leetro

MPC2860 运动控制器

软

件

手

册

（1.3 版）

乐创自动化技术有限公司
LEETRO AUTOMATION CO.,LTD.

版权申明

前言

感谢购买 MPC2860 运动控制器！MPC2860 是本公司研制的一款高性能通用控制器。本手册介绍了关于 MPC2860 的规格、使用方法，使用前请充分理解 MPC2860 的使用功能。

安全警告

注意以下警告，以免伤害操作人员及其他人员，防止机器损坏。

◆ 下面的“危险”和“警告”符号是按照其事故危险的程度来标出的。

 危 险	指示一个潜在的危险情况，如果不避免，将导致死亡或严重伤害。
 警 告	指示一个潜在的危险情况，如果不避免，将导致轻度或中度伤害，或物质损坏。

◆ 下列符号指示哪些是禁止的，或哪些是必须遵守的。

	这个符号表示禁止操作。
	这个符号表示须注意的操作。

常规安全概要

请查看下列安全防范措施以避免受伤害并防止对本产品或任何与其相连接的产品造成损伤。为避免潜在的危险，请按详细说明来使用本产品。

使用正确的电源线。请使用满足国家标准的电源线。

正确地连接和断开。先将控制卡输出连接至转接板，再将电机、驱动器连接到转接板，最后开启电源。断开时先关闭外部电源，再断开电机、驱动器与转接板的连接，最后断开控制卡与转接板的连接。

当有可疑的故障时不要进行操作。如果您怀疑本产品有损伤，请让有资格的服务人员进行检查。

不要在的湿的/潮湿环境下操作。

不要在爆炸性的空气中操作。

保持产品表面清洁和干燥。

防止静电损伤。静电释放（ESD）可能会对运动控制器及其附件中的元件造成损伤。为了防止 ESD，请小心处理控制器元件，不要触摸控制器上元器件。不要将控制器放置在可能产生静电的表面。在防护静电的袋子或容器内运输和储存控制器。

关于保修

保修时间

在指定的地点购买的产品的保修期为 1 年。

保修范围

如果在上述质保期内由于本公司责任发生了故障，本公司提供无偿修理。

以下范围不在保修范围内：

- 对于说明书及其它手册记录的不适当环境或不适当使用引起的故障。
- 用户的装置、控制软件等引起本产品意外故障。
- 由客户对本产品的改造引起的故障。
- 火灾、地震及其它自然灾害等外部主要原因引起的故障。

产品的应用范围

本产品设计制造用于普通工业应用，超出预料的用途并对人的生命或财产造成重大的影响不在产品服务范围。

联系信息

通信地址：四川省成都市高新区冯家湾科园南二路 1 号大一孵化园 8 幢 B 座（610041）

公司网站：htttp://www.leetro.com

技术支持：

- Tel：（028）85149977
- FAX：（028）85187774

目录

系统支持

操作系统支持

- Windows xp
- Windows 7 32bit/64bit

开发环境支持

- VC++
- VB
- LabVIEW
- C#
- Delphi

开发文件获取

获取步骤：

①. 访问成都乐创官网 www.leetro.com，获取最新的驱动程序安装包并安装。

②. 在安装目录下（默认是 C:\Program Files）找到名为 MPC2860 文件夹并打开，子文件夹“Develop”包含开发所需要的文件。

更新升级

1. 卸载掉以前安装的驱动程序，方法有以下两种：
   - ✧ 打开“控制面板”->“添加或删除程序”，选择 MPC2860 对应项，点击“更改/删除”按钮，按照提示一步一步完成对驱动程序的卸载。
   - ✧ 在安装目录中（默认是 C:\Program Files）中找到 MPC2860 同名文件夹，打开该目录，找到名为“UNWISE.EXE”文件，双击并运行，按照提示一步一步完成对驱动程序的卸载。
2. 查找到最新驱动程序安装包并下载安装，找到开发文件，复制并替换现有工程中对应文件即可。具体请参考“***获取位置***”项。

开发文件使用

MPC2860 提供 windows 标准的 DLL 文件，凡是支持调用该种 DLL 文件的开发环境和语言都可以用来开发 MPC2860。我司驱动程序安装包中提供了常见开发工具对应的开发文件，如：VC++、VB、C#。

开发文件使用

C++ 篇

✧ **隐式调用**

- 打开 Visual Studio 2010，创建新的 Visual C++工程。
- 获取开发所需文件“MPC2860.h”和“MPC2860.lib”(获取方法请参考“开发文件获取”一文)，将文件复制到新建工程所在目录中。
- 打开工程“工程”->“属性”，在“输入”目录中找到“附加依赖项”，输入“MPC2860.lib”，点击“确定”。
- 在工程文件中加上 #include”MPC2860.h” 即可。

✧ **显示调用**

- 打开 Visual Studio 2010， 创建新的 Visual C++工程。
- 获取开发所需文件“CLoad.h”和“CLoad.cpp” (获取方法请参考“***开发文件获取***”一文)，将文件复制到新建工程所在目录中。
- 在工程文件中添加 #include”CLoad.h”，生成 CLoad 类对象即可。

VB（VB.NET）篇

✧ **VB 6.0**

- 打开 VB6.0，创建新工程。
- 获取开发文件“MPC2860.bas”（获取方法请参考“开发文件获取”一文），将其复制到新建工程所在目录中。
- 点击“工程”下拉菜单，在下拉菜单中点击“添加模块”项，在弹出的对话框中选取添加已存在模块，在目录中找到“MPC2860.bas ”，点击“打开”即可。

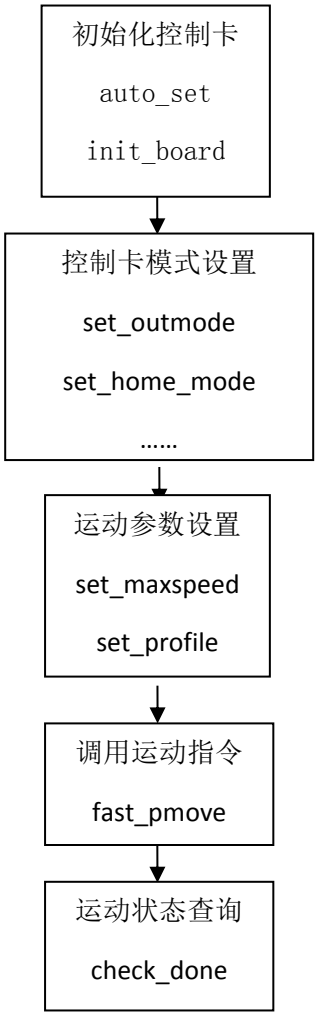
✧ **VB.NET**

- 打开 Visual Studio 2010，创建新的 Visual Basic 工程。
- 获取开发文件“MPC2860.vb” (获取方法请参考“***开发文件获取***”一文)，将其复制到新建工程所在目录中。
- 点击“工程”下拉菜单，在下拉菜单中选取“添加已存在项”，在弹出的对话框中找到“MPC2860.vb”，点击“添加”即可。

C# 篇

- 打开 Visual Studio 2010，创建新的 Visual C#工程。
- 获取开发类文件“MPC2860.cs” (获取方法请参考“开发文件获取”一文)，将其复制到新建工程所在目录中。
- 点击“工程”下拉菜单，在下拉菜单中选取“添加已存在项”，在弹出的对话框中找到“MPC2860.cs ”，点击“添加”即可。

运动指令典型流程图


初始化控制卡
auto_set
init_board
控制卡模式设置
set_outmode
set_home_mode
……
运动参数设置
set_maxspeed
set_profile
调用运动指令
fast_pmove
运动状态查询
check_done

1 软件系统概述

1.1 控制器初始化

1.1.1 指令列表

初始化函数有 2 个，若要使用 MPC2860 各项功能，必须首先依次调用函数 auto_set、init_board，完成控制器初始化后才能调用其它函数。初始化函数如表 2-1 所示。

表 1-1 控制器初始化函数

函数	返回值	说明
auto_set	≥0：轴数 -1：错误	自动检测和自动设置控制器
init_board	≥0：卡数 -1：错误	对控制器硬件和软件初始化

1.1.2 功能说明

（1）auto_set 说明

首先必须调用 auto_set 函数自动检测控制器，执行正确返回卡上轴数。检测用户设置的板卡号是否正确。

（2）init_board 说明

必须调用 init_board 函数检测函数库、驱动程序、控制器固件版本号，初始化控制器所有寄存器为默认状态，init_board 应在 auto_set 之后调用，为了防止用户误操作，auto_set 和 init_board 只能调用一次。执行正确返回计算机内安装的控制器卡数。初始化后控制器初始状态为：

- 脉冲输入模式：Pul/Dir（脉冲/方向）
- 回零模式：0 ---- 原点有效立即停止
- 速度参数

表 1-2 初始化后速度参数

最大速度	4M pps	
点对点运动	常速	2000 pps
	快速	低速 2000pps、高速 8000pps、上升/下降加速度 80000pps
直线插补运动	常速	2000 pps
	快速	低速 2000pps、高速 8000pps、上升/下降加速度 80000pps

- 快速速度曲线：梯形
- 专用输入：

表 1-3 初始化后专用输入状态

	有效状态	有效电平
Org（轴原点）	有效	低电平有效
EL±（轴限位）	有效	低电平有效
Alm（轴报警）	有效	低电平有效
Card_Alm（板卡报警）	有效	低电平有效

- 通用输出：默认输出高电平
- 默认绝对位置值： 0

例程：InitMPC2860()

```
{
    int Rtn;
    Rtn = auto_set();                    //自动设置
    if(Rtn <= 0 )
     {
        返回错误代码
      }
        else
     {
         自动设置成功，获得轴数
      }

        Rtn = init_board();                    //初始化板卡
        I f(Rtn <= 0 )
       {
              返回错误代码
       }
        else
     {
         初始化成功，获得卡数
      }
  ……
  }
```

1.2 轴属性设置

1.2.1 指令列表

表 1-4 轴属性指令列表

函数	返回值	说明
set_outmode	0：正确	设置轴的脉冲输出模式

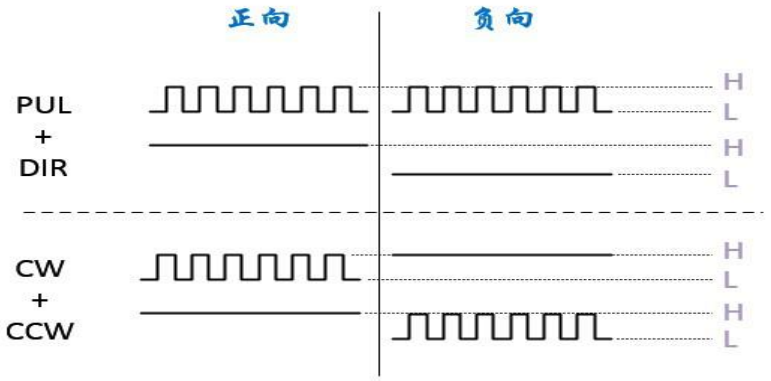
	-1：错误	
set_home_mode	0：正确 -1：错误	设置轴的回原点模式
set_dir	0：正确 -1：错误	设置方向信号电平状态
enable_el	0：正确 -1：错误	使能或禁止限位信号
enable_org	0：正确 -1：错误	使能或禁止原点信号
enable_alm	0：正确 -1：错误	使能或禁止轴报警信号
set_el_logic	0：正确 -1：错误	设置限位信号有效电平
set_org_logic	0：正确 -1：错误	设置原点信号有效电平
set_alm_logic	0：正确 -1：错误	设置轴报警信号有效电平

1.2.2 功能说明

（1）set_outmode 说明

运动控制器提供了两种脉冲输出模式：“脉冲+方向”和“正负脉冲”，默认脉冲输出方式为“脉冲+方向”。调用 set_outmode 指令，可将脉冲输出模式设置为“正负脉冲”方式。
该函数在在 init_board 函数后调用。两种模式的波形如下：

表 1-5 输出模式



（2）set_home_mode 说明

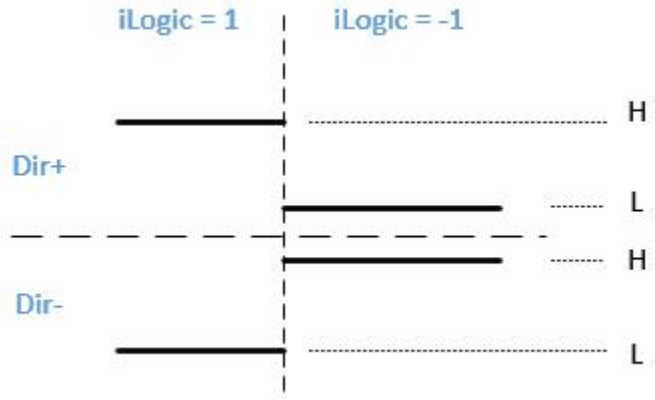
运动控制器提供四种回原点模式：
0 ---- 指定轴 Org 信号有效立即停止
1 ---- 指定轴以梯形速度运动时，遇到轴 Org 信号有效时立即减速，到达梯形低速时停止
2 ---- 检测到指定轴编码器 Z 信号输入时立即停止
3 ---- 指定轴以梯形速度运动时，遇到轴 Org 信号立即减速至低速，遇到轴编码器 Z 信号立即停

止

（3）set_dir 说明

set_dir(int ch, int iLogic)可以设置 Dir±信号的电平状态，如下图所示。

表 1-6：dir 信号电平状态



某些驱动器要求调用运动指令时，对应脉冲/方向信号输出口中的方向信号超前于脉冲信号，这时电机才能正常运转。MPC2860 输出的脉冲/方向信号几乎同时，在此种应用场景下，可以先调用 set_dir()来设置 Dir±信号的电平状态。

例如：调用 fast_pmove(3, 10000)，第 3 轴正向运动 10000 个脉冲。在发送运动指令之前，调用 set_dir(3, 1)，将方向信号的电平状态设置成和下一条运动指令方向信号一样即可。

set_dir()改变的是 Dir+和 Dir-信号的电平，并不会影响下一条运动指令的方向。调用运动指令时，轴运动方向根据运动指令中的位移值来确定：正值代表正向；负值代表负向。

（4）enable_el 说明

运动控制器为每个控制轴提供了正负限位信号输入接口。默认情况下，限位开关处于使能状态，高电平有效。这时各轴限位开关应与转接板上相应限位信号输入引脚和 OGND 相连。其接线方式请参见《MPC2860 User》中的“专用输入的连接方法”一节。当某轴的限位开关触发时，运动控制器将立即停止该方向运动，以保障系统安全。
若控制系统不使用限位信号，调用“enable_el”指令，将相应参数设置为 0，可将该引脚设置作为通用输入口使用。注意：设置限位信号使能时是正负限位一起设置。

（5）enable_org 说明

运动控制器为每个控制轴提供了原点信号输入接口。默认情况下，原点开关处于使能状态，高电平有效。这时各轴原点开关应与转接板上相应原点信号输入引脚和 OGND 相连。其接线方式请参见《MPC2860 User》中的“专用输入的连接方法”一节。当某轴作回原点运动时，若原点开关触发，运动控制器将自动停止运动。原点信号只对回原点运动有效。
若控制系统不使用原点信号，调用“enable_org”指令，将相应参数设置为 0，可将该引脚设置作为通用输入口使用。

（6）enable_alm 说明

运动控制器每个控制轴提供了轴报警信号输入接口。默认情况下，轴报警开关处于使能状态，高电平有效。这时报警开关应与转接板上相应报警信号输入引脚和 OGND 相连。其接线方式请参见《MPC2860 User》中的“专用输入的连接方法”一节。当运动控制器处于运动状态时，若报警开关触发，运动控制器对应轴将自动停止运动。
若控制系统不使用报警信号，调用“enable_alm”指令，将相应参数设置为 0，可将该引脚设置作为通用输入口使用

（7）set_el_logic 说明

在限位信号使能的情况下，运动控制器默认正负限位信号为高电平有效，即当限位信号接口有高电平输入时，触发对应轴的限位状态。通过指令“set_el_logic”的参数可设置限位开关的触发电平，将相应参数设置为 1，表示对应轴的限位开关设置为高电平触发；参数设置为 0 表示对应轴的限位开关为低电平触发。用户应根据现场条件即传感器类型设置对应的参数。注意：设置限位信号有效电平时是正负限位信号一起设置。

（8）set_org_logic 说明

在原点信号使能的情况下，运动控制器默认正负限位信号为高电平有效，即当原点信号接口有高电平输入时，触发对应轴的限位状态。通过指令“set_org_logic”的参数可设置限位开关的触发电平，将相应参数设置为 1，表示对应轴的原点开关设置为高电平触发；参数设置为 0 表示对应轴的原点开关为低电平触发。用户应根据现场条件即传感器类型设置对应的参数

（9）set_alm_logic 说明

在轴报警信号使能的情况下，运动控制器默认正负限位信号为高电平有效，即当报警信号接口有高电平输入时，触发对应轴的限位状态。通过指令“set_alm_logic”的参数可设置报警开关的触发电平，将相应参数设置为 1，表示对应轴的报警开关设置为高电平触发；参数设置为 0 表示对应轴的报警开关为低电平触发。用户应根据现场条件即传感器类型设置对应的参数

例程：

```
void main( )
{
        auto_set( );                    //检测控制器
        init_board( );                  //初始化控制器
        set_outmode ( 1,1,0);  //设置一轴脉冲输出模式为脉冲+方向模式（第三个参数 暂不使用，设置为 0）
        set_home_mode(1,0);  //设置一轴回原点模式为 Org 信号有效立即停止
        enable_el(1,1); //使能一轴限位信号
```

```
enable_org(1,1); //使能一轴原点信号

enable_alm(1,0); //禁止一轴报警信号

set _el_logic(1,0);     //设置一轴限位信号为低电平有效，此时如果一轴限位信号接口有低电平输入，会触发轴的限位状态，一轴轴对应方向的运动会停止

set _org_logic(1,0);     //设置一轴原点信号为低电平有效，此时如果原点信号接口有低电平输入，会触发轴的原点状态，如果一轴轴正在做回原点运动，

……

}
```

1.3 板卡报警信号设置及查询

1.3.1 指令列表

表 1-7 板卡报警信号设置及查询指令列表

函数	返回值	说明
enable_card_alm	0：正确 -1：错误	使能或禁止卡报警信号
set_card_alm_logic	0：正确 -1：错误	设置卡报警有效电平
check_card_alarm	0：正确 -1：错误	使能板卡报警信号后检测板卡报警信号有效状态

1.3.2 功能说明

（1） enable_card_alm 说明

运动控制器提供了一个板卡报警信号输入接口。当运动控制器处于运动状态时，若报警开关触发，运动控制器将自动停止所有运动。若控制系统不使用报警信号，调用“enable_alm”指令，将相应参数设置为 0，可将该引脚设置作为通用输入口使用

卡报警接口采用 alm+com 的接法，可适应 NPN 和 PNP 型传感器。具体接线方法如下：

① 使用 NPN 型传感器：NPN 型传感器输出有驱动能力的电平是低电平，所以外部接线时 com 端接 24V，alm 端接传感器信号

② 使用 PNP 型传感器：PNP 型传感器输出有驱动能力的电平是高电平，所以外部接线时 com 端接 GND（0V），alm 端接传感器信号

（2） set_card_alm_logic 说明

由于卡报警信号采用了带 com 端的接法，其设置有效电平的方式也与其他专用输入不一样，会受到外部接线的影响。 set_card_alm_logic（int cardno，int flag）与外部信号关系如下（接线必须采用（1）enable_card_alm 说明中的方法）：

表 1-8 卡报警信号

flag 值	报警信号是否输入	报警是否有效
0	有	有效
1	有	无效
0	无	无效
1	无	有效

其中：NPN 型传感器的报警信号输入为低电平（0V）

PNP 型传感器的报警信号输入为高电平（24V）

(3)check_card_alarm 说明

函数 check_card_alarm 用于检测板卡的报警开关状态，返回是否有有效的报警信号输入板卡。只有在调用“enable_card_alm”指令使相应专用信号使能，才能返回正确的专用输入口状态。如果 check_alm 返回 1 表示到达报警开关位置，返回 0 则表示未到达报警开关位置，若调用出错则返回-3。

例程：

```
……
    Int Rtn
    enable_card_alm(1,1); //使能一号卡报警信号
    set_card_alm_logic(1,0); //对照上表 1-8，此时如果有报警信号输入
                             （NPN 为 0V，PNP 为 24V），触发报警状态
    set_card_alm_logic(1,1); //对照上表 1-8，此时如果有报警信号输入
                             （NPN 为 0V，PNP 为 24V），不触发报警。如果报警信
                             号断开，触发报警
    Rtn=check_alarm(1);//检测一号卡报警状态，返回值： 0（无效）----- 1
                       （有效）---- -3（出错）
    ……
```

1.4 编码器设置

1.4.1 指令列表

表 1-9 编码器设置指令列表

函数	返回值	说明
set_encoder_mode	0：正确 -1：错误	设置编码器反馈模式

set_encoder	0：正确 -1：错误	设置编码器位置
get_encoder	0：正确 -1：错误	获取编码器位置

1.4.2 功能说明

（1）set_encoder_mode 说明

在默认情况下，init_board 函数将辅助编码器设置为 A/B 相 90 度相位差 4 倍频方式。另外，还可设置为增减脉冲方式，倍频也可设置为 1 倍频。此时在 init_board 函数后调用 set_encoder_mode 设置成所要求的模式。硬件接线见《MPC2860 User》中“编码器输入连接方法”一节。

（2）set_encoder 说明

设置当前编码器的绝对位置

（3）get_encoder 说明

获取当前编码器的绝对位置，如果此前调用过 set_encoder 来设置编码器的位置，获取的位置则是相对于此设置位置的值。

例程：

```
……
double count;
set_encoder_mode(1,0,4,0);//设置编码器模式为 A/B 相相位差 90 度 4 倍频模式
get_encoder(1,&count);//获取一轴编码器位置，存在指针 count 中
……
```

1.5 速度参数设置

1.5.1 指令列表

表 1-10 速度参数设置指令列表

函数	返回值	说明
set_maxspeed	0：正确 -1：错误	设置轴的最大速度
set_conspeed	0：正确 -1：错误	设置轴在常速运动方式下的速度参数
set_profile	0：正确 -1：错误	设定轴在快速运动方式下的速度参数
set_vector_conspeed	0：正确 -1：错误	设置常速运动方式下的矢量常速度参数

set_vector_profile	0：正确 -1：错误	设置快速运动方式下的矢量梯形速度参数

1.5.2 功能说明

（1）set_maxspeed 说明

用户设置一个轴的最大输出脉冲频率，该频率主要用于确定运动控制器脉冲输出倍率。MPC2860 运动控制器的输出脉冲频率由两个变量控制：脉冲分辨率和倍率，两者的乘积即输出的脉冲频率。由于运动控制器内部倍率寄存器长度是有限的，为 13 位（最大值为 8191），如果要达到 4000KHz 的最大输出脉冲频率，脉冲分辨率为（4000000/8191）=488Hz，如果实际使用的脉冲频率为 100Hz，显然 MPC2860 只能输出一个分辨率的脉冲频率（即 488Hz）。为了解决这个问题，调用 set_maxspeed 设置需要达到的最大输出脉冲频率，如 set_maxspeed（1，100）。设置后脉冲分辨率将被重新设置，为（100/8191）=0.012Hz，这样就能提高速度分辨率。但是，此时运动控制器的最大输出脉冲频率只能达到 100Hz。注意：MPC2860 的最高分辨率为可以达到 0.01，此时控制器的最大输出脉冲频率只能达到 82Hz。

在缺省情况下，init_board 函数将所有轴设置其最大速度，即 4000000（4MHz）。使用时可按照实际输出速度进行设置以获得比较好的速度精度。最大输出脉冲频率范围：81.91~4000000 Hz。

set_maxspeed 指令适用于点位运动指令、回零运动指令、连续运动指令、插补运动指令。

（2）set_conspeed 说明

函数 set_conspeed 设定一个轴在常速运动方式下的速度。常速运动速度曲线如下图所示：

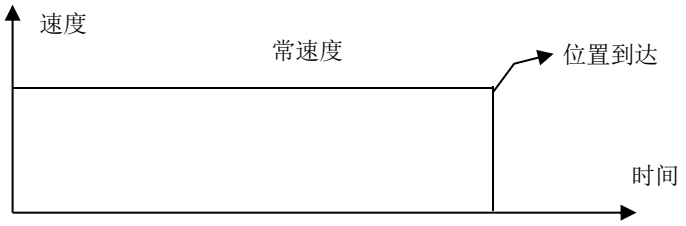


表 1-11 常速运动方式的速度曲线

该速度与运动控制器最终脉冲输出速度可能不一致。这时因为输出脉冲频率由两个变量控制：脉冲分辨率和倍率，倍率越大则误差越大。设设置的最大速度为 V_{max}，输出倍率为 multip，脉冲实际输出速度为 V_{act}，设置的常速度为 V_{con}，有如下计算公式：

$$multip = \frac{V_{\max}}{8191}$$

$$v_{act} = \text{int}(\frac{v_{con}}{multip}) \times multip$$

如果多次调用这个函数，最后一次设定的值有效，而且在下一次改变之前，一直保持有效。最大脉冲频率可设置为 4000000 Hz，最小脉冲频率可设置为 0.2 Hz。

（3）set_profile 说明

函数 set_profile 设定一个轴在快速运动方式下的低速（起始速度）、高速（目标速度）、加 / 减速度值（梯形速度模式下，减速度值可以不等于加速度值）。快速运动方式速度曲线如图所示。

该速度与运动控制器最终脉冲输出速度可能不一致，原因与 set_conspeed 一样。

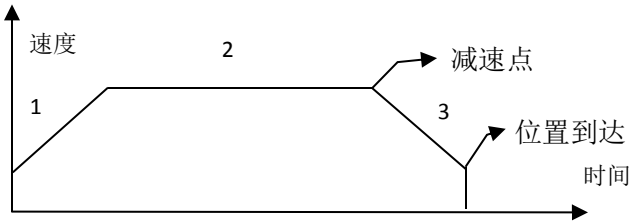


表 1-12 快速运动方式的速度曲

1：加速段，起始速度按设定的加速度加速到目标速度；

2：高速段，保持目标速度，直至运动到减速点；

3：减速段：目标速度按设定的加速度减速到起始速度；

在该快速运动方式下，低速值脉冲频率最小为 10Hz，高速值脉冲频率最大为 2000000Hz，加速值最小为 20。

（4）set_vector_conspeed 说明

函数 set_vector_conspeed 设置常速方式下的矢量速度，这个矢量速度在两轴及以上直线插补及圆弧插补常速运动中将会用到。矢量速度与 set_maxspeed 设置的最大速度无直接关系，即 set_maxspeed 函数确定的速度倍率不影响插补运动。最大脉冲频率可设置为 4000000 Hz，最小脉冲频率可设置为 1Hz。

（5）set_vector_profile 说明

函数 set_vector_profile 设置快速运动方式下的矢量低速为（起始速度）、矢量高速、矢量加/减速度（矢量减速度等于矢量加速度）。这些矢量值在两轴及以上直线插补、圆弧插补快速运动中将会用到。矢量速度与 set_maxspeed 设置的最大速度无直接关系，即 set_maxspeed 函数确定的速度倍率不影响插补运动。低速值脉冲频率最小可设置为 10Hz，高速值脉冲频率最大可设置为 4000000Hz，加速值最小可设置为 20。

例程：

```
……
set_maxspeed(1,5000);//设置一轴最大速度 5000
set_conspeed(1,2000);//设置一轴常速速度 2000
```

set_profile(1,1000,5000,50000,50000);//设置一轴梯形参数，低速 1000，高速 5000，加速度 50000，减速度 50000

set_vector_conspeed(2000);//设置矢量常速度 2000

……

1.6 点位运动

1.6.1 常速运动

1.6.1.1 指令列表

常速模式独立运动速度设置及运动函数有 7 个，如下表：

表 1-13 常速运动指令列表

函数	返回值	说明
set_maxspeed	0：正确 -1：错误	设置轴的最大速度
set_conspeed	0：正确 -1：错误	设置轴在常速运动方式下的速度参数
con_pmove	0：正确 -1：错误	1 个轴以常速做相对位置点位运动
con_pmove2	0：正确 -1：错误	2 个轴以常速做相对位置点位运动
con_pmove3	0：正确 -1：错误	3 个轴以常速做相对位置点位运动
con_pmove4	0：正确 -1：错误	4 个轴以常速做相对位置点位运动
con_pmove_to	0：正确 -1：错误	1 个轴以常速做绝对位置点位运动

1.6.1.2 功能说明

MPC2860 运动控制器提供了 5 个常速点位运动函数。点位运动是指各轴按各自设定的速度、加速度和行程运动，直到到达设定位置或调用停止指令使轴停止运动。点位运动函数中行程参数均以脉冲数为编程单位。

（1）con_pmove、con_pmove2、con_pmove3、con_pmove4 说明

分别用于启动一个轴、两个轴、三个轴或四个轴，各轴独立以常速方式作点位运动，指令中的位置参数均指相对位移量。启动运动前先调用“set_maxspeed”设置控制轴需要的最大速度（确定了运动控制器脉冲输出倍率及分辨率），然后调用指“set_conspeed”设置运动速度。注意：多轴启动时不一定同时到达。

（2）con_pmove_to 说明

用于一个轴以常速方式做点位运动，指令中的位置参数为绝对位移量，启动运动前先调用“set_maxspeed”设置控制轴需要的最大速度（确定了运动控制器脉冲输出倍率及分辨率），然后调用指令“set_conspeed”设置运动速度。

例程：

```
void main( )
{
    auto_set( );                    //检测控制器
    init_board( );                  //初始化控制器
    set_maxspeed(1,2000);   //设置 1 轴最大速度为 2000
    set_conspeed(1,2000);   //设置 1 轴常速度为 2000
    con_pmove(1,5000);      //启动 1 轴常速运动，行程 5000
     ……
    con_pmove_to(1,0); //1 轴以常速运动到开始绝对位置 0 点
}
```

1.6.2 梯形曲线运动模式

1.6.2.1 指令列表

梯形曲线独立运动速度设置及运动函数有 8 个，如下表所示：

表 1-14 梯形曲线运动指令列表

函数	返回值	说明
set_s_curve	0：正确 -1：错误	设置控制轴快速运动模式
set_maxspeed	0：正确 -1：错误	设置轴的最大速度
set_profile	0：正确 -1：错误	设置快速运动的低速、高速和加速度
fast_pmove	0：正确 -1：错误	1 个轴以快速做相对位置点位运动
fast_pmove2	0：正确 -1：错误	2 个轴以快速做相对位置点位运动
fast_pmove3	0：正确 -1：错误	3 个轴以快速做相对位置点位运动
fast_pmove4	0：正确 -1：错误	4 个轴以快速做相对位置点位运动
fast_pmove_to	0：正确 -1：错误	1 个轴以快速做绝对位置点位运动

1.6.2.2 功能说明

MPC2860 运动控制器提供了 8 个梯形模式点位运动处理函数。点位运动是指各轴按各自设定的速度、加速度和行程运动，直到到达设定位置或调用停止指令使轴停止运动。点位运动函数中行程参数均以脉冲数为编程单位。

（1）set_s_curve 说明

MPC2860 运动控制器为点位运动轴提供了两种快速运动模式：梯形曲线，S 形曲线，使用“set_s_curve”函数设置某个轴在快速运行时采用哪一种加减速方式。系统默认为梯形速度模式。

（2）fast_pmove、fast_pmove2、fast_pmove3、fast_pmove4 说明

分别用于启动一个轴、两个轴、三个轴或四个轴，各轴独立以快速方式作点位运动，指令中的位置参数均指相对位移量。梯形速度由指令“set_profile”设置。启动运动前先调用“set_maxspeed”设置控制轴需要的最大速度（确定了运动控制器脉冲输出倍率及分辨率）。多轴启动时不一定同时到达。

（3）fast_pmove_to 说明

用于启动一个轴以快速方式作点位运动，指令中的位置参数为绝对位移量。梯形速度由指令“set_profile”设置。启动运动前先调用“set_maxspeed”设置控制轴需要的最大速度（确定了运动控制器脉冲输出倍率及分辨率）。

```
void main( )
{
        auto_set( );                    //检测控制器
        init_board( );                  //初始化控制器
        set_s_curve( 1,0);              //设置梯形速度模式
        set_maxspeed(1,5000);   //设置 1 轴最大速度为 2000
        set_profile(1,100,5000,3000,3000); //设置 1 轴初速、高速、加速度和减速度
        fast_pmove (1,5000);     //启动 1 轴梯形快速运动，运动距离 5000
          ……
       fast_pmove_to(1,6000);//1 轴以梯形快速运动方式运动到绝对位置 6000
 }
```

1.6.3S 型曲线运动模式

1.6.3.1 指令列表

表 1-15 S 型曲线指令列表

函数	返回值	说明
set_s_curve	0：正确 -1：错误	设置控制轴快速运动模式

set_s_section	0：正确 -1：错误	设置 S 型升降速加加速度
set_profile	0：正确 -1：错误	设置快速运动的低速、高速、加速度和减速度
fast_pmove	0：正确 -1：错误	1 个轴以快速做相对点位运动
fast_pmove2	0：正确 -1：错误	2 个轴以快速做相对点位运动
fast_pmove3	0：正确 -1：错误	3 个轴以快速做相对点位运动
fast_pmove4	0：正确 -1：错误	4 个轴以快速做相对点位运动
fast_pmove_to	0：正确 -1：错误	1 个轴以快速做绝对点位运动

1.6.3.2 功能说明

（1）set_s_section 说明

功能：设置轴 S 型曲线运动加加速度。“set_profile”设置 S 形的初速、高速、最大加速度和最大减速度，但是 S 曲线的最大加速度和最大减速度要求相等。“set_s_section”在“set_profile”之后调用，设置 S 形曲线运动的加加速度和加减速度。S 曲线加速度运动只能在快速点位运动中有效，插补运动不提供 S 曲线加减速模式。加加速段和减加速段的加加速度值，范围 10~375M ，且加减速段和减减速段的加减速度值必须相等，否则调用会失败。

例程

```
void main( )

{

        auto_set( );                    //检测控制器

        init_board( );                  //初始化控制器

        set_s_curve( 1,1);              //设置 S 形速度模式

        set_profile(1,100,5000,1000);   //设置 1 轴初速、高速和最大加速度

        set_s_section(1,80000,80000); //设置 1 轴的 S 曲线加加速度和加减速度。

        fast_pmove (1,5000);    //启动 1 轴 S 形快速运动，行程为 5000

}
```

1.7 连续运动函数

1.7.1 指令列表

连续运动参数设置和运动函数有如表所示：

表 1-16 连续运动指令列表

函数	返回值	说明
set_maxspeed	0：正确 -1：错误	设置轴运动的最大速度
set_profile	0：正确 -1：错误	设置轴快速运动的参数
set_conspeed	0：正确 -1：错误	设置轴常速运动的参数
con_vmove	0：正确 -1：错误	1 个轴以常速做连续运动
con_vmove2	0：正确 -1：错误	2 个轴以常速做连续运动
con_vmove3	0：正确 -1：错误	3 个轴以常速做连续运动
con_vmove4	0：正确 -1：错误	4 个轴以常速做连续运动
fast_vmove	0：正确 -1：错误	1 个轴以快速做连续运动
fast_vmove2	0：正确 -1：错误	2 个轴以快速做连续运动
fast_vmove3	0：正确 -1：错误	3 个轴以快速做连续运动
fast_vmove4	0：正确 -1：错误	4 个轴以快速做连续运动

1.7.2 功能说明

连续运动是指各轴按各自设定的速度、加速度运动，直到有相应方向的限位、报警信号，或者调用停止指令才停止运动。

（1）con_vmove、con_vmove2、con_vmove3、con_vmove4 说明

分别用于同时启动一个轴、两个轴、三个轴或四个轴，各轴独立以常速方式作连续运动，常速度由指令“set_conspeed”设置。

（2）fast_vmove、fast_vmove2、fast_vmove3、fast_vmove4 说明

分别用于同时启动一个轴、两个轴、三个轴或四个轴，各轴独立以快速方式作连续运动，梯形速度由指令“set_profile”设置。

例程：

```
void main( )
{
        auto_set( );                    //检测控制器
        init_board( );                  //初始化控制器
        set_maxspeed(1,10000);          //设置最大速度
        set_profile(1,100,10000,8000);//设置初速、高速、加速度
        fast_vmove(1,1);                //启动 1 轴正向连续运动
……
}
```

1.8 回原点运动函数

1.8.1 指令列表

回原点运动参数设置和运动函数如表 1-17 所示。

表 1-17 回原点运动函数

函数	返回值	说明
set_maxspeed	0：正确 -1：错误	设置轴运动的最大速度
set_profile	0：正确 -1：错误	设置轴快速运动的参数
set_conspeed	0：正确 -1：错误	设置轴常速运动的参数
set_home_mode	0：正确 -1：错误	设置轴回零运动的模式
con_hmove	0：正确	1 个轴以常速做回零运动

	-1：错误	
con_hmove2	0：正确 -1：错误	2 个轴以常速做回零运动
con_hmove3	0：正确 -1：错误	3 个轴以常速做回零运动
con_hmove4	0：正确 -1：错误	4 个轴以常速做回零运动
fast_hmove	0：正确 -1：错误	1 个轴以快速做回零运动
fast_hmove2	0：正确 -1：错误	2 个轴以快速做回零运动
fast_hmove3	0：正确 -1：错误	3 个轴以快速做回零运动
fast_hmove4	0：正确 -1：错误	4 个轴以快速做回零运动

1.8.2 功能说明

轴按各自设定的速度、加速度运动，直到有外部原点信号、限位信号或报警信号，或者调用停止指令才停止运动。运动控制器提供了 4 种回原点方式，通过接口函数“set_home_mode”设置工作方式，详见“轴属性设置”

（1）con_hmove、con_hmove2、con_hmove3、con_hmove4 说明

分别用于同时启动一个轴、两个轴、三个轴或四个轴，各轴独立以常速方式作回原点运动，常速度由指令“set_conspeed”设置。

（2）fast_hmove、fast_hmove2、fast_hmove3、fast_hmove4 说明

分别用于同时启动一个轴、两个轴、三个轴或四个轴，各轴独立以快速方式作回原点运动，梯形速度由指令“set_profile”设置。

例程

```
void main( )
{
        auto_set( );                    //检测控制器
```

```
init_board( );                    //初始化控制器
set_home_mode(1,0);    //设置 1 轴回原点模式 0
set_maxspeed(1,1000);  //设置最大速度
set_conspeed(1,1000);  //设置回零速度
con_hmove(1,1);                   //启动 1 轴正向回零运动
……
}
```

1.9 线性插补运动函数

1.9.1 指令列表

线性插补运动设置和运动函数如表所示：

表 1-18 线性插补运动指令列表

函数	返回值	说明
set_vector_conspeed	0：正确 -1：错误	设置常速运动的速度
set_vector_profile	0：正确 -1：错误	设置快速运动的速度参数
con_line2	0：正确 -1：错误	两个轴作常速直线插补运动(相对位置)
con_line3	0：正确 -1：错误	三个轴作常速直线插补运动(相对位置)
con_line4	0：正确 -1：错误	四个轴作常速直线插补运动(相对位置)
fast_line2	0：正确 -1：错误	两个轴作快速直线插补运动(相对位置)
fast_line3	0：正确 -1：错误	三个轴作快速直线插补运动(相对位置)
fast_line4	0：正确 -1：错误	四个轴作快速直线插补运动(相对位置)

con_line2_to	0：正确 -1：错误	两个轴作常速直线插补运动(绝对位置)
con_line3_to	0：正确 -1：错误	三个轴作常速直线插补运动(绝对位置)
con_line4_to	0：正确 -1：错误	四个轴作常速直线插补运动(绝对位置)
fast_line2_to	0：正确 -1：错误	两个轴作快速直线插补运动(绝对位置)
fast_line3_to	0：正确 -1：错误	三个轴作快速直线插补运动(绝对位置)
fast_line4_to	0：正确 -1：错误	四个轴作快速直线插补运动(绝对位置)

1.9.2 功能说明

插补运动是指两轴及多轴按照一定的算法进行联动，被控轴同时启动，并同时到达目标位置。插补运动以矢量速度运行，矢量速度分为常矢量速度和梯形矢量速度。插补运动函数中行程参数以脉冲数为编程单位。

MPC2860 运动控制器提供了 12 个线性插补运动函数。

（1）con_line2、con_line3、con_line4、con_line2_to、con_line3_to、con_line4_to 说明

分别启动两个轴、三个轴、四个轴以常矢量速度作线性联动，每个被控轴的运动速度为常矢量速度在该轴上的分速度，各个被控轴同时启动，并同时到达目标位置。常矢量速度由指令“set_vector_conspeed”设置。其中 **con_line2、con_line3、con_line4** 的位移参数为相对坐标，**con_line2_to、con_line3_to、con_line4_to** 的位移参数为绝对坐标。

（2）fast_line2、fast_line3、fast_line4、fast_line2_to、fast_line3_to、fast_line_to 说明

分别启动两个轴、三个轴、四个轴以梯形矢量速度作线性联动，每个被控轴的运动速度、加速度为梯形矢量速度、矢量加速度在该轴上的分量，各个被控轴同时启动，并同时到达目标位置。梯形矢量速度由指令“set_vector_profile”设置。其中 **fast_line2、fast_line3、fast_line4** 的位移参数为相对坐标，**fast_line2_to、fast_line3_to、fast_line_to** 的位移参数为绝对坐标。

例程：

```
……
set_vector_conspeed(2000);//设置常速矢量速度 2000
```

con_line2(1,10000,2,15000);//2 轴插补，一轴位移 10000（相对位置），2 轴位移 15000（相对位置）

……

1.10 制动函数

1.10.1 指令列表

表 1-19 制动函数指令列表

函数	返回值	说明
sudden_stop	0：正确 -1：错误	立即制动一个运动轴
sudden_stop2	0：正确 -1：错误	立即制动两个运动轴
sudden_stop3	0：正确 -1：错误	立即制动三个运动轴
sudden_stop4	0：正确 -1：错误	立即制动四个运动轴
decel_stop	0：正确 -1：错误	光滑制动一个运动轴
decel_stop2	0：正确 -1：错误	光滑制动两个运动轴
decel_stop3	0：正确 -1：错误	光滑制动三个运动轴
decel_stop4	0：正确 -1：错误	光滑制动四个运动轴
move_pause	0：正确 -1：错误	暂停一个运动轴
move_resume	0：正确 -1：错误	恢复一个轴的运动

1.10.2 功能说明

（1）sudden_stop、sudden_stop2、sudden_stop3、sudden_stop4 说明

立即运动方式下立即制动函数。它们使被控轴立即中止运动。这个函数执行后，控制器立即停止向电机驱动器发送脉冲，使之停止运动。

（2）decel_stop、decel_stop2、decel_stop3、decel_stop4 说明

立即运动方式下光滑制动函数。只对快速运动指令（梯形速度、S 形速度）有效，即只有“fast”开始的运动指令（如：fast_hmove、fast_vmove、fast_pmove2 等）才能进行光滑制动。它们可以使被控轴的速度先从高速降至低速（由 set_proflile 设定），然后停止运动。光滑制动函数可有效防止快速运动时系统的过冲现象。

（3）move_pause、move_resume 说明

这两个指令用于暂停立即运动模式下某轴的当前运动和恢复轴的运动。

对于快速直线插补运动、快速点位运动、快速连续运动、快速回原点运动等，调用“move_pause”函数，控制轴即以设置的梯形速度减速直到停止；调用“move_resume”后，控制轴又以梯形速度加速到高速。

对于常速直线插补运动、常速点位运动、常速连续运动、常速回原点运动，调用“move_pause”函数，控制轴立即停止运动；调用“move_resume”后，控制轴立即以常速度运动。

例程：

```
……
set_profile(1,2000,5000,50000,50000);//设置一轴梯形运动参数
fast_pmove(1,50000);//启动一轴快速运动
……
move_pause(1);//暂停一轴
……
move_resume(1);//恢复一轴运动
……
```

1.11 位置设置和读取函数

1.11.1 位置设置函数

1.11.1.1 指令列表

位置设置函数有 2 个，如下表：

表 1-20 位置设置和读取函数

函数	返回值	说明
set_abs_pos	0：正确 -1：错误	设置一个轴的绝对位置值
reset_pos	0：正确 -1：错误	设置一个轴的位置值为零

1.11.1.2 功能说明

（1）set_abs_pos 说明

调用该函数可将控制轴当前绝对位置改为设定值，但从上一个位置到该位置之间不会产生轴的实际运动。调用该函数并将第二个参数设为 0 可实现“reset_pos”函数的功能。函数中位置参数以脉冲数为编程单位。

当控制轴处于运动状态时，该指令将不起作用。

（2）reset_pos 说明

函数该函数将控制轴的绝对位置和相对位置复位至 0，通常在轴的原点找到时调用，调用这个函数后，当前位置值变为 0，这以后，所有的绝对位置值均是相对于这一点的。

当控制轴处于运动状态时，该指令将不起作用。该函数同时自动将辅助编码器计数值清零。

例程：

```
……
set_abs_pos(1,0);//设置 1 轴当前绝对位置为 0，此时功能等同于 rest_pos
……
```

1.11.2 位置读取函数

1.11.2.1 指令列表

表 1-21 位置读取函数指令列表

函数	返回值	说明
get_abs_pos	0：正确 -1：错误	读取一个轴的绝对位置值

1.11.2.2 功能说明

（1）get_abs_pos 说明

调用该函数可读取控制轴当前绝对位置值。如果执行过回原点运动（回原点后应调用“reset_pos”指令），那么这个绝对位置是相对于原点位置的；如果没有执行过回原点运动，那么这个绝对位置是相对于开机时的位置。函数中位置参数以脉冲数为编程单位。

该指令获取的控制轴绝对位置是由控制器输出脉冲的数量决定，在丢步或过冲等情况下，不能反映实际的位置值。

1.12 状态处理函数

1.12.1 运动状态查询函数

1.12.1.1 指令列表

运动状态查询函数如表 1-22 所示。

表 1-22 运动状态查询函数

函数	返回值	说明
check_status	其它：状态值 -1：错误	读取指定轴的状态
get_cur_dir	0：停止状态 -1：负向 1：正向 -2：错误	读取指定轴的当前运动方向
check_done	0：停止状态 1：运动状态 -1：错误	检查指定轴的运动是否已经完毕
get_rate	非负（成功） 负值（错误）	读取当前运动的速度
get_max_axe	轴数（成功） 0：错误	读取总轴数
get_board_num	总卡数（成功） 0：错误	获取计算机内运动控制卡总卡数
get_axe	轴数（成功） 0：错误	获取制定板卡上的轴数
get_conspeed	常速速度（成	获取指定轴设定的做常速点位运动

	功） -1：错误	时的速度
get_vector_conspeed	常速速度（成功） -1：错误	获取指定轴数设定的做常速插补运动时的矢量速度
get_profile	0：正确 -1：错误	获取指定轴设定的做快速点位运动时的梯形速度参数
get_vector_profile	0：正确 -1：错误	获取指定轴数设定的做快速插补运动时的矢量梯形速度参数

1.12.1.2 功能说明

（1）check_status 说明

MPC2860 控制控制器每个控制轴都有 1 个 32 位状态寄存器，在运动过程中，用户可以通过调用指令“check_status”查询轴的工作状态。该状态寄存器中每位（bit）的含义如下表所示。

表 1-23 状态寄存器含义

Bit 位	含义	状态
31	保留	
30	原点输入	1：有效　　0：无效
29	正限位输入	1：有效　　0：无效
28	负限位输入	1：有效　　0：无效
27		
26	报警停止	1：有效　　0：无效
25	轴报警	1：有效　　0：无效
24	总报警	1：有效　　0：无效
23	编码器输入	1：有效　　0：无效
22	限位停止	1：停止　　0：运动
21	保留	
20		
19		
18		
17		
16		
15	总停止	1：有效　　0：无效
14	自动降速点	1：到达自动降速点
13	指令停止	1：急停有效

12	Z 脉冲停止	1：急停有效
11	原点停止 2	1：遇原点缓停至遇到 Z 脉冲停止
10	原点停止 1	1：急停有效
9	原点缓停	1：缓停有效
8	保留	
7		
6		
5		
4		
3		
2	减速状态	1：运动处于降速阶段
1	升速状态	1：运动处于升速阶段
0	运动状态	1：停止　　0：运动

（2）check_done 说明

分别用于检测指定轴的运动状态,只有在轴停止运动后，才能对该轴发出下一条运动指令。若控制轴尚在运动，系统将抛弃后面的运动指令。

（3）get_cur_dir 说明

读取指定轴的当前运动方向。

（4）get_rate 说明

读取控制轴的当前运动速度。有可能读取的速度与用户设置的速度有差异。这主要是由于控制器速度分辨率引起的差异。因为输出脉冲频率由两个变量控制：脉冲分辨率和倍率，两者的乘积为实际输出的脉冲频率。函数 set_maxspeed 设置最大输出脉冲频率即为修改脉冲分辨率，使用时可按照实际输出速度设置最大速度以获得比较好的速度精度。

（5）get_max_axe 说明

获取计算机内板卡的总轴数，调用成功返回轴数，失败则返回 0。

（6）get_board_num 说明

获取计算机内板卡的总卡数，调用成功返回卡数，失败则返回 0。

（7）get_axe 说明

获取制定控制卡的轴数，调用成功返回轴数，失败则返回 0

（8）get_conspeed 说明

获取指定轴数设定的点位运动常速速度，调用成功返回速度值，失败则返回-1

（9）get_vector_conspeed 说明

获取指定轴数设定的做常速插补运动时的矢量速度，调用成功返回速度值，失败则返回-1

（10）get_profile 说明

获取指定轴设定的做快速点位运动时的梯形速度参数，包括低速，高速，加速度和减速度。调用成功返回 0，失败则返回-1

（11）get_vector_profile 说明

获取指定轴数设定的做快速插补运动时的矢量梯形速度参数，包括矢量低速，矢量高速，矢量加速度和矢量减速度。调用成功返回 0，失败则返回-1

1.12.2 专用输入检测函数

1.12.2.1 指令列表

表 1-24 专用输入检测函数

函数	返回值	说明
check_limit	0：无限位信号 1：正限位信号 -1：负限位信号 -3：错误	检查指定轴是否到达限位开关位置
check_home	0：无原点信号 1：有原点信号 -3：错误	检查指定轴是否到达原点开关位置
check_alarm	0：无报警信号 1：有报警信号 -3：错误	检查指定轴是否到达报警开关位置
check_sfr	其它：IO 状态 -1：错误	读取专用输入口所有的开关量状态
check_sfr_bit	0：低电平 1：高电平 -1：错误	读取专用输入口某位的开关量状态

1.12.2.2 功能说明

（1）check_limit、check_home、check_alarm 说明

分别用于检测指定轴的运动状态、限位信号状态、原点信号状态和轴报警信号状态。调用“check_status”函数可读取出整个状态寄存器值，而其余这几个函数分别获取其中部分信号

状态。注意：只有在这些专用输入使能的情况下，即“enable_org”、“enable_limit”、“enable_alm”等指令使相应专用信号使能，上述函数才能返回正确的专用输入口状态。

（2）check_sfr、check_sfr_bit 说明

运动控制器将个控制轴的原点、限位、报警信号保存在一个专用寄存器中，若不使用这些专用输入信号，可用“enable_org”、“enable_limit”、“enable_alm”、“enable_sd”等指令使相应专用信号无效，此时，这些端口可用作通用输入口。

另外，编码器信号也保存在该寄存器中，若没有使用编码器，可将相应端口作通用输入使用，接线方法与编码器单端输入接法相同。当编码器接口作通用输入时，编码器正极性端只能接+5V。

通过 check_sfr(int cardno) 一次性读取所有专用输入口电平状态，根据需要进行位操作就可以获取某一位或某几位专用输入口的电平状态；cardno 是卡号，从 1 开始。

通过 check_sfr_bit(int cardno, int **bitno**) 一次只能读取某一个专用输入口的电平状态。**bitno 是位标记，从 1 开始**，如：获取第 2 卡 EL3-的电平状态，调用 check_sfr_bit(2, 6)，而不是 check_sfr_bit(2, 5);。当作通用输入口时的接线图请参见《MPC2860 User.PDF》中的“通用输入、输出的连接方法”一栏。

专用输入寄存器各位定义下所示。

表 1-25 专用输入寄存器各位含义

Bit 位	含义	电平状态
31	保留	
30	ALM6：第六轴报警	0：低电平 1：高电平
29	ALM5：第五轴报警	0：低电平 1：高电平
28	ALM4：第四轴报警	0：低电平 1：高电平
27	EZ6：第六轴编码器 Z 信号	0：低电平 1：高电平
26	EZ5：第五轴编码器 Z 信号	0：低电平 1：高电平
25	EZ4：第四轴编码器 Z 信号	0：低电平 1：高电平
24	ORG6：第六轴原点	0：低电平 1：高电平
23	ORG5：第五轴原点	0：低电平 1：高电平
22	ORG4：第四轴原点	0：低电平 1：高电平
21	EL6-：第六轴负限位	0：低电平 1：高电平
20	EL5-：第五轴负限位	0：低电平 1：高电平
19	EL4-：第四轴负限位	0：低电平 1：高电平
18	EL6+：第六轴正限位	0：低电平 1：高电平
17	EL5+：第五周正限位	0：低电平 1：高电平
16	EL4+：第四轴正限位	0：低电平 1：高电平
15	ALM3：第三轴报警信号	0：低电平 1：高电平
14	ALM2：第二轴报警信号	0：低电平 1：高电平
13	ALM1：第一轴报警信号	0：低电平 1：高电平
12	ALARM：板卡报警	0：低电平 1：高电平
11	EZ3：第三轴编码器 Z 信号	0：低电平 1：高电平

10	**EZ2**：第二轴编码器 Z 信号	**0**：低电平 **1**：高电平
9	**EZ1**：第一轴编码器 Z 信号	**0**：低电平 **1**：高电平
8	**ORG3**：第三轴原点	**0**：低电平 **1**：高电平
7	**ORG2**：第二轴原点	**0**：低电平 **1**：高电平
6	**ORG1**：第一轴原点	**0**：低电平 **1**：高电平
5	**EL3-**：第三轴负限位	**0**：低电平 **1**：高电平
4	**EL2-**：第二轴负限位	**0**：低电平 **1**：高电平
3	**EL1-**：第一轴负限位	**0**：低电平 **1**：高电平
2	**EL3+**：第三轴正限位	**0**：低电平 **1**：高电平
1	**EL2+**：第二轴正限位	**0**：低电平 **1**：高电平
0	**EL1+**：第一轴正限位	**0**：低电平 **1**：高电平

1.12.3 伺服专用函数

1.12.3.1 指令列表

表 1-26 伺服专用函数指令列表

函数	返回值	说明
checkin_axis_INP	0：低电平 1：高电平 -1：错误	获取伺服到位信号输入
checkin_axis_SRDY	0：低电平 1：高电平 -1：错误	获取伺服准备好信号输入
outport_bit_RST	0：正确 -1：错误	伺服报警清除输出
outport_bit_CL	0：正确 -1：错误	伺服偏差报警计数清除输出
outport_bit_SEVERON	0：正确 -1：错误	伺服使能输出

1.12.3.2 指令说明

（1）checkin_axis_INP 说明

获取伺服当前轴的到位信号（INP）的物理电平状态，伺服到位信号含义请参考伺服相关说明文档。若返回 1 则为高电平，返回 0 则为低电平。

（2）checkin_axis_SRDY 说明

获取伺服当前轴的准备好信号（SRDY）的物理电平状态，伺服准备好信号含义请参考伺服相关说明文档。若返回 1 则为高电平，返回 0 则为低电平。

（3） outport_bit_RST 说明

设置轴所控制的伺服的报警信号清除输出口状态，0 为低电平，1 为高电平。伺服报警信号清除控制口的详细含义请参考伺服相关说明文档。

（4） outport_bit_CL 说明

设置轴所控制伺服的偏差计数清除输出口状态，0 为低电平，1 为高电平。伺服的偏差计数清除控制口的详细含义请参考伺服相关说明文档

（5）checkin_axis_ SEVERON 说明

设置轴所控制伺服使能输出口状态，0 为低电平，1 为高电平。伺服的使能控制口的详细含义请参考伺服相关说明文档。

1.12.4 通用 IO 口操作函数

1.12.4.1 指令列表

运动控制器提供带光电隔离 32 路通用输入口和 32 路通用输出口。通用数字 IO 操作函数如表所示：

表 1-27 数字 IO 操作函数

函数	返回值	说明
checkin_byte	其它：IO 状态 -1：错误	读取扩展输入口所有的开关量状态
checkin_bit	0：低电平 1：高电平 -1：错误	读取扩展输入口某位的开关量状态
outport_byte	0：正确 -1：错误	设置主信号板上通用输出口各位的开关量状态
outport_byte_ex	0：正确 -1：错误	设置扩展 IO 板 EA1616C 上的通用输出口状态。
outport_bit	0：正确 -1：错误	设置通用输出口某位的开关量状态，包括主信号板和 IO 扩展板上

1.12.4.2 功能说明

（1）checkin_byte、checkin_bit 说明

用户使用该指令读取运动控制器 32 路通用输入口状态，其中 1-16 路位于信号转接板 P100-03，17-32 路位于扩展 IO 卡 EA1616C 上。

“checkin_byte” 从运动控制器的 32 位通用输入口读入所有输入开关量状态。

“checkin_bit” 从运动控制器的 32 位通用输入口读入某一位输入开关量状态。

通用输入口接线图请参见《MPC2860 User.PDF》中的“通用输入、输出的连接方法”一栏。

（2）outport_byte、outport_byte_ex、outport_bit、说明

“outport_byte”同时设置主卡上通用输出口开关量状态。

“outport_byte_ex” 同时设置 IO 扩展板 EA1616C 上通用输出口开关量状态

“outport_bit” 设置通用输出口某位的开关量状态。

注意：通用输出口 1 和 2 固定为高速位置比较输出，不能控制为普通通用输出口。参数 data 的 D2-D15 位分别控制转接板 P100-03 上 OUT3、OUT4、OUT5……OUT16。D0 和 D1 位没有实际对应的物理位置。

通用输出口接线图请参见《MPC2860 User.PDF》中的“通用输入、输出的连接方法”一栏。

1.13 其它功能

1.13.1 反向间隙处理

1.13.1.1 指令列表

反向间隙处理函数有 3 个，如表所示。

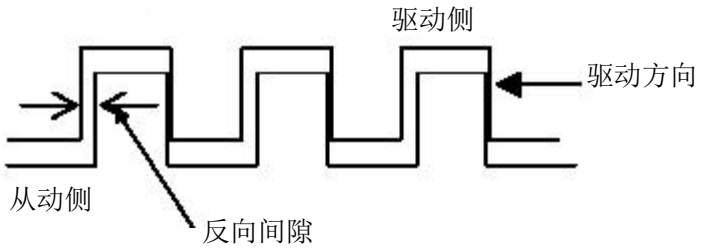
表 1-28 速度设置函数

函数	返回值	说明
set_backlash	0：正确 -1：错误	设置由于机构换向形成间隙的补偿值
start_backlash	0：正确 -1：错误	开始补偿由于机构换向间隙而导致的位置误差
end_backlash	0：正确 -1：错误	停止补偿由于机构换向间隙而导致的位置误差

1.13.1.2 功能说明

（1）set_backlash、start_backlash、end_backlash 说明

作为驱动装置，如果使用齿轮机构或其它传动装置，两个齿之间或多或少存在间隙，如下图所示。当驱动侧的驱动方向发生改变时（如下图右到左或左到右），通过反向间隙补偿值的设定增加驱动侧的运动脉冲数，减小机械间隙的影响。



当控制系统存在反向间隙时，使用 MPC2860 提供的三个函数可有效消除反向间隙，保证系统位置精度。使用步骤如下：

根据反向间隙大小，调用函数“set_backlash”设置一个轴的补偿值。

设置好反向间隙补偿值后，调用函数“start_backlash”启动反向间隙补偿。以后控制轴反向时，运动控制器自动增加反向间隙补偿。

当需要结束反向间隙补偿，调用函数“end_backlash”取消反向间隙补偿。以后控制轴反向时，运动控制器不再进行反向间隙补偿。

例程

```
void main( )
{
        auto_set( );                    //检测控制器
        init_board( );                  //初始化控制器
        set_maxspeed(1,1000);   //设置最大速度
        set_conspeed(1,1000);   //设置回零速度
        set_backlash(1,100);    //设置轴 1 的反向间隙为 100（脉冲）
        start_backlash(1);              //启动反向间隙补偿
        con_pmove(1,10000);     //启动 1 轴正向移动 10000
        ……//等待轴 1 停止
        con_pmove(1,-10000);    //启动 1 轴反向移动 10000，此后系统自动多运动 100 的行
                                程，以补偿反向间隙
……
        end_backlash(1);        //结束反向间隙补偿
……
}
```

1.13.2 运动中变速度

1.13.2.1 指令列表

运动控制器提供运动中变速度功能，操作函数有 1 个，如表所示。

表 1-29 变速度和变加速度处理函数

函数	返回值	说明
change_speed	0：正确 -1：错误	实现运动中变速的功能

1.13.2.2 功能说明

（1）change_speed 说明

运动控制器提供运动中改变控制轴速度和加速度功能。调用“change_speed”立即改变控制轴运动速度，但最大值不能超过“set_maxspeed”设置的最大速度，最小值不能低于 0.2Hz。注意：必须在发出运动指令前设置好最大速度。若最大速度较大，由于运动控制器内部速度分辨率的限制，无法实现较精确的运动速度，关于分辨率问题可参见 set_maxspeed 函数说明。

需要注意的是，这个指令只有在梯形运动模式中才有效。

加速变更：

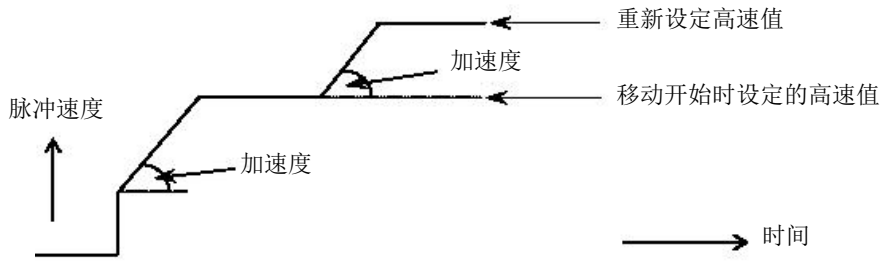


图 1-30 升速过程变速示意图

减速变更：

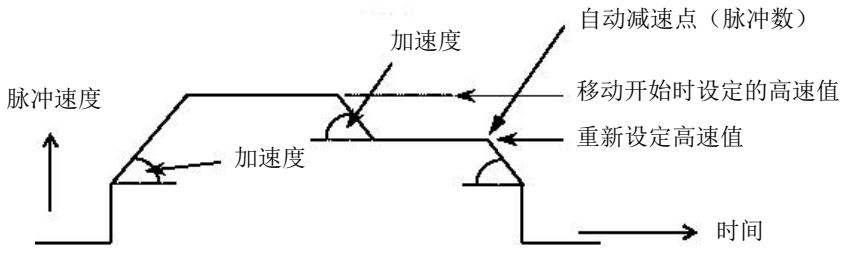


图 1-31 减速过程变速示意图

例程：

```
void main( )
{
        auto_set( );                    //检测控制器
        init_board( );                  //初始化控制器
        set_maxspeed(1,10000); //设置最大速度
        set_profile(1,200,5000,3000);     //设置梯形运动速度
        fast_pmove(1,100000);  //启动 1 轴正向移动 100000
……      //其它操作
        change_speed(1,6000);  //改变轴 1 工作速度为 6000，加速度为 3000
……
        change_speed(1,1000);  //改变轴 1 工作速度为 1000，加速度为 3000
……
}
```

1.13.3 动态改变目标位置

1.13.3.1 指令列表

表 1-32 动态改变目标位置函数

函数	返回值	说明
change_pos	0：正确 -1：错误	实现运动中动态改变目标相对位置的功能
change_pos_to	0：正确 -1：错误	实现运动中动态改变目标绝对位置的功能

1.13.3.2 功能说明

（1） change_pos 说明

函数说明：在相对位置模式下使用函数 change_pos 来动态改变目标位置。单轴点位运动过程中，在常速模式或梯形速度模式下，若用户发现发出的运动指令终点位置需要改变，可在该点位运动结束前或者结束后调用“change_pos”动态改变终点位置。系统将自动按新的终点位置运动。该终点位置以上一条用户发出的运动指令（fast_pmove、con_pmove）的起点为起点。可多次调用该函数来改变目标位置。

参数说明：

ch------轴号

pos------新的相对目标位置

函数返回值：0（成功）---- 1（失败）

指令类型：立即指令，调用后立即生效

（2） change_pos_to 说明

函数说明：在绝对位置模式下使用函数 **change_pos_to** 来动态改变目标位置。梯形速度模式下的单轴快速点位运动（**fast_pmove_to**）和常速单轴点位运动（con_pmove_to）过程中，若用户发现发出的运动指令终点位置需要改变，可在该点位运动结束前或者结束后调用“**change_pos_to**”动态改变终点位置。系统将自动按新的终点位置运动。该终点位置以整个绝对位置坐标系的起点为起点。可多次调用该函数来改变目标位置。

参数说明：

ch------轴号

pos------新的绝对目标位置

函数返回值：0（成功）---- 1（失败）

指令类型：立即指令，调用后立即生效

1.13.4 可掉电保护数据读写功能

1.13.4.1 指令列表

表 1-33 可掉电保护数据读写函数

函数	返回值	说明
write_password_flash	0：写成功； -1：参数错误导致的写失败； -2：校验密码错误导致的写失败。	带密码保护功能的数据区写函数
read_password_flash	0：读取数据成功； -1：参数错误导致的读失败； -2：校验密码错误导致的读失败。	带密码保护功能的数据区读函数
clear_password_flash	0：擦除数据成功； -1：参数错误导致的擦除失败； -2：校验密码错误导致的擦除失 败。	擦除带密码保护功能的数据区
write_flash	0：写成功；	一般数据区的写函数

	-1：参数错误导致的写失败；	
read_flash	0：读取数据成功； -1：参数错误导致的读失败；	一般数据区的读函数
clear_flash	0：擦除数据成功； -1：参数错误导致的擦除失败；	擦除一般数据区数据

1.13.4.2 功能说明

MPC2860 提供 1M bytes 掉电数据保护存储空间。1M bytes 分为 16 个逻辑片区，逻辑片区序号为 0~15，每个逻辑片区共 64K bytes 的存储空间。存储空间中存储的数据类型为 long 型。因此，每个逻辑片区又分为 64K/4 = 16K 个逻辑地址，序号 0 ~ 16383(16K -1)。掉电保护区如下表所示：

表 1-34 掉电数据保护区

	逻辑片区	逻辑地址
数据片区	15	序号：0~16383（共 16K）
	14	序号：0~16383（共 16K）
	13	序号：0~16383（共 16K）
	12	序号：0~16383（共 16K）
	11	序号：0~16383（共 16K）
	10	序号：0~16383（共 16K）
	9	序号：0~16383（共 16K）
	8	序号：0~16383（共 16K）
	7	序号：0~16383（共 16K）
	6	序号：0~16383（共 16K）
	5	序号：0~16383（共 16K）
	4	序号：0~16383（共 16K）
	3	序号：0~16383（共 16K）
	2	序号：0~16383（共 16K）
	1	序号：0~16383（共 16K）
密码片区	0	序号：0~16383（共 16K）

密码片区（片区 0）中最后一个逻辑地址(16383)对应的区域用来存储密码，出厂密码为 0xffffffff。对密码片区（片区 0）进行读写操作需要提供其 16383 号地址存储的 long 型密码。

（1） write_password_flash、 read_password_flash、 clear_password_flash 说明

用于对带密码保护功能的数据区进行写、读和擦除操作。对数据区进行写操作之前必须对该片区域进行擦除操作，否则无法写入待写数据。 擦除操作对整片有效， 可以对指定片区的区域进行擦除，使得该区域的 64K 字节空间所有位为“1”。 读写操作对 4 个字节的连续地址有效，所以进行擦除操作时请谨慎！

（2） write_flash、 read_flash、 clear _flash 说明

用于对一般数据区进行写、读和擦除操作。对数据区进行写操作之前必须对该片区域进行擦除操作，否则无法写入待写数据。擦除操作对整片有效，可以对指定片区的区域行擦除，使得该区域的 64K 字节空间所有位为“1”。读写操作对 4 个字节的连续地址有效，所以进行擦除操作时请谨慎！

例程：

```
void main( )
{
Long data;
auto_set( );  //检测控制器
init_board( );  //初始化控制器

clear_password_flash(1, 0xffffffff);  //擦除区域 0
write_password_flash(1,16383,0xccccaaaa,0xffffffff); //将读写密码修改为 0xccccaaaa
write_password_flash(1,1, 0x01, 0xccccaaaa); //对区域 0 的 1 号地址写入数据 0x01
read_password_flash(1,1,&data, 0xccccaaaa); //读取区域 0 的 1 号地址所保存的数值
}
```

1.13.5 多点位置比较输出

1.13.5.1 指令列表

表 1-35 多位置比较输出函数

函数	返回值	说明
start_comparePulse	0：正确 -1：错误	在 HSIO 口直接输出一个脉冲或者电平信号
start_compareData	0：正确 -1：失败 7：参数设置错误	启动非等间距位置比较输出并设置相关参数
start_compareLinear	0：正确 -1：失败	启动等间距重复位置比较输出并设置相关参数

	7：参数设置错误	
get_compareStatus	0：正确 -1：错误	查询已输出的位置比较数量
compare_stop	0：正确 -1：错误	取消位置比较输出

1.13.5.2 功能说明

在轴运动过程中（包括点位运动和插补运动，多轴运动时只与其中某一个轴进行位置比较输出），控制器自动将编码器位置或内部脉冲计数器位置与设定位置进行比较，当编码器位置或内部脉冲计数器位置到达设定位置时，在该轴对应的高速 IO（HSIO）口输出脉冲（差分输出）或反转当前的电平信号。位置比较输出电压为 5V(高电平为 3.5V～5V 均可)。注意：该功能中所述的“位置”均为相对位置，即相对于当前运动指令起点的位置。

每张 MPC2860 卡有两根轴具备多点位置比较输出功能。采用固定轴号和输出端口（HSIO）的方式。即 1 轴对应 HSIO1 口，2 轴对应 HSIO2 口，不需要配置。

注意：信号转接板 P100-03 的 OUT1 为 HSI01，OUT2 为 HSI02，这两个 OUT 口固定为位置比较输出口，不能再当作通用输出口使用。具体接线方式见用户手册《MPC2860 User.PDF》

（1）start_comparePulse 说明

在 HSIO 口直接输出一个脉冲或者电平信号，调用前检查是否已经启动位置比较输出，若正在进行位置比较输出，无法在 HSIO 口输出电平或脉冲。

（2）start_compareData 说明

启动非等间距位置比较输出并设置相关参数，使用数组方式进行输入，以当前位置为起始点，输入比较位置相对起始位置的距离数组。此时需保证位置数组中的数据为单调增加的正向距离值或单调减少的负向距离值。

（3）start_compareLinear 说明

启动等间距重复位置比较输出并设置相关参数。若比较位置为大量连续的等间距输出，则使用此函数，此时仅需输入起始比较位置、间隔距离以及重复比较次数。等间距方式只输出脉冲信号。

（4）get_compareStatus 说明

读取位置比较已输出的上升沿个数。若以电平方式输出，则返回值为通道 1 或通道 2 已输出的上升沿个数，但与已完成的位置比较个数不等（以电平方式输出时，已完成的位置比较个数为已输出上升沿和已输出下降沿个数的总和）；若以脉冲方式输出，则返回值为通道 1 或通道 2 已输出的上升沿个数，与已完成的位置比较个数相等

（5） compare_stop 说明

取消位置比较输出。在“非等间距位置比较输出”、“等间距位置比较输出”和“直接输出一个脉冲或者电平信号”三种模式之间进行切换时，必须先调用“**compare_stop**”取消当前输出模式。

1.13.6 板卡号和版本读取

1.13.6.1 指令列表

运动控制器提供 1 个板卡号读取函数和 3 个版本读取函数，如表所示。

表 1-36 板卡号和版本读取函数

函数	返回值	说明
check_IC	其它：卡号 -1：错误	查询用户设置的控制卡的本地 ID 号
get_lib_ver	0	查询函数库的版本
get_sys_ver	0：正确 -1：错误	查询驱动程序的版本
get_card_ver	0：正确 -1：错误	查询运动控制器固件版本

1.13.6.2 功能说明

（1）check_IC 说明

运动控制器上设计了一个旋钮开关，可以设定多块板卡共用时各板卡的本地 ID 号。旋钮开关最大设定值为 0xFH，目前只能设定为 0x0H～0x7H，只能支持 4 块板卡共用。使用函数“check_IC”可读取出板卡的本地 ID 号。使用“check_IC”时，计算机中只安装一张控制卡。

运动控制器出厂时，初始化本地 ID 号均为 0，如果要多卡共用，用户需要改变该设置。例如，若 4 卡共用，则需要将各板卡依次设置为 0、1、2、3。若设置的本地 ID 号有重复或为其它值，控制系统调用板卡初始化函数“auto_set”和“init_board”后，会提示初始化失败。调用“get_err”、“get_last_err”等函数可读取错误信息。

（2）get_lib_ver、get_sys_ver、get_card_ver 说明

分别用于读取运动控制器附带的函数库版本号、驱动程序版本号、板卡版本号。

函数返回后，版本号存放在函数的参数指针变量中。

1.13.7 错误代码处理函数

1.13.7.1 指令列表

运动控制器可返回最近 10 个错误状态，以便用户了解系统最近状况。错误代码操作函数有 3 个，如表 1-37 所示。

表 1-37 错误代码操作函数

函数	返回值	说明
get_err	0：正确 -1：错误	获得最近 10 个错误代码
get_last_err	0：正确 -1：错误	获得最近一个错误代码
reset_err	0	将所有（10 个）错误代码复位清零

1.13.7.2 功能说明

（1）get_err 说明

读取最近的 10 个中的错误信息。运动控制器提供了如下错误信息表。

表 1-38 初始化错误：

错误代码	**对应 Error Code**	**含义**
0x00000004	ERR_AUTO_SET_NOT	初始化未调用 auto_set()
0x00000006	ERR_WDM_COMM	与底层通信失败：未安装驱动或驱动未正确加载
0x00000007	ERR_INIT_NO_BOARD	检测不到板卡
0x00000008	ERR_INIT_NO_AXES	板卡上检测不到轴
0x00000009	ERR_INIT_BOARD_NOT	未进行初始化或初始化未成功
0x0000000a	ERR_INIT_TOO_BOARD	检测到的板卡数目超过允许的卡数；MPC2860 默认支持 4 卡共用
0x0000000b	ERR_INIT_TOO_AXES_CARD	检测到板卡上的轴数超过板卡设计的轴数；MPC2860 每卡 6 轴
0x0000000c	ERR_INIT_NO_NAMECARD	错误的板卡 ID 号
0x0000000d	ERR_INIT_TOO_NAMECARD	板卡 ID 号设置过大
0x0000000e	ERR_INIT_NAMECARD_NO1	使用单卡时，卡的本地 ID 没有设置为 0：将拨码开关拨到 0 即可
0x0000000f	ERR_INIT_NAMECARD_NO2	多卡共用时，第一张板卡的本地

		ID 不是 0
0x00000010	ERR_INIT_NAMECARD_NO3	多卡共用时，各板卡的本地 ID 号不是依次增大
0x00000011	ERR_INIT_LIB	函数库版本错误
0x00000012	ERR_INIT_SYS	驱动程序版本错误
0x00000013	ERR_INIT_CARD_TYPE	板卡类型出错

表 1-39 API 参数错误：

错误代码	对应 Error Code	含义
0x02000001	ERR_PARAM_CH	轴号设置错误
0x02000002	ERR_PARAM_SPEED	速度设置错误
0x02000005	ERR_PARAM_CARD	卡号设置错误
0x02000006	ERR_SET_HOME_MODE	回零模式设置错误
0x02000009	ERR_PARAM_FLASHNO	Flash 序号错误
0x0200000a	ERR_PARAM_BITNO	I/O 操作中 bit 位超出最大允许范围
0x0200000b	ERR_PARAM_FLASHPICE	Flash 片区序号错误
0x02000017	ERR_PARA_SAME_CH	多轴运动指令调用相同的轴号
0x02000018	ERR_PARAM_BACKLASH	反向间隙值小于 0
0x02000019	ERR_PARAM_VALUE	参数设置错误
0x0200001b	ERR_PARAM_CH_FUN	轴功能冲突
0x0200001c	ERR_PARAM_CARD_FUN	卡功能冲突
0x0200001d	ERR_PARAM_FUN	系统属性功能冲突
0x0200001e	ERR_BEFOREMOTION	运动指令调用的轴正处在运动状态
0x0200001f	ERR_CHECK_RANGE	运动位移设置错误

（2）get_last_err 说明

返回最近一次错误代码。

（3）reset_err 说明

将 10 个错误代码变量清零。

2 函数库

2.1 函数列表

2-1 函数列表

	函数名	描述

控制器初始化	auto_set	自动检测和自动设置控制器
	init_board	对控制器硬件和软件初始化
轴属性设置	set_outmode	设置轴的脉冲输出模式
	set_home_mode	设置轴的回原点模式
	set_dir	设置方向信号电平状态
	enable_el	使能或禁止限位信号
	enable_org	使能或禁止原点信号
	enable_alm	使能或禁止轴报警信号
	set_el_logic	设置限位信号有效电平
	set_org_logic	设置原点信号有效电平
	set_alm_logic	设置轴报警信号有效电平
板卡报警信号设置及查询	enable_card_alm	使能或禁止卡报警信号
	set_card_alm_logic	设置卡报警有效电平
	check_card_alarm	使能板卡报警信号后检测板卡报警信号有效状态
编码器设置	set_encoder_mode	设置编码器反馈模式
	set_encoder	设置编码器位置
	get_encoder	获取编码器位置
速度参数设置	set_maxspeed	设置轴的最大速度
	set_conspeed	设置轴在常速运动方式下的速度参数
	set_s_curve	设置控制轴快速运动模式
	set_profile	设定轴在快速运动方式下的速度参数
	set_vector_conspeed	设置常速运动方式下的矢量常速度参数
	set_vector_profile	设置快速运动方式下的矢量梯形速度参数
	set_s_section	设置 S 型升降速加加速度
点位运动	con_pmove	1 个轴以常速做相对位置点位运动
	con_pmove2	2 个轴以常速做相对位置点位运动
	con_pmove3	3 个轴以常速做相对位置点位运动
	con_pmove4	4 个轴以常速做相对位置点位运动
	con_pmove_to	1 个轴以常速做绝对位置点位运动
	fast_pmove	1 个轴以快速做相对位置点位运动
	fast_pmove2	2 个轴以快速做相对位置点位运动
	fast_pmove3	3 个轴以快速做相对位置点位运动
	fast_pmove4	4 个轴以快速做相对位置点位运动
	fast_pmove_to	1 个轴以快速做绝对位置点位运动
连续运动	con_vmove	1 个轴以常速做连续运动
	con_vmove2	2 个轴以常速做连续运动
	con_vmove3	3 个轴以常速做连续运动
	con_vmove4	4 个轴以常速做连续运动
	fast_vmove	1 个轴以快速做连续运动
	fast_vmove2	2 个轴以快速做连续运动
	fast_vmove3	3 个轴以快速做连续运动
	fast_vmove4	4 个轴以快速做连续运动
回原点运动	con_hmove	1 个轴以常速做回零运动
	con_hmove2	2 个轴以常速做回零运动
	con_hmove3	3 个轴以常速做回零运动

	con_hmove4	4 个轴以常速做回零运动
	fast_hmove	1 个轴以快速做回零运动
	fast_hmove2	2 个轴以快速做回零运动
	fast_hmove3	3 个轴以快速做回零运动
	fast_hmove4	4 个轴以快速做回零运动
线性插补运动函数	con_line2	两个轴作常速直线插补运动(相对位置)
	con_line3	三个轴作常速直线插补运动(相对位置)
	con_line4	四个轴作常速直线插补运动(相对位置)
	con_line2_to	两个轴作常速直线插补运动(绝对位置)
	con_line3_to	三个轴作常速直线插补运动(绝对位置)
	con_line4_to	四个轴作常速直线插补运动(绝对位置)
	fast_line2	两个轴作快速直线插补运动(相对位置)
	fast_line3	三个轴作快速直线插补运动(相对位置)
	fast_line4	四个轴作快速直线插补运动(相对位置)
	fast_line2_to	两个轴作快速直线插补运动(绝对位置)
	fast_line3_to	三个轴作快速直线插补运动(绝对位置)
	fast_line4_to	四个轴作快速直线插补运动(绝对位置)
制动函数	sudden_stop	立即制动一个运动轴
	sudden_stop2	立即制动两个运动轴
	sudden_stop3	立即制动三个运动轴
	sudden_stop4	立即制动四个运动轴
	decel_stop	光滑制动一个运动轴
	decel_stop2	光滑制动两个运动轴
	decel_stop3	光滑制动三个运动轴
	decel_stop4	光滑制动四个运动轴
	move_pause	暂停一个运动轴
	move_resume	恢复一个轴的运动
位置设置和读取函数	set_abs_pos	设置一个轴的绝对位置值
	reset_pos	复位一个轴的当前位置值为零
	get_abs_pos	读取一个轴的绝对位置值
运动状态查询函数	check_status	读取指定轴的状态
	get_cur_dir	读取指定轴的当前运动方向
	check_done	检查指定轴的运动是否已经完毕
	get_rate	读取当前运动的速度
	get_max_axe	读取总轴数
	get_board_num	获取计算机内运动控制卡总卡数
	get_axe	获取制定板卡上的轴数
	get_conspeed	获取指定轴设定的做常速点位运动时的速度
	get_vector_conspeed	获取指定轴数设定的做常速插补运动时的矢量速度
	get_profile	获取指定轴设定的做快速点位运动时的梯形速度参数
	get_vector_profile	获取指定轴数设定的做快速插补运动时的矢量梯形速度参数
专用输入检测函数	check_limit	检查指定轴是否到达限位开关位置
	check_home	检查指定轴是否到达原点开关位置

	check_alarm	检查指定轴是否到达报警开关位置
	check_sfr	读取专用输入口所有的开关量状态
	check_sfr_bit	读取专用输入口某位的开关量状态
伺服专用函数	checkin_axis_INP	获取伺服到位信号输入
	checkin_axis_SRDY	获取伺服准备好信号输入
	outport_bit_RST	伺服报警清除输出
	outport_bit_CL	伺服偏差报警计数清除输出
	outport_bit_SEVERON	伺服使能输出
通用 IO 口操作函数	checkin_byte	读取扩展输入口所有的开关量状态
	checkin_bit	读取扩展输入口某位的开关量状态
	outport_byte	设置主信号板上通用输出口各位的开关量状态
	outport_byte_ex	设置扩展 IO 板 EA1616C 上的通用输出口状态
	outport_bit	设置通用输出口某位的开关量状态，包括主信号板和 IO 扩展板上
反向间隙处理	set_backlash	设置由于机构换向形成间隙的补偿值
	start_backlash	开始补偿由于机构换向间隙而导致的位置误差
	end_backlash	停止补偿由于机构换向间隙而导致的位置误差
运动中变速度	change_speed	实现运动中变速的功能
动态改变目标位置	change_pos	实现运动中动态改变目标相对位置的功能
	change_pos_to	实现运动中动态改变目标绝对位置的功能
可掉电保护数据读写功能	write_password_flash	带密码保护功能的数据区写函数
	read_password_flash	带密码保护功能的数据区读函数
	clear_password_flash	擦除带密码保护功能的数据区
	write_flash	一般数据区的写函数
	read_flash	一般数据区的读函数
	clear_flash	擦除一般数据区数据
多点位置比较输出	start_comparePulse	在 HSIO 口直接输出一个脉冲或者电平信号
	start_compareData	启动非等间距位置比较输出并设置相关参数
	start_compareLinear	启动等间距重复位置比较输出并设置相关参数
	get_compareStatus	查询已输出的位置比较数量
	compare_stop	取消位置比较输出
板卡号和版本读取	check_IC	查询用户设置的控制卡的本地 ID 号
	get_lib_ver	查询函数库的版本
	get_sys_ver	查询驱动程序的版本
	get_card_ver	查询运动控制器固件版本
错误代码处理函数	get_err	获得最近 10 个错误代码
	get_last_err	获得最近一个错误代码
	reset_err	将所有（10 个）错误代码复位清零

2.2 函数说明

2.2.1 控制器初始化

- int auto_set(void)

  功能： 板卡初始化，第一个调用

返回值： 总轴数（正确）----- 负值（错误）

- int init_board(void)
  功能： 板卡初始化，第二个调用
  返回值： 总卡数（正确）----- 负值（错误）

2.2.2 轴属性设置

- int set_outmode(int ch, int mode, int logic)
  功能： 设置脉冲输出模式（Pul/Dir、CW/CCW）

  参数：

  ch ----- 轴号

  mode ---- 模式： 0 (CW/CCW) – 1 (Pul/Dir)

  logic ---- 暂不使用，设置为 0

  返回值： 0（正确）----- -1（错误）

- int set_home_mode(int ch, int origin_mode)
  功能： 设置回零模式

  参数：

  ch ---- 轴号

  mode ---- 模式

  0 ---- 指定轴 Org 信号有效立即停止

  1 ---- 指定轴以梯形速度运动时，遇到轴 Org 信号有效时立即减速，到达梯形低速时停止

  2 ---- 检测到指定轴编码器 Z 信号输入时立即停止

  3 ---- 指定轴以梯形速度运动时，遇到轴 Org 信号立即减速至低速，遇到轴编码器 Z 信号立即停止

  返回值： 0（正确）----- -1（错误）

- int set_dir(int ch, int iLogic)
  功能： 设置方向信号的电平状态
  参数：
  ch ----- 轴号
  iLogic ---- 方向： 1 （正向） -- -1 （负向）
  返回值： 0（正确）----- -1（错误）

- int enable_el(int ch, int flag)
  功能： 使能轴限位信号，不使能时，可以将其当作普通输入口来使用，

  参数：
  ch ----- 轴号
  flag ---- 使能标志： 1 （使能） -- 0 （不使能）
  返回值： 0（正确）----- -1（错误）

- int enable_org(int ch, int flag)
  功能： 使能轴 ORG 信号，不使能时，可以将其当作普通输入口来使用
  参数：
  ch ----- 轴号
  flag ---- 使能标志： 1 （使能） -- 0 （不使能）
  返回值： 0（正确）----- -1（错误）

- int enable_alm(int ch, int flag)
  功能： 使能轴报警信号，不使能时，可以将其当作普通输入口来使用
  参数：
  ch ----- 轴号
  flag ---- 使能标志： 1 （使能）-- 0 （不使能）
  返回值： 0（正确）----- -1（错误）

- int set_el_logic(int ch, int flag)
  功能：设置轴限位信号有效电平
  参数：
  ch ----- 轴号
  flag ---- 有效电平： 1 （高电平） -- 0 （低电平）
  返回值： 0（正确）----- -1（错误）

- int set_org_logic(int ch, int flag)
  功能：设置轴 ORG 信号有效电平
  参数：
  ch ----- 轴号
  flag ---- 有效电平： 1 （高电平） -- 0 （低电平）
  返回值： 0（正确）----- -1（错误）

- int set_alm_logic(int ch, int flag)
  功能：设置轴报警信号有效电平
  参数：
  ch ----- 轴号
  flag ---- 有效电平： 1 （高电平） -- 0 （低电平
  返回值： 0（正确）----- -1（错误）

2.2.3 板卡报警信号设置及查询

- int enable_card_alm(int cardno, int flag)
  功能： 使能卡报警信号，不使能时，可以将其当作普通输入口来使用
  参数：
  cardno ----- 卡号
  flag ---- 使能标志： 1 （使能） -- 0 （不使能）
  返回值： 0（正确）----- -1（错误）

- int set_card_alm_logic(int cardno, int flag)
  功能：设置卡报警信号有效电平
  参数：
  cardno ----- 卡号
  flag ---- 有效电平： 1 （高电平） -- 0 （低电平）
  返回值： 0（正确）----- -1（错误）

- int check_card_alarm(int cardno)
  功能：使能板卡报警信号后检测板卡报警信号有效状态
  参数：
  cardno ----- 卡号
  返回值： 0（无效）----- 1（有效）---- -3（出错）

2.2.4 编码器设置

- int set_encoder_mode(long ch,long mode,long multip,long count_unit)
  功能：设置编码器模式
  参数：
  ch ----轴号
  mode ---- 信号模式： 1 （增减脉冲) -- 0 (A/B 90 度相位差)
  multip ---- 倍频数：1 或 4
  count_unit ---- 暂不使用，设置为 0
  返回值： 0（正确）----- -1（错误）

- int set_encoder(int ch,long encodercount)
  功能：设置编码器位置
  参数：
  ch ---- 轴号
  encodercount----设置的编码器绝对位置值
  返回值：0（正确）----- -1（错误）

- int get_encoder(int ch,long *count)
  功能：获取编码器位置

参数：
ch ---- 轴号
count---保存编码器绝对位置值的长整形数据指针。
返回值：0（正确）----- -1（错误）

2.2.5 速度参数设置

- int set_maxspeed(int ch, double speed)
  功能：设置轴最大运行速度，控制运行时速度的精度
  参数：
  ch ---- 轴号
  speed ---- 最大速度（正值），范围 10~4M pps
  返回值： 0（正确）----- -1（错误）

- int set_conspeed(int ch, double conspeed)
  功能：设置轴常速运行速度
  参数：
  ch ----- 轴号
  speed ---- 常速速度（正值），范围 10~4M pps
  返回值： 0（正确）----- -1（错误）

- int set_s_curve(int ch, int mode)
  功能：设置快速运动曲线模式
  参数：
  ch ---- 轴号
  mode ---- 0：梯形加减速 1：S 型加减速
  返回值： 0（正确）----- -1（错误）

- int set_profile(int ch, double vl, double vh, double ad, double dc)
  功能：设置轴快速运动参数
  参数：
  ch ---- 轴号
  vl ---- 快速运动低速（正值），范围：10~4M pps
  vh ---- 快速运动高速（正值），范围 :10≤vl≤vh≤4M pps
  ad ---- 上升加速度（正值）, 范围 :10≤ad≤12.5M pps
  dc ---- 下降加速度（正值）, 范围 :10≤dc≤12.5M pps
  返回值： 0（正确）----- -1（错误）

- int set_s_section(int ch, double accel_sec, double decel_sec)
  功能：设置轴 S 型曲线运动加加速度。“set_profile”设置 S 形的初速、高速、最大加速度和最大减速度，但是 S 曲线的最大加速度和最大减速度要求相等。“set_s_section”在“set_profile”之后调用，设置 S 形曲线运动的加加速度和加减速度。S 曲线加速度运动只能在快速点位运动中有效，插补运动不提供 S 曲线加减速模式。

参数：

ch ---- 轴号

accel_sec ---- 加加速段和减加速段的加加速度值，范围 10~375M ；

decel_sec ---- 加减速段和减减速段的加减速度值，和 accel_sec 值相同，否则调用会失败

返回值： 0（正确）----- -1（错误）

- int set_vector_conspeed(double conspeed)

功能：设置常速插补运动运行速度（矢量速度）

参数：

speed ---- 常速速度（正值），范围 10~4M pps

返回值： 0（成功）----- -1（错误）

- int set_vector_profile(double vec_vl, double vec_vh, double vec_ad, double vec_dc)

功能：设置快速插补运动参数（矢量速度）

参数：

vec_vl ---- 快速运动低速（正值），范围 10~4M pps

vec_vh ---- 快速运动高速（正值），范围 10~4M pps

vec_ad ---- 上升加速度（正值）, 范围 10~12.5M pps

vec_dc ---- 下降加速度（正值）, 范围 10~12.5M pps

返回值：0（成功）----- -1（错误）

2.2.6 点位运动

- int con_pmove(int ch, double step)

功能：单轴常速点位运动指令

参数：

ch ---- 轴号

step ---- 相对当前位置的位移（正值为正向，负值为负向）

返回值： 0（正确）----- -1（错误）

- int con_pmove_to(int ch, double step)

功能：单轴常速点位绝对位置运动指令

参数：

ch ---- 轴号

step ---- 绝对位置值

返回值： 0（正确）----- -1（错误）

- int con_pmove2(int ch1, double step1, int ch2, double step2)

功能：两轴常速点位运动指令

参数：

ch1 ---- 轴号

step1 ---- 相对当前位置的位移（正值为正向，负值为负向）

ch2 ---- 轴号

step2 ---- 相对当前位置的位移（正值为正向，负值为负向）
返回值： 0（正确）----- -1（错误）

- int con_pmove3(int ch1, double step1, int ch2, double step2, int ch3, double step3)
  功能：三轴常速点位运动指令
  参数：
  ch1 ---- 轴号
  step1 ---- 相对当前位置的位移（正值为正向，负值为负向）
  ch2 ---- 轴号
  step2 ---- 相对当前位置的位移（正值为正向，负值为负向）
  ch3 ---- 轴号
  step3 ---- 相对当前位置的位移（正值为正向，负值为负向）
  返回值： 0（正确）----- -1（错误）

- int con_pmove4(int ch1, double step1, int ch2, double step2, int ch3, double step3, int ch4, double step4)
  功能：四轴常速点位运动指令
  参数：
  ch1 ---- 轴号
  step1 ---- 相对当前位置的位移（正值为正向，负值为负向）
  ch2 ---- 轴号
  step2 ---- 相对当前位置的位移（正值为正向，负值为负向）
  ch3 ---- 轴号
  step3 ---- 相对当前位置的位移（正值为正向，负值为负向）
  ch4 ---- 轴号
  step4 ---- 相对当前位置的位移（正值为正向，负值为负向）
  返回值： 0（正确）----- -1（错误）

- int fast_pmove(int ch, double step)
  功能：单轴快速点位运动指令
  参数：
  ch ---- 轴号
  step ---- 相对当前位置的位移（正值为正向，负值为负向）
  返回值： 0（正确）----- -1（错误）

- int fast_pmove_to(int ch, double step)
  功能：单轴快速点位绝对位置运动指令
  参数：
  ch ---- 轴号
  step ---- 绝对位置值
  返回值： 0（正确）----- -1（错误）

- int fast_pmove2(int ch1, double step1, int ch2, double step2)
  功能：两轴快速点位运动指令

参数：
ch1 ---- 轴号
step1 ---- 相对当前位置的位移（正值为正向，负值为负向）
ch2 ---- 轴号
step2 ---- 相对当前位置的位移（正值为正向，负值为负向）
返回值： 0（正确）----- -1（错误）

- int fast_pmove3(int ch1, double step1, int ch2, double step2, int ch3, double step3)
功能：三轴快速点位运动指令
参数：
ch1 ---- 轴号
step1 ---- 相对当前位置的位移（正值为正向，负值为负向）
ch2 ---- 轴号
step2 ---- 相对当前位置的位移（正值为正向，负值为负向）
ch3 ---- 轴号
step3 ---- 相对当前位置的位移（正值为正向，负值为负向）
返回值： 0（正确）----- -1（错误）

- int fast_pmove4(int ch1, double step1, int ch2, double step2, int ch3, double step3, int ch4, double step4)
功能：四轴快速点位运动指令
参数：
ch1 ---- 轴号
step1 ---- 相对当前位置的位移（正值为正向，负值为负向）
ch2 ---- 轴号
step2 ---- 相对当前位置的位移（正值为正向，负值为负向）
ch3 ---- 轴号
step3 ---- 相对当前位置的位移（正值为正向，负值为负向）
ch4 ---- 轴号
step4 ---- 相对当前位置的位移（正值为正向，负值为负向）
返回值： 0（正确）----- -1（错误）

2.2.7 连续运动

- int con_vmove(int ch, int dir)
功能：单轴常速连续运动指令
参数：
ch ---- 轴号
dir ---- 方向（ 1：正向 -1：负向）
返回值： 0（正确）----- -1（错误）

- int con_vmove2(int ch1, int dir1, int ch2, int dir2)
功能：两轴常速连续运动指令
参数：

ch1 ---- 轴号
dir1 ---- 方向（ 1：正向 -1：负向）
ch2 ---- 轴号
dir2 ---- 方向（ 1：正向 -1：负向）
返回值： 0（正确）----- -1（错误）

- int con_vmove3(int ch1, int dir1, int ch2, int dir2, int ch3, int dir3)
  功能：三轴常速连续运动指令
  参数：
  ch1 ---- 轴号
  dir1 ---- 方向（ 1：正向 -1：负向）
  ch2 ---- 轴号
  dir2 ---- 方向（ 1：正向 -1：负向）
  ch3 ---- 轴号
  dir3 ---- 方向（ 1：正向 -1：负向）
  返回值： 0（正确）----- -1（错误）

- int con_vmove4(int ch1, int dir1, int ch2, int dir2, int ch3, int dir3, int ch4, int dir4)
  功能：四轴常速连续运动指令
  参数：
  ch1 ---- 轴号
  dir1 ---- 方向（ 1：正向 -1：负向）
  ch2 ---- 轴号
  dir2 ---- 方向（ 1：正向 -1：负向）
  ch3 ---- 轴号
  dir3 ---- 方向（ 1：正向 -1：负向）
  ch4 ---- 轴号
  dir4 ---- 方向（ 1：正向 -1：负向）
  返回值： 0（正确）----- -1（错误）

- int fast_vmove(int ch, int dir)
  功能：单轴快速连续运动指令
  参数：
  ch ---- 轴号
  dir ---- 方向（ 1：正向 -1：负向）
  返回值： 0（正确）----- -1（错误）

- int fast_vmove2(int ch1, int dir1, int ch2, int dir2)
  功能：两轴快速连续运动指令
  参数：
  ch1 ---- 轴号
  dir1 ---- 方向（ 1：正向 -1：负向）
  ch2 ---- 轴号
  dir2 ---- 方向（ 1：正向 -1：负向）

返回值： 0（正确）----- -1（错误）

- int fast_vmove3(int ch1, int dir1, int ch2, int dir2, int ch3, int dir3)
  功能：三轴快速连续运动指令
  参数：
  ch1 ---- 轴号
  dir1 ---- 方向（ 1：正向 -1：负向）
  ch2 ---- 轴号
  dir2 ---- 方向（ 1：正向 -1：负向）
  ch3 ---- 轴号
  dir3 ---- 方向（ 1：正向 -1：负向）
  返回值： 0（正确）----- -1（错误）

- int fast_vmove4(int ch1, int dir1, int ch2, int dir2, int ch3, int dir3, int ch4, int dir4)
  功能：四轴快速连续运动指令
  参数：
  ch1 ---- 轴号
  dir1 ---- 方向（ 1：正向 -1：负向）
  ch2 ---- 轴号
  dir2 ---- 方向（ 1：正向 -1：负向）
  ch3 ---- 轴号
  dir3 ---- 方向（ 1：正向 -1：负向）
  ch4 ---- 轴号
  dir4 ---- 方向（ 1：正向 -1：负向）
  返回值： 0（正确）----- -1（错误）

2.2.8 回原点运动

- int con_hmove(int ch, int dir1)
  功能：单轴常速回零运动指令
  参数：
  ch ---- 轴号
  dir ---- 方向（ 1：正向 -1：负向）
  返回值： 0（正确）----- -1（错误）

- int con_hmove2(int ch1, int dir1, int ch2, int dir2)
  功能：两轴常速回零运动指令
  参数：
  ch1 ---- 轴号
  dir1 ---- 方向（ 1：正向 -1：负向）
  ch2 ---- 轴号
  dir2 ---- 方向（ 1：正向 -1：负向）
  返回值： 0（正确）----- -1（错误）

- int con_hmove3(int ch1, int dir1, int ch2, int dir2, int ch3, int dir3)
  功能：三轴常速回零运动指令
  参数：
  ch1 ---- 轴号
  dir1 ---- 方向（ 1：正向 -1：负向）
  ch2 ---- 轴号
  dir2 ---- 方向（ 1：正向 -1：负向）
  ch3 ---- 轴号
  dir3 ---- 方向（ 1：正向 -1：负向）
  返回值： 0（正确）----- -1（错误）

- int con_hmove4(int ch1, int dir1, int ch2, int dir2, int ch3, int dir3, int ch4, int dir4)
  功能：四轴常速回零运动指令
  参数：
  ch1 ---- 轴号
  dir1 ---- 方向（ 1：正向 -1：负向）
  ch2 ---- 轴号
  dir2 ---- 方向（ 1：正向 -1：负向）
  ch3 ---- 轴号
  dir3 ---- 方向（ 1：正向 -1：负向）
  ch4 ---- 轴号
  dir4 ---- 方向（ 1：正向 -1：负向）
  返回值： 0（正确）----- -1（错误）

- int fast_hmove(int ch, int dir)
  功能：单轴快速回零运动指令
  参数：
  ch ---- 轴号
  dir ---- 方向（ 1：正向 -1：负向）
  返回值： 0（正确）----- -1（错误）

- int fast_hmove2(int ch1, int dir1, int ch2, int dir2)
  功能：两轴快速回零运动指令
  参数：
  ch1 ---- 轴号
  dir1 ---- 方向（ 1：正向 -1：负向）
  ch2 ---- 轴号
  dir2 ---- 方向（ 1：正向 -1：负向）
  返回值： 0（正确）----- -1（错误）

- int fast_hmove3(int ch1, int dir1, int ch2, int dir2, int ch3, int dir3)
  功能：三轴快速回零运动指令
  参数：

ch1 ---- 轴号
dir1 ---- 方向（ 1：正向 -1：负向）
ch2 ---- 轴号
dir2 ---- 方向（ 1：正向 -1：负向）
ch3 ---- 轴号
dir3 ---- 方向（ 1：正向 -1：负向）
返回值： 0（正确）----- -1（错误）

- int fast_hmove4(int ch1, int dir1, int ch2, int dir2, int ch3, int dir3, int ch4, int dir4)
  功能：四轴快速回零运动指令
  参数：
  ch1 ---- 轴号
  dir1 ---- 方向（ 1：正向 -1：负向）
  ch2 ---- 轴号
  dir2 ---- 方向（ 1：正向 -1：负向）
  ch3 ---- 轴号
  dir3 ---- 方向（ 1：正向 -1：负向）
  ch4 ---- 轴号
  dir4 ---- 方向（ 1：正向 -1：负向）
  返回值： 0（正确）----- -1（错误）

2.2.9 线性插补运动函数

- int con_line2(int ch1, double step1, int ch2, double step2)
  功能：两轴常速直线插补运动指令
  参数：
  ch1 ---- 轴号
  step1 ---- 相对当前位置的位移（正值为正向，负值为负向）
  ch2 ---- 轴号
  step2 ---- 相对当前位置的位移（正值为正向，负值为负向）
  返回值：0（成功）----- -1（错误）

- int con_line3(int ch1, double step1, int ch2, double step2, int ch3, double step3)
  功能：三轴常速直线插补运动指令
  参数：
  ch1 ---- 轴号
  step1 ---- 相对当前位置的位移（正值为正向，负值为负向）
  ch2 ---- 轴号
  step2 ---- 相对当前位置的位移（正值为正向，负值为负向）
  ch3 ---- 轴号
  step3 ---- 相对当前位置的位移（正值为正向，负值为负向）
  返回值：0（成功）----- -1（错误）

- int con_line4(int ch1, double step1, int ch2, double step2, int ch3, double step3, int ch4, double step4)
  功能：四轴常速直线插补运动指令
  参数：
  ch1 ---- 轴号
  step1 ---- 相对当前位置的位移（正值为正向，负值为负向）
  ch2 ---- 轴号
  step2 ---- 相对当前位置的位移（正值为正向，负值为负向）
  ch3 ---- 轴号
  step3 ---- 相对当前位置的位移（正值为正向，负值为负向）
  ch4 ---- 轴号
  step4 ---- 相对当前位置的位移（正值为正向，负值为负向）
  返回值：0（成功）----- -1（错误）

- int con_line2_to(int ch1, double step1, int ch2, double step2)
  功能：两轴常速直线插补运动指令
  参数：
  ch1 ---- 轴号
  step1 ---- 相对零点位置的位移（绝对位置）
  ch2 ---- 轴号
  step2 ---- 相对零点位置的位移（绝对位置）
  返回值：0（成功）----- -1（错误）

- int con_line3_to(int ch1, double step1, int ch2, double step2, int ch3, double step3)
  功能：三轴常速直线插补运动指令
  参数：
  ch1 ---- 轴号
  step1 ----相对零点位置的位移（绝对位置）
  ch2 ---- 轴号
  step2 ----相对零点位置的位移（绝对位置）
  ch3 ---- 轴号
  step3 ----相对零点位置的位移（绝对位置）
  返回值：0（成功）----- -1（错误）

- int con_line4_to(int ch1, double step1, int ch2, double step2, int ch3, double step3, int ch4, double step4)
  功能：四轴常速直线插补运动指令
  参数：
  ch1 ---- 轴号
  step1 ----相对零点位置的位移（绝对位置）
  ch2 ---- 轴号
  step2 ----相对零点位置的位移（绝对位置）

ch3 ---- 轴号
step3 ----相对零点位置的位移（绝对位置）
ch4 ---- 轴号
step4 ----相对零点位置的位移（绝对位置）
返回值：0（成功）----- -1（错误）

- int fast_line2(int ch1, double step1, int ch2, double step2)
  功能：两轴快速直线插补运动指令
  参数：
  ch1 ---- 轴号
  step1 ---- 相对当前位置的位移（正值为正向，负值为负向）
  ch2 ---- 轴号
  step2 ---- 相对当前位置的位移（正值为正向，负值为负向）
  返回值：0（成功）----- -1（错误）

- int fast_line3(int ch1, double step1, int ch2, double step2, int ch3, double step3)
  功能：三轴快速直线插补运动指令
  参数：
  ch1 ---- 轴号
  step1 ---- 相对当前位置的位移（正值为正向，负值为负向）
  ch2 ---- 轴号
  step2 ---- 相对当前位置的位移（正值为正向，负值为负向）
  ch3 ---- 轴号
  step3 ---- 相对当前位置的位移（正值为正向，负值为负向）
  返回值： 0（成功）----- -1（错误）

- int fast_line4(int ch1, double step1, int ch2, double step2, int ch3, double step3, int ch4, double step4)
  功能：四轴快速直线插补运动指令
  参数：
  ch1 ---- 轴号
  step1 ---- 相对当前位置的位移（正值为正向，负值为负向）
  ch2 ---- 轴号
  step2 ---- 相对当前位置的位移（正值为正向，负值为负向）
  ch3 ---- 轴号
  step3 ---- 相对当前位置的位移（正值为正向，负值为负向）
  ch4 ---- 轴号
  step4 ---- 相对当前位置的位移（正值为正向，负值为负向）
  返回值：0（成功）----- -1（错误）

- int fast_line2_to(int ch1, double step1, int ch2, double step2)
  功能：两轴快速直线插补运动指令
  参数：

ch1 ---- 轴号
step1 ---- 相对零点位置的位移（绝对位置）
ch2 ---- 轴号
step2 ----相对零点位置的位移（绝对位置）
返回值：0（成功）----- -1（错误）

- int fast_line3_to(int ch1, double step1, int ch2, double step2, int ch3, double step3)
  功能：三轴快速直线插补运动指令
  参数：
  ch1 ---- 轴号
  step1 ----相对零点位置的位移（绝对位置）
  ch2 ---- 轴号
  step2 ----相对零点位置的位移（绝对位置）
  ch3 ---- 轴号
  step3 ----相对零点位置的位移（绝对位置）
  返回值： 0（成功）----- -1（错误）

- int fast_line4_to(int ch1, double step1, int ch2, double step2, int ch3, double step3, int ch4, double step4)
  功能：四轴快速直线插补运动指令
  参数：
  ch1 ---- 轴号
  step1 ----相对零点位置的位移（绝对位置）
  ch2 ---- 轴号
  step2 ----相对零点位置的位移（绝对位置）
  ch3 ---- 轴号
  step3 ----相对零点位置的位移（绝对位置）
  ch4 ---- 轴号
  step4 ----相对零点位置的位移（绝对位置）
返回值：0（成功）----- -1（错误）

2.2.10 制动函数

- int sudden_stop(int ch)
  功能：单轴急停
  参数：
  ch ---- 轴号
  返回值： 0（正确）----- -1（错误）

- int sudden_stop2(int ch1, int ch2)
  功能：两轴急停
  参数：
  ch1 ---- 轴号

ch2 ---- 轴号
返回值： 0（正确）----- -1（错误）

- int sudden_stop3(int ch1, int ch2, int ch3)
  功能：三轴急停
  参数：
  ch1 ---- 轴号
  ch2 ---- 轴号
  ch3 ---- 轴号
  返回值： 0（正确）----- -1（错误）

- int sudden_stop4(int ch1, int ch2, int ch3, int ch4)
  功能：四轴急停
  参数：
  ch1 ---- 轴号
  ch2 ---- 轴号
  ch3 ---- 轴号
  ch4 ---- 轴号
  返回值： 0（正确）----- -1（错误）

- int decel_stop(int ch)
  功能：快速运动模式下单轴缓停
  参数：
  ch ---- 轴号
  返回值： 0（正确）----- -1（错误）

- int decel_stop2(int ch1, int ch2)
  功能：快速运动模式下二轴缓停
  参数：
  ch1 ---- 轴号
  ch2 ---- 轴号
  返回值： 0（正确）----- -1（错误）

- int decel_stop3(int ch1, int ch2, int ch3)
  功能：快速运动模式下三轴缓停
  参数：
  ch1 ---- 轴号
  ch2 ---- 轴号
  ch3 ---- 轴号
  返回值： 0（正确）----- -1（错误）

- int decel_stop4(int ch1, int ch2, int ch3, int ch4)
  功能：快速运动模式下四轴缓停
  参数：

ch1 ---- 轴号
ch2 ---- 轴号
ch3 ---- 轴号
ch4 ---- 轴号
返回值： 0（正确）----- -1（错误）

- int move_pause(int ch)
  功能：运动暂停
  参数：
  ch ---- 轴号
  返回值： 0（正确）----- -1（错误）

- int move_resume(int ch)
  功能：暂停后恢复运动
  参数：
  ch ---- 轴号
  返回值： 0（正确）----- -1（错误）

2.2.11 位置设置和读取函数

- int set_abs_pos(int ch, double pos)
  功能：设置轴当前绝对位置
  参数：
  ch ---- 轴号
  pos ---- 绝对位置值（正值）
  返回值： 0（正确）----- -1（错误）

- int reset_pos(int ch)
  功能：将轴的绝对位置和相对位置清零
  参数：
  ch ---- 轴号
  返回值： 0（正确）----- -1（错误）

- int get_abs_pos(int ch, double *pos)
  功能：获取轴当前绝对位置值
  参数：
  ch ---- 轴号
  pos ---- 绝对位置
  返回值： 0（成功）----- -1（错误）

2.2.12 运动状态查询函数

- int check_status(int ch)
  功能：获取轴当前状态（ORG、EL、ALM 等有效状态）
  参数：
  ch ---- 轴号
  返回值： 正值（成功）----- -1（错误）

- int get_cur_dir(int ch)
  功能：获取轴当前方向
  参数：
  ch ---- 轴号
  返回值：-2（错误）---- -1（负向）----- 0（停止） ---- 1（正向）

- int check_done(int ch)
  功能：获取轴当前运动状态。在调用下一条运动指令时，必须先调用此 API 确认当前轴已经处于停止状态，否则不执行后续运动指令
  参数：
  ch ---- 轴号
  返回值： 0（停止）----- 1（运动）---- -1（出错）

- double get_rate(int ch)
  功能：获取轴运动速度
  参数：
  ch ---- 轴号
  返回值：0（成功）----- -1（错误）

- int get_max_axe()
  功能：获取总轴数
  返回值： 轴数（成功）----- 0（错误）

- int get_board_num()
  功能：获取总卡数
  返回值： 轴数（成功）----- 0（错误）

- int get_axe(int cardno)
  功能：获取指定板卡上轴数
  参数：
  cardno ---- 板卡
  返回值： 轴数（成功）----- 0（错误）

- double get_conspeed(int ch)
  功能：获取轴常速点位运动速度
  参数：
  ch ---- 轴号
  返回值： 正值速度（成功）----- -1（错误）

- int get_profile(int ch, double *vl, double *vh, double *ad, double *dc)
  功能：获取轴当前快速运动速度
  参数：
  ch ---- 轴号
  vl ---- 低速
  vh ---- 高速
  vec_ad ---- 上升加速度
  vec_dc ---- 下降加速度
  返回值： 0（成功）----- -1（错误）

- double get_vector_conspeed()
  功能： 获取轴常速插补运动速度
  返回值： 速度值（成功）----- -1（错误）

- int get_vector_profile(double *vec_vl, double *vec_vh, double *vec_ad, double *vec_dc)
  功能：获取快速插补运动速度
  参数：
  vl ---- 低速
  vh ---- 高速
  vec_ad ---- 上升加速度
  vec_dc ---- 下降加速度
  返回值： 0（成功）----- -1（错误）

2.2.13 专用输入检测函数

- int check_limit(int ch)
  功能：使能限位信号后获取轴当前限位信号有效状态
  参数：
  ch ---- 轴号
  返回值： 0（无效）----- 1（正限位有效）---- -1（负限位有效）---- -3（出错）

- int check_home(int ch)
  功能：使能轴原点信号后获取轴当前原点信号有效状态
  参数：
  ch ---- 轴号
  返回值： 0（无效）----- 1（有效）---- -3（出错）

- int check_alarm(int ch)
  功能：使能轴报警信号后获取轴当前报警信号有效状态
  参数：
  ch ---- 轴号
  返回值： 0（无效）----- 1（有效）---- -3（出错）

- int check_sfr(int cardno)
  功能：读取专用输入口电平状态（ORG、EL、轴 ALM、Z 信号和板 ALARM 信号）
  参数：
  cardno ---- 卡号
  返回值： 非负（专用输入口状态）----- -1（错误）

- int check_sfr_bit(int cardno, int bitno)
  功能：读取某位专用输入口电平状态（ORG、EL、轴 ALM、Z 信号和板卡 ALARM 信号）
  参数：
  cardno ---- 卡号
  bitno ---- 专用输入口位标记，范围 1 ~ 31
  返回值： 1（高电平）---- 0（低电平）----- -1（错误）

2.2.14 伺服专用函数

- checkin_axis_INP(int ch)
  功能：获取当前轴伺服到位信号（INP）的物理电平状态
  参数：
  ch ---- 轴号
  返回值： 1（高电平状态）----- 0（低电平状态）

- checkin_axis_SRDY(int ch)
  功能：获取当前轴伺服准备好信号（SRAY）的物理电平状态
  参数：
  ch ---- 轴号
  返回值： 1（高电平状态）----- 0（低电平状态）

- int outport_bit_RST (int ch,int status)
  功能：设置轴所控制伺服的报警清除输出口状态
  参数：
  ch ---- 轴号
  status ---- 状态： 0（低电平） 1（高电平）
  返回值： 0（正确）----- -1（错误

- int outport_bit_CL (int ch,int status)
  功能：设置轴所控制伺服的偏差计数清除输出口状态
  参数：
  ch ---- 轴号
  status ---- 状态： 0（低电平） 1（高电平）
  返回值： 0（正确）----- -1（错误

- int outport_bit_SEVERON (int ch,int status)
  功能：设置轴所控制伺服使能输出口状态
  参数：
  ch ---- 轴号
  status ---- 状态： 0（低电平） 1（高电平）
  返回值： 0（正确）----- -1（错误

2.2.15 通用 IO 口操作函数

- int checkin_byte(int cardno)
  功能：读取所有通用输入口状态
  参数：
  cardno ---- 卡号
  返回值 1（低电平）-------0（高电平）----- -1（错误）

- int checkin_bit(int cardno, int bitno)
  功能：读取某位通用输入口状态
  参数：
  cardno ---- 卡号
  bitno ---- 通用输入口位标记，范围 1 ~ 32（需配合 EA1616C 实现 17-32 位的通用输入状态读取。）
  返回值： 1（高电平）---- 0（低电平）----- -1（错误）

- int outport_byte(int cardno, int data)
  功能：设置主信号转接板上的通用输出口状态
  参数：
  cardno ---- 卡号；
  data ---- 主信号转接板上的通用输出口状态，某位为 0 表示输出低电平，为 1 表示无输出。
  注意：通用输出口 1 和 2 固定为高速位置比较输出，不能控制为普通通用输出口，参数 data 的 D2-D15 位分别控制转接板 P100-03 上 OUT3、OUT4、OUT5……OUT16。D0 和 D1 位没有实际对应的物理位置。
  返回值： 0（正确）----- -1（错误）

- int outport_byte_ex(int cardno, int data)
  功能：设置扩展 IO 板 EA1616C 上的通用输出口状态。
  参数：
  cardno ---- 卡号
  data ---- 扩展 IO 板 EA1616C 上的通用输出口状态，某位为 0 表示输出低电平，为 1 表示无输出。
  返回值： 0（正确）----- -1（错误）

- int outport_bit(int cardno, int bitno, int status)

  功能：设置某位通用输出口状态，包括主信号转接板上和扩展 IO 板。

  参数：

  cardno ---- 卡号

  bitno ---- 通用输出口位标记，范围 1 ~ 34。

  注意：通用输出口 1 和 2 固定为高速位置比较输出，不能控制为普通通用输出口，参数 bitno 为 3-16 时分别控制转接板 P100-03 上 OUT3、OUT4、OUT5……OUT16，参数 bitno 为 19-34 时分别控制 EA1616C 上 OUT1、OUT2、OUT3……OUT16，参数 bitno 为 17 和 18 时没有对应的物理 OUT 口。

  status ---- 状态： 0 表示输出低电平，为 1 表示无输出。

  返回值： 0（正确）----- -1（错误）

2.2.16 反向间隙处理

- int start_backlash(int ch)

  功能：开启反向间隙补偿功能，调用之前先调用 set_backlash()来设置补偿值，否则，就采用默认值 20 个脉冲。

  参数：

  ch ---- 轴号

  返回值： 0（正确）----- -1（错误）

- int end_backlash(int ch)

  功能：关闭反向间隙补偿功能，和 start_backlash()成对调用

  参数：

  ch ---- 轴号

  返回值： 0（成功）----- -1（错误）

- int set_backlash(int ch, double blash)

  功能：设置反向间隙补偿值

  参数：

  ch ---- 轴号

  blash ---- 补偿值（正）

  返回值： 0（正确）----- -1（错误）

2.2.17 运动中变速度

- int change_speed（int ch，double speed）

  功能：运动中改变速度

  参数：

  ch ---- 轴号

  speed---目标速度

  返回值： 0（成功）----- -1（错误）

2.2.18 动态改变目标位置

- int change_pos(int ch, double pos)
  功能：动态改变目标位置，目标位置为相对上条指令起始位置
  参数：
  ch ---- 轴号
  pos ---- 目标位（相对于上条运动指令起始位置）
  返回值： 0（成功）----- -1（错误）

- int change_pos_to(int ch, double pos)
  功能：动态改变目标位置，目标位置为相对与零点的位置
  参数：
  ch ---- 轴号
  pos ---- 目标位（相对于零点位置）
  返回值： 0（成功）----- -1（错误）

2.2.19 可掉电保护数据读写功能

- int write_password_flash(int cardno, int no, long data, long password)
  功能：写入数据到密码片区（片区 0）某个逻辑地址中
  参数：
  cardno ---- 卡号
  no ---- 地址编号：0~16383
  data ---- 写入的数据
  password ---- 校验密码（16383 号逻辑地址内的数据）
  返回值： 0（正确）----- -1（写入参数错误）---- -2（校验密码出错）

- int read_password_flash(int cardno, int no, long *data, long password)
  功能：读取密码片区（片区 0）某个逻辑地址中的数据
  参数：
  cardno ---- 卡号
  no ---- 地址编号：0~16382，16383 号逻辑地址用来存储密码，只读。
  data ---- 保存指定地址内的数据
  password ---- 校验密码（16383 号逻辑地址内的数据）
  返回值： 0（正确）----- -1（参数错误）---- -2（校验密码出错）

- int clear_password_flash(int cardno, long password)
  功能：擦写密码片区（片区 0），将整个片区所有 bit 位值“1”
  参数：
  cardno ---- 卡号
  password ---- 校验密码

返回值： 0（正确）----- -1（参数错误）---- -2（校验密码出错）

- int write_flash(int cardno, int piece, int no, long data)

功能：写入数据到数据片区（片区 1~15）的某个逻辑地址中

参数：

cardno ---- 卡号

piece ---- 片区号：1~15

no ---- 地址： 0~16383

data ---- 写入的数据

返回值： 0（正确）----- -1（参数错误）

- int read_flash(int cardno, int piece, int no, long *data)

功能：读数据片区（片区 1~15）中某个逻辑地址中的数据

参数：

cardno ---- 卡号

piece ---- 片区号

no ---- 地址： 0~16383

data ---- 存储读出的数据

返回值： 0（正确）----- -1（参数错误）

- int clear_flash(int cardno, int piece)

功能：擦写数据区，将指定片区数据各 bit 位置 1

参数：

cardno ---- 卡号

piece ---- 片区号：1~15

返回值： 0（正确）----- -1（参数错误）

2.2.20 多点位置比较输出

- int start_comparePulse(int ch,int pulsetype,int level,long time)

功能：在 HSIO 口直接输出一个脉冲或者电平信号

参数：

ch ---- 轴号

pulsetype ---- HSIO 口输出信号类型：

0 --- 输出脉冲信号，脉宽由参数 time 确定

1 --- 电平信号

level------ pulseType 为 0 时，本参数设置 HSIO 是否输出脉冲

0 -- 表示本通道不输出脉冲

1 -- 表示本通道输出脉冲

pulseType 为 1 时，本参数设置 HSIO 的输出电平

0 -- 表示输出低电平

1 -- 表示输出高电平

time ---- 参数 pulsetype 为 0 时，用来设定脉宽，范围[1，65535]，单位：us
返回值： 0（正确）----- -1（出错）

- int start_compareData(int ch,int source,int pulsetype,int startLevel,int time,long *pBuf,int count)
  功能：启动非等间距位置比较输出
  参数：
  ch ---- 轴号
  source ---- 位置比较数据来源：
  0 ： 指令脉冲位置
  1 ： 编码器反馈位置
  pulsetype ---- HSIO 口输出信号类型：
  0 ： 输出脉冲信号，脉宽由参数 time 确定
  1 ： 电平信号
  startLevel ---- 设置 HSIO 信号的初始电平：0 – 低电平 1 – 高电平
  建议设置为 1（初始化后默认状态）
  time ---- 参数 pulsetype 为 0 时，用来设定脉宽， 范围[1，65535]，单位：us
  pBuf ---- HSIO 的比较位置缓冲区，位置值为相对当前位置的距离，必须是单调上升的正数序列或单调下降的负数序列
  count ---- pBuf 数组的长度，最大值为 4096
  返回值： 0（正确）----- -1（出错）7---（设置出错，检查位置数组是否单调上升的正数序列或单调下降的负数序列）

- int start_compareLinear(int ch,int source,long startPos,long repeatTimes, int interval,int time)
  功能：启动等间距位置比较输出
  参数：
  ch ---- 轴号
  source ---- 位置比较数据来源：
  0 ： 指令脉冲位置
  1 ： 编码器反馈位置
  startPos ---- 位置比较起始位置，单位：pulse
  repeatTimes ---- 位置比较重复次数
  interval ----位置间隔，单位：pulse。与 startPos 同为正数或同为负数，不能为 0
  time ---- 参数 pulsetype 为 0 时，用来设定脉宽，范围[1， 65535]，单位：us
  返回值： 0（正确）----- 7（参数设置出错）

- int get_compareStatus(int ch,int *pCount)
  功能：查询已输出位置比较输出信号数目
  参数：
  pCount ---- 存储位置比较已输出的上升沿个数。若以电平方式输出，则返回值

为通道 1 或通道 2 已输出的上升沿个数，但与已完成的位置比较个数不等（以电平方式输出时，已完成的位置比较个数为已输出上升沿和已输出下降沿个数的总和）；若以脉冲方式输出，则返回值为通道 1 或通道 2 已输出的上升沿个数，与已完成的位置比较个数相等。

返回值： 0（正确）----- -1（出错）

- int compare_stop(int ch)
  功能：取消位置比较输出
  参数：
  ch ---- 轴号
  返回值：0（正确）----- -1（出错）

2.2.21 板卡号和版本读取

- int check_IC (int cardno)
  功能：获取指定板卡的 ID 号
  参数：
  cardno ---- 卡号

  返回值： 1~总卡数（正确）----- 其他（错误）

- int get_lib_ver(long* major, long *minor1, long *minor2)
  功能：获取函数库版本
  参数：
  major ---- 主版本号
  minor1 ---- 一级版本号
  minor2 ---- 二级版本号
  返回值： 0（正确）----- -1（参数错误）

- int get_sys_ver(long* major, long *minor1, long *minor2)
  功能：获取驱动程序版本
  参数：
  major ---- 主版本号
  minor1 ---- 一级版本号
  minor2 ---- 二级版本号
  返回值： 0（正确）----- -1（参数错误）

- int get_card_ver(int cardno, long* type, long* major, long *minor1, long *minor2)
  功能：获取板卡版本
  参数：
  cardno ---- 板卡
  type ---- 板卡类型
  major ---- 主版本号

minor1 ---- 一级版本号
minor2 ---- 二级版本号
返回值： 0（正确）----- -1（参数错误）

2.2.22 错误代码处理函数

- int get_err(int index, int *data)
  功能：获取错误代码
  参数：
  index ---- 错误索引：1~10
  data ---- 存储错误代码
  返回值： 0（正确）----- -1（index 错误或系统没有错误）

- int get_last_err()
  功能：获取最近一次错误代码
  返回值： 0（没有错误）----- Error Code（正确）

- int reset_err()
  功能：清除最近 10 次错误代码
  返回值： 0（正确）----- -1（错误）