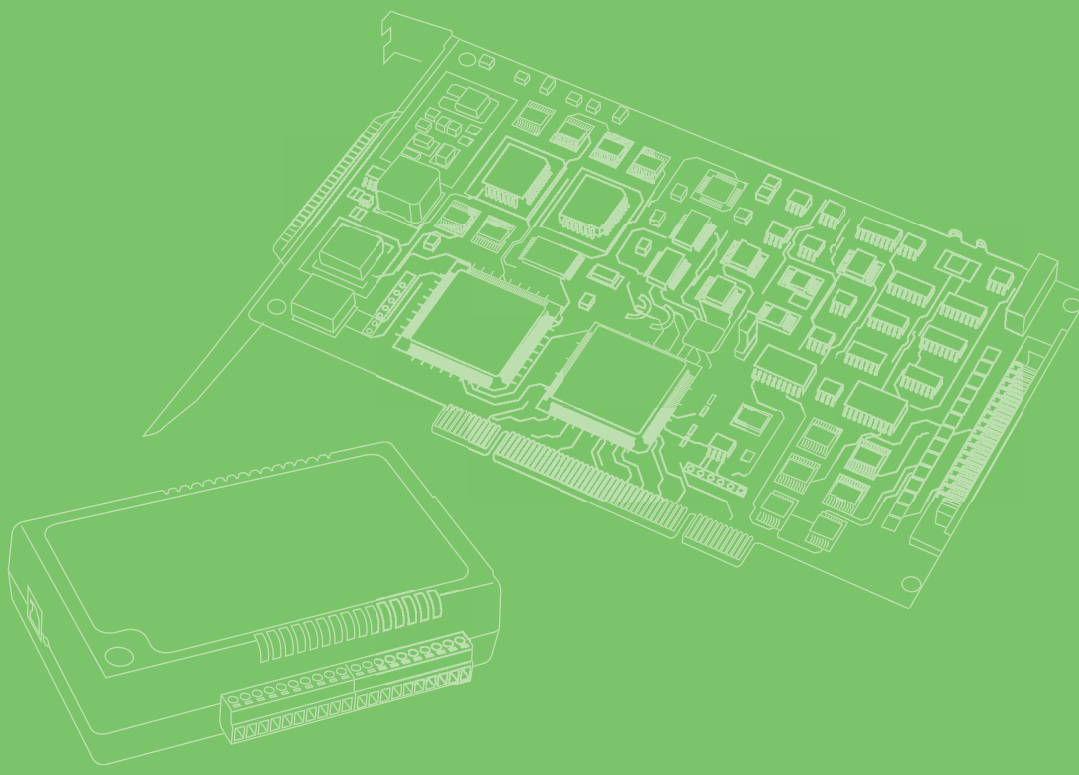


用户手册

PCI-1285/1285E 系列

基于 DSP 的 SoftMotion PCI 控制器

ADVANTECH

Enabling an Intelligent Planet

版权声明

认可声明

PCI-1285/1285E 用户手册中文第一版，参照 PCI-1285/1285E 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：2003128520　　第一版

中国印刷　　2013 年 5 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-1285/1285E 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-1285/1285E 的基本信息、特殊特性以及详细规格。

PCI-1285/1285E 系列是基于 DSP 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 DSP，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。

研华 SoftMotion 支持以下特性：龙门同步控制、电子齿轮和电子凸轮；线性、曲线和螺旋曲线插补；实现缓冲分段轨迹的连续运动；切线跟随运动以保证 Z 轴与 X-Y 曲线相切；通过任何第三方机器视觉解决方案支持高速位置比较和触发。

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-1285/1285E 具有以下特性 （特性根据产品型号不同而不同）：

- 4xAB 模式的编码器输入为 10 MHz，CW/CCW 模式的编码器输入为 2.5 MHz
- 脉冲输出高达 5 Mpps
- DSP 中具有轨迹规划的内存缓冲 （10 K 个点）
- 支持电子齿轮和螺旋插补
- 支持电子凸轮，提供 256 个点用以描述凸轮轮廓 （缓存位于 DSP 中）
- 硬件紧急输入
- 看门狗定时器
- 通过 ORG 和 index 支持位置锁存
- 位置比较触发高达 100 KHz，DSP 中的内存缓冲高达 100 K
- 可编程中断
- 通过半闭环脉冲串控制支持龙门模式
- RDY/LTC 专用输入通道 & SVON/CMP/CAM-DO/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-1285/1285E 板卡
- 用户手册
- 驱动和软件
- 实用程序
- 连接 PCI 板卡和端子板的 PCL-101100SB 接线电缆
- ADAM-3956 端子板
- 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 和松下公司的 A4/A5/MINAS A and Delta 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-101100SB – PCL-101100SB 是一根 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。

接线电缆表	
PCI-1285/1285E	PCL-101100SB

接线板

- 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-1285/1285E 板卡，则需要两块接线板进行 8 轴控制。松下公司的 A4/A5/MINAS 伺服、安川电气的 Sigma V，三菱公司的 J3 和台达的 A2 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-3956 和松下 A4/A5 伺服。
- PCL-10153PA5LS – PCL-10153PA5LS 是一根 50 针电缆，连接 ADAM-3956 和松下 MINAS A 伺服。
- PCL-10153YS5 – PCL-10153YS5 是一根 50 针电缆，连接 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-3956 和三菱公司的 J3 伺服。
- PCL-10153DA2 – PCL-10153DA2 是一根 50 针电缆，连接 ADAM-3956 和台达 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-1285/1285E 包装后，请首先检查里面的物品。包装内应包括以下各项：

- PCI-1285/1285E 板卡
- 所附光盘 （包含 DLL 驱动和用户手册）

PCI-1285/1285E 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-1285/1285E 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-1285/1285E 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-1285/1285E 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-1285/1285E 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 （100 针电缆、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-1285/1285EI/O 接口针脚定义

PCI-1285/1285E 的 I/O 接口是一个 200 针接口，可通过 PCL-101100SB 屏蔽电缆连接其它附件。

图 3.1 和图 3.2 为 PCI-1285/1285E 上 200 针 I/O 接口的针脚定义。表 3.1 为 I/O 接口信号说明。

注！ *PCL-101100SB 屏蔽电缆专为 PCI-1285/1285E 系列板卡设计，能够减少模拟信号线中的噪声。请参考第 1.4 节 “ 附件 ”。*

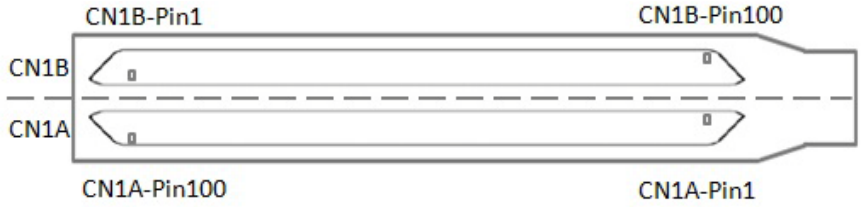


图 3.1： PCI-1285/PCI-1285E 的接口 CN1A, CN1B 针脚定义

CN1A

VEX	1	51	VEX
EMG	2	52	NC/EMG
X0_LMT+	3	53	X2_LMT+
X0_LMT-	4	54	X2_LMT-
X0_IN1/LTC	5	55	X2_IN1/LTC
X0_IN2/RDY	6	56	X2_IN2/RDY
X0_ORG	7	57	X2_ORG
X1_LMT+	8	58	X3_LMT+
X1_LMT-	9	59	X3_LMT-
X1_IN1/LTC	10	60	X3_IN1/LTC
X1_IN2/RDY	11	61	X3_IN2/RDY
X1_ORG	12	62	X3_ORG
X0_INP	13	63	X2_INP
X0_ALM	14	64	X2_ALM
X0_ECA+	15	65	X2_ECA+
X0_ECA-	16	66	X2_ECA-
X0_ECB+	17	67	X2_ECB+
X0_ECB-	18	68	X2_ECB-
X0_ECZ+	19	69	X2_ECZ+
X0_ECZ-	20	70	X2_ECZ-
X1_INP	21	71	X3_INP
X1_ALM	22	72	X3_ALM
X1_ECA+	23	73	X3_ECA+
X1_ECA-	24	74	X3_ECA-
X1_ECB+	25	75	X3_ECB+
X1_ECB-	26	76	X3_ECB-
X1_ECZ+	27	77	X3_ECZ+
X1_ECZ-	28	78	X3_ECZ-
X0_IN3/JOG+	29	79	X2_IN3/JOG+
X0_IN4/JOG-	30	80	X2_IN4/JOG-
X1_IN3	31	81	X3_IN3
X1_IN4	32	82	X3_IN4
GND	33	83	GND
X0_OUT4/CAM-DO	34	84	X2_OUT4/CAM-DO
X0_OUT5/CMP	35	85	X2_OUT5/CMP
X0_OUT6/SVON	36	86	X2_OUT6/SVON
X0_OUT7/ERC	37	87	X2_OUT7/ERC
X0_CW+/PULS+	38	88	X2_CW+/PULS+
X0_CW-/PULS-	39	89	X2_CW-/PULS-
X0_CCW+/DIR+	40	90	X2_CCW+/DIR+
X0_CCW-/DIR-	41	91	X2_CCW-/DIR-
GND	42	92	GND
X1_OUT4/CAM-DO	43	93	X3_OUT4/CAM-DO
X1_OUT5/CMP	44	94	X3_OUT5/CMP
X1_OUT6/SVON	45	95	X3_OUT6/SVON
X1_OUT7/ERC	46	96	X3_OUT7/ERC
X1_CW+/PULS+	47	97	X3_CW+/PULS+
X1_CW-/PULS-	48	98	X3_CW-/PULS-
X1_CCW+/DIR+	49	99	X3_CCW+/DIR+
X1_CCW-/DIR-	50	100	X3_CCW-/DIR-

CN1B

VEX	1	51	VEX
EMG	2	52	NC/EMG
X4_LMT+	3	53	X6_LMT+
X4_LMT-	4	54	X6_LMT-
X4_IN1/LTC	5	55	X6_IN1/LTC
X4_IN2/RDY	6	56	X6_IN2/RDY
X4_ORG	7	57	X6_ORG
X5_LMT+	8	58	X7_LMT+
X5_LMT-	9	59	X7_LMT-
X5_IN1/LTC	10	60	X7_IN1/LTC
X5_IN2/RDY	11	61	X7_IN2/RDY
X5_ORG	12	62	X7_ORG
X4_INP	13	63	X6_INP
X4_ALM	14	64	X6_ALM
X4_ECA+	15	65	X6_ECA+
X4_ECA-	16	66	X6_ECA-
X4_ECB+	17	67	X6_ECB+
X4_ECB-	18	68	X6_ECB-
X4_ECZ+	19	69	X6_ECZ+
X4_ECZ-	20	70	X6_ECZ-
X5_INP	21	71	X7_INP
X5_ALM	22	72	X7_ALM
X5_ECA+	23	73	X7_ECA+
X5_ECA-	24	74	X7_ECA-
X5_ECB+	25	75	X7_ECB+
X5_ECB-	26	76	X7_ECB-
X5_ECZ+	27	77	X7_ECZ+
X5_ECZ-	28	78	X7_ECZ-
X4_IN3/JOG+	29	79	X6_IN3/JOG+
X4_IN4/JOG-	30	80	X6_IN4/JOG-
X5_IN3	31	81	X7_IN3
X5_IN4	32	82	X7_IN4
GND	33	83	GND
X4_OUT4/CAM-DO	34	84	X6_OUT4/CAM-DO
X4_OUT5/CMP	35	85	X6_OUT5/CMP
X4_OUT6/SVON	36	86	X6_OUT6/SVON
X4_OUT7/ERC	37	87	X6_OUT7/ERC
X4_CW+/PULS+	38	88	X6_CW+/PULS+
X4_CW-/PULS-	39	89	X6_CW-/PULS-
X4_CCW+/DIR+	40	90	X6_CCW+/DIR+
X4_CCW-/DIR-	41	91	X6_CCW-/DIR-
GND	42	92	GND
X5_OUT4/CAM-DO	43	93	X7_OUT4/CAM-DO
X5_OUT5/CMP	44	94	X7_OUT5/CMP
X5_OUT6/SVON	45	95	X7_OUT6/SVON
X5_OUT7/ERC	46	96	X7_OUT7/ERC
X5_CW+/PULS+	47	97	X7_CW+/PULS+
X5_CW-/PULS-	48	98	X7_CW-/PULS-
X5_CCW+/DIR+	49	99	X7_CCW+/DIR+
X5_CCW-/DIR-	50	100	X7_CCW-/DIR-

图 3.2： PCI-1285/PCI-1285E 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
LTC	-	输入	位置锁存
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
CAM-DO	EGND	输出	位置区间 DO
CMP	EGND	输出	比较脉冲输出
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. *X1~X7 分别表示每个轴的 ID。*
2. *RDY & LTC 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、CMP、CAM-DO 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. *方便起见，运动实用程序中将使用 Xn_OUT4/CAM-DO、Xn_OUT5/CMP、Xn_OUT6/SVON 和 Xn_OUT7/ERC。n:0~7*
5. *Xn_IN3 有三種切換功能： 通用、JOG+ 和 MPG+ （手动脉冲器）。其中 n:0, 2, 4, 6*
6. *Xn_IN4 有三種切換功能： 通用輸入、JOG- 和 MPG- （手动脉冲器）。其中 n:0, 2, 4, 6*

3.2 DIP 开关的位置

图 3.2 为 PCI-1285/1285E 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-1285/1285E 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。

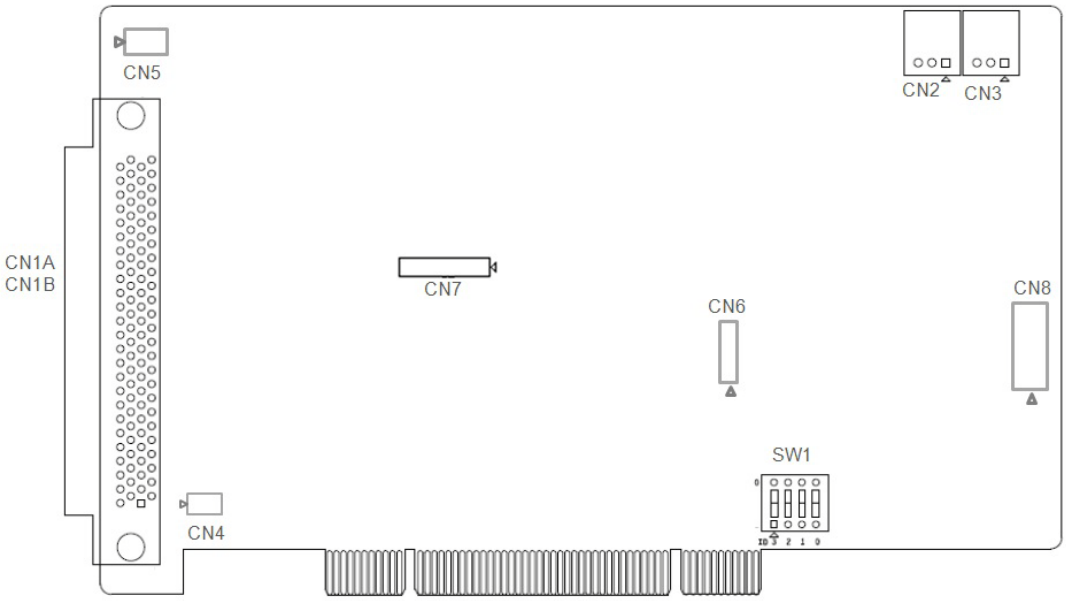


图 3.3： 跳线和 DIP 开关的位置

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 / 计数器顺时针模式；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 3.4： 光耦合器接口

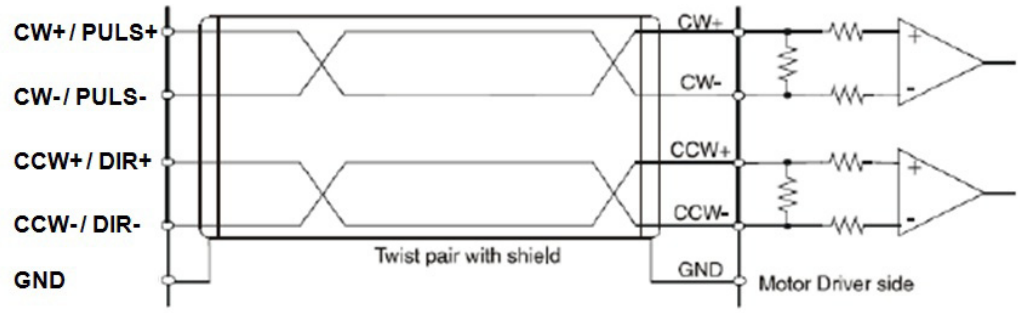


图 3.5： 线性驱动接口

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。

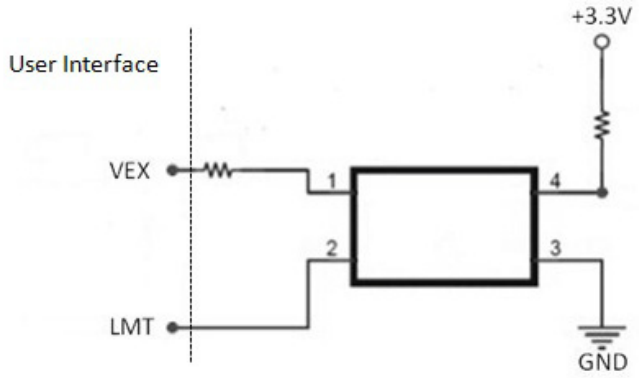


图 3.6： 限位输入信号的电路图

3.5 位置锁存 [LTC]

这是一个通用输入针脚，用于锁存同时位置信息。用户可通过编程读取位置计数器。有关详细信息，请参考第 6 章。

3.6 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。

3.7 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。

3.8 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。

3.9 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。

3.10 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-1285/1285E 的默认设置为正交输入 （4xAB 相位）。下图为一个通道的接口电路：

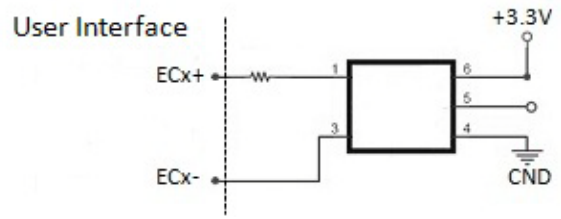


图 3.7： 编码器反馈的电路图

在上述电路图中，PCI-1285/1285E 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 10 MHz。

3.11 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。

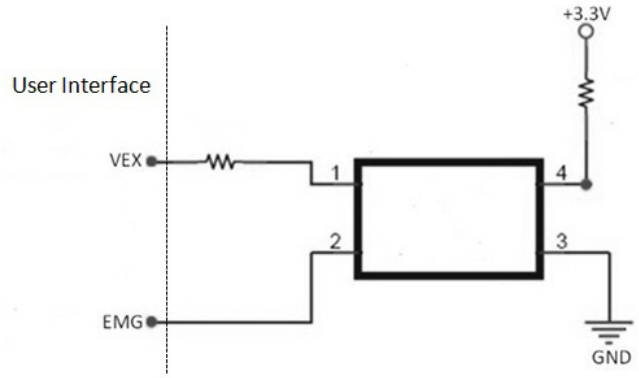


图 3.8： 紧急停止输入信号的电路图

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。

3.12 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！ *请勿直接将 VEX 接脚连接电感性负载。*

3.13 位置窗口输出 [CAM-DO]

如下图所示，用户可定义间隔和电平以生成一个具有指定持续时间的数字量输出。

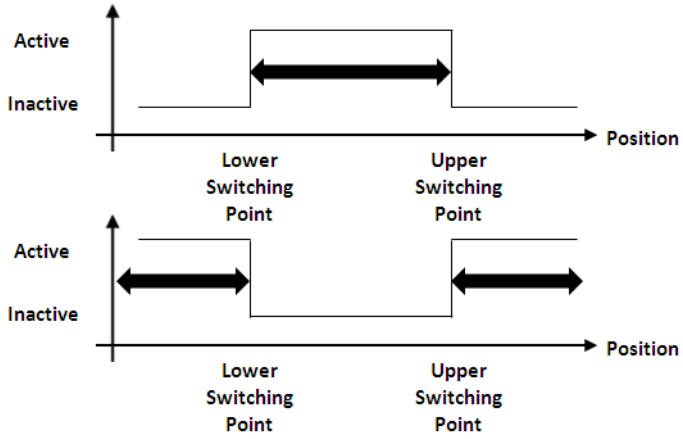


图 3.9： 位置窗口输出的电路图

3.14 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。

3.15 清除伺服误差计数器 [ERC]

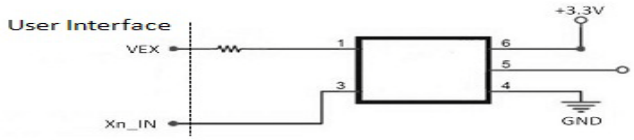
伺服驱动可生成偏差计数器清除信号，板卡可接收该信号作为通用输入。以下情况将清除计数器：返回原点、紧急停止情况、伺服报警以及行程限位激活。

3.16 位置比较输出 [CMP]

位置比较输出专为能够使用位置比较输出同步其它第三方视觉设备的客户设计。对于 PCI-1285/1285E，位置比较输出通道由针脚定义 - CMP 确定。

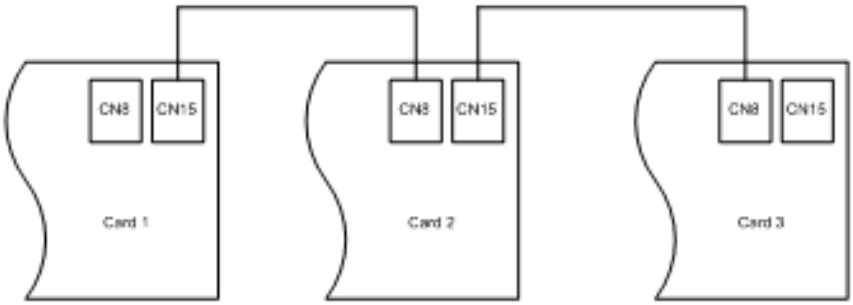
3.17 JOG 和 MPG

针脚定义 - Xn_IN3 & Xn_IN4 可支持 JOG 和 MPG 模式。这两个针脚可互相切换。Xn_IN3 有三种功能：通用数字量输入、JOG+ 和 MPG_A。Xn_IN4 同样也有三种功能：通用数字量输入、JOG- 和 MPG_B。电路图如下所示：



3.18 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。



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。

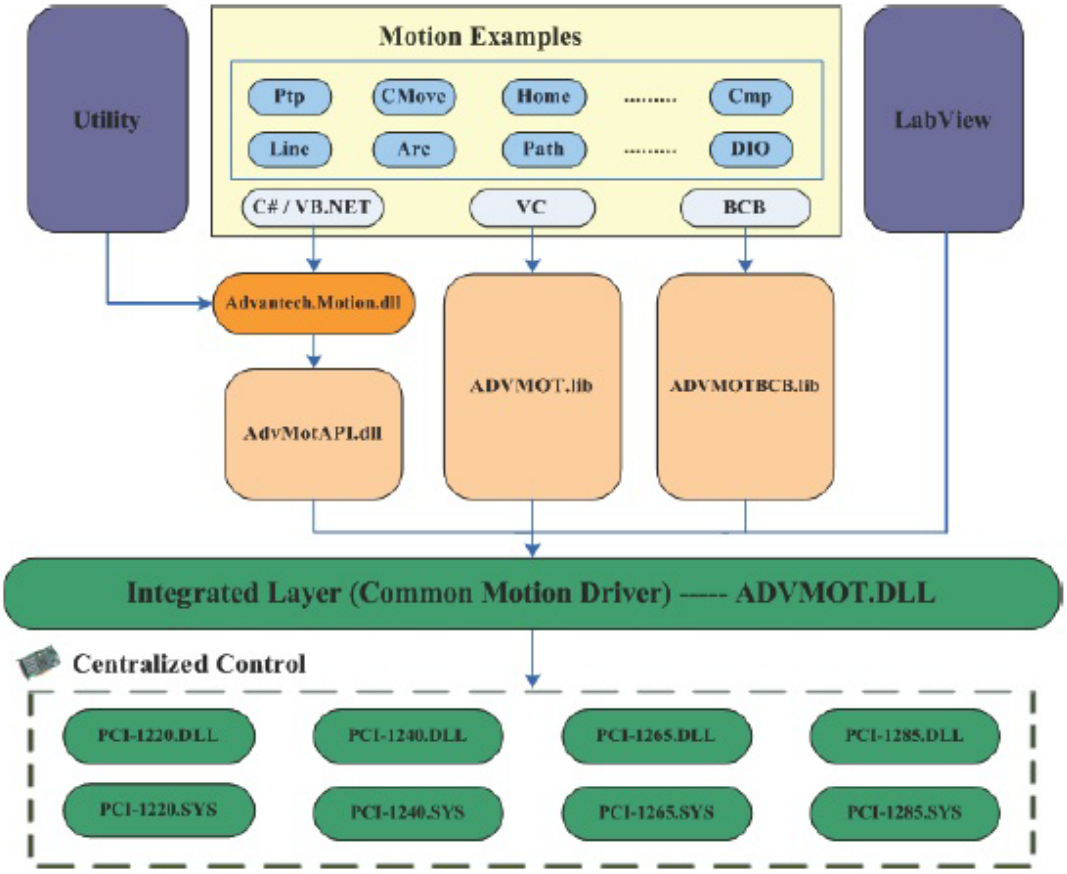
该架构包括三层：设备驱动层、整合层和应用层。用户无需了解如何操作特定设备的特定驱动，只需了解通用运动驱动即可。即使支持该架构的设备发生变化，应用也无需修改。

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-1285E 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

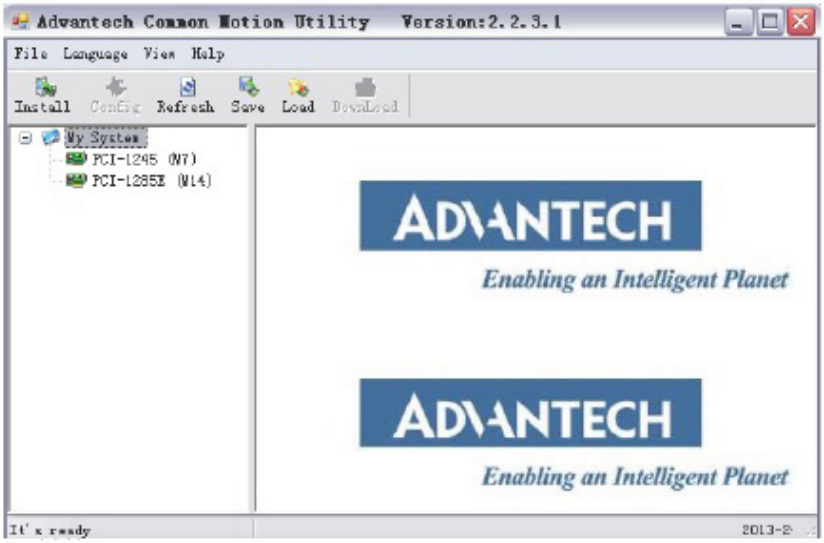
测试工具按照通用运动架构由 .Net 控件类库开发。.Net 控件类库包括组件（Device、Axis 和 Group）以及控件 （AxisSetupView、AxisScopeView、AxisDiagView、GroupPathView 和 GroupSpeedView）。新版测试工具与旧版 AdvMotionUtility 界面风格一致，功能兼容。 新版测试工具支持 PCI-1220U、PCI-1240U 和 PCI-1245/1245V/1245E/1265/1285/1285E 系列产品。

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 / 圆弧 / 螺旋 （Line/Arc/Helix）插补运动以及切向跟随运动。
4. Synchronized Motion：主要介绍同步运动操作， 包括电子凸轮 （E-CAM），电子齿轮 （E-Gear）和龙门 （Gantry）运动。
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。
7. Analog Input：展示 Device 的模拟输入通道状态。

5.2 Main Form



5.2.1 主窗体

5.2.1.1 File

单击 [Exit] 终止该进程。

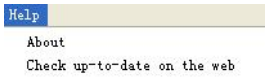
5.2.1.2 Language

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

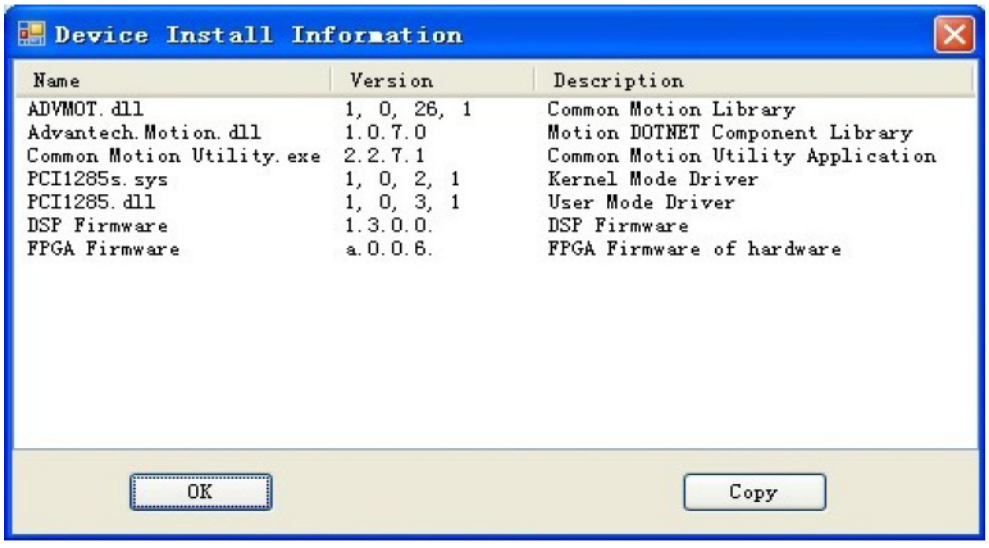


[About] 菜单项提供设备驱动和测试工具的版权声明信息。单击 [Check up-to-date on the web] 将链接到公司的网站，通过比较 “Install” 界面的的版本信息检查固件、驱动和测试工具是否为最新版。

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！ *只有基于 DSP 的运动控制卡才会显示 DSP 和 FPGA 信息。*

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

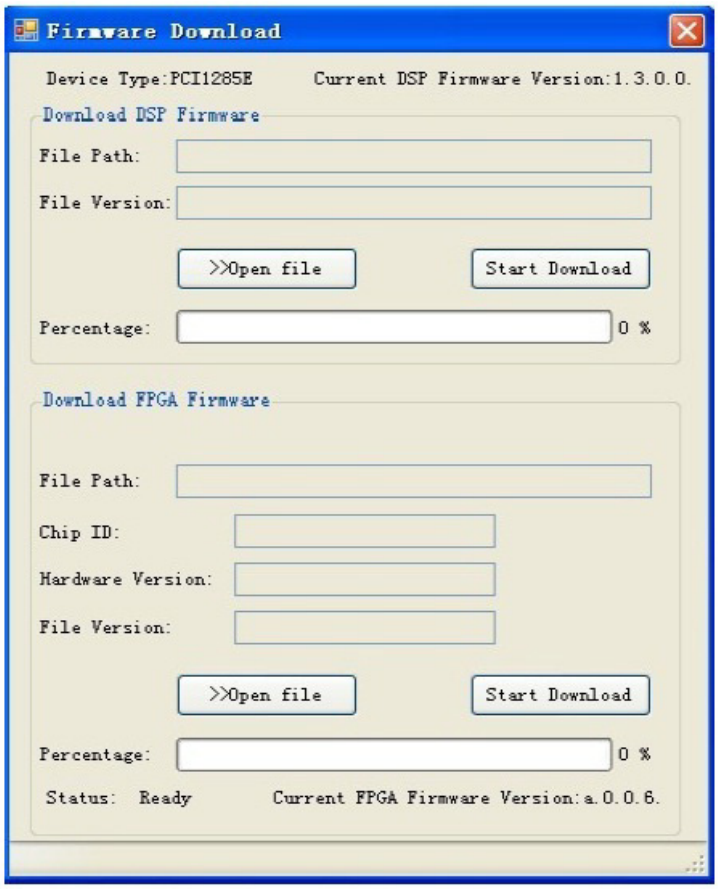
该按钮用于保存所选设备的轴的全部属性。

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-1285/1285E 系列产品是基于 DSP 的运动控制器。单击设备后，用户将看到如下界面。



该工具按钮可实现 DSP Firmware 和 FPGA Firmware 的下载。FPGA Firmware 又分为 U2 和 U7 的下载。FPGA Firmware 的下载动作同 DSP Firmware。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。*

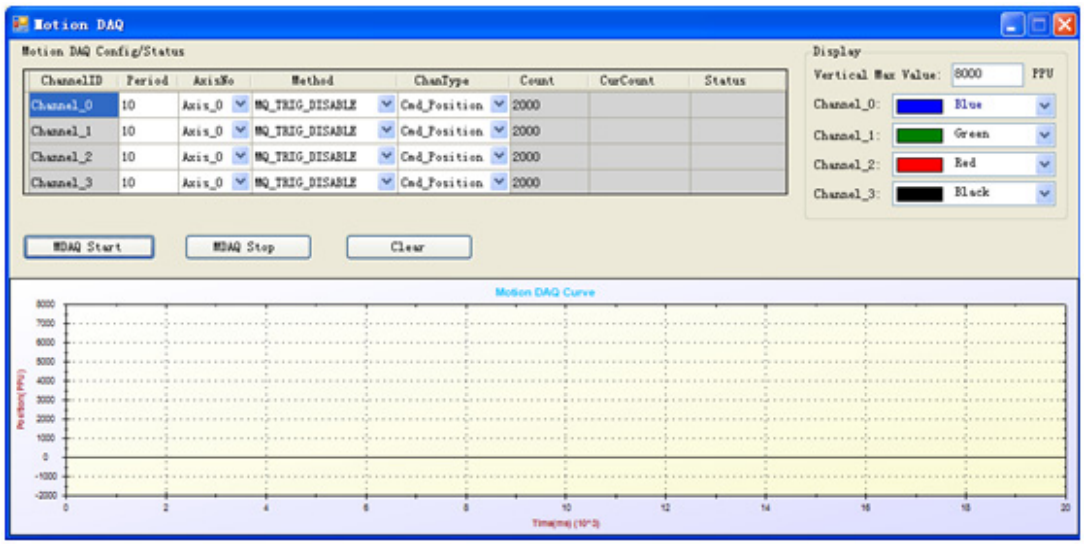
5.2.2.6 Motion DAQ

该工具按钮主要展示运动数据获取 （Motion Data Acquisition）功能。在 PCI-1245/1245V/1245E/1265 中，提供了 4 个通道进行运动数据的采集，任一通道都可采集任一轴的理论 （Command）/ 实际 (Actual)/ 偏差位置 （Lag，理论与实际之差）的运动数据，采集数据的最大数为 2000。

注！

目前只有 PCI-1245，PCI-1245V，PCI-1245E 及 PCI-1265 支持此功能，能看到此工具按钮。PCI-1285/1285E 暂不支持此功能。

单击按钮后，用户将看到如下界面：



此界面由以下几部分组成：

1. Motion DAQ 配置 / 状态

   可直接在 DataGridView 中进行各个通道采集数据的配置。其中，ChannelID、Count、CurCount 和 Status 列是只读的 （背景色为灰色的）。

   ChannelID：表示为哪一个通道。共提供 Channel_0，Channel_1，Channel_2，Channel_3 4 个通道。

   AxisNo：可选择设备中的任一轴。

   Period：采样周期，即每隔多长时间 （ms）采集一个数据。范围为 1-255ms。为了统一图形框的横坐标最大值，各通道的 Period 将采用同一个值。因此，若一个通道的 Period 值改了，则所有通道的 Period 值将会随着变化。

   Method：触发方式。数据采集的触发方式有如下几种：

   0：MQ_TRIG_DISABLE：禁用数据采集功能

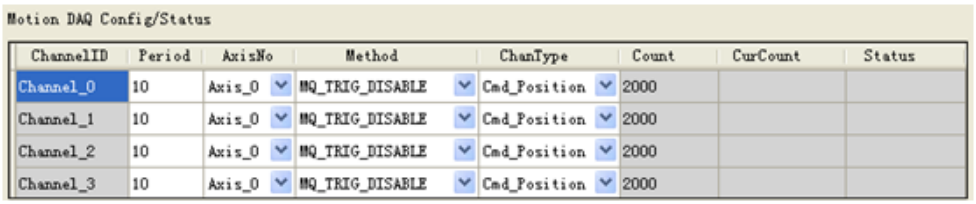
   1：MQ_TRIG_SW：由软件触发 （点击 MDAQ Start 触发）

   2：MQ_TRIG_DI：由 DI 触发 （保留）

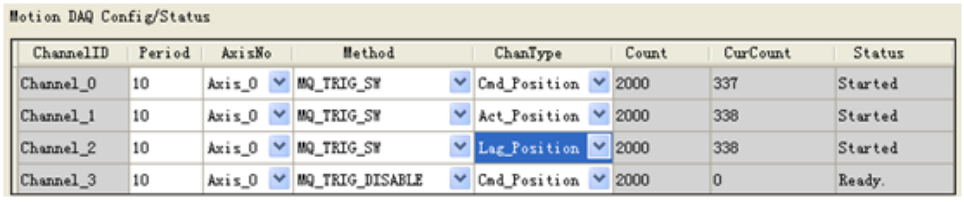
3：MQ_TRIG_AX0_START：0 轴开始运动时触发
4：MQ_TRIG_AX1_START：1 轴开始运动时触发
5：MQ_TRIG_AX2_START：2 轴开始运动时触发
6：MQ_TRIG_AX3_START：3 轴开始运动时触发
7：MQ_TRIG_AX4_START：4 轴开始运动时触发
8：MQ_TRIG_AX5_START：5 轴开始运动时触发
9：MQ_TRIG_AX6_START：6 轴开始运动时触发
10：MQ_TRIG_AX7_START：7 轴开始运动时触发
11：MQ_TRIG_AX8_START：8 轴开始运动时触发
12：MQ_TRIG_AX9_START：9 轴开始运动时触发
13：MQ_TRIG_AX10_START：10 轴开始运动时触发
14：MQ_TRIG_AX11_START：11 轴开始运动时触发

目前，在 PCI-1245/1245V/1245E 中，只支持 0-6 种触发方式；在 PCI-1265 中，只支持 0-8 种触发方式。此外，由 DI 触发的方式暂保留。
ChanType：数据采集的来源。可选值如下：
0：Cmd_Position：理论位置 （Command Position）。
1：Act_Position：实际位置 （Actual Position）。
2：Lag_Position：偏差位置 （Lag Position），即理论位置与实际位置的差值。
3：Cmd_Velocity （保留）：理论速度 （Command Velocity）。
Count：采集数据的个数，范围为 0-2000，Utility 中默认为最大值 2000。
CurCount：开始采集运动数据后，会返回当前已采集的数据个数。
Status：显示当前采集的状态：
0：Ready：尚未启动运动数据采集功能；
1：Wait Trigger：已启动运动数据采集功能，但条件尚未触发；
2：Started：正在进行运动数据采集；

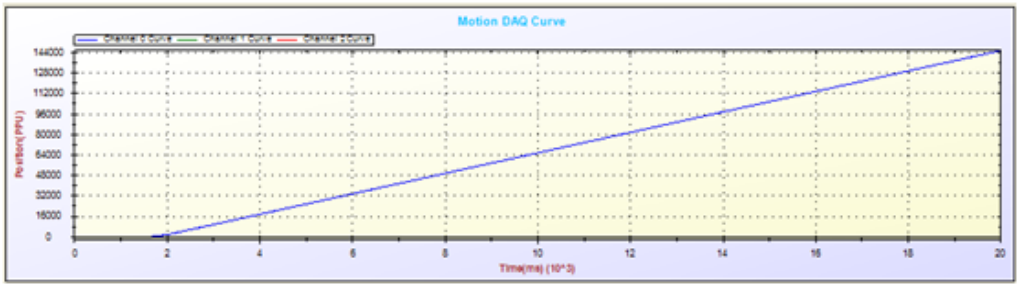
鼠标离开编辑框后，设置值将生效。可在 DataGridView 查看目前各个通道的配置情况，如下图：



2. 功能操作
MDAQ Start：启动运动数据采集功能。当满足触发条件后，开始采集运动数据；
MDAQ Stop：停止运动数据采集。
Clear：清除各个 Channel 的曲线图。
3. 图形显示
当 MDAQ Start 开始后，若满足触发条件，则开始采集数据，可查看当前各个通道采样点数和目前的状态，如下图示：



当数据采集完后，将在下面的图形框中画出各个通道的相应采集数据的曲线图，如下图所示：



4. Display
右上角的 Display 区域用于配置各个通道曲线的颜色和图形框纵坐标最大值。从下列框中选中相应的颜色，切换后，对应通道曲线的颜色将会随着变化。

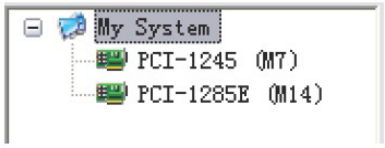
5.2.2.7 Hide Tree

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”；

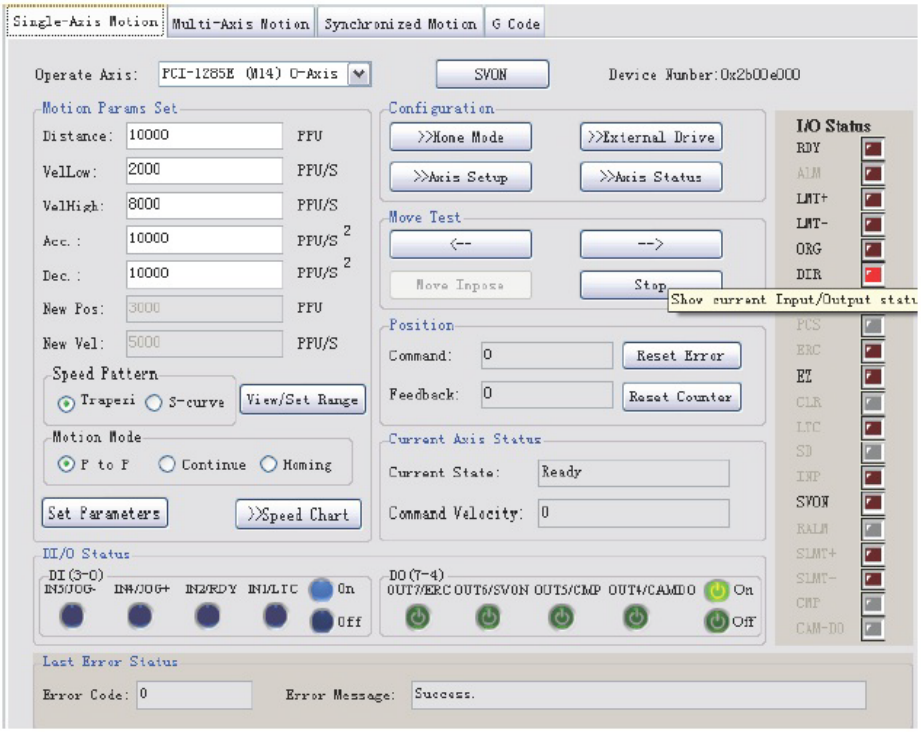
若 Device Tree 已隐藏，则点击将显示 Device Tree，按钮上的文字变为 “Hide Tree”。

5.2.3 设备树



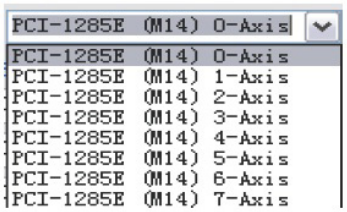
单击设备树中的任一设备，即可看到操作界面。

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

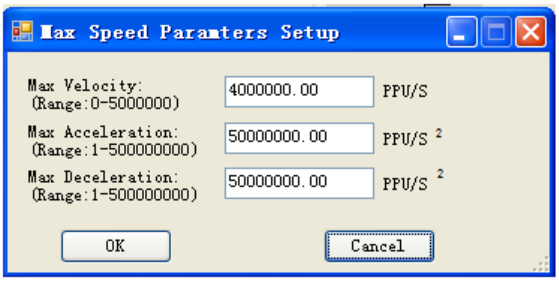
主要包括以下设置：点对点运动距离 （Distance）、单轴运动的初速度 （VelLow）、运行速度 （VelHigh）、加速度 （Acc.）、减速度 （Dec.）及叠加运动 （Move Impose）的运动距离 （New Pos.）和运动速度 （New Vel）。

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。



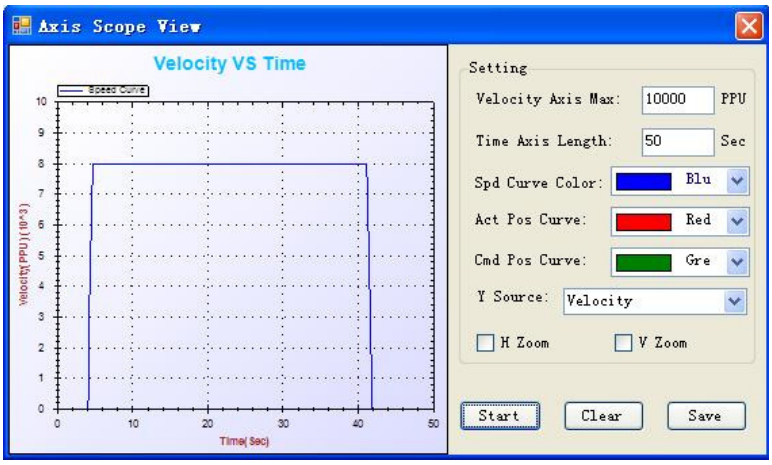
注！ *单轴运动的 VelHigh 不能高于 Max Velocity； Acc. 不能高于 Max Acceleration 且 Dec. 不能高于 Max Deceleration。*

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。

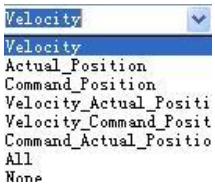


其中，右边为设置栏和操作按钮项，左边为单轴运动的运动 / 速度曲线图。

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

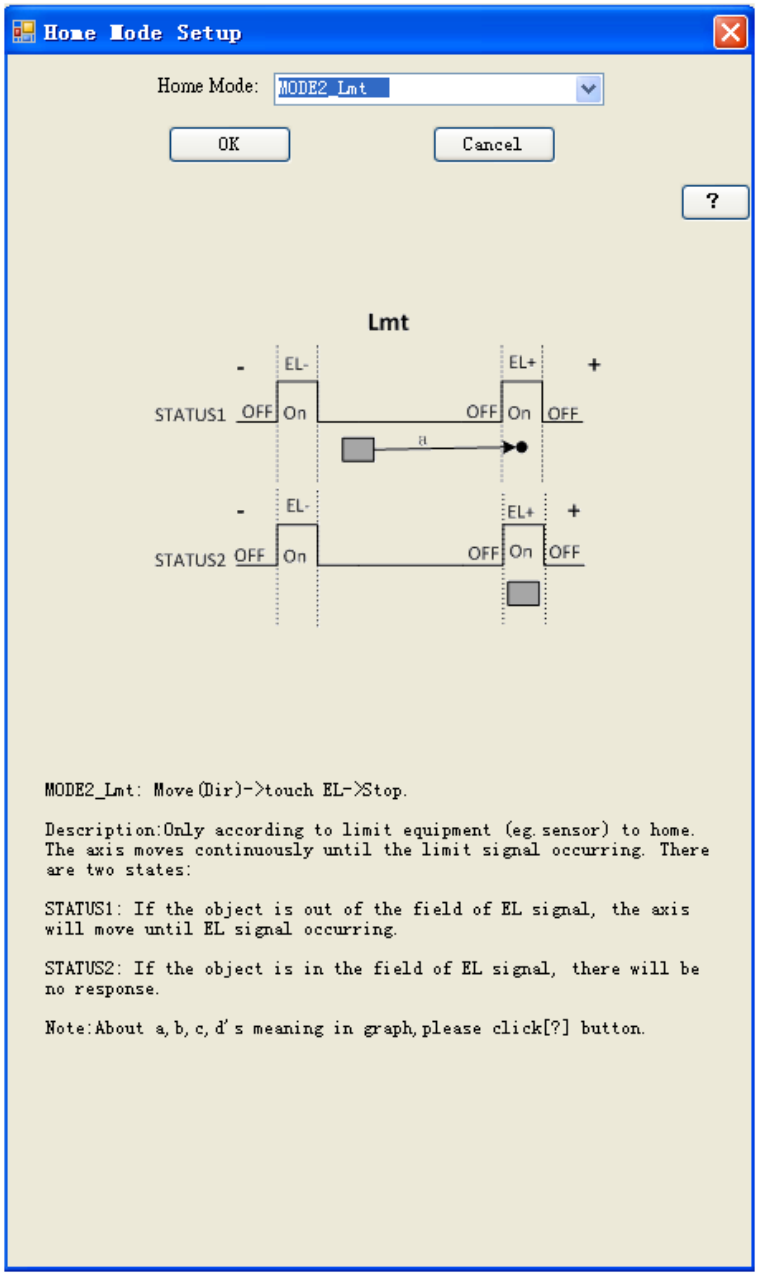
包括轴的回 Home 模式 （Home Mode）、外部驱动 (External Drive) 模式、属性配置和 I/O 状态显示。

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。

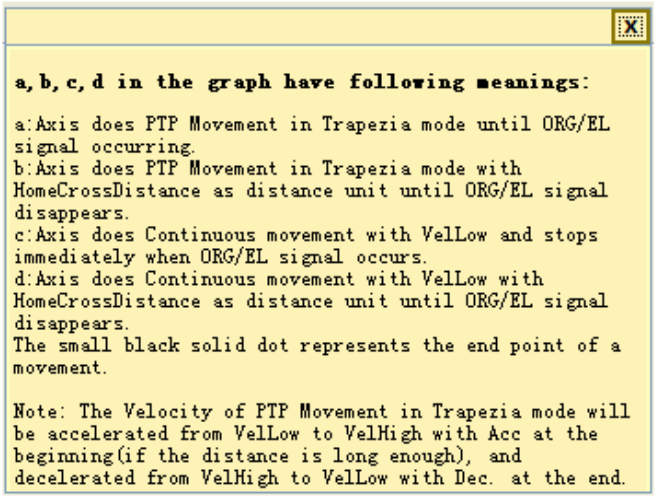
有关详细信息，请参考 PCI-1240 中有关 “Home Mode” 的描述信息或通用 API 编程指南中的 Acm_AxHome 函数。

单击 [Home Mode] 将出现如下对话框：



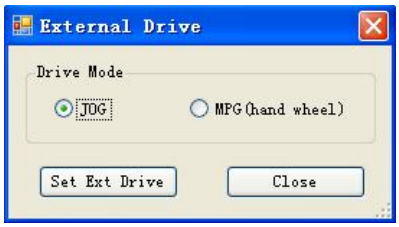
用户可从下拉列表框中选择任一模式，下面有相应的图解说明。用户可单击 [OK] 选择 “Home Mode” 下拉列表框中的一种模式，或单击 [Cancle] 取消操作。默认设置为 "Mode1_Abs"。

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。

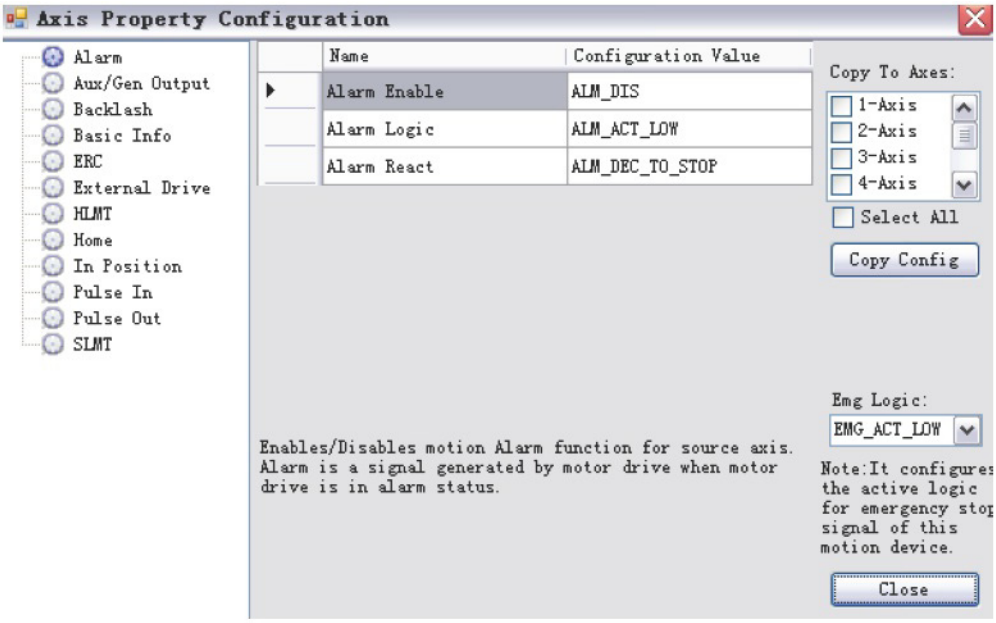


选择 “JOG” 或 “MPG” 并单击 [Set Ext Drive]，外部驱动模式将设置成功，用户即可操作外部驱动。单击 [Close] 将关闭对话框，外部驱动设置为 “Disable”。

注！ *对于 PCI-1285/1285E series 系列产品，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。
Cam DO	CamDO Enable	启用 / 禁用源轴的 CAM DO 功能。
	CamDO Logic	设置 CAM DO 信号的有效逻辑电平。
	CamDO Compare Source	设置 CAM DO 信号的比较源。
	CamDO Mode	设置 CAM DO 信号的模式。
	CamDO Direction	设置 CAM DO 的方向。
	CamDO Low Limit	设置 CAM DO 信号的低限位。
	CamDO High Limit	设置 CAM DO 信号的高限位。

Comparator	Compare Enable	启用 / 禁用源轴的轴比较器。
	Compare Source	设置比较器的源。
	Compare Method	设置比较器的方法。
	Compare Pulse Mode	设置比较器的脉冲模式。
	Compare Pulse Logic	设置比较器脉冲的有效逻辑电平。
	Compare Pulse Width	设置比较器的脉冲宽度。
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。
Latch	Latch Enable	启用 / 禁用源轴的锁存功能。
	Latch Logic	设置锁存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。
SD	SD Enable	启用 / 禁用源轴的 SD 信号。
	SD Logic	设置 SD 信号的有效逻辑电平。
	SD React	设置 SD 信号的反应模式。
	SD Latch	设置 SD 信号的 Latch 控制。
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。
Vibration	Vibration Enable	启用 / 禁用抑制源轴机械系统的振动。
	Vibration Reverse Time	设置源轴的振动抑制时间。该功能主要是通过完成命令运动后马上先负向输出单个脉冲，然后正向输出单个脉冲来实现抑制机械系统的振动。
	Vibration Forward Time	设置源轴的振动抑制时间。

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-1245/65，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。*

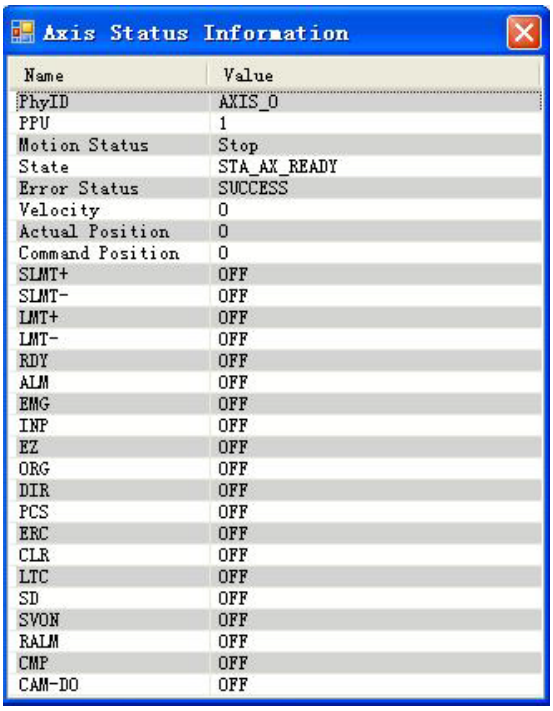
设置项目编辑完成后，属性值将在鼠标离开编辑框时生效 （已经在设备中设置）。

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。

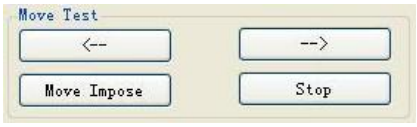
5.3.4.4 Axis Status

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：

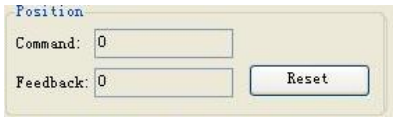


选择运动模式后，单击 [<--] 或 [-->]，轴将进行正向 / 反向的点对点 / 连续 / 返回原点运动。

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。

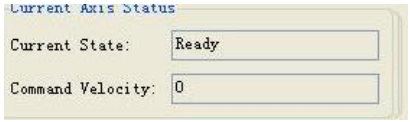
5.3.6 Position



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”。

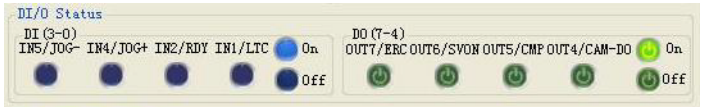
5.3.7 Current Axis Status



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”。

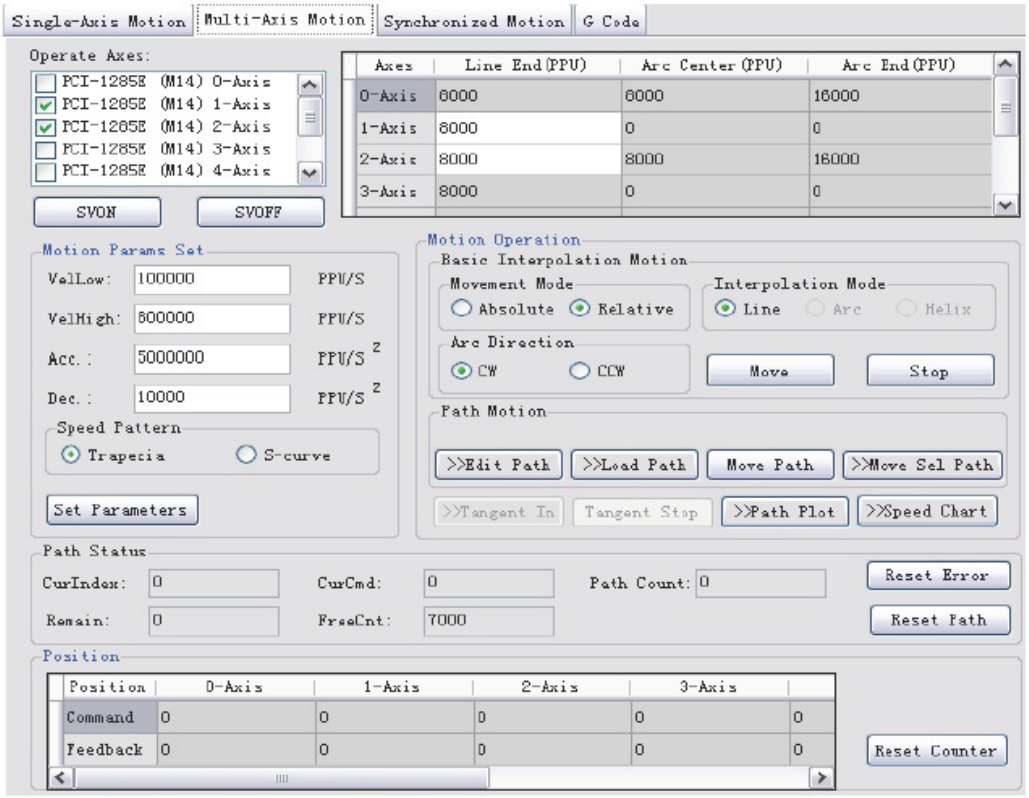
5.3.10 I/O Status

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

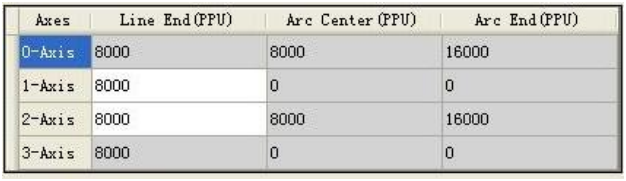
窗口中的列表框列出来所选设备的全部轴，勾选对应轴，将其添加到群组中。如果添加到群组的轴的数量小于 2，则群组状态将为 “Disable”。如果添加到群组的轴的数量大于或等于 2，群组状态将为 “Ready”，用户配置好适当参数后，即可进行适当插补操作。

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

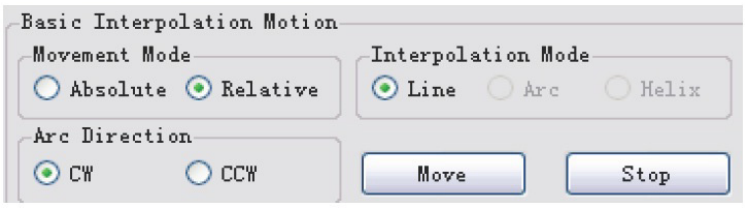
点击 [SVON]，会将 Group 中所有轴的伺服开启。

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）、圆弧插补 （Arc）以及螺旋插补 （Helix），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 Arc Direction

CW 表示顺时针；

CCW 表示逆时针。

5.4.4.3.3 Interpolation Mode

Line：直线插补。

Arc：圆弧插补。

Helix：螺旋插补。

5.4.4.3.4 Move

相应配置完成后，单击 [Move]，群组将执行指定插补运动。

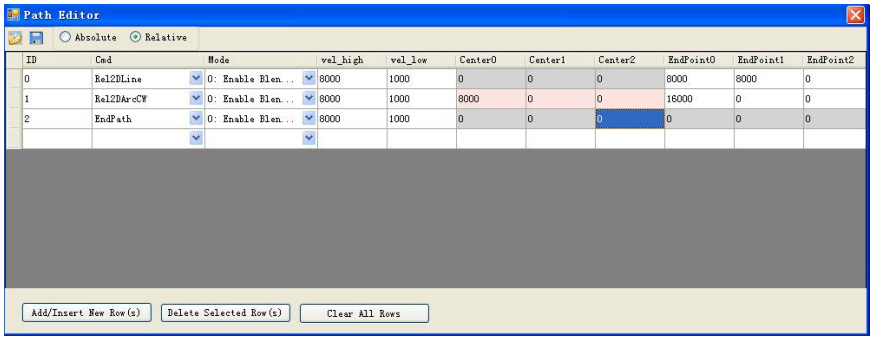
5.4.4.3.5 Stop

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 Path Motion

5.4.4.4.1 Edit Path

单击 [Edit Path]，将出现以下对话框：



其中，顶部的工具栏包括“Open File”、“Save File”和“Movement Mode”。

1. Open File ：打开相应路径下的 Path 文件，可以是一个二进制文件（.bin）或一个逗号分隔值文件（.CSV）。
2. Save File ：将编辑数据保存到路径文件，可以是一个二进制文件（.bin）或一个逗号分隔值文件（.CSV）。CSV 文件可以由 Excel 打开，方便事后查看修改。但是如果用户想要通过 [Load Path] 运行 Path，由于目前设备只支持通过 [Load Path] 导入 .bin 文件，因此必须将数据保存为 .bin 格式。
3. Movement mode：“Absolute”或“Relative”。如果用户选择“Absolute”，则 Cmd 栏中列出的命令将为绝对运动相关的命令；同样，如果用户选择“Relative”，则 Cmd 栏中列出的命令将为相对运动相关的命令。
4. Path 的编辑项目包括命令（Cmd）、运动模式（Blending/No Blending）、运动速度（vel_high）、初始速度（vel_low）、圆心（Center）和终点（EndPoint）。其中，默认的圆弧插补有 2 个轴（Center0/Center1）。EndPoint 的数量由所选设备插补支持的轴的最大数量决定，如 PCI-1285 最大支持 2 轴 Direct 直线插补运动因此有 EndPoint0、EndPoint1。
5. Add/Insert New Row(s)：单击之后，会出现如下对话框，用户可编辑添加 / 插入的行的数量。

单击 [OK]，相应数量的行将会添加到所选行之后 / 插入到所选行中。

6. Delete Selected Row(s)：删除所选行。
7. Clear All Rows：清除所有行。

5.4.4.4.2 Load Path

单击 [Load Path]，如果群组状态为“Ready”，用户可以从“Open File”对话框将所选 Path 文件（.bin 格式）导入到设备。

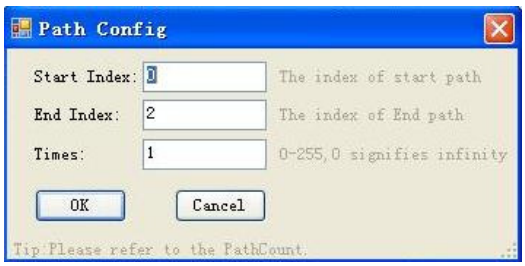
5.4.4.4.3 Move Path

加载 Path 之后，如果编辑的 Path 正确，Path 将按照序列号一个接一个运行。用户可以通过 [Path Status]、[Path Plot] 和 [Speed Chart] 观察运动曲线和相应状态。

5.4.4.4.4 Move Sel Path

从导入的 Path 中选择路径进行连续插补运动。

加载 Path 之后，单击 [Move Sel Path] 弹出如下对话框：



1. Start Index：选择 Path 的起始序列号。
2. End Index：选择 Path 的终止序列号。
3. Times：执行次数。值的范围为 0 到 255。如果设置为 0，表示这是一个无限循环，直到用户单击 “Stop” 终止循环。

5.4.4.4.5 Speed Forward

将使能 Group 的速度前瞻功能。有关详细信息，请参考编程指南属性列表中列出的群组中有关 CFG_GpSFEnable 的描述信息。

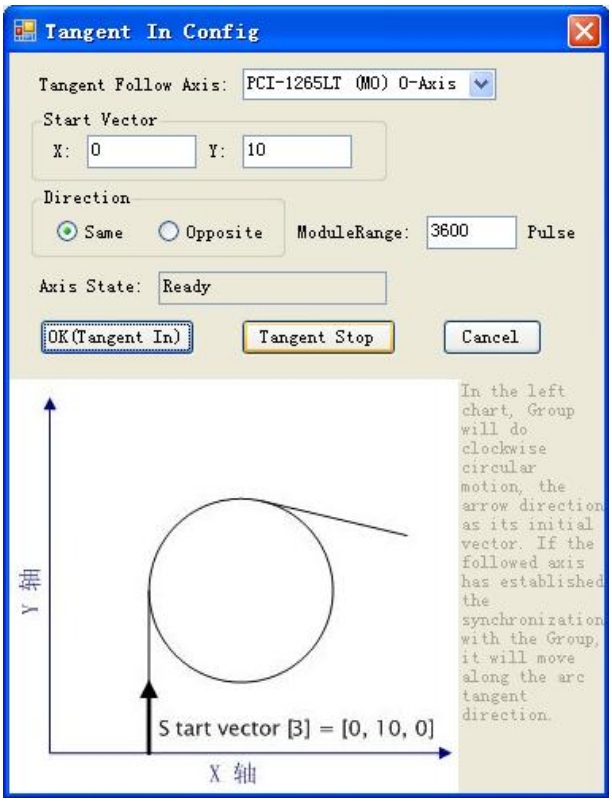
5.4.4.4.6 Blending Time

有关详细信息，请参考编程指南属性列表中列出的群组中有关 CFG_GpBldTime 的描述信息。

5.4.4.5 切向跟随运动

5.4.4.5.1 Tangent In

单击 [Tangent In] 将出现如下对话框。



用户需配置以下参数：

1. Tangent Follow Axis：选择切向跟随轴。由于该轴不能为群组中添加的轴，因此下拉列表框中不包括群组中添加的轴。
2. Start Vector：切向跟随运动的起始向量。在该测试工具中，默认参考面为 X-Y，用户只需要设置 X 向量和 Y 向量，无需编辑 Z 轴。
3. Direction：运动中切向跟随轴的方向，可以和群组运动方向相同或相反。
4. ModuleRange：切向跟随轴的转距 （即跟随轴旋转一周 （360 度）的脉冲数）。

配置下面有相关示意图和描述信息。

单击 [OK (Tangent In)]，切向跟随轴将与群组建立切向跟随同步关系。

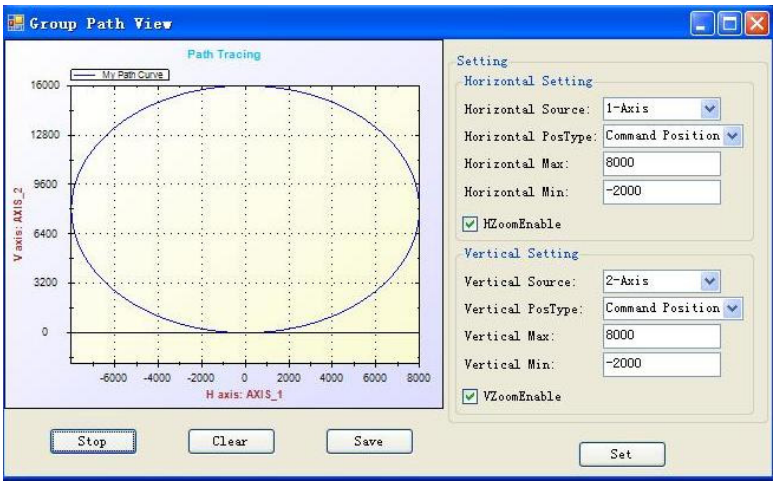
之后，如果群组进行插补运动，跟随轴将沿着插补运动的切向方向运动。如果切向跟随轴已经建立和群组的切向跟随同步运动，再次单击 [Tangent In]，对话框中的参数值将为配置值，用户可单击 [Tangent Stop] 解除同步关系。单击 [Cancel]，将不做任何修改同时关闭对话框。PCI-1285E 暂不支持切向跟随功能。

5.4.4.5.2 Tangent Stop

单击 [Tangent Stop] 解除切向跟随轴和群组之间的同步关系。PCI-1285E 暂不支持切向跟随功能。

5.4.4.6 Path Plot

显示群组的运动曲线。 单击 [Path Plot]，将出现以下对话框：



5.4.4.6.1 Setting

设置水平和垂直坐标。

1. Horizontal setting
   a. Horizontal Source：水平数据源，默认为 Group 的第一轴 （按照添加顺序排序），可以选择 Group 中的任一轴。
   b. Horizontal PosType：水平位置类型，用户可选择命令或反馈位置。
   c. Horizontal Max：水平最大坐标。
   d. Horizontal Min：水平最小坐标。
2. Vertical setting：与水平设置相同。

5.4.4.6.2 Set

单击 [Set] 使设置生效。

5.4.4.6.3 Start

单击 [Start]，图形框即开始绘制曲线。 如果群组在运动中，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” 。

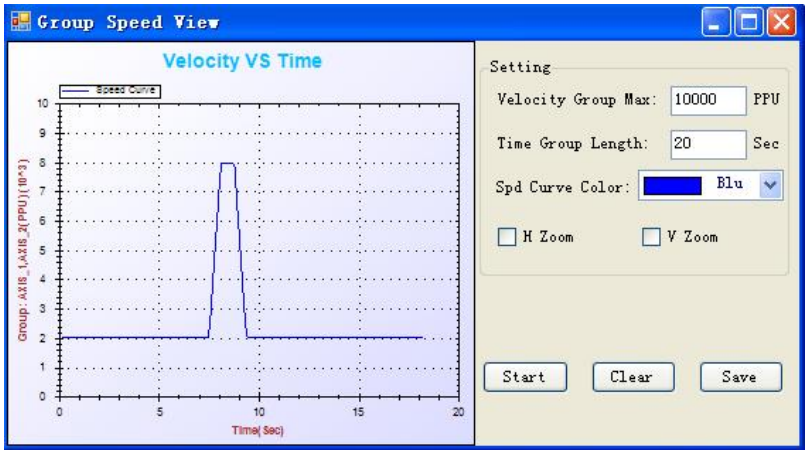
5.4.4.6.4 Clear

单击 [Clear] 将清除图形框中当前显示的曲线图。

5.4.4.6.5 Save

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

5.4.4.7 Speed Chart



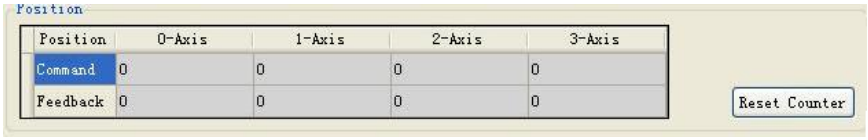
设置和操作与 “Single Axis Motion” 中的 [Speed Chart] 类似。

5.4.5 Path Status

显示路径状态。

CurIndex：当前正在运行的路径的序号。
CurCmd：当前正在运行的路径的命令码。
Remain：未执行的路径号。
FreeCnt：路径缓冲中的剩余空间。
Path Count：加载路径文件的总路径数。

5.4.6 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。

5.4.7 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。

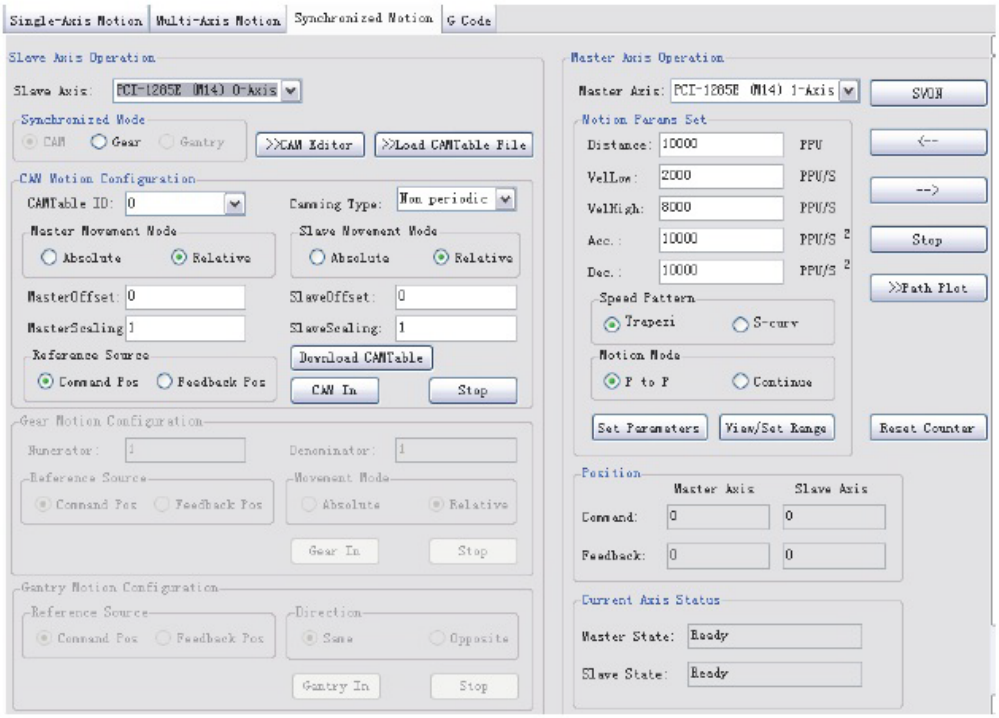
Last Error Status：显示上一个错误信息。

Axis Name：发生错误的轴。

Error Code：错误代码。

Error Message：具体的错误信息。

5.5 Synchronized Motion



5.5.1 Slave Axis Operation

5.5.1.1 Slave Axis

选择设备的其中一个轴作为从轴。

注！ *主轴和从轴不能为同一轴。默认的从轴为所选设备的 0 轴。*

5.5.1.2 Synchronized Mode

选择同步模式：“CAM”、“Gear” 和 “Gantry”。在配置和建立同步运动之前，用户必须先选择同步模式。

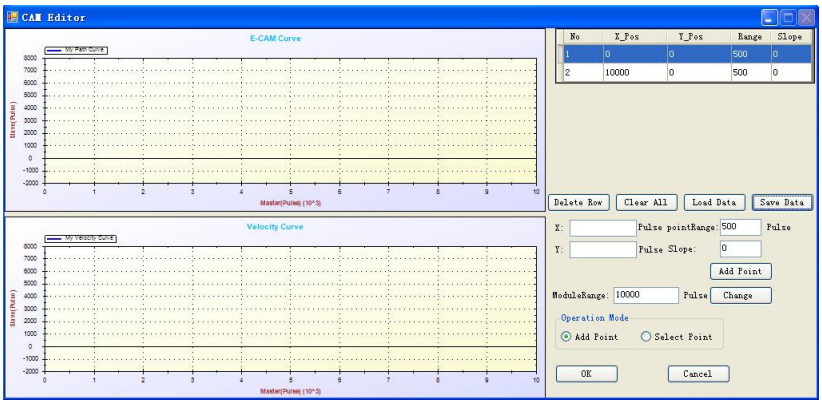
5.5.1.3 CAM Motion

在 “Synchronized Mode” 选择 “CAM”，然后用户即可进行 CAM 设置和操作。

注！ *1285E 暂不支持该功能，Utility 以灰阶显示。*

5.5.1.4 CAM Editor

单击 [CAM Editor]，将出现以下对话框：



左上角为 E-CAM 曲线图形框；左下角为速度曲线图形框；右上角为 CAM 表；右下角为操作面板。

1. Operation Mode

   a. Add Point：用户可直接在 E-CAM 曲线上添加 CAM 点。无论何时用户添加了一个点，CAM 曲线都将重新绘制。其中，CAM 点由一个红色小实线圈表示，而 CAM 曲线由蓝色曲线表示。 在这种操作模式中，鼠标的形状为十字。当首次打开对话框或者还未编辑 CAM 曲线，操作模式默认为 “Add Point” 模式。

   b. Select Point：用户可以选择相应的 CAM 点进行拖拉运动。同时，CAM 曲线也将随之改变。 在这种操作模式中，鼠标的形状为箭头。当再次打开对话框或者已经编辑 CAM 曲线，操作模式默认为 “Select Point” 模式。

2. CAM 表

   通过编辑 CAM 点形成的 CAM 表显示在对话框的右上角。值得注意的是，CAM 表中第一行的 X_Pos 和 Y_Pos，即首个 CAM 点，必须为 0，这表示主轴的起始位置为 0，从轴的起始位置为 0。CAM 表中最后一行的 X_Pos 和 Y_Pos，即最后一个 CAM 点，必须为 （ModuleRange，0），这表示主轴旋转了一圈，从轴返回至起始位置 0。

   a. No：CAM 点的序号。

   b. X_Pos：水平位置坐标 （主轴位置）。

   c. Y_Pos：垂直位置坐标 （从轴位置）。

   d. Range：参考点和 CAM 点之间的距离。有关详细信息，请参考通用 API 编程指南中的 Acm_DevDownloadCAMTable 函数的说明信息。默认值为 pointRange 的编辑值。用户可通过编辑改变该值。当用户添加多个 CAM 点时，pointRange 也将随之改变。如果用户想要改变编辑 CAM 点的 pointRange，可直接在 CAM 表中进行修改。

   e. Slope：CAM 点的两个参考点之间的斜率。有关详细信息，请参考通用 API 编程指南中的 Acm_DevDownloadCAMTable 函数中对 PointSlope 的说明信息。默认值为 “Slope” 编辑框中的值，用户可通过编辑修改该值。后续添加的 CAM 点的斜率将

为编辑框中修改后的值。如果用户想要改变编辑 CAM 点的 Slope，可直接在 CAM 表中进行修改。

注！ *斜率值的范围为 -10 到 10。如果值小于 -10，则默认为 -10。如果值大于 10，则默认为 10。*

3. CAM 表操作
   a. Delete Row：删除所选行。
   b. Clear All：清除所有 CAM 点 （除起始点和终结点）。
   c. Load Data： 插入所选 CAM 表文件。文件格式可以为二进制文件 （.bin）或 Excel 可读的 .csv 文件。
   d. Save Data: 保存 CAM 表。文件格式可以为二进制文件 （.bin）或 Excel 可读的 .csv 文件。但是如果用户想要通过 [Load CAMTable File] 操作导入 CAM 表，则需要将 CAM 表保存为 .bin 格式，因为目前设备仅支持通过 [Load CAMTable File] 导入 bin 文件。
4. Add Point
   添加 CAM 点时，用户需要编辑右下角的 “X_Pos”、“Y_Pos”、“pointRange” 和 “Slope”，然后单击 [Add Point] 添加点。
5. Change (ModuleRange)
   主轴的转矩 （ModuleRange）默认设置为 10000。如果用户想要编辑，可在 ModuleRange 框中进行编辑，然后单击 [Change] 完成。修改之后，E-CAM 曲线和速度曲线的水平最大坐标将为修改值。如果修改值之前编辑过 CAM 点，则 CAM 点的 X_Pos 和 pointRange 将变为 ModuleRange （修后后）/Pre_ModuleRange （修改前）倍。
6. OK
   单击 [OK] 保存 CAM 表。用户可通过 [Download CAMTable] 将 CAM 表将入硬件。
7. Cancel
   单击 [Cancel] 放弃编辑。

5.5.1.4.1 Load CAMTable File

单击 [Load CAMTable File] 选择二进制文件，CAM 表将导入硬件。

注！ *保存之前，用户必须首先设置 “CAMTableID”。 值为 0 或 1。完成该步后，在解除同步关系之前 CAMTableID 将不能修改。*

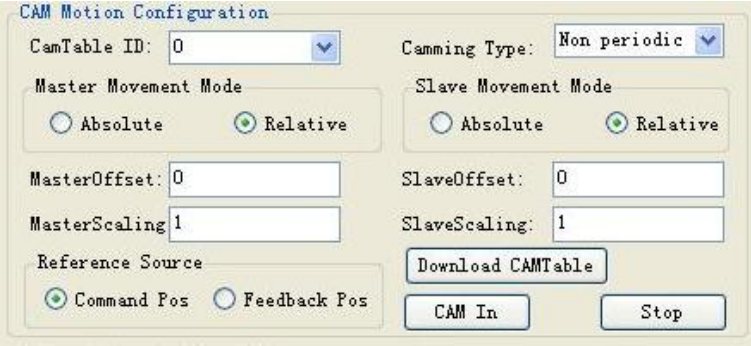
5.5.1.4.2 Download CAMTable

如果在 CAM Editor 中编辑过 CAM 表，用户可通过 [Download CAMTable] 将 CAM 表导入硬件。

注！ *保存之前，用户必须首先设置 “CAMTableID”。 值为 0 或 1。完成该步后，在解除同步关系之前 CAMTableID 将不能修改。*

5.5.1.4.3 Configuration

配置 CAM 运动并建立 CAM 同步。



建立 CAM 同步之前，用户必须配置以下参数：

1. Camming Type：
   a. Non periodic：非循环模式。 如果用户选择这种模式，当主轴运行完一个完整周期之后，从轴将不会按照 CAM 曲线跟随主轴运动。
   b. Periodic：循环模式。 如果用户选择这种模式，从轴将始终按照 CAM 运动中 CAM 曲线跟随主轴运动。
2. Master Movement Mode
   a. Absolute：如果用户选择这种模式，主轴的当前位置将被 CAM 曲线作为水平坐标的起始点。
   b. Relative：如果用户选择这种模式，主轴将以当前命令 / 实际位置作为起始点进行相对运动。
3. Slave Movement Mode
   a. Absolute：如果用户选择这种模式，从轴将从当前的命令 / 实际位置以设定的速度参数去追赶 CamTable 中对应的从轴的值。
   b. Relative：如果用户选择这种模式，从轴将以当前的命令 / 实际位置按照 CAM 曲线跟随主轴做相对运动。
4. Master Offset：相对于主轴的偏移值。
5. Slave Offset：相对于从轴的偏移值。
6. Master Scaling：主轴比率因数。CAM 曲线在水平方向的缩放。
7. Slave Scaling：从轴比率因数。CAM 曲线在垂直方向的缩放。
8. Reference Source：主轴的位置参考源。
   a. Command Position：参考源为理论位置。
   b. Feedback Position：参考源为反馈 （实际）位置。

5.5.1.4.4 CAM In

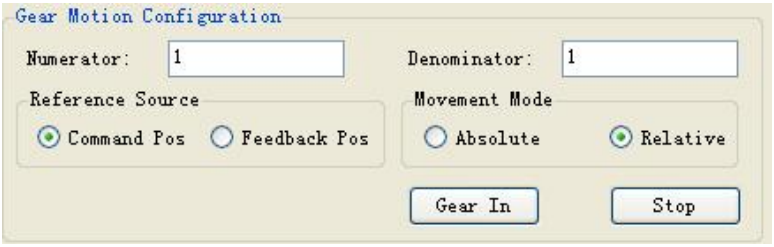
单击 [CAM In]，从轴将与主轴建立凸轮同步关系，从轴的状态将变为 “Synchronous Driving”。此后，如果主轴为 “P to P” 或 “Continue” 运动，从轴将按照 CAM 曲线和配置跟随主轴运动。

5.5.1.4.5 Stop

单击 [Stop] 解除同步关系。从轴的状态将恢复为 Ready。

5.5.1.5 Gear Motion

在 “Synchronized Mode” 中选择 “Gear”，用户即可配置和操作齿轮运动。



5.5.1.5.1 Configuration

建立齿轮同步之前，用户必须配置以下参数：

1. Numerator：齿轮比的分子。
2. Denominator：齿轮比的分母。
3. Reference Source：主轴的位置参考源。
   a .Command Position：参考源为理论位置。
   b .Feedback Position：参考源为反馈位置。
4. Movement Mode:
   a. Absolute：若设置为该模式，从轴将以设定的速度参数去追赶主轴的命令 / 实际位置，直至两者一致。
   b. Relative：若设置为该模式，从轴将与主轴保持初始时的位置差。

5.5.1.5.2 Gear In

单击 [Gear In]，从轴将与主轴建立齿轮同步关系，从轴的状态将变为 “Synchronous Driving”。此后，如果主轴为 “P to P” 或 “Continue” 运动，从轴将按照配置跟随主轴运动。

5.5.1.5.3 Stop

单击 [Stop] 解除同步关系，从轴状态将为 “Ready”。

5.5.1.6 Gantry Motion

在 “Synchronized Mode” 中选择 “Gantry ”，用户即可配置和操作龙门运动。

5.5.1.6.1 Configuration

建立龙门同步之前，用户必须配置以下参数：

1. Reference Source：主轴的位置参考源。
   a. Command Position：参考源为理论位置。
   b. Feedback Position：参考源为反馈位置。
2. Direction：从轴相对于主轴的运动方向。
   a. Same：与主轴相同。
   b. Opposite：与主轴相反。

5.5.1.6.2 Gantry In

单击 [Gantry In]，从轴将与主轴建立龙门同步关系，从轴的状态将变为 “Synchronous Driving”。此后，如果主轴为 “P to P” 或 “Continue” 运动，从轴将按照配置跟随主轴运动。

5.5.1.6.3 Stop

单击 [Stop] 解除同步关系，从轴状态将为 “Ready”。

5.5.2 Master Axis Operation

5.5.2.1 Master Axis

从所选设备的所有轴中选择一个轴作为主轴。

注！ 主轴和从轴不能为同一轴。默认的主轴为所选设备的 1 轴。

5.5.2.2 Motion Params Set

设置步骤与 “Single-axis Motion” -> “Motion Params Set” 相同。

5.5.2.3 Operation

5.5.2.3.1 SVON

单击 [SVON]，主轴和从轴的伺服将开启，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”。

5.5.2.3.2 其它操作

其它操作，请参考 “Single-axis Motion” -> “Move Test”。值得注意的是，同步关系建立之后，从轴将跟随主轴运动。 用户可通过 [Path Plot] 查看运动曲线。

5.5.2.3.3 Position

显示主轴和从轴的当前命令 （理论）位置和反馈 （实际）位置。

单击 [Reset Counter] 将计数复位至 0。

5.5.2.4 State

用户可通过状态栏查看主轴和从轴的当前状态。有关详细信息，请参考通用 API 编程指南中的 Acm_AxGetState 函数中对 State 的说明信息。

5.6 Digital Input

主要显示设备的数字量输入端口的状态。

PCI-1265 有 8 个 DI。

如上图所示，从右到左分别为数字量输入 DI0 ~ DI7。其中， 表示 DI 有效 （On），bit 的值为 1； 表示 DI 无效 （Off），bit 的值为 0。 “Hex” 表示 8 个 DI 组成的字节的十六进制值。

PCI-1285/1285E 暂不支持该功能。

5.7 Digital Output

主要显示设备的数字量输出端口的状态，以及 DO 上的相应 ON/OFF 操作。

PCI-1265 有 8 个 DO。

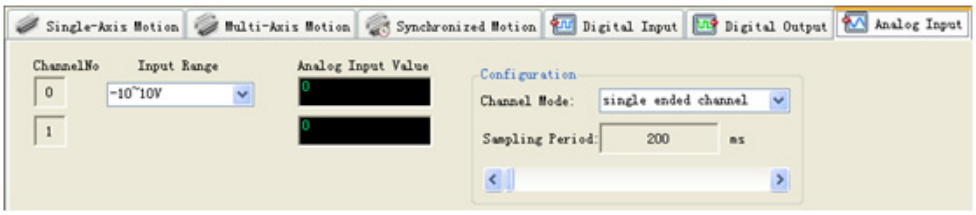
如上图所示，从右到左分别为数字量输出 DO0-DO7。其中， 表示 DO 有效（On），bit 的值为 1； 表示 DO 无效 （Off），bit 的值为 0。“Hex” 表示 8 个 DO 组成的字节的十六进制值。

PCI-1285/1285E 暂不支持该功能。

5.8 Analog Input

显示设备的模拟量输入通道的状态。

PCI-1265 有两个 AI 通道。



如上图所示，有如下几个参数：

Channel No：通道索引号。PCI-1265 有两个模拟量输入，通道索引为 0 和 1。

Input Range：模拟量输入范围。

1. Analog Input Value：根据采样周期，从输入通道采集的模拟量输入值。
2. Configuration
   a. Channel Mode：支持单端通道和差分通道。
   b. Sampling period：用户可通过滚动条修改该值，值的范围为 200 ~ 10000 ms。

注！ *PCI-1285/1285E 暂不支持模拟量输入功能。*

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-1285/1285E 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	FLAOT	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	FLAOT *	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。

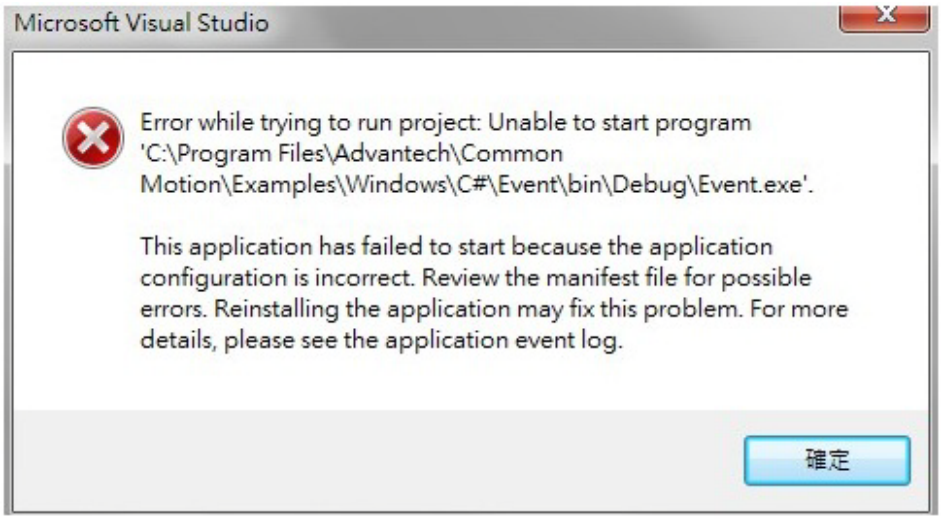
有七种事件类型：

事件名称	说明
EVT_AX_MOTION_DONE	当前运动完成时，触发事件。
EVT_AX_COMPARED	当满足比较条件时，触发事件。(PCI-1245/1245E/1265 不支持)
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。

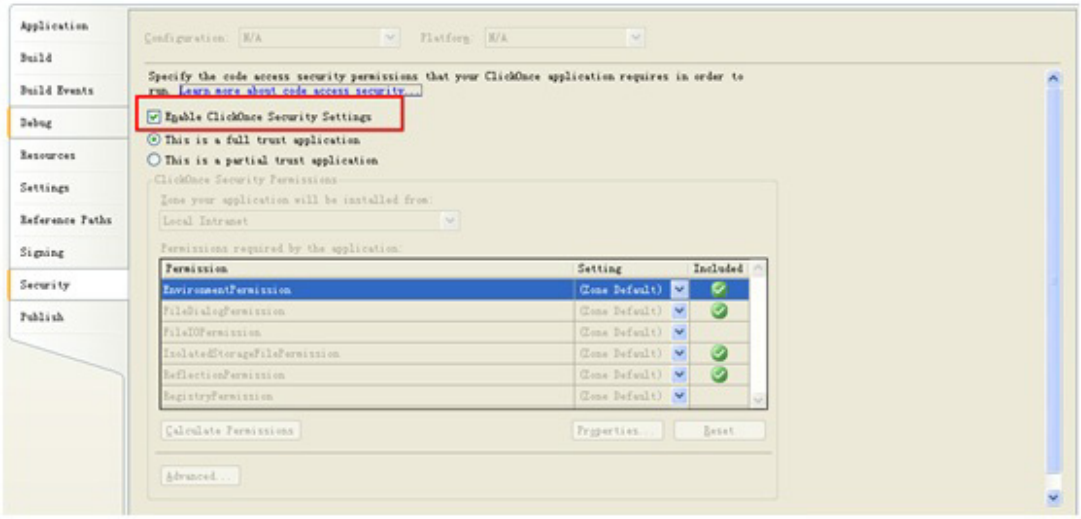
请参考 Acm_EnableMotionEvent 和 Acm_CheckMotionEvent。

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-1285/1285E 的软件架构

PCI-1285/1285E 软件架构基于通用运动架构，如下图所示：

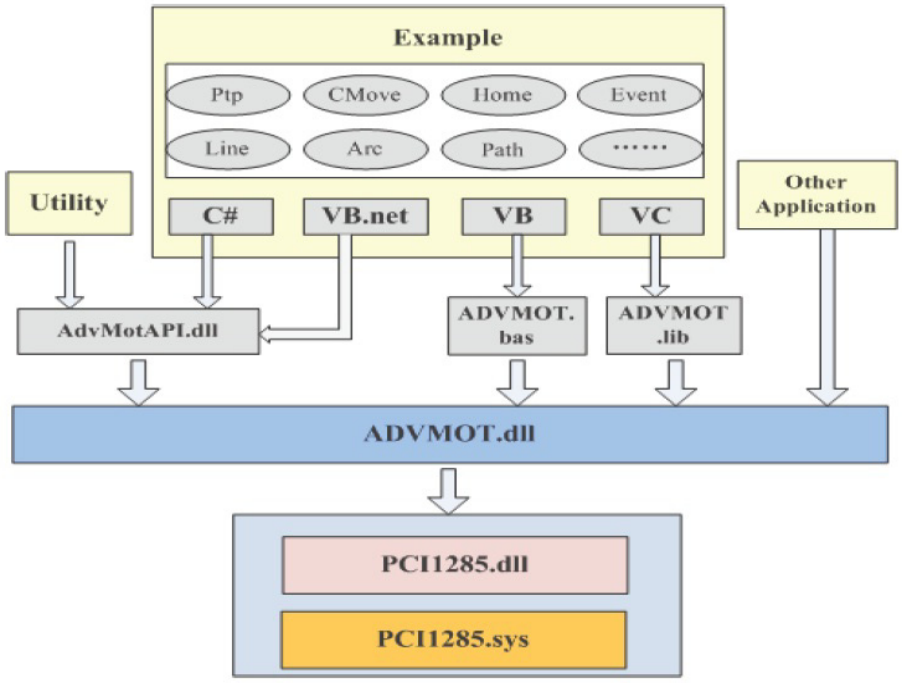


图 6.1： PCI-1285/1285E 的软件架构

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

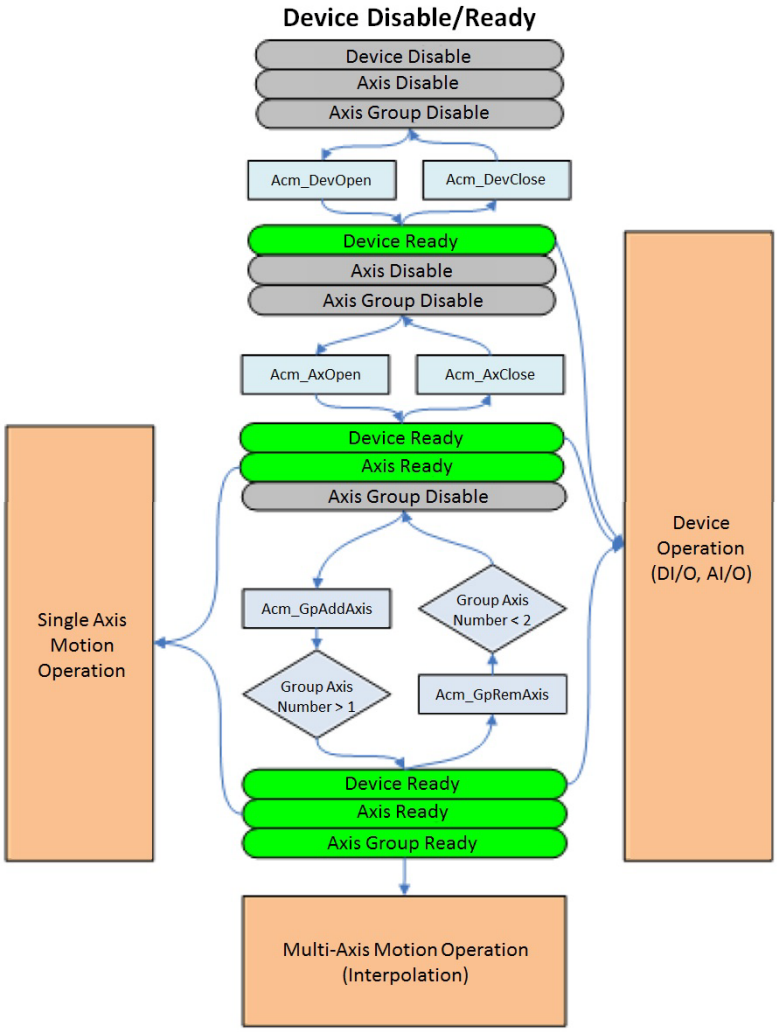


图 6.2： 基本操作流程图

6.2.2.2 单轴流程图

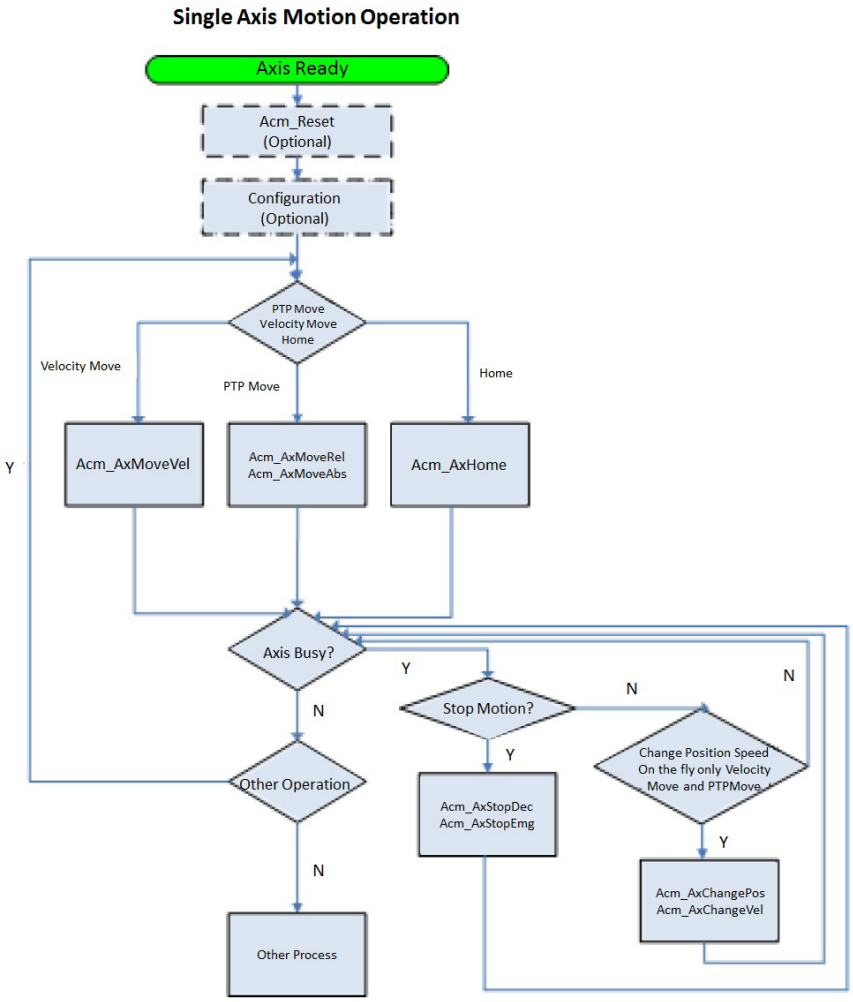


图 6.3： 单轴操作流程图

6.2.2.3 多轴流程图

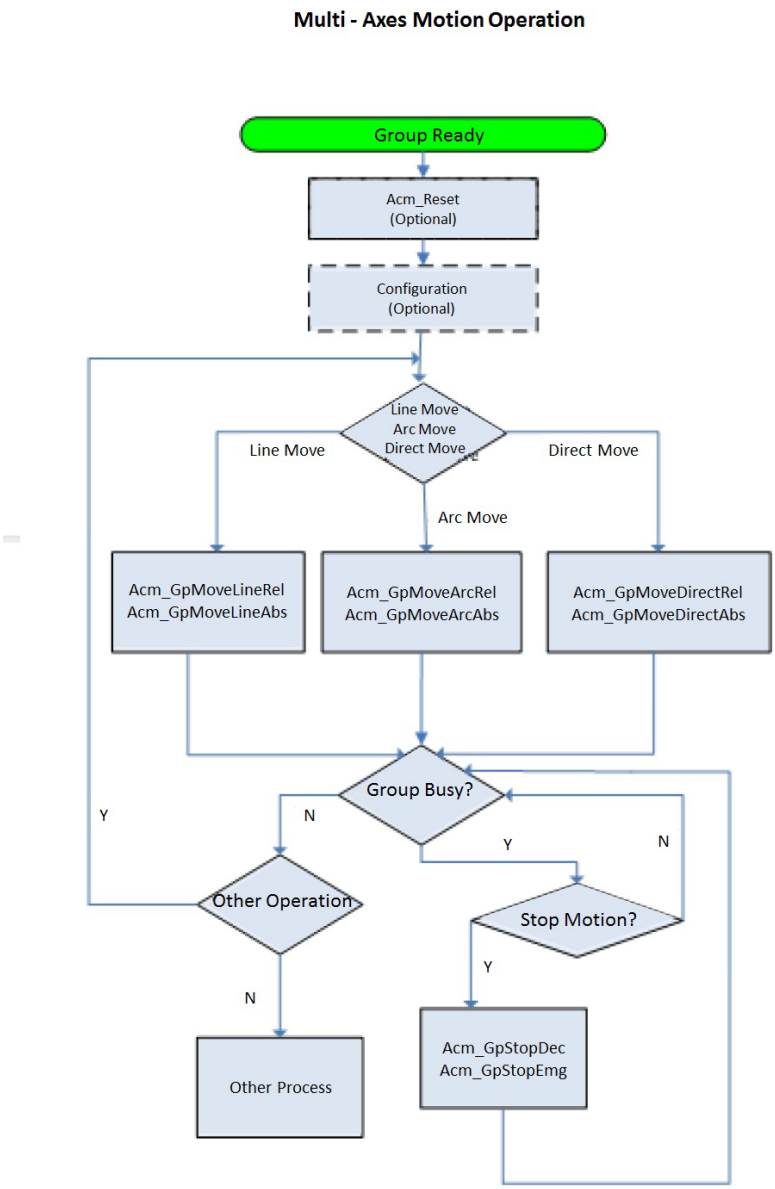


图 6.4： 多轴操作流程图

6.2.2.4 E-Cam 流程图

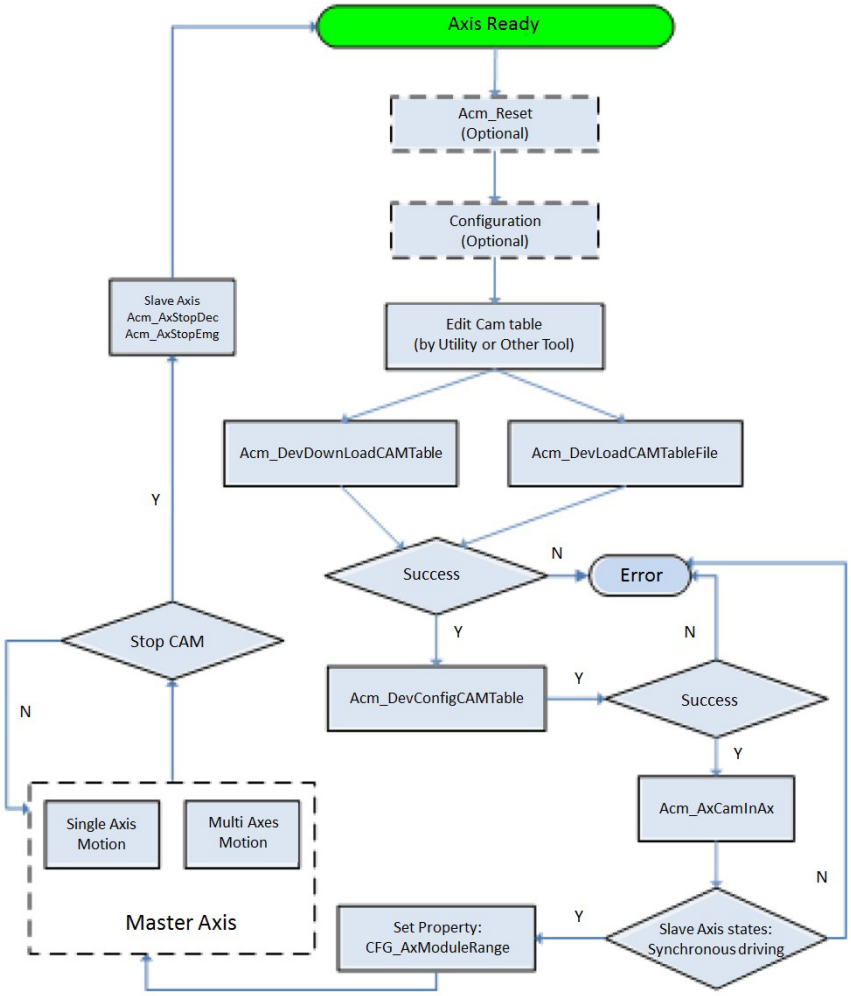


图 6.5： Cam 操作流程图

6.2.2.5 E-Gear/Gantry 流程图

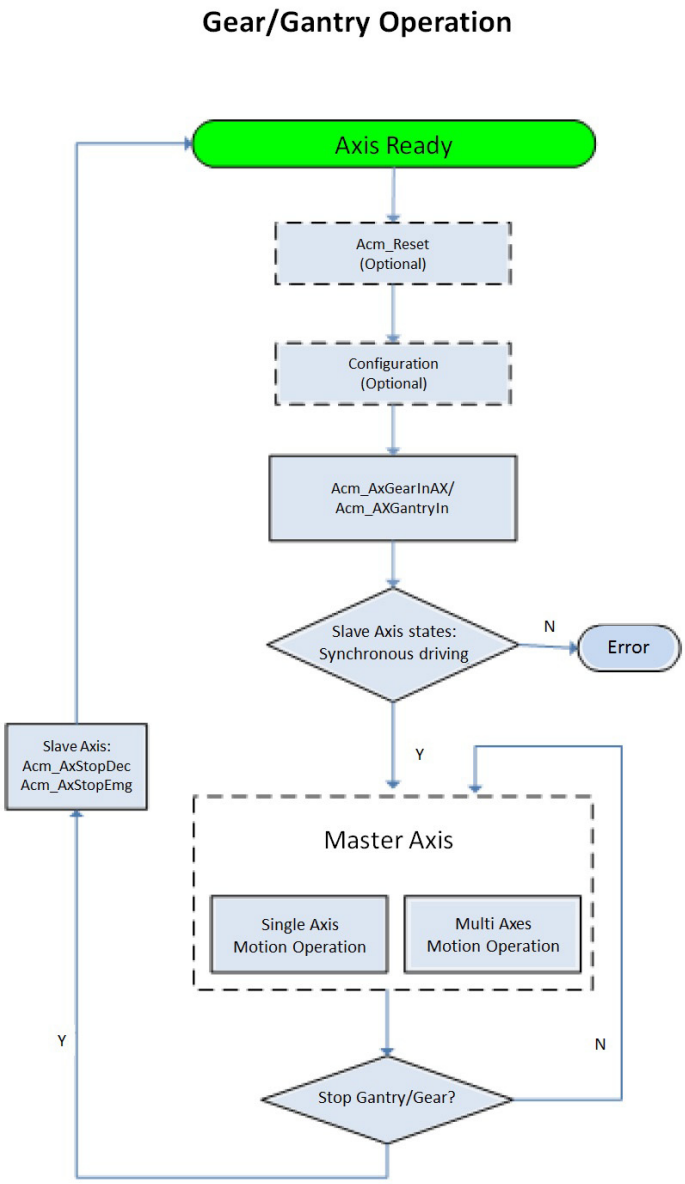


图 6.6： Gear/Gantry 操作流程图

6.2.2.6 切线跟随流程图

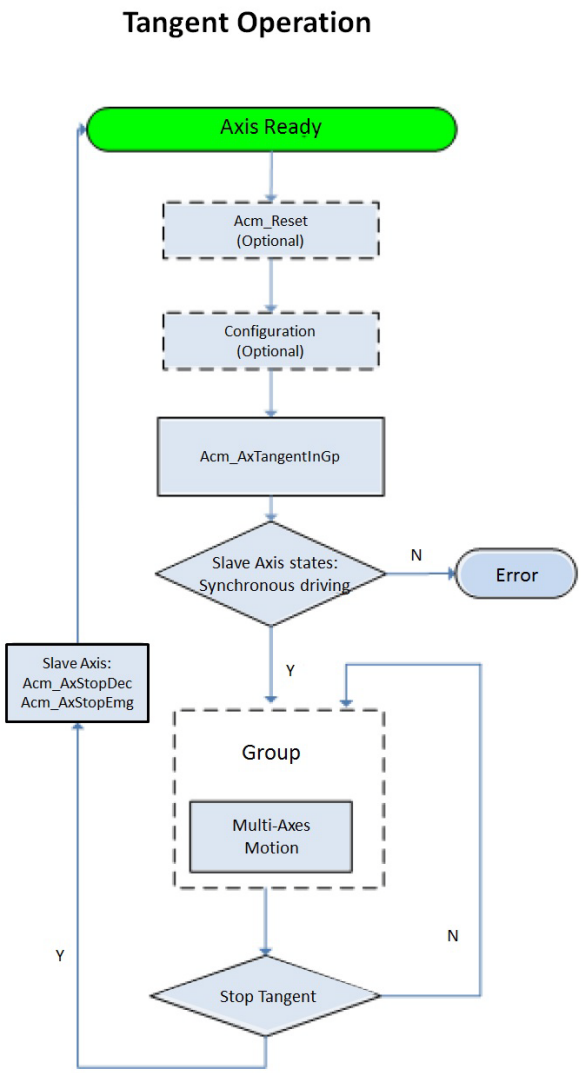


图 6.7： 切线跟随流程图

6.2.3 示例支持列表

示例	VC	C#	VB	VB .NET	BCB	LabView	说明
ARC	√	√		√	√	√	展示如何控制一个插补群组的圆弧运动。
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。
Compare	√	√		√	√	√	展示如何使用比较功能。
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
Path	√	√		√	√	√	展示如何使用一个插补群组的轨迹操作功能（即连续插补）。
PTP	√	√	√	√	√	√	展示如何控制单轴的点到点运动。
SetCard Relation	√	√		√	√	√	展示如何控制多个 PCI-1220 设备之间的关系。
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。
MoveImpose	√	√		√	√	√	展示如何使用叠加运动功能。
Latch	√	√		√	√	√	展示如何使用锁存功能。
Helix	√	√		√	√	√	展示如何控制一个插补群组的螺旋运动。
E-CAM	√	√		√	√	√	展示如何使用电子凸轮 （E-CAM）功能。
E-Gear	√	√		√	√	√	展示如何使用电子齿轮 （E-Gear）功能。
Tangent	√	√		√	√	√	展示如何使用切向跟随功能。
Gantry	√	√		√	√	√	展示如何使用龙门功能。
Device DIO	√	√		√	√	√	展示设备的数字量输入 / 输出功能。
Device AI							展示设备的模拟量输入功能。

6.2.4 PCI-1285/1285E 支持的 API 列表

类型		方法 / 事件	PCI-1285	PCI-1285E	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。
		Acm_DevDownloadCAMTable	X	X	加载 CamTable 中的数据。
		Acm_DevConfigCAMTable	X	X	配置 Cam。
		Acm_DevLoadCAMTableFile	X	X	加载 CamTable 文件。
		Acm_DevMDaqConfig	X	X	设定 MDaq 相关配置。
		Acm_DevMDaqGetConfig	X	X	获取设定的 MDaq 配置。
		Acm_DevMDaqStart	X	X	启动 MDaq 功能。
		Acm_DevMDaqStop	X	X	停止 MDaq 功能。
		Acm_DevMDaqReset	X	X	重置 MDaq 相关资料。
		Acm_DevMDaqGetStatus	X	X	获取当前 MDaq 状态。
		Acm_DevMDaqGetData	X	X	获取已记录的 MDaq 数据。
	事件	EVT_AX_MOTION_DONE	√	√	当前运动完成时，触发事件。
		EVT_AX_COMPARED	X	X	当满足比较条件时，触发事件。
		EVT_AX_LATCH	X	X	当 Latch 发生时，触发此事件。
		EVT_AX_ERROR	X	X	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。
DAQ	数字量输入 / 输出	Acm_DaqDiGetByte	X	X	以字节形式获取数字量输入值。
		Acm_DaqDiGetBit	X	X	以位形式获取数字量输入值。
		Acm_DaqDoSetByte	X	X	以字节形式设定数字量输出值。
		Acm_DaqDoSetBit	X	X	以位形式设定数字量输出值。
		Acm_DaqDoGetByte	X	X	以字节形式获取数字量输出值。
		Acm_DaqDoGetBit	X	X	以位形式获取数字量输出值。
	模拟量输入 / 输出	Acm_DaqAiGetRawData	X	X	获取模拟量输入值的二进制值。
		Acm_DaqAiGetVoltData	X	X	获取电压值。

	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群组动作的运行速度。
		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。
		Acm_AxMoveImpose	√	X	在当前运动上叠加新的运动。
	同时运动	Acm_AxSimStartSuspendAbs	√	X	暂停同步的绝对模式下的点到点运动。
		Acm_AxSimStartSuspendRel	√	X	暂停同步的相对点到点运动。
		Acm_AxSimStartSuspendVel	√	X	暂停同步的连续运动。
		Acm_AxSimStart	√	X	开始同步运动。
		Acm_AxSimStop	√	X	停止同步运动。
	返回原点	Acm_AxHome	√	√	命令回原点运动。

轴	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。
	比较	Acm_ AxSetCmpData	√	X	设置比较数据。
		Acm_AxSetCmpTable	√	X	设置比较数据表。
		Acm_AxSetCmpAuto	√	X	设置线性比较数据。
		Acm_AxGetCmpData	√	X	获取当前比较数据。
	锁存	Acm_AxGetLatchData	√	X	获取锁存数据。
		Acm_AxTriggerLatch	√	X	触发锁存数据。
		Acm_AxResetLatch	√	X	复位锁存信息。
		Acm_AxGetLatchFlag	√	X	获取锁存标记。
	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。
	应用	Acm_AxCamInAx	X	X	命令 E-Cam。
		Acm_AxGearInAx	√	√	命令 E-Gear。
		Acm_AxGantryInAx	√	X	命令 Gantry。
		Acm_AxPhaseAx	√	√	在电子凸轮或电子齿轮过程中使从轴进行相位超前或落后动。
		Acm_AxTangentInGp	√	X	命令切线跟随功能。
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_GpChangeVel	√	√	命令群组在插补运动时改变速度。
		Acm_GpChangeVelByRate	√	X	按照设定的比例改变当前正在执行的群组动作的运行速度。
		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。

群组	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_GpMoveCircularRel	√	X	命令相对圆弧插补。
		Acm_GpMoveCircularAbs	√	X	命令绝对圆弧插补。
		Acm_GpMoveCircularRel_3P	√	X	命令相对三点圆弧插补。
		Acm_GpMoveCircularAbs_3P	√	X	命令绝对三点圆弧插补。
		Acm_GpMoveCircularRel_Angle	√	X	通过相对圆心坐标，旋转角度以及方向进行圆弧插补。
		Acm_GpMoveCircularAbs_Angle	√	X	通过绝对圆心坐标，旋转角度以及方向进行圆弧插补。
		Acm_GpMoveDirectAbs	√	√	命令绝对直接线性插补。
		Acm_GpMoveDirectRel	√	√	命令相对直接线性插补。
		Acm_GpMoveHelixRel	√	X	命令相对螺旋插补。
		Acm_GpMoveHelixAbs	√	X	命令绝对螺旋插补。
		Acm_GpMoveHelixRel_3P	√	X	命令相对三点螺旋插补。
		Acm_GpMoveHelixAbs_3P	√	X	命令绝对三点螺旋插补。
	路径	Acm_GpAddPath	√	√	向路径缓存中添加一个路径。
		Acm_GpResetPath	√	√	清空路径缓存中的数据路径。
		Acm_GpLoadPath	√	√	加载一个路径文件。
		Acm_GpUnloadPath	√	√	卸载路径。
		Acm_GpMovePath	√	√	开始执行路径缓存中的所有路径命令。
		Acm_GpGetPathStatus	√	√	获取当前路径状态。
		Acm_GpMoveSelPath	√	√	执行指定范围的路径。
		Acm_GpGetPathIndexStatus	√	√	获取指定索引路径的状态。
	暂停&恢复	Acm_GpPauseMotion	√	√	暂停群组运动命令。
		Acm_GpResumeMotion	√	√	恢复暂停的群组运动命令

6.2.5 属性支持列表

类型		属性	PCI-1285	PCI-1285E
设备	特性	FT_DevIpoTypeMap	√	√
		FT_DevAxesCount	√	√
		FT_DevFunctionMap	√	√
		FT_DevOverflowCntr	√	√
		FT_DevMDAQTypeMap	√	√
		FT_DevMDAQTrigMap	√	√
		FT_DevMDAQMaxChan	√	√
		FT_DevMDAQMaxBufCount	√	√
	配置	CFG_DevBoardID	√	√
		CFG_DevBaseAddress	√	√
		CFG_DevInterrupt	√	√
		CFG_DevBusNumber	√	√
		CFG_DevSlotNumber	√	√
		CFG_DevDriverVersion	√	√
		CFG_DevDllVersion	√	√
		CFG_DevFirmVersion	√	√
		CFG_DevFPGAVersion	√	√
		CFG_DevFPGA_1Version	√	√
		CFG_DevEmgLogic	√	√
DAQ	特性	FT_DaqDiMaxChan	X	X
		FT_DaqDoMaxChan	X	X
		FT_DaqAiRangeMap	X	X
		FT_DaqAiMaxSingleChan	X	X
		FT_DaqAiMaxDiffChan	X	X
		FT_DaqAiResolution	X	X
	配置	CFG_DaqAiChanType	X	X
		CFG_DaqAiRanges	X	X

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_AxAlmEn	√	√
		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_AxEzLogic	√	√
		CFG_AxHomeResetEnable	√	√
		PAR_AxHomeCrossDistance	√	√
		PAR_AxHomeExSwitchMode	√	√
	背隙	FT_AxBacklashMap	√	√
		CFG_AxBacklashEnable	√	√
		CFG_AxBacklashPulses	√	√
		CFG_AxBacklashVel	√	√
	比较	FT_AxCompareMap	√	√
		CFG_AxCmpSrc	√	X
		CFG_AxCmpMethod	√	X
		CFG_AxCmpPulseMode	√	X
		CFG_AxCmpPulseLogic	√	X
		CFG_AxCmpPulseWidth	√	X
		CFG_AxCmpEnable	√	X
	锁存	FT_AxLatchMap	√	√
		CFG_AxLatchLogic	√	X
		CFG_AxLatchEnable	√	X
	Aux/Gen 输出	FT_AxGenDOMap	√	√
		FT_AxGenDIMap	√	√
		CFG_AxGenDoEnable	√	√
	外部驱动	FT_AxExtDriveMap	√	√
		FT_AxExtMasterSrcMap	√	√
		CFG_AxExtMasterSrc	√	√
		CFG_AxExtSelEnable	X	X
		CFG_AxExtPulseNum	√	√
		CFG_AxExtPulseInMode	√	√
		CFG_AxExtPresetNum	√	√
	CAM DO	FT_AxCamDOMap	√	√
		CFG_AxCamDOEnable	√	X
		CFG_AxCamDOLoLimit	√	X
		CFG_AxCamDOHiLimit	√	X
		CFG_AxCamDOCmpSrc	√	X
		CFG_AxCamDOLogic	√	X
	模块	CFG_AxModuleRange	√	X
	同步启停	FT_AxSimStartSourceMap	√	√
		CFG_AxSimStartSource	√	X

轴	DI Stop	FT_AxIN1Map	X	X
		FT_AxIN2Map	X	X
		FT_AxIN4Map	X	X
		FT_AxIN5Map	X	X
		CFG_AxIN1StopEnable	X	X
		CFG_AxIN1StopReact	X	X
		CFG_AxIN1StopLogic	X	X
		CFG_AxIN2StopEnable	X	X
		CFG_AxIN2StopReact	X	X
		CFG_AxIN2StopLogic	X	X
		CFG_AxIN4StopEnable	X	X
		CFG_AxIN4StopReact	X	X
		CFG_AxIN4StopLogic	X	X
		CFG_AxIN5StopEnable	X	X
		CFG_AxIN5StopReact	X	X
		CFG_AxIN6StopLogic	X	X
群组	系统	PAR_GpGroupID	√	√
		CFG_GpAxesInGroup	√	√
	应用	CFG_GpSFEnable	√	X
		CFG_GpBldTime	√	X
		PAR_GpRefPlane	√	X
	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-1285/1285E 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。

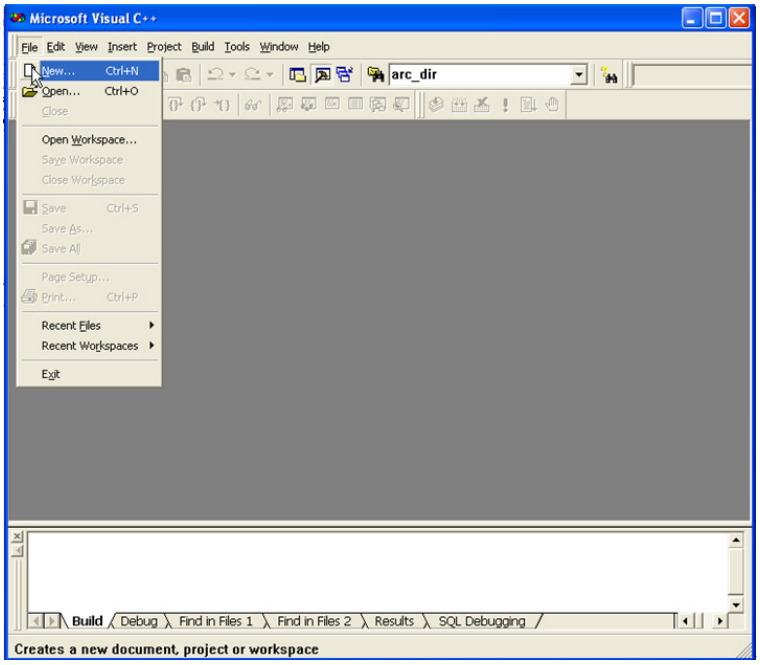


图 6.8： 打开文件创建一个新的 VC 应用

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。

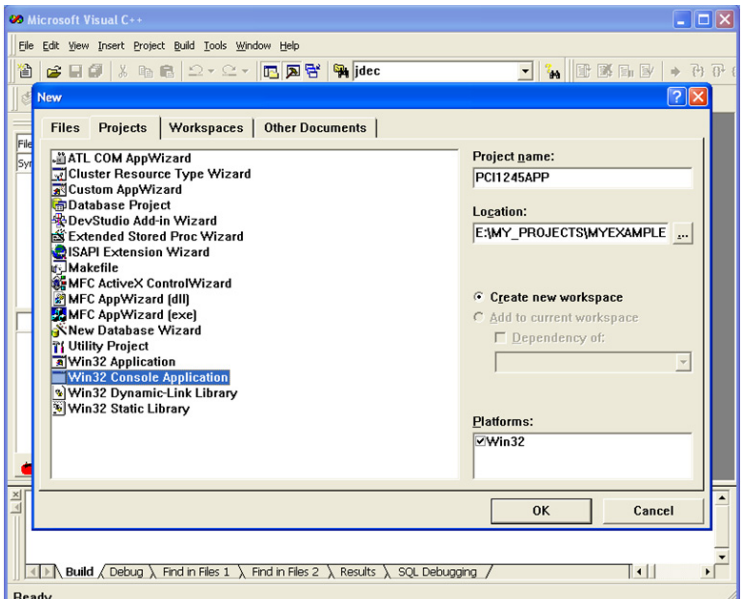


图 6.9： 创建一个新的 VC 控制台应用

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 convension” 应为 “_stdcall”，因此用户需按照如下所示配置 “Calling convension”：

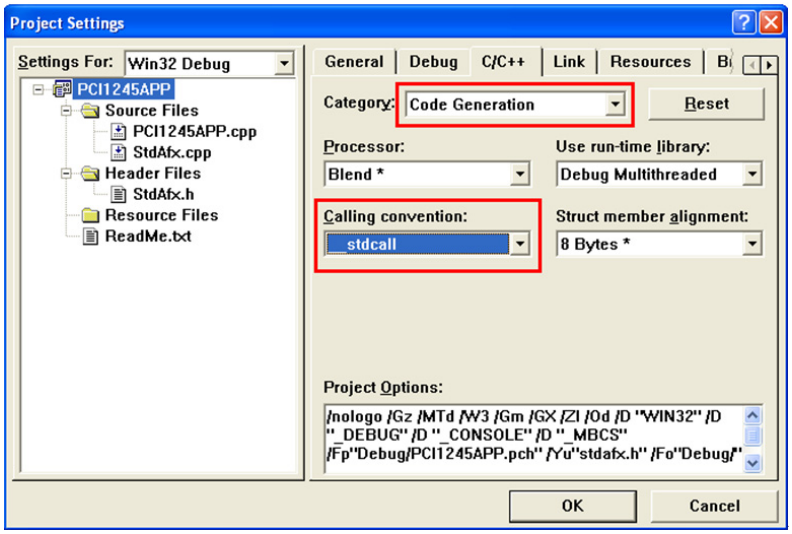


图 6.10： 设置 “Caling convention”

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 6.11： 该示例的文件内容

因此，路径设置如下所示。

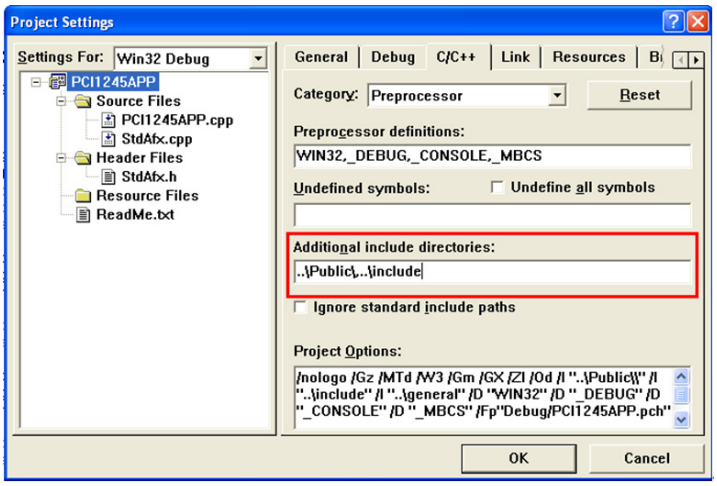


图 6.12： 添加头文件路径

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。

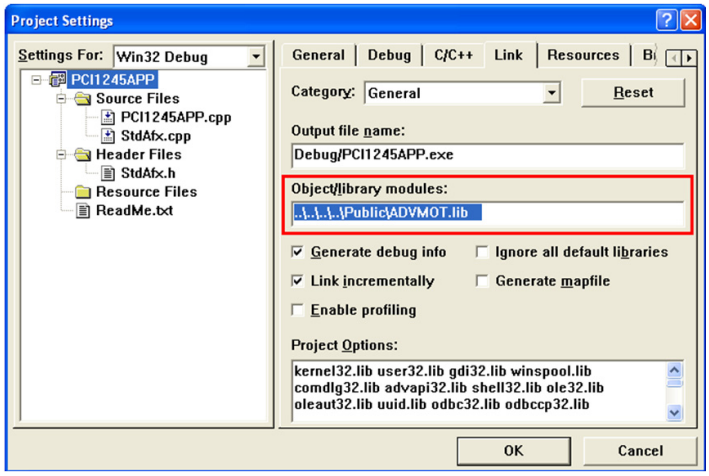


图 6.13： 设置 Lib 文件路径

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。

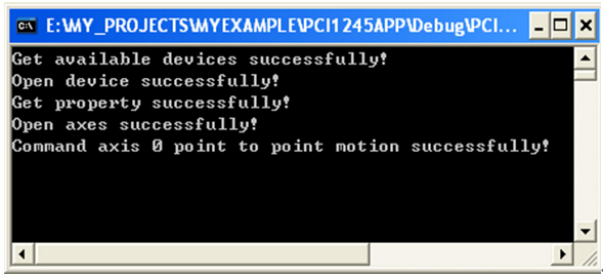


图 6.14： VC 控制台示例的结果

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 6.15： 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。

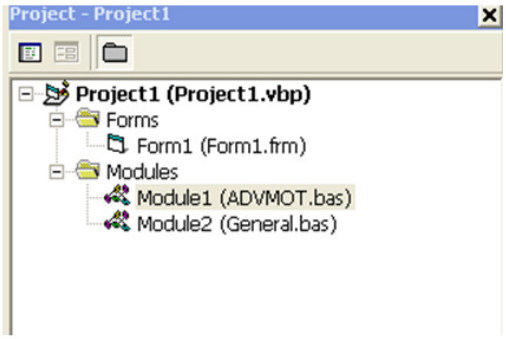


图 6.16： 将模块文件添加到工程中

4. 设计表框。

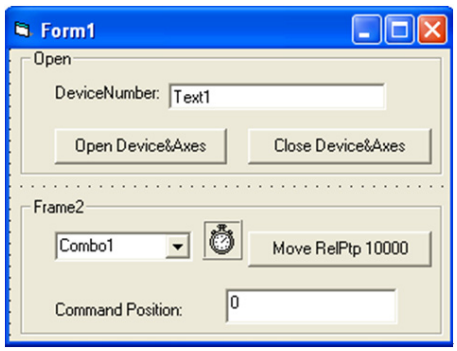


图 6.17： 设计表框。

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：

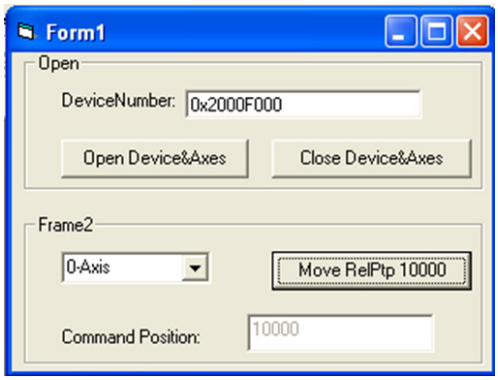


图 6.18： 执行结果

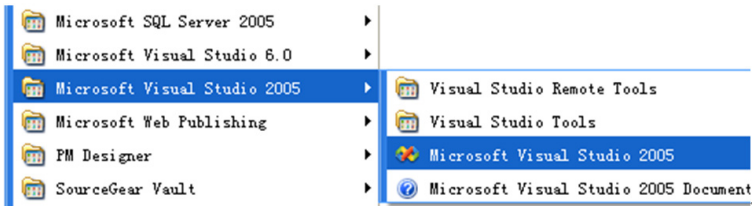
6.2.6.3 创建一个新的 C# 应用

如需使用 PCI-1285/1285E 系列基于 DSP 的 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：

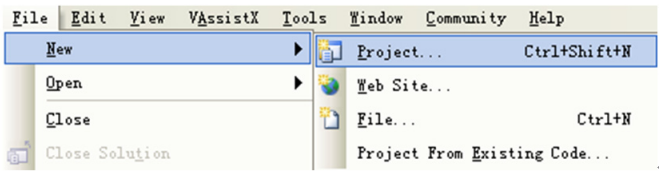
1. **创建一个新的工程**

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

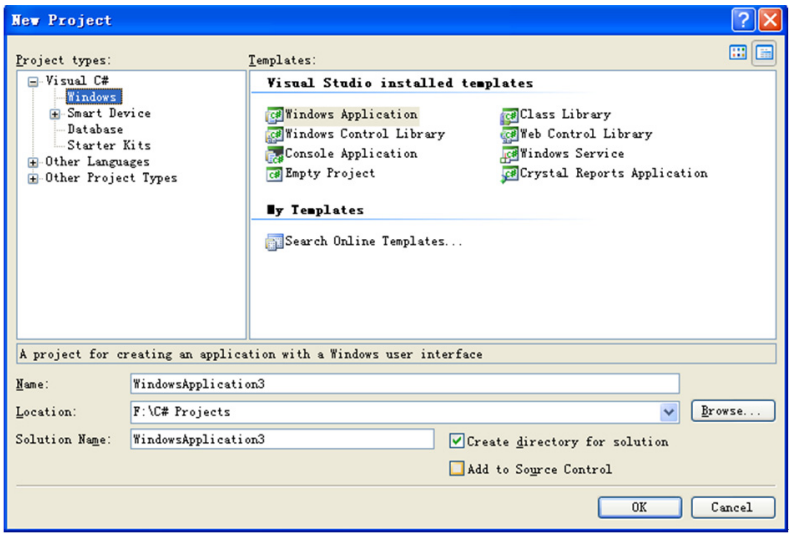


Microsoft Visual Studio 2005 的开发环境如下图所示：

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

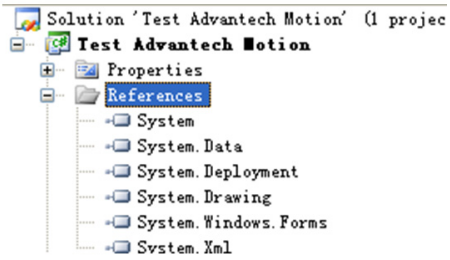


在新的窗口中，默认语言为 “Visual C#”。选择 [Windows Application] 模板、设置 “Name”、“Location” 和 “Solution Name” （保留默认），然后单击 ”OK “。

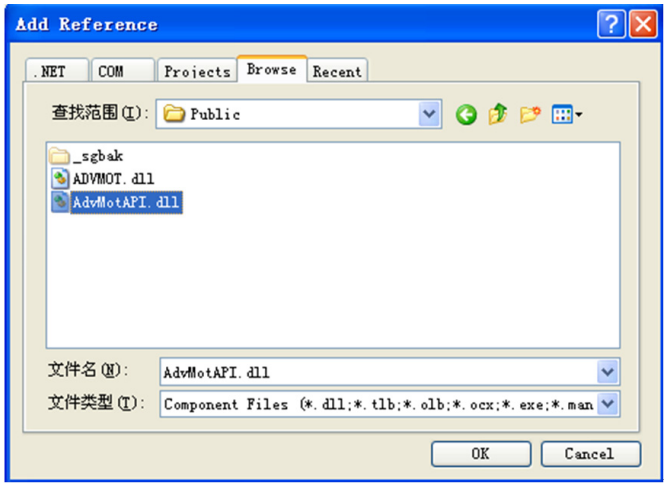


2. **添加相关 DLL 文件的引用**

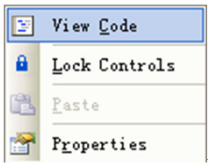
a. 单击开发环境右上角处的 [References]，如下图所示：



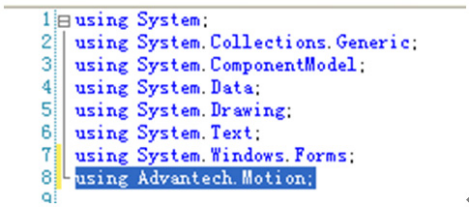
b. 单击 [Add Reference] 对话框的 [Browse]，从搜索路径选择 “Public” 文件夹中的 “AdvMotAPI.dll”，然后单击 [OK]，如下图所示：



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：



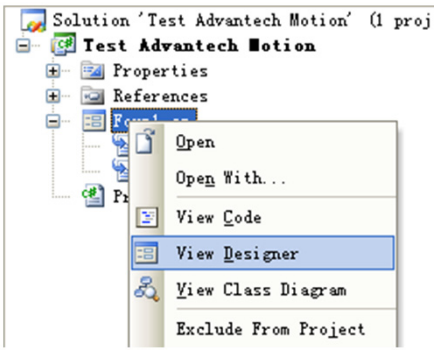
d. 在原始参考命名空间下添加 “using Advantech.Motion”，如下图所示：



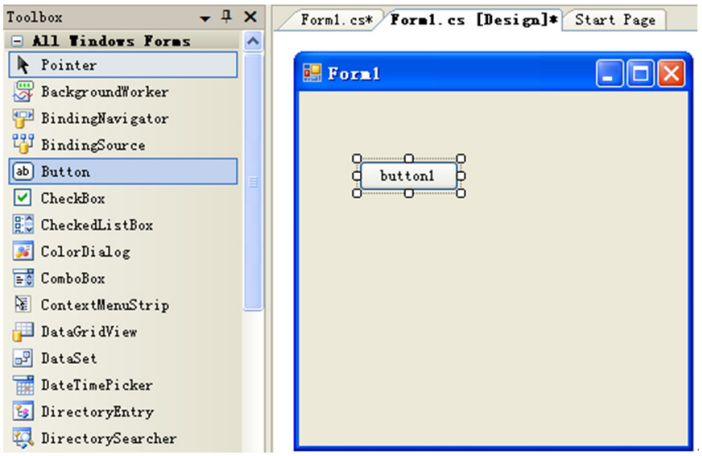
3. **编码**

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：



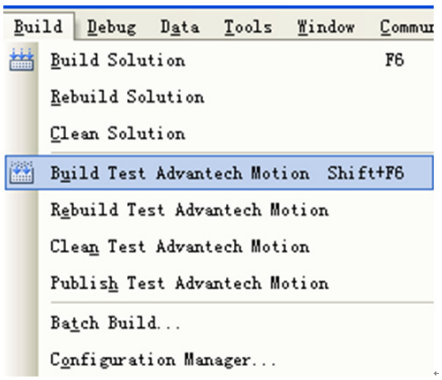
有关详细信息，请参考 Microsoft Visual C # 用户手册。

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-1285/1285E 系列基于 DSP 的 SoftMotion PCI 控制器中的 C# 示例。

4. **测试程序**

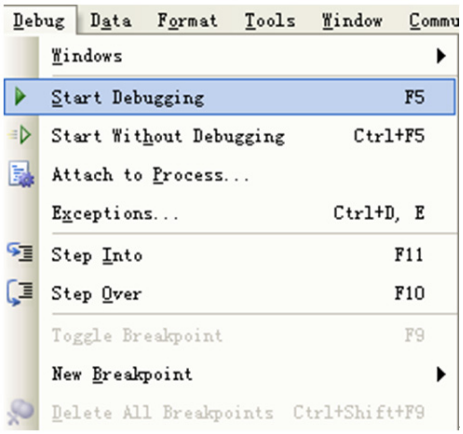
用户编程之后或想要编译程序，可从菜板栏中单击 [Build] ---> [Build Solution]\[Build Test Advantech Motion]，如下图所示：



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：

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：



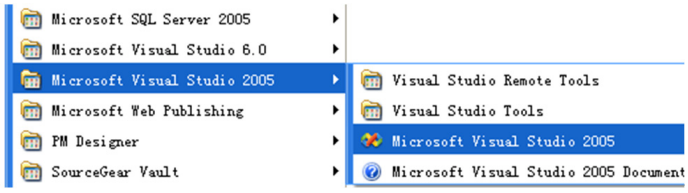
6.2.6.4 创建一个新的 VB.net 应用

如需使用 PCI-1285/1285E 系列基于 DSP 的 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 Visual Basic 工程：

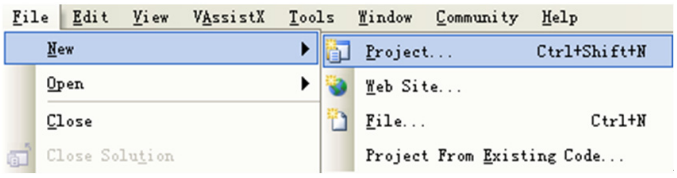
1. **创建一个新的工程**

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

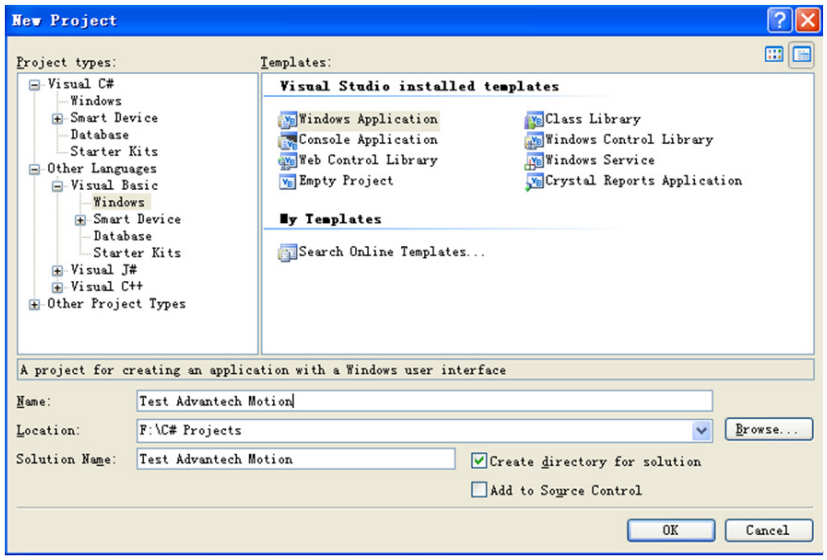


Microsoft Visual Studio 2005 的开发环境如下图所示：

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

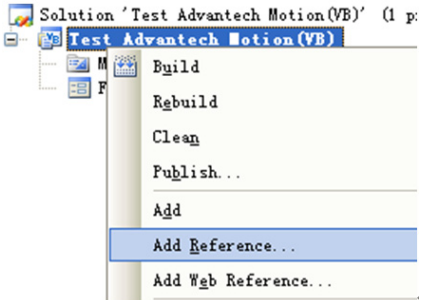


在新的窗口中，选择 [Other Languages]--->[Visual Basic] 并选择 [Windows Application] 模板、设置 “Name”、“Location” 和 “Solution Name” （保留默认），然后单击 ”OK “。

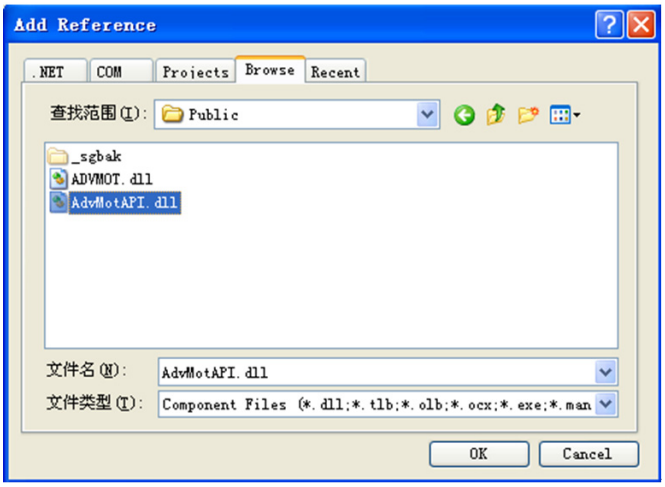


2. **添加相关 DLL 文件的引用**

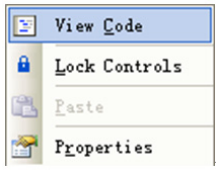
a. 单击开发环境右上角处的 [References]，如下图所示：



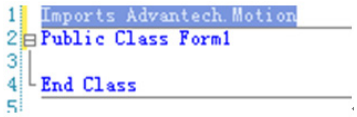
b. 单击 [Add Reference] 对话框的 [Browse]，从搜索路径选择 “Public” 文件夹中的 “AdvMotAPI.dll”，然后单击 [OK]，如下图所示：



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：



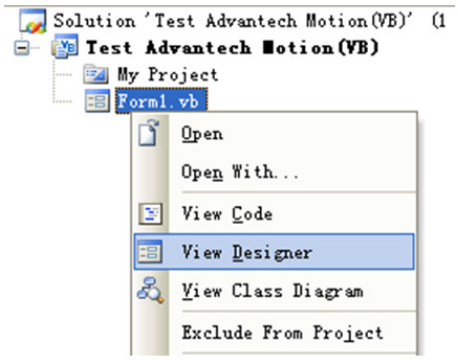
d. 在原始参考命名空间下添加 “Imports Advantech.Motion”，如下图所示：



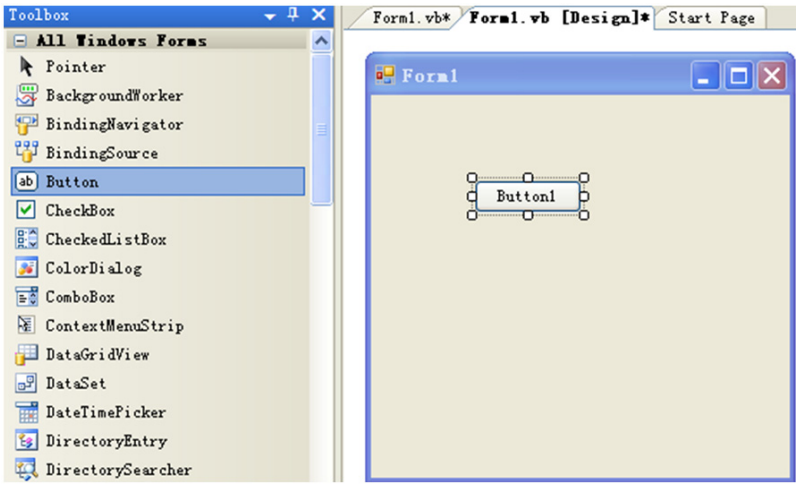
3. **编码**

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：



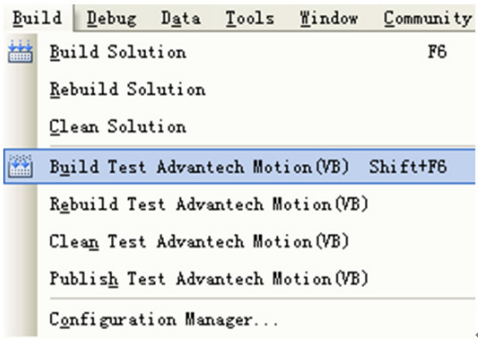
有关详细信息，请参考 Microsoft Visual Basic 用户手册。

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-1245/1245E/1265 系列基于 DSP 的 SoftMotion PCI 控制器中的 VB.NET 示例。

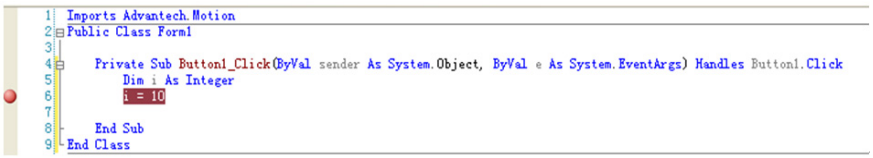
4. **测试程序**

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-1285/1285E。

NumOfSubDevices:

仅用于 AMONET 系统，在 PCI-1285/1285E 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。

6.3.2 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 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-1285/1285E，Axis_Num 为 8。

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 31…4 3 2 1 0 说明 未定义 EVT_AX_VH_END EVT_AX_VH_START EVT_AX_COMPARED (Not support) EVT_AX_MOTION_DONE 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 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 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。

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

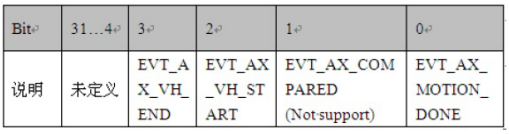
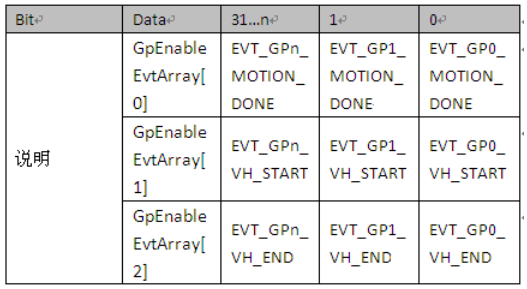
格式：

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

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。

6.3.2.9 Acm_DevDownloadCAMTable

格式：

U32 Acm_DevDownloadCAMTable (HAND DeviceHandle, U32 CamTableID, PF64 pMasterArray, PF64 pSlaveArray, PF64 pPointRangeArray, PF64 pPointSlopeArray, U32 ArrayElements)

目的：

该函数用于加载凸轮关系表，描述主轴和从轴之间的关系。

参数：

名称	类型	IN/OUT	说明
DeviceHandle	HAND	IN	设备句柄。 该句柄为来自 Acm_DevOpen 的设备句柄。
CamTableID	U32	IN	CAM 表的标识符。 PCI-1245 和 PCI-1265 保留了 2 组 CAM 表。 因此 ID 为 0 和 1。
pMasterArray	PF64	IN	CAM 表中主轴的位置列表。
pSlaveArray	PF64	IN	CAM 表中从轴的位置列表。
pPointRangeArray	PF64	IN	CAM 表中各点与辅助点之间的距离。
pPointSlopeArray	PF64	IN	CAM 表中各点与其辅助点之间连线的斜率。
ArrayElements	U32	IN	pMasterArray/ pSlaveArray/ pPointRangeArray/ pPointSlopeArray 阵列中的元素个数。 最大元素个数为 128。

返回值：

错误代码

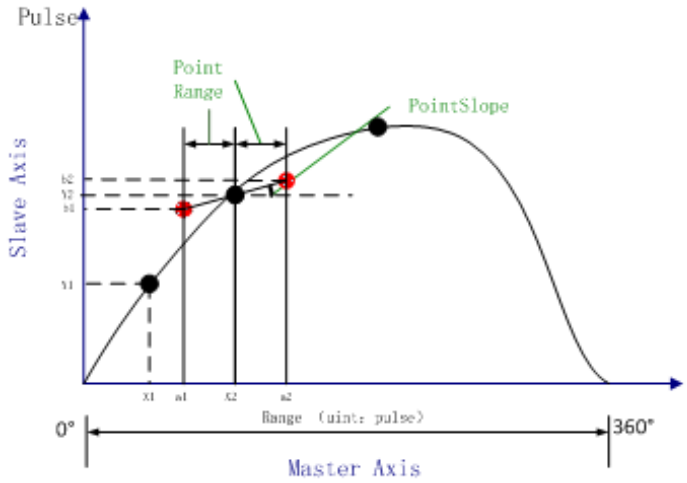
注解：

凸轮的主要特征是主轴和从轴之间的动态比列以及相位移动。主轴和从轴之间的关系表就是 CAM 表。

凸轮操作是靠这张表执行的 （二维 - 描述主和从位置）。表格中的数据关系应该为单调上升或单调下降，随着主轴运动同时向前或向后执行。

PCI-1285/1285E 不支持该 API。

参数的含义如下：



如上图所示，当主轴旋转 360° 时，**Range** 为所需的脉冲个数。图上的黑色点组成 **MasterArray** 的主轴位置 （如 X1 和 X2）和 **SlaveArray** 中的相应从轴位置。由黑色点以及指定的 PointRange 和 PointSlope 产生的红色点称为辅助参考点。CAM 曲线是通过由 MasterArray 和 SlaveArray 组成的 CAM 表拟合出来的。水平轴为当主轴旋转到某个角度时对应的脉冲数。垂直轴为当主轴旋转到某个角度时从轴对应的位置。

Range 必须通过属性 CFG_AxModuleRange 设置在主轴内。

有关 CAM 的操作信息，请参考第 6.2.2 节的 E-CAM 流程图。

6.3.2.10 Acm_DevConfigCAMTable

格式：

U32 Acm_DevConfigCAMTable (HAND DeviceHandle, U32 CamTableID, U32 Periodic, U32 MasterAbsolute, U32 SlaveAbsolute)

目的：

该函数设置 CAM 表的相关参数。

参数：

名称	类型	IN/OUT	说明
DeviceHandle	HAND	IN	设备句柄。该句柄为来自 Acm_DevOpen 的设备句柄。
CamTableID	U32	IN	CAM 表的标识符， 由 Acm_DevDownloadCAMTable 分配。设备内保留了 2 个 CAM 表。因此 ID 为 0 和 1。
Periodic	U32	IN	定期或非定期执行 CAM 曲线。 0：非定期； 1：定期。
MasterAbsolute	U32	IN	指定 CAM 曲线相对 （0）或绝对 （1）于主轴。 0：相对； 1：绝对。
SlaveAbsolute	U32	IN	指定 CAM 曲线相对 （0）或绝对 （1）于从轴。 0：相对； 1：绝对。

返回值：

错误代码

注解：

凸轮操作是靠这张表执行的 （二维 - 描述主和从位置）。

凸轮系统有两种类型：

■Periodic

定期：一旦开始执行，凸轮系统将立刻从曲线的起始处重新开始。如下图所示：

非定期：执行曲线后，凸轮执行完一个周期后，结束凸轮关系的执行。

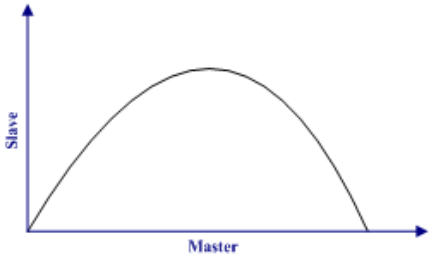
■MasterAbsolute 和 SlaveAbsolute

绝对：当绝对凸轮系统设置后，基于 CamTable 的控制值或从值将被认为是绝对的。系统会补偿同步过程中主从轴之间的任何偏差。当达到同步时，控制值和从值之间将建立一种确定的相位关系。

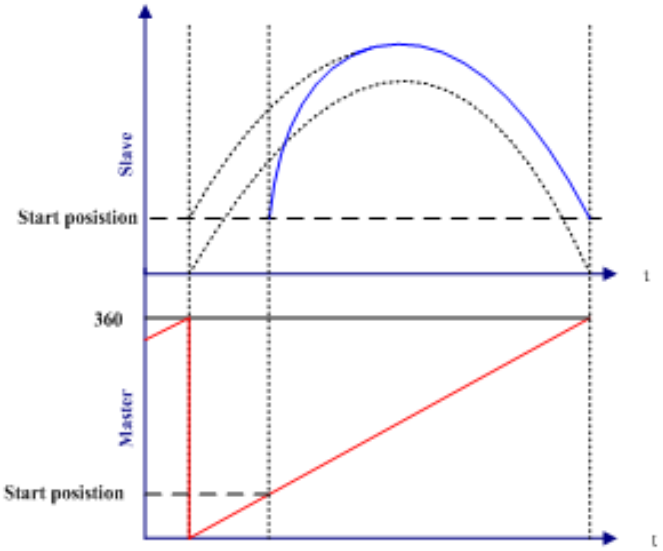
相对：当相对凸轮系统设置后，则表示系统不会补偿在同步过程中控制值和从值之间的任何偏移。

比如，实用程序中剖面图如下所示：

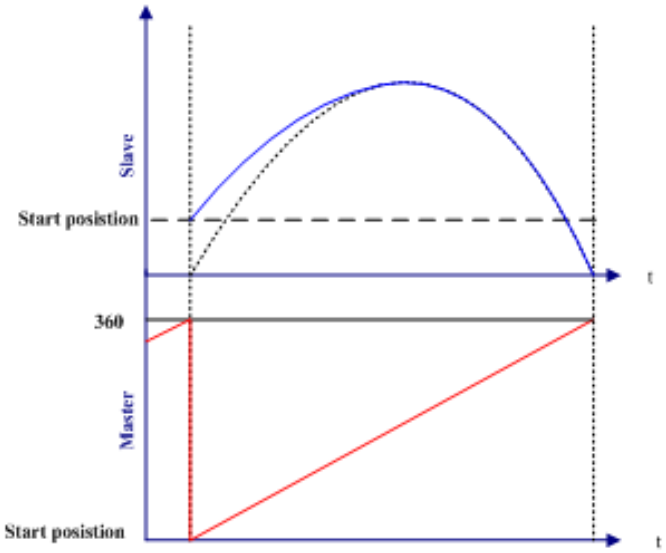
初始 CAM 曲线如下图所示：



- 主轴：绝对；从轴：相对。



- 主轴：相对；从轴：绝对。



PCI-1285/1285E 不支持该 API。

6.3.2.11 Acm_DevLoadCAMTableFile

格式：

U32 Acm_DevLoadCAMTableFile (HAND DeviceHandle, PI8 FilePath, U32 CamTableID, PU32 Range, PU32 PointsCount)

目的：

加载编辑过的 Cam Table 文件，并通过 Utility 将其保存至设备。

参数：

名称	类型	IN 或 OUT	说明
DeviceHandle	HAND	IN	来自 Acm_DevOpen 的设备句柄。
FilePath	PI8	IN	指针指向保存 CamTable 文件路径的字符串。
CamTableID	U32	IN	CamTableID：0 或 1。
Range	PU32	OUT	主轴在一个周期内需要的脉冲个数。
PointsCount	PU32	OUT	CamTable 中的点数量。

返回值：

错误代码

注解：

CamTable 文件在 Utility 中保存为 .bin 格式。当该 API 成功加载文件时，则不需要调用 API Acm_DevDownLoadCACMTable。

有关 E-CAM 的操作信息，请参考第 6.2.2 节的 E-CAM 流程图。

PCI-1285/1285E 不支持该 API。

6.3.2.12 Acm_DevMDaqConfig

格式：

U32 Acm_DevMDaqConfig (HAND DeviceHandle, U16 ChannelID,
U32 Period, U32 AxisNo, U32 Method, U32 ChanType, U32 Count)

目的：

设定 MotionDAQ 相关配置

参数：

名称	类型	IN/OUT	说明
DeviceHandle	HAND	IN	来自 Acm_DevOpen 的设备句柄。
ChannelID	U16	IN	指定 Channel ID，范围：0 ~ 3。
Period	U32	IN	设置每记录一笔资料的间隔时间。范围：0 ~ 255。单位：ms。
AxisNo	U32	IN	指定记录哪一个轴的信息。范围：0 ~ 3 （PCI1245/45E），0 ~ 5 （PCI1265）。
Method	U32	IN	触发 MDaq 的方式。 0：禁用； 1：软体触发方式； 2：DI 触发； 3：轴 0 启动触发； 4：轴 1 启动触发； 5：轴 2 启动触发； 6：轴 3 启动触发； 7：轴 4 启动触发； （PCI1245(E) 不支持） 8：轴 5 启动触发； （PCI1245(E) 不支持） 9：轴 6 启动触发； （PCI1265/PCI1245(E) 不支持） 10：轴 7 启动触发； （PCI1265/PCI1245(E) 不支持） 11：轴 8 启动触发； （PCI1265/PCI1245(E) 不支持） 12：轴 9 启动触发； （PCI1265/PCI1245(E) 不支持） 13：轴 10 启动触发； （PCI1265/PCI1245(E) 不支持） 14：轴 11 启动触发。（PCI1265/PCI1245(E) 不支持）
ChanType	U32	IN	获取 Channel Type： 0：理论位置； 1：实际位置； 2：理论位置与实际位置的偏差值； 3：理论速度值。（PCI1265/PCI1245(E) 不支持）
Count	U32	IN	指定要记录的资料数量。范围：0 ~ 2000。

返回值：

错误代码

注解：

PCI-1285/1285E 不支持该 API。

6.3.2.13 Acm_DevMDaqGetConfig

格式：

U32 Acm_DevMDaqGetConfig (HAND DeviceHandle, U16 ChannelID, PU32 Period, PU32 AxisNo, PU32 Method, PU32 ChanType, PU32 Count)

目的：

获取指定 ChannelID 的 MotionDAQ 相关设定值。

参数：

名称	类型	IN/OUT	说明
DeviceHandle	HAND	IN	Device Handle
ChannelID	U16	IN	指定获取哪一个 MDaq Channel 的配置信息。范围：0 ~ 3。
Period	PU32	OUT	获取每记录一笔资料的间隔时间。范围：0 ~ 255，单位：ms。
AxisNo	PU32	OUT	获取记录哪一个轴的相关资料。范围：0 ~ 3(PCI1245/45E)，0 ~ 5(PCI1265)。
Method	PU32	OUT	获取触发 MDaq 的方式。 0：禁用； 1：软体触发方式； 2：DI 触发； 3：轴 0 启动触发； 4：轴 1 启动触发； 5：轴 2 启动触发； 6：轴 3 启动触发； 7：轴 4 启动触发； 8：轴 5 启动触发； 9：轴 6 启动触发； 10：轴 7 启动触发； 11：轴 8 启动触发； 12：轴 9 启动触发； 13：轴 10 启动触发； 14：轴 11 启动触发。
ChanType	PU32	OUT	获取 Channel Type： 0：理论位置； 1：实际位置； 2：理论位置与实际位置的偏差值； 3：理论速度值。
Count	PU32	OUT	获取设定的最大记录笔数，范围：0 ~ 2000。

返回值：

错误代码

注解：

PCI-1285/1285E 不支持该 API。

6.3.2.14 Acm_DevMDaqStart

格式：

U32 Acm_DevMDaqStart(HAND DeviceHandle)

目的：

启动 MotionDAQ 功能，以记录相关资料。

参数：

名称	类型	IN/OUT	说明
DeviceHandle	HAND	IN	Device Handle

注解：

当 Acm_DevMDaqConfig 中 Method 设定为 MQ_TRIG_SW 时，通过此 API 下达命令以触发 MotionDAQ 功能。

PCI-1285/1285E 不支持该 API。

6.3.2.15 Acm_DevMDaqStop

格式：

U32 Acm_DevMDaqStop (HAND DeviceHandle)

目的：

停止记录 MotionDAQ 相关资料。

参数：

名称	类型	IN/OUT	说明
DeviceHandle	HAND	IN	Device Handle

注解：

PCI-1285/1285E 不支持该 API。

6.3.2.16 Acm_DevMDaqReset

格式：

U32 Acm_DevMDaqReset (HAND DeviceHandle, U16 ChannelID)

目的：

清除相应 ChannelID 的 MotionDAQ 资料。

参数：

名称	类型	IN/OUT	说明
DeviceHandle	HAND	IN	Device Handle
ChannelID	U16	IN	Channel ID。范围：0 ~ 3。

注解：

PCI-1285/1285E 不支持该 API。

6.3.2.17 Acm_DevMDaqGetStatus

格式：

U32 Acm_DevMDaqGetStatus(HAND DeviceHandle, U16 ChannelID, PU16 CurrentCnt, PU8 Status)

目的：

获取相应 ChannelID 的 MotionDAQ 状态。

参数：

名称	类型	IN/OUT	说明
DeviceHandle	HAND	IN	Device Handle
ChannelID	U16	IN	Channel ID。范围：0 ~ 3。
CurrentCnt	PU16	OUT	返回当前已记录的笔数
Status	PU8	OUT	返回当前状态： 0：Ready 1：Waiting Trigger 2：Started

注解：

PCI-1285/1285E 不支持该 API。

6.3.2.18 Acm_DevMDaqGetData

格式：

U32 Acm_DevMDaqGetData(HAND DeviceHandle, U16 ChannelID, U16 StartIndex, U16 MaxCount, PI32 DataBuffer)

目的：

获取相应 Channel 已经记录的 MotionDAQ 的指定范围内的所有资料。

参数：

名称	类型	IN/OUT	说明
DeviceHandle	HAND	IN	Device Handle
ChannelID	U16	IN	Channel ID。范围：0 ~ 3。
StartIndex	U16	IN	设置获取资料的起始位置。
MaxCount	U16	IN	设置获取资料的数量。
DataBuffer	PI32	OUT	返回指定范围内的资料。

注解：

PCI-1285/1285E 不支持该 API。

6.3.3 DAQ

6.3.3.1 数字量输入 / 输出

6.3.3.1.1 Acm_DaqDiGetByte

格式：

U32 Acm_DaqDiGetByte (HAND DeviceHandle, U16 DiPort, PU8 ByteData)

目的：

以字节形式获取指定 DI 端口的数据

参数：

名称	类型	IN 或 OUT	说明
DeviceHandle	HAND	IN	来自 Acm_DevOpen 的设备句柄。
DiPort	U16	IN	数值应为 0。
ByteData	PU8	OUT	用于获取数值的指针。

返回值：

错误代码

注解：

PCI-1285 和 PCI-1285E 不支持 DI。

6.3.3.1.2 Acm_DaqDiGetBit

格式：

U32 Acm_DaqDiGetBit (HAND DeviceHandle, U16 DiChannel, PU8 BitData)

目的：

获取指定 DI 通道的位数据。

参数：

名称	类型	IN 或 OUT	说明
DeviceHandle	HAND	IN	来自 Acm_DevOpen 的设备句柄。
DiChannel	U16	IN	DI 通道。
BitData	PU8	OUT	返回的位数据， 0 或 1。

返回值：

错误代码

注解：

PCI-1285 和 PCI-1285E 不支持 DI。

6.3.3.1.3 Acm_DaqDoSetByte

格式：

U32 Acm_DaqDoSetByte (HAND DeviceHandle, U16 DoPort, U8 ByteData)

目的：

设置指定 DO 端口的值。

参数：

名称	类型	IN 或 OUT	说明
DeviceHandle	HAND	IN	来自 Acm_DevOpen 的设备句柄。
DoPort	U16	IN	DO 端口。
ByteData	U8	IN	需要设置的值。

返回值：

错误代码

注解：

PCI-1285 和 PCI-1285E 不支持 DO。

6.3.3.1.4 Acm_DaqDoSetBit

格式：

U32 Acm_DaqDoSetBit (HAND DeviceHandle, U16 DoChannel, U8 BitData)

目的：

设置指定 DO 通道的值。

参数：

名称	类型	IN 或 OUT	说明
DeviceHandle	HAND	IN	来自 Acm_DevOpen 的设备句柄。
DoChannel	U16	IN	DO 通道。
BitData	U8	IN	需要设置的值， 0 或 1。

返回值：

错误代码

注解：

PCI-1285 和 PCI-1285E 不支持 DO。

6.3.3.1.5 Acm_DaqDoGetByte

格式：

U32 Acm_DaqDoGetByte(HAND DeviceHandle, U16 DoPort,PU8 ByteData)

目的：

获取指定 DO 端口的字节数据。

参数：

名称	类型	IN 或 OUT	说明
DeviceHandle	HAND	IN	来自 Acm_DevOpen 的设备句柄。
DoPort	U16	IN	DO 端口。
ByteData	PU8	OUT	返回值。

返回值：

错误代码

注解：

PCI-1285 和 PCI-1285E 不支持 DO。

6.3.3.1.6 Acm_DaqDoGetBit

格式：

U32 Acm_DaqDoGetBit(HAND DeviceHandle, U16 DoChannel, PU8 BitData)

目的：

获取指定 DO 通道的位值。

参数：

名称	类型	IN 或 OUT	说明
DeviceHandle	HAND	IN	来自 Acm_DevOpen 的设备句柄。
DoChannel	U16	IN	DO 通道 0 ~ 7。
BitData	PU8	OUT	返回值 0 或 1。

返回值：

错误代码

注解：

PCI-1285 和 PCI-1285E 不支持 DO。

6.3.3.2 模拟量输入 / 输出输出

6.3.3.2.1 Acm_DaqAiGetRawData

格式：

U32 Acm_DaqAiGetRawData(HAND DeviceHandle, U16 AiChannel, PU16 AiData)

目的：

获取模拟量输入值的二进制值

参数：

名称	类型	In 或 Out	描述
DeviceHandle	HAND	IN	由 Acm_DevOpen 产生的设备句柄。
AiChannel	U16	IN	AI 通道：0 或 1。
AiData	PU16	OUT	指向返回的二进制 AI 值的指针。

返回值：

错误代码

注解：

PCI-1285/1285E 不支持 AI 功能。

在 PCI-1265 中总共有两个 AI 通道，分别为 0 和 1。

可通过CFG_DaqAiChanType 属性来设定通道是单端输入还是差分输入，如果是单端输入，则用户可以分别获取到 AI 值，而如果是差分输入，用户只能通过通道 1 来获取 AI 值。

6.3.3.2.2 Acm_DaqAiGetVoltData

格式：

U32 Acm_DaqAiGetVoltData(HAND DeviceHandle, U16 AiChannel, PF32 AiData)

目的：

获取实际模拟量输入值。

参数：

名称	类型	In 或 Out	描述
DeviceHandle	HAND	IN	由 Acm_DevOpen 产生的设备句柄。

AiChannel	U16	IN	AI 通道：0 或 1。
AiData	PF32	OUT	指向返回的模拟量值的指针。

返回值：

错误代码

注解：

PCI-1285/1285E 不支持 AI 功能。

在 PCI-1265 中总共有两个 AI 通道，分别为 0 和 1。

可通过 CFG_DaqAiChanType 属性来设定通道是单端输入还是差分输入，如果是单端输入，则用户可以分别获取到 AI 值，而如果是差分输入，用户只能通过通道 1 来获取 AI 值。

6.3.4 轴

6.3.4.1 系统

6.3.4.1.1 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。
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-1285/1285E 的物理轴编号为：0、1、2、3、4、5、6、7。

6.3.4.1.2 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。

6.3.4.1.3 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：

6.3.4.2 运动 IO

6.3.4.2.1 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：

6.3.4.2.2 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-1245/1245E/1265 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-1285/1285E 不支持）； 11：LTC ---- 锁存信号输入； 12：SD ---- 减速信号输入 （PCI-1285/1285E 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）； 19：CAMDO ---- 凸轮区间 DO （OUT4）。

返回值：

错误代码

注解：

6.3.4.3 运动状态

6.3.4.3.1 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：

6.3.4.3.2 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：

6.3.4.4 速度运动

6.3.4.4.1 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。

6.3.4.4.2 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。

6.3.4.4.3 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：

6.3.4.4.4 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：

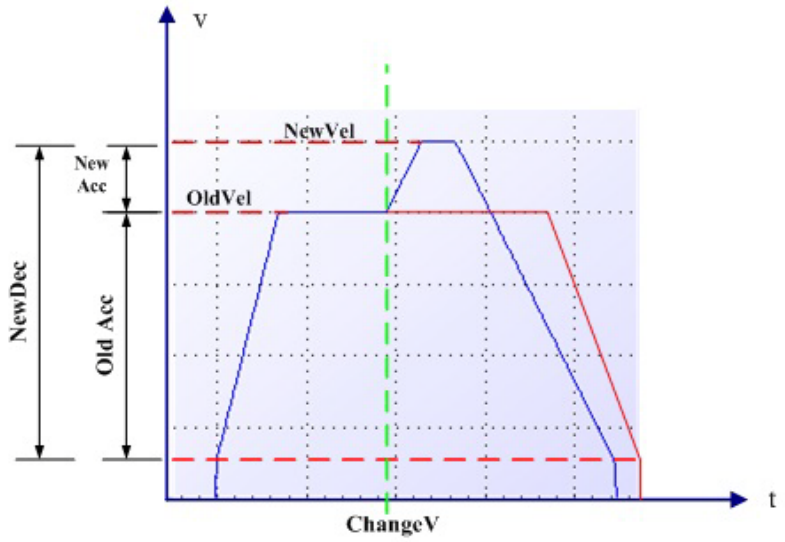
错误代码

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。



6.3.4.4.5 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：

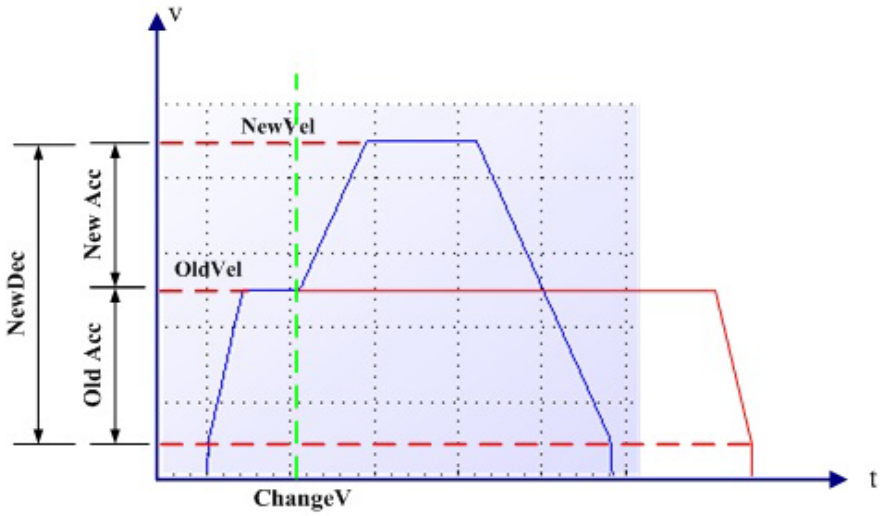
错误代码

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。



6.3.4.5 点到点运动

6.3.4.5.1 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。

6.3.4.5.2 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。

6.3.4.5.3 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。

6.3.4.5.4 Acm_AxMoveImpose

格式：

U32 Acm_AxMoveImpose (HAND AxisHandle, F64 Position, F64 NewVel)

目的：

在当前运动上叠加新的运动。

参数：

名称	类型	IN 或 OUT	说明
AxisHandle	HAND	IN	来自 Acm_AxOpen 的轴句柄。
Position	F64	IN	新运动的相对位置。（单位 = PPU）
NewVel	F64	IN	该叠加运动的速度。（单位 = PPU/s）

返回值：

错误代码

注解：

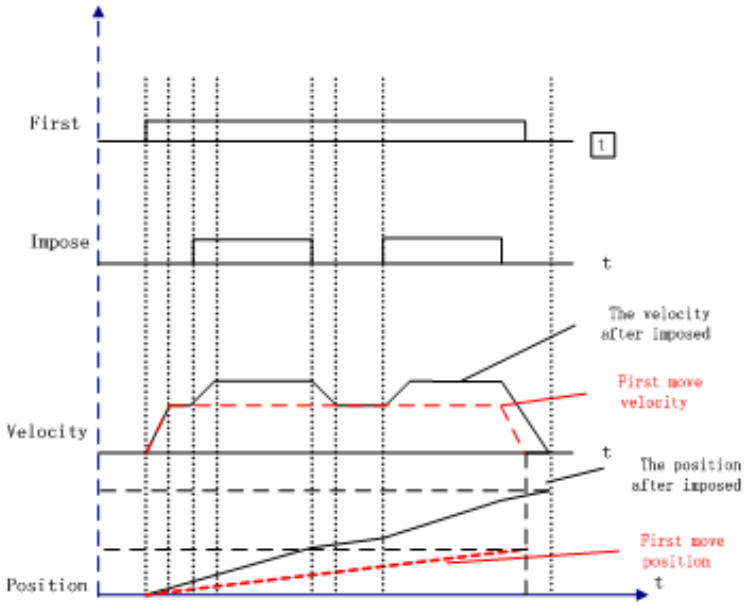
该函数仅支持该新叠加的运动的速度曲线为 T 型曲线。

运动停止的终止位置将为原始位置加 / 减 Position， 且驱动速度也将改变。

整个速度曲线由该运动的 **NewVel** 决定，由原始运动的 PAR_AxVelLow、PAR_AxVelHigh、PAR_AxAcc、PAR_AxDec 和 PAR_AxJerk 决定。

Position + 原始位置的范围：-2147483647/ **PPU** ~ 2147483647/ **PPU**，且 **NewVel** + 原始 FH 的范围：0 ~ CFG_AxMaxVel。

比如，如下图所示：



注！ *不能在叠加运动上叠加新的运动。*

PCI-1285E 不支持该 API。

6.3.4.6 同步起停运动

6.3.4.6.1 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**。

PCI-1285E 不支持该 API。

6.3.4.6.2 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**。

PCI-1285E 不支持该 API。

6.3.4.6.3 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。

PCI-1285E 不支持该 API。

6.3.4.6.4 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。
PCI-1285E 不支持该 API。

6.3.4.6.5 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。
PCI-1285E 不支持该 API。

6.3.4.7 原点

6.3.4.7.1 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。

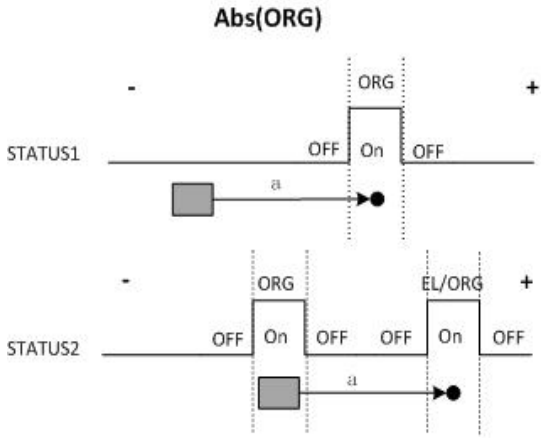
c：以 VelLow 进行等速运动，遇到 ORG/EL 信号后立即停止。

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)。*

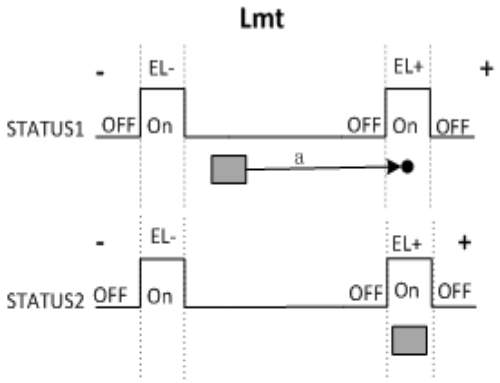
1. MODE1_Abs: Move (Dir) ->touch ORG->Stop.
   只依照原点设备 （如传感器）返回原点。对象会一直运动，直至原点信号发生。
   比如：
   Dir：正。
   Org Logic (CFG_AxOrgLogic)：高准位。
   EL （硬限位开关）逻辑 (CFG_AxElLogic)：高准位。



- 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。

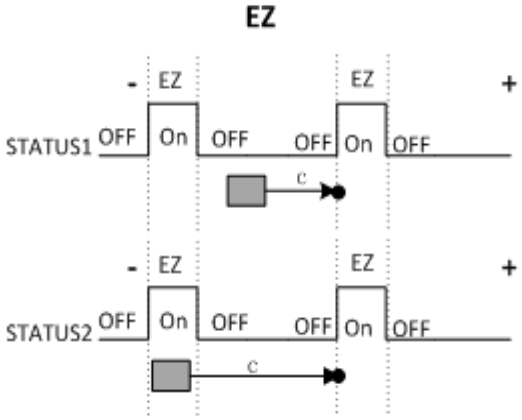
2. MODE2_Lmt: Move(Dir)->touch EL->Stop
   只依照限位设备 （如传感器）返回原点。对象会一直运动，直至限位信号发生。

比如：
Dir：正。
限位逻辑 (CFG_AxElLogic)：高准位。



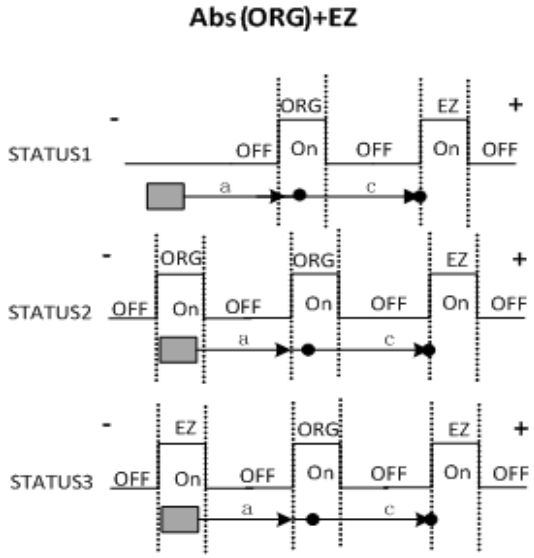
- STATUS1：如果对象超出 EL 信号区域，当写入原点命令时，对象将一直运动直到 EL 信号发生。
- STATUS2：如果对象在 EL 信号区域内，将不作出响应。

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。

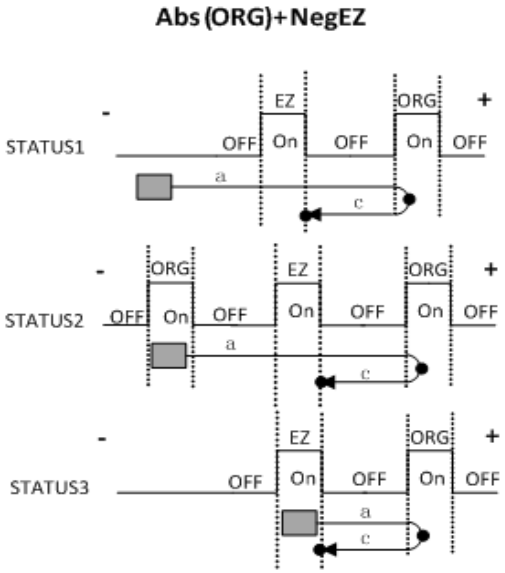
4. MODE4_Abs_Ref: ORG+EZ, Move(Dir) ->touch ORG ->Stop ->Move(Dir)->touch EZ ->Stop
这是一种组合的原点模式。首先，对象将一直运动直到原始信号发生，然后将保持以 ORG 同一方向继续运动，直到 EZ 信号发生。
比如：
Dir：正。
ORG 逻辑：高准位。
EZ 逻辑：高准位。



- 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。

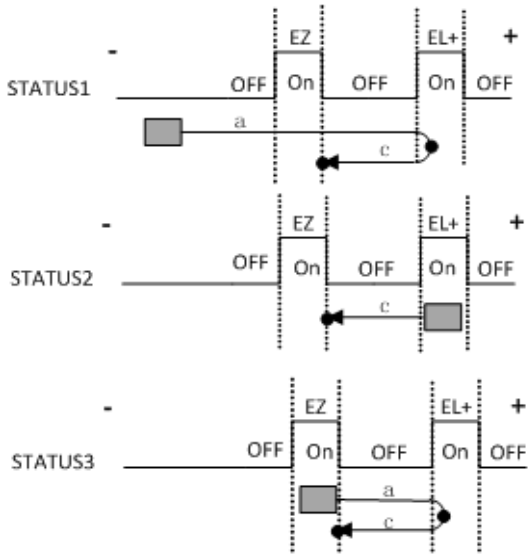


- STATUS1：如果对象超出 EZ 信号和 ORG 信号区域，当写入原点命令时：首先，对象会一直运动直到 ORG 信号发生，然后继续以相反方向运动，直到 EZ 信号发生。
- STATUS2：如果对象在 ORG 信号区域内，写入原点命令时，对象开始运动。首先，ORG 信号消失，然后下一个 ORG 信号发生，同时倒转运动方向。最后，当 EZ 信号发生时运动停止。
- STATUS3：如果对象在 EZ 信号区域内，写入原点命令时，对象开始运动。首先，EZ 信号消失，然后 ORG 信号发生，同时倒转运动方向。最后，当 EZ 信号发生时运动停止。

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。

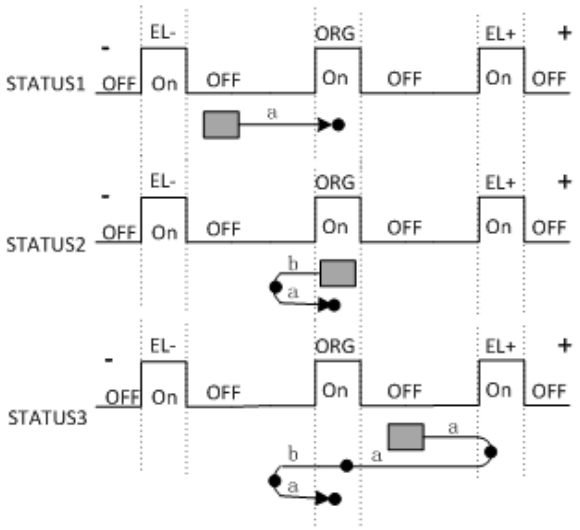
Lmt+EZ



- 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。

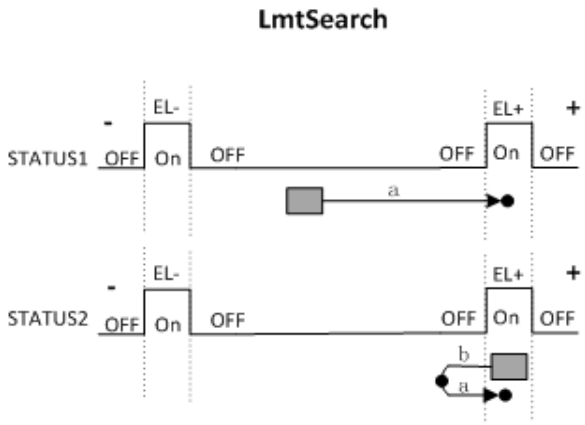
7. MODE7_AbsSearch: Move (Dir) ->Search ORG ->Stop.
这是一种搜索 ORG 信号从无到有转换的模式。
比如：
Dir：正。
ORG 逻辑：高准位。
限位逻辑：高准位。

AbsSearch



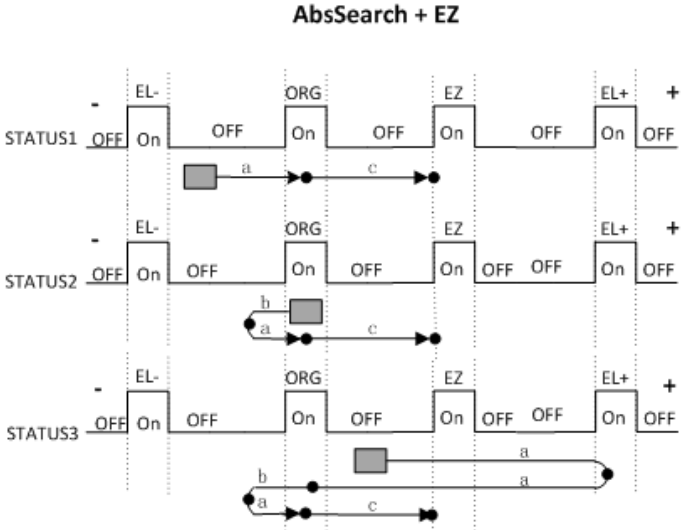
- STATUS1：如果没有 ORG 信号发生，则 ORG 信号发生时对象将停止运动。
- STATUS2：如果对象在 ORG 信号区域内， 对象以相反方向运动直到信号消失，然后转换方向继续运动，直到 ORG 信号发生。
- STATUS3：如果没有 ORG 信号发生，在运动时 EL 信号首先发生，对象倒转方向并继续运动，然后 ORG 信号将从有到无。然后，再次倒转方向并运动，直到 ORG 信号发生， 运动停止。

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。



- 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。

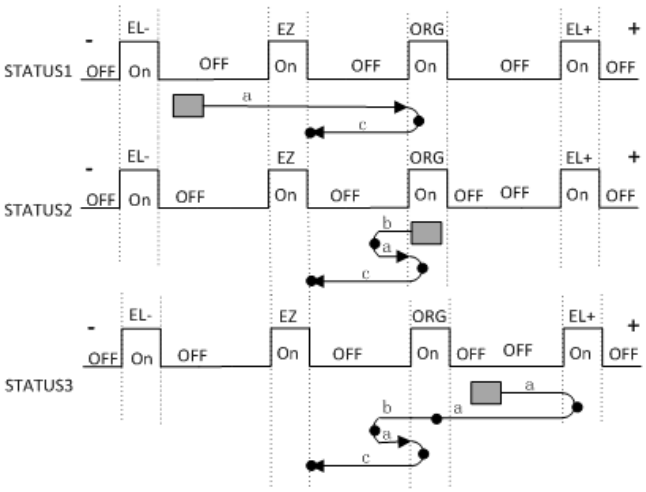
9. MODE9_AbsSearch_Ref: Search ORG + EZ, Move (Dir) ->Search ORG ->Stop ->Move (Dir) ->touch EZ ->Stop.
首先，对象以 MODE7_AbsSearch 方式运动，然后以相同方向运动直到 EZ 信号发生。
比如：
Dir：正。
限位逻辑：高准位。
ORG 逻辑：高准位。



- 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。

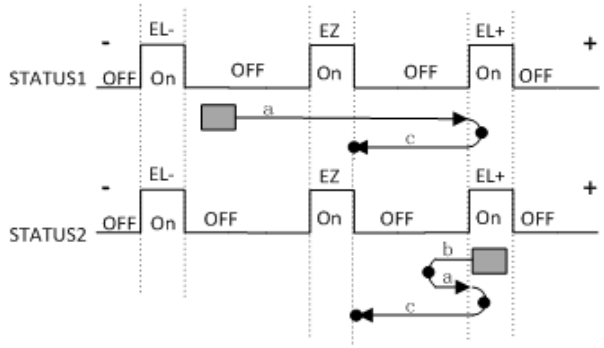
AbsSearch + NegEZ



- 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。

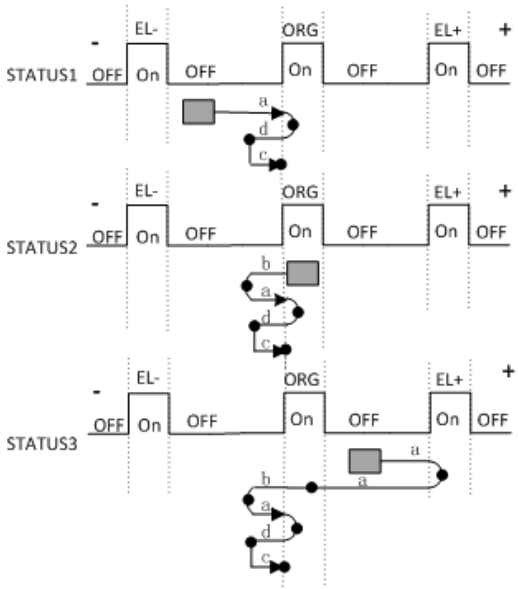
LmtSearch + NegEZ



- 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。

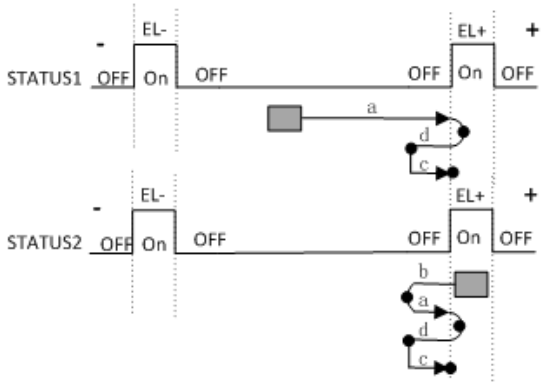
AbsSearchReFind



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

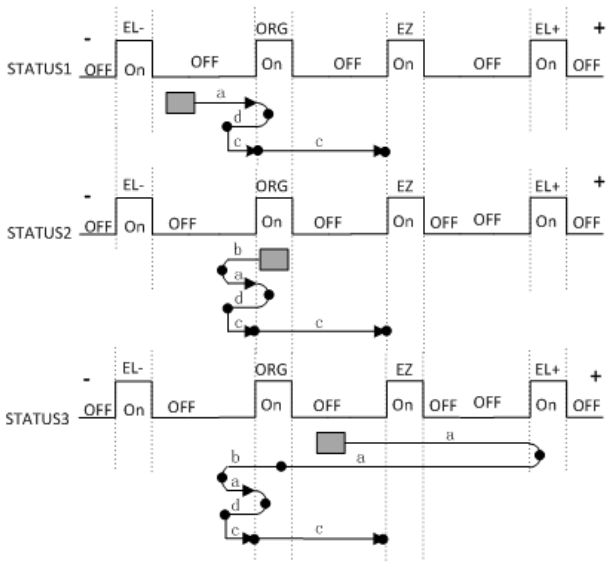
13. MODE13_ LmtSearchRefind: Search EL +Refind EL, Move (Dir) ->Search EL ->Stop->Move (-Dir) ->Leave EL(FL) ->Stop-> Move (-Dir)->Refind EL(FL)->Stop.
首先，轴以 MODE8_LmtSearch 模式运动；然后轴反向以低速 （VelLow）等速运动直至 EL 信号消失；然后轴再次反向以低速 （VelLow）等速运动直至 EL 信号发生。
比如：
Dir：正。
限位逻辑：高准位。

LmtSearchReFind



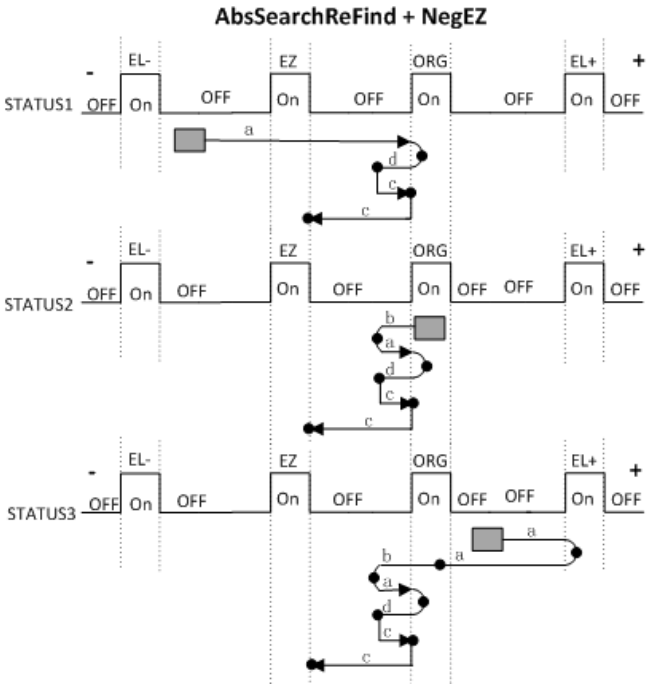
14. MODE14_AbsSearchRefind_Ref: Search ORG +Refind ORG+EZ, Move (Dir) ->Search ORG ->Stop->Move (-Dir) ->Leave ORG(FL) ->Stop-> Move (-Dir)->Refind ORG(FL)->Stop->Move (Dir) ->touch EZ ->Stop.
首先，轴以 MODE7_AbsSearch 模式运动；然后轴反向以低速 （VelLow）等速运动直至 ORG 信号消失；然后轴再次反向以低速 （VelLow）等速运动直至 ORG 信号发生；最后轴沿相同方向運動至找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

AbsSearchReFind + EZ



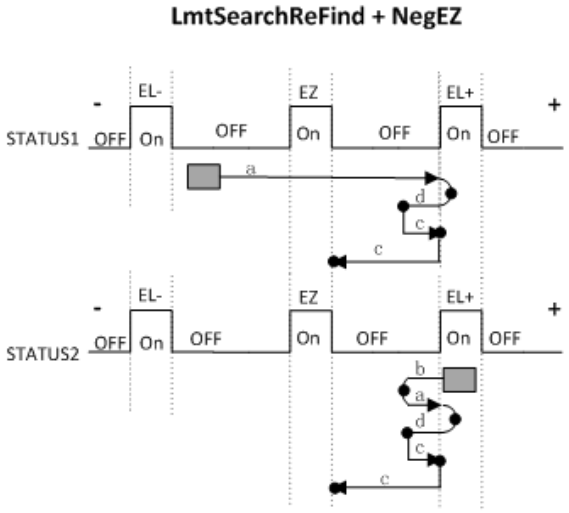
AbsSearch 过程有三种状况，具体请参考 MODE7_AbsSearch 中的描述。

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。

6.3.4.8 位置 / 计数器控制

6.3.4.8.1 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：

6.3.4.8.2 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：

6.3.4.8.3 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：

6.3.4.8.4 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：

6.3.4.9 比较

6.3.4.9.1 Acm_AxSetCmpData

格式：

U32 Acm_AxSetCmpData (HAND AxisHandle, F64 CmpPosition)

目的：

设置指定轴的比较数据。

参数：

名称	类型	IN 或 OUT	说明
AxisHandle	HAND	IN	来自 Acm_AxOpen 的轴句柄。
CmpPosition	F64	IN	需要比较的数据。（单位：PPU）

返回值：

错误代码

注解：

如果属性 CFG_AxCmpMethod 设置为 **MTD_GREATER_POSITION**，**CmpPosition** 应大于当前位置 （理论位置或实际位置）。

如果属性 CFG_AxCmpMethod 设置为 **MTD_SMALLER_POSITION**，**CmpPosition** 应小于当前位置 （理论位置或实际位置）。

设置比较数据前，用户需要首先将 CFG_AxCmpEnable 设置为 **CMP_ENABLE**。 如果用户想要关闭比较功能，只需将属性CFG_AxCmpEnable设置为**CMP_DISABLE**，而无需清除比较数据。用户可将 CFG_AxCmpSrc 设置为 **SRC_COMMAND_POSITION** 或 **SRC_ACTUAL_POSITION**。

调用 Acm_AxSetCmpData、Acm_AxSetCmpAuto 和 Acm_AxSetCmpTable 的任一函数将清除之前比较数据。

如果启用属性 CFG_AxEnableGenDO，该功能将禁用。

PCI-1285E 不支持该 API。

6.3.4.9.2 Acm_AxSetCmpTable

格式：

U32 Acm_AxSetCmpTable (HAND AxisHandle, PF64 TableArray,
I32 ArrayCount)

目的：

设置指定轴的比较数据列表。

参数：

名称	类型	IN 或 OUT	说明
AxisHandle	HAND	IN	来自 Acm_AxOpen 的轴句柄。
TableArray	PF64	IN	比较数据表。（单位：PPU）
ArrayCount	I32	IN	表中的比较数据个数。

返回值：

错误代码

注解：

如果属性 CFG_AxCmpMethod 设置为 **MTD_GREATER_POSITION**，表中的首个数据应大于当前位置（理论位置或实际位置），且比较数据应大于表中的上一个比较数据。比较表设置之后，首个数据将加载至比较器。如果当前位置与比较器匹配，将产生比较信号，下一个比较数据将自动加载至比较器。如果属性 CFG_AxCmpMethod 设置为**MTD_SMALLER_POSITION**，表中的首个数据应小于当前位置（理论位置或实际位置），且比较数据应小于表中的上一个比较数据。首个数据将加载至比较器，如果当前位置与比较器匹配，将产生比较信号，下一个比较数据将自动加载至比较器。

设置比较数据前，用户需要首先将 CFG_AxCmpEnable 设置为 **CMP_Enable**。如果用户想要关闭比较功能，只需将属性CFG_AxCmpEnable设置为CMP_Disable，而无需清除比较数据。用户可将 CFG_AxCmpSrc 设置为 SRC_COMMAND_POSITION 或 SRC_ACTUAL_POSITION。

调用 Acm_AxSetCmpData、Acm_AxSetCmpAuto 和 Acm_AxSetCmpTable 其中的任一函数将清除之前比较数据。

最多有 100,000 个比较数据。

目前该功能只支持 X 轴和 Y 轴。

如果启用 CFG_AxGenDoEnable，比较功能将禁用，因此将不会输出比较信号。

PCI-1285E 不支持该 API。

6.3.4.9.3 Acm_AxSetCmpAuto

格式：

```
U32 Acm_AxSetCmpAuto (HAND AxisHandle, F64 Start, F64 End,
F64 Interval)
```

目的：

设置指定轴的线性比较数据。

参数：

名称	类型	IN 或 OUT	说明
AxisHandle	HAND	IN	来自 Acm_AxOpen 的轴句柄。
Start	F64	IN	首个比较数据。（单位：PPU）
End	F64	IN	最后比较的数据。（单位：PPU）
Interval	F64	IN	比较间隔。（单位：PPU）

返回值：

错误代码

注解：

如果属性 CFG_AxCmpMethod 设置为 **MTD_GREATER_POSITION**，Start 数据应大于当前位置 （理论位置或实际位置）。 首个数据将加载至比较器，如果当前位

置与比较器匹配，将产生比较脉冲，下一个比较数据将自动加载至比较器。如果 属性 CFG_AxCmpMethod 设置为 **MTD_SMALLER_POSITION**，Start 数据应小于当前位置 （理论位置或实际位置）。 首个数据将加载至比较器，如果当前位置与比较器匹配，将产生比较脉冲，下一个比较数据将自动加载至比较器。

设置比较数据前，用户需要首先将 CFG_AxCmpEnable 设置为 **CMP_Enable**。如果用户想要关闭比较功能，只需将属性CFG_AxCmpEnable设置为**CMP_Disable**，而无需清除比较数据。用户可将 **CFG_AxCmpSrc** 设置为 SRC_COMMAND_POSITION 或 SRC_ACTUAL_POSITION。

调用 Acm_AxSetCmpData、Acm_AxSetCmpAuto 和 Acm_AxSetCmpTable 其中的任一函数将清除之前比较数据。

最多有 100,000 个比较数据。

目前该功能只支持 X 轴和 Y 轴。

如果启用 CFG_AxGenDoEnable，比较功能将禁用，因此将不会输出比较信号。

在此模式下，Compare Event 只发生在第一次触发比较的情况下。

PCI-1285E 不支持该 API。

6.3.4.9.4 Acm_AxGetCmpData

格式：

U32 Acm_AxGetCmpData (HAND AxisHandle, PF64 CmpPosition)

目的：

获取比较器中的当前比较数据。

参数：

名称	类型	IN 或 OUT	说明
AxisHandle	HAND	IN	来自 Acm_AxOpen 的轴句柄。
CmpPosition	PF64	OUT	比较数据。（单位：PPU）

返回值：

错误代码

注解：

PCI-1285E 不支持该 API。

6.3.4.10锁存

6.3.4.10.1 Acm_AxGetLatchData

格式：

U32 Acm_AxGetLatchData (HAND AxisHandle, U32 PositionNo, F64 Position)

目的：

触发锁存后获取设备中的锁存数据。

参数：

名称	类型	IN 或 OUT	说明
AxisHandle	HAND	IN	来自 Acm_AxOpen 的轴句柄。
PositionNo	U32	IN	0：理论位置 1：实际位置
Position	PF64	OUT	锁存数据。（单位 = PPU）

返回值：

错误代码

注解：

PCI-1285E 不支持该 API。

6.3.4.10.2 Acm_AxTriggerLatch

格式：

U32 Acm_AxTriggerLatch (HAND AxisHandle)

目的：

触发命令以锁存位置数据。

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：

PCI-1285E 不支持该 API。

6.3.4.10.3 Acm_AxResetLatch

格式：

U32 Acm_AxResetLatch (HAND AxisHandle)

目的：

清除设备中的锁存数据和锁存标记。

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：

PCI-1285E 不支持该 API。

6.3.4.10.4 Acm_AxGetLatchFlag

格式：

U32 Acm_AxGetLatchFlag (HAND AxisHandle, PU8 LatchFlag)

目的：

获取设备中的锁存标记。

参数：

名称	类型	IN 或 OUT	说明
AxisHandle	HAND	IN	来自 Acm_AxOpen 的轴句柄。
LatchFlag	PU8	OUT	标记数据。
			1：数据锁存。
			0：数据未锁存。

返回值：

错误代码

注解：

PCI-1285E 不支持该 API。

6.3.4.11 Aux/Gen 输出

6.3.4.11.1 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，CamDo、Erc 和 Compare 功能将自动禁用。由于两种功能使用同一输出针脚 （OUT4 ~ OUT7），因此 Acm_AxSetCmpData、Acm_AxSetCmpAuto 和 Acm_AxSetCmpTable 将不能生成触发脉冲。

6.3.4.11.2 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。

6.3.4.11.3 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：

6.3.4.12外部驱动

6.3.4.12.1 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：

6.3.4.13凸轮 / 齿轮

6.3.4.13.1 Acm_AxCamInAx

格式：

U32 Acm_AxCamInAx (HAND AxisHandle, HAND MasAxisHandle,
F64 MasterOffset, F64 SlaveOffset, F64 MasterScaling,
F64 SlaveScaling, U32 CamTableID, U32 RefSrc)

目的：

该函数使用 CAM 表开始从 （其它）轴和主 （首个）轴之间的 CAM 同步。

Camming 是一个表内完成 （二维 - 描述主和从位置）。 表应严格为单调上升或下降，随着主轴同时向相反和相同方向发展。

参数：

名称	类型	IN 或 OUT	说明
AxisHandle	HAND	IN	来自 Acm_AxOpen 的轴句柄，该句柄应为从 （其它）轴句柄。
MasAxisHandle	HAND	IN	来自 Acm_AxOpen 的轴句柄，该句柄应为主 （首个）轴句柄。
MasterOffset	F64	IN	主轴方向的坐标偏移值。
SlaveOffset	F64	IN	从轴方向的坐标偏移值。
MasterScaling	F64	IN	主轴坐标中 CAM 的比例因子。整个主轴坐标乘以该比例因子，结果应大于 0。
SlaveScaling	F64	IN	从轴坐标中 CAM 的比例因子。整个从轴坐标乘以该比例因子，结果应大于 0。
CamTableID	U32	IN	CAM 表的标识符。由 Acm_DevDownloadCAMTable 分配。PCI-1245 和 PCI-1265 保留了 2 个 CAM 表。因此 ID 为 0 和 1。

RefSrc	U32	IN	CAM 表参考位置： 0：理论位置； 1：实际位置。

返回值：

错误代码

注解：

如果该命令设置成功，当主轴正在进行连续运动或点到点运动时，从轴会按照 CamTable 拟合出来的 CAM 曲线移动。

不能通过 Acm_AxSetCmdPosition 或 Acm_AxSetAcutalPosition 设置从轴和主轴的理论 / 实际位置。

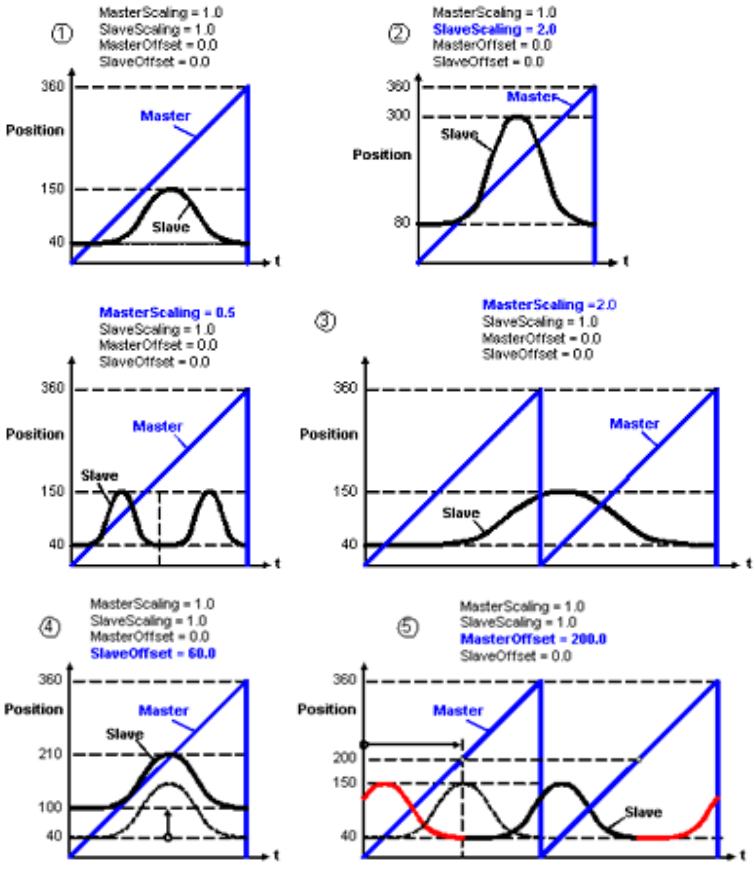
用户可通过调用 Acm_AxStopDec 和 Acm_AxStopEmg 终止从轴的跟从关系，且从轴将为 Readby 状态。

主轴旋转 360° 的脉冲个数应由属性 CFG_AxModuleRange 设置。编辑过的 CamTable 需要由 Acm_DevDownloadCAMTable 设置，且相关的 E-cam 配置需要通过 Acm_DevConfigCAMTable 设置。

综上，如果成功调用 **Acm_AxCamInAx**，则从轴将为同步状态。 然后，从轴将跟随主轴运动。

参数 **MasterScaling**、**SlaveScaling**、**MasterOffset** 和 **SlaveOffset** 用于按照编辑过的 CamTable 调整当前 Camtable。

有关比例和偏移的图片如下所示：



有关 E-CAM 操作的详细信息，请参考第 6.2.2 节的 **E-CAM 流程图**。

请参考 Acm_DevDownLoadCAMTable、Acm_DevCofigCAMTable 和 Acm_DevLoadCAMTableFile。

PCI-1285/1285E 不支持该 API。

6.3.4.13.2 Acm_AxGearInAx

格式：

U32 Acm_AxGearInAx (HAND AxisHandle, HAND MasAxisHandle, I32 Numerator, I32 Denominator, U32 RefSrc, U32 Absolute)

目的：

该函数按照比例开始从 （其它）轴和主 （首个）轴之间的 Gear 同步。

参数：

名称	类型	IN 或 OUT	说明
AxisHandle	HAND	IN	来自 Acm_AxOpen 的轴句柄，该句柄应为从 （其它）轴句柄。
MasAxisHandle	HAND	IN	来自 Acm_AxOpen 的轴句柄，该句柄应为主 （首个）轴句柄。
Numerator	I32	IN	齿轮比的分子。
Denominator	I32	IN	齿轮比的分母。
RefSrc	U32	IN	从轴参考主轴的理论位置 （0）或实际位置 （1）。
Absolute	U32	IN	同步关系是相对的还是绝对的。 0：相对； 1：绝对。

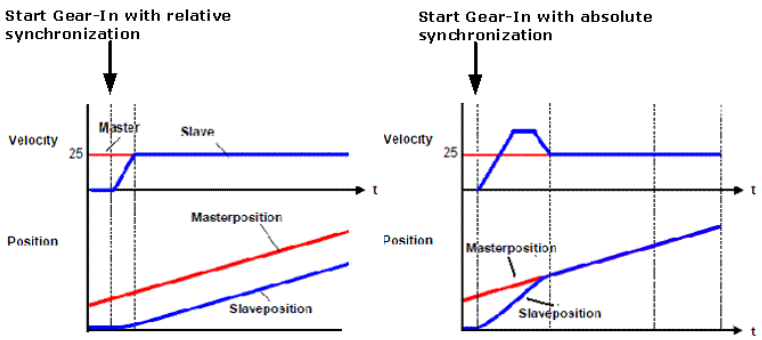
返回值：

错误代码

注解：

如果该函数调用成功，从轴将跟随主轴运动。在 Gear 运动中，不能通过 Acm_AxSetCmdPosition 或 Acm_AxSetActualPosition 复位主轴和从轴的理论 / 实际位置。 通过调用 Acm_AxStopDec 和 Acm_AxStopEmg 可终止主轴和从轴的关系，且轴将变为 SteadBy 状态。

- **齿轮比：**分子 / 分母。如果值为正数，那么从轴将以与主从相同的方向运动；否则，从轴将以与主轴相反的方向运动。
- **绝对：**
  - Absolute=1：绝对关系。从轴将补偿主轴的偏移。 在这种模式下，如果起始位置不为 0，则从轴的加速度和减速需大于 1000 pps。
  - Absolute=0：相对关系。从轴将不会补偿主轴的偏移。



有关 E-Gear 的操作信息，请参考第 6.2.2 节的 **E-Gear/Gantry 流程图**。

6.3.4.13.3 Acm_AxPhaseAx

格式：

U32 Acm_AxPhaseAx(HAND AxisHandle, F64 Acc, F64 Dec, F64 PhaseSpeed, F64 PhaseDist)

目的：

在电子凸轮或电子齿轮过程中使从轴进行相位超前或落后运动。

参数：

名称	类型	IN 或 OUT	说明
AxisHandle	HAND	IN	来自 Acm_AxOpen 的轴句柄。
Acc	F64	IN	相位超前 / 落后运动的加速度，单位：PPU/s^2
Dec	F64	IN	相位超前 / 落后运动的减速度，单位：PPU/s^2
PhaseSpeed	F64	IN	相位超前 / 落后运动的速度，单位：PPU/s
PhaseDist	F64	IN	相位超前 / 落后运动的距离，单位：PPU

返回值：

错误代码

注解：

在电子凸轮或电子齿轮过程中，通过此 API 可使从轴超前 / 落后原先的运动，当 PhaseDist>0，则进行相位超前运动，当 PhaseDist<0，则进行相位落后运动。

若剩余 Pulse 不足，则从轴无法达到指定的相位。且当相位超前 / 落后运动未结束之前，再次下达此命令，则返回错误。

注！ *因浮点数计算误差，从轴通过属性 CFG_AxMaxAcc/CFG_AxMaxDec 设定的最大加速度 / 最大减速度的值须比 Acc/Dec 至少大 100,000。*

6.3.4.14 龙门 / 切线跟随

6.3.4.14.1 Acm_AxTangentInGp

格式：

U32 Acm_AxTangentInGp (HAND AxisHandle, HAND MasGroupHandle, PI16 StartVectorArray, U8 Working_plane, I16 Direction)

目的：

命令轴按照与群组路径切线相同的方向运动。

参数：

名称	类型	IN 或 OUT	说明
AxHandle	HAND	IN	来自 Acm_AxOpen 的轴句柄。
MasGroupHandle	HAND	IN	来自 Acm_GpAddAxis 的群组句柄。
StartVectorArray	PI16	IN	必须为三维。
Working_plane	U8	IN	0：XY； 1：YZ； 2：XZ。
Direction	I16	IN	相同：0； 相反：1

返回值：

错误代码

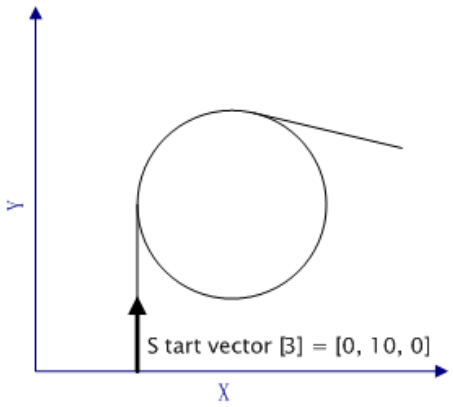
注解：

如果该函数调用成功，轴将为同步状态，且按照与群组路径切线相同的方向跟随群组运动。轴和群组不能为复位理论 / 实际位置。通过 Acm_AxStopDec/Acm_AxStopEmg可终止轴的同步状态，且将变为SteadBy状态。

跟随轴不能为群组中的一个轴。

StartVectorArray 为初始向量，也是轴的起始方向。

比如，StartVectorArray[3] = {0, 10, 0}，Working plane = 0：XY 平面。



注！

起始向量必须尽可能与群组运动的起始方向接近，否则可能出现运动加速大于设备最大加速的错误。 如果两个连接路径之间的角度过大，也会发生错误，因此用户须注意路径之间的角度。 计算最大角度偏差的公式如下：

比如：

设置：

模数范围 （CFG_AxModuleRange）：3600 个脉冲。

最大加速 （CFG_AxMaxAcc）：10 ^ 7。

斜率转换的最大角度：

10^7 X 10^-6 X 360° /3600 = 1

PCI-1285E 不支持该 API。

6.3.4.14.2 Acm_AxGantryInAx

格式：

U32 Acm_AxGantryInAx (HAND AxisHandle, HAND MasAxisHandle, I16 RefMasterSrc, I16 Direction)

目的：

命令两个轴进行 E-Gantry 运动。

参数：

名称	类型	IN 或 OUT	说明
AxisHandle	HAND	IN	来自 Acm_AxOpen 的轴句柄。
MasAxisHandle	HAND	IN	来自 Acm_GpAddAxis 的轴句柄。
RefMasterSrc	I16	IN	参考源： 0：理论位置； 1：实际位置 （预留）。

Direction	I16	IN	和主轴的方向： 0：相同； 1：相反。

返回值：

错误代码

注解：

从轴将于主轴同步运动。 通过调用 Acm_AxStopDec 或 Acm_AxStopEmg 可终止主轴和从轴的关系，且轴将变为 SteadBy 状态。

Gantry 有如下几种限制：

1. 不能给从轴设置任何运行命令；
2. 从轴不能添加到任何群组中；
3. 如果轴已经为群组中的一个轴，那么该轴不能作为 Gantry 的从轴。

如果复位主轴的理论/实际位置，则从轴的命令/实际位置也应复位至相同值。

有关 Gantry 的操作信息，请参考第 6.2.2 节的 **E-Gear/Gantry 流程图**。

PCI-1285E 不支持该 API。

6.3.4.15停止运动

6.3.4.15.1 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-cam/E-gear/Tangent 运动中的从轴，该 API 能够用于停止同步关系。

6.3.4.15.2 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-cam/E-gear/Tangent 运动中的从轴，该 API 能够用于停止同步关系。

6.3.4.15.3 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：

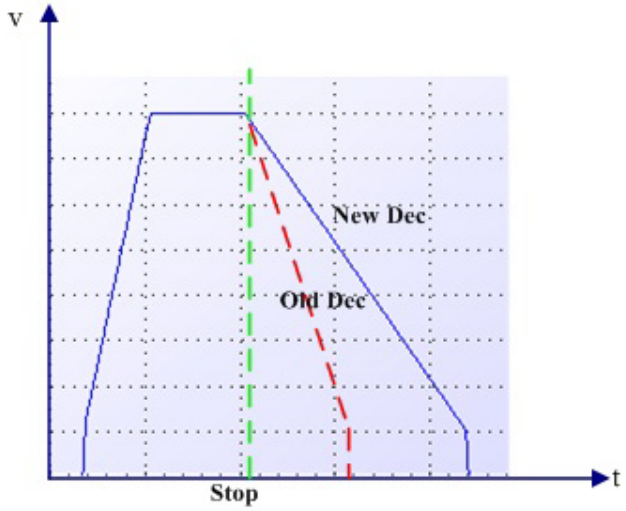
名称	类型	IN 或 OUT	说明
AxisHandle	HAND	IN	来自 Acm_AxOpen 的轴句柄。
NewDec	F64	IN	减速停止时的减速度，单位：PPU/s^2

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。



6.3.5 群组

6.3.5.1 系统

6.3.5.1.1 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-1285/1285E 最多有 4 个群组。PCI-1285 每个群组最多有 8 个轴，PCI-1285E 每个群组最多有 2 个轴。

相同的轴不能添加到不同的群组中。

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。

6.3.5.1.2 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。

6.3.5.1.3 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。

6.3.5.1.4 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。

6.3.5.2 运动状态及速度

6.3.5.2.1 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。

6.3.5.2.2 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。

6.3.5.3 运动停止

6.3.5.3.1 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：

6.3.5.3.2 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：

6.3.5.4 插补运动

6.3.5.4.1 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 （具有最小 Physical ID 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-1285 最多只支持 3 个轴线性插补，PCI-1285E 最多只支持 2 个轴线性插补。

6.3.5.4.2 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 （具有最小 Physical ID 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-1285 最多只支持 3 个轴线性插补，PCI-1285E 最多只支持 2 个轴线性插补。

6.3.5.4.3 Acm_GpMoveCircularRel

格式：

U32 Acm_GpMoveCircularRel (HAND GroupHandle, PF64 CenterArray, PF64 EndArray, PU32 pArrayElements, I16 Direction)

目的：

命令群组执行相对 ARC 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。
CenterArray	PF64	IN	轴相对于圆心点的相对距离。
EndArray	PF64	IN	轴相对于终止点的相对距离。
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）。
Direction	I16	IN	方向： 0：DIR_CW （顺时针）； 1：DIR_CCW （逆时针）。

返回值：

错误代码

注解：

CenterArray 和 **EndArray** 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 Y 轴和 U 轴，那么 **CenterArray** 中的第一个数据表示 Y 轴的圆心距离，第二个数据表示 U 轴的圆心距离。**CenterArray** 和 **EndArray** 中距离的单位为群组中每个轴的 PPU。

PCI-1285 最多只支持 2 个轴 ARC 插补。PCI-1285E 不支持该 API。

6.3.5.4.4 Acm_GpMoveCircularAbs

格式：

U32 Acm_GpMoveCircularAbs (HAND GroupHandle, PF64 CenterArray, PF64 EndArray, PU32 pArrayElements, I16 Direction)

目的：

命令群组执行绝对 ARC 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。
CenterArray	PF64	IN	圆心点的绝对距离。
EndArray	PF64	IN	终止点的绝对距离。
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）。
Direction	I16	IN	方向： 0：DIR_CW （顺时针）； 1：DIR_CCW （逆时针）。

返回值：

错误代码

注解：

CenterArray 和 **EndArray** 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 Y 轴和 U 轴，那么 **CenterArray** 中的第一个数据表示 Y 轴的圆心位置，第二个数据表示 U 轴的圆心位置。**CenterArray** 和 **EndArray** 中距离的单位为群组中每个轴的 PPU。

PCI-1285 最多只支持 2 个轴 ARC 插补。PCI-1285E 不支持该 API。

6.3.5.4.5 Acm_GpMoveCircularRel_3P

格式：

U32 Acm_GpMoveCircularRel_3P (HAND GroupHandle, PF64 RefArray, PF64 EndArray, PU32 pArrayElements, I16 Direction)

目的：

命令群组通过三个指定点执行相对 ARC 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。
RefArray	PF64	IN	参考点的相对距离。
EndArray	PF64	IN	终止点的相对距离。
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）。
Direction	I16	IN	方向： 0：DIR_CW （顺时针）； 1：DIR_CCW （逆时针）。

返回值：

错误代码

注解：

RefArray 和 **EndArray** 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 Y 轴和 U 轴，那么 **RefArray** 中的第一个数据表示 Y 轴的参考距离，第二个数据表示 U 轴的参考距离。**RefArray** 和 **EndArray** 中距离的单位为群组中每个轴的 PPU。

PCI-1285 最多只支持 2 个轴 ARC 插补。PCI-1285E 不支持该 API。

6.3.5.4.6 Acm_GpMoveCircularAbs_3P

格式：

U32 Acm_GpMoveCircularAbs_3P (HAND GroupHandle, PF64 RefArray, PF64 EndArray, PU32 pArrayElements, I16 Direction)

目的：

命令群组通过三个指定点执行绝对 ARC 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。
RefArray	PF64	IN	参考点的绝对位置。
EndArray	PF64	IN	终止点的绝对位置。
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）。
Direction	I16	IN	方向： 0：DIR_CW （顺时针）； 1：DIR_CCW （逆时针）。

返回值：

错误代码

注解：

RefArray 和 **EndArray** 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 Y 轴和 U 轴，那么 **RefArray** 中的第一个数据表示

Y 轴的参考位置，第二个数据表示 U 轴的参考位置。**RefArray** 和 **EndArray** 中距离的单位为群组中每个轴的 PPU。

PCI-1285 最多只支持 2 个轴 ARC 插补。PCI-1285E 不支持该 API。

6.3.5.4.7 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 （具有最小 Physical ID 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-1285 最多只支持 8 个轴直接插补，PCI-1285E 最多只支持 2 个轴直接插补。

6.3.5.4.8 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 （具有最小 Physical ID 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-1285 最多只支持 8 个轴直接插补，PCI-1285E 最多只支持 2 个轴直接插补。

6.3.5.4.9 Acm_GpChangeVel

格式：

U32 Acm_GpChangeVel (HAND GroupHandle, F64 NewVelocity)

目的：

命令群组在插补运动时改变速度。

参数：

名称	类型	IN 或 OUT	说明
GpHandle	HAND	IN	来自 Acm_GpAddAxis 的群组句柄。
NewVelocity	F64	IN	新速度。（单位：PPU/s）

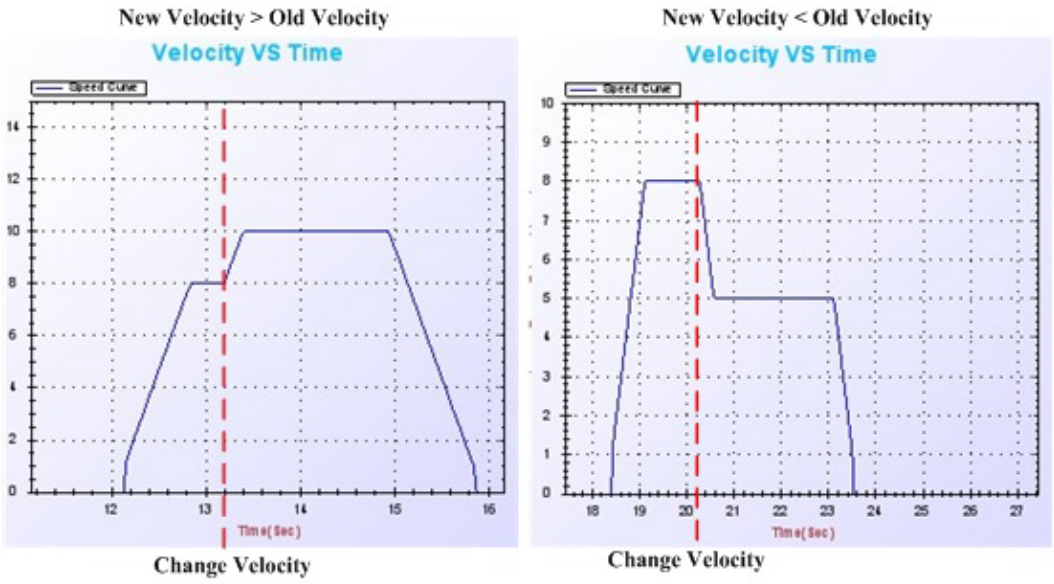
返回值：

错误代码

注解：

当群组在运动的过程中，可以对其下达改变速度命令，运动卡在收到此命令后会根据情况作出改变速度的动作。当群组在执行线性插补、圆弧插补、螺旋插补以及连续插补等动作时，可以通过此命令改变运动的速度。若此命令成功下达，如果在下次运动之前未进行重新设定速度，NewVelocity 会作用到下次的运动中。

1. 当执行单步插补动作时，速度改变情形如下：



如果剩余的Pulse数不足以支持达到设定的新速度，则板卡会自行计算能达到的速度值。

2. 当执行连续插补时

■BufferMode：Blending Disabled

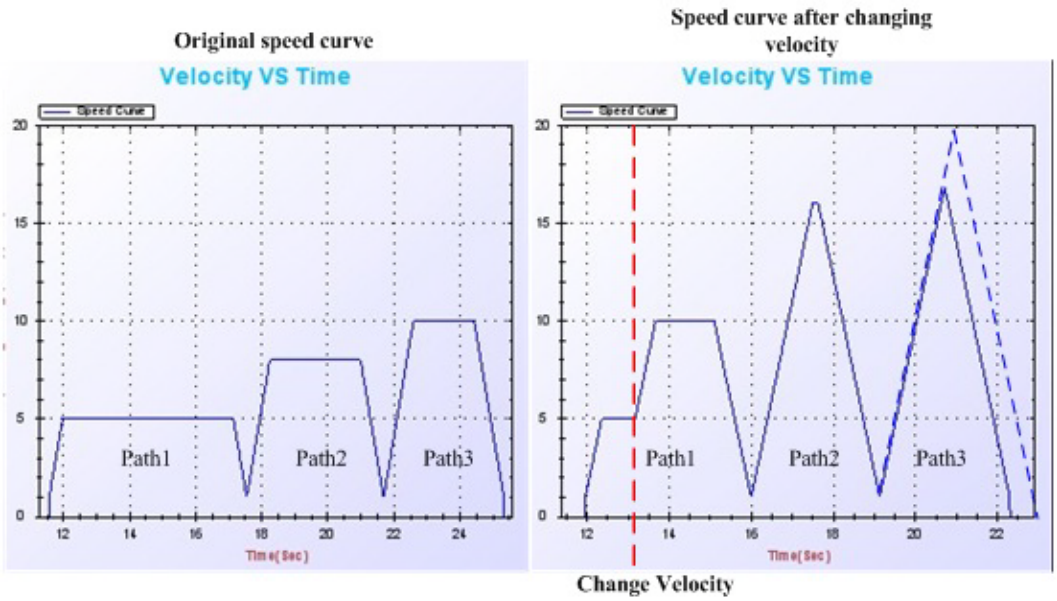
在此模式下，Path Buffer 中的各段 Path，都有自己完整的加减速过程，在运动的过程中，如果执行 ChangeV 操作，则当前执行段的 Path 的速度会加速 / 减速到新速度，则以后各段的速度会成比例增加 / 减少。各段在改变速度时，如果段剩余的 Pulse 不够，则自动计算能达到的新速度。

例如：path1：VL = 1000，VH = 5000

path 2：VL = 1000，VH = 8000

path 3：VL = 1000，VH = 10000

当执行 Path1 的过程中，下达 ChangeV，New Velocity = 10000，则第二段 path 的速度应该为 16000，第三段的速度应该为 20000。但是实际执行情况如下图所示：



■BlendingMode：Blending Enable，BlendingTime >0

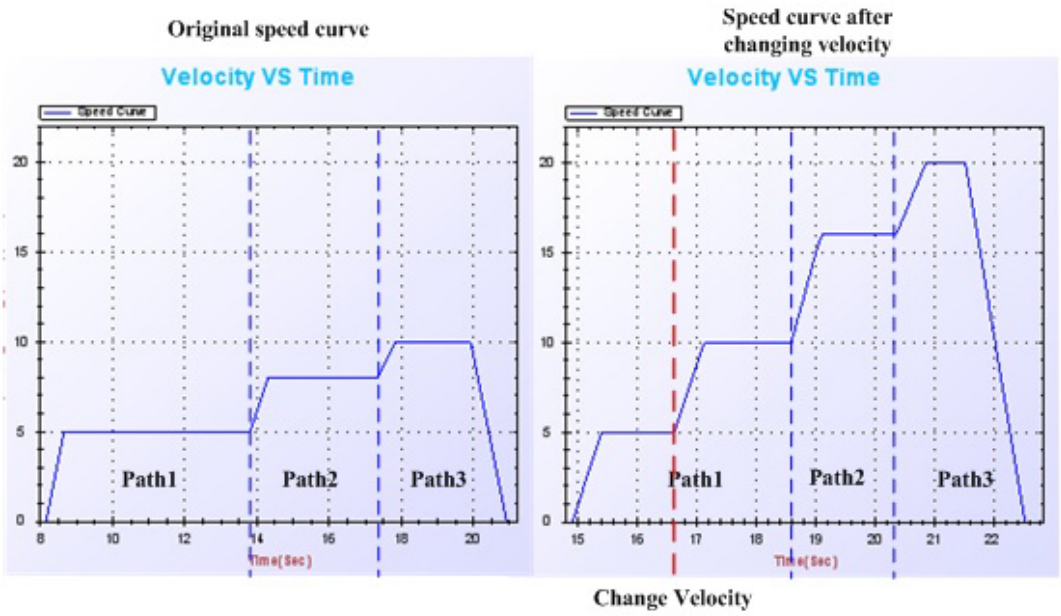
在此模式下，Path Buffer 中的各段 Path，没有完整的加减速过程，其速度轨迹由 BlendingTime 以及各段的 FL 决定，详细请参阅 Acm_GpAddPath。在运动的过程中，如果执行 ChangeV 操作，则当前执行段的 Path 的速度会加速 / 减速到新速度，则以后各段的速度会成比例增加 / 减少。各段在改变速度时，如果段剩余的 Pulse 不够，则自动计算能达到的新速度。

例如：path1：VL = 0，VH = 5000

path 2：VL = 0，VH = 8000

path 3：VL = 0，VH = 10000

当执行 Path1 的过程中，下达 ChangeV，New Velocity = 10000，则第二段 path 的速度应该为 16000，第三段的速度应该为 20000。如下图所示：



如果 Change velocity 命令下达在 Blending 阶段，则 ChangeV 推迟到下一段 Path 执行。

■FlyMode：Blending Enable，BlendingTime=0

此模式下的速度轨迹与 Blending 模式下类似，只是其速度轨迹中两段 Path 之间不会减速到 FL。详细请参阅 Acm_GpAddPath。在运动的过程中，如果执行 ChangeV 操作，则当前执行段的 Path 的速度会加速 / 减速到新速度，则以后各段的速度会成比例增加 / 减少。各段在改变速度时，如果段剩余的 Pulse 不够，则自动计算能达到的新速度。

PCI-1285E 不支持该 API。

注！ *减速段不支持 ChangeV 操作。*

6.3.5.4.10 Acm_GpMoveHelixAbs

格式：

U32 Acm_GpMoveHelixAbs (HAND GroupHandle, PF64 CenterArray, PF64 EndArray, PU32 pArrayElements, I16 Direction)

目的：

命令群组进行绝对螺旋运动。

参数：

名称	类型	IN 或 OUT	说明
GpHandle	HAND	IN	来自 Acm_GpAddAxis 的群组句柄。
CenterArray	PF64	IN	圆心点的绝对距离。
EndArray	PF64	IN	终止点的绝对距离。
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）。
Direction	I16	IN	方向： 0：DIR_CW （顺时针）； 1：DIR_CCW （逆时针）。

返回值：

错误代码

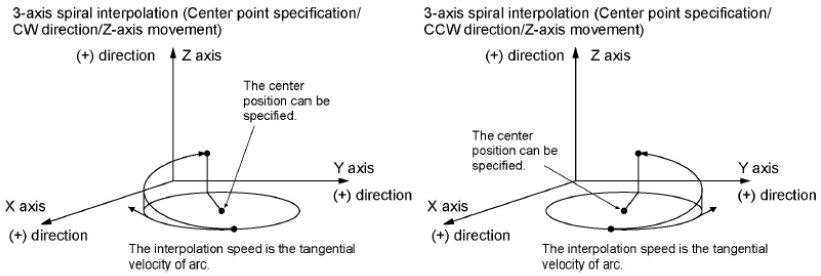
注解：

该命令只支持 3 个轴。**CenterArray** 和 **EndArray** 中的元素必须按照轴的 PhysicalID 的顺序。用户可通过属性 PAR_GpRefPlane 选择群组中的两个轴进行 ARC 插补运动，其它轴决定螺旋线的高度。**CenterArray** 和 **EndArray** 中距离的单位为群组中每个轴的 PPU。

比如：

Group （Y、Z、U），CenterArray （Y、Z、U），EndArray （Y、Z、U）。如果 PAR_GpRefPlane =1(YZ_Plane)，Z 轴和 U 轴将进行 ARC 插补运动，EndArray 中的 Y 值为螺旋线的高度。

螺旋线如下图所示：



PCI-1285E 不支持该 API。

6.3.5.4.11 Acm_GpMoveHelixRel

格式：

U32 Acm_GpMoveHelixRel (HAND GroupHandle, PF64 CenterArray, PF64 EndArray, PU32 pArrayElements, I16 Direction)

目的：

命令群组进行相对螺旋运动。

参数：

名称	类型	IN 或 OUT	说明
GpHandle	HAND	IN	来自 Acm_GpAddAxis 的群组句柄。
CenterArray	PF64	IN	圆心点的相对距离。
EndArray	PF64	IN	终止点的相对距离。
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）。
Direction	I16	IN	方向： 0：DIR_CW （顺时针）； 1：DIR_CCW （逆时针）。

返回值：

错误代码

注解：

请参考 Acm_GpMoveHelixAbs。

PCI-1285E 不支持该 API。

6.3.5.4.12 Acm_GpMoveHelixAbs_3P

格式：

U32 Acm_GpMoveHelixAbs_3P (HAND GroupHandle, PF64 RefArray, PF64 EndArray, PU32 pArrayElements, I16 Direction)

目的：

命令群组通过三个指定点执行绝对螺旋插补。

参数：

名称	类型	IN 或 OUT	说明
GpHandle	HAND	IN	来自 Acm_GpAddAxis 的群组句柄。
RefArray	PF64	IN	参考点的绝对距离。
EndArray	PF64	IN	终止点的绝对距离。
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）。
Direction	I16	IN	方向： 0：DIR_CW （顺时针）； 1：DIR_CCW （逆时针）。

返回值：

错误代码

注解：

必须为群组中的 3 个轴。

通过指定三个点命令螺旋运动。参数 **RefArray**、**CenterArray** 和 **EndArray** 值的顺序必须按照轴的 PhysicalID 的顺序。**RefArray** 和 **EndArray** 中距离的单位为群组中每个轴的 PPU。

用户可通过 PAR_GpRefPlane 选择群组中的两个轴进行 ARC 插补运动。

比如：

Group （Y、Z、U），**RefArray （Y、Z、U）**，**CenterArray （Y、Z、U）**，**EndArray （Y、Z、U）**，PAR_GpRefPlane =1(YZ_Plane)。

Z 轴和 U 轴将进行 ARC 插补运动，**EndArray** 中的 Y 值为螺旋线的高度。

PCI-1285E 不支持该 API。

6.3.5.4.13 Acm_GpMoveHelixRel_3P

格式：

```
U32 Acm_GpMoveHelixRel_3P (HAND GroupHandle, PF64 RefArray,
PF64 EndArray, PU32 pArrayElements, I16 Direction)
```

目的：

命令群组通过三个指定点执行相对螺旋插补。

参数：

名称	类型	IN 或 OUT	说明
GpHandle	HAND	IN	来自 Acm_GpAddAxis 的群组句柄。
RefArray	PF64	IN	参考点的相对距离。
EndArray	PF64	IN	终止点的相对距离。
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）。
Direction	I16	IN	方向： 0：DIR_CW （顺时针） 1：DIR_CCW （逆时针）

返回值：

错误代码

注解：

请参考 Acm_GpMoveHelixAbs_3P。

PCI-1285E 不支持该 API。

6.3.5.4.14 Acm_GpMoveCircularRel_Angle

格式：

U32 Acm_GpMoveCircularRel_Angle (HAND GroupHandle,
PF64 CenterArray, U16 Degree, PU32 ArrayElements,I16 Direction)

目的：

通过相对圆心坐标，旋转角度以及方向进行圆弧插补。

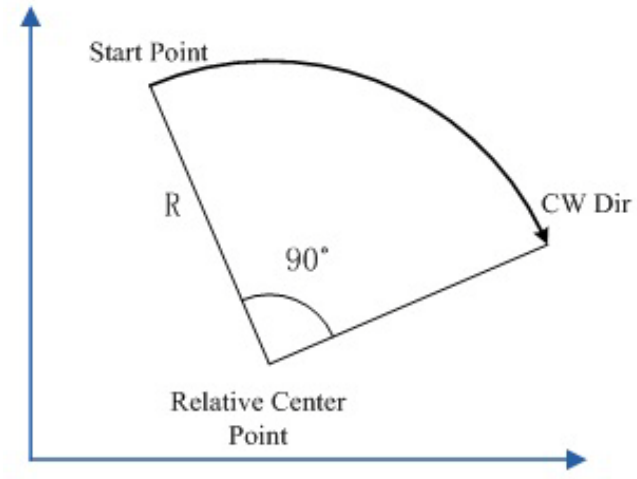
参数：

名称	类型	IN 或 OUT	说明
GroupHandle	HAND	IN	群组句柄，来自 Acm_GpAddAxis
CenterArray	PF64	IN	圆心相对于起点的相对距离
Degree	U16	IN	旋转角度，范围：0~360
pArrayElements	PU32	IN	轴数
Direction	I16	IN	方向： 0: CW-Dir 1: CCW_Dir

返回值：

错误代码

注解：



PCI-1285E 不支持该 API。

6.3.5.4.15 Acm_GpMoveCircularAbs_Angle

格式：

U32 Acm_GpMoveCircularAbs_Angle (HAND GroupHandle,
PF64 CenterArray, U16 Degree, PU32 ArrayElements,I16 Direction)

目的：

通过相对圆心坐标，旋转角度以及方向进行圆弧插补。

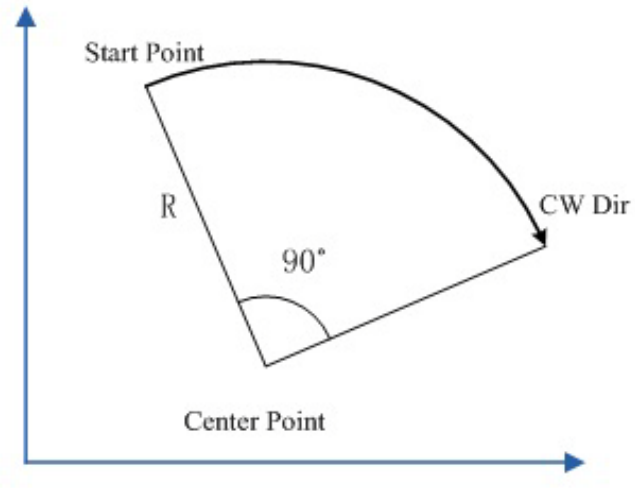
参数：

名称	类型	IN 或 OUT	说明
GroupHandle	HAND	IN	群组句柄，来自 Acm_GpAddAxis
CenterArray	PF64	IN	圆心坐标
Degree	U16	IN	旋转角度，范围：0~360
ArrayElements	PU32	IN	轴数
Direction	I16	IN	方向： 0: CW-Dir 1: CCW_Dir

返回值：

错误代码

注解：



PCI-1285E 不支持该 API。

6.3.5.4.16 Acm_GpChangeVelByRate

格式：

U32 Acm_GpChangeVelByRate(HAND GroupHandle, U32 Rate)

目的：

按照设定的比例改变当前正在执行的群组动作的运行速度。

参数：

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

返回值：

错误代码

注解：

新速度 = 群组之前的运行速度 * Rate * 0.01。Rate 须大于 0 且小于群组中 ID 最小的轴的最大速度与当前 Group 的速度的比值。新速度仅对此次运动有效。关于在插补或连续插补过程中改变速度的情形，详细可参考 Acm_GpChangeVel.

PCI-1285E 不支持该 API。

6.3.5.5 路径

6.3.5.5.1 Acm_GpAddPath

格式：

U32 Acm_GpAddPath (HAND GroupHandle, U16 MoveCmd, U16 MoveMode, F64 FH, F64 FL, PF64 EndPoint_DataArray, PF64 CenPoint_DataArray, PU32 ArrayElements)

目的：

将一个插补路径添加至系统路径缓存。

参数：

名称	类型	IN 或 OUT	说明
GpHandle	HAND	IN	来自 Acm_GpAddAxis 的群组句柄。
MoveCmd	U16	IN	运动命令： 0：EndPath 1：Abs2DLine； 2：Rel2DLine； 3：Abs2DArcCW； 4：Abs2DArcCCW； 5：Rel2DArcCW； 6：Rel2DArcCCW； 7：Abs3DLine； 8：Rel3DLine； 9：Abs4DLine； （不支持） 10：Rel4DLine； （不支持） 11：Abs2DDirect； 12：Rel2DDirect； 13：Abs3DDirect； 14：Rel3DDirect； 15：Abs4DDirect； 16：Rel4DDirect； 17：Abs5DDirect； 18：Rel5DDirect； 19：Abs6DDirect； 20：Rel6DDirect； 21：Abs3DArcCW； （不支持） 22：Rel3DArcCW； （不支持） 23：Abs3DArcCCW； （不支持） 24：Rel3DArcCCW； （不支持） 25：Abs3DHelixCW 26：Rel3DHelixCW 27：Abs3DHelixCCW 28：Rel3DHelixCCW 29：GPDELAY （单位：ms）
MoveMode	U16	IN	运动模式： 0：使能速度交接模式； 1：禁用速度交接模式。
FH	F64	IN	运行速度 （单位：群组的 PPU/s）/GPDELAY 命令中的延迟时间 （单位：ms）。
FL	F64	IN	低速度 （起始速度）（单位：群组的 PPU/s）。
EndPoint_DataArray	PF64	IN	终点坐标 （单位：每个轴的 PPU）。
CenPoint_DataArray	PF64	IN	中心点坐标 （单位：每个轴的 PPU）。
ArrayElements	PU32	IN/OUT	阵列元素的个数必须小于群组中轴的个数，否则将返回群组中的轴个数。

返回值：

错误代码

注解：

系统缓存中每个路径的群组句柄必须相同。因此，如果系统缓存中存在一些未执行路径，且用户想要通过调用 **Acm_GpAddPath** 添加新的路径，参数 GroupHandle 必须和之前未执行的路径群组句柄相同。系统路径缓存的当前状态可通过调用 Acm_GpGetPathStatus 获取。 调用 Acm_GpMovePath 后，缓存中的路径数据能够有序地加载至寄存器。

绝对命令和相对命令不能再系统路径缓存中混合，除 **EndPath** 和 **GPDELAY** 外，否则将返回错误。

ArrayElements 参数是 **EndPoint_DataArray** 参数以及 **CenPoint_DataArray** 参数的元素个数，且不能小于群组中的轴数。

按照运动命令，所需的轴个数不大于群组的轴个数时，所有路径可加载至相同的系统缓存。如：群组中的轴个数为 4，那么 **Rel2DLine**、**Rel3DLine** 和 **4DDirect** 等的路径可加载至设备。如果群组中的轴个数大于路径中所需的轴个数，将选择群组中前面的轴实现运动。尤其对于 **Abs2DArcCW**、**Abs2DArcCCWRel2DArcCW** 和 **Rel2DArcCCW**，用户可通过 Par_GpRefPlane 选择群组中前面三个轴中的两个轴实现运动。最后，必须将 EndPath 命令添加至路径缓存中。

群组中最多有 10000 个路径。

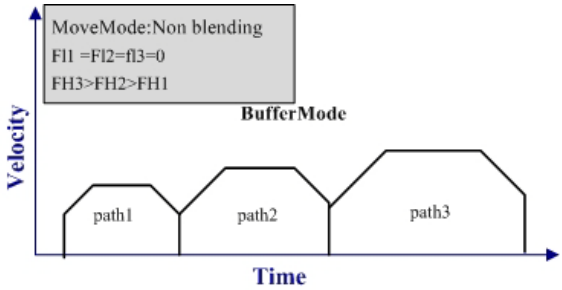
调用 Acm_GpAddPath 之前，用户可通过 CFG_GpSFEnable 启用 / 禁用速度前瞻功能，通过 CFG_GpBldTime 设置速度交接时间。

GPDELAY：延迟命令。群组将延迟一段时间再执行下一路径。用户可通过 FH 设置时间。当通过 CFG_GpSFEnable 启用速度前瞻功能时，该命令的路径不能添加。延迟事件的单位为 ms。

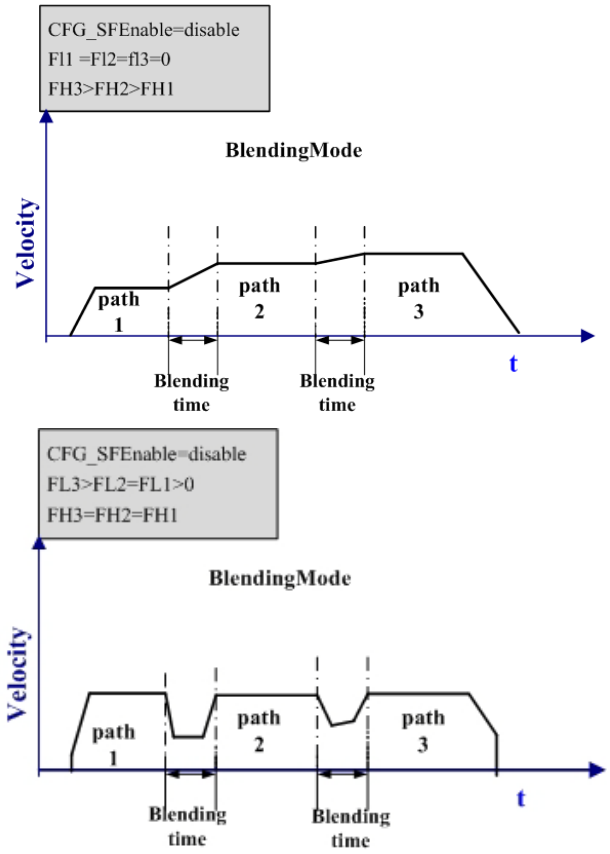
MoveMode：速度交接模式和非交接模式。在速度交模式中，应禁用 CFG_GpSFEnable。

根据 **MoveMode** 和 CFG_GpBldTime 的设置，有三种模式，分别为：BufferMode、BlendingMode 和 FlyMode。

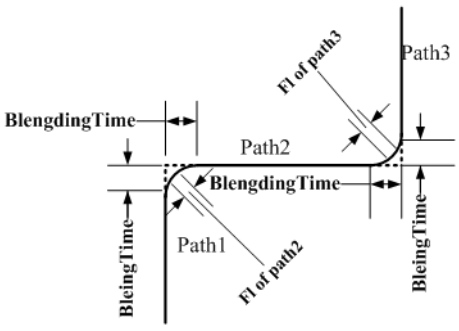
1. **BufferMode**：当 **MoveMode** 为非交接模式。在这种模式中，每个路径都包括加速和减速的整个过程。这种不支持速度前瞻功能，因此应禁用 CFG_GpSFEnable。



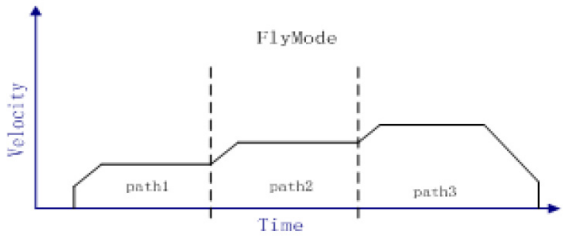
2. **BlendingMode**：当 **MoveMode** 为交接模式，且 CFG_GpBldTime 的值大于 0。这种不支持 S 型速度曲线。当通过 CFG_SFEnable 启用速度前瞻功能时，由于路径运动中使用的所有速度参数为群组速度设置，因此无需使用参数 FL 和 FH，所有路径支持相同的驱动速度。比如，CFG_SFEnable = Disable，BlendingTime>0。速度曲线如下所示：



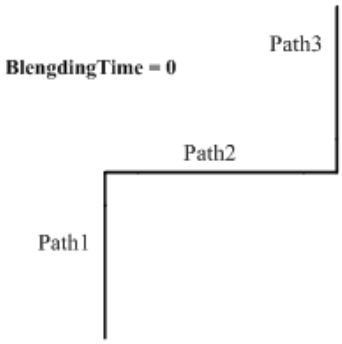
路径如下所示：



3. FlyMode：当 MoveMode 为交接模式，且 CFG_GpBldTime 的值为 0。当通过 CFG_SFEnable 启用速度前瞻功能时，由于路径运动中使用的所有速度参数为群组速度设置，因此无需使用参数 FL 和 FH，所有路径支持相同的驱动速度。速度曲线如下所示：



运动路径如下所示：



6.3.5.5.2 Acm_GpResetPath

格式：

U32 Acm_GpResetPath (PHAND GroupHandle)

目的：

清空系统路径缓存。 如果有群组正在执行路径，则路径运动将停止。

参数：

名称	类型	IN 或 OUT	说明
GpHandle	PHAND	IN	指针指向来自 Acm_GpAddAxis 的群组句柄。

返回值：

错误代码

注解：

6.3.5.5.3 Acm_GpLoadPath

格式：

U32 Acm_GpLoadPath(HAND GroupHandle, PI8 FilePath, PHAND PathHandle, PU32 pTotalCount)

目的：

加载来自路径文件的路径数据。一次可加载最多 600 个路径数据。

参数：

名称	类型	IN 或 OUT	说明
GpHandle	HAND	IN	来自 Acm_GpAddAxis 的群组句柄。
FilePath	PI8	IN	指向需要加载的运动路径文件。
PathHandle	PHAND	OUT	返回指针指向路径句柄。
pTotalCount	PU32	OUT	返回路径文件中路径数据的实际总个数。

返回值：

错误代码

注解：

路径数据文件 （二进制）通常由运动实用程序的 [Path Editor] 生成。如果用户熟悉研华运动产品，那么可以自行创建文件。当不再使用 PathHandle 或应用关闭时，必须通过 Acm_GpUnloadPath 卸载 PathHandle，同时 PathHandle 中包含的路径将从驱动中删除。

6.3.5.5.4 Acm_GpUnloadPath

格式：

U32 Acm_GpUnloadPath (HAND GroupHandle, PHAND PathHandle)

目的：

卸载路径数据。

参数：

名称	类型	IN 或 OUT	说明
GpHandle	HAND	IN	来自 Acm_GpAddAxis 的群组句柄。
PathHandle	PHAND	IN	指针指向来自 Acm_GpLoadPath 的路径句柄。

返回值：

错误代码

注解：

6.3.5.5.5 Acm_GpMovePath

格式：

U32 Acm_GpMovePath (HAND GroupHandle, HAND PathHandle)

目的：

开始连续插补运动 （路径）。

参数：

名称	类型	IN 或 OUT	说明
GpHandle	HAND	IN	来自 Acm_GpAddAxis 的群组句柄。
PathHandle	PHAND	IN	指针指向来自 Acm_GpLoadPath 的路径句柄。

返回值：

错误代码

注解：

如果 Acm_GpLoadPath 返回 **PathHandle**，那么路径数据将首先加载到系统路径缓存，然后执行路径缓存中的路径。 如果 **PathHandle** 为 NULL，将直接执行系统路径缓存中的路径数据。

6.3.5.5.6 Acm_GpGetPathStatus

格式：

U32 Acm_GpGetPathStatus (HAND GroupHandle, PU32 pCurIndex, PU32 pCurCmdFunc, PU32 pRemainCount, PU32 pFreeSpaceCount)

目的：

获取路径缓存的当前状态。

参数：

名称	类型	IN 或 OUT	说明
GpHandle	HAND	IN	来自 Acm_GpAddAxis 的群组句柄。
pCurIndex	PU32	OUT	返回路径缓存中当前正在执行的路径对应的的当前索引。

pCurCmdFunc	PU32	OUT	返回正在执行的当前命令函数。 0：EndPath； 1：Abs2DLine； 2：Rel2DLine； 3：Abs2DArcCW； 4：Abs2DArcCCW； 5：Rel2DArcCW； 6：Rel2DArcCCW； 7：Abs3DLine； 8：Rel3DLine； 9：Abs4DLine； （不支持） 10：Rel4DLine； （不支持） 11：Abs2DDirect； 12：Rel2DDirect； 13：Abs3DDirect； 14：Rel3DDirect； 15：Abs4DDirect； 16：Rel4DDirect； 17：Abs5DDirect； 18：Rel5DDirect； 19：Abs6DDirect； 20：Rel6DDirect； 21：Abs3DArcCW； （不支持） 22：Rel3DArcCW； （不支持） 23：Abs3DArcCCW； （不支持） 24：Rel3DArcCCW； （不支持） 25：Abs3DHelixCW 26：Rel3DHelixCW 27：Abs3DHelixCCW 28：Rel3DHelixCCW 29：GPDELAY （单位：ms）
pRemainCount	PU32	OUT	返回路径中未执行的路径数据的个数。
pFreeSpaceCount	PU32	OUT	返回路径缓存中剩余空间的个数。

返回值：

错误代码

注解：

用户必须输入 GroupHandle 并获取该群组的路径状态。

6.3.5.5.7 Acm_GpMoveSelPath

格式：

```
U32 Acm_GpMoveSelPath (HANDGroupHandle, HAND PathHanle,
U32 StartIndex, U32EndIndex, U8Repeat)
```

目的：

执行路径缓存中指定的起始索引到终止索引范围内的路径。

参数：

名称	类型	IN 或 OUT	说明
GpHandle	HAND	IN	来自 Acm_GpAddAxis 的群组句柄。
PathHandle	HAND	IN	指针指向来自 Acm_GpLoadPath 的路径句柄。
StartIndex	U32	IN	起始索引。（0 ~ 9999）
EndIndex	U32	IN	结尾索引。（0 ~ 9999）
Repeat	U8	IN	重复个数。（0 ~ 255）

返回值：

错误代码

注解：

命令运动路径索引在 StartIndex 和 EndIndex 之间的路径。

如果 **PathHandle** 是 Null，将运动系统路径缓存中的指定路径；如果 PathHandle 不是 Null，PathHandle 中的路径将首先加载至系统路径缓存，然后运动到指定路径。不支持绝对模式。

如果 **Repeat** 的值为 0，那么将连续且重复执行指定路径，直到停止群组运动。

如果 **EndIndex** 的值大于系统路径缓冲中的路径个数，将把 StartIndex 路径和最后一个索引路径之间的路径运动至系统路径缓冲。

StartIndex 和 EndIndex 是针对当前 path buffer 中所保留的 path。例如当前 path buffer 中剩余有 10 段 Path（可通过调用 Acm_GpGetPathStatus 查看剩余的 path 数量），则 StartIndex 和 EndIdnex 则可取范围为 0~9。

6.3.5.5.8 Acm_GpGetPathIndexStatus

格式：

```
Acm_GpGetPathIndexStatus (HAND GroupHandle, U32 Index,
PU16 CmdFunc, PU16 MoveMode, PF64 FH, PF64 FL,
F64 EndPoint_DataArray, PF64 CenPoint_DataArray,
PU32 ArrayElements)
```

目的：

获取系统路径缓存中指定索引路径的状态。

参数：

名称	类型	IN 或 OUT	说明
GpHandle	HAND	IN	来自 Acm_GpAddAxis 的群组句柄。
Index	U32	IN	路径的索引。

CmdFunc	PU16	OUT	返回正在执行的当前命令函数。 0：EndPath； 1：Abs2DLine； 2：Rel2DLine； 3：Abs2DArcCW； 4：Abs2DArcCCW； 5：Rel2DArcCW； 6：Rel2DArcCCW； 7：Abs3DLine； 8：Rel3DLine； 9：Abs4DLine； （不支持） 10：Rel4DLine； （不支持） 11：Abs2DDirect； 12：Rel2DDirect； 13：Abs3DDirect； 14：Rel3DDirect； 15：Abs4DDirect； 16：Rel4DDirect； 17：Abs5DDirect； 18：Rel5DDirect； 19：Abs6DDirect； 20：Rel6DDirect； 21：Abs3DArcCW； （不支持） 22：Rel3DArcCW； （不支持） 23：Abs3DArcCCW； （不支持） 24：Rel3DArcCCW； （不支持） 25：Abs3DHelixCW 26：Rel3DHelixCW 27：Abs3DHelixCCW 28：Rel3DHelixCCW 29：GPDELAY （单位：ms）
MoveMode	PU16	OUT	运动模式： 0：非混合； 1：混合。
FH	PF64	OUT	单位：主轴的 PPU （具有最小 PhysicalID 的轴）。
FL	PF64	OUT	单位：主轴的 PPU （具有最小 PhysicalID 的轴）。
EndPoint_DataArray	PF64	OUT	单位：各轴的 PPU （具有最小 PhysicalID 的轴）。
CenPoint_DataArray	PF64	OUT	单位：各轴的 PPU （具有最小 PhysicalID 的轴）。
ArrayElements	PU32	IN/OUT	返回轴个数。

返回值：

错误代码

注解：

如果用户想要了解路径设置，可调用该 API。

StartIndex 和 EndIndex 是针对当前 path buffer 中所保留的 path。例如当前 path buffer 中剩余有 10 段 Path（可通过调用 Acm_GpGetPathStatus 查看剩余的 path 数量），则 StartIndex 和 EndIdnex 则可取范围为 0~9。

6.3.5.6 暂停和恢复

6.3.5.6.1 Acm_GpPauseMotion

格式：

U32 Acm_GpPauseMotion(HAND GroupHandle)

目的：

暂停群组运动命令。

参数：

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

返回值：

错误代码

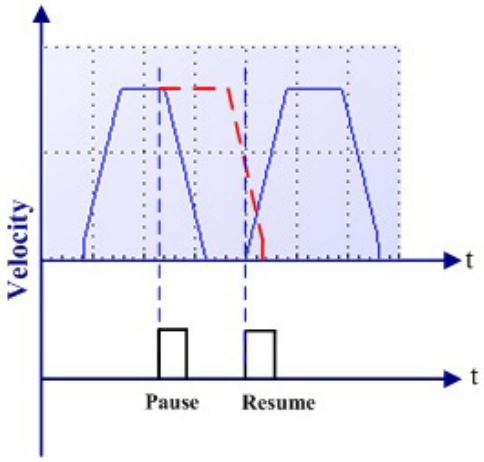
注解：

当 Group 在运动的过程中，可以对其下达暂停命令，则板卡在收到命令后，会减速停止下来。当对其再做恢复命令后，则继续执行暂停之前的尚未完成的部分。

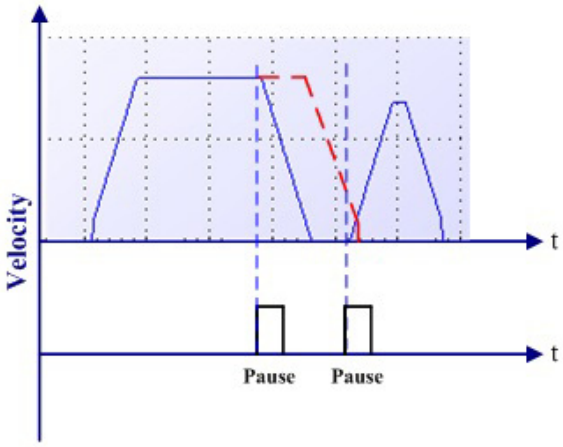
暂停和恢复功能支持 Group 的几乎所有的动作，包括直线插补、圆弧插补、螺旋插补以及连续插补功能。

针对暂停和恢复功能其对速度曲线的影响有如下几种情况 （以 T 型速度曲线为例）：

1. 单一插补动作的过程中，如果下达暂停命令，其速度曲线为：

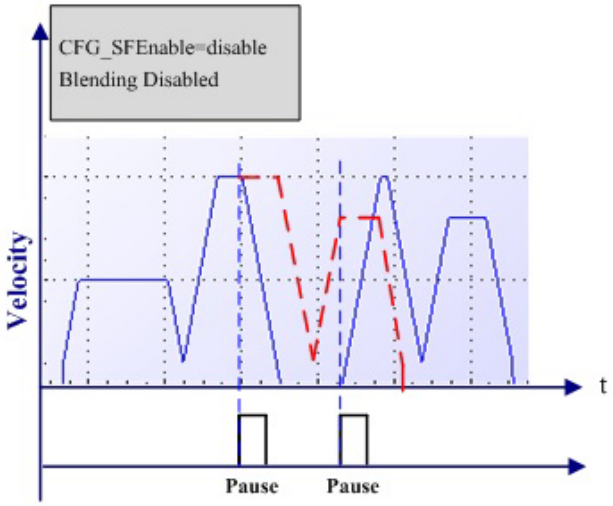


当恢复暂停的动作时硬体会自行计算剩余的Pulse会是否足以支持加速到之前的速度，若不能达到之前的速度， 其速度曲线如下图所示：



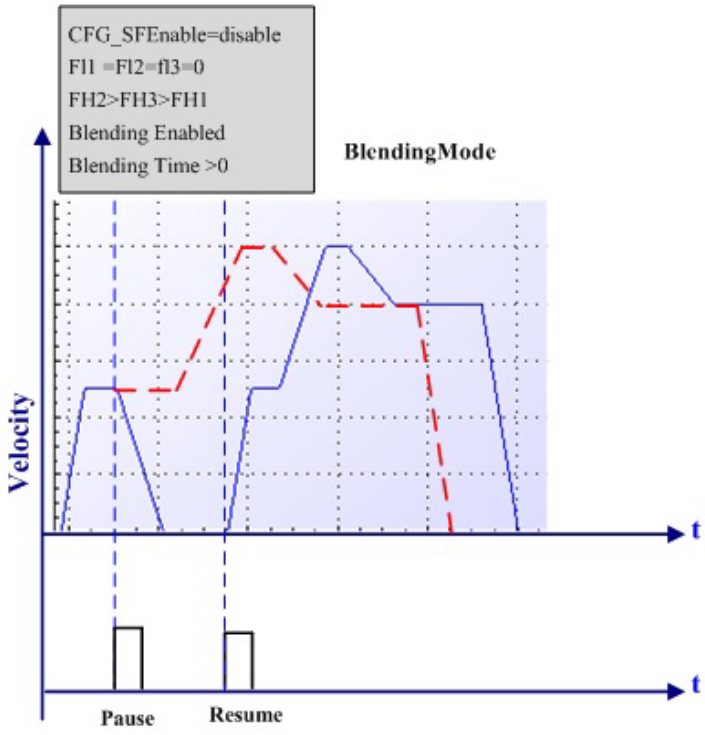
2. 连续插补动作过程中，在不同的模式下，其情况也有所不同。

■Buffer Mode：Non-Blending



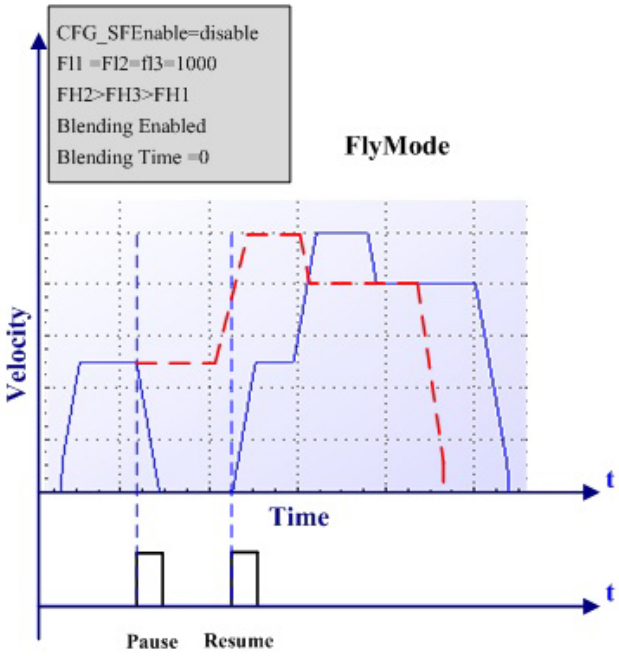
在 BufferMode 中，每段都有自己的加减速过程，当收到暂停命令后，当前正在执行的某段会减速停止，当恢复时，板卡则会计算停止的那段剩余的 Pulse 所能够支持的速度，然后继续执行剩余的路段。

■BlendingMode：Blending Enable，BlendingTime>0，在 Blending Mode 下，分为两种情况，当 Acm_GpAddPath 中的 FL=0 和 FL>0 时，其速度轨迹是不同的，详细请参阅 Acm_GpAddPath 介绍。下面以禁用速度前瞻功能且 FL =0 的情况下为例说明暂停和恢复功能。



在 Blending 模式下，如果暂停命令下达时，当前正在执行 Blending 段，则暂停命令延迟到 Blending 结束后执行。

■FlyMode：Blending Enable，BlendingTime =0



6.3.5.6.2 Acm_GpResumeMotion

格式：

U32 Acm_GpResumeMotion(HAND GroupHandle)

目的：

恢复暂停的群组运动命令

参数：

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

返回值：

错误代码

注解：

详细信息请参阅 Acm_GpPauseMotion。

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
2	DO
3	AI （PCI-1285/1285E 不支持）
4	AO （PCI-1285/1285E 不支持）
5	定时器
6	计数器
7	DAQ DI （PCI-1285/1285E 不支持）
8	DAQ DO （PCI-1285/1285E 不支持）
9	DAQ AI （PCI-1285/1285E 不支持）
10	DAQ AO （PCI-1285/1285E 不支持）
11	Emg
12~31	未定义

注解：

6.4.1.1.4 FT_DevMDAQTypeMap

数据类型：

U32

R/W：

R

属性 ID：

6

含义：

MotionDAQ 所支持的资料类型。

位	说明
0	理论位置。
1	实际位
2	理论位置与实际位置的偏差
3	理论速度值

注解：

6.4.1.1.5 FT_DevMDAQTrigMap

数据类型：

U32

R/W：

R

属性 ID：

7

含义：

触发 MotionDAQ 功能的几种方式：

位	说明
0	禁用 MotionDAQ 功能
1	软体命令触发方式 （即下达 Start 命令触发）
2	DI 触发
3	指定轴运动开始即触发 MotionDAQ 功能

注解：

6.4.1.1.6 FT_DevMDAQMaxChan

数据类型：

U32

R/W：

R

属性 ID：

8

含义：

记录 MotionDAQ 资料的最大通道数

注解：

在 PCI-1285/1285E 中所支持的最大通道数为 8。

6.4.1.1.7 FT_DevMDAQMaxBufCount

数据类型：

U32

R/W：

R

属性 ID：

9

含义：

每个 MotionDAQ 通道最多可记录的 MotionDAQ 资料笔数。

注解：

在 PCI-1285/1285E 中每个 MotionDAQ 通道最多可记录 2000 笔资料。

6.4.1.1.8 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-1285/1285E 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-1285/1285E，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：

214

含义：

获取固件版本。格式为：1.0.0.1.

注解：

6.4.1.2.9 CFG_DevCPLDVersion

数据类型：

char*

R/W：

R

属性 ID：

219

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 DAQ

6.4.2.1 特性

6.4.2.1.1 FT_DaqDiMaxChan

数据类型：

U32

R/W：

R

属性 ID：

50

含义：

获取 DI 通道的最大个数。

注解：

6.4.2.1.2 FT_DaqDoMaxChan

数据类型：

U32

R/W：

R

属性 ID：

51

含义：

获取 DO 通道的最大个数。

注解：

6.4.2.1.3 FT_DaqAiRangeMap

数据类型：

U32

R/W：

R

属性 ID：

52

含义：

获取支持的 AI 范围。1：支持； 0：不支持。

位	说明
0	+/- 10V
1	+/- 5V
2	+/- 2.5V
3	+/- 1.25V
4	+/- 0.625
5 ~ 15	未定义
16	0 ~ 20 mA
17 ~ 31	未定义

注解：

PCI-1285/1285E 不支持该 API。

6.4.2.1.4 FT_DaqAiMaxSingleChan

数据类型：

U32

R/W：

R

属性 ID：

54

含义：

获取设备的最大单端 AI 通道个数。

注解：

PCI-1285/1285E 不支持该 API。

6.4.2.1.5 FT_DaqAiMaxDiffChan

数据类型：

U32

R/W：

R

属性 ID：

55

含义：

获取设备的最大差分 AI 通道个数。

注解：

PCI-1285/1285E 不支持该 API。

6.4.2.1.6 FT_DaqAiResolution

数据类型：

U32

R/W：

R

属性 ID：

56

含义：

获取设备的 AI 分辨率，单位为 bit。

注解：

PCI-1285/1285E 不支持该 API。

6.4.2.2 配置

6.4.2.2.1 CFG_DaqAiChanType

数据类型：

U32

R/W：

R/W

属性 ID：

251

含义：

0：单端； 1：差分。

注解：

PCI-1285/1285E 不支持该 API。

6.4.2.2.2 CFG_DaqAiRanges

数据类型：

U32

R/W：

R/W

属性 ID：

252

含义：

设定通道的采值范围。具体如下所示：

值	描述
0x0	+/- 10 V
0x1	+/- 5 V
0x2	+/- 2.5 V
0x3	+/- 1.25 V

注解：

PCI-1285/1285E 不支持该 API。

6.4.3 轴

6.4.3.1 特性

6.4.3.1.1 系统

6.4.3.1.1.1 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：

6.4.3.1.2 速度模式

6.4.3.1.2.1 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-1285/1285E，该值为 5,000,000。

6.4.3.1.2.2 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-1285/1285E，该值为 500,000,000。

6.4.3.1.2.3 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-1285/1285E，该值为 500,000,000。

6.4.3.1.2.4 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-1285/1285E，该值为 1。

6.4.3.1.3 脉冲输入

6.4.3.1.3.1 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：

6.4.3.1.3.2 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：

6.4.3.1.4 脉冲输出

6.4.3.1.4.1 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-1285/1285E，该值为 1。

6.4.3.1.4.2 FT_AxPulseOutModeMap

数据类型：

U32

R/W：

R

属性 ID：

309

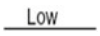
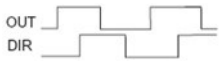
含义：

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-1285/1285E 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				

6.4.3.1.5 报警

6.4.3.1.5.1 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：

6.4.3.1.6 到位

6.4.3.1.6.1 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：

6.4.3.1.7 ERC

6.4.3.1.7.1 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：

6.4.3.1.7.2 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：

6.4.3.1.8 SD

6.4.3.1.8.1 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-1285/1285E，该值为 0。

6.4.3.1.9 硬件限位

6.4.3.1.9.1 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：

6.4.3.1.10 软件限位

6.4.3.1.10.1 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：

6.4.3.1.10.2 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：

6.4.3.1.11 原点

6.4.3.1.11.1 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：

6.4.3.1.11.2 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。

6.4.3.1.12 背隙补偿

6.4.3.1.12.1 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：

6.4.3.1.13 比较

6.4.3.1.13.1 FT_AxCompareMap

数据类型：

U32

R/W：

R

属性 ID：

324

含义：

获取轴支持的比较特性。

位	说明
0	启用
1	逻辑
2	CmpSrc
3	CmpMethod
4	CmpPulseMode
5	CmpPulseWidth
6 ~ 31	未定义

注解：

6.4.3.1.14 锁存

6.4.3.1.14.1 FT_AxLatchMap

数据类型：

U32

R/W：

R

属性 ID：

325

含义：

获取轴支持的锁存特性。

位	说明
0	启用
1	逻辑
2 ~ 31	未定义

注解：

6.4.3.1.15 Cam DO

6.4.3.1.15.1 FT_AxCamDOMap

数据类型：

U32

R/W：

R

属性 ID：

326

含义：

获取轴支持的 CamDO 特性。

位	说明
0	启用
1	CamDOLogic
2	CamDOCmpSrc
3	CamDOLoLimit
4	CamDOHiLimit
5	CamDOMode
6	CamDODir
7 ~ 31	未定义

注解：

6.4.3.1.16 外部驱动

6.4.3.1.16.1 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。

6.4.3.1.16.2 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：

6.4.3.1.17 Aux/Gen 输出

6.4.3.1.17.1 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：

6.4.3.1.17.2 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 ~ IN4。

位	说明
0	IN1/LTC
1	IN2/RDY
2	IN3/JOG+
3	IN4/JOG-
4 ~ 31	未定义

注解：

6.4.3.1.18 同步起停

6.4.3.1.18.1 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。
12	从轴 _4 的比较信号启动同步运动。
13	从轴 _5 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。
20	当轴 _4 停止时启动同步运动。
21	当轴 _5 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-1285 中，默认值为：16776961。在 PCI-1285E 中，默认值为：0。

6.4.3.1.19 触发停止

6.4.3.1.19.1 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：

6.4.3.1.19.2 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：

6.4.3.1.19.3 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：

6.4.3.1.19.4 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：

6.4.3.2 配置

6.4.3.2.1 系统

6.4.3.2.1.1 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。

6.4.3.2.1.2 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：

6.4.3.2.2 速度模式

6.4.3.2.2.1 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-1285/1285E，该默认值为 4,000,000。

6.4.3.2.2.2 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-1285/1285E，该默认值为 50,000,000。

6.4.3.2.2.3 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-1285/1285E，该默认值为 50,000,000。

6.4.3.2.2.4 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-1285/1285E，该值为 1。

6.4.3.2.3 Pulse In

6.4.3.2.3.1 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：

6.4.3.2.3.2 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：

6.4.3.2.3.3 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：

6.4.3.2.4 脉冲输出

6.4.3.2.4.1 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
64	A/B 相位 （PCI-1245 和 PCI-1265 不支持）
128	B/A 相位 （PCI-1245 和 PCI-1265 不支持）
256	CW/CCW，OUT 负逻辑
512	CW/CCW，DIR 负逻辑

注解：

对于 PCI-1285/1285E，该默认值为 16。

请参考 FT_AxPulseOutMode。

6.4.3.2.5 报警

6.4.3.2.5.1 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-1285/1285E，该默认值为 1。

6.4.3.2.5.2 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-1285/1285E，该默认值为 0。

6.4.3.2.5.3 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-1285/1285E，该默认值为 1。

6.4.3.2.6 到位

6.4.3.2.6.1 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-1285/1285E，该默认值为 0。

6.4.3.2.6.2 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-1285/1285E，该默认值为 1。

6.4.3.2.7 ERC

6.4.3.2.7.1 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-1285/1285E，该默认值为 1。

6.4.3.2.7.2 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-1285/1285E，该默认值为 1。

6.4.3.2.8 硬件限位

6.4.3.2.8.1 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-1285/1285E，该默认值为 1。

6.4.3.2.8.2 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-1285/1285E，该默认值为 1。

6.4.3.2.8.3 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-1285/1285E，该默认值为 1。

6.4.3.2.9 软件限位

6.4.3.2.9.1 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：

6.4.3.2.9.2 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：

6.4.3.2.9.3 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-1285/1285E，该默认值为 1。

6.4.3.2.9.4 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-1285/1285E，该默认值为 1。

6.4.3.2.9.5 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：

6.4.3.2.9.6 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：

6.4.3.2.10 原点

6.4.3.2.10.1 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-1285/1285E，该默认值为 1。

6.4.3.2.10.2 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-1285/1285E，该默认值为 1。

6.4.3.2.10.3 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：

6.4.3.2.10.4 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：

6.4.3.2.11 背隙补偿

6.4.3.2.11.1 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-1285/1285E，该默认值为 0。

6.4.3.2.11.2 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-1285/1285E，该默认值为 0。

6.4.3.2.11.3 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-1285/1285E，该默认值为 0。

6.4.3.2.12 比较

6.4.3.2.12.1 CFG_AxCmpSrc

数据类型：

U32

R/W：

RW

属性 ID：

603

含义：

设置 / 获取比较源。

值	说明
0	理论位置
1	实际位置

注解：

PCI-1285E 不支持该属性。对于 PCI-1285，该默认值为 0 。

6.4.3.2.12.2 CFG_AxCmpMethod

数据类型：

U32

R/W：

RW

属性 ID：

604

含义：

设置 / 获取比较方法。

值	说明
0	>= 位置计数器
1	<= 位置计数器
2	= 计数器 （无方向）（不支持）

注解：

PCI-1285E 不支持该属性。对于 PCI-1285，该默认值为 0 。

6.4.3.2.12.3 CFG_AxCmpPulseLogic

数据类型：

U32

R/W：

RW

属性 ID：

606

含义：

设置 / 获取比较信号的逻辑准位。

值	说明
0	低准位
1	高准位

注解：

PCI-1285E 不支持该属性。对于 PCI-1285，该默认值为 0 。

6.4.3.2.12.4 CFG_AxCmpPulseWidth

数据类型：

U32

R/W：

RW

属性 ID：

607

含义：

获取 / 设置比较信号延迟的宽度。

值	说明
0	5 Microsecond(us)
1	10 Microsecond(us)
2	20 Microsecond(us)
3	50 Microsecond(us)
4	100 Microsecond(us)
5	200 Microsecond(us)
6	500 Microsecond(us)
7	1000 Microsecond(us)

注解：

PCI-1285E 不支持该属性。对于 PCI-1285，该默认值为 0。

6.4.3.2.12.5 CFG_AxCmpEnable

数据类型：

U32

R/W：

RW

属性 ID：

608

含义：

启用 / 禁用轴比较功能。

值	说明

0	禁用
1	启用

注解：

PCI-1285E 不支持该属性。对于 PCI-1285，该默认值为 0 。

6.4.3.2.12.6 CFG_AxCmpPulseMode

数据类型：

U32

R/W：

RW

属性 ID：

605

含义：

设置 / 获取轴比较模式。

值	说明
0	脉冲模式
1	开关模式

注解：

开关模式：当比较信号发生时，切换 DO 值。

PCI-1285E 不支持该属性。对于 PCI-1285，该默认值为 0 。

6.4.3.2.13 锁存

6.4.3.2.13.1 CFG_AxLatchLogic

数据类型：

U32

R/W：

RW

属性 ID：

601

含义：

设置 / 获取锁存信号的逻辑准位。

值	说明
0	低准位
1	高准位

注解：

当锁存触发时，将锁存理论位置、实际位置和滞后距离。

PCI-1285E 不支持该属性。

6.4.3.2.13.2 CFG_AxLatchEnable

数据类型：

U32

R/W：

RW

属性 ID：

631

含义：

启用 / 禁用锁存功能。

值	说明
0	禁用
1	启用

注解：

PCI-1285E 不支持该属性。

6.4.3.2.14 Aux/Gen 输出

6.4.3.2.14.1 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**，属性 CFG_AxCmpEnable、CFG_AxCamDoEnable 和 CFG_AxErcEnableMode 会自动禁用。
函数 Acm_AxSetCmpData、Acm_AxSetCmpTable 和 Acm_AxSetCmpAuto 将不能输出信号。

6.4.3.2.15 外部驱动

6.4.3.2.15.1 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-1285/1285E，仅支持 0。

6.4.3.2.15.2 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-1285/1285E，默认值为 1。该值必须大于 0。

6.4.3.2.15.3 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

6.4.3.2.15.4 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-1285/1285E，默认值为 1。该值必须大于 0。

6.4.3.2.16 凸轮区间 DO

6.4.3.2.16.1 CFG_AxCamDOEnable

数据类型：

U32

R/W：

RW

属性 ID：

622

含义：

设置 / 获取 Cam DO 启用 / 禁用。

值	说明
0	禁用
1	启用

注解：

CamDo 和 OUT4 使用同一针脚。如果启用 CFG_AxGenDoEnable，OUT4 不能够输出 CamDo 信号。

PCI-1285E 不支持该属性。

6.4.3.2.16.2 CFG_AxCamDOLoLimit

数据类型：

U32

R/W：

RW

属性 ID：

623

含义：

设置 / 获取凸轮区间的低限位。

注解：

如果启用 CamDo，当理论 / 实际位置处于低限位值和高限位值之间时，将触发 CamDo 信号。

范围：-2147483647 ~ 2147483647。

PCI-1285E 不支持该属性。

6.4.3.2.16.3 CFG_AxCamDOHiLimit

数据类型：

U32

R/W：

RW

属性 ID：

624

含义：

设置 / 获取凸轮区间的高限位。

注解：

如果启用 CamDo，当理论 / 实际位置处于低限位值和高限位值之间时，将触发 CamDo 信号。

对于 PCI-1285，该默认值为 0。PCI-1285E 不支持该属性。

范围：-2147483647 ~ 2147483647。

6.4.3.2.16.4 CFG_AxCamDOCmpSrc

数据类型：

U32

R/W：

RW

属性 ID：

627

含义：

设置 / 获取凸轮区间的比较源。

值	说明
0	理论位置
1	反馈位置

注解：

对于 PCI-1285，该默认值为 0。PCI-1285E 不支持该属性。

6.4.3.2.16.5 CFG_AxCamDOLogic

数据类型：

U32

R/W：

RW

属性 ID：

628

含义：

设置 / 获取 CamDO 的逻辑准位。

值	说明
0	低准位
1	高准位

注解：

对于 PCI-1285，该默认值为 1。PCI-1285E 不支持该属性。

6.4.3.2.17 模数

6.4.3.2.17.1 CFG_AxModuleRange

数据类型：

U32

R/W：

RW

属性 ID：

629

含义：

设置当轴旋转 360° 时的脉冲个数。

注解：

该值必须为 4 的倍数。

该值用于切向运动和 E-Cam 运动。请参考 Acm_AxTangentInGp、Acm_DevDownLoadCAMTable 和 Acm_AxCamInAx。

对于 PCI-1285，该默认值为 0。PCI-1285E 不支持该属性。

范围：0 ~ 8,000,000。

6.4.3.2.18 同步起停

6.4.3.2.18.1 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。
4096	从轴 _4 的比较信号启动同步运动。
8192	从轴 _5 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。
1048576	当轴 _4 停止时启动同步运动。
2097152	当轴 _5 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。

PCI-1285E 不支持该属性。

6.4.3.2.19 触发停止

6.4.3.2.19.1 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

6.4.3.2.19.2 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

6.4.3.2.19.3 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

6.4.3.2.19.4 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

6.4.3.2.19.5 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

6.4.3.2.19.6 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

6.4.3.2.19.7 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

6.4.3.2.19.8 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

6.4.3.2.19.9 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

6.4.3.2.19.10 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

6.4.3.2.19.11 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

6.4.3.2.19.12 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

6.4.3.3 参数

6.4.3.3.1 速度模式

6.4.3.3.1.1 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。

6.4.3.3.1.2 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。

6.4.3.3.1.3 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。

6.4.3.3.1.4 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。

6.4.3.3.1.5 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。

6.4.3.3.2 原点

6.4.3.3.2.1 PAR_AxHomeCrossDistance

数据类型：

F64

R/W：

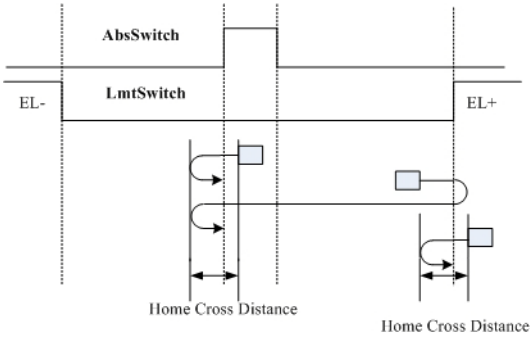
RW

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。



6.4.3.3.2.2 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

6.4.4 群组

6.4.4.1 配置

6.4.4.1.1 系统

6.4.4.1.1.1 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
4	4 轴
5	5 轴
6	6 轴
7	7 轴

注解：

6.4.4.1.2 路径

6.4.4.1.2.1 CFG_GpBldTime

数据类型：

U32

R/W：

RW

属性 ID：

808

含义：

设置 / 获取添加的路径与前一个路径的交接时间。

注解：

该值必须为 0 ~ 65535，且必须为 2 的倍数。单位：ms。
请参考 Acm_GpAddPath。PCI-1285E 不支持此功能。

6.4.4.1.2.2 CFG_GpSFEnable

数据类型：

U32

R/W：

RW

属性 ID：

809

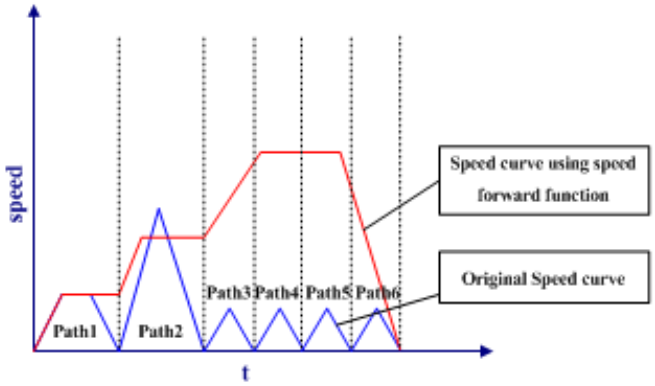
含义：

启用 / 禁用速度前瞻功能。

值	说明
0	禁用
1	启用

注解：

不支持 S 形速度曲线。在这种模式中，不使用 Acm_GpAddPath 中的速度参数，仅使用群组的速度设置。PCI-1285E 不支持此功能。



6.4.4.2 参数

6.4.4.2.1 速度模式

6.4.4.2.1.1 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。

6.4.4.2.1.2 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。

6.4.4.2.1.3 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。

6.4.4.2.1.4 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。

6.4.4.2.1.5 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。

6.4.4.2.2 系统

6.4.4.2.2.1 PAR_GpGroupID

数据类型：

U32

R/W：

R

属性 ID：

706

含义：

通过 GroupHandle 获取 GroupID。

注解：

对于 PCI-1285/1285E，只有 4 个 GroupID 可用，分别为 0、1、2、3。用户不能同时处理多于四个群组。如果已经有四个群组，那么用户必须关闭一个群组才能创建新的群组。

6.4.4.2.3 路径

6.4.4.2.3.1 PAR_GpRefPlane

数据类型：

U32

R/W：

RW

属性 ID：

709

含义：

设置 / 获取螺旋运动和圆弧插补的参考平面。

值	说明
0	XY 平面
1	YZ 平面
2	XZ 平面

注解：

请参考 Acm_GpMoveHelixAbs、Acm_GpMoveHelixRel、Acm_GpMoveHelixAbs_3p 和 Acm_GpMoveHelixRel_3p。

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

软件功能比较表

A.1 软件功能比较表

	项目	说明	PCI-1285	PCI-1285E
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	√	√
		叠加运动	√	
	多个轴 （群组）运动	多达 4 个群组	4 个群组	4 个群组
		Line：多达 8 个轴	3 个轴	2个轴
		2 轴圆弧插补	√	
		速度重设	√	
		螺旋	√	
	返回原点	16 种模式	√	√
	路径表运动	3 个表，大小：10K 个点	√	√
		启动 / 停止运动列表	√	
		线性运动轨迹：多达 8 个轴	√	√
		圆弧运动轨迹： 2 轴	√	
		增加延迟执行功能	√	√
		开始 / 停止 / 重复	√	√
		路径的自动交接功能	√	
应用功能	龙门			
	速度前瞻	速度前瞻 （参考 acm_gpmovepath）	√	
	切线跟随		√	
	电子齿轮		√	√
	电子凸轮			
	错误检查	错误状态，看门狗	√	√
	CAM DO	凸轮区间 DO	√	
	位置锁存		√	
	同步启停	同时开始 / 停止	√	
中断		轴停止 / 群组停止	√	√
触发功能	单一比较	多达 8 个通道	√	
	表比较	多达 2 个通道	√	
	线性比较	（表大小：100K 个点）	√	

附录 B

规格

B.1 轴

Item	Description
Number of axis	8
Type of control output	Pulse Type

B.2 数字量输入

Item		Description
Channels		LMT+,LMT-, ORG, INP, ALM, EMG, LTC, RDY
Type		One terminal, opto-isolated
Input voltage	L(max)	4Vdc
	H(min)	10Vdc
	H(max)	30Vdc
Max. input delay time		150us
Protection		2,500V Isolation
Input resistance		8.4kΩ

B.3 高速数字量输入

Item		Description
Channels		JOG+, JOG-
Type		One terminal, opto-isolated
Input voltage	L(max)	3Vdc
	H(min)	10Vdc
	H(max)	30Vdc
Max. input delay time		1us
Protection		2,500V Isolation
Input resistance		8.4kΩ

B.4 数字量输出

Item		Description
Channels		SVON, ERC, CAM-DO, CMP
Type		One terminal, opto-isolated, sink type
Operation Voltage	Low	10Vdc
	High	30Vdc
Max. sink current		60mA per channel
Max. output delay time		15us
Protection		2,500V Isolation

B.5 输入脉冲

Item		Description
Max frequency		2.5MHz x1, x2, x4 (A/B phase only)
Type		Two terminal, opto-isolated
Input voltage	L(max)	1Vdc
	H(min)	3.5Vdc
	H(max)	10Vdc
Protection		2,500V Isolation
Min. width for Hi / Lo pulse		200ns

B.6 输出脉冲

Item		Description
Max frequency		5Mpps
Type		Two terminal, opto-isolated
Output voltage	L(max)	0.7Vdc
	H(min)	2Vdc
	H(max)	3.9Vdc
Output current		3Vdc/18mA
Output signal mode		Differential line driving output
Protection		2,500V Isolation
Control range		32bit

B.7 一般规格

Item		Description
Connector		(HDRA) Mini-SCSI 200 x1
Dimensions		175mm x 100mm
Certifications		CE, FCC Class A
Power consumption	Typical	3.3V/160mA; 5V/530mA
	Max	3.3V/400mA; 5V/650mA
Temperature	Operating	0-60°C (refer to IEC 60068-2-1,2)
	Storage	-20~85°C
Relative Humidity		5~95% RH non-condensing (refer to IEC 60068-2-3)
External Power Voltage		DC +12 ~ 24 V

ADVANTECH

Enabling an Intelligent Planet

www.advantech.com.cn