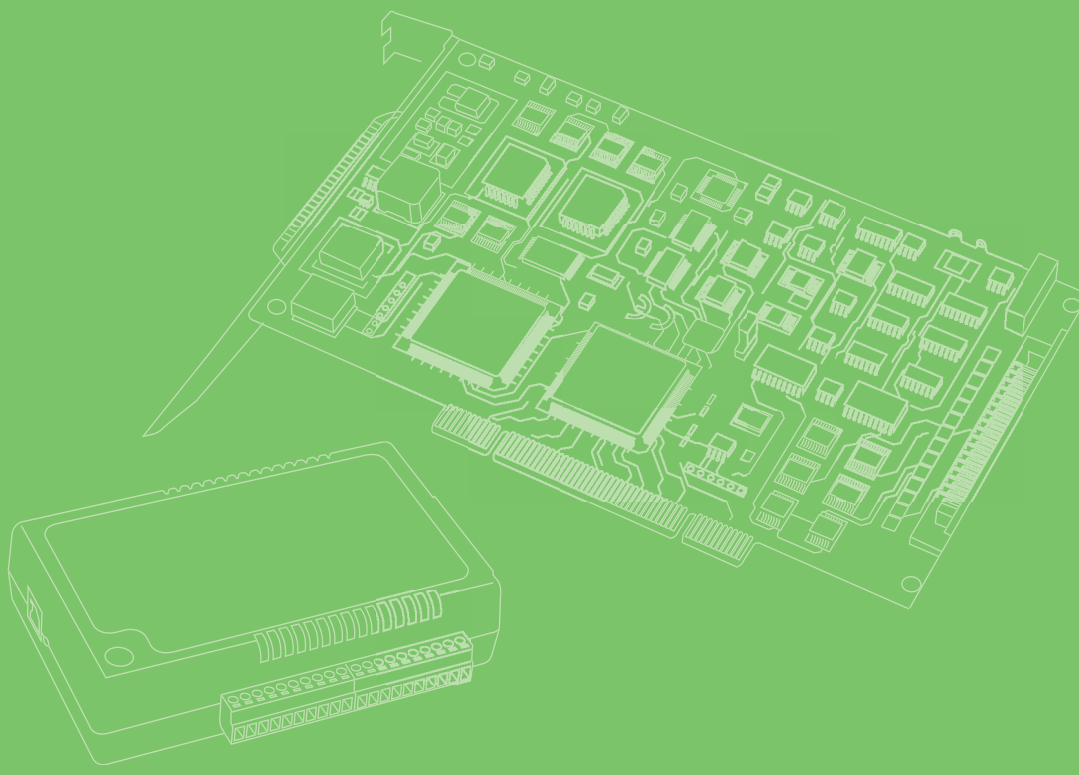


用户手册

PCI-1245L

SoftMotion PCI 控制器

ADVANTECH

Enabling an Intelligent Planet

版权声明

认可声明

PCI-1245L 用户手册中文第一版，参照 PCI-1245L 用户手册英文第一版。

符合性声明

CE

本设备已通过 CE 测试，符合以屏蔽电缆进行外部接线的环境规格标准。建议用户使用屏蔽电缆，此种电缆可从研华公司购买。如需订购，请与当地分销商联系。

料号：2003124530　　第一版

中国印刷　　2013 年 7 月

产品质量保证 （两年）

从购买之日起，研华为原购买商提供两年的产品质量保证。但对那些未经授权的维修人员维修过的产品不予提供质量保证。研华对于不正确的使用、灾难、错误安装产生的问题有免责权利。

如果研华产品出现故障，在质保期内我们提供免费维修或更换服务。对于出保产品，我们将会酌情收取材料费、人工服务费用。请联系相关销售人员了解详细情况。

如果您认为您购买的产品出现了故障，请遵循以下步骤：

1. 收集您所遇到的问题信息 （例如，CPU 主频、使用的研华产品及其它软件、硬件等）。请注意屏幕上出现的任何不正常信息显示。
2. 打电话给您的供货商，描述故障问题。请借助手册、产品和任何有帮助的信息。
3. 如果您的产品被诊断发生故障，请从您的供货商那里获得 RMA （Return Material Authorization） 序列号。这可以让我们尽快地进行故障产品的回收。
4. 请仔细地包装故障产品，并在包装中附上完整的售后服务卡片和购买日期证明（如销售发票）。我们对无法提供购买日期证明的产品不提供质量保证服务。
5. 把相关的 RMA 序列号写在外包装上，并将其运送给销售人员。

技术支持与服务

1. 有关该产品的最新信息，请访问研华公司的网站：
   http://support.advantech.com.cn
2. 用户若需技术支持，请与当地分销商、销售代表或研华客服圆心联系。进行技术咨询前，用户须将下面各项产品信息收集完整：
   - 产品名称及序列号
   - 外围附加设备的描述
   - 用户软件的描述 （操作系统、版本、应用软件等）
   - 产品所出现问题的完整描述
   - 每条错误信息的完整内容

包装清单

安装系统之前，用户需确认包装中含有本设备以及下面所列各项，并确认设备完好。若有任何不符，请立即与经销商联系。

- PCI-1245L 板卡
- 附带的 CD-ROM 光盘 （包括 DLL 驱动）
- 快速入门手册

安全措施 - 静电防护

为了保护您和您的设备免受伤害或损坏，请遵照以下安全措施：

- 操作设备之前，请务必断开机箱电源，以防触电。不可在电源接通时接触 CPU 卡或其它卡上的任何元件。
- 在更改任何配置之前请断开电源，以免在您连接跳线或安装卡时，瞬间电涌损坏敏感电子元件。

目录

第 1 章

概述

本章介绍 PCI-1245L 的基本信息、特殊特性以及详细规格。

PCI-1245L 是 4 轴的 SoftMotion PCI 总线控制器卡，专为各种电机自动化和其它机器自动化的广泛应用设计。板卡配有高性能 FPGA，其中包括 SoftMotion 算法，能够实现运动轨迹和时间控制，以满足精确运动中的同步应用需求。

PCI-1245L 支持以下 SoftMotion 特性：手轮及 MPG 控制、可编程的加速度和减速度；T&S 形速度曲线及 2 轴线性插补和同步起停等功能。

所有研华运动控制器均采用 “Common Motion API” 架构，采用统一的用户编程接口。程序员无需大规模修改应用码即可集成任何研华 SoftMotion 運動控制器。该架构能够帮助用户轻松维护和升级应用。

1.1 特性

PCI-1245L 具有以下特性：

- 4xAB 模式的编码器输入为 4 MHz，CW/CCW 模式的编码器输入为 1 MHz
- 脉冲输出高达 1 Mpps，可经由跳线设置成差动输出或是单端 +5V 输出
- 硬件紧急输入
- 看门狗定时器
- 可编程中断
- RDY 专用输入通道 & SVON/ERC 专用输出通道可切换用于通用输入和输出

1.2 应用

- 精密 X-Y-Z 位置控制
- 精密旋转控制
- 半导体封装组装设备，高速贴片试验机

1.3 安装指南

开始安装前，请确认已收到下列物品：

- PCI-1245L 板卡
- 用户手册
- 驱动和软件
- 实用程序
- ADAM-3956 端子板 （ 4 轴 ） 搭配 PCL-101100M 接线电缆 (100-pin to 100-pin SCSI ）；另外，也可使用两组 ADAM-3955 端子板 （ 2 轴 ） 搭配 PCL-10251 接线电缆 （ 100-pin to 2 x 50-pin SCSI 接头 ）
- 连接端子板和伺服驱动的任何 PCL-10153MJ3/PCL-10153YS5/PCL-10153PA5/PCL-10153PA5LS/PCL-10153DA2 电缆 （比如，可以支持三菱公司的 J3、安川电气的 Sigma V、松下公司的 A4/A5/MINAS A 和台達的 A2）
- 带 PCI 总线插槽的工业级 PC

1.4 附件

研华为产品提供了完整的附件组合。附件包括：

连接接线板的接线电缆

- PCL-10251 - PCL-10251 是一根 100 针转两组 50 针屏蔽电缆。为使信号质量更好，信号线采用双绞线方式布线。从而可以有效减少来自其他信号源的串扰和噪音。

- PCL-101100M - PCL-101100M是一根100针对100针屏蔽电缆。为使信号质量更好，信号线采用双绞线方式布线。从而可以有效减少来自其他信号源的串扰和噪音。

接线板

- ADAM-3955 - ADAM-3955 专为轻松连接伺服驱动而设计。接线板采用 2 轴设计。比如，如果用户使用 PCI-1245L 板卡，则需要两块接线板进行 4 轴控制。松下公司的 A4/A5/MINAS A、安川电气的 Sigma V、三菱公司的 J3 和台達的 A2 伺服均提供快速连接传输电缆。
- ADAM-3956 - ADAM-3956 专为轻松连接伺服驱动而设计。接线板采用 4 轴设计。比如，如果用户使用 PCI-1245L 板卡，僅需要一块接线板进行 4 轴控制。針對松下公司的 A4/A5 及 MINAS A、安川电气的 Sigma V 、三菱公司的 J3 和台達的 A2 伺服，均提供快速连接传输电缆。

连接伺服的传输电缆

- PCL-10153PA5 - PCL-10153PA5是一根50针电缆，连接ADAM-3955/ADAM-3956和松下 A4/A5 伺服。
- PCL-10153PA5LS - PCL-10153PA5LS是一根50针电缆，连接ADAM-3955/ADAM-3956和松下的 MINAS A 伺服。
- PCL-10153YS5 - PCL-10153YS5是一根50针电缆，连接ADAM-3955/ADAM-3956和安川电气的 Sigma V 伺服。
- PCL-10153MJ3 - PCL-10153MJ3是一根50针电缆，连接ADAM-3955/ADAM-3956和三菱公司的 J3 伺服。
- PCL-10153DA2 - PCL-10153DA2 是一根 50 针电缆，连接 ADAM-3955/ADAM-3956 和台達的 A2 伺服。

第 2 章

安装

本章介绍安装驱动和硬件的详细步骤。

2.1 打开包装

收到 PCI-1245L 包装后，请首先检查里面的物品。包装内应包括以下各项：

- PCI-1245L 板卡
- 所附光盘 （包含 DLL 驱动和用户手册）

PCI-1245L 卡的一些电子元件极易受到静电放电 （ESD）的损害。如果保护措施不当，则集成电路和某些元件极易被 ESD 损害。

将卡从静电屏蔽袋中取出之前，用户应按照以下步骤的指导来防止可能的 ESD 损害：

- 用手触摸机箱的金属部分来释放身体所附的静电，或者也可以使用接地母线。
- 打开静电屏蔽袋之前，使其接触机箱的金属部分。
- 取卡时，只能握住卡的金属托架。

将卡取出后，请首先：

- 检查卡上是否有明显的外部损伤 （元件松动或损坏等）。如果有明显损坏，请立即联系我们的服务部门或者当地销售代表。切勿将损坏的卡安装至系统。

另外，安装时也请注意以下事项：

- 用户还应避免接触带有静电的材料，如塑料、乙烯基和泡沫聚苯乙烯。
- 拿卡时请只握住卡的边缘。不要碰触接口或电子元件露在外部的金属针脚。

2.2 安装驱动

建议用户在安装 PCI-1245L 板卡之前，首先安装板卡驱动。

板卡的 DLL 驱动安装程序位于产品包装所附光盘中。按照以下步骤安装驱动软件：

1. 将产品所附光盘插入光驱。
2. 如果用户的系统开启了自动播放功能，安装程序将自动运行。

注！ *如果用户的系统没有启用自动播放功能，请使用 Windows Explorer 或 Windows Run 命令来执行光盘中的 “SETUP.EXE”。*

3. 根据操作系统选择合适的 Windows OS 选项。然后请按照指导逐步完成 DLL 驱动的安装。
4. 然后自动安装 PCI-1245L 运动实用程序。

如需驱动相关的更多信息，请参考设备驱动手册的在线版本：

Start\Advantech Automation\Motion \(Board Name)\

示例源代码可在相应安装文件夹下找到，如默认安装路径为：

\Program Files\Advantech\ Motion \(Board Name)\Examples

2.3 安装硬件

注！ *安装卡之前，请确认已安装了驱动。（请参考 2.2 节 “ 安装驱动 ”）*

DLL 驱动安装完成之后，用户即可将 PCI-1245L 卡插入计算机的任一 PCI 插槽。若有任何疑问，请参考计算机的用户手册或其它相关文档。请按照以下步骤安装卡。

1. 关掉计算机并断开连接至计算机的所有附件。

警告！ *安装／移除任何板卡、连接／断开任何电缆时，请关闭计算机电源。*

2. 断开连接到计算机后部的电源线和其它电缆。
3. 移除计算机顶盖。
4. 选择一个未占用的 +3.3/+5 V PCI 卡插槽。卸下将扩展槽盖固定在系统中的螺丝。保存好固定接口卡支架的螺丝。
5. 小心握住 PCI-1245L 板卡上部边缘。将支架上的孔与扩展槽上孔对齐，将金手指接口与扩展槽插槽对齐。将板卡轻轻插入插槽中并固定。确保板卡牢固卡入插槽中。请避免用力过大，否则也许会损坏卡。
6. 用螺丝将 PCI 卡托架固定在计算机后面板导轨上。
7. 将所需附件 （电缆、接线端子等）连接至 PCI 卡。
8. 重新放回计算机顶盖，并重新连接步骤 2 中断开的电缆。
9. 开启计算机。

第 3 章

信号连接

本章介绍输入和输出信号的连接信息。

3.1 PCI-1245L I/O 接口针脚定义

PCI-1245L 的 I/O 接口是一个 100 针接口，可通过 PCL-10251 屏蔽电缆连接两个 ADAM-3955 端子板或是通过 PCL-101100M 屏蔽电缆连接一个 ADAM-3956 端子板。

图 3.1 为 PCI-1245L 上 100 针 I/O 接口的针脚定义。表 3.1 为 I/O 接口信号说明。

VEX	1	51	VEX
EMG	2	52	NC / EMG
X_LMT+	3	53	Z_LMT+
X_LMT-	4	54	Z_LMT-
X_IN1	5	55	Z_IN1
X_IN2 / RDY	6	56	Z_IN2 / RDY
X_ORG	7	57	Z_ORG
Y_LMT+	8	58	U_LMT+
Y_LMT-	9	59	U_LMT-
Y_IN1	10	60	U_IN1
Y_IN2 / RDY	11	61	U_IN2 / RDY
Y_ORG	12	62	U_ORG
X_INP	13	63	Z_INP
X_ALM	14	64	Z_ALM
X_ECA+	15	65	Z_ECA+
X_ECA-	16	66	Z_ECA-
X_ECB+	17	67	Z_ECB+
X_ECB-	18	68	Z_ECB-
X_ECZ+	19	69	Z_ECZ+
X_ECZ-	20	70	Z_ECZ-
Y_INP	21	71	U_INP
Y_ALM	22	72	U_ALM
Y_ECA+	23	73	U_ECA+
Y_ECA-	24	74	U_ECA-
Y_ECB+	25	75	U_ECB+
Y_ECB-	26	76	U_ECB-
Y_ECZ+	27	77	U_ECZ+
Y_ECZ-	28	78	U_ECZ-
X_IN4 / JOG+	29	79	Z_IN4 / JOG+
X_IN5 / JOG-	30	80	Z_IN5 / JOG-
Y_IN4 / JOG+	31	81	U_IN4 / JOG+
Y_IN5 / JOG-	32	82	U_IN5 / JOG-
EGND	33	83	EGND
X_OUT4	34	84	Z_OUT4
X_OUT5	35	85	Z_OUT5
X_OUT6 / SVON	36	86	Z_OUT6 / SVON
X_OUT7 / ERC	37	87	Z_OUT7 / ERC
X_CW+/PULS+/+5V	38	88	Z_CW+/PULS+/+5V
X_CW- / PULS-	39	89	Z_CW- / PULS-
X_CCW+/DIR+/+5V	40	90	Z_CCW+/DIR+/+5V
X_CCW- / DIR-	41	91	Z_CCW- / DIR-
EGND	42	92	EGND
Y_OUT4	43	93	U_OUT4
Y_OUT5	44	94	U_OUT5
Y_OUT6 / SVON	45	95	U_OUT6 / SVON
Y_OUT7 / ERC	46	96	U_OUT7 / ERC
Y_CW+/PULS+/+5V	47	97	U_CW+/PULS+/+5V
Y_CW- / PULS-	48	98	U_CW- / PULS-
Y_CCW+/DIR+/+5V	49	99	U_CCW+/DIR+/+5V
Y_CCW- / DIR-	50	100	U_CCW- / DIR-

图 3.1： PCI-1245L 的 I/O 接口针脚定义

表 3.1：I/O 接口信号描述

信号名称	参考	方向	说明
VEX	–	输入	外部电源 （12 ~ 24 VDC）
EMG	–	输入	紧急停止 （适用于所有轴）
LMT+	–	输入	+ 方向极限
LMT–	–	输入	– 方向极限
RDY	–	输入	伺服就绪
ORG	–	输入	原点位置
INP	–	输入	到位信号
ALM	–	输入	伺服报警
ECA+	–	输入	编码器相位 A+
ECA–	–	输入	编码器相位 A–
ECB+	–	输入	编码器相位 B+
ECB–	–	输入	编码器相位 B–
ECZ+	–	输入	编码器相位 Z+
ECZ–	–	输入	编码器相位 Z–
EGND	–	–	信号地
IN	EGND	输入	通用数字量输入
OUT	EGND	输出	通用数字量输出
SVON	EGND	输出	伺服使能
ERC	EGND	输出	清除误差计数器
CW+ / PULS+	EGND	输出	输出脉冲 CW/ 脉冲 +
CW– / PULS–	EGND	输出	输出脉冲 CW/ 脉冲 –
CCW+ / DIR+	EGND	输出	输出脉冲 CCW/DIR+
CCW– / DIR–	EGND	输出	输出脉冲 CCW/DIR–

注！

1. *X、Y、Z 和 U 分别表示每个轴的 ID。*
2. *RDY 专用输入通道设计为可切换，并支持通用输入通道应用。*
3. *SVON 和 ERC 专用输出通道设计为可切换，并支持通用输出通道应用。*
4. *IN4 有三种切换功能：通用输入、JOG+ 和 MPG+ （手动脉冲器）。*
5. *IN5 有三种切换功能：通用输入、JOG– 和 MPG– （手动脉冲器）。*

3.2 DIP 开关的位置

图 3.2 为 PCI-1245L 上每个 DIP 开关的名称和位置。开关用于设置板卡 ID。

BoardID 开关

PCI-1245L 板卡有一个内置 DIP 开关 （SW1），可用于定义每块板卡中运动实用程序的唯一识别码。用户可参考下表 3.2 在寄存器中定义唯一识别码。当机箱内安装多块板卡时，板卡 ID 开关可通过每块卡的设备编号来帮助用户识别各个卡。

板卡 ID 开关的出厂设置为 0。如果用户需要将其更改为其它数字，请参考表 3.2 设置 SW1。

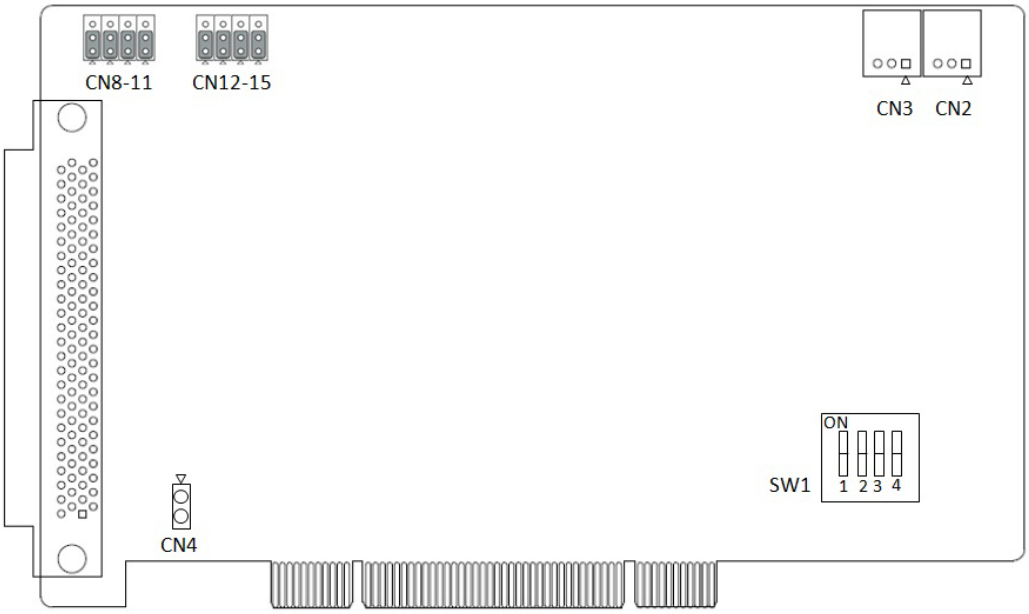


图 3.2： 跳线和 DIP 开关的位置

表 3.2：BoardID 设置

板卡 ID 设置 （SW1）				
板卡 ID （Dec.）	开关位置			
	ID3 (1)	ID2 (2)	ID1 (3)	ID0 (4)
*0	●	●	●	●
1	●	●	●	❍
:				
14	❍	❍	❍	●
15	❍	❍	❍	❍
❍= 闭合	●= 打开	* = 默认		

3.3 输出脉冲 [CW±/PULS±、CCW±/DIR±]

脉冲命令有两种类型：一种是顺时针 / 逆时针模式；另一种是脉冲 / 方向模式。CW+/PULS+ 和 CW-/PULS- 是差分信号对，CCW+/DIR+ 和 CCW-/DIR- 是不同的信号对。脉冲输出模式的默认设置为脉冲 / 方向。用户可通过编程修改输出模式。

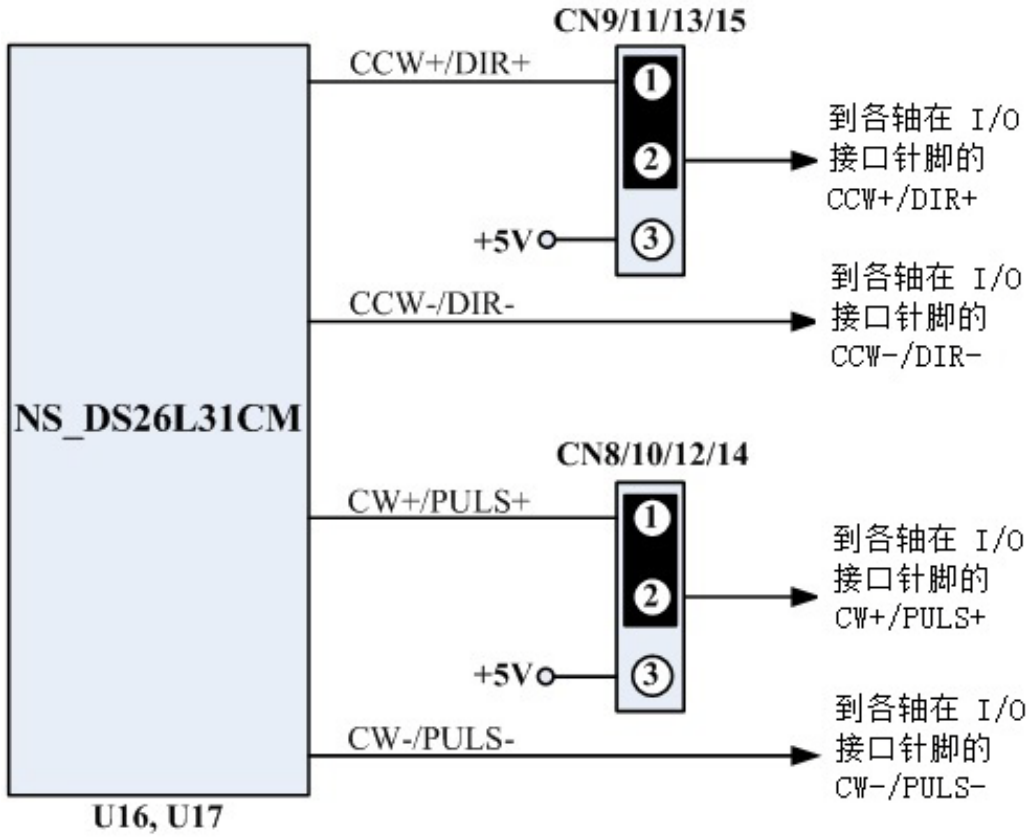


图 3.3： 驱动脉冲的输出讯号回路

图 3.3 所示的电路为预设的输出设定（CN8-15 的第一与第二接脚短路）是微分（差动）输出模式。若需改成单端输出，使用者可以变更跳线接头，当 CN8-15 的第二与第三接脚短路，使任一轴在 I/O 接口针脚上的 CCW+/DIR+ 输出变成 +5V，例如：一起将 CN12 和 CN13 的第二与第三接脚短路时，第 Z 轴的在图 3.1 的 I/O 接口针脚之 CCW+/DIR+ 和 CW+/DIR+ 输出会变成 +5V。

注意！

对于步进马达的驱动而言，CN14 与 CN15 需一起用来调整第 X 轴的输出模式； CN10 与 CN11 需一起用来调整第 Y 轴的输出模式； CN12 与 CN13 需一起用来调整第 Z 轴的输出模式； CN8 与 CN9 需一起用来调整第 U 轴的输出模式。PCB 板上有在 CN8-15 上加注对应的轴编号，例如：0 表示第 X 轴、1 表示第 Y 轴、2 表示第 Z 轴、3 表示第 U 轴。

注意！

当调整 CN8-15 使 I/O 接口针脚上的 CCW+/DIR+ 输出 +5V 电压时，请避免使 +5V 电压过载，这 +5V 电压能提供的最大电流量（四轴总合）为 120mA。

表 3.3：CN8-15 跳线设置

跳线	CN8	CN9	CN10	CN11	CN12	CN13	CN14	CN15
	I/O 接口针脚輸出							
	Pin 99	Pin 97	Pin 49	Pin 47	Pin 90	Pin 88	Pin 40	Pin 38
	U_CCW+/ DIR+	U_CW+/ PULS+	Y_CCW+/ DIR+	Y_CW+/ PULS+	Z_CCW+/ DIR+	Z_CW+/ PULS+	X_CCW+/ DIR+	X_CW+/ X_CW+/

表 3.3：CN8-15 跳线设置

+5V +5V +5V +5V +5V +5V +5V +5V

图 3.4： 光耦合器接口

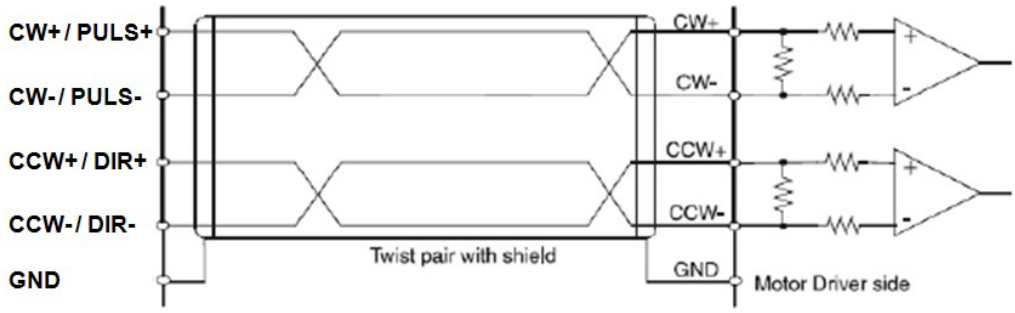


图 3.5： 线性驱动接口

3.4 行程限位开关输入［ LMT+/- ］

行程限位开关用于保护系统。 该输入信号通过光耦合器和 RC 过滤器连接。 采用限位开关时，外部电源 VEX DC 12 ~ 24 V 将成为光耦合器的电压源。 因此，将启用越程功能。

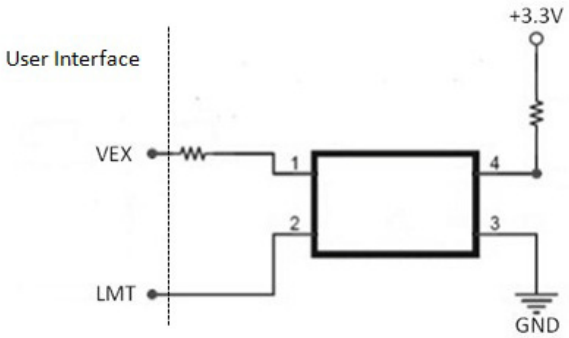


图 3.6： 限位输入信号的电路图

3.5 伺服就绪信号［RDY］

这是一个通用数字量输入，用于检查伺服驱动连接的伺服就绪状态。比如，在执行任何命令之前，用户可以检查状态。用户还能够将该 RDY 作为其它应用的通用输入。

3.6 原点位置 [ORG]

原点位置定义每个轴的原始位置或原始信号。有关编程设置，请参考第六章。

3.7 到位信号 [INP]

到位范围 （或偏差）通常由伺服驱动定义。当电机运动并在该范围 （或偏差）内汇聚，伺服驱动将发出信号表示电机处于指到位置。

3.8 伺服误差 & 报警 [ALM]

该输入来自伺服驱动，将生成报警信号提示操作错误。

3.9 编码器输入 [ECA+/-、ECB+/-、ECZ+/-]

编码器反馈信号到达时，将 ECA+/ECA- 连接至编码器输出的相位 A。这是一个差分对。同样，也适用于 ECB+/- 和 ECZ+/-。PCI-1245L 的默认设置为正交输入 （4xAB 相位）。下图为建议接口图：

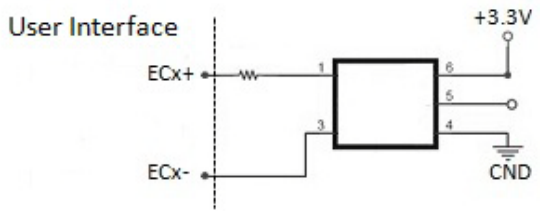


图 3.7： 编码器反馈的电路图

在上述电路图中，PCI-1245L 采用高速光耦合器用于隔离。源的编码器输出可为差分模式或开集模式。可接受的最大 4xAB 相位反馈频率约为 4 MHz。

3.10 紧急停止输入 （EMG）

紧急停止输入信号启用时，所有轴的驱动脉冲输出均停止。

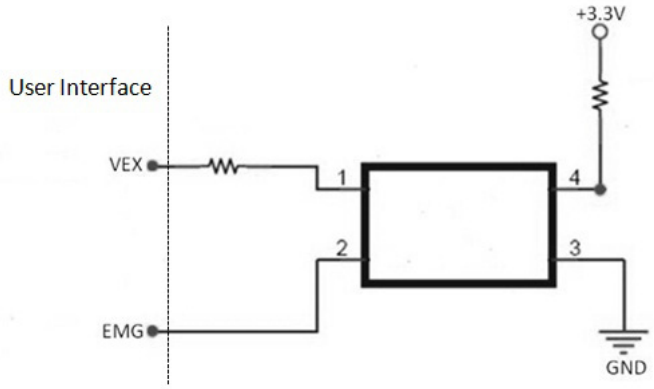


图 3.8： 紧急停止输入信号的电路图

该信号应用于与外部电源 DC 12 ~ 24 V 的组合应用中。由于光耦合器和 RC 过滤器的延迟，电路的响应时间约为 0.25 毫秒。

3.11 外部电源输入 （VEX）

每个轴的所有输入信号都需要外部电源。 请按照要求使用 DC 12 ~ 24 V 电压。

注意！ *请勿直接将 VEX 连接至电感性负载。*

3.12 激活开启伺服 [SVON]

SVON 会生成一个数字量输出，激活伺服驱动以进入运动状态。

3.13 清除伺服误差计数器 [ERC]

伺服驱动可生成偏差计数器清除信号，板卡可接收该信号作为通用输入。以下情况将清除计数器：返回原点、紧急停止情况、伺服报警以及行程限位激活。

3.14 数字量输入和输出

提供 PCI-1245L 的数字量输入和数字量输出之外部配线建议。

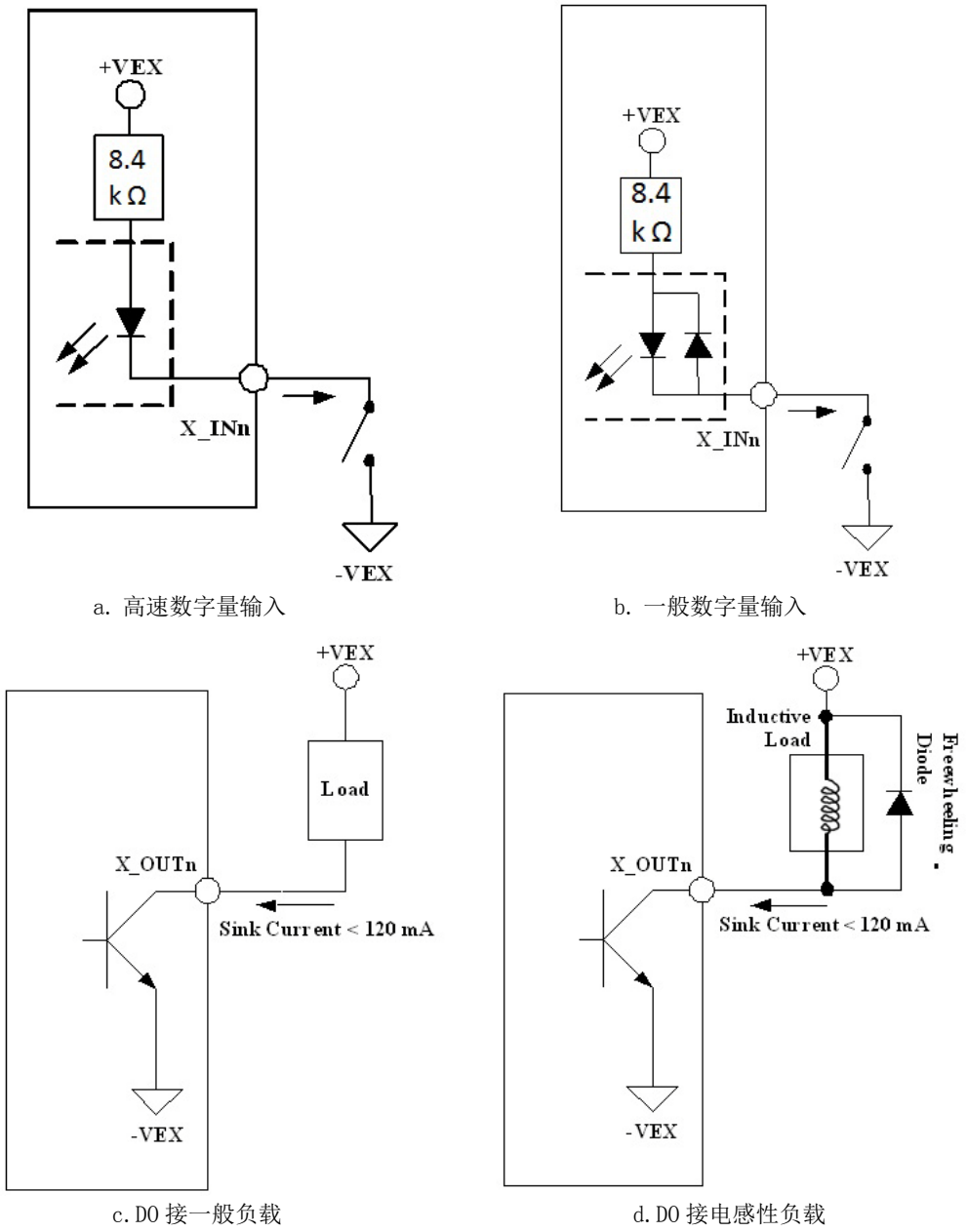


图 3.9： 数字量输出和输入

3.15 JOG 和 MPG

针脚定义 – X_IN4 & X_IN5 可支持 JOG 和 MPG 模式。这两个针脚可互相切换。X_IN4 有三种功能：通用数字量输入、JOG+ 和 MPG+。X_IN5 同样也有三种功能：通用数字量输入、JOG- 和 MPG-。同理，Y、Z、U 轴也具有同样功能。

3.16 多块板卡同时开始和停止

连接每块板卡上的 CN2 和 CN3 可支持多块板卡同时开始和停止。有关同时开始和停止的功能调用，请参考第六章。

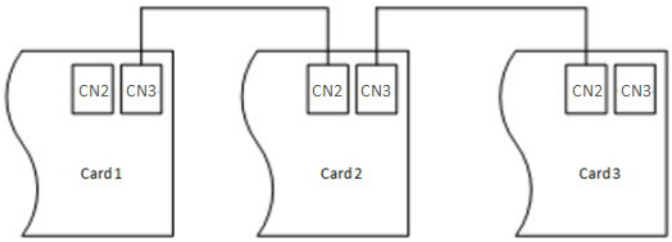


图 3.10： 多块板卡连接

第 4 章

通用运动 API

本章介绍通用运动的架构和理念。

4.1 通用运动架构简介

为了统一所有研华运动设备的用户接口，所有研华运动设备采用了新的软件架构，名为“ 通用运动架构 ”。该架构定义了所有用户接口和具有的所有运动功能，包括单个轴和多轴。这种统一的编程平台使用户能够以相同的方式操作设备。

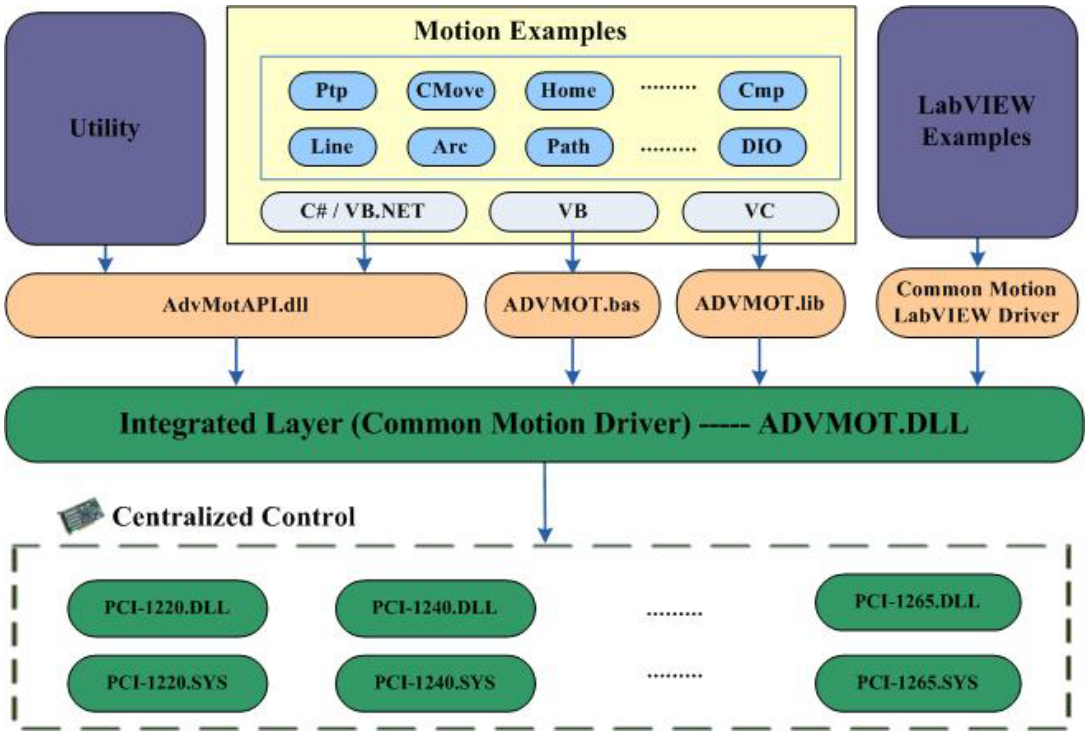
该架构包括三层：设备驱动层、整合层和应用层。用户无需了解如何操作特定设备的特定驱动，只需了解通用运动驱动即可。即使支持该架构的设备发生变化，应用也无需修改。

研华通用运动（ACM）架构定义了三种类型的操作对象：设备、轴和群组。每个类型都有自己的方法、属性和状态。

请按照以下步骤开始单轴运动：

开启设备 -> 开启该设备的一个轴 -> 配置该轴实例 -> 开始运动。

所有操作可通过调用相应 ACM API 完成。通用运动架构规定了设备、轴和群组的一般调用流程。有关详细信息，请参考**调用流程**章节内容。



4.2 设备编号

设备编号由 32 位组成：

第四个字节	第三个字节	第二个高字节	第二个低字节	第一个字节
主卡 / 设备类型 ID	主卡 / 设备板卡 ID （或 BaseAddr）		环路	从卡 ID

- 第四个字节
  主卡 / 设备类型 ID （指主设备类型 ID 表）。
- 第三个字节和第二个高字节：
  主卡 / 设备板卡 ID （或基地址）。
- 第二个低字节：
  远程设备使用的主环路编号将 0 作为本地设备的默认值。
- 第一个字节：
  远程设备使用的从卡 ID 将 0 作为本地设备的默认值。

本地设备编号

第四个字节	第三个字节	第二个高字节	第二个低字节	第一个字节
主卡类型 ID	板卡 ID （或 BaseAddr）		0	0

比如，PCI-1245 的 BoardID 为 1，设备编号 （十六进制）则为：

27	001	0	0

因此，设备编号为 0x27001000。

4.3 API 和属性的命名规则

命名规则基于三个对象：设备对象、轴对象和群组对象。用户将在 API 中发现很多缩写。缩写及其含义如下表所示：

表 4.1：缩写及其含义

缩写	全称	备注
PPU	每个单元脉冲	运动的一个虚拟单元
Dev	设备	
Ax	轴	
Gp	群组	多个轴
Mas	主	基于通信机制的设备的主轴或主板卡
Daq		AI/AO/DI/DO 的共用名称
Rel	相对	
Abs	绝对	
Cmd	命令	
Vel	速度	
Acc	加速度	
Dec	减速度	
Emg	紧急	紧急停止
Sd	放慢	
Info	信息	
Cmp	比较	
Inp	到位	
EZ	编码 Z	
EL	硬件限位	
Mel	负向限位	
Pel	正向限位	
Org	原点	
Ext	外部	
FT	特性	特性属性
CFG	配置	配置属性
PAR	参数	参数属性
Ipo	插补	
Chan	通道	

API 的命名规则

API 的命名规则如下：

- Acm_DevXXX：表示该 API 将执行设备功能，如设备属性设置。
  如 Acm_DevSetProperty。
- Acm_DaqXXX：表示该 API 将执行 DI、DO、AI 或 AO 的功能。
  如 Acm_DaqDiGetByte。
- Acm_AxXXXX：表示该 API 将执行轴功能，如单轴运动、返回原点。
  如 Acm_AxHome。
- Acm_GpXXXX：表示该 API 将执行群组的功能，如插补运动。
  如 Acm_GpMoveLinearRel。

属性的命名规则

属性由三种类型：特性、配置和参数。

特性：特性属性和硬件特性有关。命名规则如下：

- FT_DevXXX：用于设备，如 FT_DevAxisCount。
- FT_DaqXXX：用于 DI、DO、AI 和 AO，如 FT_DaqDiMaxChan。
- FT_AxXXX：用于轴对象，如 FT_Ax。

配置：配置属性值可变化，但不会经常变化。

- CFG_DevXXX：用于设备，如 CFG_DevBoardID。
- CFG_AxXXXX：用于轴，如 CFG_AxMaxVel。
- CFG_DaqXXX：用于 DI、DO、AI 和 AO，如 CFG_DaqDiMaxChan。
- CFG_GpXXXX：用于群组对象，如 CFG_GpAxisInGroup。

参数：参数属性值会经常变化。

- PAR_DevXXX：用于设备。
- PAR_AxXXXX：用于轴，如 PAR_AxVelLow。
- PAR_DaqXXX：用于 DI、DO、AI 和 AO。
- PAR_GpXXXX：用于群组，如 PAR_GpGroupID。

第 5 章

测试工具

本章结合图示综合介绍测试工具。

5.1 简介

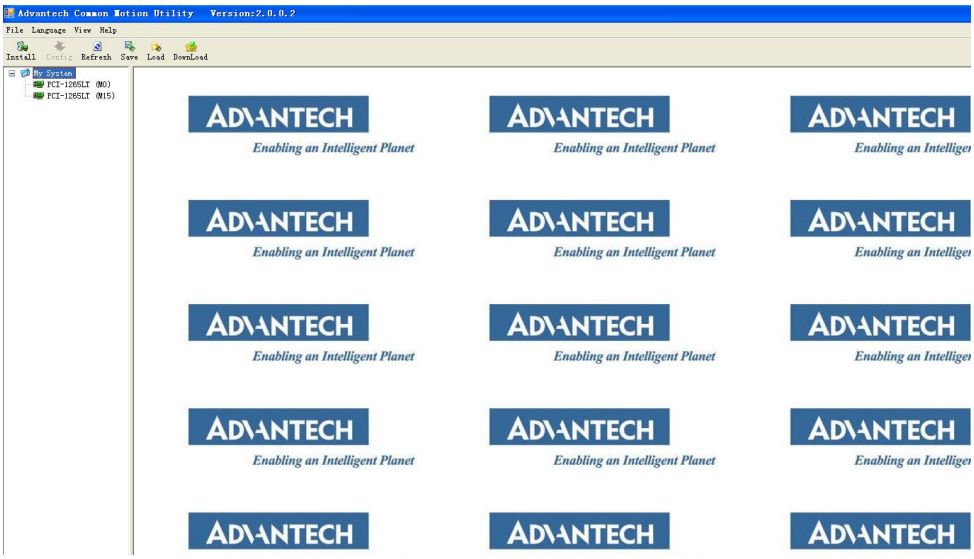
测试工具按照通用运动架构由 .Net 控件类库开发。.Net 控件类库包括组件（Device、Axis 和 Group）以及控件 （AxisSetupView、AxisScopeView、AxisDiagView、GroupPathView 和 GroupSpeedView）。新版测试工具与旧版 AdvMotionUtility 界面风格一致，功能兼容。 新版测试工具支持 PCI-1220U、PCI-1240U、PCI-1245L 和 PCI-1245/1245V/1245E/1265/1285/1285E 系列产品。

5.1.1 内容

根据操作顺序，将介绍以下接口：

1. Main Form：包括主菜单、工具栏和设备树。
2. Single-axis Motion：主要介绍单轴的 I/O 和属性配置、状态和运动操作 （点对点 （PTP）/ 连续 / 返回原点运动）。
3. Multi-axis Motion：主要介绍轴组 （Group）的插补运动操作，包括直线运动。
4. Synchronized Motion：主要介绍同步运动操作。
5. Digital Input：展示 Device 的数字输入端口状态。
6. Digital Output：展示 Device 的数字输出端口状态。

5.2 Main Form



5.2.1 主窗体

5.2.1.1 File

单击 [Exit] 终止该进程。

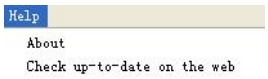
5.2.1.2 Language

通过该菜单可切换测试工具的语言。测试工具支持三种语言：英文、简体中文和繁体中文。选择一种语言后，相应的菜单项为勾选状态。关闭测试工具时，所选语言会将当前选择的 Utility 操作语言保存至注册表。再次打开时，测试工具的语言将为上次使用的语言。

5.2.1.3 View

该菜单允许用户显示 / 隐藏工具栏、状态栏和设备树。如果可以看到工具栏 / 状态栏 / 设备树，则对应菜单项为勾选状态。

5.2.1.4 Help

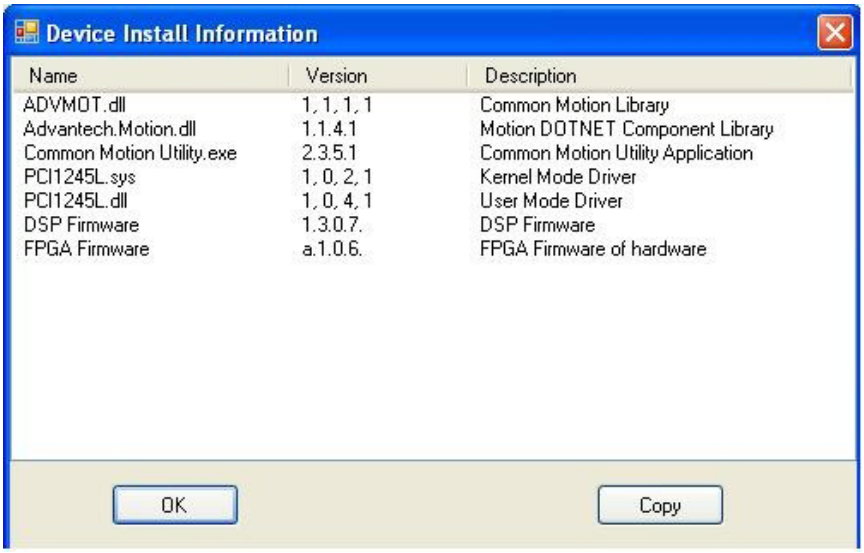


[About]菜单项提供设备驱动和测试工具的版权声明信息。单击[Check up-to-date on the web]将链接到公司的网站，通过比较 “Install” 界面的的版本信息检查固件、驱动和测试工具是否为最新版。

5.2.2 工具栏

5.2.2.1 Install

单击 [Install] 将弹出一个新窗口，显示驱动、硬件、固件和测试工具的版本信息。



点击 [Copy]，前面两列的信息将被保存到剪贴板中，方便客户将版本信息保存到 Word/Txt 中。

第一栏为名称，第二栏为版本号，第三栏为说明信息。ADVMOT.dll 是运动板卡的通用开发接口，Advantech.Motion.dll 为 DOTNET 运动控件程序集。Common Motion

Utility.exe 是正在运行的测试工具。第四行和第五行是驱动文件 （内核模式和用户模式），取决于设备类型。第六行为 DSP 固件，第七行为硬件的 FPGA。

注意！ *PCI-1245L 无 DSP，是基于 FGGA 的运动控制卡。*

5.2.2.2 Refresh

该按钮用于刷新功能。单击［Refresh］将重新加载设备树。操作后，默认没有选择任何设备。

5.2.2.3 Save

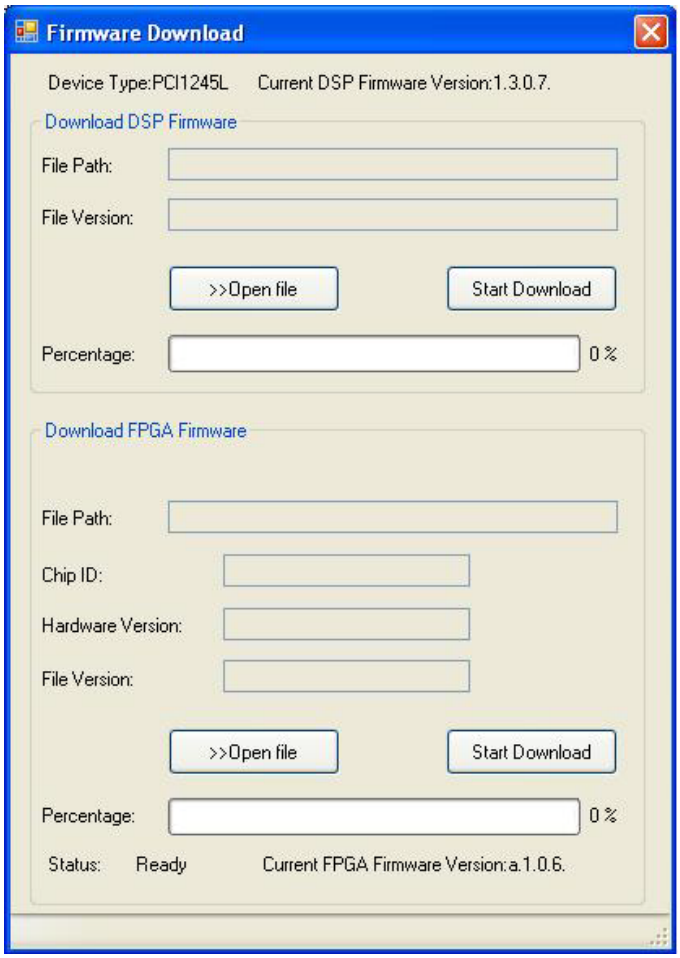
该按钮用于保存所选设备的轴的全部属性。

5.2.2.4 Load

该按钮用于导入所选设备的全部轴的配置信息。选择设备后，单击该按钮将出现“Open Dialog” 对话框。选择之前导出的配置文件并单击［OK］，用户即可将配置文件导入到设备硬件。

5.2.2.5 Download

PCI-1245L 运动控制器。单击设备后，用户将看到如下界面。



该工具按钮可实现 FPGA Firmware 的下载。PCI-1245L 无 DSP Firmware 下载功能。需要注意的是，FPGA Firmware 下载后，需要关机后再重启才真正更新。

对话框的顶部显示当前设备类型、设备名称及固件版本。单击［Open File］选择获取的最新固件文件。单击［Start Download］将激活硬件的下载过程，进度栏将显示任务进程。

注！

1. *单击［Start Download］后，下载固件至硬件过程中，对话框将不能关闭。*
2. *下载过程中，如果由于断电或其它问题导致下载过程未能完成，硬件需要返回研华公司进行固件升级。*

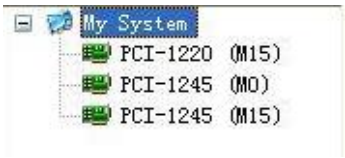
5.2.2.6 Hide Tree

为方便用户隐藏／显示设备树，提供此工具按钮。

若 Device Tree 目前显示，则点击将隐藏 Device Tree，按钮上的文字变为 “Show Tree”；

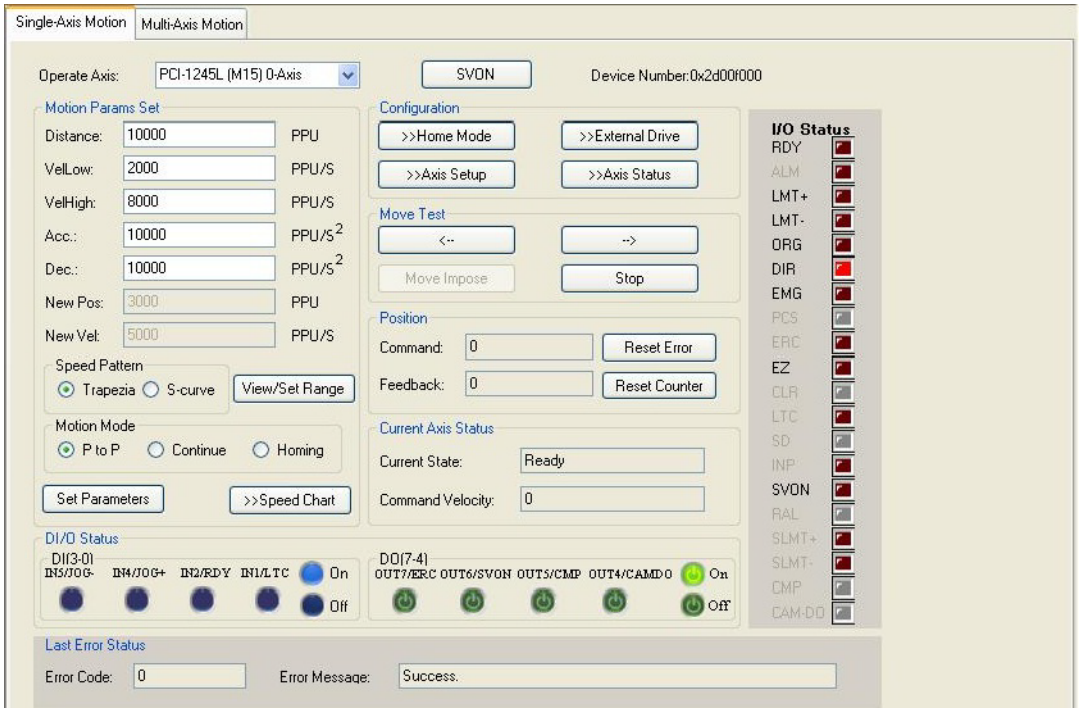
若 Device Tree 已隐藏，则点击将显示 Device Tree，按钮上的文字变为 “Hide Tree”。

5.2.3 设备树



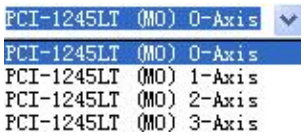
单击设备树中的任一设备，即可看到操作界面。

5.3 Single-Axis Motion



5.3.1 Operate Axis

选择操作轴。单击复选框下拉列表图标，将显示所选设备的全部轴：



5.3.2 Motion Params Set

完成操作参数设置后，单击 [Set Parameters] 将参数值设置到设备中。

5.3.2.1 基本参数设置

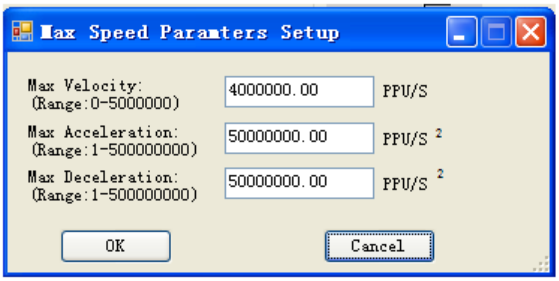
主要包括以下设置：点对点运动距离 （Distance）、单轴运动的初速度 （VelLow）、运行速度 （VelHigh）、加速度 （Acc.）、减速度 （Dec.）及叠加运动 （Move Impose）的运动距离 （New Pos.）和运动速度 （New Vel）。

5.3.2.2 Speed Pattern

设置运动的速度模式，可设置为梯形 （Trapezi）或 S 形 （S-curve）。

5.3.2.3 View/Set Range

单击 [View/Set Range] 检查或设置最大速度、加速度和减速度。对话框如下所示。



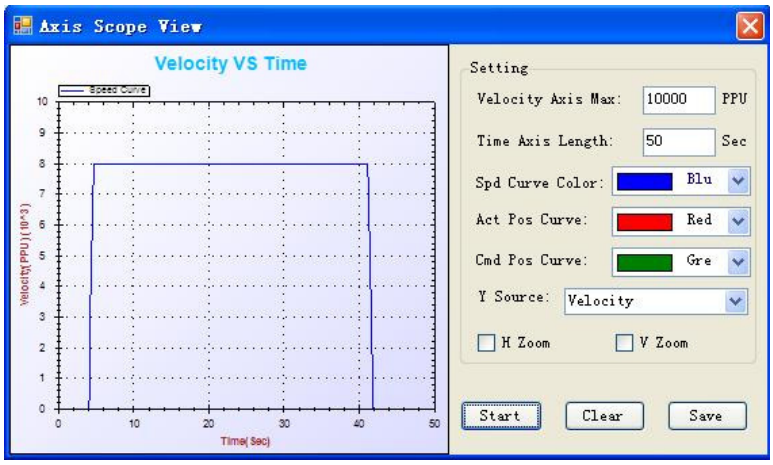
注！ *单轴运动的 VelHigh 不能高于 Max Velocity； Acc. 不能高于 Max Acceleration 且 Dec. 不能高于 Max Deceleration。*

5.3.2.4 Move Mode

选择运动模式。单轴运动有三种运动模式： P to P （点对点）、Continue （恒速连续运动）以及 Homing （返回原点运动）。

5.3.2.5 Speed Chart

单击 [Speed Chart] 可看到速度曲线图。

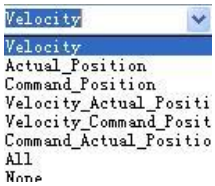


其中，右边为设置栏和操作按钮项，左边为单轴运动的运动 / 速度曲线图。

5.3.2.5.1 设置

设置项目如下：

1. Velocity Axis Max：设置最大垂直坐标。
2. Time Axis Length：设置最大水平坐标 （水平轴宽度，单位为 Sec）。
3. Spd Curve Color：设置速度曲线的颜色。
4. Act Pos Curve：设置实际位置曲线的颜色。
5. Cmd Pos Curve：设置理论位置曲线的颜色。
6. Y Source：垂直坐标的数据源。用户可按照下图所示选择速度、理论位置和实际位置的一种或任意组合。



7. H Zoom：若勾选，表示水平缩放功能启用，用户可通过鼠标放大选择区域。
8. V Zoom：若勾选，表示垂直缩放功能启用，用户可通过鼠标放大选择区域。

设置项目编辑完成后，所有设置值将在鼠标离开编辑框时生效。

5.3.2.5.2 Start

单击 [Start]，图形框即开始绘制曲线。如果轴在运动中，那么用户可以看到运动轨迹。单击之后，[Start] 按钮上的文字将变成 “Stop” ；单击 [Stop]，曲线图绘制将停止，且文字将变回 “Start” 。

5.3.2.5.3 Clear

单击 [Clear] 将清除图形框中当前显示的曲线图。

5.3.2.5.4 Save

单击 [Save]，路径的曲线图将存为 .png、.gif、.jpg、.tif 或 .bmp 格式。

5.3.3 SVON

单击 [SVON]，轴的伺服将开启，且按钮上的文字将变为 “SVOFF” ；单击 [SVOFF]，轴的伺服将关闭，且文字将变回 “SVON” 。

5.3.4 Configuration

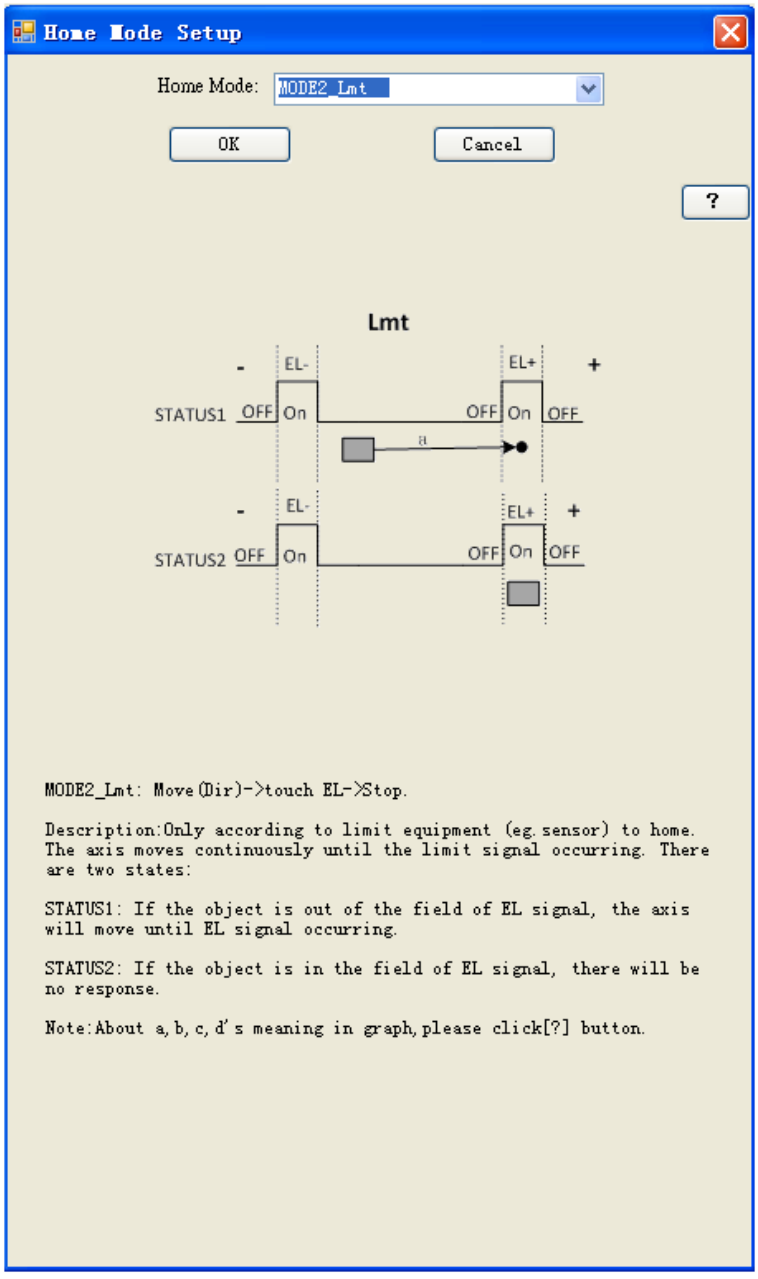
包括轴的回 Home 模式 （Home Mode）、外部驱动 (External Drive) 模式、属性配置和 I/O 状态显示。

5.3.4.1 Home Mode

进行该轴的 Homing 操作前，需选择回 Home 模式。板卡提供了 16 种模式，是 ORG （返回原点）、Lmt （返回限位点）和 EZ （找到 Z 相位）的一种或任意组合。

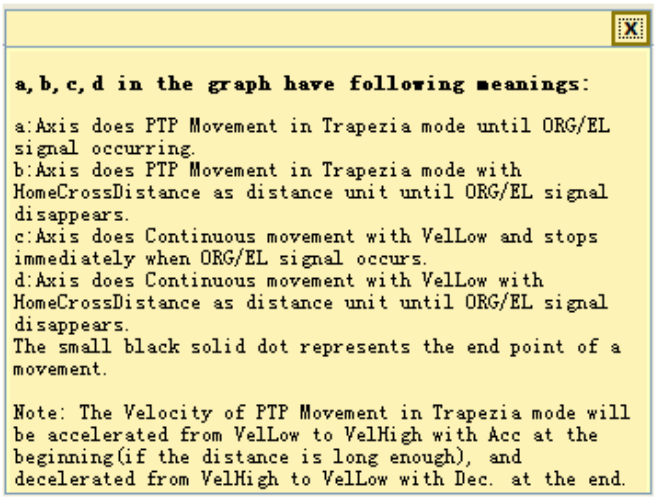
有关详细信息，请参考通用 API 编程指南中的 Acm_AxHome 函数。

单击 [Home Mode] 将出现如下对话框：



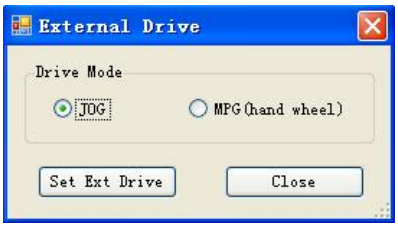
用户可从下拉列表框中选择任一模式，下面有相应的图解说明。用户可单击 [OK] 选择 “Home Mode” 下拉列表框中的一种模式，或单击 [Cancle] 取消操作。默认设置为 "Mode1_Abs"。

点击中间的 [?] 按钮将出现一个帮助窗体，说明图形中 a，b，c，d 和小黑实心圆点 的含义，需要时可点击查看，如下图：



5.3.4.2 External Drive

单击 [External Dirve] 将出现如下对话框，用户可选择一种外部驱动模式（JOG/MPG）来操作外部驱动。

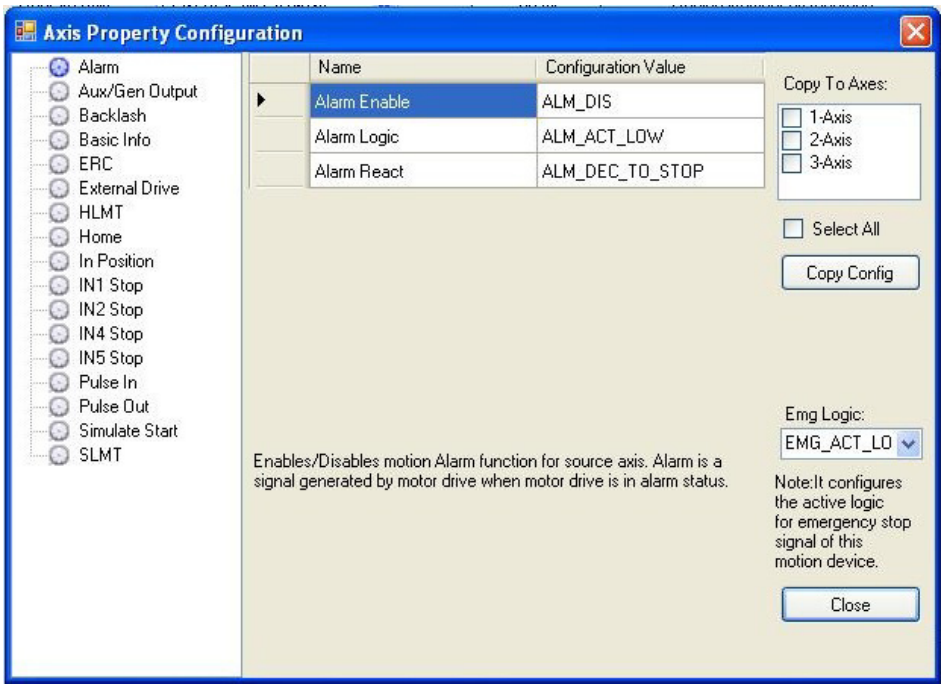


选择 “JOG” 或 “MPG” 并单击 [Set Ext Drive]，外部驱动模式将设置成功，用户即可操作外部驱动。单击 [Close] 将关闭对话框，外部驱动设置为 “Disable”。

注！ *对于 PCI-1245L，只有 0 轴可作为外部驱动的主轴。*

5.3.4.3 Axis Setup

单击该按钮可检查 / 设置轴的属性和 I/O，如下图所示：



左侧的树状图显示轴属性的分类。当用户单击对应项目时，右侧的数据显示将列出分类中的属性和对应属性值。有关详细信息，请参考编程指南中列出的轴特性、配置和参数描述信息。属性分类如下：

分类	名称	简要介绍
Alarm	Alarm Enable	启用 / 禁用源轴的运动报警功能。
	Alarm Logic	设置报警信号的有效逻辑电平。
	Alarm React	设置报警信号的反应模式。
Aux/Gen Output	AuxOut Enable	启用 / 禁用源轴群组的 AddPathDwell() 的轴 Aux-Output。
	AuxOut Time	设置源轴在群组 AddPathDwell 功能中，Aux 输出的启动时间。
	GenDo Enable	启用 / 禁用轴 DO 作为源轴的通用 DO 功能。
Backlash	Backlash Enable	启用 / 禁用源轴的背隙补偿功能。
	Backlash Pulses	设置源轴的补偿脉冲个数。方向发生变化时，发送命令之前，轴输出背隙校正脉冲。
	Backlash Velocity	设置背隙补偿的速度。
Basic Info	PhyID	源轴的物理 ID。
	PPU	该轴的虚拟单位：PPU 个脉冲 / 单位。可以根据实际的马达来设置 PPU，以消除不同马达精度不同的问题。
	ModuleRange	该轴 （旋转轴）的转矩，即旋转一周的 Pulse 数。
ERC	Erc Logic	设置 ERC 信号的有效逻辑电平。
	Erc On Time	设置 ERC 的启动时间。
	Erc Off Time	设置 ERC 的关闭时间。
	Erc Enable Mode	启用 / 禁用源轴的 ERC 输出。

External Drive	Ext Master Src	表示该轴被哪个轴的外来信号控制。
	Ext Sel Enable	当外部驱动使能时，该属性表示通过数字输入通道使能轴驱动选择。
	Ext Pulse Num	手轮模式时，输入脉冲边缘触发时输出的驱动脉冲。
	Ext Preset Num	JOG 模式时，输入脉冲边缘触发时输出的驱动脉冲。
	Ext Pulse In Mode	设置外部驱动的脉冲输入模式。
HLMT	HLMT Enable	启用 / 禁用硬件限位信号。
	HLMT Logic	设置硬件限位信号的有效逻辑电平。
	HLMT React	设置硬件限位信号的反应模式。
Home	Home Ex Mode	设置 HomeEx() 的停止模式。
	Home Cross Distance	设置 Homing 运动中，Search 模式时的每一次 Search 的距离 （详见 Home Mode 中的描述）。
	Home Ex Switch Mode	设置 HomeEx() 的停止条件。
	ORG Logic	设置 ORG 信号的有效逻辑电平。
	EZ Logic	设置 EZ 信号的有效逻辑电平。
	Home Reset Enable	源轴返回原点后，启用 / 禁用复位逻辑计数器。
	ORG React	设置 ORG 信号的反应模式。
In Position	Inp Enable	启用 / 禁用源轴的到位功能。
	Inp Logic	设置到位信号的有效逻辑电平。
Pulse In	Pulse In Mode	设置源轴的编码器反馈脉冲输入模式。
	Pulse In Logic	设置编码器反馈脉冲输入信号的有效逻辑电平。
	Pulse In Source	设置编码器反馈脉冲输入信号的源。
	Pulse In Max Frequency	设置编码器脉冲输入信号的最大频率。
Pulse Out	Pulse Out Mode	设置源轴的命令脉冲输出模式。
Simulate Start	Simulate Start Source	设置同步启停的触发源。
SLMT	SLMT Mel Enable	启用 / 禁用源轴的负方向软件限位。
	SLMT Pel Enable	启用 / 禁用源轴的正方向软件限位。
	SLMTN React	设置负方向软件限位的反应模式。
	SLMTP React	设置正方向软件限位的反应模式。
	SLMTN Value	设置负方向软件限位的值。
	SLMTP Value	设置正方向软件限位的值。
Speed Pattern	Max Velocity	配置源轴的最大速度。
	Max Acc	配置源轴的最大加速度。
	Max Dec	配置源轴的最大减速度。
	Max Jerk	配置源轴的最大加加速度。
	Vel Low	设置源轴的低速度 （起始速度）（单位：PPU/S）。
	Vel High	设置源轴的高速度 （运行速度）（单位：PPU/S）。
	Acc	设置源轴的加速度 （单位：PPU/S2）。
	Dec	设置源轴的减速度 （单位：PPU/S2）。
	Jerk	设置速度曲线类型：源轴的 T 形曲线或 S 形曲线。

注！ *在测试工具中，如果所选设备没有对应功能，则项目不会显示在树状图的左边。比如，如果所选设备为 PCI-1245L，该板卡不支持减速（SD）和振动抑制功能，因此用户在树状图左边无法看到该项。同时，由于单轴对话框支持速度参数设置，因此不会显示速度模式项。*

注！ *选择“Pulse Out”时，“Pulse Out Mode”属性描述内容下面将出现对应模式的图解信息。*

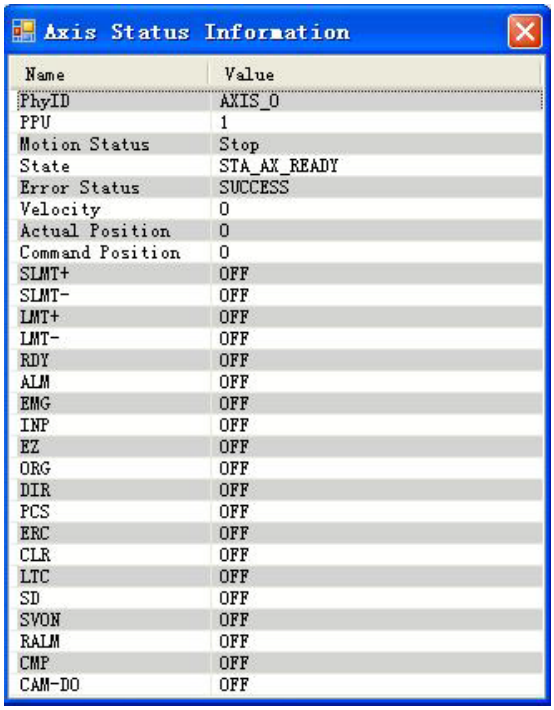
设置项目编辑完成后，属性值将在鼠标离开编辑框时生效（已经在设备中设置）。

如果用户想要将属性设置复制给其它轴，只需勾选右侧复选框中相应的轴，然后单击[Copy Config]。

单击[Close]关闭窗口。

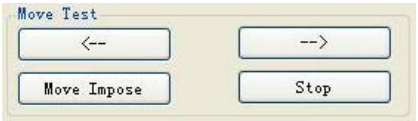
5.3.4.4 Axis Status

单击按钮查看指定的轴信息。比如，PhyID、PPU、基本状态（Motion Status、State 和 Error Status 等）以及 I/O 状态（Alarm 和 SLMTP/N 等）。



5.3.5 Move Test

操作如下：

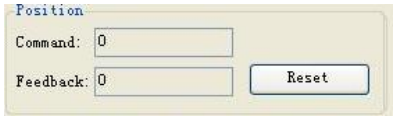


选择运动模式后，单击[<--]或[-->]，轴将进行正向 / 反向的点对点 / 连续 / 返回原点运动。

运动速度到达点对点运动的 VelHigh 后，用户可单击 [Move Impose] 生成一个叠加运动。叠加运动的距离等于 New Pos 的值，叠加运动的速度等于 New Vel 的值。通过单击 [Speed Chart]，用户能够观察具体的运动 / 速度曲线。

单击 [Stop] 停止运动。

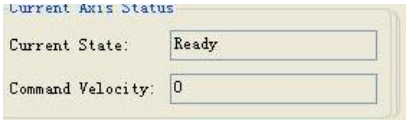
5.3.6 Position



通过 “Position” 状态，用户能够在操作时观察理论位置和反馈位置。

单击 [Reset] 可将值复位为 “0”。

5.3.7 Current Axis Status



用户可查看当前状态和理论速度。有关详细信息，请参考通用 API 编程指南中的 Acm_AxGetState 函数的说明信息。

5.3.8 DI/O Status

显示所选轴的4个DI端口和4个DO端口的当前状态。 还可以将DO设置为 “ON/OFF”。

5.3.8.1 DI

如上图所示，DI（3-0）状态从右到左依次为 DI0 到 DI3。其中， 表示 DI 有效（On）且值为 1， 表示 DI 无效 （Off）且值为 0。

5.3.8.2 DO

如上图所示，DO（7-4）状态从右到左依次为 DO4 到 DO7。其中， 表示 DO 有效（On）且值为 1， 表示 DO 无效 （Off）且值为 0。

5.3.9 Last Error Status

用户可以查看最新错误代码和错误信息。如果没有错误，错误代码为 “0”，错误信息为 “SUCCESS”。

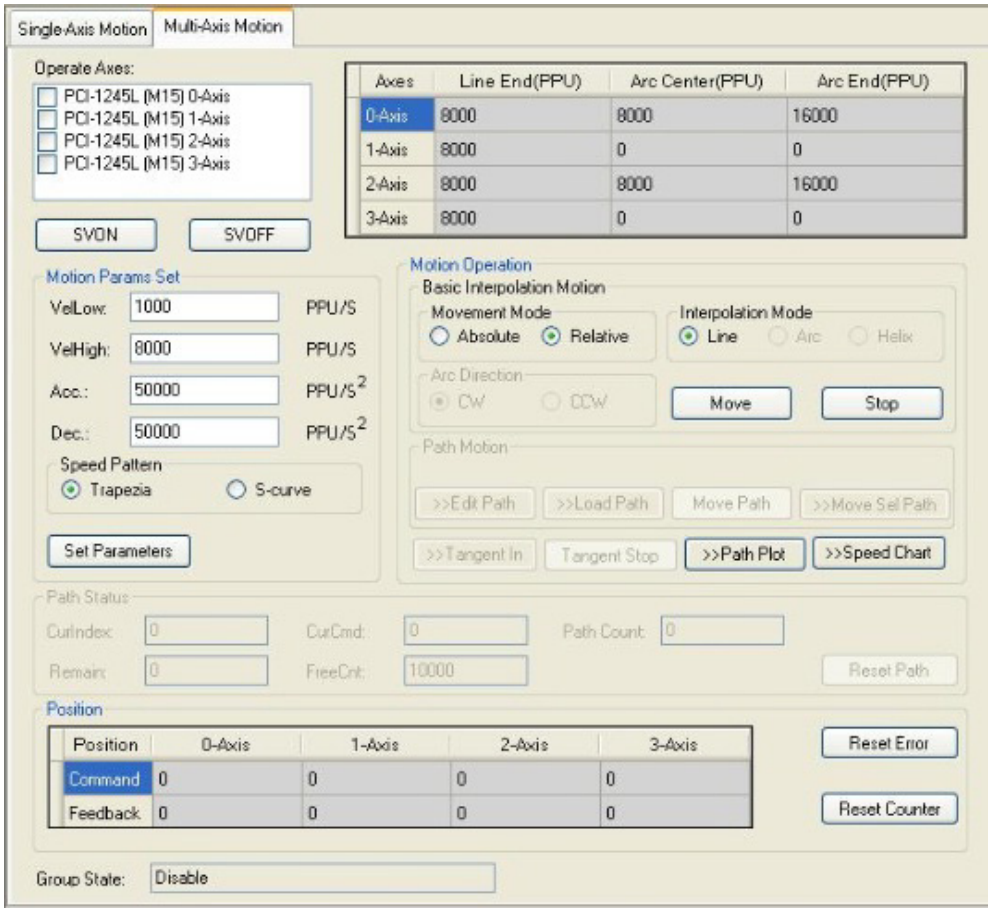
5.3.10I/O Status

用户能够从 LED 栏中了解 I/O 状态。其中， 表示设备不支持该功能或者没有对应 I/O； 表示设备支持该功能，但是没有触发 （OFF）； 表示对应 I/O 触发 （ON）。

有关详细信息，请参考通用 API 编程指南中的 Acm_AxGetMotionIO 函数的状态信息。

若不支持相应的功能或没有对应的 I/O，相应的文字也灰阶掉；若支持相应功能，但此功能没有启用 （对应的 Enable 属性设置为 Disable），相应的文字也灰阶；若支持相应的功能，且此功能已启用 （对应的 Enable 属性设置为 Enable，若此功能有相应的 Enable 属性），则相应的文字正常显示。

5.4 Multi-Axis Motion



5.4.1 Operate Axes

窗口中的列表框列出来所选设备的全部轴，勾选对应轴，将其添加到群组中。如果添加到群组的轴的数量小于 2，则群组状态将为 “Disable”。如果添加到群组的轴的数量大于或等于 2，群组状态将为 “Ready”，用户配置好适当参数后，即可进行适当插补操作。

5.4.2 Motion Params Set

运动参数设置。可对 Group 进行初速度 （VelLow）、运行速度 （VelHigh）、加速度 （Acc.）、减速度 （Dec.）及速度类型 （Speed Pattern）的设置。

5.4.3 Motion Ends

请按照下图配置运动的圆心 / 端点。

Axes	Line End(PPU)	Arc Center(PPU)	Arc End(PPU)
0-Axis	8000	8000	16000
1-Axis	8000	0	0
2-Axis	8000	8000	16000
3-Axis	8000	0	0

对话框将参考群组轴和插补模式自动将编辑框变为可写。如上图所示，1-axis 和 2-axis 添加到群组中且选择了线性插补模式，因此可写的编辑框为 “Line End (PPU)” 栏的 “1-axis” 和 “2-axis” 行，背景色为白色的区域。背景色为灰色的编辑框表示不可编辑。

5.4.4 Motion Operation

5.4.4.1 SVON

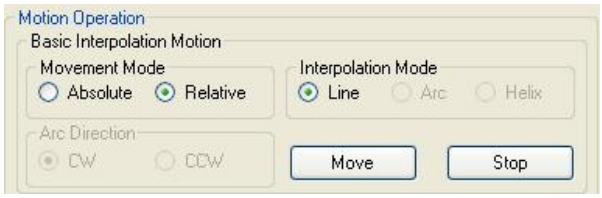
点击 [SVON]，会将 Group 中所有轴的伺服开启。

5.4.4.2 SVOFF

单击 [SVOFF]，群组中轴的伺服将关闭。

5.4.4.3 Basic Interpolation Motion

基本插补运动包括直线插补 （Line），如下图所示。



5.4.4.3.1 Movement Mode

Absolute：插补运动将直接使用设置的位置参数。

Relative：插补运动将添加初始偏移到位置参数，然后使用。

5.4.4.3.2 Interpolation Mode

Line：直线插补。

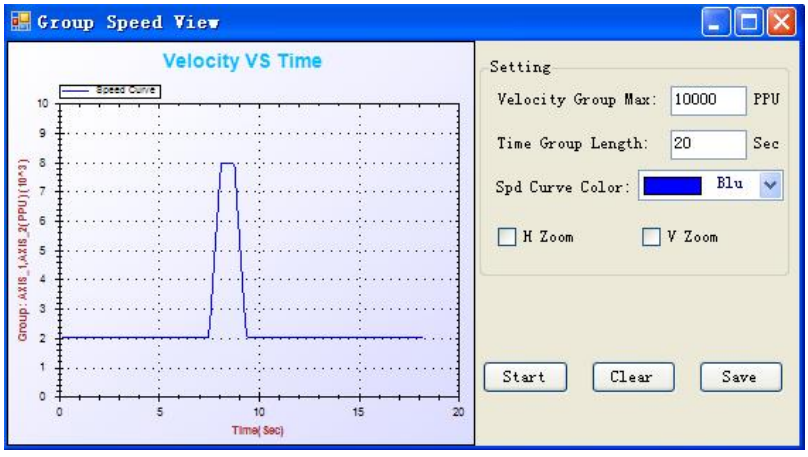
5.4.4.3.3 Move

相应配置完成后，单击 [Move]，群组将执行指定插补运动。

5.4.4.3.4 Stop

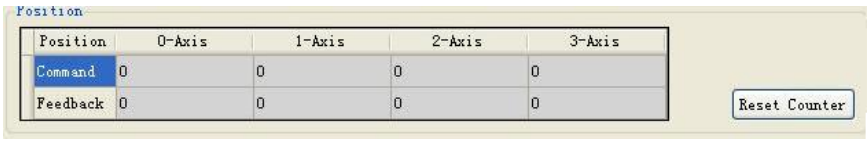
当群组正在进行插补运动，单击 [Stop]，插补运动将停止。

5.4.4.4 Speed Chart



设置和操作与 “Single Axis Motion” 中的 [Speed Chart] 类似。

5.4.5 Position



显示设备中所有轴的当前命令和反馈位置。

单击 [Reset Counter] 将计数复位至 0。

5.4.6 State & Status

Group State：显示当前群组的状态。有关详细信息，请参考通用 API 编程指南中的 Acm_GpGetState 函数的说明信息。

Last Error Status：显示上一个错误信息。

Axis Name：发生错误的轴。

Error Code：错误代码。

Error Message：具体的错误信息。

第 6 章

编程指南

本章将详细介绍每个功能的 API 编程。

6.1 简介

本章为用户提供了 API 及其定义，并向用户展示如何使用 API。

PCI-1245L 设备驱动基于通用运动架构。有关通用运动架构的详细信息，请参考第 4.3 节内容。根据该架构，所有功能和属性可分为以下三类：**设备对象、轴对象（单一轴）和群组对象（多个轴）**。使用 API 功能和属性之前，需要了解以下几个基本概念。

- API 及属性的命名：通用运动架构下的所有 API 和属性均遵循统一的命名规则。详情请参考第 4.3.3 节内容。
- 数据类型重定义：为了简化代码，对共同数据类型进行重新定义。
- 错误代码：所有 API 都将返回代码显示调用成功 / 失败 （错误原因）。

6.1.1 数据类型再定义

数据类型再定义和 Windows 共同数据类型表如下所示：

新类型	Windows 数据类型	备注
U8	UCHAR	8 bit 无符号整数
U16	USHORT	16 bit 无符号整数
U32	ULONG	32 bit 无符号整数
U64	ULONGLONG	64 bit 无符号整数
I8	CHAR	8 bit 带符号整数
I16	SHORT	16 bit 带符号整数
I32	INT	32 bit 带符号整数
I64	LONGLONG	64 bit 带符号整数
F32	FLOAT	32 bit 浮点变量
F64	DOUBLE	64 bit 浮点变量
PU8	UCHAR *	指针指向 8 bit 无符号整数
PU16	USHORT *	指针指向 16 bit 无符号整数
PU32	ULONG *	指针指向 32 bit 无符号整数
PU64	ULONGLONG *	指针指向 64 bit 无符号整数
PI8	CHAR *	指针指向 8 bit 带符号整数
PI16	SHORT *	指针指向 16 bit 带符号整数
PI32	INT*	指针指向 32 bit 带符号整数
PI64	LONGLONG *	指针指向 64 bit 带符号整数
PF32	FLOAT *	指针指向 32 bit 浮点变量
PF64	DOUBLE *	指针指向 64 bit 浮点变量

首个字符 F/I/U 表示数据类型，数字表示数据长度。

6.1.2 关于错误代码

调用通用运动架构中的 API 时，每个 API 都将获得一个返回代码。返回代码表示调用结果。 关于错误代码的详细信息，请参考附录。用户可根据 Acm_GetErrorMessage 查看返回的错误代码对应的错误信息。根据错误信息，用户能够进行适当修改。

6.1.3 关于事件

事件是对象之间发送和处理消息的过程。用户可启用 / 禁用事件。事件使能后，一旦条件满足，驱动中就会触发事件，用户就会收到通知。禁用事件时，即使事件在驱动内触发，用户也不会得到通知。

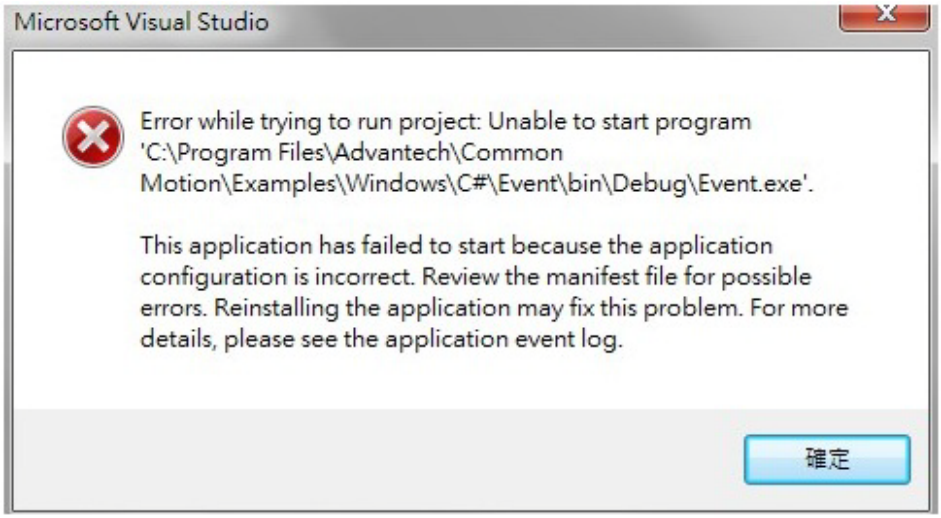
有七种事件类型：

事件名称	说明
EVT_AX_MOTION_DONE	当前运动完成时，触发事件。
EVT_AX_VH_START	当运动速度达到 High Speed 时，触发此事件。
EVT_AX_VH_END	当开始减速时，触发此事件。
EVT_GPn_MOTION_DONE	当群组运动完成时，触发事件。n 代表 group_id。（可通过 Acm_DevGetProperty 获取属性 PAR_GpGroupID 的属性值。）
EVT_GPn_VH_START	当群组运动速度达到 high speed 时，触发此事件。n 代表 group_id。
EVT_GPn_VH_END	当群组运动开始进入减速段时，触发此此事件。n 代表 group_id。

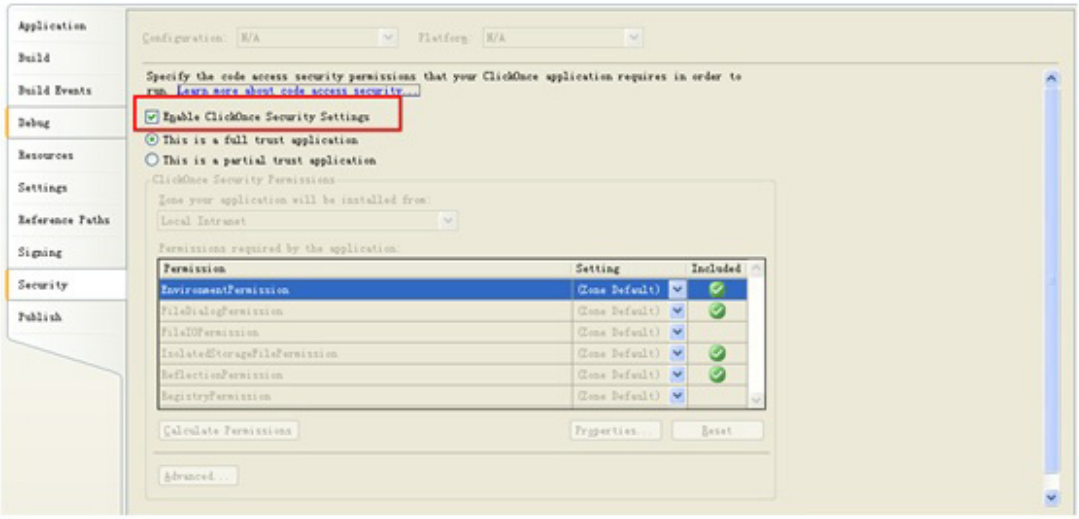
请参考 Acm_EnableMotionEvent 和 Acm_CheckMotionEvent。

6.1.4 关于在 Win7 下使用 Common Motion API 的注意事项

1. 由于 Acm_GetAvailableDevs 要获取应用程序所运行的电脑上所有可用板卡的信息，需要读取注册表中相关信息，而在 Win7 系统下，此操作需要有管理员的权限方可正常执行。因此，如果应用程序要调用此函数，请添加相应的 Manifest 文件，并将应用程序的权限提升为管理员权限 （详请见附件中的 “ 关于提升应用程序的权限 ”）。
2. 在使用 VS2008 或者 VS2010 打开 C#/VB.net 范例，若执行的时候提示如下错误：



请将 Project properties 对话框中的 Security 栏中的 “Enable ClickOnce Security Setting” 的打钩去掉，重新编译即可正常运行。



6.1.5 关于提升应用程序的权限

1. 若是使用 Microsoft Visual Studio 2005(VS2005) 进行开发，可从 C#/VB.net 范例中 Properties 文件夹下拷贝 Manifest 文件：app.manifest 到本工程的 Projecties 文件夹下，通过 Project->Add Existing Item 选项添加到工程中即可。
2. 若使用 Microsoft Visual C++ 6.0 进行开发，可从 VC 范例中拷贝拷贝 Manifest 文件：App.manifest 到本工程路径下，在资源中导入这个文件，资源类型 24，资源 ID 为 1 即可。
3. 若使用 Microsoft Visual Studio 2008/2010 进行开发，

   方法一：可以同上面 VS2005 一样，从范例中拷贝 app.manifest 文件加载到所开发的工程中；

   方法二：直接更改工程权限管理的设置即可：在工程属性 -->Configuration Properties-->Linker-->Manifest File-->UAC Execution Level-->requireAdministrator。

   方法三：勾选 Project properties 对话框中的 Security 栏中的 “Enable ClickOnce Security Setting” 选项，在 Properties 下将自动生成 Manifest 文件，打开 Manifest 文件，将如下截图中的红框行改为 “<requestedExecutionLevel level=″requireAdministrator″ uiAccess=″false″ />”；然后再将 Project properties 对话框中的 Security 栏中的 “Enable ClickOnce Security Setting” 选项的勾去掉即可。

```
<requestedPrivileges xmlns="urn:schemas-microsoft-com:asm.v3">
  <!-- UAC Manifest Options
       If you want to change the Windows User Account Control level replace the
       requestedExecutionLevel node with one of the following.

  <requestedExecutionLevel  level="asInvoker" uiAccess="false" />
  <requestedExecutionLevel  level="requireAdministrator" uiAccess="false" />
  <requestedExecutionLevel  level="highestAvailable" uiAccess="false" />

       Specifying requestedExecutionLevel node will disable file and registry virtualization.
       If you want to utilize File and Registry Virtualization for backward
       compatibility then delete the requestedExecutionLevel node.
  -->
  <requestedExecutionLevel level="asInvoker" uiAccess="false" />
</requestedPrivileges>
```

6.2 快速入门

6.2.1 PCI-1245L 的软件架构

PCI-1245L 软件架构基于通用运动架构，如下图所示：

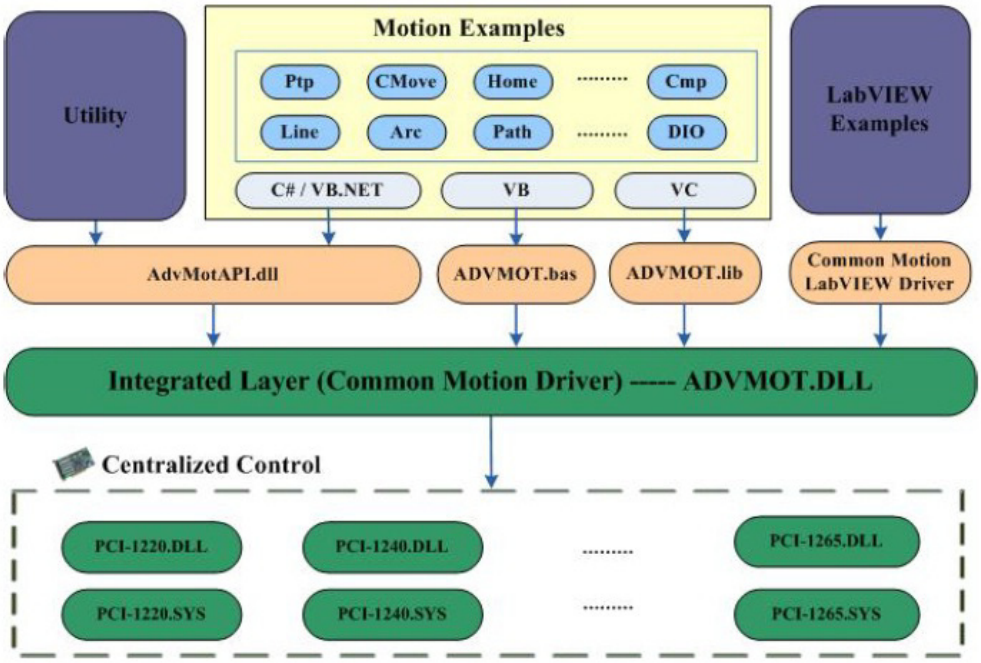


图 6.1： PCI-1245L 的软件架构

用于实现设备功能的所有 API 都可从 **ADVMOT.DLL** （为用户提供的一个通用接口）获取。AdvMotAPI.dll、ADVMOT.bas 和 ADVMOT.lib 都是基于 ADVMOT.dll 产生，方便用户轻松开发应用程序。AdvMotAPI.dll用于C#应用程序和VB.net应用程序，包括Utility、C# 示例和 VB.net 示例。ADVMOT.bas 用于开发 VB 应用程序。ADVMOT.lib 用于开发 VC 应用程序。

6.2.2 流程图

6.2.2.1 基本流程

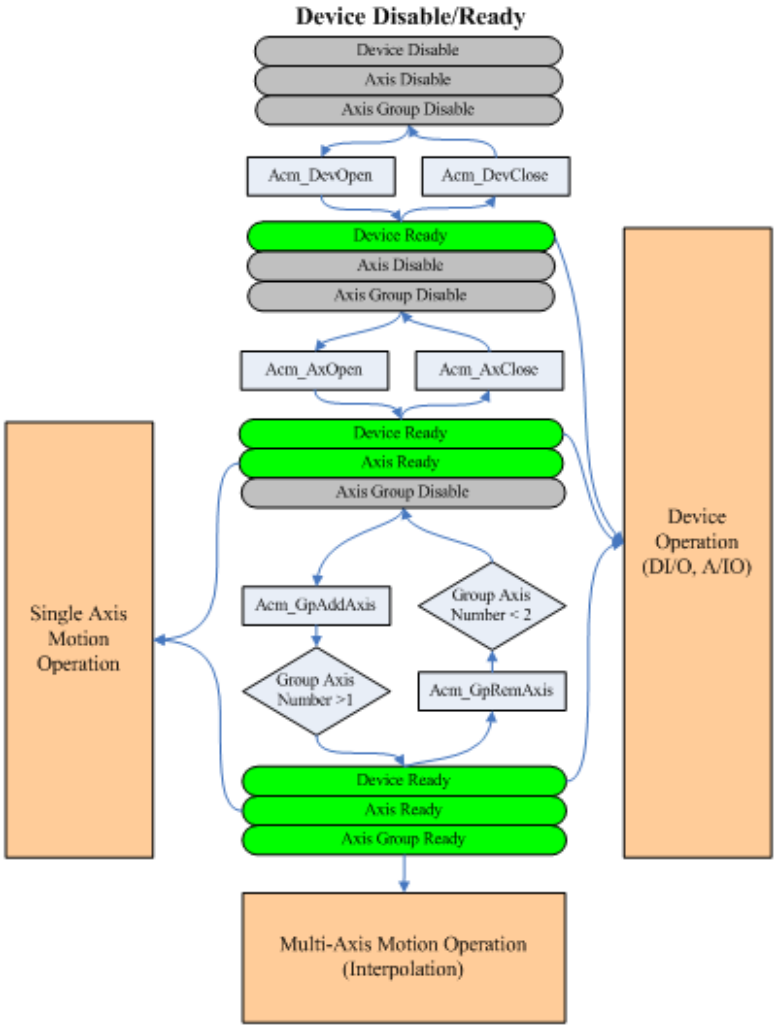


图 6.2： 基本操作流程图

6.2.2.2 单轴流程图

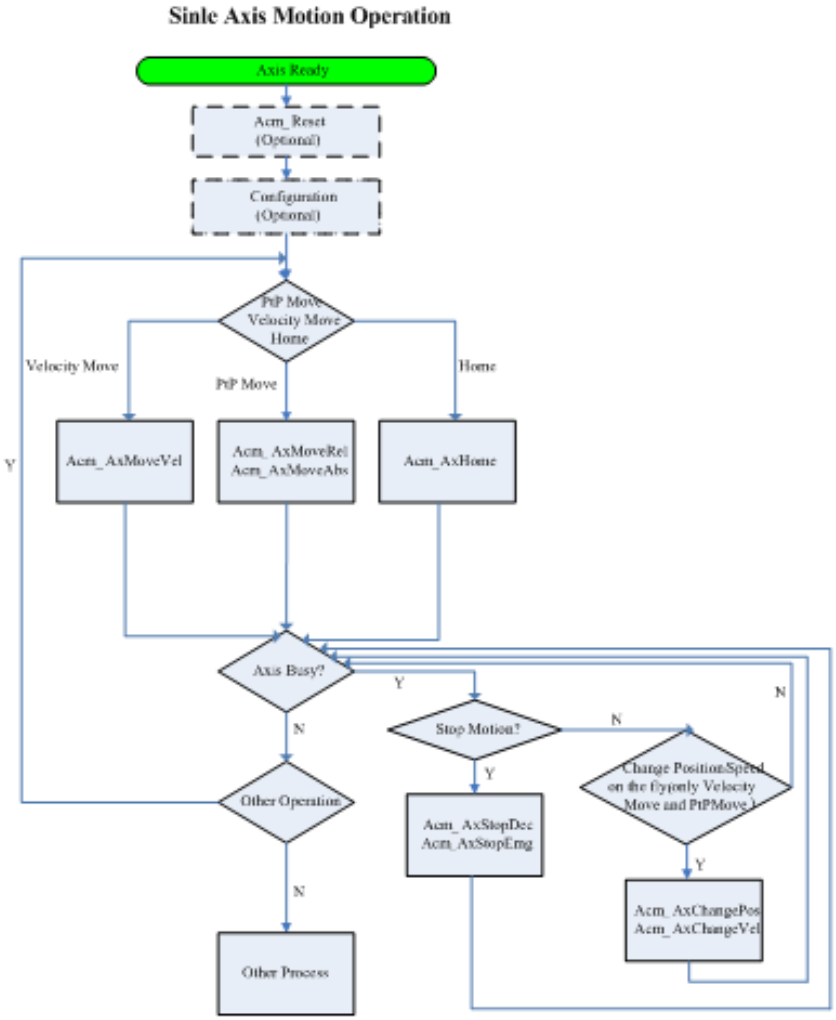


图 6.3： 单轴操作流程图

6.2.2.3 多轴流程图

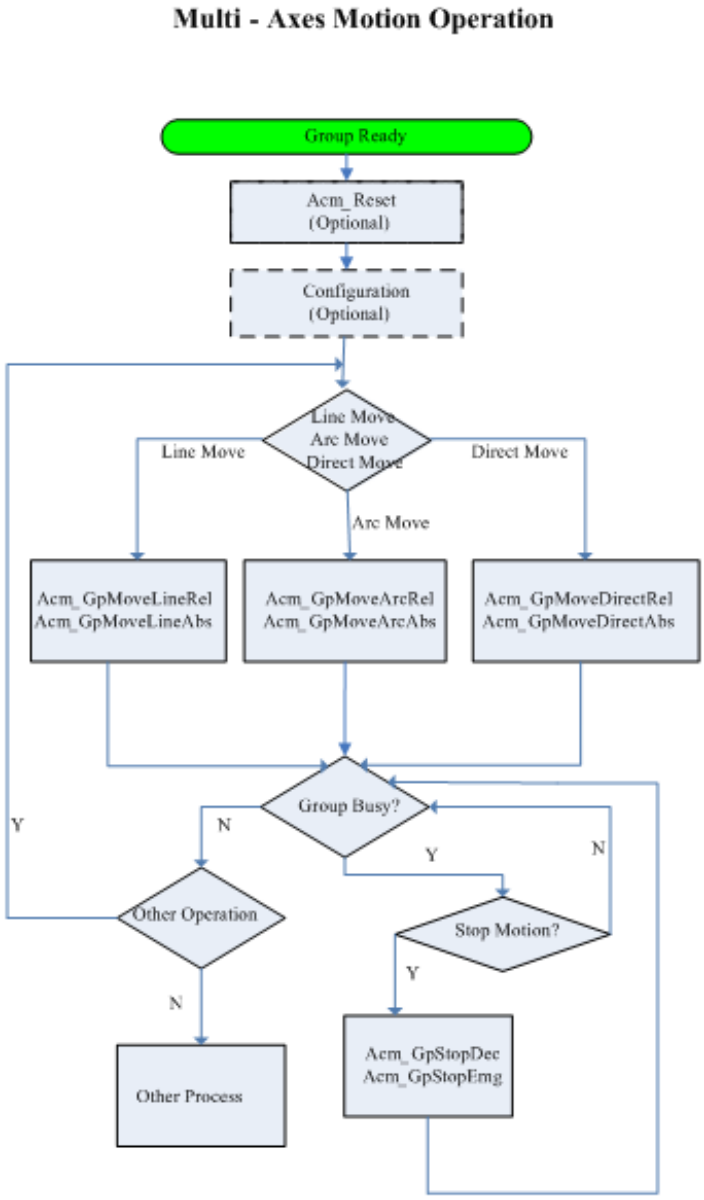


图 6.4： 多轴操作流程图

6.2.3 示例支持列表

示例	VC	C#	VB	VB .NET	说明
Change_P	√	√		√	展示如何改变正在进行的 1 轴运动位置。
Change_V	√	√		√	展示如何改变正在进行的 1 轴运动速度。
Cmove	√	√		√	展示如何使用 ACM API 控制单轴连续运动。
DIO	√	√		√	展示轴的数字量输入 / 输出功能。
Event	√	√		√	展示如何使用事件。
Home	√	√		√	展示如何使用返回原点功能。
Line	√	√		√	展示如何控制一个插补群组的线性运动。
MPG_JOG	√	√		√	展示如何在指定设备和轴上使用外部驱动功能
PTP	√	√	√	√	展示如何控制单轴的点到点运动。
SimulateOpe	√	√			展示如何控制多个轴之间的同步起停功能。

Direct	√	√			展示如何控制一个插补群组的直线插补。
DataProtect	√	√	√	√	读 / 写私有资料。

6.2.4 PCI-1245L 支持的 API 列表

类型		方法 / 事件	PCI-1245L	说明
设备	方法	Acm_DevOpen	√	打开设备。
		Acm_DevClose	√	关闭设备。
		Acm_DevLoadConfig	√	加载配置文件。
		Acm_GetProperty	√	获取属性。
		Acm_SetProperty	√	设置属性。
		Acm_GetLastError	√	获取最后一次错误。
		Acm_CheckMotionEvent	√	检查事件是否发生。
		Acm_EnableMotionEvent	√	启用 / 禁用事件。
	事件	EVT_AX_MOTION_DONE	√	当前运动完成时，触发事件。
		EVT_AX_ERROR	√	当错误发生时，触发此事件。
		EVT_AX_VH_START	√	当运动速度达到 High Speed 时，触发此事件。
		EVT_AX_VH_END	√	当开始减速时，触发此事件。
		EVT_GPn_MOTION_DONE	√	当群组运动完成时，触发事件。
		EVT_GPn_VH_START	√	当群组运动速度达到 high speed 时，触发此事件。
		EVT_GPn_VH_END	√	当群组运动开始进入减速段时，触发此事件。
	私有资料读写	Acm_DevReadEEPROM_Ex	√	读取 EEPROM 私有资料。
		Acm_DevWriteEEPROM_Ex	√	写入 EEPROM 私有资料。

轴	系统	Acm_AxOpen	√	打开轴。
		Acm_AxClose	√	关闭轴。
		Acm_AxResetError	√	当轴处于发生错误停止时，复位错误。
	运动 I/O	Acm_AxSetSvOn	√	打开伺服驱动器。
		Acm_AxGetMotionIO	√	获取运动 IO 的状态。
	运动状态	Acm_AxGetMotionStatus	√	获取当前运动状态。
		Acm_AxGetState	√	获取轴的状态。
	停止	Acm_AxStopDec	√	减速停止。
		Acm_AxStopEmg	√	紧急停止。
		Acm_AxStopDecEx	√	下达停止命令时可指定减速度。
	速度运动	Acm_AxMoveVel	√	命令连续运动。
		Acm_AxChangeVel	√	当前运动的理论速度发生变化。
		Acm_AxChangeVelByRate	√	按照设定的比例改变当前正在执行的动作的运行速度。
		Acm_AxChangeVelEx	√	在运动的过程中可同时改变速度，加速度和减速度。
		Acm_AxChangeVelExByRate	√	在运动的过程中可根据比率改变运行速度，同时可改变加速度和减速度。
		Acm_AxGetCmdVelocity	√	获取当前理论速度。
	点到点运动	Acm_AxMoveRel	√	命令相对点到点运动。
		Acm_AxMoveAbs	√	命令绝对点到点运动。
		Acm_AxChangePos	√	改变点到点运动的终点位置。
	同时运动	Acm_AxSimStartSuspendAbs	√	暂停同步的绝对模式下的点到点运动。
		Acm_AxSimStartSuspendRel	√	暂停同步的相对点到点运动。
		Acm_AxSimStartSuspendVel	√	暂停同步的连续运动。
		Acm_AxSimStart	√	开始同步运动。
		Acm_AxSimStop	√	停止同步运动。
	返回原点	Acm_AxHome	√	命令回原点运动。
	位置/计数器	Acm_AxSetCmdPosition	√	设置理论位置。
		Acm_AxGetCmdPosition	√	获取理论位置。
		Acm_AxSetActualPosition	√	设置实际位置。
		Acm_AxGetActualPosition	√	获取实际位置。
	Aux/Gen 输出	Acm_AxDoSetBit	√	设置以位形式的 DO 值。
		Acm_AxDoGetBit	√	获取以位形式的 DO 值。
		Acm_AxDiGetBit	√	获取以位形式的 DI 值。
	外部驱动	Acm_AxSetExtDrive	√	设置外部驱动。

群组	系统	Acm_GpAddAxis	√	向群组中添加轴。
		Acm_GpRemAxis	√	从群组中移除轴。
		Acm_GpClose	√	关闭群组。
		Acm_GpResetError	√	当群组处于发生错误停止时，复位错误。
	运动状态	Acm_GpGetState	√	获取群组的当前状态。
	速度	Acm_GpGetCmdVel	√	获取群组的当前的速度值。
	运动停止	Acm_GpStopDec	√	减速停止。
		Acm_GpStopEmg	√	紧急停止。
	插补运动	Acm_GpMoveLinearRel	√	命令相对线性插补。
		Acm_GpMoveLinearAbs	√	命令绝对线性插补。
		Acm_GpMoveDirectAbs	√	命令绝对直接线性插补。
		Acm_GpMoveDirectRel	√	命令相对直接线性插补。

6.2.5 属性支持列表

类型		属性	PCI-1245L
设备	特性	FT_DevIpoTypeMap	√
		FT_DevAxesCount	√
		FT_DevFunctionMap	√
		FT_DevOverflowCntr	√
	配置	CFG_DevBoardID	√
		CFG_DevBaseAddress	√
		CFG_DevInterrupt	√
		CFG_DevBusNumber	√
		CFG_DevSlotNumber	√
		CFG_DevDriverVersion	√
		CFG_DevDllVersion	√
		CFG_DevFwVersion	√
		CFG_DevFPGA_1Version	√
		CFG_DevEmgLogic	√

轴	系统	FT_AxFunctionMap	√
		CFG_AxPPU	√
		CFG_AxPhyID	√
	速度模式	FT_AxMaxVel	√
		FT_AxMaxAcc	√
		FT_AxMaxDec	√
		FT_AxMaxJerk	√
		CFG_AxMaxVel	√
		CFG_AxMaxAcc	√
		CFG_AxMaxDec	√
		CFG_AxMaxJerk	√
		PAR_AxVelLow	√
		PAR_AxVelHigh	√
		PAR_AxAcc	√
		PAR_AxDec	√
		PAR_AxJerk	√
	脉冲输入	FT_AxPulseInMap	√
		FT_AxPulseInModeMap	√
		CFG_AxPulseInMode	√
		CFG_AxPulseInLogic	√
		CFG_AxPulseInMaxFreq	√
	脉冲输出	FT_AxPulseOutMap	√
		FT_AxPulseOutModeMap	√
		CFG_AxPulseOutMode	√
	报警	FT_AxAlmMap	√
		CFG_AxAlmLogic	√
		CFG_AxAlmEnable	√
		CFG_AxAlmReact	√
	到位	FT_AxInpMap	√
		CFG_AxInpEnable	√
		CFG_AxInpLogic	√
	ERC	FT_AxErcMap	√
		FT_AxErcEnableModeMap	√
		CFG_AxErcLogic	√
		CFG_AxErcEnableMode	√
	SD	FT_AxSdMap	√
	硬件限位	FT_AxElMap	√
		CFG_AxElReact	√
		CFG_AxElLogic	√
		CFG_AxElEnable	√
	软件限位	FT_AxSwMelMap	√
		FT_AxSwPelMap	√
		CFG_AxSwMelEnable	√
		CFG_AxSwPelEnable	√
		CFG_AxSwMelReact	√
		CFG_AxSwPelReact	√
		CFG_AxSwMelValue	√
		CFG_AxSwPelValue	√

轴	返回原点	FT_AxHomeMap	√
		CFG_AxOrgLogic	√
		CFG_AxOrgReact	√
		CFG_AxEzLogic	√
		CFG_AxHomeResetEnable	√
		PAR_AxHomeCrossDistance	√
		PAR_AxHomeExSwitchMode	√
	背隙	FT_AxBacklashMap	√
		CFG_AxBacklashEnable	√
		CFG_AxBacklashPulses	√
		CFG_AxBacklashVel	√
	Aux/Gen 输出	FT_AxGenDOMap	√
		FT_AxGenDIMap	√
		CFG_AxGenDoEnable	√
	外部驱动	FT_AxExtDriveMap	√
		FT_AxExtMasterSrcMap	√
		CFG_AxExtMasterSrc	√
		CFG_AxExtPulseNum	√
		CFG_AxExtPulseInMode	√
		CFG_AxExtPresetNum	√
	同步启停	FT_AxSimStartSourceMap	√
		CFG_AxSimStartSource	√
	DI Stop	FT_AxIN1Map	√
		FT_AxIN2Map	√
		FT_AxIN4Map	√
		FT_AxIN5Map	√
		CFG_AxIN1StopEnable	√
		CFG_AxIN1StopReact	√
		CFG_AxIN1StopLogic	√
		CFG_AxIN2StopEnable	√
		CFG_AxIN2StopReact	√
		CFG_AxIN2StopLogic	√
		CFG_AxIN4StopEnable	√
		CFG_AxIN4StopReact	√
		CFG_AxIN4StopLogic	√
		CFG_AxIN5StopEnable	√
		CFG_AxIN5StopReact	√
		CFG_AxIN5StopLogic	√
群组	系统	PAR_GpGroupID	√
		CFG_GpAxesInGroup	√
	速度模式	PAR_GpVelLow	√
		PAR_GpVelHigh	√
		PAR_GpAcc	√
		PAR_GpDec	√
		PAR_GpJerk	√

6.2.6 创建一个新的应用

在 PCI-1245L 下创建一个新的应用，用户需要安装范例安装包。“Advantech\Motion Common\Examples”文件夹中有很多用不同语言开发的示例，用户可按照这些示例开发一个新的应用。

安装范例包之后，用户可在 “\Advantech\Motion Common” 文件夹中发现两个文件夹：“Include” 和 “Public”。“Public” 文件夹中的文件可供用户通过不同语言创建应用。文件和开发语言的关系如图 6.1 所示。

6.2.6.1 创建一个新的 VC 控制台应用

创建一个新的控制台应用的步骤如下：

1. 从主菜单中单击 **“File/New”** 创建 Visual C++ 程序需要的应用工程和源代码。

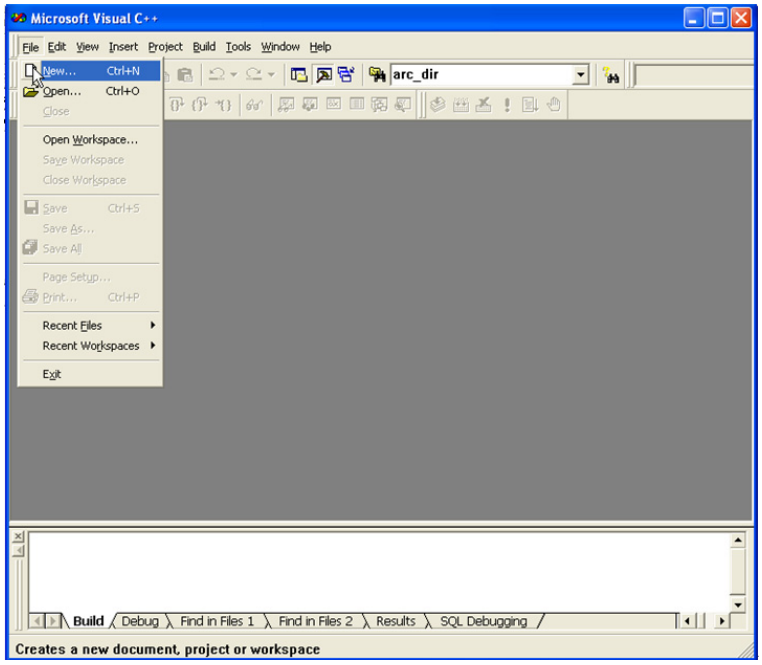


图 6.5： 打开文件创建一个新的 VC 应用

2. 将新工程的类型设置为 **“Win32 Console Application”**，将平台设置为 **“Win32”**，然后指定一个工程文目录。

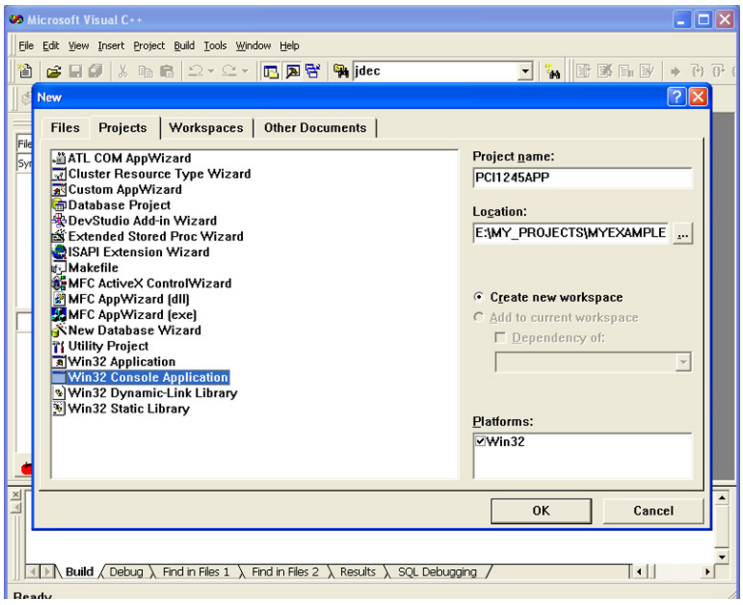


图 6.6： 创建一个新的 VC 控制台应用

单击 “OK”，用户可以选择创建一种控制台应用。 这样，就创建了一个新的控制台应用。

3. 配置新工程。 用户应添加头文件和必要 Lib 文件的路径，并在 “Project Setting” 中配置工程。
用户可通过 “Menu - Poject - Settings” 打开 “Project Setting” 窗口；或右击新的工程，然后选择 “Setting” 打开窗口。配置如下：
a. 在通用运动架构中，“Calling convention” 应为 “_stdcall”，因此用户需按照如下所示配置 “Calling convention”：

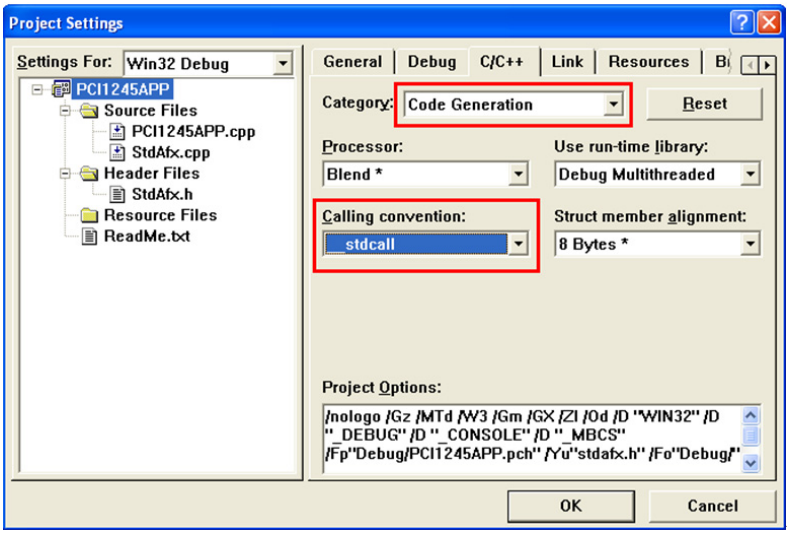


图 6.7： 设置 “Calling convention”

b. 设置头文件路径，如下路径包括用户可能用到的所有头文件。 请注意路径需要设置正确。 比如，包含该工程的文件内容如下所示：

图 6.8： 该示例的文件内容

因此，路径设置如下所示。

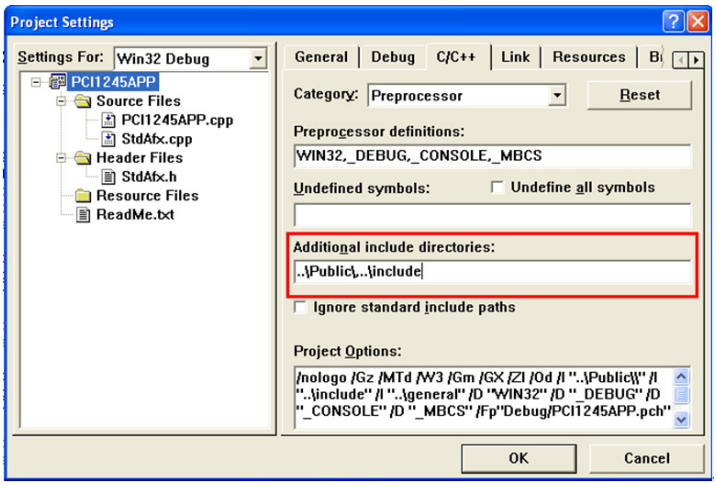


图 6.9： 添加头文件路径

c. 设置必要的 Lib 文件。

名为 “ADVMOT.lib” 的 Lib 文件，对应 “systemroot\ system32\” 目录下的 “ADVMOT.dll” 文件，可帮助用户轻松开发应用。 安装示例包之后，Lib 文件位于 “Public” 文件夹中。

用户应注意头文件的路径。

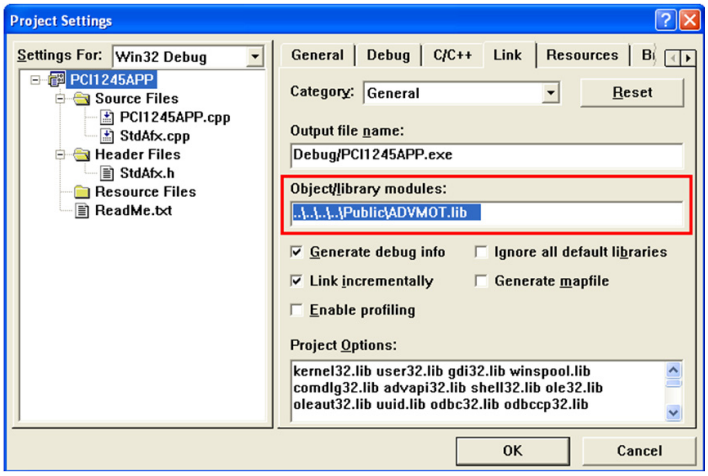


图 6.10： 设置 Lib 文件路径

工程设置完成之后，用户可成功建立该工程。

4. 编写代码。

```
#include "stdafx.h"
#include <wtypes.h>
#include <stdio.h>
#include "AdvMotApi.h"

#define  MAX_CNT 100

int main(int argc, char* argv[])
```

```
{
   ULONG errcde;
   HAND devHandle;
   HAND axHandle[MAX_CNT];
   ULONG devNum , devCnt,buffLen, axisCntPerDev;
   USHORT i;
   DEVLIST devList[MAX_CNT];
   //Step1.  Get  available  devices  by  calling  API
  "Acm_GetAvailableDevs"
   errcde = Acm_GetAvailableDevs(devList, MAX_CNT, &devCnt);
   if (errcde!=0)
   {
        printf("Can not find available device! \n");
        getchar();
        return 0;
   }
     printf("Get available devices successfully! \n");
   //Step2. Open device.
     devNum = devList[0].dwDeviceNum;
   errcde = Acm_DevOpen(devNum, &devHandle);
   if (errcde!=0)
   {
          printf("Open device is failed! \n");
          getchar();
          return 0;
   }
      printf("Open device successfully! \n");
//Step3. After open device successfully, user can get necessary
  property.
      buffLen=sizeof(axisCntPerDev);
errcde     =     Acm_GetProperty     (devHandle,FT_DevAxesCount,
  axisCntPerDev, &buffLen );
if (errcde!=SUCCESS)
{
            Acm_DevClose(&devHandle);
            printf("Get property is failed! \n");
            getchar();
            return 0;
    }
     printf("Get property successfully! \n");
   //Step2. Open the axes.
     for (i=0; i<axisCntPerDev; i++)
```

```
    {
                errcde = Acm_AxOpen(devHandle, i, &axHandle[i]);
                if (errcde!=0)
                {
                    printf("Open axis_0 is failed! \n");
                    getchar();
                    return 0;
                 }
          }
    printf("Open axes successfully! \n");
       //Stp3. Move relative Axis 0 Point to Point motion.
       errcde = Acm_AxMoveRel(axHandle[0], 10000);
       if (errcde!=0)
       {
                   printf("move axis_0 is failed! \n");
                   getchar();
                   return 0;
       }
         printf("Command axis 0 to move point to point successfully!
       \n");
       // Step 4. At last, Close axis and device before application
       exit.
          for (i=0; i<axisCntPerDev; i++)
          {
            errcde = Acm_AxClose(&axHandle[i]);
            if (errcde!=0)
            {
                printf("Open axis_0 is failed! \n");
                getchar();
                return 0;
            }
          }
        Acm_DevClose(&devHandle);
         getchar();
       return 0;
}
```

5. 执行结果如下。

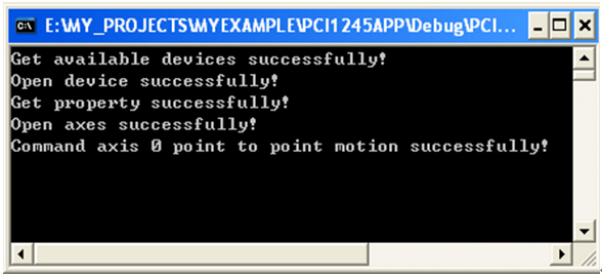


图 6.11： VC 控制台示例的结果

6.2.6.2 创建一个新的 Visual Basic 应用

创建一个新的控制台应用的步骤如下：

1. 打开 Visual Basic 6.0 开发程序，将出现如下窗口：

图 6.12： 加载 VB 开发环境

2. 选择 “Standard EXE” 图标，然后按 “Open” 按钮。这样，就创建了一个新工程。
3. 将模块添加到工程中。从 “View” 菜单中单击 “Project Explorer”。通过单击 “Project” 窗口菜单中的 “Add Module”，添加 ADVMOT.bas（安装示例包后，位于 “Advantech\Motion Common\Public” 路径下）模块和 general.bas（安装示例包后，位于 “\Advantech\Motion Common\Examples” 路径下）模块。

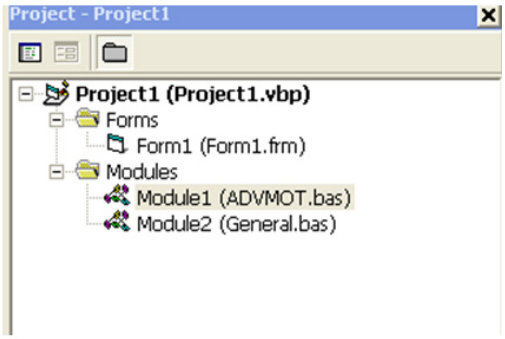


图 6.13： 将模块文件添加到工程中

4. 设计表框。

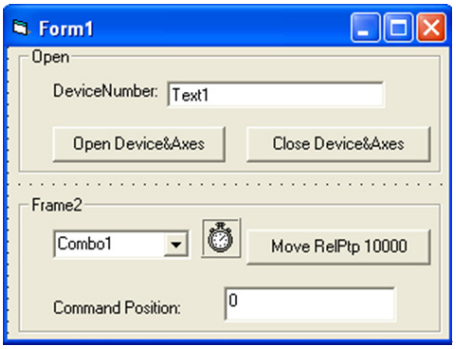


图 6.14： 设计表框。

5. 编写代码。
变量定义如下：

```
Option Explicit
Dim m_DevHand As Long
Dim m_dwDevNum As Long
Dim AxisPerDev As Long
Dim m_AxisHand() As Long
Dim m_CurAxis As Long
Dim m_avaDevs() As DEVLIST
```

表框加载之后，可通过 API “Acm_GetAvailableDevs” 找到可用设备。代码如下：

```
Private Sub Form_Load()
    Dim Result As Long
    Dim i, DeviceNumber As Long
    Dim strTemp As String
    ReDim m_avaDevs(16)
ReDim m_AxisHand(32)
//Get available devices by Acm_GetAvailableDevs
         Result  =  Acm_GetAvailableDevs(m_avaDevs(0),  MAX_DEVICES,
  DeviceNumber)
    If Result <> SUCCESS Then
       MsgBox "no available device in system", vbOKOnly, "error"
       Exit Sub
    End If
    If DeviceNumber <> 0 Then
        m_dwDevNum = m_avaDevs(0).dwDeviceNum
        tx_DevNum.Text = "0x" + Hex(m_dwDevNum)
        Timer1.Interval = 200
    Else
           MsgBox "no available device in system", vbOKOnly, "error"
    End IfEnd Sub
```

单击 “Open Device&Axes” 打开设备及设备中的轴。定时器启用。下拉列表框中包括所有轴。代码如下：

```
Private Sub btn_OpenDev_Click()
    Dim Result As Long, i As Long, slaveDevs() As Long
    Dim strTemp As String
    Dim buffLen As Long
    Dim AxisNumber As Long
   //Open device.
    Result = Acm_DevOpen(m_dwDevNum, m_DevHand)
    If Result <> SUCCESS Then
        MsgBox "Open Device Failed", vbOKOnly, "PTP"
        Exit Sub
    End If

buffLen = 64
// Get Axis count by getting property.
     Result = Acm_GetProperty(m_DevHand, FT_DevAxesCount, AxisPerDev,
  buffLen)
    If Result <> SUCCESS Then
        Acm_DevClose (m_DevHand)
        MsgBox "get axis number error", vbOKOnly, "PTP"
        Exit Sub
    End If
```

```
    // Open all of axes
For AxisNumber = 0 To AxisPerDev - 1 Step 1
                    Result    =    Acm_AxOpen(m_DevHand,    AxisNumber,
   m_AxisHand(AxisNumber))
        If Result <> SUCCESS Then
            MsgBox "Open Axis Failed", vbOKOnly, "PTP"
            Exit Sub
        End If
        Acm_AxSetCmdPosition m_AxisHand(AxisNumber), 0
        If Result <> SUCCESS Then
            MsgBox "Set command position failed", vbOKOnly, "PTP"
            Exit Sub
        End If
        strTemp = AxisNumber & "-Axis"
        cm_Axis.AddItem strTemp
    Next
    cm_Axis.ListIndex = 0
    m_CurAxis = 0
    Timer1.Enabled = True
End Sub
```

单击下拉列表框选择轴，代码如下：

```
Private Sub cm_Axis_Click()
    m_CurAxis = cm_Axis.ListIndex
End Sub
```

定时器用于获取所选轴的理论位置。代码如下：

```
Private Sub Timer1_Timer()
    Dim CurPos() As Double
    Dim strTemp As String
ReDim CurPos(32)
// Get command position of selected axis
    Acm_AxGetCmdPosition m_AxisHand(m_CurAxis), CurPos(m_CurAxis)
    strTemp = CurPos(m_CurAxis)
    tx_CmdPos.Text = strTemp
End Sub
```

单击“Close Device&Axes”关闭设备及设备中的轴。定时器禁用。代码如下：

```
Private Sub btn_Close_Click()
    Dim AxisNum As Long
    For AxisNum = 0 To AxisPerDev - 1 Step 1
        Acm_AxClose m_AxisHand(AxisNum)
    Next
    Acm_DevClose m_DevHand
    cm_Axis.Clear
    Timer1.Enabled = False
End Sub
```

6. 结果如下：

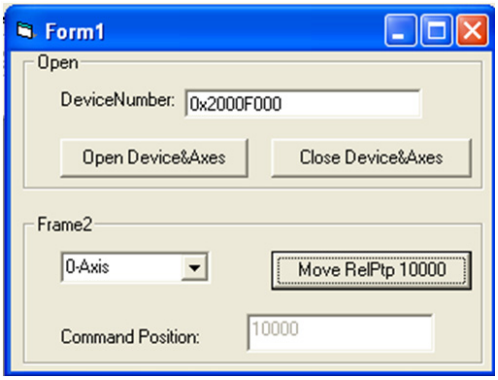


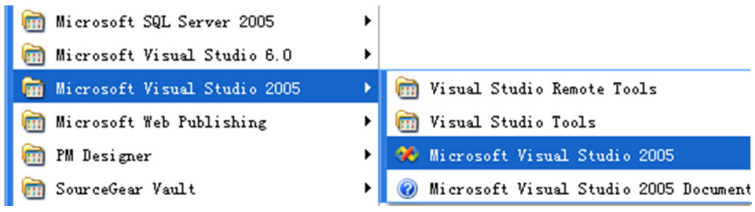
图 6.15： 执行结果

6.2.6.3 创建一个新的 C# 应用

如需使用 PCI-1245L SoftMotion PCI 控制器，则需要 ADVMOT.dll 和相关驱动文件。请确认开发前已安装驱动。

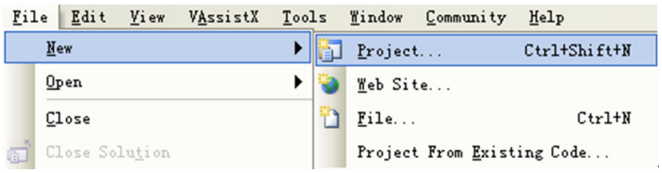
请按照以下步骤创建一个 C# 工程：

1. **创建一个新的工程**
   从“Start - Microsoft Visual Studio 2005”中选择 [Microsoft Visual Studio 2005]，如下图所示：

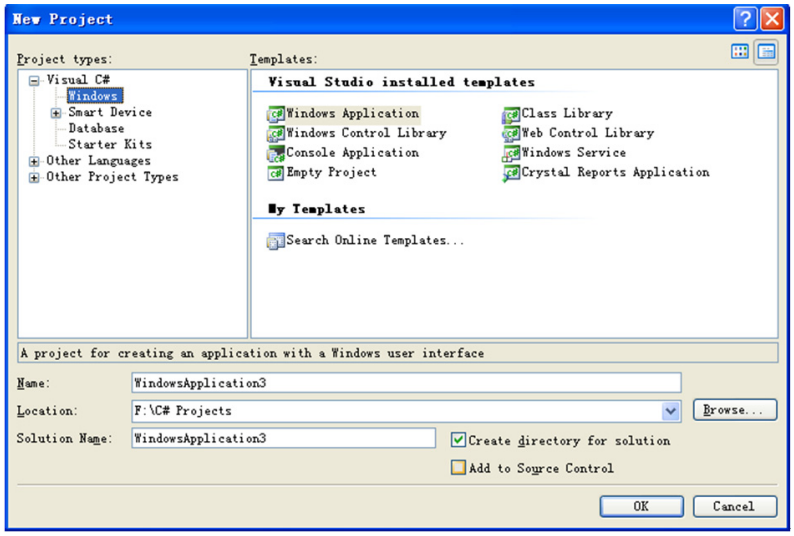


Microsoft Visual Studio 2005 的开发环境如下图所示：

要创建一个新的工程，请从主菜单选择 [File] ---> [New] ---> [Project]，如下图所示：

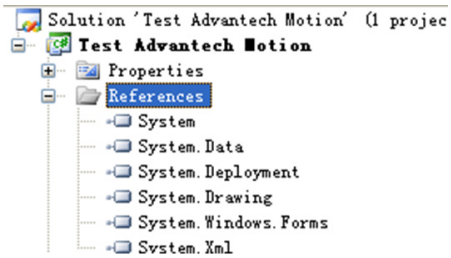


在新的窗口中，默认语言为 “Visual C#”。选择 [Windows Application] 模板、设置 “Name”、“Location” 和 “Solution Name” （保留默认），然后单击 ”OK “。

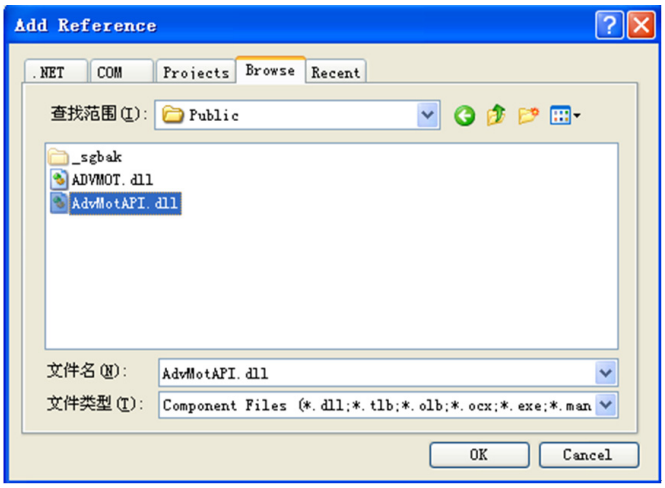


2. **添加相关 DLL 文件的引用**

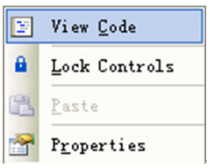
a. 单击开发环境右上角处的 [References]，如下图所示：



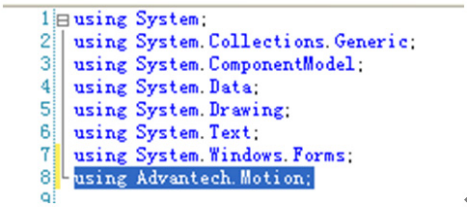
b. 单击 [Add Reference] 对话框的 [Browse]，从搜索路径选择 “Public” 文件夹中的 “AdvMotAPI.dll”，然后单击 [OK]，如下图所示：



c. 在编辑界面上右击，选择 [View Code] 进入程序源代码编辑界面，如下图所示：



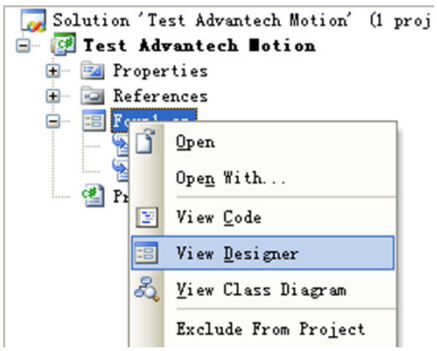
d. 在原始参考命名空间下添加 “using Advantech.Motion”，如下图所示：



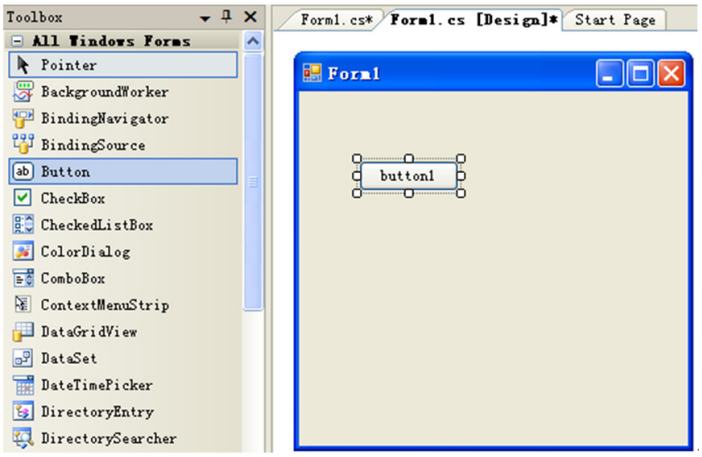
3. **编码**

a. UI 设计

双击 [Form1.cs] 或在 [Form1.cs] 上右击选择 [View Designer]，将出现 UI 编辑界面，如下图所示：



用户可从左边的工具栏中拖动任何控件 / 组件来编辑用户界面，如下图所示：



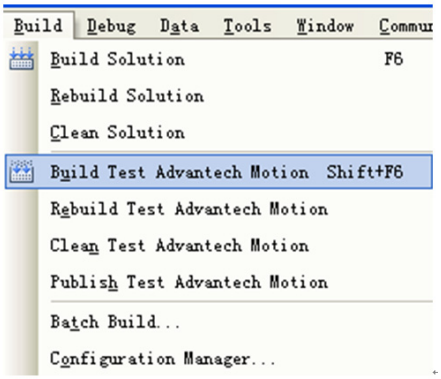
有关详细信息，请参考 Microsoft Visual C # 用户手册。

b. 编码

在 [Form1.cs] 上右击选择 [View Code]，将进入编码界面。用户可在控件 / 组件的相关方法 / 事件中进行编码。有关详细信息，请参考 PCI-1245L 的 C# 示例。

4. **测试程序**

用户编程之后或想要编译程序，可从菜板栏中单击 [Build] ---> [Build Solution]\[Build Test Advantech Motion]，如下图所示：



用户可直接单击工具栏中的 ，程序将一直运行 （如果没有发生错误）。

如果用户想要调试程序，可通过 [F9] 键在代码的相应行设置断点，如下图所示：

```
namespace Test_Advantech_Motion
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
    }
}
```

单击 [Debug] ---> [Start Debugging] 开始调试。当运行到断点时，用户可通过 [F11] 或 [F10] 键进入 / 跳过，如下图所示：

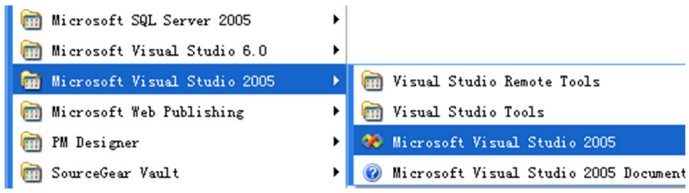
6.2.6.4 创建一个新的 VB.net 应用

如需使用 PCI-1245L，则需要 ADVMOT.dll 和相关驱动文件。请确认开发前已安装驱动。

请按照以下步骤创建一个 Visual Basic 工程：

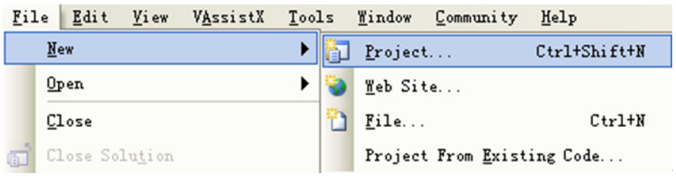
1. **创建一个新的工程**

   从 “Start - Microsoft Visual Studio 2005” 中选择 [Microsoft Visual Studio 2005]，如下图所示：

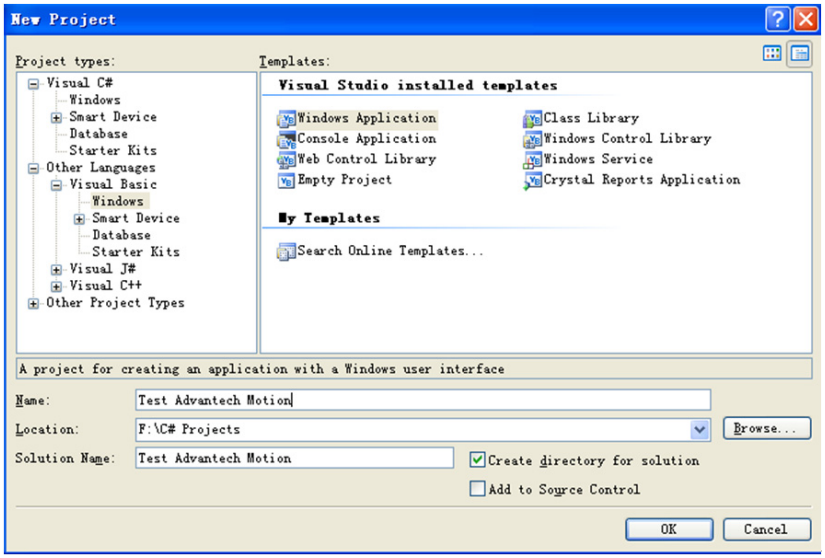


Microsoft Visual Studio 2005 的开发环境如下图所示：

要创建一个新的工程，请从主菜单选择 [File] ---> [New] ---> [Project]，如下图所示：

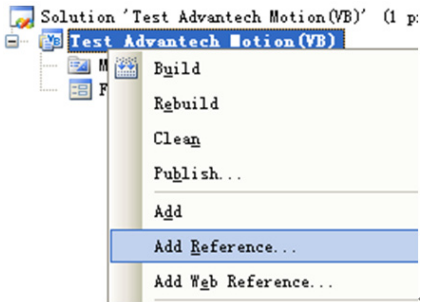


在新的窗口中，选择 [Other Languages]--->[Visual Basic] 并选择 [Windows Application] 模板、设置 “Name”、“Location” 和 “Solution Name” （保留默认），然后单击 ”OK “。

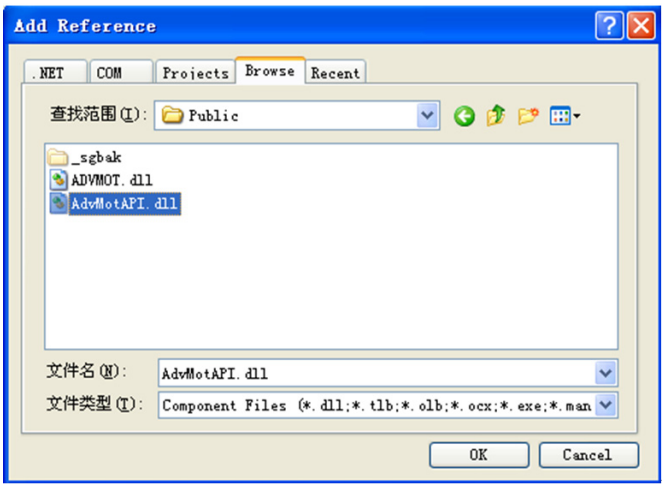


2. **添加相关 DLL 文件的引用**

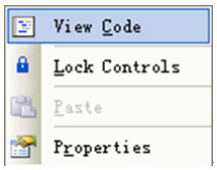
a. 单击开发环境右上角处的 [References]，如下图所示：



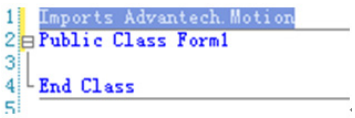
b. 单击 [Add Reference] 对话框的 [Browse]，从搜索路径选择 “Public” 文件夹中的 “AdvMotAPI.dll”，然后单击 [OK]，如下图所示：



c. 在编辑界面上右击，选择[View Code]进入程序源代码编辑界面，如下图所示：



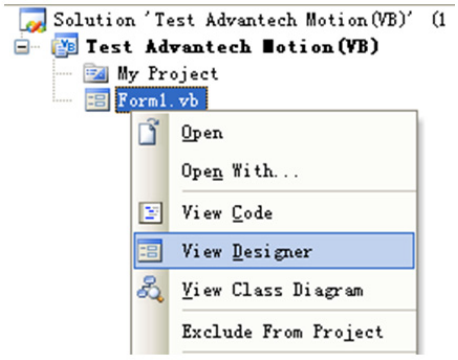
d. 在原始参考命名空间下添加 “Imports Advantech.Motion”，如下图所示：



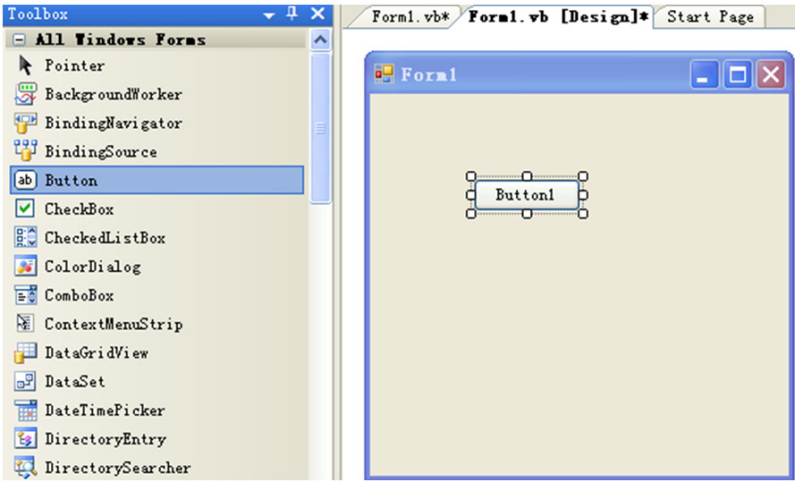
3. **编码**

a. UI 设计

双击 [Form1.vb] 或在 [Form1.vb] 上右击选择 [View Designer]，将出现 UI 编辑界面，如下图所示：



用户可从左边的工具栏中拖动任何控件 / 组件来编辑用户界面，如下图所示：



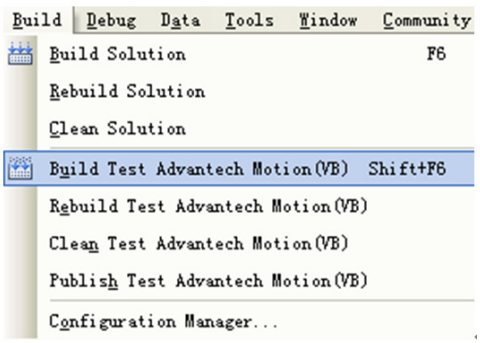
有关详细信息，请参考 Microsoft Visual Basic 用户手册。

b. 编码

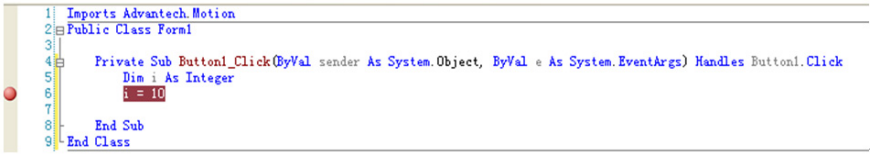
在 [Form1.vb] 上右击选择 [View Code]，将进入编码界面。用户可在控件 / 组件的相关方法/事件中进行编码。有关详细信息，请参考PCI-1245L的VB.NET示例。

4. **测试程序**

用户编程之后或想要编译程序，可从菜板栏中单击 [Build] ---> [Build Solution]\[Build Test Advantech Motion(VB)] ，如下图所示：



用户可直接单击工具栏中的，程序将一直运行 （如果没有发生错误）。
如果用户想要调试程序，可通过 [F9] 键在代码的相应行设置断点，如下图所示：



单击 [Debug] ---> [Start Debugging] 开始调试。当运行到断点时，用户可通过 [F11] 或 [F10] 键进入 / 跳过，如下图所示：

6.3 函数列表

6.3.1 通用 API

6.3.1.1 Acm_GetAvailableDevs

格式：

U32 Acm_GetAvailableDevs (DEVLIST *DeviceList, U32 MaxEntries, PU32 OutEntries)

目的：

获取已成功加载驱动的设备的可用设备编号和设备名称列表。

参数：

名称	类型	IN 或 OUT	说明
DeviceList	DEVLIST*	OUT	指针指向返回可用设备信息列表。
MaxEntries	U32	IN	需要获取的最大设备计数。
OutEntries	PU32	OUT	指向实际返回的设备数的指针。

返回值：

错误代码

注解：

DEVLIST 的结构为：

```
typedef struct tagPT_DEVLIST
{
    DWORD          DeviceNum;
    CHAR               DeviceName[50];
    SHORT             NumOfSubDevices;
} DEVLIST, *LPDEVLIST;
```

DeviceNum:

Acm_DevOpen 所需要的 Device Number。

DeviceName:

设备名。例如，PCI-1245L。

NumOfSubDevices:

仅用于 AMONET 系统，在 PCI-1245L 中始终为 0。

6.3.1.2 Acm_GetErrorMessage

格式：

BOOL Acm_GetErrorMessage (U32 ErrorCode, LPTSTR lpszError, U32 nMaxError)

目的：

根据 API 返回的错误代码，获取错误信息。

参数：

名称	类型	IN 或 OUT	说明
ErrorCode	U32	IN	API 返回的错误代码。
lpszError	LPTSTR	OUT	指针指向错误信息串。
nMaxError	U32	IN	接收错误信息的最大串长度。

返回值：

如果成功，返回非零；如果没有错误信息文本内容，返回 0。

注解：

Acm_GetErrorMessage 将复制 nMaxError -1 个字符到缓存，并在字符串末尾添加 \0。如果缓存过小，错误信息可能被截断。

6.3.1.3 Acm_DevWriteEEPROM_Ex

格式：

U32 Acm_DevWriteEEPROM_Ex(HAND DeviceHandle, U16 PrivateID, PU32 PassWordArray, U32 PassArrayCnt, PU32 WriteArray, U32 BufferLength)

目的：

根据 API 返回的错误代码，获取错误信息。

参数：

名称	类型	IN 或 OUT	说明
DeviceHandle	HAND	IN	设备句柄
PrivateID	U16	IN	0-7
PassWordArray	PU32	IN	密码数组指针。
PassArrayCnt	U32	IN	密码长度，一组的长度为 2。
WriteArray	PU32	IN	写入资料数组指针。
BufferLength	U32	IN	资料长度，一组的长度为 2。

返回值：

错误代码

注解：

密码和保护资料的出厂默认值为 0. 写入资料时，将不核对密码，自动覆盖原来的。

6.3.1.4 Acm_DevReadEEPROM_Ex

格式：

Acm_DevReadEEPROM_Ex(HAND DeviceHandle, U16 PrivateID, PU32 PassWordArray, U32 PassArrayCnt, PU32 ReadArray, U32 BufferLength)

目的：

输入正确的密码，读取私有资料。一共 8 组，每组密码为 8 bytes，资料为 8 bytes。

参数：

名称	类型	IN 或 OUT	说明
DeviceHandle	HAND	IN	设备句柄
PrivateID	U16	IN	0-7
PassWordArray	PU32	IN	密码数组指针。
PassArrayCnt	U32	IN	密码长度，一组的长度为 2。
ReadArray	PU32	OUT	接收读取的私有资料。
BufferLength	U32	IN	私有资料长度，一组的长度为 2。

返回值：

错误代码

注解：

读取时，如果密码输入错误超过 3 次，将无法读取。重启后，错误次数归零。

6.3.2 设备对象

6.3.2.1 Acm_DevOpen

格式：

U32 Acm_DevOpen (U32 DeviceNumber, PHAND DeviceHandle)

目的：

打开一个指定设备以获取设备句柄。

参数：

名称	类型	IN 或 OUT	说明
DeviceNumber	U32	IN	设备编号
DeviceHandle	PHAND	OUT	返回一个指针，指向设备句柄。

返回值：

错误代码

注解：

对设备执行任何操作之前，请先调用该函数。

6.3.2.2 Acm_DevClose

格式：

U32 Acm_DevClose (PHAND DeviceHandle)

目的：

关闭设备。

参数：

名称	类型	IN 或 OUT	说明
DeviceHandle	PHAND	IN	指针指向设备句柄。

返回值：

错误代码

注解：

最后，必须通过该函数关闭设备。

6.3.2.3 Acm_DevLoadConfig

格式：

U32 Acm_DevLoadConfig (HAND DeviceHandle, PI8 ConfigPath)

目的：

根据加载的配置文件，设置设备的所有配置。

参数：

名称	类型	IN 或 OUT	说明
DeviceHandle	HAND	IN	来自 Acm_DevOpen 的设备句柄。
ConfigPath	PI8	IN	指针指向保存配置文件路径的字符串。

返回值：

错误代码

注解：

配置文件可以为二进制或文本文件。如果文件扩展名为 .bin，则驱动会以二进制格式读取文件。否则，驱动将以 .INI （文本格式）读取文件。

用户应通过 Utility 调试设备并设置必要配置信息，然后将这些配置信息保存至文件。 通过调用 **Acm_DevLoadConfig**，用户应用程序可加载该配置文件。

如果用户想要将配置信息保存为.bin 文件格式，那么配置信息的保存数据结构 （MOT_DEV_CONFIG）应为：

```
typedef struct _MOT_AX_CONFIG
{
    ULONG PlsPerUnit;
    DOUBLE MaxVel;
    DOUBLE MaxAcc;
    DOUBLE MaxDec;
    DOUBLE MaxJerk;
    DOUBLE VelHigh;
    DOUBLE VelLow;
    DOUBLE Dec;
    DOUBLE Acc;
    ULONG PlsInMde;
    ULONG PlsInLgc;
    ULONG PlsInMaxFreq;
    ULONG PlsOutMde;
    ULONG AlmEnable;
    ULONG AlmLogic;
    ULONG AlmReact;
    ULONG InpEnable;
    ULONG InpLogic;
    ULONG ErcLogic;
    ULONG ErcEnMde;
    ULONG ElEnable;
    ULONG ElLogic;
    ULONG ElReact;
    ULONG SwMelEnable;
    ULONG SwPelEnable;
    ULONG SwMelReact;
    ULONG SwPelReact;
    ULONG SwMelValue;
    ULONG SwPelValue;
    ULONG OrgLogic;
    ULONG OrgReact;
    ULONG EzLogic;
    ULONG HomeModeEx;
    ULONG HomeExSwitchMode;
    DOUBLE HomeCrossDis;
    ULONG HomeResetEnable;
    ULONG BacklashEnable;
    ULONG BacklashPulses;
    ULONG BacklashVel;
    ULONG CmpSrc;
    ULONG CmpMethod;
    ULONG CmpPulseLogic;
```

```
    ULONG CmpPulseWidth;
    ULONG CmpEnable;
    ULONG CmpPulseMode;
    ULONG LatchLogic;
    ULONG LatchEnable;
    ULONG GenDoEnable;
    ULONG ExtMasterSrc;
    ULONG ExtSelEnable;
    ULONG ExtPulseNum;
    ULONG ExtPulseInMode;
    ULONG ExtPresetNum;
    ULONG CamDoEnable;
    ULONG CamDOLoLimit;
    ULONG CamDOHiLimit;
    ULONG CamDoCmpSrc;
    ULONG CamDoLogic;
    ULONG ModuleRange;
    ULONG SimStartSource;
} MOT_AX_CONFIG, *PMOT_AX_CONFIG;

typedef struct _MOT_DAQ_CONFIG
{
    ULONG AiChanType;
    ULONG AiRanges;
} MOT_DAQ_CONFIG, *PMOT_DAQ_CONFIG;

typedef struct _MOT_DEV_CONFIG
{
    MOT_DAQ_CONFIG DaqConfig;
    MOT_Ax_CONFIG Axis_Cfg[Axis_Num];
} MOT_DEV_CONFIG, *PMOT_DEV_CONFIG;
```

对于 PCI-1245L，Axis_Num 为 4。

6.3.2.4 Acm_GetProperty

格式：

U32 Acm_GetProperty(HAND Handle, U32 ProperyID, PVOID Buffer, PU32 BufferLength)

目的：

通过分配的 PropertyID 获取属性 （特性属性、配置属性或参数属性）值。

参数：

名称	类型	IN 或 OUT	说明
Handle	HAND	IN	对象句柄。 该句柄可以是来自 Acm_DevOpen 的设备句柄，或来自 Acm_AxOpen 的轴句柄，或来自 Acm_GpAddAxis 的群组句柄。
ProperyID	U32	IN	要查询的属性 ID。
Buffer	PVOID	OUT	属性的数据缓存。

BufferLength	PU32	IN/OUT	IN，属性的缓存大小； OUT，返回的数据所需长度。

返回值：

错误代码

注解：

用户应注意数据类型或**缓存**的 **BufferLength**，获取相应 PropertyID 的属性值。如果**缓存**过小，返回值将为错误代码 **“InvalidInputParam”**。因此，驱动将返回 BufferLength 中属性的实际大小。

有关 PerpertyID 的详细信息，请参考属性列表。

6.3.2.5 Acm_SetProperty

格式：

U32 Acm_SetProperty (HAND Handle, U32 ProperyID, PVOID Buffer, U32 BufferLength).

目的：

设定指定的 PropertyID 对应的属性值。

参数：

名称	类型	IN 或 OUT	说明
Handle	HAND	IN	对象句柄。 该句柄可以是来自 Acm_DevOpen 的设备句柄，或来自 Acm_AxOpen 的轴句柄，或来自 Acm_GpAddAxis 的群组句柄。
ProperyID	U32	IN	要设置的属性 ID。
Buffer	PVOID	IN	属性的数据缓存。
BufferLength	U32	IN	属性的缓存大小。

返回值：

错误代码

注解：

对于一些属性，鉴于精确度考虑，驱动中会对某些属性值进行调整。 因此，这些属性的输出值也许和输入值不同，如 PAR_AxJerk。

在属性列表中，并非所有属性均能设置新的属性值，只有可写属性的属性值可重新设置。

用户应注意所需的数据类型和数据长度属性。 如果 **BufferLength** 的值小于实际数据大小，将返回错误代码 “InvalidInputParamter”。

有关 PerpertyID 的详细信息，请参考属性列表。

6.3.2.6 Acm_GetLastError

格式：

U32 Acm_GetLastError (HAND ObjectHandle)

目的：

获取设备、轴或群组最后一个的错误代码。

参数：

名称	类型	IN 或 OUT	说明
DeviceHandle	HAND	IN	对象句柄。 该句柄可以是来自 Acm_DevOpen 的设备句柄，或来自 Acm_AxOpen 的轴句柄，或来自 Acm_GpAddAxis 的群组句柄。

返回值：

错误代码

注解：

有关错误代码的详细信息，请参考 Acm_GetErrorMessage。

6.3.2.7 Acm_CheckMotionEvent

格式：

U32 Acm_CheckMotionEvent (HAND DeviceHandle, PU32 AxEvtStatusArray, PU32 GpEvtStatusArray, U32 AxArrayElements, U32 GpArrayElements, U32 Millisecond)

目的：

检测轴和群组的事件状态。

参数：

名称	类型	IN 或 OUT	说明
DeviceHandle	HAND	IN	来自 Acm_DevOpen 的设备句柄。
AxEvtStatusArray	PU32	IN	Array[n]：返回的每个轴的中断事件状态。N 表示运动设备的轴的个数。 每个阵列元素为 32-bit 数据类型，每个 bit 都表示不同的事件类型： Bit n = 1：事件发生； Bit n = 0：事件未发生或禁用。
GpEvtStatusArray	PU32	IN	Array[n]：返回的每个群组的中断事件状态。GpEvtStatusArray 是一个 32-bit 数据类型阵列，其中的第一个元素用于表示群组的运动结束事件，其中每一个 bit 用于表示相应 GroupID 的运动结束事件是否发生；第二个元素用于表示群组的运行速度达到事件，其中每一个 bit 用于表示相应 GroupID 的群组的此事件是否发生；第三个元素用于表示群组的减速开始事件，其中每一个 bit 用于表示相应 GroupID 的群组的此事件是否发生。用户可以根据自己的需要设置此数组。 Bit n = 1：事件发生； Bit n = 0：事件未发生或禁用。 注：EVT_GPn_MOTION_DONE/EVT_GPn_VH_START/ EVT_GPn_VH_END，n 为 GroupID。 可从 PAR_GpGroupID 属性获取。
AxArrayElements	U32	IN	AxEvtStatusArray 中元素个数。
GpArrayElements	U32	IN	GpEvtStatusArray 中元素个数。
Millisecond	U32	IN	设定每次 Check 事件时的等待时间。

AxEvtStatusArray 的 Bit 定义：

Bit	31…6	5	4	3	2	1	0
Description	Reserved	EVT_AX_VH_END	EVT_AX_VH_START	EVT_AX_ERROR	EVT_AX_LATCHED (PCI-1245L not support)	EVT_AX_COMPARED (PCI-1245L not support)	EVT_AX_MOTION_DONE

GpEvtStatusArray 的 Bit 定义：

Bit	Data	31…n	1	0
说明	GpEnableEvtArray[0]	EVT_GPn_MOTION_DONE	EVT_GP1_MOTION_DONE	EVT_GP0_MOTION_DONE
	GpEnableEvtArray[1]	EVT_GPn_VH_START	EVT_GP1_VH_START	EVT_GP0_VH_START
	GpEnableEvtArray[2]	EVT_GPn_VH_END	EVT_GP1_VH_END	EVT_GP0_VH_END

返回值：

错误代码

注解：

如果用户想要获取轴或群组的事件状态，可通过调用 Acm_EnableMotionEvent 启用这些事件。
用户应创建一个新的线程，用于检查事件状态。

6.3.2.8 Acm_EnableMotionEvent

格式：

U32 Acm_EnableMotionEvent (HAND DeviceHandle, PU32 AxEnableEvtArray, PU32 GpEnableEvtArray, U32 AxArrayElements, U32 GpArrayElements)

目的：

启用轴和群组的事件状态。

参数：

名称	类型	IN 或 OUT	说明
DeviceHandle	HAND	IN	来自 Acm_DevOpen 的设备句柄。
AxEnableEvtArray	PU32	IN	Array[n]，启用每个轴的中断事件，n 表示运动设备的轴个数。 每个阵列元素为 32-bit 数据类型，每个 bit 都表示不同的事件类型： Bit n = 1：启用事件； Bit n = 0：禁用事件。
GpEnableEvtArray	PU32	IN	Array[n]：启用每个群组的中断事件。GpEnableEvtArray 是一个 32-bit 数据类型阵列，其中的第一个元素用于设置群组的运动结束事件，其中每一个 bit 用于设置相应 GroupID 的群组的运动结束事件是否使能；第二个元素用于设置群组的运行速度达到事件，其中每一个 bit 用于设置相应 GroupID 的群组的此事件是否使能；第三个元素用于设置群组的减速开始事件，其中每一个 bit 用于设置相应 GroupID 的群组的此事件是否使能。用户可以根据自己的需要设置此数组。 Bit n = 1：启用事件； Bit n = 0：禁用事件。 注：对于 EVT_GPn_MOTION_DONE/ EVT_GPn_VH_START/EVT_GPn_VH_END，n 为 GroupID。可从 PAR_GpGroupID 属性获取。
AxArrayElements	U32	IN	AxEvtStatusArray 元中元素个数。
GpArrayElements	U32	IN	GpEvtStatusArray 中元素个数。
Millisecond	U32	IN	每次检查以毫秒为单位制定超时值

AxEnableEvtArray 说明：

Bit	31..6	5	4	3	2	1	0
Description	Reserved	EVT_AX_VH_END	EVT_AX_VH_START	EVT_AX_ERROR	EVT_AX_LATCHED (PCI-1245L not support)	EVT_AX_COMPARED (PCI-1245L not support)	EVT_AX_MOTION_DONE

GpEnableEvtArray 说明：

Bit	Data	31...n	1	0
说明	GpEnableEvtArray[0]	EVT_GPn_MOTION_DONE	EVT_GP1_MOTION_DONE	EVT_GP0_MOTION_DONE
	GpEnableEvtArray[1]	EVT_GPn_VH_START	EVT_GP1_VH_START	EVT_GP0_VH_START
	GpEnableEvtArray[2]	EVT_GPn_VH_END	EVT_GP1_VH_END	EVT_GP0_VH_END

返回值：

错误代码

注解：

启用轴或群组的一些事件之后，可从 Acm_CheckMotionEvent 获取事件状态。

6.3.3 轴

6.3.3.1 系统

6.3.3.1.1 Acm_AxOpen

格式：

U32 Acm_AxOpen (HAND DeviceHandle, U16 PhyAxis, PHAND AxisHandle)

目的：

打开指定轴，获取该轴的对象句柄。

参数：

名称	类型	IN 或 OUT	说明
DeviceHandle	HAND	IN	来自 Acm_DevOpen 的设备句柄。
PhyAxis	U16	IN	物理轴编号。（PCI-1245L：0、1、2、3。）
AxisHandle	PHAND	OUT	返回一个指针，指向轴句柄。

返回值：

错误代码

注解：

进行任何轴操作之前，应首先调用该 API。PCI-1245L 的物理轴编号为：0、1、2、3。

6.3.3.1.2 Acm_AxClose

格式：

U32 Acm_AxClose (PHAND AxisHandle)

目的：

关闭已经打开的轴。

参数：

名称	类型	IN 或 OUT	说明
AxisHandle	PHAND	IN	指针指向轴句柄。

返回值：

错误代码

注解：

调用该 API 之后，将不能再使用轴句柄。

6.3.3.1.3 Acm_AxResetError

格式：

U32 Acm_AxResetError (HAND AxisHandle)

目的：

复位轴的状态。如果轴处于 “ErrorStop” 状态，则调用该函数后状态将变为 “Ready ” 。

参数：

名称	类型	IN 或 OUT	说明
AxisHandle	HAND	IN	来自 Acm_AxOpen 的轴句柄。

返回值：

错误代码

注解：

6.3.3.2 运动 IO

6.3.3.2.1 Acm_AxSetSvOn

格式：

U32 Acm_AxSetSvOn (HAND AxisHandle, U32 OnOff)

目的：

打开 / 关闭伺服驱动器。

参数：

名称	类型	IN 或 OUT	说明
AxisHandle	HAND	IN	来自 Acm_AxOpen 的轴句柄。
OnOff	U32	IN	设置 SVON 信号的动作。 0：关闭； 1：打开。

返回值：

错误代码

注解：

6.3.3.2.2 Acm_AxGetMotionIO

格式：

U32 Acm_AxGetMotionIO (HAND AxisHandle, PU32 Status)

目的：

获取轴的运动 I/O 状态。

参数：

名称	类型	IN 或 OUT	说明
AxisHandle	HAND	IN	来自 Acm_AxOpen 的轴句柄。
Status	PU32	OUT	位 说明 0：RDY---- RDY 针脚输入； 1：ALM ---- 报警信号输入； 2：LMT+ ---- 限位开关 + ； 3：LMT- ---- 限位开关 - ； 4：ORG---- 原始开关； 5：DIR ---- DIR 输出； 6：EMG ---- 紧急信号输入； 7：PCS ---- PCS 信号输入 （PCI-1245L 不支持）； 8：ERC ---- 输出偏转计数器清除信号至伺服电机驱动 (OUT7)； 9：EZ ---- 编码器 Z 信号； 10：CLR ---- 外部输入至清除位置计数器 （PCI-1245L 不支持）； 11：LTC ---- 锁存信号输入 (PCI-1245L 不支持)； 12：SD ---- 减速信号输入 （PCI-1245L 不支持）； 13：INP ---- 到位信号输入； 14：SVON ---- 伺服开启 （OUT6）； 15：ALRM ----报警复位输出状态； 16：SLMT+ ---- 软件限位 + ； 17：SLMT- ---- 软件限位 - ； 18：CMP-----比较信号 （OUT5）(PCI-1245L 不支持)； 19：CAMDO ---- 凸轮区间 DO （OUT4）(PCI-1245L 不支持)。

返回值：

错误代码

注解：

6.3.3.3 运动状态

6.3.3.3.1 Acm_AxGetMotionStatus

格式：

U32 Acm_AxGetMotionStatus (HAND AxisHandle, PU32 Status)

目的：

获取轴的当前运动状态。

参数：

名称	类型	IN 或 OUT	说明
AxisHandle	HAND	IN	来自 Acm_AxOpen 的轴句柄。
Status	PU32	OUT	位 说明 0：Stop ---- 停止； 1：Res1 ---- 保留； 2：WaitERC---- 等待 ERC 完成； 3：Res2 ---- 保留； 4：CorrectBksh ---- 背隙补偿； 5：Res3 ---- 保留； 6：InFA ---- 处于特定速度中 = FA; 7：InFL ---- 处于低速中 = FL； 8：InACC ---- 加速中； 9：InFH ---- 处于最大速度中 = FH； 10：InDEC ---- 减速中； 11：WaitINP---- 到位等待。

返回值：

错误代码

注解：

6.3.3.3.2 Acm_AxGetState

格式：

U32 Acm_AxGetState (HAND AxisHandle, PU16 State)

目的：

获取轴的当前状态。

参数：

名称	类型	IN 或 OUT	说明
AxisHandle	HAND	IN	来自 Acm_AxOpen 的轴句柄。

Status	PU16	IN	轴状态： 值　　说明 0：STA_AxDisable ---- 轴被禁用，用户可打开并激活。 1：STA_AxReady ---- 轴已准备就绪，等待新的命令。 2：STA_Stopping ---- 轴停止。 3：STA_AxErrorStop ---- 出现错误，轴停止。 4：STA_AxHoming ---- 轴正在执行返回原点运动。 5：STA_AxPtpMotion ---- 轴正在执行 PTP 运动。 6：STA_AxContiMotion ---- 轴正在执行连续运动。 7：STA_AxSyncMotion ---- 轴在一个群组中，群组正在执行插补运动；或轴是一个从轴，正在执行 E-cam/E-gear/Gantry 运动。 8：STA_AX_EXT_JOG ---- 轴由外部信号控制。当外部信号激活时，轴将执行 JOG 模式运动。 9：STA_AX_EXT_MPG ---- 轴由外部信号控制。当外部信号激活时，轴将执行 MPG 模式运动。

返回值：

错误代码

注解：

6.3.3.4 速度运动

6.3.3.4.1 Acm_AxMoveVel

格式：

U32 Acm_AxMoveVel (HAND AxisHandle, U16 Direction)

目的：

命令轴按照规定速度进行没有终点的运动。

参数：

名称	类型	IN 或 OUT	说明
AxisHandle	HAND	IN	来自 Acm_AxOpen 的轴句柄。
Direction	U16	IN	方向： 0：正方向； 1：负方向。

返回值：

错误代码

注解：

速度曲线由以下属性决定：PAR_AxVelLow、PAR_AxVelHigh、PAR_AxAcc、PAR_AxDec 和 PAR_AxJerk。

6.3.3.4.2 Acm_AxChangeVel

格式：

U32 Acm_AxChangeVel (HAND AxisHandle, F64 NewVelocity)

目的：

当轴在运动过程中，命令轴改变速度。

参数：

名称	类型	IN 或 OUT	说明
AxisHandle	HAND	IN	来自 Acm_AxOpen 的轴句柄。
NewVelocity	F64	IN	新速度。（单位 = PPU/s）

返回值：

错误代码

注解：

速度曲线由以下属性决定：PAR_AxVelLow、**NewVelocity**、PAR_AxAcc、PAR_AxDec 和 PAR_AxJerk。NewVelocity 的范围： 0 ~ CFG_AxMaxVel。

若此命令成功下达，在下次运动之前若未重新设定速度，则 NewVelocity 会作用到下次运动中。

6.3.3.4.3 Acm_AxGetCmdVelocity

格式：

U32 Acm_AxGetCmdVelocity (HAND AxisHandle, PF64 Velocity)

目的：

获取指定轴的当前理论速度。

参数：

名称	类型	IN 或 OUT	说明
AxisHandle	HAND	IN	来自 Acm_AxOpen 的轴句柄。
Velocity	PF64	OUT	返回理论速度。（单位 = PPU/s）

返回值：

错误代码

注解：

6.3.3.4.4 Acm_AxChangeVelEx

格式：

U32 Acm_AxChangeVelEx (HAND AxisHandle, F64 NewVelocity, F64 NewAcc, F64 NewDec)

目的：

在运动的过程中可同时改变速度，加速度和减速度。

参数：

名称	类型	IN 或 OUT	说明
AxisHandle	HAND	IN	来自 Acm_AxOpen 的轴句柄。
NewVelocity	F64	IN	新的速度值，单位：PPU/s
NewAcc	F64	IN	新的加速度值，单位：PPU/s^2
NewDec	F64	IN	新的减速度值，单位：PPU/s^2

返回值：

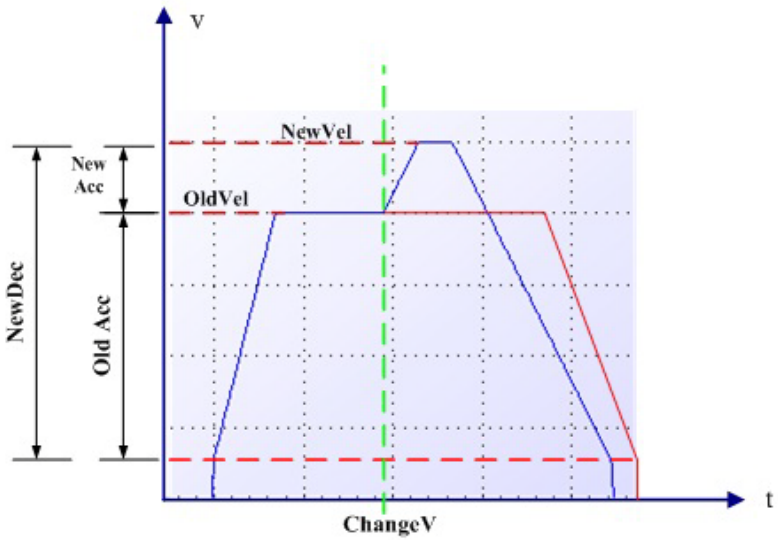
错误代码

注解：

NewVelocity 不能超过通过 CFG_AxMaxVel 设定的最大值，**NewAcc** 不能超过 CFG_AxMaxAcc 设定的最大加速度，**NewDec** 不能超过 CFG_AxMaxDec 设定的最大减速度。

若 **NewAcc** 或 **NewDec** 为 0，则使用上一次设定的加速度或减速度值。

若此命令成功下达，在下次运动之前若未重新设定速度，则 **NewVelocity** 会作用到下次运动中。



6.3.3.4.5 Acm_AxChangeVelExByRate

格式：

U32 Acm_AxChangeVelExByRate (HAND AxisHandle, U32 Rate, F64 NewAcc, F64 NewDec)

目的：

在运动的过程中可根据比率改变运行速度，同时可改变加速度和减速度。

参数：

名称	类型	IN 或 OUT	说明
AxisHandle	HAND	IN	来自 Acm_AxOpen 的轴句柄。
Velocity	U32	IN	速度改变百分值。NewVel = OldVel * Rate *0.01。
NewAcc	F64	IN	新的加速度值，单位：PPU/s^2
NewDec	F64	IN	新的减速度值，单位：PPU/s^2

返回值：

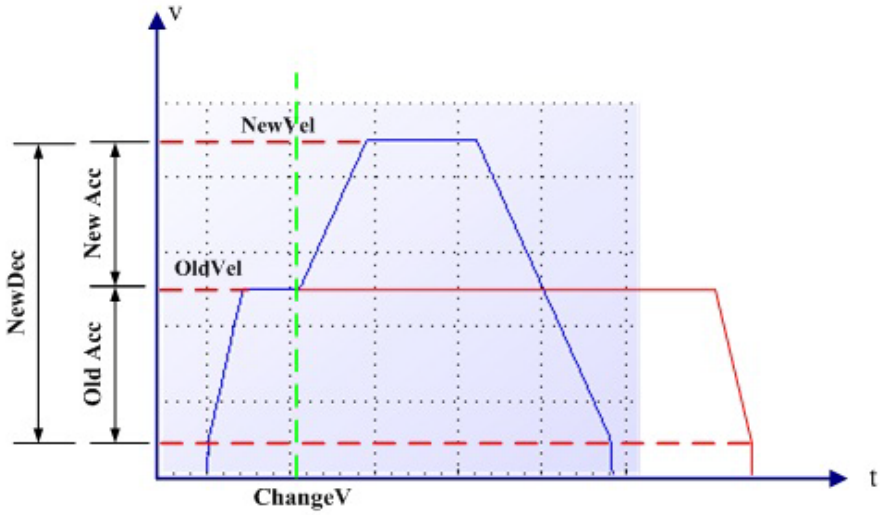
错误代码

注解：

NewVel = OldVel*Rate*0.01。根据 Rate 计算出来的 NewVel 不能超过通过 CFG_AxMaxVel 设定的最大值，NewAcc 不能超过 CFG_AxMaxAcc 设定的最大加速度，NewDec 不能超过 CFG_AxMaxDec 设定的最大减速度。

若 NewAcc 或 NewDec 为 0，则使用上一次设定的加速度或减速度值。

新的速度，NewAcc 和 NewDec 仅对当前运动有效。



6.3.3.4.6 Acm_AxChangeVelByRate

格式：

U32 Acm_AxChangeVelByRate (HAND AxisHandle, U32 Rate)

目的：

按照设定的比例改变当前正在执行的动作的运行速度。

参数：

名称	类型	IN 或 OUT	说明
AxisHandle	HAND	IN	来自 Acm_AxOpen 的轴句柄。
Rate	U32	IN	速度改变百分比值。

返回值：

错误代码

注解：

新速度 = 之前的运行速度 * Rate * 0.01。Rate须大于0且小于CFG_AxMaxVel与之前的速度的比值。新速度仅对当前运动有效。

6.3.3.5 点到点运动

6.3.3.5.1 Acm_AxMoveRel

格式：

U32 Acm_AxMoveRel (HAND AxisHandle, F64 Distance)

目的：

开始单轴的相对点到点运动。

参数：

名称	类型	IN 或 OUT	说明
AxisHandle	HAND	IN	来自 Acm_AxOpen 的轴句柄。
Distance	F64	IN	相对距离。（单位 = PPU）

返回值：

错误代码

注解：

速度曲线由以下属性决定：PAR_AxVelLow、PAR_AxVelHigh、PAR_AxAcc、PAR_AxDec 和 PAR_AxJerk。

Distance 的范围：-2147483647/PPU ~ 2147483647/PPU。

6.3.3.5.2 Acm_AxMoveAbs

格式：

U32 Acm_AxMoveAbs (HAND AxisHandle, F64 Position)

目的：

开始单轴的绝对点到点运动。

参数：

名称	类型	IN 或 OUT	说明
AxisHandle	HAND	IN	来自 Acm_AxOpen 的轴句柄。
Position	F64	IN	绝对位置。（单位 = PPU）

返回值：

错误代码

注解：

速度曲线由以下属性决定：PAR_AxVelLow、PAR_AxVelHigh、PAR_AxAcc、PAR_AxDec 和 PAR_AxJerk。

Distance 的范围：-2147483647/PPU ~ 2147483647/PPU。

6.3.3.5.3 Acm_AxChangePos

格式：

U32 Acm_AxChangePos (HAND AxisHandle, F64 NewDistance)

目的：

当轴处于点到点运动时，命令轴改变终点位置。

参数：

名称	类型	IN 或 OUT	说明
AxisHandle	HAND	IN	来自 Acm_AxOpen 的轴句柄。
NewDistance	F64	IN	新的相对距离。（单位 = PPU）

返回值：

错误代码

注解：

Distance 的范围： -2147483647/PPU ~ 2147483647/PPU。

6.3.3.6 同步起停运动

6.3.3.6.1 Acm_AxSimStartSuspendAbs

格式：

U32 Acm_AxSimStartSuspendAbs (HAND AxisHandle, F64 EndPoint)

目的：

设定轴为等待状态。当轴收到启动信号开始运动时，轴将开始点到点绝对运动，直至到达指定终止位置。

参数：

名称	类型	IN 或 OUT	说明
AxisHandle	HAND	IN	来自 Acm_AxOpen 的轴句柄。
EndPoint	F64	IN	运动到的绝对位置。（单位 = PPU）

返回值：

错误代码

注解：

如果不止一个轴想要进行同步操作，那么每个轴都需要调用该函数。

EndPoint 的范围：-2147483647/ PPU ~ 2147483647/ PPU。

6.3.3.6.2 Acm_AxSimStartSuspendRel

格式：

U32 Acm_AxSimStartSuspendRel (HAND AxisHandle, F64 Distance)

目的：

设定轴为等待状态。当轴收到启动信号开始运动时，轴将开始点到点相对运动，直至到达指定终止位置。

参数：

名称	类型	IN 或 OUT	说明
AxisHandle	HAND	IN	来自 Acm_AxOpen 的轴句柄。
EndPoint	F64	IN	运动到的相对位置。（单位 = PPU）

返回值：

错误代码

注解：

如果不止一个轴想要进行同步操作，那么每个轴都需要调用该函数。

EndPoint 的范围：-2147483647/ PPU ~ 2147483647/ PPU。

6.3.3.6.3 Acm_AxSimStartSuspendVel

格式：

U32 Acm_AxSimStartSuspendVel (HAND AxisHandle, U16 DriDir)

目的：

设定轴为等待状态。当轴收到启动信号开始运动时，轴将开始连续运动。

参数：

名称	类型	IN 或 OUT	说明
AxisHandle	HAND	IN	来自 Acm_AxOpen 的轴句柄。
DriDir	U16	IN	方向： 0：正方向； 1：负方向。

返回值：

错误代码

注解：

如果不止一个轴想要进行同步操作，那么每个轴都需要调用该函数。

6.3.3.6.4 Acm_AxSimStart

格式：

U32 Acm_AxSimStart (HAND AxisHandle)

目的：

发出同步启动信号，以启动所有等待启动触发的轴。

参数：

名称	类型	IN 或 OUT	说明
AxisHandle	HAND	IN	来自 Acm_AxOpen 的轴句柄。

返回值：

错误代码

注解：

如果不止一个轴等待启动触发，那么在调用该 API 之前，用户应通过 CFG_AxSimStartSource 设置同时开始 / 停止的模式。

6.3.3.6.5 Acm_AxSimStop

格式：

U32 Acm_AxSimStop (HAND AxisHandle)

目的：

发出同步停止信号，以停止所有等待停止触发的轴。

参数：

名称	类型	IN 或 OUT	说明
AxisHandle	HAND	IN	来自 Acm_AxOpen 的轴句柄。

返回值：

错误代码

注解：

进行同步操作时，如果使用 STA 信号启动模式，则用户能够在任一轴上进行该操作以停止所有轴。 否则，每个同时轴都需要调用该 API 才能停止同时运动。

有关同步启停模式的详细信息，请参考 CFG_AxSimStartSource。

6.3.3.7 原点

6.3.3.7.1 Acm_AxHome

格式：

U32 Acm_AxHome (HAND AxisHandle, U32 HomeMode, U32 Dir)

目的：

命令轴开始回原点运动，典型原点运动的 16 种类型组成扩展原点。

参数：

名称	类型	IN 或 OUT	说明
AxisHandle	HAND	IN	来自 Acm_AxOpen 的轴句柄。
HomeMode	U32	IN	HomeMode： 0：MODE1_Abs； 1：MODE2_Lmt； 2：MODE3_Ref； 3：MODE4_Abs_Ref； 4：MODE5_Abs_NegRef； 5：MODE6_Lmt_Ref； 6：MODE7_AbsSearch； 7：MODE8_LmtSearch； 8：MODE9_AbsSearch_Ref； 9：MODE10_AbsSearch_NegRef； 10：MODE11_LmtSearch_Ref； 11：MODE12_AbsSearchReFind； 12：MODE13_LmtSearchReFind； 13：MODE14_AbsSearchReFind_Ref； 14：MODE15_AbsSearchReFind_NegRef； 15：MODE16_LmtSearchReFind_Ref。
Dir	U32	IN	0：正方向； 1：负方向。

返回值：

错误代码

注解：

在 MODE3_Ref~MODE16_LmtSearchReFind_Ref 的 Home 过程中，某些阶段会使用初始速度，所以使用这些 HomeMode 时，通过 PAR_AxVelLow 设定的初始速度须大于 0。

如果属性 CFG_AxHomeResetEnable 设置为 “True”，则原点运动终止后，理论位置和实际位置将复位至零。使用该方法之前，应通过 PAR_AxHomeCrossDistance 设置跨越距离。

速度曲线由 PAR_AxVelLow、PAR_AxVelHigh、PAR_AxAcc、PAR_AxDec 和 PAR_AxJerk 决定。

说明：如下图示中标注的 a,b,c,d 含义分别如下：

a：梯形 PTP 运动到碰到 ORG/EL 信号后减速停止。

b：以 HomeCrossDistance 为距离单位进行梯形 PTP 运动，直至结束后 ORG/EL 信号无效。

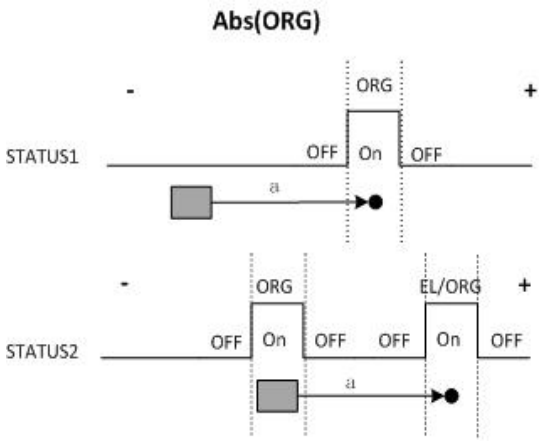
c：以 VelLow 进行等速运动，遇到 ORG/EL 信号后立即停止。

d：以 HomeCrossDistance 为距离单位以 VelLow 进行等速运动，直至结束后 ORG/EL 信号无效。

• ：小实心黑圆点表示一段运动的结束点。

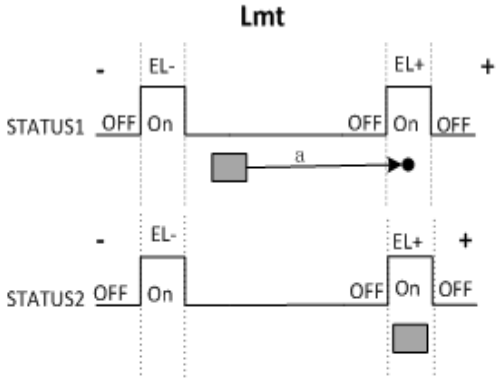
注！ *梯形 PTP 运动特性：开始时以 Acc 从 VelLow 加速到 VelHigh（若距离足够长），停止时以 Dec 从 VelHigh 减速到 VelLow）。*

1. MODE1_Abs: Move (Dir) ->touch ORG->Stop.
   只依照原点设备 （如传感器）返回原点。对象会一直运动，直至原点信号发生。
   比如：
   Dir：正。
   Org Logic (CFG_AxOrgLogic)：高准位。
   EL （硬限位开关）逻辑 (CFG_AxElLogic)：高准位。



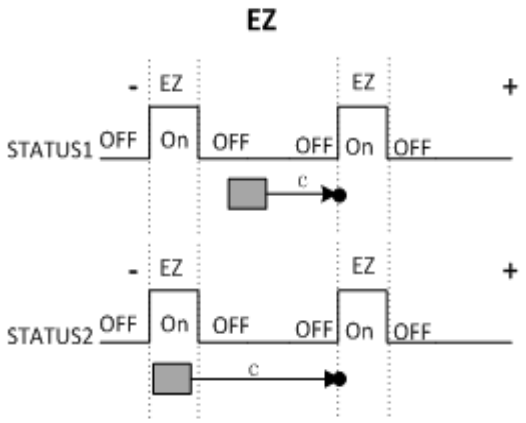
- STATUS1：如果对象超出 ORG 信号区域，当写入原点命令时，对象将一直运动直到 ORG 信号发生。
- STATUS2：如果对象在 ORG 信号区域内或和 ORG 开关的方向相反，对象将一直运动直到 ORG 信号 （如果有多于一个 ORG 开关或轴设备关闭）或 EL 信号 （轴处于发生错误停止状态）发生。

2. MODE2_Lmt: Move(Dir)->touch EL->Stop
   只依照限位设备 （如传感器）返回原点。对象会一直运动，直至限位信号发生。
   比如：
   Dir：正。
   限位逻辑 (CFG_AxElLogic)：高准位。



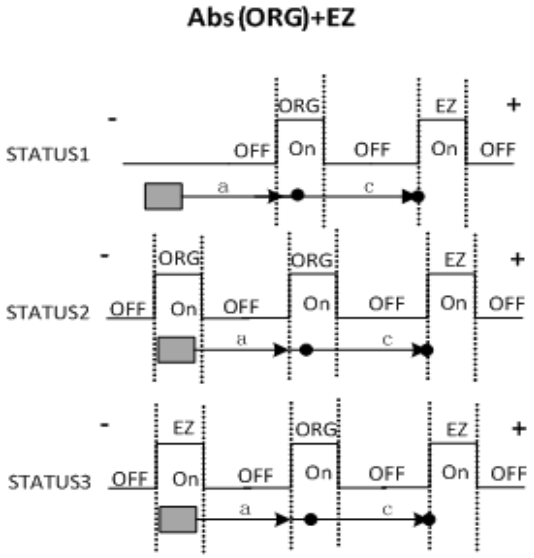
- STATUS1：如果对象超出 EL 信号区域，当写入原点命令时，对象将一直运动直到 EL 信号发生。
- STATUS2：如果对象在 EL 信号区域内，将不作出响应。

3. MODE3_Ref: Move (Dir) ->touch EZ->Stop.
只按照 EZ 返回原点。 对象会一直运动，直至 EZ 信号发生。
比如：
Dir：正。
EZ 逻辑 (CFG_AxEzLogic)：高准位。



- STATUS1：如果对象超出 EZ 信号区域，当写入原点命令时，对象将一直运动直到 EZ 信号发生。
- STATUS2：如果对象在 EZ 信号区域内，对象将一直运动直到 EZ 信号发生。

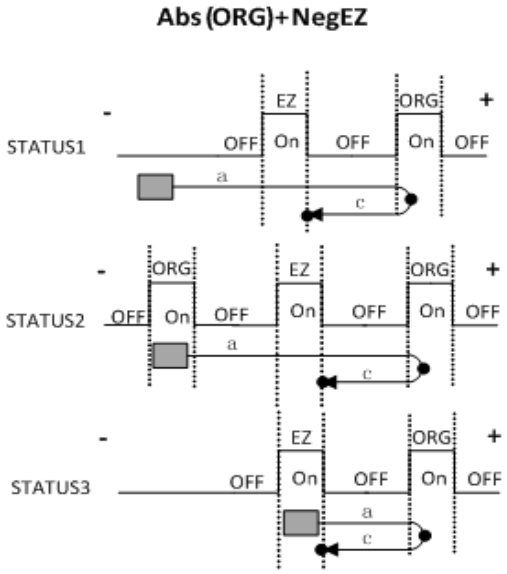
4. MODE4_Abs_Ref: ORG+EZ, Move(Dir) ->touch ORG ->Stop ->Move(Dir)->touch EZ ->Stop
这是一种组合的原点模式。首先，对象将一直运动直到原始信号发生，然后将保持以 ORG 同一方向继续运动，直到 EZ 信号发生。
比如：
Dir：正。
ORG 逻辑：高准位。
EZ 逻辑：高准位。



- STATUS1：如果对象超出 EZ 信号和 ORG 信号区域，当写入原点命令时：首先，对象会一直运动直到 ORG 信号发生，然后继续运动，直到 EZ 信号发生。
- STATUS2：如果对象在 ORG 信号区域内，写入原点命令时，对象开始运动。首先，ORG 信号消失，然后下一个 ORG 信号发生。最后，当 EZ 信号发生时运动停止。
- STATUS3：如果对象在 EZ 信号区域内，写入原点命令时，对象开始运动。首先，EZ 信号消失，然后 ORG 信号发生。最后，当 EZ 信号发生时运动停止。

注！ *当 EL 信号发生时，原点模式将停止。*

5. MODE5_Abs_NegRef: ORG+EZ, Move (Dir) ->touch ORG ->Stop ->Move (-Dir) ->touch EZ ->Stop.
   这是一种组合的原点模式。首先，对象将一直运动直到原始信号发生，然后将以与 ORG 相反方向继续运动，直到 EZ 信号发生。
   比如：
   Dir：正。
   ORG 逻辑：高准位。
   EZ 逻辑：高准位。

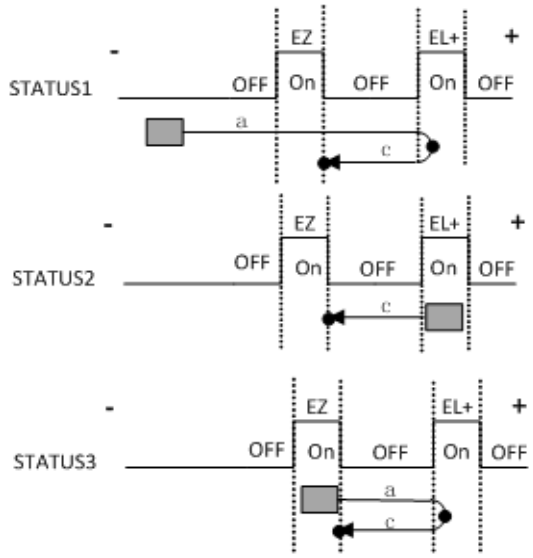


- STATUS1：如果对象超出 EZ 信号和 ORG 信号区域，当写入原点命令时：首先，对象会一直运动直到 ORG 信号发生，然后继续以相反方向运动，直到 EZ 信号发生。
- STATUS2：如果对象在 ORG 信号区域内，写入原点命令时，对象开始运动。首先，ORG 信号消失，然后下一个 ORG 信号发生，同时倒转运动方向。最后，当 EZ 信号发生时运动停止。
- STATUS3：如果对象在 EZ 信号区域内，写入原点命令时，对象开始运动。首先，EZ 信号消失，然后 ORG 信号发生，同时倒转运动方向。最后，当 EZ 信号发生时运动停止。

注！ *当 EL 信号发生时，原点模式将停止。*

6. MODE6_Lmt_Ref: EL + NegEZ, Move (Dir) ->touch EL ->Stop -> Move (-Dir) ->touch EZ ->Stop.
   首先，对象将一直运动直到原始信号发生，然后将以与 ORG 相反方向继续运动，直到 EZ 信号发生。
   比如：
   Dir：正。
   EZ 逻辑：高准位。
   限位逻辑：高准位。

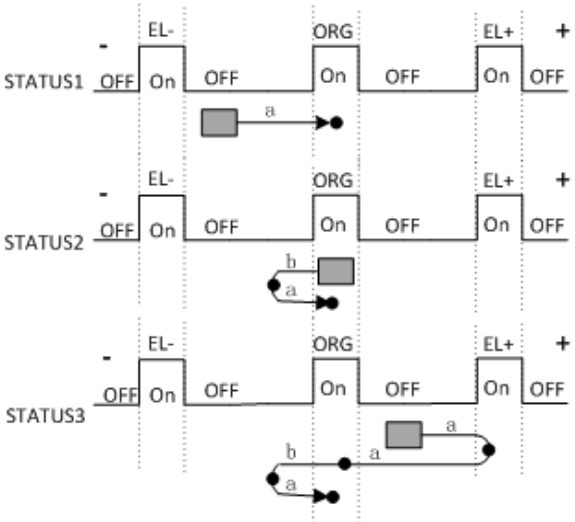
Lmt+EZ



- STATUS1：如果对象超出 EZ 信号和 EL 信号区域，当写入原点命令时：首先，对象会一直运动直到 EL 信号发生，然后继续以相反方向运动，直到 EZ 信号发生。
- STATUS2：如果对象在 EL 信号区域内，对象将一直以相反方向运动，直到 EZ 信号发生。
- STATUS3：如果对象在 EZ 信号区域内，写入原点命令时，对象开始运动。首先，EZ 信号消失，然后 EL 信号发生，同时倒转运动方向。最后，当 EZ 信号发生时运动停止。

7. MODE7_AbsSearch: Move (Dir) ->Search ORG ->Stop.
这是一种搜索 ORG 信号从无到有转换的模式。
比如：
Dir：正。
ORG 逻辑：高准位。
限位逻辑：高准位。

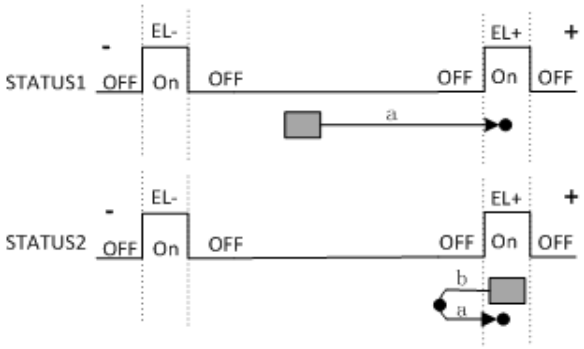
AbsSearch



- STATUS1：如果没有 ORG 信号发生，则 ORG 信号发生时对象将停止运动。
- STATUS2：如果对象在 ORG 信号区域内， 对象以相反方向运动直到信号消失，然后转换方向继续运动，直到 ORG 信号发生。
- STATUS3：如果没有 ORG 信号发生，在运动时 EL 信号首先发生，对象倒转方向并继续运动，然后 ORG 信号将从有到无。然后，再次倒转方向并运动，直到 ORG 信号发生， 运动停止。

8. MODE8_LmtSearch: Move (Dir) ->Search EL ->Stop.
这是一种搜索限位信号从无到有转换的模式。
比如：
Dir：正。
限位逻辑：高准位。

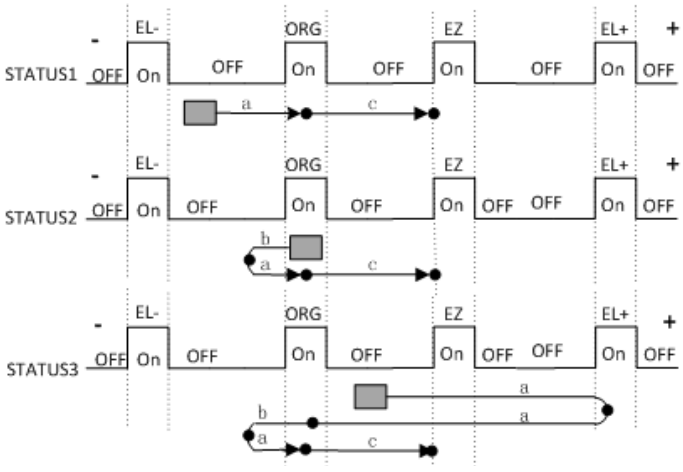
LmtSearch



- STATUS1：如果限位信号在对象运动过程中首先发生，则原点过程终止。
- STATUS2：如果对象在限位信号区域内， 对象以相反方向运动直到信号消失，然后转换方向继续运动，直到限位信号发生。

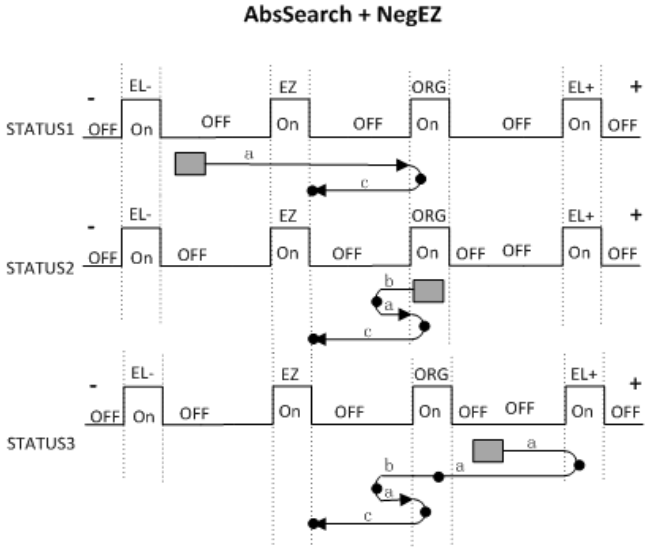
9. MODE9_AbsSearch_Ref: Search ORG + EZ, Move (Dir) ->Search ORG ->Stop ->Move (Dir) ->touch EZ ->Stop.
首先，对象以 MODE7_AbsSearch 方式运动，然后以相同方向运动直到 EZ 信号发生。
比如：
Dir：正。
限位逻辑：高准位。
ORG 逻辑：高准位。

AbsSearch + EZ



- STATUS1：如果对象超出 EZ 信号和 ORG 信号区域，写入原点命令时：首先，对象将一直运动，直到 ORG 信号发生，然后继续运动，直到 EZ 信号发生。
- STATUS2：如果对象在 ORG 信号区域内，写入原点命令时：首先，对象倒转方向并运动，ORG 信号消失，然后再次倒转方向并继续运动，ORG 信号再次发生。 最后，当 EZ 信号发生时运动停止。
- STATUS3：如果没有 ORG 信号发生，EL 信号在 ORG 信号之前发生，当 EL 信号发生时对象倒转方向并继续运动，然后 ORG 信号从有到无。然后再次倒转方向并继续运动，ORG 信号将再次发生并消失。最后，当 EZ 信号发生时运动停止。

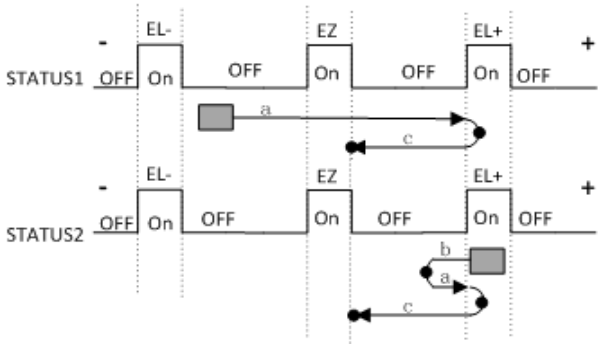
10. MODE10_AbsSearch_NegRef: Search ORG + NegEZ, Move (Dir) ->Search ORG ->Stop ->Move (-Dir) ->touch EZ ->Stop.
首先，对象以 MODE7_AbsSearch 方式运动，然后以相反方向运动直到 EZ 信号发生。
比如：
Dir：正。
限位逻辑：高准位。
ORG 逻辑：高准位。



- STATUS1：如果对象超出 EZ 信号和 ORG 信号区域，当写入原点命令时：首先，对象会一直运动直到 ORG 信号发生，然后倒转方向并继续运动，直到 EZ 信号发生。
- STATUS2：如果对象在 ORG 信号区域内，写入原点命令时：首先，对象倒转方向并运动，ORG 信号消失，然后再次倒转方向并继续运动，ORG 信号再次发生，继续倒转方向并运动。最后，当 EZ 信号发生时运动停止。
- STATUS3：如果没有 ORG 信号发生，EL 信号在 ORG 信号之前发生，当 EL 信号发生时对象倒转方向并继续运动，然后 ORG 信号从有到无。然后再次倒转方向并继续运动，ORG 信号将再次发生，再次倒转方向。最后，当 EZ 信号发生时运动停止。

11. MODE11_LmtSearch_Ref: Search EL +NegEZ, Move (Dir) ->Search EL ->Stop->Move (-Dir) ->touch EZ ->Stop.
首先，对象以 MODE8_LmtSearch 方式运动，然后以相反方向运动直到 EZ 信号发生。
比如：
Dir：正。
限位逻辑：高准位。

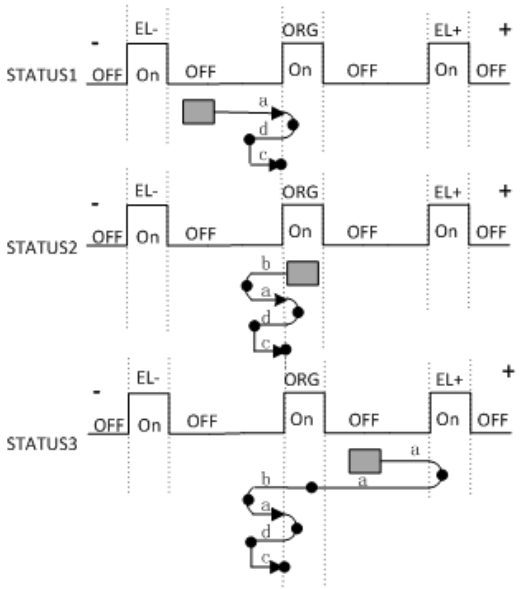
LmtSearch + NegEZ



- STATUS1：当对象不在限位信号区域内，首先，对象会一直运动直到 EL 信号发生，然后倒转方向并继续运动，直到 EZ 信号发生。
- STATUS2：当对象不在限位信号区域内，首先，对象倒转方向并运动，EL 信号消失，然后再次倒转方向并继续运动，EL 信号再次发生，再次倒转方向并运动。最后，当 EZ 信号发生时运动停止。

12. MODE12_ AbsSearchRefind: Search ORG +Refind ORG, Move (Dir) ->Search ORG ->Stop->Move (-Dir) ->Leave ORG(FL) ->Stop-> Move (-Dir)->Refind ORG(FL)->Stop.
首先，轴以 MODE7_AbsSearch 模式运动；然后轴反向以低速 （VelLow）等速运动直至 ORG 信号消失；然后轴再次反向以低速 （VelLow）等速运动直至 ORG 信号发生。
比如：
Dir：正。
ORG 逻辑：高准位。
限位逻辑：高准位。

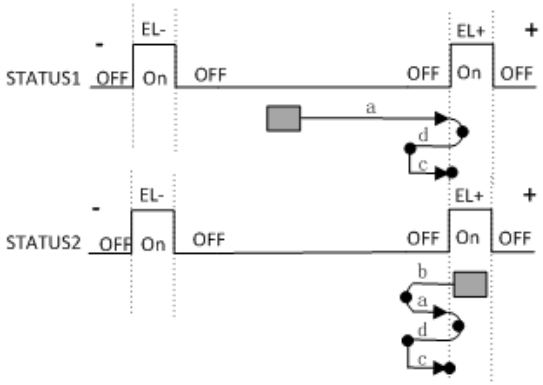
AbsSearchReFind



AbsSearch 过程有三种状况，具体请参考 MODE7_AbsSearch 中的描述。

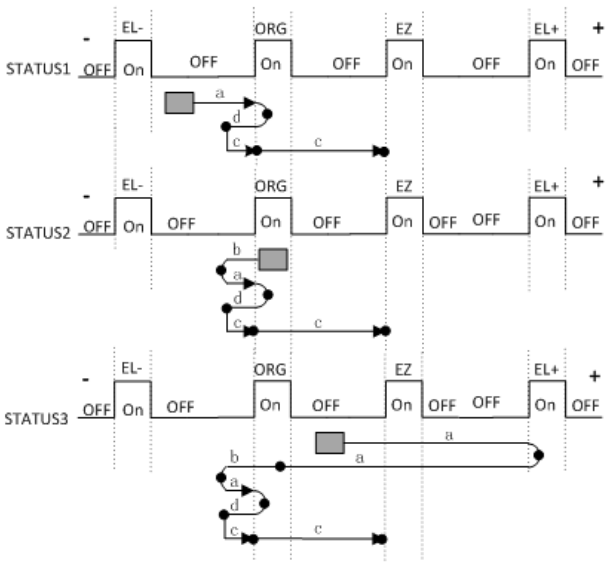
13. MODE13_ LmtSearchRefind: Search EL +Refind EL, Move (Dir) ->Search EL ->Stop->Move (-Dir) ->Leave EL(FL) ->Stop-> Move (-Dir)->Refind EL(FL)->Stop.
首先，轴以 MODE8_LmtSearch 模式运动；然后轴反向以低速 （VelLow）等速运动直至 EL 信号消失；然后轴再次反向以低速 （VelLow）等速运动直至 EL 信号发生。
比如：
Dir：正。
限位逻辑：高准位。

LmtSearchReFind



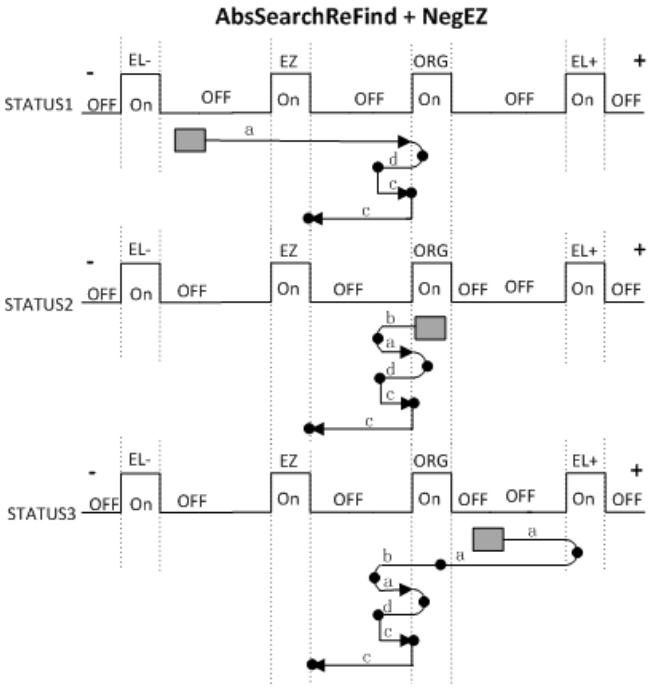
14. MODE14_AbsSearchRefind_Ref: Search ORG +Refind ORG+EZ, Move (Dir) ->Search ORG ->Stop->Move (-Dir) ->Leave ORG(FL) ->Stop-> Move (-Dir)->Refind ORG(FL)->Stop->Move (Dir) ->touch EZ ->Stop.
首先，轴以 MODE7_AbsSearch 模式运动；然后轴反向以低速 （VelLow）等速运动直至 ORG 信号消失；然后轴再次反向以低速 （VelLow）等速运动直至 ORG 信号发生；最后轴沿相同方向運動至找到 Z 相。
比如：
Dir：正。
限位逻辑：高准位。
ORG 逻辑：高准位。

AbsSearchReFind + EZ



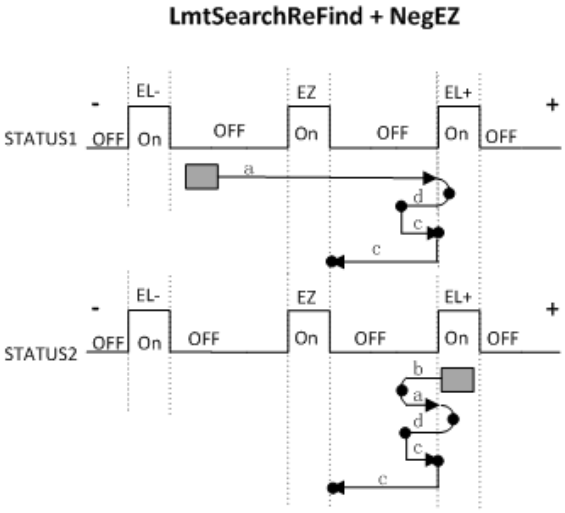
AbsSearch 过程有三种状况，具体请参考 MODE7_AbsSearch 中的描述。

15. MODE15_AbsSearchRefind_NegRef: Search ORG +Refind ORG+NegEZ, Move (Dir) ->Search ORG ->Stop->Move (-Dir) ->Leave ORG (FL)->Stop-> Move (-Dir)->Refind ORG(FL)-> Stop-> Move (-Dir) ->touch EZ ->Stop.
首先，轴以 MODE7_AbsSearch 模式运动；然后轴反向以低速 （VelLow）等速运动直至 ORG 信号消失；然后轴再次反向以低速 （VelLow）等速运动直至 ORG 信号发生；最后轴再次反向运动直至 EZ 信号发生。
比如：
Dir：正。
限位逻辑：高准位。
ORG 逻辑：高准位。



AbsSearch 过程有三种状况，具体请参考 MODE7_AbsSearch 中的描述。

16. MODE16_LmtSearchRefind_Ref: Search EL +Refind EL+EZ, Move (Dir) ->Search EL ->Stop->Move (-Dir) ->Leave EL(FL) ->Stop-> Move (-Dir)->Refind EL(FL)->Stop->Move (-Dir) ->touch EZ ->Stop.
首先，轴以 MODE8_LmtSearch 模式运动；然后轴反向以低速 （VelLow）等速运动直至 EL 信号消失；然后轴再次反向以低速 （VelLow）等速运动直至 EL 信号发生，最后轴再次反向运动直至 EZ 信号发生。
比如：
Dir：正。
限位逻辑：高准位。



LmtSearch 过程有三种状况，具体请参考 MODE8_LmtSearch 中的描述。

6.3.3.8 位置 / 计数器控制

6.3.3.8.1 Acm_AxSetCmdPosition

格式：

U32 Acm_AxSetCmdPosition (HAND AxisHandle, F64 Position)

目的：

设置指定轴的理论位置。

参数：

名称	类型	IN 或 OUT	说明
AxisHandle	HAND	IN	来自 Acm_AxOpen 的轴句柄。
Position	F64	IN	新的理论位置。（单位：PPU）

返回值：

错误代码

注解：

6.3.3.8.2 Acm_AxGetCmdPosition

格式：

U32 Acm_AxGetCmdPosition (HAND AxisHandle, PF64 Position)

目的：

获取指定轴的当前理论位置。

参数：

名称	类型	IN 或 OUT	说明
AxisHandle	HAND	IN	来自 Acm_AxOpen 的轴句柄。
Position	PF64	OUT	返回理论位置。（单位：PPU）

返回值：

错误代码

注解：

6.3.3.8.3 Acm_AxSetActualPosition

格式：

U32 Acm_AxSetActualPosition (HAND AxisHandle, F64 Position)

目的：

设置指定轴的实际位置。

参数：

名称	类型	IN 或 OUT	说明
AxisHandle	HAND	IN	来自 Acm_AxOpen 的轴句柄。
Position	F64	IN	新的实际位置。（单位：PPU）

返回值：

错误代码

注解：

6.3.3.8.4 Acm_AxGetActualPosition

格式：

U32 Acm_AxGetActualPosition (HAND AxisHandle, PF64 Position)

目的：

获取指定轴的当前实际位置。

参数：

名称	类型	IN 或 OUT	说明
AxisHandle	HAND	IN	来自 Acm_AxOpen 的轴句柄。
Position	PF64	IN	返回实际位置。（单位：PPU）

返回值：

错误代码

注解：

6.3.3.9 Aux/Gen 输出

6.3.3.9.1 Acm_AxDoSetBit

格式：

Acm_AxDoSetBit (HAND AxisHandle, U16 DoChannel, U8 BitData)

目的：

设定指定通道的 DO 值。

参数：

名称	类型	IN 或 OUT	说明
AxisHandle	HAND	IN	来自 Acm_AxOpen 的轴句柄。
DoChannel	U16	IN	数字量输出通道 （4 ~ 7）。
BitData	U8	IN	DO 值： 0 或 1

返回值：

错误代码

注解：

如果用户想要使用该共用 DO 功能，必须首先将 CFG_AxGenDoEnable 设置为 GEN_DO_EN。 启用 CFG_AxGenDoEnable 时，ERC 功能将被禁用。两种功能使用同一输出针脚 （OUT7）。

6.3.3.9.2 Acm_AxDoGetBit

格式：

U32 Acm_AxDoGetBit (HAND AxisHandle, U16 DoChannel, PU8 BitData)

目的：

获取指定通道的数字量输出 bit 值。

参数：

名称	类型	IN 或 OUT	说明
AxisHandle	HAND	IN	来自 Acm_AxOpen 的轴句柄。
DoChannel	U16	IN	数字量输出通道 （4 ~ 7）。
BitData	PU8	OUT	DO 值：0 或 1。

返回值：

错误代码

注解：

请参考 Acm_AxDoSetBit。

6.3.3.9.3 Acm_AxDiGetBit

格式：

U32 Acm_AxDiGetBit (HAND AxisHandle, U16 DiChannel, PU8 BitData)

目的：

获取指定通道的 DI 值。

参数：

名称	类型	IN 或 OUT	说明
AxisHandle	HAND	IN	来自 Acm_AxOpen 的轴句柄。
DiChannel	U16	IN	数字量输入通道。（0 ~ 3）
BitData	PU8	OUT	DI 值：0 或 1。

返回值：

错误代码

注解：

6.3.3.10 外部驱动

6.3.3.10.1 Acm_AxSetExtDrive

格式：

U32 Acm_AxSetExtDrive (HAND AxisHandle, U16 ExtDrvMode)

目的：

启用或禁用外部驱动模式。

参数：

名称	类型	IN 或 OUT	说明
AxisHandle	HAND	IN	来自 Acm_AxOpen 的轴句柄。
ExtDrvMode	U16	IN	0：禁用 （停止命令） 。
			1：JOG 模式。
			2：MPG 模式。
			3：JOG 步进模式 （预留）。

返回值：

错误代码

注解：

6.3.3.11 停止运动

6.3.3.11.1 Acm_AxStopDec

格式：

U32 Acm_AxStopDec (HAND AxisHandle)

目的：

命令轴减速停止。

参数：

名称	类型	IN 或 OUT	说明
AxisHandle	HAND	IN	来自 Acm_AxOpen 的轴句柄。

返回值：

错误代码

注解：

如果轴处于同步驱动模式，如 E-gear 运动中的从轴，该 API 能够用于停止同步关系。

6.3.3.11.2 Acm_AxStopEmg

格式：

U32 Acm_AxStopEmg (HAND AxisHandle)

目的：

命令轴立刻停止 （无减速）。

参数：

名称	类型	IN 或 OUT	说明
AxisHandle	HAND	IN	来自 Acm_AxOpen 的轴句柄。

返回值：

错误代码

注解：

如果轴处于同步驱动模式，如 E-gear 运动中的从轴，该 API 能够用于停止同步关系。

6.3.3.11.3 Acm_AxStopDecEx

格式：

U32 Acm_AxStopDecEx (HAND AxisHandle, F64 NewDec)

目的：

下达停止命令时可指定减速度。

参数：

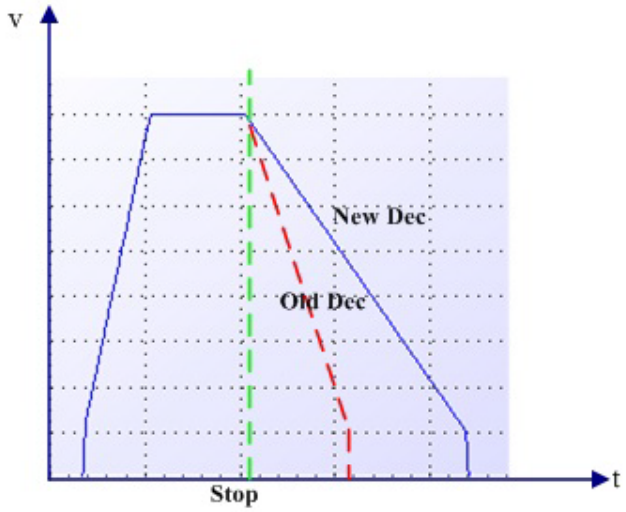
名称	类型	IN 或 OUT	说明
AxisHandle	HAND	IN	来自 Acm_AxOpen 的轴句柄。
NewDec	F64	IN	减速停止时的减速度，单位：PPU/s^2

返回值：

错误代码

注解：

如果减速停止命令下达之后，剩余的脉冲量不足以支持指定的 NewDec，则会发生断尾现象。



6.3.4 群组

6.3.4.1 系统

6.3.4.1.1 Acm_GpAddAxis

格式：

U32 Acm_GpAddAxis (PHAND GpHandle, HAND AxHandle)

目的：

添加一个轴到指定群组。

参数：

名称	类型	IN 或 OUT	说明
GpHandle	PHAND	IN/OUT	指针指向群组句柄 （Null 或无）。
AxHandle	HAND	IN	来自 Acm_AxOpen 的轴句柄。

返回值：

错误代码

注解：

如果 **GpHandle** 指向 NULL，驱动会创建一个新的群组并将轴添加至该群组。 如果 **GpHandle** 指向一个有效群组句柄，驱动将只把群组添加到群组中。

PCI-1245L 最多有 1 个群组。

群组中的主轴为具有最小 **PhysicalID** 的轴。

添加第一个轴时，群组的参数会初始化， 如 CFG_GpPPU、PAR_GpVelLow、PAR_GpVelHigh、PAR_GpAcc、PAR_GPDec 和 PAR_GpJerk。

6.3.4.1.2 Acm_GpRemAxis

格式：

U32 Acm_GpRemAxis (HAND GpHandle, HAND AxHandle)

目的：

从指定群组中移除一个轴。

参数：

名称	类型	IN 或 OUT	说明
GpHandle	HAND	IN	来自 Acm_GpAddaxis 的群组句柄。
AxHandle	HAND	IN	来自 Acm_AxOpen 的轴句柄。

返回值：

错误代码

注解：

调用 **Acm_GpRemAxis** 之后，群组中没有轴，**GpHandle** 仍可使用。 用户可使用该群组句柄添加其它轴。 但是，如果用户调用 Acm_GpClose 关闭该群组句柄，群组句柄不能再次使用。

6.3.4.1.3 Acm_GpClose

格式：

U32 Acm_GpClose (PHAND pGroupHandle)

目的：

移除群组中的所有轴并关闭群组句柄。

参数：

名称	类型	IN 或 OUT	说明

GpHandle	PHAND	IN	指针指向要关闭的群组句柄。

返回值：

错误代码

注解：

如果群组数量大于设备的最大群组数，则不能创建新的群组。这时，如果用户想要创建新的群组，则必须关闭一个现有群组。

6.3.4.1.4 Acm_GpResetError

格式：

U32 Acm_GpResetError (HAND GroupHandle)

目的：

复位群组状态。

参数：

名称	类型	IN 或 OUT	说明
GpHandle	HAND	IN	来自 Acm_GpAddAxis 的群组句柄。

返回值：

错误代码

注解：

如果群组处于 STA_GP_ERROR_STOP 状态，那么调用该函数后，状态将变为 STA_GP_READY。

6.3.4.2 运动状态及速度

6.3.4.2.1 Acm_GpGetState

格式：

U32 Acm_GpGetState (HAND GroupHandle, PU16 pState)

目的：

获取群组的当前状态。

参数：

名称	类型	IN 或 OUT	说明
GpHandle	HAND	IN	来自 Acm_GpAddAxis 的群组句柄。
pState	PU16	OUT	群组状态： 0：STA_GP_DISABLE 1：STA_GP_READY 2：STA_GP_STOPPING 3：STA_GP_ERROR_STOP 4：STA_GP_MOTION 5：STA_GP_AX_MOTION （不支持） 6：STA_GP_MOTION_PATH

返回值：

错误代码

注解：

如果群组的一个轴正在执行单轴运动命令，群组状态将不会改变。

6.3.4.2.2 Acm_GpGetCmdVel

格式：

U32 Acm_GpGetCmdVel(HAND GroupHandle, PF64 CmdVel)

目的：

获取群组的当前的速度值。

参数：

名称	类型	IN 或 OUT	说明
GroupHandle	HAND	IN	来自 Acm_GpAddAxis 的群组句柄。
CmdVel	PF64	OUT	返回群组的当前速度值，单位：PPU/s。(PPU 为轴 ID 最小的轴的 PPU)

返回值：

错误代码

注解：

通过 API，可以获得群组在执行插补或连续插补动作时当前的速度值。

6.3.4.3 运动停止

6.3.4.3.1 Acm_GpStopDec

格式：

U32 Acm_GpStopDec (HAND GroupHandle)

目的：

命令该群组中的轴减速停止。

参数：

名称	类型	IN 或 OUT	说明
GpHandle	HAND	IN	来自 Acm_GpAddAxis 的群组句柄。

返回值：

错误代码

注解：

6.3.4.3.2 Acm_GpStopEmg

格式：

U32 Acm_GpStopEmg(HAND GroupHandle)

目的：

命令该群组中的轴立刻停止 （无减速）。

参数：

名称	类型	IN 或 OUT	说明
GpHandle	HAND	IN	来自 Acm_GpAddAxis 的群组句柄。

返回值：

错误代码

注解：

6.3.4.4 插补运动

6.3.4.4.1 Acm_GpMoveLinearRel

格式：

U32 Acm_GpMoveLinearRel(HAND GroupHandle, PF64 DistanceArray, PU32 pArrayElements)

目的：

命令群组执行相对线性插补。

参数：

名称	类型	IN 或 OUT	说明
GpHandle	HAND	IN	来自 Acm_GpAddAxis 的群组句柄。
DistanceArray	PF64	IN	群组中轴的距离阵列，阵列元素的每个值都表示轴的相对位置。
pArrayElements	PU32	IN/OUT	阵列中的元素个数 （该个数必须等于该群组中的轴个数，否则将返回群组中的轴个数）。

返回值：

错误代码

注解：

DistanceArray 中的数据顺序必须遵循 X 轴、Y 轴、Z 轴和 U 轴的顺序。比如，如果一个群组有两个轴：Y 轴和 U 轴。**DistanceArray** 中的第一个数据表示 Y 轴的相对距离，第二个数据表示 U 轴的相对距离。**DistanceArray** 中距离的单位为群组中每个轴的 PPU。

线性插补和直接插补的不同之处在于：线性插补的速度被分解为其中各个轴的向量速度，轴将以该速度运动。多数情况中，线性插补应用于以直角组合的轴。 而直接插补的线性速度设置为主轴 （移动距离最长的轴）的速度，其他轴会与主轴同时启动同时停止。多数情况中，直接插补应用于以斜角 组 合 的轴。

PCI-1245L 只支持 2 个轴线性插补。

6.3.4.4.2 Acm_GpMoveLinearAbs

格式：

U32 Acm_GpMoveLinearAbs (HAND GroupHandle, PF64 PositionArray, PU32 pArrayElements)

目的：

命令群组执行绝对线性插补。

参数：

名称	类型	IN 或 OUT	说明
GpHandle	HAND	IN	来自 Acm_GpAddAxis 的群组句柄。
PositionArray	PF64	IN	群组中轴的位置阵列，阵列元素的每个值都表示轴的绝对位置。
pArrayElements	PU32	IN/OUT	阵列中的元素个数 （该个数必须等于该群组中的轴个数，否则将返回群组中的轴个数）。

返回值：

错误代码

注解：

PositionArray 中的数据顺序必须遵循 X 轴、Y 轴、Z 轴和 U 轴的顺序。比如，如果一个群组有两个轴：Y 轴和 U 轴。**PositionArray** 中的第一个数据表示 Y 轴的绝对距离，第二个数据表示 U 轴的绝对距离。**PositionArray** 中距离的单位为群组中每个轴的 PPU。

线性插补和直接插补的不同之处在于：线性插补的速度被分解为其中各个轴的向量速度，轴将以该速度运动。多数情况中，线性插补应用于以直角组合的轴。 而直接插补的线性速度设置为主轴 （移动距离最长的轴）的速度，其他轴会与主轴同时启动同时停止。多数情况中，直接插补应用于以斜角 组 合 的轴。

PCI-1245L 只支持 2 个轴线性插补。

6.3.4.4.3 Acm_GpMoveDirectAbs

格式：

U32 Acm_GpMoveDirectAbs (HAND GroupHandle, PF64 PositionArray, PU32 ArrayElements)

目的：

命令群组执行绝对直接线性插补。

参数：

名称	类型	IN 或 OUT	说明
GpHandle	HAND	IN	来自 Acm_GpAddAxis 的群组句柄。
PositionArray	PF64	IN	群组中轴的距离阵列，阵列元素的每个值都表示轴的绝对位置。
pArrayElements	PU32	IN/OUT	阵列中的元素个数 （该个数必须等于该群组中的轴个数，否则将返回群组中的轴个数）。

返回值：

错误代码

注解：

PositionArray 中的数据顺序必须遵循 X 轴、Y 轴、Z 轴和 U 轴等的顺序。比如，如果一个群组有两个轴： Y 轴和 U 轴。**PositionArray** 中的第一个数据表示 Y 轴的绝对距离，第二个数据表示 U 轴的绝对距离。**PositionArray** 中距离的单位为群组中每个轴的 PPU。

线性插补和直接插补的不同之处在于：线性插补的速度被分解为其中各个轴的向量速度，轴将以该速度运动。 多数情况中，线性插补应用于以直角组合的轴。而直接插补的线性速度设置为主轴 （移动距离最长的轴）的速度，其 他轴与主轴同时启动，同时停止。多数情况中，直接插补应用于以斜角 组 合 的轴。

PCI-1245L 仅支持 2 个轴直接插补。

6.3.4.4.4 Acm_GpMoveDirectRel

格式：

U32 Acm_GpMoveDirectRel (HAND GroupHandle, PF64 DistanceArray, PU32 ArrayElements)

目的：

命令群组执行相对直接线性插补。

参数：

名称	类型	IN 或 OUT	说明
GpHandle	HAND	IN	来自 Acm_GpAddAxis 的群组句柄。
DistanceArray	PF64	IN	群组中轴的距离阵列，阵列元素的每个值都表示轴的相对位置。
ArrayElements	PU32	IN/OUT	阵列中的元素个数 （该个数必须等于该群组中的轴个数，否则将返回群组中的轴个数）。

返回值：

错误代码

注解：

DistanceArray 中的数据顺序必须遵循 X 轴、Y 轴、Z 轴和 U 轴等的顺序。比如，如果一个群组有两个轴： Y 轴和 U 轴。**DistanceArray** 中的第一个数据表示 Y 轴的相对距离，第二个数据表示 U 轴的相对距离。**DistanceArray** 中距离的单位为群组中每个轴的 PPU。

线性插补和直接插补的不同之处在于：线性插补的速度被分解为其中各个轴的向量速度，轴将以该速度运动。 多数情况中，线性插补应用于以直角组合的轴。而直接插补的线性速度设置为主轴 （移动距离最长的轴）的速度，其 他轴与主轴同时启动，同时停止。多数情况中，直接插补应用于以斜角 组 合 的轴。

PCI-1245L 仅支持 2 个轴直接插补。

6.4 属性列表

6.4.1 设备

6.4.1.1 特性

6.4.1.1.1 FT_DevIpoTypeMap

数据类型：

U32

R/W：

R

属性 ID：

0

含义：

获取设备支持的插补类型。1：支持； 0：不支持。

位	说明
0	线性插补，2 轴
1	线性插补，3 轴
2	线性插补，4 轴
3	线性插补，5 轴
4	线性插补，6 轴
5 ~ 7	未定义
8	圆弧插补，2 轴
9	圆弧插补，3 轴
10	螺旋
11 ~ 15	未定义
16	同步电子齿轮
17	同步电子凸轮
18	龙门控制
19	切线跟随
20 ~ 23	未定义
24	选择路径
25 ~ 31	未定义

注解：

6.4.1.1.2 FT_DevAxisCount

数据类型：

U32

R/W：

R

属性 ID：

1

含义：

获取该设备的轴个数。

注解：

6.4.1.1.3 FT_DevFunctionMap

数据类型：

U32

R/W：

R

属性 ID：

2

含义：

获取设备支持的功能。1：支持； 0：不支持。

位	说明
0	运动
1	DI （PCI-1245L 不支持）
2	DO （PCI-1245L 不支持）
3	AI （PCI-1245L 不支持）
4	AO （PCI-1245L 不支持）
5	定时器
6	计数器
7	DAQ DI （PCI-1245L 不支持）
8	DAQ DO （PCI-1245L 不支持）
9	DAQ AI （PCI-1245L 不支持）
10	DAQ AO （PCI-1245L 不支持）
11	Emg

注解：

6.4.1.1.4 FT_DevOverflowCntr

数据类型：

U32

R/W：

R

属性 ID：

3

含义：

位置计数器的最大值。

注解：

在 PCI-1245L 中，值为 2147483647。

6.4.1.2 配置

6.4.1.2.1 CFG_DevBoardID

数据类型：

U32

R/W：

R

属性 ID：

201

含义：

获取设备 ID。对于 PCI-1245L，该属性值将为 0 ~ 15。

注解：

6.4.1.2.2 CFG_DevBaseAddress

数据类型：

U32

R/W：

R

属性 ID：

203

含义：

返回 IO 基地址。

注解：

6.4.1.2.3 CFG_DevInterrupt

数据类型：

U32

R/W：

R

属性 ID：

204

含义：

获取设备中断编号。

注解：

6.4.1.2.4 CFG_DevBusNumber

数据类型：

U32

R/W：

R

属性 ID：

205

含义：

获取设备总线编号。

注解：

6.4.1.2.5 CFG_DevSlotNumber

数据类型：

U32

R/W：

R

属性 ID：

206

含义：

获取设备插槽编号。

注解：

6.4.1.2.6 CFG_DevDriverVersion

数据类型：

char*

R/W：

R

属性 ID：

207

含义：

获取 SYS 驱动版本。格式为：1.0.0.1。

注解：

6.4.1.2.7 CFG_DevDllVersion

数据类型：

char*

R/W：

R

属性 ID：

208

含义：

获取 DLL 驱动版本。格式为：1.0.0.1.。

注解：

6.4.1.2.8 CFG_DevFwVersion

数据类型：

char*

R/W：

R

属性 ID：

208

含义：

获取固件版本。格式为：1.0.0.1.

注解：

6.4.1.2.9 CFG_DevFPGA_1Version

数据类型：

char*

R/W：

R

属性 ID：

218

含义：

获取设备的 CPLD 版本。格式为：1.0.0.1.。

注解：

6.4.1.2.10 CFG_DevEmgLogic

数据类型：

U32

R/W：

RW

属性 ID：

220

含义：

设定紧急停止信号的逻辑准位。

位	说明
0	低准位
1	高准位

6.4.2 轴

6.4.2.1 特性

6.4.2.1.1 系统

6.4.2.1.1.1 FT_AxFunctionMap

数据类型：

U32

R/W：

R

属性 ID：

301

含义：

获取轴支持的功能。1：支持； 0：不支持。

位	说明
0	到位
1	报警
2	清除伺服驱动中的偏转计数器
3	减速
4	硬件限位开关
5	软件限位开关
6	原点传感器
7	编码 Z 相位传感器
8	背隙校正
9	抑制振动
10	返回原点
11	叠加
12	比较
13	锁存
14	CAMDO
15	外部驱动
16	同步启停
17 ~ 31	未定义

注解：

6.4.2.1.2 速度模式

6.4.2.1.2.1 FT_AxMaxVel

数据类型：

F64

R/W：

R

属性 ID：

302

含义：

获取轴支持的最大速度。（单位：脉冲 /s）

注解：

对于 PCI-1245L，该值为 1,000,000。

6.4.2.1.2.2 FT_AxMaxAcc

数据类型：

F64

R/W：

R

属性 ID：

303

含义：

获取轴支持的最大加速度。（单位：脉冲 /s^2）

注解：

对于 PCI-1245L，该值为 100,000,000。

6.4.2.1.2.3 FT_AxMaxDec

数据类型：

F64

R/W：

R

属性 ID：

304

含义：

获取轴支持的最大减速度。（单位： 脉冲 /s^2）

注解：

对于 PCI-1245L，该值为 100,000,000。

6.4.2.1.2.4 FT_AxMaxJerk

数据类型：

F64

R/W：

R

属性 ID：

305

含义：

获取轴支持的最大加加速度。（单位：脉冲 /S^3）

注解：

对于 PCI-1245L，该值为 1。

6.4.2.1.3 脉冲输入

6.4.2.1.3.1 FT_AxPulseInMap

数据类型：

U32

R/W：

R

属性 ID：

306

含义：

获取该运动设备支持的脉冲输入特性。

位	说明
0	模式
1	逻辑
2	源
3 ~ 31	未定义

注解：

6.4.2.1.3.2 FT_AxPulseInModeMap

数据类型：

U32

R/W：

R

属性 ID：

307

含义：

获取轴支持的脉冲输入模式。

位	说明
0	1X A/B
1	2X A/B
2	4X A/B
3	CW/CCW
4 ~ 31	未定义

注解：

6.4.2.1.4 脉冲输出

6.4.2.1.4.1 FT_AxPulseOutMap

数据类型：

U32

R/W：

R

属性 ID：

308

含义：

获取该运动设备支持的脉冲输出特性。

位	说明
0	模式
1 ~ 31	未定义

注解：

对于 PCI-1245L，该值为 1。

6.4.2.1.4.2 FT_AxPulseOutModeMap

数据类型：

U32

R/W：

R

属性 ID：

309

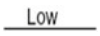
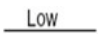
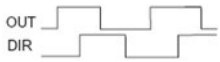
含义：

获取该运动设备支持的脉冲输出模式。

位	说明
0	OUT/DIR
1	OUT/DIR，OUT 负逻辑
2	OUT/DIR，DIR 负逻辑
3	OUT/DIR，OUT&DIR 负逻辑
4	CW/CCW
5	CW/CCW，CW&CCW 负逻辑
6	A/B 相位
7	B/A 相位
8	CW/CCW，OUT 负逻辑 （不支持）
9	CW/CCW，DIR 负逻辑 （不支持）
10 ~ 31	未定义

注解：

在 PCI-1245L 中，值为 63。

位	说明	正方向		负方向	
		OUT 输出	DIR 输出	OUT 输出	DIR 输出
0	OUT/DIR				
1	OUT/DIR， OUT 负逻辑				
2	OUT/DIR， DIR 负逻辑				
3	OUT/DIR， OUT&DIR 负逻辑				
4	CW/CCW				
5	CW/CCW， CW&CCW 负逻辑				
6	A/B 相位				
7	B/A 相位				

6.4.2.1.5 报警

6.4.2.1.5.1 FT_AxAlmMap

数据类型：

U32

R/W：

R

属性 ID：

310

含义：

获取该运动轴支持的报警特性。

位	说明
0	启用
1	逻辑
2	反应
3 ~ 31	未定义

注解：

6.4.2.1.6 到位

6.4.2.1.6.1 FT_AxInpMap

数据类型：

U32

R/W：

R

属性 ID：

311

含义：

获取该运动轴支持的到位特性。

位	说明
0	模式
1	逻辑
2 ~ 31	未定义

注解：

6.4.2.1.7 ERC

6.4.2.1.7.1 FT_AxErcMap

数据类型：

U32

R/W：

R

属性 ID：

312

含义：

获取该运动轴支持的 ERC 特性。

位	说明
0	启用模式
1	逻辑
2	启动时间 （不支持）
3	关闭时间 （不支持）
4 ~ 31	未定义

注解：

6.4.2.1.7.2 FT_AxErcEnableModeMap

数据类型：

U32

R/W：

R

属性 ID：

313

含义：

获取轴支持的 ERC 模式。.

位	说明
0	返回原点后 ERC 输出
1	EMG/ALM/EL 激活时 ERC 输出
2	返回原点后或 EMG/ALM/EL 激活时 ERC 输出
3 ~ 31	未定义

注解：

6.4.2.1.8 SD

6.4.2.1.8.1 FT_AxSdMap

数据类型：

U32

R/W：

R

属性 ID：

316

含义：

获取该运动轴支持的减速 （SD）特性。

位	说明
0	启用
1	逻辑
2	反应
3 ~ 31	未定义

注解：

对于 PCI-1245L，该值为 0。

6.4.2.1.9 硬件限位

6.4.2.1.9.1 FT_AxElMap

数据类型：

U32

R/W：

R

属性 ID：

317

含义：

获取该运动轴支持的硬件终端限位 （EL）特性。.

位	说明
0	启用
1	逻辑
2	反应
3 ~ 31	未定义

注解：

6.4.2.1.10 软件限位

6.4.2.1.10.1 FT_AxSwMelMap

数据类型：

U32

R/W：

R

属性 ID：

318

含义：

获取运动轴支持的软件负方向限位特性。

位	说明
0	启用
1	反应
2	值
3 ~ 31	未定义

注解：

6.4.2.1.10.2 FT_AxSwPelMap

数据类型：

U32

R/W：

R

属性 ID：

319

含义：

获取运动轴支持的软件正方向限位特性。

位	说明
0	启用
1	反应
2	值
3 ~ 31	未定义

注解：

6.4.2.1.11 原点

6.4.2.1.11.1 FT_AxHomeMap

数据类型：

U32

R/W：

R

属性 ID:

320

含义：

获取轴所支持的原点相关特性。

位	描述
0	原点模式
1	ORG 逻辑
2	EZ 逻辑
3	启用复位

注解：

6.4.2.1.11.2 FT_AxHomeModeMap

数据类型：

U32

R/W：

R

属性 ID：

332

含义：

所支援的回原点方式。

位	说明
0	MP_MODE1_Abs
1	MP_MODE2_Lmt
2	MP_MODE3_Ref
3	MP_MODE4_Abs_Ref
4	MP_MODE5_Abs_NegRef
5	MP_MODE6_Lmt_Ref
6	MP_MODE7_AbsSearch
7	MP_MODE8_LmtSearch
8	MP_MODE9_AbsSearch_Ref
9	MP_MODE10_AbsSearch_NegRef
10	MP_MODE11_LmtSearch_Ref
11	MP_MODE12_AbsSearchReFind
12	MP_MODE13_LmtSearchReFind
13	MP_MODE14_AbsSearchReFind_Ref
14	MP_MODE15_AbsSearchReFind_NegRef
15	MP_MODE16_LmtSearchReFind_Ref

注解：

有关每种方式详细信息，请参考 Acm_AxHome 介绍。

6.4.2.1.12 背隙补偿

6.4.2.1.12.1 FT_AxBackLashMap

数据类型：

U32

R/W：

R

属性 ID：

321

含义：

获取该运动轴支持的背隙补偿特性。

位	说明
0	启用
1	值
2 ~ 31	未定义

注解：

6.4.2.1.13 外部驱动

6.4.2.1.13.1 FT_AxExtDriveMap

数据类型：

U32

R/W：

R

属性 ID：

327

含义：

获取轴支持的外部驱动特性。

位	说明
0	ExtMasterSrc
1	ExtSelEnable
2	ExtPulseNum
3	ExtPulseMode
4	ExtPresetNum
5 ~ 31	未定义

注解：

默认值为 29。

6.4.2.1.13.2 FT_AxExtMasterSrcMap

数据类型：

U32

R/W：

R

属性 ID：

328

含义：

获取轴支持的外部驱动源。

位	说明
0	轴 0
1	轴 1
2	轴 2
3	轴 3
4 ~ 31	未定义

注解：

6.4.2.1.14 Aux/Gen 输出

6.4.2.1.14.1 FT_AxGenDOMap

数据类型：

U32

R/W：

R

属性 ID：

329

含义：

获取轴支持的通用输出，OUT4 ~ OUT7。

位	说明
0	OUT4/CAM_DO
1	OUT5/TRIG_Position
2	OUT6/SVON
3	OUT7/ERC
4 ~ 31	未定义

注解：

6.4.2.1.14.2 FT_AxGenDIMap

数据类型：

U32

R/W：

R

属性 ID：

330

含义：

获取轴支持的通用输入，IN1 ~ IN5。

位	说明
0	IN1
1	IN2/RDY
2	IN4/JOG+
3	IN5/JOG-
4 ~ 31	未定义

注解：

6.4.2.1.15 同步起停

6.4.2.1.15.1 FT_AxSimStartSourceMap

数据类型：

U32

R/W：

R

属性 ID：

331

含义：

轴支持的同步起停模式。

位	说明
0	从设备 STA 针脚的信号上启动同步运动模式。（默认）
1~7	未定义。
8	从轴 _0 的比较信号启动同步运动。
9	从轴 _1 的比较信号启动同步运动。
10	从轴 _2 的比较信号启动同步运动。
11	从轴 _3 的比较信号启动同步运动。

14 ~ 15	未定义。
16	当轴 _0 停止时启动同步运动。
17	当轴 _1 停止时启动同步运动。
18	当轴 _2 停止时启动同步运动。
19	当轴 _3 停止时启动同步运动。
22 ~ 31	未定义。

注解：

获取轴支持的同时开始模式。 请参考 CFG_AxSimStartSource。
在 PCI-1245L 中，默认值为 986881。

6.4.2.1.16 触发停止

6.4.2.1.16.1 FT_AxIN1Map

数据类型：

U32

R/W：

R

属性 ID：

333

含义：

IN1 触发停止功能特性。

位	说明
0	启用
1	逻辑
2	反应

注解：

6.4.2.1.16.2 FT_AxIN2Map

数据类型：

U32

R/W：

R

属性 ID：

334

含义：

IN2 触发停止功能特性。

位	说明
0	启用
1	逻辑
2	反应

注解：

6.4.2.1.16.3 FT_AxIN4Map

数据类型：

U32

R/W：

R

属性 ID：

336

含义：

IN4 触发停止功能特性。

位	说明
0	启用
1	逻辑
2	反应

注解：

6.4.2.1.16.4 FT_AxIN5Map

数据类型：

U32

R/W：

R

属性 ID：

337

含义：

IN5 触发停止功能特性。

位	说明
0	启用
1	逻辑
2	反应

注解：

6.4.2.2 配置

6.4.2.2.1 系统

6.4.2.2.1.1 CFG_AxPPU

数据类型：

U32

R/W：

RW

属性 ID：

551

含义：

Pulse per uint （PPU），一个虚拟单位。

该属性值必须大于 0。

该属性值的变化将影响 CFG_AxMaxVel、CFG_AxMaxAcc、CFG_AxMaxDec、PAR_AxVelHigh、PAR_AxVelLow、PAR_AxAcc、PAR_AxDec、PAR_GpVelHigh、PAR_GpVelLow、PAR_GpAcc、PAR_GpDec 和 PAR_HomeCrossDistance。

注解：

默认值为 1。

6.4.2.2.1.2 CFG_AxPhyID

数据类型：

U32

R/W：

R

属性 ID：

552

含义：

获取轴的物理 ID。

值	含义
0	0- 轴
1	1- 轴
2	2- 轴
3	3- 轴
4	4- 轴
5	5- 轴

注解：

6.4.2.2.2 速度模式

6.4.2.2.2.1 CFG_AxMaxVel

数据类型：

F64

R/W：

RW

属性 ID：

553

含义：

配置运动轴的最大速度 （单位：PPU/s）。

注解：

该属性的最大值 = FT_AxMaxVel / CFG_AxPPU，最小值 = 1 / CFG_AxPPU。
对于 PCI-1245L，该默认值为 1,000,000。

6.4.2.2.2.2 CFG_AxMaxAcc

数据类型：

F64

R/W：

RW

属性 ID：

554

含义：

配置运动轴的最大加速度 （单位：PPU/S^2）。

注解：

该属性的最大值 = FT_AxMaxAcc / CFG_AxPPU，最小值 = 1 / CFG_AxPPU。
对于 PCI-1245L，该默认值为 50,000,000。

6.4.2.2.2.3 CFG_AxMaxDec

数据类型：

F64

R/W：

RW

属性 ID：

555

含义：

配置运动轴的最大减速度 （单位：PPU/S^2）。

注解：

该属性的最大值 = FT_AxMaxDec / CFG_AxPPU，最小值 = 1 / CFG_AxPPU。
对于 PCI-1245L，该默认值为 50,000,000。

6.4.2.2.2.4 CFG_AxMaxJerk

数据类型：

F64

R/W：

R

属性 ID：

556

含义：

获取运动轴的最大加加速度配置。

注解：

对于 PCI-1245L，该值为 1。

6.4.2.2.3 Pulse In

6.4.2.2.3.1 CFG_AxPulseInMode

数据类型：

U32

R/W：

RW

属性 ID：

557

含义：

设置 / 获取编码器反馈脉冲输入模式。

值	说明
0	1XAB
1	2XAB
2	4XAB
3	CCW/CW

注解：

6.4.2.2.3.2 CFG_AxPulseInLogic

数据类型：

U32

R/W：

RW

属性 ID：

558

含义：

设置 / 获取编码器返回脉冲的逻辑。

值	说明
0	不倒转方向
1	倒转方向

注解：

6.4.2.2.3.3 CFG_AxPulseInMaxFreq

数据类型：

U32

R/W：

RW

属性 ID：

632

含义：

设置 / 获取频率中编码最大脉冲。

值	说明
0	500 KHz
1	1 MHz
2	2 MHz
3	4 MHz

注解：

6.4.2.2.4 脉冲输出

6.4.2.2.4.1 CFG_AxPulseOutMode

数据类型：

U32

R/W：

RW

属性 ID：

560

含义：

设置 / 获取命令脉冲输出模式。

值	说明
1	OUT/DIR
2	OUT/DIR，OUT 负逻辑
4	OUT/DIR，DIR 负逻辑
8	OUT/DIR，OUT&DIR 负逻辑
16	CW/CCW
32	CW/CCW，CW&CCW 负逻辑
256	CW/CCW，OUT 负逻辑
512	CW/CCW，DIR 负逻辑

注解：

对于 PCI-1245L，该默认值为 16。

请参考 FT_AxPulseOutMode。

6.4.2.2.5 报警

6.4.2.2.5.1 CFG_AxAlmLogic

数据类型：

U32

R/W：

RW

属性 ID：

562

含义：

设置 / 获取报警信号的有效逻辑电平。

值	说明
0	低准位
1	高准位

注解：

对于 PCI-1245L，该默认值为 1。

6.4.2.2.5.2 CFG_AxAlmEnable

数据类型：

U32

R/W：

RW

属性 ID：

561

含义：

启用 / 禁用运动报警功能。报警是当电机驱动处于报警状态时，电机驱动生成的一个信号。

值	说明
0	禁用
1	启用

注解：

对于 PCI-1245L，该默认值为 0。

如需修改 CFG_AxAlmEnable 值，请先行修改 CFG_AxAlmReact 和 CFG_AxAlmLogic。

6.4.2.2.5.3 CFG_AxAlmReact

数据类型：

U32

R/W：

RW

属性 ID：

563

含义：

设置 / 获取接收 ALARM 信号时的停止模式。

值	说明
0	电机立刻停止。
1	电机减速，然后停止。

注解：

对于 PCI-1245L，该默认值为 1。

6.4.2.2.6 到位

6.4.2.2.6.1 CFG_AxInpEnable

数据类型：

U32

R/W：

RW

属性 ID：

564

含义：

启用 / 禁用到位功能。

值	说明
0	禁用
1	启用

注解：

对于 PCI-1245L，该默认值为 0。

6.4.2.2.6.2 CFG_AxInpLogic

数据类型：

U32

R/W：

RW

属性 ID：

565

含义：

设置 / 获取到位信号的有效逻辑电平。

值	说明
0	低准位
1	高准位

注解：

对于 PCI-1245L，该默认值为 1。

6.4.2.2.7 ERC

6.4.2.2.7.1 CFG_AxErcLogic

数据类型：

U32

R/W：

RW

属性 ID：

566

含义：

设置 / 获取 ERC 信号的有效逻辑电平。

值	**说明**
0	低准位
1	高准位

注解：

对于 PCI-1245L，该默认值为 1。

6.4.2.2.7.2 CFG_AxErcEnableMode

数据类型：

U32

R/W：

RW

属性 ID：

569

含义：

设置 / 获取 ERC 输出模式或禁用 ERC 功能。

值	说明
0	禁用
1	返回原点后 ERC 输出
2	EMG/ALM/EL 激活时 ERC 输出 （不支持）
3	返回原点后或 EMG/ALM/EL 激活时 ERC 输出 （不支持）

注解：

对于 PCI-1245L，该默认值为 0。

6.4.2.2.8 硬件限位

6.4.2.2.8.1 CFG_AxElReact

数据类型：

U32

R/W：

RW

属性 ID：

576

含义：

设置 / 获取 EL 信号的反应模式。

值	说明
0	电机立刻停止
1	电机减速，然后停止

注解：

对于 PCI-1245L，该默认值为 0。

6.4.2.2.8.2 CFG_AxElLogic

数据类型：

U32

R/W：

RW

属性 ID：

575

含义：

设置 / 获取硬件限位信号的逻辑准位。

值	说明

0	低准位
1	高准位

注解：

对于 PCI-1245L，该默认值为 0。

6.4.2.2.8.3 CFG_AxElEnable

数据类型：

U32

R/W：

RW

属性 ID：

574

含义：

设置 / 获取硬件限位功能启用 / 禁用。

值	说明
0	禁用
1	启用

注解：

对于 PCI-1245L，该默认值为 1。
如需修改 CFG_AxElEnable 值，请先行修改 CFG_AxElReact 和 CFG_AxElLogic。

6.4.2.2.9 软件限位

6.4.2.2.9.1 CFG_AxSwMelEnable

数据类型：

U32

R/W：

RW

属性 ID：

577

含义：

启用 / 禁用负方向软件限位功能。

值	说明
0	禁用
1	启用

注解：

6.4.2.2.9.2 CFG_AxSwPelEnable

数据类型：

U32

R/W：

RW

属性 ID：

578

含义：

启用 / 禁用正方向软件限位功能。

值	说明
0	禁用
1	启用

注解：

6.4.2.2.9.3 CFG_AxSwMelReact

数据类型：

U32

R/W：

RW

属性 ID：

579

含义：

设置 / 获取负方向软件限位的反应模式。

值	说明
0	电机立刻停止
1	电机减速，然后停止

注解：

对于 PCI-1245L，该默认值为 1。

6.4.2.2.9.4 CFG_AxSwPelReact

数据类型：

U32

R/W：

RW

属性 ID：

580

含义：

设置 / 获取正方向软件限位的反应模式。

值	说明
0	电机立刻停止
1	电机减速，然后停止

注解：

对于 PCI-1245L，该默认值为 1。

6.4.2.2.9.5 CFG_AxSwMelValue

数据类型：

I32

R/W：

RW

属性 ID：

581

含义：

设置 / 获取负方向软件限位的值。该属性值的范围为：
-2,147,483,647 ~ +2,147,483,647。

注解：

6.4.2.2.9.6 CFG_AxSwPelValue

数据类型：

I32

R/W：

RW

属性 ID：

582

含义：

设置 / 获取正方向软件限位的值。该属性值的范围为：
-2,147,483,647 ~ +2,147,483,647。

注解：

6.4.2.2.10 原点

6.4.2.2.10.1 CFG_AxOrgLogic

数据类型：

U32

R/W：

RW

属性 ID：

589

含义：

设置 / 获取 ORG 信号的逻辑准位。

值	说明
0	低准位
1	高准位

注解：

对于 PCI-1245L，该默认值为 0。

6.4.2.2.10.2 CFG_AxEzLogic

数据类型：

U32

R/W：

RW

属性 ID：

591

含义：

设置 / 获取 EZ 信号的有效逻辑电平。

值	说明
0	低准位
1	高准位

注解：

对于 PCI-1245L，该默认值为 0。

6.4.2.2.10.3 CFG_AxHomeResetEnable

数据类型：

U32

R/W：

RW

属性 ID：

602

含义：

返回原点后，启用 / 禁用逻辑计数器的复位功能。

值	说明
0	禁用
1	启用

注解：

6.4.2.2.10.4 CFG_AxOrgReact

数据类型：

U32

R/W：

RW

属性 ID：

634

含义：

设定回原点结束时的行为模式。

值	说明
0	立即停止
1	减速停止

注解：

6.4.2.2.11 背隙补偿

6.4.2.2.11.1 CFG_AxBacklashEnable

数据类型：

U32

R/W：

RW

属性 ID：

593

含义：

启用 / 禁用背隙补偿。

值	说明
0	禁用
1	启用

注解：

对于 PCI-1245L，该默认值为 0。

6.4.2.2.11.2 CFG_AxBacklashPulses

数据类型：

U32

R/W：

RW

属性 ID：

594

含义：

设置 / 获取补偿脉冲个数。（单位：脉冲）

注解：

该值的范围为 0 ~ 4095。当方向发生变化时，轴在发送命令前会先输出背隙补偿脉冲。

对于 PCI-1245L，该默认值为 10。

6.4.2.2.11.3 CFG_AxBacklashVel

数据类型：

U32

R/W：

RW

属性 ID：

630

含义：

设置 / 获取背隙补偿的速度。（单位：脉冲 /s）
该速度是在原运行速度上进行叠加。

注解：

对于 PCI-1245L，该默认值为 1000。

6.4.2.2.12 Aux/Gen 输出

6.4.2.2.12.1 CFG_AxGenDoEnable

数据类型：

U32

R/W：

RW

属性 ID：

610

含义：

启用 / 禁用轴的通用 DO 功能。

值	说明
0	禁用
1	启用

注解：

如果启用属性 CFG_AxGenDoEnable，CFG_AxErcEnableMode 会自动禁用。

6.4.2.2.13 外部驱动

6.4.2.2.13.1 CFG_AxExtMasterSrc

数据类型：

U32

R/W：

RW

属性 ID：

611

含义：

设置 / 获取外部驱动的输入针脚。

值	说明
0	轴 0
1	轴 1 （不支持）
2	轴 2 （不支持）
3	轴 3 （不支持）

注解：

对于 PCI-1245L，仅支持 0。

6.4.2.2.13.2 CFG_AxExtSelEnable

数据类型：

U32

R/W：

RW

属性 ID：

612

含义：

当采用外部驱动时，通过数字输入通道提供驱动轴选项。

值	说明
0	禁用
1	启用 （不支持）

注解：

对于 PCI-1245L，仅支持 0。

6.4.2.2.13.3 CFG_AxExtPulseNum

数据类型：

U32

R/W：

RW

属性 ID：

613

含义：

当轴的外部驱动模式为MPG且A/B或B/A相位信号触发时，设置理论脉冲个数。

注解：

对于 PCI-1245L，默认值为 1。该值必须大于 0。

6.4.2.2.13.4 CFG_AxExtPulseInMode

数据类型：

U32

R/W：

RW

属性 ID：

617

含义：

设置 / 获取外部驱动脉冲输入模式。

值	说明
0	1XAB
1	2XAB
2	4XAB
3	CCW/CW

注解

6.4.2.2.13.5 CFG_AxExtPresetNum

数据类型：

U32

R/W：

RW

属性 ID：

618

含义：

当 “JOG” 模式接收到输入脉冲的一个有源沿时，设置 / 获取外部驱动个数。

注解：

对于 PCI-1245L，默认值为 1。该值必须大于 0。

6.4.2.2.14 同步起停

6.4.2.2.14.1 CFG_AxSimStartSource

数据类型：

U32

R/W：

RW

属性 ID：

633

含义：

设置 / 获取当前轴的同步起停模式。

值	说明
0	禁用
1	从设备 STA 针脚的信号上启动同步运动模式。（默认）
256	从轴 _0 的比较信号启动同步运动。
512	从轴 _1 的比较信号启动同步运动。
1024	从轴 _2 的比较信号启动同步运动。
2048	从轴 _3 的比较信号启动同步运动。
65536	当轴 _0 停止时启动同步运动。
131072	当轴 _1 停止时启动同步运动。

262144	当轴 _2 停止时启动同步运动。
524288	当轴 _3 停止时启动同步运动。

注解：

如果成功调用 Acm_AxSimStartSuspendAbs、Acm_AxSimStartSuspendRel 或 Acm_AxSimStartSuspendVel，轴将为等待状态。调用 Acm_AxSimStart 之后，轴开始运动；调用 Acm_AxSimStop 之后，轴停止运动。

同时开始模式应通过该属性设置。如果值为 1，等待轴将开始运动 （取决于 STA信号）。调用Acm_AxSimStart或Acm_AxSimStop仅需要等待轴中的一个轴。

如果值为 256 ~ 8192，同时开始信号来自比较信号。每个轴需要分配比较信号源，但是不能指定自身为比较源。每个同时轴需要调用 Acm_AxSimStop 以停止运动。

如果值为 65536 ~ 2097152，当指定轴的运动停止时，等待轴将开始同时运动。每个轴需要指定一个轴，该轴不能是其本身。每个同时轴需要调用 Acm_AxSimStop 以停止运动。

如果值为 0，同时运动禁用。

用户可从 FT_AxSimStartSourceMap 获取轴支持的同时模式。

6.4.2.2.15 触发停止

6.4.2.2.15.1 CFG_AxIN1StopEnable

数据类型：

U32

R/W：

R&W

属性 ID：

635

含义：

启用 / 禁用 IN1 触发停止功能。

值	说明
0	禁用
1	启用

6.4.2.2.15.2 CFG_AxIN1StopReact

数据类型：

U32

R/W：

R&W

属性 ID：

636

含义：

设定 / 获取 IN1 触发时的停止模式。

值	说明
0	立即停止
1	减速停止

6.4.2.2.15.3 CFG_AxIN1StopLogic

数据类型：

U32

R/W：

R&W

属性 ID：

637

含义：

设定 / 获取 IN1 触发停止功能的逻辑准位。

值	说明
0	低准位
1	高准位

6.4.2.2.15.4 CFG_AxIN2StopEnable

数据类型：

U32

R/W：

R&W

属性 ID：

638

含义：

启用 / 禁用 IN2 触发停止功能。

值	说明
0	禁用
1	启用

6.4.2.2.15.5 CFG_AxIN2StopReact

数据类型：

U32

R/W：

R&W

属性 ID：

639

含义：

设定 / 获取 IN2 触发时的停止模式。

值	说明
0	立即停止
1	减速停止

6.4.2.2.15.6 CFG_AxIN2StopLogic

数据类型：

U32

R/W：

R&W

属性 ID：

640

含义：

设定 / 获取 IN2 触发停止功能的逻辑准位。

值	说明

0	低准位
1	高准位

6.4.2.2.15.7 CFG_AxIN4StopEnable

数据类型：

U32

R/W：

R&W

属性 ID：

641

含义：

启用 / 禁用 IN4 触发停止功能。

值	说明
0	禁用
1	启用

6.4.2.2.15.8 CFG_AxIN4StopReact

数据类型：

U32

R/W：

R&W

属性 ID：

642

含义：

设定 / 获取 IN4 触发时的停止模式。

值	说明
0	立即停止
1	减速停止

6.4.2.2.15.9 CFG_AxIN4StopLogic

数据类型：

U32

R/W：

R&W

属性 ID：

643

含义：

设定 / 获取 IN4 触发停止功能的逻辑准位。

值	说明
0	低准位
1	高准位

6.4.2.2.15.10 CFG_AxIN5StopEnable

数据类型：

U32

R/W：

R&W

属性 ID：

644

含义：

启用 / 禁用 IN5 触发停止功能。

值	说明
0	禁用
1	启用

6.4.2.2.15.11 CFG_AxIN5StopReact

数据类型：

U32

R/W：

R&W

属性 ID：

645

含义：

设定 / 获取 IN5 触发时的停止模式。

值	说明
0	立即停止
1	减速停止

6.4.2.2.15.12 CFG_AxIN5StopLogic

数据类型：

U32

R/W：

R&W

属性 ID：

646

含义：

设定 / 获取 IN5 触发停止功能的逻辑准位。

值	说明
0	低准位
1	高准位

6.4.2.3 参数

6.4.2.3.1 速度模式

6.4.2.3.1.1 PAR_AxVelLow

数据类型：

F64

R/W：

RW

属性 ID：

401

含义：

设置 / 获取该轴的低速度 （起始速度）（单位：PPU/S）。

注解：

该属性值必须小于或等于 PAR_AxVelHigh。默认值为 2000 PPU。

6.4.2.3.1.2 PAR_AxVelHigh

数据类型：

F64

R/W：

RW

属性 ID：

402

含义：

设置 / 获取该轴的高速度 （驱动速度）（单位：PPU/s）。

注解：

该属性值必须小于 CFG_AxMaxVel 且大于 PAR_AxVelLow。默认值为 8000。

6.4.2.3.1.3 PAR_AxAcc

数据类型：

F64

R/W：

RW

属性 ID：

403

含义：

设置 / 获取该轴的加速度 （单位： PPU/s2）。

注解：

该属性值必须小于或等于 CFG_AxMaxAcc。默认值为 10000。

6.4.2.3.1.4 PAR_AxDec

数据类型：

F64

R/W：

RW

属性 ID：

404

含义：

设置 / 获取该轴的减速度 （单位： PPU/s^2）。

注解：

该属性值必须小于或等于 CFG_AxMaxDec。默认值为 10000。

6.4.2.3.1.5 PAR_AxJerk

数据类型：

F64

R/W：

RW

属性 ID：

405

含义：

设置速度曲线的类型：T 形曲线或 S 形曲线。

值	说明
0	T 形曲线 （默认）
1	S 形曲线

注解：

实际加加速度通过驱动计算。

如果 PAR_AxJerk 设置为 1，PAR_AxAcc 表示最大加速度，而非加速度；PAR_AxDec 表示最大减速度，而非减速度。

6.4.2.3.2 原点

6.4.2.3.2.1 PAR_AxHomeCrossDistance

数据类型：

F64

R/W：

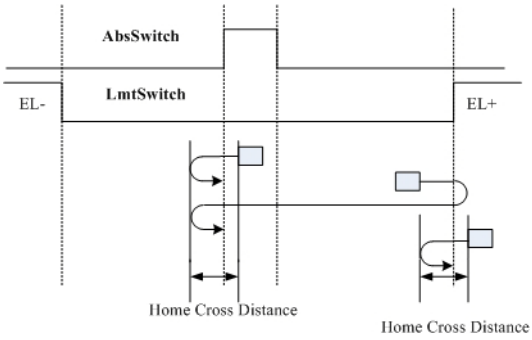
RW

属性 ID：

408

含义：

设置原点跨越距离 （单位：PPU）。该属性值必须大于 0，默认值为 10000。



6.4.2.3.2.2 PAR_AxHomeExSwitchMode

数据类型：

U32

R/W：

RW

属性 ID：

407

含义：

设置 Acm_AxHomeEx 的停止条件。

值	定义	说明
0	Level On	传感器开启 （激活）
1	Level Off	传感器关闭 （未激活）
2	Rising Edge	传感器从关闭到开启的转换
3	Falling Edge	传感器从开启到关闭的转换

6.4.3 群组

6.4.3.1 配置

6.4.3.1.1 系统

6.4.3.1.1.1 CFG_GpAxesInGroup

数据类型：

U32

R/W：

R

属性 ID：

806

含义：

获取关于哪个 （哪些）轴在该群组中的信息。

位	说明
0	0 轴
1	1 轴
2	2 轴
3	3 轴

注解：

6.4.3.2 参数

6.4.3.2.1 速度模式

6.4.3.2.1.1 PAR_GpVelLow

数据类型：

F64

R/W：

RW

属性 ID：

701

含义：

设置该群组的低速度 （起始速度）（单位：PPU/s）。该属性值必须小于或等于 Par_GpVelHigh。默认值为添加的第一个轴的 PAR_AxVelLow。

6.4.3.2.1.2 PAR_GpVelHigh

数据类型：

F64

R/W：

RW

属性 ID：

702

含义：

设置该群组的高速度 （驱动速度）（单位：PPU/s）。该属性值必须小于添加的第一个轴的 CFG_AxMaxVel，且大于 Par_GpVelLow。默认值为添加的第一个轴的 PAR_AxVelHigh。

6.4.3.2.1.3 PAR_GpAcc

数据类型：

F64

R/W：

RW

属性 ID：

703

含义：

设置该群组的加速度 （单位： PPU/s^2）。该属性值必须小于或等于添加的第一个轴的 CFG_AxMaxAcc。默认值为添加的第一个轴的 PAR_AxAcc。

6.4.3.2.1.4 PAR_GpDec

数据类型：

F64

R/W：

RW

属性 ID：

704

含义：

设置该群组的减速度 （单位： PPU/s^2）。该属性值必须小于或等于添加的第一个轴的 CFG_AxMaxDec。默认值为添加的第一个轴的 PAR_AxDec。

6.4.3.2.1.5 PAR_GpJerk

数据类型：

F64

R/W：

RW

属性 ID：

705

含义：

设置速度曲线类型：T 形曲线或 S 形曲线。

值	说明
0	T 形曲线 （默认）
1	S 形曲线

注解：

如果 PAR_GpJerk 设置为 1，PAR_GpAcc 表示最大加速度，而非加速度；PAR_GpDec 表示最大减速度，而非减速度。默认值为添加的第一个轴的加加速度。

6.4.3.2.2 系统

6.4.3.2.2.1 PAR_GpGroupID

数据类型：

U32

R/W：

R

属性 ID：

706

含义：

通过 GroupHandle 获取 GroupID。

注解：

PCI-1245L, 只有一个 GroupID 可用，为 0。

6.5 错误代码

错误代码	错误
0x00000000	SUCCESS
0x80000000	InvalidDevNumber
0x80000001	DevRegDataLost
0x80000002	LoadDllFailed
0x80000003	GetProcAddrFailed
0x80000004	MemAllocateFailed
0x80000005	InvalidHandle
0x80000006	CreateFileFailed
0x80000007	OpenEventFailed
0x80000008	EventTimeOut
0x80000009	InvalidInputParam
0x8000000a	PropertyIDNotSupport
0x8000000b	PropertyIDReadOnly
0x8000000c	ConnectWinIrqFailed
0x8000000d	InvalidAxCfgVel
0x8000000e	InvalidAxCfgAcc
0x8000000f	InvalidAxCfgDec
0x80000010	InvalidAxCfgJerk
0x80000011	InvalidAxParVelLow
0x80000012	InvalidAxParVelHigh
0x80000013	InvalidAxParAcc
0x80000014	InvalidAxParDec
0x80000015	InvalidAxParJerk
0x80000016	InvalidAxPulseInMode
0x80000017	InvalidAxPulseOutMode
0x80000018	InvalidAxAlarmEn
0x80000019	InvalidAxAlarmLogic
0x8000001a	InvalidAxInPEn
0x8000001b	InvalidAxInPLogic
0x8000001c	InvalidAxHLmtEn
0x8000001d	InvalidAxHLmtLogic
0x8000001e	InvalidAxHLmtReact
0x8000001f	InvalidAxSLmtPEn
0x80000020	InvalidAxSLmtPReact
0x80000021	InvalidAxSLmtPValue
0x80000022	InvalidAxSLmtMEn
0x80000023	InvalidAxSLmtMReact
0x80000024	InvalidAxSLmtMValue
0x80000025	InvalidAxOrgLogic
0x80000026	InvalidAxOrgEnable
0x80000027	InvalidAxEzLogic
0x80000028	InvalidAxEzEnable
0x80000029	InvalidAxEzCount
0x8000002a	InvalidAxState

0x8000002b	InvalidAxInEnable
0x8000002c	InvalidAxSvOnOff
0x8000002d	InvalidAxDistance
0x8000002e	InvalidAxPosition
0x8000002f	InvalidAxHomeModeKw
0x80000030	InvalidAxCntInGp
0x80000031	AxInGpNotFound
0x80000032	AxisInOtherGp
0x80000033	AxCannotIntoGp
0x80000034	GpInDevNotFound
0x80000035	InvalidGpCfgVel
0x80000036	InvalidGpCfgAcc
0x80000037	InvalidGpCfgDec
0x80000038	InvalidGpCfgJerk
0x80000039	InvalidGpParVelLow
0x8000003a	InvalidGpParVelHigh
0x8000003b	InvalidGpParAcc
0x8000003c	InvalidGpParDec
0x8000003d	InvalidGpParJerk
0x8000003e	JerkNotSupport
0x8000003f	ThreeAxNotSupport
0x80000040	DevIpoNotFinished
0x80000041	InvalidGpState
0x80000042	OpenFileFailed
0x80000043	InvalidPathCnt
0x80000044	InvalidPathHandle
0x80000045	InvalidPath
0x80000046	IoctlError
0x80000047	AmnetRingUsed
0x80000048	DeviceNotOpened
0x80000049	InvalidRing
0x8000004a	InvalidSlaveIP
0x8000004b	InvalidParameter
0x8000004c	InvalidGpCenterPosition
0x8000004d	InvalidGpEndPosition
0x8000004e	InvalidAddress
0x8000004f	DeviceDisconnect
0x80000050	DataOutBufExceeded
0x80000051	SlaveDeviceNotMatch
0x80000052	SlaveDeviceError
0x80000053	SlaveDeviceUnknow
0x80000054	FunctionNotSupport
0x80000055	InvalidPhysicalAxis
0x80000056	InvalidVelocity
0x80000057	InvalidAxPulseInLogic
0x80000058	InvalidAxPulseInSource
0x80000059	InvalidAxErcLogic
0x8000005a	InvalidAxErcOnTime

0x8000005b	InvalidAxErcOffTime
0x8000005c	InvalidAxErcEnableMode
0x8000005d	InvalidAxSdEnable
0x8000005e	InvalidAxSdLogic
0x8000005f	InvalidAxSdReact
0x80000060	InvalidAxSdLatch
0x80000061	InvalidAxHomeResetEnable
0x80000062	InvalidAxBacklashEnable
0x80000063	InvalidAxBacklashPulses
0x80000064	InvalidAxVibrationEnable
0x80000065	InvalidAxVibrationRevTime
0x80000066	InvalidAxVibrationFwdTime
0x80000067	InvalidAxAlarmReact
0x80000068	InvalidAxLatchLogic
0x80000069	InvalidFwMemoryMode
0x8000006a	InvalidConfigFile
0x8000006b	InvalidAxEnEvtArraySize
0x8000006c	InvalidAxEnEvtArray
0x8000006d	InvalidGpEnEvtArraySize
0x8000006e	InvalidGpEnEvtArray
0x8000006f	InvalidIntervalData
0x80000070	InvalidEndPosition
0x80000071	InvalidAxisSelect
0x80000072	InvalidTableSize
0x80000073	InvalidGpHandle
0x80000074	InvalidCmpSource
0x80000075	InvalidCmpMethod
0x80000076	InvalidCmpPulseMode
0x80000077	InvalidCmpPulseLogic
0x80000078	InvalidCmpPulseWidth
0x80000079	InvalidPathFunctionID
0x8000007a	SysBufAllocateFailed
0x8000007b	SpeedFordFunNotSpported
0x80000096	SlaveIOUpdateError
0x80000097	NoSlaveDevFound
0x80000098	MasterDevNotOpen
0x80000099	MasterRingNotOpen
0x800000c8	InvalidDIPort
0x800000c9	InvalidDOPort
0x800000ca	InvalidDOValue
0x800000cb	CreateEventFailed
0x800000cc	CreateThreadFailed
0x800000cd	InvalidHomeModeEx
0x800000ce	InvalidDirMode
0x800000cf	AxHomeMotionFailed
0x800000d0	ReadFileFailed
0x800000d1	PathBufIsFull
0x800000d2	PathBufIsEmpty

0x800000d3	GetAuthorityFailed
0x800000d4	GpIDAllocatedFailed
0x800000d5	FirmWareDown
0x800000d6	InvalidGpRadius
0x800000d7	InvalidAxCmd
0x800000d8	InvalidaxExtDrv
0x800000d9	InvalidGpMovCmd
0x800000da	SpeedCurveNotSupported
0x800000db	InvalidCounterNo
0x800000dc	InvalidPathMoveMode
0x800000dd	PathSelStartCantRunInSpeedForwareMode
0x800000de	InvalidCamTableID
0x800000df	InvalidCamPointRange
0x800000e0	CamTableIsEmpty
0x800000e1	InvalidPlaneVector
0x800000e2	MasAxIDSameSlvAxID
0x800000e3	InvalidGpRefPlane
0x800000e4	InvalidAxModuleRange
0x800000e5	DownloadFileFailed
0x800000e6	InvalidFileLength
0x800000e7	InvalidCmpCnt
0x800000e8	JerkExceededMaxValue
0x800000e9	AbsMotionNotSupport
0x800000ea	InvalidAiRange
0x800000eb	AIScaleFailed
0x80002000	HLmtPExceeded
0x80002001	HLmtNExceeded
0x80002002	SLmtPExceeded
0x80002003	SLmtNExceeded
0x80002004	AlarmHappened
0x80002005	EmgHappened
0x80002006	TimeLmtExceeded
0x80002007	DistLmtExceeded
0x80002008	InvalidPositionOverride
0x80002009	OperationErrorHappened
0x8000200a	SimultaneousStopHappened
0x8000200b	OverflowInPAPB
0x8000200c	OverflowInIPO
0x8000200d	STPHappened
0x8000200e	SDHappened
0x8000200f	AxsiNoCmpDataLeft
0x80004001	DevEvtTimeOut
0x80004002	DevNoEvt
0x10000001	Warning_AxWasInGp
0x10000002	Warning_GpInconsistRate
0x10000003	Warning_GpInconsistPPU
0x80005001	ERR_SYS_TIME_OUT
0x80005002	Dsp_PropertyIDNotSupport

0x80005003	Dsp_PropertyIDReadOnly
0x80005004	Dsp_InvalidParameter
0x80005005	Dsp_DataOutBufExceeded
0x80005006	Dsp_FunctionNotSupport
0x80005007	Dsp_InvalidConfigFile
0x80005008	Dsp_InvalidIntervalData
0x80005009	Dsp_InvalidTableSize
0x8000500a	Dsp_InvalidTableID
0x8000500b	Dsp_DataIndexExceedBufSize
0x8000500c	Dsp_InvalidCompareInterval
0x8000500d	Dsp_InvalidCompareRange
0x8000500e	Dsp_PropertyIDWriteOnly
0x8000500f	Dsp_NcError
0x80005010	Dsp_CamTableIsInUse
0x80005011	Dsp_EraseBlockFailed
0x80005012	Dsp_ProgramFlashFailed
0x80005014	Dsp_ReadPrivateOverMaxTimes
0x80005015	Dsp_InvalidPrivateID
0x80005017	Dsp_LastOperationNotOver
0x80005018	Dsp_WritePrivateTimeout
0x80005101	Dsp_InvalidAxCfgVel
0x80005102	Dsp_InvalidAxCfgAcc
0x80005103	Dsp_InvalidAxCfgDec
0x80005104	Dsp_InvalidAxCfgJerk
0x80005105	Dsp_InvalidAxParVelLow
0x80005106	Dsp_InvalidAxParVelHigh
0x80005107	Dsp_InvalidAxParAcc
0x80005108	Dsp_InvalidAxParDec
0x80005109	Dsp_InvalidAxParJerk
0x8000510a	Dsp_InvalidAxPptValue
0x8000510b	Dsp_InvalidAxState
0x8000510c	Dsp_InvalidAxSvOnOff
0x8000510d	Dsp_InvalidAxDistance
0x8000510e	Dsp_InvalidAxPosition
0x8000510f	Dsp_InvalidAxHomeMode
0x80005110	Dsp_InvalidPhysicalAxis
0x80005111	Dsp_HLmtPExceeded
0x80005112	Dsp_HLmtNExceeded
0x80005113	Dsp_SLmtPExceeded
0x80005114	Dsp_SLmtNExceeded
0x80005115	Dsp_AlarmHappened
0x80005116	Dsp_EmgHappened
0x80005117	Dsp_CmdValidOnlyInConstSec
0x80005118	Dsp_InvalidAxCmd
0x80005119	Dsp_InvalidAxHomeDirMode
0x8000511a	Dsp_AxisMustBeModuloAxis
0x8000511b	Dsp_AxIdCantSameAsMasId
0x8000511c	Dsp_CantResetPosiOfMasAxis

0x8000511d	Dsp_InvalidAxExtDrvOperation
0x8000511e	Dsp_AxAccExceededMaxAcc
0x8000511f	Dsp_AxVelExceededMaxVel
0x80005120	Dsp_NotEnoughPulseForChgV
0x80005121	Dsp_NewVelMustGreaterThanVelLow
0x80005122	Dsp_InvalidAxGearMode
0x80005123	Dsp_InvalidGearRatio
0x80005201	Dsp_InvalidAxCntInGp
0x80005202	Dsp_AxInGpNotFound
0x80005203	Dsp_AxisInOtherGp
0x80005204	Dsp_AxCannotIntoGp
0x80005205	Dsp_GpInDevNotFound
0x80005206	Dsp_InvalidGpCfgVel
0x80005207	Dsp_InvalidGpCfgAcc
0x80005208	Dsp_InvalidGpCfgDec
0x80005209	Dsp_InvalidGpCfgJerk
0x8000520a	Dsp_InvalidGpParVelLow
0x8000520b	Dsp_InvalidGpParVelHigh
0x8000520c	Dsp_InvalidGpParAcc
0x8000520d	Dsp_InvalidGpParDec
0x8000520e	Dsp_InvalidGpParJerk
0x8000520f	Dsp_JerkNotSupport
0x80005210	Dsp_ThreeAxNotSupport
0x80005211	Dsp_DevIpoNotFinished
0x80005212	Dsp_InvalidGpState
0x80005213	Dsp_OpenFileFailed
0x80005214	Dsp_InvalidPathCnt
0x80005215	Dsp_InvalidPathHandle
0x80005216	Dsp_InvalidPath
0x80005217	Dsp_GpSlavePositionOverMaster
0x80005219	Dsp_GpPathBufferOverflow
0x8000521a	Dsp_InvalidPathFunctionID
0x8000521b	Dsp_SysBufAllocateFailed
0x8000521c	Dsp_InvalidGpCenterPosition
0x8000521d	Dsp_InvalidGpEndPosition
0x8000521e	Dsp_InvalidGpCmd
0x8000521f	Dsp_AxHasBeenInInGp
0x80005220	Dsp_InvalidPathRange

附录 A

软件功能表

A.1 软件功能表

	项目	说明	PCI-1245L
运动功能	单轴运动	手轮控制	√
		MPG 控制	√
		T&S 形速度曲线	√
		可程设的加速度和减速度	√
		点到点运动	√
		位置 / 速度重设	√
		连续运动	√
		背隙补偿	√
		停止	√
	多个轴 （群组）运动	群组	1 个群组
		Line：轴	2 个轴
	返回原点	16 种模式	√
	同步启停	同时开始 / 停止	√
中断	轴	轴停止	√
		轴错误	√
		轴 VH 开始	√
		轴 VH 停止	√
	群组	群组停止	√
		群组 VH 开始	√
		群组 VH 结束	√

附录 B

规格

B.1 轴

项目	描述
轴数	4
控制输出类型	脉冲

B.2 数字量输入

项目		描述
通道		LMT+,LMT-, ORG, INP, ALM, EMG, RDY
类型		一个终端，光隔离
输入电压	L(max)	4 V_{DC}
	H(min)	10 V_{DC}
	H(max)	30 V_{DC}
最大输入延迟时间		150 us
输入阻抗		8.4 kΩ
保护		2,500 V 隔离

B.3 高速数字量输入

项目		描述
通道		JOG+, JOG-
类型		一个终端，光隔离
输入电压	L(max)	3 V_{DC}
	H(min)	10 V_{DC}
	H(max)	30 V_{DC}
最大输入延迟时间		2 us
输入阻抗		8.4 kΩ
保护		2,500 V 隔离

B.4 数字量输出

项目		描述
通道		SVON, ERC
类型		一个终端，光隔离，Sink
工作电压	Low	5 V_{DC}
	High	30 V_{DC}
最大灌电流		每通道 120 mA
最大输出延迟时间		60 us
保护		2,500 V 隔离

B.5 高速数字量输出

项目		描述
通道		OUT4, OUT5
类型		一个终端，光隔离，Sink
工作电压	Low	5 V_{DC}
	High	30 V_{DC}
最大灌电流		每通道 120 mA
最大输出延迟时间		20 us
保护		2,500 V 隔离

B.6 输入脉冲

项目		描述
通道		ECA+,ECA-,ECB+,ECB-,ECZ+,ECZ-
类型		两个终端，光隔离
最大输入频率		1MHz x1, x2, x4 （仅用于 A/B 相位）
输入电压	L(max)	1 V_{DC}
	H(min)	3.5 V_{DC}
	H(max)	10 V_{DC}
保护		2,500 V 隔离

B.7 输出脉冲

项目		描述
最大频率		1 Mpps
类型		两个终端，光隔离
输出电压	L(max)	2 V_{DC} /35 mA
	H(min)	3.9 V_{DC} /0 mA
输出电流		2Vdc/35mA; 2.5Vdc/30mA; 3Vdc/15mA; 3.4Vdc/1mA;3.9Vdc/0mA
输出信号模式		差分线路驱动器输出
保护		2,500 V 隔离

B.8 一般规格

项目		描述
接口		SCSI D-SUB-100P
产品尺寸		175mm x 100mm
认证		CE,FCC A 级认证
功耗	典型	+5 V / 0.6 A
	最大	+5 V / 1 A
温度	工作	0 ~ 60° C (参考 IEC 60068-2-1,2)
	存储	-20 ~ 85° C
相对湿度		5 ~ 95% RH，非凝结 (参考 IEC 60068-2-3)
外部电压		DC +12 ~ 24 V

ADVANTECH

Enabling an Intelligent Planet

www.advantech.com.cn