KPCI-884 步进电机控制卡

使用说明书

Ver 2.1

北京科瑞兴业科技有限公司

北京科瑞兴业科技有限公司　　地址：北京市海淀区知春里 28 号开源商务写字楼 212/213 室
邮政编码：100086　　电话：010-51650651　010-62527214　　传真：010-62657424
http://www.krxgk.com
Sales E-mail: sgq@krxgk.com　　Tech Support E-mail: lilanzhen007@126.com

第一章　　概述

KPCI-884 运动控制卡的简介:

KPCI-884 是一款能够同时控制 4 个伺服电机或步进电机的运动控制卡,它以高频率脉冲串输出方式，控制伺服和步进电机的运动。在一个系统中，可嵌入多块卡同时使用。（即：最多能控制 16 个以上的步进电机同时运动）。该卡能精确地控制所发出的脉冲频率（电机速度）、脉冲个数（电机转角）及脉冲频率变化率（电机加速度），它能满足步进电机的各种复杂的控制要求。可对电机进行位置控制、插补驱动、加速/减速等控制。它含有丰富的，功能齐全的软件库函数资源。在 Windows9X/2000 环境下，用户可直接使用我们为您提供的"DLL"动态链接库函数；以最大方便地使您在 Microsoft Visual BASIC 、Visual C++及各种其他软件环境中使用。同时提供了 VB 和 VC 两种格式的 KPCI-884 测试示范软件，可演示此卡功能。

第二章　　产品 KPCI-884 的特点

■ **独立 4 轴驱动**

可以同时分别控制 4 个步进电机驱动轴的运动，每个轴都可以进行定速直线驱动，加/减速驱动，S 曲线加/减速驱动。4 轴的性能相同。

■ **速度控制**

可以运行固定速度驱动；直线加/减速驱动；S 曲线加/减速驱动。可以使用程序控制和外部手动控制 2 种操作方法驱动。脉冲输出的速度可以在驱动中自由变更。

■ **S-曲线加/减速驱动**

每个轴可以用 S-曲线进行加/减速设定；使用 S-曲线命令还可以对抛物线加/减速驱动输出脉冲进行设定，此外对于定量驱动使用独特的方法避免在 S-曲线加/减速中发生**三角波形**。该方法由动态连接库完成，用户不必顾虑。

■　**固定线速度控制**

这是一种在插补驱动中保持插补轴合成速度的功能，2 轴同时输出脉冲时,第 2 轴可以设定为 1.414 倍脉冲周期。

■ **2 轴/3 轴位模式插补**

可以用指定的驱动速度连续输出插补脉冲，用这种方式可以产生任何插补曲线

■ **连续插补**

直线插补 → 圆弧插补 → 直线插补 → … 这样可以不停地运行每个插补节点的插补驱动。

■**位置控制**

每轴都含有硬件构成的 2 个 32 位位置计数器。一个是在内部管理驱动脉冲输出的逻辑位置计数器，另一个是管理从外部编码器输入的脉冲的实际位置计数器。此外，还有 2 个 32 位比较寄存器，用于与逻辑位置计数器或者实际位置计数器的位置大小相比较；在驱动中，可

以从状态寄存器读出比较寄存器和逻辑/实际位置计数器之间的大小关系。对应的命令函数是 int ReadCOMP(int num, unsigned short axis)

■由外部信号驱动

每个轴都可以由外部信号（nEXOP+、nEXOP-）进行+/-方向运行的定量驱动和连续驱动。这功能可在手动操作时，减轻 CPU 的负担，且使各轴可以平稳地运动。

■实时监控功能

在电机运动过程中，可以实时读出逻辑位置 ReadLP，实际位置 ReadEP，驱动速度 ReadCV，加速度 ReadCA，加/减速状态(加速、定速、减速) MotorStatus。

■通用 IO 控制

通用 IO 控制 28 路，即：12 路光隔输入 DC 0-24V；16 路光隔输出 DC 0-24V，每一个轴对应 4 路输出和 3 路输入。

第三章　KPCI-884 的技术指标

3.1 输入信号：　各轴共用的紧急停车，各轴独立的正反向极限限位，各轴独立的开关量输入点，各轴独立的正反向点动控制信号。输入信号均有光隔。
3.2 输出信号：　各轴独立的光隔输出点：24VDC，每点可输出 100mA。
3.3 隔离电压：　>1000V。
3.4 外型尺寸：　195mm×150mm。
3.5 使用环境：　工作温度：0°C-70°C
相对湿度：0%-95%RH　(不凝露)
存储温度：-55°C-+85°C
3.6 两/三轴线性插补　速度：1PPS—4MPPS
精度：±0.5LSB
3.7 两轴圆弧插补　速度：1PPS—4MPPS
精度：±0.5LSB
3.8 连续插补　速度：1PPS—2MPPS
3.9 供电电压：　外供电 DC24V/3W

第四章　控制卡的安装与设置

在安装控制卡前，首先要关掉主机电源，将主机机壳打开,将 KPCI-884 插在任意一个空余的 PCI 插槽中，再用档板螺丝将本卡固定好。注意：一定确保在计算机关闭电源后再插拔控制卡。

4.1 卡上拨码开关的设置

当计算机中插入了多块 KPCI-884 板卡时，拨码开关用于区分各个板卡。比如计算机上插入了 4 块板卡，则板卡号可根据您的情况分别设置为 0，1，2，3。当需要操作某一个板卡时，则只需要在程序中指定这个板卡的拨码开关指定的板卡号、打开板卡进行相应的操作即可。拨码开关的使用方法：拨到上边为 1，下边为 0。遵循二进制，左边为低位，右边为

高位。

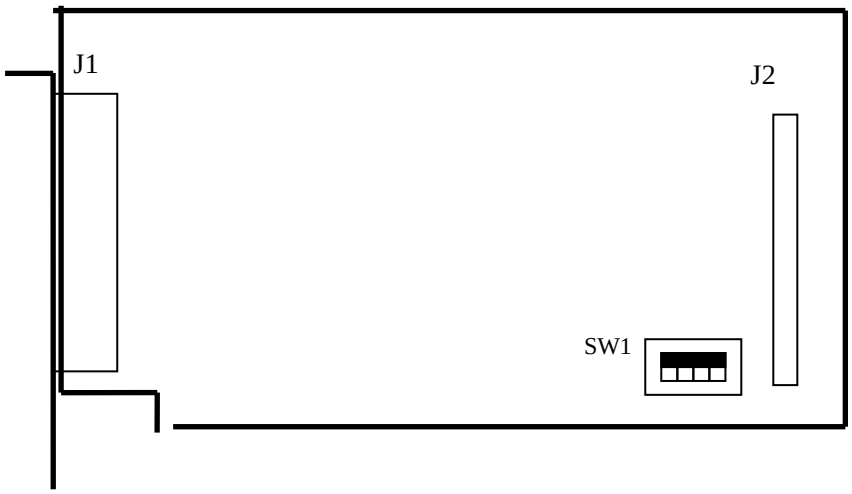
注意：板卡号从 0 开始，即如果您的计算机中只插入了一块板卡，则拨码开关必须拨成 0。即当插入多块板卡时，最高板卡号比实际板卡数目少一。

KPCI-884 卡是 PCI 总线运动控制卡，由计算机自动分配地址，卡上有一个 4 位拨码开关，是用来设定 KPCI-884 的设备号的，设定范围从 0~15。注意：设备地址不能大于计算机上安装 KPCI-884 板卡的数量。

表 1.2　S1 拨码开关的缺省设置

开关	SW1	SW2	SW3	SW4
设备 0	ON（0）	ON（0）	ON（0）	ON（0）
设备 1	ON（0）	ON（0）	ON（0）	OFF(1)
设备 2	ON（0）	ON（0）	OFF(1)	ON（0）
•••				
设备 14	OFF（1）	OFF（1）	OFF(1)	ON（0）
设备 15	OFF（1）	OFF（1）	OFF(1)	OFF(1)

4.2 引脚定义



板卡上 J1 各个引脚的定义

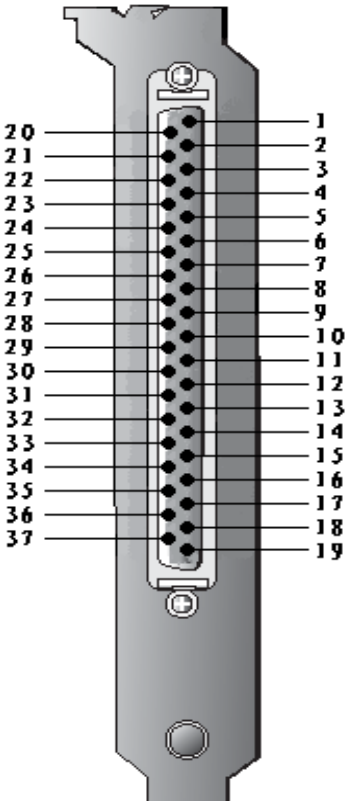
图中 J1　62 针 D 型插头连线表，对应图（62 针 D 型插头外型简图）。

序号	定义	说明	序号	定义	说明
1	+24V	外接电源 VCC	32	YEXOP+	Y 轴正方向点动
2	0V	外接电源地 GND	33	Z_out1	Z 轴通用数字 I/O 输出点
3	X_out1	X 轴通用数字 I/O 输出点	34	Z_out4	Z 轴通用数字 I/O 输出点
4	X_out4	X 轴通用数字 I/O 输出点	35	Z_in3	Z 轴通用数字 I/O 输入点
5	X_in3	X 轴通用数字 I/O 输入点	36	ZCW/CP	Z 轴正脉冲输出(或 Z 轴脉冲输出)
6	XCW/CP	X 轴正脉冲输出（或 X 轴脉冲输出）	37	ZEXOP-	Z 轴负方向点动
7	XEXOP-	X 轴负方向点动	38	U_out2	U 轴通用数字 I/O 输出点
8	Y_out2	Y 轴通用数字 I/O 输出点	39	U_in1	U 轴通用数字 I/O 输入点
9	Y_in1	Y 轴通用数字 I/O 输入点	40	ULMT+	U 轴正向限位输入
10	YLTM+	Y 轴正向限位输入	41	UCCW/DIR	U 轴负方向脉冲（或 U 轴 DIR）
11	YCCW/DIR	Y 轴负方向脉冲（或 U 轴 DIR）	42	0V	外接电源地 GND

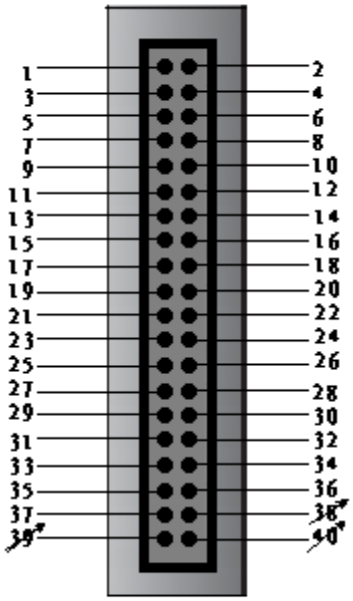
12	0V	外接电源地	43	空	
13	Z_out3	Z 轴通用数字 I/O 输出点	44	EMG	紧急停车输入
14	Z_in2	Z 轴通用数字 I/O 输入点	45	X_out3	X 轴通用数字 I/O 输出点
15	ZLTM-	Z 轴负向限位输入	46	X_in2	X 轴通用数字 I/O 输入点
16	ZEXOP+	Z 轴正方向点动	47	XLMT-	X 轴负向限位输入
17	U_out1	U 轴通用数字 I/O 输出点	48	XEXOP+	X 轴正方向点动
18	U_out4	U 轴通用数字 I/O 输出点	49	Y_out1	Y 轴通用数字 I/O 输出点
19	U_in3	U 轴通用数字 I/O 输入点	50	Y_out4	Y 轴通用数字 I/O 输出点
20	UCW/CP	U 轴正脉冲输出（或 U 轴脉冲输出）	51	Y_in3	Y 轴通用数字 I/O 输入点
21	UEXOP-	U 轴负方向点动	52	YCW/CP	Y 轴正脉冲输出(或Y轴脉冲输出)
22	+24V	外接电源 VCC	53	YEXOP-	Y 轴负方向点动
23	0V	外接电源地 GND	54	Z_out2	Z 轴通用数字 I/O 输出点
24	X_out2	X 轴通用数字 I/O 输出点	55	Z_in1	Z 轴通用数字 I/O 输入点
25	X_in1	X 轴通用数字 I/O 输入点	56	ZLMT+	Z 轴正向限位输入
26	XLMT+	X 轴正向限位输入	57	ZCCW/DIR	Z 轴负方向脉冲（或 Z 轴 DIR）
27	XCCW/DIR	X 轴负方向脉冲（或 X 轴 DIR）	58	0V	外接电源地 GND
28	0V	外接电源地 GND	59	U_out3	U 轴通用数字 I/O 输出点
29	Y_out3	Y 轴通用数字 I/O 输出点	60	U_in2	U 轴通用数字 I/O 输入点
30	Y_in2	Y 轴通用数字 I/O 输入点	61	ULMT-	U 轴负向限位输入
31	YLMT-	Y 轴负向限位输入	62	UEXOP+	U 轴正方向点动

板卡上 J2 各个引脚的定义

本卡通过一条专用的转接线将 40 芯扁平带缆插座转接为一个 37 芯母插座，用档片固定在计算机机箱上，这样大大方便了本卡与外部设备的连接。两个插座的对应图如下图所示。



37 针 D 型插头外型简图



37 芯母插头外型简图

其中引脚 38、39、40 未使用

J2　37 芯 D 型母插头连线表，

37 芯插座序号(转接线)	40 芯插座序号（板卡上）	信号定义	说明
1	1	XINPOS	X 轴位置输入
2	3	XALARM	X 轴伺服误差（输入）
3	5	XECAP	X 轴编码器 A 信号（输入）
4	7	XECAN	X 轴编码器/A 信号（输入）
5	9	XECBP	X 轴编码器 B 信号（输入）
6	11	XECBN	X 轴编码器/B 信号（输入）
7	13	XECZP	X 轴编码器 Z 信号（输入）
8	15	XECZN	X 轴编码器/Z 信号（输入）
9	17	0V	外接电源地 GND
10	19	YINPOS	Y 轴位置输入
11	21	YALARM	Y 轴伺服误差（输入）
12	23	YECAP	Y 轴编码器 A 信号（输入）
13	25	YECAN	Y 轴编码器/A 信号（输入）
14	27	YECBP	Y 轴编码器 B 信号（输入）
15	29	YECBN	Y 轴编码器/B 信号（输入）
16	31	YECZP-	Y 轴编码器 Z 信号（输入）
17	33	YECZN-	Y 轴编码器/Z 信号（输入）
18	35	0V	外接电源地 GND
19	37	ZINPOS	Z 轴位置输入
20	2	ZALARM	Z 轴伺服误差（输入）
21	4	ZECAP	Z 轴编码器 A 信号（输入）
22	6	ZECAN	Z 轴编码器/A 信号（输入）
23	8	ZECBP	Z 轴编码器 B 信号（输入）
24	10	ZECBN	Z 轴编码器/B 信号（输入）
25	12	ZECZP	Z 轴编码器 Z 信号（输入）
26	14	ZECZN	Z 轴编码器/Z 信号（输入）
27	16	0V	外接电源地 GND
28	18	UINPOS	U 轴位置输入
29	20	UALARM	U 轴伺服误差（输入）
30	22	UECAP	U 轴编码器 A 信号（输入）
31	24	UECAN	U 轴编码器/A 信号（输入）
32	26	UECBP	U 轴编码器 B 信号（输入）
33	28	UECBN	U 轴编码器/B 信号（输入）
34	30	UECZP	U 轴编码器 Z 信号（输入）
35	32	UECZN	U 轴编码器/Z 信号（输入）
36	34	0V	外接电源地 GND

37	36、37、38、39、40	空	

4.3 J1、J2 上部分引脚功能的详细介绍（n 表示轴 X、Y、Z、U）

nLMT+、nLMT- 每个轴的正/负方向超程限位信号输入：当超程限制开关传感器有效时，相应轴硬件限位信号（nLMT+、nLMT-）输入管脚用来中止脉冲输出。或者说：各轴独立的正反向极限限位输入点用来起到极限保护作用，有效时相应轴驱动脉冲会立即停止输出。限位报警表示机床移动部分碰了行程限位开关，在执行加工程序时，所有轴都将停止运动，通过按“复位”键取消剩余运动，再进入手轮或手动方式，向反向移动拖板，退出限位开关，则报警会自动解除。

nCW/CP 、nCCW/DIR 可以通过函数 PulseOutMode 来选择板卡脉冲输出的工作模式：一种是 CW/CCW 类型，称作双脉冲换向方式，板卡发送两路脉冲信号（脚注为 CW 和 CCW）来驱动步进电机驱动器工作，当其中一路（如 CW）有脉冲信号时，电机正向运行，当另一路（CCW）有脉冲信号时，电机反向运行。另一种是 Pulse/DIR 类型，称为单脉冲方式，CP 用来发送控制脉冲，而 DIR 信号决定电机的旋转方向；比如说 DIR 为高电平时电机顺时针旋转，而此信号为低电平时电机逆时针旋转。在 CW/CCW 模式下，进行正方向驱动时，驱动脉冲从 nCW/CP 端口输出；进行负方向驱动时，驱动脉冲从 nCCW/DIR 输出。在 Pulse/DIR 模式下，nCW/CP 负责输出驱动脉冲，而 nCCW/DIR 则负责输出驱动方向信号。脉冲输出类型选择如表所示：

脉冲输出方式	驱动方向	输出脉冲波形			
		CW（正）	CCW（负）	CP（脉冲）	DIR（方向）
CW\CCW 方式	正驱动方向	脉冲	电平		
	负驱动方向	电平	脉冲		
Pulse/DIR 方式	正驱动方向			脉冲	高电平
	负驱动方向			脉冲	低电平

EMG 用于急停系统的管脚。该管脚的正常状态是高电平，当急停管脚变为低电平时，所有轴都将立即停止。所以该管脚可以连接外部急停信号端。

nEXOP+、nEXOP- 该管脚连接点动输入信号端。在进行系统定位的时候，常用到点动定位，而点动定位的点动距离可由软件设定，或者使用默认值(软件设定)。

n_out1~4 每个轴都有 4 个数字信号输出端口，可以用来控制几个冷却液的启停或用来控制其它的执行部件等。具体连接那些部件需要根据板卡应用的实际情况来定。

n_in1~ 3 每个轴同时也有 3 个数字信号输入端口。可以用来接收机床原点到位减速信号、原点到位信号、系统位置反馈信号等。具体连接那些信号也需要根据板卡应用的实际情况来定。

nINPOS 来自伺服驱动器，用于位置检测的一个输入信号，在伺服驱动器完成一个位置命令后有效。用户可以使能或屏蔽该信号。

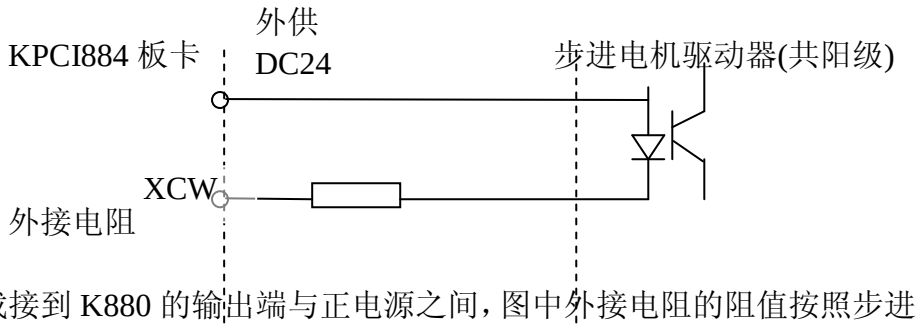
nALARM 来自伺服驱动器，用于驱动器报警信号输入。当伺服驱动器处于不正常状态时，该信号有效，使相对应的轴停止脉冲输出。当使能 nALARM 功能时，如果板卡正在输出脉冲，当 nALARM 有效后，该输出脉冲将立即停止。

nECAP、nECAN、nECBP、nECBN、nECZP、nECZN： 反馈编码器信号，分别连接到编码器的 A、/A、B、/B、Z、/Z 信号输出。板卡的位置反馈的默认设置是积分输入。差动脉冲反馈在设置输入脉冲模式后有效。nECZP/N 用于输入编码器 Z 信号反馈。

第五章　运动系统的的安装

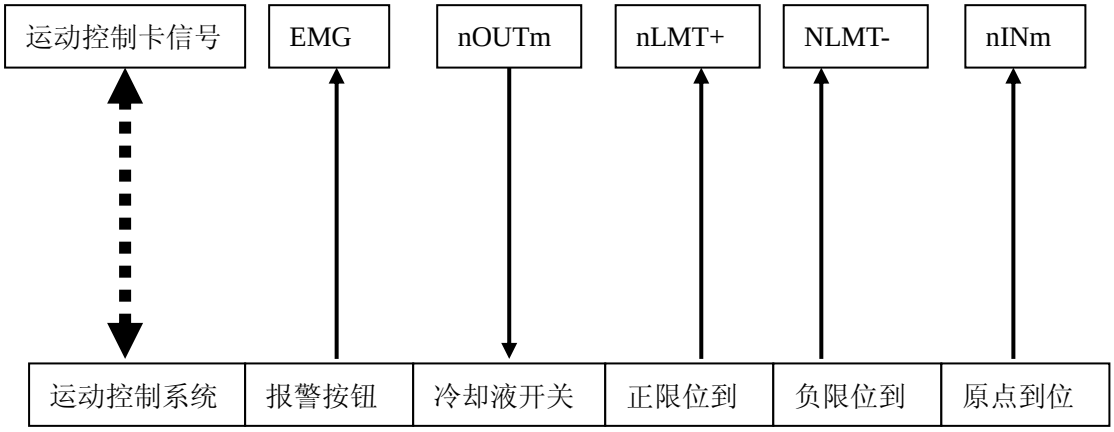
5.1 板卡与伺服驱动器的接线

板卡与步进电机驱动器接线示意图：

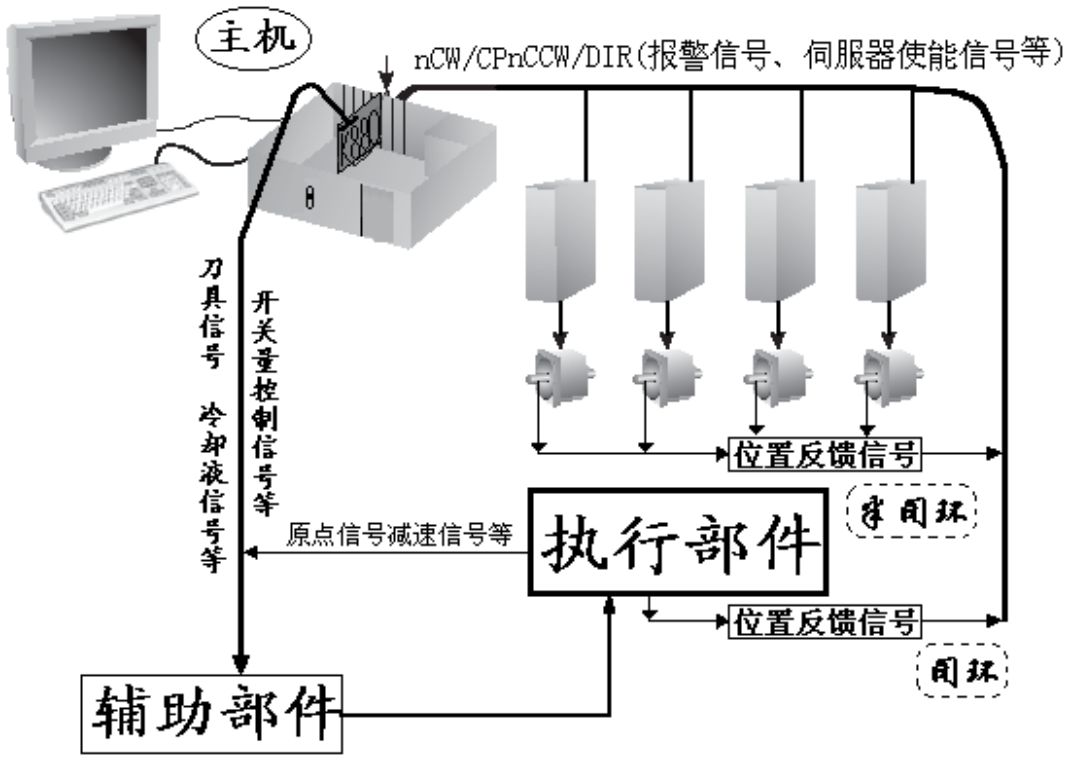


上图中负载接到 K880 的输出端与正电源之间，图中外接电阻的阻值按照步进电机驱动器的说明书确定。

5.2　板卡其它部分信号线、控制线与运动系统的连接



5.3 板卡与伺服驱动、运动系统的安装



5.4 注意事项

5.4.1 从将板卡插入主机箱内到接线布线都要断电操作；

5.4.2 注意主机箱的静电的释放，保证避免静电干扰；

5.4.3 控制线最好采用屏蔽线且与电机线或电源线等强电线分开；

5.4.4 严格禁止和接线端子相连的线头上镀锡！否则会引起接线端子发热损坏，一般采用导线压片或直接接入；

5.4.5 驱动器在关机后，至少一分钟以上才能再开机。

第六章 板卡的详细功能说明以及相关技术参数

脉冲当量：在一个脉冲作用下，工作台移动的一个最小基本长度单位（um）。如：用 MCX314AS 的位模式插补来实现样条/函数曲线等插补，需要处理大量的数据。例如，对于脉冲当量为 1um 的系统，3 轴插补一段 10mm(1mm=1000um) 的直线就需要{[(10mm/1um)/8]BIT/1024}BYTE*3*2≈7.3KB。

输出脉冲的倍率（M） M=8 000 000/R

S 曲线加减速度变化设定值 （PPS/s^2） =（1/k）62.5*10^6M（变量 K 的范围 1~65535），其中 62.5*10^6 为常数

实际加速度（单位：PPS/s）=A*125M（A 为设定的加速度）

实际减速度（单位：PPS/s）=D *125M（D 为设定的 A 减速度）

实际驱动速度（单位：PPS）=V*M（V 为设定的驱动速度）

实际初始速度（单位：PPS）=SV*M（SV 为设定的初始速度）

6.1 定长脉冲输出驱动

如果想实现将一个电机移动 N 步；（即 KPCI-884 卡向步进电机发出 N 个脉冲）需要设置加/减速 A/D，驱动速度 V，初始速度 SV，输出脉冲数 P。KPCI-884 将产生脉冲且自动输出。当输出的脉冲数等于设置的脉冲数 P 时，KPCI-884 立即停止输出脉冲。

6.1.1 在驱动中改变输出脉冲数

在定长脉冲驱动中，输出脉冲数是可以改变的。脉冲输出状况将如图 6.1.2 或图 6.1.3。

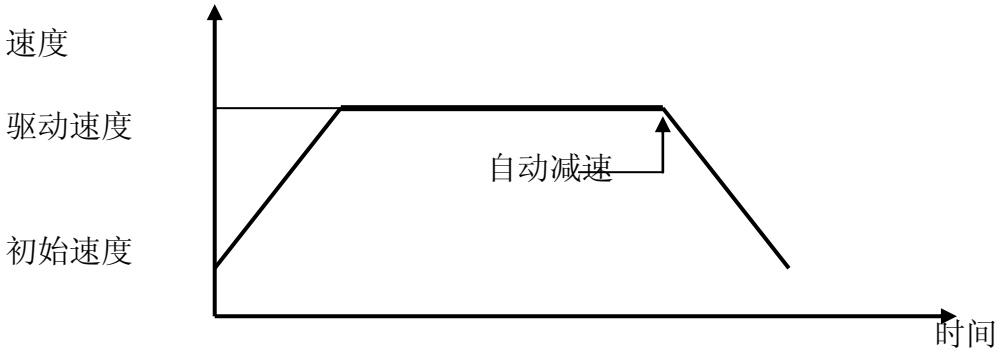


图 6.1.1 定长脉冲驱动模式

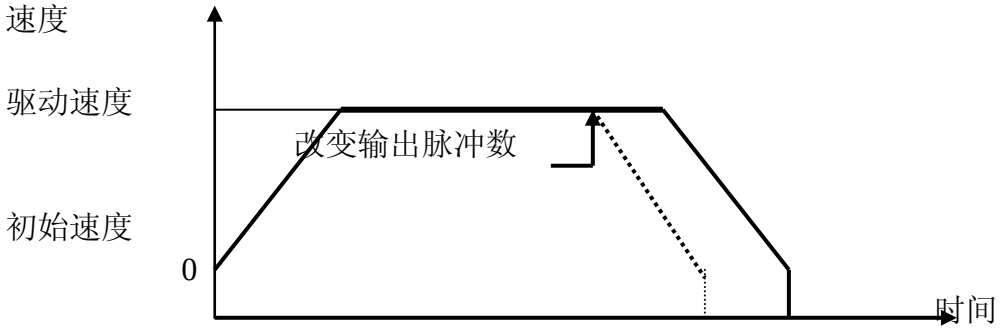


图 6.1.2 在匀速段增加脉冲数

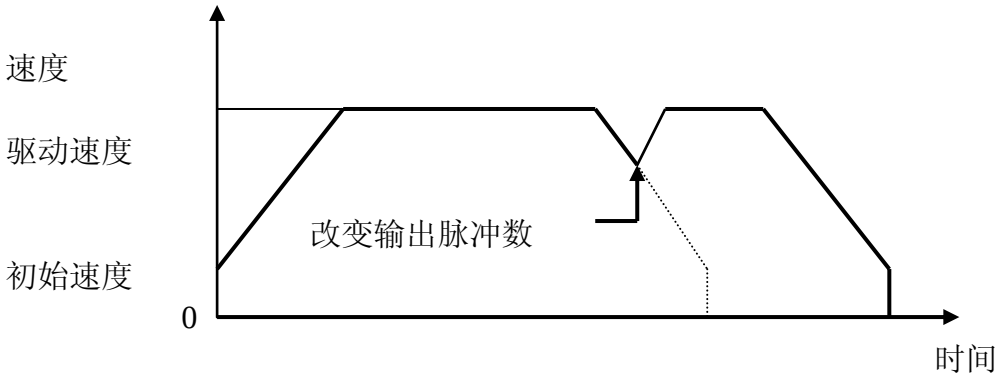


图 6.1.3 在降速段增加脉冲数

如果是减少输出脉冲，那么输出脉冲将立即停止，如图 6.1.4

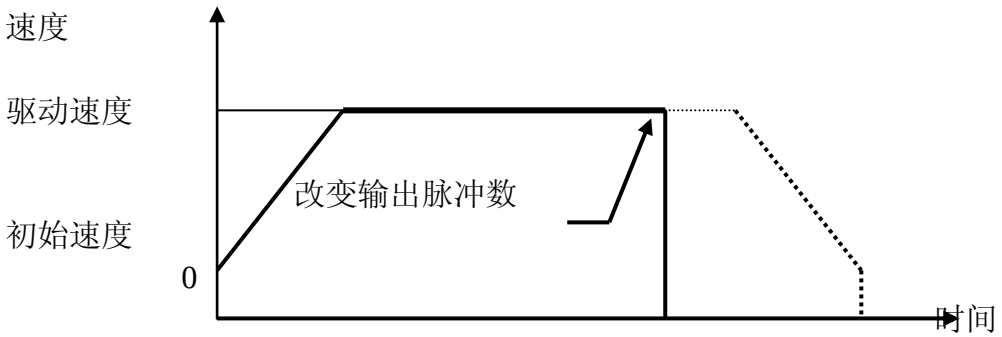


图 6.1.4 减少脉冲数

6.1.2 手动设置定长脉冲加/减速驱动时的减速点

通常定长脉冲加/减速驱动时的减速是由本卡自动控制的，如图 6.1.1 所示。但是在直线定长脉冲加/减速驱动中速度改变过于频繁时，应由用户自行预先设定减速点。

6.1.3 加/减速驱动的缓冲值设置

用户在定长脉冲驱动情况下可以改变加/减速点，如图 6.1.1 所示。本卡将自动地计算出加/减速点，并且使加速段的脉冲数等于减速段的脉冲数。

当为减速设置缓冲值（shift pulses）时，本卡将会因为缓冲值提前开始减速。减速完成后剩余的脉冲数(shift pulses)将会以初始速度输出，如图 6.1.5。本卡初始化时，缓冲脉冲数(shift pulses)的默认值为 8。在直线加/减速定长脉冲驱动时并不需要改变缓冲脉冲数。

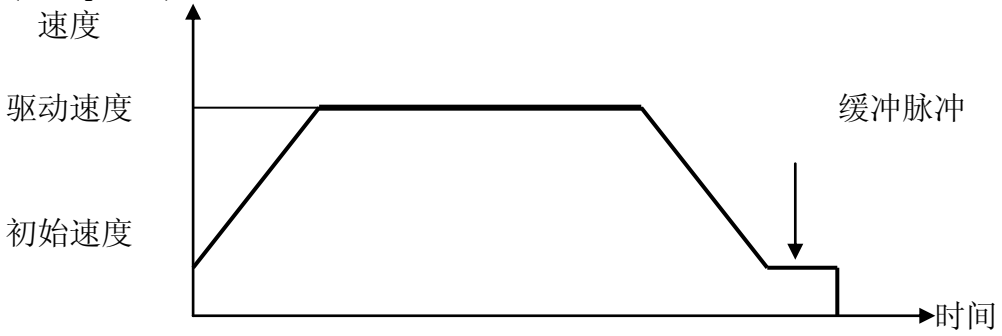


图 6.1.5 定长方式下的缓冲脉冲

6.2 连续脉冲驱动输出

当将本卡的脉冲输出模式设置为连续驱动状态时，本卡将一直以特定的速度驱动脉冲输出直至接收到停止命令或是外部停止信号，如图 6.2.1 所示。

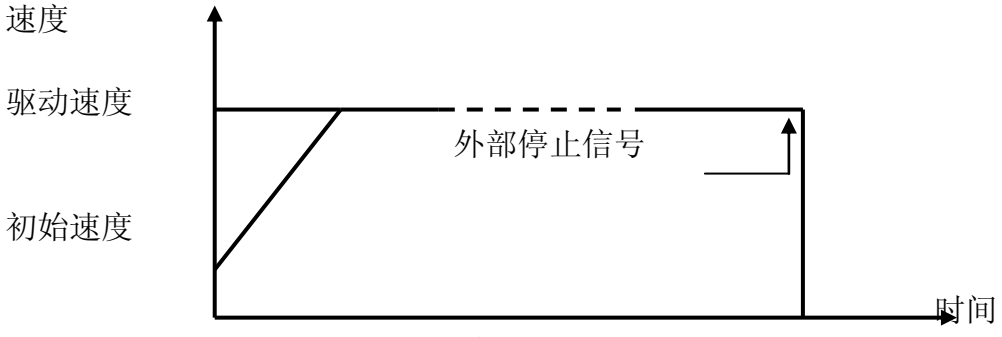


图 6.2.1 连续脉冲驱动

可用”减速至停 DecStop”和”立即停止 InstStop”等函数中断连续驱动脉冲；也可由外部信号使其制动。

6.3 恒速驱动

当本卡的驱动速度设置得低于初始速度时，它并不执行加/减驱动，而是开始恒速驱动，如图 6.3.1。

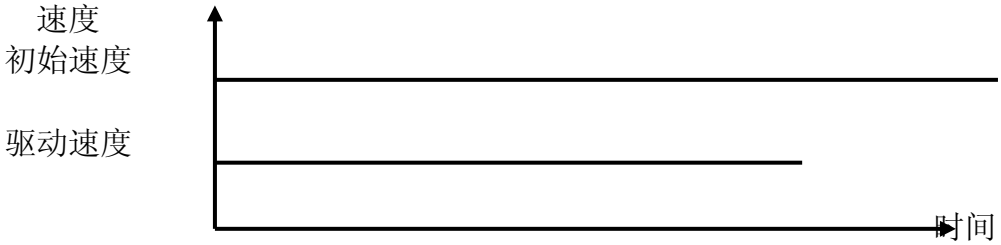


图 6.2.1　恒速驱动

6.4 线性加/减速驱动

线性加/减速驱动按线性规律将输出脉冲速度从初始速度增加至所需要的速度。同时，本卡还会记录下加速段的脉冲数；并与剩余的脉冲数进行比较；当剩余脉冲数小于加速脉冲数时，本卡便开始从驱动速度减速至初始速度。

当在加速驱动过程中出现减速指令，或当定长脉冲数小于所需要的驱动速度的时候。则本卡将在加速阶段便开始减速，如图 6.4.1 所示。通常，加速脉冲数和减速脉冲数是相等的。但是，当使用手动减速时(HandDec)则就不同了。

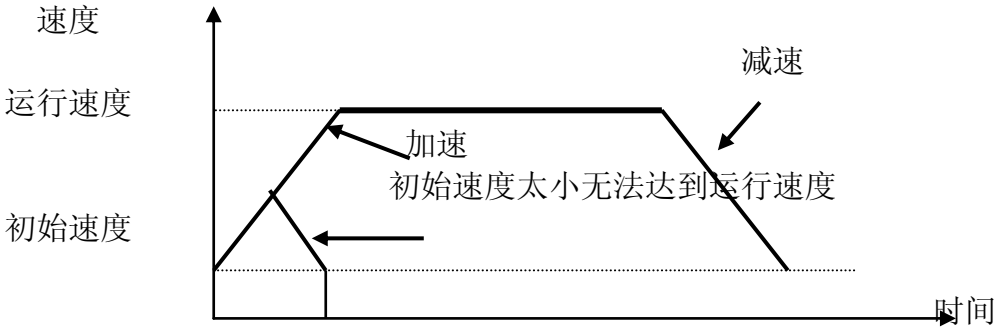


图 6.4.1　直线加/减速

6.5 线性加/减速参数设定示例

如图 6.5.1,从实际初始速度 500 PPS 加速至 15000 PPS，时间为 0.3 S (秒)

则　输出频率的倍数 M=2（初始化后默认值为 M=1）

初始速度　　SV=500 PPS/M=250 PPS

驱动速度　　V=15000PPS/M=7500 PPS

加速度　　A=[(15000 - 500)PPS /0.3S]/M=24167 PPS/S

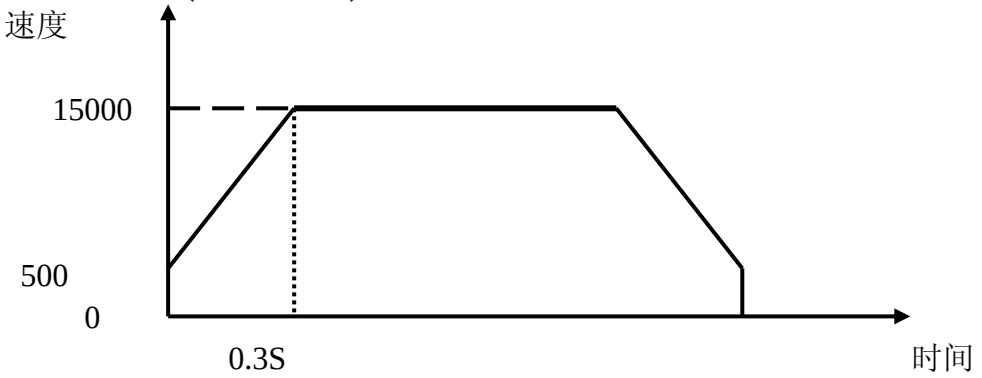


图 6.5.1 定长脉冲驱动模式

如图 6.5.2,从实际初始速度 500 PPS 加速至 15000 PPS，时间为 0.3 S (秒)
则　最高输出频率的倍数 M=2（初始化后默认值为 M=1）
初始速度　　SV=500 PPS/M=250 PPS
驱动速度　　V=15000PPS/M=7500 PPS
加速度　　A=[(15000 - 500)PPS /0.3S]/M=24167 PPS/S
减速度　　D=[(15000 - 500)PPS /0.1S]/M=72500 PPS/S

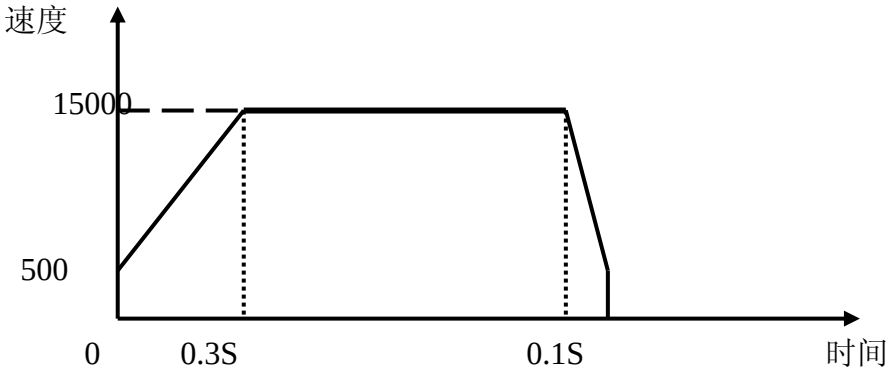
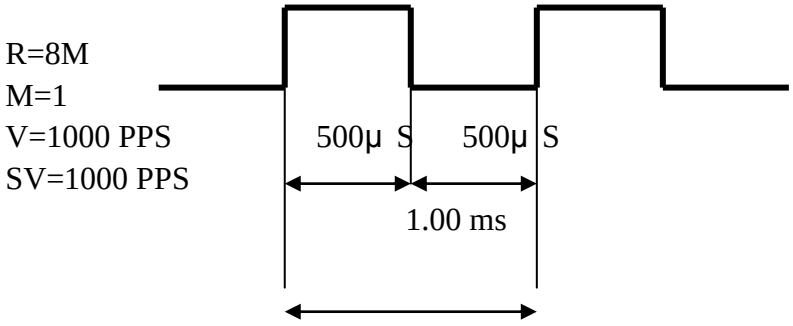


图 6.5.2 定长脉冲驱动模式加/减速度不同

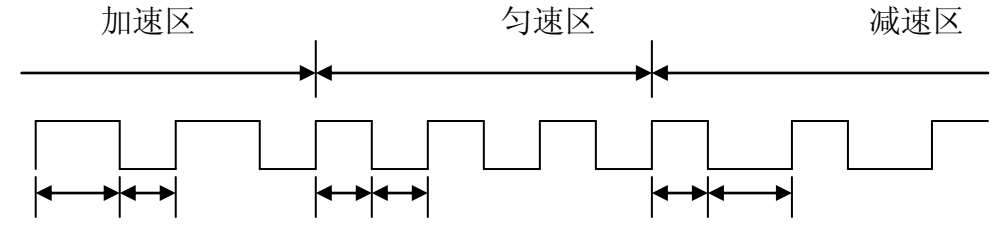
6.6 脉冲宽度和速度的精度

6.6.1 驱动脉冲的占空比

每个轴的正或负方向的驱动脉冲时间周期是由本卡的系统时钟 CLK(16MHz)决定的。这个时间周期有一个 SCLK 的误差。误差为±125ns。每个脉冲的占空比都(如图 6.6.1)为 50%。当主频被设置为 R=8M，M=1（倍数）V=1000 PPS 时，驱动脉冲：500μ S 处于高电平，500μ S 处于低电平；整个脉冲周期为 1 ms。



当处于加速时，低电平脉冲长度小于高电平脉冲长度；驱动速度将会提高，反之，当处于减速时，低电平脉冲长度大于高电平脉冲长度；驱动速度将会降低。



THA　TLA　　THC TLC　　THD TLD
THA >TLA　　THC =TLC　　THD <TLD

6.6.2 驱动速度精度

因为本卡的 CLK 时钟为 8MHz，因此，用户最好将速度（输出脉冲频率）设在 CLK 周期（125ns）的整倍数上；否则，驱动脉冲将可能不稳定。本卡初始化后，最高速度（输出脉冲频率）默认值为 8K。驱动速度越高，精度越低。但是，即使驱动速度很高，本卡仍就能保持相对的精度；驱动脉冲的精度仍在±0.1 之内。不会影响驱动电机的工作状态，因为这个误差是会被电机系统的惯性吸收的。

最高输出脉冲频率可以通过 SetM 函数设置，默认值 M=1，默认最高输出脉冲频率 8K。

倍 数 M	最 高 输 出 频 率 /PPS
M=1	MAX(V)=8000
M=2	MAX(V)=16,000
M=3	MAX(V)=24,000
…	… …
M=500	MAX(V)=4,000,000

当 M 被设置为 500 时，最高输出频率为 4M。加/减速度也随着 M 的数值变化而变化

6.6.3 逻辑位置计数器和实际位置计数器

KPCI-884 对每一个轴都有一个逻辑位置计数器和实际位置计数器。当发出一个正向脉冲时，计数器将自动加 1，反之，减 1。

6.6.4 比较寄存器和软件限位

KPCI-884 对每一个轴都有 2 个 32 位寄存器（上下限位寄存器）用来与逻辑位置计数器或实际位置计数器进行比较。当逻辑位置计数器或实际位置计数器的数值达到限位寄存器时，将会减速至停机。

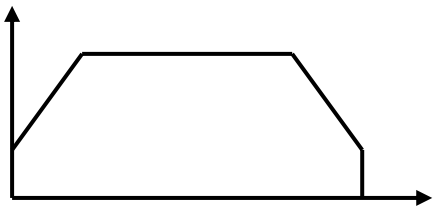


图 6.6.4 软件限位停机

6.7 硬件限位

硬件限位信号（nLMT±）输入端用来终止脉冲输出。

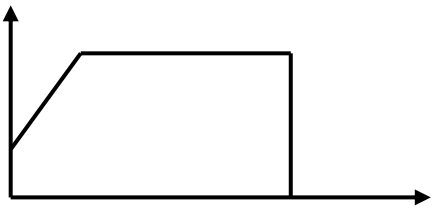


图 6.7.1 硬件限位停机

6.8 紧急停车

KPCI-884 有一个用于急停的输入端 J1-44（EMG）。正常状态为高电平；当急停信号 EMG 变位低电平时，所有轴将立即停止

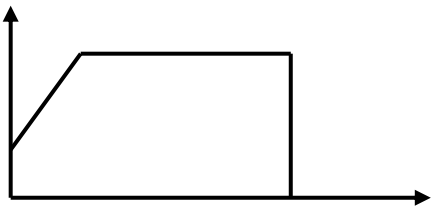


图 6.8.1 EMG 为低电平紧急停机

6.9 插补

KPCI-884 可实现任意 2 轴或 3 轴的直线插补，任意 2 轴的圆弧插补。插补运动是指任意 2 轴或 3 轴按照一定的算法进行联动，被控轴同时启动，并同时到达目标位置。对于直线插补，圆弧插补，最大驱动速度为 4 MPPS。

在直线插补过程中，如果触发硬件限位或软件限位，则会立即停止直线插补驱动。而且任何轴的任何一个限位有效时，都会导致本卡停止插补。但是，在圆弧插补时，硬件或软件的信号将不会中断插补驱动。

6.9.1 长轴和短轴

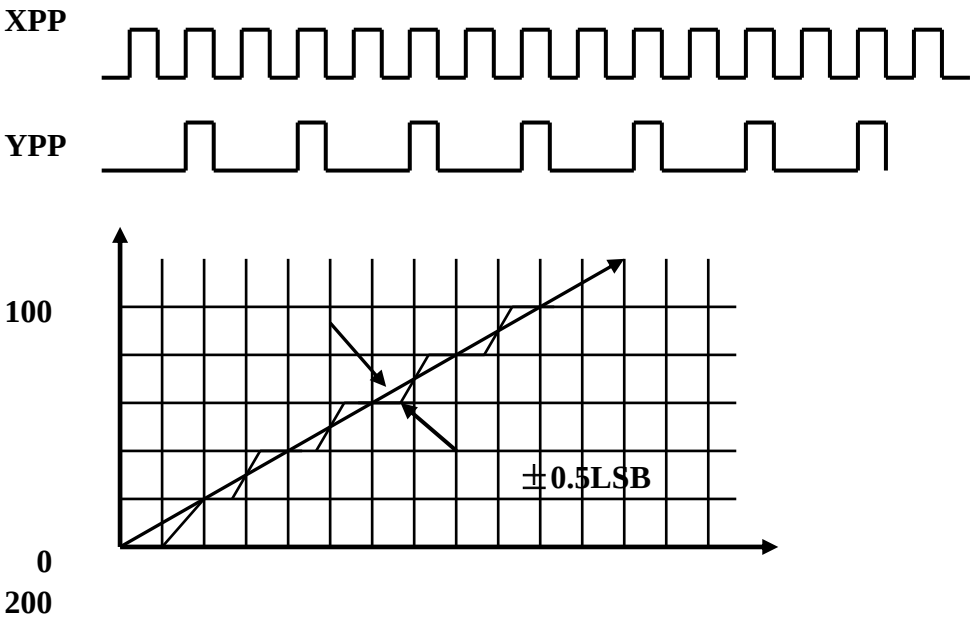
在插补中，移动距离最长的轴为”长轴”，另外的轴为”短轴”。”长轴”输出一个均匀的脉冲序列，而”短轴”的驱动脉冲依赖于”长轴”和 2 轴之间的关系。

6.9.2 插补的运动方向

对于单轴的非插补驱动，输出的脉冲数是无符号的，其方向是由函数中的 fx 参数所控制的。对于插补，输出脉冲数是有符号的。范围是从-8 386 607 到+8 386 807

6.9.3 2 轴或 3 轴的直线插补

X,Y 2 个轴的直线插补，从当前位置到相对位置（X：+200，Y：+100）



6.9.4 圆弧插补

圆弧插补可以从 4 轴中任选 2 轴进行。从当前位置开始，根据所指定的圆心和终点位置以及插补的方向(按顺时针或逆时针)来进行。**坐标设定值是对当前坐标的相对值。**图 6.9.1 说明了顺时针和逆时针插补的定义。”长轴”定义为 X 轴

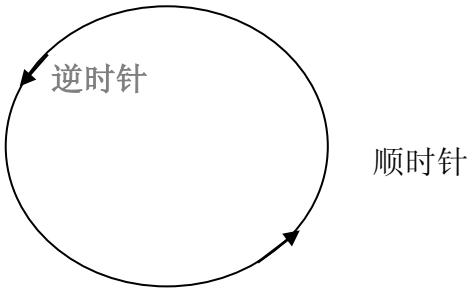
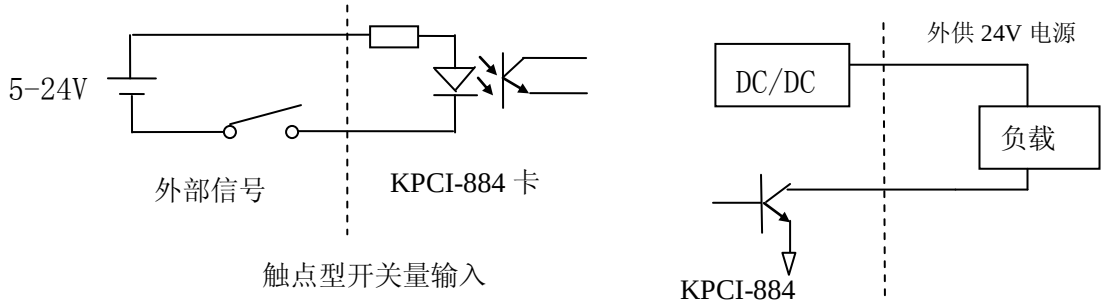


图 6.9.1

6.10 通用输入/输出

KPCI-884 对于每一个轴都有 3 个通用开关量输入和 4 个通用开关量输出点。



集电极反向输出，提供 20 毫安的灌电流

6.11 中断

中断可由 X，Y，Z，U 轴和位模式插补、连续插补产生，但是所有的中断信号共享计算机 CPU 的同一个中断号。对于各种中断情况的区别及编程，请参考如下的使用说明和函数说明。

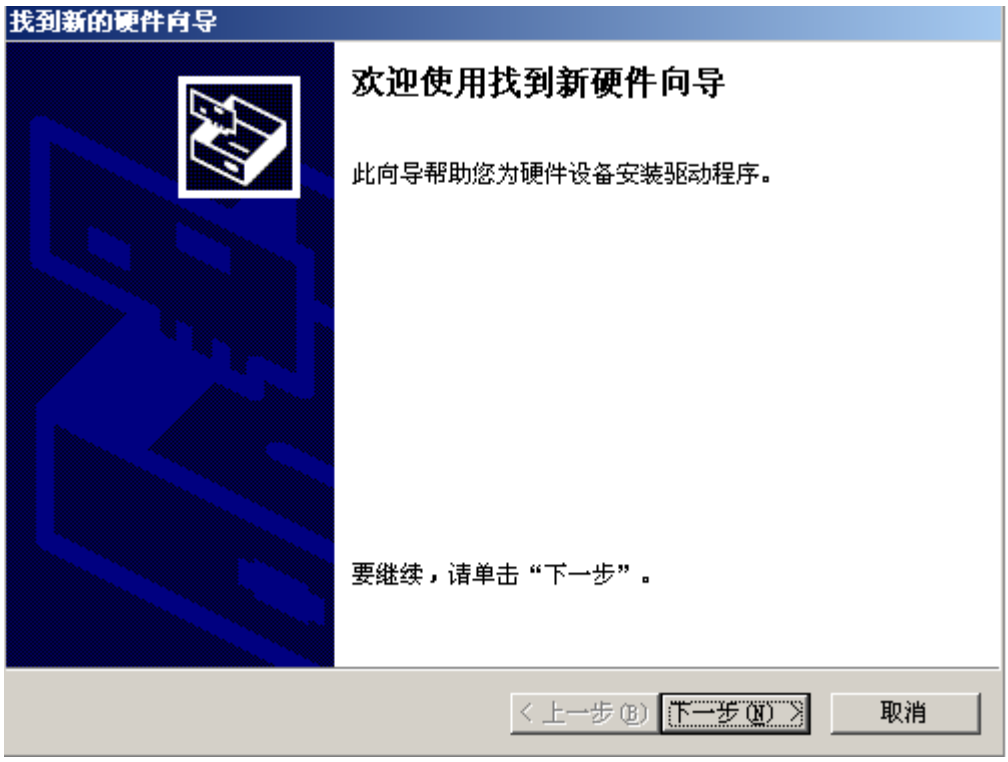
第七章 库函数 KPCI-884.dll 的安装和使用说明

7.1 Windows9X 系统环境下的安装

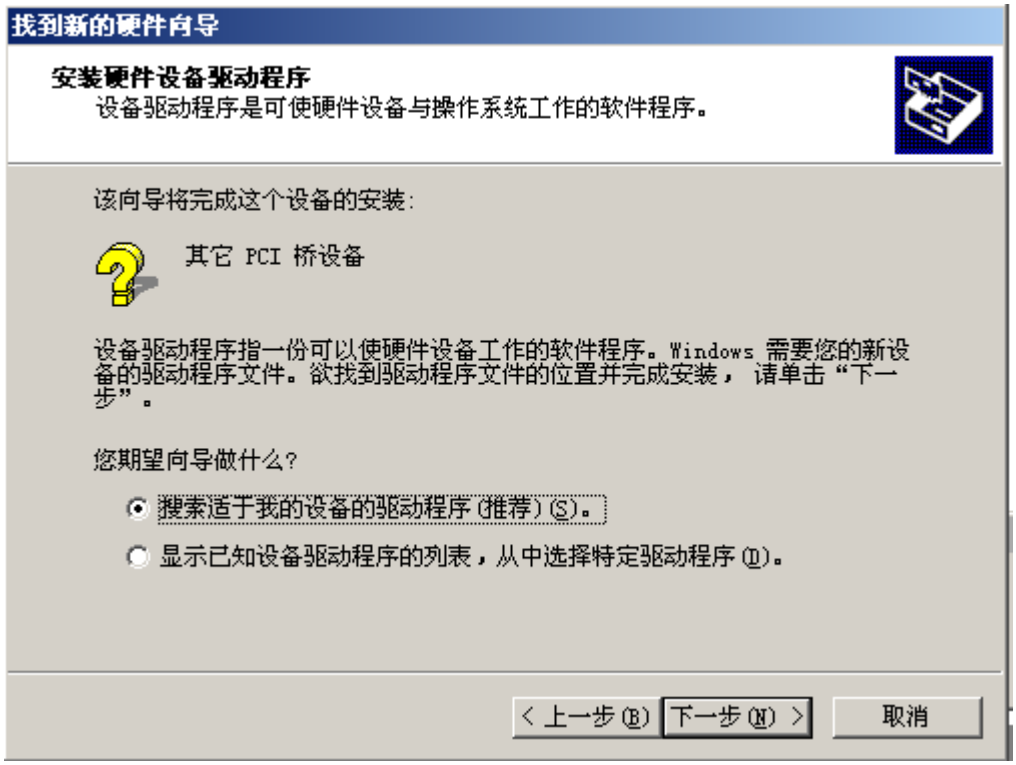
Windows98 下的安装跟 Windows2000 下的安装类似，请参考以下的安装过程

7.2 Windows 2000（XP）系统环境下的安装

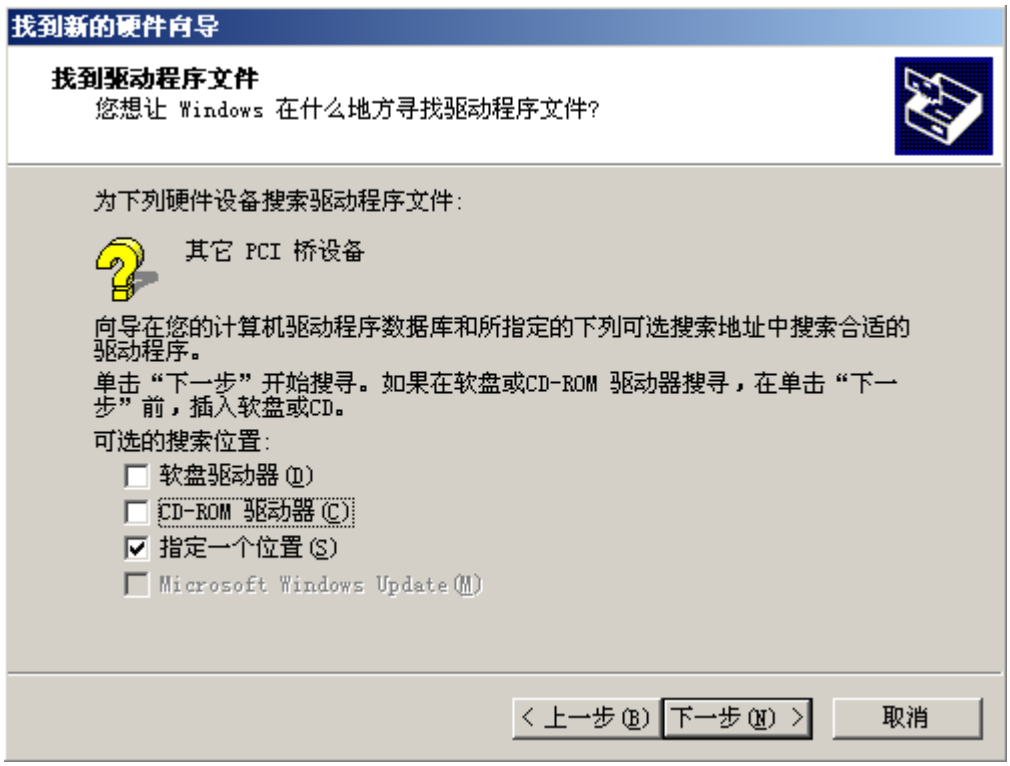
如果是 Windows 2000（或 XP）系统环境，插上板卡，启动计算机，进入 Windows2000(或 XP)系统。则系统自动弹出发现新设备提示框：



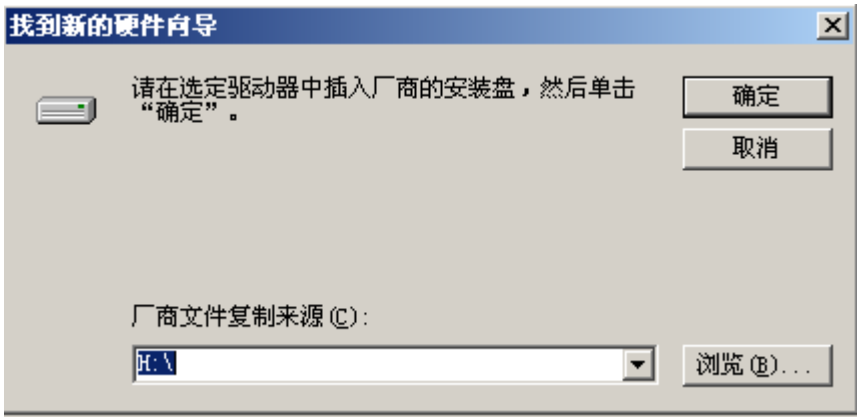
点击“下一步”即可，此时，系统出现如下的窗体：



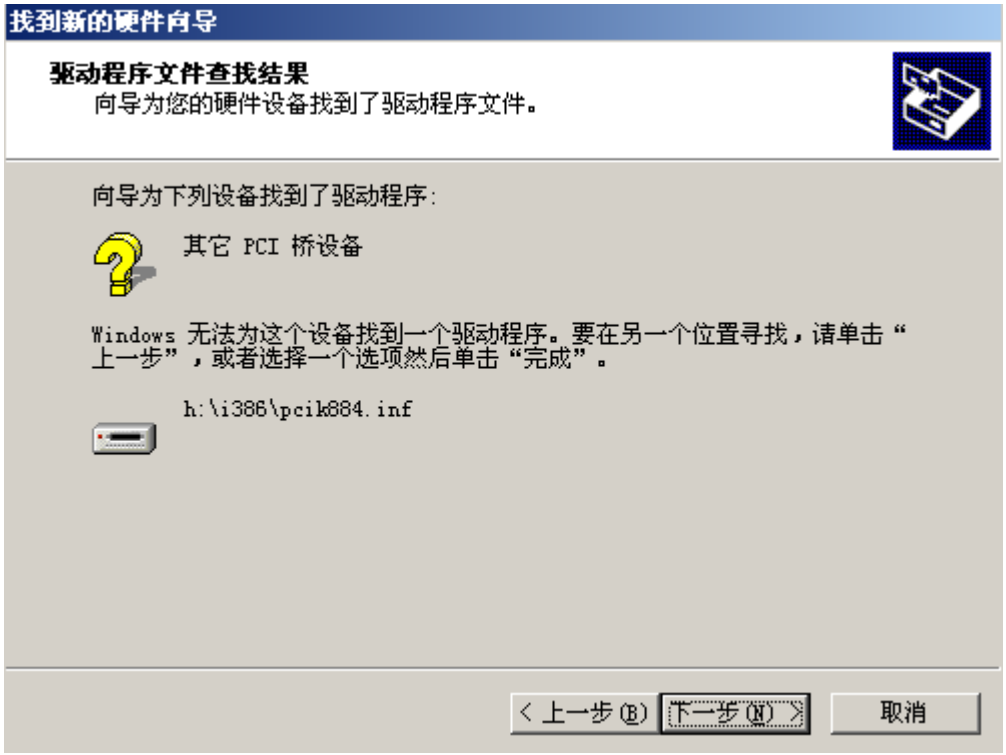
点击“下一步”即可，此时，系统出现如下的窗体：



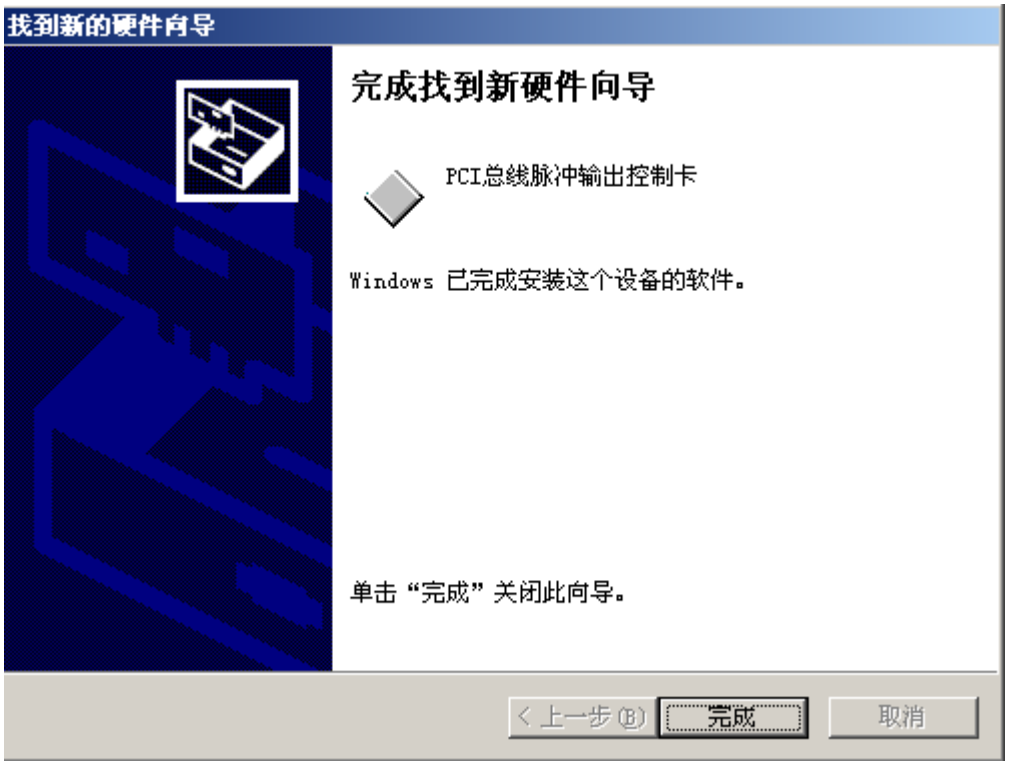
点击“下一步”即可，此时，系统出现如下的窗体：



点击“浏览”，在科日新的驱动安装光盘里边“步进伺服电机运动控制卡/KPC-I884 步进伺服电机运动控制卡/驱动程序/Win2000”文件夹，选定“PCIK884.inf”文件，然后点击确定。

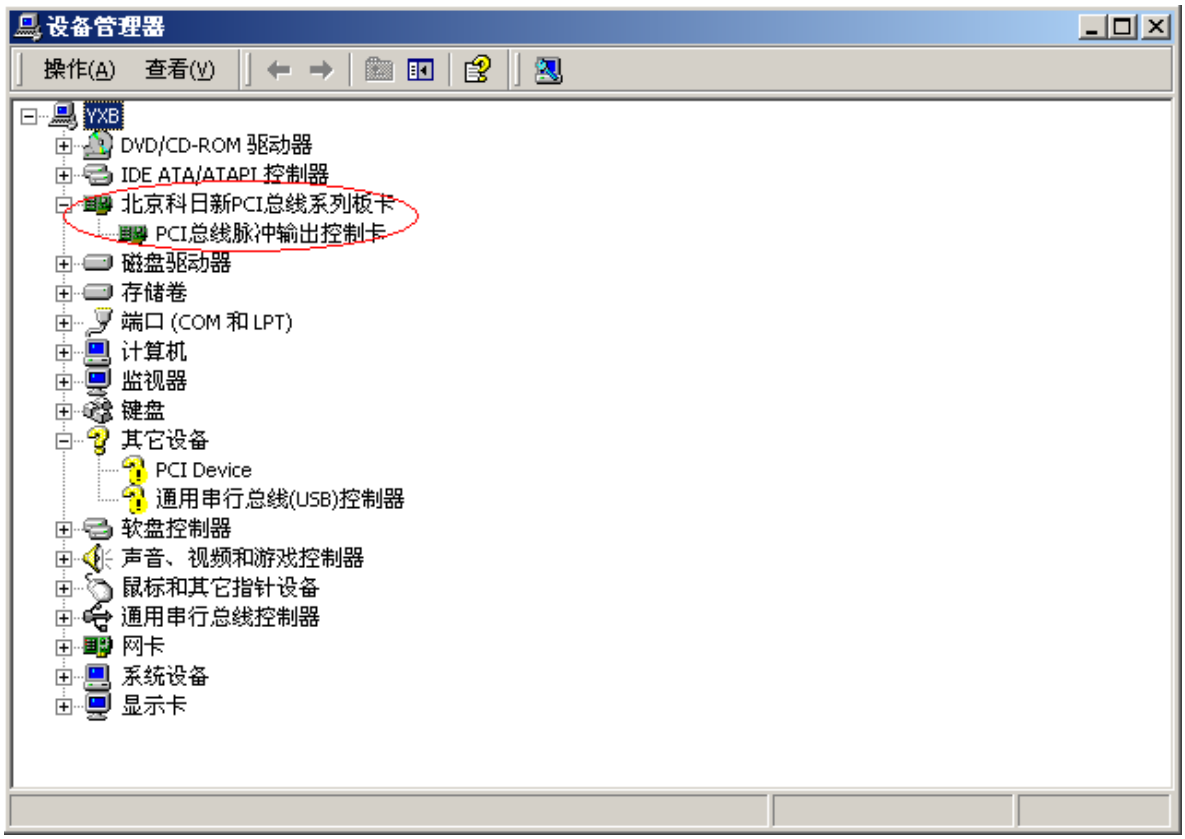


点击“下一步”即可，此时，系统出现如下的窗体：



单击完成按钮。

确认板卡安装成功：打开“控制面板→系统→设备管理器”，如下图所示：



进入“光盘/步进伺服电机运动控制卡/KPC-I884 步进伺服电机运动控制卡/动态连接库”里面有函数库。进入“光盘/步进伺服电机运动控制卡/KPCI-884 步进伺服电机运动控制卡/示例程序”里面有编程示例。

第八章　VB 下调用 DLL 内部函数说明

使用 VB 时；在窗体的通用说明区对 KPCI884.DLL 库函数进行说明，这样用户就可在 VB 下调用 KPCI884 的库函数，对 KPCI884 卡进行编程了。

Public Declare Function 函数名 Lib "KPCI884.DLL" (ByVal num As Integer, ByVal adr As Integer) As Integer

例 1：Public Declare Function Init884 Lib "KPCI884.DLL" (ByVal num As Integer) As Integer

```
… …
return_code=Init884 (0)          ‘该卡定义为 0 号，
… …
```

例 2：Public Declare Function PulseOutMode Lib "KPCI884.DLL" (ByVal num As Integer , ByVal axis As Integer, ByVal mode As Integer) As Integer

```
… …
return_code=PulseOutMode ( 2 ,3, 1);     // 将 2 号卡的 3 号轴置为 Pulse\DIR 方式工作
… …
```

VB 中的声明格式如下：

当动态连接库存放在系统目录里的时：Public Declare Function 函数名 Lib "KPCI884.DLL" (ByVal num As Integer,ByVal adr As Integer) As Integer

当动态连接库存放在某工作目录下如“D:\DRXIC”下时：Public Declare Function 函数名 Lib " D:\DRXIC\KPCI884.DLL" (ByVal num As Integer,ByVal adr As Integer) As Integer

(1) Public Declare Function Init884 Lib " KPCI884.DLL" (ByVal num As Integer, ByVal adr As Integer) As Integer (点击超连接处可以查看函数原型)

VB 中调用此函数进行板卡初始化时，采用 CALL 指令：

Call Init880(1, &H300) 表示初始化板卡号为 1 地址为 H300 的卡。

(2) Public Declare Function PulseOutMode Lib " KPCI884.DLL" (ByVal num As Integer, ByVal axis As Integer, ByVal mode As Integer) As Integer

VB 中调用此函数进行轴模式操作时，如下：

Call PulseOutMode(2,3,1) 表示将 2 号板卡的 3 号轴设置成 Pulse/DIR 工作方式。

(3) Public Declare Function Line2D Lib " KPCI884.DLL" (ByVal NUM As Long, ByVal axis1 As Long, ByVal axis2 As Long, ByVal m_v_a As Long, ByVal m_v_sv As Integer, ByVal m_v_v As Integer, ByVal m_v_p1 As Long, ByVal m_v_p2 As Long) As Integer

VB 中调用此函数进行 2 轴直线插补时，如下：

Call Line2D(Num, 1, 3, a, sv, v, m_v_p1 , m_v_p2) 表示在 1 号主轴即 X 轴、3 号从轴即 Z 轴所处的平面即 ZOX 平面上进行直线插补，初始点为当前点，目标点为 m_v_p1 和 m_v_p2，a 为加速度，sv 为初始速度，v 为驱动速度。

(4) Public Declare Function Circle_ConstSpeed Lib " KPCI884.DLL" (ByVal NUM As Long, ByVal axis1 As Long, ByVal axis2 As Long, ByVal fm As Long, ByVal m_c1 As Long, ByVal m_c2 As Long, ByVal m_p1 As Long, ByVal m_p2 As Long, ByVal m_v_sv As Integer, ByVal m_v_v As Integer) As Integer

VB 中调用此函数进行圆弧插补时，如下：

Call Circle_ConstSpeed(Num, 1, 3, fm, 5000, 0, 5000, -5000, 1000, 1500)表示在 1 号主轴即 X 轴、3 号从轴即 Z 轴所处的平面即 ZOX 平面上进行圆弧插补，起始点为当前点，fm 表示插补的顺逆方向，m_c1、m_c2 表示圆心，m_p1、 m_p2 为目标点。

(4)带有指针函数的函数 BPInter2D 的声明与调用：

函数原形： extern "C" __declspec(dllexport) int __stdcall BPInter2D(int num,double(CALLBACK *pfun)(double),long *px0,long *py0,long N,UINT pm,unsigned short m_v_a,unsigned short m_v_sv,unsigned short m_v_v)

函数声明：vb 中调用动态连接库 KPCI884.DLL 中的函数 BPInter2D（）时，须在.BAS 中声明为：Public Declare Function BPInter2D Lib " KPCI884.DLL"(ByVal NUM As Integer, ByVal pfun As Long, px0 As Long, py0 As Long, ByVal N As Long, ByVal pm As Integer, ByVal m_v_a As Integer, ByVal m_v_sv As Integer, ByVal m_v_v As Integer) As Integer

详细说明以及具体用法：

（1）：带回调功能的函数指针 double(CALLBACK *pfun)(double)在 vb 中只能声明为 LONG 型，即：ByVal pfun As Long。当要调用 BPInter2D 函数时，需要传递指针函数 pfun 的地址，需要添加关键字 AddressOf，即 AddressOf pfun。经初步测试，带有 CALLBACK 关键字的 BPInter2D 函数在 vc 中调用时正常工作。而 vc 中声明为 long *px0,long *py0 的两个指针在 vb 中可以引用传地址关键字 ByRef 来声明 ByRef px0 As Long, ByRef py0 As Long。

（2）：因为用到了回调函数 CALLBACK，因此在 vb 中声明时代码必须将回调函数放在标准的.BAS 模块中，而不可放在窗体及附加类模块中。其所调用的函数也必须放在该模块中。如：在模块中添加一个正切函数 f（x）

声明和调用	注释
Function fun(ByVal x As Double)As Double fun=Tan(x/100) End Function	注意：Tan（number）：必要的 number 参数是 Double 或任何有效的数值表达式，表示一个以弧度为单位的角度。Tan（）的返回值就是一个 Double 类型，所以也不需要转换。

```
'如果想用2位模式插补得到一个正切函数曲线的一部分
'x属于[10，80]，即tan(0.1)~tan(0.8)。
'可在窗体中输入一下内容:
Dim Num As Integer          '注: 卡号 (1 - 8)
Dim px, py, m As Long       '注：px为函数自变量x的初始点10
                            'py=tan(px)=tan(10/100)。
Dim N As Integer            '注：N为插补自变量变化方向与范围
                            '（80-10）=+70，正号表示是向x轴的
                            '正方向插补，70表示插补距离。


Dim pm As Integer '注：pm为插补平面1：XOY平面，
                  '2：ZOX平面，3：YOZ平面

        Num = 1
        px = 10
        py = Tan(10 / 100)
        N = 70
        pm = 3
```

调用 Call BPInter2D(Num , AddressOf fun, px, py, N, pm , 250, 1000, 2000)

第九章　VC 函数调用举例说明

9.1 使用板卡前，首先要使用 Init884 函数对该卡进行初始化，对各电机轴的控制是通过板卡号和轴号（1-4）确定的。例如：如果用户要使用第 2 块卡控制电机运动，首先要利用拨码开关，对卡号设置。（参考本书的 3.1 部分）例如：

```
int return_code;
num=2 ;                                  //卡号
return_code= Init884 ( num);             //初始化函数，返回值于 return_code 中
```

9.2 使用 Move_DV 定长脉冲驱动函数启动电机：

int Move_DV(int num,unsigned short axis,unsigned short m_v_a,unsigned short m_v_sv, unsigned short m_v_v,int fx,long m_v_p);

1:　num　卡号

2：axis　轴选择（轴选择 1：X 轴，2：Y 轴，3：Z 轴，4：U 轴）

3：m_v_a　加速度，（范围-1 000 000 → -125; +125 →+1 000 000）

4：m_v_sv 初始速度（范围 1-8000）

5：m_v_v　驱动速度（范围 1-8000）

6：fx　方向：1：正方向，0：负方向

7：m_v_p　定长驱动脉冲数。（范围 1 →2 147 483 647）

例如：要使 2 号卡的 3 号电机以 Pulse\DIR 方式；100 PPS(脉冲数/秒)初始速度；1500 PPS 驱动速度；加速度为 1400 PPS/s；正方向转动；脉冲数为 10000；(输出频率的倍数 M=1,默认值) 参考下例：

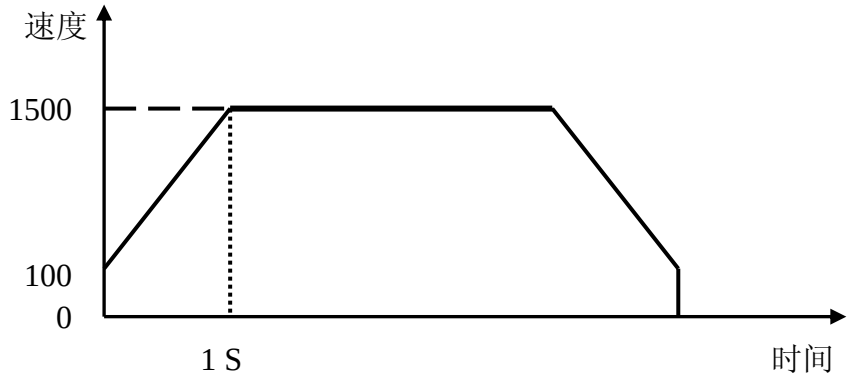


图 7.1.1

```
int return_code;
unsigned short num;                          //卡号
unsigned short axis;                         //轴号
unsigned short m_v_a;                        //加速度
unsigned short m_v_sv;                       //初始速度
unsigned short m_v_v;                        //驱动速度，
int fx;                                      //方向
long m_v_p;                                  //脉冲数
num=2 ;                                      //卡号
   axis=3 ;                                  //轴号
m_v_a=1400;                                  //加速度
m_v_sv=100;                                  //初始速度
m_v_v=1500;                                  //驱动速度
fx=1;                                        //正方向
m_v_p=10000;                                 //脉冲数
return_code= PulseOutMode ( 2 ,3, 1);        // Pulse\DIR 方式工作
return_code= Move_DV ( num, axis, m_v_a, m_v_sv, m_v_v, fx, m_v_p);
// 0：成功
//-1：卡号错误
//-2 ：轴号错误
//-3：数据范围超出。
…     …
```

注意：需要停止电机转动时；可使用 DecStop(int num,unsigned short axis)函数,(函数声明略) DecStop(2,3)。

9.3 使用 Move_LV 连续脉冲驱动函数启动电机：

int Move_LV(int num,unsigned short axis,unsigned short m_v_a,
unsigned short m_v_sv,unsigned short m_v_v,int fx);

1: num 卡号

2：axis 轴选择（轴选择 1：X 轴，2：Y 轴，3：Z 轴，4：U 轴）

3：m_v_a 加速度设置，（范围-1 000 000 → -125; +125 →+1 000 000）

4：m_v_sv 初始速度设置（范围 1-8000）
5：m_v_v 驱动速度设置（范围 1-8000）
6：fx 方向设置：1：正方向，0：负方向。
例如：要使 3 号卡的 1 号电机以 CW/CCW 方式；200 PPS(脉冲数/秒)初始速度；2000 PPS 驱动速度；加速度为 400 PPS/s；负方向连续运动，当输出的脉冲数达到 24000 时；紧急停车。可参考下例程序：

```
   … …
int return_code;
   long position;
unsigned short num;
unsigned short axis;
unsigned short m_v_a;
unsigned short m_v_sv;
unsigned short m_v_v,int fx;
long m_v_p;
num=3 ;                                  //卡号
   axis=1 ;                              //X 轴
m_v_a=1200;                              //加速度
m_v_sv=200;                              //初始速度
m_v_v=2000;                              //驱动速度
fx=0;                                    //负方向
if(! PulseOutMode( num, axis, 0))        // CW/CCW 方式工作
{ position=24000;
SetLP(num,axis,0);                       //逻辑位置计数器清零
return_code= Move_LV( num, axis, m_v_a, m_v_sv, m_v_v, fx, m_v_p)；//启动电机
//return_code 返回码 0：成功
//-1：卡号错误
//-2 ：轴号错误
//-3：数据范围超出。
while (position < ReadLP(num,axis))      //输出的脉冲数是否达到 24000
    {
      … …
… …
    }
InstStop( num, axis)；                   //紧急制动电机
}
```

注意：读取已输出的脉冲数时； 应先用 SetLP 函数对逻辑位置计数器清零 。

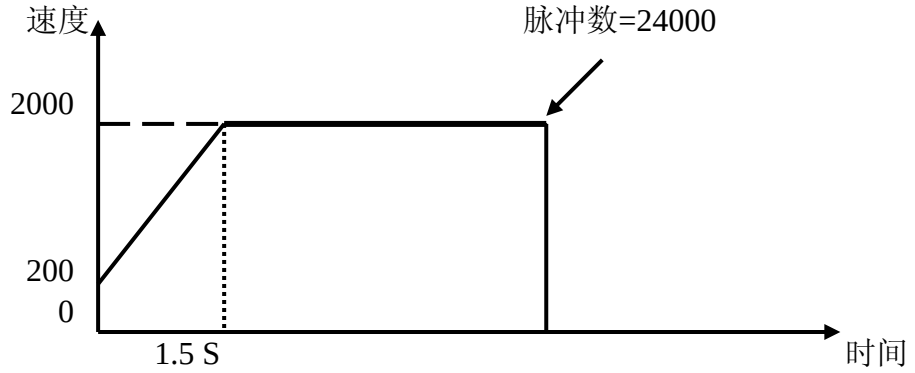


图 7.1.2

如果设置软件限位,可以使用”SetCP”函数执行：

```
… …
PulseOutMode( num, axis, 0);                    // CW/CCW 方式工作
position=24000;
SetLP(num,axis,0);                              //逻辑位置计数器清零
SetCP(num,axis,position);
return_code= Move_LV( num, axis, m_v_a, m_v_sv, m_v_v, fx);
//启动电机
```

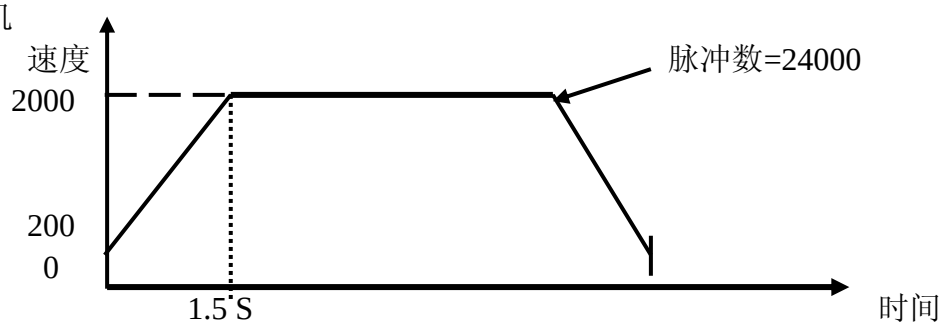


图 7.1.3

例如: 要读取第 2 块卡的 4 号电机（U 轴）的输出脉冲数

```
long data;
SetLP(2,4,0);
……
Move_LV(…);
……
data=ReadLP(2,4);
```

9.4 使用其他函数的组合可以更灵活的控制电机

Init884，PulseOutMode,SetSV，SetV，SetA，SetP，Start_DV。

例如：要使 1 号卡的 4 号（U 轴）电机以 Pulse/DIR 方式；100 PPS 初始速度；1000 PPS 驱动速度；1800 PPS/s 加速度；正方向运动 5000 步后；再以 2000 PPS 驱动速度；5000 PPS/s 加速度；运转 12000 步；(输出频率的倍数 M=1,默认值) 参考下例：

```
    … …
PulseOutMode( 1, 4, 1);                         // Pulse/DIR 方式工作
Move_DV( 1, 4, 1800, 100, 1000, 1, 5000);       //启动
```

```
            While(! Stopped(1, 4) )                    //判断电机是否运转结束
{                                                      //否,未停机
    if (kbhit( ) )                                     //键盘输入中断运行
      {
         getch( );
         exit(1);
      }
      }
       SetA(1, 4, 5000);                               //设置加速度 5000
  SetV(1, 4, 2000);                                    //驱动速度 2000
  SetP(1, 4, 12000);                                   //脉冲数 12000
  Start_DV(1, 4, 1);                                   //启动
```

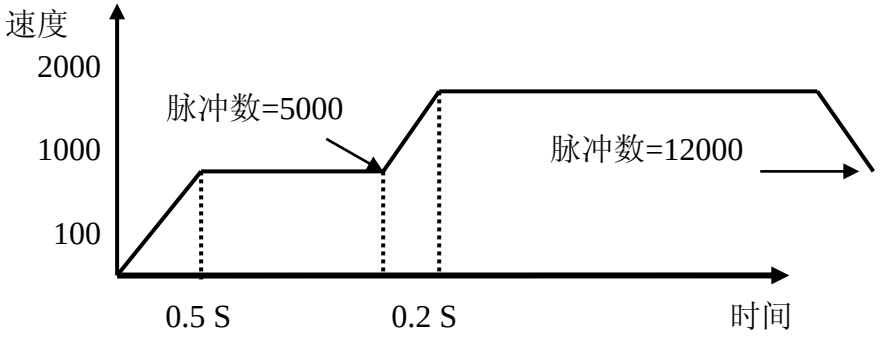


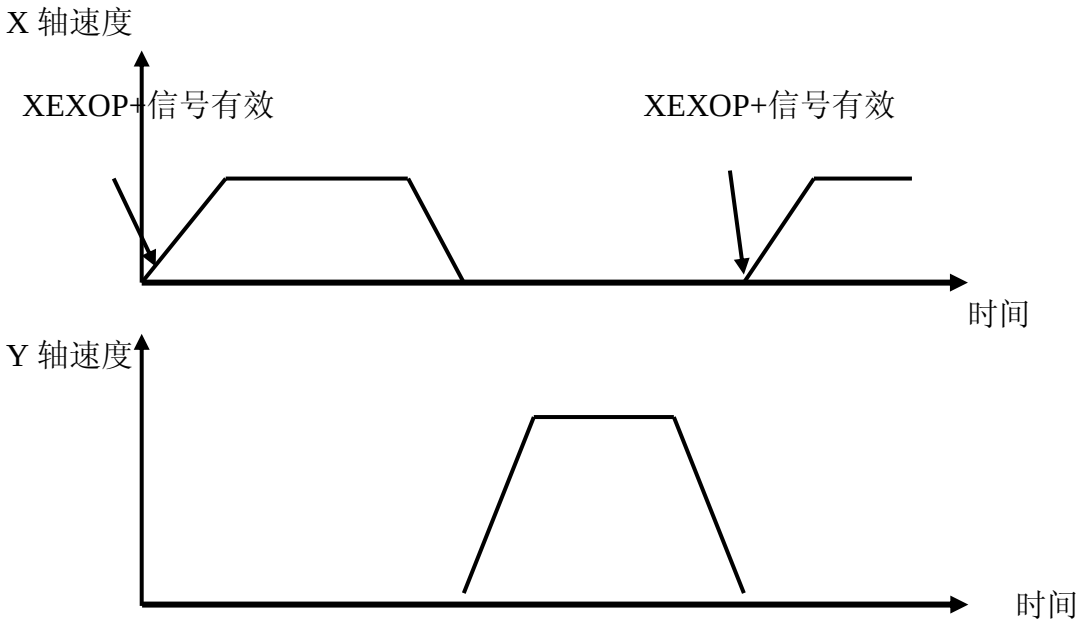
图 7.1.4

9.5 当用户使用软限位控制电机时，应首先使用 ReadCOMP 函数读取某轴的限位状态，如果：返回值=1，达到上限位；返回值=2，达到下限位。
例如：判断达到限位后，X_out3 输出“1”。

```
int num,axis_x;X_out3;
num=1;
axis_x=1;
X_out3=3;
… …
if((ReadCOMP（num,axis_x）==1)or(ReadCOMP（num,axis_x）==2))
{
… …
SetOUTBIT(num,axis_x,X_out3,1);                  // X_out3 输出“1”
… …

 }
```

9.6 当外部端子 XEXOP+信号(J2 端子的 33 脚)有效时(低电平)；启动 2 号卡的 1 号电机（X 轴）以 Pulse/DIR 模式；100 PPS 初始速度；1000 PPS 驱动速度，1800 PPS/s 加速度；正方向运行 4000 步后，启动 2 号电机以 Pulse/DIR 模式；100 PPS 初始速度；2400 PPS 驱动速度，1960 PPS/s 加速度；正方向运行 2000 步后，再使 XEXOP+信号有效。(输出频率的倍数 M=1,默认值)




```
int y_position;
y_position=2000;
PulseOutMode( 2, 1, 1);                  // Pulse/DIR 方式工作
Set_DV( 2, 1, 1800, 100, 1000,4000);     //设置 X 轴参数
Set_DV( 2, 2, 1960, 100, 2400,2000);     //设置 Y 轴参数
SetLP(2,1,0);                            //X 轴逻辑位置计数器清零
DV_OutEnable(2,1);                       // X 轴 EXOP 信号有效使能
SetLP(2,2,0);                            //Y 轴逻辑位置计数器清零
StopOutEnable(2,2);                      // Y 轴 EXOP 信号无效
While(ReadLP(2,1)>0 )                    // XEXOP+信号是否有效,电机是否启动
{                                        // XEXOP+信号有效
while (Stopped(2,1)==1)                  //X 轴电机停机
{
Start_DV(2,2,1);                         //启动 Y 轴电机
… …                                      //  使 XEXOP+信号无效，X 轴电机制动
while (y_position == ReadLP(2,2))        //Y 轴输出的脉冲数是否达到 2000
   {
… …                                      //  使 XEXOP+信号有效，电机再次启动
}
… …
}
… …
}
```

9.7 用 EXOP+外部信号可以实现电机的"点动"功能

用 ReadBIT 函数读取 I/O 端口状态：返回值是该端口某一位开关量的状态

例如：读取 1 号卡的 Y 轴的 Y_in1 和 Y_in2 的状态。如果 Y_in1 的状态是“0”且 Y_in2 的状态是“1”，则 X_out3 输出“1”；否则，X_out3 输出“0”。（注意：硬件 I/O 只有 3 个输入点；而输出有 4 个）

```
unsigned short num;
unsigned short axis;
unsigned short Y_in1,Y_in2,X_out3;
unsigned short y1,y2;
                    … …
num=1;                                  //卡号
axis_x=1 ;                              //X 轴
axis_y=2 ;                               //Y 轴
Y_in1=1;
Y_in2=2;
X_out3=3;
   … …
y1=ReadBIT(num,axis_y,Y_in1);
y2=ReadBIT(num,axis_y,Y_in2);
if (y1==0 && y2==1)
{                                       //Y_in1 的状态是“0”且 Y_in2 的状态是“1”
    SetOUTBIT(num,axis_x,X_out3,1)      // X_out3 输出“1”
}
else   SetOUTBIT(num,axis_x,X_out3,0)   // X_out3 输出“0”
                  … …
```

9.8 用 ReadBTYE 函数读取 I/O 端口状态：返回值是一个字节的整数（低 4 位有效）;D3,D2,D1 位为指定轴的 I/O 状态。第 1，2，3 位的值代表通用输入第 1，2，3 点的电平高低状态。例如：读取 2 号卡的 U 轴的 U_in1,U_in2 和 U_in3 的状态。如果 U_in1, U_in2, U_in3 的状态全是“1”;则

Z_out4, Z_out3,Z_out2, Z_out1 分别输出“1,0,1,0”;

U_out4, U_out3,U_out2,U_out1 分别输出“1,1,0,1”;

否则，Z_out4, Z_out3,Z_out2, Z_out1 分别输出“0,1,0,1”;

U_out4, U_out3,U_out2,U_out1 分别输出“0,0,1,0”

（注意：硬件 I/O 只有 3 个输入点；而输出有 4 个）

```
unsigned short num;
unsigned short axis_z,axis_u ;
   … …
num=2;                                  //卡号
axis_z=3 ;                              //Z 轴
axis_u=4 ;                              //U 轴
… …
if (ReadBYTE(num,axis_u)==0x07)         //二进制数为 00000111
      {                                 // U_in3,U_in2 和 U_in1 的状态全是“1”
          SetOUTBYTE(num,axis_z,0x0A);  // Z_out4, Z_out3,Z_out2, Z_out1
                                         //分别输出“1,0,1,0”
```

```
                                        //二进制数为 00001010
        SetOUTBYTE(num,axis_u,0x0D);   // U_out4, U_out3,U_out2, U_out1
                                        //分别输出“1,1,0,1”
      }                                 //二进制数为 00001101
  else  SetOUTBYTE(num,axis_z,0x05);    // Z_out4, Z_out3,Z_out2,Z_out1
                                        //分别输出“0,1,0,1”
                                        //二进制数为 00000101
SetOUTBYTE(num,axis_z,0x02);            // U_out4, U_out3,U_out2,U_out1
                                        //分别输出“0,0,1,0”
                                       //二进制数为 00000010
```

… …

9.9 中断函数使用方法：

用户可以选择使用本卡提供的 9 种中断源，并立即执行由此中断源产生后用户希望进行的动作。

9 种中断源分别是：

1: 产生一个增量脉冲，其上升沿会触发中断

2：P≥C-， 逻辑/实际位置计数器的值大于等于 COMP-寄存器的值时;

3: P<C-， 逻辑/实际位置计数器的值小于等于 COMP-寄存器的值时

4: P<C+， 逻辑/实际位置计数器的值小于等于 COMP+寄存器的值时

5: P≥C+， 逻辑/实际位置计数器的值大于等于 COMP+寄存器的值时

6：加减速驱动中，脉冲开始减速时

7：在加减速驱动中，加速过程结束时。

8：一次驱动过程结束时

9：连续插补中一次插补结束时

请注意下述过程编程

例：设先执行正向定长运动：

使用函数： Move_DV(1,1,125,500,1000,1,10000)

等到板卡驱动结束时，就执行反向定长运动，如下函数

Move_DV(1,1,125,500,1000,0,10000)

我们可以使用中断的方式完成此工作。选择使用中断源 8（一次运动完成产生中断），先编写一个中断服务子函数 Move()

```
void Move()
{
    Move_DV(1,1,125,500,1000,0,10000);      /反向定长运动
}
```

则在板卡初始化后执行下面的程序：

```
        void (*pf)();                       //指定中断服务子程序
       pf=Move;
SetInterRupt(1,1,8,pf);                     //设置中断
Move_DV(1,1,125,500,1000,1,10000)           //执行正向定长运动
```

当此运动完成时，会在产生中断后，执行中断服务子程序，即反向定长运动的动作

使用中断进行连续插补方法：

例 2：设连续依次进行以下三个插补：

```
Circle_ConstSpeed(kh,1,2,1,5000,0,5000,-5000,1000,1500);
Line2D(kh,1,2,250,500,2000,3000,3000);
Line2D(kh,1,2,250,500,2000,6000,6000);
```

则可以进行以下编程：

```
void move2()
{
    Line2D(kh,1,2,250,500,2000,6000,6000);

}
void move1()
{
    void (*pf2)();
    pf2=move2;
    Line2D(kh,1,2,250,500,2000,3000,3000);
    SetInterRupt(kh,1,9,pf2);
}
```

在板卡初始化以后执行以下代码：

```
void (*pf1)();
     pf1=move1;
SetInterRupt(kh,1,9,pf1);
Circle_ConstSpeed(kh,1,2,1,5000,0,5000,-5000,1000,1500);
```

9.10 批处理方式的使用：批处理方式主要在多条不同的运动指令连续执行时使用。在这种方式下只有上一条运动指令控制的所有轴全部运动完毕后才开始下一条运动指令的执行。

```
    SetFlag();
    Move_DV(0,1,125,500,1000,1,1500);
    Move_DV(0,2,125,500,1000,1,1500);
    Move_DV(0,3,125,500,1000,1,1500);
    DisableFlag();
```

运行的结果是 0 号板卡的第一个轴运行 1500 个脉冲长度，然后第二个轴运行 1500 个脉冲长度，最后是第三个轴运行 1500 个脉冲长度。

批处理方式同样可以用于连续插补。

第十章　软件编程和动态连接库调用说明

KPCI884 函数列表

序号	函数名	函数功能
	控制卡设置函数	
1	Init884	初始化 KPCI884 卡
2	SetM	设置实际驱动速度（脉冲频率） (默认值为 8000)
3	PulseOutMode	设置驱动脉冲输出模式 (默认值为 Pulse/DIR 方式)
4	PulseInMode	设置编码器输入脉冲模式（默认为方波脉冲输入方式）
5	RePosition	复位
6	Close884	关闭 KPCI884 卡
	设置和状态查询函数	
7	SetP	设置输出脉冲数（定长运动使用）
8	SetCP	设置软件上限位
9	ClearCP	清除软件上限位
10	SetCM	设置软件下限位
11	ClearCM	清除软件下限位
12	ReadCOMP	读限位
13	ReadLMT	读硬件限位状态
14	LMTMD	硬件限位信号方式 (默认值：立即停机)
15	HLMT	设定硬件正负限位信号的有效电平(默认值：低电平有效)
16	SetLP	设置逻辑位置计数器
17	ReadLP	获取逻辑位置计数器
18	SetEP	设置实际位置计数器
19	ReadEP	获取实际位置计数器
20	MotorStatus	获取电机状态
	速度，加速度函数	
21	SetSV	初始速度设置
22	SetV	驱动速度设置
23	ReadCV	读取当前速度
24	SetA	加速度设置
25	SetDec	减速度设置
26	ReadCA	读取当前加速度
27	AutoDec	自动减速
28	HandDec	手动减速
	制动函数	
29	DecStop	减速至停
30	InstStop	急停
31	Stopped	判断电机是否停止
32	ReadEMG	读紧急停车状态
	运动函数	
33	Set_DV	设置电机定长运动
34	Set_LV	设置电机连续运动

35	Start_DV	定长脉冲启动命令
36	Start_LV	连续脉冲启动命令
37	Move_DV	启动电机定长运动
38	Move_LV	启动电机连续运动
39	DV_OutEnable	定长方式下的外部信号控制函数
40	LV_OutEnable	连续方式下的外部信号控制函数
41	StopOutEnable	禁止外部控制
	I/O 口操作函数	
42	ReadBIT	读取 I/O 端口某一位开关量状态
43	SetOUTBIT	设置 I/O 端口某一位开关量状态
44	ReadBYTE	读取 I/O 端口某一轴开关量状态（字节）
45	SetOUTBYTE	设置 I/O 端口某一轴开关量状态（字节）
46	Alarm_Enable	设置伺服报警信号使能
47	Alarm_Disable	使伺服报警信号不使能
48	SetMCB	设置插补模式下的实际驱动速度（脉冲频率）
	插补函数	
49	Line2D	2 轴线性插补
50	Line3D	3 轴线性插补
51	Circle_Constspeed	2 轴圆弧插补
	中断函数	
52	SetInterRupt	中断设置函数
53	DisableInterrupt	关闭中断（与 SetInterRupt 对应）
	批处理函数	
54	SetFlag	设置成批处理方式
55	DisableFlag	取消批处理方式
	位插补函数	
56	BPInter2D	两轴位插补函数

10.1 KPCI884.dll 函数说明

1） int Init884 (int num)

//初始化 KPCI884 卡

参　数：num 卡号

返回码：0：成功；-1 表示打开板卡失败；-2 表示没有这个卡号对应的板卡

　　-3：表示可能板卡没有安装好或者板卡硬件出现问题

注意：如果在同一台计算机上使用多块板卡，请保证在使用板卡之前一定要执行此函数来初始化相应的板卡。

2） int SetM(int num, unsigned short axis, int wdata)

//设置实际驱动速度（脉冲频率）　　(默认值为 8000)

参　数：num 卡号(1—8)，

　　axis 轴选择（轴选择 1：X 轴，2：Y 轴，3：Z 轴，4：U 轴）

　　wdata 数据设定（范围：1—500）

返回码：0：成功，-1：卡号错误，-2：轴号错误，-3：数据范围超出。

实际驱动速度（单位：PPS）=8000 * wdata （默认值为 8000 ）
例如：如果希望驱动速度为 40 kPPS，
可以设定 wdata=5。因为 V=8000 * 5 =40 K

3）int PulseOutMode(int num,unsigned short axis, int mode)
//设置驱动脉冲输出模式 (默认值为 Pulse/DIR 方式)
参 数：num 卡号(1—8)，
axis 轴选择（轴选择 1：X 轴，2：Y 轴，3：Z 轴，4：U 轴）
mode 模式设置：0：CW/CCW 方式，1：Pulse/DIR 方式
返回码：0：成功，-1：卡号错误,-2 :轴号错误,-3:模式错误。

4) int PulseInMode(int num,unsigned short axis,unsigned short mode,unsigned short xf)
//设置编码器输入脉冲的类型 （默认值为方向脉冲输入方式）
参数：num 卡号(1—8)，
axis 轴选择（轴选择 1：X 轴，2：Y 轴，3：Z 轴，4：U 轴）
mode 模式设置： 0：方波脉冲输入方式，1：正反向脉冲输入方式
xf 细分设置：1：1/1 细分；2：1/2 细分；3：1/4 细分
返回码：0，命令执行成功，-1：卡号错误，-2：轴号错误，-3：参数设置错误

5）int RePosition (int num)
//复位
参 数：num 卡号(1—8)，
返回码：0：成功，-1：卡号错误。
该卡控制的所有电机将初始化；等待重新设置参数

6）int Close884()
//关闭板卡

7）int SetP(int num, unsigned short axis, long dwdata)
//设置输出脉冲数 (定长运动使用)
参 数：num 卡号(1—8)，
axis 轴选择（轴选择 1：X 轴，2：Y 轴，3：Z 轴，4：U 轴）
dwdata 数据设定（范围：0—268 435 455）
返回码：0：成功，-1：卡号错误， -2： 轴号错误，-3：数据范围超出。
输出脉冲数设定：设定在定长脉冲运行时总的脉冲数。该值为绝对无符号数。输出的脉冲数可以在驱动过程中改变。

8）int SetCP(int num,unsigned short axis, long dwdata, UINT fs)
//设置软件上限位
参 数：num 卡号(1—8)，
axis 轴选择（轴选择 1：X 轴，2：Y 轴，3：Z 轴，4：U 轴）
dwdata 数据设定（范围 ：-2 147 483 648 → +2 147 483 648）
fs 0:逻辑位置； 1：实际位置
返回码：0：成功，-1：卡号错误， -2： 轴号错误 。

9）int ClearCP(int num,unsigned short axis)
//清除软件上限位
参　数：num 卡号(1—8)，
　　axis 轴选择（轴选择 1：X 轴，2：Y 轴，3：Z 轴，4：U 轴）
返回码：0：成功，-1：卡号错误，　-2：　轴号错误。

10）int SetCM(int num,unsigned short axis, long dwdata,UINT fs)
//设置软件下限位
参　数：num 卡号(1—8)，
　　axis 轴选择（轴选择 1：X 轴，2：Y 轴，3：Z 轴，4：U 轴）
　　dwdata 数据设定（范围 ：-2 147 483 648 → +2 147 483 648）
　　fs 0:逻辑位置；　　1：实际位置
返回码：0：成功，-1：卡号错误，-2：轴号错误。

11）int ClearCM(int num,unsigned short axis)
//清除软件下限位
参　数：num 卡号(1—8)，
　　axis 轴选择（轴选择 1：X 轴，2：Y 轴，3：Z 轴，4：U 轴）
返回码：0：成功，-1：卡号错误，2：　轴号错误。

12）int ReadCOMP(int num, unsigned short axis)
//读软件限位
参　数：num 卡号(1—8)，
　　axis 轴选择（轴选择 1：X 轴，2：Y 轴，3：Z 轴，4：U 轴
返回码：-1：卡号错误，-2：　轴号错误 1：达到上限位，2：达到下限位。0：未到限位

13）int LMTMD (int num,unsigned short axis,UINT fs)
//硬件限位信号方式　　(默认值：立即停机)
参　数：num 卡号(1—8)，
　　axis 轴选择（轴选择 1：X 轴，2：Y 轴，3：Z 轴，4：U 轴）
　　fx 选择　　1：减速停机，0：立即停机
返回码：0：成功，-1：卡号错误，-2：轴号错误，-3 方式设置错误。

14）int HLMT(int num,unsigned axis,UINT zf,UINT dp)
//设定硬件正负限位信号的有效电平　　(默认值：低电平有效)
参　数：num 卡号(1—8)，
　　axis 轴选择（轴选择 1：X 轴，2：Y 轴，3：Z 轴，4：U 轴）
　　zf 选择　　1：表示设置正限位，2：表示设置负限位
　　dp 选择　　1：表示高电平有效 0：表示低电平有效
返回码：0：成功，-1：卡号错误，-2：轴号错误，-3 其他参数设置错误。

15）int ReadLMT(int num, unsigned short axis)
//读硬件限位状态
参　数：num 卡号(1—8)，
　　axis 轴选择（轴选择 1：X 轴，2：Y 轴，3：Z 轴，4：U 轴）
返回码：　1：硬件上限位有效

2：硬件下限位有效；
0：限位信号无效
-1：卡号错误，
-2： 轴号错误

16）int SetLP(int num,unsigned short axis, long dwdata)
//设置逻辑位置计数器
参 数：num 卡号(1—8)，
axis 轴选择（轴选择 1：X 轴，2：Y 轴，3：Z 轴，4：U 轴）
dwdata 数据设定（范围：-2 147 483 647—+2 147 483 647 ）
返回码：0：成功，-1：卡号错误，-2： 轴号错误，-3：数据范围超出。
步进脉冲从此逻辑位置开始计数

17）long ReadLP(int num,unsigned short axis)
//获取逻辑位置
参 数：num 卡号(1—8)，
axis 轴选择（轴选择 1：X 轴，2：Y 轴，3：Z 轴，4：U 轴）
返回码：逻辑位置数值 -1：卡号错误，-2： 轴号错误。

18）int SetEP(int num,unsigned short axis, long dwdata)
//设置实际位置计数器
参 数：num 卡号(1—8)，
axis 轴选择（轴选择 1：X 轴，2：Y 轴，3：Z 轴，4：U 轴）
dwdata 数据设定（范围：-2 147 483 647—+2 147 483 647 ）
返回码：0：成功，-1：卡号错误，-2： 轴号错误，-3：数据范围超出。
步进脉冲从此实际位置开始计数

19）long ReadEP(int num,unsigned short axis)
//获取实际轴位置
参 数：num 卡号(1—8)
axis 轴选择（轴选择 1：X 轴，2：Y 轴，3：Z 轴，4：U 轴）
返回码：实际位置数值 -1：卡号错误，-2： 轴号错误。

20）int MotorStatus(int num, unsigned short axis)
//电机状态回读
参 数：num 卡号(1—8)，
axis 轴选择（轴选择 1：X 轴，2：Y 轴，3：Z 轴，4：U 轴）
返回码： -1：卡号错误，
-2：轴号错误，
1：电机达到硬件上限位，
2：电机达到硬件下限位，
3：电机由于外部信号处于停机状态。
4：正在加速
5：匀速
6：正在减速

21）int SetSV(int num,unsigned short axis, unsigned short wdata)
//初始速度设置
参 数：num 卡号(1—8)，
axis 轴选择（轴选择 1：X 轴，2：Y 轴，3：Z 轴，4：U 轴）
wdata 数据设定（范围：1—8000）
返回码：0：成功，-1：卡号错误，-2： 轴号错误，-3：数据范围超出。

22) int SetV(int num,unsigned short axis, unsigned short wdata)
//驱动速度设置
参 数：num 卡号(1—8)，
axis 轴选择（轴选择 1：X 轴，2：Y 轴，3：Z 轴，4：U 轴）
wdata 数据设定（范围：1—8000）
返回码：0：成功，-1：卡号错误，-2：轴号错误，-3：数据范围超出。

23) int ReadCV(int num,unsigned short axis)
//获取当前速度
参 数：num 卡号(1—8)，
axis 轴选择（轴选择 1：X 轴，2：Y 轴，3：Z 轴，4：U 轴）
返回码：当前速度数值 -1：卡号错误；-2： 轴号错误。

24) int SetA(int num,unsigned short axis, int wdata)
//加速度设置
参 数：num 卡号(1—8)，
axis 轴选择（轴选择 1：X 轴，2：Y 轴，3：Z 轴，4：U 轴）
wdata 数据参数（范围：-1000 000—— -125 ； +125——+1000 000）
当-125< wdata <125 加速度为 0
返回码：0：成功，-1：卡号错误，-2 ：轴号错误，-3：数据范围超出。

25) int SetDec(int num,unsigned short axis, int wdata)
//减速度设置
参 数：num 卡号(1—8)，
axis 轴选择（轴选择 1：X 轴，2：Y 轴，3：Z 轴，4：U 轴）
wdata 数据参数（范围：-1000 000—— -125 ； +125——+1000 000）
当-125< wdata <125 加速度为 0
返回码：0：成功，-1：卡号错误，-2 ：轴号错误，-3：数据范围超出。

26） int ReadCA(int num,unsigned short axis)
//获取当前加速度
参 数：num 卡号(1—8)，
axis 轴选择（轴选择 1：X 轴，2：Y 轴，3：Z 轴，4：U 轴）
返回码：当前加速度数值. 正数：加速；负数：减速 零：匀速
-1：卡号错误；-2： 轴号错误。

27） int AutoDec(int num, unsigned short axis)
//自动减速
参　数：num 卡号(1—8)，
　　axis 轴选择（轴选择 1：X 轴，2：Y 轴，3：Z 轴，4：U 轴）
返回码：　0：成功；　-1：卡号错误，　-2：　轴号错误

28） int HandDec(int num, unsigned short axis, long　DecPulse)
//手动减速
参　数：num 卡号(1—8)，
　　axis 轴选择（轴选择 1：X 轴，2：Y 轴，3：Z 轴，4：U 轴）
　　DecPulse 为减速数值
注意：此函数只能在定长脉冲模式下使用
返回码：　0：成功；　-1：卡号错误，　-2：　轴号错误

29） int DecStop(int num,unsigned short axis)
//减速至停
参　数：num 卡号(1—8)，
　　axis 轴选择（轴选择 1：X 轴，2：Y 轴，3：Z 轴，4：U 轴）
返回码：0：成功，-1：卡号错误，-2 轴号错误。

30） int InstStop(int num,unsigned short axis)
//急停
参　数：num 卡号(1—8)，
　　axis 轴选择（轴选择 1：X 轴，2：Y 轴，3：Z 轴，4：U 轴）
返回码：0：成功，-1：卡号错误，　-2：　轴号错误。

31） int Stopped(int num, unsigned int axis)
//判断电机是否停止
参　数：num 卡号(1—8)，
　　axis 轴选择（轴选择 1：X 轴，2：Y 轴，3：Z 轴，4：U 轴）
返回码：-1：卡号错误，-2：　轴号错误，0：未停，1：停止。

32） int ReadEMG(int num, unsigned short axis)
//读紧急停车状态
参　数：num 卡号(1—8)，
　　axis 轴选择（轴选择 1：X 轴，2：Y 轴，3：Z 轴，4：U 轴）
返回码：　0：未停车；1：已停车 ；-1：卡号错误，　-2：　轴号错误

33）int Set_DV(int num,unsigned short axis,unsigned short m_v_a,unsigned short m_v_sv,unsigned short m_v_v,long m_v_p)
//设置电机定长运动
参　数：num 卡号(1—8)，
　　axis 轴选择（轴选择 1：X 轴，2：Y 轴，3：Z 轴，4：U 轴）
　　m_v_a 加速度，（范围：-1000 000—— -125 ； +125——+1000 000）
　　当 -125< m_v_a <125 加速度为 0
　　m_v_sv 初始速度，（范围：1—8000）

m_v_v 驱动速度，　（范围：1—8000）
m_v_p 定长驱动脉冲数。（范围：0—268 435 455）
返回码：0：成功，-1：卡号错误，-2 ：轴号错误，-3：数据范围超出。

34）int Set_LV(int num,unsigned short axis,unsigned short m_v_a,unsigned short m_v_sv,unsigned short m_v_v)
//设置电机连续运动
参　数：num 卡号(1—8)，
axis 轴选择（轴选择 1：X 轴，2：Y 轴，3：Z 轴，4：U 轴）
m_v_a 加速度，　（范围：-1000 000—- -125 ； +125—-+1000 000）
当 -125< m_v_a <125 加速度为 0
m_v_sv 初始速度，（范围：1—8000）
m_v_v 驱动速度，　(范围：1—8000)
返回码：0：成功，-1：卡号错误，　-2：　轴号错误，-3：数据范围超出。

35）int Start_DV(int num,unsigned short axis,int fx)
//定长脉冲启动命令
参　数：num 卡号(1—8)，
axis 轴选择（轴选择 1：X 轴，2：Y 轴，3：Z 轴，4：U 轴）
fx 方向选择　　1：正方向，0：负方向。
返回码：0：成功，-1：卡号错误，　-2 ：轴号错误，-4：fx 错误
初始速度,加速度,驱动速度 分别由 SetSV,SetA 和 SetV 函数或 Set_DV 函数设置

36）int Start_LV(int num,unsigned short axis,int fx)
//连续脉冲启动命令
参　数：num 卡号(1—8)，
axis 轴选择（轴选择 1：X 轴，2：Y 轴，3：Z 轴，4：U 轴）
fx 方向选择　1：正方向，0：负方向。
返回码：0：成功，-1：卡号错误，　-2 ：轴号错误
初始速度,加速度,驱动速度 分别由 SetSV,SetA 和 SetV 函数或 Set_LV 函数设置

37）int Move_DV(int num,unsigned short axis,unsigned short m_v_a,unsigned short m_v_sv,unsigned short m_v_v,int fx,long m_v_p)
//启动电机定长运动
参　数：num 卡号(1—8)，
axis 轴选择（轴选择 1：X 轴，2：Y 轴，3：Z 轴，4：U 轴）
m_v_a 加速度（范围：-1000 000—- -125 ； +125—-+1000 000）
当 -125< m_v_a <125 加速度为 0
m_v_sv 初始速度，（范围：1—8000）
m_v_v 驱动速度，　（范围：1—8000）
fx 方向　　(1：正方向，0：负方向)，
m_v_p 定长驱动脉冲数。（范围：0—268 435 455）
返回码：0：成功，-1：卡号错误，-2 ：轴号错误，-3：数据范围超出，-4：　fx 错误

38） int Move_LV(int num,unsigned short axis,unsigned short m_v_a,unsigned short m_v_sv,unsigned short m_v_v,int fx)

//启动电机连续运动

参　数：num 卡号(1—8)，

axis 轴选择（轴选择 1：X 轴，2：Y 轴，3：Z 轴，4：U 轴）

m_v_a 加速度，(范围：-1000 000—- -125 ； +125—-+1000 000)

当 -125< m_v_a <125 加速度为 0

m_v_sv 初始速度，(范围：1—8000)

m_v_v 驱动速度，　(范围：1—8000)

fx 方向设置：　　(1：正方向，0：负方向)。

返回码：0：成功，-1：卡号错误，　-2：　轴号错误，-3：数据范围超出，-4：fx 错误

39） int　DV_OutEnable (int num, unsigned short axis)

//定长方式下的外部信号控制函数，执行此函数后可以通过使端子 XEXOP+，XEXOP-（Y 轴，Z 轴，U 轴类同）的信号有效，由外部信号直接控制电机的定长运动，直至所设驱动脉冲数完成

参　数：num 卡号(1—8)，

axis 轴选择（轴选择 1：X 轴，2：Y 轴，3：Z 轴，4：U 轴）

返回码：0：成功，-1：卡号错误，-2：轴号错误。

40） int　LV_OutEnable (int num, unsigned short axis)

//连续方式下外部信号控制函数，执行此函数后可以通过使端子 XEXOP+，XEXOP-（Y 轴，Z 轴，U 轴类同）的信号有效，由外部信号直接控制电机的连续运动

参　数：num 卡号(1—8)，

axis 轴选择（轴选择 1：X 轴，2：Y 轴，3：Z 轴，4：U 轴）

返回码：0：成功，-1：卡号错误，　-2：　轴号错误。

41）int　StopOutEnable (int num, unsigned short axis)

//禁止外部控制,执行此函数后将使端子 XEXOP+，XEXOP-（Y 轴，Z 轴，U 轴类同）的信号无效。

参　数：num 卡号(1—8)，

axis 轴选择（轴选择 1：X 轴，2：Y 轴，3：Z 轴，4：U 轴）

返回码：0：成功，-1：卡号错误，　-2 ：轴号错误。

42） int ReadBIT(int num, unsigned short axis,short bits)

//读取 I/O 端口某一位开关量状态

参　数：num 卡号(1—8)，

axis 轴选择（轴选择 1：X 轴，2：Y 轴，3：Z 轴，4：U 轴）

bits 返回字节的第 N 位（N 值的范围：1~3）

返回码：0 或 1　表示某一轴的 I/O 端口状态　　-1：卡号错误，　-2：　轴号错误

（注意：硬件 I/O 只有 3 个输入点；而输出有 4 个）

43） int SetOUTBIT(int num, unsigned short axis,unsigned short bits,unsigned short data)

//设置 I/O 输出口某一位的开关量状态

参　数：num 卡号(1—8)，

axis 轴选择（轴选择 1：X 轴，2：Y 轴，3：Z 轴，4：U 轴）

bits 输出字节的第 N 位（N 值的范围：1~4），

data 的取值为 1 或 0

返回码：0：成功 -1：卡号错误， -2： 轴号错误

44） int ReadBYTE (int num, unsigned short axis)

//读取 I/O 端口某轴的开关量状态

参　数：num 卡号(1—8)，

axis 轴选择（轴选择 1：X 轴，2：Y 轴，3：Z 轴，4：U 轴）

data 的取值为字节的低 3 位。

返回码：返回值为某一轴的 I/O 端口状态 -1：卡号错误， -2： 轴号错误

注意：硬件 I/O 只有 3 个输入点；（低 3 位数据分别对应_in1,_in2,_in3）而输出有 4 个。

45） int SetOUTBYTE(int num, unsigned short axis,unsigned short data)

//设置 I/O 输出口某一轴的开关量状态

参　数：num 卡号(1—8)，

axis 轴选择（轴选择 1：X 轴，2：Y 轴，3：Z 轴，4：U 轴）

data 的取值为字节的低 4 位。

返回码： 0：成功 -1：卡号错误， -2： 轴号错误

46）Alarm_Enable(int num,unsigned short axis,UINT dp)

参　数：num 卡号(1—8)，

axis 轴选择（轴选择 1：X 轴，2：Y 轴，3：Z 轴，4：U 轴）

dp 有效电平选择（0：高电平，1 低电平）

返回码： 0：成功 -1：卡号错误， -2： 轴号错误， 3： 电平选择错误

47）Alarm_Disable(int num,unsigned short axis)

参　数：num 卡号(1—8)，

axis 轴选择（轴选择 1：X 轴，2：Y 轴，3：Z 轴，4：U 轴）

返回码： 0：成功 -1：卡号错误， -2： 轴号错误

48） int SetMCB (int num, float fdata)

//设置插补模式下的实际驱动速度（脉冲频率）

参　数：num 卡号(1—8)

fdata 数据设定（范围：1—500）

返回码：0：成功，-1：卡号错误，-2：轴号错误，-3：数据范围超出。

49）int Line2D (int num, UNIT axis1, UNIT axis2,unsigned short m_v_sv, unsigned short m_v_v,unsigned long m_v_p1,unsigned long m_v_p2)

//2 轴线性插补

参　数：num 卡号(1—8)，

axis1,axis2 轴选择（轴选择 1：X 轴，2：Y 轴，3：Z 轴，4：U 轴）

axis1 为主轴 且 axis1≠axis2
m_v_sv 初始速度，（范围：1—8000）
m_v_v 驱动速度， （范围：1—8000）
m_v_p1 主轴的终点坐标值。（范围：-8 386 607—+8 386 607）
m_v_p2 从轴的终点坐标值。（范围：-8 386 607—+8 386 607）
返回码：0：成功，-1：卡号错误，-2 ：轴号错误，-3：数据范围超出。

50） int Line3D (int num, UNIT axis1, UNIT axis2, UNIT axis3,unsigned short m_v_sv,unsigned short m_v_v,unsigned long m_v_p1, long m_v_p2, long m_v_p3)
//3 轴线性插补
参 数：num 卡号(1—8)，
axis1,axis2，axis3 轴选择（轴选择 1：X 轴，2：Y 轴，3：Z 轴，4：U 轴）
axis1 为主轴 且 axis1≠axis2≠axis3
m_v_sv 初始速度，（范围：1—8000）
m_v_v 驱动速度， （范围：1—8000）
m_v_p1 主轴的终点坐标值。（范围：-8 386 607—+8 386 607）
m_v_p2 从轴的终点坐标值。（范围：-8 386 607—+8 386 607）
m_v_p3 从轴的终点坐标值。（范围：-8 386 607—+8 386 607）
返回码：0：成功，-1：卡号错误，-2 ：轴号错误，-3：数据范围超出。

51） int Circle_ConstSpeed(int num,UINT axis1,UINT axis2,UINT fm,int m_c1,int m_c2,unsigned long m_p1,unsigned long m_p2,unsigned short m_v_sv,unsigned short m_v_v)
// 恒定线速度下的 2 轴圆弧插补
参 数：num 卡号(1—8)，
axis1,axis2 轴选择（轴选择 1：X 轴，2：Y 轴，3：Z 轴，4： U 轴）
axis1 为主轴 且 axis1≠axis2

m_v_sv 初始速度，（范围：1—8000）
m_v_v 驱动速度， （范围：1—8000）
m_c1 圆心的主轴坐标值。（范围：-8 388 608—+8 388 607）
m_c2 圆心的从轴坐标值。（范围：-8 388 608—+8 388 607）
m_p1 主轴的终点坐标值。（范围：-8 386 607—+8 386 607）
m_p2 从轴的终点坐标值。（范围：-8 386 607—+8 386 607）
返回码：0：成功，-1：卡号错误，-2 ：轴号错误，-3：数据范围超出。

52） int SetInterRupt(int CardNum,unsigned short Axis,int IntClass,void (*pf)())
//中断函数
参 数：CardNum 卡号(1—8)，
axis 轴选择（轴选择 1：X 轴，2：Y 轴，3：Z 轴，4：U 轴）
IntClass 中断的种类 1:中断信号由各输出的脉冲上升沿触发
2：P>=C-;(P:逻辑/实际位置计数器。C：限位寄存器)
3:P<C-;(P:逻辑/实际位置计数器。C：限位寄存器)
4:P<C+;(P:逻辑/实际位置计数器。C：限位寄存器)
5:P>=C+;(P:逻辑/实际位置计数器。C：限位寄存器)
6：加减速驱动中，开始减速时发中断信号；

7：在加减速驱动中，当加速完成时发中断信号。
8：驱动结束时发
9：连续插补中插补结束时

Pf() 中断服务子函数，中断来时，自动执行该函数，具体使用见编程示例

53）void DisableInterrupt()
//取消中断工作方式

54） void SetFlag()
//设置成批处理工作方式，批处理工作方式介绍见 8.10 说明）

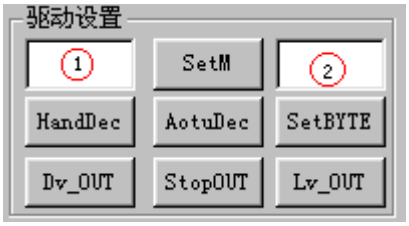
55）void DisableFlag()
取消批处理工作方式，详细介绍见 8.10 说明）

56）**int BPInter2D(int num,double(*pfun)(double),long *px0,long *py0,long N, UINT pm,unsigned short m_v_a,unsigned short m_v_sv,unsigned short m_v_v)**
//两轴位模式插补
参数：num 板卡号；
pfun 被插补函数指针；
px0 插补自变量指针；
py0 插补因变量指针；
N 插补自变量变化方向与范围；
Pm 插补平面 1：XOY 平面，2：ZOX 平面，3：YOZ 平面
m_v_a 主轴的加速度；
m_v_sv 主轴的启动速度；
m_v_v 主轴的驱动速度；
本函数的详细使用请参考上面介绍的位模式编程说明

10.2 测试程序使用说明：

KPCI-884 运动控制卡的 VB 和 VC 测试软件的操作界面采用统一的格式，其控制界面如图：

当没有其它控制类板卡安插在机箱板卡上时，该板卡的默认卡号为 0，单击 初 始 化 按钮即可对板卡进行初始化，如果板卡初始化失败，则弹出“板卡初始化错误，请检查板卡”；如果成功，则可以继续进行测试。在输出模式框内可以选择需要测试的轴 X、Y、Z、U，模式默认为 1（即 1：Pulse\DIR 方式,如果选择 0 模式的话即为 0：CW/CCW 方式），单击 确 定 进行轴模式选择确定。



驱动设置里面可以调用各种驱动函数。①里面可以输入脉冲频率的倍数，然后点击“SetM”输出；②里面可以输入要输出到某轴 I/O 端口的开关量状态，调用的函数是 int SetOUTBYTE(int num, unsigned short axis,unsigned short data)，点击“SetBYTE”按钮可以输出该开关量。

HandDec 用来设置手动减速点，内部调用了函数 HandDec (Num, Axis, 5000)；

AotuDec 用来设置自动减速，调用了函数 int AutoDec(int num, unsigned short axis)；

Dv_OUT 用来设置电机定长运动，调用了函数 int Set_DV(int num,unsigned short axis,unsigned short m_v_a,unsigned short m_v_sv,unsigned short m_v_v,long m_v_p)，而相应设定值为 Set_DV(num, axis , 400, 400, 400, 2000)；

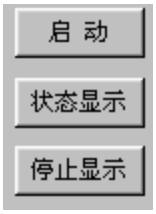
StopOUT 用来禁止外部控制,执行此函数后将使端子 nEXOP+，nEXOP-的信号无效。调用了函数 int StopOutEnable (int num, unsigned short axis)

Lv_OUT 用来设置电机连续运动，调用了函数 int Set_LV(int num,unsigned short axis,unsigned short m_v_a,unsigned short m_v_sv,unsigned short m_v_v) ，而相应设定值为 Set_LV(Num, Axis, 1250, 500, 7000)；

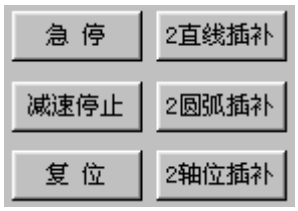
限位设定里面的 

用来设定和清除上下限位设定的。“OK”里面调用了 SetCP 和 SetCM 两个函数，而“Clear”里面调用了 ClearCP 和 ClearCM 两个函数。其上下限位设定值可以通过其上的文本输入框输入，且保证不要超过所允许的范围（-2147483648-+2147483648）；

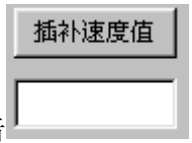
限位设定方式里面从上到下依次是“停止方式设定”、“限位方式设定”和“电平有效方式设定”。内部调用了 LMTMD（硬件限位信号方式—默认值：立即停机）和 HLMT（设定硬件正负限位信号的有效电平—默认值：低电平有效）



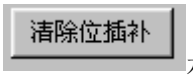
“**启动**”按钮是用来启动选定的轴工作的，内部根据以上的选定模式来调用 Move_DV（启动电机定长运动）或者调用 Move_LV（启动电机连续运动）；点击“**状态显示**”按钮，则“读取参数值”、“外部输入/输出信号状态”和“电机驱动状态”三个框体内的各个参数就可以作相应显示。而点击“**停止显示**”就会停止显示各个状态。



如右图的“急停”、“减速停止”、“复位”、“2 直线插补”和“2 圆弧插补”和“2 轴位插补”按其字面意思进行函数调用。详细内容可查阅超连接。



点击中上面的“插补速度值”则相应下面的文本框内则会显示当前轴的运动速度。内部调用了读取当前速度的函数 int GetCurrentV(int num)



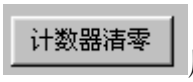
是用来停止某选定轴的位插补的，调用函数是 void BPClear(int Num)

测试 

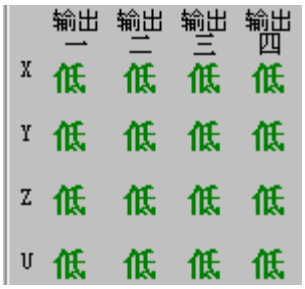
操作，需要设置加速度 SetA、初始速度 SetSV、驱动速度 SetV、加速度变化率 SetK、驱动脉冲定长 SetP、定长脉冲启动命令 Start_DV 等。相应的 

是用来停止 S 形曲线加/减速的，调用的函数是 int SClear(int num,unsigned short axis)。

中 SetEP 和 ReadEP 可以用来设置和读取实际位置计数器，属于眼 K881 板卡专用。而 ReadBYTE 用来读取某轴 I/O 端口的开关量状态，取得的值显示在其下的文本框内。

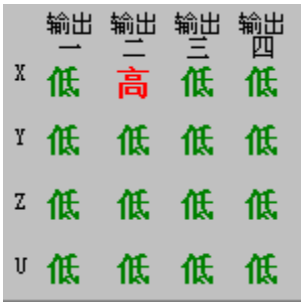


用来清除逻辑位置计数器中的内容，调用的函数是 int SetLP(Num, Axis, 0)。



外部输入/输出信号状态框体内的各轴在测试输出时，如：若要测试 X 轴输出端二，可以在对应 X 输出二的下面点击其信号字“低”或“高”，就可以反复改变 J1 端子引脚 22 的输出情况。

输出内容可以用示波器测试。



第十一章　我们的承诺

本产品自售出之日起两年内，凡用户遵守存储，运输及使用要求，而产品质量低于技术指标的，凭保修单免费维修，因违反操作规程的要求而造成损坏的，需交纳器件和维修费。

第十二章　产品成套单

12.1　KPCI-884 步进电机控制卡壹块。
12.2　北京科瑞兴业产品光盘壹张
12.3　62 芯 D 型插头壹套
12.4　37 芯 D 型插头壹套
12.5　KPCI-884D 接线端子板（选配件）