ADT850

四轴运动控制卡

用户手册

版权声明

商标声明

目　　录

第一章 概要

产品简介

ADT850 卡是基于 PCI 总线的高性能四轴伺服/步进控制卡，支持一个系统中使用多达 16 块控制卡，可控制 64 路伺服/步进电机，支持即插即用，位置可变环形，可在运动中随时改变速度，可使用连续插补等先进功能。

脉冲输出方式可用单脉冲（脉冲+方向）或双脉冲（脉冲+脉冲）方式，最大脉冲频率 4MHz，采用先进技术使输出频率在很高的时候也能使频率误差小于 0.1%。

位置管理采用两个加/减计数器，一个用于内部管理驱动脉冲输出的逻辑位置计数器，一个用于接收外部的输入，输入信号可以是 A/B 相输入的编码器或光栅尺，也可是上/下脉冲的输入信号，作为实际位置计数器，计数器位数高达 32 位，最大范围–2,147,483,648~+2,147,483,647。外部输入也可用于手轮输入，作为普通的计数。

提供伺服接口信号，如编码器信号，到位信号（INPOS），报警信号（ALARM），伺服开启（SERVO ON）等。

多种控制方式，如定量运动，连续运动，回零运动，多轴插补，圆弧插补等。插补一般用定速运动，也可用直线/S 曲线加减速，（S 曲线加减速不能用于圆弧插补）。

插补带有连续插补功能，即在插补过程中输入下一点的插补数据，以保证脉冲的连续，使插补达到更快更好的性能。最大插补速度可达 2Mhz。

速度控制可用定速和直线/S 曲线加减速，可做非对称直线加减速，可用自动/手动减速，在定量驱动时可防止速度曲线产生三角波形。

每轴有 2 个 32 位比较寄存器，用于产生中断或作为软件限位。

每轴有 8 个输入信号，包括 2 个正负限位信号，3 个停止信号，1 个伺服到位信号，1 个伺服报警信号和 1 个通用输入信号，除限位信号外，其余信号可通过设置成无效来作为通用输入信号，3 个停止信号可作为原点信号、减速信号、编码器 Z 相搜寻使用，所有数字输入信号均有积分型的滤波器，可选 8 种滤波时间常数，以防止干扰。

中断可设置成在各种情况下产生，如加/减速驱动的定速开始时，定速完毕时，驱动完毕时，位置计数器和比较器之间的大小关系有变化时等等，此外连续插补发生下一个数据请求时的中断。

提供 DOS、WINDOWS95/98/NT/2000/XP 开发库，可用 VC++、VB、BC++等进行软件开发，

主要性能

- 32 位 PCI 总线，即插即用
- 4 轴伺服/步进电机控制，每轴可独立控制，互不影响
- 脉冲输出的频率误差小于 0.1%
- **最大脉冲输出频率为 4MHz**
- 脉冲输出可用单脉冲（脉冲+方向）或双脉冲（脉冲+脉冲）方式
- **4** 轴 **均有位置反馈输入，32 位计数，最大计数范围 -2,147,483,648~+2,147,483,647**
- 直线或 S 曲线进行加/减速
- **非对称直线加/减速运动**
- 2-4 轴直线插补
- CW、CCW 圆弧插补
- **可用连续插补功能，最大驱动速度 2MHz**
- 每轴都有 2 个 32 位比较寄存器用于逻辑位置计数器或者实际位置计数器的位置大小比较，可用于软件限位，或产生中断
- 可接收伺服马达驱动器的各种信号，如 2 相编码器信号、到位信号、报警信号等
- 每轴有 3 个 STOP 信号，可用于原点搜寻、编码器 Z 相搜寻
- **运动中可实时改变速度**
- 运动中可以实时读出逻辑位置、实际位置、驱动速度、加速度、加/减速状态（加速中、定速中、减速中）
- 可编程中断，可以由多种原因产生中断
- 每轴有 8 入 8 出数字 I/O，除 2 限位信号外均可作为通用 I/O 使用，数字输出可用于伺服开启，伺服报警复位等信号
- 每一个输入信号的输入端都装备积分型的滤波器，可以设定哪一个输入信号的滤波器功能变为有效或无效，滤波器的时间常数从 8 个种类里可以选择 1 个
- 支持在一个系统中使用多达 16 个控制卡
- 支持 DOS、WINDOWS95/98/NT/2000/XP 等操作系统

应用范围

- **多轴雕铣系统**
- **机器人系统**
- **间座标测量系统**
- **基于 PC 的数控系统**

第二章 硬件安装

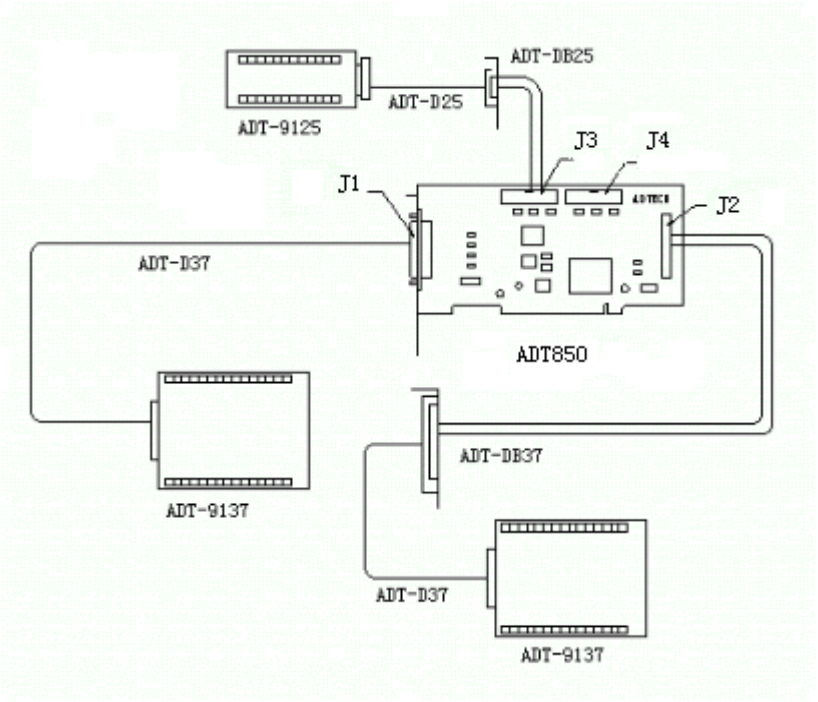
☞ 检查配件

1. ADT-850 用户手册（本手册）
2. ADT-850 四轴 PCI 总线高性能运动控制卡
3. ADT-850 用户光盘
4. ADT-9137 37 芯信号接线板 2 块（选件）
5. ADT-D37 37 芯屏蔽连接线 2 条（选件）
6. ADT-9125 25 芯开关信号接线板 2 块（选件）
7. ADT-D25 25 芯屏蔽连接线 2 条（选件）
8. ADT-DB25 25 芯扁平线 2 条
9. ADT-DB37 37 芯扁平线 1 条

☞ 安装

1. 关闭电脑电源（注：ATX 电源需总电源关闭）。
2. 打开电脑机箱后盖。
3. 选择一条未占用的 PCI 插槽,插入 ADT-850。
4. 检查 ADT-850 的金手指是否完整插入 PCI 插槽,拧整螺丝。
5. 根据用户情况决定是否安装 P1、P3、P4 接口线。

第三章 电气连接

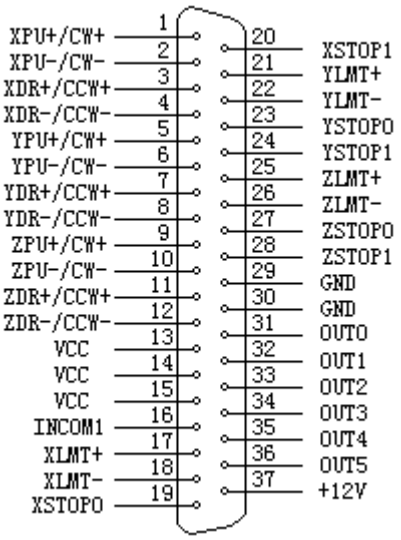


一块 Adt850 卡有四个输入/输出接口，其中 J1、J2 为 37 针插座，J3、J4 为 25 针插座。

J1 为 X、Y、Z 轴的脉冲输出、开关量输入和开关量输出 OUT0-OUT5 的信号接线；**J2** 为 X、Y、Z、W 轴的编码器输入和开关量输入的信号接线；**J3** 为 W 轴的脉冲输出、开关量输入和开关量输出 OUT6-OUT15 信号的接线；**J4** 为开关量输出 OUT16-OUT31 信号的接线

信号定义如下：

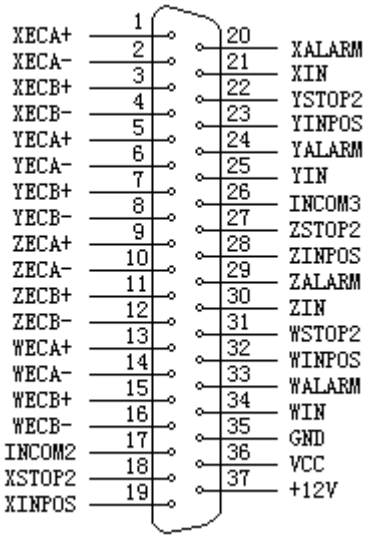
J1 线号说明



线 号	符 号	说 明
1	XPU+/CW+	X 脉冲信号+
2	XPU-/CW-	X 脉冲信号-
3	XDR+/CCW+	X 方向信号+
4	XDR-/CCW-	X 方向信号-
5	YPU+/CW+	Y 脉冲信号+
6	YPU-/CW-	Y 脉冲信号-
7	YDR+/CCW+	Y 方向信号+
8	YDR-/CCW-	Y 方向信号-
9	ZPU+/CW+	Z 脉冲信号+
10	ZPU-/CW-	Z 脉冲信号-
11	ZDR+/CCW+	Z 方向信号+
12	ZDR-/CCW-	Z 方向信号-
13	VCC	内部+5V 电源正端　不可接外接电源

14	VCC	内部+5V 电源正端　不可接外接电源
15	VCC	内部+5V 电源正端　不可接外接电源
16	INCOM1	光耦输入公共端（下面的信号）
17	XLMT+	X 正向限位信号
18	XLMT-	X 反向限位信号
19	XSTOP0	X 停止信号 0，可做通用输入信号
20	XSTOP1	X 停止信号 1，可做通用输入信号
21	YLMT+	Y 正向限位信号
22	YLMT-	Y 反向限位信号
23	YSTOP0	Y 停止信号 0，可做通用输入信号
24	YSTOP1	Y 停止信号 1，可做通用输入信号
25	ZLMT+	Z 正向限位信号
26	ZLMT-	Z 反向限位信号
27	ZSTOP0	Z 停止信号 0，可做通用输入信号
28	ZSTOP1	Z 停止信号 1，可做通用输入信号
29	GND	内部电源地线
30	GND	内部电源地线
31	OUT0	开关量输出点
32	OUT1	
33	OUT2	
34	OUT3	
35	OUT4	
36	OUT5	
37	+12V	内部+12V 电源正端　不可接外接电源

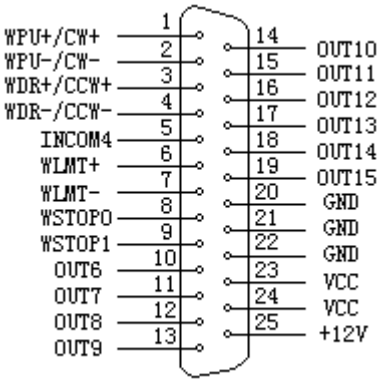
J2 线号说明



线号	符号	说明
1	XECA+	X 轴编码器 A 相输入+
2	XECA-	X 轴编码器 A 相输入-
3	XECB+	X 轴编码器 B 相输入+
4	XECB-	X 轴编码器 B 相输入-
5	YECA+	Y 轴编码器 A 相输入+
6	YECA-	Y 轴编码器 A 相输入-
7	YECB+	Y 轴编码器 B 相输入+
8	YECB-	Y 轴编码器 B 相输入-
9	ZECA+	Z 轴编码器 A 相输入+
10	ZECA-	Z 轴编码器 A 相输入-
11	ZECB+	Z 轴编码器 B 相输入+
12	ZECB-	Z 轴编码器 B 相输入-

13	WECA+	W 轴编码器 A 相输入+
14	WECA-	W 轴编码器 A 相输入-
15	WECB+	W 轴编码器 B 相输入+
16	WECB-	W 轴编码器 B 相输入-
17	INCOM2	光耦输入公共端（下面的信号）
18	XSTOP2	X 停止信号 2，可做通用输入信号
19	XINOPOS	X 伺服到位信号，可做通用输入信号
20	XALARM	X 伺服报警信号，可做通用输入信号
21	XIN	X 通用输入信号
22	YSTOP2	Y 停止信号 2，可做通用输入信号
23	YINPOS	Y 伺服到位信号，可做通用输入信号
24	YALARM	Y 伺服报警信号，可做通用输入信号
25	YIN	Y 通用输入信号
26	INCOM3	光耦输入公共端（下面的信号）
27	ZSTOP2	Z 停止信号 2，可做通用输入信号
28	ZINPOS	Z 伺服到位信号，可做通用输入信号
29	ZALARM	Z 伺服报警信号，可做通用输入信号
30	ZIN	Z 通用输入信号
31	WSTOP2	W 停止信号 2，可做通用输入信号
32	WINPOS	W 伺服到位信号，可做通用输入信号
33	WALARM	W 伺服报警信号，可做通用输入信号
34	WIN	W 通用输入信号
35	GND	内部电源地线
36	VCC	内部+5V 电源正端　不可接外接电源
37	+12V	内部+12V 电源正端　不可接外接电源

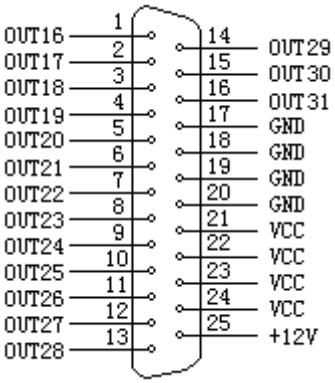
J3 线号说明



线 号	符 号	说 明
1	WPU+/CW+	W 脉冲信号+
2	WPU-/CW-	W 脉冲信号-
3	WDR+/CCW+	W 方向信号+
4	WDR-/CCW-	W 方向信号-
5	INCOM4	光耦输入公共端（下面的信号）
6	WLMT+	W 正向限位信号
7	WLMT-	W 反向限位信号
8	WSTOP0	W 停止信号 0，可做通用输入信号
9	WSTOP1	W 停止信号 1，可做通用输入信号
10	OUT6	开关量输出点
11	OUT7	
12	OUT8	
13	OUT9	
14	OUT10	
15	OUT11	
16	OUT12	

17	OUT13	
18	OUT14	
19	OUT15	
20	GND	内部电源地线
21	GND	内部电源地线
22	GND	内部电源地线
23	VCC	内部+5V 电源正端
24	VCC	内部+5V 电源正端
25	+12V	内部+12V 电源正端

J4 线号说明



线 号	符 号	说 明
1	OUT16	开关量输出点
2	OUT17	
3	OUT18	
4	OUT19	
5	OUT20	
6	OUT21	
7	OUT22	

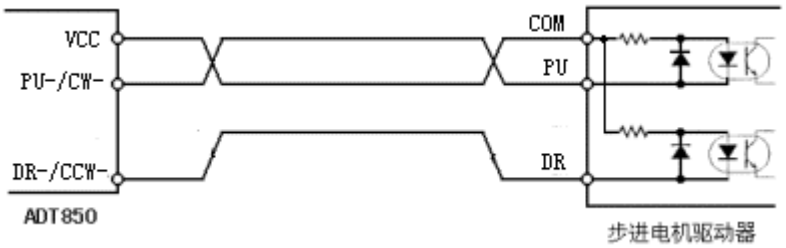
8	OUT23	
9	OUT24	
10	OUT25	
11	OUT26	
12	OUT27	
13	OUT28	
14	OUT29	
15	OUT30	
16	OUT31	
17	GND	内部电源地线
18	GND	
19	GND	
20	GND	
21	VCC	内部+5V 电源正端
22	VCC	
23	VCC	
24	VCC	
25	+12V	内部+12V 电源正端

☞ 脉冲/方向输出信号的连接

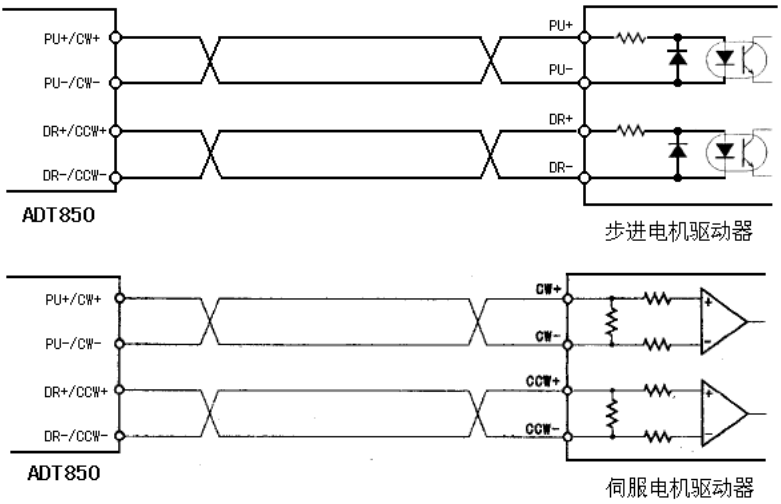
脉冲输出为差动输出方式

可与步进/伺服驱动器很方便的连接

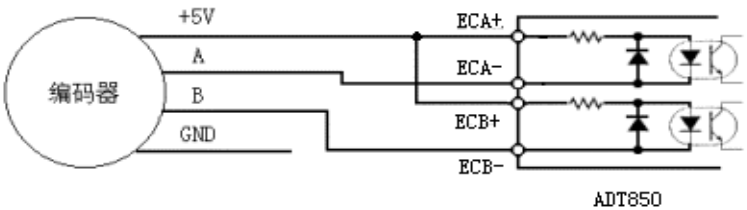
下图为脉冲与方向的阳极已连通的接法



下图为脉冲与方向信号独立的接法，建议采用此种方法，因为是差动接法，抗干扰性强。

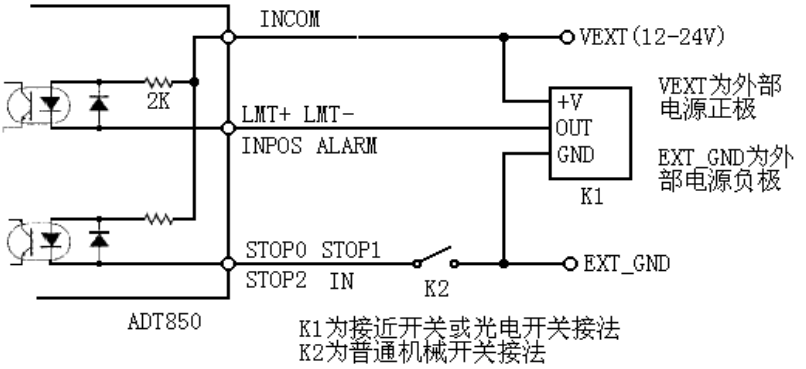


☞ 编码器输入信号的连接

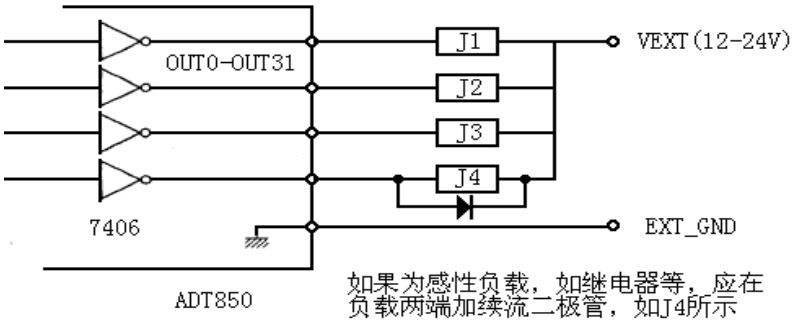


以上以使用+5V 电源的编码器为例，+5V 电源可使用内部电源，也可外接 5V 电源。

☞ 数字输入的连接


INCOM
VEXT(12-24V)
2K
LMT+ LMT-
INPOS ALARM
+V
OUT
GND
K1
VEXT为外部
电源正极
EXT_GND为外
部电源负极
STOP0 STOP1
STOP2 IN
K2
EXT_GND
ADT850
K1为接近开关或光电开关接法
K2为普通机械开关接法

☞ **数字输出的连接**


OUT0-OUT31
J1
J2
J3
J4
VEXT(12-24V)
7406
EXT_GND
ADT850
如果为感性负载，如继电器等，应在
负载两端加续流二极管，如J4所示

第四章 软件安装

ADT850 卡在 WINDOWS95/98/NT/2000/XP 下必须安装驱动程序才能使用，在 DOS 下则无须安装驱动程序。

ADT850 卡 与 ADT850 卡只是在接线上有些不同，软件方面完全一样。

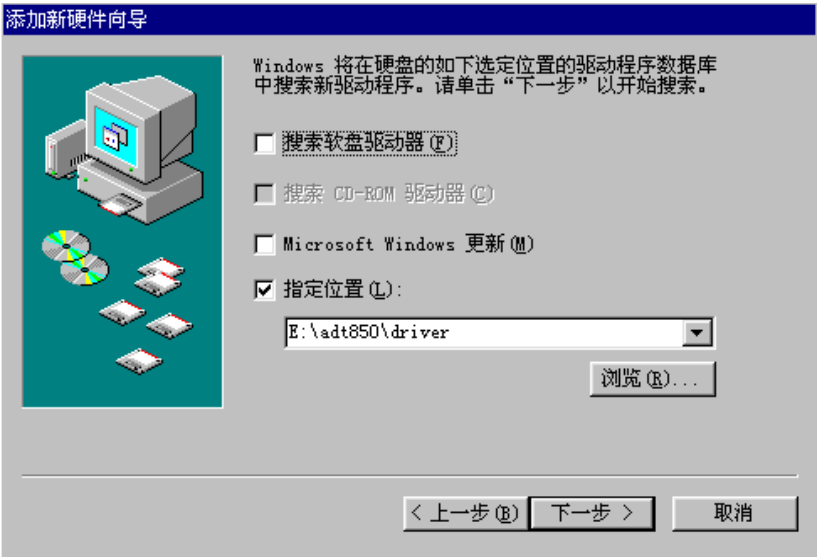
☞ WIN95/98 下驱动程序的安装

以下用 Windows98 中文版为例，说明驱动程序的安装。

在将 ADT850 卡安装到电脑上的 PCI 插槽后，电脑开机后应发现新硬件，出现如下图的提示

单击“下一步”后出现如下画面

按上图所示选择，再单击“下一步”，出现如下画面



以上假设 E 盘为光盘，按上图输入路径，或者使用“浏览”寻找 ADT850.INF 文件所在的路径，单击“下一步”后，应出现如下画面

单击“下一步”后出现如下画面：

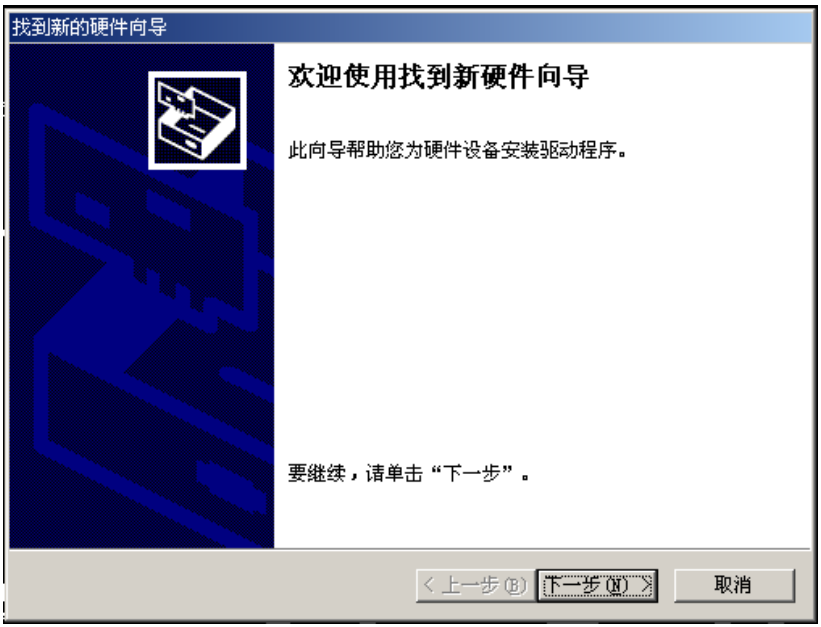
单击“完成”后，即完成 ADT850 卡的安装。

☞ **Winnt 下驱动程序的安装**
☞ **WinXP 下驱动程序的安装**
☞ **Win2000 下驱动程序的安装**

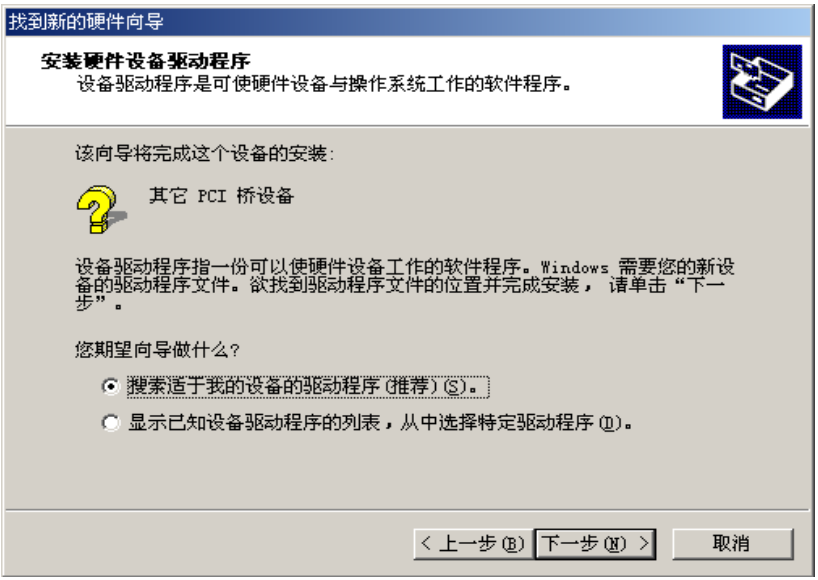
以下用 Windows2000 Professional 中文版为例，说明驱动程序的安装，其余版本的 Windows2000 与此类似。

在将 ADT850 卡安装到电脑上的 PCI 插槽后，开机时应以管理员身份登陆，电脑开机后应发现新硬件，出现如下图的提示

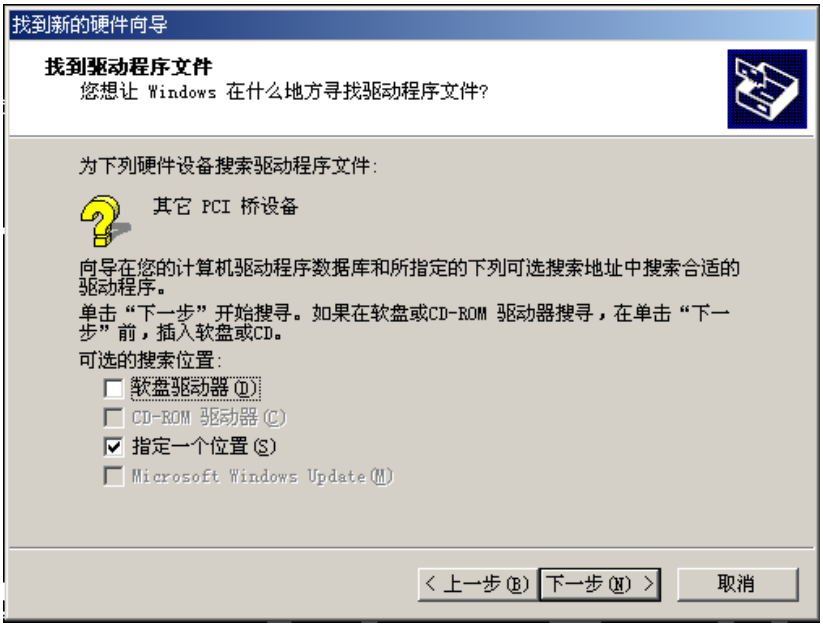
然后出现如下画面



单击“下一步”后，再显示如下画面

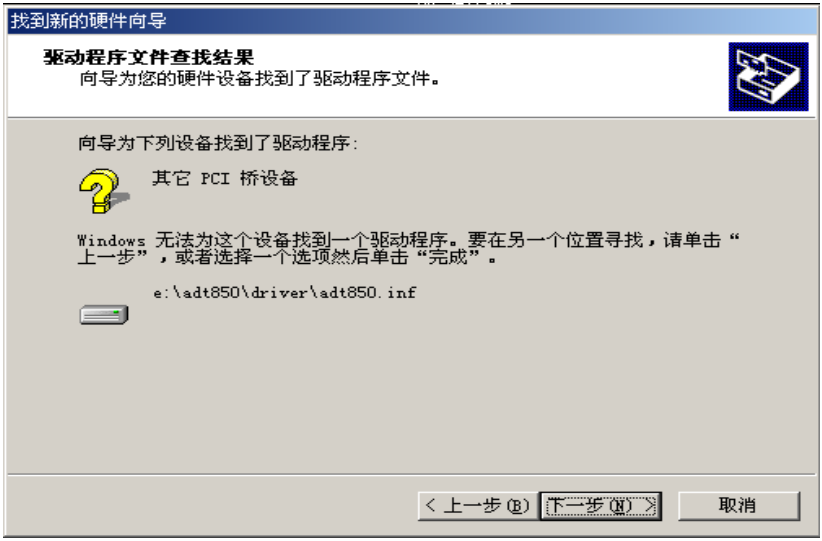


按上图选择后，再单击“下一步”后，出现如下画面

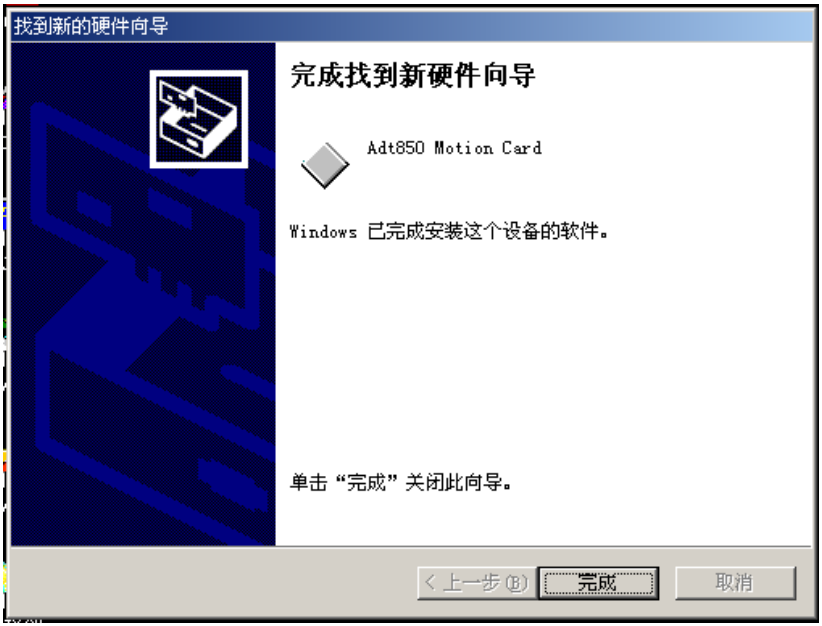


再按上图选择“指定一个位置”，单击“下一步”，出现如下画面

以上假设 E 盘为光盘，或者使用“浏览”寻找 ADT850.INF 文件所在的路径，单击“确定”后，应出现如下画面



单击”下一步”后出现如下画面



单击“完成”后，即完成 ADT850 卡的安装

第五章 功能说明

一、定量驱动

定量驱动的意思是以固定速度或加/减速度输出指定数量的脉冲。需要移动到确定的位置或进行确定的动作时，使用此功能。加/减速的定量驱动如下图所示，输出脉冲的剩余数比加速累计的脉冲数少时就开始减速，输出指定的脉冲数后驱动也结束。

进行加/减速的定量驱动需要设定下列参数：

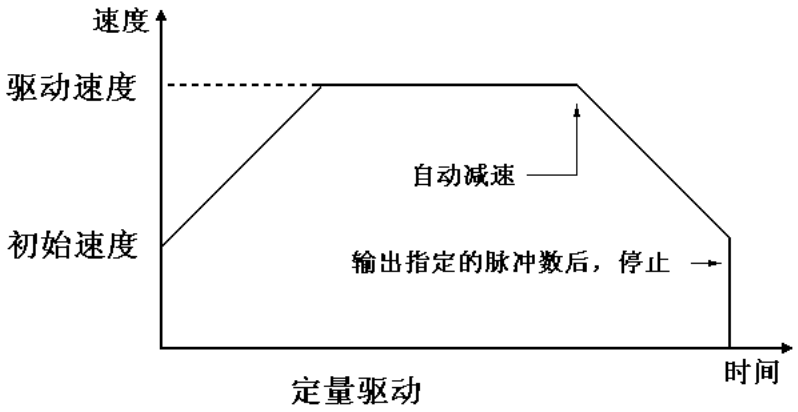
a) 范围 R

b) 加/减速 A/D

c) 初始速度 SV

d) 驱动速度 V

e) 输出脉冲数 P



加/减速定量驱动是一般如上图所示从计算的减速点开始自动减速，此外也可以用手动减速。在下列的情况下，不能正确地计算自动减速点或无法算出此减速点，所以需要手动地计算减速点：

- 直线加/减速定量驱动中需要经常变更速度。
- 用加/减速运行圆弧插补、连续插补。
  需要改为手动减速模式，设定减速点。

二、连续驱动

在连续驱动中，连续输出驱动脉冲直至高位的停止命令或外部的停止信号有效。需要运行原点搜寻、扫描操作、控制马达旋转速度时，使用此功能。

有两种停止命令，一个是减速停止，另一个是立即停止。每个轴都有用于减速/立即停止的STOPO、STOP1、STOP2的3点外部信号。每个信号都可以设定有效/无效电平。STOPO、STOP1、STOP2信号在加/减速驱动中为减速停止，在定速驱动中，为立即停止。

连续驱动的原点搜寻动作

把原点接近信号、原点信号、编码器Z相信号等安排在STOPO、STOP1、STOP2。在各轴上设定各信号的有效/无效和逻辑电平，高速搜寻时，用加/减速连续驱动，当设定的有效信号处于激活电平时就减速停止。低速搜寻时，用定速连续驱动。当设定的有效信号处于激活电平时，就立即停止。为了以加/减速连续驱动，除了输出脉冲数以外都要设定和定量驱动一样的参数。

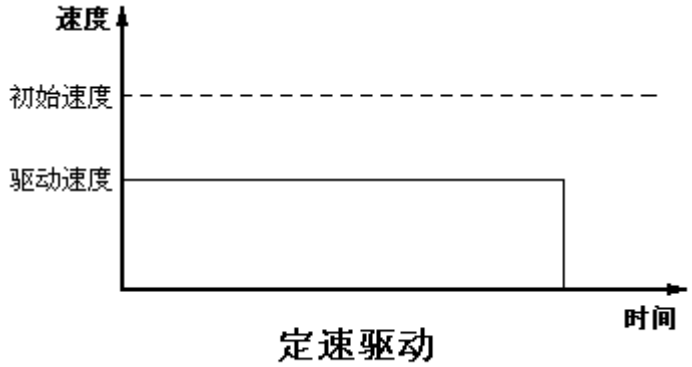
三、速度曲线

3.1 定速驱动

定速驱动就是以一成不变的速度输出驱动脉冲。如果设定驱动速度小于初始速度，就没有加/减速驱动，而是定速驱动。使用搜寻原点、编码器Z相等信号时，找到信号后马上要立即停止的话，不必进行加/减速驱动，而是一开始就运行低速的定速驱动。

为了定速驱动，下列参数应相应预先设定：

- 范围 R
- 初始速度 SV
- 驱动速度 V



定速驱动

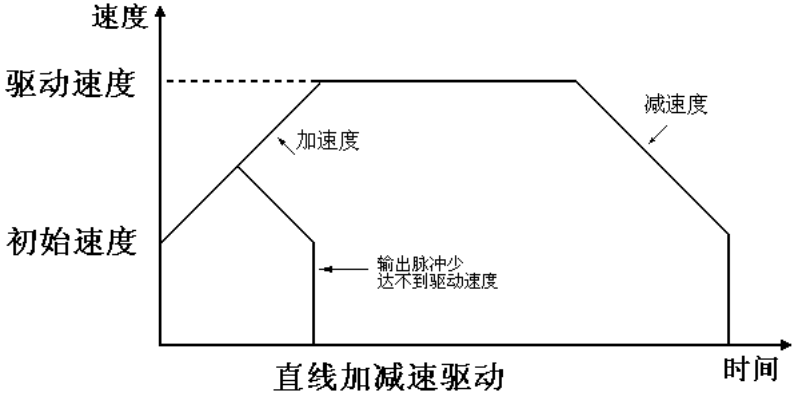
3.2 直线加/减速驱动

直线加/减速驱动是线性地从驱动开始的初始速度加速到指定的驱动速度。

定量驱动时，加速的计数器记录加速所累计的脉冲数。当剩余输出脉冲数少于加速脉冲后，就开始减速（自动减速）。减速时将用指定的减速度线性地减速至初始速度。

为了直线加/减速驱动，下列参数需预先设定：

- 范围 R
- 加速度 A 加速度和减速度
- 减速度 D 加/减速度个别设定时的减速度(必要时)
- 初始速度 SV
- 驱动速度 V

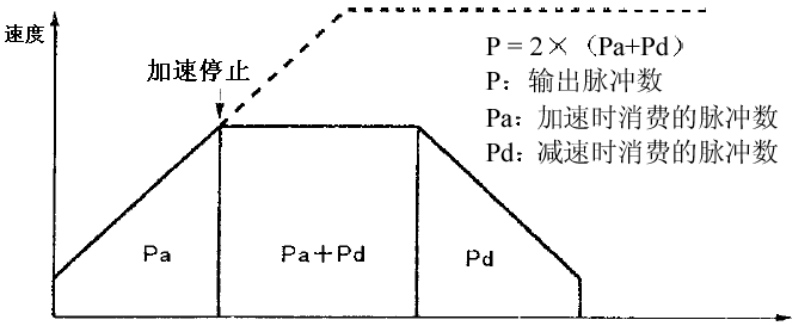


直线加减速驱动

定量驱动的三角防止

在直线加减速的定量驱动中，如果输出脉冲数小于要求的数时，会产生如上图的三角波形，此时三角防止功能会启动。

三角防止功能是在直线加减速的定量驱动中，哪怕输出脉冲数小于要求的数也防止三角波形，在加速中加速时和减速时消费的脉冲数大于总输出脉冲数的1/2 后，停止加速， 保持在定速域。因此哪怕输出脉冲小于输出脉冲数的1/2 ，也在定速域。



直线加减速驱动的三角防止

3.3 非对称直线加/减速驱动

往垂直方向移动对象物时，对对象物有重力加速度的负担，所以在这样加速度和减速度不同的非对称直线加减速的定量驱动中，最好变更上下移动的加速度和减速度。此时可以运行自动减速，事先不用设定手动减速点。图1是加速度比减速度大的例子，图2是减速度比加速度大的例子。

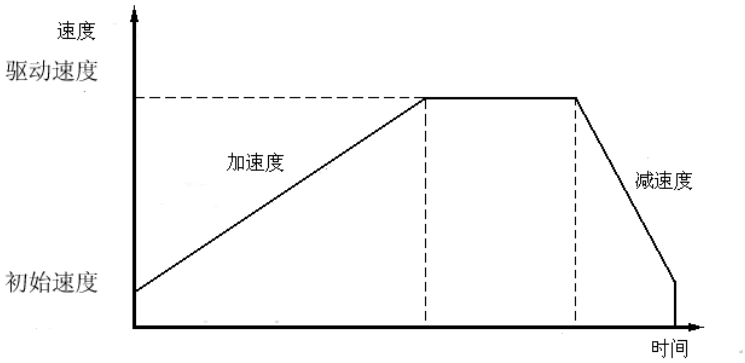


图 1　非对称直线加减速驱动(加速度＜减速度)

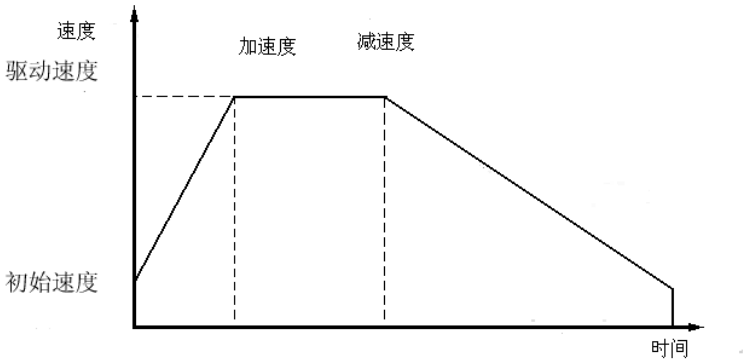


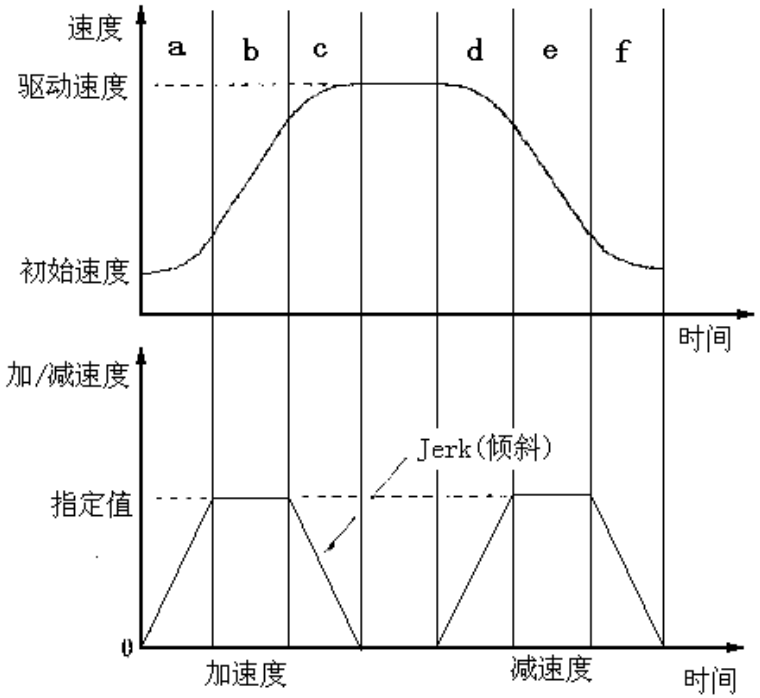
图 2　非对称直线加减速驱动(加速度＞减速度)

另外跟通常的直线加减速驱动一样需要设定下述的参数：

- 范围 R
- 加速度 A
- 减速度 D
- 初始速度 SV
- 驱动速度 V

3.4 S曲线加/速驱动

驱动速度加/减速时，可线性地增加/减少加速度/减速度以产生S型速度曲线，S曲线加/减速驱动如下图所示运行。



S 曲线加减速驱动

驱动开始加速时，加速度以指定的加速度的增加率(K) 从0线性增加至指定的数值(A)。 因此，这个速度曲线成为二次级抛物线(a 区间)。加速度达到指定数值(A)后保持此数值，这时速度曲线是直线型的，速度在加速中（b区间）。目标速度（V）和当前速度的差值比相应时间增加所增加的速度少时，加速度趋向0。减少率和增加率一样，指定的减速度的减少率（K）线性地减少，这时速度曲线成为二次抛物线（C区间）。本书定义这种具有部分固定加速度的加速为部分S曲线加速。

另一方面，在a区间若在加速度达到指定数值（A）前，目标速度（V）和当前速度的差值比相应时间增加所增加的速度少时，b区间就消失，只有a和c区间。这种没有固定加速度的加速称为完全S曲线加速。

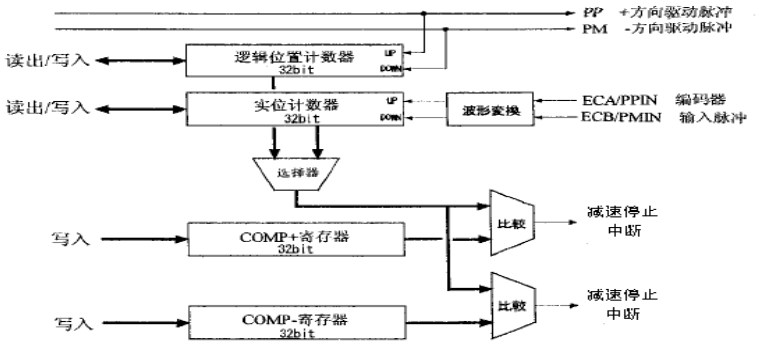
要执行S曲线加/减速，用户必须设定加速方式为S曲线加速，然后设定下列参数：

- 范围　R
- 加速度/减速度的变化率　K
- 加速度　A
- 减速度　D（必要时）
- 初始速度　SV
- 驱动速度　V

运行S 曲线加/减速驱动时的注意点:

- 运行S曲线加/减速定量驱动时驱动中不能变更驱动速度。
- 运行S曲线加/减速时不能驱动圆弧插补、连续插补。

四、位置管理



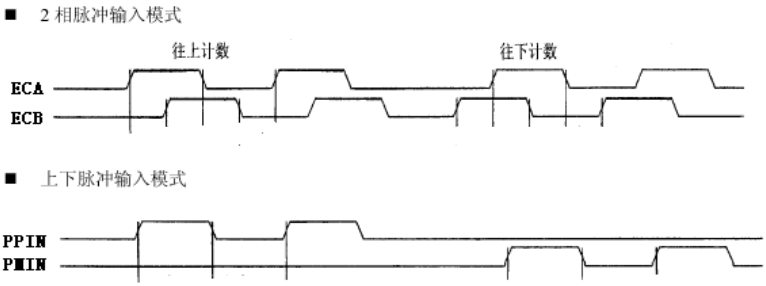
位置管理部分方框图

4.1 逻辑位置计数器和实位计数器

逻辑位置计数器是计数ADT850卡中的正/负方向输出脉冲，输出一个正向脉冲时，向上计1，输出一个负方向脉冲时，向下计1。

实位计数器计数来自外部编码器的输入脉冲，可以用命令选择输入脉冲的类型为A/B相信号或者独立2脉冲向上/下计数信号，计数方向可设定。

可以在任何时候写入或读出2个计数器的数据，计数范围在-2,147,483,648~+2,147,483,647之间



4.2 比较寄存器和软件限位

每轴有2个32位寄存器（COMP+ COMP-），能与逻辑位置计数器和实位计数器比较大小。把2个比较寄存器的比较对象定为逻辑计数器还是实位计数器可以设定，COMP+寄存器主要用来检测逻辑/实位计数器某个范围的上限，COMP-寄存器主要用来检测逻辑/实位计数器某个范围的下限。

软件限位设为有效后，在驱动中，如果逻辑/实位计数器的值大于COMP+的值就执行减速停止，此后只能执行负方向驱动命令至逻辑/实位计数器的值小于COMP+的值。同样，如果逻辑/实位计数器的值小于COMP-的值就执行减速停止，此后只能执行正方向驱动命令至逻辑/实位计数器的值大于COMP-的值。

可以在任何时候写COMP+寄存器和COMP-寄存器。

4.2 位置计数器的可变环形

逻辑位置计数器及实位数器是32位长的上/下环形计数器，因此从32位长的最大数值FFFFFFFFh往+方向计数的话，最后计数计到0，从0往-方向计数的话最后计数计到FFFFFFFFh。可变环功能是可以把这个环形计数器的最

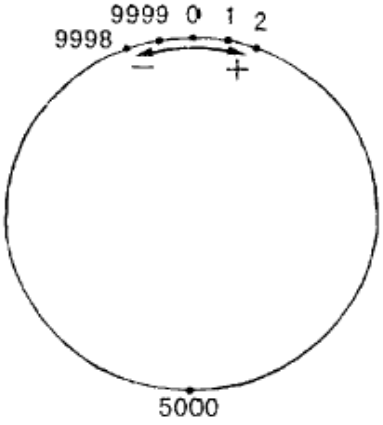
大数值任选设定，如果定位不是直线而是旋转运动的话，用此功能控制位置很方便。使可变环功能有效后，COMP+寄存器设定逻辑位置计数器的最大数值，在COMP-寄存器设定实位计数器的最大数值。

如在X轴为旋转轴时，假设X轴为10000个脉冲旋转一圈，设定可变环功能有效，在COMP+寄存器设定9999，同时使用实位计数器的话，在COMP-寄存器设定9,999。

这时计数动作：

往+方向，向上计数时：… → 9998 → 9999 → 0 → 1 …

往-方向，向下计数时：… → 1 → 0 → 9999 → 9998 …



位置计数器的可变环最大值 9,999 操作

这样就不需要考虑计数值超过10000时的计算问题，计数范围一定在0-9999之间。

注意

- 每个轴都要设定可变环功能的有效/无效但是不能分别设定逻辑位置计数器和实位计数器的有效/无效
- 使可变环功能有效后不能使用软件限制功能

五、插补

ADT850卡可以进行2-4轴的直线插补、2轴圆弧插补。

在插补驱动过程中，插补运算是在指定X轴的基本脉冲时序下运行的，因此进行插补命令之前，先要设定指定X轴的初始速度、驱动速度等参数。（Z-W插补时是以Z轴速度为基准）

■ **插补时的越限错误**

插补驱动时每个驱动轴都能进行硬件限制和软件限制，在插补驱动中任何轴的限制有动作，插补驱动就停止。

注意

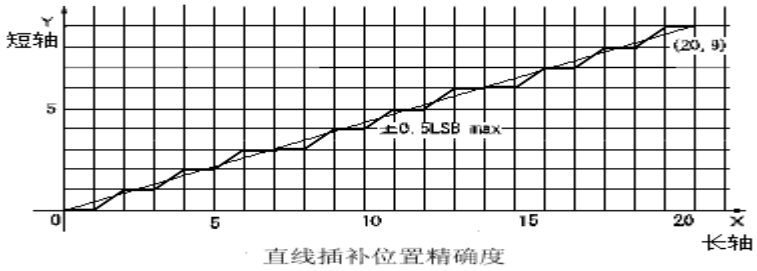
运行圆弧插补时任何方向（+方向/-方向）的硬件限制、软件限制有效，插补都会停止。因此，使用圆弧插补要非常小心，不能脱离限制区域。

■ **伺服马达的到位信号**

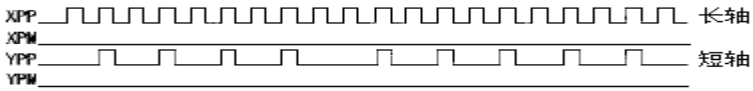
在插补驱动中，各轴到位信号INPOS一旦有效插补驱动就结束，结束后所有轴的INPOS信号处于有效电平。

5.1 2-4轴直线插补

设定相对于当前位置的终点坐标后就开始进行线性插补。直线插补的坐标范围是带符号的24位字长，插补范围为从各轴当前位置到-8,388,607~+8,388,607之间。



直线插补位置精确度

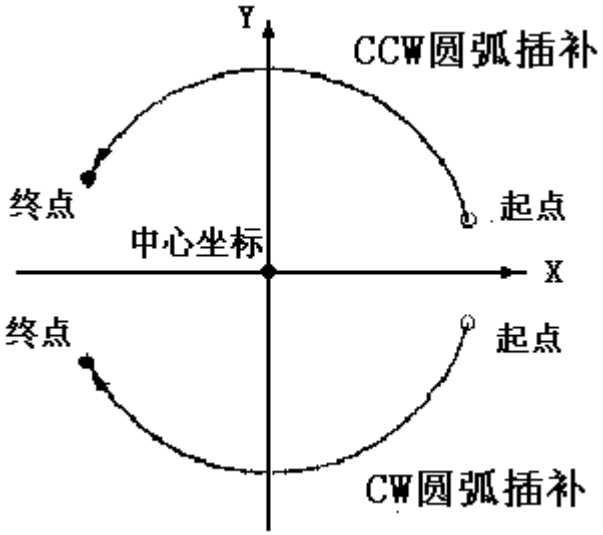


终点(X: 20, Y: 9)驱动脉冲输出例子

如上图所示，对指定直线的位置精确度，在整个插补范围内有±0.5LSB。上图还有直线插补的驱动脉冲输出例子，在设定的终点数值中绝对值最大的轴是长轴，在插补驱动中此轴一直输出脉冲，其它的轴是短轴，根据直线插补算术的结果，有时输出脉冲，有时不输出脉冲。

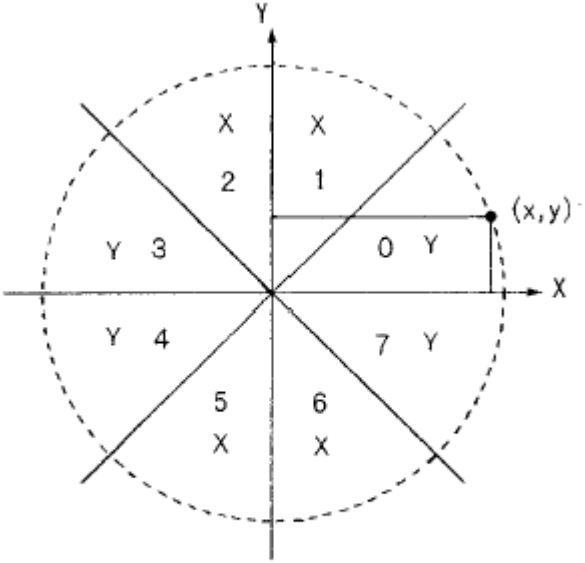
5.2 圆弧插补

设定相对当前位置始点的圆弧中心坐标及终点坐标后执行圆弧插补。



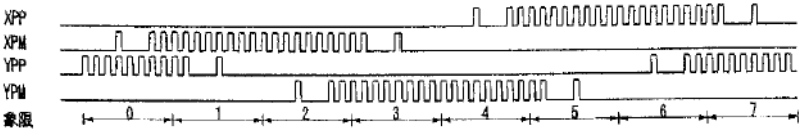
CW圆弧插补从当前坐标至终点坐标以顺时针方向绕中心坐标画圆弧，CCW圆弧插补以逆时针方向绕中心坐标画圆弧，如果终点设为（0，0）能画整个圆。

至于圆弧插补的算法如下图所示由X轴和Y轴定义一个平面，绕中心坐标把它分为0~7的8个象限，如图所示在0象限的插补坐标（X，Y）上，Y绝对值一直比X的绝对值小，绝对值小的轴为短轴。1、2、 5、6象限是X轴，0、3、4、7象限是Y轴，短轴在这些象限之间一直输出驱动脉冲，长轴根据圆弧插补运算结果，有时输出脉冲，有时不输出脉冲。

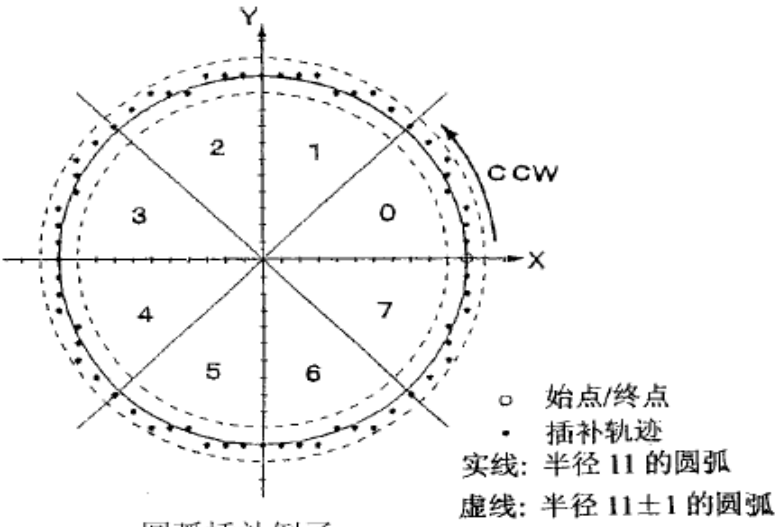


圆弧插补算法 0~7 象限和短轴

下面是输出一个整圆的例子，以及输出脉冲的示例



圆弧插补驱动脉冲输出例子



圆弧插补例子

■ **终点判断**

对于圆弧插补，在插补驱动开始前，把当前坐标设为（0， 0），根据中心坐标的数值决定半径画圆。圆弧算法的误差在插补驱动范围内有1个脉冲，因此，指定的终点可能不在圆的轨迹上。圆弧插补进入终点所在的象限时，只要结束点值与终点的短轴数值一致圆弧插补就结束。

5.3 连续插补

对于没有连续插补功能的控制卡，如果需要在上一插补点结束后继续下一插补，只能不断查询上一插补是否完成，然后输出下一插补的数据，如果上位机的速度较慢，或者上位机运行多任务操作系统，在两次插补之间就会出现停顿，会影响插补的效果，而且插补速度很难提高。

而ADT850卡带有连续插补功能，可以很好的解决这一问题，它可在上一插补未结束时，输出下一插补的数据，即使在很慢的电脑上，也可达到好的效果。

连续插补是直线→圆弧插补→直线插补→…这样在每个插补节点之间

不停地驱动，连续插补。在连续插补驱动中，先读取连续插补的允许写入状态和插补驱动状态，如果插补未结束并且允许写入，即可写入下一插补命令。因此，在所有的插补节点中，从连续插补驱动开始至结束的时间，必须长于设定下一个插补节点的数据和发命令的时间。

- **连续插补中发生的错误**

在连续插补驱动过程中，若发生越限等错误驱动，就立即在当前插补节点上停止。在停止的插补节点上，下一节点的数据和命令虽在，但命令是无效的。此外，在发插补命令前，必须检查错误，若没有检查，当发生错误停止驱动后，这些数据和命令将无效，而从下面第2个插补节点开始运行，一定要进行检查，若发现错误的话，要脱离连续插补的循环。

连续插补中有圆弧插补时，圆弧插补终点的短轴数值也许会比真值偏差1个脉冲，因此为了避免累积每个节点的误差，事先要确认每个圆弧插补的终点，然后考虑怎么运行连续插补。

5.4 加/减速驱动的插补

插补一般用定速驱动，不过ADT850卡可以用直线加/减速驱动或S 曲线加/减速驱动（只可做直线插补）运行插补。

在连续插补时为了实现加/减速驱动，可以使用减速有效命令 和减速无效命令。在插补驱动时，减速有效命令是使自动减速或手动减速变为有效，减速无效命令是使它变为无效，复位时都是无效状态。用加/减速单独运行插补驱动时，驱动开始之前一定要设定成减速有效状态，在驱动中写入减速有效命令也不能变为有效。

- **直线插补的加/减速驱动**

在直线插补中可以运行直线加/减速驱动及S曲线加减速驱动减速

- **圆弧插补的加/减速驱动**

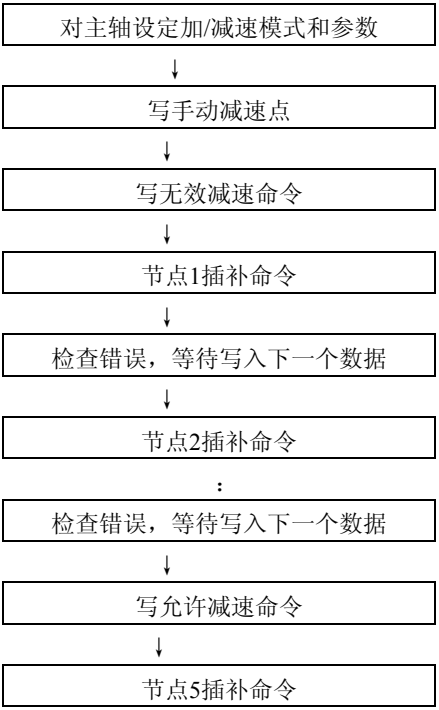
在圆弧插补位模式插补中只能用手动减速的直线加/ 减速驱动不能使用S曲线加/减速驱动及自动减速

- **连续插补的加/减速驱动**

在连续插补中，只能用手动减速的直线加/减速驱动，不能用S曲线加/减速驱动及自动减速。在连续插补中，要事先设定手动减速点，这个手动减速点设定在运行减速的最终节点上，并设定从X轴输出的基本脉冲的数

值，连续插补时，先把减速设定为无效，然后开始插补驱动，在要减速的最终插补节点上，写入插补命令之前，写入允许减速命令，开始最终插补节点的驱动时，减速就有效。从最终插补节点的开始计算主轴输出的基本脉冲数大于手动减速点的数值时，减速就开始。

比如从插补节点1至5的连续插补中，在最终节点5上用手动减速的话，有下述的程序。



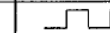
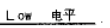
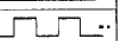
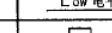
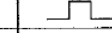
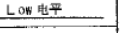
由从节点5 开始的X轴基本脉冲数的数值来设定手动减速点，比如假定减速化费2,000 脉冲，在节点5上输出的基本脉冲的总脉冲数是5,000 的话，手动减速点就设定为5,000-2,000=3,000。

开始至停止一定要在1个节点内运行减速，减速停止的最终插补节点需要从其X或Z轴输出的基本脉冲总数要大于在减速中化费的脉冲数。

六、脉冲输出方式

驱动输出脉冲有下图所示的2种脉冲输出方式，以独立2脉冲方式，正方向驱动时由PU/CW 输出驱动脉冲，负方向驱动时由DR/CCW输出驱动脉冲，采用1脉冲方式则由PU/CW输出驱动脉冲，由DR/CCW输出方向信号。

脉冲/方向都是正逻辑设定时

脉冲输出方式	驱动方向	输出信号波形	
		PU/CW 信号	DR/CCW 信号
独立2脉冲方式	+方向驱动输出		
	-方向驱动输出		
1脉冲方式	+方向驱动输出		
	-方向驱动输出		

七、硬件限制信号

硬件限制信号（LMT+，LMT-）是限定正方向和负方向驱动脉冲的输入信号，当限制信号的逻辑电平和限制信号有效时，可以由命令选择减速停止或立即停止。

八、伺服马达对应的信号

与伺服马达驱动器连接的输入信号有到位信号（INPOS） 和警报信号（ALARM）。每个信号的有效/无效及逻辑电平都可以设定。

INPOS 输入信号与伺服马达定位完毕信号对应，设定模式为有效时，一个驱动结束后，等待INPOS 输入信号有效，驱动状态返回结束。ALARM 输入信号接受从伺服马达驱动器的警报信号，设定为有效时，一直监视 ALARM 输入信号，若信号有效，在驱动中立即停止驱动。这些用于伺服马达驱动器的输入信号可以用通用I/O函数读其状态，通用输出信号可用于偏置计数器清除、警报复位、伺服开启等。

第六章 ADT850 库函数列表

一、基本参数设置类

int adt850_initial(void);
int adt850_end(void);
int set_stop0_mode(int cardno, int axis, int value,int logic);
int set_stop1_mode(int cardno, int axis, int value,int logic);
int set_stop2_mode(int cardno, int axis, int value,int logic);
int set_actualcount_mode(int cardno, int axis, int value,int dir,int freq);
int set_pulse_mode(int cardno, int axis, int value,int logic, int dir_logic);
int set_limit_mode(int cardno, int axis, int value,int logic);
int set_softlimit_mode1(int cardno, int axis, int value);
int set_softlimit_mode2(int cardno, int axis, int value);
int set_softlimit_mode3(int cardno, int axis, int value);
int set_inpos_mode(int cardno, int axis, int value,int logic);
int set_alarm_mode(int cardno, int axis, int value, int logic);
int set_ad_mode(int cardno, int axis, int value);
int set_dec1_mode(int cardno, int axis, int value);
int set_dec2_mode(int cardno, int axis, int value);
int set_circle_mode(int cardno, int axis, int value);
int set_input_filter(int cardno,int axis,int number,int value);

二、驱动状态检查类

int get_status(int cardno,int axis,int *value)
int get_stopdata(int cardno,int axis,int *value)
int get_inp_status(int card,int no,int *value)
int get_inp_status2(int card,int no,int *value)

三、运动参数设定类

int set_range(int cardno,int axis,long value);
int set_acac(int cardno,int axis,long value);
int set_acc(int cardno,int axis,long value);
int set_dec(int cardno,int axis,long value);
int set_startv(int cardno,int axis,long value);
int set_speed(int cardno,int axis,long value);
int set_command_pos(int cardno,int axis,long value);
int set_actual_pos(int cardno,int axis,long value);
int set_comp1(int cardno,int axis,long value);
int set_comp2(int cardno,int axis,long value);
int set_dec_pos(int cardno,int axis,long value);

四、运动参数检查类

int get_command_pos(int cardno,int axis,long *pos)
int get_actual_pos(int cardno,int axis,long *pos)
int get_speed(int cardno,int axis,long *speed)
int get_ad(int cardno,int axis,long *ad)

五、驱动类

int pmove(int cardno,int axis,long pulse)
int continue_move(int cardno,int axis,int dir)
int dec_stop(int cardno,int axis)

```
int sudden_stop(int cardno,int axis)
int inp_move2(int card,int no,long pulse1,long pulse2)
int inp_cw_arc(int card,int no,long x,long y,long i,long j)
int inp_ccw_arc(int card,int no,long x,long y,long i,long j)
int inp_move3(int cardno,long pulse1,long pulse2,long pulse3)
int inp_move4(int cardno,long pulse1,long pulse2,long
        pulse3,long pulse4)
int inp_dec_enable(int cardno,int no);
int inp_dec_disable(int cardno,int no);
int inp_arc_z_move(int cardno, long x,long y,long i,long
        j ,long z,int dir,int speed);
```

六、开关量输入输出类

```
int read_di(int cardno,unsigned long *value)
int write_do(int cardno,unsigned long value)
int read_bit(int cardno,int number)
int write_bit(int cardno,int number,int value)
```

七、中断类(仅用于 DOS)

```
int set_int_enable(int cardno,int intflag);
int set_int_factor(int cardno,int axis,unsigned
long int_factor);
int get_int_status(int cardno,int axis,unsigned long int_factor);
```

第七章 ADT850 库函数详解

一、基本参数设置类

1.1 初始化adt850卡

int adt850_initial(void);

返回值为系统中 adt850 卡的个数

如果为 3，则下面的可用卡号分别为 0、1、2；

如果为 0 说明没有安装 ADT850 卡；

如果为-1 说明 ADT850 卡驱动程序未正确安装。

初始化应在应用程序第一个调用，以确认可使用的卡数以及初始化一些参数。

初始化后各状态为：

脉冲输出方式为 脉冲+方向方式

反馈输入为 A/B 相编码脉冲输入，4 倍频

停止信号 STOP0，STOP1，STOP2 均为无效

限位信号 LMT+，LMT- 为低电平有效，立即停止

软件限位无效

伺服到位信号 nINPOS 无效

伺服报警信号 nALARM 无效

加速方式为直线加/减速，对称加/减速，自动减速

计数器的可变环功能无效

输入滤波无效

1.2 释放adt850卡

int adt850_end(void);

返回值 0：正确 1：错误

此函数应在程序结束时调用。（仅用于 Windows NT/2000）。

用于将 ADT850 卡占用的资源释放。

1.3 设置stop0信号的有效/无效和逻辑电平

int set_stop0_mode(int cardno, int axis, int value,int logic);

cardno　卡号

axis　轴号（1-4）

value0：无效　1：有效

logic　0：低电平停止 1：高电平停止

返回值　0：正确　1：错误

初始化时状态为：信号无效，低电平停止

注意：停止方式取决于是加减速驱动还是匀速驱动，对加减速驱动，是减速停止，对匀速驱动，是立即停止。STOP1、STOP2信号也是一样。

1.4 设置stop1信号的有效/无效和逻辑电平

int set_stop1_mode(int cardno, int axis, int value,int logic);

cardno　卡号

axis　轴号（1-4）

value0：无效　1：有效

logic　0：低电平停止 1：高电平停止

返回值　0：正确　1：错误

初始化时状态为：信号无效，低电平停止

1.5 设置stop2信号的有效/无效和逻辑电平

int set_stop2_mode(int cardno, int axis, int value,int logic);

cardno　卡号

axis　轴号（1-4）

value0：无效　1：有效

logic　0：低电平停止 1：高电平停止

返回值　0：正确　1：错误

初始化时状态为：信号无效，低电平停止

注意：STOP2 信号有效时同时可清除实位计数器。

因为由于伺服系统或机械系统的延迟，如果在停止驱动后再用软件清除实位计数器，会使原点位置有一定的误差，可使用本功能达到更高的精度。

只有 STOP2 有此功能。

1.6 设置实位计数器（编码器输入）的工作方式

int set_actualcount_mode(int cardno, int axis, int value,int dir,int freq);

cardno　卡号

axis　轴号（1-4）

value输入脉冲方式

0：A/B 脉冲输入　　1：上/下（PPIN/PMIN）脉冲输入

dir　计数方向

0：　A 超前 B 或 PPIN 脉冲输入向上计数

B 超前 A 或 PMIN 脉冲输入向下计数

1：　B 超前 A 或 PMIN 脉冲输入向上计数

A 超前 B 或 PPIN 脉冲输入向下计数

freq　A/B 脉冲输入时的倍频，上/下脉冲输入时无效

0：4 倍频

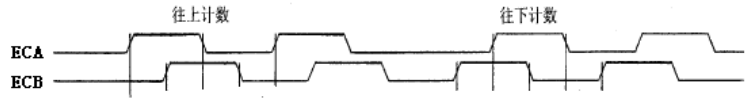
1：2 倍频

2：不倍频

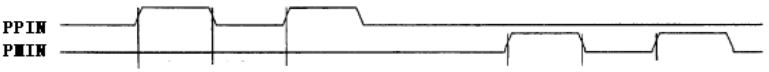
返回值　0：正确　　1：错误

初始化时状态为：A/B 相脉冲输入，方向为 0，4 倍频

■ 2 相脉冲输入模式



■ 上下脉冲输入模式



对于大多数位置反馈装置，均采用编码器或光栅尺，应设置为 A/B 相脉冲输入方式，对此种方式，可采用倍频技术提高精度，可设定为 4

倍频或 2 倍频，也可不使用倍频，对于 4 倍频，如果采用每转 1000 个脉冲的编码器，正转一圈，计数值应增加 4000，即精度提高了 4 倍。

1.7 设置输出脉冲的工作方式

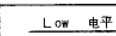
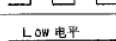
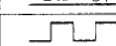
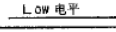
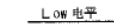
int set_pulse_mode(int cardno, int axis, int value,int logic,int dir_logic);

cardno　卡号

axis　轴号（1-4）

value0：脉冲+脉冲方式　1：脉冲+方向方式

脉冲/方向都是正逻辑设定时

脉冲输出方式	驱动方向	输出信号波形	
		PU/CW信号	DR/CCW信号
独立2脉冲方式	+方向驱动输出		
	-方向驱动输出		
1脉冲方式	+方向驱动输出		
	-方向驱动输出		

logic　0：　正逻辑脉冲　　1：　负逻辑脉冲

正逻辑脉冲：  　　负逻辑脉冲： 

dir-logic　0：方向输出信号正逻辑　1：方向输出信号负逻辑

dir_logic	正方向脉冲输出时	负方向脉冲输出时
0	Low	Hi
1	Hi	Low

返回值　0：正确　　1：错误

初始化时状态为：脉冲+方向方式，正逻辑脉冲，方向输出信号正逻辑

1.8 设定正/负方向限位输入nLMT信号的模式设定

int set_limit_mode(int cardno, int axis, int value,int logic);

cardno　卡号

axis　轴号（1-4）

value　0：有效时立即停止　　1：有效时减速停止

logic　　0：低电平有效　　1：高电平有效

返回值　　0：正确　　1：错误

初始化时状态为：立即停止，低电平有效

✍ 注意：　限位信号不能设置成有效/无效。

1.9 COMP+寄存器做软件限制的设定

int set_softlimit_mode1(int cardno,　int axis,　int value);

cardno　　卡号

axis　　轴号（1-4）

value　　0：无效　　1：有效

返回值　　0：正确　　1：错误

初始化时状态为：无效

✍ **注意：软件限位总是减速停止，那么计数值就可能回超出设定值，因此在设定范围时应考虑到这一点。**

1.10 COMP-寄存器做软件限制的设定

int set_softlimit_mode2(int cardno,　int axis,　int value);

cardno　　卡号

axis　　轴号（1-4）

value　　0：无效　　1：有效

返回值　　0：正确　　1：错误

初始化时状态为：无效

✍ **注意：同上**

1.11 COMP+/-寄存器的比较对象设定

int set_softlimit_mode3(int cardno,　int axis,　int value);

cardno　　卡号

axis　　轴号（1-4）

value　　0：逻辑位置计数器　　1：实际位置计数器

返回值　　0：正确　　1：错误

初始化时状态为：逻辑位置计数器

此函数是设定软件限位的比较对象。

1.12 伺服到位信号nINPOS的设定

int set_inpos_mode(int cardno, int axis, int value,int logic);

cardno	卡号	
axis	轴号（1-4）	
value	0：无效	1：有效
logic	0：低电平有效	1：高电平有效
返回值	0：正确	1：错误

初始化时状态为：无效，低电平有效

注意：如果 nINPOS 未与伺服接线，或者使用步进电机，请不要设成有效。

1.13 伺服报警信号nALARM的设定

int set_alarm_mode(int cardno, int axis, int value,int logic);

cardno	卡号	
axis	轴号（1-4）	
value	0：无效	1：有效
logic	0：低电平有效	1：高电平有效
返回值	0：正确	1：错误

初始化时状态为：无效，低电平有效

注意：如果 nALARM 未与伺服接线，或者使用步进电机，请不要设成有效。

1.14 加/减速方式的设定

int set_ad_mode(int cardno, int axis, int value);

cardno	卡号	
axis	轴号（1-4）	
value	0：直线加/减速	1：S曲线加/减速
返回值	0：正确	1：错误

初始化时状态为：直线加/减速

1.15 非对称梯形加/减速的设定

int set_dec1_mode(int cardno, int axis, int value);

cardno　卡号

axis　轴号（1-4）

value 减速度使用

0：加速度的值（即对称加减速）

1：减速度的值（即非对称加减速）

返回值　0：正确　1：错误

初始化时状态为：对称加减速

1.16 加/减速定量驱动的减速方式的设定

int set_dec2_mode(int cardno, int axis, int value);

cardno　卡号

axis　轴号（1-4）

value　0：自动减速　1：手动减速

返回值　0：正确　1：错误

初始化时状态为：自动减速

注意：大部分情况可使用自动减速，使用手动减速应设定好减速点。

1.17 计数器的可变环功能的设定

int set_circle_mode(int cardno, int axis, int value);

cardno　卡号

axis　轴号（1-4）

value　0：无效　1：有效

返回值　0：正确　1：错误

初始化时状态为：无效

可变环功能说明参见前面。

1.18 输入信号滤波功能设置

int set_input_filter(int cardno,int axis,int number,int value);

cardno　卡号

axis　轴号

number　输入类别

1：LMT+、LMT-、STOP0、STOP1

2：STOP2

3：nINPOS、nALARM

4：nIN

即可分别设置上面四类输入信号的滤波状态

value　0：滤波无效

1：滤波有效

初始化时状态为：无效

1.19 输入信号滤波时间常数设置

int set_filter_time(int cardno,int axis,int value);

cardno　卡号

axis　轴号

value 范围1-8，含义如下：

value	可以除去最大噪音幅度	输入信号延迟
1	1.75 μ SEC	2 μ SEC
2	224 μ SEC	256 μ SEC
3	448 μ SEC	512 μ SEC
4	896 μ SEC	1.024mSEC
5	1.792 mSEC	2.048mSEC
6	3.584 mSEC	4.096mSEC
7	7.168 mSEC	8.192mSEC
8	14.336mSEC	16.384mSEC

二、驱动状态检查类

2.1 获取各轴的驱动状态

int get_status(int cardno,int axis,int *value)

cardno　卡号

axis　轴号（1-4）

value　驱动状态的指针

0：驱动结束

非 0：value 为两个字节长度的值，各位的含义如下：

D0 为最低位　D15 为最高位

D0：表示逻辑/实位计数器和 COMP+寄存器的大小关系

1 逻辑/实位计数器>=COMP+寄存器

0 逻辑/实位计数器<COMP+寄存器

D1：表示逻辑/实位计数器和COMP-寄存器的大小关系

1 逻辑/实位计数器>=COMP-寄存器

0 逻辑/实位计数器<COMP-寄存器

D2：在加/减速驱动中，加速时为1

D3：在加/减速驱动中，定速时为1

D4：在加/减速驱动中，减速时为1

D5：在S曲线加/减速驱动中，加速度/减速度增加时为1

D6：在S曲线加/减速驱动中，加速度/减速度不变时为1

D7：在S曲线加/减速驱动中，加速度/减速度减少时为1

D8-D15：未用

返回值　0：正确　　1：错误

2.2 获取各轴的错误停止信息

int get_stopdata(int cardno,int axis,int *value)

cardno　卡号

axis　轴号

value　错误状态的指针

0：无错误

非 0：value 为两个字节长度的值，各位的含义如下：

D0 为最低位　D15 为最高位

D0：由STOP0停止

D1：由STOP1停止

D2：由STOP2停止

D3；由正限位LMT+停止

D4：由负限位LMT+停止

D5：由伺服报警停止

D6：COMP+ 寄存器限位驱动停止

D7：COMP- 寄存器限位驱动停止

D8-D15：未用

返回值　0：正确　　1：错误

2.3 获取插补的驱动状态

int get_inp_status(int card,int no,int *value)

cardno　卡号

no　1：X-Y 轴插补或 3 轴以上插补

2：Z-W 轴插补

value 插补状态的指针

0：插补结束　　1：正在插补

返回值　0：正确　　1：错误

2.4 取连续插补的允许写入状态

int get_inp_status2(int card,int no,int *value)

cardno　卡号

no　1：X-Y 轴插补或 3 轴以上插补

2：Z-W 轴插补

value 状态的指针

0：不允许写入　　1：允许写入

返回值　0：正确　　1：错误

注意：如果驱动结束，则状态也为 0，因此在连续插补时，还须注意是否有错误发生，以便退出连续插补。

三、运动参数设定类

以下参数在初始化后值不确定，使用前必须设定

3.1 范围设定

int set_range(int cardno,int axis,long value);

cardno　卡号

axis　轴号

valueR值 范围（8000000-16000）

返回值　0：正确　　1：错误

注意：范围是决定速度、加/减速度、加/减速度的变化率的倍率参数，假设把范围数值作为R 倍率是下述的算式

倍率＝8000000 / R

因为驱动速度、初始速度、加/减速度等参数的设定范围在1~8000。若需要设定为8000以上的数值的话，必须提高倍率。提高倍率后可以高速驱动，但是速度分辨率变粗。请在使用的速度范围内设定最小的数值。比如如果需要40KPPS 的速度，在速度范围内1~8000 倍率中最好是5， 即设定R 为1600000。

R值范围是8000000-16000，相应的倍率为1-500。

在驱动中请不要变更范围（R） 否则速度会变乱。

如：

set_range(0,1,800000)；

为设定第一块卡的X轴的倍率为8000000/800000=10倍

set_range(0,3,16000)；

为设定第一块卡的Z轴的倍率为8000000/16000=500倍

3.2 加/减速度的变化率设定

int set_acac(int cardno,int axis,long value);

cardno 卡号

axis 轴号

valueK值（1-65535）

返回值 0：正确 1：错误

加/减速度的变化率设定值是决定S曲线加/减速的加速度及减速度在单位时间内增加/减少率的参数。把加/减速度的变化率的设定值作为K的话加/减速度的变化率由下述算式表示。

加减速度的变化率（PPS / SEC^2 ）=(62500000/K)*倍率

即：

加减速度的变化率（PPS / SEC^2 ）=(62500000/K)*(8000000/R)

加/减速度的变化率设定值K的设定范围是1~65,535

如：

set_range(0,1,800000);

set_acac(0,1,100);

则加减速度的变化率为（62500000/100）*（8000000/800000）

=6250000 PPS/SEC^2

3.3 加速度设定

int set_acc(int cardno,int axis,long value);

cardno 卡号

axis 轴号

valueA值（1-8000）

返回值 0：正确 1：错误

它是作为直线加减速驱动中直线加速及减速的参数，在S 曲线加/减速驱动中，加速度及减速度线性从0 增加至加速度的设定值。

加速度设定值为A 加速度是下述算式

加速度（PPS/SEC）=A*125*倍率

即 加速度（PPS/SEC）=A*125*（8000000/R）

加速度设定值A 的设定范围是1~8,000。

如：

set_range(0,1,80000);

set_acc(0,1,100);

则加速度为：

100*125*（8000000/80000）=1250000 PPS/SEC

3.4 减速度设定

int set_dec(int cardno,int axis,long value);

cardno 卡号

axis 轴号

valueD值（1-8000）

返回值 0：正确 1：错误

在加速度/减速度的分别设定模式中，它是直线加/ 减速度驱动在减速时的减速度参数。

在这个模式的S曲线加/减速驱动中，减速度直线从0增加至减速度的设

定值，把减速度设定值作为D，减速度是下述算式

减速度（PPS/SEC）=D*125*倍率

即 减速度（PPS/SEC）=D*125*（8000000/R）

3.5 初始速度设定

int set_startv(int cardno,int axis,long value);

cardno 卡号

axis 轴号

valueSV值（1-8000）

返回值 0：正确 1：错误

它是加/ 减速驱动的加速开始时的速度和减速结束时的速度，初始速度设定数值为SV 的话，初始速度是下述算式

初始速度（PPS）=SV*倍率

即 初始速度（PPS）=SV*（8000000/R）

3.6 驱动速度设定

int set_speed(int cardno,int axis,long value);

cardno 卡号

axis 轴号

valueV值（1-8000）

返回值 0：正确 1：错误

它是加/减速驱动中达到定速区域的速度。定速驱动从该速度开始运行。把这个驱动速度设定在初始速度以下的话，不运行加/减速驱动，开始就运行定速驱动

驱动速度（PPS）=V*倍率

即 驱动速度（PPS）=V*（8000000/R）

在驱动中可以随便变更驱动速度，在加/减速度驱动的定速区域中，如果重新设定驱动速度，就开始加速或减速到重新设定的速度。达到重新设定的速度后，开始运行定速驱动。

注意事项

S曲线加/减速的定量脉冲驱动在驱动中不能变更驱动速度。此外S曲线加/减速的连续驱动如果在加速中，减速中变更速度，就不能运行正确的S曲线，所以请在定速区域内变更运行。

直线加/减速的定量脉冲驱动时，在驱动中如果经常变更驱动速度，则在输出脉冲结束的减速时，出现以初始速度拖曳驱动的可能性较大。

3.7 逻辑位置计数器设定

设定逻辑位置计数器的数值

int set_command_pos(int cardno,int axis,long value);

cardno 卡号

axis 轴号

value范围值（-2147483648~+2147483647）

返回值 0：正确 1：错误

逻辑位置计数器任何时候都能写、任何时候都能读

3.8 实际位置计数器设定

设定实际位置计数器的数值

int set_actual_pos(int cardno,int axis,long value);

cardno 卡号

axis 轴号

value范围值（-2147483648~+2147483647）

返回值 0：正确 1：错误

实际位置计数器任何时候都能写、任何时候都能读

3.9 COMP+寄存器设定

设定COMP+寄存器的数值

int set_comp1(int cardno,int axis,long value);

cardno 卡号

axis 轴号

value范围值（-2147483648~+2147483647）

返回值 0：正确 1：错误

COMP+寄存器任何时候都能写

3.10 COMP-寄存器设定

设定COMP-寄存器的数值

int set_comp2(int cardno,int axis,long value);

cardno　卡号

axis　轴号

value范围值（-2147483648~+2147483647）

返回值　0：正确　1：错误

COMP-寄存器任何时候都能写

3.11 手动减速点设定

设定手动减速点的数值

int set_dec_pos(int cardno,int axis,long value);

cardno　卡号

axis　轴号

value范围值（0~268435455）

返回值　0：正确　1：错误

设定在手动减速模式的加减速定量驱动上的减速点

手动减速点 = 输出脉冲数 - 化费在减速上的脉冲数

在使用手动减速模式时必须先设定好手动减速点

四、运动参数检查类

以下函数在任何时候均可调用

4.1 获取各轴的逻辑位置

int get_command_pos(int cardno,int axis,long *pos)

cardno　卡号

axis　轴号

pos　逻辑位置值的指针

返回值　0：正确　1：错误

此函数可随时得到轴的逻辑位置，在电机未失步的情况下可代表轴的当前位置。

4.2 获取各轴的实际位置（即编码器反馈输入值）

int get_actual_pos(int cardno,int axis,long *pos)

cardno　卡号

axis　　轴号

pos　　实际位置值的指针

返回值　　0：正确　　1：错误

此函数可随时得到轴的实际位置，在电机有失步的情况下也能知道轴的当前位置。

4.3 获取各轴的当前驱动速度

int get_speed(int cardno,int axis,long *speed)

cardno　　卡号

axis　　轴号

speed　　当前驱动速度的指针

返回值　　0：正确　　1：错误

数据的单位和驱动设定数值V一样。

此函数可随时得到轴的驱动动速度。

4.4 获取各轴的当前加速度

int get_ad(int cardno,int axis,long *ad)

cardno　　卡号

axis　　轴号

ad　　当前加速度的指针

返回值　　0：正确　　1：错误

数据的单位和驱动设定数值A一样。

此函数可随时得到轴的驱动动速度。

五、驱动类

5.1 定量驱动

int pmove(int cardno,int axis,long pulse)

cardno　　卡号

axis　　轴号

pulse输出的脉冲数

>0：正方向移动

<0：负方向移动

范围（-268435455~+268435455）

返回值　0：正确　　1：错误

写入驱动命令之前一定要正确地设定速度曲线所需的参数

5.2 连续驱动

int continue_move(int cardno,int axis,int dir)

cardno　卡号

axis　轴号

dir　驱动的方向

0：正方向移动

1：负方向移动

返回值　0：正确　　1：错误

写入驱动命令之前一定要正确地设定速度曲线所需的参数

5.3 驱动减速停止

int dec_stop(int cardno,int axis)

cradno　卡号

axis　轴号

返回值　0：正确　　1：错误

在驱动脉冲输出过程中，此命令作出减速停止，驱动速度比初始速度慢的时候也可以用本命令立即停止。

5.4 驱动立即停止

int sudden_stop(int cardno,int axis)

cardno　卡号

axis　轴号

返回值　0：正确　　1：错误

立即停止正在驱动中的脉冲输出，在加/减速驱动中也立即停止。

5.5 两轴直线插补

int inp_move2(int card,int no,long pulse1,long pulse2)

cardno　卡号

no　参与插补的轴号

1：X-Y

2：Z-W

pulse1,pulse2　移动的相对距离

范围（-8388608~+8388607）

返回值　0：正确　　1：错误

注意：X-Y插补的速度以X轴速度为基准，Y无须设定。Z-W插补的速度以Z轴速度为基准，W无须设定。

5.6　顺时针CW圆弧插补

int inp_cw_arc(int card,int no,long x,long y,long i,long j)

cardno　卡号

no　参与插补的轴号

1：X-Y

2：Z-W

x,y　圆弧插补的终点位置（相对于起点）

范围（-8388608~+8388607）

i,j　圆弧插补的圆心点位置（相对于起点）

范围（-8388608~+8388607）

返回值　0：正确　　1：错误

注意：X-Y插补的速度以X轴速度为基准，Y无须设定。Z-W插补的

`　速度以Z轴速度为基准，W无须设定。

5.7　逆时针CCW圆弧插补

int inp_ccw_arc(int card,int no,long x,long y,long i,long j)

cardno　卡号

no　1：X-Y

2：Z-W

x,y　圆弧插补的终点位置（相对于起点）

范围（-8388608~+8388607）

i,j　圆弧插补的圆心点位置（相对于起点）

范围（-8388608~+8388607）

返回值　0：正确　　1：错误

注意：X-Y插补的速度以X轴速度为基准，Y无须设定。Z-W插补的

速度以Z轴速度为基准，W无须设定。

5.8 三轴直线插补

int inp_move3(int cardno,long pulse1,long pulse2,long pulse3)

cardno　卡号

pulse1,pulse2,pulse3

X-Y-Z轴移动的相对距离

范围（-8388608~+8388607）

三轴插补只能为X、Y、Z轴

返回值　0：正确　　1：错误

注意：三轴插补的速度以X轴速度为基准，Z轴的倍率和驱动速度应设置成和X轴的倍率和驱动速度相同，Z轴的初始速度应设置成和X轴的**驱动速度**一样（不是X轴的初始速度）。Y无须设定。

5.9 四轴直线插补

int inp_move4(int cardno,long pulse1,long pulse2,long pulse3,long pulse4)

cardno　卡号

pulse1,pulse2,pulse3,pulse4

X-Y-Z-W轴移动的相对距离

范围（-8388608~+8388607）

返回值　0：正确　　1：错误

注意：四轴插补的速度以X轴速度为基准，Z轴的倍率和驱动速度应设置成和X轴的倍率和驱动速度相同，Z轴的初始速度应设置成和X轴的**驱动速度**一样（不是X轴的初始速度）。Y、W无须设定。

5.10 插补减速允许

int inp_dec_enable(int cardno,int no)

cardno　卡号

no　1：X-Y或X-Y-Z或X-Y-Z-W插补

2：Z-W插补

返回值　0：正确　1：错误

5.11 插补减速禁止

int inp_dec_disable(int cardno,int no)

cardno　卡号

no　1：X-Y或X-Y-Z或X-Y-Z-W插补

2：Z-W插补

返回值 0：正确　1：错误

此函数与上一函数用于加/减速驱动的插补驱动，对于单个插补驱动，应设为允许，对连续插补，应设为禁止，在最后一点再设为允许。具体应用可见后面的例子。

5.12 插补错误清除

int inp_clear(int cardno);

cardno　卡号

返回值　0：正确　1：错误

三轴、四轴插补在不正常停止时，Z、W轴会一直停在驱动状态，需要使用上述函数使其恢复到正常状态，不正常停止是指在驱动过程中使用了停止命令或产生了限位动作等情况。两轴插补不需要使用。

5.13 三轴螺旋线插补

int inp_arc_z_move(int cardno, long x,long y,long i,long j ,long z,int dir,int speed)

cardno　卡号

x,y　x y 圆弧的终点位置（相对于起点）

i,j　x y 圆弧的圆心点位置（相对于起点）

z　z 的移动距离

dir　0：顺时针圆弧 1：逆时针圆弧

speed运行速度

三轴螺旋线插补只能为X、Y、Z轴，匀速

返回值　0：正确　　1：错误

六、开关量输入输出类

6.1　读所有开关量输入状态

int read_di(int cardno,unsigned long *value)

cardno　　卡号

value　输入状态值的指针

value为四个字节的值

D0为低位　D31为高位

相应位为0为输入低电平　为1为输入高电平

D0：X LMT+	D1：X LMT-
D2：X STOP0	D3：X STOP1
D4：X STOP2	D5：X INPOS
D6：X ALARM	D7：X IN
D8：Y LMT+	D9：Y LMT-
D10：Y STOP0	D11：Y STOP1
D12：YSTOP2	D13：Y INPOS
D14：Y ALARM	D15：Y IN
D16：Z LMT+	D17：Z LMT-
D18：Z STOP0	D19：Z STOP1
D20：Z STOP2	D21：Z INPOS
D22：Z ALARM	D23：Z IN
D24：W LMT+	D25：W LMT-
D26：W STOP0	D27：W STOP1
D28：W STOP2	D29：W INPOS
D30：W ALARM	D31：W IN

返回值　　0：正确　　1：错误

6.2　设置所有开关量输出状态

int write_do(int cardno,unsigned long value)

cardno　　卡号

value　输出状态值

value为四个字节的值

D0为低位　D31为高位

相应位为0为输出低电平 为1为输出高电平

D0：OUT0

D1：OUT1

。。。

D31：OUT31

返回值　0：正确　　1：错误

6.3 读单个输入点

int read_bit(int cardno,int number)

cardno　卡号

number　输入点（0-31）

返回值　0：低电平

1：高电平

-1：错误

输入点数含义如下：

0：X LMT+	1：X LMT-
2：X STOP0	3：X STOP1
4：X STOP2	5：X INPOS
6：X ALARM	7：X IN
8：Y LMT+	9：Y LMT-
10：Y STOP0	11：Y STOP1
12：YSTOP2	13：Y INPOS
14：Y ALARM	15：Y IN
16：Z LMT+	17：Z LMT-
18：Z STOP0	19：Z STOP1
20：Z STOP2	21：Z INPOS
22：Z ALARM	23：Z IN

24：W LMT+	25：W LMT-
26：W STOP0	27：W STOP1
28：W STOP2	29：W INPOS
30：W ALARM	31：W IN

6.4 输出单点

int write_bit(int cardno,int number,int value)

cardno 卡号

number 输出点（0-31）

value 0：低 1：高

返回值 0：正确

1：错误

输出数对应相应的输出号。

七、中断类(仅用于DOS)

7.1 设置中断允许位

int set_int_enable(int cardno,int intflag);

cardno 卡号

intflag 0 禁止中断 1 允许中断

返回值 0：正确 1：错误

7.2 设置中断事件

设置使用的中断事件

int set_int_factor(int cardno,int axis,unsigned long int_factor);

cardno 卡号

axis 轴号

int-factor 中断事件

D0：未用

D1：逻辑/实位计数器的值大于COMP-寄存器的值

D2：逻辑/实位计数器的值小于COMP-寄存器的值

D3：逻辑/实位计数器的值小于COMP+寄存器的值

D4：逻辑/实位计数器的值大于COMP+寄存器的值

D5：加/减速驱动中，结束在定速区域输出脉冲

D6：加/减速驱动中，开始在定速区域输出脉冲

D7：驱动结束时

返回值　0：正确　　1：错误

7.3 获得中断信息

获得产生的中断

int get_int_status(int cardno,int axis,unsigned long *int_factor);

cardno　卡号

axis 轴号

int-factor 中断事件的指针

D0：未用

D1：逻辑/实位计数器的值大于COMP-寄存器的值

D2：逻辑/实位计数器的值小于COMP-寄存器的值

D3：逻辑/实位计数器的值小于COMP+寄存器的值

D4：逻辑/实位计数器的值大于COMP+寄存器的值

D5：加/减速驱动中，结束在定速区域输出脉冲

D6：加/减速驱动中，开始在定速区域输出脉冲

D7：驱动结束时

返回值　0：正确　　1：错误

第八章 运动控制开发编程示例

所有函数均为立即返回，而不是等待驱动结束后才返回，所有驱动操作都是由ADT850卡完成，上位机主要做实时监控及界面处理的工作。

☞ 开发DOS下的运动控制系统（Borland C++ 3.1）

DOS下的开发库共有三个文件，在配套光盘的\adt850\dos\lib\目录下，一个是头文件adt850.h，另外两个是库函数，一个是大模式库adt850l.lib（large memory model），一个是巨模式库adt850h.lib（huge memory model），可根据不同的编译环境选择。

a．单轴定量匀速运动

目的：

让X轴的步进电机以1000 pps的速度运动10000步：

程序如下：

```
#include “adt850.h”
void main()
```

```
{
    int cardno;
    cardno=adt850_initial();
    if(cardno<=0) return;              //未安装ADT850卡
                                       //以下只对第一块卡X轴操作
                                       //如果有多块卡，即cardno>1
                                       //可修改卡号，操作其他卡

    set_pulse_mode(0,1,1,0,0);         //设置X轴为脉冲+方向方式
    set_range(0,1,8000000);            //设置倍率为1
    set_startv(0,1,1000);
    set_speed(0,1,1000);               //如果起始速度大于或等于驱
                                       //动速度，则为匀速运动
    pmove(0,1,10000);                  //开始驱动
    int s;
    while(1)
    {
        get_status(0,1,&s);            //读驱动状态
        if(s==0)break;                 //驱动结束跳出
        ……                             //可执行读键盘，显示位置等函数
    }
    return ;
    }
```

b．单轴定量对称梯形加/减速运动

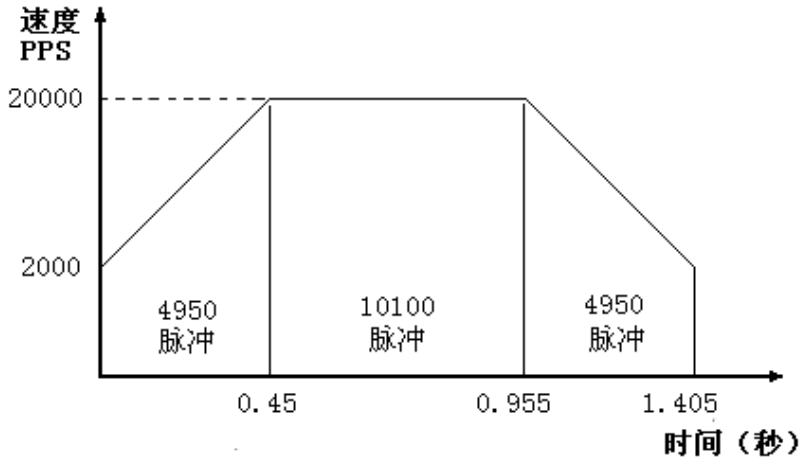
目的：

让X轴以下列速度运动20000步

起始速度：2000 pss

驱动速度：20000 pss

加/减速度：40000 pss



加速时间应为（20000-2000）/40000=0.45 秒

加速脉冲应为0.45*（20000+2000）/2=4950 个

减速与加速相同。

```
#include “adt850.h”
void main()
{
    int cardno;
    cardno=adt850_initial();
    if(cardno<=0) return;          //未安装ADT850卡
                                   //以下只对第一块卡X轴操作
                                   //如果有多块卡，即cardno>1
                                   //可修改卡号，操作其他卡

    set_pulse_mode(0,1,1,0,0);     //设置X轴为脉冲+方向方式
    set_ad_mode(0,1,0);            //设置为梯形加减速
    set_dec1_mode(0,1,0);          //对称加减速
    set_dec2_mode(0,1,0);          //采用自动减速
```

```
    set_range(0,1,1600000);          //设置倍率为5
    set_startv(0,1,400);             //起始速度   2000/5=400
    set_speed(0,1,4000);             //驱动速度   20000/5=4000
    set_acc(0,1,64);                 //加/减速度   40000/125/5=64
    pmove(0,1,20000);                //开始驱动
    int s;
    while(1)
    {
        get_status(0,1,&s);          //读驱动状态
        if(s==0)break;               //驱动结束跳出
        ……                           //可执行读键盘，显示位置等函数
    }
    return ;
}
```

c．单轴定量非对称梯形加/减速运动

目的：

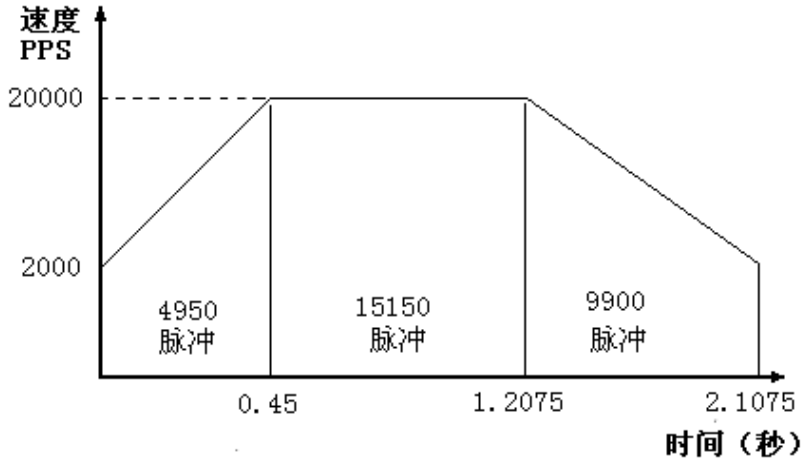
让X轴以下列速度运动30000步

起始速度：2000 pss

驱动速度：20000 pss

加速度：40000 pss

减速度：20000 pss



加速时间应为（20000-2000）/40000=0.45 秒

加速脉冲应为0.45*（20000+2000）/2=4950 个

减速时间应为（20000-2000）/20000=0.9 秒

减速脉冲应为0.9*（20000+2000）/2=9900 个

```
#include “adt850.h”
void main()
{
    int cardno;
    cardno=adt850_initial();
    if(cardno<=0) return;          //未安装ADT850卡
                                   //以下只对第一块卡X轴操作
                                   //如果有多块卡，即cardno>1
                                   //可修改卡号，操作其他卡

    set_pulse_mode(0,1,1,0,0);     //设置X轴为脉冲+方向方式

    set_ad_mode(0,1,0);            //设置为梯形加减速
```

```
    set_dec1_mode(0,1,1);          //非对称加减速
    set_dec2_mode(0,1,0);          //采用自动减速

    set_range(0,1,1600000);        //设置倍率为5
    set_startv(0,1,400);           //起始速度   2000/5=400
    set_speed(0,1,4000);           //驱动速度   20000/5=4000
    set_acc(0,1,64);               //加速度   40000/125/5=64
    set_dec(0,1,32);               //减速度   20000/125/5=32

    pmove(0,1,20000);              //开始驱动
    int s;
    while(1)
    {
        get_status(0,1,&s);        //读驱动状态
        if(s==0)break;             //驱动结束跳出
        ……                         //可执行读键盘，显示位置等函数
    }
    return ;
}
```

d. 单轴定量S曲线加/减速运动

例一：完全S曲线

目的：

让X轴以下列速度作完全S曲线加速运动20000步

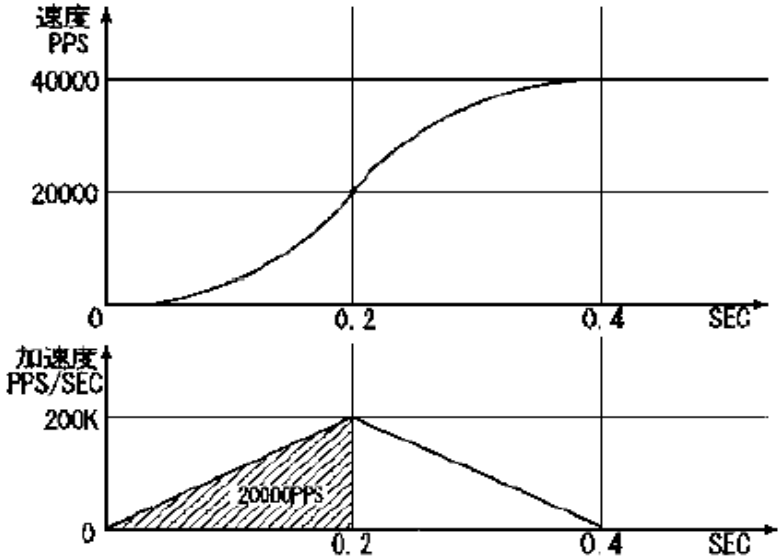
起始速度：1000 pss

驱动速度：40000 pss

加速时间：0.4秒

首先在计算上不考虑初始速度（把它作为0）。因为是完全S曲线加速，所以在0.4 秒的1/2（0.2 秒）把速度增加至40KPPS 的1/2（20KPPS）在剩下的0.2秒增加至40KPPS。这时加速度线性地增加直至0.2秒。这个积分数值

（斜线的面积-下图）等于增加速度20KPPS。因此0.2 秒的加速度是20000*2/0.2=200000 PPS/SEC，加速度的增加率是200000/0.2=1000000 PPS/SEC/SEC




```
#include “adt850.h”
void main()
{
    int cardno;
    cardno=adt850_initial();
    if(cardno<=0) return;          //未安装ADT850卡
                                   //以下只对第一块卡X轴操作
                                   //如果有多块卡，即cardno>1
                                   //可修改卡号，操作其他卡
    set_pulse_mode(0,1,1,0,0);     //设置X轴为脉冲+方向方式
```

```
    set_ad_mode(0,1,1);              //设置为S曲线加减速
    set_dec1_mode(0,1,0);            //对称加减速
    set_dec2_mode(0,1,0);            //采用自动减速

    set_range(0,1,800000);           //设置倍率为10
    set_startv(0,1,100);             //起始速度   1000/10=100
    set_speed(0,1,4000);             //驱动速度   40000/10=4000
    set_acc(0,1,160);                //加/减速度   200000/125/10=160
    set_acac(0,1,625);               //62500000*10/1000000=625
    pmove(0,1,20000);                //开始驱动
    int s;
    while(1)
    {
        get_status(0,1,&s);          //读驱动状态
        if(s==0)break;               //驱动结束跳出
        ……                           //可执行读键盘，显示位置等函数
    }
    return ;
}
```

例二：部分S曲线

目的：

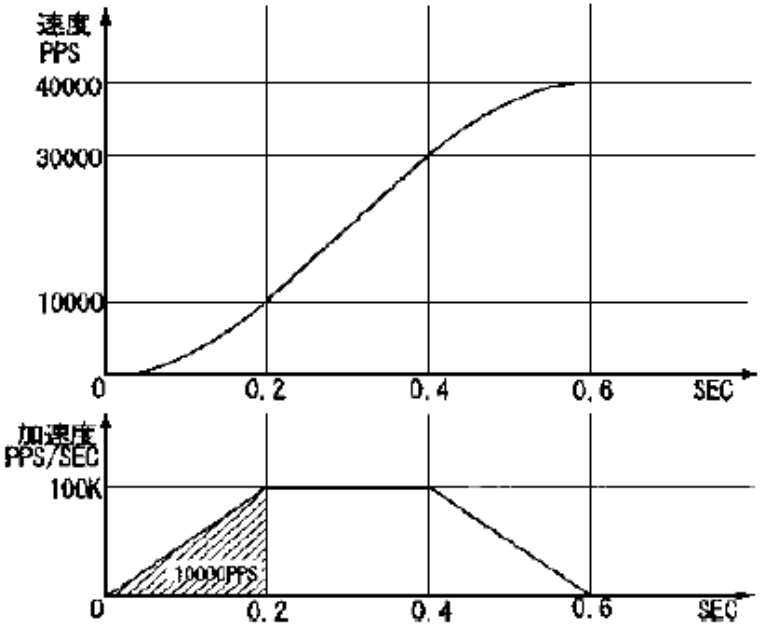
让X轴以下列速度作部分S曲线加速运动20000步

如下图所示用0.2秒抛物线地加速至10KPPS，再用0.2秒从10KPPS直线地加速至30KPPS，再用剩下的0.2秒从30KPPS抛物线地加速至40KPPS。

起始速度：1000 pss

驱动速度：40000 pss

加速时间：0.6秒



和上例一样，不用考虑初始速度，在0.2 秒内加速度线性增加至10KPPS，这时的积分数值上图的斜线面积，相当于起动速度10KPPS，因此0.2秒的加速度是10000*2/0.2=100000 PPS/SEC，加速度的增加率是100000/0.2=500000 PPS/SEC/SEC

程序如下：

```
#include “adt850.h”
void main()
{
    int cardno;
    cardno=adt850_initial();
    if(cardno<=0) return;              //未安装ADT850卡
                                       //以下只对第一块卡X轴操作
```

```
                                        //如果有多块卡，即cardno>1
                                        //可修改卡号，操作其他卡

set_pulse_mode(0,1,1,0,0);              //设置X轴为脉冲+方向方式

set_ad_mode(0,1,1);                     //设置为S曲线加减速
set_dec1_mode(0,1,0);                   //对称加减速
set_dec2_mode(0,1,0);                   //采用自动减速

set_range(0,1,800000);                  //设置倍率为10
set_startv(0,1,100);                    //起始速度    1000/10=100
set_speed(0,1,4000);                    //驱动速度    40000/10=4000
set_acc(0,1,80);                        //加/减速度    100000/125/10=80
set_acac(0,1,1250);                     //62500000*10/500000=1250
pmove(0,1,20000);                       //开始驱动
int s;
while(1)
{
    get_status(0,1,&s);                 //读驱动状态
    if(s==0)break;                      //驱动结束跳出
    ……                                  //可执行读键盘，显示位置等函数
}
return ;
}
```

e． 单轴连续驱动

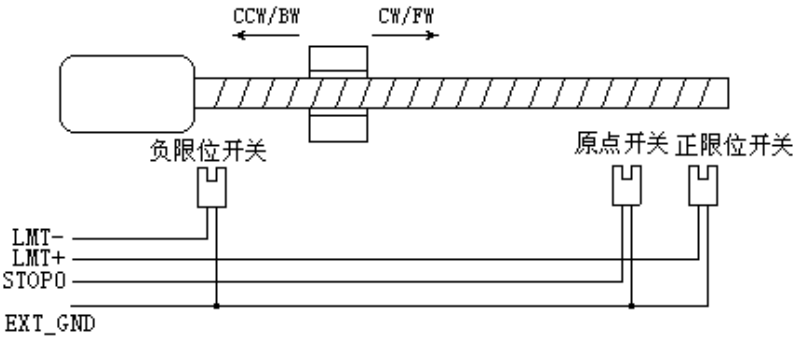
单轴的连续驱动与定量驱动的设置，在加/减速方面的设置是相同的，只是不需设定脉冲数，连续输出驱动脉冲，直至发出停止命令或外部的停止信号有效时才停止，在需要运行原点搜寻扫描，操作控制马达旋转速度时使用。

停止命令和外部的停止信号均有两种停止方式，一是立即停止，一是

减速停止，一般情况下，立即停止用于匀速驱动，减速停止用于高速驱动，外部的停止信号有两种，一是限位信号（LMT+，LMT-），限位通常是用来作为保护，一是停止信号（STOP），停止信号有3个，分别为STOP0、STOP1、STOP2，可分别设置停止有效/无效，在加减速驱动时为减速停止，在匀速驱动时为立即停止，因此结合连续驱动函数，可很简单的做出各种原点搜寻方式。

以下是一个匀速回原点的例子。

采用STOP0为原点信号

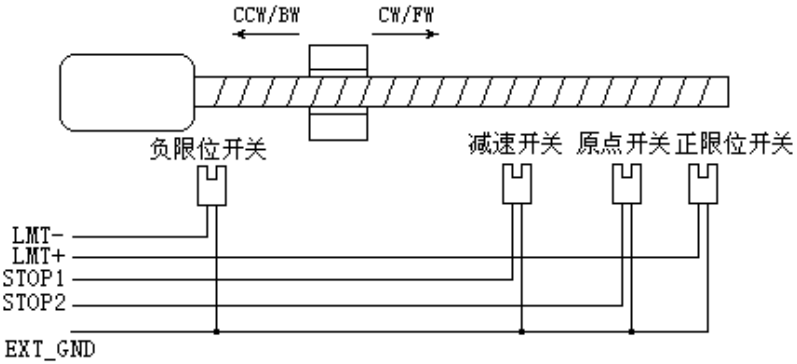



```
……        //初始部分与前相同，略
set_stop0_mode(0,1,1,0);              //STOP0有效，低电平停止
set_range(0,1,8000000);               //倍率为1
set_startv(0,1,200);                  //驱动速度为200
set_speed(0,1,200);
continue_move(0,1,0);                 //正方向移动,假设原点在正方向
int s;
while(1)
{
    get_status(0,1,&s);               //读驱动状态
    if(s==0)break;                    //驱动结束跳出
}
```

```
set_stop0_mode(0,1,0,0);        //STOP0无效，找到原点后应将//
                                停止信号取消，以免在正常//运行
                                时产生不必要的停止。
……                              //
```

以下是一个加/减速回原点的例子。

采用STOP1为减速信号，STOP2为原点信号




```
……                              //初始部分与前相同，略
set_stop1_mode(0,1,1,0);        //STOP1有效，低电平停止
set_ad_mode(0,1,0);             //设置为梯形加减速
set_dec1_mode(0,1,0);           //对称加减速
set_dec2_mode(0,1,0);           //采用自动减速
set_range(0,1,8000000);         //倍率为1
set_startv(0,1,200);            //初始速度为200
set_speed(0,1,2000);            //驱动速度为2000
set_acc(0,1,32);                //加/减速度   4000/125=32
continue_move(0,1,0);           //正方向移动,假设减速开关在正//方向
int s;
while(1)
{
```

```
        get_status(0,1,&s);          //读驱动状态
        if(s==0)break;               //驱动结束跳出
    }
    set_stop1_mode(0,1,0,0);         //STOP1无效
    set_stop2_mode(0,1,1,0);         //STOP2有效，低电平停止
    set_startv(0,1,200);             //初始速度为200
    set_speed(0,1,200);              //驱动速度为200
    continue_move(0,1,0);            //正方向移动,假设原点在正方向
    while(1)
    {
        get_status(0,1,&s);          //读驱动状态
        if(s==0)break;               //驱动结束跳出
    }
    set_stop2_mode(0,1,0,0);         //STOP2无效
    ……                               //
```

f. 手动减速

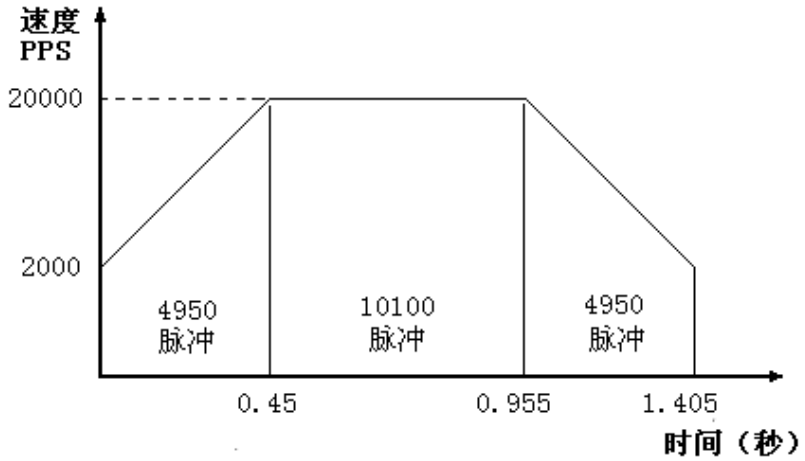
目的：

让X轴以下列速度运动20000步，采用手动减速方式

起始速度：2000 pss

驱动速度：20000 pss

加/减速度：40000 pss



加速时间应为（20000-2000）/40000=0.45 秒

加速脉冲应为0.45*（20000+2000）/2=4950 个

减速点应为20000-4950=15050。

```
#include “adt850.h”
void main()
{
    int cardno;
    cardno=adt850_initial();
    if(cardno<=0) return;          //未安装ADT850卡
                                   //以下只对第一块卡X轴操作
                                   //如果有多块卡，即cardno>1
                                   //可修改卡号，操作其他卡

    set_pulse_mode(0,1,1,0,0);     //设置X轴为脉冲+方向方式

    set_ad_mode(0,1,0);            //设置为梯形加减速
    set_dec1_mode(0,1,0);          //对称加减速
```

```
    set_dec2_mode(0,1,1);          //采用手动减速
    set_dec_pos(0,1,15050);        //设定减速点

    set_range(0,1,1600000);        //设置倍率为5
    set_startv(0,1,400);           //起始速度   2000/5=400
    set_speed(0,1,4000);           //驱动速度   20000/5=4000
    set_acc(0,1,64);               //加/减速度   40000/125/5=64
    pmove(0,1,20000);              //开始驱动
    int s;
    while(1)
    {
        get_status(0,1,&s);        //读驱动状态
        if(s==0)break;             //驱动结束跳出
        ……          //可执行读键盘，显示位置等函数
    }
    return ;
}
```

其他方式的手动减速与此类同，关键是减速点的计算。

g. 动态位置的显示与命令停止

采用例b的加/减速方式，但采用连续驱动，按”s”键立即停止，按”d”键减速停止。

```
#include <stdio.h>
#include <conio.h>
#include <dos.h>
#include “adt850.h”
void main()
{
```

```
int cardno;
cardno=adt850_initial();
if(cardno<=0) return;              //未安装ADT850卡
                                   //以下只对第一块卡X轴操作
                                   //如果有多块卡，即cardno>1
                                   //可修改卡号，操作其他卡
set_command_pos(0,1,0);            //X轴逻辑位置清零
set_actual_pos(0,1,0);             // X轴实际位置清零
set_pulse_mode(0,1,1,0,0);         //设置X轴为脉冲+方向方式

set_ad_mode(0,1,0);                //设置为梯形加减速
set_dec1_mode(0,1,0);              //对称加减速
set_dec2_mode(0,1,0);              //采用自动减速

set_range(0,1,1600000);            //设置倍率为5
set_startv(0,1,400);               //起始速度   2000/5=400
set_speed(0,1,4000);               //驱动速度   20000/5=4000
set_acc(0,1,64);                   //加/减速度   40000/125/5=64

continue_move(0,1,0);              //开始驱动
int s;
int key=-1;
long pos1,pos2;
while(1)
{
    get_status(0,1,&s);            //读驱动状态
    if(s==0)break;                 //驱动结束跳出
    get_command_pos(0,1,&pos1);
    get_actual_pos(0,1,&pos2);
    gotoxy(1,10);
```

```
            printf("command_pos=%8ld",pos1);
            gotoxy(1,12);
            printf("actual_pos=%8ld",pos2);
            if(kbhit())
                key=getch();
            else
                key=-1;
            if(key=='s')
                sudden_stop(0,1);
            if(key=='d')
                dec_stop(0,1);
        }
        return ;
    }
```

h． 多轴运动

以上虽为单轴操作，但实际上可同时设置另外几轴的数据，互相之间并不影响，如在X轴驱动时，设置好Y轴的参数，然后驱动Y轴，对X轴的运动不会有任何影响，如此可独立操作四轴，

下面是一个简单的例子，X轴以匀速（1000pps）运动1000步，Y轴以直线加减速运动300000步（起始速度10000pps，驱动速度200000pps，加减时间 0.2秒），Z轴以完全S曲线加速连续运动（起始速度 100 pps，驱动速度4000 pss，加速时间1.2秒），W轴以匀速（300000 pps）连续运行，按‘s’键停止。

首先计算Y轴的加速度，为（200000-10000）/0.2=950000

然后计算Z轴的加速度及加速度的变化率，加速度计算方法为：首先在0.6秒内加速至2000 pps，然后0.6秒内再加至4000 pps，则加速度为2000*2/0.6=6667 pps/sec，加速度的变化率为6667/0.6=11111 pps/sec/sec。

程序如下：

```
#include <stdio.h>
#include <conio.h>
#include <dos.h>
#include “adt850.h”
void main()
{
    int cardno;
    cardno=adt850_initial();
    if(cardno<=0) return;          //未安装ADT850卡
                                   //以下只对第一块卡X轴操作
                                   //如果有多块卡，即cardno>1
                                   //可修改卡号，操作其他卡
    set_pulse_mode(0,1,1,0,0);     //设置X轴为脉冲+方向方式
    set_pulse_mode(0,2,1,0,0);     //设置Y轴为脉冲+方向方式
    set_pulse_mode(0,3,1,0,0);     //设置Z轴为脉冲+方向方式
    set_pulse_mode(0,4,1,0,0);     //设置W轴为脉冲+方向方式

    //Y轴加减速设置
    set_ad_mode(0,2,0);            //设置为梯形加减速
    set_dec1_mode(0,2,0);          //对称加减速
    set_dec2_mode(0,2,0);          //采用自动减速

    //Z轴加减速设置
    set_ad_mode(0,3,1);            //设置为梯形加减速
    set_dec1_mode(0,3,0);          //对称加减速
    set_dec2_mode(0,3,0);          //采用自动减速

    //X轴
    set_range(0,1,8000000);        //设置倍率为1
```

```
set_startv(0,1,1000);          //起始速度   1000
set_speed(0,1,1000);           //驱动速度   1000

//Y轴
set_range(0,2,320000);         //设置倍率为8000000/320000=25
set_startv(0,2,400);           //起始速度  10000/25=400
set_speed(0,2,8000);           //驱动速度   200000/25=8000
set_acc(0,2,304);              //加速度   950000/125/25=304

//Z轴
set_range(0,3,8000000);
set_startv(0,3,100);
set_speed(0,3,4000);
set_acc(0,3,53);               // 6667/125=53.3
set_acac(0,3,5625);            // 62500000/11111=5625

//W轴
set_range(0,4,200000);         //倍率为40
set_startv(0,4,7500);          // 300000/40=7500
set_speed(0,4,7500);

pmove(0,1,1000);               //开始驱动
pmove(0,2,300000);
continue_move(0,3,0);
continue_move(0,4,0);

int s1,s2,s3,s4;
while(1)
{
    get_status(0,1,&s1);       //读X驱动状态
```

```
            get_status(0,2,&s2);          //读Y驱动状态
            get_status(0,3,&s3);          //读Z驱动状态
            get_status(0,4,&s4);          //读W驱动状态

            if(s1==0 && s2==0 && s3==0 && s4==0)break;
                                          //驱动结束跳出
            if(kbhit())
                key=getch();
            else
                key=-1;
            if(key=='s')
            {
                dec_stop(0,3);
                dec_stop(0,4);
            }
        }
        return ;
    }
```

i. 两轴直线插补（匀速）

插补速度是以X轴速度为基准，开始插补前只要设好X轴的参数即可。（Z-W插补以Z轴为基准）

下面是一个匀速直线插补的简单例子，圆弧插补与多轴直线插补的匀速驱动基本相同。

程序如下：

```
#include "adt850.h"
void main()
{
    int cardno;
    cardno=adt850_initial();
```

```
if(cardno<=0) return;              //未安装ADT850卡
                                   //以下只对第一块卡X轴操作
                                   //如果有多块卡，即cardno>1
                                   //可修改卡号，操作其他卡

set_pulse_mode(0,1,1,0,0);         //设置X轴为脉冲+方向方式
set_pulse_mode(0,2,1,0,0);         //设置Y轴为脉冲+方向方式
set_pulse_mode(0,3,1,0,0);         //设置Z轴为脉冲+方向方式
set_pulse_mode(0,4,1,0,0);         //设置W轴为脉冲+方向方式

set_range(0,1,8000000);            //设置X倍率为1
set_startv(0,1,1000);              //X起始速度   1000
set_speed(0,1,1000);                   //X驱动速度   1000

set_range(0,3,8000000);            //设置Z倍率为1
set_startv(0,3,2000);              //Z起始速度   2000
set_speed(0,3,2000);               //Z驱动速度   2000

inp_move2(0,1,10000,-20000);       //X-Y开始插补
                                   //X正向移动10000步
                                   //Y反向移动20000步
inp_move2(0,2,20000,-30000);       //Z-W开始插补
                                   //Z正向移动20000步
                                   //W反向移动30000步
int s1,s2;
while(1)
{
    get_inp_status(0,1,&s1);       //读X-Y插补状态
    get_inp_status(0,2,&s2);       //读Z-W插补状态
    if(s1==0 && s2==0)break;       //插补结束跳出
```

```
        ……                          //可执行读键盘，显示位置等函数
    }
    return ;
}
```

j． 两轴直线插补（加/减速）

两轴直线插补的加/减速驱动，只要将X轴（或Z轴）设置成直线加减速或S曲线加减速即可，注意，驱动开始之前须设成减速有效状态。

将上面的例子改成加减速驱动，X-Y直线加减速，Z-W曲线加减速。

程序如下：

```
#include “adt850.h”
void main()
{
    int cardno;
    cardno=adt850_initial();
    if(cardno<=0) return;           //未安装ADT850卡
                                    //以下只对第一块卡X轴操作
                                    //如果有多块卡，即cardno>1
                                    //可修改卡号，操作其他卡

    set_pulse_mode(0,1,1,0,0);      //设置X轴为脉冲+方向方式
    set_pulse_mode(0,2,1,0,0);      //设置Y轴为脉冲+方向方式
    set_pulse_mode(0,3,1,0,0);      //设置Z轴为脉冲+方向方式
    set_pulse_mode(0,4,1,0,0);      //设置W轴为脉冲+方向方式

    set_ad_mode(0,1,0);
    set_dec1_mode(0,1,0);
    set_dec2_mode(0,1,0);
    set_range(0,1,8000000);         //设置X倍率为1
    set_startv(0,1,1000);           //X起始速度   1000
```

```
    set_speed(0,1,8000);            //X驱动速度   8000
    set_acc(0,1,1000);

    set_ad_mode(0,3,1);
    set_dec1_mode(0,3,0);
    set_dec2_mode(0,3,0);
    set_range(0,3,4000000);         //设置Z倍率为2
    set_startv(0,3,1000);           //Z起始速度   2000
    set_speed(0,3,8000);            //Z驱动速度   16000
    set_acc(0,3,1000);
    ser_acac(0,3,1000);

    inp_move2(0,1,10000,-20000);    //X-Y开始插补
                                    //X正向移动10000步
                                    //Y反向移动20000步
    inp_move2(0,2,20000,-30000);    //Z-W开始插补
                                    //Z正向移动20000步
                                    //W反向移动30000步
    int s1,s2;
    while(1)
    {
        get_inp_status(0,1,&s1);    //读X-Y插补状态
        get_inp_status(0,2,&s2);    //读Z-W插补状态
        if(s1==0 && s2==0)break; //插补结束跳出
        ……                          //可执行读键盘，显示位置等函数
    }
    return ;
}
```

k．三轴直线插补（加减速）

三轴直线插补只能为X-Y-Z轴，以X轴的速度为基准，Z轴的倍率和驱动速度应设置成和X轴的倍率和驱动速度相同，Z轴的初始速度应设置成和X轴的**驱动速度**一样（不是X轴的初始速度）。Y无须设定。

下面是一个直线加减速的简单的例子，其余加减速方式只须改一下设置即可。

```
#include “adt850.h”
void main()
{
    int cardno;
    cardno=adt850_initial();
    if(cardno<=0) return;              //未安装ADT850卡
                                       //以下只对第一块卡X轴操作
                                       //如果有多块卡，即cardno>1
                                       //可修改卡号，操作其他卡

    set_pulse_mode(0,1,1,0,0);         //设置X轴为脉冲+方向方式
    set_pulse_mode(0,2,1,0,0);         //设置Y轴为脉冲+方向方式
    set_pulse_mode(0,3,1,0,0);         //设置Z轴为脉冲+方向方式
    set_pulse_mode(0,4,1,0,0);         //设置W轴为脉冲+方向方式

    set_ad_mode(0,1,0);
    set_dec1_mode(0,1,0);
    set_dec2_mode(0,1,0);
    set_range(0,1,8000000);            //设置X倍率为1
    set_range(0,3,8000000);            //Z倍率与X倍率相同
    set_startv(0,1,1000);              //X起始速度   1000
    set_speed(0,1,8000);               //X驱动速度   8000
    set_startv(0,3,8000);              //Z起始速度同X驱动速度
    set_speed(0,3,8000);               //Z驱动速度同X驱动速度
```

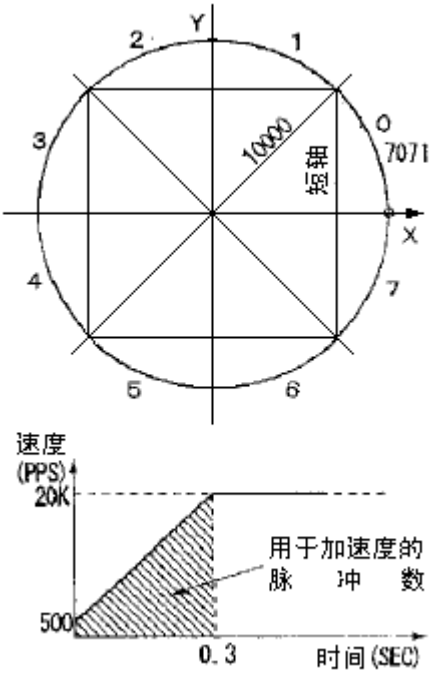
```
    set_acc(0,1,1000);

    inp_move3(0,5000,10000,-20000);    //X-Y-Z开始插补
                                       //X正向移动5000步
                                       //Y正向移动10000步
                                       //Z反向移动20000步
    int s1;
    while(1)
    {
        get_inp_status(0,1,&s1);       //读X-Y-Z插补状态
        if(s1==0 )break;               //插补结束跳出
        ……                             //可执行读键盘，显示位置等
函数
    }
    return ;
}
```

l. 两轴圆弧插补（加减速）

两轴圆弧插补一般用匀速驱动，但也可用加减速驱动，不过只能用手动减速的直线加减速驱动，不能使用S曲线加减速及自动减速。

匀速驱动较简单，只须设置X轴（或Z轴）的初始速度与驱动速度一样即可。

加减速驱动需要设置好减速点，下面以驱动一半径为10000的完整圆为例，说明驱动方法。



如上图所示，半径10000 的圆通过从0 至7 象限，在每个象限上短轴一直输出脉冲，所以短轴每1个象限输出10000/1.414=7071 脉冲,因此在整个圆上从主轴输出的基本脉冲数是7071*8=56568。

此外，把初始速度设定为500 pps,在0.3秒之内用直线加速把驱动速度增加至20000pps的话,加速度是(20000-500)/0.3=65000 pps/sec，加速时化费的脉冲数是上图的斜线部面积（500+20000）*0.3/2=3075，因此如果减速度和加速度一样，手动减速点就设定为56568-3075=53493。

如果做一个半圆，总脉冲为56568/2=28284，减速点为28284-3075=25209，其余相同。

程序如下：

```
#include “adt850.h”
void main()
```

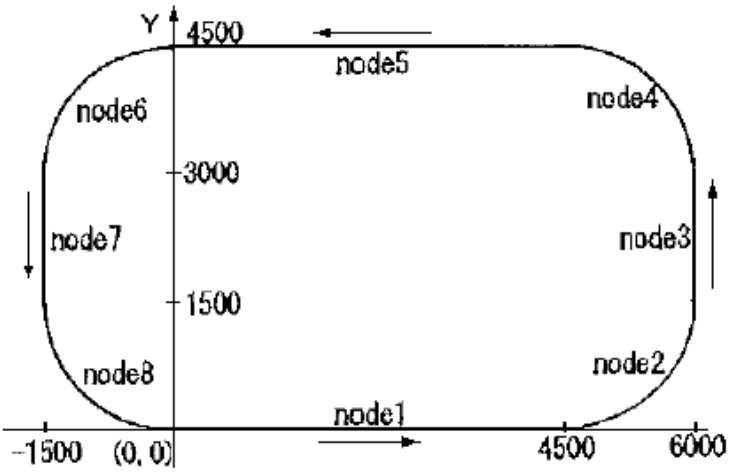
```
{
    int cardno;
    cardno=adt850_initial();
    if(cardno<=0) return;                    //未安装ADT850卡
                                             //以下只对第一块卡X轴操作
                                             //如果有多块卡，即cardno>1
                                             //可修改卡号，操作其他卡

    set_pulse_mode(0,1,1,0,0);               //设置X轴为脉冲+方向方式
    set_pulse_mode(0,2,1,0,0);               //设置Y轴为脉冲+方向方式
    set_pulse_mode(0,3,1,0,0);               //设置Z轴为脉冲+方向方式
    set_pulse_mode(0,4,1,0,0);               //设置W轴为脉冲+方向方式

    set_ad_mode(0,1,0);                      //直线加减速
    set_dec1_mode(0,1,0);                    //对称加减速
    set_dec2_mode(0,1,1);                    //手动减速
    set_range(0,1,1600000);                  //设置X倍率为5
    set_startv(0,1,100);                     //X起始速度    500/5
    set_speed(0,1,4000);                     //X驱动速度    20000/5
    set_acc(0,1,104);                        //X加速度      65000/125/5
    set_dec_pos(0,1,53493);                  //手动减速点

    set_ad_mode(0,3,0);                      //同上
    set_dec1_mode(0,3,0);
    set_dec2_mode(0,3,1);
    set_range(0,3,1600000);
    set_startv(0,3,100);
    set_speed(0,3,4000);
    set_acc(0,3,104);
    set_dec_pos(0,3,25209);                  //手动减速点
```

```
    inp_cw_arc(0,1,0,0,0,10000);          //X-Y顺时针圆弧插补
                                          //终点0，0 即画一个整圆
                                          //圆心位置0，10000
    inp_ccw_arc(0,2,0,20000,0,10000);     //Z-W逆时针插补
                                          //终点0，20000 即画一个半圆
                                          //圆心位置0，10000
    int s1,s2;
    while(1)
    {
        get_inp_status(0,1,&s1);          //读X-Y插补状态
        get_inp_status(0,2,&s2);          //读Z-W插补状态
        if(s1==0 && s2==0)break;          //插补结束跳出
        ……                                //可执行读键盘，显示位置等
函数
    }
    return ;
}
```

☞ 连续插补

匀速的连续插补与单个插补相似，只要在上一插补数据发出后，读取下一插补数据的允许写入状态，如果允许的话，写入下一插补数据，同时应检查驱动状态，如果驱动因错误而停止，应退出连续插补的循环。

加减速的连续插补则复杂一点，只能用手动减速的直线加减速驱动，不能用S曲线加减速驱动及自动减速。要事先设好减速点，先把减速设为无效，在最后一个节点上，将减速设为有效，开始至停止一定要在1个节点内运行减速，减速停止的最终插补节点输出的基本脉冲总数要大于在减速中化费的脉冲数。



连续插补轨迹的例子

以上图为例，起始速度设为500 pps，驱动速度为2000 pps，加速度为6000 pps/sec，加速时间为（2000-500）/6000=0.25秒，则加速所花费的脉冲为（2000+500）*0.25/2=313，上图最后一段为圆弧插补，半径为1500，四分之一圆弧，脉冲数可参考上例，应为1500/1.414*2=2121，则减速点应为2121-313=1808。

程序如下：

```
#include “adt850.h”
void main()
{
    int cardno;
    int s1,s2;
    cardno=adt850_initial();
    if(cardno<=0) return;          //未安装ADT850卡
                                   //以下只对第一块卡X轴操作
                                   //如果有多块卡，即cardno>1
                                   //可修改卡号，操作其他卡
```

```
set_pulse_mode(0,1,1,0,0);        //设置X轴为脉冲+方向方式
set_pulse_mode(0,2,1,0,0);        //设置Y轴为脉冲+方向方式

set_ad_mode(0,1,0);               //直线加减速
set_ded1_mode(0,1,0);             //对称加减速
set_dec2_mode(0,1,1);             //手动减速
set_range(0,1,8000000);           //设置X倍率为1
set_startv(0,1,500);              //X起始速度    500
set_speed(0,1,2000);              //X驱动速度    2000
set_acc(0,1,48);                  //X加速度      6000/125
set_dec_pos(0,1,1808);            //手动减速点

//第一段
inp_dec_disable(0,1);
inp_move2(0,1,4500,0);
while(1)
{
    get_stopdata(0,1,&s1);        //读错误信息
    if(s1!=0)goto err;            //有错误退出
    get_inp_status2(0,1,&s2);
    if(s2!=0)break;               //允许写
}

//第二段
inp_ccw_arc(0,1,1500,1500,0,1500);
while(1)
{
    get_stopdata(0,1,&s1);        //读错误信息
    if(s1!=0)goto err;            //有错误退出
```

```
        get_inp_status2(0,1,&s2);
        if(s2!=0)break;                 //允许写
    }

    //第三段
    inp_move2(0,1,0,1500);
    while(1)
    {
        get_stopdata(0,1,&s1);          //读错误信息
        if(s1!=0)goto err;              //有错误退出
        get_inp_status2(0,1,&s2);
        if(s2!=0)break;                 //允许写
    }

    //第四段
    inp_ccw_arc(0,1,-1500,1500,-1500,0);
    while(1)
    {
        get_stopdata(0,1,&s1);          //读错误信息
        if(s1!=0)goto err;              //有错误退出
        get_inp_status2(0,1,&s2);
        if(s2!=0)break;                 //允许写
    }

    //第五段
    inp_move2(0,1,-4500,0);
    while(1)
    {
        get_stopdata(0,1,&s1);          //读错误信息
        if(s1!=0)goto err;              //有错误退出
```

```
        get_inp_status2(0,1,&s2);
        if(s2!=0)break;                 //允许写
}

//第六段
inp_ccw_arc(0,1,-1500,-1500,0,-1500);
while(1)
{
        get_stopdata(0,1,&s1);          //读错误信息
        if(s1!=0)goto err;              //有错误退出
        get_inp_status2(0,1,&s2);
        if(s2!=0)break;                 //允许写
}

//第七段
inp_move2(0,1,0,-1500);
while(1)
{
        get_stopdata(0,1,&s1);          //读错误信息
        if(s1!=0)goto err;              //有错误退出
        get_inp_status2(0,1,&s2);
        if(s2!=0)break;                 //允许写
}
//第八段
inp_dec_enable(0,1);                    //允许减速
inp_ccw_arc(0,1,1500,-1500,1500,0);
while(1)
{
        get_stopdata(0,1,&s1);          //读错误信息
        if(s1!=0)goto err;              //有错误退出
```

```
            get_inp_status(0,1,&s2);
            if(s2==0)break;                 //插补结束
        }
        return ;
err:
        return ;
}
```

m．中断使用

中断一般仅在DOS下使用，因本卡所有函数均为立即返回，中断的使用不是必要的，大部分情况都不必使用中断。

☞ 用VB开发WINDOWS下的运动控制系统

VB下的开发库共有三个文件，在配套光盘的\adt850\win\vb\目录下，一个是动态库ADT850.DLL，一个是模块文件ADT850.BAS，一个是WindowsNT/2000使用的文件winio.sys，动态库中的函数已在模块文件ADT850.BAS中有声明，基本使用方法与DOS下的编程相同。

详细的例子在用户光盘上。

☞ 用VC开发WINDOWS下的运动控制系统

VC下的开发库共有三个文件，在配套光盘的\adt850\win\vc\目录下，一个是静态库ADT850.LIB，一个是头文件ADT850.H，一个是WindowsNT/2000使用的文件winio.sys，动态库中的函数已在头文件ADT850.H中有声明，基本使用方法与DOS下的编程相同。

详细的例子在用户光盘上。

第九章 运动控制开发要点

本卡在编程时常会遇到一些问题，其实，大部分问题是由于对本控制卡的原理不理解而产生的，以下就一些常见的、易产生误解的情况作一些说明。

一、卡的初始化

在程序的开始首先应调用adt850_initial()函数，确认adt850卡的安装是否正确，然后设置脉冲输出的模式，位置反馈的模式，限位开关的工作模式，伺服信号的使用与否，是否使用软件限位等等，以上参数应根据具体的机器来设置，一般只应在程序初始化时设置一次，以后不应再设置。

另外范围设定 set_range 函数一般应根据使用的最大脉冲频率来设定，此后一般不应再变化，每轴范围设定可以不同，如X轴最大频率为100K，因 set_speed 最大值为8000，倍率应为100000/8000=12.5，则范围R值应为8000000/12.5=640000，

即

set_range(cardno,1,640000);

当要输出100K频率时

set_speed(cardno,1,8000)

当要输出10K频率时

set_speed(cardno,1,800)

最低输出频率为1*12.5=12.5Hz

通常应根据可能使用的最大使用频率确定R值，除非最低频率不能满足使用要求，否则在使用过程中不应改变 R 值，

二、速度的设定

2.1 匀速运动

参数的设置很简单，只需要将驱动速度设置成小于或等于起始速度，其余的参数不用设置。

相关函数：

set_startv

set_speed

注意：函数使用的值应乘上倍率才为实际的速度。

2.2 对称直线加减速

这是最常用的一种方式，需设置起始速度、驱动速度，加速度，采用自动减速。

相关函数：

set_startv

set_speed

set_acc

set_ad_mode （设为直线加减速方式）

set_dec1_mode (设为对称方式)

set_dec2_mode （设为自动减速）

注意：加速度函数使用的值应乘上倍率再乘上125才为实际的加速度。

2.3 非对称直线加减速

此方式一般用于往垂直方向移动对象物时，加速时间与减速时间不相同，需比上面多设定减速度的值

相关函数：

set_startv

set_speed

set_acc

set_dec

set_ad_mode （设为直线加减速方式）

set_dec1_mode (设为非对称方式)

set_dec2_mode （设为自动减速）

2.4 S曲线加减速

对于一些负载较重的方式，为了达到较好的加速效果，采用S曲线加速，此时加加速度的值需设置，加加速度的计算对S曲线的形状有很大的影响，可参考前面的例子。

相关函数：

set_startv

set_speed

set_acc

set_acac

set_ad_mode　　（设为S曲线加减速方式）

set_dec2_mode　（设为自动减速）

2.5 手动减速

手动减速只用于自动减速不能正常使用的情况下，如圆弧插补的加减速驱动，连续插补的加减速驱动，此时要计算手动减速点，具体可参考前面的例子。

2.6 插补速度

关于插补的速度，虽然是设置X轴（或Z轴）的速度，但并不是代表X轴是以这个速度运动，而是长轴的速度，如：

在X-Y直线插补时，X移动距离大于Y移动距离时，X为长轴，X轴的驱动速度为设定的速度，X移动距离小于Y移动距离时，Y为长轴，Y轴的驱动速度为设定的速度。

在X-Y圆弧插补时，长轴的确定是以象限来确定，详细的说明可见前面的“插补”的说明。

对于两轴插补，只需设定X轴（或Z轴）的参数即可，

对于三轴、四轴插补，是设X轴的参数，同时Z轴的倍率应与X轴相同，Z轴的起始速度应与X轴的驱动速度（不是起始速度）相同，Z轴的驱动速度应与X轴的驱动速度相同，才能保证插补正确运行。

对于插补，一般使用匀速驱动，因此只要设定起始速度和驱动速度即可，加减速插补也可使用，只要设定好X轴参数即可，但要注意的是S曲线插补不能用于圆弧插补和连续插补。

另外插补减速允许和插补减速禁止函数的使用，默认情况是插补减速禁止，即如果使用加减速插补，在驱动时只有加速，而没有减速过程，此情况用于连续插补，如果只用单个插补，应在驱动前使用插补减速允许命令。

以上关于速度之所以设置较多，是为了能达到更好的性能，充分发挥本卡的功能，其实对于绝大部分应用，许多功能不会用上，通常只有要求达到很快的运行速度时，又要保证效果时才用到。此时的设置较复杂也是应该的。

三、插补错误清除函数的使用

对于三轴、四轴插补，如果没有正确走完，如在驱动过程中发出了停止命令，撞到了限位开关，软件限位有效等，即只要没有走完要求的距离，就产生了错误，应调用插补错误清除函数清除这个错误，否则Z、W轴的运动将回不正常。

对两轴插补则不用使用此函数。

四、STOP0、STOP1、STOP2信号

首先STOP0、STOP1、STOP2为每个轴都有的信号，因此共有12个STOP信号，此信号主要用于机器回原点时使用，回原点方式可根据情况使用一个或多个信号，但需注意的是此信号为减速停止，对于采用高速回零时应注意，可采用原点开关前加一减速开关，即采用两个STOP信号，一个作为原点开关，一个作为减速开关。也可只用一个信号，在碰到STOP信号后减速停止后，反向匀速运动再次碰到时停止。

另外STOP2有一特殊功能，即设置在有效时，如果用STOP2信号停止时，实际位置计数器会由STOP2清零，这是为了保证在原点位置处计数器同时为0，因其他的控制卡一般为在驱动停止后用软件将计数器置零，此时对于伺服驱动时，即使脉冲已停止发出，因伺服中有累积，仍会前进一点，造成位置有一点误差，而采用本卡上述方法回零，在回零后如果实际位置不为零，而是有一点数值，这是正常的，不用再调用函数将位置清零。

五、伺服信号

伺服到位信号和伺服报警信号必须在确认信号已接上时才能使用，如果未接伺服到位信号，而使伺服到位信号有效，将会出现驱动状态总是不会停止，这是因为此时是以到位信号作为驱动结束的标志。

其余的伺服信号，如伺服ON信号，报警清除信号可用通用输出信号驱动。