的锁存值-> MC_get _latch_value

5. 清除锁存标志，开始下一次循环-> MC_reset_latch_flag

例程：位置锁存的操作(C++语言)

```
MC_reset_latch_flag(g_handle);                          //清除锁存标志
MC_config_LTC_PIN(g_handle,0,0,1);              //设置锁存模式为原点信号锁存
int  flag=0;                                                    //锁存状态标识
while(1)
{
    flag = MC_get_latch_flag(g_handle);              //读取锁存标志
    if (flag&(0x1L<<16)                                  //获取到第 0 轴有锁存标志
    {
        int pos = MC_get_latch_value(g_handle,0);   //获取锁存值
        MC_reset_latch_flag(g_handle);                  //清除锁存标志
    }
}
```

2.2.5.4 一维位置比较输出

恒昱控制 MCX00A 系列运动控制器提供了一维位置比较输出的功能，可用于将编码器位置或内部脉冲计数器位置与设定位置比较，当编码器位置或内部脉冲计数器位置到达设定位置时，触发相应输出口动作，单卡可以一次设置 256 个比较点。

在使用位置比较功能前，最好先通过 MC_Compare_ClearAllPoint 指令清除一次所有比较点，防止上次比较队列中还有多余未触发的比较点；然后可以通过 MC_Compare_GetPointsSpace 指令判断位置比较点是否添加成功（通常位置比较点添加成功后，可添加的位置比较点数量会减 1），或者用来观察位置比较队列是否已经满了（当位置比较缓冲队列已满时，可添加的位置比较点数量为 0），最后可以通过 MC_Compare_GetPointsRunned 指令查询比较点是否已经触发。

相关函数如表 2-8 所示：

表 2-8 一维位置比较相关函数说明

	名称	功能	参考
1	MC_Compare_Config	配置比较器	5.1.2.4.4
2	MC_Compare_GetConfig	读取配置比较器	5.1.2.4.4
3	MC_Compare_ClearAllPoint	清除所有比较点	5.1.2.4.4
4	MC_Compare_AddPoint	添加位置比较点	5.1.2.4.4
5	MC_Compare_GetCurPoint	读取当前比较点	5.1.2.4.4
6	MC_Compare_GetPointsRunned	查询已经比较过的点	5.1.2.4.4
7	MC_Compare_GetPointsSpace	查询可以加入的比较点数量	5.1.2.4.4

例程：位置比较输出操作

MC_Compare_ClearAllPoint(g_handle);

MC_set_position(0,0, 0);

MC_Compare_Config (g_handle,1, 0, 0); //设置 0 号 phandle 的位置比较功能

MC_Compare_AddPoint(g_handle, 10000, 1, 3, 1); //添加 0 号 phandle 的比较点，触发动作为反向 OUT1 口

MC_Compare_AddPoint(g_handle, 20000, 1, 3, 1);//添加 0 号 phandle 的比较点，触发动作为反向 OUT1 口

MC_Compare_AddPoint(g_handle, 30000, 1, 3, 1); //添加 0 号 phandle 的比较点，触发动作为反向 OUT1 口

MC_set_profile(g_handle,0, 0, 1000, 500, 500);

MC_pmove(g_handle,0, 50000,1);//启动控制器第 0 轴运动，比较点触发 OUT 端口状态变化

2.2.5.5 多组一维位置比较输出

恒昱控制 MCX00A 系列运动控制器提供了多组一维位置比较输出的功能，可用于将编码器位置或内部脉冲计数器位置与设定位置比较，当编码器位置或内部脉冲计数器位置到达设定位置时，触发相应输出口动作，单卡提供 0—23 号比较器，每个比较器有 256 个比较点。

在使用位置比较功能前，最好先通过 MC_Compare_ClearAllPoint_Muti 指令清除一次所有比较点，防止上次比较队列中还有多余未触发的比较点；然后可以通过 MC_Compare_GetPointsRunned_Muti 指令判断位置比较点是否添加成功（通常位置比较点添加成功后，可添加的位置比较点数量会减 1），或者用来观察位置比较队列是否已经满了（当位置比较缓冲队列已满时，可添加的位置比较点数

量为 0），最后可以通过 MC_Compare_GetPointsRunned_Muti 指令查询比较点是否已经触发。

相关函数如表 2-9 所示：

表 2-9 多组一维位置比较相关函数说明

	名称	功能	参考
1	MC_Compare_Config_Muti	设置比较器配置	5.1.2.4.7
2	MC_Compare_GetConfig_Muti	读取比较器的设置	5.1.2.4.7
3	MC_Compare_ClearAllPoint_Muti	清除所有比较点	5.1.2.4.7
4	MC_Compare_AddPoint_Muti	添加位置比较点	5.1.2.4.7
5	MC_Compare_GetCurPoint_Muti	读取当前比较点	5.1.2.4.7
6	MC_Compare_GetPointsRunned_Muti	查询已经比较过的点	5.1.2.4.7
7	MC_Compare_GetPointsSpace_Muti	查询可以加入的比较点数量	5.1.2.4.7
8	MC_Compare_Set_PulseTime_Muti	设置功能号 9 的脉冲宽度	5.1.2.4.7
9	MC_compare_set_filter_Muti	设置锁存器滤波频率	5.1.2.4.7
10	MC_compare_get_filter_Muti	读取锁存器滤波频率	5.1.2.4.7

2.2.5.6 高速一维位置比较功能

高速一维位置比较功能和一维比较类似，但是高速位置比较精度更高，精度可保证在一个脉冲以内。可以用于高速测量、飞拍等功能。MCX00A 支持 4 路高速比较输出，OUT1 、OUT2、OUT3 和 OUT4 作为高速一维位置比较输出口。每路高速比较输出，支持 256 段空间。

表 2-11 高速一维位置比较相关函数说明

	名称	功能	参考
1	MC_hcmp_config	配置高速比较输出	5.1.2.4.9
2	MC_hcmp_get_config	读取高速比较输出配置	5.1.2.4.9
3	MC_hcmp_clear_points	清除高速比较输出数据	5.1.2.4.9
4	MC_hcmp_add_point	添加高速位置比较点	5.1.2.4.9
5	MC_hcmp_get_current_point	读取当前运行的比较点数据	5.1.2.4.9
6	MC_hcmp_get_points_runned	读取已经运行的比较点个数	5.1.2.4.9
7	MC_hcmp_get_points_remained	查询可以加入的比较点数量	5.1.2.4.9

高速比较通道	输出口位置	功能
0	OUT1	高速比较输出 0 通道
1	OUT2	高速比较输出 1 通道
2	OUT3	高速比较输出 2 通道
3	OUT4	高速比较输出 3 通道

例程：

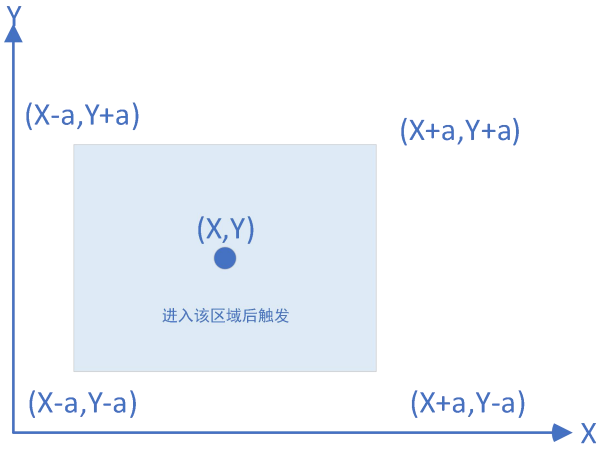
```
//添加高速位置比较点前先清除一次比较点
MC_hcmp_clear_points(g_handle,0);
//使能控制器第 0 组位置比较
MC_hcmp_config(g_handle,0,1,0,0);
//设置通道 0 在编码器位置为 1000 的地方触发比较点输出 1000us 脉冲
MC_hcmp_add_point(g_handle,0,1000,0,0,1000);

//需要退出高速位置比较模式时
//先禁止高速位置比较模式
MC_hcmp_config(g_handle,0,0,0,0);
```

2.2.5.7 高速二维位置比较功能

恒昱控制 MCX00A 系列运动控制器提供了高速二维位置比较输出的功能，可用于将编码器位置或指令位置与设定位置比较，当编码器位置或指令位置到达设定位置时，触发相应输出口动作，单卡提供 1 组比较器，每个比较器有 256 个比较点，OUT1 、OUT2、OUT3 和 OUT4 作为高速二维位置比较输出口，可以输出置其中一个或多个输出口。

高速二维比较，可以选择比较指定轴的指令位置或者编码器位置。如下图所示通过设置比较间隔，当指定两个轴的位置(指令或编码器)和目标位置距离都小于比较间隔时(1-255)，触发指定输出口输出指定时间(最小 1 微秒，最大 65 秒)的脉冲。



高速二维比较采样 FPGA 纯硬件比较，比较实时性非常高，可以达到微秒级精度，特别适合高速飞拍，飞行点胶等场合。在使用位置比较功能前，最好先通过 MC_hcmp2d_clear_points 指令清除一次所有比较点，防止上次比较队列中还有多余未触发的比较点；然后可以通过 MC_hcmp2d_get_points_remained 指令判断位置比较点是否添加成功（通常位置比较点添加成功后，可添加的位置比较点数量会减 1），或者用来观察位置比较队列是否已经满了（当位置比较缓冲队列已满时，可添加的位置比较点数量为 0），最后可以通过 MC_hcmp2d_get_points_runned 指令查询比较点是否已经触发。

表 2-12 高速二维位置比较相关函数说明

	名称	功能	参考
1	MC_hcmp2d_config	配置高速比较输出	5.1.2.4.11
2	MC_hcmp2d_get_config	读取高速比较输出配置	5.1.2.4.11
3	MC_hcmp2d_clear_points	清除高速比较输出数据	5.1.2.4.11
4	MC_hcmp2d_add_point	添加高速位置比较点	5.1.2.4.11
5	MC_hcmp2d_get_current_point	读取当前运行的比较点数据	5.1.2.4.11
6	MC_hcmp2d_get_points_runned	读取已经运行的比较点个数	5.1.2.4.11
7	MC_hcmp2d_get_points_remained	查询可以加入的比较点数量	5.1.2.4.11

2.2.6 伺服驱动器专用

恒昱控制 MCX00A 系列运动控制器控制伺服驱动器时，软件系统为每个轴提供了伺服驱动专用的 6 个信号（SVON，ALM）的相应的功能函数，其具体功

能和相应的函数如下：

2.2.6.1 SEVON：伺服驱动器使能

恒昱控制 MCX00A 系列运动控制器为每轴均提供了伺服驱动器使能（SEVON）信号输出接口及操作函数。当该信号为无效状态时，伺服驱动器不使能，电机处于自由状态；当该信号有效时，伺服驱动器使能，电机将锁紧。与之相关函数如下表 2-22 所示：

表 2-22 SEVON 信号相关函数说明

	名称	功能	参考
1	MC_write_SEVON_PIN	设置指定轴的伺服使能端子的电平状态	4.1.2.9
2	MC_read_SEVON_PIN	读取指定轴的伺服使能端子的电平状态	4.1.2.9

例程：输出与读取指定轴伺服控制

……

MC_write_SEVON_PIN(g_handle,0,1); //输出控制器 0 轴伺服使能端子高电平

status= MC_read_SEVON_PIN(g_handle,0); //读取控制器 0 轴伺服使能端子的状态

……

2.2.6.2 ALM：伺服报警

恒昱控制 MCX00A 系列运动控制器为每轴均提供了伺服驱动器报警（ALM）信号输入接口及相应的操作函数。当伺服驱动器发生错误时将自动将该信号置为有效，用来报告伺服驱动器或电机出现了错误。该信号可以通过软件对其有效电平进行设置，同时还可以设置该信号有效时制动模式。若设置模式为立即停止，则 ALM 信号有效时，MCX00A 系列控制器接将立即停止脉冲输出，该过程是一个硬件处理过程。与之相关函数如表 2-25 所示：

表 2-25 ALM 信号相关函数说明

	名称	功能	参考
1	MC_config_ALM_PIN	设置 ALM 的有效电平及其工作方式	4.1.2.9
2	MC_get_config_ALM_PIN	读取 ALM 的有效电平及其工作方式	4.1.2.9
3	MC_axis_io_status	读取指定轴有关运动信号的状态，包含指定轴的专用 I/O 状态	4.1.2.9

例程：ALM 信号设置

……

MC_config_ALM_PIN(g_handle,0,1,1,1); //设定控制器 0 轴 ALM 信号有

效，且为减速停止制动方式

……

2.2.7 单轴多轴运动控制

2.2.7.1 单轴运动控制

恒昱控制 MCX00A 系列运动控制器在表述运动轨迹时可以用绝对坐标和相对坐标 2 种模式（如图 2-5 所示）。2 种模式各有优点。如：在绝对坐标模式中用一系列坐标点定义一条曲线，如果要修改中间某点坐标时，不会影响后续点的坐标；在相对坐标模式中，用一系列坐标点定义一条曲线，用循环命令可以重复这条曲线轨迹多次。

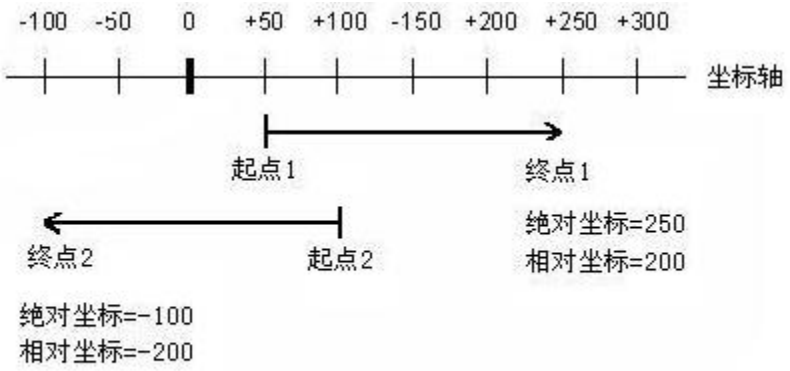


图2-5 绝对坐标与相对坐标中轨迹终点的不同表达方式

在 MCX00A 系列函数库中距离或位置的单位为脉冲；速度单位为脉冲/秒；时间单位为秒。最基本的位置控制是指从当前位置运动到另一个位置，一般称为点位运动或定长运动。

MCX00A 系列控制器在执行单轴控制时，可使电机按照梯形速度曲线或 S 形速度曲线运动模式进行点位运动或连续运动。

2.2.7.1.1 梯形速度曲线运动模式

梯形速度曲线运动是位置控制中最基本的运动模式。在此模式下移动一段指定距离时，其运动速度按梯形曲线变化，如图 2-6 所示。

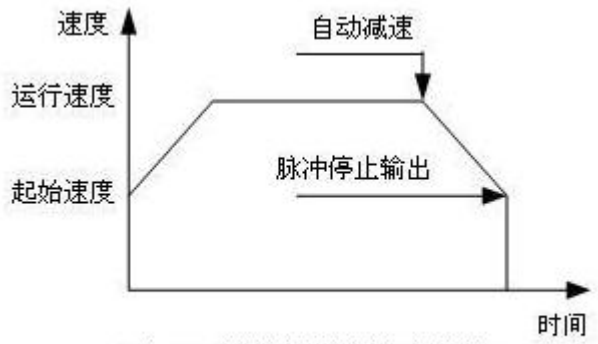


图2-6 简单的梯形速度曲线

运动速度之所以要按梯形曲线变化，是因为：电机转轴和被拖动的物体具有惯性，不可能在瞬间内达到指定速度，因此应该给予一定的加速时间；减速时亦是类似，否则电机会因为瞬间力矩不足而出现丢步、过冲（步进系统）或振荡（伺服系统）现象。

实现以梯形速度曲线运动的点位控制函数如表 2-12 所示：

表 2-12 梯形点位控制相关函数说明

	名称	功能	参考
1	MC_set_profile_ex	设置指定轴的起始速度、最大运行速度、结束速度、加速时间、减速时间	4.1.2.5.1
2	MC_get_profile_ex	读取指定轴的起始速度、最大运行速度、结束速度、加速时间、减速时间	4.1.2.5.1
3	MC_pmove	使指定轴做点位运动	4.1.2.5.1
4	MC_pmove_profile	设置轴速度并启动点位运动	4.1.2.5.1

注意：

如需使用梯形速度曲线运动模式，应先把函数 MC_set_s_profile()里面的参数 s_para 设为 0。详情可参考 5.1.2.5.1。

例程：执行以非对称梯形速度曲线作点位运动

……

MC_set_profile_ex(g_handle,0,500,6000,300,0.02,0.01)；//设置控制器 0 号轴起始速度为 500 脉冲/秒、运行速度为 6000 脉冲/秒、结束速度为 300 脉冲/秒、加速时间为 0.02 秒、减速时间为 0.01 秒。

MC_pmove(g_handle,0,50000,0)；//设置控制器 0 号轴、运动距离为 50000 个脉冲、相对坐标，并开始执行运动

……

在单轴运行过程中，运动速度 Max_Vel 和目标位置 Dist 均可以实时改变如

图 2-7 所示。

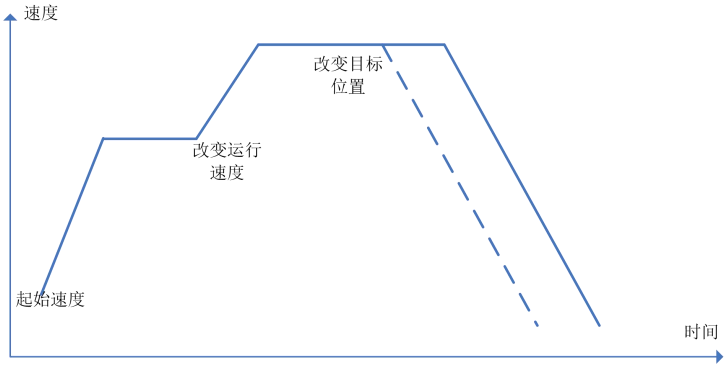


图 2-7 点位运动在线变速和变位置速度曲线

若在减速时改变目标位置，电机的速度将如图 2-8 所示发生变化。

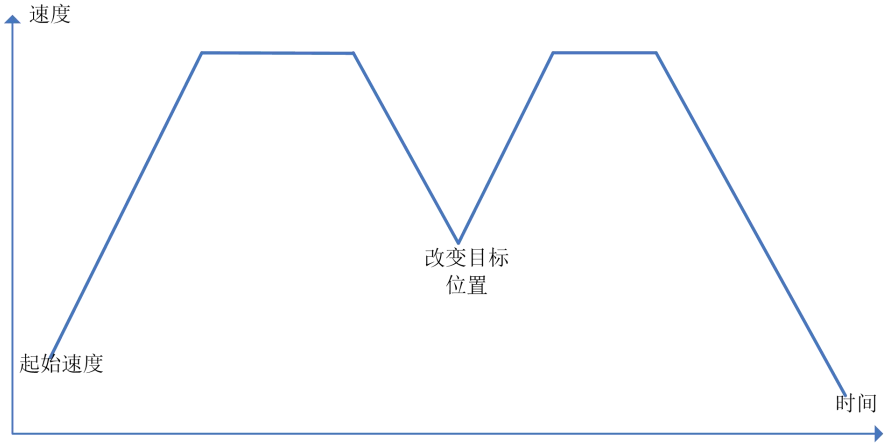


图 2-8 点位运动减速过程在线变位置速度曲线

实现这 2 个功能的函数如表 2-13 所示：

表 2-13 改变当前运行速度或目标位置相关函数说明

	名称	功能	参考
1	MC_change_pmove_speed	改变单轴点位运动中当前运行速度的函数	4.1.2.5.1
2	MC_reset_target_position	改变目标位置函数	4.1.2.5.1

例程：改变速度、改变终点位置

……

```
/*设置梯形曲线速度、加、减速时间*/
MC_set_profile_ex(g_handle,0,500,6000,500,0.02,0.01);
MC_pmove(g_handle,0,50000,0);                    //设置卡号、轴号、运
```

动距离 50000、相对坐标模式

```
If(“改变速度条件”)                          //如果改变速度条件满足，则执行改变速度命令
{
Curr_Vel= 9000；                             //设置新的速度
MC_change_pmove_speed(g_handle,0,Curr_Vel)；      //执行改变速度指令
}
If(“改变终点位置条件”)                       //如果改变终点位置条件满足，则执行改变终点位置命令
{
  MC_reset_target_position(g_handle,0,55000)；   //改变终点位置为 55000
}
……
```

如果将运动中的运行速度设置得小于起始速度，整个运动过程中将会以起始速度作恒速运动。

如果运动距离很短，当距离小于或等于(Max_Vel+Min_Vel)*Tacc 时，理论上速度曲线将变为三角形；但 MCX00A 系列运动控制器有自动调整功能，将三角形的尖峰去，以避免速度变化太大发生冲击现象。请参见图 2-9 所示：

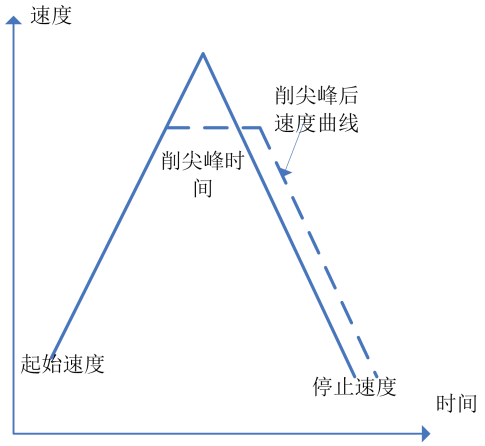


图 2-9 梯形速度曲线在线削尖峰功能

控制轴运动的指令发出后，可以用 MC_check_done 函数检测电机当前运动状态，详情可参考 5.1.2.5.2。

2.2.7.1.2 S 形速度曲线运动模式

梯形速度曲线虽然简单，但它的速度曲线不平滑，其加速度有突变，因而运

动中有冲击现象，容易引起机器噪声和传动机构的磨损。若将加速度改为线性变化，则速度曲线相应将变得光滑，升速和减速阶段均变得象 S 形形状。采用此种速度曲线，运动更平稳，且有助于缩短加速过程、降低运动装置的振动和噪声，以及延长机械传动部分的寿命。

设置 S 形速度曲线及其点位运动的函数如表 2-14 所示：

表 2-14 S 形速度控制相关函数说明

	名称	功能	参考
1	MC_set_s_profile	设置 S 型速度曲线的 S 段参数	4.1.2.5.1
2	MC_get_s_profile	读取 S 型速度曲线的 S 段参数	4.1.2.5.1
3	MC_pmove	让指定轴以对称 S 形速度曲线作点位运动	4.1.2.5.1

注意：

a.由于执行 S 形速度曲线运动时机器的振动较小，用户可以加大加速度，即提高速度曲线上线性升速区域的斜率，从而缩短加速或减速时间，从而缩短整个运动的时间。因此，S 形曲线在运动速度要求十分高的设备中被广泛使用；

b.使用 S 形曲线的目的是产生平滑运动，但是如果因为距离太短或加速太慢原因导致电机速度在加速段不能升至设定的最大值 Max_Vel 时，理论上加速段将突然切换至减速段，从而导致在速度曲线的中部出现尖三角，并因此引起该轴出现较大震动和相关问题。为了避免出现这种问题，MCX00A 系列内置有自动调整功能，使得加减速段的过渡保持平滑，如图 2-10 所示。

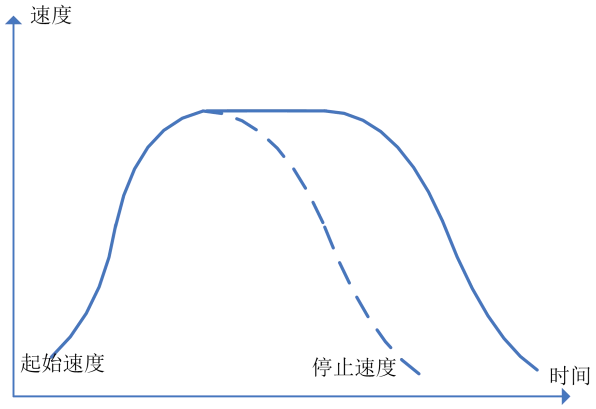


图 2-10 S 形速度曲线

单轴运行的情况下，S 形速度曲线运动过程中也可以调用 MC_change_pmove_speed 和 MC_reset_target_position 函数实时改变运行速度和目标位置，如图 2-11 所示。

……

```
/*设置梯形曲线速度、加、减速时间*/
MC_set_profile_ex(g_handle,0,500,6000,300,0.02,0.01);
MC_set_s_profile(g_handle,0,0.5);
MC_pmove(g_handle,0,50000,0);                    //设置卡号、轴号、运动距离 50000、相对坐标模式
If(“改变速度条件”)          //如果改变速度条件满足，则执行改变速度命令
{
Curr_Vel= 9000;                                  //设置新的速度
MC_change_pmove_speed(g_handle,0,Curr_Vel);  //执行改变速度指令
}
If(“改变终点位置条件”)                    //如果改变终点位置条件满足，则执行改变终点位置命令
{
MC_reset_target_position(g_handle,0,55000);      //改变终点位置为 55000
}
……
```

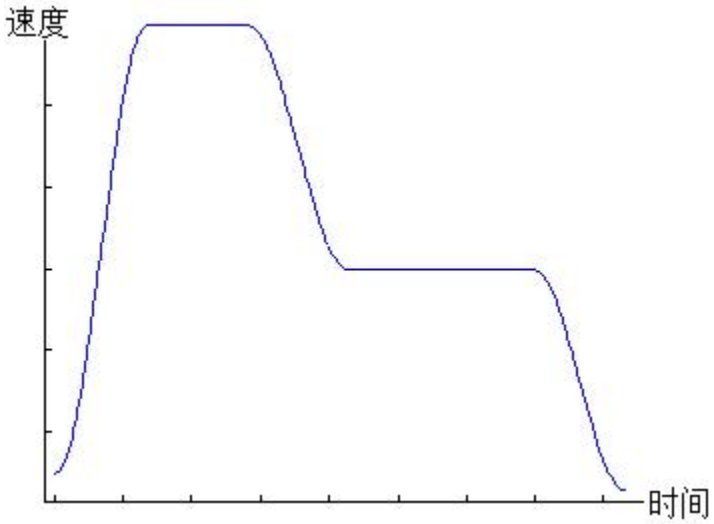


图 2-11 非对称 S 型曲线在线变速度和变位置

2.2.7.1.3 连续运动模式及速度控制函数

连续运动模式中，MCX00A 系列控制器可以控制电机以梯形或 S 形速度曲线在指定的加速时间内从起始速度加速至运行速度，然后以该速度持续运行，直至调用停止指令或者该轴遇到限位信号才会按启动时的速度曲线减速停止。连续运动的函数如表 2-15 所示：

表 2-15 连续运动及速度控制相关函数说明

	名称	功能	参考
1	MC_vmove	让指定轴加速到指定的运行速度后，连续运行	4.1.2.5.2
2	MC_check_done	检测指定轴的运动状态，停止或是在运行中	4.1.2.5.2
3	MC_change_speed	实时改变指定轴的当前运动模式下的运动速度	4.1.2.5.2
4	MC_read_current_speed	读取指定轴的当前运动速度	4.1.2.5.2
5	MC_decel_stop	指定轴减速停止	4.1.2.5.2
6	MC_imd_stop	使指定轴立即停止，没有减速过程	4.1.2.5.2
7	MC_emg_stop	使所有运动轴紧急停止	4.1.2.5.2
8	MC_check_done_all	检测所有轴的运动状态	4.1.2.5.2
9	MC_check_done_reason	检测指定轴的运动状态并返回停止原因	4.1.2.5.2

2.2.7.1.4 加减速过程的距离（脉冲数）计算

对于梯形速度曲线运动，加减速段的运动距离（脉冲数）可以按以下公式计算：

Dacc =（1/2）×（Max_Vel+Min_Vel）× Tacc

Ddec =（1/2）×（Max_Vel+Min_Vel）× Tdec

其中：Dacc，Ddec 分别为加速段距离和减速段距离；

Min_Vel，Max_Vel 为起始速度和运行速度；

Tacc，Tdec 为加速时间和减速时间。

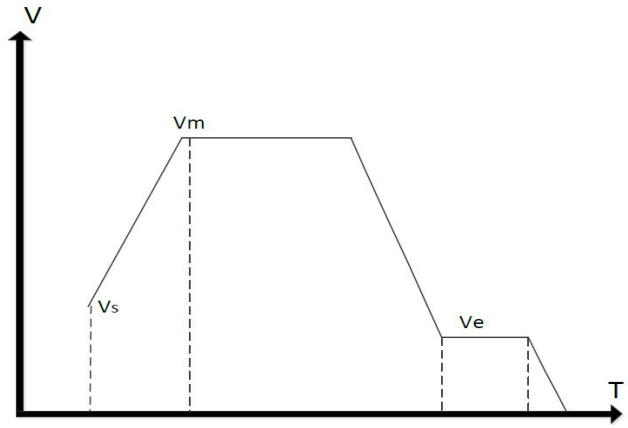
以上公式也完全适合于 S 曲线的情况。

2.2.6.1.9 软着陆功能

软着陆功能主要是让点位运动先以一个比较大的速度启动运行，当轴马上要接近目标位置时，减速至设定的低速速度，然后保持匀速到达目标位置，从而达到缓降的效果。

这样可以大幅度降低运动部件对接触点的冲击，尤其是用于防止焊头靠近晶圆时划伤或者压碎芯片，同时又能保证整体运动的效率，还可以降低对驱动器和设备结构的要求。

如下图所示，控制器将点位运动分为两段，第一段以 vs, vm, ve 进行规划，第二段以最大速度 ve 进行规划。



相关函数如表2-17-4 所示：

表 2-17-4 软着陆功能相关函数说明

	名称	功能	参考
1	MC_pmove_flex	使指定轴做点位运动（软着陆）	5.1.2.5.1

例程：软着陆功能

//设置控制器 0 号轴起始速度为 500 脉冲/秒、运行速度为 6000 脉冲/秒、结束速度为 300 脉冲/秒、加速时间为 0.02 秒、减速时间为 0.01 秒。

MC_set_profile_ex(g_handle,0,500,6000,300,0.02,0.01)；

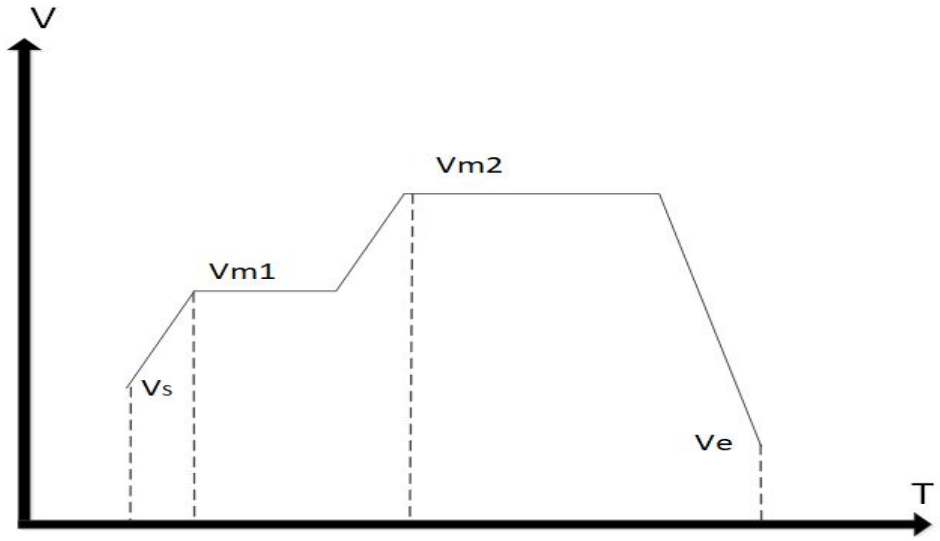
//设置控制器 0 号轴、运动距离为 50000 个脉冲、绝对坐标，低速长度为 2000 脉冲，低速速度为 1000 脉冲/秒，并开始执行运动。

MC_pmove_flex(g_handle,0,50000,1,2000,1000)；

2.2.6.1.10 缓拉功能

缓拉功能主要是让点位运动启动时以一个比较低的速度匀速运动，然后运动指定的低速长度，当低速段结束后快速加速到最大速度，最终到达目标位置，从而保证点位运动启动时比较平稳，同时又能保证整体运动的效率。常用于点胶结构点胶完成后的断胶动作。

如下图所示，控制器将点位运动分为两段，第一段以 vs（起始速度）加速到 Vm1（低速速度），然后保持匀速走完低速长度，第二段是当低速段结束后快速加速到 Vm2（最大速度），最终到达目标位置，并减速到 Ve（停止速度）正好运动停止。



相关函数如表2-17-5 所示：

表 2-17-5 缓拉功能相关函数说明

	名称	功能	参考
1	MC_pmove_flex_ex	使指定轴做点位运动（缓拉）	5.1.2.5.1

例程：缓拉功能

//设置控制器 0 号轴起始速度为 500 脉冲/秒、运行速度为 6000 脉冲/秒、结束速度为 300 脉冲/秒、加速时间为 0.02 秒、减速时间为 0.01 秒。

MC_set_profile_ex(g_handle,0,500,6000,300,0.02,0.01)；

//设置控制器 0 号轴、运动距离为 50000 个脉冲、绝对坐标，低速长度为 2000 脉冲，低速速度为 1000 脉冲/秒，并开始执行运动。

MC_pmove_flex_ex(g_handle,0,50000,1,2000,1000)；

2.2.6.1.12 反向间隙功能

反向间隙误差是指由于机械传动结构间隙的存在。执行部件在运动过程中，轴从正向运动切换到负向运动时，或者从负向运动切换到正向运动时，执行部件的运动量与理论量存在误差，最后将反映为叠加至工件上的加工精度的误差。为了消除反向间隙产生的误差，提高机器的加工精度和定位精度，MCX00A 系列控制器提供了反向间隙补偿功能。用户只要在初始化的时候调用反向间隙误差补偿功能指令 MC_set_backlash，反向间隙误差补偿功能将会生效；也可以通过指令 MC_set_backlash，把补偿值设为 0 来关闭反向间隙误差补偿功能。

我们一般认为轴的回零方向是不存在反向间隙的。所以当轴把运动方向从回

零方向切换到非回零方向时，控制器会自动进行反向间隙补偿；当轴把运动方向从非回零方向切换回零方向时，控制器则不会进行任何操作。

轴的回零方向通过 MC_config_home_mode 函数配置。

相关函数如表2-17-7 所示：

表 2-17-7 反向间隙功能相关函数说明

	名称	功能	参考
1	MC_set_backlash	设置间隙补偿值	5.1.2.9
2	MC_get_backlash	读取间隙补偿值	5.1.2.9

例程：反向间隙功能

```
//设置反向间隙功能补偿值为 10
MC_set_backlash(g_handle,0,10);
//设置控制器 0 号轴起始速度为 500 脉冲/秒、运行速度为 6000 脉冲/秒、结束速度为 300 脉冲/秒、加速时间为 0.02 秒、减速时间为 0.01 秒。
MC_set_profile_ex(g_handle,0,500,6000,300,0.02,0.01);
//设置控制器 0 号轴、运动距离为 50000 个脉冲、绝对坐标，并开始执行点位运动。
MC_pmove(g_handle,0,50000,1);
```

2.2.7.2 插补运动

2.2.7.2.1 直线插补运动

插补运动不但能保证起点、终点位置准确外，X 轴和 Y 轴的脉冲是按照直线斜率成比例发出的，所以在插补运动过程中的每一个时刻，其运动轨迹与理论曲线的误差总是小于一个脉冲当量，如图 2-14 所示:

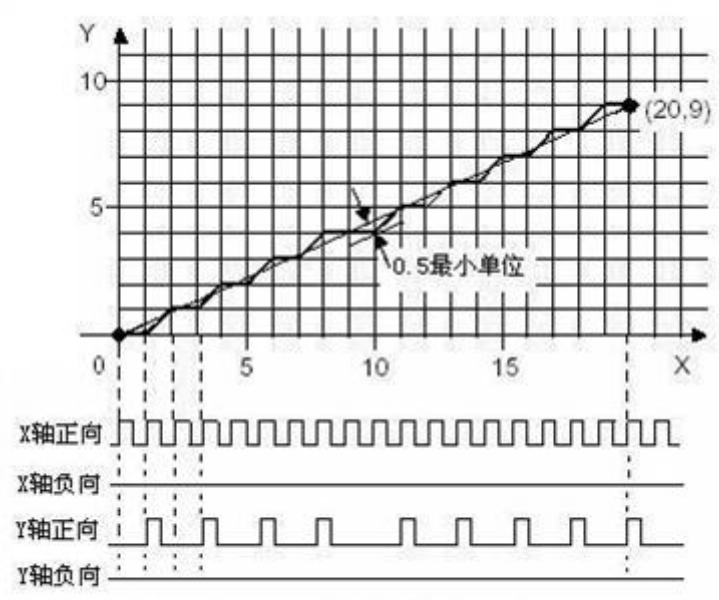


图2-14 直线插补示意图

MCX00A 系列运动控制器可以进行任意 2 轴、3 轴和 4 轴甚至多轴直线插补，插补工作由控制器上硬件执行，用户只需将插补运动的速度、加速度、终点位置等参数写入相关函数， 而无需介入插补过程中的计算工作。

二轴直线插补：

如图 2-15 所示，2 轴直线插补从 P0 点运动至 P1 点，X、Y 轴同时启动，并同时到达终点；X、Y 轴的运动速度之比为Δ X：Δ Y，二轴合成的矢量速度为：

$$\frac{\Delta P}{\Delta t}=\sqrt{\left(\frac{\Delta x}{\Delta t}\right)^2+\left(\frac{\Delta Y}{\Delta t}\right)^2}$$

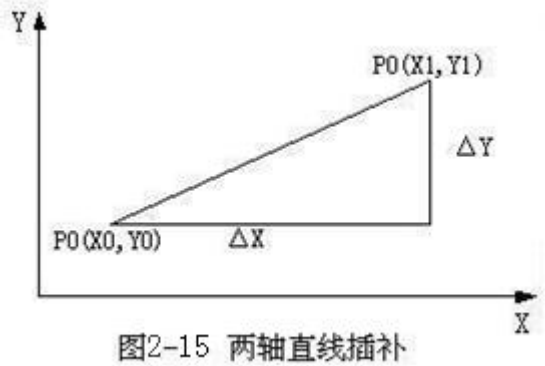


图2-15 两轴直线插补

三轴直线插补：

如图 4-17 所示，XYZ 3 轴直线插补从 P0 点运动至 P1 点。插补过程中 3 轴

的速度比为Δ X：Δ Y：Δ Z，三轴合成的矢量速度为：

$$\frac{\Delta P}{\Delta t}=\sqrt{(\frac{\Delta x}{\Delta t})^2+(\frac{\Delta Y}{\Delta t})^2+(\frac{\Delta Z}{\Delta t})^2}$$

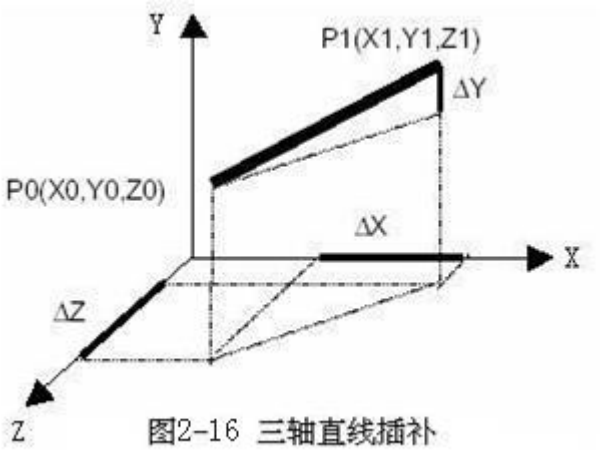


图2-16 三轴直线插补

四轴直线插补：

4 轴插补可以理解为在 4 维空间里的直线插补。一般情况是 3 个轴进行直线插补，另一个旋转轴也按照一定的比例关系和这条空间直线一起运动。其合成矢量速度为：

$$\frac{\Delta P}{\Delta t}=\sqrt{(\frac{\Delta x}{\Delta t})^2+(\frac{\Delta Y}{\Delta t})^2+(\frac{\Delta Z}{\Delta t})^2+(\frac{\Delta U}{\Delta t})^2}$$

直线插补运动相关函数如表2-18-1所示：

表 2-18-1 直线插补运动相关函数说明

	名称	功能	参考
1	MC_set_vector_profile_muticoor	插补运动参数设置（带插补器参数）	4.1.2.6.1
2	MC_line2_muticoor	指定任意两轴直线插补（带插补器参数）	4.1.2.6.1
3	MC_line3_muticoor	指定任意三轴直线插补（带插补器参数）	4.1.2.6.1
4	MC_lineN_muticoor	指定多轴直线插补（带插补器参数）	4.1.2.6.1

例程：直线插补

```
long poslist[3]={10000, 10000, 10000};
WORD iaxislist[3]={0,1,2};
```

```
MC_set_vector_profile_muticoor(g_handle,0,0,30000,0.01,0.01);
//设置插补运动参数
MC_lineN_muticoor(g_handle,0,3,iaxislist,poslist,0,300000,0.01,1);

//指定 3 轴做绝对位置模式下的直线插补运动
```

2.2.7.2.2 连续插补运动

MCX00A 系列卡允许多轴电机进行连续多段插补运动，而且通过运动速度设置函数的简单设置就可以实现多线段恒定合成速度连续插补，线段之间没有加/减速过程，连续插补的效果可以用图 2-18 来表示：

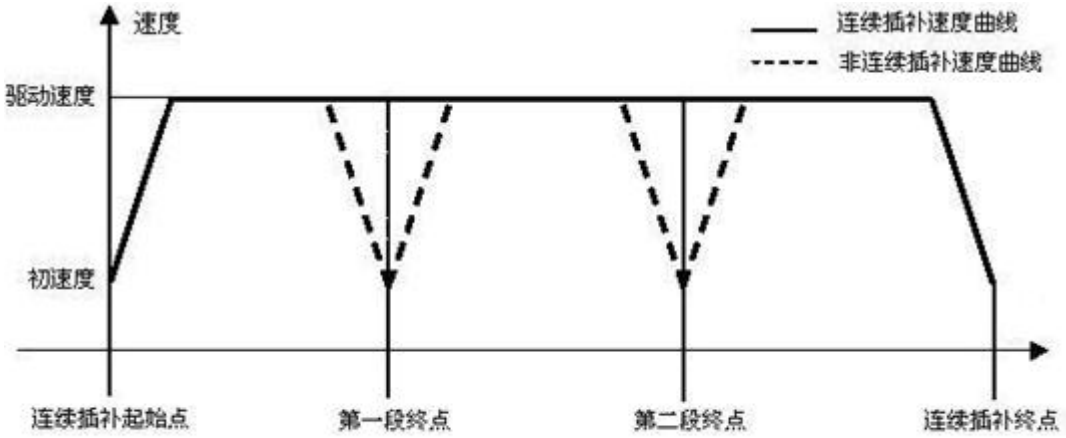


图2-18 连续插补运动

在 MCX00A 系列控制器上具有 4096 级硬件缓冲功能，能预先存储 4096 步运动的数据。控制器当前运动的数据存储在工作寄存器 R（register）中，下一个运动的数据存储在预置缓冲区 PR（pre-register）中，如图 2-19 所示。当前运动完成，PR 中的数据自动移至 R 中开始执行，此时寄存器 PR1 变为空。PC 机查询到 PR 为空后，即可补充下一运动的数据。如此不断循环，直至完成所有运动。

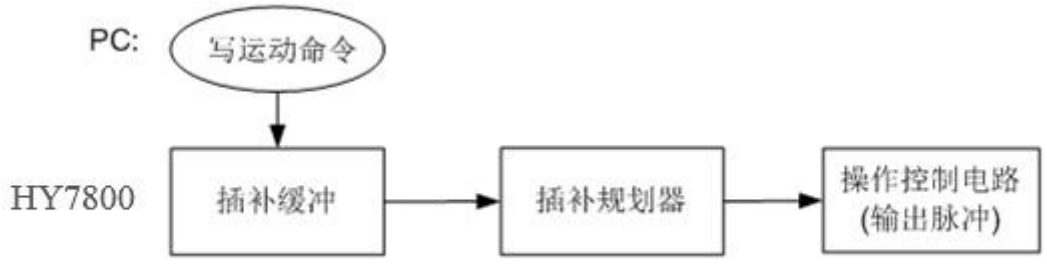


图2-19 多级缓冲寄存器连续插补的工作原理

连续插补注意事项：

1. 预置缓冲区不为空时，用户不能写入新的运动命令，否则将导致错误。

2. 如果在插补过程中，出现触发限位而停止，那么后续写入的资料和命令都是无效的。

3. 如果在插补运动结束后不调用 **MC_conti_close_list_muticoor** 函数，会导致 **MC_check_done** 函数返回值一直为 0，即检测到指定轴仍在运行之中。

例程：连续插补

```
long poslist[4]={0,0};
long cen[4]={100000,0};
WORD iaxislist[8]={0,1,2,3,4,5,6,7};
WORD iaxisgear[2] = {6,7};
long poslist0[4]={0,0};
long poslist1[4]={10000,0};
long poslist2[4]={10000,10000};
long poslist3[4]={10000,0};
long poslist4[4]={0,0};
long poslist5[4]={-10000,0};
MC_conti_set_mode_muticoor(g_handle,0,100,80000,0.05,0.00);//插补运动参数
MC_conti_open_list_muticoor(g_handle,0,2,iaxislist);  //打开插补缓冲区
MC_conti_lookahead_init_muticoor(g_handle,0,100,0.1,300000); //速度前瞻初始化
MC_conti_extern_lines_muticoor(g_handle,0,2,iaxislist,poslist0,60000,0.05,100,1);
MC_conti_extern_lines_muticoor(g_handle,0,2,iaxislist,poslist1,60000,0.05,100,1);
MC_conti_gear_muticoor(g_handle,0,1,iaxisgear,poslist1);  //刀向跟随
MC_conti_extern_lines_muticoor(g_handle,0,2,iaxislist,poslist2,60000,0.05,100,1);
MC_conti_gear_muticoor(g_handle,0,1,iaxisgear,poslist5);
MC_conti_extern_lines_muticoor(g_handle,0,2,iaxislist,poslist3,60000,0.05,100,1);
MC_conti_io_muticoor(g_handle,0,0xf000,0x0000);
MC_conti_delay_muticoor(g_handle,0,2000);  //缓冲区延时
MC_conti_io_muticoor(g_handle,0,0xf000,0xf000);  //插补缓冲区 IO
MC_conti_extern_lines_muticoor(g_handle,0,2,iaxislist,poslist4,60000,0.05,100,1);
MC_conti_pushdata_muticoor(g_handle,0,1);  //将前瞻缓冲区数据压入控制器
MC_conti_start_list_muticoor(g_handle,0);  //启动插补运动
MC_conti_close_list_muticoor(g_handle,0);  //关闭插补缓冲区
```

2.2.7.2.3 连续插补 IO 控制

MCX00A 系列卡在连续插补运动过程中支持 IO 控制，可以精确地把 IO 触发事件与连续、平滑的多轴插补运动轨迹深度融合。它确保了在外部设备（如激光器、电磁阀）动作与机械运动之间达到高精度同步。

表 2-18-3 连续插补 IO 控制相关函数说明

	名称	功能	参考
1	MC_conti_io_muticoor	插补缓冲区 IO	5.1.2.6.5
2	MC_conti_io_action_remained	查询插补 IO 剩余缓冲大小	5.1.2.6.5
3	MC_conti_clear_io_action	清除插补 IO 缓冲	5.1.2.6.5
4	MC_conti_delay_outbit_to_start	连续插补中相对于轨迹段起点 IO 滞后输出	5.1.2.6.5
5	MC_conti_delay_outbit_to_stop	连续插补中相对于轨迹段终点 IO 滞后输出	5.1.2.6.5
6	MC_conti_ahead_outbit_to_stop	连续插补中相对于轨迹段终点 IO 提前输出	5.1.2.6.5
7	MC_conti_pwm_muticoor	在插补缓冲中添加 PWM 输出	5.1.2.6.5
8	MC_conti_canio_da_muticoor	在插补缓冲中添加 DA 输出	5.1.2.6.5
9	MC_conti_wait_input_muticoor	连续插补等待 IO 输入信号再向下运行	5.1.2.6.5

例程：连续插补 IO 控制

```
long  poslist[4]={0,0};
//当前 XY 轴分别对应轴 0 和轴 1
WORD  iaxislist[2]={0,1};
long  poslist0[2]={0,0};
long  poslist1[2]={1000,3000};
long  poslist2[2]={3000,5000};
long  poslist3[2]={5000,6000};
long  poslist4[2]={0,0};
long  poslist5[2]={-10000,0};
//清除插补 IO 缓冲
MC_conti_clear_io_action(g_handle,0,0);
//打开插补列表
MC_conti_open_list_muticoor(g_handle,0,2,iaxislist);
//设置插补速度参数
MC_conti_set_mode_muticoor(g_handle,0,100,10000,0.05,0.00);
//运动到起点 0,0（绝对位置）
MC_conti_extern_lines_muticoor(g_handle,0,2,iaxislist,poslist0,10000,0.05,100,1);
```

```
//到达 A 点时输出 out5，输出脉冲时间为 10ms，输出电平为低电平（绝对位置）
MC_conti_delay_outbit_to_stop(g_handle,0,5,0,0,10);
//运动到 A 点（绝对位置）
MC_conti_extern_lines_muticoor(g_handle,0,2,iaxislist,poslist1,10000,0.05,100,1);
//到达 B 点时输出 out5，输出脉冲时间为 10ms，输出电平为低电平（绝对位置）
MC_conti_delay_outbit_to_stop(g_handle,0,5,0,0,10);
//运动到 B 点（绝对位置）
MC_conti_extern_lines_muticoor(g_handle,0,2,iaxislist,poslist2,10000,0.05,100,1);
//到达 C 点时
MC_conti_extern_lines_muticoor(g_handle,0,2,iaxislist,poslist3,10000,0.05,100,1);
//启动插补
MC_conti_start_list_muticoor(g_handle,0);
//关闭插补缓冲
MC_conti_close_list_muticoor(g_handle,0);
```

2.2.7.2.4 速度前瞻功能

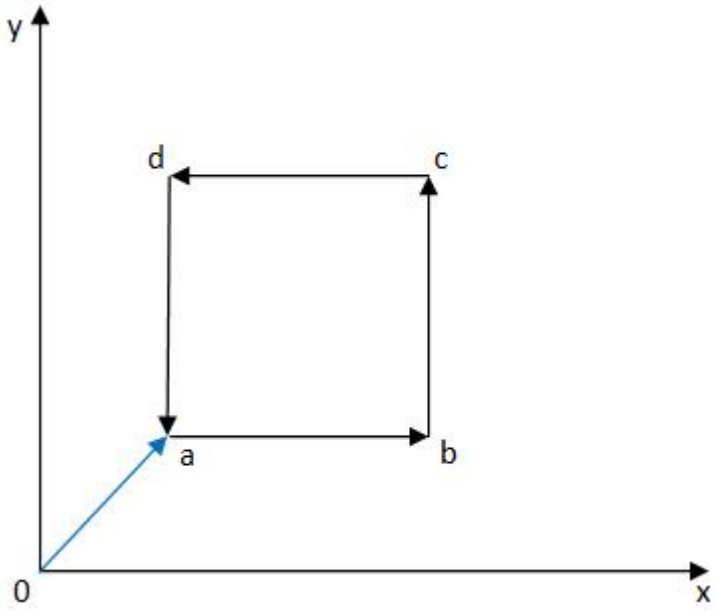
在数控加工等应用中，加工路径的尖锐拐角及高曲率容易引起过切、机床震动和数据饥饿等现象，严重影响加工质量和加工速度。因此要求对电机进行平滑的控制，对速度轨迹提前规划，实现速度的平滑过渡，防止过切等现象，能有效保证较高的加工效率及加工精度。以防止较大的冲击影响零件的加工质量。MCX00A 系列运动控制器的前瞻预处理功能可以根据用户的运动路径计算出平滑的速度规划，减少机床的冲击，从而提高加工精度。相关函数如表 2-19 所示：

表 2-19 速度前瞻相关函数说明

	名称	功能	参考
1	MC_conti_lookahead_init_muticoor	速度前瞻初始化（带插补器参数）	4.1.2.6.4

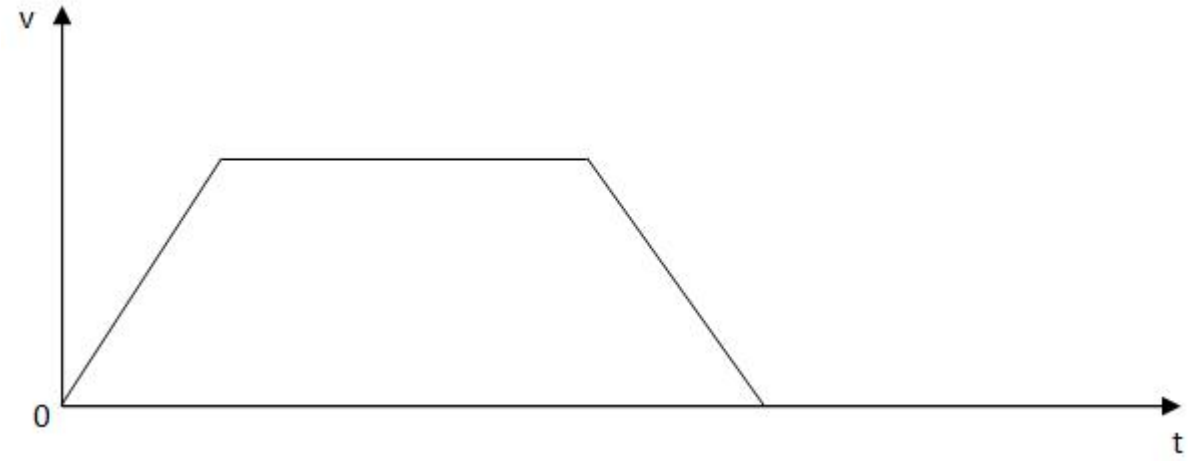
下面来举例说明有速度前瞻和没有速度前瞻所体现的区别：

假设机床将要加工一个长方形的产品，刀具所走的轨迹如下图 2-20（1）所示，设从 a 点到 b 点有 2000 个单位长度，有 20 段规划。b 点到 c 点距离 1000 个单位长度，有 10 段规划。每段规划 100 个单位长度。

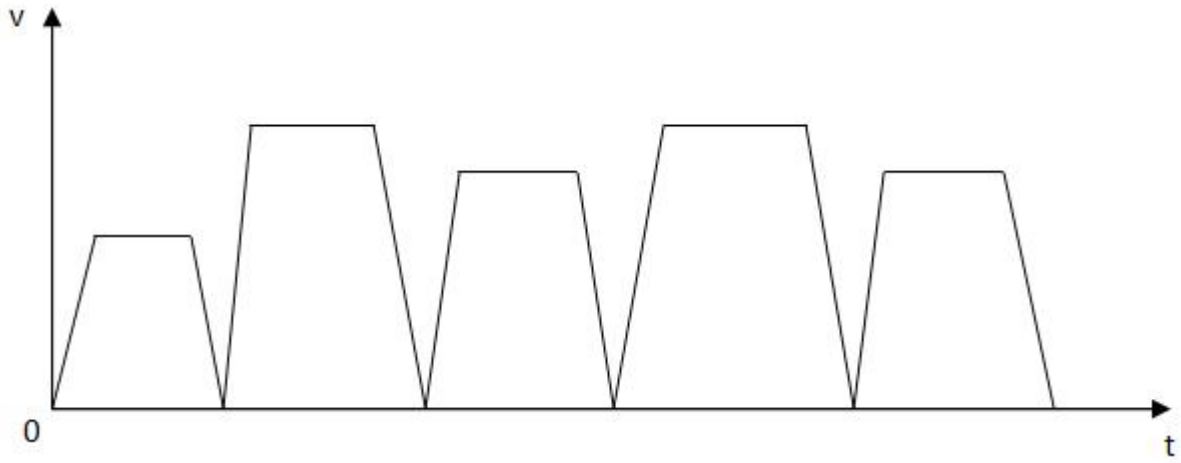


（1）

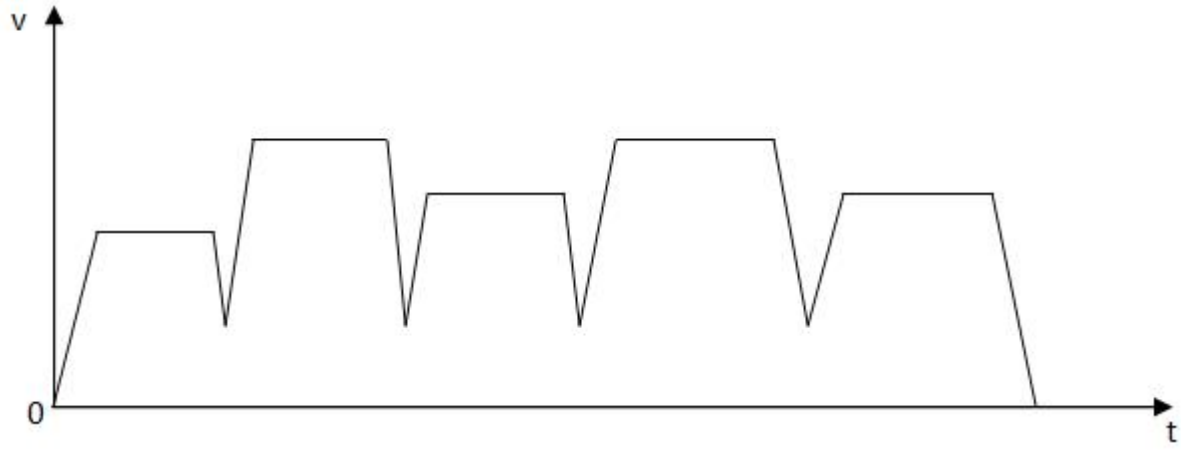
如果按照图（2）所示的进行速度规划，即在拐角处不减速，会导致加工精度在一定程度上降低，而且很有可能在拐弯处对刀具和零件造成较大冲击。可是如果按照图（3）所示的进行速度规划，即在拐角处减速为 0，可以最大限度上提高加工精度，但是会导致加工速度降下来。如果按照图（4）所示的速度规划，在拐角处将速度减到一个合理值，既可以满足加工精度又能提高加工速度，就是一个好的速度规划。



（2）



（3）



（4）

为了实现类似图(4)所示的好的速度规划，前瞻预处理模块不仅要知道当前运动的位置参数，还要提前知道后面若干段运动的位置参数，这就是所谓的前瞻。例如在对图(1)中的轨迹做前瞻预处理时，我们设定控制器提前读取 30 段运动轨迹到缓存区中，然后它会自动分析出在第 20 段将会出现拐点，并按照用户设定的拐角时间计算在拐角处的终点速度。前瞻预处理模块也会依照用户设定的最大加速度值计算速度规划，使任何加减速过程都不会超过这个值，防止对机械部分产生破坏性冲击力。

使用和不使用前瞻预处理功能模块的速度曲线对比图如图 2-21 所示：

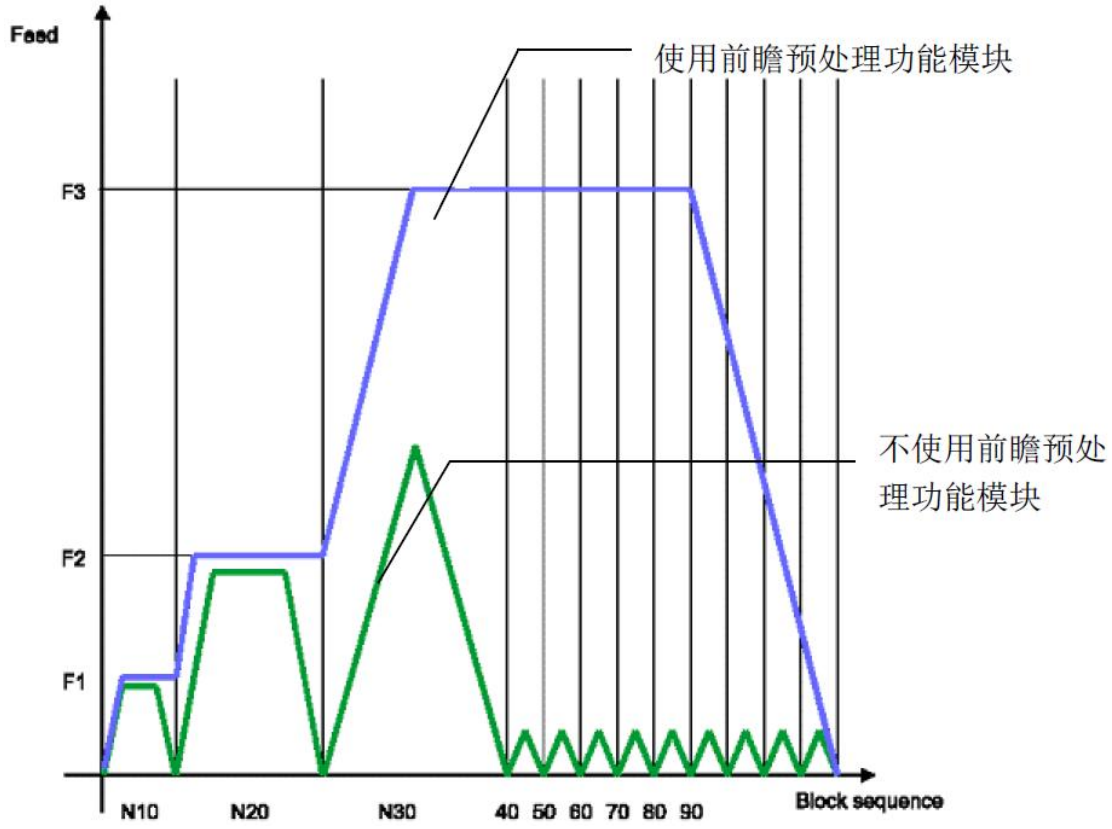


图 2-21 使用和不使用前瞻预处理功能模块的速度曲线对比图

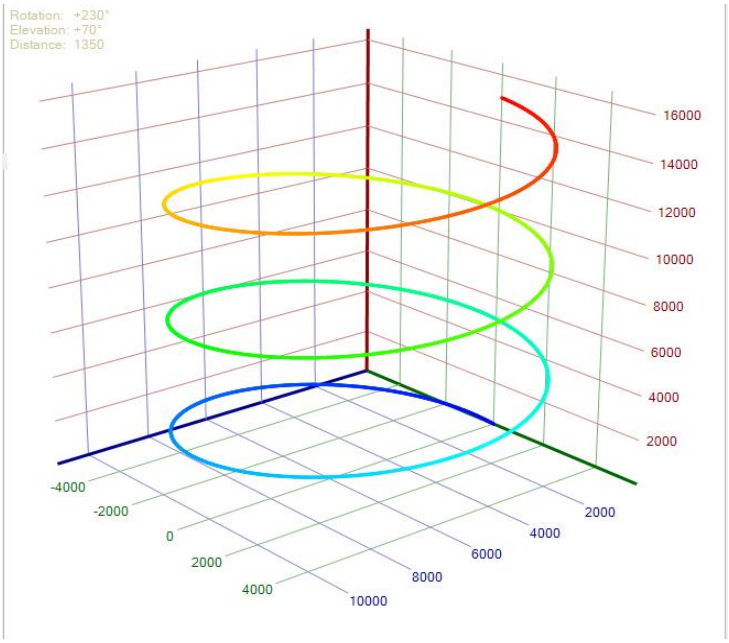
2.2.7.2.5 刀向跟随功能

刀向跟随，就是在插补运动的过程中，使非插补轴随着插补运动的合成位移的变化而变化，从而实现在加工过程中，刀具始终处于合适的加工方向和位置的工艺。相关函数如表 2-20 所示：

表 2-20 刀向跟随相关函数说明

	名称	功能	参考
1	MC_conti_gear_muticoor	刀向跟随功能（带插补器参数）	4.1.2.6.5

刀向跟随效果示意图如下图所示：在两轴（XY）圆弧插补运动的过程中，Z 轴作为刀向跟随轴，随着 XY 轴的插补运动向上抬升；当 XY 轴完成一段圆弧插补时，Z 轴刚好同时到达目标位置。



例程：刀向跟随功能

```
 long poslist[2]={0,0};
WORD iaxislist[2]={0,1};
WORD iaxisgear[4] = {2};//刀向跟随的轴号列表
long poslist0[4]={0,0};
long poslist1[4]={5000};//刀向跟随的位置列表
long poslist2[4]={10000};//刀向跟随的位置列表
long poslist3[4]={15000};//刀向跟随的位置列表
long pos[2] = {8000,8000};
long cen[2] = {0,0};

//设置连续插补运动参数
MC_conti_set_mode_muticoor(g_handle,0,100,4000,0.05,0.00);
//打开连续插补缓冲区
MC_conti_open_list_muticoor(g_handle,0,2,iaxislist);
//启动速度前瞻
MC_conti_lookahead_init_muticoor(g_handle,0,50,0.1,30000);

//设置刀向跟随
MC_conti_gear_muticoor(g_handle,0,1,iaxisgear,poslist1);
//启动两轴圆弧插补
```

```
MC_conti_extern_arc_muticoor(g_handle,0,iaxislist,pos,cen,1,2000,0.05,0.1,0);

//设置刀向跟随
MC_conti_gear_muticoor(g_handle,0,1,iaxisgear,poslist2);
//启动两轴圆弧插补
MC_conti_extern_arc_muticoor(g_handle,0,iaxislist,pos,cen,1,2000,0.05,0.1,0);

//设置刀向跟随
MC_conti_gear_muticoor(g_handle,0,1,iaxisgear,poslist3);
//启动两轴圆弧插补
MC_conti_extern_arc_muticoor(g_handle,0,iaxislist,pos,cen,1,2000,0.05,0.1,0);

//压入前瞻缓冲数据到连续插补缓冲区
MC_conti_pushdata_muticoor(g_handle,0,1);
//启动插补运动
MC_conti_start_list_muticoor(g_handle,0);
//关闭插补缓冲区
MC_conti_close_list_muticoor(g_handle,0);
```

2.2.7.3 电子凸轮

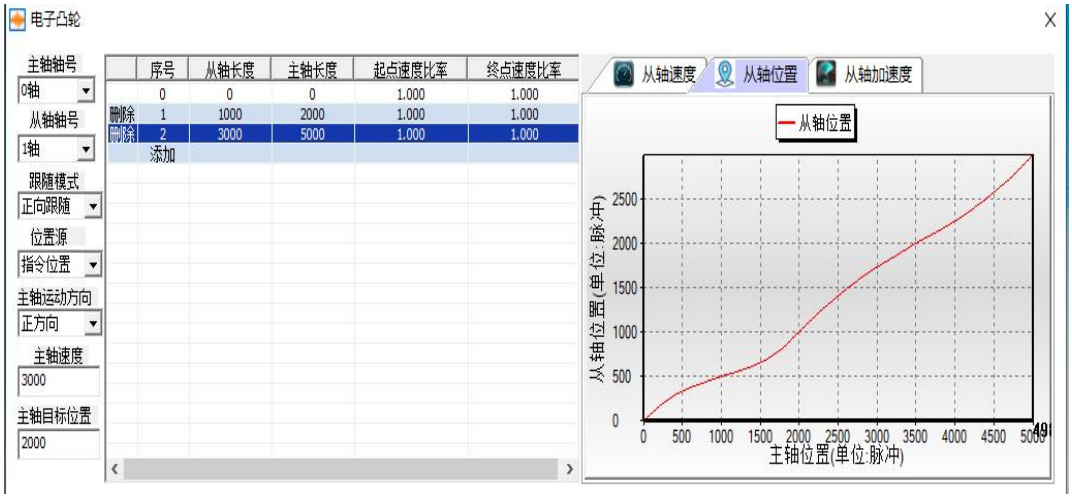
恒昱控制 MCX00A 系列控制器提供电子凸轮运动功能。电子凸轮是利用构造的凸轮曲线来模拟机械凸轮，以达到机械凸轮系统相同的凸轮轴与主轴之间相对运动。我们把被跟随的轴叫主轴，把跟随的轴叫从轴。在电子凸轮模式下，能够实现单个或者多个从轴跟随主轴运动。电子凸轮特点如下：

(1) 从轴可以跟随主轴的正向运动、负向运动或者双方向运动。

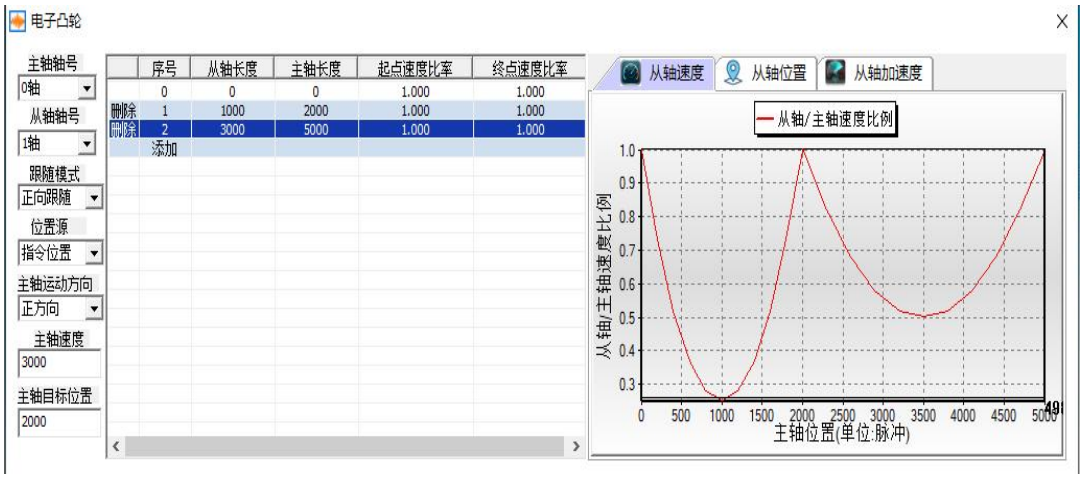
(2) 与从轴之间的同步，通过用户设定的多个数据段自动规划，每个数据段包含主轴位移，从轴位移，从轴/主轴的速度比例三个参数，即主轴在到达目标位置时，从轴也到达设定的目标位置，这个过程中的速度由设定的速度比例决定。

(3) 在电子凸轮模式下可以自由设置从轴速度和主轴速度的比例，当主轴速度变化时，从轴会根据设定好的速度比例自动改变速度。

从轴位置-主轴位置示意图、从轴/主轴速度比例-主轴位置示意图如下图所示：



从轴位置-主轴位置示意图



从轴/主轴速度比例-主轴位置示意图

表 2-21-3 电子凸轮相关函数说明

	名称	功能	参考
1	MC_ecam_disengage	取消凸轮主轴和从轴链接	5.1.2.11
2	MC_ecam_engage	启动凸轮主轴和从站链接	5.1.2.11
3	MC_ecam_fifo_datain	凸轮数据压入	5.1.2.11
4	MC_ecam_fifo_clear	凸轮数据清除	5.1.2.11
5	MC_ecam_fifo_remian	查询凸轮数据剩余段数	5.1.2.11
6	MC_ecam_get_curmark	读取当前正在运行的凸轮数据段标号	5.1.2.11
7	MC_ecam_state	查询凸轮运动的状态	5.1.2.11
8	MC_ecam_set_repeatlength	设置往复运动的模	5.1.2.11
9	MC_ecam_get_repeatlength	读取往复运动的模	5.1.2.11

10	MC_ecam_fifo_ioaction	在凸轮点后增加一个凸轮顶杆事件	5.1.2.11

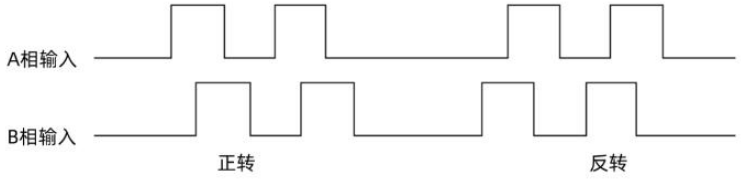
例程：

```
//跟随轴退出和任何参考轴链接
  MC_ecam_disengage(g_handle,1,0);
//清除凸轮表
  MC_ecam_fifo_clear(g_handle,1);
//添加数据点，参考轴运动到-500 时，跟随轴运动到 1000，跟随轴速度到参考轴速度
  MC_ecam_fifo_datain(g_handle,1,1000,-500,1.0,1.0,1,1);
//添加数据点，参考轴运动到-1500 时，跟随轴运动到 2000，速度保持和参考轴速度相同
  MC_ecam_fifo_datain(g_handle,1,2000,-1500,1.0,1.0,1,2);
  //添加数据点，参考轴运动到-2000 时，跟随轴运动到 3000，跟随轴减速停止
  MC_ecam_fifo_datain(g_handle,1,3000,-2000,-1.0,0.0,1,3);
  //启动跟随运动
  MC_ecam_engage(g_handle,1,0,-1000,0,0);
```

2.2.7.4 手轮功能

电子手轮也称为手动脉冲发生、手脉、手摇脉冲发生器等。用于数控机床、印刷机械等的零位补正和信号分割。当手轮旋转时，编码器产生与手轮运动相对应的信号。通过数控系统选定坐标并对坐标进行定位。

手动脉冲发生器的中心有轴的光电码盘，其上有环形通、暗的刻线；当用户摇动手轮后，由光电发射和接收器件读取，获得 2 组正弦波信号 HA、HB，每个正弦波相差 90 度相位差。由于 HA、HB 两信号相差 90 度，可通过判断 A 相或 B 相脉冲超前，给出正转脉冲或反转脉冲；当控制器接收到脉冲信号后，将脉冲信号转换为数值信号，并进一步将这个信号发送到输出接口，从而控制伺服电机正转或反转。这种转换和处理使得电子手轮能够精确地控制设备的运动，无论是微调还是大范围的移动。



手轮介绍：

1.通过手轮上的“轴选择旋钮”选择需要移动的坐标轴；

2.通过“倍率选择旋钮”选择合适的移动倍率（×1/×10/×100）；

3.旋转“手轮摇柄”移动坐标轴。顺时针旋转为正向移动，逆时针旋转为负向移动，旋转速度快慢可以控制坐标轴的运动速度；

手轮与控制器：

1.电子手轮的轴选择旋钮上方的选项如 X、Y、Z 会分别对应出一条输出信号线，可根据实际应用的电子手轮引脚定义，分别接入控制器的输入口上。当用户切换电子手轮上的轴选择旋钮选项时，轴选项对应的信号线会输出信号到控制器的输入口上，此时上位机软件可以根据输入口信号状态判断用户拨到了哪个轴选项，并调用相关的控制器函数（MC_set_handwheel_inmode），改变手轮功能控制的轴号。

2. 电子手轮的倍率旋钮上方的选项如 X1、X10、X100 会分别对应出一条输出信号线，可根据实际应用的电子手轮引脚定义，分别接入控制器的输入口上。当用户切换电子手轮上的倍率旋钮选项时，倍率选项对应的信号线会输出信号到控制器的输入口上，此时上位机软件可以根据输入口信号状态判断用户拨到了哪个倍率选项，并调用相关的控制器函数（MC_set_handwheel_inmode），改变手轮功能的倍率。

表 2-21-4 手轮运动相关函数说明

	名称	功能	参考
1	MC_set_handwheel_inmode	设置手轮脉冲信号的计数方式	5.1.2.10
2	MC_get_handwheel_inmode	读取手轮脉冲信号的计数方式	5.1.2.10
3	MC_handwheel_move	启动手轮运动	5.1.2.10

例程：

```
//设置当前轴的运动参数
MC_set_profile_ex(g_handle,0,0,10000,0,0.01,0.01);
//设置控制器第 0 轴的手轮脉冲信号计数方式
MC_set_handwheel_inmode(g_handle,0,0,10);
//启动控制器第 0 轴的手轮运动
MC_handwheel_move(g_handle,0);
```

2.2.8 通用 I/O 控制

恒昱控制 MCX00A 系列运动控制器硬件上为用户提供了通用的数字输入输出接口，MC400A 最多 20 路通用输入和最多 24 路通用输出，MC800A 最多 8 路通用输入和最多 24 路通用输出。用户可以使用这些数字 I/O 口用于输入开关信号、传感器信号等信号，或者输出继电器、电磁阀等输出设备的控制信号。相关的函数如表 2-28 所示：

表 2-28 通用 IO 相关函数说明

	名称	功能	参考
1	MC_read_inbit	读取指定控制器的某一个输入口的电平状态	4.1.2.11
2	MC_write_outbit	设置指定控制器的某一个输出口的电平状态	4.1.2.11
3	MC_read_outbit	读取指定控制器的某一个输出口的电平状态	4.1.2.11
4	MC_read_inport	读取指定控制器的全部通用输入口的电平状态	4.1.2.11
5	MC_read_outport	读取指定控制器的全部通用输出口的电平状态	4.1.2.11
6	MC_write_outport	设置指定控制器的全部通用输出口的电平状态	4.1.2.11
7	MC_reverse_outbit	输出口延时翻转	4.1.2.11

例程：通用输入输出

If(MC_read_inbit(g_handle,input1)==0) //读取控制器 INPUT1 口的状态，判断按键是否按下

```
{
MC_write_outbit(g_handle,out1, 0); //如果按键按下，控制器 OUT1 口输出为 0，LED 发光
}
else
{
MC_write_outbit(g_handle,out1, 1); //如果按键未按，控制器 OUT1 口输出为 1，LED 不亮
} ……
```

2.2.9 I/O 延时翻转功能

控制器支持 I/O 延时翻转功能。该函数执行后，首先输出一个与当前输出口电平相反的信号，到达设置的延时时间后，控制器会自动把输出口翻转一次电平。相关函数如表 2-28-2 所示：

表 2-28-2 I/O 延时翻转功能相关函数说明

	名称	功能	参考
1	MC_reverse_outbit	输出口延时翻转	5.1.2.12

例程：I/O 延时翻转功能

```
//设置控制器 out0 先输出一个与当前电平相反的信号，在 50ms 后翻转电平
MC_reverse_outbit(g_handle,0,50);
```

2.2.10PWM 输出信号

恒昱控制 MCX00A 系列运动控制器为用户提供了 4 路 PWM 输出信号。对应输出口为 12、13、14 和 15。

PWM 的频率是指 1 秒钟内信号从高电平到低电平再回到高电平的次数(一个周期)，频率的单位是赫兹。那么当频率为 50Hz 时，那么一个周期就是 20ms，1 秒钟就有 50 次 PWM 周期。（换算公式：T=1/f（周期=1/频率））

PWM 的占空比是指一个脉冲周期内，高电平的时间（脉宽时间）与整个周期时间的比例，单位：%（0%～100%）。那么如下图所示：

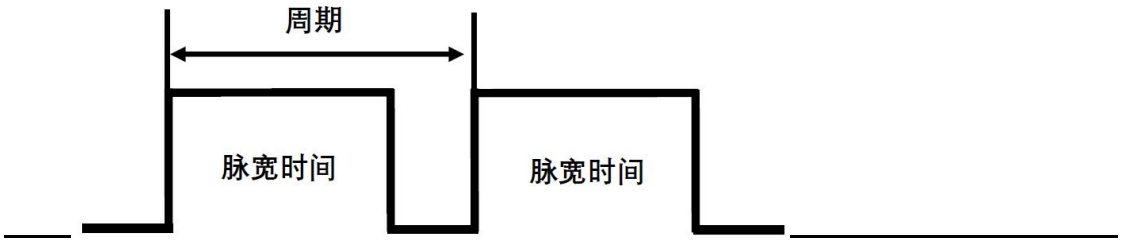


表 2-29 PWM 功能相关函数说明

	名称	功能	参考
1	MC_write_pwm	输出 PWM	4.1.2.12

例程：输出 PWM

MC_write_pwm（g_handle,1,50,100,1500）;//设置控制器，通道 1，占空比 50%，频率为 100HZ，输出时间为 1500ms。

2.2.11CAN 扩展 IO 控制

恒昱控制 MCX00A 系列运动控制器硬件上为用户提供了 CAN IO 的扩展模块，CAN IO 默认波特率为 500K，最多可以接四个 CAN IO 的扩展模块，一个 CAN IO 的扩展模块最多为 20 路输入和 20 路输出。

表 2-30 CAN IO 的扩展模块相关函数说明

	名称	功能	参考
1	MC_init_canio	初始化 CAN IO 的扩展模块	4.1.2.13
2	MC_canio_readinbit	读取 CAN IO 输入口的状态	4.1.2.13
3	MC_canio_readinport	读取 CAN IO 输入端口的值	4.1.2.13
4	MC_canio_readoutport	读取 CAN IO 输出口的状态	4.1.2.13
5	MC_canio_writeoutbit	设置 CAN IO 输出口的状态	4.1.2.13
6	MC_canio_writeoutport	设置 CAN IO 输出端口的值	4.1.2.13

例程：CAN 扩展 IO 控制

MC_init_canio(g_handle,1,1,0);//设置控制器 CAN 扩展 IO,CAN ID 号为 1，CAN IO 使能，最后一个参数保留。

MC_canio_readinbit(g_handle,1,0);//读取控制器 CAN 扩展 IO,CAN ID 号为 1，输入口位号为 0 的 CAN IO 输入口的状态。

MC_canio_readoutport(g_handle,1);//读取控制器 CAN 扩展 IO，CAN ID 号为 1 的所有 CAN IO 输出口的状态。

MC_canio_writeoutbit(g_handle,1,1,1);//设置控制器 CAN 扩展 IO，CAN ID 号为 1，输出口位号位 1 的 CAN IO 输出口为高电平状态。

2.2.12IO 口映射功能

恒昱控制 MCX00A 控制器支持把专用输入信号功能映射到其他输入信号

上，因此可以灵活使用所有输入口。

表 2-31-2 IO 映射功能相关函数说明

	名称	功能	参考
1	MC_set_axis_io_map	设置 IO 映射	4.1.2.15
2	MC_get_axis_io_map	读取 IO 映射配置	4.1.2.15

例程：

MC_set_axis_io_map(g_handle,0,3,6,0,1); //将急停信号映射至通用输入口 0

2.2.13 参数文件写入读取功能

恒昱控制 MCX00A 控制器支持把所有轴的专用信号配置写入参数文件（.ini 文件）进行保存，或者把参数文件（.ini 文件）里面保存的参数写入控制器，方便用户简单、快速、准确地初始化运动参数参数配置，也可以通过编辑参数文件内容直接更改控制器的参数配置。

表 2-31-3 参数文件写入读取功能相关函数说明

	名称	功能	参考
1	MC_LoadConfig	把参数文件（.ini）里面的保存的参数写入控制器	4.1.2.16
2	MC_ReadConfig	把控制器的参数配置读取出来保存在参数文件（.ini）	4.1.2.16

例程：

```
CString g_FilePath;
 LPTSTR p_filepath = g_FilePath.GetBuffer(0);
 //把对应的文件路径的参数文件加载到控制器
  MC_LoadConfig(g_handle,p_filepath);

CString g_FilePath;
 LPTSTR p_filepath = g_FilePath.GetBuffer(0);
 //把控制器的运动参数保存到指定的文件路径上的文件中
  MC_ReadConfig(g_handle,p_filepath);
```

2.2.14 信号软件滤波

恒昱控制 MCX00A 控制器支持对所有输入信号进行滤波。在电磁环境很差时，建议将滤波时间加大(比如设备有交流电机、变频器等)，增加设备的稳定性。可以通过下面所述的函数进行滤波时间配置，也可以直接通过参数文件导入。

通用输入口滤波函数为：MC_config_input_filter 和

MC_get_config_input_filter；

特殊信号，如限位、报警、急停等信号的滤波函数为：MC_config_special_input_filter 和 MC_get_config_special_input_filter；

原点信号的滤波函数为：MC_config_home_pin_filter 和 MC_get_config_home_pin_filter；

所有滤波时间的单位为 250us，默认通用输入的滤波时间为 3，特殊信号的滤波时间为 6，原点信号的滤波时间为 3。

相应函数参数说明，请参见后续章节。

2.3 演示程序

恒昱控制 MCX00A 系列运动控制器为了方便用户使用 C/C++或 VB 等编程软件对其进行应用开发，尤其针对典型的运动方式，例如单轴运动、回零动作、插补运动、编码器读取、数据锁存等等，提供了示例源代码，用户可以直接将软件 CD 中相应目录中的代码直接拷贝到您的程序工程中使用。

演示软件的设计，大大简化了用户的调试过程。将 MCX00A 系列的软件 CD 盘插入计算机光驱，在相应的目录下，例如“演示界面”，将其全部拷贝到计算机硬盘的任意指定位置后，运行 MCX00A 系列 MCCoreMotion.exe，即可对控制器的各项主要功能进行检测和使用，还可以通过使用这个软件对您的整个自动化系统进行初步的运行和调试。详细资料可参考： 4.演示软件及应用。

2.4 例子程序

恒昱控制 MCX00A 系列运动控制器为了方便用户使用 C/C++或 VB 等编程软件对其进行应用开发，尤其针对典型的功能，如：单轴运动、回原点、插补运动、编码器读取、数据锁存等等，提供了示例源代码。

用户可以直接将配套的 CD 中相应目录中的代码直接拷贝到您的程序工程中使用。

2.4.1 单轴运动例程

此例程为四个轴的单轴点位运动，为每个轴增加了设置点位运动的运行速度、加速度和运动距离的编辑框，同时例程中还设有启动，停止，减速停，急停，位置清零和退出功能按钮。这个例程用到了控制器的初始化/关闭函数，读取各运动轴当前速度/位置函数，设置轴运动速度函数/点位运动函数，轴的运动状态函数，减速停函数/急停函数。通过这个例程的学习可以了解 MCX00A 系列控制器基本函数的调用，详细代码请参考光盘例程“例 1_单轴运动”。

单轴运动例程主界面如图 2-22 所示：

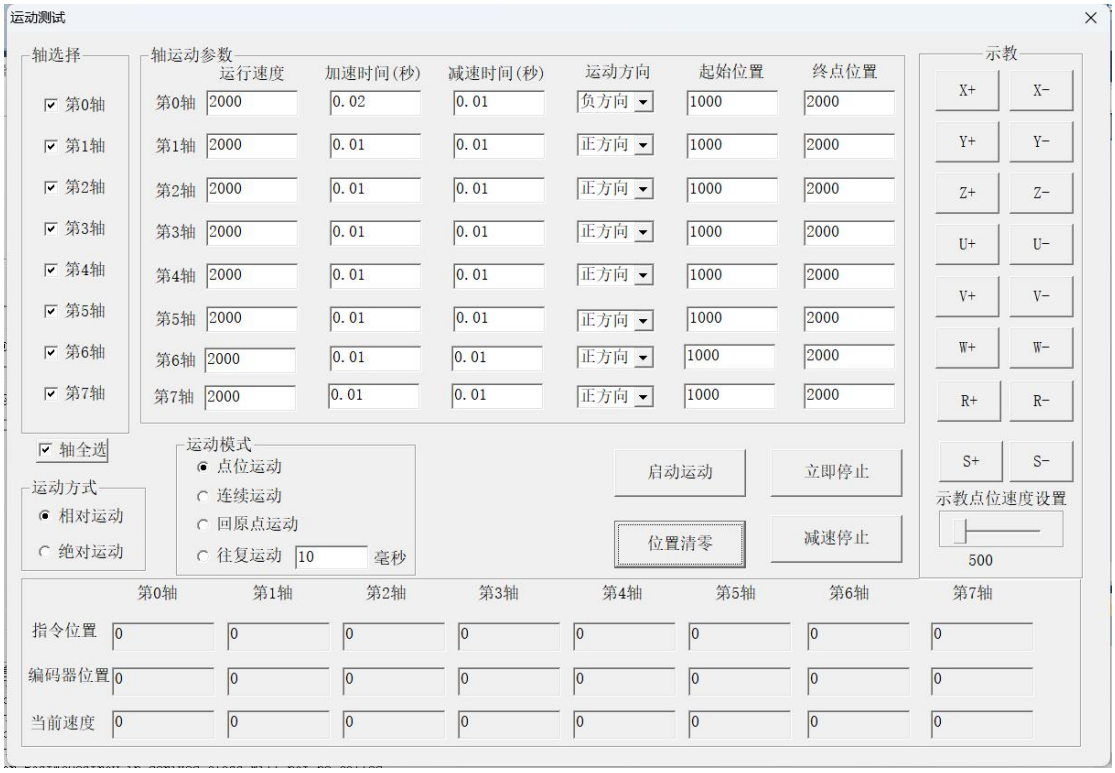


图2-22 单轴运动例程主界面

2.4.2 通用专用输入输出信号例程

此例程为 MCX00A 系列控制器通用数字 I/O 信号和每个轴的专用 I/O 信信号的检测例程。程序里有所有通用输入/输出信号的检测图标，可以对每个输出口的输出状态进行操作，并可以检测每个轴的专用信号的输入状态。详细代码请参考光盘例程“例 7_通用专用输入输出”。其主界面如图 2-25 所示：

图2-25 通用专用输入输出例程主界面

2.4.3 CAN 扩展输入输出信号例程

此例程是 MCX00A 系列控制器为用户提供了 CAN IO 扩展模块的检测例程。程序里面有所有 CAN IO 的输入/输出信号的检测图标，可以对每个输出口的输出状态进行操作，并可以检测每个输入口的输入状态。其主界面如图 2-26 所示。

图 2-26 CAN IO 的扩展模块输入输出例程主界面

3 附录

3.1 运动控制函数库

恒昱控制 MCX00A 系列运动控制器的运动控制函数库共有 32 类 300 个函数，分类列表如下：

3.1.1 函数列表

	函数名	描 述
初始化函数	MC_OpenEth	使用网络连接控制器
	MC_Close	关闭控制器，释放资源
	MC_OpenCom	关闭指定控制器
	MC_get_lib_version	读取函数库版本
	MC_get_total_axes	读取指定卡轴数
	MC_get_controller_sn	读取控制器唯一序列号
脉冲模式设置函数	MC_set_pulse_outmode	设定脉冲输出模式
	MC_get_pulse_outmode	读取脉冲输出模式
回原点函数	MC_config_home_mode	设置回原点模式
	MC_get_config_home_mode	读取回原点模式
	MC_home_move	启动回原点运动
位置计数器控制函数	MC_get_position	读取指定轴的指令位置
	MC_set_position	设定指定轴的当前位置
	MC_get_position_all	读取所有轴的指令位置
	MC_get_aimposition	读取指定轴目标位置
	MC_get_counter_flag	读取计数器标志
	MC_reset_counter_flag	复位计数器的计数标志
	MC_reset_clear_flag	复位计数器的清零标志
编码器相关函数	MC_counter_config	设定编码器的计数方式
	MC_get_counter_config	读取编码器的计数方式
	MC_set_encoder	设置指定轴编码器反馈脉冲计数值
	MC_get_encoder	读取指定轴编码器反馈脉冲计数值
	MC_get_encoder_all	读取所有轴编码器反馈脉冲计数值
	MC_config_encoder_dir	设置编码器的反馈脉冲计数方向
位置锁存相关	MC_config_latch_mode	设置锁存方式

函数	MC_get_config_latch_mode	读取锁存方式的设置
	MC_get_latch_value	读取编码器锁存器的值
	MC_get_latch_flag	读取锁存器标志
	MC_reset_latch_flag	复位锁存器标志
	MC_reset_latch_flag_ex	复位指定轴锁存器标志
	MC_triger_chunnel	选择全部锁存时的外触发信号通道
连续位置锁存相关函数	MC_conti_latch_config	设置连续锁存功能
	MC_conti_latch_get_config	读取连续锁存功能
	MC_conit_latch_clear	清除连续锁存缓存
	MC_conit_latch_count	读取连续锁存缓存已经缓存个数
	MC_conit_latch_value	读取连续锁存缓存数据
一维位置比较功能函数	MC_Compare_Config	设置比较器
	MC_Compare_GetConfig	读取比较器的设置
	MC_Compare_ClearAllPoint	清除所有比较点
	MC_Compare_AddPoint	添加比较点
	MC_Compare_GetCurPoint	读取当前比较点
	MC_Compare_GetPointsRunned	查询已经比较过的点
	MC_Compare_GetPoints_Space	查询可以加入的比较点数量
多组一维位置比较功能函数	MC_Compare_Set_PulseTime_Muti	设置功能号 9 的脉冲宽度
	MC_compare_set_filter_Muti	设置锁存器滤波频率
	MC_compare_get_filter_Muti	读取锁存器滤波频率
	MC_Compare_Config_Muti	设置比较器配置
	MC_Compare_GetConfig_Muti	读取比较器配置
	MC_Compare_ClearAllPoint_Muti	清楚所有比较点
	MC_Compare_AddPoint_Muti	添加比较点
	MC_Compare_GetCurPoint_Muti	读取当前比较点
	MC_Compare_GetPointsRunned_Muti	查询已经比较过的比较点
	MC_Compare_GetPointsSpace_Muti	查询可以加入的比较点数量
高速位置比较功能	MC_hcmp_config	配置高速比较输出
	MC_hcmp_get_config	读取配置高速比较输出
	MC_hcmp_clear_points	清除高速比较输出数据
	MC_hcmp_add_point	高速比较输出位置配置
	MC_hcmp_get_current_point	读取当前运行的比较点数据
	MC_hcmp_get_points_runned	读取已经运行的比较点个数

	MC_hcmp_get_points_remained	查询可以加入的比较点数量
高速二维位置比较功能	MC_hcmp2d_config	配置高速二维比较输出
	MC_hcmp2d_get_config	读取高速二维比较输出配置
	MC_hcmp2d_clear_points	清除高速二维比较输出数据
	MC_hcmp2d_add_point	添加高速二维比较点
	MC_hcmp2d_get_current_point	读取当前运行的比较点位置
	MC_hcmp2d_get_points_runned	读取已经运行的比较点个数
	MC_hcmp2d_get_points_remained	查询可以加入的比较点数量
点位运动控制函数	MC_set_profile_ex	设置指定轴速度曲线运动模式的起始速度、最大速度、加速时间、减速时间
	MC_get_profile_ex	读取速度曲线设置
	MC_set_s_profile	设置 S 形速度曲线
	MC_get_s_profile	读取 S 形速度曲线的设置
	MC_pmove	单轴位置控制（点位运动）
	MC_pmove_profile	设置轴速度并启动点位运动
	MC_pmove_flex	单轴位置控制（点位运动）（软着陆）
	MC_pmove_flex_ex	单轴位置控制（点位运动）（前段缓拉）
	MC_reset_target_position	在运动中改变目标位置
连续运动及运动速度控制函数	MC_vmove	使指定轴做连续运动
	MC_check_done_	读取指定轴的运动状态
	MC_config_softlimit	设置软件限位
	MC_get_config_softlimit	读取软限位的设置
	MC_change_pmove_speed	在线改变指定轴的点位运动速度
	MC_change_speed	改变指定轴的运行速度
	MC_read_current_speed	读取指定轴的当前速度
	MC_get_aimposition	读取指定轴目标位置
	MC_decel_stop	单轴减速停止
	MC_imd_stop	单轴立即停止
	MC_Stop	单轴停止
	MC_emg_stop	紧急停止所有轴
	MC_check_done_all	读取所有轴的运动状态
	MC_check_done_reason	读取指定轴的运动状态并返回停止原因
	MC_set_vector_s_profile_muticoor	设置插补运动的 S 形加速比例
	MC_conti_set_mode_muticoor	设置连续插补参数（带插补器参数）

	MC_conti_get_mode_muticoor	读取连续插补参数（带插补器参数）
	MC_conti_lookahead_init_muticoor	速度前瞻初始化（带插补器参数）
	MC_conti_io_muticoor	插补缓冲区 IO（带插补器参数）
	MC_conti_io_action_remained	查询插补 IO 剩余缓冲大小
	MC_conti_clear_io_action	清除插补 IO 缓冲
	MC_conti_delay_outbit_to_start	连续插补相对于轨迹段起点 IO 滞后输出
	MC_conti_delay_outbit_to_stop	连续插补相对于轨迹段终点 IO 滞后输出
	MC_conti_ahead_outbit_to_stop	连续插补相对于轨迹段终点 IO 提前输出
	MC_conti_pwm_muticoor	插补缓冲区中添加 PWM 输出（带插补器参数）
	MC_conti_canio_da_muticoor	在插补缓冲中添加 DA 输出（带插补器参数）
	MC_conti_wait_input_muticoor	插补缓冲区中等待输入信号再向下运行
	MC_conti_delay_muticoor	插补缓冲区延时（带插补器参数）
	MC_conti_gear_muticoor	刀向跟随功能（带插补器参数）
	MC_conti_check_done_muticoor	检测插补状态（带插补器参数）
	MC_conti_state_muticoor	检查插补当前状态（带插补器参数）
	MC_conti_open_list_muticoor	打开插补缓冲区（带插补器参数）
	MC_conti_close_list_muticoor	关闭插补缓冲区（带插补器参数）
	MC_conti_check_remain_space_muticoor	读取连续插补剩余缓冲区段数（带插补器参数）
	MC_conti_pushdata_muticoor	将前瞻缓冲区数据压入控制器（带插补器参数）
	MC_conti_decel_stop_list_muticoor	插补减速停止（带插补器参数）
	MC_conti_sudden_stop_list_muticoor	插补立即停止（带插补器参数）
	MC_conti_pause_list_muticoor	插补暂停（带插补器参数）
	MC_conti_set_pause_output_muticoor	设置暂停输出 IO
	MC_conti_start_list_muticoor	启动插补运动（带插补器参数）
	MC_conti_read_current_mark_muticoor	读取当前运动段 mark 标志（带插补器参数）
	MC_conti_set_corner_arc_radius_muticoor	配置插补运动拐角圆弧过渡
	MC_conti_pmove_muticoor	直线插补点位运动功能
	MC_conti_extern_lines_muticoor	连续直线插补（imark）（带插补器参数）

	MC_conti_extern_arc_muticoor	连续圆弧插补（imark）（带插补器参数）
	MC_conti_extern_arc_3p_muticoor	连续圆弧插补（imark）（三点定圆）（带插补器参数）
	MC_conti_line_glue	连续两轴直线插补分段均匀打点
	MC_read_vector_speed_muticoor	读取当前卡的插补速度
	MC_line2_muticoor	任意 2 轴直线插补（带插补器参数）
	MC_line3_muticoor	任意 3 轴直线插补（带插补器参数）
	MC_lineN_muticoor	指定多轴直线插补（带插补器参数）
	MC_arc_move_muticoor	指定 2 轴圆弧插补（绝对位置）
	MC_rel_arc_move_muticoor	指定 2 轴圆弧插补（相对位置）
	MC_config_ALM_PIN	设置 ALM 信号
	MC_get_config_ALM_PIN	读取 ALM 信号的设置
	MC_config_EZ_PIN	设置 EZ 信号
	MC_get_config_EZ_PIN	读取 EZ 信号的设置
	MC_config_LTC_PIN	设置 LTC 信号
	MC_get_config_LTC_PIN	读取 LTC 信号的设置
	MC_config_EL_PIN	设置 EL 信号
	MC_get_config_EL_PIN	读取 EL 信号的设置
	MC_config_HOME_PIN_logic	设置 HOME 信号
	MC_get_config_HOME_PIN_logic	读取 HOME 信号的设置
	MC_write_SEVON_PIN	输出 SEVON 信号
	MC_read_SEVON_PIN	读取 SEVON 信号
	MC_set_backlash	设置间隙补偿值
	MC_get_backlash	读取间隙补偿值
	MC_read_RDY_PIN	读取 RDY 状态
	MC_write_ERC_PIN	控制 ERC 信号输出
	MC_config_EMG_PIN	设置 EMG 信号
	MC_get_config_EMG_PIN	读取 EMG 信号的设置
	MC_axis_io_status	读取指定轴有关运动信号的状态
	MC_axis_io_status_all	读取卡上所有轴有关运动信号的状态
手轮运动控制函数	MC_set_handwheel_inmode	设置手轮脉冲输入方式
	MC_get_handwheel_inmode	读取手轮脉冲输入方式
	MC_handwheel_move	启动手轮运动
凸轮运动控制	MC_ecam_disengage	取消凸轮参考轴和从轴链接

函数	MC_ecam_engage	启动凸轮参考轴和从站链接
	MC_ecam_fifo_datain	凸轮数据压入
	MC_ecam_fifo_clear	凸轮数据清除
	MC_ecam_fifo_remian	查询凸轮数据剩余段数
	MC_ecam_get_curmark	当前正在运行的凸轮数据段标号
	MC_ecam_state	读取当前凸轮运动的状态
	MC_ecam_mode	读取当前凸轮运动的跟随模式
	MC_ecam_set_repeatlength	设置凸轮运动模
	MC_ecam_get_repeatlength	读取凸轮运动模
	MC_ecam_fifo_ioaction	在凸轮点后增加一个凸轮顶杆事件
通用 I/O 接口函数	MC_read_inbit	读取输入口的状态
	MC_write_outbit	设置输出口的状态
	MC_read_outbit	读取输出口的状态
	MC_read_inport	读取输入端口的值
	MC_read_outport	读取输出端口的值
	MC_write_outport	设置输出端口的值
	MC_write_outport_ex	设置输出端口的值
	MC_write_outbit_mask	设置多个输出口的状态
	MC_reverse_outbit	输出口延时并电平取反
输出 PWM 函数	MC_write_pwm	立即输出 PWM
	MC_write_pwmf	立即输出 PWM
CAN IO 接口函数	MC_init_canio	初始化 CAN IO 扩展模块
	MC_canio_linkstate	读取 CAN IO 扩展模块的连接状态
	MC_canio_readinbit	读取 CAN IO 输入口的状态
	MC_canio_readinport	读取 CAN IO 输入端口的值
	MC_canio_readoutbit	读取 CAN IO 指定输出口状态
	MC_canio_readoutport	读取 CAN IO 输出口的状态
	MC_canio_writeoutbit	设置 CAN IO 输出口的状态
	MC_canio_writeoutport	设置 CAN IO 输出端口的值
	MC_init_can_analog	初始化 CAN 模拟量扩展模块
	MC_canio_read_ain	读取 CAN 模块的指定通道号 AD 值
	MC_canio_read_aout	读取 CAN 模块的指定通道号 DA 值
	MC_canio_set_aout	设置 CAN 模块的指定通道号 DA 值

输入口高速计数函数	MC_set_h_count_value	设置输入口高速计数值
	MC_get_h_count_value	读取输入口高速计数值
IO口映射功能函数	MC_set_axis_io_map	设置 IO 映射配置
	MC_get_axis_io_map	读取 IO 映射配置
参数文件写入函数	MC_LoadConfig	把参数文件（.ini 文件）里面保存的参数写入控制器
控制器设置密码函数	MC_read_user_password	读取控制器密码
输入信号滤波时间函数	MC_config_special_input_filter	设置特殊输入信号滤波时间
	MC_get_config_special_input_filter	读取特殊输入信号滤波时间
	MC_config_input_filter	设置通用输入信号滤波时间
	MC_get_config_input_filter	读取通用输入信号滤波时间
	MC_config_home_pin_filter	设置原点信号滤波时间
	MC_get_config_home_pin_filter	读取原点信号滤波时间

3.1.2 函数说明

下面我们对这些函数分类进行详细说明。

3.1.2.1 初始化函数

int32 MC_OpenEth(char *ipaddr, MCHANDLE * phandle)

功　能：使用网络链接控制器

参　数：ipaddr　要链接的控制器 IP 字符串，控制器默认 IP 地址为 192.168.1.221

　　phandle 返回的链接句柄，后续函数使用

返回值：0-控制器链接成功，非 0 表示链接失败

int32 MC_Close(MCHANDLE　handle)

功　能：关闭控制器，释放系统资源

参　数：phandle　　链接句柄

返回值：错误码

int32 MC_OpenCom(uint comid,uint BAUD, MCHANDLE * phandle)

功　能：使用串口链接控制器

参　数：comid　要链接的控制器串口号，如为 COM1，则为:1

　　BAUD　串口波特率:如 9600，115200 等

phandle 返回的链接句柄，后续函数使用

返回值：0-控制器链接成功，非 0 表示链接失败

uint32 MC_get_total_axes(MCHANDLE phandle)

功能：读取指定控制器总轴数

参 数：phandle 链接句柄

返回值：控制器总轴数

uint32 MC_get_lib_version(void)

功 能：读取软件版本号。

参 数：无

返回值：控制器函数库的版本号

uint32 MC_get_card_version(MCHANDLE phandle)

功 能：读取控制器的硬件版本号

参 数：phandle 链接句柄

返回值：硬件版本号

uint32 MC_get_card_soft_version(MCHANDLE phandle, WORD* firm_id, uint32* sub_firm_id)

功 能：读取卡硬件的固件版本号

参 数：链接句柄

Firm_id: 固件版本号 id

sub_firm_id: 固件版本号子 id

返回值：错误码

uint32 MC_get_controller_sn(MCHANDLE phandle, uint32* sn)

功 能：读取控制器唯一序列号

参 数： phandle 链接句柄（（拨码开关说明）

sn 控制器序列号（32 位）

返回值：错误码

3.1.2.2 脉冲模式设置函数

uint32 MC_set_pulse_outmode(MCHANDLE phandle, WORD axis, WORD outmode)

uint32 MC_get_pulse_outmode(MCHANDLE phandle, WORD axis, WORD *outmode)

功　能：设置/读取指定轴的脉冲输出模式

参　数：phandle　　链接句柄

axis　　指定轴号（轴号范围说明）

outmode　　脉冲输出方式选择，其值如下表所示：

输出脉冲类型 outmode	正方向脉冲		负方向脉冲	
	OUT 输出端	DIR 输出端	OUT 输出端	DIR 输出端
0		高电平		低电平
1		高电平		低电平
2		低电平		高电平
3		低电平		高电平
4		高电平	高电平	
5		低电平	低电平	

返回值：错误码

注意：在调用运动函数（如：MC_vmove 等）输出脉冲之前，一定要根据驱动器接收脉冲的模式调用 MC_set_pulse_outmode 设置控制器脉冲输出模式。

3.1.2.3 回原点运动函数

3.1.2.3.1 回原点运动函数

uint32 MC_config_home_mode(MCHANDLE phandle, WORD axis, WORD home_dir, double vel, WORD mode, WORD EZ_count)

uint32 MC_get_config_home_mode(MCHANDLE phandle, WORD axis, WORD* home_dir, double* vel, WORD* home_mode, WORD* EZ_count)

功　能：设定/读取指定轴的回原点模式

参　数：phandle　　链接句柄

axis　　指定轴号（轴号范围说明）

home_dir　　回零方向，　1:正向, 2:负向

vel　　回零速度，0 为低速(该轴起始速度)回原点，1 为高速(该轴运行速度)回原点。

mode　　回原点运动模式（回零模式详细说明）

0 - 一次回零（碰到原点减速停止）
1 - 二次回零
2 - 一次回零加回找
3 - 一次回零后加同向 EZ 回零
4 - 以 EZ 作为原点进行一次回零
6 - 一次回零（碰到原点减速停止）
12 - 二次回零（如果当前原点信号已经有效，即在传感器的感应区域内时，高速反向退出原点，再进行二次回零；反之则回零过程和回零模式 1（二次回零）一样）
16 - 原点信号锁存回零（原点信号设置为锁存信号，当原点信号被触发时，控制器会锁存当前轴的编码器位置并让当前轴减速停止，电机停止后再移动当前轴到锁存位置，将锁存位置作为原点）
18 - EZ 信号锁存回零（EZ 信号设置为锁存信号，EZ 点信号被触发时，控制器会锁存当前轴的编码器位置并让当前轴减速停止，电机停止后再移动当前轴到锁存位置，将锁存位置作为原点）

20 - 一次限位回零
21 - 一次限位回零加反找
22 - 二次限位回零
40 - 一次回零后加反向 EZ 回零
41 - 一次限位后加反向 EZ 回零

EZ_count 保留参数

返回值：错误码

注意：1）在调用 **MC_config_home_mode** 函数前需要先调用 **MC_set_profile_ex** 函数设置运动参数，因为回零运动的速度参考 **MC_set_profile_ex** 函数设置的最大运行速度；否则回零运动的速度参考的是上一次调用 **MC_set_profile_ex** 函数设置最大运行速度或者卡默认运动速度，可能导致回零运动速度不是用户想要的速度。

2）第一次回零的速度参考参数 **vel**，如果当前设置 **mode** 参数为 1、2、3 时回零运动有反找过程，那么反找的回零速度为当前的轴的起始速度。

3）在使用回零模式 3、4、16 和 18 时，必须保证当前轴的指令位置和编码器位置在回零运动开始前是一致的；使用回零模式 16 时控制器会自动把锁存模式设置为原点锁存，使用回零模式 18 时控制器会自动把锁存模式设置为 EZ 锁存；如需切换锁存模式可以在回零运动结束后自行更改锁存模式，具体可参考 **MC_config_LTC_PIN** 函数说明。

4）回零完成后需要手动清除指令位置和编码器位置，在清除编码器位置之前请确保电机已经停稳再进行操作。

uint32 MC_home_move(MCHANDLE phandle, WORD axis)

功 能：单轴回原点运动

参 数：phandle 链接句柄

axis 指定轴号（轴号范围说明）

返回值：错误码

注意：回零运动完成后指令位置和编码器位置不会自动清零，需要调用 **MC_set_position** 清零指令位置和调用 **MC_set_encoder** 清零编码器位置。

3.1.2.4 位置计数锁存比较

3.1.2.4.1 位置计数器控制函数

long MC_get_position(MCHANDLE phandle, WORD axis)

功 能：读取指定轴的指令脉冲位置

参 数：phandle 链接句柄

axis 指定轴号（轴号范围说明）

返回值：指定运动轴的命令脉冲数，单位：脉冲

uint32 MC_set_position(MCHANDLE phandle, WORD axis, long current_position)

功 能：设置指定轴的指令脉冲位置

参 数：phandle 链接句柄

axis 指定轴号（轴号范围说明）

current_position 绝对位置值

返回值：错误码

注意：在总线模式下需要进入 CSP 模式后 **MC_set_position** 才能生效（HY7800E）

long MC_get_position_all(MCHANDLE phandle, long *position)

功 能：读取所有轴的指令脉冲位置

参 数：phandle 链接句柄

position 绝对位置值（以数组方式读取，例如数组的第 0 号元素对应第 0 轴的指令位置）

返回值：错误码

long MC_get_aimposition(MCHANDLE phandle, WORD iaxis)

功 能：读取指定轴目标位置

参 数：phandle 链接句柄

axis 指定轴号（轴号范围说明）

返回值：轴目标位置

long MC_get_counter_flag(MCHANDLE phandle)

功 能：读取指定控制器的计数器的标识位

参 数：phandle 链接句柄

返回值：见表

返回值位号	描述
0	指定卡的 0 轴借位触发标志：计数器每下溢出时触发一次
1	进位触发标志：计数器每上溢出时触发一次
2	符号标志：计数器上溢出为 0，下溢出为 1
3	上下计数标志：上计数时为 1，下计数时为 0
4～7	指定卡的第 1 轴，同上
8～11	指定卡的第 2 轴，同上
12～15	指定卡的第 3 轴，同上
16～19	指定卡的第 4 轴，同上
20～23	指定卡的第 5 轴，同上
24～27	指定卡的第 6 轴，同上
28～31	指定卡的第 7 轴，同上

uint32 MC_reset_counter_flag(MCHANDLE phandle)

功 能：复位计数器的计数标志位

参 数：phandle 链接句柄

返回值：错误码

uint32 MC_reset_clear_flag(MCHANDLE phandle)

功 能：复位计数器的清零标志位，范围（0—N - 1,N 为卡数）

参 数：phandle 链接句柄

返回值：错误码

3.1.2.4.2 编码器相关函数

uint32 MC_counter_config(MCHANDLE phandle, WORD axis, WORD mode)

uint32 MC_get_counter_config(MCHANDLE phandle, WORD axis, WORD* mode)

功 能：设置/读取编码器的计数方式

参　数：phandle　　链接句柄
　　axis　　指定轴号（轴号范围说明）
　　mode　　编码器的计数方式：
　　0 -- 非 A/B 相脉冲信号（脉冲+方向信号）
　　1 -- 1 倍 A/B 相脉冲信号
　　2 -- 2 倍 A/B 相脉冲信号
　　3 -- 4 倍 A/B 相脉冲信号
返回值：错误码

long MC_get_encoder(MCHANDLE phandle, WORD axis)
功　能：读取指定轴编码器反馈位置脉冲计数值，范围：28 位有符号数
参　数：phandle　　链接句柄
　　axis　　指定轴号（轴号范围说明）
返回值：位置反馈脉冲值，单位：脉冲数

uint32 MC_set_encoder(MCHANDLE phandle, WORD axis, long encoder_value)
功　能：设置指定轴编码器反馈脉冲计数值，范围：28 位有符号数
参　数：phandle　　链接句柄
　　axis　　指定轴号（轴号范围说明）
　　encoder_value　编码器的设定值。
返回值：错误码

long MC_get_encoder_all(MCHANDLE phandle, long* enposall)
功　能：读取指定轴编码器反馈位置脉冲计数值，范围：28 位有符号数
参　数：phandle　　链接句柄
　　enposall　　位置反馈脉冲值（以数组方式读取，例如数组的第 0 号元素对应第 0 轴的位置反馈脉冲值）
返回值：错误码
注意：enposall 创建数组的大小应大于等于当前控制器可控制的最大轴数，可通过 MC_get_total_axes 读取。例如 HY7800 卡最大轴数则为 8。

uint32 MC_config_encoder_dir(MCHANDLE phandle, WORD axis, WORD ifreverse)
功　能：设置/读取指定轴编码器反馈脉冲计数值方向
参　数：phandle　　链接句柄

axis 指定轴号（轴号范围说明）
ifreverse 编码器的计数方向：0－不取反，1－取反
返回值：错误码

3.1.2.4.3 位置锁存相关函数

注意：锁存信号接口为每一个轴的 EZ 信号或 ORG 信号，EZ 信号为 5V 信号，ORG 信号为 24V 信号。如果是使用 ORG 信号锁存，可直接把光纤的 24V 输入信号接入 ORG 信号。配置锁存方式通过 MC_config_LTC_PIN 函数进行设置。

uint32 MC_config_latch_mode(MCHANDLE phandle, WORD all_enable)
uint32 MC_get_config_latch_mode(MCHANDLE phandle, WORD* all_enable)
功 能：设置/获取锁存方式为单轴锁存或是八轴同时锁存
参 数：phandle 链接句柄
all_enable 锁存方式 ：0－单独锁存， 1－八轴同时锁存
返回值：错误码

long MC_get_latch_value(MCHANDLE phandle, WORD axis)
功 能：读取编码器锁存器的值
参 数：phandle 链接句柄
axis 指定轴号（轴号范围说明）
返回值：锁存器内的编码器脉冲数，单位：脉冲

long MC_get_latch_flag(MCHANDLE phandle)
功 能：读取指定控制器的锁存器的标志位
参 数：phandle 链接句柄
返回值：见表

返回值位号	描述
0	1：指定卡的 0 轴有触发信号.
1	1：1 轴有触发信号.
2	1：2 轴有触发信号.
3	1：3 轴有触发信号.
4	1：4 轴有触发信号.
5	1：5 轴有触发信号.
6	1：6 轴有触发信号.
7	1：7 轴有触发信号.

8	1：指定卡的 0 轴有清零信号
9	1：1 轴有清零信号
10	1：2 轴有清零信号
11	1：3 轴有清零信号
12	1：4 轴有清零信号
13	1：5 轴有清零信号
14	1：6 轴有清零信号
15	1：7 轴有清零信号
16	1：0 轴位置已锁存
17	1：1 轴位置已锁存
18	1：2 轴位置已锁存
19	1：3 轴位置已锁存
20	1：4 轴位置已锁存
21	1：5 轴位置已锁存
22	1：6 轴位置已锁存
23	1：7 轴位置已锁存
24	1：指定卡的 0 轴位置已清零
25	1：1 轴位置已清零
26	1：2 轴位置已清零
27	1：3 轴位置已清零
28	1：4 轴位置已清零
29	1：5 轴位置已清零
30	1：6 轴位置已清零
31	1：7 轴位置已清零

注：第 8-15 位分别对应读取 0-7 轴的 EZ 信号状态，第 16-23 位对应读取 0-7 轴的锁存标志（锁存标志：通过读取锁存标志判断位置锁存功能是否已触发，请在读取到锁存标志之后再去读取编码器锁存器的值 **MC_get_latch_value**）。

uint32 MC_reset_latch_flag(MCHANDLE phandle)

功　能：复位指定控制器所有轴的锁存器的标志位

参　数：phandle　　链接句柄

返回值：错误码

uint32 MC_reset_latch_flag_ex(MCHANDLE phandle, WORD iaxis)

功　能：复位指定控制器指定轴的锁存器的标志位

参　数：phandle　　链接句柄

　　axis　　轴号（轴号范围说明）

返回值：错误码

3.1.2.4.4 连续位置锁存相关函数

long MC_conti_latch_config(MCHANDLE phandle, WORD axis, WORD enable, WORD src)

long MC_conti_latch_get_config(MCHANDLE phandle, WORD iaxis, WORD *enable, WORD *src)

功　能：设置/读取连续锁存功能

参　数：phandle　　链接句柄

axis　　指定轴号（轴号范围说明）

enable　　1：使能比较功能，　0：禁止比较功能

src　　锁存源(保留)

返回值：错误码

long MC_conti_latch_clear(MCHANDLE phandle, WORD iaxis)

功　能：清除连续锁存缓存

参　数：phandle　　链接句柄

axis　　指定轴号（轴号范围说明）

返回值：错误码

long MC_conti_latch_count(MCHANDLE phandle, WORD iaxis, WORD *cnt)

功　能：读取连续锁存缓存已经缓存个数

参　数：phandle　　链接句柄

iaxis　　指定轴号（轴号范围说明）

cnt　　已经缓存数据个数

返回值：错误码

long MC_conti_latch_value(MCHANDLE phandle, WORD iaxis, long *data)

功　能：读取连续锁存缓存数据，如果个数大于 0，可以继续读取

参　数：phandle　　链接句柄

iaxis　　指定轴号（轴号范围说明）

data　　锁存数据

返回值：错误码

注意：本函数每调用一次，就回返回一个锁存数据

3.1.2.4.5 一维位置比较功能函数

uint32 MC_Compare_Config(MCHANDLE phandle, WORD enable, WORD axis,WORD cmp_source)

uint32 MC_Compare_GetConfig(MCHANDLE phandle, WORD* enable, WORD* axis, WORD* cmp_source)

功 能：设置/读取比较器配置

参 数：phandle 链接句柄

enable 1：使能比较功能， 0：禁止比较功能

axis 轴号（轴号范围说明）

cmp_source 比较源， 0：比较指令位置， 1：比较编码器位置

返回值：错误码

uint32 MC_Compare_ClearAllPoint(MCHANDLE phandle)

功 能：清除所有比较点

参 数: phandle 链接句柄

返回值：错误码

uint32 MC_Compare_AddPoint(MCHANDLE phandle, long pos, WORD dir, WORD action, long actpara)

功 能：添加比较点

参 数：phandle 链接句柄

Pos： 位置坐标

dir： 比较方向 0: 小于等于 1： 大于等于

action： 比较点触发动作

actpara： 比较点触发对应的输出口序号

返回值：错误码

MC_compare_add_point 函数 action, actpara 参数值含义：

action	actpara	功能
1	IO 号	打开 IO
2	IO 号	关闭 IO
3	IO 号	取反 IO
5	IO 号	输出 100us 脉冲
6	IO 号	输出 1ms 脉冲
7	IO 号	输出 10ms 脉冲

8	IO 号	输出 100ms 脉冲
11	速度值	当前轴变速
13	轴号	停止指定轴
21	高速输出口序号	输出指定宽度的脉冲
31	DA 输出口值	模块 id +16*输出 DA 的通道号 +256*输出电压值。（输出电压值对应关系可参考此处）如：从 CAN 模拟量模块 1 的第 2 通道输出 DA 电压 10V 电压输入下面参数 para = 1+162+256*4096

注意：比较点触发的顺序和比较点的添加顺序相同，和比较点的触发位置大小无关，遵循先进先出原则。

uint32 MC_Compare_GetCurPoint(MCHANDLE phandle)

功　能：读取当前比较点的触发位置

参　数：phandle　　链接句柄

返回值：当前比较点对应的触发位置

uint32 MC_Compare_GetPointsRunned(MCHANDLE phandle)

功　能：查询已经比较过的点数量

参　数：phandle　　链接句柄

返回值：一维比较已经比较过的比较点数量

uint16 MC_Compare_GetPointsSpace(MCHANDLE phandle)

功　能：查询可以加入的比较点数量

参　数：phandle　　链接句柄

返回值：一维比较可以加入的比较点数量

3.1.2.4.6　多组一维位置比较功能函数

uint32 MC_Compare_Set_PulseTime_Muti(MCHANDLE phandle, WORD queue, float ftimes)

功　能：设置功能号 9 的脉冲宽度，单位 ms

参　数：phandle　　链接句柄

queue　　比较器序号（0-23）

ftimes 脉冲宽度 （单位：100 微秒）

返回值：错误码

注 意：如果设置脉冲宽度数值不需要改变，该函数只调用一次即可。

uint32 MC_compare_set_filter_Muti(MCHANDLE phandle, uint32 frequency)

功 能：设置锁存器滤波频率（该函数不开放）

参 数：phandle 链接句柄

frequency 滤波频率，高于该频率的滤除（单位：赫兹）

返回值：错误码

uint32 MC_compare_get_filter_Muti(MCHANDLE phandle, uint32* frequency)

功 能：读取锁存器滤波频率（该函数不开放）

参 数：phandle 链接句柄

*frequency 滤波频率，高于该频率的滤除（单位：赫兹）

返回值：错误码

uint32 MC_Compare_Config_Muti(MCHANDLE phandle, WORD queue, WORD enable, WORD axis, WORD cmp_source)

uint32 MC_Compare_GetConfig_Muti(MCHANDLE phandle, WORD queue, WORD* enable, WORD* axis, WORD* cmp_source)

功 能：设置/读取比较器配置

参 数：phandle 链接句柄

queue 比较器序号（0-23）

enable 1：使能比较功能 0：禁止比较功能

axis 指定轴号（轴号范围说明）

cmp_source 比较源（0：比较指令位置 1：比较编码器位置）

返回值：错误码

uint32 MC_Compare_ClearAllPoint_Muti(MCHANDLE phandle, WORD queue)

功 能：清除所有比较点

参 数：phandle 链接句柄

queue 比较器序号（0-23）

返回值：错误码

uint32 MC_Compare_AddPoint_Muti(MCHANDLE phandle, WORD queue, long pos, WORD dir, WORD action, long actpara)

功　能：添加比较点

参　数：phandle　　链接句柄

queue　　比较器序号（0-23）

pos　　对应轴的触发位置

dir　　比较方向（0：小于等于 1：大于等于）

action　　比较点触发动作

actpara　　比较点触发对应的输出口序号

MC_compare_add_point_ Muti 函数 action, actpara 参数值含义：

action	actpara	功能
1	IO 号	打开 IO
2	IO 号	关闭 IO
3	IO 号	取反 IO
5	IO 号	输出 100us 脉冲
6	IO 号	输出 1ms 脉冲
7	IO 号	输出 10ms 脉冲
8	IO 号	输出 100ms 脉冲
9	IO 号	输出脉冲，脉冲宽度由函数制定
11	速度值	当前轴变速
13	轴号	停止指定轴
31	DA 输出口值	模块 id +16*输出 DA 的通道号 +256*输出电压值。（输出电压值对应关系可参考此处）如：从 CAN 模拟量模块 1 的第 2 通道输出 DA 电压 10V 电压输入下面参数 para = 1+162+256*4096

返回值：错误码

注意：比较点触发的顺序和比较点的添加顺序相同，和比较点的触发位置大小无关，遵循先进先出原则。使用功能号 9 时，需要先调用函数 MC_compare_set_pulsetimes_Muti 设置脉冲宽度，单位 ms。

uint32 MC_Compare_GetCurPoint_Muti(MCHANDLE phandle, WORD queue)

功　能：读取当前比较点位置

参 数：phandle 链接句柄
queue 比较器序号（0-23）
返回值：当前比较点对应的触发位置

uint32 MC_Compare_GetPointsRunned_Muti(MCHANDLE phandle, WORD queue)
功 能：查询已经比较过的比较点数量
参 数：phandle 链接句柄
queue 比较器序号（0-23）
返回值：已经比较过的比较点数量

uint32 MC_Compare_GetPointsSpace_Muti(MCHANDLE phandle, WORD queue)
功 能：查询可添加的比较点数量
参 数：phandle 链接句柄
queue 比较器序号（0-23）
返回值：可添加的比较点数量
注意：每成功添加一个比较点，可添加的比较点数量就会减 1；当比较点成功触发后可添加的比较点数量就会加 1，已经比较的比较点数量会加 1。

3.1.2.4.7 高速一维位置比较功能函数

uint32 MC_hcmp_config(MCHANDLE phandle, WORD hcmp, WORD enable, WORD axis, WORD cmp_source)
功 能：配置高速比较输出。
参 数：phandle 链接句柄
hcmp 高速比较通道号（范围: 0-3）
enable 使能高速比较 0-禁止 1-使能
axis 保留参数，写 0
cmp_source 比较源，保留参数，默认为编码器位置
返回值：错误码

控制器类型	高速比较通道号	对应输出口序号范围	轴号范围
MC400A	0－3	OUT1,OUT2,OUT3,OUT4	0－3
MC800A	0－3	OUT1,OUT2,OUT3,OUT4	0－3

uint32 MC_hcmp_get_config(MCHANDLE phandle, WORD hcmp, WORD* enable, WORD* axis, WORD* cmp_source)
功 能：读取高速比较输出配置。

参 数：phandle 链接句柄
hcmp 高速比较通道号（范围：0-3）
enable 使能高速比较 0-禁止 1-使能
axis 保留参数
cmp_source 比较源，保留参数，默认为编码器位置
返回值：错误码

uint32 MC_hcmp_clear_points(MCHANDLE phandle, WORD hcmp)
功 能：清除高速比较输出数据
参 数：phandle 链接句柄
hcmp 高速比较通道号（范围：0-3）
返回值：错误码

uint32 MC_hcmp_add_point(MCHANDLE phandle, WORD hcmp, long pos, WORD dir, WORD action, long actpara)
功 能：添加高速位置比较点
参 数：phandle 链接句柄
hcmp 高速比较通道号（范围：0-3）
pos 高速比较输出位置，只支持编码器位置
dir 保留参数
action 保留参数
actpara 比较输出的脉冲宽度（单位：微秒）
返回值：错误码
注意：比较点触发的顺序和比较点的添加顺序相同，和比较点的触发位置大小无关，遵循先进先出原则。

控制器类型	高速比较通道号	对应输出口序号范围	轴号范围
MC400A	0－3	OUT1,OUT2,OUT3,OUT4	0－3
MC800A	0－3	OUT1,OUT2,OUT3,OUT4	0－3

long MC_hcmp_get_current_point(MCHANDLE phandle, WORD hcmp)
功 能：读取当前运行的比较点数据
参 数：phandle 链接句柄
hcmp 高速比较通道号（范围：0-3）
返回值：当前正在运行的比较点数据

long MC_hcmp_get_points_runned(MCHANDLE phandle, WORD hcmp)
功 能：读取已经运行的比较点个数
参 数：phandle 链接句柄

hcmp 高速比较通道号（范围：0-3）

返回值：已经运行的比较点个数

long MC_hcmp_get_points_remained(MCHANDLE phandle, WORD hcmp)

功 能：查询可以加入的比较点数量

参 数：phandle 链接句柄

hcmp 高速比较通道号（范围：0-3）

返回值：剩余比较点空间（范围：0-256）

3.1.2.4.8 高速二维位置比较功能函数

uint32 MC_hcmp2d_config(MCHANDLE phandle, WORD queue, WORD enable, WORD axis0, WORD axis1, WORD gap, WORD cmp_source)

uint32 MC_hcmp2d_get_config(MCHANDLE phandle, WORD hcmp, WORD* enable, WORD* axis, WORD *gap, WORD* cmp_source)

功 能：设置/读取比较器配置

参 数：phandle 链接句柄

queue 高速比较器序号(0-1)

enable 1：使能比较功能， 0：禁止比较功能

axis0 比较的第 0 轴的轴号

axis1 比较的第 1 轴的轴号

gap 比较误差范围(1-255)

cmp_source 比较源， 0：比较指令位置， 1：比较编码器位置

返回值：错误码

uint32 MC_hcmp2d_clear_points(MCHANDLE phandle, WORD hcmp)

功 能：清除高速比较输出数据

参 数：phandle 链接句柄

hcmp 高速比较器序号(0-1)

返回值：错误码

uint32 MC_hcmp2d_add_point(MCHANDLE phandle, WORD queue, long pos0,long pos1, WORD dir0, WORD dir1, WORD action, long actpara)

功 能：添加高速位置比较点

参 数：phandle 链接句柄

queue 高速比较器序号(0-1)

pos0 比较 0 轴的输出位置
pos1 比较 1 轴的输出位置
dir0 保存参数
dir1 保留参数
action 按位设置需要输出的比较通道号，如设置为 0x5(二进制 101)，表示该点的比较结果输出至 12,14 口
actpara 比较输出的脉冲宽度（单位：微秒）

返回值：错误码

uint32 MC_hcmp2d_get_current_point(MCHANDLE phandle, WORD hcmp, long *curpos)

功 能：读取当前运行的比较点位置
参 数：phandle 链接句柄
hcmp 高速比较器序号(0-1)
curpos 当前比较点位置列表
返回值：错误码

long MC_hcmp2d_get_points_runned(MCHANDLE phandle, WORD hcmp)

功 能：读取已经运行的比较点个数
参 数：phandle 链接句柄
hcmp 高速比较器序号(0-1)
返回值：已经运行的比较点个数

long MC_hcmp2d_get_points_remained(MCHANDLE phandle, WORD hcmp)

功 能：查询可以加入的比较点数量
参 数：phandle 链接句柄
hcmp 高速比较器序号(0-1)
返回值：剩余比较点空间（范围：0-256）

3.1.2.5 单轴位置运动控制函数

3.1.2.5.1 点位运动函数

uint32 MC_set_profile_ex(MCHANDLE phandle, WORD axis, double Min_Vel,double Max_Vel, double stopspeed, double Tacc, double Tdec)

uint32 MC_get_profile_ex(MCHANDLE phandle, WORD axis, double* Min

_Vel, double* Max_Vel, double* stopspeed, double* Tacc, double* Tdec)

功　能：设置/读取指定轴速度曲线运动模式的起始速度、最大速度、停止速度、加速时间、减速时间

参　数：phandle　　链接句柄
axis　　指定轴号（轴号范围说明）
Min_Vel　　起始速度，单位：脉冲/秒
Max_Vel　　最大速度，单位：脉冲/秒
stopspeed　　停止速度，单位：脉冲/秒
Tacc　　加速时间，单位：秒
Tdec　　减速时间，单位：秒

返回值：错误码

uint32 MC_set_s_profile(MCHANDLE phandle, WORD axis, double s_para)
uint32 MC_get_s_profile(MCHANDLE phandle, WORD axis, double* s_para)

功　能：设置/读取速度曲线运动模式的参数。

参　数：phandle　　链接句柄
axis　　指定轴号（轴号范围说明）
s_para　　S 形加速段占总加速段的比例：当参数设置为 0 时，为梯形速度曲线运动模式，当参数为 0～1.0 时，为 S 形速度曲线运动模式。

返回值：错误码

uint32 MC_pmove(MCHANDLE phandle, WORD axis, long Dist, WORD posi_mode)

功　能：使指定轴做点位运动。

参　数：phandle　　链接句柄
axis　　指定轴号（轴号范围说明）
Dist　　（绝对/相对）位移值，单位：脉冲数
posi_mode　　位移模式设定：0 表示相对位移，1 表示绝对位移

返回值：错误码

注意：在相对位移模式下：当脉冲数设置为正数时，点位运动往正方向走；当脉冲数为负数时，点位运动往负方向走。在绝对位移模式下：如果指定轴当前指令位置大于目标位置（position），点位运动往负方向走；如果指定轴当前指位置小于目标位置（position），点位运动往正方向走；如果指定轴当前指令位等于目标位置（position），则点位运动不运行，停留在原地。

uint32 MC_pmove_profile(MCHANDLE phandle, WORD iaxis, long Dist, WORD posi_mode, double vss, double vms, double ves, double accs, double decs)

功　能：设置轴速度并启动点位运动。

参　数：phandle　　链接句柄
　　iaxis　　指定轴号（轴号范围说明）
　　Dist　　（绝对/相对）位移值，单位：脉冲数
　　posi_mode　　位移模式设定：0 表示相对位移，1 表示绝对位移
　　vss　　起始速度，单位：脉冲/秒
　　vms　　最大速度，单位：脉冲/秒
　　ves　　结束速度，单位：脉冲/秒
　　accs　　加速时间，单位：秒
　　decs　　减速时间，单位：秒

返回值：错误码

uint32 MC_pmove_flex(MCHANDLE phandle, WORD axis, long Dist, WORD posi_mode, long distflex, long speedlow)

功　能：使指定轴做点位运动。（软着陆）

参　数：phandle　　链接句柄
　　axis　　指定轴号（轴号范围说明）
　　Dist　　（绝对/相对）位移值，单位：脉冲数
　　posi_mode　　位移模式设定：0 表示相对位移，1 表示绝对位移
　　distlex　　低速长度（单位：脉冲）
　　speedlow　　低速速度（单位：脉冲/秒）

返回值：错误码

uint32 MC_pmove_flex_ex(MCHANDLE phandle, WORD axis, long Dist, WORD posi_mode, long distflex, long speedlow)

功　能：使指定轴做点位运动。（前段缓拉）

参　数：phandle　　链接句柄
　　axis　　指定轴号（轴号范围说明）
　　Dist　　（绝对/相对）位移值，单位：脉冲数
　　posi_mode　　位移模式设定：0 表示相对位移，1 表示绝对位移
　　distlex　　低速长度（单位：脉冲）

speedlow 低速速度（单位：脉冲/秒）

返回值：错误码

uint32 MC_change_pmove_speed(MCHANDLE phandle, WORD axis, double Curr_Vel)

功 能：在线改变指定轴的当前运动速度。该函数只适用于单轴点位运动中的变速。

参 数：phandle 链接句柄

axis 要设置的轴号（轴号范围说明）

Curr_Vel 新的运行速度（单位：脉冲/秒），不能设为负数。

返回值：错误码

注意：使用该函数只能改变点位运动中当前运行速度的大小，不能改变速度的方向。

uint32 MC_reset_target_position(MCHANDLE phandle, WORD axis, long dist)

功 能：在单轴点位运动中改变目标位置。

参 数：phandle 链接句柄

axis 指定轴号（轴号范围说明）

dist 绝对位置值

返回值：错误码

注意：该函数以绝对位置模式改变单轴点位运动的目标位置。如果调用该函数时，轴在运动过程中，则进行在线变位置；如果该轴在停止过程中，则执行绝对坐标的点位运动。

3.1.2.5.2 连续运动及速度控制函数

uint32 MC_vmove(MCHANDLE phandle, WORD axis, WORD dir, double vel)

功 能：使指定轴做连续运动

参 数：phandle 链接句柄

axis 指定轴号（轴号范围说明）

dir 指定运动的方向，其中 0 表示负方向，1 表示正方向

vel vmove 的速度，单位：脉冲/秒

返回值：错误码。对该运动可能会返回 22 错误码。当正向运动返回错误码 22 时，表示正向软限位触发；当反向运动返回 22 错误码，表示负向软限位触发。

WORD MC_check_done(MCHANDLE phandle, WORD axis)

功 能：检测指定轴的运动状态，停止或是在运行中。

参　数：phandle　　链接句柄
　　axis　　指定轴号（轴号范围说明）
返回值：0 表示指定轴正在运行，1 表示指定轴已停止。
注意：**MC_check_done** 函数仅检测控制器是否发送脉冲停止，不保证脉冲一定按预定设置全部脉冲发送完成（如：上位机立即停止，触发 EL、ALM、EMG 信号等情况轴异常停止）。轴的运动状态只表示控制器规划当前轴的运动状态，由于电机跟随滞后、机械系统震荡等一些原因，一般在规划静止一段时间以后，电机才能完全停止。

uint32 MC_config_softlimit(MCHANDLE phandle, WORD axis, WORD ON_OFF, WORD source_sel, WORD SL_action, long N_limit, long P_limit)
uint32 MC_get_config_softlimit(MCHANDLE phandle, WORD axis, WORD * ON_OFF, WORD* source_sel, WORD* SL_action, long* N_limit, long* P_limit)
功　能：设置/读取软件限位的使能, 限位数值, 响应动作
参　数：phandle　　链接句柄
　　axis　　指定轴号（轴号范围说明）
　　ON_OFF　　软限位使能，　0–禁止；　1–使能
　　source_sel　　比较源选择，　保留，　设置为指令脉冲。
　　SL_action　　限位动作，　0–减速停止，　1–立即停止
　　N_limit　　负限位值
　　P_limit　　正限位值
返回值：错误码
注意：回零过程中软件限位不起作用

uint32 MC_change_speed(MCHANDLE phandle, WORD iaxis, double Curr_Vel)
功　能：单轴变速函数
参　数：phandle　　链接句柄
　　iaxis　　指定轴号（轴号范围说明）
　　Curr_Vel　　新的运行速度（单位：脉冲/秒），不能设为负数。
返回值：错误码
注意：如果当前为点位运动，则执行点位在线变速，如果当前为速度运动，则执行速度运动在线变速。

double MC_read_current_speed(MCHANDLE phandle, WORD axis)
功　能：读取指定轴的当前速度值（单位：脉冲/秒）

参　数：phandle　　链接句柄
　　axis　　指定轴号（轴号范围说明）
返回值：指定轴的速度脉冲数

uint32 MC_decel_stop(MCHANDLE phandle, WORD axis, double dec)
功　能：指定轴减速停止，调用此函数时立即减速，直到停止脉冲输出
参　数：phandle　　链接句柄
　　axis　　指定轴号（轴号范围说明）
　　dec　　减速度，保留，暂不起作用。
返回值：错误码
注意：**减速停止的减速度等于 MC_set_profile_ex 参数中的最大运行速度（Max_Vel）减去结束速度（stopspeed）的差除以减速时间（Tdec）**

uint32 MC_imd_stop(MCHANDLE phandle, WORD axis)
功　能：使指定轴立即停止，没有任何减速的过程
参　数：phandle　　链接句柄
　　axis　　指定轴号（轴号范围说明）
返回值：错误码

uint32 MC_emg_stop(MCHANDLE handle)
功　能：使所有的运动轴紧急停止。
参　数：phandleid　　链接句柄
返回值：错误码

int MC_check_done_all(MCHANDLE phandle)
功　能：检测所有轴的运动状态，停止或是在运行中。
参　数：phandle　　链接句柄
返回值：0 表示还有至少一个轴正在运行中，1 表示所有轴已停止。

WORD MC_check_done_reason(MCHANDLE phandle, WORD iaxis, WORD *stopreason)
功　能：检测指定轴的运动状态并返回停止原因
参　数：phandle　链接句柄
　　iaxis　指定轴号（轴号范围说明）
　　stopreason　停止原因　0—运动结束后正常停止
　　1—EL（硬件限位信号）产生停止
　　2—EMG（急停信号）产生停止
　　4—ALM（报警信号）产生停止
　　8—SOFTLIMITION（软限位）产生停止
　　16—中途强制停止，上层软件调用
　　32—途要求暂停，目前只有插补器使用

64—回零时内部使用，检测到 HOME 时
128—手轮时一段时间没有摇动，则停止
256—未知原因

返回值：0 表示指定轴正在运行，1 表示指定轴已停止。

3.1.2.6 插补运动函数

3.1.2.6.1 单段直线插补运动函数

注：1.不带插补器序号参数的函数默认使用 0 号插补器进行插补运动。
2.以下插补运动函数为单段插补运动函数，和连续插补运动无关。

uint32 MC_set_vector_profile_muticoor(MCHANDLE phandle, WORD CrdNum, double s_para, double Max_Vel, double Tacc, double Tdec)

uint32 MC_get_vector_profile_muticoor(MCHANDLE phandle, WORD CrdNum, double* s_para, double* Max_Vel, double* Tacc, double* Tdec)

功 能：设置/读取插补矢量运动的起始速度、运行速度、加速时间、减速时间

参 数：phandle: 链接句柄
CrdNum 插补器序号（参考列表）
s_para 起始速度（单位：脉冲/秒）
Max_Vel 运行速度（单位：脉冲/秒）
Tacc 加速时间（单位：秒）
Tdec 减速时间（单位：秒）

返回值：错误码

uint32 MC_set_vector_s_profile_muticoor(MCHANDLE phandle, WORD CrdNum, double s_para)

功 能：设置/读取插补矢量运动的 S 形加速段占总加速段的比例

参 数：phandle: 链接句柄
CrdNum 插补器序号（参考列表）
s_para S 形加速段占总加速段的比例：当参数设置为 0 时，为梯形速度曲线运动模式，当参数为 0～1.0 时，为 S 形速度曲线运动模式。

返回值：错误码

double MC_read_vector_speed_muticoor(MCHANDLE phandle, WORD CrdNum)

功 能：读取当前卡的插补速度（单位：脉冲/秒）

参 数：phandle 链接句柄
CrdNum 插补器序号（参考列表）

返回值：插补速度

uint32 MC_line2_muticoor(MCHANDLE phandle, WORD coor, WORD axis 1, long Dist1, WORD axis2, long Dist2, long synendspeed, long synspeed, d ouble synacc, WORD posi_mode)

功　能：指定任意两轴做单段直线插补运动

参　数：phandle　　链接句柄
coor　　插补器序号（参考列表）
axis1　　指定两轴插补的第一轴（轴号范围说明）
Dist1　　指定 axis1 的位移值，单位：脉冲数
axis2　　指定两轴插补的第二轴（轴号范围说明）
Dist2　　指定 axis2 的位移值，单位：脉冲数
synendspeed　　插补结束速度（单位：脉冲/秒）
synspeed　　插补速度（单位：脉冲/秒）
synacc　　加速时间（单位：秒）
posi_mode　　位移方式（0 表示相对位置，1 表示绝对位置）

返回值：错误码

uint32 MC_line3_muticoor(MCHANDLE phandle, WORD coor, WORD *axi s, long Dist1, long Dist2, long Dist3, long synendspeed, long synspeed, dou ble synacc, WORD posi_mode)

功　能：指定任意三轴做单段直线插补运动

参　数：phandle　　链接句柄
coor　　插补器序号（参考列表）
*axis　　轴号列表的指针（轴号范围说明）
Dist1　　指定 axis[0]轴的位移值，单位：脉冲数
Dist2　　指定 axis[1]轴的位移值，单位：脉冲数
Dist3　　指定 axis[2]轴的位移值，单位：脉冲数
synendspeed　　插补结束速度（单位：脉冲/秒）
synspeed　　插补速度（单位：脉冲/秒）
synacc　　加速时间（单位：秒）
posi_mode　　位移方式（0 表示相对位置，1 表示绝对位置）

返回值：错误码

uint32 MC_lineN_muticoor(MCHANDLE phandle, WORD coor, WORD iaxi snum, WORD *iaxislist, long *poslist, long synendspeed, long synspeed, dou ble synacc, WORD posi_mode)

功　能：指定多轴做单段直线插补运动

参　数：phandle　　链接句柄
coor　　插补器序号（参考列表）
iaxisnum　　参与插补轴数（轴数说明）
*iaxislist　　插补轴号列表（轴号范围说明）

*poslist 位置列表
synendspeed 插补结束速度（单位：脉冲/秒）
synspeed 插补速度（单位：脉冲/秒）
synacc 加速时间（单位：秒）
posi_mode 位移方式（0 表示相对位置，1 表示绝对位置）
返回值：错误码

uint32 MC_arc_move_muticoor(MCHANDLE phandle, WORD coor, WORD *iaxislist, long *rel_pos, long *rel_cen, WORD arc_dir, long synendspeed, double synspeed, double synacc)

功 能：指定任意的两轴以当前位置为起点，按指定的圆心、目标相对位置和方向作圆 弧插补运动（绝对位置）（单段两轴圆弧插补）
参 数：phandle 链接句柄
coor 插补器序号（参考列表）
\ *iaxislist 插补轴号列表（轴号范围说明）
*rel_pos 目标绝对位置列表指针
*rel_cen 圆心绝对位置列表指针
arc_dir 圆弧方向（0 表示顺时针，1 表示逆时针）
synendspeed 插补结束速度（单位：脉冲/秒）
synspeed 插补速度（单位：脉冲/秒）
synacc 插补加速时间（单位：秒）
返回值：错误码

uint32 MC_rel_arc_move_muticoor(MCHANDLE phandle, WORD coor, WORD *iaxislist, long *rel_pos, long *rel_cen, WORD arc_dir, long synendspeed, double synspeed, double synacc)

功 能：指定任意的两轴以当前位置为起点，按指定的圆心、目标相对位置和方向作圆 弧插补运动（相对位置）（单段两轴圆弧插补）
参 数：phandle 链接句柄
coor 插补器序号（参考列表）
\ *iaxislist 插补轴号列表（轴号范围说明）
*rel_pos 目标相对位置列表指针
*rel_cen 圆心相对位置列表指针
arc_dir 圆弧方向（0 表示顺时针，1 表示逆时针）
synendspeed 插补结束速度（单位：脉冲/秒）
synspeed 插补速度（单位：脉冲/秒）
synacc 插补加速时间（单位：秒）
返回值：错误码

3.1.2.6.2 连续插补速度设置

注：1.当处于速度前瞻模式中时，参数中的 synendvel（插补结束速度）不起作用。
2.以下函数为连续插补运动相关函数。
3.不带插补器序号参数的函数默认使用 0 号插补器进行插补运动。
4.EtherCAT 系列点位版本不支持连续插补功能。
5.在非速度前瞻模式下，当前插补段的插补结束速度(synendvel)是下一个插补段的起始速度。

插补器范围	0～2

uint32 MC_conti_set_mode_muticoor(MCHANDLE phandle, WORD CrdNum, double conti_startsp, double conti_synspeed, double conti_synacc, double mineventime)

功 能：设置连续插补参数（带插补器参数）
参 数： phandle 链接句柄
CrdNum 插补器序号（参考列表）
conti_startsp 起始速度（单位：脉冲/秒）
conti_synspeed 最大速度（单位：脉冲/秒）(插补段每段速度不能超过该参数)
conti_synacc 加速时间（单位：秒）(插补段每段加速时间不能超过该参数)
mineventime 削尖峰时间（单位：秒）
返回值：错误码
注：在进行连续插补运动时请一定调用该函数配置连续插补运动的运动参数，否则会引起连续插补运动不起作用。conti_synspeed（最大速度）会限制每一个插补线段的最大插补运行速度。

uint32 MC_conti_get_mode_muticoor(MCHANDLE phandle, WORD CrdNum, double* conti_startsp, double* conti_synspeed, double* conti_synacc, double* mineventime)

功 能：读取连续插补参数（带插补器参数）
参 数： phandle 链接句柄
CrdNum 插补器序号（参考列表）
*conti_startsp 起始速度（单位：脉冲/秒）
*conti_synspeed 最大速度（单位：脉冲/秒）
*conti_synacc 加速时间（单位：秒）
*mineventime 削尖峰时间（单位：秒）
返回值：错误码

3.1.2.6.3 连续插补状态设置

uint32 MC_conti_open_list_muticoor(MCHANDLE phandle, WORD CrdNum, WORD axisNum, WORD *piaxisList)

功 能：打开插补缓冲区（带插补器参数）

参 数： phandle 链接句柄
CrdNum 插补器序号（参考列表）
axisNum 轴数（轴号范围说明）
*piaxisList 连续插补轴号列表

返回值：错误码

注意：在打开插补缓冲区后，插补轴号列表上的所有轴进入连续插补模式；只有当插补运行结束或者调用上位机停止指令终止插补运动后，参与连续插补运动的轴才能退出连续插补运动模式。

uint32 MC_conti_close_list_muticoor(MCHANDLE phandle, WORD CrdNum)

功 能：关闭插补缓冲区（带插补器参数）

参 数： phandle 链接句柄
CrdNum 插补器序号（参考列表）

返回值：错误码

注：如果在插补运动结束后不调用该函数，会导致 **MC_check_done** 函数返回值一直为 0，即检测到指定轴仍在运行之中。

uint32 MC_conti_pause_list_muticoor(MCHANDLE phandle, WORD CrdNum)

功 能：插补暂停（带插补器参数）

参 数： phandle 链接句柄
CrdNum 插补器序号（参考列表）

返回值：错误码

注意：当插补暂停后，可以再次调用 MC_conti_start_list_muticoor 指令，此时运动控制器将继续运行之前未完成的连续插补轨迹。

uint32 MC_conti_set_pause_output_muticoor(MCHANDLE phandle, WORD icoor, WORD iaction, uint32 iomask, uint32 ioutval)

功 能：设置暂停输出 IO（带插补器参数）

参 数： phandle 链接句柄
icoor 插补器序号（参考列表）
iaction 动作
0- 不进行任何动作
1- 暂停时指定输出口输出，恢复时不恢复暂停前 IO 动作
2- 暂停时指定输出口输出，恢复时恢复暂停前 IO 动作
3- 停止时指定输出口输出

IoMask 输出口标志（bit0—bit15 对应 out0-out15，对应的位为 0，不进行任何操作，对应的位为 1，将指定输出口置为 IoValue 相应位的值）（范围：OUT0-15）
12、13、14、15 号输出口为高速输出口
IoValue 输出电平状态（bit0—bit15 对应 out0-out15，把对应位置 0：打开输出口，把对应位置 1：关闭输出口

返回值：错误码
注意：如果需要该函数生效，需要在调用 **MC_conti_pause_list_muticoor** 前设置

uint32 MC_conti_start_list_muticoor(MCHANDLE phandle, WORD CrdNum)
功 能：启动插补运动（带插补器参数）
参 数： phandle 链接句柄
CrdNum 插补器序号（参考列表）
返回值：错误码

uint32 MC_conti_decel_stop_list_muticoor(MCHANDLE phandle, WORD CrdNum)
功 能：插补减速停止（带插补器参数）
参 数： phandle 链接句柄
CrdNum 插补器序号（参考列表）
返回值：错误码

uint32 MC_conti_sudden_stop_list_muticoor(MCHANDLE phandle, WORD CrdNum)
功 能：插补立即停止（带插补器参数）
参 数： phandle 链接句柄
CrdNum 插补器序号（参考列表）
返回值：错误码
注意：可以调用该函数清除连续插补缓冲

uint32 MC_conti_delay_muticoor(MCHANDLE phandle, WORD CrdNum, WORD delaytime)
功 能：缓冲区延时（带插补器参数）
参 数： phandle 链接句柄
CrdNum 插补器序号（参考列表）
delaytime 延时时间（单位：ms）
返回值：错误码

uint32 MC_conti_check_done_muticoor(MCHANDLE phandle, WORD coor)

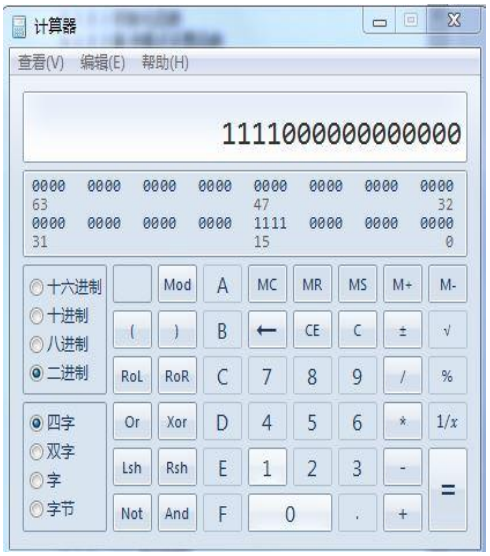
功　能：插补过程中检测插补状态（带插补器参数）
参　数：　phandle　　链接句柄
　　coor　　插补器序号（参考列表）
返回值：1-插补完成，0-插补未完成
注意：**MC_conti_check_done_muticoor** 可以在关闭插补缓冲区后检测插补状态；即使当前插补轴处于暂停状态，此函数也不会返回 1（即插补运动完成），只有当所有轴插补运动和插补 IO 结束才会返回 1（即插补运动完成）。

WORD MC_conti_state_muticoor(MCHANDLE phandle, WORD coor)
功　能：检测插补运动状态（带插补器参数）
参　数：　phandle　　链接句柄
　　coor　　插补器序号（参考列表）
返回值：1-插补运行中　2-插补暂停　3-所有轴插补停止
注意：如果在连续插补过程中未关闭插补缓冲区时，需要检测所有插补段是否已经完成，可以通过 **MC_conti_check_remain_space_muticoor** 读取连续插补剩余缓冲段数和用 **MC_conti_state_muticoor** 检查插补运动状态；当连续插补剩余缓冲段数为 4096 段和插补运动状态为插补插补暂停或所有轴插补停止条件都满足的情况下认为插补运动已经完成，插补运动处于暂停状态，可以继续添加新的线段进行插补。
附加说明：当插补运动处于插补暂停状态（即 2）时，此时所有的运动状态（MC_check_done）返回 1 即为停止状态，允许参与连续插补的所有轴运行点位运动。最后当连续插补运动需要重新启动前，请手动把参与连续插补的所有轴运行回插补暂停时的位置上。

3.1.2.6.4　连续插补 IO 控制

uint32 MC_conti_io_muticoor(MCHANDLE phandle, WORD CrdNum, long IoMask, long IoValue)
功　能：插补缓冲区 IO（带插补器参数）
参　数：　phandle　　链接句柄
　　CrdNum　　插补器序号（参考列表）
　　IoMask　　输出口标志（bit0—bit15 对应 out0-out15，对应的位为 0，不进行任何操作，对应的位为 1，将指定输出口置为 IoValue 相应位的值）（范围：OUT0-15）
　　12、13、14、15 号输出口为高速输出口
　　IoValue　　输出电平状态（bit0—bit15 对应 out0-out15，把对应位置 0：打开输出口，把对应位置 1：关闭输出口
返回值：错误码
例子：MC_conti_io_muticoor(0,0,0x1000,0x000);　　//打开第 12 号输出口
　　MC_conti_io_muticoor(0,0,0x1000,0x1000);　　//关闭第 12 号输出口
　　MC_conti_io_muticoor(0,0,0x2000,0x000);　　//打开第 13 号输出口
　　MC_conti_io_muticoor(0,0,0x2000,0x2000);　　//关闭第 13 号输出口

MC_conti_io_muticoor(0,0,0xF000,0x000); //同时打开第 12,13,14,15 号输出口
MC_conti_io_muticoor(0,0,0xF000,0xF000); //同时关闭第 12,13,14,15 号输出口



uint32 MC_conti_io_action_remained(MCHANDLE phandle, WORD CrdNum)

功 能：查询插补 IO 剩余缓冲大小（带插补器参数）
参 数： phandle 链接句柄
CrdNum 插补器序号（参考列表）
返回值：插补 IO 剩余缓冲大小（最大为 256 段）

uint32 MC_conti_clear_io_action(MCHANDLE phandle, WORD CrdNum, uint32 Io_Mask)

功 能：清除插补 IO 缓冲（带插补器参数）
参 数： phandle 链接句柄
CrdNum 插补器序号（参考列表）
Io_Mask 保留
返回值：错误码
注意：使用 MC_conti_delay_outbit_to_start，MC_conti_delay_outbit_to_stop，MC_conti_ahead_outbit_to_stop 前必须调用 MC_conti_clear_io_action 清除插补 IO 缓冲并且初始化，否则相关函数功能不起作用。

uint32 MC_conti_delay_outbit_to_start(MCHANDLE phandle, WORD CrdNum, WORD bitno, WORD on_off, WORD delay_value, WORD delay_mode, double ReverseTime)

功 能：连续插补中相对于轨迹段起点 IO 滞后输出（带插补器参数）
参 数： phandle 链接句柄
CrdNum 插补器序号（参考列表）

bitno	指定输出口序号（范围：0-23）
on_off	0-打开输出口 1-关闭输出口
delay_value	延后的距离或者时间 距离：（单位：脉冲）（超出下段轨迹距离则在下段轨迹终点执行） 时间：（单位：ms）
delay_mode	延时模式（1-时间 2-距离）
ReverseTime	翻转时间（单位：ms）（设为 0 则 IO 电平不翻转）

返回值：错误码

注意：1）使用该指令前必须调用 MC_conti_clear_io_action 清除插补 IO 缓冲并且初始化，否则相关函数功能不起作用。
2）设置的 IO 操作，将在该指令的下一条轨迹中起作用。
3）延时模式设置为 2（距离）时，位置源为指令位置计数器。

uint32 MC_conti_delay_outbit_to_stop(MCHANDLE phandle, WORD CrdNum, WORD bitno, WORD on_off, WORD delay_time, double ReverseTime)

功 能：连续插补中相对于轨迹段终点 IO 滞后输出（带插补器参数）

参 数：

phandle	链接句柄
CrdNum	插补器序号（参考列表）
bitno	指定输出口序号（范围：0-23）
on_off	0-打开输出口 1-关闭输出口
delay_value	延后时间（单位：ms）
ReverseTime	翻转时间（单位：ms）（设为 0 则 IO 电平不翻转）

返回值：错误码

注意：1）使用该指令前必须调用 MC_conti_clear_io_action 清除插补 IO 缓冲并且初始化，否则相关函数功能不起作用。
2）设置的 IO 操作，将在该指令的下一条轨迹中起作用。
3）延时模式设置为 2（距离）时，位置源为指令位置计数器。

uint32 MC_conti_ahead_outbit_to_stop(MCHANDLE phandle, WORD CrdNum, WORD bitno, WORD on_off, WORD ahead_value, WORD ahead_mode, double ReverseTime)

功 能：连续插补中相对于轨迹段终点 IO 提前输出（带插补器参数）

参 数：

phandle	链接句柄
CrdNum	插补器序号（参考列表）
bitno	指定输出口序号（范围：0-23）
on_off	0-打开输出口 1-关闭输出口
ahead_value	提前的距离或者时间 距离：（单位：脉冲） 时间：（单位：ms）

ahead_mode 延时模式（1-时间 2-距离）
ReverseTime 翻转时间（单位：ms）（设为 0 则 IO 电平不翻转）
返回值：错误码
注意：1）使用该指令前必须调用 MC_conti_clear_io_action 清除插补 IO 缓冲并且初始化，否则相关函数功能不起作用。
2）设置的 IO 操作，将在该指令的下一条轨迹中起作用。
3）延时模式设置为 2（距离）时，位置源为指令位置计数器。

uint32 MC_conti_pwm_muticoor(MCHANDLE phandle, WORD CrdNum, WORD channel, WORD duty, WORD freq, WORD timems)
功 能：在插补缓冲中添加 PWM 输出（带插补器参数）
参 数： phandle 链接句柄
CrdNum 插补器序号（参考列表）
channel 通道号：0–3 （对应 OUT12-15）
duty 占空比 范围（0～100），0 是常开，100 是常闭
freq 频率（单位：赫兹）（范围：1-5000）
timems 输出时间（单位：100us）（范围：0-65535）
（设为 0 则关闭 PWM，设为 65535 则一直输出 PWM）
返回值：错误码
注：对应输出口 1、2、3、4，频率*输出时间=打点个数，当频率设为 1k，输出时间为 20ms，打点数为 20 个；该函数在插补过程当中不在插补 IO 缓冲空间中，而是在连续插补缓冲空间中。

uint32 MC_conti_canio_da_muticoor(MCHANDLE phandle, WORD CrdNum, WORD canid, WORD channel, WORD daval)
功 能：在插补缓冲中添加 DA 输出（带插补器参数）
参 数： phandle 链接句柄
CrdNum 插补器序号（参考列表）
canid 扩展 CAN 模块 ID 号(1-4)
channel 通道号(0-3)
daval 输出 DA 值(0-4096) 对应（电压输出：0～10V； DAC 分辨率：12 位）
返回值：错误码

uint32 MC_conti_wait_input_muticoor(MCHANDLE phandle, WORD CrdNum,WORD IoNum, WORD IoValue, uint32 Timems, long iMark)
功 能：连续插补等待 IO 输入信号再向下运行
参 数： phandle 链接句柄
CrdNum 插补器序号（参考列表）

IoNum	输入口序号
IoValue	电平状态（0—表示低电平，1—表示高电平）
Timems	超时时间（单位：ms，最大 65535）
iMark	指定该段 mark 序号（可通过 MC_conti_read_current_mark_muticoor 获取当前插补器实际运行到哪一段插补缓冲）

返回值：错误码

3.1.2.6.5 连续插补缓冲区检测

uint32 MC_conti_check_remain_space_muticoor(MCHANDLE phandle, WORD CrdNum)（带插补器参数）

功　能：读取连续插补剩余缓冲段数

参　数：phandle　链接句柄

　　CrdNum　插补器序号（参考列表）

返回值：连续插补剩余缓冲段数

uint32 MC_conti_read_current_mark_muticoor(MCHANDLE phandle, WORD CrdNum)

功　能：读取当前运动段 mark 标志（带插补器参数）

参　数：phandle　链接句柄

　　CrdNum　插补器序号（范围：0 或 1）

返回值：当前连续插补运动段 mark 标志

注意：当前插补运动段序号会保持为最后一次运动的插补段序号，不受其他条件影响。

double MC_conti_get_total_length_muticoor(MCHANDLE phandle, WORD coor)

功　能：读取连续插补总插补长度（带插补器参数）

参　数：phandle　链接句柄

　　coor　插补器序号（参考列表）

返回值：当前连续插补总插补长度

double MC_conti_get_remain_length_muticoor(MCHANDLE phandle, WORD coor)

功　能：读取连续插补缓冲剩余插补长度（带插补器参数）

参　数：phandle　链接句柄

　　coor　插补器序号（参考列表）

返回值：当前连续插补缓冲剩余插补长度

3.1.2.6.6　连续插补运动

uint32 MC_conti_set_corner_arc_radius_muticoor(MCHANDLE phandle, W

ORD CrdNum, double radius)

功 能：配置插补运动拐角圆弧过渡（带插补器参数）

参 数： phandle 链接句柄

CrdNum 插补器序号（参考列表）

radius 圆弧半径（单位：脉冲）

返回值：错误码

uint32 MC_conti_pmove_muticoor(MCHANDLE phandle, WORD CrdNum, WORD iaxis, long ipos, WORD ifabs, WORD iwaitdone, WORD iMark)

功 能：直线插补点位运动功能（带插补器参数）

参 数： phandle 链接句柄

CrdNum 插补器序号（参考列表）

iaxis 指定轴号（轴号范围说明）

ipos 点位运动目标位置

ifabs 位移模式设定：0 表示相对位移，1 表示绝对位移

iwaitdone 插补点位运动的模式 ：0-表示启动点位运动下一段不等待
1-等待当前点位运动结束运行下一段

iMark 指定该段 mark 序号（可通过 MC_conti_read_current_mark_muticoor 获取当前插补器实际运行到哪一段插补缓冲）

返回值：错误码

uint32 MC_conti_extern_lines_muticoor(MCHANDLE phandle, WORD CrdNum, WORD axisNum, WORD *piaxisListw, long *pPosList, double synspeed, double synacc, double synendvel, WORD posi_mode, long imark)

功 能：连续直线插补（imark-指定该段 mark 标志）（带插补器参数）

参 数： phandle 链接句柄

CrdNum 插补器序号（参考列表）

axisNum 轴数

*piaxisListw 连续插补轴号列表（轴号范围说明）

*pPosList 位置列表

synspeed 插补速度（单位：脉冲/秒）

synacc 加速时间（单位：秒）

synendvel 插补结束速度（单位：脉冲/秒）

posi_mode 位移方式（0 表示相对位置，1 表示绝对位置）

imark 指定该段 mark 序号（可通过 MC_conti_read_current_mark_muticoor 获取当前插补器实际运行到哪一段插补缓冲）

返回值：错误码

。

uint32 MC_conti_extern_arc_muticoor(MCHANDLE phandle, WORD CrdNum,WORD *axis, long *rel_pos, long *rel_cen, WORD arc_dir, double syns

peed, double synacc, double synendvel, WORD posi_mode, long imark)

功　能：连续两轴圆弧插补（imark-指定该段 mark 标志）（带插补器参数）

参　数：

phandle	链接句柄
CrdNum	插补器序号（参考列表）
*axis	轴号列表指针（轴号范围说明）
*rel_pos	终点位置列表指针（单位：脉冲数）
*rel_cen	圆心位置列表指针（单位：脉冲数）
arc_dir	圆弧方向（0 表示顺时针，1 表示逆时针）
synspeed	插补速度（单位：脉冲/秒）
synacc	加速时间（单位：秒）
synendvel	插补结束速度（单位：脉冲/秒）
posi_mode	位移方式（0 表示相对位置，1 表示绝对位置）
imark	指定该段 mark 序号（可通过 MC_conti_read_current_mark_muticoor 获取当前插补器实际运行到哪一段插补缓冲）

返回值：错误码

注意：当前位置作为圆弧或圆的起点坐标，该函数通过起点（圆上一点）、圆心、圆弧方向、终点坐标（圆上一点）来确定一个圆或圆弧。

uint32 MC_conti_extern_arc_3p_muticoor(MCHANDLE phandle, WORD CrdNum, WORD *axis, long *rel_midpos, long *rel_endpos, double synspeed, double synacc, double synendvel, WORD posi_mode, long imark)

功　能：连续三点圆弧插补（imark-指定该段 mark 标志）（带插补器参数）（两轴圆弧插补）

参　数：

phandle	链接句柄
CrdNum	插补器序号（参考列表）
*axis	轴号列表指针（轴号范围说明）
*rel_midepos	中间位置列表指针（单位：脉冲数）
*rel_endpos	目标位置列表指针（单位：脉冲数）
synspeed	插补速度（单位：脉冲/秒）
synacc	加速时间（单位：秒）
synendvel	插补结束速度（单位：脉冲/秒）
posi_mode	位移方式（0 表示相对位置，1 表示绝对位置）
imark	指定该段 mark 序号（可通过 MC_conti_read_current_mark_muticoor 获取当前插补器实际运行到哪一段插补缓冲）

返回值：错误码

注意：起点、中间点和终点不可以在同一直线上。

uint32 MC_conti_line_glue(MCHANDLE phandle, WORD CrdNum, WORD axisNum, WORD *piaxisListw, long *pPosList, double synspeed, double synacc, double synendvel, WORD posi_mode, long imark, WORD pwmChannel, double *pwmSetPara, double Aheaddelay, double FinishDelay, double SegLength)

功　能：连续两轴直线插补分段均匀打点

参 数： phandle 链接句柄
CrdNum 插补器序号（参考列表）
piaxisListw 轴号列表指针
pPosList 目标位置列表指针
synspeed 插补速度（单位：脉冲/秒）
synacc 加速时间（单位：秒）
synendvel 插补结束速度（单位：脉冲/秒）
posi_mode 位移方式（0 表示相对位置，1 表示绝对位置）
imark imark 指定该段 mark 标志
pwmChannel PWM 通道号（范围：0-3 对应 OUT12-OUT15）
pwmSetPara PWM 参数列表指针 0- duty 占空比 范围（0～100，0 是常开，100 是常闭
1- frq 频率 （单位：赫兹）（范围：1-5000）
2- time100us 输出时间（单位：100us）
Aheaddelay PWM 输出前延时时间（单位：ms）
FinishDelay PWM 输出后延时时间（单位：ms）
SegLength 分段长度（单位：脉冲）
返回值：错误码

连续插补例程：

```
WORD iaxislist[2]={0,1};
long poslist0[2]={10000,10000};
long poslist1[2]={20000,20000};

MC_conti_set_mode_muticoor(g_handle,0,100,80000,0.05,0.00);
//设置控制器连续插补运动起始速度为 100 脉冲/秒，最大速度为 80000 脉冲/秒，加速时间为 0.05，削尖峰时间为 0。

MC_conti_open_list_muticoor (g_handle,0,2,iaxislist);
//设置控制器，轴数为 2，轴号列表为 0 和 1 打开插补缓冲。

MC_conti_lookahead_init_muticoor(g_handle,0,100,0.1,300000);
//设置控制器速度前瞻 100 段，拐角时间为 0.1 秒，拐角最大加速度为 300000 脉冲/秒 2。
//以上初始化函数请勿封装在函数中反复调用，放在程序的初始化部分调用该函数一次即可。
//在速度前瞻模式中插补结束速度参数无效，插补结束点速度受速度规划影响

MC_conti_extern_lines_muticoor (g_handle,0,2,iaxislist,poslist0,60000,0.05,100,1);
//设置控制器轴数为 2，插补矢量速度是 60000 脉冲/秒，加速时间为 0.05 秒，插补结束速度为 100 脉冲/秒，绝对位置模式。

MC_conti_extern_lines_muticoor (g_handle,0,2,iaxislist,poslist1,60000,0.05,100,1);
```

//设置控制器轴数为 2，插补矢量速度是 60000 脉冲/秒，加速时间为 0.05 秒，插补结束速度为 100 脉冲/秒，绝对位置模式。

MC_conti_pushdata_muticoor(g_handle,0,1);
//设置控制器压入数据阻塞模式。建议在线程使用时选择非阻塞模式，在按钮中使用时选择阻塞模式。
//在速度前瞻模式下插补线段会优先进入速度前瞻缓冲区，需要调用该函数把放在速度前瞻缓冲区中的插补线段压入连续插补缓冲区进行连续插补。

MC_conti_start_list_muticoor(g_handle,0);
//启动控制器插补运动。

MC_conti_close_list_muticoor(g_handle,0);
//关闭控制器插补缓冲区。

3.1.2.6.7 速度前瞻函数

uint32 MC_conti_pushdata_muticoor(MCHANDLE phandle, WORD CrdNum, WORD mode)

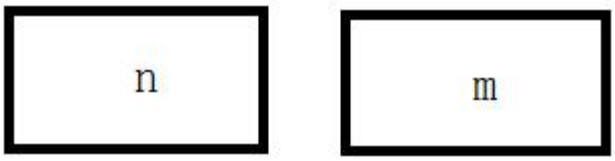
功 能：将前瞻缓冲区数据压入控制器（带插补器参数）
参 数： phandle 链接句柄
CrdNum 插补器序号（参考列表）
mode 阻塞方式：
0——非阻塞方式：
1——阻塞方式：阻塞模式就是一直等待，直到压完插补线段为止
返回值：0-全部前瞻缓存已经压入，没有任何错误。90000- 控制器插补缓冲区已经满了，但还有缓存插补段存在前瞻缓冲区，需等待控制器插补缓冲区有剩余空间时再次调用该指令。
注：如果缓冲区满，需多次压入。
编程建议：在线程中使用该函数建议用阻塞方式，在按钮的消息响应函数中使用该函数建议用非阻塞方式。

uint32 MC_conti_lookahead_init_muticoor(MCHANDLE phandle, WORD CrdNum, WORD LookAheadNum, double corner_time, double corner_accmax)

功 能：速度前瞻初始化
参 数： phandle 链接句柄
CrdNum 插补器序号（参考列表）
LookAheadNum 前瞻段数
corner_time 拐角时间（单位：秒）
corner_accmax 拐角最大加速度（脉冲/秒^2）注：此处是加速度不是加速时间
注意：拐角速度=拐角时间乘以拐角最大加速度

返回值：错误码

提示：MC_conti_lookahead_init_muticoor 函数设置加速度记为 ACC1，MC_conti_set_mode_muticoor 设置的加速时间和最大速度也会有一个加速度，记为 ACC2。
(1) ACC2 用于限制接下来的直线、圆弧等线段的最大加速度，如果后面线段的加速度设置超过该值，以 ACC2 为准。
(2) ACC1 和 corner_time(拐角时间)的乘积为拐角速度，这个速度记为 V，后面拐角的速度会根据矢量速度夹角和 V 成一定比例关系，但拐角速度不会超过 V
(3) ACC1 还用于小圆限速，ACC1 越大圆弧速度影响越小，否则圆弧会降速。
(4)由于 ACC1 和 corner_time(拐角时间)的乘积为拐角速度，再调整时，可以加大 ACC1，减小 corner_time(拐角时间),保证乘积不变，这样不会影响圆弧速度。
(5) 只要乘积不变拐角速度就不受影响。



速度前瞻缓冲区　　连续插补缓冲区

连续插补中使用速度前瞻详解：

我们把速度前瞻缓冲区的大小设为 n 段（速度前瞻缓冲区的大小等于速度前瞻函数设置的前瞻段数参数），把连续插补缓冲区剩余缓冲的大小设为 m（默认连续插补缓冲区大小为 4096 段），我们把客户实际运用时需要连续插补的段数设为 N。

当（N<n）时，m=4096　//因为在速度前瞻模式下，连续插补中需要插补的每一段优先放进速度前瞻缓冲区里面，当速度前瞻缓冲区未被填满时（即 N<n 时），连续插补缓冲是不会到连续插补缓冲区里面的，所以 m=4096。

当（n<N<4096+n）时，m=4096-(N-n)　//当速度前瞻缓冲区被填满时，多出的连续插补缓冲段就会自动放入连续插补缓冲区当中，所以 m=4096-(N-n)。

当（N>4096+n）时　//即速度前瞻缓冲区和连续插补缓冲区都被填满时，可调用 MC_conti_check_remain_space_muticoor 读取当前剩余插补缓冲段数。

例如：

```
while(1)
    {
     //这条语句判断剩余插补缓冲数是否小于等于 1
        if(MC_conti_check_remain_space_muticoor(g_handle,0) <= 1)
        {
            AfxGetApp()->PumpMessage();  //不加这条指令会导致程序卡死
            Sleep(1);
        }
        else
        {
```

```
                break;
            }
        }
```

3.1.2.6.8 刀向跟随函数

uint32 MC_conti_gear_muticoor(MCHANDLE phandle, WORD CrdNum, WORD axisNum, WORD *piaxisListw,long *pPosList)

功 能：刀向跟随功能（后续必须紧跟插补运动指令）

参 数：

phandle	链接句柄
CrdNum	插补器序号（参考列表）
axisNum	轴数
*piaxisListw	插补轴号列表
*pPosList	位置列表（绝对位置坐标）

返回值：错误码

注意：下条插补运动指令如果运行的矢量长度为 0，刀向跟随不生效。

3.1.2.7 驱动器专用接口信号的设定函数

uint32 MC_config_ALM_PIN(MCHANDLE phandle, WORD axis, WORD enable, WORD alm_logic, WORD alm_action)

uint32 MC_get_config_ALM_PIN(MCHANDLE phandle, WORD axis, WORD* enable, WORD* alm_logic, WORD* alm_action)

功 能：设置/读取 ALM 信号使能和有效电平及其工作方式

参 数：

phandle	链接句柄
axis	指定轴号（轴号范围说明）
enable	使能/禁止信号功能：0—无效，1—有效
alm_logic	ALM 信号的输入电平：0—低电平有效，1—高电平有效
alm_action	ALM 信号的制动方式：0—立即停止，1—减速停止

返回值：错误码

uint32 MC_config_EZ_PIN(MCHANDLE phandle, WORD axis, WORD ez_logic, WORD ez_mode)

uint32 MC_get_config_EZ_PIN(MCHANDLE phandle, WORD axis, WORD * ez_logic, WORD* ez_mode)

功 能：设置/读取指定轴的 EZ 信号的有效电平及其作用

参 数：

phandle	链接句柄
axis	指定轴号（轴号范围说明）
ez_logic	EZ 信号有效电平：0—低有效，1—高有效
ez_mode	EZ 信号的工作方式： 0—EZ 信号无效

1—EZ 是计数器复位信号

返回值：错误码

uint32 MC_config_LTC_PIN(MCHANDLE phandle, WORD axis, WORD ltc_logic, WORD ltc_mode)

uint32 MC_get_config_LTC_PIN(MCHANDLE phandle, WORD axis, WORD* ltc_logic, WORD* ltc_mode)

功　能：设置/读取指定轴“锁存”信号的有效电平及其和工作方式。

参　数：phandle　　链接句柄

axis　　指定轴号（轴号范围说明）

ltc_logic　　LTC 信号逻辑电平：0—低有效，1—高有效

ltc_mode　　0—EZ 信号锁存，1—原点信号锁存

返回值：错误码

注意：在使用原点信号锁存功能的时候，无法正常读取伺服的 EZ 信号。

uint32 MC_config_EL_PIN(MCHANDLE phandle, WORD axis, WORD enable, WORD el_logic, WORD el_mode)

uint32 MC_get_config_EL_PIN(MCHANDLE phandle, WORD axis, WORD* enable, WORD* el_logic, WORD* el_mode)

功　能：设置/读取 EL 信号的有效电平及制动方式

参　数：phandle　　链接句柄

axis　　指定轴号（轴号范围说明）

enable　　允许/禁止信号功能：0—正负限位禁止
1—正负限位使能
2—正限位禁止、负限位使能
3—正限位使能、负限位禁止

el_logic　　EL 有效电平：0—正负限位低电平有效
1—正负限位高电平有效
2—正限位低电平有效，负限位高有效
3—正限位高电平有效，负限位低有效

el_mode　　EL 有效电平和制动方式：
0—立即停止
1—减速停止

返回值：错误码

uint32 MC_config_HOME_PIN_logic(MCHANDLE phandle, WORD axis, WORD org_logic, WORD filter)

uint32 MC_get_config_HOME_PIN_logic(MCHANDLE phandle, WORD axis, WORD* org_logic, WORD* filter)

功　能：设置/读取 ORG 信号的有效电平，以及允许/禁止滤波功能
参　数：phandle　　链接句柄
　　axis　　指定轴号（轴号范围说明）
　　org_logic　　ORG 信号的有效电平：0—低电平有效，1—高电平有效
　　filter　　保留
返回值：错误码

uint32 MC_write_SEVON_PIN(MCHANDLE phandle, WORD axis, WORD on_off)
功　能：输出对指定轴的伺服使能端子的控制
参　数：phandle　　链接句柄
　　axis　　指定轴号（轴号范围说明）
　　on_off　　设定电平状态：0—低电平，对应打开伺服使能
　　1—高电平，对应关闭伺服使能
返回值：错误码
注意：本函数只适用于脉冲卡的伺服使能端口控制，总线卡请使用 MC_ect_set_axis_enable。

int MC_read_SEVON_PIN(MCHANDLE phandle, WORD axis)
功　能：读取指定轴的伺服使能端子的电平状态
参　数：phandle　　链接句柄
　　axis　　指定轴号（轴号范围说明）
返回值：0—低电平，1—高电平
注意：本函数只适用于脉冲卡的伺服使能端口读取，总线卡请使用 MC_ect_get_axisenable。

uint32 MC_set_backlash(MCHANDLE phandle, WORD axis, long backlash)
uint32 MC_get_backlash(MCHANDLE phandle, WORD axis, long* backlash)
功　能：设置/读取间隙补偿值
参　数：phandle　　链接句柄
　　axis　　指定轴号（轴号范围说明）
　　backlash　　间隙补偿值，　单位：脉冲
返回值：错误码
注意：在轴停止状态下再调用该指令。

int MC_read_RDY_PIN(MCHANDLE phandle, WORD axis)
功　能：读取指定运动轴的“伺服准备好”端子的电平状态
参　数：phandle　　链接句柄
　　axis　　指定轴号（轴号范围说明）
返回值：0—低电平，1—高电平

uint32 MC_write_ERC_PIN(MCHANDLE phandle, WORD axis, WORD sel)

功 能：控制指定轴报警清除端子信号的输出

参 数：phandle 链接句柄

axis 指定轴号（轴号范围说明）

sel 0—输出 ERC 信号，1—关闭 ERC 信号

返回值：错误码

uint32 MC_read_ERC_PIN(MCHANDLE phandle, WORD axis)

功 能：读取指定轴“报警清除”端子信号的电平状态

参 数：phandle 链接句柄

axis 指定轴号（轴号范围说明）

返回值：0—输出 ERC 信号，1—关闭 ERC 信号

uint32 MC_config_EMG_PIN(MCHANDLE phandle, WORD enable, WORD emg_logic)

uint32 MC_get_config_EMG_PIN(MCHANDLE phandle, WORD* enbale, WORD* emg_logic)

功 能: 设置/读取 EMG 信号的有效电平，急停信号有效后会立即停止所有轴。

参 数: phandle 链接句柄

enable 0-禁止 1-使能

emg_logic: EMG 信号的有效电平：0—低电平有效，1—高电平有效

备注：EMG 信号硬件接口 MC400A 和 MC800A 默认为 IN32

返回值：错误码

WORD MC_axis_io_status(MCHANDLE phandle, WORD axis)

功 能：读取指定轴有关运动信号的状态，包含指定轴的专用 I/O 状态。

参 数： phandle 链接句柄

axis 指定轴号（轴号范围说明）

返回值：见表

位号	信号名称	描述
0	ALM	ALM（伺服报警）信号
1	EL+	EL+（正方向限位）信号
2	EL-	EL-（负方向限位）信号
3	EMG	EMG（急停）信号
4	HOME	HOME（原点）信号
5	保留	保留

6	SL+	软限位信号，最大值
7	SL-	软限位信号，最小值
8	INP	INP（伺服到位）信号
9	HANDA	手轮信号，A 相。
10	RDY	RDY（伺服准备）信号
11	EA	编码器信号，A 相
12	EB	编码器信号，B 相
其它位	保留	

备注：EMG 信号有效时对应位是 0，信号无效时是 1；其余信号有效时对应位是 1，信号无效时是 0。

例程：

```
//读取控制器 0 号轴的轴信号状态
WORD iret = MC_axis_io_status(g_handle,0);
// ALM（伺服报警信号）
if (iret & 0x01)
{
    printf("伺服报警信号触发\n");
}
else
{
    printf("伺服报警信号未触发\n");
}

// EL+（正限位信号）
if (iret & 0x02)
{
    printf("正限位信号触发\n");
}
else
{
    printf("正限位信号未触发\n");
}

// EL-（负限位信号）
if (iret & 0x04)
{
    printf("负限位信号触发\n");
}
else
{
    printf("负限位信号未触发\n");
```

```
}

// EMG（急停信号）
if (iret & 0x08)
{
    printf("急停信号未触发\n");
}
else
{
    printf("急停信号触发\n");
}

// HOME（原点信号）
if (iret & 0x10)
{
    printf("原点信号触发\n");
}
else
{
    printf("原点信号未触发\n");
}
```

WORD MC_axis_io_status_all(MCHANDLE phandle, uint32 *stateall, int32 *pPos,int32 *pEos,int32 *pCurspeed)

功　能：读取所有轴有关运动信号的状态，包含轴的专用 I/O 状态。

参　数：　phandle　　链接句柄

　　stateall　　(数组大小至少为 10)

　　数组元素 0：所有轴原点状态(第 0 位表示第 0 轴状态)

　　数组元素 1：所有轴报警信号状态(第 0 位表示第 0 轴状态)

　　数组元素 2：所有轴 INP 状态(第 0 位表示第 0 轴状态)

　　数组元素 3：所有轴正限位状态(第 0 位表示第 0 轴状态)

　　数组元素 4：所有轴负限位状态(第 0 位表示第 0 轴状态)

　　数组元素 5：所有轴正软限位状态(第 0 位表示第 0 轴状态)

　　数组元素 6：所有轴负软限位状态(第 0 位表示第 0 轴状态)

　　数组元素 7：所有轴 SD 信号状态(第 0 位表示第 0 轴状态)

　　数组元素 8：所有轴单轴停止信号状态(第 0 位表示第 0 轴状态)

　　数组元素 9：所有轴 EMG 信号状态(第 0 位有效)

返回值：错误码

3.1.2.8 手轮运动控制函数

uint32 MC_set_handwheel_inmode(MCHANDLE phandle, WORD axis, WORD inmode, double multi)

uint32 MC_get_handwheel_inmode(MCHANDLE phandle, WORD axis, WORD* inmode, double* multi)

功　能：设置/读取手轮脉冲信号的计数方式

参　数：phandle　　链接句柄

axis　　指定轴号（轴号范围说明）

inmode　　表示输入方式：0 -- 非 A/B 相脉冲信号
1 -- 1 倍 A/B 相脉冲信号
2 -- 2 倍 A/B 相脉冲信号
3 -- 4 倍 A/B 相脉冲信号

multi　　计数器的计数方向及倍率设置：设置手轮的倍率, >=0 表示默认方向, <0 表示与默认方向相反。

返回值：错误码

uint32 MC_handwheel_move(MCHANDLE phandle, WORD axis)

功　能：启动指定轴的手轮脉冲运动

参　数：phandle　　链接句柄

axis　　指定轴号（轴号范围说明）

返回值：错误码

3.1.2.9 凸轮运动控制函数

uint32 MC_ecam_disengage(MCHANDLE phandle, WORD iaxis, WORD mode)

功　能：取消凸轮主轴和从轴链接

参　数：phandle　　链接句柄

iaxis　　指定轴号（轴号范围说明）

mode　　停止模式，当前保留 （1-减速停止 0-立即停止）

返回值：错误码

uint32 MC_ecam_engage(MCHANDLE phandle, WORD iaxis, WORD imaster, long istartpos, WORD ifencoder, WORD mode)

功　能：启动凸轮主轴和从站链接

参　数：phandle　　链接句柄

iaxis　　指定轴号（轴号范围说明）

imaster　　主轴轴号

istartpos	主轴到达该绝对位置时启动跟随（单位：脉冲）
ifencoder	主轴是否为编码器轴（0-不是 1-是）
mode	跟随模式 0-反向跟随（主轴运动方向为负方向时跟随） 1-正向跟随（主轴运动方向为正方向时跟随） 2-双向跟随 3-反向跟随而且自动重复跟随 4-正向跟随而且自动重复跟随 5-双向跟随而且自动重复跟随

返回值：错误码
注意：为了减少跟随滞后，从轴的轴号应大于主轴的轴号。

uint32 MC_ecam_fifo_datain(MCHANDLE phandle, WORD iaxis, long slave_endpos, long master_endpos, float speed_ratio_start, float speed_ratio_end, WORD mode, uint32 iMark)

功　能：凸轮数据压入，运动距离为负向就反向跟随，否则就正向跟随，对于一些输出或其他操作可以使用比较功能

参　数：

phandle	链接句柄
iaxis	指定轴号（轴号范围说明）
slave_endpos	当前段从轴运行的相对长度，即相对于电子凸轮跟随起点的长度
master_endpos	当前段参考轴运行的相对长度，即相对于电子凸轮跟随起点的长度
speed_ratio_start	当前段起点从轴和主轴的速度比率，即从/主速度比率，保留 3 位小数
speed_ratio_end	当前段终点从轴和主轴的速度比率，即从/主速度比率，保留 3 位小数
mode	保留参数
iMark	当前数据段标号

返回值：错误码

uint32 MC_ecam_fifo_clear(MCHANDLE phandle, WORD iaxis)

功　能：凸轮数据清除

参　数：

phandle	链接句柄
iaxis	指定轴号（轴号范围说明）

返回值：错误码

uint32 MC_ecam_fifo_remian(MCHANDLE phandle, WORD iaxis)

功　能：查询凸轮数据剩余段数

参　数：

phandle	链接句柄

iaxis 指定轴号（轴号范围说明）

返回值：凸轮数据剩余段数（最大 64 段）

uint32 MC_ecam_get_curmark(MCHANDLE phandle, WORD iaxis)

功 能：读取当前正在运行的凸轮数据段标号

参 数：phandle 链接句柄

iaxis 指定轴号（轴号范围说明）

返回值：正在运行的凸轮数据段标号

uint32 MC_ecam_state(MCHANDLE phandle, WORD iaxis, WORD *state);

功 能：查询凸轮运动的状态

参 数：phandle 链接句柄

iaxis 指定轴号（轴号范围说明）

state 运动状态

1- 运行

2- 暂停状态

3- 暂停状态

4- 停止状态

返回值：错误码

uint32 MC_ecam_mode(MCHANDLE phandle, WORD iaxis, WORD *mode)

功 能：设置凸轮运动的跟随模式

参 数：phandle 链接句柄

iaxis 指定轴号（轴号范围说明）

mode 跟随模式

0-反向跟随

1-正向跟随

2-双向跟随

3-反向跟随而且自动重复跟随

4-正向跟随而且自动重复跟随

5-双向跟随而且自动重复跟随

返回值：错误码

uint32 MC_ecam_set_repeatlength(MCHANDLE phandle, WORD iaxis, uint32 length)

uint32 MC_ecam_get_repeatlength(MCHANDLE phandle, WORD iaxis, uint32 *length)

功 能：设置或者读取往复运动的模。如设置起始跟随的位置为 0，往复运动的模为 100，则凸轮跟随周期为：100，每 100 位置循环一个周期

参 数：phandle 链接句柄

iaxis 指定轴号（轴号范围说明）

length 凸轮跟随的模，默认为 0

返回值：错误码

uint32 MC_ecam_fifo_ioaction(MCHANDLE phandle, WORD iaxis, uint32 bitmask, uint32 on_off, long delayms, WORD type, uint32 iMark)

功 能：在凸轮点后增加一个凸轮顶杆事件

参 数：phandle 链接句柄

iaxis 指定轴号（轴号范围说明）

bitmask 输出口掩码

on_off 输出电平(0—打开，1—关闭)

delayms 翻转时间（单位：ms）,写 0 不反转

type 保留参数写 0

iMark 保留参数写 0

返回值：错误码

注意：掩码对应位为 0，不进行任何操作，对应的位为 1，将指定输出口置为 on_off 相应位的值。bit0-23 代表 0-23 位输出口。

3.1.2.10 通用 IO 控制函数

int MC_read_inbit(MCHANDLE phandle, WORD bitno)

功 能：读取指定控制器的某一位输入口的电平状态

参 数：phandle 链接句柄

bitno 指定输入口序号

返回值：0 表示低电平；1 表示高电平

控制器类型	bitno	对应输入口序号范围
MC400A	1－32	IN1-IN32
	1－4	ORG0-3
	9-11-13-15	EL0+～EL3+
	10 -12-14-16	EL0-～EL3-
MC800A	1－32	IN1-IN32
	1－8	ORG0-7
	9-11-13-15-17-19-21-23	EL0+～EL7+
	10-12-14-16-18-20-22-24	EL0-～EL7-

uint32 MC_write_outbit(MCHANDLE phandle, WORD bitno, WORD on_off)

功 能：设置指定控制器的某一位输出口的电平状态

参 数：phandle 链接句柄

bitno 指定输出口序号（范围：如下表所示）

on_off 输出电平：0—表示输出低电平，1—表示输出高电平

返回值：错误码

控制器类型	bitno	对应输出口序号范围
MC400A	1 – 24	OUT1-24
MC800A	1 –24	OUT1-24

int MC_read_outbit(MCHANDLE phandle, WORD bitno)

功 能：读取指定控制器的某一位输出口的电平状态

参 数：phandle 链接句柄

bitno 指定输出口位号（取值范围：如下表所示）

返回值：0 表示低电平；1 表示高电平。

控制器类型	bitno	对应输出口序号范围
MC400A	1 – 24	OUT1-24
MC800A	1 –24	OUT1-24

long MC_read_inport(MCHANDLE phandle, WORD iport)

功 能：读取指定控制器的全部通用输入口的电平状态

参 数：phandle 链接句柄

iport 端口号

返回值：bit0 – bit31 位值分别代表第 0 – 31 号输入端口值。

注意：ORG（原点信号）和 ALM（伺服报警信号）可根据有效电平设置变化读取正确的信号状态，其余专用信号则不行（默认按信号低电平有效读取）。

控制器类型	位号	iport = 0
MC400A	0– 31	In1-In32
MC800A	0– 31	In1-In32

long MC_read_outport(MCHANDLE phandle)

功 能：读取指定控制器的全部通用输出口的电平状态

参 数：phandle 链接句柄

返回值：位号 0-23 位对应输出口 OUT1-24

long MC_read_outport_ex(MCHANDLE phandle, WORD iport)

功 能：读取指定控制器的全部输出口的电平状态

参 数：phandle 链接句柄

port 端口号

返回值：

控制器类型	位号	iport = 0
MC400A	0-23	OUT1-OUT24

控制器类型	位号	iport＝0
MC800A	0-23	OUT1-OUT24

uint32 MC_write_outport(MCHANDLE phandle, uint32 port_value)

功　能：设置控制器的全部通用输出口的电平状态

参　数：phandle　　链接句柄

port_value　　输出口端口值（对应位写 1-高电平，0-低电平）

bit0－bit23 位值分别代表第 1－24 号输出端口值。

返回值：错误码

例子：MC_write_outport(0,0xFFFFFF);　//关闭控制器的 OUT0-23

uint32 MC_write_outport_ex(MCHANDLE phandle, WORD port, uint32 port_value)

功　能：设置控制器的全部通用输出口的电平状态

参　数：phandle　　链接句柄

port　　端口号

port_value　　输出口端口值（对应位写 1-高电平，0-低电平）

bit0－bit23 位值分别代表第 1－24 号输出端口值。

返回值：错误码

例子：MC_write_outport(0,0xFFFFFF);　//关闭控制器的 OUT0-23

控制器类型	位号	iport＝0
MC400A	0-23	OUT1-OUT24

控制器类型	位号	iport＝0
MC800A	0-23	OUT1-OUT24

uint32 MC_write_outbit_mask(MCHANDLE phandle, uint32 bitmask, uint32 on_off)

功　能：设置控制器的多个通用输出口的电平状态

参　数：phandle　　链接句柄

Bitmask　　输出端口值（如果掩码对应的位为 0，不进行任何操作，掩码对应的位为 1，将指定输出口置为 on_off 相应位的值）

bit0－bit23 位值分别代表第 1－24 号输出端口值。

on_off　　IO 值（把对应位置 0：打开输出口，对应位置 1：关闭输出口）

返回值：错误码

例子：MC_write_outbit_mask (0,0x7,0x00);　//打开控制器的 OUT0-OUT2

MC_write_outbit_mask (0,0x7,0x7);　//关闭控制器的 OUT0-OUT2

uint32 MC_reverse_outbit(MCHANDLE phandle, WORD bitno, WORD state, dou

ble reversetime)

功　能：输出口延时指定时间后，把当前输出口的信号电平取反

参　数：phandle　　链接句柄
bitno　　指定输出口序号（范围：1-24）
State　　翻转后的状态　0-高电平 1-低电平
reversetimems　　输出口翻转时间（单位：ms）

返回值：错误码

注意：比如当前输出口是高电平状态（1），那么在调用指令后会延时指定时间，等时间到时把当前输出口状态取反，即为低电平状态（0）。

3.1.2.11 输出 PWM 函数

uint32 MC_write_pwm(MCHANDLE phandle, WORD channel, double ioduty,WORD iofreq, WORD timems)

功　能：立即输出 PWM

参　数：phandle　　链接句柄
channel　　通道号 范围（0—3）（对应 OUT1-4）
ioduty　　占空比 范围（0～100），0 是常开，100 是常闭
iofreq　　频率（单位：赫兹）（范围：1-5000）
timems　　输出时间（单位：100us）（范围：0-65535）
（设为 0 则关闭 PWM，设为 65535 则一直输出 PWM）

返回值：错误码

注：对应输出口 1、2、3、4，频率*输出时间=打点个数，当频率设为 1k，输出时间为 20ms，打点数为 20 个。

uint32 MC_write_pwmf(MCHANDLE phandle, WORD channel, double dfduty,WORD iofreq, WORD timems)

功　能：立即输出 PWM

参　数：phandle　　链接句柄
channel　　通道号（范围：0—3）（对应 OUT1-4）
dfduty　　占空比(范围：0-100)浮点型，0 是常开，100 是常闭
iofreq　　频率（单位：赫兹）（范围：1-5000）
timems　　输出时间（单位：100us）（范围：0-65535）
（设为 0 则关闭 PWM，设为 65535 则一直输出 PWM）

返回值：错误码

注：对应输出口 1、2、3、4，频率*输出时间=打点个数，当频率设为 1k，输出时间为 20ms，打点数为 20 个。

3.1.2.12 CAN 扩展 IO 控制函数

uint32 MC_init_canio(MCHANDLE phandle, WORD canid, WORD ifenable,

WORD baudrate)

功 能：初始化 CAN IO 扩展模块

参 数：phandle 链接句柄

canid can io 扩展模块的 ID 号，(取值范围：1－4)

ifenable can io 扩展模块是否使能（0 为不使能，1 为使能）

baudrate 保留参数

返回值：错误码

uint32 MC_canio_linkstate(MCHANDLE phandle, WORD canid, WORD *statelink)

功 能：读取 CAN IO 扩展模块的连接状态

参 数：phandle 链接句柄

canid can io 扩展模块的 ID 号，(取值范围：1－4)

statelink 是否连接(0-断开，1-连接)

返回值：错误码

注意：如果发现 CAN IO 模块连接断开，可以尝试使用 **MC_init_canio** 重连。

uint32 MC_canio_readinbit(MCHANDLE phandle, WORD canid, WORD bitno)

功 能：读取 CAN IO 输入口状态

参 数：phandle 链接句柄

canid can io 扩展模块的 ID 号，(取值范围：1－4)

bitno 指定输入口位号（取值范围：0－19）

返回值：0 表示低电平；1 表示高电平

uint32 MC_canio_readinport(MCHANDLE phandle, WORD canid)

功 能：读取 CAN IO 输入端口的值

参 数：phandle 链接句柄

canid can io 扩展模块的 ID 号，(取值范围：1－4)

返回值：通过读取该函数返回值的第 0-19 位分别获取 CANIO 模块硬件上的 IN1-IN20 的状态。

uint32 MC_canio_readoutbit(MCHANDLE phandle, WORD canid, WORD bitno)

功 能：读取 CAN IO 指定输出口状态

参 数：phandle 链接句柄

canid can io 扩展模块的 ID 号，(取值范围：1－4)

bitno 指定输出口位号（取值范围：0－19）

返回值：0 表示低电平；1 表示高电平

uint32 MC_canio_readoutport(MCHANDLE phandle, WORD canid)

功　能：读取所有 CAN IO 输出口的状态
参　数：phandle　　链接句柄
　　canid　　can io 扩展模块的 ID 号，(取值范围：1－4)
返回值：通过读取该函数返回值的第 0-19 位分别获取 CANIO 模块硬件上的 OUT1-OUT20 的状态。

uint32 MC_canio_writeoutbit(MCHANDLE phandle, WORD canid, WORD bitno, WORD state)

功　能：设置 CAN IO 输出口的状态
参　数：phandle　　链接句柄
　　canid　　can io 扩展模块的 ID 号，(取值范围：1－4)
　　bitno　　指定输出口位号（取值范围：0－19）
　　state　　输出电平：0－表示输出低电平，1－表示输出高电平
返回值：错误码

uint32 MC_canio_writeoutport(MCHANDLE phandle, WORD canid, uint32 state)

功　能：设置 CAN IO 输出端口的值
参　数：phandle　　链接句柄
　　canid　　can io 扩展模块的 ID 号，(取值范围：1－4)
　　state　　bit0－bi19 位值分别代表第 1–20 号输出端口值。
返回值：错误码
例子：MC_canio_writeoutport(0,1,0xFFFFF);　//关闭控制器的 1 号 CANIO 模块的 1–20 号输出口。

uint32 MC_init_can_analog(MCHANDLE phandle, WORD canid, WORD ifenable, WORD type)

功　能：初始化 CAN 模拟量扩展模块
参　数：phandle　　链接句柄
　　canid　　can 扩展模块的 ID 号，(取值范围：1－4)
　　ifenable　　can 扩展模块是否使能（0 为不使能，1 为使能）
　　type　　2- C3400（4 路 DA 和 4 路 AD）
　　4- C2400（4 路 DA 和 4 路 AD）
返回值：错误码

long MC_canio_read_ain(MCHANDLE phandle, WORD canid, WORD channel)

功　能：读取 CAN 模块的指定通道号 AD（模拟量输入）

参 数：phandle 链接句柄
canid can 扩展模块的 ID 号，(取值范围：1－4)
channel 通道号（范围：0-3）
返回值：指定通道号的 AD 值。（范围：0-10000）对应（电压输出：0～10V；电流输出：0～20mA 或 4～20mA）

uint32 MC_canio_read_aout(MCHANDLE phandle, WORD canid, WORD channel)

功 能：读取 CAN 模块的指定通道号 DA（模拟量输出）
参 数：phandle 链接句柄
canid can 扩展模块的 ID 号，(取值范围：1－4)
channel 通道号（范围：0-3）
返回值：指定通道号的 DA 值（范围：0-4096）对应（电压输出：0～10V；电流输出：0～20mA 或 4～20mA； DAC 分辨率：12 位）

uint32 MC_canio_set_aout(MCHANDLE phandle, WORD canid, WORD channel, WORD value)

功 能：设置 CAN 模块的指定通道号 DA（模拟量输出）
参 数：phandle 链接句柄
canid can 扩展模块的 ID 号，(取值范围：1－4)
channel 通道号（范围：0-3）
value DA 值（范围：0-4096）对应（电压输出：0～10V；电流输出：0～20mA 或 4～20mA； DAC 分辨率：12 位）
返回值：错误码

3.1.2.13 输入口高速计数功能函数

uint32 MC_set_h_count_value(MCHANDLE phandle, WORD bitno, uint32 CountValue)

uint32 MC_get_h_count_value(MCHANDLE phandle, WORD bitno, uint32* CountValue)

功 能：设置/读取输入口高速功能计数值
参 数：phandle 链接句柄
bitno 通道号（范围：0-3）
CountValue 计数值
返回值：错误码

3.1.2.14 IO 口映射功能函数

uint32 MC_set_axis_io_map(MCHANDLE phandle, WORD axis, WORD IoType, WORD MapIoType, WORD MapIoIndex, double Filter)

uint32 MC_get_axis_io_map(MCHANDLE phandle, WORD axis, WORD IoType, WORD *MapIoType, WORD *MapIoIndex, double *Filter)

功　能：设置/读取 IO 映射配置

参　数：phandle　　链接句柄

axis　　指定轴号（轴号范围说明）

IoType　　需要映射的类型

0-PEL正限位，1-NEL负限位，2-ORG原点，

3-EMG急停，4-SD减速信号（保留），5-ALM报警信号，

8-ESTOP单轴急停信号）

MapIoType　　目标信号的类型（0-PEL正限位，1-NEL负限位，2-OR 原点，3-ALM报警信号，6-IN通用输入）

（设置成 255 则取消原有映射关系）

MapIoIndex　　轴IO映射索引号

1）当目标信号的类型设置为6时，表示该映射对应的通用输入端口号，具体范围见下表

2）当目标信号的类型设置为0~5时，此参数可设置为该映射所对应的具体轴号，（轴号范围说明）

3）设置该值为255表示取消轴IO映射关系

Filter　　特殊信号滤波时间（单位：250us）

返回值：错误码

例子：MC_set_axis_io_map(g_handle,0,3,6,0,1);　//将急停信号映射至通用输入口 0

控制器类型	输入口序号范围
MC400A	1-32
MC800A	1-32

3.1.2.15 参数文件写入读取函数

uint32 MC_LoadConfig(MCHANDLE phandle, char* fpath)

功　能：把参数文件（.ini 文件）里面保存的参数写入控制器

参　数：phandle　　链接句柄

fpath　　参数文件路径

返回值：错误码

uint32 MC_ReadConfig(MCHANDLE phandle, char* fpath)

功 能：把控制器的参数配置读取出来保存在参数文件（.ini 文件）中

参 数：phandle 链接句柄

fpath 参数文件路径

返回值：错误码

3.1.2.16 控制器设置密码函数

uint32 MC_read_user_password(MCHANDLE phandle)

功 能：读取控制器密码

参 数：phandle 链接句柄

返回值：控制器当前密码

3.1.2.17 输入信号滤波时间函数

uint32 MC_config_special_input_filter(MCHANDLE phandle, double timems)

uint32 MC_get_config_special_input_filter(MCHANDLE phandle, double *timems)

功 能：设置/读取特殊输入信号滤波时间（包括限位(±EL)，伺服报警(ALM)，急停(EMG)信号）

参 数：phandle 链接句柄

timems 滤波时间（单位：250us）

返回值：错误码

uint32 MC_config_input_filter(MCHANDLE phandle, double timems)

uint32 MC_get_config_input_filter(MCHANDLE phandle, double *timems)

功 能：设置/读取通用输入（IN）信号滤波时间

参 数：phandle 链接句柄

timems 滤波时间（单位：250us）

返回值：错误码

uint32 MC_config_home_pin_filter(MCHANDLE phandle, double time250us)

uint32 MC_get_config_home_pin_filter(MCHANDLE phandle, double *time250us)

功 能：设置/读取原点（ORG）信号滤波时间

参 数：phandle 链接句柄

timems 滤波时间（单位：250us）

返回值：错误码

3.1.3 运动函数错误码说明

错误码	名称	含义
0	ERR_NOERR	成功
1	ERR_UNKNOWN	未知错误
2	ERR_PARAERR	参数错误
3	ERR_TIMEOUT	通讯超时，请检查控制器插槽是否正常；清理控制器金手指；如果还有类似情况出现请联系供应商
4	ERR_CONTROLLERBUSY	控制器状态忙
6	ERR_CONTILINE	连续插补出错，查看是否插补缓冲已经关闭或插补已经停止；
10	ERR_SENDERR	数据传输错误，请检查控制器插槽是否正常；清理控制器金手指；如果还有类似情况出现请联系供应商
20	ERR_FIRMWARE_INVALID_PARA	控制器返回参数错误
20	ERR_FIRMWARE_INVALID_PARA	在使用运动缓存功能时，缓存空间已满，无法继续添加指令，例如调用MC_fifo_pmove时返回错误。
22	ERR_FIRMWARE_STATE_ERR	控制器返回当前状态不允许操作
23	ERR_FIRMWARE_STATE_ERR	控制器返回当前状态不允许操作
23	ERR_FIRMWARE_STATE_ERR	调用插补功能指令报错代表没进入插补模式，最可能没调用MC_conti_open_list_muticoor打开插补缓冲区
24	ERR_FIRMWARE_PHANDLE_NOT_SUPPORT	控制器不支持的功能
25	ERR_FIRMWARE_PHANDLE_NOT_SUPPORT	控制器不支持的功能
10010	ERR_INVALID_PARA	控制器返回参数错误：圆弧插补，查看给定参数是否能绘制圆弧(起点到圆心长度和终点到圆心长度不一致、三点圆弧时是否在一条直线上)；轴数是否正确，是否超过当前控制器支持最大轴数；插补器个数，输入输出口个数等等。
10100	ERR_LIB_NOTSUPPORT	当前控制器不支持该命令。
10014	ERR_LIB_STATE_ERR	控制器状态错误。比如轴插补运动没有结束，又重新启动轴运动；比如当前在点位运动中，又重新启动轴运动；当前轴在非插补运动中，启动插补运动等等。

3.2 常见问题库

出现问题	解决建议
板卡插上后，PC 机系统还不能识别控制器	检查板卡驱动是否正确安装，在WINDOWS 的设备管理器（可参看WINDOWS 帮助文件）中查看驱动程序安装

出现问题	解决建议
	是否正常。如果发现有相关的黄色感叹号标志，说明安装不正确，需要按照软件部分安装指引，重新安装； 计算机主板兼容性差，请咨询主板供应商； PCI插槽是否完好； PCI 金手指是否有异物，可用酒精清洗。
PC 机不能和控制器通讯	PCI金手指是否有异物，可用酒精清洗； 参考软件手册检查应用软件是否编写正确。
板卡和驱动器电机连接后，发出脉冲时，电机不转动	板卡上的设置脉冲发送方式和驱动器的输入脉冲方式是否匹配，跳线J1－J8是否正确； 可以用 MotionMCX00A系列演示软件进行测试，观察脉冲计数等是否正常； 是否已经接上供给脉冲和方向的外部电源。
控制器已经正常工作，正常发出脉冲，但电机不转动	检查驱动器和电机之间的连接是否正确。可以使用MotionMCX00A系列 演示软件进行测试。 确保驱动器工作正常，没有出现报警。
电机可以转动，但工作不正常	检查控制器和驱动器是否正确接地，抗干扰措施是否做好； 脉冲和方向信号输出端光电隔离电路中使用的限流电阻过大，工作电流偏小。
能够控制电机，但电机出现振荡或是过冲	可能是驱动器参数设置不当，检查驱动器参数设置； 应用软件中加速、减速时间和运动速度设置不合理。
能够控制电机，但工作时，回原点定位不准	检查屏蔽线是否接地； 原点信号开关是否工作正常； 所有编码信号和原点信号是否受到干扰。
限位信号不起作用	限位传感器工作不正常； 限位传感器信号受干扰； 应用程序紊乱。
不能读入编码器信号	请检查编码器信号类型是否是脉冲TTL方波； 参看所选编码器说明书，检查接线是否正确； 编码器供电是否正常； 检查函数调用是否正确。
对编码器的读数不准确	检查全部编码器及触发源的接线； 做好信号线的接地屏蔽。
不能锁存编码器读数	检查触发源的接线； 检查函数的调用是否正确。
锁存数据的重复精度差	检查函数调用； 程序中是否进行了去抖动处理； 触发信号的设定。
数字输入信号不能读	接线是否正常；

出现问题	解决建议
取	检查函数调用。
数字输出信号不正常	接线是否正常； 检查函数调用。

4 总线使用

4.1 总线概述

恒昱控制 MC400A 和 MC800A 运动控制器可以控制 EtherCat 总线伺服驱动器，最大支持 16 轴，适合于多轴点位运动、插补运动、轨迹规划、手轮控制、编码器位置检测、 IO 控制、位置比较、位置锁存等功能的应用。 调试软件的使用及脉冲轴的性能和函数说明请参见前面的章节。

对总线可以理解为和驱动器的交互方式，由脉冲改为总线数据交互方式。

MC_ect_write_sdo

4.2 总线控制器特殊函数列表

	函数名	描述
	MC_ect_get_cycletime	读取总线的周期
	MC_ect_get_errcode	读取总线的错误码
	MC_ect_write_sdo	EtherCat SDO 设置操作
	MC_ect_read_sdo	EtherCat SDO 读取操作
	MC_ect_read_rxpdo	EtherCat RXPDO 读取操作
	MC_ect_read_txpdo	EtherCat TXPDO 读取操作
	MC_ect_write_rxpdo	EtherCat RXPDO 设置操作
	MC_ect_set_slave_state	EtherCat 设置从站的状态
	MC_ect_get_slave_state	EtherCat 读取从站的状态
	MC_ect_set_axistype	EtherCat 设置轴的状态
	MC_ect_get_axistype	EtherCat 读取轴的状态
	MC_ect_set_axismap	EtherCat 设置轴的映射
	MC_ect_get_axismap	EtherCat 读取轴的映射
	MC_ect_set_axismap_module	EtherCat 将指定从站的指定模块映射到指定轴
	MC_ect_get_axismap_module	EtherCat 读取从站的指定模块映射的轴号

	MC_ect_get_axis_address_module	EtherCat 读取轴对应的从站地址和模块号
	MC_ect_get_slave_module	EtherCat 读取所有从站模块个数
	MC_ect_set_simrun	EtherCat 设置主站为模拟运行
	MC_ect_get_maxslave	EtherCat 读取所有的链接从站个数
	MC_ect_get_axis_address	EtherCat 读取轴对应的从站地址
	MC_ect_get_slaveinfo	EtherCat 读取指定从站的信息
	MC_ect_set_master_state	EtherCat 设置主站的状态
	MC_ect_get_master_state	EtherCat 读取主站的状态
	MC_ect_set_axis_enable	EtherCat 使能电机轴
	MC_ect_get_axisenable	总线轴读取是否使能
	MC_ect_get_totalslave	EtherCat 读取所有从站个数
	MC_ect_set_home_profile	EtherCat 设置回原点参数
	MC_ect_get_home_profile	EtherCat 读取回原点参数
	MC_ect_home_move	EtherCat 启动总线轴回原点
	MC_ect_get_IfHoming	EtherCat 读取回零标志
	MC_ect_home_abort	总线轴退出原点运动
	MC_ect_get_axis_state_machine	读取总线轴状态
	MC_ect_set_axis_opmode	设置总线轴模式
	MC_ect_get_axis_opmode	读取总线轴模式
	MC_ect_set_axis_targettorque	设置轴的目标力矩值
	MC_ect_get_axis_targettorque	读取轴的目标力矩值
	MC_ect_get_axis_acttorque	读取轴的实际力矩值
	MC_ect_get_bus_state	读取总线错误
	MC_ect_get_bus_mismatch	EtherCat 读取从站丢包和

		从站描述不匹配的信息
	MC_ect_read_axis_errorcode	EtherCat 读取轴错误码
	MC_ect_clear_axis_errorcode	EtherCat 清除轴错误码
	MC_ect_load_config	下载 EtherCat IO 配置文件到控制器
	MC_ect_read_config	从控制器读取并生成 EtherCat 控制器 IO 配置文件
	MC_ect_get_device_ionum	读取 EtherCat 控制器 IO 数量
	MC_ect_set_iomap	EtherCat 控制器 IO 配置
	MC_ect_set_iomap_more	EtherCat 控制器 IO 配置
	MC_ect_get_axis_io_in	读取 EtherCAT 总线驱动器的数字量输入状态 32 bit 数据
	MC_ect_get_axis_io_inbit	读取 EtherCAT 总线驱动器的数字量单个输入状态
	MC_ect_set_axis_io_out	设置 EtherCAT 总线驱动器的数字量输出状态 32 bit 数据
	MC_ect_get_axis_io_out	读取 EtherCAT 总线驱动器的数字量输出状态 32 bit 数据
	MC_ect_set_axis_io_outbit	设置 EtherCAT 总线驱动器的数字量单个输出状态
	MC_ect_get_axis_io_outbit	读取 EtherCAT 总线驱动器的数字量单个输出状态

4.3 总线卡特殊函数说明

以下为总线卡特殊函数。

uint32 MC_ect_get_cycletime(MCHANDLE phandle, WORD portnum, uint32 *icycletime)

功 能：读取总线的周期

参 数：phandle 链接句柄
portnum 端口号(保留参数，默认写 2)
slaveid 从站号(1001 开始，1001 表示第一个从站)
icycletime 总线周期（单位：微秒）

返回值：错误码

uint32 MC_ect_get_errcode(MCHANDLE phandle, WORD portnum, uint32* errcode)

功 能：读取总线的错误码

参 数：phandle 链接句柄
portnum 端口号(保留参数，默认写 2)
Errcode 总线错误码

返回值：错误码

uint32 MC_ect_write_sdo(MCHANDLE phandle, WORD portnum, WORD slaveid, WORD index, WORD subindex, WORD bitlength, uint32 ivalue)

功 能：EtherCat SDO 设置操作

参 数：phandle 链接句柄
portnum 端口号(保留参数，默认写 2)
slaveid 从站号(1001 开始，1001 表示第一个从站)
index 对象字典的索引（按 10 进制数值传递，比如索引为 2000，应输入 0x2000 或 8192）
subindex 对象字典的子索引
bitlength 数据位宽（单位：bit）
ivalue 写入的数值

返回值：错误码

uint32 MC_ect_read_sdo(MCHANDLE phandle, WORD portnum, WORD slaveid, WORD index, WORD subindex, WORD bitlength, uint32 *ivalue)

功 能：EtherCat SDO 读取操作

参 数：phandle 链接句柄
portnum 端口号(保留参数，默认写 2)
slaveid 从站号(1001 开始，1001 表示第一个从站)
index 对象字典的索引（按 10 进制数值传递，比如索引为 2000，应输入 0x2000 或 8192）
subindex 对象字典的子索引
bitlength 数据位宽（单位：bit）

*ivalue 读取的数值
返回值：错误码

uint32 MC_ect_read_rxpdo(MCHANDLE phandle, WORD portnum, WORD slaveid, WORD index, WORD subindex, WORD varbitlength, uint32 *dwvalue)

功 能：EtherCat RXPDO 读取操作
参 数：phandle 链接句柄
portnum 端口号(保留参数，默认写 2)
slaveid 从站号(1001 开始，1001 表示第一个从站)
index 对象字典的索引（按 10 进制数值传递，比如索引为 2000，应输入 0x2000 或 8192）
subindex 该 RxPDO 的子索引
varbitlength 数据位宽（单位：bit）
*dwvalue 读取的数值
返回值：错误码

uint32 MC_ect_read_txpdo(MCHANDLE phandle, WORD portnum, WORD slaveid, WORD index, WORD subindex, WORD varbitlength, uint32 *dwvalue)

功 能：EtherCat TXPDO 读取操作
参 数：phandle 链接句柄
portnum 端口号(保留参数，默认写 2)
slaveid 从站号(1001 开始，1001 表示第一个从站)
index 对象字典的索引（按 10 进制数值传递，比如索引为 2000，应输入 0x2000 或 8192）
subindex 该 TxPDO 的子索引
varbitlength 数据位宽（单位：bit）
*dwvalue 读取的数值
返回值：错误码

uint32 MC_ect_write_rxpdo(MCHANDLE phandle, WORD portnum, WORD slaveid, WORD index, WORD subindex, WORD varbitlength, uint32 dwvalue)

功 能：EtherCat RXPDO 设置操作
参 数：phandle 链接句柄
portnum 端口号(保留参数，默认写 2)
slaveid 从站号(1001 开始，1001 表示第一个从站)
index 对象字典的索引（按 10 进制数值传递，比如索引为 2000，应输入 0x2000 或 8192）

	subindex	该 RxPDO 的子索引
	varbitlength	数据位宽（单位：bit）
	dwvalue	写入的数值

返回值：错误码

uint32 MC_ect_set_slave_state(MCHANDLE phandle, WORD portnum, WORD slaveid, WORD state, WORD timeout)

功 能：EtherCat 设置从站的状态

参 数：	phandle	链接句柄
	portnum	端口号(保留参数，默认写 2)
	slaveid	从站号(1001 开始，1001 表示第一个从站)
	state	从站的状态
	timeout	超时时间（单位：ms）

返回值：错误码

uint32 MC_ect_get_slave_state(MCHANDLE phandle, WORD portnum, WORD slaveid, WORD *state, WORD timeout)

功 能：EtherCat 读取从站的状态

参 数：	phandle	链接句柄
	portnum	端口号(保留参数，默认写 2)
	slaveid	从站号(1001 开始，1001 表示第一个从站)
	*state	从站的状态
	timeout	超时时间（单位：ms）

返回值：错误码

uint32 MC_ect_set_axistype(MCHANDLE phandle, WORD iaxis, WORD type)

功 能：EtherCat 设置轴的状态，可以变为虚轴或总线轴

参 数：	phandle	链接句柄
	iaxis	轴号（范围：0-15）
	type	轴的状态 1-脉冲轴 2-虚拟轴 3-总线轴

返回值：错误码

uint32 MC_ect_get_axistype(MCHANDLE phandle, WORD iaxis, WORD *type)

功 能：EtherCat 读取轴的状态

参 数：	phandle	链接句柄

	iaxis	轴号（范围：0-15）
	*type	轴的状态
		1-脉冲轴
		2-虚拟轴
		3-总线轴

返回值：错误码

uint32 MC_ect_set_axismap(MCHANDLE phandle, WORD portnum, WORD slaveid, WORD iaxis)

功　能：EtherCat 设置轴的映射

参　数：	phandle	链接句柄
	portnum	端口号(保留参数，默认写 2)
	slaveid	从站号(1001 开始，1001 表示第一个从站)
	iaxis	轴号（范围：0-63）

返回值：错误码

uint32 MC_ect_get_axismap(MCHANDLE phandle, WORD portnum, WORD slaveid, WORD *iaxis)

功　能：EtherCat 读取轴的映射

参　数：	phandle	链接句柄
	portnum	端口号(保留参数，默认写 2)
	slaveid	从站号(1001 开始，1001 表示第一个从站)
	*iaxis	轴号（范围：0-15）

返回值：错误码

uint32 MC_ect_set_axismap_module(MCHANDLE phandle, WORD portnum, WORD slaveid, WORD module, WORD iaxis)

功　能：EtherCat 将指定从站的指定模块映射到指定轴

参　数：	phandle	链接句柄
	portnum	端口号(保留参数，默认写 2)
	slaveid	从站号(1001 开始，1001 表示第一个从站)
	module	指定模块（从 0 开始）
	iaxis	轴号（范围：0-63）

返回值：错误码

uint32 MC_ect_get_axismap_module(MCHANDLE phandle, WORD portnum, WORD slaveid, WORD module, WORD *iaxis)

功　能：EtherCat 读取从站的指定模块映射的轴号

参　数：	phandle	链接句柄
	portnum	端口号(保留参数，默认写 2)

slaveid 从站号(1001 开始，1001 表示第一个从站)
module 指定模块（从 0 开始）
iaxis 轴号（范围：0-63）
返回值：错误码

uint32 MC_ect_get_axis_address_module(MCHANDLE phandle, WORD iaxis, WORD *slaveaddr, WORD *slaveaddr_1, WORD *module)

功 能：EtherCat 读取轴对应的从站地址和模块号
参 数：phandle 链接句柄
iaxis 轴号（范围：0-63）
slaveddr 从站号(从 1001 开始)
slaveddr_1 从站地址(65536 开始，自增地址)
module 模块号
返回值：错误码

uint32 MC_ect_get_slave_module(MCHANDLE phandle, WORD portnum, WORD slaveid, WORD *num)

功 能：EtherCat 读取所有从站模块个数
参 数：phandle 链接句柄
portnum 端口号(保留参数，默认写 2)
slaveid 从站号(1001 开始，1001 表示第一个从站)
num 模块个数
返回值：错误码

uint32 MC_ect_set_simrun(MCHANDLE phandle, WORD portnum, WORD simrun)

功 能：EtherCat 设置主站为模拟运行
参 数：phandle 链接句柄
portnum 端口号(保留参数，默认写 2)
simrun 0-关闭模拟运行
1-启动模拟运行（用于没有伺服电机时进行总线调试）
返回值：错误码

uint32 MC_ect_get_maxslave(MCHANDLE phandle, WORD portnum, WORD *num)

功 能：EtherCat 读取所有的链接从站个数
参 数：phandle 链接句柄
portnum 端口号(保留参数，默认写 2)
*num EtherCat 链接的从站个数
返回值：错误码

uint32 MC_ect_get_axis_address(MCHANDLE phandle, WORD iaxis, WORD *slaveaddr, WORD *slaveaddr_1)

功 能：EtherCat 读取轴对应的从站地址

参 数：phandle 链接句柄
iaxis 轴号
slaveaddr 从站地址(1001 开始)
slaveaddr_1 从站地址(65536 开始,自增地址)

返回值：错误码

uint32 MC_ect_get_slaveinfo(MCHANDLE phandle, WORD portnum, WORD slaveid, uint32 *info)

功 能：EtherCat 读取指定从站的信息

参 数：phandle 链接句柄
portnum 端口号(保留参数，默认写 2))
slaveid 从站号(1001 开始，1001 表示第一个从站)
info 0- EC_DEVICE_TYPE
DEVICE_TYPE_IN = 0 输入信号
DEVICE_TYPE_OUT=1 输出信号
DEVICE_TYPE_INOUT=2 输入输出信号
DEVICE_TYPE_AIN=3 模拟量输入信号
DEVICE_TYPE_AOUT=4 模拟量输出信号
DEVICE_TYPE_AINOUT=5 模拟量输入输出信号
DEVICE_TYPE_MOTOR=6 电机信息

1- FIXID
2-Vendor 供应商
3-Product Code 产品标识号

返回值：错误码

uint32 MC_ect_set_master_state(MCHANDLE phandle, WORD portnum, WORD state, WORD timeout)

功 能：EtherCat 设置主站的状态

参 数：phandle 链接句柄
portnum 端口号(保留参数)
state 主站的状态：UNKNOWN=0
INIT=1
PREOP=2
SAFEOP=4
OP=8
BOOTSTRAP=3
BCppDummy=0xFFF

timeout 超时时间（单位：ms）

返回值：错误码

uint32 MC_ect_get_master_state(MCHANDLE phandle, WORD portnum, WORD *state, WORD timeout)

功 能：EtherCat 读取主站的状态

参 数： phandle 链接句柄

portnum 端口号(保留参数)

*state 主站的状态：UNKNOWN=0
INIT=1
PREOP=2
SAFEOP=4
OP=8
BOOTSTRAP=3
BCppDummy=0xFFF

timeout 超时时间（单位：ms）

返回值：错误码

uint32 MC_ect_set_axis_enable(MCHANDLE phandle, WORD iaxis, WORD ifenable)

功 能：EtherCat 使能电机轴

参 数： phandle 链接句柄

iaxis 轴号

ifenable 1-电机使能，0 关闭使能

返回值：错误码

WORD MC_ect_get_axisenable(MCHANDLE phandle, WORD iaxis)

功 能：总线轴读取是否使能

参 数： phandle 链接句柄

iaxis 轴号

返回值： 总线轴使能状态（1-电机使能，0 关闭使能）

uint32 MC_ect_get_totalslave(MCHANDLE phandle, WORD portnum, WORD *num)

功 能：EtherCat 读取所有从站个数

参 数： phandle 链接句柄

portnum 端口号(保留参数)

num 返回从站个数

返回值：错误码

uint32 MC_ect_set_home_profile(MCHANDLE phandle, WORD iaxis, WORD homemode, double Highvel, double low_vel, double tacc, double tdec, double offsetpos)

功 能：EtherCat 设置回原点参数，会将电机的模式强制设置为 6，回原点模式

参 数：

phandle	链接句柄
iaxis	轴号
homemode	回原点模式
Highvel	回原点高速（单位：脉冲/秒）
low_vel	回原点低速（单位：脉冲/秒）
tacc	加速时间（单位：秒）
tdec	减速时间（单位：秒）
offsetpos	原点偏移(保留参数，请设置为 0)

返回值：错误码

注意：每家驱动器回零对所支持的回零模式的说明各不相同，详情请查看各驱动器手册对于回零运动的说明。

uint32 MC_ect_get_home_profile(MCHANDLE phandle, WORD iaxis, WORD *homemode, double *Highvel, double *low_vel, double *tacc, double *tdec, double *offsetpos)

功 能：EtherCat 读取回原点参数

参 数：

phandle	链接句柄
iaxis	轴号
homemode	回原点模式
Highvel	回原点高速（单位：脉冲/秒）
low_vel	回原点低速（单位：脉冲/秒）
tacc	加速时间（单位：秒）
tdec	减速时间（单位：秒）
offsetpos	原点偏移(保留参数)

返回值：错误码

uint32 MC_ect_home_move(MCHANDLE phandle, WORD iaxis)

功 能：EtherCat 启动总线轴回原点 ，强制设置为 6 模式，

参 数：

phandle	链接句柄
iaxis	轴号

返回值：错误码

注意：回原点过程中，要退出回原点运动，可以使用 MC_ect_home_abort；请使用 MC_ect_get_axis_state_machine 来检测回原点运动是否完成。回原点运动完成后，延时短暂时间，确保电机停稳后，再清除编码器和当前位置。然后再将伺服模式切换回 CSP 模式。

uint32 MC_ect_get_IfHoming(MCHANDLE phandle, WORD iaxis)

功 能：EtherCat 读取回零标志

参 数： phandle 链接句柄
iaxis 轴号

返回值： 0—回零未完成 1—回零完成 2—回零错误

注意：控制器通过读取对象字典 6041（状态字）的值，判断第 10 位为 1，第 12 位为 1，第 13 位为 0 则认为回零成功。

uint32 MC_ect_home_abort(MCHANDLE phandle, WORD iaxis)

功 能：总线轴退出原点运动

参 数： phandle 链接句柄
iaxis 轴号

返回值：错误码

uint32 MC_ect_get_axis_state_machine(MCHANDLE phandle, WORD iaxis, WORD *statemachine)

功 能： 读取总线轴状态，输出值为 TxPDO，对象字典 6041（状态字）的值。

参 数： phandle 链接句柄
iaxis 轴号
statemachine 返回总线轴的状态机

位序号	名称		描述
第 0 位	伺服准备好	READ TO SWITCH ON	1-有效，0-无效
第 1 位	可以开启伺服运行	SWITCH ON	1-有效，0-无效
第 2 位	伺服运行	OPERATION ENABLED	1-有效，0-无效
第 3 位	故障	FAULT	1-有效，0-无效
第 4 位	主回路上电	VOLTAGE ENABLED	1-有效，0-无效
第 5 位	快速停机	QUICK STOP	0-有效，1-无效
第 6 位	伺服不可运行	SWITCH ON DISABLED	1-有效，0-无效
第 7 位	警告	WARNING	1-有效，0-无效
第 8 位	保留位		
第 9 位	远程控制	REMOTE	1-有效，0-无效
第 10 位	目标到达	TAGET REACH	1-有效，0-无效
第 11 位	内部限制有效	INTERNAL LIMIT ACTIVE	1-有效，0-无效
第 12 位	保留位		
第 13 位	保留位		
第 14 位	保留位		
第 15 位	保留位		

返回值：错误码

注意：使用伺服自带回原点，PP,PV 等模式时，需要检测第 10 位。第 3 位为 1 时，表示伺服已经出错。

uint32 MC_ect_set_axis_opmode(MCHANDLE phandle, WORD portnum, WORD iaxis, WORD mode)

uint32 MC_ect_get_axis_opmode(MCHANDLE phandle, WORD portnum, WORD iaxis,WORD *mode)

功 能：设置/读取总线轴模式

参 数：

phandle	链接句柄
portnum	端口号(保留参数)
iaxis	轴号
mode	支持以下两种模式 6 -回原点模式 8-CSP 模式 10-力矩模式

返回值：错误码

uint32 MC_ect_set_axis_targettorque(MCHANDLE phandle, WORD portnum, WORD iaxis, long torque)

uint32 MC_ect_get_axis_targettorque(MCHANDLE phandle, WORD portnum,WORD iaxis, long *torque)

功 能：设置/读取轴的目标力矩值

参 数：

phandle	链接句柄
portnum	端口号(保留参数)
iaxis	轴号
torque	力矩值

返回值：错误码

uint32 MC_ect_get_axis_acttorque(MCHANDLE phandle, WORD portnum, WORD iaxis, long *acttorque)

功 能：读取轴的实际力矩值

参 数：

phandle	链接句柄
portnum	端口号(保留参数)
iaxis	轴号
torque	力矩值

返回值：错误码

uint32 MC_ect_get_bus_state(MCHANDLE phandle, WORD portnum, uint32 *state)

功 能：读取总线错误

参 数：

phandle	链接句柄
portnum	端口号(保留参数)

state	总线状态（返回非 0 值说明总线无法建立） 65537-循环命令：工作计数器错误 65538-主站初始化命令：工作计数器错误 65539-从站初始化命令：工作计数器错误 65543 - EoE mbox 发送：工作计数器错误 65544 - CoE mbox 发送：工作计数器错误 65545 - FoE mbox 发送：工作计数器错误 65546 - 在已发送的以太网帧上没有响应 65547 - 在从服务器发送的 ecat init 命令上未得到响应或意外响应 65548 - 未收到发送的 ecat master init 命令的响应 65550 - 等待 mailbox init 命令响应时超时 65551 - 接收循环帧时，并非所有从站都处于工作状态 65552 - 以太网链接（电缆）未连接 65554 - 冗余：检测到断线 65555 - 接收循环帧时，至少有一个从站处于错误状态（BRD AL-STATUS） 65556 – 从站错误 (AL status code) 65557 – 站点地址丢失（或从站丢失）-FPRD 至 AL_ 状态失败 65559 - SoE mbox 发送：工作计数器错误 65560 - SoE mbox 写入响应错误 65561 - CoE mbox SDO 异常终止 65562 - 客户端注册已删除，可能由其他线程（RAS）调用 ecatConfigureMaster 65563 - 冗余：线路已修复 65564 - FoE mbox 终止 65565 - 接收到无效的邮箱数据

返回值：错误码

uint32 MC_ect_get_bus_mismatch(MCHANDLE phandle, WORD portnum, uint32 *info)

功 能：EtherCat 读取从站丢包和从站描述不匹配的信息

参 数：	phandle	链接句柄
	portnum	端口号(保留参数)
	info	info[0]=是否为第一个从站 info[1]=从站地址 info[2]=从站自增地址 info[3]=从站端口

返回值：错误码

注意：info[0]和 info[1]同时返回 0 则表示从站通讯正常

uint32 MC_ect_read_axis_errorcode(MCHANDLE phandle, WORD portnum, WORD iaxis, uint32 *code)

功 能：EtherCat 读取轴错误码实际操作 603F 对象字典

参 数： phandle 链接句柄
portnum 端口号(保留参数)
iaxis 轴号（范围：0-31）
code 轴错误码（对应 603F 对象字典）

返回值：错误码

uint32 MC_ect_clear_axis_errorcode(MCHANDLE phandle, WORD portnum, WORD iaxis)

功 能：EtherCat 清除轴错误码

参 数： phandle 链接句柄
portnum 端口号(保留参数)
iaxis 轴号（范围：0-31）

返回值：错误码

int MC_ect_load_config(MCHANDLE phandle, char* fpath)

功 能：下载 EtherCat IO 配置文件到控制器

参 数： phandle 链接句柄
fpath 文件路径

返回值：错误码

int MC_ect_read_config(MCHANDLE phandle, char* fpath)

功 能：从控制器读取并生成 EtherCat 控制器 IO 配置文件

参 数： phandle 链接句柄
fpath 文件路径

返回值：错误码

uint32 MC_ect_get_device_ionum(MCHANDLE phandle, WORD type, WORD *usIoNum)

功 能：读取 EtherCat 控制器 IO 数量 。如果使用接线盒或者接线板，该函数返回总线拓扑(包括标配接线端子板)的 IO 数量

参 数： phandle 链接句柄
type 类型 0-IN 1- OUT 2-AD 3-DA
usIoNum 当前类型个数

返回值：错误码

uint32 MC_ect_set_iomap(MCHANDLE phandle, WORD iobit, WORD map

type, WORD usMapPdoSlaveId, WORD usMapPdoSlaveType, WORD usMapPdoIndex, WORD usMapPdoSubIndex, WORD usMapPdoStartBit)

功 能：EtherCat 控制器 IO 配置 可以和 MC_set_axis_io_map 配合使用

参 数：

phandle	链接句柄
iobit	IO 口或 AD 或 DA 的通道号
maptype	映射类型 0-IN 1- OUT 2-AD 3-DA
usMapPdoSlaveId	映射从站 ID 号 1001 开始
usMapPdoSlaveType	映射从站类型 保留参数
usMapPdoIndex	Pdo 映射索引
usMapPdoSubIndex	Pdo 映射子索引
usMapPdoStartBit	Pdo 起始位号

返回值：错误码

uint32 MC_ect_set_iomap_more(MCHANDLE phandle, WORD iobit, WORD bitsize, WORD maptype, WORD usMapPdoSlaveId, WORD usMapPdoSlaveType, WORD usMapPdoIndex, WORD usMapPdoSubIndex, WORD usMapPdoStartBit)

功 能：EtherCat 控制器 IO 配置 可以和 MC_set_axis_io_map 配合使用

参 数：

phandle	链接句柄
iobit	IO 口或 AD 或 DA 的通道号
bitsize	映射位个数
maptype	映射类型 0-IN 1- OUT 2-AD 3-DA
usMapPdoSlaveId	映射从站 ID 号 1001 开始
usMapPdoSlaveType	映射从站类型 保留参数
usMapPdoIndex	Pdo 映射索引
usMapPdoSubIndex	Pdo 映射子索引
usMapPdoStartBit	Pdo 起始位号

返回值：错误码

uint32 MC_ect_get_axis_io_in(MCHANDLE phandle, WORD iaxis)

功 能：读取 EtherCAT 总线驱动器的数字量输入状态 32 bit 数据

参 数：

phandle	链接句柄
iaxis	EtherCAT 总线轴轴号

返回值：EtherCAT 总线驱动器的数字量输入状态 32 bit 数据

uint32 MC_ect_get_axis_io_inbit(MCHANDLE phandle, WORD iaxis, WORD bit)

功 能：读取 EtherCAT 总线驱动器的单个数字量输入状态

参 数： phandle 链接句柄

iaxis EtherCAT 总线轴轴号
bit 输入口对应位号
返回值：EtherCAT 总线驱动器的数字量指定输入口状态

uint32 MC_ect_set_axis_io_out(MCHANDLE phandle, WORD iaxis, uint32 iostate)

功 能：设置 EtherCAT 总线驱动器的数字量输出状态 32 bit 数据
参 数： phandle 链接句柄
iaxis EtherCAT 总线轴轴号
iostate 总线驱动器的数字量输出状态 32 bit 数据
返回值：错误码

uint32 MC_ect_get_axis_io_out(MCHANDLE phandle, WORD iaxis)

功 能：读取 EtherCAT 总线驱动器的数字量输出状态 32 bit 数据
参 数： phandle 链接句柄
iaxis EtherCAT 总线轴轴号
返回值：EtherCAT 总线驱动器的数字量输出状态 32 bit 数据

uint32 MC_ect_set_axis_io_outbit(MCHANDLE phandle, WORD iaxis, WORD bit, WORD state)

功 能：设置 EtherCAT 总线驱动器的单个数字量输出状态
参 数： phandle 链接句柄
iaxis EtherCAT 总线轴轴号
bit 输出口对应位号
state 输出口状态（0 或 1）
返回值：错误码

uint32 MC_ect_get_axis_io_outbit(MCHANDLE phandle, WORD iaxis, WORD bit)

功 能：读取 EtherCAT 总线驱动器的单个数字量输出状态
参 数： phandle 链接句柄
iaxis EtherCAT 总线轴轴号
bit 输出口对应位号
返回值：指定输出口状态（0 或 1）

4.4 总线卡 IO 模块使用说明

1. 使用本地 IO 接线板时，本地 IO 数量为 8 个输出（范围：OUT0-OUT7），8 个输入（范围：IN0-IN7），总线 IO 排序向后延伸，即输出口从 OUT8 开始，输入口从 IN8 开始。

2. 调用 MC_ect_get_device_ionum 函数获取总线 IO 口数量，最后直接调用 MC_read_inbit，MC_write_outbit，MC_read_outbit 等函数可操作或读取单个 IO，调用 MC_write_outport_ex，MC_read_outport_ex，MC_read_inport 等函数可一次性操作或读取 32 个 IO。例程:

```
WORD EctOutNum = 0;
int iret = MC_ect_get_device_ionum(0, 1, &EctOutNum);
if(iret == 0)
{
    //EctOutNum 是总线 IO 模块输出口的数量
}
WORD EctInNum = 0;
iret = MC_ect_get_device_ionum(0, 0, &EctInNum);
if(iret == 0)
{
    // EctInNum 是总线 IO 模块输入口的数量
}
```

4.5 总线卡函数错误码说明

错误码	名称	含义
100001	FEATURE NOT SUPPORTED	函数或属性不可用
100002	INVALID INDEX	无效索引
100003	INVALID OFFSET	无效偏移量
100004	CANCEL	取消
100005	INVALID SIZE	无效大小
100006	INVALID DATA	无效数据
100007	NOT READY	未准备好
100008	BUSY	堆栈当前正忙
100009	CANNOT QUEUE ACYCLIC ECAT COMMAND	无法对非循环 ECAT 命令进行排队
100010	NO MEMORY LEFT	可用的可分配内存不足
100011	INVALID PARAMETER	无效参数
100012	NOT FOUND	无法找到
100013	DUPLICATED FIXED ADDRESS DETECTED	检测到重复的固定地址
100014	INVALID STATE	无效状态
100015	CANNOT ADD SLAVE TO TIMER LIST	无法将从属添加到计时器列表
100016	TIMEOUT	超时
100017	OPEN FAILED	打开失败

错误码	名称	含义
100018	SEND FAILED	发送失败
100019	INSERT MAILBOX ERROR	邮箱命令无法存储到内部命令
100020	INVALID COMMAND	未知邮箱命令代码
100021	UNKNOWN MAILBOX PROTOCOL COMMAND	未知的邮箱协议或邮箱命令，未知的协议关联
100022	ACCESS DENIED	拒绝访问
100023	IDENTIFICATION FAILED	标识命令失败
100024	CREATE LOCK FAILED	创建锁失败
100026	PRODUCT KEY INVALID	产品密钥无效
100027	WRONG CONFIGURATION FORMAT	配置格式错误
100028	FEATURE DISABLED	功能已禁用
100030	BUS CONFIG MISMATCH	总线配置不匹配
100031	ERROR READING CONFIG FILE	读取配置文件时出错
100033	CYCLIC COMMANDS ARE MISSING	缺少循环命令
100034	AL_STATUS REGISTER READ MISSING IN XML FILE FOR AT LEAST ONE STATE	读取状态寄存器丢失，至少一个状态的 XML 文件
100035	FATAL INTERNAL MCSM	主控制状态机处于未定义状态
100036	SLAVE ERROR	从站出错
100037	FRAME LOST, IDX MISMATCH	帧丢失，IDX 不匹配
100038	AT LEAST ONE ETHERCAT COMMAND IS MISSING IN THE RECEIVED FRAME	至少有一个 ETHERCAT 命令在接收到的帧中丢失
100039	CYCLIC COMMAND WKC ERROR	总线错误 查看网线连接是否正常
100040	IOCTL EC_IOCTL_DC_LATCH_REQ_LTIMV ALS INVALID IN DCL AUTO READ MODE	如果直流闭锁正在运行，则不能使用此功能“自动读取”模式
100041	AUTO INCREMENT ADDRESS INCREMENT MISMATCH	自动递增地址递增不匹配
100042	SLAVE IN INVALID STATE, E.G. NOT IN OP (API NOT CALLABLE IN THIS STATE)	从机处于无效状态，例如不在 OP 中（API 在此状态下不可调用）
100043	STATION ADDRESS LOST OR SLAVE MISSING - FPRD TO AL_STATUS FAILED	状态地址丢失或找不到从站
100044	TOO MANY CYCLIC COMMANDS IN XML CONFIGURATION FILE XML	XML 文件配置错误
100045	ETHERNET LINK CABLE DISCONNECTED	网线掉线
100046	MASTER CORE NOT ACCESSIBLE	主站不可访问
100047	MAILBOX SEND: WORKING COUNTER	网线掉线
100048	MAILBOX RECEIVE: WORKING COUNTER	网线掉线
100049	NO MAILBOX SUPPORT	没有邮箱支持
100051	0036 协议错误	
100055	超过从站限制	

错误码	名称	含义
100057	激活码错误	
100064-100094	SDO 操作失败 请检查对象字典设置	
100379	LINE CROSS	从站之间存在有网线连接交叉，接错的情况

4.6 总线卡常用 PDO 对象表

主索引（HEX）	子索引（HEX）	对象名称	数据类型	PDO 映射
603F	00	错误码（Error Code）	UINT16	TxPDO
6040	00	控制字（Controlword）	UINT16	RxPDO
6041	00	状态字（Statusword）	UINT16	TxPDO
6060	00	操作模式（Modes of oprration）	INT8	TxPDO
6061	00	操作模式显示（modes of operation display）	INT8	RxPDO
6064	00	实际位置（position actual vlaue）	INT32	TxPDO
606C	00	速度实际值（Velocity actual value）	INT32	TxPDO
607A	00	目标位置（Target position）	INT32	RxPDO
60FD	00	数字输入（Digital inputs）	UINT32	TxPDO
60FE	01	物理输出开启（Physical outputs）	UINT32	RxPDO
60FE	02	物理输出使能（Bit mask）	UINT32	RxPDO
607C	00	原点偏移（Home offset）	INT32	NO
6098	00	回零模式（Homing method）	INT8	NO
6099	01	回零高速（Speed during search for switch）	UINT32	NO
6099	02	回零低速（Speed during search for zero）	UINT32	NO
609A	00	回零加速度（Homing acceleration）	UINT32	NO
6092	01	每转脉冲数（Feed）	UINT32	NO
6071	00	目标转矩（Target torque）	INT16	RxPDO
6072	00	最大转矩（Max torque）	UINT16	RxPDO

6077	00	转矩实际值（Torque actual value）	INT16	TxPDO
6080	00	最大电机速度（Max motor speed）	UINT32	RxPDO
6087	00	转矩变化率（Torque slope）	UINT32	RxPDO
60B2	00	转矩偏移值（Torque offset）	INT16	RxPDO
60E0	00	正向转矩限制	UINT16	RxPDO
60E1	00	负向转矩限制	UINT16	RxPDO

4.7 总线卡例程说明

4.7.1 轴设置 CSP 模式

例程：

```
WORD OPMODE = 0;
//在设置 CSP 模式之前最好先读一次当前模式。
//如果当前是 CSP 模式时，则不需要再次设置，否则可能会引起一些异常。
//
MC_ect_get_axis_opmode(g_handle,2,0,&OPMODE);
if(OPMODE != 8)
{
    //把当前轴设置为 CSP 模式
     MC_ect_set_axis_opmode(g_handle,2,0,8);
}
```

4.7.2 点位运动

例程：

```
WORD OPMODE = 0;
//设置轴运动参数
 MC_set_profile_ex(g_handle,0,0,5000,0,0.01, 0.01);
MC_ect_get_axis_opmode(g_handle,2,0,&OPMODE);
 if(OPMODE != 8)
 {
    //把当前轴设置为 CSP 模式
    MC_ect_set_axis_opmode(g_handle,2,0,8);
 }
```

```
//获取目标位置
int dis = GetDlgItemInt(IDC_EDIT_AIMPOS_ECT);
int mode = 0;
//相对位置模式下
if(m_combo_movemode_ect.GetCurSel()==0)
{
    //正方向运动
    if(m_combo_dir_ect.GetCurSel()==0)
    {
        //运行点位运动
        MC_pmove(g_handle,0,dis,mode);
    }
     else
    {
        //运行点位运动
        dis = dis*(-1);
        MC_pmove(g_handle,0,dis,mode);
    }

}
else
{
   //运行点位运动
    mode = 1;
    MC_pmove(g_handle,0,dis,mode);
}
```

4.7.3 连续运动

例程：

```
WORD OPMODE = 0;
//设置轴运动参数
 MC_set_profile_ex(g_handle,0,0,5000,0,0.01, 0.01);
MC_ect_get_axis_opmode(g_handle,2,0,&OPMODE);
 if(OPMODE != 8)
 {
```

```
        //把当前轴设置为 CSP 模式
         MC_ect_set_axis_opmode(g_handle,2,0,8);
    }
    int dir = 0;
    //正方向运动
    if(m_combo_dir_ect.GetCurSel()==0)
    {
        dir = 1;
        iret = MC_vmove(g_handle,0,dir,5000);
    }
    else
    {
        iret = MC_vmove(g_handle,0,dir,5000);
    }
```

4.7.4 总线回零运动

例程：

```
    WORD OPMODE = 0;
    //设置轴运动参数
     MC_set_profile_ex(g_handle,0,0,5000,0,0.01, 0.01);
    MC_ect_get_axis_opmode(g_handle,2,0,&OPMODE);
   if(OPMODE != 6)
    {
        //把当前轴设置为 HOME 模式
        MC_ect_set_axis_opmode(g_handle,2,0,6);
    }
    //设置总线回零模式，回零模式说明参考伺服说明书
    MC_ect_set_home_profile(g_handle,0,mode,vm,vs,ts,te,0);
     MC_ect_home_move(g_handle,0);
```

4.7.5 总线回零运动停止

例程：

```
    WORD OPMODE = 0;
```

```
MC_ect_get_axis_opmode(g_handle,2,0,&OPMODE);
if(OPMODE == 6)
{
    总线回零运动停止
    MC_ect_home_abort(g_handle,0);
}
//指定轴立即停止
MC_imd_stop(g_handle,0);
```

4.7.6 总线回零是否完成

例程：

```
//读取轴总线回零是否完成
int iret = MC_ect_get_IfHoming(g_handle,0);
if(iret == 0)
{
    //总线回零未完成
}
else if(iret == 1)
{
    //总线回零完成
}
else if(iret == 2)
{
    //总线回零错误
}
```

4.7.7 位置清零

例程：

```
WORD OPMODE = 0;
MC_ect_get_axis_opmode(g_handle,2,0,&OPMODE);
if(OPMODE != 8)
{
    //把当前轴设置为 CSP 模式
```

```
        MC_ect_set_axis_opmode(g_handle,2,0,8);
    }
    //回零运动完成后，需要短暂延时一段时间等待电机停止
    //再清零当前轴的指令位置和编码器位置
    MC_set_position(g_handle,m_gAxisIndex,0);
     MC_set_encoder(g_handle,m_gAxisIndex,0);
```

4.7.8 总线伺服使能

例程：

```
//打开伺服使能
MC_ect_set_axis_enable(g_handle,0,1);
//关闭伺服使能
MC_ect_set_axis_enable(g_handle,0,0);
int iret = MC_ect_get_axisenable(g_handle,0);
if(iret == 1)
{
    //伺服使能打开
}
 else
{
    //伺服使能关闭
}
```

4.7.9 伺服报警清除

例程：

```
MC_ect_clear_axis_errorcode(g_handle,2,0)
```

4.7.10 读写 SDO

例程：

```
//对 1001 号站的对象字典 0x6040，子索引为 0，位宽 32 位写入 SDO
MC_ect_write_sdo(g_handle,2,1001,0x6040,0,32,1);
```

```
//对 1001 号站的对象字典 0x6040，子索引为 0，位宽 32 位读取 SDO
uint32 ireadval = 0;
MC_ect_read_sdo(g_handle,2,1001,0x6040,0,32, &ireadval);
```

4.7.11 读写 rxpdo

例程：

```
//对 1001 号站的对象字典 0x6040，子索引为 0，位宽 32 位写入 rxpdo
MC_ect_write_rxpdo(g_handle,2,1001,0x6040,0,32,1);

//对 1001 号站的对象字典 0x6040，子索引为 0，位宽 32 位读取 rxpdo
uint32 ireadval = 0;
MC_ect_read_rxpdo(g_handle,2,1001,0x6040,0,32, &ireadval);
```

4.7.12 读取 txpdo

例程：

```
//对 1001 号站的对象字典 0x6041，子索引为 0，位宽 32 位读取 txpdo
uint32 ireadval = 0;
MC_ect_read_txpdo(g_handle,2,1001,0x6041,0,32, &ireadval);
```

4.7.13 读取总线状态

例程：

```
//读取总线主站状态
WORD num = 0;
 MC_ect_get_master_state(g_handle,2,&num,2000);
```

4.7.14 读取总线错误码

例程：

```
//读取总线错误,非零说明总线无法建立
uint32 idata = 0;
 MC_ect_get_bus_state(g_handle,2,&idata);
```

4.7.15 读取从站通讯状态

例程：

```
//读取总线错误,非零说明总线无法建立
WORD slaveId = 0;
 uint32 info[10] = {0};
 MC_ect_get_bus_mismatch(g_handle, 2, info);
  if(info[0] == 1)
  {
      //第一个从站不匹配，链接正常
      slaveId = 1001;
  }
  else
  {
      if(info[1] != 0)
      {
          slaveId = info[1] - 1000;
          //从第 slaveId 个站以后不正常，端口为 info[3]，请检查网线连接或从站工作是否正常。
      }
  }
```