PCI-9018

编程说明书

深圳市崧高技术有限公司

2019.11.2

版权声明

联系方式

深圳市崧高技术有限公司

地址： 深圳市南山区南山大道 1092 号亿利达大厦二栋 10 层 10A18

电话： 0755-86966230

传真： 0755-86534075

电子邮件：support@sunglow-tech.cn

网站：www.sunglow-tech.cn

目　　录

1 操作原理

1.1 脉冲输入与输出

1.1.1 脉冲输出

控制卡使用脉冲指令控制步进/伺服电机，支持两种脉冲输出模式：

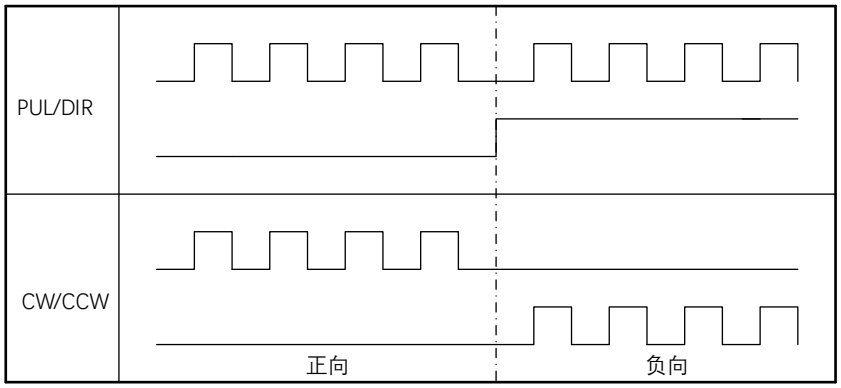
◆ 脉冲/方向模式（PUL/DIR 模式）

在该模式下，PUL 信号输出脉冲指令；脉冲的数目表示控制轴运动的距离，脉冲的频率表示控制轴的运动速度。 DIR 信号表示控制轴运动的方向

◆ 双脉冲模式（CW/CCW 模式）

该模式下，PUL 和 DIR 引脚分别表示正向脉冲（CW）输出、反向脉冲(CCW)输出。CW 输出的脉冲驱动控制轴正向运动， CCW 引脚输出的脉冲驱动控制轴反向运动。

其示意图如下：



控制卡的 PUL, DIR 信号输出为差分信号：PUL-与 PUL+, DIR-与 DIR+，上图中的 PUL、DIR 信号为经过差分驱动器(AM26LS31)前的信号。

1.1.2 脉冲输入

控制卡的脉冲输入用于对外部的位置反馈装置（比如旋转编码器、光栅尺）反馈的位置脉冲进行计数。详细描述请参考1.4

1.2 运动规划

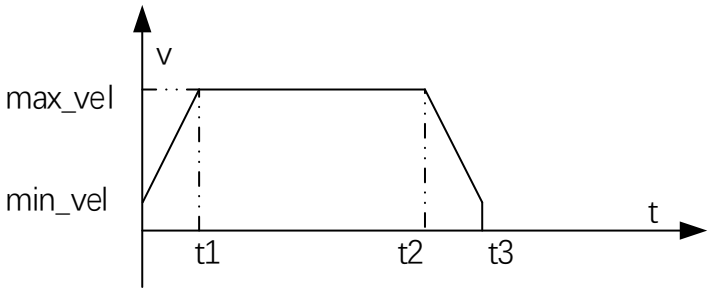
1.2.1 点位运动

点位运动也叫点到点运动，控制轴运动指定距离或者运动到指定位置。

当控制轴处于停止状态时，发送点位运动命令(p9018_axis_incr_move 或 p9018_axis_abs_move)将使控制轴以指定的速度运动指定的距离；控制轴在运动过程中，如果收到停止命令(p9018_axis_abort)，控制轴减速停止；如果出现异常情况时，控制轴立即停止（没有减速过程）。

目标位置可以用绝对位置方式或相对位置方式来表示。在相对位置方式下，使用位置偏移（目标位置与当前命令位置的差值）作为参数，位置偏移的符号决定了控制轴运动的方向。 在绝对位置方式下，目标位置和当前命令位置的差值符号决定了控制轴运动的方向。加速度和减速度值不同时，构成非对称速度曲线。

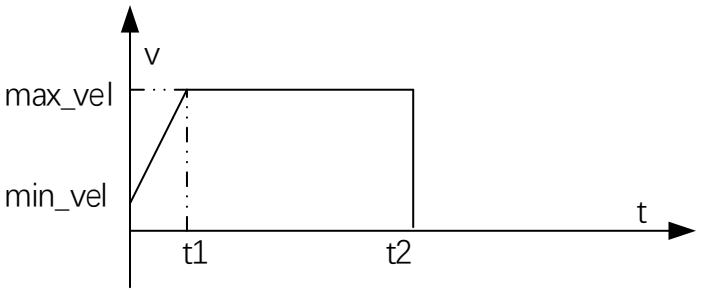
下图演示了 T 型加减速曲线的速度-时间关系：



控制轴以起始速度 min_vel 开始运行，加速到最大速度后（t1 时刻），保持恒速运行，速度为 max_vel，运行到减速点后（t2 时刻），开始减速， 到达目标位置后停止运行（t3 时刻）。

速度-时间曲线与时间轴围成的闭合多边形的面积为运动距离。

下图演示了控制轴在点位运动过程中对异常情况的处理：



如上图所示，控制轴在运行过程中如果遇到异常（t2 时刻），比如硬件限位有效、告警输入有效、急停输入有效，将直接降速为 0，没有减速过程。

假设起始速度 min_vel，最大速度 max_vel，参数的单位是脉冲/秒(P/s)， 加速度 acc、减速度 dec 的单位为脉冲/平方秒(P/s^2)， 加速时间 Tacc, 减速时间 Tdec 的计算公式为：

Tacc = (max_vel – min_vel)/acc

Tdec= (max_vel – min_vel)/dec

相关函数如下：

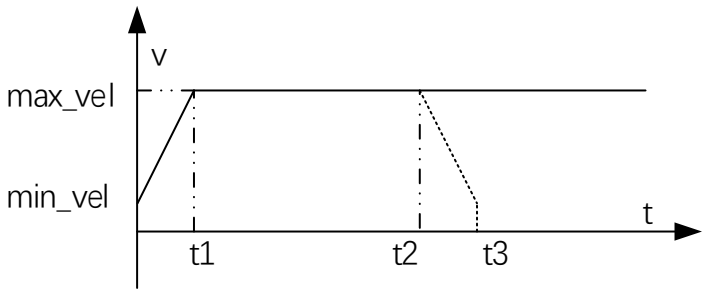
函数	描述
int p9018_axis_incr_move(int axis, int incr)	使用相对位置进行点位运动
int p9018_axis_abs_move(int axis, int pos)	使用绝对位置进行点位运动

1.2.2 连续运动

连续运动也叫速度模式运动，控制轴收到该命令后，从起始速度加速到指定的速度后持续运动，直到收到停止命令或者出现异常情况；

如果收到停止命令，控制轴将减速停止；如果出现异常情况时，控制轴立即停止（没有减速过程）。

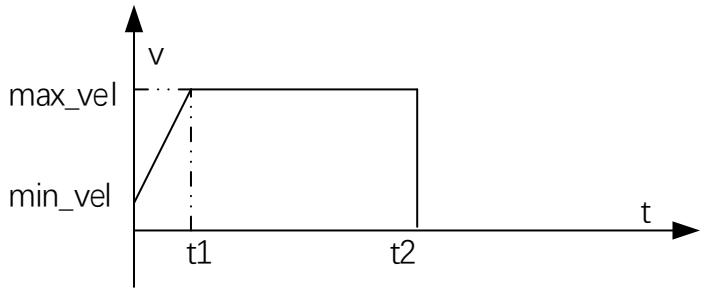
下图演示了 T 型加减速曲线的速度-时间关系：



控制轴以起始速度 min_vel 开始运行，加速到最大速度 max_vel 后（t1 时刻），保持恒速运行。

控制轴如果收到停止命令（p9018_axis_abort）将开始减速，速度降到 min_vel 以下后停止运动（参考 t2 时刻的虚线示意）。

下图演示了控制轴在连续运动过程中对异常情况的处理：



如上图所示，控制轴在运行过程中如果遇到异常（t2 时刻），比如硬件限位有效、告警输入有效、急停输入有效，将直接降速为 0，没有减速过程。

相关函数如下：

函数	描述
int p9018_axis_cont_move(int axis, double vel)	连续运动

1.2.3 查找原点

控制轴查找原点，是指控制轴查找外部的参考点（通常是机械开关或光电开关），为其命令位置建立参考点的过程。

当控制轴处于停止状态时，发送查找原点命令(p9018_axis_home)将使控制轴启动原点查找，原点条件满足后，控制轴减速并停止运动。原点查找过程是否结束，可以通过 p9018_axis_get_status 函数来查询。

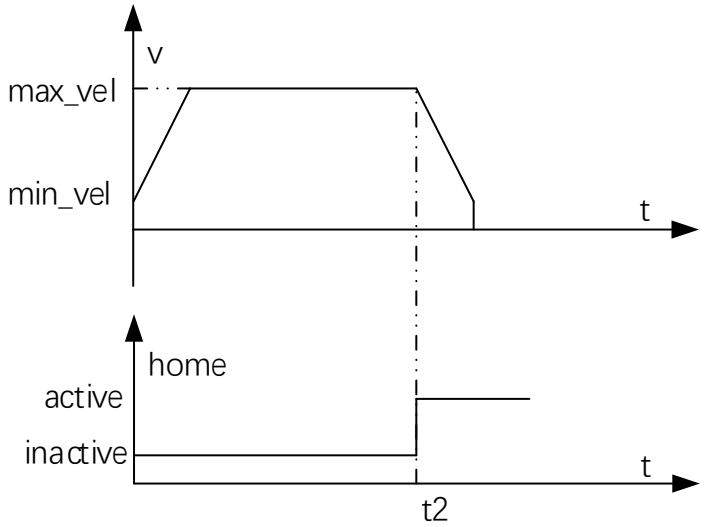
查找原点时的运动方向，由 p9018_axis_home 中速度参数 vel 的符号来决定：当 vel 大于零时，控制轴正向运动来查找原点；当 vel 小于零时，控制轴负向运动来查找原点。

PCI-9018 支持两种查找原点模式：

1) 仅查找原点

在查找原点过程中，当原点输入从无效变为有效时，控制轴的命令位置计数器被清零（表示此机械位置为控制轴的命令位置的参考零点），控制轴同时开始减速， 速度降到起始速度以下后，停止运动。

下图演示了控制轴仅查找原点过程中的的速度-时间关系：



控制轴以起始速度 min_vel 开始查找原点，加速到最大速度 max_vel 后保持恒速运行；

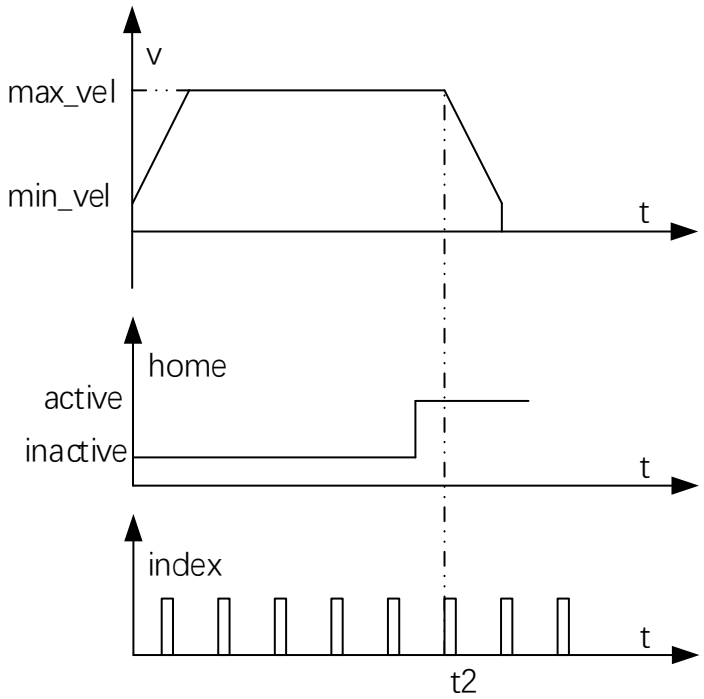
在查找原点过程中，当原点输入从无效变为有效时（t2 时刻），控制轴的命令位置计数器清零，控制轴同时开始减速， 速度降到起始速度 min_vel 以下后，停止运动。

2) 查找原点和 Index

在查找原点过程中，当原点输入从无效变为有效后，控制轴等待 Index 输入的上升沿，如果收

到有效的 index 输入的上升沿，控制轴的命令位置计数器和反馈位置计数器被清零（表示此机械位置为控制轴的命令位置的参考零点），控制轴同时开始减速， 速度降到起始速度以下后，停止运动。

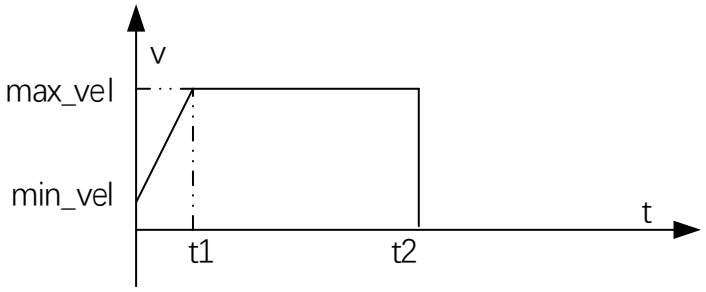
下图演示了控制轴查找原点和 Index 过程中的的速度-时间关系：



控制轴以起始速度 min_vel 开始查找原点，加速到最大速度 max_vel 后保持恒速运行；

在查找原点过程中，当原点输入从无效变为有效后的第一个 index 上升沿时（t2 时刻），控制轴的命令位置计数器和反馈位置计数器被清零，控制轴同时开始减速， 速度降到起始速度 min_vel 以下后，停止运动。

下图演示了控制轴在查找原点过程中对异常情况的处理：



如上图所示，控制轴在查找原点过程中如果遇到异常（t2 时刻），比如硬件限位有效、告警输入有效、急停输入有效，将直接降速为 0，没有减速过程。

相关函数如下：

函数	描述

int p9018_axis_home(int axis, double vel, int use_index)	控制轴查找原点

1.3 控制轴专用输入输出

控制卡为每个控制轴提供了正限位输入(EL+)、负限位输入(EL-)、原点输入(Home)、告警输入(Fault)输入和使能输出(Enable)。

控制卡对这些专用输入信号进行了滤波处理，以减小外界的电磁干扰对控制卡的影响；这些信号的状态可以通过 p9018_axis_get_status 函数进行查询。

1.3.1 限位输入

当控制轴正向运动时，如果正限位(EL+)有效，则控制轴立即停止，没有减速过程；当控制轴反向运动时，如果负限位(EL-)有效，控制轴立即停止，没有减速过程；限位输入信号有效导致控制轴停止运动后，必须向控制轴发送反向运动的命令，否则控制轴不能运动。这样的机制可以有效保护机械部分，提高了安全性。

限位输入的类型可以通过 p9018_axis_set_parameter 来设置：常开类型、常闭类型。控制卡默认设置为常开类型。 常闭类型的开关在正常情况下闭合，只有机械部分撞到限位开关后，开关才断开，控制轴停止运动；如果有其它异常情况，比如限位开关的信号线断开的时候，也会认为是限位开关有效，控制轴不能运动，这样可以避免设备在故障情况下继续工作，提高机械的安全性，所以建议限位开关使用常闭类型。

1.3.2 原点输入

控制轴的原点输入信号用于控制轴查找原点。在“仅查找原点”模式下，控制轴查找原点时只使用原点输入信号(Home)。在“查找原点和 Index”模式下，控制轴查找原点时除了使用原点输入信号(Home)外，还要使用编码器输入的 Index 信号。

原点输入的类型、Index 信号边沿可以通过 p9018_axis_set_parameter 函数进行设置。是否查找 Index 信号可通过 p9018_axis_home 函数的 use_index 参数来指定。

1.3.3 告警输入

控制轴的告警输入用于检测伺服驱动器的告警输出信号。

当控制轴告警输入被使能并且处于运动状态时，如果告警输入信号有效，控制轴立即停止运动，没有减速过程。

告警输入的使能和类型可以通过 p9018_axis_set_parameter 来设置，支持常开类型和常闭类型，默认设置为常开类型。

1.3.4 使能输出

控制轴的使能输出用于控制伺服驱动器的伺服使能信号。当使能输出有效（输出导通）时，伺服驱动器的功率变换装置上电；当使能输出无效（输出截止）时，伺服驱动器的功率变换装置断电。合理的使用使能输出，可降低伺服电机的功耗。

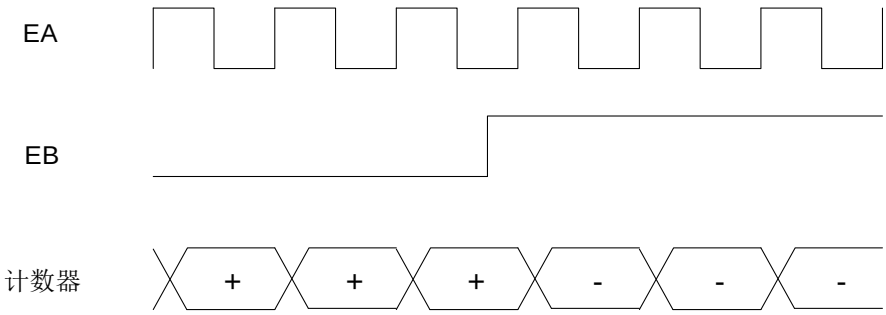
相关函数如下：

函数	描述
int p9018_axis_set_servo_on(int axis, int on)	设置控制轴伺服使能输出

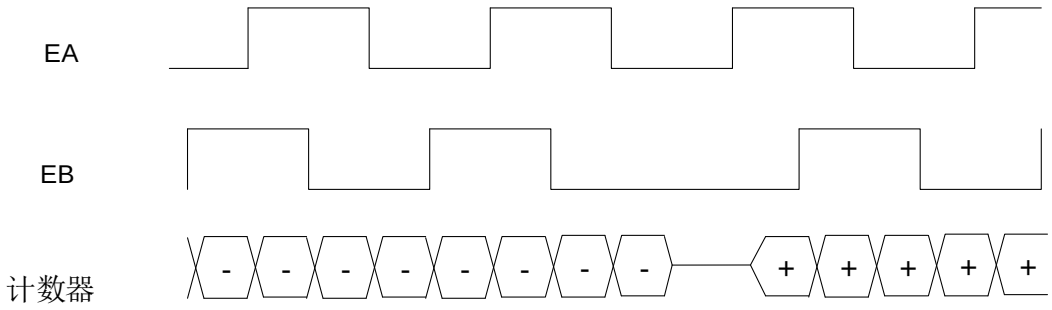
1.4 位置计数器

控制卡为每个控制轴提供了两个 28 位的计数器，用来对输出脉冲（命令位置计数器）和反馈脉冲（反馈位置计数器）进行计数。其中反馈位置计数器支持两种格式输入：脉冲与方向(PUL/DIR) ，正交编码(4xAB)。

在脉冲与方向(PUL/DIR)输入模式下， EB 为低电平时 EA 出现一个上升沿，计数器值加 1；EB 为高电平时 EA 出现一个上升沿，计数器减 1；示意图如下：



在正交编码(4xAB)输入格式下，如果 EA 信号相位超前 EB 信号 90 度，则表示电机在正向运转，计数器值增加；如果 EB 信号相位超前 EA 信号 90 度，则表示电机在反向运转，计数器值递减；示意图如下：



控制卡的反馈脉冲输入为差分信号，上图中的 EA 为 EA+与 EA-为经过差分接收器后的信号（EA+与 EA-的电压差）， EB 为 EB+与 EB-经过差分接收器后的信号（EB+与 EB-的电压差）。

计数器的初值可以通过 p9018_axis_set_cmd_pos、p9018_axis_set_act_pos 函数来设置；计数器值可以通过 p9018_axis_get_status 函数来读出。

关于计数器溢出

控制轴的计数器长度为 28 位，使用二进制补码，计数范围在 -134217728 ~ 134217727。如果计数器值已经计到 134217727 后又接收到正向脉冲，则产生向上溢出，计数器值变为-134217728; 如果计数器值已经计到-134217728 后又接收到反向脉冲，则产生向下溢出，计数器值变为 134217727; 用户的应用程序应该注意判别该异常情况。

1.5 位置比较器

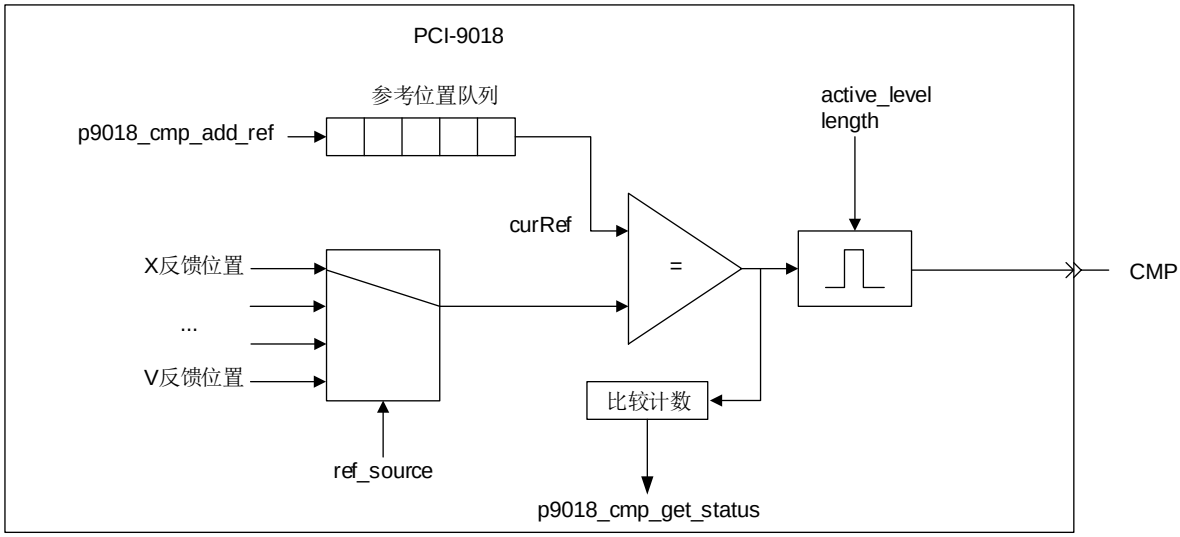
控制卡提供了一个位置比较器及输出。该位置比较器内置了一个 512 级的参考位置队列，来保存比较器的参考位置。参考位置被写入参考位置队列后，位置比较器可依次从参考位置队列中取出参考值与控制轴的当前位置进行比较，两者相等时比较器输出一个脉冲，该脉冲的有效电平可以使用 p9018_comp_enable 函数里面的 level 参数进行设置，脉冲的宽度为 length 参数指定。

位置比较器只有在比较条件满足时，才会从参考位置队列中取下一个参考值。如果当前比较条件没有满足，则一直处于等待状态，不会取下一个参考值。为了避免位置比较器发生异常，如果控制轴运动方向为正向，存入参考位置队列中的位置数据必须以增大的顺序存入；如果控制轴运动方向为负向，存入参考位置队列中的位置数据必须以减小的顺序存入。

位置比较器使用控制轴的反馈位置计数器作为位置源，支持的参考位置范围为-134217728 ~134217727，因此用户向参考位置队列中写入的位置值不能超过这个范围。

使能位置比较器的函数 p9018_comp_enable 有返回值，用户程序需要检查该返回值是否正常;如果没有检查该返回值，用户程序会无法获知位置比较器的异常状态。

位置比较器设计原理如下：



1.6 安全保护

控制轴的正限位(EL+)输入、负限位(EL-)输入、告警输入(Fault)、急停输入(Abort)构成了控制轴的安全保护输入，用于处理控制轴的异常情况。控制轴在运动过程中如果遇到异常，比如限位有效、告警输入、急停输入有效，将直接降速为 0，没有减速过程。

当限位（正限位、负限位）有效导致控制轴停止运动时，控制轴无法执行继续向限位方向运动的命令，但可以执行离开限位方向的运动命令。

在告警输入有效期间，控制轴无法执行任何运动命令，因此用户必须进行干预并解除伺服驱动器的告警后，控制轴才可以继续执行运动命令。

在急停输入有效期间，控制轴无法执行任何运动命令。

2 函数说明

2.1 前提约定

2.1.1 物理单位

名称	英文名称	单位	描述
位置	Position	P	脉冲个数
速度	Velocity	P/s, PPS	脉冲每秒
加速度	Acceleration	P/s^2	脉冲每平方秒
加速度率	Jerk	P/s^3	脉冲每立方秒
电压	voltage	V	伏特
时间	time	s	秒
时间	time	ns	纳秒, 1 s = 1000000000 ns

由于硬件限制，速度的最大值为 4100000 P/s。

2.1.2 数据类型

数据类型	范围	说明
int	-2147483647 ~ 2147483648	有符号整形，32 位
unsigned int	0~4294967295	无符号整形，32 位
double	-1.7976931348623158e+308 ~ 1.7976931348623158e+308	双精度浮点数，64 位
	NaN	Not A Number, FPU 产生异常后的返回值，比如负数的平方根
	Inf	infinity, FPU 产生异常后的返回值，比如被零除

2.1.3 返回值定义

函数库内函数返回值为 int 类型，其常用的定义如下：

返回值	描述	原因分析
0	命令发送并执行成功	
1	无效参数	
2	分配系统资源失败	
7	无法访问驱动程序或通讯失败	设备未安装驱动程序或未成功安装设备
12	设备已经打开	有多个应用程序打开设备
202	控制轴忙	控制轴未停止时接收到新的运动命令
205	控制轴限位有效	控制轴限位有效、告警输入有效时接收到新的运动命令

2.1.4 控制轴编号

控制卡函数库内函数使用了卡号(card_no)或控制轴轴号(axis)作为参数。

控制卡上的拨码开关 S1 设置决定了控制卡的卡号(出厂默认设置为 0), 一旦卡号设定后，该控制卡上的 X, Y, Z, A,B,C,U,V 八个轴的轴号(axis)分别为：

控制轴名称	控制轴轴号
X	卡号 * 8 + 0
Y	卡号 * 8 + 1
Z	卡号 * 8 + 2
A	卡号 * 8 + 3
B	卡号 * 8 + 4
C	卡号 * 8 + 5
U	卡号 * 8 + 6
V	卡号 * 8 + 7

2.1.5 函数库使用

控制卡提供了支持 Windows XP/Vista/Win7/Win10 的驱动程序、动态链接库(函数调用方式为stdcall)、配置文件(PCI9018.ini)供用户使用。用户使用前先要安装驱动程序，驱动程序在PCI-9018\driver 目录下，动态链接库、导入库和配置文件在 lib 目录下, 头文件在 include 目录下。

在安装驱动程序时，操作系统可自动将对应版本的动态链接库(x86, x64)安装到系统目录下，因此用户在开发应用程序时，可不必将动态链接库复制到应用程序目录下。该特性对于使用.NET 开发的、默认配置为 Any CPU 的应用程序非常重要。

在 Windows 系统下，用户可以使用任何能够支持动态链接库的开发工具来开发应用程序。下面分别以 C/C++, C#, VB.NET 为例讲解如何在这些开发工具中使用运动控制卡的动态链接库。

使用 C/C++(以 Visual Studio 2005 为例)来开发运动控制程序：

1) 启动 Visual Studio, 新建一个工程, 工程类型选择 Visual C++\MFC 或者 Visual C++\Win32；
2) 将 PCI-9018\lib 目录内 PCI9018.ini 文件复制到可执行程序目录中；
   将 PCI-9018\lib 目录内 PCI9018.lib 文件复制到工程文件目录中；
   将 PCI-9018\include 目录内 PCI9018.h 文件复制到工程文件目录中；
3) 选择“Project”菜单下的“Settings…”菜单项；
4) 切换到“Link”标签页，在“Object/library modules”栏中输入 lib 文件名 PCI9018.lib；
5) 在应用程序文件中加入函数库头文件的声明，例如： #include “PCI9018.h”
6) 至此，用户就可以在 Visual C++中调用函数库中的任何函数，开始编写应用程序。

使用 C#(以 Visual Studio 2005 为例)来开发运动控制程序：

1) 启动 Visual Studio 2005，新建一个工程, 工程类型选择 Visual C#\Windows\Windows Application；
2) 将 PCI-9018\lib 目录内 PCI9018.ini 文件复制到可执行程序目录中；
   将 PCI-9018\include 目录内 PCI9018.cs 文件复制到工程文件目录中；
3) 在“工程”菜单下，选择“添加已有项目”，在弹出的对话框里面选择 PCI9018.cs;
4) 至此，用户就可以在工程里面中调用函数库中的任何函数，开始编写应用程序。

使用 VB.NET(以 Visual Studio 2005 为例)来开发运动控制程序：

1) 启动 Visual Studio 2005，新建一个工程, 工程类型选择 Visual Basic\Windows\Windows Application；
2) 将 PCI-9018\lib 目录内 PCI9018.ini 文件复制到可执行程序目录中；

将 PCI-9018\include 目录内 PCI9018.vb 文件复制到工程文件目录中；

3) 在“工程”菜单下，选择“添加已有项目”，在弹出的对话框里面选择 PCI9018.vb;

4) 至此，用户就可以在工程里面中调用函数库中的任何函数，开始编写应用程序。

2.1.6 函数列表

PCI-9018 的函数依据功能，可以分为几大类：

- 控制卡初始化
- 控制轴参数设置
- 控制轴运动
- 控制轴管理
- 数字量输入输出
- 位置比较器控制
- 控制卡查询

函数名称	功能描述
p9018_open	初始化控制卡。
p9018_close	关闭控制卡。
p9018_axis_set_parameter p9018_axis_set_real_parameter	设置控制轴参数。
p9018_axis_set_profile	设置控制轴速度规划参数（速度、加速度、加加速度）。
p9018_axis_cont_move	控制轴连续运动。
p9018_axis_abs_move	控制轴点位运动（使用绝对位置）。
p9018_axis_incr_move	控制轴点位运动（使用相对位置）。
p9018_axis_abort	控制轴停止。
p9018_axis_home	控制轴查找原点。
p9018_axis_get_status p9018_axis_get_status_ex	控制轴状态查询。
p9018_axis_set_servo_on	设置控制轴使能输出。
p9018_axis_set_cmd_pos	设置控制轴命令位置。
p9018_axis_set_act_pos	设置控制轴反馈位置。
p9018_motion_set_dout_ex	设置所有数字量输出(DO)状态。
p9018_motion_set_dout	设置单个数字量输出(DO)状态。
p9018_motion_get_dout_ex	查询所有数字量输出(DO)状态。

p9018_motion_get_din_ex	查询所有数字量输入(DI)状态。
p9018_motion_get_din	查询单个数字量输入(DI)状态。
p9018_cmp_enable	设置位置比较器参数。
p9018_cmp_set_ref	向位置比较器的参考位置队列添加一个参考位置。
p9018_cmp_get_status	查询位置比较器的状态。
p9018_motion_get_info	查询控制卡版本信息。

2.2 控制卡初始化

控制命令	int p9018_open(int* count, int* ID)
功能描述	查找并初始化系统里面所有的 PCI-9018。 调用该函数后，如果当前文件夹下存在 PCI9018.ini 配置文件，那么函数库将从 PCI9018.ini 加载默认参数；其默认的配置： 脉冲输出设置为 PUL/DIR 模式，反馈位置计数器的输入格式设置为 4xAB 模式。
参数	cout – 查找到的控制卡数目，可选参数，不使用时设置为 NULL ID – 查找到的控制卡卡号，建议使用含有 16 个元素的 int 类型数组作为该参数，可选参数，不使用时设置为 NULL
返回	参考返回值定义
前置条件	无
后置条件	无

控制命令	int p9018_close(void)
功能描述	关闭系统里面所有的 PCI-9018，释放使用到的系统资源。
参数	无
返回	参考返回值定义
前置条件	p9018_open 被成功调用过
后置条件	无

2.3 控制轴参数设置

2.3.1 速度与加速度设置

控制命令	int p9018_axis_set_profile(int axis, double min_vel, double max_vel, double accel, double decel, double jerk)
功能描述	设置控制轴的起始速度、最大速度、加速度、减速度、加加速度。
参数	axis - 轴号 min_vel– 起始速度，单位 P/s max_vel- 最大速度，单位 P/s accel – 加速度，单位 P/s^2 decel – 减速度（减速过程使用的加速度），单位 P/s^2 jerk – 加加速度，单位 P/s^3
返回	参考返回值定义
前置条件	无
后置条件	无

2.3.2 脉冲输入输出设置

控制命令	int p9018_axis_set_parameter(int axis, PN_OutputMode, int value)
功能描述	设置控制轴脉冲输出格式（PUL/DIR, CW/CCW）。 初始化后，控制轴默认使用 PUL/DIR 模式。
参数	axis - 轴号 value –脉冲输出格式，0 表示 PUL/DIR 格式， 1 表示 CW/CCW 格式
返回	参考返回值定义
前置条件	无
后置条件	无

控制命令	int p9018_axis_set_parameter(int axis, PN_InvertOutputDirection, int value)
功能描述	反转控制轴脉冲输出信号的方向。 初始化后，控制轴默认不反转脉冲输出的方向。
参数	axis - 轴号 value –方向反转设置，非零表示反转，零表示不反转
返回	参考返回值定义

前置条件	无
后置条件	无

控制命令	int p9018_axis_set_parameter(int axis, PN_InputMode, int value)
功能描述	设置控制轴的脉冲输入模式（PUL/DIR 或 4x AB)。 初始化后，控制轴默认使用正交编码模式(4x AB)。
参数	axis - 轴号 value – 0 表示使用 PUL/DIR 模式，1 表示使用 4xAB 模式
返回	参考返回值定义
前置条件	无
后置条件	无

控制命令	int p9018_axis_set_parameter(int axis, PN_InvertInputDirection, int value)
功能描述	反转控制轴编码器计数器的计数方向。 初始化后，控制轴默认不反转编码器计数器的计数方向。
参数	axis - 轴号 value –计数方向反转设置，非零表示反转，零表示不反转
返回	参考返回值定义
前置条件	无
后置条件	无

2.3.3 限位输入设置

控制命令	int p9018_axis_set_parameter(int axis, PN_PosLimitSwitchType, int value)
功能描述	设置控制轴正限位和负限位输入信号的类型（常开、常闭）。 初始化后，控制轴默认使用常开类型。
参数	axis - 轴号 value –输入信号类型，零表示常开类型，非零表示常闭类型
返回	参考返回值定义
前置条件	无
后置条件	无

控制命令	int p9018_axis_set_parameter(int axis, PN_PosLimitSwitchEnable, int value)
功能描述	设置控制轴正限位输入信号的使能（使能、禁止）。

	初始化后，控制轴默认使能正限位输入。
参数	axis - 轴号 value –禁止与使能设置，零表示禁止，非零表示使能
返回	参考返回值定义
前置条件	无
后置条件	无

控制命令	int p9018_axis_set_parameter(int axis, PN_NegLimitSwitchEnable, int value)
功能描述	设置控制轴负限位输入信号的使能（使能、禁止）。 初始化后，控制轴默认使能负限位输入。
参数	axis - 轴号 value –禁止与使能设置，零表示禁止，非零表示使能
返回	参考返回值定义
前置条件	无
后置条件	无

2.3.4 查找原点设置

控制命令	int p9018_axis_set_parameter(int axis, PN_HomeSwitchType, int value)
功能描述	设置控制轴原点输入信号的类型（常开、常闭）。 初始化后，控制轴默认使用常开类型。
参数	axis - 轴号 value –输入信号类型，零表示常开类型，非零表示常闭类型
返回	参考返回值定义
前置条件	无
后置条件	无

控制命令	int p9018_axis_set_parameter(int axis, PN_IndexActiveLevel, int value)
功能描述	设置控制轴在查找原点和 Index 时 Index 输入信号的极性（上升沿、下降沿）。 初始化后，控制轴默认使用上升沿。
参数	axis - 轴号 value –Index 输入极性，零表示下降沿（低电平），非零表示上升沿（高电平）
返回	参考返回值定义

前置条件	无
后置条件	无

2.3.5 告警输入设置

控制命令	int p9018_axis_set_parameter(int axis, PN_AmpFaultType, int value)
功能描述	设置控制轴告警输入信号的类型（常开、常闭）。 初始化后，控制轴默认使用常开类型。
参数	axis - 轴号 value –输入信号类型，零表示常开类型，非零表示常闭类型
返回	参考返回值定义
前置条件	无
后置条件	无

控制命令	int p9018_axis_set_parameter(int axis, PN_AmpFaultEnable, int value)
功能描述	设置控制轴告警输入信号的使能（使能、禁止）。 初始化后，控制轴默认使能告警输入。
参数	axis - 轴号 value – 禁止与使能设置，零表示禁止，非零表示使能
返回	参考返回值定义
前置条件	无
后置条件	无

2.4 控制轴运动

控制命令	int p9018_axis_abs_move(int axis, int pos)
功能描述	启动控制轴运动到指定位置（使用绝对位置）。 如果控制轴已经处于硬件限位情况下，控制轴只接收能够使其解除硬件限位的命令，比如：控制轴已经处于负向硬件限位，此时只接受正向运动的命令；如果处于负向软件限位，此时只可接受负向运动的命令；
参数	axis – 轴号

	pos - 使用绝对位置表示的目标位置，单位 P
返回	参考返回值定义
前置条件	控制轴处于停止状态、告警输入无效、急停输入无效、运动方向上的硬件限位无效
后置条件	无

控制命令	int p9018_axis_incr_move(int axis, int incr)
功能描述	启动控制轴运动指定距离（使用相对位置）。 如果控制轴已经处于硬件限位情况下，控制轴只接收能够使其解除硬件限位的命令，比如：控制轴已经处于负向硬件限位，此时只接受正向运动的命令；如果处于负向软件限位，此时只可接受负向运动的命令；
参数	axis – 轴号 incr - 使用相对位置表示的运动距离，单位 P
返回	参考返回值定义
前置条件	控制轴处于停止状态、告警输入无效、急停输入无效、运动方向上的硬件限位无效
后置条件	无

控制命令	int p9018_axis_cont_move(int axis, double vel)
功能描述	启动控制轴进行连续运动（速度模式运动）。 如果控制轴已经处于硬件限位情况下，控制轴只接收能够使其解除硬件限位的命令，比如：控制轴已经处于负向硬件限位，此时只接受正向运动的命令；如果处于负向软件限位，此时只可接受负向运动的命令；
参数	axis – 轴号 vel – 运动速度，其符号决定运动方向，单位 P/s
返回	参考返回值定义
前置条件	控制轴处于停止状态、告警输入无效、急停输入无效、运动方向上的硬件限位无效
后置条件	无

控制命令	int p9018_axis_home(int axis, double vel, int use_index)
功能描述	启动控制轴查找原点。

	如果控制轴已经处于硬件限位情况下，控制轴只接收能够使其解除硬件限位的命令，比如：控制轴已经处于负向硬件限位，此时只接受正向运动的命令；如果处于负向软件限位，此时只可接受负向运动的命令； 在仅查找原点(use_index=0)模式下，查找原点是指查找控制轴原点输入从无效到有效的边沿，如果原点输入在调用该命令前已经处于有效状态，该函数立即返回，控制轴不做任何运动。因此在此种情况下，应用程序应使控制轴反向运动、确认离开原点后，再调用该命令来查找原点。 在查找原点和 Index(use_index=1)模式下，如果原点输入在调用该命令前已经处于有效状态，该函数会忽略查找控制轴原点输入从无效到有效的过程，直接去查找 Index 信号的边沿。如果控制轴的此种行为不是期待的行为，那么在此种情况下，应用程序应使控制轴反向运动、确认离开原点后，再调用该命令来查找原点。
参数	axis – 轴号 vel – 运动速度，其符号决定运动方向，单位 P/s use_index - 零表示仅查找原点，非零表示查找原点和 Index
返回	参考返回值定义
前置条件	控制轴处于停止状态、告警输入无效、急停输入无效、运动方向上的硬件限位无效
后置条件	无

控制命令	int p9018_axis_abort(int axis)
功能描述	停止控制轴，控制轴减速到零速度后停止运动。
参数	axis – 轴号
返回	参考返回值定义
前置条件	无
后置条件	无

2.5 控制轴管理

控制命令	int p9018_axis_set_cmd_pos(int axis, int pos)
功能描述	设置控制轴的命令位置。 **注意：** **该函数直接重置控制轴的命令位置，没有通过原点查找来完成；因此用户在使用该**

	函数时要非常谨慎，尽量避免使用。
参数	axis - 轴号 pos – 设置值，范围：-134217728 ~ 134217727，单位: P
返回	参考返回值定义
前置条件	控制轴处于停止状态
后置条件	无

控制命令	int p9018_axis_set_act_pos(int axis, int pos)
功能描述	设置控制轴的编码器反馈位置。
参数	axis - 轴号 pos – 设置值，范围：-134217728 ~ 134217727，单位: P
返回	参考返回值定义
前置条件	控制轴处于停止状态
后置条件	无

控制命令	int p9018_axis_set_servo_on(int axis, int on)
功能描述	设置控制轴的伺服使能输出。
参数	axis - 轴号 on – 使能输出的状态，非零表示使能（输出导通），零表示禁止（输出截止）
返回	参考返回值定义
前置条件	无
后置条件	无

2.6 控制轴状态查询

控制命令	int p9018_axis_get_status(int axis, MOT_AXIS_STAT * pStat)
功能描述	查询控制轴的状态信息。
参数	axis - 轴号 pStat - 控制轴的状态信息缓冲区，结构体类型，其成员定义如下： cmdPos – 命令位置，单位: P cmdVel – 当前命令速度，单位: P/s actPos – 反馈位置，单位: P homeSwitch – 原点输入的状态，非零表示有效，零表示无效

	negHardLimit – 负向限位输入的状态，非零表示有效，零表示无效 posHardLimit – 正向向限位输入的状态，非零表示有效，零表示无效 fault – 告警输入的状态，非零表示有效，零表示无效 abort – 急停输入的状态，非零表示有效，零表示无效 inpos – 控制轴停止状态，非零表示停止，零表示正在运动 enabled – 控制轴使能输出状态，非零表示导通，零表示截止 error – 控制轴错误状态，非零表示上次的运动命令执行过程中出现异常，零表示没有出现异常 没有说明的其它结构体成员保留为以后使用。
返回	参考返回值定义
前置条件	无
后置条件	无

控制命令	int p9018_axis_get_status_ex(int axis, int* buff, int cnt)
功能描述	查询控制轴的状态信息。
参数	axis - 轴号 buff - 控制轴的状态信息缓冲区; cnt – 缓冲区元素个数 在缓冲区大小足够的前提下，其成员定义如下： buff[0] – 命令位置，单位: P buff[1] – 当前命令速度，单位: P/s buff[3] – 反馈位置，单位: P buff[8] – 原点输入的状态，非零表示有效，零表示无效 buff[9] – 负向限位输入的状态，非零表示有效，零表示无效 buff[10] – 正向向限位输入的状态，非零表示有效，零表示无效 buff[11] – 告警输入的状态，非零表示有效，零表示无效 buff[12] – 急停输入的状态，非零表示有效，零表示无效 buff[15] – 控制轴停止状态，非零表示停止，零表示正在运动 buff[16] – 控制轴使能输出状态，非零表示导通，零表示截止 buff[19] – 控制轴错误状态，非零表示上次的运动命令执行过程中出现异常，零表示没有出现异常
返回	参考返回值定义
前置条件	无

后置条件	无

2.7 数字量输入输出

控制命令	int p9018_motion_set_dout(int card_no, int index, int value)
功能描述	设置控制卡上单路数字量输出状态。
参数	card_no – 控制卡卡号，范围：0～15 index - 要设置的数字量输出编号，范围 0 ~ 15 value – 输出状态，非零表示导通（输出通道的晶体管集电结导通）； 零表示截至（输出通道的晶体管集电结截至）
返回	参考返回值定义
前置条件	无
后置条件	无

控制命令	int p9018_motion_set_dout_ex(int card_no, UINT value)
功能描述	设置控制卡上所有的数字量输出。
参数	card_no – 控制卡卡号，范围：0～15 value – 输出端口状态, 32 位无符号数，其中位 0~ 15 状态分别对应 DO[0]~ DO[15] 的状态； 1 表示导通（输出通道的晶体管集电结导通）； 零表示截至（输出通道的晶体管集电结截至）。
返回	参考返回值定义
前置条件	无
后置条件	无

控制命令	int p9018_motion_get_dout(int card_no, int index, int* pValue)
功能描述	查询控制卡上单路数字量输出状态。
参数	card_no – 控制卡卡号，范围：0～15 index – 要设置的数字量输出编号，范围 0 ~ 15 pValue – 输出状态, 非零表示导通（输入通道光耦的集电结导通）； 零表示截至（输入通道光耦的集电结截至）。
返回	参考返回值定义
前置条件	无
后置条件	无

控制命令	int p9018_motion_get_dout_ex(int card_no, UINT* pValue)
功能描述	查询控制卡上所有数字量输出状态。
参数	card_no – 控制卡卡号，范围：0～15 pValue – 输出端口状态, 32 位无符号数，其中位 0~ 15 状态分别对应 DO[0]~ DO[15] 的状态； 1 表示导通（输出通道的晶体管集电结导通）； 零表示截至（输出通道的晶体管集电结截至）。
返回	参考返回值定义
前置条件	无
后置条件	无

控制命令	int p9018_motion_get_din(int card_no, int index, int* pValue)
功能描述	查询控制卡上单路数字量输入状态。
参数	card_no – 控制卡卡号，范围：0～15 index – 要设置的数字量输入编号，范围 0 ~ 15 pValue – 输入状态, 非零表示导通（输入通道光耦的集电结导通）； 零表示截至（输入通道光耦的集电结截至）。
返回	参考返回值定义
前置条件	无
后置条件	无

控制命令	int p9018_motion_get_din_ex(int card_no, UINT* pValue)
功能描述	查询控制卡上所有数字量输入状态。
参数	card_no – 控制卡卡号，范围：0～15 pValue – 输入端口状态, 32 位无符号数，其中位 0~ 15 状态分别对应 DI[0]~ DI[15] 的状态；非零表示导通（输入通道光耦的集电结导通）； 零表示截至（输入通道光耦的集电结截至）。
返回	参考返回值定义
前置条件	无
后置条件	无

2.8 位置比较器

控制命令	int p9018_cmp_enable(int card_no, int index, int enable, int active_level, int ref_source, int length)
功能描述	设置位置比较器参数。 当比较器被禁止时，其参考值队列内数据被清空、比较成功次数被清零。
参数	card_no – 控制卡卡号，范围：0～15 index – 比较器编号，默认使用 0 enable – 使能设置， 非零表示使能位置比较器； 零表示禁止位置比较器 active_level – 位置比较器条件满足时的输出脉冲电平，非零表示输出高电平； 零表示输出低电平 ref_source – 比较器参考位置源，0~7 分别表示 X、Y、Z、A、B、C、U、V 的反馈位置 length – 位置比较器条件满足时输出脉冲的持续时间，范围 1000~ 2000000， 单位为 ns
返回	参考返回值定义
前置条件	无
后置条件	无

控制命令	int p9018_cmp_set_ref(int card_no, int index, int pos)
功能描述	向位置比较器的参考位置队列添加一个参考位置。 如果队列已满，将无法保存新添加的数据，因此在向队列中添加数据前，最好先查询下队列的状态是否为满。
参数	card_no – 控制卡卡号，范围：0～15 index – 比较器编号，默认使用 0 pos – 要添加的参考位置，范围: -134217728 ~134217727
返回	参考返回值定义
前置条件	无
后置条件	无

控制命令	int p9018_cmp_get_status(int card_no, int index, MOT_CMP_STAT* pStatus)
功能描述	查询位置比较器的状态。
参数	card_no – 控制卡卡号，范围：0～15 index – 比较器编号，默认使用 0

	pStatus – 比较器状态缓冲区，结构体类型，其成员定义如下： queFull 为 1 表示队列满，为 0 表示队列未满； queEmpty 为 1 表示队列空，为 0 表示队列未空； queItem 表示队列内的参考值个数； curRef 表示当前使用的参考值； matchCount 表示已经成功比较的次数；比较器被禁止时，该值被清零。
返回	参考返回值定义
前置条件	无
后置条件	无

2.9 控制卡信息查询

控制命令	int p9018_motion_get_info(int card_no, UINT* buff, int count)
功能描述	查询控制卡信息。
参数	card_no – 控制卡卡号，范围：0～15 buff – 信息缓冲区，其定义如下： buff[0]为控制卡的卡号 buff[1]为控制卡上的控制轴数目 buff[2]为控制卡上的数字量输入输出数目 buff[4]为 FPGA 版本号， buff[5]为驱动程序版本号， buff[6]为函数库版本号 count – 缓冲区元素个数
返回	参考返回值定义
前置条件	无
后置条件	无

3 范例代码

下面是 PCI-9018 控制卡上 X 轴先执行原点查找,然后执行点位运动的 C++范例代码:

```
#include "stdafx.h"
#include <windows.h>       //Sleep, UINT, QueryPerformanceCounter, QueryPerformanceFrequency
#include <stdio.h>
#include <stdarg.h>
#include "PCI9018.h"

#define AXIS_HOMING_TIMEOUT    (30000) //允许查找原点的最长时间，单位ms

int capture_home(int axis, double vel, int use_index);
int wait_axis(int axis, int timeout_ms);
double etime();
int myPrint(const char *_fmt, ...);

/*! \fn int _tmain(int argc, _TCHAR* argv[])
 *    \brief entry point
 */
int _tmain(int argc, _TCHAR* argv[])
{
      int count = 0, board_id[16], rc = 0;
      int axis, card_no, pos = 0;
      int err = 0;

      rc = p9018_open(&count, board_id);
      if(rc)
      {
            myPrint("Open fails\n" );
            return -1;
      }

      if(count < 1)
      {
            myPrint("Find no card\n" );
            p9018_close();
            return -1;
      }

      card_no = board_id[0];
```

```
    axis = card_no*8 + 0; //操作控制卡上第一个控制轴
    pos = 10000;

    rc = p9018_axis_set_profile(axis, 1000.0, 100000.0, 1000000.0, 1000000.0, 1000000000.0);
    if(rc)
        myPrint("axis_set_profile[%d] fails, rc: %d\n", axis, rc);

    //先查找原点，成功后再执行点位运动
    rc = capture_home(axis, -5000.0, 0);
    if(rc){
        myPrint("capture_home[%d] fails(%d)\n", axis, rc);
    }
    else{
        rc = p9018_axis_abs_move(axis, pos);
        if(rc)
            myPrint("axis_abs_move[%d] fails, rc: %d\n", axis, rc);

        rc = wait_axis(axis, 0);
        if(rc){
            myPrint("wait_axis[%d] fails(%d)\n", axis, rc);
        }

    }

    //close all card
    p9018_close();
    return 0;
}

/*! \fn int wait_axis(int axis)
 *  \brief 等待控制轴停止运动（正常停止、因异常停止或超时）
 *
 *  \param axis - 轴号
 *  \param timeout_ms - 允许最长时间，单位:ms；大于零是检查超时；小于等于零时不检查超时
 *
 *  \return 0-控制轴表示正常停止运动（运行过程中未出现异常），-控制轴因出现异常而停止运动，-超时
 */
int wait_axis(int axis, int timeout_ms)
{
    int rc = 0;
    double t;
    MOT_AXIS_STAT axis_status;

    axis_status.error = 0;
    t = etime();
```

```
    while(true)
    {
        rc = p9018_axis_get_status(axis, &axis_status);
        if(!rc && axis_status.inpos)
            break;

        Sleep(1);    //释放系统资源

        if((timeout_ms > 0) && (etime() > (t + timeout_ms * 0.001)))
        {
            //超时
            return 2;
        }
    }
    //停止时未发生异常
    if(!axis_status.error)
        return 0;

    //因发生异常而停止
    return 1;
}

/*! \fn int capture_home(int axis, double vel, int use_index)
 *  \brief 控制轴二次查找原点范例
 *
 *  二次原点查找可兼顾速度快和精度高的优点
 *
 *  \param axis - 控制轴轴号
 *  \param vel - 原点查找速度，其符号决定方向
 *  \param use_index - 是否查找Index
 *
 *  \return 零表示运行成功，非零表示有错误出现
 */
int capture_home(int axis, double vel, int use_index)
{
    int rc = 0;
    MOT_AXIS_STAT axis_status;
    //控制轴查找原点前原点信号有效时控制轴回退的距离
    //使得选择该值，使得控制轴能够离开原点
    int step_dist = 2000;

    //控制轴第一次查找原点，如果原点信号已经有效，控制轴回退一段距离
    rc = p9018_axis_get_status(axis, &axis_status);
    if(!rc){
        if(axis_status.homeSwitch){
```

```
                rc = p9018_axis_incr_move(axis, (vel < 0.0)? step_dist: -step_dist);
                if(!rc){
                    rc = wait_axis(axis, 0);
                }
                if(rc){
                    myPrint("fails to move backward[%d], rc: %d\n", axis, rc);
                    return rc;
                }
            }
            Sleep(100);
            rc = p9018_axis_home(axis, vel, use_index);
            if(!rc){
                rc = wait_axis(axis, AXIS_HOMING_TIMEOUT);
            }
            if(rc){
                myPrint("fails to home[%d], rc: %d\n", axis, rc);
                return rc;
            }
        }

        if(rc){
            myPrint("fails to search home[%d], rc: %d\n", axis, rc);
            return rc;
        }

        //控制轴第二次查找原点，先回退一段距离，然后使用更低速度查找原点
        Sleep(100);
        rc = p9018_axis_incr_move(axis, (vel < 0.0)? step_dist: -step_dist);
        if(!rc){
            rc = wait_axis(axis, 0);
        }

        if(rc){
            myPrint("fails to move backward[%d], rc: %d\n", axis, rc);
            return rc;
        }
        Sleep(100);
        rc = p9018_axis_home(axis, vel * 0.1, use_index);
        if(!rc){
            rc = wait_axis(axis, AXIS_HOMING_TIMEOUT);
        }
        return rc;
}

/*! \fn double etime()
```

```
 *   \brief retrieves the high-resolution counter and convert it into seconds
 *   \param none
 *   \return timer value in seconds
 */
double etime()
{
	LARGE_INTEGER t, f;
	t.QuadPart = 0;
	f.QuadPart = 0;
	QueryPerformanceCounter(&t);
	QueryPerformanceFrequency(&f);
	if(f.QuadPart)
	{
		return (double)t.QuadPart/(double)f.QuadPart;
	}
	else
	{
		return 0.0;
	}
}

/*! \fn int myPrint(const char *_fmt, ...)
 *   \brief  调试信息输出（可修改代码来来定义输出不同设备）
 *
 *
 *   \param  变参类型
 *
 *   \return  格式化后的字符串长度，小于零时表示出错
 */
int myPrint(const char *_fmt, ...)
{
	int retval = 0;
	char buff[80+1];

	va_list args;
	va_start(args, _fmt);

	//retval = vsnprintf(buff, 80, _fmt, args);
	retval = vsnprintf_s(buff, sizeof(buff), _TRUNCATE, _fmt, args);	//safe version for VC++

	printf("%s", buff);		//output message to standard output
	//OutputDebugStringA(buff); //output message to system debugger

	va_end(args);
	return retval;
```

```
}
```

4 修订记录

日期	版本	修改说明
2019-11-2	1.00	创建