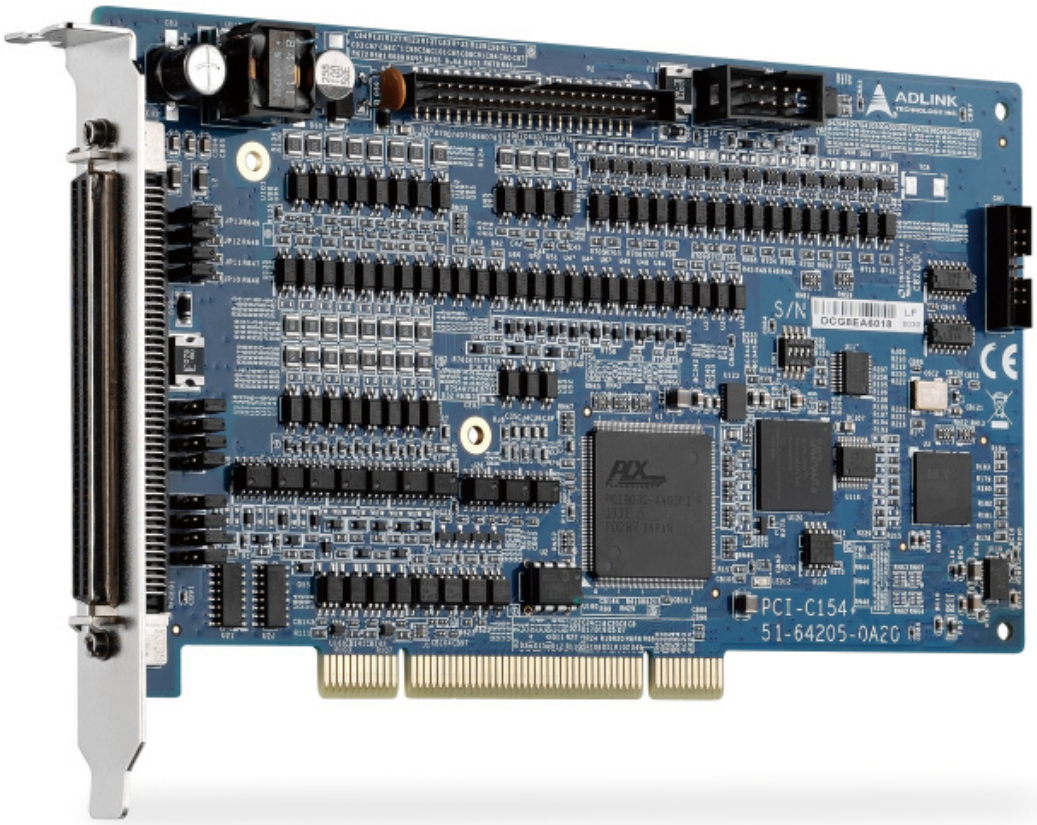
ADLINK TECHNOLOGY INC.

PCI-C154+

4 轴高阶脉冲式运动控制卡

使用手册



手册版本：1.00
更新日期：Dec 03,2014
料号：50-11254-1000

Recycled Paper

Advance Technologies; Automate the World.

变更记录

变更	日期	变更内容
1.00	2014-12-03	初版



请注意，本手册是 PCI-C154+ 函数库参考的初步版本。虽然我们已经尽了最大的努力确保手册的内容准确无误，但后续版本可能包含对规范和操作的更改，这些更改可能是细微的更改、重大更改，也可能是新增的本手册中不包括的全新章节和模块。

如果您要了解详细信息或有任何疑问，请访问我们的网站，网址为 http//.www.adlinktech.com，或者可联系您当地的销售中心，具体联系方式见维修信息。

前言

Copyright 2014 ADLINK Technology, Inc.

本文当中包含受版权保护的专有资料。保留所有权利。事先未经制造商的书面批准，不得以任何机械、电子或其它方式复制本手册的任何部分。

免责声明

为了改进产品的可靠性、设计和功能，本文档中的信息如有更改，恕不另行通知，且本文档中的信息并不代表制造商所作的承诺。

若因产品或文档使用不当而造成直接、间接、特殊、意外或从属损坏 （即使已告知可能造成这种损坏），制造商将不承担任何责任。

环境责任

ADLINK 坚持遵守欧盟颁布的危害物质禁用指令 (RoHS) 以及废弃电子电气设备指令 (WEEE)，以此履行公司对全球环境保护的社会责任。环境保护是 ADLINK 最优先考虑的问题。我们采取的措施可确保我们的产品、制造过程、部件和原材料对环境造成的影响可达到最低程度。产品达到使用寿命时，我们鼓励客户按照其所在国家或公司规定的产品处理和 / 或回收计划处理产品。

商标

本手册中提到的产品名称仅用于标识目的，可能是其各自所属公司的商标和 / 或注册商标。

惯例

请注意本参考文件中使用的下列常用说明，以确保用户可正确地执行某些任务和指令。


註釋

帮助用户执行任务的附加信息、帮助和提示。


注意

用于避免尝试完成任务时造成***轻微***人身伤害、部件损坏、数据丢失和 / 或程序中断的信息。


警告

用于避免尝试完成特定任务时造成***严重***人身伤害、部件损坏、数据丢失和 / 或程序中断的信息

目录

变更记录 ii

前言 iii

图目录 xi

表目录 xiii

1 简介 1

1.1 产品功能 2
1.2 产品规格 4
1.3 软件支援 6
1.3.1 软件函数库 6
1.3.2 MotionCreatorPro 6
1.4 端子板 7
1.5 PCI-C154+ 与 MP-C154 差异 7

2 开始安装 9

2.1 确认包装内项目 9
2.2 PCI-C154+ 外观轮廓 9
2.3 硬件安装 10
2.3.1 硬件组态 10
2.3.2 PCI 插槽选择 10
2.3.3 硬件安装程序 10
2.4 软件安装流程 10
2.5 CN3 连接器信号定义 11
2.6 P2 连接器信号定义 12
2.7 JP2 至 JP9 jumper 设定 15
2.8 JP10 至 JP13 jumper 设定 15
2.9 SW1 卡片编号开关 16
2.10 CN4 脉冲发生器输入连接器 17

3 信号连接接口 18
3.1 脉冲控制命令信号 (OUT/DIR) 18
3.2 编码器输入信号 (EA, EB and EZ) 20
3.3 原点开关信号 (ORG) 22
3.4 机械限位开关信号 (PEL/MEL) 23
3.5 到位 / 零速度检出信号 (INP) 23
3.6 伺服警报信号 (ALM) 24
3.7 位置计数误差清除信号 (ERC) 25
3.8 伺服激磁信号 (SVON) 26
3.9 伺服预备信号 (RDY) 26
3.10 位置比较触发输出信号 (TRG) 27
3.11 多功能输入信号 (DI/SD/CLR/EMG/LTC) 28
3.12 脉冲发生器输入信号 (PA/PB) 29
3.13 扩展通用输出信号 (EDO) 30
3.14 扩展通用输入信号 (EDI) 32

4 操作原理介绍 33
4.1 运动控制器简介 33
4.1.1 电压运动控制接口 (Voltage Motion Control Interface) 33
4.1.2 脉冲运动控制接口 (Pulse Motion Control Interface) 33
4.1.3 网络型运动控制接口 (Network Motion Control Interface) 33
4.1.4 实时性运动控制软件 (Software Real-Time Motion Control Kernel) . 34
4.1.5 运动控制 DSP(DSP Motion Control Kernel) .. 34
4.1.6 运动控制芯片 (ASIC Motion Control Kernel)) 34
4.1.7 运动控制型态比较表 34
4.1.8 PCI-C154+ 运动控制器型态 35

4.2 运动控制模式 35
4.2.1 坐标系统 (Coordinate System) 35
4.2.2 绝对 / 相对运动 (Absolute/Relative Position Move) 36
4.2.3 T 型速度曲线 (T-Curve Speed Profile) 37
4.2.4 S 型速度曲线 (S-Curve Speed Profile) 38
4.2.5 速度运动模式 (Velocity Mode) 40
4.2.6 单轴运动模式 (One Axis Position Mode) 40
4.2.7 多轴直线插补运动模式 (Linear Interpolation Mode) 41
4.2.8 两轴圆弧插补运动模式 (Circular Interpolation Mode) 42
4.2.9 连续运动 (Continuous Motion) 43
4.2.10 归零运动 (Home Return) 45
4.2.11 零点位置搜寻功能 (Home Search) 51
4.2.12 手动脉冲发生器功能 (Manual Pulser) 51
4.2.13 同步启动功能 (Simultaneous Start) 52
4.2.14 速度重设功能 (Speed Override) 52
4.3 连接马达驱动器接口 53
4.3.1 脉冲命令输出接口 (OUT/DIR) 53
4.3.2 编码器输入接口 (EA/EB) 55
4.3.3 到位检出信号 (INP) 56
4.3.4 伺服警报信号 (ALARM) 57
4.3.5 误差清除信号 (ERC) 57
4.3.6 伺服激磁信号 (SVON) 57
4.3.7 伺服预备信号 (RDY) 57
4.3.8 伺服警报重置信号 58
4.4 机械开关接口 58
4.4.1 原点开关信号 (ORG) 58
4.4.2 限位开关信号 (PEL/MEL) 58
4.4.3 降速信号 (SD) 58
4.4.4 计数器清除开关信号 (CLR) 58
4.4.5 紧急停止开关信号 (EMG) 58

4.5 计数器 (Counters) 59
4.5.1 命令位置计数器 (Command Position Counter) 59
4.5.2 反馈位置计数器 (Feedback Position Counter) 59
4.5.3 位置误差计数器 (Command and Feedback Error Counter) 59
4.5.4 通用计数器 (General Purpose Counter) 60
4.6 高速位置比较触发 (High Speed Position Compare Trigger) 60
4.6.1 时间计数器触发 (Timer Trigger) 61
4.6.2 表格比较触发 (Table Compare Trigger) 61
4.6.3 线性比较触发 (Linear Compare Trigger) 61
4.6.4 单点比较触发 (Single Compare Trigger) 61
4.6.5 手动触发 (Manual Trigger) 61
4.7 高速位置撷取功能 (Position Latch) 62
4.8 其他运动控制功能 62
4.8.1 背隙补偿 与 滑移修正 (Backlash Compensation & Slip Correction) . 62
4.8.2 震动抑制功能 (Vibration Restriction) 62
4.9 中断控制 (Interrupt Control) 63
4.10 多卡操作 66

5 MotionCreatorPro 67

5.1 执行 MotionCreatorPro 67
5.2 关于 MotionCreatorPro 67
5.3 MotionCreatorPro 介绍 68
5.3.1 主选单 68
5.3.2 选择单 69
5.3.3 设定选单 70
5.3.4 单轴操作页面 73
5.3.5 两轴与四轴操作页面 78
5.3.6 二维运动操作页面 81

6 函数库 87

附录 88

A 传统专用函数列表 88
B C/C++ 函数库 92
C 系统与初始化 (System and Initialization) 93
D 脉冲输出入设定 (Pulse Input/Output Configuration) 96
E 速度运动模式 (Velocity mode motion) 98
F 单轴运动模式 (Single Axis Position Mode) 101
G 多轴直线插补运动 (Linear Interpolated Motion) 104
H 两轴圆弧插补运动 (Circular Interpolation Motion) 112
I 归零运动 (Home Return Mode) 118
J 脉冲发生器运动模式 (Manual Pulser Motion) 121
K 运动状态 (Motion Status) 124
L 运动界面 I/O(Motion Interface I/O) 125
M 中断控制 (Interrupt Control) 132
N 位置控制与计数器 (Position Control and Counters) 134
O 连续运动 (Continuous motion) 137
P 多轴同步操作 (Multiple Axes Simultaneous Operation) ... 139
Q 多功能输出入信号 (Mulit-Function DIO) 142
R 软件极限 (Soft Limit) 144
S 扩展通用输出入信号 (Extened DIO) 145
T 背隙补偿与震动抑制 (Backlash Compensation / Vibration Suppression) 147
U 函数回传值 (Return Code) 149

重要安全说明 151

服务信息 153

图目录

图 1-1： PCI-C154+ 系统方块图 1
图 1-2： 开发流程图 2
图 2-1： PCI-C154+ 产品外观轮廓图 9

表目录

表 2-1： CN3 连接器信号定义 12
表 2-2： P2 IDE 44pin 端连接器信号定义 13
表 2-3： P2 DSUB 37pin 端连接器信号定义 14
表 2-4： SW1 卡片编号开关 16
表 2-5： CN4 脉冲发生器输入连接器 17
表 4-1： 运动中断源位设定 63
表 4-2： 错误中断回传值 64
表 4-3： 位置比对与撷取中断源位设定 65
表 7-1： 数据型态定义 93

1 简介

PCI-C154+ 为高阶四轴运动控制卡，可将多张卡配置于同一系统中，提供高速脉冲信号输出 (6.55MHz)、编码器输入、32 个扩展通用输出入通通道、高速位置比较触发输出、线性与圆弧插补、连续插补功能，同时也提供位置与速度实时重设功能。

图 1-1 为 PCI-C154+ 的系统方块图，运动控制功能包括 T 型与 S 型速度曲线、线性与圆弧插补、13 种归零运动模式，所有功能皆由运动控制芯片完成，不会消耗 CPU 资源。

PCI-C154+ 同时提供三个便利的功能

1. 卡片编号设定：
   卡片编号可透过拨码开关设定，设定范围为 0~15。
2. 紧急停止输入
   紧急停止输入脚可连接于紧急按钮以停止脉冲控制命令输出。
3. 软件安全保护
   可设定 16-bit 值于 EEPROM，以防止程序抄袭。

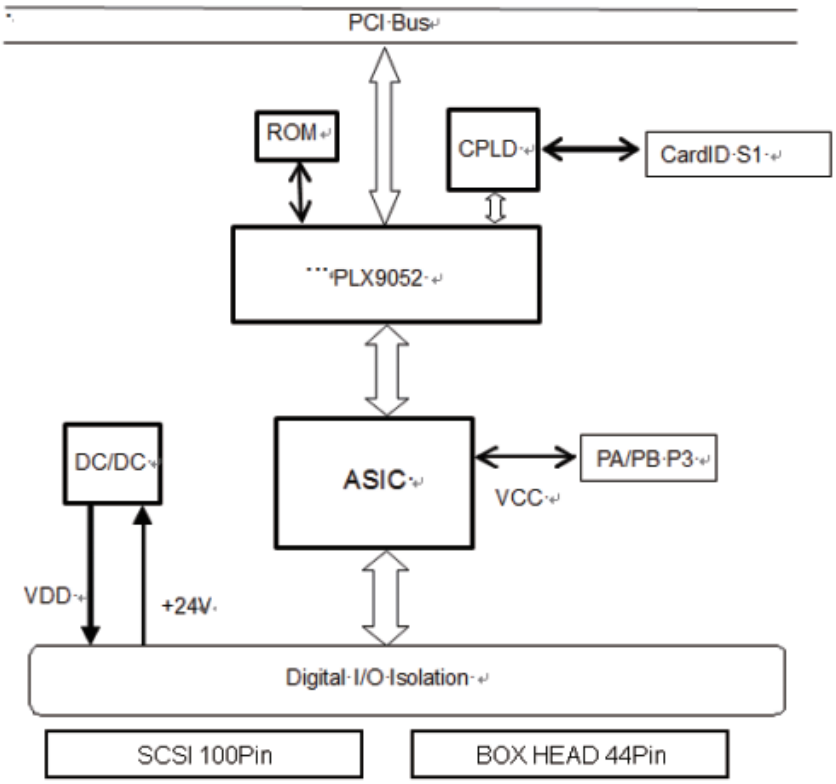


图 1-1：PCI-C154+ 系统方块图

MotionCreatorPro 为 Windows 操作系统下的开发软件，可帮助使用者于开发阶段，进行侦错，同时可显示各轴信息、I/O 信号状态等。

提供 C++ 与 VB 软件函数库，以及范例程序，说明如何操作各项功能。

图 1-2 为使用本手册流程图，请详阅各章节内容来完成开发。

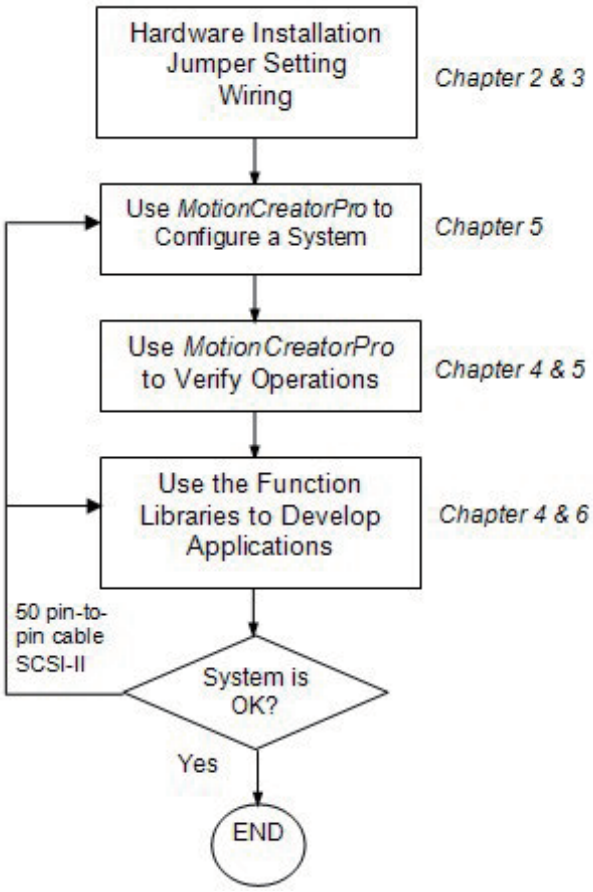


图 1-2：开发流程图

1.1 产品功能

- 32-bit PCI 总线
- 4 轴步进 / 伺服脉冲输出信号
- 最大脉冲输出频率：6.55MPPS
- 脉冲输出模式：OUT/DIR, CW/CCW, AB 相位
- 编码器输入模式：CW/CCW, AB 相位 x1, x2, x4

- 最大编码器输入频率： 3 Mhz （CW/CCW 模式或 AB 相位 X1 模式）； （AB 相位 x4 模式可达到 6.55Mhz）
- 可调整的加减速时间
- T 型与 S 型速度曲线
- 2 至 4 轴线性插补
- 2 轴圆弧插补
- 连续插补运动
- 实时位置 / 速度重设功能
- 13 种归零运动模式
- 背隙补偿与滑移修正
- 每轴提供 2 个软件极限
- 28-bit 正反向编码器反馈输入
- 原点开关 (ORG)、编码器索引信号 (EZ)、机械限位开关 (PEL/MEL)
- 32 个扩展通用输出入通道
- 所有数字输出入信号皆由 2500Vrms 隔离组件保护
- 可程序中断来源
- 多轴同步开始 / 停止
- 脉冲发生器输入接口
- 卡片编码设定
- 软件安全保护机制
- 紧急停止输入
- 软件提供最多于同一系统控制 12 张 PCI-C154+
- 高速位置比较触发
- 高速位置撷取
- 提供 MotionCreatorPro
- 提供 PCI-C154+ 软件函数库以及范例程序

1.2 产品规格

- **应用马达：**
  - ▶ 步进马达
  - ▶ 脉冲式 AC 或 DC 伺服马达
- **效能：**
  - ▶ 可控制轴数：4
  - ▶ 最大脉冲输出频率：6.55MPPS
  - ▶ 内部参考时钟：19.66MHz
  - ▶ 28-bit 正反向计数器，范围：0 至 268,435,455 或 –134,217,728 至 +134,217,727
  - ▶ 位置脉冲设定范围 (28-bit)：-134, 217, 728 至 +134, 217, 728
  - ▶ 脉冲速度设定范围：
    - ▷0.1 PPS to 6553.5 PPS. (倍数 = 0.1)
    - ▷1 PPS to 65535 PPS. (倍数 = 1)
    - ▷100 PPS to 6553500 PPS. (倍数 = 100)
  - ▶ 位置比较触发输出 (TRG) 速度：1MHz
  - ▶ 位置比较触发输出 (TRG) 脉冲宽度：0.54 us - 3.931 ms (step：480ns)
  - ▶ 位置撷取输入信号 (LTC) 速度：1MHz

- **I/O 信号：**
  - 所有 I/O 皆由光隔离组件保护，可承受 2500Vrms
  - 脉冲命令输出：OUT/DIR
  - 编码器输入：EA/EB
  - 编码器索引输入：EZ
  - 机械极限 / 原点开关：±EL/ORG
  - 多功能输入信号：DI/SD/CLR/EMG/LTC
  - 位置比较触发输出信号：TRG
  - 伺服马达接口：INP/ALM/ERC
  - 伺服激磁信号：SVON
  - 伺服预备信号：RDY
  - 脉冲发生器输入信号：PA/PB
  - 扩展通用输出信号：EDO
  - 扩展通用输入信号：EDI
- **一般规格：**
  - 连接器：100-pin SCSI-type 以及 44-pin BOX HEAD
  - 操作温度：0°C - 50°C
  - 储存温度：-20°C - 80°C
  - 湿度：5 - 85%, non-condensing
- **产品功耗**
  - PCI 插槽电源供应 (输入)：+5V DC ±5%, 900mA max
  - 外部电源供应 (输入)：+24V DC ±5%, 500mA max
- **PCI-C154+ 尺寸 (PCB 尺寸)：**
  - 185mm(L) X 106.68 mm(W)

1.3 软件支援

1.3.1 软件函数库

提供 Windows XP/7/8.1 DLLs

1.3.2 MotionCreatorPro

此为 Windows 操作系统下的软件，用于设定板卡、马达及系统，亦可用来做硬件与软件纠错。用户可透过此软件设定 I/O 逻辑及读取参数。

1.4 端子板

ADLINK 提供伺服与步进马达专用端子版 DIN-100S、 DIN-814-GP、DIN-814 系列，以及扩展通用输出入端子版 DIN-37D。

1.5 PCI-C154+ 与 MP-C154 差异

PCI-C154+ 为基于 MP-C154 功能，再新增功能，以下为其差异：

硬件部分：

	PCI-C154+	MP-C154
新增 16 组扩展通用输出 (EDO) (新增 P2 连接器，请参考 3.13 节)	sink current capability：250mA (COM)	N/A
新增 16 组扩展通用输入 (EDI) (新增 P2 连接器，请参考 3.14 节)	source current capability：6.5mA(24V)	N/A
4 组多功能输出 (CN3 连接器 - pin40,46,90,96，请参考 3.10 节)	1MHz (COM or 5V by jumper10~13)	5KHz (COM)
4 组多功能输入 (CN3 连接器 - pin39,45,89,95，请参考 3.11 节)	1MHz (5V)	100KHz (5V)

软件部分：

	PCI-C154+	MP-C154
设定 16 组扩展通用输出 (EDO)， 请参考附录 S	Yes	N/A
读取 16 组扩展通用输入 (EDI)， 请参考附录 S	Yes	N/A
位置比较触发功能， 请参考 4.6 节	4CH, 1MHz, signal/linear/FIFO(1026/CH) compare trigger trigger-out pulse width： 0.54us ~ 3.93ms(step:0.48us)	N/A
位置撷取功能， 请参考 4.7 节	4CH, 28bit encoder latch 1MHz (FIFO, 258/CH)	N/A
传统函数库 (C154_XXX)	Yes	Yes
APS 函数库	Yes	N/A

2 开始安装

本章节将教导如何成功安装 PCI-C154+，并完成硬件及软件安装，以及各式 I/O 接线。

- 确认包装内项目 (2.1 节)
- 确认 PCB (2.2 节)
- 硬件安装 (2.3 节)
- 软件安装 (2.4 节)
- 信号连接接口 (第 3 章) 以及操作原理介绍 (第 4 章)
- 连接器信号定义以及各式 I/O 接线 (本章其余小节)

2.1 确认包装内项目

除本手册之外，产品包装纸盒内应包含：

- PCI-C154+ 一张
- IDE 44p – DSUB 37p 扁平电缆一条
- 产品保证卡

如果有任何以上项目遗失或毁损，请速与您的经销商联络。并保留完整产品包装及内容物，以利新品更换或维修。

2.2 PCI-C154+ 外观轮廓

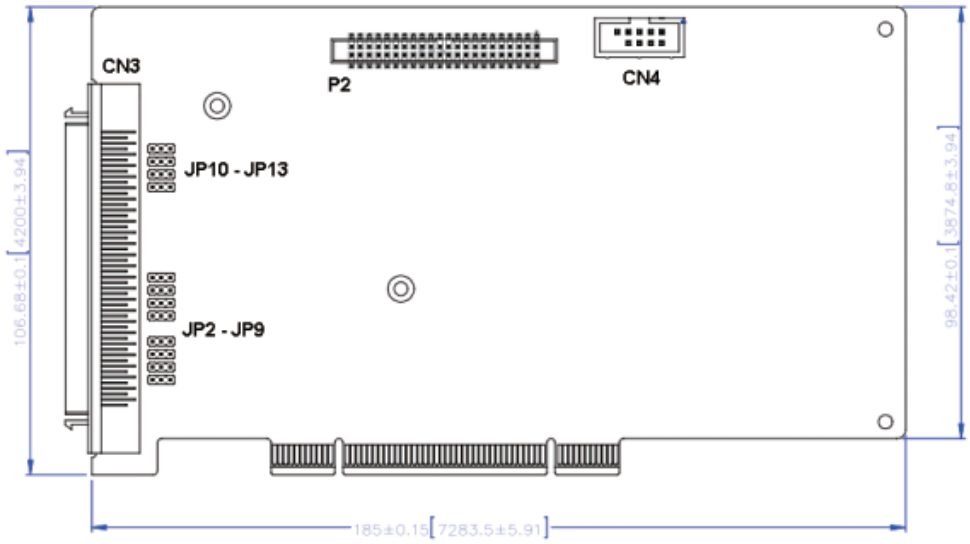


图 2-1：PCI-C154+ 产品外观轮廓图

- CN3：运动控制信号连接器 (100-pin)
- P2：扩展通用输出入信号连接器 (44-pin)
- CN4：脉冲发生器输入连接器
- SW1：卡片编码开关 (0-15)
- JP2-JP9：脉冲控制命令信号设定 (Line Driver / Open Collector)
- JP10-JP13：多功能输出信号电源设定

2.3 硬件安装

2.3.1 硬件组态

PCI-C154+ 采用 PCI Rev. 2.2 汇排流规格，只支持 33MHz 模式。系统 BIOS 可自动配置内存与 IRQ 通道。

2.3.2 PCI 插槽选择

部分系统会同时拥有 PCI 以及 ISA 插槽，请注意不可将 PCI 板卡插入 PC/AT 插槽内。

2.3.3 硬件安装程序

1. 阅读此手册并依据您的应用正确设定 jumper。
2. 关闭计算机电源及所有端子板上相关电源。
3. 将 PCI-C154+ 安装至计算机内 32-bit PCI 插槽。(通常为白色插槽)(请确保做好静电保护处理)
4. 确保利用螺丝将 PCI-C154+ 锁紧至系统机壳上。

2.4 软件安装流程

1. 执行 PCI-C154 series WDM 档案。
2. 随着提示击点，完成安装。
3. 完成安装后，请重新开启计算机。

建议：请至凌华科技官方网站页面下载最新的安装软件。

2.5 CN3 连接器信号定义

CN3 为运动控制主要连接器。

No.	Name	I/O	Function	No.	Name	I/O	Function
1	VDD	O	+5V power supply output	**51**	VDD	O	+5V power supply output
2	EXGND	-	Ext. power ground	**52**	EXGND	-	Ext. power ground
3	OUT0+	O	Pulse signal (+)	**53**	OUT2+	O	Pulse signal (+)
4	OUT0-	O	Pulse signal (-)	**54**	OUT2-	O	Pulse signal (-)
5	DIR0+	O	Dir. signal (+)	**55**	DIR2+	O	Dir. signal (+)
6	DIR0-	O	Dir. signal (-)	**56**	DIR2-	O	Dir. signal (-)
7	SVON0	O	Servo On/Off	**57**	SVON2	O	Servo On/Off
8	ERC0	O	Dev. ctr, clr. Signal	**58**	ERC2	O	Dev. ctr, clr. signal
9	ALM0	I	Alarm signal	**59**	ALM2	I	Alarm signal
10	INP0	I	In-position signal	**60**	INP2	I	In-position signal
11	RDY0	I	Multi-purpose Input signal	**61**	RDY2	I	Multi-purpose Input signal
12	EXGND		Ext. power ground	**62**	EXGND		Ext. power ground
13	EA0+	I	Encoder A-phase (+)	**63**	EA2+	I	Encoder A-phase (+)
14	EA0-	I	Encoder A-phase (-)	**64**	EA2-	I	Encoder A-phase (-)
15	EB0+	I	Encoder B-phase (+)	**65**	EB2+	I	Encoder B-phase (+)
16	EB0-	I	Encoder B-phase (-)	**66**	EB2-	I	Encoder B-phase (-)
17	EZ0+	I	Encoder Z-phase (+)	**67**	EZ2+	I	Encoder Z-phase (+)
18	EZ0-	I	Encoder Z-phase (-)	**68**	EZ2-	I	Encoder Z-phase (-)
19	VDD	O	+5V power supply output	**69**	VDD	O	+5V power supply output
20	EXGND	-	Ext. power ground	**70**	EXGND	-	Ext. power ground
21	OUT1+	O	Pulse signal (+)	**71**	OUT3+	O	Pulse signal (+)
22	OUT1-	O	Pulse signal (-)	**72**	OUT3-	O	Pulse signal (-)
23	DIR1+	O	Dir. signal (+)	**73**	DIR3+	O	Dir. signal (+)
24	DIR1-	O	Dir. signal (-)	**74**	DIR3-	O	Dir. signal (-)
25	SVON1	O	Servo On/Off	**75**	SVON3	O	Servo On/Off
26	ERC1	O	Dev. ctr, clr. Signal	**76**	ERC3	O	Dev. ctr, clr. signal
27	ALM1	I	Alarm signal	**77**	ALM3	I	Alarm signal
28	INP1	I	In-position signal	**78**	INP3	I	In-position signal
29	RDY1	I	Multi-purpose Input signal	**79**	RDY3	I	Multi-purpose Input signal
30	EXGND		Ext. power ground	**80**	EXGND		Ext. power ground
31	EA1+	I	Encoder A-phase (+)	**81**	EA3+	I	Encoder A-phase (+)
32	EA1-	I	Encoder A-phase (-)	**82**	EA3-	I	Encoder A-phase (-)
33	EB1+	I	Encoder B-phase (+)	**83**	EB3+	I	Encoder B-phase (+)
34	EB1-	I	Encoder B-phase (-)	**84**	EB3-	I	Encoder B-phase (-)

No.	Name	I/O	Function	No.	Name	I/O	Function
35	EZ1+	I	Encoder Z-phase (+)	**85**	EZ3+	I	Encoder Z-phase (+)
36	EZ1-	I	Encoder Z-phase (-)	**86**	EZ3-	I	Encoder Z-phase (-)
37	PEL0	I	End limit signal (+)	**87**	PEL2	I	End limit signal (+)
38	MEL0	I	End limit signal (-)	**88**	MEL2	I	End limit signal (-)
39	DI0	I	DI/SD/CLR/EMG/LTC	**89**	DI2	I	DI/SD/CLR/EMG/LTC
40	TRG0	O	TRG	**90**	TRG2	O	TRG
41	ORG0	I	Origin signal	**91**	ORG2	I	Origin signal
42	EXGND		Ext. power ground	**92**	EXGND		Ext. power ground
43	PEL1	I	End limit signal (+)	**93**	PEL3	I	End limit signal (+)
44	MEL1	I	End limit signal (-)	**94**	MEL3	I	End limit signal (-)
45	DI1	I	DI/SD/CLR/EMG/LTC	**95**	DI3	I	DI/SD/CLR/EMG/LTC
46	TRG1	O	TRG	**96**	TRG3	O	TRG
47	ORG1	I	Origin signal	**97**	ORG3	I	Origin signal
48	EXGND	-	Ext. power ground	**98**	EXGND	-	Ext. power ground
49	EXGND	-	Ext. power ground	**99**	E_24V	-	Isolation power Input, +24V
50	EXGND	-	Ext. power ground	**100**	E_24V	-	Isolation power Input, +24V

表 2-1：CN3 连接器信号定义

2.6 P2 连接器信号定义

P2 为扩展通用输出入 (EDI/O) 连接器。将使用 IDE 44pin – DSUB 37pin 扁平电缆连接至端子版。

注意：务必注意 P2 上 EX_GND 接线数量，建议如下 :

EDO 总电流负载为 0~1A: 连接至少一组 P2 上 E X_GND

EDO 总电流负载为 1~2A: 连接至少两组 P2 上 E X_GND

EDO 总电流负载为 2~3A: 连接至少三组 P2 上 E X_GND

EDO 总电流负载为 3~4A: 连接四组 P2 上 E X_GND

下表为 IDE 44pin 端信号定义

No.	Name	I/O	Function	No.	Name	I/O	Function
1	EX_GND	--	External Power Ground	**2**	EX_GND	--	External Power Ground
3	EDI0	I	Digital Input 0	**4**	EDI1	I	Digital Input 1
5	EDI2	I	Digital Input 2	**6**	EDI3	I	Digital Input 3

No.	Name	I/O	Function	No.	Name	I/O	Function
7	EDI4	I	Digital Input 4	**8**	EDI5	I	Digital Input 5
9	VDD	O	External +5V Power	**10**	EX_GND	--	External Power Ground
11	EDI6	I	Digital Input 6	**12**	EDI7	I	Digital Input 7
13	EDI8	I	Digital Input 8	**14**	EDI9	I	Digital Input 9
15	EDI10	I	Digital Input 10	**16**	EDI11	I	Digital Input 11
17	EX_GND	--	External Power Ground	**18**	EX_GND	--	External Power Ground
19	EDI12	I	Digital Input 12	**20**	EDI13	I	Digital Input 13
21	EDI14	I	Digital Input 14	**22**	EDI15	I	Digital Input 15
23	EDO0	O	Digital Output 0	**24**	EDO1	O	Digital Output 1
25	EDO2	O	Digital Output 2	**26**	EDO3	O	Digital Output 3
27	EX_GND	--	External Power Ground	**28**	EX_GND	--	External Power Ground
29	EDO4	O	Digital Output 4	**30**	EDO5	O	Digital Output 5
31	EDO6	O	Digital Output 6	**32**	EDO7	O	Digital Output 7
33	EDO8	O	Digital Output 8	**34**	EDO9	O	Digital Output 9
35	EX_GND	--	External Power Ground	**36**	VDD	O	External +5V Power
37	EDO10	O	Digital Output 10	**38**	EDO11	O	Digital Output 11
39	EDO12	O	Digital Output 12	**40**	EDO13	O	Digital Output 13
41	EDO14	O	Digital Output 14	**42**	EDO15	O	Digital Output 15
43	EX_GND	--	External Power Ground	**44**	EX_GND	--	External Power Ground

表 2-2：P2 IDE 44pin 端连接器信号定义

下表为 DSUB 37pin 端信号定义

No.	Name	I/O	Function	No.	Name	I/O	Function
1	EX_GND	--	External Power Ground	**20**	EX_GND	--	External Power Ground
2	EDI0	I	Digital Input 0	**21**	EDO0	O	Digital Output 0
3	EDI1	I	Digital Input 1	**22**	EDO1	O	Digital Output 1
4	EDI2	I	Digital Input 2	**23**	EDO2	O	Digital Output 2
5	EDI3	I	Digital Input 3	**24**	EDO3	O	Digital Output 3
6	EDI4	I	Digital Input 4	**25**	EDO4	O	Digital Output 4
7	EDI5	I	Digital Input 5	**26**	EDO5	O	Digital Output 5
8	EDI6	I	Digital Input 6	**27**	EDO6	O	Digital Output 6
9	EDI7	I	Digital Input 7	**28**	EDO7	O	Digital Output 7
10	EDI8	I	Digital Input 8	**29**	EDO8	O	Digital Output 8
11	EDI9	I	Digital Input 9	**30**	EDO9	O	Digital Output 9
12	EDI10	I	Digital Input 10	**31**	EDO10	O	Digital Output 10
13	EDI11	I	Digital Input 11	**32**	EDO11	O	Digital Output 11
14	EDI12	I	Digital Input 12	**33**	EDO12	O	Digital Output 12
15	EDI13	I	Digital Input 13	**34**	EDO13	O	Digital Output 13

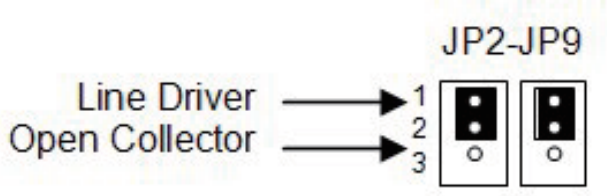
No.	Name	I/O	Function	No.	Name	I/O	Function
16	EDI14	I	Digital Input 14	**35**	EDO14	O	Digital Output 14
17	EDI15	I	Digital Input 15	**36**	EDO15	O	Digital Output 15
18	EX_GND	--	External Power Ground	**37**	EX_GND	--	External Power Ground
19	VDD	O	External +5V Power	**--**	--	--	--

表 2-3：P2 DSUB 37pin 端连接器信号定义

2.7 JP2 至 JP9 jumper 设定

JP2-JP9 使用于设定脉冲控制命令信号 (DIR/OUT) 输出接口，可分为差分 Line Driver 或开集极输出 (Open Collector Output)。详细请参阅 3.1 节；默认为差分 Line Driver 输出接口。

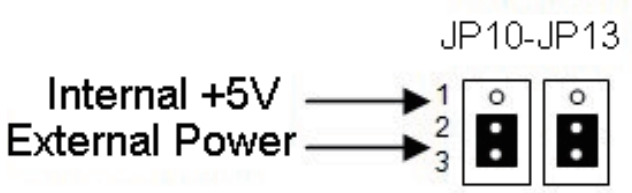
JP6 & JP7	Axis 0
JP4 & JP5	Axis 1
JP8 & JP9	Axis 2
JP2 & JP3	Axis 3



2.8 JP10 至 JP13 jumper 设定

JP10-JP13 使用于触发信号 (TRG) 输出电源设定，可分为使用内部 5V 电源 (内部 330 欧姆限流电阻) 或外部电源 (使用者需自行接限流电阻)。详细请参阅 3.10 节；预设为使用外部电源。

JP10	TRG 0
JP11	TRG 1
JP12	TRG 2
JP13	TRG 3



2.9 SW1 卡片编号开关

此开关被使用于调整各卡片编号，方便用户于应用程序中能轻易辨识该卡的卡片编号。例如当使用者将卡编号调整至”0-0-0-1”(OFF-OFF-OFF-ON)，则意味该卡编号为 ”1”，其中 LSB 为编号开关上的 ”4”，调整编号表如下：

Card ID	Switch Setting (ON=1)
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
10	1010
11	1011
12	1100
13	1101
14	1110
15	1111

表 2-4：SW1 卡片编号开关

2.10 CN4 脉冲发生器输入连接器

CN4 上脚位为脉冲发生器输入脚位。

No.	Name	Function (Axis)
1	VDD	Isolated Power +5V
2	PA+	Pulser A+ phase signal input
3	PA-	Pulser A-phase signal input
4	PB+	Pulser B+ phase signal input
5	PB-	Pulser B-phase signal input
6	EXGND	External Ground
7	N/A	Not Available
8	N/A	Not Available
9	N/A	Not Available

表 2-5：CN4 脉冲发生器输入连接器

3 信号连接接口

本章将介绍所有 I/O 信号连接时所需注意事项，以及相关设定。请于连接 PCI-C154+ 与马达驱动器前，先参阅本章。

3.1 脉冲控制命令信号 (OUT/DIR)

PCI-C154+ 提供 4 轴脉冲控制命令通道， OUT 与 DIR 差分信号分别为脉冲信号以及方向指示信号。除此之外，OUT 与 DIR 信号亦可设定为 CW 与 CCW 信号。信号逻辑详细描述，请参阅 4.3.1 节。本节将详述 OUT 与 DIR 的电气特性。每组信号皆为一对差分信号，例如，OUT0 包含了 OUT0+ 与 OUT0- 信号。下列表格列出 CN3 上的脉冲控制信号。

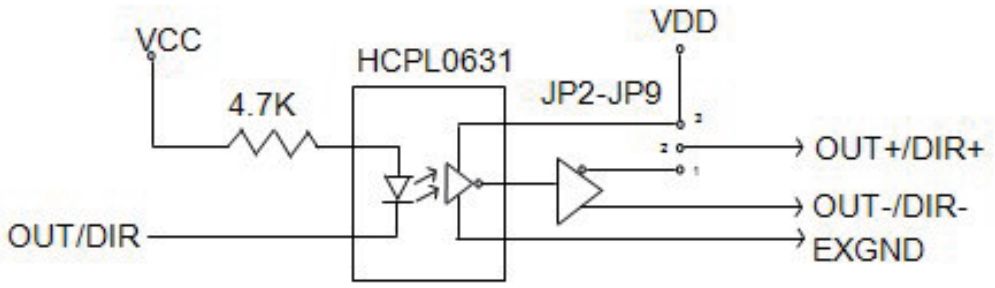
CN3 Pin No.	Signal Name	Description	Axis #
3	**OUT0+**	Pulse signals (+)	0
4	**OUT0-**	Pulse signals (-)	0
5	**DIR0+**	Direction signal (+)	0
6	**DIR0-**	Direction signal (-)	0
21	**OUT1+**	Pulse signals (+)	1
22	**OUT1-**	Pulse signals (-)	1
23	**DIR1+**	Direction signal (+)	1
24	**DIR1-**	Direction signal (-)	1
53	**OUT2+**	Pulse signals (+)	2
54	**OUT2-**	Pulse signals (-)	2
55	**DIR2+**	Direction signal (+)	2
56	**DIR2-**	Direction signal (-)	2
71	**OUT3+**	Pulse signals (+)	3
72	**OUT3-**	Pulse signals (-)	3
73	**DIR3+**	Direction signal (+)	3
74	**DIR3-**	Direction signal (-)	3

OUT 或 DIR 信号可经由 jumper 设定为差分 Line Driver 或开集极输出 (Open Collector Output)。将 jumper 脚位 1 与脚位 2 短路，或脚位 2 与脚位 3 短路，来选择输出模式。如下表：

Output Signal	For differential line driver output, close breaks between 1 and 2 of:	For open collector output, close breaks between 2 and 3 of:
OUT0+	JP6	JP6
DIR0+	JP7	JP7
OUT1+	JP4	JP4
DIR1+	JP5	JP5
OUT2+	JP9	JP9
DIR2+	JP8	JP8
OUT3+	JP3	JP3
DIR3+	JP2	JP2

OUT 与 DIR 之默认设定为差分 Line Driver 模式。

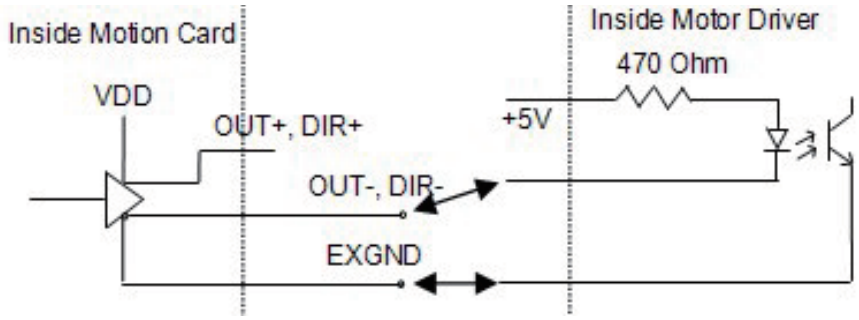
下图为 PCI-C154+ 上 OUT 与 DIR 电路。



注意

若脉冲控制信号输出为开集极输出模式， OUT- 与 DIR- 为输出 OUT 与 DIR 信号；其最大 Sink Current 为 20mA。

建议：将 jumper 脚位 2 与脚位 3 短路，且将 OUT-/DIR- 连接于驱动器的脉冲控制信号输入 。如下图所示：



注意

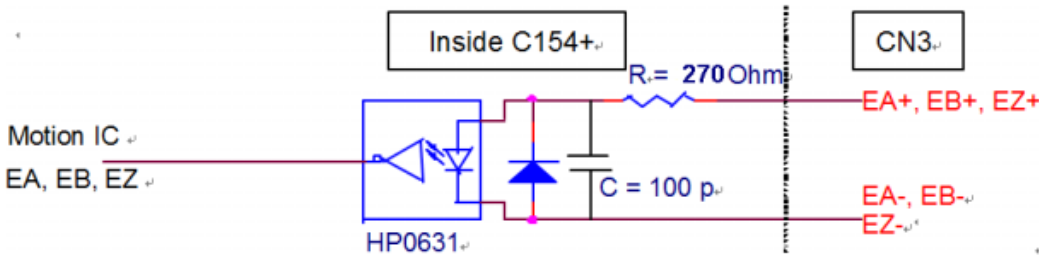
Sink Current 不可超过 20mA，以避免 26LS31 组件烧毁！

3.2 编码器输入信号 (EA, EB and EZ)

PCI-C154+ 提供 4 组编码器输入通道，每组通道包括 EA、 EB 与 EZ 信号。而每组 EA、EB 与 EZ 信号包含一对差分信号 。EA 与 EB 为位置计数信号， EZ 为原点指示信号。其对应脚位如下表：

CN3 Pin No.	Signal Name	Axis #	CN3 Pin No.	Signal Name	Axis #
13	**EA0+**	0	**14**	**EA0-**	0
15	**EB0+**	0	**16**	**EB0-**	0
31	**EA1+**	1	**32**	**EA1-**	1
33	**EB1+**	1	**34**	**EB1-**	1
63	**EA2+**	2	**64**	**EA2-**	2
65	**EB2+**	2	**66**	**EB2-**	2
81	**EA3+**	3	**82**	**EA3-**	3
83	**EB3+**	3	**84**	**EB3-**	3
CN3 Pin No.	**Signal Name**	**Axis #**	**CN3 Pin No.**	**Signal Name**	**Axis #**
17	**EZ0+**	0	**18**	**EZ0-**	0
35	**EZ1+**	1	**36**	**EZ1-**	1
67	**EZ2+**	2	**68**	**EZ2-**	2
85	**EZ3+**	3	**86**	**EZ3-**	3

EA、 EB 与 EZ 信号输入电路如下图：

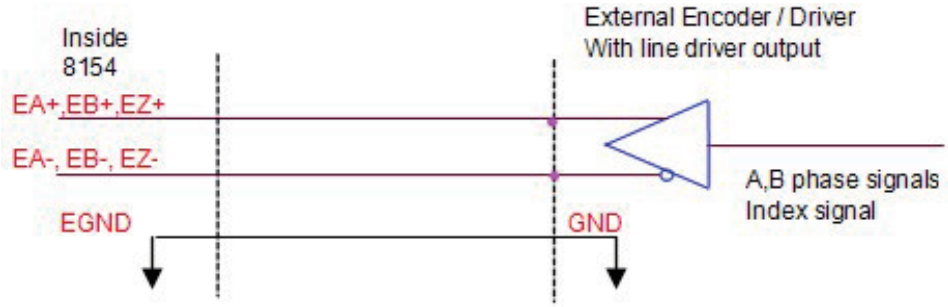


请注意每组差分编码器输入信号之电压至少要超过 3.5V。差分信号将转换成数字信号，输入运动控制芯片中。

以下将介绍如何连接外部电路，编码器输入信号可连接编码器或马达驱动器，其信号可分为 (1) 差分 Line Driver 或 (2) 开集极输出。

- **连接 Line Driver 输出**

若要驱动 PCI-C154+ 编码器输入隔离组件，驱动器输出必须提供至少 3.5V 差分信号，以及至少 8mA 的驱动能力；除此之外，两端的地线必须相连。最大输入频率为 3MHz，但实际状况取决于接线长度以及信号质量。



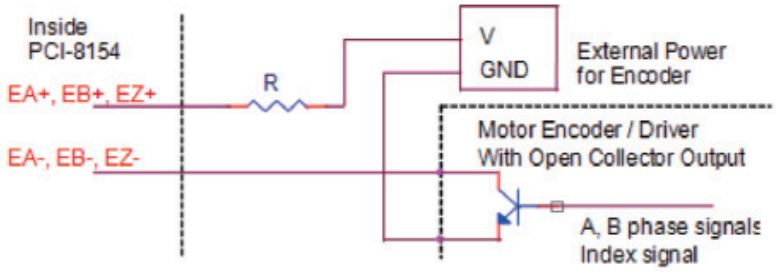
- **连接开集极输出**

若连接开集极输出，必须连接外部电源，部分马达驱动器可提供此电源。 PCI-C154+、编码器与外部电源连接图如下。

注意必须连接外部电源之限流电阻 R，以保护 PCI-C154+ 电路，下表列出不同外部电源之建议电阻值。

Encoder Power (V)	External Resistor R
+5V	0Ω (None)
+12V	1.5kΩ
+24V	3.0kΩ

I_f = 8mA



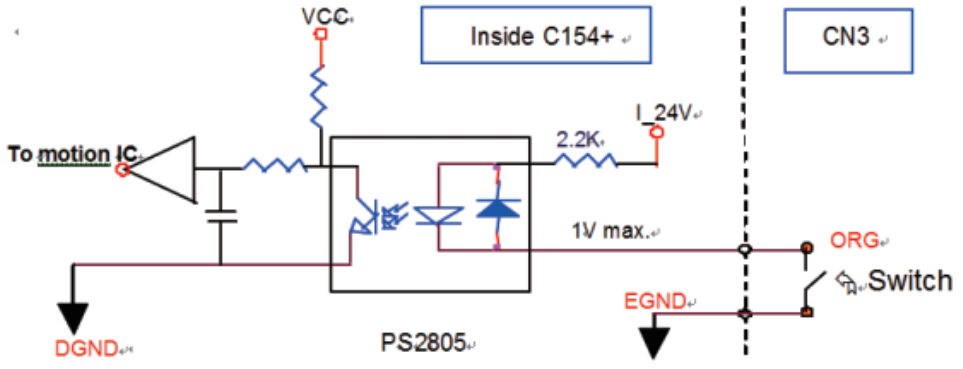
更多的编码器输入信号操作信息，请参阅 4.3.2 节

3.3 原点开关信号 (ORG)

原点输入信号 (ORG0-ORG3) 为机构原点的输入信号，其对应脚位如下：

CN3 Pin No.	Signal Name	Axis #
41	**ORG0**	0
47	**ORG1**	1
91	**ORG2**	2
97	**ORG3**	3

原点开关信号电路如下图。通常会使用限位开关来做为运动轴的原点，而驱动限位开关必须至少为 +24V@6mA。 PCI-C154+ 内部提供滤波电路，过滤高频噪声，来防止操作过程中产生的信号干扰。



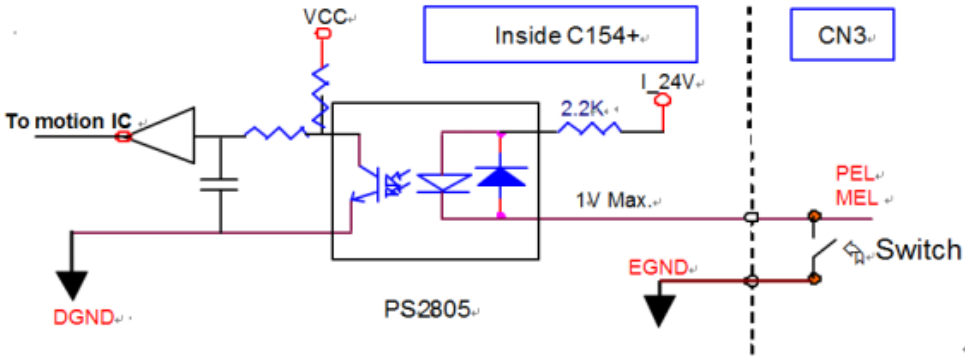
当运动控制器正在操作归零运动时，原点开关信号将用来限制脉冲控制命令信号 (OUT/DIR)，原点开关信号详细操作请参阅 4.4.1 节。

3.4 机械限位开关信号 (PEL/MEL)

各轴皆有一对机械限位开关信号 (PEL 与 MEL)。PEL 为正方向机械限位开关信号； MEL 为负方向机械限位开关信号，相关对应脚位如下表：

CN3 Pin No.	Signal Name	Axis #	CN3 Pin No.	Signal Name	Axis #
37	**PEL0**	0	**38**	**MEL0**	0
43	**PEL1**	1	**44**	**MEL1**	1
87	**PEL2**	2	**88**	**MEL2**	2
93	**PEL3**	3	**94**	**MEL3**	3

机械限位开关电路图如下。驱动外部限位开关必须至少为 +24V@8mA，无论是 A 接点（常开接点）或 B 接点（常闭接点），PCI-C154+ 皆可支持，详细设定请参考 C154_set_limit_logic 功能。

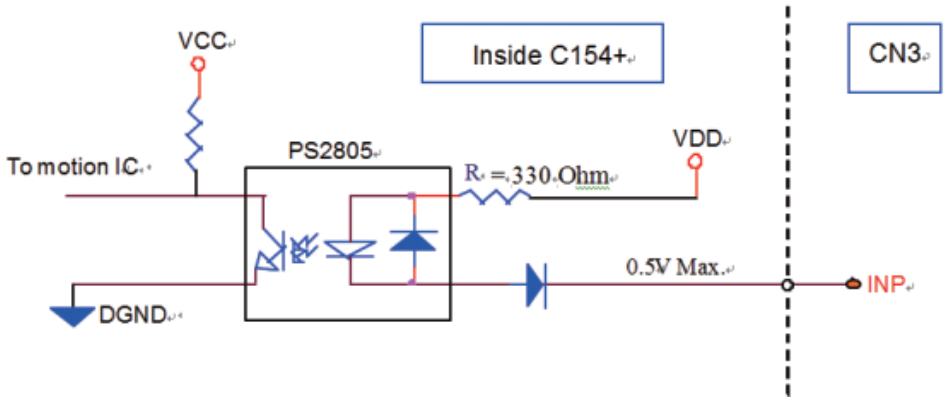


3.5 到位 / 零速度检出信号 (INP)

到位检出信号由伺服马达驱动器输出，用来表示位置偏差，若无位置偏差，伺服驱动器的位置讯息将为 0。其对应脚位如下表：

CN3 Pin No.	Signal Name	Axis #
10	**INP0**	0
28	**INP1**	1
60	**INP2**	2
78	**INP3**	3

到位检出信号电路图如下：



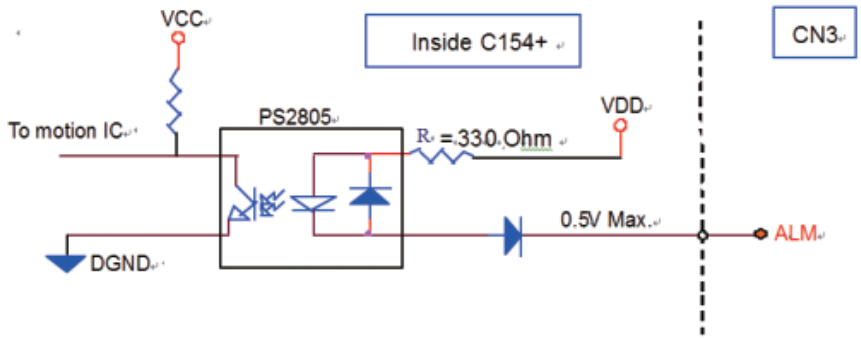
到位检出信号通常为伺服马达驱动器产生，且通常为开集极输出信号，外部电路必须提供至少 8mA 电流来驱动。

3.6 伺服警报信号 (ALM)

伺服警报信号用来指示伺服驱动器之警报状态。其对应脚位如下表：

CN3 Pin No.	Signal Name	Axis #
9	ALM0	0
27	ALM1	1
59	ALM2	2
77	ALM3	3

伺服警报输入电路如下图。伺服警报信号一般由伺服马达驱动器提供，且为开集极输出信号，外部电路必须提供至少 8mA 电流来驱动。



3.7 位置计数误差清除信号 (ERC)

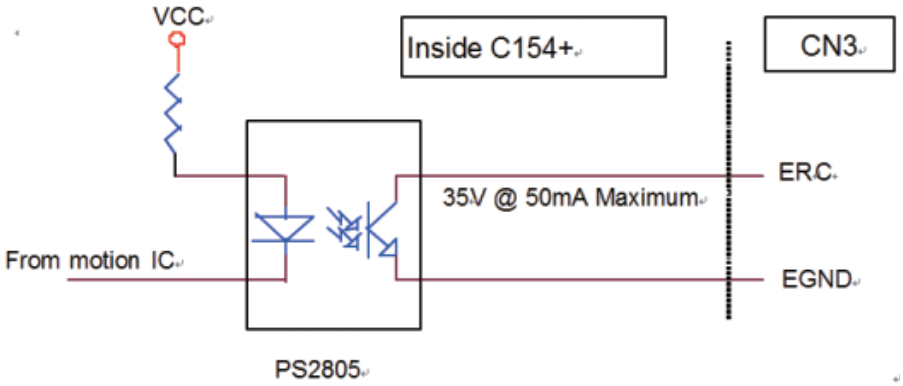
位置计数误差清除信号于下列 4 种状态下会触发：

1. 归零运动完成时
2. 机械限位开关被触发时
3. 警报信号使得 OUT/DIR 停止输出时
4. 软件端 (使用者) 发出紧急停止命令时

其对应脚位如下表：

CN3 Pin No.	Signal Name	Axis #
8	**ERC0**	0
26	**ERC1**	1
58	**ERC2**	2
76	**ERC3**	3

位置计数误差清除信号用来清除伺服马达驱动器的位置计数误差，其为开集极信号，最大驱动能力为 35V@50mA。

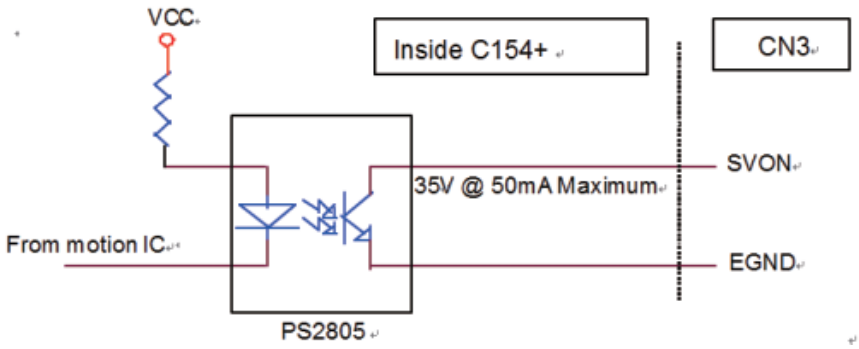


3.8 伺服激磁信号 (SVON)

伺服激磁信号可为伺服驱动器激磁信号，或为通用数字输出信号。其对应脚位如下表：

CN3 Pin No.	Signal Name	Axis #
7	SVON0	0
25	SVON1	1
57	SVON2	2
75	SVON3	3

伺服激磁信号电路图如下：

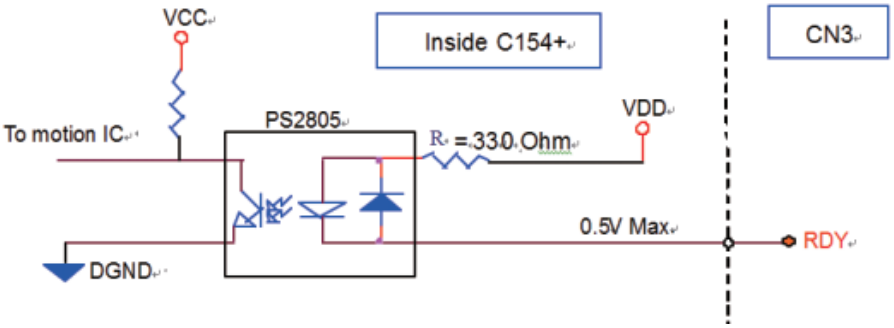


3.9 伺服预备信号 (RDY)

可为马达驱动器伺服预备输入信号，或通用输入信号，其对应脚位如下表：

CN3 Pin No.	Signal Name	Axis #
11	RDY0	0
29	RDY1	1
61	RDY2	2
79	RDY3	3

伺服预备输入信号电路图如下：



3.10 位置比较触发输出信号 (TRG)

PCI-C154+ 提供四组位置比较触发输出信号通道。位置比较触发功能详细请参阅 4.6 节

其对应脚位如下：

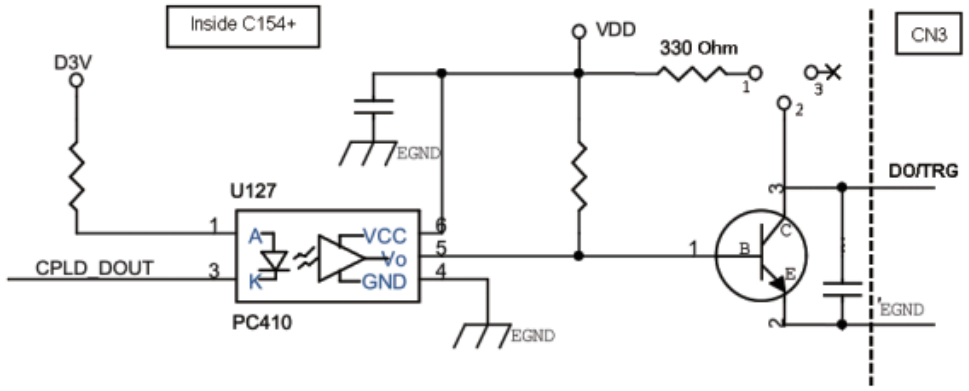
CN3 Pin No.	Signal Name	Axis #
40	**TRG_0**	0
46	**TRG_1**	1
90	**TRG_2**	2
96	**TRG_3**	3

多功能输出信号可经由jumper设定为使用内部5V电源或外部电源。将 jumper 脚位 1 与脚位 2 短路，或脚位 2 与脚位 3 短路，来选择输出模式。如下表：

Output Signal	For using internal +5V, close breaks between 1 and 2 of:	For using external power, close breaks between 2 and 3 of:
TRG_0	JP10	JP10
TRG_1	JP11	JP11
TRG_2	JP12	JP12
TRG_3	JP13	JP13

TRG 之预设设定为使用外部电源。

位置比较触发输出信号电路图如下：



注意

TRG 最大 Sink Current 为 50mA。
Power on 时， TRG 信号将会 active 。

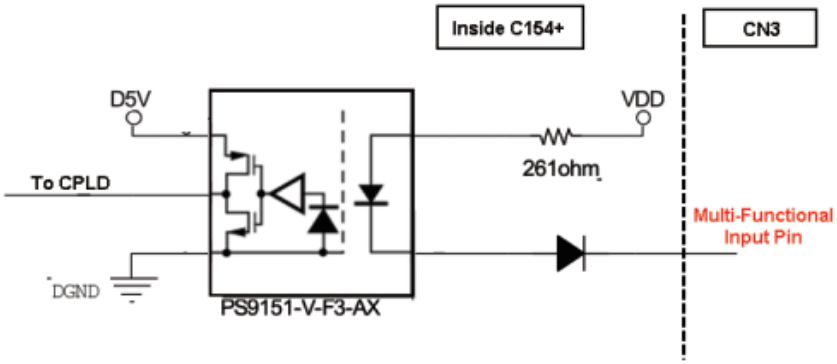
3.11 多功能输入信号 (DI/SD/CLR/EMG/LTC)

PCI-C154+ 提供 4 组多功能输入信号通道，每个输入信号可设定为通用数字信号 (DI)、降速信号 (SD)、计数器清除信号 (CLR)、紧急停止信号 (EMG)、位置撷取信号 (LTC)；详细设定请参阅附录 Q

多功能输入信号位于 CN3，其对应脚位如下：

CN3 Pin No.	Signal Name	Axis #
39	**DI/SD/CLR/EMG/LTC_0**	0
45	**DI/SD/CLR/EMG/LTC_1**	1
89	**DI/SD/CLR/EMG/LTC_2**	2
95	**DI/SD/CLR/EMG/LTC_3**	3

多功能输入信号接线图如下：



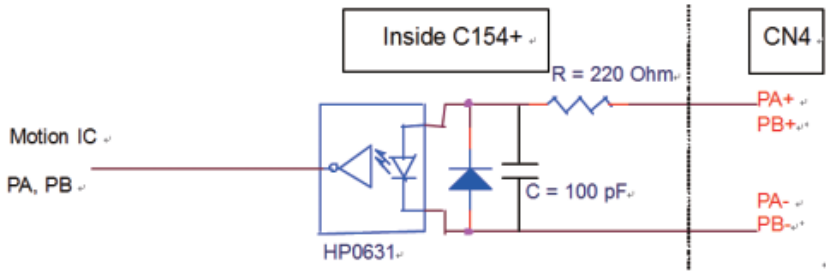
3.12 脉冲发生器输入信号 (PA/PB)

PCI-C154+ 透过 CN4 提供差分脉冲输入信号，如下表。脉冲发生器行为如同编码器， A-B 相位信号产生位置信息驱动马达。

CN4 Pin No.	Signal Name	Axis #	CN4 Pin No.	Signal Name	Axis #
2	**PA+**	0-3	**3**	**PA-**	0-3
4	**PB+**	0-3	**5**	**PB-**	0-3

脉冲发生器信号可使用于轴 0 到轴 3，使用者可透过 c154_disable_pulser_input 函数来设定各轴的启动或关闭。外部电路至少提供 8mA 电流来驱动。

差分脉冲输入接线图如下：



注意必须连接外部电源之限流电阻 R，以保护 PCI-C154+ 电路，下表列出不同外部电源之建议电阻值。

Encoder Power (V)	External Resistor R
+5V	0Ω (None)
+12V	1.5kΩ
+24V	3.0kΩ

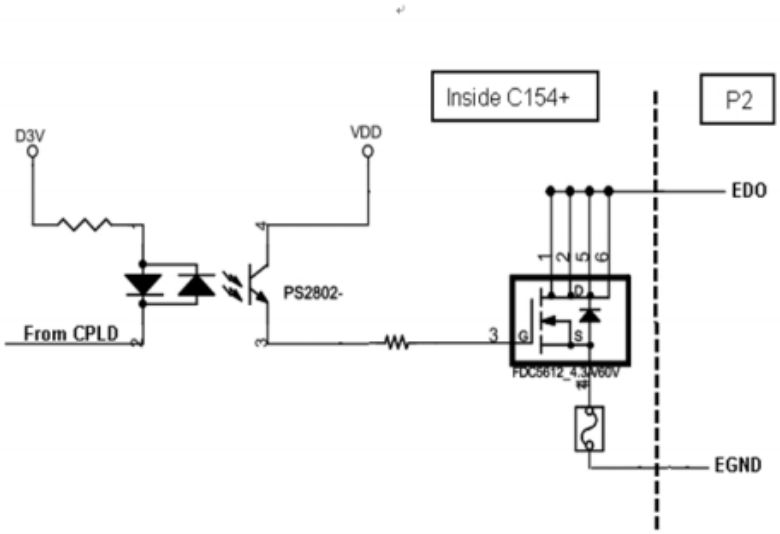
I_f = 8mA

3.13 扩展通用输出信号 (EDO)

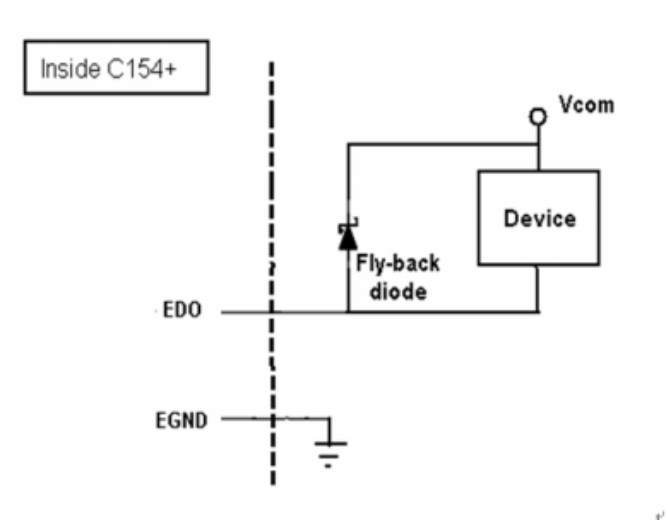
PCI-C154+ 透过 P2 提供 16 组扩展的通用输出信号，每个通道最大 sink current 能力为 250mA，对应脚位如下表：

P2 Pin No.	Signal Name	Axis #	P2 Pin No.	Signal Name	Axis #
23	**EDO0**	0	**24**	**EDO1**	1
25	**EDO2**	2	**26**	**EDO3**	3
29	**EDO4**	4	**30**	**EDO5**	5
31	**EDO6**	6	**32**	**EDO7**	7
33	**EDO8**	8	**34**	**EDO9**	9
37	**EDO10**	10	**38**	**EDO11**	11
39	**EDO12**	12	**40**	**EDO13**	13
41	**EDO14**	14	**42**	**EDO15**	15

扩展通用输出信号电路图如下：



若使用高电流电感式负载，建议电路图如下：



注意

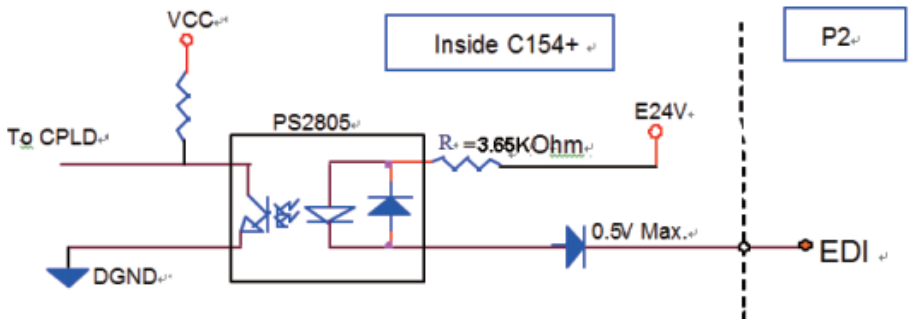
务必注意 P2 上 E X_GND 接线数量，建议如下 :
EDO 总电流负载为 0~1A: 连接至少一组 P2 上 E X_GND
EDO 总电流负载为 1~2A: 连接至少两组 P2 上 E X_GND
EDO 总电流负载为 2~3A: 连接至少三组 P2 上 E X_GND
EDO 总电流负载为 3~4A: 连接四组 P2 上 E X_GND

3.14 扩展通用输入信号 (EDI)

PCI-C154+ 透过 P2 提供 16 路扩展的通用输入信号，每个通道最大驱动能力为 6.5mA，对应脚位如下表：

P2 Pin No.	Signal Name	Axis #	P2 Pin No.	Signal Name	Axis #
3	**EDI0**	0	**4**	**EDI1**	1
5	**EDI2**	2	**6**	**EDI3**	3
7	**EDI4**	4	**8**	**EDI5**	5
11	**EDI6**	6	**12**	**EDI7**	7
13	**EDI8**	8	**14**	**EDI9**	9
15	**EDI0**	10	**16**	**EDI11**	11
19	**EDI2**	12	**20**	**EDI13**	13
21	**EDI4**	14	**22**	**EDI15**	15

扩展通用输入信号电路图如下：



4 操作原理介绍

本章将介绍如何操作本运动控制卡

4.1 运动控制器简介

马达的控制分成两种层面：马达控制与运动控制。马达控制一般与PWM、电源控制、闭回路控制、霍尔效应传感器与向量空间等相关；运动控制则与速度曲线规划、路径规划、多轴同步与坐标等相关。

4.1.1 电压运动控制接口 (Voltage Motion Control Interface)

早期马达运动控制使用电压信号作为控制命令，电压大小代表马达旋转速度，而电压变化量则为加减速度。此种运动控制接口也称做模拟运动控制，噪声通常为此种控制接口最大的问题，而且做位置控制时，通常需要反馈位置信号以及闭回路控制算法来达成，这会增加运动控制的复杂性，导致使用者不易上手。

4.1.2 脉冲运动控制接口 (Pulse Motion Control Interface)

第二种控制接口为脉冲信号，脉冲数代表马达旋转步数，而脉冲频率则为旋转速度，脉冲频率改变速度则为马达加减速度。此接口让用户更容易控制伺服或步进马达。

在多轴控制时，运动控制显得比马达控制重要，在工业应用上，可靠度是非常重要的参数，马达驱动器制造商提供好的效能，而运动控制器制造商提供强大且多样的软件，整合此两项产品将使整体机台更加完美。

4.1.3 网络型运动控制接口 (Network Motion Control Interface)

网络型运动控制接口为新型的控制接口，马达驱动器与运动控制器间的命令并非模拟信号也非脉冲信号，其命令为网络封包，封包内容包括位置信息以及马达信息。此种控制接口更为可靠，因为其命令为数字且封包形式，但由于运动控制必须为实时性 (real time)，因此网络型接口必须具备 1ms 的实时性，一般商业网络无法达成此特性，必须为特殊的网络型态，例如 Mitsubishi SSCNET。网络型接口亦可利用光纤来增加通讯能力。

4.1.4 实时性运动控制软件 (Software Real-Time Motion Control Kernel)

对于运动控制核心而言，可分成三种型态：DSP、控制芯片以及实时性软件。

运动控制系统必须有绝对的实时性控制与运算能力，若无此能力，马达将无法平顺地运转。许多设备制造商会使用 PC 的运算能力来达成，使用拥有反馈信号以及电压或脉冲输出的板卡即可；但此方法会花费许多软件设计功夫，且需实时性软件来达成实时性效能，这将增加系统复杂性，不过对于高阶运动控制设计者而言，此方式较为弹性。

4.1.5 运动控制 DSP(DSP Motion Control Kernel)

DSP 运动控制器解决了实时性问题，DSP 为微处理器，它自行拥有OS可处理所有运动控制运算，因此系统无需使用实时性软件；虽然 DSP 无实时性问题，但其运算能力依然不比 PC 的 CPU，且对使用者而言， DSP 运动控制器的软件接口也不易上手。

4.1.6 运动控制芯片 (ASIC Motion Control Kernel))

运动控制芯片同时解决了实时性软件与 DSP 的问题，运动控制芯片无实时性问题，且用户仅需对控制芯片设定参数，运动控制即可轻易实现。

4.1.7 运动控制型态比较表

	Software	ASIC	DSP
Price	Fair	Cheap	Expensive
Functionality	Highest	Low	Normal
Maintenance	Hard	Easy	Fair

	Analog	Pulses	Network
Price	High	Low	Normal
Signal Quality	Fair	Good	Reliable
Maintenance	Hard	Easy	Easy

4.1.8 PCI-C154+ 运动控制器型态

PCI-C154+ 为脉冲模式运动控制卡，由三部分组成，运动控制芯片、PCI 板卡以及运动控制软件函数库；运动控制芯片可于 Window XP/7/8.1 系统中透过运动控制软件函数库来操作，使用者可轻易地透过凌华科技所提供的运动控制软件函数库控制马达。所有速度参数将由函数库计算。

例如，使用 T 型速度曲线进行单轴点对点运动，用户仅需填写目标位置、速度以及加速时间，马达即会依照曲线运行，而此将不会消耗 CPU 资源，所有控制脉冲信号将由芯片产生；而精确的目标位置则由马达驱动器的闭回路控制，并非控制器，控制器只负责依照速度曲线传送正确的脉冲数，因此程序、机构与电机工程师将可轻松地排除问题。

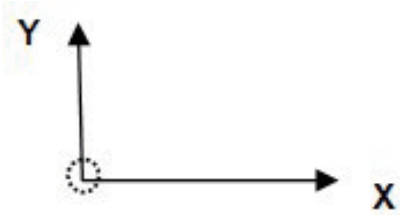
4.2 运动控制模式

运动控制，即控制各轴马达依据特定速度曲线、轨迹以及特定同步条件下进行运动，此节将描述 PCI-C154+ 的运动控制模式。

4.2.1 坐标系统 (Coordinate System)

PCI-C154+ 采用笛卡儿坐标系，长度单位为 pulse，而实际物理量取决于机构设计以及马达分辨率；例如，马达架设于滚珠螺杆上，而螺杆移动 10mm 需马达转动 10,000pulses，则每 pulse 等于 10mm/10,000p=1um。

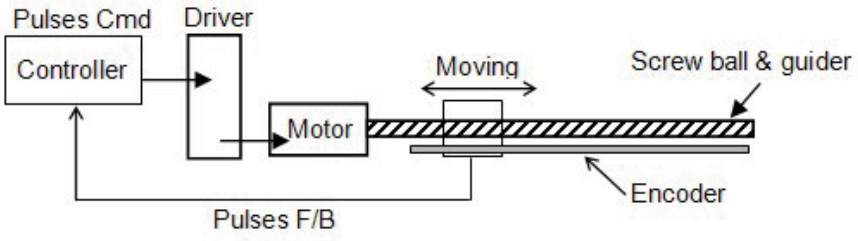
因此若想移动 15mm，则设定命令位置为 15,000pulses。而欲移动 15.0001mm，控制器将会储存小于 1puse 的残余值，且加入下一个命令中。



运动控制器只能传送累加脉冲信号给驱动器，亦即只能传送相对位置命令给驱动器，不过用户仍然可输入绝对位置命令，而运动控制器将会依据目前位置与目标位置来计算出相对位置命令；例如，目前位置为 1000，目标位置为 9000，用户可输入绝对位置 9000，而

运动控制器将会计算出相对位置 8000，接着传送 8000 脉冲信号给驱动器，使马达移动至位置 9000。

有时用户会另外使用线性标尺或外部编码器来确认机构位置，于此状况要如何建立坐标系统呢 ？ 例如外部编码器 10,000pulses 代表 1mm，而控制器命令 1,000pulses 使马达移动 1mm，这代表若使用者送出命令1,000pulses，将会从外部编码器取得反馈10,000pulses；若要使马达移动至绝对位置 10,000，而目前由外部编码器所读回的反馈位置为 3,500，则应再输出位置命令(10000-3500)/(10000/1000)=650pulse。使用者只须设定 ”move ratio”，控制器将会自动计算以上结果。 Move ratio = 反馈分辨率 / 命令分辨率 (feedback resolution/command resolution)。

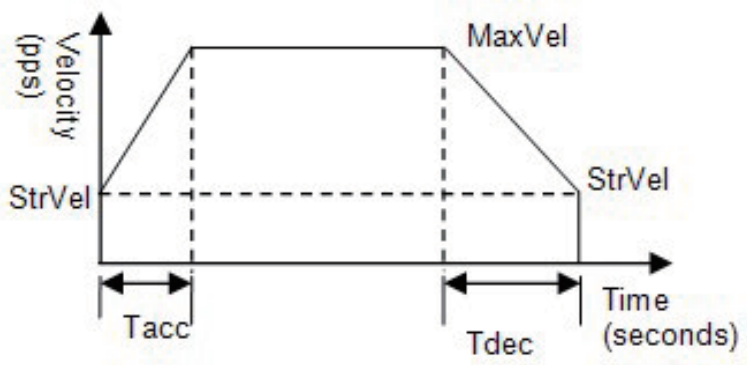


4.2.2 绝对 / 相对运动 (Absolute/Relative Position Move)

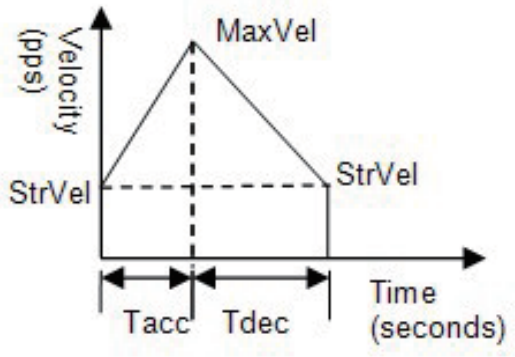
PCI-C154+ 提供两种运动命令供用户做位置定位运动：绝对运动命令为用户给定一目标位置，控制器将会控制马达从目前位置移动至目标位置；而相对运动命令为使用者给定一相对距离，控制器将会控制马达从目前位置移动相对距离至某位置。于此两种运动，用户皆可指定速度曲线。

4.2.3 T 型速度曲线 (T-Curve Speed Profile)

T 型速度曲线系指，加减速区段为一阶线性速度曲线 (加速度为固定值)。其曲线如下：



速度曲线下的面积代表运动距离，有时曲线会呈现三角形，因为移动距离太小，以致无法达到所设定运动参数，当此状况发生时，控制器将降低最大速度，且保持加减速度，来符合用户所设定移动距离，其曲线图将会如下：

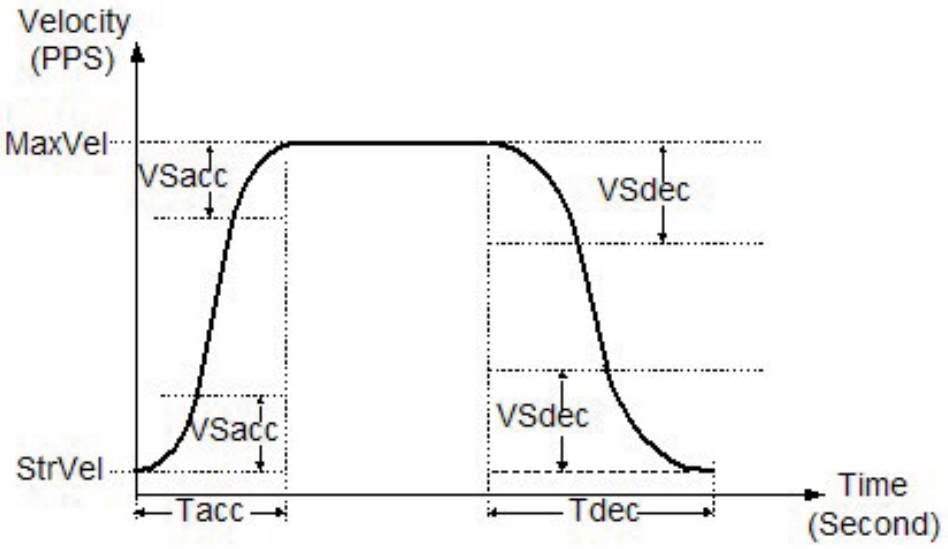


此运动曲线可应用于单轴或多轴插补运动，以及速度或位置运动模式。

4.2.4 S 型速度曲线 (S-Curve Speed Profile)

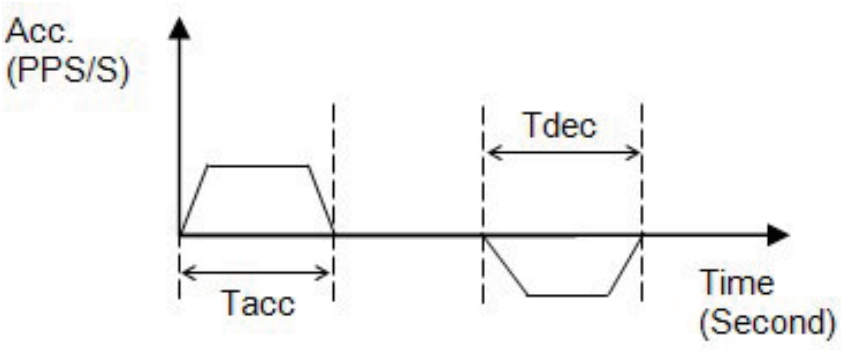
S 型速度曲线系指，加减速区段为二阶速度曲线。此曲线可降低马达开始或停止运动时所产生的振动，为了加快加速和减速时间，于 S 曲线上半部与下半部间插入线性部分，以维持最大加速率。

其曲线与参数如下：



- Tacc：加速度时间 (S)
- Tdec：减速度时间 (S)
- StrVel：起始速度 (PPS)
- MaxVel：最大速度 (PPS)
- VSacc：加速端的 S 曲线参数 (PPS)，加速端 S 曲线内的速度差值
- VSdec：减速端的 S 曲线参数 (PPS)，减速端 S 曲线内的速度差值

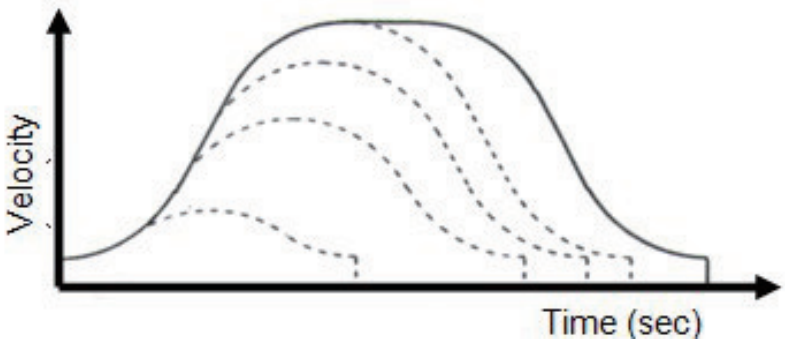
当 VSacc or VSdec=0，加减速端将为纯 S 曲线，没有直线线段。下图为加速度时序图。



如同 T 型曲线，当移动距离不足时，控制器会自动修正最大速度以维持运动的顺畅，如下图。

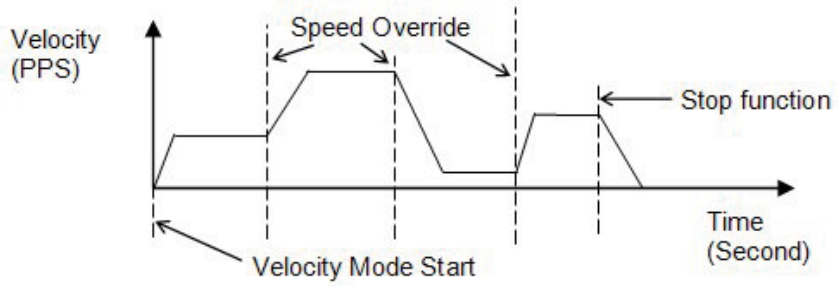
其中 MaxVel, Tacc, Tdec, VSacc, VSdec 数值将会降低，而 StrVel, Acc, jerk 则保持不变。

此运动曲线可应用于单轴或多轴插补运动，以及速度或位置运动模式。



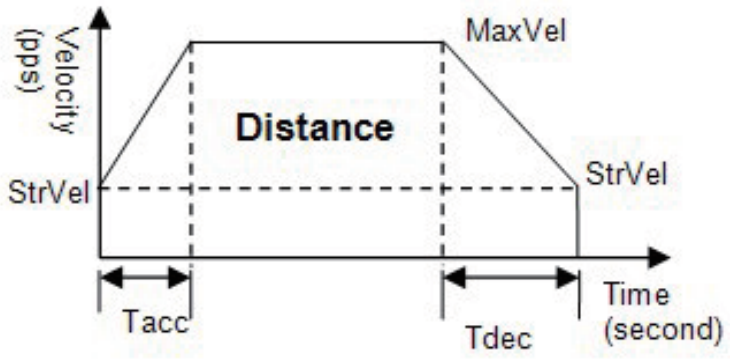
4.2.5 速度运动模式 (Velocity Mode)

此运动模式系指，当指令下达后该运动轴会依照指定的运动曲线运行，运动将会持续下去直到下达停止运动的命令。其运动速度会依照 T 型或 S 型曲线加速至最大速度，且持续维持为最大速度，直到下达更新速度命令或停止命令。注意，其更新速度，不可与原本运动速度方向相反，此类运动模式曲线如下：



4.2.6 单轴运动模式 (One Axis Position Mode)

此模式系指，控制器输出特定脉冲数量，以达到用户所设定位置或距离，其最小距离为 1 pulse ；然而 PCI-C154+ 提供浮点数功能，以符合特定物理距离，小于 1 pulse 的距离将会储存于软件内，且累加至下一个运动命令中。控制器提供三种运动曲线来达成定位运动， 1.T 型曲线， 2.S 型曲线， 3. 混和型 S 型曲线；下图为 T 型曲线范例图：

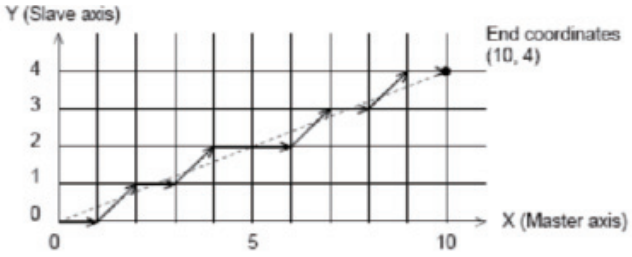


距离为速度曲线下的面积。

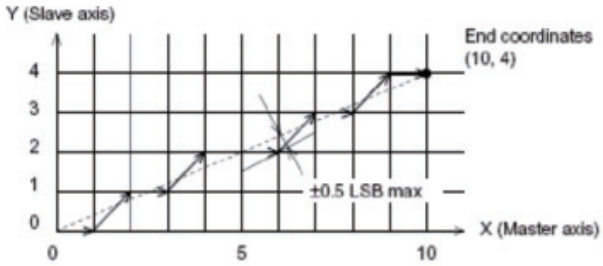
4.2.7 多轴直线插补运动模式 (Linear Interpolation Mode)

此运动模式系指，多轴同步运动，且同时到达目标位置，轴与轴间的速度比例为固定值。

下图，假设从 (0,0) 运动至 (10,4)



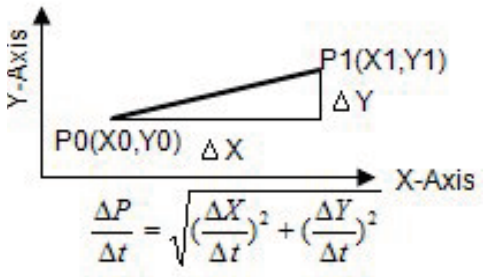
X 或 Y 轴的脉冲输出路径与真实线性路径有 1/2pulse 差异，如下图:



若要停止插补运动，只需对群组中的第一运动轴下达停止指令即可。

如下图，直线插补指移动 XY 位置从 P0 至 P1。两轴同时开始以及同时停止，且动作路径为直线。

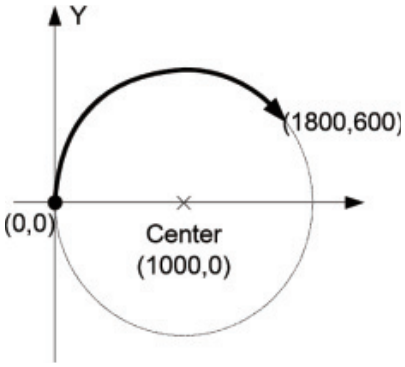
X 轴与 Y 轴速度比例为 (ΔX: ΔY)，其合成速度为：



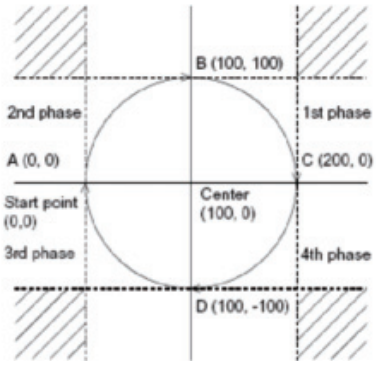
若欲使用直线插补功能，需先设定合成速度的起始速度 StrVel 以及最大速度 MaxVel

4.2.8 两轴圆弧插补运动模式 (Circular Interpolation Mode)

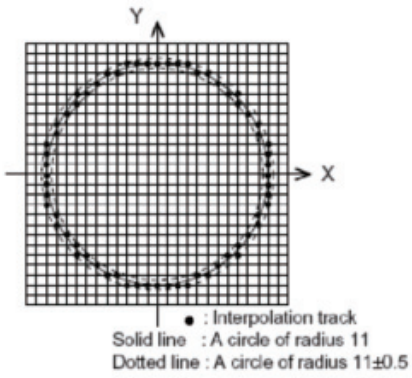
此运动模式系指，XY 轴同时开始运动且同时结束运动，而运动路径为圆弧曲线，MaxVel 为切线速度。请注意，若圆弧曲线的终点不正确，则运动轴将会不断地做绕圆运动。



控制器将会控制运动轴至用户所设定的目标位置，即便目标位置并非精准的座落在圆弧上；但是若目标位置在下图斜线内，则运动轴将不停旋转。



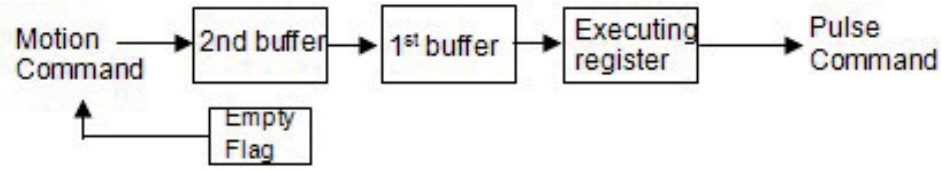
圆弧插补的命令精确度如下图所示，精确度为 ±1/2 pulse 内。



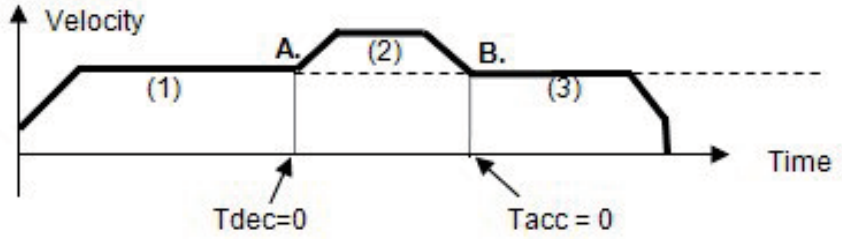
4.2.9 连续运动 (Continuous Motion)

因控制器内部有三个命令缓存器，所以用户可于不中断前一个运动的状态下，下达新的运动命令。

当第一个命令正在执行时，使用者可下达第二个命令于第一缓存器，第三个命令于第二缓存器。当第一个命令结束时，控制器将自动把第二个命令移至执行缓存器，第三个命令移至第一缓存器，此时用户将可下达第四个命令于第二缓存器。一般来说，使用者有足够的时间，于执行缓存器结束前下达命令于第二缓存器，因此可达成不间断地连续运动，架构图如下：

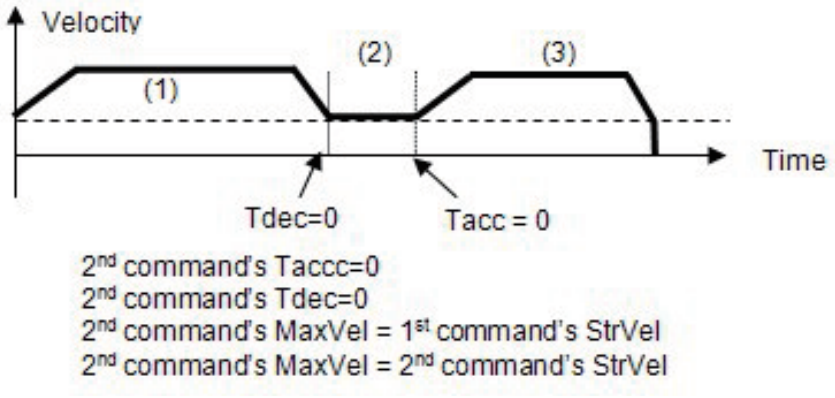


为了让速度连续，速度命令必须正确地设定，如下图，于此连续运动中有三个运动命令，而第二个命令速度高于其他两个，其速度相关设定如下图所示：



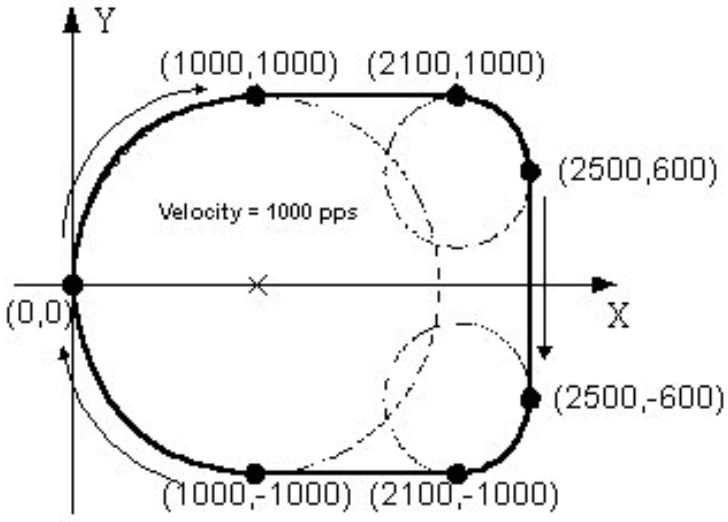
1st command's Tdec=0
2nd command's StrVel = 1st command's MaxVel
3rd command's Tacc=0
3rd command's MaxVel=2nd command's StrVel

若第二个命令速度低于其他两个，则其设定如下图所示：



2nd command's Taccc=0
2nd command's Tdec=0
2nd command's MaxVel = 1st command's StrVel
2nd command's MaxVel = 2nd command's StrVel

连续圆弧插补运动也为相同概念。



若启动 INP 确认信号，将导致命令与命令间产生些微延迟。若欲消除此延迟，平滑连续运动，请关闭 INP 确认信号。

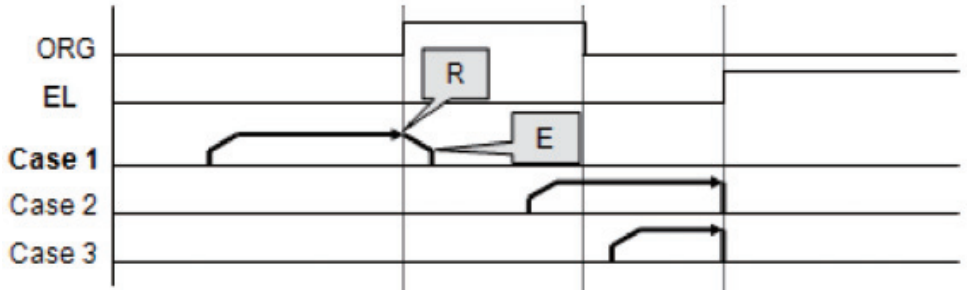
4.2.10 归零运动 (Home Return)

归零运动系指回到坐标轴上的零点位置。一般于机台启动时，必须先找到机台的零点位置，有时用户会利用 ORG、 EZ 或 EL 脚位当作零点位置。

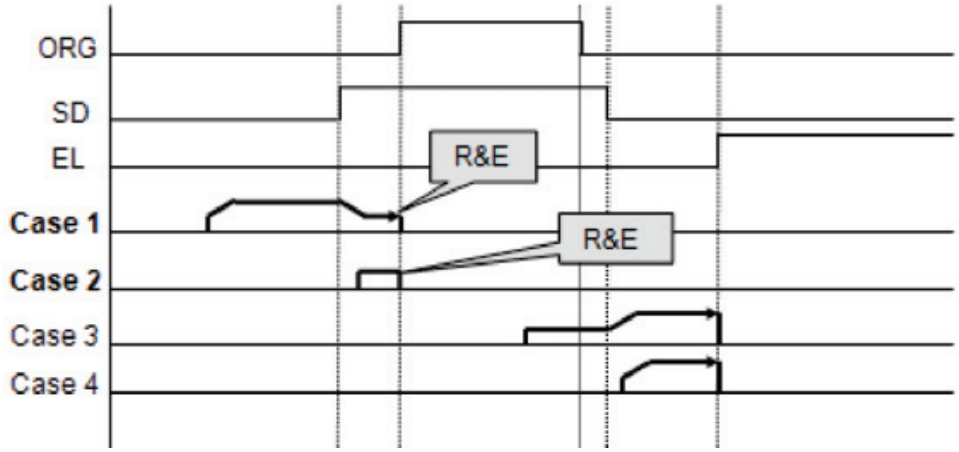
PCI-C154+ 提供多种归零运动，这些运动皆由控制芯片完成，不会增加 CPU 负载。当归零运动结束，达到设定条件 (例如 ORG 上升边缘触发) 时，目标计数器将会被重置为零；有时计数器重置后，控制器依然会输出脉冲信号使机台减速，当马达停止时，即使归零运动结束，但计数器并非为零，此时计数器的数值为相对机台零点位置的数值。

以下图标为各种归零运动模式，R 为计数器 (命令与位置计数器) 重置， E 为 ERC 信号输出， FA 为预先设定之低速的归零速度。

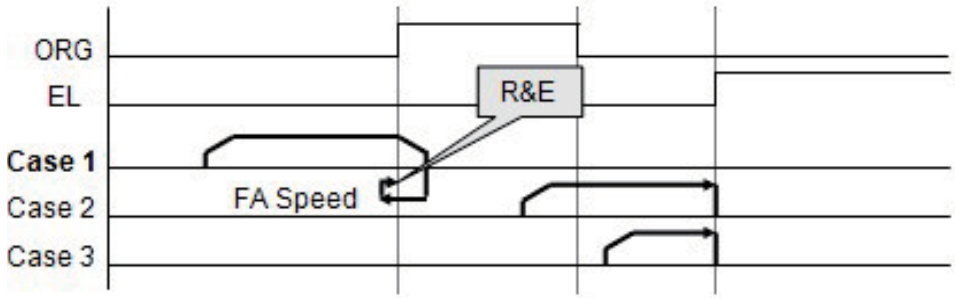
- **Home mode=0: (ORG Turn ON then reset counter)**
  - ▶ When SD is not installed



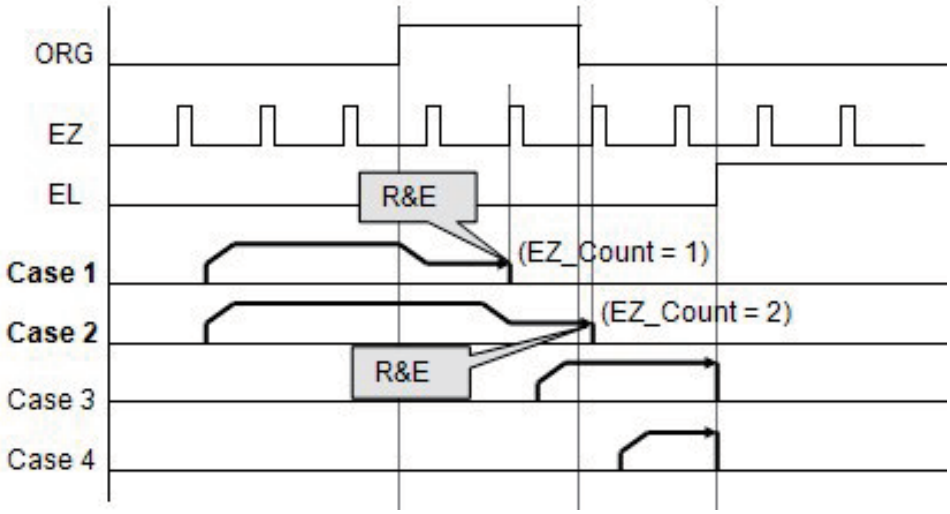
  - ▶ When SD is installed and SD is not latched



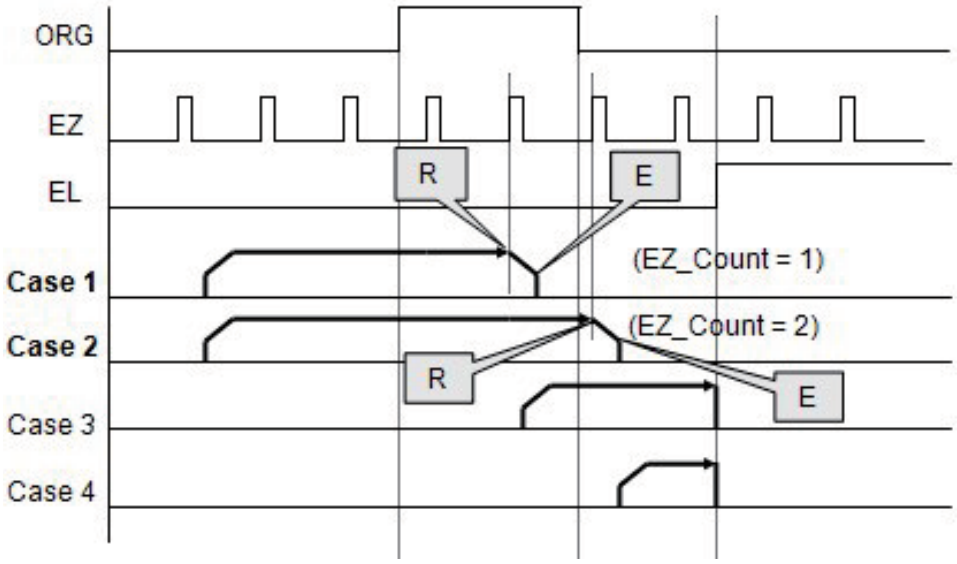
- **Home mode=1: (Twice ORG turn ON then reset counter)**



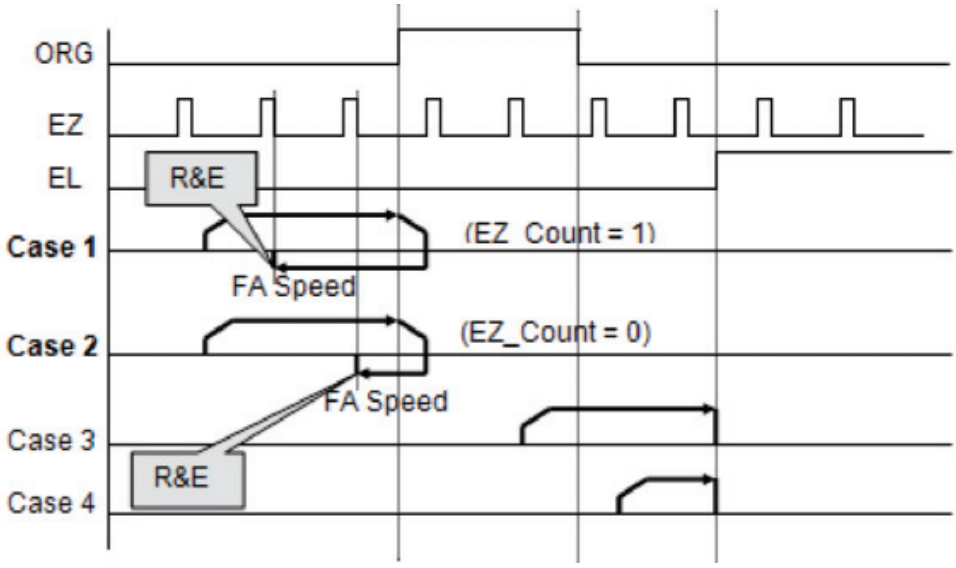
- **Home mode=2: (ORG ON then Slow down to count EZ numbers and reset counter)**



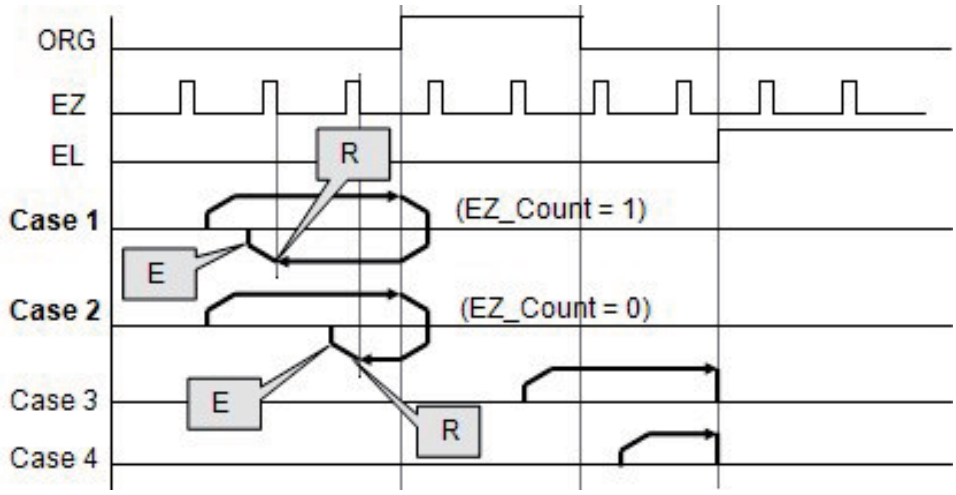
- **Home mode=3: (ORG ON then count EZ numbers and reset counter)**



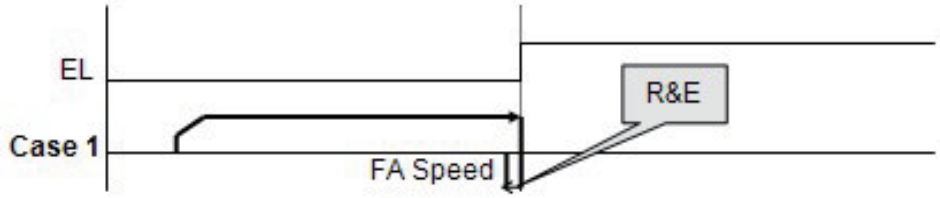
- **Home mode=4: (ORG On then reverse to count EZ number and reset counter)**



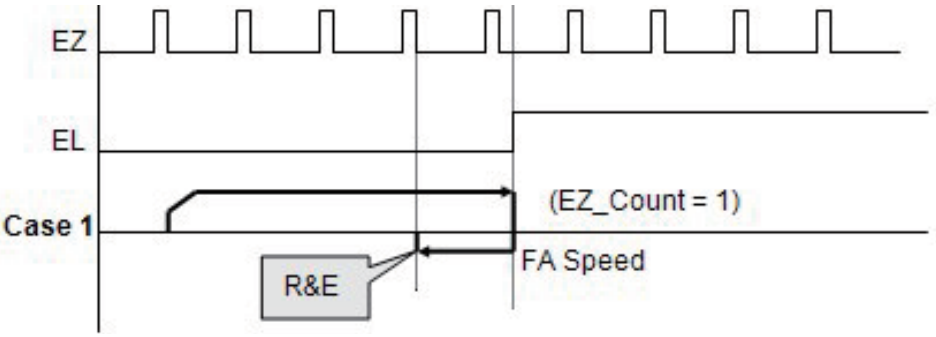
- **Home mode=5: (ORG On then reverse to count EZ number and reset counter, not using FA Speed)**



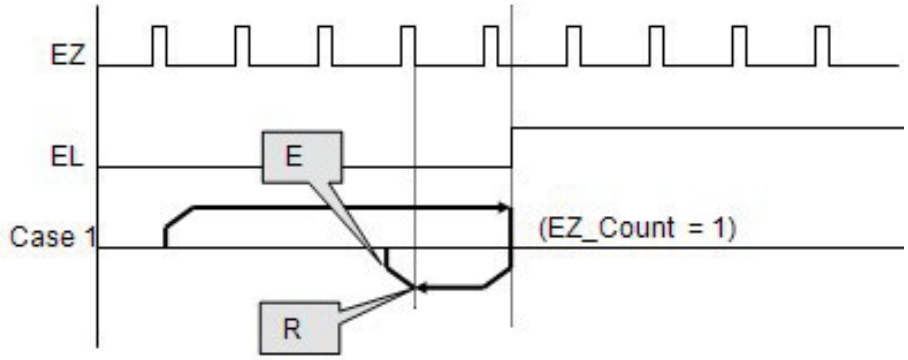
- **Home mode=6: (EL On then reverse to leave EL and reset counter)**



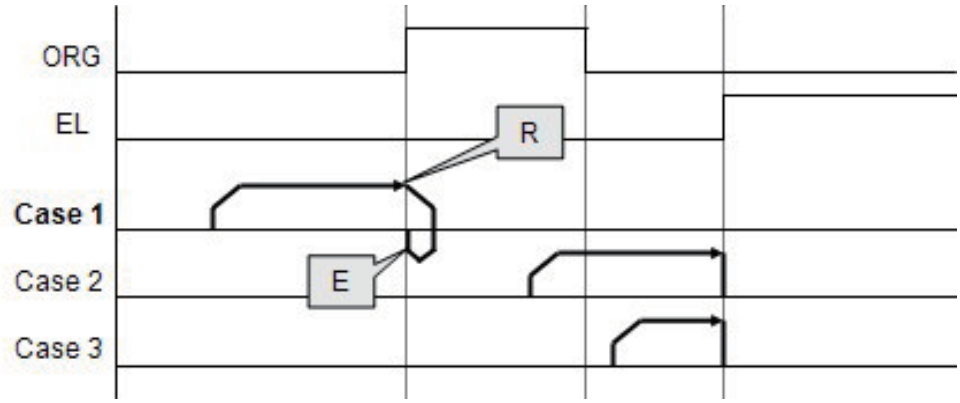
- **Home mode=7: (EL On then reverse to count EZ number and reset counter)**



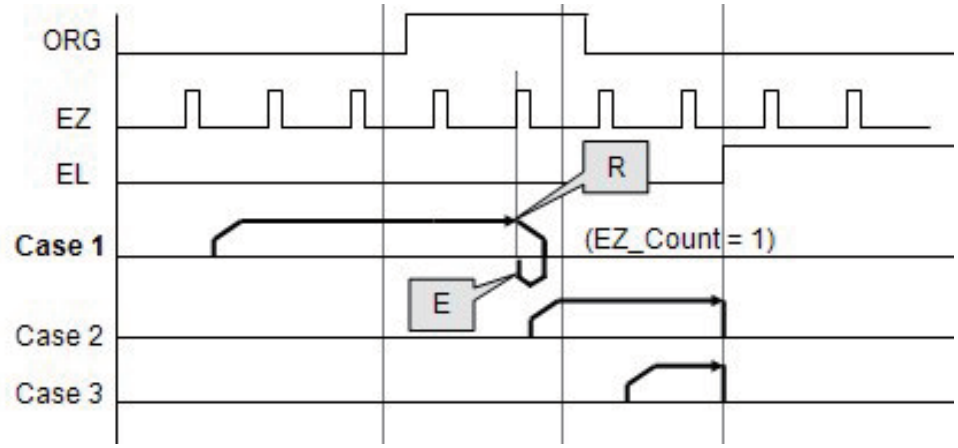
- **Home mode=8: (EL On then reverse to count EZ number and reset counter, not using FA Speed)**



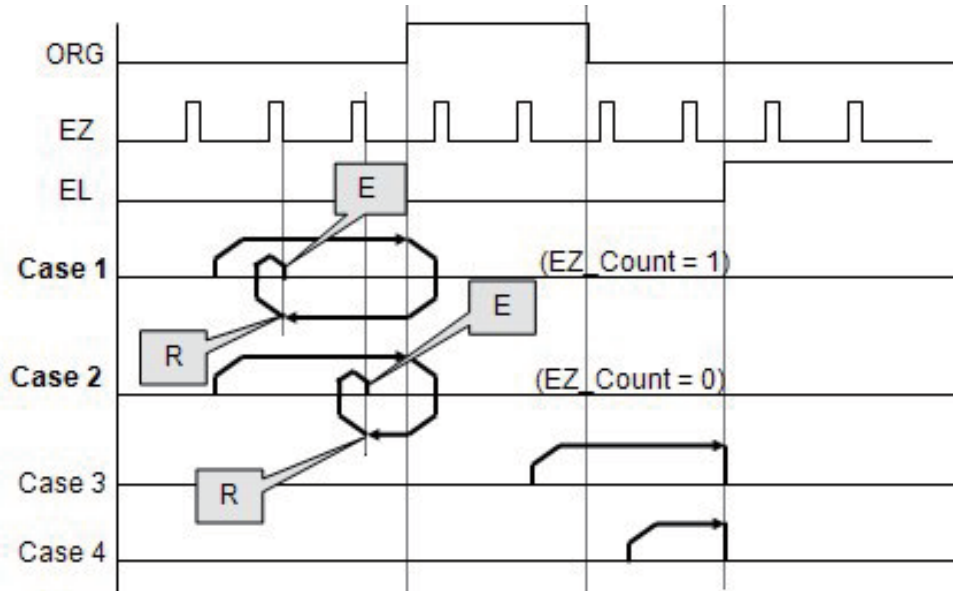
- **Home mode=9: (ORG On then reverse to zero position, an extension from mode 0)**



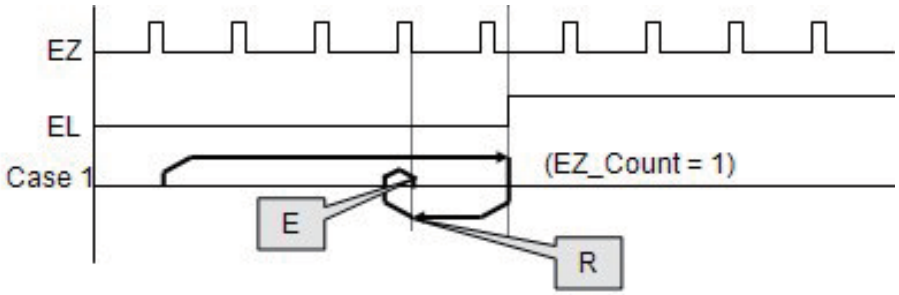
- **Home mode=10: (ORG On then counter EZ and reverse to zero position, an extension from mode 3)**



- **Home mode=11: (ORG On then reverse to counter EZ and reverse to zero position, an extension from mode 5)**

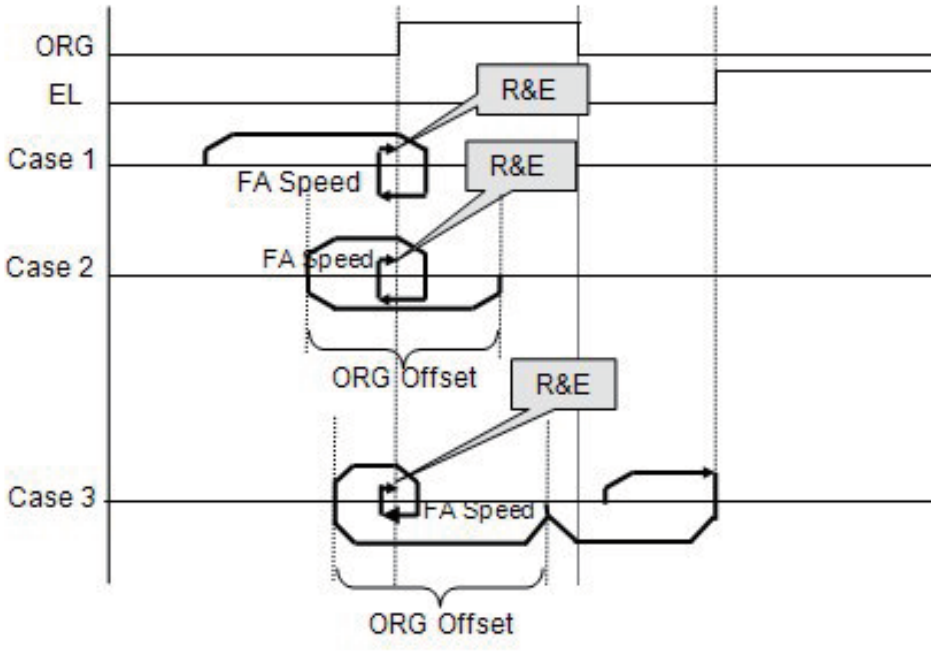


- **Home mode=12: (EL On then reverse to count EZ number and reverse to zero position, an extension from mode 8)**



4.2.11 零点位置搜寻功能 (Home Search)

在归零运动模式上增加搜寻零点位置功能，下图范例为 Home Mode=1 增加搜寻零点功能。 ORG offset 不可为零，建议为 ORG area 长度两倍。

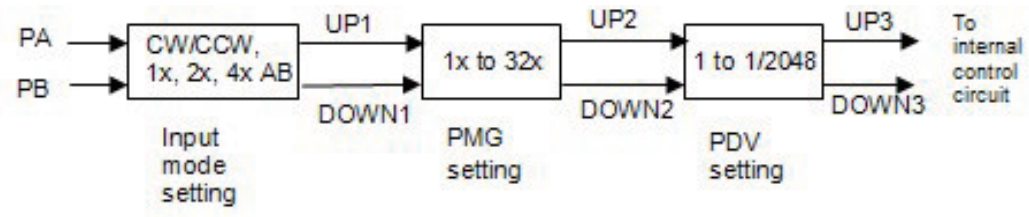


4.2.12 手动脉冲发生器功能 (Manual Pulser)

手动脉冲发生器输出脉冲信号至控制器，接着再输出至脉冲输出信号脚位，脉冲信号经过控制器，可除频或倍频后再输出。

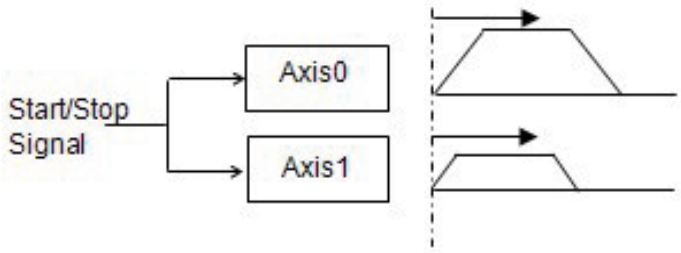
控制器可接收两种脉冲信号模式，CW/CCW 与 AB 相位模式，若为 AB 相位模式，则可为 1、 2 或 4 倍。

下图为脉冲比例方块图：



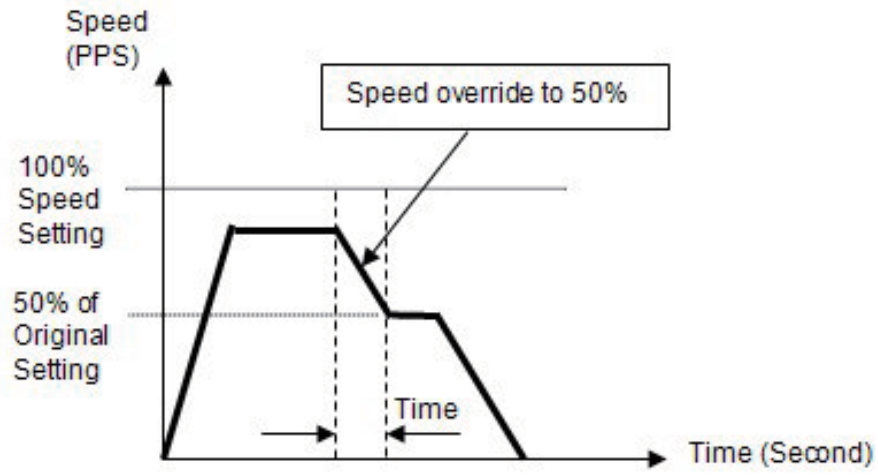
4.2.13 同步启动功能 (Simultaneous Start)

藉由内部信号，可同时启动多个运动轴，可使用软件开始功能当作同步信号。



4.2.14 速度重设功能 (Speed Override)

在运动过程中，用户可任意改变速度命令，此参数设定方式为下达原始设定速度的百分比例；于运动开始前，用户可先设定一速度做为 100% 的速度，接着于运动过程中下达百分比例数值，即可更新速度。若用户没设定 100% 的速度值，则默认 100% 速度为最后运动命令之最大速度；此功能可应用于任何运动模式，若速度曲线为 S 曲线，则速度重设时将会是纯 S 曲线 (pure s-curve)；若为 T 曲线，则速度重设时为 T 曲线。



4.3 连接马达驱动器接口

马达驱动器上有多种 I/O 脚位供外部运动控制器使用，于此 PCI-C154+ 提供数个专用 I/O 脚位连接，这些脚位可分为两类，一类为脉冲 I/O 信号接口，包括脉冲命令输出以及编码器输入接口；另一类为数字 I/O 信号接口，包括伺服激磁信号 (SVON)、伺服警报信号 (ALARM)、到位检出信号 (INP)、伺服预备信号 (RDY)、伺服警报重置信号、紧急停止信号 (EMG)。

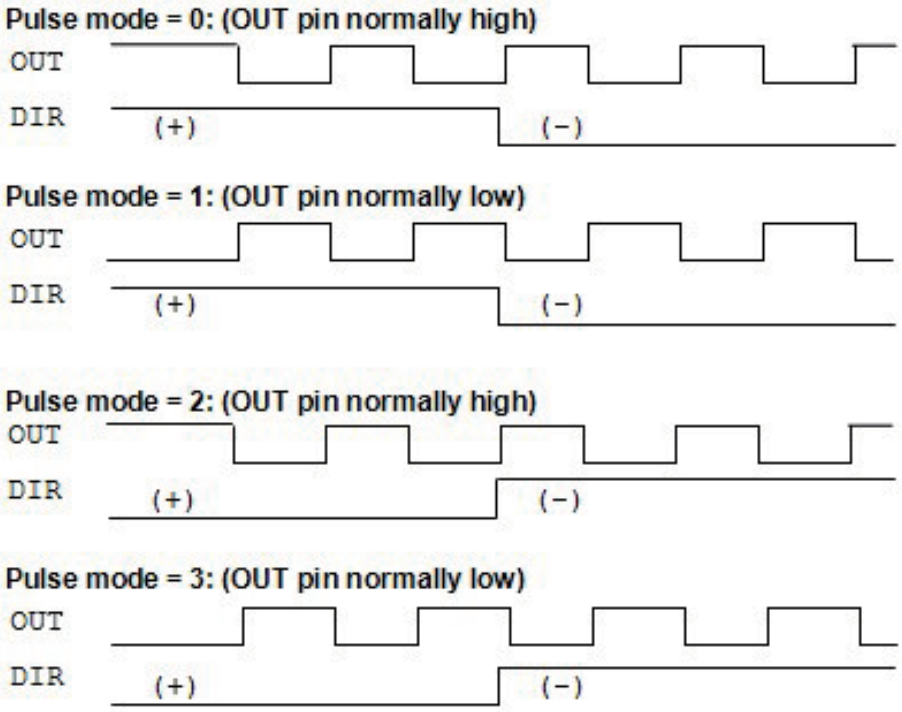
4.3.1 脉冲命令输出接口 (OUT/DIR)

控制器下达脉冲命令透过驱动器来控制伺服 / 步进马达，请先设定驱动器为位置模式以接收位置脉冲命令。脉冲命令由两对信号组成，脚位定义为 OUT 与 DIR，分别为一对差分信号，可设定为 (1) 单脉冲输出模式 (OUT/DIR)， (2) 双脉冲输出模式 (CW/CCW)，模式设定必须与驱动器相符，对应设定如下表：

Mode	Output of OUT pin	Output of DIR pin
Dual pulse output (CW/CCW)	Pulse signal in plus (or CW) direction	Pulse signal in minus (or CCW) direction
Single pulse output (OUT/DIR)	Pulse signal	Direction signal (level)

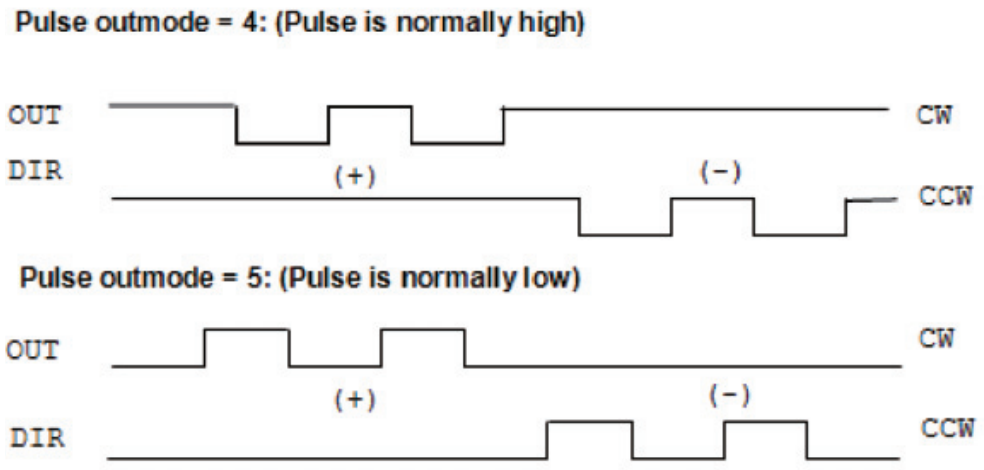
- **单脉冲输出模式 (OUT/DIR)**

OUT 脚位输出脉冲信号，输出的脉冲数代表距离（ 单位：pulse)，而输出的频率为速度 (单位：pps) ； DIR 脚位为方向信号。输出波形如下图：



- **双脉冲输出模式 (CW/CCW)**

此模式， OUT 与 DIR 脚位分别代表 CW(顺时针) 与 CCW(逆时针) 脉冲信号，输出的脉冲数代表距离 (单位：pulse)，而输出的频率为速度 (单位: pps) ；由 CW 输出脉冲信号使马达往正方向移动，反之 CCW 输出脉冲信号则使马达往负方向移动。输出波形如下图:



脉冲控制命令信号为 28 位命令计数器。

4.3.2 编码器输入接口 (EA/EB)

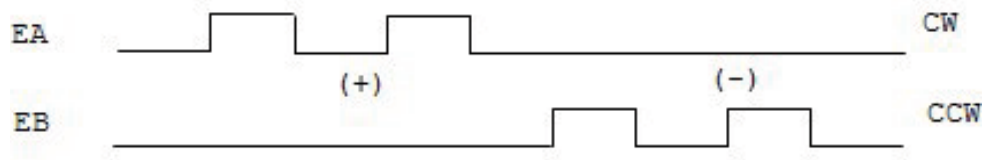
PCI-C154+ 于每轴皆提供 28 位反馈位置计数器，其脉冲信号来源为 EA 与 EB 脚位，这两信号脚位皆为差分信号，可接受两种信号模式， (1)CW/CCW 模式， (2)AB 相位模式，其中 AB 相位模式可为 1、 2 与 4 倍频， 4 倍 AB 相位模式为最常见的编码器输入模式。

例如，每转一圈，编码器将输出 2000pulses，则当 AB 相位为 4 倍时，将会从反馈位置计数器上读到 8000pulses。

若用户不使用编码器，则计数器反馈来源必须设定为脉冲命令信号，否则计数器数值将永远为零。若设定为脉冲命令信号，则反馈位置数值为计数内部脉冲命令信号数目。

下图为两种模式脉冲反馈信号：

- **正负脉冲输入模式 (CW/CCW)**

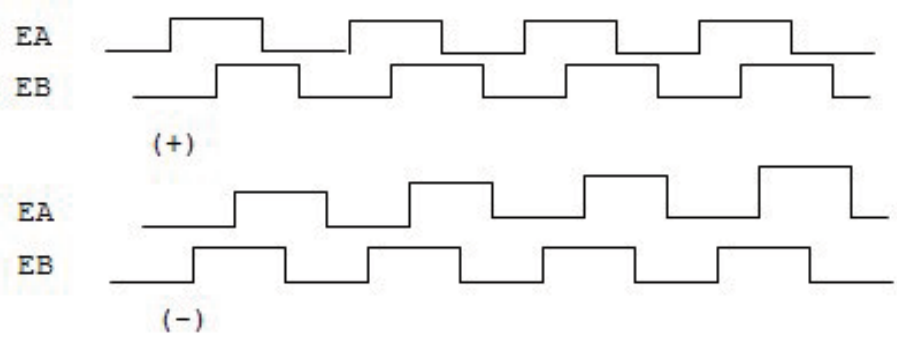


此模式波形与**双脉冲输出模式**相同。EA 信号为向上计数，EB 信号为向下计数。

- **90° 相位差信号输入模式 (AB 相位)**

于此模式，EA 信号将会领先或落后 EB 信号 90° 相位，当 EA 信号领先相位时，计数器会向上计数，反之 EA 信号落后时，计数器则向下计数。

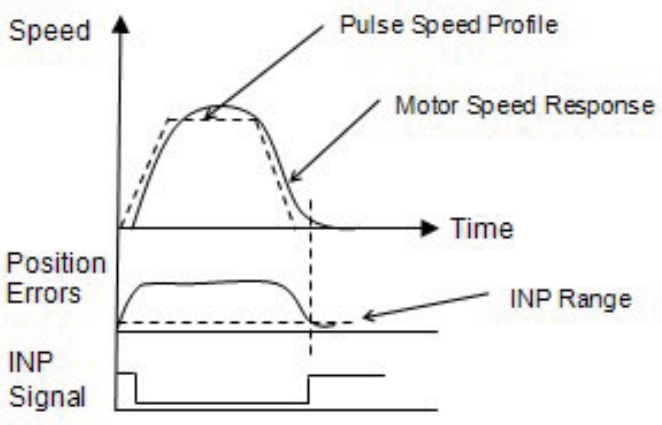
其波形如下图：



EZ 信号为编码器参考原点信号，可用来定义机构的零点位置，操作前请先正确设定信号逻辑。

4.3.3 到位检出信号 (INP)

到位检出信号为马达驱动器的输出信号，目的为通知控制器马达已经移动至某位置的误差范围内，此预设的误差范围称做 in-position value。当控制器下达位置命令，运动忙碌状态 (busy status) 将会持续 (TRUE)，直到到位检出信号触发。用户可关闭检测到位检出信号，于此模式，当所有位置命令下达后，忙碌状态将结束 (FALSE)。



4.3.4 伺服警报信号 (ALARM)

伺服警报信号为马达驱动器的输出信号，目的为通知控制器，马达或驱动器发生错误；当控制器收到此信号，脉冲控制命令将回停止发送，且 ALM 信号将为 ON。造成警报的原因可能为马达速度过大、电流过大或负载过大等，详细内容请查询马达驱动器手册。

伺服警报信号逻辑必须正确设定，必须与马达驱动器相符，否则 ALM 将会永远为 ON，脉冲控制命令将无法送出。

4.3.5 误差清除信号 (ERC)

误差清除信号为控制器的输出信号，目的为通知驱动器清除误差计数器，此误差为命令脉冲与反馈脉冲间的差异。马达驱动器依据此误差计数器当作控制基准，当误差数值较大时，将会下达较大的速度命令；若误差为零时，代表速度命令将为零。

误差清除信号将于以下四种情况自动由控制器发出至马达驱动器：

1. 归零运动完成时
2. 机械限位开关被触发时
3. 警报信号使得 OUT/DIR 停止输出时
4. 发出紧急停止命令时

4.3.6 伺服激磁信号 (SVON)

伺服激磁信号为控制器的输出信号，于连接器上为 SVON 脚位。此信号用来切换驱动器控制状态，当此信号 ON 时，驱动器开始进入控制状态。请注意当控制轴为垂直方向时，且关闭伺服激磁信号 (OFF)，控制轴将不受控制，机构会因此掉落；伺服警报信号 (ALM) 或紧急停止信号 (EMG) 为 ON 时也会发生相同状况。

4.3.7 伺服预备信号 (RDY)

伺服预备信号为控制器的输入信号，控制器将不会对此信号做任何对应动作；用户可连接马达驱动器的RDY信号来检查驱动器是否为预备状态；用户也可把此信号当作通用输入信号。此信号将不会影响控制器。

4.3.8 伺服警报重置信号

当伺服驱动器发生错误时，驱动器将会发出伺服警报信号 (ALM)，可藉由关掉电源来重置警报信号，但通常操作过程中不可关掉电源，因此伺服驱动器会提供一个信号脚位来重置警报信号；本控制器在各轴皆提供伺服警报重置信号，来重置伺服警报状态。

4.4 机械开关接口

PCI-C154+ 提供了多个机械开关输入接口，包括原点开关 (ORG)、限位开关 (±EL)、降速信号 (SD)、紧急停止开关 (EMG)、计数器清除开关 (CLR) ；这些开关相关功能皆由运动控制芯片直接处理，不经由上层软件，因此其反应时间相当迅速。

4.4.1 原点开关信号 (ORG)

本控制器在每个控制轴皆提供原点开关信号，用来表示原点位置；做归零运动前，必须先正确设定此信号的逻辑。

4.4.2 限位开关信号 (PEL/MEL)

通常限位开关架设于控制轴的两个端点，用于防止撞机；马达移动过程中，若触碰限位开关时，运动控制器将停止发送脉冲控制命令信号，且同时输出 ERC 信号。

4.4.3 降速信号 (SD)

降速信号用于强制控制轴降速至其开始运动时的起始速度，以避免机构设备高速运动至机械极限；降速信号在正负方向皆有作用。

4.4.4 计数器清除开关信号 (CLR)

此输入信号将会重置运动控制器内部的计数器。

4.4.5 紧急停止开关信号 (EMG)

当此信号输入时，运动控制器将会立即停止所有控制轴的运动，以防止设备损坏；一般此信号会连接于设备上之紧急停止开关上。在此建议使用常闭接点 (B 接点) 方式接线。

4.5 计数器 (Counters)

每个控制轴备有四种计数器：

- 命令位置计数器：计数命令脉冲
- 反馈位置计数器：计数反馈脉冲
- 位置误差计数器：计数命令与反馈脉冲误差
- 通用计数器：计数对象可为命令位置、反馈位置、手动脉冲输入或控制器参考时钟之半周期
- 目标位置纪录器：纪录软件端所设定的目标位置

4.5.1 命令位置计数器 (Command Position Counter)

命令位置计数器为可正反计数的 28 位计数器，用于记录当下的命令位置。

PCI-C154+ 为开回路控制卡，马达驱动器接收控制卡的命令后，直接驱动马达。若马达没有运作，用户可以检查命令位置计数器是否有更新数据，若有，代表脉冲控制命令有传送至驱动器，则问题可能发生在驱动器上；请试着检查驱动器上是否有接收到脉冲控制命令。

命令位置计数器单位为pulse。而此计数器可藉以下三种方式进行重置：1. 藉由计数器清除信号 (CLR)，2. 归零运动完成时，3. 藉由软件设定。

4.5.2 反馈位置计数器 (Feedback Position Counter)

反馈位置计数器为可正反计数的 28 位计数器，其信号来源为编码器输入信号 (EA/EB)，显示马达位置；倘若无外部编码器输入信号，此计数器亦可设定其信号来源为命令位置。

命令位置与反馈位置并非永远为相同比例，使用者必须依实际状况设定，令其比例为 1:1。

4.5.3 位置误差计数器 (Command and Feedback Error Counter)

位置误差计数器用于纪录命令位置与反馈位置间的误差，由命令位置减去反馈位置。

若命令位置与反馈位置比例并非 1:1，则此位置误差无实质意义。

此计数器为 16 位计数器。

4.5.4 通用计数器 (General Purpose Counter)

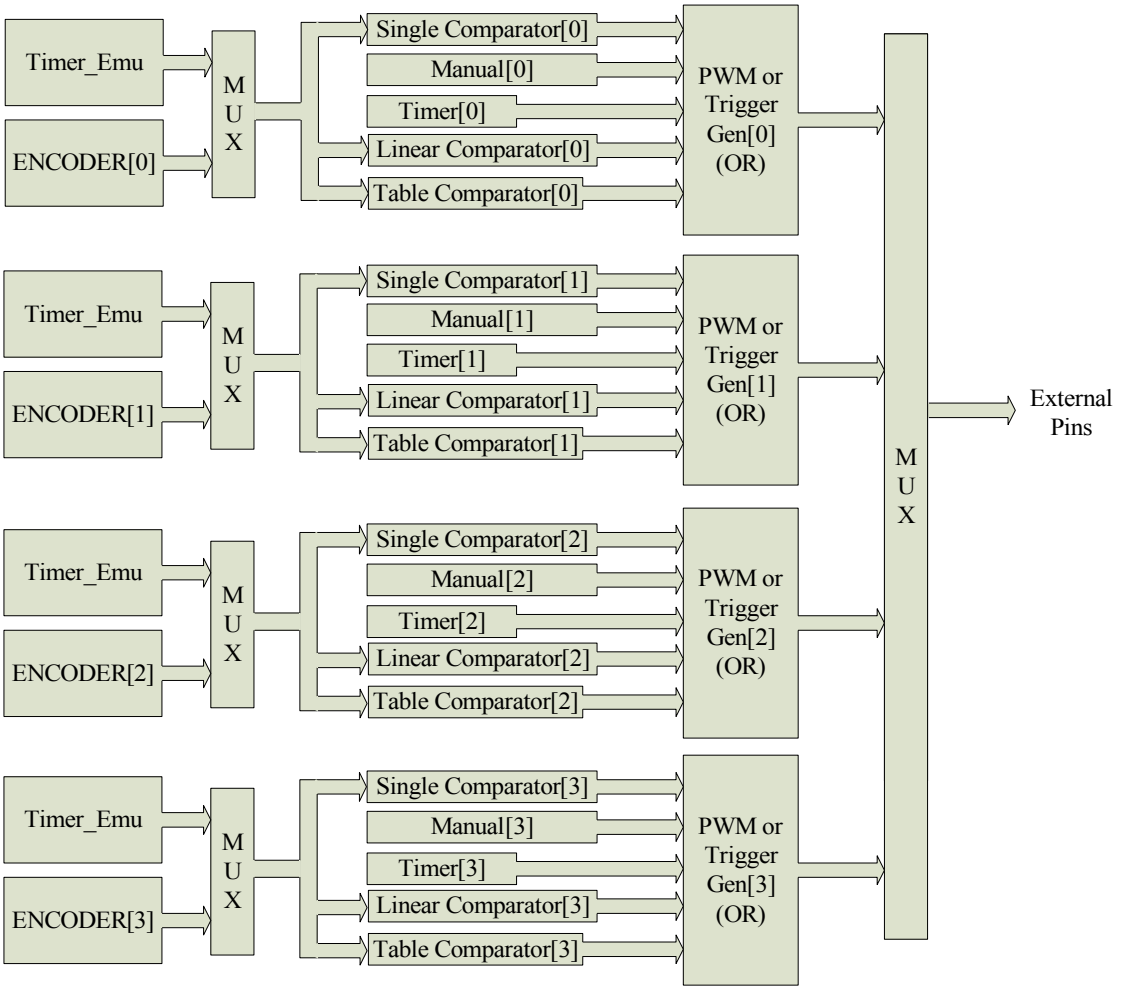
通用计数信号来源可为：命令位置、反馈位置、手动脉冲输入或控制器参考时钟之半周期

1. 命令位置 – 如同命令位置计数器
2. 反馈位置 – 如同反馈位置计数器
3. 手动脉冲输入 – 默认值
4. 控制器参考时钟之半周期 – 计数频率约 9.8MHz

4.6 高速位置比较触发 (High Speed Position Compare Trigger)

总共分为时间计数器 (Timer)、表格比较 (Table Compare)、线性比较 (Linear Compare)、单点比较 (Single Compare)、手动 (Manual) 触发 五种触发方式以逻辑 OR 的方式将所有致能的触发源送入脉冲宽度调变模块 (PWM) 转为触发信号发出。

而表格比较、线性比较、单点比较的比较源可为，编码器或时间记时器。



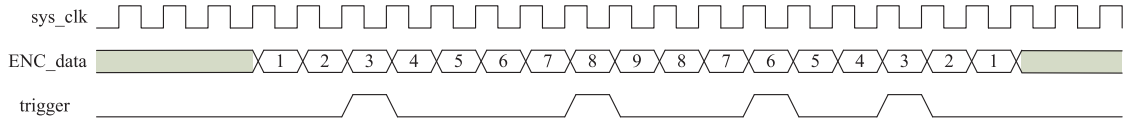
4.6.1 时间计数器触发 (Timer Trigger)

此功能为一个可以设定以时间为基础的周期性触发源，在设定的计数器值倒数至 0 的时候会送出一个触发信号并且重新开始倒数。

此功能可设定暂停的功能。

4.6.2 表格比较触发 (Table Compare Trigger)

这项触发内部是一个深度 1026 的 FIFO，每次输入位置数值就会往 FIFO 推一级，而每次比对到的时候 FIFO 会自动将当前的值推出 FIFO 换下一笔来比对。比方说设定触发长度是 1， FIFO 依序填入 2,7,7,4，编码器产生的位置读数依序是 1,2,3,4,5,6,7,8,9,8,7,6,5,4,3,2,1，则会得到下列波型



此功能也可以透过软件手动跳跃，以上图的例子来说，在编码器位置第一次到达 5 的时候，透过软件，跳过下次判定则编码器位置为 8 的时候就不会有触发产生。

4.6.3 线性比较触发 (Linear Compare Trigger)

此功能为一个可以产生以编码器位置为基础的周期性触发信号，可以设定的项目包含起始位置，间距 (有方向性 , signed) 以及重复次数，例如输入了起始位置 2，间距 8，重复次数 2 则卡片会在编码器位置 2、 10 以及 18 处依序发出 3 个触发信号。产生的数列方向由间距的正负决定。

此功能亦可以透过软件手动跳跃 .

此功能可设定在剩余比对次数为 0 的时候自动停止，使用者只需要再次设定使能缓存器，则此功能将会以一样的参数重新产生触发信号。

4.6.4 单点比较触发 (Single Compare Trigger)

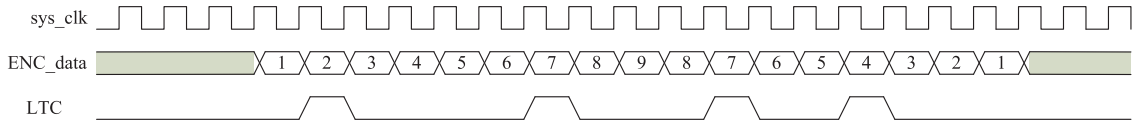
此功能设定一个固定的值，当编码器位置等于这个值的时候会送出一个单一处发信号。

4.6.5 手动触发 (Manual Trigger)

此功能为透过软件发送的一个手动触发信号。

4.7 高速位置撷取功能 (Position Latch)

这项功能为一个 FIFO 用来依照外部位置撷取输入信号 (LTC) 或手动触发信号来储存编码器的位置信息，用户可以读取这个 FIFO 依序读出已经储存的值。以下面波型为例使用者读取这个撷取的位置 4 次则依序得到 2,7,7,4。



4.8 其他运动控制功能

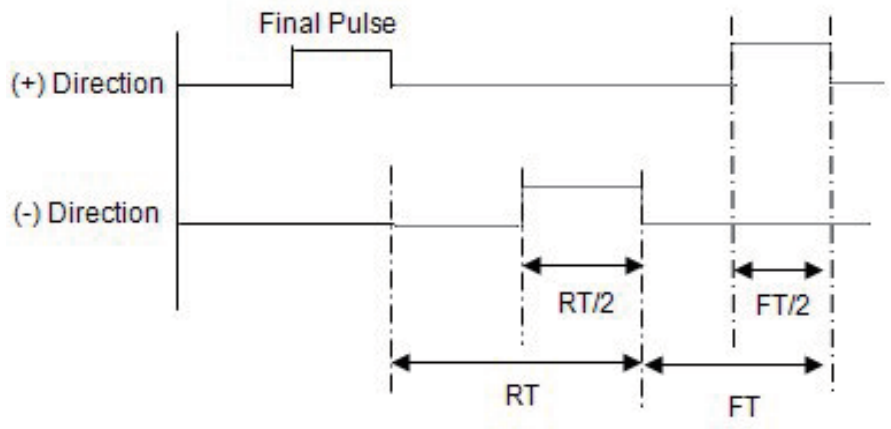
以下章节将介绍控制器提供的其他运动控制功能

4.8.1 背隙补偿 与 滑移修正 (Backlash Compensation & Slip Correction)

控制器提供背隙补偿与滑移修正功能，补偿 (修正) 的脉冲速度将为 FA(预先设定之速度)。每当运动方向改变时，将会实行背隙补偿；而运动命令下达时，无论方向为何，滑移修正功能皆会实行。修正脉冲数将由软件设定。

4.8.2 震动抑制功能 (Vibration Restriction)

此功能实现方式为，运动命令结束时，控制器会于反方向输出一脉冲信号，紧接着于正方向再输出一脉冲信号，其时序图如下：(RT 为反方向时间， FT 为正方向时间)



4.9 中断控制 (Interrupt Control)

运动控制器可发送中断信号给主系统，用户可利用 c154_int_control() 来启动或停止中断服务。

于 PCI-C154+ 上有四种中断信号源：1. 运动中断源， 2. 错误中断源，3. 通用输出入中断源，4. 位置比对与撷取中断源；运动、通用输出入与位置比对 / 撷取中断源可任意选择启动，错误中断源则不可。可藉由 c154_set_motion_int_factor() 选择运动中断源，位设定如下表：

- **运动中断源位设定**

Bit	Description
0	Normally Stop
1	Next command in buffer starts
2	Command pre-register 2 is empty and allow new command to write
3	Reserved
4	Acceleration Start
5	Acceleration End
6	Deceleration Start
7	Deceleration End
8	+Soft limit
9	-Soft limit
10	Reserved
11	Reserved
12	Reserved
13	Counter is reset by CLR input
14	Reserved
15	Counter is latched by ORG Input
16	SD input turns on
17	Reserved
18	Reserved
19	CSTA input or c154_start_move_all() turns on
20-31	Reserved

表 4-1：运动中断源位设定

错误中断源不可选择，但可由 c154_wait_error_interrupt() 回传值得知错误状况

- **错误中断回传值**

Bit	Description
0	+Soft Limit is ON and axis is stopped
1	-Soft Limit is ON and axis is stopped
2	Comparator 3 is ON and axis is stopped
3	General Comparator or comparator 4 is ON and axis is stopped
4	Trigger Comparator or comparator 5 is ON and axis is stopped
5	+End Limit is on and axis is stopped
6	-End Limit is on and axis is stopped
7	ALM is happened and axis is stop
8	CSTP is ON or c154_stop_move_all is on and axis is stopped
9	CEMG is on and axis is stopped
10	SD input is on and axis is slowed down to stop
11	Reserved
12	Interpolation operation error and stop
13	axis is stopped from other axis's error stop
14	Pulse input buffer overflow and stop
15	Interpolation counter overflow
16	Encoder input signal error but axis is not stopped
17	Pulse input signal error but axis is not stopped
18~31	Reserved

表 4-2：错误中断回传值

- **位置比对与撷取中断源位设定**

Bit	Description
0	Latch fifo if full state
1	Latch fifo is in empty state
2	Latch fifo is above level(Be equal to or greater than level)
3	Latch fifo is in full overflow
4	Linear comparator is finished
5	PWM signal overlaps
6	Comparator fifo is in full state
7	Comparator fifo is in empty state
8	Comarator fifo is below level(Be equal to or less than level)
9	Comparator fifo is in full overflow
10	EA noise interrupt occurs
11	EB noise interrupt occurs
12	EZ noise interrupt occurs
13	LTC noise interrupt occurs
14-31	Reserved

表 4-3：位置比对与撷取中断源位设定

使用中断的步骤如下：

1. ***c154_int_control(CARD_ID, Enable=1/Disable=0);***
2. 设定中断源为事件或通用输出入中断
3. ***c154_set_motion_int_facor(AXIS0,0x01);*** // 轴0正常停止
4. ***c154_wait_motion_interrupt(AXIS0,0x01,1000)*** // 用 1000ms 等待轴 0 正常停止中断发生
5. ***I16 ErrNo=c154_wait_error_interrupt(AXIS0,2000);*** // 用 2000ms 等待错误中断发生

4.10 多卡操作

使用者可将多张 PCI-C154+ 卡组装于同一系统内，系统 BIOS 将会自动配置内存与 IRQ 通道。

当将多张 PCI-C154+ 组装于同一系统时，请注意需于板卡上调整卡片编号开关 (SW1)，而控制轴编号将由卡片编号决定；例如使用三张卡片于同一系统时，其卡片编号与控制轴编号对应关系如下图：

	Axis No.	X	Y	Z	U
Card ID					
0		0	1	2	3
2		8	9	10	11
3		12	13	14	15

请注意，若同一系统内有卡片编号相同时，程序将会不正常运作。

5 MotionCreatorPro

当硬件安装好后 (参照第 2 章及第 3 章)，系统运作前，必须正确设定所有运动控制卡以及系统。本章将导引如何设定控制系统、以及手动测试 XP/7/8.1PCI-C154+ 卡，来验证如何正确地操作。MotionCreatorPro 提供简单且强大的设定、测试以及侦错功能，并透过 PCI-C154+ 来操作控制系统。

注意 MotionCreatorPro 能于 Windows XP/7/8.1 操作，且屏幕分辨率需高于 1024x768。不可于 DOS 环境下执行。

5.1 执行 MotionCreatorPro

当于 Windows XP/7/8.1 下安装 PCI-C154+ 软件驱动程序后，MotionCreatorPro 将位于 ”<chosen path>\PCI-Motion\MotionCreatorPro” 或 ” 开始 > 程序集 >PCI-Motion>MotionCreXP/7/8.1atorPro”。

5.2 关于 MotionCreatorPro

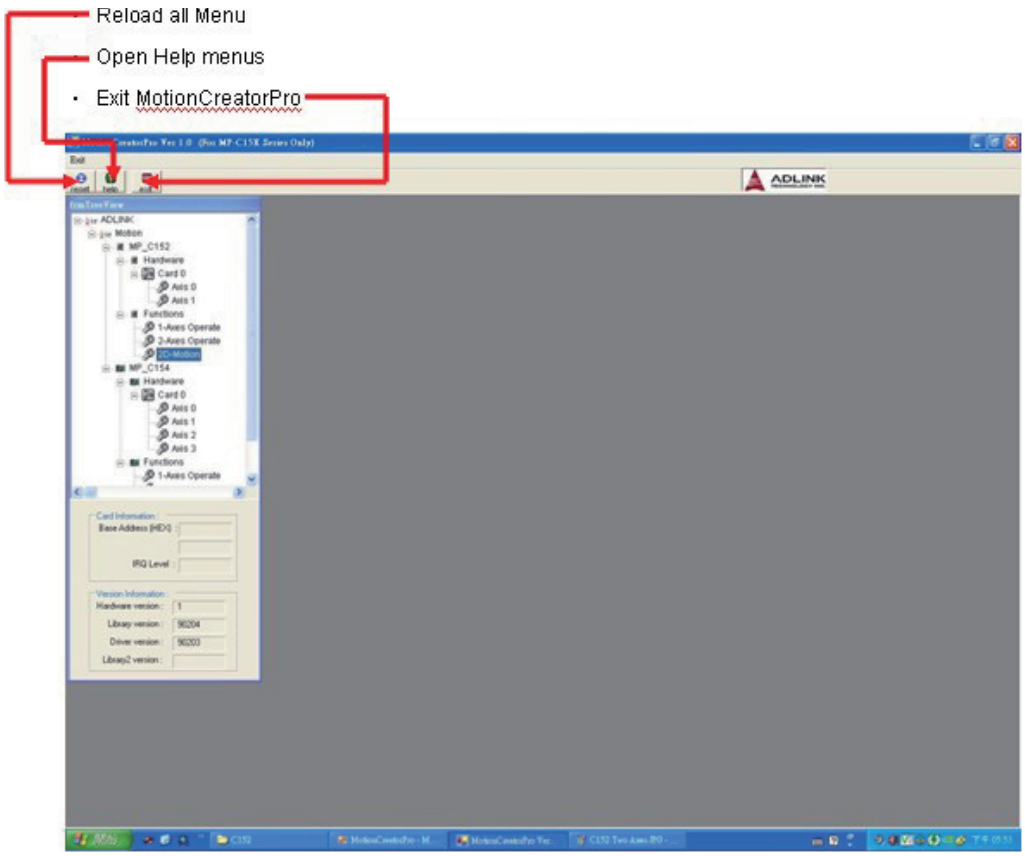
执行 MotionCreatorPro 之前，请先注意以下提醒

1. MotionCreatorPro 为使用 VB.NET 2003 编译而成，只能执行于 Windows XP/7/8.1，且屏幕分辨率需高于 1024x768。不可于 DOS 环境下执行。
2. 使用者可利用 MotionCreatorPro 来储存 PCI-C154+ 设定参数。储存的设定将于下次 MotionCreatorPro 启动时自动导入。 **C154.ini 与 C154MC.ini 于 windows 根目录下，**用来储存所有设定。
3. 欲复制设定参数于其他系统，请将 C154.ini 与 C154MC.ini 复制于其他系统 windows 根目录下。
4. 若多张 PCI-C154+ 欲使用相同设定参数档，用户可于开发程序中利用 **c154_config_from_file()**，将设定导入；此程序亦可于 DOS 环境下使用。

5.3 MotionCreatorPro 介绍

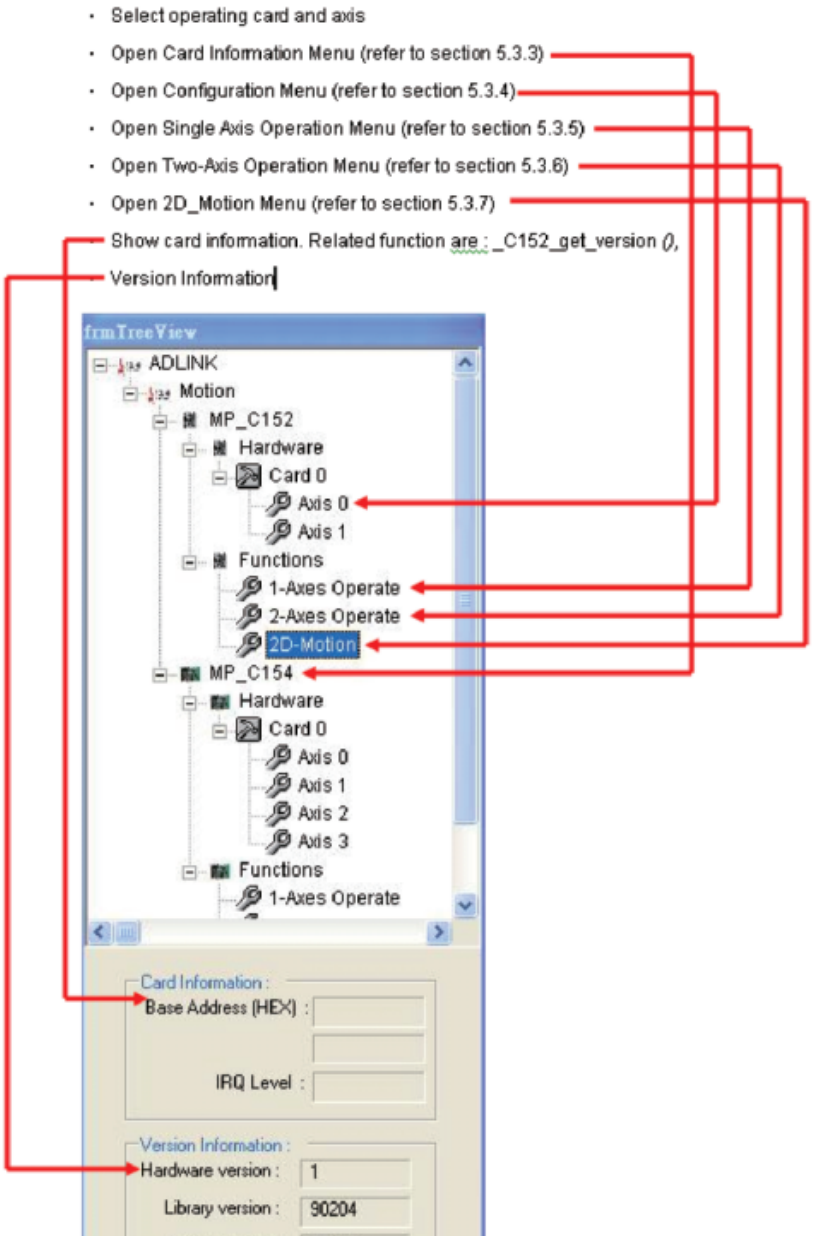
5.3.1 主选单

当 MotionCreatorPro 执行后，首先会看见主选单，其功用如下图：



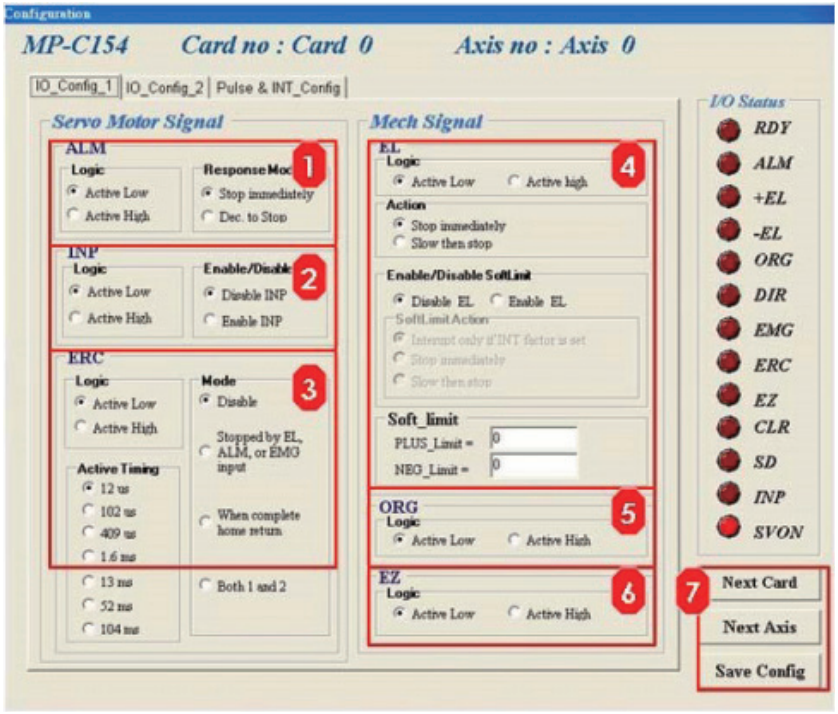
5.3.2 选择单

选择单功能如下：



5.3.3 设定选单

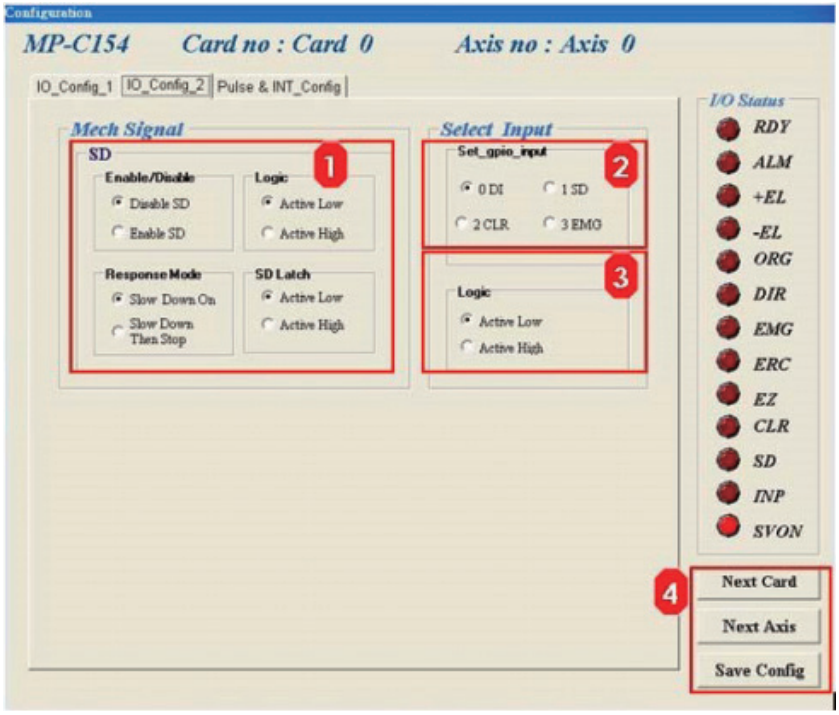
于此设定选单内，使用者可设定 ALM, INP, ERC, EL, ORG 及 EZ。



1. **ALM Logic and Response mode**：设定 ALM 逻辑以及对应行为，其对应函数为 c154_set_alm()。
2. **INP Logic and Enable/Disable selection**：设定 INP 逻辑，以及选择是否启动 INP 功能，其对应函数为 c154_set_inp()。
3. **ERC Logic and Active timing**：设定 ERC 逻辑，以及启动条件，其对应函数为 c154_set_erc()。
4. **EL Response mode**：设定 EL 逻辑，以及对应行为，其对应函数为 c154_set_limit_logic()。
5. **ORG Logic**：设定 ORG 逻辑，其对应函数为 c154_set_home_config()。
6. **EZ Logic**：设定 EZ 逻辑，其对应函数为 c154_set_home_config()。
7. **按钮**：
   - ▷ **Next Axis**：进入下一轴设定。

▷ **Save Config**：储存目前设定于 C154.ini 与 C154MC.ini。

于此设定选单内，可设定 SD 与 Select Input。

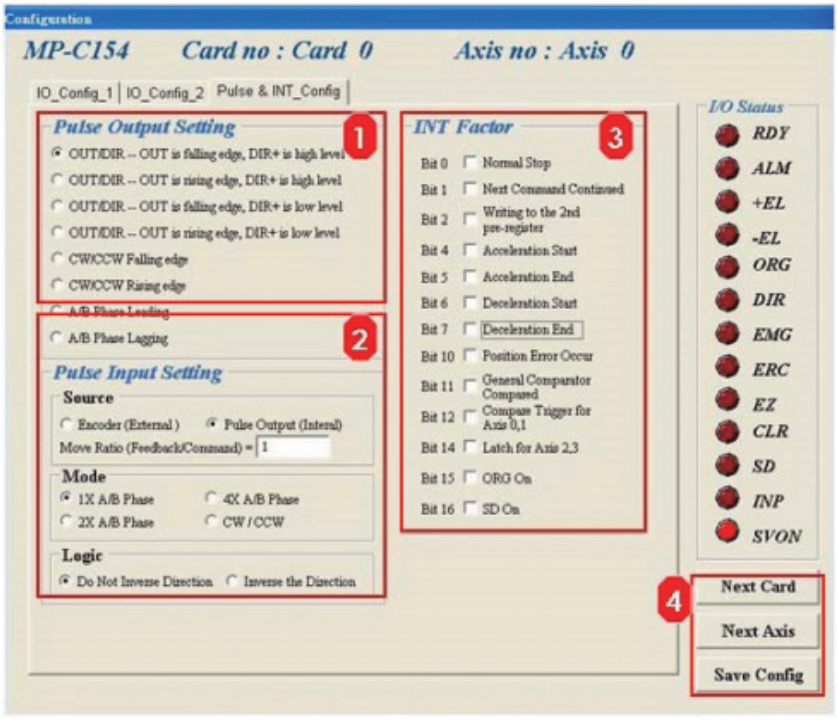


1. **SD Configuration**：设定 SD 信号，其对应函数为 c154_set_sd()。
2. **Set_gpio_input**：设定多功能输入信号，其对应函数为 c154_set_gpio_input_function()。
3. **Logic**：设定多功能输入信号逻辑，其对应函数为 c154_set_gpio_input_function()。
4. **按钮**：

   ▷ **Next Axis**：进入下一轴设定。

   ▷ **Save Config**：储存目前设定于 C154.ini 与 C154MC.ini。

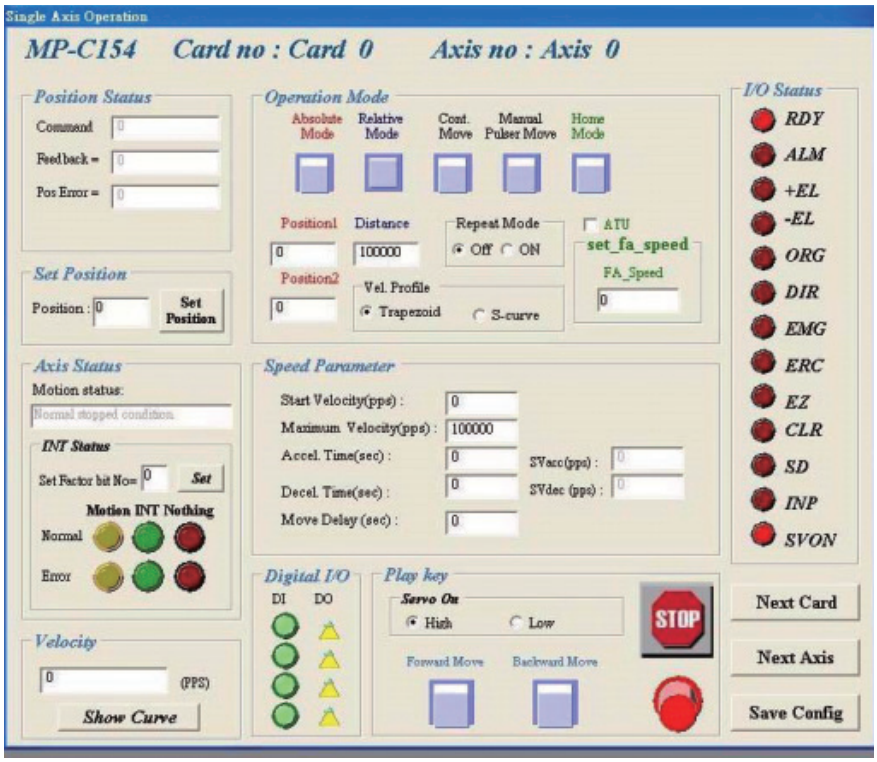
   于此设定选单内，可设定 pulse input/output、 move ratio 以及 INT factor。



1. **Pulse Output Mode**：设定 OUT/DIR 输出模式，其对应函数为 c154_set_pls_outmode()。
2. **Pulse Input**：设定 EA/EB 输入模式，其对应函数为 c154_set_pls_iptmode()、 c154_set_feedback_src()。
3. **INT Factor**：选择事件中断条件，其对应函数为 c154_set_int_factor()。
4. **按钮**：
   - ▷ **Next Axis**：进行下一轴设定。
   - ▷ **Save Config**：储存目前设定于 C154.ini 与 C154MC.ini。

5.3.4 单轴操作页面

于此页面，使用者可做单轴的操作，包括速度模式运动、相对 / 绝对运动、手动脉冲输入运动以及归零运动。



1. **Position**：
   - ▷ **Command**：显示命令位置，其对应函数为 c154_get_command()。
   - ▷ **Feedback**：显示反馈位置，其对应函数为 c154_get_position()。
   - ▷ **Pos Error**：显示位置误差，其对应函数为 c154_get_error_counter()。
2. **Position Reset**：按下此按钮，所有位置计数器将会被设定于特定数值，其对应函数为：

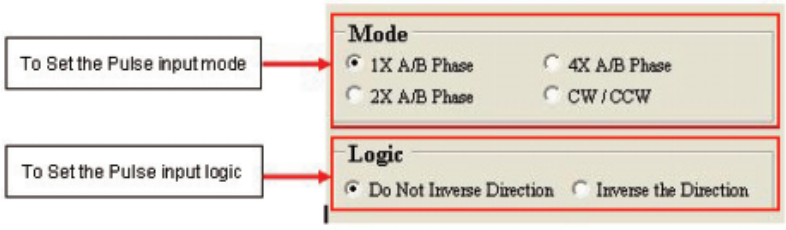
   c154_set_position()

   c154_set_command()

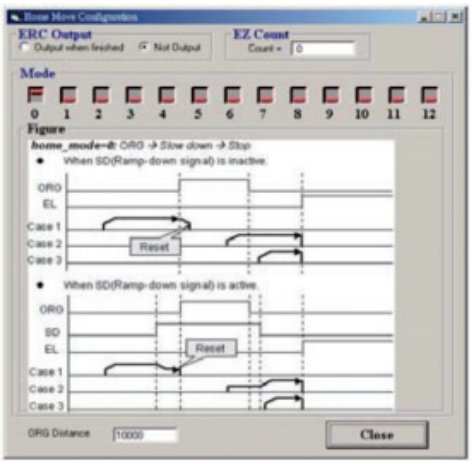
   c154_reset_error_counter()

   c154_reset_target_pos()

3. **Motion Status**：显示函数 c154_motion_done 所传回数值，其对应函数为 c154_motion_done()。
4. **NT Status**：
   - ▷ **int_factor bit no**：设定 int_factor.
   - ▷ **Normal INT**：显示一般中断状态，其对应函数为 c154_wait_motion_interrupt()。
   - ▷ **Error INT**：显示错误中断状态，其对应函数为 c154_wait_error_interrupt()。
   - ▷ **GPIO INT**：显示通用输出入信号中断状态，其对应函数为 c154_wait_gpio_interrupt()。
5. **Velocity**：速度绝对值，其单位为 PPS，其对应函数为 c154_get_current_speed()。
6. **Show Velocity Curve Button**：按下此按钮，将会开启速度对应时间的曲线图，其速度值采样率为 100ms ；同样按此按钮，将会关闭窗口；欲清除数据，请于曲线图上按下左键。
7. **Operation Mode**：选择操作模式。
   - ▷ **Absolute Mode**：“Position1” 与 “position2” 用于设定绝对运动的目标位置，其对应函数为 c154_start_ta_move(), c154_start_sa_move()。
   - ▷ **Relative Mode**：“Distance” 用于设定相对运动移动距离，其对应函数为 c154_start_tr_move(), c154_start_sr_move()。
   - ▷ **Cont. Move**：速度模式运动，其对应函数为 c154_tv_move(), c154_start_sv_move()。
   - ▷ **Manual Pulser Move**：手动脉冲输入运动，按下此按钮，将会开启手动脉冲输入运动设定窗口。



▷ **Home Mode**：归零运动，按下此按钮将会开启归零运动设定窗口，其对应函数为 c154_set_home_config()。若勾选 “ATU”，当运动开始时，将会执行自动归零运动。



ERC Output：若勾选，当归零运动结束时，将会发送 ERC 信号。

EZ Count：设定 EZ 计数次数，将会影响归零运动模式。

Mode：选择归零运动模式，一共有 13 种模式可供选择。

Home Mode figure：此图解释不同归零运动模式的方法。

Close：按下此按钮以关闭窗口。

8. **Position**：设定绝对位置，此值只适用于绝对运动模式。
9. **Distance**：设定相对移动距离，此值只适用于相对运动模式。
10. **Repeat Mode**：当选择 “On” 时，将会做反复运动 (forward<->backward 或 position1<->position2)，此只适用于绝对运动或相对运动模式。
11. **Vel. Profile**：选择速度曲线 (T 型与 S 型曲线)，可应用于绝对运动、相对运动、速度模式运动。
12. **FA Speed/ATU:** 设定 FA 速度，其对应函数为 c154_set_fa_speed()。若点选 ATU，当运动开始时，即会执行自动归零运动。
13. **Motion Parameters**：设定单轴运动参数，若为手动脉冲输入运动，此参数将不可参考，运动速度与距离将由脉冲输入决定。

    ▷ **Start Velocity**：设定起始速度，单位为 PPS，此参数只于绝对 / 相对运动中可参考；正负符号代表方向，例如 -100 代表以 100 的速度往负方向移动。

- ▷ **Maximum Velocity**：设定最大运动速度，单位为 PPS，此参数只于绝对 / 相对运动中可参考；正负符号代表方向，例如 -5000 代表以 5000 的速度往负方向移动。
- ▷ **Accel. Time**：设定加速度时间，单位为秒。
- ▷ **Decel. Time**：设定减速度时间，单位为秒。
- ▷ **SVacc**：设定 S-curve 于加速度段的范围，单位为 PPS。
- ▷ **SVdec**：设定 S-curve 于减速度段的范围，单位为 PPS。
- ▷ **Move Delay**：此功能只适用于反复运动模式，此值将决定两个运动间的延迟时间。

14. **Digital I/O**：显示以及设定数字输出入信号，其对应函数为 c154_get_gpio_output(), c154_get_gpio_input(), c154_set_gpio_output()。

15. **Servo On**：设定伺服激磁输出信号，其对应函数为 c154_set_servo()。

16. **Play Key:**

Left play button：按下此按钮， PCI-C154+ 将依据上述设定开始输出脉冲控制命令信号。

- ▷ 于 ” 绝对运动模式 ”，将控制轴移动至 position1。
- ▷ 于 ” 相对运动模式 ”，将控制轴向前运动。
- ▷ 于 ” 速度模式运动 ”，将依据速度设定，控制轴往正方向运动。
- ▷ 于 ” 手动脉冲输入运动模式 ”，将控制轴进行手动脉冲运动。

最大速度限制为 “Maximum Velocity”。

Right play button：按下此按钮， PCI-C154+ 将依据上述设定开始输出脉冲控制命令信号。

- ▷ 于 ” 绝对运动模式 ”，将控制轴移动至 position2。
- ▷ 于 ” 相对运动模式 ”，将控制轴向后运动。
- ▷ 于 ” 速度模式运动 ”，将依据速度设定，控制轴往反方向运动。

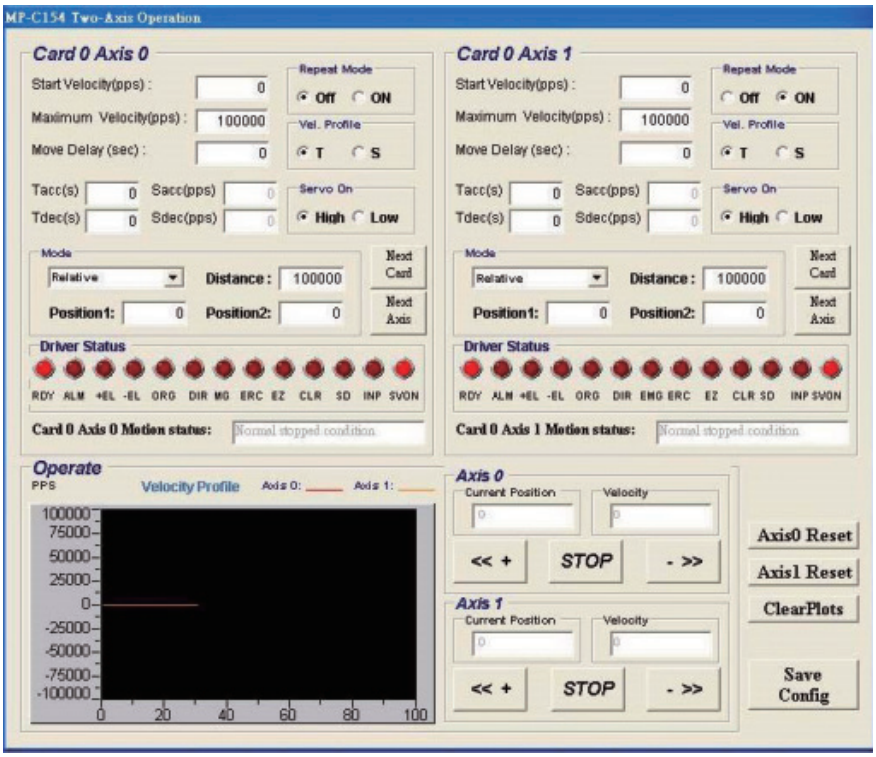
- ▷ 于 ” 手动脉冲输入运动模式 ”，将控制轴进行手动脉冲运动。

最大速度限制为 “Maximum Velocity”。

17. **Stop Button**：按下此按钮，将导致 PCI-C154+ 开始减速至停止，减速时间依据 “Decel Time” 设定，其对应函数为 c154_sd_stop()。
18. **I/O Status**：显示相关输出入信号状态，灯亮代表事件发生，其对应函数为 c154_get_io_status()。
19. **Buttons** ：
    - ▷ **Next Card**：进行下一张卡设定。
    - ▷ **Next Axis**：进行下一轴设定。
    - ▷ **Save Config**：储存目前设定于 C154.ini 与 C154MC.ini。

5.3.5 两轴与四轴操作页面

于此操作页面，使用者可以设定两轴运动参数，包括速度模式运动，以及绝对 / 相对运动。



1. **Motion Parameters**：设定单轴运动参数；若运动模式为 ” 手动脉冲输入模式 ”，此参数将被忽略。
   - ▷ **Start Velocity**：设定起始速度，单位为 PPS，此参数只于绝对 / 相对运动中可参考；正负符号代表方向，例如 -100 代表以 100 的速度往负方向移动。
   - ▷ **Maximum Velocity**：设定最大运动速度，单位为 PPS，此参数只于绝对 / 相对运动中可参考；正负符号代表方向，例如 -5000 代表以 5000 的速度往负方向移动。
   - ▷ **Tacc**：设定加速度时间，单位为秒。
   - ▷ **Tdec**：设定减速度时间，单位为秒。
   - ▷ **Sacc**：设定 S-curve 于加速度段的范围，单位为 PPS。
   - ▷ **Sdec**：设定 S-curve 于减速度段的范围，单位为 PPS。

2. **Repeat Mode**：当选择 “On” 时，将会做反复运动 (forward<->backward 或 position1<->position2)，此只适用于绝对运动或相对运动模式。
3. **Vel. Profile**：选择速度曲线 (T 型与 S 型曲线)，可应用于绝对运动、相对运动、速度模式运动。
4. **Servo ON-OFF**：输出伺服激磁信号。
5. **Operation Mode**：选择运动模式。
   - ▷ **Absolute Mode**：“Position1” 与 “Position2” 用于设定绝对运动的目标位置，其对应函数为 c154_start_ta_move(), c154_start_sa_move()。
   - ▷ **Relative Mode**：“Distance” 用于设定相对运动移动距离，其对应函数为 c154_start_tr_move(), c154_start_sr_move()。
6. **Distance**：设定相对移动距离，此值只适用于相对运动模式。
7. **Position**：设定绝对位置，此值只适用于绝对运动模式。
8. **Buttons**：
   - ▷ **Next Card**：进行下一张卡设定。
   - ▷ **Next Axis**：进行下一轴设定。
9. **I/O Status**：显示相关输出入信号状态，灯亮代表事件发生，其对应函数为 c154_get_io_status()。
10. **Motion status**：显示函数 c154_motion_done 所传回数值，其对应函数为 c154_motion_done()。
11. **Current Position**：
    - ▷ **Command**：显示命令位置，其对应函数为 c154_get_command()。
    - ▷ **Velocity**：速度绝对值，单位为 PPS，其对应函数为 c154_get_current_speed()。

12.**Play Key**：

Left play button：按下此按钮， PCI-C154+ 将依据上述设定开始输出脉冲控制命令信号。

- ▷ 于 ” 绝对运动模式 ”，将控制轴移动至 position1。
- ▷ 于 ” 相对运动模式 ”，将控制轴向前运动。

Right play button：按下此按钮， PCI-C154+ 将依据上述设定开始输出脉冲控制命令信号。

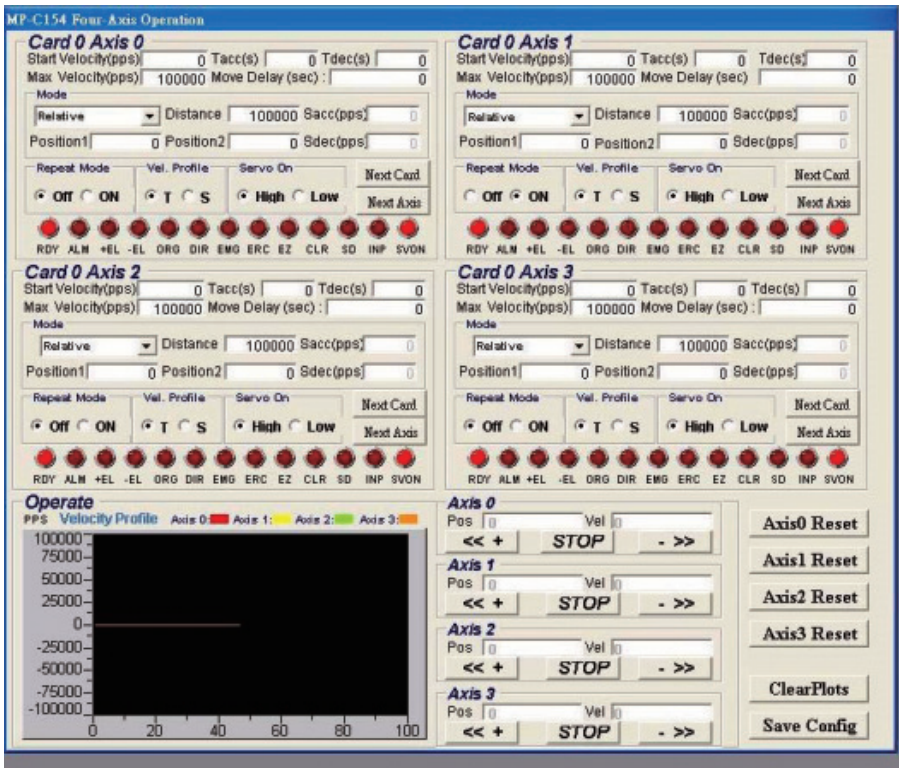
- ▷ 于 ” 绝对运动模式 ”，将控制轴移动至 position2。
- ▷ 于 ” 相对运动模式 ”，将控制轴向后运动。

Stop Button：按下此按钮，将导致 PCI-C154+ 开始减速至停止，减速时间依据 “Decel Time” 设定， 其对应函数为 c154_sd_stop()。

13.**Buttons**：

- ▷ **Axis0 Reset**：重整轴 0 所有设定。
- ▷ **Axis1 Reset**：重整轴 1 所有设定。
- ▷ **ClearPlots**：清除图表。
- ▷ **Save Config**：储存目前设定于 C154.ini 与 C154MC.ini。

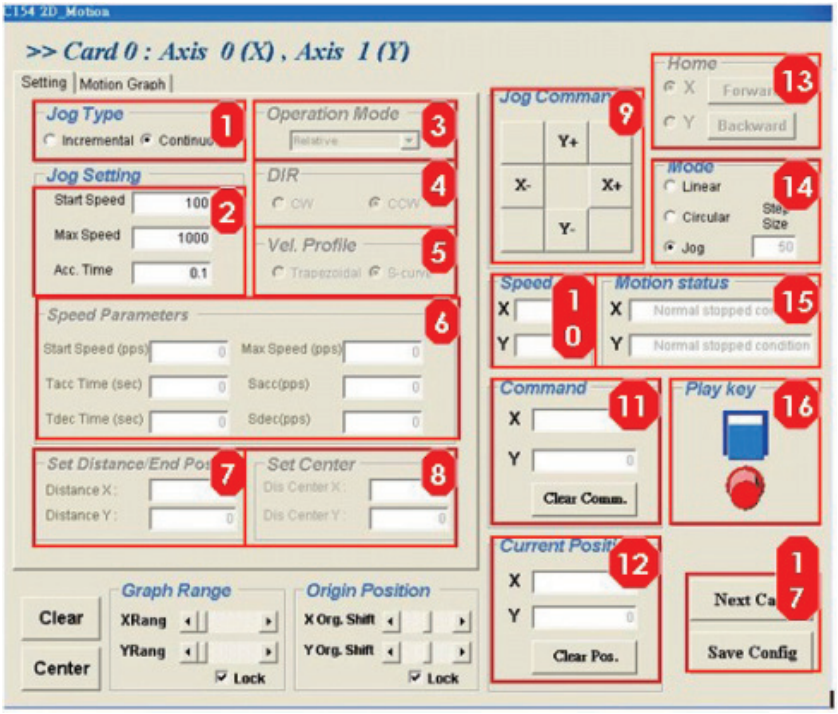
14.四轴操作页面 ：



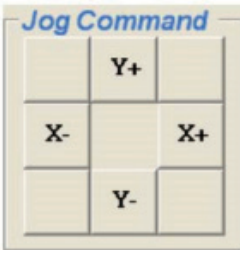
5.3.6 二维运动操作页面

按下 2D-Motion 按钮，将会出现以下页面，此页面包括：

- 线性插补
- 圆弧插补
- 寸进模式 Jog
- 连续模式 Jog
- 其他控制组件



1. **Jog Type**：



Continuous Jog：选择连续 Jog 运动时，当按下方向按钮，控制轴会开始连续运动，直到放开方向按钮，各轴便会立刻停止运动。

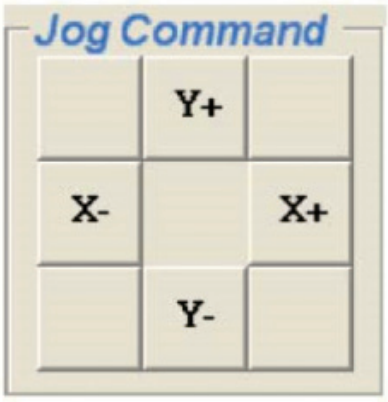


Incremental Jog：选择寸进 Jog 运动时，当按下方向按钮，控制轴会移动一段距离，此距离由 ”Step Size” 内数值决定。

2. **Jog Setting**：设定控制轴参数。
   - ▷ **Start Velocity**：设定起始速度，单位为 PPS。
   - ▷ **Maximum Velocity**：设定最大运动速度，单位为 PPS。
   - ▷ **Tacc**：设定加速度时间，单位为秒。
3. **Operation Mode**：选择运动模式。

- **Absolute Mode**：当选择 “Linear Interpolation Mode” 时， “Position” 用于设定绝对运动的目标位置；当选择 “Circular Interpolation Mode” 时，“ABS EndPos” 与 “ABS Center” 用于设定绝对运动的目标位置与圆心位置。其对应函数为 c154_start_ta_move(), c154_start_sa_move()。
- **Relative Mode**：当选择 “Linear Interpolation Mode” 时， “Distance” 用于设定相对运动移动距离；当选择 “Circular Interpolation Mode” 时， “Dis EndPos” 与 “Dis Center” 用于设定相对运动移动距离与圆心位置。其对应函数为 c154_start_tr_move(), c154_start_sr_move()。

4. **DIR**：选择圆弧插补方向。
5. **Vel. Profile**：选择速度曲线 (T 型与 S 型曲线)，可应用于线性插补以及圆弧插补。
6. **Speed Parameters**：设定控制轴参数。
   - **Start Velocity**：设定起始速度，单位为 PPS。
   - **Maximum Velocity**：设定最大运动速度，单位为 PPS。
   - **Accel. Time**：设定加速度时间，单位为秒。
   - **Decel. Time**：设定减速度时间，单位为秒。
   - **SVacc**：设定 S-curve 于加速度段的范围，单位为 PPS。
   - **SVdec**：设定 S-curve 于减速度段的范围，单位为 PPS。
7. **Set Distance/End Pos**：设定绝对运动目标位置，或相对运动移动距离，可应用于线性插补与圆弧插补。
8. **Set Center**：设定圆弧插补模式的圆心位置。
9. **Jog Command**：Jog 移动按钮。



10. **Velocity**：运动速度的绝对值，单位为 PPS，其对应函数为 c154_get_current_speed()。
11. **Interpolation Command**：显示命令位置，其对应函数为 c154_get_command()。
12. **Current Position**：显示反馈位置，其对应函数为 c154_get_position()。
13. **Home Mode**：归零运动，按下此按钮，将会开启归零运动设定页面，其对应函数为 c154_set_home_config()。
14. **Mode**：

Linear Interpolation：当设定完运动参数后，于 ”Step Size” 内输入距离，按下 Run 钮后开始做线性插补运动。

Circular Interpolation ：设定圆心位置、目标位置以及插补方向后，按下 Run 钮后开始做圆弧插补运动。相关设定请参考附录 H。

Jog Type：

- ▷ **Continuous Jog** ：选择连续 Jog 运动时，当按下方向按钮，控制轴会开始连续运动，直到放开方向按钮，各轴便会立刻停止运动。
- ▷ **Incremental Jog**：选择吋进 Jog 运动时，当按下方向按钮，控制轴会移动一段距离，此距离由 ”Step Size” 内数值决定。

15. **Motion status**：显示函数 c154_motion_done 所传回的数值，其对应函数为 c154_motion_done()。

16.**Play Key**：

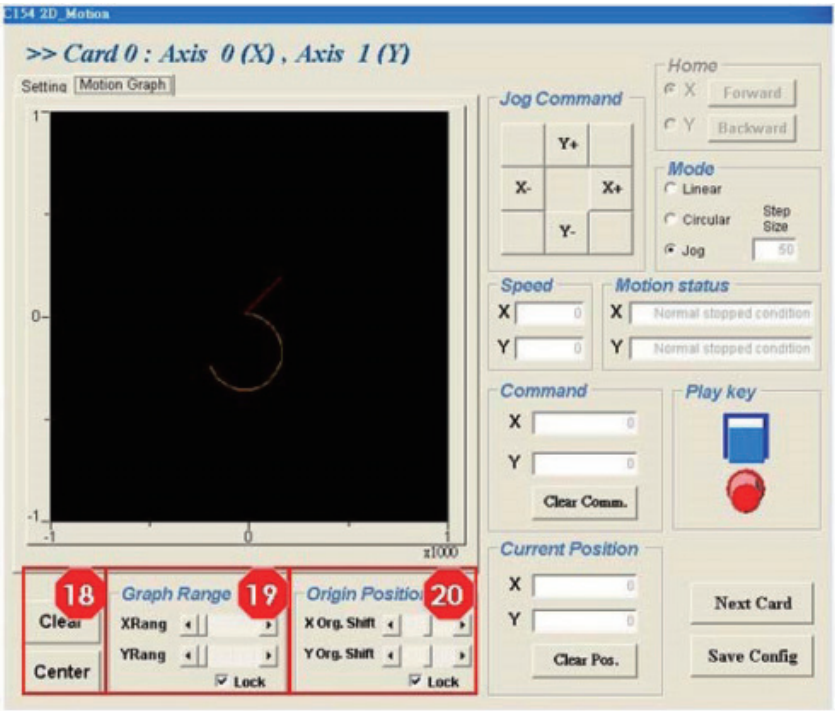
Play button：按下此按钮， PCI-C154+ 将依据设定参数开始输出脉冲控制命令信号。

- ▷ 于 “Linear Mode” 中，其对应函数为 c154_start_tr_move_xy, c154_start_sr_move_xy。
- ▷ 于 “Circular Mode” 中，其对应函数为 c154_start_tr_arc_xy, c154_start_sr_arc_xy。

Stop Button：按下此按钮，将导致 PCI-C154+ 开始减速至停止，减速时间依据 “Decel Time” 设定，其对应函数为 c154_sd_stop()。

17.**Buttons**：

- ▷ **Next Card**：进行下一张卡设定。
- ▷ **Save Config**：储存目前设定于 C154.ini 与 C154MC.ini。



18. **Graph Range Frame**：
    - ▷ **Clear**：清除图表。
    - ▷ **Center**：显示图表于中心位置。
19. **Graph Range**：控制 X 与 Y 轴显示范围。
20. **Origin Position**：控制图表显示位置。

6 函数库

PCI-C154+ 提供两种函数库供使用者使用，分别为 APS 函数库 (Automation Product Software)，与传统专用函数库。 APS 函数库提供统一的软件接口来操控凌华科技多种运动控制卡片，若用户使用 APS 函数库进行开发，当更换凌华科技不同运动控制产品时，用户不需重新学习新函数即可控制不同板卡，甚至可以直接重复使用已开发的应用程序，此弹性将大大提升用户的开发时间。APS 函数库详细内容请参考 APS 手册。以下附录为传统函数库描述。

请注意，传统函数库尚未提供 ” 高速位置比较触发功能 (4.6 节)” 与 ” 高速位置撷取功能 (4.7 节)”。

附录

本附录将描述 PCI-C154+ 的传统专用函数，使用者可以利用这些函数于 C、C++ 以及 Visual Basic 语言环境中；若要使用 Delphi 开发环境，使用者必须手动转换 c154.h 档案。

A 传统专用函数列表

系统与初始化 (System & Initialization), 附录 C

函数名称	描述
c154_initial	卡片初始化
c154_close	卡片资源释放
c154_get_version	读取卡片硬件与软件版本
c154_set_security_key	设定卡片安全码
c154_check_security_key	检查卡片安全码
c154_reset_security_key	重置卡片安全码至初始值
c154_config_from_file	从档案设定 PCI-C154+ 参数

脉冲输出入设定 (Pulse Input/Output Configuration), 附录 D

函数名称	描述
c154_set_pls_outmode	设定脉冲控制命令输出信号模式
c154_set_pls_iptmode	设定反馈脉冲输入信号模式
c154_set_feedback_src	设定反馈计数器信号源

速度运动模式 (Velocity mode motion), 附录 E

函数名称	描述
c154_tv_move	控制运动轴依照 T 型速度曲线加速至特定速度持续运动
c154_sv_move	控制运动轴依照 S 型速度曲线加速至特定速度持续运动
c154_sd_stop	减速停止
c154_emg_stop	立即停止
c154_get_current_speed	读取目前速度 (PPS)
c154_speed_override	实时重设速度
c154_set_max_override_speed	实时重设最大速度

单轴运动模式 (Single Axis Position Mode), 附录 F

函数名称	描述
c154_start_tr_move	依照 T 型速度曲线进行相对距离运动
c154_start_ta_move	依照 T 型速度曲线进行绝对位置运动
c154_start_sr_move	依照 S 型速度曲线进行相对距离运动
c154_start_sa_move	依照 S 型速度曲线进行绝对位置运动
c154_set_move_ratio	设定命令脉冲信号与反馈脉冲信号比例

多轴直线插补运动 (Linear Interpolated Motion), 附录 G

函数名称	描述
c154_start_tr_move_xy	依 T 型速度曲线进行 XY 两轴相对距离直线插补运动
c154_start_ta_move_xy	依 T 型速度曲线进行 XY 两轴绝对位置直线插补运动
c154_start_sr_move_xy	依 S 型速度曲线进行 XY 两轴相对距离直线插补运动
c154_start_sa_move_xy	依 S 型速度曲线进行 XY 两轴绝对位置直线插补运动
c154_start_tr_move_zu	依 T 型速度曲线进行 ZU 两轴相对距离直线插补运动
c154_start_ta_move_zu	依 T 型速度曲线进行 ZU 两轴绝对位置直线插补运动
c154_start_sr_move_zu	依 S 型速度曲线进行 ZU 两轴相对距离直线插补运动
c154_start_sa_move_zu	依 S 型速度曲线进行 ZU 两轴绝对位置直线插补运动
c154_start_tr_line2	依 T 型速度曲线进行任意两轴相对距离直线插补运动
c154_start_ta_line2	依 T 型速度曲线进行任意两轴绝对位置直线插补运动
c154_start_sr_line2	依 S 型速度曲线进行任意两轴相对距离直线插补运动
c154_start_sa_line2	依 S 型速度曲线进行任意两轴绝对位置直线插补运动
c154_start_tr_line3	依 T 型速度曲线进行任意三轴相对距离直线插补运动
c154_start_ta_line3	依 T 型速度曲线进行任意三轴绝对位置直线插补运动
c154_start_sr_line3	依 S 型速度曲线进行任意三轴相对距离直线插补运动
c154_start_sa_line3	依 S 型速度曲线进行任意三轴绝对位置直线插补运动
c154_start_tr_line4	依 T 型速度曲线进行四轴相对距离直线插补运动
c154_start_ta_line4	依 T 型速度曲线进行四轴绝对位置直线插补运动
c154_start_sr_line4	依 S 型速度曲线进行四轴相对距离直线插补运动
c154_start_sa_line4	依 S 型速度曲线进行四轴绝对位置直线插补运动

两轴圆弧插补运动 (Circular Interpolation Motion), 附录 H

函数名称	描述
c154_start_tr_arc_xy	依 T 型速度曲线进行 XY 两轴相对距离圆弧插补运动
c154_start_ta_arc_xy	依 T 型速度曲线进行 XY 两轴绝对位置圆弧插补运动
c154_start_sr_arc_xy	依 S 型速度曲线进行 XY 两轴相对距离圆弧插补运动
c154_start_sa_arc_xy	依 S 型速度曲线进行 XY 两轴绝对位置圆弧插补运动
c154_start_tr_arc_zu	依 T 型速度曲线进行 ZU 两轴相对距离圆弧插补运动

函数名称	描述
c154_start_ta_arc_zu	依 T 型速度曲线进行 ZU 两轴绝对位置圆弧插补运动
c154_start_sr_arc_zu	依 S 型速度曲线进行 ZU 两轴相对距离圆弧插补运动
c154_start_sa_arc_zu	依 S 型速度曲线进行 ZU 两轴绝对位置圆弧插补运动
c154_start_tr_arc2	依 T 型速度曲线进行任意两轴相对距离圆弧插补运动
c154_start_ta_arc2	依 T 型速度曲线进行任意两轴绝对位置圆弧插补运动
c154_start_sr_arc2	依 S 型速度曲线进行任意两轴相对距离圆弧插补运动
c154_start_sa_arc2	依 S 型速度曲线进行任意两轴绝对位置圆弧插补运动

归零运动 (Home Return Mode), 附录 I

函数名称	描述
c154_set_home_config	设定归零运动模式与信号逻辑
c154_home_move	进行归零运动
c154_home_search	进行零点位置搜寻

脉冲发生器运动模式 (Manual Pulser Motion), 附录 J

函数名称	描述
c154_set_pulser_iptmode	设定脉冲发生器输入模式
c154_disable_pulser_input	关闭脉冲发生器输入信号
c154_pulser_vmove	开始藉由脉冲发生器进行速度模式运动
c154_pulser_pmove	开始藉由脉冲发生器进行位置模式运动
c154_set_pulser_ratio	设定脉冲发生器输入与脉冲控制命令输出比例

运动状态 (Motion Status), 附录 K

函数名称	描述
c154_motion_done	读取 PCI-C154+ 运动状态

运动界面 I/O(Motion Interface I/O), 附录 L

函数名称	描述
c154_set_servo	设定 SVON 信号 ON/OFF
c154_set_clr_mode	设定 CLR 模式
c154_set_inp	设定 INP 逻辑与操作模式
c154_set_alm	设定 ALM 逻辑与操作模式
c154_set_erc	设定 ERC 逻辑与时序
c154_set_erc_out	手动输出 ERC
c154_clr_erc	清除 ERC 信号
c154_set_sd	设定 SD 逻辑与操作模式
c154_enable_sd	启动 SD 信号

函数名称	描述
c154_set_limit_logic	设定 EL 逻辑
c154_set_limit_mode	设定 EL 操作模式
c154_get_io_status	读取 PCI-C154+ 运动界面 I/O 状态

中断控制 (Interrupt Control), 附录 M

函数名称	描述
c154_int_control	启动中断功能
c154_wait_error_interrupt	等待错误中断事件
c154_wait_motion_interrupt	等待运动中断事件
c154_set_motion_int_factor	选择运动中断事件

位置控制与计数器 (Position Control and Counters), 附录 N

函数名称	描述
c154_get_position	读取反馈位置计数器数值
c154_set_position	设定反馈位置计数器数值
c154_get_command	读取命令位置计数器数值
c154_set_command	设定命令位置计数器数值
c154_get_error_counter	读取位置误差计数器数值
c154_reset_error_counter	清除位置误差计数器数值
c154_get_general_counter	读取通用计数器数值
c154_set_general_counter	设定通用计数器

连续运动 (Continuous Motion), 附录 O

函数名称	描述
c154_set_continuous_move	启动 / 关闭连续运动
c154_check_continuous_buffer	检视命令缓存器缓冲是否为满的
c154_dwell_move	设定暂停运动 (dwell move)

多轴同步运动 (Multiple Axes Simultaneous Operation), 附录 P

函数名称	描述
c154_set_tr_move_all	多轴同步 T 型速度曲线相对距离运动设定
c154_set_ta_move_all	多轴同步 T 型速度曲线绝对位置运动设定
c154_set_sr_move_all	多轴同步 S 型速度曲线相对距离运动设定
c154_set_sa_move_all	多轴同步 S 型速度曲线绝对位置运动设定
c154_start_move_all	开始进行多轴同步运动
c154_stop_move_all	停止多轴同步运动

多功能输出入信号 (General-Purpose Input/Output), 附录 Q

函数名称	描述
c154_set_gpio_output	设定多功能输出信号
c154_get_gpio_output	读取多功能输出信号状态
c154_get_gpio_input	读取多功能输入信号状态
c154_set_gpio_input_function	设定多功能输入信号功能

软件极限 (Soft Limit), 附录 R

函数名称	描述
c154_disable_soft_limit	关闭软件极限功能
c154_enable_soft_limit	启动软件极限功能
c154_set_soft_limit	设定软件极限数值

扩展通用输出入信号 (Extened DIO), 附录 S

函数名称	描述
c154_set_gpio_output_ex	设定扩展通用输出信号
c154_get_gpio_output_ex	读取扩展通用输出信号状态
c154_get_gpio_input_ex	读取扩展通用输入信号状态
c154_set_gpio_output_ex_CH	依据通道设定扩展通用输出信号
c154_get_gpio_output_ex_CH	依据通道读取扩展通用输出信号状态
c154_get_gpio_input_ex_CH	依据通道读取扩展通用输入信号状态

背隙补偿与震动抑制 (Backlash Compensation / Vibration Suppression), 附录 T

函数名称	描述
c154_backlash_comp	设定补偿数值
c154_suppress_vibration	设定抑制震动之脉冲输出延迟时间
c154_set_fa_speed	设定 FA 速度 (应用于背隙 / 滑移补偿与归零运动)

B C/C++ 函数库

所有函数以及数据型态皆在 c154.h 中，建议使用这些数据型态在您的应用程序中，以下表格列出所有数据型态名称以及其范围。

数据型态	描述	范围
U8	8-bit ASCII character	0 to 255
I16	16-bit signed integer	-32768 to 32767

数据型态	描述	范围
U16	16-bit unsigned integer	0 to 65535
I32	32-bit signed long integer	-2147483648 to 2147483647
U32	32-bit unsigned long integer	0 to 4294967295
F32	32-bit single-precision floating-point	-3.402823E38 to 3.402823E38
F64	64-bit double-precision floating-point	-1.797683134862315E308 to 1.797683134862315E309
Boolean	Boolean logic value	TRUE, FALSE

表 7-1：数据型态定义

由 PCI-C154+ 函数之名称即可得知其功能，函数于 C 语言下命名原则如下：

{hardware_model}_{action_name}，例如：c154_initial()。

C　系统与初始化 (System and Initialization)

@ 名称

c154_initial – 卡片初始化

c154_close – 卡片资源释放

c154_get_version – 读取卡片硬件与软件版本

c154_set_security_key – 设定卡片安全码

c154_check_security_key – 检查卡片安全码

c154_reset_security_key – 重置卡片安全码至初始值

c154_config_from_file – 从档案设定 PCI-C154+ 参数

@ 描述

c154_initial：

此函数用来初始化PCI-C154+，所有控制卡必须在使用其他函数前先利用此函数来初始化。可藉由参数 Manual_ID 来选择使用手动或自动设定卡片编号。

c154_close：

此函数用来结束 PCI-C154+ 以及释放其资源，于用户应用程序结束前必须调用调用此函数。

c154_get_version：

让使用者读取硬件、驱动程序以及函数库版本。

c154_set_security_key：

设定安全码至板卡上。

c154_check_security_key：

检查板卡安全码。

c154_reset_security_key：

重置板卡上的安全码，预设码为 0。

c154_config_from_file：

利用此函数来加载 PCI-C154+ 配置文件，可先使用 MotionCreatorPro 来测试设定 PCI-C154+，接着储存配置文件，档案为系统目录下的 C154.ini。

当执行此函数后，所有系统上的 PCI-C154+ 板卡皆会依照配置文件 C154.ini 内参数，执行下列函数：

c154_set_limit_logic

c154_set_inp

c154_set_erc

c154_set_alm

c154_set_pls_iptmode

c154_set_pls_outmode

c154_set_move_ratio

c154_set_latch_source

c154_set_feedback_src

c154_set_home_config

c154_set_soft_limit

c154_set_fa_speed

c154_set_sd

@ 语法

C/C++(Windows XP/7/8.1)

I16 c154_initial(U16 *CardID_InBit, I16 Manual_ID);

I16 c154_close(void);

I16 c154_get_version(I16 card_id, I16 *firmware_ver, I32 *driver_ver, I32 *dll_ver);

I16 c154_set_security_key(I16 card_id, I16 old_secu_code, I16 new_secu_code);

I16 c154_check_security_key(I16 card_id, I16 secu_code);

I16 c154_reset_security_key(I16 card_id);

I16 c154_config_from_file();

Visual Basic 6 (Windows XP/7/8.1)

c154_initial(CardID_InBit As Integer, ByVal Manual_ID As Integer) As Integer c154_close() As Integer

c154_get_version(ByVal card_id As Integer, firmware_ver As Integer, driver_ver As Long, dll_ver As Long) As Integer

c154_set_security_key(ByVal card_id As Integer, ByVal old_secu_code As Integer, ByVal new_secu_code As Integer) As Integer

c154_check_security_key(ByVal card_id As Integer, ByVal secu_code As Integer) As Integer

c154_reset_security_key(ByVal card_id As Integer) As Integer

c154_config_from_file() As Integer

@ 参数

***CardID_InBit**：显示系统上的卡片编码，此数值以位方式解读，位 n 用来表示编码为 n 的卡片是否存在，例如数值若是 0x12，则系统上卡片的编码 (card_id) 分别为 1、 4。

Manual_ID：启用卡片编码开关 (SW1) 决定卡片编号，卡片编号可由下两种方式决定：

0：由 PCI 插槽顺序决定

1：由卡片编码开关 (SW1) 决定

card_id：PCI-C154+ 卡片编号，卡片编号可由卡片编码开关 (SW1) 或 PCI 插槽顺序决定，请参阅 c154_initial()。

firmware_ver：硬件版本

driver_ver：驱动程序版本

dll_ver：函数库版本

old_secu_code：旧安全码

new_secu_code：新安全码

secu_code：安全码

D 脉冲输出入设定 (Pulse Input/Output Configuration)

@ 名称

c154_set_pls_iptmode – 设定反馈脉冲输入信号模式

c154_set_pls_outmode – 设定脉冲控制命令输出信号模式

c154_set_feedback_src – 设定反馈计数器信号源

@ 描述

c154_set_pls_iptmode：

设定反馈脉冲输入信号模式，有 4 种模式可供设定，请注意，唯有 c154_set_feedback_src() 内的 Src 选择反馈信号，设定此函数才有意义。

c154_set_pls_outmode：

设定脉冲控制命令输出模式，有 6 种模式可供设定。

c154_set_feedback_src：

若外部可提供编码器反馈信号，则可设定此函数参数 Src 来启动内部计数器依据 c154_set_pls_iptmode() 所设定之反馈信号计数，否则计数器将依据脉冲控制命令输出计数。

@ 语法

C/C++(Windows XP/7/8.1)

I16 c154_set_pls_iptmode(I16 AxisNo, I16 pls_iptmode, I16 pls_logic);

I16 c154_set_pls_outmode(I16 AxisNo, I16 pls_outmode);

I16 c154_set_feedback_src(I16 AxisNo, I16 Src);

Visual Basic6 (Windows XP/7/8.1)

c154_set_pls_iptmode(ByVal AxisNo As Integer, ByVal pls_iptmode As Integer, ByVal pls_logic As Integer) As Integer

c154_set_pls_outmode(ByVal AxisNo As Integer, ByVal pls_outmode As Integer) As Integer

c154_set_feedback_src(ByVal AxisNo As Integer, ByVal Src As Integer) As Integer

@ 参数

AxisNo：运动轴编号

card_id	Physical axis	AxisNo
0	0	0
	1	1
	2	2
	3	3
1	0	4
	1	5
	...	...

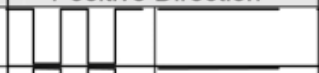
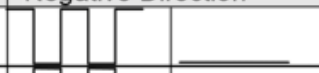
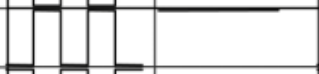
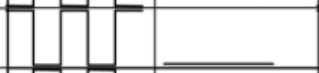
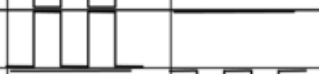
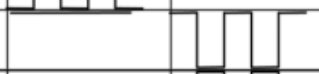
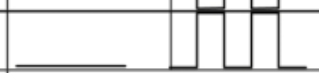
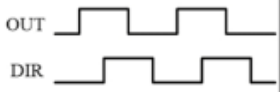
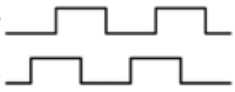
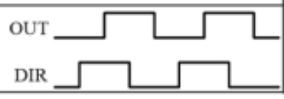
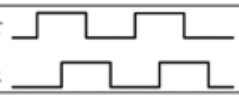
pls_iptmode：反馈脉冲输入模式 (EA/ EB 信号)

Value	Meaning
0	1X A/B
1	2X A/B
2	4X A/B
3	CW/CCW

pls_logic：反馈脉冲输入信号逻辑

Value	Meaning
0	Not inverse direction
1	inverse direction

pls_outmode：脉冲控制命令输出模式

Value	Type	Positive Direction	Negative Direction
0	OUT/DIR		
1	OUT/DIR		
2	OUT/DIR		
3	OUT/DIR		
4	CW / CCW		
5	CW / CCW		
6	AB		
7	AB		

Src：计数器来源

Value	Meaning
0	External signal feedback
1	Command pulse

E　速度运动模式 (Velocity mode motion)

@ 名称

c154_tv_move – 控制运动轴依照 T 型速度曲线加速至特定速度持续运动

c154_sv_move – 控制运动轴依照 S 型速度曲线加速至特定速度持续运动

c154_emg_stop – 立即停止

c154_sd_stop – 减速停止

c154_get_current_speed – 读取目前速度 (PPS)

c154_speed_override – 实时重设速度 (PPS)

c154_set_max_override_speed – 实时重设最大速度 (PPS)

@ 描述

c154_tv_move：

依据 T 型速度曲线，控制运动轴加速至特定速度，运动轴将持续依照特定速度运转，直到速度改变或下达停止命令。运动方向由速度参数的正负符号决定。

c154_sv_move：

依据 S 型速度曲线，控制运动轴加速至特定速度，运动轴将持续依照特定速度运转，直到速度改变或下达停止命令。运动方向由速度参数的正负符号决定。

c154_emg_stop：

利用此函数可立即停止正在做任何运动模式的控制轴。

c154_sd_stop：

依据 T 型或 S 型速度曲线，使正在做任何运动模式的控制轴减速停止；其速度曲线由一开始的运动模式决定。

c154_get_current_speed：

读取目前脉冲命令输出速度 (pulse/sec)，可于任何时间使用此函数来读取当下速度。

c154_speed_override：

实时重设速度功能，用户可先利用 c154_set_max_override_speed() 设定一速度做为 100% 的速度，接着于运动过程中下达百分比例数值，即可更新速度。若用户没透过 c154_set_max_override_speed() 设定速度值，则默认 100% 速度为最后运动命令之最大速度，详细请参阅 4.2.14 节。

c154_set_max_override_speed：

设定速度重设功能之 100% 速度。

@ 语法

C/C++(Windows XP/7/8.1)

I16 c154_tv_move(I16 AxisNo, F64 StrVel, F64 MaxVel, F64 Tacc);

I16 c154_sv_move(I16 AxisNo, F64 StrVel, F64 MaxVel, F64 Tacc, F64 SVacc);

I16 c154_emg_stop(I16 AxisNo);

I16 c154_sd_stop(I16 AxisNo, F64 Tdec);

I16 c154_get_current_speed(I16 AxisNo, F64 *speed)

I16 c154_speed_override(I16 AxisNo, F64 NewVelPercent, F64 Time);

I16 c154_set_max_override_speed(I16 AxisNo, F64 OvrdSpeed, I16 Enable);

Visual Basic6 (Windows XP/7/8.1)

c154_tv_move(ByVal AxisNo As Integer, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double) As Integer

c154_sv_move(ByVal AxisNo As Integer, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal SVacc As Double) As Integer

c154_emg_stop(ByVal AxisNo As Integer) As Integer

c154_sd_stop(ByVal AxisNo As Integer, ByVal Tdec As Double) As Integer

c154_get_current_speed(ByVal AxisNo As Integer, ByRef Speed As Double) As Integer

c154_speed_override(ByVal AxisNo As Integer, ByVal NewVelPercent As Double, ByVal Time As Double) As Interger

c154_set_max_override_speed(ByVal AxisNo As Integer, ByVal OvrdSpeed As Double, ByVal Enable As Integer) As Integer

@ 参数

AxisNo：运动轴编号

card_id	Physical axis	AxisNo
0	0	0
	1	1
	2	2
	3	3

card_id	Physical axis	AxisNo
1	0	4
	1	5
	...	...

StrVel：起始速度 (PPS)

MaxVel：最大速度 (PPS)

Tacc：加速时间 (S)

SVacc：加速端的 S 曲线参数

注：SVacc = 0 为纯 S 曲线，详细内容请参阅 4.2.4 节

Tdec：减速时间 (S)

***Speed**：当下速度 (PPS)

NewVelPercent：速度重设功能之速度比例

Time：切换至重设速度之加减速时间

OvrdSpeed：速度重设功能之 100% 速度

Enable：使用 OvrdSpeed 当作速度重设功能之 100% 速度，否则最后运动命令之最大速度将作为 100% 速度

F 单轴运动模式 (Single Axis Position Mode)

@ 名称

c154_start_tr_move – 依照 T 型速度曲线进行相对距离运动

c154_start_ta_move – 依照 T 型速度曲线进行绝对位置运动

c154_start_sr_move – 依照 S 型速度曲线进行相对距离运动

c154_start_sa_move – 依照 S 型速度曲线进行绝对位置运动

c154_set_move_ratio – 设定命令脉冲信号与反馈脉冲信号比例

@ 描述

General：

运动方向由参数 Pos 或 Dist 的正负符号决定；若运动距离太短以至于无法达到用户设定之运动速度，控制器将会自动降低 MaxVel 值，且缩短 Tacc、Tdec、VSacc 以及 VSdec 时间，以确保加减速及 Jerk 保持不变。

c154_start_tr_move：

依据 T 型速度曲线，控制运动轴由 StrVel 加速至 MaxVel，持续以 MaxVel 速度运转，接着减速停止至特定相对距离；加减速时间 (Tacc, Tdec) 各自独立设定。执行此函数后，程序可马上执行下一个函数，不需等到运动结束。

c154_start_ta_move：

依据 T 型速度曲线，控制运动轴由 StrVel 加速至 MaxVel，持续以 MaxVel 速度运转，接着减速停止至特定绝对位置；加减速时间 (Tacc, Tdec) 各自独立设定。执行此函数后，程序可马上执行下一个函数，不需等到运动结束。

c154_start_sr_move：

依据 S 型速度曲线，控制运动轴由 StrVel 加速至 MaxVel，持续以 MaxVel 速度运转，接着减速停止至特定相对距离；加减速时间 (Tacc, Tdec) 各自独立设定。执行此函数后，程序可马上执行下一个函数，不需等到运动结束。

c154_start_sa_move：

依据 T 型速度曲线，控制运动轴由 StrVel 加速至 MaxVel，持续以 MaxVel 速度运转，接着减速停止至特定绝对位置；加减速时间 (Tacc, Tdec) 各自独立设定。执行此函数后，程序可马上执行下一个函数，不需等到运动结束。

c154_set_move_ratio：

设定比例参数，当机构分辨率不同时可利用此函数设定比例；例如，若反馈编码器分辨率为控制命令的 2 倍，则设定参数 “move_ratio” 为 2。

@ 语法

C/C++(Windows XP/7/8.1)

I16 c154_start_tr_move(I16 AxisNo, F64 Dist, F64 StrVel, F64 MaxVel, F64 Tacc, F64 Tdec);

I16 c154_start_ta_move(I16 AxisNo, F64 Pos, F64 StrVel, F64 MaxVel, F64 Tacc, F64 Tdec);

I16 c154_start_sr_move(I16 AxisNo, F64 Dist, F64 StrVel, F64 MaxVel, F64 Tacc, F64 Tdec, F64 SVacc, F64 SVdec);

I16 c154_start_sa_move(I16 AxisNo, F64 Pos, F64 StrVel, F64 MaxVel, F64 Tacc, F64 Tdec, F64 SVacc, F64 SVdec);

I16 c154_set_move_ratio(I16 AxisNo, F64 move_ratio);

Visual Basic6 (Windows XP/7/8.1)

c154_start_tr_move(ByVal AxisNo As Integer, ByVal Dist As Double, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double) As Integer

c154_start_ta_move(ByVal AxisNo As Integer, ByVal Pos As Double, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double) As Integer

c154_start_sr_move(ByVal AxisNo As Integer, ByVal Dist As Double, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double, ByVal SVacc As Double, ByVal SVdec As Double) As Integer

c154_start_sa_move(ByVal AxisNo As Integer, ByVal Pos As Double, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double, ByVal SVacc As Double, ByVal SVdec As Double) As Integer

c154_set_move_ratio(ByVal AxisNo As Integer, ByVal move_ratio As Double) As Integer

@ 参数

AxisNo：运动轴编号

card_id	Physical axis	AxisNo
0	0	0
	1	1
	2	2
	3	3
1	0	4
	1	5
	...	...

Dist：运动相对距离 (Pulse)

Pos：运动绝对位置 (Pulse)

StrVel：起始速度 (PPS)

MaxVel：最大速度 (PPS)

Tacc：加速时间 (S)

Tdec：减速时间 (S)

SVacc：加速端的 S 曲线参数

注：SVacc = 0 为纯 S 曲线，详细内容请参阅 4.2.4 节

SVdec：减速端的 S 曲线参数

注：SVacc = 0 为纯 S 曲线，详细内容请参阅 4.2.4 节

Move_ratio：反馈输入与命令输出分辨率比例 (feedback resolution)/(command resolution)，此值不可为 0

G 多轴直线插补运动 (Linear Interpolated Motion)

@ 名称

c154_start_tr_move_xy – 依 T 型速度曲线进行 XY 两轴相对距离直线插补运动

c154_start_ta_move_xy – 依 T 型速度曲线进行 XY 两轴绝对位置直线插补运动

c154_start_sr_move_xy – 依 S 型速度曲线进行 XY 两轴相对距离直线插补运动

c154_start_sa_move_xy – 依 S 型速度曲线进行 XY 两轴绝对位置直线插补运动

c154_start_tr_move_zu – 依 T 型速度曲线进行 ZU 两轴相对距离直线插补运动

c154_start_ta_move_zu – 依 T 型速度曲线进行 ZU 两轴绝对位置直线插补运动

c154_start_sr_move_zu – 依 S 型速度曲线进行 ZU 两轴相对距离直线插补运动

c154_start_sa_move_zu – 依 S 型速度曲线进行 ZU 两轴绝对位置直线插补运动

c154_start_tr_line2 – 依 T 型速度曲线进行任意两轴相对距离直线插补运动

c154_start_ta_line2 – 依 T 型速度曲线进行任意两轴绝对位置直线插补运动

c154_start_sr_line2 – 依 S 型速度曲线进行任意两轴相对距离直线插补运动

c154_start_sa_line2 – 依 S 型速度曲线进行任意两轴绝对位置直线插补运动

c154_start_tr_line3 – 依 T 型速度曲线进行任意三轴相对距离直线插补运动

c154_start_ta_line3 – 依 T 型速度曲线进行任意三轴绝对位置直线插补运动

c154_start_sr_line3 – 依 S 型速度曲线进行任意三轴相对距离直线插补运动

c154_start_sa_line3 – 依 S 型速度曲线进行任意三轴绝对位置直线插补运动

c154_start_tr_line4 – 依 T 型速度曲线进行四轴相对距离直线插补运动

c154_start_ta_line4 – 依 T 型速度曲线进行四轴绝对位置直线插补运动

c154_start_sr_line4 – 依 S 型速度曲线进行四轴相对距离直线插补运动

c154_start_sa_line4 – 依 S 型速度曲线进行四轴绝对位置直线插补运动

@ 描述

这些函示将依据 T 型或 S 型速度曲线执行直线插补运动。

详细比较请参阅下表：

Function	Total axes	Velocity Profile	Relative Absolute	Target Axes
c154_start_tr_move_xy	2	T	R	Axes 0 & 1
c154_start_ta_move_xy	2	T	A	Axes 0 & 1
c154_start_sr_move_xy	2	S	R	Axes 0 & 1
c154_start_sa_move_xy	2	S	A	Axes 0 & 1

Function	Total axes	Velocity Profile	Relative Absolute	Target Axes
c154_start_tr_move_zu	2	T	R	Axes 2 & 3
c154_start_ta_move_zu	2	T	A	Axes 2 & 3
c154_start_sr_move_zu	2	S	R	Axes 2 & 3
c154_start_sa_move_zu	2	S	A	Axes 2 & 3
c154_start_tr_line2	2	T	R	Any 2 of 4 axes
c154_start_ta_line2	2	T	A	Any 2 of 4 axes
c154_start_sr_line2	2	S	R	Any 2 of 4 axes
c154_start_sa_line2	2	S	A	Any 2 of 4 axes

注：执行直线插补运动的 2 轴为同一张板卡上 4 轴的任意 2 轴

Function	Total axes	Velocity Profile	Relative Absolute	Target Axes
c154_start_tr_line3	3	T	R	Any 3 of 4 axes
c154_start_ta_line3	3	T	A	Any 3 of 4 axes
c154_start_sr_line3	3	S	R	Any 3 of 4 axes
c154_start_sa_line3	3	S	A	Any 3 of 4 axes

注：执行直线插补运动的 3 轴为同一张板卡上 4 轴的任意 3 轴

Function	Total axes	Velocity Profile	Relative Absolute	Target Axes
c154_start_tr_line4	4	T	R	Any 4 of 4 axes
c154_start_ta_line4	4	T	A	Any 4 of 4 axes
c154_start_sr_line4	4	S	R	Any 4 of 4 axes
c154_start_sa_line4	4	S	A	Any 4 of 4 axes

注：执行直线插补运动的 4 轴为同一张板卡上 4 轴

Velocity profile：

T：T 型速度曲线

S：S 型速度曲线

Relative / Absolute：

R：相对距离运动

A：绝对位置运动

@ 语法

C/C++(Windows XP/7/8.1)

I16 c154_start_tr_move_xy(I16 Card_id, F64 DistX, F64 DistY, F64 StrVel, F64 MaxVel, F64 Tacc, F64 Tdec);

I16 c154_start_ta_move_xy(I16 Card_id, F64 PosX, F64 PosY, F64 StrVel, F64 MaxVel, F64 Tacc, F64 Tdec);

I16 c154_start_sr_move_xy(I16 Card_id, F64 DistX, F64 DistY, F64 StrVel, F64 MaxVel, F64 Tacc, F64 Tdec, F64 SVacc, F64 SVdec);

I16 c154_start_sa_move_xy(I16 Card_id, F64 PosX, F64 PosY, F64 StrVel, F64 MaxVel, F64 Tacc, F64 Tdec, F64 SVacc, F64 SVdec);

I16 c154_start_tr_move_zu(I16 Card_id, F64 DistX, F64 DistY, F64 StrVel, F64 MaxVel, F64 Tacc, F64 Tdec);

I16 c154_start_ta_move_zu(I16 Card_id, F64 PosX, F64 PosY, F64 StrVel, F64 MaxVel, F64 Tacc, F64 Tdec);

I16 c154_start_sr_move_zu(I16 Card_id, F64 DistX, F64 DistY, F64 StrVel, F64 MaxVel, F64 Tacc, F64 Tdec, F64 SVacc, F64 SVdec);

I16 c154_start_sa_move_zu(I16 Card_id, F64 PosX, F64 PosY, F64 StrVel, F64 MaxVel, F64 Tacc, F64 Tdec, F64 SVacc, F64 SVdec);

I16 c154_start_tr_line2(I16 *AxisArray, F64 *DistArray, F64 StrVel, F64 MaxVel, F64 Tacc, F64 Tdec);

I16 c154_start_ta_line2(I16 *AxisArray, F64 *PosArray, F64 StrVel, F64 MaxVel, F64 Tacc, F64 Tdec);

I16 c154_start_sr_line2(I16 *AxisArray, F64 *DistArray, F64 StrVel, F64 MaxVel, F64 Tacc, F64 Tdec, F64 SVacc, F64 SVdec);

I16 c154_start_sa_line2(I16 *AxisArray, F64 *PosArray, F64 StrVel, F64 MaxVel, F64 Tacc, F64 Tdec, F64 SVacc, F64 SVdec);

I16 c154_start_tr_line3(I16 *AxisArray, F64 *DistArray, F64 StrVel, F64 MaxVel, F64 Tacc, F64 Tdec);

*I16 c154_start_ta_line3(I16 *AxisArray, F64 *PosArray, F64 StrVel, F64 MaxVel, F64 Tacc,*

F64 Tdec);

*I16 c154_start_sr_line3(I16 *AxisArray, F64 *DistArray, F64 StrVel, F64 MaxVel, F64*

Tacc, F64 Tdec, F64 SVacc, F64 SVdec);

*I16 c154_start_sa_line3(I16 *AxisArray, F64 *PosArray, F64 StrVel, F64 MaxVel, F64 Tacc,*

F64 Tdec, F64 SVacc, F64 SVdec);

*I16 c154_start_tr_line4(I16 *AxisArray, F64 *DistArray, F64 StrVel, F64 MaxVel, F64 Tacc, F64 Tdec);*

*I16 c154_start_ta_line4(I16 *AxisArray, F64 *PosArray, F64 StrVel, F64 MaxVel, F64 Tacc, F64 Tdec);*

*I16 c154_start_sr_line4(I16 *AxisArray, F64 *DistArray, F64 StrVel, F64 MaxVel, F64 Tacc, F64 Tdec, F64 SVacc, F64 SVdec);*

*I16 c154_start_sa_line4(I16 *AxisArray, F64 *PosArray, F64 StrVel, F64 MaxVel, F64 Tacc, F64 Tdec, F64 SVacc, F64 SVdec);*

Visual Basic6 (Windows XP/7/8.1)

c154_start_tr_move_xy(ByVal Card_id As Integer, ByVal DistX As Double, ByVal DistY As Double, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double) As Integer

c154_start_ta_move_xy(ByVal Card_id As Integer, ByVal PosX As Double, ByVal PosY As Double, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double) As Integer

c154_start_sr_move_xy(ByVal Card_id As Integer, ByVal DistX As Double, ByVal DistY As Double, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double, ByVal SVacc As Double, ByVal SVdec As Double) As Integer

c154_start_sa_move_xy(ByVal Card_id As Integer, ByVal PosX As Double, ByVal PosY As Double, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double, ByVal SVacc As Double, ByVal SVdec As Double) As Integer

c154_start_tr_move_zu(ByVal Card_id As Integer, ByVal DistX As Double, ByVal DistY As Double, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double) As Integer

c154_start_ta_move_zu(ByVal Card_id As Integer, ByVal PosX As Double, ByVal PosY As Double, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double) As Integer

c154_start_sr_move_zu(ByVal Card_id As Integer, ByVal DistX As Double, ByVal DistY As Double, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double, ByVal SVacc As Double, ByVal SVdec As Double) As Integer

c154_start_sa_move_zu(ByVal Card_id As Integer, ByVal PosX As Double, ByVal PosY As Double, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double, ByVal SVacc As Double, ByVal SVdec As Double) As Integer

c154_start_tr_line2(AxisArray() As Integer, DistArray() As Double, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double) As Integer

c154_start_ta_line2(AxisArray() As Integer, PosArray() As Double, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double) As Integer

c154_start_sr_line2((AxisArray() As Integer, DistArray() As Double, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double, ByVal Svacc As Double, ByVal Svdec As Double) As Integer

c154_start_sa_line2(AxisArray() As Integer, PosArray() As Double, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double, ByVal Svacc As Double, ByVal Svdec As Double) As Integer

c154_start_tr_line3(AxisArray() As Integer, DistArray() As Double, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double) As Integer

c154_start_ta_line3(AxisArray() As Integer, PosArray() As Double, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double) As Integer

c154_start_sr_line3((AxisArray() As Integer, DistArray() As Double, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double, ByVal Svacc As Double, ByVal Svdec As Double) As Integer

c154_start_sa_line3(AxisArray() As Integer, PosArray() As Double, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double, ByVal Svacc As Double, ByVal Svdec As Double) As Integer

c154_start_tr_line4(AxisArray() As Integer, DistArray() As Double, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double) As Integer

c154_start_ta_line4(AxisArray() As Integer, PosArray() As Double, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double) As Integer

c154_start_sr_line4((AxisArray() As Integer, DistArray() As Double, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double, ByVal Svacc As Double, ByVal Svdec As Double) As Integer

c154_start_sa_line4(AxisArray() As Integer, PosArray() As Double, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double, ByVal Svacc As Double, ByVal Svdec As Double) As Integer

@ 参数

card_id: PCI-C154+ 卡片编号，卡片编号可由卡片编码开关 (SW1) 或 PCI 插槽顺序决定，请参阅 c154_initial()。

AxisNo：运动轴编号

card_id	Physical axis	AxisNo
0	0	0
	1	1
	2	2
	3	3
1	0	4
	1	5
	...	...

DistX：轴 0 运动相对距离 (Pulse)

DistY：轴 1 运动相对距离 (Pulse)

PosX：轴 0 运动绝对位置 (Pulse)

PosY：轴 1 运动绝对位置 (Pulse)

StrVel：起始速度 (PPS)

MaxVel：最大速度 (PPS)

Tacc：加速度时间 (S)

Tdec：减速度时间 (S)

SVacc：加速端的 S 曲线参数

注：SVacc = 0 为纯 S 曲线，详细内容请参阅 4.2.4 节

SVdec：减速端的 S 曲线参数

注：SVacc = 0 为纯 S 曲线，详细内容请参阅 4.2.4 节

***AxisArray**：执行插补运动的运动轴数组

例如：

I16 AxisArray[2] = {0, 3}; //axis 0, & axis 3 (correct)

I16 AxisArray[3] = {0,2, 3}; //axis 0, 2 & 3 (correct)

I16 AxisArray[2] = {1, 6}; //axis 1, & axis 6 (incorrect)

***DistArray**：直线插补运动的相对距离数组

例如：

I16 AxisArray[2] = {0, 3}; //axis 0, & axis 3

F64 DistArray[2] = {1000.0, 2000.0} //for axis 0 & 3

***PosArray**：直线插补运动的绝对位置数组

例如：

I16 AxisArray[3] = {0,2, 3}; //axis 0, 2 & 3

F64 PosArray[3] = {200.0, 300.0, 400.0} // absolute position for axis 0, 2 & 3

H 两轴圆弧插补运动 (Circular Interpolation Motion)

@ 名称

c154_start_tr_arc_xy – 依 T 型速度曲线进行 XY 两轴相对距离圆弧插补运动

c154_start_ta_arc_xy – 依 T 型速度曲线进行 XY 两轴绝对位置圆弧插补运动

c154_start_sr_arc_xy – 依 S 型速度曲线进行 XY 两轴相对距离圆弧插补运动

c154_start_sa_arc_xy – 依 S 型速度曲线进行 XY 两轴绝对位置圆弧插补运动

c154_start_tr_arc_zu – 依 T 型速度曲线进行 ZU 两轴相对距离圆弧插补运动

c154_start_ta_arc_zu – 依 T 型速度曲线进行 ZU 两轴绝对位置圆弧插补运动

c154_start_sr_arc_zu – 依 S 型速度曲线进行 ZU 两轴相对距离圆弧插补运动

c154_start_sa_arc_zu – 依 S 型速度曲线进行 ZU 两轴绝对位置圆弧插补运动

c154_start_tr_arc2 – 依 T 型速度曲线进行任意两轴相对距离圆弧插补运动

c154_start_ta_arc2 – 依 T 型速度曲线进行任意两轴绝对位置圆弧插补运动

c154_start_sr_arc2 – 依 S 型速度曲线进行任意两轴相对距离圆弧插补运动

c154_start_sa_arc2 – 依 S 型速度曲线进行任意两轴绝对位置圆弧插补运动

@ 描述

这些函示将依据 T 型或 S 型速度曲线执行圆弧插补运动。

详细比较请参阅下表：

Function	Total axes	Velocity Profile	Relative Absolute	Target Axes
c154_start_tr_arc_xy	2	trapezoidal	R	Axes 0 & 1
c154_start_ta_arc_xy	2	trapezoidal	A	Axes 0 & 1
c154_start_sr_arc_xy	2	S-curve	R	Axes 0 & 1
c154_start_sa_arc_xy	2	S-curve	A	Axes 0 & 1
c154_start_tr_arc_zu	2	trapezoidal	R	Axes 2 & 3
c154_start_ta_arc_zu	2	trapezoidal	A	Axes 2 & 3
c154_start_sr_arc_zu	2	S-curve	R	Axes 2 & 3
c154_start_sa_arc_zu	2	S-curve	A	Axes 2 & 3
c154_start_tr_arc2	2	trapezoidal	R	Any 2 of 4 Axis
c154_start_ta_arc2	2	trapezoidal	A	Any 2 of 4 Axis
c154_start_sr_arc2	2	S-curve	R	Any 2 of 4 Axis
c154_start_sa_arc2	2	S-curve	A	Any 2 of 4 Axis

@ 语法

C/C++(Windows XP/7/8.1)

I16 c154_start_tr_arc_xy(I16 card_id, F64 OffsetCx, F64 OffsetCy, F64 OffsetEx, F64 OffsetEy, I16 CW_CCW, F64 StrVel,F64 MaxVel,F64 Tacc,F64 Tdec);

I16 c154_start_ta_arc_xy(I16 card_id, F64 Cx, F64 Cy, F64 Ex, F64 Ey, I16 CW_CCW, F64 StrVel,F64 MaxVel,F64 Tacc,F64 Tdec);

I16 c154_start_sr_arc_xy(I16 card_id, F64 OffsetCx, F64 OffsetCy, F64 OffsetEx, F64 OffsetEy, I16 CW_CCW, F64 StrVel,F64 MaxVel,F64 Tacc,F64 Tdec,F64 SVacc,F64 SVdec);

I16 c154_start_sa_arc_xy(I16 card_id, F64 Cx, F64 Cy, F64 Ex, F64 Ey, I16 CW_CCW, F64 StrVel,F64 MaxVel,F64 Tacc,F64 Tdec,F64 SVacc,F64 SVdec);

I16 c154_start_tr_arc_zu(I16 card_id, F64 OffsetCx, F64 OffsetCy, F64 OffsetEx, F64 OffsetEy, I16 CW_CCW, F64 StrVel,F64 MaxVel,F64 Tacc,F64 Tdec);

I16 c154_start_ta_arc_zu(I16 card_id, F64 Cx, F64 Cy, F64 Ex, F64 Ey, I16 CW_CCW, F64 StrVel,F64 MaxVel,F64 Tacc,F64 Tdec);

I16 c154_start_sr_arc_zu(I16 card_id, F64 OffsetCx, F64 OffsetCy, F64 OffsetEx, F64 OffsetEy, I16 CW_CCW, F64 StrVel,F64 MaxVel,F64 Tacc,F64 Tdec,F64 SVacc,F64 SVdec);

I16 c154_start_sa_arc_zu(I16 card_id, F64 Cx, F64 Cy, F64 Ex, F64 Ey, I16 CW_CCW, F64 StrVel,F64 MaxVel,F64 Tacc,F64 Tdec,F64 SVacc,F64 SVdec);

I16 c154_start_tr_arc2(I16 *AxisArray, F64 *OffsetCenter, F64 *OffsetEnd, I16 CW_CCW, F64 StrVel,F64 MaxVel,F64 Tacc,F64 Tdec);

I16 c154_start_ta_arc2(I16 *AxisArray, F64 *CenterPos, F64 *EndPos, I16 CW_CCW, F64 StrVel,F64 MaxVel,F64 Tacc,F64 Tdec);

I16 c154_start_sr_arc2(I16 *AxisArray, F64 *OffsetCenter, F64 *OffsetEnd, I16 CW_CCW, F64 StrVel,F64 MaxVel,F64 Tacc,F64 Tdec, F64 SVacc,F64 SVdec);

I16 c154_start_sa_arc2(I16 *AxisArray, F64 *CenterPos, F64 *EndPos, I16 CW_CCW, F64 StrVel,F64 MaxVel, F64 Tacc, F64 Tdec, F64 SVacc, F64 SVdec);

Visual Basic6 (Windows XP/7/8.1)

c154_start_tr_arc_xy(ByVal card_id As Integer, ByVal OffsetCx As Double, ByVal OffsetCy As Double, ByVal OffsetEx As Double, ByVal OffsetEy As Double, ByVal CW_CCW As Integer, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double);

c154_start_ta_arc_xy(ByVal card_id As Integer, ByVal Cx As Double, ByVal Cy As Double, ByVal Ex As Double, ByVal Ey As Double, ByVal CW_CCW As Integer, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double) As Integer

c154_start_sr_arc_xy(ByVal card_id As Integer, ByVal OffsetCx As Double, ByVal OffsetCy As Double, ByVal OffsetEx As Double, ByVal OffsetEy As Double, ByVal CW_CCW As Integer, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double, ByVal Svacc As Double, ByVal Svdec As Double) As Integer

c154_start_sa_arc_xy(ByVal card_id As Integer, ByVal Cx As Double, ByVal Cy As Double, ByVal Ex As Double, ByVal Ey As Double, ByVal CW_CCW As Integer, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double, ByVal Svacc As Double, ByVal Svdec As Double) As Integer

c154_start_tr_arc_zu(ByVal card_id As Integer, ByVal OffsetCx As Double, ByVal OffsetCy As Double, ByVal OffsetEx As Double, ByVal OffsetEy As Double, ByVal CW_CCW As Integer, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double) As Integer

c154_start_ta_arc_zu(ByVal card_id As Integer, ByVal Cx As Double, ByVal Cy As Double, ByVal Ex As Double, ByVal Ey As Double, ByVal CW_CCW As Integer, ByVal StrVel As Double, ByVal MaxVel As

Double, ByVal Tacc As Double, ByVal Tdec As Double) As Integer

c154_start_sr_arc_zu(ByVal card_id As Integer, ByVal OffsetCx As Double, ByVal OffsetCy As Double, ByVal OffsetEx As Double, ByVal OffsetEy As Double, ByVal CW_CCW As Integer, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double, ByVal Svacc As Double, ByVal Svdec As Double) As Integer

c154_start_sa_arc_zu(ByVal card_id As Integer, ByVal Cx As Double, ByVal Cy As Double, ByVal Ex As Double, ByVal Ey As Double, ByVal CW_CCW As Integer, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double, ByVal Svacc As Double, ByVal Svdec As Double) As Integer

c154_start_tr_arc2(AxisArray() As Integer, OffsetCenter() As Double, OffsetEnd() As Double, Byval CW_CCW As Integer, ByVal StrVel As Double , ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double) As Integer

c154_start_ta_arc2(AxisArray() As Integer, CenterPos() As Double, EndPos() As Double, Byval CW_CCW As Integer, ByVal StrVel As Double , ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double) As Integer

c154_start_sr_arc2(AxisArray() As Integer, OffsetCenter() As Double, OffsetEnd() As Double, Byval CW_CCW As Integer, ByVal StrVel As Double , ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double, ByVal Svacc As Double, ByVal Svdec As Double) As Integer

c154_start_sa_arc2(AxisArray() As Integer, CenterPos() As Double, EndPos() As Double, Byval CW_CCW As Integer, ByVal StrVel As Double , ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double,

ByVal Svacc As Double, ByVal Svdec As Double) As Integer

@ 参数

card_id: PCI-C154+ 卡片编号，卡片编号可由卡片编码开关 (SW1) 或 PCI 插槽顺序决定，请参阅 c154_initial()。

AxisNo：运动轴编号

card_id	Physical axis	AxisNo
0	0	0
	1	1
	2	2
	3	3
1	0	4
	1	5
	...	...

OffsetCx：X 轴圆心相对距离

OffsetCy：Y 轴圆心相对距离

OffsetEx：X 轴终点相对距离

OffsetEy：Y 轴终点相对距离

Cx：X 轴圆心绝对位置

Cy：Y 轴圆心绝对位置

Ex：X 轴终点绝对位置

Ey：Y 轴终点绝对位置

CW_CCW：圆弧插补运动方向

Value	Meaning
0	Clockwise(cw)
1	Counterclockwise(ccw)

StrVel：起始速度 (PPS)

MaxVel：最大速度 (PPS)

Tacc：加速度时间 (S)

Tdec：减速度时间 (S)

SVacc：加速端的 S 曲线参数

注：SVacc = 0 为纯 S 曲线，详细内容请参阅 4.2.4 节

SVdec：减速端的 S 曲线参数

注：SVacc = 0 为纯 S 曲线，详细内容请参阅 4.2.4 节

***AxisArray**：执行插补运动的运动轴数组

例如：

I16 AxisArray[2] = {0, 3}; //axis 0, & axis 3 (correct)

I16 AxisArray[2] = {1, 6}; //axis 1, & axis 6 (incorrect)

***OffsetCenter**：圆心的相对距离数组 (相对于起始点)

例如：

F64 OffsetCenter[2] = {2000.0, 0.0}; //offset from start position(initial point) for 1st & 2nd axes

***OffsetEnd**：终点的相对距离数组 (相对于起始点)

例如：

F64 OffsetEnd[2] = {4000.0, 0.0}; //offset from start position(initial point for 1st & 2nd axes

***CenterPos**：圆心的绝对位置数组

例如：

F64 CenterPos[2] = {2000.0, 0.0}; //absolute center position for 1st & 2nd axes

***EndPos**：终点的绝对位置数组

例如：

F64 EndPos[2] = {4000.0, 0.0}; //absolute end position for 1st & 2nd axes

I 归零运动 (Home Return Mode)

@ 名称

c154_set_home_config – 设定归零运动模式与信号逻辑

c154_home_move – 进行归零运动

c154_home_search – 进行零点位置搜寻

@ 描述

c154_set_home_config：

设定归零运动模式、ORG 逻辑、EZ 逻辑、EZ 计数次数以及 ERC 输出选项；详细请参阅 4.2.10。

c154_home_move：

执行此函数，将会开始依据 c154_set_home_config() 所设定做归零运动。运动方向由 MaxVel 的正负符号决定，停止行为由参数 home_mode 决定。使用者需特别注意运动方向，或当触及限位开关时所对应行为，以及所有会导致运动轴停止之条件，详细请参阅 4.2.10 节。

c154_home_search：

执行此函数，将会开始依据 c154_set_home_config() 所设定做零点位置搜寻运动。运动方向由 MaxVel 的正负符号决定，停止行为由参数 home_mode 决定。使用者需特别注意运动方向，或当触及限位开关时所对应行为，以及所有会导致运动轴停止之条件，详细请参阅 4.2.11 节。

@ 语法

C/C++(Windows XP/7/8.1)

I16 c154_set_home_config(I16 AxisNo, I16 home_mode, I16 org_logic, I16 ez_logic, I16 ez_count, I16 erc_out);

I16 c154_home_move(I16 AxisNo, F64 StrVel, F64 MaxVel, F64 Tacc);

I16 c154_home_search(I16 AxisNo, F64 StrVel, F64 MaxVel, F64 Tacc, F64 ORGOffset);

Visual Basic (Windows XP/7/8.1)

c154_set_home_config(ByVal AxisNo As Integer, ByVal home_mode As Integer, ByVal org_logic As Integer, ByVal ez_logic As Integer, ByVal ez_count As Integer, ByVal erc_out As Integer) As Integer

c154_home_move(ByVal AxisNo As Integer, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double) As Integer

c154_home_search(ByVal AxisNo As Integer, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal ORGOffset As Double) As Integer

@ 参数

AxisNo：运动轴编号

card_id	Physical axis	AxisNo
0	0	0
	1	1
	2	2
	3	3
1	0	4
	1	5
	...	...

home_mode：归零运动停止模式，可设定 0 至 12，详细请参阅 4.2.10 节

org_logic：ORG 逻辑

Value	Meaning
0	Active low
1	Active high

ez_logic：EZ 逻辑

Value	Meaning
0	Active low
1	Active high

ez_count：0-15，请参阅 4.2.10 节

erc_out：ERC 输出模式

Value	Meaning
0	no ERC out
1	ERC signal out when home-move finishing

StrVel：起始速度 (PPS)

MaxVel：最大速度 (PPS)

Tacc：加速度时间 (S)

ORGOffset：当接触 ORG 信号后，移动位移量 (pulse)，详细请参阅 4.2.11 节

J 脉冲发生器运动模式 (Manual Pulser Motion)

@ 名称

c154_disable_pulser_input – 关闭脉冲发生器输入信号

c154_pulser_pmove – 开始藉由脉冲发生器进行位置模式运动

c154_pulser_vmove – 开始藉由脉冲发生器进行速度模式运动

c154_set_pulser_ratio – 设定脉冲发生器输入与脉冲控制命令输出比例

c154_set_pulser_iptmode – 设定脉冲发生器输入模式

@ 描述

c154_disable_pulser_input：

设定是否启动脉冲发生器输入

c154_pulser_pmove：

下达此函数命令，运动轴将会依据脉冲发生器输入之脉冲开始运动，直到下达 c154_disable_pulser_input() 关闭脉冲发生器输入，或运动轴到达设定距离，才会停止运动。

c154_pulser_vmove：

下达此函数命令，运动轴将会依据脉冲发生器输入之脉冲开始运动，直到下达 c154_disable_pulser_input() 关闭脉冲发生器输入，才会停止运动。

c154_set_pulser_ratio：

设定脉冲发生器输入与脉冲控制命令输出比例，公式如下：

Output Pulse Count = Input Pulser Count x (MultiF + 1) x DivF /2048

The DivF = 1~2047 Divide Factor

The MultiF= 0~31 Multiplication Factor

c154_set_pulser_iptmode：

设定脉冲发生器输入模式。

@ 语法

C/C++(Windows XP/7/8.1)

I16 c154_disable_pulser_input(I16 AxisNo, U16 Disable);

I16 c154_pulser_pmove(I16 AxisNo, F64 Dist, F64 SpeedLimit);

I16 c154_pulser_vmove(I16 AxisNo, F64 SpeedLimit);

I16 c154_set_pulser_ratio(I16 AxisNo, I16 DivF, I16 MultiF);

I16 c154_set_pulser_iptmode(I16 AxisNo, I16 InputMode, I16 Inverse);

Visual Basic (Windows XP/7/8.1)

c154_disable_pulser_input(ByVal AxisNo As Integer, ByVal Disable As Integer) As Integer

c154_pulser_pmove(ByVal AxisNo As Integer, ByVal Dist As Double, ByVal SpeedLimit As Double) As Integer

c154_pulser_vmove(ByVal AxisNo As Integer, ByVal SpeedLimit As Double) As Integer

c154_set_pulser_ratio(ByVal AxisNo As Integer, ByVal DivF As Integer, ByVal MultiF As Integer) As Integer

c154_set_pulser_iptmode(ByVal AxisNo As Integer, ByVal InputMode As Integer, ByVal Inverse As Integer) As Integer

@ 参数

AxisNo：运动轴编号

card_id	Physical axis	AxisNo
0	0	0
	1	1
	2	2
	3	3

card_id	Physical axis	AxisNo
1	0	4
	1	5
	...	...

Disable：关闭脉冲产生器输入

Disable = 1, disable pulser

Disable = 0, enable pulser

Dist：运动相对距离 (Pulse)

SpeedLimit：速度限制 (PPS)

例如，若SpeedLimit为100PPS，则运动轴最快可旋转100PPS，即使脉冲发生器输入速度大于 100PPS。

DivF：除法因子 (1-2047)

注：若输入 0 或 2048，则除法电路将 OFF。

MultiF：乘法因子 (0-31)

InputMode：脉冲发生器输入模式 (PA 与 PB)

Value	Meaning
0	1X AB phase type pulse input
1	2X AB phase type pulse input
2	4X AB phase type pulse input
3	CW/CCW type pulse input

Inverse：设定运动方向是否与脉冲发生器方向相同

Value	Meaning
0	no inverse
1	Reverse moving direction

K 运动状态 (Motion Status)

@ 名称

c154_motion_done – 读取 PCI-C154+ 运动状态

@ 描述

c154_motion_done：

读取 PCI-C154+ 运动状态，回传值内容如下：

0	Normal stopped condition
1	Waiting for DR
2	Waiting for CSTA input
3	Waiting for an internal synchronous signal
4	Waiting for another axis to stop
5	Waiting for a completion of ERC timer
6	Waiting for a completion of direction change timer
7	Correcting backlash
8	Wait PA/PB
9	At FA speed
10	At FL Speed
11	Accelerating
12	At FH Speed
13	Decelerating
14	Wait INP
15	Others(Controlling Start)
16	SALM
17	SPEL
18	SMEL
19	SEMG
20	SSTP
21	SERC

@ 语法

C/C++(Windows XP/7/8.1)

I16 c154_motion_done(I16 AxisNo)

Visual Basic (Windows XP/7/8.1)

c154_motion_done(ByVal AxisNo As Integer) As Integer

@ 参数

AxisNo：运动轴编号

card_id	Physical axis	AxisNo
0	0	0
	1	1
	2	2
	3	3
1	0	4
	1	5
	...	...

L 运动界面 I/O(Motion Interface I/O)

@ 名称

c154_set_servo – 设定 SVON 信号 ON/OFF

c154_set_clr_mode – 设定 CLR 模式

c154_set_inp – 设定 INP 逻辑与操作模式

c154_set_alm – 设定 ALM 逻辑与操作模式

c154_set_erc – 设定 ERC 逻辑与时序

c154_set_erc_out – 手动输出 ERC

c154_clr_erc – 清除 ERC 信号

c154_set_sd – 设定 SD 逻辑与操作模式

c154_enable_sd – 启动 SD 信号

c154_set_limit_logic – 设定 PEL/MEL 逻辑

c154_set_limit_mode – 设定 PEL/MEL 操作模式

c154_get_io_status – 读取 PCI-C154+ 运动界面 I/O 状态

@ 描述

c154_set_servo：

设定 SVON 信号 ON 或 OFF，默认值为 1(OFF)(与 GND 断路)。

c154_set_clr_mode：

CLR 输入信号可重置计数器 (命令、反馈、误差、通用计数器)，此函数可设定重置模式，有 4 种模式可供设定。

c154_set_inp：

设定 INP 逻辑，使用这亦可透过此函数决定是否启动 INP 功能，默认为关闭。

c154_set_alm：

设定 ALARM 逻辑，可设定当收到 ALARM 信号后的反应动作，共有 2 种模式。

c154_set_erc：

设定 ERC 逻辑、模式以及脉冲宽度。

c154_set_erc_out：

手动输出 ERC 信号。

c154_clr_erc：

重置 ERC 信号，当 ERC 为 Level 模式输出时。

c154_set_sd：

设定 SD 逻辑、撷取与操作模式，用户可藉由 c154_enable_sd() 来决定是否启动此功能，默认值为关闭。

c154_enable_sd：

启动 SD 信号输入，默认为关闭。

c154_set_limit_logic：

设定 EL 逻辑。

c154_set_limit_mode：

设定当 EL 信号触发时，运动停止模式。

c154_get_io_status：

读取各轴 I/O 状态，定义如下：

Bit	Name	Description
0	RDY	RDY pin input
1	ALM	Alarm Signal
2	+EL	Positive Limit Switch
3	-EL	Negative Limit Switch
4	ORG	Origin Switch
5	DIR	DIR output

Bit	Name	Description
6	EMG	EMG status
7		
8	ERC	ERC pin output
9	EZ	Index signal
10	CLR	Clear signal
11		
12	SD	Slow Down signal input
13	INP	In-Position signal input
14	SVON	Servo-ON output status

@ 语法

C/C++(Windows XP/7/8.1)

I16 c154_set_servo(I16 AxisNo, I16 on_off);

I16 c154_set_clr_mode(I16 AxisNo, I16 clr_mode, I16 targetCounterInBit);

I16 c154_set_inp(I16 AxisNo, I16 inp_enable, I16 inp_logic);

I16 c154_set_alm(I16 AxisNo, I16 alm_logic, I16 alm_mode);

I16 c154_set_erc(I16 AxisNo, I16 erc_logic, I16 erc_pulse_width, I16 erc_mode);

I16 c154_set_erc_out(I16 AxisNo);

I16 c154_clr_erc(I16 AxisNo);

I16 c154_set_sd(I16 AxisNo, I16 sd_logic, I16 sd_latch, I16 sd_mode);

I16 c154_enable_sd(I16 AxisNo, I16 enable);

I16 c154_set_limit_logic(I16 AxisNo, U16 Logic);

I16 c154_set_limit_mode(I16 AxisNo, I16 limit_mode);

I16 c154_get_io_status(I16 AxisNo, U16 *io_sts);

Visual Basic (Windows XP/7/8.1)

c154_set_servo(ByVal AxisNo As Integer, ByVal on_off As Integer) As Integer

c154_set_clr_mode(ByVal AxisNo As Integer, ByVal clr_mode As Integer, ByBal targetCounterInBit as Integer) As Integer

c154_set_inp(ByVal AxisNo As Integer, ByVal inp_enable As Integer, ByVal inp_logic As Integer) As Integer

c154_set_alm(ByVal AxisNo As Integer, ByVal alm_logic As Integer, ByVal alm_mode As Integer) As Integer

c154_set_erc(ByVal AxisNo As Integer, ByVal erc_logic As Integer, ByVal erc_pulse_width As Integer, ByVal erc_mode As Integer) As Integer

c154_set_erc_out(ByVal AxisNo As Integer) As Integer
c154_clr_erc(ByVal AxisNo As Integer) As Integer

c154_set_sd(ByVal AxisNo As Integer, ByVal sd_logic As Integer, ByVal sd_latch As Integer, ByVal sd_mode As Integer) As Integer

c154_enable_sd(ByVal AxisNo As Integer, ByVal Enable As Integer) As Integer

c154_set_limit_logic(ByVal AxisNo As Integer, ByVal Logic As Integer) As Integer

c154_set_limit_mode(ByVal AxisNo As Integer, ByVal limit_mode As Integer) As Integer

c154_get_io_status(ByVal AxisNo As Integer, io_sts As Integer) As Integer

@ 参数

AxisNo：运动轴编号

card_id	Physical axis	AxisNo
0	0	0
	1	1
	2	2
	3	3

card_id	Physical axis	AxisNo
1	0	4
	1	5
	...	...

on_off：SVON 信号 ON 或 OFF

Value	Meaning
0	ON
1	OFF

clr_mode：CLR 信号输入清除模式

clr_mode = 0 , 于信号降缘 (falling edge) 清除计数器 (预设)

clr_mode = 1 , 于信号升缘 (rising edge) 清除计数器

clr_mode = 2 , 于信号为低位准 (LOW level) 时清除计数器

clr_mode = 3 , 于信号为高位准 (HIGH level) 时清除计数器

targetCounterInBit：启动或关闭清除计数器

Value	Meaning
0	Reset command counter when CLR input turns ON
1	Reset position counter when CLR input turns ON
2	Reset error counter when CLR input turns ON
3	Reset general purpose counter when CLR input turns ON

inp_enable：INP 功能启动或关闭

inp_enable = 0, 关闭 (预设)

inp_enable = 1, 启动

inp_logic：INP 逻辑

Value	Meaning
0	Active low
1	Active high

alm_logic：ALARM 逻辑

Value	Meaning
0	Active low
1	Active high

alm_mode：当 ALARM 信号触发时，运动停止模式

Value	Meaning
0	motor immediately stops (default)
1	motor decelerates then stops

erc_logic：ERC 逻辑

Value	Meaning
0	Active low
1	Active high

erc_pulse_width：ERC 输出脉冲宽度

Value	Meaning
0	12 μs
1	102 μs
2	409 μs
3	1.6 ms
4	13 ms
5	52 ms
6	104 ms
7	Level output

erc_mode：

Value	Meaning
0	Disable
1	Output ERC when stopped by EL, ALM, or EMG input
2	Output ERC when complete home return
3	Both 1 and 2

sd_logic：

Value	Meaning
0	Active low
1	Active high

sd_latch：SD 逻辑

Value	Meaning
0	Do not latch
1	latch

sd_mode：当 SD 信号触发时，运动停止模式

Value	Meaning
0	slow down only
1	Slow down then stop

enable：启动或关闭 SD 功能

Value	Meaning
0	Automatic setting
1	Manual setting (default)

Logic：PEL/MEL 逻辑

Value	Meaning
0	Normal low(normal open)
1	Normal high(normal close)

limit_mode：

Value	Meaning
0	Stop immediately
1	Slow down then stop

***io_sts：**I/O 状态

M 中断控制 (Interrupt Control)

@ 名称

c154_int_control – 启动 / 停止中断功能

c154_set_motion_int_factor – 选择运动中断事件

c154_wait_error_interrupt – 等待错误中断事件

c154_wait_motion_interrupt – 等待运动中断事件

@ 描述

c154_int_control：

启动中断功能。

c154_set_motion_int_factor：

此函数可供使用者选择运动中断源，当中断功能启动后，可执行 c154_wait_motion_interrupt() 等待中断。(错误中断源不可选择)

c154_wait_error_interrupt：

当使用 c154_int_control() 启动中断功能后，可使用此函数等待错误中断。详细请参阅 4.9 节

c154_wait_motion_interrupt：

当使用 c154_int_control() 启动中断功能，且使用 c154_set_motion_int_factor() 设定中断源后，可使用此函数等待特定中断，当此函数执行后，程序将不会停止，直到事件触发或 time out。

@ 语法

C/C++(Windows XP/7/8.1)

I16 c154_int_control(I16 card_id, I16 intFlag);

I16 c154_set_motion_int_factor(I16 AxisNo, U32 int_factor);

I16 c154_wait_error_interrupt(I16 AxisNo, I32 TimeOut_ms);

I16 c154_wait_motion_interrupt(I16 AxisNo, I16 IntFactorBitNo, I32 TimeOut_ms);

Visual Basic (Windows XP/7/8.1)

c154_int_control(ByVal card_id As Integer, ByVal intFlag As Integer) As Integer

c154_wait_error_interrupt(ByVal AxisNo As Integer, ByVal TimeOut_ms As Long) As Integer

c154_wait_motion_interrupt(ByVal AxisNo As Integer, ByVal IntFactorBitNo As Integer, ByVal TimeOut_ms As Long) As Integer

c154_set_motion_int_factor(ByVal AxisNo As Integer, ByVal int_factor As Long) As Integer

@ 参数

card_id: PCI-C154+ 卡片编号，卡片编号可由卡片编码开关 (SW1) 或 PCI 插槽顺序决定，请参阅 c154_initial()。

intFlag：启动或关闭中断功能

Value	Meaning
0	Disable
1	Enable

AxisNo：运动轴编号

card_id	Physical axis	AxisNo
0	0	0
	1	1
	2	2
	3	3
1	0	4
	1	5
	...	...

int_factor：中断源参数

(0：关闭 , 1：启动)

Value	Meaning
0	Normal stop
1	Next command in buffer starts
2	Command pre-register 2 is empty and allow new command to write

Value	Meaning
3	(Reserved) (Always set to 0)
4	Acceleration Start
5	Acceleration End
6	Deceleration Start
7	Deceleration End
8	+Soft limit or comparator 1 is ON
9	-Soft limit or comparator 2 is ON
10	Error comparator or comparator 3 is ON
11	General comparator or comparator 4 is ON
12	Trigger comparator or comparator 5 is ON
13	Counter is reset by CLR input
14	Reserved
15	Counter is latched by ORG Input
16	SD input turns on
17	(Reserved) (Always set to 0)
18	CSTA input or c154_start_move_all() turns on
19~31	Not define (Always set to 0)

TimeOut_ms：Time out 时间 (ms)，若此参数为 0，函数将马上返回，若为 -1， time out 时间将为无限长。

IntFactorBitNo：等待中断源参数

例如：IntFactorBitNo = 4，表示等待 “Acceleration Start” 中断。

N 位置控制与计数器 (Position Control and Counters)

@ 名称

c154_get_position – 读取反馈位置计数器数值

c154_set_position – 设定反馈位置计数器数值

c154_get_command – 读取命令位置计数器数值

c154_set_command – 设定命令位置计数器数值

c154_get_error_counter – 读取位置误差计数器数值

c154_reset_error_counter – 清除位置误差计数器数值

c154_get_general_counter – 读取通用计数器数值

c154_set_general_counter – 设定通用计数器

@ 描述

c154_get_position：

读取反馈位置计数器数值，请注意此值为依据 c154_set_move_ratio() 比例方程式所计算出的结果，若参数 move_ratio 为 0.5，则计数器数值将为实际编码器输入的 2 倍。反馈计数器的信号源由 c154_set_feedback_src() 选择，可为外部 EA/EB 输入信号或内部脉冲命令输出信号。

c154_set_position：

设定反馈位置计数器数值，请注意此值会先依据 c154_set_move_ratio() 进行比例运算，若参数 move_ratio 为 0.5，则计数器数值将为输入数值的 2 倍。

c154_get_command：

读取命令位置计数器数值，其信号源为脉冲命令输出信号。

c154_set_command：

设定命令位置器数器数值。

c154_get_error_counter：

读取位置误差计数器数值。

c154_reset_error_counter：

清除位置误差计数器数值。

c154_get_general_counter：

读取通用计数器数值。

c154_set_general_counter：

设定通用计数器信号源以及数值。（默认信号源为脉冲产生器输入）

@ 语法

C/C++(Windows XP/7/8.1)

I16 c154_get_position(I16 AxisNo, F64 *Pos);

I16 c154_set_position(I16 AxisNo, F64 Pos);

I16 c154_get_command(I16 AxisNo, I32 *Command);

I16 c154_set_command(I16 AxisNo, I32 Command);

I16 c154_get_error_counter(I16 AxisNo, I16 *error);

I16 c154_reXP/7/8.1set_error_counter(I16 AxisNo);

I16 c154_get_general_counter(I16 AxisNo, F64 *CntValue);

I16 c154_set_general_counter(I16 AxisNo, I16 CntSrc, F64 CntValue);

Visual Basic (Windows XP/7/8.1)

c154_get_position(ByVal AxisNo As Integer, Pos As Double) As Integer

c154_set_position(ByVal AxisNo As Integer, ByVal Pos As Double) As Integer

c154_get_command(ByVal AxisNo As Integer, Cmd As Long) As Integer

c154_set_command(ByVal AxisNo As Integer, ByVal Cmd As Long) As Integer

c154_get_error_counter(ByVal AxisNo As Integer, ByRef error As Integer) As Integer

c154_reset_error_counter(ByVal AxisNo As Integer) As Integer

c154_set_general_counter(ByVal AxisNo As Integer, ByVal CntSrc As Integer, ByVal CntValue As Double) As Integer

c154_get_general_counter(ByVal AxisNo As Integer, ByRef Pos As Double) As Integer

@ 参数

AxisNo：运动轴编号

card_id	Physical axis	AxisNo
0	0	0
	1	1
	2	2
	3	3

card_id	Physical axis	AxisNo
1	0	4
	1	5
	...	...

Pos, *Pos：反馈位置计数器数值 (c154_get/ set_position)

数值范围：-134217728 to 134217727

Cmd, *Cmd：命令位置计数器数值

数值范围：-134217728 to 134217727

***error**：位置误差计数器

数值范围：-32768 to 32767

CntSrc：通用计数器信号源

Value	Meaning
0	Command pulse
1	EA/EB
2	Pulser input
3	System clock÷2

CntValue, *CntValue：通用计数器数值

O 连续运动 (Continuous motion)

@ 名称

c154_set_continuous_move – 启动 / 关闭连续运动

c154_check_continuous_buffer – 检视命令缓存器缓冲是否为满的

c154_dwell_move – 设定暂停运动 (dwell move)

@ 描述

c154_set_continuous_move：

此函数必须于连续运动开始前与结束后执行。

c154_check_continuous_buffer：

利用此函数检视命令缓存器 (pre-register)(buffer) 是否为满的，若命令缓存器不是满的，使用者则可写入下一笔运动命令；若为满的，此新命令将会覆盖前一次命令 (2nd command pre-register)，请参阅 4.2.9 节。若此函数回传值为 1，表示缓存器满了；相反的，传回值若为 0，表示缓存器尚未满。

c154_dwell_move：

启动 dwell move，此为在特定时间内，不进行真正的运动。

例如：

c154_set_continuous_move(2, 1); // start continuous move

c154_start_tr_move(2, 20000.0, 10.0, 10000.0, 0.1, 0.1);

c154_dwell_move(2, 2000); //dwell move for 2 sec.

c154_start_sr_move(2, 20000.0, 10.0, 10000.0, 0.1, 0.1, 0, 0);

c154_set_continuous_move(2, 0); //end continuous move

@ 语法

C/C++(Windows XP/7/8.1)

I16 c154_set_continuous_move(I16 AxisNo, I16 Enable);

I16 c154_check_continuous_buffer(I16 AxisNo);

I16 c154_dwell_move(I16 AxisNo, F64 ms);

Visual Basic (Windows XP/7/8.1)

c154_set_continuous_move(ByVal AxisNo As Integer, ByVal Enable As Integer) As Integer

c154_check_continuous_buffer(ByVal AxisNo As Integer) As Integer

c154_dwell_move(ByVal AxisNo As Integer, ByVal ms As Double) As Integer

@ 参数

AxisNo：运动轴编号

card_id	Physical axis	AxisNo
0	0	0
	1	1
	2	2
	3	3
1	0	4
	1	5
	...	...

Enable：启动或关闭连续运动

Value	Meaning
0	continuous motion sequence is finished (Disable)
1	continuous motion sequence is started (Enable)

millisecond：Dwell move 时间 (ms)

P　多轴同步操作 (Multiple Axes Simultaneous Operation)

@ 名称

c154_set_tr_move_all – 多轴同步 T 型速度曲线相对距离运动设定

c154_set_ta_move_all – 多轴同步 T 型速度曲线绝对位置运动设定

c154_set_sr_move_all – 多轴同步 S 型速度曲线相对距离运动设定

c154_set_sa_move_all – 多轴同步 S 型速度曲线绝对位置运动设定

c154_start_move_all – 开始进行多轴同步运动

c154_stop_move_all – 停止多轴同步运动

@ 描述

使用这些函数可同时开始或停止多个轴的运动。同时运动的轴由参数“AxisArray”设定，而c154_set_tr_move_all()内的参数“TotalAxes”表示同时运动的轴数量。

当设定完 c154_set_xx_move_all() 后，执行 c154_start_move_all()，便会开始多轴同步运动，而执行 c154_stop_move_all()，将会使多轴同步停止。

下列范例将教导如何使用这些函数。此范例为同时移动轴 0 与轴 1，各自移动距离为 80000 与 120000，若同比例设定速度与加减速时间，则两轴将会同时到达终点。

```
I16 axes[2] = {0, 1};
F64 dist[2] = {80000.0, 120000.0},
F64 str_vel[2] = {0.0, 0.0},
F64 max_vel[2] = {4000.0, 6000.0},
F64 Tacc[2] = {0.1, 0.6},
F64 Tdec[2] = {0.1, 0.6};
c154_set_tr_move_all(2, axes, dist, str_vel, max_vel,
Tacc, Tdec);
c154_start_move_all(axes[0]);
```

@ 语法

C/C++(Windows XP/7/8.1)

```
I16 c154_set_tr_move_all(I16 TotalAxes, I16 *AxisArray,
F64 *DistA, F64 *StrVelA, F64 *MaxVelA, F64 *TaccA,
F64 *TdecA);
I16 c154_set_ta_move_all(I16 TotalAx, I16 *AxisArray,
F64 *PosA, F64 *StrVelA, F64 *MaxVelA, F64 *TaccA,
F64 *TdecA);
I16 c154_set_sr_move_all(I16 TotalAx, I16 *AxisArray,
F64 *DistA, F64 *StrVelA, F64 *MaxVelA, F64 *TaccA,
F64 *TdecA, F64 *SVaccA, F64 *SVdecA);
I16 c154_set_sa_move_all(I16 TotalAx, I16 *AxisArray,
F64 *PosA, F64 *StrVelA, F64 *MaxVelA, F64 *TaccA,
F64 *TdecA, F64 *SVaccA, F64 *SVdecA);
I16 c154_start_move_all(I16 FirstAxisNo);
I16 c154_stop_move_all(I16 FirstAxisNo);
```

Visual Basic (Windows XP/7/8.1)

c154_set_tr_move_all(ByVal TotalAxes As Integer, ByRef AxisArray As Integer, ByRef DistA As Double, ByRef StrVelA As Double, ByRef MaxVelA As Double, ByRef TaccA As Double, ByRef TdecA As Double) As Integer

c154_set_ta_move_all(ByVal TotalAxes As Integer, ByRef AxisArray As Integer, ByRef PosA As Double, ByRef StrVelA As Double, ByRef MaxVelA As Double, ByRef TaccA As Double, ByRef TdecA As Double) As Integer

c154_set_sr_move_all(ByVal TotalAxes As Integer, ByRef AxisArray As Integer, ByRef DistA As Double, ByRef StrVelA As Double, ByRef MaxVelA As Double, ByRef TaccA As Double, ByRef TdecA As Double, ByRef SVaccA As Double, ByRef SVdecA As Double) As Integer

c154_set_sa_move_all(ByVal TotalAxes As Integer, ByRef AxisArray As Integer, ByRef PosA As Double, ByRef StrVelA As Double, ByRef MaxVelA As Double, ByRef TaccA As Double, ByRef TdecA As Double, ByRef SVaccA As Double, ByRef SVdecA As Double) As Integer

c154_start_move_all(ByVal FirstAxisNo As Integer) As Integer

c154_stop_move_all(ByVal FirstAxisNo As Integer) As Integer

@ 参数

TotalAxes：同步运动的轴数量

***AxisArray**：同步运动的轴数组

***DistA**：相对距离数组 (pulse)

***StrVelA**：起始速度数组 (PPS)

***MaxVelA**：最大速度数组 (PPS)

***TaccA**：加速度时间数组 (S)

***TdecA**：减速度时间数组 (S)

***PosA**：绝对位置数组 (pulse)

***SvaccA**：加速端的 S 曲线参数数组

***SvdecA**：减速端的 S 曲线参数

FirstAxisNo：同步运动轴数组中的第一轴

Q 多功能输出入信号 (Mulit-Function DIO)

@ 名称

c154_set_gpio_output – 设定多功能输出信号

c154_get_gpio_output – 读取多功能输出信号状态

c154_get_gpio_input – 读取多功能输入信号状态

c154_set_gpio_input_function – 设定多功能输入信号功能

@ 描述

c154_set_gpio_output：

PCI-C154+ 提供四组多功能输出信号，可藉由此函数控制。

c154_get_gpio_output：

读取多功能输出信号状态。

c154_get_gpio_input：

PCI-C154+ 提供四组多功能输入信号，可藉由此函数读取状态。

c154_set_gpio_input_function：

设定多功能输入信号功能，可设定为 DI、SD、CLR 以及 EMG。

@ 语法

C/C++(Windows XP/7/8.1)

I16 c154_set_gpio_output(I16 card_id, I16 DoValue);

I16 c154_get_gpio_output(I16 card_id, I16 * DoValue);

I16 c154_get_gpio_input(I16 card_id, I16 * DiValue);

I16 c154_set_gpio_input_function(I16 card_id, I16 Channel, I16Select, I16 Logic);

Visual Basic (Windows XP/7/8.1)

c154_set_gpio_output(ByVal card_id As Integer, ByVal DoValue As Integer) As Integer

c154_get_gpio_output(ByVal card_id As Integer, DoValue As Integer) As Integer

c154_get_gpio_input(ByVal card_id As Integer, DiValue As Integer) As Integer

c154_set_gpio_input_function(ByVal card_id As Integer, ByVal Channel As Integer, ByVal Select As Integer, ByVal Logic As Integer)As Integer

@ 参数

card_id: PCI-C154+ 卡片编号，卡片编号可由卡片编码开关 (SW1) 或 PCI 插槽顺序决定，请参阅 c154_initial()。

DoValue, *DoValue：多功能输出信号数值，Bit 0-3：D_out 0-3.

***DiValue**：多功能输入信号数值，Bit 0-3：D_in 0-3

Channel：多功能输入信号通道，DI0 - DI3

Select：多功能输入信号功能选择

Value	Meaning
0	General DI (default)
1	
2	SD(active low)
3	
4	CLR (active low)
5	EMG (active low)

Logic：多功能输入信号逻辑

Value	Meaning
0	Not inverse (default)
1	Inverse

R 软件极限 (Soft Limit)

@ 名称

c154_disable_soft_limit – 关闭软件极限功能

c154_enable_soft_limit – 启动软件极限功能

c154_set_soft_limit – 设定软件极限数值

@ 描述

c154_disable_soft_limit：

关闭软件极限功能。

c154_enable_soft_limit：

启动软件极限功能，此功能如同机械限位开关功能。

c154_set_soft_limit：

设定软件极限数值。

@ 语法

C/C++(Windows XP/7/8.1)

I16 c154_disable_soft_limit(I16 AxisNo);

I16 c154_enable_soft_limit(I16 AxisNo, I16 Action);

I16 c154_set_soft_limit(I16 AxisNo, I32 PlusLimit, I32 MinusLimit);

Visual Basic (Windows XP/7/8.1)

c154_disable_soft_limit(ByVal AxisNo As Integer) As Integer

c154_enable_soft_limit(ByVal AxisNo As Integer, ByVal Action As Integer) As Integer

c154_set_soft_limit(ByVal AxisNo As Integer, ByVal PlusLimit As Long, ByVal MinusLimit As Long) As Integer

@ 参数

AxisNo：运动轴编号

card_id	Physical axis	AxisNo
0	0	0
	1	1
	2	2
	3	3
1	0	4
	1	5
	...	...

Action：当软件极限触发时，对应行为

Value	Meaning
0	INT only
1	Immediately stop
2	slow down then stop

PlusLimit：正方向软件极限数值

MinusLimit：负方向软件极限数值

S　扩展通用输出入信号 (Extened DIO)

@ 名称

c154_set_gpio_output_ex – 设定扩展通用输出信号

c154_get_gpio_output_ex – 读取扩展通用输出信号状态

c154_get_gpio_input_ex – 读取扩展通用输入信号状态

c154_set_gpio_output_ex_CH – 依据通道设定扩展通用输出信号

c154_get_gpio_output_ex_CH – 依据通道读取扩展通用输出信号状态

c154_get_gpio_input_ex_CH – 依据通道读取扩展通用输入信号状态

@ 描述

c154_set_gpio_output_ex：

PCI-C154+ 提供 16 组扩展通用输出信号，可藉由此函数控制。

c154_get_gpio_output_ex：

读取扩展通用输出信号状态。

c154_get_gpio_input_ex：

PCI-C154+ 提供 16 组扩展通用输入信号，可藉由此函数读取状态。

c154_set_gpio_output_ex_CH:

PCI-C154+ 提供 16 组扩展通用输出信号，可藉由此函数控制单一通道。

c154_get_gpio_output_ex_CH:

依据通道读取扩展通用输出信号状态。

c154_get_gpio_input_ex_CH:

PCI-C154+ 提供 16 组扩展通用输入信号，可藉由此函数读取单一通道状态。

@ 语法

C/C++(Windows XP/7/8.1)

I16 c154_set_gpio_output_ex(I16 card_id, I16 DoValue);

I16 c154_get_gpio_output_ex(I16 card_id, I16 * DoValue);

I16 c154_get_gpio_input_ex(I16 card_id, I16 * DiValue);

I16 c154_set_gpio_output_ex_CH(I16 card_id, I16 Channel, I16 Value);

I16 c154_get_gpio_output_ex_CH(I16 card_id, I16 Channel, I16 * Value);

I16 c154_get_gpio_input_ex_CH(I16 card_id, I16 Channel, I16 * Value);

Visual Basic (Windows XP/7/8.1)

c154_set_gpio_output_ex(ByVal card_id As Integer, ByVal DoValue As Integer) As Integer

c154_get_gpio_output_ex(ByVal card_id As Integer, DoValue As Integer) As Integer

c154_get_gpio_input_ex(ByVal card_id As Integer, DiValue As Integer) As Integer

c154_set_gpio_output_ex_CH(ByVal card_id As Integer, ByVal Channel As Integer, ByVal Value As Integer) As Integer

c154_get_gpio_output_ex_CH(ByVal card_id As Integer, ByVal Channel As Integer, Value As Integer) As Integer

c154_get_gpio_input_ex_CH(ByVal card_id As Integer, ByVal Channel As Integer, Value As Integer) As Integer

@ 参数

card_id: PCI-C154+ 卡片编号，卡片编号可由卡片编码开关 (SW1) 或 PCI 插槽顺序决定，请参阅 c154_initial()。

DoValue, *DoValue：扩展通用输出信号数值， Bit 0-15：D_out 0-15.

***DiValue：**扩展通用输入信号数值，Bit 0-15：D_in 0-15

Channel: 信号通道， 0-15

Value: 单一通道信号数值，0 或 1

T　背隙补偿与震动抑制 (Backlash Compensation / Vibration Suppression)

@ 名称

c154_backlash_comp – 设定补偿数值

c154_suppress_vibration – 设定抑制震动之脉冲输出延迟时间

c154_set_fa_speed – 设定 FA 速度 (应用于背隙 / 滑移补偿与归零运动)

@ 描述

c154_backlash_comp：

当运动方向改变时，PCI-C154+会实行背隙补偿；利用此函数可设定补偿数值。

c154_suppress_vibration：

控制器会于反方向输出一脉冲信号，紧接着于正方向再输出一脉冲信号，来抑制系统震动，利用此函数可设定脉冲输出延迟时间(RT, FT)。详细请参阅 4.8.2 节。

c154_set_fa_speed：

设定背隙与滑移补偿运动速度 (FA)，此速度同时也是归零运动中的反向低速。

@ 语法

C/C++(Windows XP/7/8.1)

I16 c154_backlash_comp(I16 AxisNo, I16 CompPulse, I16 Mode);

I16 c154_suppress_vibration(I16 AxisNo, U16 ReverseTime, U16 ForwardTime);

I16 c154_set_fa_speed(I16 AxisNo, F64 FA_Speed);

Visual Basic (Windows XP/7/8.1)

c154_backlash_comps (ByVal AxisNo As Integer, ByVal CompPulse As Integer, ByVal Mode As Integer) As Integer

c154_suppress_vibration(ByVal AxisNo As Integer, ByVal ReverseTime As Integer, ByVal ForwardTime As Integer) As Integer

c154_set_fa_speed(ByVal AxisNo As Integer, ByVal FA_Speed As Double) As Integer

@ 参数

AxisNo：运动轴编号

card_id	Physical axis	AxisNo
0	0	0
	1	1
	2	2
	3	3
1	0	4
	1	5
	...	...

CompPulse：补偿数值， 12 bit

Mode：

Value	Meaning
0	Turns off
1	enable backlash compensation
2	Slip correction

ReverseTime：反向脉冲延迟时间， 0 – 65535，单位：1.6 us

ForwardTime：正向脉冲延迟时间， 0 – 65535，单位：1.6 us

FA_Speed：FA 速度 (PPS)

U 函数回传值 (Return Code)

函数回传值定义于 “C154_err.h”，其内容描述如下表

Code	Meaning
0	No error
-10000	Error Card number
-10001	Error operation system version
-10002	Error card’s ID conflict
-10300	Error other process exist
-10301	Error card not found
-10302	Error Open driver failed
-10303	Error ID mapping failed
-10304	Error trigger channel

Code	Meaning
-10305	Error trigger type
-10306	Error event already enabled
-10307	Error event not enable yet
-10308	Error on board FIFO full
-10309	Error unknown command type
-10310	Error unknown chip type
-10311	Error card not initial
-10312	Error position out of range
-10313	Error motion busy
-10314	Error speed error
-10315	Error slow down point
-10316	Error axis range error
-10317	Error compare parameter error
-10318	Error compare method
-10319	Error axis already stop
-10320	Error axis INT wait failed
-10321	Error user code write failed
-10322	Error array size exceed
-10323	Error factor number
-10324	Error enable range
-10325	Error auto accelerate time
-10326	Error dwell time
-10327	Error dwell distance
-10328	Error new position
-10329	Error motion not in running
-10330	Error velocity change time
-10331	Error speed target
-10332	Error velocity percent
-10333	Error position change backward
-10334	Error counter number
-10335	Error gpio input function parameter
-10336	Error channel number
-10337	Error ERC mode
-10338	Error security code

重要安全说明

出于用户安全方面的考虑，运行 / 操作设备之前，请阅读并遵守本手册中和相关设备上标出的全部**说明**、**警告**、**小心**和**注意**的内容。

- 请认真阅读这些安全说明。
- 请保留本用户手册，以备将来参考之用。
- 请阅读本手册的规定部分，了解关于本设备操作环境的详细信息。
- 安装或拆卸设备时：
  - 关闭电源并拔下所有电源线 / 电缆。
- 为了避免发生电击和 / 或造成设备损坏：
  - 使设备远离水或液体源；
  - 使设备远离高温或高湿度的环境；
  - 使设备保持良好的通风状况 (不得堵塞或盖上通风口)；
  - 请务必使用建议的电压和电源设置；
  - 始终在易于使用的电气插座附近安装并操作设备；
  - 请固定好电源线 （不得将任何物体放在电源线上）；
  - 仅在稳固的表面和 / 或建议的配件上安装 / 连接和操作设备；
  - 并且如果长期不使用设备，请关闭设备电源并拔下电源插头。

- 绝不要擅自维修设备。只能由有资质的人员维修设备。

可提供锂电池作为不中断电源、备用电源或应急电源。



如果更换的电池型号不正确，可能存在爆炸危险。请正确处理使用过的电池。

- 如有下列情况，本产品限由授权的技术人员处理：
  - 电源线或插头受损；
  - 设备进水；
  - 设备暴露于高湿 / 高水气的环境；
  - 设备无法运作，或不按使用手册所描述方式运作；
  - 遭摔落及 / 或损坏；
  - 有明显的破损。

服务信息

如需任何服务或协助请联系本公司。

ADLINK Technology, Inc.
Address: 9F, No.166 Jian Yi Road, Zhonghe District
New Taipei City 235, Taiwan
新北市中和區建一路 166 號 9 樓
Tel: +886-2-8226-5877
Fax: +886-2-8226-5717
Email: service@adlinktech.com

Ampro ADLINK Technology, Inc.
Address: 5215 Hellyer Avenue, #110, San Jose, CA 95138, USA
Tel: +1-408-360-0200
Toll Free: +1-800-966-5200 (USA only)
Fax: +1-408-360-0222
Email: info@adlinktech.com

ADLINK Technology (China) Co., Ltd.
Address: 上海市浦东新区张江高科技园区芳春路 300 号 (201203)
300 Fang Chun Rd., Zhangjiang Hi-Tech Park,
Pudong New Area, Shanghai, 201203 China
Tel: +86-21-5132-8988
Fax: +86-21-5132-3588
Email: market@adlinktech.com

ADLINK Technology Beijing
Address: 北京市海淀区上地东路 1 号盈创动力大厦 E 座 801 室(100085)
Rm. 801, Power Creative E, No. 1,
Shang Di East Rd., Beijing, 100085 China
Tel: +86-10-5885-8666
Fax: +86-10-5885-8626
Email: market@adlinktech.com

ADLINK Technology Shenzhen
Address: 深圳市南山区科技园南区高新南七道 数字技术园
A1 栋 2 楼 C 区 (518057)
2F, C Block, Bldg. A1, Cyber-Tech Zone, Gao Xin Ave. Sec. 7,
High-Tech Industrial Park S., Shenzhen, 518054 China
Tel: +86-755-2643-4858
Fax: +86-755-2664-6353
Email: market@adlinktech.com

LiPPERT ADLINK Technology GmbH
Address: Hans-Thoma-Strasse 11, D-68163, Mannheim, Germany
Tel: +49-621-43214-0
Fax: +49-621 43214-30
Email: emea@adlinktech.com

ADLINK Technology, Inc. (French Liaison Office)
Address: 15 rue Emile Baudot, 91300 Massy CEDEX, France
Tel: +33 (0) 1 60 12 35 66
Fax: +33 (0) 1 60 12 35 66
Email: france@adlinktech.com

ADLINK Technology Japan Corporation
Address: 〒101-0045 東京都千代田区神田鍛冶町 3-7-4
神田 374 ビル 4F
KANDA374 Bldg. 4F, 3-7-4 Kanda Kajicho,
Chiyoda-ku, Tokyo 101-0045, Japan
Tel: +81-3-4455-3722
Fax: +81-3-5209-6013
Email: japan@adlinktech.com

ADLINK Technology, Inc. (Korean Liaison Office)
Address: 서울시 서초구 서초동 1675-12 모인터빌딩 8 층
8F Mointer B/D,1675-12, Seocho-Dong, Seocho-Gu,
Seoul 137-070, Korea
Tel: +82-2-2057-0565
Fax: +82-2-2057-0563
Email: korea@adlinktech.com

ADLINK Technology Singapore Pte. Ltd.
Address: 84 Genting Lane #07-02A, Cityneon Design Centre,
Singapore 349584
Tel: +65-6844-2261
Fax: +65-6844-2263
Email: singapore@adlinktech.com

ADLINK Technology Singapore Pte. Ltd. (Indian Liaison Office)
Address: 1st Floor, #50-56 (Between 16th/17th Cross) Margosa Plaza,
Margosa Main Road, Malleswaram, Bangalore-560055, India
Tel: +91-80-65605817, +91-80-42246107
Fax: +91-80-23464606
Email: india@adlinktech.com

ADLINK Technology, Inc. (Israeli Liaison Office)
Address: 6 Hasadna St., Kfar Saba 44424, Israel
Tel: +972-9-7446541
Fax: +972-9-7446542
Email: israel@adlinktech.com